Laboratorio de Microcomputadoras - 86.07

Desarrollo de una interfaz de manejo y visualización de los datos de un contador Geiger-Müller

Profesor:	Ing. Guillermo Campiglio
Cuatrimestre/Año:	2º/2018
Turno de las clases prácticas	Miércoles
Jefe de trabajos prácticos:	Ing. Ricardo Arias
Docente guía:	Ing. Ricardo Arias

Autores				Seguimiento del proyecto								
Nombre	Apellido	Padrón	E-mail									
Juan Pablo	Goyret	99081	juanpablogoyret@gmail.com									
Federico	Nuñez Frau	98211	fedenunez32@gmail.com									
Gabriel	Vidal	97772	vidalgabriel93@gmail.com									

Observaciones:

.....
.....
.....
.....
.....
.....
.....
.....
.....

Fecha de aprobación				Firma J.T.P

Coloquio	
Nota final	
Firma profesor	

Índice

1. Descripción

El proyecto consistió en la elaboración de una interfaz de manejo y visualización de los datos de un contador Geiger-Müller, el cual es un circuito analógico generador de pulsos de tensión a partir de las partículas radioactivas que son recibidas en un tubo Geiger- Müller. Este último es un tubo de vidrio entre cuyos bornes se debe crear una tensión en torno a los cientos de volts y que, al recibir una partícula radioactiva, genera un pulso de corriente entre su ánodo y su cátodo, el cual es luego transformado a un pulso de tensión por un circuito secundario. Dado que las partículas arriban al tubo en instantes de tiempo aleatorios, los pulsos también se generan de este modo.

Estos pulsos de tensión son contados por un microcontrolador Atmega328p embebido en una placa Arduino, el cual realiza la tarea de determinar cuantos de estos han sido detectados en una unidad de tiempo dada, al igual que registrar los tiempos de llegada de cada pulso, para luego informarlo al usuario.

Como se observa en el diagrama de bloques de la Figura 1, los pulsos de tensión son comunicados al microcontrolador a través de opto-acopladores rápidos, para aislar eléctricamente al circuito analógico del digital (además, ambos se encuentran en placas separadas). Se cuenta además con una salida del microcontrolador opto-acoplada para encender la fuente de alta tensión que excita el tubo Geiger-Müller.

El microcontrolador informa la cantidad de pulsos de tensión recibidos por unidad de tiempo al usuario por medio de un panel LCD 2x16; así como también a una PC por medio de USB, empleando el sistema de comunicación UART y utilizando el protocolo SCPI (*Standard Commands for Programmable Instruments*) [1]. Este último medio es también utilizado para enviar los tiempos de llegada de cada pulso, permitiéndole al usuario realizar, por ejemplo (pero no contemplados en este trabajo), análisis estadísticos de la radiación recibida. Esta conexión permite, además, dar comienzo o detener mediciones a través de la PC.

Además, el sistema cuenta con un teclado conformado por 6 pulsadores: flechas en las 4 direcciones, un botón de cancelar y uno de aceptar; con las siguientes funciones:

- Controlar el largo del intervalo de tiempo en el cual se cuentan los pulsos antes de informar un valor al usuario o "ventana de tiempo". Este valor se modifica mediante un menú de configuración visible en el display LCD.
- Dar comienzo o detener las mediciones.
- Configurar un umbral de pulsos recibidos por unidad de tiempo, el cual, de ser superado, implica el encendido de un LED y/o la activación de una alarma.

El microcontrolador utilizado se encuentra embebido en una placa Arduino UNO Rev3. Sobre este se acopla mediante pines largos una placa aparte (utilizando el formato *poncho* o *shield*), el cual contiene conectores para dos cables dirigidos a 2 placas extra. Una de ellas posee el display LCD, el teclado, el LED y el buzzer correspondiente a la alarma sonora. Por otro lado, la otra placa cuenta con la fuente de alta tensión encargada de alimentar al tubo del contador, y comunicar los pulsos al Atmega328.

El circuito analógico será alimentado mediante un transformador de 220V a 6V, contando el bloque digital con una alimentación independiente.

En las Figuras 2 y 3 se muestran los diagramas de flujo del menú de control y del sistema contador de pulsos respectivamente.

[1] "SCPI 1999.0".Internet: http://www.ivifoundation.org/docs/scpi-99.pdf [Mayo, 1999]

2. Diagrama de bloques

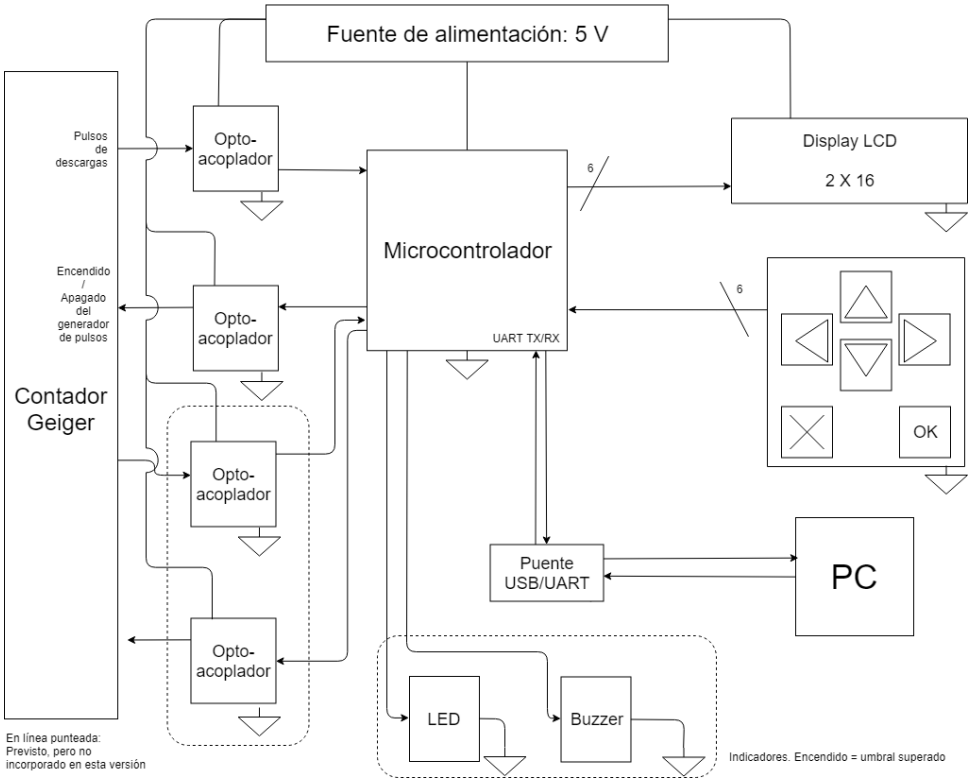


Figura 1: Diagrama de bloques.

3. Diagramas de flujo

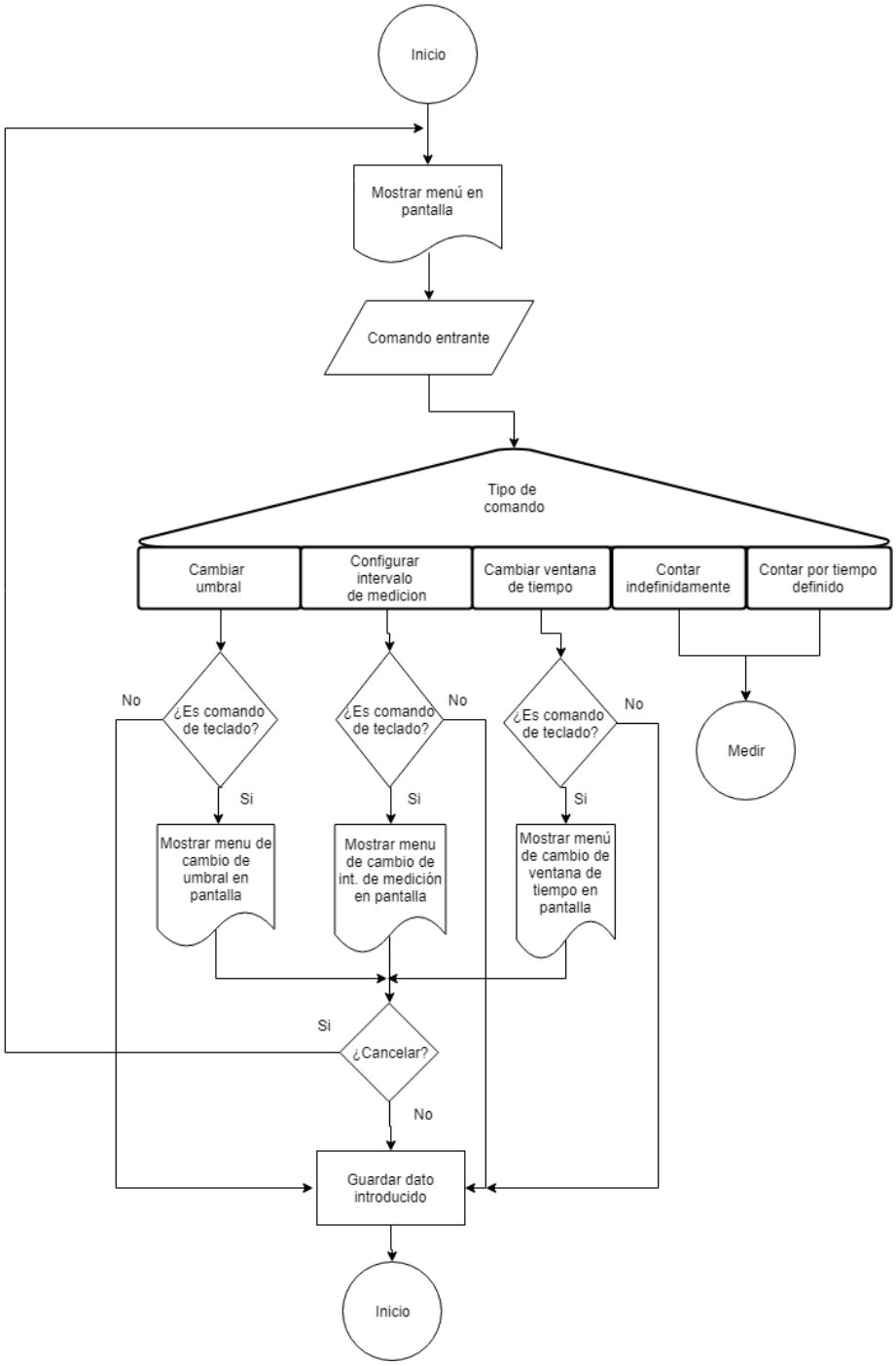


Figura 2: Diagrama de flujo del menú de control del contador.

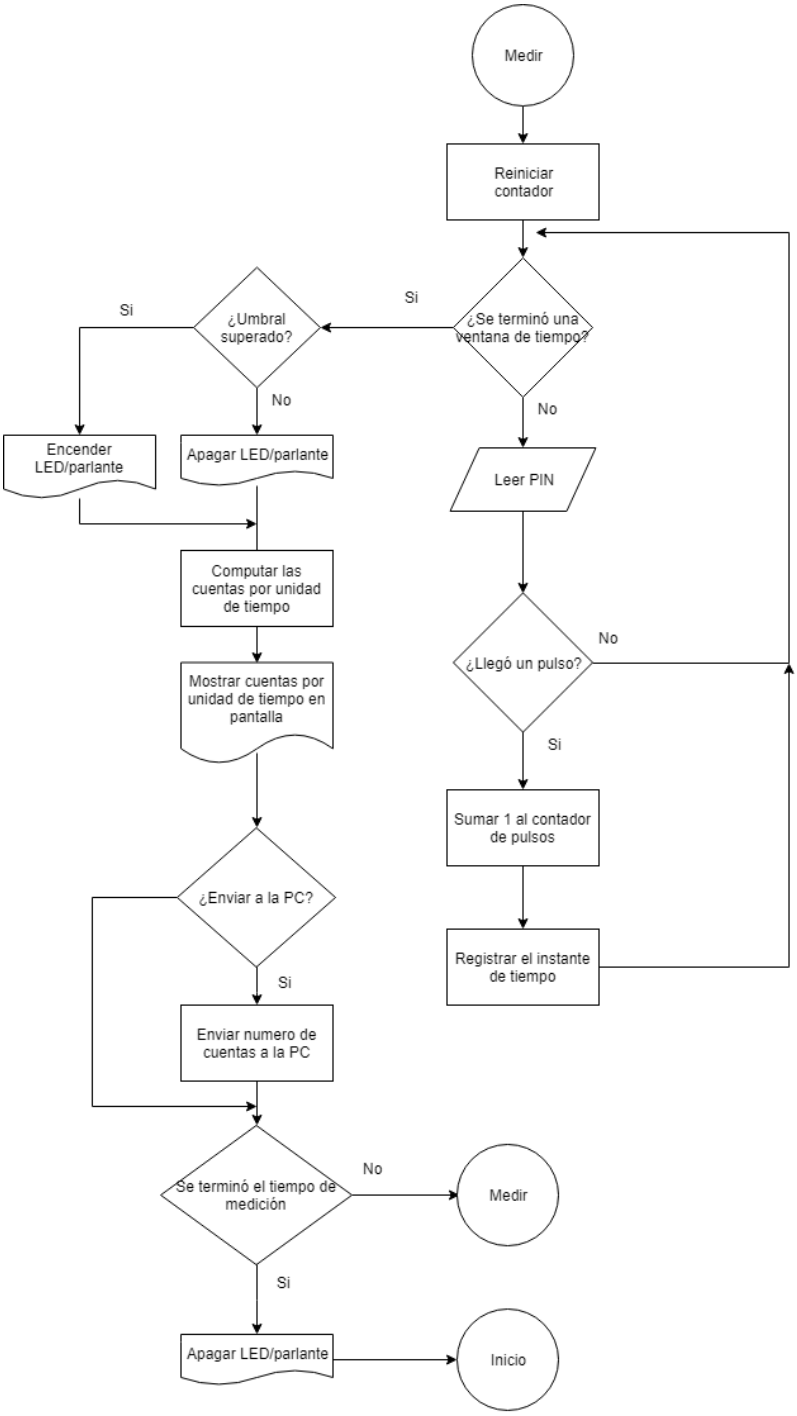
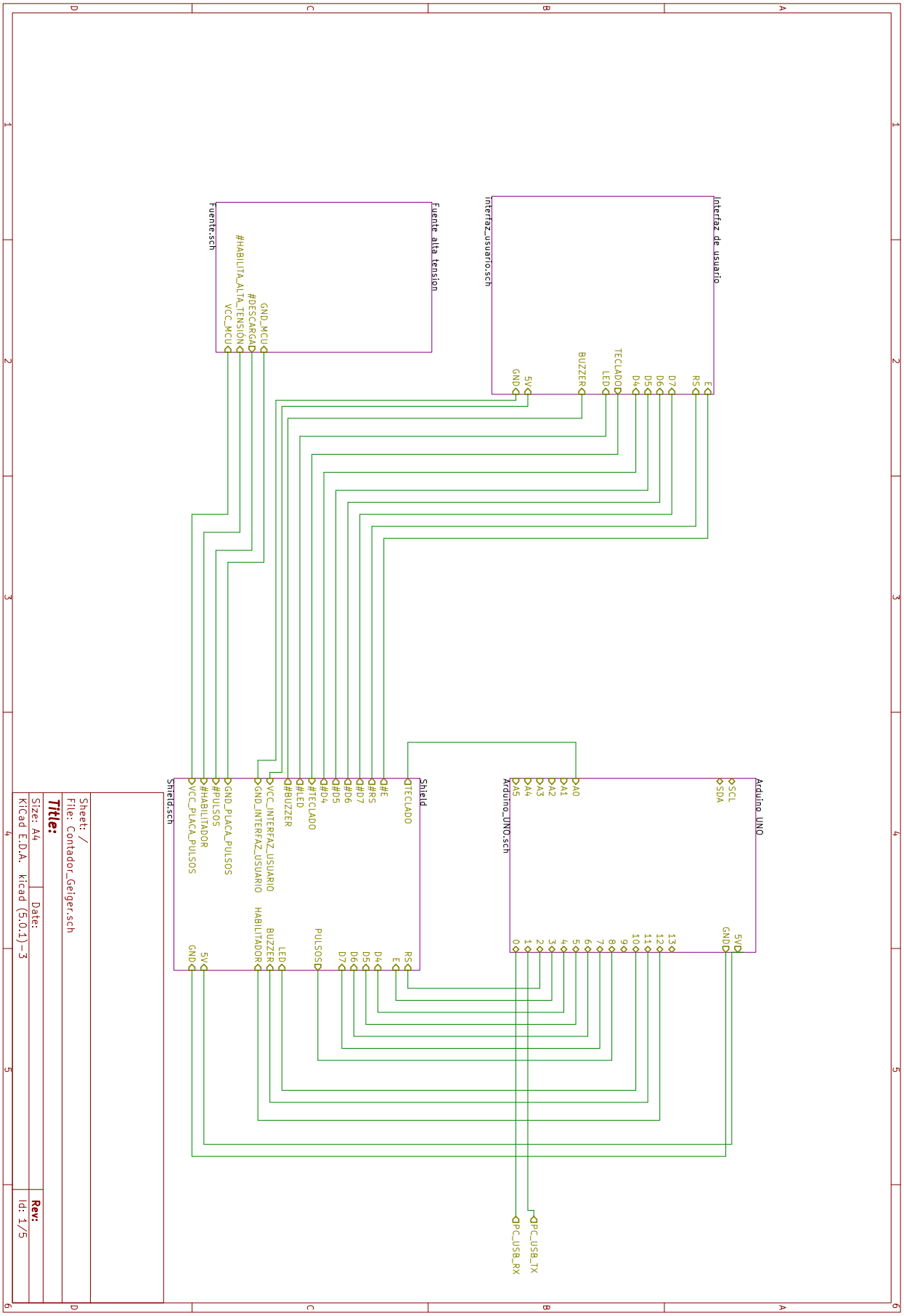
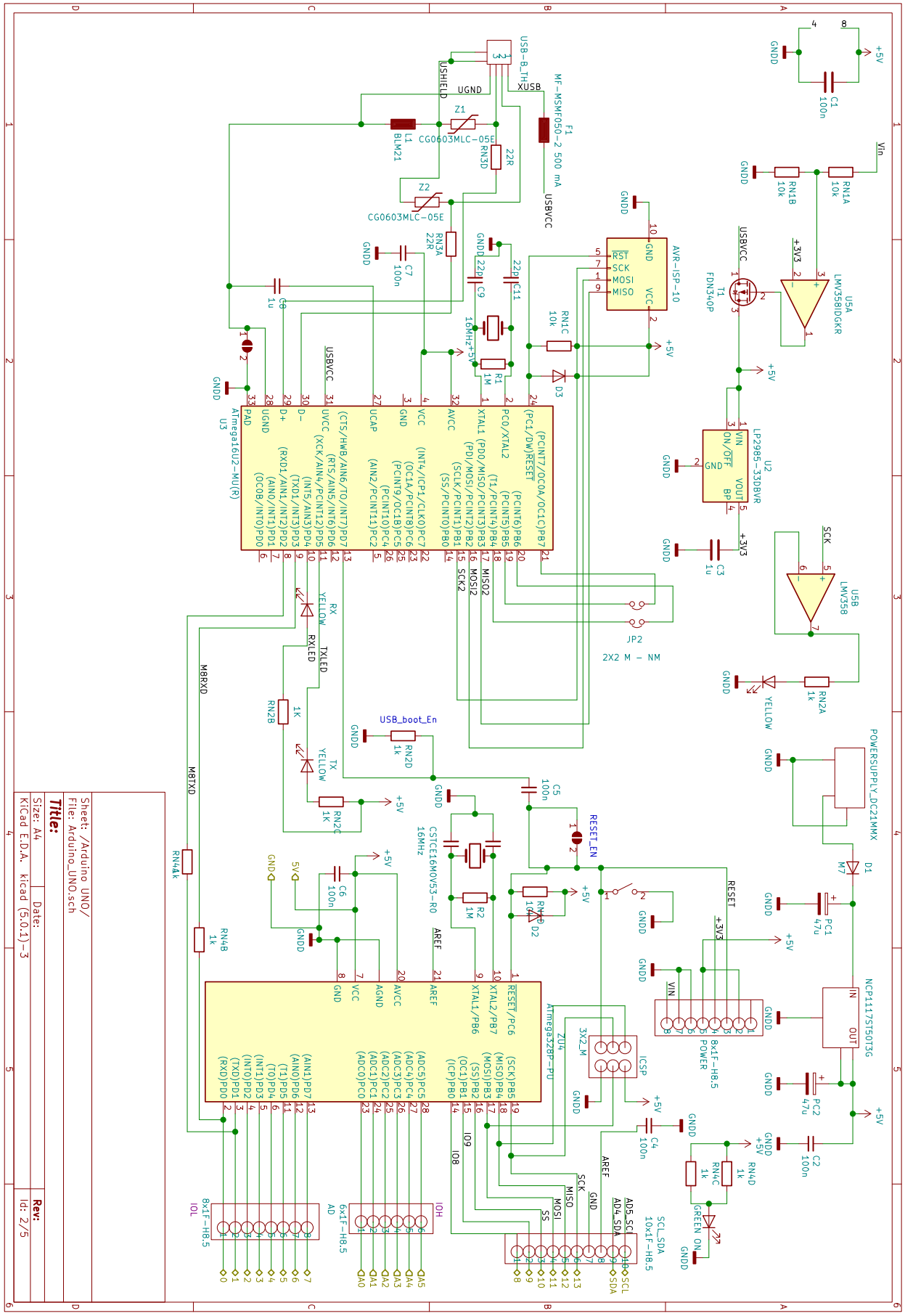


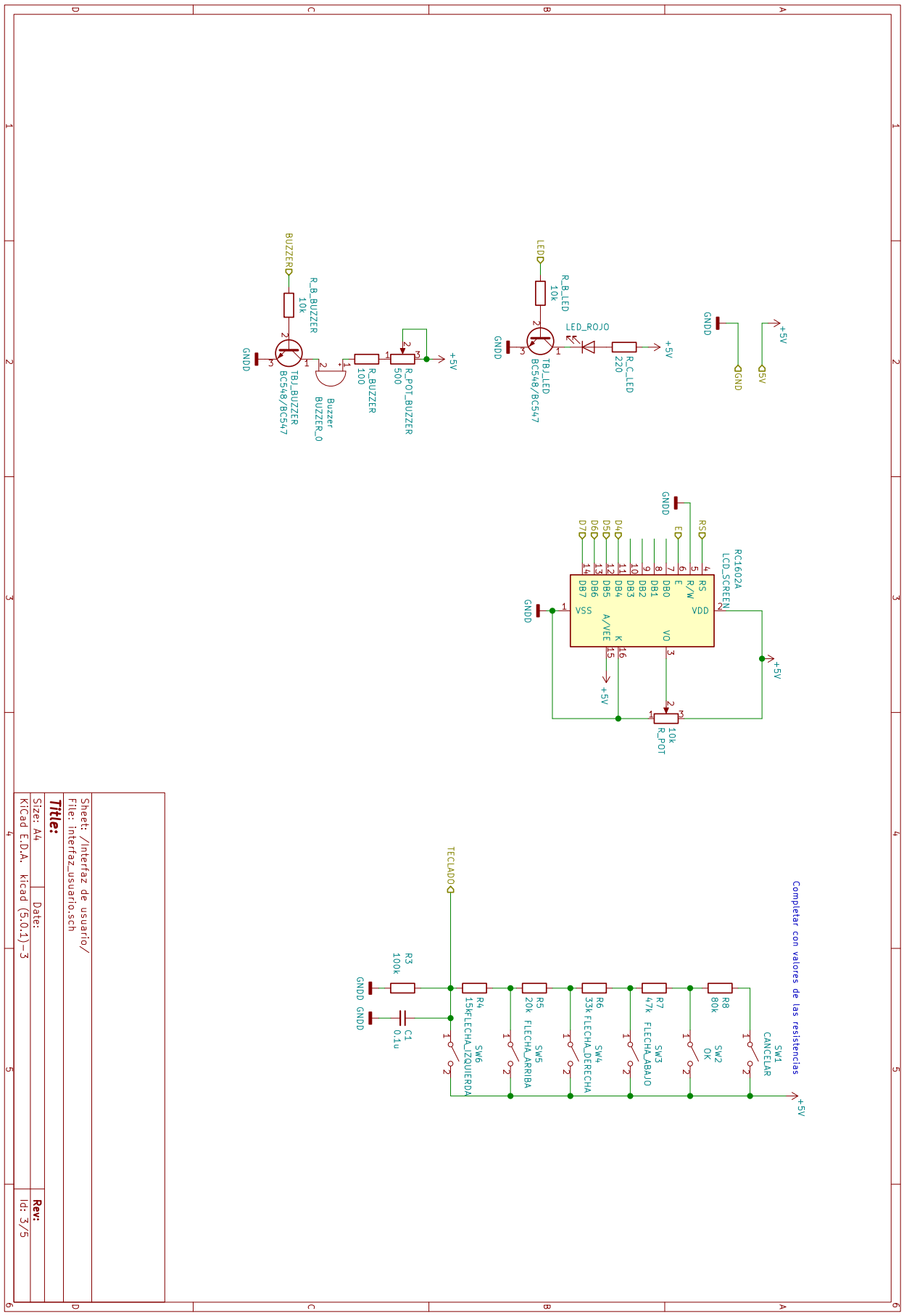
Figura 3: Diagrama de flujo del algoritmo de medición.

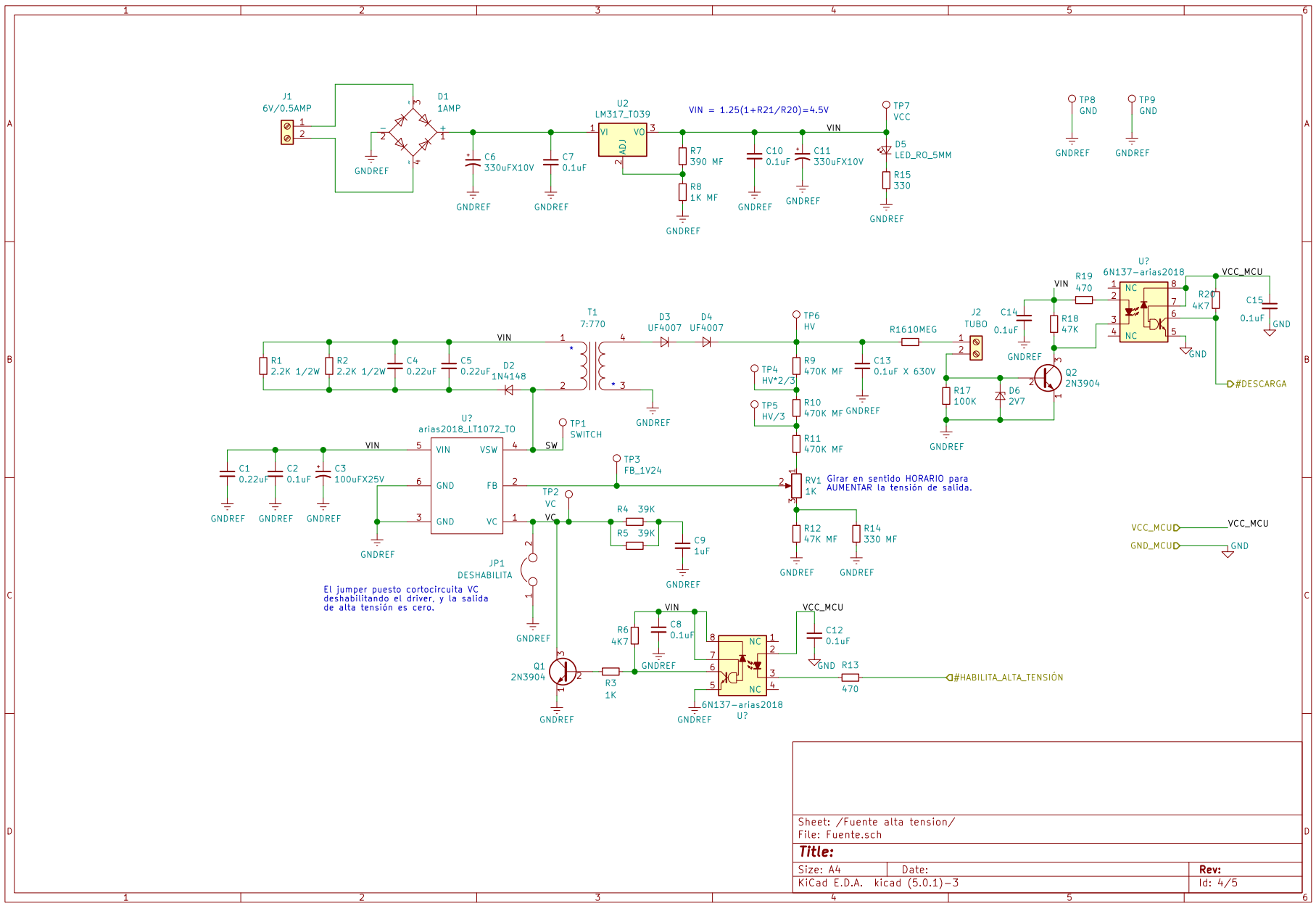
4. Esquemático

A continuación se incluye el esquemático del proyecto elaborado en *Kicad*. Este contempla el microcontrolador Atmega328p embebido en la placa Arduino, su *shield*, la placa del contador y la del panel LCD/teclado.

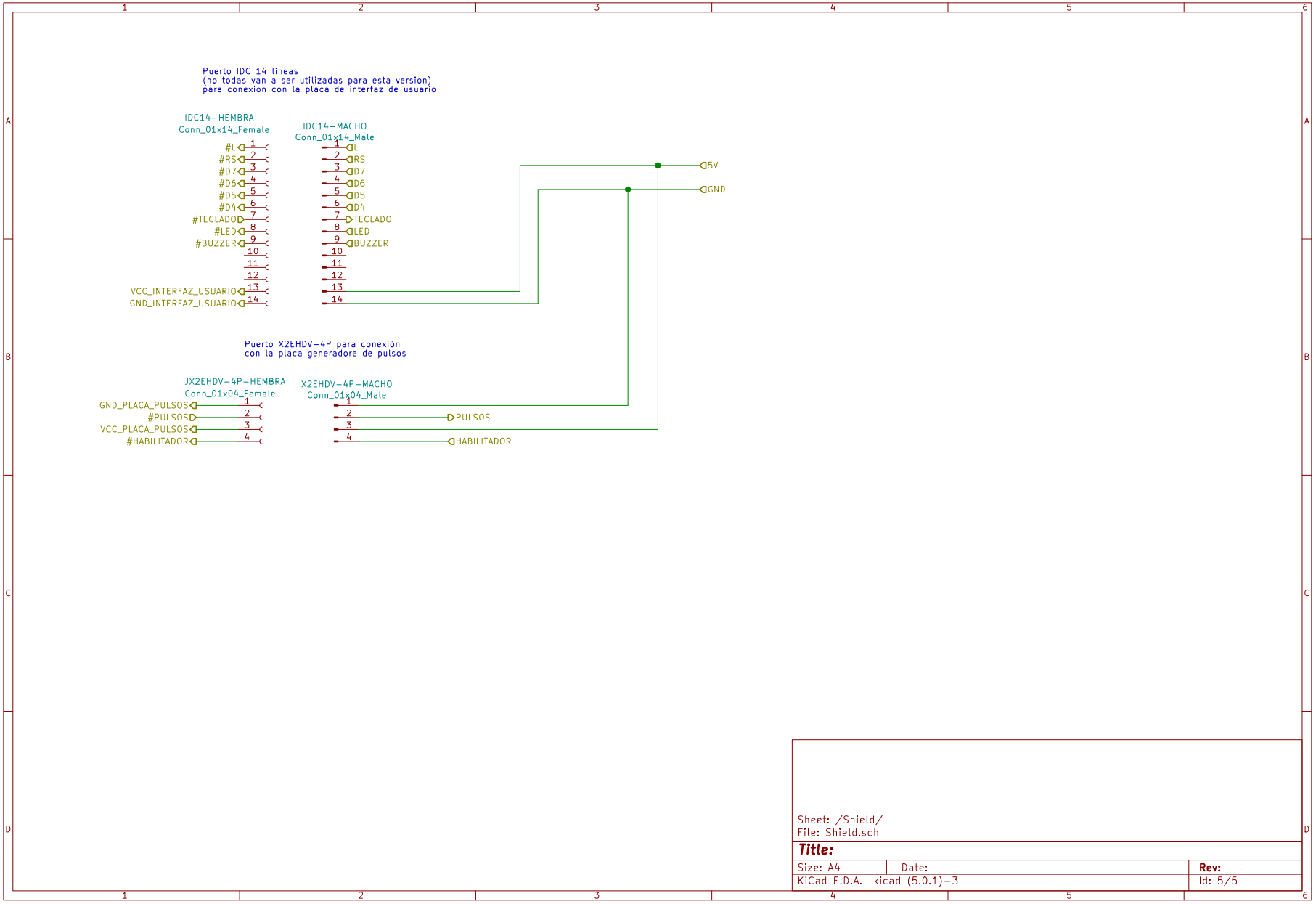

Fuente.sch
Fuente alta tension
GND_MCU
#DESCARGAR
#HABILITA_ALTA_TENSION
VCC_MCU
Interfaz_usuario.sch
Interfaz de usuario
5V
GND
BUZZER
LED
TECLADO
D4
D5
D6
D7
RS
E
Shield.sch
Shield
GND_PLACA_PULSOS
#PULSOS
#HABILITADOR
VCC_PLACA_PULSOS
VCC_INTERFAZ_USUARIO
GND_INTERFAZ_USUARIO
#BUZZER
#LED
#TECLADO
#D4
#D5
#D6
#D7
#RS
#E
TECLADO
HABILITADOR
PULSOS
Arduino_UNO.sch
Arduino UNO
A0
A1
A2
A3
A4
A5
SCL
SDA
5V
GND
0
1
2
3
4
5
6
7
8
9
10
11
12
13
PC_USB_TX
PC_USB_RX
Sheet: /
File: Contador_Geiger.sch
Title:
Size: A4
Date:
KiCad E.D.A. kicad (5.0.1)-3
Rev:
Id: 1/5


Sheet: /Arduino UNO/
File: Arduino_UNO.sch
Title:
Size: A4
Date:
KiCad E.D.A. kicad (5.0.1)-3
Rev:
Id: 2/5


Completar con valores de las resistencias
Sheet: /Interfaz de usuario/
File: interfaz_usuario.sch
Title:
Size: A4
Date:
KiCad E.D.A. kicad (5.0.1)-3
Rev:
Id: 3/5



Sheet: /Fuente alta tension/		
File: Fuente.sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad (5.0.1)-3		Id: 4/5


Puerto IDC 14 lineas
(no todas van a ser utilizadas para esta version)
para conexion con la placa de interfaz de usuario
IDC14-HEMBRA
Conn_01x14_Female
IDC14-MACHO
Conn_01x14_Male
#E
#RS
#D7
#D6
#D5
#D4
#TECLADO
#LED
#BUZZER
VCC_INTERFAZ_USUARIO
GND_INTERFAZ_USUARIO
E
RS
D7
D6
D5
D4
TECLADO
LED
BUZZER
5V
GND
Puerto X2EHDV-4P para conexión
con la placa generadora de pulsos
JX2EHDV-4P-HEMBRA
Conn_01x04_Female
X2EHDV-4P-MACHO
Conn_01x04_Male
GND_PLACA_PULSOS
#PULSOS
VCC_PLACA_PULSOS
#HABILITADOR
PULSOS
HABILITADOR
Sheet: /Shield/
File: Shield.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad (5.0.1)-3
Id: 5/5

5. Resultados

Luego del armado del dispositivo, se realizaron mediciones utilizando una fuente de radiación de cromo, la cual se encuentra en la Figura 4. El objetivo fue medir fotones residuales emitidos por dicha fuente que escapan de la cápsula una vez que la radiación principal ha sido filtrada por ella.

Para ello se colocó un tubo SI-8BG en la bornera correspondiente de la placa del contado Geiger. A continuación, se llevaron a cabo dos tipos de mediciones: una sin colocar el tubo cerca de la fuente (para medir radiación de fondo, como se ve en la Figura 5), y otra pegando el tubo en la pared de la fuente, como se aprecia en la Figura 6.

Durante el proceso de medición se utilizó un contador Geiger comercial GALERT 500 P de GammaSys a fines de comparar la respuesta de uno y otro dispositivo.

Figura 4: Fuente de radiación de cromo utilizada.

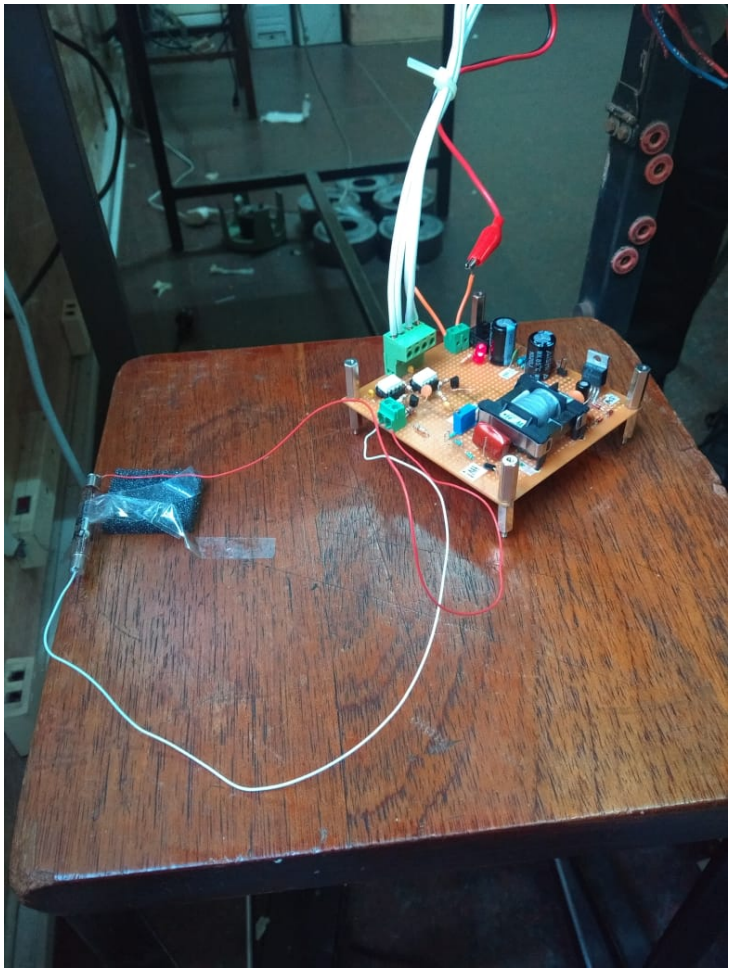


Figura 5: Foto de la placa del contador Geiger junto con el tubo SI-8BG, estando este alejando de la fuente de radiación. Este arregló se utilizó para medir radiación ambiente.

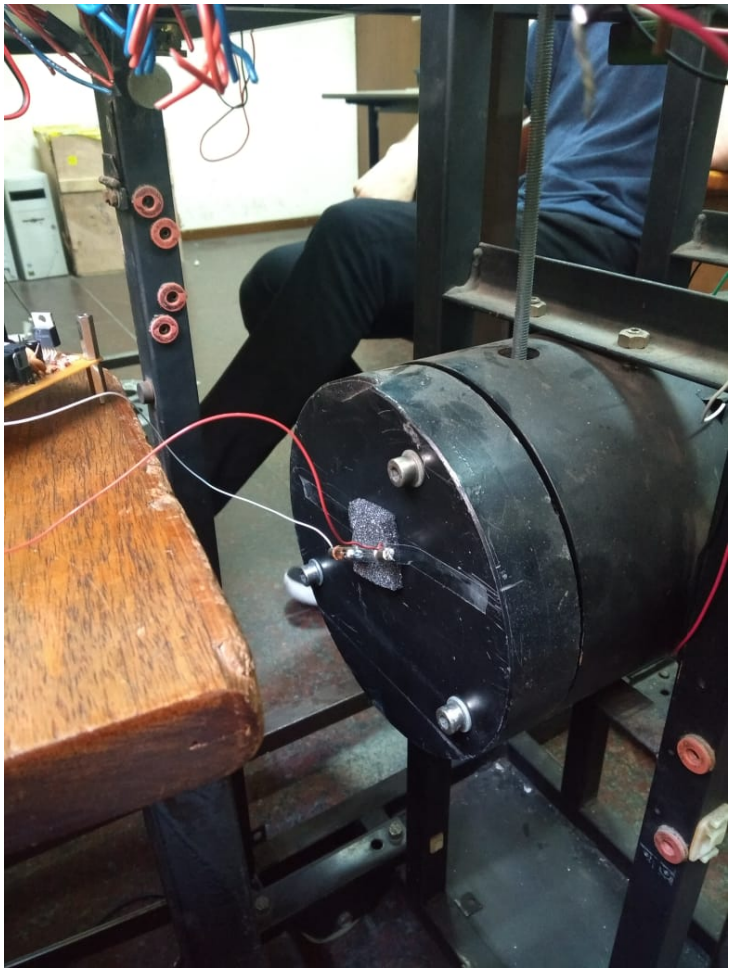


Figura 6: Foto de la placa del contador Geiger junto con el tubo SI-8BG, estando este pegado a la pared de la fuente de radiación.

Se hicieron pruebas con 3 tubos distintos del mismo modelo y variando las tensiones de alimentación entre los 430V y los 550V. Los dos primeros tubos mostraron resultados incongruentes. Uno de ellos reportaba constantemente valores de radiación nulos tanto estando pegado a la fuente de radiación, como también estando alejado de ella; mientras que las pruebas realizadas con el contador GALERT 500 P mostraban diferencias en ambos casos. Por otro lado, el segundo tubo derivaba siempre en un promedio de 100 cuentas por segundo tanto estando al lado de la fuente de radiación como fuera de ella.

Finalmente, el tercer tubo reportó mediciones nulas estando alejado de la fuente de radiación (resultado apropiado dada la baja sensibilidad del modelo de tubo utilizado), mientras que estando pegado a ella detectaba entre 100 y 200 cuentas por segundo en promedio, valor que coincidía aproximadamente con el reportado por el GALERT 500 P. Sin embargo, después de realizar varias mediciones, pasó a dar resultados nulos tanto estando junto a la fuente de radiación como fuera de ella. Esto último odría haberse debido a un problema en una soldadura del tubo o a un daño en el mismo durante su manipulación.

En las Figuras 7 y 8 se muestran las cuentas por segundo obtenidas al colocar el tubo lejos de la fuente y al situarlo pegado a ella respectivamente. En la Figura 7 se puede ver que la cantidad de cuentas por segundo era nula para toda medición, mientras que en la Figura 8 se aprecia una cantidad de pulsos por segundo distinta de cero. Finalmente, en la Figura 9, se muestra el promedio nulo visible en la pantalla LCD, correspondiente a la misma medición que la de la Figura 7.

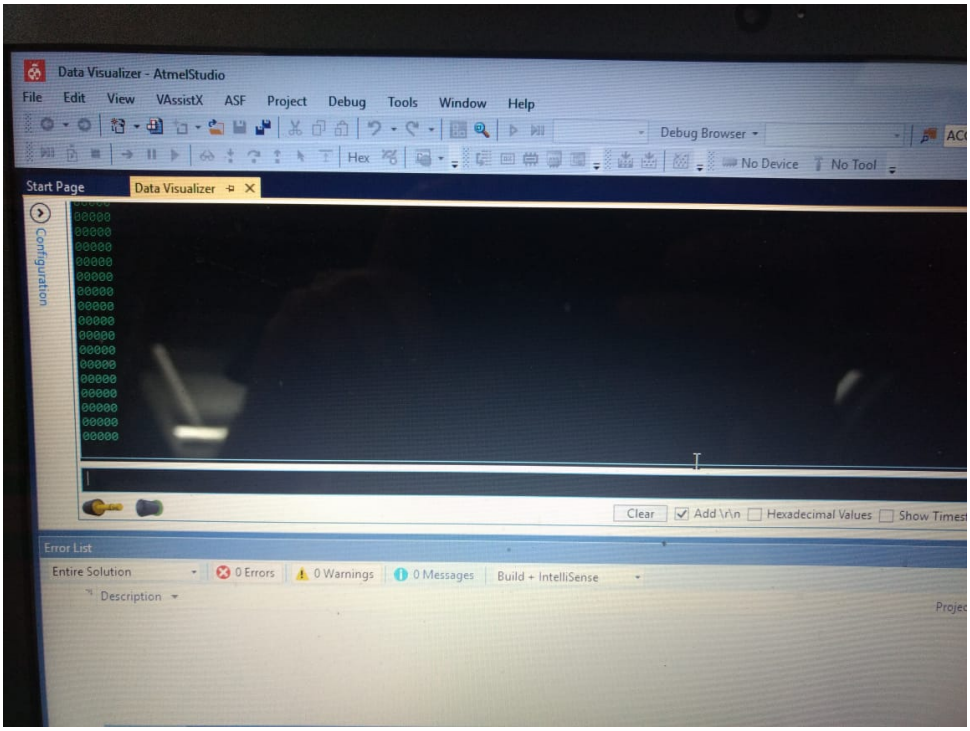


Figura 7: Foto de los datos reportados por el contador mediante UART para la medición de radiación ambiental.

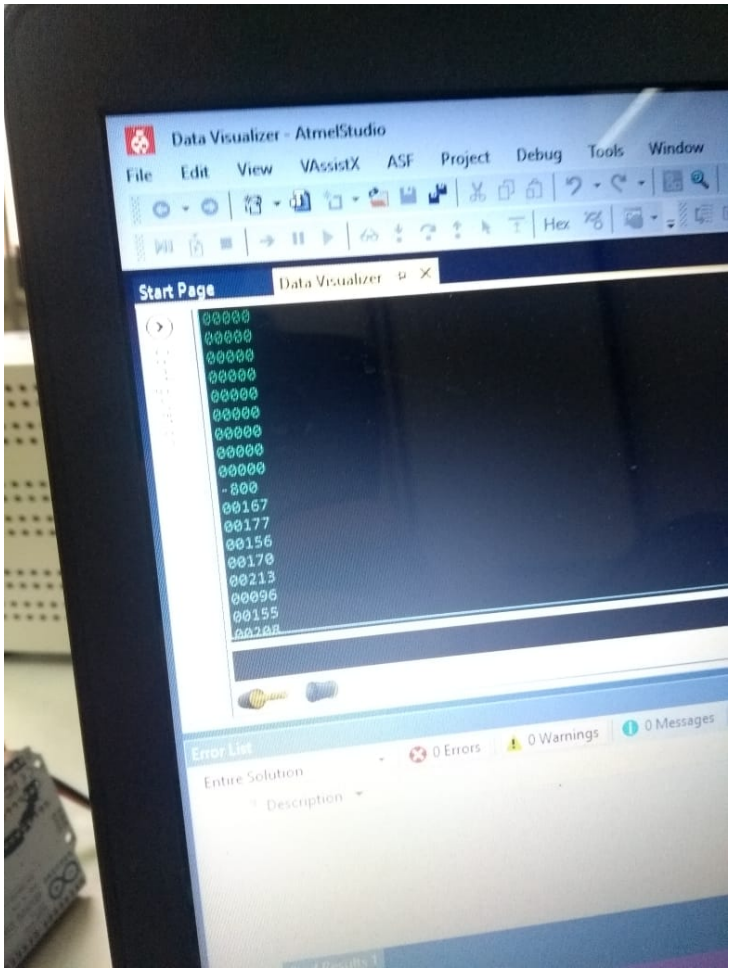


Figura 8: Comparación entre la cantidad de pulsos por segundo reportada por el contador para medición de radiación ambiente (los valores "00000"), y estando el tubo pegado a la pared de la fuente de radiación, que son los valores distintos de cero.

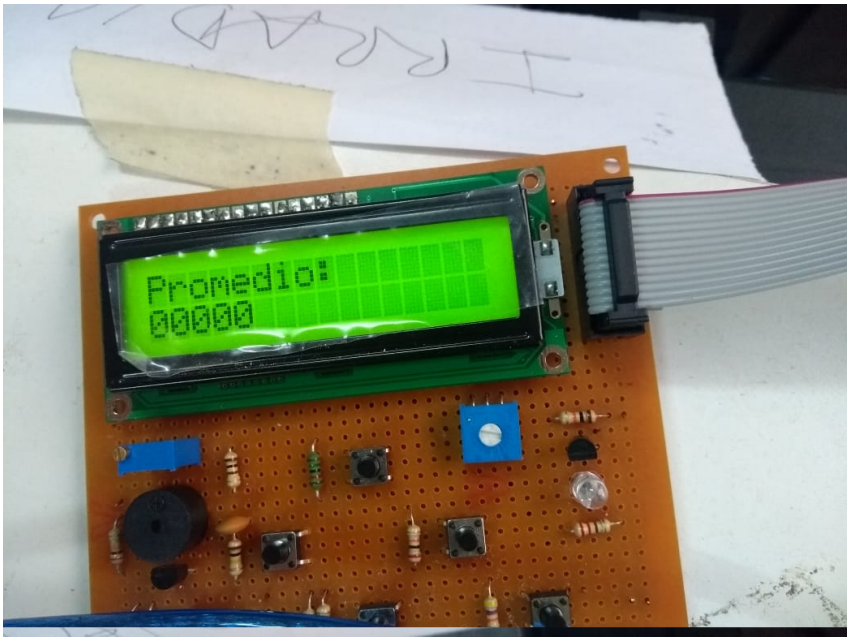


Figura 9: Foto del promedio reportado por el contador para la medición de radiación ambiental.

Finalmente, en la Figura 10 se muestra al Arduino conectado a la PC para recibir los datos por UART, al igual que a la pantalla LCD para mostrar el promedio. El cable blanco saliendo del puerto verde del *shield* se encuentra conectado a la fuente de alta tensión que alimenta al tubo, la cual se mostró en la Figura 5.

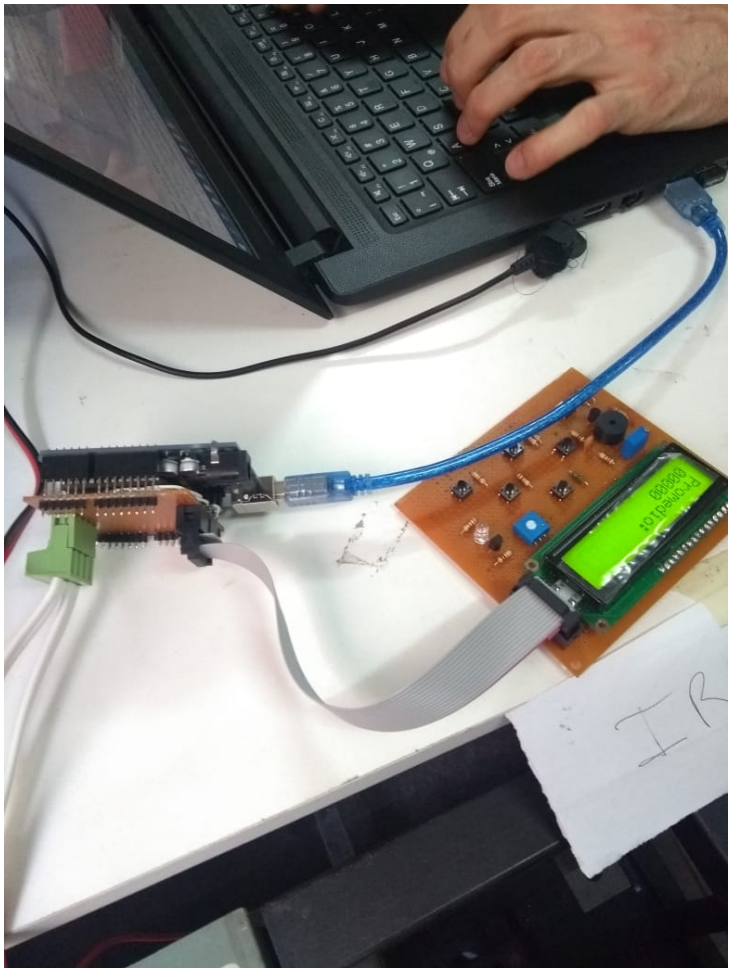


Figura 10: Foto del Arduino conectado a la pantalla LCD y a la PC durante la medición.

6. Apéndices

Apéndice I: mediciones

A continuación, en la Tabla 1 se muestran los resultados reportados por el contador por UART para distintas mediciones con el tubo colocado en la pared de la fuente de radiación. Se realizaron 12 mediciones de 10 segundos cada una. Cada dato se corresponde con la cantidad de pulsos detectados durante 1 segundo. Vale aclarar que estás mediciones fueron para un mismo tubo, que fue el tercero utilizado.

No se han incluido los resultados para mediciones con el tubo sin colocar en la pared de la fuente debido a que en ningún momento se detectaron pulsos para este caso.

Medición	$t = 1$	$t = 2$	$t = 3$	$t = 4$	$t = 5$	$t = 6$	$t = 7$	$t = 8$	$t = 9$	$t = 10$
1ra	00167	00177	00156	00170	00213	00096	00155	00208	00157	00106
2da	00093	00146	00171	00155	00146	00075	00124	00075	00157	00093
3ra	00135	00062	00081	00065	00151	00117	00149	00101	00118	00043
4ta	00052	00077	00047	00039	00116	00080	00115	00127	00084	00073
5ta	00084	00089	00082	00114	00103	00153	00077	00185	00111	00016
6ta	00068	00106	00097	00098	00044	00093	00149	00121	00099	00176
7ta	00105	00087	00117	00132	00117	00134	00056	00114	00127	00101
8va	00106	00047	00135	00055	00069	00032	00043	00087	00055	00099
9na	00254	00104	00084	00036	00078	00085	00107	00120	00082	00083
10ma	00152	00243	00241	00023	00246	00188	00206	00037	00249	00206
11ra	00200	00240	00186	00193	00047	00012	00226	00218	00228	00201
12da	00176	00247	00197	00178	00250	00224	00209	00225	00170	00185

Cuadro 1: Cuentas por segundo obtenidas durante 10 segundos para 12 mediciones distintas con el tubo pegado a la pared de la fuente de radiación.

Apéndice II: código fuente

main.asm

```
;
; Pruebas_UART.asm
;
; Created: 15/10/2018 21:53:12
; Author : Juan Pablo Goyret
; Descripcion: archivo para realizar pruebas sobre transmision UART

.include "m328pdef.inc"
.include "macros.inc"

; =============================================================
; ======================== Variables globales ==================
; =============================================================

; EVENTO: registro reservado para identificar eventos en el sistema que determinen
; acciones dentro de las maquinas de estados
; IMPORTANTE: la activacion de cada bit del registro simboliza la ocurrencia
; de un evento especifico. Si el bit esta en 0, entonces no ha ocurrido
; el evento
; Eventos asociados a cada bit:
;     0  Timeout recibiendo una cadena
;     1  Overflow de buffer recibiendo una cadena
;     2  Iniciar una lectura. Es equivalente a la recepcion de un comando READ? por UART
;     3  Abortar una lectura. Es equivalente a la recepcion de un comando ABORt por UART
;     4  Ha llegado un pulso
;     5  Se estan enviando datos por UART
;     6  Se ha superado el umbral de pulsos configurado
;     7  Se ha presionado una tecla
.def EVENTO = R25

; EVENTO2: registro en RAM reservado para identificar eventos en el sistema que determinen
; acciones dentro de las maquinas de estados
; IMPORTANTE: la activacion de cada bit del registro simboliza la ocurrencia
; de un evento especifico. Si el bit esta en 0, entonces no ha ocurrido
; el evento
; Eventos asociados a cada bit:
;     0  Se ha recibido un caracter
;     1  Se ha terminado de recibir una cadena
;     2  El sistema se encuentra ocupado durante un pedido externo
;     3  Indica la disminucion del registro que almacena el multiplicador de ventana
.dseg
EVENTO2: .BYTE 1
.cseg


; Constantes para identificar cada bit de EVENTO
.equ TIMEOUT_UART = 0
.equ OV_BUFFER_RX_UART = 1
.equ COMENZAR_MEDICION = 2
.equ DETENER_MEDICION = 3
.equ PULSO_RECIBIDO = 4
.equ ENVIANDO_DATOS_UART = 5
.equ UMBRAL_SUPERADO = 6
.equ TECLA_PRESIONADA = 7

; Constantes para identificar cada bit de EVENTO2
.equ BIT_CARACTER_RECIBIDO = 0
.equ BIT_FIN_CADENA = 1
.equ SISTEMA_OCUPADO = 2
.equ BIT_MUL_VENTANA_DEC = 3

; ESTADO: variable para identificar el estado de las maquinas de estados
```

```
; IMPORTANTE: cada bit, al estar activado, indica que la maquina debe estar
; en un estado especifico. Dos bits no pueden estar activos al mismo tiempo.
;
; Estados asociados a cada bit:
; --> De 0 a 3 inclusive con estados de la maquina de estados de la UART
;     0  Estado oscioso de la UART
;     1  Recibiendo una cadena
;     2  Intepretando una cadena
;     3  Se produjo un error en la recepcion de datos por UART
; --> De 4 a 7 inclusive son estados de la maquina de estados para realizar mediciones
;     4  Estado oscioso del sistema de mediciones
;     5  ACTUALMENTE SIN USO
;     6  Midiendo devolviendo los tiempos de llegada de cada pulso
;     7  Midiendo sin devolver los tiempos de llegada de cada pulso
.def ESTADO = R24

.def BOTONES_TECLADO = R16                          ; Se enciende determinaod bit,
   dependiendo del botón pulsado

; Constantes para identificar cada bit de ESTADO
.equ EST_OSCIOSO_UART = 0
.equ EST_RECIBIENDO_CADENA = 1
.equ EST_INTERPRETANDO_CADENA = 2
.equ EST_ERROR_UART = 3
.equ EST_OSCIOSO_MEDICION = 4
.equ EST_MEDIR_DEVOLVER_TIEMPOS = 6
.equ EST_MEDIR_DEVOLVER_TOTAL = 7

;MULTIPLICADOR PARA AMPLIAR EL OCR1A
.DEF MUL_DE_VENTANA = R15

; =====================================================================
; ========================= Codigo principal ==========================
; =====================================================================

.org 0
    JMP INICIO

; Interrupcion de buffer de RX lleno
.org URXCaddr
    JMP USART_RX

; Interrupcion de buffer de TX vacio
.org UDREaddr
    JMP USART_UDRE

; Interrupcion de overflow del timer0
.org OVF0addr
    JMP OVERFLOW_TIMER0

.org OVF2addr                           ;Interrupción por overflow del timer 2
    JMP LCD_INTERRUPCION_OVERFLOW

.org ICP1addr                           ;Interrupción por detección de flanco de en el ICP1
    JMP INTERRUPCION_PULSO_DETECTADO

.org OC1Aaddr                           ;Interrupción por comparación del TIMER 1
    JMP INTERRUPCION_FIN_VENTANA

.org ACIaddr
    JMP INTERRUPCION_COMPARADOR

.org ADCCaddr
        RJMP ADC_ISR

.org INT_VECTORS_SIZE
```

```
128
129 .include "teclado.asm"
130 .include "eeprom.asm"
131 .include "timer0.asm"
132 .include "uart.asm"
133 .include "scpi.asm"
134 .include "LCD.asm"
135 .include "TIMER_2.asm"
136 .include "TIMER_1.asm"
137 .include "Mediciones.asm"
138 .include "Uso_general.asm"
139 .include "configuracion.asm"
140 .include "maquina_estados_uart.asm"
141 .include "maquina_estados_mediciones.asm"
142 .include "maquina_estados_LCD.asm"
143 .include "pulsos_registro_desplazamiento.asm"
144 .include "comparador.asm"
145
146 INICIO:
147
148     ; No hay eventos por procesar al comenzar el programa
149     CLR EVENTO
150     STS EVENTO2, EVENTO
151
152     ; Hace que las maquinas de estado comiencen en estado oscioso
153     CLR ESTADO
154     SBR ESTADO, (1<<EST_OSCIOSO_MEDICION)
155     SBR ESTADO, (1<<EST_OSCIOSO_UART)
156
157     ; Inicializar el stack
158     LDI R16, LOW(RAMEND)
159     OUT SPL, R16
160     LDI R16, HIGH(RAMEND)
161     OUT SPH, R16
162
163     ; Borrar el registro de motivos del timer0
164     LDS R16, MOTIVO_TIMER0
165     CLR R16
166     STS MOTIVO_TIMER0, R16
167
168     ; Limpiar el contenido del registro de desplazamiento para enviar los tiempos de los
            pulsos
169     CLR CANT_CARACTERES_GUARDADOS
170
171     ; Cargar configuracion del dispositivo por default
172     CALL CARGAR_CONFIGURACION_EEPROM
173 ;   CALL CONFIGURAR_REGISTROS_DEFAULT
174
175
176     ; === Inicializar puertos ===
177     ; Puertos del LCD
178     CALL CONFIGURAR_PUERTOS_LCD
179
180     ; Configurar perifericos, timers, etc y sus interrupciones
181     CLI
182     CALL CONF_COMPARADOR
183     CALL CONF_ADC
184     ; Configurar UART
185     CALL INICIALIZAR_UART
186     CALL ACTIVAR_TX_RX
187     CALL ACTIVAR_INT_RX
188     CALL ACTIVAR_ISR_TIMER0_OV
189     CALL INIT_TIMER_2           ;Inicializo el TIMER 2, para las interrupciones del LCD
190     CALL INIT_TIMER_1           ;Inicializo el TIMER 1, para captar los pulsos de las
            mediciones
```

```
191     SEI                         ;Prendo interrupciones globales, para poder inicializar el
            LCD
192
193     CALL INIT_LCD               ;Inicializo el LCD
194
195     ; Enviar mensaje de prueba por UART
196     LDI R18, LOW(MENSAJE_TX)
197     LDI R19, HIGH(MENSAJE_TX)
198     CALL CARGAR_BUFFER
199
200     ; Activar interrupcion por buffer UDR0 vacio para
201     ; enviar el contenido del buffer
202     CALL ACTIVAR_INT_TX_UDRE0
203
204 /*  ; Mensaje de bienvenida
205     LDI ZL, LOW(MENSAJE_BIENVENIDA_LCD<<1)
206     LDI ZH, HIGH(MENSAJE_BIENVENIDA_LCD<<1)
207     CALL STRING_WRT_LCD_FLASH*/
208
209     ; Se configura para comenzar en el menú principal
210
211     LDI ESTADOS_LCD, VOLVER_MENU_PRINCIPAL                          ; Se habilita la
            escritura en ambos renglones del LCD, y se entra en el menú principal,
212                                                                     ; mostrando en el segundo
                                                                            renglón la opción de
                                                                        entrar al menú de
                                                                        umbral.
213
214
215 BUCLE_PRINCIPAL:
216
217     CALL VERIFICAR_APAGADO_TUBO
218
219     CLR R16
220     ; ====================================================================
221     ; Si se ha presionado una tecla, entonces identificarla
222
223     SBRC EVENTO, TECLA_PRESIONADA
224     CALL ADC_LEER_TECLA
225     ; LA TECLA PRESIONADA QUEDA ALMACENADA EN LOS REGISTROS R18 Y R19 PERO SOLO
226     ; DE FORMA TEMPORAL. LA MAQUINA DE ESTADOS DEL LCD DEBE SER COLOCADA INMEDIATAMENTE
227     ; DEBAJO DE CODIGO PORQUE SINO OTRA FUNCION PODRIA BORRAR LOS VALORES DE R18 Y R19
228
229
230
231     ; ====================================================================
232     ; MAQUINAS DE ESTADO
233     ; ====================================================================
234     ; ====================================================================
235     ; Maquina de estados dedicada a mostrar por pantalla al usuario, los menús
            correspondientes
236
237     CPI ESTADOS_LCD, MIDIENDO_LCD
238     BREQ SKIPEAR_MAQUINA_ESTADOS_LCD
239
240     CALL MAQUINA_ESTADOS_LCD
241
242     SKIPEAR_MAQUINA_ESTADOS_LCD:
243     ; ====================================================================
244     ; Maquina de estados dedicada a la gestion de las mediciones
245     CALL MAQUINA_ESTADOS_MEDICIONES
246
247     ; ====================================================================
248     ; Maquina de estados dedicada a la recepcion de una cadena por UART
249      CALL MAQUINA_ESTADOS_UART
250
```

```
251     ; ========================================================================
252
253     ; ========================================================================
254     ; FIN MAQUINAS DE ESTADO
255     ; ========================================================================
256
257     RJMP BUCLE_PRINCIPAL
258     RJMP END
259
260 END: RJMP END
```

maquina_estados_LCD.asm

```
1  ; Maquina de estados para mostrar los menús en el LCD
2
3  .DEF ESTADOS_LCD = R22
4  ; [7][6][5][4][3][2][1][0]
5  ; Bits 7 y 6: indican si se debe escribir o no en el lcd
6
7  ; Bit 5: Estado configurar duración de ventana
8
9  ; Bit 4: Indica si estoy en el Menú Principal o no
10 ; Bit 3: Estado configurar Buzzer/LED de umbral
11 ; Bit 2: Estado configurar largo de medición
12
13 ; Bit 1: Indica si estoy en el estado de medición o no
14
15 ; Bit 0: Estado cambiar umbral
16
17 ; Para la presentación del Menú principal, se va a mostrar como:
18
19 /*      ==================          ==================
20         = Menú principal =    -->   = Menú principal =
21         = <Conf Umbral>  =  <--     = <Conf Vent  >  =
22         ==================          ==================*/
23
24 ; Sucede que, dado que se cuenta con un teclado con flechas para recorrer los menús,
25 ; para respetar un orden de los submenús disponibles, se deben definir subestados dentro
26 ; del menú principal. Por ejemplo en el primer dibujo, si se presiona la tecla
27 ; flecha derecha, se cambia al segundo dibujo. Pero si luego se presiona la tecla
28 ; izquierda, se cambia de nuevo al primer dibujo. Son estados diferentes.
29 ; Este problema se encara de la siguiente forma:
30
31 ; Si se tiene en el registro ESTADOS_LCD = XX000001. Como el bit 4 está en 0, entonces
32 ; quiere decir que no estoy en el menú principal. Como el bit 0 está en 1, entonces estoy
33 ; en el menú cambiar umbral. En cambio si ESTADOS_LCD = XX010001. Ahora el bit 4 está
34 ; en 1, por lo que estoy en el menú principal y como el bit 0 está en 1, estoy mostrando
35 ; el mensaje del primer dibujo.
36
37 ; Entonces:
38 ; XX 01 0001 -> Menú principal, se muestra en el segundo renglón Configurar Umbral
39 ; XX 11 0000 -> Menú principal, se muestra en el segundo renglón Configurar Ventana
40 ; XX 01 0100 -> Menú principal, se muestra en el segundo renglón Configurar tiempo total de
     medición
41 ; XX 01 1000 -> Menú principal, se muestra en el segundo renglón Configurar o no el uso del
     Buzzer/LED de superación de umbral
42
43 ; XX 10 0001 -> Menú Configurar Ventana, se muestra en el esgundo renglón, 1ms
44 ; XX 10 0010 -> Menú Configurar Ventana, se muestra en el segundo renglón, 10ms
45 ; XX 10 0100 -> Menú Configurar Ventana, se muestra en el segundo renglón, 100ms
46 ; XX 10 1000 -> Menú Configurar Ventana, se muestra en el segundo renglón, 1000ms
47
48 ; XX 00 0001 -> Menú Configurar Umbral
49
50 ; XX 00 0100 -> Menú Configurar Tiempo Total de Medición
51
```

```
52 ; XX 00 1000 -> Menú Configurar uso o no del Buzzer, no
53 ; XX 00 1001 -> Menú Configurar uso o no del Buzzer, sí
54
55
56 ; XX 00 0010 -> Comenzar la medición
57
58 ; XX 00 0000 -> Se está midiendo, por lo que no se permite el acceso al menú LCD
59
60 ; 11 11 1111 -> Se terminó de medir
61
62 ;
    ********************************************************************************************
63
64 .EQU ESCRIBIR_PRIMER_RENGLON = 7                    ; Bit que indica que se puede escribir en
     el primer renglón: 1=sí, 0=no
65 .EQU ESCRIBIR_SEGUNDO_RENGLON = 6                   ; Bit que indica que se puede escribir en
     el segundo renglón: 1=sí, 0=no
66
67 .EQU CONF_VENTANA_LCD = 5                          ; 1 = estoy en el menú de configurar la
    ventana
68 .EQU MENU_PRINCIPAL_LCD = 4                        ; 1 = estoy en el menú principal, 0 = no
    estoy en el menú principal
69
70 .EQU CONF_BUZZER_LCD = 3                           ; 1 = estoy en el menu de configurar el
    buzzer, 0 = no estoy ahí
71 .EQU CONF_TIEMPO_TOTAL_LCD = 2                    ; 1 = estoy en el menú de configurar el
    tiempo de medición
72
73 .EQU COMENZAR_MEDICION_LCD = 1                     ; 1 = estado midiendo, 0 = no midiendo
74
75 .EQU CONF_UMBRAL_LCD = 0                           ; 1 = estoy en el menú de configurar el
    umbral
76
77 ;
    ********************************************************************************************
78 ; Bits para el menú de ventana
79
80 .EQU VENTANA_LCD_1ms = 0
81 .EQU VENTANA_LCD_10ms = 1
82 .EQU VENTANA_LCD_100ms = 2
83 .EQU VENTANA_LCD_1000ms = 3
84 ;
    ********************************************************************************************
85 ; Bits para el menú de Buzzer/LED
86
87 .EQU BUZZER_LCD_SI = 0
88
89 ;
    ********************************************************************************************
90 ; Valores para el menú del tiempo
91
92 .EQU CURSOR_POS_1_TIEMPO = 0x01
93 .EQU CURSOR_POS_2_TIEMPO = 0x02
94 .EQU CURSOR_POS_3_TIEMPO = 0x03
95 .EQU CURSOR_POS_4_TIEMPO = 0x04
96 .EQU CURSOR_POS_5_TIEMPO = 0x05
97 ;
    ********************************************************************************************
98 ; Valores para el menú del umbral
99
100 .EQU CURSOR_POS_1_UMBRAL = 0x01
```

```
.EQU CURSOR_POS_2_UMBRAL = 0x02
.EQU CURSOR_POS_3_UMBRAL = 0x03
.EQU CURSOR_POS_4_UMBRAL = 0x04
.EQU CURSOR_POS_5_UMBRAL = 0x05


;*****************************************************************************************

; Valores para entrar a los diferentes menús
.EQU ENTRAR_MENU_UMBRAL = 0xC1
.EQU ENTRAR_MENU_VENTANA = 0xE1
.EQU ENTRAR_MENU_TIEMPO_TOTAL = 0xC4
.EQU ENTRAR_MENU_COMENZAR_MEDICION = 0xC2
.EQU ENTRAR_MENU_BUZZER = 0xC8
.EQU VOLVER_MENU_PRINCIPAL = 0xD1

.EQU MIDIENDO_LCD = 0x00
.EQU MEDICION_FINALIZADA_LCD = 0xFF
;*****************************************************************************************

; Botones del registro del teclado

.EQU BOTON_OK = 1
.EQU BOTON_CANCELAR = 0
.EQU BOTON_INFERIOR = 2
.EQU BOTON_DERECHO = 3
.EQU BOTON_SUPERIOR = 4
.EQU BOTON_IZQUIERDA = 5
.EQU BOTON_ERROR = 7
;*****************************************************************************************

.dseg
; RAM MENU UMBRAL
    ================================================================================
MENU_UMBRAL_ASCII: .byte 6                          ; Se va a escribir la cadena ascii que se
     muestra por pantalla, del valor del umbral a setear
MENU_UMBRAL_POSICION_CURSOR: .byte 1                ; Este byte indica el peso del valor que
    se quiere modificar (decena,centena,unidad,etc...)
                                                    ; [7][6][5][4][3][2][1][0] ->
                                                    ; 0 = primer posicion, 1 = segunda
                                                        posicion, 2 = tercer posicion, 3 =
                                                        cuarta posicion, 4 = quinta posicion

; RAM MENU TIEMPO TOTAL
    ================================================================================

MENU_TIEMPO_TOTAL_ASCII: .byte 6                    ; El ascii del valor del tiempo total a
    setear

MENU_TIEMPO_TOTAL_POSICION_CURSOR: .byte 1          ; Idem el byte de la posición del cursor
    para el umbral

;*****************************************************************************************

.cseg
;*****************************************************************************************



MAQUINA_ESTADOS_LCD:
; Máquina de estados del LCD, para saber qué menú se está visualizando.
```

```
149
150     CPI ESTADOS_LCD, MEDICION_FINALIZADA_LCD
151     BREQ EST_MENU_FIN_MEDICION
152
153     SBRC ESTADOS_LCD, MENU_PRINCIPAL_LCD
154     RJMP EST_MENU_PRINCIPAL
155
156     SBRC ESTADOS_LCD, CONF_VENTANA_LCD
157     RJMP EST_CONFIG_VENTANA
158
159     SBRC ESTADOS_LCD, CONF_BUZZER_LCD
160     RJMP EST_CONF_BUZZER
161
162     SBRC ESTADOS_LCD, CONF_UMBRAL_LCD
163     RJMP EST_CONF_UMBRAL
164
165     SBRC ESTADOS_LCD, CONF_TIEMPO_TOTAL_LCD
166     RJMP EST_CONFIGURAR_TIEMPO_TOTAL
167
168     SBRC ESTADOS_LCD, COMENZAR_MEDICION_LCD
169     RJMP EST_COMENZAR_MEDICION
170
171
172
173
174 RET
175
176 ;
    ********************************************************************************************
177
178 EST_MENU_FIN_MEDICION:
179
180     SBRC BOTONES_TECLADO, BOTON_OK
181     LDI ESTADOS_LCD, VOLVER_MENU_PRINCIPAL
182
183     RET
184
185 ;
    ********************************************************************************************
186
187 EST_MENU_PRINCIPAL:
188
189     SBRS ESTADOS_LCD, ESCRIBIR_PRIMER_RENGLON           ; Si está en 0 el bit que habilita
        escribir el primer renglón, se saltea la escritura
190     RJMP SEGUIR_1_MENU_PRINCIPAL
191
192     COMANDO_LCD LCD_HOME_SCREEN                         ; Se escribe en el renglón superior
193     BORRAR_RENGLON                                      ; Se borra el primer renglón
194     COMANDO_LCD LCD_HOME_SCREEN                         ; Se posiciona al extremo superior
        izquierdo
195     CADENA_FLASH_LCD MENSAJE_MENU_PRINCIPAL             ; Se escribe el mensaje de menú
        principal
196
197     CBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)           ; Se deshabilita la opción de
        describir en el primer renglón
198
199 SEGUIR_1_MENU_PRINCIPAL:
200     SBRS ESTADOS_LCD, ESCRIBIR_SEGUNDO_RENGLON
201     RJMP SEGUIR_2_MENU_PRINCIPAL
202
203     COMANDO_LCD LCD_CAMBIAR_RENGLON
204     BORRAR_RENGLON
205     COMANDO_LCD LCD_CAMBIAR_RENGLON
206     CADENA_FLASH_LCD MENSAJE_MENU_PRINCIPAL_UMBRAL
```

```
207     SBR ESTADOS_LCD, (1<<CONF_UMBRAL_LCD)
208     CBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)
209
210 SEGUIR_2_MENU_PRINCIPAL:
211     ;SBRS BOTONES_TECLADO, BOTON_ERROR
212     CALL ACCION_BOTON_MENU_PRINCIPAL                    ; Se va a chequear en un registro, si
            se presionó algún botón, y que acción tomar, para el menú principal
213
214 RET
215
216 ;
    ***************************************************************************************************
217 EST_CONF_UMBRAL:
218
219     SBRS ESTADOS_LCD, ESCRIBIR_PRIMER_RENGLON           ; Si está en 0 el bit que
            habilita escribir el primer renglón, se saltea la escritura
220     RJMP SEGUIR_1_CONF_UMBRAL
221
222     COMANDO_LCD LCD_HOME_SCREEN                         ; Se posiciona en el primer
            renglón
223     BORRAR_RENGLON                                      ; Se borra el primer renglón
224     COMANDO_LCD LCD_HOME_SCREEN                         ; Se posiciona en el primer
            renglón
225     CADENA_FLASH_LCD MENSAJE_MENU_UMBRAL                ; Se muestra el texto que
            indica que me encuentro en el menú de cambio de umbral
226
227     CBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)           ; Se deshabilita la
            opción de escribir en el primer renglón
228
229     PUSH R29
230     PUSH R17
231     PUSH R16
232     PUSH XH
233     PUSH XL
234
235     LDS R29, REGISTRO_UMBRAL
236     LDS R17, REGISTRO_UMBRAL+1
237     LDS R16, REGISTRO_UMBRAL+2
238     LDI XL, LOW(MENU_UMBRAL_ASCII)
239     LDI XH, HIGH(MENU_UMBRAL_ASCII)
240
241     CALL DEC_TO_ASCII_24_BITS
242
243     POP XL
244     POP XH
245     POP R16
246     POP R17
247     POP R29
248
249     PUSH R17                                            ; Se inicializa el cursor en
            la posición de unidades
250     LDI R17, CURSOR_POS_1_UMBRAL
251     STS MENU_UMBRAL_POSICION_CURSOR, R17
252     POP R17
253
254 SEGUIR_1_CONF_UMBRAL:
255
256     SBRS ESTADOS_LCD, ESCRIBIR_SEGUNDO_RENGLON              ; Si está en 0 se saltea
            la escritura del segundo renglón
257     RJMP SEGUIR_2_CONF_UMBRAL
258
259     COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el segundo
            renglón
260     BORRAR_RENGLON                                      ; Se limpia el renglón
261     COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el extremo
```

```
        izquierdo del segundo renglón
262     COMANDO_LCD LCD_CURSOR_ON
263     PUSH ZH
264     PUSH ZL
265     LDI ZL, LOW(MENU_UMBRAL_ASCII)                          ; Se va a escribir lo que
            haya previamente cargado en el puntero Z
266     LDI ZH, HIGH(MENU_UMBRAL_ASCII)
267     CALL STRING_WRT_LCD
268     POP ZL
269     POP ZH
270
271     PUSH R17
272     LDS R17, MENU_UMBRAL_POSICION_CURSOR                    ; Se carga donde se
            encuentra el cursor
273     SIGO_SHIFTEO_UMBRAL:
274         CPI R17, 0x00
275         BREQ LISTO_SHIFTEO_UMBRAL
276         COMANDO_LCD LCD_SHIFTEAR_CURSOR_IZQUIERDA
277         DEC R17
278         RJMP SIGO_SHIFTEO_UMBRAL
279
280     LISTO_SHIFTEO_UMBRAL:
281     POP R17
282
283     CBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)          ; Se deshabilita la
            escritura del segundo renglón
284
285 SEGUIR_2_CONF_UMBRAL:
286
287 CALL ACCION_BOTON_CONF_UMBRAL                               ; Si se presionó un boton
        , se toma una acción en base a ese botón
288
289
290
291 RET
292
293 ;
        ***************************************************************************************************
294 EST_CONFIG_VENTANA:
295
296     SBRS ESTADOS_LCD, ESCRIBIR_PRIMER_RENGLON               ; Si está en 0 el bit que
            habilita escribir el primer renglón, se saltea la escritura
297     RJMP SEGUIR_1_CONF_VENTANA
298
299     COMANDO_LCD LCD_HOME_SCREEN                             ; Se posiciona en el primer
            renglón
300     BORRAR_RENGLON                                          ; Se borra el primer renglón
301     COMANDO_LCD LCD_HOME_SCREEN                             ; Se posiciona en el primer
            renglón
302     CADENA_FLASH_LCD MENSAJE_MENU_VENTANA                   ; Se muestra el texto que
            indica que me encuentro en el menú de cambio de umbral
303
304     CBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)           ; Se deshabilita la
            opción de escribir en el primer renglón
305
306 SEGUIR_1_CONF_VENTANA:
307
308     SBRS ESTADOS_LCD, ESCRIBIR_SEGUNDO_RENGLON              ; Si está en 0 se saltea la
            escritura del segundo renglón
309     RJMP SEGUIR_2_CONF_VENTANA
310
311     COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el segundo
            renglón
312     BORRAR_RENGLON                                          ; Se limpia el renglón
313     COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el extremo
```

```
        izquierdo del segundo renglón
314
315     CADENA_FLASH_LCD MENSAJE_1_ms
316
317     ;CALL STRING_WRT_FLASH                                  ; Se va a escribir lo que
            haya previamente cargado en el puntero Z
318     SBR ESTADOS_LCD, (1<<VENTANA_LCD_1ms)                                              ;
             El puntero Z se carga en la toma de decisiones en función del botón presionado
319     CBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)          ; Se deshabilita la
            escritura del segundo renglón
320
321 SEGUIR_2_CONF_VENTANA:
322
323 CALL ACCION_BOTON_CONF_VENTANA                              ; Se debe tomar una accion en
     base al boton presionado por el usuario
324
325 RET
326 ;
    ********************************************************************************************
327 EST_CONFIGURAR_TIEMPO_TOTAL:
328
329     SBRS ESTADOS_LCD, ESCRIBIR_PRIMER_RENGLON               ; Si está en 0 el bit que
            habilita escribir el primer renglón, se saltea la escritura
330     RJMP SEGUIR_1_CONF_TIEMPO
331
332     COMANDO_LCD LCD_HOME_SCREEN                             ; Se posiciona en el primer
            renglón
333     BORRAR_RENGLON                                          ; Se borra el primer renglón
334     COMANDO_LCD LCD_HOME_SCREEN                             ; Se posiciona en el primer
            renglón
335     CADENA_FLASH_LCD MENSAJE_MENU_TIEMPO                    ; Se muestra el texto que
            indica que me encuentro en el menú de cambio de umbral
336
337     CBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)           ; Se deshabilita la
            opción de escribir en el primer renglón
338
339     PUSH R29
340     PUSH R17
341     PUSH R16
342     PUSH XH
343     PUSH XL
344
345     LDS R29, VENTANAS_A_MEDIR_RAM
346     LDS R17, VENTANAS_A_MEDIR_RAM+1
347     LDS R16, VENTANAS_A_MEDIR_RAM+2
348     LDI XL, LOW(MENU_TIEMPO_TOTAL_ASCII)
349     LDI XH, HIGH(MENU_TIEMPO_TOTAL_ASCII)
350
351     CALL DEC_TO_ASCII_24_BITS
352
353     POP XL
354     POP XH
355     POP R16
356     POP R17
357     POP R29
358
359     PUSH R17                                                ; Se inicializa el cursor en
            la posición de unidades
360     LDI R17, CURSOR_POS_1_TIEMPO
361     STS MENU_TIEMPO_TOTAL_POSICION_CURSOR, R17
362     POP R17
363
364 SEGUIR_1_CONF_TIEMPO:
365
366     SBRS ESTADOS_LCD, ESCRIBIR_SEGUNDO_RENGLON              ; Si está en 0 se saltea la
```

```
        escritura del segundo renglón
367     RJMP SEGUIR_2_CONF_TIEMPO
368
369     COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el segundo
            renglón
370     BORRAR_RENGLON                                          ; Se limpia el renglón
371     COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el extremo
            izquierdo del segundo renglón
372     COMANDO_LCD LCD_CURSOR_ON
373     PUSH ZH
374     PUSH ZL
375     LDI ZL, LOW(MENU_TIEMPO_TOTAL_ASCII)                        ; Se va a escribir lo
             que haya previamente cargado en el puntero Z
376     LDI ZH, HIGH(MENU_TIEMPO_TOTAL_ASCII)
377     CALL STRING_WRT_LCD
378     POP ZL
379     POP ZH
380
381     PUSH R17
382     LDS R17, MENU_TIEMPO_TOTAL_POSICION_CURSOR                  ; Se carga donde se
            encuentra el cursor
383     SIGO_SHIFTEO:
384         CPI R17, 0x00
385         BREQ LISTO_SHIFTEO
386         COMANDO_LCD LCD_SHIFTEAR_CURSOR_IZQUIERDA
387         DEC R17
388         RJMP SIGO_SHIFTEO
389
390     LISTO_SHIFTEO:
391     POP R17
392
393     CBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)              ; Se deshabilita la
            escritura del segundo renglón
394
395 SEGUIR_2_CONF_TIEMPO:
396
397 CALL ACCION_BOTON_CONF_TIEMPO                                   ; Si se presionó un boton, se
         toma una acción en base a ese botón
398
399 RET
400 ;
    ********************************************************************************************
401 EST_COMENZAR_MEDICION:
402
403     SBRS ESTADOS_LCD, ESCRIBIR_PRIMER_RENGLON                   ; Si está en 0 el bit que
            habilita escribir el primer renglón, se saltea la escritura
404     RJMP SEGUIR_1_COMENZAR_MEDICION
405
406     COMANDO_LCD LCD_HOME_SCREEN                                 ; Se posiciona en el primer
            renglón
407     BORRAR_RENGLON                                              ; Se borra el primer renglón
408     COMANDO_LCD LCD_HOME_SCREEN                                 ; Se posiciona en el primer
            renglón
409     CADENA_FLASH_LCD MENSAJE_MENU_MIDIENDO                      ; Se muestra el texto que
            indica que me encuentro en el menú de cambio de umbral
410
411     CBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)               ; Se deshabilita la
            opción de escribir en el primer renglón
412
413 SEGUIR_1_COMENZAR_MEDICION:
414     SBRS ESTADOS_LCD, ESCRIBIR_SEGUNDO_RENGLON                  ; Escribo el segundo renglón
415     RJMP SEGUIR_2_COMENZAR_MEDICION
416
417     COMANDO_LCD LCD_CAMBIAR_RENGLON
418     BORRAR_RENGLON
```

```
419     COMANDO_LCD LCD_CAMBIAR_RENGLON
420
421     CBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)
422
423     SEGUIR_2_COMENZAR_MEDICION:
424
425     LDI ESTADOS_LCD, MIDIENDO_LCD                                   ; Se carga el
            estado de estar midiendo. Durante este estado no se entrará al menú de
426                                                                     ; la pantalla LCD
427
428     SBR ESTADO, (1<<EST_OSCIOSO_MEDICION)                           ; Bits necesarios
             para poder comenzar a medir
429     SBR EVENTO, (1<<COMENZAR_MEDICION)
430
431 RET
432 ;
    ********************************************************************************************
433 EST_CONF_BUZZER:
434
435     SBRS ESTADOS_LCD, ESCRIBIR_PRIMER_RENGLON                   ; Si está en 0 el bit que
            habilita escribir el primer renglón, se saltea la escritura
436     RJMP SEGUIR_1_CONF_BUZZER
437
438     COMANDO_LCD LCD_HOME_SCREEN                                 ; Se posiciona en el primer
            renglón
439     BORRAR_RENGLON                                              ; Se borra el primer renglón
440     COMANDO_LCD LCD_HOME_SCREEN                                 ; Se posiciona en el primer
            renglón
441     CADENA_FLASH_LCD MENSAJE_MENU_BUZZER                        ; Se muestra el texto que
            indica que me encuentro en el menú de cambio de umbral
442
443     CBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)                   ; Se deshabilita la
            opción de escribir en el primer renglón
444
445 SEGUIR_1_CONF_BUZZER:
446
447     SBRS ESTADOS_LCD, ESCRIBIR_SEGUNDO_RENGLON                  ; Si está en 0 se saltea la
            escritura del segundo renglón
448     RJMP SEGUIR_2_CONF_BUZZER
449
450     COMANDO_LCD LCD_CAMBIAR_RENGLON                             ; Me paro en el segundo
            renglón
451     BORRAR_RENGLON                                              ; Se limpia el renglón
452     COMANDO_LCD LCD_CAMBIAR_RENGLON                             ; Me paro en el extremo
            izquierdo del segundo renglón
453
454     CADENA_FLASH_LCD MENSAJE_BUZZER_NO
455
456     CBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)                  ; Se deshabilita la
            escritura del segundo renglón
457
458 SEGUIR_2_CONF_BUZZER:
459
460     CALL ACCION_BOTON_CONF_BUZZER                                   ; Se debe tomar una
            accion en base al boton presionado por el usuario
461
462 RET
463 ;
    ********************************************************************************************
464 ; Menú de toma de acciones en función del botón presionado, en cada menú
465
466
467 ACCION_BOTON_MENU_PRINCIPAL:
468 ; Menú donde se toma una decisión en base al botón presionado,
```

```
469 ; dentro del menú principal
470
471     SBRS BOTONES_TECLADO, BOTON_IZQUIERDA                    ; Salteo si no se presionó el
            boton izquierda
472     RJMP CHEQUEAR_BOTON_DERECHO_MENU_PRINCIPAL
473 ; =====================================================
474         SBRS ESTADOS_LCD, CONF_UMBRAL_LCD                        ; Salteo si no estaba
                mostrando la opción de entrar al menú de umbral
475         RJMP ACCION_MENU_PRINCIPAL_IZQ_VENTANA
476
477         COMANDO_LCD LCD_CAMBIAR_RENGLON                          ; Me paro en el segundo
                renglón
478         BORRAR_RENGLON                                           ; Se limpia el segundo
                renglón
479         COMANDO_LCD LCD_CAMBIAR_RENGLON                          ; Me paro al inicio del
                segundo renglón
480
481         CADENA_FLASH_LCD MENSAJE_MENU_PRINCIPAL_MEDIR            ; Escribo la opción de poder
                entrar a medir
482         CBR ESTADOS_LCD, (1<<CONF_UMBRAL_LCD)                        ; Se quita la opción de
                poder ingresar al menú de umbral
483         SBR ESTADOS_LCD, (1<<COMENZAR_MEDICION_LCD)                  ; Se setea la opción de
                poder ingresar al menú para comenzar la medición
484         RET
485 ; =====================================================
486         ACCION_MENU_PRINCIPAL_IZQ_VENTANA:
487
488         SBRS ESTADOS_LCD, CONF_VENTANA_LCD                       ; Salteo si no estaba
                mostrando la opción de comenzar la medición
489         RJMP ACCION_MENU_PRINCIPAL_IZQ_LARGO_TOTAL
490
491         COMANDO_LCD LCD_CAMBIAR_RENGLON                          ; Me paro en el segundo
                renglón
492         BORRAR_RENGLON                                           ; Se limpia el segundo
                renglón
493         COMANDO_LCD LCD_CAMBIAR_RENGLON                          ; Me paro al inicio del
                segundo renglón
494         CADENA_FLASH_LCD MENSAJE_MENU_PRINCIPAL_UMBRAL           ; Escribo la opción de poder
                configurar el umbral
495         CBR ESTADOS_LCD, (1<<CONF_VENTANA_LCD)
496         SBR ESTADOS_LCD, (1<<CONF_UMBRAL_LCD)
497         RET
498 ; =====================================================
499         ACCION_MENU_PRINCIPAL_IZQ_LARGO_TOTAL:
500
501         SBRS ESTADOS_LCD, CONF_TIEMPO_TOTAL_LCD
502         RJMP ACCION_MENU_PRINCIPAL_IZQ_BUZZER
503
504         COMANDO_LCD LCD_CAMBIAR_RENGLON
505         BORRAR_RENGLON
506         COMANDO_LCD LCD_CAMBIAR_RENGLON
507         CADENA_FLASH_LCD MENSAJE_MENU_PRINCIPAL_VENTANA
508         CBR ESTADOS_LCD, (1<<CONF_TIEMPO_TOTAL_LCD)
509         SBR ESTADOS_LCD, (1<<CONF_VENTANA_LCD)
510         RET
511 ; =====================================================
512         ACCION_MENU_PRINCIPAL_IZQ_BUZZER:
513
514         SBRS ESTADOS_LCD, CONF_BUZZER_LCD
515         RJMP ACCION_MENU_PRINCIPAL_IZQ_COMENZAR_MEDICION
516
517         COMANDO_LCD LCD_CAMBIAR_RENGLON
518         BORRAR_RENGLON
519         COMANDO_LCD LCD_CAMBIAR_RENGLON
520         CADENA_FLASH_LCD MENSAJE_MENU_PRINCIPAL_DURACION
521         CBR ESTADOS_LCD, (1<<CONF_BUZZER_LCD)
```

```
522         SBR ESTADOS_LCD, (1<<CONF_TIEMPO_TOTAL_LCD)
523         RET
524 ; =====================================================
525         ACCION_MENU_PRINCIPAL_IZQ_COMENZAR_MEDICION:
526
527         SBRS ESTADOS_LCD, COMENZAR_MEDICION_LCD
528         RJMP CHEQUEAR_BOTON_DERECHO_MENU_PRINCIPAL
529
530         COMANDO_LCD LCD_CAMBIAR_RENGLON
531         BORRAR_RENGLON
532         COMANDO_LCD LCD_CAMBIAR_RENGLON
533         CADENA_FLASH_LCD MENSAJE_MENU_PRINCIPAL_BUZZER
534         CBR ESTADOS_LCD, (1<<COMENZAR_MEDICION_LCD)
535         SBR ESTADOS_LCD, (1<<CONF_BUZZER_LCD)
536         RET
537 ; =====================================================
538
539     CHEQUEAR_BOTON_DERECHO_MENU_PRINCIPAL:
540
541     SBRS BOTONES_TECLADO, BOTON_DERECHO
542     RJMP CHEQUEAR_BOTON_ARRIBA_MENU_PRINCIPAL
543
544 ; =====================================================
545         SBRS ESTADOS_LCD, CONF_UMBRAL_LCD                       ; Salteo si no estaba
                mostrando la opción de entrar al menú de umbral
546         RJMP ACCION_MENU_PRINCIPAL_DER_VENTANA
547
548         COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el segundo
                renglón
549         BORRAR_RENGLON                                          ; Se limpia el segundo
                renglón
550         COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro al inicio del
                segundo renglón
551
552         CADENA_FLASH_LCD MENSAJE_MENU_PRINCIPAL_VENTANA         ; Escribo la opción de poder
                entrar a medir
553         CBR ESTADOS_LCD, (1<<CONF_UMBRAL_LCD)                   ; Se quita la opción de
                poder ingresar al menú de umbral
554         SBR ESTADOS_LCD, (1<<CONF_VENTANA_LCD)                  ; Se setea la opción de
                poder ingresar al menú para comenzar la medición
555         RET
556 ; =====================================================
557         ACCION_MENU_PRINCIPAL_DER_VENTANA:
558
559         SBRS ESTADOS_LCD, CONF_VENTANA_LCD                      ; Salteo si no estaba
                mostrando la opción de comenzar la medición
560         RJMP ACCION_MENU_PRINCIPAL_DER_LARGO_TOTAL
561
562         COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el segundo
                renglón
563         BORRAR_RENGLON                                          ; Se limpia el segundo
                renglón
564         COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro al inicio del
                segundo renglón
565
566         CADENA_FLASH_LCD MENSAJE_MENU_PRINCIPAL_DURACION        ; Escribo la opción de poder
                configurar el umbral
567         CBR ESTADOS_LCD, (1<<CONF_VENTANA_LCD)
568         SBR ESTADOS_LCD, (1<<CONF_TIEMPO_TOTAL_LCD)
569         RET
570 ; =====================================================
571         ACCION_MENU_PRINCIPAL_DER_LARGO_TOTAL:
572
573         SBRS ESTADOS_LCD, CONF_TIEMPO_TOTAL_LCD
574         RJMP ACCION_MENU_PRINCIPAL_DER_BUZZER
575
```

```
576         COMANDO_LCD LCD_CAMBIAR_RENGLON
577         BORRAR_RENGLON
578         COMANDO_LCD LCD_CAMBIAR_RENGLON
579
580         CADENA_FLASH_LCD MENSAJE_MENU_PRINCIPAL_BUZZER
581         CBR ESTADOS_LCD, (1<<CONF_TIEMPO_TOTAL_LCD)
582         SBR ESTADOS_LCD, (1<<CONF_BUZZER_LCD)
583         RET
584 ; ====================================================
585         ACCION_MENU_PRINCIPAL_DER_BUZZER:
586
587         SBRS ESTADOS_LCD, CONF_BUZZER_LCD
588         RJMP ACCION_MENU_PRINCIPAL_DER_COMENZAR_MEDICION
589
590         COMANDO_LCD LCD_CAMBIAR_RENGLON
591         BORRAR_RENGLON
592         COMANDO_LCD LCD_CAMBIAR_RENGLON
593         CADENA_FLASH_LCD MENSAJE_MENU_PRINCIPAL_MEDIR
594         CBR ESTADOS_LCD, (1<<CONF_BUZZER_LCD)
595         SBR ESTADOS_LCD, (1<<COMENZAR_MEDICION_LCD)
596         RET
597 ; ====================================================
598         ACCION_MENU_PRINCIPAL_DER_COMENZAR_MEDICION:
599
600         SBRS ESTADOS_LCD, COMENZAR_MEDICION_LCD
601         RJMP CHEQUEAR_BOTON_ARRIBA_MENU_PRINCIPAL
602
603         COMANDO_LCD LCD_CAMBIAR_RENGLON
604         BORRAR_RENGLON
605         COMANDO_LCD LCD_CAMBIAR_RENGLON
606
607         CADENA_FLASH_LCD MENSAJE_MENU_PRINCIPAL_UMBRAL
608         CBR ESTADOS_LCD, (1<<COMENZAR_MEDICION_LCD)
609         SBR ESTADOS_LCD, (1<<CONF_UMBRAL_LCD)
610         RET
611 ; ====================================================
612     CHEQUEAR_BOTON_ARRIBA_MENU_PRINCIPAL:
613
614     SBRS BOTONES_TECLADO, BOTON_SUPERIOR
615     RJMP CHEQUEAR_BOTON_ABAJO_MENU_PRINCIPAL
616     RET                                                 ; Para el caso del menú
            principal, el botón superior no activa nada
617
618     CHEQUEAR_BOTON_ABAJO_MENU_PRINCIPAL:
619
620     SBRS BOTONES_TECLADO, BOTON_INFERIOR
621     RJMP CHEQUEAR_BOTON_OK_MENU_PRINCIPAL
622     RET                                                 ; Para el caso del menú
            principal, el botón inferior no activa nada
623
624     CHEQUEAR_BOTON_OK_MENU_PRINCIPAL:
625     SBRS BOTONES_TECLADO, BOTON_OK
626     RJMP CHEQUEAR_BOTON_CANCELAR_MENU_PRINCIPAL
627
628 ; ====================================================
629         SBRS ESTADOS_LCD, CONF_UMBRAL_LCD                   ; Salteo si no estaba
                mostrando la opción de entrar al menú de umbral
630         RJMP ACCION_MENU_PRINCIPAL_OK_VENTANA
631
632         LDI ESTADOS_LCD, ENTRAR_MENU_UMBRAL                 ; Se setean los estados
                para entrar al menú de umbral
633
634 /*      CBR ESTADOS_LCD, (1<<MENU_PRINCIPAL_LCD)                ; Se setea que ya no
    se está en el menu principal
635         SBR ESTADOS_LCD, (1<<CONF_UMBRAL_LCD)               ; Se setea que se quiere
                entrar al menu de configurar el umbral
```

```
636         SBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)               ; Se habilita que se
               escriba en el primer renglon*/
637         RET
638 ; =====================================================
639         ACCION_MENU_PRINCIPAL_OK_VENTANA:
640         SBRS ESTADOS_LCD, CONF_VENTANA_LCD
641         RJMP ACCION_MENU_PRINCIPAL_OK_DURACION
642
643
644         LDI ESTADOS_LCD, ENTRAR_MENU_VENTANA
645 /*      CBR ESTADOS_LCD, (1<<MENU_PRINCIPAL_LCD)                    ; Se indica que ya no
    estoy en el menu principal
646         SBR ESTADOS_LCD, (1<<CONF_VENTANA_LCD)                      ; Se indica que ahora voy
                a estar en el menú de configuracion de ventana
647         SBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)               ; Se habilita la
               escritura del primer renglon
648         SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)*/
649         RET
650 ; =====================================================
651         ACCION_MENU_PRINCIPAL_OK_DURACION:
652         SBRS ESTADOS_LCD, CONF_TIEMPO_TOTAL_LCD
653         RJMP ACCION_MENU_PRINCIPAL_OK_BUZZER
654
655         LDI ESTADOS_LCD, ENTRAR_MENU_TIEMPO_TOTAL
656
657 /*      CBR ESTADOS_LCD, (1<<MENU_PRINCIPAL_LCD)                        ; Se indica que ya no
     se está en el el menú principal
658         SBR ESTADOS_LCD, (1<<CONF_TIEMPO_TOTAL_LCD)                 ; Se indica que ahora
               estoy en el menú de configurar buzzer
659         SBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)               ; Se habilita la
               escritura del primer renglón
660         SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)*/
661
662         RET
663 ; =====================================================
664         ACCION_MENU_PRINCIPAL_OK_BUZZER:
665         SBRS ESTADOS_LCD, CONF_BUZZER_LCD
666         RJMP ACCION_MENU_PRINCIPAL_OK_MEDIR
667
668         LDI ESTADOS_LCD, ENTRAR_MENU_BUZZER
669
670 /*      CBR ESTADOS_LCD, (1<<MENU_PRINCIPAL_LCD)                        ; Se inhabilita que
    estoy en el menú principal
671         SBR ESTADOS_LCD, (1<<CONF_BUZZER_LCD)                       ; Se habilita que se va a
                entrar en el menú de configurar el buzzer
672         SBR EStADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)                   ; Se habilita la
               escritura del primer renglón*/
673         RET
674 ; =====================================================
675         ACCION_MENU_PRINCIPAL_OK_MEDIR:
676         SBRS ESTADOS_LCD, COMENZAR_MEDICION_LCD
677         RJMP NO_ACCION_MENU_PRINCIPAL_OK
678
679         LDI ESTADOS_LCD, ENTRAR_MENU_COMENZAR_MEDICION
680
681 /*      CBR ESTADOS_LCD, (1<<MENU_PRINCIPAL_LCD)                        ; Se inhabilita que
    estoy en el menú principal
682         SBR ESTADOS_LCD, (1<<COMENZAR_MEDICION_LCD)                 ; Se habilita que se
               entra en el menú medir
683         SBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)               ; Se habilita la
               inscripción del primer renglón
684         SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)*/
685         RET
686
687         NO_ACCION_MENU_PRINCIPAL_OK:
688 ; =====================================================
```

```
689     CHEQUEAR_BOTON_CANCELAR_MENU_PRINCIPAL:
690
691     SBRS BOTONES_TECLADO, BOTON_CANCELAR
692     RJMP NO_BOTON_MENU_PRINCIPAL
693
694     RET                                                     ; En particular en el menú
            principal, no se toma ninguna acción, si se presiona cancelar
695     NO_BOTON_MENU_PRINCIPAL:
696 RET
697 ;
    ********************************************************************************************

698 ACCION_BOTON_CONF_VENTANA:
699
700     SBRS BOTONES_TECLADO, BOTON_IZQUIERDA                   ; Si estoy en la ventana
            y presiono el botón izquierdo, entro aquí
701     RJMP CHEQUEAR_BOTON_DERECHO_VENTANA
702
703         SBRS ESTADOS_LCD, VENTANA_LCD_1ms                   ; Si no estoy en la
                posición de mostrar 1 ms, salteo esta parte
704         RJMP BOTON_IZQUIERDA_VENTANA_10ms                   ; Sigo comprobando si
                estoy en la posición de mostrar 10ms
705
706             COMANDO_LCD LCD_CAMBIAR_RENGLON                 ; Me paro en el segundo
                    renglón
707             BORRAR_RENGLON                                  ; Se limpia el segundo
                    renglón
708             COMANDO_LCD LCD_CAMBIAR_RENGLON
709
710             CADENA_FLASH_LCD MENSAJE_1000_ms                ; Se escribe el mensaje
                    de 1000ms
711
712             SBR ESTADOS_LCD, (1<<VENTANA_LCD_1000ms)
713             CBR ESTADOS_LCD, (1<<VENTANA_LCD_1ms)
714
715             RJMP NO_ACCION_VENTANA
716
717         BOTON_IZQUIERDA_VENTANA_10ms:
718         SBRS ESTADOS_LCD, VENTANA_LCD_10ms
719         RJMP BOTON_IZQUIERDA_VENTANA_100ms
720
721             COMANDO_LCD LCD_CAMBIAR_RENGLON                 ; Me paro en el segundo
                    renglón
722             BORRAR_RENGLON                                  ; Se limpia el segundo
                    renglón
723             COMANDO_LCD LCD_CAMBIAR_RENGLON
724
725             CADENA_FLASH_LCD MENSAJE_1_ms                   ; Se escribe el mensaje
                    de 1_ms
726
727             SBR ESTADOS_LCD, (1<<VENTANA_LCD_1ms)
728             CBR ESTADOS_LCD, (1<<VENTANA_LCD_10ms)
729
730             RJMP NO_ACCION_VENTANA
731
732         BOTON_IZQUIERDA_VENTANA_100ms:
733         SBRS ESTADOS_LCD, VENTANA_LCD_100ms
734         RJMP BOTON_IZQUIERDA_VENTANA_1000ms
735
736             COMANDO_LCD LCD_CAMBIAR_RENGLON                 ; Me paro en el segundo
                    renglón
737             BORRAR_RENGLON                                  ; Se limpia el segundo
                    renglón
738             COMANDO_LCD LCD_CAMBIAR_RENGLON
739
740             CADENA_FLASH_LCD MENSAJE_10_ms                  ; Se escribe el mensaje
```

```
                de 10 ms
741
742             SBR ESTADOS_LCD, (1<<VENTANA_LCD_10ms)
743             CBR ESTADOS_LCD, (1<<VENTANA_LCD_100ms)
744
745             RJMP NO_ACCION_VENTANA
746
747         BOTON_IZQUIERDA_VENTANA_1000ms:
748         SBRS ESTADOS_LCD, VENTANA_LCD_1000ms
749         RJMP CHEQUEAR_BOTON_DERECHO_VENTANA
750
751             COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el segundo
                renglón
752             BORRAR_RENGLON                                          ; Se limpia el segundo
                renglón
753             COMANDO_LCD LCD_CAMBIAR_RENGLON
754
755             CADENA_FLASH_LCD MENSAJE_100_ms                         ; Se escribe el mensaje
                de 100ms
756
757             SBR ESTADOS_LCD, (1<<VENTANA_LCD_100ms)
758             CBR ESTADOS_LCD, (1<<VENTANA_LCD_1000ms)
759
760             RJMP NO_ACCION_VENTANA
761
762     CHEQUEAR_BOTON_DERECHO_VENTANA:
763
764     SBRS BOTONES_TECLADO, BOTON_DERECHO
765     RJMP CHEQUEAR_BOTON_SUPERIOR_VENTANA
766
767         SBRS ESTADOS_LCD, VENTANA_LCD_1ms                                       ; Si no estoy
                en la posición de mostrar 1 ms, salteo esta parte
768         RJMP BOTON_DERECHA_VENTANA_10ms                         ; Sigo comprobando si estoy
               en la posición de mostrar 10ms
769
770             COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el segundo
                renglón
771             BORRAR_RENGLON                                          ; Se limpia el segundo
                renglón
772             COMANDO_LCD LCD_CAMBIAR_RENGLON
773
774             CADENA_FLASH_LCD MENSAJE_10_ms                          ; Se escribe el mensaje
                de 10 ms
775
776             SBR ESTADOS_LCD, (1<<VENTANA_LCD_10ms)
777             CBR ESTADOS_LCD, (1<<VENTANA_LCD_1ms)
778
779             RJMP NO_ACCION_VENTANA
780
781         BOTON_DERECHA_VENTANA_10ms:
782         SBRS ESTADOS_LCD, VENTANA_LCD_10ms
783         RJMP BOTON_DERECHA_VENTANA_100ms
784
785             COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el segundo
                renglón
786             BORRAR_RENGLON                                          ; Se limpia el segundo
                renglón
787             COMANDO_LCD LCD_CAMBIAR_RENGLON
788
789             CADENA_FLASH_LCD MENSAJE_100_ms                         ; Se escribe el mensaje
                de 100 ms
790
791             SBR ESTADOS_LCD, (1<<VENTANA_LCD_100ms)
792             CBR ESTADOS_LCD, (1<<VENTANA_LCD_10ms)
793
794             RJMP NO_ACCION_VENTANA
```

```
795
796         BOTON_DERECHA_VENTANA_100ms:
797         SBRS ESTADOS_LCD, VENTANA_LCD_100ms
798         RJMP BOTON_DERECHA_VENTANA_1000ms
799
800             COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el segundo
                    renglón
801             BORRAR_RENGLON                                          ; Se limpia el segundo
                    renglón
802             COMANDO_LCD LCD_CAMBIAR_RENGLON
803
804             CADENA_FLASH_LCD MENSAJE_1000_ms
805
806             SBR ESTADOS_LCD, (1<<VENTANA_LCD_1000ms)
807             CBR ESTADOS_LCD, (1<<VENTANA_LCD_100ms)
808
809             RJMP NO_ACCION_VENTANA
810
811         BOTON_DERECHA_VENTANA_1000ms:
812         SBRS ESTADOS_LCD, VENTANA_LCD_1000ms
813         RJMP CHEQUEAR_BOTON_SUPERIOR_VENTANA
814
815             COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el segundo
                    renglón
816             BORRAR_RENGLON                                          ; Se limpia el segundo
                    renglón
817             COMANDO_LCD LCD_CAMBIAR_RENGLON
818
819             CADENA_FLASH_LCD MENSAJE_1_ms
820
821             SBR ESTADOS_LCD, (1<<VENTANA_LCD_1ms)
822             CBR ESTADOS_LCD, (1<<VENTANA_LCD_1000ms)
823
824             RJMP NO_ACCION_VENTANA
825
826
827
828     CHEQUEAR_BOTON_SUPERIOR_VENTANA:
829     SBRS BOTONES_TECLADO, BOTON_SUPERIOR
830     RJMP CHEQUEAR_BOTON_INFERIOR_VENTANA
831
832             RJMP NO_ACCION_VENTANA
833
834
835     CHEQUEAR_BOTON_INFERIOR_VENTANA:
836     SBRS BOTONES_TECLADO, BOTON_INFERIOR
837     RJMP CHEQUEAR_BOTON_OK_VENTANA
838
839         RJMP NO_ACCION_VENTANA
840
841     CHEQUEAR_BOTON_OK_VENTANA:
842     SBRS BOTONES_TECLADO, BOTON_OK
843     RJMP CHEQUEAR_BOTON_CANCELAR_VENTANA
844     PUSH R16
845
846         VENTANA_GUARDAR_1_ms:
847
848         SBRS ESTADOS_LCD, VENTANA_LCD_1ms                           ; Si el usuario está
                visulizando 1 ms, se guarda dicha configuración
849         RJMP VENTANA_GUARDAR_10_ms
850
851             LDI R16, VENTANA_1ms
852             RJMP VENTANA_GUARDAR_SEGUIR
853
854         VENTANA_GUARDAR_10_ms:                                      ; Si el usuario está
                visulizando 10 ms, se guarda dicha configuración
```

```
855         SBRS ESTADOS_LCD, VENTANA_LCD_10ms
856         RJMP VENTANA_GUARDAR_100_ms
857
858             LDI R16, VENTANA_10ms
859             RJMP VENTANA_GUARDAR_SEGUIR
860
861         VENTANA_GUARDAR_100_ms:                                 ; Si el usuario está
                visulizando 100 ms, se guarda dicha configuración
862         SBRS ESTADOS_LCD, VENTANA_LCD_100ms
863         RJMP VENTANA_GUARDAR_1000ms
864
865             LDI R16, VENTANA_100ms
866             RJMP VENTANA_GUARDAR_SEGUIR
867
868         VENTANA_GUARDAR_1000ms:                                 ; Si el usuario está
                visulizando 1000 ms, se guarda dicha configuración
869         SBRS ESTADOS_LCD, VENTANA_LCD_1000ms
870         RJMP VENTANA_GUARDAR_SEGUIR
871
872             LDI R16, VENTANA_1000ms
873             RJMP VENTANA_GUARDAR_SEGUIR
874
875         VENTANA_GUARDAR_SEGUIR:
876
877         STS REGISTRO_VENTANA_TIEMPO, R16
878         POP R16
879
880         LDI ESTADOS_LCD, VOLVER_MENU_PRINCIPAL
881 /*      SBR ESTADOS_LCD, (1<<MENU_PRINCIPAL_LCD)                ; Configuro que vuelvo al
        menú principal
882         SBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)
883         SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)          ; Indico que puedo
                escribir nuevamente en el segundo renglón*/
884         RJMP NO_ACCION_VENTANA
885
886     CHEQUEAR_BOTON_CANCELAR_VENTANA:
887     SBRS BOTONES_TECLADO, BOTON_CANCELAR
888     RJMP NO_ACCION_VENTANA
889
890         LDI ESTADOS_LCD, VOLVER_MENU_PRINCIPAL
891 /*      SBR ESTADOS_LCD, (1<<MENU_PRINCIPAL_LCD)                    ; Configuro que
        vuelvo al menú principal
892         SBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)
893         SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)          ; Indico que puedo
                escribir nuevamente en el segundo renglón*/
894         RJMP NO_ACCION_VENTANA
895
896
897     NO_ACCION_VENTANA:
898
899 RET
900
901 ;   ***************************************************************************************************
902 ACCION_BOTON_CONF_TIEMPO:
903
904     PUSH R17
905     LDS R17, MENU_TIEMPO_TOTAL_POSICION_CURSOR                  ; Se carga el dato de la
            posición del cursor
906
907     SBRS BOTONES_TECLADO, BOTON_IZQUIERDA
908     RJMP CHEQUEAR_BOTON_DERECHO_TIEMPO
909
910         ; Se chequea donde está el cursor, y donde debería estar ahora
911         CPI R17, CURSOR_POS_5_TIEMPO                            ; Si me encuentro en la
```

```
            posición más significativa, entonces no hago nada, ya que no puedo mover cursor
912         BRNE TIEMPO_INCREMENTAR_CURSOR
913         RJMP NO_ACCION_TIEMPO
914
915         TIEMPO_INCREMENTAR_CURSOR:
916         COMANDO_LCD LCD_SHIFTEAR_CURSOR_IZQUIERDA               ; Si no, incremento el valor
                de la posición del cursor.
917         INC R17
918         RJMP NO_ACCION_TIEMPO
919
920     CHEQUEAR_BOTON_DERECHO_TIEMPO:                              ; Idem botón izquierdo pero
            para el botón derecho
921
922     SBRS BOTONES_TECLADO, BOTON_DERECHO
923     RJMP CHEQUEAR_BOTON_SUPERIOR_TIEMPO
924
925         CPI R17, CURSOR_POS_1_TIEMPO
926         BRNE TIEMPO_DECREMENTAR_CURSOR
927         RJMP NO_ACCION_TIEMPO
928
929         TIEMPO_DECREMENTAR_CURSOR:
930         COMANDO_LCD LCD_SHIFTEAR_CURSOR_DERECHA
931         DEC R17
932         RJMP NO_ACCION_TIEMPO
933
934     CHEQUEAR_BOTON_SUPERIOR_TIEMPO:                                 ; Se utiliza para
            incrementar el total del valor de la cantidad de ventanas a medir
935     SBRS BOTONES_TECLADO, BOTON_SUPERIOR
936     RJMP CHEQUEAR_BOTON_INFERIOR_TIEMPO
937
938         PUSH ZL
939         PUSH ZH
940
941         LDI ZL, LOW(MENU_TIEMPO_TOTAL_ASCII)                    ; Se levanta el ASCII
                de la cantidad de ventanas a medir, para modificarlo según corresponda
942         LDI ZH, HIGH(MENU_TIEMPO_TOTAL_ASCII)
943
944         CPI R17, CURSOR_POS_1_TIEMPO                            ; Chequeo si se está
                en la posición menos significativa
945         BRNE CHEQUEO_POS_2_TIEMPO_ARRIBA
946
947             ADIW ZH:ZL, CURSOR_POS_5_TIEMPO -1                  ; Se corre el cursor
                    hasta la posición menos significativa
948             RJMP BOTON_SUPERIOR_CAMBIAR_VALOR_TIEMPO
949
950
951         CHEQUEO_POS_2_TIEMPO_ARRIBA:
952
953         CPI R17, CURSOR_POS_2_TIEMPO
954         BRNE CHEQUEO_POS_3_TIEMPO_ARRIBA
955
956             ADIW ZH:ZL, CURSOR_POS_4_TIEMPO -1                  ; Se corre el cursor
                    hasta la posición de las decenas
957             RJMP BOTON_SUPERIOR_CAMBIAR_VALOR_TIEMPO
958
959         CHEQUEO_POS_3_TIEMPO_ARRIBA:
960
961         CPI R17, CURSOR_POS_3_TIEMPO
962         BRNE CHEQUEO_POS_4_TIEMPO_ARRIBA
963
964             ADIW ZH:ZL, CURSOR_POS_3_TIEMPO -1                  ; Se corre el cursor
                    hasta la posición de las centenas
965             RJMP BOTON_SUPERIOR_CAMBIAR_VALOR_TIEMPO
966
967         CHEQUEO_POS_4_TIEMPO_ARRIBA:
968
```

```
969         CPI R17, CURSOR_POS_4_TIEMPO
970         BRNE CHEQUEO_POS_5_TIEMPO_ARRIBA
971
972             ADIW ZH:ZL, CURSOR_POS_2_TIEMPO -1                          ; Se corre el cursor
                    hasta la posición de las unidades de mil
973             RJMP BOTON_SUPERIOR_CAMBIAR_VALOR_TIEMPO
974
975         CHEQUEO_POS_5_TIEMPO_ARRIBA:
976
977         CPI R17, CURSOR_POS_5_TIEMPO
978         BRNE BOTON_SUPERIOR_TIEMPO_SEGUIR
979
980             ADIW ZH:ZL, CURSOR_POS_1_TIEMPO -1                          ; Se corre el cursor
                    hasta la posición de las decenas de mil
981             RJMP BOTON_SUPERIOR_CAMBIAR_VALOR_TIEMPO
982
983         BOTON_SUPERIOR_CAMBIAR_VALOR_TIEMPO:                            ; Por último, se
                modifica el valor correspondiente y se guarda
984
985             PUSH R18
986             LD R18, Z                                                   ; Se carga el
                    caracter ascii correspondiente
987             CPI R18, '9'                                                ; Si el caracter es
                    un 9 ascii, no se puede incrementar más
988             BREQ BOTON_SUPERIOR_TIEMPO_0
989
990             INC R18
991
992             RJMP BOTON_SUPERIOR_TIEMPO_SEGUIR
993
994             BOTON_SUPERIOR_TIEMPO_0:
995                 LDI R18, '0'                                            ; Se carga un 0 ascii
996
997         BOTON_SUPERIOR_TIEMPO_SEGUIR:
998             ST Z, R18                                                   ; Se guarda el dato
                    incrementado
999             POP R18                                                     ; Se recuperan los
                    datos que había en los registros originalmente
1000            POP ZH
1001            POP ZL
1002
1003            SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)              ; Se habilita la
                    escritura en el segundo renglón
1004            RJMP NO_ACCION_TIEMPO
1005
1006    CHEQUEAR_BOTON_INFERIOR_TIEMPO:
1007    SBRS BOTONES_TECLADO, BOTON_INFERIOR
1008    RJMP CHEQUEAR_BOTON_OK_TIEMPO
1009
1010        PUSH ZL
1011        PUSH ZH
1012
1013        LDI ZL, LOW(MENU_TIEMPO_TOTAL_ASCII)                            ; Se levanta el ASCII
                de la cantidad de ventanas a medir, para modificarlo según corresponda
1014        LDI ZH, HIGH(MENU_TIEMPO_TOTAL_ASCII)
1015
1016        CPI R17, CURSOR_POS_1_TIEMPO                                    ; Chequeo si se está
                en la posición menos significativa
1017        BRNE CHEQUEO_POS_2_TIEMPO_ABAJO
1018
1019            ADIW ZH:ZL, CURSOR_POS_5_TIEMPO -1                          ; Se corre el cursor
                    hasta la posición menos significativa
1020            RJMP BOTON_INFERIOR_CAMBIAR_VALOR_TIEMPO
1021
1022
1023        CHEQUEO_POS_2_TIEMPO_ABAJO:
```

```
1024
1025        CPI R17, CURSOR_POS_2_TIEMPO
1026        BRNE CHEQUEO_POS_3_TIEMPO_ABAJO
1027
1028            ADIW ZH:ZL, CURSOR_POS_4_TIEMPO -1                          ; Se corre el cursor
                    hasta la posición de las decenas
1029            RJMP BOTON_INFERIOR_CAMBIAR_VALOR_TIEMPO
1030
1031        CHEQUEO_POS_3_TIEMPO_ABAJO:
1032
1033        CPI R17, CURSOR_POS_3_TIEMPO
1034        BRNE CHEQUEO_POS_4_TIEMPO_ABAJO
1035
1036            ADIW ZH:ZL, CURSOR_POS_3_TIEMPO -1                          ; Se corre el cursor
                    hasta la posición de las centenas
1037            RJMP BOTON_INFERIOR_CAMBIAR_VALOR_TIEMPO
1038
1039        CHEQUEO_POS_4_TIEMPO_ABAJO:
1040
1041        CPI R17, CURSOR_POS_4_TIEMPO
1042        BRNE CHEQUEO_POS_5_TIEMPO_ABAJO
1043
1044            ADIW ZH:ZL, CURSOR_POS_2_TIEMPO -1                          ; Se corre el cursor
                    hasta la posición de las unidades de mil
1045            RJMP BOTON_INFERIOR_CAMBIAR_VALOR_TIEMPO
1046
1047        CHEQUEO_POS_5_TIEMPO_ABAJO:
1048
1049        CPI R17, CURSOR_POS_5_TIEMPO
1050        BRNE BOTON_INFERIOR_TIEMPO_SEGUIR
1051
1052            ADIW ZH:ZL, CURSOR_POS_1_TIEMPO -1                          ; Se corre el cursor
                    hasta la posición de las decenas de mil
1053            RJMP BOTON_INFERIOR_CAMBIAR_VALOR_TIEMPO
1054
1055        BOTON_INFERIOR_CAMBIAR_VALOR_TIEMPO:                            ; Por último, se
               modifica el valor correspondiente y se guarda
1056
1057            PUSH R18
1058            LD R18, Z                                                   ; Se carga el
                    caracter ascii correspondiente
1059            CPI R18, '0'                                                ; Si el caracter es
                    un 0 ascii, no se puede decrementar más
1060            BREQ BOTON_INFERIOR_TIEMPO_9
1061
1062            DEC R18
1063
1064            RJMP BOTON_INFERIOR_TIEMPO_SEGUIR
1065
1066            BOTON_INFERIOR_TIEMPO_9:
1067                LDI R18, '9'                                            ; Se carga un 9 ascii
1068
1069        BOTON_INFERIOR_TIEMPO_SEGUIR:
1070            ST Z, R18                                                   ; Se guarda el dato
                    incrementado
1071            POP R18                                                     ; Se recuperan los
                    datos que había en los registros originalmente
1072            POP ZH
1073            POP ZL
1074
1075            SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)              ; Se habilita la
                    escritura en el segundo renglón
1076            RJMP NO_ACCION_TIEMPO
1077
1078
1079    CHEQUEAR_BOTON_OK_TIEMPO:
```

```
1080      SBRS BOTONES_TECLADO, BOTON_OK
1081      RJMP CHEQUEAR_BOTON_CANCELAR_TIEMPO
1082
1083          PUSH XH
1084          PUSH XL
1085          PUSH YH
1086          PUSH YL
1087          PUSH R16
1088
1089          LDI XL, LOW(MENU_TIEMPO_TOTAL_ASCII)                    ; Se carga el puntero
                  de donde se encuentra guardado del valor modificado
1090          LDI XH, HIGH(MENU_TIEMPO_TOTAL_ASCII)
1091          LDI YL, LOW(VENTANAS_A_MEDIR_RAM)                       ; Se carga el puntero
                  donde se quiere guardar el valor convertido a hexa
1092          LDI YH, HIGH(VENTANAS_A_MEDIR_RAM)
1093          LDI R16, MAX_NUM_DIGITOS_5                              ; Se carga cuantos
                 digitos se quiere convertir a hexa
1094          CALL TRANSFORMAR_DE_ASCII_A_BINARIO_Y_GUARDAR_5_DIGITOS ; Se guarda el valor
                 de la cantidad de ventanas a medir, modificado
1095
1096          POP R16
1097          POP YL
1098          POP YH
1099          POP XL
1100          POP XH
1101
1102          LDI ESTADOS_LCD, VOLVER_MENU_PRINCIPAL
1103
1104  /*      SBR ESTADOS_LCD, (1<<MENU_PRINCIPAL_LCD)                ; Se configura para
      regresar al menpu principal
1105          SBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)
1106          SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)*/
1107          COMANDO_LCD LCD_CURSOR_OFF
1108
1109          RJMP NO_ACCION_TIEMPO
1110
1111      CHEQUEAR_BOTON_CANCELAR_TIEMPO:
1112      SBRS BOTONES_TECLADO, BOTON_CANCELAR
1113      RJMP NO_ACCION_TIEMPO
1114
1115          LDI ESTADOS_LCD, VOLVER_MENU_PRINCIPAL
1116  /*      SBR ESTADOS_LCD, (1<<MENU_PRINCIPAL_LCD)
1117          SBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)
1118          SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)*/
1119          COMANDO_LCD LCD_CURSOR_OFF
1120
1121
1122      NO_ACCION_TIEMPO:
1123
1124      STS MENU_TIEMPO_TOTAL_POSICION_CURSOR, R17
1125      POP R17
1126
1127  RET
1128  ;
      ***************************************************************************************************
1129  ACCION_BOTON_CONF_UMBRAL:
1130
1131      PUSH R17
1132      LDS R17, MENU_UMBRAL_POSICION_CURSOR                    ; Se carga el dato de la posición
             del cursor
1133
1134      SBRS BOTONES_TECLADO, BOTON_IZQUIERDA
1135      RJMP CHEQUEAR_BOTON_DERECHO_UMBRAL
1136
1137          ; Se chequea donde está el cursor, y donde debería estar ahora
```

```
1138        CPI R17, CURSOR_POS_5_UMBRAL                    ; Si me encuentro en la
                posición más significativa, entonces no hago nada, ya que no puedo mover cursor
1139        BRNE TIEMPO_INCREMENTAR_CURSOR_UMBRAL
1140        RJMP NO_ACCION_UMBRAL
1141
1142        TIEMPO_INCREMENTAR_CURSOR_UMBRAL:
1143        COMANDO_LCD LCD_SHIFTEAR_CURSOR_IZQUIERDA       ; Si no, incremento el valor
                de la posición del cursor.
1144        INC R17
1145        RJMP NO_ACCION_UMBRAL
1146
1147    CHEQUEAR_BOTON_DERECHO_UMBRAL:                      ; Idem botón izquierdo pero
            para el botón derecho
1148
1149    SBRS BOTONES_TECLADO, BOTON_DERECHO
1150    RJMP CHEQUEAR_BOTON_SUPERIOR_UMBRAL
1151
1152        CPI R17, CURSOR_POS_1_UMBRAL
1153        BRNE UMBRAL_DECREMENTAR_CURSOR
1154        RJMP NO_ACCION_UMBRAL
1155
1156        UMBRAL_DECREMENTAR_CURSOR:
1157        COMANDO_LCD LCD_SHIFTEAR_CURSOR_DERECHA
1158        DEC R17
1159        RJMP NO_ACCION_UMBRAL
1160
1161    CHEQUEAR_BOTON_SUPERIOR_UMBRAL:                     ; Se utiliza para
            incrementar el total del valor de la cantidad de ventanas a medir
1162    SBRS BOTONES_TECLADO, BOTON_SUPERIOR
1163    RJMP CHEQUEAR_BOTON_INFERIOR_UMBRAL
1164
1165        PUSH ZL
1166        PUSH ZH
1167
1168        LDI ZL, LOW(MENU_UMBRAL_ASCII)                  ; Se levanta el ASCII de la
                cantidad de ventanas a medir, para modificarlo según corresponda
1169        LDI ZH, HIGH(MENU_UMBRAL_ASCII)
1170
1171        CPI R17, CURSOR_POS_1_UMBRAL                    ; Chequeo si se está
                en la posición menos significativa
1172        BRNE CHEQUEO_POS_2_UMBRAL_ARRIBA
1173
1174            ADIW ZH:ZL, CURSOR_POS_5_UMBRAL -1          ; Se corre el cursor
                    hasta la posición menos significativa
1175            RJMP BOTON_SUPERIOR_CAMBIAR_VALOR_UMBRAL
1176
1177
1178        CHEQUEO_POS_2_UMBRAL_ARRIBA:
1179
1180        CPI R17, CURSOR_POS_2_UMBRAL
1181        BRNE CHEQUEO_POS_3_UMBRAL_ARRIBA
1182
1183            ADIW ZH:ZL, CURSOR_POS_4_UMBRAL -1          ; Se corre el cursor
                    hasta la posición de las decenas
1184            RJMP BOTON_SUPERIOR_CAMBIAR_VALOR_UMBRAL
1185
1186        CHEQUEO_POS_3_UMBRAL_ARRIBA:
1187
1188        CPI R17, CURSOR_POS_3_UMBRAL
1189        BRNE CHEQUEO_POS_4_UMBRAL_ARRIBA
1190
1191            ADIW ZH:ZL, CURSOR_POS_3_UMBRAL -1          ; Se corre el cursor
                    hasta la posición de las centenas
1192            RJMP BOTON_SUPERIOR_CAMBIAR_VALOR_UMBRAL
1193
1194        CHEQUEO_POS_4_UMBRAL_ARRIBA:
```

```
1195
1196            CPI R17, CURSOR_POS_4_UMBRAL
1197            BRNE CHEQUEO_POS_5_UMBRAL_ARRIBA
1198
1199                ADIW ZH:ZL, CURSOR_POS_2_UMBRAL -1                          ; Se corre el cursor
                        hasta la posición de las unidades de mil
1200                RJMP BOTON_SUPERIOR_CAMBIAR_VALOR_UMBRAL
1201
1202            CHEQUEO_POS_5_UMBRAL_ARRIBA:
1203
1204            CPI R17, CURSOR_POS_5_UMBRAL
1205            BRNE BOTON_SUPERIOR_UMBRAL_SEGUIR
1206
1207                ADIW ZH:ZL, CURSOR_POS_1_UMBRAL -1                          ; Se corre el cursor
                        hasta la posición de las decenas de mil
1208                RJMP BOTON_SUPERIOR_CAMBIAR_VALOR_UMBRAL
1209
1210            BOTON_SUPERIOR_CAMBIAR_VALOR_UMBRAL:                            ; Por último, se
                    modifica el valor correspondiente y se guarda
1211
1212                PUSH R18
1213                LD R18, Z                                                   ; Se carga el
                        caracter ascii correspondiente
1214                CPI R18, '9'                                                ; Si el caracter es
                        un 9 ascii, no se puede incrementar más
1215                BREQ BOTON_SUPERIOR_UMBRAL_0
1216
1217                INC R18
1218
1219                RJMP BOTON_SUPERIOR_UMBRAL_SEGUIR
1220
1221                BOTON_SUPERIOR_UMBRAL_0:
1222                    LDI R18, '0'                                            ; Se carga un 0 ascii
1223
1224            BOTON_SUPERIOR_UMBRAL_SEGUIR:
1225                ST Z, R18                                                   ; Se guarda el dato
                        incrementado
1226                POP R18                                                     ; Se recuperan los
                        datos que había en los registros originalmente
1227                POP ZH
1228                POP ZL
1229
1230                SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)              ; Se habilita la
                        escritura en el segundo renglón
1231                RJMP NO_ACCION_UMBRAL
1232
1233        CHEQUEAR_BOTON_INFERIOR_UMBRAL:
1234        SBRS BOTONES_TECLADO, BOTON_INFERIOR
1235        RJMP CHEQUEAR_BOTON_OK_UMBRAL
1236
1237            PUSH ZL
1238            PUSH ZH
1239
1240            LDI ZL, LOW(MENU_UMBRAL_ASCII)                                  ; Se levanta el ASCII de la
                    cantidad de ventanas a medir, para modificarlo según corresponda
1241            LDI ZH, HIGH(MENU_UMBRAL_ASCII)
1242
1243            CPI R17, CURSOR_POS_1_UMBRAL                                    ; Chequeo si se está
                    en la posición menos significativa
1244            BRNE CHEQUEO_POS_2_UMBRAL_ABAJO
1245
1246                ADIW ZH:ZL, CURSOR_POS_5_UMBRAL -1                          ; Se corre el cursor
                        hasta la posición menos significativa
1247                RJMP BOTON_INFERIOR_CAMBIAR_VALOR_UMBRAL
1248
1249
```

```
1250        CHEQUEO_POS_2_UMBRAL_ABAJO:
1251
1252        CPI R17, CURSOR_POS_2_UMBRAL
1253        BRNE CHEQUEO_POS_3_UMBRAL_ABAJO
1254
1255            ADIW ZH:ZL, CURSOR_POS_4_UMBRAL -1                    ; Se corre el cursor
                    hasta la posición de las decenas
1256            RJMP BOTON_INFERIOR_CAMBIAR_VALOR_UMBRAL
1257
1258        CHEQUEO_POS_3_UMBRAL_ABAJO:
1259
1260        CPI R17, CURSOR_POS_3_UMBRAL
1261        BRNE CHEQUEO_POS_4_UMBRAL_ABAJO
1262
1263            ADIW ZH:ZL, CURSOR_POS_3_UMBRAL -1                    ; Se corre el cursor
                    hasta la posición de las centenas
1264            RJMP BOTON_INFERIOR_CAMBIAR_VALOR_UMBRAL
1265
1266        CHEQUEO_POS_4_UMBRAL_ABAJO:
1267
1268        CPI R17, CURSOR_POS_4_UMBRAL
1269        BRNE CHEQUEO_POS_5_UMBRAL_ABAJO
1270
1271            ADIW ZH:ZL, CURSOR_POS_2_UMBRAL -1                    ; Se corre el cursor
                    hasta la posición de las unidades de mil
1272            RJMP BOTON_INFERIOR_CAMBIAR_VALOR_UMBRAL
1273
1274        CHEQUEO_POS_5_UMBRAL_ABAJO:
1275
1276        CPI R17, CURSOR_POS_5_UMBRAL
1277        BRNE BOTON_INFERIOR_UMBRAL_SEGUIR
1278
1279            ADIW ZH:ZL, CURSOR_POS_1_UMBRAL -1                    ; Se corre el cursor
                    hasta la posición de las decenas de mil
1280            RJMP BOTON_INFERIOR_CAMBIAR_VALOR_UMBRAL
1281
1282        BOTON_INFERIOR_CAMBIAR_VALOR_UMBRAL:                      ; Por último, se
               modifica el valor correspondiente y se guarda
1283
1284            PUSH R18
1285            LD R18, Z                                             ; Se carga el
                    caracter ascii correspondiente
1286            CPI R18, '0'                                          ; Si el caracter es
                    un 0 ascii, no se puede decrementar más
1287            BREQ BOTON_INFERIOR_UMBRAL_9
1288
1289            DEC R18
1290
1291            RJMP BOTON_INFERIOR_UMBRAL_SEGUIR
1292
1293            BOTON_INFERIOR_UMBRAL_9:
1294                LDI R18, '9'                                      ; Se carga un 9 ascii
1295
1296        BOTON_INFERIOR_UMBRAL_SEGUIR:
1297            ST Z, R18                                             ; Se guarda el dato
                    incrementado
1298            POP R18                                               ; Se recuperan los
                    datos que había en los registros originalmente
1299            POP ZH
1300            POP ZL
1301
1302            SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)        ; Se habilita la
                    escritura en el segundo renglón
1303            RJMP NO_ACCION_UMBRAL
1304
1305
```

```
1306     CHEQUEAR_BOTON_OK_UMBRAL:
1307     SBRS BOTONES_TECLADO, BOTON_OK
1308     RJMP CHEQUEAR_BOTON_CANCELAR_UMBRAL
1309
1310         PUSH XH
1311         PUSH XL
1312         PUSH YH
1313         PUSH YL
1314         PUSH R16
1315
1316         LDI XL, LOW(MENU_UMBRAL_ASCII)                          ; Se carga el puntero de
                donde se encuentra guardado del valor modificado
1317         LDI XH, HIGH(MENU_UMBRAL_ASCII)
1318         LDI YL, LOW(REGISTRO_UMBRAL)                              ; Se carga el puntero
                donde se quiere guardar el valor convertido a hexa
1319         LDI YH, HIGH(REGISTRO_UMBRAL)
1320         LDI R16, MAX_NUM_DIGITOS_5                                  ; Se carga cuantos
                digitos se quiere convertir a hexa
1321         CALL TRANSFORMAR_DE_ASCII_A_BINARIO_Y_GUARDAR_5_DIGITOS     ; Se guarda el valor
                de la cantidad de ventanas a medir, modificado
1322
1323         POP R16
1324         POP YL
1325         POP YH
1326         POP XL
1327         POP XH
1328
1329         LDI ESTADOS_LCD, VOLVER_MENU_PRINCIPAL
1330
1331 /*      SBR ESTADOS_LCD, (1<<MENU_PRINCIPAL_LCD)                    ; Se configura para
    regresar al menpu principal
1332         SBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)
1333         SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)*/
1334         COMANDO_LCD LCD_CURSOR_OFF
1335
1336         RJMP NO_ACCION_UMBRAL
1337
1338     CHEQUEAR_BOTON_CANCELAR_UMBRAL:
1339     SBRS BOTONES_TECLADO, BOTON_CANCELAR
1340     RJMP NO_ACCION_UMBRAL
1341
1342         LDI ESTADOS_LCD, VOLVER_MENU_PRINCIPAL
1343 /*      SBR ESTADOS_LCD, (1<<MENU_PRINCIPAL_LCD)
1344         SBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)
1345         SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)*/
1346         COMANDO_LCD LCD_CURSOR_OFF
1347
1348
1349     NO_ACCION_UMBRAL:
1350
1351     STS MENU_UMBRAL_POSICION_CURSOR, R17
1352     POP R17
1353
1354
1355
1356 RET
1357 ;
    ********************************************************************************************

1358 ACCION_BOTON_CONF_BUZZER:
1359
1360     SBRS BOTONES_TECLADO, BOTON_IZQUIERDA
1361     RJMP CHEQUEAR_BOTON_DERECHO_BUZZER
1362
1363         SBRS ESTADOS_LCD, BUZZER_LCD_SI
1364         RJMP BOTON_IZQUIERDA_BUZZER_NO
```

```
1365
1366            COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el segundo
                    renglón
1367            BORRAR_RENGLON                                          ; Se limpia el segundo
                    renglón
1368            COMANDO_LCD LCD_CAMBIAR_RENGLON
1369
1370            CADENA_FLASH_LCD MENSAJE_BUZZER_NO                      ; Se escribe el mensaje
                    de 1000ms
1371
1372            CBR ESTADOS_LCD, (1<<BUZZER_LCD_SI)
1373
1374            RJMP NO_ACCION_BUZZER
1375
1376        BOTON_IZQUIERDA_BUZZER_NO:
1377        SBRC ESTADOS_LCD, BUZZER_LCD_SI
1378        RJMP CHEQUEAR_BOTON_DERECHO_BUZZER
1379
1380            COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el segundo
                    renglón
1381            BORRAR_RENGLON                                          ; Se limpia el segundo
                    renglón
1382            COMANDO_LCD LCD_CAMBIAR_RENGLON
1383
1384            CADENA_FLASH_LCD MENSAJE_BUZZER_SI                      ; Se escribe el mensaje
                    de 1_ms
1385
1386            SBR ESTADOS_LCD, (1<<BUZZER_LCD_SI)
1387
1388            RJMP NO_ACCION_BUZZER
1389
1390    CHEQUEAR_BOTON_DERECHO_BUZZER:
1391    SBRS BOTONES_TECLADO, BOTON_DERECHO
1392    RJMP CHEQUEAR_BOTON_SUPERIOR_BUZZER
1393
1394        SBRS ESTADOS_LCD, BUZZER_LCD_SI                     ; Si no estoy en la posición de
                mostrar 1 ms, salteo esta parte
1395        RJMP BOTON_DERECHA_BUZZER_NO                            ; Sigo comprobando si estoy
                en la posición de mostrar 10ms
1396
1397            COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el segundo
                    renglón
1398            BORRAR_RENGLON                                          ; Se limpia el segundo
                    renglón
1399            COMANDO_LCD LCD_CAMBIAR_RENGLON
1400
1401            CADENA_FLASH_LCD MENSAJE_BUZZER_NO                          ; Se escribe el
                    mensaje de 10 ms
1402
1403            CBR ESTADOS_LCD, (1<<BUZZER_LCD_SI)
1404
1405            RJMP NO_ACCION_BUZZER
1406
1407        BOTON_DERECHA_BUZZER_NO:
1408        SBRC ESTADOS_LCD, BUZZER_LCD_SI
1409        RJMP CHEQUEAR_BOTON_SUPERIOR_BUZZER
1410
1411            COMANDO_LCD LCD_CAMBIAR_RENGLON                         ; Me paro en el segundo
                    renglón
1412            BORRAR_RENGLON                                          ; Se limpia el segundo
                    renglón
1413            COMANDO_LCD LCD_CAMBIAR_RENGLON
1414
1415            CADENA_FLASH_LCD MENSAJE_BUZZER_SI                          ; Se escribe el
                    mensaje de 100 ms
1416
```

```
1417            SBR ESTADOS_LCD, (1<<BUZZER_LCD_SI)
1418
1419            RJMP NO_ACCION_BUZZER
1420
1421    CHEQUEAR_BOTON_SUPERIOR_BUZZER:
1422    SBRS BOTONES_TECLADO, BOTON_SUPERIOR
1423    RJMP CHEQUEAR_BOTON_INFERIOR_BUZZER
1424
1425            RJMP NO_ACCION_BUZZER
1426
1427    CHEQUEAR_BOTON_INFERIOR_BUZZER:
1428    SBRS BOTONES_TECLADO, BOTON_INFERIOR
1429    RJMP CHEQUEAR_BOTON_OK_BUZZER
1430
1431        RJMP NO_ACCION_BUZZER
1432
1433    CHEQUEAR_BOTON_OK_BUZZER:
1434    SBRS BOTONES_TECLADO, BOTON_OK
1435    RJMP CHEQUEAR_BOTON_CANCELAR_BUZZER
1436
1437    PUSH R16
1438    LDS R16, REGISTRO_CONF_GENERAL
1439
1440        BUZZER_GUARDAR_SI:
1441
1442        SBRS ESTADOS_LCD, BUZZER_LCD_SI                    ; Si el usuario está visulizando
            1 ms, se guarda dicha configuración
1443        RJMP BUZZER_GUARDAR_NO
1444
1445            SBR R16, (1<<BIT_SENAL_SONORA)
1446            RJMP BUZZER_GUARDAR_SEGUIR
1447
1448        BUZZER_GUARDAR_NO:                                 ; Si el usuario está
            visulizando 10 ms, se guarda dicha configuración
1449        SBRC ESTADOS_LCD, BUZZER_LCD_SI
1450        RJMP NO_ACCION_BUZZER
1451
1452            CBR R16, (1<<BIT_SENAL_SONORA)
1453            RJMP BUZZER_GUARDAR_SEGUIR
1454
1455        BUZZER_GUARDAR_SEGUIR:
1456        STS REGISTRO_CONF_GENERAL, R16
1457        POP R16
1458        LDI ESTADOS_LCD, VOLVER_MENU_PRINCIPAL
1459 /*     SBR ESTADOS_LCD, (1<<MENU_PRINCIPAL_LCD)           ; Configuro que vuelvo al
     menú principal
1460        SBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)
1461        SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)     ; Indico que puedo
            escribir nuevamente en el segundo renglón*/
1462        RJMP NO_ACCION_BUZZER
1463
1464    CHEQUEAR_BOTON_CANCELAR_BUZZER:
1465    SBRS BOTONES_TECLADO, BOTON_CANCELAR
1466    RJMP NO_ACCION_BUZZER
1467
1468        LDI ESTADOS_LCD, VOLVER_MENU_PRINCIPAL
1469 /*     SBR ESTADOS_LCD, (1<<MENU_PRINCIPAL_LCD)           ; Configuro que
    vuelvo al menú principal
1470        SBR ESTADOS_LCD, (1<<ESCRIBIR_PRIMER_RENGLON)
1471        SBR ESTADOS_LCD, (1<<ESCRIBIR_SEGUNDO_RENGLON)     ; Indico que puedo
            escribir nuevamente en el segundo renglón*/
1472        RJMP NO_ACCION_BUZZER
1473
1474    NO_ACCION_BUZZER:
1475
1476 RET
```

```
1477
1478
1479
1480
1481 ;*****************************************************************************************************
1482 ;*****************************************************************************************************
1483 ; Mensajes que se utilizan durante la interfaz LCD
1484 ; ================================================
1485 MENSAJE_MENU_PRINCIPAL: .db "Menu principal", 0
1486 MENSAJE_MENU_PRINCIPAL_UMBRAL: .db FLECHA_IZQUIERDA_ASCII, " Conf umbral  ",
        FLECHA_DERECHA_ASCII, 0
1487 MENSAJE_MENU_PRINCIPAL_VENTANA: .db FLECHA_IZQUIERDA_ASCII, " Conf ventana ",
        FLECHA_DERECHA_ASCII, 0
1488 MENSAJE_MENU_PRINCIPAL_DURACION: .db FLECHA_IZQUIERDA_ASCII, "Conf duracion ",
        FLECHA_DERECHA_ASCII, 0
1489 MENSAJE_MENU_PRINCIPAL_BUZZER: .db FLECHA_IZQUIERDA_ASCII, "Config. Buzz ",
        FLECHA_DERECHA_ASCII, 0
1490 MENSAJE_MENU_PRINCIPAL_MEDIR: .db FLECHA_IZQUIERDA_ASCII, "    Medir     ",
        FLECHA_DERECHA_ASCII, 0
1491 ; ================================================
1492 MENSAJE_MENU_UMBRAL: .db "Umbral ?", 0
1493 ; ================================================
1494 MENSAJE_MENU_VENTANA: .db "Ventana ?", 0
1495 MENSAJE_1_ms: .db FLECHA_IZQUIERDA_ASCII,"     1 ms     ",FLECHA_DERECHA_ASCII, 0
1496 MENSAJE_10_ms: .db FLECHA_IZQUIERDA_ASCII,"     10 ms    ",FLECHA_DERECHA_ASCII, 0
1497 MENSAJE_100_ms: .db FLECHA_IZQUIERDA_ASCII,"    100 ms    ",FLECHA_DERECHA_ASCII, 0
1498 MENSAJE_1000_ms: .db FLECHA_IZQUIERDA_ASCII,"    1000 ms   ",FLECHA_DERECHA_ASCII, 0
1499 ; ================================================
1500 MENSAJE_MENU_TIEMPO: .db "Repetir ventana?", 0
1501 ; ================================================
1502 MENSAJE_MENU_BUZZER: .db "Conf. buzzer", 0
1503 MENSAJE_BUZZER_NO: .db FLECHA_IZQUIERDA_ASCII,"  Buzzer des. ",FLECHA_DERECHA_ASCII, 0
1504 MENSAJE_BUZZER_SI: .db FLECHA_IZQUIERDA_ASCII,"  Buzzer act. ",FLECHA_DERECHA_ASCII, 0
1505 ; ================================================
1506 MENSAJE_MENU_MIDIENDO: .db "Midiendo...", 0
1507 ; ================================================
```

maquina_estados_mediciones.asm

```
1  ; Descripcion: maquina de estados perteneciente a la gestion
2  ; de las mediciones
3  ; Utiliza los registros EVENTO y ESTADO descriptos en main.asm
4
5
6  ;*****************************************************************************************************
7
8
9  MAQUINA_ESTADOS_MEDICIONES:
10
11
12     SBRC ESTADO, EST_OSCIOSO_MEDICION
13     RJMP ESTADO_OSCIOSO_MEDICION
14
15     SBRC ESTADO, EST_MEDIR_DEVOLVER_TIEMPOS
16     RJMP ESTADO_MEDIR_DEVOLVER_TIEMPOS
17
18     SBRC ESTADO, EST_MEDIR_DEVOLVER_TOTAL
19     RJMP ESTADO_MEDIR_DEVOLVER_TOTAL
20
21     ; Pasar a la maquina de estado de UART
```

```
22      RET
23
24
25  ;***********************************************************************************************
26
27
28  ESTADO_OSCIOSO_MEDICION:
29
30      ; Terminar de enviar la informacion del registro de desplazamiento si es que quedaron
            datos pendientes
31      SBRS EVENTO, ENVIANDO_DATOS_UART
32      CALL QUITAR_PULSO_REGISTRO_DESP_Y_ENVIAR_UART
33
34      ; Si se ha recibido un comando (por UART o teclado) de comenzar
35      ; una medicion, entonces iniciar la medicion y
36      ; cambiar al estado ESTADO_MEDIR
37
38      SBRS EVENTO, COMENZAR_MEDICION
39      RJMP _FIN_ESTADO_OSCIOSO_MEDICION
40      ; ================================================
41
42      CALL INICIAR_MEDICION                                          ;FUNCION A
            EJECUTAR PARA COMENZAR UNA MEDICION
43
44      ; ================================================
45      CBR EVENTO, (1<<DETENER_MEDICION)
46      CBR ESTADO, (1<<EST_OSCIOSO_MEDICION)
47
48      ; No hay pulsos almacenados en el registro de desplazamiento
49      CLR CANT_CARACTERES_GUARDADOS
50
51      ; Chequear si la configuracion indica que se deben enviar el total de los pulsos
52      ; o sus tiempos por UART y cambiar al estado correspondiente
53      LDS R16, REGISTRO_CONF_GENERAL
54      SBRC R16, BIT_ENVIAR_TIEMPO_PULSOS
55      RJMP _CONF_ESTADO_ENVIAR_TIEMPO_PULSOS
56      SBR ESTADO, (1<<EST_MEDIR_DEVOLVER_TOTAL)
57
58      ;CLR R4
59      CALL INICIAR_TIMER_1                                ;Se enciende el TIMER 1, para
            comenzar a medir
60      RJMP _FIN_ESTADO_OSCIOSO_MEDICION
61
62  _CONF_ESTADO_ENVIAR_TIEMPO_PULSOS:
63      SBR ESTADO, (1<<EST_MEDIR_DEVOLVER_TIEMPOS)
64      CALL INICIAR_TIMER_1                                ;Se enciende el TIMER 1, para
            comenzar a medir
65      RJMP _FIN_ESTADO_OSCIOSO_MEDICION
66
67  _FIN_ESTADO_OSCIOSO_MEDICION:
68      RET
69
70  ;***********************************************************************************************
71  ESTADO_MEDIR_DEVOLVER_TIEMPOS:
72      CBR EVENTO, (1<<COMENZAR_MEDICION)
73
74      ; Chequear si se ha decrementado el multiplicador de ventana, y si es asi, enviar un
            indicador a la PC
75      LDS R16, EVENTO2
76      SBRS R16, BIT_MUL_VENTANA_DEC
77      RJMP _ESTADO_MEDIR_CARGAR_REGISTRO_DESPLAZAMIENTO
78
```

```
79      LDI ZL, LOW(MENSAJE_DEC_MUL<<1)
80      LDI ZH, HIGH(MENSAJE_DEC_MUL<<1)
81      CALL CARGAR_CADENA_REGISTRO_DESPLAZAMIENTO
82
83      LDS R16, EVENTO2
84      CBR R16, (1<<BIT_MUL_VENTANA_DEC)
85      STS EVENTO2, R16
86
87  _ESTADO_MEDIR_CARGAR_REGISTRO_DESPLAZAMIENTO:
88      ; Si se recibio el tiempo de un pulso,
89      ; entonces leer el registro donde se almacena dicho valor (ICR1)
90      SBRS EVENTO, PULSO_RECIBIDO
91      RJMP _ESTADO_MEDIR_ENVIAR_TIEMPO_PULSO
92
93      ; Limpiar el bit que indica que se recibio un pulso porque ya fue leido
94      CBR EVENTO, (1<<PULSO_RECIBIDO)
95
96      ; Leer el tiempo del pulso y cargarlo en el registro de desplazamiento
97      LDS R16, ICR1L
98      LDS R17, ICR1H
99
100     CALL CARGAR_PULSO_REGISTRO_DESPLAZAMIENTO
101
102
103 _ESTADO_MEDIR_ENVIAR_TIEMPO_PULSO:
104     ; Chequear si se estan enviando datos por UART. Si es asi, no cargar de nuevo el buffer y
105     ; esperar a la siguiente iteracion de la maquina
106     SBRS EVENTO, ENVIANDO_DATOS_UART
107     CALL QUITAR_PULSO_REGISTRO_DESP_Y_ENVIAR_UART
108     RJMP _ESTADO_MEDIR_CHEQUEAR_FIN_VENTANA
109
110
111 ESTADO_MEDIR_DEVOLVER_TOTAL:
112     CBR EVENTO, (1<<COMENZAR_MEDICION)
113     ; Si se ha recibido (por UART o por teclado) un comando de
114     ; comenzar una medición, entonces, una vez configurada esta,
115     ; se entra en este estado.
116
117 _ESTADO_MEDIR_CHEQUEAR_FIN_VENTANA:
118     ;SBI PORTB, 1
119     BRTC NO_SE_TERMINO_VENTANA                      ;Si el flag T está en 0, se
            continua con la ventana
120     CLT                                             ;Si no, es porque se terminó una
            ventana, y por una interrupción, se pone en 1
121
122
123     ; VENTANA TERMINADA:
124
125
126
127     LDS MUL_DE_VENTANA, MUL_DE_VENTANA_RAM          ;Se recupera el dato del
            multiplicador de ventana, para continuar con la siguiente ventana
128
129
130     CALL ACUMULAR_PULSOS_DETECTADOS                 ;Se guardan los pulsos que se
            contaron en la ventana, en RAM
131
132     ; Chequear umbral, y si se ha superado el valor establecido encender el LED/BUZZER
133     CALL VERIFICAR_UMBRAL
134
135     SBRS ESTADO, EST_MEDIR_DEVOLVER_TOTAL
136     RJMP _CONTINUAR_ESTADO_MEDIR_CHEQUEAR_FIN_VENTANA
137
138     MOV R16, CONTADOR_DE_PULSOS
139     CLR R17
140     CLR R29
```

```
141     CALL CONVERTIR_A_BINARIO_Y_ENVIAR_UART
142
143     RJMP _CONTINUAR_ESTADO_MEDIR_CHEQUEAR_FIN_VENTANA
144
145
146 _CONTINUAR_ESTADO_MEDIR_CHEQUEAR_FIN_VENTANA:
147
148     CLR CONTADOR_DE_PULSOS                                  ;Se limpian los pulsos de la
            ventana anterior
149
150     ;SBR EVENTO, (1<<FIN_VENTANA)                           ;Se enciende el flag de fin de
            ventana en ESTADO, para indicarle a la UART que debe enviar los datos medidos
151     ; TESTEO DE SI SE TERMINÓ DE MEDIR
152
153     PUSH R16
154     DEC VENTANAS_A_MEDIR_LOW
155     BRNE NO_ES_CERO
156     LDI R16, 0x00
157     CP VENTANAS_A_MEDIR_MID, R16
158     BRNE NO_ES_CERO
159     CP VENTANAS_A_MEDIR_HIGH, R16
160     BRNE NO_ES_CERO
161
162 CERO:
163     CALL APAGAR_TIMER_1                                     ;Finalizadas la medición total,
            se apaga el timer 1.
164     ;CBI PORTB, 1
165     POP R16                                                 ;Se apaga el LED para indicar fin
             de medición
166     LDI ESTADOS_LCD, MEDICION_FINALIZADA_LCD
167     RJMP MOSTRAR_PROMEDIO
168 NO_ES_CERO:
169     LDI R16, 0xFF
170     CP VENTANAS_A_MEDIR_LOW, R16
171     BRNE LISTO_COMPARACION
172     DEC VENTANAS_A_MEDIR_MID
173     CP VENTANAS_A_MEDIR_MID, R16
174     BRNE LISTO_COMPARACION
175     DEC VENTANAS_A_MEDIR_HIGH
176 LISTO_COMPARACION:
177
178     POP R16
179
180 NO_SE_TERMINO_VENTANA:
181
182
183
184     ; Si se ha recibido un comando (por UART o teclado) de abortar
185     ; una medicion, entonces abortar y
186     ; cambiar al estado ESTADO_OSCIOSO_MEDICION
187     SBRC EVENTO, DETENER_MEDICION
188     RJMP _ESTADO_MEDIR_ABORTAR
189
190     RET
191
192 ;
    ***************************************************************************************************
193 ;
    ***************************************************************************************************
194
195 MOSTRAR_PROMEDIO:
196     ; Se muestra el promedio por pantalla al usuario
197     COMANDO_LCD LCD_HOME_SCREEN         ; Se posiciona en el extremo superior izquierdo
198
```

```
199     BORRAR_RENGLON                      ; Se borra el renglon
200
201     COMANDO_LCD LCD_HOME_SCREEN         ; Se posiciona en el extremos superior izquierdo
202
203     CADENA_FLASH_LCD MENSAJE_PROMEDIO_LCD
204
205
206     LDI ZL, LOW(CONTADOR_DE_PULSOS_RAM)             ;Se carga el puntero a donde se guardan
            los pulsos
207     LDI ZH, HIGH(CONTADOR_DE_PULSOS_RAM)
208     LDI XL, LOW(VENTANAS_A_MEDIR_RAM)               ;Se carga el puntero a donde se guardan
            las ventanas a medir
209     LDI XH, HIGH(VENTANAS_A_MEDIR_RAM)
210     CALL PROMEDIO                                   ;Se realiza elpromedio y se guarda en
            PROMEDIO_RAM [16 bits]
211     LDS R16, PROMEDIO_RAM                           ;Se cargan los bytes del promedio en dos
            registros para hacer promedio
212     LDS R17, PROMEDIO_RAM+1
213     LDS R29, PROMEDIO_RAM+2
214     LDI XL, LOW(NUMERO_ASCII)
215     LDI XH, HIGH(NUMERO_ASCII)
216
217     CALL DEC_TO_ASCII_24_BITS
218
219     COMANDO_LCD LCD_CAMBIAR_RENGLON
220     BORRAR_RENGLON
221     COMANDO_LCD LCD_CAMBIAR_RENGLON
222
223     LDI ZL, LOW(NUMERO_ASCII)                       ;Se carga el puntero a NUMERO_ASCII, que
            es donde se guardo el promedio
224     LDI ZH, HIGH(NUMERO_ASCII)
225     CALL STRING_WRT_LCD                             ;Se escriben los numeros en el LCD
226     ; ********************************************************************************
227
228     ; ================================================
229     CBR ESTADO, (1<<EST_MEDIR_DEVOLVER_TIEMPOS)
230     CBR ESTADO, (1<<EST_MEDIR_DEVOLVER_TOTAL)
231     SBR ESTADO, (1<<EST_OSCIOSO_MEDICION)
232
233     ; Apagar LED/buzzer si es que se encuentra activado
234     CBI PORTB, PIN_PORTB_BUZZER ; Apagar buzzer
235     CBI PORTB, PIN_PORTB_LED ; Apagar LED
236
237     ; Enviar codigo avisando que se termino una medicion
238     LDI R18, LOW(MENSAJE_FIN_MEDICION)
239     LDI R19, HIGH(MENSAJE_FIN_MEDICION)
240     CALL CARGAR_BUFFER
241     CALL ACTIVAR_INT_TX_UDRE0
242
243     RET
244
245 ;
    ********************************************************************************************************
246
247 _ESTADO_MEDIR_ABORTAR:
248     ; Limpiar el evento asociado a abortar una medicion
249     CBR EVENTO, (1<<DETENER_MEDICION)
250
251     CALL APAGAR_TIMER_1                             ; Se apaga el timer 1
252
253     ;CBI PORTB, 1
254
255     CBR ESTADO, (1<<EST_MEDIR_DEVOLVER_TIEMPOS)
256     CBR ESTADO, (1<<EST_MEDIR_DEVOLVER_TOTAL)
257     SBR ESTADO, (1<<EST_OSCIOSO_MEDICION)
```

```
258
259     ; Apagar LED/buzzer si es que se encuentra activado
260     CBI PORTB, PIN_PORTB_BUZZER ; Apagar buzzer
261     CBI PORTB, PIN_PORTB_LED ; Apagar LED
262
263     ; Enviar codigo avisando que se termino una medicion
264     LDI R18, LOW(MENSAJE_FIN_MEDICION)
265     LDI R19, HIGH(MENSAJE_FIN_MEDICION)
266     CALL CARGAR_BUFFER
267     CALL ACTIVAR_INT_TX_UDRE0
268
269     RET
```

maquina_estados_uart.asm

```
1  ; Descripcion: maquina de estados perteneciente a la gestion
2  ; de la comunicacion por UART
3  ; Utiliza los registros EVENTO y ESTADO descriptos en main.asm
4
5  MAQUINA_ESTADOS_UART:
6
7      ; Verificar estado oscioso
8      SBRC ESTADO, EST_OSCIOSO_UART
9      RJMP ESTADO_OSCIOSO_UART
10
11     ; Verificar estado de recepcion de cadena
12     SBRC ESTADO, EST_RECIBIENDO_CADENA
13     RJMP ESTADO_RECIBIENDO_CADENA
14
15     ; Verificar estado de interpretacion de cadena
16     SBRC ESTADO, EST_INTERPRETANDO_CADENA
17     RJMP ESTADO_INTERPRETANDO_CADENA
18
19     ; Verificar estado de error
20     SBRC ESTADO, EST_ERROR_UART
21     RJMP ESTADO_ERROR
22
23     RET
24
25 ESTADO_OSCIOSO_UART:
26
27     ; Chequear si se recibio un caracter, y si es asi, ir al estado de recepcion de cadena
28     LDS R16, EVENTO2
29     SBRS R16, BIT_CARACTER_RECIBIDO
30     RET
31
32     ; Verificar si el teclado se encuentra en uso. Si es asi, enviar un error al usuario
33     ; indicando que el sistema se encuentra ocupado
34     CALL CHEQUEAR_TECLADO_EN_USO
35     BRCS _ESTADO_OSCIOSO_ERROR_TECLADO_EN_USO
36
37     ; Indicar que el sistema no se encuentra ocupado
38     LDS T0, EVENTO2
39     CBR T0, (1<<SISTEMA_OCUPADO)
40     STS EVENTO2, T0
41
42     CBR ESTADO, (1<<EST_OSCIOSO_UART)
43     SBR ESTADO, (1<<EST_RECIBIENDO_CADENA)
44
45     ; Encender timer0 y activar sus interrupciones por overflow
46     ; para corroborar cuando se produjo un timeout
47     LDS R16, MOTIVO_TIMER0
48     SBR R16, (1<<MOTIVO_TIMER0_UART) ; Indicar al timer el motivo de su ejecucion
49     STS MOTIVO_TIMER0, R16
50     CALL ENCENDER_TIMER_0
51
```

```
52      RET
53
54  _ESTADO_OSCIOSO_ERROR_TECLADO_EN_USO:
55      LDS T0, EVENTO2
56      CBR T0, (1<<BIT_CARACTER_RECIBIDO)
57      SBR T0, (1<<SISTEMA_OCUPADO)
58      STS EVENTO2, T0
59
60      ; Configurar puntero al comienzo del buffer de recepcion
61      LDI T0, LOW(BUFFER_RX)
62      MOV PTR_RX_L, T0
63      LDI T0, HIGH(BUFFER_RX)
64      MOV PTR_RX_H, T0
65      RET
66
67
68  ESTADO_RECIBIENDO_CADENA:
69      ; Si se ha terminado de recibir una cadena correctamente, ir al estado para interpretar
            la cadena
70      LDS R16, EVENTO2
71      SBRC R16, BIT_FIN_CADENA
72      RJMP _IR_A_ESTADO_INTERPRETAR_CADENA
73      ; Si se produjo un error por overflow del buffer o timeout en la recepcion de datos;
74      ; pasar al estado de error
75      SBRC EVENTO, TIMEOUT_UART
76      RJMP _IR_A_ESTADO_ERROR
77      SBRC EVENTO, OV_BUFFER_RX_UART
78      RJMP _IR_A_ESTADO_ERROR
79
80      RET
81
82  _IR_A_ESTADO_INTERPRETAR_CADENA:
83
84      ; Limpiar el bit de evento de timer0 que corresponde a la UART
85      ; Esto permite que otra funcionalidad, como el teclado, pueda usar el timer0
86      ; mediante otros eventos
87      LDS R16, MOTIVO_TIMER0
88      CBR R16, (1<<MOTIVO_TIMER0_UART)
89      STS MOTIVO_TIMER0, R16
90
91      CALL APAGAR_TIMER_0
92
93      CBR R16, (1<<BIT_CARACTER_RECIBIDO)
94      STS EVENTO2, R16
95
96      CBR R16, (1<<BIT_CARACTER_RECIBIDO)
97      STS EVENTO2, R16
98
99      CBR ESTADO, (1<<EST_RECIBIENDO_CADENA)
100     SBR ESTADO, (1<<EST_INTERPRETANDO_CADENA)
101     RET
102
103 _IR_A_ESTADO_ERROR:
104     ; Limpiar el bit de evento de timer0 que corresponde a la UART
105     ; Esto permite que otra funcionalidad, como el teclado, pueda usar el timer0
106     ; mediante otros eventos
107     LDS R16, MOTIVO_TIMER0
108     CBR R16, (1<<MOTIVO_TIMER0_UART)
109     STS MOTIVO_TIMER0, R16
110
111     CALL APAGAR_TIMER_0
112
113     CBR R16, (1<<BIT_CARACTER_RECIBIDO)
114     STS EVENTO2, R16
115
116     CBR ESTADO, (1<<EST_RECIBIENDO_CADENA)
```

```
117     SBR ESTADO, (1<<EST_ERROR_UART)
118     RET
119
120 ESTADO_INTERPRETANDO_CADENA:
121     CALL INTERPRETAR_CADENA_COMANDOS
122
123     CLR BYTES_RECIBIDOS
124
125     ; Volver a estado oscioso
126     CBR ESTADO, (1<<EST_INTERPRETANDO_CADENA)
127     SBR ESTADO, (1<<EST_OSCIOSO_UART)
128     RET
129
130 ; =========================
131 ; Estado de error: aqui se realiza un switch donde se interpreta cada error
132 ; y se toma una medida en consecuencia
133 ESTADO_ERROR:
134
135     SBRC EVENTO, OV_BUFFER_RX_UART
136     ;APAGAR_LED_ARDUINO
137
138     SBRC EVENTO, TIMEOUT_UART
139     RJMP _ERROR_TIMEOUT_RX
140
141     SBRC EVENTO, OV_BUFFER_RX_UART
142     RJMP _ERROR_OVERFLOW_RX
143
144     ; Volver a estado oscioso
145     CBR ESTADO, (1<<EST_ERROR_UART)
146     SBR ESTADO, (1<<EST_OSCIOSO_UART)
147     RET
148
149 _ERROR_TIMEOUT_RX:
150     CLR BYTES_RECIBIDOS
151
152     CALL ENVIAR_ERROR_TIMEOUT
153
154     ; Reiniciar el buffer de RX
155     CALL LIMPIAR_BUFFER_RX
156
157     ; Limpiar el bit asociado al evento de un timeout
158     CBR EVENTO, (1<<TIMEOUT_UART)
159
160     ; Volver a estado oscioso
161     CBR ESTADO, (1<<EST_ERROR_UART)
162     SBR ESTADO, (1<<EST_OSCIOSO_UART)
163     RET
164
165 _ERROR_OVERFLOW_RX:
166     ;TODO: ver como gestionar cuando hay un overflow y despues un \n
167
168     CALL LIMPIAR_BUFFER_RX
169
170     CALL ENVIAR_ERROR_OVERFLOW
171
172     ; Limpiar el bit asociado al evento de un overflow
173     CBR EVENTO, (1<<OV_BUFFER_RX_UART)
174
175     ; Volver a estado oscioso
176     CBR ESTADO, (1<<EST_ERROR_UART)
177     SBR ESTADO, (1<<EST_OSCIOSO_UART)
178     RET
```

comparador.asm

```
1 ; Descripcion: funciones para configurar el comparador
```

```
2
3  ; ====================================================================
4  ; ========================= Funciones ================================
5  ; ====================================================================
6
7  ; Descripcion: configurar comparador para utilizando para el teclado
8  ; Recibe: -
9  ; Devuelve: -
10 CONF_COMPARADOR:
11
12     CONF_LED_ARDUINO ; TODO: ES UNA LINEA DE PRUEBA. REMOVER
13
14     ; Activar multiplexor para elegir el puerto negativo del comparador
15     SET_BIT ADCSRB, ACME
16
17     ; Desactivar el ADC para poder realizar una comparacion recibiendo datos por el puerto
           PC0
18     CLEAR_BIT ADCSRA, ADEN
19
20     ; Configurer como puerto negativo al PC0
21     CLEAR_BIT ADMUX, MUX0
22     CLEAR_BIT ADMUX, MUX1
23     CLEAR_BIT ADMUX, MUX2
24     CLEAR_BIT ADMUX, MUX3
25
26     ; Configurar la interrupcion del comparador para que
27     ; salte cuando este saca un 1 logico
28     SET_BIT ACSR, ACIS1
29     CLEAR_BIT ACSR, ACIS0
30
31     ; Configurar el bandgap
32     SET_BIT ACSR, ACBG
33
34     ; Activar la interrupcion del comparador
35     SET_BIT ACSR, ACIE
36
37     RET
38
39 ; Descripcion: interrupcion del comparador
40 INTERRUPCION_COMPARADOR:
41
42     PUSH R16
43     IN  R16, SREG
44     PUSH R16    ; salva en la pila el estado actual de los flags (C, N, V y Z)
45
46     ; Chequear si el timer0 se encuentra en uso por parte de la UART. Si es asi, no realizar
47     ; ninguna accion para evitar una colision
48     CALL CHEQUEAR_UART_EN_USO
49     BRCS _RET_INTERRUPCION_COMPARADOR
50
51     ; Encender timer0 para esperar cierto tiempo hasta realizar la conversion
52     LDS R16, MOTIVO_TIMER0
53     SBR R16, (1<<MOTIVO_TIMER0_TECLADO)
54     STS MOTIVO_TIMER0, R16
55     CALL ENCENDER_TIMER_0
56
57     ; Encender el ADC (esto impedirá recibir datos por el puerto PC0 durante la conversion)
58     SET_BIT ADCSRA, ADEN
59
60     POP R16
61     OUT SREG, R16
62     POP R16 ; salva en la pila el estado actual de los flags (C, N, V y Z)
63
64 _RET_INTERRUPCION_COMPARADOR:
65     RETI
```

```
; Descripcion: funciones que graban registros en RAM
; para la configuracion del dispositivo

; =====================================================================
; ======================== Registros auxiliares =======================
; =====================================================================
.UNDEF T0
.UNDEF T1
.UNDEF T2
.UNDEF T3
.UNDEF T4
.UNDEF T5
.DEF T0 = R16
.DEF T1 = R17
.DEF T2 = R18
.DEF T3 = R19
.DEF T4 = R20
.DEF T5 = R21

; =====================================================================
; ========================== Constantes ===============================
; =====================================================================
; Define el maximo de numero de digitos que puede tener la cantidad
; de ventanas en las que medir, el tiempo de una ventana y el valor del umbral,
; en sus respectivas unidades
; TODO: dividir esta constante en una para cada parametro
.EQU MAX_NUM_DIGITOS = 4
.EQU MAX_NUM_DIGITOS_5 = 5

; Constantes que contienen la posicion de cada configuracion
; dentro del registro REGISTRO_CONF_GENERAL
.EQU BIT_SENAL_SONORA = 0
.EQU BIT_BLOQUEO_TECLADO = 1
.EQU BIT_ENVIAR_TIEMPO_PULSOS = 2
.EQU BIT_ACTIVAR_FUENTE = 3

; =====================================================================
; ========================= Registros en RAM ==========================
; =====================================================================
.dseg

; Registro para guardar la duracion de la ventana de tiempo
; Valores posibles:
; 0 --> 1ms
; 1 --> 10ms
; 2 --> 100ms
; 3 --> 1s
REGISTRO_VENTANA_TIEMPO: .BYTE 1

; Registro para guardar la cantidad de ventanas a medir
VENTANAS_A_MEDIR_RAM: .BYTE 3                                   ;[nibble alto]:[
    nibble intermedio]:[nibble bajo]
;VENTANAS_A_MEDIR_RAM: .BYTE 1

; Registro para guardar el valor del umbral de pulsos
; por ventana por encima del cual se enciende
; el led/buzzer
REGISTRO_UMBRAL: .BYTE 3                                        ;[nibble alto]:[
    nibble intermedio]:[nibble bajo]

; Registro para configuraciones generales:
; Un bit en '1' significa que el sistema esta activo
; Bits:
; 0 - Senal sonora
; 1 - Bloqueo del teclado
```

```
; 2 - Enviar el tiempo de llegada de cada pulso por UART
; 3 - Apagar fuente de alta tension del tubo del contador Geiger
REGISTRO_CONF_GENERAL: .BYTE 1

; ====================================================================
; ========================= Funciones ================================
; ====================================================================
.cseg

; Descripcion: configurar los siguientes registros con su valor
; por default:
; REGISTRO_VENTANA_TIEMPO = 2 ---> Duracion de la venta: 100ms
; VENTANAS_A_MEDIR_RAM = 10 ---> Cantidad de ventanas a medir antes de terminar la medicion: 10
; REGISTRO_UMBRAL = 10 ---> Cantidad de pulsos por ventana antes encender el led/buzzer: 10
CONFIGURAR_REGISTROS_DEFAULT:
    LDI T0, 3
    STS REGISTRO_VENTANA_TIEMPO, T0
    LDI T0, 10
    STS VENTANAS_A_MEDIR_RAM+1, T0
    LDI T0, 0
    STS VENTANAS_A_MEDIR_RAM, T0
    LDI T0, 1
    STS REGISTRO_UMBRAL+1, T0
    LDI T0, 0
    STS REGISTRO_UMBRAL, T0
    LDI T0, 0
    STS REGISTRO_CONF_GENERAL, T0
    RET



; Descripcion: parseo del tamano de ventana recibido mediante el comando
; CONFigure:WINDow N, donde N es el tamano de ventana
; Recibe: posicion de memoria del buffer de RX donde comienza N en el puntero X
CONFIGURAR_VENTANA_TIEMPO:
    PUSH T0
    PUSH T1

    CLR T0

    LD T1, X+
    CPI T1, '1'
    BRNE _RETORNAR_ERROR_CONF_COUNT_REGISTER

_BUCLE_CONF_COUNT_REGISTER:
    LD T1, X+
    CPI T1, '0'
    BREQ _DECREMENTAR_AUX_CONF_COUNT_REGISTER
    CPI T1, '\r' ; TODO: CAMBIAR POR UN \n
    BREQ _GUARDAR_REGISTRO_COUNT_REGISTER
    RJMP _RETORNAR_ERROR_CONF_COUNT_REGISTER

_DECREMENTAR_AUX_CONF_COUNT_REGISTER:
    CPI T0, 3; Si el numero ingresado es mayor a 1000, devolver un error
    BREQ _RETORNAR_ERROR_CONF_COUNT_REGISTER
    INC T0
    RJMP _BUCLE_CONF_COUNT_REGISTER

_RETORNAR_ERROR_CONF_COUNT_REGISTER:
    CALL ENVIAR_ERROR_PARSER

    POP T1
    POP T0
    RET

```

```
129 _GUARDAR_REGISTRO_COUNT_REGISTER:
130     STS REGISTRO_VENTANA_TIEMPO, T0
131 
132     POP T1
133     POP T0
134     RET
135 
136 ; ================================================================
137 ; Descripcion: lee el registro donde se encuentra definida la ventana de tiempo y devuelve su
        valor
138 ; Entradas: -
139 ; Salidas: -
140 DEVOLVER_VENTANA_TIEMPO:
141     PUSH T0
142     PUSH R18
143     PUSH_WORD XL,XH
144 
145     CALL IR_COMIENZO_BUFFER_TX
146 
147     LDI R18, '1'
148     CALL CARGAR_BUFFER_TX_CON_CARACTER
149 
150     LDS T0, REGISTRO_VENTANA_TIEMPO
151     CPI T0, 0
152 
153 _BUCLE_DEVOLVER_VENTANA_TIEMPO:
154     BREQ _ENVIAR_VENTANA_TIEMPO
155     LDI R18, '0'
156     CALL CARGAR_BUFFER_TX_CON_CARACTER
157     DEC T0
158     RJMP _BUCLE_DEVOLVER_VENTANA_TIEMPO
159 
160 _ENVIAR_VENTANA_TIEMPO:
161     LDI R18, '\n'
162     CALL CARGAR_BUFFER_TX_CON_CARACTER
163 
164     CALL ENVIAR_DATOS_UART
165 
166     POP_WORD XL,XH
167     POP R18
168     POP T0
169     RET
170 
171 ; ================================================================
172 ; Descripcion: activa el bit para iniciar una medicion en el registro EVENTO
173 ; Entradas: -
174 ; Salidas: -
175 CONFIGURAR_EVENTO_INICIAR_MEDICION:
176     SBR EVENTO, (1<<COMENZAR_MEDICION)
177     RET
178 
179 ; ================================================================
180 ; Descripcion: activa el bit para detener una medicion en el registro EVENTO
181 ; Entradas: -
182 ; Salidas: -
183 CONFIGURAR_EVENTO_DETENER_MEDICION:
184     SBR EVENTO, (1<<DETENER_MEDICION)
185     RET
186 
187 ; ================================================================
188 ; Descripcion: devuelve el valor del registro VENTANAS_A_MEDIR_RAM
189 ; Entradas: -
190 ; Salidas: -
191 DEVOLVER_NUMERO_VENTANAS:
192 
193     LDS R29, VENTANAS_A_MEDIR_RAM
```

```
194     LDS R17, VENTANAS_A_MEDIR_RAM+1
195     LDS R16, VENTANAS_A_MEDIR_RAM+2
196
197     CALL CONVERTIR_A_BINARIO_Y_ENVIAR_UART
198
199     RET
200
201 ; ================================================================
202 ; Descripcion: devuelve el valor del registro REGISTRO_UMBRAL
203 ; Entradas: -
204 ; Salidas: -
205 DEVOLVER_UMBRAL:
206
207     LDS R29, REGISTRO_UMBRAL
208     LDS R17, REGISTRO_UMBRAL+1
209     LDS R16, REGISTRO_UMBRAL+2
210
211     CALL CONVERTIR_A_BINARIO_Y_ENVIAR_UART
212
213     RET
214
215 ; ================================================================
216 ; Descripcion: indica si el tubo del contador se encuentra encendido
217 ; Entradas: -
218 ; Salidas: -
219 /*
220 DEVOLVER_CONF_ENCENDIDO_TUBO:
221
222     ; Cargar primera parte del mensaje en buffer
223     LDI R18, LOW(MENSAJE_ESTADO_FUENTE)
224     LDI R19, HIGH(MENSAJE_ESTADO_FUENTE)
225     CALL CARGAR_BUFFER
226
227     ; Chequear si el tubo se encuentra encendido
228     LDS R16, REGISTRO_CONF_GENERAL
229     SBRC R16, BIT_ACTIVAR_FUENTE
230     RJMP _FUENTE_APAGADA
231
232     CALL CARGAR_MENSAJE_ON_UART
233
234     CALL ENVIAR_DATOS_UART
235     RET
236
237 _FUENTE_APAGADA:
238
239     CALL CARGAR_MENSAJE_OFF_UART
240
241     CALL ENVIAR_DATOS_UART
242     RET
243 */
244
245 ; ================================================================
246 ; Descripcion: indica si una opcion en el registro REGISTRO_CONF_GENERAL esta activada
247 ; Entradas:
248 ; --> R18 y R19 con la posicion de memoria del mensaje correspondiente a la opcion de
      configuracion
249 ; --> R17 con el bit asociado a la configuracion en REGISTRO_CONF_GENERAL
250 ; Salidas: -
251 DEVOLVER_CONF_BIT:
252
253     ; Cargar primera parte del mensaje en buffer utilizando los registros R18 y R19
254     CALL CARGAR_BUFFER
255
256     ; Chequear si el tubo se encuentra encendido
257     LDS R16, REGISTRO_CONF_GENERAL
258     AND R16, R17
```

```
259     BRNE _OPCION_ACTIVADA
260
261     CALL CARGAR_MENSAJE_OFF_UART
262
263     CALL ENVIAR_DATOS_UART
264     RET
265
266 _OPCION_ACTIVADA:
267
268     CALL CARGAR_MENSAJE_ON_UART
269
270     CALL ENVIAR_DATOS_UART
271     RET
272
273 ; ================================================================
274 ; Descripcion: cargar el mensaje ON en el buffer de TX
275 ; Recibe: -
276 ; Salidas: -
277 CARGAR_MENSAJE_ON_UART:
278     LDI R18, 'O'
279     CALL CARGAR_BUFFER_TX_CON_CARACTER
280     LDI R18, 'N'
281     CALL CARGAR_BUFFER_TX_CON_CARACTER
282     LDI R18, '\n'
283     CALL CARGAR_BUFFER_TX_CON_CARACTER
284     RET
285
286 ; ================================================================
287 ; Descripcion: cargar el mensaje OFF en el buffer
288 ; Recibe: -
289 ; Salidas: -
290 CARGAR_MENSAJE_OFF_UART:
291     LDI R18, 'O'
292     CALL CARGAR_BUFFER_TX_CON_CARACTER
293     LDI R18, 'F'
294     CALL CARGAR_BUFFER_TX_CON_CARACTER
295     LDI R18, 'F'
296     CALL CARGAR_BUFFER_TX_CON_CARACTER
297     LDI R18, '\n'
298     CALL CARGAR_BUFFER_TX_CON_CARACTER
299     RET
300
301
302 ; ================================================================
303 ; Descripcion: devolver la configuracion de todo el dispositivo por UART
304 ; Recibe: -
305 ; Salidas: -
306 DEVOLVER_CONFIGURACION:
307
308     ; Numero de ventanas
309     LDI R18, LOW(MENSAJE_CANTIDAD_VENTANAS)
310     LDI R19, HIGH(MENSAJE_CANTIDAD_VENTANAS)
311     CALL CARGAR_BUFFER
312     CALL ENVIAR_DATOS_UART
313     SBR EVENTO, (1<<ENVIANDO_DATOS_UART)
314
315     CALL BUCLE_POLLING_DEVOLVER_CONFIGURACION
316
317     CALL DEVOLVER_NUMERO_VENTANAS
318     SBR EVENTO, (1<<ENVIANDO_DATOS_UART)
319
320     CALL BUCLE_POLLING_DEVOLVER_CONFIGURACION
321
322     ; Tiempo de la ventana
323     LDI R18, LOW(MENSAJE_VENTANA_DURACION)
324     LDI R19, HIGH(MENSAJE_VENTANA_DURACION)
```

```
325     CALL CARGAR_BUFFER
326     CALL ENVIAR_DATOS_UART
327     SBR EVENTO, (1<<ENVIANDO_DATOS_UART)
328 
329     CALL BUCLE_POLLING_DEVOLVER_CONFIGURACION
330 
331     CALL DEVOLVER_VENTANA_TIEMPO
332     SBR EVENTO, (1<<ENVIANDO_DATOS_UART)
333 
334     CALL BUCLE_POLLING_DEVOLVER_CONFIGURACION
335 
336     ; Valor del umbral
337     LDI R18, LOW(MENSAJE_TRIGGER_UMBRAL)
338     LDI R19, HIGH(MENSAJE_TRIGGER_UMBRAL)
339     CALL CARGAR_BUFFER
340     CALL ENVIAR_DATOS_UART
341     SBR EVENTO, (1<<ENVIANDO_DATOS_UART)
342 
343     CALL BUCLE_POLLING_DEVOLVER_CONFIGURACION
344 
345     CALL DEVOLVER_UMBRAL
346     SBR EVENTO, (1<<ENVIANDO_DATOS_UART)
347 
348     CALL BUCLE_POLLING_DEVOLVER_CONFIGURACION
349 
350     ; Encendido del tubo
351     LDI R18, LOW(MENSAJE_ESTADO_FUENTE)
352     LDI R19, HIGH(MENSAJE_ESTADO_FUENTE)
353     LDI R17, (1<<BIT_ACTIVAR_FUENTE)
354 
355     CALL DEVOLVER_CONF_BIT
356     SBR EVENTO, (1<<ENVIANDO_DATOS_UART)
357 
358     CALL BUCLE_POLLING_DEVOLVER_CONFIGURACION
359 
360     ; Envio de tiempos de cada pulso
361     LDI R18, LOW(MENSAJE_ESTADO_ENVIAR_TIEMPOS)
362     LDI R19, HIGH(MENSAJE_ESTADO_ENVIAR_TIEMPOS)
363     LDI R17, (1<<BIT_ENVIAR_TIEMPO_PULSOS)
364 
365     CALL DEVOLVER_CONF_BIT
366     SBR EVENTO, (1<<ENVIANDO_DATOS_UART)
367 
368     CALL BUCLE_POLLING_DEVOLVER_CONFIGURACION
369 
370     ; Activacion del buzzer
371     LDI R18, LOW(MENSAJE_ESTADO_SENAL_SONORA)
372     LDI R19, HIGH(MENSAJE_ESTADO_SENAL_SONORA)
373     LDI R17, (1<<BIT_SENAL_SONORA)
374 
375     CALL DEVOLVER_CONF_BIT
376     SBR EVENTO, (1<<ENVIANDO_DATOS_UART)
377 
378     CALL BUCLE_POLLING_DEVOLVER_CONFIGURACION
379 
380     ; Bloqueo de teclado
381     LDI R18, LOW(MENSAJE_ESTADO_BLOQUEO_TECLADO)
382     LDI R19, HIGH(MENSAJE_ESTADO_BLOQUEO_TECLADO)
383     LDI R17, (1<<BIT_BLOQUEO_TECLADO)
384 
385     CALL DEVOLVER_CONF_BIT
386 
387     RET
388 
389 
390 ;MENSAJE_CANTIDAD_VENTANAS: .db "Numero de ventanas:", '\n', 0
```

```
;MENSAJE_TRIGGER_UMBRAL: .db "Umbral del trigger:", '\n', 0
;MENSAJE_VENTANA_DURACION: .db "Duracion de la ventana:", '\n', 0

BUCLE_POLLING_DEVOLVER_CONFIGURACION:
_BUCLE_POLLING_DEVOLVER_CONFIGURACION:
    SBRC EVENTO, ENVIANDO_DATOS_UART
    RJMP _BUCLE_POLLING_DEVOLVER_CONFIGURACION

    RET

; ================================================================
; Descripcion: configura la senal sonora
; Recibe: puntero X con la posicion de memoria del buffer de RX donde comienza el dato booleano
; Salidas: -
CONFIGURAR_SENAL_SONORA:

    CALL VERIFICAR_BOOLEANO_UART

    BRCS _RET_CONFIGURAR_SENAL_SONORA

    CLR T1

    CPI T0, '0'
    BREQ _DESACTIVAR_SENAL_SONORA

    ; Activar senal sonora
    ORI T1, (1<<BIT_SENAL_SONORA)
    STS REGISTRO_CONF_GENERAL, T1
    RET

_DESACTIVAR_SENAL_SONORA:
    ANDI T1, ~(1<<BIT_SENAL_SONORA)
    STS REGISTRO_CONF_GENERAL, T1

_RET_CONFIGURAR_SENAL_SONORA:
    RET


; ================================================================
; Descripcion: configura el bloqueo del teclado
; Recibe: puntero X con la posicion de memoria del buffer de RX donde comienza el dato booleano
; Salidas: -
CONFIGURAR_BLOQUEO_TECLADO:
    CALL VERIFICAR_BOOLEANO_UART
    BRCS _RET_CONFIGURAR_BLOQUEO_TECLADO

    LDS T1, REGISTRO_CONF_GENERAL

    CPI T0, '0'
    BREQ _DESACTIVAR_BLOQUEO_TECLADO

    ; Bloquear teclado
    ORI T1, (1<<BIT_BLOQUEO_TECLADO)
    STS REGISTRO_CONF_GENERAL, T1
    RET

_DESACTIVAR_BLOQUEO_TECLADO:
    ANDI T1, ~(1<<BIT_BLOQUEO_TECLADO)
    STS REGISTRO_CONF_GENERAL, T1

_RET_CONFIGURAR_BLOQUEO_TECLADO:
    RET

; ================================================================
```

```
455 ; Descripcion: configura el envio de los tiempos de cada pulso por UART
456 ; Recibe: puntero X con la posicion de memoria del buffer de RX donde comienza el dato
        booleano
457 ; Salidas: -
458 CONFIGURAR_ENVIO_TIEMPOS_PULSOS:
459     CALL VERIFICAR_BOOLEANO_UART
460     BRCS _RET_CONFIGURAR_ENVIO_TIEMPOS_PULSOS
461 
462     LDS T1, REGISTRO_CONF_GENERAL
463 
464     CPI T0, '0'
465     BREQ _DESACTIVAR_ENVIO_PULSOS
466 
467     ; Bloquear teclado
468     ORI T1, (1<<BIT_ENVIAR_TIEMPO_PULSOS)
469     STS REGISTRO_CONF_GENERAL, T1
470     RET
471 
472 _DESACTIVAR_ENVIO_PULSOS:
473     ANDI T1, ~(1<<BIT_ENVIAR_TIEMPO_PULSOS)
474     STS REGISTRO_CONF_GENERAL, T1
475 
476 _RET_CONFIGURAR_ENVIO_TIEMPOS_PULSOS:
477     RET
478 
479 
480 ; ================================================================
481 ; Descripcion: configura el envio de los tiempos de cada pulso por UART
482 ; Recibe: puntero X con la posicion de memoria del buffer de RX donde comienza el dato
        booleano
483 ; Salidas: -
484 CONFIGURAR_APAGADO_FUENTE:
485     CALL VERIFICAR_BOOLEANO_UART
486     BRCS _RET_CONFIGURAR_APAGADO_FUENTE
487 
488     LDS T1, REGISTRO_CONF_GENERAL
489 
490     CPI T0, '0'
491     BREQ _DESACTIVAR_APAGADO_FUENTE
492 
493     ; Bloquear teclado
494     ORI T1, (1<<BIT_ACTIVAR_FUENTE)
495     STS REGISTRO_CONF_GENERAL, T1
496     RET
497 
498 _DESACTIVAR_APAGADO_FUENTE:
499     ANDI T1, ~(1<<BIT_ACTIVAR_FUENTE)
500     STS REGISTRO_CONF_GENERAL, T1
501 
502 _RET_CONFIGURAR_APAGADO_FUENTE:
503     RET
504 
505 ; ================================================================
506 ; Descripcion: detecta si se ha recibido un parametro booleano por UART
507 ; Recibe: puntero X con la posicion de memoria del buffer de RX donde comienza el dato
        booleano
508 ; Salidas: activa el bit de carry si el dato es no es booleano, sino lo desactiva.
509 VERIFICAR_BOOLEANO_UART:
510     LD T0, X+
511     LD T1, X
512 
513     CPI T1, '\r' ; TODO: CAMBIAR POR \n, el verdadero caracter de fin de trama
514     BRNE _RET_ERROR_VERIFICAR_BOOLEANO_UART
515 
516     CPI T0, '0'
517     BREQ _RET_VERIFICAR_BOOLEANO_UART
```

```
518
519     CPI T0, '1'
520     BREQ _RET_VERIFICAR_BOOLEANO_UART
521
522 _RET_ERROR_VERIFICAR_BOOLEANO_UART:
523     SEC
524     CALL ENVIAR_ERROR_PARSER
525     RET
526
527 _RET_VERIFICAR_BOOLEANO_UART:
528     CLC
529     RET
530
531
532 ; ============================================================
533 ; Descripcion: validacion de datos numericos que llegan por UART en formato ASCII
534 ; Recibe: puntero X con la posicion de memoria del buffer de RX donde comienza el numero
535 ; Salidas: R16 con el valor de la cantidad de digitos si no hubo un error
536 ;          ACtiva el bit de carry si se produjo un error
537 ; NOTA: como maximo pueden recibirse 5 digitos en formato ASCII
538 VALIDAR_ASCII_UART:
539     CLR R16
540
541 _BUCLE_VALIDAR_ASCII_UART:
542     LD T1, X+
543     CPI T1, '\r';TODO: CAMBIAR POR \n
544     BREQ _FIN_BUCLE_VALIDAR_ASCII_UART
545     INC R16
546     CPI R16, MAX_NUM_DIGITOS_5+1
547     BREQ _RET_ERROR_VALIDAR_ASCII_UART
548     CPI T1, '0'
549     BRLO _RET_ERROR_VALIDAR_ASCII_UART
550     CPI T1, '9'+1
551     BRSH _RET_ERROR_VALIDAR_ASCII_UART
552     RJMP _BUCLE_VALIDAR_ASCII_UART
553
554 _FIN_BUCLE_VALIDAR_ASCII_UART:
555     ; Chequear si no se detectaron digitos
556     CPI R16, 0
557     BREQ _RET_ERROR_VALIDAR_ASCII_UART
558
559     ; Regresar el puntero X a su posicion original si no hubo errores
560     ; es necesario para la etapa posterior de transformacion de ASCII a binario
561     MOV T1, R16
562     INC T1
563     SUB XL, T1
564     IN T2, SREG
565     SBRC T2, SREG_C
566     SUBI XH, 1
567
568     ; Borrar el bit de carry indicando que todo salio correctamente
569     CLC
570
571     RET
572
573 _RET_ERROR_VALIDAR_ASCII_UART:
574     ; Activar el bit de carry indicando que se produjo un error
575     SEC
576
577     RET
578
579 ; ====================================================================
580 ; ======================== Espacios de memoria reservados ==========
581 ; ====================================================================
582 ; Mensajes para reportar configuracion
583 MENSAJE_ESTADO_FUENTE: .db "Estado de la fuente: ", 0
```

```
584 MENSAJE_ESTADO_SENAL_SONORA: .db "Señal sonora: ", 0
585 MENSAJE_ESTADO_BLOQUEO_TECLADO: .db "Bloqueo del teclado durante medicion: ", 0
586 MENSAJE_ESTADO_ENVIAR_TIEMPOS: .db "Envio de tiempos de llegada de cada pulso: ", 0
587 MENSAJE_CANTIDAD_VENTANAS: .db "Numero de ventanas:", '\n', 0
588 MENSAJE_TRIGGER_UMBRAL: .db "Umbral del trigger:", '\n', 0
589 MENSAJE_VENTANA_DURACION: .db "Duracion de la ventana:", '\n', 0
```

descripcion_registros.txt

```
1  En el siguiente archivo se listan las funciones de cada registro (R0-R31)
2  dentro del proyecto:
3
4  R0 - Operaciones de multiplicacion
5  R1 - Operaciones de multiplicacion
6  R2 - Reservado: Contador del timer 0
7  R3 - Reservado: Contador del timer 0
8  R4 - De uso general
9  R5 - De uso general
10 R6 - De uso general
11 R7 - De uso general
12 R8 - Reservado: Puntero al buffer de recepcion de UART (LSB)
13 R9 - Reservado: Puntero al buffer de recepcion de UART (MSB)
14 R10 - Reservado: Puntero al buffer de transmision de UART (LSB)
15 R11 - Reservado: Puntero al buffer de transmision de UART (MSB)
16 R12 - Reservado: Cantidad de caracteres almacenados en el buffer de transmision de datos de
       UART (BYTES_TRANSMITIR)
17 R13 - Reservado: Cantidad de caracteres almacenados en el buffer de recepcion de datos de
       UART (BYTES_RECIBIDOS)
18 R14 - De uso general
19 R15 - Reservado: multiplicador para externder el tamano de la ventana de tiempo de las
       mediciones (MUL_DE_VENTANA)
20 R16 - De uso general
21 R17 - De uso general
22 R18 - De uso general
23 R19 - De uso general
24 R20 - De uso general
25 R21 - De uso general
26 R22 - Reservado:
27 R23 - De uso general
28 R24 - Reservado: almacenamiento de los estados de las maquinas de estado (ESTADO)
29 R25 - Reservado: almacenamiento de eventos que alteran los estados de las maquinas de estado
       (EVENTO)
30 R26 - Puntero X
31 R27 - Puntero X
32 R28 - Puntero Y
33 R29 - Puntero Y
34 R30 - Puntero Z
35 R31 - Puntero Z
36
37 Por "uso general" se entiende que el registro no debe guardar un valor
38 que haga de "memoria" de una variable entre distintas ejecuciones de una tarea/funcion/
       subrutina
```

eeprom.asm

```
1  ; Descripcion: funciones relacionadas con el uso de la memoria EEPROM
2  .undef T0
3  .def T0 = R16
4
5  ; Decripcion: almacena en memoria RAM la ultima configuracion del dispositivo
6  ; Recibe: registros de configuracion
7  ; Devuelve: -
8  GUARDAR_CONFIGURACION_EEMPROM:
9      CLR R17
10     CLR R18
11
```

```
12      LDS R19, REGISTRO_VENTANA_TIEMPO
13      CALL GUARDAR_REGISTRO_EEPROM
14
15      INC R17
16      LDS R19, VENTANAS_A_MEDIR_RAM
17      CALL GUARDAR_REGISTRO_EEPROM
18
19      INC R17
20      LDS R19, VENTANAS_A_MEDIR_RAM+1
21      CALL GUARDAR_REGISTRO_EEPROM
22
23      INC R17
24      LDS R19, VENTANAS_A_MEDIR_RAM+2
25      CALL GUARDAR_REGISTRO_EEPROM
26
27      INC R17
28      LDS R19, REGISTRO_UMBRAL
29      CALL GUARDAR_REGISTRO_EEPROM
30
31      INC R17
32      LDS R19, REGISTRO_UMBRAL+1
33      CALL GUARDAR_REGISTRO_EEPROM
34
35      INC R17
36      LDS R19, REGISTRO_UMBRAL+2
37      CALL GUARDAR_REGISTRO_EEPROM
38
39      INC R17
40      LDS R19, REGISTRO_CONF_GENERAL
41      CALL GUARDAR_REGISTRO_EEPROM
42
43      RET
44
45  ; Descripcion: guarda un byte en memoria EEPROM
46  ; Recibe:
47  ; -> R17: LSB de la direccion de memoria
48  ; -> R18: MSB de la direccion de memoria
49  ; -> R19: byte a guardar
50  ; Devuelve: -
51  GUARDAR_REGISTRO_EEPROM:
52  ; Esperar a que el dato anterior se haya escrito
53  _BUCLE_GUARDAR_REGISTRO_EEPROM:
54      IN T0, EECR
55      SBRC T0, EEPE
56      RJMP _BUCLE_GUARDAR_REGISTRO_EEPROM
57
58      OUT EEARH, R18
59      OUT EEARL, R17
60
61      OUT EEDR, R19
62
63      CLR T0
64      ORI T0, (1<<EEMPE)
65      OUT EECR, T0
66
67      CLR T0
68      ORI T0, (1<<EEPE)
69      OUT EECR, T0
70
71      RET
72
73  ; Decripcion: configura el dispositivo (mediante registros en RAM) a partir
74  ; de la ultima configuracion almacenada en EEPROM
75  ; Recibe: registros de configuracion
76  ; Devuelve: -
77  CARGAR_CONFIGURACION_EEPROM:
```

```
78      CLR R17
79      CLR R18
80
81      CALL LEER_REGISTRO_EEPROM
82      STS REGISTRO_VENTANA_TIEMPO, R19
83      INC R17
84
85      CALL LEER_REGISTRO_EEPROM
86      STS VENTANAS_A_MEDIR_RAM, R19
87      INC R17
88
89      CALL LEER_REGISTRO_EEPROM
90      STS VENTANAS_A_MEDIR_RAM+1, R19
91      INC R17
92
93      CALL LEER_REGISTRO_EEPROM
94      STS VENTANAS_A_MEDIR_RAM+2, R19
95      INC R17
96
97      CALL LEER_REGISTRO_EEPROM
98      STS REGISTRO_UMBRAL, R19
99      INC R17
100
101     CALL LEER_REGISTRO_EEPROM
102     STS REGISTRO_UMBRAL+1, R19
103     INC R17
104
105     CALL LEER_REGISTRO_EEPROM
106     STS REGISTRO_UMBRAL+2, R19
107     INC R17
108
109     CALL LEER_REGISTRO_EEPROM
110     STS REGISTRO_CONF_GENERAL, R19
111
112     RET
113
114
115 ; Decripcion: devuelve el dato almacenado en un registro de memoria EEPROM
116 ; Recibe:
117 ; -> R17: LSB de la direccion de memoria
118 ; -> R18: MSB de la direccion de memoria
119 ; Devuelve:
120 ; -> R19: byte leido
121 LEER_REGISTRO_EEPROM:
122 _BUCLE_CARGAR_REGISTRO_EEPROM:
123     IN T0, EECR
124     SBRC T0, EEPE
125     RJMP _BUCLE_GUARDAR_REGISTRO_EEPROM
126
127     OUT EEARH, R18
128     OUT EEARL, R17
129
130     CLR T0
131     ORI T0, (1<<EERE)
132     OUT EECR, T0
133
134     IN R19, EEDR
135
136     RET
```

LCD.asm

```
1 ;Fede: En este archivo pongo las funciones que se usan para manejar el LCD. En cada una
      detallo qué hacen, registros que usan, etc.
2
3 ;Se hace una serie de definiciones previas para simplificar la lectura del código y la
```

```
    escritura también.
4
5   .EQU PUERTO_LCD = PORTD                         ;EL PUERTO D PARA ENVIAR LOS DATOS AL LCD
6   .EQU PIN_LCD = PIND                             ;PARA LEER DATOS DEL PUERTO D
7   .EQU PIN_ENABLE = 2                             ;EL PIN 5 DEL PUERTO D CONECTADO AL
        ENABLE
8   .EQU PIN_RS = 3                                 ;EL PIN 4 DEL PUERTO D CONECTADO AL RS
9   .EQU DDR_LCD = DDRD                             ;REGISTRO PARA CONFIGURAR EL PUERTO QUE
        ENVIA LOS DATOS, COMO SALIDA
10
11
12
13  .EQU LCD_N_BITS = 0x28                          ;COMANDO DE INICIALIZACIÓN DEL LCD EN
        MODO 4 BITS
14  ;.EQU LCD_N_BITS = 0x20                         ;COMANDO PARA INDICAT MODO DE 4 BITS, Y
        UNA LINEA DE ESCRITURA
15  .EQU LCD_DISPLAY_ON = 0x0C                      ;COMANDO PARA ENCENDER EL DISPLAY
16  .EQU LCD_CLR_SCREEN = 0x01                      ;COMANDO PARA LIMPIAR LA PANTALLA
17  .EQU LCD_HOME_SCREEN = 0x02                     ;COMANDO PARA PARARSE EN EL EXTREMO
        SUPERIOR IZQUIERDO DE LA PANTALLA
18  .EQU LCD_CURSOR_DERECHA = 0x18                  ;COMANDO PARA ESCRIBIR HACIA LA DERECHA
        EN LA PANTALLA
19  .EQU LCD_SHIFTEAR_CURSOR_DERECHA = 0x14         ;COMANDO PARA MOVER EL CURSOR PARA
        ESCRIBIR, HACIA LA DERECHA
20  .EQU LCD_SHIFTEAR_CURSOR_IZQUIERDA = 0x10       ;COMANDO PARA MOVER EL CURSOR PARA
        ESCRIBIR, HACIA LA IZQUIERDA
21  .EQU LCD_CAMBIAR_RENGLON = 0xC0                 ;COMANDO PARA ESCRIBIR EN EL RENGLÓN DE
        ABAJO
22
23  .EQU LCD_CURSOR_ON = 0x0F                       ;COMANDO PARA PONER EL CURSOR PARPADEANTE
24  .EQU LCD_CURSOR_OFF = 0x0C                      ;COMANDO PARA QUITAR EL CURSOR
        PARPADEANTE
25
26
27
28  .EQU FLECHA_IZQUIERDA_ASCII = 0x7F              ;ASCII para la flecha para la izquierda
29  .EQU FLECHA_DERECHA_ASCII = 0x7E                ;ASCII para la flecha para la derecha
30
31  ;
        ================================================================================================
32  ;
        ================================================================================================
33
34  ;Se define una MACRO para la invocación de DELAYS para el LCD
35  .MACRO DELAY_LCD
36      PUSH R16
37      LDI R16, @0
38      MOV R14, R16
39      POP R16
40  DELAY_1:
41      PUSH R16
42      MOV R16, R14
43      CPI R16, 0x00
44      BREQ END_DELAY_1
45      POP R16
46      RJMP DELAY_1
47  END_DELAY_1:
48  POP R16
49  .ENDMACRO
50
51  ;Se define una macro para enviar comandos al LCD
52  .MACRO COMANDO_LCD
53      PUSH R14
54      PUSH R16
```

```
55      LDI R16, @0
56      MOV R14, R16
57      POP R16
58      CALL CMNDWRT_LCD
59      POP R14
60  .ENDMACRO
61
62  ; Se define una macro para enviar cadenas de caracteres desde la flash
63
64  .MACRO CADENA_FLASH_LCD
65      PUSH ZH
66      PUSH ZL
67      LDI ZL, LOW(@0<<1)
68      LDI ZH, HIGH(@0<<1)
69      CALL STRING_WRT_LCD_FLASH
70      POP ZL
71      POP ZH
72  .ENDMACRO
73
74  ; Se define una macro para borrar el renglón
75
76  .MACRO BORRAR_RENGLON
77      CADENA_FLASH_LCD ESPACIO
78  .ENDMACRO
79
80
81  ;   ================================================================================================
82  ;   ================================================================================================
83
84  ;PUERTO D
85  ;[D4][D5][D6][D7][RS][E][-][-]
86  ;D4-D7: Por estos pines se envía el dato al LCD. Se va a usar el modo de escritura de 4
        pines.
87  ;RS: RS = 1 permite escribir datos en la pantalla, RS = 0 permite enviar comandos al LCD.
88  ;E: Se envía un pulso de entre 500 y 300 ns en este pin, para que el LCD reciba el dato de
        los pines D4-D7.
89
90  ;*************************LOS COMANDOS ESTÁN EN EL DATASHEET DEL LCD
        **************************
91
92  INIT_LCD:
93  ;Función que realiza las tareas de inicialización del LCD. La inicialización consiste en el
        envío de una serie de comandos.
94  ;Se inicializa con escritura por 4 bits, encendiendo el display y limpiando la pantalla. Por
        último se iniciliza para una escritura
95  ;hacia la derecha y posicionando el cursor en el extremo superior izquierdo de la pantalla.
96  ;Registros utilizados: R16, R14
97      PUSH R16
98      DELAY_LCD 0x96              ;SE ESPERAN 15ms
99      LDI R16, LCD_N_BITS         ;CANTIDAD DE BITS A UTILIZAR PARA EL SCREEN, EN ESTE CASO 4
            BITS
100     MOV R14, R16
101     CALL CMNDWRT_LCD            ;SE ESCRIBE EL COMANDO EN EL LCD
102     LDI R16, LCD_DISPLAY_ON     ;ENCENDER DISPLAY
103     MOV R14, R16
104     CALL CMNDWRT_LCD            ;SE ESCRIBE EL COMANDO EN EL LCD
105     LDI R16, LCD_CLR_SCREEN     ;COMANDO PARA LIMPIAR LA PANTALLA
106     MOV R14, R16
107     CALL CMNDWRT_LCD            ;SE ESCRIBE EL COMANDO
108     DELAY_LCD 0x14              ;DELAY PARA QUE EL LCD EJECUTE EL COMANDO, 2ms
109     LDI R16, LCD_CURSOR_DERECHA ;SHIFTEAR CURSOR A LA DERECHA
110     MOV R14, R16
```

```
111     CALL CMNDWRT_LCD            ;ESCRIBO EN EL LCD
112     LDI R16, LCD_HOME_SCREEN    ;SE POSICIONA EL CURSOR EN EL EXTREMO SUPERIOR IZQUIERDO PARA
            COMENZAR A ESCRIBIR
113     MOV R14, R16
114     CALL CMNDWRT_LCD            ;SE ENVÍA EL COMANDO
115     DELAY_LCD 0x14              ;SE HACE UN DELAY DE 2ms PARA QUE SE EJECUTE EL COMANDO
116     POP R16
117     RET
118 ;
    ********************************************************************************************

119 DATAWRT_LCD:
120 ;Escribe un dato en la pantalla LCD, previamente cargado en el registro R16
121 ;REGISTROS UTILIZADOS: R14, R21, R20
122     PUSH R20                    ;SE COPIAN LOS DATOS PARA NO PERDERLOS
123     PUSH R21
124     ;PUSH R14
125     MOV R20, R14                ;SE COPIA EL DATO A ENVIAR, DEL REGISTRO 16 AL REGISTRO 27
126     ANDI R20, 0xF0              ;SE APLICA UNA MASCARA 11110000 PARA QUEDARSE CON EL NIBBLE
            ALTO
127     IN R21, PIN_LCD             ;SE COPIA EL DATO DEL PUERTO
128     ANDI R21, 0x0F              ;SE ELIMINAN LOS DATOS DEL NIBBLE ALTO.
129     OR R21, R20                 ;Y SE COLOCA EL DATO A ENVIAR, EN EL NIBBLE ALTO.
130     OUT PUERTO_LCD, R21         ;SE COLOCA EL DATO EN EL PUERTO
131     SBI PUERTO_LCD, PIN_RS      ;PIN RS EN 1, PARA PODER ESCRIBIR DATOS EN EL LCD
132     SBI PUERTO_LCD, PIN_ENABLE  ;SE PONE UN 1 EN ENABLE PARA EMPEZAR A ENVIAR EL DATO
133     CALL SDELAY                 ;SE LLAMA A UN DELAY YA QUE EL PULSO DEBE TENER UN CIERTO
            ANCHO
134     CBI PUERTO_LCD, PIN_ENABLE  ;SE PONE EL ENABLE EN 0 PARA TERMINAR LA TRANSMISIÓN DEL
            NIBBLE ALTO
135
136     MOV R20, R14                ;SE COPIA EL DATO ORIGINAL NUEVAMENTE AL REGISTRO 27
137     SWAP R20                    ;SE INVIERTEN LOS NIBBLES DEL DATO
138     ANDI R20, 0xF0              ;SE APLICA LA MASCARA NUEVAMENTE PARA RECUPERAR SOLO EL
            NIBBLE
139     IN R21, PIN_LCD             ;SE COPIA EL DATO DEL PUERTO
140     ANDI R21, 0x0F              ;SE ELIMINAN LOS DATOS DEL NIBBLE ALTO.
141     OR R21, R20                 ;Y SE COLOCA EL DATO A ENVIAR, EN EL NIBBLE ALTO.
142     OUT PUERTO_LCD, R21         ;SE COLOCA EL DATO EN EL PUERTO
143     SBI PUERTO_LCD, PIN_ENABLE  ;SE PONE UN 1 EN ENABLE PARA EMPEZAR A ENVIAR EL DATO
144     CALL SDELAY                 ;SE LLAMA A UN DELAY YA QUE EL PULSO DEBE TENER UN CIERTO
            ANCHO
145     CBI PUERTO_LCD, PIN_ENABLE  ;SE PONE EL ENABLE EN 0 PARA TERMINAR LA TRANSMISIÓN DEL
            NIBBLE ALTO
146
147     DELAY_LCD 0x01              ;SE PONE UN DELAY DE 60 us. SEGUN EL LIBRO ESTE DELAY VA
            CUANDO SE MANDAN COMANDOS, NO SE PORQUE LO PONE ACÁ                 ;SE RECUPERAN
            LOS DATOS
148     ;POP R14
149     POP R21
150     POP R20
151     RET
152 ;
    ********************************************************************************************

153 CMNDWRT_LCD:
154 ;Envía un comando a la pantalla LCD, previamente alojado en el R16.
155 ;REGISTROS UTILIZADOS: R14, R21, R20
156     PUSH R20                    ;SE COPIAN LOS DATOS PARA NO PERDERLOS
157     PUSH R21
158     ;PUSH R14
159     MOV R20, R14                ;SE COPIA EL DATO A ENVIAR, DEL REGISTRO 16 AL REGISTRO 27
160     ANDI R20, 0xF0              ;SE APLICA UNA MASCARA 11110000 PARA QUEDARSE CON EL NIBBLE
            ALTO
161     IN R21, PIN_LCD             ;EL DATO DEL PUERTO, PARA NO PERDERLO
162     ANDI R21, 0x0F              ;ME QUEDO CON EL NIBBLE NO UTILIZADO CUANDO ENVÍO EL DATO
```

```
163     OR R21, R20                 ;COMBINO EL DATO CON LA CONFIGURACION DEL RESTO DEL PUERTO
164     OUT PUERTO_LCD, R21         ;SE COLOCA EL DATO EN EL PUERTO
165     CBI PUERTO_LCD, PIN_RS      ;PIN RS EN 0, PARA PODER ENVIAR DATOS AL LCD
166     SBI PUERTO_LCD, PIN_ENABLE  ;SE PONE UN 1 EN ENABLE PARA EMPEZAR A ENVIAR EL DATO
167     CALL SDELAY                 ;SE LLAMA A UN DELAY YA QUE EL PULSO DEBE TENER UN CIERTO
            ANCHO
168     CBI PUERTO_LCD, PIN_ENABLE  ;SE PONE EL ENABLE EN 0 PARA TERMINAR LA TRANSMISIÓN DEL
            NIBBLE ALTO
169
170     MOV R20, R14                ;SE COPIA EL DATO ORIGINAL NUEVAMENTE AL REGISTRO 27
171     SWAP R20                    ;SE INVIERTEN LOS NIBBLES DEL DATO
172     ANDI R20, 0xF0              ;SE APLICA LA MASCARA NUEVAMENTE PARA RECUPERAR SOLO EL
            NIBBLE
173     IN R21, PIN_LCD             ;EL DATO DEL PUERTO, PARA NO PERDERLO
174     ANDI R21, 0x0F              ;ME QUEDO CON LA PARTE QUE NO ES DE DATOS
175     OR R21, R20                 ;COMBINO EL DATO A ENVIAR Y LA CONFIG DEL PUERTO
176     OUT PUERTO_LCD, R21         ;SE COLOCA EL DATO EN EL PUERTO
177     SBI PUERTO_LCD, PIN_ENABLE  ;SE PONE UN 1 EN ENABLE PARA EMPEZAR A ENVIAR EL DATO
178     CALL SDELAY                 ;SE LLAMA A UN DELAY YA QUE EL PULSO DEBE TENER UN CIERTO
            ANCHO
179     CBI PUERTO_LCD, PIN_ENABLE  ;SE PONE EL ENABLE EN 0 PARA TERMINAR LA TRANSMISIÓN DEL
            NIBBLE ALTO
180
181     DELAY_LCD 0x01              ;SE PONE UN DELAY DE 100 us. SEGUN EL LIBRO ESTE DELAY VA
            CUANDO SE MANDAN COMANDOS, NO SE PORQUE LO PONE ACÁ                         ;SE
            RECUPERAN LOS DATOS
182     ;POP R14
183     POP R21
184     POP R20
185     RET
186 ;
    ********************************************************************************************

187 SDELAY:
188 ;Delay de dos ciclos de máquina (para 16 megahertz, serían unos 125ns, sumados al call previo
        y al ret, un total de 625 ns).
189 ;Este delay se utiliza para generar el pulso en el pin ENABLE, cuando se envía un dato/
        comando al LCD.
190     NOP
191     NOP
192     RET
193 ;
    ********************************************************************************************

194 LCD_INTERRUPCION_OVERFLOW:
195 ;Se ejecuta cuando ocurre un overflow del timer. Se revisa el registro 18. Si hay un valor
        distinto de cero
196 ;Se decrementa en 1. Si no, no se hace nada
197 ;Registros utilizados: R14, R16
198     PUSH R16
199
200     IN R16, SREG
201     PUSH R16
202
203     MOV R16, R14
204     CPI R16, 0x00
205     BREQ TRUE
206     DEC R14
207 TRUE:
208     POP R16
209     OUT SREG, R16
210     POP R16
211     RETI
212 ;
    ********************************************************************************************
```

```
213 ; TODO: CAMBIAR DDRB PORQUE ES EL PUERTO QUE RECIBE LOS PULSOS
214 CONFIGURAR_PUERTOS_LCD:
215     PUSH R16
216     LDI R16, 0xFF
217     OUT DDR_LCD, R16            ;PUERTO DEL LCD COMO SALIDA
218     LDI R16, 0xFE               ;|O|O|O|O|O|O|O|I|
219     OUT DDRB, R16               ;Puerto B como salida para el LED
220     POP R16
221     RET
222
223 ;
    ***************************************************************************************************

224 STRING_WRT_LCD:
225 ;Función para escribir cadenas de caracteres en el LCD. Recibe el puntero a la posición de la
        string y la envía caracter por caracter al LCD. Recibe el
226 ;puntero Z cargado en la posición del STRING para enviar el dato.
227 ;Registros utilizados: R16
228     PUSH R16
229     PUSH R14
230
231     ESCRIBIR_SIGUIENTE_CARACTER:
232         LD R16, Z+                      ;Se carga el primer caracter al registro R16, que
                utiliza la función DATAWRT_LCD, para enviar el dato
233         CPI R16, 0                      ;Si es 0 es porque terminé la string.
234         BREQ END_STRING
235         MOV R14, R16
236         CALL DATAWRT_LCD                ;Se invoca a la función para escribir el dato
237         RJMP ESCRIBIR_SIGUIENTE_CARACTER    ;Se sigue escribiendo hasta terminar la string
238     END_STRING:
239         POP R14
240         POP R16
241         RET
242 ;
    ***************************************************************************************************

243
244 STRING_WRT_LCD_FLASH:
245 ;Función para escribir cadenas de caracteres en el LCD. Recibe el puntero a la posición de la
        string y la envía caracter por caracter al LCD. Recibe el
246 ;puntero Z cargado en la posición del STRING para enviar el dato. LA STRING DEBE ESTAR EN
       FLASH!
247 ;Registros utilizados: R16
248     PUSH R16
249     PUSH R14
250
251     ESCRIBIR_SIGUIENTE_CARACTER_FLASH:
252         LPM R16, Z+                     ;Se carga el primer caracter al registro R16, que
                utiliza la función DATAWRT_LCD, para enviar el dato
253         CPI R16, 0                      ;Si es 0 es porque terminé la string.
254         BREQ END_STRING_FLASH
255         MOV R14, R16
256         CALL DATAWRT_LCD                ;Se invoca a la función para escribir el dato
257         RJMP ESCRIBIR_SIGUIENTE_CARACTER_FLASH  ;Se sigue escribiendo hasta terminar la
                string
258     END_STRING_FLASH:
259         POP R14
260         POP R16
261         RET
262 ;
    ***************************************************************************************************

263
264
265 MENSAJE_BIENVENIDA_LCD: .db "Contador Geiger", 0
266 MENSAJE_PROMEDIO_LCD: .db "Promedio:", 0
```

```
267 ESPACIO: .db "                ", 0
```

macros.inc

```
1  ; timers.asm
2  ; Created: 24/10/2018 9:16
3  ; Author : Juan Pablo Goyret
4  ; Descripcion: biblioteca con macros utiles
5
6
7  ; Descripcion: limpia el bit @1 en el resistro @0
8  .MACRO CLEAR_BIT
9  .if @0<0x40
10         IN R16, @0
11 .else
12         LDS R16, @0
13 .endif
14     ANDI R16, ~(1 << @1)
15 .if @0<0x40
16         OUT @0,R16
17 .else
18         STS @0,R16
19 .endif
20 .ENDMACRO
21
22 ; Descripcion: activa el bit @1 en el resistro @0
23 .MACRO SET_BIT
24 .if @0<0x40
25         IN R16, @0
26 .else
27         LDS R16, @0
28 .endif
29     ORI R16, (1 << @1)
30 .if @0<0x40
31         OUT @0,R16
32 .else
33         STS @0,R16
34 .endif
35 .ENDMACRO
36
37 ; Descripcion: pone todo el registro @0 en cero
38 ; salvo por el bit @1
39 .MACRO WRITE_REG
40     LDI R16, @1
41 .if @0<0x40
42         OUT @0,R16
43 .else
44         STS @0,R16
45 .endif
46 .ENDMACRO
47
48 ; Descripcion: hace un push al stack de dos registros @0 y @1
49 .MACRO PUSH_WORD
50     PUSH @0
51     PUSH @1
52 .ENDMACRO
53
54 ; Descripcion: hace un pop del stack de dos registros @0 y @1
55 .MACRO POP_WORD
56     POP @1
57     POP @0
58 .ENDMACRO
59
60 ; Descripcion: sumar dos registros de 16 bits
61 ; Parametros: @0 (MSB del primer registro)
62 ;             @1 (LSB del primer registro)
```

```
;             @2 (MSB del segundo registro)
;             @3 (LSB del segundo registro)
.MACRO SUMAR_REGISTROS_16_BITS
    ADD @1, @3
    ADC @0, @2
.ENDMACRO


; Descripcion: sumar dos registros de 24 bits
; Parametros: @0 (MSB del primer registro)
;             @1 (byte intermedio del primer registro)
;             @2 (LSB del primer registro)
;             @3 (MSB del segundo registro)
;             @4 (byte intermedio del segundo registro)
;             @5 (LSB del segundo registro)
.MACRO SUMAR_REGISTROS_24_BITS
    ADD @2, @5
    ADC @1, @4
    ADC @0, @3
.ENDMACRO

; =====================================================================
; ==================== Macros de debugging ============================
; =====================================================================

; Descripcion: hacer un toggle del led del Arduino
.MACRO TOGGLE_LED_ARDUINO
    PUSH R16
    PUSH R17

    IN  R16, PORTB
    LDI R17, (1 << 5)
    EOR R16, R17
    OUT PORTB, R16

    POP R17
    POP R16
.ENDMACRO


; Descripcion: encender LED del Arduino
.MACRO ENCENDER_LED_ARDUINO
    PUSH R16

    IN  R16, PORTB
    ORI R16, (1 << 5)
    OUT PORTB, R16

    POP R16
.ENDMACRO

; Descripcion: encender LED del Arduino
.MACRO APAGAR_LED_ARDUINO
    PUSH R16

    IN  R16, PORTB
    ANDI R16, ~(1 << 5)
    OUT PORTB, R16

    POP R16
.ENDMACRO

; Descripcion: configurar pin del Arduino que posee un LED conectado
; para encenderlo a gusto
.MACRO CONF_LED_ARDUINO
    PUSH R16
```

```
129
130 /*  IN R16, DDRB
131     SBR R16, 5*/
132
133     SBI DDRB, 5
134     CBI PORTB, 5
135
136 /*  LDI R16, (1 << 5)
137     OUT DDRB, R16
138     LDI R16, ~(1 << 5)
139     OUT PORTB, R16*/
140
141     POP R16
142 .ENDMACRO
```

Mediciones.asm

```
1  ;Funciones relacionadas con las mediciones a realizar.
2  ;Se las invoca una vez inicializado el clock.
3  ; =============================================================================
4  ; Variables en RAM
5  .dseg
6
7  MUL_DE_VENTANA_RAM: .byte 1             ;Memoria para guardar el mul de ventana en ram
8
9  CONTADOR_DE_PULSOS_RAM: .byte 3         ;Se irán acumulando el total de pulsos medidos en
       cada ventana: [nibble bajo];[nibble medio];[nibble alto]
10 ; =============================================================================
11 .cseg
12
13 .DEF UMBRAL_A_SUPERAR = R21             ;En el registro R21 se guardará la configuración del
       umbral a superar.
14
15 .DEF CONTADOR_DE_PULSOS = R23           ;En el registro R23 se lleva la cuenta de la cantidad
        de pulsos medidos.
16
17 .DEF VENTANAS_A_MEDIR_LOW = R7
18 .DEF VENTANAS_A_MEDIR_MID = R5
19 .DEF VENTANAS_A_MEDIR_HIGH = R13
20
21 .DEF VENTANA_DE_TIEMPO = R26            ;0 = 1ms, 1 = 10ms, 2 = 100ms, 3 = 1s
22 ;.DEF VENTANA_DE_TIEMPO = R19
23
24 .EQU VENTANA_LOW = 0xA8                 ;SE UTILIZA UNA VENTANA DE 100 ms, para ello se
       cuenta 25000 con el timer1, a un prescaler de clk/64 = 250khz.
25 .EQU VENTANA_HIGH = 0x61
26 .EQU DURACION_TOTAL = 100               ;CANTIDAD DE VECES QUE SE VA A REALIZAR EL REINICIO
       DEL CLOCK, PARA SEGUIR MIDIENDO: EN ESTE CASO, CON UN
27                                         ;PRESCALER DE CLK/64, CONTAR 50 VECES ES 5 segundos.
28
29 ;
       ********************************************************************************************
30 ;CONFIGURACIÓN DEL UMBRAL
31 .EQU PIN_PORTB_LED = 2
32 .EQU PIN_PORTB_BUZZER = 3
33
34 ;
       ********************************************************************************************
35 ; PUERTO DE ACTIVACION/DESACTIVACION DEL CONTADOR
36 .EQU PUERTO_ACTIVAR_CONTADOR = 4
37
38 ;
       ********************************************************************************************
```

```
;CONFIGURACIÓN DE LA DURACIÓN DE LA VENTANA

.EQU VENTANA_1ms = 0
.EQU VENTANA_10ms = 1
.EQU VENTANA_100ms = 2
.EQU VENTANA_1000ms = 3

;VALORES DEL COMPARADOR DEL TIMER, DEPENDIENDO DE LA VENTANA SELECCIONADA

.EQU VENTANA_HIGH_1ms = 0x00            ;0x00FA = 250
.EQU VENTANA_LOW_1ms = 0xFA

.EQU VENTANA_HIGH_10ms = 0x09           ;0x09C4 = 2500
.EQU VENTANA_LOW_10ms = 0xC4

.EQU VENTANA_HIGH_100ms = 0x61          ;0x61A8 = 25000
.EQU VENTANA_LOW_100ms = 0xA8

.EQU VENTANA_HIGH_1000ms = 0xC3         ;0xC350 = 50000
.EQU VENTANA_LOW_1000ms = 0x50

;REPETIDORES DE OVERFLOW DEL TIMER PARA COMPLETAR UNA VENTANA

.EQU REPETIDOR_OVERFLOW_TIMER_1ms = 0x01
.EQU REPETIDOR_OVERFLOW_TIMER_10ms = 0x01
.EQU REPETIDOR_OVERFLOW_TIMER_100ms = 0x01
.EQU REPETIDOR_OVERFLOW_TIMER_1000ms = 0x05

;
    ********************************************************************************************


CONFIG_VENTANA_DE_TIEMPO:
;Función para setear la ventana de tiempo que se utilizará para las mediciones. Se utiliza el
     registro VENTANA_DE_TIEMPO para saber qué ventana se utiliza.

;SETEO DE LA CONFIGURACIÓN DE LA VENTANA DE TIEMPO: VENTANA_DE_TIEMPO = 0, 1ms;
    VENTANA_DE_TIEMPO = 1, 10ms; VENTANA_DE_TIEMPO = 2, 100ms;
;VENTANA_DE_TIEMPO = 3, 1000ms
;DEPENDIENDO DE CUAL ES EL VALOR DE VENTANA_DE_TIEMPO, SE HACE UN SALTO PARA HACER LA
    CONFIGURACIÓN ADECUADA
    PUSH R19
    PUSH R20
    PUSH R21

    CPI VENTANA_DE_TIEMPO, VENTANA_1ms
    BREQ CONFIG_1ms

    CPI VENTANA_DE_TIEMPO, VENTANA_10ms
    BREQ CONFIG_10ms

    CPI VENTANA_DE_TIEMPO, VENTANA_100ms
    BREQ CONFIG_100ms

    CPI VENTANA_DE_TIEMPO, VENTANA_1000ms
    BREQ CONFIG_1000ms

    CONFIG_1ms:
        LDI R20, VENTANA_HIGH_1ms                   ;ANCHO DE VENTANA (NIBBLE ALTO) 1ms
        LDI R21, VENTANA_LOW_1ms                    ;ANCHO DE VENTANA (NIBBLE BAJO) 1ms
        LDI R19, REPETIDOR_OVERFLOW_TIMER_1ms       ;CANTIDAD DE VECES QUE SE DEBE HACER
            OVERFLOW AL TIMER 1 PARA QUE SE CUMPLA UNA VENTANA
        RJMP CARGAR_VENTANA                         ;UNA VEZ CONFIGURADO EL VALOR SE CARGA

    CONFIG_10ms:
        LDI R20, VENTANA_HIGH_10ms
```

```
99          LDI R21, VENTANA_LOW_10ms
100         LDI R19, REPETIDOR_OVERFLOW_TIMER_10ms
101         RJMP CARGAR_VENTANA
102
103     CONFIG_100ms:
104         LDI R20, VENTANA_HIGH_100ms
105         LDI R21, VENTANA_LOW_100ms
106         LDI R19, REPETIDOR_OVERFLOW_TIMER_100ms
107         RJMP CARGAR_VENTANA
108
109     CONFIG_1000ms:
110         LDI R20, VENTANA_HIGH_1000ms
111         LDI R21, VENTANA_LOW_1000ms
112         LDI R19, REPETIDOR_OVERFLOW_TIMER_1000ms
113         RJMP CARGAR_VENTANA
114
115     CARGAR_VENTANA:
116         MOV MUL_DE_VENTANA, R19                 ;SE CARGA EL MULTIPLICADOR DE VENTANA
117         STS OCR1AH, R20                         ;SE CARGA EL VALOR DE COMPARACIÓN DEL TIMER,
                EN OCR1AH|OCR1AL
118         STS OCR1AL, R21                         ;AUTOMATICAMENTE CUANDO SE CARGA EL LOW
                NIBBLE, SE CARGAN AMBOS AL MISMO TIEMPO
119         RJMP FIN_CONFIG_VENTANA                 ;SE TERMINÓ DE CONFIGURAR LA VENTANA, SE SALE
                 DE LA FUNCIÓN
120
121     FIN_CONFIG_VENTANA:
122         POP R21
123         POP R20
124         POP R19
125         RET
126
127
128 ;
    ********************************************************************************************
129
130
131 CONFIG_REPETIR_VENTANAS:
132
133 ;Función para cargar la configuración de la cantidad de veces que se va a repetir la ventana.
        Se utiliza el registro VENTANAS_A_MEDIR,
134 ;donde se indica la cantidad de veces que se repite la medición con dicha ventana.
135
136 ;REGISTROS UTILIZADOS: R22:R12
137 ; ====================VERSIÓN ANTERIOR
    ============================================================
138     ;LDI VENTANAS_A_MEDIR, DURACION_TOTAL    ;SE CARGA CUANTAS VECES SE VA A REINICIAR EL
            CONTADOR
139 ;
    ============================================================================================
140
141
142     LDS VENTANAS_A_MEDIR_HIGH, VENTANAS_A_MEDIR_RAM+2       ;Se carga el nibble bajo de la
            configuración del usuario, en un registro VENTANAS_A_MEDIR_LOW
143     MOV VENTANAS_A_MEDIR_LOW, VENTANAS_A_MEDIR_HIGH
144     LDS VENTANAS_A_MEDIR_HIGH, VENTANAS_A_MEDIR_RAM+1
145     MOV VENTANAS_A_MEDIR_MID, VENTANAS_A_MEDIR_HIGH         ;Se carga el nibble intermedio de
             la configuración del usuario, en un registro VENTANAS_A_MEDIR MID
146     LDS VENTANAS_A_MEDIR_HIGH, VENTANAS_A_MEDIR_RAM         ;Se carga el nibble alto de la
            configuración del usuario, en un registro VENTANAS_A_MEDIR HIGH
147
148
149     RET
150
151
```

```
152 ;*****************************************************************************************
153
154
155 CONFIG_CONTADOR_DE_PULSOS:
156
157 ;Se inicializa el registro CONTADOR_DE_PULSOS, donde se cuenta la cantidad de pulsos totales medidos.
158 ;REGISTROS_UTILIZADOS: R23
159
160     LDI CONTADOR_DE_PULSOS, 0x00
161     STS CONTADOR_DE_PULSOS_RAM, CONTADOR_DE_PULSOS
162     LDI CONTADOR_DE_PULSOS, 0x00
163     STS CONTADOR_DE_PULSOS_RAM+1, CONTADOR_DE_PULSOS
164     LDI CONTADOR_DE_PULSOS, 0x00
165     STS CONTADOR_DE_PULSOS_RAM+2, CONTADOR_DE_PULSOS
166 RET
167
168
169 ;*****************************************************************************************
170
171
172 INICIAR_TIMER_1:
173 ;Función que, una vez configurada la medición, inicializa el timer 1 para realizar las mediciones. Se inicializa con prescaler dividido 64.
174     PUSH R17                            ;SE PUSHEA PARA NO PERDER EL REGISTRO
175     LDS R17, TCCR1B                     ;SE CARGA EL DATO DEL TCCR1B EN R17
176     ORI R17, (1<<CS10)|(1<<CS11)        ;SE CONFIGURA EL PRESCALER A 16MHz/64 = 250 kHz == UNA CUENTA HASTA 262.14 ms
177     ANDI R17, ~(1<<CS12)
178     STS TCCR1B, R17                     ;EN ESTE INSTANTE SE INICIA EL TIMER 1, ES DECIR, YA SE EMPEZÓ A MEDIR LOS PULSOS
179     POP R17                             ;SE RECUPERA EL DATO GUARDADO PREVIAMENTE
180 RET
181
182 ;*****************************************************************************************
183 /*ACUMULAR_PULSOS_DETECTADOS:
184 ; Se guardan los pulsos detectados en una variable en RAM.
185 ; REGISTROS UTILIZADOS: R17
186     PUSH R17
187
188     LDS R17, CONTADOR_DE_PULSOS_RAM     ;Se levanta el dato de la cantidad de pulsos medidos hasta ahora (nibble bajo)
189     ADD R17, CONTADOR_DE_PULSOS         ;Se le suma la cantidad de pulsos medidios en la ventana actual
190     STS CONTADOR_DE_PULSOS_RAM, R17     ;Se guarda el acumulado en ram
191
192     BRCC END_ACUMULAR_PULSOS            ;Si el carry es 0, no se incrementa el nibble alto
193
194     LDS R17, CONTADOR_DE_PULSOS_RAM + 1
195     INC R17
196     STS CONTADOR_DE_PULSOS_RAM+1, R17
197
198     BRCC END_ACUMULAR_PULSOS
199
200     LDS R17, CONTADOR_DE_PULSOS_RAM+2
201     INC R17
202     STS CONTADOR_DE_PULSOS_RAM+2, R17
203
204     END_ACUMULAR_PULSOS:
205
```

```
206         POP R17
207         RET*/
208 ;
    ********************************************************************************************

209 ACUMULAR_PULSOS_DETECTADOS:
210 ; Segunda versión de la función de acumular los pulsos
211
212     PUSH R16
213     PUSH R17
214     PUSH R18
215     PUSH R19
216
217     LDI R19, 0x00
218
219     LDS R16, CONTADOR_DE_PULSOS_RAM
220     LDS R17, CONTADOR_DE_PULSOS_RAM+1
221     LDS R18, CONTADOR_DE_PULSOS_RAM+2
222
223     ADD R16, CONTADOR_DE_PULSOS                 ; Se suma el nibble bajo
224     ADC R17, R19                                ; Si hay carry, se suma uno al nibble medio
225     ADC R18, R19                                ; Si vuelve a haber carry se suma uno al
            nibble alto
226
227     STS CONTADOR_DE_PULSOS_RAM, R16             ; Se guardan los datos actualizados
228     STS CONTADOR_DE_PULSOS_RAM+1, R17
229     STS CONTADOR_DE_PULSOS_RAM+2, R18
230
231
232     POP R19
233     POP R18
234     POP R17
235     POP R16
236 RET
237
238
239
240 ;
    ********************************************************************************************

241 INICIAR_MEDICION:
242 ; Función que carga la configuración seteada por el usuario en los registros determinados, e
        inicializa el TIMER 1, para realizar las mediciones.
243     PUSH VENTANA_DE_TIEMPO
244
245     ;SBI PORTB, 1                                    ;Se enciende un LED indicando que se
            comenzó a medir (solo para debuggear, luego este LED no se usará)
246
247
248     LDS VENTANA_DE_TIEMPO, REGISTRO_VENTANA_TIEMPO  ;Se carga la configuración de la
            duración de la ventana
249     CALL CONFIG_VENTANA_DE_TIEMPO                   ;Se setea la configuración del timer
            según lo indicado previamente por el usario
250
251     STS MUL_DE_VENTANA_RAM, MUL_DE_VENTANA          ;Se guarda la configuración del
            multiplicador de ventana en RAM.
252
253     CALL CONFIG_REPETIR_VENTANAS                    ;Se carga la configuración seteada
            por el usuario, en los registros correspondientes, para iniciar mediciones,
254                                                     ;de la cantidad de ventanas a medir
255     CALL CONFIG_CONTADOR_DE_PULSOS                  ;Se inicializa el registro R23, donde
             se guarda la cantidad de pulsos, en 0. Además se inicializa el
256                                                     ;contador en RAM, en 0
257
258     POP VENTANA_DE_TIEMPO
259 RET
```

```
260
261 ;    ********************************************************************************************************
262
263
264 ;    ********************************************************************************************************
265 ;    ********************************************************************************************************
266
267 INTERRUPCION_PULSO_DETECTADO:
268 ;Función de interrupción para cuando se detecta un pulso en ICP1.
269 ;Se incrementa el contador de pulsos en 1 y se guarda el instante de tiempo en RAM.
270
271 ;REGISTROS UTILIZADOS: INSTANTE_DE_TIEMPO_LOW, INSTANTE_DE_TIEMPO_HIGH
272     PUSH R16
273     IN R16, SREG
274     PUSH R16
275
276     INC CONTADOR_DE_PULSOS          ;SE INCREMENTA EL CONTADOR DE PULSOS EN 1
277
278     SBR EVENTO, (1<<PULSO_RECIBIDO) ; ACTIVAR EL BIT DEL EVENTO INDICADOR DE QUE SE RECIBIO
        UN PULSO
279
280     POP R16
281     OUT SREG, R16
282     POP R16
283
284 RETI
285
286 ;    ********************************************************************************************************
287
288 INTERRUPCION_FIN_VENTANA:
289 ;Función de interrupción para cuando se detecta el fin de una ventana de tiempos. Se invoca
    cuando el TIMER1 llega a su valor de comparación.
290 ;Se decrementa la cantidad de veces a medir en 1.
291
292 ;REGISTROS UTILIZADOS: VENTANAS_A_MEDIR
293     PUSH R16
294     IN R16, SREG
295     PUSH R16
296
297     ; CONFIGURAR EVENTO DE DISMINUCION DE MUL DE VENTANA
298     LDS R16, EVENTO2
299     SBR R16, (1<<BIT_MUL_VENTANA_DEC)
300     STS EVENTO2, R16
301     ; ======
302
303     DEC MUL_DE_VENTANA
304     BRNE FIN_ISR_VENTANA
305     ;SET
306     POP R16
307     ORI R16, (1<<SREG_T)
308     OUT SREG, R16
309     POP R16
310     RETI
311 FIN_ISR_VENTANA:
312
313     POP R16
314     OUT SREG, R16
315     POP R16
```

```
316
317     RETI
318
319 ;
    ***************************************************************************************
320 VERIFICAR_UMBRAL:
321 ; Verifica si se ha superado el valor del umbral y enciende el LED/BUZZER si eso ha ocurrido
322 ; Para encender o no el buzzer chequear el registro REGISTRO_CONF_GENERAL
323 ; Recibe: CONTADOR_DE_PULSOS, REGISTRO_UMBRAL, REGISTRO_CONF_GENERAL
324 ; Devuelve: -
325
326     PUSH R16
327
328     ; Comparar LSB
329     LDS R16, REGISTRO_UMBRAL
330     CP R16, CONTADOR_DE_PULSOS
331
332     ; Si el valor de CONTADOR_DE_PULSOS es mayor al umbral, encender LED/BUZZER
333     ; sino, apagarlos y retornar
334     BRLO _ENCENDER_SENAL_VERIFICAR_UMBRAL
335
336     ; Apagar LED
337     CBI PORTB, PIN_PORTB_LED
338
339     ; Apagar buzzer
340     CBI PORTB, PIN_PORTB_BUZZER
341
342     RJMP _RET_VERIFICAR_UMBRAL
343
344 _ENCENDER_SENAL_VERIFICAR_UMBRAL:
345     ; Encender LED
346     SBI PORTB, PIN_PORTB_LED
347
348     ; Chequear si se ha configurado el buzzer para encenderse cuando se supera el umbral
349     LDS R16, REGISTRO_CONF_GENERAL
350     SBRC R16, BIT_SENAL_SONORA
351     SBI PORTB, 3
352
353 _RET_VERIFICAR_UMBRAL:
354     POP R16
355     RET
356
357 ;
    ***************************************************************************************
358 VERIFICAR_APAGADO_TUBO:
359 ; Verifica si se indicado al dispositivo que apague el tubo del contador
360 ; Para eso, chequea REGISTRO_CONF_GENERAL
361 ; Recibe: REGISTRO_CONF_GENERAL
362 ; Devuelve: -
363 ; NOTA: EL TUBO DEL CONTADOR SE ENCIENDE PONIENDO UN '0' EN EL PUERTO DE SALIDA
364     LDS R16, REGISTRO_CONF_GENERAL
365     SBRC R16, BIT_ACTIVAR_FUENTE
366     RJMP _DESACTIVAR_FUENTE
367
368     ; Encender el tubo del contador
369     SBI PORTB, PUERTO_ACTIVAR_CONTADOR
370
371     RET
372
373 _DESACTIVAR_FUENTE:
374     CBI PORTB, PUERTO_ACTIVAR_CONTADOR
375     RET
```

```
; Descripcion: registro de desplazamiento destinado a almacenar los tiempos de
; pulsos que arriban al contador, para luego ser enviados por UART

; =====================================================================
; ========================= Registros reservados ======================
; =====================================================================
.def CANT_CARACTERES_GUARDADOS = R5

; =====================================================================
; ========================= Registros auxiliares ======================
; =====================================================================
.undef T0
.undef T1
.undef T2
.undef T3
.def T0 = R16
.def T1 = R17
.def T2 = R18
.def T3 = R19

; =====================================================================
; ============================ Funciones ==============================
; =====================================================================

; Descripcion: recibe una cadena de 6 caracteres almacenada en FLASH y la
; carga en el registro de desplazamiento
; Recibe: puntero Z con la direccion de memoria de la cadena de caracteres en FLASH
; Devuelve: -
CARGAR_CADENA_REGISTRO_DESPLAZAMIENTO:
    PUSH T0
    PUSH T1
    PUSH XL
    PUSH XH
    ; Cargar comienzo del buffer
    LDI XH, HIGH(REGISTRO_DESPLAZAMIENTO)
    LDI XL, LOW(REGISTRO_DESPLAZAMIENTO)

    ;Ir al final del buffer
    CLR T0
    ADD XL, CANT_CARACTERES_GUARDADOS
    ADC XH, T0

    ; Cargar cadena en el registro de desplazamiento
    LDI T0, 6
_BUCLE_CARGAR_CADENA_REGISTRO_DESPLAZAMIENTO:
    LPM T1, Z+
    ST X+, T1
    DEC T0
    BRNE _BUCLE_CARGAR_CADENA_REGISTRO_DESPLAZAMIENTO

    LDI T0, 6
    ADD CANT_CARACTERES_GUARDADOS, T0 ; Se han introducido 6 nuevos caracteres en el registro

    POP XH
    POP XL
    POP T1
    POP T0
    RET

; Descripcion: transforma el tiempo de un pulso a ASCII y
; lo guarda al final de un registro de desplazamiento
; Recibe: valor del tiempo del pulso (16 bits) en los registros R17 y R16
; Devuelve: -
CARGAR_PULSO_REGISTRO_DESPLAZAMIENTO:
    PUSH T2
```

```
66      PUSH XL
67      PUSH XH
68      ; Cargar comienzo del buffer
69      LDI XH, HIGH(REGISTRO_DESPLAZAMIENTO)
70      LDI XL, LOW(REGISTRO_DESPLAZAMIENTO)
71
72      ;Ir al final del buffer
73      CLR T2
74      ADD XL, CANT_CARACTERES_GUARDADOS
75      ADC XH, T2
76
77      ; Transformar el pulso (que esta guardado en los registros R17 y R16) de binario
78      ; a ASCII y guardar en (REGISTRO_DESPLAZAMIENTO + CANT_CARACTERES_GUARDADOS)
79      ; Este ocupara 6 bytes (incluyendo un caracter nulo)
80      CALL DEC_TO_ASCII_16_BITS
81      LDI T2, 6
82      ADD CANT_CARACTERES_GUARDADOS, T2 ; Se han introducido 6 nuevos caracteres en el registro
83
84      POP XH
85      POP XL
86      POP T2
87
88      RET
89
90  ; ================================================================
91  ; Descripcion: toma 6 bytes del registro de desplazamiento y los
92  ; envia por UART
93  ; Recibe: -
94  ; Devuelve: -
95  QUITAR_PULSO_REGISTRO_DESP_Y_ENVIAR_UART:
96
97      PUSH T0
98      PUSH T1
99      PUSH T2
100     PUSH T3
101     PUSH XL
102     PUSH XH
103     PUSH YL
104     PUSH YH
105     ; Si no hay pulsos guardados en el registro, entonces retornar
106     CLR T0
107     CP CANT_CARACTERES_GUARDADOS, T0;
108     BREQ _RET_QUITAR_PULSO_REGISTRO_DESP_Y_ENVIAR_UART
109
110     ; Cargar comienzo del buffer
111     LDI XH, HIGH(REGISTRO_DESPLAZAMIENTO)
112     LDI XL, LOW(REGISTRO_DESPLAZAMIENTO)
113
114     ; Enviar un tiempo de pulso
115     MOV R19, XH
116     MOV R18, XL
117
118     ; Enviar el valor del tiempo por UART
119     LDI R20, AGREGAR_NEWLINE ; Enviar una coma despues de cada tiempo
120     CALL CARGAR_BUFFER_DESDE_RAM
121     CALL ENVIAR_DATOS_UART
122
123     ; Al emitir el tiempo de un pulso, se han removido 6 caracteres del registro
124     LDI T0, 6
125     SUB CANT_CARACTERES_GUARDADOS, T0
126
127     ; Si se han removido todos los tiempo de pulso del registro, entonces regresar
128     CLR T0
129     CP CANT_CARACTERES_GUARDADOS, T0;
130     BREQ _RET_QUITAR_PULSO_REGISTRO_DESP_Y_ENVIAR_UART
131
```

```
132     LDI T0,  6; Cargar en T0 la cantidad de caracteres comprendidos por el tiempo del pulso
133           ; (5 digitos mas \0)
134
135 _BUCLE_1_QUITAR_PULSO:
136
137     ; Ir al comienzo del registro
138     LDI XH, HIGH(REGISTRO_DESPLAZAMIENTO)
139     LDI XL, LOW(REGISTRO_DESPLAZAMIENTO)
140
141     ; Ir al comienzo del siguiente tiempo almacenado en el registro
142     CLR T2
143     ADD XL, T0
144     ADC XH, T2
145
146
147     LDI T2, 1 ; SACAR
148     CLR T3
149     MOVW YH:YL, XH:XL
150     LD T3, -X ; Dummy read para que X apunte a la posicion anterior al comienzo del siguiente
            tiempo,
151                 ; mientras que Y lo haga al primer caracter es este ultimo
152
153     MOV T1, CANT_CARACTERES_GUARDADOS
154     ; Desplazar cada caracter del buffer una vez hacia arriba en la memoria
155 _BUCLE_2_QUITAR_PULSO:
156     LD T3, Y+
157     ST X+, T3
158     DEC T1
159     BRNE _BUCLE_2_QUITAR_PULSO
160
161     DEC T0
162     BRNE _BUCLE_1_QUITAR_PULSO
163     ; Desplazar el registro segun la cantidad de pulsos que hayan almacenados
164
165
166
167 _RET_QUITAR_PULSO_REGISTRO_DESP_Y_ENVIAR_UART:
168
169     POP YH
170     POP YL
171     POP XH
172     POP XL
173     POP T3
174     POP T2
175     POP T1
176     POP T0
177
178     RET
179
180 .dseg
181 ; Registro que permite almacenar el tiempo de 60 pulsos, ya que cada uno ocupa
182 ; 6 bytes (tienen un maximo valor de 50.000)
183 REGISTRO_DESPLAZAMIENTO: .BYTE 60
184
185 .cseg
```

scpi.asm

```
1 ; Descripcion: funciones para interpretacion de la informacion recibida mediante UART
2 ; a traves del protocolo scpi
3 ;
      ================================================================================================
4 ; Comandos de acceso remoto para el control del dispositivo
5 ;
6 ; Todo comando es una secuencia de carcateres ASCII terminados en '\n'.
```

```
7  ; El "controlador" es cualquier ente que se comunica por RS-232 con el dispositivo usando la
     codificación 1-8-1
8  ; (start-data-stop) a una velocidad de 76800 bps.
9  ;
10 ; *IDN?   (El controlador) pide la identificación del dispositivo
11 ; (el dispositivo) devuelve "...."
12 ;
13 ; *RST  Reestablece los valores por default de ventana de tiempo, tiempo total de medición y
     umbral
14 ; parar la medición que se estuviera realizando en ese instante, al igual que la transmisión
     de datos hacia la PC.
15 ;
16 ; ABORt Abortar una medición en curso
17 ;
18 ; READ?  --- si va con signo de pregunta, debería devolver un parámetro
19 ; READ --- Hace una (única) medición y retorna el resultado
20 ; Función: comenzar a contar pulsos y devolver a la PC el valor de la cantidad que se
     detectaron cada vez que
21 ; finalice una ventana de tiempo. Repetir hasta que haya transcurrido el tiempo total de
     medición configurado.
22 ;
23 ; CONFigure?
24 ; Funcion: Devuelve los valores de ventana de tiempo, el intervalo de medición y el umbral.
25 ;
26 ; CONFigure:WINDow N
27 ; Function: Fija un valor de ventana con N en Segundos
28 ; CONFigure:WINDow?
29 ; Function: Devuelve el valor de ventana
30 ;
31 ; CONFigure:COUNt N
32 ; Funcion: Cambiar el tiempo total de la medición a N segundos
33 ; CONFigure:COUNt?
34 ; Funcion: Informa el tiempo total fijado para una medición
35 ;
36 ; CONFigure:TRIGger N
37 ; Funcion: Configura el umbral de encendido del LED.
38 ; CONFigure:TRIGger?
39 ; Funcion: Devuelve el umbral de encendido del LED.
40 ;
41 ; CONFigure:DATA <BOOLEAN>
42 ; Funcion: configura el dispositivo para el enviar los valores de los tiempos en vez
43 ; del total por ventana (SOLO PARA UART)
44 ; CONFigure:DATA?
45 ; Funcion: Informa si el dispositivo se encuentra configurado para enviar los tiempos de los
     pulsos al medir
46 ;
47 ; CONTrol:APOWer
48 ; Función: encender o apagar el circuito del tubo.
49 ; CONTrol:APOWer?
50 ; Informa si el tubo del contador Geiger se encuentra encendido
51 ;
52 ; SYSTem:BEEPer:STATe <BOOLEAN>
53 ; Funcion: controla el encendido/apagado de la senal sonora
54 ; Si recibe un 1 (ON), se bloquea el teclado. Si es 0 (OFF), se desbloquea
55 ; SYSTem:BEEPer:STATe?
56 ; Funcion: Informa si el buzzer se encuentra configurado para hacer ruido al superarse un
     umbral dado
57 ;
58 ; *SAV
59 ; Función: guardar la última configuración de la ventana de tiempo, el tiempo total de
     medición y el umbral
60 ; en memoria para que sean automáticamente adoptados una vez que el dispositivo se resetee.
61 ;
62 ; SYSTem:KLOCk <BOOLEAN>
63 ; Funcion: inhabilitar teclado durante la medicion
64 ; Si recibe un 1 (ON), se bloquea el teclado. Si es 0 (OFF), se desbloquea
```

```
65 ; SYSTem:KLOCk?
66 ; Funcion: Informa si el teclado se encuentra configurado para bloquearse durante una
       medicion
67 ;
68 ;
69 ; CONCEPTOS IMPORTANTES:
70 ; En este archivo se identifican los comandos por niveles. Nivel 0 se corresponde
71 ; con un comando perteneciente a un systema según el estandar SCPI, es decir,
72 ; el primer comando que podria aparecer en una secuencia de comandos.
73 ; Por ejemplo, en CONFigure:WINDow, CONFigure sería el comando de nivel 0.
74 ; Por otra parte, los comandos que le siguen hacia la derecha se identifican como
75 ; de nivel 1,2,3...,N. En el ejemplo, WINDow sería un comando de nivel 1.
76 ;
77 ; Se entiende por una cadena de comandos a una secuencia de comandos
78 ; separada por el caracter ':'. Por ejemplo: CONTrol:APOWer es una cadena
79 ; de 2 comandos, siendo CONTrol y APOWer el segundo
80
81 ; ======================================================================
82 ; ========================= Constantes  ================================
83 ; ======================================================================
84 ; Valor asociado a cada comando para las opciones de lo switches
85 ; --> Comandos de nivel 0:
86 .equ COMANDO_IDN = 0
87 .equ COMANDO_RST = 1
88 .equ COMANDO_SAV = 2
89 .equ COMANDO_ABORT_1 = 3
90 .equ COMANDO_ABORT_2 = 4
91 .equ COMANDO_READ = 5
92 .equ COMANDO_CONF_1 = 6
93 .equ COMANDO_CONF_2 = 8
94 .equ COMANDO_CONF_QUERY_1 = 7
95 .equ COMANDO_CONF_QUERY_2 = 9
96 .equ COMANDO_MEMORY_1 = 10
97 .equ COMANDO_MEMORY_2 = 11
98 .equ COMANDO_CONTROL_1 = 12
99 .equ COMANDO_CONTROL_2 = 13
100 .equ COMANDO_STATUS_1 = 14
101 .equ COMANDO_STATUS_2 = 15
102 .equ COMANDO_SYSTEM_1 = 16
103 .equ COMANDO_SYSTEM_2 = 17
104
105 ; -> Comandos de nivel 1 de CONFigure:
106 .equ COMANDO_WINDOW_1 = 0
107 .equ COMANDO_WINDOW_2 = 1
108 .equ COMANDO_WINDOW_QUERY_1 = 2
109 .equ COMANDO_WINDOW_QUERY_2 = 3
110 .equ COMANDO_COUNT_1 = 4
111 .equ COMANDO_COUNT_2 = 5
112 .equ COMANDO_COUNT_QUERY_1 = 6
113 .equ COMANDO_COUNT_QUERY_2 = 7
114 .equ COMANDO_TRIGGER_1 = 8
115 .equ COMANDO_TRIGGER_2 = 9
116 .equ COMANDO_TRIGGER_QUERY_1 = 10
117 .equ COMANDO_TRIGGER_QUERY_2 = 11
118 .equ COMANDO_DATA = 12
119 .equ COMANDO_DATA_QUERY = 13
120
121 ; -> Comandos de nivel 1 de CONTrol:
122 .equ COMANDO_APOWER_1 = 0
123 .equ COMANDO_APOWER_2 = 1
124 .equ COMANDO_APOWER_QUERY_1 = 2
125 .equ COMANDO_APOWER_QUERY_2 = 3
126
127 ; -> Comandos de nivel 1 de CONDition:
128 .equ COMANDO_CONDITION_QUERY_1 = 0
129 .equ COMANDO_CONDITION_QUERY_2 = 1
```

```
130
131 ; -> Comandos de nivel 1 de SYSTem:
132 .equ COMANDO_SYSTEM_BEEPER_1 = 0
133 .equ COMANDO_SYSTEM_BEEPER_2 = 1
134 .equ COMANDO_SYSTEM_BEEPER_QUERY_1 = 2
135 .equ COMANDO_SYSTEM_BEEPER_QUERY_2 = 3
136 .equ COMANDO_SYSTEM_KLOCK_1 = 4
137 .equ COMANDO_SYSTEM_KLOCK_2 = 5
138 .equ COMANDO_SYSTEM_KLOCK_QUERY_1 = 6
139 .equ COMANDO_SYSTEM_KLOCK_QUERY_2 = 7
140
141 ; -> Comandos de nivel 2 de BEEPer:
142 .equ COMANDO_SYSTEM_STATE_1 = 0
143 .equ COMANDO_SYSTEM_STATE_2 = 1
144 .equ COMANDO_SYSTEM_STATE_QUERY_1 = 2
145 .equ COMANDO_SYSTEM_STATE_QUERY_2 = 3
146
147 ; Numero de comandos por nivel
148 .equ TOTAL_COMANDOS_NIVEL_0 = 18
149 .equ TOTAL_COMANDOS_NIVEL_1_CONFIGURE = 14
150 .equ TOTAL_COMANDOS_NIVEL_1_MEMORY = 1
151 .equ TOTAL_COMANDOS_NIVEL_1_CONTROL = 4
152 .equ TOTAL_COMANDOS_NIVEL_1_STATUS = 2
153 .equ TOTAL_COMANDOS_NIVEL_1_SYSTEM = 8
154 .equ TOTAL_COMANDOS_NIVEL_2_BEEPER = 4
155
156 ; Largo en caracteres de comando mas largo
157 ; NOTA: debe ser actualizado a un nuevo valor
158 ; cuando se incorpore un comando con un largo mayor
159 ; al maximo preexistente
160 .equ MAX_TAMANO_COMANDO = 10
161
162 ; =====================================================================
163 ; ========================= Registros auxiliares ======================
164 ; =====================================================================
165 .undef T0
166 .undef T1
167 .undef T2
168 .undef T3
169 .undef T4
170 .undef T5
171 .def T0 = R16
172 .def T1 = R17
173 .def T2 = R18
174 .def T3 = R19
175 .def T4 = R20
176 .def T5 = R21
177
178 ; =====================================================================
179 ; ========================== Macros ===================================
180 ; =====================================================================
181 ; Macro para reducir el tamano de cada switch
182 ; Parametros:
183 ; -> @0 valor a comparar con @1
184 ; -> @1 valor a comparar con @0
185 ; -> @2 label del espacio de memoria a donde saltar mediante JMP
186 .MACRO JMP_IF_EQUAL
187     CPI @0, @1
188     IN T3, SREG
189     SBRC T3, SREG_Z
190     JMP @2
191 .ENDMACRO
192
193
194 ; =====================================================================
195 ; ========================== Funciones ================================
```

```
196 ; ====================================================================
197
198 INTERPRETAR_CADENA_COMANDOS:
199
200     PUSH_WORD  T0, T1
201     PUSH_WORD  T2, T3
202     PUSH_WORD  T4, T5
203
204     PUSH_WORD XL, XH
205     PUSH_WORD YL, YH
206     PUSH_WORD ZL, ZH
207
208     ; CODIGO DE PRUEBA
209 /*  MOVW XH:XL, PTR_RX_H:PTR_RX_L
210
211     ldi r16, 7
212     mov BYTES_RECIBIDOS, r16
213
214     LDI R16, 'S'
215     ST X+, R16
216     LDI R16, 'T'
217     ST X+, R16
218     LDI R16,'A'
219     ST X+, R16
220     LDI R16, 'T'
221     ST X+, R16
222     LDI R16, 'u'
223     ST X+, R16
224     LDI R16, 's'
225     ST X+, R16
226     LDI R16, '\r'
227     ST X+, R16
228     LDI R16,'\n'
229     ST X+, R16  */
230     ; ======*/
231
232
233     ; Cargar parámetros de la funcion INTERPRETAR_COMANDO para parsear
234     ; el primer comando de la cadena
235     MOVW XH:XL, PTR_RX_H:PTR_RX_L
236     LDI T4, LOW(COMANDOS_NIVEL_0)
237     LDI T5, HIGH(COMANDOS_NIVEL_0)
238     LDI T2, TOTAL_COMANDOS_NIVEL_0
239
240     CALL INTERPRETAR_COMANDO
241
242     ; Recuperar en T0 el indice de la tabla en memoria flash del comando recibido
243     LDI T0, TOTAL_COMANDOS_NIVEL_0
244     SUB T0, T2
245
246     ; Verificar que no se este midiendo
247     SBRC ESTADO, EST_MEDIR_DEVOLVER_TIEMPOS
248     RJMP _CHEQUEAR_ABORT
249     SBRC ESTADO, EST_MEDIR_DEVOLVER_TOTAL
250     RJMP _CHEQUEAR_ABORT
251
252     ; Si el indice el valor devuelto en T2 es 0,
253     ; entonces no se ha recibido un comando correcto
254     ; emitir un mensaje de error
255     JMP_IF_EQUAL T2, 0, _RETORNAR_ERROR_INTERPRETAR_CADENA_COMANDOS
256
257     ; ==========================
258     ; Iniciar switch para identificar cual fue el primer comando
259
260 _OPCION_COMANDO_IDN:
261     JMP_IF_EQUAL T0, COMANDO_IDN, _COMANDO_IDN
```

```
262
263 _OPCION_COMANDO_RST:
264     JMP_IF_EQUAL T0, COMANDO_RST, _COMANDO_RST
265
266 _OPCION_COMANDO_SAV:
267     JMP_IF_EQUAL T0, COMANDO_SAV, _COMANDO_SAV
268
269 _OPCION_COMANDO_READ:
270     JMP_IF_EQUAL T0, COMANDO_READ, _COMANDO_READ
271
272 _OPCION_COMANDO_CONF:
273     JMP_IF_EQUAL T0, COMANDO_CONF_1, _SWITCH_COMANDO_CONF
274     JMP_IF_EQUAL T0, COMANDO_CONF_2, _SWITCH_COMANDO_CONF
275
276 _OPCION_COMANDO_CONF_QUERY:
277     JMP_IF_EQUAL T0, COMANDO_CONF_QUERY_1, _COMANDO_CONF_QUERY
278     JMP_IF_EQUAL T0, COMANDO_CONF_QUERY_2, _COMANDO_CONF_QUERY
279
280 /*_OPCION_COMANDO_MEMORY:
281     JMP_IF_EQUAL T0, COMANDO_MEMORY_1, _SWITCH_COMANDO_MEMORY
282     JMP_IF_EQUAL T0, COMANDO_MEMORY_2, _SWITCH_COMANDO_MEMORY*/
283
284 _OPCION_COMANDO_CONTROL:
285     JMP_IF_EQUAL T0, COMANDO_CONTROL_1, _SWITCH_COMANDO_CONTROL
286     JMP_IF_EQUAL T0, COMANDO_CONTROL_2, _SWITCH_COMANDO_CONTROL
287
288 /*_OPCION_COMANDO_STATUS:
289     JMP_IF_EQUAL T0, COMANDO_STATUS_1, _SWITCH_COMANDO_STATUS
290     JMP_IF_EQUAL T0, COMANDO_STATUS_1, _SWITCH_COMANDO_STATUS*/
291
292 _OPCION_COMANDO_SYSTEM:
293     JMP_IF_EQUAL T0, COMANDO_SYSTEM_1, _SWITCH_COMANDO_SYSTEM
294     JMP_IF_EQUAL T0, COMANDO_SYSTEM_2, _SWITCH_COMANDO_SYSTEM
295
296 _OPCION_COMANDO_ABORT:
297     JMP_IF_EQUAL T0, COMANDO_ABORT_1, _COMANDO_ABORT
298     JMP_IF_EQUAL T0, COMANDO_ABORT_2, _COMANDO_ABORT
299
300     JMP _RETORNAR_ERROR_INTERPRETAR_CADENA_COMANDOS
301
302 _CHEQUEAR_ABORT:
303     JMP_IF_EQUAL T0, COMANDO_ABORT_1, _COMANDO_ABORT
304     JMP_IF_EQUAL T0, COMANDO_ABORT_2, _COMANDO_ABORT
305
306     JMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS
307
308 ;
    ================================================================================================
309 ;
    ================================================================================================
310 ; Switches individuales para reconocer los comandos de nivel 1 asociados a cada comando de
    nivel 0
311
312 ; Switch del comando CONFigure
313 _SWITCH_COMANDO_CONF:
314
315     ; Cargar los parametros para la funcion INTERPRETAR_COMANDO
316     LDI T4, LOW(COMANDOS_NIVEL_1_CONFIGURE)
317     LDI T5, HIGH(COMANDOS_NIVEL_1_CONFIGURE)
318     LDI T2, TOTAL_COMANDOS_NIVEL_1_CONFIGURE
319
320     CALL INTERPRETAR_COMANDO
321
322     ; Si el indice el valor devuelto en T2 es 0,
```

```
323     ; entonces no se ha recibido un comando correcto
324     ; emitir un mensaje de error
325     JMP_IF_EQUAL T2, 0, _RETORNAR_ERROR_INTERPRETAR_CADENA_COMANDOS
326 
327     ; Recuperar en T0 el indice de la tabla en memoria flash del comando recibido
328     LDI T0, TOTAL_COMANDOS_NIVEL_1_CONFIGURE
329     SUB T0, T2
330 
331 _OPCION_COMANDO_CONF_WINDOW:
332     JMP_IF_EQUAL T0, COMANDO_WINDOW_1, _COMANDO_CONF_WINDOW
333     JMP_IF_EQUAL T0, COMANDO_WINDOW_2, _COMANDO_CONF_WINDOW
334 
335 _OPCION_COMANDO_CONF_WINDOW_QUERY:
336     JMP_IF_EQUAL T0, COMANDO_WINDOW_QUERY_1, _COMANDO_CONF_WINDOW_QUERY
337     JMP_IF_EQUAL T0, COMANDO_WINDOW_QUERY_2, _COMANDO_CONF_WINDOW_QUERY
338 
339 _OPCION_COMANDO_CONF_COUNT:
340     JMP_IF_EQUAL T0, COMANDO_COUNT_1, _COMANDO_CONF_COUNT
341     JMP_IF_EQUAL T0, COMANDO_COUNT_2, _COMANDO_CONF_COUNT
342 
343 _OPCION_COMANDO_CONF_COUNT_QUERY:
344     JMP_IF_EQUAL T0, COMANDO_COUNT_QUERY_1, _COMANDO_CONF_COUNT_QUERY
345     JMP_IF_EQUAL T0, COMANDO_COUNT_QUERY_2, _COMANDO_CONF_COUNT_QUERY
346 
347 _OPCION_COMANDO_CONF_TRIGGER:
348     JMP_IF_EQUAL T0, COMANDO_TRIGGER_1, _COMANDO_CONF_TRIGGER
349     JMP_IF_EQUAL T0, COMANDO_TRIGGER_2, _COMANDO_CONF_TRIGGER
350 
351 _OPCION_COMANDO_CONF_TRIGGER_QUERY:
352     JMP_IF_EQUAL T0, COMANDO_TRIGGER_QUERY_1, _COMANDO_CONF_TRIGGER_QUERY
353     JMP_IF_EQUAL T0, COMANDO_TRIGGER_QUERY_2, _COMANDO_CONF_TRIGGER_QUERY
354 
355 _OPCION_COMANDO_CONF_DATA:
356     JMP_IF_EQUAL T0, COMANDO_DATA, _COMANDO_CONF_DATA
357 
358 _OPCION_COMANDO_CONF_DATA_QUERY:
359     JMP_IF_EQUAL T0, COMANDO_DATA_QUERY, _COMANDO_CONF_DATA_QUERY
360 
361     RJMP _RETORNAR_ERROR_INTERPRETAR_CADENA_COMANDOS
362 
363 ; ================================================================================
364 ; Switch del comando MEMory
365 /*_SWITCH_COMANDO_MEMORY:
366 
367     ; Cargar los parametros para la funcion INTERPRETAR_COMANDO
368     LDI T4, LOW(COMANDOS_NIVEL_1_MEMORY)
369     LDI T5, HIGH(COMANDOS_NIVEL_1_MEMORY)
370     LDI T2, TOTAL_COMANDOS_NIVEL_1_MEMORY
371 
372     CALL INTERPRETAR_COMANDO
373 
374     ; Si el indice el valor devuelto en T2 es 0,
375     ; entonces no se ha recibido un comando correcto
376     ; emitir un mensaje de error
377     JMP_IF_EQUAL T2, 0, _RETORNAR_ERROR_INTERPRETAR_CADENA_COMANDOS
378 
379     ; Recuperar en T0 el indice de la tabla en memoria flash del comando recibido
380     LDI T0, TOTAL_COMANDOS_NIVEL_1_MEMORY
381     SUB T0, T2
382 
383 _OPCION_COMANDO_MEMORY_DATA_QUERY:
384     JMP_IF_EQUAL T0, COMANDO_DATA_QUERY, _COMANDO_MEMORY_DATA
385 
386     JMP _RETORNAR_ERROR_INTERPRETAR_CADENA_COMANDOS*/
387 
388 ; ================================================================================
```

```
; Switch del comando CONTrol
_SWITCH_COMANDO_CONTROL:

    ; Cargar los parametros para la funcion INTERPRETAR_COMANDO
    LDI T4, LOW(COMANDOS_NIVEL_1_CONTROL)
    LDI T5, HIGH(COMANDOS_NIVEL_1_CONTROL)
    LDI T2, TOTAL_COMANDOS_NIVEL_1_CONTROL

    CALL INTERPRETAR_COMANDO

    ; Si el indice el valor devuelto en T2 es 0,
    ; entonces no se ha recibido un comando correcto
    ; emitir un mensaje de error
    JMP_IF_EQUAL T2, 0, _RETORNAR_ERROR_INTERPRETAR_CADENA_COMANDOS

    ; Recuperar en T0 el indice de la tabla en memoria flash del comando recibido
    LDI T0, TOTAL_COMANDOS_NIVEL_1_CONTROL
    SUB T0, T2

_OPCION_COMANDO_CONTROL_APOWER:
    JMP_IF_EQUAL T0, COMANDO_APOWER_1, _COMANDO_CONTROL_APOWER
    JMP_IF_EQUAL T0, COMANDO_APOWER_2, _COMANDO_CONTROL_APOWER

_OPCION_COMANDO_CONTROL_APOWER_QUERY:
    JMP_IF_EQUAL T0, COMANDO_APOWER_QUERY_1, _COMANDO_CONTROL_APOWER_QUERY
    JMP_IF_EQUAL T0, COMANDO_APOWER_QUERY_2, _COMANDO_CONTROL_APOWER_QUERY

    RJMP _RETORNAR_ERROR_INTERPRETAR_CADENA_COMANDOS


;    ============================================================================================

; Switch del comando STATus
/*_SWITCH_COMANDO_STATUS:

    ; Cargar los parametros para la funcion INTERPRETAR_COMANDO
    LDI T4, LOW(COMANDOS_NIVEL_1_STATUS)
    LDI T5, HIGH(COMANDOS_NIVEL_1_STATUS)
    LDI T2, TOTAL_COMANDOS_NIVEL_1_STATUS

    CALL INTERPRETAR_COMANDO

    ; Si el indice el valor devuelto en T2 es 0,
    ; entonces no se ha recibido un comando correcto
    ; emitir un mensaje de error
    CPI T2, 0
    IN T3, SREG
    SBRC T3, SREG_Z
    JMP _RETORNAR_ERROR_INTERPRETAR_CADENA_COMANDOS

    ; Recuperar en T0 el indice de la tabla en memoria flash del comando recibido
    LDI T0, TOTAL_COMANDOS_NIVEL_1_STATUS
    SUB T0, T2

_OPCION_COMANDO_CONDITION_STATUS:
    JMP_IF_EQUAL T0, COMANDO_CONDITION_QUERY_1, _COMANDO_CONDITION_STATUS
    JMP_IF_EQUAL T0, COMANDO_CONDITION_QUERY_2, _COMANDO_CONDITION_STATUS*/

;    ============================================================================================

; Switch del comando SYSTem
_SWITCH_COMANDO_SYSTEM:

    ; Cargar los parametros para la funcion INTERPRETAR_COMANDO
```

```
451     LDI T4, LOW(COMANDOS_NIVEL_1_SYSTEM)
452     LDI T5, HIGH(COMANDOS_NIVEL_1_SYSTEM)
453     LDI T2, TOTAL_COMANDOS_NIVEL_1_SYSTEM
454 
455     CALL INTERPRETAR_COMANDO
456 
457     ; Si el indice el valor devuelto en T2 es 0,
458     ; entonces no se ha recibido un comando correcto
459     ; emitir un mensaje de error
460     CPI T2, 0
461     IN T3, SREG
462     SBRC T3, SREG_Z
463     JMP _RETORNAR_ERROR_INTERPRETAR_CADENA_COMANDOS
464 
465     ; Recuperar en T0 el indice de la tabla en memoria flash del comando recibido
466     LDI T0, TOTAL_COMANDOS_NIVEL_1_SYSTEM
467     SUB T0, T2
468 
469 _OPCION_COMANDO_SYSTEM_BEEPER:
470     JMP_IF_EQUAL T0, COMANDO_SYSTEM_BEEPER_1, _SWITCH_COMANDO_BEEPER
471     JMP_IF_EQUAL T0, COMANDO_SYSTEM_BEEPER_2, _SWITCH_COMANDO_BEEPER
472 
473 _OPCION_COMANDO_SYSTEM_KLOCK:
474     JMP_IF_EQUAL T0, COMANDO_SYSTEM_KLOCK_1, _COMANDO_SYSTEM_KLOCK
475     JMP_IF_EQUAL T0, COMANDO_SYSTEM_KLOCK_2, _COMANDO_SYSTEM_KLOCK
476 
477 _OPCION_COMANDO_SYSTEM_KLOCK_QUERY:
478     JMP_IF_EQUAL T0, COMANDO_SYSTEM_KLOCK_QUERY_1, _COMANDO_SYSTEM_KLOCK_QUERY
479     JMP_IF_EQUAL T0, COMANDO_SYSTEM_KLOCK_QUERY_2, _COMANDO_SYSTEM_KLOCK_QUERY
480 
481 ;    ==========================================================================================
482 ; Switch del comando SYSTem
483 _SWITCH_COMANDO_BEEPER:
484 
485     ; Cargar los parametros para la funcion INTERPRETAR_COMANDO
486     LDI T4, LOW(COMANDOS_NIVEL_2_BEEPER)
487     LDI T5, HIGH(COMANDOS_NIVEL_2_BEEPER)
488     LDI T2, TOTAL_COMANDOS_NIVEL_2_BEEPER
489 
490     CALL INTERPRETAR_COMANDO
491 
492     ; Si el indice el valor devuelto en T2 es 0,
493     ; entonces no se ha recibido un comando correcto
494     ; emitir un mensaje de error
495     CPI T2, 0
496     IN T3, SREG
497     SBRC T3, SREG_Z
498     JMP _RETORNAR_ERROR_INTERPRETAR_CADENA_COMANDOS
499 
500     ; Recuperar en T0 el indice de la tabla en memoria flash del comando recibido
501     LDI T0, TOTAL_COMANDOS_NIVEL_2_BEEPER
502     SUB T0, T2
503 
504 _OPCION_COMANDO_SYSTEM_STATE:
505     JMP_IF_EQUAL T0, COMANDO_SYSTEM_STATE_1, _COMANDO_SYSTEM_BEEPER_STATE
506     JMP_IF_EQUAL T0, COMANDO_SYSTEM_STATE_2, _COMANDO_SYSTEM_BEEPER_STATE
507 
508 _OPCION_COMANDO_SYSTEM_STATE_QUERY:
509     JMP_IF_EQUAL T0, COMANDO_SYSTEM_STATE_QUERY_1, _COMANDO_SYSTEM_BEEPER_STATE_QUERY
510     JMP_IF_EQUAL T0, COMANDO_SYSTEM_STATE_QUERY_2, _COMANDO_SYSTEM_BEEPER_STATE_QUERY
511 
512 ; Fin de switches para comandos de nivel 1
513 
514 
```

```
515 ; ==========================================================================================
516 ; Acciones a realizarse segun con que opcion de cada switch se haya correspondido
517 ; la cadena de comandos
518
519 ; Acciones asociadas a comandos de nivel 0
520 _COMANDO_IDN:
521     CALL ENVIAR_IDENTIDAD
522     RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS
523
524 _COMANDO_RST:
525     CALL CONFIGURAR_REGISTROS_DEFAULT
526     RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS
527
528 _COMANDO_SAV:
529     CALL GUARDAR_CONFIGURACION_EEMPROM ; TODO: incorporar funcion de reset
530     RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS
531
532 _COMANDO_ABORT:
533     CALL CONFIGURAR_EVENTO_DETENER_MEDICION
534     RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS
535
536 _COMANDO_READ:
537     CALL CONFIGURAR_EVENTO_INICIAR_MEDICION
538     RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS
539
540 ; Acciones asociadas a comandos de nivel 1 del sistema CONFigure
541 _COMANDO_CONF_QUERY:
542     CALL DEVOLVER_CONFIGURACION
543     RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS
544
545 _COMANDO_CONF_WINDOW:
546     CALL CONFIGURAR_VENTANA_TIEMPO
547     RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS
548
549 _COMANDO_CONF_WINDOW_QUERY:
550     CALL DEVOLVER_VENTANA_TIEMPO
551     RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS
552
553 _COMANDO_CONF_COUNT:
554     LDI YL, LOW(VENTANAS_A_MEDIR_RAM)
555     LDI YH, HIGH(VENTANAS_A_MEDIR_RAM)
556
557     CALL VALIDAR_ASCII_UART
558     BRCS __ERR_COMANDO_CONF_COUNT
559     CALL TRANSFORMAR_DE_ASCII_A_BINARIO_Y_GUARDAR_5_DIGITOS
560     RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS
561
562 __ERR_COMANDO_CONF_COUNT:
563     CALL ENVIAR_ERROR_PARSER
564     RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS
565
566 _COMANDO_CONF_COUNT_QUERY:
567     CALL DEVOLVER_NUMERO_VENTANAS
568     RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS
569
570 _COMANDO_CONF_TRIGGER:
571     LDI YL, LOW(REGISTRO_UMBRAL)
572     LDI YH, HIGH(REGISTRO_UMBRAL)
573
574     CALL VALIDAR_ASCII_UART
575     BRCS __ERR_COMANDO_CONF_TRIGGER
576     CALL TRANSFORMAR_DE_ASCII_A_BINARIO_Y_GUARDAR_5_DIGITOS
577     RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS
578
```

```
__ERR_COMANDO_CONF_TRIGGER:
    CALL ENVIAR_ERROR_PARSER
    RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS

_COMANDO_CONF_TRIGGER_QUERY:
    CALL DEVOLVER_UMBRAL
    RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS

_COMANDO_CONF_DATA:
    CALL CONFIGURAR_ENVIO_TIEMPOS_PULSOS
    RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS

_COMANDO_CONF_DATA_QUERY:
    LDI R18, LOW(MENSAJE_ESTADO_ENVIAR_TIEMPOS)
    LDI R19, HIGH(MENSAJE_ESTADO_ENVIAR_TIEMPOS)
    LDI R17, (1<<BIT_ENVIAR_TIEMPO_PULSOS)

    CALL DEVOLVER_CONF_BIT

    RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS

; Acciones asociadas a comandos de nivel 1 del sistema CONTrol
_COMANDO_SYSTEM_BEEPER_STATE:
    CALL CONFIGURAR_SENAL_SONORA
    RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS

_COMANDO_SYSTEM_BEEPER_STATE_QUERY:
    LDI R18, LOW(MENSAJE_ESTADO_SENAL_SONORA)
    LDI R19, HIGH(MENSAJE_ESTADO_SENAL_SONORA)
    LDI R17, (1<<BIT_SENAL_SONORA)

    CALL DEVOLVER_CONF_BIT
    RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS

_COMANDO_CONTROL_APOWER:
    CALL CONFIGURAR_APAGADO_FUENTE ;
    RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS

_COMANDO_CONTROL_APOWER_QUERY:
    LDI R18, LOW(MENSAJE_ESTADO_FUENTE)
    LDI R19, HIGH(MENSAJE_ESTADO_FUENTE)
    LDI R17, (1<<BIT_ACTIVAR_FUENTE)

    CALL DEVOLVER_CONF_BIT
    RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS

; Acciones asociadas a comandos de nivel 1 del sistema STAtus
/*_COMANDO_STATUS_CONDITION:
    CALL ENVIAR_IDENTIDAD ; TODO: incorporar funcion para retornar datos guardados en memoria
    RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS*/

; Acciones asociadas a comandos de nivel 1 del sistema MEMory
_COMANDO_SYSTEM_KLOCK:
    CALL CONFIGURAR_BLOQUEO_TECLADO
    RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS

_COMANDO_SYSTEM_KLOCK_QUERY:
    LDI R18, LOW(MENSAJE_ESTADO_BLOQUEO_TECLADO)
    LDI R19, HIGH(MENSAJE_ESTADO_BLOQUEO_TECLADO)
    LDI R17, (1<<BIT_BLOQUEO_TECLADO)

    CALL DEVOLVER_CONF_BIT
    RJMP _RETORNAR_INTERPRETAR_CADENA_COMANDOS

; =====================
; Retorno de la funcion INTERPRETAR_CADENA_COMANDOS
```

```
645 _RETORNAR_ERROR_INTERPRETAR_CADENA_COMANDOS:
646     CALL ENVIAR_ERROR_PARSER
647
648 _RETORNAR_INTERPRETAR_CADENA_COMANDOS:
649     POP_WORD ZL, ZH
650     POP_WORD YL, YH
651     POP_WORD XL, XH
652
653     POP_WORD T4, T5
654     POP_WORD T2, T3
655     POP_WORD T0, T1
656
657     RET
658
659 ; Fin INTERPRETAR_CADENA_COMANDOS
660
661
662 ; ====================================================================
663 ; ================ Funciones a ejecutar por comando ==================
664 ; ====================================================================
665 ; NOTA: este bloque permanece en este archivo solamente con fines de prueba,
666 ; es probable que las funciones a ejecutar asociadas a cada cadena de comandos
667 ; se encuentren en archivos separados asociados a distintos componentes del
668 ; dispositivo
669
670 ENVIAR_IDENTIDAD:
671     PUSH R18
672     PUSH R19
673
674     LDI R18, LOW(MENSAJE_IDENTIDAD)
675     LDI R19, HIGH(MENSAJE_IDENTIDAD)
676     CALL CARGAR_BUFFER
677
678     ; Activar interrupcion por buffer UDR0 vacio para
679     ; enviar el contenido del buffer
680     CALL ACTIVAR_INT_TX_UDRE0
681
682     POP R19
683     POP R18
684     RET
685
686 ; ====================================================================
687 ; ========================= Funciones ================================
688 ; ====================================================================
689
690 ; Descripcion: parsea un comando almacenado en el buffer de RX, y ejecuta
691 ; una funcion si este es valido o devuelve un mensaje de error si no lo es
692 ; Entrada: BUFFER_RX, BYTES_RECIBIDOS
693 ; Devuelve: -
694 INTERPRETAR_COMANDO:
695
696     LSL T4
697     ROL T5
698     ;MOV YL, T4
699     ;MOV YH, T5
700     MOV ZL, T4
701     MOV ZH, T5
702
703     LPM T3, Z+ ;Adquirir el largo del comando almacenado en memoria
704
705     MOV T4, XL ; Guardar el comienzo del primer caracter almacenado en memoria
706     MOV T5, XH
707
708 __BUCLE_INT_COMANDO_IEEE:
709     LD T0, X+
710     ; Si se llego al caracter '\r' o ':', se termino de procesar el comando
```

```
711
712     CPI T3, 0; Si el comando recibido tiene un largo mayor a aquel con el que se esta
713               ; comparando, entonces pasar a comparar con el siguiente
714     ;CPI T3, MAX_TAMANO_COMANDO+1
715     BREQ  _FIN_COMANDO_EN_MEMORIA
716
717     DEC T3
718
719     LPM T1, Z+
720
721     ; Comparar caracter de los comandos, si son distintos saltar a la posicion
722     ; en memoria del siguiente comando almacenado en FLASH
723     CP T0, T1
724     BRNE ___SIGUIENTE_COMANDO
725     RJMP __BUCLE_INT_COMANDO_IEEE
726
727 _FIN_COMANDO_EN_MEMORIA:
728     CPI T0, '\r'
729     BREQ _RET_INTERPRETAR_COMANDO
730     CPI T0, ':'
731     BREQ _RET_INTERPRETAR_COMANDO
732     CPI T0, ' '
733     BREQ _RET_INTERPRETAR_COMANDO
734
735 ___SIGUIENTE_COMANDO:
736     DEC T2
737     BREQ _RET_INTERPRETAR_COMANDO
738
739
740
741     ADD ZL, T3
742     LDI T3, 0
743     ADC ZH, T3
744
745     LPM T3, Z+
746
747     ; Incrementar el puntero Y que almacena el origen del comando en memoria
748 ;   LDI T0, MAX_TAMANO_COMANDO
749 ;   ADD YL, T0
750 ;   LDI T0, 0
751 ;   ADC YH, T0
752
753     ; Transferir la posicion del comando a Z
754 ;   MOV ZL, YL
755 ;   MOV ZH, YH
756
757     ; Volver al inicio del comando
758     ;CLR T3
759     MOVW XH:XL, T5:T4
760
761     RJMP __BUCLE_INT_COMANDO_IEEE
762
763 ; Fin ___SIGUIENTE_COMANDO
764 ; Fin __BUCLE_INT_COMANDO_IEEE
765
766 _RET_ERROR_INTERPRETAR_COMANDO:
767     LDI T2, 0
768     RET
769
770 _RET_INTERPRETAR_COMANDO:
771     RET
772
773 ; =====================================================================
774 ; ======================== Tabla de comandos ==========================
775 ; =====================================================================
776
```

```
777 COMANDOS_NIVEL_0: .db   5, "*IDN?", 4 ,"*RST", 4, "*SAV", \
778                         4, "ABOR", 5, "ABORt", \
779                         5, "READ?", \
780                         4, "CONF", 5, "CONF?", \
781                         9, "CONFigure", 10, "CONFigure?", \
782                         3, "MEM", 6, "MEMory", \
783                         4, "CONT", 7, "CONTrol", \
784                         4, "STAT", 6, "STATus", \
785                         4, "SYST", 6, "SYSTem"
786
787 COMANDOS_NIVEL_1_CONFIGURE: .db 4, "WIND", 6, "WINDow", 5, "WIND?", 7, "WINDow?", \
788                                 4, "COUN", 5, "COUNt", 5, "COUN?", 6, "COUNt?", \
789                                 4, "TRIG", 7, "TRIGger", 5, "TRIG?", 8, "TRIGger?", \
790                                 4, "DATA", 5, "DATA?"
791
792 COMANDOS_NIVEL_1_MEMORY: .db 5, "DATA?"
793
794 COMANDOS_NIVEL_1_CONTROL: .db 4, "APOW", 6, "APOWer", \
795                               5, "APOW?", 7, "APOWer?"
796
797 COMANDOS_NIVEL_1_STATUS: .db  5, "COND?", 10, "CONDition?"
798
799 COMANDOS_NIVEL_1_SYSTEM: .db  4, "BEEP", 6, "BEEPer", \
800                               5, "BEEP?", 7, "BEEPer?", \
801                               4, "KLOC", 5, "KLOCk", \
802                               5, "KLOC?", 6, "KLOCk?"
803
804 COMANDOS_NIVEL_2_BEEPER: .db  4, "STAT", 5, "STATe", \
805                               5, "STAT?", 6, "STATe?"
806
807 MENSAJE_IDENTIDAD: .db "Contador Geiger - FIUBA - 2018 - Nuñez Frau, Goyret, Vidal ", '\n', 0
```

teclado.asm

```
1  ; Descripcion: funciones para manejo del teclado
2
3  ; =====================================================================
4  ; ========================= Registros auxiliares ======================
5  ; =====================================================================
6
7  .DEF aux_1 = R16
8  .DEF aux_2 = R17
9  .DEF key_low = R18
10 .DEF key_high = R19
11
12 ; =====================================================================
13 ; ========================= Constantes ================================
14 ; =====================================================================
15
16 .EQU KEY_DDR = DDRC
17 .EQU KEY_PORT = PORTC
18
19 ;ADEN | ADSC | ADATE | ADIF | ADIE | ADPS2 | ADPS1 | ADPS0
20 /*b7=1 habilita el ADC; b6=1 empieza la conversion; b5=1 auto trigger, b4=1 flag de ADC
       completado y actalizado;
21 b3=1 seta la interrupcion del ADC cuando se completó la conversión; b2-0: seteo de prescaler
       */
22
23 ;Rango de tensión: +/- 0,25V
24
25 .EQU KEY_1_MIN_LOW = 0x33    ;    V1_MIN = 1.5 V ; ADC = 307 --> 0x0133
26 .EQU KEY_1_MIN_HIGH = 0x01
27
28 .EQU KEY_1_MAX_LOW = 0x9A    ;    V1_MAX = 2 V ; ADC = 410 --> 0x019A
29 .EQU KEY_1_MAX_HIGH = 0x01
30 ;=====================================================================
```

```
.EQU KEY_2_MIN_LOW = 0xB8    ;    V2_MIN = 2.15 V ; ADC = 440 --> 0x01B8
.EQU KEY_2_MIN_HIGH = 0x01

.EQU KEY_2_MAX_LOW = 0x1F    ;    V2_MAX = 2.65 V ; ADC = 543 --> 0x021F
.EQU KEY_2_MAX_HIGH = 0x02
;=============================================================================
.EQU KEY_3_MIN_LOW = 0x3D    ;    V3_MIN = 2.8V ; ADC = 573 --> 0x023D
.EQU KEY_3_MIN_HIGH = 0x02

.EQU KEY_3_MAX_LOW = 0xA4    ;    V3_MAX = 3.3 V ; ADC = 676 --> 0x02A4
.EQU KEY_3_MAX_HIGH = 0x02
;=============================================================================
.EQU KEY_4_MIN_LOW = 0xC2    ;    V4_MIN = 3.45 V ; ADC = 706 --> 0x02C2
.EQU KEY_4_MIN_HIGH = 0x02

.EQU KEY_4_MAX_LOW = 0x29    ;    V4_MAX = 3.95 V ; ADC = 809 --> 0x0329
.EQU KEY_4_MAX_HIGH = 0x03
;=============================================================================
.EQU KEY_5_MIN_LOW = 0x47    ;    V5_MIN = 4.1 V ; ADC = 839 --> 0x0347
.EQU KEY_5_MIN_HIGH = 0x03

.EQU KEY_5_MAX_LOW = 0xAF    ;    V5_MAX = 4.6 V ; ADC = 943 --> 0x03AF
.EQU KEY_5_MAX_HIGH = 0x03
;=============================================================================
.EQU KEY_6_MIN_LOW = 0xCC    ;    V6_MIN = 4,75V ; ADC = 972,8 --> 0x03CC
.EQU KEY_6_MIN_HIGH = 0x03

.EQU KEY_6_MAX_LOW = 0x34    ;    V6_MAX = 5,25V ; ADC = 1075,2 --> 0x0434
.EQU KEY_6_MAX_HIGH = 0x04
;=============================================================================
.EQU KEY_1_REG = 0x01
.EQU KEY_2_REG = 0x02
.EQU KEY_3_REG = 0b00000100
.EQU KEY_4_REG = 0b00001000
.EQU KEY_5_REG = 0b00010000
.EQU KEY_6_REG = 0b00100000
.EQU KEY_ERROR_REG = 0b10000000


; ====================================================================
; ========================= Funciones ================================
; ====================================================================

; Descripcion: configurar teclado
; Recibe: -
; Devuelve: -
CONF_ADC:
        ; Configurar la AVCC como la referencia externa del ADC
        SET_BIT ADMUX, REFS0
        CLEAR_BIT ADMUX, ADLAR

        ; Activar la interrupcion del ADC
        SET_BIT ADCSRA, ADIE

        ; Configurar el prescaler en 128
        SET_BIT ADCSRA, ADPS0
        SET_BIT ADCSRA, ADPS1
        SET_BIT ADCSRA, ADPS2

        RET

; ====================================================================
; Descripcion: interrupcion del ADC
ADC_ISR:
        PUSH aux_1
        IN aux_1, SREG
```

```
97          PUSH aux_1
98
99          ; Activar evento de que se ha presionado una tecla
100         SBR EVENTO, (1<<TECLA_PRESIONADA)
101
102         POP aux_1
103         OUT SREG, aux_1
104         POP aux_1
105
106         RETI
107
108 ; ====================================================================
109 ; Descripcion: leer el resultado de una medicion del ADC y setea un bit
110 ; de acuerdo a la tecla que fue presionada
111 ; Recibe: -
112 ; Devuelve: -
113 ADC_LEER_TECLA:
114
115         LDS aux_1, ADCL
116         LDS aux_2, ADCH
117
118         TOGGLE_LED_ARDUINO
119
120         PUSH R16
121
122         ; Encender timer0 para esperar cierto tiempo hasta activar nuevamente el boton
123         LDS R16, MOTIVO_TIMER0
124         SBR R16, (1<<MOTIVO_TIMER0_TECLADO_POST_PRESION)
125         STS MOTIVO_TIMER0, R16
126         CALL ENCENDER_TIMER_0
127
128         POP R16
129
130         ; Limpiar el bit de evento asociado a una tecla presionada
131         CBR EVENTO, (1<<TECLA_PRESIONADA)
132
133 ; DESCOMENTAR ESTE CODIGO PARA PROCESAR LA TECLA
134     LOWER_OR_EQUAL_KEY_6_MAX:
135             LDI key_low, KEY_6_MAX_LOW
136             LDI key_high, KEY_6_MAX_HIGH
137             CP aux_1, key_low
138             CPC aux_2, key_high
139             BRLO HIGHER_OR_EQUAL_KEY_6_MIN
140             RJMP SET_KEY_ERROR_REG
141     HIGHER_OR_EQUAL_KEY_6_MIN:
142             LDI key_low, KEY_6_MIN_LOW
143             LDI key_high, KEY_6_MIN_HIGH
144             CP aux_1, key_low
145             CPC aux_2, key_high
146             BRSH SET_KEY_6_REG
147     LOWER_OR_EQUAL_KEY_5_MAX:
148             LDI key_low, KEY_5_MAX_LOW
149             LDI key_high, KEY_5_MAX_HIGH
150             CP aux_1, key_low
151             CPC aux_2, key_high
152             BRLO HIGHER_OR_EQUAL_KEY_5_MIN
153             RJMP SET_KEY_ERROR_REG
154     HIGHER_OR_EQUAL_KEY_5_MIN:
155             LDI key_low, KEY_5_MIN_LOW
156             LDI key_high, KEY_5_MIN_HIGH
157             CP aux_1, key_low
158             CPC aux_2, key_high
159             BRSH SET_KEY_5_REG
160     LOWER_OR_EQUAL_KEY_4_MAX:
161             LDI key_low, KEY_4_MAX_LOW
162             LDI key_high, KEY_4_MAX_HIGH
```

```
163             CP aux_1, key_low
164             CPC aux_2, key_high
165             BRLO HIGHER_OR_EQUAL_KEY_4_MIN
166             RJMP SET_KEY_ERROR_REG
167     HIGHER_OR_EQUAL_KEY_4_MIN:
168             LDI key_low, KEY_4_MIN_LOW
169             LDI key_high, KEY_4_MIN_HIGH
170             CP aux_1, key_low
171             CPC aux_2, key_high
172             BRSH SET_KEY_4_REG
173     LOWER_OR_EQUAL_KEY_3_MAX:
174             LDI key_low, KEY_3_MAX_LOW
175             LDI key_high, KEY_3_MAX_HIGH
176             CP aux_1, key_low
177             CPC aux_2, key_high
178             BRLO HIGHER_OR_EQUAL_KEY_3_MIN
179             RJMP SET_KEY_ERROR_REG
180     HIGHER_OR_EQUAL_KEY_3_MIN:
181             LDI key_low, KEY_3_MIN_LOW
182             LDI key_high, KEY_3_MIN_HIGH
183             CP aux_1, key_low
184             CPC aux_2, key_high
185             BRSH SET_KEY_3_REG
186     LOWER_OR_EQUAL_KEY_2_MAX:
187             LDI key_low, KEY_2_MAX_LOW
188             LDI key_high, KEY_2_MAX_HIGH
189             CP aux_1, key_low
190             CPC aux_2, key_high
191             BRLO HIGHER_OR_EQUAL_KEY_2_MIN
192             RJMP SET_KEY_ERROR_REG
193     HIGHER_OR_EQUAL_KEY_2_MIN:
194             LDI key_low, KEY_2_MIN_LOW
195             LDI key_high, KEY_2_MIN_HIGH
196             CP aux_1, key_low
197             CPC aux_2, key_high
198             BRSH SET_KEY_2_REG
199     LOWER_OR_EQUAL_KEY_1_MAX:
200             LDI key_low, KEY_1_MAX_LOW
201             LDI key_high, KEY_1_MAX_HIGH
202             CP aux_1, key_low
203             CPC aux_2, key_high
204             BRLO HIGHER_OR_EQUAL_KEY_1_MIN
205             RJMP SET_KEY_ERROR_REG
206     HIGHER_OR_EQUAL_KEY_1_MIN:
207             LDI key_low, KEY_1_MIN_LOW
208             LDI key_high, KEY_1_MIN_HIGH
209             CP aux_1, key_low
210             CPC aux_2, key_high
211             BRSH SET_KEY_1_REG
212             RJMP SET_KEY_ERROR_REG
213
214 SET_KEY_6_REG:
215         LDI aux_1, KEY_6_REG
216         LDI aux_2, 1<<ADSC
217         RET
218
219 SET_KEY_5_REG:
220
221         LDI aux_1, KEY_5_REG
222         LDI aux_2, 1<<ADSC
223         RET
224
225 SET_KEY_4_REG:
226
227         LDI aux_1, KEY_4_REG
228         LDI aux_2, 1<<ADSC
```

```
229         RET
230
231 SET_KEY_3_REG:
232
233         LDI aux_1, KEY_3_REG
234         LDI aux_2, 1<<ADSC
235         RET
236
237 SET_KEY_2_REG:
238         LDI aux_1, KEY_2_REG
239         LDI aux_2, 1<<ADSC
240         RET
241
242 SET_KEY_1_REG:
243         ;CBI PORTB, 1
244         LDI aux_1, KEY_1_REG
245         LDI aux_2, 1<<ADSC
246         RET
247
248 SET_KEY_ERROR_REG:
249         LDI aux_1, KEY_ERROR_REG
250         LDI aux_2, 1<<ADSC
251         RET
```

TIMER0.asm

```
1  ; timers.asm
2  ; Created: 24/10/2018 9:16
3  ; Author : Juan Pablo Goyret
4  ; Descripcion: biblioteca con funciones para manejo del timer 0
5
6  ; Utilidad del timer 0: identificar un timeout (o exceso de tiempo)
7  ; en la recepcion de una cadena de caracteres por UART.
8  ; Este genera una interrupcion si ha transcurrido un cierto tiempo
9  ; desde la llegada de un primer caracter sin que se haya recibido un
10 ; caracter de fin de trama
11
12 ; =====================================================================
13 ; ==================== Variables auxiliares ===========================
14 ; =====================================================================
15 .UNDEF T0
16 .UNDEF T1
17 ; R2 Y R3 deben ser reservados para el timer dado que son contadores
18 ; de la cantidad de overflows que se produjeron hasta un determinado
19 ; instante en el tiempo
20 .DEF T0 = R2
21 .DEF T1 = R3
22
23 ; =====================================================================
24 ; ==================== Registros en RAM ===============================
25 ; =====================================================================
26 .dseg
27 ; Descripcion: registro que almacena el motivo por el cual se ha iniciado
28 ; el timer0
29 MOTIVO_TIMER0: .BYTE 1
30 ; Cada bit posee asociado un motivo:
31 ; 0 = overflow uart
32 ; 1 = teclado
33
34 .cseg
35
36 ; =====================================================================
37 ; ==================== Constantes =====================================
38 ; =====================================================================
39 ; Motivos de activacion del timer 0
40 .equ MOTIVO_TIMER0_UART = 0
```

```
41 .equ MOTIVO_TIMER0_TECLADO = 1
42 .equ MOTIVO_TIMER0_TECLADO_POST_PRESION = 2
43
44 ; Clock interno, prescaler = 64
45 ; --> con clk=16MHz, aproximadamente 1ms hasta que se llega a MAX
46 .equ CLOCK_TIMER = 3
47
48 ; Numero de ocurrencias de overflow del timer0 que tiene que ocurrir
49 ; para que se produzca un timeout
50 .equ TIMEOUT_T0_UMBRAL = 130
51 ; Con el prescaler configurado en 64, transcurren aproximadamente 62ms
52 ; desde la recepcion del primer caracter hasta el timeout.
53 ; Dado que para un buffer de 60 caracteres con transmision serie
54 ; de 76800 baudios el tiempo aproximado desde la recepcion del primer caracter
55 ; hasta el número 60 (suponiendo que la transmsion entre la PC y
56 ; el microcontrolador no sufre interrupciones durante todo el proceso)
57 ; es de: 60*8/76800 = 6ms, entonces el tiempo
58 ; transcurrido hasta un timeout es aproximadamente 10 ese valor
59
60 ; ====================================================================
61 ; ==================== Interrupciones ================================
62 ; ====================================================================
63
64 ; Timer/Counter0 Overflow
65 ; Descripcion: se dispara cuando el timer0 ha superado 0xFF y se ha
66 ; limpiado
67 OVERFLOW_TIMER0:
68
69     PUSH R16
70     IN  R16, SREG
71     PUSH R16    ; salva en la pila el estado actual de los flags (C, N, V y Z)
72
73     LDI R16, 1
74     ADD T0, R16
75     LDI R16, 0
76     ADC T1, R16
77
78     LDI R16, HIGH(TIMEOUT_T0_UMBRAL)
79     CP T1, R16
80     BRNE _RETORNAR_OVERFLOW_TIMER0
81
82     LDI R16, LOW(TIMEOUT_T0_UMBRAL)
83     CP T0, R16
84     BRNE _RETORNAR_OVERFLOW_TIMER0
85
86     ; Analizar el motivo por el cual se llamo al timer0 en un principio
87     LDS R16, MOTIVO_TIMER0
88
89     SBRC R16, MOTIVO_TIMER0_UART
90     RJMP _TIMEOUT_OVERFLOW_TIMER0
91
92     SBRC R16, MOTIVO_TIMER0_TECLADO
93     RJMP _TECLADO_OVERFLOW_TIMER0
94
95     SBRC R16, MOTIVO_TIMER0_TECLADO_POST_PRESION
96     RJMP _TECLADO_OVERFLOW_TIMER0_POST_PRESION
97
98     RJMP _RETORNAR_OVERFLOW_TIMER0
99
100 _TIMEOUT_OVERFLOW_TIMER0:
101
102     ; Limpiar motivo del evento
103     CBR R16, (1<<MOTIVO_TIMER0_UART)
104     STS MOTIVO_TIMER0, R16
105
106     ; Indicar que se produjo un error de timeout en el registro de eventos
```

```
107     SBR EVENTO, (1<<TIMEOUT_UART)
108 
109     CALL APAGAR_TIMER_0
110     RJMP _RETORNAR_OVERFLOW_TIMER0
111 
112 _TECLADO_OVERFLOW_TIMER0:
113 
114     ; Limpiar motivo del evento
115     CBR R16, (1<<MOTIVO_TIMER0_TECLADO)
116     STS MOTIVO_TIMER0, R16
117 
118     ; Hacer una conversion
119     SET_BIT ADCSRA, ADSC
120 
121     CALL APAGAR_TIMER_0
122     RJMP _RETORNAR_OVERFLOW_TIMER0
123 
124 _TECLADO_OVERFLOW_TIMER0_POST_PRESION:
125 
126     ; Limpiar motivo del evento
127     CBR R16, (1<<MOTIVO_TIMER0_TECLADO_POST_PRESION)
128     STS MOTIVO_TIMER0, R16
129 
130     ; Desactivar el ADC para activar el comparador
131     CLEAR_BIT ADCSRA, ADEN
132 
133     CALL APAGAR_TIMER_0
134     RJMP _RETORNAR_OVERFLOW_TIMER0
135 
136 _RETORNAR_OVERFLOW_TIMER0:
137     POP R16
138     OUT SREG, R16    ; Reestablece los flags
139     POP R16
140     RETI
141 
142 ; ====================================================================
143 ; ==================== Funciones generales ===========================
144 ; ====================================================================
145 
146 ; Descripcion: encender timer0 con una configuracion determinada de clock
147 ; Entradas: ninguna
148 ; Devuelve: -
149 ENCENDER_TIMER_0:
150     ; Configurar velocidad de funcionamiento del timer
151     WRITE_REG TCCR0B,  CLOCK_TIMER
152     RET
153 
154 ; ================================================
155 ; Descripcion: apaga timer0
156 ; Entradas: ninguna
157 ; Devuelve: -
158 APAGAR_TIMER_0:
159     ; Limpiar los contadores de overflow
160     CLR T0
161     CLR T1
162     ; Resetear el contador del timer
163     WRITE_REG TCNT0, 0 ; TODO: ver si es necesario hacer esto siempre
164     WRITE_REG TCCR0B,  0
165     RET
166 
167 
168 ; ================================================
169 ; Descripcion: activar interrupcion por overflow
170 ; Entradas: ninguna
171 ; Devuelve: -
172 ACTIVAR_ISR_TIMER0_OV:
```

```
173     SET_BIT TIMSK0, TOIE0
174     RET
175
176 ; ================================================
177 ; Descripcion: desactivar interrupcion por overflow
178 ; Entradas: ninguna
179 ; Devuelve: -
180 DESACTIVAR_ISR_TIMER0_OV:
181     CLEAR_BIT TIMSK0, TOIE0
182     RET
183
184
185 ; ================================================================================
186 ; Descripcion: indica si el teclado se encuentra en uso verificando si el timer
187 ; se encuentra encendido por ese motivo
188 ; Entradas: ninguna
189 ; Devuelve: bit de carry en 1 si el teclado se encuentra en uso o 0 en caso
190 ; contrario
191 CHEQUEAR_TECLADO_EN_USO:
192     LDS R16, MOTIVO_TIMER0
193
194     CLC
195
196     SBRC R16, MOTIVO_TIMER0_TECLADO
197     SEC
198
199     SBRC R16, MOTIVO_TIMER0_TECLADO_POST_PRESION
200     SEC
201
202     RET
203
204 ; ================================================================================
205 ; Descripcion: indica si el timer 0 se encuentra en uso por la uart
206 ; Entradas: ninguna
207 ; Devuelve: bit de carry en 1 si el teclado se encuentra en uso o 0 en caso
208 ; contrario
209 CHEQUEAR_UART_EN_USO:
210     LDS R16, MOTIVO_TIMER0
211
212     CLC
213
214     SBRC R16, MOTIVO_TIMER0_UART
215     SEC
216
217     RET
```

TIMER_1.asm

```
1  ;Fede: En este archivo pongo la inicialización del TIMER1, que se usa para guardar los
       instantes de tiempo de los pulsos capturados
2
3  ;REGISTRO TCCR1B:
4
5  ;[ICNC1][ICES1][-][WGM13][WGM12][CS12][CS11][CS10]
6
7  ;ICNC1: Si está seteado se filtra la entrada del pin ICP1
8  ;ICES1: Si está seteado, se captura por flanco ascendente en ICP1
9
10 ;CS12|CS11|CS10: Definen el prescaler del TIMER2.
11 ;000 -> Timer apagado
12 ;001 -> Encendido, sin prescaler
13 ;010 -> clk/8
14 ;011 -> clk/64
15 ;100 -> clk/256
16 ;101 -> clk/1024
17 ;110 -> Se utiliza para configurar un clock externo, por el pin T1
```

```
;111 -> Idem 110
;
    ;*****************************************************************************************

;REGISTROS: ICR1H|ICR1L

;En ellos se guarda el dato capturado del instante de tiempo del timer 1. Para sacar el valor
     de este registro y guardarlo
;en otro lado, se debe leer primero el ICR1L. Cuando se lee este primer registro
    automáticamente se guarda el valor de
;ICR1H en un registro temporario TEMP, de donde se puede levantar el dato.

;
    ;*****************************************************************************************


;REGISTRO TIMSK1:

;[-][-][ICIE1][-][-][OCIE1B][OCIE1A][TOIE1]

;ICIE1: Si está seteado y las interrupciones globales están activadas, se activa la
    interrupción por detección
;de flanco en ICP1.

;TOIE1: Si se setea, se habilita la interrupción por overflow del TIMER1.

;
    ;*****************************************************************************************


;REGISTRO TIFR1:

;[-][-][ICF1][-][-][OCF1B][OCF1A][TOV1]

;ICF1: Cuando se detecta un flanco en ICP1, se setea automáticamente. Luego de ejectudada la
    interrupción, se vuelve a poner en 0
;automáticamente.

;TOV1: Cuando hay overflow en el TIEMER1, se setea. Si están habilitadas las interrupciones
    globales y la interrupción por
;overflow, se ejecuta la interrupción.

INIT_TIMER_1:
;Se inicializa el TIMER1, que se utiliza para capturar el instante de tiempo de los pulsos.
    Se setea el prescaler del mismo,
;y se habilitan las interrupciones por flanco en el pin ICP1 y detección por flanco
    ascendente.

;NO CONFIGURE EL PRESCALER PORQUE NO SE TODAVIA A QUE VALOR LO VAMOS A PONER

;Registros utilizados: R17
    PUSH R17
    LDS R17, TCCR1B                     ;SE CARGA EL DATO DEL TCCR1B EN R17
    ;ORI R17, (1<<CS10)                 ;SE CONFIGURA EL PRESCALER
    ANDI R17, ~((1<<ICES1)|(1<<CS11)|(1<<CS12)|(1<<CS10))
    ORI R17, (1<<ICNC1)|(1<<WGM12)
    ;ORI R17, (1<<ICES1)|(1<<ICNC1)|(1<<WGM12)                  ;SE SETEA LA DETECCIÓN DE
        FLANCO ASCENDENTE
    STS TCCR1B, R17                     ;SE GUARDA LA CONFIGURACIÓN EN LA POSICIÓN DEL
        REGISTRO

    LDS R17, TIMSK1                     ;SE CARGA EL DATO DE LA CONFIGURACIÓN DE
        INTERRUPCIONES DEL TIMER 1
    ORI R17, (1<<ICIE1)|(1<<OCIE1A)     ;SE HABILITA LA INTERRUPCIÓN POR DETECCIÓN DE FLANCO
        Y POR COMPARACION DEL TIMER 1
    STS TIMSK1, R17                     ;SE GUARDA LA CONFIGURACIÓN DE INTERRUPCIONES DEL
```

```
        TIMER 1
67      POP R17
68      RET
69
70  APAGAR_TIMER_1:
71  ; Se inhabiluta el TIMER 1, al finalizar la medición total.
72  ; Registros utilizados: R17
73      PUSH R17
74      LDS R17, TCCR1B
75      ANDI R17, ~((1<<CS11)|(1<<CS12)|(1<<CS10))
76      STS TCCR1B, R17
77      POP R17
78      RET
```

TIMER_2.asm

```
1   ;Fede: En este archivo pongo la inicialización del TIMER2, que voy a usar para los delays del
        LCD.
2   ;REGISTRO TCCR2B:
3
4   ;[FOC2A][FOC2B][-][-][WGM22][CS22][CS21][CS20]
5
6   ;CS22|CS21|CS20: Definen el prescaler del TIMER2.
7   ;000 -> Timer apagado
8   ;001 -> Encendido, sin prescaler
9   ;010 -> clk/8
10  ;011 -> clk/64
11  ;100 -> clk/256
12  ;101 -> clk/1024
13  ;110 -> Se utiliza para configurar un clock externo, por el pin T0
14  ;111 -> Idem 110
15
16
17  ;
        ********************************************************************************************
18
19  ;REGISTRO TIMSK2:
20
21  ;[-][-][-][-][-][OCIE2B][OCIE2A][TOIE2]
22
23  ;TOIE2: Si se setea, se habilita la interrupción por overflow del TIMER2.
24
25  ;
        ********************************************************************************************
26
27  ;REGISTRO TIFR2:
28
29  ;[-][-][-][-][-][OCF2B][OCF2A][TOV2]
30
31  ;TOV2: Cuando hay overflow en el TIEMER2, se setea. Si están habilitadas las interrupciones
        globales y la interrupción por
32  ;overflow, se ejecuta la interrupción.
33
34  INIT_TIMER_2:
35  ;Se inicializa el timer 2. Particularmente se utiliza para funciones de delay para la
        pantallita LCD.
36  ;Se utiliza un prescaler de clk/8, y dado que el clock es de 16MHz -> este timer cuenta hasta
        127 us.
37  ;REGISTROS UTILIZADOS: R17
38
39      PUSH R17                            ;Se pushea en el stack, el dato del registro R17,
            para no perderlo
40      LDS R17, TCCR2B                     ;Se carga el registro TCCR2B para modificarlo
41      ORI R17, (1<<CS21)                  ;CS21 = 1
```

```
42      ANDI R17, ~((1<<CS20)|(1<<CS22))     ;CS20 = CS22 = 0, con esta configuración el prescaler
            se encuentra en clk/8
43      STS  TCCR2B, R17                ;Se carga la configuración en el registro del TIMER2
44
45      LDS R17, TIMSK2                 ;Se carga el registro TIMSK2, para modificarlo
46      ORI R17, (1<<TOIE2)             ;Se setea el bit para habilitar las interrupciones
            por overflow del TIMER2
47      STS TIMSK2, R17                 ;Se carga la configuración en el registro TIMSK0
48      POP R17                         ;Se recupera el dato precargado
49      RET
50
51  ;
    ************************************************************************************************

52  PRENDER_TIMER_2:
53
54      PUSH R17                        ;Se pushea en el stack, el dato del registro R17,
            para no perderlo
55      LDS R17, TCCR2B                 ;Se carga el registro TCCR2B para modificarlo
56      ORI R17, (1<<CS21)              ;CS21 = 1
57      ANDI R17, ~((1<<CS20)|(1<<CS22))     ;CS20 = CS22 = 0, con esta configuración el prescaler
            se encuentra en clk/8
58      STS  TCCR2B, R17                ;Se carga la configuración en el registro del TIMER2
59      POP R17
60      RET
```

TP_labo.componentinfo.xml

```
1  ï»¿<?xml version="1.0" encoding="utf-8"?>
2  <Store xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="
       AtmelPackComponentManagement">
3      <ProjectComponents>
4          <ProjectComponent z:Id="i1" xmlns:z="http://schemas.microsoft.com/2003/10/
               Serialization/">
5              <CApiVersion></CApiVersion>
6              <CBundle></CBundle>
7              <CClass>Device</CClass>
8              <CGroup>Startup</CGroup>
9              <CSub></CSub>
10             <CVariant></CVariant>
11             <CVendor>Atmel</CVendor>
12             <CVersion>1.2.0</CVersion>
13             <DefaultRepoPath>C:/Program Files (x86)\Atmel\Studio\7.0\Packs</DefaultRepoPath>
14             <DependentComponents xmlns:d4p1="http://schemas.microsoft.com/2003/10/
                   Serialization/Arrays" />
15             <Description></Description>
16             <Files xmlns:d4p1="http://schemas.microsoft.com/2003/10/Serialization/Arrays">
17                 <d4p1:anyType i:type="FileInfo">
18                     <AbsolutePath>C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\
                           ATmega_DFP\1.2.209\avrasm\inc</AbsolutePath>
19                     <Attribute></Attribute>
20                     <Category>include</Category>
21                     <Condition>AVRASM</Condition>
22                     <FileContentHash i:nil="true" />
23                     <FileVersion></FileVersion>
24                     <Name>avrasm/inc</Name>
25                     <SelectString></SelectString>
26                     <SourcePath></SourcePath>
27                 </d4p1:anyType>
28                 <d4p1:anyType i:type="FileInfo">
29                     <AbsolutePath>C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\
                           ATmega_DFP\1.2.209\avrasm\inc\m328pdef.inc</AbsolutePath>
30                     <Attribute></Attribute>
31                     <Category>header</Category>
32                     <Condition>AVRASM</Condition>
33                     <FileContentHash>YpGdK/vCoyGIUkMyqjAbKQ==</FileContentHash>
```

```
                    <FileVersion></FileVersion>
                    <Name>avrasm/inc/m328pdef.inc</Name>
                    <SelectString></SelectString>
                    <SourcePath></SourcePath>
                </d4p1:anyType>
                <d4p1:anyType i:type="FileInfo">
                    <AbsolutePath>C:/Program Files (x86)\Atmel\Studio\7.0\Packs\atmel\
                        ATmega_DFP\1.2.209\avrasm\templates\main.asm</AbsolutePath>
                    <Attribute>template</Attribute>
                    <Category>source</Category>
                    <Condition>AVRASM</Condition>
                    <FileContentHash>jDLWnQAhtmDKGf3WzOcDfA==</FileContentHash>
                    <FileVersion></FileVersion>
                    <Name>avrasm/templates/main.asm</Name>
                    <SelectString>Main file (.asm)</SelectString>
                    <SourcePath></SourcePath>
                </d4p1:anyType>
            </Files>
            <PackName>ATmega_DFP</PackName>
            <PackPath>C:/Program Files (x86)/Atmel/Studio/7.0/Packs/atmel/ATmega_DFP/1.2.209/
                Atmel.ATmega_DFP.pdsc</PackPath>
            <PackVersion>1.2.209</PackVersion>
            <PresentInProject>true</PresentInProject>
            <ReferenceConditionId>ATmega328P</ReferenceConditionId>
            <RteComponents xmlns:d4p1="http://schemas.microsoft.com/2003/10/Serialization/
                Arrays">
                <d4p1:string></d4p1:string>
            </RteComponents>
            <Status>Resolved</Status>
            <VersionMode>Fixed</VersionMode>
            <IsComponentInAtProject>true</IsComponentInAtProject>
        </ProjectComponent>
    </ProjectComponents>
</Store>
```

uart.asm

```
; uart.asm
;
; Created: 15/10/2018 21:53:12
; Author : Juan Pablo Goyret
; Descripcion: biblioteca con funciones para a comunicacion por puerto serie

; Instrucciones:
; ==> Para una cadena armada dinamicamente
; 1- Llamar a IR_COMIENZO_BUFFER_TX, para configurar
; el puntero X en el comienzo del buffer de TX
; 2- Cargar un caracter en R18 (es parametro de CARGAR_BUFFER_TX_CON_CARACTER)
; 3- Llamar a CARGAR_BUFFER_TX_CON_CARACTER.
; 4- Repetir los pasos 2 y 3 para la cantidad de caracteres que se desee
; 5- Agregar el caracter de fin de trama \n por medio
; de los pasos 2 y 3 para cumplir con el estandar SCPI
; 6- Llamar a ENVIAR_DATOS_UART para iniciar el envio de datos
;
; Ejemplo
;   CALL IR_COMIENZO_BUFFER_TX
;   LDI R18, 'A'
;   CALL CARGAR_BUFFER_TX_CON_CARACTER
;   LDI R18, 'B'
;   CALL CARGAR_BUFFER_TX_CON_CARACTER
;   LDI R18, 'C'
;   CALL CARGAR_BUFFER_TX_CON_CARACTER
;   CALL ENVIAR_DATOS_UART
;
; ==> Para enviar una cadena estatica
; 1- Cargar en R18 y R19 la direccion de memoria de la cadena (son el LSB y el MSB
```

```
    respectivamente)
; 2- Llamar a CARGAR_BUFFER
; 3- Llamar a ACTIVAR_INT_TX_UDRE0 para activar la interrupcione de TX de la UART y comenzar
    con el envio de datos

; =====================================================================
; ========================= Constantes ================================
; =====================================================================
.equ BUFFER_SIZE = 60

; Baud rate: con 16MHz de system clock
; 12 = 76.8k baudios, 0.2% error
; 103 = 9.6k baudios, 0.2% error
.equ BAUD_RATE = 12

; Valores posibles del registro R20 para configurar
; el caracter de fin de trama de CARGAR_BUFFER_DESDE_RAM
.equ NO_FIN_DE_TRAMA = 0
.equ AGREGAR_COMA = 1
.equ AGREGAR_PUNTO_Y_COMA = 2
.equ AGREGAR_NEWLINE = 3

; =====================================================================
; ======================== Registros Reservados =======================
; =====================================================================
; Punteros a los buffer de transmision y recepcion
.def PTR_TX_L = R8
.def PTR_TX_H = R9
.def PTR_RX_L = R10
.def PTR_RX_H = R11
.def BYTES_TRANSMITIR = R12
.def BYTES_RECIBIDOS = R13

; =====================================================================
; ======================== Registros auxiliares =======================
; =====================================================================
.undef T0
.undef T1
.def T0 = R16
.def T1 = R17

; =====================================================================
; ========================== Interrupciones ===========================
; =====================================================================

; Tipo de interrupcion: Data Register Empty interrupt
; Descripcion: enviar un caracter del buffer de transmision
; cada vez que el buffer de datos UDR0 se encuentra vacio.
; Una vez que el buffer de transmision se hacia vaciado, se desactiva
; la interrupcion
; Entradas: BUFFER_TX, PTR_TX_L, PTR_TX_H
; Devuelve: -
USART_UDRE:

    PUSH T0
    PUSH XL
    PUSH XH
    IN T0, SREG
    PUSH T0

    ; Indicar que se esta enviando una cadena
    SBR EVENTO, (1<<ENVIANDO_DATOS_UART); TODO: MOVER ESTO A LA FUNCION QUE ACTIVA EL ENVIO
        DE DATOS

    ; Cargar puntero X con la direccion del buffer de transmision
    MOVW XH:XL, PTR_TX_H:PTR_TX_L
```

```
93
94      LD T0, X+
95      STS UDR0, T0
96
97      DEC BYTES_TRANSMITIR
98      BREQ _FIN_TRANSMISION_UART
99
100     MOVW PTR_TX_H:PTR_TX_L, XH:XL
101
102     POP T0
103     OUT SREG, T0
104     POP XH
105     POP XL
106     POP T0
107     RETI
108
109 _FIN_TRANSMISION_UART:
110
111     ; Deshabilitar las interrupciones de TX
112     CALL DESACTIVAR_INT_TX_UDRE0
113
114     ; Indicar que no se estan enviando datos por UART
115     CBR EVENTO, (1<<ENVIANDO_DATOS_UART)
116
117     ; Configurar puntero al comienzo del buffer de transmision
118     LDI T0, LOW(BUFFER_TX)
119     MOV PTR_TX_L, T0
120     LDI T0, HIGH(BUFFER_TX)
121     MOV PTR_TX_H, T0
122
123     POP T0
124     OUT SREG, T0
125     POP XH
126     POP XL
127     POP T0
128     RETI
129
130 ; ====================================================================
131 ; Tipo de interrupcion: Receive Complete interrupt
132 ; Descripcion: guarda un caracter en el buffer de RX cuando
133 ; se dispara la interrupcion que indica que el buffer UDR0
134 ; esta lleno.
135 ; Cuando se termina de recibir una cadena, se llama a la funcion FUNCION
136 ; para interpretarla
137 ; Se interpreta que se ha dejado de recibir una cadena cuando:
138 ; --> Esta termina con un caracter nulo
139 ; --> Su largo es igual o mayor al del buffer de lectura
140 ;
141 ; Entradas: ninguna
142 ; Devuelve: BUFFER_RX, BYTES_RECIBIDOS
143 USART_RX:
144
145     PUSH T0
146     PUSH XL
147     PUSH XH
148     IN T0, SREG
149     PUSH T0
150
151     ; Entrar en estado de recepcion de cadena
152     LDS T0, EVENTO2
153     SBR T0, (1<<BIT_CARACTER_RECIBIDO)
154     STS EVENTO2, T0
155     ;CBR ESTADO, (1<<EST_OSCIOSO_UART)
156     ;SBR ESTADO, (1<<EST_RECIBIENDO_CADENA)
157
158     MOVW XH:XL, PTR_RX_H:PTR_RX_L
```

```
159
160     ; Leer caracter que llego por UART
161     LDS T0, UDR0
162
163     ; Sumar caracter al buffer
164     ST X+, T0
165     INC BYTES_RECIBIDOS
166
167     ; Verificar si el caracter recibido es newline
168     CPI T0, '\n'
169     BREQ _FIN_CADENA
170
171     ; Verificar si la cantidad de bytes recibidos ha excedido
172     ; el tamano del buffer de RX
173     MOV T0, BYTES_RECIBIDOS
174     CPI T0, BUFFER_SIZE
175     BREQ _OVERFLOW_BUFFER
176
177     MOVW PTR_RX_H:PTR_RX_L, XH:XL
178
179     POP T0
180     OUT SREG, T0
181     POP XH
182     POP XL
183     POP T0
184     RETI
185
186 _FIN_CADENA:
187     ; Salir del estado de recepcion de cadena y entrar al
188     ; de procesamiento de cadena
189     LDS T0, EVENTO2
190
191     SBRC T0, SISTEMA_OCUPADO
192     RJMP __ERR_SISTEMA_OCUPADO
193
194     SBR T0, (1<<BIT_FIN_CADENA)
195     STS EVENTO2, T0
196     RJMP __SEGUIR_FIN_CADENA
197
198 __ERR_SISTEMA_OCUPADO:
199     CALL ENVIAR_ERROR_OCUPADO
200
201     ; Limpiar el bit que indica que el sistema se encuentra ocupado
202     CBR T0, (1<<SISTEMA_OCUPADO)
203     STS EVENTO2, T0
204
205 __SEGUIR_FIN_CADENA:
206     ; Configurar puntero al comienzo del buffer de recepcion
207     LDI T0, LOW(BUFFER_RX)
208     MOV PTR_RX_L, T0
209     LDI T0, HIGH(BUFFER_RX)
210     MOV PTR_RX_H, T0
211
212     POP T0
213     OUT SREG, T0
214     POP XH
215     POP XL
216     POP T0
217     RETI
218
219 _OVERFLOW_BUFFER:
220     ; Indicar en el registro de eventos que se produjo un error
221     ; de overflow de buffer
222     SBR EVENT0, (1<<OV_BUFFER_RX_UART)
223
224     ; Configurar puntero al comienzo del buffer de recepcion
```

```
225     LDI T0, LOW(BUFFER_RX)
226     MOV PTR_RX_L, T0
227     LDI T0, HIGH(BUFFER_RX)
228     MOV PTR_RX_H, T0
229 
230     POP T0
231     OUT SREG, T0
232     POP XH
233     POP XL
234     POP T0
235     RETI
236 
237 ; =====================================================================
238 ; ==================== Funciones generales ============================
239 ; =====================================================================
240 
241 ; Descripcion: encargada de inicializar UART
242 ; Entradas: ninguna
243 ; Devuelve: -
244 INICIALIZAR_UART:
245 
246     PUSH T0
247     PUSH T1
248     PUSH XL
249     PUSH XH
250     PUSH YL
251     PUSH YH
252 
253     ; Comunicacion asincronica
254     ; 8 bits de data frame
255     ; 1 solo bit de stop
256     ; Sin bit de paridad
257     LDI T0, (1 << UCSZ01 | 1 << UCSZ00)
258     STS UCSR0C, T0
259 
260     ; Velocidad de 9600 baudios teniendo en cuenta
261     ; un clock de 16MHz
262     LDI T0, 0
263     STS UBRR0H, T0
264     LDI T0, BAUD_RATE
265     STS UBRR0L, T0
266 
267 
268     ; Configurar puntero al comienzo del buffer de transmision
269     LDI T0, LOW(BUFFER_TX)
270     MOV PTR_TX_L, T0
271     LDI T0, HIGH(BUFFER_TX)
272     MOV PTR_TX_H, T0
273 
274     ; Configurar puntero al comienzo del buffer de recepcion
275     LDI T0, LOW(BUFFER_RX)
276     MOV PTR_RX_L, T0
277     LDI T0, HIGH(BUFFER_RX)
278     MOV PTR_RX_H, T0
279 
280     ; ====================================
281     ; Subrutina para limpiar los buffers
282 
283     CLR BYTES_RECIBIDOS
284 
285     CLR T0
286     LDI T1, BUFFER_SIZE
287 
288     ; Cargar puntero X con la direccion del buffer de transmision
289     MOVW XH:XL, PTR_TX_H:PTR_TX_L
290     MOVW YH:YL, PTR_RX_H:PTR_RX_L
```

```
291
292 _LOOP_LIMPIAR_BUFFER:
293     ST X+, T0
294     ST Y+, T0
295     DEC T1
296     BRNE _LOOP_LIMPIAR_BUFFER
297
298
299     POP YH
300     POP YL
301     POP XH
302     POP XL
303     POP T1
304     POP T0
305     RET
306
307
308 ; ============================
309 ; Descripcion: ubicar los punteros PTR_RX_L y PTR_RX_H
310 ; al comienzo del buffer de RX y limpiar el contador
311 ; de bits recibidos
312 ; Entradas: ninguna
313 ; Devuelve: -
314 LIMPIAR_BUFFER_RX:
315     PUSH T0
316
317     LDI T0, LOW(BUFFER_RX)
318     MOV PTR_RX_L, T0
319     LDI T0, HIGH(BUFFER_RX)
320     MOV PTR_RX_H, T0
321
322     CLR BYTES_RECIBIDOS
323
324     POP T0
325     RET
326
327 ; ========================================================================
328 ; ======= Funciones de activacion de recepcion o transmision ========
329 ; ========================================================================
330
331 ; Descripcion: habilitar la recepcion de datos por puerto serie
332 ; Entradas: ninguna
333 ; Devuelve: -
334 ACTIVAR_RX:
335     PUSH T0
336     LDS T0, UCSR0B
337     ORI T0, 1 << RXEN0
338     STS UCSR0B, T0
339     POP T0
340     RET
341
342 ; ============================================================
343 ; Descripcion: habilitar la transmision de datos por puerto serie
344 ; Entradas: ninguna
345 ; Devuelve: -
346 ACTIVAR_TX:
347     PUSH T0
348     LDS T0, UCSR0B
349     ORI T0, 1 << TXEN0
350     STS UCSR0B, T0
351     POP T0
352     RET
353
354 ; ============================================================
355 ; Descripcion: habilitar la transmision y recepcion de datos por puerto serie
356 ; Entradas: ninguna
```

```
357 ; Devuelve: -
358 ACTIVAR_TX_RX:
359     PUSH T0
360     LDS T0, UCSR0B
361     ORI T0, (1 << RXEN0 | 1 << TXEN0)
362     STS UCSR0B, T0
363     POP T0
364     RET
365 
366 ; ============================================================
367 ; Descripcion: deshabilitar la recepcion de datos por puerto serie
368 ; Entradas: ninguna
369 ; Devuelve: -
370 DESACTIVAR_RX:
371     PUSH T0
372     LDS T0, UCSR0B
373     ANDI T0, ~(1 << RXEN0)
374     STS UCSR0B, T0
375     POP T0
376     RET
377 
378 ; ============================================================
379 ; Descripcion: deshabilitar la recepcion de datos por puerto serie
380 ; Entradas: ninguna
381 ; Devuelve: -
382 DESACTIVAR_TX:
383     PUSH T0
384     LDS T0, UCSR0B
385     ANDI T0, ~(1 << TXEN0)
386     STS UCSR0B, T0
387     POP T0
388     RET
389 
390 ; ============================================================
391 ; Descripcion: deshabilitar la recepcion de datos por puerto serie
392 ; Entradas: ninguna
393 ; Devuelve: -
394 DESACTIVAR_TX_RX:
395     PUSH T0
396     LDS T0, UCSR0B
397     ANDI T0, ~(1 << RXEN0 | 1 << TXEN0)
398     STS UCSR0B, T0
399     POP T0
400     RET
401 
402 ; ============================================================
403 ; ========== Funciones de activacion de interrupciones =========
404 ; ============================================================
405 
406 ; Descripcion: activar la interrupcion generada cuando el registro de TX de UART se encuentra
        vacio
407 ; Entradas: ninguna
408 ; Devuelve: -
409 ACTIVAR_INT_TX_UDRE0:
410     PUSH T0
411     LDS T0, UCSR0B
412     ORI T0, 1 << UDRIE0
413     STS UCSR0B, T0
414     POP T0
415     RET
416 
417 ; ============================================================
418 ; Descripcion: activar la interrupcion generada cuando se ha terminado una transmision por
      UART
419 ; Entradas: ninguna
420 ; Devuelve: -
```

```
421 ACTIVAR_INT_TX_TXC0:
422     PUSH T0
423     LDS T0, UCSR0B
424     ORI T0, 1 << TXCIE0
425     STS UCSR0B, T0
426     POP T0
427     RET
428 
429 ; ============================================================
430 ; Descripcion: activar la interrupcion generada cuando un dato recibido por UART se encuentra
        disponible
431 ; para ser leido
432 ; Entradas: ninguna
433 ; Devuelve: -
434 ACTIVAR_INT_RX:
435     PUSH T0
436     LDS T0, UCSR0B
437     ORI T0, 1 << RXCIE0
438     STS UCSR0B, T0
439     POP T0
440     RET
441 
442 ; ============================================================
443 ; Descripcion: activar la interrupcion generada cuando el registro de TX de UART se encuentra
        vacio
444 ; Entradas: ninguna
445 ; Devuelve: -
446 DESACTIVAR_INT_TX_UDRE0:
447     PUSH T0
448     LDS T0, UCSR0B
449     ANDI T0, ~(1 << UDRIE0)
450     STS UCSR0B, T0
451     POP T0
452     RET
453 
454 ; ============================================================
455 ; Descripcion: activar la interrupcion generada cuando se ha terminado una transmision por
      UART
456 ; Entradas: ninguna
457 ; Devuelve: -
458 DESACTIVAR_INT_TX_TXC0:
459     PUSH T0
460     LDS T0, UCSR0B
461     ANDI T0, ~(1 << TXCIE0)
462     STS UCSR0B, T0
463     POP T0
464     RET
465 
466 ; ============================================================
467 ; Descripcion: activar la interrupcion generada cuando un dato recibido por UART se encuentra
        disponible
468 ; para ser leido
469 ; Entradas: ninguna
470 ; Devuelve: -
471 DESACTIVAR_INT_RX:
472     PUSH T0
473     LDS T0, UCSR0B
474     ANDI T0, ~(1 << RXCIE0)
475     STS UCSR0B, T0
476     POP T0
477     RET
478 
479 ; ================================================================
480 ; ========== Funciones de manejo de cadenas o caracteres =========
481 ; ================================================================
482 
```

```
; ============================================================
; Descripcion: recibe una cadena de caracteres almacenada en memoria
; flash y la carga en el buffer
; Entradas: direccion de una tabla almacenada en R18(LSB) y R19(MSB)
; Devuelve: -
CARGAR_BUFFER:
    PUSH T0
    PUSH T1
;   PUSH XL
;   PUSH XH
    PUSH ZL
    PUSH ZH

    CLR BYTES_TRANSMITIR

; Cargar puntero X con la direccion del buffer de transmision
    MOVW XH:XL, PTR_TX_H:PTR_TX_L

; Cargar puntero Z con la direccion de memoria provista por R18(LSB) y R19(MSB)
    MOV ZL, R18
    MOV ZH, R19
    LSL ZL
    ROL ZH

; Varible auxiliar
    LDI T1, BUFFER_SIZE

_LOOP_CARGAR_BUFFER:

    LPM T0, Z+
    CPI T0, 0
    BREQ _RETORNAR_CARGAR_BUFFER

    ST X+, T0
    INC BYTES_TRANSMITIR
    DEC T1
    BREQ _RETORNAR_CARGAR_BUFFER
    RJMP _LOOP_CARGAR_BUFFER

_RETORNAR_CARGAR_BUFFER:

    ; Activar interrupcion por buffer UDR0 vacio para
    ; enviar el contenido del buffer
;   CALL ACTIVAR_INT_TX_UDRE0


    POP ZH
    POP ZL
;   POP XH
;   POP XL
    POP T1
    POP T0
    RET

; ============================================================
; Descripcion: carga el buffer de TX con los datos almacenados en el buffer de RX
; para hacer un echo de lo recibido
; Entradas: BUFFER_RX
; Devuelve: BUFFER_TX

UART_ECHO:
    PUSH T0
    PUSH T1
    PUSH XL
    PUSH XH
    PUSH YL
```

```
549     PUSH YH
550
551     CLR BYTES_TRANSMITIR
552
553 ; Cargar puntero X con la direccion del buffer de transmision
554     MOVW XH:XL, PTR_TX_H:PTR_TX_L
555
556 ; Cargar puntero Y con la direccion del buffer de recepcion
557     MOVW YH:YL, PTR_RX_H:PTR_RX_L
558
559 ; Varible auxiliar
560     MOV T1, BYTES_RECIBIDOS
561
562 _LOOP_UART_ECHO:
563
564     LD T0, Y+
565     CPI T0, 0
566     BREQ _RETORNAR_CARGAR_BUFFER
567
568     ST X+, T0
569     INC BYTES_TRANSMITIR
570     DEC T1
571     BREQ _RETORNAR_CARGAR_BUFFER
572     RJMP _LOOP_UART_ECHO
573
574 _RETORNAR_UART_ECHO:
575
576     POP YH
577     POP YL
578     POP XH
579     POP XL
580     POP T1
581     POP T0
582     RET
583
584 ; ================================================
585 ; Descripcion: carga en el buffer de TX una cadena de texto almacenada en una
586 ; tabla en memoria RAM
587 ; Entradas:
588 ; -> puntero a cadena en RAM (registros R18 y R19)
589 ; -> registro R20 indicando que tipo de caracter se quiere enviar al final de la cadena
590 ; (a modo de caracter de fin de trama)
591 ; -- R20 = 0x00 : no colocar ningun caracter
592 ; -- R20 = 0x01 : agregar ','
593 ; -- R20 = 0x02 : agregar ';'
594
595 CARGAR_BUFFER_DESDE_RAM:
596     PUSH T0
597     PUSH T1
598     PUSH XL
599     PUSH XH
600     PUSH YL
601     PUSH YH
602
603     CLR BYTES_TRANSMITIR
604
605 ; Cargar puntero X con la direccion del buffer de transmision
606     MOVW XH:XL, PTR_TX_H:PTR_TX_L
607
608 ; Cargar puntero Z con la direccion de memoria provista por R18(LSB) y R19(MSB)
609     MOV YL, R18
610     MOV YH, R19
611
612 ; Varible auxiliar
613     LDI T1, BUFFER_SIZE
614
```

```
615 _LOOP_CARGAR_BUFFER_RAM:
616
617     LD T0, Y+
618     CPI T0, 0
619     BREQ _CARGAR_CARACTER_FIN_TRAMA
620
621     ST X+, T0
622     INC BYTES_TRANSMITIR
623     DEC T1
624     BREQ _CARGAR_CARACTER_FIN_TRAMA
625     RJMP _LOOP_CARGAR_BUFFER_RAM
626
627 _CARGAR_CARACTER_FIN_TRAMA:
628
629     CPI R20, NO_FIN_DE_TRAMA
630     BREQ _RET_CARGAR_BUFFER_DESDE_RAM
631
632     CPI R20, AGREGAR_COMA
633     BREQ _AGREGAR_COMA
634
635     CPI R20, AGREGAR_PUNTO_Y_COMA
636     BREQ _AGREGAR_PUNTO_Y_COMA
637
638     CPI R20, AGREGAR_NEWLINE
639     BREQ _AGREGAR_NEWLINE
640
641     RJMP _RET_CARGAR_BUFFER_DESDE_RAM
642
643 _AGREGAR_NEWLINE:
644     LDI T0, '\n'
645     ST X+, T0
646     INC BYTES_TRANSMITIR
647     RJMP _RET_CARGAR_BUFFER_DESDE_RAM
648
649 _AGREGAR_COMA:
650     LDI T0, ','
651     ST X+, T0
652     INC BYTES_TRANSMITIR
653     RJMP _RET_CARGAR_BUFFER_DESDE_RAM
654
655 _AGREGAR_PUNTO_Y_COMA:
656     LDI T0, 0X3B ; En ASCII 0x3B = ';'
657     ST X+, T0
658     INC BYTES_TRANSMITIR
659     RJMP _RET_CARGAR_BUFFER_DESDE_RAM
660
661 _RET_CARGAR_BUFFER_DESDE_RAM:
662
663     POP YH
664     POP YL
665     POP XH
666     POP XL
667     POP T1
668     POP T0
669     RET
670
671
672 ; ============================
673 ; Descripcion: coloca el puntero X al comienzo del buffer de transmision
674 ; Entradas: ninguna
675 ; Devuelve: puntero X con la direccion del comienzo del buffer de transmision
676 IR_COMIENZO_BUFFER_TX:
677     PUSH T0
678
679     LDI XL, LOW(BUFFER_TX)
680     LDI XH, HIGH(BUFFER_TX)
```

```
681
682     CLR BYTES_TRANSMITIR
683
684     POP T0
685     RET
686
687 ; ====================================================
688 ; Descripcion: cargar el buffer de tx con un caracter
689 ; Entrada: R18
690 ; Salida: -
691 CARGAR_BUFFER_TX_CON_CARACTER:
692     ; Cargar puntero X con la direccion del buffer de transmision
693     INC BYTES_TRANSMITIR
694     ST X+, R18
695
696     RET
697
698 ; Descripcion: comenzar con el envio de datos almacenados en el buffer de TX
699 ; Entradas: ninguna
700 ; Salidas: -
701 ENVIAR_DATOS_UART:
702     CALL ACTIVAR_INT_TX_UDRE0
703     RET
704
705 ; ====================================================
706 ; Descripcion: transformar un numero binario a ASCII y enviar por UART
707 ; Entrada: registros R17 y R16 con el valor de 16 bits a convertir
708 ; Salida: -
709 CONVERTIR_A_BINARIO_Y_ENVIAR_UART:
710
711     CALL IR_COMIENZO_BUFFER_TX
712
713     ; Convertir los bits de la cantidad de cuentas a ASCII
714     ;CALL DEC_TO_ASCII_16_BITS
715     CALL DEC_TO_ASCII_24_BITS
716
717     ; Reemplazar caracter \0 por \n\0
718     LDI XL, LOW(BUFFER_TX + CANTIDAD_DE_DIGITOS)
719     LDI XH, HIGH(BUFFER_TX + CANTIDAD_DE_DIGITOS)
720     LDI R18, '\n'
721     ST X+, R18
722     LDI R18, 0
723     ST X, R18
724
725     ; Setear los registros R19 y R18 con la direccion de memoria
726     ; del buffer donde DEC_TO_ASCII_16_BITS guarda el resultado
727     ; de la conversion
728     ;LDI R19, HIGH(NUMERO_ASCII)
729     ;LDI R18, LOW(NUMERO_ASCII)
730
731     ; Transferir los datos del registro donde DEC_TO_ASCII_16_BITS
732     ; guarda el resultado de la conversion al buffer de tx
733     ;CALL CARGAR_BUFFER_DESDE_RAM
734
735     LDI T0, CANTIDAD_DE_DIGITOS+1 ; Se incluye un +1 por el agregado del caracter de fin
            trama
736     MOV BYTES_TRANSMITIR, T0
737     CALL ENVIAR_DATOS_UART
738
739     RET
740
741 ; ====================================================================
742 ; ======================== Envio de mensajes de error ==============
743 ; ====================================================================
744 ; Descripcion: funciones que envian mensajes de error por UART en base
745 ; a los comandos recibidos
```

```
746
747 ; CODIGOS DE ERROR:
748 ; 100 - Error de comando
749 ; 363 - Overflow del buffer
750 ; 365 - Error de timeout
751
752 ENVIAR_ERROR_PARSER:
753     PUSH R18
754     PUSH R19
755
756     LDI R18, LOW(MENSAJE_ERROR_COMANDO)
757     LDI R19, HIGH(MENSAJE_ERROR_COMANDO)
758     CALL CARGAR_BUFFER
759
760     ; Activar interrupcion por buffer UDR0 vacio para
761     ; enviar el contenido del buffer
762     CALL ACTIVAR_INT_TX_UDRE0
763
764     POP R19
765     POP R18
766     RET
767
768 ENVIAR_ERROR_TIMEOUT:
769     PUSH R18
770     PUSH R19
771
772     LDI R18, LOW(MENSAJE_ERROR_RECEPCION_TIMEOUT)
773     LDI R19, HIGH(MENSAJE_ERROR_RECEPCION_TIMEOUT)
774     CALL CARGAR_BUFFER
775
776     ; Activar interrupcion por buffer UDR0 vacio para
777     ; enviar el contenido del buffer
778     CALL ACTIVAR_INT_TX_UDRE0
779
780     POP R19
781     POP R18
782     RET
783
784 ENVIAR_ERROR_OVERFLOW:
785     PUSH R18
786     PUSH R19
787
788     LDI R18, LOW(MENSAJE_ERROR_RECEPCION_OV)
789     LDI R19, HIGH(MENSAJE_ERROR_RECEPCION_OV)
790     CALL CARGAR_BUFFER
791
792     ; Activar interrupcion por buffer UDR0 vacio para
793     ; enviar el contenido del buffer
794     CALL ACTIVAR_INT_TX_UDRE0
795
796     POP R19
797     POP R18
798     RET
799
800 ENVIAR_ERROR_OCUPADO:
801     PUSH R18
802     PUSH R19
803
804     LDI R18, LOW(MENSAJE_ERROR_RECEPCION_OCUPADO)
805     LDI R19, HIGH(MENSAJE_ERROR_RECEPCION_OCUPADO)
806     CALL CARGAR_BUFFER
807
808     ; Activar interrupcion por buffer UDR0 vacio para
809     ; enviar el contenido del buffer
810     CALL ACTIVAR_INT_TX_UDRE0
811
```

```
812     POP R19
813     POP R18
814     RET
815
816 ; ====================================================================
817 ; ========================= Espacios de memoria reservados ==========
818 ; ====================================================================
819 .dseg
820 BUFFER_TX: .BYTE BUFFER_SIZE
821 BUFFER_RX: .BYTE BUFFER_SIZE
822
823 MENSAJE_RAM_PRUEBAS: .BYTE 20
824
825 .cseg
826 ; Mensaje a enviar al iniciarse el sistema
827 MENSAJE_TX: .DB '\n',"Interfaz de manejo por PC del contador Geiger",'\n',0
828
829 ; Mensaje indicando la disminucion del multiplicador
830 MENSAJE_DEC_MUL: .DB "END  ",0 ; Debe tener 6 caracteres de largo incluyendo el nulo
831
832 ; ========= CODIGOS ==========
833 ; Codigos de error
834 MENSAJE_ERROR_COMANDO: .db "-100", '\n', 0
835 MENSAJE_ERROR_RECEPCION_OV: .DB "-363",'\n',0
836 MENSAJE_ERROR_RECEPCION_TIMEOUT: .DB "-365",'\n',0
837 MENSAJE_ERROR_RECEPCION_OCUPADO: .DB "-284",'\n',0
838
839 ; Otros codigos
840 MENSAJE_FIN_MEDICION: .db "-800", '\n', 0
```

Uso_general.asm

```
1  ; Se colocan funciones de uso general del proyecto
2
3
4  .EQU CANTIDAD_DE_DIGITOS = 5
5  .dseg
6  NUMERO_ASCII: .byte 7               ; Se piden 6 espacios de RAM para guardar los 5 dígitos
       del número convertido, seguido de un potencial indicador
7                              ; indicador de fin de trama (\n) y un caracter nulo
8  ;NUMERO_ASCII: .byte 9                                  ; de string.
9
10
11
12
13 PROMEDIO_RAM: .byte 3       ;Se guarda el resultado del promedio                [nibble_bajo:
       nibble_medio:nibble_alto]
14 .cseg
15 ; ***************************************************************************************
16 .MACRO DEC_WORD                      ; MACRO PARA PODER HACER UNA RESTA DE UNA PALABRA DE 16
       BITS
17 ; WORD = [@1:@0]
18     DEC @0                           ;Decremento NIBBLE BAJO
19
20     CPI @0, 0xFF                     ;Si hizo overflow, entonces debo restar el nibble alto
21     BRNE FIN_DEC_WORD                ;Si no hizo overflow, entonces no debo restar nada más
22
23     DEC @1
24
25 FIN_DEC_WORD:
26 .ENDMACRO
27 ; ***************************************************************************************
28
29 DEC_TO_ASCII_16_BITS:
30 ; Se convierte un numero alojado en R17:R16 en sus caracteres ASCII y se guardan en la RAM
       como una string, uno a continuación del otro. Se realiza una
```

```
; comparación con una tabla alojada en flash para separar los digitos y convertir cada uno por separado.
; Registros utilizados:
;   ZL, ZH, XL, XH, R19, R4, R5
    PUSH ZL
    PUSH ZH
    PUSH R19
    PUSH R4
    PUSH R5
    PUSH R21

    LDI R21, 0x01

    LDI ZL, LOW(TABLA_DEC_TO_ASCII*2)          ; PUNTERO A LA PRIMER POSICION DE LA TABLA DE COMPARACION
    LDI ZH, HIGH(TABLA_DEC_TO_ASCII*2)

SIGUIENTE_DIGITO:
    LDI R19,'0'-1                              ; ESTE REGISTRO SE UTILIZA PARA CONVERTIR CADA DIGITO A ASCII

    LPM R4, Z+                                 ; SE LEVANTA EL NUMERO DE LA TABLA, QUE SE CORRESPONDE CON EL DIGITO A CONVERTIR DEL NUMERO ORIGINAL
    LPM R5, Z+

SEGUIR_DIGITO:
    INC R19                                    ; SE INCREMENTA EN UNO CADA VEZ QUE SE LE RESTA EN NUMERO DE COMPARACIÓN, AL NÚMERO ORIGIANL

    SUB R16, R4                                ; SE LE RESTA AL NUMERO ORIGINAL, EL NUMERO DE LA TABLA DE COMPARACIÓN
    SBC R17, R5
    BRSH SEGUIR_DIGITO                         ; SI EL NUMERO ORIGINAL SIGUE SIENDO MAYOR AL NUMERO DE COMPARACIÓN, SE SIGUE RESTANDO HASTA QUE SUCEDA
                                               ; LO CONTRARIO

    ADD R16, R4                                ; SI EL NUMERO ORIGINAL ES MENOR AL NUMERO DE COMPARACIÓN, SE LE VUELVE A SUMAR EL NÚMERO DE COMPARACIÓN
    ADC R17, R5

    ST X+, R19                                 ; SE GUARDA LA CANTIDAD DE VECES QUE SE RESTÓ EL NUMERO DE COMPARACIÓN, QUE CORRESPONDE CON EL DÍGITO
                                               ; MÁS SIGNIFICATIVO DEL NÚMERO ORIGINAL
    ;CPI    ZL,LOW(TABLA_DEC_TO_ASCII*2)+CANTIDAD_DE_DIGITOS    ; SI SE ALCANZÓ EL FIN DE TABLA DE COMPARACIÓN, SE TERMINA LA CONVERSIÓN
    CP  R4,R21
    BRNE SIGUIENTE_DIGITO                      ; SI NO SE ALCANZÓ EL FIN DE TABLA, SE CONTINÚA CON LA COMPARACIÓN

    LDI R19, 0x00
    ST X, R19

    POP R21
    POP R5
    POP R4
    POP R19
    POP ZH
    POP ZL

    RET


; ********************************************************************************
DEC_TO_ASCII_24_BITS:
; Se convierte un numero alojado en R29:R17:R16 en sus caracteres ASCII y se guardan en la RAM como una string, uno a continuación del otro. Se realiza una
```

```
85 ; comparación con una tabla alojada en flash para separar los digitos y convertir cada uno
       por separado.
86 ; Registros utilizados:
87 ;   ZL, ZH, XL, XH, R19, R4, R5
88     PUSH ZL
89     PUSH ZH
90     PUSH R19
91     PUSH R4
92     PUSH R5
93     PUSH R6
94     PUSH R21
95
96     LDI R21, 0x01
97
98     LDI ZL, LOW(TABLA_DEC_TO_ASCII_5_DIGITOS*2)          ; PUNTERO A LA PRIMER POSICION DE LA
           TABLA DE COMPARACION
99     LDI ZH, HIGH(TABLA_DEC_TO_ASCII_5_DIGITOS*2)
100
101 _SIGUIENTE_DIGITO_24_BITS:
102    LDI R19,'0'-1                                ; ESTE REGISTRO SE UTILIZA PARA CONVERTIR
           CADA DIGITO A ASCII
103
104    LPM R6, Z+                                   ; SE LEVANTA EL NUMERO DE LA TABLA, QUE SE
           CORRESPONDE CON EL DIGITO A CONVERTIR DEL NUMERO ORIGINAL
105    LPM R5, Z+
106    LPM R4, Z+
107
108 _SEGUIR_DIGITO_24_BITS:
109    INC R19                                      ; SE INCREMENTA EN UNO CADA VEZ QUE SE LE
           RESTA EN NUMERO DE COMPARACIÓN, AL NÚMERO ORIGIANL
110
111    SUB R16, R4                                  ; SE LE RESTA AL NUMERO ORIGINAL, EL NUMERO
           DE LA TABLA DE COMPARACIÓN
112    SBC R17, R5
113    SBC R29, R6
114    BRSH _SEGUIR_DIGITO_24_BITS                          ; SI EL NUMERO ORIGINAL SIGUE SIENDO
           MAYOR AL NUMERO DE COMPARACIÓN, SE SIGUE RESTANDO HASTA QUE SUCEDA
115                                                 ; LO CONTRARIO
116
117    ADD R16, R4                                  ; SI EL NUMERO ORIGINAL ES MENOR AL NUMERO DE
           COMPARACIÓN, SE LE VUELVE A SUMAR EL NÚMERO DE COMPARACIÓN
118    ADC R17, R5
119    ADC R29, R6
120
121    ST X+, R19                                   ; SE GUARDA LA CANTIDAD DE VECES QUE SE RESTÓ
           EL NUMERO DE COMPARACIÓN, QUE CORRESPONDE CON EL DÍGITO
122                                                 ; MÁS SIGNIFICATIVO DEL NÚMERO ORIGINAL
123    CP  R4, R21                                  ; SI SE ALCANZÓ EL FIN DE TABLA DE
           COMPARACIÓN, SE TERMINA LA CONVERSIÓN
124    BRNE _SIGUIENTE_DIGITO_24_BITS                       ; SI NO SE ALCANZÓ EL FIN DE TABLA,
           SE CONTINÚA CON LA COMPARACIÓN
125
126    LDI R19, 0x00
127
128    ST X, R19
129
130    POP R21
131    POP R6
132    POP R5
133    POP R4
134    POP R19
135    POP ZH
136    POP ZL
137
138    RET
139 ; SE DEBE AGREGAR ESTA TABLA AL FIN DEL MAIN, PARA REALIZAR LA CONVERSIÓN A ASCII
```

```
140 TABLA_DEC_TO_ASCII: .dw 10000,1000,100,10,1
141
142 ; ======================================================
143 ; Descripcion: transforma un numero ascii de hasta 4 digitos en binario y
144 ; lo guarda en un registro de dos bytes en RAM
145 ; Recibe:
146 ; -> Posicion de memoria del buffer de RX donde comienza N en el puntero X
147 ; -> Puntero Y con el registro donde guardar el resultado
148 ; Salidas: -
149 TRANSFORMAR_DE_ASCII_A_BINARIO_Y_GUARDAR:
150     PUSH_WORD T0, T1
151     PUSH_WORD T2, T3
152     PUSH_WORD T4, T5
153
154     CLR T0
155
156 _BUCLE_CONF_NUM_VENT_CONTAR_DIGITOS:
157     LD T1, X+
158     CPI T1, '\r';TODO: CAMBIAR POR \n
159     BREQ _DETERMINAR_BINARIO
160     INC T0
161     CPI T0, MAX_NUM_DIGITOS+1
162     BREQ _RET_TRANSF_ASCII_A_BIN
163     CPI T1, '0'
164     BRLO _RET_TRANSF_ASCII_A_BIN
165     CPI T1, '9'+1
166     BRSH _RET_TRANSF_ASCII_A_BIN
167     RJMP _BUCLE_CONF_NUM_VENT_CONTAR_DIGITOS
168
169 _DETERMINAR_BINARIO:
170
171     CPI T0, 0
172     BREQ _RET_TRANSF_ASCII_A_BIN
173
174     ; Regresar el puntero X a su posicion original
175     MOV T1, T0
176     INC T1
177     SUB XL, T1
178     IN T2, SREG
179     SBRC T2, SREG_C
180     SUBI XH, 1
181
182     ; Registros temporales donde almacenar el numero de ventanas
183     CLR T1; (LSB)
184     CLR T2; (MSB)
185
186 _DIGITO_4:
187     CPI T0, 4
188     BRNE _DIGITO_3
189     DEC T0
190
191     LDI T5, 0x03
192     LDI T4, 0xE8
193     CALL AUX_TRANSF_ASCII_A_BIN
194
195 _DIGITO_3:
196     CPI T0, 3
197     BRNE _DIGITO_2
198     DEC T0
199
200     LDI T5, 0
201     LDI T4, 0x64
202     CALL AUX_TRANSF_ASCII_A_BIN
203
204 _DIGITO_2:
205     CPI T0, 2
```

```
206     BRNE _DIGITO_1
207     DEC T0
208
209     LDI T5, 0
210     LDI T4, 0x0A
211     CALL AUX_TRANSF_ASCII_A_BIN
212
213 _DIGITO_1:
214     CPI T0, 1
215     BRNE _GUARDAR_NUMERO_VENTANAS
216
217     LDI T5, 0
218     LDI T4, 1
219     CALL AUX_TRANSF_ASCII_A_BIN
220
221 _GUARDAR_NUMERO_VENTANAS:
222     ST Y+, T2
223     ST Y, T1
224
225     POP_WORD T4, T5
226     POP_WORD T2, T3
227     POP_WORD T0, T1
228     RET
229
230 _RET_TRANSF_ASCII_A_BIN:
231     CALL ENVIAR_ERROR_PARSER
232
233     POP_WORD T4, T5
234     POP_WORD T2, T3
235     POP_WORD T0, T1
236     RET
237
238 ;TABLA_DEC_TO_ASCII_5_DIGITOS: .db  0x98, 0x96, 0x80, 0x0F, 0x42, 0x40, 0x01, 0x86, 0xA0, 0
        x00, 0x27, 0x10, 0x00, 0x03, 0xE8, 0x00, 0x00, 0x64, 0x00, 0x00, 0x0A, 0x00, 0x00, 0x01
239 TABLA_DEC_TO_ASCII_5_DIGITOS: .db 0x00, 0x27, 0x10, 0x00, 0x03, 0xE8, 0x00, 0x00, 0x64, 0x00,
         0x00, 0x0A, 0x00, 0x00, 0x01
240
241 ; ============================================================
242 ; Descripcion: funcion auxiliar de TRANSFORMAR_DE_ASCII_A_BINARIO_Y_GUARDAR.
243 ; Multiplica una constante por un numero decimal y lo suma a una variable
244 ; Entradas:
245 ; -> Puntero X con la posicion del caracter que representa al
246 ; numero N.
247 ; -> Registros T4 y T5 con la constante
248 ; -> Registros T2 y T1 con las variable sobre la que se sumara
249 ; Salidas:
250  ; -> Registros T2 y T1 con el nuevo valor de la variable
251 AUX_TRANSF_ASCII_A_BIN:
252     LD T3, X+
253     CPI T3, '0'
254     BREQ _RET_AUX_TRANSF_ASCII_A_BIN
255     SUBI T3, '0'
256 _BUCLE_AUX_TRANSF_ASCII_A_BIN:
257     SUMAR_REGISTROS_16_BITS T2, T1, T5, T4
258     DEC T3
259     BRNE _BUCLE_AUX_TRANSF_ASCII_A_BIN
260
261 _RET_AUX_TRANSF_ASCII_A_BIN:
262     RET
263
264
265 ; ========================================================
266 ; Descripcion: transforma un numero ascii de hasta 5 digitos en binario y
267 ; lo guarda en un registro de dos bytes en RAM
268 ; Recibe:
269 ; -> Posicion de memoria del buffer de RX donde comienza N en el puntero X
```

```
; -> Puntero Y con el registro donde guardar el resultado
; -> R16 con la cantidad de digitos a convertir
; Salidas: -
TRANSFORMAR_DE_ASCII_A_BINARIO_Y_GUARDAR_5_DIGITOS:
    PUSH T1
    PUSH_WORD T2, T3
    PUSH_WORD T4, T5
    PUSH_WORD R4, R5

    ; == CODIGO DE PRUEBA ===
/*  CPI R16, 5
    BREQ ENCENDER
    RJMP NO_ENCENDER

    ENCENDER:
    ENCENDER_LED_ARDUINO

    NO_ENCENDER:*/
    ; === FIN CODIGO DE PRUEBA ===

    ; Registros temporales donde almacenar el numero de ventanas
    CLR R5; (MSB)
    CLR T2;
    CLR T1; (LSB)

_DIGITO_5_5:
    CPI R16, 5
    BRNE _DIGITO_4_5
    DEC R16

    LDI T5, 0x27
    LDI T4, 0x10
    CALL AUX_TRANSF_ASCII_A_BIN_5_DIGITOS

_DIGITO_4_5:
    CPI R16, 4
    BRNE _DIGITO_3_5
    DEC R16

    LDI T5, 0x03
    LDI T4, 0xE8
    CALL AUX_TRANSF_ASCII_A_BIN_5_DIGITOS

_DIGITO_3_5:
    CPI R16, 3
    BRNE _DIGITO_2_5
    DEC R16

    LDI T5, 0
    LDI T4, 0x64
    CALL AUX_TRANSF_ASCII_A_BIN_5_DIGITOS

_DIGITO_2_5:
    CPI R16, 2
    BRNE _DIGITO_1_5
    DEC R16

    LDI T5, 0
    LDI T4, 0x0A
    CALL AUX_TRANSF_ASCII_A_BIN_5_DIGITOS

_DIGITO_1_5:
    CPI R16, 1
    BRNE _GUARDAR_NUMERO_VENTANAS_5_DIGITOS

    CLR R4
```

```
336     LDI T5, 0
337     LDI T4, 1
338     CALL AUX_TRANSF_ASCII_A_BIN_5_DIGITOS
339
340 _GUARDAR_NUMERO_VENTANAS_5_DIGITOS:
341     ST Y+, R5
342     ST Y+, T2
343     ST Y, T1
344
345     POP_WORD R4, R5
346     POP_WORD T4, T5
347     POP_WORD T2, T3
348     POP T1
349     RET
350
351 ; ============================================================
352 ; Descripcion: funcion auxiliar de TRANSFORMAR_DE_ASCII_A_BINARIO_Y_GUARDAR.
353 ; Multiplica una constante por un numero decimal y lo suma a una variable
354 ; Entradas:
355 ; -> Puntero X con la posicion del caracter que representa al
356 ; numero N.
357 ; -> Registros T4 y T5 con la constante
358 ; -> Registros T2 y T1 con las variable sobre la que se sumara
359 ; Salidas:
360  ; -> Registros T2 y T1 con el nuevo valor de la variable
361 AUX_TRANSF_ASCII_A_BIN_5_DIGITOS:
362     LD T3, X+
363     CPI T3, '0'
364     BREQ _RET_AUX_TRANSF_ASCII_A_BIN_5_DIGITOS
365     SUBI T3, '0'
366     CLR R4
367 _BUCLE_AUX_TRANSF_ASCII_A_BIN_5_DIGITOS:
368     SUMAR_REGISTROS_24_BITS R5, T2, T1, R4, T5, T4
369     DEC T3
370     BRNE _BUCLE_AUX_TRANSF_ASCII_A_BIN_5_DIGITOS
371
372 _RET_AUX_TRANSF_ASCII_A_BIN_5_DIGITOS:
373     RET
374
375
376
377
378
379 ; ============================================================
380 PROMEDIO:
381
382 ; Calcula una división entre un número de 3 bytes, apuntado por Z y un númemero apuntado por
      X, de 3 bytes tambien. Devuelve el resultado en
383 ; una direccion llamada PROMEDIO_RAM
384     PUSH R5
385     PUSH R6
386     PUSH R7
387     PUSH R23
388     PUSH R14
389     PUSH R16
390     PUSH R17
391     PUSH R18
392     PUSH R19
393     PUSH R20
394     PUSH R21
395
396
397     LDI R16, 0x00
398     MOV R5, R16                             ; Se inicializa el resultado en 0
399     MOV R6, R16
400     MOV R7, R16
```

```
401     ; Se carga el dividendo, al que apunta Z: R16:R17:R18
402     LD R18, Z+                              ; Se carga nibble bajo
403     LD R17, Z+                              ; Se carga nibble medio
404     LD R16, Z+                              ; Se carga nibble alto
405
406     ; Se carga el divisor, al que apunta X: R21:R20:R19
407
408     LD R19, X+                              ; Se carga nibble alto
409     LD R20, X+                              ; Se carga nibble medio
410     LD R21, X+                              ; Se carga nibble bajo
411
412
413     DIVIDIR_24_BITS:
414     LDI R23, 0xFF
415
416
417     SEGUIR_RESTANDO:
418         INC R7
419         BRNE SIGO
420         INC R6
421         BRNE SIGO
422         INC R5
423     SIGO:
424         SUB R18, R21
425         SBC R17, R20
426         SBC R16, R19
427
428         BRCC SEGUIR_RESTANDO                    ; Si no hay overflow, sigo restando
429
430         DEC R7
431         CP R7, R23
432         BRNE NO_DECREMENTAR_SUPERIOR
433         DEC R6
434         CP R6, R23
435         BRNE NO_DECREMENTAR_SUPERIOR
436         DEC R5
437     NO_DECREMENTAR_SUPERIOR:
438         LDI ZL, LOW(PROMEDIO_RAM)
439         LDI ZH, HIGH(PROMEDIO_RAM)
440
441         ST Z+, R7
442         ST Z+, R6
443         ST Z+, R5
444
445
446
447 /*  DIVIDIR:
448      LDI R23, 0xFF
449
450      CPI R16, 0x00
451      BREQ SEGUIR_RESTA_16_BITS
452
453     SEGUIR_RESTANDO:
454         INC R6
455         BRNE SIGO
456         INC R7
457     SIGO:
458         SUB R18, R20
459         SBC R17, R19
460         SBCI R16, 0x00
461         CPI R16, 0x00
462         BRNE SEGUIR_RESTANDO
463
464     SEGUIR_RESTA_16_BITS:
465         INC R6
466         BRNE RESTA_16_BITS
```

```
467         INC R7
468
469     RESTA_16_BITS:
470         SUB R18, R20
471         SBC R17, R19
472         BRSH SEGUIR_RESTA_16_BITS
473         DEC R6
474         CP R6, R23
475         BRNE NO_DECREMENTAR_SUPERIOR
476         DEC R7
477     NO_DECREMENTAR_SUPERIOR:
478
479         LDI ZL, LOW(PROMEDIO_RAM)
480         LDI ZH, HIGH(PROMEDIO_RAM)
481
482         ST Z+, R6
483         ST Z+, R7*/
484
485
486         POP R21
487         POP R20
488         POP R19
489         POP R18
490         POP R17
491         POP R16
492         POP R14
493         POP R23
494         POP R7
495         POP R6
496         POP R5
497
498         RET
499 ; ============================================================
```