F i u b a
Facultad de Ingeniería
Universidad de Buenos Aires

Laboratorio de Microcomputadoras
(66.09/86.07)

Informe Final del Proyecto

Proyecto:	**Corrector de impedancia de electrodos**	
Autores:	Cassani, María Victoria	95.145
	Ferrari Bihurriet, Francisco	92.275
	Gomez, Kevin Leonel	93.906
Turno de TP:	1 (Martes 19 a 22 hs)	
Año y Cuatrimestre:	2015	Segundo
Docente Guía:	Stola, Gerardo	

Observaciones Generales

Calificación Final	
Firma del Docente	
Fecha	

Índice

1. Objetivo del Proyecto

El presente proyecto consiste en el diseño y desarrollo de un sistema de calibración de impedancia y ajuste por electrodeposición (galvanoplastía). Este instrumento se utilizará en el futuro en aplicaciones biomédicas, más precisamente en la fabricación de electrodos de registro extracelular.

2. Descripción del Proyecto

El equipo consiste en un circuito con un microcontrolador que genera señales senoidales y continuas para medir y corregir las impedancias, respectivamente. Incluye una cuba conductora, que contiene una solución acuosa donde se colocan los electrodos. Éstos son similares a un filamento de cobre recubierto con esmalte aislante en su totalidad, excepto en los extremos, los cuales uno va sumergido en la solución y el otro conectado al equipo por medio de un conector especial.

El sistema funciona con 1, 4, 8, 12 ó 16 electrodos, corrigiendo de a uno por vez. El usuario coloca los electrodos procurando que no se produzcan cortocircuitos entre los mismos (se debería detectar esto antes de comenzar). Hay tres opciones: medir, calibrar y corregir. La primera permite ver el/los valor/es de la/s impedancia/s colocada/s en los conectores. La segunda, calibrar el equipo, tal como su nombre lo indica. La opción de corregir permite que el usuario seleccione la impedancia final deseada, el tiempo máximo de espera para la corrección y dé la orden de inicio. Luego de algunos chequeos básicos se da comienzo al proceso. Al terminar con todos los electrodos, se muestra la impedancia final de cada uno.

En líneas generales, el sistema consta de módulos de amplificación y filtros de PWM que permiten generar una señal senoidal con el microcontrolador, además de controlar la señal de offset. Asimismo, hay una fuente de corriente controlada por tensión que permite inyectar corriente continua al sistema para poder disminuir las impedancias necesarias por galvanoplastía.

La señal de salida del PWM es una corriente senoidal de 1 kHz, con uno de los siguientes valores de amplitud: $\{30\,\text{nA}, 80\,\text{nA}, 120\,\text{nA}, 200\,\text{nA}\}$. Éstos se seleccionan con un multiplexor dependiendo del rango de impedancias: $\{0-2\,\text{M}\Omega, 0-8\,\text{M}\Omega, 0-20\,\text{M}\Omega, 0-60\,\text{M}\Omega\}$. Considerando la ley de Ohm y midiendo el valor pico de tensión, se calcula la impedancia del electrodo en estudio como $Z@1\,\text{kHz} = \frac{\hat{v}}{\hat{i}}$. Este valor se puede disminuir porque la solución acuosa contiene oro o plata, de modo de que se produce un recubrimiento por galvanoplastía sobre el electrodo y así disminuye la impedancia. Si este proceso es necesario o no, dependerá del resultado previo y de la comparación entre el valor obtenido en la medición y el de referencia establecido previamente por el usuario. Se emplea un multiplexor analógico para ir seleccionando cada electrodo y otro en paralelo con éste, que será eventualmente utilizado para verificar que no haya cortocircuitos entre electrodos.

Cada 1 min se mide la impedancia y se corrige de ser necesario. Este proceso se repite tantas veces como se requiera hasta llegar al valor de impedancia deseada determinado previamente en la configuración. Los valores aceptables para empezar el proceso están entre 1 MΩ y 8 MΩ.

Además, hay que pautar un tiempo máximo de sensado y corrección, para que el sistema deje de actuar en caso de haber pasado ese tiempo sin haber llegado al valor buscado. Éste puede valer 10, 20, 30, 40, 50 ó 60 minutos.

El dispositivo tiene una pantalla LCD donde se muestran las opciones de configuración para seleccionar a través de un menú y también los resultados del proceso, además de los mensajes de error o alerta (en caso de ocurrir algún imprevisto o no llegar al valor deseado en el tiempo pautado, por ejemplo). Las opciones se seleccionan desde un teclado de 4 botones (`ok`, `cancelar`, `mover izquierda`, `mover derecha`).

Además, como periféricos principales están los circuitos mencionados antes, a través de los cuales se interactúa directamente con los electrodos. Asimismo, otros elementos importantes con los que interactúa el microcontrolador son los multiplexores, que tienen 2 ó 4 entradas de control, según sean de 4 canales o 16, y la cuba electrolítica donde se produce la reacción química para disminuir la impedancia, que está compuesta por un material conductor, el cual también debe conectarse.

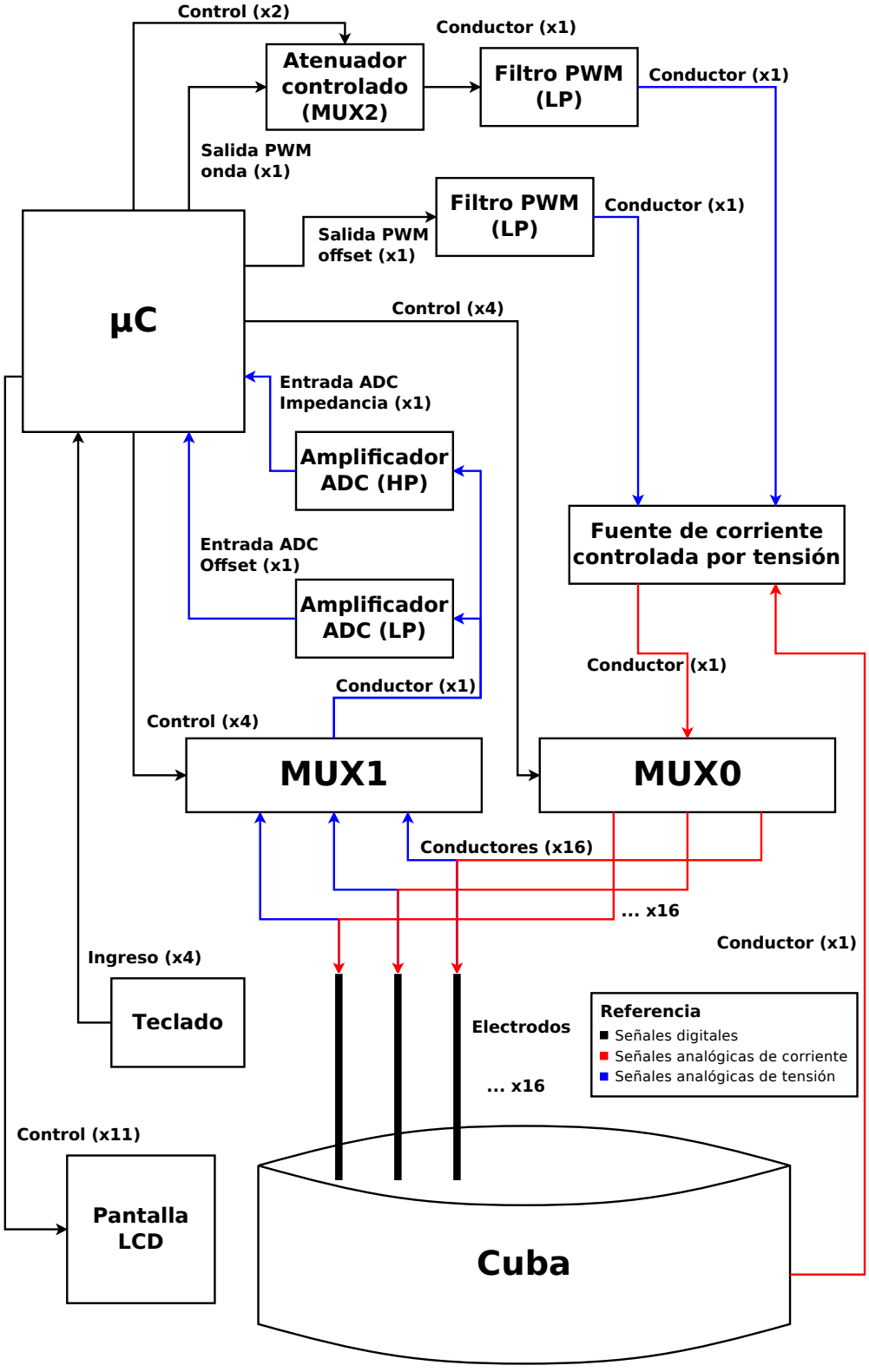
Las especificaciones del sistema son:

Tensión de alimentación	5 V
Rango de impedancias de los electrodos medidos	0 MΩ - 60 MΩ (@ 1 kHz)
Impedancia buscada	$\simeq$ 1 MΩ
Cantidad de electrodos a conectar	1, 4, 8, 12 ó 16
Tiempo máximo de corrección por electrodo	90 min (*)
Modo medición	señal senoidal, 1 kHz
Modo corrección	señal continua, 30/80/120/200 nA
Error de medición / corrección	$< 20\,\%$ / $< 20\,\%$

(*) una vez cumplido este tiempo se abandonan los esfuerzos.

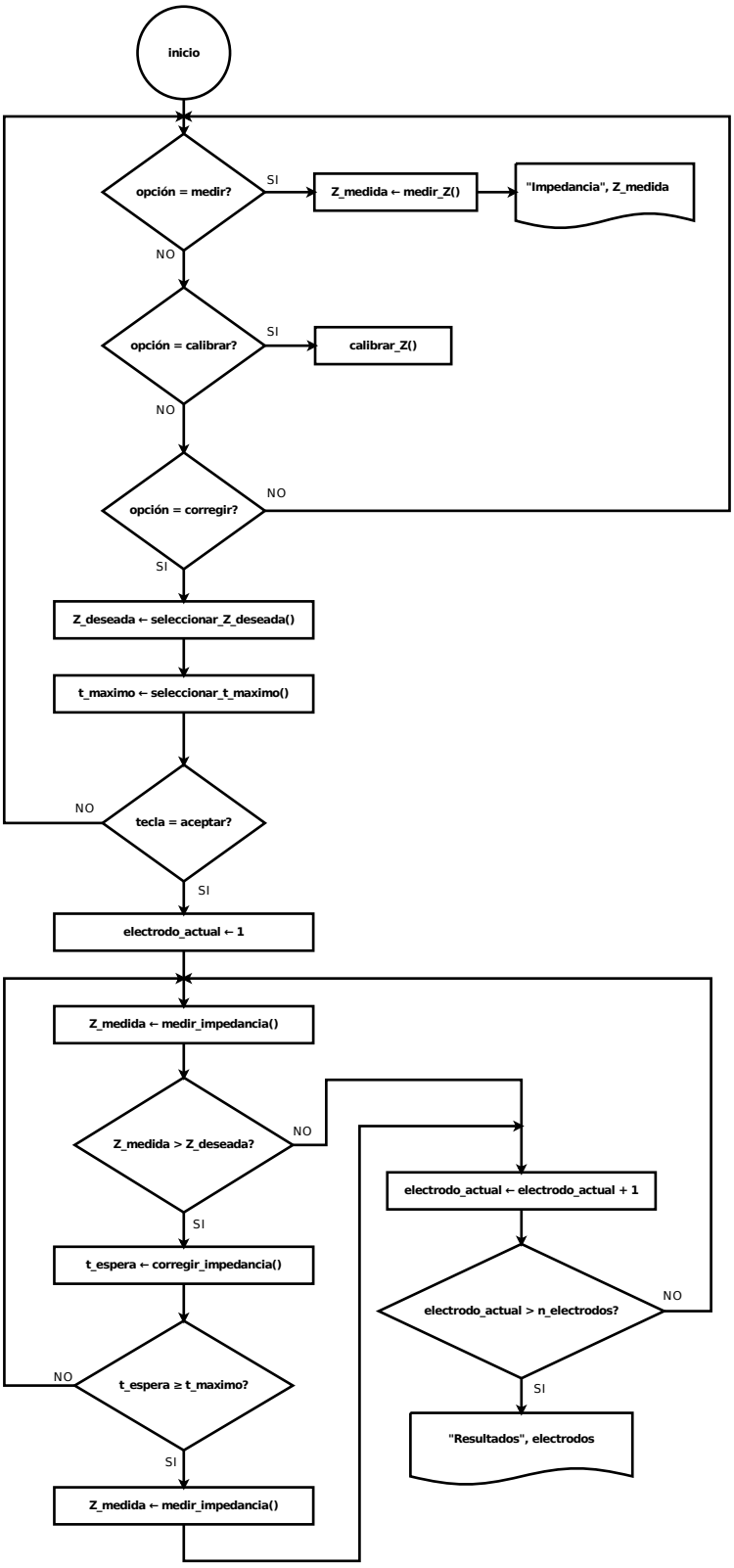
3. Diagrama en Bloques (hardware)

El diagrama de bloques aquí abajo muestra la interconexión de los dispositivos más relevantes del sistema.



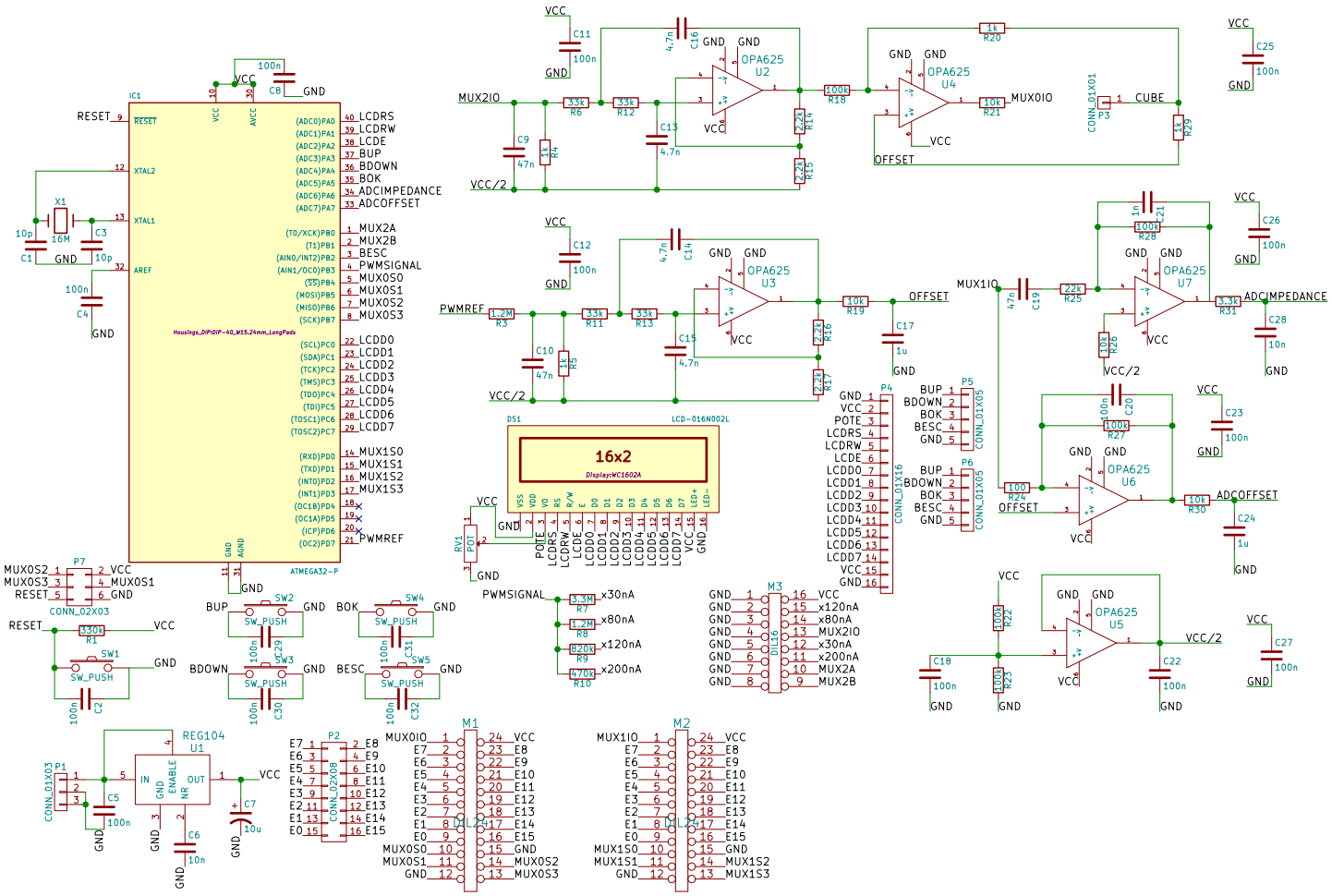
4. Diagrama de Flujo (firmware)

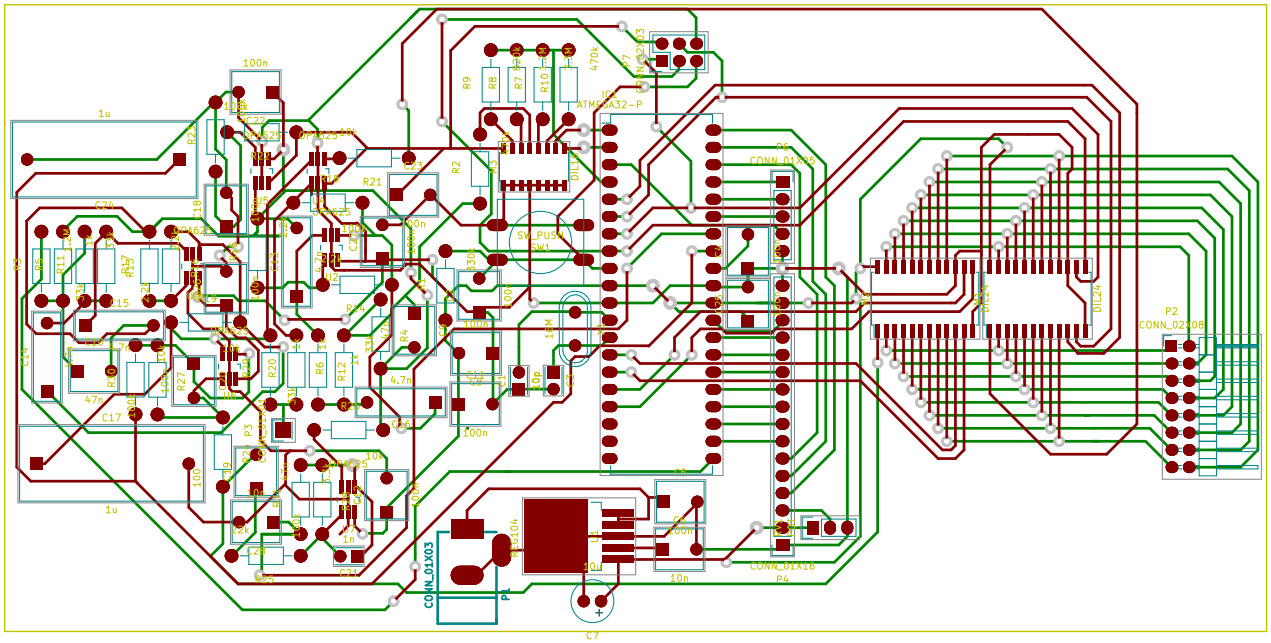
El diagrama de flujo final del proyecto es el siguiente:

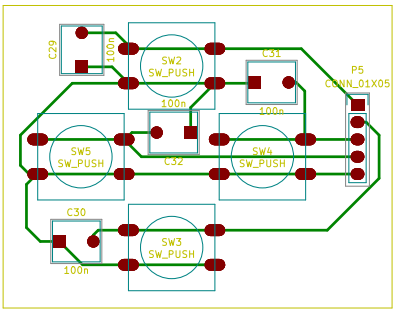


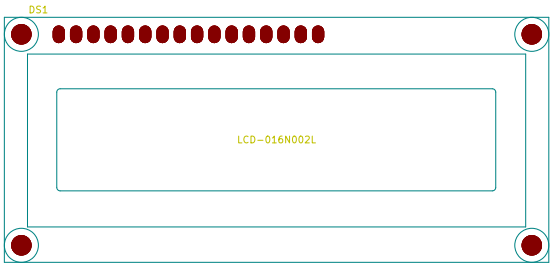
5. Circuito Esquemático

A continuación se muestra el circuito esquemático correspondiente al prototipo, así como también el diseño del PCB. Ambos fueron realizados con el programa KiCad.









6. Listado de Componentes y Costos

Los componentes que se emplearon para la construcción del dispositivo se muestran en la tabla siguiente:

Componente	**Costo**
Atmega32	$ 164.5
OPA625 x6 (*)	$ 166.2
REG104 (*)	$ 70.0
CD74HC4067 x2 (*)	$ 16.8
SN74LV4052 (*)	$ 5.0
Resistencias x30	$ 9.6
Potenciómetro	$ 6.7
Capacitores x31	$ 43.1
Display LCD	$ 120.0
Pulsadores x5	$ 18.8
Cristal 16 Mhz	$ 6.0
Conectores	$ 8.6
Placa Epoxy	$ 120.0
Zócalo	$ 3.0
Jack power	$ 3.2
TOTAL	$ 761.5

Los componentes marcados con un asterisco(*) son de Texas Instruments y en el marco del IIBM y de la Facultad se obtuvieron como muestras gratis. Por ende, para este trabajo en particular el costo de los componentes utilizados fue de $503,5.

En cuanto a las horas-hombre involucradas, se puede estimar un número de 300 horas/hombre.

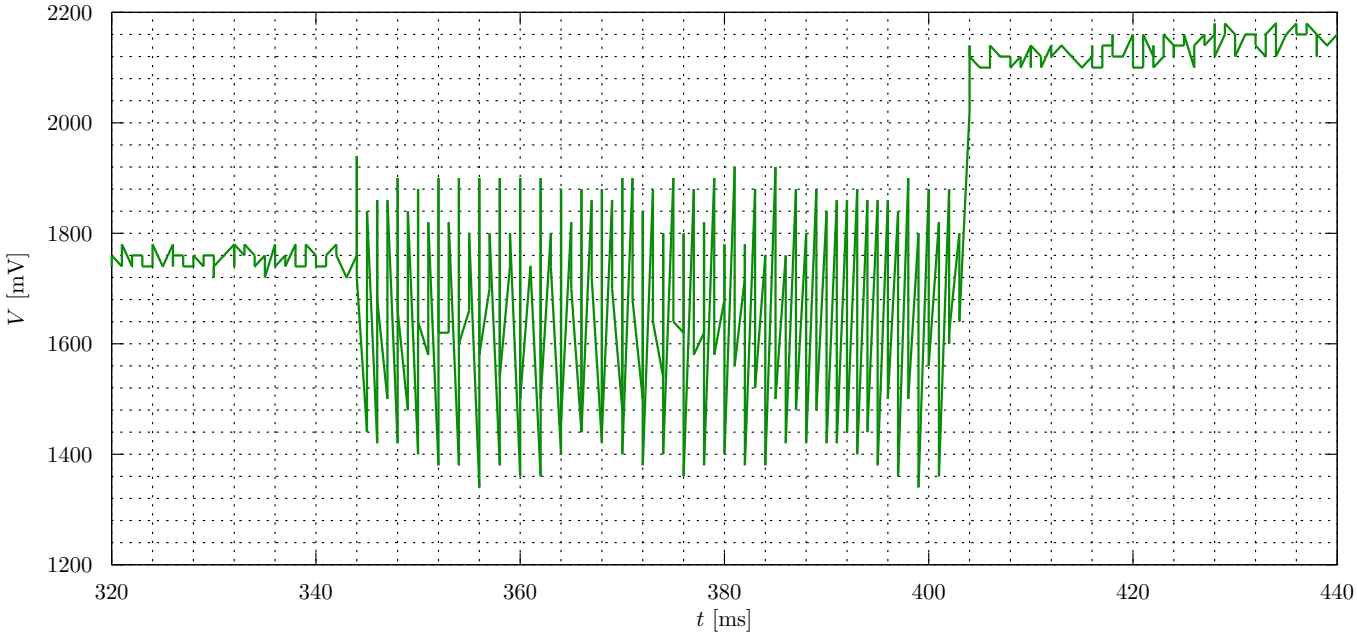
7. Resultados

Debido a las complicaciones en el diseño del hardware y los reiterados cambios que fueron hechos con el desarrollo del proyecto, se modificó el circuito de manera de poder medir impedancias del orden de los $k\Omega$, en vez de $M\Omega$, ya que las señales eran muy pequeñas y no se diferenciaban de las señales de ruido. Los nuevos rangos de medición son: $\{0 - 143\,\mathrm{k\Omega}, 143\,\mathrm{k\Omega} - 243\,\mathrm{k\Omega}, 243\,\mathrm{k\Omega} - 335\,\mathrm{M\Omega}, 335\,\mathrm{k\Omega} - 900\,\mathrm{M\Omega}\}$.

Con estas modificaciones se pudo obtener un funcionamiento acorde a lo esperado, visualizando valores de impedancia en el display con un error aproximado del 15 %. Este error se debe a la alinealidad que genera el circuito. Como hipótesis, es posible considerar que se debe a que la fuente de corriente ya no se comporta linealmente con estos nuevos valores de corriente (se aumentaron 100 veces).

Fue posible observar el correcto funcionamiento de las rutinas de impresión por display, conversión ADC, obtención de la mediana a través de ordenamiento de tablas por burbujeo, utilización de pulsadores externos, timers, interrupciones, PWM. Se pudo observar, con el uso de un osciloscopio, el funcionamiento de la rutina de corrección de electrodos, observando un valor de continua. En la imagen a continuación se puede ver cómo en una primera etapa no

hay corriente circulando, luego se observa la senoidal que sirve para medir y finalmente un valor de continua que permite corregir el valor de la impedancia.



8. Conclusiones

Como mejoras, se puede mencionar que es altamente recomendable contar con una opción más en el menú para determinar si dos electrodos están o no en cortocircuito. Para ello, se emplea el segundo multiplexor (que aparece en el diseño pero no forma parte de este proyecto). Se podría realizar inyectando señal en cada electrodo y chequeando sobre cada uno de los restantes que no se reciba un nivel elevado de ésta. Además, agre

Por otro lado, sería útil para el usuario que hubiera alguna identificación para ver cuándo mide correctamente cada electrodo, si hay dos en cortocircuito y demás cuestiones que momentáneamente sólo aparecen al final del proceso (leds, por pantalla, etc), así el usuario no tiene que esperar hasta el final para ver si funcionó correctamente o no.

9. Apéndices

9.1. Cálculos

A continuación se muestran los cálculos involucrados en la obtención de los parámetros p por el que habrá que multiplicar a la medición del valor pico para obtener la impedancia. Vale aclarar que estos son valores iniciales y luego serán corregidos durante la calibración automática del equipo.

Los valores de lectura del ADC (8 bits), se procesan de la siguiente manera, para obtener un único valor pico, $\hat{L}$:

1. Se muestrean 9 períodos.
2. Se obtienen los máximos y mínimos locales de cada período.
3. Se ordenan dichos máximos y mínimos en dos tablas, de menor a mayor.
4. Se obtiene la mediana de cada tabla, eligiendo el valor central de la tabla ordenada.
5. Se calcula $\hat{L} = \frac{<MAX> - <MIN>}{2}$.

Una vez realizado esto se tiene en cuenta lo siguiente:

- Como se dijo, $\hat{L}$ es el valor pico (en 8 bits) obtenido luego de procesar la señal muestreada.
- k_{ADC} es la constante que lleva desde valores de lectura de 8 bits del ADC a *volts*.
- G es la ganancia de tensión del amplificador del ADC.

Entonces:

$$Z = \frac{\hat{v}}{\hat{\imath}} = \frac{G\,\hat{v}_{ADC}}{\hat{\imath}} = \frac{G\,k_{ADC}}{\hat{\imath}}\,\hat{L} = \alpha\,\hat{L}$$

Como se ha visto, más allá de los factores que intervienen en la constante, la impedancia es proporcional al valor pico de lectura. Ahora, como la unidad interna de representación son *décimas de mega ohm*, lo que se obtiene es $\beta = \frac{\alpha}{0{,}1\,\mathrm{M\Omega}}$, de forma tal que $\beta\,\hat{L}$ es la impedancia en la unidad deseada.

Dado que β resulta menor a 1, y que no se dispone de una operación de división por hardware, se utilizará el parámetro $p = \lceil \gamma\,\beta \rceil$, donde γ es una potencia de dos. En este caso se eligió $\gamma = 2^8 = 256$ y en consecuencia:

$$Z\ [[decimegohm]] = \beta\,\hat{L} = \frac{p\,\hat{L}}{\gamma} = \frac{p\,\hat{L}}{256} = (p\,\hat{L}) >> 8$$

Entonces el valor del parámetro p está dado por:

$$p = \lceil \gamma\,\beta \rceil = \left\lceil \gamma\,\frac{\alpha}{0{,}1\,\mathrm{M\Omega}} \right\rceil = \left\lceil \gamma\,\frac{G\,k_{ADC}}{0{,}1\,\mathrm{M\Omega}\,\hat{\imath}} \right\rceil$$

Dado que depende de $\hat{\imath}$, la cual es distinta en cada rango de medición, para esto y algunos cálculos extras de errores se utilizó la siguiente planilla.

Rango [MΩ]		I máx [nA]	Escala: x/128 (por soft)	I [nA]	ΔV [mV] si ΔZ = 0,1MΩ	ΔZ [MΩ] si ΔV = kADC	ΔZ [MΩ] si ΔV = 96,5mV	Vexcursión [V]	α [kΩ]	β = α / 0,1 MΩ	p = ⌈γ β⌉	Error absoluto [kΩ]	Error relativo [%]
0	2	200,00	128	200,00	90,91	0,021	0,106	1,82	21,48	0,215	55	100	10,00%
		212,31	100,00%	212,31	96,51	0,020	0,100	1,93	20,24	0,202	52	100	10,00%
0	8	80,00	84	52,50	23,86	0,082	0,404	1,91	81,85	0,818	210	100	2,50%
		83,26	65,63%	54,64	24,84	0,079	0,389	1,99	78,64	0,786	202	100	2,50%
0	20	30,00	92	21,56	9,80	0,199	0,985	1,96	199,28	1,993	511	100	1,00%
		30,29	71,88%	21,77	9,90	0,197	0,975	1,98	197,34	1,973	506	200	2,00%
0	60	30,00	30	7,03	3,20	0,611	3,020	1,92	611,11	6,111	1565	100	0,33%
		30,29	23,44%	7,10	3,23	0,605	2,990	1,94	605,18	6,052	1550	400	1,33%

G OpAmp ADC	Vexcursión deseado [V]	kADC [mV]	γ = 2^k
4,55	2,00	19,53	256

Corriente deseada [nA]	R multiplexada [kΩ]	Corriente final [nA]
30	3300	30,29
80	1200	83,26
120	820	121,80
200	470	212,31

9.2. Hojas de datos

Los componentes empleados son:

Componente	Hoja de datos
Atmega32	http://www.atmel.com/images/doc2503.pdf
OPA625	http://www.ti.com/lit/ds/symlink/opa625.pdf
REG104	http://www.ti.com/lit/ds/symlink/reg104.pdf
CD74HC4067	http://www.ti.com/lit/ds/symlink/cd74hc4067.pdf
SN74LV4052	http://www.ti.com/lit/ds/symlink/sn74lv4052a.pdf
Display LCD	http://elmicro.com/files/lcd/gdm1602a_datasheet.pdf

9.3. Códigos fuente

En las siguientes páginas se incluyen a modo de apéndice todos los archivos que conforman el firmware de equipo.

9.3.1. configs.asm

```
;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||;
;|||||||||||||||||||||||||||||||| Constantes ||||||||||||||||||||||||||||||||;
;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||;

;---------------------------- Configuración del PWM ---------------------------;
;
; Compare Output Mode: COMn1:COMn0 = 1:0 => Clear OCn on compare match
; Waveform Generation Mode: WGMn1:WGMn0 1:1 => Fast PWM
; Clock select: CSn2:CSn1:CSn0 = 0:0:1 => CPU clock, no prescaling
;
; * Timer/Counter-n Control Register: (n = 0, 2. Timer/Counter0, Timer/Counter2)
;     |  FOCn  |  WGMn0 |  COMn1 |  COMn0 |  WGMn1 |  CSn2  |  CSn1  |  CSn0  |
;     |   0    |   1    |   1    |   0    |   1    |   0    |   0    |   1    |
;
; Timer0 Overflow Interrupt Enable: TOIE0 = 1 => Enabled
; Timer2 Overflow Interrupt Enable: TOIE2 = 0 => Disabled
;
; * Timer/Counter Interrupt Mask Register:
;     |  OCIE2 |  TOIE2 | TICIE1 | OCIE1A | OCIE1B |  TOIE1 |  OCIE0 |  TOIE0 |
;     |   x    |   0    |   x    |   x    |   x    |   x    |   x    |   1    |
;
.equ PWM_FAST_PWM_CONFIG_T1 = (1<<COM01) | (1<<WGM01) | (1<<WGM00) | (1<<CS00)
.equ PWM_FAST_PWM_CONFIG_T2 = (1<<COM21) | (1<<WGM21) | (1<<WGM20) | (1<<CS20)
.equ PWM_OFF_PWM_CONFIG     = 0
.equ PWM_OV_INTERRUPT_MASK  = (1<<TOIE0)
.equ PWM_SINE_TABLE_LEN     = 62
.equ PWM_SINE_MEDIAN        = 127

;--------------------- Configuración del Timer1 (16 bits) ---------------------;
;
; Waveform Generation Mode: WGM13:WGM12:WGM11:WGM10 0:0:0:0 => Normal mode
; Clock select: CS12:CS11:CS10 = 0:1:1 => CPU clock, divided by 64 -> 250 kHz
;
; * Timer/Counter1 Control Register A:
;     | COM1A1 | COM1A0 | COM1B1 | COM1B0 |  FOC1A |  FOC1B |  WGM11 |  WGM10 |
;     |   0    |   0    |   0    |   0    |   0    |   0    |   0    |   0    |
;
; * Timer/Counter1 Control Register B:
;     |  ICNC1 |  ICES1 |  *Res* |  WGM13 |  WGM12 |  CS12  |  CS11  |  CS10  |
```

```
;     |    0   |    0   |    0   |    0   |    0   |    0   |    1   |    1   |
;
.equ TIMER1_CLOCK_64_PRESCALER = (1<<CS11) | (1<<CS10)
.equ TIMER1_OFF               = 0
.equ TIMER1_50ms_DELAY_START  = -12500 ; 12500 / 250 kHz = 50 ms

;---------------------------- Configuración del ADC ----------------------------;
;
; Reference Selection: REFS1:REFS0 = 0:1 => AVCC with ext capacitor at AREF pin
; ADC Left Adjust Result: ADLAR = 1 => On (8 bits precision reading ADCH only)
; Analog Channel and Gain Selection Bits: MUX4:MUX3:MUX2:MUX1:MUX0
;   1) ImpedanceMeasure  = 6
;   2) OffsetCalibration = 7
;
; * ADC Multiplexer Selection Register:
;     |  REFS1 |  REFS0 |  ADLAR |  MUX4  |  MUX3  |  MUX2  |  MUX1  |  MUX0  |
;     |    0   |    1   |    1   |    0   |    0   |    x   |    x   |    x   |
;
; ADC enable: ADEN = 1 => Enabled
; ADC Auto Trigger Mode: ADATE = 0 => Disabled
; ADC Interrupt Enable: ADIE = 1 => Enabled
; ADC Prescaler Select Bits: ADPS2:ADPS1:ADPS0 = 1:0:1 => fCk_ADC = fCk/32
;
; * ADC Control and Status Register A:
;     |  ADEN  |  ADSC  |  ADATE |  ADIF  |  ADIE  |  ADPS2 |  ADPS1 |  ADPS0 |
;     |    1   |    0   |    0   |    0   |    1   |    1   |    0   |    1   |
;

.equ ADC_AREF_LEFT_ADJUST_CONFIG = (1<<REFS0) | (1<<ADLAR)
.equ ADC_ENABLE_AUTO_INT_PRESC   = (1<<ADEN)  | (1<<ADIE) | \
                                   (1<<ADPS2) | (1<<ADPS0)
.equ ADC_DISABLE            = 0
.equ ADC_PERIODS_TO_SAMPLE  = 9
.equ ADC_SAMPLES_PER_PERIOD = 37 ; Ya que fADC = 37,037 kHz (sampling freq)
.equ ADC_SAMPLES_TABLE_LEN  = ADC_PERIODS_TO_SAMPLE * ADC_SAMPLES_PER_PERIOD

; Enumerativo para la entrada a medir
.equ ADC_IMPEDANCE_MEASURE  = 6
.equ ADC_OFFSET_CALIBRATION = 7

;--------------------- Configuración de rangos de medición --------------------;
; Valores del multiplexor MUX2
.equ MUX2_x30nA  = 0
.equ MUX2_x80nA  = 1
.equ MUX2_x120nA = 2
.equ MUX2_x200nA = 3

; Enumerativo para los rangos de medición
.equ MEAS_RANGE_1 = 0
.equ MEAS_RANGE_2 = 1
.equ MEAS_RANGE_3 = 2
.equ MEAS_RANGE_4 = 3
;---------------------------- Registros especiales ----------------------------;
.def tmp   = R16        ; Temporario, siempre se podrá pisar sin salvarlo
.def param = R17        ; Parámetro para rutinas
.def iter  = R18        ; Iterador, múltiple uso, salvar al pisarlo
.def iter2 = R19        ; Iterador, múltiple uso, salvar al pisarlo

; XXX XXX XXX XXX XXX XXX -- NOTE -- XXX XXX XXX XXX XXX XXX
; Registros siempre en uso por interrupciones, no usar para otra cosa durante
; las mediciones! --> R20, X (R27:R26), R25:R24, Y (R29:R28)
.def tbl_i  = R20 ; Iterador de tabla en RAM, en uso durante mediciones!
.def tbl_jl = R24 ; Iterador de 16-bits (parte baja), en uso durante mediciones!
.def tbl_jh = R25 ; Iterador de 16-bits (parte alta), en uso durante mediciones!


;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|//////////////////////| Direcciones reservadas en RAM |\\\\\\\\\\\\\\\\\\\\\\\\|;
;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
.dseg
```

```
.org SRAM_START

; Tabla: seno escalado, listo para actualizar el PWM
PWM_SINE_RAM_TABLE:
    .byte PWM_SINE_TABLE_LEN

; Tabla: entrada del ADC muestreada durante ADC_PERIODS_TO_SAMPLE períodos
ADC_SAMPLES_RAM_TABLE:
    .byte ADC_SAMPLES_TABLE_LEN

; Tabla: máximos de cada período, luego se ordenará para obtener la mediana
ADC_MAXS_RAM_TABLE:
    .byte ADC_PERIODS_TO_SAMPLE

; Tabla: mínimos de cada período, luego se ordenará para obtener la mediana
ADC_MINS_RAM_TABLE:
    .byte ADC_PERIODS_TO_SAMPLE

; Conversión de BCD a ASCII, en esta posición queda el resultado en ASCII
BCD_TO_ASCII_CONVERT_RAM:
    .byte 5*16


;|///////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|/////////////////////////| Vector de interrupciones |\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|///////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
.cseg

; Interrupción reset -> MAIN
.org 0x0
    jmp     MAIN

; Interrupción de overflow del Timer0
.org OVF0addr
    jmp     PWM_DUTY_CYCLE_UPDATE_ISR

; Interrupción de conversión completa del ADC
.org ADCCaddr
    jmp     ADC_SAMPLE_STORE_TO_RAM_ISR

; Interrupción de botón de ESC (INT2)
.org INT2addr
    jmp     INT2_ESC_BUTTON_ISR

; Final del vector de interrupciones
.org INT_VECTORS_SIZE


;|///////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|//////////////////////////////| Datos en flash |\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|///////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
PWM_SINE_FLASH_TABLE:
    .db 0x00, 0x0D, 0x19, 0x26, 0x32, 0x3D, 0x48, 0x52, 0x5B, 0x64, 0x6B, 0x72
    .db 0x77, 0x7B, 0x7D, 0x7F, 0x7F, 0x7E, 0x7B, 0x78, 0x73, 0x6D, 0x66, 0x5E
    .db 0x55, 0x4B, 0x40, 0x35, 0x29, 0x1C, 0x10, 0x03, 0xF6, 0xEA, 0xDD, 0xD1
    .db 0xC6, 0xBB, 0xB0, 0xA7, 0x9E, 0x97, 0x90, 0x8B, 0x86, 0x83, 0x81, 0x81
    .db 0x82, 0x84, 0x87, 0x8C, 0x91, 0x98, 0xA0, 0xA9, 0xB3, 0xBD, 0xC8, 0xD4
    .db 0xE0, 0xED

; === Multiplexor MUX2 para cada rango de medición ===
MEAS_RANGE_FLASH_SIN_MUX2_VALUES:
    .db MUX2_x200nA, MUX2_x120nA, MUX2_x80nA, MUX2_x30nA

; === Valor de amplitud para cada rango de medición ===
; 128 --> 100,0 % --> 200,0 nA de corriente pico
;  84 -->  65,6 % -->  52,5 nA de corriente pico
;  92 -->  71,9 % -->  21,6 nA de corriente pico
;  30 -->  23,4 % -->   7,0 nA de corriente pico
MEAS_RANGE_FLASH_SINAMPS:
    .db 128, 128, 128, 128
```

```
180
181 ; === Valores de piso para cada rango de medición, en kilo ohm (16 bit!) ===
182 MEAS_RANGE_FLASH_FLOOR_VALUES:
183     .dw 0, 143, 243, 335 ; Unidad: kohm
184
185 ; === Valores del parámetro p inicial para cada rango de medición (16 bit!) ===
186 MEAS_RANGE_FLASH_P_FACTOR_DEFAULTS:
187     .dw 421, 649, 901, 2607
188
189 ; === Valores de continua de corrección para cada rango de medición ===
190 MEAS_RANGE_FLASH_CONTINUE_MUX2_VALUES:
191     .db MUX2_x30nA, MUX2_x80nA, MUX2_x120nA, MUX2_x200nA
192
193 ; === Valor inicial de calibración del PWM de Offset ===
194 PWM_OFFSET_FLASH_CALIB_VALUE:
195     .db 210, 218, 218, 223
```

9.3.2. main.asm

```
1  #ifdef AVRA
2      .nolist
3      .include "m32def.inc"
4      .list
5  #endif
6
7
8  .include "configs.asm"
9  .include "pwm.asm"
10 .include "adc.asm"
11 .include "measure_routines.asm"
12 .include "MEDIR.asm"
13 .include "BCD.asm"
14 .include "LCD.asm"
15 .include "MENU.asm"
16 .include "CORREGIR_FIN.asm"
17
18
19 ;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||;
20 ;|||||||||||||||||||||||||||||||||||| Main ||||||||||||||||||||||||||||||||||||;
21 ;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||;
22 MAIN:
23     ldi     tmp,HIGH(RAMEND)
24     out     SPH,tmp
25     ldi     tmp,LOW(RAMEND)
26     out     SPL,tmp   ; Stack pointer
27     sei               ; Habilita las interrupciones (global)
28
29 ;----------------------- Configuración de los puertos -----------------------;
30 ;
31 ;       PA0 --> RS   |                          PC0 --> D0   |
32 ;       PA1 --> R/W  | LCD                      PC1 --> D1   |
33 ;       PA2 --> E    |                          PC2 --> D2   |
34 ;       PA3 <-- ButtonLeft                      PC3 --> D3   | LCD
35 ;       PA4 <-- ButtonRight                     PC4 --> D4   |
36 ;       PA5 <-- ButtonOk                        PC5 --> D5   |
37 ;       PA6 <== ADC: ImpedanceMeasure           PC6 --> D6   |
38 ;       PA7 <== ADC: OffsetCalibration          PC7 --> D7   |
39 ;
40 ;
41 ;       PB0 --> A    | MUX2                     PD0 --> S0   |
42 ;       PB1 --> B    |                          PD1 --> S1   | MUX1
43 ;       PB2 <== INT2: ButtonEsc                 PD2 --> S2   |
44 ;       PB3 ==> OC0: PWMSineWave                PD3 --> S3   |
45 ;       PB4 --> S0   |                          PD4 *Unused*
46 ;       PB5 --> S1   | MUX0                     PD5 *Unused*
47 ;       PB6 --> S2   |                          PD6 *Unused*
48 ;       PB7 --> S3   |                          PD7 ==> OC2: PWMOffsetAdjust
49 ;
50     ; Salidas y entradas
```

```
51      ldi     tmp,0b00000111
52      out     DDRA,tmp
53      ldi     tmp,0b11111011
54      out     DDRB,tmp
55      ldi     tmp,0b11111111
56      out     DDRC,tmp
57      ldi     tmp,0b10001111
58      out     DDRD,tmp
59
60      ; Resistencias pull-up
61      ldi     tmp,0b00111000
62      out     PORTA,tmp
63      ldi     tmp,0b00000100
64      out     PORTB,tmp
65      ldi     tmp,0b01110000
66      out     PORTD,tmp
67
68      ; Interrupción de escape
69      ldi     tmp,(1<<INT2) ; Interrupción INT2 (flanco descendente por defecto)
70      out     GICR,tmp
71
72  ;-------------------------------------------------------------------------------;
73      jmp     MENU
74
75      rjmp    MAIN ; Si se llega hasta aquí por error, se comienza de nuevo
76
77
78  ;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
79  ;|////////////////////////| Delay de 50 milisegundos |\\\\\\\\\\\\\\\\\\\\\\\\|;
80  ;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
81  ;
82  ; Utiliza el Timer1. Solo usa el registro temporal, este nunca se salva.
83  ;
84  DELAY_50ms:
85      ldi     tmp,HIGH(TIMER1_50ms_DELAY_START)
86      out     TCNT1H,tmp
87      ldi     tmp,LOW(TIMER1_50ms_DELAY_START)
88      out     TCNT1L,tmp                      ; Carga del contador del Timer1
89
90      ldi     tmp,TIMER1_CLOCK_64_PRESCALER
91      out     TCCR1B,tmp                      ; Activación del Timer1
92
93  keep_waiting:
94      in      tmp,TIFR
95      sbrs    tmp,TOV1        ; Saltea si el flag de overflow está encendido
96      rjmp    keep_waiting
97
98      ldi     tmp,TIMER1_OFF
99      out     TCCR1B,tmp                      ; Desactivación del Timer1
100     ldi     tmp,(1<<TOV1)
101     out     TIFR,tmp                        ; Borrado del flag de overflow
102
103     ret
104
105
106 ;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
107 ;|////////////////////////| Delay en segundos (1 a 255) |\\\\\\\\\\\\\\\\\\\\\\\\|;
108 ;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
109 ;
110 ; param (R17) <- cantidad de segundos, 1 < param < 255
111 ; Utiliza la rutina de delay de 50 ms. Salva todos los registros que arruina.
112 ;
113 DELAY_PARAM_SECONDS:
114     push    iter
115     push    iter2 ; Registros salvados en el stack
116
117     mov     iter,param ; Contador de segundos, se carga con el parámetro
118 loop_1s:
119     ldi     iter2,20   ; Contador de 50 milisegundos: 20 * 50 ms = 1000 ms = 1 s
120 loop_50ms:
```

```
121     rcall   DELAY_50ms
122     dec     iter2       ; Decremento del contador de 50 milisegundos
123     brne    loop_50ms   ; Si se contaron 20 vueltas de 50 ms se deja de repetir
124     dec     iter        ; Decremento del contador de segundos
125     brne    loop_1s     ; Si se contaron 60 vueltas de 1 s se deja de repetir
126
127     pop     iter2
128     pop     iter  ; Registros recuperados del stack
129     ret
```

9.3.3. MENU.asm

```
1  INT2_ESC_BUTTON_ISR:
2          POP R16
3          POP R16
4          SEI      ; Deshace los cambios de la llamada a la interrupción y continúa hacia
               MENU
5
6  MENU:
7          LDI param,MEAS_RANGE_4
8          CALL PWM_SINE_STOP     ; Se inicializa el PWM de senoidal
9          CALL PWM_OFFSET_START ; Se inicializa la referencia del OpAmp
10
11         ; Se limpia la RAM de pantalla LCD
12         LDI R16,80
13         LDI ZH,HIGH(BCD_TO_ASCII_CONVERT_RAM)
14         LDI ZL,LOW(BCD_TO_ASCII_CONVERT_RAM)
15         LDI R17,'*'
16 LCD_RAM_INIT:
17         ST Z+,R17
18         DEC R16
19         BRNE LCD_RAM_INIT
20         CALL LCD_INIT
21
22 MED:
23         LDI R16, 0X01
24         CALL CMNDWRT      ;Pone el cursor al principio de la 1ra línea
25         CALL DELAY_1_6ms
26
27
28         LDI R16,'M'
29         CALL DATAWRT
30         LDI R16,'E'
31         CALL DATAWRT
32         LDI R16,'D'
33         CALL DATAWRT
34         LDI R16,'I'
35         CALL DATAWRT
36         LDI R16,'R'
37         CALL DATAWRT
38         LDI R16,' '
39         CALL DATAWRT
40         LDI R16,' '
41         CALL DATAWRT
42         LDI R16,' '
43         CALL DATAWRT
44         LDI R16,'O'
45         CALL DATAWRT
46         LDI R16,'K'
47         CALL DATAWRT
48
49
50         LDI R16,$C0
51         CALL CMNDWRT
52
53         LDI R16,60
54         CALL DATAWRT
55         LDI R16,62
56         CALL DATAWRT
```

```
57
58 BOTONES:
59
60          IN R17, PINA
61          BST R17,3
62          BRTC CALIBRAR_1
63          BST R17,4
64          BRTC CORREGIR_JMP
65          BST R17,5
66          BRTC MEDIR_JMP
67
68 JMP BOTONES
69
70 MEDIR_JMP:
71          JMP MEDIR_MENU
72
73 CORREGIR_JMP:
74          JMP CORREGIR_1
75
76 CALIBRAR_1:
77          LDI R16, 0X01
78          CALL CMNDWRT      ;Pone el cursor al principio de la 1ra línea
79          CALL DELAY_1_6ms
80
81
82          LDI R16,'C'
83          CALL DATAWRT
84          LDI R16,'A'
85          CALL DATAWRT
86          LDI R16,'L'
87          CALL DATAWRT
88          LDI R16,'I'
89          CALL DATAWRT
90          LDI R16,'B'
91          CALL DATAWRT
92          LDI R16,'R'
93          CALL DATAWRT
94          LDI R16,'A'
95          CALL DATAWRT
96          LDI R16,'R'
97          CALL DATAWRT
98          LDI R16,' '
99          CALL DATAWRT
100         LDI R16,' '
101         CALL DATAWRT
102         LDI R16,' '
103         CALL DATAWRT
104         LDI R16,'O'
105         CALL DATAWRT
106         LDI R16,'K'
107         CALL DATAWRT
108
109
110         LDI R16,$C0
111         CALL CMNDWRT
112
113         LDI R16,60
114         CALL DATAWRT
115         LDI R16,62
116         CALL DATAWRT
117
118         CALL DELAY_50ms
119 BOT:
120         IN R17, PINA
121         BST R17,3
122         BRTC CORREGIR_1
123         BST R17,4
124         BRTC MED_JMP
125         BST R17,5
126         BRTC CALIBRAR_JMP
```

```
127 JMP BOT
128
129 CALIBRAR_JMP:
130         JMP CALIBRAR
131
132 MED_JMP:
133         JMP MED
134
135
136 MEDIR_MENU:
137         CALL MEDIR
138         JMP RESULTADOS_INIC
139         RET
140
141 CALIBRAR:
142     RJMP CALIBRAR
143
144 CORREGIR_1:
145         LDI R16, 0X01
146         CALL CMNDWRT      ;Pone el cursor al principio de la 1ra línea
147         CALL DELAY_1_6ms
148
149
150         LDI R16,'C'
151         CALL DATAWRT
152         LDI R16,'O'
153         CALL DATAWRT
154         LDI R16,'R'
155         CALL DATAWRT
156         LDI R16,'R'
157         CALL DATAWRT
158         LDI R16,'E'
159         CALL DATAWRT
160         LDI R16,'G'
161         CALL DATAWRT
162         LDI R16,'I'
163         CALL DATAWRT
164         LDI R16,'R'
165         CALL DATAWRT
166         LDI R16,' '
167         CALL DATAWRT
168         LDI R16,' '
169         CALL DATAWRT
170         LDI R16,' '
171         CALL DATAWRT
172         LDI R16,'O'
173         CALL DATAWRT
174         LDI R16,'K'
175         CALL DATAWRT
176
177
178         LDI R16,$C0
179         CALL CMNDWRT
180
181         LDI R16,60
182         CALL DATAWRT
183         LDI R16,62
184         CALL DATAWRT
185
186         CALL DELAY_50ms
187 BOT2:
188         IN R17, PINA
189         BST R17,3
190         BRTC MED_JMP1
191         BST R17,4
192         BRTC CALIBRAR_JMP1
193         BST R17,5
194         BRTC CORREGIR
195 JMP BOT2
196
```

```
197 MED_JMP1:
198     JMP MED
199
200 CALIBRAR_JMP1:
201     JMP CALIBRAR_1
202
203 CORREGIR:
204         LDI R22, 0
205         LDI R16, 0X01
206         CALL CMNDWRT
207         CALL DELAY_1_6ms
208
209         LDI R16,'I'
210         CALL DATAWRT
211         LDI R16,'M'
212         CALL DATAWRT
213         LDI R16,'P'
214         CALL DATAWRT
215         LDI R16,'E'
216         CALL DATAWRT
217         LDI R16,'D'
218         CALL DATAWRT
219         LDI R16,'A'
220         CALL DATAWRT
221         LDI R16,'N'
222         CALL DATAWRT
223         LDI R16,'C'
224         CALL DATAWRT
225         LDI R16,'I'
226         CALL DATAWRT
227         LDI R16,'A'
228         CALL DATAWRT
229
230         LDI R16,$C0
231         CALL CMNDWRT
232
233         LDI R16,0X30
234         CALL DATAWRT
235         LDI R16,0X30
236         CALL DATAWRT
237         LDI R16,'k'
238         CALL DATAWRT
239         LDI R16,'O'
240         CALL DATAWRT
241         LDI R16,'h'
242         CALL DATAWRT
243         LDI R16,'m'
244         CALL DATAWRT
245
246     CALL DELAY_50ms
247
248         LDI R20,0
249         LDI R21,0
250
251 BOTONES_2:
252         IN R17 ,PINA
253         BST R17,3  ;Pin 3 del puerto A (AUMENTAR)
254         BRTC AUMENTAR
255         BST R17,4  ;Pin 4 del puerto A (REDUCIR)
256         BRTC REDUCIR_OK
257         BST R17,5  ;PIN 5 DEL PUERTO A (OK)
258         BRTC TMAX_OK
259 JMP BOTONES_2
260
261 REDUCIR_OK:JMP REDUCIR ;CON EL BRANCH NO ALCANZA PARA SALTAR
262 TMAX_OK:JMP TMAX
263
264 AUMENTAR:
265         INC R20
266         CPI R20,10
```

```
267         BRNE AUMENTAR_A
268         INC R21
269         LDI R20,0
270
271
272 AUMENTAR_A:
273
274         CPI  R21,8      ;DEJA HASTA 7,9 MOHM
275         BRSH BOTONES_2
276
277         LDI R16, 0X01
278         CALL CMNDWRT
279         CALL DELAY_1_6ms
280
281         LDI R16,'I'
282         CALL DATAWRT
283         LDI R16,'M'
284         CALL DATAWRT
285         LDI R16,'P'
286         CALL DATAWRT
287         LDI R16,'E'
288         CALL DATAWRT
289         LDI R16,'D'
290         CALL DATAWRT
291         LDI R16,'A'
292         CALL DATAWRT
293         LDI R16,'N'
294         CALL DATAWRT
295         LDI R16,'C'
296         CALL DATAWRT
297         LDI R16,'I'
298         CALL DATAWRT
299         LDI R16,'A'
300         CALL DATAWRT
301
302         LDI R16,$C0
303         CALL CMNDWRT
304
305         LDI R22,48     ;PARA PASAR A ASCII
306         ADD R21,R22
307         ADD R20,R22
308
309         MOV R16,R21
310         CALL DATAWRT
311         MOV R16,R20
312         CALL DATAWRT
313         LDI R16,'k'
314         CALL DATAWRT
315         LDI R16,'O'
316         CALL DATAWRT
317         LDI R16,'h'
318         CALL DATAWRT
319         LDI R16,'m'
320         CALL DATAWRT
321
322         SUB R21,R22   ;VUELVO A LA VARIABLE
323         SUB R20,R22
324
325         CALL DELAY_50ms
326         CALL DELAY_50ms
327
328         JMP BOTONES_2
329
330 REDUCIR:
331         CPI R20,0
332         BREQ REDUCIR_B
333         DEC R20
334         JMP REDUCIR_A
335 REDUCIR_B:
336         CPI R21,0
```

```
337          BREQ SALIDA
338          DEC R21
339          LDI R20,9
340          JMP REDUCIR_A
341 SALIDA:
342          JMP BOTONES_2
343
344
345
346 REDUCIR_A:
347
348          LDI R16, 0X01
349          CALL CMNDWRT
350          CALL DELAY_1_6ms
351
352          LDI R16,'I'
353          CALL DATAWRT
354          LDI R16,'M'
355          CALL DATAWRT
356          LDI R16,'P'
357          CALL DATAWRT
358          LDI R16,'E'
359          CALL DATAWRT
360          LDI R16,'D'
361          CALL DATAWRT
362          LDI R16,'A'
363          CALL DATAWRT
364          LDI R16,'N'
365          CALL DATAWRT
366          LDI R16,'C'
367          CALL DATAWRT
368          LDI R16,'I'
369          CALL DATAWRT
370          LDI R16,'A'
371          CALL DATAWRT
372
373          LDI R16,$C0
374          CALL CMNDWRT
375
376          LDI R22,48     ;PARA PASAR A ASCII
377          ADD R21,R22
378          ADD R20,R22
379
380          MOV R16,R21    ;CAMBIAR
381          CALL DATAWRT
382          MOV R16,R20    ;CAMBIAR
383          CALL DATAWRT
384          LDI R16,'k'
385          CALL DATAWRT
386          LDI R16,'O'
387          CALL DATAWRT
388          LDI R16,'h'
389          CALL DATAWRT
390          LDI R16,'m'
391          CALL DATAWRT
392
393          SUB R21,R22   ;VUELVO A LA VARIABLE
394          SUB R20,R22
395
396          CALL DELAY_50ms
397          CALL DELAY_50ms
398
399          JMP BOTONES_2
400
401 TMAX:
402          LDI R16, 0X01
403          CALL CMNDWRT
404          CALL DELAY_1_6ms
405
406          LDI R16,'T'
```

```
407         CALL DATAWRT
408         LDI R16,'M'
409         CALL DATAWRT
410         LDI R16,'A'
411         CALL DATAWRT
412         LDI R16,'X'
413         CALL DATAWRT
414
415         LDI R16,$C0
416         CALL CMNDWRT
417
418         LDI R16,0X30
419         CALL DATAWRT
420         LDI R16,0X30
421         CALL DATAWRT
422
423         LDI R16,'M'
424         CALL DATAWRT
425         LDI R16,'I'
426         CALL DATAWRT
427         LDI R16,'N'
428         CALL DATAWRT
429
430         LDI R23,0
431         LDI R22,48
432
433         CALL DELAY_50ms
434
435
436     BOTONES_3:
437         IN R17 ,PINA
438         BST R17,3  ;Pin 3 del puerto A (AUMENTAR)
439         BRTC AUMENTAR_T
440         BST R17,4  ;Pin 4 del puerto A (REDUCIR)
441         BRTC REDUCIR_T
442         BST R17,5  ;PIN 5 DEL PUERTO A (OK)
443         BRTC FIN
444     JMP BOTONES_3
445     FIN:
446         LDI R16, 0X01
447         CALL CMNDWRT
448         CALL DELAY_1_6ms
449
450         LDI R16,'E'
451         CALL DATAWRT
452         LDI R16,'S'
453         CALL DATAWRT
454         LDI R16,'P'
455         CALL DATAWRT
456         LDI R16,'E'
457         CALL DATAWRT
458         LDI R16,'R'
459         CALL DATAWRT
460         LDI R16,'E'
461         CALL DATAWRT
462         LDI R16,'.'
463         CALL DATAWRT
464         LDI R16,'.'
465         CALL DATAWRT
466         LDI R16,'.'
467         CALL DATAWRT
468         JMP CORREGIR_ELECTRODO
469
470     REDUCIR_T:
471
472         CPI R23,0
473         BRNE REDUCIR_TP
474         JMP BOTONES_3
475
476
```

```
477     AUMENTAR_T:
478         CPI R23,9
479         BRSH BOTONES_3
480         INC R23
481
482
483     AUMENTAR_TP:
484
485         LDI R16, 0X01
486         CALL CMNDWRT
487         CALL DELAY_1_6ms
488
489         LDI R16,'T'
490         CALL DATAWRT
491         LDI R16,'M'
492         CALL DATAWRT
493         LDI R16,'A'
494         CALL DATAWRT
495         LDI R16,'X'
496         CALL DATAWRT
497
498         LDI R16,$C0
499         CALL CMNDWRT
500
501         ADD R23,R22
502
503         MOV R16,R23
504         CALL DATAWRT
505         LDI R16,0X30
506         CALL DATAWRT
507
508         LDI R16,'M'
509         CALL DATAWRT
510         LDI R16,'I'
511         CALL DATAWRT
512         LDI R16,'N'
513         CALL DATAWRT
514
515         SUB R23,R22
516
517         CALL DELAY_50ms
518
519     JMP BOTONES_3
520
521
522     REDUCIR_TP:
523
524         DEC R23
525         LDI R16, 0X01
526         CALL CMNDWRT
527         CALL DELAY_1_6ms
528
529         LDI R16,'T'
530         CALL DATAWRT
531         LDI R16,'M'
532         CALL DATAWRT
533         LDI R16,'A'
534         CALL DATAWRT
535         LDI R16,'X'
536         CALL DATAWRT
537
538         LDI R16,$C0
539         CALL CMNDWRT
540
541         ADD R23,R22
542
543         MOV R16,R23
544         CALL DATAWRT
545         LDI R16,'O'
546         CALL DATAWRT
```

```
547
548         LDI R16,'M'
549         CALL DATAWRT
550         LDI R16,'I'
551         CALL DATAWRT
552         LDI R16,'N'
553         CALL DATAWRT
554
555         SUB R23,R22
556
557         CALL DELAY_50ms
558
559     JMP BOTONES_3
560
561
562
563
564         FIN2:RJMP FIN2
565
566     ;
            ********************************************************************************************************

567
568         LCD_INIT:
569             CBI PORTA,2
570             CALL DELAY_50ms
571             LDI R16,0X38
572             CALL CMNDWRT
573             LDI R16,0X0E
574             CALL CMNDWRT
575             LDI R16,0X01
576             CALL CMNDWRT
577             CALL DELAY_50ms
578             LDI R16,0X06
579             CALL CMNDWRT
580             RET
581
582         ;
                ********************************************************************************************************

583
584         DELAY_40ms:
585             LDI R17,25
586         DR2:
587             CALL DELAY_1_6ms
588             DEC R17
589             BRNE DR2
590             RET
591
592         ;
                ********************************************************************************************************

593         DELAY_1_6ms:
594             PUSH R17
595             LDI R17,16
596         DR1:
597         CALL DELAY_100us
598         DEC R17
599         BRNE DR1
600         POP R17
601         RET
602
603         ;
                ********************************************************************************************************

604         DELAY_100us:
605             PUSH R17
606             LDI R17,13
607         DR0:  CALL SDELAY
608         DEC R17
```

```
609         BRNE DR0
610         POP R17
611         RET
612
613         ;
                ****************************************************************************************************
614
615         SDELAY:
616               NOP
617               NOP
618               RET
619
620         ;
                ****************************************************************************************************
621         CMNDWRT:
622             OUT PORTC,R16
623             CBI PORTA,0
624             CBI PORTA,1
625             SBI PORTA,2
626             CALL DELAY_100us
627             CBI PORTA,2
628             CALL DELAY_50ms
629             RET
630         ;
                ****************************************************************************************************
631         DATAWRT:
632             OUT PORTC,R16
633             SBI PORTA,0
634             CBI PORTA,1
635             SBI PORTA,2
636             CALL DELAY_100us
637             CBI PORTA,2
638             CALL DELAY_50ms
639             RET
```

9.3.4. MEDIR.asm

```
1  MEDIR:
2      ldi     param,MEAS_RANGE_4
3      call    PWM_OFFSET_START
4
5  medir_gral:
6      call    PWM_SINE_START
7      call    DELAY_50ms
8      call    ADC_SAMPLING_TO_RAM_FROM_IMPEDANCE_MEASURE_IN
9      call    PWM_SINE_STOP
10     call    GET_THE_PEAK_VALUE_IN_R2_FROM_ADC_RAM_TABLE
11     call    GET_DECIMEG_IMPEDANCE_FROM_PARAM_RANGE_R2_PEAK_VALUE_IN_R1_R0
12
13     ldi     ZH,HIGH(MEAS_RANGE_FLASH_FLOOR_VALUES<<1)
14     ldi     ZL,LOW(MEAS_RANGE_FLASH_FLOOR_VALUES<<1)
15
16     mov     R2,param
17     lsl     R2          ; R2 = param * 2
18     clr     tmp
19     add     ZL,R2       ; Z = Z + param * 2
20     adc     ZH,tmp      ; Actualización de la parte alta con el carry (tmp = 0)
21
22     lpm     R2,Z+ ; Lectura del parámetro P desde flash, little-endian (L)
23     lpm     R3,Z  ; Lectura del parámetro P desde flash, little-endian (H)
24
25     cp      R0,R2       ; Comparación, parte baja
26     cpc     R1,R3       ; Comparación, parte alta
27     brsh    medir_listo ; Si R1:R0 (medición actual) >= R3:R2 (piso de rango)
28     dec     param       ; Si no, paso a un rango más bajo
29     brne    medir_gral  ; Repetir medición
```

```
30
31 medir_listo:
32     call    BCD
33     ret
```

9.3.5. CORREGIR_FIN.asm

```
1  CORREGIR_ELECTRODO:
2      ; R21 -> R11 Tiene las decenas de la impedancia en kohms
3      ; R20 -> R10 Tiene la unidad de la impedancia en kohms
4      ; R23 -> R12 Tiene TMAX en cantidades de a 10 min
5
6      clr     iter    ; iterador de la cantidad de minutos de corrección
7      mov     R10,R20 ; copia hacia registros no utilizados por las rutinas
8      mov     R11,R21 ; copia hacia registros no utilizados por las rutinas
9      mov     R12,R23 ; copia hacia registros no utilizados por las rutinas
10
11 continuar_corrigiendo:
12     ; Se mide para saber en qué rango corregir (queda en param)
13     ; o si no es necesario hacerlo (medición en R1:R0)
14     call    MEDIR
15
16     ; Se copia la impedancia medida hacia R3:R2
17     mov     R3,R1
18     mov     R2,R0
19
20     ; Verificación de la impedancia
21     ldi     tmp,10
22     mul     tmp,R11 ; R1:R0 <- Decenas de la impedancia en kohms * 10
23     add     R0,R10  ; R1:R0 <- R1:R0 + Unidad de la impedancia en kohms
24     clr     tmp
25     adc     R1,tmp  ; Suma del carry a la parte alta (ya que tmp = 0)
26
27     ; R1:R0 <- valor de impedancia deseada que se comparará con R3:R2 (medida)
28     cp      R0,R2
29     cpc     R1,R3
30     brsh    final_correccion ; Si R1:R0 (deseada) >= R3:R2 (medida)
31
32     call    PWM_CONTINUE_CORRECTION_START
33     ldi     param,60
34     call    DELAY_PARAM_SECONDS
35     inc     iter
36
37     ; Verificación de TMAX
38     ldi     tmp,10
39     mul     tmp,R12 ; R1:R0 <- TMAX en minutos, como es < 90, sólo interesa R0
40     ; Aquí: R0 = TMAX; iter = TACTUAL (en minutos)
41     cp      iter,R0
42     brsh    final_correccion ; Si iter >= R0, se ha cumplido el tiempo máximo
43
44     rjmp    continuar_corrigiendo
45
46 final_correccion:
47     call    PWM_SINE_STOP
48     call    DELAY_50ms
49     call    MEDIR
50     jmp     RESULTADOS_INIC
```

9.3.6. BCD.asm

```
1
2      BCD:
3
4              LDI ZH,HIGH(BCD_TO_ASCII_CONVERT_RAM)
5              LDI ZL,LOW(BCD_TO_ASCII_CONVERT_RAM)
6              .def rBin1H =r1  ;REGISTROS MULTIPLICACION
7              .def rBin1L =r0
8              .def rBin2H =r19
```

```
9               .def rBin2L =r20
10              .def rmp =r18
11
12
13      Bin2ToAsc5:
14      rcall Bin2ToBcd5
15      ldi rmp,4
16      mov rBin2L,rmp
17  Bin2ToAsc5a:
18      ld rmp,z
19      tst rmp
20      brne Bin2ToAsc5b
21      ldi rmp,' '
22      st z+,rmp
23      dec rBin2L
24      brne Bin2ToAsc5a
25      ld rmp,z
26  Bin2ToAsc5b:
27      inc rBin2L
28  Bin2ToAsc5c:
29      subi rmp,-'0'
30      st z+,rmp
31      ld rmp,z
32      dec rBin2L ;
33      brne Bin2ToAsc5c
34      sbiw ZL,5 ;DIRECCIÓN FINAL
35      ret
36
37
38
39      Bin2ToBcd5:
40      push rBin1H
41      push rBin1L
42      ldi rmp,HIGH(10000)
43      mov rBin2H,rmp
44      ldi rmp,LOW(10000)
45      mov rBin2L,rmp
46      rcall Bin2ToDigit
47      ldi rmp,HIGH(1000)
48      mov rBin2H,rmp
49      ldi rmp,LOW(1000)
50      mov rBin2L,rmp
51      rcall Bin2ToDigit
52      ldi rmp,HIGH(100)
53      mov rBin2H,rmp
54      ldi rmp,LOW(100)
55      mov rBin2L,rmp
56      rcall Bin2ToDigit
57      ldi rmp,HIGH(10)
58      mov rBin2H,rmp
59      ldi rmp,LOW(10)
60      mov rBin2L,rmp
61      rcall Bin2ToDigit
62      st z,rBin1L
63      sbiw ZL,4
64      pop rBin1L
65      pop rBin1H
66      ret
67
68
69  Bin2ToDigit:
70      clr rmp
71  Bin2ToDigita:
72      cp rBin1H,rBin2H
73      brcs Bin2ToDigitc
74      brne Bin2ToDigitb
75      cp rBin1L,rBin2L
76      brcs Bin2ToDigitc
77  Bin2ToDigitb:
78      sub rBin1L,rBin2L
```

```
79      sbc rBin1H,rBin2H
80      inc rmp
81      rjmp Bin2ToDigita
82  Bin2ToDigitc:
83      st z+,rmp
84      ret
```

9.3.7. LCD.asm

```
1
2
3   RESULTADOS_INIC:
4               LDI R19,1 ;Electrodo 1
5               LDI ZH,HIGH(BCD_TO_ASCII_CONVERT_RAM)
6               LDI ZL,LOW(BCD_TO_ASCII_CONVERT_RAM)
7               LDI R22,48
8
9
10  RESULTADOS:
11
12              CALL LCD_INIT ;Funcion para inicializar la LCD
13              MOV R21,R19
14
15              CPI R21,10
16              BRLO RESULTADOS_1
17
18              DIEZ:JMP DIEZ_I
19
20  RESULTADOS_1:
21
22              ADD R19,R22
23              LDI R16,'E'
24              CALL DATAWRT
25              LDI R16,'l'
26              CALL DATAWRT
27              LDI R16,'e'
28              CALL DATAWRT
29              LDI R16,'c'
30              CALL DATAWRT
31              LDI R16,'t'
32              CALL DATAWRT
33              LDI R16,'r'
34              CALL DATAWRT
35              LDI R16,'o'
36              CALL DATAWRT
37              LDI R16,'d'
38              CALL DATAWRT
39              LDI R16,'o'
40              CALL DATAWRT
41              LDI R16,' '
42              CALL DATAWRT
43              MOV R16,R19    ;El numero del electrodo(1,2,..16)
44              CALL DATAWRT
45              LDI R16,$C0    ;Esto hay que revisarlo, es para bajar a la segunda linea del
                    LCD
46              CALL CMNDWRT
47              LD R16,Z+    ;El resultado de la corrección
48              CALL DATAWRT
49              LD R16,Z+    ;El resultado de la corrección
50              CALL DATAWRT
51              LD R16,Z+    ;El resultado de la corrección
52              CALL DATAWRT
53              LD R16,Z+    ;El resultado de la corrección
54              CALL DATAWRT
55              LD R16,Z    ;El resultado de la corrección
56              CALL DATAWRT
57              LDI R16,'k'
58              CALL DATAWRT
59              LDI R16,'O'
```

```
60              CALL DATAWRT
61              LDI R16,'h'
62              CALL DATAWRT
63              LDI R16,'m'
64              CALL DATAWRT
65              SUB R19,R22
66
67              LDI R16,0X0F
68              CALL CMNDWRT
69
70              CALL DELAY_50ms
71
72
73  BOTONES_4:
74          IN R17,PINA
75          BST R17,3  ;Pin 3 del puerto A (AUMENTAR)
76          BRTC SUBIR
77          BST R17,4  ;Pin 4 del puerto A (REDUCIR)
78          BRTC BAJAR
79  JMP BOTONES_4
80
81  SUBIR:
82          ADIW ZL,1 ; Avanza el puntero
83          INC R19
84          CPI R19,17 ;Si se pasa del 16, vuelve al 1
85          BREQ PRIMERO
86          JMP RESULTADOS
87  PRIMERO:
88          LDI R19,16
89          SBIW ZL,1
90          JMP BOTONES_4
91  BAJAR:
92          DEC R19
93          CPI R19,0  ; Si se pasa del 1, vuelve al 16
94          BREQ ULTIMO
95          SBIW ZL,9  ; Retrocede el puntero
96          JMP RESULTADOS
97  ULTIMO:
98          LDI R19,1
99          LDI ZH,HIGH(BCD_TO_ASCII_CONVERT_RAM)
100         LDI ZL,LOW(BCD_TO_ASCII_CONVERT_RAM)
101         LDI R16, 0X01
102         CALL CMNDWRT      ;Pone el cursor al principio de la 1ra línea
103         CALL DELAY_1_6ms
104         JMP RESULTADOS_1
105
106
107 DIEZ_I:
108             SUBI R21,10
109             ADD R21,R22
110
111             LDI R16,'E'
112             CALL DATAWRT
113             LDI R16,'l'
114             CALL DATAWRT
115             LDI R16,'e'
116             CALL DATAWRT
117             LDI R16,'c'
118             CALL DATAWRT
119             LDI R16,'t'
120             CALL DATAWRT
121             LDI R16,'r'
122             CALL DATAWRT
123             LDI R16,'o'
124             CALL DATAWRT
125             LDI R16,'d'
126             CALL DATAWRT
127             LDI R16,'o'
128             CALL DATAWRT
129             LDI R16,' '
```

```
            CALL DATAWRT
            LDI R16,0x31
            CALL DATAWRT
            MOV R16,R21    ;El numero del electrodo(1,2,..16)
            CALL DATAWRT
            LDI R16,$C0    ;Esto hay que revisarlo, es para bajar a la segunda linea del LCD
            CALL CMNDWRT
            LD R16,Z+    ;El resultado de la corrección
            CALL DATAWRT
            LD R16,Z+    ;El resultado de la corrección
            CALL DATAWRT
            LD R16,Z+    ;El resultado de la corrección
            CALL DATAWRT
            LD R16,Z+    ;El resultado de la corrección
            CALL DATAWRT
            LD R16,Z    ;El resultado de la corrección
            CALL DATAWRT
            LDI R16,'k'
            CALL DATAWRT
            LDI R16,'O'
            CALL DATAWRT
            LDI R16,'h'
            CALL DATAWRT
            LDI R16,'m'
            CALL DATAWRT

            SUB R21,R22  ;VUELVO A LA VARIABLE


            LDI R16,0X0F
            CALL CMNDWRT

            CALL DELAY_50ms


            JMP BOTONES_4
```

9.3.8. pwm.asm

```
;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||;
;||||||||| Interrupción de actualización del ciclo de trabajo del PWM |||||||||;
;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||;
PWM_DUTY_CYCLE_UPDATE_ISR:
    push    tmp ; En una interrupción hay que salvar el registro temporal
    in      tmp,SREG
    push    tmp ; También hay que salvar el status register

    ld      tmp,X+     ; Próximo valor del ciclo de trabajo desde RAM
    out     OCR0,tmp   ; Valor actualizado de ciclo de trabajo
    dec     tbl_i      ; Decremento del iterador de la tabla en RAM
    brne    skip_go_beginning ; Saltea si tbl_i no es cero
    ; Go beginning
    rcall   SINE_RAM_TABLE_GO_BEGINNING
skip_go_beginning:
    pop     tmp
    out     SREG,tmp ; Se recupera el status register
    pop     tmp ; Se recupera el registro temporal
    reti


;|////////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|///| Apuntado de la tabla del seno en RAM e inicialización del contador |\\\|;
;|////////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;
; NOTA: tbl_i (R18) y X (R27:R26) siempre en uso.
;
SINE_RAM_TABLE_GO_BEGINNING:
    ldi     XH,HIGH(PWM_SINE_RAM_TABLE)
```

```
30      ldi     XL,LOW(PWM_SINE_RAM_TABLE) ; Tabla de onda escalada en RAM
31      ldi     tbl_i,PWM_SINE_TABLE_LEN   ; Iterador de la tabla en RAM
32      ret
33
34
35  ;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
36  ;|//////////////////////| Carga de la tabla del seno |\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
37  ;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
38  ;
39  ; param (R17) <- escalado de amplitud x 128, ejemplo: 25% = 32/128 => param = 32
40  ; Al finalizar la tabla estará cargada en PWM_SINE_RAM_TABLE, escalada por
41  ; param/128 y con un valor medio de PWM_SINE_MEDIAN. Salva todos los registros
42  ; que arruina.
43  ;
44  LOAD_SINE_RAM_TABLE_SCALED:
45      push    R0
46      push    R1
47      push    tbl_i
48      push    ZH
49      push    ZL
50      push    XH
51      push    XL ; Registros salvados en el stack
52
53      ldi     ZH,HIGH(PWM_SINE_FLASH_TABLE<<1)
54      ldi     ZL,LOW(PWM_SINE_FLASH_TABLE<<1) ; Inicialización de puntero en flash
55      rcall   SINE_RAM_TABLE_GO_BEGINNING ; Carga el puntero X y el contador tbl_i
56
57  loop_sine_table:
58      lpm     tmp,Z+     ; Lectura desde flash, del sample original
59      mulsu   tmp,param  ; Sample escalado y multiplicado x 128 en R1:R0
60      rol     R0 ;> División por 128: se multiplica por 2 el entero de 16 bits
61      rol     R1 ;> con shifts y luego se divide por 256 quedándose con R1 (MSB)
62      ldi     tmp,PWM_SINE_MEDIAN
63      add     R1,tmp     ; En R1 queda el sample más la media
64      st      X+,R1      ; Carga en RAM del sample final
65      dec     tbl_i      ; Decremento del contador
66      brne    loop_sine_table
67
68      pop     XL
69      pop     XH
70      pop     ZL
71      pop     ZH
72      pop     tbl_i
73      pop     R1
74      pop     R0 ; Registros recuperados del stack
75      ret
76
77
78  ;-------------------------------------------------------------------------------;
79  ;---------------- Macros que utilizarán las siguientes rutinas -----------------;
80  ;-------------------------------------------------------------------------------;
81
82  ; Carga en el puntero Z la dirección de flash recibida como argumento + param
83  .macro flash_point_Z_plus_param
84      ldi     ZH,HIGH(@0<<1)
85      ldi     ZL,LOW(@0<<1) ; Puntero en flash
86      clr     tmp       ; Se borra el registro temporario
87      add     ZL,param  ; Se desplaza en la tabla según el parámetro de entrada
88      adc     ZH,tmp    ; Se suma el acarreo a la parte alta (tmp = 0)
89  .endmacro
90
91  ; Escribe el valor del MUX2 según R0, sin chequear su contenido
92  .macro set_MUX2_with_R0_value
93      cli     ; Se desactivan las interrupciones para hacer el cambio
94      in      tmp,PORTB                   ; Lectura desde el puerto B
95      cbr     tmp,(1<<PORTB0)|(1<<PORTB1) ; Limpieza de los bit del MUX2
96      or      tmp,R0                      ; Se graban los bit del MUX2 con R0
97      out     PORTB,tmp                   ; Escritura hacia el puerto B
98      sei     ; Se vuelven a activar las interrupciones
99  .endmacro
```

```
100
101
102 ;|/////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
103 ;|/////////| Encender la senoidal, para el rango de medición deseado |\\\\\\\\\|;
104 ;|/////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
105 ;
106 ; param (R17) <- rango, 4 opciones: MEAS_RANGE_X con X en {1, 2, 3, 4}
107 ; Encenderá el PWM de la senoidal para un rango de medición determinado,
108 ; especificado en el parámetro de entrada, por medio de un valor "enumerativo"
109 ; MEAS_RANGE_X, no hace chequeos de borde, se asume que se recibe un valor
110 ; correcto. Salva todos los registros que arruina.
111 ;
112 PWM_SINE_START:
113     push    R0
114     push    param
115     push    ZH
116     push    ZL ; Registros salvados en el stack
117
118     ; Selección del multiplexor MUX2
119     flash_point_Z_plus_param MEAS_RANGE_FLASH_SIN_MUX2_VALUES
120     lpm     R0,Z
121     set_MUX2_with_R0_value
122
123     ; Creación de tabla en RAM
124     flash_point_Z_plus_param MEAS_RANGE_FLASH_SINAMPS
125     lpm     param,Z  ; Valor correspondiente de amplitud para la rutina
126     rcall   LOAD_SINE_RAM_TABLE_SCALED
127
128     ; Inicialización del puntero X y el contador tbl_i, no se pueden usar más!
129     rcall   SINE_RAM_TABLE_GO_BEGINNING
130
131     ; Configuración del Timer0 como PWM
132     ldi     tmp,PWM_FAST_PWM_CONFIG_T1
133     out     TCCR0,tmp    ; Habilita el PWM en modo rápido
134     in      tmp,TIMSK
135     sbr     tmp,PWM_OV_INTERRUPT_MASK
136     out     TIMSK,tmp    ; Habilita la interrupción de overflow Timer0
137
138     pop     ZL
139     pop     ZH
140     pop     param
141     pop     R0 ; Registros recuperados del stack
142     ret
143
144
145 ;|/////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
146 ;|/////////////| Apagar la onda de salida (dejar PWM al 50 % fijo) |\\\\\\\\\\\\|;
147 ;|/////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
148 ;
149 ; Apaga la onda del PWM, es decir que deja lo deja en su valor medio de forma
150 ; constante (50 % de duty cycle). Salva todos los registros que arruina.
151 ;
152 PWM_SINE_STOP:
153     push    R0 ; Registros salvados en el stack
154
155     ; Se pone el MUX2 en x30nA para minimizar errores de offset
156     ldi     tmp,MUX2_x30nA
157     mov     R0,tmp
158     set_MUX2_with_R0_value
159
160     ; Configuración del Timer0 como PWM
161     ldi     tmp,PWM_FAST_PWM_CONFIG_T1
162     out     TCCR0,tmp    ; Habilita el PWM en modo rápido
163     in      tmp,TIMSK
164     cbr     tmp,PWM_OV_INTERRUPT_MASK
165     out     TIMSK,tmp    ; Deshabilita la interrupción de overflow Timer0
166
167     ldi     tmp,PWM_SINE_MEDIAN
168     out     OCR0,tmp     ; Pone el valor medio en la salida
169
```

```
    pop     R0 ; Registros recuperados del stack
    ret


;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|//| Encender la continua de corrección, para el rango de medición deseado |\|;
;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;
; param (R17) <- rango, 4 opciones: MEAS_RANGE_X con X en {1, 2, 3, 4}
; Encenderá el PWM en el modo de corrección, es decir, señales continuas. Según
; el rango especificado en el parámetro de entrada, por medio de un valor
; "enumerativo" MEAS_RANGE_X, no hace chequeos de borde, se asume que se recibe
; un valor correcto. Salva todos los registros que arruina.
;
PWM_CONTINUE_CORRECTION_START:
    push    R0
    push    ZH
    push    ZL ; Registros salvados en el stack

    ; Selección del multiplexor MUX2
    flash_point_Z_plus_param MEAS_RANGE_FLASH_CONTINUE_MUX2_VALUES
    lpm     R0,Z
    set_MUX2_with_R0_value

    ; Deshabilitación del Timer0 como PWM
    ldi     tmp,PWM_OFF_PWM_CONFIG
    out     TCCR0,tmp   ; Deshabilita el PWM en modo rápido
    in      tmp,TIMSK
    cbr     tmp,PWM_OV_INTERRUPT_MASK
    out     TIMSK,tmp   ; Deshabilita la interrupción de overflow Timer0

    ; Corriente continua máxima, 0 en un sentido, 1 en otro
    ; TODO: ver cuál es el correcto
    sbi     PORTB,PORTB3
    ;cbi     PORTB,PORTB3

    pop     ZL
    pop     ZH
    pop     R0 ; Registros recuperados del stack
    ret


;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|/////////////////| Encender el PWM de Offset de referencia |\\\\\\\\\\\\\\\\|;
;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;
; param (R17) <- rango, 4 opciones: MEAS_RANGE_X con X en {1, 2, 3, 4}
; Lee desde flash (TODO: EEPROM) los datos de calibración de Offset.
; Salva todos los registros que arruina. Incluso param (R17) y R2.
;
PWM_OFFSET_START:
    push    ZH
    push    ZL ; Registros salvados en el stack

    ; Configuración del Timer2 como PWM
    ldi     tmp,PWM_FAST_PWM_CONFIG_T2
    out     TCCR2,tmp  ; Habilita el PWM en modo rápido

    flash_point_Z_plus_param PWM_OFFSET_FLASH_CALIB_VALUE
    lpm     tmp,Z      ; Carga el valor medio (duty cycle) desde flash
    out     OCR2,tmp   ; Pone el valor medio en la salida

    pop     ZL
    pop     ZH ; Registros recuperados del stack
    ret
```

9.3.9. adc.asm

```
;||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||;
```

```
;|||||||||||||| Interrupción del ADC para cada conversión realizada ||||||||||||;
;|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||;
ADC_SAMPLE_STORE_TO_RAM_ISR:
    push    tmp ; En una interrupción hay que salvar el registro temporal

    in      tmp,ADCH    ; Muestra de 8 bits, ya que el ajuste es a izquierda
    st      Y+,tmp
    sbiw    tbl_jl,1    ; Decremento del contador de 16 bits
    brne    skip_stop_sampling ; Saltea si tbl_j no es cero
    ; Stop sampling
    clt ; El borrado del flag T indica que la conversión ha terminado
    ldi     tmp,ADC_DISABLE
    out     ADCSRA,tmp ; Se desactiva el ADC
    out     ADCSRA,tmp ; Se desactiva el ADC
    rjmp    skip_adc_reenabling

skip_stop_sampling:
    sbi     ADCSRA,ADSC ; Inicio de la conversión
skip_adc_reenabling:
    pop     tmp ; Se recupera el registro temporal

    reti


;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|/////////| Registro de 9 períodos en RAM a 37 muestras por período |\\\\\\\\\|;
;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;
; Al finalizar la tabla estará cargada en ADC_SAMPLES_RAM_TABLE con
; ADC_SAMPLES_TABLE_LEN muestras. Salva todos los registros que arruina.
;
ADC_SAMPLING_TO_RAM_FROM_IMPEDANCE_MEASURE_IN:
    ldi     YH,HIGH(ADC_SAMPLES_RAM_TABLE)
    ldi     YL,LOW(ADC_SAMPLES_RAM_TABLE)     ; Inicialización del puntero
    ldi     tbl_jh,HIGH(ADC_SAMPLES_TABLE_LEN)
    ldi     tbl_jl,LOW(ADC_SAMPLES_TABLE_LEN) ; Inicialización del contador
    set ; El flag T de SREG indicará que el muestreo está en curso

    ldi     tmp,ADC_AREF_LEFT_ADJUST_CONFIG | ADC_IMPEDANCE_MEASURE
    out     ADMUX,tmp

    ldi     tmp,ADC_ENABLE_AUTO_INT_PRESC
    out     ADCSRA,tmp

    sbi     ADCSRA,ADSC ; Inicio de la conversión

    ; Esperar hasta que todo esté muestreado en RAM
wait_for_sampling_completion:
    brts    wait_for_sampling_completion

    ret
```

9.3.10. measure_routines.asm

```
;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|////////| Búsqueda de extremos locales (máximo y mínimo por período) |\\\\\\\\|;
;|//////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;
; Genera ADC_MAXS_RAM_TABLE y ADC_MINS_RAM_TABLE con los máximos y mínimos
; locales de cada período, luego las tablas son ordenadas por otra rutina para
; encontrar ambas medianas. Salva todos los registros que arruina.
;
SEARCH_FOR_LOCAL_EXTREMES_AND_LOAD_MIN_MAX_TABLES:
    push    R1
    push    R2
    push    R3
    push    iter
    push    iter2
    push    XH
```

```
16      push    XL
17      push    YH
18      push    YL
19      push    ZH
20      push    ZL ; Registros salvados en el stack
21
22      ldi     XH,HIGH(ADC_SAMPLES_RAM_TABLE)
23      ldi     XL,LOW(ADC_SAMPLES_RAM_TABLE) ; Puntero a la tabla de muestras
24      ldi     YH,HIGH(ADC_MAXS_RAM_TABLE)
25      ldi     YL,LOW(ADC_MAXS_RAM_TABLE)    ; Puntero a la tabla de máximos
26      ldi     ZH,HIGH(ADC_MINS_RAM_TABLE)
27      ldi     ZL,LOW(ADC_MINS_RAM_TABLE)    ; Puntero a la tabla de mínimos
28
29      ldi     iter,ADC_PERIODS_TO_SAMPLE    ; Contador para cada período
30  loop_periods:
31      ldi     iter2,ADC_SAMPLES_PER_PERIOD  ; Contador para cada sample
32      ld      R2,X  ; Registro para alojar el máximo, se carga con el primer valor
33      ld      R3,X  ; Registro para alojar el mínimo, se carga con el primer valor
34  loop_samples_one_period:
35      ; Se buscará mínimo y máximo (enteros no signados) dentro de este loop
36      ld      R1,X+ ; Carga de un sample y post incremento
37      cp      R2,R1 ; Si R2 >= R1 NO hay que actualizar el máximo (R2)
38      brsh    skip_update_max
39      mov     R2,R1 ; Actualización del máximo, si R1 > R2
40  skip_update_max:
41      cp      R1,R3 ; Si R1 >= R3 NO hay que actualizar el mínimo (R3)
42      brsh    skip_update_min
43      mov     R3,R1 ; Actualización del mínimo, si R1 < R3
44  skip_update_min:
45      dec     iter2 ; Decremento del contador de samples
46      brne    loop_samples_one_period
47      ; En este punto, el máximo del período está en R2 y el mínimo en R3
48      st      Y+,R2 ; Agregado del máximo en la tabla y post incremento
49      st      Z+,R3 ; Agregado del mínimo en la tabla y post incremento
50      dec     iter  ; Decremento del contador de períodos
51      brne    loop_periods
52
53      pop     ZL
54      pop     ZH
55      pop     YL
56      pop     YH
57      pop     XL
58      pop     XH
59      pop     iter2
60      pop     iter
61      pop     R3
62      pop     R2
63      pop     R1 ; Registros recuperados del stack
64      ret
65
66
67  ;|/////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
68  ;|///| Obtención de la mediana de la tabla apuntada por Z, de largo param |\\\|;
69  ;|/////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
70  ;
71  ; param (R17) <- largo de la tabla apuntada por Z
72  ; Ordena por el método de burbujeo una tabla de enteros no signados en memoria,
73  ; apuntada por Z, de largo param (R17). Luego toma el valor del medio para el
74  ; caso en que la cantidad de elementos sea impar, o el izquierdo de los dos
75  ; centrales, en el caso en que la cantidad de elementos es par. De esta forma
76  ; devuelve la mediana en R1. Salva todos los registros que arruina, incluso R17
77  ; y Z. NOTA: en el caso en que la cantidad de elementos fuera par, debería
78  ; devolver la media aritmética de ambos valores centrales, pero este caso no se
79  ; va a dar.
80  ;
81  CALCULATE_TO_R1_MEDIAN_IN_TABLE_POINTED_BY_Z_LENGTH_IN_PARAM:
82      push    R2
83      push    R3
84      push    iter
85      push    YH
```

```
86      push    YL ; Registros salvados en el stack
87
88      ; Ordenamiento por burbujeo, en una tabla pequeña no es un algoritmo tan
89      ; ineficiente, para la aplicación se justifica
90  table_has_changed_loop:
91      clt ; El flag T de SREG indicará si la tabla ha cambiado, en principio no
92      mov     iter,param ; Se inicializa el contador
93      dec     iter       ; Se mirará uno hacia adelante, se recorrerá uno menos
94      mov     YH,ZH
95      mov     YL,ZL      ; Se inicializa el puntero Y en Z
96  loop_table_elements:
97      ld      R2,Y       ; Carga un elemento de la tabla
98      ldd     R3,Y+1     ; Carga el siguiente elemento de la tabla
99      cp      R3,R2      ; Compara, si R3 >= R2, no hay que intercambiarlos
100     brsh    do_not_interchange
101     set ; La tabla va a cambiar, entonces se indica en el flag T
102     st      Y,R3       ; En la primera posición se pone la segunda
103     std     Y+1,R2     ; En la segunda posición se pone la primera
104 do_not_interchange:
105     adiw    YL,1       ; Incremento de Y
106     dec     iter       ; Decremento del contador
107     brne    loop_table_elements
108     brts    table_has_changed_loop
109
110     ; En este punto la tabla está ordenada, ahora la mediana se obtendrá de
111     ; la mitad de la misma
112     clr     R2
113     mov     tmp,param
114     asr     tmp        ; tmp/2: índice de la mitad de la tabla
115     mov     YH,ZH
116     mov     YL,ZL      ; Se inicializa el puntero Y en Z
117     add     YL,tmp     ; Se le suma a Y el lugar de la mitad de la tabla
118     adc     YH,R2      ; Se suma el acarreo a la parte alta (R2 = 0)
119     ld      R1,Y       ; Finalmente se obtiene la mediana en R1 para devolverla
120
121     pop     YL
122     pop     YH
123     pop     iter
124     pop     R3
125     pop     R2 ; Registros recuperados del stack
126     ret
127
128
129 ;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
130 ;|////| Procesamiento de muestras del ADC en RAM para obtener valor pico |\\\\|;
131 ;|////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
132 ;
133 ; Al finalizar, R2 contendrá el valor pico en bits, de la medición muestreada en
134 ; RAM por el ADC. Salva todos los registros que arruina.
135 ;
136 GET_THE_PEAK_VALUE_IN_R2_FROM_ADC_RAM_TABLE:
137     push    R1
138     push    param
139     push    ZH
140     push    ZL ; Registros salvados en el stack
141
142     rcall   SEARCH_FOR_LOCAL_EXTREMES_AND_LOAD_MIN_MAX_TABLES
143
144     ldi     param,ADC_PERIODS_TO_SAMPLE ; Cantidad de elementos en tablas
145     ldi     ZH,HIGH(ADC_MAXS_RAM_TABLE)
146     ldi     ZL,LOW(ADC_MAXS_RAM_TABLE)  ; Puntero a la tabla de máximos
147     rcall   CALCULATE_TO_R1_MEDIAN_IN_TABLE_POINTED_BY_Z_LENGTH_IN_PARAM
148     mov     R2,R1                       ; R2 = MAX
149
150     ldi     ZH,HIGH(ADC_MINS_RAM_TABLE)
151     ldi     ZL,LOW(ADC_MINS_RAM_TABLE)  ; Puntero a la tabla de mínimos
152     rcall   CALCULATE_TO_R1_MEDIAN_IN_TABLE_POINTED_BY_Z_LENGTH_IN_PARAM
153     sub     R2,R1                       ; R1 = MIN, valor pico = (MAX-MIN)/2
154     lsr     R2                          ; R2 contiene el valor pico medido
155
```

```
    pop     ZL
    pop     ZH
    pop     param
    pop     R1 ; Registros recuperados del stack
    ret


;|///////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;|///| Obtener impedancia en décimas de mega ohms, según rango de medición |\\|;
;|///////////////////////////////////////|\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\|;
;
; param (R17) <- rango, 4 opciones: MEAS_RANGE_X con X en {2, 8, 20, 60}
; R2 <- valor pico de la medición en bits (valor de lectura L)
; Al finalizar R1:R0 contendrá el valor de impedancia en décimas de mega ohm.
; Lee desde flash (TODO: EEPROM) los datos de calibración de cada rango.
; Salva todos los registros que arruina. Incluso param (R17) y R2.
;
GET_DECIMEG_IMPEDANCE_FROM_PARAM_RANGE_R2_PEAK_VALUE_IN_R1_R0:
    push    R2
    push    R3
    push    R4
    push    R5
    push    R6
    push    R7
    push    param
    push    ZH
    push    ZL ; Registros salvados en el stack

    ldi     ZH,HIGH(MEAS_RANGE_FLASH_P_FACTOR_DEFAULTS<<1)
    ldi     ZL,LOW(MEAS_RANGE_FLASH_P_FACTOR_DEFAULTS<<1)

    lsl     param      ; param = param * 2
    clr     tmp
    add     ZL,param   ; Z = Z + param
    adc     ZH,tmp     ; Actualización de la parte alta con el carry (tmp = 0)

    lpm     R6,Z+ ; Lectura del parámetro P desde flash, little-endian (L)
    lpm     R7,Z  ; Lectura del parámetro P desde flash, little-endian (H)

    ; Multiplicación: R2:R1:R0 = R7:R6 * R2
    mul     R7,R2      ; R1:R0 = R2 * R7, parte alta del producto
    mov     R5,R1      ; Se va a salvar en R5:R4
    mov     R4,R0      ; R5:R4 = R1:R0
    mul     R6,R2      ; R1:R0 = R2 * R6, parte baja del producto
    clr     R2         ; R2 = 0
    add     R1,R4      ; R1 = R1 + R4, suma de ambas partes del producto
    adc     R2,R5      ; R2 = R5 + carry

    ; División por 256 (el parámetro P fue calculado para esto)
    mov     R0,R1      ; Basta con desplazar los dos registros
    mov     R1,R2      ; Resultado de la medición en décimas de mega ohm en R1:R0

    pop     ZL
    pop     ZH
    pop     param
    pop     R7
    pop     R6
    pop     R5
    pop     R4
    pop     R3
    pop     R2 ; Registros recuperados del stack
    ret


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;; Solo para testear, se cargan estos datos en RAM, generados con octave ;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;TEST_SAMPLES_FLASH_TABLE:
;     .db 0x81, 0x90, 0xA2, 0xAF, 0xC0, 0xC9, 0xD5, 0xDB, 0xE2, 0xE4, 0xE5, 0xDE
;     .db 0xD9, 0xD2, 0xC4, 0xB7, 0xAB, 0x98, 0x87, 0x77, 0x66, 0x56, 0x48, 0x3B
```

```
;       .db 0x30, 0x29, 0x1F, 0x1E, 0x1B, 0x1C, 0x22, 0x2A, 0x36, 0x40, 0x50, 0x5E
;       .db 0x6F, 0x7E, 0x92, 0xA2, 0xAF, 0xBF, 0xCC, 0xD3, 0xDD, 0xE0, 0xE4, 0xE4
;       .db 0xE0, 0xD9, 0xD1, 0xC6, 0xB9, 0xA9, 0x99, 0x8B, 0x77, 0x67, 0x57, 0x48
;       .db 0x3D, 0x2F, 0x25, 0x20, 0x1B, 0x1E, 0x20, 0x23, 0x2C, 0x34, 0x40, 0x4E
;       .db 0x5F, 0x6F, 0x80, 0x8F, 0xA2, 0xB1, 0xC0, 0xCB, 0xD5, 0xDA, 0xE3, 0xE3
;       .db 0xE2, 0xDF, 0xDA, 0xD0, 0xC5, 0xBA, 0xAC, 0x9D, 0x89, 0x79, 0x69, 0x56
;       .db 0x4A, 0x3C, 0x31, 0x29, 0x22, 0x1D, 0x1D, 0x1E, 0x21, 0x2B, 0x34, 0x41
;       .db 0x4C, 0x5E, 0x6F, 0x7C, 0x8F, 0xA0, 0xAD, 0xBF, 0xC9, 0xD5, 0xDA, 0xE1
;       .db 0xE3, 0xE5, 0xE0, 0xDB, 0xD2, 0xC6, 0xB9, 0xAC, 0x9B, 0x8B, 0x7C, 0x67
;       .db 0x5A, 0x4C, 0x3D, 0x30, 0x29, 0x23, 0x1E, 0x1B, 0x1C, 0x24, 0x28, 0x35
;       .db 0x3D, 0x4B, 0x5E, 0x6C, 0x7F, 0x8D, 0xA0, 0xAE, 0xBF, 0xCA, 0xD6, 0xDA
;       .db 0xE3, 0xE4, 0xE2, 0xE0, 0xDB, 0xD1, 0xC7, 0xBB, 0xAE, 0x9B, 0x8C, 0x79
;       .db 0x68, 0x5A, 0x4A, 0x3C, 0x2F, 0x2A, 0x22, 0x1C, 0x1B, 0x1F, 0x24, 0x28
;       .db 0x31, 0x3F, 0x4C, 0x5D, 0x6C, 0x7E, 0x8F, 0x9D, 0xAE, 0xBC, 0xC8, 0xD2
;       .db 0xDD, 0xE1, 0xE4, 0xE2, 0xE2, 0xDD, 0xD3, 0xC9, 0xBD, 0xAE, 0x9B, 0x8C
;       .db 0x7B, 0x6A, 0x59, 0x4A, 0x3F, 0x32, 0x27, 0x20, 0x1E, 0x1C, 0x1E, 0x22
;       .db 0x2B, 0x32, 0x3F, 0x4C, 0x5D, 0x6C, 0x7B, 0x8E, 0x9F, 0xAD, 0xBB, 0xCA
;       .db 0xD2, 0xDB, 0xDF, 0xE3, 0xE4, 0xE0, 0xDA, 0xD2, 0xC8, 0xBB, 0xAD, 0x9E
;       .db 0x8C, 0x7B, 0x6A, 0x5A, 0x4C, 0x3F, 0x31, 0x29, 0x22, 0x1D, 0x1D, 0x1D
;       .db 0x22, 0x2A, 0x33, 0x3D, 0x4A, 0x5C, 0x6A, 0x7B, 0x8C, 0x9C, 0xAC, 0xBC
;       .db 0xC7, 0xD4, 0xDA, 0xE2, 0xE2, 0xE4, 0xE2, 0xDC, 0xD5, 0xCA, 0xBD, 0xAE
;       .db 0x9E, 0x8C, 0x7C, 0x6B, 0x59, 0x4C, 0x3D, 0x31, 0x2B, 0x20, 0x1F, 0x1D
;       .db 0x1E, 0x23, 0x29, 0x31, 0x3F, 0x4C, 0x5C, 0x6A, 0x79, 0x8E, 0x9D, 0xAD
;       .db 0xBB, 0xC8, 0xD3, 0xDB, 0xDF, 0xE3, 0xE2, 0xE1, 0xDA, 0xD3, 0xCB, 0xBC
;       .db 0xB0, 0xA0, 0x8F, 0x7E, 0x6E, 0x5C, 0x4B, 0x3E, 0x34, 0x2B, 0x21, 0x1C
;       .db 0x1E, 0x1D, 0x20, 0x29, 0x30, 0x3F, 0x4B, 0x59, 0x69, 0x00
;
;TEST_SAMPLES_FLASH:
;       ldi     ZH,HIGH(TEST_SAMPLES_FLASH_TABLE<<1)
;       ldi     ZL,LOW(TEST_SAMPLES_FLASH_TABLE<<1)
;       ldi     XH,HIGH(ADC_SAMPLES_RAM_TABLE)
;       ldi     XL,LOW(ADC_SAMPLES_RAM_TABLE)
;       ldi     YH,HIGH(ADC_SAMPLES_TABLE_LEN) ; Ojo: no es un puntero!
;       ldi     YL,LOW(ADC_SAMPLES_TABLE_LEN)  ; Es un contador de 16 bits!
;loop_test_samples_table:
;       lpm     tmp,Z+
;       st      X+,tmp
;       sbiw    YL,1 ; Decremento del contador
;       brne    loop_test_samples_table
;
;       rcall   GET_THE_PEAK_VALUE_IN_R2_FROM_ADC_RAM_TABLE
;here:
;       rjmp    here
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```