PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
Facultad de Ciencias e Ingeniería

DISEÑO DE UN CONTRASTADOR DE TEMPERATURA PARA TERMÓMETROS DE TERMOCUPLAS TIPO K

Anexos

Presentado por el bachiller:

Daniel Arturo Bedoya Martínez

Asesor:

Willy Carrera Soria

Lima, Agosto del 2013

ÍNDICE DE ANEXOS

ANEXO A

En la Fig. A.1 se muestra un ejemplo de una tabla (estandarizada) estipulada por el NIST(National Institute of Standards and Technology).

-40	-1.527	-1.563	-1.600	-1.636	-1.673	-1.709	-1.745	-1.781	-1.817	-1.853
-30	-1.156	-1.193	-1.231	-1.268	-1.305	-1.342	-1.379	-1.416	-1.453	-1.490
-20	-0.777	-0.816	-0.854	-0.892	-0.930	-0.968	-1.005	-1.043	-1.081	-1.118
-10	-0.392	-0.431	-0.469	-0.508	-0.547	-0.585	-0.624	-0.662	-0.701	-0.739
0	-0.000	-0.039	-0.079	-0.118	-0.157	-0.197	-0.236	-0.275	-0.314	-0.353
0	-0.000	0.039	0.079	0.119	0.158	0.198	0.238	0.277	0.317	0.357
10	0.397	0.437	0.477	0.517	0.557	0.597	0.637	0.677	0.718	0.758
20	0.798	0.838	0.879	0.919	0.960	1.000	1.041	1.081	1.122	1.162
30	1.203	1.244	1.285	1.325	1.366	1.407	1.448	1.489	1.529	1.570
40	1.611	1.652	1.693	1.734	1.776	1.817	1.858	1.899	1.940	1.981
50	2.022	2.064	2.105	2.146	2.188	2.229	2.270	2.312	2.353	2.394
60	2.436	2.477	2.519	2.560	2.601	2.643	2.684	2.726	2.767	2.809
70	2.850	2.892	2.933	2.975	3.016	3.058	3.100	3.141	3.183	3.224
80	3.266	3.307	3.349	3.390	3.432	3.473	3.515	3.556	3.598	3.639
90	3.681	3.722	3.764	3.805	3.847	3.888	3.930	3.971	4.012	4.054
100	4.095	4.137	4.178	4.219	4.261	4.302	4.343	4.384	4.426	4.467
110	4.508	4.549	4.590	4.632	4.673	4.714	4.755	4.796	4.837	4.878
120	4.919	4.960	5.001	5.042	5.083	5.124	5.164	5.205	5.246	5.287
130	5.327	5.368	5.409	5.450	5.490	5.531	5.571	5.612	5.652	5.693
140	5.733	5.774	5.814	5.855	5.895	5.936	5.976	6.016	6.057	6.097
150	6.137	6.177	6.218	6.258	6.298	6.338	6.378	6.419	6.459	6.499
160	6.539	6.579	6.619	6.659	6.699	6.739	6.779	6.819	6.859	6.899
170	6.939	6.979	7.019	7.059	7.099	7.139	7.179	7.219	7.259	7.299
180	7.338	7.378	7.418	7.458	7.498	7.538	7.578	7.618	7.658	7.697
190	7.737	7.777	7.817	7.857	7.897	7.937	7.977	8.017	8.057	8.097
200	8.137	8.177	8.216	8.256	8.296	8.336	8.376	8.416	8.456	8.497
210	8.537	8.577	8.617	8.657	8.697	8.737	8.777	8.817	8.857	8.898
220	8.938	8.978	9.018	9.058	9.099	9.139	9.179	9.220	9.260	9.300
230	9.341	9.381	9.421	9.462	9.502	9.543	9.583	9.624	9.664	9.705
240	9.745	9.786	9.826	9.867	9.907	9.948	9.989	10.029	10.070	10.111
250	10.151	10.192	10.233	10.274	10.315	10.355	10.396	10.437	10.478	10.519
°C	0	1	2	3	4	5	6	7	8	9

Figura A.1 - Voltajes estandarizados (en mV) para una termocupla tipo K
Fuente: www.arian.cl/downloads/nt-002.pdf

En la Tabla A1 se mencionan los tipos de termocupla que existen en la actualidad junto con los materiales de los cuales está hecha su juntura, su rango de operación de temperaturas, su máximo voltaje de salida y su uso en la industria.

Tabla A1 - Tipos de Termocupla

Tipo de Termopar	Materiales del termopar	Rango de temperaturas (°C)	Máximo Voltaje (mV)	Uso en la industria
K	Níquel/cromo y Aluminio/cromo	[-180,1372]	54.8	Fundición de metales.
J	Hierro y cobre/níquel	[-180,750]	42.2	Tecnologías de Inyección y Extrusión (plásticos)
T	Cobre y cobre/níquel	[-250,400]	20.8	Industria de Alimentos
R	Platino/Rhodio y Platino	[-50,1767]	21.09	Industria Siderúrgica
S	Platino/Rhodio y Platino	[-50,1767]	18.68	Industria Siderúrgica
B	Platino/Rhodio y Platino/Rhodio	[0,1820]	13.814	Industria Siderúrgica
E	Cromo/Níquel y cobre/níquel	[-40,900]	61.2	Se usan en bajas temperaturas

Fuente de Tabla A1: SMITH, Cecil L. 2009. Basic process measurements. 1era Edición. New Jersey: Wiley. Consulta: 08 de Septiembre del 2012. <http://books.google.com.pe/books?id=Dc9sx_RHgoUC&pg=PA115>

Fuente de Tabla A1: CASTELLANO, Kleyber. 2009. Termocupla [Diapositivas]. Maracaibo: Sin editorial. Consulta: 08 de Septiembre del 2012. <http://www.slideshare.net/blacksaturn/termocupla>

ANEXO B

La siguiente lista menciona algunos de los fabricantes y proveedores, más conocidos a nivel mundial, de los simuladores de termocuplas.

- Altek Industries Corp.Div.of Transmation Products Group
- Automation Service, Test Equipment Div.
- Azonix Corp.
- Beta Products Div.of Hathaway Process Instrumentation Corp.
- Biddle Instruments
- Chino Works Div.Chino Corp.
- Elan Technical Corp.
- Cole-Parmer
- Druck Inc.
- Ever Ready Thermometer Co.Div.of Apogent Technologies Company
- Fluke
- Hart Scientific Inc.
- GE Kaye Instruments Div.of GE Industrial Systems
- Love Controls Co.Div.of Dwyer Instruments
- Mikron Instrument Co.
- Omega Engineering Co.
- Onicon Inc.
- Panalarm Div.Ametek Inc.
- Prime Technology Inc.
- Promac Inc.Div.of Hathaway Process Instrumentation Corp.
- Rochester Instrument Systems Inc.Div.of Ametek
- S-Products Inc.
- Techne Inc.
- Thermo Electric Co. Inc.
- Yokogawa Corp. of America

Fuente: SCRIBD. 2009. Calibradores y Simuladores. Lima.
Consulta: 11 de Septiembre del 2012.
<http://www.scribd.com/doc/47283571/Calibradores-y-Simuladores>

ANEXO C

En la Fig. C.1 pueden apreciarse los circuitos, del esquemático del equipo, correspondientes a los bloques de Fuente de Alimentación y Control. En esta misma figura, también se muestra el circuito Indicador de Alimentación Baja. Por otro lado, en la Fig. C.2 se presentan los circuitos de los bloques de Generación de Señal, Acondicionamiento de Señal y Verificación de Señal, trabajando en conjunto.

En la Fig. C.3 se muestra la máscara de componentes del circuito impreso; mientras que en la Fig. C.4 se aprecia la máscara de conexiones, pero sin los planos de tierra digital y tierra analógica. Estos planos son mostrados en la Fig. C.5. Se muestran a color para diferenciar las dos capas (rojo y azul) utilizadas en el diseño del circuito impreso. El plano de tierra analógica corresponde a la región delimitada que se encuentra en la zona central de la tarjeta. El plano de tierra digital es el que se encuentra cerca del perímetro de la tarjeta del circuito impreso. Puede apreciarse que ambos planos están distribuidos no sólo en una, sino en las dos capas que se utilizan en la tarjeta.

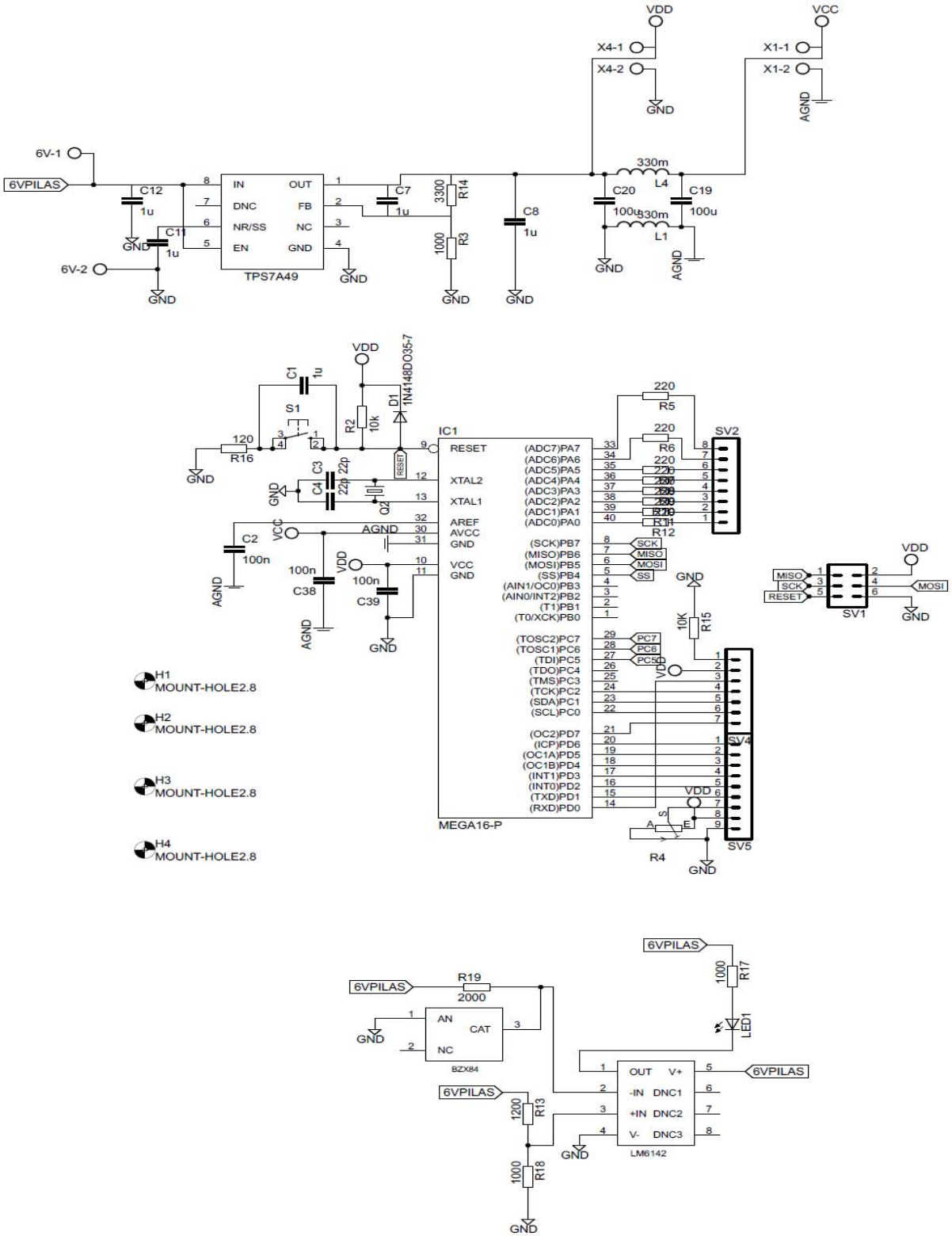

Figura C.1 – Bloques de Fuente de Alimentación, Control y el Circuito Indicador de Alimentación Baja

Fuente: Elaboración Propia

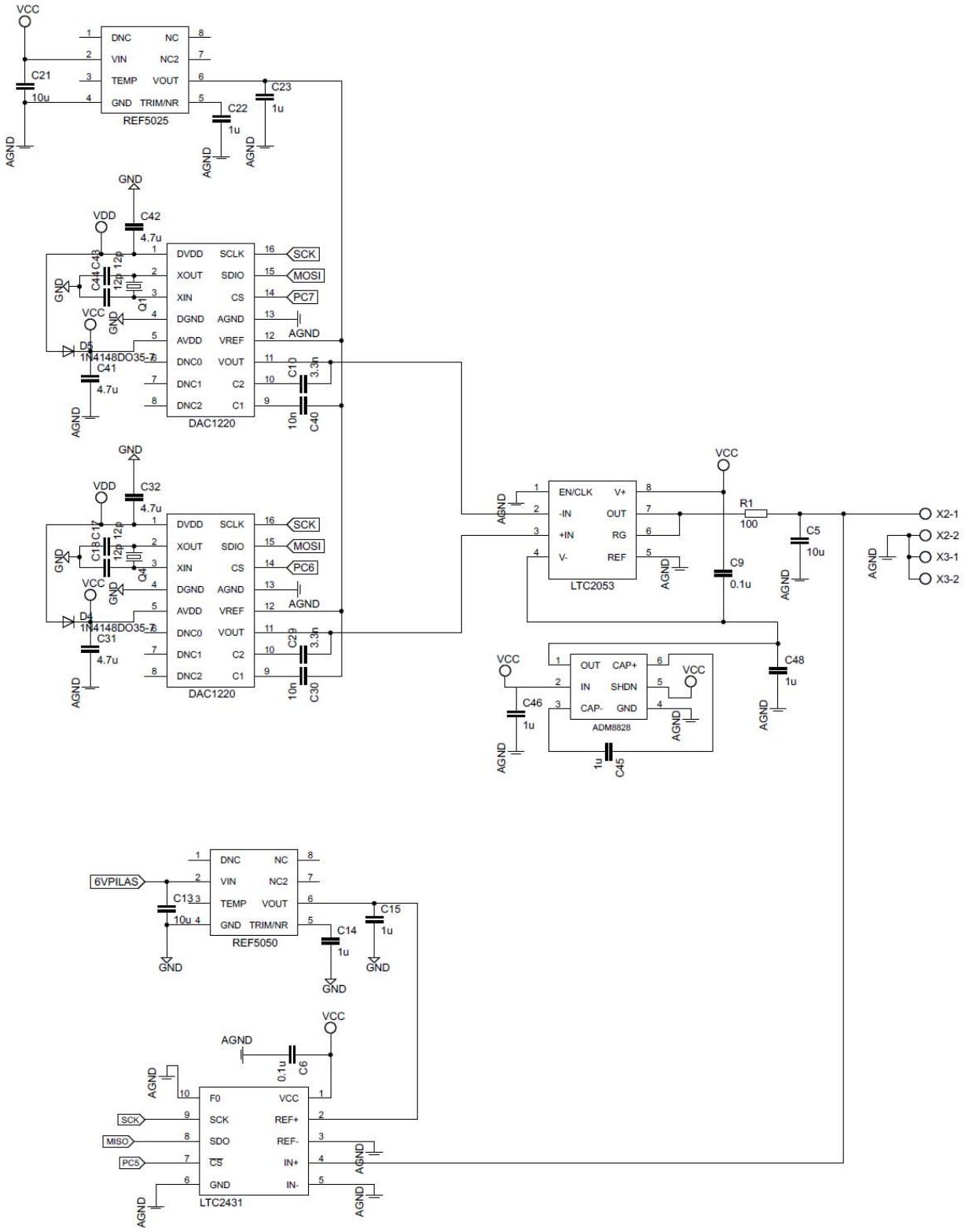

Figura C.2 – Bloques de Generación de Señal, Acondicionamiento de Señal y Verificación de Señal

Fuente: Elaboración Propia

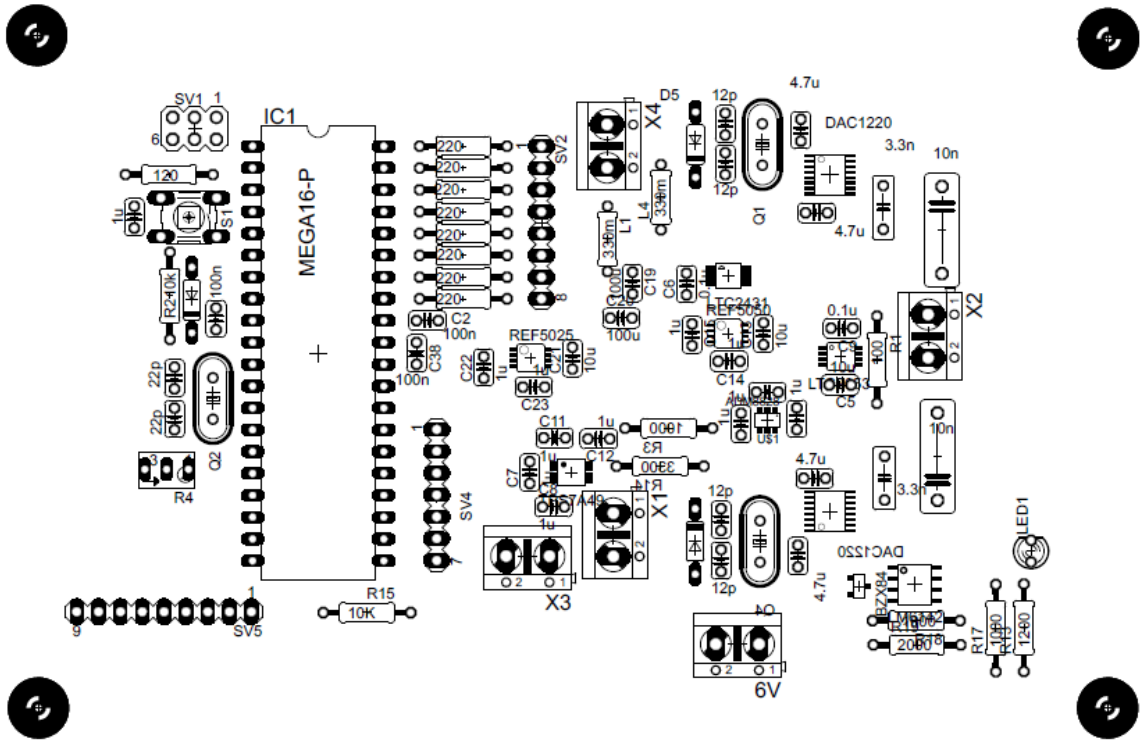

Figura C.3 - Máscara de componentes de la tarjeta del equipo
Fuente: Elaboración Propia

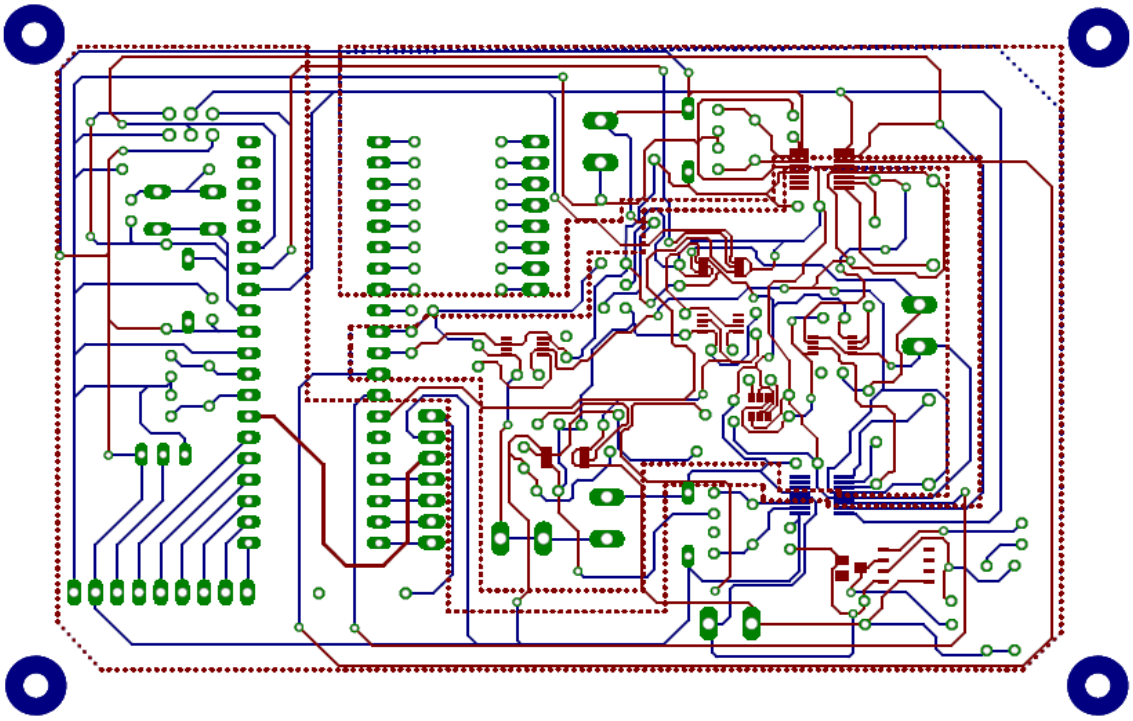

Figura C.4 - Máscara de conexiones de la tarjeta del equipo sin planos de tierra
Fuente: Elaboración Propia

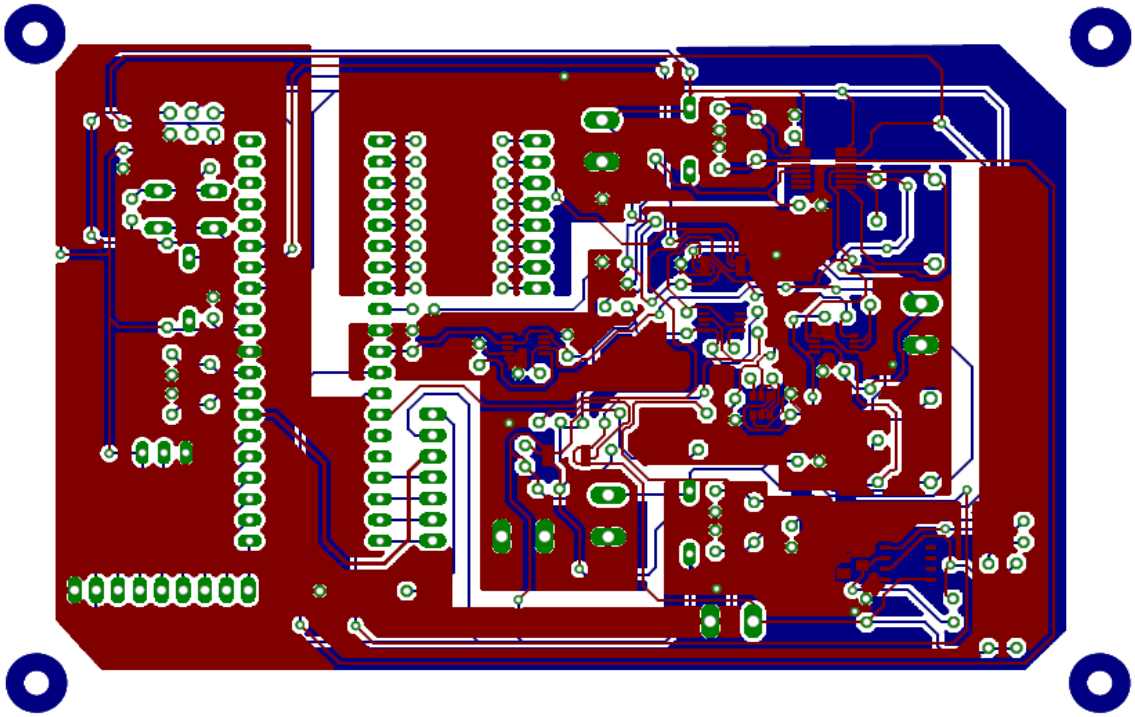

Figura C.5 - Máscara de conexiones de la tarjeta del equipo con los planos de tierra utilizados
Fuente: Elaboración Propia

ANEXO D

En este apartado se presentará el programa de funcionamiento completo del contrastador, escrito en lenguaje ensamblador; y que ha sido simulado por partes con la ayuda del software VMLAB. Debe indicarse que los comentarios son mostrados en color azul para que puedan ser identificados fácilmente en el programa.

Programa Principal:

```
; PROGRAMA CONTRASTADOR DE TEMPERATURA
; AUTOR: DANIEL BEDOYA MARTINEZ
; PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
; FACULTAD DE CIENCIAS E INGENIERÍA
; ****************************************************

.include "c:\vmlab\include\m16def.inc"

.def instruccion=r16
.def dato=r17
.dseg
dacvariable: .byte $03; se reserva 3 bytes para el código del DAC variable
dacfijo: .byte $03; se reserva 3 bytes para el código del DAC fijo
adccodigo: .byte $03; se reserva 3 bytes para el código del ADC
voltaje_temp: .byte $0f; variable de ayuda
dac_resta: .byte $03; se reserva 3 bytes para la resta de los DACs
digitos_finales: .byte $05; variable de ayuda
temp_de16bits: .byte $0f; variable de ayuda
.cseg
.org $000
; Frecuencia osc del sistema del ATmega16 = 2 Mhz
; Frecuencia del dac, pines xin, xout = 2 Mhz
; Frecuencia del pin clock DACs = max 200 Khz
; Frecuencia del pin clock ADC = max 2 Mhz
; Frecuencia de trabajo para el ADC y los DACs = 125 Khz
; ADC ==> cpol=0,cpha=1,dord=0
; DAC ==> cpol=0,cpha=1,dord=0
; spr0=1, spr1=0, spi2x=0
; Vref DAC = 2.5 V, 5 V de alimentación
; Vref ADC = 5 V, 5 V de alimentación

; Puerto A ---> Matricial
; Puerto B ---> Comunicación SPI
; Puerto C ---> Control LCD/Chip Selects
; Puerto D ---> Bus de Datos para LCD
```

```
; r21 contador de dígitos
; r22 habilitador y deshabilitador de teclas
; r25 primer dígito
; r26 segundo digito
; r27 tercer dígito
; r28 si el número es positivo o negativo
; r29 el valor de temperatura ingresado

rjmp    inicio
configura_puertos:

push    r16
ldi     r16,$ff ;puerto D (Bus de Datos del LCD) se configura como salida
out     ddrd,r16
ldi     r16,$ff ;pc0, pc1 y pc2: líneas de control del LCD
out     ddrc,r16;pc0=e pc1=rs pc2=r/w, pc7,pc6,pc5,pc4 configurados como
;salida
cbi     ddrc,pc3 ;pc3 configurado como entrada
ldi     r16,$f0
out     portc,r16
ldi     r16,$bf ;pb6 configurado como entrada
out     ddrb,r16
sbi     portc,pc4
sbi     portc,pc5
sbi     portc,pc6
sbi     portc,pc7
pop     r16
ret

; sck=pb7
; mosi=pb5
; miso=pb6
; ss=pb4

configura_spi:

push    r17
ldi     r17, (1<<ddb5)| (1<<ddb7)
out     ddrb,r17
sbi     ddrb,pb4 ; configuramos ss como salida para evitar cambio a modo
; esclavo, habilitar spi, master, fijar velocidad reloj fck/16
ldi     r17,(1<<spe)|(0<<dord)|(1<<mstr)|(1<<cpha)|(1<<spr0)
; spcr: /spie/spe/dord/mstr/cpol/cpha/spr1/spro/
out     spcr,r17 ; [spi2x spr1 spr0]---> controlan frecuencia de sck
ldi     r17,(0<<spi2x)
out     spsr,r17
```

```
pop     r17
ret

;---- subrutinas del spi----------------

; subrutina que transmite el byte colocado en SPDR
spi_master_transmision:
out     spdr,r16
wait_transmision:
sbis    spsr,spif
rjmp wait_transmision
ret

; subrutina principal para la comunicación con el DAC variable
begin_comuni_dac1:
; comenzamos con la configuración del DAC variable
; insr: r/w/mb1/mb0/0/a3/a2/a1/a0
push    r16
cbi     portc,pc6 ; activamos la comunicación con el DAC variable
nop
nop
ldi     r16,0b00100100
rcall   retar_peque ; 30 us
rcall   spi_master_transmision
rcall   retar_peque ; 30 us
ldi     r16,0b01100000 ; byte 1 del CMR
rcall   spi_master_transmision
ldi     r16,0b10110000 ; byte 0 del CMR
rcall   spi_master_transmision
rcall   retar_peque; 30 us
rcall   nuevo_insr_datos_dac1
sbi     portc,pc6; desactivamos la comunicación con el DAC variable
nop
nop
rcall   retar_peque; 30 us
pop     r16
ret

; subrutina principal para la comunicación con el DAC fijo
begin_comuni_dac2:
; comenzamos con la configuración del DAC fijo
; insr: r/w/mb1/mb0/0/a3/a2/a1/a0
push    r16
cbi     portc,pc7 ; activamos la comunicación con el DAC fijo
nop
nop
```

```
ldi     r16,0b00100100
rcall   retar_peque ; 30 us
rcall   spi_master_transmision
rcall   retar_peque ; 30 us
ldi     r16,0b01100000 ; byte 1 del CMR
rcall   spi_master_transmision
ldi     r16,0b10110000 ; byte 0 del CMR
rcall   spi_master_transmision
rcall   retar_peque ; 30 us
rcall   nuevo_insr_datos_dac2
nop
nop
sbi     portc,pc7 ; desactivamos la comunicación con el DAC fijo
rcall   retar_peque ; 30 us
pop     r16
ret

nuevo_insr_datos_dac1:

push    r16
sbi     portc,pc6 ; desactivamos la comunicación con el DAC variable
rcall   retar_peque ; 30 us
cbi     portc,pc6 ; activamos la comunicación con el DAC variable
nop
nop
rcall   retar_peque ; 30us
rcall   datos_al_dac
nop
nop
rcall   retar_peque ; 30 us
pop     r16
ret

nuevo_insr_datos_dac2:

push    r16
sbi     portc,pc7 ; desactivamos la comunicación con el DAC fijo
rcall   retar_peque ; 30 us
cbi     portc,pc7 ; activamos la comunicación con el DAC fijo
nop
nop
rcall   retar_peque ; 30 us
rcall   datos_al_dac
nop
nop
rcall   retar_peque ; 30 us
```

```
pop     r16
ret

; subrutina que envía la trama de 20 bits a cualquiera de los DACs
datos_al_dac:
push    r16
ldi     r16,0b01000000
rcall   spi_master_transmision
rcall   retar_peque ; 30 us
mov     r16,r1
rcall   spi_master_transmision
mov     r16,r2
rcall   spi_master_transmision
mov     r16,r3
rcall   spi_master_transmision
pop     r16
ret
retar_peque: ; retardo de 30 us
push    r16 ; 2 ciclos
push    r17 ; 2 ciclos
clr     r17 ; 1 ciclo
lazo_retar_peque:
inc     r17 ; 1 ciclo
cpi     r17,10
brne    lazo_retar_peque ; 2 ciclos
pop     r17 ; 2 ciclos
pop     r16 ; 2 ciclos
ret ; 4 ciclos

;---------------------------------------- retardos ----------------------------------------

; subrutina de retardo para t16
retardo_t16:
ldi     r16,0
lazo_ext_t16:
inc     r16
rcall   retardo_50us
cpi     r16,6
brne    lazo_ext_t16
ret

; subrutina de retardo para t17
retardo_t17: ; 10 ciclos = 10 us
push    r17 ; 2 ciclos
push    r16 ; 2 ciclos
ldi     r16,$0
```

```
loop:
inc     r16
cpi     r16,$06
breq    exit
rjmp    loop
exit:
pop     r16 ; 2 ciclos
pop     r17 ; 2 ciclos
ret ; 4 ciclos

; subrutina de retardo para t18
retardo_t18:
ldi     r16,0
lazo_ext_t18:
inc     r16
rcall   retardo_50us
cpi     r16,14
brne    lazo_ext_t18
ret

; subrutina de retardo para t19
retardo_t19:
ldi     r16,0
lazo_ext_t19:
inc     r16
rcall   retardo_50us
cpi     r16,22
brne lazo_ext_t19
ret

retardo_50us:
push r16
push r17
clr r16 ; 1 ciclo
lazo_ext_100us:
clr r17 ; 1 ciclo
lazo_int_100us:
inc r17 ; 1 ciclo
cpi r17,9 ; 1 ciclo
brne lazo_int_100us ;1
inc r16 ; 1 ciclo
cpi r16,2 ; 1 ciclo
brne lazo_ext_100us ; 1 ciclo
pop r17
pop r16
ret
```

```
; subrutina de retardo para el LCD
retardo5xms:
push r17
ldi r17,0
lazo_retardo:
rcall retardoxms
inc r17
cpi r17,10
brne lazo_retardo
pop r17
ret

retardoxms:
push r16 ; 2 ciclos
push r17 ; 2 ciclos
clr r16 ; 1 ciclo
lazo_ext:
clr r17; 1 ciclo
lazo_int:
inc r17 ; 1 ciclo
brne lazo_int ; 2 ciclos
inc r16 ; 1 ciclo
cpi r16,60 ; 1 ciclo
brne lazo_ext  ; 2 ciclos
pop r17 ; 2 ciclos
pop r16 ; 2 ciclos
ret ; 5 ciclos

retardo1ms:
push r17
ldi r17,0
lazo_retardo_1ms:
rcall retardo5xms
inc r17
cpi r17,3
brne lazo_retardo_1ms
pop r17
ret

; subrutina que envía una instrucción al LCD
writeir:
push r17
ldi r17,$f0 ; rw=0, rs=0 y e=0
out portc,r17
ldi r17,$f1 ; rw=0, rs=0 y e=1
out portc,r17
```

```
out portd,instruccion ; se envía la instrucción
ldi r17,$f0 ; rw=0, rs=0 y e=0
out portc,r17
ldi r17,$f4 ; rw=1, rs=0 y e=0
out portc,r17
pop r17
ret

writedr:
push r16
push r17
ldi r16,$f2 ; rw=0, rs=1 y e=0
out portc,r16
ldi r16,$f3 ; rw=0, rs=1 y e=1
out portc,r16
out portd,dato ; se envía el dato
ldi r16,$f2 ; rw=0, rs=1 y e=0
out portc,r16
ldi r16,$f6 ; rw=1, rs=1 y e=0
out portc,r16
pop r17
pop r16
ret

; subrutina que espera a que bf = 0
checkbf:
push r16
push r17
ldi r16,0 ; bus de datos como entrada
out ddrd,r16

lecturabf:
ldi r16,$f4 ; rw=1, rs=0 y e=0
out portc,r16
ldi r16,$f5 ; rw=1, rs=0 y e=1
out portc,r16
nop
in r17,pind ; analiza bit bf (db7)
ldi r16,$f4 ; rw=1, rs=0 y e=0
out portc,r16
andi r17,0b10000000
cpi r17,0
brne lecturabf ; si el LCD esta ocupado => espera
ldi r16,$ff ; bus de datos lcd: salida
out ddrd,r16
pop r17
```

```
pop r16
ret

; subrutina que configura el LCD
configura_lcd:

push r16
push r17
rcall retardo5xms ; espera 15 ms
ldi instruccion,$38 ; trabajar con datos de 8 bits y 2 lineas
rcall writeir
rcall checkbf
;*****************************************************************************
; función seleccionar modo de funcionamiento: 0 0 0 0 0 1 id s
; id=1 --> incrementa la dirección ddram
; id=0 --> decrementa
; s=1 --> desplazamiento de toda la pantalla
; (con id=1 --> desplaz. a la izquierda)
; (con id=0 --> desplaz. a la derecha)
; s=0 --> no desplaza
ldi instruccion,6 ; el cursor avanza a la derecha cada vez que se imprime
rcall writeir ; un caracter
rcall checkbf

;*****************************************************************************
; funcion on/off :      0 0 0 0 1 d c b
; d=0 --> apagar la pantalla; d=1 -->encender
; c=0 --> desactivar cursor;  d=1 -->activar
; b=0 --> no parpadea el caracter señalado por el cursor
ldi instruccion,$0e ; enciende pantalla y muestra cursor
rcall writeir
rcall checkbf
;*****************************************************************************
pop r17
pop r16
ret

; subrutina que calibra el DAC variable
calibrar_dac1:
push r16
push r17
cbi portc,pc6 ; activamos la comunicacion con el DAC variable
ldi r16,0b00100100 ; byte del INSR
rcall retar_peque ; 30 us
rcall spi_master_transmision
rcall retar_peque ; 30 us
```

```
ldi r16,0b00100000 ; byte 1 del CMR 00 10 00 00
rcall spi_master_transmision
ldi r16,0b10110001; byte 0 del CMR 10 11 00 01
rcall spi_master_transmision
rcall retar_peque
sbi portc,pc6
pop r17
pop r16
ret

; subrutina que calibra el DAC fijo
calibrar_dac2:
push r16
push r17
cbi portc,pc7 ; activamos la comunicacion con el DAC fijo
ldi r16,0b00100100 ; byte del INSR
rcall retar_peque ; 30 us
rcall spi_master_transmision
rcall retar_peque ; 30 us
ldi r16,0b00100000 ; byte 1 del CMR 00 10 00 00
rcall spi_master_transmision
ldi r16,0b10110001; byte 0 del CMR 10 11 00 01
rcall spi_master_transmision
rcall retar_peque
sbi portc,pc7
pop r17
pop r16
ret

; subrutina que manda el patrón de reseteo a los DACs para que comiencen
con ; el código $00000
resetear_dacs:
ldi r16,$ff
out ddrc,r16
ldi r16,$f0
out portc,r16
ldi r16,$ff
out ddrb,r16
ldi r16,$00
out portb,r16
rcall retardo5xms
cbi portc,pc6
cbi portc,pc7
rcall retardo_t16
ldi r16,$80
out portb,r16
```

```
rcall retardo_t16
ldi r16,$00
out portb,r16
rcall retardo_t17
ldi r16,$80
out portb,r16
rcall retardo_t18
ldi r16,$00
out portb,r16
rcall retardo_t17
ldi r16,$80
out portb,r16
rcall retardo_t19
ldi r16,$00
out portb,r16
rcall retardo_t16
sbi portc,pc6
sbi portc,pc7
ret

; subrutina para simular el código recibido del ADC
leer_adc:
push r16
push xh
push xl
ldi xh,high(adccodigo)
ldi xl,low(adccodigo)
ldi r16,$00 ; este valor cambia dependiendo del valor de temperatura
st x+,r16
ldi r16,$06 ; este valor cambia dependiendo del valor de temperatura
st x+,r16
ldi r16,$e8 ; este valor cambia dependiendo del valor de temperatura
st x,r16
pop xl
pop xh
pop r16
ret

; **********inicio del programa principal**********
inicio:
ldi r16,high(ramend) ; cargamos la pila
out sph,r16
ldi r16,low(ramend)
out spl,r16

rcall retardoxms ; retardos para el oscilador de los DACs
```

```
rcall retardoxms
rcall resetear_dacs ; se envía patrón de reseteo a los DACs
rcall configura_spi ; se configura la comunicación SPI
rcall retar_peque ; 30 us
rcall configura_puertos ; configuración de puertos de E/S
rcall configura_lcd ; configuracion de la pantalla LCD
rcall ins_limpiar_display
rcall retardo5xms
rcall calibrar_dac1 ; subrutina para calibrar el DAC variable
rcall retardo1ms ; 700 ms
rcall calibrar_dac2 ; subrutina para calibrar el DAC fijo
rcall retardo1ms ; 700 ms
rcall ins_limpiar_display
ldi zh,high(mensaje*2) ; z apunta al inicio del mensaje
ldi zl,low(mensaje*2)
ldi r16,0 ; contador de caracteres a visualizar
rcall envio_mensaje
ldi r22,0 ; se inicializa r22 en cero puesto que es el habilitador de teclas

rcall detecta_neutro
rcall begin_comuni_dac2 ; se envían códigos al DAC fijo
rjmp reiniciar

; acá se envía el mensaje para que el usuario ingrese el valor de temperatura
; deseado
envio_mensaje:
leer_otroc:
rcall checkbf
lpm dato,z+ ; lee y muestra caracter en lcd
rcall writedr
inc r16
cpi r16,20 ; muestra del mensaje, los 20 primeros caracteres
breq segundalinea
rjmp leer_otroc
segundalinea:
rcall checkbf
ldi instruccion,$c0
rcall writeir
ldi r16,0
leer_otroc2:
rcall checkbf
lpm dato,z+
rcall writedr
inc r16
cpi r16,18
breq recibir_dato
```

```
rjmp leer_otroc2

recibir_dato:
rcall checkbf
rcall ins_cambia_linea
ldi r21,0;cargamos el contador de digitos a cero
ret

multipor10:
push r16
push r17
ldi r16,0
mov r17, r24
multi:
add r24,r17
inc r16
cpi r16,4
brne multi
rol r24
pop r17
pop r16
ret

; subrutinas de instrucciones mas usadas
ins_mov_izquierda: ; se mueve el cursor  hacia la izquierda
rcall checkbf
ldi instruccion,$10
rcall writeir
rcall checkbf
ret

ins_mov_derecha:
rcall checkbf
ldi instruccion,$14 ; se mueve el cursos  hacia la derecha
rcall writeir
rcall checkbf
ret

ins_limpiar_display: ; limpiar el display
rcall checkbf
ldi instruccion,$01
rcall writeir
rcall checkbf
ret
```

```
ins_cambia_linea:
rcall checkbf
ldi instruccion,$94 ; esta instrucción cambia a la 3era línea
rcall writeir
rcall checkbf
ret

ins_cambia_4talinea:
rcall checkbf
ldi instruccion,$d4 ; esta instrucción cambia a la 4ta linea
rcall writeir
rcall checkbf
ret

; subrutinas de escritura más usadas
mostrar_1:
rcall checkbf
ldi dato,$31 ; valor 1 en código ascii
rcall writedr
rcall checkbf
ret

mostrar_0:
rcall checkbf
ldi dato,$30 ; valor 0 en código ascii
rcall writedr
rcall checkbf
ret

mostrar_9:
rcall checkbf
ldi dato,$39        ; valor 9 en ascii
rcall writedr
rcall checkbf
ret

mostrar_menos:
rcall checkbf
ldi dato,$2d        ; menos (-) en ascii
rcall writedr
rcall checkbf
ret

mostrar_espacio:
rcall checkbf
ldi dato,$20        ; espacio en ascii
```

```
rcall writedr
rcall checkbf
ret

envia_dato_display:
rcall checkbf
mov dato,r20
rcall writedr
rcall checkbf
ret

;subrutinas de programa
muestra_primer_digito:
cpi r25,45
breq  m_negativo
ldi r20,48       ; se envia el primer digito
add r20,r25
rcall envia_dato_display
rjmp fin_primerdigito
m_negativo:
rcall mostrar_menos
fin_primerdigito:
ret

tipo_numero:
cpi r25, 45
breq registro_negativo
ldi r28,1
rjmp fin_tiponumero
registro_negativo:
ldi r28,2
fin_tiponumero:
ret

hallar_numero:
cpi r28,1
breq hallar_positivo
cpi r28,2
breq hallar_negativo

hallar_positivo:
cpi r21, 1
breq hallar_1_digito_pos
cpi r21, 2
breq hallar_2_digito_pos
cpi r21, 3
```

```
breq hallar_3_digito_pos
hallar_1_digito_pos:
mov r29,r25
rjmp fin_hallarnumero
hallar_2_digito_pos:
mov r24, r25
rcall multipor10
mov r29, r24
add r29, r26
rjmp fin_hallarnumero
hallar_3_digito_pos:
mov r24, r25
rcall multipor10
rcall multipor10
mov r29, r24
mov r24,r26
rcall multipor10
add r29, r24
add r29,r27
rjmp fin_hallarnumero

hallar_negativo:
cpi r21, 2
breq hallar_1_digito_neg
cpi r21, 3
breq hallar_2_digito_neg
hallar_1_digito_neg:
mov r29, r26
rjmp fin_hallarnumero
hallar_2_digito_neg:
mov r24, r26
rcall multipor10
mov r29, r24
add r29, r27
rjmp fin_hallarnumero

fin_hallarnumero:
ret

valida_menos_cero:
cpi r28,2
breq int_menos_cero
rjmp fin_valida_menos_cero
int_menos_cero:
cpi r29,0
breq modifica_menos_cero
```

```
rjmp fin_valida_menos_cero
modifica_menos_cero:
dec r21
ldi r25,0
ldi r28,1
fin_valida_menos_cero:
ret

mensaje:
.db "ingrese temperatura en grados celsius:"

mensaje1:
.db "ingrese valores de -30 a 250 grados "

mensaje2:
.db "temperatura: "

mensaje3:
.db "volt ideal:"

mensaje4:
.db "volt real:"

; envío del mensaje 1
envio_mensaje_2:
leer_otroc_22:
rcall checkbf
lpm dato,z+   ;lee y muestra caracter en LCD
rcall writedr
inc r16
cpi r16,20     ;muestra del texto mensaje sólo 20 caracteres
breq segundalinea_22
rjmp leer_otroc_22
segundalinea_22:
rcall checkbf
ldi instruccion,$c0
rcall writeir
ldi r16,0
leer_otroc2_22:
rcall checkbf
lpm dato,z+
rcall writedr
inc r16
cpi r16,16
breq recibir_dato2
rjmp leer_otroc2_22
```

```
recibir_dato2:
rcall checkbf
rcall ins_cambia_linea
ldi r21,0;se reinicia el contador de caracteres si hubo un número ingresado
;incorrecto
ret

;subrutina de reinicio por valor ingresado fuera del rango
restart_msj:
rcall ins_limpiar_display
ldi zh,high(mensaje1*2) ; z apunta al inicio del mensaje
ldi zl,low(mensaje1*2)
ldi r16,1 ; contador de caracteres empieza con 1
rcall envio_mensaje_2
rcall retardo5xms   ; espera de 5ms
rcall retardo1ms
rcall retardo1ms
rcall ins_limpiar_display  ;limpiar el display
ldi zh,high(mensaje*2);z apunta al inicio del mensaje
ldi zl,low(mensaje*2)
ldi r16,0        ;contador de caracteres
rcall envio_mensaje
rcall retardo5xms
rjmp reiniciar

; programa para el teclado matricial
reiniciar:
ldi r16,$f0     ; configura los 4 lsb como entrada
out ddra,r16    ; y los 4 msb como salida
ldi r16,$0f
out porta,r16  ; activa la resistencia de pull up

begin:
in r16,pina
cpi r16,$0f
brne confirmacion
rjmp begin

confirmacion:
;esta rutina espera un corto periodo de tiempo mientras el interruptor
;se estabiliza para evitar que el micro detecte la misma tecla varias
;veces al igual que posible ruido.
;vuelve a leer las entradas
rcall delay
in r16,pina
cpi r16,$0f
```

```
;si detecta el cero de nuevo, procesa la detección
brne deteccion
;si fue un falso contacto, vuelve a empezar
rjmp begin

deteccion:
clr r17 ; inicializamos el código de escaneo temporal

; buscar las columnas
colscan:
sbis pina,0 ;prueba la columna 0
ldi r24,$01
sbis pina,1 ;prueba la columna 1
ldi r24,$02
sbis pina,2 ;prueba la columna 2
ldi r24,$04
sbis pina,3 ;prueba la columna 3
ldi r24,$08
;ahora que se encontró la columna, tienen que invertirse
;los sentidos del nibble bajo y el alto para buscar la
;fila
rcall invertir
;ahora invertidos, buscaremos si alguna fila está en nivel bajo
in r16,pina
cpi r16,$f0 ;si hay algun cero en las columnas, buscar la tecla
brne filadetectada
rjmp reiniciar ;si no hay fila con cero, devuelve los puertos
;a su estado original y reinicia el programa

filadetectada:
;buscar la fila
filascan:

sbis pina,4 ;prueba la fila 0
rjmp fila0
sbis pina,5 ;prueba la fila 1
rjmp fila1
sbis pina,6 ;prueba la fila 2
rjmp fila2
sbis pina,7 ;prueba la fila 3
rjmp fila3
rjmp reiniciar ;si no halla la fila, reinicia

;a continuación se escribirá el valor correspondiente
;a la fila, se usa el comando rol para multiplicar por 2
;pero se limpia antes la bandera de acarreo para evitar
```

```
;que se sume y entregue valores erróneos.

fila0:
ldi r16,$10
add r24,r16
rjmp adq_valor

fila1:
ldi r16,$20
add r24,r16
rjmp adq_valor

fila2:
ldi r16,$40
add r24,r16
rjmp adq_valor

fila3:
ldi r16,$80
add r24,r16
rjmp adq_valor

adq_valor:     ; valores de la 1ra columna
cpi r24, $11
breq valor1
cpi r24, $12
breq valor2
cpi r24, $14
breq valor3
cpi r24, $18
breq valor_up
cpi r24, $21        ; valores de la 2da columna
breq valor4
cpi r24, $22
breq valor5
cpi r24, $24
breq valor6
cpi r24, $28
breq valor_down_ancla
cpi r24, $41       ; valores de la 3ra columna
breq valor7
cpi r24, $42
breq valor8
cpi r24, $44
breq valor9_ancla
cpi r24, $48
```

```
breq valor_menos_ancla      ; valores de la 4ta columna
cpi r24, $81
breq valor_clr_ancla
cpi r24, $82
breq valor0_ancla
cpi r24, $84
breq valor_menu_ancla
cpi r24, $88
breq valor_enter

; se utilza estos ganchos debido a que las funciones tipo breq salieron de su
;rango
valor_menu_ancla: rjmp valor_menu
valor_menos_ancla:rjmp valor_menos
valor_clr_ancla: rjmp valor_clr
valor_down_ancla:rjmp valor_down
valor9_ancla:rjmp valor9
valor0_ancla:rjmp valor0

valor_enter: ; función de la tecla enter
cpi r22,1
breq inhabilita_enter
cpi r21, 0   ; si r21=0 signfica que no existe nada q tenga q ser procesado por
;lo cual se regresa a esperar los datos
breq inhabilita_enter
rjmp procesa_temperatura
inhabilita_enter:rjmp reiniciar

valor_up: ;función de la tecla flecha up
cpi r22,1
breq inhabilita_up
rjmp incrementa_10
inhabilita_up:rjmp reiniciar

valor1:              ;función de la tecla 1
cpi r22,1
breq inhabilita_1
ldi r18,$01
rjmp loop_lcd
inhabilita_1:rjmp reiniciar

valor2:              ;función de la tecla 2
cpi r22,1
breq inhabilita_2
ldi r18,$02
rjmp loop_lcd
```

```
inhabilita_2:rjmp reiniciar

valor3:          ;función de la tecla 3
cpi r22,1
breq inhabilita_3
ldi r18,$03
rjmp loop_lcd
inhabilita_3:rjmp reiniciar
valor4:          ;función de la tecla 4
cpi r22,1
breq inhabilita_4
ldi r18,$04
rjmp loop_lcd
inhabilita_4:rjmp reiniciar

valor5:          ;función de la tecla 5
cpi r22,1
breq inhabilita_5
ldi r18,$05
rjmp loop_lcd
inhabilita_5:rjmp reiniciar

valor6:          ;función de la tecla 6
cpi r22,1
breq inhabilita_6
ldi r18,$06
rjmp loop_lcd
inhabilita_6:rjmp reiniciar

valor7:          ;función de la tecla 7
cpi r22,1
breq inhabilita_7
ldi r18,$07
rjmp loop_lcd
inhabilita_7:rjmp reiniciar

valor8:          ;función de la tecla 8
cpi r22,1
breq inhabilita_8
ldi r18,$08
rjmp loop_lcd
inhabilita_8:rjmp reiniciar

valor9:          ;función de la tecla 9
cpi r22,1
breq inhabilita_9
```

```
ldi r18,$09
rjmp loop_lcd
inhabilita_9:rjmp reiniciar

valor0:          ; función de la tecla 0
cpi r22,1
breq inhabilita_0
cpi r21, 1
breq primer_cero
brne segundo_cero
primer_cero:
cpi r25,0
breq cero_1
segundo_cero:
ldi r18,$00
rjmp loop_lcd
cero_1:
rjmp reiniciar
inhabilita_0:rjmp reiniciar

valor_clr:      ;funcion de la tecla clear
cpi r22,1
breq inhabilita_clr
cpi r21, 0   ; si r21=0 signfica que no existe nada q tenga q ser borrado por lo
;cual se se regresa a esperar los datos
breq inhabilita_clr
dec r21
rcall ins_mov_izquierda
rcall mostrar_espacio
rcall ins_mov_izquierda
rcall retardo5xms
rjmp reiniciar
inhabilita_clr:rjmp reiniciar

valor_down: ; funcion de la tecla flecha down
cpi r22,1
breq inhabilita_down
rjmp decrementa_10
inhabilita_down:rjmp reiniciar

valor_menos: ; funcion de la tecla menos(2nd)
cpi r22,1
breq inhabilita_menos
cpi r21, 0   ; si es igual a 0 se escribe el signo menos no decrementa el registro
;R21
breq numero_negativo
```

```
brne numero_errado  ; si se ingresa un menos despues de un un digito se
; reinicia
numero_negativo:
inc r21
rcall mostrar_menos
ldi r25,$2d
rcall retardo5xms
rjmp reiniciar
numero_errado:
rjmp restart_msj
inhabilita_menos:rjmp reiniciar

valor_menu:          ;funcion de la tecla menu
ldi r22,0
rcall ins_limpiar_display
ldi zh,high(mensaje*2)          ;z apunta al inicio del mensaje
ldi zl,low(mensaje*2)
ldi r16,0                       ;contador de caracteres
rcall envio_mensaje
rcall retardo5xms
rjmp reiniciar

; se hace la transferencia de los digitos, aqui se pasa de ascii a valor número
;con ayuda del registro r20, en el que se carga el número 48 , para hacer una
;suma posterior con el registro r18
loop_lcd:
inc r21
ldi r20,48
mov r24, r18
cpi r21,1
breq primer_digito
cpi r21,2
breq segundo_digito
cpi r21,3
breq tercer_digito

primer_digito:
mov r25,r24
rjmp continua_proceso
segundo_digito:
mov r26, r24
rjmp continua_proceso
tercer_digito:
mov r27, r24
rjmp continua_proceso
continua_proceso:
```

```
add r20,r18
rcall envia_dato_display
rcall retardo5xms
rcall retardo5xms
verificacion_digitos:
cpi r21,4
breq  msnj_error
rjmp reiniciar
msnj_error:
rjmp restart_msj

delay:
;respaldar variables globales en la pila
push r16
push r17
;inicia secuencia de retraso
loop0: ldi r17,$20
loop1: ldi r16,$01
loop2: dec r16
brne loop2
dec r17
brne loop1
;restaurar las variables desde la pila
pop r17
pop r16
ret

invertir: ;inversión de los pines
ldi r16,0b00001111
out ddra,r16
;activando las resistencias pull-up
ldi r16,$f0
out porta,r16
;ahora el nible más significativo del puerto A es entrada con pullup y el nible
menos significativo del puerto A es salida
ret

desinvertir: ;devolver los pines al estado inicial
ldi r16,0b11110000
out ddra,r16
;activando las resistencias pull-up en la parte alta
ldi r16,$0f
out porta,r16
;ahora la parte baja es entrada con pullup y la parte
;alta son salidas con ceros
ret
```

```
;estructura que procesa la temperatura
procesa_temperatura:
rcall tipo_numero          ; se verifica si el número es negativo o positivo
rcall hallar_numero        ; esta subrutina halla en un sólo registro el número
;ingresado
rcall valida_menos_cero
rjmp valida_temperatura
inicio_proceso_temperatura:
rcall ins_limpiar_display
;en esta etapa se envia el mensaje de temperatura
ldi zh,high(mensaje2*2)              ;z apunta al inicio del mensaje
ldi zl,low(mensaje2*2)
ldi r16,0                            ;contador de caracteres
leer_digito:
rcall checkbf
lpm dato,z+                  ;lee y muestra caracter en LCD
rcall writedr
inc r16
cpi r16,13                   ;muestra solo 13 caracteres
breq proce
rjmp leer_digito
proce:
cpi r21, 1             ; se verifica si es de un digito
breq muestra_1
cpi r21, 2             ; se verifica si es de dos digitos
breq muestra_2
cpi r21, 3             ; se verifica si es de tres digitos
breq muestra_3

muestra_1:
rcall muestra_primer_digito
rjmp fin_muestra_digitos
muestra_2:
rcall muestra_primer_digito
ldi r20,48
add r20,r26
rcall envia_dato_display
rjmp fin_muestra_digitos
muestra_3:
rcall muestra_primer_digito
ldi r20,48
add r20,r26
rcall envia_dato_display
ldi r20,48
add r20,r27
rcall envia_dato_display
```

```
rjmp fin_muestra_digitos
;en esta etapa se envia le mensaje de voltaje
fin_muestra_digitos:
rcall checkbf
ldi dato,' '
rcall writedr
rcall checkbf
ldi dato,223
rcall writedr
rcall checkbf
ldi dato,'c'
rcall writedr

rcall ins_cambia_linea
ldi r22,1 ;r22 es el registro q deshabilita las demas teclas, dejando solo menu
;activo

;en esta etapa se configura los 2 DACs
;se tiene q cargar las tablas especificas de la temperatura ingresada
cpi r28,1
breq tabla_pos
cpi r28,2
breq tabla_neg

tabla_pos:
rcall detecta_posit   ;r1=most significant byte,r2=byte intermedio y r3=least
;significant byte
rjmp envia_dac
tabla_neg:
rcall detecta_negat
rjmp envia_dac

; en esta rutina del programa se valida si la temperatura ingresada es la
correcta
valida_temperatura:
cpi r28,2
breq  valida_negativo
cpi r28,1
breq  valida_positivo
valida_positivo:
cpi r21,3
brsh valida_numero3_digitos      ; si se tiene un numero menor de 3 digitos es
una  temperatura correcta
rjmp temperatura_correcta

valida_numero3_digitos:
```

```
cpi r25,3   ; si se da esta condicion es por que la temperatura ingresada
brsh temperatura_incorrecta   ; seria de 3xx en adelante por lo cual es un valor
fuera del rango
cpi r25,2
breq valida_menor250  ; si es del orden de 2xx se verifica que no sobrepase a
250 grados
rjmp temperatura_correcta
valida_menor250:
cpi r26,6
brsh temperatura_incorrecta
cpi r26,5
breq valida_menor50
rjmp temperatura_correcta
valida_menor50:
cpi r27,1
brsh temperatura_incorrecta
rjmp temperatura_correcta

valida_negativo:
cpi r29,31
brsh temperatura_incorrecta
brlo temperatura_correcta
temperatura_correcta:
rjmp inicio_proceso_temperatura
temperatura_incorrecta:
rjmp  restart_msj

; subrutina para la envío de códigos al DAC variable
envia_dac:
rcall begin_comuni_dac1 ; se envían los códigos hacia el DAC variable
rcall hallar_resta_dacs ; se halla la resta de los DACs y se guarda en memoria
ldi zh,high(mensaje3*2) ;z apunta al inicio del mensaje voltaje,a partir de aqui
solo se calcula el voltaje mostrado en el lcd
ldi zl,low(mensaje3*2)     ;tambien se debe guardar el dato de voltaje para
;efectuar la resta o suma posterior
ldi r16,0
leer_digito_v:
rcall checkbf
lpm dato,z+ ;lee y muestra caracter en lcd
rcall writedr
inc r16
cpi r16,11 ;muestra del texto de solo 8 caracteres
brne leer_digito_v
;en esta parte del programa, se cargan las tablas de voltajes
rcall detecta_posit_negat ; se halla el valor de voltaje a mostrar en el lcd, y se
;guarda en ram
```

```
rcall leer_adc ; con esta subrutina se simula el código enviado del ADC
rcall realizar_correccion ; se halla el nuevo código del DAC variable
rcall corregir_dac_variable ; se manda dicho código hallado al DAC variable
rjmp reiniciar ; volver a sensar el teclado para el ingreso de un nuevo valor de
temperatura

hallar_resta_dacs:;se halla la resta ideal de los dacs para compararla
;con el adc
push r16
push r17
push xh
push xl
ldi r16,$00
ldi r17,$00
ldi xh,high(dacvariable)
ldi xl,low(dacvariable)
ld r1,x+
ld r2,x+
ld r3,x
ldi xh,high(dacfijo)
ldi xl,low(dacfijo)
ld r4,x+
ld r5,x+
ld r6,x

rcall restar_numero_24bit

ldi xh,high(dac_resta)
ldi xl,low(dac_resta)
st x+,r1
st x+,r2
st x,r3
pop xl
pop xh
pop r17
pop r16
ret

realizar_correccion:
push r16
push r17
push r18
push r19
push xh
push xl
ldi xh,high(dac_resta)
```

```
ldi xl,low(dac_resta)
ld r16,x+
ld r17,x+
ld r18,x
ldi xh,high(adccodigo)
ldi xl,low(adccodigo)
ld r19,x+
cp r16,r19
breq cont_analisis;si son iguales continua analisis
brsh dac_mayor;si es mayor, entonces la resta de los DACs es mayor
rjmp adc_mayor
cont_analisis:
ld r19,x+
cp r17,r19
breq cont_analisis2
brsh dac_mayor
rjmp adc_mayor
cont_analisis2:
ld r19,x
cp r18,r19
breq son_iguales
brsh dac_mayor
rjmp adc_mayor
son_iguales:
ldi r23,$00
rjmp no_hay_error
dac_mayor:;se debe sumar el codigo del DAC variable
ldi r16,$00
mov r15,r16
rcall hallar_diferencia;nos devuelve en r23 la diferencia entre los conversores
rjmp paso_siguiente
adc_mayor:;se debe restar el codigo del DAC variable
ldi r16,$01
mov r15,r16
rcall hallar_diferencia;nos devuelve en r23 la diferencia entre los conversores
paso_siguiente:
;se corrige el voltaje del lcd primero y luego el DAC variable, pero sin alterar lo
mostrado en el lcd
mov r15,r16;guardamos en r15 el valor de r16, para saber quien es mayor la
resta de los DACs o el ADC
cpi r29,181;si es entre 0 y 180, utiliza la segunda opcion
brlo segunda_opcion
rcall corregir_voltaje_lcd_temp_altas
rjmp exit_dif
segunda_opcion:
rcall fix_lcd_voltage
```

```
rjmp exit_dif
no_hay_error:
rcall enviar_msj_volt_real
ldi xh,high(voltaje_temp)
ldi xl,low(voltaje_temp)
lazo_no:
rcall checkbf
ld dato,x+      ;lee y muestra caracter en LCD
cpi r17,0
breq exit_dif
rcall writedr
rjmp    lazo_no
exit_dif:
pop xl
pop xh
pop r19
pop r18
pop r17
pop r16
ret

; subrutina que halla la diferencia entre la resta de los DACs y el ADC
hallar_diferencia:
push r16
push r17
push xh
push xl

cpi r16,$00; si es '0' entonces resta de DACs menos ADC
breq dac_menos_adc
;si es '1' ADC menos resta de DACs
adc_menos_dac:
ldi r16,$00
ldi r17,$00
ldi xh,high(adccodigo)
ldi xl,low(adccodigo)
ld r1,x+
ld r2,x+
ld r3,x
ldi xh,high(dac_resta)
ldi xl,low(dac_resta)
ld r4,x+
ld r5,x+
ld r6,x
rjmp restar
dac_menos_adc:
```

```
ldi r16,$00
ldi r17,$00
ldi xh,high(dac_resta)
ldi xl,low(dac_resta)
ld r1,x+
ld r2,x+
ld r3,x
ldi xh,high(adccodigo)
ldi xl,low(adccodigo)
ld r4,x+
ld r5,x+
ld r6,x
restar:
rcall restar_numero_24bit

mov r23,r3;en R23 guardamos la diferencia entre la resta de los DACs y el ADC
mov r14,r23
pop xl
pop xh
pop r17
pop r16
ret

;subrutina que resta dos numeros de 3 bytes en r1:r2:r3 - r4:r5:r6
restar_numero_24bit:
push r16
push r17
clr r16
clr r17
sub r3,r6
rol r16

sub r2,r16
rol r17
sub r1,r17
clr r17

sub r2,r5
rol r17

sub r1,r17
sub r1,r4
pop r17
pop r16
ret
```

```
;subrutina para la correccion del codigo del DAC variable segun el codigo del
;ADC, es decir si dicho codigo es mayor o menor que la diferencia de los dacs
corregir_dac_variable:
push r16
push r17
push r18
push xh
push xl
ldi xh,high(dacvariable);no se modifica el voltaje de la tabla, osea el ideal
ldi xl,low(dacvariable)
ld r1,x+
ld r2,x+
ld r3,x
ldi r17,$00
ldi r18,$00
mov r16,r15
mov r23,r14
cpi r16,$01;ADC mayor que resta de los DACs
breq restar_dac

sumar_dac:
add r3,r23
adc r2,r17
clr r17
adc r1,r17
rjmp finish

restar_dac:
sub r3,r23
rol r17
sub r2,r17
rol r18
sub r1,r18

finish:
rcall arregla_reg
rcall begin_comuni_dac1
pop xl
pop xh
pop r18
pop r17
pop r16
ret
;utilizas el puntero Y (r28,r29)
corregir_voltaje_lcd_temp_altas:
push r16
```

```
push r17
push r18
push r19
push r20
push xh
push xl
push yh
push yl
clr r1
clr r2
clr r3
clr r17
clr r18
clr r19
mov r20,r16;para saber si hay que restar o sumar el voltaje
ldi xh,high(voltaje_temp)
ldi xl,low(voltaje_temp)
ldi yh,high(digitos_finales)
ldi yl,low(digitos_finales)
ld r16,x+
cpi r16,'-'
breq es_negativo
inc r17;cuando es positivo
bucle_corr:
ldi r18,$00;si r18 es 0, el voltaje es positivo
ld r16,x+
cpi r16,'.'
breq cuenta_3mas
inc r17;contador de digitos antes del punto
rjmp bucle_corr

es_negativo:
ldi r18,$01;si r18 es 1, el voltaje es negativo
ld r16,x+
cpi r16,'.'
breq cuenta_3mas
inc r17;contador de digitos antes del punto
rjmp es_negativo

cuenta_3mas:
ld r16,x+
st y+,r16
inc r19
cpi r19,$03
brne cuenta_3mas
clr r16;aca se colocan los 4 codigos ascii despues del '.', incluido el cero
```

```
st y,r16
ldi yh,high(digitos_finales)
ldi yl,low(digitos_finales)
ld r16,y+;cargas el primer dígito despues del punto
subi r16,$30
ldi r19,10
mul r16,r19
mov r2,r0;tenemos el primer digito multiplicado por 10
ldi r19,100
mul r2,r19
;aca tenemos en r1 y r0 el numero de 4 digitos como 5000,3000,6000
mov r10,r0
mov r11,r1
ldi r19,100
ld r16,y+
subi r16,$30
mul r16,r19
add r10,r0
adc r11,r1
ldi r19,10
ld r16,y+
subi r16,$30
mul r16,r19
add r10,r0
adc r11,r1
ld r16,y;no le restamos $30 porque es $00, no $30
add r10,r16
clr r19
adc r11,r19
;en r11:r10 tenemos el valor de 5241,6542,3321
ldi r19,48;esto es el lsb de 4.8 uv, vamos a restar o sumar múltiplos de este
valor
clr r16
cpi r18,$01
brne voltaje_es_positivo

voltaje_es_negativo:
cpi r20,$00;si r20 es $00, se resta pero en negativo se suma
breq lazo_suma
rjmp lazo_resta

voltaje_es_positivo:
cpi r20,$00;si r20 es $00 se resta
breq lazo_resta
lazo_suma:
add r10,r19
```

```
adc r11,r16
dec r23
cpi r23,$00
breq finalizo
rjmp lazo_suma

lazo_resta:
clr r16
sub r10,r19
rol r16
sub r11,r16
dec r23
cpi r23,$00
breq finalizo
rjmp lazo_resta

finalizo:
;ya tengo en r11:r10 el nuevo numero despues del '.', despues de restarle o sumarle los multiplos de 4.8 uv
rcall hallar_nuevos_digitos
ldi xh,high(voltaje_temp)
ldi xl,low(voltaje_temp)
rcall mostrar_voltaje_real
pop yl
pop yh
pop xl
pop xh
pop r20
pop r19
pop r18
pop r17
pop r16
ret

;se utiliza el puntero Y, es decir r28,r29
hallar_nuevos_digitos:
push r16
push r17
push r18
push r19
push r21
push yh
push yl
clr r12
clr r13
ldi r21,$00;contador de ayuda
```

```
ldi yh,high(digitos_finales)
ldi yl,low(digitos_finales)
mov r16,r11
cpi r16,$03;comparacion menos o mas de mil
brlo es_menor_a_1000
breq comparar
cpi r16,$04
brsh es_mayor_a_1000

comparar:
mov r16,r10
cpi r16,$e8
brsh es_mayor_a_1000
rjmp es_menor_a_1000
es_mayor_a_1000:
ldi r17,$01
mov r1,r11;guardamos el número original
mov r0,r10
ldi r19,$03
ldi r18,$00
rjmp digits

es_menor_a_1000:
ldi r17,$02
ldi r19,$03
ldi r21,$01
ldi r18,$00

digits:
ldi yh,high(digitos_finales)
ldi yl,low(digitos_finales)
rcall bucle_division_10
add yl,r19
adc yh,r18
st y,r10
mov r11,r13
mov r10,r12
dec r19
inc r21
cpi r21,$03
breq se_tienen_los_digitos
rjmp digits


se_tienen_los_digitos:
ldi yh,high(digitos_finales)
```

```
ldi yl,low(digitos_finales)
cpi r17,$02
breq aumenta_un_cero
rjmp innecesario
aumenta_un_cero:
ldi r16,$00
st y+,r16
st y,r10
rjmp backdoor
innecesario:
st y,r10
backdoor:
pop yl
pop yh
pop r21
pop r19
pop r18
pop r17
pop r16
ret

mostrar_voltaje_real:
push r16
push r17
push r19
push r20
push xh
push xl
rcall enviar_msj_volt_real
cpi r18,$01;el voltaje es negativo
breq valor_negativo
inc r17
mov r19,r17
bucle_voltaje:
rcall checkbf
ld r17,x+
rcall writedr
dec r19
cpi r19,$00
breq agrega_digitos
rjmp bucle_voltaje

valor_negativo:
inc r17
inc r17
mov r19,r17
```

```
bucle_voltaje2:
rcall checkbf
ld r17,x+
rcall writedr
dec r19
cpi r19,$00
breq agrega_digitos
rjmp bucle_voltaje2

agrega_digitos:
ldi yh,high(digitos_finales)
ldi yl,low(digitos_finales)
ldi r19,$04

lazo_final:
rcall checkbf
ld r17,y+
ldi r20,$30
add r17,r20;para hallar su codigo ascii
rcall writedr
dec r19
cpi r19,$00
brne lazo_final

cpi r18,$01
breq v_negativo
ldi r20,$05
rjmp keep
v_negativo:
ldi r20,$06

keep:
ldi xh,high(voltaje_temp)
ldi xl,low(voltaje_temp)
add xl,r20
clr r20
adc xh,r20
ldi r20,$03
ultimos_caracteres:;espacio,m,v (últimos caracteres)
rcall checkbf
ld r17,x+
rcall writedr
dec r20
cpi r20,$00
brne ultimos_caracteres
pop xl
```

```
pop xh
pop r20
pop r19
pop r17
pop r16
ret

fix_lcd_voltage:
push   r16
push   r17
push   r18
push   r19
push   r20
push   xh
push   xl
push   yh
push   yl
clr    r1
clr    r2
clr    r3
clr    r17
clr    r18
clr    r19;contador para los 3 digitos despues del punto
mov    r20,r16;para saber si hay que restar o sumar el voltaje
ldi    xh,high(voltaje_temp)
ldi    xl,low(voltaje_temp)
ldi    yh,high(temp_de16bits)
ldi    yl,low(temp_de16bits)
ld     r16,x+
cpi    r16,'-'
breq es_negativo_2
st     y+,r16
es_positivo_2:
ldi    r18,$00;si r18 es 0, el voltaje es positivo
ld     r16,x+
cpi    r16,'.'
breq   cuenta_3mas_2
st     y+,r16
rjmp   es_positivo_2

es_negativo_2:
ldi    r18,$01;si r18 es 1, el voltaje es negativo
ld     r16,x+
cpi    r16,'.'
breq cuenta_3mas_2
st     y+,r16
```

```
rjmp    es_negativo_2

cuenta_3mas_2:
ld      r16,x+
st      y+,r16
inc     r19
cpi     r19,$03
brne    cuenta_3mas_2

;se obtienen los dígitos
clr     r16
st      y,r16
ldi     yh,high(temp_de16bits)
ldi     yl,low(temp_de16bits)

ld      r16,y+ ; cargas el primer digito
subi    r16,$30
ldi     r19,10
mul     r16,r19

mov     r2,r0 ; tenemos el primer digito multiplicado por 10
ldi     r19,100
mul     r2,r19

; aca tenemos en r1 y r0 el número de 4 dígitos como 5000,3000,6000, falta
;multiplicar por 10 para obtener 50000,30000,60000
mov     r10,r0
mov     r11,r1
ldi     r19,$01
mov     r16,r0
mov     r17,r1
lazo_x10:
add     r10,r16
adc     r11,r17
inc     r19
cpi     r19,10
brne    lazo_x10 ; si no ha terminado continua con el lazo
;en esta parte tenemos en r11:r10 10000,20000,40000,50000

;por 1000
ld      r16,y+
subi    r16,$30
ldi     r19,10
mul     r16,r19
mov     r2,r0
ldi     r19,100
```

```
mul     r2,r19
add     r10,r0
adc     r11,r1

;por 100
ld      r16,y+
subi    r16,$30
ldi     r19,100
mul     r16,r19
add     r10,r0
adc     r11,r1

;por 10
ld      r16,y+
subi    r16,$30
ldi     r19,10
mul     r16,r19
add     r10,r0
adc     r11,r1

ld      r16,y ; no le restamos $30 por que es un $00 no un $30
add     r10,r16
clr     r19
adc     r11,r19
;en     r11:r10 ; acá tenemos valores como 52410,65420,33210
ldi     r19,48 ; esto es el lsb de 4.8 uv, vamos a restar o sumar multiplos de
este ;valor
clr     r16

cpi     r18,$01
brne    voltaje_es_positivo_2

voltaje_es_negativo_2:
cpi     r20,$00 ; si r20 es $00, se resta pero en negativo se suma
breq    lazo_suma_2
rjmp    lazo_resta_2

voltaje_es_positivo_2:
cpi     r20,$00 ; si r20 es $00 se resta
breq    lazo_resta_2

lazo_suma_2:
add     r10,r19
adc     r11,r16
dec     r23
cpi     r23,$00
```

```
breq    finalizo_2
rjmp    lazo_suma_2

lazo_resta_2:
clr     r16
sub     r10,r19
rol     r16
sub     r11,r16
dec     r23
cpi     r23,$00
breq    finalizo_2
rjmp    lazo_resta_2

finalizo_2:
; ya se tiene en r11:r10 el nuevo número de 5 digitos, despues de restarle o
;sumarle los multiplos de 4.8 uv
rcall   hallar_nuevo_voltaje ; de 5 cifras significativas
ldi     xh,high(voltaje_temp)
ldi     xl,low(voltaje_temp)
rcall   mostrar_voltaje_real_2
pop     yl
pop     yh
pop     xl
pop     xh
pop     r20
pop     r19
pop     r18
pop     r17
pop     r16
ret

hallar_nuevo_voltaje:
push    r16
push    r17
push    r18
push    r19
push    r21
push    yh
push    yl
clr     r12
clr     r13
ldi     r21,$00 ; contador de ayuda
ldi     r18,$00
ldi     r19,$04
digits_2:
ldi     yh,high(temp_de16bits)
```

```
ldi     yl,low(temp_de16bits)
rcall   bucle_division_10
inc     r21
add     yl,r19
adc     yh,r18
st      y,r10 ; este es el residuo hallado
mov     r16,r12
cpi     r16,10
brlo    confirma_ultimo_valor
continua_process:
mov     r11,r13
mov     r10,r12
dec     r19
breq    se_tienen_los_digitos_2
rjmp    digits_2

confirma_ultimo_valor:
mov     r16,r13
cpi     r16,$00
brne    continua_process
dec     r19
ldi     yh,high(temp_de16bits)
ldi     yl,low(temp_de16bits)
add     yl,r19
adc     yh,r18
st      y,r12
se_tienen_los_digitos_2:
inc r21 ; en esta parte obtenemos la cantidad de dígitos del número hallado con
;R21

ldi     yh,high(temp_de16bits)
ldi     yl,low(temp_de16bits)

ldi     r16,$05
sub     r16,r21
mov     r21,r16
lazo_ceros:
ldi     r16,$00
cpi     r21,$00
breq    ceros_terminado
st      y+,r16
dec     r21
rjmp    lazo_ceros

ceros_terminado:
pop     yl
```

```
pop     yh
pop     r21
pop     r19
pop     r18
pop     r17
pop     r16
ret

; recibes el numero a ser dividido entre 10 en r11:r10, y en r10 obtienes el último ;dígito
bucle_division_10:
push    r16
push    r17
push    r18
push    r19
clr     r12
clr     r13
seguir_restando:
clr     r16
ldi     r19,10
sub     r10,r19
rol     r16
sub     r11,r16
cp      r10,r19
brlo    revisar_fin
ldi     r18,$01
clr     r17
add     r12,r18 ; cuenta para el siguiente numero de 3 cifras, luego de 2 cifras
adc     r13,r17
rjmp seguir_restando

revisar_fin:
clr     r17
cp      r11,r17 ; confirma que r11 esta en $00, con lo cual tenemos un número de ;8 bits en r11:r10
breq exit_bucle
ldi     r18,$01
clr     r17
add     r12,r18 ; cuenta para el siguiente número de 3 cifras, luego de 2 cifras
adc     r13,r17
rjmp    seguir_restando
exit_bucle:
ldi     r18,$01
clr     r17
add     r12,r18 ; cuenta para el siguiente numero de 3 cifras
adc     r13,r17
```

```
pop     r19
pop     r18
pop     r17
pop     r16
ret

enviar_msj_volt_real:
push    r16
push    r17
rcall   ins_cambia_4talinea
ldi     zh,high(mensaje4*2) ; z apunta al inicio del mensaje voltaje,a partir de
;aquí sólo se calcula el voltaje mostrado en el LCD
ldi     zl,low(mensaje4*2) ; también se debe guardar el dato de voltaje para
;efectuar la resta o suma posterior
ldi     r16,0
leer_digit:
rcall   checkbf
lpm     dato,z+ ; lee y muestra caracter en LCD
rcall   writedr
inc     r16
cpi     r16,10 ; se muestra el texto de solo 10 caracteres
brne    leer_digit
pop     r17
pop     r16
ret

mostrar_voltaje_real_2:
push    r16
push    r17
push    r19
push    r20
push    r21
push    yh
push    yl
ldi     yh,high(temp_de16bits)
ldi     yl,low(temp_de16bits)
rcall   enviar_msj_volt_real
ldi     r21,$00
ldi     r19,$00
cpi     r18,$01;el voltaje es negativo
breq    valor_negativo_2

bucle_voltaje_2:
cpi     r19,$01
breq    agrega_punto
return:
```

```
rcall   checkbf
ld      r17,y+
ldi     r20,$30
add     r17,r20;para hallar su codigo ascii
rcall   writedr
inc     r19
cpi     r19,$05
breq    caracteres_finales
rjmp    bucle_voltaje_2

agrega_punto:
rcall   checkbf
ldi     r17,'.'
rcall   writedr
rjmp    return

valor_negativo_2:
rcall   checkbf
ldi     r17,'-'
rcall   writedr

bucle_voltaje_2_2:
cpi     r19,$01
breq    agrega_punto_2
return_2:
rcall   checkbf
ld      r17,y+
ldi     r20,$30
add     r17,r20 ; para hallar su código ascii
rcall   writedr
inc     r19
cpi     r19,$05
breq    caracteres_finales
rjmp    bucle_voltaje_2_2

agrega_punto_2:
rcall   checkbf
ldi     r17,'.'
rcall   writedr
rjmp    return_2

caracteres_finales: ; espacio,m,v ultimos caracteres
rcall   checkbf
ldi     r17,' '
rcall   writedr
rcall   checkbf
```

```
ldi     r17,'m'
rcall   writedr
rcall   checkbf
ldi     r17,'v'
rcall   writedr

pop     yl
pop     yh
pop     r21
pop     r20
pop     r19
pop     r17
pop     r16
ret

arregla_reg:
push    r16
push    r17
push    r18
push    r19
push    r21
push    r22
push    r23
; esta rutina arregla los codigos antes de ser mandados cualquiera de los DACs
mov     r19,r3
mov     r23,r3
swap    r23
andi    r23,$f0
mov     r18,r2
mov     r22,r2
swap    r19
andi    r19,$0f
swap    r22
andi    r22,$f0
add     r22,r19
swap    r18
andi    r18,$0f
mov     r21,r1
swap    r21
andi    r21,$f0
add     r21,r18

mov     r3,r23 ; nuevos valores reubicados
mov     r2,r22
mov     r1,r21
```

```
pop     r23
pop     r22
pop     r21
pop     r19
pop     r18
pop     r17
pop     r16

ret

;codigos correspondiente a valores positivos de temperatura
detecta_posit:
push    r16
push    r17
push    xh
push    xl
ldi     zh,high(tablaposit*2)
ldi     zl,low(tablaposit*2)
ldi     r18,0
mov     r16,r29 ; valor de temperatura en positivo
add     r16,r16
adc     zh,r18
add     zl,r16
adc     zh,r18
lpm     r2,z+
lpm     r3,z
cpi     r29,26
brlo es_tres
ldi     r16,$04
rjmp    coloca_valor_r1
es_tres:
ldi     r16,$03
coloca_valor_r1:
clr     r1
add     r1,r16
ldi     xh,high(dacvariable) ; cargamos el inicio de ram reservado para el DAC
;variable
ldi     xl,low(dacvariable)
st      x+,r1 ; guardamos el codigo del DAC variable
st      x+,r2
st      x,r3
rcall   arregla_reg ; se arreglan los registros para poder ser enviados al DAC
;variable
pop     xl
pop     xh
pop     r17
```

```
pop     r16
ret

; codigos correspondiente a valores negativos de temperatura
detecta_negat:
push    r16
push    r17
push    xh
push    xl
ldi     zh,high(tablanegat*2)
ldi     zl,low(tablanegat*2)
ldi     r18,0
mov     r16,r29 ; valor de temperatura en negativo
dec     r16 ; aca notamos la diferencia con el caso del valor positivo
add     r16,r16
add     zl,r16
adc     zh,r18
lpm     r2,z+
lpm     r3,z
ldi     r16,$03
clr     r1
add     r1,r16
ldi     xh,high(dacvariable) ; cargamos el inicio de ram reservado para el DAC
;variable
ldi     xl,low(dacvariable)
st      x+,r1 ; guardamos el codigo del DAC variable
st      x+,r2
st      x,r3
rcall   arregla_reg ; se arreglan los registros para poder ser enviados al DAC
;variable
pop     xl
pop     xh
pop     r17
pop     r16
ret

detecta_neutro:
push    r16
push    r17
push    xh
push    xl
ldi     zh,high(tablaneutro*2)
ldi     zl,low(tablaneutro*2)
ldi     r18,0
lpm     r2,z+
lpm     r3,z
```

```
ldi     r16,$04
clr     r1
add     r1,r16
ldi     xh,high(dacfijo) ; cargamos el inicio de ram reservado para el DAC fijo
ldi     xl,low(dacfijo)
st      x+,r1 ; guardamos los valores correspondientes al DAC fijo
st      x+,r2
st      x,r3
rcall   arregla_reg ; se arreglan los registros para poder ser enviados al DAC
fijo
pop     xl
pop     xh
pop     r17
pop     r16
ret

;-----------------------------------------------incremento/decremento-----------------------
-
;-----------------------------------------incremento-----------------------------------------
-
; función que incrementa en 10 la temperatura ingresada en el display
incrementa_10:
cpi     r21, 1 ; se verifica si es de un dígito
breq    aumenta_unidades
cpi     r21, 2 ; se verifica si es de dos dígitos
breq    aumenta_decenas
cpi      r21, 3 ; se verifica si es de tres dígitos
breq    aumenta_centenas
rjmp    reiniciar

aumenta_unidades:
rcall   ins_mov_izquierda
cpi     r25,45
breq    numero_invalido_inc ; se verifica si el número en el display es valido
rcall   mostrar_1 ; se agrega 1 por que se aumenta una decena (1 en ASCII)
ldi     r20, 48 ; se escribe el dígito que ya habia sido ingresado anteriormente
mov     r26,r25
add     r20,r25
rcall   envia_dato_display
ldi     r25,$01
inc     r21
rcall   retardo5xms
rjmp    reiniciar

numero_invalido_inc:
rcall   ins_mov_derecha
```

```
rcall   retardo5xms
rjmp    reiniciar

aumenta_decenas:
rcall   ins_mov_izquierda
cpi     r25, 45
breq    aumenta_negativo_dec
brne    aumenta_positivo_dec

aumenta_positivo_dec:
rcall   ins_mov_izquierda
cpi     r25,$09 ; se comprueba si el dígito de las decenas es igual a 9
breq    inc_a_centenas1 ; cuando el aumento involucra a las centenas
inc     r25
ldi     r20, 48
add     r20,r25
rcall   envia_dato_display
rcall   ins_mov_derecha
rcall   retardo5xms
rjmp    reiniciar

inc_a_centenas1:
rcall   mostrar_1
rcall   mostrar_0
ldi     r20, 48
add     r20,r26
rcall   envia_dato_display
mov     r27, r26
ldi     r26,$00
ldi     r25,$01
rcall   retardo5xms
inc     r21
rjmp    reiniciar

aumenta_negativo_dec:
rcall   ins_mov_izquierda
ldi     r20, $0a
sub     r20,r26
mov     r26,r20
ldi     r20, 48 ; se escribe el dígito menos 10
add     r20,r26
rcall   envia_dato_display
rcall   mostrar_espacio
rcall   ins_mov_izquierda
dec     r21
mov     r25,r26
```

```
rcall   retardo5xms
rjmp    reiniciar

; subrutina que se utiliza para aumentar en 10 el número, cuando este tiene 3
;dígitos
aumenta_centenas:
rcall   ins_mov_izquierda
cpi     r25, 45
breq    aumenta_negativo_cen
brne    aumenta_positivo_cen

aumenta_positivo_cen:
rcall   ins_mov_izquierda
cpi     r26,$09 ; se comprueba si las decenas son guales a 9
breq    inc_a_centenas2 ; cuando es 9 al aumentar si pasara a las centenas que
;se da en tres digitos
inc     r26
ldi     r20, 48
add     r20,r26
rcall   envia_dato_display
rcall   ins_mov_derecha
rcall   retardo5xms
rjmp    reiniciar

inc_a_centenas2: ; esta subrutina opera cuando se tiene valores del tipo x9x
rcall   ins_mov_izquierda
cpi     r25,$09
breq    aumenta_millar
ldi     r20, 48
add     r20,r25
rcall   envia_dato_display
rcall   mostrar_0
ldi     r20, 48
add     r20,r27
rcall   envia_dato_display
ldi     r26,$00
rcall   retardo5xms
rjmp    reiniciar
aumenta_millar: ; esta función se ejecuta en caso se tenga valores de 99x
rcall   mostrar_1
rcall   mostrar_0
rcall   mostrar_0
ldi     r20, 48
add     r20,r27
rcall   envia_dato_display
ldi     r25,$01
```

```
ldi     r26,$00
ldi     r27,$00
inc     r21
rcall   retardo5xms
rjmp    verificacion_digitos ; se manda la verificacion de digitos debido a que son demasiados

aumenta_negativo_cen:
cpi     r26,$01
breq    aumenta_neg
rcall   ins_mov_izquierda
dec     r26
ldi     r20, 48
add     r20,r26
rcall   envia_dato_display
rcall   ins_mov_derecha
rcall   retardo5xms
rjmp    reiniciar

aumenta_neg:
cpi     r27,0
breq    poner_cero
rcall   ins_mov_izquierda
ldi     r20, 48
add     r20,r27
rcall   envia_dato_display
rcall   mostrar_espacio
rcall   ins_mov_izquierda
dec     r21
mov     r26,r27
rcall   retardo5xms
rjmp    reiniciar

poner_cero:
rcall   ins_mov_izquierda
rcall   ins_mov_izquierda
rcall   mostrar_0
rcall   mostrar_espacio
rcall   mostrar_espacio
rcall   ins_mov_izquierda
rcall   ins_mov_izquierda
dec     r21
dec     r21
ldi     r25,0
rcall   retardo5xms
rjmp    reiniciar
```

```
;-----------------------------------decremento-----------------------------------------------------
--
; funcion que decrementa en 10 unidades la temperatura en display
decrementa_10:
cpi     r21, 1 ; se verifica si es de un digito
breq    disminuye_unidades
cpi     r21, 2 ; se verifica si es de dos digitos
breq    disminuye_decenas
cpi     r21, 3 ; se verifica si es de tres digitos
breq    disminuye_centenas
rjmp    reiniciar

disminuye_unidades:
rcall   ins_mov_izquierda
cpi     r25,45 ; se comprueba que el número sea válido (-), es decir dentro del
;rango
breq    numero_invalido_dec
rcall   mostrar_menos
cpi     r25,0
breq    restar_cero ; se comprueba si el número es cero
ldi     r20, $0a
sub     r20,r25
mov     r26,r20
ldi     r20, 48 ; se visualiza el digito menos 10 unidades
add     r20,r26
rcall   envia_dato_display
ldi     r25,$2d
inc     r21
rcall   retardo5xms
rjmp    reiniciar

restar_cero:
rcall   mostrar_1
rcall   mostrar_0
inc     r21
inc     r21
ldi     r25,$2d
ldi     r26,1
ldi     r27,0
rcall   retardo5xms
rjmp    reiniciar

numero_invalido_dec:
rcall   ins_mov_derecha
rcall   retardo5xms
rjmp    reiniciar
```

```
disminuye_decenas:
rcall   ins_mov_izquierda
cpi     r25, 45
breq    disminuye_negativo_dec
brne    disminuye_positivo_dec
disminuye_positivo_dec:
rcall   ins_mov_izquierda
cpi     r25,1 ; se comprueba si el digito de las decenas es igual a 1
breq dec_a_unidades1; salta a la rutina de decrementa a unidades cuando se
; tiene en 2 digitos
dec     r25
ldi     r20, 48
add     r20,r25
rcall   envia_dato_display
rcall   ins_mov_derecha
rcall   retardo5xms
rjmp    reiniciar

dec_a_unidades1:
ldi     r20, 48
add     r20,r26
rcall   envia_dato_display
rcall   mostrar_espacio
rcall   ins_mov_izquierda
mov     r25,r26
dec     r21
rcall   retardo5xms
rjmp    reiniciar

; esta parte del programa resta diez unidades a números negativos
disminuye_negativo_dec:
rcall   mostrar_1
ldi     r20, 48 ; se escribe el valor negativo
add     r20,r26
rcall   envia_dato_display
mov     r27,r26
ldi     r26,$01
inc     r21
rcall   retardo5xms
rjmp    reiniciar ; reiniciar si se tiene 3 digitos
disminuye_centenas:
rcall   ins_mov_izquierda
cpi     r25, 45
breq    disminuye_negativo_cen
brne    disminuye_positivo_cen
```

```
disminuye_positivo_cen:
rcall   ins_mov_izquierda
cpi     r26,0 ; se comprueba si las decenas son guales a 9
breq    dec_a_unidades2 ; se salta a la rutina de decrementa a unidades
cuando se tiene en 2 digitos
dec     r26
ldi     r20, 48
add     r20,r26
rcall   envia_dato_display
rcall   ins_mov_derecha
rcall   retardo5xms
rjmp    reiniciar

dec_a_unidades2:
rcall   ins_mov_izquierda
dec     r25
cpi     r25,0
breq    reduce_a_decenas
ldi     r20, 48
add     r20,r25
rcall   envia_dato_display
rcall   mostrar_9
ldi     r20, 48
add     r20,r27
rcall   envia_dato_display
ldi     r26, $09
rcall   retardo5xms
rjmp    reiniciar

; el programa salta aqui cuando se reduce a decenas
reduce_a_decenas:
rcall   mostrar_9
ldi     r20, 48
add     r20,r27
rcall   envia_dato_display
rcall   mostrar_espacio
rcall   ins_mov_izquierda
ldi     r25, $09
mov     r26,r27
dec     r21
rcall   retardo5xms
rjmp    reiniciar

disminuye_negativo_cen:
rcall   ins_mov_izquierda
cpi     r26,9 ; se comprueba se las unidades de las decenas son guales a 9
```

```
breq    dis_decenas
inc     r26
ldi     r20, 48
add     r20,r26
rcall   envia_dato_display
rcall   ins_mov_derecha
rcall   retardo5xms
rjmp    reiniciar

dis_decenas:
rcall   mostrar_1
rcall   mostrar_0
ldi     r20, 48
add     r20,r27
rcall   envia_dato_display ; el dato se envía en el registro R20
ldi     r26,$01
ldi     r27,$00
inc     r21
rcall   retardo5xms
rjmp    verificacion_digitos
; subrutina para hallar los valores de voltaje mostrados en el LCD
detecta_posit_negat:
push    r16
push    r17
push    r28
cpi     r28,$01
breq    voltposit
cpi     r28,$02
breq    voltnegat
; voltajes para temperaturas positivas
voltposit:
cpi     r29, 0
brlo    voltposit
cpi     r29, 251
brsh    voltposit
ldi     zh, high(tabla_positiva_volt)
ldi     zl, low(tabla_positiva_volt)
clr     r17
add     zl, r29
adc     zh, r17 ; z = tabla+numero binario
add     zl, zl
adc     zh, zh ; z= z*2
lpm     xl, z+
lpm     xh, z ; x dirección de inicio del nombre
movw    zl, xl
rcall   tipea_lcd
```

```
rjmp    salida_tab_vol

; voltajes para temperaturas negativas
voltnegat:
cpi     r29, 1
brlo    voltnegat
cpi     r29, 31
brsh    voltnegat
ldi     zh, high(tabla_negativa_volt)
ldi     zl, low(tabla_negativa_volt)
clr     r17
dec     r29
add     zl, r29
adc     zh, r17 ; z = tabla+numero binario
add     zl, zl
adc     zh, zh ; z= z*2
lpm     xl, z+
lpm     xh, z ; x dirección de inicio del nombre
movw    zl, xl  ; aca el zh:zl está apuntando a la dirección de v0,v1,vn1,vn2, etc.
rcall   tipea_lcd
salida_tab_vol:
inc     r29
pop     r28
pop     r17
pop     r16
ret

; esta subrutina permite mostrar en el lcd, los caracteres de las tablas de
voltaje, por ejemplo: 2.234mv
tipea_lcd:
push    r16
push    r17
push    zl
push    zh
push    xh
push    xl
ldi     xh,high(voltaje_temp)
ldi     xl,low(voltaje_temp)
clc
add     zl, zl
adc     zh, zh
lcd_loop:
rcall   checkbf
lpm     dato,z+; lee y muestra caracter en lcd
st      x+,dato
cpi     r17,0
```

```
breq    salir
rcall   writedr
rjmp    lcd_loop
salir:
pop     xl
pop     xh
pop     zh
pop     zl
pop     r17
pop     r16
ret

; tabla de voltajes correspondiente a las temperaturas negativas
vn1:    .db     "-0.039 mv",0
vn2:    .db     "-0.079 mv",0
vn3:    .db     "-0.118 mv",0
vn4:    .db     "-0.157 mv",0
vn5:    .db     "-1.226 mv",0
vn6:    .db     "-0.236 mv",0
vn7:    .db     "-0.275 mv",0
vn8:    .db     "-0.314 mv",0
vn9:    .db     "-0.353 mv",0
vn10:   .db     "-1.429 mv",0
vn11:   .db     "-0.431 mv",0
vn12:   .db     "-0.469 mv",0
vn13:   .db     "-0.508 mv",0
vn14:   .db     "-0.547 mv",0
vn15:   .db     "-1.623 mv",0
vn16:   .db     "-0.624 mv",0
vn17:   .db     "-0.662 mv",0
vn18:   .db     "-0.701 mv",0
vn19:   .db     "-0.739 mv",0
vn20:   .db     "-1.817 mv",0
vn21:   .db     "-0.816 mv",0
vn22:   .db     "-0.854 mv",0
vn23:   .db     "-0.892 mv",0
vn24:   .db     "-0.930 mv",0
vn25:   .db     "-2.025 mv",0
vn26:   .db     "-1.005 mv",0
vn27:   .db     "-1.043 mv",0
vn28:   .db     "-1.081 mv",0
vn29:   .db     "-1.118 mv",0
vn30:   .db     "-2.226 mv",0

; tabla de voltajes correspondiente a las temperaturas positivas
v0:     .db     "-1.060 mv",0
```

```
v1:     .db     "0.039 mv",0,0
v2:     .db     "0.079 mv",0,0
v3:     .db     "0.119 mv",0,0
v4:     .db     "0.158 mv",0,0
v5:     .db     "-0.831 mv",0
v6:     .db     "0.238 mv",0,0
v7:     .db     "0.277 mv",0,0
v8:     .db     "0.317 mv",0,0
v9:     .db     "0.357 mv",0,0
v10:    .db     "-0.638 mv",0
v11:    .db     "0.437 mv",0,0
v12:    .db     "0.477 mv",0,0
v13:    .db     "0.517 mv",0,0
v14:    .db     "0.557 mv",0,0
v15:    .db     "-0.440 mv",0
v16:    .db     "0.637 mv",0,0
v17:    .db     "0.677 mv",0,0
v18:    .db     "0.718 mv",0,0
v19:    .db     "0.758 mv",0,0
v20:    .db     "-0.242 mv",0
v21:    .db     "0.838 mv",0,0
v22:    .db     "0.879 mv",0,0
v23:    .db     "0.919 mv",0,0
v24:    .db     "0.960 mv",0,0
v25:    .db     "-0.046 mv",0
v26:    .db     "1.041 mv",0,0
v27:    .db     "1.081 mv",0,0
v28:    .db     "1.122 mv",0,0
v29:    .db     "1.162 mv",0,0
v30:    .db     "0.150 mv",0,0
v31:    .db     "1.244 mv",0,0
v32:    .db     "1.285 mv",0,0
v33:    .db     "1.325 mv",0,0
v34:    .db     "1.366 mv",0,0
v35:    .db     "0.350 mv",0,0
v36:    .db     "1.448 mv",0,0
v37:    .db     "1.489 mv",0,0
v38:    .db     "1.529 mv",0,0
v39:    .db     "1.570 mv",0,0
v40:    .db     "0.551 mv",0,0
v41:    .db     "1.652 mv",0,0
v42:    .db     "1.693 mv",0,0
v43:    .db     "1.734 mv",0,0
v44:    .db     "1.776 mv",0,0
v45:    .db     "0.776 mv",0,0
v46:    .db     "1.858 mv",0,0
```

```
v47:    .db     "1.899 mv",0,0
v48:    .db     "1.940 mv",0,0
v49:    .db     "1.981 mv",0,0
v50:    .db     "0.992 mv",0,0
v51:    .db     "2.064 mv",0,0
v52:    .db     "2.105 mv",0,0
v53:    .db     "2.146 mv",0,0
v54:    .db     "2.188 mv",0,0
v55:    .db     "1.231 mv",0,0
v56:    .db     "2.270 mv",0,0
v57:    .db     "2.312 mv",0,0
v58:    .db     "2.353 mv",0,0
v59:    .db     "2.394 mv",0,0
v60:    .db     "1.461 mv",0,0
v61:    .db     "2.477 mv",0,0
v62:    .db     "2.519 mv",0,0
v63:    .db     "2.560 mv",0,0
v64:    .db     "2.601 mv",0,0
v65:    .db     "1.656 mv",0,0
v66:    .db     "2.684 mv",0,0
v67:    .db     "2.726 mv",0,0
v68:    .db     "2.767 mv",0,0
v69:    .db     "2.809 mv",0,0
v70:    .db     "1.851 mv",0,0
v71:    .db     "2.892 mv",0,0
v72:    .db     "2.933 mv",0,0
v73:    .db     "2.975 mv",0,0
v74:    .db     "3.016 mv",0,0
v75:    .db     "2.048 mv",0,0
v76:    .db     "3.100 mv",0,0
v77:    .db     "3.141 mv",0,0
v78:    .db     "3.183 mv",0,0
v79:    .db     "3.224 mv",0,0
v80:    .db     "2.246 mv",0,0
v81:    .db     "3.307 mv",0,0
v82:    .db     "3.349 mv",0,0
v83:    .db     "3.390 mv",0,0
v84:    .db     "3.432 mv",0,0
v85:    .db     "2.461 mv",0,0
v86:    .db     "3.515 mv",0,0
v87:    .db     "3.556 mv",0,0
v88:    .db     "3.598 mv",0,0
v89:    .db     "3.639 mv",0,0
v90:    .db     "2.675 mv",0,0
v91:    .db     "3.722 mv",0,0
v92:    .db     "3.764 mv",0,0
```

```
v93:  .db   "3.805 mv",0,0
v94:  .db   "3.847 mv",0,0
v95:  .db   "2.902 mv",0,0
v96:  .db   "3.930 mv",0,0
v97:  .db   "3.971 mv",0,0
v98:  .db   "4.012 mv",0,0
v99:  .db   "4.054 mv",0,0
v100: .db   "3.129 mv",0,0
v101: .db   "4.137 mv",0,0
v102: .db   "4.178 mv",0,0
v103: .db   "4.219 mv",0,0
v104: .db   "4.261 mv",0,0
v105: .db   "3.315 mv",0,0
v106: .db   "4.343 mv",0,0
v107: .db   "4.384 mv",0,0
v108: .db   "4.426 mv",0,0
v109: .db   "4.467 mv",0,0
v110: .db   "3.501 mv",0,0
v111: .db   "4.549 mv",0,0
v112: .db   "4.590 mv",0,0
v113: .db   "4.632 mv",0,0
v114: .db   "4.673 mv",0,0
v115: .db   "3.704 mv",0,0
v116: .db   "4.755 mv",0,0
v117: .db   "4.796 mv",0,0
v118: .db   "4.837 mv",0,0
v119: .db   "4.878 mv",0,0
v120: .db   "3.908 mv",0,0
v121: .db   "4.960 mv",0,0
v122: .db   "5.001 mv",0,0
v123: .db   "5.042 mv",0,0
v124: .db   "5.083 mv",0,0
v125: .db   "4.136 mv",0,0
v126: .db   "5.164 mv",0,0
v127: .db   "5.205 mv",0,0
v128: .db   "5.246 mv",0,0
v129: .db   "5.287 mv",0,0
v130: .db   "4.362 mv",0,0
v131: .db   "5.368 mv",0,0
v132: .db   "5.409 mv",0,0
v133: .db   "5.450 mv",0,0
v134: .db   "5.490 mv",0,0
v135: .db   "4.592 mv",0,0
v136: .db   "5.571 mv",0,0
v137: .db   "5.612 mv",0,0
v138: .db   "5.652 mv",0,0
```

```
v139:  .db    "5.693 mv",0,0
v140:  .db    "4.818 mv",0,0
v141:  .db    "5.774 mv",0,0
v142:  .db    "5.814 mv",0,0
v143:  .db    "5.855 mv",0,0
v144:  .db    "5.895 mv",0,0
v145:  .db    "5.022 mv",0,0
v146:  .db    "5.976 mv",0,0
v147:  .db    "6.016 mv",0,0
v148:  .db    "6.057 mv",0,0
v149:  .db    "6.097 mv",0,0
v150:  .db    "5.186 mv",0,0
v151:  .db    "6.177 mv",0,0
v152:  .db    "6.218 mv",0,0
v153:  .db    "6.258 mv",0,0
v154:  .db    "6.298 mv",0,0
v155:  .db    "5.392 mv",0,0
v156:  .db    "6.378 mv",0,0
v157:  .db    "6.419 mv",0,0
v158:  .db    "6.459 mv",0,0
v159:  .db    "6.499 mv",0,0
v160:  .db    "5.598 mv",0,0
v161:  .db    "6.579 mv",0,0
v162:  .db    "6.619 mv",0,0
v163:  .db    "6.659 mv",0,0
v164:  .db    "6.699 mv",0,0
v165:  .db    "5.804 mv",0,0
v166:  .db    "6.779 mv",0,0
v167:  .db    "6.819 mv",0,0
v168:  .db    "6.859 mv",0,0
v169:  .db    "6.899 mv",0,0
v170:  .db    "6.010 mv",0,0
v171:  .db    "6.979 mv",0,0
v172:  .db    "7.019 mv",0,0
v173:  .db    "7.059 mv",0,0
v174:  .db    "7.099 mv",0,0
v175:  .db    "6.216 mv",0,0
v176:  .db    "7.179 mv",0,0
v177:  .db    "7.219 mv",0,0
v178:  .db    "7.259 mv",0,0
v179:  .db    "7.299 mv",0,0
v180:  .db    "6.421 mv",0,0
v181:  .db    "7.378 mv",0,0
v182:  .db    "7.418 mv",0,0
v183:  .db    "7.458 mv",0,0
v184:  .db    "7.498 mv",0,0
```

```
v185:  .db    "6.632 mv",0,0
v186:  .db    "7.578 mv",0,0
v187:  .db    "7.618 mv",0,0
v188:  .db    "7.658 mv",0,0
v189:  .db    "7.697 mv",0,0
v190:  .db    "6.842 mv",0,0
v191:  .db    "7.777 mv",0,0
v192:  .db    "7.817 mv",0,0
v193:  .db    "7.857 mv",0,0
v194:  .db    "7.897 mv",0,0
v195:  .db    "7.048 mv",0,0
v196:  .db    "7.977 mv",0,0
v197:  .db    "8.017 mv",0,0
v198:  .db    "8.057 mv",0,0
v199:  .db    "8.097 mv",0,0
v200:  .db    "7.253 mv",0,0
v201:  .db    "8.177 mv",0,0
v202:  .db    "8.216 mv",0,0
v203:  .db    "8.256 mv",0,0
v204:  .db    "8.296 mv",0,0
v205:  .db    "7.459 mv",0,0
v206:  .db    "8.376 mv",0,0
v207:  .db    "8.416 mv",0,0
v208:  .db    "8.456 mv",0,0
v209:  .db    "8.497 mv",0,0
v210:  .db    "7.665 mv",0,0
v211:  .db    "8.577 mv",0,0
v212:  .db    "8.617 mv",0,0
v213:  .db    "8.657 mv",0,0
v214:  .db    "8.697 mv",0,0
v215:  .db    "7.871 mv",0,0
v216:  .db    "8.777 mv",0,0
v217:  .db    "8.817 mv",0,0
v218:  .db    "8.857 mv",0,0
v219:  .db    "8.898 mv",0,0
v220:  .db    "8.077 mv",0,0
v221:  .db    "8.978 mv",0,0
v222:  .db    "9.018 mv",0,0
v223:  .db    "9.058 mv",0,0
v224:  .db    "9.099 mv",0,0
v225:  .db    "8.292 mv",0,0
v226:  .db    "9.179 mv",0,0
v227:  .db    "9.220 mv",0,0
v228:  .db    "9.260 mv",0,0
v229:  .db    "9.300 mv",0,0
v230:  .db    "8.506 mv",0,0
```

```
v231:  .db     "9.381 mv",0,0
v232:  .db     "9.421 mv",0,0
v233:  .db     "9.462 mv",0,0
v234:  .db     "9.502 mv",0,0
v235:  .db     "8.713 mv",0,0
v236:  .db     "9.583 mv",0,0
v237:  .db     "9.624 mv",0,0
v238:  .db     "9.664 mv",0,0
v239:  .db     "9.705 mv",0,0
v240:  .db     "8.919 mv",0,0
v241:  .db     "9.786 mv",0,0
v242:  .db     "9.826 mv",0,0
v243:  .db     "9.867 mv",0,0
v244:  .db     "9.907 mv",0,0
v245:  .db     "9.134 mv",0,0
v246:  .db     "9.989 mv",0,0
v247:  .db     "10.029 mv",0
v248:  .db     "10.070 mv",0
v249:  .db     "10.111 mv",0
v250:  .db     "9.348 mv",0,0

tabla_positiva_volt:
.dw     v0, v1, v2, v3, v4
.dw     v5, v6, v7, v8, v9
.dw     v10, v11, v12, v13, v14
.dw     v15, v16, v17, v18, v19
.dw     v20, v21, v22, v23, v24
.dw     v25, v26, v27, v28, v29
.dw     v30, v31, v32, v33, v34
.dw     v35, v36, v37, v38, v39
.dw     v40, v41, v42, v43, v44
.dw     v45, v46, v47, v48, v49
.dw     v50, v51, v52, v53, v54
.dw     v55, v56, v57, v58, v59
.dw     v60, v61, v62, v63, v64
.dw     v65, v66, v67, v68, v69
.dw     v70, v71, v72, v73, v74
.dw     v75, v76, v77, v78, v79
.dw     v80, v81, v82, v83, v84
.dw     v85, v86, v87, v88, v89
.dw     v90, v91, v92, v93, v94
.dw     v95, v96, v97, v98, v99
.dw     v100, v101, v102, v103, v104
.dw     v105, v106, v107, v108, v109
.dw     v110, v111, v112, v113, v114
.dw     v115, v116, v117, v118, v119
```

```
.dw     v120, v121, v122, v123, v124
.dw     v125, v126, v127, v128, v129
.dw     v130, v131, v132, v133, v134
.dw     v135, v136, v137, v138, v139
.dw     v140, v141, v142, v143, v144
.dw     v145, v146, v147, v148, v149
.dw     v150, v151, v152, v153, v154
.dw     v155, v156, v157, v158, v159
.dw     v160, v161, v162, v163, v164
.dw     v165, v166, v167, v168, v169
.dw     v170, v171, v172, v173, v174
.dw     v175, v176, v177, v178, v179
.dw     v180, v181, v182, v183, v184
.dw     v185, v186, v187, v188, v189
.dw     v190, v191, v192, v193, v194
.dw     v195, v196, v197, v198, v199
.dw     v200, v201, v202, v203, v204
.dw     v205, v206, v207, v208, v209
.dw     v210, v211, v212, v213, v214
.dw     v215, v216, v217, v218, v219
.dw     v220, v221, v222, v223, v224
.dw     v225, v226, v227, v228, v229
.dw     v230, v231, v232, v233, v234
.dw     v235, v236, v237, v238, v239
.dw     v240, v241, v242, v243, v244
.dw     v245, v246, v247, v248, v249
.dw     v250

tabla_negativa_volt:
.dw     vn1, vn2, vn3, vn4
.dw     vn5, vn6, vn7, vn8, vn9
.dw     vn10, vn11, vn12, vn13, vn14
.dw     vn15, vn16, vn17, vn18, vn19
.dw     vn20, vn21, vn22, vn23, vn24
.dw     vn25, vn26, vn27, vn28, vn29
.dw     vn30

; esta tabla contiene los códigos a ser enviados al dac variable para las
temperaturas negativas
tablanegat:
;db msb,lsb
.db $03,$dc;-1
.db $03,$d3;-2
.db $03,$ca;-3
.db $03,$c2;-4
.db $fe,$fe;-5
```

```
.db $03,$b0;-6
.db $03,$a7;-7
.db $03,$9f;-8
.db $03,$96;-9
.db $fe,$d4;-10
.db $03,$85;-11
.db $03,$7c;-12
.db $03,$73;-13
.db $03,$6a;-14
.db $fe,$ab;-15
.db $03,$59;-16
.db $03,$50;-17
.db $03,$48;-18
.db $03,$3f;-19
.db $fe,$82; -20
.db $03,$2e;-21
.db $03,$25;-22
.db $03,$1c;-23
.db $03,$13;-24
.db $fe,$57;-25
.db $03,$02;-26
.db $02,$f9;-27
.db $02,$f1;-28
.db $02,$e8;-29
.db $fe,$2d;-30

; esta tabla contiene los códigos a ser enviados al dac variable para las
temperaturas positivas
tablaposit:
; db msb,lsb
.db $ff,$21;0
.db $03,$ed;1
.db $03,$f6;2
.db $03,$fe;3
.db $04,$07;4
.db $ff,$51;5
.db $04,$19;6
.db $04,$21;7
.db $04,$2a;8
.db $04,$33;9
.db $ff,$7a;10
.db $04,$44;11
.db $04,$4d;12
.db $04,$56;13
.db $04,$5e;14
.db $ff,$a3;15
```

```
.db $04,$70;16
.db $04,$78;17
.db $04,$81;18
.db $04,$89;19
.db $ff,$cd;20
.db $04,$9b;21
.db $04,$a3;22
.db $04,$ac;23
.db $04,$b5;24
.db $ff,$f6;25
.db $04,$c6;26
.db $04,$cf;27
.db $04,$d7;28
.db $04,$e0;29
.db $00,$1f;30
.db $04,$f1;31
.db $04,$fa;32
.db $05,$02;33
.db $05,$0b;34
.db $00,$49;35
.db $05,$1c;36
.db $05,$25;37
.db $05,$2d;38
.db $05,$35;39
.db $00,$73;40
.db $05,$47;41
.db $05,$4f;42
.db $05,$58;43
.db $05,$60;44
.db $00,$a2;45
.db $05,$71;46
.db $05,$7a;47
.db $05,$82;48
.db $05,$8b;49
.db $00,$d0;50
.db $05,$9c;51
.db $05,$a4;52
.db $05,$ad;53
.db $05,$b5;54
.db $01,$02;55
.db $05,$c6;56
.db $05,$cf;57
.db $05,$d7;58
.db $05,$e0;59
.db $01,$32;60
.db $05,$f1;61
```

```
.db $05,$f9;62
.db $06,$01;63
.db $06,$0a;64
.db $01,$5b;65
.db $06,$1b;66
.db $06,$23;67
.db $06,$2b;68
.db $06,$34;69
.db $01,$84;70
.db $06,$45;71
.db $06,$4d;72
.db $06,$56;73
.db $06,$5e;74
.db $01,$ad;75
.db $06,$6f;76
.db $06,$77;77
.db $06,$80;78
.db $06,$88;79
.db $01,$d7;80
.db $06,$99;81
.db $06,$a1;82
.db $06,$aa;83
.db $06,$b2;84
.db $02,$04;85
.db $06,$c3;86
.db $06,$cb;87
.db $06,$d3;88
.db $06,$dc;89
.db $02,$30;90
.db $06,$ed;91
.db $06,$f5;92
.db $06,$fd;93
.db $07,$06;94
.db $02,$60;95
.db $07,$16;96
.db $07,$1f;97
.db $07,$27;98
.db $07,$30;99
.db $02,$90;100
.db $07,$40;101
.db $07,$48;102
.db $07,$51;103
.db $07,$59;104
.db $02,$b7;105
.db $07,$6a;106
.db $07,$72;107
```

```
.db $07,$7b;108
.db $07,$83;109
.db $02,$de;110
.db $07,$94;111
.db $07,$9c;112
.db $07,$a5;113
.db $07,$ad;114
.db $03,$08;115
.db $07,$be;116
.db $07,$c6;117
.db $07,$cf;118
.db $07,$d7;119
.db $03,$33;120
.db $07,$e8;121
.db $07,$f0;122
.db $07,$f9;123
.db $08,$01;124
.db $03,$63;125
.db $08,$12;126
.db $08,$1a;127
.db $08,$23;128
.db $08,$2b;129
.db $03,$92;130
.db $08,$3c;131
.db $08,$44;132
.db $08,$4d;133
.db $08,$55;134
.db $03,$c3;135
.db $08,$66;136
.db $08,$6f;137
.db $08,$77;138
.db $08,$80;139
.db $03,$f2;140
.db $08,$90;141
.db $08,$99;142
.db $08,$a1;143
.db $08,$aa;144
.db $04,$1d;145
.db $08,$bb;146
.db $08,$c3;147
.db $08,$cc;148
.db $08,$d4;149
.db $04,$3f;150
.db $08,$e5;151
.db $08,$ee;152
.db $08,$f6;153
```

```
.db $08,$ff;154
.db $04,$6a;155
.db $09,$10;156
.db $09,$18;157
.db $09,$21;158
.db $09,$29;159
.db $04,$95;160
.db $09,$3a;161
.db $09,$43;162
.db $09,$4b;163
.db $09,$54;164
.db $04,$c1;165
.db $09,$65;166
.db $09,$6e;167
.db $09,$76;168
.db $09,$7f;169
.db $04,$f2;170
.db $09,$90;171
.db $09,$99;172
.db $09,$a1;173
.db $09,$aa;174
.db $05,$17;175
.db $09,$bb;176
.db $09,$c3;177
.db $09,$cc;178
.db $09,$d5;179
.db $05,$42;180
.db $09,$e6;181
.db $09,$ee;182
.db $09,$f7;183
.db $09,$ff;184
.db $05,$6e;185
.db $0a,$11;186
.db $0a,$19;187
.db $0a,$22;188
.db $0a,$2b;189
.db $05,$9a;190
.db $0a,$3c;191
.db $0a,$45;192
.db $0a,$4d;193
.db $0a,$56;194
.db $05,$cb;195
.db $0a,$67;196
.db $0a,$70;197
.db $0a,$78;198
.db $0a,$81;199
```

```
.db $05,$f1;200
.db $0a,$93;201
.db $0a,$9b;202
.db $0a,$a4;203
.db $0a,$ad;204
.db $06,$1c;205
.db $0a,$be;206
.db $0a,$c7;207
.db $0a,$cf;208
.db $0a,$d8;209
.db $06,$47;210
.db $0a,$e9;211
.db $0a,$f2;212
.db $0a,$fb;213
.db $0b,$03;214
.db $06,$72;215
.db $0b,$15;216
.db $0b,$1e;217
.db $0b,$26;218
.db $0b,$2f;219
.db $06,$9d;220
.db $0b,$40;221
.db $0b,$49;222
.db $0b,$52;223
.db $0b,$5b;224
.db $06,$ca;225
.db $0b,$6c;226
.db $0b,$75;227
.db $0b,$7d;228
.db $0b,$86;229
.db $06,$f7;230
.db $0b,$98;231
.db $0b,$a0;232
.db $0b,$a9;233
.db $0b,$b2;234
.db $07,$23;235
.db $0b,$c3;236
.db $0b,$cc;237
.db $0b,$d5;238
.db $0b,$de;239
.db $07,$4e;240
.db $0b,$ef;241
.db $0b,$f8;242
.db $0c,$00;243
.db $0c,$09;244
.db $07,$7b;245
```

```
.db $0c,$1b;246
.db $0c,$24;247
.db $0c,$2c;248
.db $0c,$35;249
.db $07,$a8;250

; tabla para el DAC fijo
tablaneutro:
; db msb,lsb
.db $00,$00
```

ANEXO E

En este apartado referenciamos las hojas de datos de los componentes que se han seleccionado en el diseño del equipo.

DAC1220

<http://www.datasheetcatalog.org/datasheet/texasinstruments/dac1220.pdf>

LTC2431

<http://cds.linear.com/docs/Datasheet/24301f.pdf>

LTC2053

<http://cds.linear.com/docs/Datasheet/2053syncfc.pdf>

REF5025-REF5050

<http://www.ti.com/lit/ds/symlink/ref5025.pdf>

TPS7A49

<http://www.ti.com/lit/ds/symlink/tps7a4901.pdf>

LM6142

<http://www.ti.com/lit/ds/symlink/lm6142.pdf>

ATmega16

<http://www.atmel.com/Images/doc2466.pdf>

BZX84 (Diodo Zener)

<http://www.nxp.com/documents/data_sheet/BZX84_SERIES.pdf>

ADM8828

<http://www.analog.com/static/imported-files/data_sheets/ADM8828_8829.pdf>