ANEXOS

Anexo A: Medición del flujo.

Distrito	Flujo del agua fría (l/S)	Flujo del agua caliente (l/S)	Bomba o tanque	Variación porcentual (%)
Ate	0.13	0.14	Ninguno	7.14
Barranco	0.02	0.02	Ninguno	0
Cercado de Lima	0.12	0.11	Tanque	8.33
Chorrillos	0.11	0.12	Tanque	8.33
Jesús María	0.12	0.13	Tanque	7.69
La Molina	0.11	0.11	Ambos	0
La Victoria	0.15	0.15	Tanque	0
Lince	0.05	0.05	Tanque	0
Los Olivos	0.02	0.02	Tanque	0
Magdalena	0.14	0.15	Tanque	6.67
Miraflores	0.12	0.11	Ninguno	8.33
Pueblo libre	0.13	0.15	Tanque	13.33
Puente Piedra	0.03	0.03	Ninguno	0
San Borja	0.15	0.14	Bomba	6.67
San Isidro	0.08	0.07	Tanque	12.5
San Juan Lurigancho	0.03	0.03	Tanque	0
San Luis	0.13	0.12	Tanque	7.69
San Miguel	0.08	0.08	Tanque	0
Santa Anita	0.16	0.15	Tanque	6.25
Santiago de Surco	0.05	0.05	Ambos	0
Surquillo	0.04	0.04	Tanque	0
			Promedio	4.43

Anexo B: LM35 Datasheet

National Semiconductor

November 2000

LM35
Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

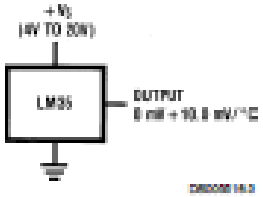
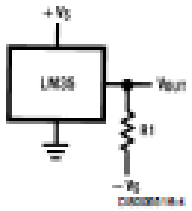


FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)



Choose $R_1 = -V_S/50\ \mu A$

V_{OUT} = +1,500 mV at +150°C

= +250 mV at +25°C

= −550 mV at −55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp.;	
TO-46 Package,	−60°C to +180°C
TO-92 Package,	−60°C to +150°C
SO-8 Package,	−65°C to +150°C
TO-220 Package,	−65°C to +150°C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300°C
TO-92 and TO-220 Package, (Soldering, 10 seconds)	260°C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55°C to +150°C
LM35C, LM35CA	−40°C to +110°C
LM35D	0°C to +100°C

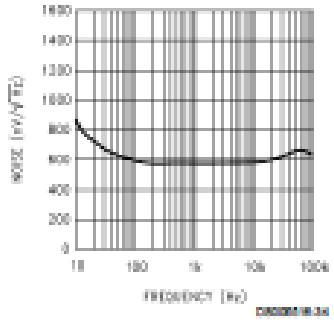
Electrical Characteristics

(Notes 1, 6)

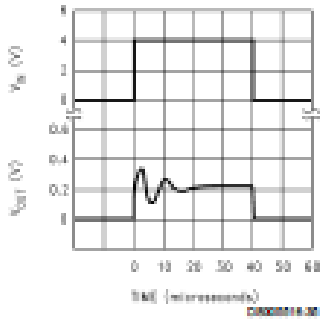
Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy (Note 7)	T_A=+25°C	±0.2	±0.5		±0.2	±0.5		°C
	T_A=−10°C	±0.3			±0.3		±1.0	°C
	T_A=T_{MAX}	±0.4	±1.0		±0.4	±1.0		°C
	T_A=T_{MIN}	±0.4	±1.0		±0.4		±1.5	°C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	±0.18		±0.35	±0.15		±0.3	°C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	+10.0	+9.8, +10.1		+10.0		+9.8, +10.1	mV/°C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25°C	±0.4	±1.0		±0.4	±1.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	±0.6		±3.0	±0.6		±3.0	mV/mA
Line Regulation (Note 3)	T_A=+25°C	±0.01	±0.05		±0.01	±0.05		mV/V
	4V≤V_S≤30V	±0.02		±0.1	±0.02		±0.1	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25°C	56	67		56	67		µA
	V_S=+5V	105		131	91		114	µA
	V_S=+30V, +25°C	56.2	68		56.2	68		µA
	V_S=+30V	106.6		133	91.6		116	µA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25°C	0.2	1.0		0.2	1.0		µA
	4V≤V_S≤30V	0.6		2.0	0.6		2.0	µA
Temperature Coefficient of Quiescent Current		+0.39		+0.5	+0.39		+0.5	µA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			°C

Typical Performance Characteristics (Continued)

Noise Voltage



Start-Up Response



Applications

The LM35 can be applied easily in the same way as other integrated-circuit temperature sensors. It can be glued or cemented to a surface and its temperature will be within about 0.01°C of the surface temperature.

This presumes that the ambient air temperature is almost the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature. This is especially true for the TO-92 plastic package, where the copper leads are the principal thermal path to carry heat into the device, so its temperature might be closer to the air temperature than to the surface temperature.

To minimize this problem, be sure that the wiring to the LM35, as it leaves the device, is held at the same temperature as the surface of interest. The easiest way to do this is to cover up these wires with a bead of epoxy which will insure that the leads and wires are all at the same temperature as the surface, and that the LM35 die's temperature will not be affected by the air temperature.

The TO-46 metal package can also be soldered to a metal surface or pipe without damage. Of course, in that case the V− terminal of the circuit will be grounded to that metal. Alternatively, the LM35 can be mounted inside a sealed-end metal tube, and can then be dipped into a bath or screwed into a threaded hole in a tank. As with any IC, the LM35 and accompanying wiring and circuits must be kept insulated and dry, to avoid leakage and corrosion. This is especially true if the circuit may operate at cold temperatures where condensation can occur. Printed-circuit coatings and varnishes such as Humiseal and epoxy paints or dips are often used to insure that moisture cannot corrode the LM35 or its connections.

These devices are sometimes soldered to a small light-weight heat fin, to decrease the thermal time constant and speed up the response in slowly-moving air. On the other hand, a small thermal mass may be added to the sensor, to give the steadiest reading despite small deviations in the air temperature.

Temperature Rise of LM35 Due To Self-heating (Thermal Resistance, θ_{JA})

	TO-46, no heat sink	TO-46*, small heat fin	TO-92, no heat sink	TO-92**, small heat fin	SO-8 no heat sink	SO-8** small heat fin	TO-220 no heat sink
Still air	400°C/W	100°C/W	180°C/W	140°C/W	220°C/W	110°C/W	90°C/W
Moving air	100°C/W	40°C/W	90°C/W	70°C/W	105°C/W	90°C/W	26°C/W
Still oil	100°C/W	40°C/W	90°C/W	70°C/W			
Stirred oil	50°C/W	30°C/W	45°C/W	40°C/W			
(Clamped to metal, infinite heat sink)	(24°C/W)				(55°C/W)		

*Wakefield type 201, or 1" disc of 0.020" sheet brass, soldered to case, or similar.
**TO-92 and SO-8 packages glued and leads soldered to 1" square of 1/16" printed circuit board with 2 oz. foil or similar.

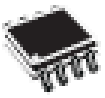
Anexo C: UA741 Datasheet

UA741

GENERAL PURPOSE SINGLE OPERATIONAL AMPLIFIER

- LARGE INPUT VOLTAGE RANGE
- NO LATCH-UP
- HIGH GAIN
- SHORT-CIRCUIT PROTECTION
- NO FREQUENCY COMPENSATION
- REQUIRED
- SAME PIN CONFIGURATION AS THE UA709

N
DIP8
(Plastic Package)



D
SO8
(Plastic Micropackage)

DESCRIPTION

The UA741 is a high performance monolithic operational amplifier constructed on a single silicon chip. It is intented for a wide range of analog applications.

- Summing amplifier
- Voltage follower
- Integrator
- Active filter
- Function generator

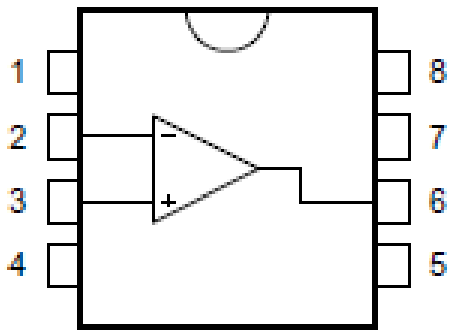
The high gain and wide range of operating voltages provide superior performances in integrator, summing amplifier and general feedback applications. The internal compensation network (6dB/ octave) insures stability in closed loop circuits.

ORDER CODE

Part Number	Temperature Range	Package	
		N	D
UA741C	0°C, +70°C	•	•
UA741I	-40°C, +105°C	•	•
UA741M	-55°C, +125°C	•	•
Example : UA741CN			

N = Dual in Line Package (DIP)
D = Small Outline Package (SO) - also available in Tape & Reel (DT)

PIN CONNECTIONS (top view)



1 - Offset null 1
2 - Inverting input
3 - Non-inverting input
4 - V_{CC}^-
5 - Offset null 2
6 - Output
7 - V_{CC}^+
8 - N.C.

November 2001 1/5

ELECTRICAL CHARACTERISTICS

V_{CC} = ±15V, T_{amb} = +25°C (unless otherwise specified)

Symbol	Parameter	Min.	Typ.	Max.	Unit
V_{io}	Input Offset Voltage ($R_s \leq 10k\Omega$) T_{amb} = +25°C $T_{min} \leq T_{amb} \leq T_{max}$		1	5 6	mV
I_{io}	Input Offset Current T_{amb} = +25°C $T_{min} \leq T_{amb} \leq T_{max}$		2	30 70	nA
I_{ib}	Input Bias Current T_{amb} = +25°C $T_{min} \leq T_{amb} \leq T_{max}$		10	100 200	nA
A_{vd}	Large Signal Voltage Gain (V_O = ±10V, R_L = 2kΩ) T_{amb} = +25°C $T_{min} \leq T_{amb} \leq T_{max}$	50 25	200		V/mV
SVR	Supply Voltage Rejection Ratio ($R_s \leq 10k\Omega$) T_{amb} = +25°C $T_{min} \leq T_{amb} \leq T_{max}$	77 77	90		dB
I_{CC}	Supply Current, no load T_{amb} = +25°C $T_{min} \leq T_{amb} \leq T_{max}$		1.7	2.8 3.3	mA
V_{icm}	Input Common Mode Voltage Range T_{amb} = +25°C $T_{min} \leq T_{amb} \leq T_{max}$	±12 ±12			V
CMR	Common Mode Rejection Ratio ($R_S \leq 10k\Omega$) T_{amb} = +25°C $T_{min} \leq T_{amb} \leq T_{max}$	70 70	90		dB
I_{OS}	Output short Circuit Current	10	25	40	mA
$\pm V_{opp}$	Output Voltage Swing T_{amb} = +25°C, R_L = 10kΩ R_L = 2kΩ $T_{min} \leq T_{amb} \leq T_{max}$, R_L = 10kΩ R_L = 2kΩ	12 10 12 10	14 13		V
SR	Slew Rate V_i = ±10V, R_L = 2kΩ, C_L = 100pF, unity Gain	0.25	0.5		V/µs
t_r	Rise Time V_i = ±20mV, R_L = 2kΩ, C_L = 100pF, unity Gain		0.3		µs
K_{ov}	Overshoot V_i = 20mV, R_L = 2kΩ, C_L = 100pF, unity Gain		5		%
R_i	Input Resistance	0.3	2		MΩ
GBP	Gain Bandwith Product V_i = 10mV, R_L = 2kΩ, C_L = 100pF, f = 100kHz	0.7	1		MHz
THD	Total Harmonic Distortion f = 1kHz, A_v = 20dB, R_L = 2kΩ, V_O = $2V_{pp}$, C_L = 100pF, T_{amb} = +25°C		0.06		%
e_n	Equivalent Input Noise Voltage f = 1kHz, R_s = 100Ω		23		$\frac{nV}{\sqrt{Hz}}$
$\varnothing_m$	Phase Margin		50		Degrees

Anexo D: DAC0800 Datasheet

National Semiconductor

September 2006

DAC0800/DAC0802
8-Bit Digital-to-Analog Converters

General Description

The DAC0800 series are monolithic 8-bit high-speed current-output digital-to-analog converters (DAC) featuring typical settling times of 100 ns. When used as a multiplying DAC, monotonic performance over a 40 to 1 reference current range is possible. The DAC0800 series also features high compliance complementary current outputs to allow differential output voltages of 20 Vp-p with simple resistor loads. The reference-to-full-scale current matching of better than ±1 LSB eliminates the need for full-scale trims in most applications, while the nonlinearities of better than ±0.1% over temperature minimizes system error accumulations.

The noise immune inputs will accept a variety of logic levels. The performance and characteristics of the device are essentially unchanged over the ±4.5V to ±18V power supply range and power consumption at only 33 mW with ±5V supplies is independent of logic input levels.

The DAC0800, DAC0802, DAC0800C and DAC0802C are a direct replacement for the DAC-08, DAC-08A, DAC-08C, and DAC-08H, respectively. For single supply operation, refer to AN-1525.

Features

- Fast settling output current: 100 ns
- Full scale error: ±1 LSB
- Nonlinearity over temperature: ±0.1%
- Full scale current drift: ±10 ppm/°C
- High output compliance: −10V to +18V
- Complementary current outputs
- Interface directly with TTL, CMOS, PMOS and others
- 2 quadrant wide range multiplying capability
- Wide power supply range: ±4.5V to ±18V
- Low power consumption: 33 mW at ±5V
- Low cost

Typical Application

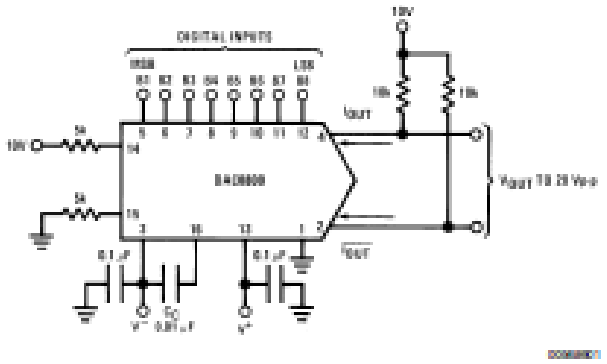


FIGURE 1. ±20 V_{p-p} Output Digital-to-Analog Converter (Note 4)

Ordering Information

Non-Linearity	Temperature Range (T_A)	Order Numbers				
		J Package (J16A) *		N Package (N16E) *		SO Package (M16A)
±0.1% FS	0°C to +70°C	DAC0802LCJ	DAC-08HQ	DAC0802LCN	DAC-08HP	DAC0802LCM
±0.19% FS	−55°C to +125°C	DAC0800LJ	DAC-08Q			
±0.19% FS	0°C to +70°C	DAC0800LCJ	DAC-08EQ	DAC0800LCN	DAC-08EP	DAC0800LCM

* Devices may be ordered by using either order number.

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage ($V^+ - V^-$)	±18V or 36V
Power Dissipation (Note 2)	500 mW
Reference Input Differential Voltage (V14 to V15)	V^- to V^+
Reference Input Common-Mode Range (V14, V15)	V^- to V^+
Reference Input Current	5 mA
Logic Inputs	V^- to V^- plus 36V
Analog Current Outputs (V_{S-} = –15V)	4.25 mA
ESD Susceptibility (Note 3)	TBD V
Storage Temperature	–65°C to +150°C
Lead Temp. (Soldering, 10 seconds)	
Dual-In-Line Package (plastic)	260°C
Dual-In-Line Package (ceramic)	300°C
Surface Mount Package	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C

Operating Conditions (Note 1)

	Min	Max	Units
Temperature (T_A)			
DAC0800L	–55	+125	°C
DAC0800LC	0	+70	°C
DAC0802LC	0	+70	°C
V^+	(V^-) + 10	(V^-) + 30	V
V_{-}	–15	–5	V
I_{REF} (V^- = –5V)	1	2	mA
I_{REF} (V^- = –15V)	1	4	mA

Electrical Characteristics

The following specifications apply for V_S = ±15V, I_{REF} = 2 mA and $T_{MIN} \le T_A \le T_{MAX}$ unless otherwise specified. Output characteristics refer to both I_{OUT} and $\overline{I_{OUT}}$.

Symbol	Parameter	Conditions	DAC0802LC			DAC0800L/ DAC0800LC			Units
			Min	Typ	Max	Min	Typ	Max	
	Resolution		8	8	8	8	8	8	Bits
	Monotonicity		8	8	8	8	8	8	Bits
	Nonlinearity				±0.1			±0.19	%FS
t_s	Settling Time	To ±½ LSB, All Bits Switched "ON" or "OFF", T_A=25°C		100	135				ns
		DAC0800L					100	135	ns
		DAC0800LC					100	150	ns
t_{PLH}, t_{PHL}	Propagation Delay	T_A=25°C							
	Each Bit			35	60		35	60	ns
	All Bits Switched			35	60		35	60	ns
TCI_{FS}	Full Scale Tempco			±10	±50		±10	±50	ppm/°C
V_{OC}	Output Voltage Compliance	Full Scale Current Change <½ LSB, R_{OUT}>20 MΩ, Typical	–10		18	–10		18	V
I_{FS4}	Full Scale Current	V_{REF} = 10.000V, R14 = R15 = 5.000 kΩ, T_A=25°C	1.984	1.992	2.00	1.94	1.99	2.04	mA
I_{FSS}	Full Scale Symmetry	I_{FS4}–I_{FS2}		±0.5	±4.0		±1	±8.0	µA
I_{ZS}	Zero Scale Current			0.1	1.0		0.2	2.0	µA
I_{FSR}	Output Current Range	V^- = –5V	0	2.0	2.1	0	2.0	2.1	mA
		V^- = –8V to –18V	0	2.0	4.2	0	2.0	4.2	

Anexo E: Calculo ADC

Voltaje antes del amplificador (V)	Voltaje amplificado (V)	Valor Decimal teórico	Valor Binario obtenido
0	0	0	0
0.01	0.05	2.55	10
0.02	0.1	5.1	101
0.03	0.15	7.65	111
0.04	0.2	10.2	1010
0.05	0.25	12.75	1100
0.06	0.3	15.3	1111
0.07	0.35	17.85	10001
0.08	0.4	20.4	10100
0.09	0.45	22.95	10110
0.1	0.5	25.5	11001
0.11	0.55	28.05	11100
0.12	0.6	30.6	11110
0.13	0.65	33.15	100001
0.14	0.7	35.7	100011
0.15	0.75	38.25	100110
0.16	0.8	40.8	101000
0.17	0.85	43.35	101011
0.18	0.9	45.9	101101
0.19	0.95	48.45	110000
0.2	1	51	110011
0.21	1.05	53.55	110101
0.22	1.1	56.1	111000
0.23	1.15	58.65	111010
0.24	1.2	61.2	111101
0.25	1.25	63.75	111111
0.26	1.3	66.3	1000010
0.27	1.35	68.85	1000100
0.28	1.4	71.4	1000111
0.29	1.45	73.95	1001001
0.3	1.5	76.5	1001100
0.31	1.55	79.05	1001111
0.32	1.6	81.6	1010001
0.33	1.65	84.15	1010100
0.34	1.7	86.7	1010110
0.35	1.75	89.25	1011001
0.36	1.8	91.8	1011011
0.37	1.85	94.35	1011110
0.38	1.9	96.9	1100000
0.39	1.95	99.45	1100011

0.4	2	102	1100110
0.41	2.05	104.55	1101000
0.42	2.1	107.1	1101011
0.43	2.15	109.65	1101101
0.44	2.2	112.2	1110000
0.45	2.25	114.75	1110010
0.46	2.3	117.3	1110101
0.47	2.35	119.85	1110111
0.48	2.4	122.4	1111010
0.49	2.45	124.95	1111100
0.5	2.5	127.5	1111111
0.51	2.55	130.05	10000010
0.52	2.6	132.6	10000100
0.53	2.65	135.15	10000111
0.54	2.7	137.7	10001001
0.55	2.75	140.25	10001100
0.56	2.8	142.8	10001110
0.57	2.85	145.35	10010001
0.58	2.9	147.9	10010011
0.59	2.95	150.45	10010110
0.6	3	153	10011001
0.61	3.05	155.55	10011011
0.62	3.1	158.1	10011110
0.63	3.15	160.65	10100000
0.64	3.2	163.2	10100011
0.65	3.25	165.75	10100101
0.66	3.3	168.3	10101000
0.67	3.35	170.85	10101010
0.68	3.4	173.4	10101101
0.69	3.45	175.95	10101111
0.7	3.5	178.5	10110010
0.71	3.55	181.05	10110101
0.72	3.6	183.6	10110111
0.73	3.65	186.15	10111010
0.74	3.7	188.7	10111100
0.75	3.75	191.25	10111111
0.76	3.8	193.8	11000001
0.77	3.85	196.35	11000100
0.78	3.9	198.9	11000110
0.79	3.95	201.45	11001001
0.8	4	204	11001100
0.81	4.05	206.55	11001110
0.82	4.1	209.1	11010001
0.83	4.15	211.65	11010011
0.84	4.2	214.2	11010110

0.85	4.25	216.75	11011000
0.86	4.3	219.3	11011011
0.87	4.35	221.85	11011101
0.88	4.4	224.4	11100000
0.89	4.45	226.95	11100010
0.9	4.5	229.5	11100101
0.91	4.55	232.05	11101000
0.92	4.6	234.6	11101010
0.93	4.65	237.15	11101101
0.94	4.7	239.7	11101111
0.95	4.75	242.25	11110010
0.96	4.8	244.8	11110100
0.97	4.85	247.35	11110111
0.98	4.9	249.9	11111001
0.99	4.95	252.45	11111100
1	5	255	11111111

Anexo F: Atmega16 Datasheet

Features

- High-performance, Low-power Atmel® AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory segments
  - 16 Kbytes of In-System Self-programmable Flash program memory
  - 512 Bytes EEPROM
  - 1 Kbyte Internal SRAM
  - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
  - Data retention: 20 years at 85°C/100 years at 25°C[1]
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    - 8 Single-ended Channels
    - 7 Differential Channels in TQFP Package Only
    - 2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7V - 5.5V for ATmega16L
  - 4.5V - 5.5V for ATmega16
- Speed Grades
  - 0 - 8 MHz for ATmega16L
  - 0 - 16 MHz for ATmega16
- Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16L
  - Active: 1.1 mA
  - Idle Mode: 0.35 mA
  - Power-down Mode: < 1 µA

ATMEL®

8-bit AVR® Microcontroller with 16K Bytes In-System Programmable Flash

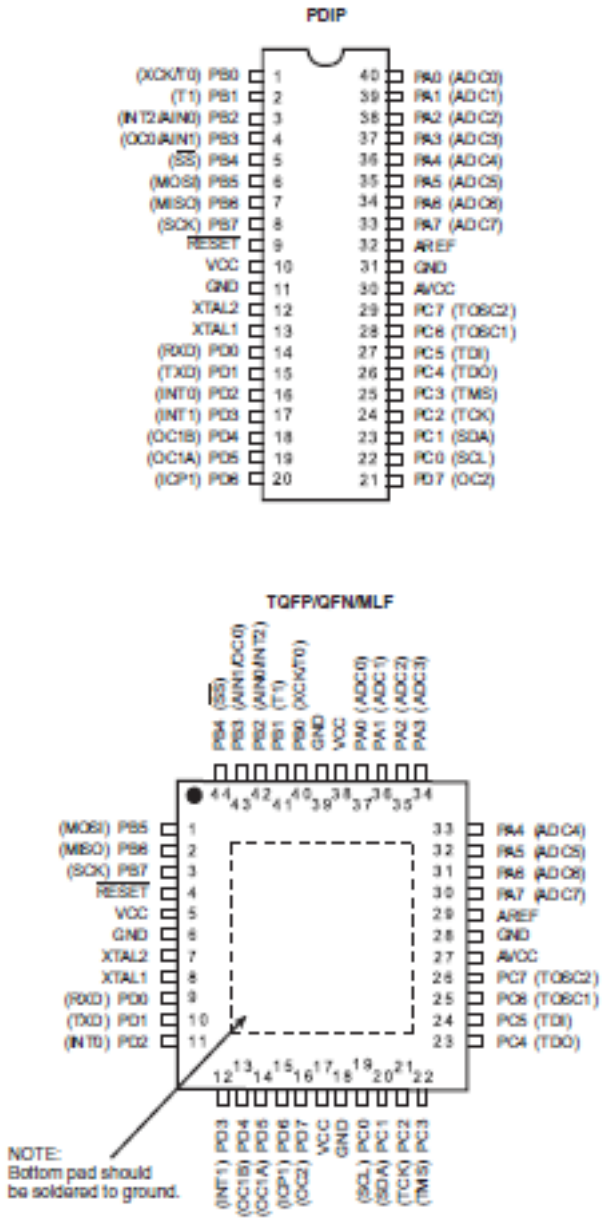
ATmega16
ATmega16L

Rev. 2466T–AVR–07/10

ATMEL®

Pin Configurations

Figure 1. Pinout ATmega16



Disclaimer

Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

Anexo G: Diagramas de flujo

Diagrama N°1: En el programa principal, primero se configuran los puertos, el LCD y se inicializan las variables. Se pregunta al usuario si desea configurar o iniciar, luego de lo cual se realiza el proceso escogido y se deja la ducha abierta hasta que el usuario lo decida.

Diagrama N°2: En la subrutina de configuración, se dan las instrucciones para seleccionar la temperatura deseada, se sensan los botones “+”, “-” y se va variando la temperatura, se verifica que la temperatura deseada se encuentre dentro del rango establecido, se asigna el porcentaje de apertura de las válvulas y se guarda la configuración escogida cuando se ha presionado “fin”.

Diagrama N°3: En la subrutina de selección, se solicita al usuario que elija alguno de los usuarios, se lee de la memoria EEPROM la temperatura asignada al usuario escogido y se asigna el porcentaje de apertura de las válvulas según la temperatura.

Diagrama N°4: En la subrutina guardar, se verifica si es la primera vez que se usa la memoria EEPROM, en tal caso se guarda la temperatura escogida en el primer espacio de memoria, en caso contrario se revisa un indicador del número de usuario, libre en el cual se guarda la temperatura escogida y se actualiza el indicador, en caso el indicador sea mayor a la cantidad de usuarios disponibles se espera a que el usuario elija el número de usuario a reemplazar.

Diagrama N°5: En la subrutina función, se desarrolla la ecuación de correspondencia del flujo del agua fría o caliente en función de la temperatura deseada, para lo cual primero se verifica que se pueda alcanzar dicha temperatura y se fija la apertura de una de las válvulas al 100% para determinar la apertura de la otra, según la temperatura deseada sea mayor o menor al promedio de las temperaturas en las tuberías.

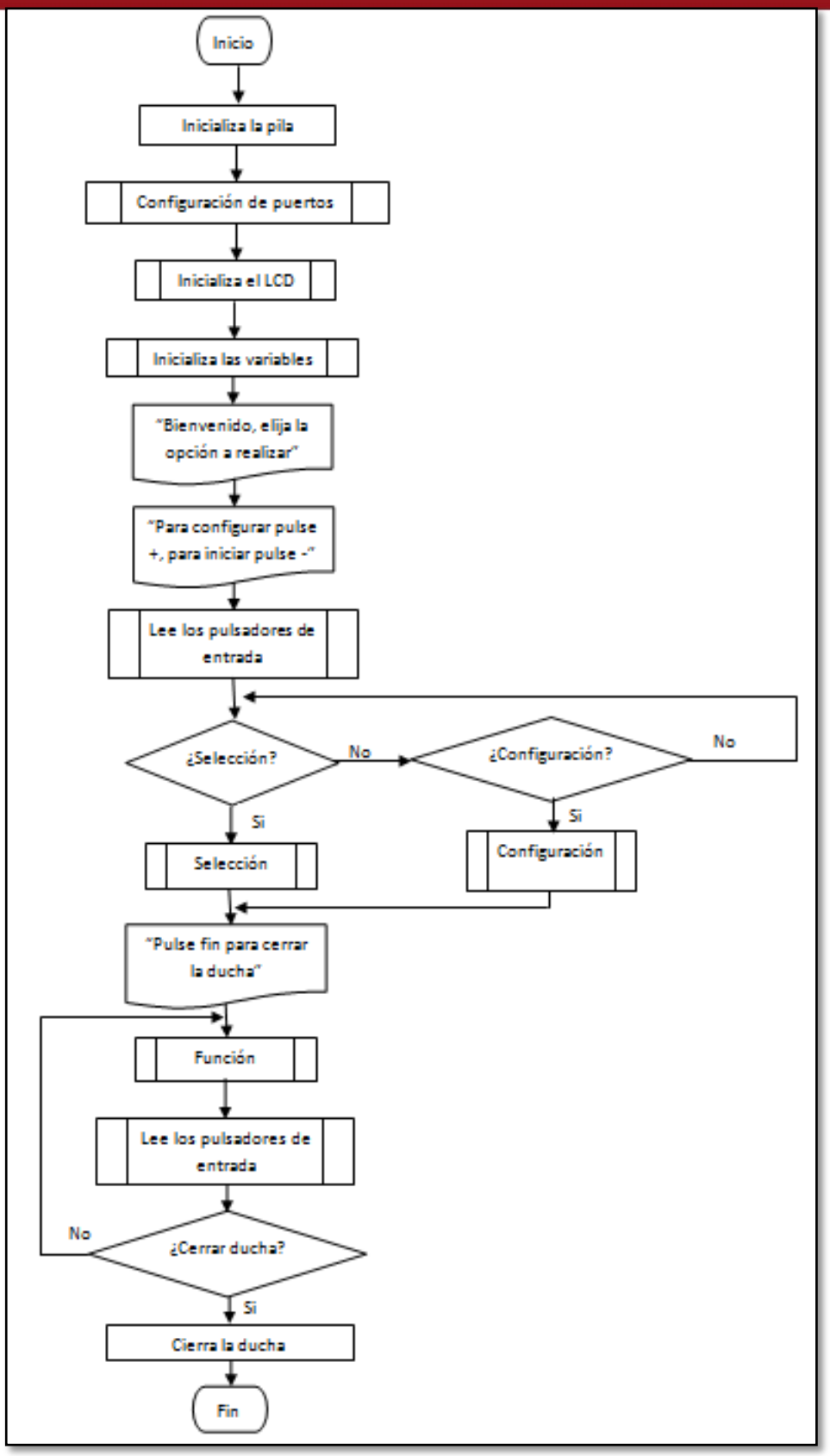


Diagrama N°1 Programa Principal.

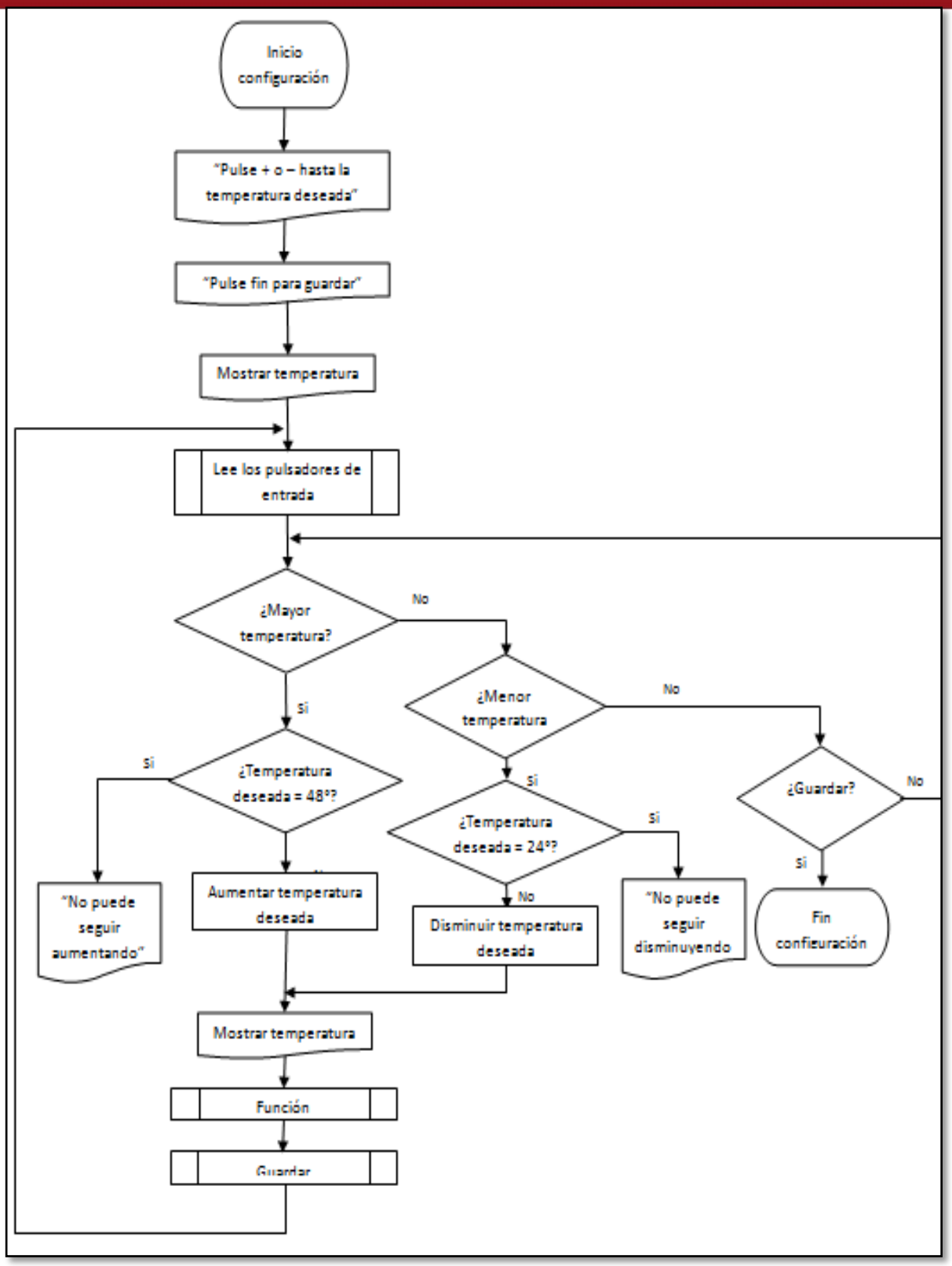


Diagrama N°2 Configuración de la temperatura.

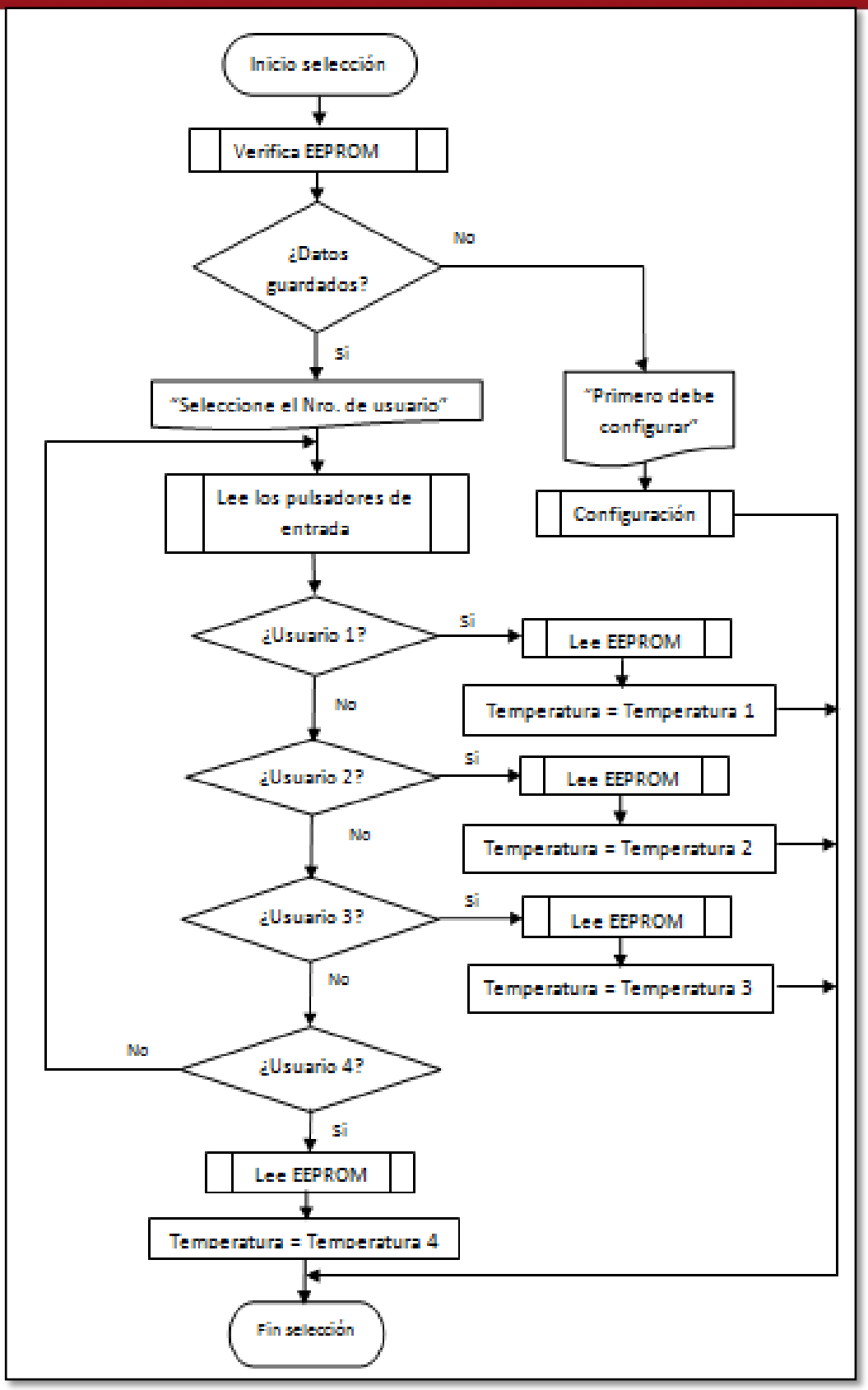


Diagrama N°3 Selección del usuario.

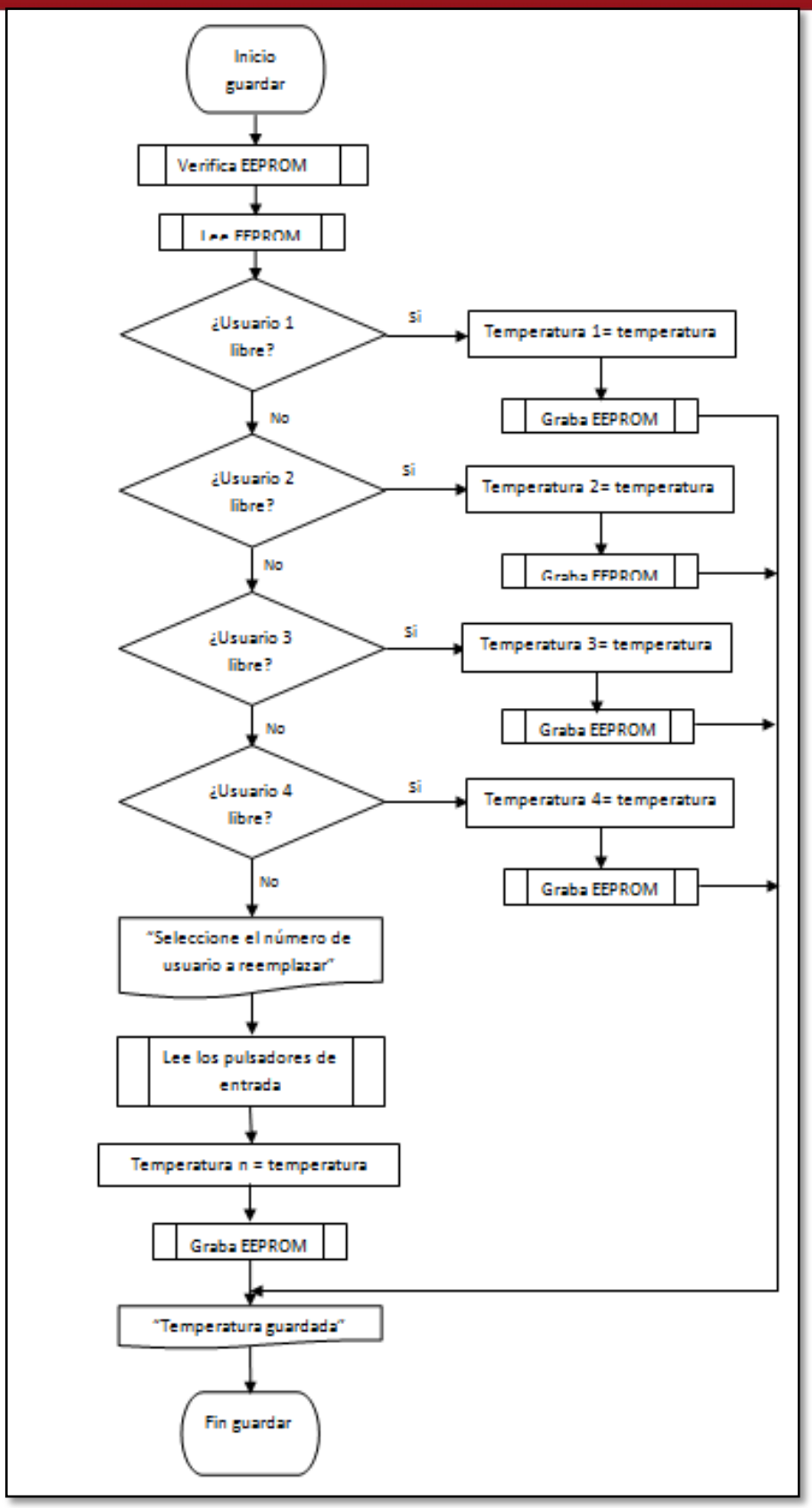


Diagrama N°4 Guardar temperatura escogida.

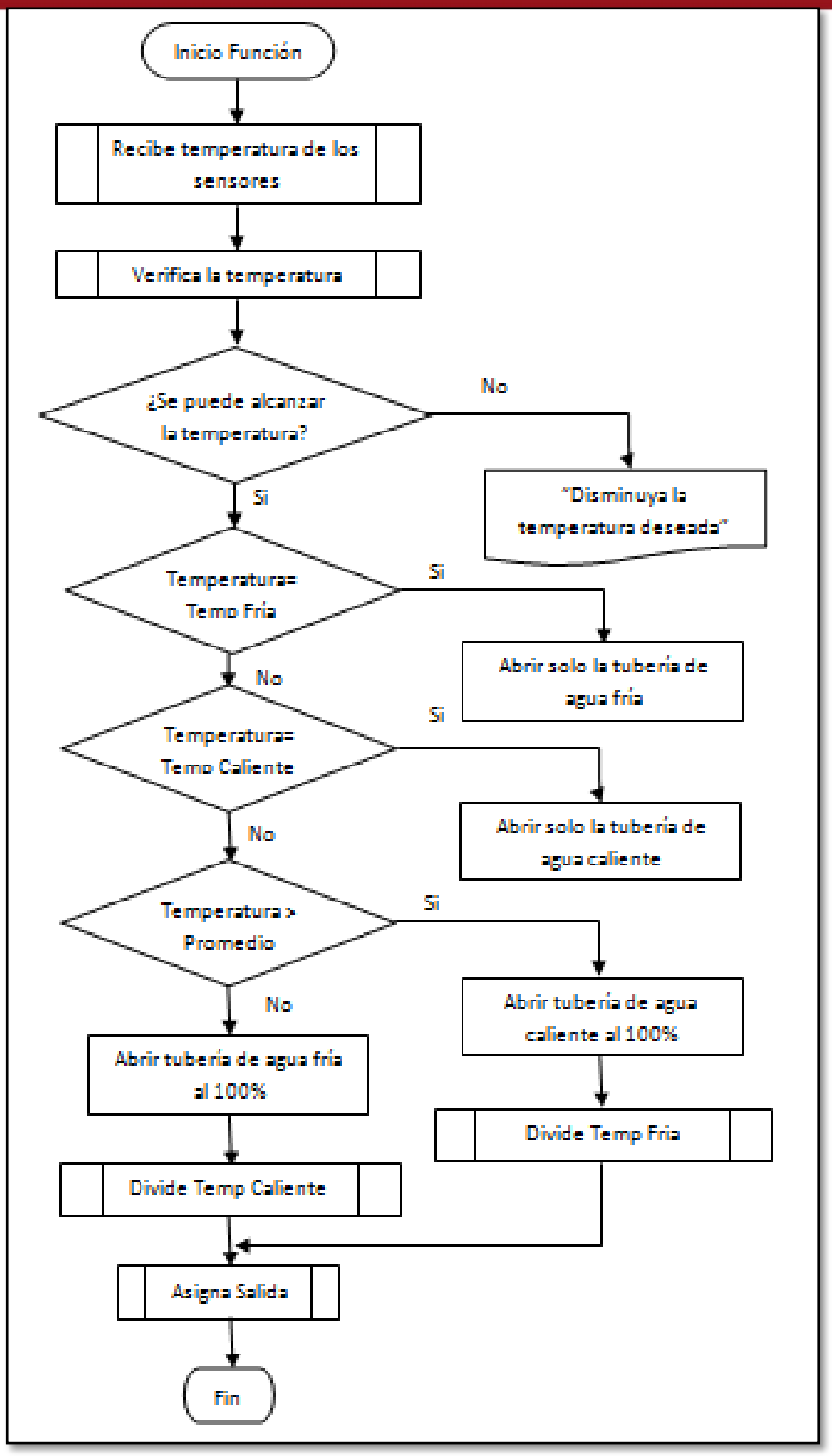


Diagrama N°5 Función de correspondencia entre temperatura y flujo.

Anexo H: Circuitos impresos

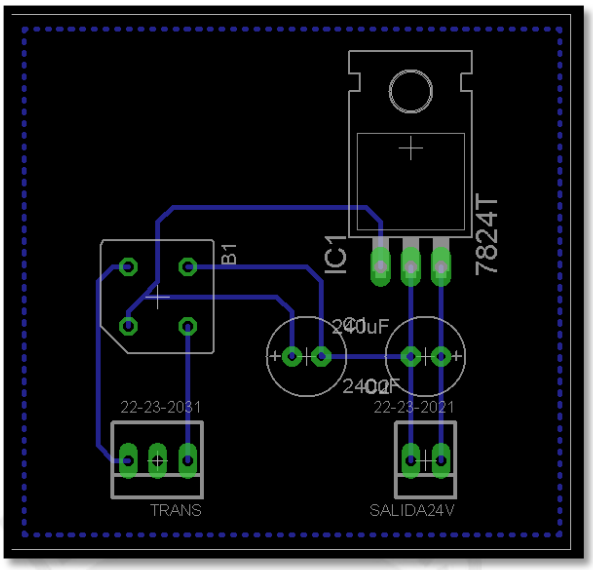


Fig. N°1 Diseño del circuito impreso de la alimentación de la parte de potencia para los actuadores de válvula

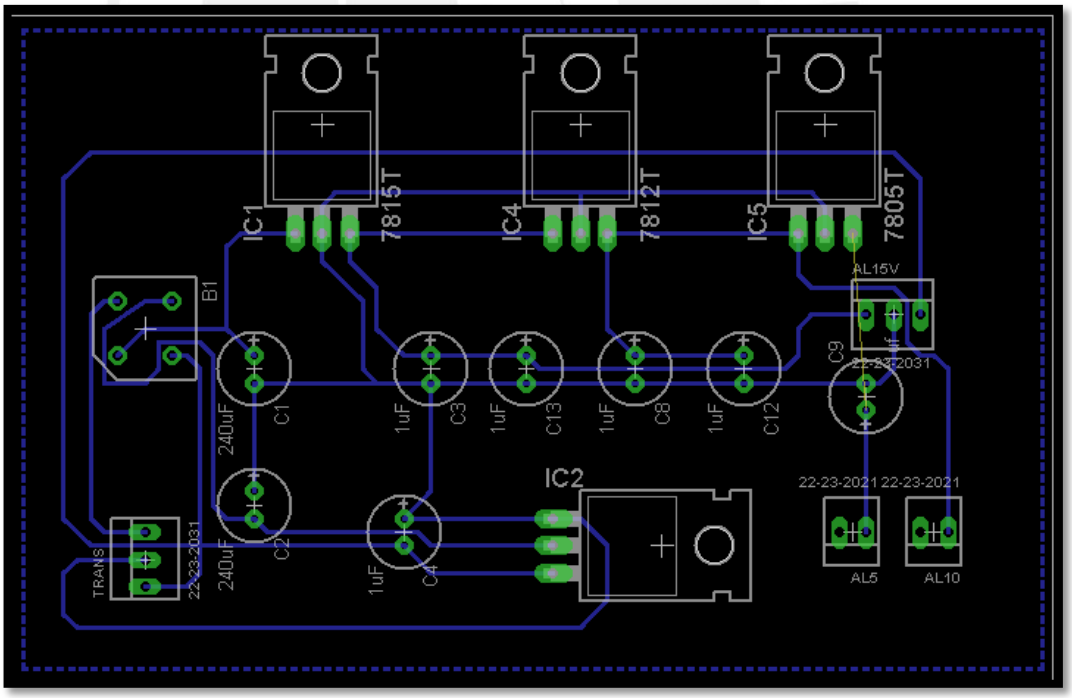


Fig. N°2 Diseño del circuito impreso de la alimentación de la parte de control.

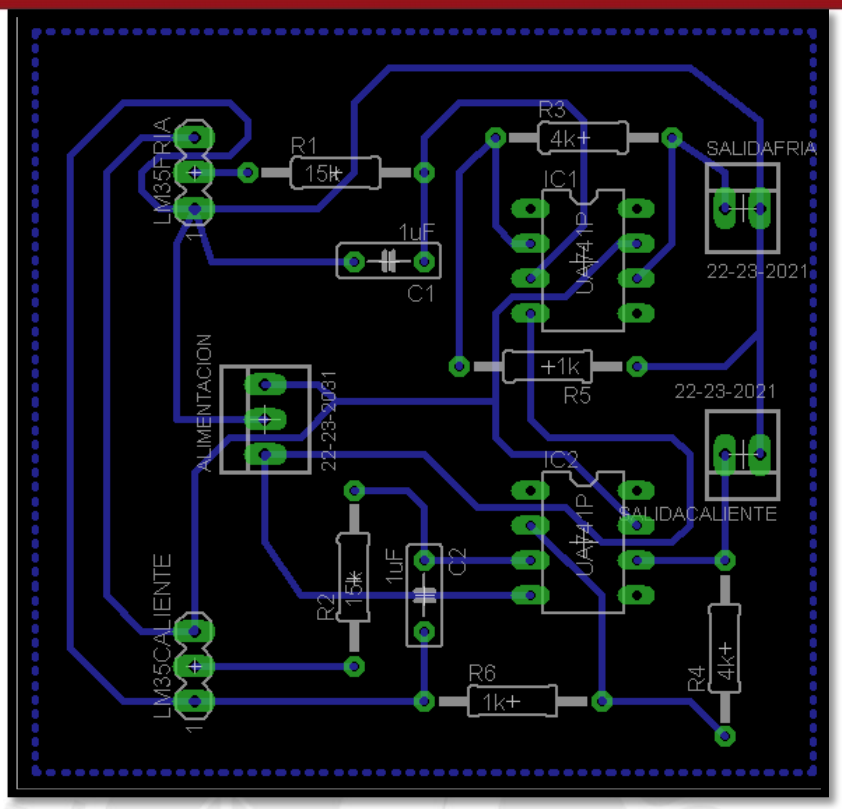


Fig. N°3 Diseño del circuito impreso del acondicionamiento de las señales de entrada.

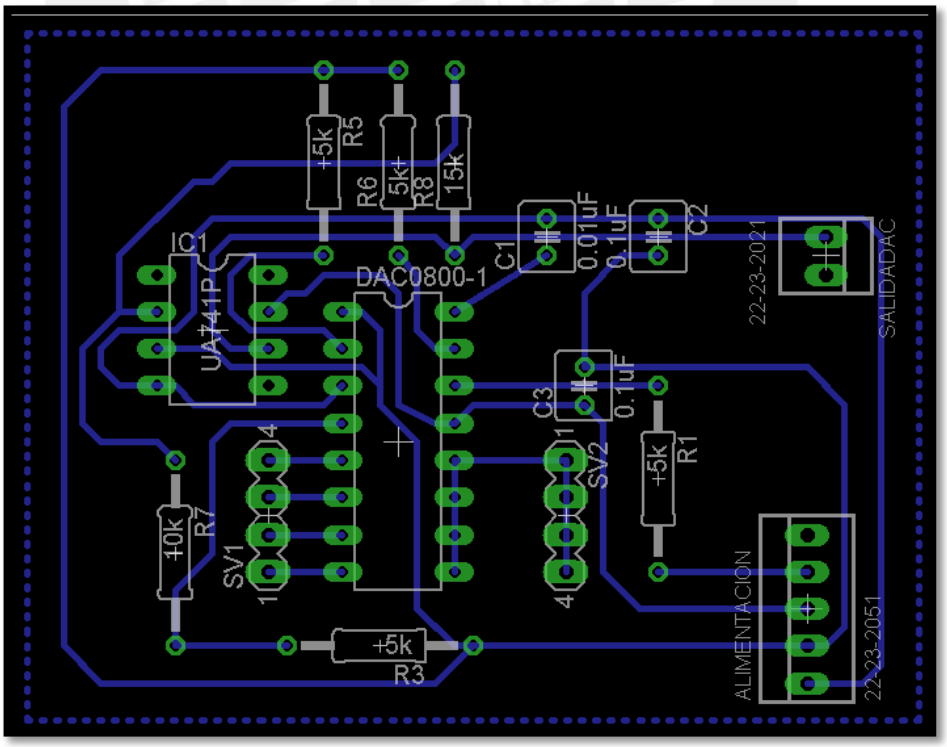


Fig. N°4 Diseño del circuito impreso del acondicionamiento de las señales de salida para cada actuador.

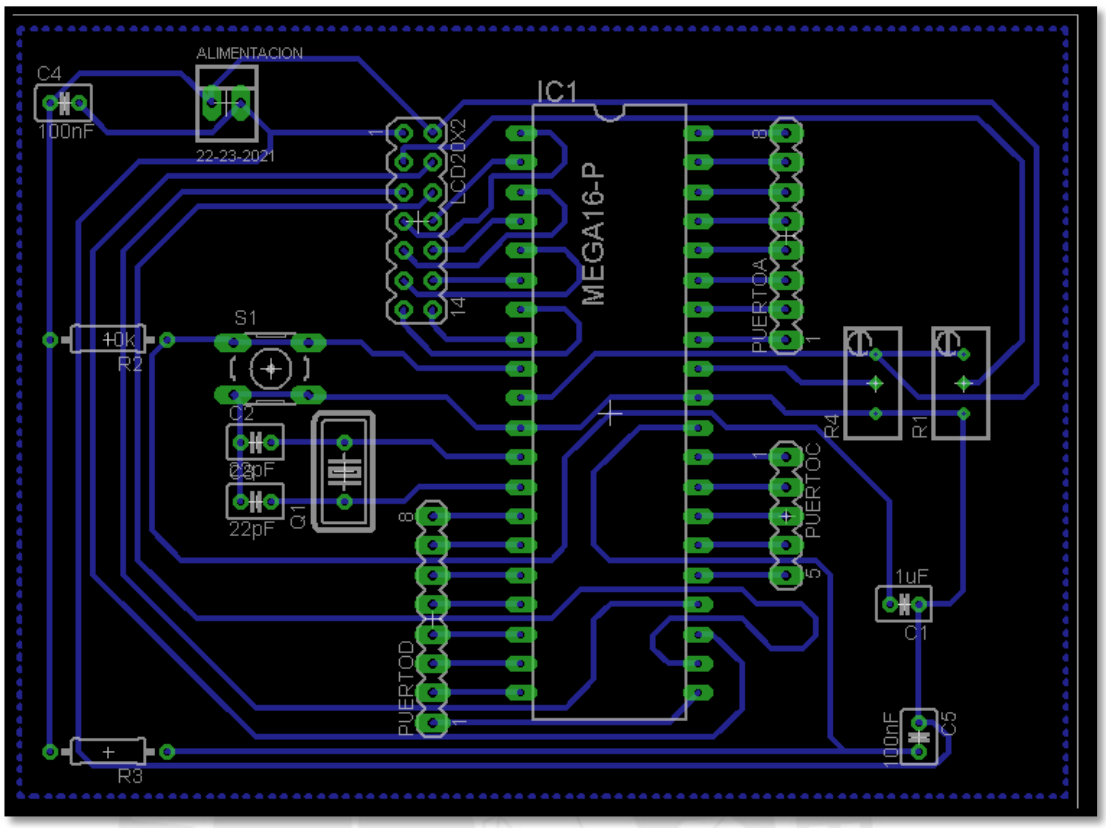


Fig. N°5 Diseño del circuito impreso del Atmega16 y el LCD.

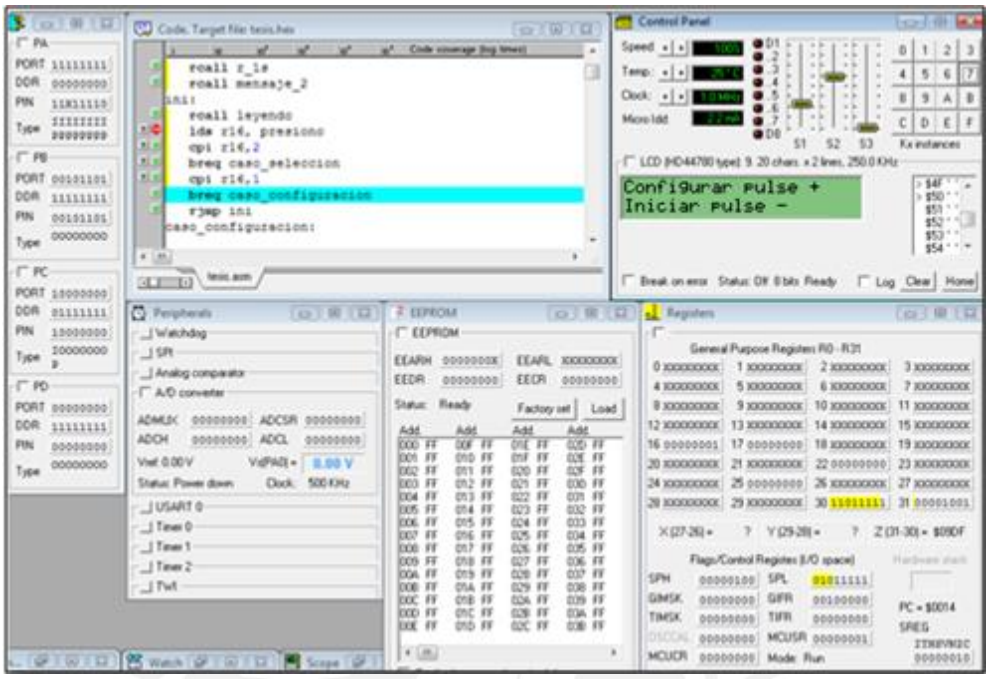


Fig. N°4.5.1.3. Menú de inicio. Primer uso, memoria EEPROM libre.

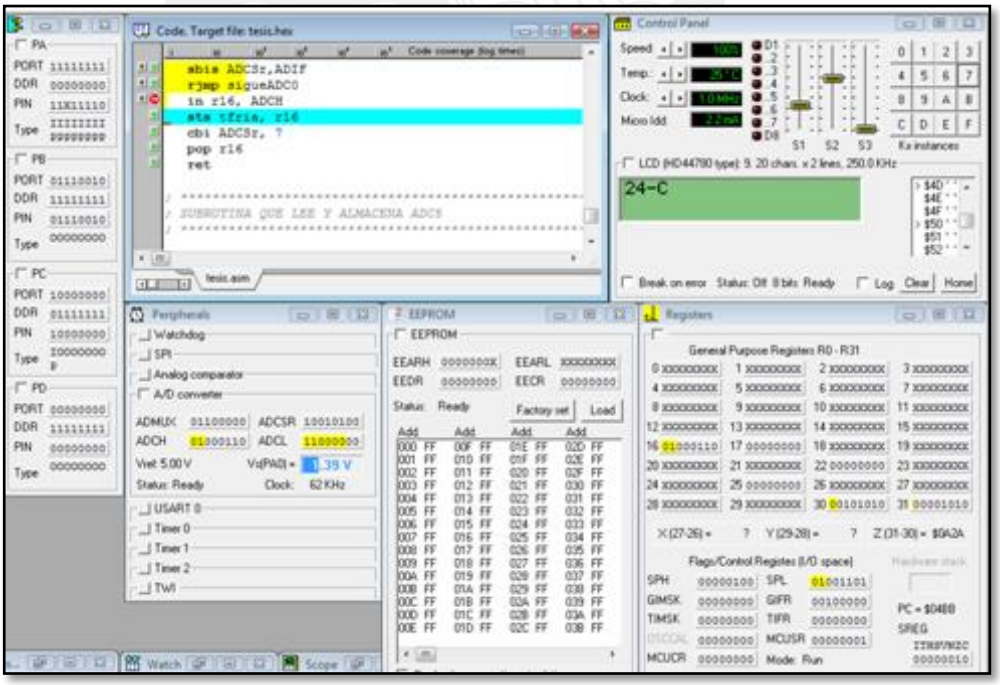


Fig. N°4.5.1.4. Ingreso de temperatura fría.

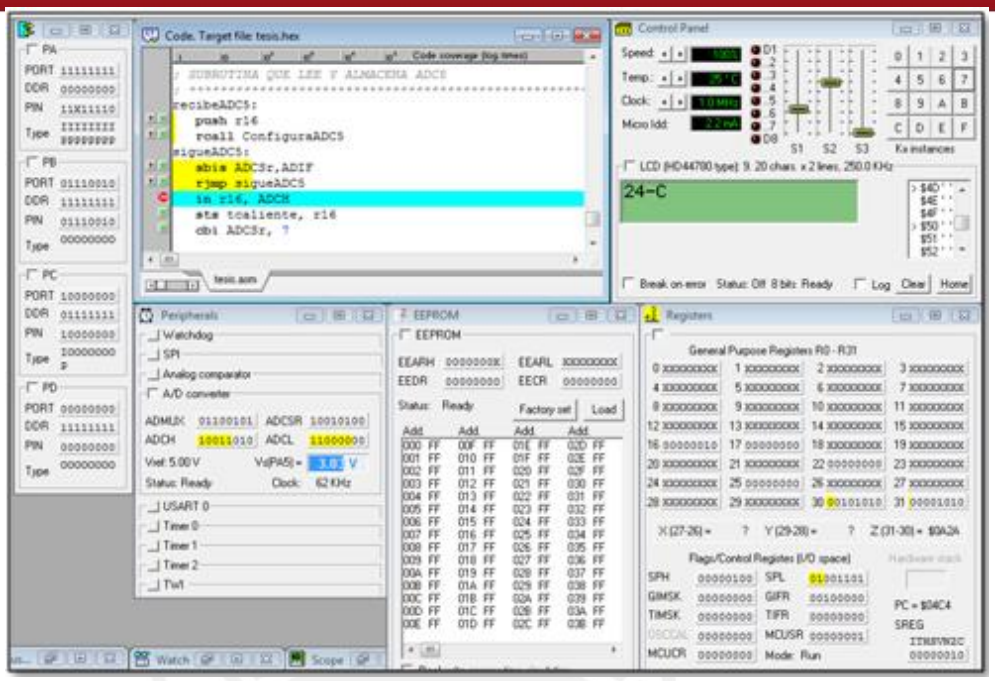


Fig. N°4.5.1.5. Ingreso de temperatura caliente.

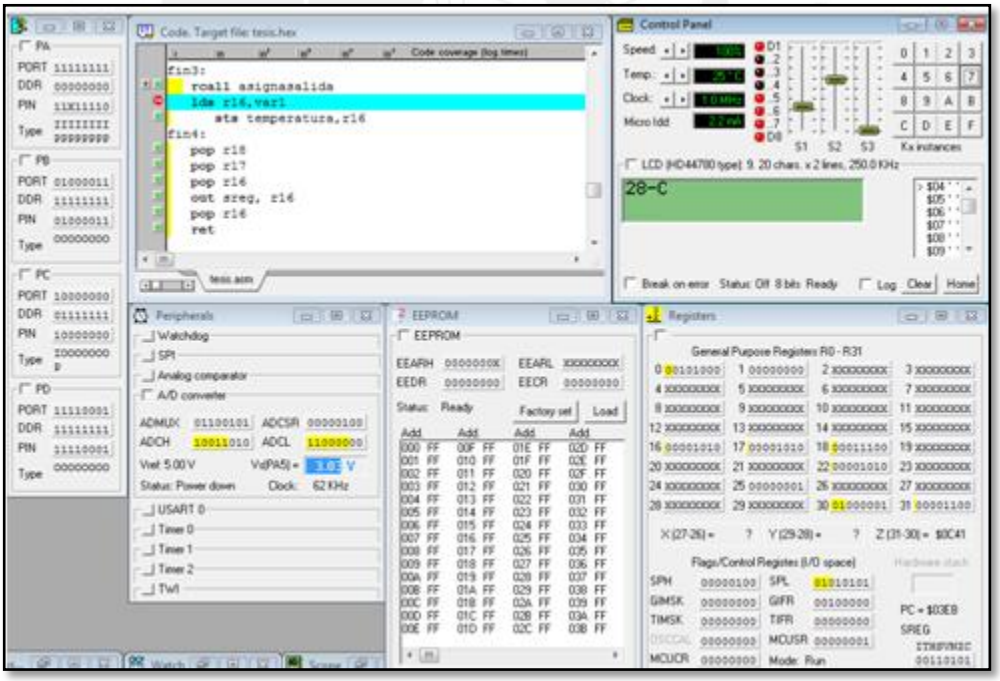


Fig. N°4.5.1.6. Salida de los actuadores, puerto D, representado en los leds para 28°C.

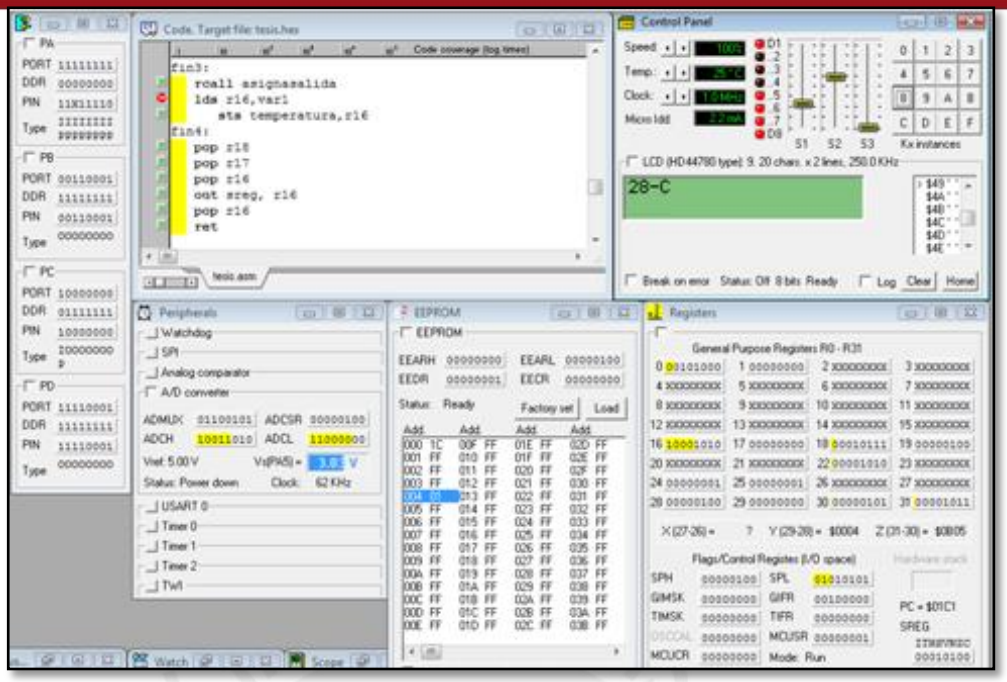


Fig. N°4.5.1.7. Configuración guardada en EEPROM para el usuario 1.

Anexo J: Código ensamblador

```
; *****************************************************
; BASIC .ASM template file for AVr
; *****************************************************
.include "C:\VMLAB\include\m16def.inc"
        .def    Rf=r22
        .def    Rc=r25
        .def    Lcd_dato = r17
.dseg
 .org $060
        Temperatura:    .byte 1
        Presiono:       .byte 1
        Verificar:      .byte 1
        Cont_lee:       .byte 1
        Cantguardada:   .byte 1
        Cont_escribe:   .byte 1
        Vadc0:          .byte 1
        Vadc5:          .byte 1
        Tfria:          .byte 1
        Tcaliente:      .byte 1
        Ffria:          .byte 1
        Fcaliente:      .byte 1
        Var2:           .byte 1
        Var1:           .byte 1
        Verificador:    .byte 1
.cseg
reset:
        rjmp start
; *****************************************************
; PROGRAMA PRINCIPAL
; *****************************************************
start:
        ldi r16, high(ramend)
        out sph, r16
        ldi r16, low(ramend)
        out spl, r16
        rcall conf_port
        rcall ini_LCD
        rcall ini_variables
inicio:
        clr r16
        out portd,r16
        rcall mensaje_1
        rcall r_1s
        rcall r_1s
        rcall mensaje_2
ini:
        rcall leyendo
        lds r16, presiono
        cpi r16,2
        breq caso_seleccion
        cpi r16,1
        breq caso_configuracion
        rjmp ini
caso_configuracion:
        rcall configuracion
        rcall r_1s
        rjmp fin_principal1
```

```
caso_seleccion:
        rcall seleccion
        rcall funcion
fin_principal1:
        rcall mensaje_6
fin_principal2:
        rcall leyendo
        lds r16,presiono
        cpi r16,3
        breq fin_principal3
        rjmp fin_principal2
fin_principal3:
        rcall r_1s
        rcall r_1s
        rjmp start
; *******************************************************
; FUNCION QUE LEE QUE ENTRADA HA SIDO PRESIONADA
; *******************************************************
leyendo:
        push r16
        push r18
        in r16, sreg
        push r16
        push r19
        clr r19
inileyendo:
        sbic pina,7
        rjmp otro1
        rjmp pin_mas
otro1:
        sbic pina,6
        rjmp otro2
        rjmp pin_menos
otro2:
        sbic pinc,7
        rjmp otro3
        rjmp pin_fin
otro3:
        sbic pina,1
        rjmp otro4
        rjmp pin_usu1
otro4:
        sbic pina,2
        rjmp otro5
        rjmp pin_usu2
otro5:
        sbic pina,3
        rjmp otro6
        rjmp pin_usu3
otro6:
        sbic pina,4
        rjmp inileyendo
        rjmp pin_usu4
pin_mas:
        inc  r19
        cpi  r19,200
        breq ini_pin_mas
        rjmp pin_mas
ini_pin_mas:
        sbis pina,7
```

```
        rjmp ini_pin_mas
        ldi r16,1
        sts presiono,r16
        rjmp finleyendo1
pin_menos:
        inc  r19
        cpi  r19,200
        breq ini_pin_menos
        rjmp pin_menos
ini_pin_menos:
        sbis pina,6
        rjmp ini_pin_menos
        ldi r16,2
        sts presiono,r16
        rjmp finleyendo1
pin_fin:
        inc  r19
        cpi  r19,200
        breq ini_pin_fin
        rjmp pin_fin
ini_pin_fin:
        sbis pinc,7
        rjmp ini_pin_fin
        ldi r16,3
        sts presiono,r16
        rjmp finleyendo1
pin_usu1:
        inc  r19
        cpi  r19,200
        breq ini_pin_usu1
        rjmp pin_usu1
ini_pin_usu1:
        sbis pina,1
        rjmp ini_pin_usu1
        ldi r16,4
        sts presiono,r16
        rjmp finleyendo1
pin_usu2:
        inc  r19
        cpi  r19,200
        breq ini_pin_usu2
        rjmp pin_usu2
ini_pin_usu2:
        sbis pina,2
        rjmp ini_pin_usu2
        ldi r16,5
        sts presiono,r16
        rjmp finleyendo1
pin_usu3:
        inc  r19
        cpi  r19,200
        breq ini_pin_usu3
        rjmp pin_usu3
ini_pin_usu3:
        sbis pina,3
        rjmp ini_pin_usu3
        ldi r16,6
        sts presiono,r16
        rjmp finleyendo1
pin_usu4:
```

```
        inc  r19
        cpi  r19,200
        breq ini_pin_usu4
        rjmp pin_usu4
ini_pin_usu4:
        sbis pina,4
        rjmp ini_pin_usu4
        ldi r16,7
        sts presiono,r16
        rjmp finleyendo1
finleyendo1:
        clr r19
finleyendo2:
        inc r19
        cpi r19,200
        breq finleyendo3
        rjmp finleyendo2
finleyendo3:
        pop r19
        pop r16
        out sreg, r16
        pop r18
        pop r16
        ret
; *****************************************************
; CONFIGURACION DE PUERTOS
; *****************************************************
conf_port:
        ldi r16, 0b01111111
        out DDRC, r16       ;Puerto C: Salidas pines de control RS= PC0, RW= PC1, E=
PC6, Entrada PC7= Fin
        ldi r16, 0b11111111
        out DDRB, r16       ; Puerto B: Salida de datos al LCD
        ldi r16, 0b11111111
        out DDRD, r16       ; Puerto D: Salida a los convertidores D/A, bits 7-4= Válvula fria,
bits 3-0= Válvula caliente
        ldi r16, 0b00000000
        out DDRA, r16       ; Puerto A: Entradas PA1= Usuario1, PA2= Usuario2,
PA3=Usuario3, PA4= Usuario4
        ldi r16, 0b11111111
        out porta, r16              ;PA0= Sensor agua fría, PA5= Sensor agua caliente, PA6=
Aumentar, PA7= Disminuir
        ldi r16, 0b10000000
        out portc, r16
        ret
; *****************************************************
; INICIALIZACIÓN DE LAS VARIABLES
; *****************************************************
ini_variables:
        clr r16
        sts Verificar, r16
        sts Cont_lee, r16
        sts Cantguardada, r16
        sts Cont_escribe, r16
        sts var1, r16
        sts Var2, r16
        sts Ffria,r16
        sts Fcaliente,r16
        sts Presiono, r16
        clr Rf
```

```
        clr Rc
        ldi r16,24
        sts Temperatura, r16
        ret
; *****************************************************
; CONFIGURACION LCD
; *****************************************************
ini_LCD:
        ldi r17, 0b00111100
        rcall envia_ir
        rcall r_15u
        ldi r17, 0b00001111
        rcall envia_ir
        rcall r_15u
        ldi r17, 0b00000001
        rcall envia_ir
        rcall r_15u
        ldi r17, 0b00000110
        rcall envia_ir
        ret
; *****************************************************
; MENSAJES AL LCD
; *****************************************************
mensaje_24:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje24*2)
        ldi ZL, LOW(mensaje24*2)
        rcall envia_msj
        ret
mensaje_28:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje28*2)
        ldi ZL, LOW(mensaje28*2)
        rcall envia_msj
        ret
mensaje_32:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje32*2)
        ldi ZL, LOW(mensaje32*2)
        rcall envia_msj
        ret
mensaje_36:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje36*2)
        ldi ZL, LOW(mensaje36*2)
    rcall envia_msj
        ret
mensaje_40:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje40*2)
        ldi ZL, LOW(mensaje40*2)
    rcall envia_msj
        ret
mensaje_44:
        rcall clr_LCD
```

```
        rcall r_50ms
        ldi ZH, HIGH(mensaje44*2)
        ldi ZL, LOW(mensaje44*2)
        rcall envia_msj
        ret
mensaje_48:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje48*2)
        ldi ZL, LOW(mensaje48*2)
        rcall envia_msj
        ret
mensaje_1:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje1*2)
        ldi ZL, LOW(mensaje1*2)
        rcall envia_msj
        ret
mensaje_2:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje2*2)
        ldi ZL, LOW(mensaje2*2)
        rcall envia_msj
        ret
mensaje_3:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje3*2)
        ldi ZL, LOW(mensaje3*2)
        rcall envia_msj
        ret
mensaje_4:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje4*2)
        ldi ZL, LOW(mensaje4*2)
        rcall envia_msj
        ret
mensaje_5:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje5*2)
        ldi ZL, LOW(mensaje5*2)
        rcall envia_msj
        ret
mensaje_6:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje6*2)
        ldi ZL, LOW(mensaje6*2)
        rcall envia_msj
        ret
mensaje_7:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje7*2)
        ldi ZL, LOW(mensaje7*2)
        rcall envia_msj
```

```
        ret
mensaje_8:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje8*2)
        ldi ZL, LOW(mensaje8*2)
        rcall envia_msj
        ret
mensaje_9:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje9*2)
        ldi ZL, LOW(mensaje9*2)
        rcall envia_msj
        ret
mensaje_10:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje10*2)
        ldi ZL, LOW(mensaje10*2)
        rcall envia_msj
        ret
mensaje_11:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje11*2)
        ldi ZL, LOW(mensaje11*2)
        rcall envia_msj
        ret
mensaje_12:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje12*2)
        ldi ZL, LOW(mensaje12*2)
        rcall envia_msj
        ret
mensaje_13:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje13*2)
        ldi ZL, LOW(mensaje13*2)
        rcall envia_msj
        ret
mensaje_14:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje14*2)
        ldi ZL, LOW(mensaje14*2)
        rcall envia_msj
        ret
mensaje_15:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje15*2)
        ldi ZL, LOW(mensaje15*2)
        rcall envia_msj
        ret
mensaje_16:
        rcall clr_LCD
        rcall r_50ms
```

```
        ldi ZH, HIGH(mensaje16*2)
        ldi ZL, LOW(mensaje16*2)
        rcall envia_msj
        ret
mensaje_17:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje17*2)
        ldi ZL, LOW(mensaje17*2)
        rcall envia_msj
        ret
mensaje_18:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje18*2)
        ldi ZL, LOW(mensaje18*2)
        rcall envia_msj
        ret
mensaje_19:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje19*2)
        ldi ZL, LOW(mensaje19*2)
        rcall envia_msj
        ret
mensaje_20:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje20*2)
        ldi ZL, LOW(mensaje20*2)
        rcall envia_msj
        ret
mensaje_21:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje21*2)
        ldi ZL, LOW(mensaje21*2)
        rcall envia_msj
        ret
mensaje_22:
        rcall clr_LCD
        rcall r_50ms
        ldi ZH, HIGH(mensaje22*2)
        ldi ZL, LOW(mensaje22*2)
        rcall envia_msj
        ret
; ******************************************************
; ENVIA INSTRUCCION AL LCD
; ENTRADA: R17= DATO DEL LCD, PC0= RS, PC1= RW, PC6= E
; ******************************************************
envia_ir:
        cbi portc,0
        cbi portc,1
        cbi portc,6
        nop
        sbi portc,6
        nop
        out portb,r17
        rcall r_15u
        cbi portc,6
```

```
        nop
        ret
; *****************************************************
; ENVIA DATO AL LCD
; ENTRADA: R17= DATO DEL LCD, PC0= RS, PC1= RW, PC6= E
; *****************************************************
envia_dr:
        cbi portc,0
        cbi portc,1
        cbi portc,6
        nop
        sbi portc,6
        nop
        sbi portc,0
        nop
        out portb,r17
        rcall r_15u
        cbi portc,6
        cbi portc,0
        nop
        ret
; *****************************************************
; ENVIA MSJ AL LCD
; ENTRADA: DIRECCIÓN DEL MENSAJE A MOSTRAR
; *****************************************************
envia_msj:
        lpm r17, Z+
        cpi r17,'*'
        breq sec_line
        cpi r17, 0
        breq end_envia_msj
        rcall envia_dr
        rjmp envia_msj
sec_line:
        ldi r17, 0b11000000
        rcall envia_ir
        rjmp envia_msj
end_envia_msj:
        ret
; *****************************************************
; LIMPIA LCD
; COLOCA PUNTERO AL INICIO DE PANTALLA
; *****************************************************
clr_lcd:
        push r16
        in r16, sreg
        push r16
        ldi r17,$01
        rcall envia_ir
        pop r16
        out sreg, r16
        pop r16
        ret
; *****************************************************
; RETARDO 15us
; *****************************************************
r_15u:
        nop
        nop
        nop
```

```
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        ret
; *****************************************************
; RETARDO 50ms
; *****************************************************
r_50ms:
        push r16
        in r16, sreg
        push  r16
        push r18
        ldi r18,60
lazo0_r_50ms:
        ldi r16,$FF
lazo1_r_50ms:
        dec  r16
        brne lazo1_r_50ms
        dec r18
        brne lazo0_r_50ms
        pop r18
        pop r16
        out sreg, r16
        pop r16
        ret
; *****************************************************
; RETARDO 1s
; *****************************************************
r_1s:
        push r16
        in r16, sreg
        push  r16
        push r18
        push r19
        ldi r19,6
lazo2_r_1s:
        ldi r18,$FF
lazo0_r_1s:
        ldi r16,$FF
lazo1_r_1s:
        dec  r16
        brne lazo1_r_1s
        dec r18
        brne lazo0_r_1s
        dec r19
        brne lazo2_r_1s
        pop r19
        pop r18
        pop r16
        out sreg, r16
        pop r16
        ret
; *****************************************************************
; SUBRUTINA QUE GUARDA LA TEMPERATURA DESEADA
; *****************************************************************
guardar:
```

```
        push r20
        push r16
        in r16, sreg
        push  r16
        ldi r16,5
        sts cont_lee, r16
        rcall Verifica_EEPROM
        lds r16, verificar
        cpi r16, 0
        breq guarda1
sigue:
        rcall leeEEPROM
        sts cantguardada,r20
        lds r16,cantguardada
        cpi r16, 1
        breq guarda2
        cpi r16, 2
        breq guarda3
        cpi r16, 3
        breq guarda4
        rcall mensaje_21
        rcall r_1s
        rcall r_1s
        rcall mensaje_22
        rcall r_1s
        rcall r_1s
ini_EEPROM_leyendo:
        rcall leyendo
        lds r16,presiono
        cpi r16, 4
        breq guarda1
        cpi r16, 5
        breq guarda2
        cpi r16, 6
        breq guarda3
        cpi r16, 7
        breq guarda4
        rjmp ini_EEPROM_leyendo
guarda1:
        ldi r16,1
        sts cont_escribe,r16
        lds r24, temperatura
        rcall grabaEEPROM
        rcall mensaje_12
        rjmp sal1
guarda2:
        ldi r16,2
        sts cont_escribe,r16
        lds r24, temperatura
        rcall grabaEEPROM
        rcall mensaje_13
        rjmp sal1
guarda3:
        ldi r16,3
        sts cont_escribe,r16
        lds r24, temperatura
        rcall grabaEEPROM
        rcall mensaje_14
        rjmp sal1
guarda4:
```

```
            ldi r16,4
            sts cont_escribe,r16
            lds r24, temperatura
            rcall grabaEEPROM
            rcall mensaje_15
sal1:
            ldi r16,5
            sts cont_escribe,r16
            lds r16,cantguardada
            inc r16
            sts cantguardada,r16
            mov r24,r16
            rcall grabaEEPROM
            pop r16
            out sreg, r16
            pop r16
            pop r20
            ret
; ********************************************************************
; SUBRUTINA QUE GRABA EN EEPROM
; ENTRADA: R24= DATO A SER GUARDADO EN EEPROM
; YH-YL: DIRECCIÓN DONDE SE GUARDA EL RESULTADO
; ********************************************************************
grabaEEPROM:
            push r16
            in r16, sreg
            push  r16
            push r24
            ldi yh, high(0)
            ldi yl, low (0)
suma1:
            lds r16, cont_escribe
            cpi r16, 1
            breq suma1_1
            rjmp suma2
suma1_1:
            rjmp finsuma
suma2:
            lds r16, cont_escribe
            cpi r16, 2
            breq suma2_1
            rjmp suma3
suma2_1:
            adiw YL,1
            rjmp finsuma
suma3:
            lds r16, cont_escribe
            cpi r16, 3
            breq suma3_1
            rjmp suma4
suma3_1:
            adiw YL,2
            rjmp finsuma
suma4:
            lds r16, cont_escribe
            cpi r16, 4
            breq suma4_1
            rjmp suma5
suma4_1:
            adiw YL,3
```

```
        rjmp finsuma
suma5:
        lds r16, cont_escribe
        cpi r16, 5
        breq suma5_1
        rjmp finsuma
suma5_1:
        adiw YL,4
        rjmp finsuma
finsuma:
        out EEArH,YH
        out EEArL,YL
        out EEDr,r24
        sbi EECr,EEMWE
        sbi EECr,EEWE
Espera:
        sbic EECr,EEWE
        rjmp Espera
        pop r24
        pop r16
        out sreg, r16
        pop r16
        ret
; ********************************************************************
; SUBRUTINA QUE LEE EEPROM POR PRIMERA VEZ
; ********************************************************************
leeEEPROM1:
        out EEArH,YH
        out EEArL,YL
        sbi EECr,EErE
        in r20,EEDr
Espera1:
        sbic EECr,EErE
        rjmp Espera1
        ret
; ********************************************************************
; SUBRUTINA QUE LEE EEPROM
; ********************************************************************
leeEEPROM:
        ldi yh, high(0)
        ldi yl, low (0)
elije1:
        lds r16, cont_lee
        cpi r16, 1
        breq elije1_1
        rjmp elije2
elije1_1:
        rjmp elije
elije2:
        lds r16, cont_lee
        cpi r16, 2
        breq elije2_1
        rjmp elije3
elije2_1:
        adiw YL,1
        rjmp elije
elije3:
        lds r16, cont_lee
        cpi r16, 3
        breq elije3_1
```

```
            rjmp elije4
elije3_1:
            adiw YL,2
            rjmp elije
elije4:
            lds r16, cont_lee
            cpi r16, 4
            breq elije4_1
            rjmp elije5
elije4_1:
            adiw YL,3
            rjmp elije
elije5:
            lds r16, cont_lee
            cpi r16, 5
            breq elije5_1
            rjmp elije1
elije5_1:
            adiw YL,4
elije:
            out EEArH,YH
            out EEArL,YL
            sbi EECr,EErE
            in r20,EEDr
Espera1_1:
            sbic EECr,EErE
            rjmp Espera1_1
            ret
; ********************************************************************
; SUBRUTINA QUE CONFIGURA LA TEMPERATURA PARA CADA USUARIO
; ********************************************************************
configuracion:
            push r16
            in r16, sreg
            push  r16
            rcall mensaje_3
            rcall r_1s
            rcall r_1s
            rcall mensaje_4
            rcall r_1s
            rcall r_1s
            rjmp ini_cen24
inierror1:
            rcall mensaje_9
            rcall r_1s
            rcall r_1s
            rjmp ini_cen24
finconfigura24:
            rjmp finconfigura
ini_cen24:
            ldi r16,24
            sts temperatura,r16
            rcall funcion
            rcall mensaje_24
inicen24:
            rcall leyendo
            lds r16, presiono
            cpi r16, 2
            breq inierror1
            cpi r16, 1
```

```
        breq ini_cen28
        cpi r16, 3
        breq finconfigura24
        rjmp inicen24
finconfigura28:
        rjmp finconfigura
ini_cen28:
        ldi r16,28
        sts temperatura,r16
        rcall funcion
        rcall mensaje_28
inicen28:
        rcall leyendo
        lds r16, presiono
        cpi r16, 2
        breq ini_cen24
        cpi r16, 1
        breq ini_cen32
        cpi r16, 3
        breq finconfigura28
        rjmp inicen28
finconfigura32:
        rjmp finconfigura
ini_cen32:
        ldi r16,32
        sts temperatura,r16
        rcall funcion
        rcall mensaje_32
inicen32:
        rcall leyendo
        lds r16, presiono
        cpi r16, 2
        breq ini_cen28
        cpi r16, 1
        breq ini_cen36
        cpi r16, 3
        breq finconfigura32
        rjmp inicen32
finconfigura36:
        rjmp finconfigura
ini_cen36:
        ldi r16,36
        sts temperatura,r16
        rcall funcion
        rcall mensaje_36
inicen36:
        rcall leyendo

lds r16, presiono
        cpi r16, 2
        breq ini_cen32
        cpi r16, 1
        breq ini_cen40
        cpi r16, 3
        breq finconfigura36
        rjmp inicen36
finconfigura40:
        rjmp finconfigura
ini_cen40:
        ldi r16,40
```

```
        sts temperatura,r16
        rcall funcion
        rcall mensaje_40
inicen40:
        rcall leyendo
        lds r16, presiono
        cpi r16, 2
        breq ini_cen36
        cpi r16, 1
        breq ini_cen44
        cpi r16, 3
        breq finconfigura40
        rjmp inicen40
finconfigura44:
        rjmp finconfigura
ini_cen44:
        ldi r16,44
        sts temperatura,r16
        rcall funcion
        rcall mensaje_44
inicen44:
        rcall leyendo
        lds r16, presiono
        cpi r16, 2
        breq ini_cen40
        cpi r16, 1
        breq ini_cen48
        cpi r16, 3
        breq finconfigura44
        rjmp inicen44
finconfigura48:
        rjmp finconfigura
ini_cen48:
        ldi r16,48
        sts temperatura,r16
        rcall funcion
        rcall mensaje_48
inicen48:
        rcall leyendo
        lds r16, presiono
        cpi r16, 2
        breq ini_cen44
        cpi r16, 1
        breq inierror2
        cpi r16, 3
        breq finconfigura48
        rjmp inicen48
inierror2:
        rcall mensaje_10
        rcall r_1s
        rcall r_1s
        rjmp ini_cen48
finconfigura:
        rcall mensaje_7
        rcall r_1s
        rcall r_1s
        rcall guardar
        rcall r_1s
        rcall r_1s
        pop r16
```

```
        out sreg, r16
        pop r16
        ret
; ********************************************************************
; SUBRUTINA QUE SELECCIONA LA TEMPERATURA DESEADA
; ********************************************************************
seleccion:
        push r16
        in r16, sreg
        push  r16
        rcall Verifica_EEPROM
        lds r16,verificar
        cpi r16, 0
        breq nopuede
        rcall mensaje_8
        rcall r_1s
        rjmp elecc1
nopuede:
        rcall mensaje_16
        rcall r_1s
        rcall r_1s
        rcall configuracion
        rjmp sal2
elecc1:
        ldi r16,5
        sts cont_lee,r16
        rcall leeEEPROM
        sts cantguardada,r20
inielecc1:
        rcall leyendo
        lds r16,presiono
        cpi r16,4
        breq obtiene1
        cpi r16,5
        breq obtiene2
        cpi r16,6
        breq obtiene3
        cpi r16,7
        breq obtiene4
        rjmp elecc1
obtiene1:
        lds r16,cantguardada
        cpi r16,1
        brlo inielecc1
        ldi r16,1
        sts cont_lee,r16
        rcall leeEEPROM
        sts temperatura, r20
        rjmp sal2
obtiene2:
        lds r16,cantguardada
        cpi r16,2
        brlo inielecc1
        ldi r16,2
        sts cont_lee, r16
        rcall leeEEPROM
        sts temperatura, r20
        rjmp sal2
obtiene3:
        lds r16,cantguardada
```

```
            cpi r16,3
            brlo inielecc1
            ldi r16,3
            sts cont_lee, r16
            rcall leeEEPROM
            sts temperatura, r20
            rjmp sal2
obtiene4:
            lds r16,cantguardada
            cpi r16,4
            brlo inielecc1
            ldi r16,4
            sts cont_lee,r16
            rcall leeEEPROM
            sts temperatura, r20
sal2:
            pop r16
            out sreg, r16
            pop r16
            ret
; ********************************************************************
; SUBRUTINA QUE VERIFICA SI ES LA PRIMERA VEZ DEL FUNCIONAMIENTO
; ********************************************************************
Verifica_EEPROM:
            ldi yh, high(0)
            ldi yl, low (0)
            rcall leeEEPROM1
            cpi r20,$FF
            breq verificar_1
            rjmp verificar_0
verificar_1:
            ldi r16, 0
            sts verificar,r16
            rjmp finVerifica_EEPROM
verificar_0:
            ldi r16,1
            sts verificar,r16
            rjmp finVerifica_EEPROM
finVerifica_EEPROM:
            ret
;*********************************************************************
; SUBRUTINA QUE REALIZA LA FUNCIÓN DE CORRESPONDENCIA: TEMPERATURA-
FLUJO
;*********************************************************************
funcion:
            push r16
            in r16, sreg
            push  r16
            push r17
            push r18
            rcall re_temp
            rcall verifica_temp
            lds r16, verificador
            cpi r16, 2
            breq  Esperando
            rjmp sin_esperar
esperando:
            rcall mensaje_19
            rcall r_1s
            rcall r_1s
```

```
        rjmp fin4
sin_esperar:
        lds r16,temperatura
        sts var1,r16
        ldi r16, 24
        sts Tfria, r16
        ldi r16, 60
        sts Tcaliente, r16
        lds r16,Tfria
        lds r17,Tcaliente
        lds r18,temperatura
        cp r18,r16
        breq casoesp1
        cp r18,r17
        breq casoesp2
        rjmp ok
casoesp1:
        ldi r16,10
        sts Ffria,r16
        ldi r16,0
        sts Fcaliente,r16
        rjmp fin3
casoesp2:
        ldi r16,10
        sts Fcaliente,r16
        ldi r16,0
        sts Ffria,r16
        rjmp fin3
ok:
        lds r17,temperatura
        sts var1,r17
        lds r16,Tcaliente
        sts var2,r16
        sub r16,r17
        sts Tcaliente,r16
        lds r16,temperatura
        lds r17,tfria
        sub r16,r17
        sts temperatura, r16
        lds r16,tfria
        lds r17,var2
        add r16, r17
        ror r16
        brcs impar
        rjmp compara
impar:
        ldi r17, 1
        add r16, r17
        rjmp compara
compara:
        lds r17, var1
        sub r16, r17
        brsh mmenor
        brlo mmayor
mmenor:
        ldi r17,10
        lds r16, temperatura
        mul r16,r17
        sts temperatura,r0
        rcall divideTf
```

```
        rjmp fin1
mmayor:
        ldi r17,10
        lds r16,tcaliente
        mul r16, r17
        sts tcaliente,r0
        rcall divideTc
        rjmp fin1
fin1:
        cpi rf, 10
        brsh fijo101
        sts Ffria,rf
        rjmp fin2
fijo101:
        ldi r16,10
        sts Ffria,r16
        rjmp fin2
fin2:
        cpi rc, 10
        brsh fijo102
        sts Fcaliente,rc
        rjmp fin3
fijo102:
        ldi r16,10
        sts Fcaliente,r16
fin3:
        rcall asignasalida
        lds r16,var1
        sts temperatura,r16
fin4:
        pop r18
        pop r17
        pop r16
        out sreg, r16
        pop r16
        ret
; ********************************************************************
; SUBRUTINA QUE ASIGNA LA APERTURA DE LAS VALVULAS
; ********************************************************************
asignasalida:
        push r16
        in r16, sreg
        push  r16
        push r17
        push r18
        lds r16, Ffria
        rcall asigna
        lsl r16
        lsl r16
        lsl r16
        lsl r16
        andi r16, 0b11110000
        mov r17,r16
        lds r16, Fcaliente
        rcall asigna
        andi r16, 0b00001111
        add r17,r16
        out portd, r17
        pop r18
        pop r17
```

```
        pop r16
        out sreg, r16
        pop r16
        ret
; ***************************************************************
; SUBRUTINA QUE ASIGNA LA APERTURA DE LAS VALVULAS
; ***************************************************************
asigna:
        cpi r16, 0
        breq abre0
        cpi r16, 1
        breq abre1
        cpi r16, 2
        breq abre2
        cpi r16, 3
        breq abre3
        cpi r16, 4
        breq abre4
        cpi r16, 5
        breq abre5
        cpi r16, 6
        breq abre6
        cpi r16, 7
        breq abre7
        cpi r16, 8
        breq abre8
        cpi r16, 9
        breq abre9
        cpi r16, 10
        breq abre10
        rjmp asigna
abre0:
        ldi r16,0b00000000
        rjmp salcaso
abre1:
        ldi r16,0b00000001
        rjmp salcaso
abre2:
        ldi r16,0b00000011
        rjmp salcaso
abre3:
        ldi r16,0b00000100
        rjmp salcaso
abre4:
        ldi r16,0b00000110
        rjmp salcaso
abre5:
        ldi r16,0b00000111
        rjmp salcaso
abre6:
        ldi r16,0b00001001
        rjmp salcaso
abre7:
        ldi r16,0b00001010
        rjmp salcaso
abre8:
        ldi r16,0b00001100
        rjmp salcaso
abre9:
        ldi r16,0b00001101
```

```
        rjmp salcaso
abre10:
        ldi r16,0b00001111
        rjmp salcaso
salcaso:
        ret
; ***************************************************************
; SUBRUTINA QUE DIVIDE LA TEMP CALIENTE ENTRE LA TEMP DESEADA
; ***************************************************************
divideTc:
        push r16
        push r17
        lds r16,tcaliente
        lds r17,temperatura
        clr rf
        clr rc
divideTc1:
        sub r16,r17
        brlo finTc
        inc rf
        ldi rc,10
        rjmp divideTc1
finTc:
        lsr r16
        add r17,r16
        brsh fintotal2
        inc rf
fintotal2:
        pop r17
        pop r16
        ret
; ***************************************************************
; SUBRUTINA QUE DIVIDE LA TEMP DESEADA ENTRE LA TEMP CALIENTE
; ***************************************************************
divideTf:
        push r16
        push r17
        clr rc
        clr rf
        lds r16,temperatura
        lds r17,Tcaliente
divideTf1:
        sub r16,r17
        brlo finTf
        inc rc
        ldi rf,10
        rjmp divideTf1
finTf:
        lsr r17
        add r16,r17
        brsh fintotal1
        inc rc
fintotal1:
        pop r17
        pop r16
        ret
; ***************************************************************
; SUBRUTINA QUE RECIBE LA TEMPERATURA
; ***************************************************************
re_temp:
```

```
        push r16
        push r17
        push r18
        rcall recibeADC0
        rcall recibeADC5
        lds r16,tfria
        ldi r17,255
        ldi r18,100
iniadcf:
        cp r16,r17
        brsh muestraf
        subi r17,2
        dec r18
        cp r16,r17
        brsh muestraf
        subi r17,3
        dec r18
        rjmp iniadcf
muestraf:
        sts tfria,r18
        lds r16,tcaliente
        ldi r17,255
        ldi r18,100
iniadcc:
        cp r16,r17
        brsh muestrac
        subi r17,2
        dec r18
        cp r16,r17
        brsh muestrac
        subi r17,3
        dec r18
        rjmp iniadcc
muestrac:
        sts tcaliente,r18
        pop r18
        pop r17
        pop r16
        ret
; ********************************************************************
; SUBRUTINA QUE VERIFICA LA TEMPERATURA
; ********************************************************************
verifica_temp:
        push r20
        push r19
        push r18
        push r17
        push r16
        ldi r19,0
        ldi r20,0
        lds r16,tfria
        lds r17,tcaliente
        lds r18,temperatura
        cp r16,r18
        brsh fin_verifica_temp_fria
        ldi r19,1
fin_verifica_temp_fria:
        cp r17,r18
        brsh fin_verifica_temp_caliente
        ldi r20,1
```

```
fin_verifica_temp_caliente:
        add r19,r20
        sts verificador,r19
        pop r16
        pop r17
        pop r18
        pop r19
        pop r20
        ret
; ***************************************************************
; SUBRUTINA QUE CONFIGURA ADC0
; ***************************************************************
ConfiguraADC0:
        push r16
        ldi r16, 0b01100000
        out ADMUX, r16
        ldi r16, 0b11010100
        out ADCSr, r16
        pop r16
        ret
; ***************************************************************
; SUBRUTINA QUE CONFIGURA ADC5
; ***************************************************************
ConfiguraADC5:
        push r16
        ldi r16, 0b01100101
        out ADMUX, r16
        ldi r16, 0b11010100
        out ADCSr, r16
        pop r16
        ret
; ***************************************************************
; SUBRUTINA QUE LEE Y ALMACENA ADC0
; ***************************************************************
recibeADC0:
        push r16
        rcall ConfiguraADC0
sigueADC0:
        sbis ADCSr,ADIF
        rjmp sigueADC0
        in r16, ADCH
        sts tfria, r16
        cbi ADCSr, 7
        pop r16
        ret
; ***************************************************************
; SUBRUTINA QUE LEE Y ALMACENA ADC5
; ***************************************************************
recibeADC5:
        push r16
        rcall ConfiguraADC5
sigueADC5:
        sbis ADCSr,ADIF
        rjmp sigueADC5
        in r16, ADCH
        sts tcaliente, r16
        cbi ADCSr, 7
        pop r16
        ret
```

```
; *****************************************************
; MENSAJES
; *****************************************************
Mensaje1:
        .db "Bienvenido! Elija la",'*',"opcion a realizar",0
Mensaje2:
        .db "Configurar pulse +",'*',"Iniciar pulse -",0
Mensaje3:
        .db "Pulse + o - hasta la",'*',"temperatura deseada",0
Mensaje4:
        .db "Pulse fin despues",'*',"de configurar",0
Mensaje5:
        .db "Seleccione el numero",'*',"de usuario",0
Mensaje6:
        .db "Pulse fin para",'*',"cerrar la ducha",0
Mensaje7:
        .db "Guardando...",0
Mensaje8:
        .db "Seleccione el N° de",'*',"usuario",0
Mensaje9:
        .db "No puede seguir",'*',"disminuyendo",0
Mensaje10:
        .db "No puede seguir",'*',"aumentando",0
Mensaje12:
        .db "Se ha guardado en el",'*',"usuario 1",0
Mensaje13:
        .db "Se ha guardado en el",'*',"usuario 2",0
Mensaje14:
        .db "Se ha guardado en el",'*',"usuario 3",0
Mensaje15:
        .db "Se ha guardado en el",'*',"usuario 4",0
Mensaje16:
        .db "Primero debe",'*',"configurar",0
Mensaje17:
        .db "Apague el  equipo",0
Mensaje18:
        .db "Apague la terma",'*',"temperatura muy alta",0
Mensaje19:
        .db "Encienda la terma o",'*',"baje la temperatura",0
Mensaje20:
        .db "Esperando...",0
Mensaje21:
        .db "Todos los usuarios",'*',"estan siendo usados",0
Mensaje22:
        .db "Pulse el numero de",'*',"usuario a reemplazar",0
Mensaje24:
        .db "24°C",0
Mensaje28:
        .db "28°C",0
Mensaje32:
        .db "32°C",0
Mensaje36:
        .db "36°C",0
Mensaje40:
        .db "40°C",0
Mensaje44:
        .db "44°C",0
Mensaje48:
        .db "48°C",0
```