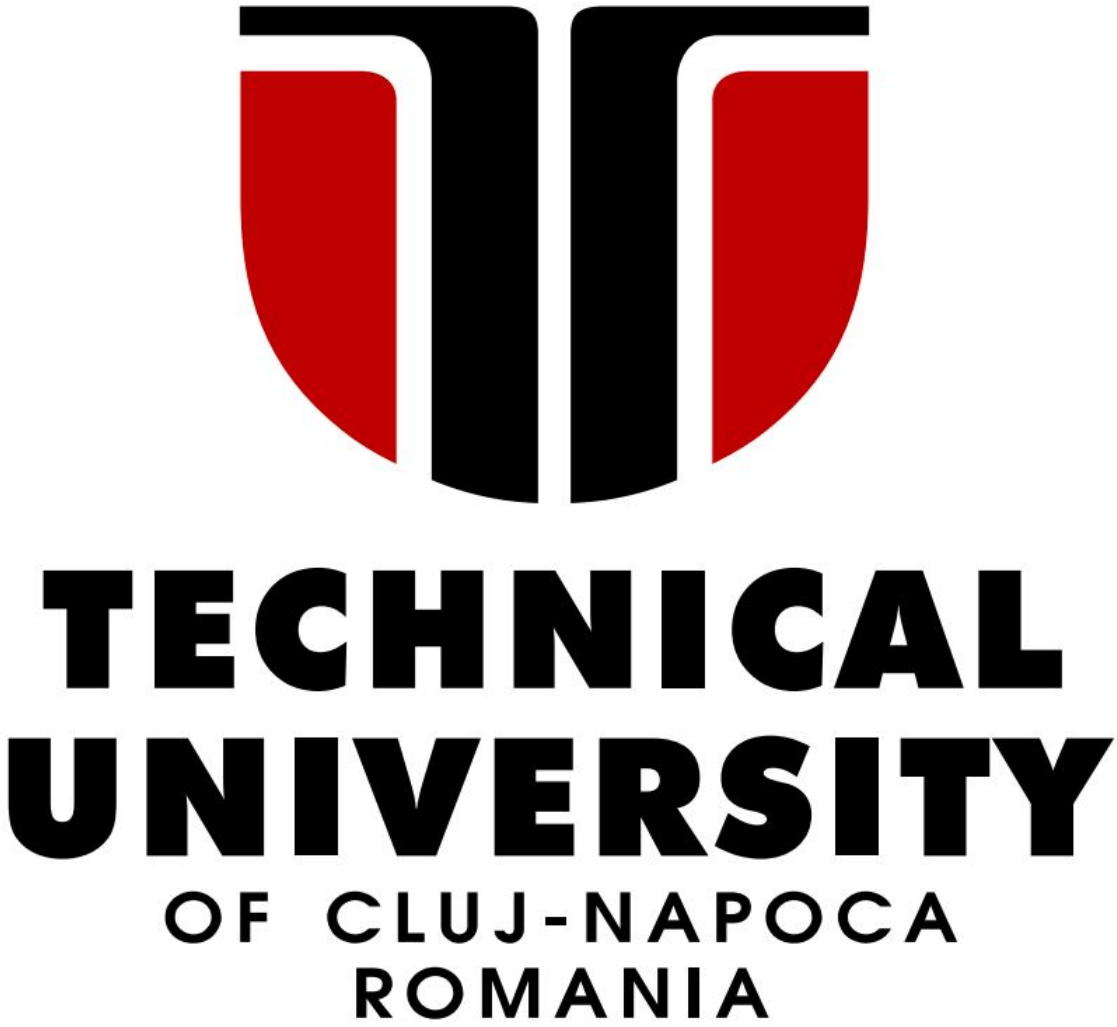
Microcontrollers

-Room thermostat-

Student: Bogdan Meseșan

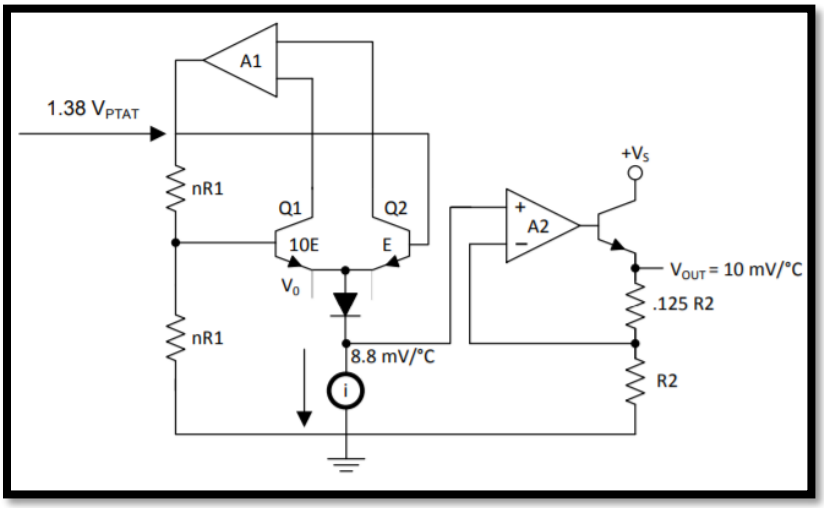
Coordinating teacher: Andreea Ignat



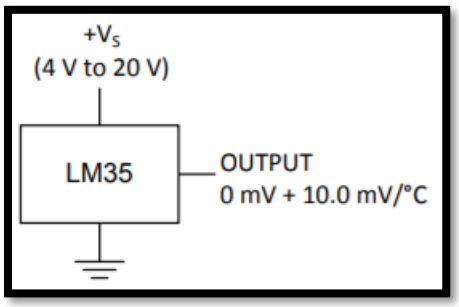
Contents

1. The temperature sensor

The temperature sensor I have chose is the LM35, which is a semiconductor sensor which comes in the form of an integrated circuit. It is a voltage output temperature sensor, which has two identical diodes with temperature-sensitive voltage vs. current characteristics that can be used to monitor changes in temperature. A linear response and a low impedance are offered at the output and it draw only 60μA from the supply, which limits its power dissipation. The operating range od the sensor is rated between −55°C and 150°C. The temperature-sensing element is comprised of a delta-V BE architecture. Its accuracy is of ±0.5°C at 25°C and of ±1° for a higher temperature range.



The temperature-sensing element is the buffered by a class A amplifier, then the voltage is provided to the output pin, with a 0.5Ω output impedance. The transfer function of the sensor is provided by the following equation: VOUT = 10 mv/°C × T.



In the following application, the supply voltage for the sensor will be +5Vdc, and the output will then be amplified and sent to an Analog to Digital converter, having the same +5V reference voltage. Finally, the information from the sensor will be processed by the microprocessor.

2 The ADC stage

For the current application, I have chosen to use the ADC 0808, produced by Texas Instruments. The ADC uses successive approximation as the conversion technique, which means that is has a 256R voltage divider and a SAR register. It has 8 channels which can be accessed via an 8-channel multiplexer.

Key Specifications:

Resolution: 8 Bits; Total Error: ±1/2 LSB; Conversion Time: 100 µ; $V_{FS=5V}$

Because the ADC is on 8bits, then VLSB is: $V_{LSB} = \frac{5V}{2^8-1} = 19.6\,mV$.

The application will measure temperatures between 0 and 50°C, which means that at 0°C I expect an output of logic 0000 0000b, and at 50°C I expect 1111 1111b. The error will be of ±1 LSB.

ADC pins:

The explanation of the pins:

-IN0, … , IN7 – represent the 8 different input channels of the ADC; only one channel is active at a given time. The selection of the channel is made via the address line decoder.

-ADD_A, ADD_B, ADD_C – address lines used to select a particular channel.

Table 1. Analog Channel Selection

SELECTED ANALOG CHANNEL	ADDRESS LINE		
	C	B	A
IN0	L	L	L
IN1	L	L	H
IN2	L	H	L
IN3	L	H	H
IN4	H	L	L
IN5	H	L	H
IN6	H	H	L
IN7	H	H	H

-ALE – Address Latch Enable is used enable the selected channel

-Vref(+) ,Vref(-) – Reference voltage levels; for our system: Vref(+) = 5V, Vref(-) = GND;

-Clock – the typical clock frequency for this ADC is 640kHz, for minimum conversion time

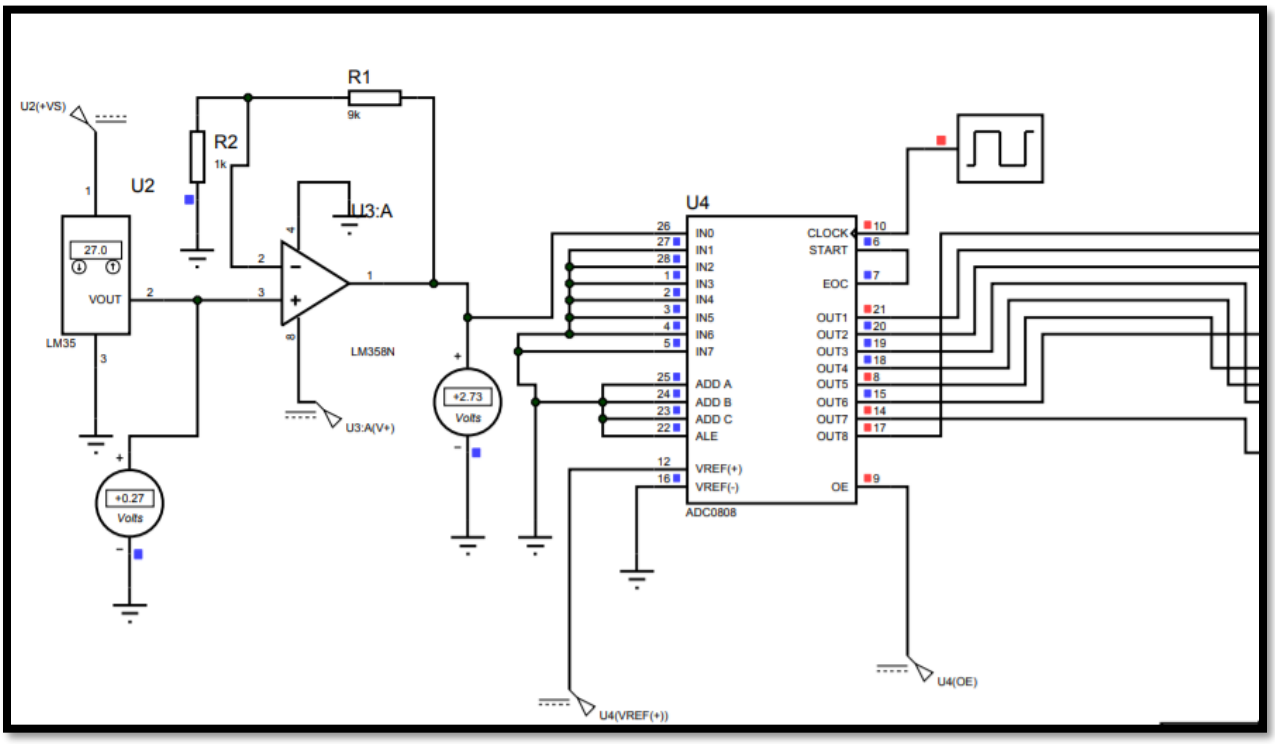
-Start, EOC – The ADC's SAR is reset on the positive edge of the Start Conversion pulse. A conversion in progress can be interrupted by the pulse received from a new conversion. In order to have a continuous conversion, the End of Conversion pin will be connected to the Start pin.

-Out1, … , Out8 – The digital output of the ADC

-OE – Output enable, must be powered to 5V

I will use a noninverting amplifier, with amplification A = 10 to amplify the output of the analog sensor: $A = 1 + \frac{R_F}{R_G} = 10$. I have chosen the LM358N operational amplifier, with a positive supply of 5V and a negative supply of 0V.

The circuit of the ADC stage:



In the software part of the project, a lookup table will be designed in order to process the output of the ADC in a correct manner. For every temperature level, the processor will interpret the information at its input via a filtering routine.

3. LCD configuration

For the current application I have chosen to use the LM016L LCD, which has a 16-character x 2-line display and a 5x8 dot cursor. Pins:

VDD – Power supply pin with a typical power supply of 5V

VSS - Ground

VEE – Contrast adjustment

RS – 0: Instruction code input; 1: Data Input

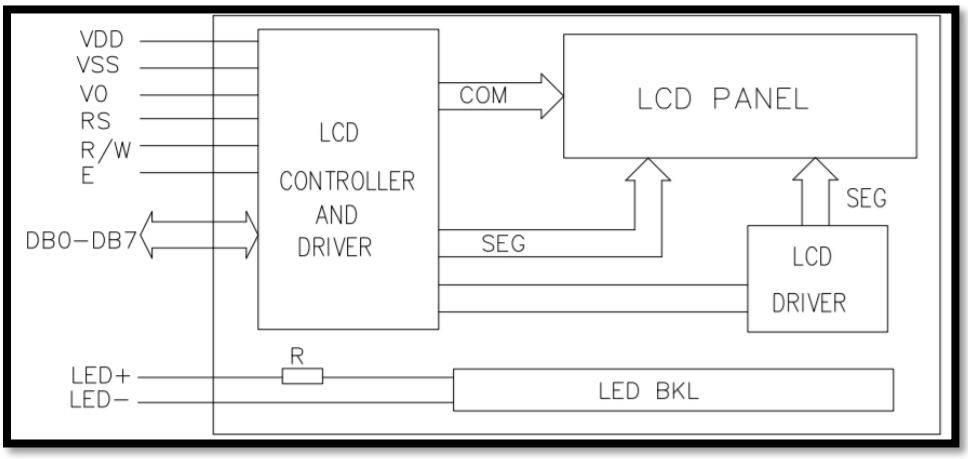
R/W – 0: Data Write; 1: Data Read

E – Operation Enable Signal

DB0 – DB3 – low order bi-directional data bus lines, not used during 4-bit operation

DB4 – DB7 – high order data bus lines, which are used during 4-bit operation

Block diagram:



The LCD module has two types of interfaces: 4-bit interface and 8-bit interface, depending on the microprocessor have for our application.

Address counter (AC):

Stores the current address of the DDRAM/CGRAM. After writing from DDRAM/CGRAM, the AC is automatically increased by 1; when a read operation is done from the DDRAM/CGRAM the AC is decreased by 1.

Display Data RAM (DDRAM):

DDRAM stores display data of maximum 80x8 bits. The position of the next character to be written is set via DDRAM.

Display position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DDRAM address	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
DDRAM address	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

Character Generator RAM (CGRAM):

By writing certain data to the CGRAM, user defined characters can be used.

Character Generator ROM (CGROM):

CGROM has 204 characters with 5 x 8 dot patterns and 32 characters with 5 x 10 dot patterns.

Instructions:

Instruction	Instruction code										Description	Execution time (fosc= 270 KHZ
	RS	R/W	DB7	DB6	DB 5	DB4	DB3	DB2	DB 1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Write "20H" to DDRA and set DDRAM address to "00H" from AC	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	-	Set DDRAM address to "00H" From AC and return cursor to Its original position if shifted. The contents of DDRAM are not changed.	1.53ms
Entry mode Set	0	0	0	0	0	0	0	1	I/D	SH	Assign cursor moving direction And blinking of entire display	39us
Display ON/ OFF control	0	0	0	0	0	0	1	D	C	B	Set display (D), cursor (C), and Blinking of cursor (B) on/off Control bit.	
Cursor or Display shift	0	0	0	0	0	1	S/C	R/L	-	-	Set cursor moving and display Shift control bit, and the Direction, without changing of DDRAM data.	39us
Function set	0	0	0	0	1	DL	N	F	-	-	Set interface data length (DL: 8- Bit/4-bit), numbers of display Line (N: =2-line/1-line) and, Display font type (F: 5x11/5x8)	39us
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address Counter.	39us
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address Counter.	39us
Read busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Whether during internal Operation or not can be known By reading BF. The contents of Address counter can also be read.	0us
Write data to Address	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43us
Read data From RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43us

Content:

Clear Display-writes 20h to al DDRAM addresses and resets the AC to 00h

Return home – resets the cursor to its initial position and sets the DDRAM address to 00h

Entry mode set – sets the moving directions of the cursor and display; I/D = 1 => DDRAM address incremented; I/D = 0 => DDRAM address decremented; SH = 1 => shift of entire display;

Display On/Off – D -> 0: display turned off, but data remains in DDRAM; 1: display turned on;

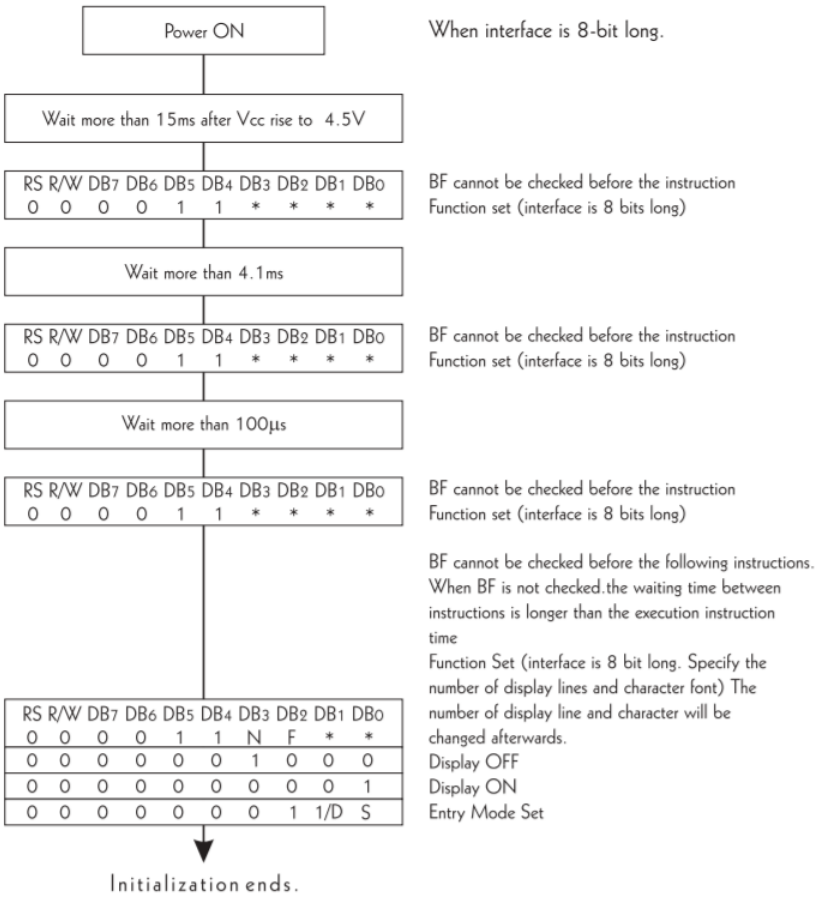
C-> 0: cursor turned off, but I/D preserves data; 1: cursor turned on; B-> 0: cursor blink off; 1: cursor blink on

Function set – DL -> 0: 4-bit interface; 1: 8-bit interface; N-> 0: use only 1 line; 1: use line 1 and line 2; F-> 0: 5x8 dots display format; 1: 5x11 display format

Standard character pattern:

8-bit interface:

Initialization:



The sequence I use to initialize the Display:

-instruction1: function set

-instruction2: display off

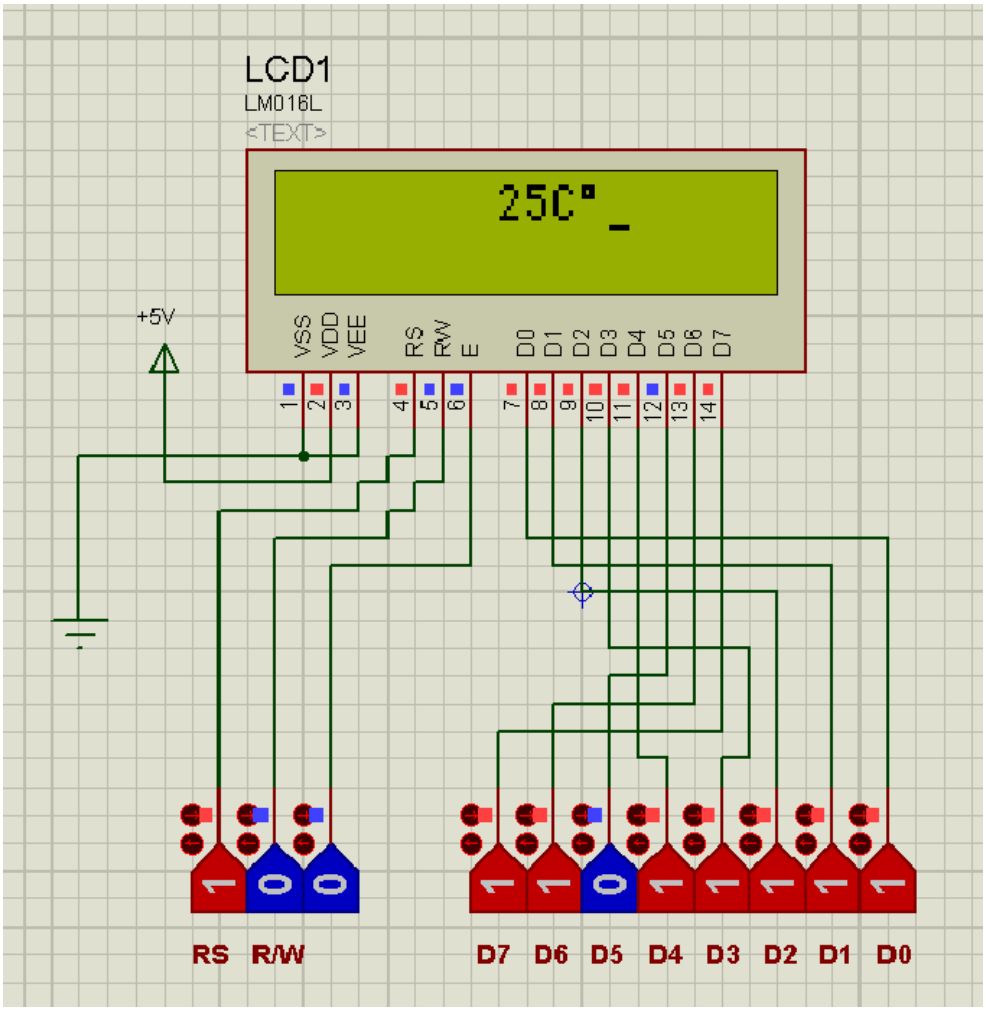
-instruction3: display on

-instruction4: display on/off control

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	1	1	1

Display:

RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
0	0	1	0	0	0	0	1	1	1
1	0	0	0	1	1	0	0	1	0
1	0	0	0	1	1	0	1	0	1
1	0	0	1	0	0	0	0	1	1
1	0	1	1	0	1	1	1	1	1



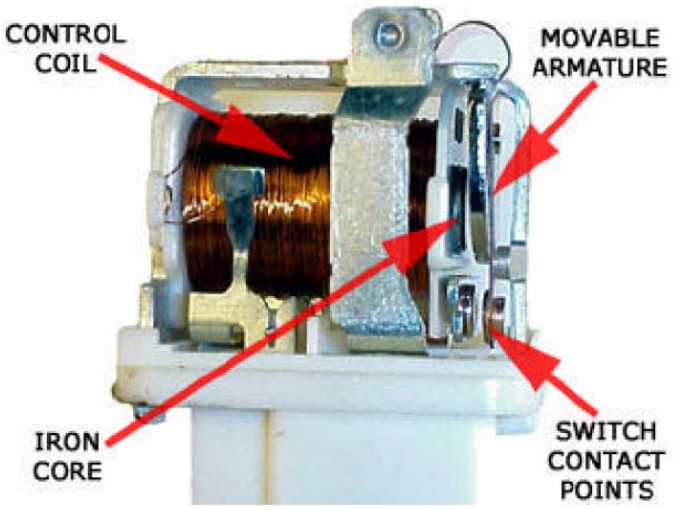
4. The relay stage

The operating principle of a relay

A relay is an electromagnetic switch that is used to turn on and off via a low power signal. It is operated both electrically and mechanically because is uses an electromagnet and a set of contacts is the switching mechanism. One common application of the relay is in systems where one signal is used to control multiple circuits.

A typical relay is constituted of:

- Electromagnet – an electric coil, which acts as the sensing unit and can be powered either by AC or DC current
- Movable Armature - operates either to close the open contacts or to open the closed contacts
- Switch point contacts - connected to the load and controlled by the coil in the control circuit
- Spring - used so as to produce an air gap in the circuit when the relay becomes de-energized

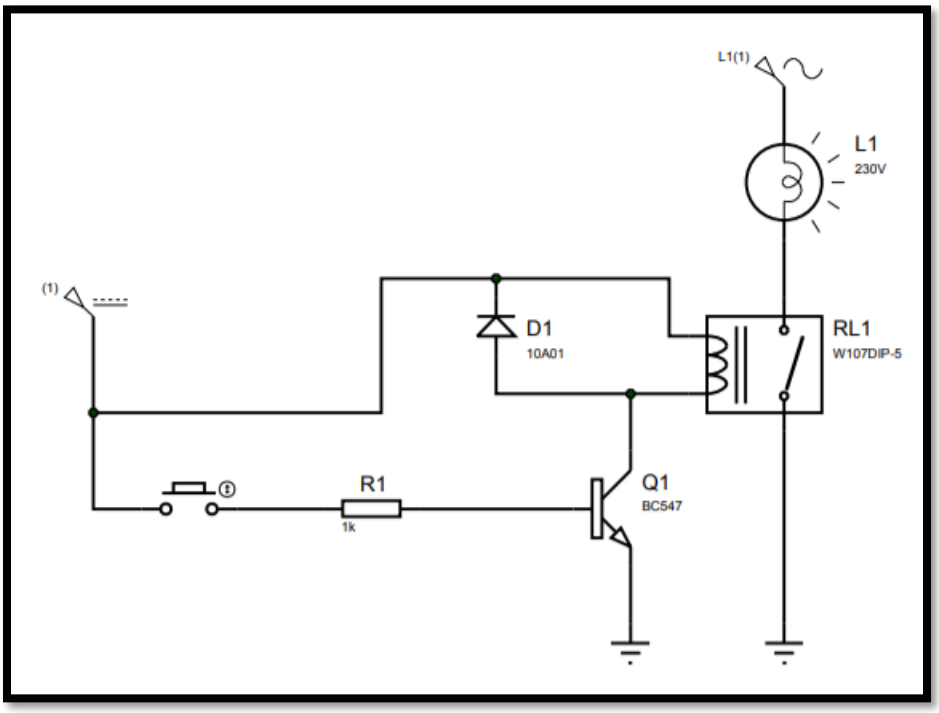


How does a relay work? Current starts flowing through the coil until it exceeds a certain threshold value. The electromagnet starts energizing and intensifies the magnetic field. The upper contact arm starts to be attracted to the lower part until they form a short circuit between the power and the load. When the current is shut off, the movable armature is returned by a force to its initial position. This force is provided by two factors: the spring and gravity.

A relay can be used either in low-voltage applications, where they reduce the noise of the whole circuit, or in high-voltage applications, where they provide protection. In our temperature control system, we will use the relay in On/Off operation.

The command circuit:

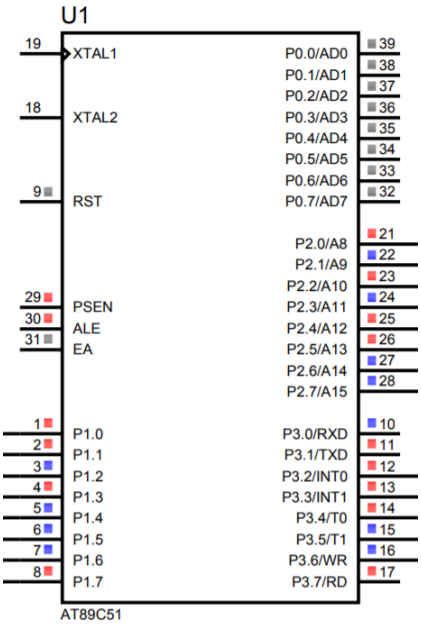
The main problem with relay coils is that they have highly inductive loads. When current flows, a self-induced magnetic field is generated around the coil. When the current goes off, a large electromotive force voltage is produced as the magnetic flux collapses within the coil. This voltage may very high in comparison to the switching voltage and may damage any semiconductor device, like a transistor.



To prevent this effect, connect a reverse biased diode across the relay coil. The EMF voltage will turn the diode on, which in turn, will dissipate the energy stored in the relay. This diode is called Fly-Back diode. When the relay is on, the bipolar transistor will start to conduct, effectively linking the relay to ground. The 1k resistor in the base of the transistor is used to limit the base current. The relay will be commanded from the microcontroller, when the measured temperature is lower than the reference temperature.

5. The microcontroller

The component that I have chosen for my application is the AT89C51, which does not have an internal ADC. Therefore, I will use the ADC0808 as an external ADC.



The microcontroller has 4 ports:

-Port 0 -> Open-drain, 8-bit bi-directional I/O port

-Port 1 -> 8-bit bi-directional I/O port with internal pull-ups; I will use this port to read data sent from the ADC

-Port 2 ->8-bit bi-directional I/O port with internal pull-ups;

I will use P2.5, P2.6 and P2.7 to control the RS, R/W and EN pins of the LCD

-Port 3-> 8-bit bi-directional I/O port with internal pull-ups; I will use Port 3 to send instructions to the LCD controller

Utility of the Ports:

-Port 1: Input port, connected to the output of the ADC, tasked with receiving temperature information in digital form.

-Port 2: Output port, connected to the Data lines of the LCD, tasked with sending instructions to the Display

-Port 3: Output port:

P3.0-Relay control;

P3.2 – "Button+" external interrupt 0;

P3.3 - "Button-" external interrupt 1;

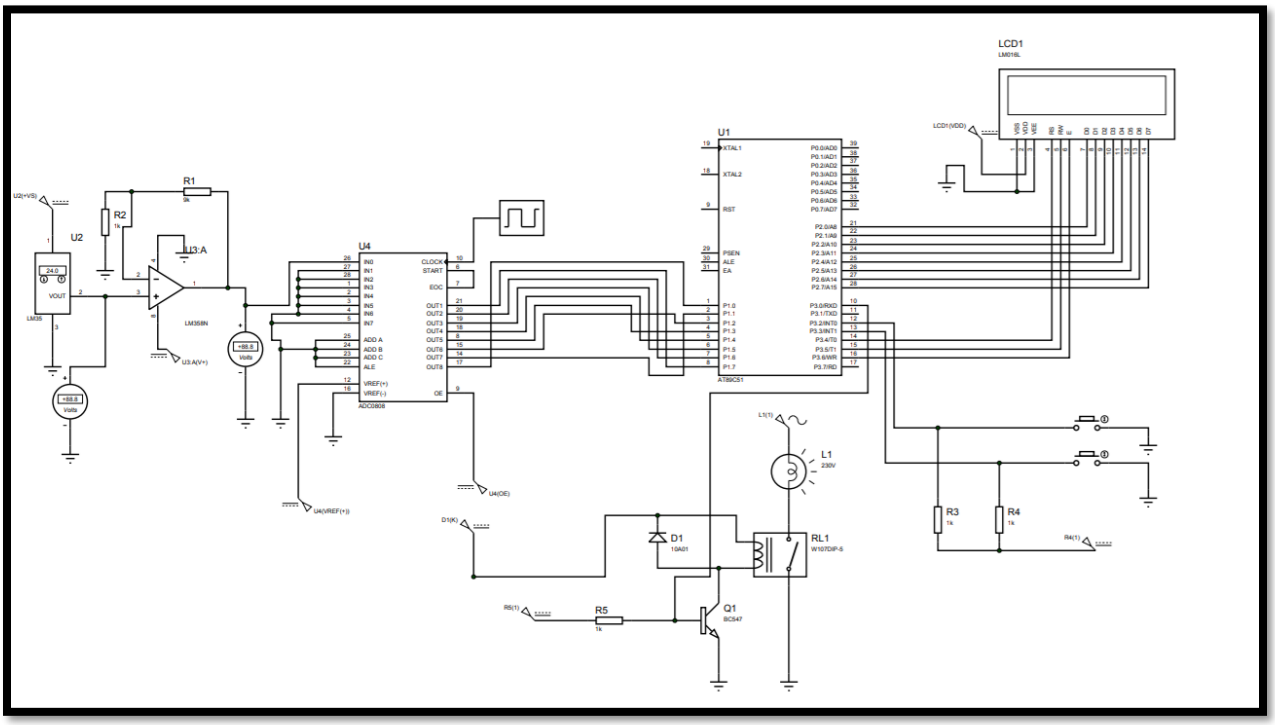
P3.4 – LCD RS;

P3.5 – LCD R/W;

P3.6 – LCD Enable;

6. Software implementation

The final circuit:



6.1. The ASM program

Main:

```
$title(Main_Project5)

MainP segment Code

Extrn code (InitLCD);
Extrn code (ADCread);
Extrn code (NrDisplay);
Extrn code (delay_1ms);
;Extrn code (delay_70ms);
Extrn code (Test_comparare);
org 0000h;
ljmp start;bypass interrupt

org 03h;
inc R7;
reti;

org 13h;
dec R7;
reti;


RSEG MainP;
org 30h;
start:
mov P1, #0FFh;set P1 as input port for the adc
mov Ie, #10000101b; enable external int 1
setb tcon.2;activeaza itnreruperea pe falling edge
setb tcon.0;activeaza itnreruperea pe falling edge
mov R7, #25;
acall InitLCD;
mainloop:
acall ADCread;
acall NrDisplay;
acall Test_comparare;
sjmp mainloop;
jmp $;
end;
```

Delay_1ms:

delay_1ms.a51

```
1  Name RutinaDelay1ms
2  Public delay_1ms
3
4  DelaySeg Segment code
5  RSEG DelaySeg
6  delay_1ms:
7  mov tmod, #00000001b;
8  mov th0, #0FCh;
9  mov tl0, #66h;
10 setb TR0;
11 stau:
12 jnb TF0,stau;
13 clr TF0;
14 clr TR0;
15
16 ret;
17 end;
```

Send instruction:

Sendinstr.a51

```
1  Name RutinaSendInstr;
2  Public sendInstr;
3  SendInstrSeg Segment Code;
4  RSEG SendInstrSeg;
5  Extrn Code (delay_1ms);
6  sendInstr:
7
8  setb P3.6;
9  acall delay_1ms;
10 cpl P3.6;
11 acall delay_1ms;
12
13 ret;
14 end;
```

Init Lcd:

InitLCD.a51

```
1  Name RutinaInitLCD;
2  Public InitLCD;
3
4  instr1 equ 00111000b;function set
5  instr2 equ 00001000b;display off
6  instr3 equ 00000001b;display on
7  instr4 equ 00001100b;display on/off control
8
9  LCDSeg Segment code
10 RSEG LCDSeg;
11 Extrn Code (sendInstr);
12 Extrn Code (TempStringDisp);
13 InitLCD:
14
15 clr P3.4;RS
16 clr P3.5;R/W
17 clr P3.6;ENABLE
18 mov P2, 0h;
19
20 mov P2, #instr1;
21 acall sendInstr;
22
23 mov P2, #instr2;
24 acall sendInstr;
25
26 mov P2, #instr3;
27 acall sendInstr;
28
29 mov P2, #instr4;
30 acall sendInstr;
31
32 ;Display string 'Temperature:'
33 acall TempStringDisp;
34 ret;
35 end;
```

Temperature_string display:

TempStringDisp.a51

```
Name RutinaTempStringDisp;
Public TempStringDisp;

instrT equ 01010100b;
instrE equ 01000101b;
instrM equ 01001101b;
instrP equ 01010000b;
instrR equ 01010010b;
instrA equ 01000001b;
instrU equ 01010101b;
instr2P equ 00111010b;
instrF equ 01000110b;

TempStringDispSeg Segment code
RSEG TempStringDispSeg;
Extrn Code (sendInstr);

TempStringDisp:

;RS = 1, deoarece trimitem caractere pentru display
setb P3.4;
mov P2, #instrT;
acall sendInstr;

mov P2, #instrE;
acall sendInstr;

mov P2, #instrM;
acall sendInstr;

mov P2, #instrP;
acall sendInstr;

mov P2, #instrE;
acall sendInstr;

mov P2, #instrR;
acall sendInstr;

mov P2, #instrA;
acall sendInstr;

mov P2, #instrT;
acall sendInstr;

mov P2, #instrU;
acall sendInstr;

mov P2, #instrR;
acall sendInstr;

mov P2, #instrE;
acall sendInstr;

mov P2, #instr2P;
acall sendInstr;

;Rs = 0 deoarece urmeaza sa setam pozitia
clr P3.4;

mov P2, #11000000b;setez pozitia corecta
acall sendInstr;

setb P3.4;

mov P2, #instrT;
acall sendInstr;

mov P2, #instrR;
acall sendInstr;

mov P2, #instrE;
acall sendInstr;

mov P2, #instrF;
acall sendInstr;

mov P2, #instr2P;
acall sendInstr;

ret;
end;
```

Test_compare.a51

```
Name RutinaTest_comparare
Public Test_comparare

Test_comparareSeg Segment code
RSEG Test_comparareSeg

Test_comparare:
mov a, R7;Referinta

loop_cmp_releu:
djnz R7, dec_Adc
sjmp opreste_releu;
dec_Adc:
djnz R6, loop_cmp_releu;
sjmp porneste_releu;

porneste_releu:
setb P3.0;
sjmp stop_comparare_releu;
opreste_releu:
clr P3.0;

stop_comparare_releu:
mov R7, a;
ret;
end;
```

ADCread.a51*

```
2  Public ADCread;
3  Cseg at 150h
4      my0: db 00000001b;1
5      my1: db 00000110b;6
6      my2: db 00001011b;11
7      my3: db 00010000b;16
8      my4: db 00010110b;!!22
9      my5: db 00011011b;27
10     my6: db 00100000b;32
11     my7: db 00100101b;37
12     my8: db 00101010b;42
13     my9: db 00101111b;47
14     my10: db 00110100b;52
15     my11: db 00111001b;57
16     my12: db 00111110b;62
17     my13: db 01000100b;!!68
18     my14: db 01001001b;73
19     my15: db 01001110b;78
20     my16: db 01010011b;83
21     my17: db 01011000b;88
22     my18: db 01011101b;93
23     my19: db 01100010b;98
24     my20: db 01100111b;103
25     my21: db 01101100b;108
26     my22: db 01110010b;!!114
27     my23: db 01110111b;119
28     my24: db 01111100b;124
29     my25: db 10000001b;129
30     my26: db 10000110b;134
31     my27: db 10001011b;139
32     my28: db 10010000b;144
33     my29: db 10010101b;149
34     my30: db 10011010b;154
35     my31: db 10100000b;!!160
36     my32: db 10100101b;165
37     my33: db 10101010b;170
38     my34: db 10101111b;175
39     my35: db 10110011b;!!179
40
41 ADCreadSeg Segment code
42 RSEG ADCreadSeg;
43 ADCread:
44 mov R3, 0FFh;tot timpul vom crede ca este o eroare initial
45 mov b, P1;punem in B valoarea citita de la P1
46 mov dptr, #100h;
47 loopadc:
48
49 mov a, dpl;
50 cjne a, #36, continuareverif;
51 sjmp stop_nok;
52
53 continuareverif:
54 mov a, #50h;
55 movc a, @a+dptr;
56 inc dptr;
57
58 cjne a, b, loopadc;
59 sjmp stopok;
60
61 stop_nok:
62 mov R3, #0FFh;cod ca nu e ok
63 sjmp stop;
64
65 stopok:
66 mov R3, 00h;cod ca e ok
67
68 mov a, dpl;
69 dec a;
70 mov R6, a;
71 stop:
72 ret;
73 end;
```

NrDisplay.a51

```
1  Name RutinaNrDisplay;
2  Public NrDisplay;
3
4  Cseg at 186h
5      disp0: db 00110000b;48
6      disp1: db 00110001b;49
7      disp2: db 00110010b;50
8      disp3: db 00110011b;51
9      disp4: db 00110100b;52
10     disp5: db 00110101b;53
11     disp6: db 00110110b;54
12     disp7: db 00110111b;55
13     disp8: db 00111000b;56
14     disp9: db 00111001b;57
15
16 NrDisplaySeg Segment code
17 RSEG NrDisplaySeg;
18 Extrn Code (sendInstr);
19 Extrn code (DPTRFilter);
20 Extrn code (Display_up_nr);
21 Extrn code (DPTRFilter_ref);
22 Extrn code (Display_down_nr);
23 NrDisplay:
24 acall DPTRFilter;
25 acall Display_up_nr;
26 acall DPTRFilter_ref;
27 acall Display_down_nr;
28
29 gata_display:
30 ret;
31 end;
```

I consider in R6-the value which was read from the ADC and in R7-the reference temperature value.

DPTRfilter.a51

```
Name RutinaDPTRFilter;
Public DPTRFilter;

DPTRFilterSeg Segment code
RSEG DPTRFilterSeg;

DPTRFilter:
cjne R3,#0FFh, start_filter;
sjmp gata;

start_filter:

    mov a, R6;
    mov dptr, #0h;
    constrloop:

    test0:
    cjne a, #0, test10;
    sjmp H0;

    test10:
    cjne a, #10, test20;
    sjmp H10;

    test20:
    cjne a, #20, test30;
    sjmp H20;

    test30:
    cjne a, #30, FormL;
    sjmp H30;

    FormL:
    dec a;
    inc dpl;
    sjmp constrloop;

    H0:
    mov dph, #0;
    sjmp gata;

    H10:
    mov dph, #1;
    sjmp gata;

    H20:
    mov dph, #2;
    sjmp gata;

    H30:
    mov dph, #3;
    sjmp gata;

    gata:
    ret;
end;
```

Display_up_nr.a51*

```
Name RutinaDisplay_up_nr;
Public Display_up_nr;

instrgrd equ 11011111b;

Display_up_nrSeg Segment code
RSEG Display_up_nrSeg;
Extrn Code (sendInstr);

Display_up_nr:
    clr P3.4;setez RS pe 0
mov P2, #10001100b;setez pozitia corecta
acall sendInstr;
mov a, dph;
mov b, dpl;
setb P3.4;Rs = 1

regular_display:
mov a, dph;
mov b, dpl;

setb P3.4;

mov dptr, #186h;trimitem MSB
movc a, @a+dptr;
mov P2, a;
acall sendInstr;

mov a, b;
movc a, @a+dptr;
mov P2, a;
acall sendInstr;
;punem bulina de grade
mov P2, #instrgrd;
acall sendInstr;

ret;
end;
```

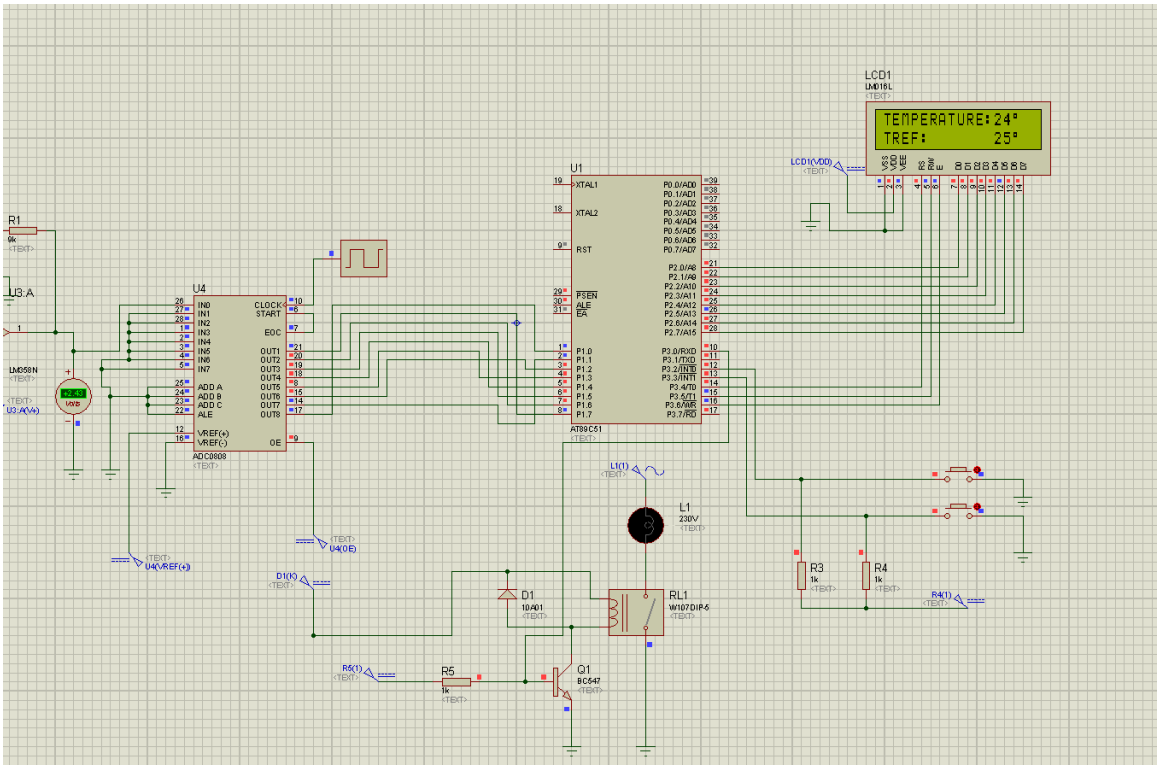
Display_down_nr.a51*

```
Name RutinaDisplay_down_nr;
Public Display_down_nr;

instrgrd equ 11011111b;

Display_down_nrSeg Segment code
RSEG Display_down_nrSeg;

Extrn Code (sendInstr);

Display_down_nr:
clr P3.4;setez RS pe 0
mov P2, #11001100b;setez pozitia corecta
acall sendInstr;
mov a, dph;
mov b, dpl;
setb P3.4;Rs = 1

regular_display:
mov a, dph;
mov b, dpl;

setb P3.4;

mov dptr, #186h;trimitem MSB
movc a, @a+dptr;
mov P2, a;
acall sendInstr;

mov a, b;
movc a, @a+dptr;
mov P2, a;
acall sendInstr;
;punem bulina de grade
mov P2, #instrgrd;
acall sendInstr;
ret;
end;
```

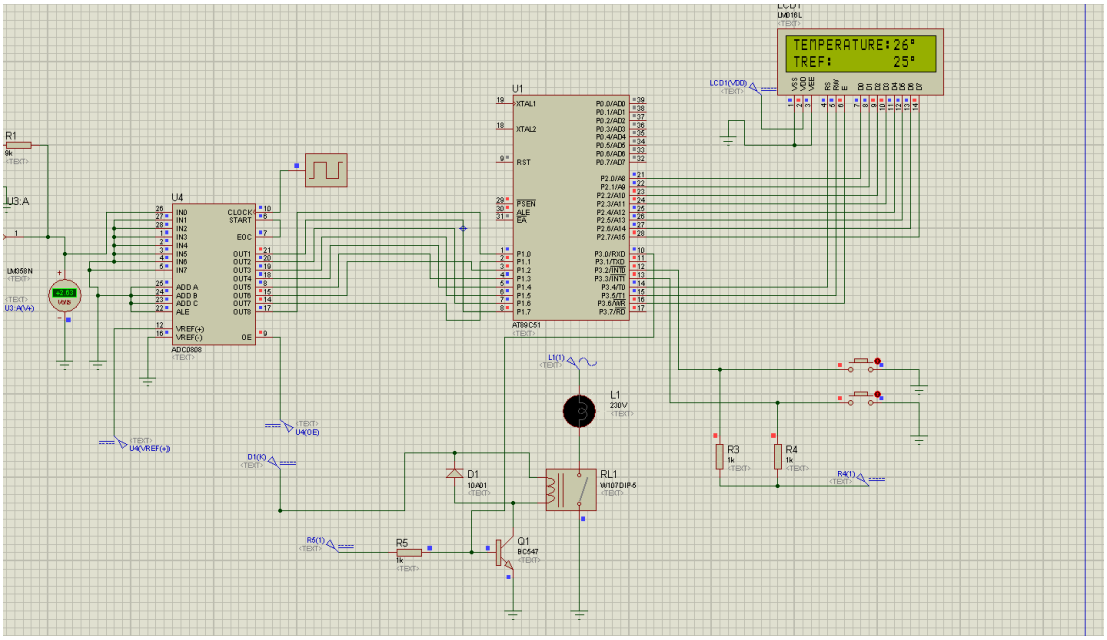
The second DPTR filter is similar to the first one, but it filters the value in R7-Vref

Simulation of the ASM program:

In this case, the reference temperature is higher than the reading from the ADC, therefore P3.0 will be set HIGH and the relay will be turned on.



The ‘-‘ button was pressed, the reference temperature decreased, being equal to the reading temperature, thus switching the relay off.



The program in C

Main.c*

```c
#include<reg51.h>
void delay_ms();
void Init_Lcd();
unsigned int ADC_Read();
void display_number(unsigned int Adc, unsigned int ref);
void Test_relay(unsigned int adc, unsigned int ref);
void ISR_ex0(void);
void ISR_ex1(void);
unsigned int ref = 25;
void main(void)
{
  unsigned int adc;
  IE = 0x85;
  IT0=1;
  IT1=1;
  Init_Lcd();
  while(1)
  {
    delay_ms();
    adc = ADC_Read();
    delay_ms();
    display_number(adc, ref);
    delay_ms();
    Test_relay(adc, ref);
  }
}
void ISR_ex0(void)  interrupt 0
{
  ref++;
}
void ISR_ex1(void)  interrupt 2
{
  ref--;
}
```

Init_Lcd.c

```c
#include<reg51.h>

void delay_ms();
void instr_set();
void temp_string_display();
void tref_string_display();
void send_instr(unsigned int instruct);

sbit RS = P3^4;
sbit RW = P3^5;
sbit EN = P3^6;

void Init_Lcd()
{
  RS = 0;
  RW = 0;
  EN = 0;

  instr_set();
  temp_string_display();
  tref_string_display();
}

void send_instr(unsigned int instruct)
{
  P2 = instruct;
  delay_ms();
  EN = 1;
  delay_ms();
  EN =~ EN;
}

void instr_set()
{
  unsigned int i;
  unsigned char instr[4] = {56, 8, 1, 12};
  for (i = 0; i < 4; i++)
  {
    send_instr(instr[i]);
  }
}

void temp_string_display()
{
  //instructiunile necesare pe a afisa 'TEMPERATURE:'
  unsigned char instr[12] = {84, 69, 77, 80, 69, 82, 65, 84, 85, 82, 69, 58};
  unsigned int i;
  RS = 1;
  for (i = 0; i < 12; i++)
  {
    send_instr(instr[i]);
  }
}

void tref_string_display()
{
  unsigned char instr[5] = {84, 82, 69, 70, 58};
  unsigned int i;
  //setez a doua linie a LCD-ului
  RS = 0;
  send_instr(192);

  RS = 1;
  //instructiunile necesare pe a afisa 'TREF:'
  for (i = 0; i < 5; i++)
  {
    send_instr(instr[i]);
  }
}
```

ADC_Read.c

```c
#include<reg51.h>

unsigned int ADC_Read()
{
  unsigned int x = P1;
  unsigned int i;
  unsigned int value = 1;
  if (x == 179)
    return (35);
  for (i = 0; i < 36; i++)
  {
    if (value == x)
    {
      return (i);
    }
    if (i == 3 || i == 12 || i == 21 || i == 30 || i == 34)
      value = value + 6;
    else
      value = value + 5;
  }
  return (0);
}
```

Display_number.c

```c
#include<reg51.h>

sbit RS = P3^4;
sbit RW = P3^5;
sbit EN = P3^6;

void display_digit(unsigned int dig);
void send_instr(unsigned int instruct);
void delay_ms();
void display_upper(unsigned int x);
void display_lower(unsigned int x);

void display_number(unsigned int Adc, unsigned int ref)
{
  display_upper(Adc);
  display_lower(ref);
}

void display_upper(unsigned int x)
{
  RS = 0;
  //setez pozitia dorita
  send_instr(140);
  RS = 1;
  delay_ms();
  //MSB
  display_digit(x/10);
  delay_ms();
  //LSB
  display_digit(x%10);
  send_instr(223);
```

```
  send_instr(223);
}

void display_lower(unsigned int x)
{
  RS = 0;
  //setez pozitia dorita
  send_instr(204);
  RS = 1;
  delay_ms();
  //MSB
  display_digit(x/10);
  delay_ms();
  //LSB
  display_digit(x%10);
  send_instr(223);
}

void display_digit(unsigned int dig)
{
  RS = 1;
  send_instr(dig + 48);
}
```

Delay_ms.c

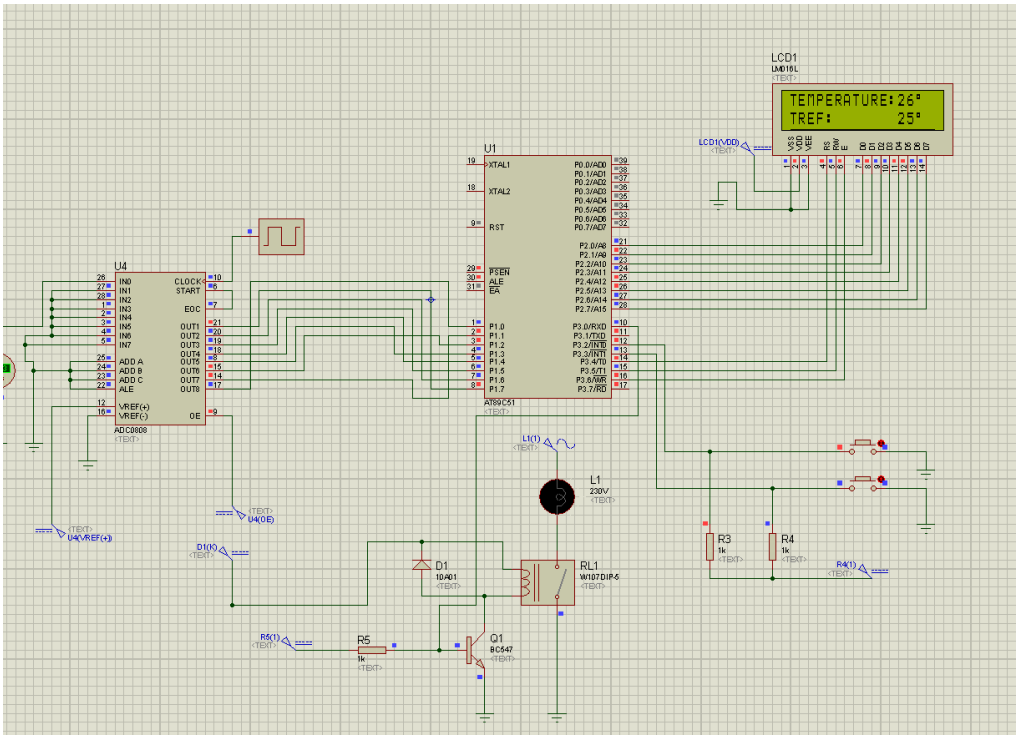
```
#include <reg51.h>
//functie care creeaza un delay de n ms
void delay_ms()
{
  TMOD = 0x01;
  TH0 = 0xFC;
  TL0 = 0x66;
  TR0 = 1;
  while(TF0 == 0)
    ;
  TF0 = 0;
  TR0 = 0;
}
```

Test_Relay.c

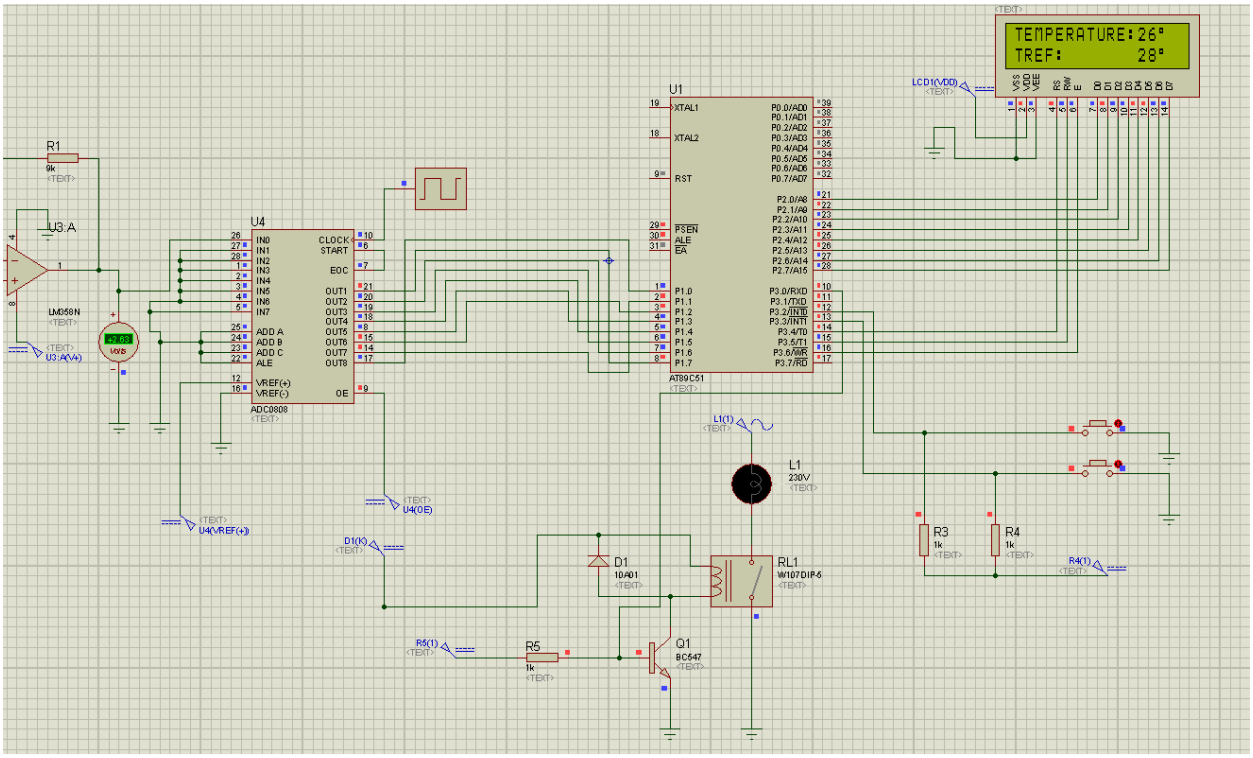
```
#include<reg51.h>

sbit Rellay = P3^0;

void Test_relay(unsigned int adc, unsigned int ref)
{
  if (adc >= ref)
    Rellay = 0;
  else
    Rellay = 1;
}
```

Simulation of the C program:

The read temp is higher than the reference=>relay off:



The relay turns on in this case:



References:

http://www.ti.com/lit/ds/symlink/lm35.pdf

http://www.ti.com/lit/ds/symlink/adc0804-n.pdf

http://www.circuitstoday.com/op-amp-circuits-in-proteus

http://www.circuitstoday.com/proteus-tutorial-switches-and-relays

http://www.ti.com/lit/ds/symlink/adc0808-n.pdf

https://www.youtube.com/watch?v=iNvtgEB7q6U

https://www.youtube.com/watch?v=TZuXfapExic&fbclid=IwAR0bdc5HNcUX-dmrg3ojPR8QmZzLVZXNkvM8RzKCjaBRwToiRhBWWAiBBfI

http://www.picaxe.com/docs/led008.pdf

https://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-3.3v.pdf

https://www.techshopbd.com/uploads/product_document/lm016l%20lcd.pdf

http://www.keil.com/dd/docs/datashts/philips/8xc752_ds.pdf

https://circuitglobe.com/relay.html

https://www.electroschematics.com/8648/run-stop-relay-circuit/

https://www.pc-control.co.uk/relays.htm

http://www.circuitstoday.com/working-of-relays

https://www.explainthatstuff.com/howrelayswork.html

https://microcontrollerslab.com/interrupts-8051-microcontroller/

http://www.keil.com/support/man/docs/armcc/armcc_chr1359124246903.htm