

MINISTERUL EDUCAȚIEI NAȚIONALE



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Room Thermostat

Microcontrollers project

Student: Petraru Denisa-Ionela

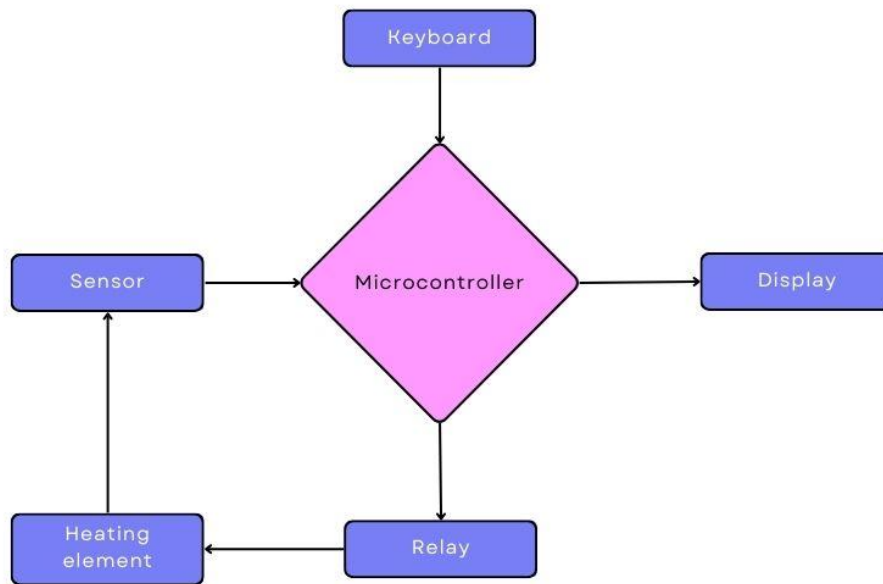
Group: 2031

Coordinator: Olteanu Mirela

Contents

Block scheme.....	3
Sensor classes.....	4
1. Resistance Temperature Detectors (RTDs).....	4
2. Negative Temperature Coefficient (NTC) Thermistors	4
3. Thermocouples.....	5
4. Semiconductor Sensors	6
Choosing the sensor	6
Temperature Sensor: LM35	7
Humidity Sensor: TH02	8
Connecting LM35 to the Microcontroller	9
Connecting the sensor	9
Connecting the amplifier.....	10
Choosing the operational amplifier.....	12
Connecting the ADC.....	13
Connecting the Microcontroller	15
8051 architecture.....	15
80C51 microcontroller	17
Connecting the Relay	23
Final schematic	24
Final code.....	25
Bibliography	33

Block scheme

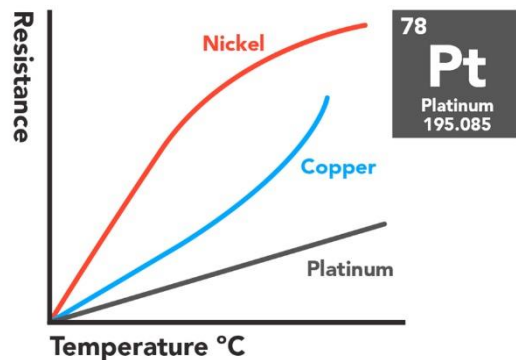


Sensor classes

1. Resistance Temperature Detectors (RTDs)

As temperature changes, the resistance of any metal changes as well. This difference in resistance is what RTD temperature sensors are based on. The RTD's resistance increases linearly when the temperature increases. An RTD is a resistor with well-defined resistance vs. temperature characteristics.

RTDs are made from a film and a glass or ceramic core with wire wrapped around it for greater accuracy. Platinum makes up the most accurate RTDs.

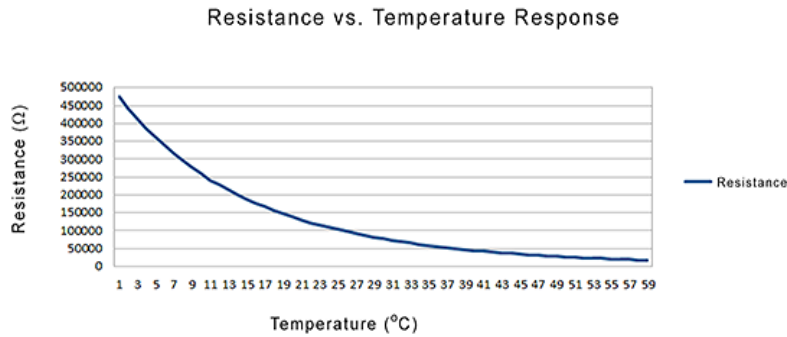


In terms of how it works, the RTD follows a basic principle. When the temperature of a metal increases, the resistance to the flow of electricity increases as well. An electrical current is passed through the sensor, the resistance element is used to measure the resistance of the current being passed through it. As the temperature of the resistance element increases the electrical resistance also increases.

2. Negative Temperature Coefficient (NTC) Thermistors

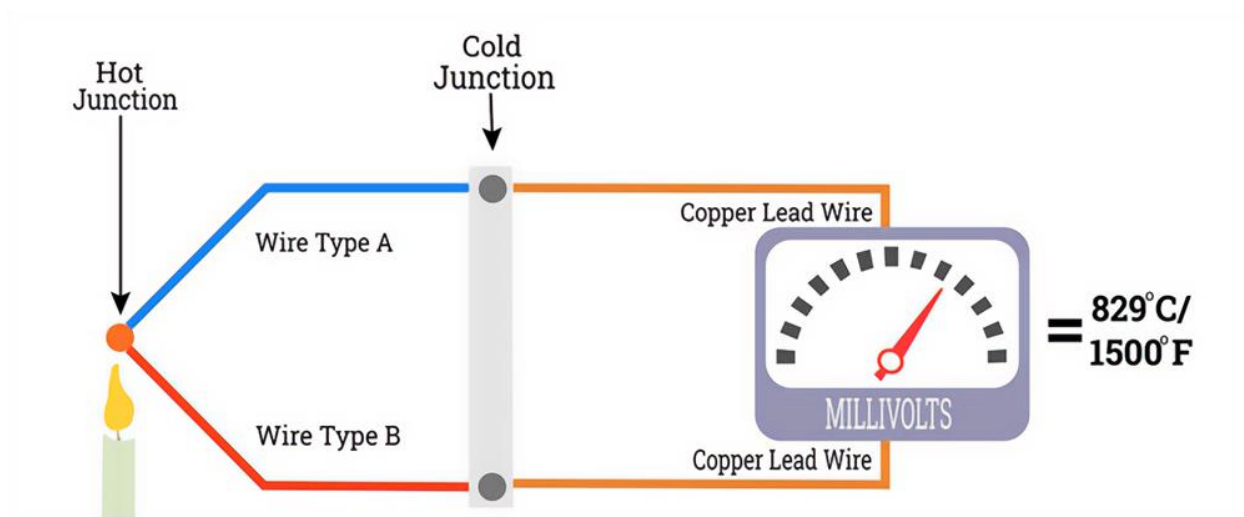
Thermistors are similar to RTDs as temperature changes cause measurable resistance changes. Thermistors, however, are made from semiconductor materials. An NTC thermistor provides higher resistance at low temperatures.

The output of an NTC thermistor is non-linear due to its exponential nature; however, it can be linearized based on its application. Thus, in the thermistors operating range, we can see a large resistance change for a very small temperature change. This makes for a highly sensitive device.



3. Thermocouples

Thermocouples are made by joining two dissimilar metal wires, electrically bonded at two points. The connecting end is called the “hot junction”, and the other end is known as the “cold junction”. The varying voltage between the two metals mirrors proportional changes in temperature.



The working principle is very simple. When the two dissimilar metal wires are fused, they create a thermoelectric result, which provides a small voltage. This causes a Seebeck Effect.

The Seebeck Effect is a phenomenon in which a temperature difference of two dissimilar conductors produces a voltage difference between the two substances.

The biggest disadvantage to using a thermocouple sensor is its small output voltage, making it challenging to measure the temperature. This results in a low accuracy. Thermocouples operate across the widest temperature range.

4. Semiconductor Sensors

Semiconductor-based temperature sensors (also known as IC sensors) have a dual integrated circuit (IC) containing two similar diodes. The diodes have temperature-sensitive voltage vs current characteristics that are used to monitor changes in temperature.

Semiconductor based temperature sensor ICs come in two different types: local temperature sensor and remote digital temperature sensor. Local temperature sensors are ICs that measure their own die temperature by using the physical properties of a transistor. Remote digital temperature sensors measure the temperature of an external transistor.

Choosing the sensor

	Temperature range	Price	Accuracy	Resolution
LM35	-55°C - 150°C	3.5\$	$\pm 1^{\circ}\text{C}$	10mV/ $^{\circ}\text{C}$
AD22100	-50°C - 150°C	7.5\$	$\pm 2^{\circ}\text{C}$	22.5mV/ $^{\circ}\text{C}$
AD592	-25°C - 105°C	12\$	$\pm 0.5^{\circ}\text{C}$	-
TC1047	-40°C - 125°C	0.7\$	$\pm 3^{\circ}\text{C}$	10mV/ $^{\circ}\text{C}$

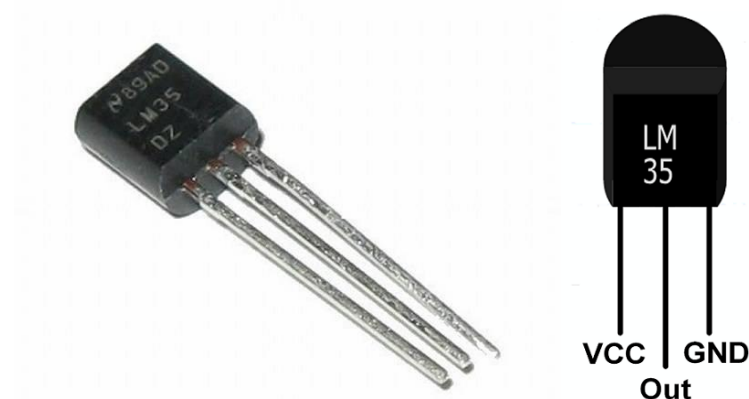
Temperature sensor accuracy is critical to ensure that temperature sensors give effective measurements. Accuracy of temperature sensors is how close the sensor output is to the physical temperature being measured.

Resolution refers to the smallest detectable increment of measurement on an instrument. A thermometer that displays temperature readings to the hundredth of a degree (e.g. 100.26°) has a greater resolution than one that only shows the tenths of a degree (e.g. 100.3°) or whole degrees (100°).

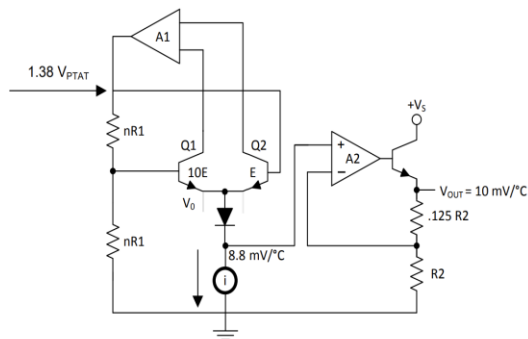
Based on the criteria from the table above, I will choose for my project the LM35 sensor, having the best resolution, the second best accuracy and the second best price.

Temperature Sensor: LM35

LM35 is a temperature measuring device having an analog output voltage proportional to the temperature. It provides output voltage in Centigrade (Celsius). It does not require any external calibration circuitry. The sensitivity of LM35 is 10 mV/degree Celsius. As temperature increases, output voltage also increases.



Functional Block Diagram



VCC: Supply Voltage (4V – 30V)

Out: It gives analog output voltage which is proportional to the temperature (in degree Celsius).

GND: Ground

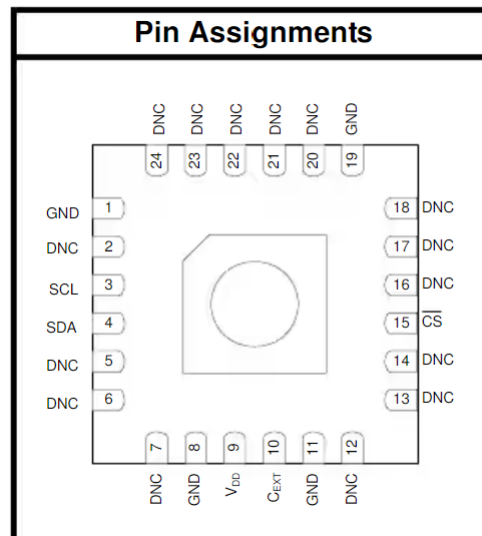
Specifications:

- Operating voltage: 4V to 30V
- Sensitivity: 10mV/°C
- Linearity Error: $\pm 1^\circ\text{C}$
- Operating Temperature: -55°C to $+150^\circ\text{C}$
- Typical Power Consumption: 60 μA

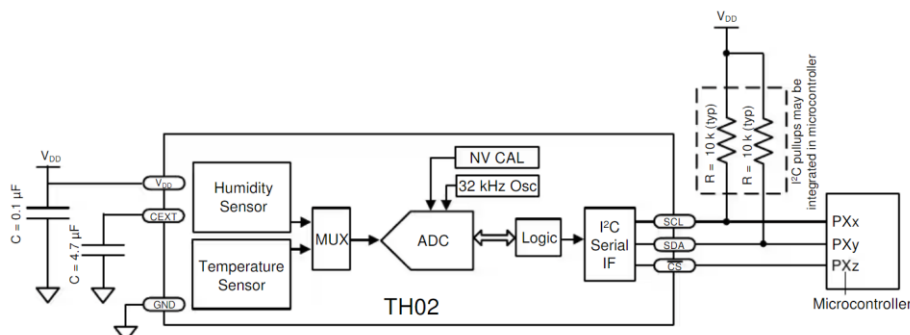
- Output Type: Analog
- Accuracy: $\pm 1^{\circ}\text{C}$

Humidity Sensor: TH02

The TH02 is a digital relative humidity and temperature sensor. This monolithic CMOS IC integrates temperature and humidity sensor elements, an analog-to-digital converter, signal processing, calibration data, and an I2C host interface.



Functional Block Diagram



Specifications:

- Operating Voltage: 2.1V to 3.6V
- Power Consumption: 240 μ A during RH conversion
- Relative Humidity Sensor: $\pm 4.5\%$ RH (maximum @0-80% RH)

Connecting LM35 to the Microcontroller

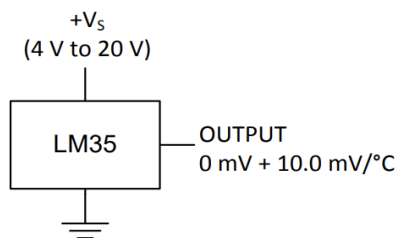
Since the temperature sensor LM35 has an analog output and the microcontroller 8051 has a digital input, we need an analog to digital converter to connect them.

The output voltage of the sensor increases with $10\text{mV}/^\circ\text{C}$ so for our temperature range ($0^\circ\text{C} - 40^\circ\text{C}$), the maximum output voltage of the sensor should be 400mV . We need to reach the full scale voltage of the ADC through a differential amplifier.

These are the elements needed further, all of them being supplied by the same source:

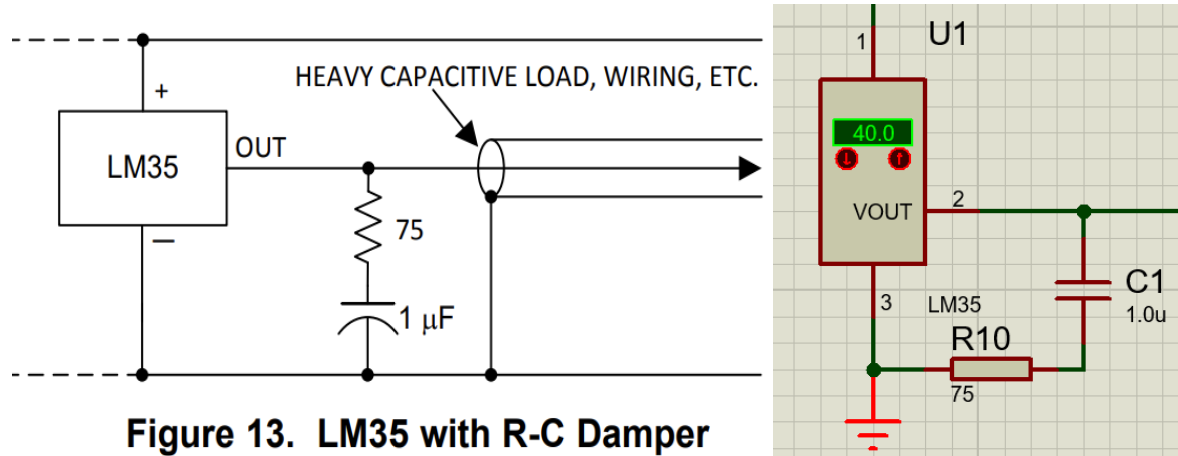


Connecting the sensor



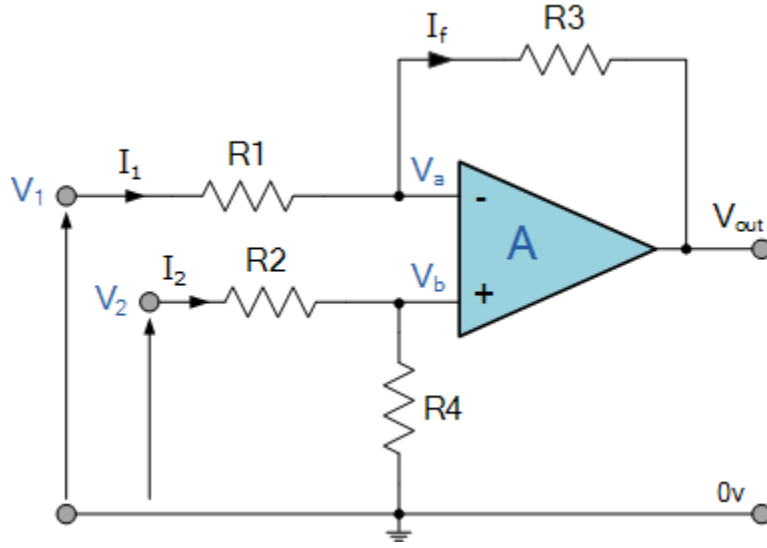
The LM35 temperature sensor has 3 pins: +Vs, Vout and GND. The supply voltage, Vs, has the absolute maximum ratings in the datasheet of -0.2V (MIN) and 35V (MAX), but in practice it needs to be comprised between 4V and 20V. The chosen supply for this project is 5V. The output of the sensor will be connected to the non-inverting input of the amplifier.

The device has a limited ability to drive heavy capacitive loads. The tolerance of capacitance can be improved with a series R-C damper from output to ground.



Connecting the amplifier

For this project, a differential amplifier is needed, because of the low output voltage of the temperature sensor. Differential amplifiers are useful in electrically noisy environments where a low amplitude electrical signal can be easily corrupted by the effect of unwanted external noise. A single-ended amplifier would be unsuitable since it would also amplify the unwanted noise signal as well as the desired input signal. The output of the sensor will be connected to the non-inverting input of the amplifier (V2), and the inverting input (V1) will be connected to ground.



The gain of the amplifier is calculated such that the maximum output of the sensor (400 mV) reaches the maximum voltage of the ADC. The maximum voltage of the ADC is found by using the formula:

$$V_{ADCmax} = V_{FS} - V_{LSB} = V_{FS} - \frac{V_{FS}}{2^n}$$

Where V_{FS} is the full-scale voltage of the ADC and n is the number of bits. In this case an 8-bit ADC will be used, with the reference voltage of 5V. The formula becomes:

$$V_{ADCmax} = 5V - \frac{5V}{2^8} = 5V - \frac{5V}{256} \cong 5V - 0.02V \cong 4.98V$$

We will approximate the result to 5V. The gain will be the following:

$$A = \frac{V_{ADCmax}}{V_{LM35max}} = \frac{5V}{400mV} \cong 12.5$$

After denoting V_2 with V_{in} and connecting V_1 to GND the following equations describe the circuit:

$$V^+ = \frac{R_4}{R_4 + R_2} \times V_{in}$$

$$V^- = \frac{R_1}{R_1 + R_3} \times V_{out}$$

Because of the negative feedback, the two input voltages will be equal.

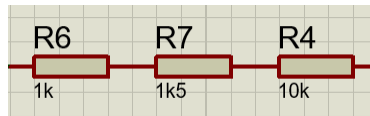
$$V^+ = V^- \rightarrow \frac{R_4}{R_4 + R_2} \times V_{in} = \frac{R_1}{R_1 + R_3} \times V_{out} \rightarrow \frac{V_{in}}{V_{out}} = \frac{R_4}{R_4 + R_2} \times \frac{R_1 + R_3}{R_1}$$

If the two sums of resistances are equal, $R_1 + R_3 = R_2 + R_4$, the gain will be: $A = \frac{R_4}{R_1}$.

The values of the resistances are chosen as follows:

$$R_1 = R_2 = 1k\Omega \quad R_3 = R_4 = 12.5k\Omega$$

In Proteus, the $12.5k\Omega$ value will be obtained by placing three resistors in series as follows:



Choosing the operational amplifier

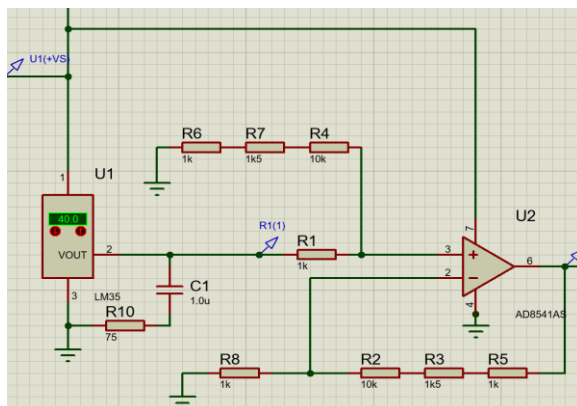
Rail-to-rail operational amplifiers

Op-amps having a common-mode input voltage range that almost covers the $GND-V_{CC}$ or V_{EE} -to- V_{CC} range are called rail-to-rail input op-amps (or full-swing op-amps).

Since the final product is wanted to be implemented in real life, a rail-to-rail operational amplifier is needed. Because we need the minimum voltage at the output to be as close as possible to 0, with a normal operational amplifier this could have been achieved only with a negative supply, which cannot be implemented with a normal battery.

The chosen amplifier is AD8541, based on the following criteria:

- Rail-to-rail input and output
- Single-supply operation: 2.7V to 5.5V
- Low supply current: $45\mu A$ /amplifier

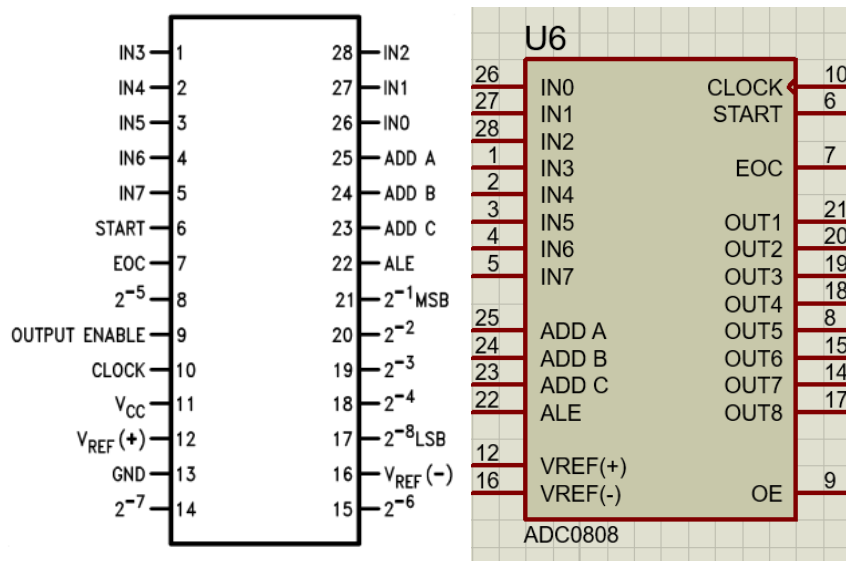


Connecting the ADC

In electronics, an analog-to-digital converter is a system that converts an analog signal, such as a sound picked up by a microphone or light entering a digital camera, into a digital signal.

The chosen ADC is ADC0808 based on the following features:

- Easy Interface to All Microprocessors
- Operates with 5 VDC
- 8-Channel Multiplexer with Address Logic
- 0V to VCC Input Range

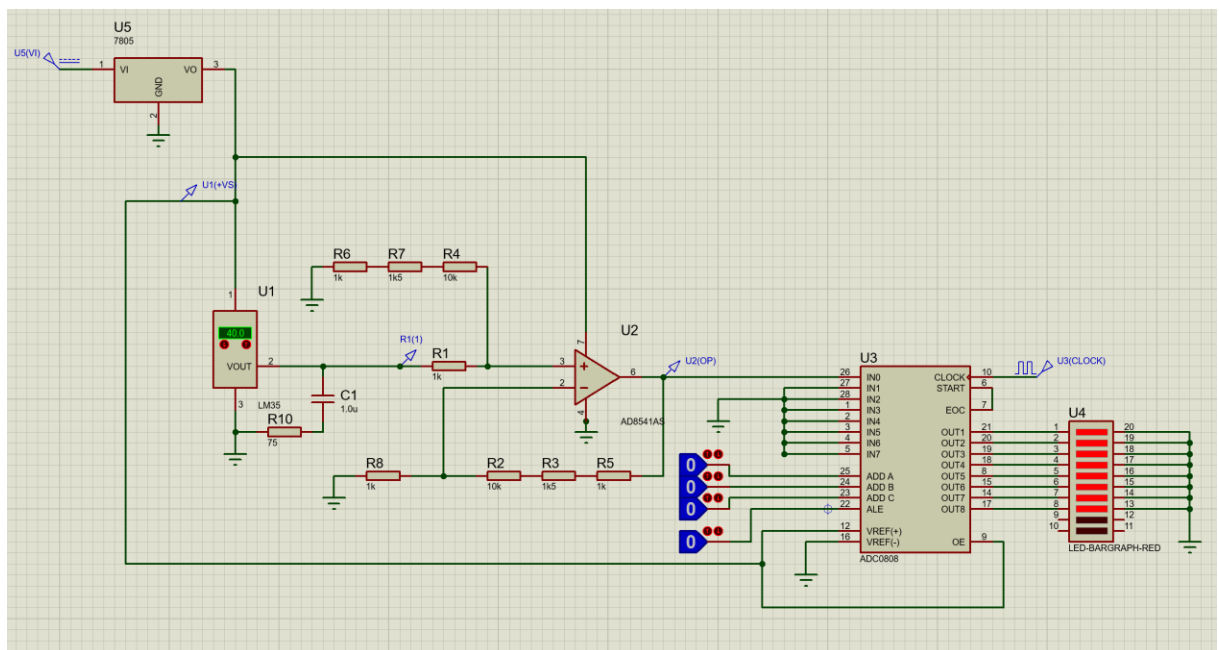


The ADC0808 data acquisition component is a monolithic CMOS device with an 8-bit analog-to-digital, 8-channel multiplexer and microprocessor compatible control logic. The 8-bit A/D converter uses successive approximation as the conversion technique. The converter features a high impedance chopper stabilized comparator, a 256R voltage divider with analog switch tree and a successive approximation register.

The heart of this single chip data acquisition system is its 8-bit analog-to-digital converter. The converter is designed to give fast, accurate, and repeatable conversions over a wide range of temperatures. The converter is partitioned into 3 major sections: the 256R ladder network, the successive approximation register, and the comparator.

The successive approximation register (SAR) performs 8 iterations to approximate the input voltage. For any SAR type converter, n-iterations are required for an n-bit converter.

The output of the amplifier could be connected to any of the 8 inputs, and then the addresses A, B and C modified accordingly. The Address Latch Enable is used to set the addresses. Start must be connected to End Of Conversion so that after finishing a conversion the circuit automatically starts a new one. The Output Enable signal causes the ADC to actually output the digital values on the output lines.



Connecting the Microcontroller

Next up, the “brain” of the project needs to be connected. The whole circuit will work controlled by the microcontroller.

A microcontroller is a small computer on a single integrated chip. It contains one or more central processing units, along with memory and programmable input/output peripherals.

Microcontrollers are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications consisting of various discrete chips.

8051 architecture

The 8051 architecture provides many functions (central processing unit (CPU), random-access memory (RAM), read-only memory (ROM), input/output (I/O) ports, serial port, interrupt control, timers) in one package.

The 8051 family became widely popular after Intel allowed other manufacturers to make and market any flavors of the 8051 they please with the condition that they remain code-compatible with the 8051. This has led to many, versions of the 8051 with different speeds and amounts of on-chip ROM marketed by more than half a dozen manufacturers. If you write your program for one, it will run on any of them regardless of the manufacturer.

8051 microcontroller

In 1981, Intel Corporation introduced an 8-bit microcontroller called the 8051. This microcontroller had 128 bytes of RAM, 4K bytes of on-chip ROM, two timers, one serial port, and four ports (each 8-bits wide) all on a single chip. At the time it was also referred to as a “system on a chip.” The 8051 is an 8-bit processor, meaning that the CPU can work on only 8 bits of data at a time. Data larger than 8 bits has to be broken into 8-bit pieces to be processed by the CPU. The 8051 has a total of four I/O ports, each 8 bits wide.

8052 microcontroller

The 8052 is another member of the 8051 family. The 8052 has all the standard features of the 8051 as well as an extra 128 bytes of RAM and an extra timer. In other words, the 8052 has 256 bytes of RAM and 3 timers. It also has 8K bytes of on-chip program ROM instead of 4K bytes.

8031 microcontroller

This chip is often referred to as a ROM-less 8051 since it has 0K bytes of on-chip ROM. To use this chip you must add external ROM to it. This external ROM must contain the program that the 8031 will fetch and execute.

In the process of adding external ROM to the 8031, you lose two ports. That leaves only 2 ports (of the 4 ports) for I/O operations. To solve this problem, you can add external I/O to the 8031.

8032 microcontroller

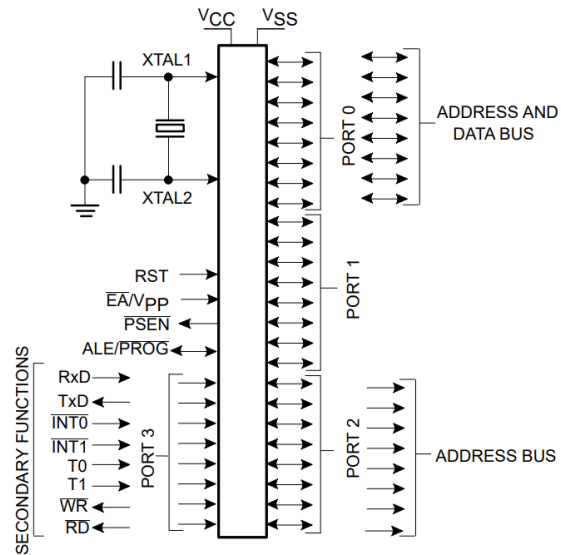
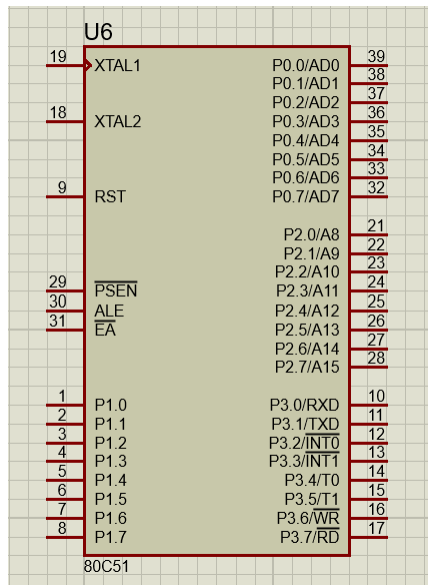
The Intel 8032 is a MCS-51 NMOS single-chip 8-bit microcontroller with 32 I/O lines, 3 Timers/Counters, 6 Interrupts/4 priority levels ROM-less, 256 Bytes on-chip RAM.

8751 microcontroller

This 8751 chip has only 4K bytes of on-chip UV-EPROM. Using this chip for development requires access to a PROM burner, as well as a UV-EPROM eraser to erase the contents of UV-EPROM inside the 8751 chip before you can program it again.

Feature	8051	8052	8031	8032	8751
ROM (program space in bytes)	4K	8K	0K	0K	4K
RAM (bytes)	128	256	128	256	128
Timers	2	3	2	3	2
I/O pins	32	32	32	32	32
Serial port	1	1	1	1	1

80C51 microcontroller



Pins:

- **PSEN** – is used for enabling the external program memory, connected to the external ROM memory chip
- **EA** – forces the controller to use the external program memory when connected to GND
- **XTAL1, XTAL2** – used for connecting the crystal to the microcontroller, which gives the clock pulses
- **RST** – used for restarting the microcontroller
- **ALE** – Address Latch Enable, positive going pulse generated when a new operation is started by the microprocessor

Cod C:

```
#include <reg51.h>
```

```
sbit rs = P3^0;
```

```
sbit en = P3^2;
```

```
unsigned int aux=0, tab[2];
```

```

float t=0;

void LCD_cmd(unsigned char);
void LCD_data(unsigned char);
void message();
void delay();
void port_init();
void temp_display();

void main(){
    port_init();
    message();

    temp_display();

}

void LCD_cmd(unsigned char x){
    P2 = x;
    rs = 0;
    en = 1;
    delay();
    en = 0;
}

void LCD_data(unsigned char x){
    P2 = x;
    rs = 1;
    en = 1;
    delay();
    en = 0;
}

void delay(){

```

```

        unsigned int i;
        for(i=0; i<1200; i++);
    }

    void message(){
        LCD_cmd(0x38); //5X7 MATRIX CRYSTAL
        LCD_cmd(0x0E); // display on, cursion on
        LCD_cmd(0x0c);
        LCD_cmd(0x80); //cursor at line 1, position 0
        LCD_data('T');
        LCD_data('e');
        LCD_data('m');
        LCD_data('p');
        LCD_data('e');
        LCD_data('r');
        LCD_data('a');
        LCD_data('t');
        LCD_data('u');
        LCD_data('r');
        LCD_data('e');
        LCD_data(':');
    }

    void port_init(){
        P1 = 0xFF; //input
        P2 = 0;    //output
        P3 = 0;    //output
    }

    void temp_display(){
        t = (P1*0.019 + 0.019)/11;
        aux = t * 10000;
        tab[1] = aux/100%10;
        tab[0] = aux/1000%10;
    }

```

```

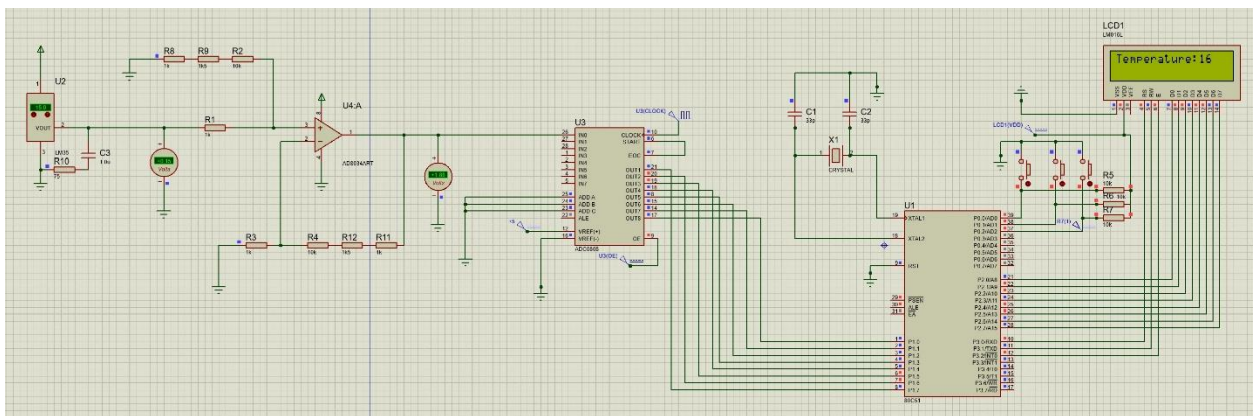
if(tab[0] == 0)
    LCD_data(0x20);
else
    LCD_data(tab[0]+'0');

LCD_data(tab[1]+'0');

}

```

Functionare C:



Cod assembly:

```

ORG 0H

MOV A, #38H ; initialiae LCD 2 LINES, 5X7 MATRIX
ACALL COMNWRT ; call command subroutine
ACALL DELAY ; give LCD some time
MOV A, #0EH ; display on, cursor on
ACALL COMNWRT ;call command subroutine
ACALL DELAY ;give LCD some time
MOV A, #01; clear LCD
ACALL COMNWRT
ACALL DELAY

MOV A, #80H ;cursor at line 1, position 0
ACALL COMNWRT

```

ACALL DELAY

MOV A, #'T' ;display letter T

ACALL DATAWRT

ACALL DELAY

MOV A, #'e'

ACALL DATAWRT

ACALL DELAY

MOV A, #'m'

ACALL DATAWRT

ACALL DELAY

MOV A, #'p'

ACALL DATAWRT

ACALL DELAY

MOV A, #'e'

ACALL DATAWRT

ACALL DELAY

MOV A, #'r'

ACALL DATAWRT

ACALL DELAY

MOV A, #'a'

ACALL DATAWRT

ACALL DELAY

MOV A, #'t'

ACALL DATAWRT

ACALL DELAY

```
MOV A, #'u'
ACALL DATAWRT
ACALL DELAY
```

```
MOV A, #'r'
ACALL DATAWRT
ACALL DELAY
```

```
MOV A, #'e'
ACALL DATAWRT
ACALL DELAY
MOV A, #' '
ACALL DATAWRT
```

```
MOV A, #':'
ACALL DATAWRT
ACALL DELAY
```

```
;ACALL DATAWRT
```

```
AGAIN: SJMP AGAIN ;stay here
```

```
COMNWRT:
```

```
    MOV P2,A ;copy reg A to port 2
        CLR P3.0 ; RS=0 for command
        CLR P3.1 ;R/W for write
        SETB P3.2 ;E=1 FOR HIGH PULSE
        ACALL DELAY
        CLR P3.2 ;E=0 for H-to-L pulse
        RET
```

```
DATAWRT: ;write data to the LCD
```

```
    MOV P2, A; copy reg A to port 2
        SETB P3.0 ;RS=1 for data
```

CLR P3.1 ;R/W for write

SETB P3.2 ;E=1 for high pulse

ACALL DELAY

CLR P3.2 ;E=0 for H-to-L pulse

RET

DELAY: MOV R3, #50 ;50 or higher for fast CPUs

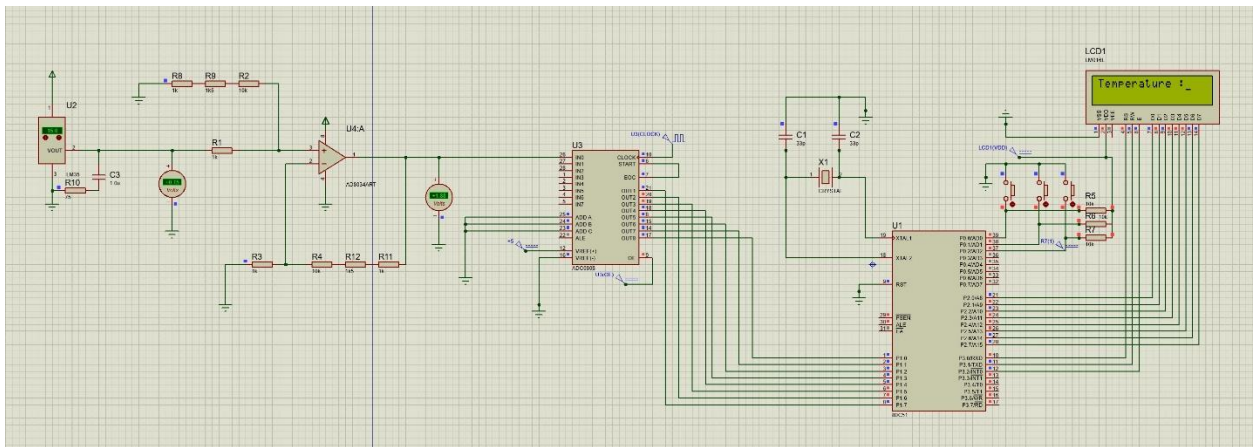
HERE2: MOV R4, #255; R4=255

HERE: DJNZ R4, HERE ;stay untill R4 becomes 0

DJNZ R3, HERE2

RET

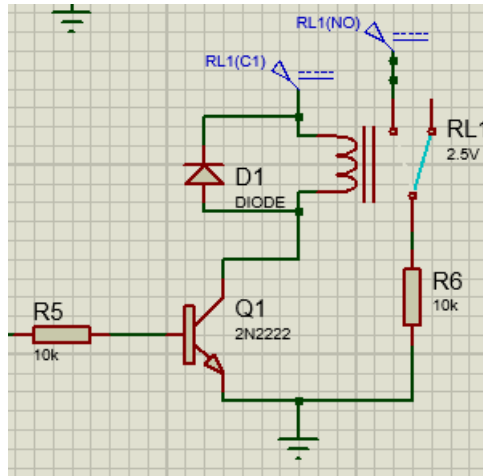
END



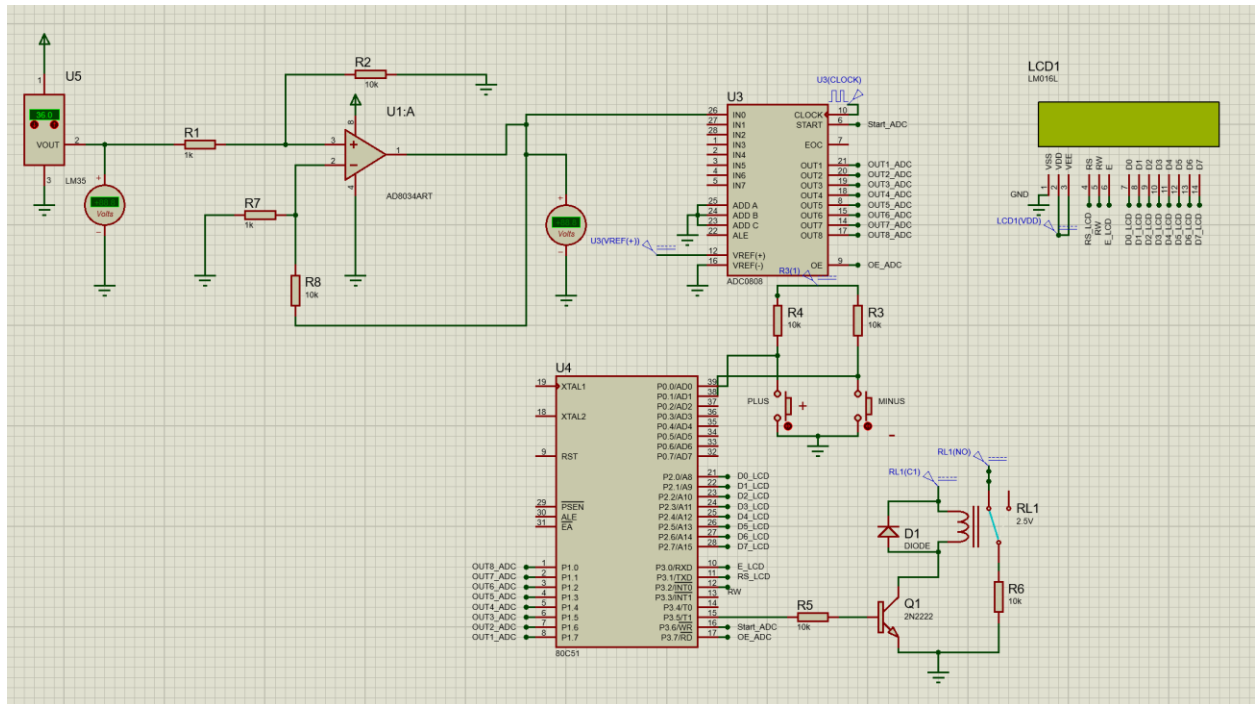
Connecting the Relay

The last, but not the least electronic component added to the schematic is the relay. A relay is an electrical device that is used to control the flow of electric current in a circuit. It is an electromagnetic switch that uses a small current to control a larger current or voltage. Relays are commonly used in various applications to provide electrical isolation, amplify signals, and control high-power devices.

The basic operation of a relay involves an electromagnet and a set of contacts. When a small current is applied to the electromagnet, it creates a magnetic field that attracts or releases a set of contacts, thereby opening or closing a circuit. This allows the relay to control the flow of electricity to another device or circuit.



Final schematic



Final code

Assembly

```
1 RS equ P3.1
2 E equ P3.0
3 functionset equ 38h ; Instruction to set LCD to display
4 ;characters on a 5x7 matrix
5 displayONoff equ 0Eh ; Instruction to turn on the LCD
6 cleardisplay equ 01h ; Instruction to clear the LCD
7 firstLineCursor equ 80h ; Positioning the cursor on the first line
8 secondLineCursor equ 0C0h ; Positioning the cursor on the second line
9
0 org 0000h
1 ; Initialize variables and set buttons as not pressed
2 mov R0,#20 ; Reference voltage in the room
3 setb P0.0 ; Set button 1 as not pressed
4 setb P0.1 ; Set button 2 as not pressed
5
6 ; Call functions to initialize and display on the LCD
7 acall LCD ; Initialize LCD
8 acall first_row_text ; Display message on the first line
9 acall second_row_text ; Display message on the second line
0 acall ADC ; Read temperature from ADC
1
2 LCD:
3 //Initialize LCD
4     mov a,#functionset ; Set LCD to display characters on a 5x7 matrix
5     acall instructiuni
6     acall delay
7
8     mov a,#displayONoff ; Turn on the LCD
9     acall instructiuni
0     acall delay
1
2     mov a,#cleardisplay ; Clear the LCD
3     acall instructiuni
4     acall delay
5
6     mov a,#firstLineCursor; Position the cursor on the first line at position 0
7     acall instructiuni
8     acall delay
9     ; Display a message on the first line
0 first_row_text:
1     mov dptr,#0900h; Point to the memory location containing the message
2     mov Rl,#0h ; Initialize loop counter
3 iar2:
4     mov a,#0h ; Clear accumulator A
5     movc a, @a+dptr ; Read character from memory
6     inc dptr ; Increment memory pointer
7     inc rl ; Increment loop counter
```

```

43 iar2:
44     mov a,#0h ; Clear accumulator A
45     movc a,@a+dptr ; Read character from memory
46     inc dptr ; Increment memory pointer
47     inc r1 ; Increment loop counter
48     acall date ; Send character to LCD
49     acall delay ; Delay for stability
50     cjne r1,#0Eh,iar2 ; Continue loop until all characters
51     ret
52
53 ; Display the message "Temp.setata" on the second line
54 second_row_text:
55     mov a,#secondLineCursor ; Position the cursor on the second line at position 0 ;Pozitionare cursor pe linia a doua la pozitia 0
56     acall instructiuni
57     acall delay
58     mov dptr,#0500h ; Point to the memory location containing the message
59     mov r1,#0h ; Initialize loop counter
60 iar3:
61     mov a,#0h ; Clear accumulator A
62     movc a,@a+dptr ; Read character from memory
63     inc dptr ; Increment memory pointer
64     inc r1 ; Increment loop counter
65     acall date ; Send character to LCD
66     acall delay ; Delay for stability
67     cjne r1,#0Eh,iar3
68     acall show_set_temp ; Display the set temperature
69     ret
70 ; Send instruction to LCD data pins
71 instructiuni:
72     mov P2,a
73     acall enable_instructiuni
74     ret
75 ; Enable instructions on the LCD
76 enable_instructiuni:
77     clr RS ;RS=0 for instructions
78     setb E ; Enable E pin ;E=1 se activeaza pinul E
79     acall delay ; Delay for stability
80     clr E ; Disable E pin ;E=0 se dezactiveaza pinul E
81     ret
82
83
84 date: ;Scrie date pe LCD
85     mov P2,a
86     acall enable_date
87     ret
88
89 enable_date:
90     setb RS ;RS=1 pentru date
91     setb E ;Rs=1 se activeaza pinul E
92     acall delay
93     clr E ;E=0 se dezactiveaza pinul E
94     ret
95
96
97 ADC:
98
99     acall readtemp
100    acall show_current_temp
101
102    acall temp_control
103    acall relay
104    sjmp adc
105 readtemp:
106    setb P3.6; we give a command to ADC in order to 'start' a conversion
107    nop
108    nop ; wait some time
109    nop
110
111    clr P3.6; we stop the 'start' conversion
112
113    waitEOC_low:
114    jb P3.2, waitEOC_low; we wait for 'EOC' pin to be LOW
115    waitEOC_high:
116    jnb P3.2, waitEOC_high ; we wait for 'EOC' pin to be HIGH again
117
118    setb P3.7 ;setting P3.7 we enable the 'OE' of the ADC
119    nop
120    nop
121    nop
122
123    mov A, P1; store in A what we have on P1
124    clr P3.7; disable 'OE' of the ADC
125    mov B, #5 ; because 1 Celsius Degree represents 5 LSB on ADC,
126    div AB ; we want to divide to those 5 LSB in order to find the temperature
127    mov r6, a ; salvam in r6 temperatura citita
128    ret

```

```

130 //Afisare temperatura setata
131 show_current_temp:
132     mov a,#8Eh ;Pozitionare cursor pe prima linie pozitia 14
133     acall instructiuni
134     acall delay
135     mov a,r6; punem in acc temperatura citita
136     mov B,#10
137     div AB ; divide at 10 in order to get the tens from A value. Ex: 45/10 = 4
138     add A, #30h ; add 30h to get the ascii value of the tens remaining in A
139     mov P2, A; mov in P2, A in order to transmit the character
140     acall date
141     acall delay
142     mov A,B ; the remainder of the division will be saved in B so we will move in A the remainder
143     add A, #30h ; get ascii character
144     mov P2, A; transmit the character to LCD
145     acall date
146     acall delay
147     ret
148
149
150 // Afisare temperatura actuala
151 show_set_temp:
152     mov a,#0CEh ; Pozitionare cursor pe a doua linie pozitia 14
153     acall instructiuni
154     acall delay
155     mov a,R0 ; punem in a valoarea temperaturii de referinta
156     mov b,#0ah ; punem in b 10
157     div ab; facem impartirea pt a obtine zecile
158     add a,#30h ; adaugam 30 pt a aobtime valoarea ascii
159     acall date
160     acall delay
161     mov a,b ; mutam in a ceea ce ne-a ramas in b in urma impartirii
162     add a,#30h ; obtinem valoareaa ascii
163     acall date
164     acall delay
165     ret

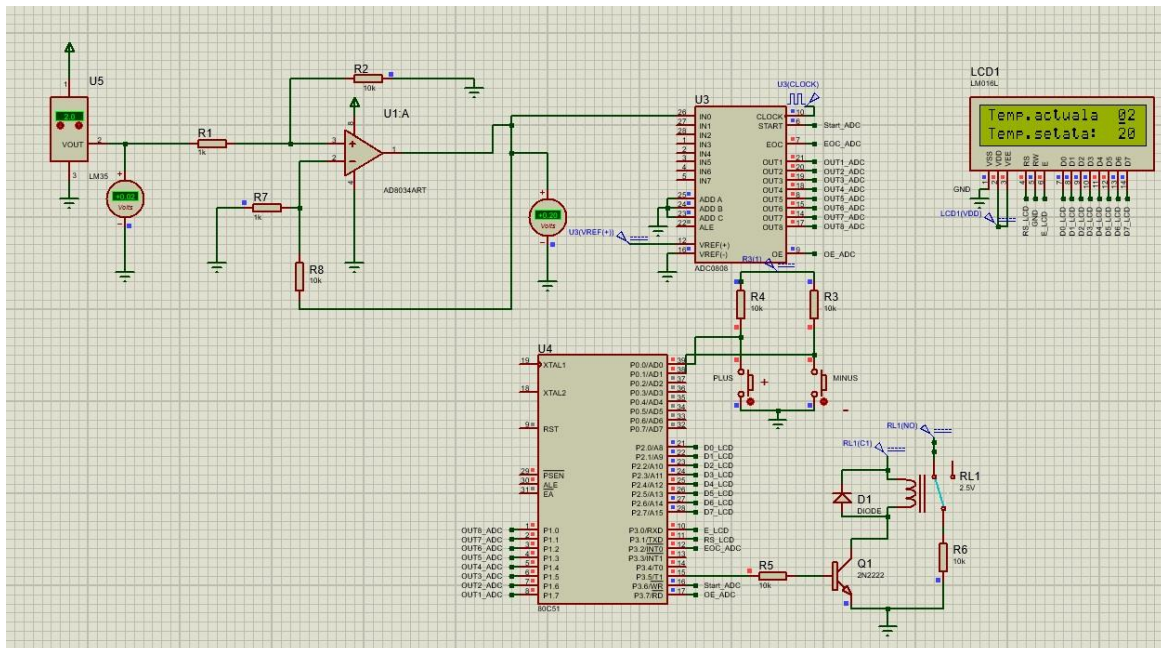
167 //Tastatura
168 temp_control:
169     jnb P0.0,Buton1 ;Se verifica daca "+" este apasat
170     jnb P0.1,Buton2 ;Se verifica daca "-" este apasat
171     sjmp gata
172
173 Buton1:
174     mov r3,#255 ; call delay
175     debouncing_1:
176         jb P0.0,Buton1
177         djnz r3,debouncing_1
178         mov a, r0; punem in a temp de referinta
179         subb a, #51 ; facem o verificare ca temp sa nu fie mai mare decat 51, adica val maxima pe care o pot avea in urma convertirii
180         jz gata ; daca diferenta e zero, inseamna ca am citit 51 de grade si nu mai putem incrementa
181         inc R0; altfel, incrementeaza valoarea de referinta
182         acall show_set_temp
183
184     b1:
185         jnb P0.0,b1 ; numai cand nu mai e apasat butonul se iese din eticheta Buton1
186         sjmp gata
187
188 Buton2:
189     mov r3,#255
190     debouncing_2:
191         jb P0.1,Buton2
192         djnz r3,debouncing_2
193         mov a, r0
194         jz gata
195         dec R0
196         acall show_set_temp
197     b2:jnb P0.1,b2
198         jmp gata
199     gata: ret

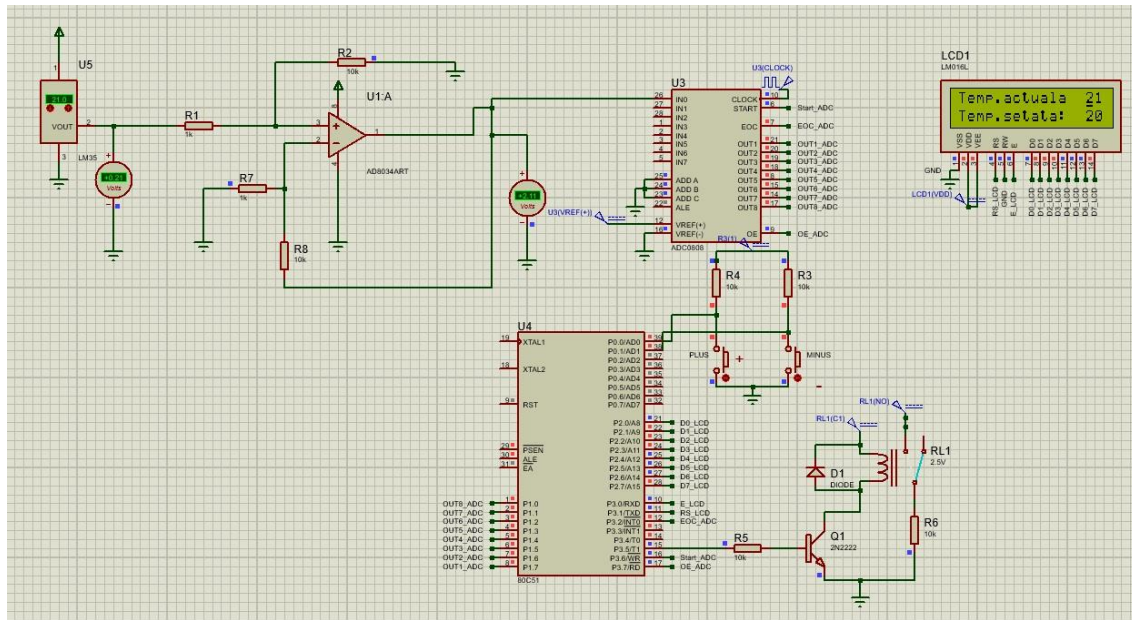
```

```

201 //Releu
202 relay:
203     push acc
204     mov a,r6 ;Valoarea citita o punem in a
205     subb a,R0 ; scadem din valoarea citita, valoarea actuala
206     jc start ; Se verifica daca temperatura setata este mai mare decat cea actuala
207     sjmp oprire
208 start:
209     setb P3.5 ; Pornire releu
210     sjmp gata2
211 oprire:
212     clr P3.5;Oprire releu
213 gata2: pop acc
214     ret
215
216 //Intarzierea
217 delay:
218     mov R7,#229 ; 229 - the value which will multiply the x time delay
219     repeat:
220         nop ; 6 cycles -1 for each nop and 2 for djnz - => 6 * 1.085 us = 6.51us delay for one loop
221         nop
222         nop
223         nop
224         nop
225         djnz R7,repeat ; 6.51 us * 229 = 1.5 ms total delay
226     ret
227
228 org 0500h
229     db "Temp.setata:"
230
231 org 0900h
232     db 'Temp.actuala', 00h
233
234
235
236 end

```





Code in C

```

1  #include <reg51.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4
5  #define dataADC P1          //define variable "dataADC" for port P0
6  #define dataLCD P2          //define variable "dataADC" for port P2
7
8  sbit rs = P3^1;             //register select (LCD)
9  sbit rw = P3^2;             //read/write (LCD)
10 sbit en = P3^0;             //enable signal (LCD)
11
12 //sbit commonpin = P3^3;      //common pin (matrix keypad)
13 sbit tplus = P0^0;           // button increase temperature (matrix keypad)
14 sbit tminus = P3^1;          // button decrease temperature (matrix keypad)
15 //sbit tset = P3^6;           // button set temperature (matrix keypad)
16
17 sbit relay = P3^5;           // relay switch signal
18
19 int tempint = 25;             //default Temperature Set value
20 int tempread;                 //temperature read from sensor
21 unsigned char tempchar[5];    //string variable to display temperature
22
23
24 void delay(unsigned int d);   //delay function
25 unsigned int read_keypad();    //test if button pushed for increase/decrease temperature
26 unsigned int relay_switch();   //test Temperature Set button
27 void lcd_cmd(unsigned char val); //command LCD
28 void lcd_init();              //initialize LCD
29 void lcd_data(unsigned char val); //send a byte of data to LCD
30 void lcd_msg(unsigned char s[]); // display a string of characters on LCD
31 unsigned int read_adc();       //reads a 8bit value from ADC
32 void dispval(unsigned int x);  //display an integer value

```

```

34 void main(){
35     relay=0;//relay initially off
36     dataLCD= 0x00; //output port
37     lcd_init();//initialize LCD
38     while(1){
39         tempread=read_adc();//read temperature from sensor
40         if(read_keypad()==1){//increase TempSet value
41             tempint++;
42             delay(75);
43         }
44         else
45             if(read_keypad()==2){//decrease TempSet value
46                 tempint--;
47                 delay(75);
48             }
49         lcd_cmd(0x80); //first line of lcd display
50         lcd_msg("Temp.actuala: ");
51         dispval(tempint);//display temp set value
52         lcd_msg("C ");
53         lcd_cmd(0xC0); //second line
54         lcd_msg("Temp.setata: ");//display temp read value
55         dispval(tempread);
56         lcd_msg("C ");
57
58         if(relay_switch()==1&&tempint<=tempread){//switch off relay
59             relay=0;
60             delay(1);
61         }
62         else if(relay_switch()==1&&tempint>tempread){//switch on relay
63             delay(1);
64             relay=1;
65         }
66     }
}

70 unsigned int i,j;
71 for(i = 0; i <= d; i++)
72     for(j = 0; j <= 200; j++);
73 }
74 unsigned int read_keypad()//test if button pushed for increase/decrease temperature
75 {
76     unsigned int j=0;
77     tplus=tminus=1;//input pins
78     commonpin= 0;
79     if(tplus==0){//button "TempIncrease" pushed -> return 1
80         j=1;
81     }
82     if(tminus==0){//button "TempDecrease" pushed -> return 2
83         j=2;
84     }
85     return j;
86 }
87 unsigned int relay_switch()//test Temperature Set button
88 {
89     tset=1;//input pin
90     commonpin= 0;
91     if(tset==0){// if tset pressed returns 1
92         return 1;
93     }
94     else
95         return 0;
96 }
97 void lcd_cmd(unsigned char val){// command LCD function
98     dataLCD = val; //send command to LCD
99     rs = 0; //command register
100    rw = 0; //write
101    en = 1; //enable for latching data to LCD from uC (high to low)
102    delay(1); //delay

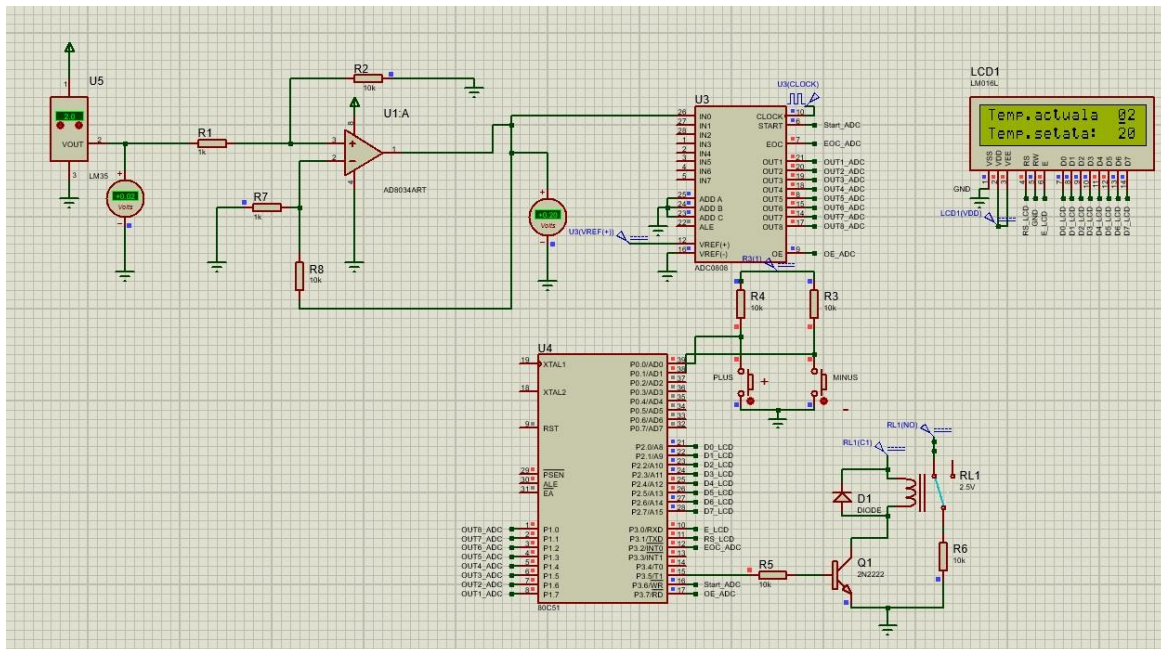
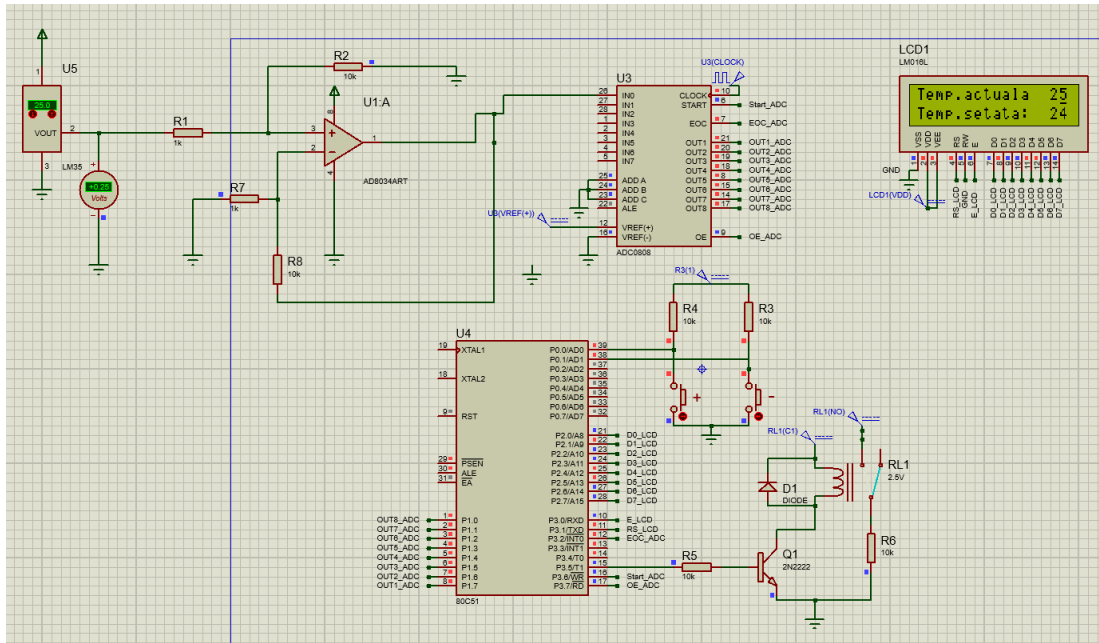
```

```

103 |   en = 0;
104 | }
105 | void lcd_init() { // initialize LCD
106 |     lcd_cmd(0x38); // 2 lines and 5x7 matrix
107 |     delay(1);
108 |     lcd_cmd(0x01); // clear display
109 |     delay(1);
110 |     lcd_cmd(0x0C); // display on and cursor off
111 |     delay(1);
112 |     lcd_cmd(0x80); // set first line as first location of LCD
113 |     delay(1);
114 |     lcd_cmd(0x06); // auto increment cursor
115 |     delay(1);
116 | }
117 | void lcd_data(unsigned char val) { // send a byte of data to LCD
118 |     dataLCD = val; // send command to LCD
119 |     rs = 1; // data register
120 |     rw = 0; // write
121 |     en = 1; // enable for latch data to LCD from uC (high to low)
122 |     delay(1); // delay
123 |     en = 0;
124 | }
125 | void lcd_msg(unsigned char s[]) { // display a string of characters on LCD
126 |     unsigned int i=0;
127 |     while(s[i] != '\0') {
128 |         lcd_data(s[i]);
129 |         i++;
130 |     }
131 | }

131 | }
132 | unsigned int read_adc() { // reads a 8bit value from ADC
133 |     unsigned int t;
134 |     dataADC=0xFF; // port 0 input
135 |     rd=1; // read disable
136 |     intr=1;
137 |     wr=0;
138 |     wr=1; // low to high -> start conversion
139 |     while(intr==1); // intr=0 -> end of conversion
140 |     rd=0; // enable read -> high to low
141 |     t=dataADC; // mov data from output of ADC (port 0 for microcontroller) to variable t
142 |     return t;
143 | }
144 | void dispval(unsigned int x) { // display an integer value
145 |     sprintf(tempchar, "%d", x);
146 |     lcd_msg(tempchar);
147 | }

```



Bibliography

- <https://www.ametherm.com/blog/thermistors/temperature-sensor-types>
- <https://atlas-scientific.com/blog/types-of-temperature-sensors/>
- <https://www.digikey.com/en/blog/types-of-temperature-sensors>
- <https://www.elprocus.com/temperature-sensors-types-working-operation/>
- <https://www.ti.com/lit/ds/symlink/lm35.pdf>
- <https://www.digikey.co.il/htmldatasheets/production/3396943/0/0/1/th02.html>
- <https://dwmzone.com/en/humidity-and-temperature/58-th02-humidity-and-temperature-sensor.html>
- <https://www.electronicwings.com/sensors-modules/lm35-temperature-sensor>