

Microcontrollers Project

Thermostat

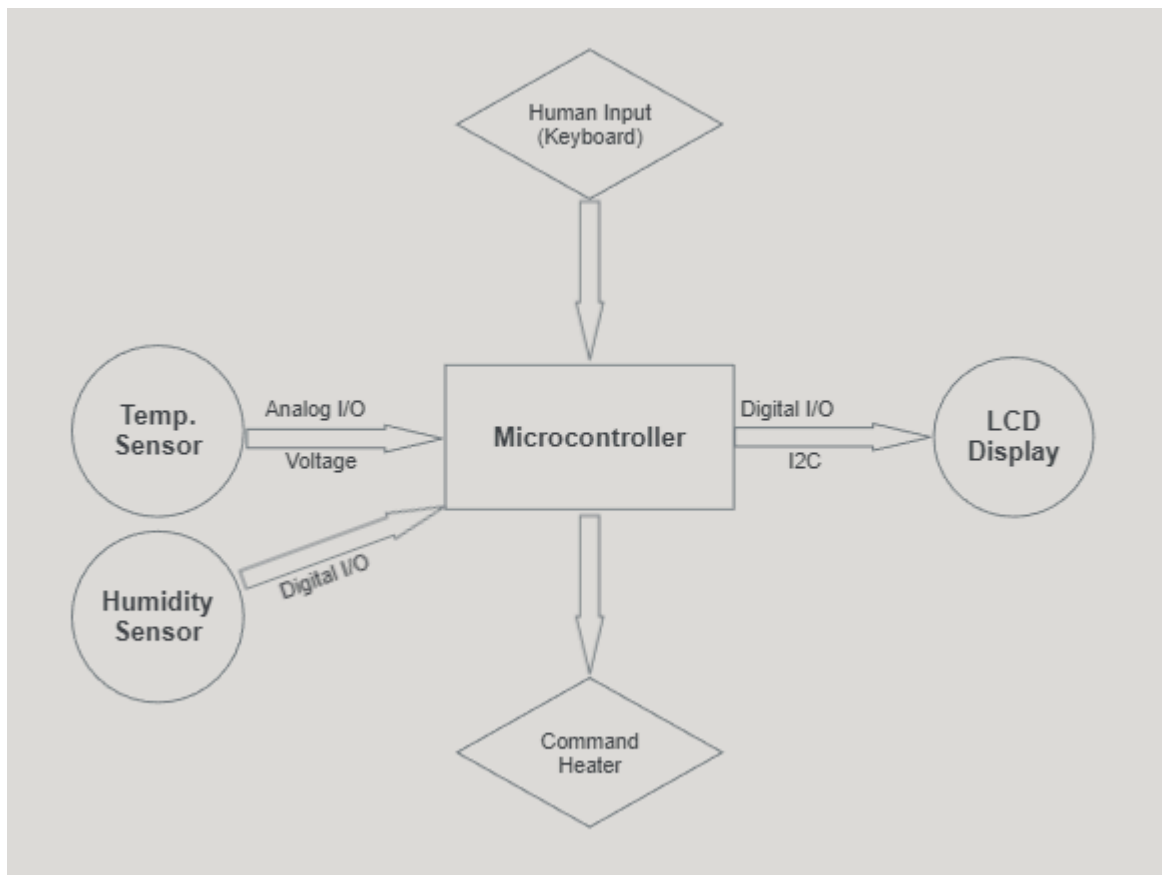
Axinte Octavian-Constantin

Table of Contents

1	Block Diagram and Working Principle	4
2	Types of temperature sensors	5
2.1	Thermocouples	5
2.2	Resistance Temperature Detector (RTD)	5
2.3	Thermistor	6
2.4	Integrated Circuit Temperature Sensor	7
3	Integrated Circuit Temperature Sensors	7
4	Types of Humidity Sensors	8
4.1	Capacitive Humidity Sensors	8
4.2	Resistive Humidity Sensors	9
4.3	Thermal Conductivity Humidity Sensors.....	10
5	Capacitive Humidity Sensors:	10
6	Sensor to Microcontroller Connection	11
6.1	Block Diagram	11
6.2	Temperature Sensor.....	11
6.3	Amplifier.....	11
6.4	Analog to Digital Convertor.....	12
6.5	Electrical Schematic	14
6.6	Simulation	15
7	LCD	16
7.1	Working Principle	17
7.2	LCD 16*2	17
7.3	Types of LCD.....	19
7.4	LM016L LCD	20
7.5	Display data on the LM016L LCD	21
7.6	Simulation	22
8	Microcontroller.....	24
8.1	Definition	24
8.2	Components.....	24
8.3	8051 Microcontroller.....	25
8.4	Comparison of 5 8051 Microcontrollers	26
8.5	AT89S52	27
9	Electrical Schematic	29

9.1	Proteus Implementation.....	29
9.2	Programming Logic.....	30
9.3	Assembly Code	35
9.4	C Code.....	41
10	References:.....	46

1 Block Diagram and Working Principle

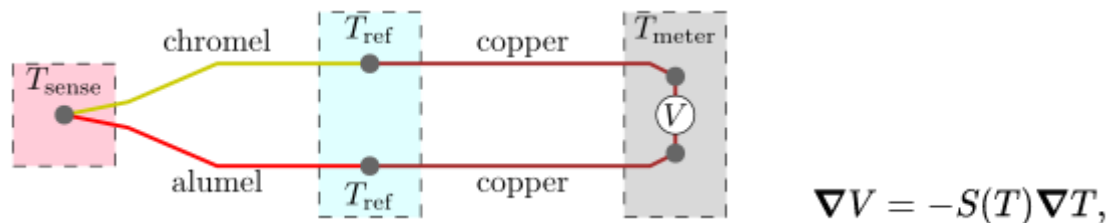


Firstly, the temperature sensor measures the temperature constantly and sends the information to the microcontroller. The microcontroller displays it on the LCD. If the user wants a greater or lower temperature, he will introduce it as input for the microcontroller and it will take a decision based on the current temperature and wanted one. The response will be sent to the heater.

2 Types of temperature sensors

2.1 Thermocouples

This sensor consists of two dissimilar metal wires, joined at one end, and connected to a thermocouple thermometer or other thermocouple-capable device at the other end. 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. It is this voltage difference that can be measured and used to calculate the temperature.



They offer **lower accuracy**, but they do work across **wider temperature ranges** than any of the other temperature sensors.

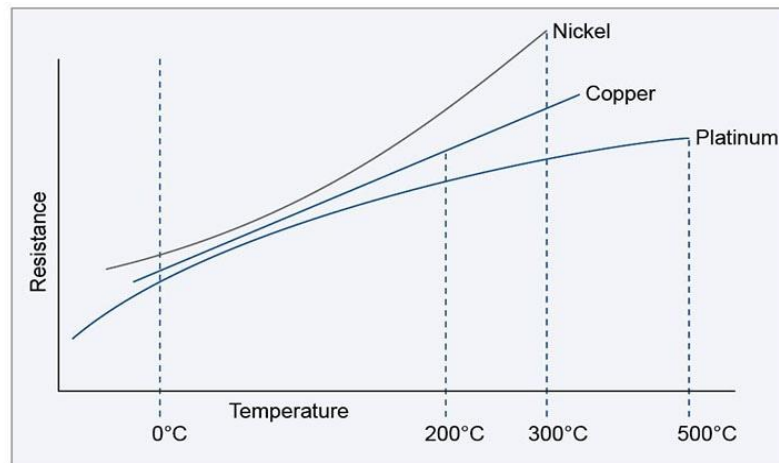
These sensors are **highly durable and cost-effective** and are important because they can work in **many different** applications - from an industrial usage thermocouple to a regular thermocouple found on utilities and regular appliances.

2.2 Resistance Temperature Detector (RTD)

An RTD (Resistance Temperature Detector) is a sensor whose resistance changes as its temperature changes. The resistance increases as the temperature of the sensor increases. An RTD is a resistor with well-defined resistance vs. temperature characteristics. Platinum is the most common and accurate material used to make RTDs.

They provide **the greatest accuracy** and are generally **the most expensive**. Resistance temperature detectors are best when high levels of accuracy are needed.

Two wire sensors are typically used in applications where accuracy is not critical. The two wire configuration allows for the simplest measurement technique, but suffers from an inherent inaccuracy due to the resistance of the sensor leads. For greater accuracy three wire sensors or four wire sensors are better.

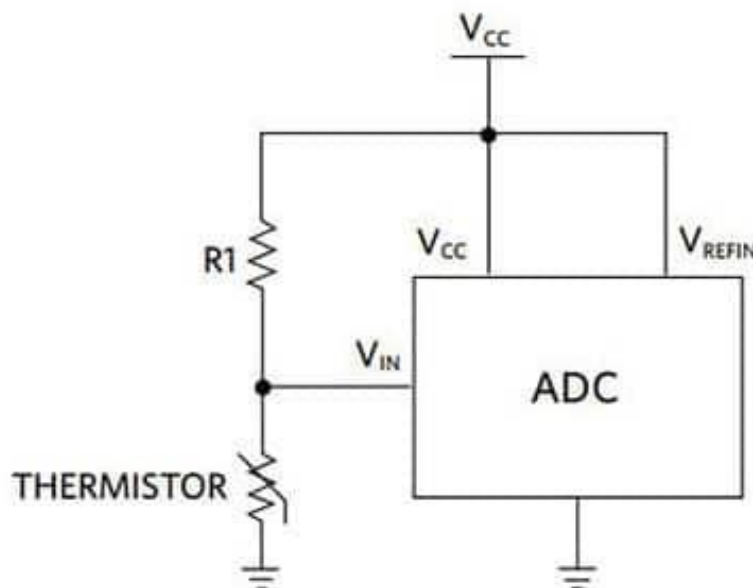


RTD Resistance versus Temperature

2.3 Thermistor

Thermistors are similar to RTDs in that temperature changes cause measurable resistance changes. Thermistors are usually made from a polymer or ceramic material. In most cases, thermistors are **cheaper** but are also **less accurate than RTDs**.

The NTC (Negative Temperature Coefficient) thermistor is the most commonly used thermistor for temperature measurement application. An NTC thermistor's resistance decreases as the temperature increases. Thermistors have a non-linear temperature resistance relationship. This requires a significant correction to interpret the data correctly. A common approach of using a thermistor, shown in the figure below, is where a thermistor and a fixed value resistor form a voltage divider with an output that is digitized by an ADC.



2.4 Integrated Circuit Temperature Sensor

An IC Temperature Sensor is a two terminal integrated circuit temperature transducer that produces an output current proportional to absolute temperature. The sensor package is small with a low thermal mass and a fast response time. It is the most **linear**, **small size** and **inexpensive**, but it requires **power supply** and it is **self-heating**, usually supports a **maximum of 200 Celsius** degrees.

3 Integrated Circuit Temperature Sensors

Name	Temperature Range [°C]	Price (Lei)	Availability: Mouser, Digi-key, Farnell	Supply Voltage [V]	Accuracy [°C]
TMP116*	-55 <-> 125	15-18	Yes Yes No	1.9 – 5.5	+/- 0.3
BME280*	-40 <-> 85	30 - 33	Yes Yes No	1.7 – 3.6	+/- 0.5
Winsen ZS05*	-20 <-> 65	-	No No No	3.3 – 5.5	+/- 1
TMP275AIDGKT	-40 <-> 125	22	No No No	2.7 – 5.5	+/- 1
BMP180*	0<->65	15	Cleste.ro	1.8 – 3.6	+/- 0.5 (at 25)
LMT87	-50 <-> 150	7.42	Yes Yes No	2.7 – 5.5	+/- 0.4

*TMP116 Interface I2C and SMBus. + EPROM

*Bosch BME280 Interface I2C and SPI + Humidity and Pressure

*BMP180 – Interface I2C

*ZS05 – Interface I2C

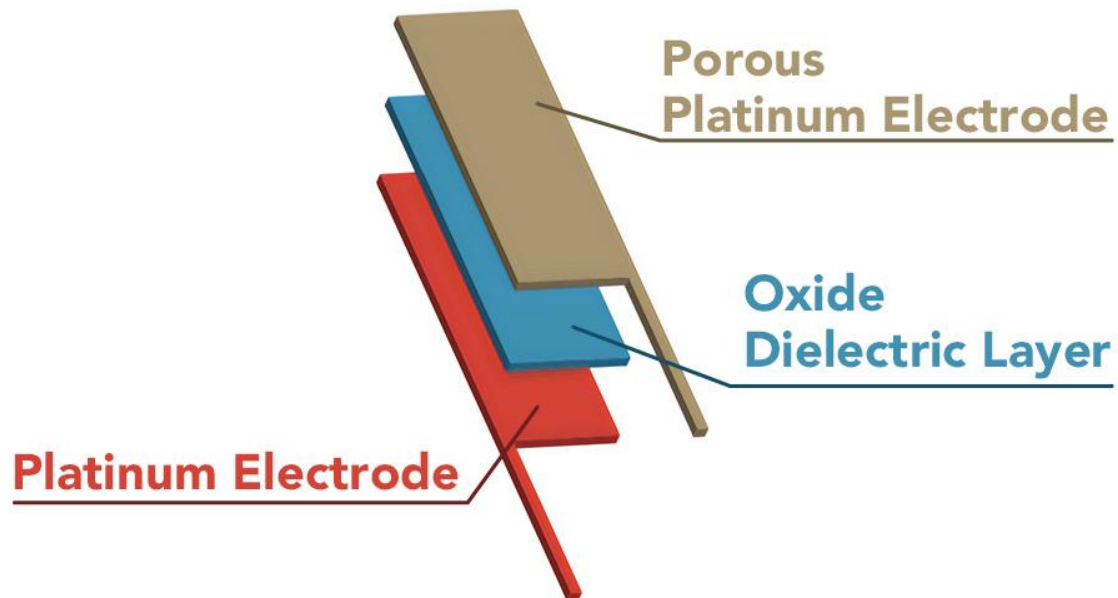
*LMT87 – Analogue output – Voltage output inversely proportional to temperature

I chose **LMT87** since it provides all the necessary conditions at the best price point.

4 Types of Humidity Sensors

4.1 Capacitive Humidity Sensors

It is estimated that 75% of humidity sensors follow the capacitive technique. These humidity sensor types rely on electrical capacitance to provide the user with a humidity value.



Capacitive relative humidity (RH) sensors consist of two metal electrode layers between a dielectric (non-conductive) material, typically a polymer film with a dielectric constant of around 2-15. The dielectric film inside the capacitive humidity sensor attracts and absorbs moisture from the surrounding air. Once the moisture contacts the electrodes, a voltage change occurs.

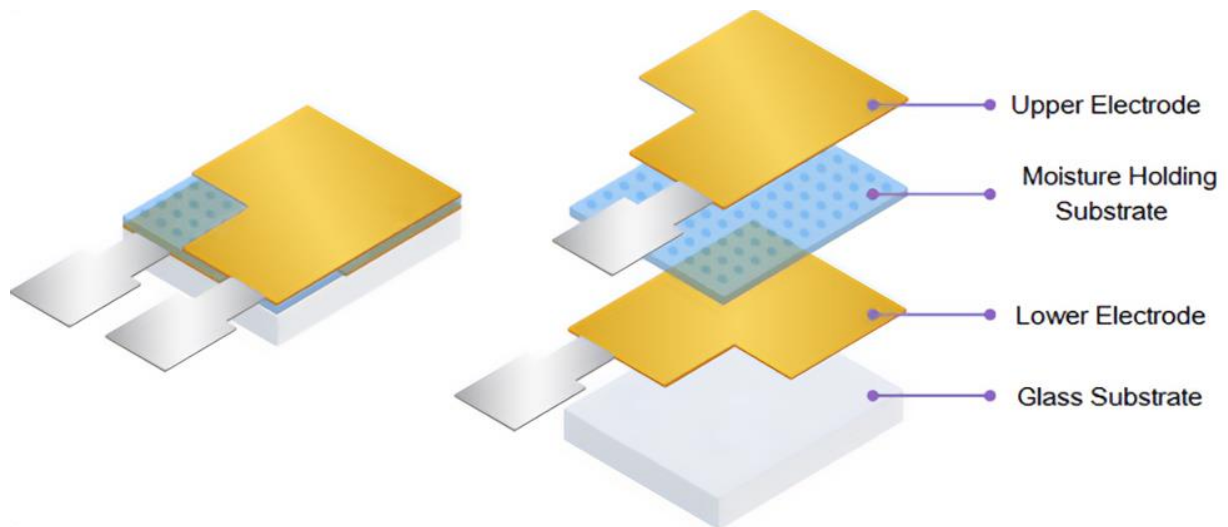
In capacitive humidity sensors, there is a direct relationship between the RH (relative humidity) of the surrounding air, the amount of moisture in the dielectric material, and the capacitance (dielectric constant) of the humidity sensor. The change in the dielectric constant is directly proportional to the RH, therefore, by measuring the dielectric constant, the RH can be calculated.

Pros: wide measurement range, almost linear output voltage, low cost, little maintenance

Cons: bad accuracy at low RH, limited distance between the uC and the sensor.

4.2 Resistive Humidity Sensors

Resistive humidity sensors, also known as electrical conductivity sensors, measure the change in resistivity between two electrodes inside a humidity probe (connected to the sensor) to establish relative humidity.



They have a similar principle to capacitive sensors; an electrical change is measured, producing an RH value. However, resistive humidity sensors use a moisture-absorbing (hygroscopic) material, so their operation principle is slightly different. The output voltage has an inverse exponential relationship to RH. As more water vapor is absorbed, the resistivity decreases due to an increase in the non-metallic conductivity material's conductivity.

Pros: low cost, small footprint, highly interchangeable, big distance between sensor and uC

Cons: sensitive to contaminants, bad accuracy at low RH

4.3 Thermal Conductivity Humidity Sensors

These types of sensors measure the absolute humidity (AH) of the surrounding air/environment by calculating the difference between thermal conductivity in dry air vs humid air. A thermal conductivity humidity sensor consists of two matched negative temperature coefficient (NTC) thermistor elements, suspended by thin wires, in a bridge circuit. One thermistor is located in an exposed chamber via several ventilation holes, exposing it to the surrounding environment. The second is hermetically encapsulated in dry nitrogen, and located in a different section within the humidity sensor.

An electrical circuit passes a current between the two thermistors, resulting in the thermistors self-heating; resistive heating increases the sensor's temperature. When one of the thermistors is exposed to humid air, the conductivity changes. The difference in resistance between the two thermistors (bridge circuit) is directly proportional to absolute humidity.

Pros: resistant, can be used in high-temperature, high-corrosive environments, great resolution

Cons: exposure to certain gases can affect humidity readings

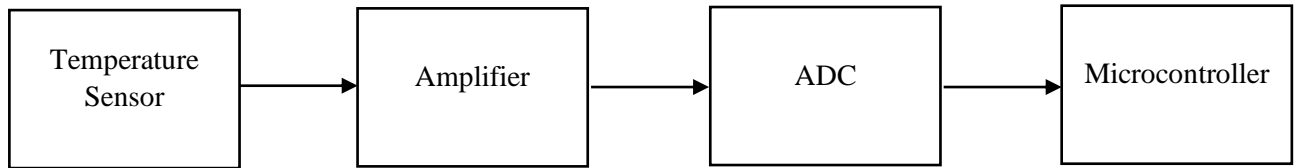
5 Capacitive Humidity Sensors:

Name	Rel. Humidity Range [%]	Price (Lei)	Availability: Mouser, Digi-key, Farnell	Supply Voltage [V]	Accuracy [%]
SHT21*	0 <-> 100	36 - 40	Yes Yes No	1.9 – 5.5	+/- 2
BME280*	0<-> 100	30 - 33	Yes Yes No	1.7 – 3.6	+/- 3
Winsen ZS05*	0 <-> 100	-	No No No	3.3 – 5.5	+/- 5
SHT40I-AD1B-R2	0 <-> 100	14.76	Yes Yes Yes	2.3 – 5.5	+/- 2
HPP845E031R4	0<-> 100	35-36	Yes Yes No	1.5 – 3.6	+/- 3

I chose SHT40I-AD1B-R2 since it is way cheaper than the rest, it is easy to find and has good accuracy. It is a digital output sensor, I2C communication channel.

6 Sensor to Microcontroller Connection

6.1 Block Diagram



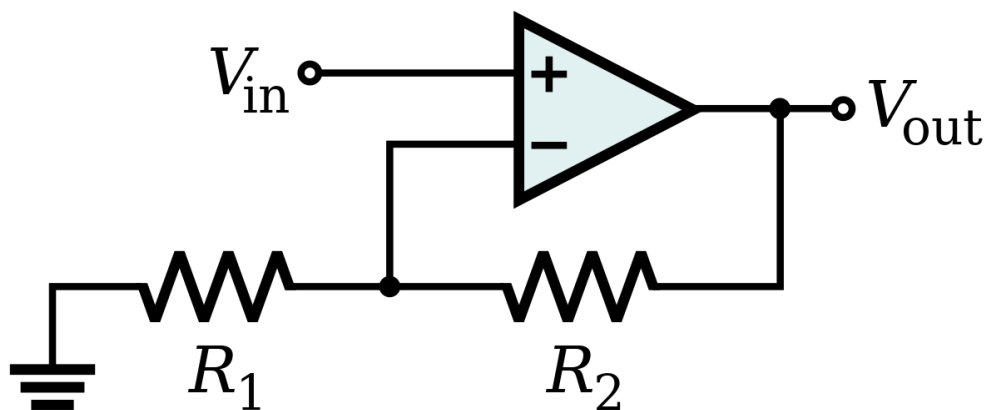
6.2 Temperature Sensor

The analogue temperature sensor has a voltage output depending on the temperature. I chose the LM35 temperature sensor for simulation due to Proteus' component availability. It is quite similar to the LMT87 I chose for implementation, although the main difference is that the output voltage is directly proportional to the temperature.

The temperature sensor LM35 has at the output $10\text{mV}/^{\circ}\text{C}$. For example, if the LM35 is measuring a temperature of 20°C , its output voltage will be 200 mV ($20^{\circ}\text{C} \times 10\text{ mV}/^{\circ}\text{C} = 200\text{ mV}$). If the temperature changes to 21°C , the output voltage will change to 210 mV ($21^{\circ}\text{C} \times 10\text{ mV}/^{\circ}\text{C} = 210\text{ mV}$), which represents a linear change in the output voltage in response to the change in temperature.

6.3 Amplifier

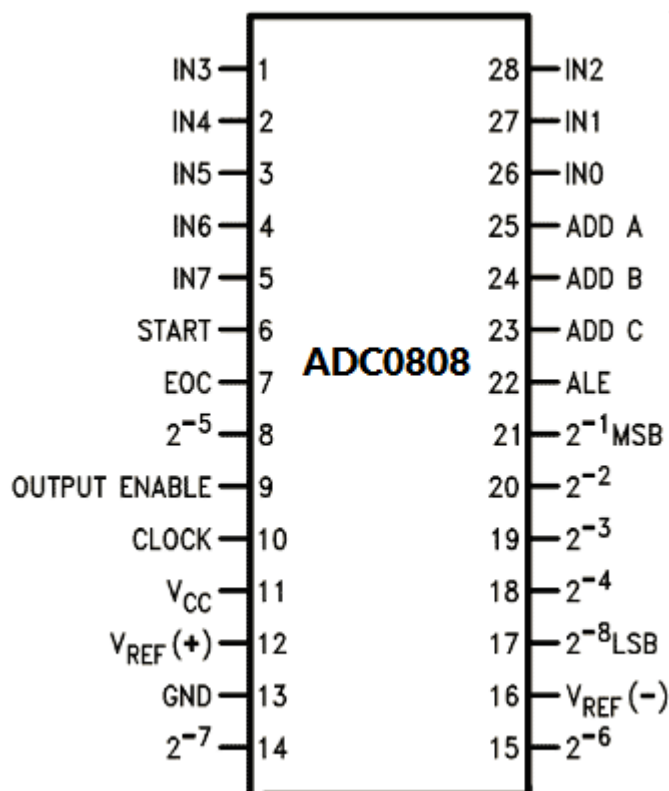
The amplifier is needed to increase the signal output of the sensor so that the ADC can offer distinct, reliable values for each temperature reading received. I implemented it using an operational amplifier in non-inverting configuration.



For simulation I chose the AD822P, a single-supply, rail-to-rail low power FET-Input OpAmp. The main criteria are:

- Offset Voltage: typical 0.1mV
It is very important so that at 0 °C even after amplification the ADC will still transmit 0. This value for offset voltage is quite a small one.
- Input Bias Current and Input Offset Current: typical 2pA
We want it to be as small as possible for the same reasons as before.
- Output Voltage Swing: typically extends to within 10mV of each rail.
Really useful to choose a rail-to-rail OpAmp so that the supply can be as low as possible.

6.4 Analog to Digital Convertor



An analog-to-digital converter changes an analog signal that's continuous in terms of both time and amplitude to a digital signal that's discrete in terms of both time and amplitude. The conversion involves quantization of the input, so it necessarily introduces a small amount of error or noise. Furthermore, instead of continuously performing the conversion, an ADC does the conversion periodically, sampling the input, limiting the allowable bandwidth of the input signal.

The ADC used is an ADC0808 since it is a reliable choice, maybe even too complex for the needs of the project but is one of the only options in Proteus. The resolution is of 8 bits,

conversion time of 100 us and single supply equal to 5V. The ADC0808 chip is designed with an 8-channel multiplexer, which allows it to select one of eight analog input channels to convert. It also has an internal clock that controls the sampling and conversion of the analog input signal. The chip uses a successive approximation technique for conversion, which involves comparing the input voltage with an internal reference voltage and gradually narrowing down the range until the output is an 8-bit digital value.

The output data of the ADC0808 is presented in parallel format, with each bit being represented by a corresponding output pin. The chip also includes a start conversion (SC) input pin, which initiates the conversion process, and an end of conversion (EOC) output pin, which indicates when the conversion is complete, and the digital output is available.

I chose to connect the output of the amplifier is to the IN0 (input 0) pin of the ADC0808 because it allows the analogue signal to be converted into a digital value that can be processed by a microcontroller or other digital device. IN0 is the first input channel of the multiplexer in the ADC0808, which selects the channel that will be converted.

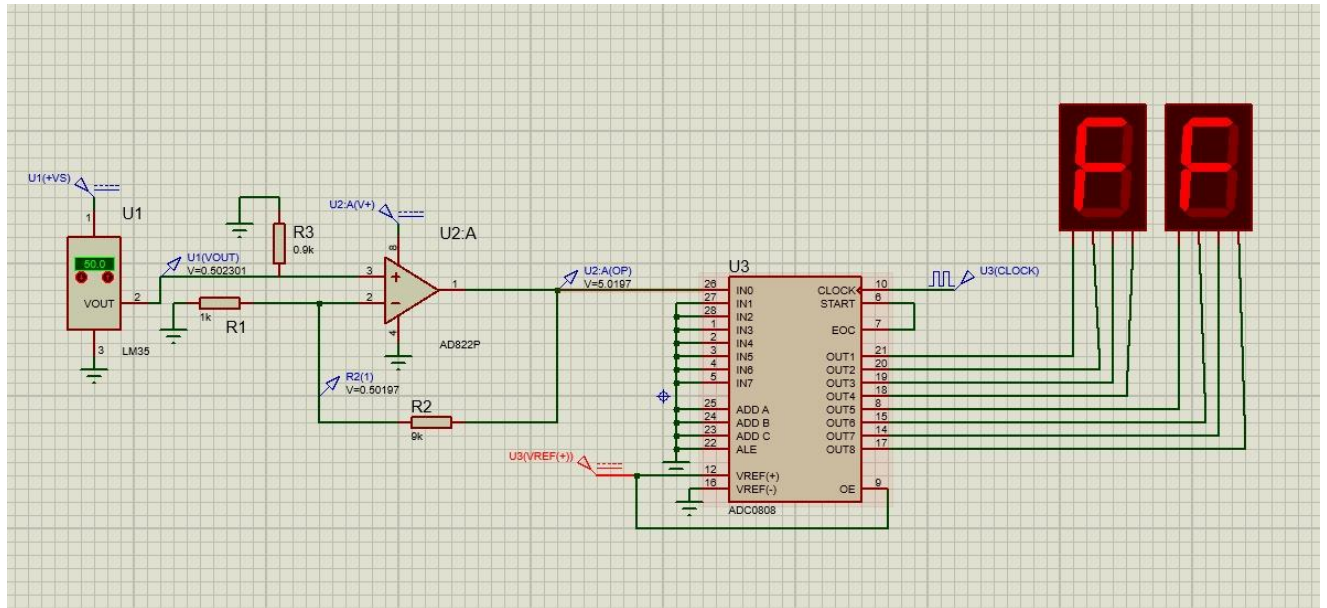
Pins 1-7 on the ADC0808 are not used in the basic mode of operation and are left unconnected (or "free") in most applications. In some advanced applications, these pins may be used for other functions, such as setting up a differential input mode, testing and calibration, or interfacing with other devices. However, in most cases, they are left unconnected to simplify the circuit design and reduce the potential for errors or interference.

By grounding the AD A, AD B, and AD C pins, the corresponding address bits are set to 0, which selects the default mode of operation. The default mode of operation for the ADC0808 is a single-ended, 8-channel multiplexed operation with an internal clock frequency of 640 kHz. In this mode, the ADC0808 sequentially samples and converts each of the eight analogue input channels, starting with channel 0 (IN0) and ending with channel 7 (IN7).

The OE(Output Enable) pin can be used to control the timing of the output data and to avoid any contention issues with other devices that may be connected to the same bus. For example, if multiple devices are connected to the same bus, the OE pin can be used to ensure that the ADC0808 only drives the bus when its data is needed and avoids any conflicts with other devices that may also be trying to drive the bus at the same time.

In summary, the OE pin of the ADC0808 is used to enable or disable the output data from the device and can be used to control the timing and avoid contention issues when multiple devices are connected to the same bus.

6.5 Electrical Schematic



The ADC working principle together with the temperature sensor will dictate the dimensioning of the amplifier's components.

$$N = \frac{V}{V_{max}} * N_{max}$$

N = number from the output of the ADC

V = analogue input voltage in ADC

$V_{max} = 5V - 1LSB$ (power supply for the ADC)

$$V_{LSB} = \frac{V_{FS}}{2^n} \Rightarrow V_{LSB} = \frac{5}{2^8} = 20mV$$

$N_{max} = 2^n = 2^8 = 255$ (maximum number generated by the converter)

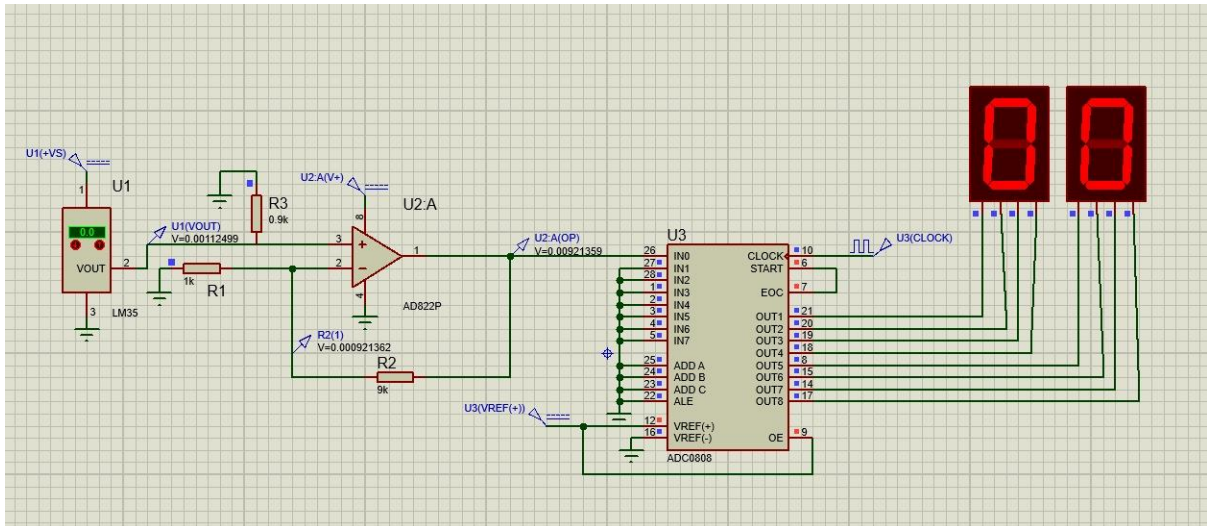
In order to obtain $N = N_{max}$ the analog input voltage of the ADC should be equal with the power supply. So at the maximum temperature of 50 °C the input voltage must be 5V so that on the 7-segment display it will be shown FFh.

In order to obtain 5V at the input of the ADC, knowing that at the output of the sensor for a temperature of 50 °C it will be 500mV it results that we need an amplification of 10.

The functioning equation of the non-inverting amplifier is: $V_{out} = (1 + \frac{R_2}{R_1}) * V_{in}$

It result that $\frac{R_2}{R_1} = 9$ and I have chosen $R_2 = 9k$ and $R_1 = 1k$

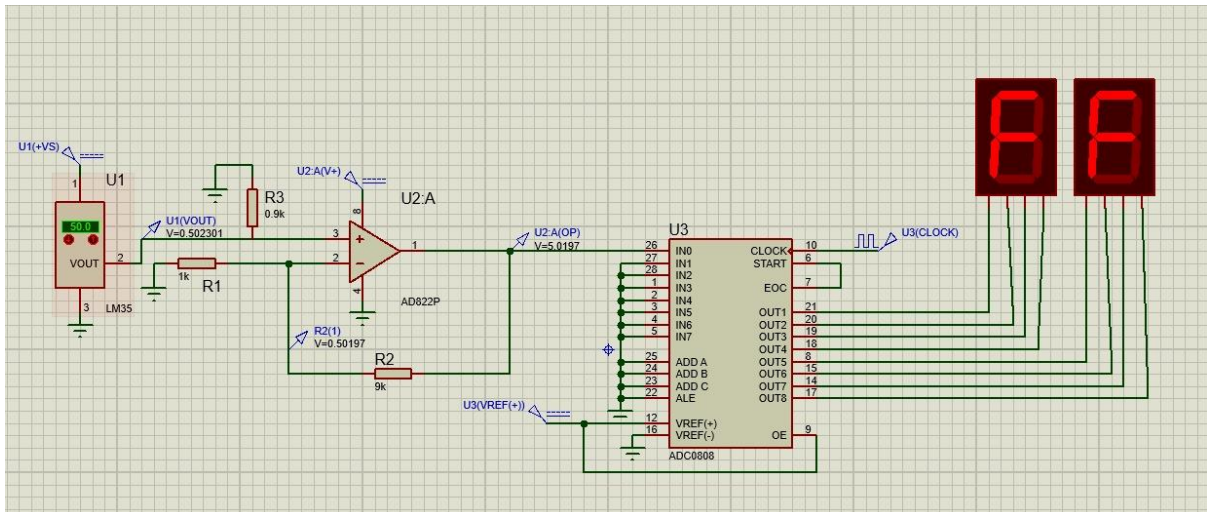
6.6 Simulation



For a temperature of 0 °C the output of the ADC is 0.

$V_{temp} = 0.0011 \Rightarrow V_{outAMP} = 0.011$

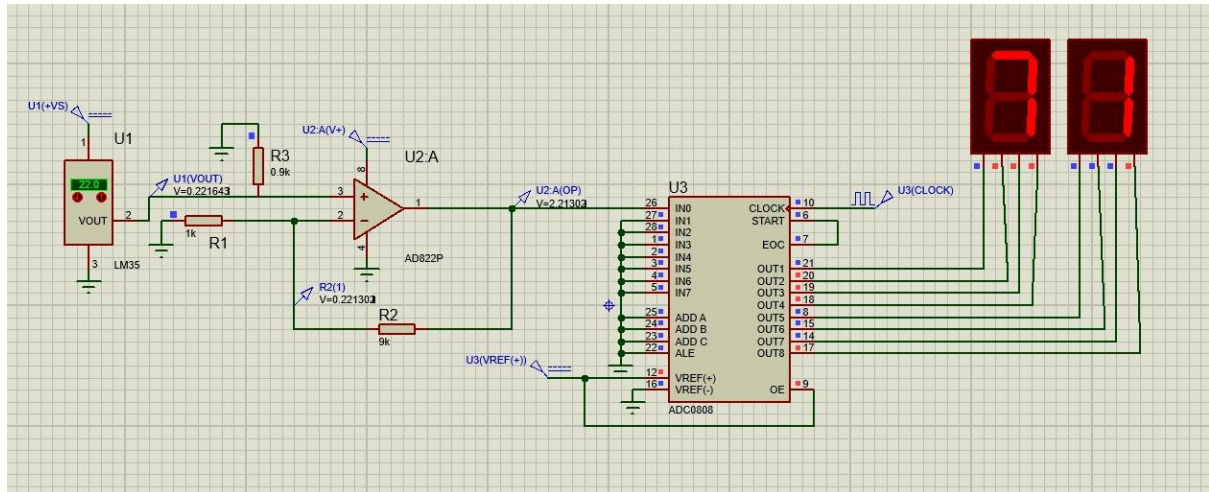
$$N = \frac{V}{V_{max}} * N_{max} = \frac{0.011}{4.98} * 256 = 0.56 \Rightarrow N = 0$$



For a temperature of 50 °C the output of the ADC is 256 (FFh).

$V_{temp} = 0.502 \Rightarrow V_{outAMP} = 5.02$

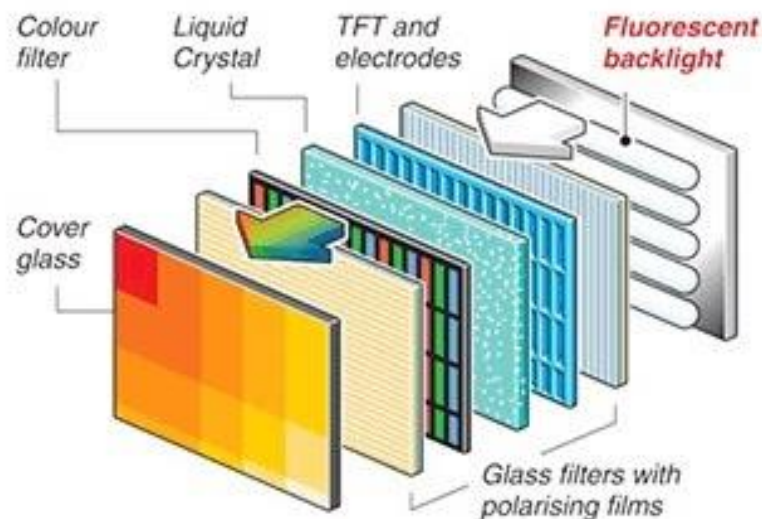
$$N = \frac{V}{V_{max}} * N_{max} = \frac{5.02}{4.98} * 256 = 257.08 \Rightarrow N = 256 \text{ (256 is maximum)}$$



For a typical room temperature of 22 °C the output of the ADC is 113 (71h), almost in the middle.

7 LCD

A liquid-crystal display (LCD) is a flat-panel display or other electronically modulated optical device that uses the light-modulating properties of liquid crystals combined with polarizers. Liquid crystals do not emit light directly but instead use a backlight or reflector to produce images in colour or monochrome.



7.1 Working Principle

An LCD (Liquid Crystal Display) is a type of flat-panel display that uses liquid crystals to produce images. Here are the basic steps of how an LCD works:

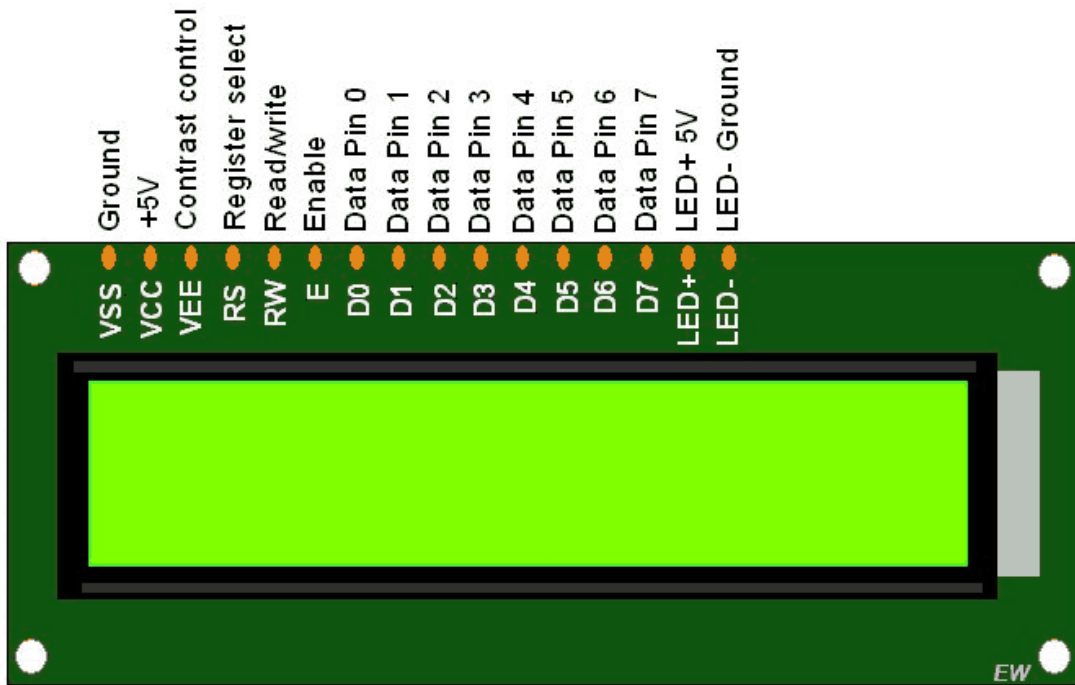
1. Light source: The first step in the process is to provide a light source behind the LCD panel. This can be a backlight or a sidelight, depending on the type of LCD.
2. Polarization: The light then passes through a polarizing filter, which aligns the light waves in a single direction.
3. Liquid crystals: The light then passes through a layer of liquid crystals, which are molecules that can change the direction of the polarized light depending on the electrical charge applied to them.
4. Voltage applied: When a voltage is applied to the liquid crystals, they align themselves in a way that either allows the polarized light to pass through or blocks it.
5. Colour filter: The next step is to add a colour filter layer on top of the liquid crystals. This layer consists of tiny red, green, and blue filters that create the full-colour spectrum.
6. Displaying images: The final step is to control the voltage applied to each pixel to create the desired image. By varying the voltage to each pixel, different amounts of light are allowed to pass through, creating the illusion of colour and brightness.

Overall, the LCD works by manipulating the polarized light passing through the liquid crystals to create an image that is displayed on the screen.

7.2 LCD 16*2

An LCD 16x2 (also known as a 16 character by 2 line display) is a common type of alphanumeric LCD display module that is widely used in a variety of electronic devices, such as digital clocks, calculators, and consumer electronics.

The LCD 16x2 has a rectangular shape, typically measuring 80mm x 36mm, with a display area of 64mm x 16mm. It has 16 columns and 2 rows of characters, with each character consisting of 5x8 pixels. The module is typically controlled by a microcontroller or other digital device, which sends commands to the display to control the content and appearance of the characters.



Here's a brief description of the function of each pin:

- VSS: Ground pin
- VCC: Power supply pin (usually +5V)
- VEE: Contrast adjustment pin (can be used to adjust the contrast of the characters on the screen)
- RS: Register Select pin (used to select whether the data being sent to the display is a command or data)
- RW: Read/Write pin (used to select whether data is being written to or read from the display)
- E: Enable pin (used to enable the display for data transfer)
- D0-D7: Data pins (used to transfer data to and from the display)
- LED+: Anode pin (used to power the backlight)
- LED-: Cathode pin (ground for the backlight)

7.3 Types of LCD

There are several types of LCD (Liquid Crystal Display) technology that are commonly used in electronic devices. Here are some of the most common types:

- **TN** (Twisted Nematic)

TN displays are the most common type of LCD display. They are relatively inexpensive and offer fast response times and low power consumption. However, they have limited viewing angles and colour reproduction.

- **IPS** (In-Plane Switching)

IPS displays offer wider viewing angles and better colour reproduction than TN displays. They are commonly used in high-end smartphones, tablets, and computer monitors.

- **VA** (Vertical Alignment)

VA displays offer high contrast ratios and deep blacks, making them well-suited for use in televisions and monitors. However, they have relatively slow response times and limited viewing angles.

- **MVA** (Multi-Domain Vertical Alignment)

MVA displays offer improved viewing angles and response times compared to VA displays, making them well-suited for use in high-end monitors and televisions.

- **OLED** (Organic Light-Emitting Diode)

OLED displays use organic compounds to emit light when an electric current is applied. They offer deep blacks, wide viewing angles, and fast response times, making them well-suited for use in smartphones, televisions, and other devices that require high-quality displays.

- **AMOLED** (Active-Matrix Organic Light-Emitting Diode)

AMOLED displays are a type of OLED display that uses a thin-film transistor (TFT) to control the flow of current to each pixel. They offer faster response times and better colour reproduction than traditional OLED displays, making them well-suited for use in high-end smartphones and other devices.

Overall, the type of LCD technology used in a device depends on the specific application and desired characteristics of the display, such as viewing angles, colour reproduction, response time, and power consumption.

7.4 LM016L LCD

The LM016L is a common type of 16x2 character LCD display module that is widely used in a variety of electronic devices. This is the LCD used for simulation.

Here's an overview of its features and specifications:

- Display type: STN (Super Twisted Nematic)
- Display format: 16 characters x 2 lines
- Character size: 5x8 dots
- Display area: 64.5mm x 14.5mm
- Viewing angle: 6 o'clock
- Backlight type: LED
- Backlight colour: Blue or Green
- Operating voltage: 5V DC
- Operating temperature: -20°C to +70°C
- Controller: HD44780-compatible
- Interface: 4-bit or 8-bit parallel

The LM016L display module typically includes a built-in controller that is compatible with the HD44780 standard, which is a widely used standard for interfacing with LCD displays. This makes it relatively easy to control the display using a microcontroller or other digital device.

The LM016L display module can be interfaced with a microcontroller or other digital device using either a 4-bit or 8-bit parallel interface. This allows for flexible interfacing and easy integration into a wide range of electronic devices.

Overall, the LM016L display module is a versatile and widely used display solution that provides a simple and efficient way to display alphanumeric characters in a variety of electronic devices.

7.5 Display data on the LM016L LCD

These are the commands executed by the HD44780 controller in order to display data on the LA016L LCD.

Table 13 8-Bit Operation, 8-Digit × 2-Line Display Example with Internal Reset

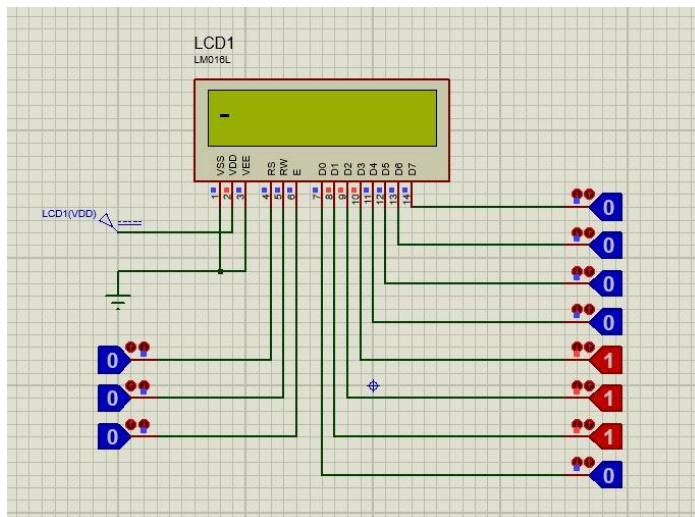
Step No.	Instruction										Display	Operation
1	Power supply on (the HD44780U is initialized by the internal reset circuit)											Initialized. No display.
2	Function set 0 0 0 0 1 1 1 0 * *											Sets to 8-bit operation and selects 2-line display and 5 × 8 dot character font.
3	Display on/off control 0 0 0 0 0 0 1 1 1 0											Turns on display and cursor. All display is in space mode because of initialization.
4	Entry mode set 0 0 0 0 0 0 0 1 1 0											Sets mode to increment the address by one and to shift the cursor to the right at the time of write to the DD/CGRAM. Display is not shifted.
5	Write data to CGRAM/DDRAM 1 0 0 1 0 0 1 0 0 0										H	Writes H. DDRAM has already been selected by initialization when the power was turned on. The cursor is incremented by one and shifted to the right.
6												
7	Write data to CGRAM/DDRAM 1 0 0 1 0 0 1 0 0 1										HITACHI	Writes I.
8	Set DDRAM address 0 0 1 1 0 0 0 0 0 0										HITACHI	Sets DDRAM address so that the cursor is positioned at the head of the second line.

HD44780U

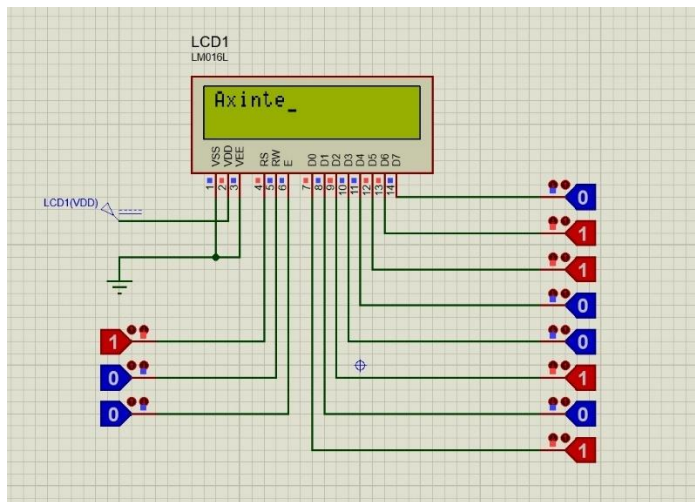
Table 13 8-Bit Operation, 8-Digit × 2-Line Display Example with Internal Reset (cont)

Step No.	Instruction										Display	Operation
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
9	Write data to CGRAM/DDRAM										<div>HITACHI</div> <div>M_</div>	Writes M.
	1	0	0	1	0	0	1	1	0	1		
10											<div>.</div> <div>.</div> <div>.</div> <div>.</div>	
11	Write data to CGRAM/DDRAM										<div>HITACHI</div> <div>MICROCO_</div>	Writes O.
	1	0	0	1	0	0	1	1	1	1		
12	Entry mode set										<div>HITACHI</div> <div>MICROCO_</div>	Sets mode to shift display at the time of write.
	0	0	0	0	0	0	0	1	1	1		
13	Write data to CGRAM/DDRAM										<div>ITACHI</div> <div>ICROCOM_</div>	Writes M. Display is shifted to the left. The first and second lines both shift at the same time.
	1	0	0	1	0	0	1	1	0	1		
14											<div>.</div> <div>.</div> <div>.</div> <div>.</div>	
15	Return home										<div>HITACHI</div> <div>MICROCOM</div>	Returns both display and cursor to the original position (address 0).
	0	0	0	0	0	0	0	0	1	0		

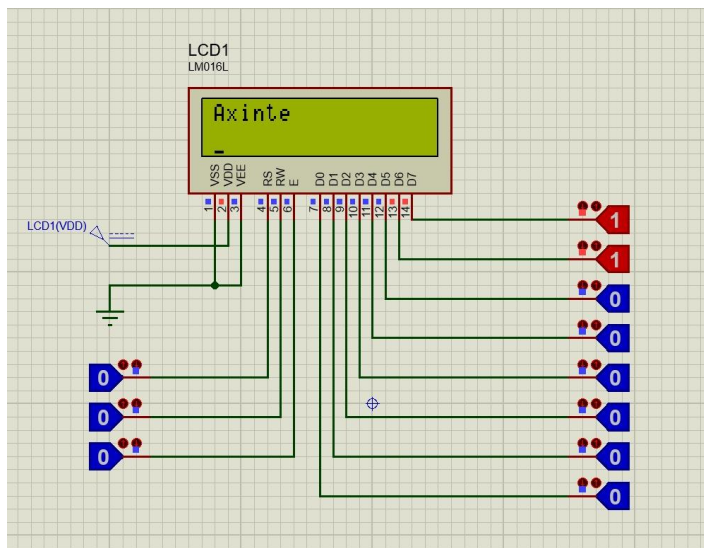
7.6 Simulation



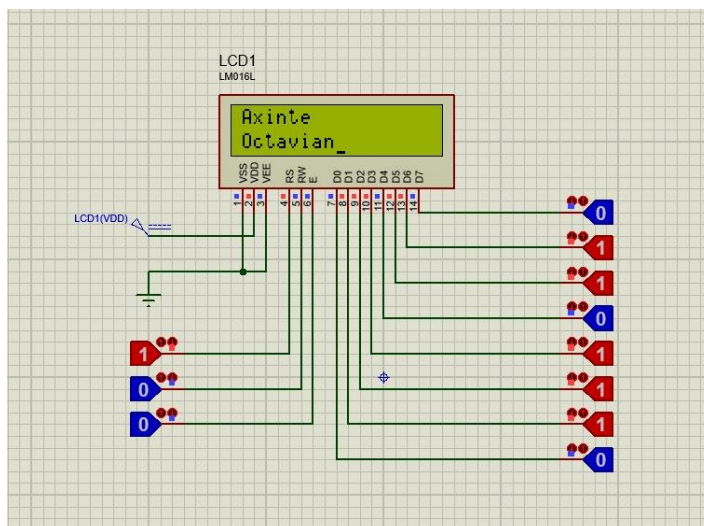
Cursor turned on ^



Last name on the first row ^



Cursor on the second row ^



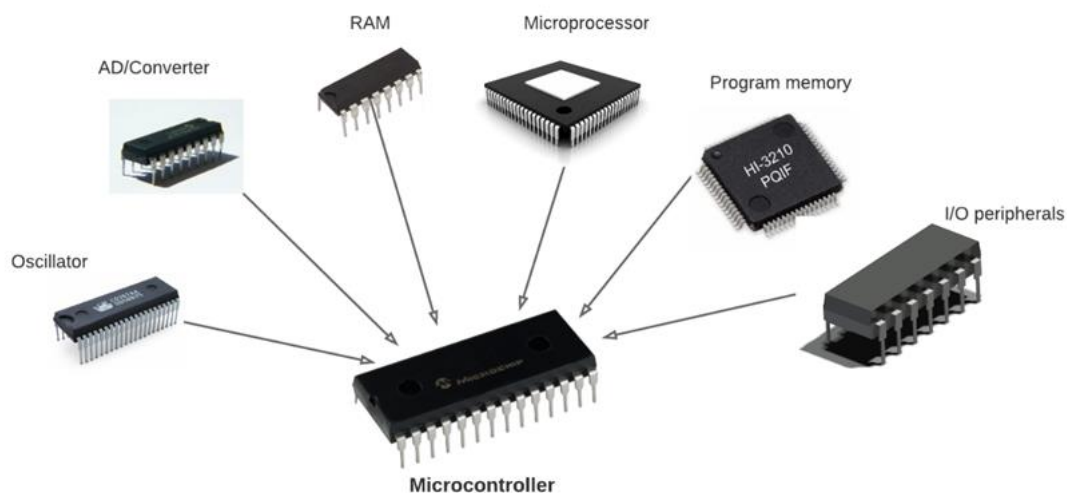
First name on the second row ^

8 Microcontroller

8.1 Definition

A microcontroller is a small computer on a single integrated circuit (IC) chip that contains a processor core, memory, and input/output peripherals. It is designed to control specific devices or systems and is used in a wide range of electronic devices, from home appliances to industrial machines.

8.2 Components



- Processor Core

The central processing unit (CPU) is the heart of the microcontroller, responsible for executing instructions and performing calculations.

- Memory

The microcontroller typically contains two types of memory - Random Access Memory (RAM) for data storage and Read-Only Memory (ROM) for storing the program code.

- Input/Output Peripherals

These components are responsible for interfacing with external devices and sensors. Examples include digital and analogue input/output pins, timers, serial communication ports, and interrupt controllers.

- Clock

The microcontroller requires a clock to synchronize its operations. This clock is usually provided by an external crystal oscillator or an internal oscillator circuit.

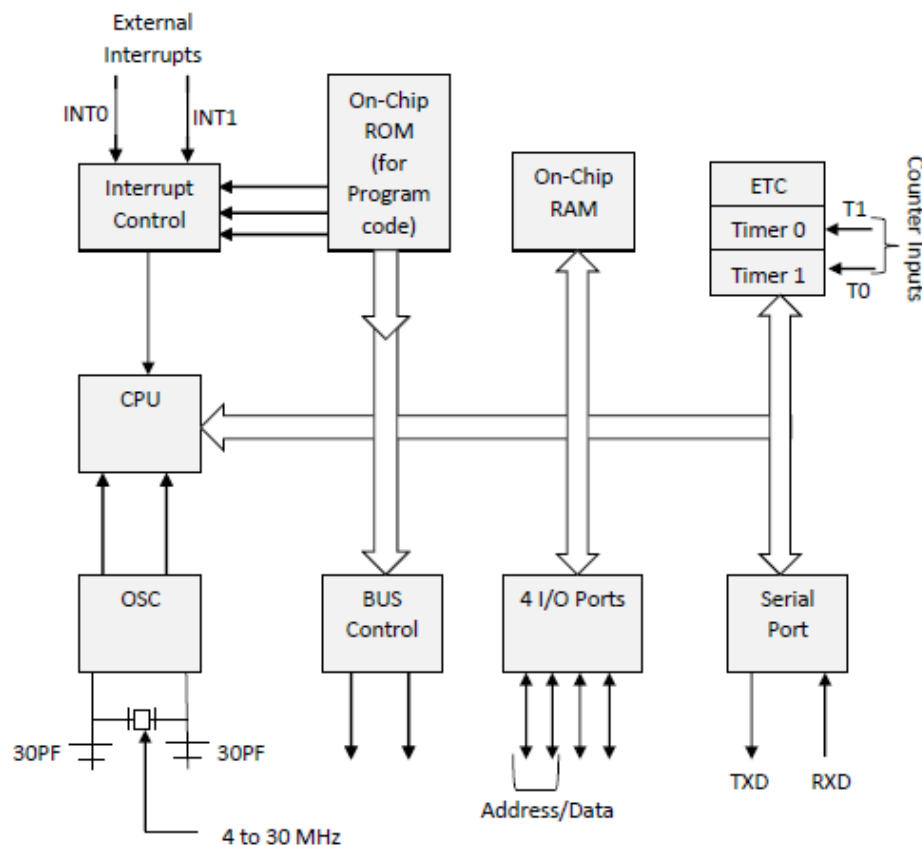
- Power Management Unit

This component regulates the power supply to the microcontroller, ensuring that it operates within safe voltage and current limits.

- Programming/Debugging Interface

This interface allows developers to program the microcontroller and debug the code running on it. It typically consists of a set of pins that connect to a programmer/debugger device.

8.3 8051 Microcontroller



The 8051 microcontroller is a family of microcontrollers that were first introduced by Intel in 1980. It is one of the most widely used microcontrollers in the world, and its popularity has led to a large number of manufacturers producing compatible chips.

The 8051 family microcontrollers have a simple architecture and are easy to program. They typically have a 8-bit data bus and 16-bit address bus, and run at speeds of up to 33 MHz.

Some of the key features of the 8051 microcontroller family include:

- On-chip memory:

The 8051 microcontrollers typically have on-chip memory, including RAM, ROM, and EEPROM, which makes them suitable for a wide range of applications.

- Interrupts:

The 8051 microcontrollers have a flexible interrupt structure, which allows for the handling of external events in real-time.

- Timers/counters:

The 8051 microcontrollers typically have a number of timers/counters that can be used for a range of applications, including generating PWM signals and measuring pulse widths.

- Serial communication:

The 8051 microcontrollers typically have one or more serial communication ports, which can be used to interface with other devices.

- GPIO pins:

The 8051 microcontrollers typically have a number of general-purpose input/output (GPIO) pins, which can be used for a range of applications, including controlling LEDs and reading switches.

8.4 Comparison of 5 8051 Microcontrollers

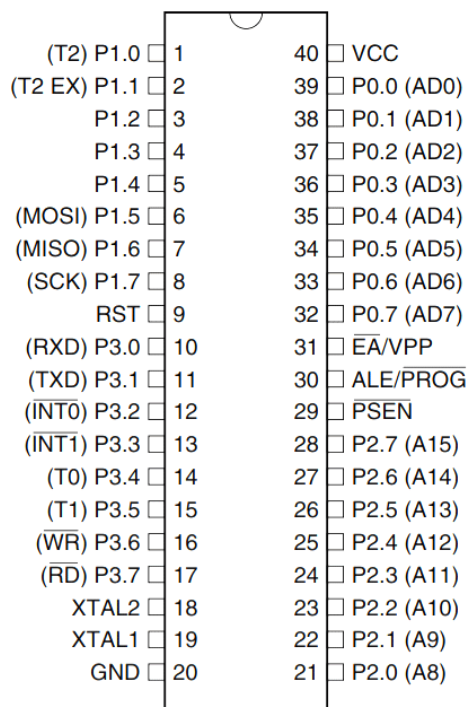
Microcontroller	Manufacturer	Flash Memory	RAM	I/O Pins	Price Lei	Timer/Counters
AT89S51	Atmel	4KB	128 bytes	32	18	2 (16-bit)
AT89S52	Atmel	8KB	256 bytes	32	20	2 (16-bit)
DS89C450	Maxim Integrated	64KB	2KB	32	92	3 (16-bit)
CY8C29466	Cypress Semiconductor	64KB	4KB	42	55	2 (16-bit)

STC89C52RC	STC Microelectronics	32KB	1KB	32	7	2 (16-bit)
------------	----------------------	------	-----	----	---	------------

The application requires small storage and no special functions so the decision is to use AT89S52 due to the low price and availability.

8.5 AT89S52

The AT89S52 microcontroller has a total of 40 pins, each of which serves a specific function. Here's a brief description of the various pins of the microcontroller:



P1.0 - P1.7: These are the eight bidirectional I/O pins of Port 1. They can be configured as input or output pins.

P2.0 - P2.7: These are the eight bidirectional I/O pins of Port 2. They can be configured as input or output pins.

P3.0 - P3.7: These are the eight bidirectional I/O pins of Port 3. They can be configured as input or output pins.

P0.0 - P0.7: These are the eight bidirectional I/O pins of Port 0. They can be configured as input or output pins.

XTAL1 and XTAL2: These are the input and output pins, respectively, of an external crystal oscillator or resonator. They provide the clock signal to the microcontroller.

RST: This is the reset pin of the microcontroller. When this pin is pulled low, the microcontroller resets and starts executing code from the beginning.

ALE: This is the Address Latch Enable pin. It is used to latch the address from the microcontroller onto an external latch.

EA/VPP: This pin is used to select the source of the program memory. When the pin is connected to VCC, the program memory is sourced from an external device. When the pin is connected to ground, the program memory is sourced from the internal flash memory. This pin is also used as the programming voltage input during in-system programming.

PSEN: This is the Program Store Enable pin. It is used to indicate that the microcontroller is accessing program memory.

INT0: This is the external interrupt 0 pin. It is used to trigger an interrupt when a signal is applied to the pin.

INT1: This is the external interrupt 1 pin. It is used to trigger an interrupt when a signal is applied to the pin.

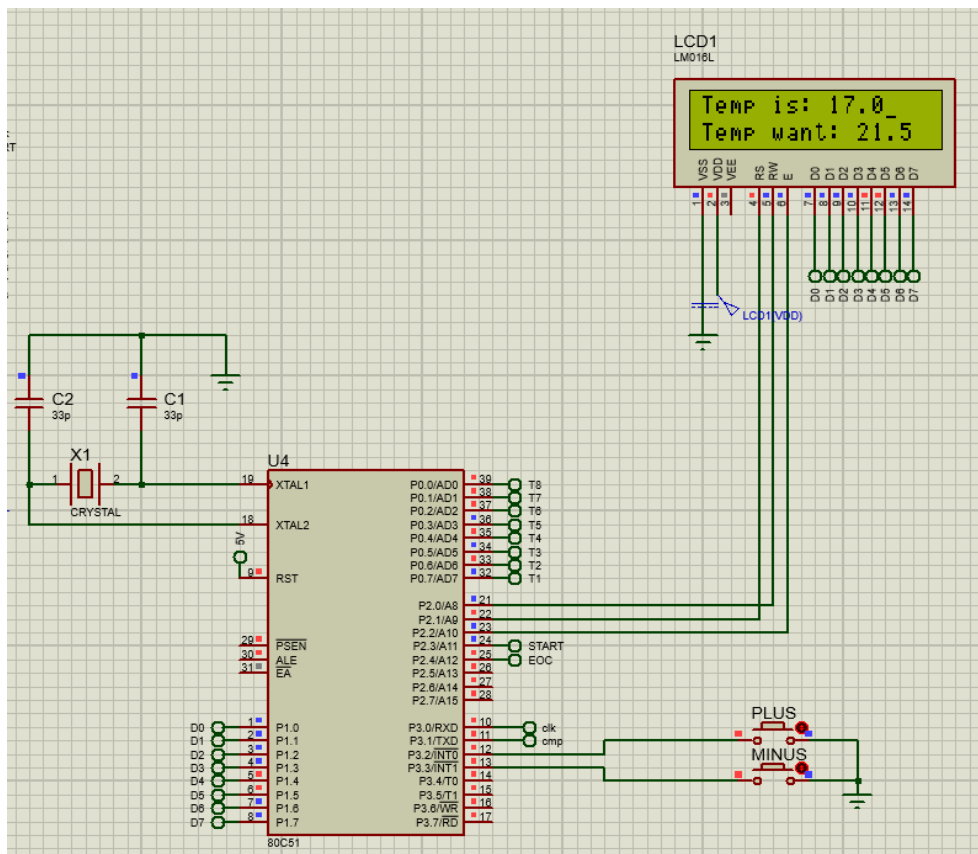
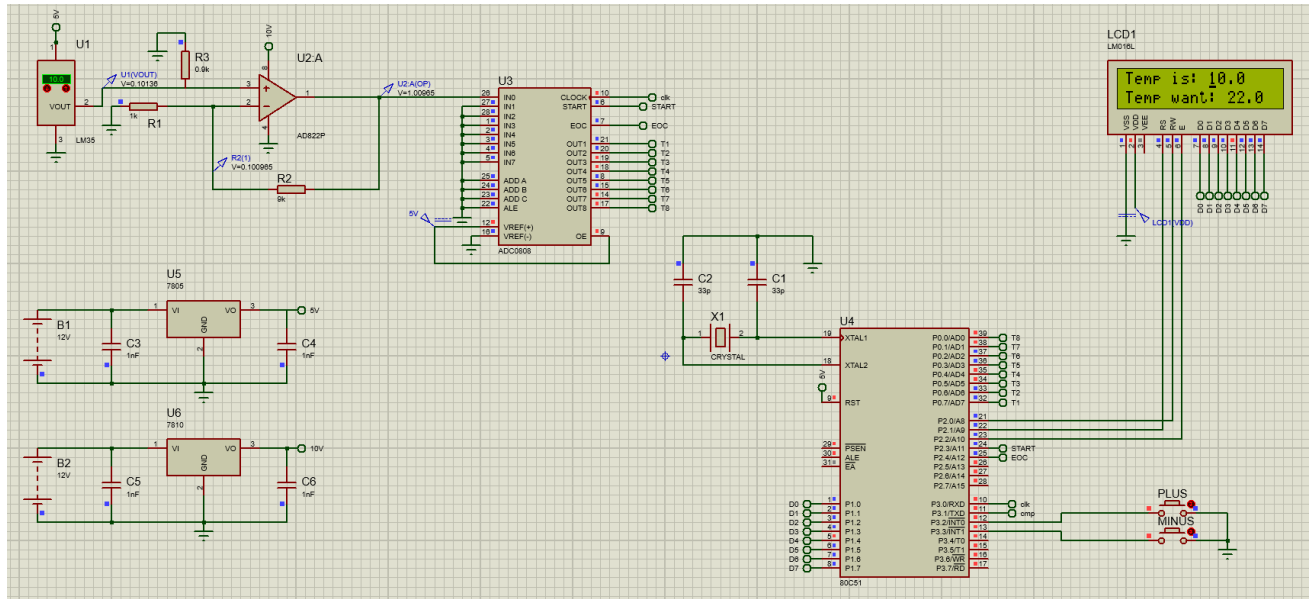
TXD: This is the transmit data pin of the built-in serial communication port.

RXD: This is the receive data pin of the built-in serial communication port.

VCC and GND: These pins provide power supply and ground connections to the microcontroller.

9 Electrical Schematic

9.1 Proteus Implementation



The output pins of the ADC0808 are connected to the first port P0 of the microcontroller. The START and EOC pins are connected to P2.3 and P2.4. The LCD pins are controlled by the first port P1 of the microcontroller. The RS, R/W and E pins are connected to P2.0, P2.1, P2.2. The keyboard represented by the 2 buttons is connected to P3.2, P3.3 (INT0, INT1). The clock signal necessary for the ADC is given by P3.0. The turn on signal for the heating system can be found at P3.1 as “cmp”.

Between XTAL1 and XTAL2 is connected a crystal oscillator with a frequency of 12MHz. Compared to an internal clock, the external crystal oscillator is more precise. The two capacitors have two purposes. The first one is to filter the unwanted frequencies that may be generated by the oscillator. Also, we use two capacitors in order to keep the phase shift unaltered.

The two buttons give the user the possibility to choose a certain wanted temperature. When a button is pressed the port pin will be activated (0 logic) and the required instruction will be executed. The two temperatures will always be compared. If the wanted temperature is greater than the present temperature than the heating system will be powered on.

I also used two LDOs to power on my circuit from a 12V battery. One has a 5V output and the other one has a 10V output.

9.2 Programming Logic

As previously mentioned, the ADC output will be between 00 and FFh for temperatures between 0 and 50 °C. The question is how to display on the LCD the temperature from the sensor using the large domain (256 values) of the ADC output. There are three possible solutions, but only one was implemented.

- Changing the amplification so that at the output of the ADC the domain will be between 0 and 32h (50). In this way, no supplementary operations are needed.
- Using a look-up table assigning for every value from 00 to FFh a certain temperature.
- Dividing the ADC output in such a way that we obtain firstly the tens digit and then the unit digit

I chose the third method since it requires the least steps to apply, despite the more complicated logic.

```

41 LOAD:
42 ACALL READY_CONVERSION
43 MOV A, P3
44 MOV B, #33h
45 DIV AB
46 add A, #30h
47 acall display
48 mov A, B
49 mov B, #5
50 div AB
51 add A, #30h
52 acall display
53 RET

```

We observe that for the 10 °C temperature the output from the ADC is 33h. Knowing that the temperature sensor has a linear characteristic it can also be computed: $FFh : 5 = 33h$.

In order to find the tens digit it is sufficient to divide the ADC output value to 33h. The quotient then is transformed in ASCII code and displayed. The remainder will have values between 0 and 32h (50). In order to have a unit between 0 and 9 then we need to normalise the interval by dividing with 5. The new quotient is transformed in ASCII code and displayed.

```

1 org 100h
2 STR: DB "Temperature is:", 0

16 ;display on LCD the string
17 MOV DPTR, #str
18 l1: MOV A, #00H
19 MOVC A, @A+DPTR
20 JZ done
21 ACALL DISPLAY
22 INC DPTR
23 SJMP l1

26 done:

```

Another important part of the code is the one above. Instead of displaying the characters one by one in code I preferred to place a string in the memory and display it in a loop. The characters will still be displayed one by one on the LCD because it is the only possibility, but the speed will make it appear at the same time for the human eye. This is also done by checking the busy flag of the LCD instead of using a delay.

```

79 ;verify if the LCD displayed the character in order to display the next one
80 READY_LCD:
81 CLR P2.2
82 SETB P1.7
83 CLR P2.1 ;RS=0
84 SETB P2.0 ; R/W = 1 => READ COMMAND REG
85 BACK: CLR P2.2
86     nop
87     nop
88     SETB P2.2
89     JB P1.7, BACK
90 CLR P2.2
91 RET
92
93 ;verify if the conversion of the ADC is done
94 READY_CONVERSION:
95 SETB P2.4 ;eoc
96 NOP
97 SETB P2.3 ; start
98 NOP
99 NOP
100 CLR P2.3
101 BACK2: JNB P2.4, BACK2
102 RET

```

For LCD we check if D7(busy flag) is one so that we can load the next character.

For the ADC we check the EOC bit so that we can start the next conversion.

```

TABLE: DB "00.0","00.5", "01.0", "01.5", "02.0", "02.5", "03.0", "03.5", "04.0", "04.5", "05.0", "05.5", "06.0", "06.5", "07.0", "07.5", "08.0", "08.5", "09.0", "09.5"
        DB "10.0", "10.5", "11.0", "11.5", "12.0", "12.5", "13.0", "13.5", "14.0", "14.5", "15.0", "15.5", "16.0", "16.5", "17.0", "17.5", "18.0", "18.5", "19.0", "19.5"
        DB "20.0", "20.5", "21.0", "21.5", "22.0", "22.5", "23.0", "23.5", "24.0", "24.5", "25.0", "25.5", "26.0", "26.5", "27.0", "27.5", "28.0", "28.5", "29.0", "29.5"
        DB "30.0", "30.5", "31.0", "31.5", "32.0", "32.5", "33.0", "33.5", "34.0", "34.5", "35.0", "35.5", "36.0", "36.5", "37.0", "37.5", "38.0", "38.5", "39.0", "39.5"
        DB "40.0", "40.5", "41.0", "41.5", "42.0", "42.5", "43.0", "43.5", "44.0", "44.5", "45.0", "45.5", "46.0", "46.5", "47.0", "47.5", "48.0", "48.5", "49.0", "49.5"

```

```
MOV r0, #44
```

For the wanted temperature I wanted to use a different technique. A look-up table.

The offset value for the table is saved in R0. I encountered many problems, one of them being that the 8b r0 is not enough. We have 100 possible temperatures, on 4b each one of them => 400 bits, but we only have 256.

The solution was to save only the index of temperature (0-99) in R0 and then when displaying to multiply it with 4 so that we arrive at the real location in the table. The first button implements the “+” option whereas the second one implements the “-” option. For this was necessary a careful computation of offset in the table. The buttons were implemented using interrupts. So the cursor always needed to be placed where it was when the interrupt signal appeared!


```

173     buton1:
174
175
176     MOV A, #0C0h
177     ACALL COMMAND
178     mov a, #11
179     cursor:
180     push acc
181     mov a, #14h
182     ACALL COMMAND
183     pop acc
184     dec a
185     jz out
186     sjmp cursor
187     out:
188
189     MOV DPTR, #TABLE
190     MOV A, r0 ;
191     mov b, #4
192     mul AB
193     push acc
194     MOVC A, @A + DPTR
195     acall display
196     pop acc
197     inc a
198     push acc
199     MOVC A, @A + DPTR
200     acall display
201     pop acc
202     inc a
203     push acc
204     MOVC A, @A + DPTR
205     acall display
206     pop acc
207
208     inc a
209     push acc
210     MOVC A, @A + DPTR
211     acall display
212     pop acc
213     inc a
214     mov b, #4
215     div AB
216     mov r0, a
217
218     MOV A, #80H // force
219     ACALL COMMAND
220     mov a, #9
221     cursor1:
222     push acc
223     mov a, #14h
224     ACALL COMMAND
225     pop acc
226     dec a
227     jz init
228     sjmp cursor1
229     init:
230
231     reti

```

The most difficult part was implementing the comparison between the current temperature and wanted temperature.

For this I saved the digits from the current temperature in R3, R4, R5. Then I compared the MSB of the current temperature with the MSB of the wanted temperature. Since there are limited mnemonics in Keil, I did this by dividing them and assessing the quotient and remainder. If there was equality between them then I compared the next bit, till the LSB. I was always careful to convert from ASCII to numbers and vice-versa. If the wanted temperature is greater than the current temperature, P3.1 will be '1', otherwise P3.0 will be '0'.

```

294 compare:
295
296 mov DPTR, #table
297 mov a, r0
298 mov b, #4
299 mul ab
300 movc a, @a+DPTR
301 subb a, #30h
302 push acc
303 mov a, r3
304 subb a, #30h
305
306 jnz merge
307 mov a, #1
308 merge:
309
310 mov b, a ; current temperaturu
311 pop acc ; wanted temperature
312 div ab
313 jz oprim
314     cjne a, #1, pornim
315     mov a, b
316     cjne a, #0, pornim2
317     ;continuum cu ur
318     mov a, r0
319     mov b, #4
320     mul ab
321     inc a
322     movc a, @a+DPTR
323     subb a, #30h
324     push acc
325     mov a, r4
326     subb a, #30h
327
328     jnz merge2
329     mov a, #1
330 merge2:
331
332     mov b, a
333     pop acc
334     div ab
335     jz oprim2
336
337     cjne a, #1, pornim3
338     mov a, b
339     cjne a, #0, pornim4
340     mov a, r0
341     mov b, #4
342     mul ab
343     inc a
344     inc a
345     inc a
346     movc a, @a+DPTR
347     subb a, #30h
348     push acc
349     mov a, r5
350     subb a, #30h
351
352     jnz merge3
353     mov a, #1
354 merge3:
355
356     mov b, a
357     pop acc
358     div ab
359     jz oprim3
360     cjne a, #1, pornim5
361     mov a, b
362     cjne a, #0, pornim6
363     sjmp oprim4
364
365     mov b, a
366     pop acc
367     div ab
368     jz oprim2
369     cjne a, #1, pornim3
370     mov a, b
371     cjne a, #0, pornim4
372     sjmp oprim4
373
374     mov b, a
375     pop acc
376     div ab
377     jz oprim2
378     cjne a, #1, pornim3
379     mov a, b
380     cjne a, #0, pornim4
381     sjmp oprim4
382
383     mov b, a
384     pop acc
385     div ab
386     jz oprim2
387     cjne a, #1, pornim3
388     mov a, b
389     cjne a, #0, pornim4
390     sjmp oprim4
391
392     mov b, a
393     pop acc
394     div ab
395     jz oprim2
396     cjne a, #1, pornim3
397     mov a, b
398     cjne a, #0, pornim4
399     sjmp oprim4
400
401     mov b, a
402     pop acc
403     div ab
404     jz oprim2
405     cjne a, #1, pornim3
406     mov a, b
407     cjne a, #0, pornim4
408     sjmp oprim4
409
410     mov b, a
411     pop acc
412     div ab
413     jz oprim2
414     cjne a, #1, pornim3
415     mov a, b
416     cjne a, #0, pornim4
417     sjmp oprim4
418
419     mov b, a
420     pop acc
421     div ab
422     jz oprim2
423     cjne a, #1, pornim3
424     mov a, b
425     cjne a, #0, pornim4
426     sjmp oprim4
427
428     mov b, a
429     pop acc
430     div ab
431     jz oprim2
432     cjne a, #1, pornim3
433     mov a, b
434     cjne a, #0, pornim4
435     sjmp oprim4
436
437     mov b, a
438     pop acc
439     div ab
440     jz oprim2
441     cjne a, #1, pornim3
442     mov a, b
443     cjne a, #0, pornim4
444     sjmp oprim4
445
446     mov b, a
447     pop acc
448     div ab
449     jz oprim2
450     cjne a, #1, pornim3
451     mov a, b
452     cjne a, #0, pornim4
453     sjmp oprim4
454
455     mov b, a
456     pop acc
457     div ab
458     jz oprim2
459     cjne a, #1, pornim3
460     mov a, b
461     cjne a, #0, pornim4
462     sjmp oprim4
463
464     mov b, a
465     pop acc
466     div ab
467     jz oprim2
468     cjne a, #1, pornim3
469     mov a, b
470     cjne a, #0, pornim4
471     sjmp oprim4
472
473     mov b, a
474     pop acc
475     div ab
476     jz oprim2
477     cjne a, #1, pornim3
478     mov a, b
479     cjne a, #0, pornim4
480     sjmp oprim4
481
482     mov b, a
483     pop acc
484     div ab
485     jz oprim2
486     cjne a, #1, pornim3
487     mov a, b
488     cjne a, #0, pornim4
489     sjmp oprim4
490
491     mov b, a
492     pop acc
493     div ab
494     jz oprim2
495     cjne a, #1, pornim3
496     mov a, b
497     cjne a, #0, pornim4
498     sjmp oprim4
499
500     mov b, a
501     pop acc
502     div ab
503     jz oprim2
504     cjne a, #1, pornim3
505     mov a, b
506     cjne a, #0, pornim4
507     sjmp oprim4
508
509     mov b, a
510     pop acc
511     div ab
512     jz oprim2
513     cjne a, #1, pornim3
514     mov a, b
515     cjne a, #0, pornim4
516     sjmp oprim4
517
518     mov b, a
519     pop acc
520     div ab
521     jz oprim2
522     cjne a, #1, pornim3
523     mov a, b
524     cjne a, #0, pornim4
525     sjmp oprim4
526
527     mov b, a
528     pop acc
529     div ab
530     jz oprim2
531     cjne a, #1, pornim3
532     mov a, b
533     cjne a, #0, pornim4
534     sjmp oprim4
535
536     mov b, a
537     pop acc
538     div ab
539     jz oprim2
540     cjne a, #1, pornim3
541     mov a, b
542     cjne a, #0, pornim4
543     sjmp oprim4
544
545     mov b, a
546     pop acc
547     div ab
548     jz oprim2
549     cjne a, #1, pornim3
550     mov a, b
551     cjne a, #0, pornim4
552     sjmp oprim4
553
554     mov b, a
555     pop acc
556     div ab
557     jz oprim2
558     cjne a, #1, pornim3
559     mov a, b
560     cjne a, #0, pornim4
561     sjmp oprim4
562
563     mov b, a
564     pop acc
565     div ab
566     jz oprim2
567     cjne a, #1, pornim3
568     mov a, b
569     cjne a, #0, pornim4
570     sjmp oprim4
571
572     mov b, a
573     pop acc
574     div ab
575     jz oprim2
576     cjne a, #1, pornim3
577     mov a, b
578     cjne a, #0, pornim4
579     sjmp oprim4
580
581     mov b, a
582     pop acc
583     div ab
584     jz oprim2
585     cjne a, #1, pornim3
586     mov a, b
587     cjne a, #0, pornim4
588     sjmp oprim4
589
590     mov b, a
591     pop acc
592     div ab
593     jz oprim2
594     cjne a, #1, pornim3
595     mov a, b
596     cjne a, #0, pornim4
597     sjmp oprim4
598
599     mov b, a
600     pop acc
601     div ab
602     jz oprim2
603     cjne a, #1, pornim3
604     mov a, b
605     cjne a, #0, pornim4
606     sjmp oprim4
607
608     mov b, a
609     pop acc
610     div ab
611     jz oprim2
612     cjne a, #1, pornim3
613     mov a, b
614     cjne a, #0, pornim4
615     sjmp oprim4
616
617     mov b, a
618     pop acc
619     div ab
620     jz oprim2
621     cjne a, #1, pornim3
622     mov a, b
623     cjne a, #0, pornim4
624     sjmp oprim4
625
626     mov b, a
627     pop acc
628     div ab
629     jz oprim2
630     cjne a, #1, pornim3
631     mov a, b
632     cjne a, #0, pornim4
633     sjmp oprim4
634
635     mov b, a
636     pop acc
637     div ab
638     jz oprim2
639     cjne a, #1, pornim3
640     mov a, b
641     cjne a, #0, pornim4
642     sjmp oprim4
643
644     mov b, a
645     pop acc
646     div ab
647     jz oprim2
648     cjne a, #1, pornim3
649     mov a, b
650     cjne a, #0, pornim4
651     sjmp oprim4
652
653     mov b, a
654     pop acc
655     div ab
656     jz oprim2
657     cjne a, #1, pornim3
658     mov a, b
659     cjne a, #0, pornim4
660     sjmp oprim4
661
662     mov b, a
663     pop acc
664     div ab
665     jz oprim2
666     cjne a, #1, pornim3
667     mov a, b
668     cjne a, #0, pornim4
669     sjmp oprim4
670
671     mov b, a
672     pop acc
673     div ab
674     jz oprim2
675     cjne a, #1, pornim3
676     mov a, b
677     cjne a, #0, pornim4
678     sjmp oprim4
679
680     mov b, a
681     pop acc
682     div ab
683     jz oprim2
684     cjne a, #1, pornim3
685     mov a, b
686     cjne a, #0, pornim4
687     sjmp oprim4
688
689     mov b, a
690     pop acc
691     div ab
692     jz oprim2
693     cjne a, #1, pornim3
694     mov a, b
695     cjne a, #0, pornim4
696     sjmp oprim4
697
698     mov b, a
699     pop acc
700     div ab
701     jz oprim2
702     cjne a, #1, pornim3
703     mov a, b
704     cjne a, #0, pornim4
705     sjmp oprim4
706
707     mov b, a
708     pop acc
709     div ab
710     jz oprim2
711     cjne a, #1, pornim3
712     mov a, b
713     cjne a, #0, pornim4
714     sjmp oprim4
715
716     mov b, a
717     pop acc
718     div ab
719     jz oprim2
720     cjne a, #1, pornim3
721     mov a, b
722     cjne a, #0, pornim4
723     sjmp oprim4
724
725     mov b, a
726     pop acc
727     div ab
728     jz oprim2
729     cjne a, #1, pornim3
730     mov a, b
731     cjne a, #0, pornim4
732     sjmp oprim4
733
734     mov b, a
735     pop acc
736     div ab
737     jz oprim2
738     cjne a, #1, pornim3
739     mov a, b
740     cjne a, #0, pornim4
741     sjmp oprim4
742
743     mov b, a
744     pop acc
745     div ab
746     jz oprim2
747     cjne a, #1, pornim3
748     mov a, b
749     cjne a, #0, pornim4
750     sjmp oprim4
751
752     mov b, a
753     pop acc
754     div ab
755     jz oprim2
756     cjne a, #1, pornim3
757     mov a, b
758     cjne a, #0, pornim4
759     sjmp oprim4
760
761     mov b, a
762     pop acc
763     div ab
764     jz oprim2
765     cjne a, #1, pornim3
766     mov a, b
767     cjne a, #0, pornim4
768     sjmp oprim4
769
770     mov b, a
771     pop acc
772     div ab
773     jz oprim2
774     cjne a, #1, pornim3
775     mov a, b
776     cjne a, #0, pornim4
777     sjmp oprim4
778
779     mov b, a
780     pop acc
781     div ab
782     jz oprim2
783     cjne a, #1, pornim3
784     mov a, b
785     cjne a, #0, pornim4
786     sjmp oprim4
787
788     mov b, a
789     pop acc
790     div ab
791     jz oprim2
792     cjne a, #1, pornim3
793     mov a, b
794     cjne a, #0, pornim4
795     sjmp oprim4
796
797     mov b, a
798     pop acc
799     div ab
800     jz oprim2
801     cjne a, #1, pornim3
802     mov a, b
803     cjne a, #0, pornim4
804     sjmp oprim4
805
806     mov b, a
807     pop acc
808     div ab
809     jz oprim2
810     cjne a, #1, pornim3
811     mov a, b
812     cjne a, #0, pornim4
813     sjmp oprim4
814
815     mov b, a
816     pop acc
817     div ab
818     jz oprim2
819     cjne a, #1, pornim3
820     mov a, b
821     cjne a, #0, pornim4
822     sjmp oprim4
823
824     mov b, a
825     pop acc
826     div ab
827     jz oprim2
828     cjne a, #1, pornim3
829     mov a, b
830     cjne a, #0, pornim4
831     sjmp oprim4
832
833     mov b, a
834     pop acc
835     div ab
836     jz oprim2
837     cjne a, #1, pornim3
838     mov a, b
839     cjne a, #0, pornim4
840     sjmp oprim4
841
842     mov b, a
843     pop acc
844     div ab
845     jz oprim2
846     cjne a, #1, pornim3
847     mov a, b
848     cjne a, #0, pornim4
849     sjmp oprim4
850
851     mov b, a
852     pop acc
853     div ab
854     jz oprim2
855     cjne a, #1, pornim3
856     mov a, b
857     cjne a, #0, pornim4
858     sjmp oprim4
859
860     mov b, a
861     pop acc
862     div ab
863     jz oprim2
864     cjne a, #1, pornim3
865     mov a, b
866     cjne a, #0, pornim4
867     sjmp oprim4
868
869     mov b, a
870     pop acc
871     div ab
872     jz oprim2
873     cjne a, #1, pornim3
874     mov a, b
875     cjne a, #0, pornim4
876     sjmp oprim4
877
878     mov b, a
879     pop acc
880     div ab
881     jz oprim2
882     cjne a, #1, pornim3
883     mov a, b
884     cjne a, #0, pornim4
885     sjmp oprim4
886
887     mov b, a
888     pop acc
889     div ab
890     jz oprim2
891     cjne a, #1, pornim3
892     mov a, b
893     cjne a, #0, pornim4
894     sjmp oprim4
895
896     mov b, a
897     pop acc
898     div ab
899     jz oprim2
900     cjne a, #1, pornim3
901     mov a, b
902     cjne a, #0, pornim4
903     sjmp oprim4
904
905     mov b, a
906     pop acc
907     div ab
908     jz oprim2
909     cjne a, #1, pornim3
910     mov a, b
911     cjne a, #0, pornim4
912     sjmp oprim4
913
914     mov b, a
915     pop acc
916     div ab
917     jz oprim2
918     cjne a, #1, pornim3
919     mov a, b
920     cjne a, #0, pornim4
921     sjmp oprim4
922
923     mov b, a
924     pop acc
925     div ab
926     jz oprim2
927     cjne a, #1, pornim3
928     mov a, b
929     cjne a, #0, pornim4
930     sjmp oprim4
931
932     mov b, a
933     pop acc
934     div ab
935     jz oprim2
936     cjne a, #1, pornim3
937     mov a, b
938     cjne a, #0, pornim4
939     sjmp oprim4
940
941     mov b, a
942     pop acc
943     div ab
944     jz oprim2
945     cjne a, #1, pornim3
946     mov a, b
947     cjne a, #0, pornim4
948     sjmp oprim4
949
950     mov b, a
951     pop acc
952     div ab
953     jz oprim2
954     cjne a, #1, pornim3
955     mov a, b
956     cjne a, #0, pornim4
957     sjmp oprim4
958
959     mov b, a
960     pop acc
961     div ab
962     jz oprim2
963     cjne a, #1, pornim3
964     mov a, b
965     cjne a, #0, pornim4
966     sjmp oprim4
967
968     mov b, a
969     pop acc
970     div ab
971     jz oprim2
972     cjne a, #1, pornim3
973     mov a, b
974     cjne a, #0, pornim4
975     sjmp oprim4
976
977     mov b, a
978     pop acc
979     div ab
980     jz oprim2
981     cjne a, #1, pornim3
982     mov a, b
983     cjne a, #0, pornim4
984     sjmp oprim4
985
986     mov b, a
987     pop acc
988     div ab
989     jz oprim2
990     cjne a, #1, pornim3
991     mov a, b
992     cjne a, #0, pornim4
993     sjmp oprim4
994
995     mov b, a
996     pop acc
997     div ab
998     jz oprim2
999     cjne a, #1, pornim3
1000     mov a, b
1001     cjne a, #0, pornim4
1002     sjmp oprim4
1003
1004     mov b, a
1005     pop acc
1006     div ab
1007     jz oprim2
1008     cjne a, #1, pornim3
1009     mov a, b
1010     cjne a, #0, pornim4
1011     sjmp oprim4
1012
1013     mov b, a
1014     pop acc
1015     div ab
1016     jz oprim2
1017     cjne a, #1, pornim3
1018     mov a, b
1019     cjne a, #0, pornim4
1020     sjmp oprim4
1021
1022     mov b, a
1023     pop acc
1024     div ab
1025     jz oprim2
1026     cjne a, #1, pornim3
1027     mov a, b
1028     cjne a, #0, pornim4
1029     sjmp oprim4
1030
1031     mov b, a
1032     pop acc
1033     div ab
1034     jz oprim2
1035     cjne a, #1, pornim3
1036     mov a, b
1037     cjne a, #0, pornim4
1038     sjmp oprim4
1039
1040     mov b, a
1041     pop acc
1042     div ab
1043     jz oprim2
1044     cjne a, #1, pornim3
1045     mov a, b
1046     cjne a, #0, pornim4
1047     sjmp oprim4
1048
1049     mov b, a
1050     pop acc
1051     div ab
1052     jz oprim2
1053     cjne a, #1, pornim3
1054     mov a, b
1055     cjne a, #0, pornim4
1056     sjmp oprim4
1057
1058     mov b, a
1059     pop acc
1060     div ab
1061     jz oprim2
1062     cjne a, #1, pornim3
1063     mov a, b
1064     cjne a, #0, pornim4
1065     sjmp oprim4
1066
1067     mov b, a
1068     pop acc
1069     div ab
1070     jz oprim2
1071     cjne a, #1, pornim3
1072     mov a, b
1073     cjne a, #0, pornim4
1074     sjmp oprim4
1075
1076     mov b, a
1077     pop acc
1078     div ab
1079     jz oprim2
1080     cjne a, #1, pornim3
1081     mov a, b
1082     cjne a, #0, pornim4
1083     sjmp oprim4
1084
1085     mov b, a
1086     pop acc
1087     div ab
1088     jz oprim2
1089     cjne a, #1, pornim3
1090     mov a, b
1091     cjne a, #0, pornim4
1092     sjmp oprim4
1093
1094     mov b, a
1095     pop acc
1096     div ab
1097     jz oprim2
1098     cjne a, #1, pornim3
1099     mov a, b
1100     cjne a, #0, pornim4
1101     sjmp oprim4
1102
1103     mov b, a
1104     pop acc
1105     div ab
1106     jz oprim2
1107     cjne a, #1, pornim3
1108     mov a, b
1109     cjne a, #0, pornim4
1110     sjmp oprim4
1111
1112     mov b, a
1113     pop acc
1114     div ab
1115     jz oprim2
1116     cjne a, #1, pornim3
1117     mov a, b
1118     cjne a, #0, pornim4
1119     sjmp oprim4
1120
1121     mov b, a
1122     pop acc
1123     div ab
1124     jz oprim2
1125     cjne a, #1, pornim3
1126     mov a, b
1127     cjne a, #0, pornim4
1128     sjmp oprim4
1129
1130     mov b, a
1131     pop acc
1132     div ab
1133     jz oprim2
1134     cjne a, #1, pornim3
1135     mov a, b
1136     cjne a, #0, pornim4
1137     sjmp oprim4
1138
1139     mov b, a
1140     pop acc
1141     div ab
1142     jz oprim2
1143     cjne a, #1, pornim3
1144     mov a, b
1145     cjne a, #0, pornim4
1146     sjmp oprim4
1147
1148     mov b, a
1149     pop acc
1150     div ab
1151     jz oprim2
1152     cjne a, #1, pornim3
1153     mov a, b
1154     cjne a, #0, pornim4
1155     sjmp oprim4
1156
1157     mov b, a
1158     pop acc
1159     div ab
1160     jz oprim2
1161     cjne a, #1, pornim3
1162     mov a, b
1163     cjne a, #0, pornim4
1164     sjmp oprim4
1165
1166     mov b, a
1167     pop acc
1168     div ab
1169     jz oprim2
1170     cjne a, #1, pornim3
1171     mov a, b
1172     cjne a, #0, pornim4
1173     sjmp oprim4
1174
1175     mov b, a
1176     pop acc
1177     div ab
1178     jz oprim2
1179     cjne a, #1, pornim3
1180     mov a, b
1181     cjne a, #0, pornim4
1182     sjmp oprim4
1183
1184     mov b, a
1185     pop acc
1186     div ab
1187     jz oprim2
1188     cjne a, #1, pornim3
1189     mov a, b
1190     cjne a, #0, pornim4
1191     sjmp oprim4
1192
1193     mov b, a
1194     pop acc
1195     div ab
1196     jz oprim2
1197     cjne a, #1, pornim3
1198     mov a, b
1199     cjne a, #0, pornim4
1200     sjmp oprim4
1201
1202     mov b, a
1203     pop acc
1204     div ab
1205     jz oprim2
1206     cjne a, #1, pornim3
1207     mov a, b
1208     cjne a, #0, pornim4
1209     sjmp oprim4
1210
1211     mov b, a
1212     pop acc
1213     div ab
1214     jz oprim2
1215     cjne a, #1, pornim3
1216     mov a, b
1217     cjne a, #0, pornim4
1218     sjmp oprim4
1219
1220     mov b, a
1221     pop acc
1222     div ab
1223     jz oprim2
1224     cjne a, #1, pornim3
1225     mov a, b
1226     cjne a, #0, pornim4
1227     sjmp oprim4
1228
1229     mov b, a
1230     pop acc
1231     div ab
1232     jz oprim2
1233     cjne a, #1, pornim3
1234     mov a, b
1235     cjne a, #0, pornim4
1236     sjmp oprim4
1237
1238     mov b, a
1239     pop acc
1240     div ab
1241     jz oprim2
1242     cjne a, #1, pornim3
1243     mov a, b
1244     cjne a, #0, pornim4
1245     sjmp oprim4
1246
1247     mov b, a
1248     pop acc
1249     div ab
1250     jz oprim2
1251     cjne a, #1, pornim3
1252     mov a, b
1253     cjne a, #0, pornim4
1254     sjmp oprim4
1255
1256     mov b, a
1257     pop acc
1258     div ab
1259     jz oprim2
1260     cjne a, #1, pornim3
1261     mov a, b
1262     cjne a, #0, pornim4
1263     sjmp oprim4
1264
1265     mov b, a
1266     pop acc
1267     div ab
1268     jz oprim2
1269     cjne a, #1, pornim3
1270     mov a, b
1271     cjne a, #0, pornim4
1272     sjmp oprim4
1273
1274     mov b, a
1275     pop acc
1276     div ab
1277     jz oprim2
1278     cjne a, #1, pornim3
1279     mov a, b
1280     cjne a, #0, pornim4
1281     sjmp oprim4
1282
1283     mov b, a
1284     pop acc
1285     div ab
1286     jz oprim2
1287     cjne a, #1, pornim3
1288     mov a, b
1289     cjne a, #0, pornim4
1290     sjmp oprim4
1291
1292     mov b, a
1293     pop acc
1294     div ab
1295     jz oprim2
1296     cjne a, #1, pornim3
1297     mov a, b
1298     cjne a, #0, pornim4
1299     sjmp oprim4
1300
1301     mov b, a
1302     pop acc
1303     div ab
1304     jz oprim2
1305     cjne a, #1, pornim3
1306     mov a, b
1307     cjne a, #0, pornim4
1308     sjmp oprim4
1309
1310     mov b, a
1311     pop acc
1312     div ab
1313     jz oprim2
1314     cjne a, #1, pornim3
1315     mov a, b
1316     cjne a, #0, pornim4
1317     sjmp oprim4
1318
1319     mov b, a
1320     pop acc
1321     div ab
1322     jz oprim2
1323     cjne a, #1, pornim3
1324     mov a, b
1325     cjne a, #0, pornim4
1326     sjmp oprim4
1327
1328     mov b, a
1329     pop acc
1330     div ab
1331     jz oprim2
1332     cjne a, #1, pornim3
1333     mov a, b
1334     cjne a, #0, pornim4
1335     sjmp oprim4
1336
1337     mov b, a
1338     pop acc
1339     div ab
1340     jz oprim2
1341     cjne a, #1, pornim3
1342     mov a, b
1343     cjne a, #0, pornim4
1344     sjmp oprim4
1345
1346     mov b, a
1347     pop acc
1348     div ab
1349     jz oprim2
1350     cjne a, #1, pornim3
1351     mov a, b
1352     cjne a, #0, pornim4
1353     sjmp oprim4
1354
1355     mov b, a
1356     pop acc
1357     div ab
1358     jz oprim2
1359     cjne a, #1, pornim3
1360     mov a, b
1361     cjne a, #0, pornim4
1362     sjmp oprim4
1363
1364     mov b, a
1365     pop acc
1366     div ab
1367     jz oprim2
1368     cjne a, #1, pornim3
1369     mov a, b
1370     cjne a, #0, pornim4
1371     sjmp oprim4
1372
1373     mov b, a
1374     pop acc
1375     div ab
1376     jz oprim2
1377     cjne a, #1, pornim3
1378     mov a, b
1379     cjne a, #0, pornim4
1380     sjmp oprim4
1381
1382     mov b, a
1383     pop acc
1384     div ab
1385     jz oprim2
1386     cjne a, #1, pornim3
1387     mov a, b
1388     cjne a, #0, pornim4
1389     sjmp oprim4
1390
1391     mov b, a
1392     pop acc
1393     div ab
1394     jz oprim2
1395     cjne a, #1, pornim3
1396     mov a, b
1397     cjne a, #0, pornim4
1398     sjmp oprim4
1399
1400     mov b, a
1401     pop acc
1402     div ab
1403     jz oprim2
1404     cjne a, #1, pornim3
1405     mov a, b
1406     cjne a, #0, pornim4
1407     sjmp oprim4
1408
1409     mov b, a
1410     pop acc
1411     div ab
1412     jz oprim2
1413     cjne a, #1, pornim3
1414     mov a, b
1415     cjne a, #0, pornim4
1416     sjmp oprim4
1417
1418     mov b, a
1419     pop acc
1420     div ab
1421     jz oprim2
1422     cjne a, #1, pornim3
1423     mov a, b
1424     cjne a, #0, pornim4
1425     sjmp oprim4
1426
1427     mov b, a
1428     pop acc
1429     div ab
1430     jz oprim2
1431     cjne a, #1, pornim3
1432     mov a, b
1433     cjne a, #0, pornim4
1434     sjmp oprim4
1435
1436     mov b, a
1437     pop acc
1438     div ab
1439     jz oprim2
1440     cjne a, #1, pornim3
1441     mov a, b
1442     cjne a, #0, pornim4
1443     sjmp oprim4
1444
1445     mov b, a
1446     pop acc
1447     div ab
1448     jz oprim2
1449     cjne a, #1, pornim3
1450     mov a, b
1451     cjne a, #0, pornim4
1452     sjmp oprim4
1453
1454     mov b, a
1455     pop acc
1456     div ab
1457     jz oprim2
1458     cjne a, #1, pornim3
1459     mov a, b
1460     cjne a, #0, pornim4
1461     sjmp oprim4
1462
1463     mov b, a
1464     pop acc
1465     div ab
1466     jz oprim2
1467     cjne a, #1, pornim3
1468     mov a, b
1469     cjne a, #0, pornim4
1470     sjmp oprim4
1471
1472     mov b, a
1473     pop acc
1474     div ab
1475     jz oprim2
1476     cjne a, #1, pornim3
1477     mov a, b
1478     cjne a, #0, pornim4
1479     sjmp oprim4
1480
1481     mov b, a
1482     pop acc
1483     div ab
1484     jz oprim2
1485     cjne a, #1, pornim3
1486     mov a, b
1487     cjne a, #0, pornim4
1488     sjmp oprim4
1489
1490     mov b, a
1491     pop acc
1492     div ab
1493     jz oprim2
1494     cjne a, #1, pornim3
1495     mov a, b
1496     cjne a, #0, pornim4
1497     sjmp oprim4
1498
1499     mov b, a
1500     pop acc
1501     div ab
1502     jz oprim2
1503     cjne a, #1, pornim3
1504     mov a, b
1505     cjne a, #0, pornim4
1506     sjmp oprim4
1507
1508     mov b, a
1509     pop acc
1510     div ab
1511     jz oprim2
1512     cjne a, #1, pornim3
1513     mov a, b
1514     cjne a, #0, pornim4
1515     sjmp oprim4
1516
1517     mov b, a
1518     pop acc
1519     div ab
1520     jz oprim2
1521     cjne a, #1, pornim3
1522     mov a, b
1523     cjne a, #0, pornim4
1524     sjmp oprim4
1525
1526     mov b, a
1527     pop acc
1528     div ab
1529     jz oprim2
1530     cjne a, #1, pornim3
1531     mov a, b
1532     cjne a, #0, pornim4
1533     sjmp oprim4
1534
1535    
```

9.3 Assembly Code

```

1  org 300h
2
3  str: DB "Temp is: ", 0
4  str2: DB "Temp want: ", 0
5
6  TABLE: DB "00.0","00.5", "01.0", "01.5", "02.0", "02.5", "03.0", "03.5", "04.0", "04.5", "05.0", "05.5", "06.0", "06.5", "07.0", "07.5", "08.0", "08.5", "09.0", "09.5"
7          DB "10.0","10.5", "11.0", "11.5", "12.0", "12.5", "13.0", "13.5", "14.0", "14.5", "15.0", "15.5", "16.0", "16.5", "17.0", "17.5", "18.0", "18.5", "19.0", "19.5"
8          DB "20.0","20.5", "21.0", "21.5", "22.0", "22.5", "23.0", "23.5", "24.0", "24.5", "25.0", "25.5", "26.0", "26.5", "27.0", "27.5", "28.0", "28.5", "29.0", "29.5"
9          DB "30.0","30.5", "31.0", "31.5", "32.0", "32.5", "33.0", "33.5", "34.0", "34.5", "35.0", "35.5", "36.0", "36.5", "37.0", "37.5", "38.0", "38.5", "39.0", "39.5"
10         DB "40.0","40.5", "41.0", "41.5", "42.0", "42.5", "43.0", "43.5", "44.0", "44.5", "45.0", "45.5", "46.0", "46.5", "47.0", "47.5", "48.0", "48.5", "49.0", "49.5"
11
12
13  ORG 0000H
14  LJMP main
15
16  org 0003h
17      LJMP buton1
18
19  org 0013h
20      LJMP buton2
21
22  ORG 001BH
23      cpl P3.0
24      retiS
25
26  org 0030h
27      main:
28  MOV A, #38H      // use 2 lines and 5*7
29  ACALL COMMAND
30  MOV A, #0EH      //cursor blinking off
31  ACALL COMMAND
32  MOV A, #01H      //clr screen
33  ACALL COMMAND
34
35  MOV IE,#10001101B
36  setb TCON.0
37  setb TCON.2
38  MOV r0, #44
39
40  ; 5KHZ => 200us Period => 100us delay : 1.08507 = 92
41  MOV TMOD,#20H;Timer1,mod2
42  MOV TL1, -92
43  MOV TH1, #TL1
44  SETB TR1
45
46  mov A, #0C0h
47  ACALL COMMAND
48  MOV DPTR, #str2
49
50  13 : MOV A, #00H
51  MOVC A, @A+DPTR
52  JZ done3
53  ACALL DISPLAY
54  INC DPTR

```

```

55 SJMP 13
56 done3:
57
58 MOV A, #80H // force cursor to first line
59 ACALL COMMAND
60 ;display on LCD the string
61 MOV DPTR, #str
62 11 : MOV A, #00H
63 MOVC A, @A+DPTR
64 JZ done
65 ACALL DISPLAY
66 INC DPTR
67 SJMP 11
68 done:
69
70
71 ;read and display the temperature on the LCD
72 temp:
73 ACALL LOAD
74 mov a, #10h
75 ACALL COMMAND
76 mov a, #10h
77 ACALL COMMAND
78 mov a, #10h
79 ACALL COMMAND
80 mov a, #10h
81 ACALL COMMAND
82 SJMP temp
85 ;convert the output from the ADC to ASCII for display
86 LOAD:
87 ACALL READY_CONVERSION
88 ;tens
89 MOV A, P0
90 MOV B, #33h
91 DIV AB
92 add A, #30h
93 mov r3, A; pt comparare temperaturi
94 acall display
95 ;units
96 mov A, B
97 mov B, #5
98 div AB
99 add A, #30h
100 mov r4, A; pt comparare temperaturi
101 acall display
102 ;first decimal
103 mov A, # "."
104 acall display
105 mov A, B
106 mov B, #3
107 div AB
108 cjne A, #0, mare
109 mov A, # "0"
110 mov r5, A; pt comparare temperaturi
111 acall display
112 sjmp gata
113 mare:
114 mov A, # "5"
115 mov r5, A ; pt comparare temperaturi
116 acall display
117 gata:
118
119 acall compare
120
121 RET

```

```

123 ;execute commands for LCD configuration
124 COMMAND:
125 ACALL READY_LCD
126 MOV P1, A
127 CLR P2.1
128 CLR P2.0
129 SETB P2.2
130 NOP
131 NOP
132 CLR P2.2
133 RET
134
135 ;display character on LCD
136 DISPLAY:
137 ACALL READY_LCD
138 MOV P1, A
139 SETB P2.1
140 CLR P2.0
141 SETB P2.2
142 NOP
143 NOP
144 CLR P2.2
145 RET
146
147 ;verify if the LCD displayed the character in order to display the next one
148 READY_LCD:
149 CLR P2.2
150 SETB P1.7
151 CLR P2.1 ;RS=0
152 SETB P2.0 ; R/W = 1 => READ COMMAND REG
153 BACK: CLR P2.2
154     nop
155     nop
156     SETB P2.2
157     JB P1.7, BACK
158 CLR P2.2
159 RET
160
161 ;verify if the conversion of the ADC is done
162 READY_CONVERSION:
163 SETB P2.4 ;eoc
164 NOP
165 SETB P2.3 ; start
166 NOP
167 NOP
168 CLR P2.3
169 BACK2: JNB P2.4, BACK2
170 RET

```

```

173     buton1:
174
175
176     MOV A, #0C0h
177     ACALL COMMAND
178     mov a, #11
179     cursor:
180     push acc
181     mov a, #14h
182     ACALL COMMAND
183     pop acc
184     dec a
185     jz out
186     sjmp cursor
187 out:
188
189     MOV DPTR, #TABLE
190     MOV A, r0
191     mov b, #4
192     mul AB
193     push acc
194     MOVC A, @A + DPTR
195     acall display
196     pop acc
197     inc a
198     push acc
199     MOVC A, @A + DPTR
200     acall display
201     pop acc
202     inc a
203     push acc
204     MOVC A, @A + DPTR
205     acall display
206     pop acc
207     inc a
208     push acc
209     MOVC A, @A + DPTR
210     acall display
211     pop acc
212     inc a
213     mov b, #4
214     div AB
215     mov r0, a
216
217
218     MOV A, #80H // force cursor
219     ACALL COMMAND
220     mov a, #9
221     cursor1:
222     push acc
223     mov a, #14h
224     ACALL COMMAND
225     pop acc
226     dec a
227     jz init
228     sjmp cursor1
229 init:
230
231     reti

```

```

234     buton2:
235
236     MOV A, #0C0h
237     ACALL COMMAND
238     mov a, #11
239     cursor2:
240     push acc
241     mov a, #14h
242     ACALL COMMAND
243     pop acc
244     dec a
245     jz out2
246     sjmp cursor2
247     out2:
248
249     MOV DPTR, #TABLE
250     MOV A, r0
251     dec a
252     dec a
253     mov B, #4
254     mul AB
255     push acc
256     MOVC A, @A + DPTR
257     acall display
258     pop acc
259     inc a
260     push acc
261     MOVC A, @A + DPTR
262     acall display
263     pop acc
264     inc a
265     push acc
266     MOVC A, @A + DPTR
267     acall display
268     pop acc
269     inc a
270     push acc
;
270     push acc
271     MOVC A, @A + DPTR
272     acall display
273     pop acc
274     inc a
275     mov b, #4
276     div AB
277     mov r0, a
278
279
280     MOV A, #80H // force cu
281     ACALL COMMAND
282     mov a, #9
283     cursor3:
284     push acc
285     mov a, #14h
286     ACALL COMMAND
287     pop acc
288     dec a
289     jz init2
290     sjmp cursor3
291     init2:
292     reti

```

```

294 compare:
295
296 mov DPTR, #table
297 mov a, r0
298 mov b, #4
299 mul ab
300 movc a, @a+DPTR
301 subb a, #30h
302 push acc
303 mov a, r3
304 subb a, #30h
305
306 jnz merge
307 mov a, #1
308 merge:
309
310 mov b, a ; current temperaturu
311 pop acc ; wanted temperature
312 div ab
313 jz oprim
314     cjne a, #1, pornim
315     mov a, b
316     cjne a, #0, pornim2
317     ;continuum cu ur
318     mov a, r0
319     mov b, #4
320     mul ab
321     inc a
322     movc a, @a+DPTR
323     subb a, #30h
324     push acc
325     mov a, r4
326     subb a, #30h
327
328     jnz merge2
329     mov a, #1
330 merge2:
331
332     mov b, a
333     pop acc
334     div ab
335     jz oprim2

```

```

cjne a, #1, pornim3
mov a, b
cjne a, #0, pornim4
mov a, r0
mov b, #4
mul ab
inc a
inc a
inc a
movc a, @a+DPTR
subb a, #30h
push acc
mov a, r5
subb a, #30h
jnz merge3
mov a, #1
merge3:
mov b, a
pop acc
div ab
jz oprim3
cjne a, #1, pornim5
mov a, b
cjne a, #0, pornim6
sjmp oprim4

```

```

363 pornim:
364 pornim2:
365 pornim3:
366 pornim4:
367 pornim5:
368 pornim6:
369 SETB P3.1
370
371 sjmp sarim
372
373 oprim:
374 oprim2:
375 oprim3:
376 oprim4:
377 CLR P3.1
378
379 sarim:
380 ret
381
382
383 END

```


9.4 C Code

```
1  #include <reg51.h>
2
3  sbit D7 = P1^7;
4  sbit START = P2^3;
5  sbit EOC = P2^4;
6  sbit R_W = P2^0;
7  sbit RS = P2^1;
8  sbit EN = P2^2;
9  sbit CLK = P3^0;
10 sbit CMP = P3^1;
11 sbit BUTP = P3^2;
12 sbit BUTM = P3^3;
13
14 void display_temperature();
15 void command_LCD(unsigned char x);
16 void display_LCD(unsigned char x);
17 void ready_LCD();
18 void ready_conversion() ;
19 void display_string(unsigned char *x);
20 void compare();
21
22 void timer1() interrupt 3{
23     CLK = ~ CLK;
24 }
25
26 void btn1() interrupt 0{
27 }
28
29
30 void btn0() interrupt 2{
31 }
32
33
34
```

```

35 code unsigned char str[] = {"Temp is:"};
36 code unsigned char str2[] = {"Temp want:"};
37 unsigned char r0, r3, r4, r5;
38
39 void main() {
40
41     command_LCD(0x38);
42     command_LCD(0x0E);
43     command_LCD(0x01);
44
45     IE = 0x8D;
46     TCON = 0x05;
47     r0 = 44;
48
49     TMOD = 0x20;
50     TL1 = -92;
51     TH1 = TL1;
52     TR1 = 1;
53
54     command_LCD(0xC0);
55
56     int i = 0;
57     while (str2[i] != '\0') {
58         display_LCD(str2[i]);
59         i++;
60     }
61
62     command_LCD(0x80);
63
64     char str[] = "Temperature: ";
65     i = 0;
66     while (str[i] != '\0') {
67         display_LCD(str[i]);
68         i++;
69     }

```

```
71 while (1) {
72     display_temperature();
73     command(0x10);
74     command(0x10);
75     command(0x10);
76     command(0x10);
77 }
78 }
79
80 void display_temperature(){
81     ready_conversion();
82
83     // tens
84     unsigned char A, B;
85     A = P0;
86     B = 0x33;
87     A = A / B;
88     A = A + 0x30;
89     r3 = A;
90     display(r3);
91
92     // units
93     A = B;
94     B = 0x05;
95     A = A / B;
96     A = A + 0x30;
97     r4 = A;
98     display(r4);
99
100    // first decimal
101    A = '.';
102    display(A);
103    A = B;
104    B = 0x03;
105    A = A / B;
```

```

106  if (A == 0) {
107      A = '0';
108  } else {
109      A = '5';
110  }
111  r5 = A;
112  display(r5);
113
114  compare();
115  }
116
117  void command_LCD(unsigned char x) {
118      ready_LCD();
119      P1 = x;
120      RS = 0;
121      R_W = 0;
122      EN = 1;
123      {}{}
124      EN = 0;
125  }
126
127  void display_LCD(unsigned char x) {
128      ready_LCD();
129      P1 = x;
130      RS = 1;
131      R_W = 0;
132      EN = 1;
133      {}{}
134      EN = 0;
135  }
136

```

```

137 void ready_LCD() {
138     EN = 0;
139     D7 = 1;
140     RS = 0;
141     R_W = 1;
142     while(D7==0) {
143         EN = 0;
144         {}{}
145         EN = 1;
146         EN = 0;
147     }
148 }
149 void ready_conversion() {
150     EOC = 1;
151     {}
152     START = 1;
153     {}{}
154     START = 0;
155     while(EOC!=0) {}
156 }
157 }
158 void display_string(unsigned char *x) {
159     while(*x!=0) {
160         display_LCD(*x);
161         x++;
162     }
163 }
164 }
165 void compare() {
166
167 }

```

10 References:

<https://www.geeksforgeeks.org/temperature-sensor-types-of-sensor/>

<https://www.digikey.com/en/blog/types-of-temperature-sensors>

<https://atlas-scientific.com/blog/humidity-sensor-types/>

<https://ro.mouser.com/>

<https://www.digikey.ro/>

<https://ro.farnell.com/>