Technical University

Of Cluj-Napoca

Faculty of Electronics,

Telecommunications and Technology

of Information

Microcontrollers Project

Room Thermostat

Student: Joldeș George

Group: 2031

Coordinator: Radu Etz

Table of content

9. Bibliography

1. Choosing the temperature sensor

1.1 Project theme

My project is a system which is going to adjust the temperature in concordance with the user preferences. On a screen, the respective value will be displayed, as well as the intensity of the light in the room in which the device is being placed.

The application is called room thermostat, and it is going to adjust the temperature in a house. For this project we don’t need a large temperature range.

The working principle is quite simple. If the temperature drops under a certain value, a heater will be turn on. In the other situation, if the temperature is too high, a fan will be turn on.

Here is a generic list with the necessary components:

- Temperature sensor
- Light sensor
- Amplifier
- ADC
- Microcontroller
- Fan
- Heater
- LCD
- Keyboard

1.2 The block diagram of the project

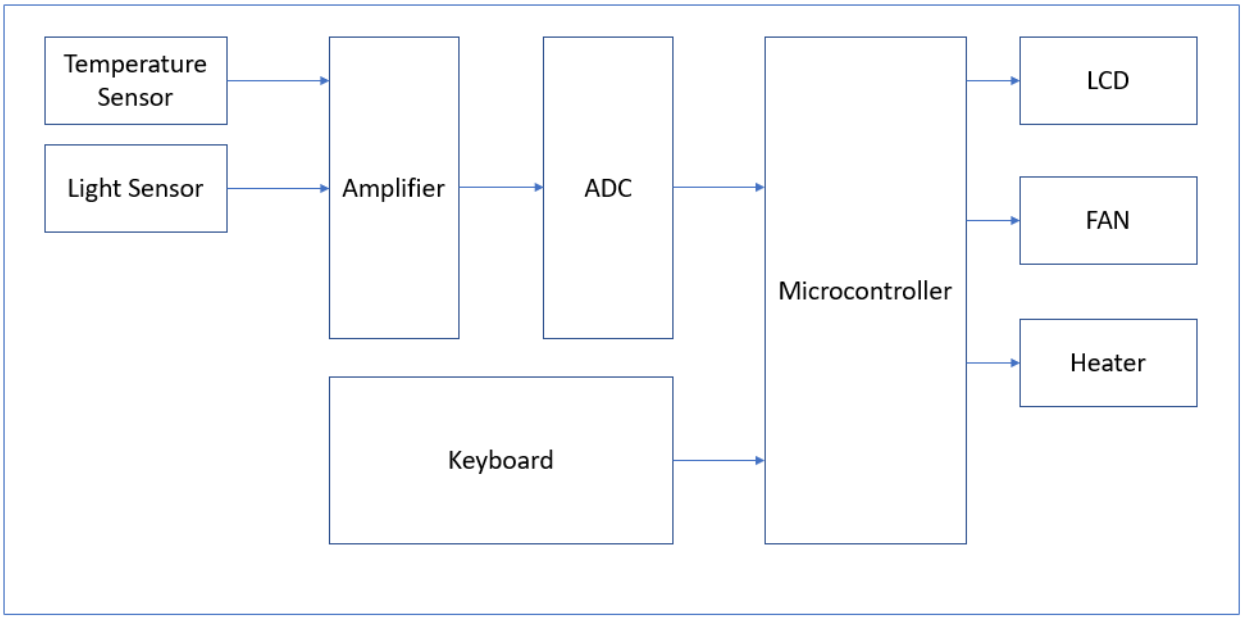

FIG. 1: THE BLOCK DIAGRAM

1.3 Ways of measuring the temperature

Thermocouple

A thermocouple is an electrical device consisting of two dissimilar electrical conductors forming an electrical junction. A thermocouple produces a temperature-dependent voltage, and this voltage can be interpreted to measure temperature.[1]

Commercial thermocouples are inexpensive, interchangeable, are supplied with standard connectors, and can measure a wide range of temperatures. In contrast to most other methods of temperature measurement, thermocouples are self-powered and require no external form of excitation. The main limitation with thermocouples is precision, system errors of less than one degree Celsius (°C) can be difficult to achieve.[1]

Thermistor

A thermistor is a type of resistor whose resistance is strongly dependent on temperature, more so than in standard resistors. There are two types of thermistors:

- **NTC** thermistors: resistance decreases as temperature rises usually due to an increase in conduction electrons bumped up by thermal agitation from valency band[1]

- **PTC** thermistors, resistance increases as temperature rises usually due to increased thermal lattice agitations particularly those of impurities and imperfections[1]

Resistance thermometer

Resistance thermometers, also called resistance temperature detectors (**RTDs**), are sensors used to measure temperature. Many RTD elements consist of a length of fine wire wrapped around a ceramic or glass core but other constructions are also used. The RTD wire is a pure material, typically platinum, nickel, or copper. The material has an accurate resistance/temperature relationship which is used to provide an indication of temperature. As RTD elements are fragile, they are often housed in protective probes.[1]

IC temperature sensor

Semiconductor temperature sensors are the devices which come in the form of integrated circuits (popularly known as IC temperature sensors). Their design results from the fact that semiconductor diodes have temperature-sensitive voltage vs. current characteristics. When two identical transistors are operated at a

constant ratio of collector current densities, the difference in base-emitter voltages is directly proportional to the absolute temperature.

1.4 Choosing a temperature measuring method

	NTC	**Thermocouples**	**RTD**	**ICs**
Temperature range	-50˚C to 250˚C (dependent on type)	-200˚C to 1250˚C (dependent on type)	-196˚C to 850˚C	-40˚C to 120˚C
Stability	-Epoxy coated: 0.2˚C/year -Hermetically sealed: 0.02˚C/year	>1˚C/year	0.1˚C-0.5˚C/year (for rated operations, used under normal conditions it can be less)	Good (for lack of numerical information)
Response time	0.12-10s (depending on size and packaging)	0.2-10s (depending on size and packaging)	1-7s	Fast (for lack of numerical information)
Price	Low to moderate	Expensive	Expensive	Low to moderate
Accuracy	Good	Errors of under 1˚C can	Highly accurate	Good/high

		be difficult to achieve		
Linearity	Low	Good	Very good	Very good

TABLE 1: BRIEF COMPARISON OF SOME TEMPERATURE MEASURING METHODS

Now I am going to choose a measuring method taking into account the advantages that they offer and the application that I want to develop.

When taking about the temperature range, I consider that all the methods offer satisfying parameters. Off course, a bigger range will be the best so, here is the thermocouple that wins.

Stability is an important for long term applications as is the one that I want to implement. So, a hermetically sealed NPC seems the best choice here.

The RTDs response time is quite unsatisfying so I'm going to take a look at the others methods which have quite similar values. ICs have also good response time.

When talking about the price, the thermistors and ICs lead the way, with the thermocouple being the most expensive.

Even though the RTDs are the most accurate on my list, for the application that I'm going to develop I see the accuracy of an NPC or of an IC more than enough. Thermocouple's value is quite unsatisfying, for at the most thermostats the minimum step being 0.5˚C.

Taking into account all the parameters that I discussed, I consider that for the project that I am going to make the best choice would be an IC sensor. They are also regularly used in those types of applications.

1.5 Presenting multiple types of ICs

	AD590	LM35	TMP35
Temperature range	-55 ˚C to 150 ˚C	-55 ˚C to 150 ˚C	-40 ˚C to 125 ˚C
Scale factor	1 uA/K	10 mV/ ˚C	10 mV/ ˚C
Price	80 RON	10 RON	13 RON
Accuracy	+/- 0.5 ˚C	+/- 0.5 ˚C	+/- 2 ˚C

TABLE 2: BRIEF COMPARISON OF MULTIPLE IC SENSORS

Being known that LM35 and AD590 are the most popular IC sensors, TMP35 can be considered being out of the equation because the other two are more available. The two sensors still remaining in the competition present quite similar specifications, while LM35 being at the same time cheaper. So, I consider that this is the sensor that I am going to use for the development of this project.

2. Choosing the light sensor

2.1 Types of light measurement

Photoresistors

Photoresistors are also knows as light dependent resistors. When light hits an object, the object absorbs the radiation from the light, and electrons move from the valence band of the semiconductor to the conduction band. If there are tons of electrons in the conduction band, the resistance will be incredibly low. Essentially, as the light increases, the resistance decreases. This means that if a photoresistor was in a completely dark room, its resistance would be at 100%. These devices are used usually in streetlamps or some photography equipment, such as a light meter. These light measurements sensors help photographers capture more accurate pictures because they help professionals understand how the light will impact the scene of the photo they're trying to take.[2]

Photodiodes

Photodiodes are a type of light sensor that converts light in electric current. Basically, when light hits a photodiode, an electron-hole pair is formed. Yet is vital to note that the light must have at least 1.1 electron volts for this pair to form. The electrons will have a negative charge, and the hole will have a negative charge. This creates depletion regions in a photodiode. The electron-hole pair can't stay in depleted zone, so they move towards the positive charge where the hole was first created. Photodiodes use this electron-hole pair to convert light into an electric current. Photodiodes are used in various devices, including smoke detectors and televisions.[2]

Phototransistors

Phototransistors are similar to photodiodes in that they both convert energy into an electric current. Yet, phototransistors are more accurate than photodiodes because they can adjust their settings based on the amount of light received. Phototransistors can view different intensities of light because these devices can alter the electrical current they create. Since phototransistors are relatively easy to use and because they're adjustable, there are various applications for them. Phototransistors are used in security systems and light control.[2]

Photomultiplier tubes

Photomultiplier tubes, members of the class of vacuum tubes, and more specifically vacuum phototubes, are extremely sensitive detectors of light in the ultraviolet, visible, and near-infrared ranges of the electromagnetic spectrum. These detectors multiply the current produced by incident light by as much as 100 million times, in multiple dynode stages, enabling (for example) individual photons to be detected when the incident flux of light is low.

The combination of high gain, low noise, high frequency response and large area of collection has maintained photomultipliers an essential place in low light level spectroscopy, confocal microscopy, Raman spectroscopy, fluorescence spectroscopy, nuclear and particle physics and many other domains.

2.2 Choosing a method of light measurement

	Photoresistor	**Photodiode**	**Phototransistor**	**Photomultiplier tubes**
Accuracy	Even if the light intensity is constant, the resistance can vary due to the temperature change. They are temperature sensitive too. Not suitable for high accuracy needed applications.	accurate	the most accurate	amplify the light signal but also amplifies the background electric noise
Available wavelength	500 nm (green) to 700 nm (red)	200 nm to 1100nm (out of the visible spectrum which is 400 nm to 800 nm)	up to 840 nm, and allows a small current to flow when is not exposed to light	200 nm to 900 nm
Linearity	not linear	excellent	good	good
Stability	not very stable, the dark	very good	very good	very good

	resistance decreases over time			
Physical size	small	small	small	large
Response time	relatively big	the fastest	relatively fast	fast
Cost	the cheapest	low	low	high

TABLE 3: BRIEF COMPARISON OF THE LIGHT MEASUREMENT METHODS

Now that I presented some of the methods which are used to measure light, it is time to choose the method that best fits my project.

The first parameter that I am going to discuss is accuracy. Here the phototransistors are clearly the best, but the photodiodes present satisfying values too. Photoresistors are also heat sensitive, which is not a good thing taking in account that the thermostat can be exposed to heat too. Photomultiplier tubes are not a really good choice either.

All the methods cover the visible spectrum of light (400 nm to 800 nm) except the photoresistors (but are pretty close too). The best are clearly the photodiodes, but all methods present satisfying parameters. So here, there is not necessary a better method than the other taking into account the project that I intend to do.

Photodiodes are the most linear of all. They definitely win this one. Phototransistors and photomultiplier tubes present good values too. I consider that they suit my requirements too.

Even though the photoresistors have the higher ground here, they have lost the battle so far, not being really suitable from the other

points of view. At this point the fight is between the phototransistors and photodiodes. I think that photomultiplier tubes are not really the best for my project because of their seize.

Considering the stability, which is pretty important in applications that involve long term use, both methods which remained to be consider (phototransistors and photodiodes) are quite good, with photodiodes being a little bit faster.

Considering all the parameters that I discussed about, I consider that a photodiode would be the best method for my application. It is linear, accurate, cheap and present good stability, which for a thermostat (my project), is very important.

2.3 Some sensors regarding the method that I have chosen

	ISL29001	**ISL76683**	**VEMD5510C**
Accuracy	15-bit resolution	16-bit resolution	+-3%
Range	0.3-10,000 lux	range can be selected via I^2C interface, range 4 varying from 0 to 64,000 lux	-
Response time	100 ms	100 ms	40 ns (rise time)
Interface method	I^2C	I^2C	-
Output type	digital	digital	analog
Linearity	3 counts to 15 counts per lux	up to 65 counts per lux	-
Ambient temperature range	-40˚C to 85˚C	-40˚C to 105˚C	-40˚C to 100˚C
Supply voltage	2.5V to 3.3V	2.5V to 3.3V	1.3V (forward voltage)
Supply current	less than 0.33 mA	less than 85 uA	50 mA (forward current)
Peak wavelength	550 nm	540 nm	550 nm
Price	≈17 RON	≈7 RON	≈9 RON

Table 4: Brief comparison of different light sensors

Taking into consideration that the temperature sensor that I have chosen had an analog output, the third sensor in the table can be considered out of the equation.

The other two sensors remain to be considered. Even though the second one (ISL76683) seems to have better parameters, the first one (ISL29001) I consider to be easier to find on the market. I found it on digikey with 17 RON. It also present good parameters, which are satisfying for the project that I intend to do.

2.4 About ISL29001

The ISL29001 is an integrated ambient light sensor with ADC and I2C interface. With a spectral sensitivity curve matched to that of the human eye, the ISL29001 provides 15-bit effective resolution while rejecting 50Hz and 60Hz flicker caused by artificial light sources.[3]

The ISL29001 contains two photodiodes. One of the photodiodes is sensitive to visible and infrared light (Diode 1) while the other diode (Diode 2) is used for temperature compensation (leakage current cancellation) and IR rejection. The ISL29001 also contains an on-chip integrating analog-to-digital converter (ADC) to convert photodiode currents into digital data.[3]

The ADC has three operating modes with two timing controls. In the first operating mode, the ADC only integrates Diode 1's current, and the digital output format is 16-bit unsigned-magnitude. In second operating mode, the ADC's operation is the same, except Diode 2's current is integrated. In the third operating mode, the ADC integrates Diode 2's current first, then Diode 1's current. The total integration time is doubled, and the digital output is the difference of the two photodiode currents (Diode 1's current – Diode 2's current). In this mode, the digital output format is 16-bit 2's-complement. Any of the three operating modes can be used with either of the two timing controls (either internally or externally controlled integration timing).[3]

The interface to the ADC is implemented using the standard I2C interface.[3]

3. Interfacing the analog sensor

3.1 Introduction

A conditioning circuit is a circuit that modifies the output of a sensor to match the parameters required by the data acquisition system, usually an ADC (Analog to Digital Convertor).

3.2 The conditioning circuit

The conditioning circuit that I realized has as components my analog temperature sensor (LM35), a non-inverting amplifier which has the gain 10, and an ADC0808. The circuit can be seen in the picture below:

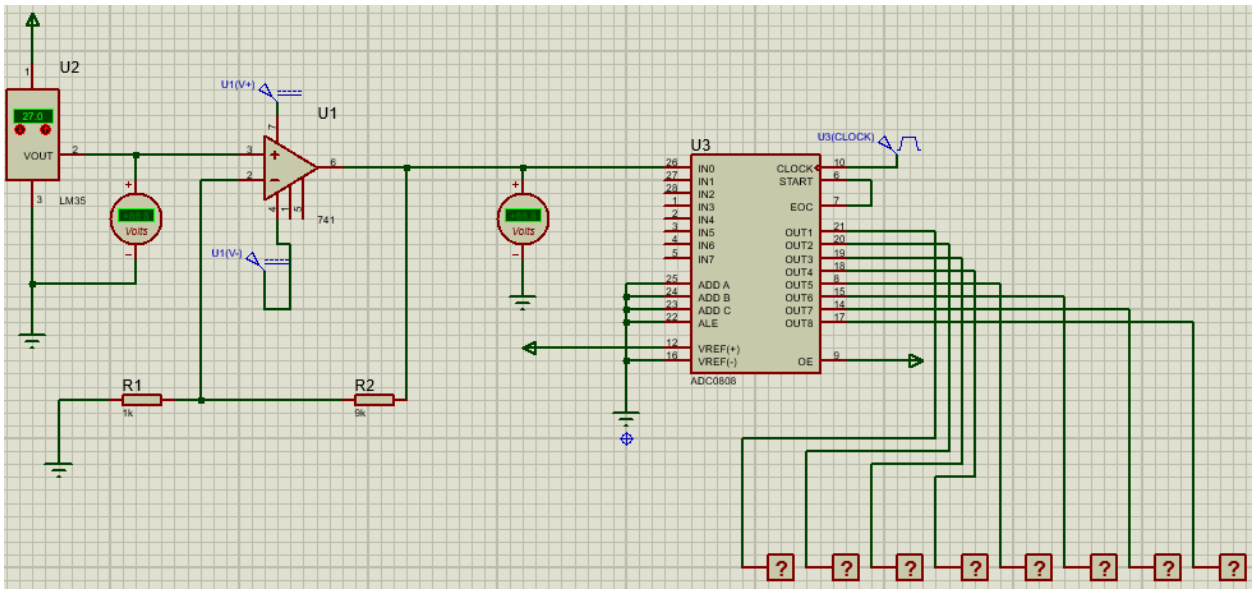

FIG. 2: THE CONDITIONING CIRCUIT

The LM35's three pins are as follows: +Vs, Vout, GND. The +Vs pin is connected at 5V, while the output of the sensor serves as input signal for the non-inverting amplifier. When the screen shot was

made, the sensor was set on 27˚C. The LM35 has a scale factor of 10mV/˚C, which means that at the output of the sensor will be 27*10=270mV. At the output of the sensor was put a DC voltmeter to measure that voltage.

The next stage of the circuit is a non-inverting amplifier. It has as input signal the output of the LM35 sensor. The gain of such an amplifier is as follows: $Av = 1 + \frac{Rf}{Rg}$. Rf represents the resistance on the feedback loop and Rg the resistance of the inverting input pin. Those resistances were sized in such a way to give a total gain of 10: Rf=9kΩ and Rg=1kΩ. Taking into consideration all that I mentioned above, at the output of the amplifier should be 2.7V. I also put a DC voltmeter at the output of the amplifier to measure that value.

In the third stage of the circuit we have the ADC. The output of the amplifier is connected to one of the analog inputs of the converter. Every digital output is connected to a logicprobe for a better legibility of the output. The clock signal is a square wave with 100 Hz frequency oscillating between 0 and 5 volts.

3.3 Making the computations

As I said earlier, the sensor measure at the time of the simulation 27˚C. This will generate an output of 270 mV, since the scale factor of the sensor is 10 mV/˚C.

$$27℃ * 10 \frac{mV}{℃} = 270\ mV$$

This value will be the input signal for the amplifier. The gain of this stage is 10. This means that at the output of the amplifier will be 2.7 V. This value is sent to the input of the converter. The equation with which the converter computes the output is the following:

$$N = \frac{Vin - Vref(-)}{Vref(+) - Vref(-)} * 256 \pm Absolute\ Accuracy$$

The reason why we multiply with 256 is because the converter is on 8 bits: 28=256.

Knowing that information, we should be able to compute the output. At the Vref(-) pin we connected the ground, which means that it's value is 0V. The result of the computation is ≈139 which in binary is 1000 1011. This is the output that we should get.

3.4 Simulating the circuit

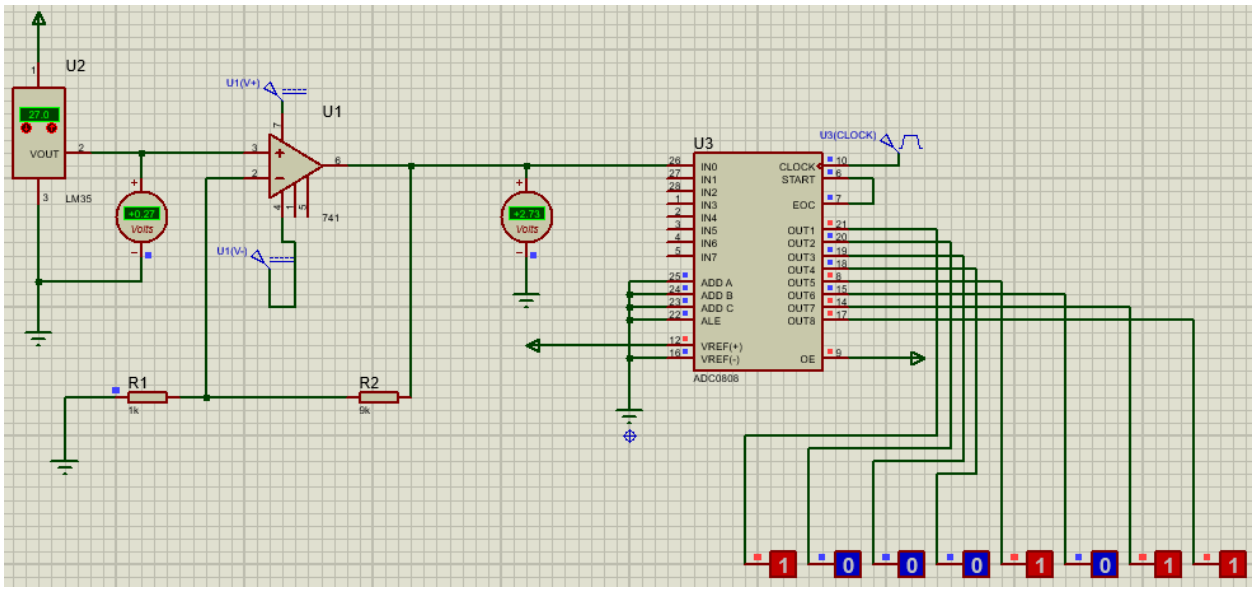

FIG. 3: RUNNING THE SIMULATION

It can be seen that the simulation went as expected. As wanted, I made a 10-gain amplifier. In the next photo I will close up the voltmeter that is found after the amplifier to show that fact that the output of the amplifier is 2.7 V.

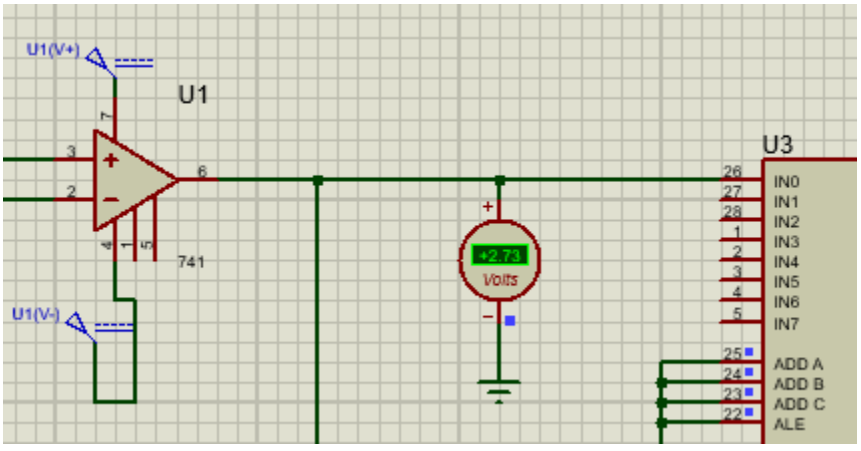

FIG. 4: FOCUSING ON THE AMPLIFIER OUTPUT

It can be seen that the value is approximatively the wanted one. Also, from the first picture of this chapter can be observed that the digital signal is the expected one.

4. Interfacing the digital sensor

4.1 I2C interface

I2C is a synchronous, multi-master, multi-slave, packed switched, single-ended, serial communication bus invented in 1982 by Philips Semiconductor. It is widely used for attaching lower-speed peripherals Ics to the processors and microcontrollers in short distance, intra-board communication.[4]

Data is sent either direction on the serial data line (SDA) by the master or slave. Only a master can start a data transfer and slaves respond to the master. It is possible to have multiple masters on a common bus, but only one could be active at a time. The SCL clock line is always driven by the master.[5]

Each I2C command initiated by a master device starts with a START condition and ends with a STOP condition. For both conditions, SCL has to be high. After the START condition, the bus is considered busy and can be used by another master only after a STOP condition is detected.[5]

- **START condition**: a high to low transition of SDA
- **STOP condition**: a low to high transition of SDA

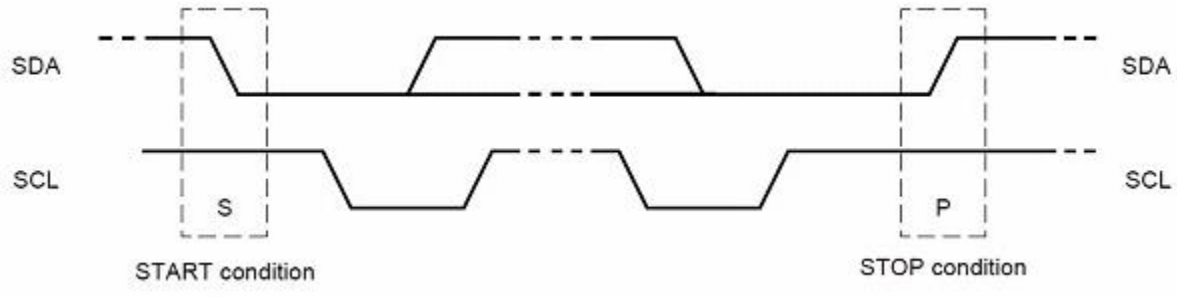

FIG. 5: EXAMPLE OF START AND STOP CONDITION

After the START condition, we need to specify the slave address. Because a single master device can send data to multiple slaves. Each slave will have a unique address. Each slave connected to the bus is software addressable by a unique 7-bit or 10-bit.[5]

The byte put on the SDA line must be 8-bit long. The data is sent on the SDA line. The most significant bit (MSB) is sent first on the SCL line produces synchronous clock. The data on the SDA line is considered valid when SCL is high. The high or low state of the data line can only be changed when the clock signal on the SCL line is LOW.[5]

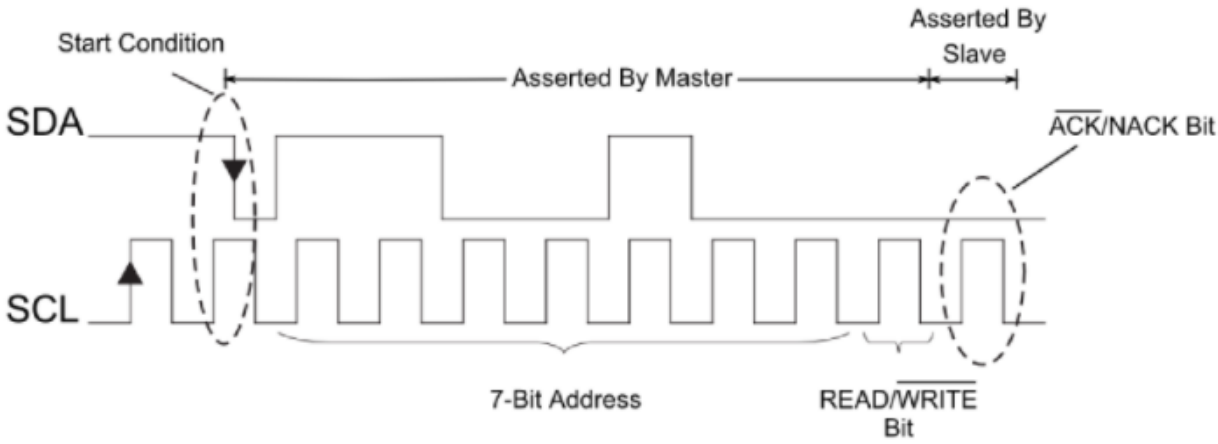

FIG. 6: START CONDITION FOLLOWED BY 7-BIT ADDRESS OF 0 × 64 AND A WRITE COMMAND SET. THE SLAVE RESPONDS WITH AN ACKNOWLEDGEMENT.

If the slave is not in a position to receive or transmit another complete byte of data it can hold the SCL line low to force the master into a wait state. Data transfer continues when the slave is ready for another byte of data and releases the clock line. To terminate the data transfer, a STOP condition is generated by the master. And if the master still wishes to communicate on the bus, it can generate another slave address along with repeated START condition without generating first STOP condition.

4.2 Program organigram

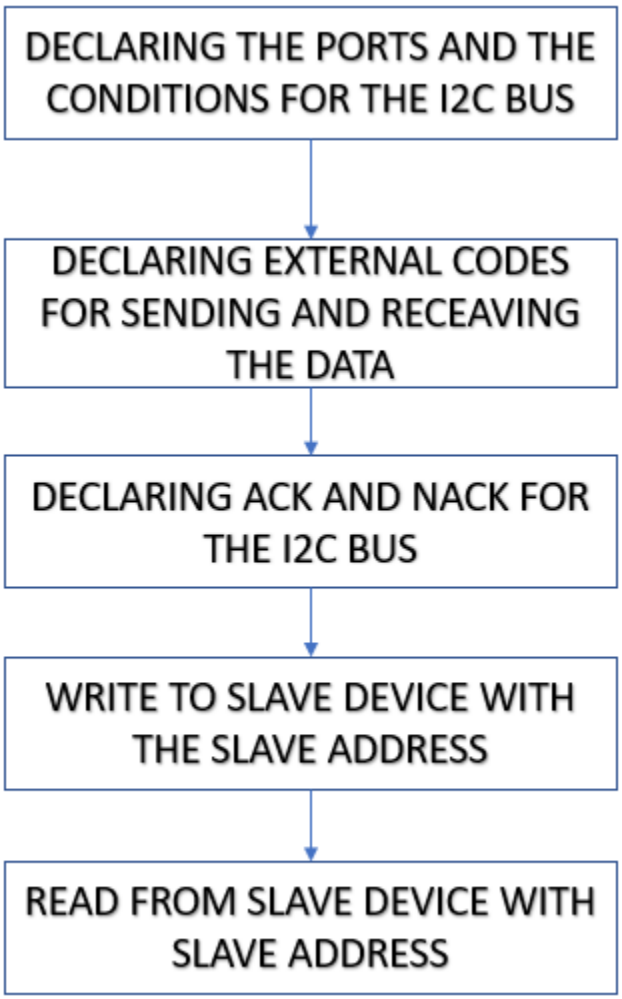

FIG. 7: THE ORGANIGRAM OF THE PROGRAM

- After sending the slave address: when the I^2C master sends the address of the slave to talk to (including the read/write bit), a slave which recognizes its address sends an **ACK**. This tells the master that the slave it is trying to reach is actually on the bus. If no slave devices recognize the address, the result is a **NACK**. In this case, the master must abort the request as there is no one to talk to.

- Within a transfer: after the side reading a byte (master on a receive or slave on a send) receives a byte, it must send a **ACK**. The major exception is if the receiver is controlling the number of bytes sent, it must send a **NACK** after the last bit to be sent. For example, on a slave-to-master transfer, the master must send a **NACK** just before sending a STOP condition to end the transfer.

4.3 Testing the interface

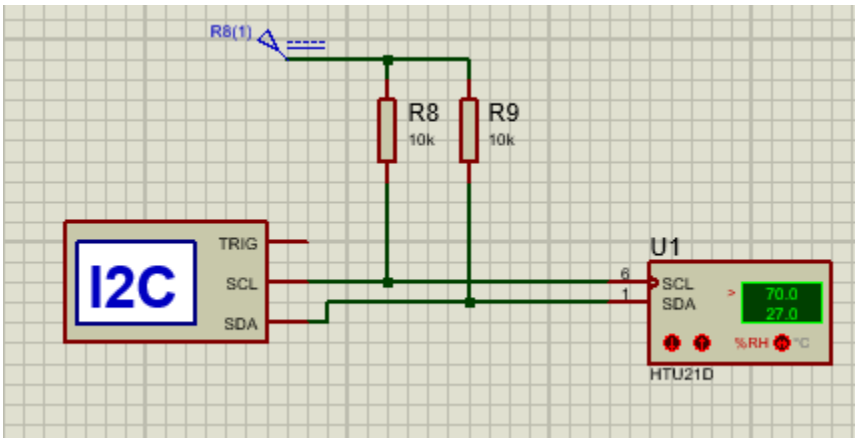

FIG. 8: THE CIRCUIT FOR INTERFACE TESTING

To test the interface, an HTU21D humidity sensor is used. The SDA and SCL buses are connected to the power supply through a pull-up resistor. This is required for the information to be transmitted.

Here I'll show the commands that are used to control the sensor:

Command	Code	Comment
Trigger Temperature Measurement	0xE3	Hold master
Trigger Humidity Measurement	0xE5	Hold master
Trigger Temperature Measurement	0xF3	No Hold master
Trigger Humidity Measurement	0xF5	No Hold master
Write user register	0xE6	
Read user register	0xE7	
Soft Reset	0xFE	

FIG. 9: THE COMMAND TABLE

Using the command sequence found in the datasheet, I succeeded to read the value of the humidity and also the value of the temperature that the sensor was measuring (the sensor is a humidity one with temperature output).

$$RH = -6 + 125 \times \frac{S_{RH}}{2^{16}}$$

FIG. 10: THE FORMULA TO COMPUTE THE OUTPUT OF THE SENSOR (FOR HUMIDITY)

Using this formula when the sensor was measuring 70% humidity, the result was 39,845.888 which in hexadecimal is approximatively 9BA5. In the picture below it can be seen that the result is very close to the expected one.

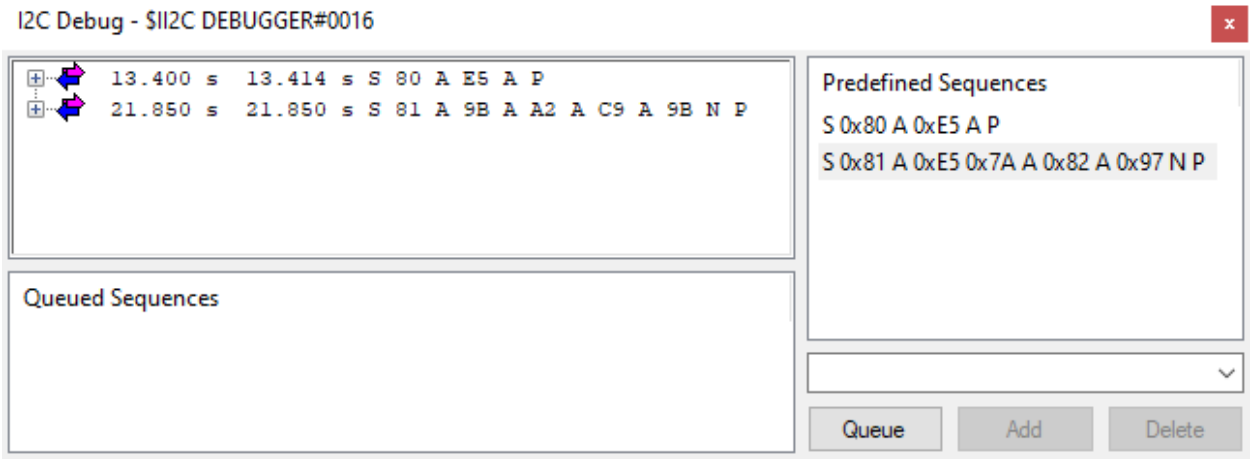

FIG. 11: THE OUTPUT OF THE SENSOR IN PROTEUS

5. Choosing the microcontroller and the peripheral

5.1 Choosing the microcontroller

A microcontroller (MCU) is a small computer on a single metal-oxide-semiconductor integrated circuit chip. A microcontroller contains one or more processor cores (CPUs) along with memory and programable input/output peripheral. Microcontrollers are designed for embedded applications, in contrast with microprocessors used in personal computers or other general-purpose applications consisting of various discrete chips.[6]

The intel MCS-51 (commonly termed 8051) is a single chip microcontroller (MCU) series developed by Intel in 1980 for use in embedded systems.[6]

Intel's original MCS-51 family was developed using NMOS technology like its predecessor Intel MCS-48, but later version, identified by a letter C in their name use CMOS technology, which is making them more power efficient.[6]

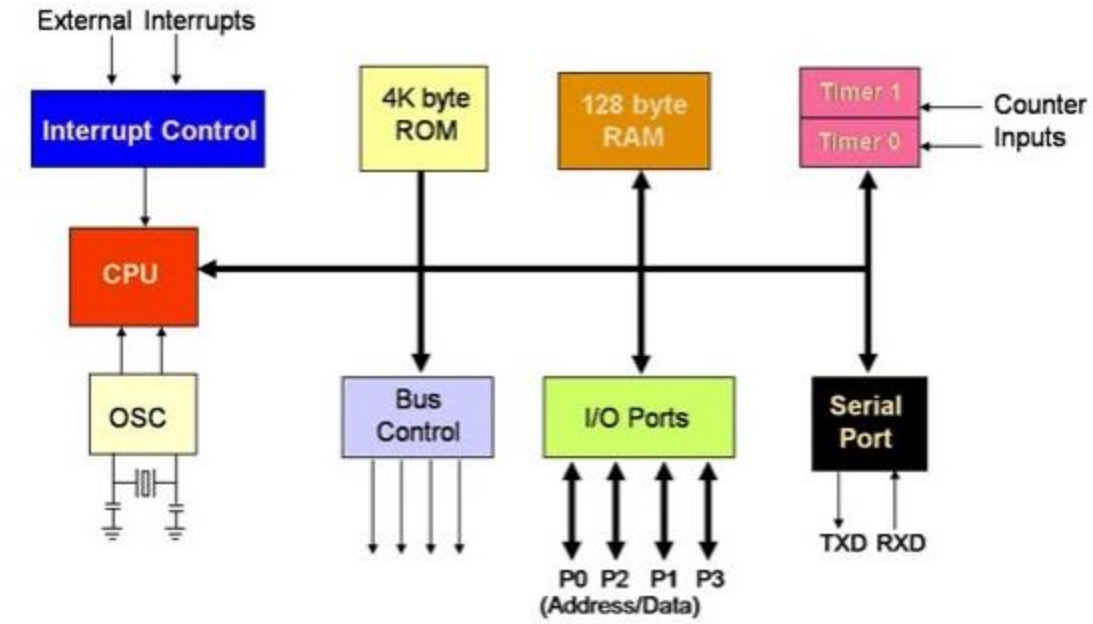

FIG. 12: INTERNAL ARCHITECTURE OF 8051

	AT89C51	P89LPC920FDH	P83CE559
Available (market)	Yes (AT89C51AC3-RDTUM)	Yes	Yes
Available (Proteus)	Yes	Yes	Yes
ROM	64 kB	2 kB	48 kB
Core size	8-bit	8-bit	8-bit
RAM	256 Bytes	256 Bytes	1526 x 8
Speed	60 MHz	Up to 18 MHz	Up to 16 MHz
Voltage supply	3V – 5.5V	2.5V – 3.6V	4.5V – 5.5V
Operating temperature	-40˚C to 85˚C	-40˚C to 85˚C	-40˚C to 85˚C
Core architecture	80C51	80C51	80C51

No. of pins	40	20	80
Price	35 RON	15 RON	45 RON

TABLE 5: MICROCONTROLLER COMPARISON

AT89C51

The AT89C51AC3 is a high-performance Flash version of the 80C51 single chip 8-bit microcontrollers.

The AT89C51AC3 provides 64K Bytes of Flash memory including In-System Programming (ISP) and IAP, 2K Bytes Boot Flash Memory, 2K Bytes EEPROM and 2048-byte ERAM.

P89LPC920FDH

The P89LPC920/921/922/9221 are single-chip microcontrollers designed for applications demanding high-integration, low-cost solutions over a wide range of performance requirements. The P89LPC920/921/922/9221 is based on a high-performance processor architecture that executes instructions in two to four clocks, six times the rate of standard 80C51 devices. Many system-level functions have been incorporated into the P89LPC920/921/922/9221 in order to reduce component count, board space, and system cost.

P83CE559

The P80CE559/P83CE559 single-chip 8-bit microcontroller is manufactured in an advanced CMOS process and is a derivative of the 80C51 microcontroller family. The P8xCE559 has the same instruction set as the 80C51.

All three microcontrollers are based on the same architecture and operate on the same temperature range (that is more than enough for my application).

All of them are available in both proteus and on the market, but I find AT89C51 being easier to be bought (easier to be found on the market).

P89LPC920FDH is clearly the cheapest of al of them, but is performance is significantly lower, having an operating frequency that can go up to just 18 MHz and a pretty small ROM memory of 2kB.

In terms of performance, I find AT89C51 being the perfect choice, at a pretty convenient price.

The first two microcontrollers have a flash memory, which is an electronic non-volatile computer memory storage medium that can be electrically erased and programmed.[7]

The last one owns a PROM memory, which is a form of digital memory where the setting of each bit is locked by a fuse or antifuse.[8]

For my project, I consider that AT89C51 is the best choice between the microcontrollers that I presented. For my simulation in Proteus, I will use the 80C51 microcontroller, which is the architecture that all the microcontrollers that I've showed are based on:

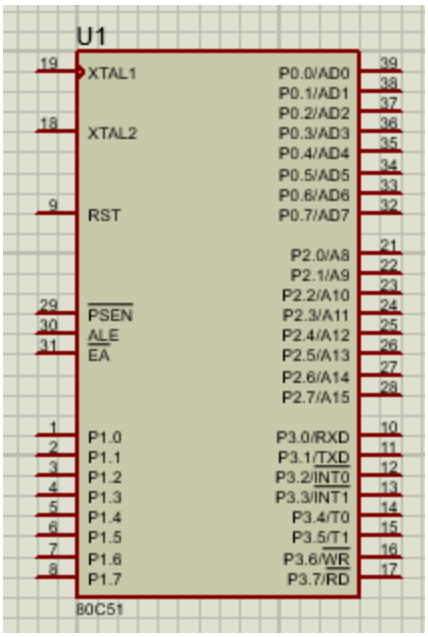

FIG. 13: 80C51 IN PROTEUS

5.2 Choosing the LCD

An LCD (Liquid-Crystal Display) 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, instead using a backlight or reflector to produce images in color or monochrome. LCDs are available to display arbitrary images of fixed images with low information content, which can be displayed or hidden, such as preset words, digits, and seven-segment displays, as in a digital clock. They use the same basic technology, except that arbitrary images are made from a matrix of small pixels, while other displays have larger elements. LCDs can either be normally on (positive) or off (negative), depending on the polarizer arrangement.[7]

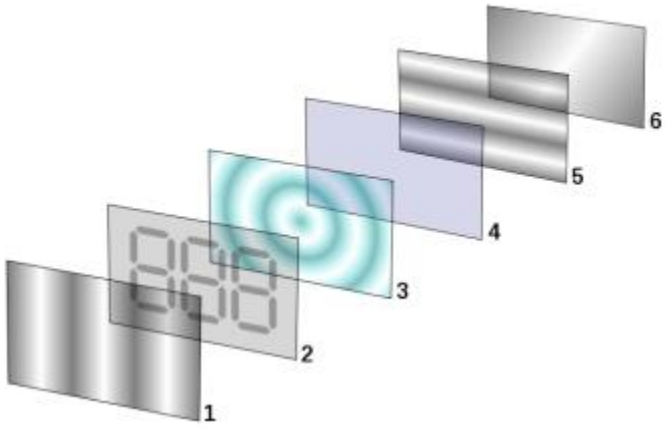

FIG. 14: REFLECTIVE TWISTED NEMATIC LIQUID CRYSTAL DISPLAY

1: Polarizing filter film with a vertical axis to polarize light as it enters

2: Glass substrate with ITO electrodes. The shapes of these electrodes will determine the shapes that will appear when the LCD is switched ON. Vertical ridges etched on the surface are smooth.

3: Twisted nematic liquid crystal.

4: Glass substrate with common electrode film (ITO) with horizontal ridges to line up with the horizontal filter.

5: Polarizing filter film with a horizontal axis to block/pass light.

6: Reflective surface to send light back to viewer.

There are two types of LCD:

- Alphanumeric
- Graphic

LCD are of different sizes, usually named after the number of rows and columns. For example, 16x2 LCD will have 2 rows and 16 columns.

All LCDs can be used in two modes of data:

- 4-bit
- 8-bit

In 4-bit mode, there is a need for two strokes to send a character (8-bit). The character is divided into nibbles. High nibble is sent first, followed by the low nibble.[10]

In 8-bit mode, the character is sent in one stroke. The 4-bit mode generates latency, but saves 4 gpio (general purpose input output) of the external controller sending data to LCD, pins that can be used elsewhere.

No.	Command	Hex value
1	Function Set: 8-bit, 1 Line, 5x7 Dots	0x30
2	Function Set: 8-bit, 2 Line, 5x7 Dots	0x38
3	Function Set: 4-bit, 1 Line, 5x7 Dots	0x20
4	Function Set: 4-bit, 2 Line, 5x7 Dots	0x28
5	Entry Mode	0x06
6	Display off Cursor off	0x08
7	Display on Cursor on	0x0E
8	Display on Cursor off	0x0C
9	Display on Cursor blinking	0x0F
10	Shift entire display left	0x18
11	Shift entire display right	0x1C
12	Move cursor left by one character	0x10
13	Move cursor right by one character	0x14

FIG. 15: PROGRAMMING LCD IN 4 BIT AND 8 BIT MODE USING 8051

Pin description:

- Pin 1 (V_{SS}): it is connected at the ground
- Pin 2 (V_{DD}): it is connected at +5V
- Pin 3 (V_{EE}): it adjusts the contrast
- Pin 4 (RS): connected to Microcontroller to shift between command/data register
- Pin 5 (RW): used to read or write data
- Pin 6 (E): connected to Microcontroller Pin and toggled between 1 and 0 for data acknowledgement
- Pin 7-14 (D0-7): the line with the index 0-7 of the data bus
- Pin 15 (A): backlight LED pin positive terminal
- Pin 16 (K): backlight LED pin negative terminal

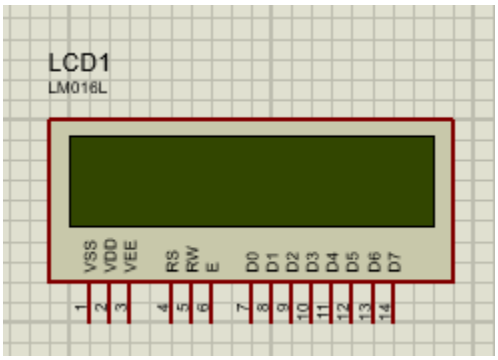

FIG. 16: LM016L IN PROTEUS

5.3 The keyboard

In order to be able to control the reference temperature value, we need a keyboard. We use two buttons to control the temperature, one to lower it and the other one to rise it. We put an additional button for reset.

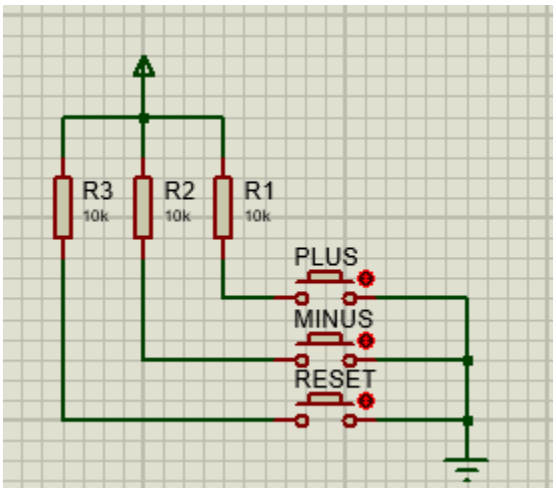

FIG. 17: THE KEYBOARD IN PROTEUS

5.4 The final scheme

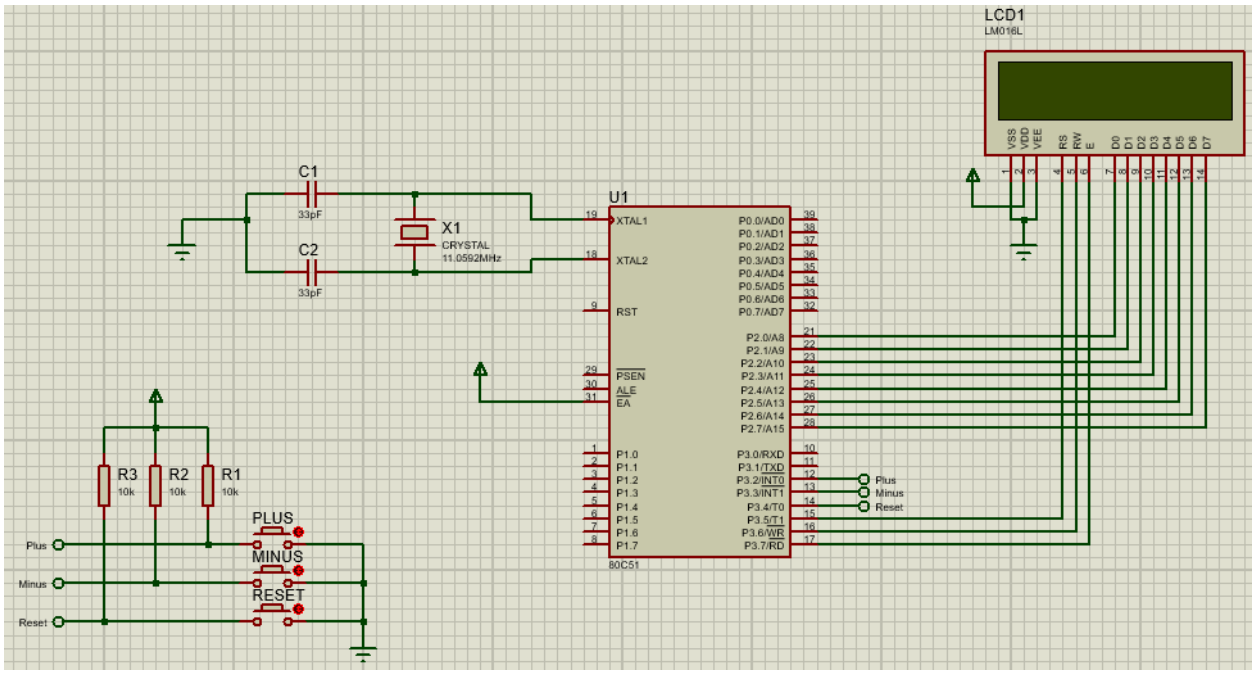

FIG. 18: THE FINAL SCHEME IN PROTEUS

5.5 The code in C

```
#include <reg51.h>
#define LCD P2

//declarare pini LCD
sbit RS = P3^5;
sbit RW = P3^6;
sbit E = P3^7;

//declarare pini tastatura
sbit PLUS = P3^2;
sbit MINUS = P3^3;
sbit RESET = P3^4;

//prototip funtctii
void delay(int);
void LCD_init(void);
void cmd(char);
void dispString(char*);
void LCD_disp(char);

//main function
void main(void)
{
    //making input
    PLUS = 1;
    MINUS = 1;
```

```
RESET = 1;

//initializing the LCD
LCD_init();

while(1)
{
	//PLUS
	if(PLUS == 0)//if button PLUS pressed
	{
		dispString("PLUS");//diplay on the screen the message
		delay(2000);
	}
	else
	{
		LCD_init();
	}
	//MINUS
	if(MINUS == 0)//if button MINUS pressed
	{
		dispString("MINUS");
		delay(2000);
	}
	else
	{
		LCD_init();
	}
```

```
            //RESET
            if(RESET == 0)//if button RESET pressed
            {
                  dispString("RESET");
                  delay(2000);
            }
            else
            {
                  LCD_init();
            }
      }
}

//delay function
void delay(int count)
{
 unsigned int i,j;
 for (i=0;i<count;i++)
 for (j=0;j<100;j++) {}//intarziere de 1ms
}

//command function
void cmd(char t)
{
      LCD = t;
      RS=0;
      RW=0;
```

```
    E=1;
    delay(5);
    E=0;
}

//initializing function
void LCD_init()
{
    cmd(0x38); //data init
    cmd(0x0C); //LCD display on and cursor off
    cmd(0x01); //clear LCD display
    cmd(0x80); //positioning cursor at the first line
}

//display character function
void dispString(char *p)
{
    while(*p)
    {
            LCD_disp(*p++);
    }
}

//display on LCD function
void LCD_disp(char x)
{
    LCD=x;
```

```
    RS=1;
    RW=0;
    E=1;
    delay(5);
    E=0;
}
```

6. The final scheme

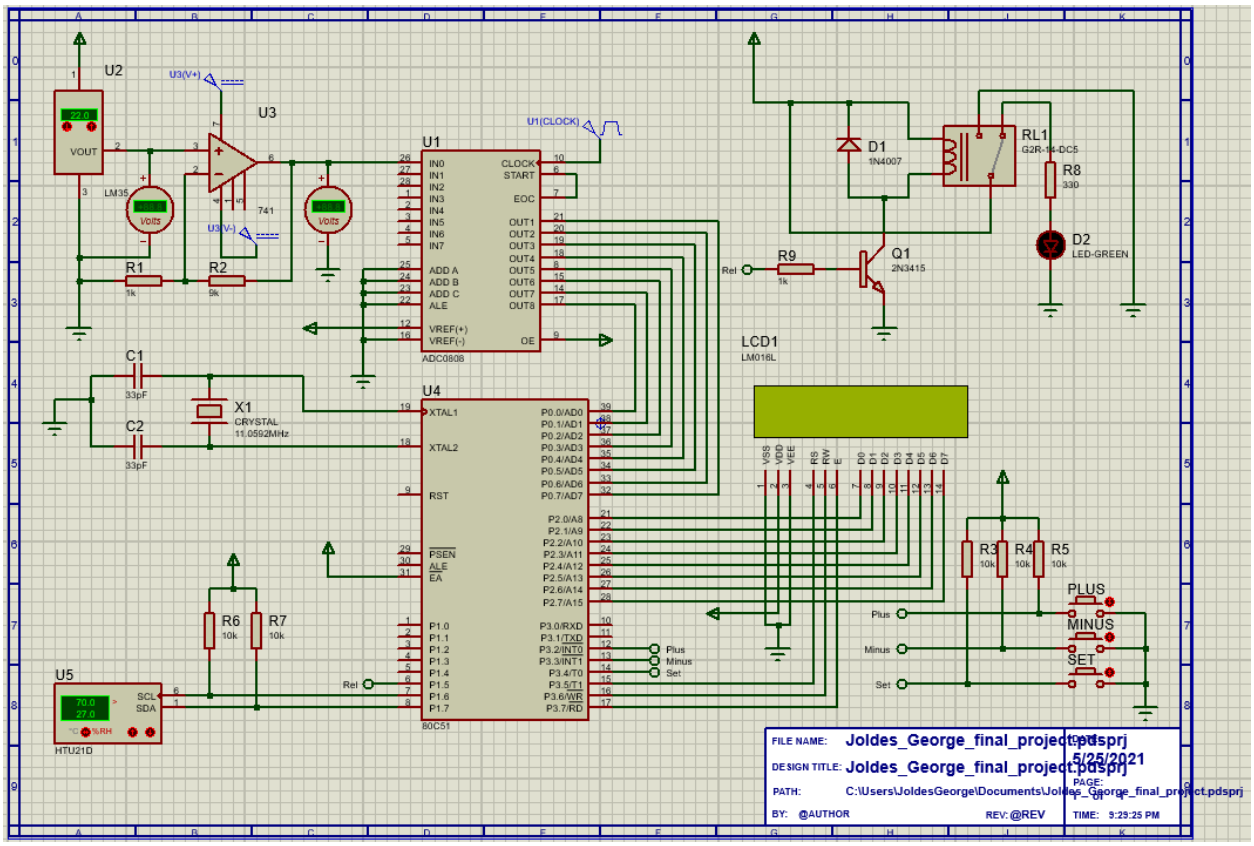

FIG. 19: THE FINAL SCHEME

7. The final code

```
org 0000h

;declare LCD pins
rs equ p3.5
rw equu p3.6
e equ p3.7

;declare buttons
plus equ p3.2
minus equ p3.3
set_btn equ p3.4
rel equ p1.5

;set the ports for the buttons as inputs ones
setb plus
setb minus
setb set_btn

;set the i2c buses
sda equ p1.7
scl equ p1.6

;initialization LCD
mov a,#38h;set the LCD on 8-bit mode, with 2 lines and every character being represented on a 5x7 matrix
acall LCD_command
```

```
acall delay
mov a,#0ch;display on and cursor off
acall LCD_command
acall delay
mov a,#01h;clear lcd display
acall LCD_command
acall delay
mov a,#80h;position the cursor at the first line of the display
acall LCD_command
acall delay

;initialization ports
mov p0,#0ffh;port 0 is set as input port
mov p1,#0ffh;port 1 is set as input port
mov p2,#00h;port 2 is set as output port

;write messages from the code memory
;display message for actual temperature
mov r0,#6;counter for dptr
mov dptr,#mesaj1
here1:
mov a,#00h
movc a,@a+dptr
mov p2,a
acall LCD_display
acall delay
inc dptr
```

```
djnz r0,here1

;display message for set temperature
mov a,#0c0h;second line of the LCD
acall LCD_command
acall delay
mov r0,#6;counter for dptr
mov dptr,#mesaj2
here2:
mov a,#00h
movc a,@a+dptr
mov p2,a
acall LCD_display
acall delay
inc dptr
djnz r0, here2

;display message for humidity
mov a,#8ch
acall LCD_command
acall delay
mov r0,#4
mov dptr,#mesaj3
here3:
mov a,#00h
movc a,@a+dptr
acall LCD_display
```

```
acall delay
inc dptr
djnz r0,here3

mov r3,#50
;routine for the set button
mov r1,#72h
set_temp:
jnb set_btn,modify
mov a,#086h
acall LCD_command
acall delay
mov a,p0

mov b,r1
cjne a,b,releu
setb rel
sjmp jump
releu:
clr rel
jump:

acall conversion
mov a,#0C6h
acall LCD_command
acall delay
mov a,r1
```

```
acall conversion

djnz r3,com
acall comunicate
acall comunicate
mov r3,#50
com:

sjmp set_temp

;sequence to modify the value
modify:
jnb plus,increment
jnb minus,decrement

mov a,#086h
acall LCD_command
acall delay
mov a,p0
acall conversion
mov a,#0c6h
acall LCD_command
acall delay
mov a,r1
acall conversion

acall comunicate
```

```
sjmp modify

increment:
inc r1
inc r1
inc r1
inc r1
inc r1
cjne r1,#71h,jump2
inc r1
jump2:
sjmp set_temp
decrement:
dec r1
dec r1
dec r1
dec r1
dec r1
cjne r1,#6dh,jump3
dec r1
jump3:
sjmp set_temp

comunicate:
mov a,#0cdh
acall LCD_command
```

```
acall delay

;initiating the comunication
acall I2C_init
acall I2C_start
mov a,#80h
acall I2C_sendByte
mov a,#0e5h
acall I2C_sendByte
acall I2C_stop

;receiving data
acall I2C_init
acall I2C_start
mov a,#81h
acall I2C_sendByte
acall I2C_readByte
acall conversion2
mov r2,a
acall I2C_ack
acall I2C_readByte
acall conversion3
add a,r2
subb a,#7
acall conversion4
acall I2C_nack
acall I2C_stop
```

```
ret

;conversion routine
conversion:
subb a,#1
mov b,#5
div ab
mov b,#10
div ab
add a,#30h
acall LCD_display
acall delay
mov a,b
add a,#30h
acall LCD_display
acall delay
mov a,#"C"
acall LCD_display
acall delay
ret

;conversion for humidity
conversion2:
mov b,#2
div ab
ret
```

```
conversion3:
mov b,#2
div ab
mov b,#10
div ab
mov b,#10
div ab
mov b,#10
div ab
ret

conversion4:
mov b,#10
div ab
add a,#30h
acall LCD_display
acall delay
mov a,b
add a,#30h
acall LCD_display
acall delay
mov a,#"%"
acall LCD_display
acall delay
ret
```

```
;LCD command routine
LCD_command:
mov p2,a
clr rs;RS=0, instruction register
clr rw;RW=0, write
setb e;E=1, start data read/write
clr e
ret

;LCD display routine
LCD_display:
mov p2,a
setb rs;RS=1, data register
clr rw;RW=0, write
setb e
acall delay
clr e
ret

I2C_init:
setb sda
setb scl
ret

I2C_start:
setb scl
clr sda
```

```
clr scl
ret

I2C_stop:
clr scl
clr sda
setb scl
setb sda
ret

I2C_ack:
clr sda
setb scl
clr scl
setb sda
ret

I2C_nack:
setb sda
setb scl
clr scl
setb scl
ret

I2C_sendByte:
mov r7,#08h
back1:
```

```
clr SCL
rlc a
mov SDA,c
setb SCL
djnz r7, back1
clr SCL
setb SDA
setb SCL
mov c, SDA
clr SCL
ret

I2C_readByte:
mov r7,#08h
back2:
clr SCL
setb SCL
mov c, SDA
rlc a
djnz r7, back2
clr SCL
setb SDA
ret

;delay function
delay:
clr tr0
```

```
orl tmod,#01
mov th0,#0eeh
mov tl0,#00h
setb tr0
again:
jnb tf0,again
clr tf0
clr tr0
ret

;messages section
mesaj1:
db 'T act:',0;message for actual temperature
mesaj2:
db 'T set:',0;message for set temperature
mesaj3:
db 'Hum:',0;message for humidity

end
```

8. Simulation in Proteus

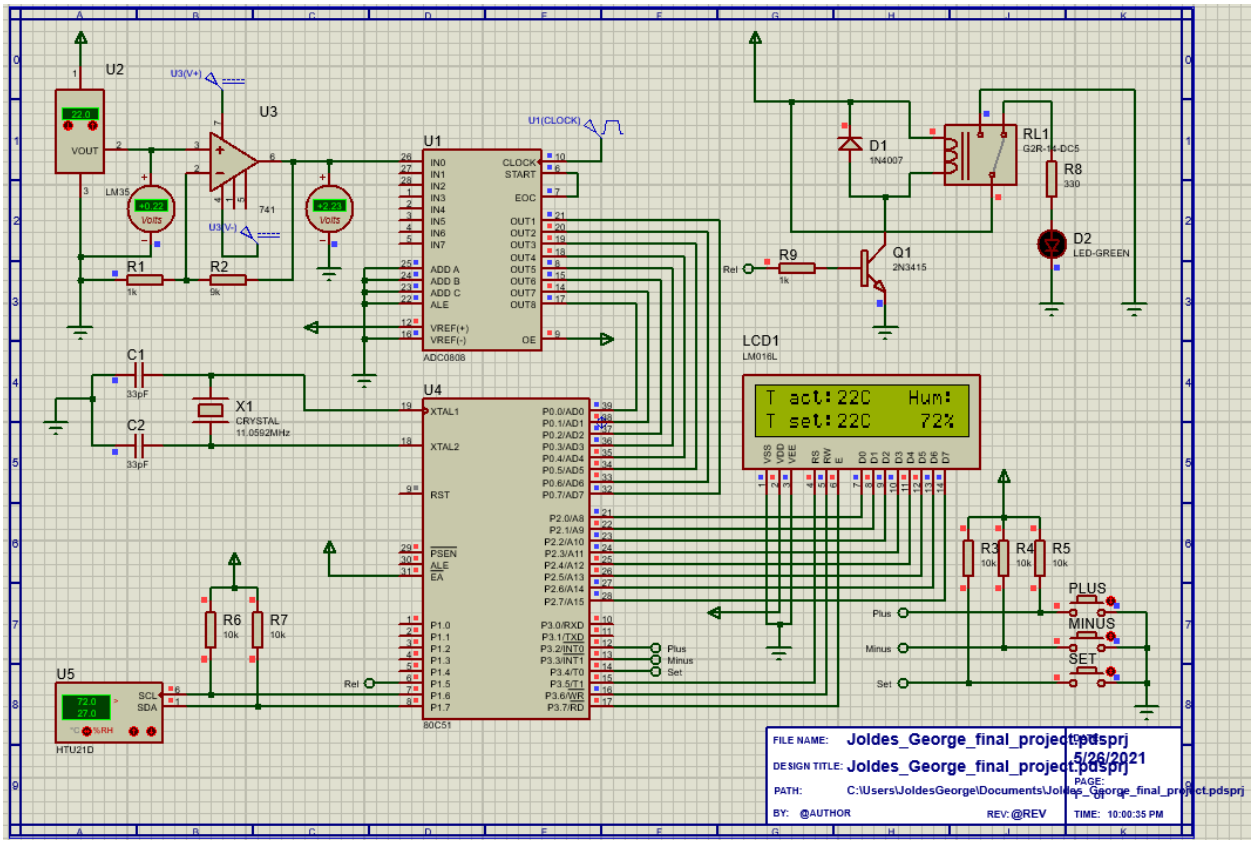

FIG. 20: THE SIMULATION

In the simulation, if the temperature of the room doesn't correspond with the temperature desired by the user, the relay will turn on. The relay would represent the executive element in the real prototype.

To change the desired temperature, the user has to press the SET button and then choose is he wants to rise the temperature or to lower it.

9. Bibliography

[1] https://www.electronics-tutorials.ws/io/io_3.html

[2] https://www.seeedstudio.com/blog/2020/01/08/what-is-a-light-sensor-types-uses-arduino-guide/

[3] https://www.mouser.com/datasheet/2/465/fn6166-23722.pdf

[4] https://en.wikipedia.org/wiki/I%C2%B2C

[5] https://www.circuitbasics.com/basics-of-the-i2c-communication-protocol/

[6] https://en.wikipedia.org/wiki/Intel_8051

[7] https://en.wikipedia.org/wiki/Liquid-crystal_display