Politecnico di Torino

Testing and Certification

Second Laboratory Session

Implementation of a Digital Thermometer using an Arduino UNO Board and LM335 Sensor

Authors:
D'Addato Mauro
Fini Simone
Marmo Sebastiano
Triarico Antonio

Professor:
Carullo Alessio

13 November 2017

POLITECNICO
DI TORINO

1 Introduction

This second laboratory session aims (as indicated by the title) to assemble and program a digital thermometer based on a LM335 sensor. In particular, the instruments used are:

- Arduino UNO Board;
- the LM335 temperature electronic sensor;
- two 2.2kΩ resistors;
- 1kΩ resistor;
- HP 34401A Multimeter (to measure the correct value of the components);
- Greisinger GFTH 95, a Digital Hygro-Thermometer.

Accuracy Specifications ± (% of reading + % of range) [1]

Function	Range [3]	Test Current or Burden Voltage	24 Hour [2] 23°C ± 1°C	90 Day 23°C ± 5°C	1 Year 23°C ± 5°C	Temperature Coefficient /°C 0°C – 18°C 28°C – 55°C
DC Voltage	100.0000 mV		0.0030 + 0.0030	0.0040 + 0.0035	0.0050 + 0.0035	0.0005 + 0.0005
	1.000000 V		0.0020 + 0.0006	0.0030 + 0.0007	0.0040 + 0.0007	0.0005 + 0.0001
	10.00000 V		0.0015 + 0.0004	0.0020 + 0.0005	0.0035 + 0.0005	0.0005 + 0.0001
	100.0000 V		0.0020 + 0.0006	0.0035 + 0.0006	0.0045 + 0.0006	0.0005 + 0.0001
	1000.000 V		0.0020 + 0.0006	0.0035 + 0.0010	0.0045 + 0.0010	0.0005 + 0.0001
Resistance [4]	100.0000 Ω	1 mA	0.0030 + 0.0030	0.008 + 0.004	0.010 + 0.004	0.0006 + 0.0005
	1.000000 kΩ	1 mA	0.0020 + 0.0005	0.008 + 0.001	0.010 + 0.001	0.0006 + 0.0001
	10.00000 kΩ	100 µA	0.0020 + 0.0005	0.008 + 0.001	0.010 + 0.001	0.0006 + 0.0001
	100.0000 kΩ	10 µA	0.0020 + 0.0005	0.008 + 0.001	0.010 + 0.001	0.0006 + 0.0001
	1.000000 MΩ	5 µA	0.002 + 0.001	0.008 + 0.001	0.010 + 0.001	0.0010 + 0.0002
	10.00000 MΩ	500 nA	0.015 + 0.001	0.020 + 0.001	0.040 + 0.001	0.0030 + 0.0004
	100.0000 MΩ	500 nA // 10 MΩ	0.300 + 0.010	0.800 + 0.010	0.800 + 0.010	0.1500 + 0.0002

Fig. 1: Specifications provided by the DMM datasheet

It is important to highlight that the accuracy considered to compute the error which affected the reading of the real R value is $\pm(0.010+0.001)$, expressed as $\pm(\%of reading+\%of range)$, while the accuracy for the real V value is $\pm(0.0035+0.0005)$.
This laboratory report will present the two different ways used in order to develop the aformentioned thermometer. Indeed, Arduino gives the possibility to choose between several voltage references for the internal analog-to-digital converter (in particular, 5V from the USB, 1.1V and external source). As it will be seen, they have their benefits and drawbacks in terms of usability and uncertainty, which must be evaluated so as to choose the best solution for the specific task.

2 First Implementation: 5V Reference

2.1 Hardware Design

The first implementation was made using the 5V provided by the USB connection as reference. This means that, since the number of bits of the internal A/D converter is 10 (for sampling frequencies less than 15 kSa/s), the quantization voltage is

$$V_q = \frac{V_{ref}}{2^{Nb}} = \frac{5}{2^{10}} \approx 0.0049V \tag{1}$$

But as well as the Arduino components, sensor specifications must be checked and respected; in fact, by looking at the LM335 datasheet, it is clear that the sensor behaves as a common Zener diode and, therefore, a conditioning circuit must be implemented, since the recommended forward current is included between 0.4mA and 5mA.

6.2 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

			MIN	NOM	MAX	UNIT
Specified Temperature	LM135, LM135A	Continuous ($T_{MIN} \le T_A \le T_{MAX}$)	−55		150	°C
		Intermittent (1)	150		200	
	LM235, LM235A	Continuous ($T_{MIN} \le T_A \le T_{MAX}$)	−40		125	°C
		Intermittent (1)	125		150	
	LM335, LM335A	Continuous ($T_{MIN} \le T_A \le T_{MAX}$)	−40		100	°C
		Intermittent (1)	100		125	
Forward Current			0.4	1	5	mA

Fig. 2: Part of the original LM335 datasheet

For this reason, a single resistor R is required, and it must satisfy the following conditions:

$$I_{min} = \frac{V_{ref} - V_{outmax}}{R} > 0.4mA \Longrightarrow R < \frac{V_{ref} - V_{outmax}}{0.4mA} \tag{2}$$

$$I_{max} = \frac{V_{ref} - V_{outmin}}{R} < 5mA \Longrightarrow R > \frac{V_{ref} - V_{outmax}}{5mA} \tag{3}$$

where V_{outmax} and V_{outmin} are the maximum and minimum (respectively) voltages expected to be present across the sensor (as illustrated by the following figure, taken by the official sensor datasheet).

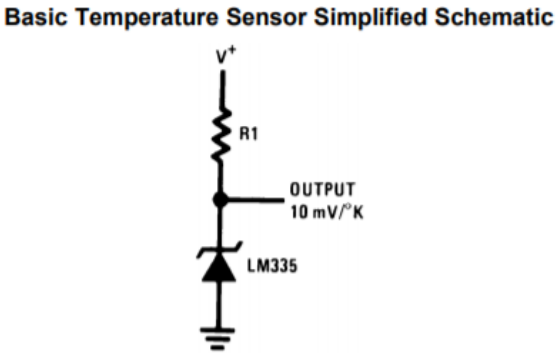

Fig. 3: Implementation of the circuit

It is also important to remember that V_{out} is referred to the absolute zero and the sensibility of the LM335 is equal to:

$$S = 10mV/K. \tag{4}$$

This means that every increase of a single Kelvin degree, V_o goes up of 10mV starting from a voltage equals to 0V (at a temperature of -273.15 °C, which will be considered as -273 °C to simplify the computations). Moreover, looking at the figure 1 again, it can be seen that the working temperature of the LM335 is included between -40 °C and 100 °C, but specifically for this experiment the range was narrowed to 5 °C minimum and 75 °C maximum.

	Temperature (C)	Temperature (K)	Voltage (V)
Minimum Value	5	278	2.78
Maximum Value	75	348	3.48

Table 1: Minimum and maximum values of the temperature and their respective voltage values

Keeping this in mind, it ended up that the value of the resistor must have been included within 444Ω and $3.8k\Omega$; so it was decided to use a $2.2k\Omega$ resistance, and the circuit was implemented, as it is shown in figure 3.

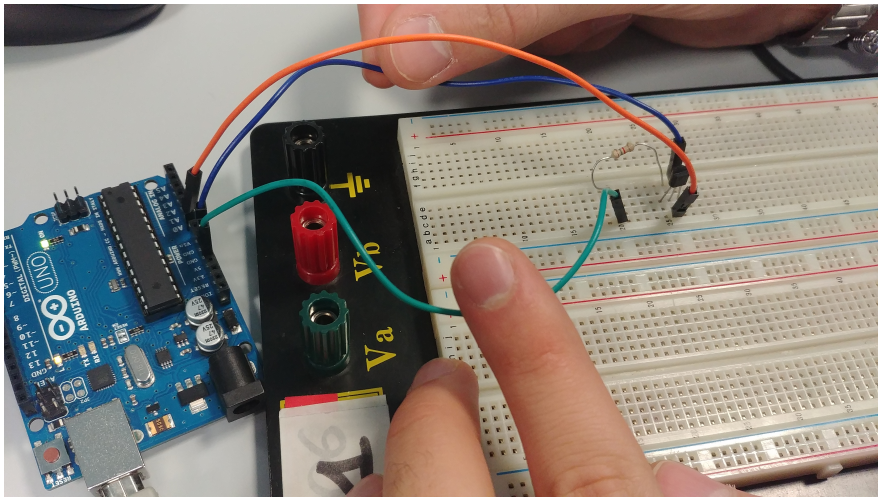

Fig. 4: Implementation of the circuit on the breadboard

2.2 Firmware Design

Having implemented the hardware, the following step was to write pieces of code so as to make the thermometer work.

```
const double V_ref=4.95560;
const double V_int=1.09801;
const double delta_D_5= 2;
const double delta_D_11= 2;
const double delta_V_ref= 0.25;
const double delta_V_int= 0.1;
const double delta_T_sensor= 0;
double input;
double temp;
double error;

void setup() {
  pinMode(A0,INPUT);
  Serial.begin(9600);
  //analogReference(INTERNAL);
}

void loop() {
  input=analogRead(A0);
  Serial.print("Input:");
  Serial.println(input);                    //print output of the ADC
  temp= 100*input*V_ref/1024;               //calibration function
  error= (V_ref/10.24)*delta_D_5 + (input/10.24)*delta_V_ref + delta_T_sensor;
  //temp= (319.753*input*V_int)/1024;
  //error= (V_int/(10.24*3.19753)*delta_D_11 + (input/(10.24*3.19753))*delta_V_int + delta_T_sensor;
  Serial.println(temp);
  Serial.println(error);
  delay(1000);
}
```

Fig. 5: Code implemented on the Arduino board

First of all, two constants (*V_ref* and *V_int*) are initialized, which correspond to the values of the voltage references taken into account. In addition, the variables for the value of V_o, the computed temperature *temp* and its error are declared. The other constants will be explained in chapter 2.3.
Inside the setup the analog pin *A0* (which is the one connected to the orange wire) is set as an INPUT and, by using the native Arduino class *Serial* and its method *begin(int baudrate)*, the USB connection to the PC is initialized and the baud rate is set to 9600 (i.e. 9600 bits/s).
Then, the loop starts. At the beginning of every cycle, by calling the function *analogRead(pin)*, the analog input is read and, therefore, converted by the ADC; subsequently, the converted value is printed.
Finally, the correct temperature is computed in the way explained below.
Data provided by the ADC (D_{out}) is equal to:

$$D_{out} = \frac{V_{out}}{V_q}, \tag{5}$$

and, remembering the correlation shown by the formula (1), it can be written

$$D_{out} = \frac{V_{out} * 2^{Nb}}{V_{ref}}. \tag{6}$$

V_{out} depends both on the sensitivity S of the sensor and the temperature T, so

$$D_{out} = \frac{S * T * 2^{Nb}}{V_{ref}}. \tag{7}$$

At this point, it is easy to explicit the temperature and calculate it, using the formula:

$$T = \frac{D_{out} * V_{ref}}{S * 2^{Nb}}, \tag{8}$$

that is used in the code.
It is important to notice that the commented lines are used to compute the temperature when the ADC reference is set to the internal 1.1V, as it will be described in the paragraph 3.2.

2.3 Computation of the Uncertainty

After the implementation of the circuit and the writing of the code, the uncertainty which affected the measure had to be computed.

$$\delta T = |\frac{\partial T}{\partial D_{out}}| * \delta D_{out} + |\frac{\partial T}{\partial V_{ref}}| * \delta V_{ref} + |\frac{\partial T}{\partial S}| * \delta S, \tag{9}$$

and, calculating the derivatives,

$$\delta T = \frac{V_{ref}}{S * 2^{Nb}} * \delta D_{out} + \frac{D_{out}}{S * 2^{Nb}} * \delta V_{ref} + \delta T_{sensor} \tag{10}$$

At this point, the unexplained constants at the beginning of the code must be described; they refer to the uncertainty of the components which are used to implement the hardware part, and their values correspond to the ones given by the datasheets of each single device:

- *delta_D_5* = the absolute uncertainty of the ADC with 5V reference. Since it is equal to $\pm 2LSB$ and, since the dimensional analysis must be correct, its value is 2;
- *delta_D_11* = the absolute uncertainty of the ADC with the internal 1.1V reference. In the same way as before, its value is 2 (two variables were declared just to make the code more clear).
- *delta_V_ref* = the uncertainty of the voltage provided by the USB connection. It is equal to $\pm 0.25V$.
- *delta_V_int* = the uncertainty of the 1.1V voltage reference. It is equal to $\pm 0.1V$.
- *delta_T_sensor* = the uncertainty given by the sensor. By looking at the figure 5, it is clear that it is not constant, but depends on the temperature measured. Since in this case the temperature range was included within 5°C and 75°C, it was decided to approximate the uncertainty value to 0.

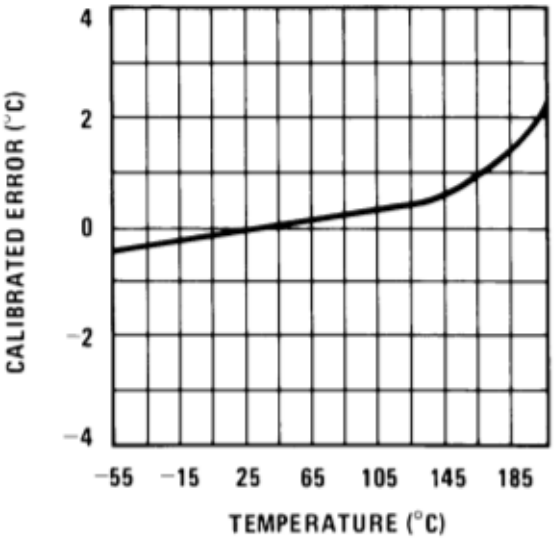

Fig. 6: Error function given by the original LM335 datasheet

Finally, measuring the room temperature, data provided by the device is shown in the following table.

ADC Output	Temperature (K)	Uncertainty (K)
606.0	293.27	15.76

Table 2: Results provided by the thermometer

According to the formula

$$|x_1 - x_2| \stackrel{?}{<} (\delta x_1 + \delta x_2) \tag{11}$$

the result is largely acceptable, since the real room temperature, measured by the digital hygro-thermometer, was 24.7 ± 0.1°C (i.e. 297.85 ± 27.32 K), and so the two measurements are compatible.

$$|293.27 - 297.85| = 4.58 < (15.76 + 27.32) = 43.08 \tag{12}$$

3 Second Implementation: 1.1V Internal Reference

3.1 Hardware Design

As it can be clearly seen from the Table 2, the measure of the temperature (that is equal to 20.12 °C) is affected by a high degree of uncertainty; so, in order to decrease it, the voltage reference of the internal ADC was changed and the internal 1.1V was chosen.
But this choice rose a problem: since the voltages expected across the sensor were included between 2.78V and 3.48V (as described in the previous sections), they would not have been considered as they are, but "truncated" to the same value of the ADC reference. For this reason, a voltage divider was needed to reduce the voltage scale so as to make it fit the reference one.
On account of the fact that the maximum voltage could have been 3.48V, the goal was to reduce the value of a factor equal to:

$$reduction factor = \frac{3.48V}{1.1V} = 3.16 \tag{13}$$

Reminding the transfer function and the design of a simple voltage divider:

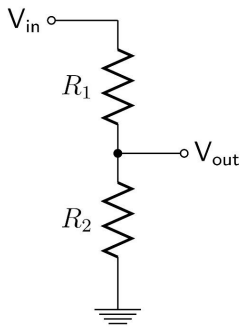

Fig. 7: Simple voltage divider

$$V_{out} = \frac{R_2}{R_1 + R_2} * V_{in} \tag{14}$$

it ended up that

$$\frac{V_{in}}{V_{out}} = \frac{R_1 + R_2}{R_2} \geq 3.16 \tag{15}$$

In order to satisfy this requirement, the chosen resistances were:

- R_1 = 2.2 kΩ (real value equal to 2.15648 ± 0.10022kΩ)
- R_2 = 1 kΩ (real value equal to 0.98132 ± 0.10009kΩ)

which allowed to implement a reduction factor of 3.19753.

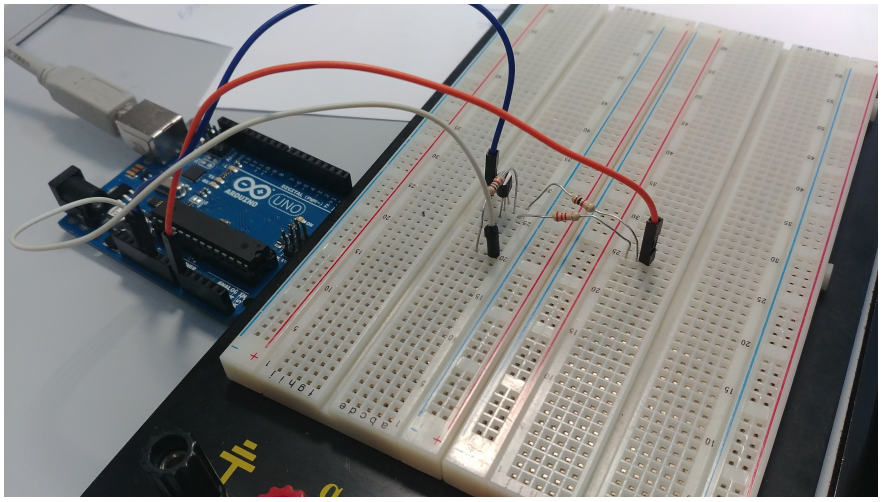

Fig. 8: Implementation of the second circuit on the breadboard

3.2 Firmware Design

The firmware required to complete this task is almost identical to the one described previously; for this reason it was decided to create a single sketch and comment the lines which were not utilized (as it can be observed in the Figure 5).
In the *setup()* the function *analogreference(INTERNAL)* is called to switch from the 5V (set by default) to the internal 1.1V. Then, the temperature and the error are computed. It is important to underline that, since V_out was reduced by 3.19753 (compared to the first thermometer implementation), the measure of T would have suffer from the same reduction, as it is shown below:

$$T = \frac{V_{outreduced}}{S} \Longrightarrow T = \frac{V_{out}}{3.19753 * S} \tag{16}$$

To avoid this collateral effect, the same reduction factor was included into the calibration function.
In order to be clearer, the difference between the functions is reported below.

```
temp= 100*input*V_ref/1024;  //temp= (319.753*input*V_int)/1024;
```

Fig. 9: Comparison between the two calibration functions implemented

```
Done compiling.

Sketch uses 3798 bytes (11%) of program storage space. Maximum is 32256 bytes.
Global variables use 218 bytes (10%) of dynamic memory, leaving 1830 bytes for local variables. Maximum is 2048 bytes.
```

Fig. 10: Sketch dimensions

3.3 Computation of the Uncertainty

The last thing to do was to calculate the uncertainty affecting the second measure.

```
error= (V_ref/10.24)*delta_D_5 + (input/10.24)*delta_V_ref + delta_T_sensor;
//error= (V_int/(10.24*3.19753)*delta_D_11 + (input/(10.24*3.19753))*delta_V_int + delta_T_sensor;
```

Fig. 11: Comparison between the two error functions implemented

As it is shown in Figure 9, the only differences between the two functions which compute the aformentioned error are:

- the value of the reference;
- the absolute uncertainty of the ADC(delta_D_5 and delta_D_11);
- the uncertainty of the reference (delta_V_ref and delta_V_int);
- the 3.19753 factor, that appears because of the relation showed by the formula (14).

Finally, data provided by the second thermometer is shown in the table below.
It is important to highlight that the two measurements were made in two different days, so the temperatures measured were dissimilar.

ADC Output	Temperature (K)	Uncertainty (K)
874.0	297.48	2.70

Table 3: Results provided by the thermometer

Also in this case the result is acceptable, since the temperature of the room, measured by means of the Greisinger GFTH 95, was 24.1 $\pm$ 0.1 °C (i.e. 297.25 $\pm$ 27.32 K), and referring to the formula (11),

$$|297.48 - 297.25| = 0.23 < (2.70 + 27.32) = 30.02 \tag{17}$$

the two measurements are compatible.