Politecnico di Torino

Testing and Certification

Fourth Laboratory Session

Implementation of a Digital Thermometer using an Arduino UNO Board and Pt100 Resistive Sensor

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

Professor:
Carullo Alessio

4 December 2017

POLITECNICO
DI TORINO

1 Introduction

In this laboratory session the goal was to develop a digital thermometer using a platinum resistive temperature detector (PRTD) and handling the outgoing voltage from the sensor by means of the Arduino Uno board.
This laboratory report will illustrate the hardware design, i.e. conditioning circuitry and physical connection of the components used, and also the software development in Arduino environment. At the end, the results, i.e. measurements and uncertainty, will be provided.

1.1 Instrumentation

The instruments used to perform the described task are:

- Arduino Uno board;
- PRTD Pt100 class II – (B);
- 1kΩ resistor;
- HP 34401A Multimeter (to measure the correct value of the resistance);
- Greisinger GFTH 95, a Digital Hygro-Thermometer;

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	Temperatur Coefficient /° 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 + 0.00
	1.000000 V		0.0020 + 0.0006	0.0030 + 0.0007	0.0040 + 0.0007	0.0005 + 0.00
	10.00000 V		0.0015 + 0.0004	0.0020 + 0.0005	0.0035 + 0.0005	0.0005 + 0.00
	100.0000 V		0.0020 + 0.0006	0.0035 + 0.0006	0.0045 + 0.0006	0.0005 + 0.00
	1000.000 V		0.0020 + 0.0006	0.0035 + 0.0010	0.0045 + 0.0010	0.0005 + 0.00
Resistance [4]	100.0000 Ω	1 mA	0.0030 + 0.0030	0.008 + 0.004	0.010 + 0.004	0.0006 + 0.00
	1.000000 kΩ	1 mA	0.0020 + 0.0005	0.008 + 0.001	0.010 + 0.001	0.0006 + 0.00
	10.00000 kΩ	100 µA	0.0020 + 0.0005	0.008 + 0.001	0.010 + 0.001	0.0006 + 0.00
	100.0000 kΩ	10 µA	0.0020 + 0.0005	0.008 + 0.001	0.010 + 0.001	0.0006 + 0.00
	1.000000 MΩ	5 µA	0.002 + 0.001	0.008 + 0.001	0.010 + 0.001	0.0010 + 0.00
	10.00000 MΩ	500 nA	0.015 + 0.001	0.020 + 0.001	0.040 + 0.001	0.0030 + 0.00
	100.0000 MΩ	500 nA || 10 MΩ	0.300 + 0.010	0.800 + 0.010	0.800 + 0.010	0.1500 + 0.00

Fig. 1: Specifications provided by the DMM datasheet

2 Theoretical Approach and Hardware Design

The used Pt100 sensor is a multipurpose temperature sensor that changes its resistance based on temperature. It is provided of 4 wires, so it allows a 3-wire or a 4-wire connection, and it is also able to measure temperature in the range (-50÷250)°C. In base on the class selected the accuracy changes and furthermore it must be taken into account that a high dissipated power of the sensor causes an auto overheating and this process could lead to a wrong temperature measurement, very different from the real one. The specific sensor used in this lab session has a resistance R0 at 0°C equal to 100Ω and it changes based on temperature following this relationship:

$$R_\theta = R_0 \cdot (1 + A \cdot \theta + B \cdot \theta^2) \tag{1}$$

with

$$A = 3.9083 \cdot 10^{-3} (^\circ C)^{-1} \tag{2}$$

$$B = -5.775 \cdot 10^{-7} (^\circ C)^{-2}, \tag{3}$$

and since the sensor class is Class B,

$$\delta\theta = 0.3^\circ C + 0.005 \cdot \theta \tag{4}$$

and its auto overheating due to the power dissipation is 100 °C/W.

2.1 Conditioning Circuit

Having said that, the first step was to design a correct conditioning circuit with a correct value of each component used. In order to do this, the voltage divider was chosen as conditioning circuit, so the voltage supply and the additional resistance value were chosen in order to maximize the sensitivity of the sensor, i.e. how large is the variation of the voltage that Arduino has to process in correspondence of 1°C temperature variation.

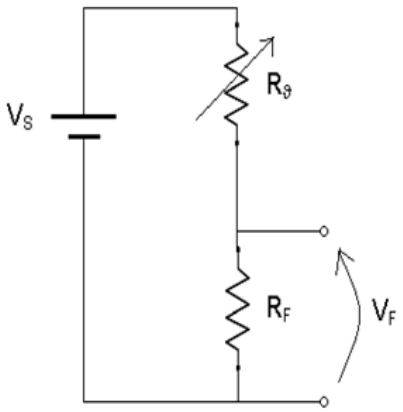

Fig. 2: Schematics of the adopted conditioning circuit

So, mathematically expressing VF, the sensitivity can be computed:

$$V_F = \frac{V_s \cdot R_F}{R_F + R_\theta} = \frac{V_s \cdot R_F}{R_F + R_0 \cdot (1 + A \cdot \theta + B \cdot \theta^2)} \tag{5}$$

$$S^\theta_{V_f} = \frac{\delta V_F}{\delta \theta} = -\frac{V_s \cdot R_F \cdot R_0 \cdot (A + 2 \cdot B \cdot \theta)}{[R_F + R_0 \cdot (1 + A \cdot \theta + B \cdot \theta^2)]} \tag{6}$$

So it is correct to choose a high power supply but it must pay attention to self overheating. Furthermore, the value that maximizes the other factor was found and it was RF = R0 = 100Ω. Apparently it seems that there is nothing else to size but it is appropriate to find an upper bound for voltage supply considering the self overheating of the sensor.

$$P_{R_0} = (\frac{V_s}{2})^2 \cdot \frac{1}{R_0} \rightarrow P_{R_0} \cdot 100 \frac{^\circ C}{W} << \delta\theta = 0.3^\circ C \rightarrow V_s << 1.095V \tag{7}$$

With this value of V_s the maximum sensitivity (more or less equal to 0.98mV/°C, that is very small) was provided. For this reason other values were needed. In fact, a higher value for RF was chosen between 5kΩ, 2kΩ and 1kΩ, maintaining fixed Vs (provided by the Arduino Uno board) equal to 5V in order to obtain a large sensitivity.

V_s [V]	R_F [$k\Omega$]	Sensitivity [$\frac{mV}{^\circ C}$]
5	5	0.37
5	2	0.86
5	1	1.54

Table 1: Results provided by the thermometer

So a RF equal to 1kΩ was chosen, obtaining a sensitivity equal to 1.54mV/°C and a temperature variation, due to the heating of the sensor, equal about to 0.2°C that was smaller than the sensor accuracy.

2.2 Connection and ADC characteristics

The voltage VF was the one that the Arduino Uno board had to read and convert into a digital number (as it is possible to see from the Fig. 2) and it depended on the voltage across the sensor, that in turn depended on the sensor resistance, whose value changed according to the temperature. Since the passive resistances of the wire used to connect the 5V provided by the board to the sensor could have altered the voltage VF read by Arduino, in order to minimize this error a 4-wire connection was adopted.

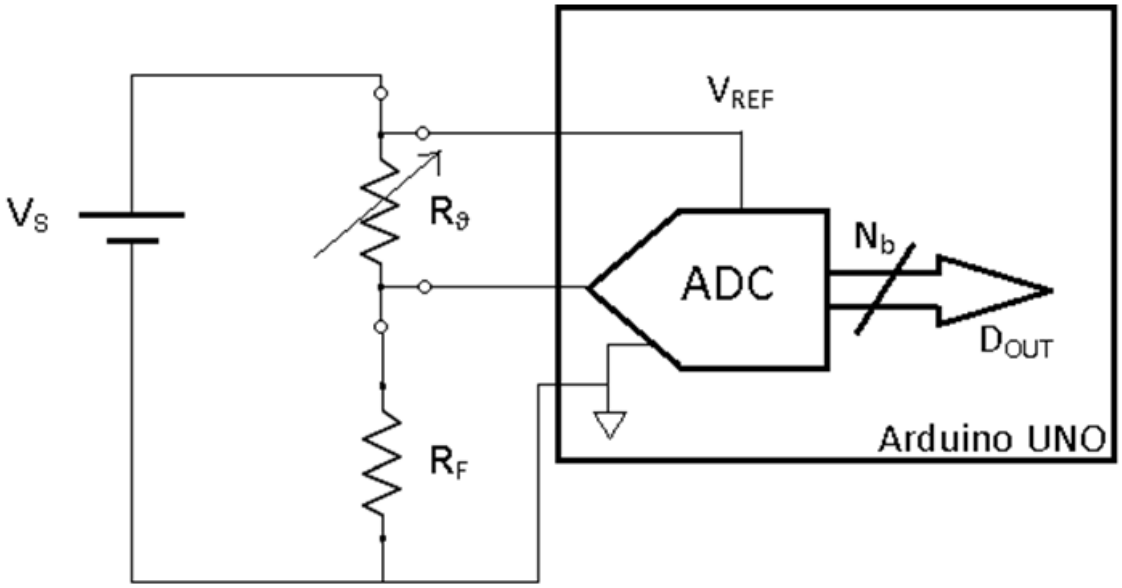

Fig. 3: Schematics of the adopted connection

- The first pin of the sensor was connected to the 5V pin of the Arduino Uno board;
- the second pin was connected to V_ref, so the ADC external reference was not 5V only, but 5V less the voltage drop due to the wire resistance;
- the third pin was connected to the A3 Arduino pin;
- the last pin was connected to the resistance R_F.

3 Micro-Controller Firmware Design

```
const double R_f = 989.23;                //fixed R expressed in Ohm...theoretical value=1kOhm
const double R_0 = 100;
const double A = 3.9083E-3;
const double B = -5.775E-7;
const double C = -4.183E-12;
const double R_thermal = 100;             //thermal resistance...it's value is °C/W
const int n_bits = 10;
double analog_in;
double bracket_term;
double first_sqrt_term;
double temp;
double S_Dout_teta;

void setup() {

  pinMode(A3,INPUT);
  Serial.begin(9600);
  analogReference(EXTERNAL);
  delay(1000);
}

void loop() {

  analog_in = analogRead(A3);
  Serial.println("analog_in");
  Serial.println(analog_in);
  bracket_term = (R_0 + R_f - (R_f*pow(2,n_bits)/analog_in))/(R_0*B);
  first_sqrt_term = pow(A,2)/(4*pow(B,2));
  temp = - A/(2*B) - sqrt(first_sqrt_term - bracket_term);

  Serial.println("TEMP:");
  Serial.println(temp);
  delay(100);
}
```

Fig. 4: Written firmware

The above firmware is a sketch that implements the temperature measurements requested. Leaving aside for a moment the first part concerning the declaration of variables and constant values, the function called *setup()* was the part of the sketch responsible to set the communication serial speed, to which pin the signal arrives (A3) and the voltage reference with which the Arduino ADC has to work. For the latter setting, since the voltage reference of the ADC had to be equal to the voltage on the second pin (so the difference between the Arduino 5V and the potential drop due to the wire resistance), *analogReference(EXTERNAL)* was used and from the hardware point of view the voltage of the second pin of the sensor was applied to the V_{ref} pin of the Arduino Uno board.
Moreover, in the "loop" function the real measurement was implemented. First of all, the input voltage was read, converted into a digital number and stored in the variable $analog_{in}$, already defined into section concerning the declaration, and then the calibration function was implemented using the A and B parameters, as well as the resistances values, also de-

fined in the declaration section. In order not to create confusion, the calibration function was split into several lines: the first $bracket_{term}$ and the second $first_{sqrt\_term}$ one computed intermediate values and in the last one *temp* they were put together to obtain the final result. At the end, the critical and the most important data was printed on video ($analog_{in}$, *temp*). The calibration function was obtained reversing the formula used to find VF:

$$\theta = -\frac{A}{2 \cdot B} - \sqrt{\frac{A^2}{4 \cdot B^2} - \frac{1}{R_0 \cdot B} \cdot \left(R_0 + R_F - \frac{V_s}{V_F} \cdot R_F \right)}, \tag{8}$$

but expressing V_F from the ADC point of view:

$$V_F = \frac{D_{out} \cdot V_{ref}}{2^{Nb}}, \tag{9}$$

where V_{ref} is the voltage reference for the ADC that, DOUT is the digital number, output of the ADC, i.e.$analog_{in}$ in the sketch, while Nb is the number of bits with which the ADC works. For this reason the calibration function become:

$$\theta = -\frac{A}{2 \cdot B} - \sqrt{\frac{A^2}{4 \cdot B^2} - \frac{1}{R_0 \cdot B} \cdot \left(R_0 + R_F - \frac{2^{Nb}}{D_{out}} \cdot R_F \right)}. \tag{10}$$

In this way it is possible to see that the number of bits affects the temperature resolution:

$$\Delta = S^{\theta}_{V_F} \cdot V_q, \; with \, V_q = \frac{V_{ref}}{2^{Nb}}. \tag{11}$$

Moreover, when the temperature measurement uncertainty is computed, it will be possible to see that the uncertainty doesn't depend on the voltage reference. In this lab session a number of bits equal to 10 was used and this choice led to a temperature resolution of about 3°C. So, in the measurements, it will have to be taken into account that a quantization interval corresponds to almost 3°C.

4 Measurements, Uncertainty and Results

4.1 Ambient Temperature Measurements

Setting the number of bits equal to 10 in the sketch and starting the program, the ambient temperature value was obtained and printed on video with the corresponding digital number. Finally, starting from the calibration function, the temperature measurement uncertainty was computed, considering:

- $\delta D_{out} = 2\text{LSB}$;
- $R_F = 987.23 \pm 0.20\Omega$;
- $\delta\theta_{sensor} = 0.3°C + 0.005 \cdot \theta$.

$$\delta\theta = \left|\frac{\delta\theta}{\delta D_{out}}\right| \cdot \delta D_{out} + \left|\frac{\delta\theta}{\delta R_F}\right| \cdot \delta R_F + \delta\theta_{sensor}. \tag{12}$$

ADC Output	Temperature [°C]	Uncertainty [°C]
922.00	24.23	4.04

Table 2: Table representing the results of the measurement

According to the formula

$$|x_1 - x_2| \overset{?}{<} (\delta x_1 + \delta x_2) \tag{13}$$

the result is largely acceptable, since the real room temperature, measured by the digital hygro-thermometer, was 24.3 ± 0.1°C (i.e. 297.45 ± 27.32 K), and so the two measurements are compatible.
The obtained result was compared to the ones got from the LM335 sensor, and they are shown in the following tables:

ADC Output	Real Temperature [K]	Temperature [K]	Uncertainty [K]
606.0	297.85	293.27	15.76

Table 3: Results provided by the LM335+Arduino system with 5V reference

ADC Output	Real Temperature [K]	Temperature [K]	Uncertainty [K]
874.0	297.25	297.48	2.70

Table 4: Results provided by the LM335+Arduino system with 1.1V reference

As it can be clearly seen, the first implementation of the thermometer based on the LM335 (the one with 5V reference) presented a great difference between the real and the measured temperature and a high degree of uncertainty, so the Pt100 based system was preferable. On the other hand, changing the reference to 1.1V, the LM335 based thermometer presented a difference between the real and the measured temperature slightly higher than the Pt100 one (0.23 K vs. 0.07 K respectively), while the uncertainty is lower; so, in order to choose which one of the two configurations should be used, it must be taken into account which one between precision or uncertainty has more importance for the user.

Finally, the sensor was got warm in order to verify the temperature value increasing on video and the following was obtained:

ADC Output	Temperature [°C]	Uncertainty [°C]
921.00	27.31	4.06

Table 5: Table representing the results of the measurement

4.2 Cold Water Temperature Measurement

As last measurement a bowl full of cold water (at a temperature of 6.0 ± 0.1 °C) was provided and the temperature measurement was requested. So immersing the sensor in the bowl, the temperature value on video started to decrease until it remains stable at a value equal to 5.98°C.

ADC Output	Temperature [°C]	Uncertainty [°C]
929.00	5.98	3.88

Table 6: Table representing the results of the measurement