Politecnico di Torino

Testing and Certification

Fifth Laboratory Session

Implementation of a Digital Thermometer using a Negative Temperature Coefficient Thermistor and an Arduino UNO Board

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

Professor:
Carullo Alessio

11 December 2017

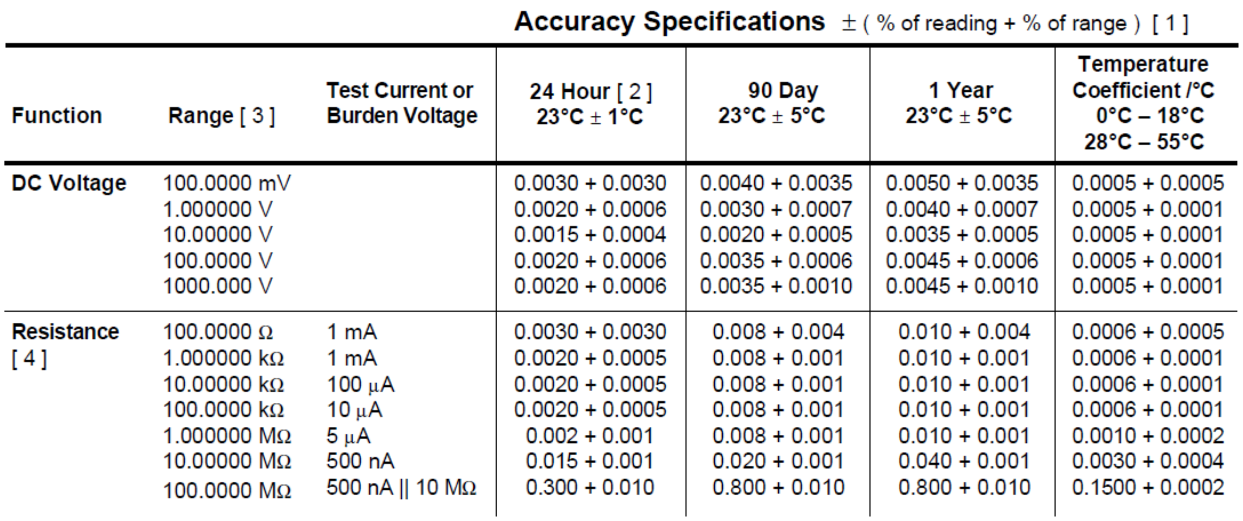
1 Introduction

This fifth laboratory session aimed to assemble an entire conditioning circuit and, then, to implement a digital thermometer based on a B57045 thermistor.
This laboratory report will present the whole process of hardware design and microcontroller programming, showing and explaining the collected data and the provided results.

1.1 Instrumentation

During the laboratory experiment, the instruments used to perform the aformentioned task were:

- Arduino UNO Board;
- B57045 thermistor;
- 10kΩ resistor;
- HP 34401A Multimeter (to measure the real 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

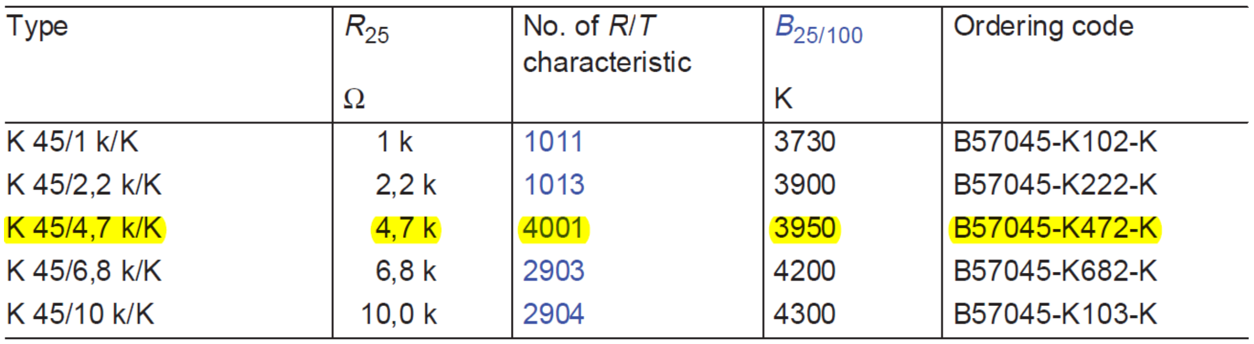
2 Theoretical Approach and Hardware Design

First of all, by means of a NTC thermistor, it is possible to compute the temperature using the following formula:

$$T = \frac{1}{\frac{1}{T_0} + \frac{1}{\beta} \cdot \ln\left(\frac{R_T}{R_0}\right)}, \tag{1}$$

where

- R_0 is the value of the resistance $@T_0$;
- β is a characteristic parameter of the sensor;
- $$R_T = A \cdot e^{\frac{\beta}{T}}. \tag{2}$$



Type	R_{25} Ω	No. of R/T characteristic	$B_{25/100}$ K	Ordering code
K 45/1 k/K	1 k	1011	3730	B57045-K102-K
K 45/2,2 k/K	2,2 k	1013	3900	B57045-K222-K
K 45/4,7 k/K	4,7 k	4001	3950	B57045-K472-K
K 45/6,8 k/K	6,8 k	2903	4200	B57045-K682-K
K 45/10 k/K	10,0 k	2904	4300	B57045-K103-K

Fig. 2: Specifications provided by the thermistor datasheet

As it is possible to see from Fig.2, the values of both β and R_0 used for this laboratory session were, respectively, 3950 K and 10kΩ at 25°C.

The chosen conditioning circuit to be implemented was a simple voltage divider, whose implementation is shown in Fig.3, and whose output voltage is given by the formula:

$$V_{out} = V_s \cdot \frac{R_F}{R_F + R_T}. \tag{3}$$

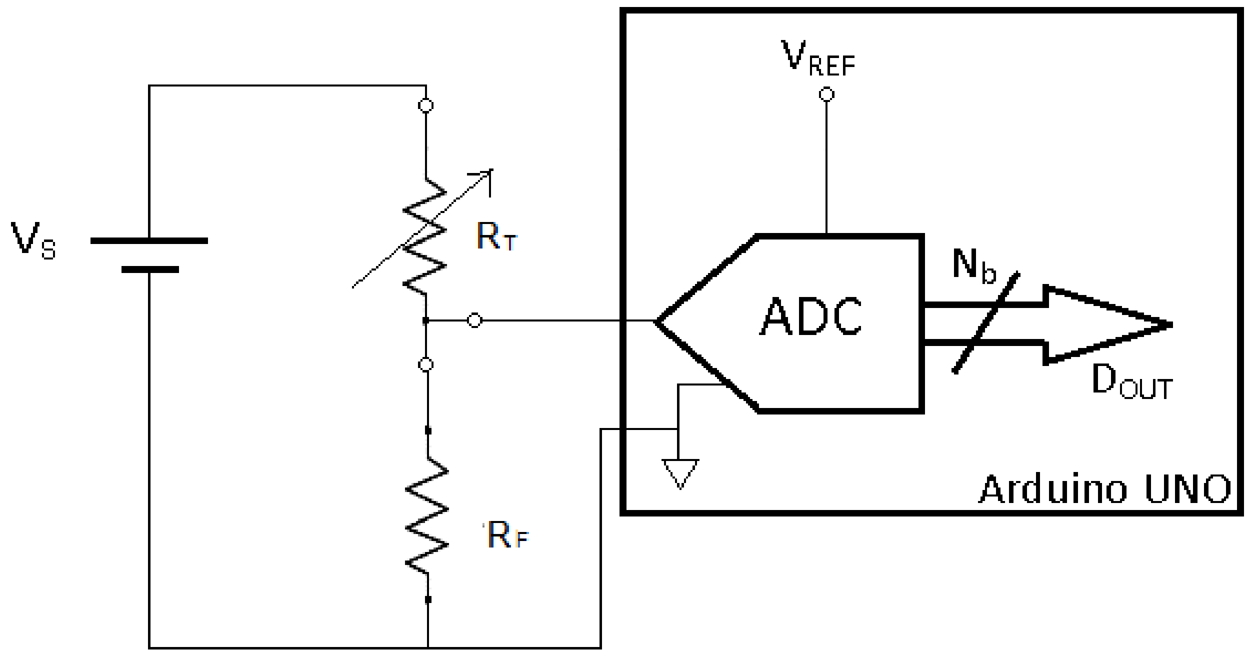


Fig. 3: Design of the whole system

Since V_s corresponded to the 5V provided by the Arduino board, and (as it will be seen) it was chosen to set the ADC reference to the same 5V of the board, it was possible to say that:

$$V_s = V_{ref}. \tag{4}$$

Having this in mind, the following computations show how the implemented calibration function was found.
By looking at the formula (3) and reorganize it,

$$R_T = R_F \cdot \frac{V_s - V_{out}}{V_{out}}, \tag{5}$$

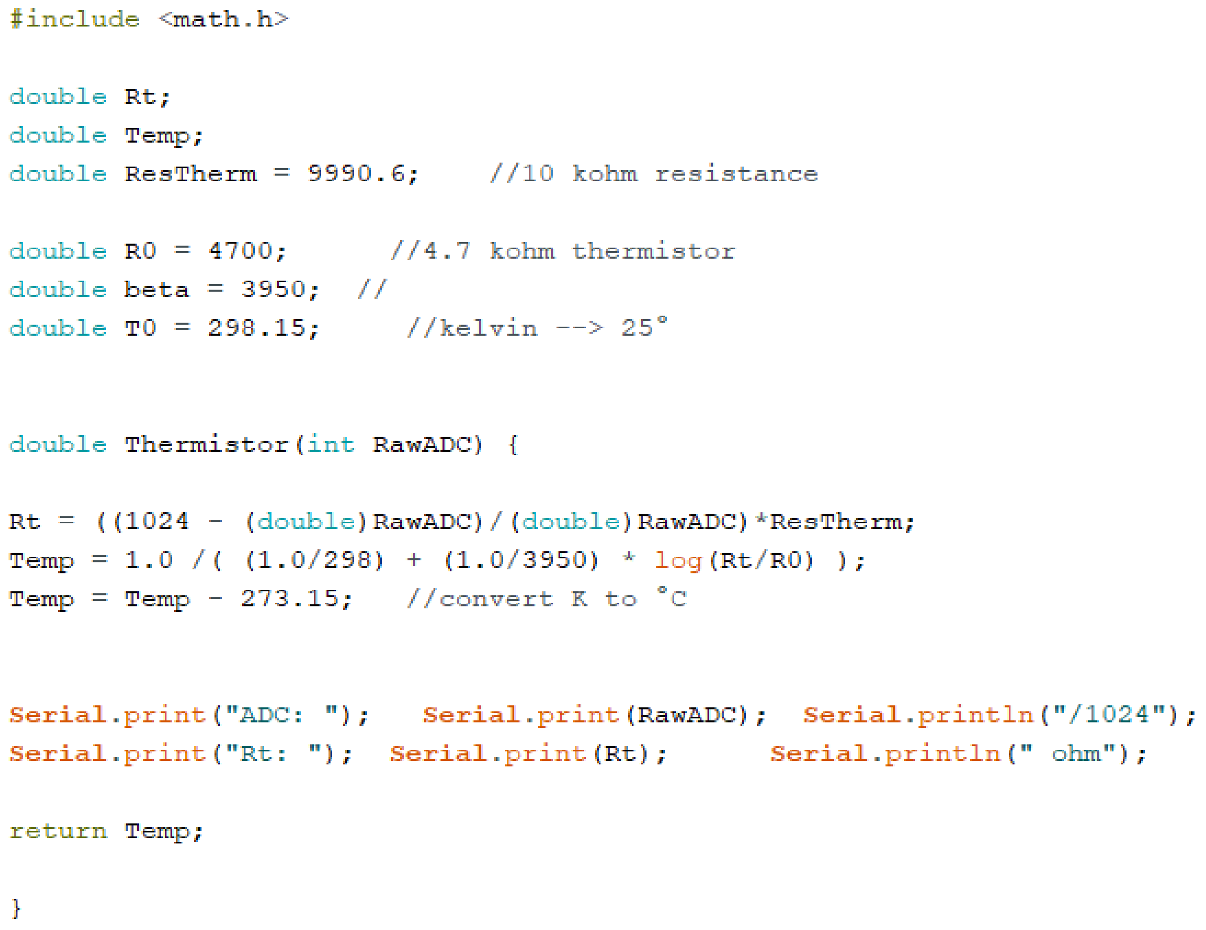
but V_{out} can be expressed as

$$V_{out} = D_{out} \cdot V_q = \frac{D_{out} \cdot V_s}{2^{Nb}}. \tag{6}$$

At the end, using both (5) and (6)...

$$R_T = R_F \cdot \frac{V_s - \frac{D_{out} \cdot V_s}{2^{Nb}}}{\frac{D_{out} \cdot V_s}{2^{Nb}}} \rightarrow R_T = \left(\frac{2^{Nb}}{D_{out}} - 1 \right) \tag{7}$$

3 Micro-Controller Firmware Design



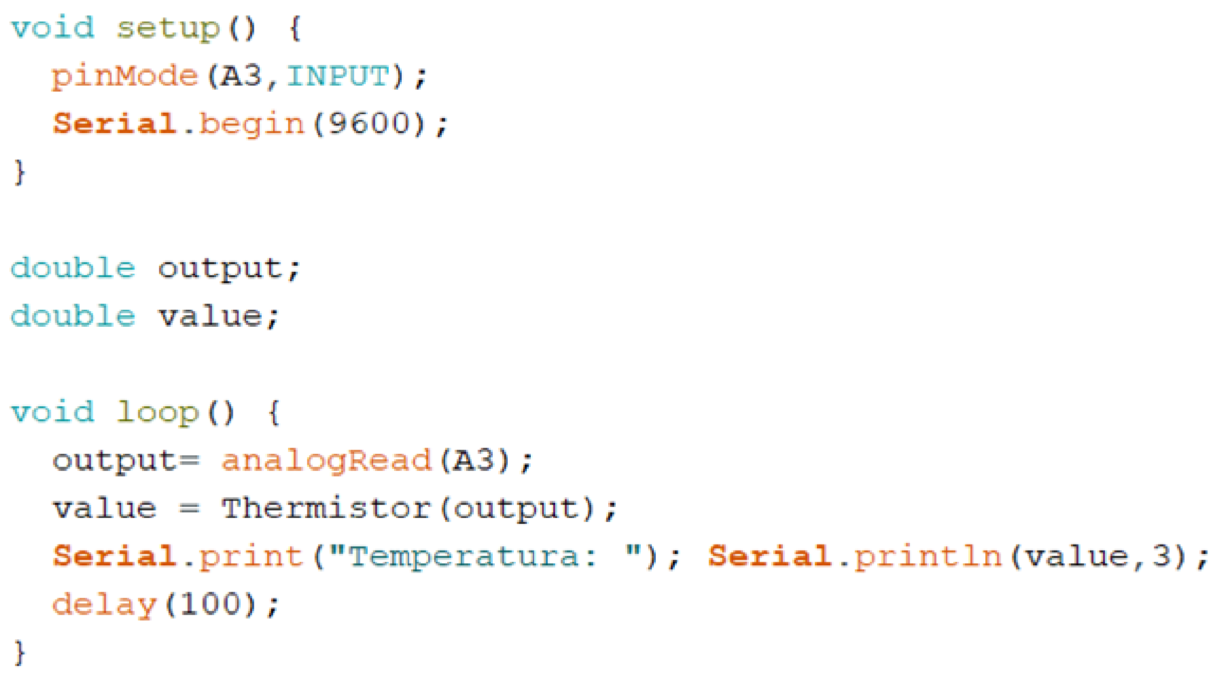
```
#include <math.h>

double Rt;
double Temp;
double ResTherm = 9990.6;    //10 kohm resistance

double R0 = 4700;      //4.7 kohm thermistor
double beta = 3950;  //
double T0 = 298.15;     //kelvin --> 25°


double Thermistor(int RawADC) {

Rt = ((1024 - (double)RawADC)/(double)RawADC)*ResTherm;
Temp = 1.0 /( (1.0/298) + (1.0/3950) * log(Rt/R0) );
Temp = Temp - 273.15;   //convert K to °C


Serial.print("ADC: ");   Serial.print(RawADC);  Serial.println("/1024");
Serial.print("Rt: ");  Serial.print(Rt);      Serial.println(" ohm");

return Temp;

}
```

Fig. 4: Function which computes the temperature

The code reported by Fig.4 is the first half of the firmware sketch.
Initially, several constants are initialized, and more specifically:

- *ResTherm*, which indicates the real value of R_F; since it was computed by means of the HP 34410A multimeter, its value is equal to $9.99060 \pm 0.00110\ k\Omega$;
- R_0, that indicates the $4.7k\Omega$ resistance of the thermistor;
- β, that indicates the 3950 K;
- T_0, set to 298.15 K (25°C).

Then, the formulas (7) and (1) are implemented, in order to obtain the value of the temperature.



```
void setup() {
  pinMode(A3,INPUT);
  Serial.begin(9600);
}

double output;
double value;

void loop() {
  output= analogRead(A3);
  value = Thermistor(output);
  Serial.print("Temperatura: "); Serial.println(value,3);
  delay(100);
}
```

Fig. 5: Setup and loop functions

In the second half of the code two functions are represented.
The first one is the *setup* one, in which the communication is initialized and the A3 pin is set as INPUT (the *analogReference* was not used since by default is set to 5V).
The second function is the *loop*, in which the A3 pin continuously reads the input voltage, converts it into a digital number and uses it to compute the temperature value (recalling the *Thermistor(output)* function.

4 Results and Uncertainty

Starting the program, the ambient temperature value was obtained and printed on video. Finally, starting from the calibration function, the temperature measurement uncertainty was computed, considering:

- $\frac{\delta\beta}{\beta} = 3\% \rightarrow \delta\beta = 118.5K$;
- $\frac{\delta R_0}{R_0} = 10\% \rightarrow \delta R_0 = 470\Omega$;
- $\delta D_{out} = 2\text{LSB}$.

$$\delta T = \left|\frac{\partial T}{\partial \beta}\right| \cdot \delta\beta + \left|\frac{\partial T}{\partial D_{out}}\right| \cdot \delta D_{out} \tag{8}$$

$$\delta T = \left|\frac{T_0^2 \cdot \ln\left(\frac{2^{Nb}}{D_{out}} - 1\right)}{\left[T_0^2 \cdot \ln\left(\frac{2^{Nb}}{D_{out}} - 1\right) + \beta\right]^2}\right| \cdot \delta\beta + \left| - \frac{2^{Nb} \cdot T_0^2 \cdot \beta}{(D_{out} - 2^{Nb}) \cdot D_{out} \cdot \left[T_0^2 \cdot \ln\left(\frac{2^{Nb}}{D_{out}} - 1\right) + \beta\right]^2}\right| \cdot \delta D_{out} \tag{9}$$

4.1 Single Reading Technique

ADC Output	R_T [Ω]	Temperature [°C]	Uncertainty [°C]
670.00	5278.62	22.262	0.689

Table 1: Table representing the results of the measurement

According to the formula

$$|x_1 - x_2| \overset{?}{<} (\delta x_1 + \delta x_2) \tag{10}$$

the result is acceptable, since the temperature measured by means of the hygro-thermometer was 22.4 ± 0.1 °C.

4.2 Multiple Reading Technique

In order to have a more precise measurement, trying to minimize the effect of the noise on the measurement, a multiple reading technique (50 samples) was implemented, and its result is shown in the table below.

R_T [Ω]	Temperature [°C]	Uncertainty [°C]
5250.21	22.310	0.652

Table 2: Table representing the results of the multiple reading technique

As it can be clearly seen, the usage of this measurement method improved (even if the result provided by the single reading was already good) the measurement of the real room temperature.

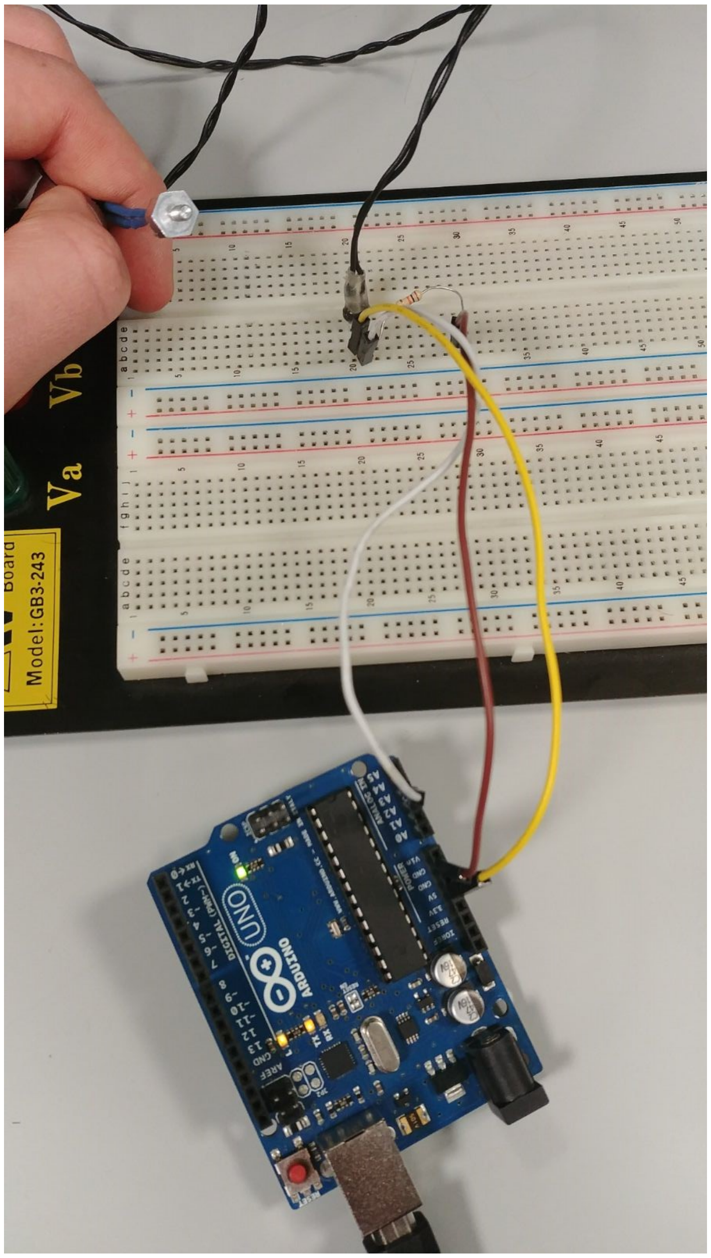


Fig. 6: Real hardware implementation