POLITECNICO
MILANO 1863

Academic Year 2015-2016

Tecnology for sensors and clinical instrumentation

# RESPIRATORY RATE MEASUREMENT BY NTC THERMISTOR


Professor ANDREA ALIVERTI

Tutor DARIO BOVIO




Andrea Mariani

Oriella Gnarra

Lorenzo Marchesi

Maria Paola Forte

Riccardo Salerno

# 1. INTRODUCTION

The aim of this project is to provide a measurement of the respiratory rate.

Respiratory rate is defined as the number of cycles of inspiration and expiration per unit time or, more generally, it is the number of breaths per minute:

$$f_b = \frac{\#\ breaths}{1\ minute} \qquad [1.1]$$

It is one of the four vital signs (including pulse rate, blood pressure and body temperature) that are considered standard for monitoring patients on acute hospital wards. A normal respiratory rate has a range of 12 ÷ 20 [breaths per minute] in adults and it is termed as *eupnea*; a higher one, *tachypnea* and a lower one, *bradypnea*.

Many types of respiratory rate measurement have been used until now: for instance, non-contact methods based on the use of ultrasonic sensors or facial tracking algorithms; otherwise, techniques built on thorax volume variations (such as in impedance pneumograph) or MEMS like capacitive pressure sensor; finally, temperature sensors can be employed for this purpose with economic advantage.

The latter method is the one used in the present work. In fact, the estimation of the rate is obtained through a thermally sensitive resistor (NTC – Negative Temperature Coefficient) whose resistance varies according to the changes in temperature. This variation is due to the subject's respiration inside the tube where the sensor is placed.

The next chapters explain the device in its main parts: Hardware (*Chapter 2*), Firmware (*Chapter 3*) and Software (*Chapter 4*). In the last chapter (*Chapter 5*) results are presented.

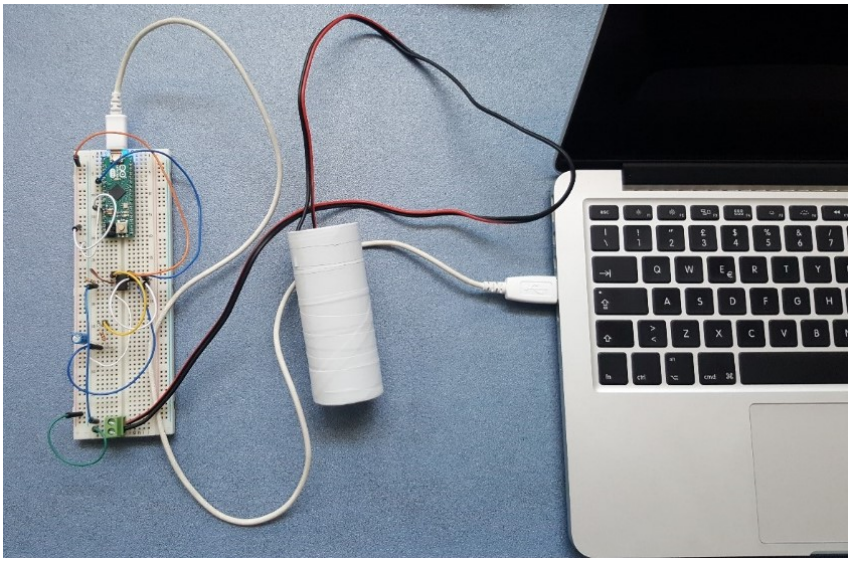


*Figure 1.1: Final device based on NTC thermistor*

# 2. HARDWARE

In order to develop the project – described in the introductive chapter – the following schematic is used:

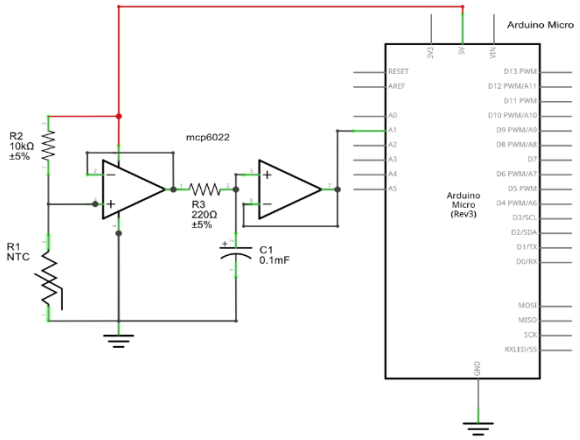



*Figure 2.1: Schematic of project's circuit*

In *Figure 2.2* the previous schematic is represented on the breadboard.

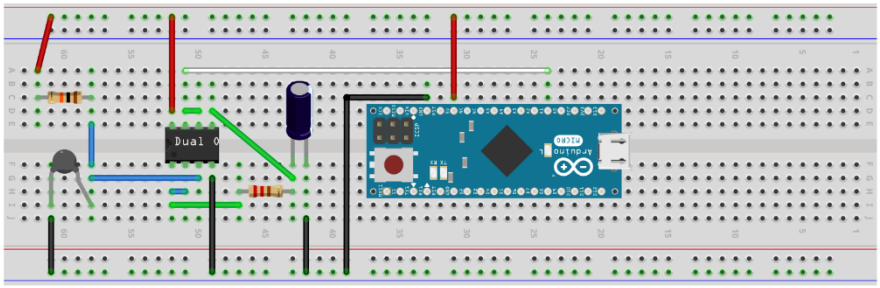


*Figure 2.2: Breadboard of project's circuit*

The basic component used to carry out this project is the Negative Thermal Coefficient (**NTC**) thermistor. In order to guarantee proper diffusion along the NTC thermistor, different methods were tested - such as bottle's necks, masks, and empty toilet paper rolls. Then, it was experimentally proved that the latter shows the best results. This is due to its properties: firstly, thanks to its section, it allows a proper passage of air (differently from a mask, which creates an isolated environment); secondly, paper is a hygroscopic substance and consequently it prevents problems due to the moisture (one of the main cause of failure for thermistors). The NTC sensor's terminals were lengthened and the sensor was fixed inside the tube.

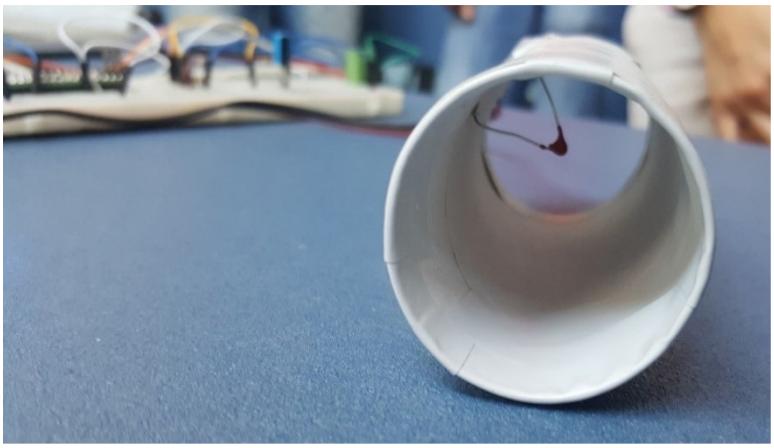


*Figure 2.3: Device and circuit in the background*

Thermistors are based on the dependence of resistance on the number of available charge carriers, according to the law:

$$\frac{1}{\rho} = \sigma = \mu N q\,, \qquad [2.1]$$

where $\rho$ is the *electrical resistivity* [Ω*m], $\sigma$ is the *electrical conductivity* [S/m], $\mu$ is the *charge mobility* [$m^2$/(V*s)], N is the *number of available charge carriers* [ ] and q is the *elementary charge* [C]. As a consequence, materials like semiconductors or metal oxides are able to detect change in temperature because an increase in this measurand causes electrons to pass to the conduction band with the simultaneous holes' generation (i.e. the number of available charge carriers grows). In turn this determines an increase in conductivity that means a reduction of resistance.

In the NTC sensors, the temperature dependence of the resistance can be considered exponential:

$$R_T = R_0 \cdot e^{B\left(\frac{1}{T} - \frac{1}{T_0}\right)}\ [\mathrm{k\Omega}], \qquad [2.2]$$

where $R_0$ is the *reference resistance* [kΩ], $T_0 = 273 + 25 = 298$ [K], and $B$ is the *characteristics temperature of the material* [K], which actually increases with an increase in temperature – assumed constant for the purpose of this work.

The current temperature is computed exploiting the temperature-voltage relation, which derives from [2.2] and this relation is:

$$T = \left[\frac{1}{T_0} + \ln\sqrt[B]{\frac{R_2 V_{A1}}{R_0 (V_{DD} - V_{A1})}}\right]^{-1}\ [\mathrm{K}] \qquad [2.3]$$

Another significant characteristic is the sensitivity, defined as:

$$\alpha = \frac{dR_T/dT}{R_T} = -\frac{B}{T^2}\ [\%/°C] \qquad [2.4]$$

This is the *slope* of the characteristic curve and it represents the variation in resistance due to temperature.

The thermistor used in this project (N_03N00103) belongs to the group of non-ferrous materials (material code N) and at the given temperature of $25[°C]$, it shows the following values:

- $R_0 = 10\ [k\Omega]$;
- $B = 4080 \pm 0.03\ [K]$;
- $\alpha = -4.6\ [\%/°C]$.

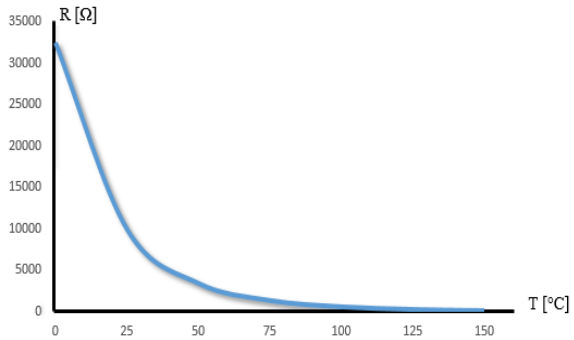



*Figure 2.4: Characteristic Curve of a NTC 10KΩ Thermistor*

Besides NTC, the remaining components used are listed below:

- operational amplifier *MCP6022*;
- two resistances ($R_2 = 10\ [k\Omega]$ and $R_3 = 220\ [\Omega]$);
- electrolytic capacitor ($C_1 = 0.1\ [mF]$);
- Arduino micro.

A brief description of the above-listed components is provided.

The operational amplifier *MCP6022* has eight pins:

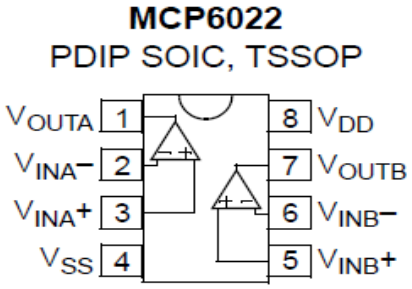



*Figure 2.5: MCP6022 configuration*

1. $V_{OUTA}$ connected to a low-pass passive filter;
2. $V_{INA^-}$ directly connected to $V_{OUTA}$, since this operational amplifier acts as a voltage follower;
3. $V_{INA^+}$ receives the inputs from NTC thermistor and 220 [Ω] resistance;
4. $V_{SS}$ connected to GND;
5. Low-pass filter is the input for the second voltage follower, having as positive input $V_{INB^+}$;
6. $V_{INB^-}$ directly connected to $V_{OUTB}$, since this operational amplifier acts as a voltage follower;
7. $V_{OUTB}$ connected to Arduino (A1);
8. $V_{DD}$ connected to 5 $[V]$ voltage.

The low-pass filter, used to avoid aliasing, has a cut-off frequency of:

$$f_{-3dB} = \frac{1}{2\pi\tau} = \frac{1}{2\pi(R_3 C_1)} = 7.23\ [\text{Hz}] \qquad [2.5]$$

By the mean of the described components, it is possible to obtain useful measurements for subjects of different ages. Taking into account the range $12 \div 60$ breaths per minute (infants $30 \div 60$ and adults $12 \div 20$), the frequency turns out to be $0.2 \div 1\ [Hz]$.

Using the provided components and a proper frequency the final cut-off frequency is $f_{-3dB} = 7.23\ [Hz]$.

Another important decision concerned the value of resistance $R_2$. This resistance, as well as $R_1$ (NTC), is part of a voltage divider. Different values were tested, such as: 1 $[k\Omega]$, 5 $[k\Omega]$, 8.2 $[k\Omega]$, 10 $[k\Omega]$ etc. The latter showed the best experimental results. In fact, few mathematical passages demonstrate that the maximization of the sensitivity leads to the choice of $R_2 = R_0$.[1]

[1] The analog input is given by: $V_{A1} = V_{DD}\frac{R_2}{R_1+R_2}$. Then the sensitivity is: $= \frac{d\,V_{A1}}{dR_1} = V_{DD}\frac{R_2}{(R_1+R_2)^2}$. In order to maximize the sensitivity wrt $R_2$: $\frac{dS}{dR_2} = V_{DD}\frac{R_1-R_2}{(R_1+R_2)^3} = 0 \underset{T=25°}{\Longleftrightarrow} R_2 = R_1 = R_0$. This is actually a maximum according to the negative sign of the second order derivative evaluated in $R_2 = R_1$.

# 3. FIRMWARE

The firmware consists mainly in three parts:

1. ***acquisition*** of the signal;
2. ***communication*** over the serial port;
3. ***visualization*** of the data coming from the A/D converter of the Arduino.

The ***acquisition*** of the signal is written in a single line code in the working loop of the micro:

```
value_NTC=AnalogRead(A1);
```

Arduino analogue port A1 is read in order to collect the data and the read value is stored in the variable `value_NTC`. Arduino analogue input has $10bit$ resolution; therefore, the obtained value is in the range $0 \div 1023$. The analog port A1 reads the value of the voltage divider between the NTC and the resistor $R_1$, and maps it within the given range. Approximately $100\ [ms]$ are required to read the analog input.

In addition, there is a $10\ [ms]$ delay between the reading from the analog input and the sending to the serial port. In this way, a sampling frequency of $100\ [Hz]$ is obtained in compliance with Shannon's Theorem.

The signal under consideration has an approximate frequency of $0.5\ [Hz]$, precisely in the range $0.2 \div 1\ [Hz]$ as mentioned in *Chapter 2*. Moreover, this frequency value was chosen to improve the quality of the temperature plot – a more detailed explanation of the reasons of this choice is given in *Chapter 2*.

Once the signal is acquired, the baud rate of the ***communication***, which represents the number of data transmitted in a second, has to be set. This is the base for digital communication.

```
Serial.begin(9600);
```

This string sets the data rate in bits per second for serial data transmission. For the communication between the microcontroller and the computer, a baud of 9600 bits per second is chosen. The default string of data is 8 data bits (no parity and only one stop bit).

The following step consists in the ***visualization*** of the stored values.

```
Serial.println(value_NTC);
```

It allows to display on the serial monitor the values taken from the analog input. These can be used in the software by reading them on the serial port.

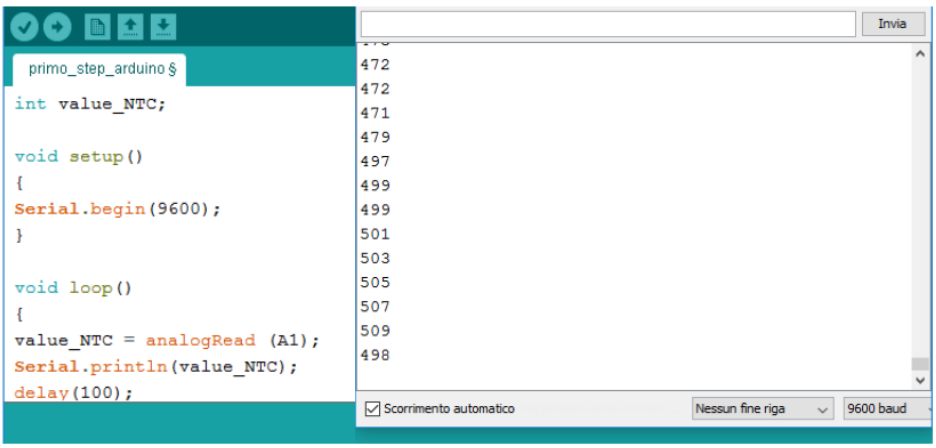



*Figure 3.1: On the left: Firmware sketch. On the right: serial monitor showing changing values of the NTC*

# 4. SOFTWARE

Three main parts constitute the software that is on the base of the project:

- Processing – Arduino serial ***communication***;
- Respiratory rate ***estimation***;
- Temperature plot and respiratory phases ***visualization***.

Firstly, it is necessary to acquire the temperature values (or better the voltage signals) incoming from Arduino.

That implies the use of a serial communication library and the definition of a temperature variable.

```
import processing.serial.*;
Serial myPort;
float temp;
```

Afterwards, in the *setup* process the serial communication is activated with a specific baud rate (paying attention to the serial port that is in use).

```
// List all the available serial ports
println(Serial.list());
// Open whatever port is the one you're using.
myPort = new Serial(this, Serial.list()[3], 9600);
// don't generate a serialEvent() unless you get a newline character:
myPort.bufferUntil('\n');
```

In the end, a function controlled by the occurrence of a new datum (asynchronous serial event) is implemented so that the values coming from Arduino can be read and stored in a desired variable.

This value is an integer number between 1 and $2^{10}$ so that it has to be converted in a voltage range ($0 \div 5\ [V]$). Finally, exploiting the temperature-voltage relation previously obtained [2.3], the current temperature is computed.

```
void serialEvent (Serial myPort) {

  // get the ASCII string:
  String inString = myPort.readStringUntil('\n');

  if (inString != null) {
    // trim off any whitespace:
    inString = trim(inString);

    r= float(inString);

    // CONVERT THE VOLTAGE INPUT INTO TEMPERATURE (IN CELSIUS DEGREES)

    vAN=r*5/1024;
    temp=pow(1./T0+log((R1*vAN/R0)/(5-vAN))*(1./beta),-1)-273.15;


  }
```

The second step is the identification of the respiratory phases: in fact, as the objective is the estimation of the respiratory frequency, the number of breaths (where a breath is the sum of expiration and inspiration) performed by the subject has to be computed so that the rate can be easily obtained as:

$$\hat{f} = \frac{\#\ breaths}{total\ time}\ [Hz] \qquad [4.1]$$

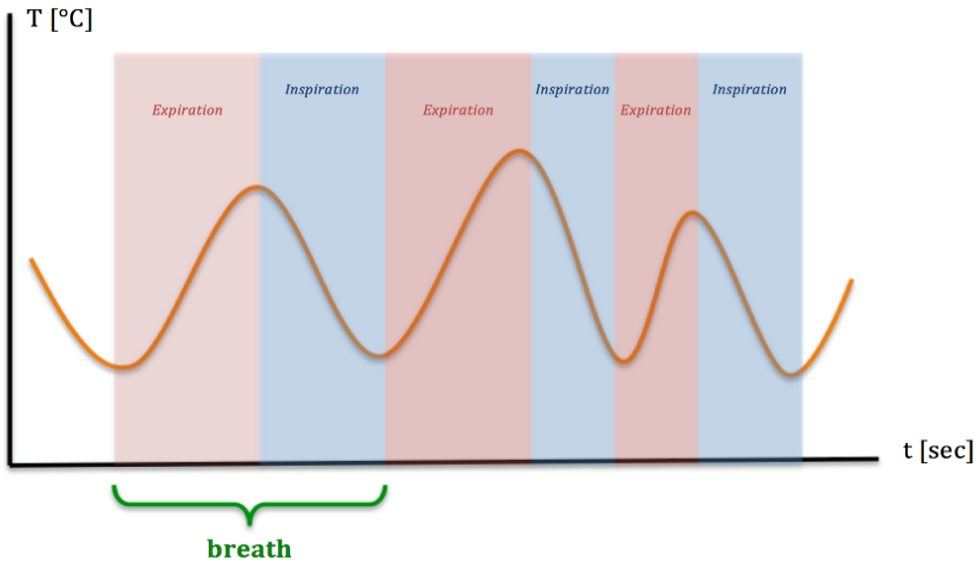



*Figure 4.1: Expiration-Inspiration*

The simplest approach consists in establishing a temperature threshold and calculating the number of breaths as the half number of times the temperature crosses this reference. However, as the figures below put into evidence, this threshold would be "season sensitive" (i.e. it should be adoptive and so modified according to the season of the year and the corresponding ambient temperature) and this approach would be failing also because of the temperature drift at the beginning of the experiment.

Consequently, the respiratory phases can be identified more precisely by detecting the minimum and the maximum of the temperature signal: a maximum corresponds to the beginning of the inspiratory phase (decreasing temperature), while a minimum is associated to the starting point of expiration (increasing temperature).

Notice that only the maximal (or the minimal) points would be sufficient to determine the number of breaths: the identification of both is related to the visual representation of respiration described in the following step.

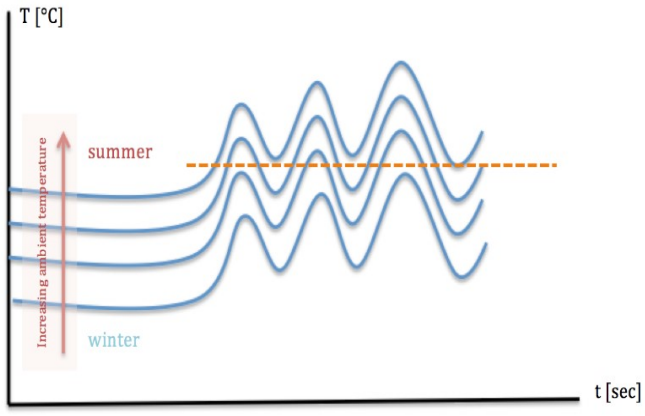



*Figure 4.2: A static threshold is not functional at different ambient temperatures*

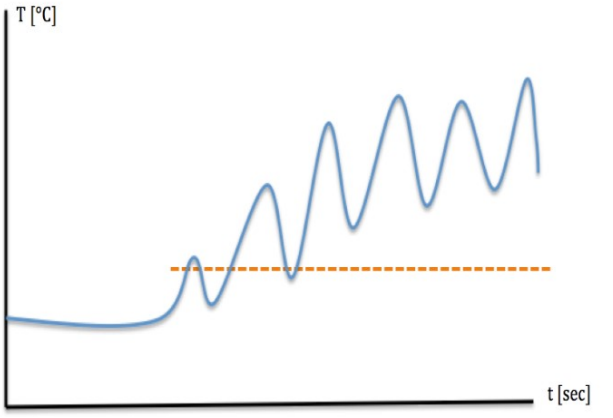
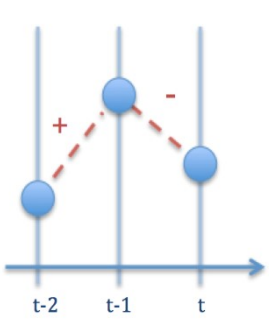
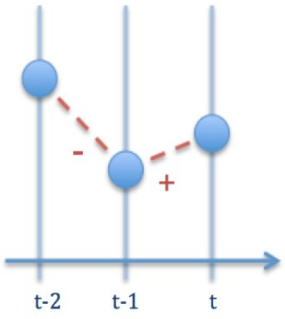



*Figure 4.3: The temperature drift causes the initial threshold not to be functional at different times*

The real time stationary points' detection is performed using the following algorithm: the latest three temperature values are considered and the comparison between the sign of the derivatives allows to identify a maximum or a minimum.




if ( y(t)<y(t-1) && y(t-1)>y(t-2) ) -> y(t-1)=MAX




if ( y(t)>y(t-1) && y(t-1)<y(t-2) ) -> y(t-1)=MIN

This algorithm presents two limits:

a) In the case of two (or more) consecutive equal values of temperature, it is not able to identify a stationary point (see the figure on the right). In order to overcome this limit the derivative is not updated in the case of consecutive equal samples so that only the first change of the signal is compared to the last change of values (if they are concordant, there's no stationary point).

b) The algorithm <u>always</u> computes stationary points: this means that the presence of noise (small amplitude oscillations) causes the counting of *phantom breaths*. In order to avoid this effect, a low pass filter is introduced (time constant so that the respiratory signal - the physiological frequency in an adult is around 12-20 breaths per minute - is not filtered, see *Hardware section*) and a lower number of samples is considered (only 1 over 25 with $f_{samp}$=100Hz).

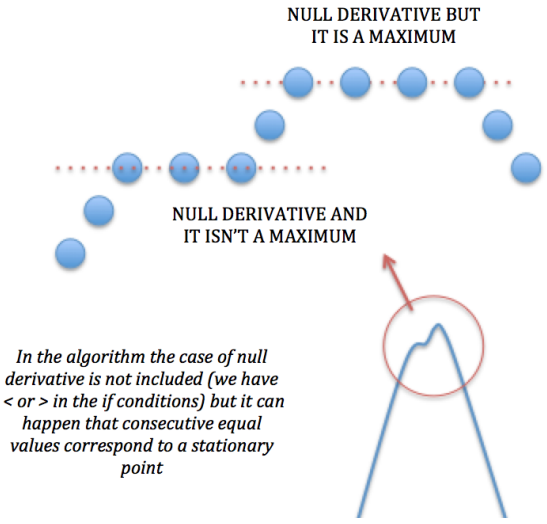



*Figure 4.4: Zero derivative possible cases*

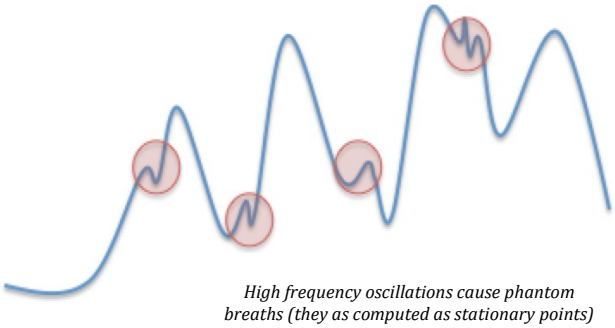



*Figure 4.5: Phantom breaths*

Finally, according to the implemented algorithm, the subject has to begin with expiration so that the measurement can start. This choice is due to a temperature change so that the expiration can be detected more easily because of its bigger amplitude (and the risk of false starts caused by noise oscillations of the signal is reduced).

The following image summarizes the fundamental principles on the base of the respiratory rate measurement that were previously described.

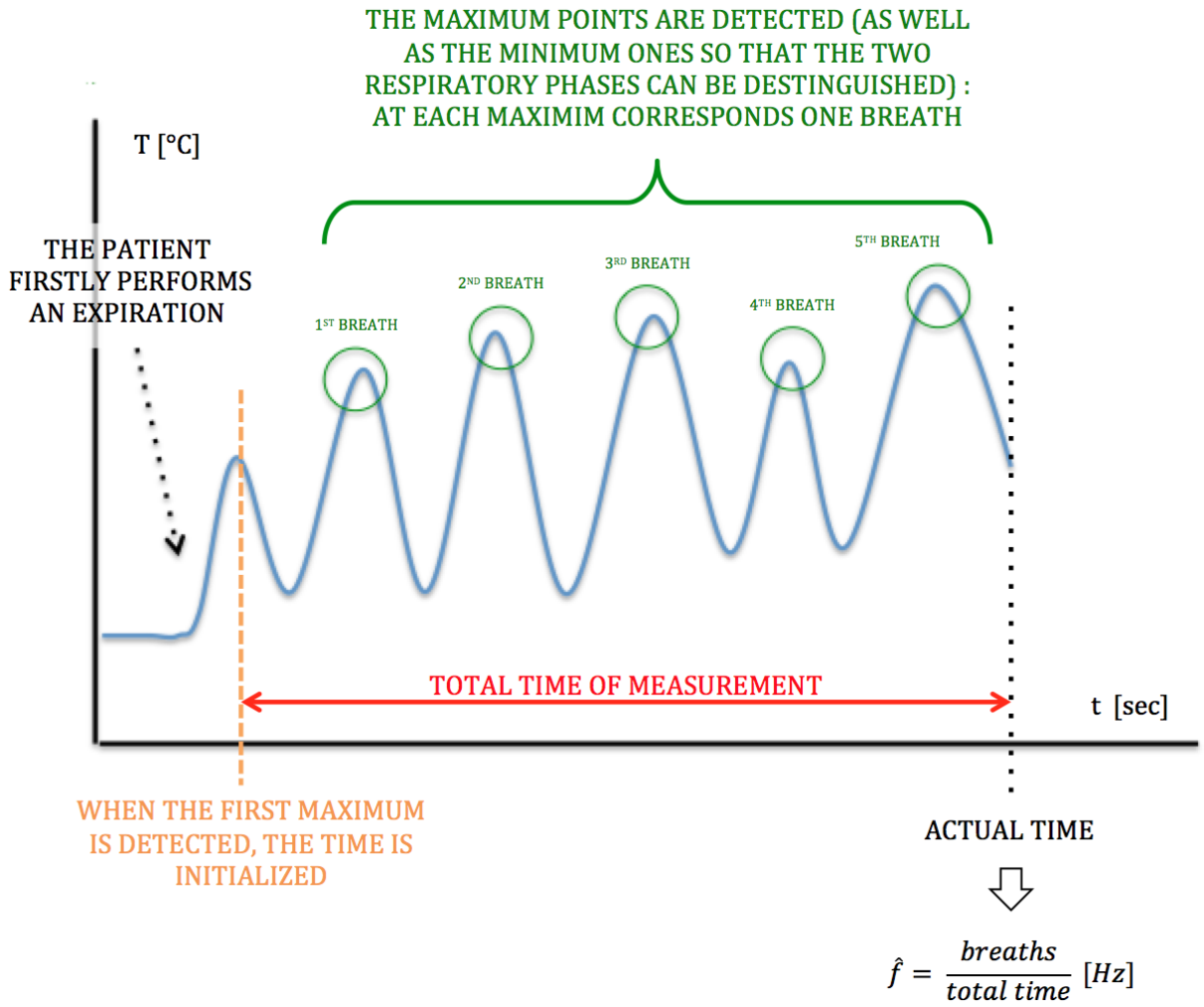



*Figure 4.6: Summary of the working principles of the algorithm*

The last step of the software infrastructure is related to the visualization of the respiratory phases. The alternation of expiration and inspiration is displayed thanks to a human sketch, whose lungs decrease and increase their volume synchronously with the respective respiratory phase, which is performed by the subject. Additionally, the temperature is real-time plotted together with the estimation of the respiratory rate.

# 5. RESULTS

As initially declared, the aim of the project is the respiratory rate measurement. The graphical interface shows the estimation of this value together with the real-time plotting of the temperature and the intuitive visualization of the respiratory phases as lung volume variations.

In this section, the results are discussed in terms of:

a) ***Visual Interface*** (accordance between temperature measurements and respiratory phases);
b) ***Respiratory rate accuracy*** (experimental evaluation of the estimation error).

### *a) Visual Interface*

We expect that when the temperature reaches a maximum, the expiration is at the end. Consequently, the lungs should reach the minimum volume. On the contrary, when a minimum in the temperature plot occurs, the volume of the lungs should be maximal because the inspiratory phase is at the end.

The following figures show that the actual behaviour respects the expectations in both the respiratory phases.

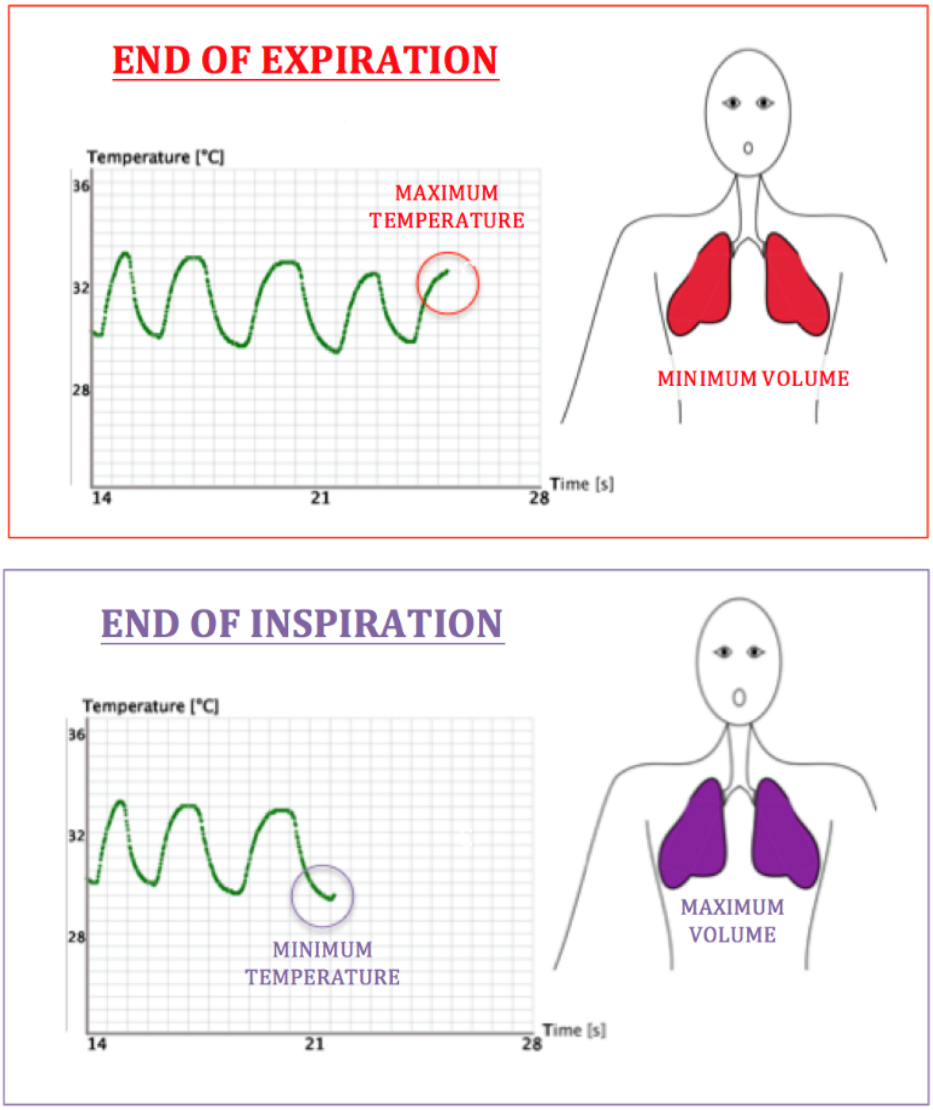



*Figure 5.1: Lung volume representation during the respiratory phases*

*b)* ***Respiratory rate accuracy***

In order to evaluate the performances of the developed measuring system, the following setup was established: the respiratory rate was estimated during 30 seconds of normal respiration of two different subjects (5 tests for each one of them). Then the experiment was repeated only varying its single-proof duration, extended to 1 minute.

The data are collected in the following tables, where $f_{real}$ is the frequency estimated by our device, while $f_{ideal}$ is the respiratory rate empirically measured counting the number of breaths in the considered time window.

| **Test over 30 sec (f expressed in breath/minute)** | | | |
|---|---|---|---|
| **SUBJECT 1** | | **SUBJECT 2** | |
| $f_{real}$ | $f_{ideal}$ | $f_{real}$ | $f_{ideal}$ |
| 22.2 | 24 | 15.5 | 16 |
| 20.1 | 22 | 14.0 | 14 |
| 20.9 | 22 | 14.2 | 16 |
| 18.6 | 20 | 15.8 | 16 |
| 18.0 | 20 | 16.0 | 18 |
| **Mean error sbj.1 =1.64** | | **Mean error sbj.2 = 0.9** | |
| **Overall mean error = 1.27** | | | |

| **Test over 60 sec (f expressed in breath/minute)** | | | |
|---|---|---|---|
| **SUBJECT 1** | | **SUBJECT 2** | |
| $f_{real}$ | $f_{ideal}$ | $f_{real}$ | $f_{ideal}$ |
| 20.3 | 20 | 15.8 | 16 |
| 22.2 | 22 | 16.8 | 17 |
| 18.6 | 19 | 16.8 | 17 |
| 17.7 | 18 | 17.1 | 17 |
| 20.5 | 21 | 18.8 | 19 |
| **Mean error sbj.1 =0.42** | | **Mean error sbj.2 = 0.18** | |
| **Overall mean error = 0.30** | | | |

These results highlight that increasing the time of measurement the mean estimation error decreases (that means the device is more accurate in determining the respiratory rate). This error is due to the fact that the estimation formula presents at the numerator the number of breaths (maximum of the temperature signal), which is updated only at the end of the respiration cycle; on the other hand the denominator is the total time of measurement which is continuously increasing. Therefore, after the detection of a maximum the frequency decreases until the next one. Another possible approach is a pointwise estimation of the respiratory rate (i.e. updating its value only in correspondence of a maximum) but it presents the limit of the non-continuity of the estimation.

In conclusion, if the test lasts 1 minute, the device is able to detect the respiratory frequency with an error inferior to one breath per minute. Therefore, if we approximate the estimated respiratory rate to the closest integer value, we find the exact frequency.