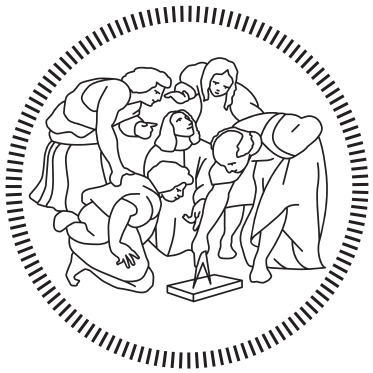
Politecnico di Milano
School of Industrial and Information Engineering
Master's Degree in Biomedical Engineering



Technologies for Sensors and Clinical Instrumentation Project:

Arduino Thermostat with NTC Thermistor Neonatal Incubator

Project by:

Mattia CAZZOLLA	Matr. 987083
Mimma CERRETO	Matr. 103090
Emma LACAGNINA	Matr. 996364
Giorgia MAIMONE	Matr. 991946
Ilaria MONACO	Matr. 997111

Academic Year 2021-2022

Contents

List of Figures

List of Tables

1 Introduction

1.1 Thermostat

A thermostat is a regulating device component which senses the temperature of a physical system and performs actions so that the system's temperature is maintained near a desired set-point.
Precise temperature measurement is essential in a wide range of applications in the medical field, such as thermal incubator, thermal regulation cells culture system, thermal regulation of biological tissue and ambient thermal regulation. [1, 2, 3, 4]
Thermostats usually operate as a "closed loop" control device, as it seeks to reduce the error between desired and measured temperatures.
The aim of this project was to develop a thermostat relying on Arduino hardware.

1.2 Thermal incubator

Of the possible applications, the main attention was given to the thermal incubator, which is one of the most common applications for thermal control in the NICU (Neonatal Intensive Care Unit).
At birth, the neonate rapidly cools in response to the relatively cold extra uterine environment. Thus, the neonatal temperature rapidly drops after birth [5, 6].
In order to survive and avoid developing serious health problems, premature newborns are kept at a constant body-like temperature (36-37°C) within an adequate range of humidity (80-90%). Studies have shown that, as a consequence of thermal monitoring and conditioning, newborns' survival rate is significantly increased ($\sim$77%). [1]
Most incubators provide heat by convection, using an air-blown electric heating system with the heating and humidification systems located under the incubator compartment. Thanks to a fan or a turbine air is taken from the outside and pass through the heating element and a temperature measuring device; then, it's humidified using a water tank before pushing it into the chamber where the newborn is.
The heating element is usually activated by an electric signal and the amount of heat that produces is usually proportional to the difference between the measured temperature and the reference value preset by the operator.

1.3 Approach to the project

Some compromises were needed to combine experimental validity of the set-up, acquired knowledge on Arduino and electric components. First of all, the dimensions of the device were reduced in order to heat the environment more easily and faster. Furthermore, a smaller size was needed to transport the prototype outside the laboratory site and to be more handily implemented.
Another issue was searching for components suitable for Arduino hardware, so the choice of heating elements was significantly restricted.
Furthermore, a humidity sensor was implemented to better supervise the ambient inside the neonatal incubator. However, due to hardware limitations, the humidity will only be measured and not actively controlled.

2 Hardware

2.1 Components

For the sake of achieving the desired application described in the previous section, the chosen Arduino set up foresees the following components:

ID	Component	Technical features	Quantity
a	Arduino UNO	ATmega328P I/O Voltage: 5 V Input voltage (nominal): 7-12 V DC Current per I/O Pin: 20 mA DC Current Vin Pin (max): 1 A	2
b	Termistor NTC	Uncalibrated	1
c	LCD Adafruit	Negative type RGB 16x2	1
d	Heating Pad Adafruit	Operating Voltage: 5V DC Operating Current: 750mA (6.5 Ω) Dimensions: 5 x 10 cm	1
e	DHT11	Operating Voltage: 5V Operating Current: 0.5 mA - 2.5 mA Humidity Range: 20% - 90% Humidity Resolution: 16-bit Accuracy: ±1%	1
f	Green LED	max current: 20 mA	1
g	Red LED	max current: 20 mA	1
h	N Mosfet D4184	Operating Voltage: 5 V max current: 40 A V	1
i	Button	-	1
j	Resistor	Resistance: $10k\,\Omega$	2
k	Resistor	Resistance: $220\,\Omega$	2
l	Transparent Plastic Box	Dimensions: 26 x 19 x 14 cm	1
m	Breadboard	Dimensions: 16.5 x 5.6 cm	2
n	AC adapter	Supplied Voltage: 12 V Supplied Current (max): 1 A	1

Table 1: Components

2.2 Connection Schematics

The schematics can be physically and logically divided into two main parts, one for each Arduino.
The first Arduino is connected only to the Mosfet (which acts as a switch) and to the heating pad and function as a current generator. This connection happens with the Vin pin (directly connected to the power supply through the AC adapter) since a high amount of current (650 mA) is necessary to properly heat the pad.
The second Arduino is mainly used for the sensoring portion of the hardware and as a control unit. Indeed it reads the values from the thermistor and from the DHT11 sensor, it handles the activation and deactivation of the heating pad both with firmware and a safety physical button. Furthermore, it gives visual information about the system through a green LED (system on) or a red LED (alarm: temperature outside safety range) and it displays the current temperature, humidity, thermostat threshold with an LCD display.
In addition, the second Arduino was introduced to overcome the problems that were detected during experiments, regarding the distribution of the current, which was not enough to power all the components.
Focusing on the on/off button, its purpose is to give an additional safety measure to protect the newborn in case of malfunctioning of the heating pad's control system. When pressed, the button allows to turn off the system and it needs to be pressed again to return to the *on* state. By default, the system starts at state *off*.
All the components are placed in series with an appropriate resistor in order to stabilize the incoming or out-coming signal and to limit the current.

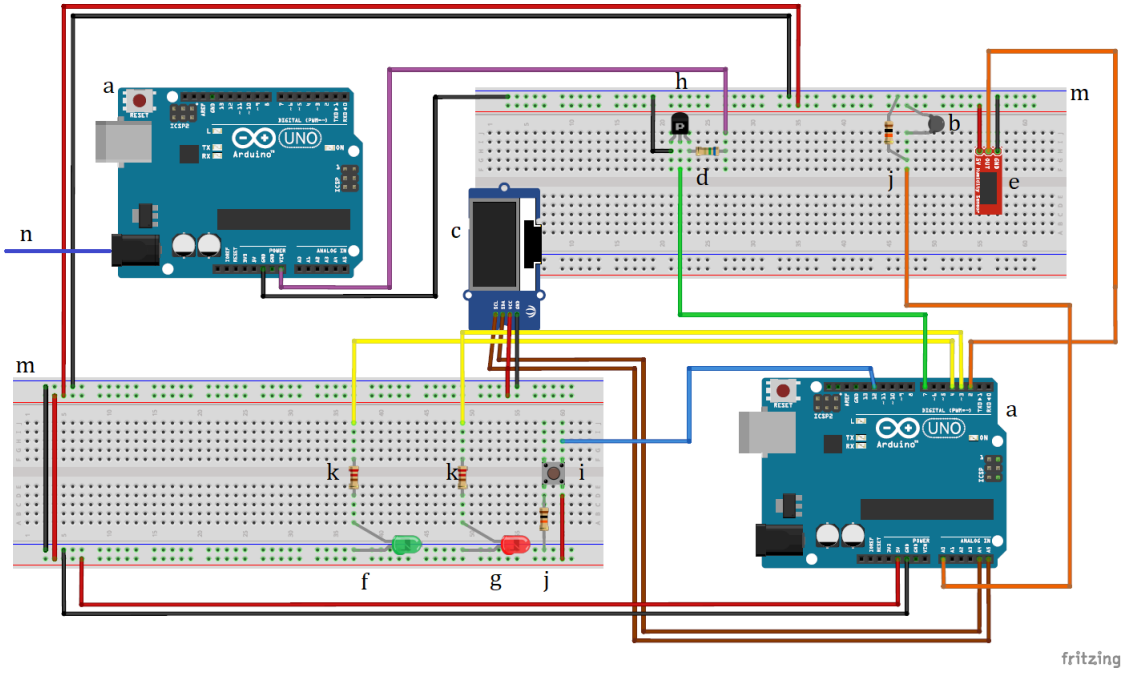


Figure 1: Breadboard schematic

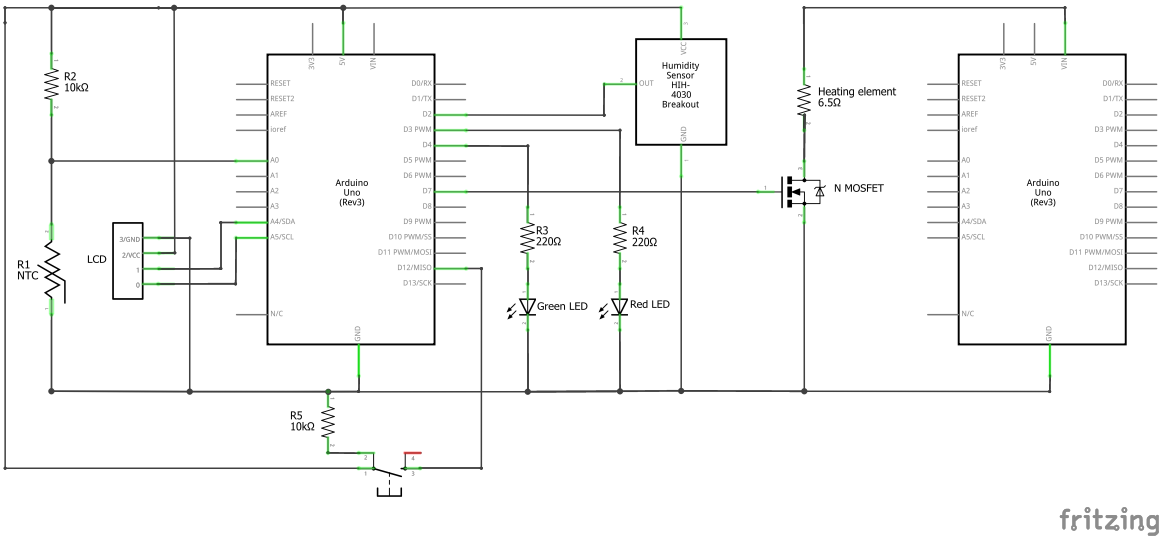


Figure 2: Circuit schematic

Besides thermistor, the most important component for this type of application is the heating element, here represented by the heating pad. This device can be schematized as a resistor that dissipates heat (Joule effects) when a current passes through it; for this reason, and since in Fritzing® there is no symbol for heating pads, in the circuit schematic it has been assimilated to a resistor of 6.5Ω in series with the mosfet.
The heating pad's dynamic is shown in Figure 3 and we can notice being quite slow. However since our application only needs to reach 36-37°C the waiting time will not be long.

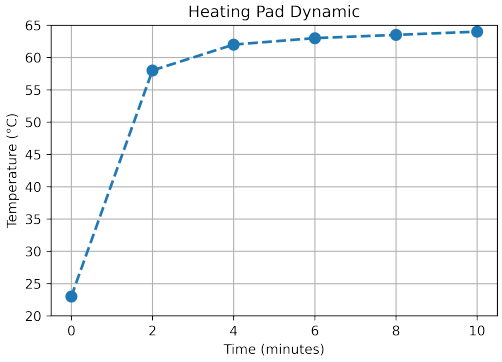


Figure 3: Heating pad dynamic

2.3 Sensor Conditioning

In this circuit, the sensor conditioning consists of a voltage divider between the thermistor and a 10k Ω resistor. The divider is necessary in order to detect any variation in the voltage across the thermistor since given a constant current, this variation can be related only to a variation of resistance and so of temperature.

The resistor is chosen of 10k Ω because from some tests the resistance of the temperature sensor shows a similar range of values at room temperature.

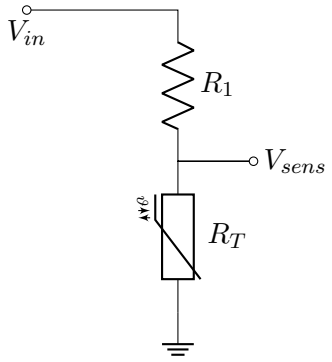


Figure 4: Voltage Divider

Thanks to the voltage divider the voltage drop across the thermistor is obtained:

$$V_{sens} = \frac{R_T}{R_1 + R_T} V_{in} \tag{1}$$

where if solved for R_T

$$R_T = \frac{R_1}{V_{sens}}(V_{in} - V_{sens}) \tag{2}$$

The thermistor characteristic is defined by the following equation

$$R_T(T) = R_{T0}\, e^{\beta\,(\frac{1}{T} - \frac{1}{T_0})} \tag{3}$$

with $T_0 = 298K$
When solved for T in Celsius

$$T = \frac{\beta}{\log(\frac{R_T}{R_0}) + \frac{\beta}{T_0}} - 273.15 \tag{4}$$

2.4 Sensor Calibration

The thermistor available was not already calibrated. So part of the project consisted of calibrating it to identify the characteristic curve with its parameters.
The following procedure is used to calibrate the system:

1. The thermistor is placed on a breadboard in series with a resistor. This conditioning circuit is modelled as a voltage divider as described in the section above.

2. The thermistor is immersed in a glass of water which temperature is obtained by using an external digital thermometer.

3. Once immersed the R_T is computed using Equation 2 and the result is shown through Arduino's Serial Monitor

4. The same procedure is repeated another two times after heating the water in the glass to different temperatures

5. The estimated Steinhart curve is obtained using [8]

With regard to the DHT11 sensor, there is no need for calibration since for this type of sensor the calibration coefficients are already defined during production, stored in an internal OTP memory and used automatically by its Arduino library during the acquisition process.

3 Firmware

In this application the firmware has two main purposes:

- read (and convert) the temperature and humidity from the sensors and write the values on the serial port
- switching on the heating element if the temperature is lower than a threshold (read from the serial port) and switching it off when the temperature is higher than the threshold or when the safety button is pressed

The firmware code is written in C and is uploaded on the Arduino.

3.1 Definition of sensor's characteristics

```
15 #define RT0 10000
16 #define B 3600
17 #define T0 298
18 #define R1 10000
```

In this portion of the code are defined the characteristics of the sensor based on the calibration results and the components used in the conditioning circuit.
For more details refer to the Results Section.

3.2 Switching on/off the heating element when safety button is pressed

```
48 // turn the heating system ON
49 if(digitalRead(ON) == HIGH && state == 0) {
50   state = 1;
51   delay(500);
52 }
53 // turn the heating system OFF
54 if(digitalRead(ON) == HIGH && state == 1) {
55   state = 0;
56   delay(500);
57 }
```

The hardware configuration includes also a button to control the activation of the heating element according to the necessary application. As already mentioned in the Hardware Section, the button is used to perform a switching control between the state on and off.
Furthermore, to debounce the signal from Arduino a delay of 500 ms is added.

3.3 Temperature and humidity acquisition

```
59 // temperature acquisition
60 float VSens = analogRead(SENSOR_PIN) / 1023.0 * 5;
61 float RSens = (R1 / VSens) * (5-VSens);
62 float T = B / (log(RSens / RT0) + (B / T0)) - 273.15;
63 // humidity acquisition
64 int H = dht.readHumidity();
```

For the acquisition of temperature data, the available space for reading incoming values from the Arduino analog pin is 10 bits. This means that it maps input voltages between 0 and the operating voltage (5 V) into integer values between 0 and 1023. So line 60 is necessary to convert

the value in the Voltage measure.
As shown in Equation 2, the resistance of the thermistor changes together with the voltage drop and it's computed as reported in line 61.
The same procedure is applied to the temperature registered by the thermistor, which is converted from Kelvin to Celsius as shown in Equation 4 and in line 62.
To measure humidity, the hardware is equipped with a DHT11 sensor and in Arduino firmware the library *DHT.h* is included to read the values through the function *dht.readHumidity()* as shown in line 64.

3.4 Sending data through serial port

```
65 // Print data in serial port
66 Serial.print("T");
67 Serial.println(T,1);
68 delay(500);
69 Serial.print("H");
70 Serial.println(H,1);
71 delay(500);
```

The values of temperature and humidity are printed by Arduino in the serial port with a cadence of one second for each type. Each value is sent as a string with the first character, either T or H, behaving as a label.

3.5 Threshold definition

```
89 // threshold acquisition
90 if (Serial.available() > 0) {
91     myString = Serial.readString();
92     threshold = myString.toFloat();
93 }
```

Since the desired threshold is set through the graphical interface (Section Software 4), the value has to be sent to the Arduino board through the Serial Port. When serial communication is enabled by data in the Serial Port, Arduino reads them as a string and then converts them into a float type.

3.6 Control of the heating element

```
95 // management of the heating element
96 if (state == 1) {
97     digitalWrite(LEDON, HIGH);
98     if (T < threshold + toll)
99         analogWrite(PAD, 250);
100    else
101        analogWrite(PAD, 0);
102 }
```

The control of the heating system consists in checking if the current temperature is lower or higher than the threshold set ± a tolerance value. The tolerance is necessary to avoid the heating element from continuously switching on and off due to small variations (like noise) around the threshold. In our case, the tolerance is stated to be 0.5 °C.

4 Software

Regarding the software, we decided to develop a GUI composed of different pages which allow the user to

- check the value of the temperature with its trend over time,
- check the value of the humidity
- set new thresholds for the thermostat
- produce an unpleasant noise aimed to attract attention to the incubator in case of malfunctioning
- customize the platform through the settings (e.g temperature scale, alarm volume)

The application is written in Processing (Java).

4.1 Code

4.1.1 Switch between pages

```
54 void draw()
55 {
56   switch(page)
57   {
58     case 0:
59       settingsPage();
60       break;
61     case 1:
62       homePage();
63       break;
64     case 2:
65       valuesPage();
66       break;
67     case 3:
68       trendPage();
69       break;
70   }
71 }
```

The variable *page* (type integer) define the page the user is currently in. When the buttons placed to change the page are pressed, the variable changes value allowing the render of a new page in the following cycle of the function *draw()*.

4.1.2 Reading data

```
564 void readData()
565 {
566   try {
567     if (myPort.available()>0)
568     {
569         val = myPort.readStringUntil('\n');
570         if (val.charAt(0) == 'T')
571         {
572           if (!fahrenheit) // Celsius scale
573             temp = float(val.substring(1));
574           else // Fahrenheit scale
```

```
            temp = C_to_F(float(val.substring(1)));
        }
        else if (val.charAt(0) == 'H')
          humidity = int(float(val.substring(1)));
    }
  }
  catch (Exception e){}
}
```

The data Arduino send is read by the application through the *readData()* function which is called at the beginning of the function *valuesPages()* and *trendPages()*. The function looks if there are available data in the serial port, if so it identifies the type of data, temperature or humidity, thanks to the label and saves the values in the global variables *temp* and *humidity*. If the user has set the Fahrenheit scale from the settings, it converts the value before saving it.

4.1.3 Set threshold

```
public void Set() {
  check = cp5.get(Textfield.class, "textValue").getText();
  if (!fahrenheit) // Celsius scale
  {
    if (isValid_C(check))
    {
      thr_C = float(cp5.get(Textfield.class, "textValue").getText());
      myPort.write(str(thr_C));
      error = false;
    }
    else
      error = true;
  }
  else // Fahrenheit scale
  {
    if (isValid_F(check))
    {
      thr_F = float(cp5.get(Textfield.class, "textValue").getText());
      myPort.write(str(F_to_C(thr_C)));
      error = false;
    }
    else
      error = true;
  }
  cp5.get(Textfield.class, "textValue").clear();
}
```

On the Trend Page, the user has the possibility to change the threshold of the thermostat. To do so he/she needs to write the value in the text-field and press the button *Set*. When the button is pressed the function Set() is called which gets the value written in the text-field and, after saving it in a global variable, it writes it in the serial port to update the Arduino firmware. Since the Arduino works only with the Celsius scale, if the threshold inserted is in Fahrenheit it needs to be converted.
To make the platform robust from the user input, a control on the value inserted in the textfield had to be implemented. This control is executed by the functions *isValid_C()* or *isValid_F()* which are based on regular expressions

```
boolean isValid_C(String str)
```

```
{
  if (str.matches("[2][5-9](\\.\\d+)?|[3][0-9](\\.\\d+)?|[4][0](\\.[0])?"))
    return true;
  else
    return false;
}
boolean isValid_F(String str)
{
  if (str.matches("[7][5-9](\\.\\d+)?|[8-9][0-9](\\.\\d+)?|[1][0][0-5](\\.[0])?"
    ))
    return true;
  else
    return false;
}
```

In case of the Celsius scale, it allows numerical text between 25 and 40 (with decimal) while the Fahrenheit scale only allows numerical text from 75 to 105 (also with decimal).
If the text is not valid the global variable *error* will be set *true* and will trigger an error message.

4.1.4 Alarm sound

```
void createAlarmSound()
{
  int resolution = 1000;
  float[] sinewave = new float[resolution];
  for (int i = 0; i < resolution; i++)
  {
    sinewave[i] = sin(TWO_PI*i/resolution);
  }
  sample = new AudioSample(this, sinewave, 1200 * resolution);
}

void activateAlarm()
{
  sample.amp(audioValue);
  sample.loop();
}

void stopAlarm()
{
  sample.stop();
}
```

As previously stated the software is able to generate an alarm sound aimed to attract attention to the incubator in case of malfunctioning.
The function *createAlarmSound()* handle the definition of the type of sound, in this case it's a simple sinusoidal wave.
The function *activateAlarm()* define the sound level thanks to the variable *audioValue* (type float from 0 to 1) and start playing the sound. The *audioValue* value is 0.4 by default but can be changed by the user in the setting page.
Lastly the *stopAlarm()* function as the name suggest stops the software from playing the alarm sound.

```
  if (!fahrenheit) // Celsius scale
  {
```

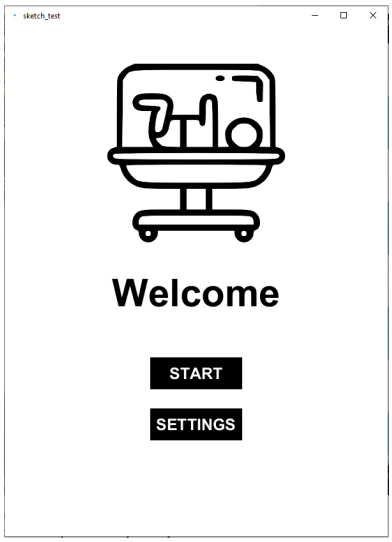
```
189     if (inThrRange(temp, 'C'))
190       stopAlarm();
191     else
192       activateAlarm();
193   }
194   else // Fahrenheit scale
195   {
196     if (inThrRange(temp, 'F'))
197       stopAlarm();
198     else
199       activateAlarm();
200   }
```

The sound need to play only if the temperature is outside a define range from the threshold. In the code of the functions *trendPage()* and *valuePage()* this control is implemented with the *inTheRange()* function working as follow

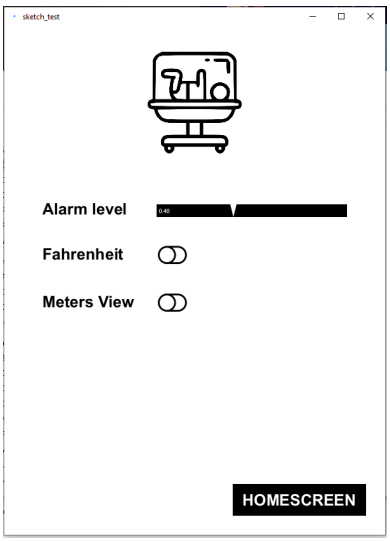
```
818 boolean inThrRange(float temp, char t)
819 {
820   if (t == 'C') // Celsius scale
821   {
822       if ((temp < thr_C - toll_alarm) || (temp > thr_C + toll_alarm))
823         return false;
824       else
825         return true;
826   }
827   else // Fahrenheit scale
828   {
829       if ((temp < C_to_F(F_to_C(thr_F)-toll_alarm)) || (temp > C_to_F(F_to_C(
    thr_F)-toll_alarm)))
830         return false;
831       else
832         return true;
833   }
834 }
```

By default the *toll_alarm* variable is set to 2°C.

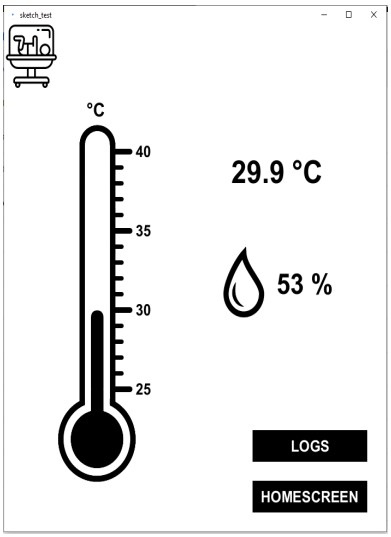
4.2 GUI



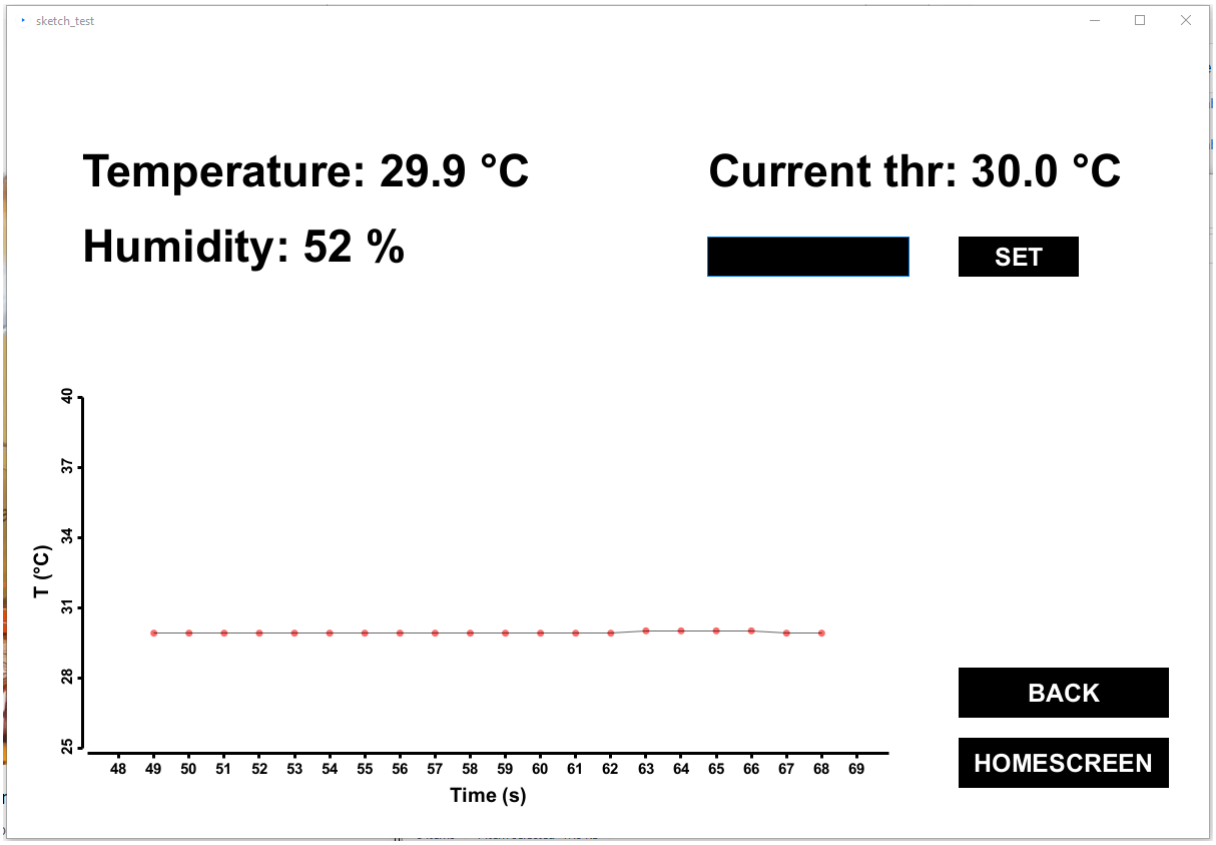
(a) Home Page



(b) Settings Page



(c) Main Page



(d) Trend Page

Figure 5: GUI pages

5 Results

5.1 Sensor Calibration

As previously noted in Section 2 Hardware, calibration of the NTC thermistor is fundamental to the correct functioning of the device.
The results obtained are summarized in the following table

R (Ω)	**T (°C)**
10848	23
3319	55
1769	75

Table 2: Calibration results

Using the values in Table 2 the parameters of the characteristic curve resulted in

$$R_0(25°\text{C}) = 9999.28\,\Omega \qquad \text{and} \qquad \beta = 3596.68\,K$$

that we decided to approximate to

$$R_0(25°\text{C}) = 10000\,\Omega \qquad \text{and} \qquad \beta = 3600\,K$$

With the approximated results the curve equation became:

$$R_T(T) = 10000\,e^{3600\,(\frac{1}{T}-\frac{1}{298})} \tag{5}$$

which we can see in the plot below with the calibration point from Table 2

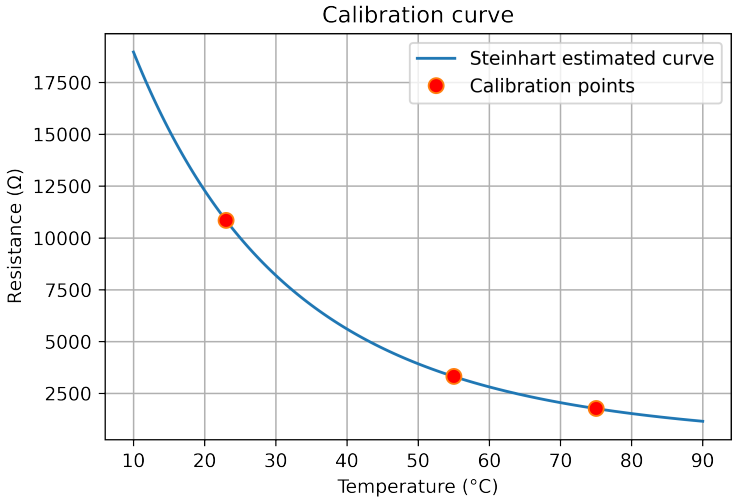
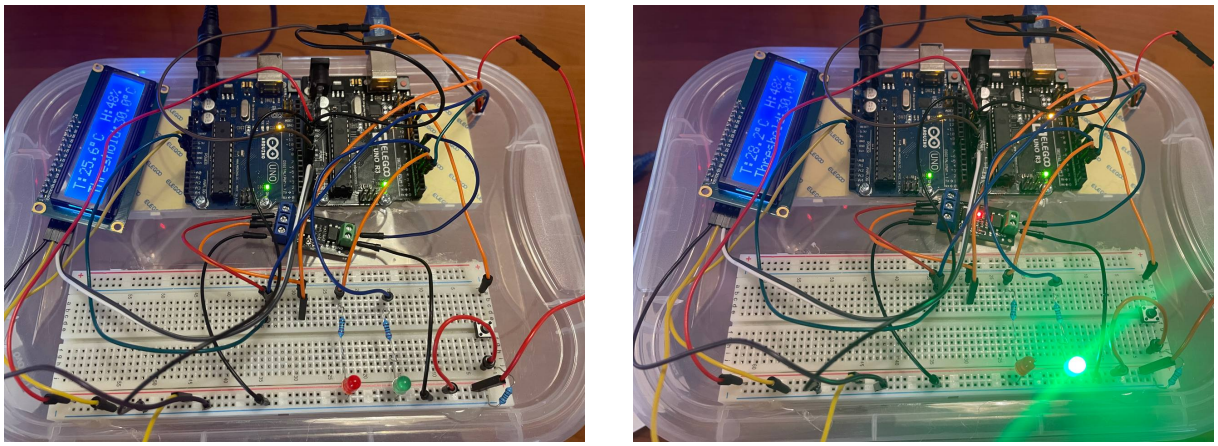


Figure 6: Calibration curve of NTC thermistor

5.2 Application Results

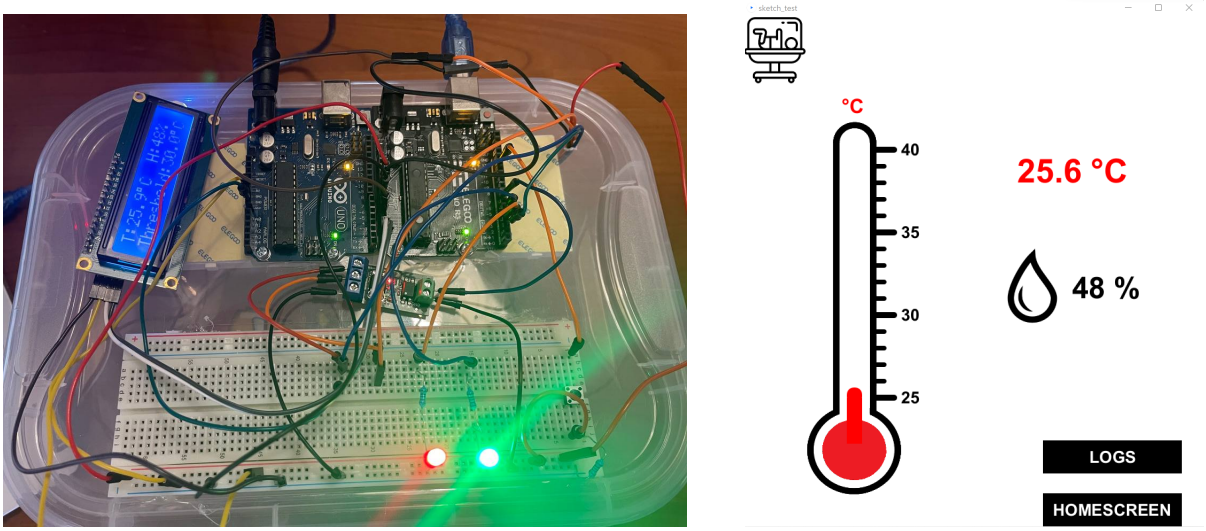
A demonstration of the sensor's application is given to the reader. Here, in Fig.7.a it is reported the hardware system when is completely off. When the operator activates it, the green LED as well, is turned on (Fig.7.b) For example the threshold is set to 30°C.



(a) System OFF (b) System ON within the range

Figure 7: System OFF and ON

As soon as the temperature inside the little environment it's below (in this case, but it could be also above) the set threshold and outside the acceptable range (Fig.8), the heating pad starts to work as previously described and the temperature inside the physical surroundings increases until the ideal value is achieved. Thus, the heating element will switch off.



(a) System ON outside the range (b) GUI when outside the range

Figure 8: System ON outside range

5.3 Conclusion

Starting from an NTC thermostat, the aim of the project is extended to deeper use. Since the invention of the neonatal thermal incubator, this tool has had a positive impact on medicine by decreasing child-mortality throughout the years. The focus of this project is to develop a practical device, easy to use, with a security button and an alarm system. New accessibility tools provide more advantages to those who need them.

To see the entire project code and some media (video demo, screenshots and powerpoint presentation) refer to the following GitHub repository:

`https://github.com/MattiaCazzolla/arduino-thermostat`

6 References

[1] Burunkaya, M., Yucel, M. *Measurement and Control of an Incubator Temperature by Using Conventional Methods and Fiber Bragg Grating (FBG) Based Temperature Sensors.* J Med Syst 44, 178 (2020). https://doi.org/10.1007/s10916-020-01650-2

[2] Reesink, H., Hanfland, P., Hertfelder, H.-J., Scharf, R.E., Högman, C.F., Hoppe, P., Moroff, G., Friedman, L., Masse, M., Walsh, T., Ala, F. and Pietersz, R.N.I. (1993), *What Is the Optimal Storage Temperature for Whole Blood Prior to Preparation of Blood Components.* Vox Sanguinis, 65: 320-327. https://doi.org/10.1111/j.1423-0410.1993.tb02174.x

[3] Imashiro C, Takeshita H, Morikura T, Miyata S, Takemura K, Komotori J. *Development of accurate temperature regulation culture system with metallic culture vessel demonstrates different thermal cytotoxicity in cancer and normal cells.* Sci Rep. 2021 Nov 2;11(1):21466. doi: 10.1038/s41598-021-00908-0. PMID: 34728686; PMCID: PMC8563756.

[4] Khan, A. Z., Utheim, T. P., Jackson, C. J., Tønseth, K. A., & Eidet, J. R. (2021). *Concise Review: Considering Optimal Temperature for Short-Term Storage of Epithelial Cells.* Frontiers in medicine, 8, 686774. https://doi.org/10.3389/fmed.2021.686774

[5] Frischer R, Penhaker M, Krejcar O, Kacerovsky M, Selamat A. *Precise Temperature Measurement for Increasing the Survival of Newborn Babies in Incubator Environments.* Sensors. 2014; 14(12):23563-23580. https://doi.org/10.3390/s141223563

[6] Asakura H. *Fetal and neonatal thermoregulation.* J Nippon Med Sch. 2004 Dec;71(6):360-70. doi: 10.1272/jnms.71.360. PMID: 15673956.

[7] `https://docs.arduino.cc/hardware/uno-rev3`

[8] `https://www.thinksrs.com/downloads/programs/therm%20calc/ntccalibrator/ntccalculator.html`