Digital Thermometer with Thermistor and Arduino[1]

Introduction

What is Arduino?

Arduino (www.arduino.cc) is an open-source prototyping platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language, and the Arduino Software (IDE)[2], which is already installed on all the computers in the physics lab.

There are many types of Arduino boards. We will be using the "UNO". The UNO has 14 digital input/output pins (which can either provide or read voltages), 6 analog inputs (for reading voltages). The DC Current per I/O Pin is $20mA$, while DC Current for 3.3V Pin (which is often used as a power source for your experiment) is $50mA$. The "GND" pins represent the ground. Details are listed at www.arduino.cc/en/Main/ArduinoBoardUno.

The custom program that you upload into the Arduino board is commonly referred to as a "sketch".

What is a Thermistor?

A thermistor stands for thermal resistor, whose resistance changes rapidly with temperature. By monitoring the resistance of the thermistor, the temperature can be deduced. The thermistor we will use is of the NTC (negative temperature coefficient) type, meaning its resistance decreases with rising temperature.

Measuring Resistance

An Arduino board measures only voltage but not resistance. In other to determine the resistance of the thermistor, therefore one has to use a $R = 10k\Omega$ resistor in series with the thermistor to build a voltage divider (see setup below). By measuring the voltage across the thermistor one can then deduce the resistance.

$$V_{thermistor} = \frac{R_{thermistor}}{R_{thermistor} + R} V_{cc} \Rightarrow 1 + \frac{R}{R_{thermistor}} = \frac{V_{cc}}{V_{thermistor}} \Rightarrow R_{thermistor} = R/(\frac{V_{cc}}{V_{thermistor}} - 1)$$

[1] This experiment and the codes are based on a tutorial on Adafruit (https://learn.adafruit.com/thermistor/using-a-thermistor), many other electronic components and tutorials are available on the site if you are interested in exploring more.

[2] IDE stands for "integrated development environment", it is the software you use to program the Arduino board.

In our experiment below, we will use the “3V3” pin to provide the voltage for the voltage divider, which means $V_{cc} = 3.3V$.
A further complication is that the Arduino board gives the voltage measurement as an integer (ADC value) from 0 to 1023, scaled by the reference voltage V_{cc}. In other words, the voltage of the thermistor read by an Arduino input pin should be computed from the ADC value with $V_{thermistor} = \frac{ADC}{1023} V_{cc}$. Therefore, in terms of the ADC value, we have $R_{thermistor} = R/(\frac{1023}{ADC} - 1)$.

AREF (Analog Reference) Pin

While the “5V” pin is supposed to provide a 5V voltage output, the actual voltage could be “noisy” because it simply passes whatever voltage the Arduino board receives from the USB cable without regulation. A more stable voltage output can be obtained by using the “3V3” pin, which gives a voltage output of $3.3V$. This can be further stabilized by connecting the “3V3” pin to the “AREF” pin, which tells the Arduino board to set the reference level for analogy reading at $3.3V$, so an ADC value of 1023 corresponds to a voltage reading of $3.3V$.

Equipment

- Arduino board
- USB cable
- Breadboard
- Jumper wires
- Multimeter
- Thermistor
- $10k\Omega$ resistor
- 220Ω resistor
- LED (at least two different colors)

Caution

The thermistor’s plastic coating cannot withstand high temperature (even though the resistor inside can), so keep the temperature you measure to below $80^{\circ}C$ just to be safe.

Testing the Arduino Board

1. Connect the Arduino board to the computer with the USB cable (with no wires connected).
2. Go to the “File” menu, then select “Examples->01 Basics->Blink”. This will open the example sketch, Blink.
3. Click the “Upload” button (looks like a right-pointing arrow). If the upload succeeds, you should see the built-in LED (below pin 13) on the board blinking.

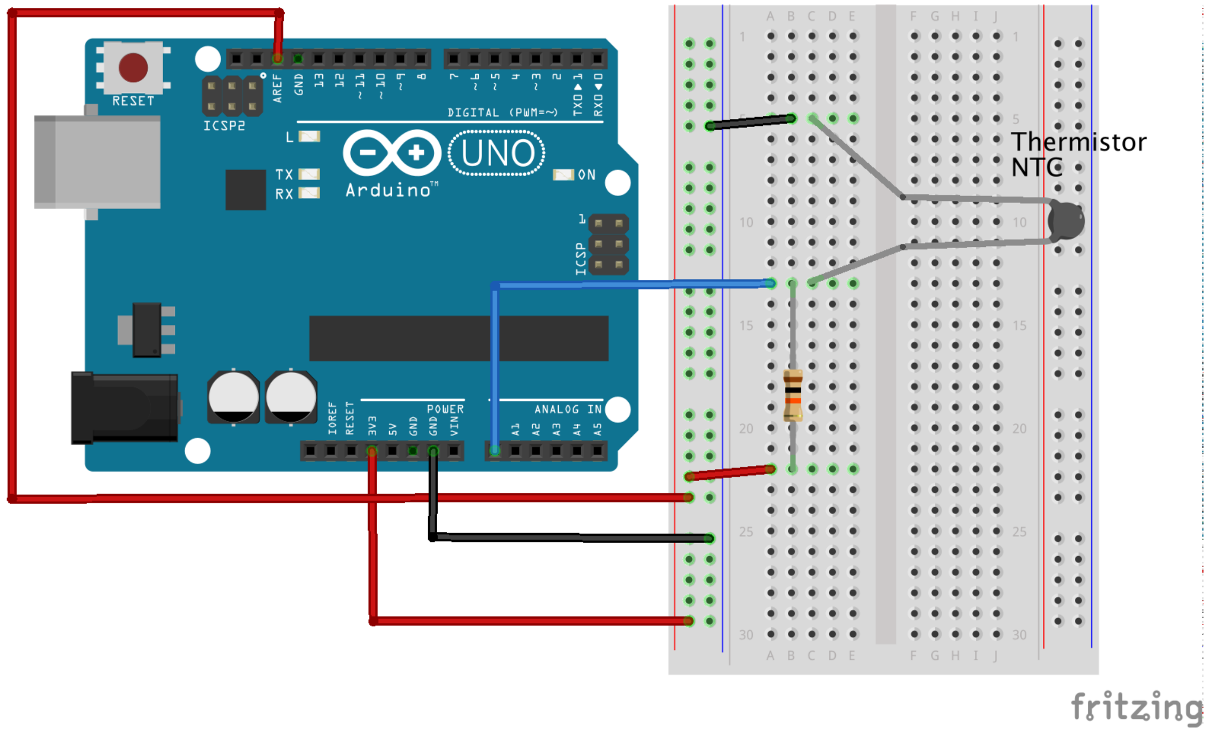
4. If it does not work, it is usually because the computer does not know which USB part is connected to the Arduino. To fix it, click on “Tools->Serial Port”, and try the other connections listed. If too many ports are listed, you can unplug the USB cable and see which port disappeared from the list, which would be the port used by the Arduino. Choose the port after replugging in the USB again, and try to upload the sketch again.
5. The current sketch has the LED turns on for 1 second and off for 1 second. Study the sketch and modify it so that the LED turns on for 3 seconds and off for 0.5 second. The Arduino programing language is very easy to learn and understand, all you need is common sense.

Testing the Thermistor

- Connect the multimeter to the thermistor to monitor its resistance.
- Hold the thermistor between your fingers, you should see the resistance changes as it is warmed by your body heat.

Setup

Figure 1: Connecting the thermistor with the breadboard using the $10k\Omega$ resistor.

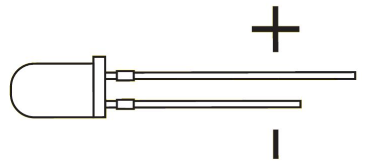


Procedure A

1. Connect the circuit in Figure 1.
2. Connect the Arduino board to the computer and open the Arduino IDE on the computer. Make sure the correct serial port is chosen (see the earlier section on Testing the Arduino Board).
3. Type in the Program 1 below. Typing instead of just copy and paste will help you understand the codes better and are often recommended to programming beginners.
4. Upload the sketch by clicking the button with the right-pointing arrow.
5. Click on the magnifying glass button on the top right hand corner of the IDE, you should see the temperature being updated once every second.
6. Test your thermometer on some hot and cold water over a range of temperature (do not go above $80^\circ C$!). Compare your Arduino measurement with a regular thermometer. Make a table to compare the two.

Procedure B

Figure 2: The long leg of the LED is the anode and should be connected to the positive terminal, while the short leg (the cathode) should go toward the ground (GND).



1. Get two 220Ω resistors and two LED lights (red and yellow, for example). The resistor should always be connected in series with the LED to protect it from too large a current.
2. Study the “Blink” example on https://www.arduino.cc/en/Tutorial/Blink and see if you can modify the codes so that it lights up the red LED when the temperature is above $60^\circ C$ and the yellow LED when the temperature is below $15^\circ C$. You will also need to use an “if...else” statement, which is explained at the reference page at https://www.arduino.cc/en/Reference/HomePage.
3. Test your new “temperature alert system” :)

Procedure C (Optional)

1. Study the example “LED Bar Graph” on https://www.arduino.cc/en/Tutorial/BarGraph.
2. See if you could modify your circuit and your codes so that the LED bar graph displays the temperature.

Codes

Program 1:

```
//To see the temperature, you will need to click on the "Serial
Monitor" button on the top right hand corner (which looks like a
magnifying glass).

//Define the pin that reads the voltage between the thermistor and
the resistor.
#define PIN_THERMISTOR A0

//Define a variable to hold the value of the resistance of our
external resistor.
const float RESISTANCE_EXTERNAL=10000;

//The “setup” function will run only once in the beginning of the
sketch.
void setup(void){
  //This tells the Arduino board to be ready to communicate with the
computer.
  Serial.begin(9600);
  //Set the reference voltage level to that of the AREF pin, which is
connected to the 3.3V pin.
  analogReference(EXTERNAL);
}

//The “loop” function will run repeatedly forever until power is
disconnected from the Arduino board.
void loop(void){
  //Declare the variables we want to use.
  float ADC_value;
  float resistance_thermistor;
  float temperature;

  //This reads the voltage from the thermistor pin, the values is in
the range of 0 to 1023
  ADC_value = analogRead(PIN_THERMISTOR);

  //Compute the resistance of the thermistor using the equation
derived in the Introduction section.
```

```
  resistance_thermistor = RESISTANCE_EXTERNAL/((1023/ADC_value) - 1);

  temperature = temperature_from_resistance(resistance_thermistor);

  //Now display the results:
  Serial.print("ADC value: ");
  Serial.println(ADC_value);

  Serial.print("Thermistor resistance in Ohms: ");
  Serial.println(resistance_thermistor);

  Serial.print("Temperature in C: ");
  Serial.println(temperature);

  Serial.println();  //Print a blank line.

  //Wait 1 second (1000ms) before repeating the loop again.
  delay(1000);
}


float temperature_from_resistance(float resistance){
  const float T_0 = 25+273.15;
  const float B_0 = 3950;
  const float R_0 = 10000;

  //Use the Steinhart-Hart equation to find the temperature from the
resistance.
  return 1/((1/T_0) +(1/B_0)*log(resistance/R_0))-273.15;
}
```