Informatics Enlightened

Station 4 – Color Thermometer

Making Temperature Glow

You have picked a project of the category "Let's make the world a better place". Why does this project make the world a better place? How many people have burned their fingers, because the stovetop was still hot? Or made a face in disgust, because their tea was already cold? It all happens, because of the fact that, humans cannot see temperature. Now, you can change that!

[1]

[2]

Your task is to construct a thermometer, which uses different colors to indicate the current temperature.

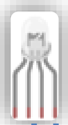
The Construction of the Circuit

For your circuit, you will need the following components:

- 3 220 Ω resistors (figure 3)
- 1 temperature sensor LM35 (figure 4)
- 1 RGB LED (figure 5)
- 4 yellow, 1 blue and 1 red wire (long)
- 1 short blue and 1 short red wire

[3]

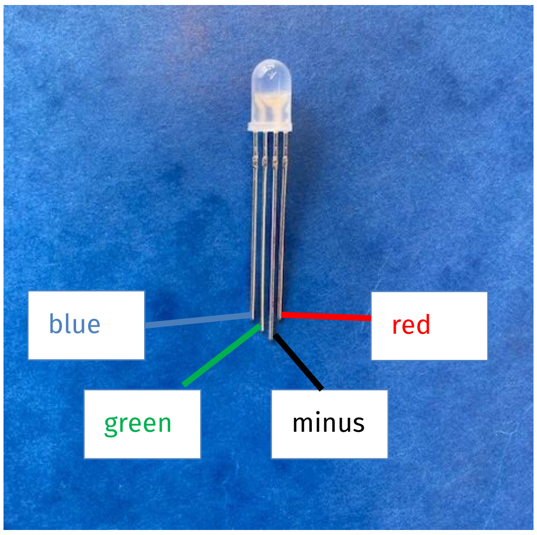
[4]


[5]

Color Mixture

RGB LEDs

RGB LEDs may look like normal LEDs (which you know from the introduction), but, in fact, they contain three different LEDs: one red, one green and one blue LED. This is why they are called RGB LED. Of course, this raises the question, why they do not have six legs, but only four. This is because the developers have already connected the negative poles in the case. The longest leg needs to be connected to the minus pole. If you place the LED in front of you in such a way, that the longest leg is in the third position of the left, you can see in the figure, which leg belongs to the positive pole of which color. You only need these three colors to represent all visible colors. But more on that later.

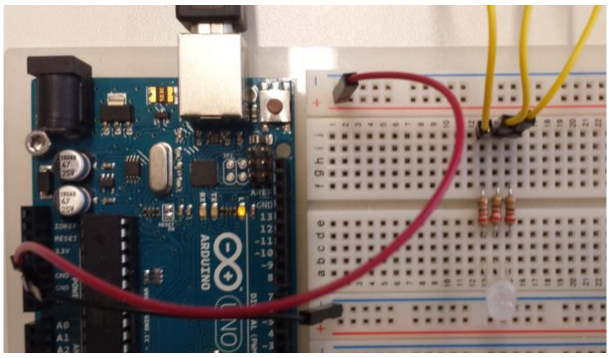


[6]

1. First, connect the plus and minus bar of the breadboard with 5V and GND on the Arduino, just like you did in the introduction.
2. Now, plug the RGB LED into the breadboard, so that the minus pin is placed in the minus row of the breadboard. The other pins are placed in their own row of the breadboard. Afterwards, each of these rows is connected to the other side of the board with the help of a resistor. There, you can already plug in three yellow wires, but the upper end hangs in the air for now.
3. After connecting the Arduino with the computer, you can get started: Connect each of these wires with the plus bar, and note your observations in the table.

[**Hint:** If you have problems to built the circuit, figure 7 shows the almost completed circuit.]

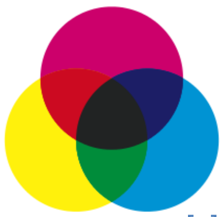
Connected Yellow Wire	**Color of the LED**
Left	
Middle	
Right	
Left and Middle	
Left and Right	
Middle and Right	
Left, Middle, and Right	


[7]

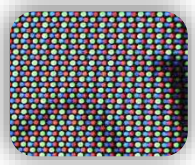
Have you filled in the table? Great! But why do these results have nothing in common with what you have learned in arts class?

Additive Color Mixture

From arts class you know the good old paint box. If you mix the colors blue and yellow, you get green. This works, because white light contains all colors. The colors from the paint box filter certain portions out of the light, so that many of these portions are eliminated, if you mix them. If you mix all colors, every portion is filtered, and the result is black. This is called the subtractive color model.


[8]

When you use a RGB LED, you do not eliminate some portions of the light as in the subtractive color model, but you mix in new light. If you mix all portions, you will get white light. Here, the basic colors are red, green and blue. With these, you can generate all colors of visible light.

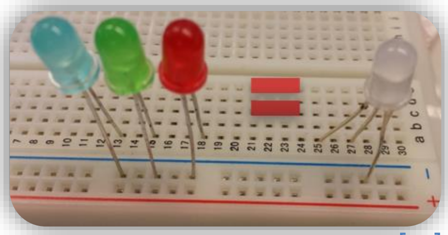

[9]

This principle is also used by televisions, and monitors. If your eyes are close to the screen, they will see, that there are many tiny red, green and blue pixels.

Of course, it makes more sense to control the RGB LED with the Arduino instead of always changing the wiring.

1. This time, do not connect the yellow wires to the plus bar, but to three digital pins of the Arduino: 9 (blue), 10 (green) and 11 (red).
2. Declare – under settings – three `int` **variables** for the three LEDs. Store in these variables pin 9 (blue), 10 (green) and 11 (red).
3. The LEDs are connected to outputs. Use **`pinMode`** to define the correct connection type.
4. Use **`digitalWrite`** to switch the LEDs on and off, first one after the other, and then in combination.
5. Use **`delays`** after each switching operation, so you can see the differences.
6. Save, and try it out. ☺

If the functionality of the RGB LED is not yet clear to you, perhaps the figure on the right will help. Imagine the RGB LED as a blue, green and red LED in one. The (here) left leg controls the blue LED, the middle leg controls the green LED and the right leg controls the red LED. The fourth leg is the connection of all three colors to the negative pole.


[10]

More than just three Colors

As you know from the table on page 2, RGB LEDs can display more than just three colors. To realize that, you just have to switch on two, or even three LEDs at the same time.

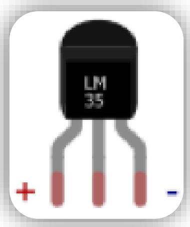
1. Save your sketch under a new name.
2. Change it, so that the RGB LED performs a color play in red, green, blue, yellow, pink, turquois and white.
3. Test your program, and do not forget to save it.

Later, you can use all these colors to display different temperatures. So, even a child can see the difference between hot and cold at first glance.

The Temperature

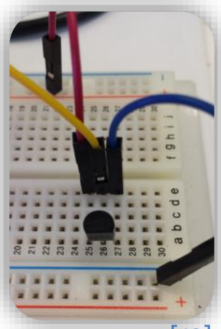
There are many different devices, which can measure temperature.
The LM35 is one of them. Even if it does not look like that, the LM35 includes a complex circuit, which makes your work much easier.

LM35



[11]

The LM35 has three legs. The complex circuit inside has to be connected to an electricity source. For that, you need the outer two legs. If the flat side is facing you, the left leg has to be connected to the 5V connection (i.e. the positive terminal). The right leg has to be connected to GND (i.e. the negative terminal). The middle pin does the actual temperature measurement. Per 1 °C a voltage of 10 mV is applied here; at 20 °C 200mV. The middle pin is connected to an analog input of the Arduino.


[12]

If the LM35 gets very hot, you have installed it the wrong way around. In that case, pull the plug quickly, and correct the error. Attention: Do not touch the LM35. You will burn your fingers.

Analog Pins

Up to now, you only know digital pins, which are used to connect e.g. an LED. These pins can only read in and out binary values, i.e. on/off, 0/1 or high and low for a high or low voltage. A temperature sensor like the LM35, for example, can not only measure two, but many intermediate values in the range between the minimum and maximum value. To realize this, you need the analog pins (A0 to A5) on the Arduino, where a whole range can be measured. To read out the analog inputs of the Arduino you can use the following function:

```
variable-name = analogRead(pin-name);
```

This function generates values between 0 and 1023. 0 means, that there is no voltage; 1023 stands for the full 5 volts.

1. To start programming you have to create a new sketch. Save it under a reasonable name.
2. Copy the variables from the sketch before. Assign them the correct variable type in the **`setup()`**.
3. Declare another **`int`** **variable** for the analog pin of the temperature sensor.
4. Before continuing, it is best to save the sensor value in another variable. The sensor just supplies whole numbers. So, you need an **`int`** **variable**. Within the **`loop()`**, assign the sensor value to this variable.

The Calculation

You dispose the following facts:

- a sensor value between 0 und 1023.

- a voltage scale between 0 and 5 V.
- a sensor, which provides 10 mV, so 0.01 V per 1 °C.

You probably guessed: There is a lot of converting to do. But do not worry, this will now be done step by step.

To calculate the temperature from the sensor value, you need to know the voltage applied to the LM35. For this, you have to convert the sensor value into the voltage. This can be done with the following formula:

voltage = (5.0 / 1023) * sensorvalue

Use 5.0 instead of 5. Thus, the Arduino calculates with floating point numbers.

The sensor provides 0.01 V per °C. Now that you know the voltage, you can divide it by the conversion factor 0.01 to get the temperature. If you are good in maths class, you know that dividing by 0.01 is the same as multiplying by 100.
Therefore, the formula now looks like this:

temperature = 100 * voltage

The Program

1. For the voltage, you use a division with decimal numbers. Therefore, you need a new variable type: **`float`**. Declare **`float` variables** for the voltage, and the temperature within the settings.
   ```
   float temperature;
   float voltage;
   ```
2. Calculate the voltage, and store it in your variable.
3. Also, calculate the temperature, and store it in the variable.
4. Start the serial monitor within **`setup()`**. Output the sensor value, the voltage, and the temperature within the **`loop()`**.
5. Insert a **`delay()`**, so that the Arduino checks the temperature only once per second.
6. Test you sketch! Look at the values on the serial monitor, and make some notes on sample values (e.g. room air, finger).

The Color Thermometer

Is everything working? Great! Now, you got everything you need to measure temperatures. And you have an RGB LED, which can display colors. So, you have everything to display temperature. What do you think about making the LED blue at room temperature? When it gets warmer, the LED can change over yellow to red. Use the following table to note at which temperature you want to display which color. And, of course, which of the three LEDs you have to light up for this.

Temperature	**Color**	**Red**	**Green**	**Blue**

Sounds like a plan. But therefore, you have to modify your sketch a bit.

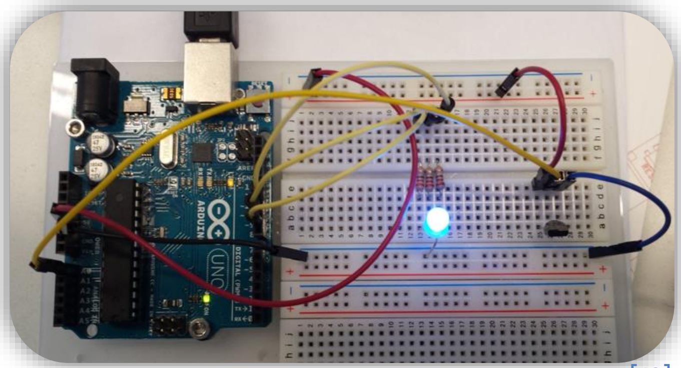
if-Commands with Conditions for Advanced Learners

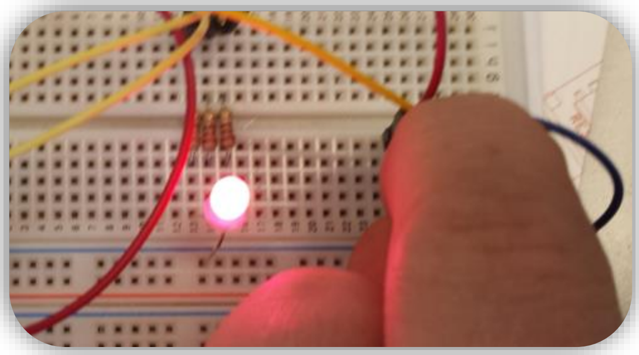
A condition does not always have to check whether two things have the same value (as a reminder: ==), but can also compare whether something is smaller (<), larger (>) or smaller/larger equal (<= or >=, respectively) to something else. Using "if", you can also check for more than one condition at the same time. Link these conditions with `&&` (logical And). However, you have to write parentheses around the individual conditions. This way, the instruction will only be executed, if both conditions are true at the same time.

```
if ( (condition1) && (condition2) ) {expressions}
```

1. Create an if statement for each of your temperature ranges. You only need one condition for the lowest and highest range, all ranges in between need two.
2. Let the right LEDs light up depending on the range. **Hint:** Do not forget to switch off the LEDs, which you do not need.

If everything is working correctly, it should look like what you can see in the following figures. If it is not working, have another look at your if conditions, and make sure you switched on and off the right LEDs!


[13]


[14]

Is everything working? Congratulations! If you want to continue even further, you can try the bonus. Just ask the tutors for the bonus sheet.

List of references:

Fig. 1 – *Source: wikipedia.org (https://de.wikipedia.org/wiki/Datei:LED_throwies_chaos.jpg), Author: Akimbomidget, CC BY-SA 2.5 (https://creativecommons.org/licenses/by-sa/2.5/deed.de), 2022-07-17.*

Fig. 2 – *Source: pxhere.com (https://pxhere.com/de/photo/537708), CC0 1.0 Universal (CC0 1.) (https://creativecommons.org/publicdomain/zero/1.0/), 2022-07-12.*

Fig. 3 to 5, 11 – *Source: Screenshot of the Fritzing Software (http://fritzing.org), CC BY-SA 3.0 Attribution-ShareAlike 3.0 Unported (https://creativecommons.org/licenses/by-sa/3.0/), 2022-06-14.*

Fig. 8 – *Source: wikipedia.org (https://de.wikipedia.org/wiki/Datei:Synthese-.svg), Author: Quark67, CC BY-SA 2.5 (https://creativecommons.org/licenses/by-sa/2.5/deed.de), 2022-07-12.*

Fig. 9 – *Source: wikipedia.org (https://de.m.wikipedia.org/wiki/Datei:Monitor_1.jpg), Author: Ernst Schütte, CC BY-SA 3.0 (https://creativecommons.org/licenses/by-sa/3.0/deed.de), 2022-07-12.*

Fig. 6, 7, 10, 12 to 14 – *Source: InfoSphere*

🎯, ⚠, , – *Source: InfoSphere*