Station 4 – Farbthermometer – Bonus 2

Neue Herausforderungen

Ihr wollt neue Herausforderungen? Die könnt ihr haben! Irgendwo haben wir die größte Herausforderung des InfoSpheres versteckt.

Bisher überprüft ihr immer, ob die Temperatur in einen bestimmten Bereich fällt und gebt der RGB-LED dann eine vorher fest einprogrammierte Farbe. Das funktioniert wunderbar, führt aber zu ziemlich harten Übergängen zwischen den Farben. Es geht aber auch anders. Wenn ihr die Temperatur kennt, könnt ihr eine passende Farbe berechnen. Farbe berechnen, wie soll das denn gehen? Ganz einfach, denn ihr könnt die rote, grüne und blaue LED ja einzeln steuern. Während der Rotanteil immer mehr ansteigt, fällt der Blauanteil mit zunehmender Temperatur.

- Ihr seid fit in Mathe? Dann könnt ihr drauf losrechnen und euch überlegen, wie ihr das Problem angeht.
- Ihr seid zwar fit in Mathe, aber seid wie waschechte Programmierende zu faul, selbst zu rechnen? Unten werden euch zwei praktische Funktionen des Arduinos vorgestellt. Aber Achtung: Das Rechnen können sie euch zwar abnehmen, aber denken müsst ihr immer noch selbst. 😉

Die map-Funktion

Die Funktion **`map()`** ist im Grunde eine mathematische Funktion, die einen Wert von einem Wertebereich in einen anderen umrechnet. Ein Beispiel: Ihr schreibt in der Schule eine Klassenarbeit. Insgesamt gibt es 40 Punkte zu holen. Eure Lehrerin hat nun aber ein Bewertungsschema in Prozent. Benutzt sie nun

```
map(punktzahl, 0, 40, 0, 100);
```

so übersetzt diese Funktion die `punktzahl` (0-40) automatisch in Prozent (0-100). Setzt ihr also z. B. 20 als `punktzahl` ein, so gibt euch die Funktion hier den Wert 50 zurück.
Das funktioniert übrigens auch umgekehrt:

```
map(punktzahl, 0, 40, 6, 1);
```

Hier wird euch für 0 Punkte eine 6, für 40 Punkte eine 1 gegeben.

Wie hilft euch das?
Wenn die rote LED bei 20 °C noch aus sein soll (also den Wert 0 hat), aber bei 25 °C so hell wie möglich leuchten soll (also 255), könnt ihr einfach

```
map(temperatur, 20, 25, 0, 255);
```

benutzen.

Die constrain-Funktion

`map()` ist eine wirklich tolle Funktion. Allerdings hat sie eine Schwäche: Wenn ihr für eure Klassenarbeit gut gelernt habt, alles richtig macht und obendrein noch 10 Bonuspunkte für Schönschrift bekommt (also 50 Punkte habt), spuckt euch `map(punktzahl, 0, 40, 6, 1);` vielleicht eine 0 oder gar eine -1 aus. Hier kommt **`constrain()`** ins Spiel. Diese Funktion sorgt dafür, dass eine Zahl in bestimmten Grenzen bleibt. Das Ergebnis von

```
constrain(wert, a, b);
```

ist also immer >=a und <=b.

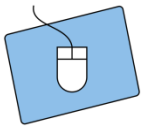
Station 4 – Farbthermometer – Bonus 2

Eure Lehrerin kann ihre berechnete Note noch korrigieren.

```
echteNote = constrain(berechneteNote, 1, 6);
```

So wird aus der 0 doch wieder eine 1.

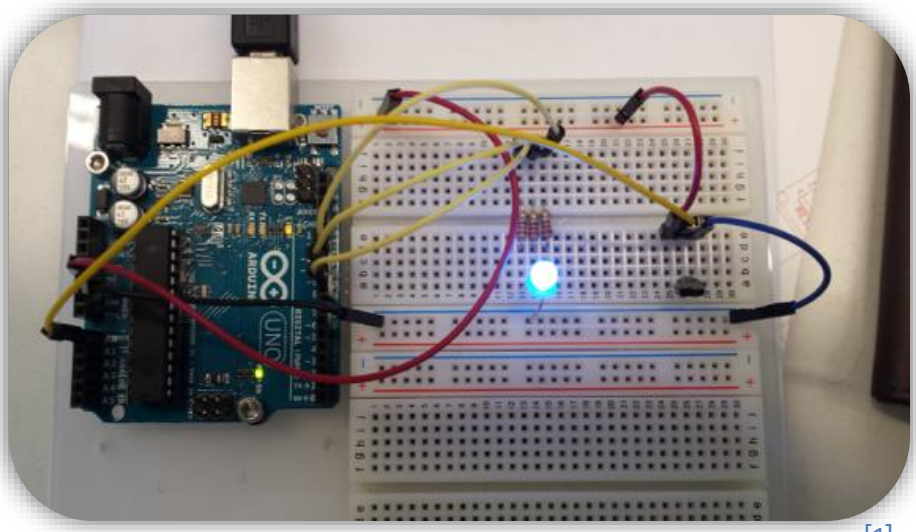
Für euch bedeutet das Folgendes: Ihr habt gelernt, dass die Werte für **`analogWrite`** zwischen 0 und 255 liegen müssen. **`constrain`** kann das für euch erledigen.



Nutzt die **`map`**– und die **`constrain`**-Funktion für euer Farbthermometer.

Wenn ihr auch diese Herausforderung gemeistert habt, könnt ihr euch eigenen Herausforderungen stellen! Hier ein paar **Anregungen**, wie ihr weitermachen könnt:

- anderer Temperaturbereich: Ist ein Fön oder Heizlüfter in der Nähe? Versucht doch mal eure Schaltung entsprechend anzupassen.
- variabler Temperaturbereich: In eurem InfoSphere-Kit findet ihr in dem Materialkästchen der Station 0 einen Taster. Könnt ihr damit den Temperaturbereich auf Knopfdruck umschalten, sodass einmal Temperaturen im Bereich 20 bis 25 °C und einmal solche über 30 °C in Farben verwandelt werden?



[1]

Viel Spaß beim Weiterbasteln!

Quellenverzeichnis:

Abb. 1, – *Quelle: InfoSphere, CC BY-SA 4.0 Attribution-ShareAlike 4.0 International (https://creativecommons.org/licenses/by-sa/4.0/)*