Station 4 – Farbthermometer – Bonus 2

Neue Herausforderungen

Ihr wollt neue Herausforderungen? Die könnt ihr haben! 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. Farben 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 drauflos rechnen und euch überlegen, wie ihr das Problem angeht!
- Ihr seid zwar fit in Mathe, aber gerade zu faul selbst zu rechnen? Unten werden euch zwei praktische Funktionen des Arduino vorgestellt. Aber Achtung, das Rechnen können sie euch abnehmen, das Denken müsst ihr aber immer noch selbst erledigen!

Map()

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. Euer Lehrer hat aber nur ein Bewertungsschema in Prozent. Benutzt er nun

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

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

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

Dies gibt euch für 0 Punkte eine Sechs, für 40 Punkte kriegt ihr eine Eins!

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.

Station 4 – Farbthermometer – Bonus 2

Constrain()

`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(punktezahl,0,40,6,1); vielleicht eine Null oder gar eine Minus Eins 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.
Euer Lehrer kann also seine berechnete Note noch korrigieren.

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

So wird aus der Null doch wieder eine Eins!

Und für euch? Ihr habt gelernt, dass die Werte für analogWrite zwischen 0 und 255 liegen müssen. Constrain kann das für euch erledigen!

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 Bastelkit findet sich der Taster vom Einstiegsprojekt. Könnt ihr damit den Temperaturbereich auf Knopfdruck umschalten? So dass einmal Temperaturen im Bereich von 20-25 °C und einmal nur solche über 30°C in Farben verwandelt werden.

Viel Spaß beim Weiterbasteln!

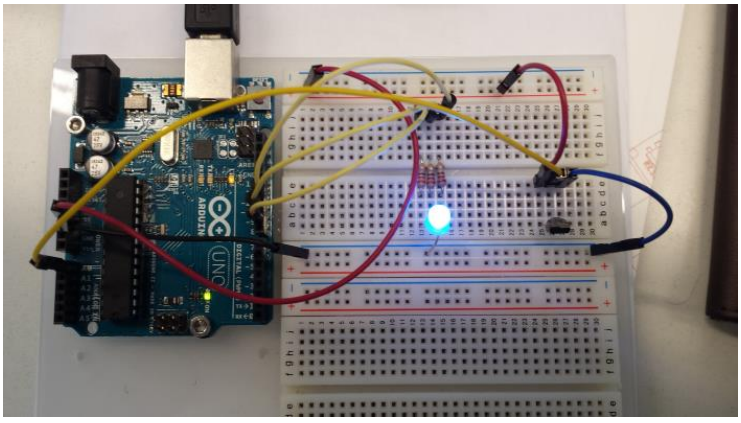

Abb.: Komplette Schlatung, Quelle: InfoSphere