Station 4 – Farbthermometer

Temperatur zum Leuchten bringen

[1]

Ihr habt euch für ein Projekt aus der Kategorie „Die Welt ein bisschen besser machen“ entschieden. Warum dieses Projekt die Welt besser macht? Wie viele Menschen haben sich schon die Finger verbrannt, weil die Herdplatte noch heiß war? Oder haben angewidert das Gesicht verzogen, weil der Tee schon kalt war? Das alles passiert nur, weil der Mensch Temperatur nicht sehen kann. Das könnt ihr jetzt ändern!

[2]

Eure Aufgabe ist es, ein Thermometer zu konstruieren, das mittels verschiedener Farben anzeigt, welche Temperatur gerade herrscht.

Aufbau der Schaltung

[3]

[4]

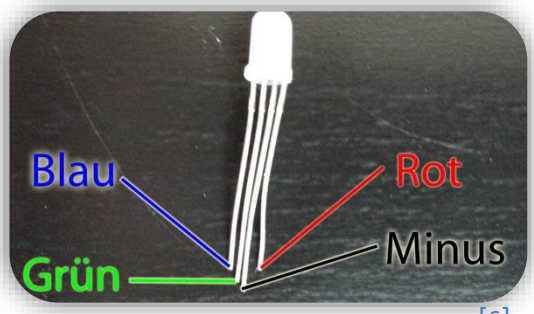
[5]

Eure Schaltung beinhaltet folgende Elemente:

- 3x 220 Ω Widerstand (Abbildung 3)
- Temperaturfühler LM35 (Abbildung 4)
- RGB-LED (Abbildung 5)
- 4 gelbe, 1 blaues und 1 rotes langes Kabel
- 1 blaues und 1 rotes kurzes Kabel

Farbmischung

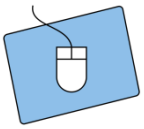
RGB-LEDs



[6]

RGB-LEDs sehen zwar aus wie normale LEDs (die ihr aus dem Einstiegsprojekt kennt), aber in Wirklichkeit verstecken sich darin drei verschiedene LEDs: eine rote, eine grüne und eine blaue LED. Daher kommt auch der Name RGB-LED. Das wirft natürlich die Frage auf, warum sie keine sechs Beinchen, sondern nur vier haben. Das liegt daran, dass der Hersteller die Minuspole schon im Gehäuse verbunden hat. Hier muss das **längste Beinchen** mit dem Minuspol verbunden werden. Legt ihr die LED so vor euch, dass das längste Beinchen an dritter Stelle liegt, könnt ihr in der Abbildung ablesen, welches Beinchen zum Pluspol welcher Farbe gehört. Diese drei Farben reichen, um alle sichtbaren Farben darzustellen. Doch dazu später mehr.

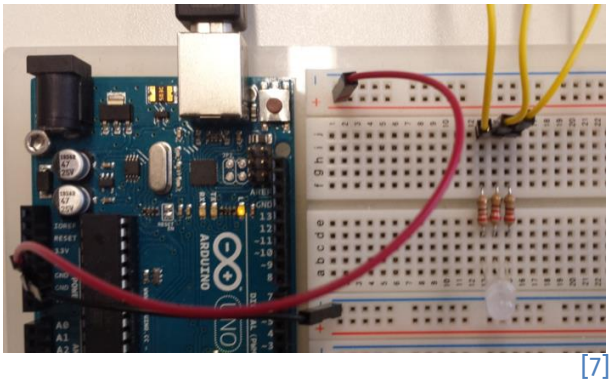
Station 4 – Farbthermometer



1. Als Erste verbindet ihr die Plus- und Minusleiste des Steckbretts mit 5V bzw. GND auf dem Arduino, so wie ihr es aus dem Einstiegsprojekt kennt.
2. Steckt die **RGB-LED** jetzt so in das Steckbrett, dass der **Minus-Pin** in der Minusleiste des Steckbretts steckt. Die **anderen Pins** kommen jeweils in eine **eigene Reihe** des Steckbretts. Anschließend werden die Reihen jeweils mit einem **Widerstand** mit der anderen Seite des Steckbretts verbunden. Dort könnt ihr auch schon **drei gelbe Steckkabel** einstecken, das obere Ende hängt aber erstmal in der Luft.
3. Nachdem ihr den Arduino mit dem Computer verbunden habt, könnt ihr auch schon loslegen: Verbindet die Steckkabel jeweils mit der Plusleiste und notiert eure Beobachtungen in der **Tabelle**.

Hinweis: Wenn ihr Probleme habt, die Schaltung zu bauen, kann euch Abbildung 7 helfen, die die fast fertige Schaltung zeigt.

Verbundene gelbe Kabel	**Farbe der LED**
links	
Mitte	
rechts	
links und Mitte	
links und rechts	
Mitte und rechts	
links, Mitte und rechts	



[7]

Habt ihr die Tabelle ausgefüllt? Super! Nun stellt sich euch sicher die Frage, wie es sein kann, dass das Ergebnis überhaupt nichts mit dem zu tun hat, was ihr im Kunstunterricht über Farbmischung gelernt habt.

Die nächste Seite wird euch bei der Beantwortung dieser Frage helfen.

Station 4 – Farbthermometer

Additive Farbmischung

Aus dem Kunstunterricht kennt ihr den guten alten Malkasten. Wenn ihr dort die Farben Blau und Gelb mischt, erhaltet ihr Grün. Das funktioniert, weil sich das weiße Licht aus allen Farben zusammensetzt. Die Farben aus dem Malkasten filtern bestimmte Anteile aus dem Licht, so dass beim Mischen mehrere Anteile wegfallen. Mischt man alle Farben, wird jeder Anteil gefiltert und das Resultat ist schwarz.

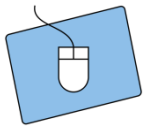

[8]

Im Gegensatz zu diesem sogenannten subtraktiven Farbmodell, nehmt ihr bei RGB-LEDs kein Licht weg, sondern mischt neues dazu. Mischt man alle Anteile, so erhält man hier weißes Licht. Die Grundfarben sind hier Rot, Grün und Blau. Mit diesen lassen sich alle Farben des sichtbaren Lichts erzeugen.

Dieses Prinzip machen sich auch Fernseher und Monitore zu nutze. Betrachtet ihr den Bildschirm aus der Nähe, so seht ihr, dass auch dieser sich aus winzigen roten, grünen und blauen Bildpunkten zusammensetzt.

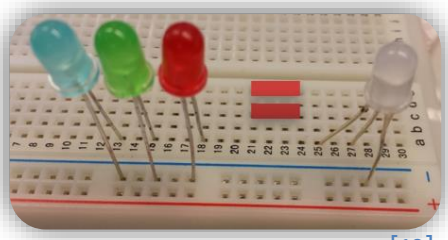

[9]

Sinnvoller als immer Kabel umstecken zu müssen, ist es, die RGB-LED über den Arduino zu steuern.



1. Verbindet die gelben Kabel dieses Mal nicht mit der Plusleiste, sondern mit digitalen Pins: Pins 9 (Blau), 10 (Grün) und 11 (Rot).
2. Erstellt im Bereich Einstellungen für die drei LEDs **Variablen** vom **Typ** `int` und speichert in ihnen die Pins 9 (Blau), 10 (Grün) und 11 (Rot) ab.
3. Die LEDs sind an Ausgängen des Arduinos angeschlossen. Legt den richtigen **Anschlusstypen** mit `pinMode` für die drei Variablen fest.
4. Schaltet die LEDs nacheinander und in verschiedenen Kombinationen mit `digitalWrite` an und aus.
5. Benutzt `delays` zwischen den Schaltvorgängen, um die Unterschiede zu sehen.
6. Speichern und ausprobieren. 😊

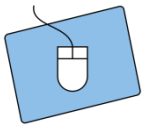
Falls euch das **Prinzip der RGB-LED** noch nicht ganz klar ist, hilft euch vielleicht die Abbildung rechts. Stellt euch die RGB-LED als eine blaue, grüne und rote LED in einer einzigen vor. Das (hier) linke Beinchen steuert die blaue, das mittlere die grüne und das rechte die rote LED. Das vierte Beinchen ist die Verbindung aller drei Farben zum Minuspol.


[10]

Station 4 – Farbthermometer

Mehr als nur drei Farben

Wie ihr bereits anhand der Tabelle auf Seite 2 gesehen habt, können RGB-LEDs noch mehr als nur drei Farben darstellen. Dazu müsst ihr zwei oder drei der eingebauten LEDs anschalten.



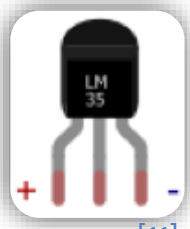
1. Speichert euren bisherigen Sketch unter einem neuen Namen ab.
2. Verändert ihn nun so, dass eure RGB-LED ein Farbspiel aus Rot, Grün, Blau, Gelb, Pink, Türkis und Weiß aufführt.
3. Testet euer Programm und vergesst das Speichern nicht.

All diese Farben könnt ihr später nutzen, um verschiedene Temperaturen darzustellen. So sieht selbst ein Kind auf den ersten Blick den Unterschied zwischen warm und kalt.

Die Temperatur

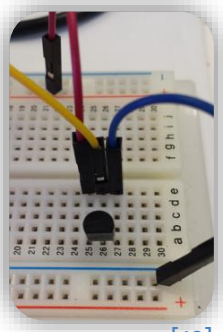
Es gibt viele verschiedene elektronische Bauteile, die Temperaturen messen zu können. Der LM35 ist eines davon. Auch wenn er nicht danach aussieht, beherbergt er eine komplexe Schaltung, die euch die Arbeit deutlich erleichtert.

LM35



[11]

Der LM35 hat drei Beinchen. Die komplexe Schaltung im Inneren muss mit Strom versorgt werden. Hierzu dienen die beiden äußeren Beinchen. Zeigt die flache Seite zu euch, so muss das **linke Beinchen** mit dem 5V-Anschluss (also der Plusleiste) verbunden werden, das **rechte Beinchen** verbindet ihr mit GND (also der Minusleiste). Der mittlere Pin erfüllt die eigentliche Funktion des **Temperaturmessers**: Pro 1 °C liegt hier eine Spannung von 10 mV an; bei 20 °C also 200 mV. Verbunden wird der mittlere Pin des LM35 mit einem **analogen Eingang** des Arduinos.


[12]

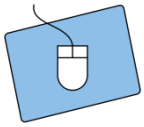
Wenn der **LM35** sehr **heiß** wird, habt ihr ihn falsch herum verbaut.
Zieht dann zügig den Stecker (nicht den LM35, sonst verbrennt ihr euch die Finger!) und korrigiert das.

Analoge Pins

Bisher kennt ihr nur **digitale Pins**, über die z. B. eine LED angeschlossen wird. An diesen Pins können nur **binäre Werte** ein- und ausgelesen werden; also ein/aus, 0/1 oder eben die bekannten Werte high und low für eine hohe bzw. niedrige Spannung. Ein Temperatursensor wie der LM35 z. B. kann aber nicht nur zwei Werte annehmen, sondern misst im Bereich zwischen dem minimalen und maximalen Wert ganz viele Zwischenwerte. Das realisieren am Arduino **analoge Pins** (A0 bis A5), an denen ein ganzer **Bereich** gemessen werden kann. Die analogen Eingänge des Arduinos werden mit der Funktion

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

ausgelesen. Diese Funktion liefert Werte zwischen 0 und 1023. 0 bedeutet, dass keine Spannung anliegt; bei 1023 liegen volle 5 Volt an.



1. Damit ihr jetzt direkt mit dem Programmieren anfangen könnt, legt ihr einen neuen Sketch an und speichert ihn unter einem sinnvollen Namen.
2. Kopiert die Variablen für die RGB-LED aus dem vorherigen Sketch. Weist ihnen auch hier in `setup()` den richtigen Typ zu.
3. Legt eine weitere **`int`-Variable** für den analogen Pin des Temperatursensors an.
4. Bevor es weitergeht, speichert ihr den Sensorwert am besten in einer **weiteren Variablen**. Der Sensor liefert nur ganze Zahlen, also braucht ihr eine Variable vom Typ `int`. Weist dieser in `loop()` den Sensorwert zu.

Die Rechnung

Folgende Fakten stehen euch zur Verfügung:

- ein Sensorwert zwischen 0 und 1023.
- eine Skala der Spannung von 0 bis 5 V.
- ein Sensor, der 10 mV, also 0,01 V pro 1 °C liefert.

Ihr ahnt es vielleicht schon: Da steht eine Menge umrechnen an. Aber keine Sorge, das wird jetzt Schritt für Schritt erledigt.
Um die Temperatur aus dem Sensorwert zu errechnen, müsst ihr die Spannung kennen, die am LM35 anliegt. Dafür müsst ihr den Sensorwert in die Spannung umrechnen. Dies gelingt mit folgender Formel:

Spannung = (5.0 / 1023) * Sensorwert

5.0 statt 5 sagt dem Arduino, dass er mit Kommazahlen rechnen soll.

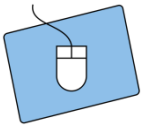
Der Sensor liefert 0,01 V pro °C. Da ihr die Spannung jetzt kennt, könnt ihr diese durch den Umrechnungsfaktor 0,01 teilen und habt die Temperatur. Wer gut in Mathe aufgepasst hat, der weiß, dass geteilt durch 0,01 das Gleiche ist wie mal 100. Daher sieht die Formel jetzt folgendermaßen aus:

Temperatur = 100 * Spannung

Auf der nächsten Seite geht es mit der Programmierung weiter.

Station 4 – Farbthermometer

Die Programmierung



1. Für die Spannung benutzt ihr eine Division mit **Kommazahlen**. Dafür braucht ihr einen neuen Variablentyp: `float`. Erstellt Variablen für Spannung und Temperatur im Bereich Einstellung vom Typen `float`:
```
float temperatur;
float spannung;
```
2. Errechnet die Spannung und speichert sie in eurer Variablen ab.
3. Berechnet die Temperatur und speichert auch sie in der angelegten Variablen ab.
4. Startet den **Serial Monitor** in `setup()` und gebt in `loop()` den Sensorwert, die Spannung und auch die Temperatur aus.
5. Fügt noch ein `delay()` ein, sodass der Arduino die Temperatur nur einmal pro Sekunde überprüft.
6. Testet euren Sketch! Schaut euch die Werte auf dem **Serial Monitor** an und macht euch ein paar Notizen zu Beispielwerten (z. B. Raumluft, Finger).

Das Farbthermometer

Funktioniert alles? Gut, jetzt habt ihr schon alles beisammen, um Temperaturen zu messen. Und ihr habt eine RGB-LED, mit der ihr mehrere Farben darstellen könnt. Ihr habt also alles, um die Temperatur anzuzeigen. Was haltet ihr davon, die LED bei Raumtemperatur blau leuchten zu lassen? Wenn es wärmer wird vielleicht über Gelb nach Rot zu wechseln? Nutzt die folgende Tabelle, um zu notieren, bei welcher Temperatur ihr welche Farbe anzeigen wollt! Und natürlich auch welche der drei LEDs dazu leuchten muss.

Temperatur	Farbe	Rot	Grün	Blau

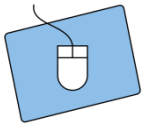
Klingt nach einem Plan! Allerdings muss euer Sketch dafür noch ein wenig erweitert werden.

if-Anweisungen mit Bedingungen für Fortgeschrittene

Eine Bedingung muss nicht immer prüfen, ob zwei Dinge denselben Wert haben (zur Erinnerung: `==`), sondern kann auch vergleichen, ob etwas kleiner (`<`), größer (`>`) oder kleiner/größer gleich (`<=` bzw. `>=`) ist. Außerdem könnt ihr mit `if` auch mehr als eine Bedingung gleichzeitig abfragen. Verknüpft diese Bedingungen dazu mit einem `&&` (logisches Und). Dabei müsst ihr aber um die einzelnen Bedingungen Klammern schreiben. So wird die Anweisung nur ausgeführt, wenn beide Bedingungen zutreffen.

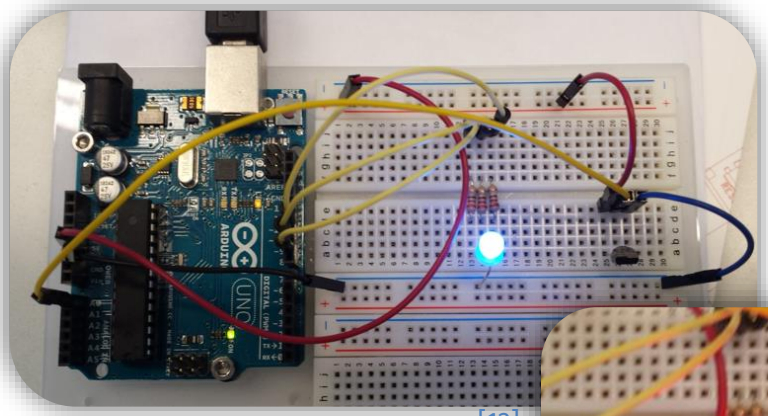
```
if ( (bedingung1) && (bedingung2) ) {anweisungen}
```

Station 4 – Farbthermometer

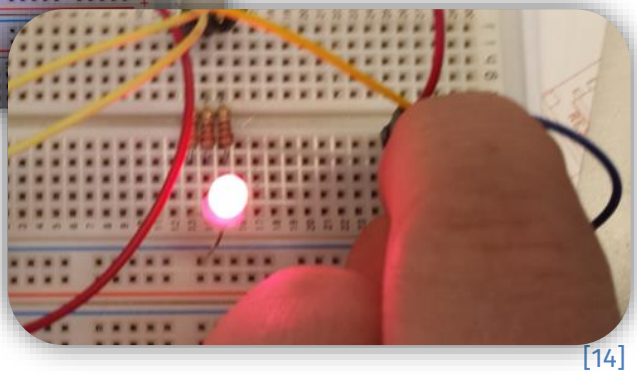


7. Erstellt für jeden eurer Temperaturbereiche eine `if`-Anweisung. Der unterste und oberste Bereich haben jeweils nur eine Bedingung, alle Bereiche dazwischen zwei.
8. Lasst je nach Bereich die richtigen LEDs leuchten.
   Hinweis: Vergesst nicht die jeweils anderen (unbenutzen) LEDs auszuschalten.

Wenn alles geklappt hat, sollte es bei euch in etwa wie in den beiden folgenden Abbildungen funktionieren. Falls nicht, checkt noch einmal eure `if`-Bedingungen und ob ihr jeweils die richtigen LEDs ein- und ausgeschaltet habt.



[13]



[14]

Hat alles geklappt? Herzlichen Glückwunsch. Wem das noch nicht genug ist, der kann sich an das Bonus-Blatt herantrauen.

Quellenverzeichnis:

Abb. 1 – *Quelle: wikipedia.org (https://de.wikipedia.org/wiki/Datei:LED_throwies_chaos.jpg), Autor: Akimbomidget, CC BY-SA 2.5, Namensnennung – Weitergabe unter gleichen Bedingungen 2.5 Generic (https://creativecommons.org/licenses/by-sa/2.5/deed.de), abgerufen am: 11.07.2022*

Abb. 2 – *Quelle: pxhere.com (https://pxhere.com/de/photo/537708), CC0 1.0 Universal (CC0 1.) Public Domain Dedication (https://creativecommons.org/publicdomain/zero/1.0/), abgerufen am: 12.07.2022*

Abb. 3 bis 5, 11 – *Quelle: Screenshot der Fritzing-Software (http://fritzing.org), CC BY-SA 3.0 Attribution-ShareAlike 3.0 Unported (https://creativecommons.org/licenses/by-sa/3.0/), abgerufen am: 14.06.2022.*

Abb. 8 – *Quelle: wikipedia.org (https://de.wikipedia.org/wiki/Datei:Synthese-.svg), Autor: Quark67, CC BY-SA 2.5, Namensnennung – Weitergabe unter gleichen Bedingungen 2.5 Generic (https://creativecommons.org/licenses/by-sa/2.5/deed.de), abgerufen am: 12.07.2022*

Abb. 9 – *Quelle: wikipedia.org (https://de.m.wikipedia.org/wiki/Datei:Monitor_1.jpg), Autor: Ernst Schütte, CC BY-SA 3.0, Namensnennung – Weitergabe unter gleichen Bedingungen 3.0 Unported (https://creativecommons.org/licenses/by-sa/3.0/deed.de), abgerufen: 12.07.2022*

Abb. 6, 7, 10, 12 bis 14 – *Quelle: InfoSphere, CC BY-SA 4.0 Attribution-ShareAlike 4.0 International (https://creativecommons.org/licenses/by-sa/4.0/)*

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