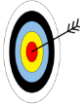


## Station 3 – Farbthermometer

## Temperatur zum Leuchten bringen

Ihr habt euch also 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!



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

## AUFBAU DER SCHALTUNG

Eure Schaltung beinhaltet folgende Elemente:

- drei 220  $\Omega$  Widerstände
- Temperatursensor TMP36
- RGB-LED
- 1 grünes, 4 blaue, 3 rote und 1 gelbes Kabel



Abb. 1: Einige LEDs



Abb. 2: Brunnen mit verbauten LEDs

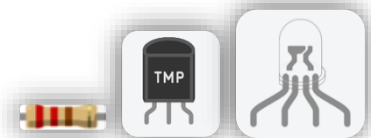


Abb. 3: Widerstand, Temperatursensor und RGB-LED

## Farbmischung

## RGB-LEDs

RGB-LEDs sehen zwar aus wie normale LEDs (die ihr ja aus dem Einstiegsprojekt kennt), aber in Wirklichkeit verstecken sich darin drei verschiedene LEDs! Nämlich 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 Minuspol schon im Gehäuse verbunden hat. Hier muss das Beinchen zwischen den Beinchen für rot und blau mit dem Minuspol verbunden werden. Habt ihr die RGB-LED auf eure Arbeitsfläche gezogen, könnt ihr mit eurer Maus über die Enden der Beinchen gehen und kriegt dann angezeigt, welches Beinchen zu welcher Farbe gehört. Normalerweise sind die Beinchen aber so angeordnet, wie ihr es in [Abbildung 4](#) sehen könnt. In Tinkercad wird das Beinchen für den Minuspol mit **Kathode** bezeichnet; lasst euch davon nicht verwirren. Diese drei Farben reichen aus, um alle sichtbaren Farben darzustellen. Doch dazu später mehr.

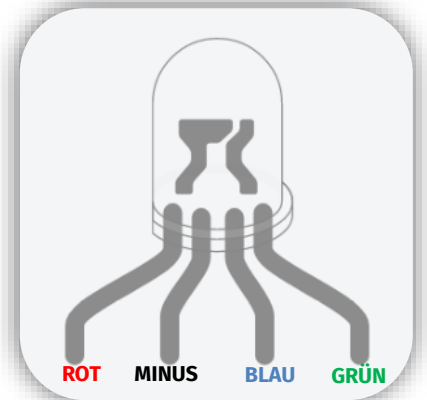


Abb. 4: RGB-LED



1. Als Erstes verbindet ihr die Plus- und Minusleiste des Steckbretts mit 5V bzw. GND auf dem Arduino, so wie ihr es aus dem Einstiegsprojekt kennt.
2. Steck die RGB-LED jetzt in das Steckbrett, und verbindet den Minus-Pin mit der Minusleiste. Die anderen Pins kommen jeweils in eine eigene Reihe des Steckbretts. Anschließend werden diese Reihen jeweils mit einem Widerstand verbunden. Dahinter könnt ihr auch schon die drei Steckkabel in der jeweiligen Farbe einstecken, das andere Ende hängt aber erstmal in der Luft.
3. Verbindet nacheinander die Steckkabel jeweils mit der Plusleiste, und notiert eure Beobachtungen in der Tabelle auf der nächsten Seite.

**[Hinweis:** Wenn ihr Probleme mit der Schaltung habt, könnt ihr in [Abbildung 5](#) auf der nächsten Seite sehen, wie sie aussehen sollte.]

## Station 3 – Farbthermometer

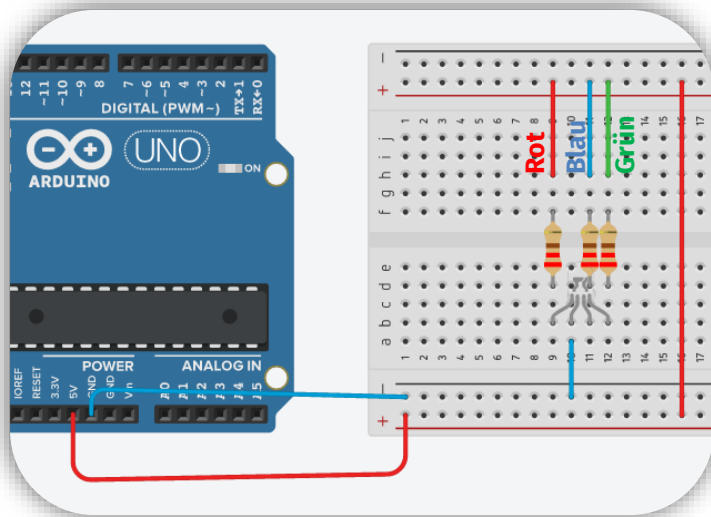


Abb. 5: Aufbau der RGB-LED Schaltung

Verbundene Kabel	Farbe der LED
Rot	
Blau	
Grün	
Rot & Blau	
Rot & Grün	
Blau & Grün	
Rot & Blau & Grün	

Habt ihr die Tabelle schon 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.

### ADDITIVE FARB MISCHUNG

Aus dem Kunstunterricht kennt ihr ja 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, sodass beim Mischen mehrere Anteile wegfallen. Mischt man alle Farben, wird jeder Anteil gefiltert und das Resultat ist schwarz.

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 blaue Bildpunkten zusammensetzt.



Abb. 6: Farbmischung

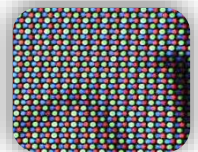
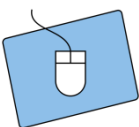


Abb. 7: Nahaufnahme Monitor

Natürlich ergibt es Sinn, die RGB-LED mit dem Arduino zu steuern, statt immer Kabel umstecken zu müssen.



1. Öffnet das Code-Fenster in eurer Schaltung, und stellt es auf „Blöcke“.
2. Die LEDs sind an den Ausgängen 9 (**Grün**), 10 (**Blau**) und 11 (**Rot**) vom Arduino angeschlossen.
3. Schaltet die LEDs nacheinander und in verschiedenen Kombinationen an und aus.
4. Benutzt **Pausen** zwischen den Schaltvorgängen, um die Unterschiede zu sehen.
5. Ausprobieren :)

Falls euch das Prinzip der RGB-LED noch nicht ganz klar ist, hilft dieses Bild vielleicht: Stellt euch die RGB-LED als eine blaue, eine grüne und eine rote LED in einer einzigen vor. Das (in diesem Bild) 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.

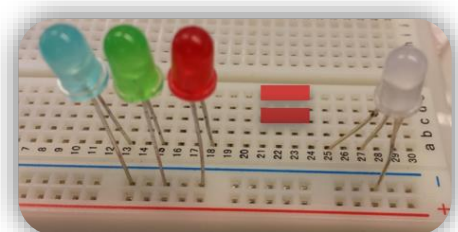


Abb. 8: Funktionsweise RGB-LED

## Station 3 – Farbthermometer

### MEHR ALS NUR DREI FARBEN

Wie ihr bereits an 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. Erweitert euren bisherigen Code nun so, das eure RGB-LED ein Farbspiel aus **Rot**, **Grün**, **Blau**, **Gelb**, **Pink**, **Türkis** und **Weiß** aufführt.
2. Testet euer Programm.

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 TMP36 ist eines davon. Auch wenn er nicht danach aussieht, beherbergt er eine komplexe Schaltung, die euch die Arbeit deutlich erleichtert.

#### TMP36

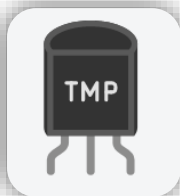


Abb. 9: TMP36

Der TMP36 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 liegen also 200 mV an. Verbunden wird der mittlere Pin des TMP36 mit einem **ANALOGEN EINGANG** des Arduino.

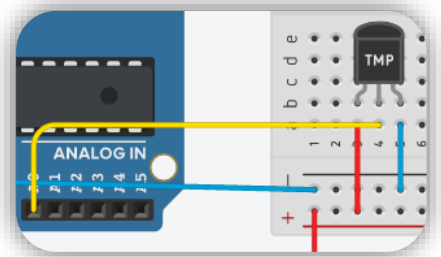


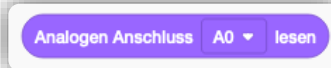
Abb. 10: Schaltung mit TMP36

#### ANALOGUE 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 **HOCH** und **NIEDRIG** für eine hohe bzw. niedrige Spannung.

Ein Temperatursensor TMP36 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 **ANALOGUE PINS** (A0 bis A5), an denen ein ganzer **BEREICH** gemessen werden kann.

Die analogen Eingänge des Arduino werden mit dem Block



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



1. Erweitert euren Code um eine weitere **Variable** für den Sensorwert. Lest den Sensorwert ein, und schaut ihn euch über den **SERIELLEN MONITOR** an.

## Station 3 – Farbthermometer

### DIE RECHNUNG

Euch stehen also folgende Fakten 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
- Es gibt eine Abweichung von 0.5 °C bei der Spannung.

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 TMP36 anliegt. Dafür müsst ihr den Sensorwert in die Spannung umrechnen. Dies gelingt mit folgender Formel:

$$\text{SPANNUNG} = (5.0 / 1023) * \text{SENSORWERT}$$

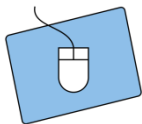
Der Sensor liefert 0,01 V pro 1 °C. Da ihr die Spannung jetzt kennt, könnt ihr diese durch den Umrechnungsfaktor 0,01 teilen und habt dann die Temperatur. Denkt daran, die 0.5 °C Abweichung von der Spannung abzuziehen. Wer gut in Mathe aufgepasst hat, der weiß, dass geteilt durch 0,01 das Gleiche ist wie mal 100. Daher sieht die Formel jetzt folgendermaßen aus:

$$\text{TEMPERATUR} = 100 * (\text{SPANNUNG} - 0.5)$$

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

Dadurch, dass es in der Blocksprache von Tinkercad keine Kommazahlen gibt, müsst ihr etwas rumtricksen, um das Programm so genau wie möglich rechnen zu lassen. Die Rechnung sieht daher bei euch so aus:

$$\begin{aligned} \text{SPANNUNG} &= 48 * \text{SENSORWERT} \\ \text{TEMPERATUR} &= (\text{SPANNUNG} - 4800) / 100 \end{aligned}$$



1. Lest als Erstes den Sensorwert ein.
2. Errechnet die Spannung, und speichert sie in eurer **Variable** ab.
3. Berechnet die Temperatur, und speichert auch sie in der angelegten **Variable** ab.
4. Schaut euch dann die errechnete Spannung und Temperatur im **SERIELLEN MONITOR** an.
5. Fügt noch eine **Pause** ein, sodass der Arduino die Temperatur nur einmal pro Sekunde überprüft.
6. Ändert die Temperatur, indem ihr während der Simulation auf den Sensor klickt und den Schieberegler bewegt.
7. Schaut euch die Werte auf dem seriellen Monitor an, und macht euch ein paar Notizen von Beispielwerten.



Abb. 11: Schieberegler Temperatur

**[Hinweis:** Sollte eure Rechnung trotzdem irgendwie nicht klappen, könnt ihr die Rechnung in der Klammer in eine kleinere Teilrechnung auslagern und das Ergebnis danach durch 100 teilen. Wenn das dann immer noch nicht funktioniert hat, fragt gerne die Betreuenden um Hilfe.]

## Station 3 – Farbthermometer

### 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.

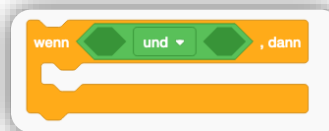
Was haltet ihr davon, die LED bei Raumtemperatur **blau** leuchten zu lassen? Wenn es wärmer wird vielleicht über **gelb** nach **rot** zu wechseln? Nutzt die folgende Tabelle, um zu notieren, bei welcher Temperatur ihr welche Farbe anzeigen wollt! Und natürlich auch, welche der drei LEDs dazu leuchten muss.

Temperatur	Farbe	Rot	Grün	Blau

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

#### WENN-DANN-ANWEISUNG MIT BEDINGUNGEN FÜR FORTGESCHRITTENE

Eine Bedingung muss nicht immer prüfen, ob zwei Dinge den gleichen Wert haben (zur Erinnerung: ==), sondern kann auch vergleichen, ob etwas kleiner (<), größer (>) oder kleiner/größer gleich (<= bzw. >=) ist. Mit **Wenn-dann**-könnt ihr auch mehr als eine Bedingung gleichzeitig abfragen. Verknüpft diese Bedingungen dazu mit einem **Und**. So wird die Anweisung nur ausgeführt, wenn beide Bedingungen zutreffen.



1. Erstellt für jeden eurer Temperaturbereiche eine **Wenn-dann**-Anweisung. Der unterste und oberste Bereich brauchen jeweils nur eine Bedingung, alle Bereiche dazwischen zwei Bedingungen.
2. Lasst je nach Bereich die richtigen LEDs leuchten.  
*[Hinweis: Vergesst nicht, die jeweils anderen (nicht benutzen) LEDs auszuschalten.]*

Wenn alles geklappt hat, sollte es bei euch ungefähr wie in den beiden Bildern unten aussehen! Falls nicht, checkt noch mal eure **Wenn-dann**-Bedingungen und ob ihr jeweils die richtigen LEDs ein- und ausgeschaltet habt.

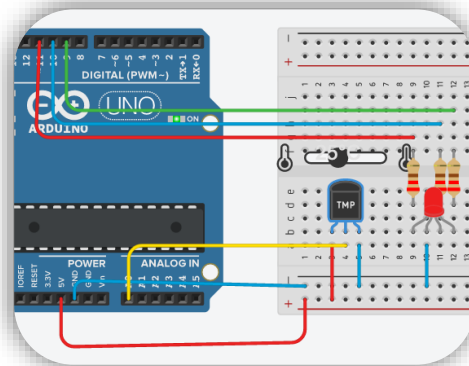
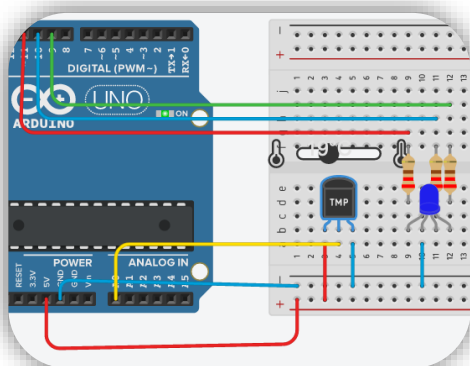


Abb. 12a und b: Komplette Schaltung. Links auf Raumtemperatur (blau), rechts heißer (rot)

## Station 3 – Farbthermometer

*Hat alles geklappt? Herzlichen Glückwunsch!*

Wenn ihr euer Programm noch erweitern möchtet, könnt ihr noch mehr Abstufungen und Farben einbauen mit mehr **Wenn-dann**-Anweisungen. Ihr könnt auch schauen, wie groß der Unterschied zwischen der eingebauten Temperaturrechnung und eurer selbstgebauten ist. Ersetzt dadurch eure Rechnung durch diesen Block, und vergleicht:



Ihr habt euch aber nicht umsonst mit der Rechnung rumgequält, keine Sorge. Die Rechnung zu verstehen, hilft euch, falls ihr später selbst programmieren wollt, da es nicht immer vorgefertigte Funktionen gibt und Sensorbauteile oft nach einem solchen Schema arbeiten.



Quellenverzeichnis:

**Abb. 1** – Quelle: [wikipedia.org](https://www.wikipedia.org/), Autor: Akimbomidget (CC BY-SA 2.5)

**Abb. 2** – Quelle: [wikipedia.org](https://www.wikipedia.org/), Autor: Diliff (CC BY-SA 3.0)





**Abb. 3, 4, 5, 9, 10, 11, 12** – Quelle: Screenshots von Tinkercad <https://www.tinkercad.com/>

**Abb. 6** – Quelle: [wikipedia.org](https://www.wikipedia.org/), Autor: Quark67 (CC SA 3.0)

**Abb. 7** – Quelle: [wikipedia.org](https://www.wikipedia.org/), Autor: Ernst Schütte (CC SA 3.0)

**Abb. 8** – Quelle: InfoSphere

**Alle Codeblock-Screenshots** – Quelle: Screenshots von Tinkercad <https://www.tinkercad.com/>

 ,  ,  ,  angefertigt vom InfoSphere-Team