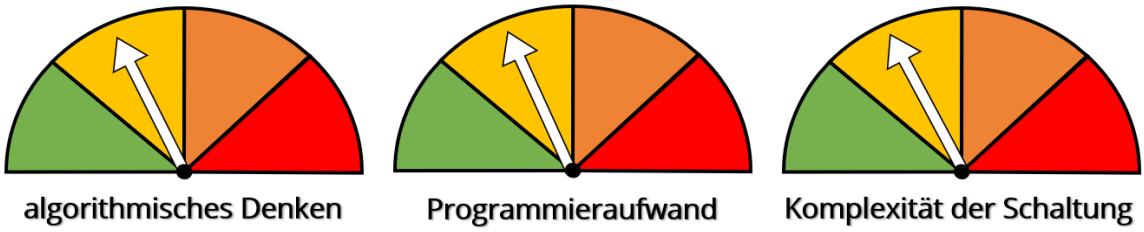
TECHNISCHE UNIVERSITÄT DRESDEN

Arduino Uno

Station 4 | Farbthermometer



Aotaro, „The Beauty of Bokeh“, pxhere.com , CC-BY 2.0

Professur für Didaktik der Informatik

UM WAS WIRD ES IN DIESER STATION GEHEN?

Wie viele Menschen haben sich schon die Finger verbrannt, weil die Herdplatte noch heiß war? Oder haben sich am heißen Essen den Mund verbrannt? Das alles passiert nur weil der Mensch Temperatur nicht sehen kann. Das könnt ihr jetzt ändern! Eure Aufgabe heute ist es, ein Thermometer zu bauen, das die Höhe der Temperatur durch Farben anzeigt.

BENÖTIGTE BAUTEILE

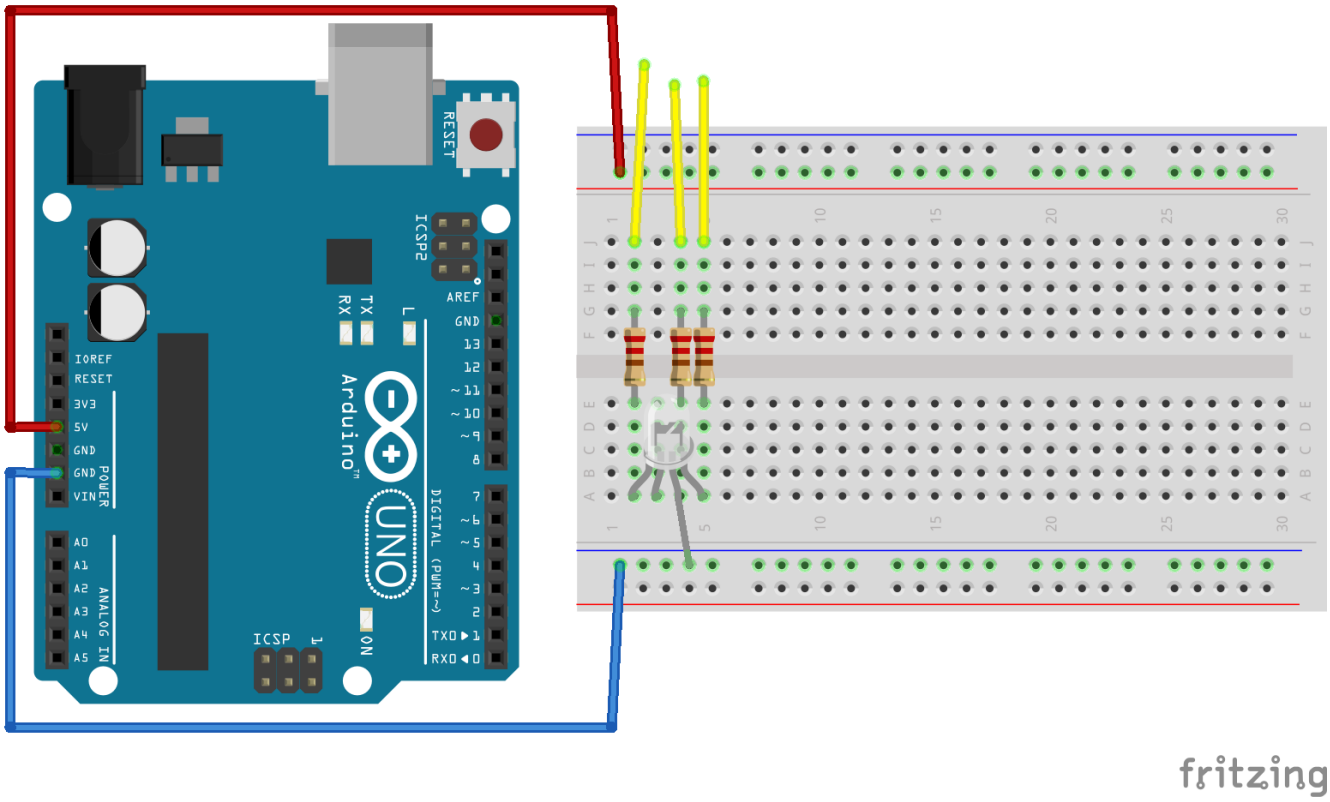
Zusätzlich zum Arduino und dem Steckbrett brauchst du folgende Bauteile:

3 Widerstände	Widerstände schützen Bauteile vor Überhitzung durch zu viel Strom. Beachte die Farbe des Widerstandes, er sollte rot-braun-gold sein.	
Temperatur-Sensor	Der Temperatursensor misst die aktuelle Temperatur.	
RGB-LED	Die RGB-LED besteht aus einer roten, einer grünen und einer blauen LED. Sie kann in unterschiedlichen Farben leuchten.	

AUFGABE 1 – FARBEN MISCHEN

Als erstes bringst du die RGB-LED in verschiedenen Farben zum Leuchten.

1. RGB-LED anschließen



Versorge das Steckbrett wieder mit Strom, indem du die äußeren Leisten mit dem + Pol und – Pol verbindest. Verbinde das längste Beinchen der RGB-LED mit dem – Pol. Verbinde die restlichen Beinchen über einen Widerstand mit einem Kabel. Lasse die Enden der Kabel noch an einem Ende frei hängen.

WAS IST EINE RGB-LED?

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 auch der Name RGB-LED. Das längste Beinchen der RGB-LED sind die Minuspole der einzelnen LEDs, die schon im Gehäuse miteinander verbunden sind. Die anderen drei Beinchen sind die + Pole der einzelnen LEDs. Wenn nicht nur eine LED leuchtet, sondern zum Beispiel die rote und grüne LED gleichzeitig, entsteht eine neue Farbe. Das kannst du dir vorstellen, wie den Farbkasten in der Schule. Welche Farben du mischen kannst, sollst du gleich herausfinden.

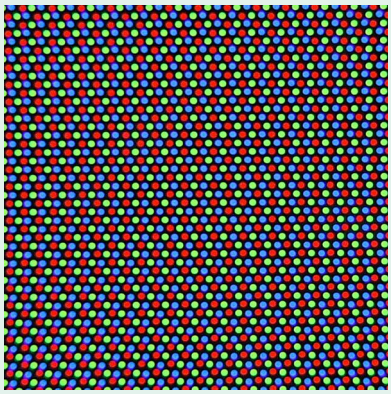
2. Farben mischen

   Stecke die frei hängenden Kabel nacheinander in die äußere Leiste, die mit dem + Pol verbunden ist. Welche Farbe nimmt die RGB-LED an? Du kannst die Farben auch mischen, indem du mehrere Kabel mit dem + Pol verbindest. Schreibe deine Ergebnisse auf.

Kabel	Farbe
links	
mitte	
rechts	
links + mitte	
links + rechts	
rechts + mitte	
links + rechts + mitte	

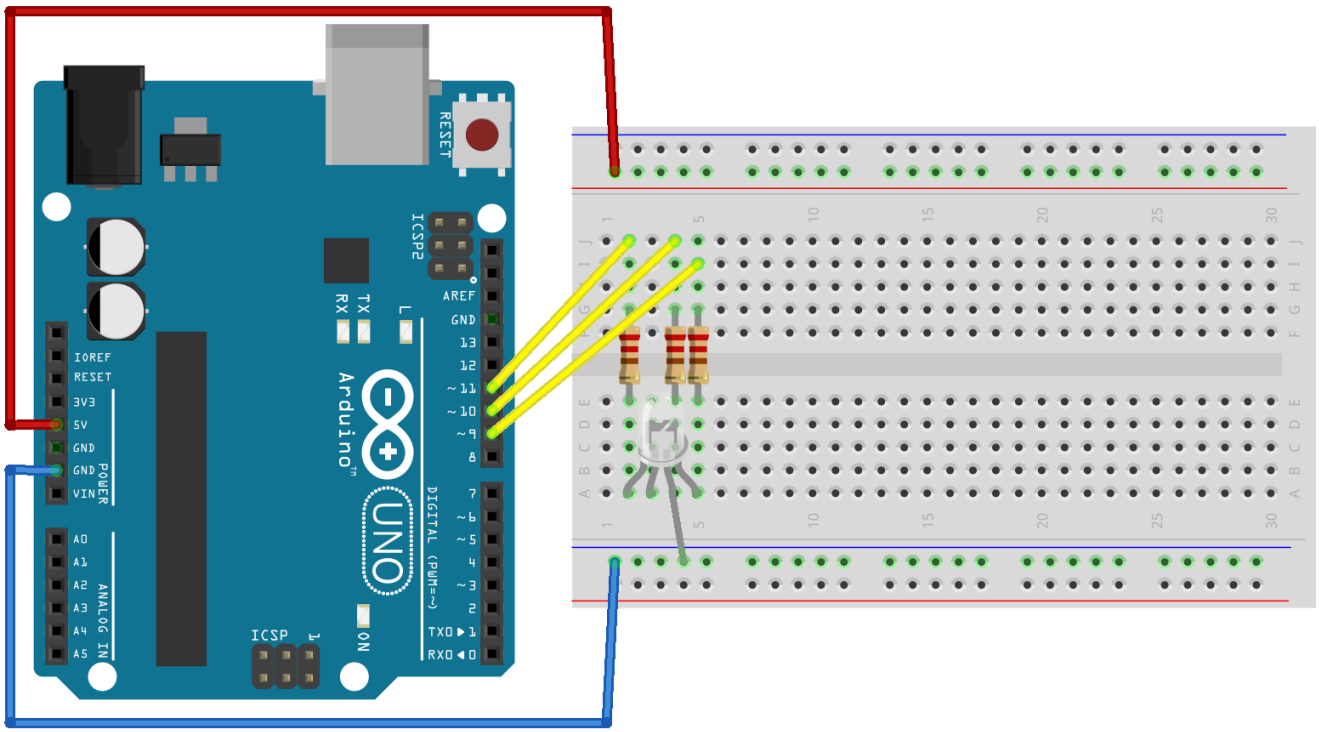
FARBMISCHUNG DER RGB-LED

Das ist aber komisch! Hast du schon einmal gesehen, dass die Mischung von mehreren Farben weiß ergibt? Tatsächlich haben LEDs eine andere Farbmischung, als du sie vielleicht von einem Malkasten kennst. Sie heißt **additive Farbmischung**. Diese nutzen auch Fernseher und andere Bildschirme. Wenn du einen Bildschirm aus der Nähe betrachtest, kannst du ganz viele rote, grüne und blaue Bildpunkte sehen. Alleine aus diesen Farben können alle weiteren Farben gemischt werden.



Marcin Floryan, „A close-up on a CRT screen“, wikipedia.org, gemeinfrei

3. RGB-LED an analoge Pins schließen
   Wenn du für die Temperaturen unterschiedliche Farben verwenden willst, ist es unpraktisch immer die Kabel umstecken zu müssen. Verbinde die drei Kabel der LEDs deswegen mit den digitalen Pins 9, 10, 11, um sie mit dem Arduino anzusteuern.

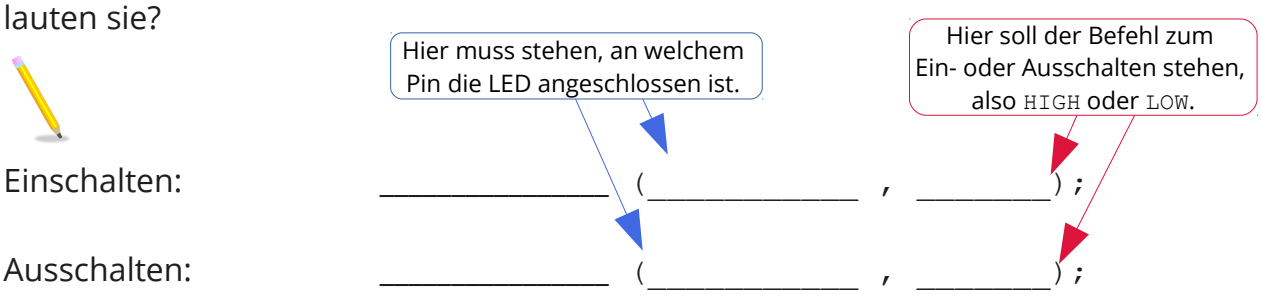


AUFGABE 2 – DISKOLICHT

Nachdem du die Farben schon durch Umstecken der Kabel gemischt hast, sollst du die Farben jetzt durch den Arduino steuern. Dazu sollst du ein kleines Diskolicht programmieren, das die Farben automatisch ändert.

1. Pin-Variable definieren und digitalen Pin verbinden
   Wie in der Einführungsstation musst du auch hier den Arduino wissen lassen, an welchen Pins die LEDs angeschlossen sind. Definiere dafür drei Variablen. Am besten ist es, den Variablen den Namen der Farbe zu geben, die an dem Pin angeschlossen ist. Beachte aber, dass Arduino keine Umlaute Ä, Ö, Ü versteht!
   Außerdem muss der Arduino wissen, dass an dem Pin etwas angeschlossen ist, das angeschaltet werden soll. Lege deswegen den `pinMode` auf `OUTPUT` fest.
   Falls du dafür Hilfe brauchst, schaue dir noch einmal in der Einstiegsstation Aufgabe 3 Punkt 2 und 3 an.

2. LEDs anschalten
   Die Befehle zum An- und Ausschalten einer LED kennst du bereits. Wie lauten sie?



   Schalte mithilfe der Befehle erst eine LED an, dann aus. Als nächstes soll eine andere LED leuchten. Bedenke wieder, dass der Arduino sehr schnell arbeitet. Er muss also eine Pause machen, nachdem die LED angeschaltet wurde, damit man sie auch sehen kann. Wie lautet noch einmal der Befehl für die Pause?

   _____

3. Farben mischen

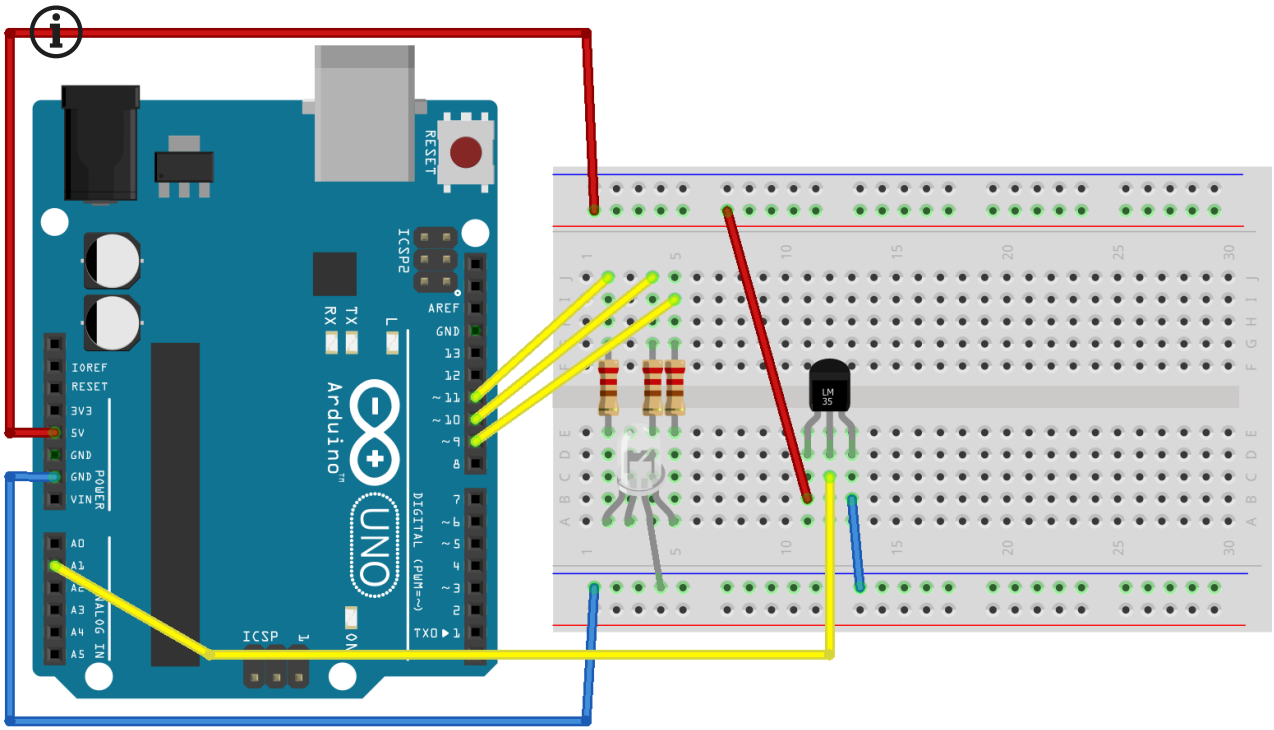
   Wenn das Einschalten der LEDs funktioniert hat, dann kannst du noch mehr Farben für das Diskolicht programmieren. Erstelle die Lichtabfolge rot – grün – blau – gelb – pink – türkis – weiß. Schalte für die gemischten Farben mehrere LEDs gleichzeitig an und dann wieder aus. Wie die Farben gemischt werden, kannst du in deiner Tabelle von vorhin nachsehen.

AUFGABE 3 – TEMPERATUR MESSEN

Super! Jetzt kannst du schon verschiedene Farben programmieren. Fehlt nur noch die Temperatur, der du eine Farbe zuweisen kannst. Als nächstes sollst du deswegen die Temperatur messen.

1. <u>neues Programm öffnen</u>
   Speicher dein Diskokugel-Programm unter dem Namen Diskokugel ab. Du wirst es später noch einmal brauchen. Jetzt sollst du aber erst einmal ein neues Programm anfangen. Klicke dafür in der Programmierumgebung auf `Datei` und dann auf `Neu`.

2. <u>Temperatursensor anschließen</u>



Wenn die Flache Seite des Temperatursensors zu dir zeigt, verbinde das linke Beinchen mit dem + Pol und das rechte Beinchen mit dem – Pol. Das mittlere Beinchen verbindest du mit einem analogen Pin.

3. <u>Pin-Variable definieren</u>
   Auch für den Temperatursensor musst du eine Variable definieren, die speichert, an welchem Pin der Sensor angeschlossen ist.

4. Speichern der Messergebnisse in einer Variable
   Nach der Messung der Temperatur soll das Ergebnis gespeichert werden. Dafür brauchst du wieder eine Variable. Du weißt aber erst nach der Messung, welche Zahl die Variable haben soll. Deswegen musst du in der Definition der Variable noch keine Zahl zuordnen. Das kann so aussehen:

```
int ergebnis;
```

5. Befehl zur Messung
   Zuletzt musst du dem Arduino noch sagen, dass er die Temperatur-messung durchführen soll. Er muss also messen, welche Spannung am analogen Pin anliegt. Der Befehl dafür muss in den `loop ()` - Teil geschrieben werden:

```
__________ = analogRead (_______);
```

Hier muss die Variable stehen, in der das Ergebnis gespeichert wird.

Hier wird angegeben, an welchem analogen Pin die Spannung gemessen werden soll.

Du hast das Programm erfolgreich installiert, aber es passiert nichts? Keine Sorge! Der Arduino misst die Temperatur. Er hat nur noch keine Möglichkeit uns das Ergebnis mitzuteilen. Schließlich hat er keinen Bildschirm, um uns das Ergebnis anzuzeigen. Oder doch?

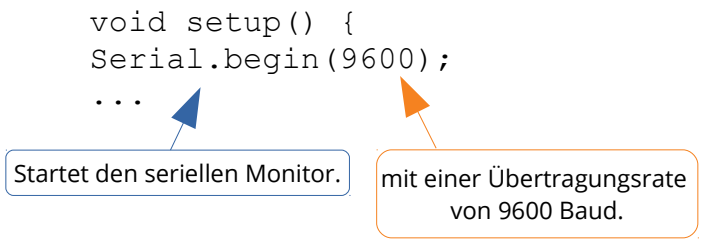
AUFGABE 4 – MESSERGEBNIS ANZEIGEN

Benutze den seriellen Monitor, um dir das Ergebnis der Temperaturmessung anzeigen zu lassen. Teste dein Programm, indem du einen Gegenstand vor die Augen des Sensors stellst.

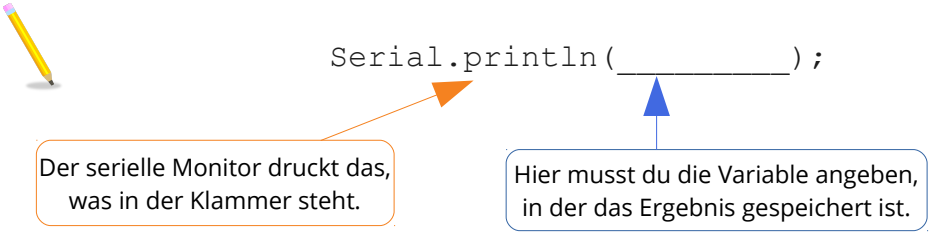
WAS IST EIN SERIELLER MONITOR?

Tatsächlich hat der Arduino einen Bildschirm, auf dem er uns Ergebnisse anzeigen kann. Diesen kannst du öffnen, wenn du in der Programmierumgebung rechts oben auf *Serieller Monitor* klickst.
Wie immer macht der Arduino aber nichts alleine. Du musst ihm also befehlen, das Ergebnis anzuzeigen.

1. Seriellen Monitor starten
   Damit der Arduino weiß, dass du den seriellen Monitor benutzen willst, musst du am Anfang des `setup()`-Teils folgenden Befehl schreiben:



2. Ergebnis durch seriellen Monitor anzeigen
   Um das Ergebnis sehen zu können, muss der serielle Monitor es nach der Messung „drucken“. Dafür brauchst du folgenden Befehl:



3. Installieren und testen
   Installiere das Programm auf dem Arduino. Öffne dann den seriellen Monitor in der Programmierumgebung. Beobachte die Messergebnisse. Verändere die Temperatur indem du zum Beispiel den Sensor durch Berühren deines Fingers erhitzen. Kannst du die Veränderung auch auf dem seriellen Monitor sehen?

4. Pausen zwischen den Messungen
   Wie du vielleicht schon bemerkt hast, sieht man innerhalb kurzer Zeit hunderte Messergebnisse. Der Arduino arbeitet sehr schnell und führt den Befehl zum Messen immer wieder aus. Du brauchst natürlich nicht alle Messergebnisse. Befiehl dem Arduino deswegen nach jeder Messung eine Pause zu machen.

   Installiere das Programm auf dem Arduino und öffne wieder den seriellen Monitor. Verändere die Länge der Pause, bis du die Messergebnisse auch alle lesen kannst.

AUFGABE 5 - ERGEBNISSE UMRECHNEN

Hmm... Irgendwie wirken die Messergebnisse komisch. Tatsächlich misst der Temperatursensor nicht in °C, sondern in Spannung. Mit der Zahl können wir aber so noch nichts anfangen. Deswegen ist deine nächste Aufgabe, das Messergebnis in °C umzurechnen.

1. Temperatur-Variable definieren

   Definiere eine Variable, die das Ergebnis der Umrechnung speichert. Das Ergebnis kann auch eine Kommazahl sein. Deswegen ist es wichtig, dass du dieses Mal nicht `int`, sondern `float` verwendest. Du brauchst ihr noch keine Zahl zuordnen. Das kann zum Beispiel so aussehen:

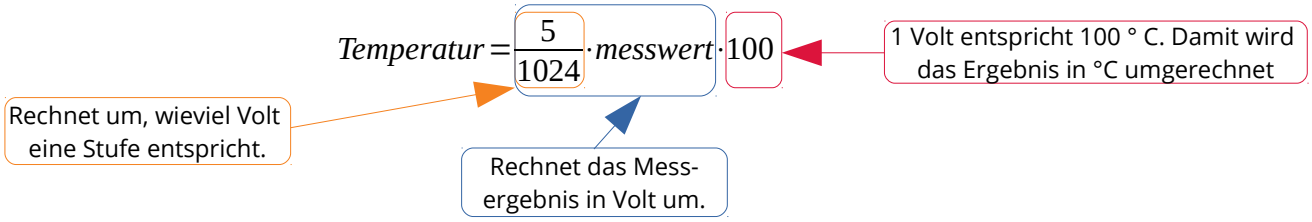
```
float temperatur ;
```

WIE FUNKTIONIERT EIN TEMPERATURSENSOR?

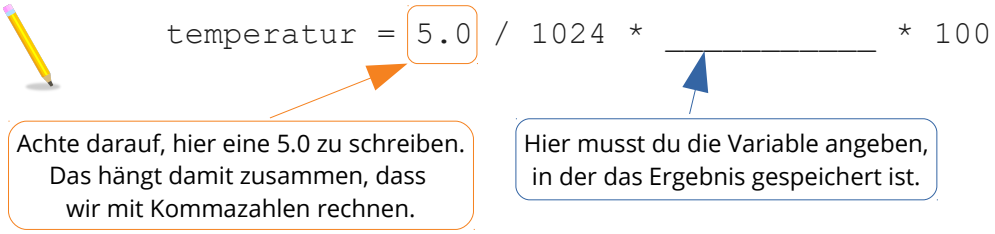
Der Temperatursensor ist mit dem Arduino über einen Schaltkreis verbunden. Die Kabel sind die einzige Verbindung über die sie Informationen austauschen können. Der Sensor schickt das Ergebnis seiner Messung als elektrisches Signal. Je höher die Temperatur ist, desto größer ist die Spannung auf dem Kabel. Die Spannung ist zwischen 0 und 5 Volt groß. Der Arduino misst die Spannung am Pin. Er muss die Information in eine Form bringen, die für den Computer geeignet ist. Deswegen speichert er das Ergebnis in einer von 1024 Stufen. Wenn du eine 0 auf dem seriellen Monitor als Ergebnis bekommst, heißt das, es ist 0 °C. Eine 1023 wäre 500°C.

2. Umrechnen

   Die Formel zum Umrechnen ist folgende:



Füge die Formel in deinen Code ein, um das Messergebnis umzurechnen.



Rechne mithilfe der Formel das Ergebnis nach der Messung um. Speichere das Ergebnis der Formel in der Temperatur-Variable.

3. <u>Temperatur drucken</u>
   Bisher steht noch im Code, dass das Messergebnis (in Stufen) gedruckt werden soll. Ändere den Code so ab, dass stattdessen die Temperatur in °C gedruckt wird.

AUFGABE 6 – FARBEN FÜR TEMPERATUREN AUSWÄHLEN

Wenn die Temperatur auf dem seriellen Monitor erscheint, kannst du zur letzten Aufgabe kommen. Jetzt sollst du die Farben für die Temperaturen auswählen.

1. Zuordnung der Farben
   Mache dir zuerst einen Plan, bei welcher Temperatur welche Farbe leuchten soll. Es wird schwierig sein, eine genaue Temperatur zu messen. Gebe deswegen in der Tabelle an, bis wieviel Grad die Farbe gelten soll. Fange in der Tabelle links mit der kältesten Temperatur an. Lasse sie Schritt für Schritt wärmer werden. Für Farbe 5 musst du keine Maximaltemperatur angeben. Warum, siehst du im nächsten Schritt. Zur Erinnerung, dir stehen die Farben rot – grün – blau – gelb – pink – türkis – weiß zur Auswahl.

	Farbe 1	Farbe 2	Farbe 3	Farbe 4	Farbe 5
gilt für Temperatur bis Farbe					X

2. Wenn – Ausdrücke
   Um deine Liste zu programmieren brauchst du **Wenn** und **Sonst, wenn –** Ausdrücke. Als Text ausgedrückt:
   Wenn Temperatur kleiner ist, als _____ Grad, soll Farbe _____ leuchten. **Sonst, wenn** die Temperatur kleiner ist, als _____ Grad, soll Farbe _____ leuchten. **Sonst** soll Farbe _____ leuchten.

   Füge die Ausdrücke in den loop()-Teil ein, nachdem die Temperatur berechnet wurde.

```
if ( temperatur < __________ ) {
Farbe 1 leuchtet }
else if ( temperatur < _________ ) {
Farbe 2 leuchtet }

else {
Farbe 5 leuchtet }
```

   Füge hier die kleinste Temperatur deiner Tabelle ein.

   Und hier die Temperatur der Farbe 2

   Für die anderen Farben kannst du an dieser Stelle weitere **Sonst, wenn** – Ausdrücke hinzufügen. Die Farbe im letzten Sonst-Ausdruck leuchtet, wenn die Temperatur größer ist, als alle anderen Zahlen. Deswegen brauchst du hier keine Maximaltemperatur angeben. Achte auch darauf, dass du die Reihenfolge der Farben aus der Tabelle beibehältst!

3. Diskolichtprogramm einfügen
   In dem Code zum Diskolicht, hast du schon das Leuchten der Farben programmiert. Kopiere aus dem Diskolicht-Code die Variablen-Definitionen und füge sie vor dem `setup()`-Teil ein. Kopiere auch die Befehle zum `pinMode()` und füge sie im `setup()`-Teil ein.

4. Farbbefehle

Im Diskolichtprogramm hast du bereits alle Farben programmiert. Suche die Farbe im Diskolichtprogramm, die du nun auch im Temperaturprogramm benutzen möchtest. Kopiere die jeweiligen Befehle zum Anschalten der benötigten Farben. Die Pause und das Ausschalten brauchst du nicht zu kopieren. Setze die kopierten Befehle in den richtigen **Wenn, dann** – Ausdruck ein.
Das kann zum Beispiel so aussehen:

Hier müsste noch eine Temperatur rein-geschrieben werden

```
if ( temperatur < __________ ) {
digitalWrite (blau, HIGH};
digitalWrite(rot, HIGH);
}
```

5. Ausschalten der Farben

Die angeschalteten LEDs müssen auch wieder ausgeschaltet werden, damit je nach Temperatur auch wirklich unterschiedliche Farben leuchten. Das Ausschalten ist dieses Mal aber anders als im Diskolichtprogramm. Und zwar hast du in deinem Programm nach der Temperaturmessung schon eine Pause eingebaut. Diese sollst du zum Leuchten der LEDs nutzen und danach die LEDs ausschalten. Unabhängig von der Farbkombination, sollst du immer alle drei LEDs der RGB-LED ausschalten. Den Befehl zum Ausschalten kennst du bereits.

```
void loop() {
messen
Pause
Umrechnung des Ergebnisses
Drucken der Temperatur
```

Füge hier das Ausschalten aller drei LEDs ein.

```
if ( temperatur < __________ ) {
Farbe 1 leuchtet }
...
else {
Farbe 5 leuchtet }
}
```

6. Programm installieren und testen
   Installiere das Programm auf dem Arduino. Teste es, indem du die Temperatur des Sensors veränderst. Die Farbe des Lichts sollte sich verändern. Falls nicht, kannst du versuchen die Maximaltemperaturen der Farben zu verändern.

Super! Mithilfe deines Programms kann man jetzt Temperaturen sehen.

Fotos: RWTH Aachen, InfoSphere
Screenshots: fritzing electronics made by easy und Arduino IDE 1.8.12 (windows)
Alle weiteren Grafiken: Patrick Binkert, EduInf@TUD