Station 4 – Farbthermometer inkl. Einstieg

Temperatur zum Leuchten bringen

Ihr habt euch also 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 aber ändern!

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

Abb. 1: Einige LEDs

Abb. 2: Ein Brunnen mit LEDs verbaut

Der Einstieg

Bevor ihr euch an das eigentliche Projekt setzt, werdet ihr zunächst einige Bauteile genauer betrachten und lernen wie ihr richtig damit umgeht.

Für diesen Einstieg wird folgendes benötigt:

- Der Arduino-Mikrocontroller inkl. Steckbrett
- 3x 220Ω Widerstand (markiert mit **rot-rot-braun-gold**)
- Ein 100kΩ-Widerstand (markiert mit **braun-schwarz-gelb-gold**)
- Eine RGB-LED
- Ein Taster
- Jede Menge Kabel

Abb. 3: RGB-LED

Wichtig: *Taster und 100kΩ Widerstand findet ihr in dem Kästchen für Station 0, RGB-LED und 220Ω Widerstände in dem Kästchen von Station 4.*

Der Arduino

Im Mittelpunkt aller heutigen Projekte steht der Arduino-Mikrocontroller, den ihr unten schematisch dargestellt seht:

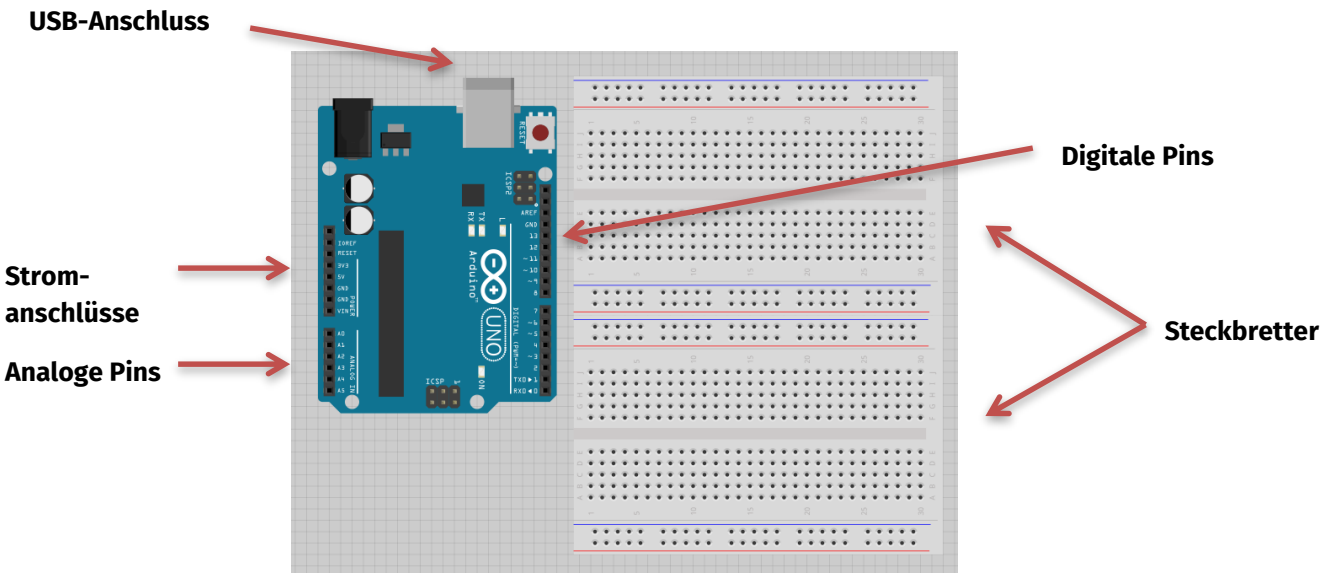

Abb. 4: Arduino-Board mit Steckbrett

Zuerst eine kurze Erläuterung der wichtigsten Komponenten:

USB-Anschluss:

Damit verbindet ihr den Arduino später mit dem Computer, um euer Programm zu übertragen. Außerdem dient dieser Anschluss als Stromversorgung.

Stromanschlüsse:

Manche Bauteile müssen mit Strom versorgt werden, auch wenn es nicht explizit drauf steht, merkt euch einfach:

GND = – (Minuspol) **5V = + (Pluspol)**

Digitale Pins:

Diese können entweder den Zustand AN (HIGH) oder AUS (LOW) annehmen. Digitale Anschlüsse können als Eingänge (z.B. für Schalter) oder Ausgänge (z.B. für Lampen) benutzt werden.

[**Hinweis:** *Bitte benutzt nur die Pins 2-13. Die anderen beiden haben eine Sonderfunktion.*]

Analoge Pins:

Diese werden im Gegensatz zu den digitalen Pins nur als Eingänge benutzt. Im Gegensatz zu den digitalen Pins können sie nicht nur zwei, sondern 1024 verschiedene Werte unterscheiden. Der Wertebereich geht von 0 (das entspricht 0V) bis 1023 (das entspricht 5V).

Das Steckbrett

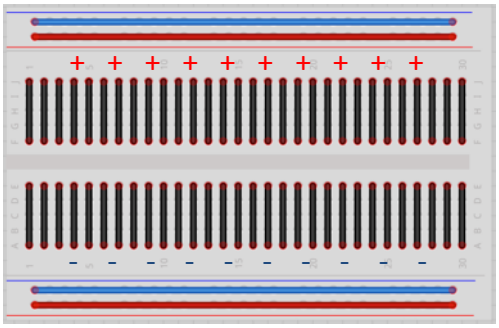

Abb. 5: Steckbrett mit gezeichneten Verbindungen

Auf der obigen Abbildung seht ihr das Innere des Steckbretts dargestellt. Hier könnt ihr sehen wie die Löcher auf der Oberseite des Steckbretts im Inneren miteinander verbunden sind.

Wichtig: *Bevor ihr etwas an der Verkabelung ändert, trennt den Arduino immer vom Strom! Zieht dafür einfach das USB-Kabel aus dem Computer.*

RGB-LEDs

RGB-LEDs sehen zwar aus wie weiße LEDs (schaut euch die kleine LED im Kasten der Station 0 dafür kurz an), aber in Wirklichkeit verstecken sich darin drei verschiedene LEDs! Nämlich eine **R**ote, eine **G**rüne und eine **B**laue LED, daher auch der Name RGB-LED. Wie ihr sehen könnt hat die normale LED zwei Beinchen, ein Beinchen für den Minus-Pol und ein Beinchen für den Plus-Pol. Das wirft natürlich die Frage auf, warum RGB-LEDs keine sechs Beinchen, sondern nur vier haben. Das liegt daran, dass der Hersteller die Minus-Pole schon im Gehäuse verbunden hat.

Hier muss das längste Beinchen mit dem Minus-Pol verbunden werden. Legt ihr die LED so vor euch, dass das längste Beinchen an dritter Stelle liegt, könnt ihr im Bild ablesen, welches Beinchen zum Plus-Pol welcher Farbe gehört.

Diese drei Farben reichen, um alle sichtbaren Farben darzustellen.

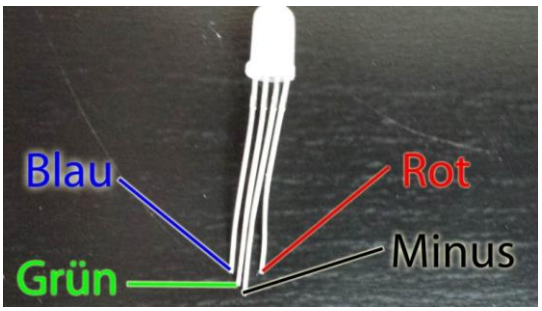

Abb. 6: RGB-LED

Es leuchtet

Genug geredet, jetzt wird gebaut!
Zuerst sollt ihr testen, ob die RGB-LED überhaupt funktioniert.

Eure ToDos

1. Nehmt ein rotes Kabel und verbindet den 5V – Anschluss mit der Plus-Leiste des Steckbretts.
2. Nehmt ein blaues Kabel und verbindet wiederum ein GND – Pin des Arduino mit dem Minus-Pol.
3. Steckt die RGB-LED jetzt so in das Steckbrett, dass der Minus-Pol in der Minus-Leiste des Steckbretts steckt. Die anderen Pins kommen jeweils in eine eigene Reihe des Steckbretts.
4. Anschließend werden die Reihen jeweils mit einem Widerstand (220Ω) mit der anderen Seite des Steckbretts verbunden. Dort könnt ihr auch drei gelbe Steckkabel einstecken, das obere Ende hängt aber erst mal in der Luft.

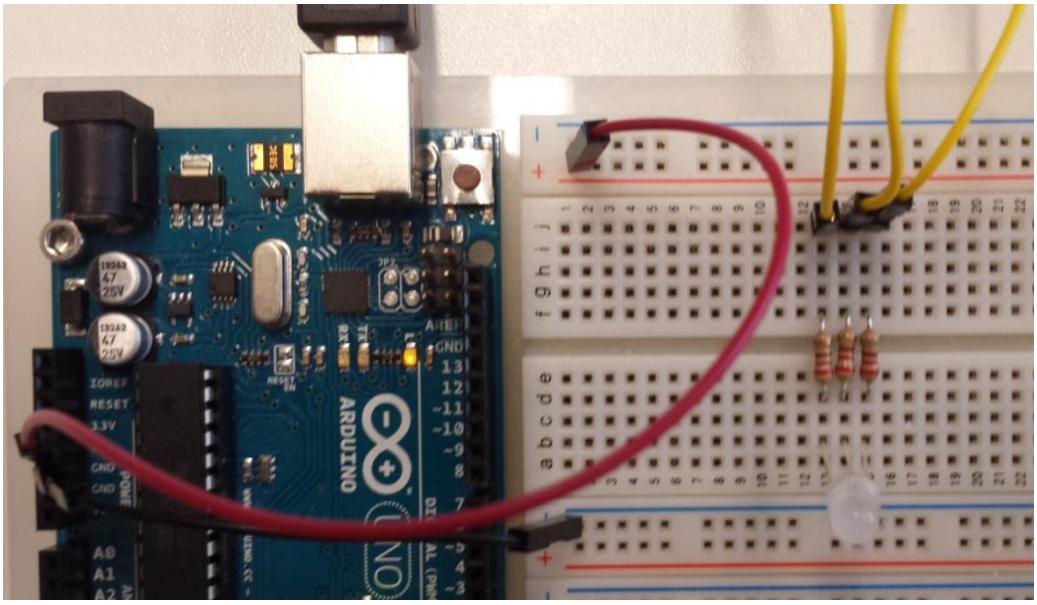

Abb. 7: Komplette Schaltung

Nachdem ihr den Arduino mit dem Computer (per USB-Kabel) verbunden habt, könnt ihr auch schon loslegen! Verbindet die Steckkabel jeweils mit der Plus-Leiste und notiert unten eure Beobachtungen.

Verbundenes Kabel	Farbe der LED
Links	
Mitte	
Rechts	
Links & Mitte	
Links & Rechts	
Mitte & Rechts	
Links & Mitte & Rechts	

Wie kann es sein, dass das Ergebnis überhaupt nichts mit dem zu tun hat, was ihr im Kunstunterricht über Farbmischung gelernt habt?

Additive Farbmischung

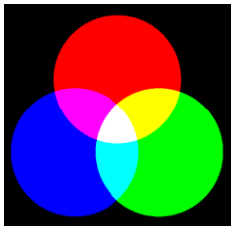

Abb. 8: Farbmischung

Aus dem Kunstunterricht kennt ihr ja den guten alten Malkasten. Wenn ihr dort die Farben **B**lau und Gelb mischt, erhaltet ihr die Farbe **G**rün. Das funktioniert, weil sich das weiße Licht aus allen Farben zusammensetzt. Die Farben aus dem Malkasten filtern bestimmte Anteile aus dem Licht, so dass bei Mischung mehrere Anteile wegfallen. Mischt man alle Farben, so wird jeder Anteil gefiltert, das Resultat ist schwarz. Im Gegensatz zu diesem sogenannten **subtraktiven Farbmodell** wird bei RGB-LEDs kein Licht weggenommen, sondern wird neues dazu gemischt. Mischt man alle Anteile, so erhält man hier weißes Licht. Die Grundfarben hier sind 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.

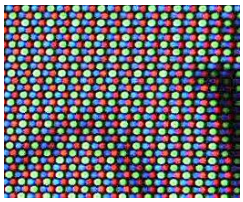

Abb. 9: Monitorbild

Station 4 – Farbthermometer inkl. Einstieg

Natürlich macht es Sinn, die RGB-LED mit dem Arduino zu steuern, statt immer Kabel umstecken zu müssen. Das sollte nun sinnvoll programmiert werden. Öffnet hierzu die Arduino-Software:

Abb. 10: Screenshot der Arduino-Software

Zunächst eine kurze Erläuterung der Bedienelemente:

Mit diesen Knöpfen überprüft ihr euer Programm. Mit dem linken Knopf kompiliert ihr euer Programm, und mit dem rechten Knopf wird das Programm sowohl kompiliert als auch an den Arduino gesendet.

[**Hinweis:** *Unter Umständen könnte dies etwas dauern!*]

Dieser Knopf legt einen neuen Sketch an.

Sketch ist der Name eines Programms im Arduino

Diese Knöpfe öffnen bzw. speichern Sketches.

Station 4 – Farbthermometer inkl. Einstieg

Programmieren

Den meisten von euch ist bestimmt die Programmiersprache Java bekannt. Der Arduino wird auf eine ähnliche Weise programmiert. Im Folgenden seht ihr das Grundgerüst, welches für jedes Programm benötigt wird:

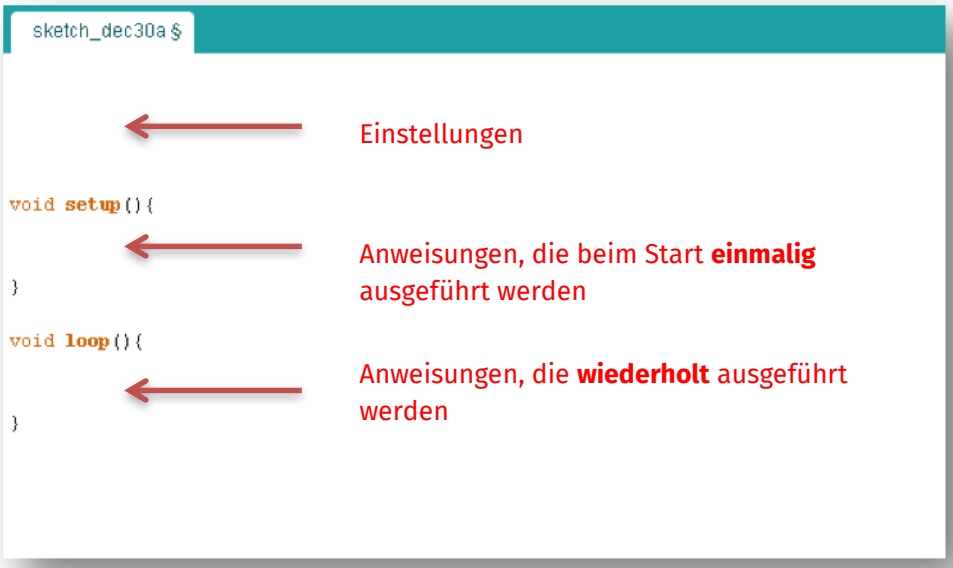

Abb. 11: Aufbau eines Arduino-Sketch

Für eine bessere Lesbarkeit könntet ihr (beispielsweise wie auch in Java) weitere Methoden und Unterprogramme schreiben, die dann in der loop() oder auch in setup() aufgerufen werden.

Um die RGB-LED nun über euer Programm zu steuern, verbindet ihr zunächst die gelben Kabel nicht mehr mit der Plus-Leiste, sondern mit den digitalen Pins 9 (Blau), 10 (Grün) und 11 (Rot).

9, 10 und 11 sind Spezial-Pins, zu erkennen an der kleinen Schlange ~

Station 4 – Farbthermometer inkl. Einstieg

Eure ToDos

1. Initialisiert am Anfang eures Sketches (Einstellungen) drei int-Variablen, in der die Nummer der Pins (Pin-Name), an dem die RGB-LED angeschlossen sind, gespeichert werden.
2. Definiert diese Pins in setup() als Ausgänge, benutzt dafür die Methode pinMode(<Pin-Name>, <Pin-Typ>).
   [Hinweis: Ihr habt drei Pins, also müsst ihr die Methode auch dreimal aufrufen. Denkt außerdem daran, dass in <Pin-Typ> festgelegt wird, ob der jeweilige Pin als Eingang (INPUT) oder Ausgang (OUTPUT) verwendet wird.]
3. Schaltet mit der Methode digitalWrite(<Pin-Name>, <Zustand>) den Strom an euren Pins ein.
   [**Hinweis:** *In <Zustand> wird festgelegt, ob der Pin an (HIGH) oder aus (LOW) ist.*]

Deklariert also die Variablen für eure Pins am Anfang eures Sketches, schaltet den pinMode() in setup() auf OUTPUT und startet mit digitalWrite() den Strom für diese Pins.

Speichert euren Code ab. Kompiliert diesen und schaut was passiert.

Nun sollte die RGB-LED wieder leuchten.

In eurem nächsten Schritt werdet ihr euer Programm so verändern, dass die RGB-LED beim Leuchten kurze Pausen macht. Hierfür benötigt ihr eine weitere Methode

delay(<Zeit>).

Die Zahl, die ihr dieser Methode übergeben könnt, entspricht der Zeit in Millisekunden.

Eure ToDos

1. Erweitert euren Code, indem ihr die oben beschriebene Methode delay(<Zeit>) einbaut.

Das Leuchten der RGB-LED sollte nun immer kurz pausieren.

Wichtig: *Falls es nicht geklappt hat, überlegt euch, wie oft* delay(<Zeit>) *benutzt werden muss.*

Mehr als 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 einfach zwei oder auch drei der eingebauten LEDs gemeinsam einschalten.

Eure ToDos

1. Speichert euren Sketch unter einem neuen Namen ab, damit ihr euer Programm einfach erweitern könnt.
2. Verändert ihn nun so, dass eure RGB-LED ein Farbenspiel aus rot, grün, blau, gelb, pink, türkis und weiß aufführt.
3. Testet euer Programm und das Speichern nicht vergessen.

All diese Farben könnt ihr später nutzen, um verschiedene Temperaturen darzustellen.
Doch davor wollen wir versuchen das Licht auch mal auszuschalten.

Der Taster


Abb. 12: Taster

Wollt ihr die RGB-LED vielleicht auch manuell ein- und ausschalten? Dafür benötigt ihr einen Taster.

Ein Taster, das ist ein Bauelement, das Strom leitet, solange der Knopf runtergedrückt ist. Nimmt man den Finger weg, fließt auch kein Strom mehr. Ihr findet den Taster in eurer Box im Kästchen der Station 0.

Ein paar Kleinigkeiten sind beim Einbau eines Tasters jedoch zu beachten. Die eine Seite des Tasters kommt an den Plus-Pol, die andere Seite wird mit einem der digitalen Pins des Arduino verbunden. Zusätzlich muss diese Seite aber noch mit einem großen Widerstand (**100 kΩ, braun-schwarz-gelb-gold**) mit dem Minus-Pol verbunden werden. Wenn euch interessiert warum, schaut auf dem Plakat Elektrotechnik unter Pull-down nach!

Station 4 – Farbthermometer inkl. Einstieg

Wir wollen also die RGB-LED nun ein- und ausschaltbar machen.

Wichtig: *Bevor ihr etwas an der Verkabelung ändert, trennt den Arduino immer vom Strom! Zieht dafür einfach das USB-Kabel aus dem Computer.*

Eure ToDos

1. Schließt den Taster wie oben beschrieben an euer Steckbrett an.
2. Legt eine neue Variable für den digitalen Pin an.
3. Macht diesen Pin zu einem Eingang (Hinweis: INPUT).
4. Verwendet für die Abhängigkeit (ob der Pin ein- oder ausgeschaltet ist) if-else-Anweisungen [**Hinweis:** *Überprüft, ob der Rückgabewert von digitalRead() HIGH oder LOW ist Testet euer Programm und das Speichern nicht vergessen.*]

Wichtig: *Wie der Großteil der Arduino-Sprache sehen auch die if-else-Anweisungen wie in Java aus.*

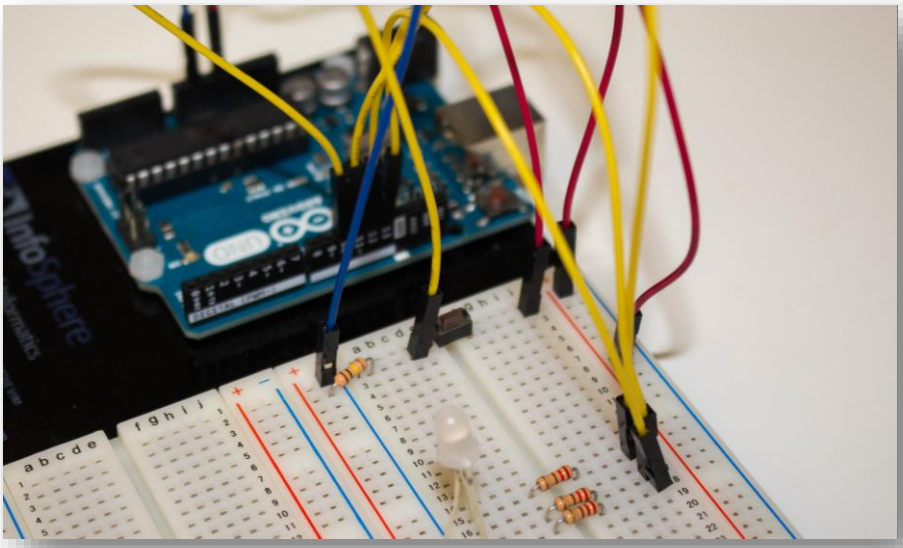

Abb. 13: Komplette Schaltung

Hervorragend, der Einstieg ist geschafft! Ihr wisst nun wie man digitale Pins auslesen und ansteuern kann. Außerdem seid ihr jetzt mit der RGB-LED vertraut und könnt nun den Farbthermometer

programmieren! Eine Sache fehlt euch aber noch: ein Temperaturfühler, damit ihr auch genau wisst, welche Temperatur herrscht.

Die Temperatur

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

LM35

Der LM35 hat drei Beinchen. Die komplexe Schaltung im Inneren muss mit Strom versorgt werden, hierfür dienen die beiden äußeren Beinchen. Zeigt die flache Seite zu euch, so muss das linke Beinchen mit dem 5V-Anschluss (also der Plus-Leiste) verbunden werden, das rechte Beinchen verbindet ihr mit GND (also der Minus-Leiste).

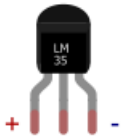

Abb. 14: Temperaturfühler

Wichtig: *Wird der LM35 sehr heiß, zieht den Stecker, denn ihr habt ihn dann falsch herum eingebaut. Lasst ihn ggf. etwas abkühlen.*

Der mittlere Pin des LM35 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 LM35 mit einem analogen Eingang auf dem Arduino.

Station 4 – Farbthermometer inkl. Einstieg

Analoge Pins

Bisher kennt ihr nur **digitale Pins**, über die z.B. der Taster angeschlossen wurde. An diesen Pins können nur **binäre Werte** ein- und ausgelesen werden – also ein/aus, d.h. 1/0, oder eben die bekannten Werte HIGH und LOW für eine hohe bzw. niedrige Spannung. Ein Temperaturmesser kommt mit zwei Werten aber nicht aus, 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 **Bereich** gemessen werden kann.

Zuallererst werdet ihr die **Daten**, die der LM35 misst, über ein kleines Programm einlesen und euch auf der seriellen Konsole ausgeben lassen.

Die serielle Konsole schickt Daten über das Kabel vom Arduino zum Computer. Ihr startet diese Übertragung indem ihr in setup() die Methode

Serial.begin(9600);

aufruft. Die übergebene Zahl 9600 gibt die Geschwindigkeit an mit der die Daten über USB gesendet werden. Um euch die Daten auch ausgeben zu lassen, braucht ihr in der loop() eine weitere Methode, nämlich

Serial.println(<Eure Daten>);.

Ihr übergebt dieser Methode einfach eine Variable oder natürlich auch einen String (die Anführungszeichen dabei nicht vergessen). Mit diesem Button  (oben rechts in der Bedienleiste) öffnet ihr die serielle Konsole.

Nun ist eure nächste Aufgabe, mit dem LM35 Werte zu messen. Hierfür könnt ihr euer bisheriges Programm einfach erweitern.

Eure ToDos

1. Legt eine Variable vom Typ int für den Pin des Temperatursensors an.
2. Legt eine weitere Variable vom Typ int an, um dort die gemessenen Sensorwerte abspeichern zu können.
3. Nun zu setup():
   a. Startet die Übertragung zur seriellen Konsole.
4. Die loop():
   a. Die gemessenen Werte sollten nun eingelesen werden. Da ihr analoge Werte auslesen möchtet, wird der Befehl

      analogRead(<Pin-Name>);

      benötigt. Dieser liefert einen int Wert zurück, den ihr der oben angelegten Variable (aus Schritt 2) direkt zuweisen könnt.
   b. Lasst euch diese Werte auf der seriellen Konsole ausgeben und versucht dabei in der Ausgabe auch eine Beschreibung dessen anzuzeigen, was ausgegeben wird.
5. Es ist an der Zeit, den Sketch zu testen! Schließt den Arduino wieder an, führt das Programm aus und beobachtet die Werte.

Euch stehen also folgende Fakten zur Verfügung:

- Ein Sensorwert zwischen 0 und 1023
- Eine Skala der Spannung von 0 bis 5V
- Ein Sensor, der 10mV, also 0,01V 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.

Neben den Variablen für die Pins und dem Sensorwert braucht ihr noch zwei weitere Variablen. Diese sollen die Spannung und die Temperatur speichern. Um die Temperatur zu errechnen, müsst ihr wissen, wie viel Spannung am Sensor anliegt. Dafür müsst ihr den Sensorwert (der zwischen 0 und 1023 liegt) in die Spannung (diese geht von 0V bis 5V) umrechnen. Dies gelingt euch mit folgender Formel:

spannung = (5.0 / 1023) · <Sensorwert>;

[**Hinweis:** *Hier ist es wichtig statt „5“ eine „5.0“ zu schreiben, um nicht nur ganze Zahlen zu erhalten!*]

Eure ToDos

1. Achtet darauf, dass ihr bei einer Division häufig eine Dezimalzahl erhaltet, also speichert die Werte unter neuen Variablen vom Typ float.
2. Berechnet nun aus der Spannung die Temperatur.
3. Lasst euch diese beiden Werte im seriellen Monitor ausgeben.

Der Sensor liefert 0,01 Volt pro °C, und da ihr nun die Spannung kennt, könnt ihr diese durch den Umrechnungsfaktor 0,01 teilen und erhaltet dadurch die Temperatur.
[Hinweis: Die Division durch 0,01 entspricht der Multiplikation mit 100.]

Das waren nun doch ziemlich viele Schritte auf einmal, daher noch eine etwas kleinere Checkliste:

Eure Checkliste

1. Habt ihr alle Variablen eingefügt, die benötigt werden?
   [**Hinweis:** *3x LED-Pins, 1x analoger Pin, 1x Sensorwert, 1x Spannung, 1x Temperatur*]
2. Haben die Variablen den richtigen Typ und die richtigen Pin-Werte?
3. Nun zu setup():
   a. Habt ihr die richtige Methode (korrekt als Ein- bzw. Ausgang) für die einzelnen Pins aufgerufen?
   b. Habt ihr die serielle Konsole gestartet?
4. In der loop():
   a. Wird der Wert des analogen Eingangs gelesen?
   b. Habt ihr die Formeln für die Spannung und die Temperatur richtig übertragen?
   c. Werden diese Werte auch auf dem seriellen Monitor ausgegeben?

Abschließend sollte es reichen, wenn die Temperatur nur einmal pro Sekunde ausgegeben wird.
[**Hinweis:** *Kennt ihr den Befehl noch? Wenn nicht, schaut einfach auf Seite 8 nochmal nach.*]

Station 4 – Farbthermometer inkl. Einstieg

Es ist an der Zeit, den Sketch zu testen! Schließt den Arduino über den USB-Anschluss an den Laptop an und startet die Übertragung. Schaut euch auf dem seriellen Monitor die Werte an und vervollständigt diese Tabelle:

	Temperatur	**Spannung**	**Sensorwert**
Raumluft			
Finger			

Diese letzten beiden Zeilen könnt ihr nutzen, falls euch noch andere Ideen einfallen.

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 was ihr benötigt, um die Temperatur anzuzeigen.

Wie wäre es, wenn die LED bei Raumtemperatur blau leuchtet? Wenn es wärmer wird vielleicht über gelb nach rot wechselt? Nutzt die folgende Tabelle, um zu notieren, bei welcher Temperatur ihr welche Farbe anzeigen wollt! Vergesst nicht zusätzlich zu notieren, welche LEDs dafür leuchten müssen.

Temperatur	Farbe	R(ot)	G(rün)	B(lau)

Hierfür müsst ihr euren Sketch dann noch etwas erweitern.

Eure ToDos

1. Erstellt für jeden Temperaturbereich eine if-Anweisung. Dabei braucht der unterste und der oberste Bereich jeweils nur eine Bedingung, alle Bereiche dazwischen jedoch zwei.
   [**Hinweis:** *Ihr benötigt hier zwei Bedingungen, da diese Bereiche zwei Grenzen haben, also eine Bedingung für die Untergrenze und eine Bedingung für die Obergrenze. Der unterste bzw. oberste Bereich hat jeweils nur eine Grenze.*]
2. Lasst je nach Bereich die richtigen LEDs leuchten.
   [**Hinweis:** *Falls es nicht geklappt hat, habt ihr vielleicht die LEDs, die nicht benötigt werden, auch wieder ausgeschaltet?*]

Wenn alles geklappt hat, sollte es bei euch etwas wie in den beiden Bildern funktionieren!

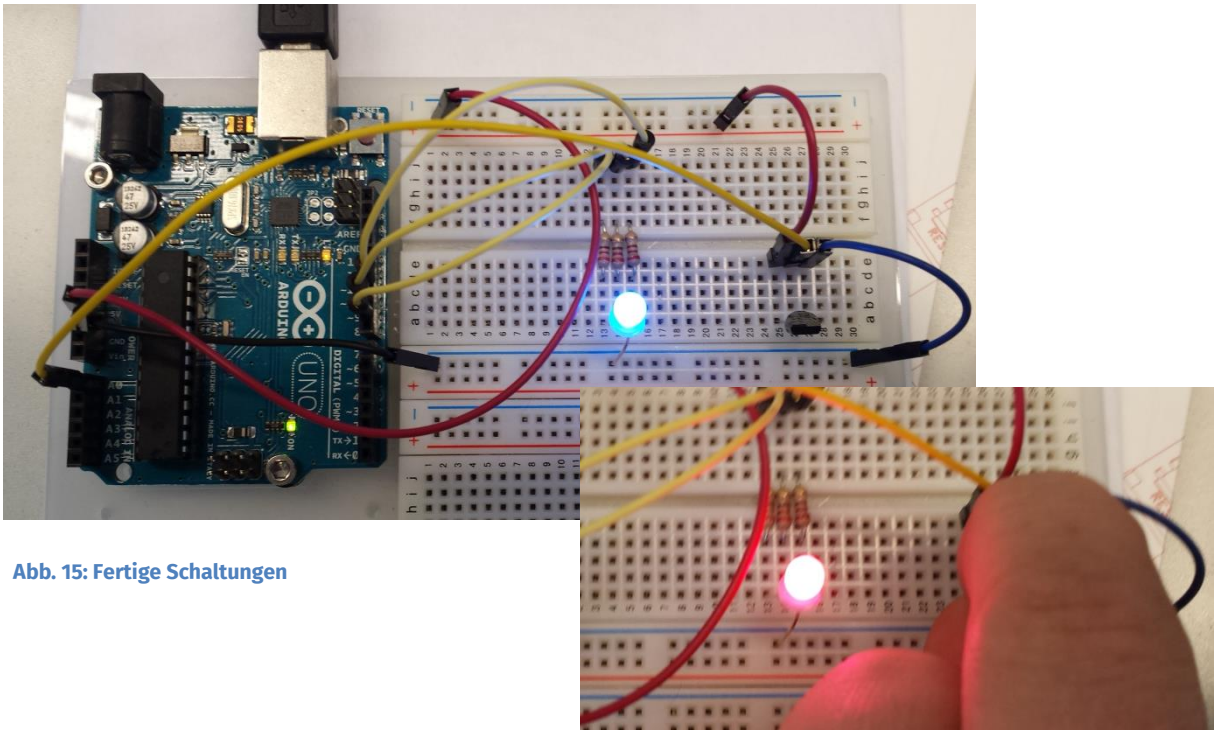

Abb. 15: Fertige Schaltungen

Alles geklappt? **Herzlichen Glückwunsch!** Wem das noch nicht genug ist, der kann sich noch an die Bonus-Aufgaben herantrauen. Fragt nach dem Bonus-Blatt.

Es lebe das Chaos… oder auch nicht!

Damit auch die nächste Gruppe alle Bauteile wiederfindet, würden wir uns freuen, wenn ihr so nett wärt und ihr die Bauteile in die zugehörigen Kästchen einsortiert.

Ins Kästchen zu **Station 0** gehören:

- der 100kΩ-Widerstand,
- der Taster.

Ins Kästchen zu **Station 4** gehören:

- die 220Ω-Widerstände,
- der Temperaturfühler,
- die RGB-LED.

Vielen Dank!!! ☺

Quellenverzeichnis:

Abb. 1 – *Quelle: wikipedia.org, Autor: Akimbomidget (CC BY-SA 2.5)*

Abb. 2 – *Quelle: wikipedia.org, Autor: Diliff (CC BY-SA 3.0)*

Abb. 8 – *Quelle: wikipedia.org, Autor: Quark67 (CC SA 3.0)*

Abb. 9 – *Quelle: wikipedia.org, Autor: Ernst Schütte (CC SA 3.0)*

Abb. 6, 7, 12, 13 - *Quelle: InfoSphere*

Abb. 3, 4, 5, 14 - *Quelle: Screenshots der Fritzing-Software (http://fritzing.org)*

Abb. 10, 11 - *Quelle: Screenshots der Arduino-Software (http://www.arduino.cc)*

Alle weiteren Grafiken/Icons - Quelle: InfoSphere