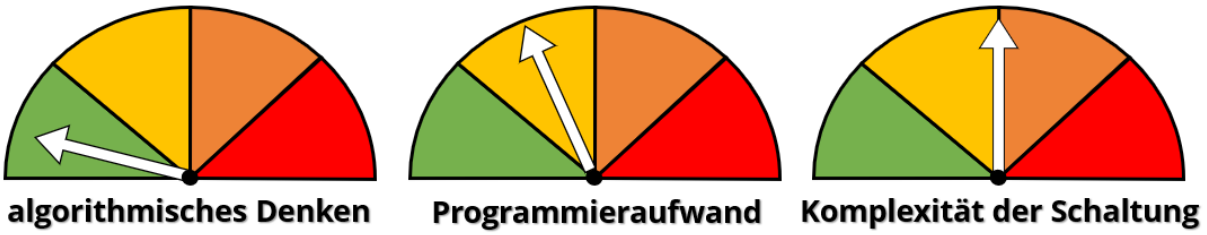
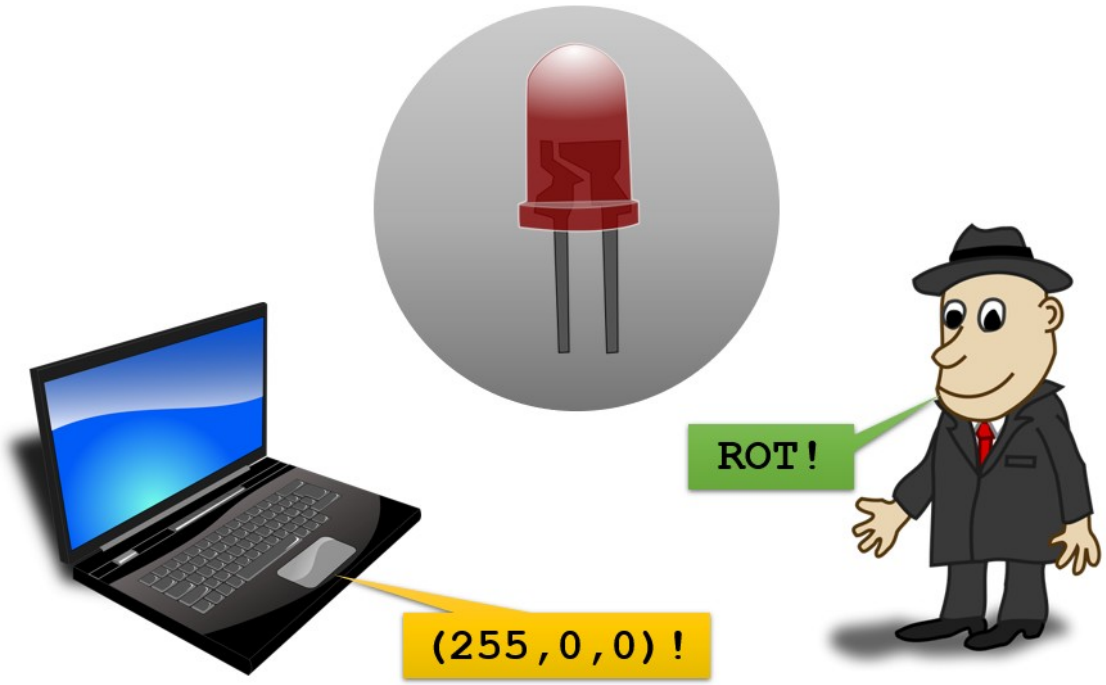
TECHNISCHE UNIVERSITÄT DRESDEN

Arduino Uno

Station 6 | Du siehst rot? – Ich sehe nur (255,0,0)!





Professur für Didaktik der Informatik

SYLBER Synergetische Lehrerbildung im exzellenten Rahmen

Das Maßnahmenpaket „TUD-Sylber² – Synergetische Lehrerbildung im exzellenten Rahmen“ wird im Rahmen der gemeinsamen „Qualitätsoffensive Lehrerbildung“ von Bund und Ländern aus Mitteln des Bundesministeriums für Bildung und Forschung gefördert.

GEFÖRDERT VOM Bundesministerium für Bildung und Forschung

ZIEL DER STATION

Was wäre die Welt nur ohne Farben? Was für viele von uns eine ziemlich triste Vorstellung wäre, ist für Farbenblinde leider alltägliche Realität. Selbstverständlich könnte man jetzt annehmen, dass es bereits Computerchips gibt, welche diese Fähigkeit ersetzen. Leider ist die Wissenschaft in dieser Hinsicht noch nicht soweit. Du aber darfst dir heute bereits einmal Gedanken machen, wie ein Computer überhaupt Farben erkennen kann.

Wir schauen uns zunächst eine andere Art von LEDs an, diese ist in der Lage verschiedene Farben darzustellen und hat auch zwei Pins mehr, als die uns bis jetzt bekannten. Dann betrachten wir einen Farbsensor und beschäftigen uns mit dem Erkennen von Farben. Wenn du das geschafft hast, kannst du dir dann aussuchen, wie du die erkannte Farbe darstellen willst:

- **Variante (1):** Darstellung mit einem kleinen Servomotor, welcher einen Pfeil in Richtung der Farbe ausrichtet
- **Variante (2):** Ausgabe über eine mehrfarbige LED

Abbildung 1: RGB-LED

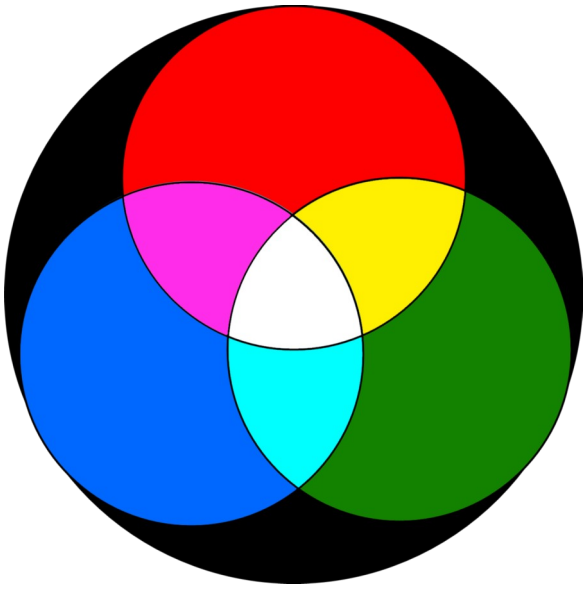
DAS RGB-FARBMODELL

Als Grundlage benötigen wir dazu Kenntnisse im Bereich der Farbmodelle. Es wird prinzipiell zwischen dem **RGB- und dem CMYK-Farbmodell** unterschieden. CMYK steht beispielsweise für die Farben **C**-Cyan, **M**-Magenta, **Y**-Yellow und **K**-Key (black), dies findet zum Beispiel bei einem Tintenstrahldrucker Anwendung. Wir brauchen heute allerdings das RGB-Farbmodell, wobei die Buchstaben dabei folgende Bedeutung haben:

R – **red** / rot **G** – **green** / grün **B** – **blue** / blau

Aus diesen drei Farben werden dann alle anderen Farben gebildet durch „additive Farbmischung“. Klingt schwer? Ist es nicht. Additiv kannst du dir mit dem artverwandten Wort „Addition“ erklären und besagt nichts anderes, als das die Farben addiert werden.

FARBWERTANGABE



Anhand der nebenstehenden Grafik wollen wir uns das Zustandekommen der Farben einmal näher anschauen.

Was fällt auf? In der Mitte überschneiden sich alle drei Lichtkreise – die Mischung erscheint Weiß. Die Farbe Schwarz wird durch die Dunkelheit im umgebenden Raum repräsentiert. Überschneiden sich immer nur zwei Kreise, so werden diese wie rechts zu sehen gemischt.

Die Angabe der resultierenden Farben erfolgt über sogenannte „Zahlen-Tupel“. Wir haben drei Zahlen, welche jeweils den Anteil der jeweiligen Farbe angeben. Diese haben jeweils einen **Wertebereich von 0 bis 255**. Pro Farbangabe können wir also 256 verschiedene Abstufung abbilden und das ganze drei mal, für die Farben rot, grün und blau. In der Summe macht dies rund **16,7 Millionen verschiedene Farben**. Das sieht dann an Beispielen wie folgt aus:

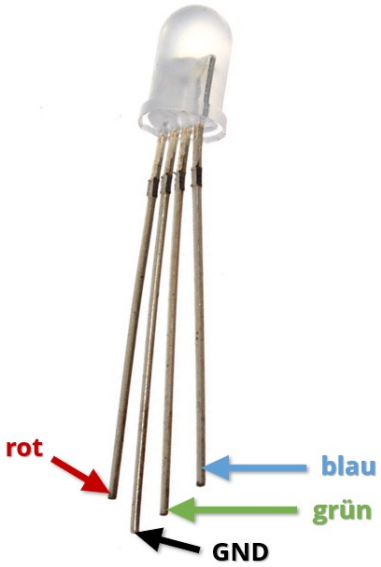
(roter Anteil, grüner Anteil, blauer Anteil)

Beispiele:

rot	(255,0,0)
grün	(0,255,0)
blau	(0,0,255)
gelb	(100,255,0)
blau	(0,255,255)
pink	(125,0,125)

DIE RGB-LED

Warum die ganze Theorie? - Wir brauchen dieses Grundwissen zur Benutzung der RGB-LED, da wir dort die Farben nach dem gleichen Verfahren darstellen wollen. Zunächst wird dir erstmal gezeigt, wie du die LED anschließen musst.



Die Farben werden nicht durch eine LED allein erzeugt, sondern auch hier sind drei verschieden farbige LEDs verbaut. Soweit so gut, aber müssten es dann dann nicht 6 Pins sein? Richtig erkannt, allerdings ist das `GND` für alle über das Gehäuse verbunden. Somit reichen 4 Anschlüsse.

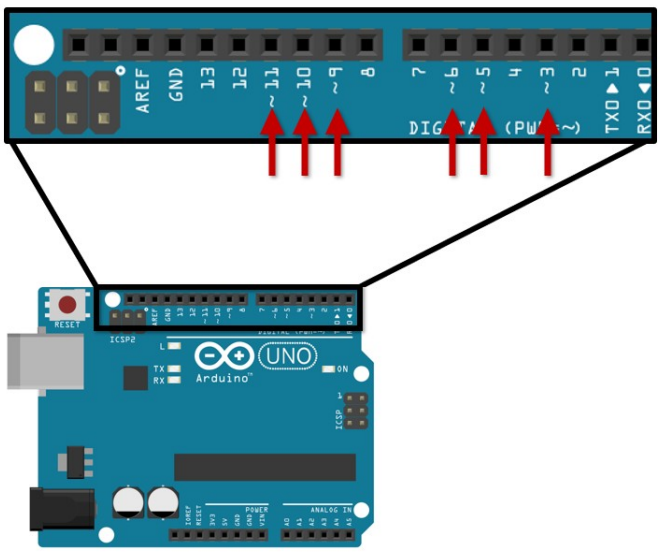
AUFGABE 1 – RGB-LED TESTEN

Verbinde zunächst den `GND`-Anschluss, mit Hilfe eines Widerstandes, mit dem Arduino. Dann kannst du über abwechselndes Verbinden der Pins mit `+5V` die Belegung nochmals überprüfen. Die LED sollte dann jeweils so leuchten wie die Pins im obigen Bild beschriftet sind. Achte dabei darauf, dass diesmal das **längere** Beinchen für `GND` genutzt wird.

Jetzt haben wir immer nur +5V angelegt, was einem Farbwert von 255 entsprechen würde. Wir wollen natürlich auch andere Farben und Farbmischungen darstellen, wie das funktioniert, erfährst unter der nächsten Überschrift.

VERSCHIEDENE FARBWERTE

Zunächst suchen wir uns auf dem Arduino Pins, welche neben der Zahl noch mit einer Tilde gekennzeichnet sind. Denn diese haben eine besondere Funktion namens **Pulsweitenmodulation**. Damit ist es uns nämlich möglich Werte zwischen 0 und 255 darzustellen. Dies funktioniert dann ähnlich zu analogen Pins mit Hilfe der Funktion `analogWrite(PIN, WERT)`.



AUFGABE 2 – RGB-LED AN RICHTIGE PINS ANSCHLIESSEN

Damit wir uns mit der Funktionalität weiter auseinandersetzen können, müssen wir die LED erstmal richtig anschließen. Verbinde alle Anschlüsse, außer `GND` mit den besonders gekennzeichneten Pins.

Jede Anschluss steht für eine Farbe, wir müssen also jedem Anschluss mitteilen welcher Farbwert dargestellt werden soll. Somit benötigen wir folgende Funktionen:

```
analogWrite(LEDrot, 0 .. 255);
analogWrite(LEDgruen, 0 .. 255);
analogWrite(LEDblau, 0 .. 255);
```

Es wäre ziemlich aufwendig für jede Farbe immer diese drei Funktionen hinzuschreiben, deswegen erleichtern wir uns die Arbeit etwas und gliedern den Code aus. Wir erstellen uns eine **Funktion,** welche wir dann mit dem Zahlen-Tupel aufrufen können. Wir nutzen als das Schlüsselwort `void` und geben unserer Funktion den Namen „rgbfarbe“.

```
void rgbfarbe (){
    …
}
```

Im Gegensatz zu unserem Reaktionsspiel wollen wir diesmal die Farbwerte mit übergeben. Wir brauchen also insgesamt drei Variablen für die drei Farben. Aussehen tut dies dann wie folgt:

```
void rgbfarbe (int rot, int gruen, int blau){
    …
}
```

Diese Variablen können wir jetzt in unserer Funktion nutzen und ersetzen die jeweiligen Werte in den `analogWrite()`-Befehlen mit diesen.

```
void rgbfarbe (int rot, int gruen, int blau){
    analogWrite(LEDrot, rot);
    analogWrite(LEDgruen, gruen);
    analogWrite(LEDblau, blau);
}
```

Der Aufruf der Funktion könnte beispielhaft für Farbe **rot** so aussehen:

```
rgbfarbe(255,0,0);
```

AUFGABE 3 – FARBDARSTELLUNG

Zunächst beginnen wir mit der Initialisierung der Variablen.

```
int LEDblau = 3;
```

Dann müssen wir innerhalb der `setup()`-Methode die jeweiligen Pin-Modi festlegen.

```
pinMode(LEDblau, OUTPUT);
```

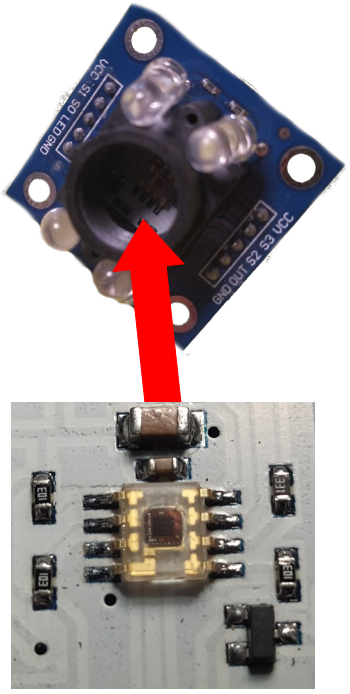
Im nächsten Schritt implementierst du die zuvor entwickelte Funktion „`rgbfarbe`“. Achte darauf, dass du diese nicht aus Versehen in eine bereits vorhandene Methode schreibst.

Im letzten Schritt wollen wir abwechselnd verschiedene Farben darstellen, dazu rufen wir unsere Funktion auf, setzen die Farbwerte in die Funktion ein und pausieren den Ablauf des Programms kurz mit Hilfe der `delay()`-Funktion.

Stelle alle Farben, welche auf Seite 3 gezeigt sind, abwechselnd dar!

Farben können wir jetzt erfolgreich darstellen, schauen wir uns jetzt einmal an, wie wir Farben mit Hilfe des Farbsensors erkennen können.

DER FARBSENSOR



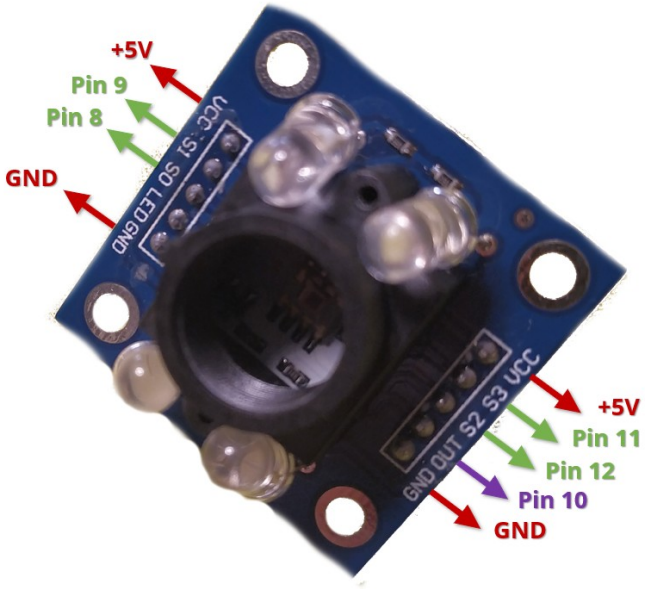
Der Farbsensor ist in der Lage zu erkennen, auf welche Farben dieser gerichtet ist. Im Zentrum des Sensors ist eine sogenannte **Photodiode**. Die genaue Funktionsweise ist auch als Photoeffekt bekannt, für dessen Beweis Albert Einstein den Nobelpreis erhalten hat. Das Wichtigste was du dazu wissen musst ist, dass die Lichtfarbe von einer physikalischen Größe namens Wellenlänge abhängt. Diese ist je nach Farbe unterschiedlich, wodurch eine Erkennung mit Hilfe der Diode möglich ist.

Damit das Objekt von dem wir die Farbe erkennen wollen, sind 4 LEDs ringsherum platziert, welche für eine gute Ausleuchtung sorgen.

ANSCHLIESSEN DES SENSORS

Das Verbinden des Sensors mit dem Arduino ist recht umfangreich, gezeigt wird dir dies im nachfolgenden Bild. Du willst aber natürlich auch wissen, welchen Zweck die jeweiligen Anschlüsse haben.

GND, VCC	Stromversorgung
S0, S1, S2, S3	Steuerung der Photodioden
OUT	Ausgabe des codierten Farbwertes
LED	wird nicht benötigt



AUFGABE 4 – VERBINDEN DES FARBSENSORS

Verbinde den Sensor mit den jeweiligen Pins, wie es die obige Grafik zeigt. Nutze dazu die Kabel, welche nur an einer Seite einen Metallstecker haben und an der anderen eine Art Buchse ist. Dies erleichtert die Verbindung mit dem Sensor, verbessert die Flexibilität bei der Positionierung und erhöht die Übersichtlichkeit,

Bevor du die Daten des Sensor einlesen kannst, musst du dich zunächst einer weiteren neuen Funktion beschäftigen – der **`pulseIn()`-Funktion**.

PULSEIN()-FUNKTION

Diese Funktion liest einen Wert von einem Pin des Arduino ein (`LOW` bzw. `HIGH`). Wenn der Wert z.B. `HIGH` ist, wartet `pulseIn()` darauf, dass der Pin auf den Wert `HIGH` wechselt, startet einen Timer und wartet anschließend darauf, dass der Pin wieder auf `LOW` wechselt. Daraufhin stoppt `pulseIn()` den Timer und gibt die Länge des Pulses in Mikrosekunden zurück.

```
pulseIn(Pin, Wert)
```

LICHT ALS WELLE

Jetzt ist es so, dass Licht eine Welle ist. Diese kann mit einer Wellenlänge beschrieben werden und wird eindeutig einer Farbe zugeordnet. Die Länge des Pulses steht dabei in direkter Verbindung mit dieser Wellenlänge und sagt uns, welche Farbe gerade erkannt wird. Die nachfolgende Grafik veranschaulicht dir diesen Zusammenhang noch einmal:

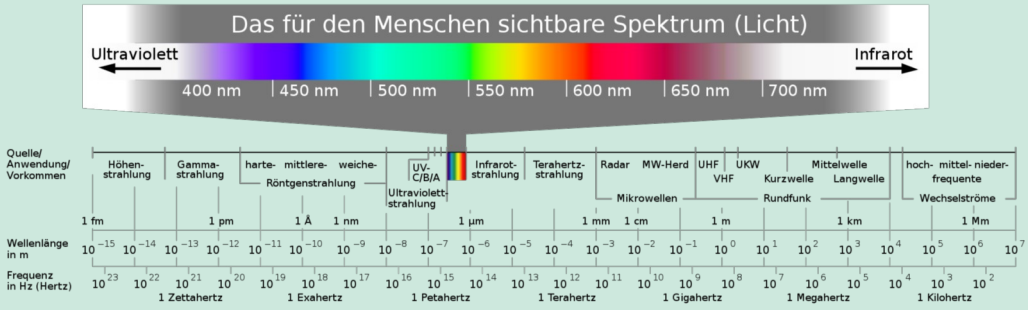


Abbildung 2: elektromagnetisches Spektrum

Wir sehen, dass die Wellenlänge von blau nach rot zunimmt, die Frequenz hingegen kleiner wird. Und diese Frequenz bestimmen wir indirekt über die Zeitdauer eines Pulses.

Wir wollen also nun, die verschiedenen Photodioden, für die jeweiligen Farben, ansteuern. Und mit Hilfe der `pulseIn()`–Funktion die Wellenlänge bestimmen. Dazu beginnen wir damit der Diode `S0` den Wert `HIGH` zuzuweisen. Diese ist für das weiße Licht zuständig und muss die ganze Zeit über angeschaltet bleiben.

Dann wird es etwas spezifischer, zunächst setzen wir `S1`, `S2` und `S3` auf `LOW`. Dann wollen wir den Wert für rotes Licht einlesen. Dazu setzen wir `S1` auf `HIGH` und nutzen die `pulseIn()`-Funktion. Aussehen tut dies dann wie folgt:

```
digitalWrite(s0, HIGH);
digitalWrite(s1, LOW);
digitalWrite(s2, LOW);
digitalWrite(s3, LOW);
digitalWrite(s1, HIGH);
rot = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
```

Soweit so gut, aber der Inhalt der `pulseIn()`-Funktion sieht auf einmal so komplex aus? Das stimmt, aber dahinter versteckt sich eine dir bereits bekannte Programmierstruktur – die **einfache Verzweigung**. Dabei handelt es sich lediglich um eine Kurzschreibweise, um die Übersichtlichkeit zu gewährleisten.

Gelesen werden kann dies wie folgt:

```
WENN digitalRead(out) gleich HIGH ist
DANN setze den Wert auf LOW ANSONSTEN auf HIGH
```

Eine Welle hat immer Berge und Täler, je nach dem an welcher Position wir diese gerade anfangen einzulesen, wollen wir die Zeit bestimmen, bis der nächste Zustand eintritt. Ist sie zu Beginn HIGH, dann wollen wir solange warten bis sie LOW ist und umgekehrt.

Die anderen Farbwerte bestimmen wir ähnlich. Wir setzen dann die blaue Diode auf HIGH und messen den Wert, genauso wie für die grüne.

```
digitalWrite(s3, HIGH);
blau = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
digitalWrite(s2, HIGH);
gruen = pulseIn(out, digitalRead(out) == HIGH ? LOW : HIGH);
```

Neben der Funktionsweise musst du auch hier wieder die **Variablen** im Vorfeld initialisieren und die jeweiligen **Pin-Modi** festlegen. Für die Dioden und zur Speicherung der Farbwerte kannst du wie folgt die Variablen festlegen:

```
int s0 = 8;
int rot = 0;
```

Die Pin-Modi sind für die jeweiligen Dioden auf `OUTPUT` zu setzen und für die Ausgabe des Farbwertes (`out`) auf `INPUT`. Umsetzen kannst du dies mit folgendem Befehl:

```
pinMode(s0, OUTPUT);
```

AUFGABE 5 – FARBSENSOR TESTEN

Beginne zunächst mit der Initialisierung aller Variablen und lege die Pin-Modi fest. Schreibe dann innerhalb der `loop()`-Methode die jeweiligen Befehle für das Auslesen der Werte und gib diese zur Kontrolle über den seriellen Monitor aus. Diesen startest du innerhalb der `setup()`-Methode mit `Serial.begin(9600)` und gibst innerhalb der `loop()`–Methode die Werte mit `Serial.print("Roter Anteil: ")` und `Serial.println(rot)` aus. Nutze zum Testen das zur Verfügung gestellte farbige Testblatt – welche Beobachtungen kannst du machen?

DIE BEDEUTUNG DER WERTE

Du hast mit Sicherheit gemerkt, dass das Einscannen einer Farbe immer den niedrigsten Wert in der Ausgabe zu Folge hat. Im Sensor selbst sind Referenzwerte für die Farben rot, grün und blau hinterlegt. Dabei gibt der jeweils ausgegebene Wert die Differenz zu diesen Standardwerten an. Umso kleiner dieser ist, desto näher sind wir an einer unserer gewünschten Farben dran. Wir können also sagen, der jeweils kleinste Wert ist unser gesuchter Farbwert.

Ob diese Bedingung jeweils zutrifft überprüfen wir mit Hilfe der **einfachen Verzweigung** und des logischen **UND-**Vergleichsoperators. Beispielhaft für die Farbe rot bedeutet das: Der Farbwert muss kleiner als der von blau und von grün sein. Im Code sieht dies dann wie folgt aus:

```
if (rot < blau && rot < gruen){
    Serial.println("Farbe rot erkannt!");
    delay(1000);
}
```

Die Wartezeit dient einfach der Nachvollziehbarkeit des Ergebnisses, damit im seriellen Monitor nicht hunderte Werte auf einmal ausgegeben werden. Die Verzweigung können wir nun für die anderen zwei Farben und für den Fall, dass nichts erkannt wurde, noch weiter verschachteln. Dies funktioniert folgendermaßen:

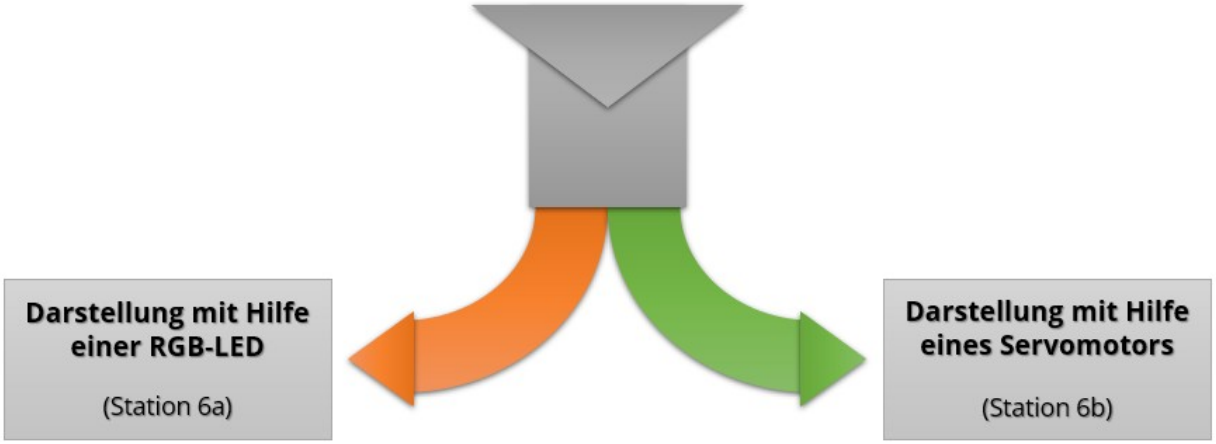
```
if (rot < blau && rot < gruen){
    Serial.println("Farbe rot erkannt!");
    delay(1000);
} else if(Bedingung für grüne Farbe){
     ...
} else if(Bedingung für blaue Farbe){
     ...
} else {
     Serial.println("Keine Farbe erkannt! :( ");
}
```

AUFGABE 6 – FARBEN ERKENNEN

Ergänze deinen Code um die oben beschriebene Fallunterscheidung zur Erkennung der Farbe. Die Farbwerte kannst du zur Kontrolle weiterhin ausgeben lassen und musst dies nicht löschen. Teste auch hier deine Schaltung wieder mit dem zur Verfügung gestellten Farbblatt.

ZEIT FÜR ENTSCHEIDUNGEN

Die Erkennung der Farben funktioniert, allerdings ist die Ausgabe über den seriellen Monitor immer an einen Computer gebunden und ziemlich langweilig. Daher kannst du dich nun nachfolgend entscheiden, wie du die Darstellung der erkannten Farben realisieren willst.



ENTSCHEIDUNG GETROFFEN?

Dann bitte die Betreuer um das jeweilige Aufgabenblatt!

Grafik auf dem Deckblatt: *Rote LED-Lampe aus, Openclipart, CC0, https://creativecommons.org/publicdomain/zero/1.0/, https://publicdomainvectors.org/de/kostenlose-vektorgrafiken/Rote-LED-Lampe-aus/61716.html*

und

Kaufmann-comic-Figur-Vektor-Bild, CC0, https://creativecommons.org/publicdomain/zero/1.0/, https://publicdomainvectors.org/de/kostenlose-vektorgrafiken/Kaufmann-comic-Figur-Vektor-Bild/6840.html

Screenshots: *fritzing electronics made by easy und Arduino IDE 1.8.12 (windows)*

Abbildung 1: *5mm RGB LED, oomlout, CC BY-SA 2.0, https://creativecommons.org/licenses/by-sa/2.0/deed.en, https://creativecommons.org/licenses/by-sa/2.0/deed.en*

Abbildung 2: *Electromagnetic spectrum, Horst Frank / Phrood / Anony, CC BY-SA 3.0, https://creativecommons.org/licenses/by-sa/3.0/deed.en, https://commons.wikimedia.org/wiki/File:Electromagnetic_spectrum_-de_c.svg*

Alle weiteren Grafiken: *Patrick Binkert, EduInf@TUD*