Workshop
Arduino-Programmierung #3

RGB-LED, PWM, Serial, Strings, Potentiometer, map

Joachim Baur

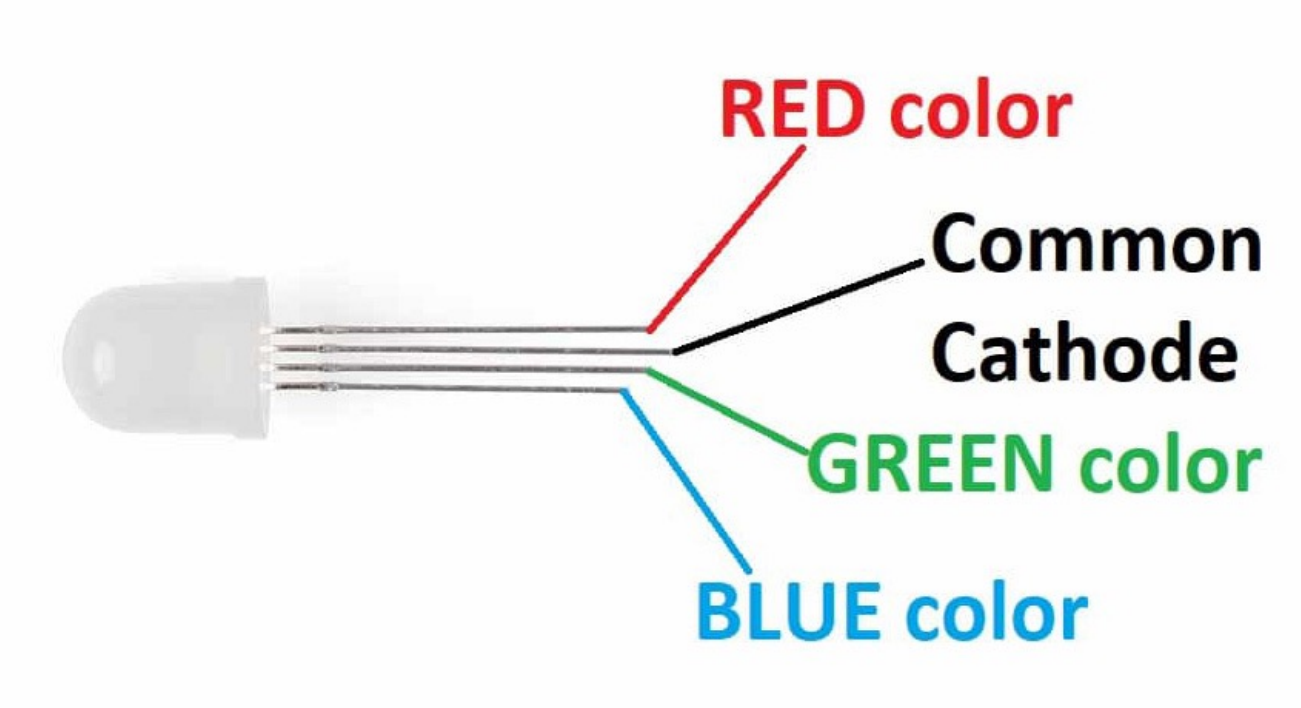
E-Mail: post@joachimbaur.de

ZTL-Alias: @joachimbaur

Download für diesen Workshop: www.joachimbaur.de/WS3.zip

RGB LED verwenden

- RGB = 3 LEDs in einer Kapsel -> Farbmischung möglich
- "Common Cathode": alle LED-Kathoden teilen sich das Minus-Beinchen (Gegenteil: "Common Anode")
- Verschiedene Vor-Widerstandswerte pro LED-Bein nötig für eine ausgewogene Helligkeit/Mischung



Vorwiderstände berechnen

Working voltage	R: 2.0-2.2V G: 3.0-3.2V B: 3.0-3.2V
Working current	20mA

Online LED-Vorwiderstandberechner: https://www.elektronik-kompendium.de/sites/bau/1109111.htm

ROT

Durchlassspannung: 2,0 V
Betriebsspannung: 5,0 V
Betriebsstrom: 20 mA

Vorwiderstand: **150 Ohm**

Grün

Durchlassspannung: 3,0 V
Betriebsspannung: 5,0 V
Betriebsstrom: 20 mA

Vorwiderstand: **100 Ohm**

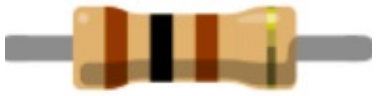
Blau

Durchlassspannung: 3,0 V
Betriebsspannung: 5,0 V
Betriebsstrom: 20 mA

Vorwiderstand: **100 Ohm**

Online Widerstand-Farbcodes: https://de.farnell.com/widerstand-farbcode-rechner





Aufbau auf Steckbrett

ROT
Pin 4
Widerstand 150 Ohm

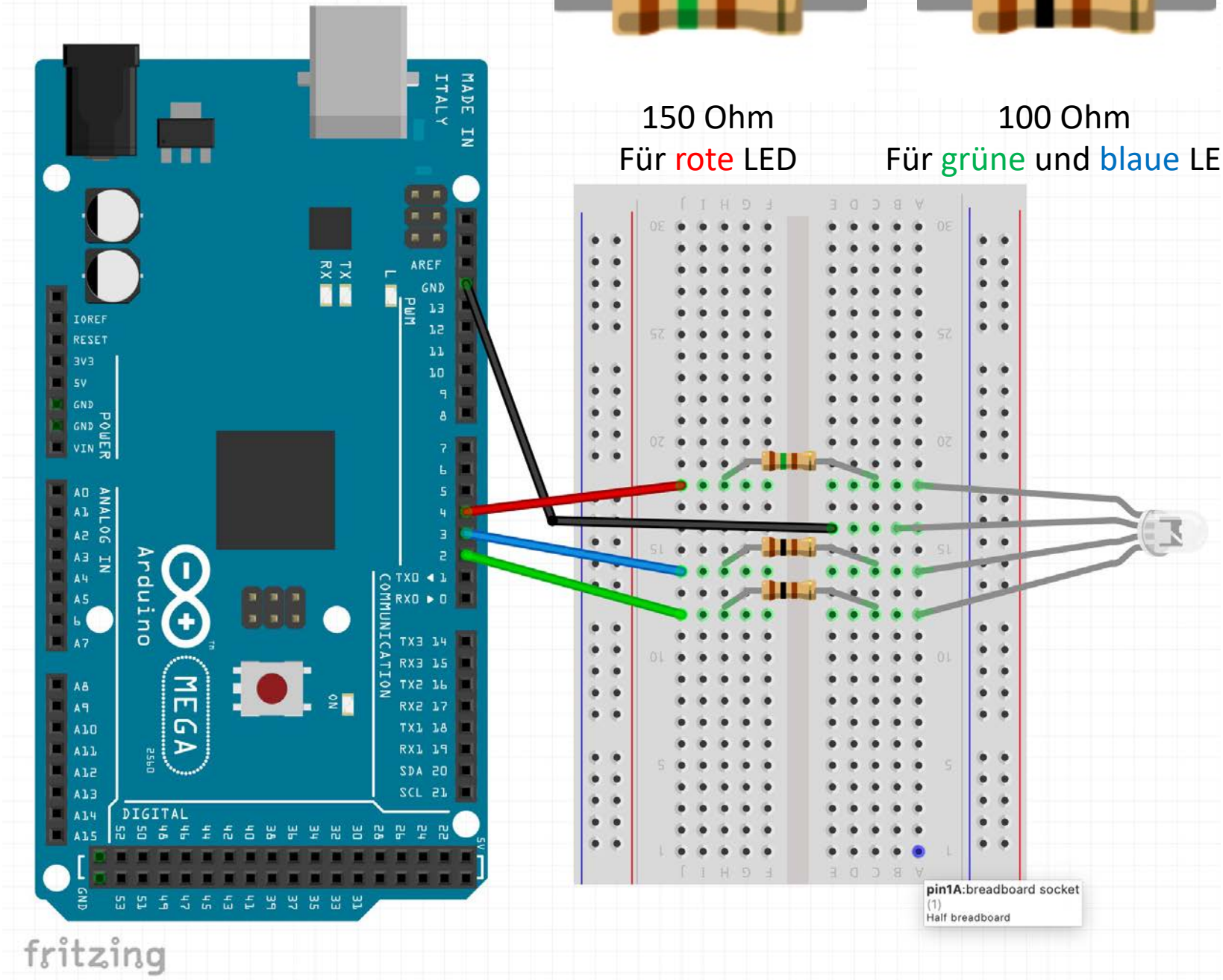
BLAU
Pin 3
Widerstand 100 Ohm

GRÜN
Pin 2
Widerstand 100 Ohm

GND
Gemeinsame
Kathode

150 Ohm
Für rote LED

100 Ohm
Für grüne und blaue LED



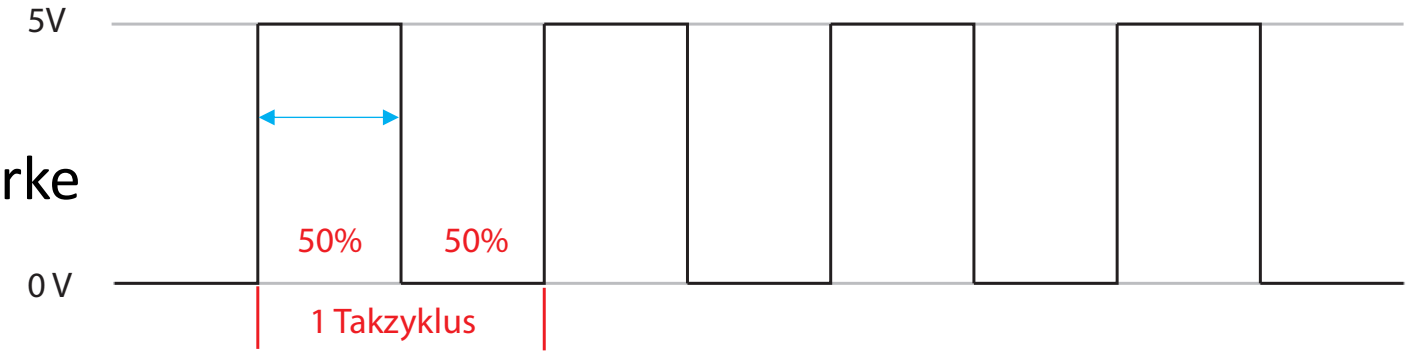
Sketch: 03_RGB_LED.ino

```
const int PIN_ROT   = 4;
const int PIN_GRUEN   = 3;
const int PIN_BLAU= 2;

void setup() {
    pinMode( PIN_ROT,    OUTPUT );
    pinMode( PIN_GRUEN,  OUTPUT );
    pinMode( PIN_BLAU,   OUTPUT );
}

void loop() {
    // LED leuchtet rot
    digitalWrite( PIN_ROT,    HIGH );
    digitalWrite( PIN_GRUEN,  LOW  );
    digitalWrite( PIN_BLAU,   LOW  );
    delay(1000);

    // LED leuchtet grün
    digitalWrite( PIN_ROT,    LOW  );
    digitalWrite( PIN_GRUEN,  HIGH );
    digitalWrite( PIN_BLAU,   LOW  );
    delay(1000);

    // LED leuchtet weiß
    digitalWrite( PIN_ROT,    HIGH );
    digitalWrite( PIN_GRUEN,  HIGH );
    digitalWrite( PIN_BLAU,   HIGH );
    delay(1000);
}
```

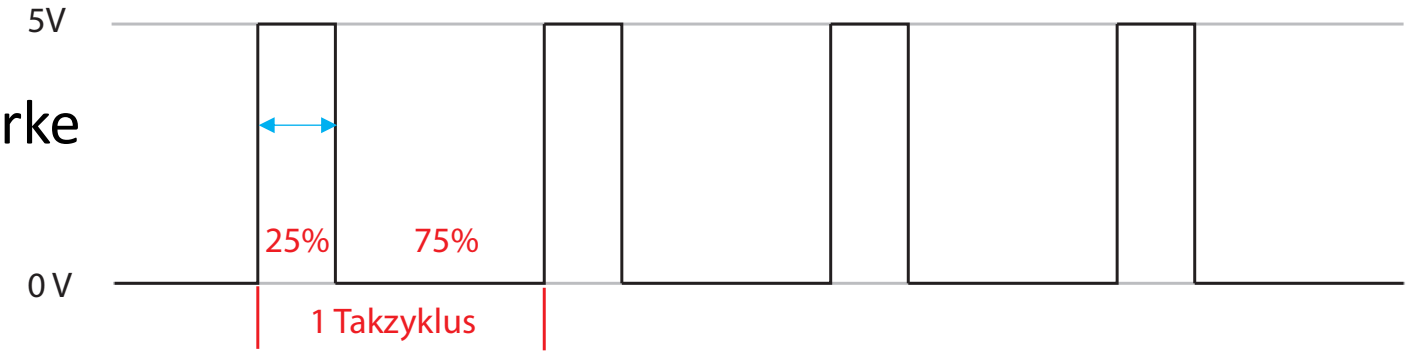
Pulsweitenmodulation (PWM)

- Ein Digitalausgang kann nur auf HIGH (5V) oder LOW (0V) Pegel sein
- Um „Zwischenwerte“ zu simulieren, wird PWM verwendet
- Dabei wird abwechselnd ein HIGH – LOW – HIGH - LOW… ausgegeben
- Wenn dies schnell genug geschieht, scheint die angeschlossene LED schwächer zu leuchten. Die Leuchtstärke wird von der Pulsweite bestimmt:

50% Leuchtstärke

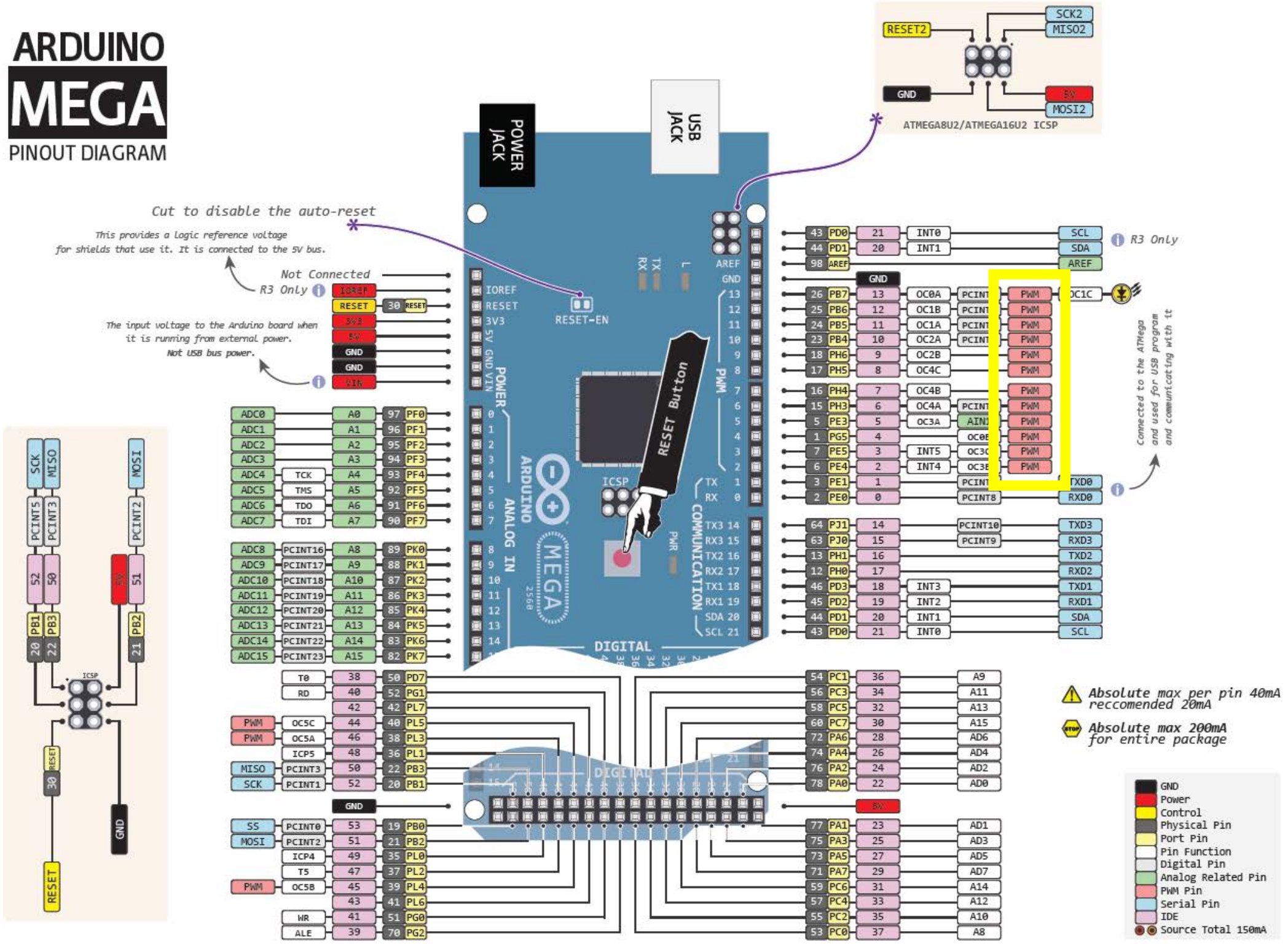


25% Leuchtstärke



ARDUINO MEGA

PINOUT DIAGRAM



PWM und analogWrite()

- Das PWM-Signal kann natürlich manuell im Code erzeugt werden (loop, digitalWrite LOW/HIGH)
- Es gibt aber im Chip eingebaute „Hardware PWM-Generatoren“, die auf bestimmten Pins automatisch und dauerhaft ein PWM-Signal ausgeben können
- Dazu wird die Funktion `analogWrite(pin,n)` benutzt
- Das `n` ist ein Wert von 0…255 für Pulsweite 0...100%
- `analogWrite( 3, 127 )` gibt also ein 50% PWM-Signal an Pin 3 aus (127 = 50% von 255)
- Damit können wir die Farbmischung der RGB-Einzel-LEDs fein justieren

Sketch: 04_RGB_Farbwechsel.ino

```
const int PIN_ROT = 4;
const int PIN_GRUEN = 3;
const int PIN_BLAU = 2;

// Start mit Rot 100%
int wertR = 255;
int wertG = 0;
int wertB = 0;

void setup() {
      pinMode( PIN_ROT,   OUTPUT );
      pinMode( PIN_GRUEN, OUTPUT );
      pinMode( PIN_BLAU,  OUTPUT );
}

void loop() {
      // ROT RGB ( 255, 0, 0 ) zu BLAU RGB ( 0, 0, 255 )
      for (int i=0; i<255; i++) {
            setzeLedAusgaenge();
            wertR--;
            wertB++;
      }
      // BLAU RGB ( 0, 0, 255 ) zu GRÜN RGB ( 0, 255, 0 )
      for (int i=0; i<255; i++) {
            setzeLedAusgaenge();
            wertB--;
            wertG++;
      }
      // GRÜN RGB ( 0, 255, 0 ) zu ROT RGB ( 255, 0, 0 )
      for (int i=0; i<255; i++) {
            setzeLedAusgaenge();
            wertG--;
            wertR++;
      }
}

void setzeLedAusgaenge() {
      analogWrite( PIN_ROT,   wertR );
      analogWrite( PIN_GRUEN, wertG );
      analogWrite( PIN_BLAU,  wertB );
      delay(10);
}
```

Die serielle Schnittstelle

- Der Arduino Mega hat 4 fest eingebaute serielle Schnittstellen („Hardware Serial“), eine davon ist über USB ansprechbar
- In der Arduino IDE gibt es den „Seriellen Monitor“, über den mit dem Mega kommuniziert werden kann (Texte/Daten hin- und herschicken)

- Es muss eine gemeinsame Übertragungsgeschwindigkeit vereinbart werden („baud“, bits pro Sekunde) diese liegt zwischen 9600 und 2000000 baud, 115200 baud reichen normalerweise

```
void setup() {
    Serial.begin( 115200 ); // seriellen USB-Port mit 115200 baud öffnen
}

void loop() {
    Serial.println("Hallo"); // String "Hallo" per USB an PC senden
    delay(1000);
}
```

Werte per Serial ausgeben

- Die Befehle print() und println() senden Text vom Arduino Mega Code an die serielle Schnittstelle (zB Konsole)
- println() schließt die aktuelle Textzeile ab und sendet sie

```
Serial.println("Wert"); // gibt den Text “Wert“ in der Konsole aus

Serial.println(10); // gibt den Text „10“ in der Konsole aus

int wert = 20;

// 3 Möglichkeiten, um den String „Wert = 20“ in der Konsole auszugeben:

Serial.println("Wert = " + String(wert)); // String-Verkettung mit „+“

Serial.print("Wert = "); // print() fügt Zeichen zur aktuellen Zeile dazu
Serial.println(wert); // println() schließt die Zeile ab und zeigt sie an

Serial.print("Wert = ");
Serial.print(wert);
Serial.println(); // „leeres“ println() zum Abschluss und Senden der Zeile

Serial.println("Wert = " + wert); // NEIN! Chaos & Anarchie!
```

String-Eigenschaften und -Methoden

- Arduino Strings sind nicht nur „Buchstabenfolgen“ sondern C++-Objekte, die noch zusätzliche nützliche Eigenschaften und Methoden haben:

```
void setup() {
    Serial.begin(115200);

    String text1 = "Hallo";

    Serial.println(text1.length());         // 5
    Serial.println(text1.charAt(0));        // "H"
    text1.setCharAt(1, 'e');
    Serial.println(text1);                  // "Hello"
    Serial.println(text1.indexOf('o'));     // 4
    Serial.println(text1.startsWith("X")); // false (0)
    text1.replace("Hel","Ki");
    Serial.println(text1);                  // "Kilo"

    String gewicht = "1000";
    int kilo = gewicht.toInt(); // 1000 in Zahl umwandeln
    Serial.println(kilo + 10);  // "1010"
}

void loop() {
}
```

https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/

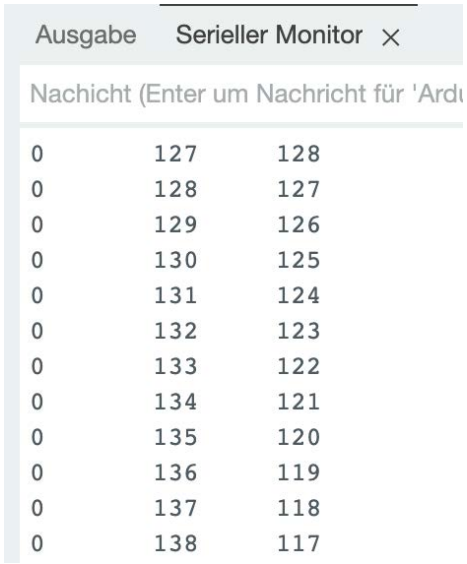
Der serielle Plotter

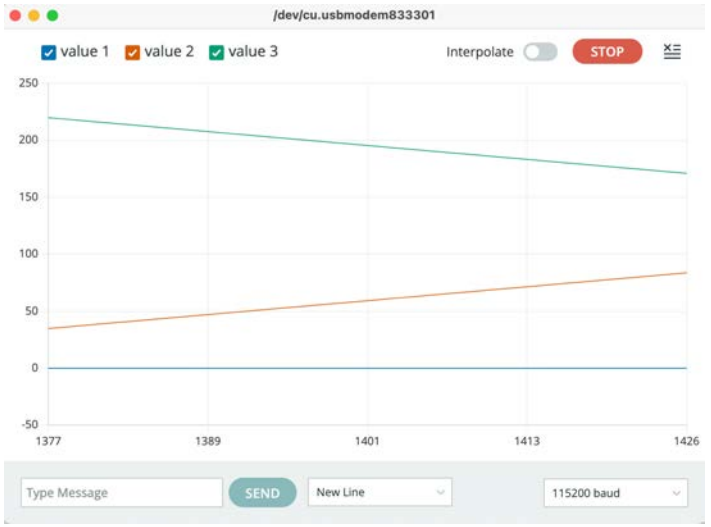
- Um statt einer reinen Textausgabe von Variablenwerten in der seriellen Konsole besser zu visualisieren, kann der Serielle Plotter benutzt werden

- Er stellt die Zahlenwerte einer Zeile als Diagramm dar.
- Trennzeichen zwischen den Zahlenwerten sind TAB ("\t"), Leerzeichen oder Komma

Sketch: 05_RGB_Farbwechsel_Serial.ino

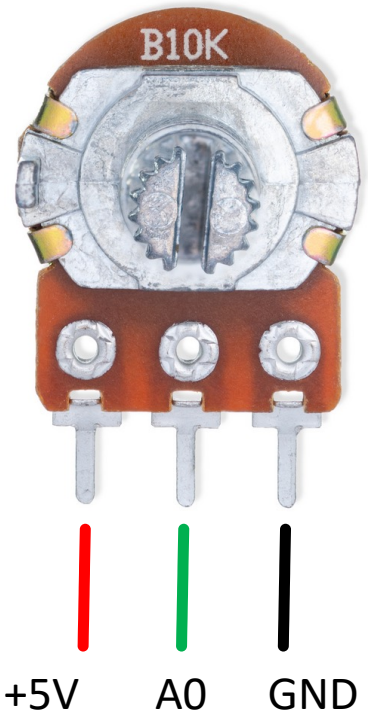
```
void setup() {
    . . .
    Serial.begin(115200);
}

void setzeLedAusgaenge() {
    . . .
    Serial.print(wertB);
    Serial.print("\t");
    Serial.print(wertR);
    Serial.print("\t");
    Serial.print(wertG);
    Serial.println();
    . . .
}
```





analogRead()

- Der Arduino Mega kann an den „Analog“-Pins A0 bis A15 die Spannung zwischen 0 und 5V messen
- Dort kann z.B. ein Potentiometer (veränderbarer Widerstand) angeschlossen werden:



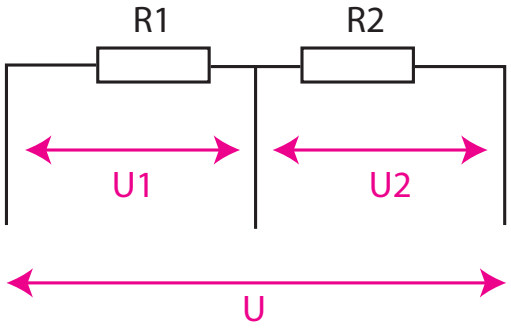
Sketch: 06_Potentiometer

```
const int POTI_PIN = A0;

void setup() {
    Serial.begin( 115200 );
}

void loop() {
    Serial.println( analogRead(POTI_PIN) );
    delay(200);
}

// Serielle Konsole: Werte von 0 bis 1023
```

Potentiometer = Spannungsteiler

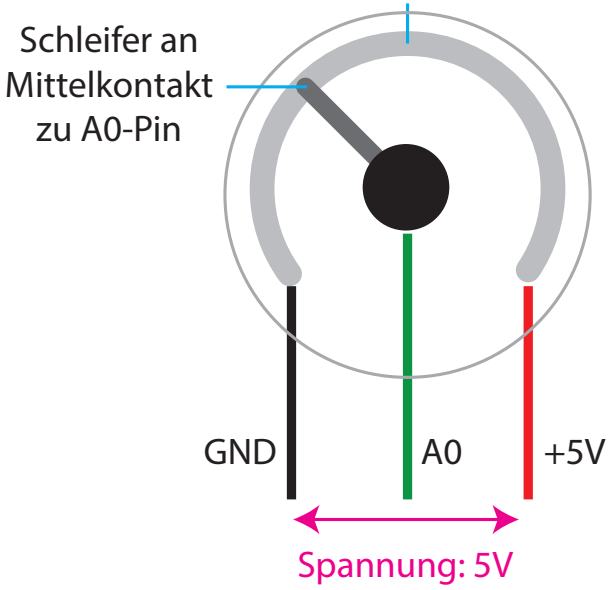
- Bei mehreren in Reihe geschalteten Widerständen fällt die Spannung zwischen den Widerständen im Verhältnis der Widerstandswerte ab
- EIn Potentiometer ist ein veränderbarer Widerstand (Drehknopf oder Schieberegler), der als Spannungsteiler funktioniert

Spannungsteiler allgemein

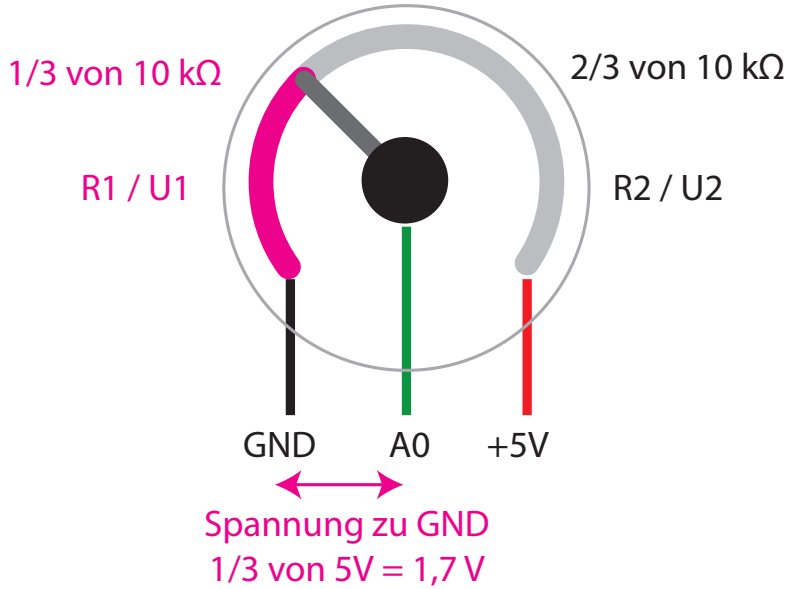


$$\frac{R_1}{U_1} = \frac{R_2}{U_2}$$

Potentiometer Widerstandsbahn 10 kΩ gesamt



Aufteilung der Spannung proportional zum Widerstand



RGB LED mit Potentiometer regeln

Zielsetzung:

- Potentiometer auslesen mit analogRead()
- Poti-Werte von 0…1023 umrechnen in LED PWM-Stufen von 0…255 (durch 4 teilen)
- LED PWM-Stufen mit analogWrite() ausgeben
- Potistellung 0: RGB-LED leuchtet voll rot
- Potistellung 1023: RGB-LED leuchtet voll blau
- Potistellung dazwischen: Mischfarben rot-blau (pink/rosa/violett)

07_RGB_Potentiometer.ino

```
const int PIN_ROT = 4;
const int PIN_GRUEN = 3;
const int PIN_BLAU = 2;

int wertR = 0;
int wertG = 0;
int wertB = 0;

const int POTI_PIN = A0;

void setup() {
     pinMode( PIN_ROT,   OUTPUT );
     pinMode( PIN_GRUEN, OUTPUT );
     pinMode( PIN_BLAU,  OUTPUT );

     pinMode( POTI_PIN, INPUT );
 }

void loop() {
     int potiWert = analogRead( POTI_PIN ); // 0...1023
     int ledStufe = potiWert / 4;           // 0...255 (255.75)

     // ledStufe 0:   Farbe = Rot ( 255, 0, 0 )
     // ledStufe 255: Farbe = Blau ( 0, 0, 255 )
     wertR = 255 - ledStufe; // Poti = 0, wertR = 255 ----- Poti = 1023, wertR =     0
     wertB = ledStufe;           // Poti = 0, wertB =      0 ----- Poti = 1023, wertB =
     255

     setzeLedAusgaenge();
}

void setzeLedAusgaenge() {
     analogWrite( PIN_ROT,   wertR );
     analogWrite( PIN_GRUEN, wertG );
     analogWrite( PIN_BLAU,  wertB );
     //delay(10);
}
```

map() funktion für Wertebereiche

- Einfacheres Umrechnen von Wertebereichen
- Nur für Ganzzahlen (int)
- 5 Argumente:
  - **1.** umzurechnende Variable
  - **2.+3. Start**bereich (von, bis) der Eingangswerte
  - **4.+5. Ziel**bereich (von, bis) der Zielwerte

```
// statt int ledStufe = potiWert / 4; // 0...255
int ledStufe = map( potiWert, 0, 1023, 0, 255 );
```

08_RGB_Poti_map.ino