

Absolute Maximum Ratings at TA=25°C

Parameter	Hyper Red	Blue	Green	Units
Power dissipation	75	120	102.5	mW
DC Forward Current	30	30	25	mA
Peak Forward Current [1]	185	150	150	mA
Reverse Voltage		5		V
Operating/Storage Temperature		-40°C To +85°C		
Lead Solder Temperature [2]		260°C For 3 Seconds		
Lead Solder Temperature [3]		260°C For 5 Seconds		

$$\text{Rot} - R = U/I \rightarrow R = 4.5V / 0.03A \rightarrow R = 150\Omega$$

$$\text{Blau} - R = U/I \rightarrow R = 4.5V / 0.03A \rightarrow R = 150\Omega$$

$$\text{Grün} - R = U/I \rightarrow R = 4.5V / 0.025A \rightarrow R = 180\Omega$$

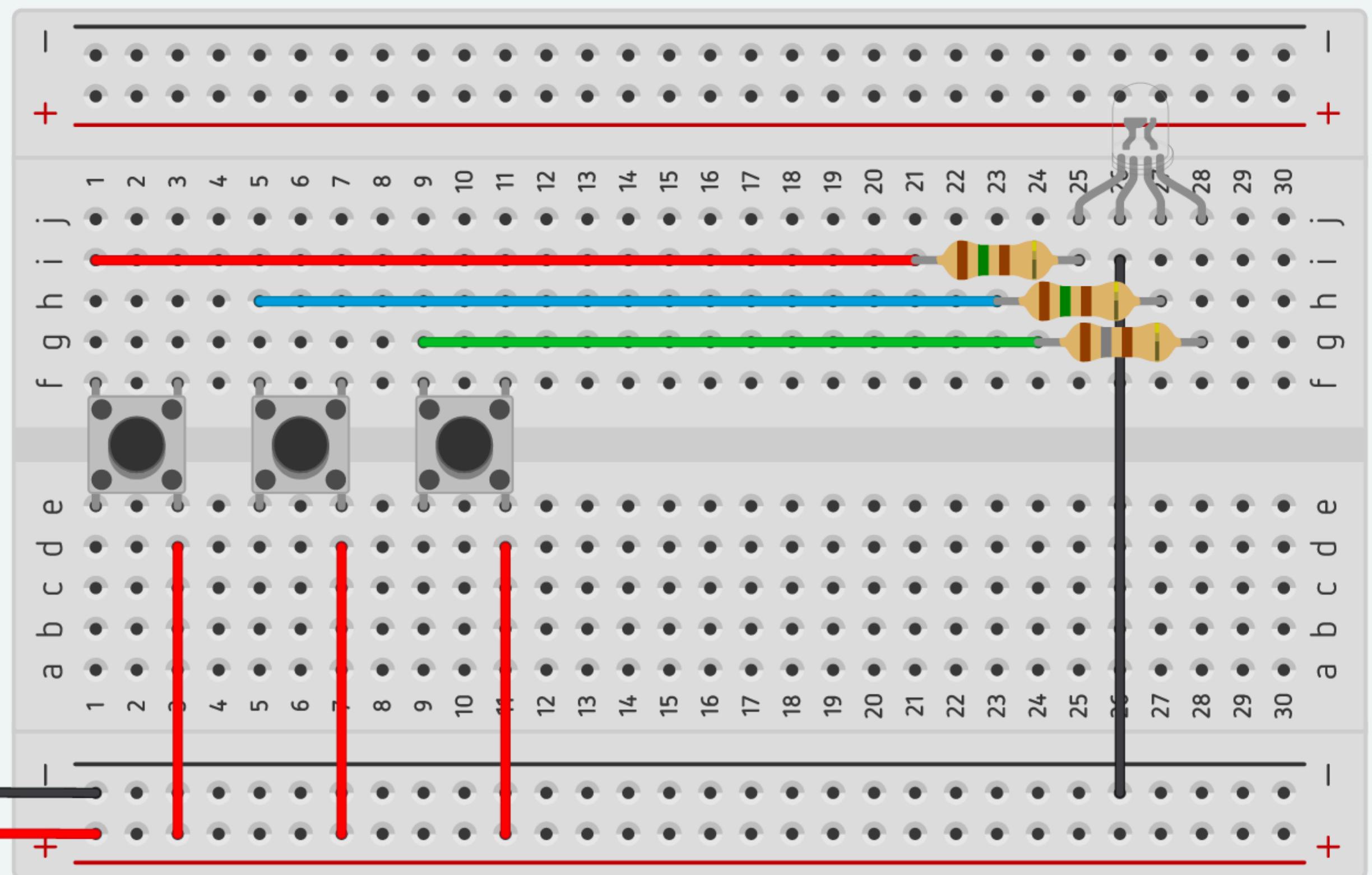
Absolute Maximum Ratings at TA=25°C

Parameter	Hyper Red	Blue	Green	Units
Power dissipation	75	120	102.5	mW
DC Forward Current	30	30	25	mA
Peak Forward Current [1]	185	150	150	mA
Reverse Voltage		5		V
Operating/Storage Temperature		-40°C To +85°C		
Lead Solder Temperature [2]		260°C For 3 Seconds		
Lead Solder Temperature [3]		260°C For 5 Seconds		

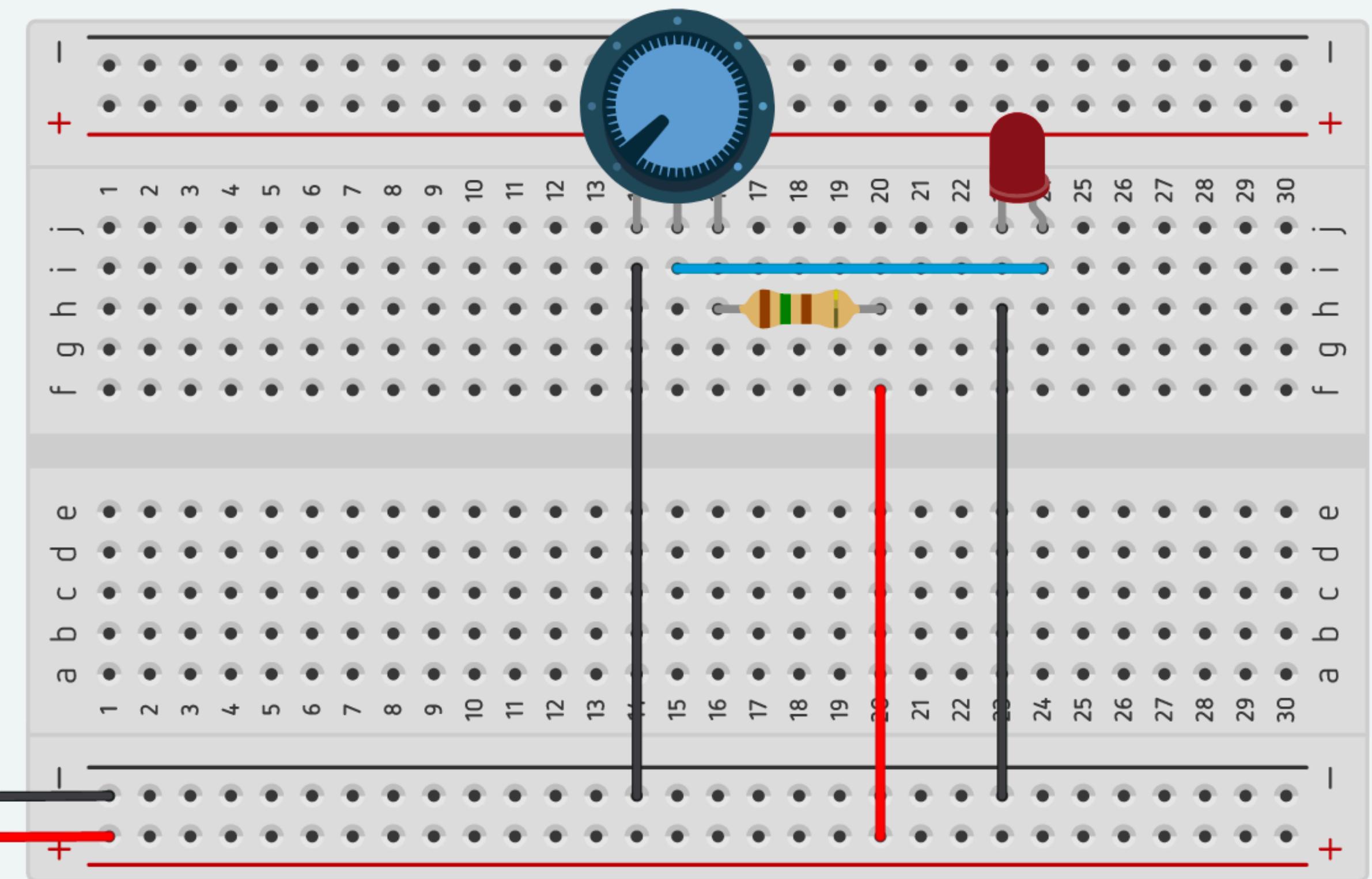
$$\text{Rot} - R = U/I \rightarrow R = 9V / 0.03A \rightarrow R = 300\Omega$$

$$\text{Blau} - R = U/I \rightarrow R = 9V / 0.03A \rightarrow R = 300\Omega$$

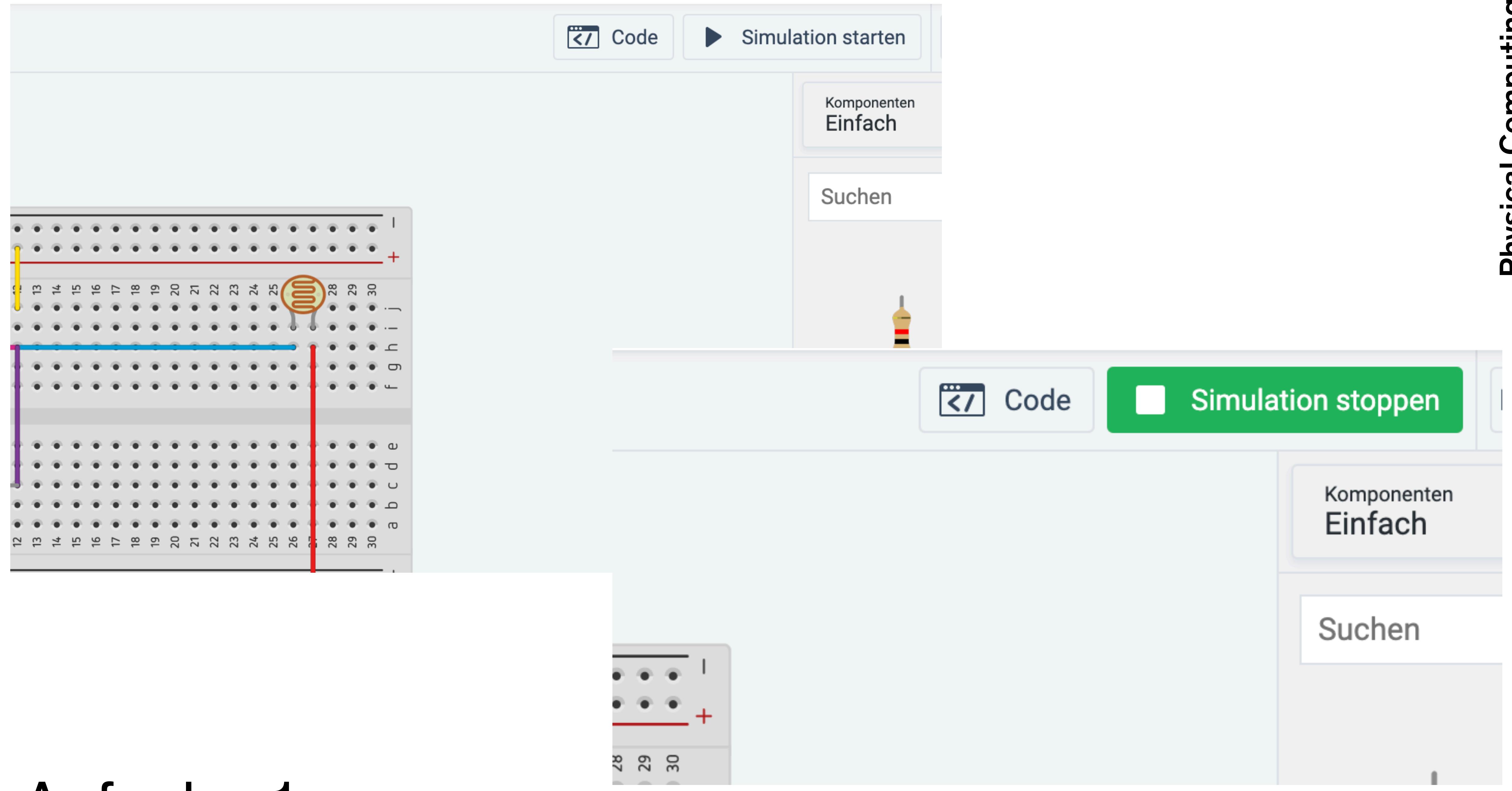
$$\text{Grün} - R = U/I \rightarrow R = 9V / 0.025A \rightarrow R = 360\Omega$$



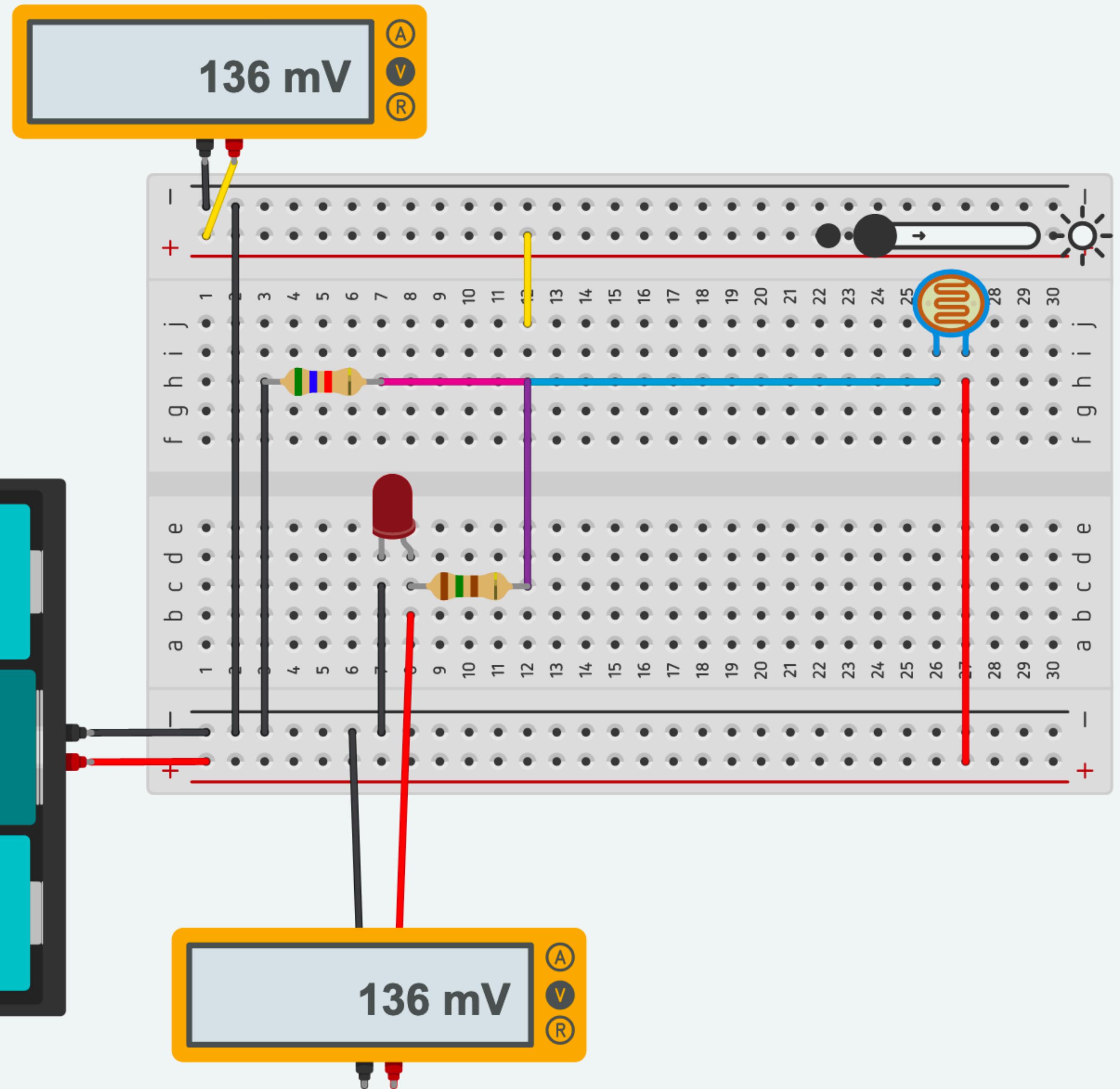
Aufgabe 1



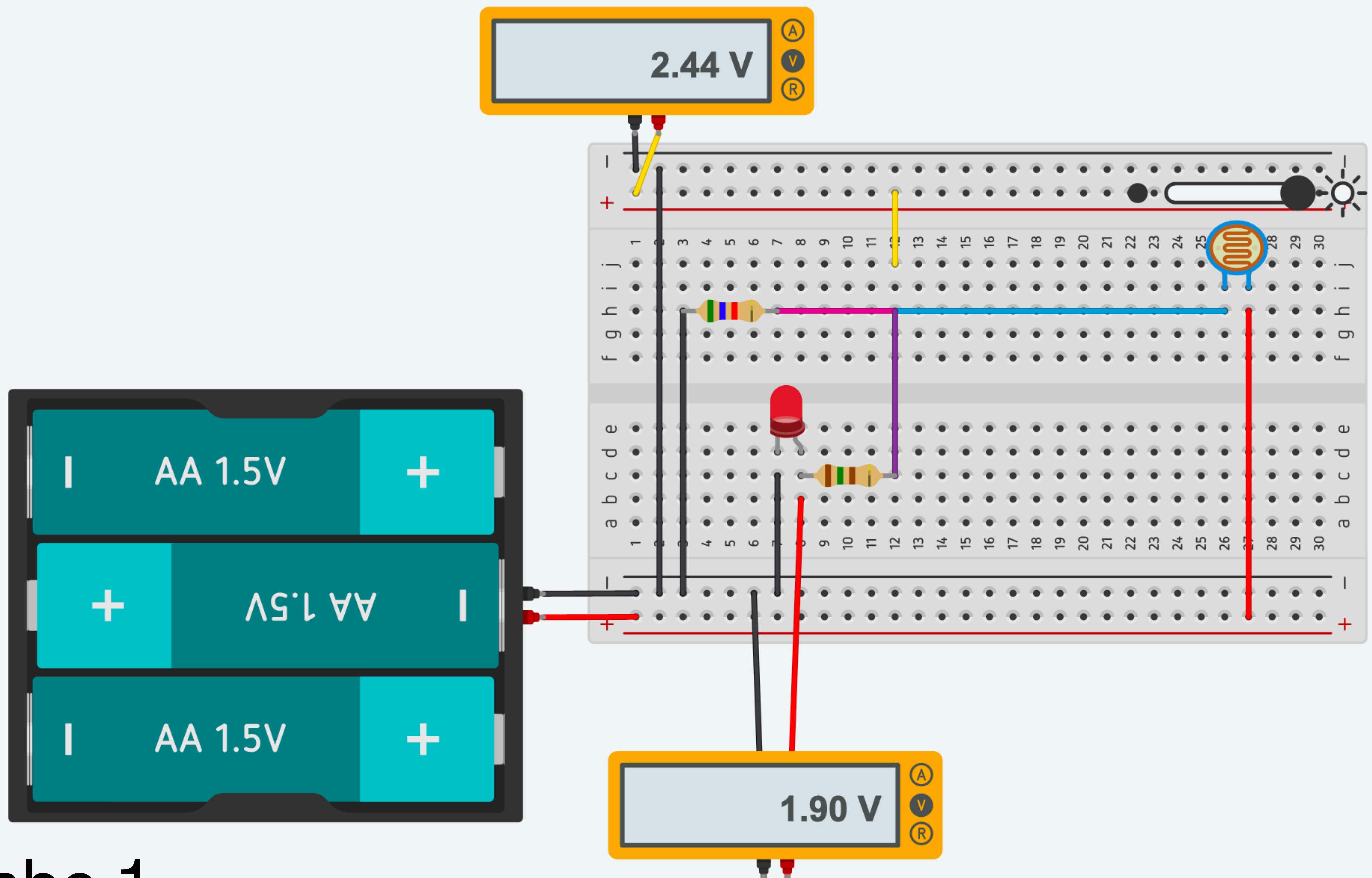
Aufgabe 1



Aufgabe 1



Aufgabe 1



Aufgabe 1

Recap

Warum Physical Computing?

Arduino Bauteile

EVA-Prinzip

Programmier-Grundlagen

Recap

Inhalte von Datenblättern

Baugröße

Ströme

Anschlüsse

Spannungen

Frequenzen

Einsatzbereich

Helligkeit

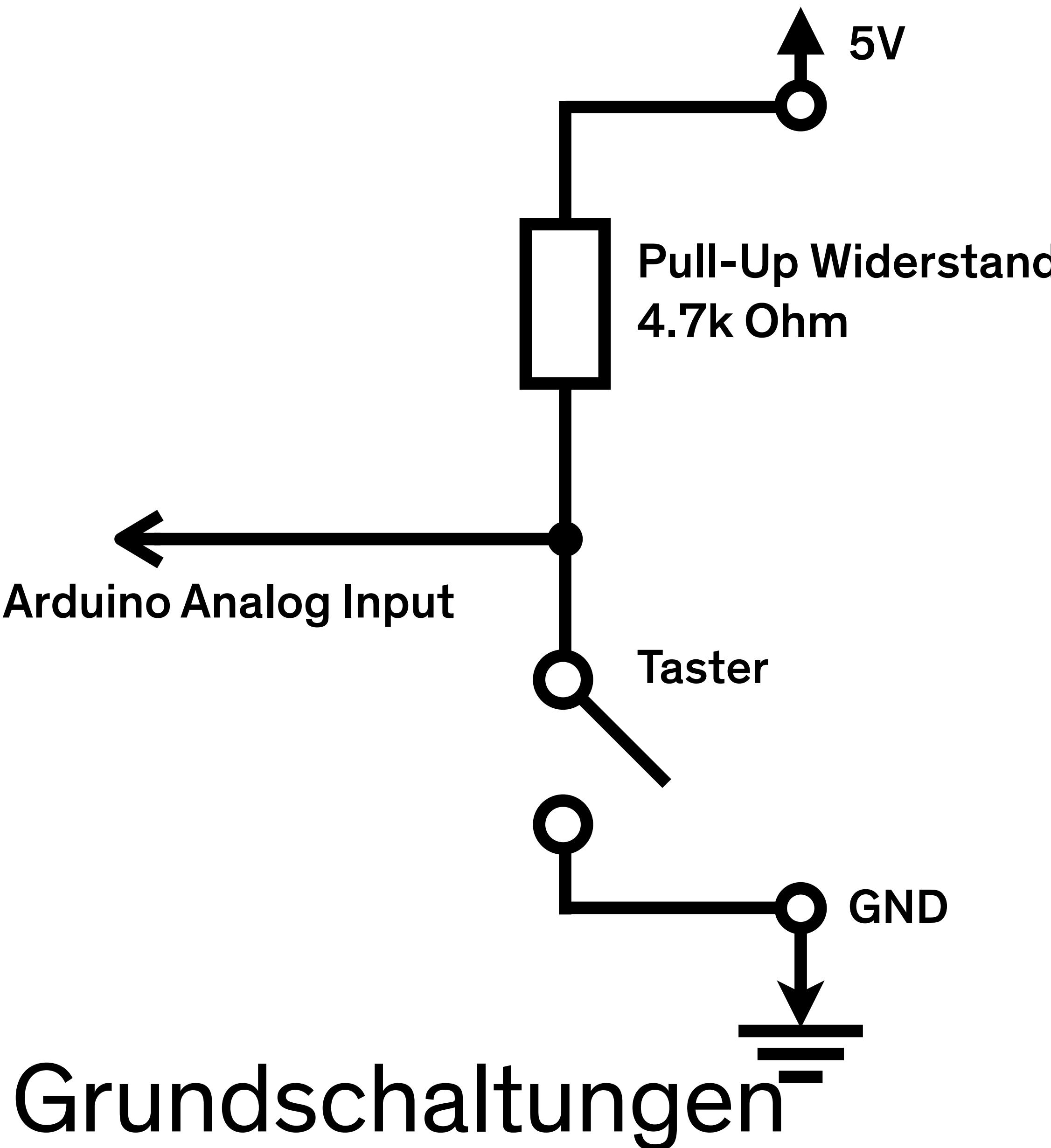
Temperatur

Toleranzen

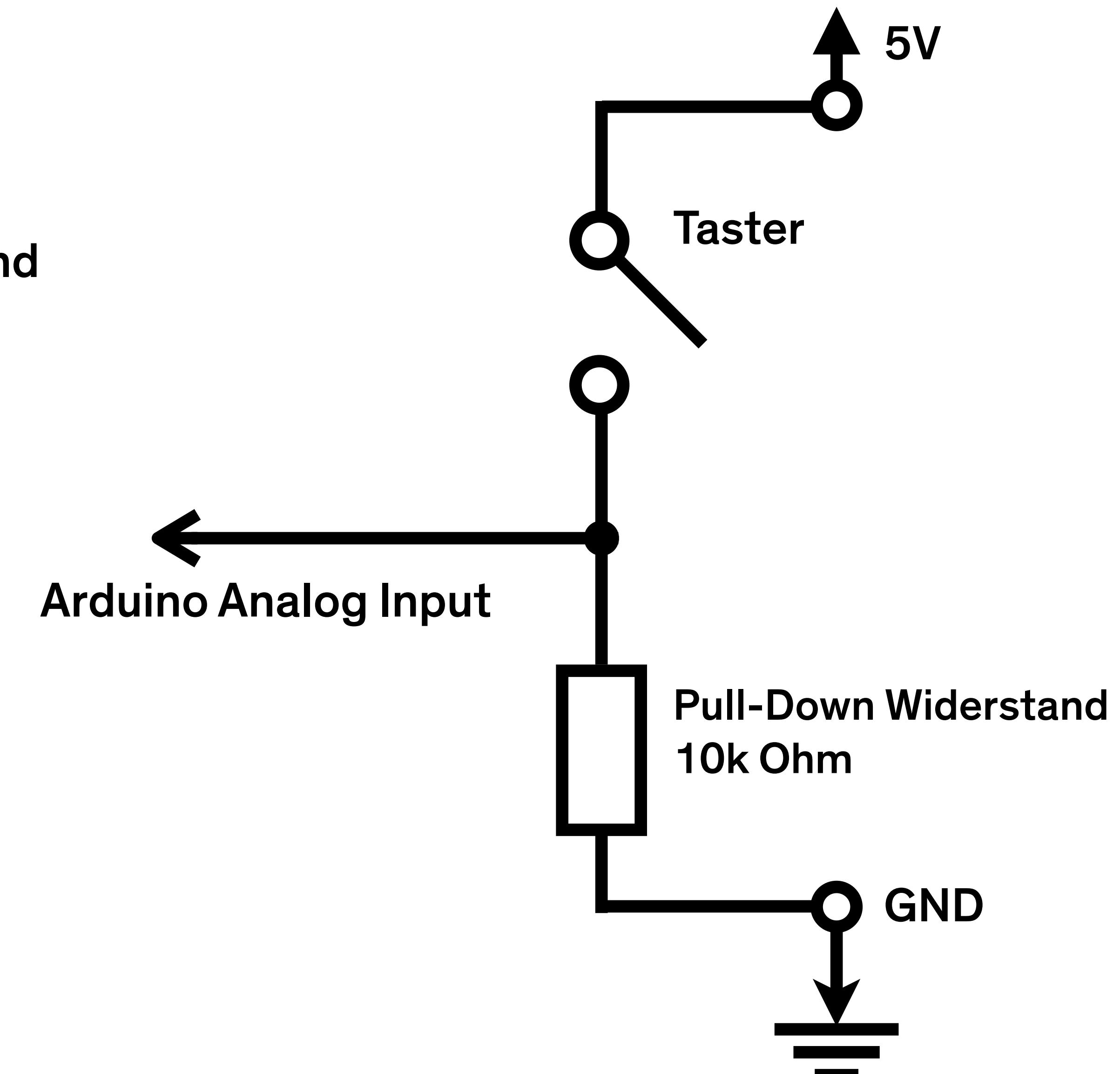
...

Datenblätter

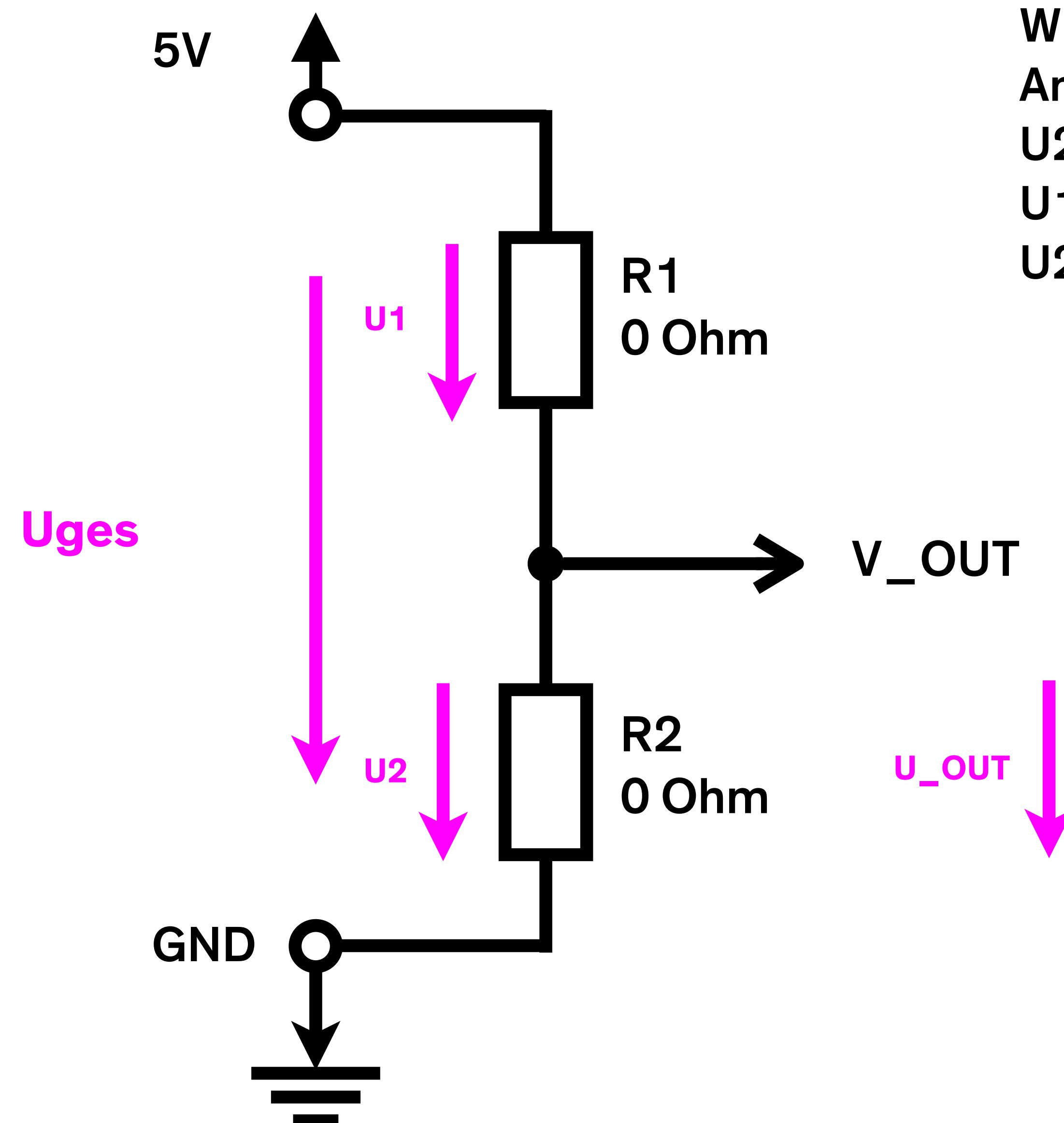
Pull-Up-Schaltung



Pull-Down-Schaltung



Grundschaltungen



Widerstandswerte müssen für die entsprechende Anwendung spezifisch gewählt werden

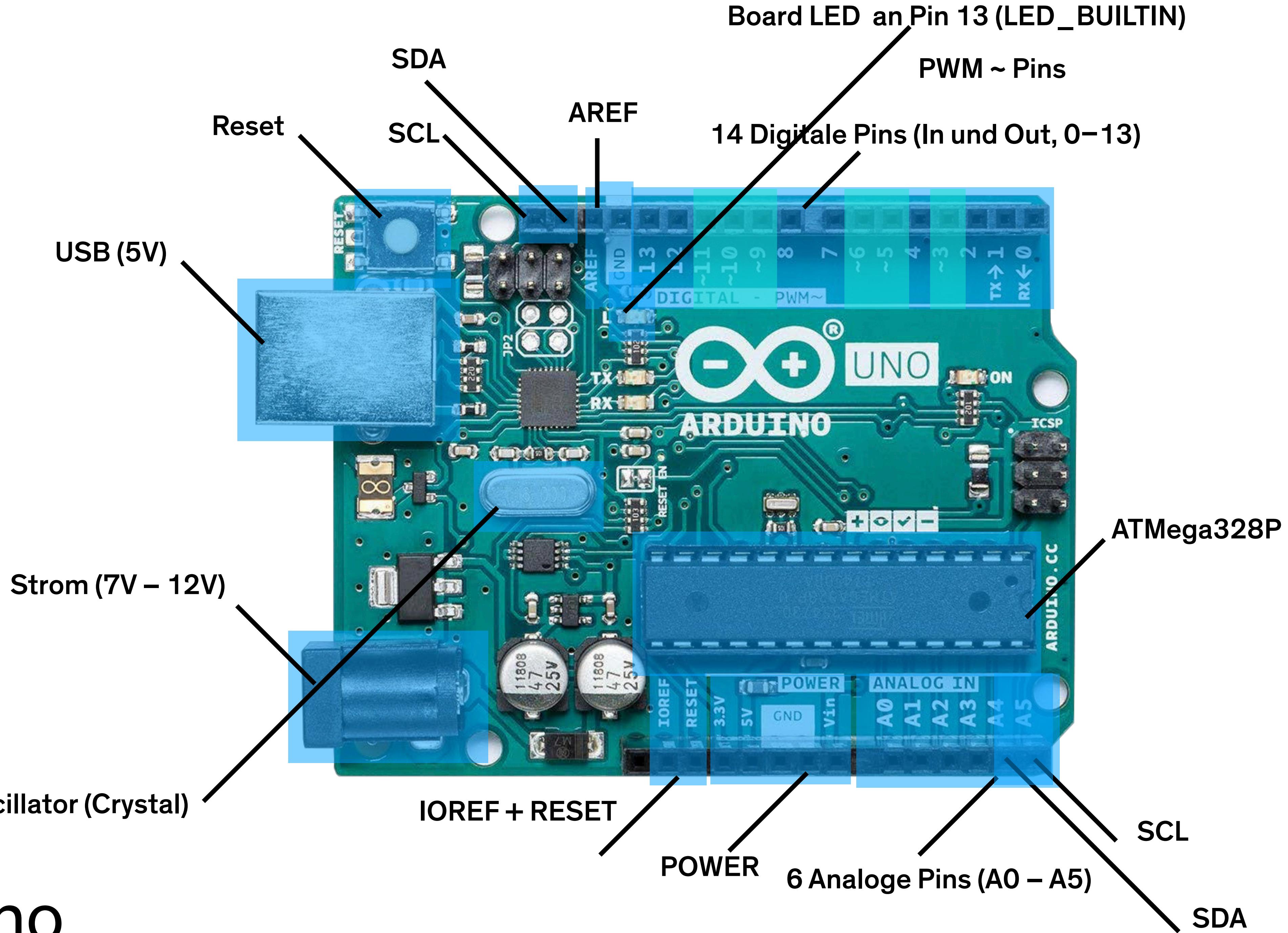
$$U_2 = U_{OUT}$$

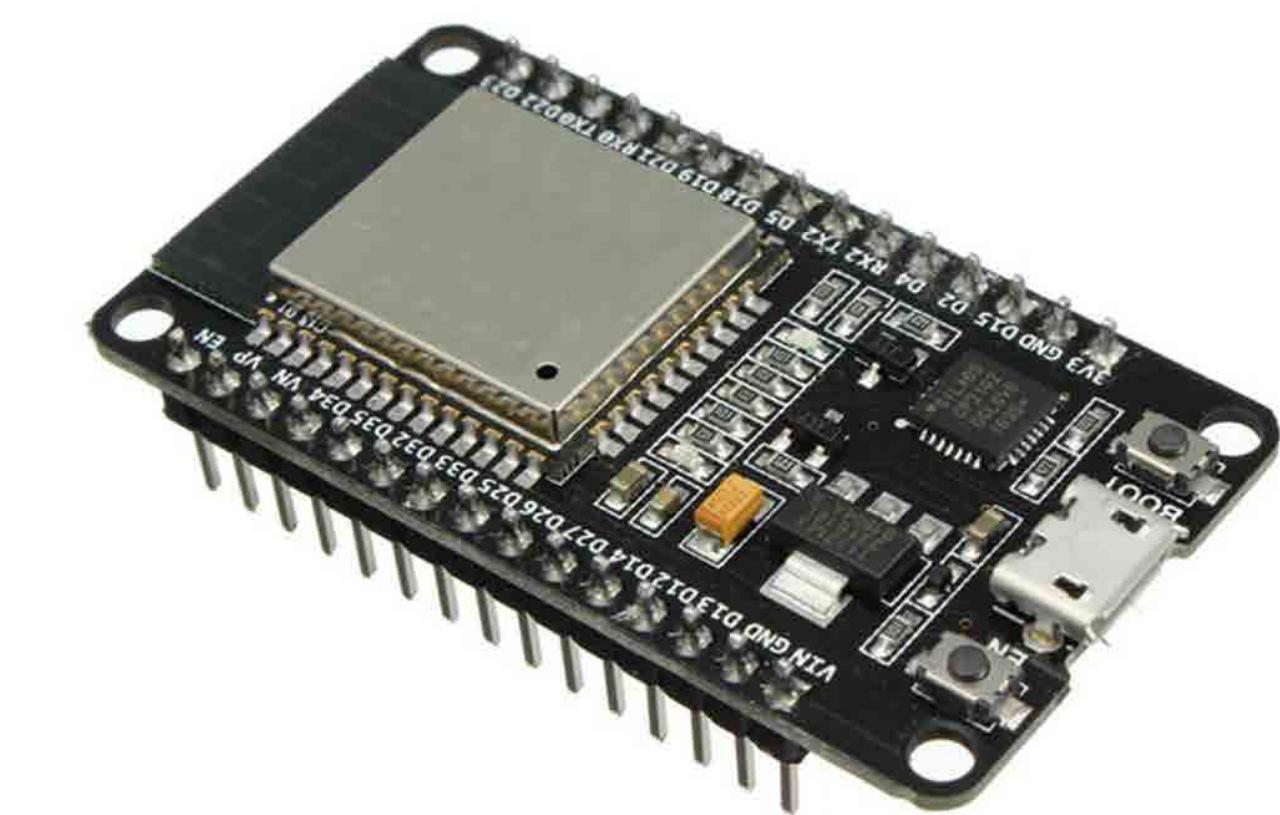
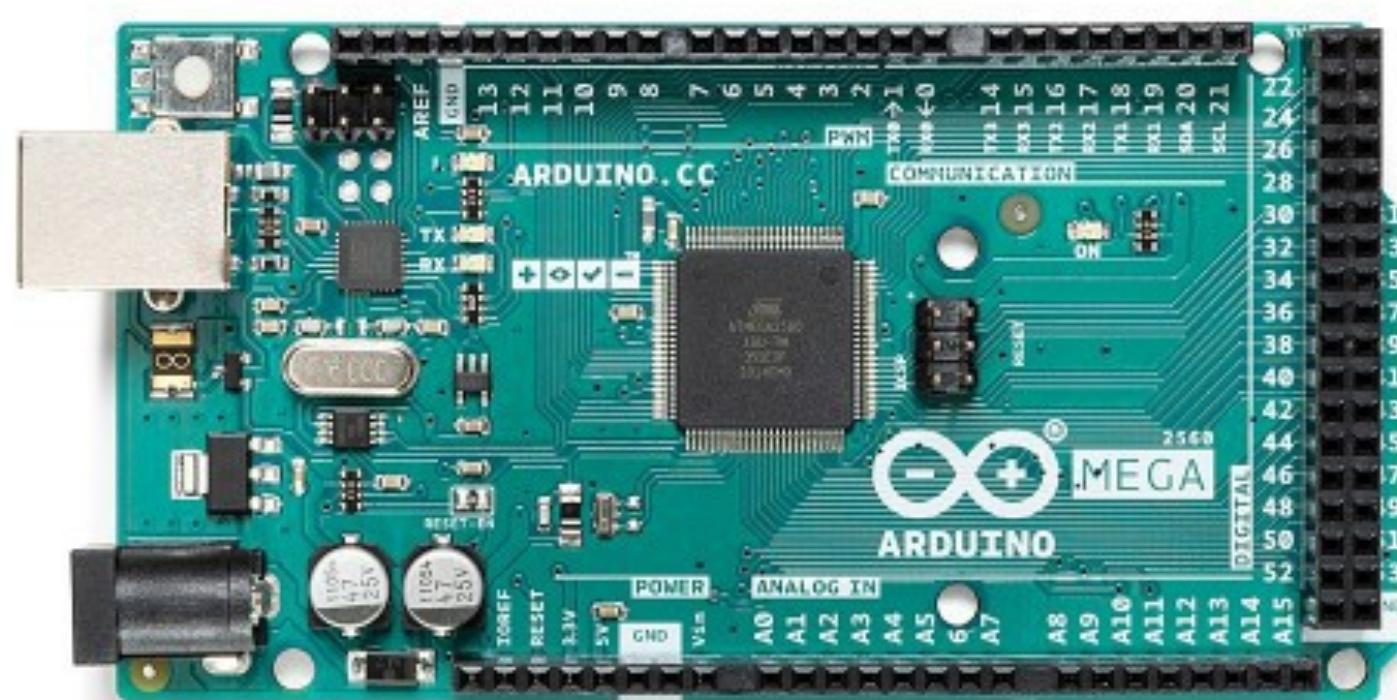
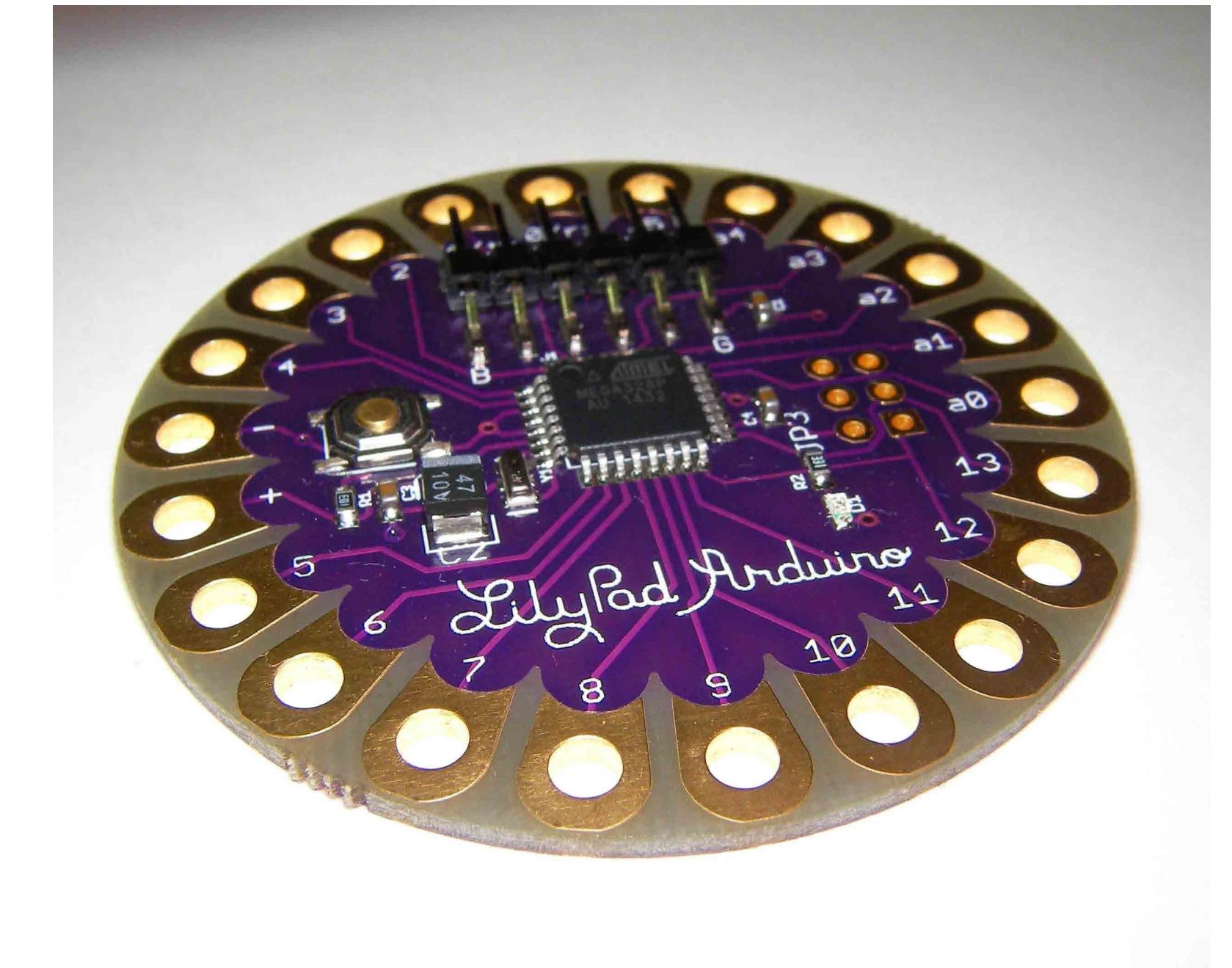
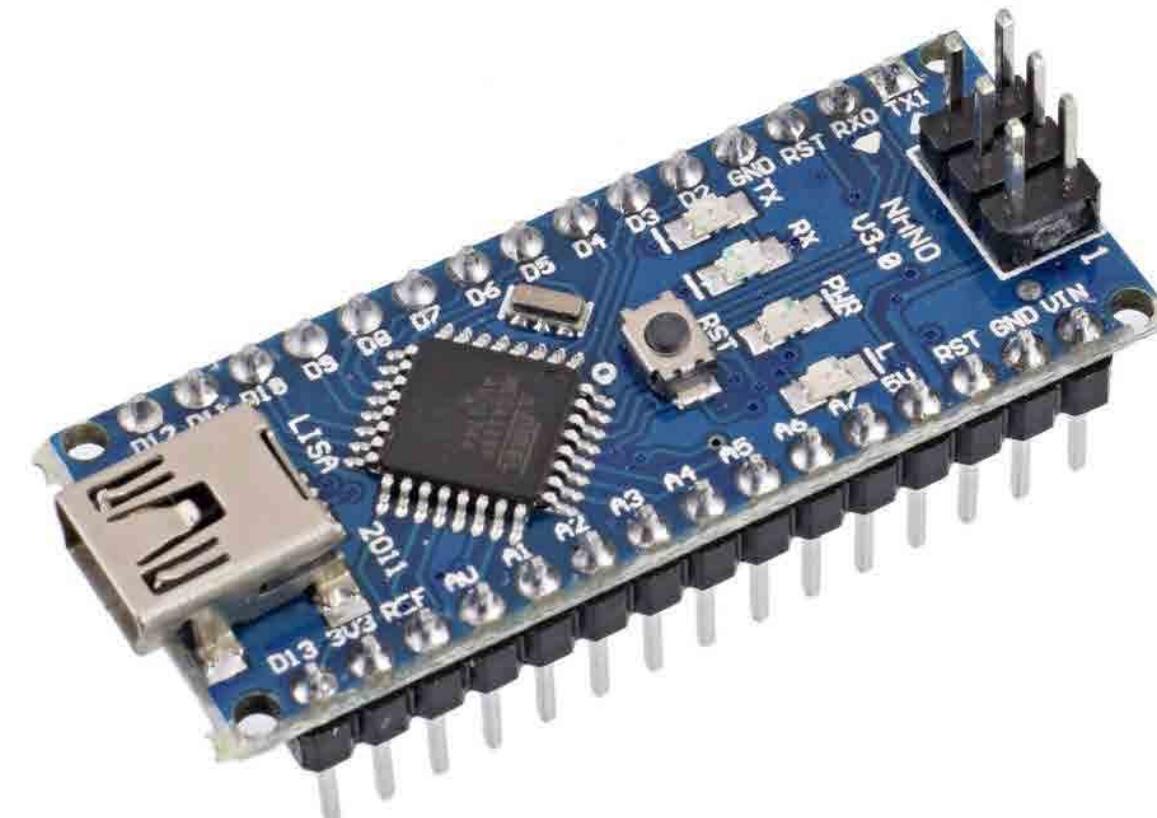
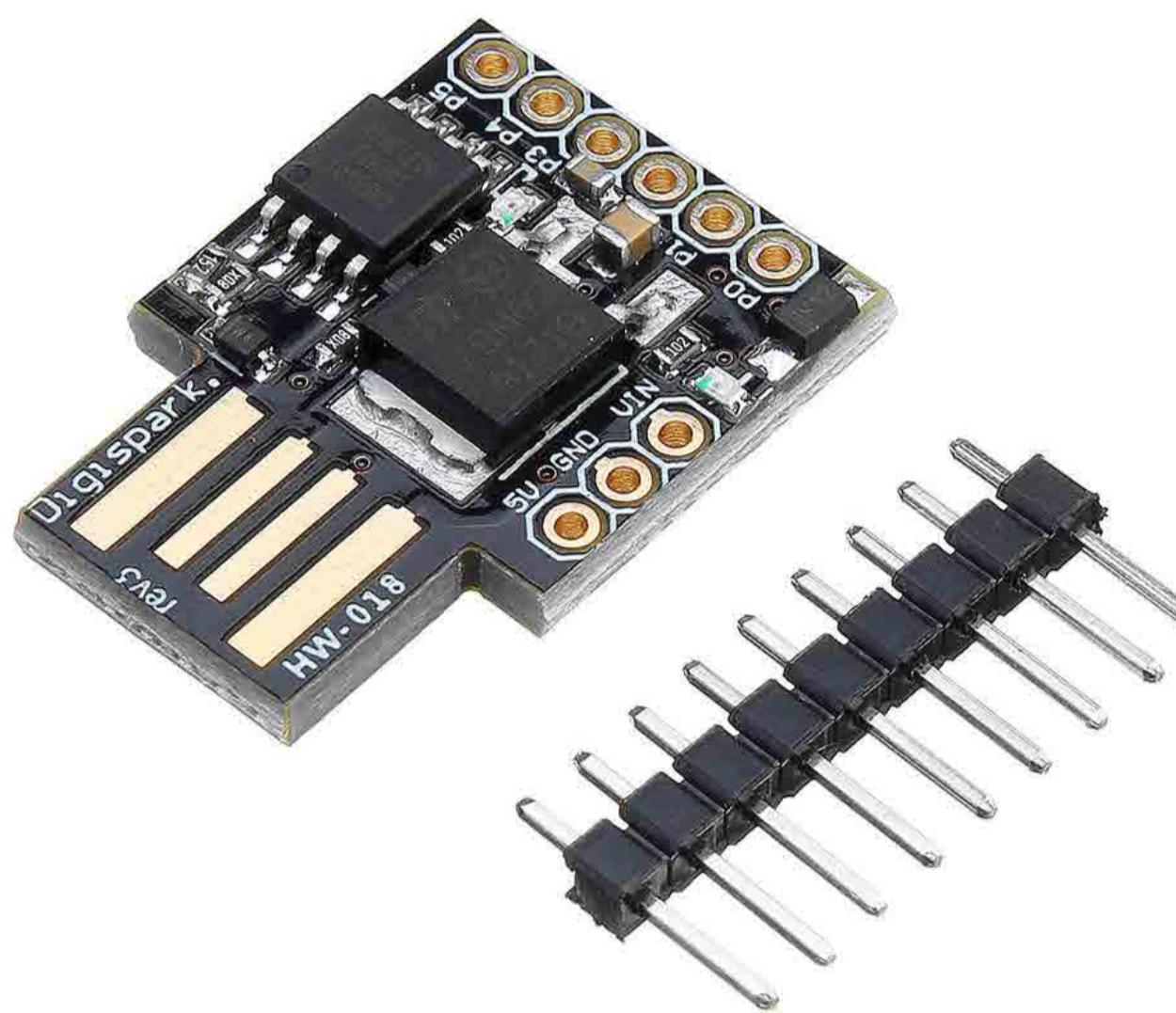
$$U_1 = (U_{ges} * R_1) / (R_1 + R_2)$$

$$U_2 = (U_{ges} * R_2) / (R_1 + R_2)$$

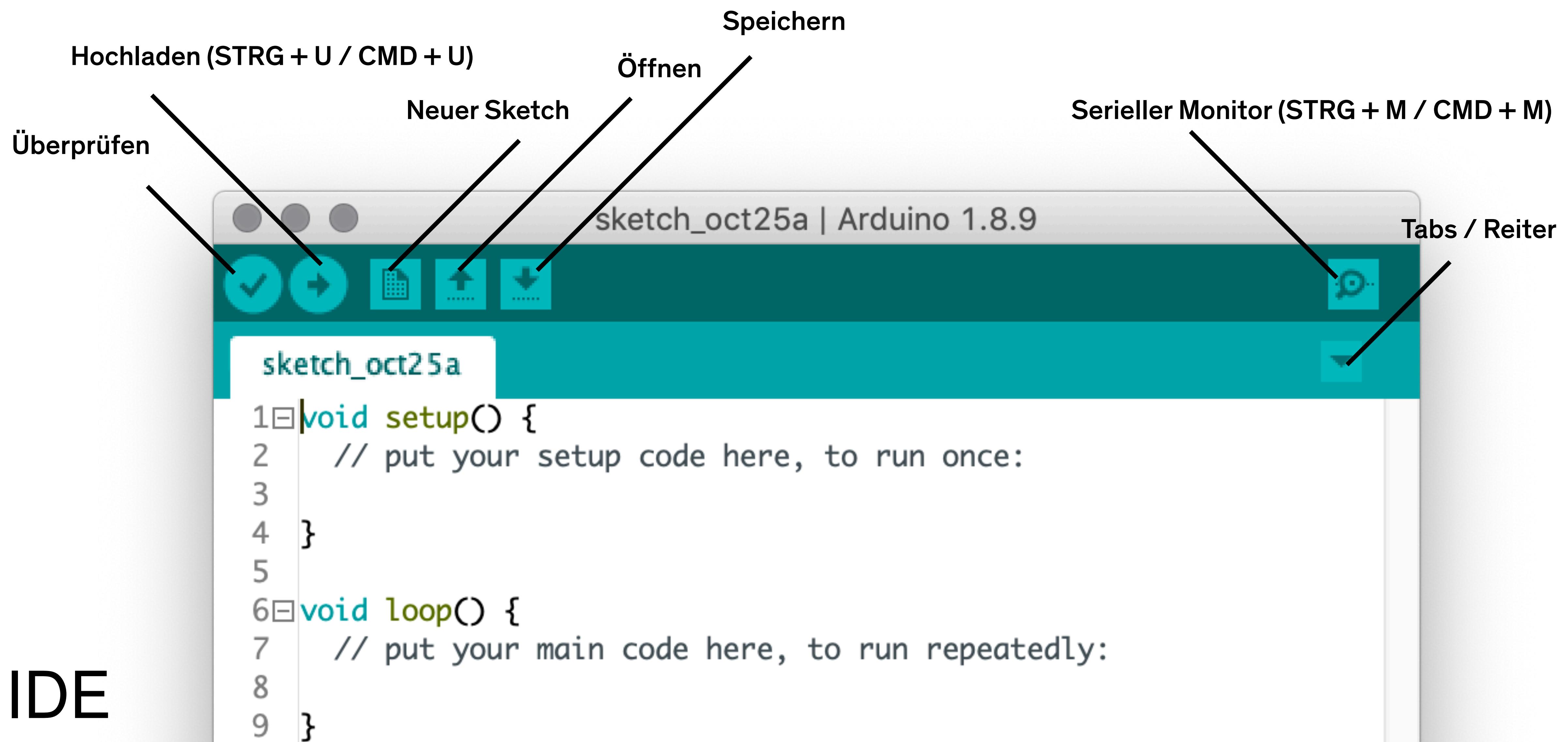
Spannungsteiler

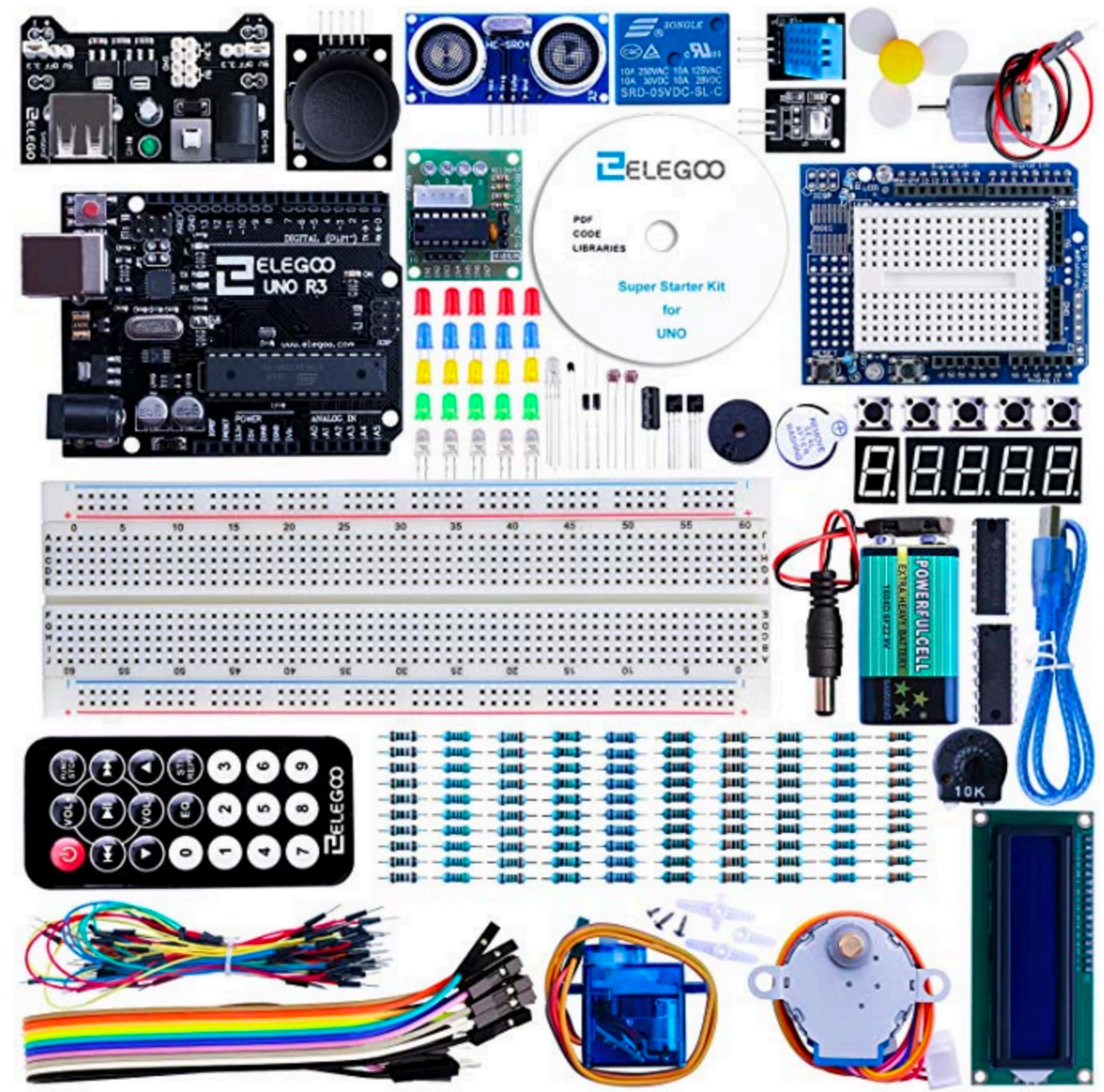
Arduino





Arduino Typen



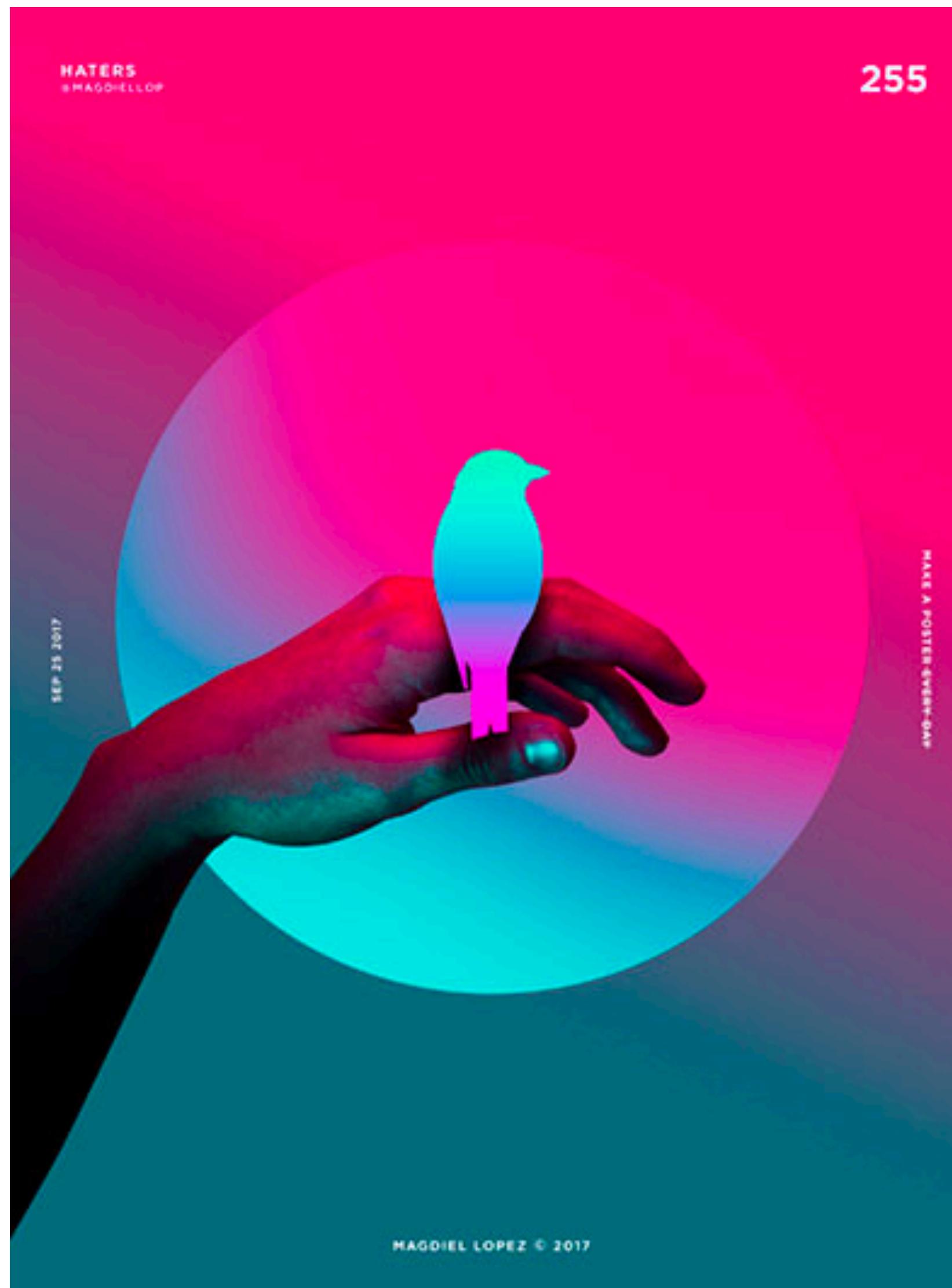


Arduino Kit

Warum Physical Computing?



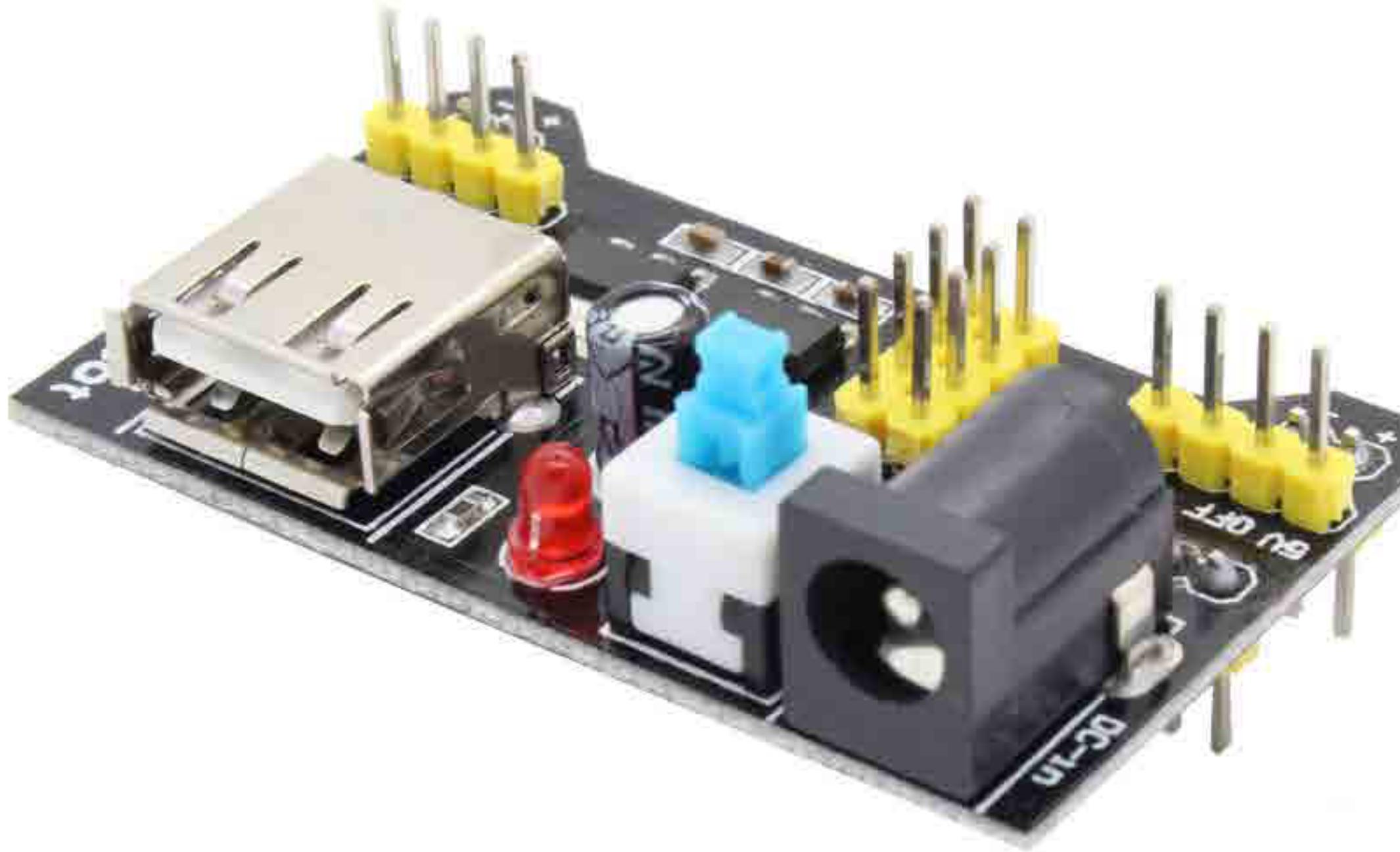
Warum Physical Computing?



Warum Physical Computing?

Arduino Bauteile

Spannungsversorgung



Bietet 3.3V und 5V (via Jumper Einstellungen)

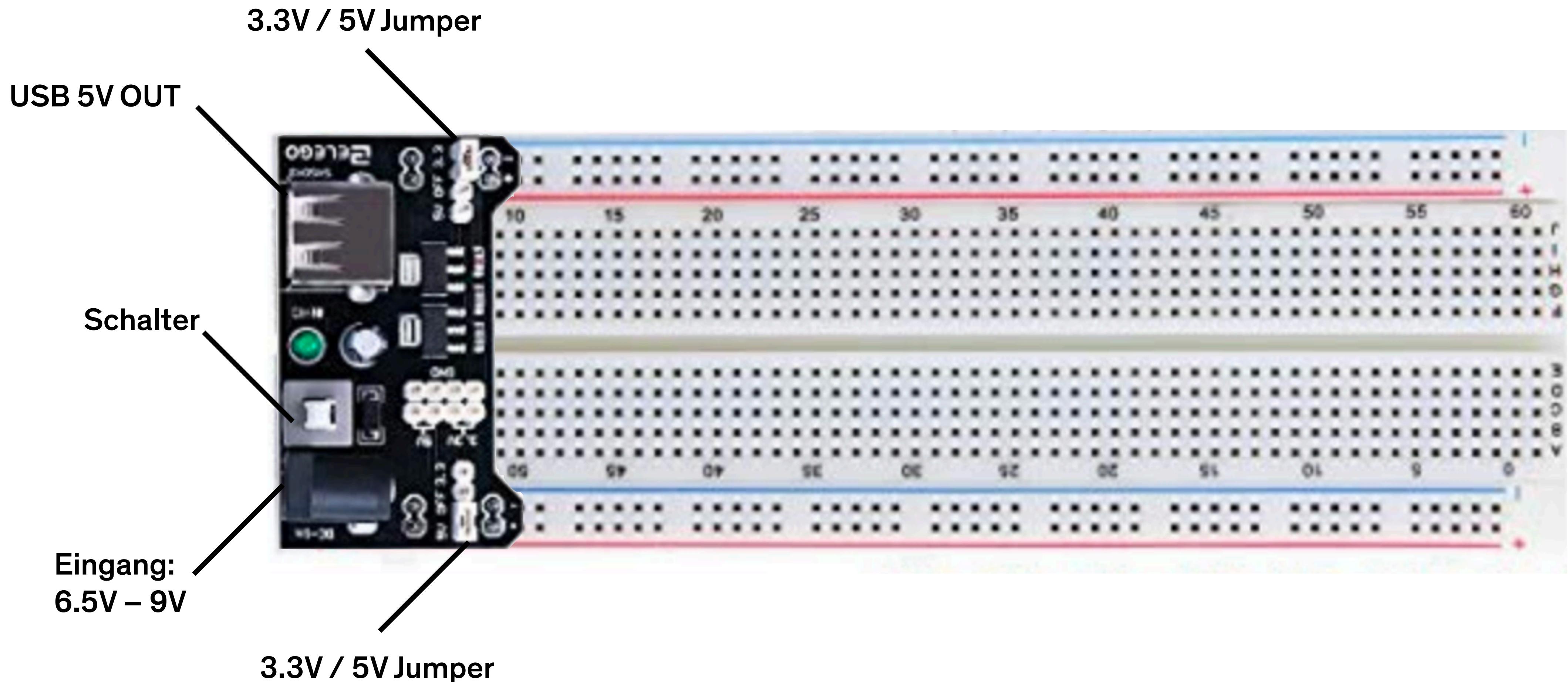
Breadboard-Kompatibel

Ein- und Ausschalter

USB- und Hohlstecker

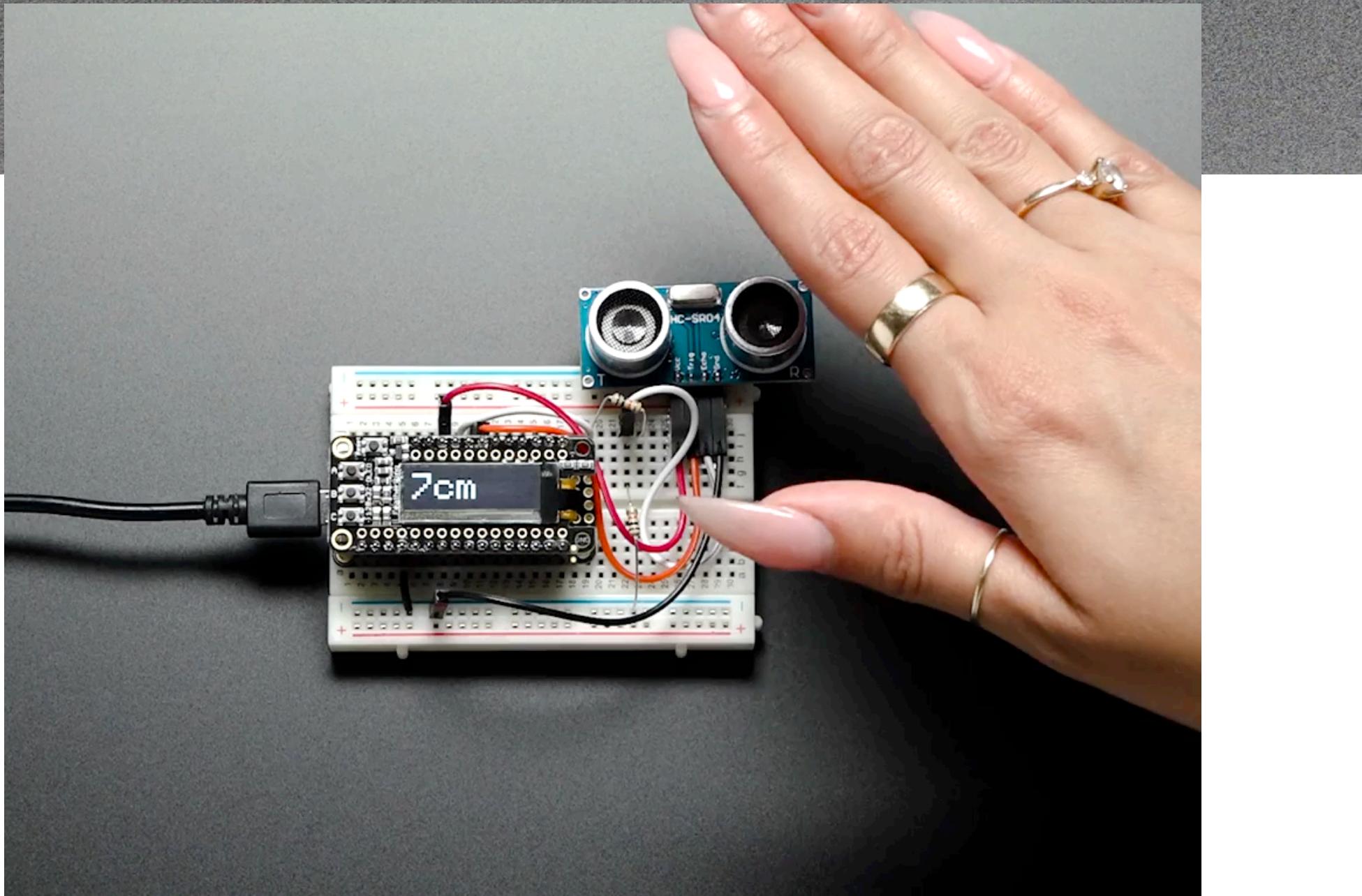
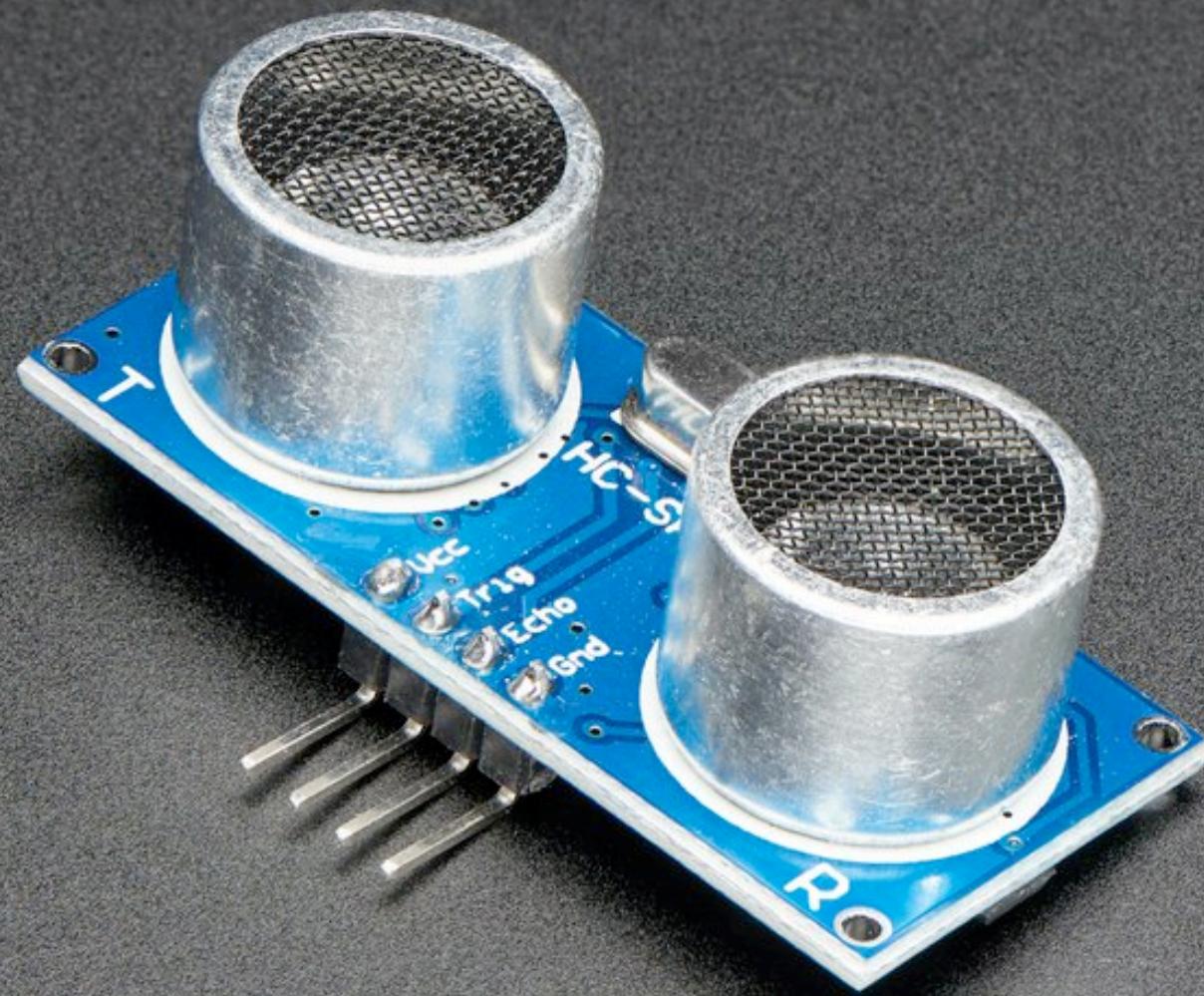
Arduino Bauteile

Spannungsversorgung



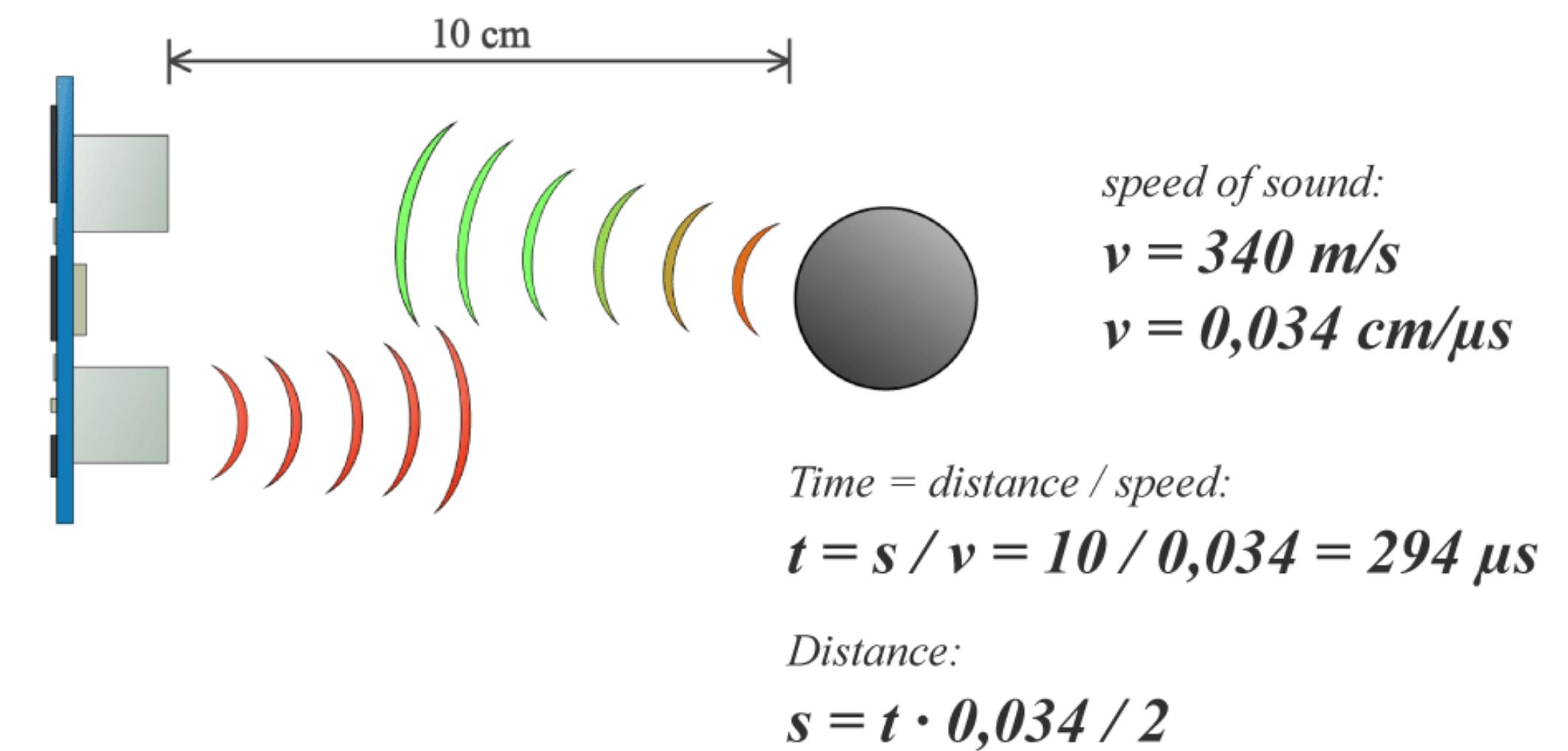
Arduino Bauteile

Ultraschallsensor



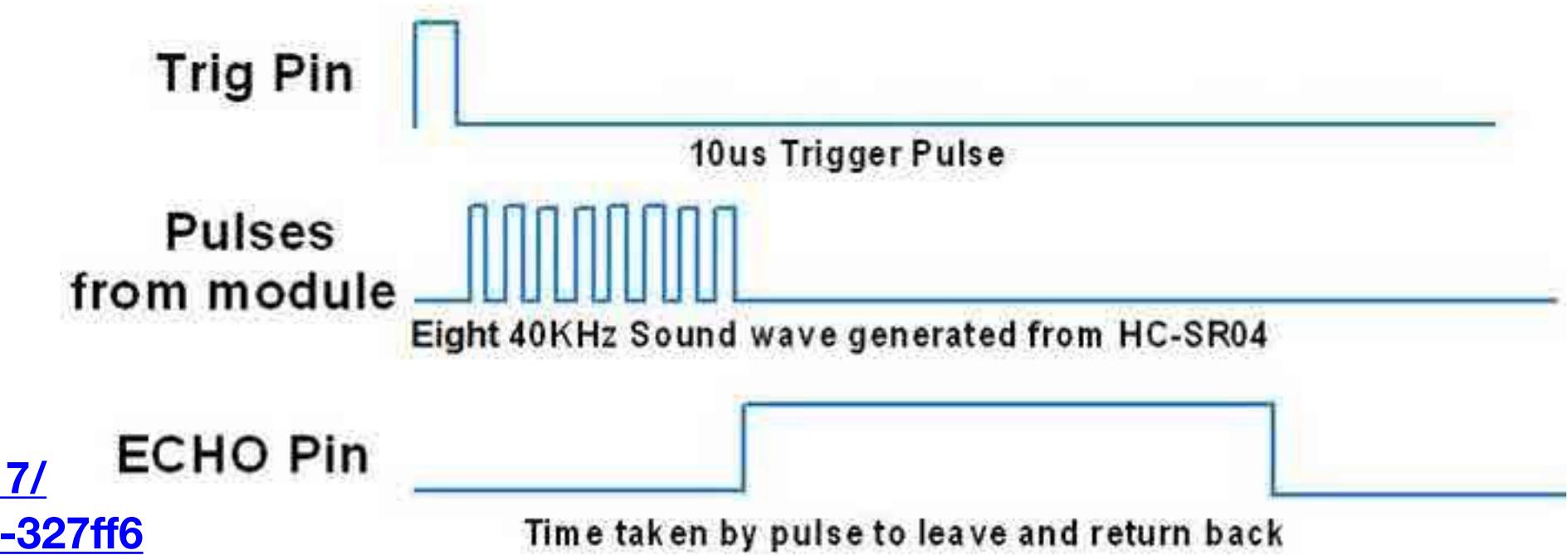
Arduino Bauteile

<https://create.arduino.cc/projecthub/abdularbi17/ultrasonic-sensor-hc-sr04-with-arduino-tutorial-327ff6>

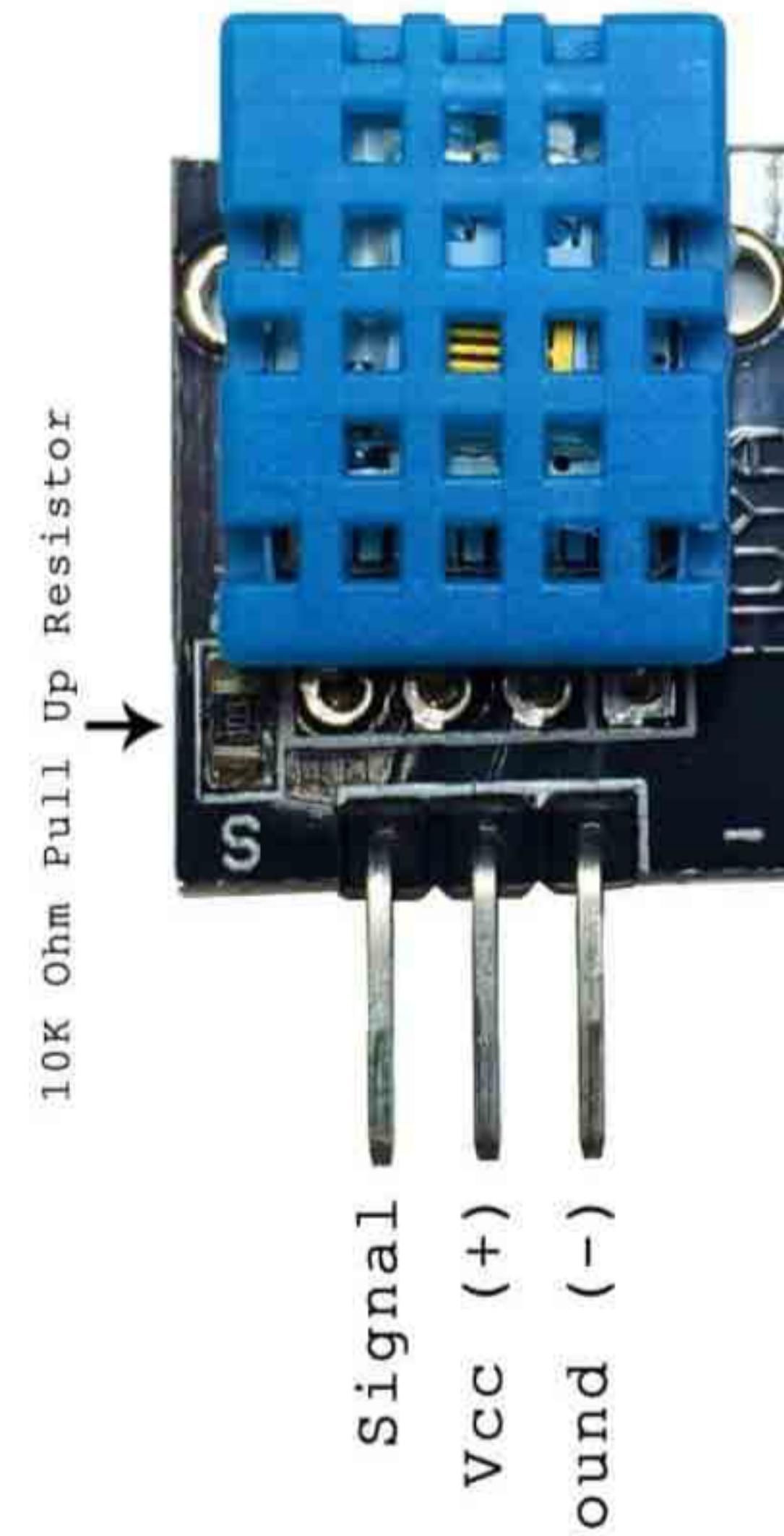


Messen von Abständen via Ultraschall

Ultrasonic HC-SR04 module Timing Diagram



Ultraschallsensor DHT-11

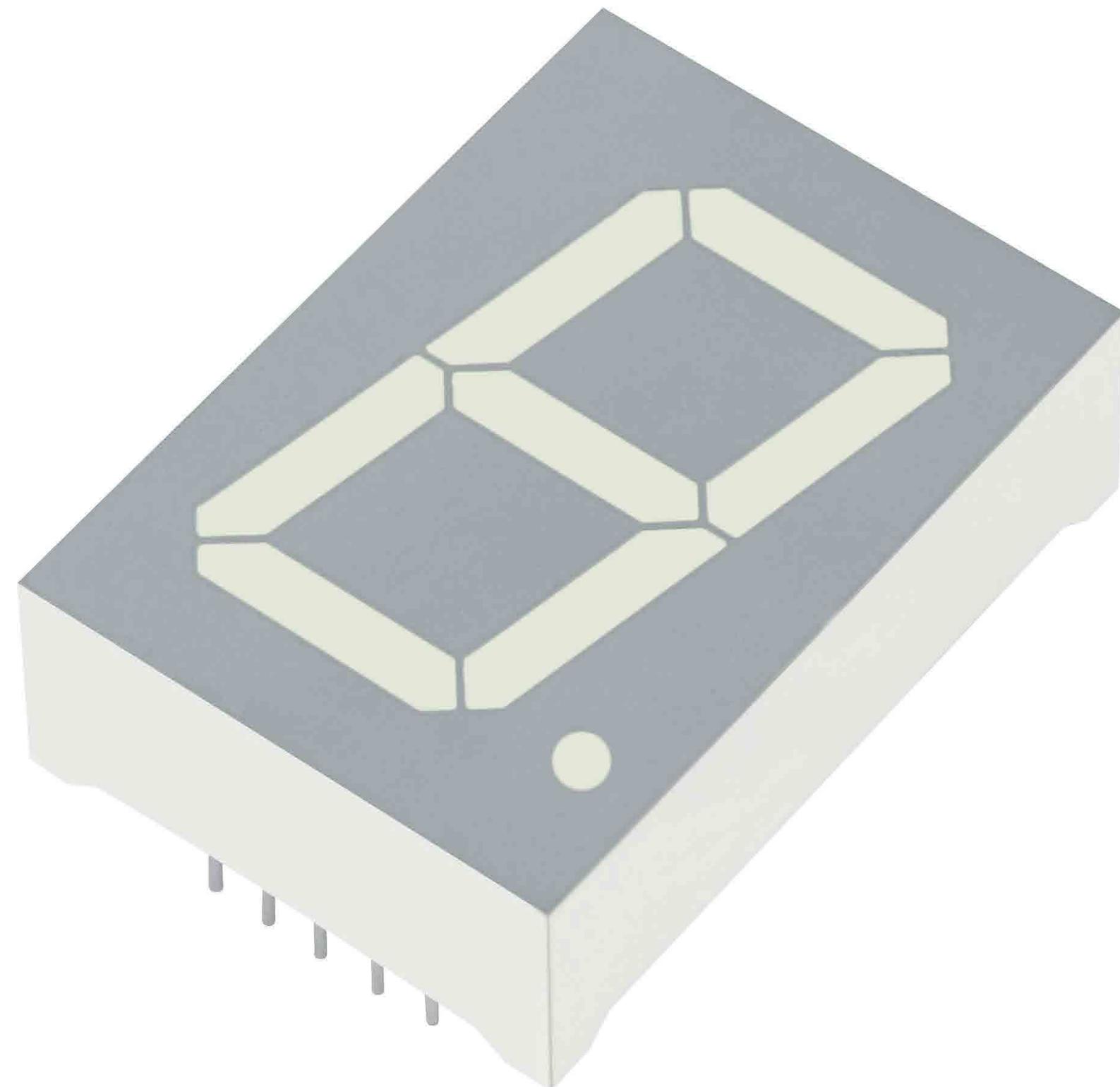


**Kombinierter Sensor zum Messen von
Temperatur & Feuchtigkeit**

Luftfeuchtigkeitsbereich ca. 20% – 80%

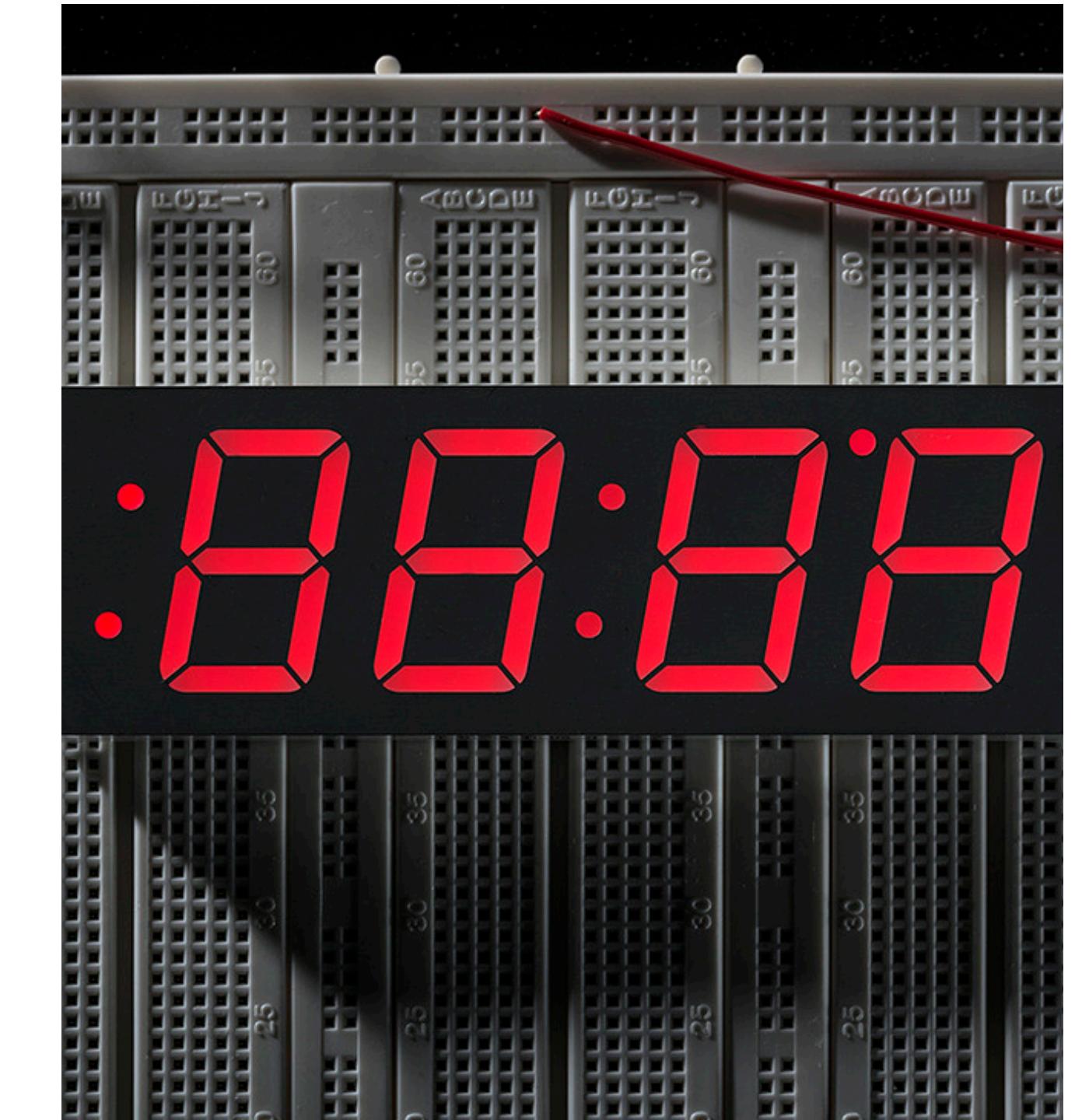
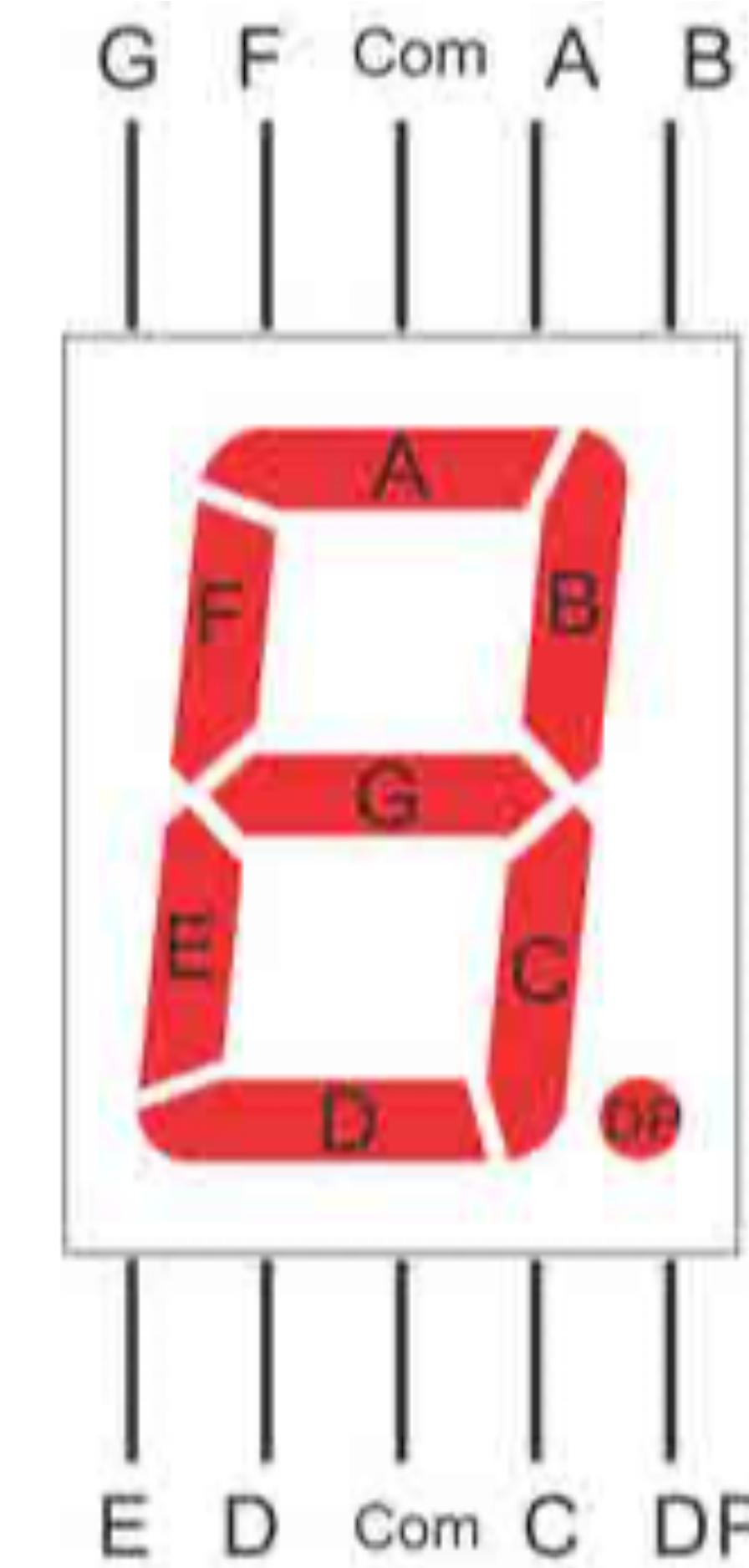
Temperaturbereich ca. 0 – 50°C

7-Segment Anzeige



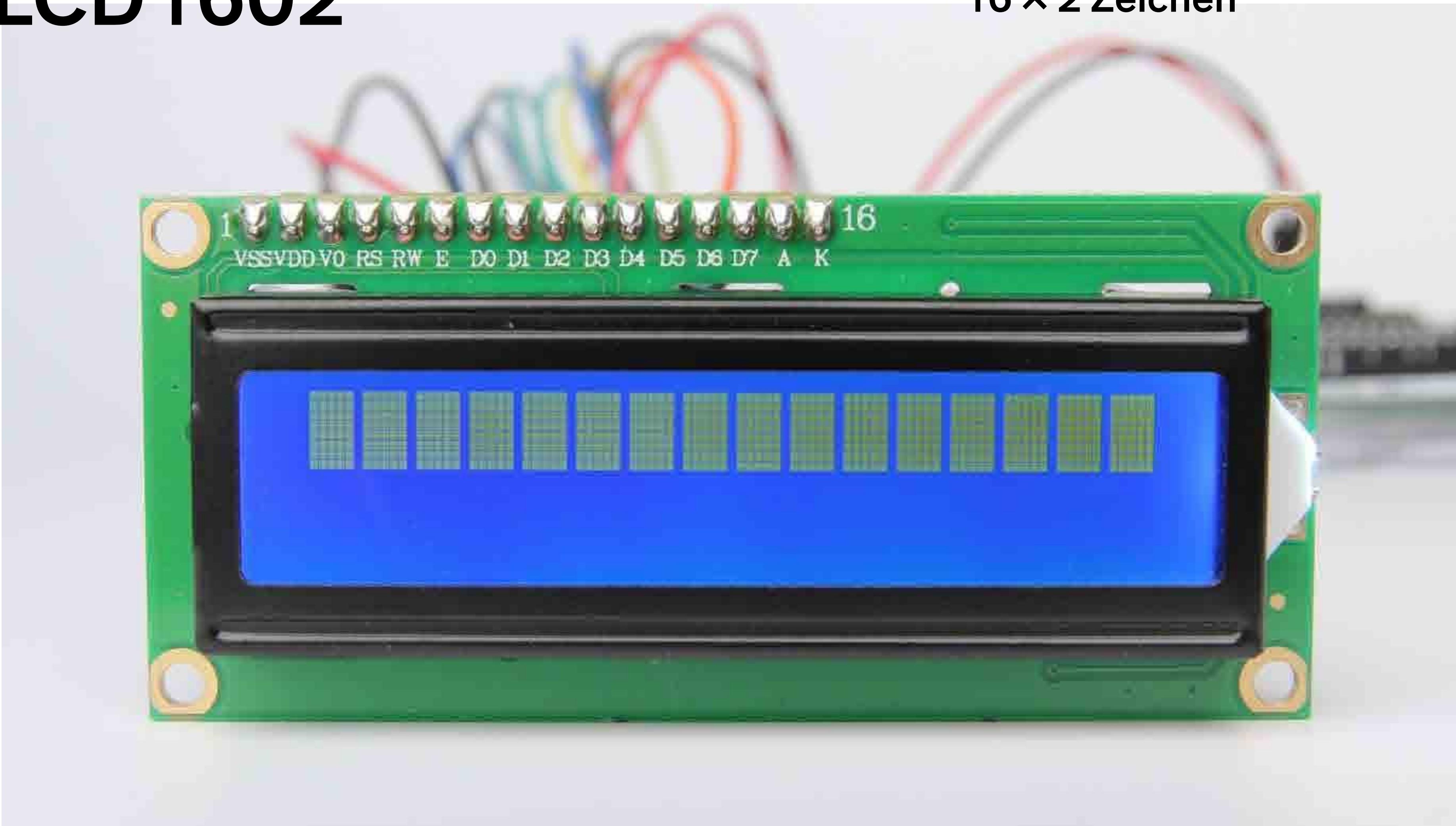
Anzeige von Zahlen

Einzelne oder verkettete Bausteine



LCD-Anzeige LCD1602

Anzeige von Zahlen und Buchstaben
16 × 2 Zeichen



LED versch. Farben



Arduino Bauteile

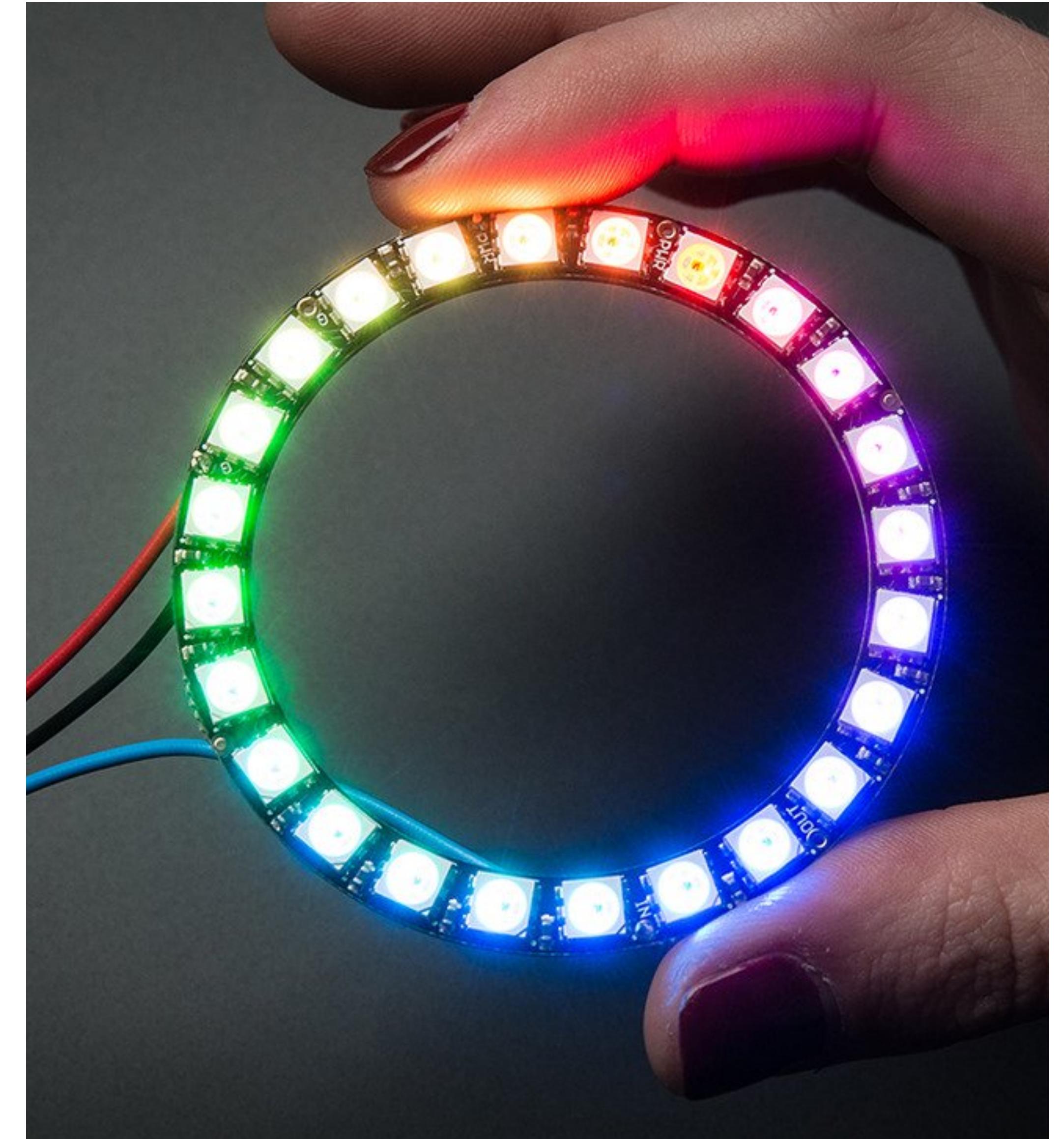
Neopixel



Arduino Bauteile

<https://funduino.de/nr-17-ws2812-neopixel>

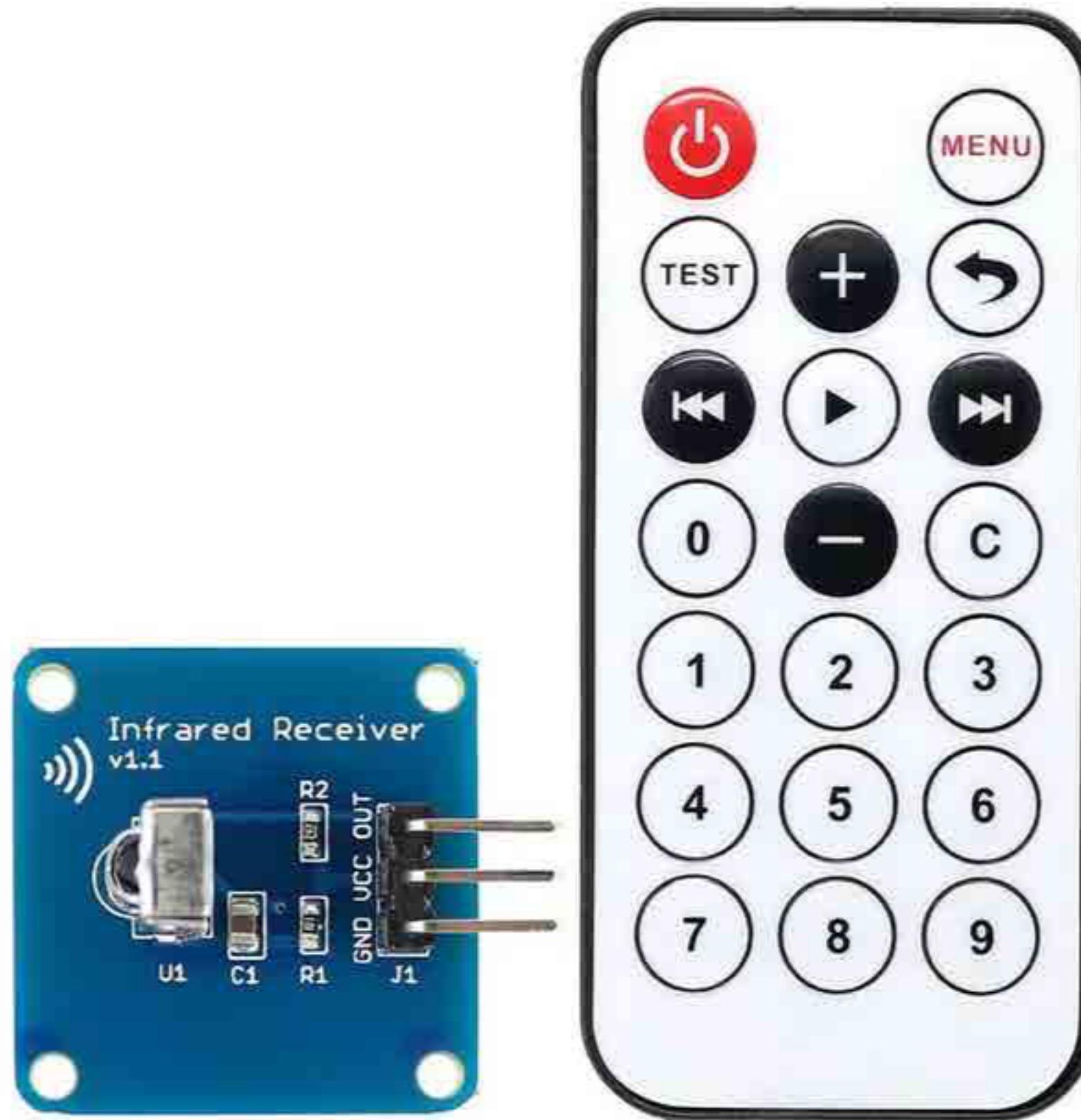
Neopixel



Arduino Bauteile

<https://funduino.de/nr-17-ws2812-neopixel>

IR-Receiver + Fernbedienung



Senden und Empfangen von Infrarot Signale

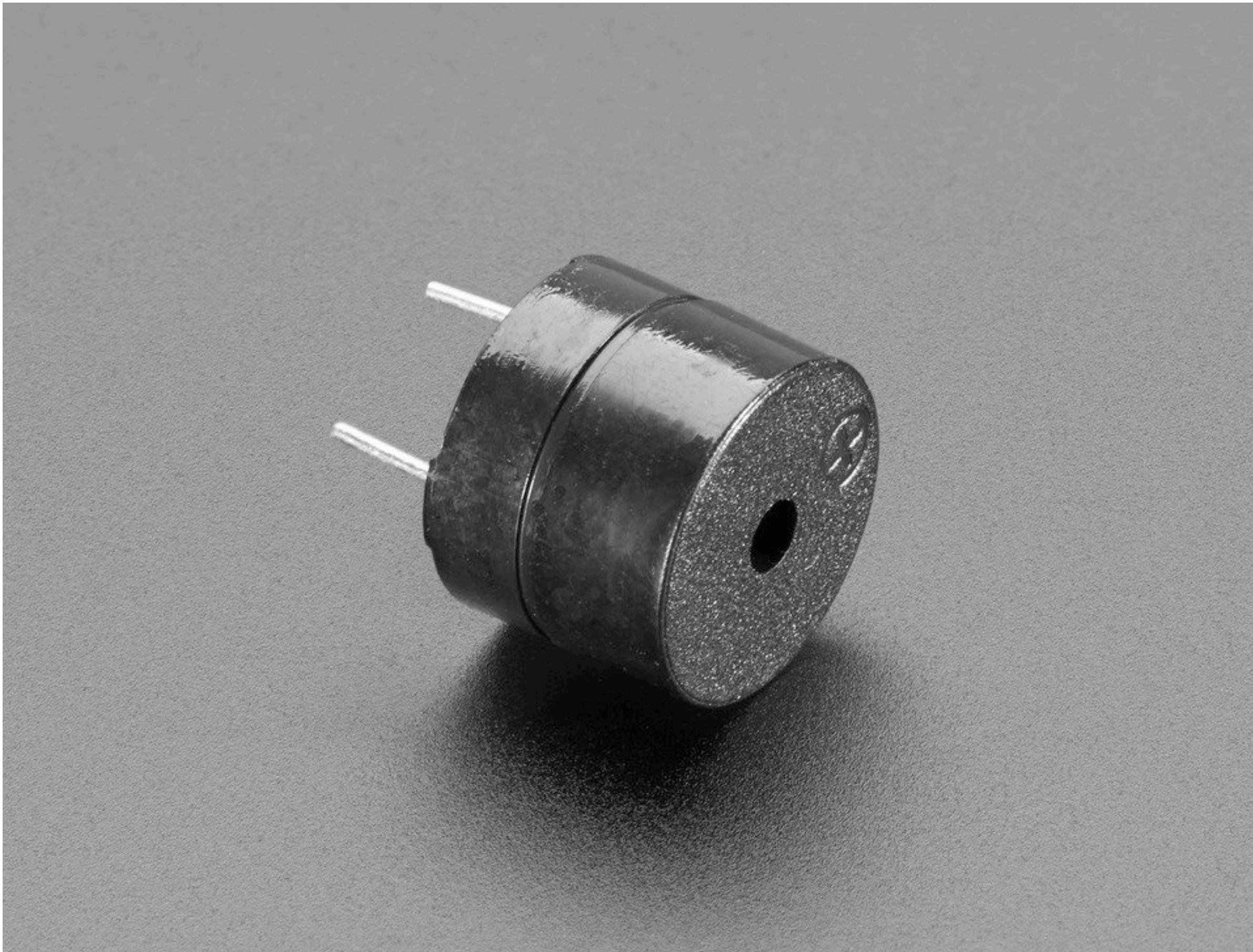
Fernsteuern des Arduino möglich

Verschiedene Signale möglich

Buzzer

Erzeugen von Tönen

Passiv / Aktiv möglich



Joystick-Modul

Bewegung in 2 Achsen möglich

Eingabemöglichkeit durch herunterdrücken



Arduino Bauteile

<https://funduino.de/nr-22-joystick-modul>

DC Motor

2 Drehrichtungen

Keine Positionsbestimmung

Achtung:

**DC Motoren nur über Treiber ansteuern,
ansonsten kann es zu Schäden am Arduino führen**

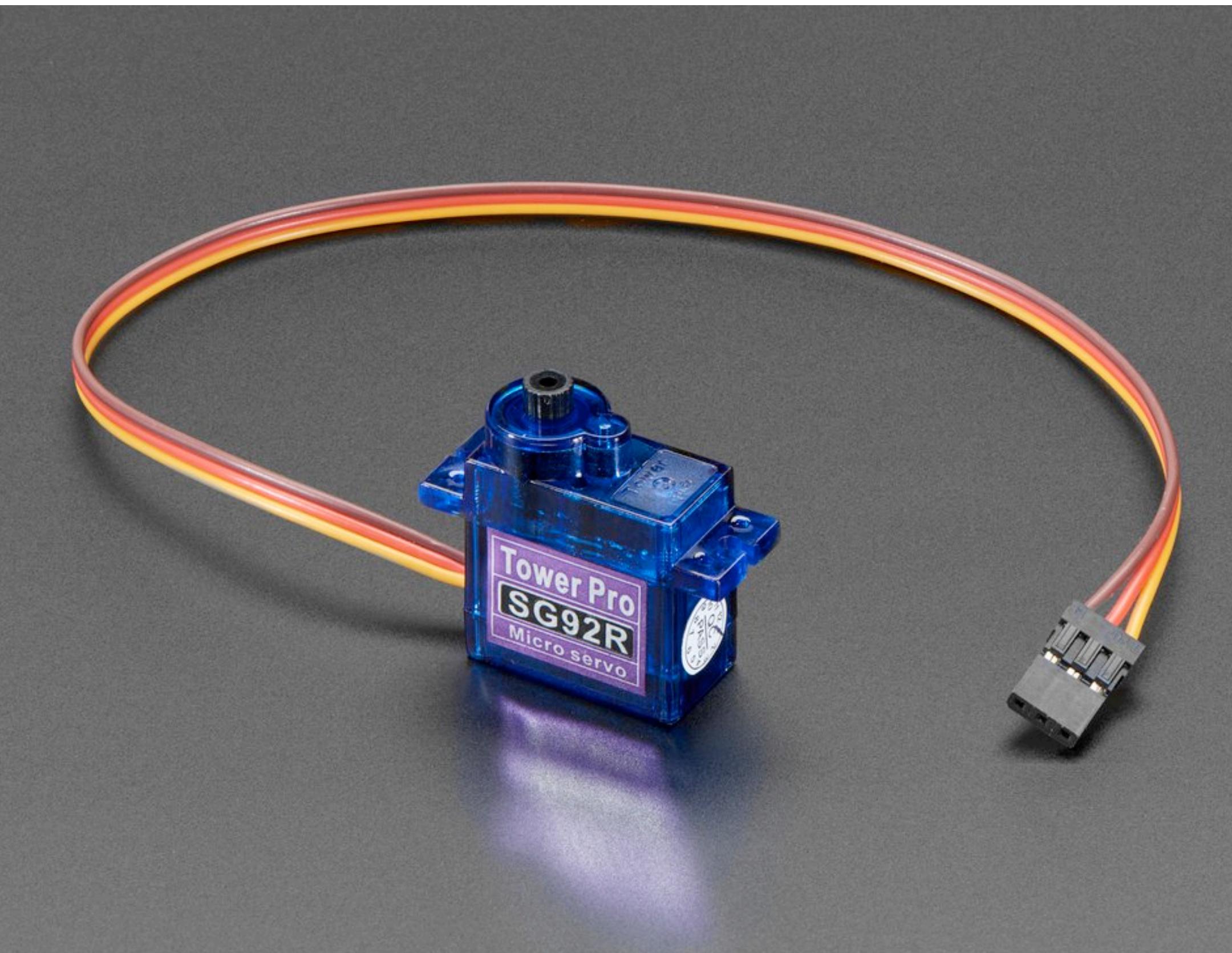
Polarisiert: Positive und negative Anschlußklemme



Arduino Bauteile

<https://starhardware.org/motorsteuerung-direkt-per-arduino/>

Servomotor SG90



2 Drehrichtungen

Rotation zwischen 0° und 180°

Enthält Zahnräder und eine Welle

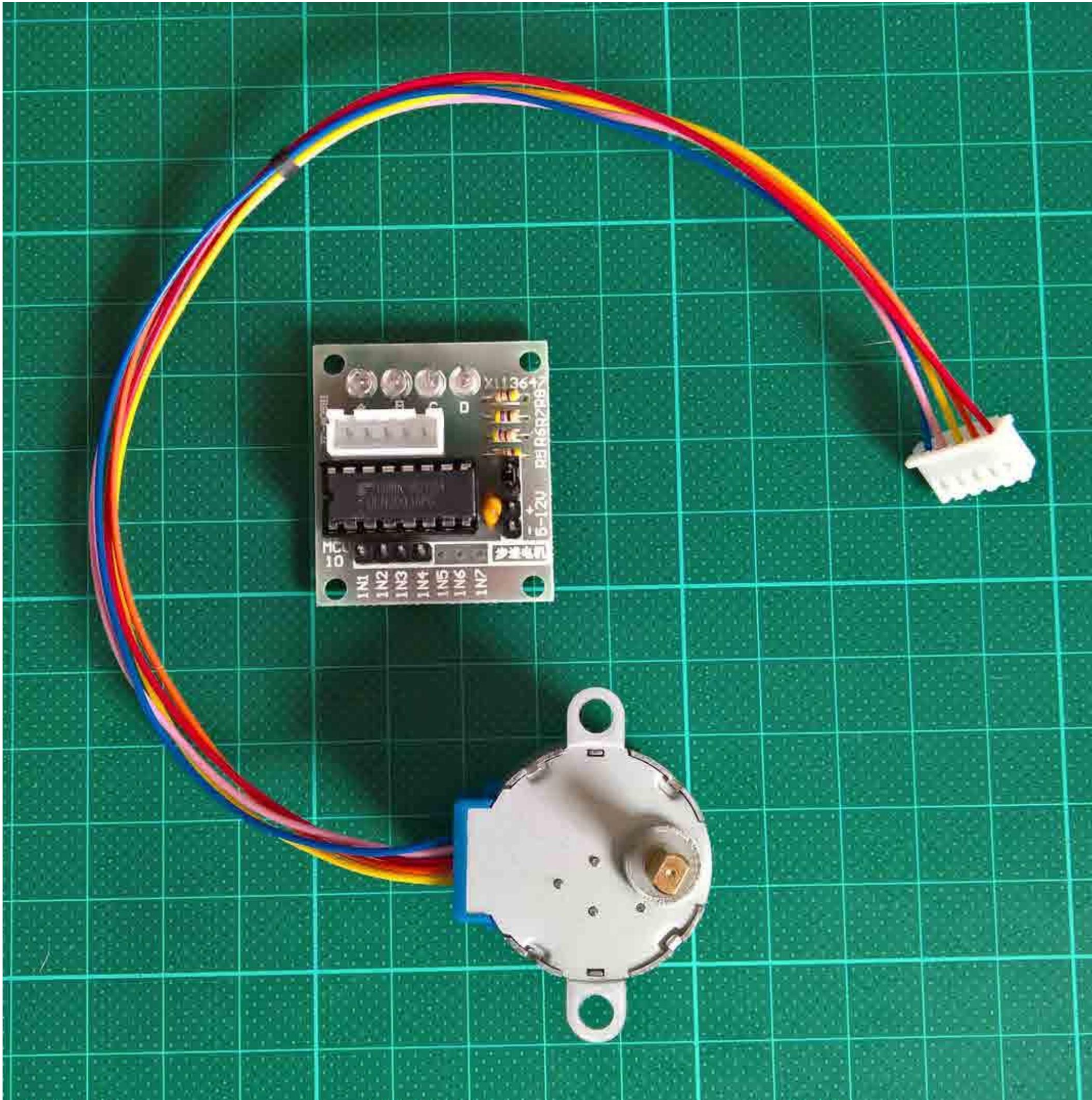
Genaue Positionierung über IC + Potentiometer

Kann direkt an den Arduino angeschlossen werden

Besser: Servomotor an separate Spannungsversorgung

Schrittmotor (Stepper)

ULN2003 Baustein



2 Drehrichtungen

Rotation in 360°

Treiber Baustein notwendig

Sehr genaue Positionierung

Enthält ein Getriebe

Positionierung in feinen Schritten

Schrittmotor (Stepper)

NEMA Typ

2 Drehrichtungen

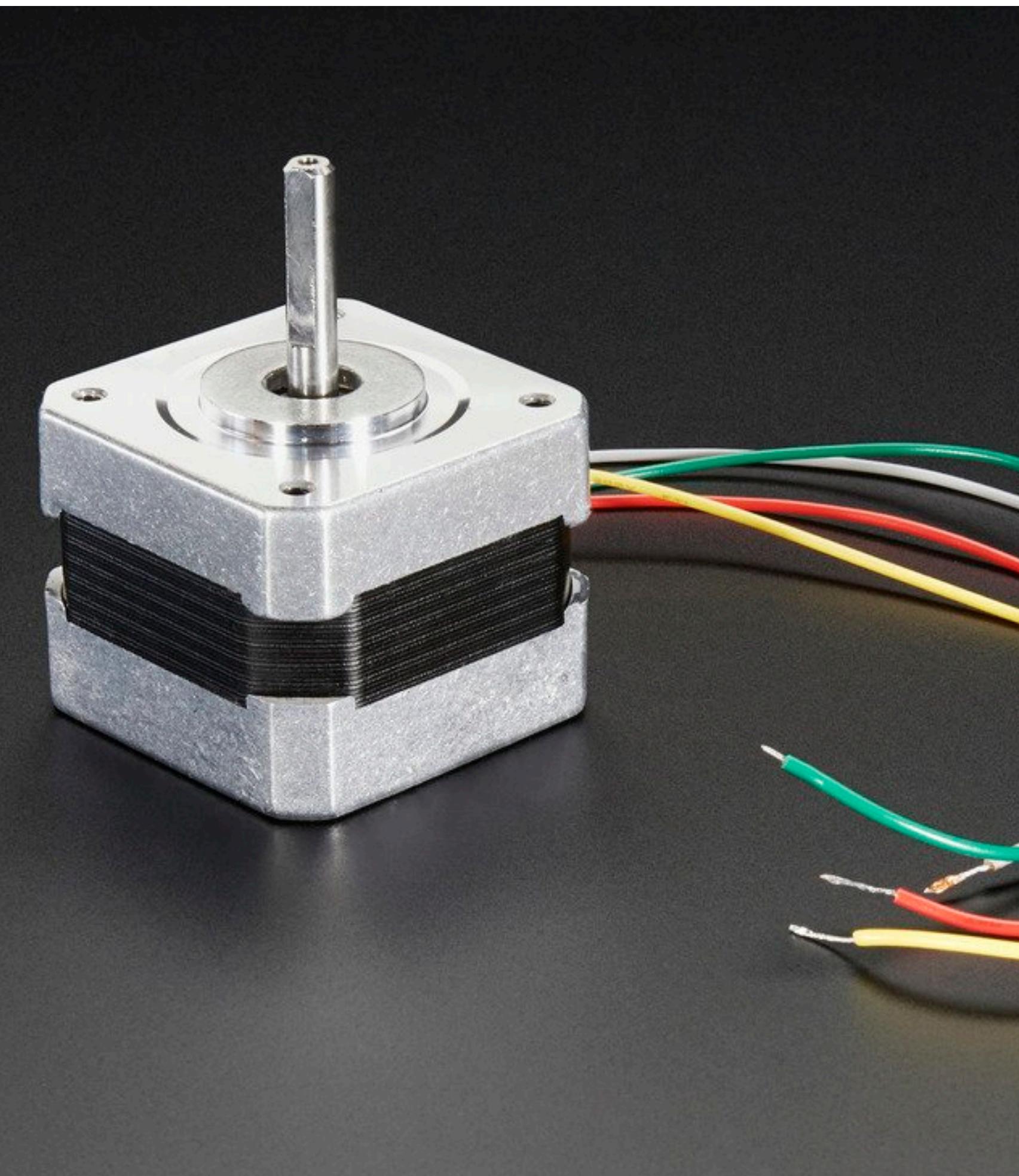
Rotation in 360°

Treiber Baustein notwendig

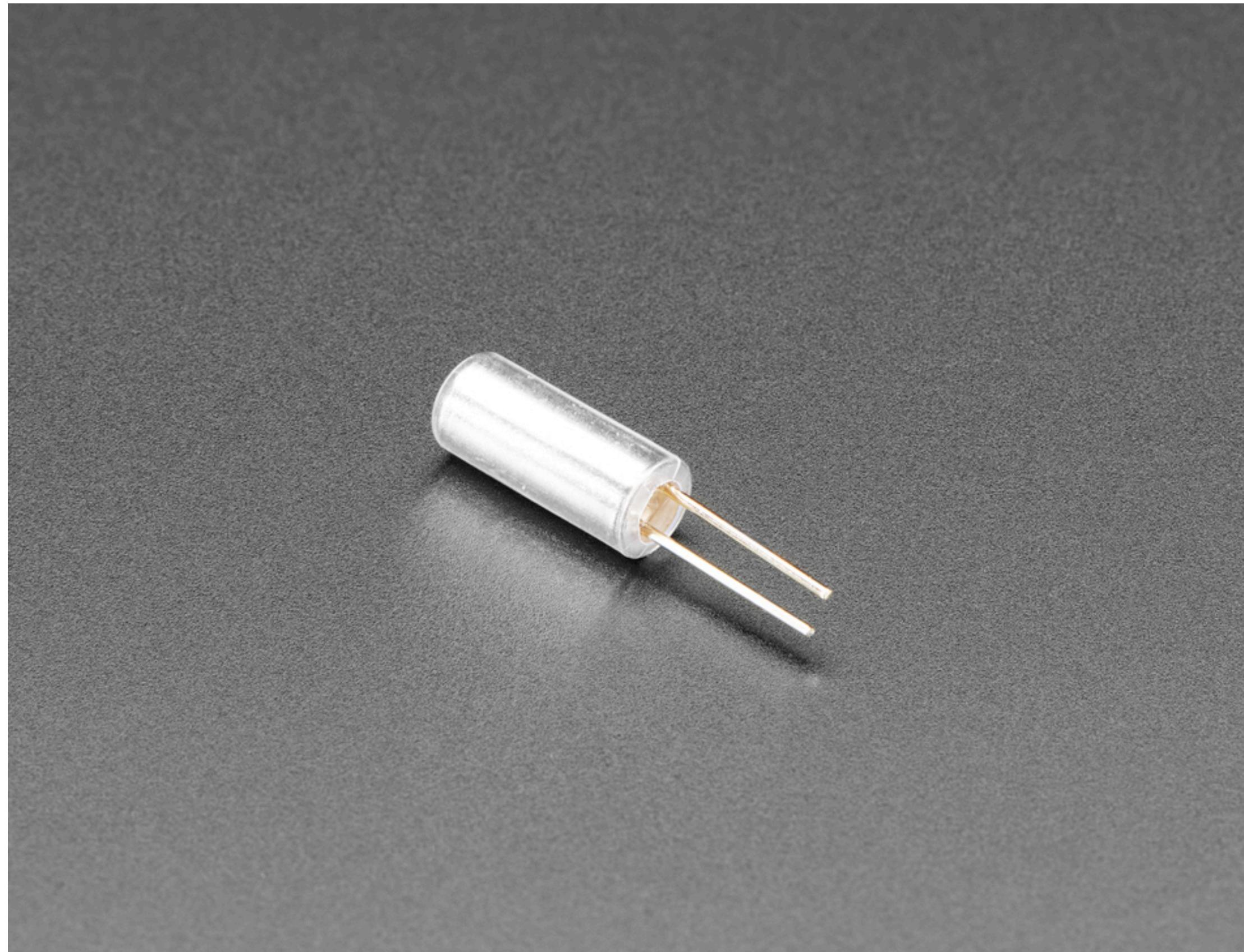
Sehr genaue Positionierung

Enthält ein Getriebe

Positionierung in feinen Schritten



Neigungssensor



Erkennt physische Neigung

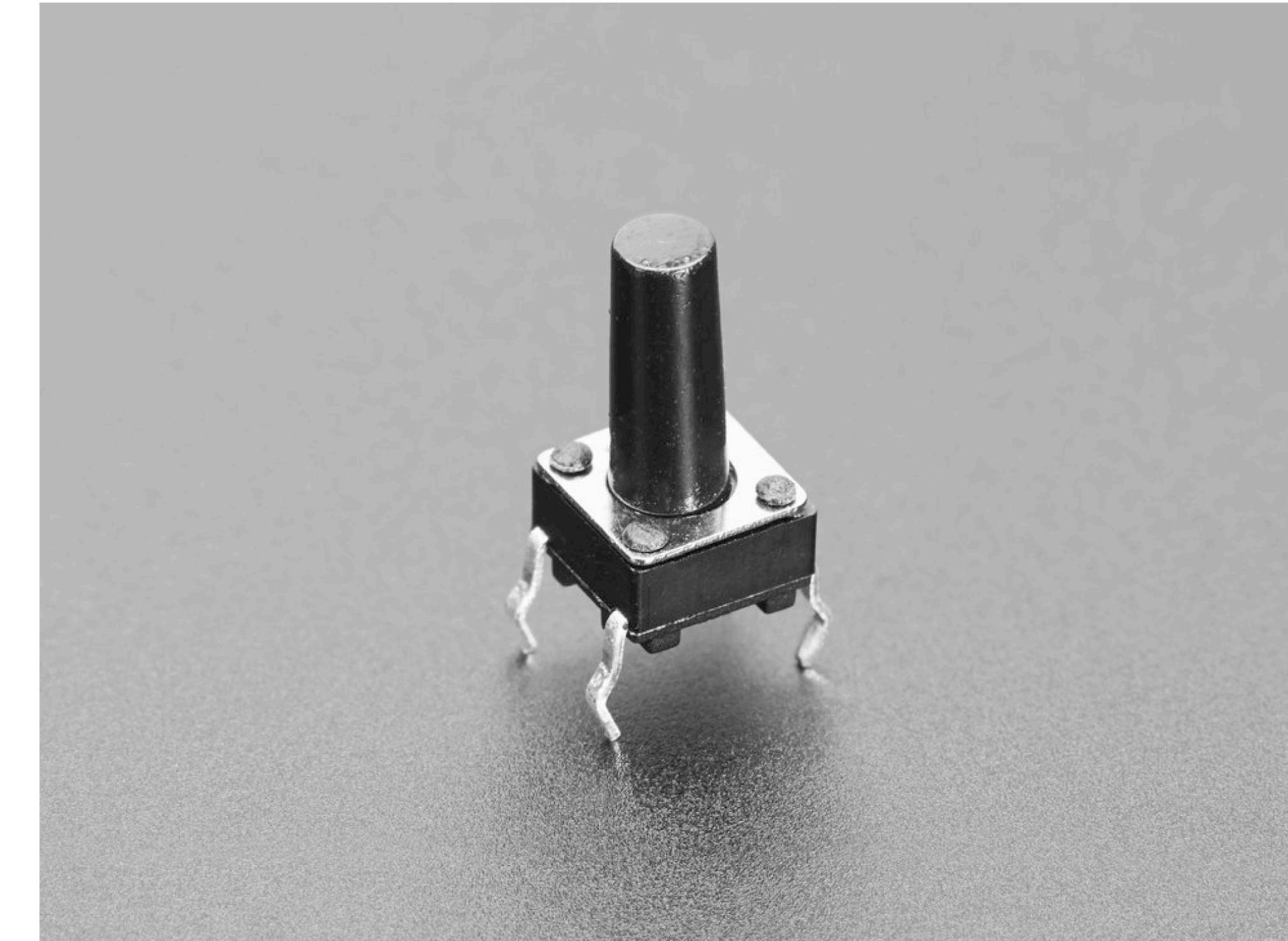
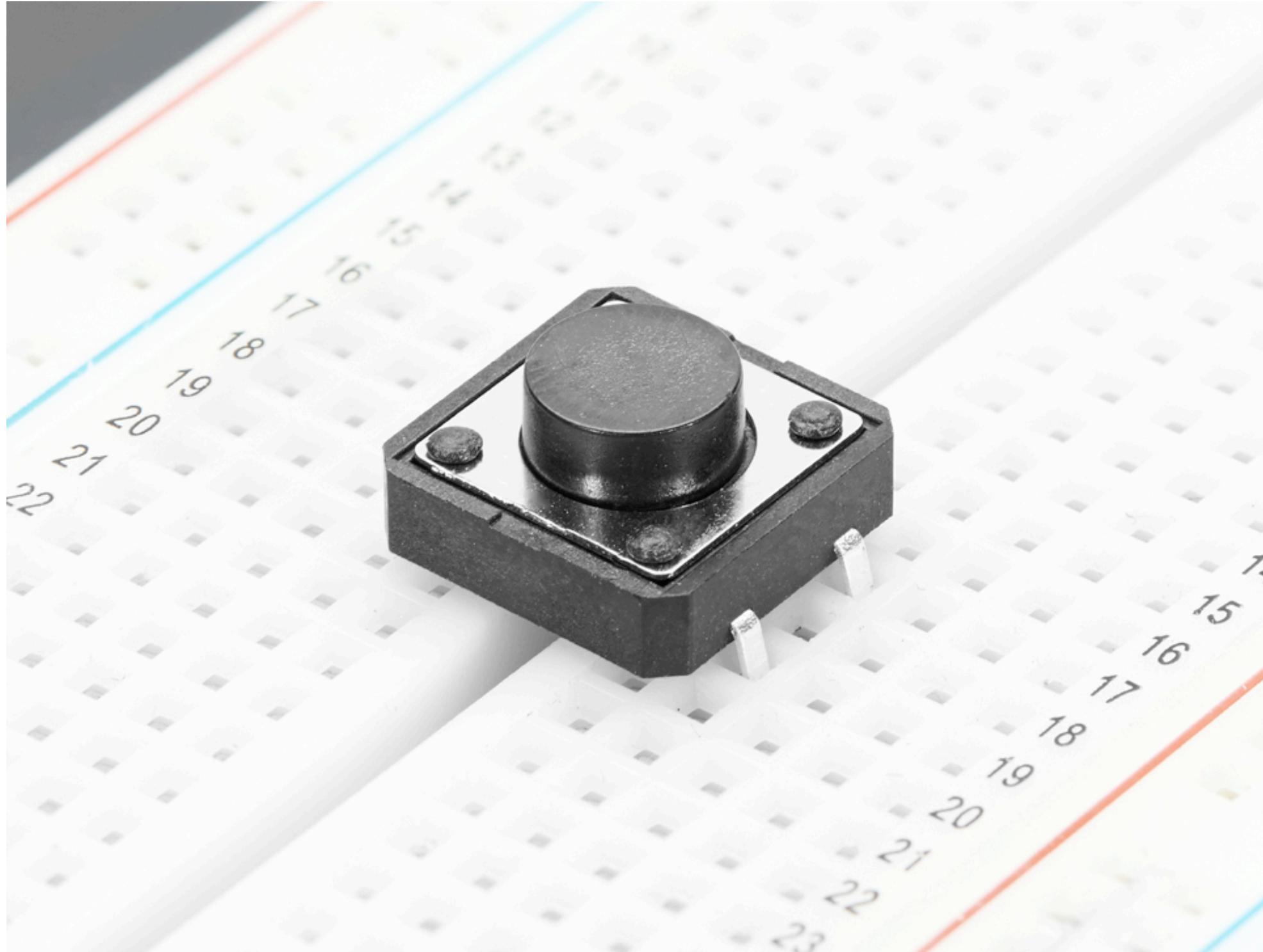
Metallischer Ball mit zwei Schaltern an beiden Enden

Gleiche Funktion wie ein Taster oder Schalter

Mikroschalter

Taster

Verschiedene Auslösehöhen



Arduino Bauteile

<https://funduino.de/nr-5-taster-am-arduino>

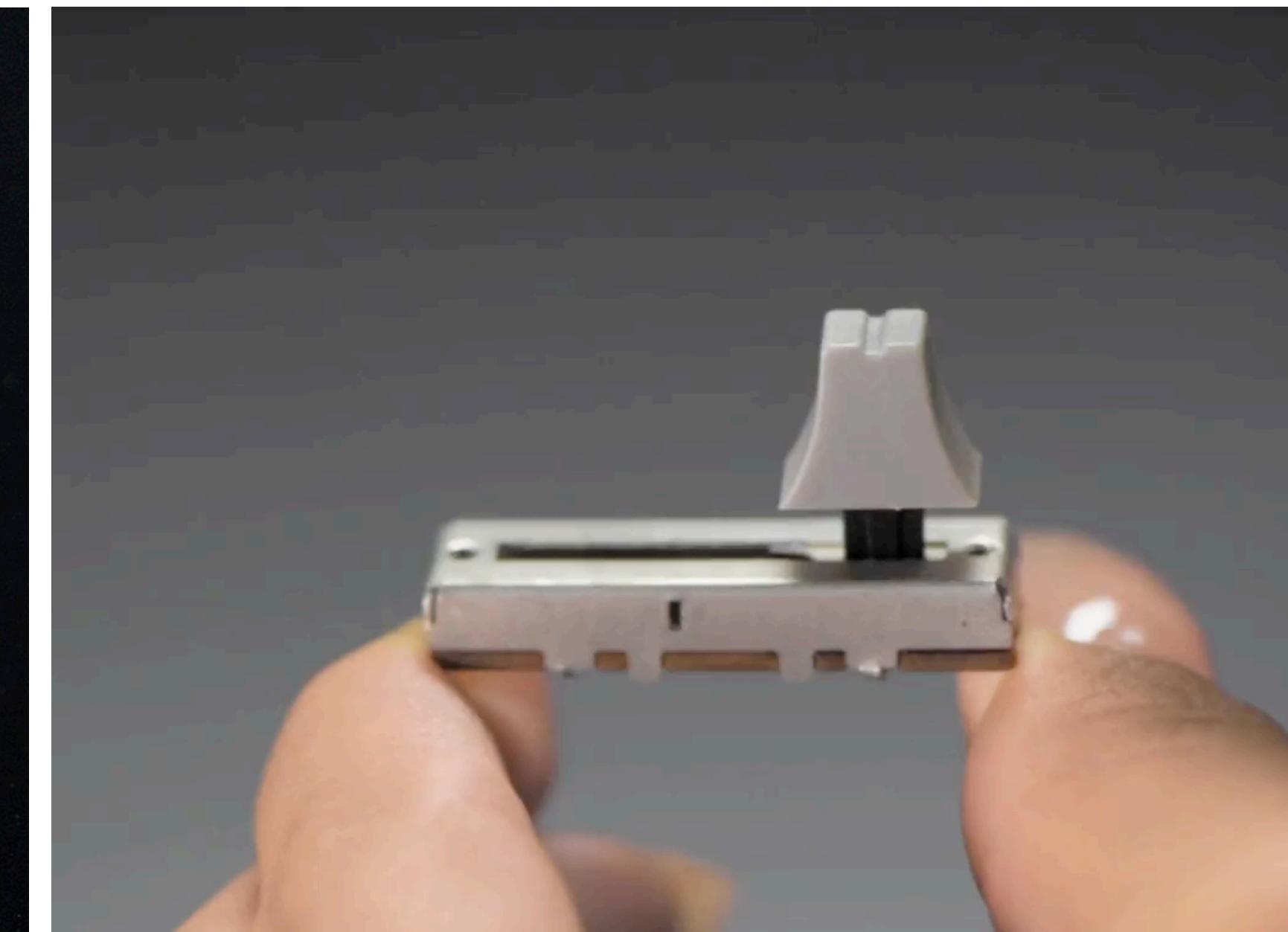
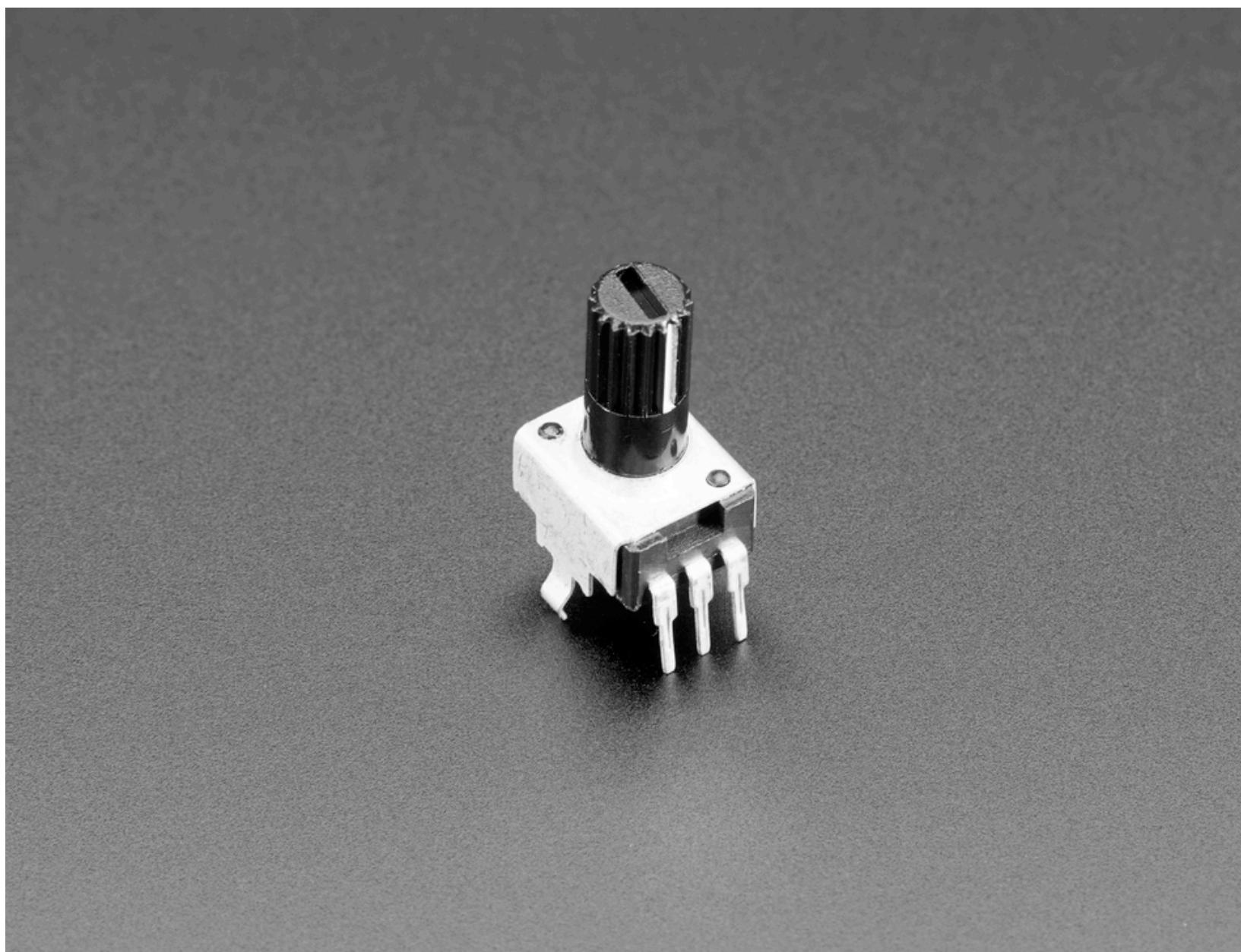
Potentiometer



Einstellbarer Widerstand

Verschiedene Widerstandsbereiche möglich

Verschiedene Bauweisen



Relais

Schalter für größere Ströme

Ermöglicht das Schalten leistungsstarker Bauteile

z.B. große Motoren, Netzspannung (230V)



Arduino Bauteile

<https://funduino.de/nr-15-relaisshield>

Thermistor

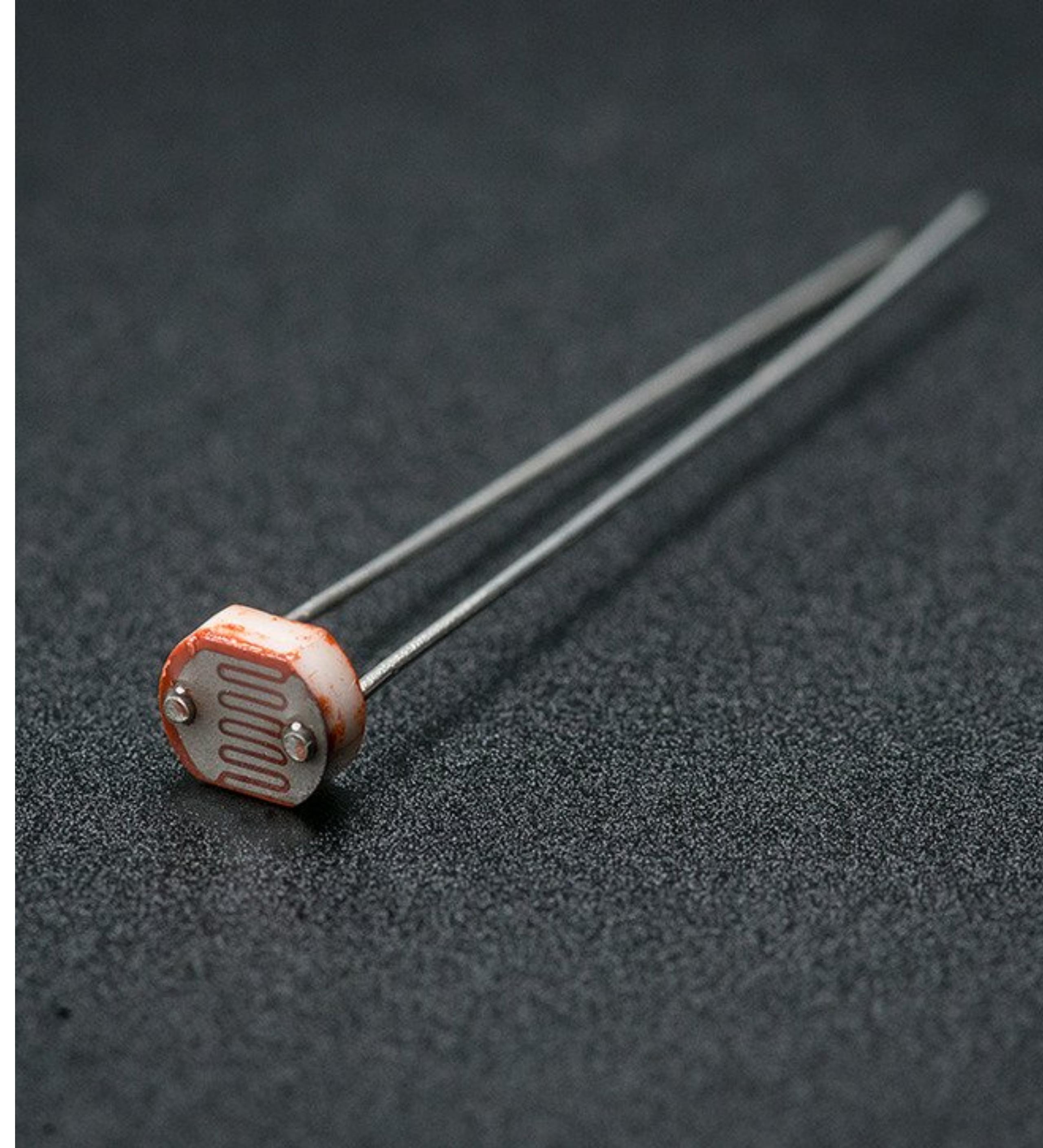


Digitales Thermometer

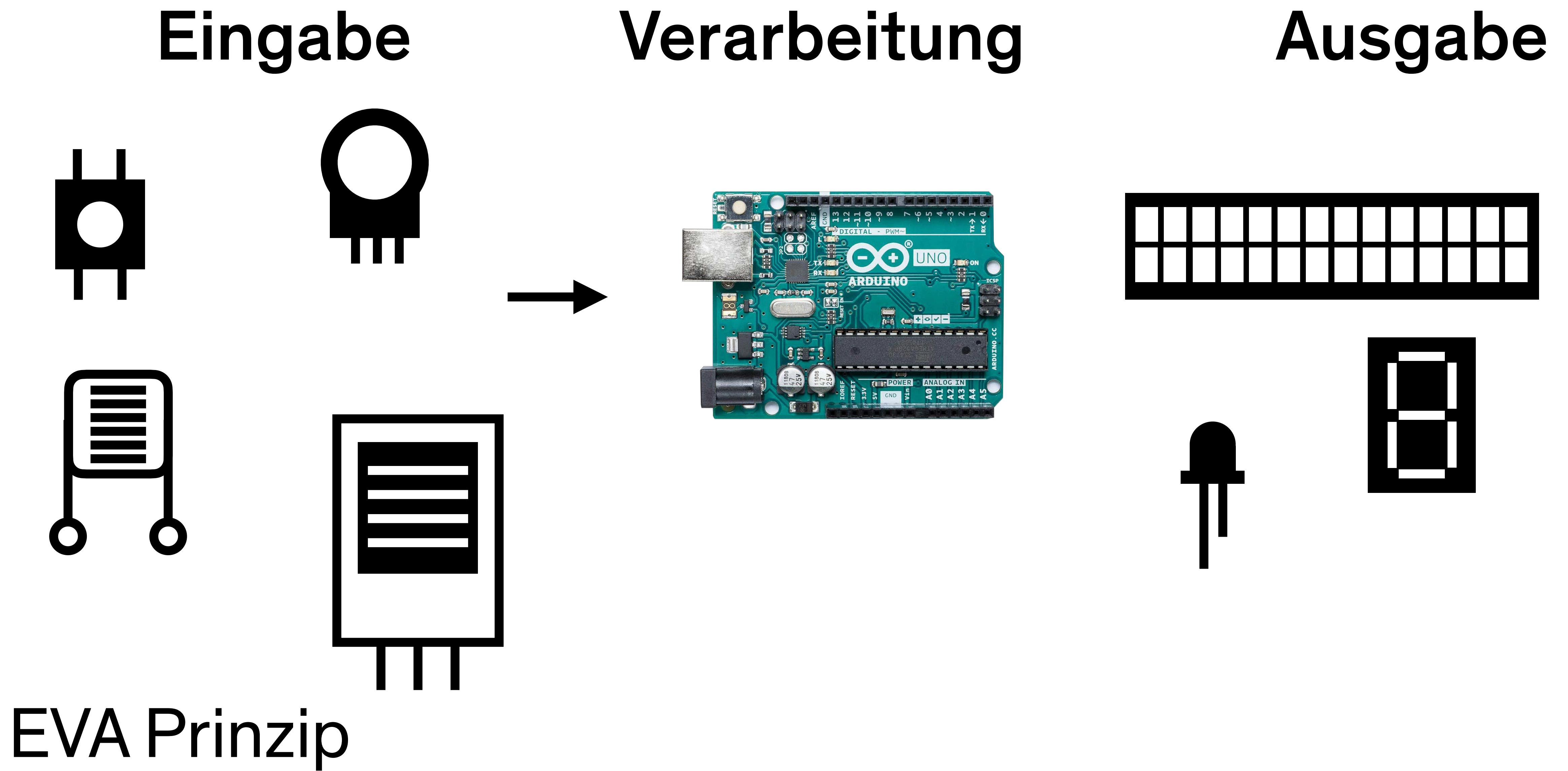
Änderung des Widerstands bei Temperaturänderung

Photowiderstand

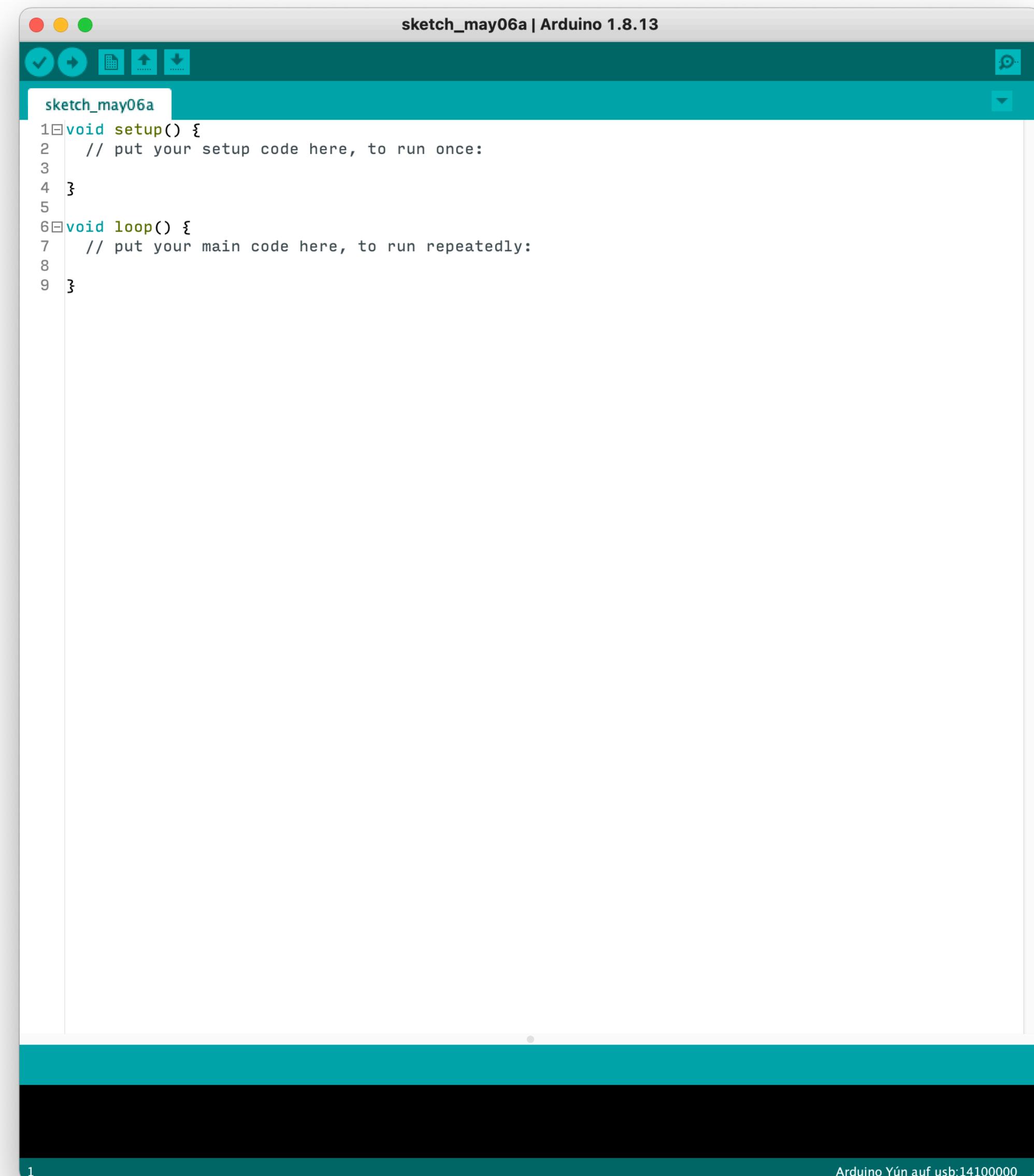
Arduino Bauteile



EVA Prinzip



Programmier Grundlagen



The screenshot shows the Arduino IDE interface with a single tab titled "sketch_may06a | Arduino 1.8.13". The code area contains the following:

```
sketch_may06a
1 void setup() {
2     // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7     // put your main code here, to run repeatedly:
8 }
```

The status bar at the bottom indicates "Arduino Yun auf usb:14100000".

Programmier Grundlagen

Struktur – setup()

Wird beim Start des Programs einmalig aufgerufen

Initialisieren von Variablen

Initialisieren von Pins

Einmalige Aufgaben:

Objekte erschaffen, Objekte initialisieren, Serielle Verbindung aufbauen uvm.



The screenshot shows the Arduino IDE interface with the title bar "sketch_may06a | Arduino 1.8.13". The code editor contains the following code:

```
1 void setup() {
2     // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7     // put your main code here, to run repeatedly:
8
9 }
```

Struktur – loop()

Wird immer wieder ausgeführt

Enthält den Programmcode

Durch das **loopen** des Codes können Veränderungen von Variablen erkannt werden



The screenshot shows the Arduino IDE interface with the title bar "sketch_may06a | Arduino 1.8.13". The code editor window displays the following sketch structure:

```
1 void setup() {  
2     // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7     // put your main code here, to run repeatedly:  
8 }  
9 }
```

The code editor has a dark teal background with light teal syntax highlighting. The tabs bar at the top shows "sketch_may06a". The toolbar above the code editor includes icons for file operations (checkmark, arrow, file, upload, download) and a pin icon.

Variablen – Konstanten

HIGH

- An einem Output mit HIGH liegen 5V an
- Soll ein Output als HIGH gelesen werden, müssen min. 3V anliegen

LOW

- An einem Output mit LOW liegen 0V an
- Soll ein Output als LOW gelesen werden, dürfen max. 1.5V anliegen

INPUT

- Definiert einen Pin als Eingang
- Der Pin wird in einen hochohmigen Zustand versetzt

OUTPUT

- Definiert einen Pin als Ausgang
- Der Pin wird in einen niederohmigen Zustand versetzt

INPUT_PULLUP

- Definiert einen Eingang mit den internen Pull-Up-Widerständen

LED_BUILTIN

- Bezeichnet den Pin, an dem die interne Board LED angeschlossen ist
- Auf den meisten Arduinos ist das Pin 13

Variablen – Konstanten

true

- Definiert als 1 oder wahr
- Kann für Vergleiche (boolesche Algebra) verwendet werden
- Alle Zahlen außer 0 sind true

false

- Definiert als 0 oder unwahr / falsch
- Kann für Vergleiche verwendet werden

Variablen – Datentypen, Primitive

bool	→ true / false
char	→ 65, 66, ...
byte	→ 0, 1, ... 244, 255
short.	→ -32,768 bis 32,767
integer	→ -32,768, ... 1, 2, 3, 4 ... 32,767
long	→ -2,147,483,648 ... 0, 1, 2, 3, 4 ... 2,147,483,647
float	→ 0.01, 0.123, 0.4567 ...
double	→ 0.1, 0.2, 3.4

Variablen – Datentypen, Weitere

Array	→ Liste von primitive Datentypen oder Objekten
Objekt	→ Sammlung von Funktionen, Datentypen, Code
String	→ "Hallo", eine Liste / Kette von chars
unsigned char	→ Vorzeichenlos, also nur positive Zahlen
unsigned int	→ Vorzeichenlos, 0 bis 65,53
unsigned long	→ 0 und 4,294,967,295
void	→ Funktionsdeklaration, Rückgabewert

Funktionen – Digital I/O

pinMode()

- Konfiguriert einen Pin als Input oder Output
- `pinMode(pin, mode)`

digitalRead()

- Liest Werte von einem digitalen Pin ein
- Mögliche Werte: HIGH oder LOW
- `digitalRead(pin)`

digitalWrite()

- Beschreibt einen digitalen Pin
- Mögliche Werte: HIGH oder LOW
- `digitalWrite(pin, value)`

Funktionen – Analog I/O

analogRead()

- Liest Werte von einem analogen Pin ein
- Spannung: 0V – 5V
- Wird umgewandelt in: 0 – 1023
- `analogRead(pin)`

analogWrite()

- Beschreibt einen PWM-fähigen Pin
- Direkt Dimmen einer LED möglich
- `analogWrite(pin, value)`

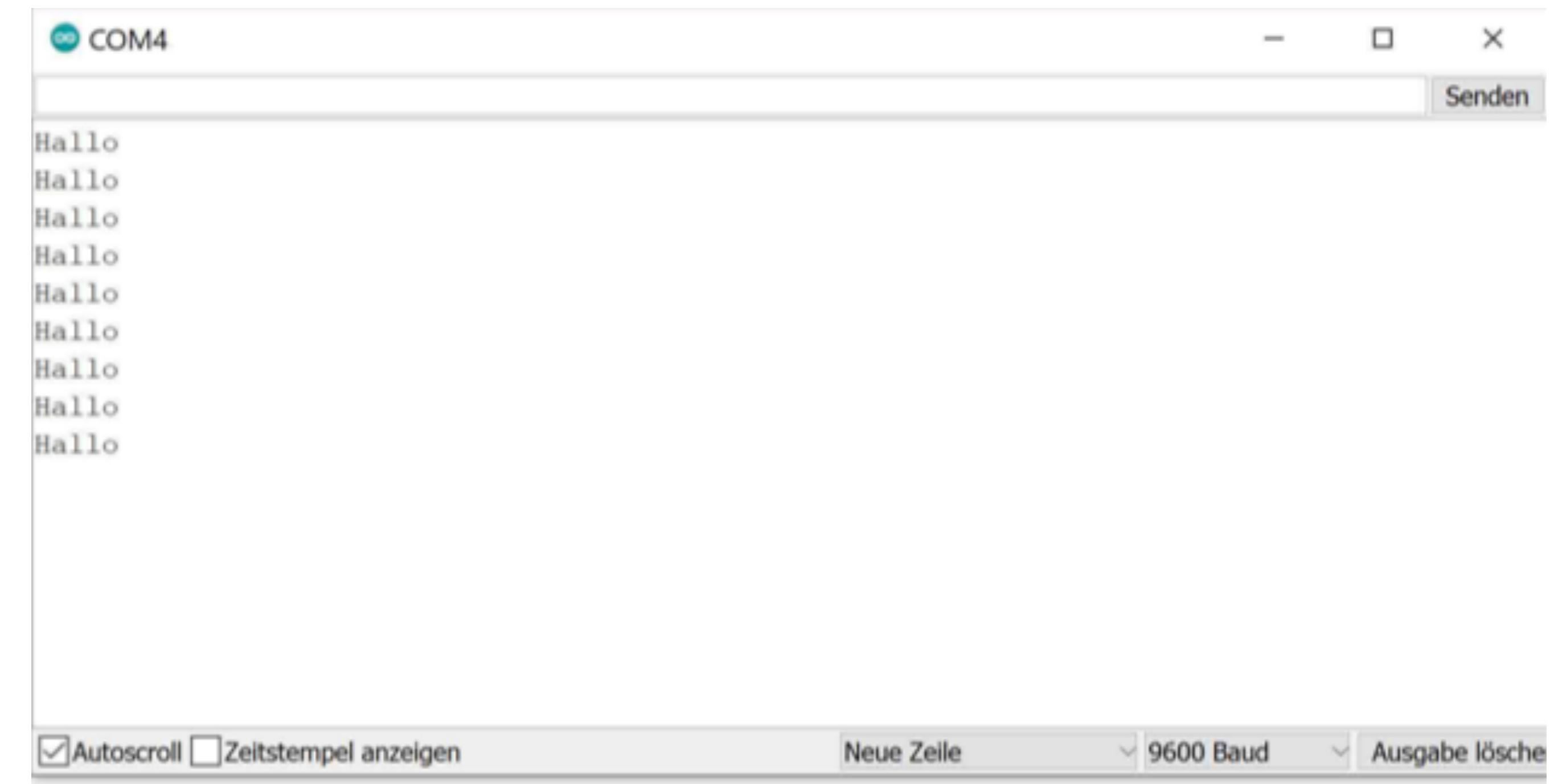
Funktionen – Serial Monitor

Erlaubt Serielle Kommunikation zwischen Arduinos oder mit dem Rechner

Kann zur Anzeige von Zuständen des Arduino über den seriellen Monitor verwendet werden

`Serial.begin(9600)`

`Serial.println("Hallo")`



Funktionen – Time

delay()

- Pausiert das Programm (das *ganze* Board)
- Wert in Millisekunden
- Während der Pause kann / wird nichts ausgeführt
- Schlecht für Synchronisierungsprozesse
- `delay(ms)`

millis()

- Gibt die Anzahl der Millisekunden seit dem Programmstart zurück
- `long timestamp = millis()`

delayMicroseconds()

- Pausiert das Programm (das *ganze* Board)
- Wert in Mikrosekunden
- Während der Pause kann / wird nichts ausgeführt
- Schlecht für Synchronisierungsprozesse
- `delayMicroseconds(us)`

micros()

- Gibt die Anzahl der Mikrosekunden seit dem Programmstart zurück
- `long timestamp = micros()`

Funktionen – Map

Bildet eine Zahl von einem Bereich in einen anderen Bereich ab

Ein Wert von fromLow wird auf toLow abgebildet

Ein Wert von fromHigh wird auf toHigh abgebildet

Dies sind die Grenzen. Werte dazwischen werden auf die entsprechenden Zwischenwerte im anderen Bereich abgebildet.

`map(value, fromLow, fromHigh, toLow, toHigh)`

`value:` Unsere Eingabe

`fromLow:` Untere Grenze des aktuellen Wertebereichs unserer Eingabe

`fromHigh:` Obere Grenze des aktuellen Wertebereichs unserer Eingabe

`toLow:` Untere Grenze des Zielbereichs des Werts

`toHigh:` Obere Grenze des Zielbereichs des Werts

Programmier Grundlagen

Funktionen – Map

`map(value, 0, 50, 0, 1000)`

value = Wert liegt zwischen 0 – 50

from-Wertebereich: 0 – 50

to-Wertebereich: 0 – 1000

`map(25, 0, 50, 0, 1000)`

$25 \rightarrow 500$

`map()` kann auch mit negativen Zahlen umgehen,
d.h. man kann selbst negative Wertebereich erfassen

Funktionen – Map

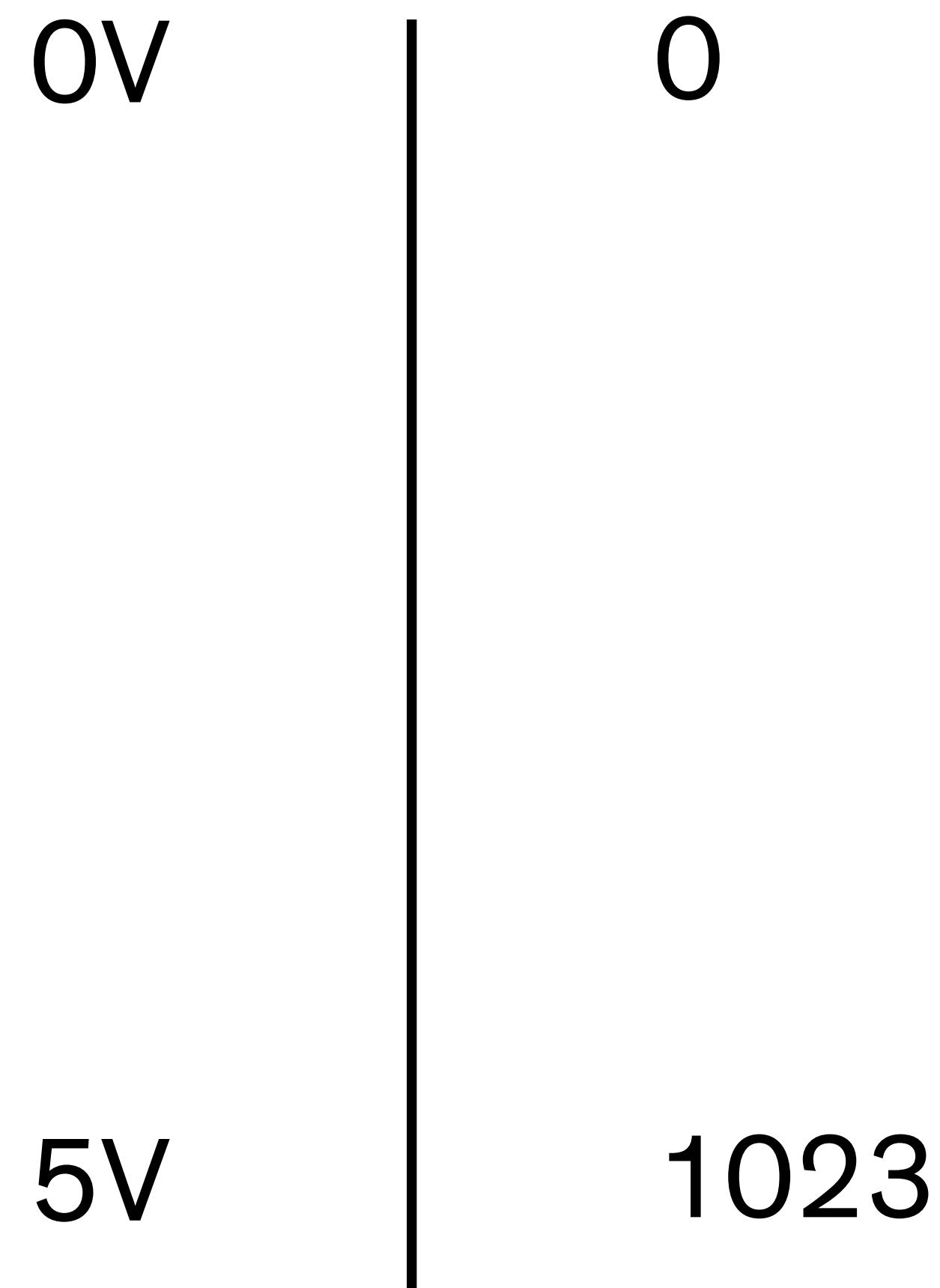


Programmier Grundlagen

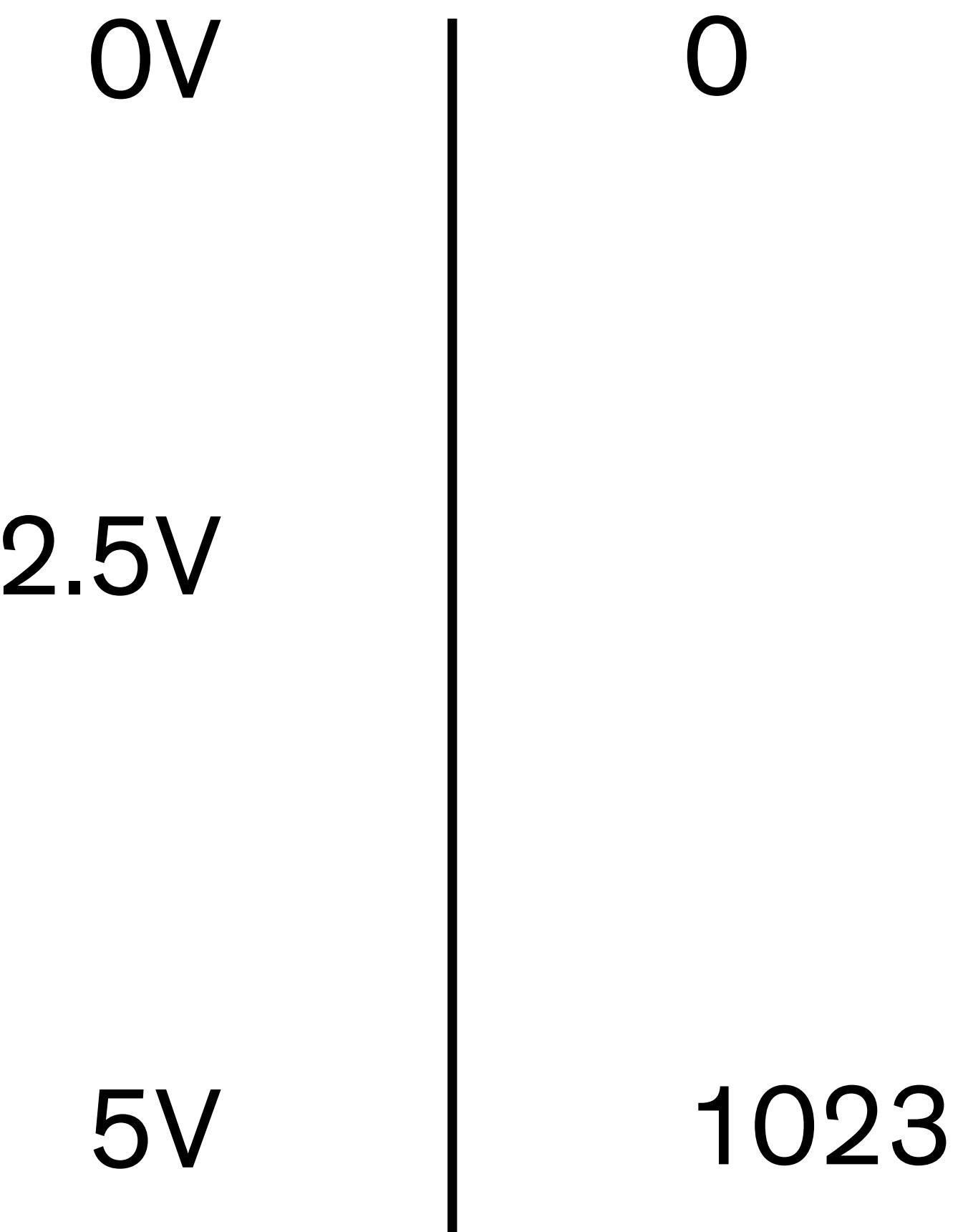
Funktionen – Map



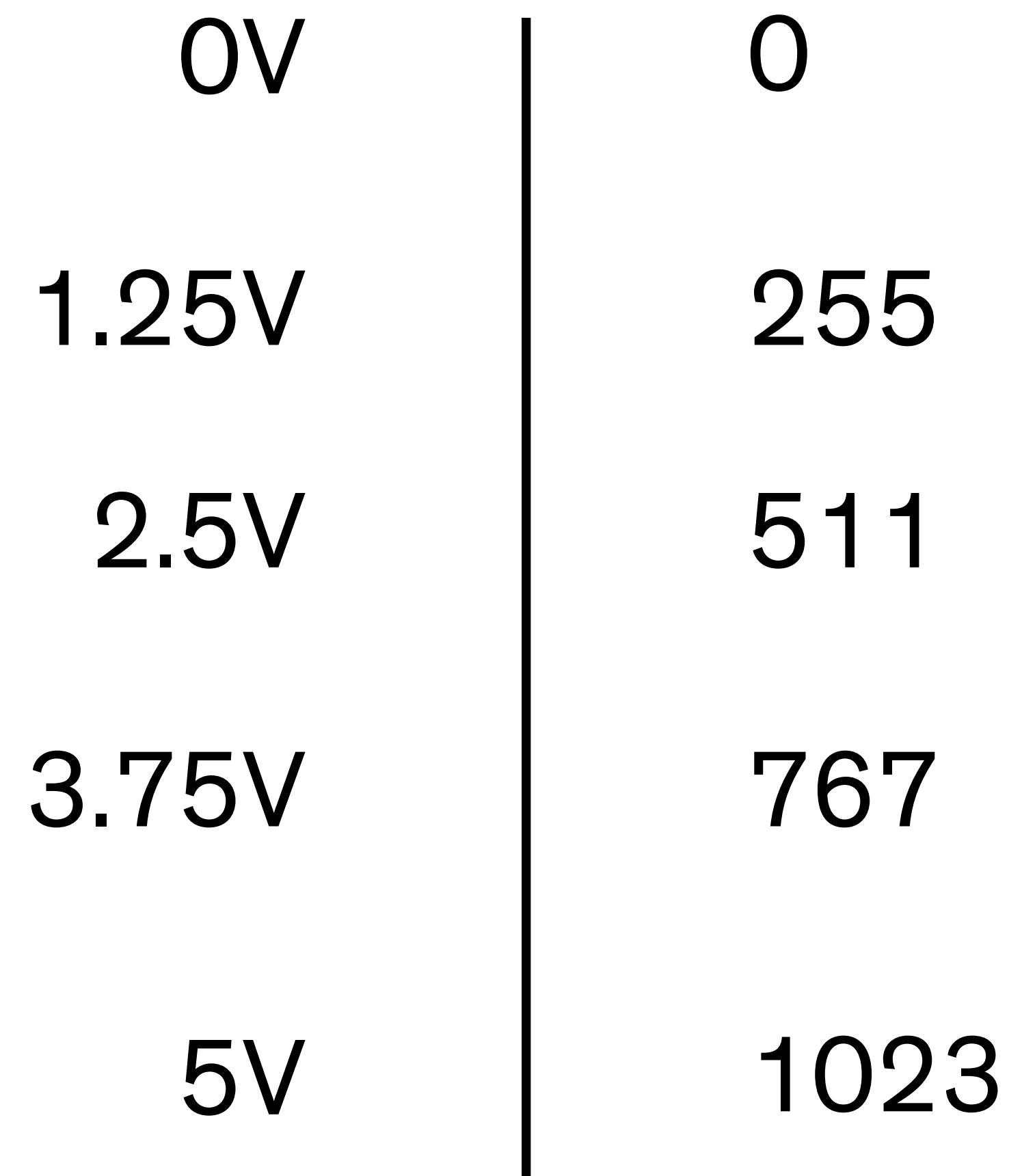
Funktionen – Map



Funktionen – Map



Funktionen – Map



Kommentare

Erleichtern die Lesbarkeit von Code

Stellen Informationen für andere bereit, z.B. in Programmier-Bibliotheken

Strukturieren den Code

Einzeilige Kommentare werden mit // beginnen

Mehrzeillige Kommentare beginnen mit /* und enden mit */

```
1 void setup() {  
2     // put your setup code here, to run once:  
3  
4 }  
5  
6 void loop() {  
7     // cooler, einzeiliger Kommentar  
8  
9     /*  
10      * Mehrzeiliger Kommentare  
11      * wo ich viel darüber erzählen kann  
12      * wie der nachfolgende Code funktioniert  
13      */  
14 }
```

Programmier Grundlagen

<https://www.arduino.cc/reference/en>

<https://funduino.de/>

Code Referenz + Link für Übungen

Entfällt – Christi Himmelfahrt / Brückentag

Vorlesung am 14.05.2021

Aufgabe 2

Baut einen LED-Dimmer mit einem Photowiderstand und eurem Arduino

Stellt Überlegungen an:

- Was passiert wenn ich 2 Photowiderstände einbaue und 2 LEDs parallel steuern will?
- Wie würde man vorgehen um "gleichzeitig" beide Inputs abzugreifen bzw. was verhindert momentan diesen Moment der Synchronisierung?

↓ Baut einen LED-Dimmer mit einem Photowiderstand und eurem Arduino

- Arduino
- Breadboard
- LED (beliebige Farbe)
- Geeignete Widerstände (URL!)
- Photowiderstand
- Spannungsteiler
- PWM
- Jumperkabel
- Serielle Ausgabe der Werte (Serial Monitor oder Plotter)

Aufgabe 2 – Bis 14.05.2021

- 1. Plant eure Aufbauten in TinkerCad (nutzt ggf. die "Simulation" Funktion)**
- 2. Zeichnet die Schaltpläne sauber mit der Hand**
- 3. Dokumentiert:**
 - Fotos: Echter Aufbau + Schaltplan**
 - Screenshots: TinkerCad**
 - Kurze Beschreibung (Bauteile, Vorgehensweise)**
 - Quellcode**
 - Name, Matrikelnummer**
 - Aufgabe, Datum**
- 4. Abgabe auf Incom als PDF im "Abgabe" Ordner**

Nutzt die beiden Vorlesungen und das Tutorium mit Christoph Schubert (s. Incom)

Aufgabe 2 – Bis 14.05.2021

Fragen?