PH Schwäbisch Gmünd
University of Education

Auszug aus dem Skript zur Veranstaltung
(SS 2019)

Einführung in Arduino

Für die Make Ostwürttemberg gekürzt.

Stand: 20. September 2019

Inhaltsverzeichnis

Aus Gründen der besseren Lesbarkeit wird auf die gleichzeitige Verwendung männlicher und weiblicher Sprachformen immer wieder verzichtet. Sämtliche Personenbezeichnungen gelten gleichwohl für beiderlei Geschlecht, sofern nicht ausdrücklich eine Einschränkung durch explizite Nennung des Geschlechts vorliegt.

Dieses Skript ist unter Verwendung von Fritzing, LaTeX und Citavi entstanden.

Tabellenverzeichnis

Abbildungsverzeichnis

Vorwort

Dieses Skript ist im Zuge der Zulassungsarbeit von David Weiler entstanden. Das Thema der Arbeit lautet: „Konzeption eines Seminars zum Thema: Einführung in Arduino für Studierende des Physik-Lehramtes“.

Es werden unterschiedliche Sensoren und Aktoren behandelt sowie die physikalischen Grundlagen, die diesen zu Grunde liegen. Diese kommen aus den Bereichen der Akustik, Optik, Elektrizitätslehre und Wärmelehre.
Schritt für Schritt werden Grundlagen des Programmierens vermittelt und es soll durch experimentelles Lernen ein Wissenszuwachs gewonnen werden. Dabei werden auch Bereiche der Statistik genutzt.

Zudem sollen Möglichkeiten aufgezeigt werden, wie die Anwendungen in der Schule Einzug finden können bzw. zu welchen Themen im Bildungsplan (BP) in Baden Württemberg sie passen.

Bei der Erstellung der Platine für die Lärmampel hat mich Florian Wengert von der EULE Wissenswerkstatt in Schwäbisch Gmünd unterstützt.
Beim Kapitel Internet of Things hat mir Kevin Kutzner von der Hochschule für Technik in Stuttgart Auszüge seines Skriptes zur Verfügung gestellt.

Bildungsplanbezug:
Grundsätzlich ist der Arduino für das Profilfach IMP (Informatik, Mathematik und Physik) wie geschaffen. Hier wird praktisch mit dem gearbeitet, was die SuS im Teilbereich Physik lernen sollen. Ein Auszug hieraus:

„Der Teilbereich Physik befasst sich einerseits mit grundlegenden Aspekten der digitalen Daten-verarbeitung und -übertragung (zum Beispiel Halbleiterphysik, Sensoren, Lichtleiter) und wendet andererseits informatische und mathematische Kompetenzen bei numerischen Berechnungen von Abhängigkeiten und Abläufen an.“

Es finden sich noch weitere Anwendungsmöglichkeiten aus dem Sachkundeunterricht, des BNT-Unterrichts (Biologie, Naturwissenschaft und Technik) und des Physikunterrichts. Alles Fächer, in denen ein Physiklehrer gerne an Schulen eingesetzt wird.

Teil I.
Arduino

1. Der Arduino

Arduino ist im Grunde genommen eine Plattform für Mikrocontroller und deren Programmierung. Es gibt zwar auch die Marke Arduino, aber grundsätzlich sind die Projekte Open-Source. Daher gibt es Mikrocontroller-Boards, die auf Arduino Boards zurückgreifen von vielen unterschiedlichen Herstellern. Dies ermöglicht es auch kostengünstig in der Schule mit dem Thema einzusteigen.
Da es viele verschiedene Mikrocontrollertypen aus der Arduino Familie gibt, muss gesagt werden, dass wir im Zuge des Seminars auf das Modell Arduino Uno R3 zurückgreifen und für eine Anwendung auf die kleinere Version, den Arduino Nano, zurückgreifen.

1.1. Die Hardware

1.1.1. Der Uno R3 Controller

Der Uno R3 ist ein relativ großer Mikrocontroller. Er bietet die Möglichkeit, einige Sensoren (das sind Bauteile die Signale an den Arduino geben) und Aktoren (also Bauteile die etwas ausführen) anzuschließen. Er hat dafür 14 digitale Pins und 6 analoge Pins. Jeder Pin kann etwa 40mA ausgeben.

Ein Mikrocontroller besteht grundsätzlich aus drei Hauptbereichen:

- Der Zentralen Recheneinheit (CPU von Central Processing Unit),
- den Speichern (Datenspeicher und Programmspeicher)
- und den Ein- und Ausgabeports (im Folgenden als Pins bezeichnet).

Die CPU führt die Rechenoperationen aus, dabei agiert sie mit den Ein- und Ausgängen, dem Speicher und kann auch auf den verbauten Oszillator, einen Zeitgeber und Interrupt-Steuerungen zurückgreifen. (vgl. Bartmann (2015), S. 5)

Der Datenspeicher (SRAM) speichert Daten, die während des Programms anfallen. Wenn man den Arduino ausschaltet sind diese Daten gelöscht. Er ist vergleichbar mit dem Arbeitsspeicher am Computer. Er hat eine Größe von 2KB.
Der Programmspeicher hingegen beinhaltet die BIOS des Mikrocontrollers, das vom Hersteller aufgespielt wurde. Zudem wird hier das Programm gespeichert, das der Mikrocontroller ausführen soll. Es ist vergleichbar mit der Festplatte am Computer. Er hat eine Größe von 32KB. (vgl. Bartmann (2015), S. 6)

An den Ein- und Ausgabepins können Sensoren wie z.B. Ultraschallsensor, Temperatursensor etc. angeschlossen werden. Zudem kann der Arduino Bauteile mit 5V oder 3.3V Spannung versorgen. Nachfolgend eine Übersicht über den Arduino Uno.

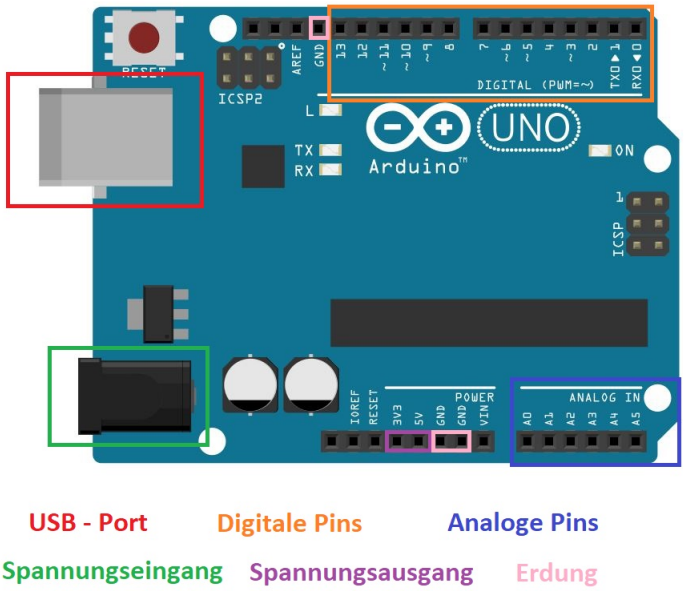

Abb. 1.1: Überblick über den Aufbau eines Arduino Uno R3 Controllers

Durch seine Steckverbindungen ist eine einfache Verbindung mit anderen Bauteilen möglich. Für umfangreichere Schaltungen benutzt man aber ein Breadboard oder eine feste Platine, wenn das Projekt auf Dauer sein soll.

1.1.2. Das Breadboard

Das Breadboard ist ein Steckbrett mit einer Lochrastergröße von 2.54mm, das für viele Systeme geeignet ist. Der Arduino und viele Sensoren für ihn besitzen dieselbe Rastergröße. Auch andere Systeme wie der Raspberry Pi nutzen diese Rastergröße, was eine Kompatibilität zu anderen Systemen ermöglicht.
Es hat zwei Seiten, die voneinander getrennt sind. An jeder Seite befinden sich zwei Reihen am Rand, die jede für sich durchgängig verbunden sind. Verbindet man also einen Pin dort mit der Erdung (GND) des Arduino, ist die gesamte Leiste geerdet. Das Feld dazwischen hat 64 Spalten, für die dasselbe gilt wie für die äußeren Reihen. Die Verbindungen sind teilweise im folgenden Bild angedeutet.

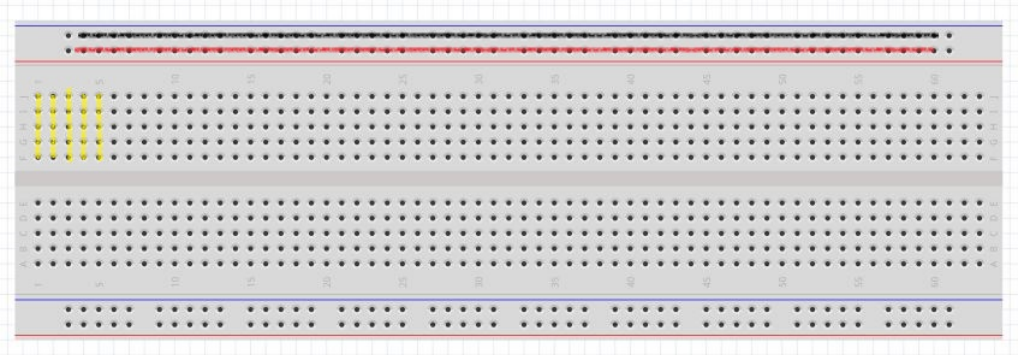

Abb. 1.2: Die Verbindungen auf einem Breadboard

1.2. Die Software - Open Roberta

Was ist Roberta?
Die „Roberta - Lernen mit Robotern“ Initiative widmet sich seit 2002 der Bildung von Lehrerinnen und Lehrern für die digitale Welt. Sie ist ein Bildungsprogramm des Fraunhofer-Instituts für Intelligente Analyse- und Informationssysteme (IAIS). Sie bieten unter anderem Unterrichtsmaterial, Fortbildungen und das Programmierbaukastensystem **Open Roberta Lab** an, das wir auch nutzen werden.

Materialien findet man als Lehrkraft auf der Homepage unter: `https://www.roberta-home.de/lehrkraefte/`

Die Software
Die benötigte Software die wir benötigen gliedert sich in zwei Teile: Die Verbindungssoftware, um unseren Arduino an das Roberta Netzwerk anzuschließen und das Lab selber, das im Browser zugänglich ist.

Als erstes benötigen wir die Software zum Verbinden des Arduino. Diese Software heißt *Open Roberta USB* und kann unter `https://github.com/OpenRoberta/robertalab-usbprogram/releases` für das jeweilige Betriebssystem heruntergeladen werden.

Nun aber zum Open Roberta Lab. Ihr erreicht es unter `https://lab.open-roberta.org/#`. Dort könnt ihr euch auch anmelden wenn ihr eure Programme speichern wollt.

Die Seite wird euch zuerst fragen, welche Art von Roboter ihr benutzen wollt. Wir nutzen *NEPO for Arduino*.
NEPO bezeichnet dabei die visuelle Programmiersprache, die wie anfangs erwähnt Scratch ähnelt, mit der wir unseren Arduino programmieren wollen.

An der oberen Leiste befinden sich die Möglichkeiten zum Speichern (Notizblock), den Arduino verbinden (Unendlichzeichen), Hilfe (Glühbirne), zum Anmelden (Mensch) und weitere Optionen.
Darunter kann man zwischen **Programm** und **Roboterkonfigurationen** wählen. Im Programm fügen wir per Drag-and-Drop die Befehle die wir benötigen hinzu. Diese sind links in unterschiedliche Kategorien eingeteilt. Um auf alle Möglichkeiten zugreifen zu können, müssen wir das Sternchen mit der 2 daneben auswählen. Für den Einsatz im Unterricht reichen zunächst die Grundfunktionen, die sich unter 1 finden.

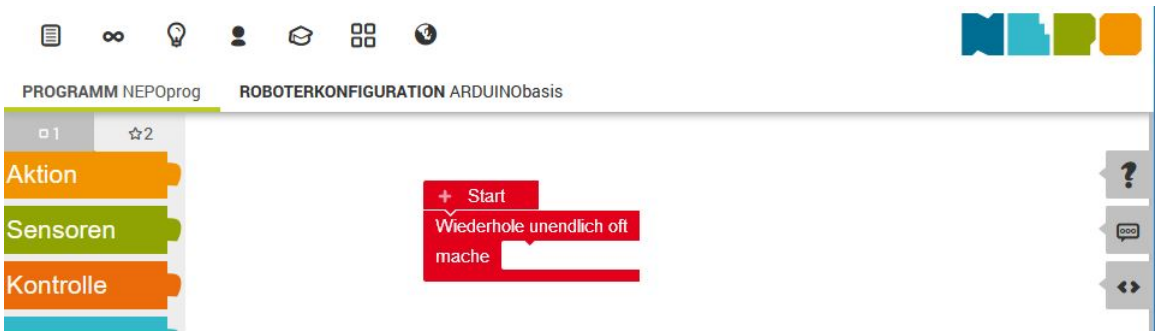

Abb. 1.3: Die Hauptseite des Open Roberta Lab

In den Roboterkonfigurationen können wir hinterlegen welcher Sensor an welchen Pin angeschlossen ist.

An der rechten Leiste haben wir unter anderem die Möglichkeit, auf den Programmcode zu schauen, der hinter der visuellen Oberfläche angelegt wird.

Am unteren Ende der Seite finden wir vier Symbole wovon zwei zu Beginn ausgegraut sind.

Das Play-Zeichen ist dazu da, ein Programm an den Arduino zu übertragen und dort zu starten. Der Pfeil nach oben in die Wolke ist zum speichern des Programms. Dazu muss man aber angemeldet sein und den Namen des Programms oben links in den Einstellungen schon gespeichert haben.

Abb. 1.4: Schaltfläche am Ende der Seite

Die Lupe ermöglicht es uns zu zoomen und der Mülleimer Blöcke zu entfernen.

2. LED-Ansteuerung

2.1. Zur Theorie von LEDs

Bevor wir nun zur Tat schreiten, soll hier zuerst erklärt werden, was LEDs sind und wie diese funktionieren.
Die Abkürzung LED steht für **Leuchtdioden** und kommt aus dem Englischen von *light emitting diode*. Diese sind pn-Halbleiter an denen eine Spannung angelegt wird. An der p-Seite entsteht so ein Elektronenüberschuss und an der n-Seite entstehen nun Löcher. Treffen nun Elektronen auf Löcher wird dabei Licht emittiert. Die p-Seite (also die Anode) wird an den ⊕-Pol einer Schaltung angeschlossen und die n-Seite (also die Kathode) an den ⊖-Pol. Schließt man die Diode andersherum an, sperrt diese und leuchtet nicht. (vgl. Tipler, Mosca und Basler (2012) und Lehn et al. (2019))

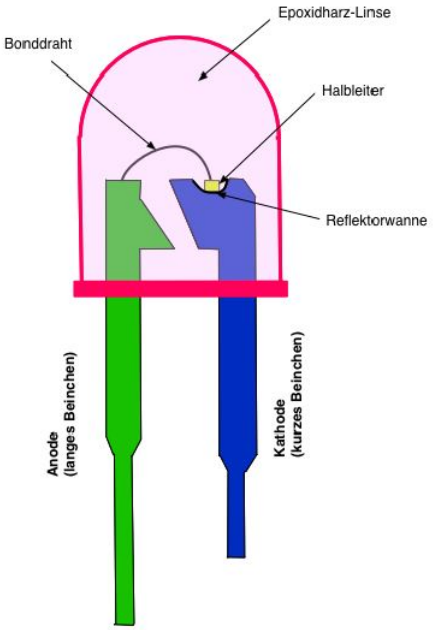

Abb. 2.1: Aufbau einer Diode aus Lehn et al. (2019), S. 98

p- bzw. n-Halbleiter sind mit Fremdatomen dotierte Siliciumkristallgitter. Reine Silicumkristalle sind schlechte elektrische Leiter, da sie mit ihren Valenzelektronen vier vollständige kovalente Bindungen eingehen.
Arsen hat z.B. ein zusätzliches Valenzelektron, das angeregt werden kann und somit zur elektrischen Leitung beitragen kann. In diesem Falle spricht man von einem **n-Halbleiter**.
Wird ein Siliciumkristall hingegen mit beispielsweise Gallium dotiert, entstehen Elektronenlöcher in der Struktur, da Gallium ein Valenzelektron weniger als Silicium hat. So entsteht ein **p-Halbleiter**.

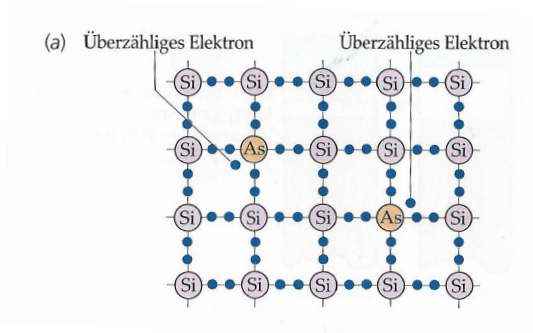

Abb. 2.2: n-Halbleiter

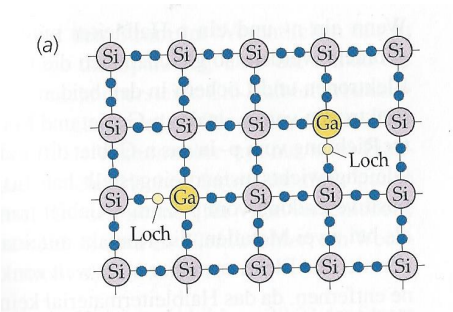

Abb. 2.3: p-Halbleiter

2.2. LEDs beim Einsatz mit Arduino

Die meisten LEDs sind für den Betrieb bei einer Spannung von 5V nicht gedacht. Daher sollte man vor jede LED einen **Vorwiderstand** anschließen. Je nach LED-Farbe variiert die abfallende Spannung zwischen 1.2V und 3V. Bei einer Stromstärke von etwa 10mA eignet sich der häufig mitgelieferte **220Ω Widerstand**. Achtet beim Anschließen einer LED immer darauf, dass die Anode (lange Seite) am ⊕-Pol und die Kathode (kurze Seite) am ⊖-Pol (GND) angeschlossen ist.

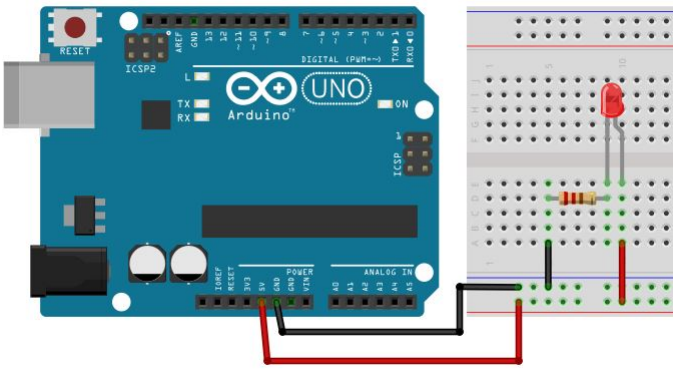

Abb. 2.4: Schaltplan für eine Dauerleuchte

Eine etwas andere Taschenlampe
Ein Button ist ein einfacher Schalter. Sein Grundzustand ist offen. Daher unterbricht er den Stromkreis, sofern er nicht gedrückt wird.
Man kann diesen also direkt in die Schaltung einbauen.

Bildungsplanbezug:
Solche ersten Versuche zum Stromkreis können schon in der Grundschule in Klasse 3/4 als Experimente eingesetzt werden. Dies würde folgenden Zielen im Bildungsplan für den Sachkundeunterricht entsprechen:
3.2.3.4 Energie - (6) beim Bau einer technischen Anlage (zum Beispiel einfacher Stromkreis) erfahren, dass man Elektrizität braucht, um zum Beispiel Räume zu beleuchten oder Geräte zu betreiben;

3.2.6 Experimente - (14) Experimente zum elektrischen Strom (elektrische Leitfähigkeit verschiedener Materialien) und dessen Wirkungen (Wärme, Licht, Bewegung).

Programmieren eines Blinklichts
Bisher haben wir den Arduino nur als Spannungsquelle wie eine Batterie verwendet. Dies ist aber nicht der große Mehrwert eines Mikrocontrollers. Daher soll nun ein Blinklicht programmiert werden, das zwei Sekunden leuchten soll und dann eine Sekunde aus sein soll. Hierfür werden wir zuerst mit Open Roberta die Aufgabe lösen und danach werden wir unsere ersten eigenen Versuche mit dem Programmieren machen.
Um die Spannungsversorgung der LED ein- bzw. auszuschalten, können wir nun

nicht mehr den 5V-Port des Arduino nutzen. Hierfür benötigen wir einen der digitalen Ausgabeports des Arduino. Wir werden dafür den Pin 13 nutzen. Verbindet daher nun das Kabel, das bisher zur ⊕-Leiste des Breadboards geführt hat mit dem Port 13.

Blinklicht mit Open Roberta

Die Plattform Open Roberta (`https://lab.open-roberta.org/#`) bietet uns die Möglichkeit, im Browser unsere Programme selber zu erstellen. Zuerst verbindet euren Arduino über das Programm **Open Roberta USB** mit der Seite. Wenn ihr euer Programm speichern wollt, dann meldet euch mit eurem Benutzerkonto an.
Wir können nun per Drag-and-Drop die Befehle, die wir benötigen, in das Programm ziehen. Eine mögliche Lösung ist rechts abgebildet.

Abb. 2.5: Beispielprogrammcode für eine blinkende LED

Was fällt auf, wenn man das Programm ausführt?

Zwei LEDs blinken im Wechsel

Nun möchten wir eine weitere LED anschließen und diese im Wechsel mit der bisherigen LED blinken lassen.

Zuerst werden wir dies mit Open Roberta lösen.
Hierzu muss zuerst der zusätzliche Pin für die neue Lampe definiert werden. Dafür gehen wir in den Reiter *Roboterkonfiguration* in dem wir wieder links die Möglichkeit haben, vorgefertigte Einstellung per Drag-and-Drop auszuwählen.
Im Reiter Aktion findet ihr die Voreinstellungen für LEDs. Wählt nun einen Pin aus, an dem ihr die zweite LED anschließen wollt.
Danach kehren wir zu unserem Programm zurück. Wir können bei den Blöcken *Schalte LED an X an* nun auswählen, welche LED an- bzw. ausgeschalten werden soll.

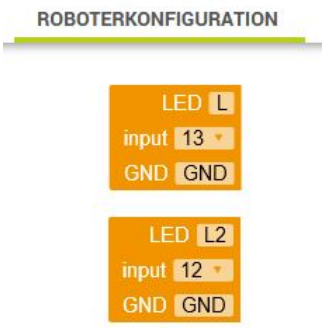

Abb. 2.6: Anschlüsse in Open Roberta definieren

Verändert euer Programm nun so ab, dass sobald die eine LED angeht, die andere ausgeht.

2.3. Theorie zur additiven Farbmischung

Wenn wir uns mit Farben beschäftigen, machen wir eigentlich einen Ausflug in die Biologie. Ohne Farbsinneszellen, die Zapfen, könnten wir keine Farben sehen. Wir könnten also die Wellenlänge von optischem Licht nicht als Farbempfindung auffassen.
Für die additive Farbmischung ist die *Farbmetrik* von großer Bedeutung. Dieser von Schrödinger geprägte Begriff bezeichnet die Maßbeziehungen von Farben untereinander. (vgl. Gobrecht, Eichler, Bergmann und Schaefer (1978), S. 642)

Überlagern sich zwei Farbeindrücke von unterschiedlich gefärbtem Licht wie z.B. von Rot und Grün, entsteht hierbei durch additive Farbmischung die Farbe Gelb.
Da das menschliche Auge drei unterschiedliche Farbrezeptoren hat (die Zapfen für **R**ot, **G**rün und **B**lau) kann man jede Farbe als Mischung aus diesen drei Farben (auch Farbvalenzen genannt) bilden. Um eine Farbe bestimmen zu können, braucht man also drei voneinander unabhängige Bestimmungsstücke (Farbe ist daher dreidimensional) und dies nutzt man auch bei RGB-LEDs.

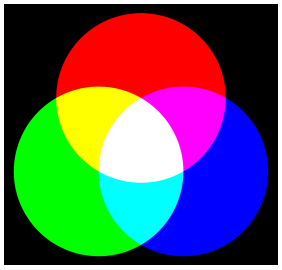

Abb. 2.7: Additive Farbmischung

2.4. Die RGB-LED

Eine RGB-LED funktioniert im Grunde wie eine normale LED, nur dass es hier drei einzelne LEDs gibt und eine gemeinsame Kathode. Somit kommt man auf vier Anschlüsse. Die einzelnen Pole der LED können einzeln angesteuert werden. Das heißt wir benötigen drei PINs am Arduino.
Die Farben der verbauten LEDs sind Blau, Grün und Rot aus deren additiver Farbmischung andere Farben entstehen.

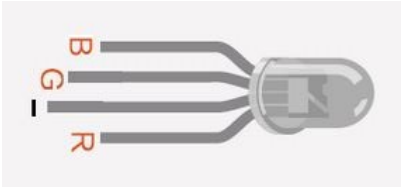

Abb. 2.8: Aufbau einer RGB-LED

Bisher haben wir digitale Ausgänge genutzt. Diese können mit dem Befehl *digitalWrite* nur den Zustand „Spannung an“ oder „Spannung aus“ haben. Das bedeutet wir könnten die Rote, Grüne und Blaue LED bei voller Leistung an oder ausschalten. Somit könnten wir nur 7 Farben (resultierend aus den Einzelfarben und deren Mischungen) darstellen.
Ein Arduino kann aber an seinen analogen Ausgängen durch **Pulsweitenmodulation (PWM)** unterschiedlich viel Energie liefern. Dies können wir uns zu Nutzen machen, um Farbtöne erzeugen zu lassen.

RGB-LED mit Open Roberta
Für Open Roberta gibt es schon fertig vorbereitete Möglichkeiten, um Farben zu generieren. Dazu muss man zuerst eine RGB-LED konfigurieren. Auch hier gibt es wieder voreingestellte Blöcke.

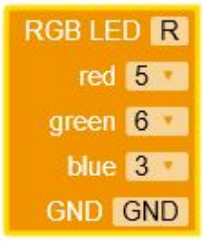

Abb. 2.9: Konfigurationen für RGB-LED in Open Roberta

Abb. 2.10: Programmierung für RGB-LED in Open Roberta

Danach kann man mit dem Farbblock im Programmcode die Farbe auswählen in der die LED leuchten soll. Dabei sind die nötigen PWM-Werte schon hinterlegt.

Die Pulsweitenmodulation
Bei der PWM ist die Frequenz immer dieselbe. Bei einem Arduino wird alle 2 Millisekunden die Spannung am Ausgang auf 5V gestellt und nach einer bestimmten Zeit abgeschaltet. So wird je nach Impulsdauer unterschiedlich viel Energie übertragen. Soll keine elektrische Energie übertragen werden, liegt der PWM Wert bei 0. Die maximale übertragene Energie liegt bei einem PWM Wert von 255 vor. Dies entspricht einem sogenannten *Tastgrad* (Impulsdauer geteilt durch Periodendauer) von 100%. Möchten wir also nur die Hälfte der Energie übertragen müssen wir also einen PWM Wert von 128 ausgeben.

Nun ist aber nicht jeder digitale PIN am Arduino dazu geeignet, eine PWM auszugeben. Man erkennt die PINs, die eine PWM durchführen können, an der *Tilde* (∼) vor der Nummer. (vgl. Bartmann (2015), S. 302ff)

3. Der Photowiderstand (LDR)

3.1. Theorie zu Photowiderständen

Eine Photowiderstand oder *Light Dependent Resistor* (LDR) nutzt den inneren Photoeffekt aus, um seinen Widerstandswert zu verändern. Bei Dunkelheit sind die Ladungsträger gebunden, sodass der Widerstandswert sehr hoch ist. Fällt allerdings Licht auf diese Halbleiter kommt es dazu, dass Ladungsträger des Kristalls in leitende Zustände gehoben werden. Das bedeutet, wird der LDR beleuchtet steigt seine Leitfähigkeit. (vgl. Meschede et al. (2010), S. 351)

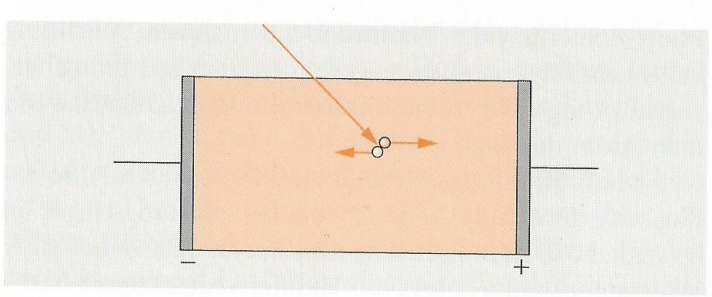

Abbildung 7.75 Photoleiter oder Halbleiter-Detektor. Licht oder ionisierende Strahlung erzeugt Ladungsträger im Kristall, meist paarweise (innerer Photoeffekt). Das angelegte Feld trennt die Ladungsträger: Photostrom

Abb. 3.1: Innerer Photoeffekt aus Meschede et al. (2010), S. 351

3.2. Photowiderstände mit Arduino

Nun wollen wir einen Photowiderstand mit dem Arduino dazu nutzen, um festzustellen wie hell es ist.
Dazu bauen wir uns eine Schaltung mit deren Hilfe wir den Zustand am LDR indirekt messen können.
Schaltet man zwei Widerstände in Reihe, addiert sich deren Widerstand. Es gilt also:

$$R_{Gesamt} = R_{LDR} + R_1 \tag{3.1}$$

Die anliegende Spannung bleibt in der Summe gleich. Aber die Spannungen werden in der Reihenschaltung auf die einzelnen Bauteile aufgeteilt. Daher gilt hier:

$$U_{Gesamt} = U_{LDR} + U_{R1} \tag{3.2}$$

Erhöht sich der Widerstand am LDR, weil es dunkler wird, so steigt auch der Spannungsunterschied am LDR. Da die Gesamtspannung aber gleich bleiben muss, verringert sich die Spannung am festen Widerstand. Diese Schaltung nennt man **Spannungsteiler**.
Der Arduino misst die Spannung immer im Vergleich zur Erdung (GND). Aber Achtung: **Beim Messen der Spannung darf keine Spannung größer als 5V am messenden PIN anliegen!**

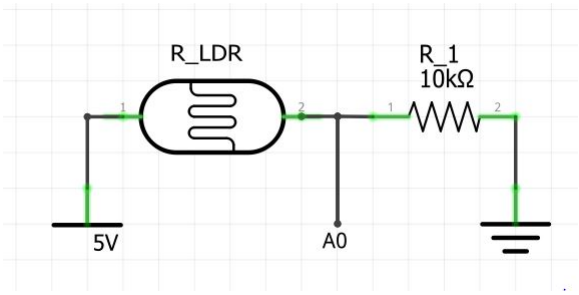

Abb. 3.2: Schaltplan für einen Spannungsteiler am LDR

Als Widerstand eignet sich ein 10kΩ-Widerstand.

Mit den analogen Eingängen können Spannungen im Bereich von 0V bis 5V gemessen werden. Diese geben im Vergleich zu digitalen Eingängen, die nur „Spannung“ und „keine Spannung“ messen können, Messwerte in 4,9mV Schritten an.
Dies liegt daran, dass ein Arduino nicht alle möglichen Spannungen messen kann und diese speichern. Er bräuchte dazu einen sehr großen Speicher, kann aber nur 10 Bit zur Verfügung stellen. Dies entspricht $2^{10} = 1024$ Möglichkeiten. Da die Messgenauigkeit aus der Referenzspannung (5V) und der Auflösung mit 1024 Schritten besteht, ergibt sich eine Genauigkeit von 4,9mV. (vgl. Bartmann (2015), S. 299ff)

Bildungsplanbezug:
Lichtabhängige Widerstände sind in Klasse 10 im Physikunterricht unter *3.3.2 - Elektromagnetismus* angesiedelt.

4. Ausgabemöglichkeiten

Nun haben wir eine Möglichkeit, Messwerte mit dem Arduino aufzunehmen. Diese haben wir bisher direkt im Programm verarbeitet und mit dem Ergebnis gearbeitet. Nun möchten wir aber den Wert auslesen. Da der Arduino keine Anzeige hat, kann man entweder über den PC oder mit einer dazugebauten Anzeige mit ihm kommunizieren. Wir werden uns diese beiden Möglichkeiten nun am Beispiel des LDR anschauen.

4.1. Serielle Kommunikation

Serielle Kommunikation mit Open Roberta
Für die Serielle Kommunikation bei Open Roberta öffnen wir diese im Programm „Open Roberta USB“. Wenn wir nun mit Open Roberta etwas programmiert haben, das auf den Seriellen Monitor zugreift, wird es uns hier ausgegeben.

Nun schreiben wir unser gewünschtes Programm. Zuerst konfigurieren wir wieder den Roboter, das heißt wir fügen einen Lichtsensor hinzu und wählen den Port aus an dem dieser angeschlossen ist.
Danach können wir im Programmcode bei *Aktion* im Reiter *Anzeige* die Option „*Zeige auf Serial Monitor*“ wählen. Dann ändern wir den Text in die Messung des Lichtsensors. Dieser wird in % ausgegeben.
Nun müssen wir noch sagen, wie oft er messen sollen, was wir wieder über eine Wartefunktion lösen können.

Legt man den LDR als nicht näher spezifizierten Sensor an, so kann man auch die analogen Werte des Sensors ausgeben.

4.2. Der LCD

Funktionsweise des LCD
Ein LCD (*liquid crystal display*) ist ein Monitor, der mit Flüssigkristallen gefüllt ist. Um die Funktionsweise zu verstehen, müssen wir zuerst auf die Wellennatur von Licht eingehen.

Betrachtet man Licht als elektromagnetische Welle kann man dieser eine Phase zuordnen. Mit Polarisationsfiltern kann man somit bestimmte Phasen des Lichts abschwächen. Liegen nun zwei (oder mehr) Polarisationsfilter übereinander, kann so das gesamte Licht behindert werden und es kommt kein Licht mehr durch die Filter.

Dieser Effekt wird beim LCD ausgenutzt. Zwischen zwei Polarisationsfolien ist eine sogenannte **nematische** Schicht. Hierin sind Stabmoleküle von Flüssigkristallen eingeschlossen. Diese werden vorbehandelt, dass sie parallel zur Polarisationsebene ausgerichtet sind. Die Polarisationsebene dreht sich dabei mit zunehmender Dicke, sodass das Licht durch die Anordnung und am Spiegel zurück dringen kann.
Je nach Material hat man hier einen grünen oder blauen LCD.

Wird nun ein elektrisches Feld angelegt, drehen sich die Moleküle und richten sich parallel im Feld aus. Das Licht kann nun nicht mehr ungehindert hindurch und in dem Feldbereich wird es deshalb dunkel. (vgl. Meschede et al. (2010), S. 573)

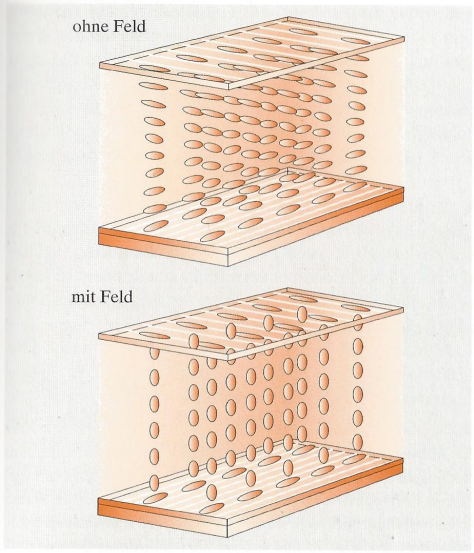

Abbildung 11.77 Anordnung der nematischen Moleküle in einem LCD ohne (*oben*) und mit Feld (*unten*)

Abb. 4.1: Anordnung von Molekülen im LCD

LCD ansteuern (Open Roberta)
Wir werden im folgenden immer einen LCD mit I2C-Modul (Inter Integrated Circuit) nutzen. Dies erleichtert uns die Eingabe und Verbindung mit dem Arduino. Für die Schule ist ebenfalls zu empfehlen, dieses Modul zu nutzen, da es die Situation deutlich vereinfacht.

Wählt man den Konfigurationsblock für den LCD mit I2C in Open Roberta aus, dann sieht man schon wie die Konfigurationen gestaltet sein müssen. Wir benötigen einen GND Pin, danach den 5V Pin und die beiden analogen Inputs A4 und A5. Diese werden in dieser Reihenfolge an das I2C-Modul angeschlossen (von oben nach unten).
In Open Roberta findet ihr unter *Anzeige* den Block um den LCD mit I2C anzusteuern. Die Spalte 0 ist direkt der Beginn und Zeile 0 ist die obere Spalte auf dem LCD. Bedenkt, dass ihr pro Zeile nur 16 Zeichen habt.

5. Der Ultraschallsensor

5.1. Theorie zum Ultraschallsensor

Wir gehen nun vom Bereich der Optik in die Akustik. Der Ultraschallsensor (wir verwenden den HC-SR04) arbeitet mit Schall und dessen Reflektion.

Das Ultraschallmodul sendet, sobald es eine Spannung am *Trigger*-Pin misst, acht 40kHz Impulse aus. Dies geschieht meist mit einem Schwingkristall, näheres hierzu später. Der Schall breitet sich aus und wird an einem Hindernis reflektiert. Kommt dieser am Sensor wieder an, misst der Ultraschallsensor wie viel Zeit vergangen ist. Am *Echo*-Pin wird vom Modul ein Signal ausgegeben, das der Länge der vergangenen Zeit zwischen Aussenden und Empfangen des akustischen Signals des Moduls entspricht. Aus dieser Zeitspanne lässt sich dann berechnen, wie weit entfernt das Hindernis ist.

Bekannt ist, dass die zurückgelegte Strecke s das Produkt aus Ausbreitungsgeschwindigkeit v und der benötigten Zeit t ist. Somit ergibt sich für unseren Sachverhalt die Formel:

$$s_{Gesamt} = v_{Schall} \cdot t \tag{5.1}$$

Da der Schall aber zum Hindernis und wieder zurück geht, haben wir als Gesamtstrecke die doppelte Strecke des Abstandes. Somit ergibt sich für den Abstand:

$$s_{Abstand} = \frac{s_{Gesamt}}{2} \tag{5.2}$$

$$s_{Abstand} = \frac{v_{Schall} \cdot t}{2} \tag{5.3}$$

Die Schallreflektion klappt vor allem bei harten ebenen Oberflächen gut. Sind die Oberflächen uneben, kann es sein, dass der Schall in andere Richtungen gestreut wird und nicht am Modul wieder ankommt. Dies kann auch bei einem schrägen Winkel zum Objekt der Fall sein und der Schall in eine andere Richtung abgelenkt. Auch weiche Oberflächen wie Stoff können den Schall schlucken.

5.2. Abstände mit dem Arduino messen

Ultraschallsensor mit Open Roberta
Wir werden zuerst den einfachen Weg gehen und mit Open Roberta versuchen, Abstände zu messen. Als direkte Ausgabe können wir den Wert entweder über den Seriellen Monitor oder über den LCD ausgeben.

Wie immer muss zuerst der Sensor konfiguriert werden. Auch hier haben wir einen vorgefertigten Block, der uns direkt vorschlägt wie wir den Ultraschallsensor anschließen können.
Der schnellste Weg, um Messwerte zu bekommen, ist jener mit dem Block *Zeige auf Serial Monitor*. Statt eines Textes nutzen wir aus dem Register *Sensor* den Block: „**gib Abstand cm Ultraschallsensor HC-SR04**“. Danach warten wir bis der nächste Wert ausgelesen werden soll.

Abb. 5.1: Programmbeispiel mit Open Robert zum Ultraschallsensor

Die Ergebnisse können wir im Seriellen Monitor, den wir über Open Roberta USB öffnen, anschauen.

Bildungsplanbezug:
Im Bildungsplan für den BNT Unterricht (Klasse 5/6) findet sich unter *3.1.5. Wirbeltiere der Punkt*: (6) den Körperbau und die Lebensweise heimischer Säugetiere als Angepasstheit erklären (z. B. Eichhörnchen, Igel, Maulwurf, Fledermaus).
In vielen Schulbüchern wird hier die Fledermaus und der Ultraschall als Methode zur Orientierung besprochen. Mit einem Arduino und dem Ultraschallsensor kann man dies z.B. praktisch nachbilden und experimentell durchführen.
Erstellt man das Modell aus Holz, entspricht das der Kategorie *3.1.10 Ein Produkt entstehen lassen* - in der es um Holzverarbeitung geht.

6. Temperaturmessung

6.1. Theorie zu den Sensoren

Viele der Temperatursensoren nutzen die Temperaturabhängigkeit der Leitfähigkeit von Metallen und Halbleitern. So auch bei den uns vorliegenden Sensoren. Beide Sensoren für die Temperatur sind NTC-Leiter, das bedeutet der Temperaturkoeffizient ist bei ihnen negativ (*negative temperature coefficient*). Bei Metallen nimmt der Widerstand hingegen mit steigender Temperatur zu, man spricht auch von Kaltleitern oder PTC-Leitern (*positive temperature coefficient*).

Bei Metallen liegt das daran, dass bei steigender Temperatur die Ionenrümpfe ins schwingen kommen und so die Elektronenbewegung behindern.
Dies ist auch so bei Halbleitern, aber diese haben keine fest vorgegebene Ladungsträgerdichte wie Metalle. Hier müssen die Ladungsträger erst frei werden, was durch anheben aus einem gebundenen Zustand zu einem beweglichen Zustand geschieht. Dieser Prozess führt bei Halbleitern zu deutlich mehr Ladungsträgern, sodass ihre Leitfähigkeit dennoch zunimmt.
In Widerstandsthermometern werden hierzu Platinspiralen genutzt. (vgl. Meschede et al. (2010), S. 350)

Zur Messung der Luftfeuchtigkeit hingegen gibt es zwei unterschiedliche Sensoren die verbreitet sind, daher stellen wir beide vor.

„**Kapazitative Sensoren** haben eine nichtleitende, aber wasseranziehende Substanz (etwa Kunststoff-Polymere, keramische Werkstoffe etc.) zwischen den zwei Elektroden eines Kondensators. Absorbiert diese Substanz nun Feuchtigkeit, nimmt diese also auf, ändert sich in der Folge die elektrische Kapazität des Sensors. Dadurch wird die Luftfeuchtigkeit messbar.

Impedanzsensoren nutzen eine hygroskopische (wasseranziehende) Schicht zwischen zwei Gleichstrom-Elektroden. Wird nun Feuchte absorbiert, also von dieser Schicht aufgenommen, ändert sich ihr ohmscher Widerstand. Eben diese Veränderung lässt sich messen und in relativer Luftfeuchtigkeit ausgeben. Als Schicht kommen hierfür beispielsweise Aluminiumoxid oder auch hygroskopische Kunststoff-Polymere zum Einsatz.“ (Quelle: `https://hygrometer.guru/luftfeuchtigkeit-messen/` vom 01.09.2019)

6.2. Der DHT11

Der DHT11 kann sowohl die Luftfeuchtigkeit als auch die Temperatur messen.
Es gibt ihn mit vier Pins oder reduziert auf drei. Bei der reduzierten Variante ist ein Widerstand der bei der einfachen Variante benötigt wird schon verbaut.

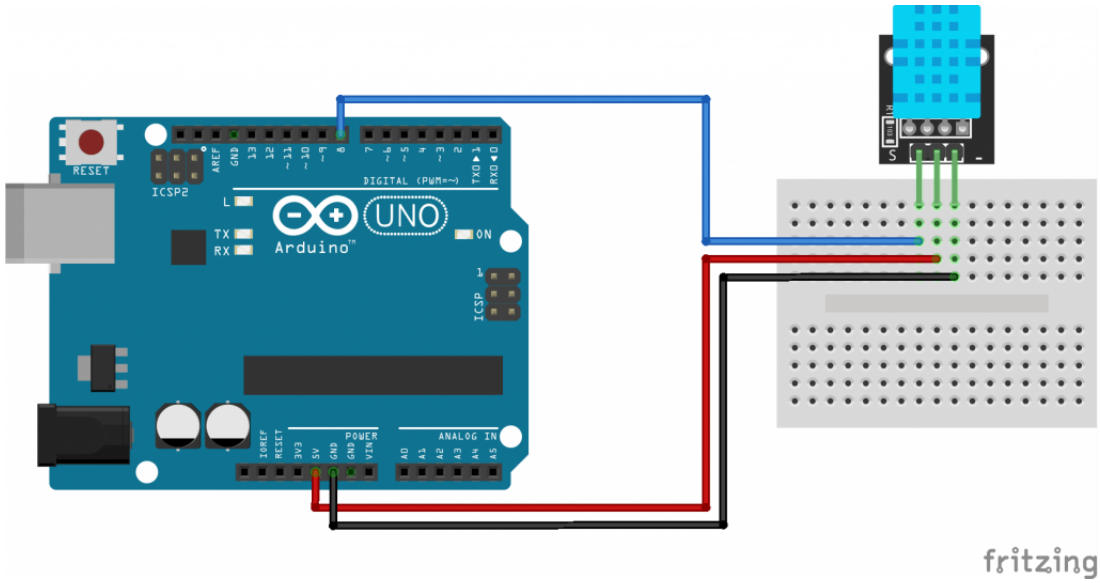

Abb. 6.1: Schaltskizze für den DHT11 von `http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/`

Bildungsplanbezug:
Temperaturmessungen fallen in den Bereich *3.3.3 - Wärmelehre* des Physikunterrichts der 10. Klasse.

Index

Literatur

Bartmann, E. (2015). *Die elektronische Welt mit Arduino entdecken: Mit dem Arduino messen, steuern und spielen ; Elektronik leicht verstehen ; 44 Arduino-Projekte für den Selbstbau* (Stark erw. 2. Aufl., 1. korr. Nachdr Aufl.). Beijing: O'Reilly.

Campbell, N. A., Reece, J. B. & Urry, L. A. (o. J.). *Biologie* (10., aktualisierte Auflage Aufl.). Zugriff auf http://lib.myilibrary.com?id=833397

Gobrecht, H., Eichler, H.-J., Bergmann, L. & Schaefer, C. (Hrsg.). (1978). *Optik* (7. Aufl. Aufl., Bd. zum Gebrauch bei akademischen Vorlesungen und zum Selbststudium / von L. Bergmann und Cl. Schaefer ; Bd. 3). Berlin: de Gruyter.

Lehn, R., Lehn, M. & Tomczyk, M. (2019). *Arduino: Von der blinkenden LED bis zum autonomen Roboter: Unterrichtsmaterialien für Lehrerinnen und Lehrer.* Bad Saulgau. Zugriff am 06.08.2019 auf https://bscw.sfz-bw.de:444/pub/bscw.cgi/371178?op=preview&back_url=325122%3fclient_size%3d958x964

Meschede, D., Vogel, H. & Gerthsen, C. (2010). *Gerthsen Physik* (24., überarb. Aufl. Aufl., Bd. 0). Berlin: Springer. Zugriff auf http://dx.doi.org/10.1007/978-3-642-12894-3 doi: 10.1007/978-3-642-12894-3

Tipler, P. A., Mosca, G. & Basler, M. (2012). *Physik: Für Wissenschaftler und Ingenieure ; [der Begleiter bis zum Bachelor* (6. dt. Aufl., korr. Nachdr Aufl.). Berlin: Springer Spektrum.

7. Widerstände

Widerstände haben einen Farbcode, der angibt welchen Widerstandswert in Ω diese haben. Die folgende Grafik macht die Berechnung leichter.

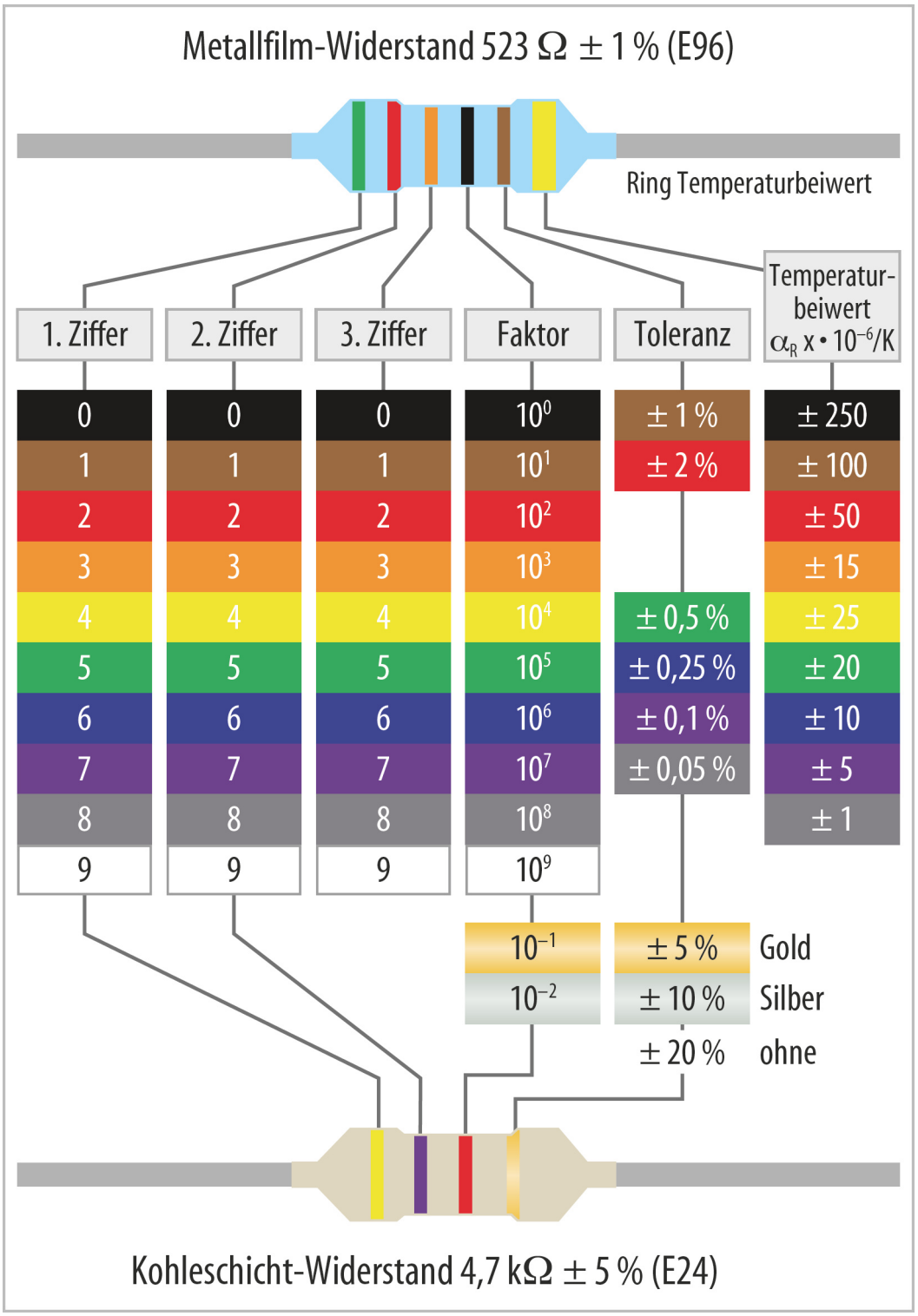

Abb. 7.1: Farbcodes von heise.de