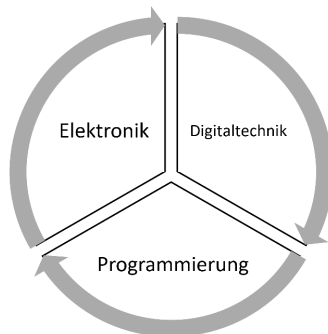


In diesem vorbereitenden Teil sollen erste Grundlagen der Elektronik gelegt werden, die in den Folgeprojekten praktische Anwendung finden. Sämtliche hier vorgestellten Grundkenntnisse sind für den kenntnisreichen, verständigen Umgang mit Mikrocontrollern unerlässlich und daher auch vollständig Schulrelevant.¹

Strukturierung und Überblick

Für den Zweck der Mikrocontrollertechnik kann ein **System** in drei Hauptkomponenten zergliedert werden:



Als Referenzmodell für Einordnung und Strukturierung – sowie als didaktisches Hilfsmittel – dient das bekannte **EVA-Modell**, *Eingabe – Verarbeitung – Ausgabe*. Dieses bringt Struktur insbesondere in die Vielzahl möglicher Sensoren und Aktoren und verortet die unterschiedlichen Möglichkeiten der Verarbeitung (analoge Schaltungen, digitale Schaltungen, Programm auf einem Mikrocontroller, Hardware programmiert in Form von etwa FPGAs mit VHDL usw.).

Modell von Strom und Spannung

Es ist sehr hilfreich, sich physikalische Sachverhalte in Modellen vorzustellen². Dabei darf der Modellcharakter nicht vergessen werden und Modelle sind geeignet auf Situationen zu übertragen. In der Elektrotechnik ist es hilfreich, sich Strom – also fließende Elektronen – als Wasser(strom) vorzustellen und die Spannung als Pumpe/Druck. Der Widerstand, **R**(Ohm, Ω), der diesem Stromfluss entgegengebracht wird, ist der elektrische Widerstand. Dabei entsteht Wärme und eine Leitung/Rohr kann davon nur so viel aushalten (Leistung [Watt]), bevor sie verglüht. An Hand dieser Modellvorstellung hat man ein anschauliches Bild vor Augen und das Lernen wird unterstützt.

Physikalisch ist der elektrische Strom, **I** [Ampere, **A**], ein Fließen von negativ geladenen Ladungen, den Elektronen, über ein leitendes Material, insbesondere Metalle, etwa in Form von Leitungen/Drähten. Die treibende Kraft für diese Bewegung ist die elektrische Spannung, **U** [Volt, **V**], welche einen Potentialunterschied darstellt. Ein Potentialunterschied ist eine Ladungsdifferenz. So herrscht zwischen 0 V und 5 V eine Spannung/Potentialdifferenz von 5 V, ebenso aber zwischen -2 V und 3 V. Wenngleich fast immer Masse/-/ground (GND) als Bezugspunkt hergenommen und als 0 Volt definiert wird, ist es wichtig zu verstehen, dass es bei Spannungen immer um den Ladungsunterschied zwischen zwei Punkten geht. Dem Stromfluss wird ein gewisser Widerstand, **R** [Ohm] entgegengesetzt, wobei jedes Bauteil – auch Leitungen – einen gewissen Widerstand besitzt. Der grundlegende Zusammenhang zwischen diesen drei **Grundgrößen der Elektronik**, *Spannung, Strom und Widerstand*, wird durch das **Ohmsche Gesetz** festgehalten: $U = R \cdot I$, welches sich mit der Eselsbrücke “URI” leicht merken lässt. Also: *Spannung = Widerstand · Stromstärke*

¹ Sollten Sie das Thema also in der Praxis unterrichten ist dieses Wissen mit zu vermitteln.

² Natürlich ist die ganze Physik ein Modell.

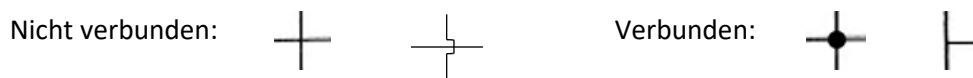
Physikalisch bewegen sich Elektronen als negative Ladungen vom Minuspol zum Pluspol. Dies wurde jedoch erst herausgefunden, nachdem die Elektronik schon verbreitet war, eine Änderung des Umgangs mit der Stromrichtung in der Praxis war nicht mehr möglich. Daher wurde diese Richtung als **physikalische Stromrichtung** festgelegt, so passiert es im Material tatsächlich. Die Technik jedoch bezieht sich immer noch auf das Modell, welches vor Kenntnis dieser Zusammenhänge entwickelt wurde. Gemäß diesem Modell werden nicht die Elektronen als Stromfluss betrachtet sondern die Stellen die sie hinterlassen, die Löcher, positive Ladungsträger. Diese fließen vom Pluspol zum Minuspol, was als **technische Stromrichtung** bekannt ist.

Merke: In der Elektronik wird der **Stromfluss** immer **von + nach –** angegeben.

Grundbauelemente

Leitungen

Leitungen sind Metalldrähte unterschiedlicher Dicke, die einzeln oder auch Meterweise auf einer Rolle gewickelt gekauft werden können. In Schaltplänen lässt es sich nicht vermeiden, sich überkreuzende Leitungen zu zeichnen. Zwei sich überkreuzende Leitungen sind nur verbunden, wenn sie mit einem Punkt versehen sind oder ein T bilden.

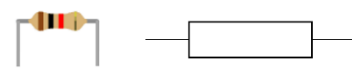


Widerstand

Ein Widerstand setzt dem Stromfluss einen Widerstand entgegen.

Dadurch wird der Stromfluss durch ein Bauelement gezielt begrenzt.

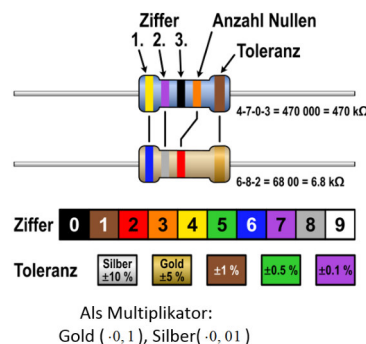
Je größer der Widerstandswert, desto mehr Widerstand wird dem Stromfluss entgegengesetzt, desto mehr Spannung fällt an ihm ab. An einem doppelt so großen Widerstand fällt auch die doppelte Spannung ab.



Merksatz:

>>Viel Öhmchen, wenig Strömchen<<

Typisches Beispiel ist eine LED, die nicht direkt an die Spannungsquelle angeschlossen werden kann, da sonst ein zu großer Stromfluss die Folge wäre, der die LED zerstört. Hier wird dann ein Widerstand vorgeschaltet (**Vorwiderstand**), der die überschüssige Spannung abfängt und den Stromfluss begrenzt. Wichtige Werte sind der Widerstandswert in *Ohm* und die Leistung in *Watt*, die der Widerstand verträgt. Üblich im Mikrocontrollerbereich sind ¼-Watt Widerstände, d.h. der Widerstand verträgt nur eine Leistung von ¼-Watt.³ Der Widerstandswert in Ohm ist *farblich codiert* auf dem Widerstand selbst abgedruckt, in sogenannten *E-Reihen* abgestuft verfügbar und kann mit einem Multimeter-Messgerät direkt nachgemessen werden.



³ Leistung = Spannung · Stromstärke, $P=U \cdot I$. Liegt der Widerstand bspw. an 3 Volt an, so verträgt er einen maximalen Strom von 83 mA (Nachweis: Übung). Andernfalls läuft er Gefahr zu schmelzen.

⁴ https://commons.wikimedia.org/wiki/File:Resistor_Color_Code.svg?uselang=de (gemeinfrei)

Produktionstechnisch bedingt gibt es bei Widerständen **Toleranzen**. Diese werden durch einen separaten Ring dargestellt. Je geringer die Toleranz desto teurer der Widerstand.

Die Angabe der Toleranz hat einen größeren Abstand zu den Ringen, so dass die Ausrichtung des Widerstands zum Ablesen klar ist. Der *Toleranzring* steht *immer rechts*. Als gute Merkhilfe gilt ferner, dass Silber und Gold stets Toleranzen oder Multiplikatoren angeben. Der Widerstandswert wird mit zwei oder drei Ringen codiert. Je geringer die Toleranz, desto feiner lässt sich mit drei Ringen der Widerstandswert angeben. Nicht jeder beliebige Widerstandswert ist vorhanden, stattdessen werden Widerstände in Reihen angeboten, besagten **E-Reihen**.

E3 (>20% Toleranz): 1,0 / 2,2 / 4,7 Ohm
 E6 (20% Toleranz): 1,0 / 1,5 / 2,2 / 3,3 / 4,7 / 6,8 Ohm
 E12 (10% Toleranz): 1,0 / 1,2 / 1,5 / 1,8 / 2,2 / 2,7 / 3,3 / 3,9 / 4,7 / 5,6 / 6,8 / 8,2 Ohm
 usw.

Durch die Multiplikatoren verbirgt sich dahinter eine ganze Reihe von Widerständen, hier am Beispiel der Reihe E12:⁵

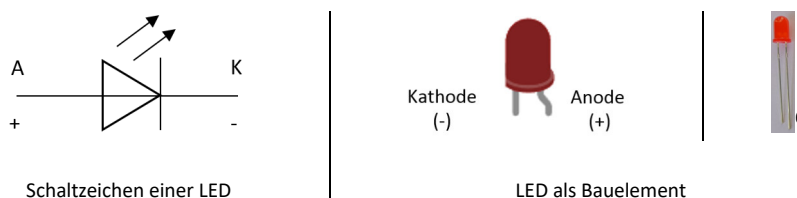
1 Ω	10 Ω	100 Ω	1 kΩ	10 kΩ	100 MΩ	1 MΩ	10 MΩ
1,2 Ω	12 Ω	120 Ω	1,2 kΩ	12 kΩ	120 MΩ	1,2 MΩ	12 MΩ
1,5 Ω	15 Ω	150 Ω	1,5 kΩ	15 kΩ	150 MΩ	1,5 MΩ	15 MΩ
1,8 Ω	18 Ω	180 Ω	1,8 kΩ	18 kΩ	180 MΩ	1,8 MΩ	18 MΩ
2,2 Ω	22 Ω	220 Ω	2,2 kΩ	22 kΩ	220 MΩ	2,2 MΩ	22 MΩ
2,7 Ω	27 Ω	270 Ω	2,7 kΩ	27 kΩ	270 MΩ	2,7 MΩ	27 MΩ
3,3 Ω	33 Ω	330 Ω	3,3 kΩ	33 kΩ	330 MΩ	3,3 MΩ	33 MΩ
3,9 Ω	39 Ω	390 Ω	3,9 kΩ	39 kΩ	390 MΩ	3,9 MΩ	39 MΩ
4,7 Ω	47 Ω	470 Ω	4,7 kΩ	47 kΩ	470 MΩ	4,7 MΩ	47 MΩ
5,6 Ω	56 Ω	560 Ω	5,6 kΩ	56 kΩ	560 MΩ	5,6 MΩ	56 MΩ
6,8 Ω	68 Ω	680 Ω	6,8 kΩ	68 kΩ	680 MΩ	6,8 MΩ	68 MΩ
8,2 Ω	82 Ω	820 Ω	8,2 kΩ	82 kΩ	820 MΩ	8,2 MΩ	82 MΩ
Grundwerte	→ über Multiplikatoren →						

Der Name der E-Reihe besagt, in wieviel Abstufungen eine Dekade unterteilt wird. Bei E12 sind dies also 12 Widerstandswerte (1 Ohm bis 10 Ohm werden in 12 Zwischenwerte aufgeteilt). Wollen Sie bspw. 300 Ohm als Widerstand haben, können Sie nur 330 Ohm oder 270 Ohm in E12 hernehmen.

LED

Eine **Leuchtdiode** ist ein verbreiteter Aktor, da durch das Leuchten seine Reaktion sichtbar gemacht wird. Eine Leuchtdiode als spezielle (leuchtende) Version einer Diode – welches ein elektrisches Ventil darstellt – lässt den Strom nur in eine Richtung durch (Durchlassrichtung), ist also polungsabhängig. Ihre Anschlüsse heißen Kathode (-) und Anode (+).

Hier ist eine Leuchtdiode in Durchlassrichtung geschaltet:

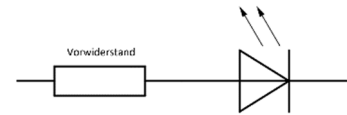


Die Anode (+) ist oftmals durch eine *gebogenes Bein* gekennzeichnet und/oder *länger* als die Kathode. Als Richtwerte gelten 2 V Spannungsabfall an der LED und 20 mA (also 0,02 A) Stromfluss.

⁵ <https://electronicsplanet.ch/en/resistor/e12-series.php>

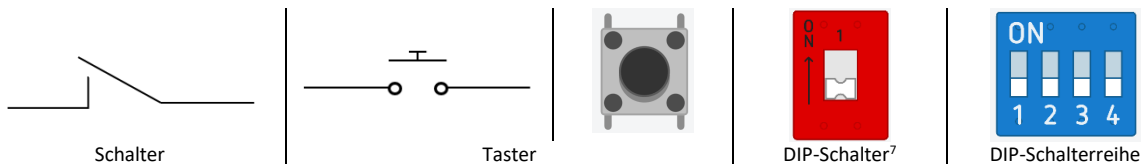
⁶ https://commons.wikimedia.org/wiki/File:5mm_LED_Light-emitting_diode_Red_1480277_8_9_HDR_Intensive.jpg?uselang=de

Eine LED wird in der Praxis fast immer mit einem **Vorwiderstand** – also einem vor die LED geschalteten Widerstand – betrieben, der den Stromfluss begrenzt und die Spannung auf den zur LED passenden Wert reguliert.



Taster/Schalter

Der Taster oder Schalter ist ein verbreiteter Sensor, der Dauerhaft (Schalter) oder Zeitweilig (Taster) einen Kontakt schließt oder öffnet.



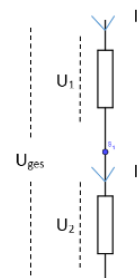
Grundschaltungen

In der Praxis werden Bauelemente primär auf zwei Arten miteinander verschaltet, der Reihenschaltung und der Parallelschaltung. Fast immer treten diese *in Kombination* auf.

Reihenschaltung

Widerstände in Reihe geschaltet addieren sich in ihrem Gesamtwiderstandswert. Die Spannungen teilen sich gemäß den Widerstandswerten anteilig auf. In Reihe geschaltete Widerstände stellen demnach einen **Spannungsteiler** dar, eine häufig benutzte Schaltung.

Es gilt also $U_{ges} = U_1 + U_2$, während die gleiche Menge an Strom, I , durch beide Bauteile fließt.



Parallelschaltung

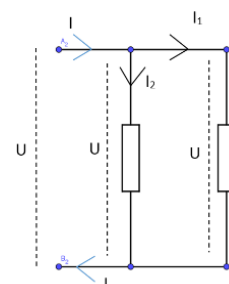
Bei der Parallelschaltung von Widerständen liegt an allen die gleiche Spannung an, gemäß ihrem Widerstandswert findet eine Stromteilung statt. Es liegt demnach ein **Stromteiler** vor (hier gilt: $I = I_1 + I_2$).

Der Wert des **Ersatzwiderstandes** R_{ges} – also ein Widerstand, der den Wert aller Widerstände repräsentiert – berechnet sich bei einer Parallelschaltung über die

Addition der Kehrwerte der Einzelwiderstände: $\frac{1}{R_{ges}} = \frac{1}{R_1} + \frac{1}{R_2}$,

allgemein $\frac{1}{R_{ges}} = \frac{1}{R_1} + \frac{1}{R_2} + \dots + \frac{1}{R_n} = \sum_{i=1}^n \frac{1}{R_i}$. Für den häufigen Fall der Parallelschaltung von zwei

Widerständen ergibt sich $R_{ges} = \frac{R_1 \cdot R_2}{R_1 + R_2}$.



⁷ DIP = Dual in-line package, eine Bauform mit zwei parallel angeordneten Anschlussreihen

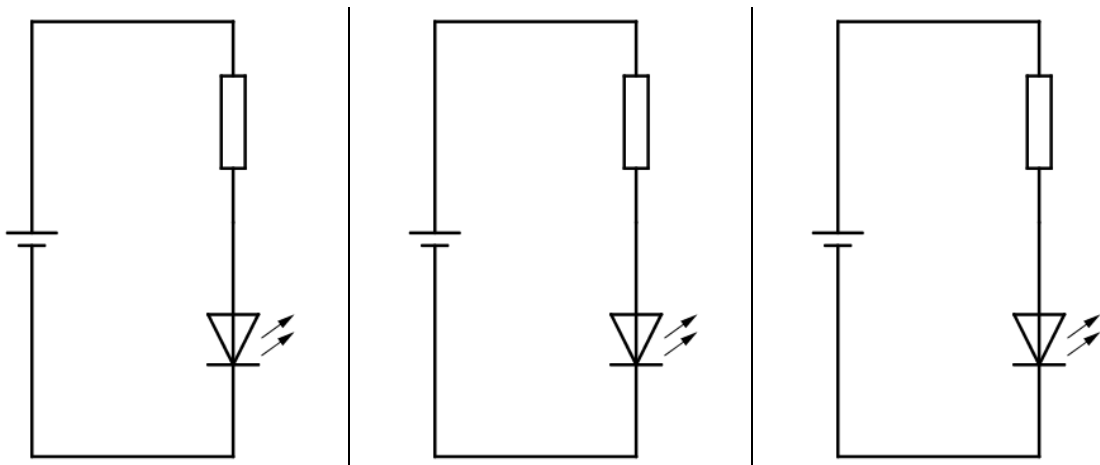
Übungen

Aufgabe 1: Ohmsches Gesetz

Eine häufige Anwendung im Bereich der Mikrocontrollerelektronik besteht darin, korrekt dimensionierte Vorwiderstände für etwa LEDs zu berechnen. Dabei ist zunächst in den entsprechenden Datenblättern der Bauteile nachzulesen, welche Werte für Strom und Spannung anzusetzen sind, anschließend führt das Ohmsche Gesetz, $U = R \cdot I$, zum Ziel. Einmal berechnet stehen solche Größen dann als Erfahrungswerte zur Verfügung.

Bedenken Sie, dass in der Elektronik viel auf Erfahrungswerten basiert und Bauteildimensionierungen *nicht* rein mathematisch erfolgen, zumal etwa Widerstände auch nur in gewissen Abstufungen (E-Reihen) vorhanden sind. Sie kommen mit Berechnungen aber zumindest in den richtigen Bereich dessen, was in Schaltplänen angegeben ist oder Sie dauerhaft in Ihren eigenen Schaltungen nutzen können.

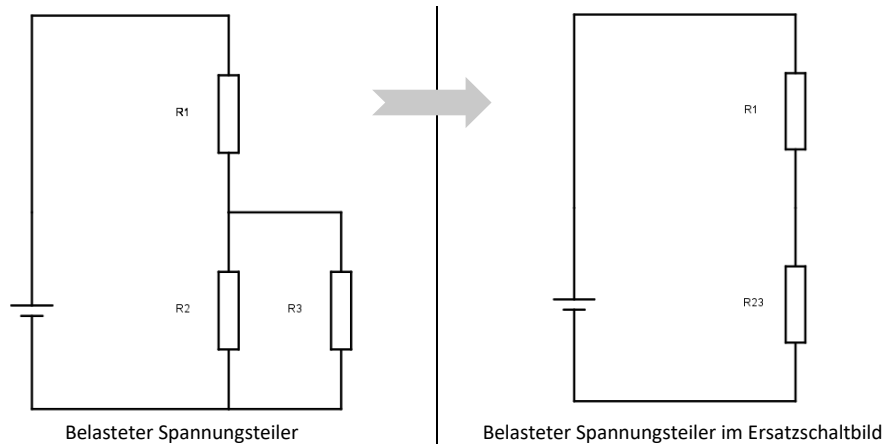
- Berechnen Sie den Vorwiderstand für eine an 9 Volt betriebene LED, an der 2 V Spannung abfallen und durch die ein Stromfluss von 20 mA fließen soll.
- Berechnen Sie den Vorwiderstand wie in (a), jedoch einmal für einen Betrieb an einer 3 V Spannungsquelle sowie an einer 5 V Spannungsquelle.
- Bilden Sie Ihre theoretisch ermittelten Widerstandswerte aus (b) auf die realen Werte aus den E-Reihen ab (E12), geben Sie also einen Widerstandswert an, der tatsächlich als fertiges Bauteil existiert.
- Ergänzen Sie die unvollständige Schaltbilder mit Ihren Werten aus (c). Sie haben damit die Dimensionierung einer Standardschaltung vorgenommen und diese in Zukunft als Vorlage zur Verfügung. Werte werden in der Elektronik direkt an das Bauteil geschrieben.



- Bei 5 V Spannung, die der Arduino liefert, werden für rote LEDs 220 Ohm oder 330 Ohm Vorwiderstände verwendet. Berechnen Sie den Stromfluss durch die LED und begründen Sie, welchen Vorwiderstand bei Dauerbetrieb besser geeignet ist.
- Ein Sensor muss mit einer Spannung von 3 Volt betrieben werden, Ihr Mikrocontroller liefert aber 5 Volt Betriebsspannung. Wie sorgen Sie für den korrekten Spannungswert?
- Berechnen Sie den Vorwiderstand für drei an 9 Volt betriebene LEDs in Reihenschaltung, an denen je 2 V Spannung abfallen und durch die ein Stromfluss von 20 mA fließen soll.

Aufgabe 2:

Das Konzept des Ersatzschaltbildes in Kombination mit Reihen- und Parallelschaltung kommt unter anderem bei einem **belasteten Spannungsteiler** vor. Ein Verbraucher – repräsentiert durch den Widerstand⁸ – hängt parallel an einem der beiden in Reihe geschalteten Widerstände und stellt die Last dar:



a) Berechnen Sie den Ersatzwiderstand R_{23} für eine Spannungsquelle von 9 V und $R_2=R_3=470$ Ohm, $R_1=100$ Ohm.

b) Berechnen Sie den Ersatzwiderstand R_{12} , wenn R_3 nicht vorhanden ist.

Aufgabe 3: Widerstand und Leistung

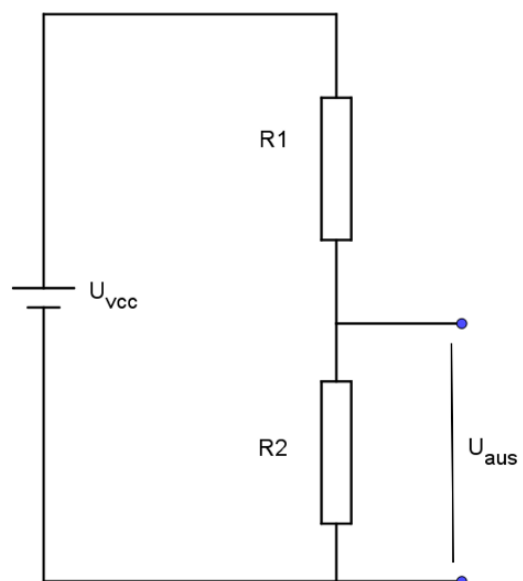
Üblich im Mikrocontrollerbereich sind $\frac{1}{4}$ -Watt Widerstände. Es wurde im Text behauptet, dass ein an 3 Volt anliegender $\frac{1}{4}$ -Watt Widerstand einen maximalen Stromfluss von 83 mA verträgt. Weisen Sie dies rechnerisch nach.

Aufgabe 4: Herleitung Spannungsteilerformel

Ein Spannungsteiler ist einer der am häufigsten vorkommenden Schaltungen und Situationen, daher lohnt sich die Herleitung einer fertigen Formel, in die dann in der Praxis bekannte Werte einfach angegeben werden können. Leiten Sie eine allgemeine Formel für die Ausgangsspannung (U_{aus}) mit den Bezeichnern des abgebildeten Spannungsteilers her.

Tipp:

Denken Sie über das Verhältnis von Spannungen und den zugehörigen Widerständen nach.



⁸ Ein Widerstand ist nicht immer nur Hilfsbauelement. Bei einer elektrischen Heizung ist er der Hauptakteur, da an ihm als Verbraucher elektrische Energie in Wärme umgewandelt wird.

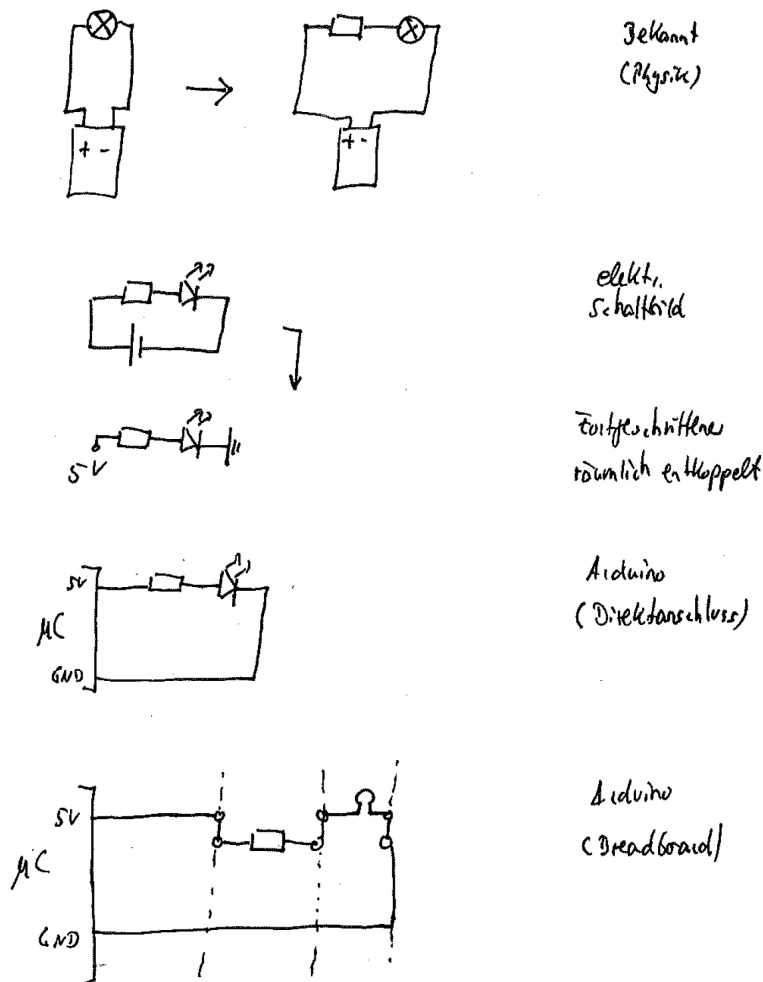
Aufgabe 5: Modelle

Eine klassische Modellvorstellung für die drei elektrotechnischen Grundgrößen Strom, Spannung und Widerstand ist das Wasserflussmodell. Wie jedes Modell hat es Vor- und Nachteile. Diese müssen insbesondere einer Lehrkraft bekannt sein, um Modelle zielgerichtet einsetzen zu können und – durch unangemessenen Einsatz – etwaigen Fehlvorstellungen vorzubeugen.

Schätzen Sie die Vor- und Nachteile des Wasserfallmodells für den Einsatz im Schulunterricht begründet ein. Fragen Sie sich dabei insbesondere, welche Fehlvorstellungen entstehen könnten und wie man diese vermeiden/reduzieren kann.

Aufgabe 6: Schaltbildebene und Schaltbildarten

Eines der Probleme beim Erlernen der Elektronik ist das verständige Lesen und Erstellen von Schaltbildern. Erschwerend kommt hinzu, dass es zum einen mehrere Möglichkeiten gibt, ein und dieselbe Schaltung darzustellen, und zum anderen verschiedene Schaltbildebene und Schaltbildarten gibt. Im Folgenden ist eine solche Progression als Tafelbild dargestellt:



Machen Sie sich diese verschiedenen Arten der Darstellung fachlich klar und überlegen Sie sich, wie sie mit diesen Dingen im Unterrichtsgang umgehen könnten, wann also was wie thematisiert werden könnte.

Arbeitsblatt 2 – Entwicklungsumgebung

In diesem Teil wird die Arbeitsumgebung eingerichtet und ein erstes Programm zum Laufen gebracht. Um die Umgebung vollumfänglich auf Funktion prüfen zu können ist ein angeschlossener Mikrocontroller (Arduino/Funduino) notwendig.

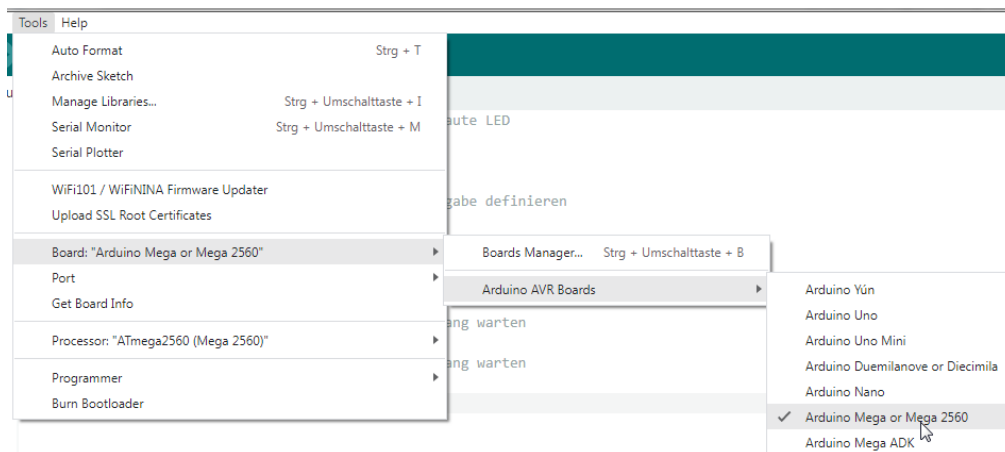
Achtung:

Installationsprozeduren können sich ändern. Nutzen Sie ggf. die Installationsreferenzen des IDE-Anbieters (www.arduino.cc).

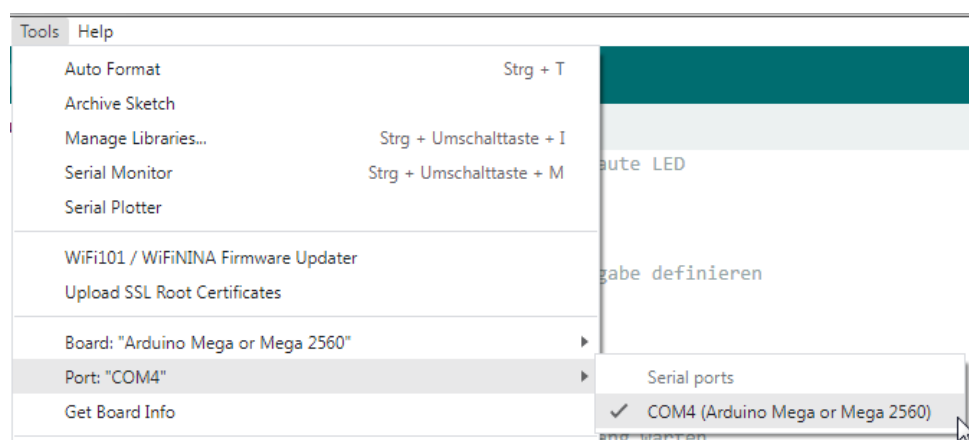
IDE installieren

Die Programmierung der Arduino-Mikrocontrollerplattform erfolgt über die Arduino-IDE, die unter www.arduino.cc kostenlos heruntergeladen werden kann. Einen Online-Editor gibt es ebenfalls. Im Idealfall läuft nach der Installation alles, das Board kann über den USB-Port angeschlossen werden und los geht es. Die Version 2.x wurde komplett neu programmiert, die 1.x-Versionen werden als *classic* bezeichnet und laufen auch noch auf älteren Rechnern. Nutzen Sie die jeweils aktuellste Version.

Unter >Werkzeuge→Board< wird das richtige Board ausgewählt. Wenn Sie *Funduino* nutzen wählen Sie >Mega2560<.



Ist der Arduino angeschlossen müssen Sie neben den Einstellungen des >Boards< wie obig gezeigt noch die COM-Schnittstelle einstellen. Wenn Sie *Funduino* nutzen und die obigen Einstellungen getätigt haben sehen Sie einen >Mega2560< angezeigt. Wählen Sie ihn aus und es kann losgehen.

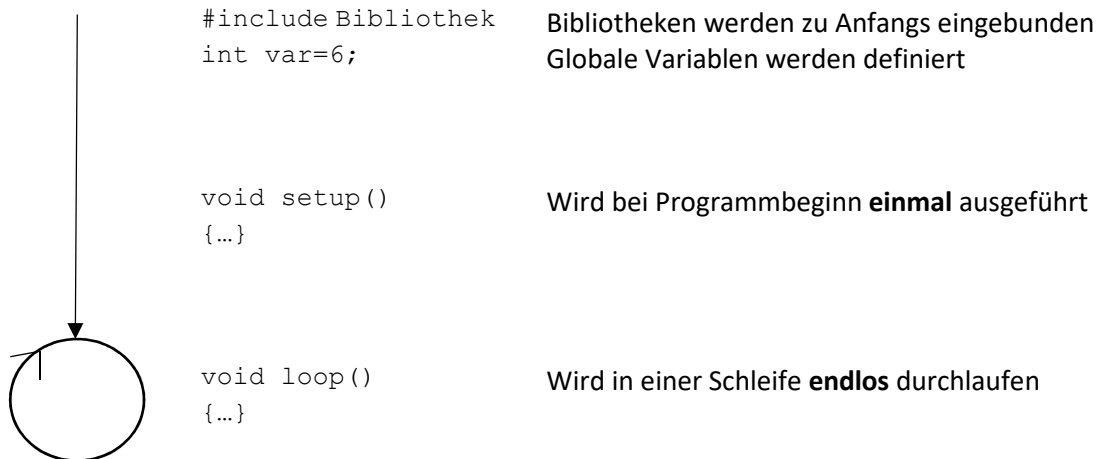


Treiber:

Der Treiber wird nicht immer automatisch erkannt und installiert. Im Verlauf der Installation lässt sich der Treiber auch selber auswählen. Er befindet sich im Arduino-Programmordner im Unterordner >Drivers<.

Programmstruktur

Die Arduino-Sprache ist eine Untermenge von C/C++. Der strukturelle Rahmen ist sehr einfach. Globale Variablen und Bibliotheken werden zuerst definiert, dann gibt es die Methoden `setup` und `loop`. Zunächst wird `setup` aufgeführt und einmalig durchlaufen. Inhaltlich trägt man hier vor allem Initialisierungen – etwa die Portkonfiguration (siehe unten) – ein. Die Methode `loop` hält was sie verspricht, sie wird unendlich oft durchlaufen.

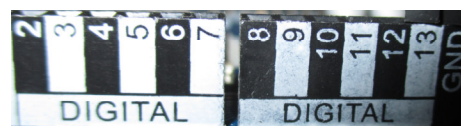


Arduino-Mikrocontroller

Die Arduino-Plattform ist frei, daher bauen verschiedene Hersteller ihre eigenen Exemplare und benennen sie nach Wunsch (Funduino, Elegoo, ...), alle sind aber kompatibel mit der Arduino-Basisplattform. Die Version 3, als R3 bezeichnet, ist sehr verbreitet, aktuell ist momentan R4. Dort erfolgte ein Prozessorwechsel, achten Sie daher auf relevante Unterschiede bei der Arbeit mit verschiedenen Versionsnummern.

Die Interaktion mit der Umwelt geschieht bei einem Mikrocontroller über seine Ein-/Ausgabepins. Dabei können diese analog oder digital arbeiten. Eine Besonderheit bei digitaler Arbeitsweise ist der Betrieb als pulswidenmoduliertes Signal (PWM), also ein steuerbarer 0/1-Anteil pro Signalperiode. Im jeweiligen Handbuch des Mikrocontrollers findet sich die genaue Beschaltung⁹.

Jeder I/O-Pin kann als Eingabe oder als Ausgabe genutzt werden. Deshalb ist bei der Programmierung vor Nutzung eines Pins zunächst einmal sein Modus festzulegen. Das geschieht mittels der Methode `pinMode`, die als Argument die Pinnummer und den Modus `INPUT` oder `OUTPUT` erhält und im `setup` notiert wird (auf Groß-/Kleinschreibung achten). Jeder **Pin** darf mit **maximal 20 mA** belastet werden. Erfolgt die Spannungsversorgung des Mikrocontrollers per USB ist zu bedenken, dass diese Schnittstelle maximal 500 mA Strom liefern kann. Pins, die **PWM** (Pulsweitenmodulation – ein pseudoanaloges Verfahren) unterstützen, haben eine **Tilde (~)** in der Beschriftung. Die **Ports** sind sowohl **auf der Platine als auch von außen auf den Steckleisten beschriftet**. Nutzen Sie die Außenbeschriftung, da sonst leicht ein Verrutschen im Pin passiert.



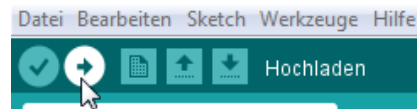
⁹ <https://docs.arduino.cc/hardware/uno-rev3>

Erstes Programm: Board-LED im Sekundentakt blinken lassen

Das *Hallo-Welt* der Mikrocontrollertechnik ist es, eine LED zum Leuchten zu bringen. Arduinos haben eine auf der Platine bereits installierte LED, die über den Pin13 angesprochen werden kann und in der Konstanten `LED_BUILTIN` hinterlegt ist.



Tippen Sie das folgende Programm in den Editor und reflektieren dabei die Befehle. Das Programm wird mittels *Haken* überprüft und kann per *Rechtspfeil* auf den Mikrocontroller übertragen werden.



```
void setup()
{
  pinMode(LED_BUILTIN, OUTPUT);    //eingebaute LED als
                                   //Ausgabe definieren
}

void loop()
{
  digitalWrite(LED_BUILTIN, HIGH); // LED anschalten - HIGH-Pegel (5V)
  delay(1000);                     // 1000 ms = 1 sec. lang warten
  digitalWrite(LED_BUILTIN, LOW);  // LED ausschalten - LOW-Pegel (0V)
  delay(1000);                     // 1000 ms = 1 sec. lang warten
}
```

Das gleiche würde erreicht, wenn Pin13 an statt der Konstanten verwendet werden würde. Der benutzte Pin wird sinnvoller Weise als globale Variable deklariert und initialisiert:

```
int ledPin = 13;                    //globale Variable, die eingebaute LED

void setup()
{
  pinMode(ledPin, OUTPUT);          //eingebaute LED als Ausgabe definieren
}

void loop()
{
  digitalWrite(ledPin, HIGH);       // LED anschalten - HIGH-Pegel (5V)
  delay(1000);                     // 1000 ms = 1 sec. lang warten
  digitalWrite(ledPin, LOW);        // LED ausschalten - LOW-Pegel (0V)
  delay(1000);                     // 1000 ms = 1 sec. lang warten
}
```

Wenn Sie das Programm erstellt und auf den Mikrocontroller übertragen haben läuft es solange, bis die Stromversorgung des Mikrocontrollers unterbrochen wird oder ein neues Programm eingespielt wird. Klappt alles haben Sie die Entwicklungsumgebung demnach richtig konfiguriert, das erste Programm geschrieben und einen klassischen Aktor (eine Leuchtdiode, LED) angesteuert.

Achtung:

Eine Sketch-Datei benötigt immer einen eigenen Ordner. Sie können also nicht mehr als eine Quellcode-Datei in einem Ordner verwalten. Die Entwicklungsumgebung wird einen darauf auch hinweisen.

Hintergrund - Digitale Ports

Das Einlesen eines digitalen Wertes erfolgt über `digitalRead`, die Ausgabe über `digitalWrite`. `DigitalRead` liefert LOW oder HIGH zurück, `digitalWrite` schreibt/setzt LOW oder HIGH, sprich 0 Volt oder 5 Volt (bzw. 3 Volt, der absolute Spannungswert ist für das Verständnis irrelevant und in der Programmierpraxis wird mit LOW und HIGH operiert statt mit konkreten Werten).

Beispiel:

```
pinMode(4, INPUT);           //setzt Pin4 als Eingabepin
wert = digitalRead(4);       //Pin4 auslesen, LOW oder HIGH in wert
                               //speichern
```

Beispiel:

```
pinMode(4, OUTPUT);          //setzt Pin4 als Ausgabepin
wert = digitalWrite(4, HIGH); //Pegel HIGH an Pin4 setzen
```

Anmerkung:

- Die Pins 0 (RX=Empfangen) und 1 (TX=Senden) werden für die serielle Schnittstelle benutzt. Um Probleme zu vermeiden nutzen Sie diese Pins am besten nicht für andere Dinge.
- Arduinos haben eine eingebaute LED, die über Pin13 angesprochen wird.

Ganz zentral für die Vorstellung und korrekte Integration elektrischer Schaltungen ist die Art und Weise, wie ein digitaler Port elektrisch verwendet wird. Einen digitalen Pin kann man sich also am besten als programmierbaren Schalter vorstellen. Der Befehl `digitalWrite(Pinnummer, HIGH)` sorgt für +5 V am Pin, der Befehl `digitalWrite(Pinnummer, LOW)` sorgt für 0 V am Pin, darüber ist der Stromkreis geschlossen.

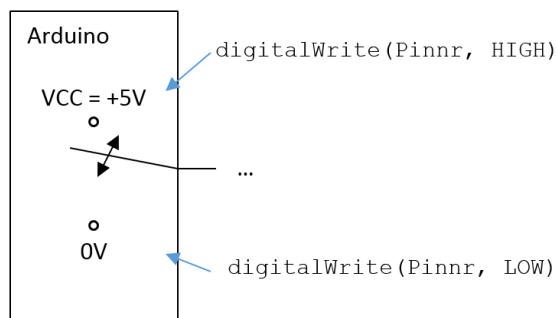


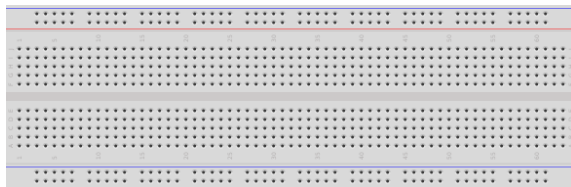
Abbildung 1: Modellvorstellung eines digitalen Ports

Arbeitsblatt 3 – Externe Schaltungen

Die Entwicklungsumgebung läuft, erste Programmkonstrukte sind bekannt und wurden verwendet. Darauf aufbauend wird nun mit externen Schaltungen gearbeitet und die Grundlagen der Elektronik mit der Mikrocontrollerwelt in Verbindung gebracht.

Steckbrett / Breadboard

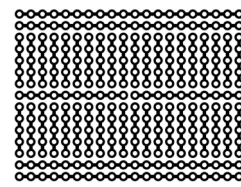
Zum Aufbau von Prototypen elektronischer Schaltungen werden häufig Steckbretter verwendet. Diese sind von besonderem Aufbau, Schaltbilder lassen sich daher *nicht* direkt auf sie übertragen.



Standard-Steckbrett¹⁰

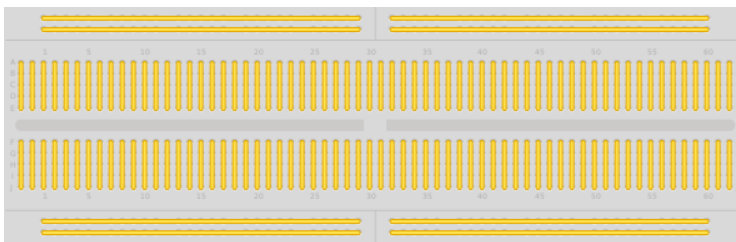


Interner Aufbau¹¹



Schematisch¹²

Es gibt auch Modelle, die in der Mitte unterbrochen sind.



Halbdurchverbundenes Steckbrett im internen Aufbau¹³



Jetzt vom obigen Typ

Lassen Sie Platz beim Schaltungsaufbau und achten Sie besonders darauf, nicht in einer Zeile/Spalte zu verrutschen, neben Wackelkontakten die häufigsten Fehler.

Externe LED betreiben

Statt der internen LED soll nun eine externe LED verwendet werden. Was ist zu tun? Zunächst ist die LED auszuwählen, verschiedene Farben unterscheiden sich in ihren elektrischen Parametern (Spannungsabfall, Stromfluss). Dann ist der passende Vorwiderstand für die Spannung zu berechnen, die der Arduino liefert. Diese Schaltung ist dann aufzubauen und anschließend das Programm zu schreiben, das dem Ganzen Leben einhaucht. Gehen wir einmal ausführlich durch die Materie und verbinden damit auch gleich die elektronischen Grundlagen.

Für eine rote LED kann als Richtwert eine Spannung von 2 V an der LED angenommen werden, ab 0,7 Volt Durchlassspannung schaltet sie durch, beginnt also zu leuchten. Der Arduino liefert 5 V am Pin.

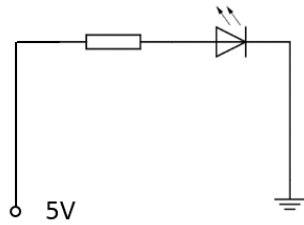
Die Schaltung zum Dauerbetrieb einer LED sieht bekanntlich so aus:

¹⁰ https://commons.wikimedia.org/wiki/File:Breadboard_full_plus.svg

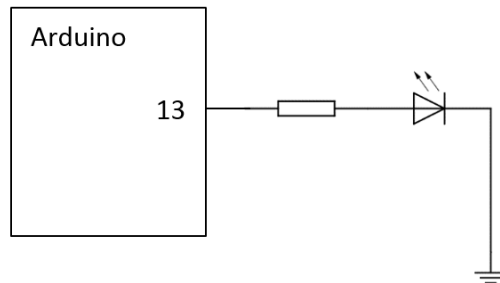
¹¹ https://commons.wikimedia.org/wiki/File:Metal_contacts_within_a_breadboard.jpg

¹² https://commons.wikimedia.org/wiki/File:Breadboard_scheme.svg?uselang=de

¹³ <https://commons.wikimedia.org/wiki/File:BreadboardContacts.svg>

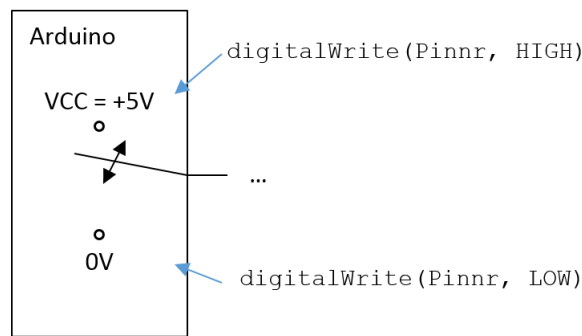


Die gedankliche Hürde ist es zu verstehen, dass +5V im Mikrocontroller anliegen können, wenn der Pin auf HIGH geschaltet wird. Demnach wird die Leitung links des Vorwiderstands auf den Pin gegeben:

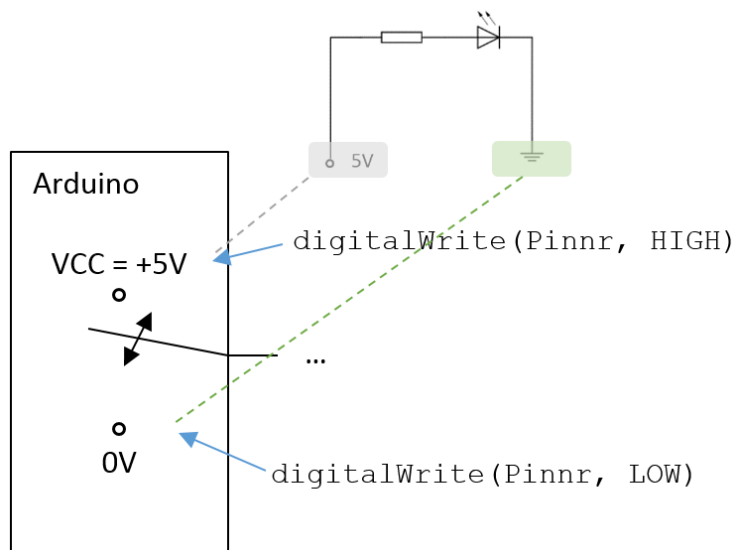


Der Strom fließt über den Pin13 durch den Vorwiderstand in die LED zur Masse.

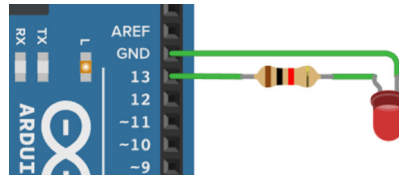
Einen digitalen Pin kann man sich also am besten als programmierbaren Schalter vorstellen. Der Befehl `digitalWrite(Pinnummer, HIGH)` sorgt für +5V am Pin, der Befehl `digitalWrite(Pinnummer, LOW)` sorgt für 0V am Pin, darüber ist der Stromkreis geschlossen.



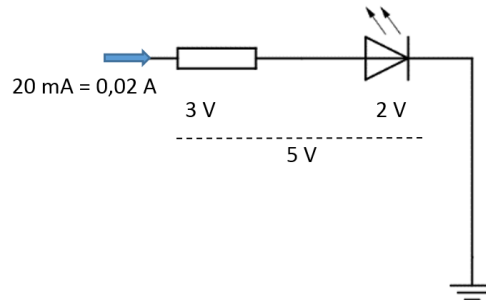
Mit Beschaltung kann man sich die Masse oder VCC 'im Mikrocontroller drin' denken.



Am Arduino verdrahtet achte man besonders auf die LED-Polung, sonst leuchtet nichts. Die eingebaute LED blinkt übrigens parallel, da sie ja immer noch über Pin13 betrieben wird.



Mit Erfahrungs-/Richtwerten für alle möglichen Bauteile werden Sie in der Elektronik häufig konfrontiert werden. Rechentechnisch ergäbe sich – mit glatten Werten großzügig gerechnet, etwas das ebenfalls nicht unüblich ist – folgendes:



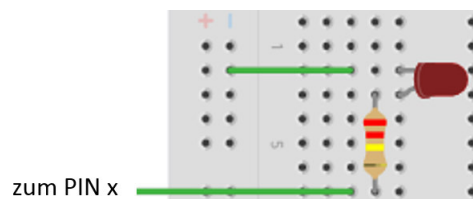
Etwa 2 Volt müssen an der LED abfallen, bleiben also von den 5 V des Arduinos noch 3 V am Widerstand übrig. Damit die LED hell leuchtet sind etwa 20 mA notwendig und dieser Strom fließt – bedingt durch die Reihenschaltung – dann auch durch den zu dimensionierenden Widerstand. Entsprechend gilt per Ohm'schem Gesetz:

$$U = R \cdot I \Leftrightarrow R = \frac{U}{I} = \frac{3V}{0,02A} = 150 \Omega$$

Die 220 Ohm sind also großzügig bemessen¹⁴ (auszuwählen aus den E-Reihen). An Leistung ist man mit Standard ¼-Watt Widerständen gut bedient, denn der Widerstand muss eine Leistung von $P = U \cdot I = 3 V \cdot 0,02 A = 0,06 W < 0,25 W$ aushalten.

Wenn das Programm des vorigen Arbeitsblattes nun läuft blinken sowohl die Board-LED als auch die externe LED gleichzeitig, da ja beide an Pin13 angeschlossen sind.

Auf einem Steckbrett (Breadboard) könnte die Umsetzung so aussehen:

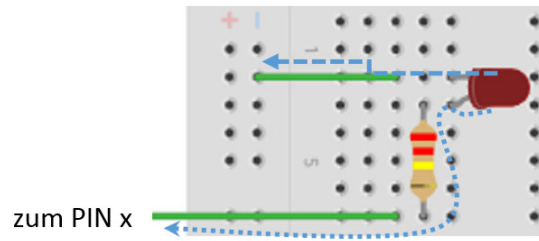


Pin x wäre im obigen Arduino-Beispielbild der Pin13. Man beachte dabei unbedingt die Durchkontaktierung einer Spalte auf dem Steckbrett.

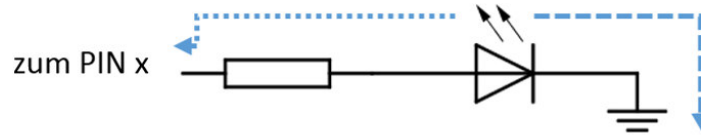


Die Ausrichtung der Bauelemente ist genau zu durchdenken, Schaltbilder können *nicht* eins zu eins von einem Schaltplan ausgehend auf ein Steckbrett übertragen werden. Die Flussrichtung des Stroms und seine Entsprechung auf dem Schaltbild sind im Folgenden nochmals explizit dargestellt:

¹⁴ Üblich sind auch 330 Ohm Widerstände, da der dadurch bedingte etwas geringere Stromfluss immer noch für volles Leuchten ausreicht und der Lebensdauer förderlich ist.



entspricht

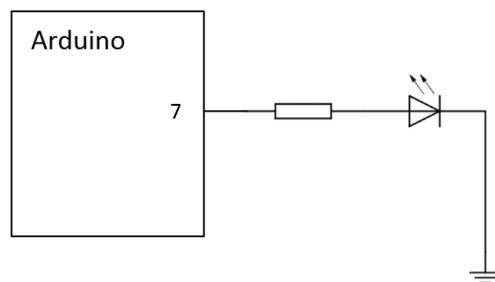


VCC über R steuerbar (high/low) GND fest verschaltet

Übungen

Aufgabe 1:

a) Bauen Sie die Schaltung auf dem Steckbrett auf (R=330 Ohm, rote LED).

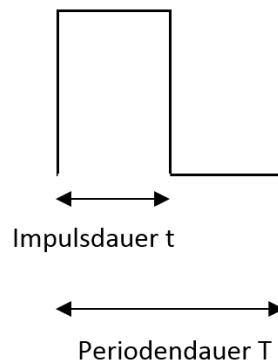


b) Schreiben Sie ein Programm, das die LED im Zwei-Sekunden-Takt blinken lässt, ohne auf das vorige Arbeitsblatt zu schauen¹⁵.

c) Das Verhältnis von Impulsdauer zu Periodendauer (also der HIGH-Anteil pro Periode) wird *Tastgrad*

genannt: $\text{Tastgrad} = \frac{\text{Impulsdauer}}{\text{Periodendauer}} = \frac{t}{T}$.

Im Signalverlauf sieht das so aus:

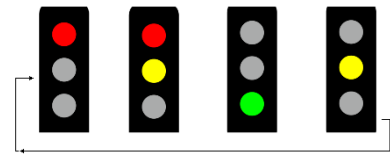


Verändern Sie das vorige Programm so, dass sich der Tastgrad bequem im Quellcode mit möglichst wenigen Änderungen modifizieren lässt.

d) Bauen Sie eine Ampel mittels dreier roter LEDs auf und schreiben Sie ein Programm, dass alle drei abwechselnd blinken lässt.

¹⁵ Ein auch methodisch an Schulen sinnvolles Vorgehen – insbesondere im Programmieranfängerunterricht – um die ganz zentralen Grundlagen was Befehle, Programmstruktur und Bedienung angeht, oft zu üben und dennoch eine gewisse Herausforderung aufzubauen. Einfach erscheinendes ist eben dann doch oftmals gar nicht so leicht, das kommt erst durch häufiges üben. Die Sicherheit einer etwaigen Kontrolllösung ist aber vorhanden und damit gleichzeitig die Rückmeldung, dass die Materie noch nicht gut genug durchdrungen wurde, wenn man auf selbige zurückgreifen muss.

e) Erweitern sie Ihr voriges Programm so, dass die Ampelphasen¹⁶ durchlaufen werden.



Anmerkung:

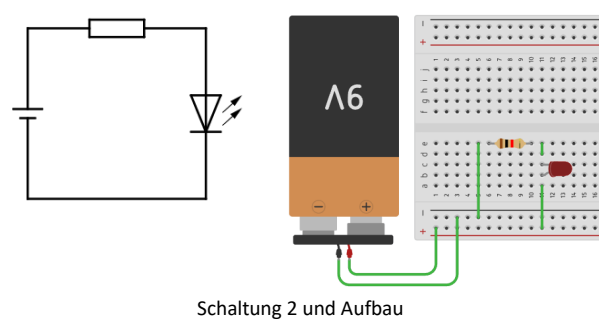
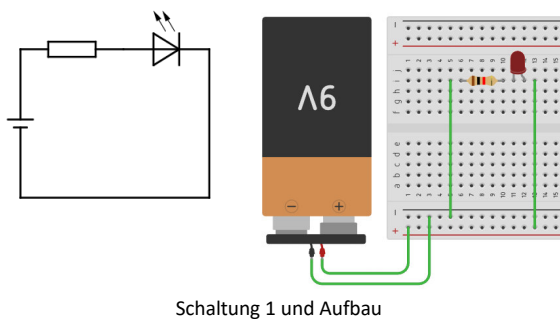
Sollten Sie verschiedenfarbige LEDs verwenden wollen, bedenken Sie, dass diese einen anderen Spannungsabfall haben und daher ggf. auch verschieden große Vorwiderstände benötigen.

LED-Farbe	U-Abfall (genauer)	U-Abfall (Merkwert)
Rot	2,1 V	2 V
Grün	2,1 V	2 V
Gelb	2,2 V	2 V
Blau	2,9 V	3 V
Weiß	3,3 V	3 V

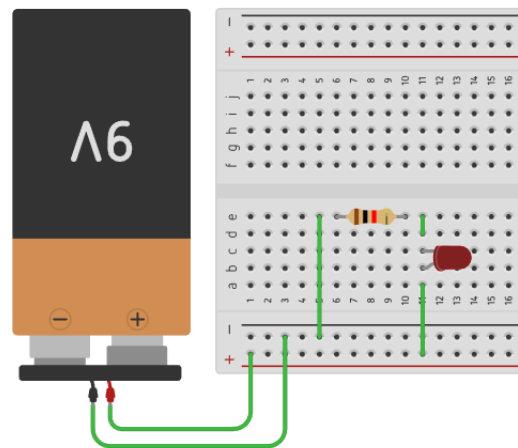
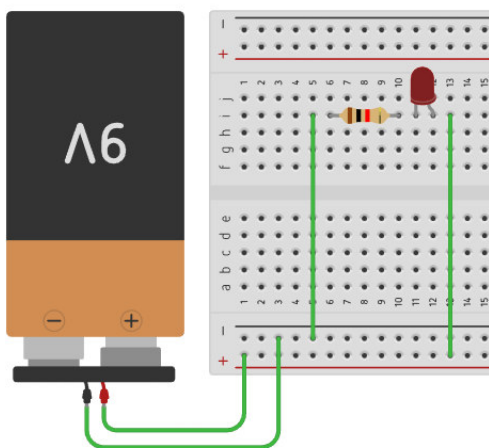
10-20 mA Stromfluss

f)

f.1) Es sind zwei Schaltungen abgebildet und jeweils ein entsprechender Aufbau auf einem Steckbrett. Welche Fehler stecken darin, warum sind diese entstanden – also die Frage nach den fehlerhaften Vorstellungen im Kopf (mentale Modelle) – und welche Korrekturen sind vorzunehmen?

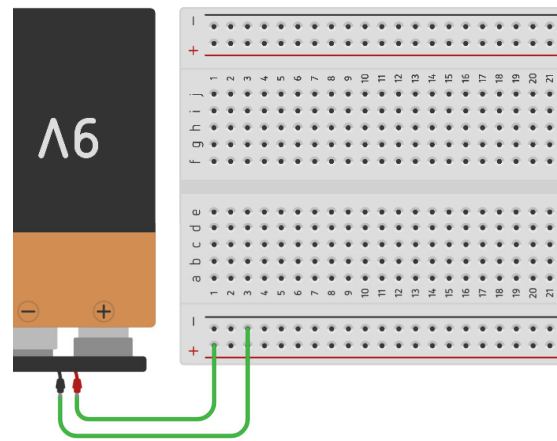
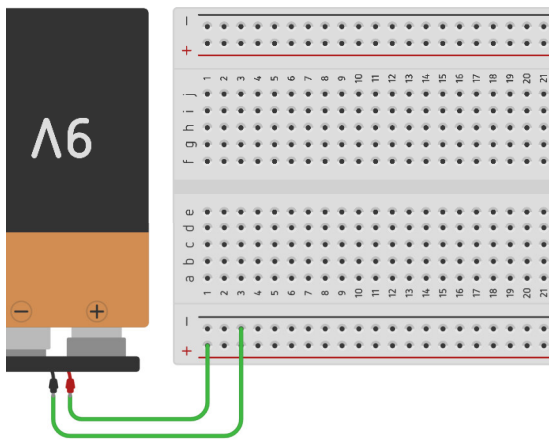
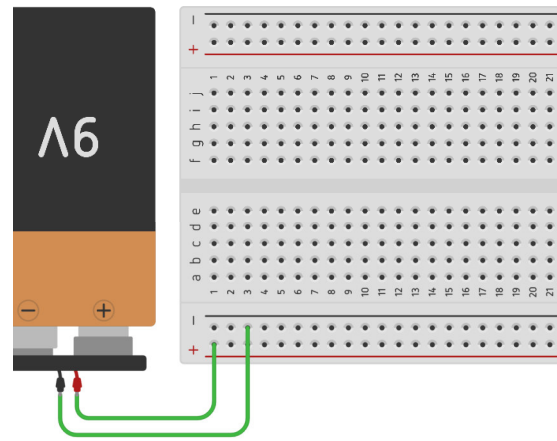
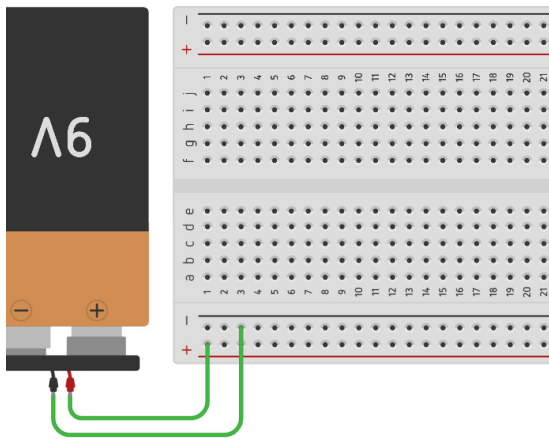


HIER: Einzeichnen der Korrekturen:

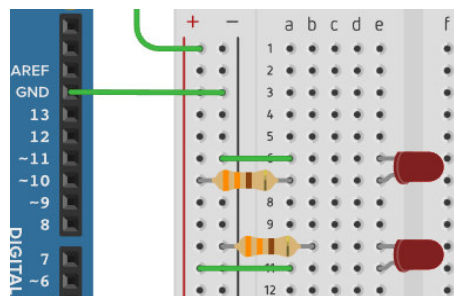
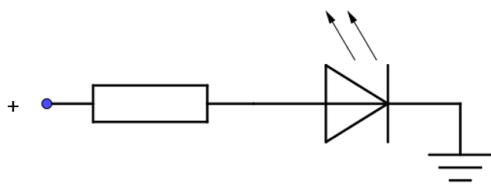


oder auf leerem Brett einzeichnen... (nächste Seite)

¹⁶ <https://commons.wikimedia.org/wiki/File:Cjam-neopixels-uk-traffic.svg>



f.2) Gegeben sei ein Schaltplan und zwei Umsetzungen. Beurteilen Sie diese.



g) Erstellen Sie eine Schaltung sowie ein Programm, welches einen 3-Bit-Binärzähler realisiert. Die Dezimalzahlen 0 bis 7 sollen zyklisch binär durch die LEDs dargestellt werden.

Ausführliche Referenzierungen freier Bilder

Giacomo Alessandrone creator QS:P170,Q100973368
(https://commons.wikimedia.org/wiki/File:Breadboard_full_plus.svg),
<https://creativecommons.org/licenses/by-sa/4.0/legalcode>

Zeroping (https://commons.wikimedia.org/wiki/File:Metal_contacts_within_a_breadboard.jpg),
<https://creativecommons.org/licenses/by/4.0/legalcode>

Breadboard-144dpi.gif: en>User:Waveguy derivative work: McSush (talk)
(https://commons.wikimedia.org/wiki/File:Breadboard_scheme.svg), „Breadboard scheme“,
<https://creativecommons.org/licenses/by-sa/3.0/legalcode>

Andreas B Mundt (<https://commons.wikimedia.org/wiki/File:BreadboardContacts.svg>),
„BreadboardContacts“, <https://creativecommons.org/licenses/by-sa/3.0/legalcode>

Manuel Strehl (original)maix (dark version)Minoa (code optimisation)
(https://commons.wikimedia.org/wiki/File:Traffic_lights_dark_yellow.svg), „Traffic lights dark
yellow“, <https://creativecommons.org/licenses/by-sa/2.5/legalcode>

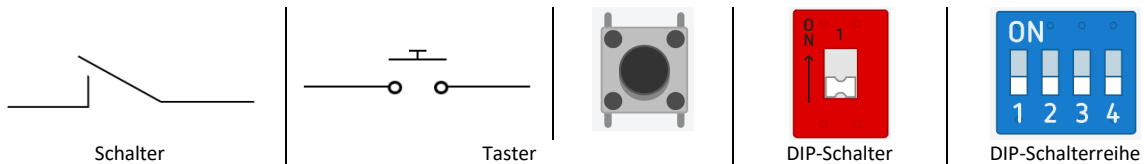
Ampelphasen: gemeinfrei


Arbeitsblatt 4 – Taster und Schalter

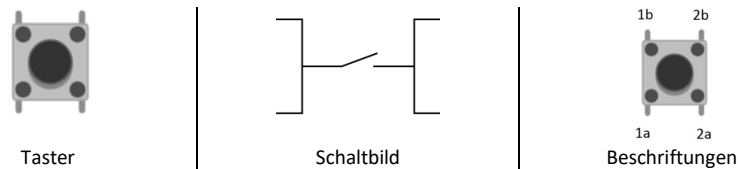
Neben den LEDs als Ausgaben sind Schalter oder Taster die klassischen Vertreter auf der Eingabeseite. Hier gibt es jedoch einiges an Details zu beachten, damit diese auch zuverlässig arbeiten.

Hintergrundwissen

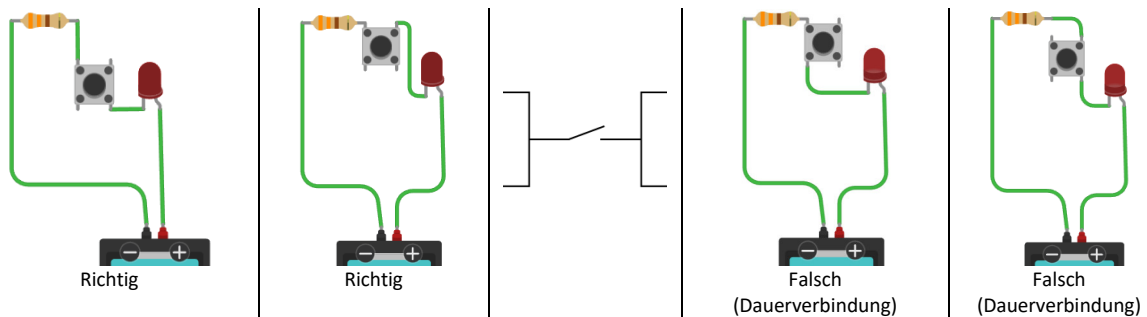
Schalter gibt es in vielfältigen Formen. Ein *Schalter* bleibt in einer Stellung stehen, während ein *Taster* wieder in seine Ausgangsstellung zurückgeht, wenn man ihn loslässt.



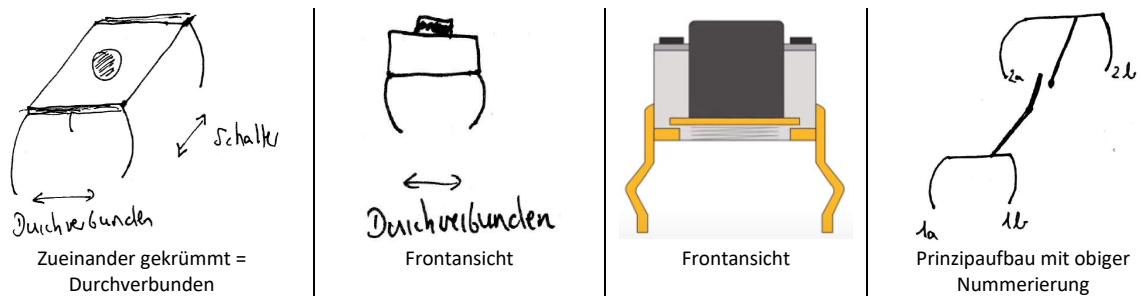
Man achte besonders bei Schaltern oder Tastern mit mehr als zwei Beinen auf die genaue Kontaktierung. Beispielsweise ist die Tasterart  bei Mikrocontrollern und ihren Bausätzen verbreitet. Hier sind vier Beine zu erkennen. Das passende Schaltbild sieht wie folgt aus:



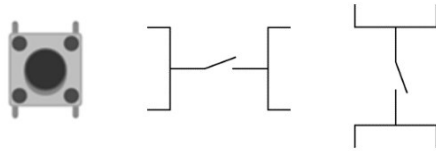
Ein typischer Fehler ist es, die falschen Beine für eine Verdrahtung auszuwählen und statt einer Schalterfunktion eine Durchverbindung zu verdrahten.



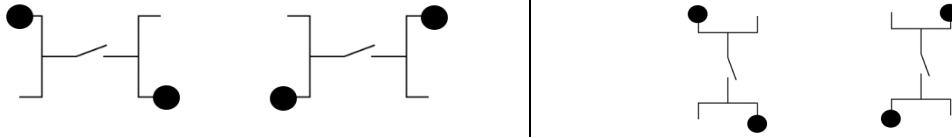
In der Praxis kann man den Schalter vor seiner Verwendung per Messgerät auf Durchgang prüfen oder sich das Bauteil anschauen. Die zueinander hin gekrümmten Beine sind miteinander verbunden:



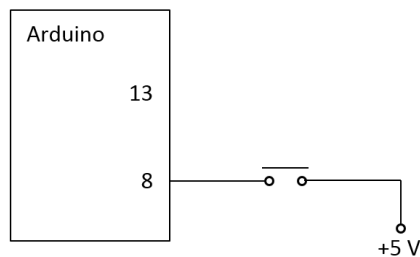
Da die Beschriftungen auf den kleinen Tastern in aller Regel fehlen und falls nicht nachgemessen werden soll gibt es eine praktische Lösung: Durch die interne Verschaltung



ist ein diagonaler Anschluss immer richtig, der Schalter verbindet und trennt, übt also eine Schalterfunktion aus.



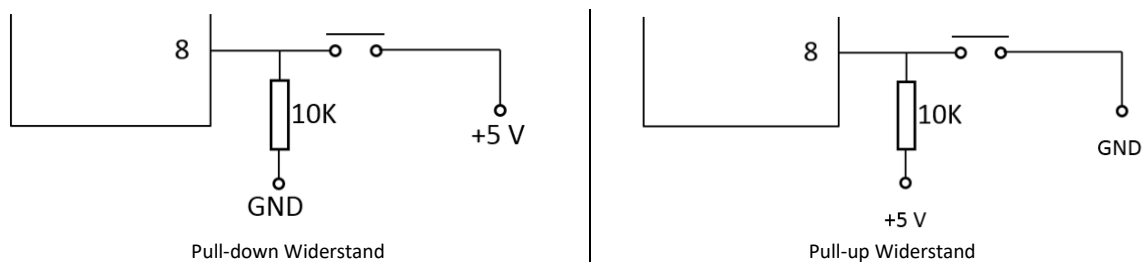
Der Anschluss eines Tasters verläuft nun *nicht* so wie vermutet:



Diese Schaltung kann funktionieren, tut es aber nicht zuverlässig, denn auf Grund äußerer Einflüsse liegt nicht zu allen Zeiten ein eindeutiger Pegel am Eingang an.

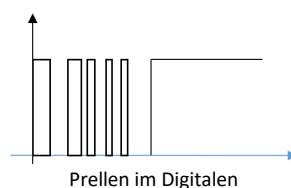
Merke:
Immer für eindeutige Pegel an den Eingängen sorgen.

Die Lösung besteht darin, bei geöffnetem Taster den Eingangspin über einen Widerstand auf Ground (pull-down) oder Betriebsspannung (pull-up) zu legen.



Durch einen pull-up Widerstand liegt allerdings bei offenem Schalter eine 1 (HIGH) am Pin an, also invertierte Logik, Vorsicht im Umgang damit.

Ein weiteres Problem bei Tastern ist das *Prellen*. Bis ein definierter Pegel erreicht wird schwingt er ein paar mal hin und her, was ggf. Probleme machen kann, wenn der Pegel vor dem Einschwingen ausgewertet wird. Daher sollten Taster grundsätzlich entprellt werden, was mit verschiedenen Methoden gelingt.



Übungen

Aufgabe 1:

a) Bauen Sie eine Schaltung mit LED, Vorwiderstand, Schalter und pull-down Widerstand auf einem Steckbrett auf. Erstellen Sie anschließend – ohne Unterlagen zu konsultieren – ein Programm, welches die LED beim Schließen des Schalters einschaltet und beim Öffnen des Schalters ausschaltet.

b) Ergänzen Sie einen Taster mit pull-down Widerstand in der Schaltung und testen Ihr voriges Programm mit diesem Taster statt einem Schalter. Die LED leuchtet also nur solange, wie Sie den Taster gedrückt halten.

c) Gegeben ist folgendes Programm:

```
int tasterPin = 2;           //Taster-Pin 2
int tasterWert = 0;        //Tasterstatus
int zaehler = 0;

void setup() {
  pinMode(tasterPin, INPUT);
}

void loop() {
  tasterWert = digitalRead(tasterPin);
  if(tasterWert == HIGH) {
    zaehler++;
  }
}
```

Es soll zählen, wie oft ein Taster gedrückt wurde. Begründen Sie, warum das so nicht funktioniert und wie man das Problem lösen könnte.

d)

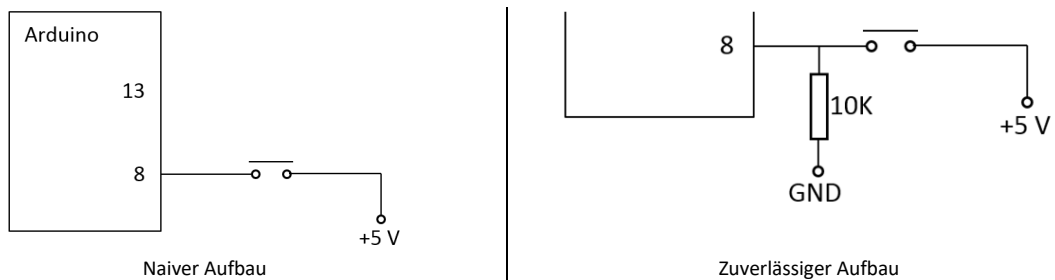
Schreiben Sie ein Programm, das bei Tastendruck den Zustand der LED umschaltet (toggelt). Ein Tastendruck schaltet sie also an, ein weiterer schaltet sie wieder aus.

e) Ein Programm zu schreiben, das blinkende LEDs mittels der delay-Methode realisiert, ist zwar schön direkt, besitzt allerdings einen entscheidenden Nachteil. Die delay-Methode blockiert den Programmablauf. In diesem Zeitintervall – bspw. zwei Sekunden – wird nicht auf Eingaben reagiert, etwa auf einen Tastendruck. Realisieren Sie eine Schaltung und ein Programm, das eine LED im 2-Sekunden-Takt blinken lässt und eine zweite LED immer dann einschaltet, wenn der Taster gedrückt ist.

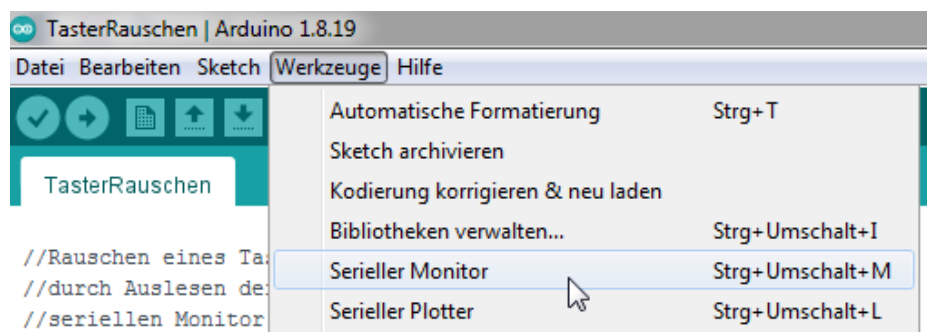
Anmerkung: Die Zeit, die seit dem Start des Sketches verstrichen ist, steht per Methode `millis()` zur Verfügung. Sie wird in Millisekunden angegeben und ist vom Typ `unsigned long`.

f) Rauschen, prellen und serieller Monitor

Der naive Betrieb eines Schalters ohne pull-up/down Widerstand funktioniert nicht zuverlässig.



Ein leistungsfähiges Hilfsmittel zur Ein-/Ausgabe und insbesondere auch zur Fehlersuche ist die Möglichkeit mit dem Arduino seriell zu kommunizieren. Das wird über die USB-Schnittstelle angeboten und im *seriellen Monitor* umgesetzt.

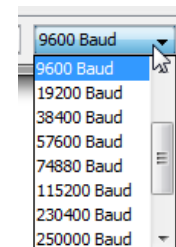


Achtung: Die im Programm festgelegte Übertragungsgeschwindigkeit (**Baudrate**) und die im seriellen Monitor **müssen übereinstimmen**, ansonsten sieht man nichts (häufiger Fehler). Ferner gibt es manchmal Probleme beim hochladen des Programms, wenn der serielle Monitor bereits geöffnet ist.

```
Serial.begin(9600);
```

Dieser Befehl muss im setup stehen und initialisiert den seriellen Monitor mit der angegebenen Baudrate.

Mittels `Serial.println(wert)` können dann Werte auf dem Terminal ausgegeben werden.



f.1) Rauschen

Bauen Sie sowohl den naiven Schalterbetrieb als auch den korrekten auf einem Steckbrett auf, tippen den Code ab und schauen sich das Verhalten des Eingangs über den seriellen Monitor an. Sie sehen dann rauschen in Aktion bzw. das Fehlen dieses unerwünschten Effektes (das Rauschen des Tasters wurde durch den Widerstand demnach erfolgreich eliminiert).

Das folgende Programm liest fortwährend den Status des Tasters an einem Pin ein und gibt ihn auf der Konsole (serieller Monitor) aus:

```
const byte buttonPin = 5;

void setup() {
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}

void loop() {
  Serial.println(digitalRead(buttonPin));
}
```

f.2) Prellen

Neben dem rauschen, welches durch nicht erzwungene eindeutige Eingangspegel verursacht und durch einen pull-up/down Widerstand gelöst werden kann, ist bei Tastern und Schaltern das Phänomen des Prellens (bouncing) zu beobachten. Bis der gewünschte Pegel beibehalten wird, wird ggf. noch ein paar Mal hin und her geschwungen, sozusagen "abgeprallt" (bouncing). Das liegt u.a. an der verbauten Feder im Inneren des Tasters oder am Vibrieren der schließenden Kontakte. Per seriellen Monitor lässt das das Prellen sichtbar machen.



Schließen Sie einen Taster mit Vorwiderstand an den Arduino an, übertragen Sie reflektierend folgenden Code und schauen Sie sich die Ausgabe des seriellen Monitors an. Sie werden feststellen, dass hin und wieder mehr als ein Tastendruck bei einem Tastendruck erkannt wird, der Taster prellt.

```
//prellen eines Tasters über seriellen Monitor sichtbar gemacht

int tasterpin = 8;
int lastButtonState = LOW;
int anzahlTastendrucke = 0;
byte buttonState;

void setup()
{
  Serial.begin(9600); //Serial.begin(115200);
  pinMode(tasterpin, INPUT);
}

void loop()
{
  buttonState = digitalRead(tasterpin);

  if (buttonState != lastButtonState)
  {
    lastButtonState = buttonState;
    if (buttonState == HIGH)
    {
      anzahlTastendrucke++;
      //Serial.println("Taster wurde %u gedrückt\n", anzahlTastendrucke);
      Serial.println(anzahlTastendrucke);
    }
  }
}
```


Die vorliegenden Materialien wurde im Rahmen des Projektes FAIBLE.nrw vom Arbeitsbereich Didaktik der Informatik der WWU-Münster erstellt und sind unter der (CC BY 4.0) - Lizenz veröffentlicht. Ausdrücklich ausgenommen von dieser Lizenz sind alle Logos. Weiterhin kann die Lizenz einzelner verwendeter Materialien, wie gekennzeichnet, abweichen. Nicht gekennzeichnete Bilder sind entweder gemeinfrei oder selbst erstellt und stehen unter der Lizenz des Gesamtwerkes (CC BY 4.0). Sonderregelung für die Verwendung im Bildungskontext:

Die CC BY 4.0-Lizenz verlangt die Namensnennung bei der Übernahme von Materialien. Da dies den gewünschten Anwendungsfall erschweren kann, genügt dem Projekt FAIBLE.nrw bei der Verwendung in informatikdidaktischen Kontexten (Hochschule, Weiterbildung etc.) ein Verweis auf das Gesamtwerk anstelle der aufwändigeren Einzelangaben nach der TULLU-Regel. In allen anderen Kontexten gilt diese Sonderregel nicht.

Das Werk ist Online unter <https://www.orca.nrw/> verfügbar.



<https://creativecommons.org/licenses/by/4.0/deed.de>

FAIBLE.nrw

ORCA.nrw
Das Landesportal für
Studium und Lehre.

Beteiligte Hochschulen:



RWTH-Aachen



Westfälische Wilhelms-
Universität Münster



Universität Duisburg-Essen



Universität Bonn



Universität Paderborn



Technische Universität Dresden



Carl von Ossietzky
Universität Oldenburg

Ein Kooperationsvorhaben empfohlen durch die:

 **DIGITALE
HOCHSCHULE
NRW**

INNOVATION DURCH KOOPERATION

gefördert durch:

Ministerium für
Kultur und Wissenschaft
des Landes Nordrhein-Westfalen

