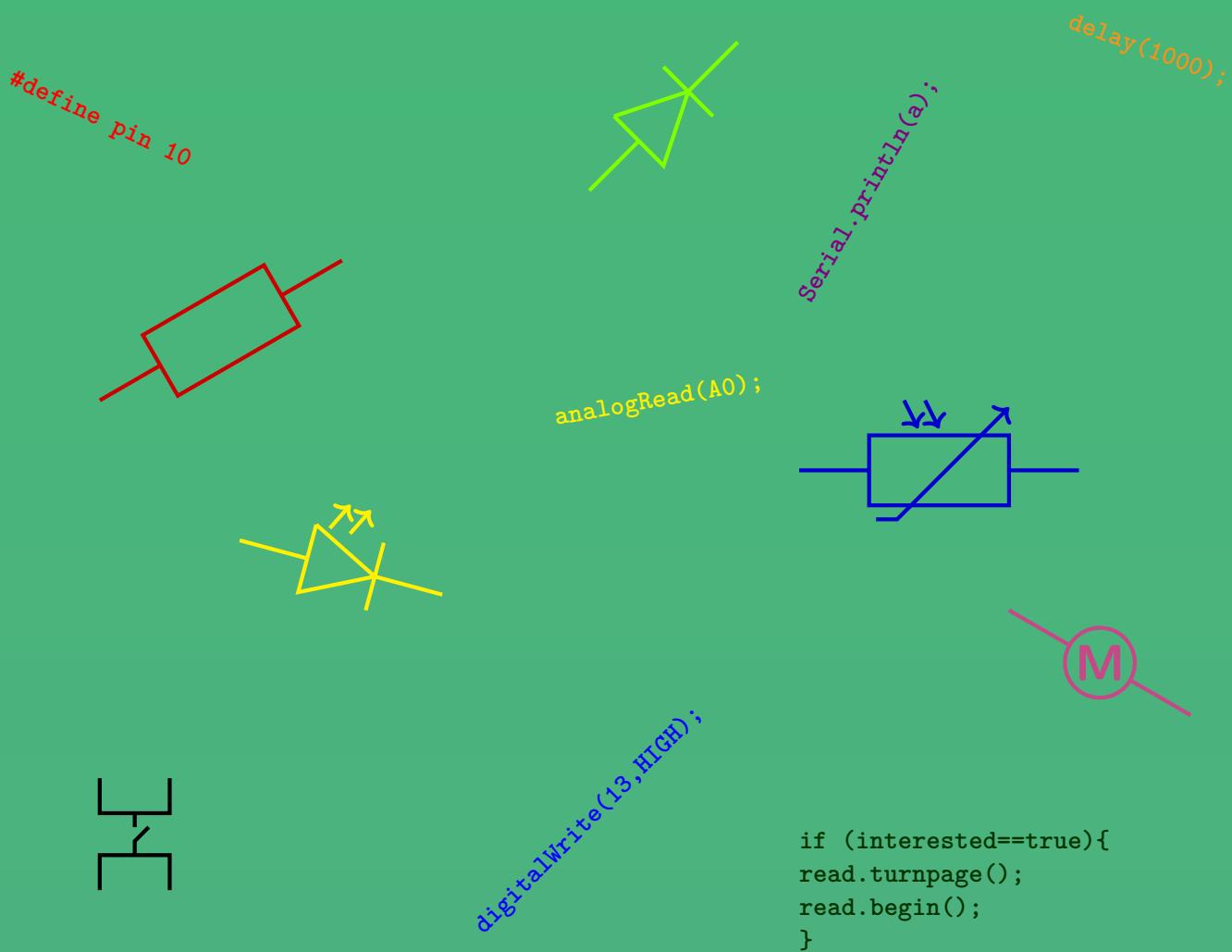




void setup{...}



Skript zur Einführung des Arduino im Wahlpflichtfach „Informatik/Physik“ (MINT-Profil)
im Schuljahr 2018/19

Sebastian Voß

E-Mail: sebastian@el-voss.de

Gymnasium Marianum

Herzog Arenberg Str. 65
49716 Meppen

Alle Bilder wurden vom Author selbst erstellt. Zeichnungen wurden mit *TikZ* angefertigt. Schaltpläne wurden mit *QElectroTech* erstellt. Dieses Skript wurde mit *LAT_EX* geschrieben. Screenshots wurden mit *Shutter* (unter Linux) bzw. mit dem *Snipping Tool* (Windows) erstellt.

Alle Programme und Schaltungen in diesem Skript wurden sorgfältig erstellt und geprüft. Der Author kann nicht für Schäden verantwortlich gemacht werden, die bei der Verwendung dieses Skripts entstehen.

Internet-Links waren zum Zeitpunkt der Erstellung dieses Skripts ebenfalls funktionsfähig und führten zu sinnvollen Inhalten. Es kann jedoch nicht ausgeschlossen werden, dass sie inzwischen zu ganz anderen Inhalten führen.

Didaktisch-methodische Bemerkungen zu diesem Skript befinden sich im Anhang. Fortbildungsaufgaben können Sie an die oben angegebene E-Mail-Adresse richten.

Dieses Skript ist unter den Creative-Commons-Bedingungen BY-NC-SA 4.0 veröffentlicht.
<https://creativecommons.org/licenses/by-nc-sa/4.0/deed.de>

Zuletzt bearbeitet am 9. August 2019.

Inhaltsverzeichnis

1. Einleitung	1
1.1. Weitere Informationen: Bücher und Internet	2
2. Eine kurze Einführung in die Programmierumgebung mBlock	3
3. Einführung in die Welt des Arduino: Digitale Ausgänge und Eingänge	5
3.1. Der Aufbau des Arduino UNO	6
3.2. Vorbereitung von <i>mBlock</i>	7
3.3. Digitale Ausgänge steuern	8
3.4. Aufbau von Schaltungen auf der Steckplatine	9
3.5. Widerstand, Spannung und elektrische Stromstärke berechnen	10
3.6. Widerstandsringe ablesen	13
3.6.1. Einfache Verwendung einer 7-Segment-Anzeige	14
3.7. Das elektrische Potential an digitalen Eingängen	15
3.7.1. Eine Analogie für das elektrische Potential	15
3.7.2. Verwendung eines Tasters	17
3.8. Vermischte Übungen	19
4. Konzepte der Informatik: Bausteine von Algorithmen	22
4.1. Programme mit Variablen strukturieren	23
4.2. Lauflichter effizient programmieren	24
4.3. Zufällige Ereignisse programmieren	25
4.4. Kommunikation mit dem Arduino: Der serielle Monitor	25
4.4.1. Exkurs: Zufallszahlen von Mikrocontrollern/-prozessoren	26
4.5. Kurze Einführung in die Codierung von Zahlen und Zeichen	27
4.5.1. Ein Einblick ins Binärsystem	27
4.5.2. Ein Einblick in die ASCII-Tabelle	28
4.6. Programme mit Struktogrammen darstellen	29
4.7. Vermischte Übungen	31
5. Analoge Ausgänge und Eingänge	33
5.1. Pulsweitenmodulation (PWM)	34
5.1.1. RGB-Farben aus Hexadezimalzahlen erzeugen	36
5.2. Spannung messen an analogen Eingängen	38
5.2.1. Bedingungen durch logische Operationen verfeinern	39

5.3. Die Verwendung eines Potentiometers (Drehreglers)	41
5.3.1. Die Verwendung eines Potentiometers ohne Mikrocontroller	42
5.4. Helligkeit messen	44
5.5. Temperatur messen	48
5.6. Exkurs: Schaltungen mit Transistoren vereinfachen	50
5.7. Das EVA-Prinzip	52
5.8. Vermischte Übungen	55
6. Erweiterung des Werkzeugkastens: Bauteilkunde	59
6.1. Neigungsschalter	60
6.2. Bewegungsmelder	61
6.3. Ultraschallsensor	62
6.4. Liquid Crystal Display (LCD)	64
6.5. Servo	66
6.6. Joystick	68
6.7. Infrarot-Fernbedienung	70
6.8. Temperatur- und Luftfeuchtigkeitssensor DHT-11	72
7. Elektromotoren steuern	73
7.1. Elektromotor und Diode	74
7.2. Steuerung mit Transistor	75
7.3. Steuerung mit Relais	77
7.4. Steuerung mit dem Motortreiber-IC L293D (inkl. Drehrichtung)	80
8. Konzepte der Informatik II	84
8.1. Funktionen	85
8.2. Automaten	87
A. Anhang	92
A.1. Ein kurzer Überblick - die Funktionseinheiten des Arduino Uno	92
A.2. Das Elegoo Starter-Kit	94
A.3. Installation der integrierten Entwicklungsumgebung (IDE) für den Arduino	95
A.4. Mit der Arduino IDE ein Programm auf den Arduino Uno übertragen	95
B. Didaktisch-Methodische Bemerkungen	98
B.1. Zielgruppe	99
B.2. Organisatorisches: Arduino Starter Kits, Raum, Zusammenarbeit	99
B.3. Zum Aufbau dieses Skripts	100
B.3.1. Aufbau der Kapitel	100
B.3.2. Erklärung zu Links, Symbolen und eingebetteten Arbeitsblättern	101
B.4. Erfahrungsbericht	102
B.5. Ausblick	102



B.6. Anpassung / Weiterentwicklung 103

1. Einleitung

Anfang der 2000er Jahre sollte der Professor Massimo Banzi seinen Studenten beibringen, wie man interaktive elektrische Schaltungen für künstlerische Projekte erstellt. Leider erforderten die damals vorhandenen Mikrocontroller einiges an Hintergrundwissen, bevor man irgend etwas mit ihnen anfangen konnte. Professor Banzi hatte dieses Wissen - er mochte seinen Studenten, die ein künstlerisches Designstudium gewählt hatten, jedoch kein Studium zum Elektroingenieur zumuten, ehe sie fähig wären, künstlerische elektronische Projekte umzusetzen.

So kam es, dass Massimo Banzi und David Cuartielles im Jahr 2005 den ersten Arduino entwickelten. Dieser sollte einfach zu handhaben, günstig anzuschaffen und - eine ziemlich neue Idee für Hardware - sein Aufbau sollte frei zugänglich sein, sodass er auch von anderen nachgebaut werden konnte. Natürlich sollte auch die Entwicklungsumgebung, mit der sich der Arduino programmieren lässt, frei verfügbar sein. Diese drei Eigenschaften des Arduino führten innerhalb weniger Jahre zu einer unglaublichen Verbreitung des Mikrocontrollers, die heute nicht nur Studenten und Universitäten betrifft, sondern auch Bastler, Künstler und Schulen weltweit. Elektronik und Programmierung wurde von einer kleinen Nische für Nerds zu einem allgemein verfügbaren Werkzeug, mit dem jeder, der sich ein wenig mit dem Thema beschäftigt, seine Kreativität auf eine neue Weise ausleben kann.

Diese Sichtweise auf den Arduino ist ein wichtiges Ziel dieses Kurses. Es geht nicht nur um den Ausbau eines theoretischen Weltverständnisses, das sonst häufig im Zentrum steht. Es geht um die Erweiterung der praxisbezogenen Fähigkeiten und Fertigkeiten, mit denen wir unsere Umwelt selbst gestalten, erweitern und reparieren können. Jeder neue Lerninhalt soll im Kontext der Fragestellung „Wozu ist das nützlich?“ oder „Was kann man damit anfangen?“ präsentiert werden. Diese Grundhaltung knüpft an die sogenannte *Maker*-Bewegung an - eine ständig größer werdende Gemeinschaft von Menschen, die neue Werkzeuge wie den Arduino, 3D-Drucker und Laser-Cutter nutzen, um selbst Dinge zu erschaffen, Dinge mit selbst ausgedachten Features zu erweitern (zu hacken) oder Dinge zu reparieren, die ganz im Sinne des Herstellers viel zu früh kaputt gegangen sind. Dabei gab es „Maker“, also Bastler, schon immer, allerdings öffnen die neuen Werkzeuge und insbesondere der Arduino Möglichkeiten, die es früher nicht gab. Dabei gehört es zum Konzept des Bastelns, dass man nicht immer alle Dinge bis ins Detail versteht, sondern sich traut, Dinge auszuprobieren und nicht zu enttäuscht zu sein, wenn diese Versuche daneben gehen. Innerhalb eines schulischen Kurses, in dem wir nicht mit selbst bezahlten Materialien experimentieren, sind diesem Ansatz Grenzen gesetzt, allerdings werden wir so weit wie möglich auch frei experimentieren. Natürlich soll das Verständnis dessen, was man da gebaut hat, nicht auf der Strecke bleiben, denn erst dieses Verständnis ermöglicht es systematisch, Fehler zu beseitigen oder neue Funktionen in vorhandene Schaltungen zu integrieren.

1.1. Weitere Informationen: Bücher und Internet

Wenn man sein Interesse und seine Freude am Basteln entdeckt hat, will man häufig weitere Informationen haben, weitere Projekte begutachten oder weitere Ideen finden, die die eigene Kreativität wiederum beflügeln. Die große Arduino-Gemeinschaft ist im Internet sehr aktiv und stellt zahlreiche Informationen bereit. Die im folgenden genannten Quellen boten mir selbst einen guten Einstieg und ich ziehe sie bei Fragen immer noch gerne heran.

Die wichtigste Anlaufstelle ist die Projekthomepage zum Arduino (www.arduino.cc). Dort finden sich nicht nur viele Informationen rund um den Arduino, sondern auch kreative Projektvorstellungen. Weitere Projekte mit fertigen Anleitungen zum Nachbauen finden sich auf Instructables (<http://www.instructables.com/>). Auch Youtube stellt zahlreiche Videos von stolzen Makern bereit, die ihr Projekt vorstellen. Ein Beispiel für einen professionell betriebenen Kanal zu Elektronik ist *GreatScott!* (<https://www.youtube.com/user/greatscottlab>). Ein weiterer professionell betriebener Kanal mit vielen guten Bastelanleitungen ist *bitluni's lab* (<https://www.youtube.com/user/bitlunislab>). Viele gute deutschsprachige Anleitungen zum Arduino findet man bei *Makerblog.at - Arduino & Co* (<https://www.youtube.com/user/makerblogAT>). Nicht auf Youtube, sondern auf ihrer eigenen Homepage bieten die Leute von *Funduino* (<https://funduino.de/>) ebenfalls gute Anleitungen für Einsteiger an.

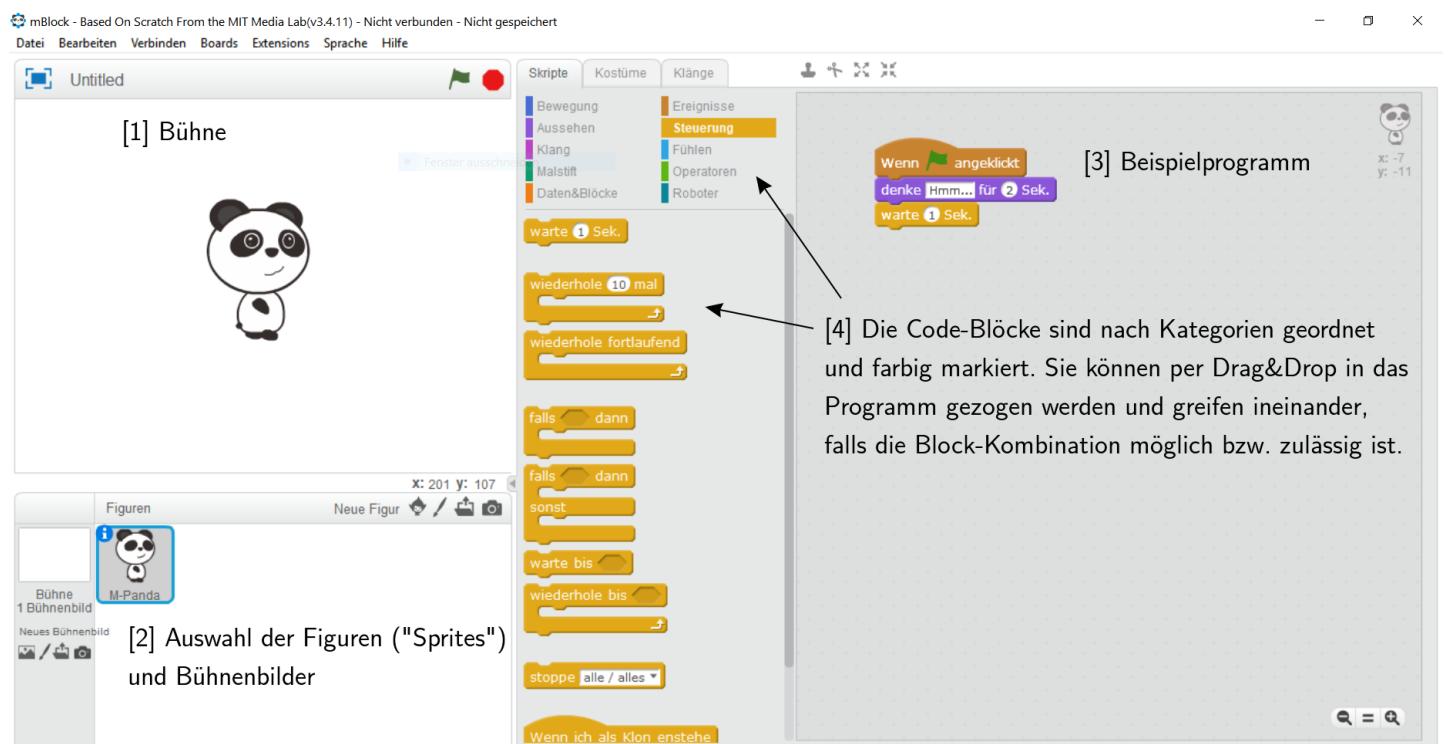
Wer ein Buch bevorzugt, findet in dem Buch *Arduino Workshops* von John Boxall einen sehr guten Ratgeber, der ganz ähnlich zu diesem Skript schrittweise und anhand von konkreten Projekten in die Welt des Arduino einführt.

Ein umfangreiches und ebenfalls praxisbezogenes Nachschlagewerk ist das Buch *Arduino Praxiseinstieg* von Thomas Brühlmann. Es ist gut sortiert und bietet eine schnelle Hilfe für die meisten Fragen rund um den Arduino.

Alle genannten Quellen haben allerdings gemeinsam, dass sie mit der textbasierten Arduino - Programmierumgebung arbeiten und daher einen etwas komplexeren Einstieg bieten als dieses Skript.

2. Eine kurze Einführung in die Programmierumgebung mBlock

Das Programm **mBlock** ist eine graphische Programmierumgebung, die auf Scratch beruht und es sehr einfach macht, erste Programmiererfahrungen zu sammeln. Abbildung 2.1 zeigt eine grobe Übersicht über die einzelnen Elemente des Programms, jedoch lernt man sie am besten kennen, indem man einfach drauf loslegt und ausprobiert, was sich damit anstellen lässt.



B 2.1. Übersicht über die Programmierumgebung *mBlock*.

Für die ersten Programmiererfahrungen machen wir noch nichts mit dem Arduino, sondern kümmern uns nur um die in mBlock eingebaute Bühne.

Ziel: Der Panda soll auf der Bühne folgendes machen, nachdem auf die grüne Fahne (→ Ereignisse) geklickt wurde:

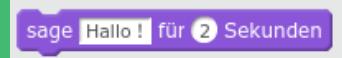
- Der Panda läuft von links nach rechts. (→ Bewegung)
- Der Panda sagt für 1 Sekunde „Hallo!“. (→ Aussehen)

- Der Panda läuft von rechts nach links.
- Der Panda wartet eine Sekunde (und macht dabei nichts).
- Der ganze Vorgang wiederholt sich fortlaufend.

Für Schnelle und Kreative: Erweitere dein Programm, indem du eine andere Bühne oder ein anderes Kostüm auswählst. Spiele mit den Befehlen herum, um dein Programm individuell anzupassen.

Programm, Befehl und Argument

Ein Programm besteht aus einer Folge von Anweisungen oder Befehlen. Man spricht auch von Algorithmen: Ein Algorithmus ist eine eindeutige Handlungsvorschrift zur Lösung eines Problems, die aus endlich vielen Anweisungen besteht (s. [Wikpedia](#)).



B 2.2. Befehl mit Textargument und Zahlargument.

Ein Befehl kann ein oder mehrere Argumente haben, die wiederum einen unterschiedlichen Typ haben können (z. B. Text oder (ganze) Zahl).

Aufgabe 1: Gehübungen

Beschreibe jeweils, was der Panda macht, wenn das abgebildete Programm läuft. Streiche, wenn möglich, alle Befehle, die man nicht benötigt.



(a)



(b)

3. Einführung in die Welt des Arduino: Digitale Ausgänge und Eingänge

Der Arduino ist ein Mikrocontroller, der vom italienischen Professor Massimo Banzi entwickelt wurde, damit seine Studenten im Bereich *Design* eine einfach zugängliche Möglichkeit fanden, Elektronik für künstlerische Projekte zu nutzen. Die vorgenommenen Vereinfachungen in der Handhabung von Mikrocontrollern gefielen jedoch nicht nur den Studenten von Massimo Banzi, sondern auch zahlreichen Studenten anderer Fachrichtungen, Schülern, Hobby-Elektronikern und sogar Fachleuten in der Industrie, sodass der Arduino rasch eine weltweite Verbreitung fand.

In diesem Kapitel lernst du...

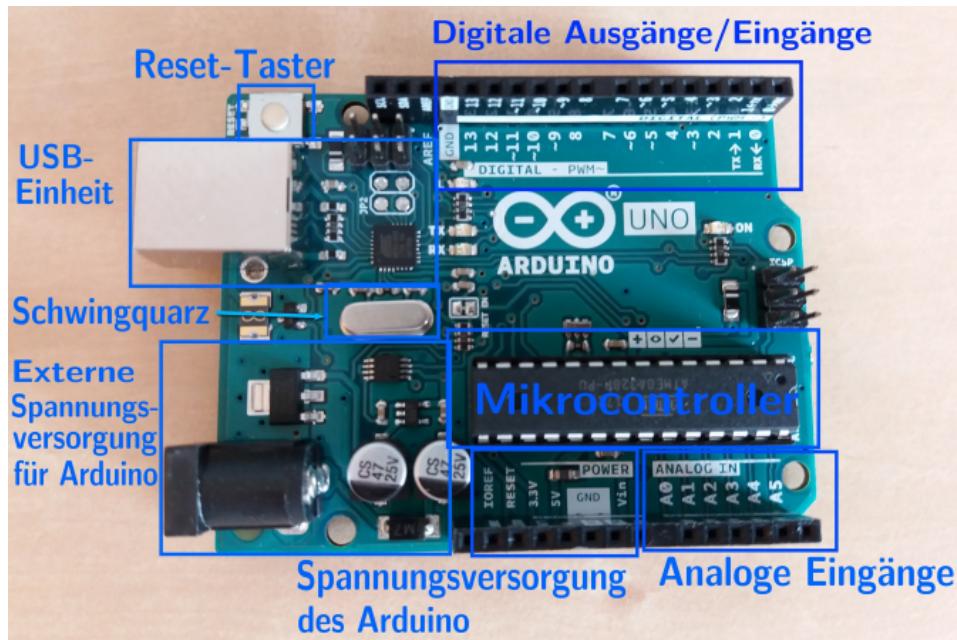
- ... wie der Arduino aufgebaut ist,
- ... wie man den Arduino mit dem PC verbindet und mit mBlock programmiert,
- ... die digitalen Pins als Ausgänge zu benutzen, um eine LED zu steuern,
- ... Schaltungen auf dem Steckbrett aufzubauen,
- ... Widerstand, Spannung und Stromstärke im Stromkreis zu berechnen,
- ... Widerstandsringe abzulesen, um die Größe des Widerstands zu bestimmen,
- ... eine RGB-LED und eine 7-Segmentanzeige zu steuern,
- ... das elektrische Potential an einem digitalen Pin einzulesen und ...
- ... mit Hilfe des elektrischen Potentials den Zustand eines Tasters abzufragen.

Projekte in diesem Kapitel:

> Ampel	9	> Raketencountdown	14
> Augentestgerät	9	> Fußgängerampel	18
> Farbenspektakel mit RGB-LED ..	12	> Juke-Box	18

3.1. Der Aufbau des Arduino UNO

Mit der Zeit entwickelten sich zahlreiche andere Modelle des Arduino, die kleiner oder größer waren, über mehr oder weniger Anschlüsse verfügten, schneller oder langsamer waren usw. Das Standardmodell ist heute der Arduino Uno, den auch wir verwenden.

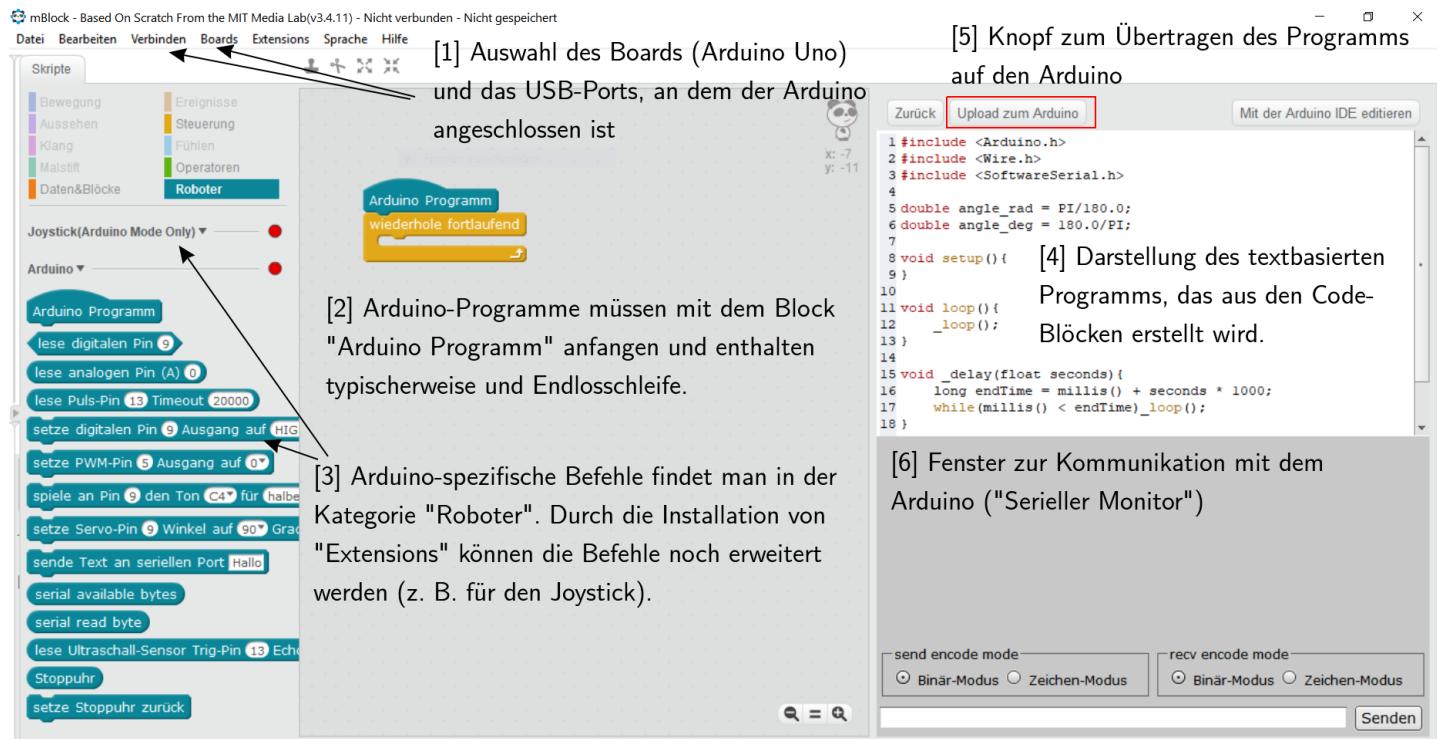


B 3.1. Die wichtigsten Komponenten eines Arduino Uno.

Abbildung 3.1 zeigt die wichtigsten Komponenten des Arduino Uno. Eine genauere Beschreibung dieser Funktionseinheiten und ihrer spezifischen Eigenschaften findet sich in Anhang A.1. Wichtig sind an dieser Stelle vor allem folgende Punkte:

- Über den USB-Anschluss und das mitgelieferte Kabel lässt sich der Arduino mit dem PC verbinden und programmieren.
- Das Programm läuft nach dem Übertragen auf dem eigentlichen Mikrocontroller, dem langen schwarzen Ding in der Mitte. Der ganze Rest auf dieser kleinen Platine dient der einfacheren Handhabung des Mikrocontrollers.
- An den Seiten befinden sich die Pin-Leisten, an die sich zum Beispiel LEDs anschließen lassen. Die Pins sind durchnummieriert, sodass sie im Programm angesprochen werden können. *GND* steht für „Ground“ oder den Minus-Kontakt. *5V* steht für den Plus-Kontakt und gibt an, dass dort stets eine Spannung von 5V anliegt, wenn der Arduino über USB oder Batterie mit Strom versorgt wird. Die durchnummerierten Digitalpins können durch das Programm ebenfalls auf 5V gesetzt werden (*HIGH*), aber auch auf 0V, sodass kein Strom fließt (*LOW*).

3.2. Vorbereitung von *mBlock*



B 3.2. Übersicht über die Programmierumgebung *mBlock* im Arduino-Modus.

Die Programmierumgebung *mBlock* wurde im Wesentlichen dafür erstellt, dass man die firmeneigenen Roboter namens *mBot* programmieren kann. Diese basieren auf einem Arduino, der die angeschlossenen Motoren und Sensoren steuert und per *mBlock* programmiert wird. Gleichzeitig ermöglicht es *mBlock* auch, den Arduino als eigenständigen Mikrocontroller zu programmieren. Dies ist der Modus, den wir benutzen werden. Man erreicht diesen Arduino-Modus unter **Bearbeiten** → **Arduino-Modus**.

Als erstes muss der USB-Port ausgewählt werden, an dem der Arduino angeschlossen ist (unter **Verbinden**) und danach muss der Arduino Uno als Board ausgewählt werden (unter **Boards**). Damit sind alle Einstellungen vorgenommen, um den Arduino Uno mit *mBlock* programmieren zu können.

3.3. Digitale Ausgänge steuern

Ziel: Es soll das erste Testprogramm auf den Arduino übertragen werden, mit dem man üblicherweise überprüft, ob der Arduino (oder ein anderer Mikrocontroller) richtig funktioniert. Dazu soll die bordeigene LED zum Blinken gebracht werden:

- Stelle LED an.
- Warte eine Sekunde.
- Stelle LED aus.
- Warte eine Sekunde.
- Wiederhole fortlaufend.

Hinweise:

Arduino-Programme müssen immer mit dem entsprechenden Startblock für ein Arduino-Programm beginnen (siehe Abbildung rechts.)

Arduino Programm

Die bordeigene LED ist mit dem (digitalen) Pin Nummer 13 verbunden und kann darüber gesteuert werden.

setze digitalen Pin 0 Ausgang auf HIGH▼

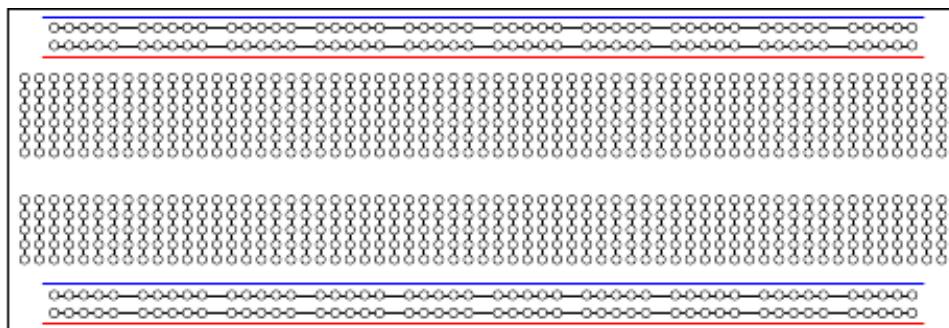
Zur Steuerung von digitalen Pins nutzt man den rechts abgebildeten Block.

Mit dem Knopf **Upload zum Arduino** lässt sich das fertige Programm auf den Arduino übertragen. Wenn alles klappt, sollte nun die bordeigene LED blinken.

Aufgabe 1: Wir nutzen in den ersten Kapiteln sehr häufig LEDs, weil sich die Grundlagen mit ihnen einfach erarbeiten lassen, aber auch weil sie eine enorme Bedeutung in der heutigen Welt haben. Notiere dir zur Verdeutlichung eine Woche lang alle Geräte, die dir begegnen oder die dir einfallen, in denen LEDs verbaut sind.

3.4. Aufbau von Schaltungen auf der Steckplatine

In der Regel braucht man für interessante Geräte zusätzliche *Hardware* (Sensoren, Motoren, ...), die am Arduino angeschlossen wird. Bevor diese fest verlötet werden, nutzt man normalerweise Steckverbindungen bzw. baut die Schaltung auf einem kleinen Steckbrett auf, auf dem man die Verbindungen schnell wieder lösen kann, falls nötig. Steckbretter sind aus dem Physikunterricht bekannt. Abbildung 3.3 zeigt, welche Kontakte auf dem Steckbrett miteinander verbunden sind.



B 3.3. Die Steckverbindungen sind außen in Längsrichtung und innen in Querrichtung miteinander verbunden.

Ziel: Eine externe LED an Pin 13 soll zum Leuchten gebracht werden. Diese kann genutzt werden, um das Blinken einer Alarmanlagen-LED zu simulieren.

Hinweise:

Es kann das Programm aus dem vorherigen Abschnitt wieder verwendet werden. Allerdings gibt es ein Problem: Wenn der digitale Pin auf HIGH gesetzt wird, bedeutet das, dass er eine Spannung von 5V gegenüber GND ausgibt. Die LED verträgt jedoch nur (je nach Farbe) gut 2 V. Daher ist die LED mit einem Vorwiderstand von 330Ω verbunden.

Die LED ist ein sogenanntes gepoltes Bauteil. Das heißt, dass einer der Kontaktstifte der LED an den Plus-Kontakt angeschlossen werden *muss* und der andere an den Minus-Kontakt angeschlossen werden *muss*.

Projekt 1: Ampel

Baue und programmiere eine Ampelschaltung!

Für Schnelle: Erweitere die Ampel um einen Nachtmodus.

Projekt 2: Augentestgerät

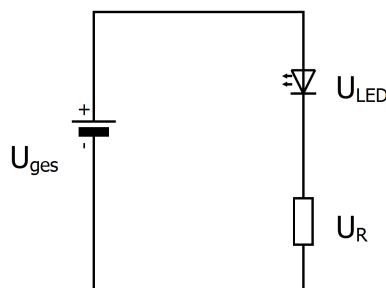
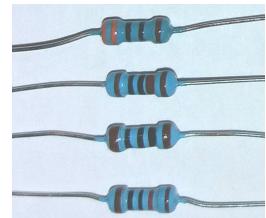
Moderne Fernseher nutzen meist eine Bildwiederholungsrate von 144 Hertz; das bedeutet, es werden 144 Bilder pro Sekunde eingespielt. Finde mithilfe des Blink-Programms heraus, ab welchem Blinkintervall du kein Flackern mehr wahrnimmst und berechne, wie viele „Bilder“ pro Sekunde sich daraus ergeben. Vergleiche diesen Wert mit der Bildwiederholungsrate im Fernsehen.

Idee: Frick, Fritsch und Trick (2015): *Einführung in Mikrocontroller - Der Arduino als Steuerzentrale*, Bad Saulgau

3.5. Widerstand, Spannung und elektrische Stromstärke berechnen

Im vorherigen Abschnitt war die Größe des Vorwiderstands mit $330\ \Omega$ vorgegeben. In unserem Bausatz finden sich jedoch viele weitere Widerstände, die teilweise größer und teilweise kleiner sind.

Vorüberlegung: Wird die LED heller oder dunkler leuchten, wenn man den Vorwiderstand vergrößert? Begründe deine Vermutung.



B 3.4. Reihenschaltung von LED und Vorwiderstand an einer Spannungsquelle.

Nach den Vorüberlegungen ist klar, dass ein größerer Widerstand für die LED kein Problem darstellt. Aber wie sieht es mit einem kleineren Widerstand aus? Der Widerstand muss schließlich in jedem Fall verhindern, dass die LED stärker belastet wird als sie aushält...

Frage: Wie groß muss der Vorwiderstand einer LED mindestens sein, damit sie nicht durchbrennt?

Hinweise:

- Wenn ein Digitalpin auf HIGH gesetzt wird, dann gibt er eine Spannung von 5V gegenüber GND aus.
- Durch eine LED darf höchstens ein Strom von 20 mA fließen.
- Je nach Farbe halten LEDs eine andere maximale Spannung aus:

Farbe	rot	gelb/grün	blau/weiß
U_{LED}	1,6 V - 2,2 V	1,9 V - 2,5 V	2,7 V - 3,5 V

Widerstand, Spannung und Stromstärke

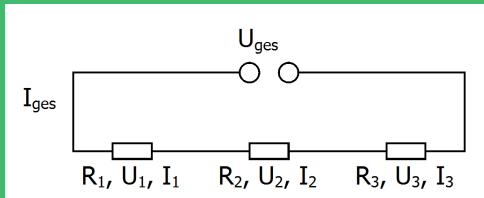
Der Widerstand R ist definiert als das Verhältnis von Spannung U zu Stromstärke I :

$$R = \frac{U}{I}$$

$$\triangle \frac{U}{R} I$$

Ein Widerstand heißt *ohmscher Widerstand*, wenn das Verhältnis $\frac{U}{I}$ stets gleich groß ist (also wenn R unabhängig von Stromstärke und Spannung konstant ist).

Elektrische Stromstärke und Spannung in der Reihenschaltung



- In einer Reihenschaltung ist die Stromstärke an jeder Stelle gleich groß:

$$I_{ges} = I_1 = I_2 = I_3 = \dots$$

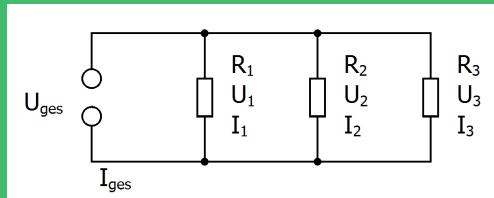
- In einer Reihenschaltung teilt sich die Gesamtspannung auf die einzelnen Bauteile auf:

$$U_{ges} = U_1 + U_2 + U_3 + \dots$$

- In einer Reihenschaltung addieren sich die Einzelwiderstände zum Gesamtwiderstand:

$$R_{ges} = R_1 + R_2 + R_3 + \dots$$

Elektrische Stromstärke und Spannung in der Parallelschaltung



- In einer Parallelschaltung teilt sich die Gesamtstromstärke auf die einzelnen Zweige auf:

$$I_{ges} = I_1 + I_2 + I_3 + \dots$$

- In einer Parallelschaltung ist die Spannung in jedem Zweig gleich groß:

$$U_{ges} = U_1 = U_2 = U_3 = \dots$$

- In einer Parallelschaltung ist der Kehrwert des Gesamtwiderstands gleich der Summe der Kehrwerte der einzelnen Widerstände:

$$\frac{1}{R_{ges}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} + \dots$$

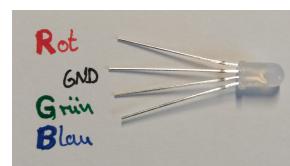
Aufgabe 2: In unserem Starter Kit ist ein 9V Akku untergebracht. Berechne den mindestens notwendigen Vorwiderstand, wenn eine rote LED an den 9V Block angeschlossen wird.

Aufgabe 3:

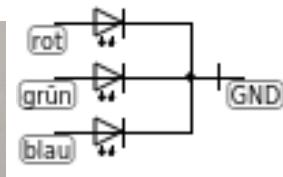
- An dem 9V Block sollen drei rote LEDs in Reihe geschaltet betrieben werden. Zeichne jeweils einen Schaltpunkt und berechne den passenden Vorwiderstand.
- An dem 9V Block sollen drei rote LEDs parallel geschaltet betrieben werden. Zeichne jeweils einen Schaltpunkt und berechne den passenden Vorwiderstand.

Projekt 3: Farbenspektakel mit RGB-LED

Mit einer RGB-LED können die verschiedensten Farben erzeugt werden, die zum Beispiel in Smartphones als Status-LED genutzt werden. RGB steht für Rot, Grün und Blau. In einer RGB-LED sind also drei LEDs gleichzeitig verbaut, die sich in unserem Fall eine gemeinsame Anode (Kontakt mit GND) teilen. Die Anode gehört zum längsten Beinchen.



B 3.5. RGB-LED



B 3.6. Verschaltung der RGB-LED.

Schließe die RGB-LED an den Arduino an. Achte darauf, dass zwischen den Plus-Kontakten der LEDs und den digitalen Pins ein Widerstand von von $330\ \Omega$ geschaltet ist. Der Widerstand lässt sich an der Reihenfolge der Ringe erkennen: Orange - Orange - Schwarz - Schwarz - Braun.

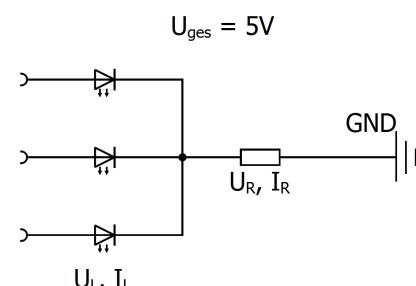


a) Experimentiere mit den verschiedenen Farben. Notiere dir alle möglichen Farbtöne und wie diese zusammengesetzt sind.

b) Zur Reflexion der Berechnung des Vorwiderstands: Begründe, warum der unten berechnete Vorwiderstand zu niedrig ist. Erkläre, wie man stattdessen vorgehen müsste und gib den korrekten Wert für einen möglichen gemeinsamen Vorwiderstand an.

Wie man weitere Farben erzeugt, erfährst du in 5.1.1.

$$\begin{aligned} I_L &= 20 \text{ mA} = 0,02 \text{ A} \implies I_R = 0,06 \text{ mA} \\ U_L &= 2,2 \text{ V} \quad (\text{max. Spannung, die rote LEDs aus-} \\ &\quad \text{halten}) \\ U_R &= 5 \text{ V} - 2,2 \text{ V} = 2,8 \text{ V} \\ R &= \frac{U_R}{I_R} = \frac{2,8 \text{ V}}{0,06 \text{ A}} \approx 46,67 \Omega \end{aligned}$$

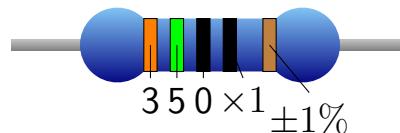


Der Vorwiderstand sollte eine Größe von mindestens $50\ \Omega$ haben.

3.6. Widerstandsringe ablesen

Leider sind die Widerstände zu klein, um ihren Wert darauf gut lesbar zu drucken. Daher werden die Widerstände mit Ringen versehen, aus deren Farbe sich die Größe des Widerstandswertes abliest. Um den für die 7-Segment-Anzeige passenden Widerstand aus dem Bausatz auszuwählen, müssen wir diesen Farbcodierung lesen können.

Bei den blauen Kohleschichtwiderständen, die wir verwenden, gibt es fünf Ringe und jede Ringfarbe steht für eine Zahl. Die ersten drei Ringe bilden die ersten drei Ziffern des Widerstandswertes ab. Die darauf folgende Ringfarbe steht für die Zehnerpotenz, die mit den drei Ziffern multipliziert werden muss. Dies dient dazu, auch größere Widerstandswerte codieren zu können. Der letzte Ring wiederum soll einen etwas größeren Abstand haben und steht für die Fehlertoleranz des Widerstandswertes. In der Praxis lässt sich allerdings nicht immer gut erkennen, welcher Ring der letzte und welcher der erste ist...



Ein Beispiel: Die Ringfarben lauten orange - grün - schwarz - schwarz - braun. Anhand der folgenden Tabelle lässt sich daraus der Wert konstruieren: $3 - 5 - 0 - \cdot 1 (= 10^0) - \pm 1\%$, kurz: $350 \Omega \pm 3,5 \Omega$.

Ringfarbe	1. Ring	2. Ring	3. Ring	4. Ring (Multiplikator)	5. Ring (Toleranz)
Schwarz	0	0	0	$\times 1 / \times 10^0$	-
Braun	1	1	1	$\times 10 / \times 10^1$	1%
Rot	2	2	2	$\times 100 / \times 10^2$	2%
Orange	3	3	3	$\times 1000 / \times 10^3$	-
Gelb	4	4	4	$\times 10.000 / \times 10^4$	-
Grün	5	5	5	$\times 100.000 / \times 10^5$	-
Blau	6	6	6	$\times 1.000.000 / \times 10^6$	-
Lila	7	7	7	-	-
Grau	8	8	8	-	-
Weiß	9	9	9	-	-
Gold	-	-	-	$\times 0.1 / \times 10^{-1}$	5%
Silber	-	-	-	$\times 0.01 / \times 10^{-2}$	10%

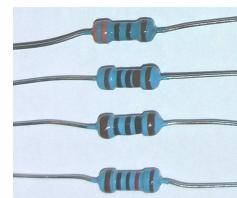
Tabelle 3.1. Tabelle zur Codierung der Widerstandswerte durch Farbringe.

Aufgabe 4: Gib die Farbcodierung für einen Widerstand mit den folgenden Werten an:

- a) $435 \Omega (\pm 2\%)$, b) $570 \text{ k}\Omega (\pm 5\%)$.

Aufgabe 5:

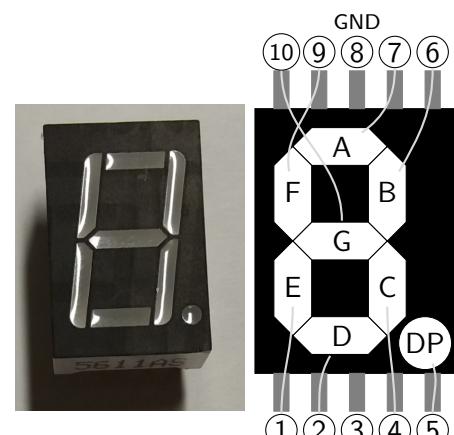
- a) Die rechte Abbildung zeigt die vier wichtigsten Widerstände, mit denen wir zu tun haben werden. Bestimme die jeweilige Größe der Widerstände.
- b) Thorsten hat die Ringe eines Widerstands abgelesen: Silber - rot - lila - lila - grün. Bestimme die Größe des Widerstands.



Zur Kontrolle: <https://www.elektronik-kompendium.de/sites/bau/1109051.htm>

3.6.1. Einfache Verwendung einer 7-Segment-Anzeige

Unsere 7-Segment-Anzeige besteht aus sieben roten LEDs, die so angeordnet sind, dass sich mit ihnen eine Zahl darstellen lässt. Zusätzlich gibt es zur leichteren Unterscheidung von 6 und 9 eine LED für den Punkt. *Jede LED lässt sich einzeln über einen der Pins ansteuern, wobei sich alle LEDs einen gemeinsamen GND-Anschluss teilen.* Der zweite GND-Anschluss soll hier nicht genutzt werden, um die Schaltung so einfach wie möglich zu halten. Es wäre sehr umständlich, für jede LED einen eigenen Vorwiderstand anzuschließen; praktischer ist es, einen einzigen Vorwiderstand zwischen GND-Anschluss der 7-Segment-Anzeige und GND-Anschluss des Arduino anzubringen.



B 3.7. 7-Segment-Anzeige

B 3.8. Pin-Diagramm der 7-Segment-Anzeige.

Aufgabe 6:

- a) Zeichne einen vereinfachten Schaltplan der 7-Segment-Anzeige, in dem die LEDs einzeln eingezeichnet sind.
- b) Als gemeinsamen Vorwiderstand der acht LEDs (Anzeige & Punkt) nutzen wir einen $330\ \Omega$ -Widerstand. Berechne die Gesamtstromstärke und die Stromstärke in jeder LED bei Darstellung einer 1 und einer 8 (jeweils ohne Punkt).

Projekt 4: Raketcoundown

Suche dir nun einen passenden Widerstand für die 7-Segment-Anzeige heraus und verbinde beide mit dem Arduino. Programmiere dann einen Raketcoundown, der von 9 rückwärts bis 0 zählt.
Tipp: Erstelle dir zuerst eine Tabelle, in der du übersichtlich festhältst, welche LEDs für welche Zahl an sein müssen und mit welchen Pins am Arduino diese verbunden sind.

Für Schnelle: Man kann mit einer 7-Segment-Anzeige auch Buchstaben darstellen und nacheinander durchlaufen lassen!

Idee: Frick, Fritsch und Trick (2015): *Einführung in Mikrocontroller - Der Arduino als Steuerzentrale*, Bad Saulgau

3.7. Das elektrische Potential an digitalen Eingängen

In diesem Abschnitt soll geklärt werden, wie man dem Arduino beibringt, auf bestimmte Eingaben aus der Umwelt zu reagieren. Konkret soll am Ende eine Fußgängerampel und eine Juke-Box gebaut werden, die auf Knopfdruck reagieren. Um verstehen zu können, wie der Arduino Signale aus der Umwelt registriert, muss jedoch zuerst geklärt werden, was sich hinter dem *elektrischen Potential* verbirgt. Dazu machen wir einen kleinen Exkurs...

3.7.1. Eine Analogie für das elektrische Potential

 Folie
öffnen

- a) Anna und Bert schauen auf dasselbe Fenster. Anna meint, das Fenster befindet sich in 1 Meter Höhe. Bert hingegen meint, das Fenster befindet sich in 4 Meter Höhe. Beide haben jeweils für sich betrachtet Recht. Wie kann das sein?
- b) Die Vase fällt einen Meter tief. Gib an, wie...
- ... Anna die Höhenenergie nach dem Fallen berechnet und wie sie die Höhenenergie berechnet, die in Bewegungsenergie umgewandelt wurde.
 - ... Bert die Höhenenergie nach dem Fallen berechnet und wie er die Höhenenergie berechnet, die in Bewegungsenergie umgewandelt wurde.

Hinweis: $E_H = m \cdot g \cdot h$

Bei der Berechnung der Höhenenergie muss stets festgelegt werden, wo das *Nullniveau* ist; das bedeutet: Wo die Höhe gemessen wird. Damit wird festgelegt: Bei dieser Grundhöhe ist die Höhenenergie null. Wenn nun eine Vase von der Fensterbank auf den Boden im Raum von Anna fällt, dann hat sie etwas Höhenenergie in Bewegungsenergie umgewandelt - und zwar genau die Höhenenergie, die einem Meter Höhendifferenz entspricht, denn sie ist einen Meter tief gefallen. *Diese Differenz in der Höhe und der Höhenenergie ist unabhängig davon, welche Grundhöhe man betrachtet.* Sowohl aus Annas als auch aus Berts Sicht ist die Vase einen Meter tief gefallen.

Elektrisches Potential

So wie die Höhendifferenz ein Maß für die Höhenenergie ist, die umgewandelt wird (z. B. in Bewegungsenergie), ist die Spannung ein Maß für die elektrische Energie, die an einer LED, einem Widerstand etc. umgewandelt wird. Das elektrische Potential hingegen ist wie die Höhe ein Maß für die elektrische Energie der Elektronen im Stromkreislauf. Es kann nur in Bezug auf ein Nullniveau („Ground“/GND) angegeben werden. Die Einheit des elektrischen Potentials ist Volt.

Elektrisches Potential am GND-Pin: 0V

Elektrisches Potential am 5V-Pin: 5V

Aufgabe 7:

a) Vervollständige die folgende Tabelle von Analogien.

Mechanik	Elektrik
Höhenenergie	
	Elektrisches Potential
Höhendifferenz	
Grundhöhe	

b) Erkläre, welche der oben genannten Größen in der Mechanik bzw. der Elektrik abhängig von der Festlegung eines Nullniveaus sind.

3.7.2. Verwendung eines Tasters

Ziel: Mithilfe des Arduino soll der Zustand eines Tasters eingelesen werden, um damit eine Fußgängerampel zu bauen.

Ein Taster ist wie ein Schalter, kann also geschlossen sein (Strom fließt) oder offen sein (Strom fließt nicht). Im Gegensatz zum Schalter springt ein Taster aber automatisch wieder in den offenen Zustand zurück, wenn er losgelassen wird.

In der unten abgebildeten Schaltplan ist dargestellt, wie man einen Taster am Arduino so anschließt, dass man seinen Zustand im digitalen Pin 3 auslesen kann.

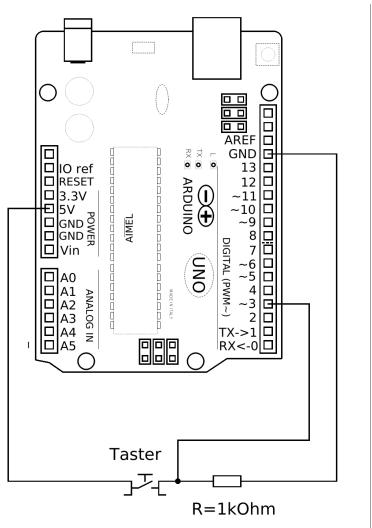


B 3.9. Drei Taster.

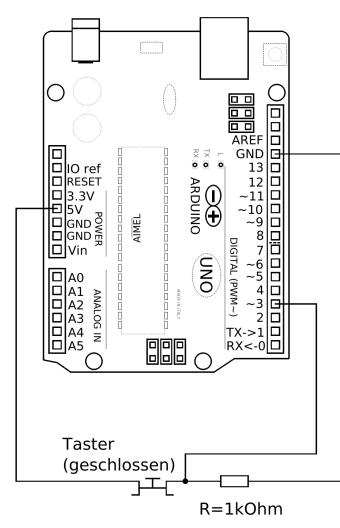
Aufgabe 8: Pulldown-Widerstand

Markiere die Kabel farbig, sodass die Kabel, die auf dem gleichen elektrischen Potential liegen, die gleiche Farbe haben. Notiere zudem den Wert des elektrischen Potentials.

Vorlage
öffnen



B 3.10. Taster offen (kein Stromfluss).



B 3.11. Taster geschlossen (Stromfluss).

Boolische Werte einlesen

Ein Potential von 5 V wird im Programmcode auch als HIGH, 1 oder TRUE bezeichnet. Ein Potential von 0 V wird im Programmcode auch als LOW, 0 oder FALSE bezeichnet. Ein digitaler Pin kann stets nur einen dieser beiden Zustände einnehmen. Potentiale von mehr als 1,4 V werden stets als HIGH bzw. TRUE interpretiert.

Dementsprechend hat der Befehl zum Einlesen des Potentials am digitalen Pin *zwei mögliche Rückgabewerte*: TRUE oder FALSE. Das erkennt man auch an der eckigen Form des Befehls.

lese digitalen Pin 3

Projekt 5: Fußgängerampel

Baue und programmiere eine Fußgängerampel!



Projekt 6: Juke-Box

Baue und programmiere eine Juke-Box!

Die Juke-Box soll zwei verschiedene, kurze Melodien anspielen können. (Zwei mögliche Beispiele mit Link zu den Noten: „Alle meine Entchen“^a, „Oh Tannenbaum“^b).

Dazu werden zwei Taster auf die beschriebene Art an zwei digitale Pins des Arduino angeschlossen. Schließe zudem an einen digitalen Pin einen Piezo-Summer an (siehe unten).

Idee: Frick, Fritsch und Trick (2015): *Einführung in Mikrocontroller - Der Arduino als Steuerzentrale*, Schülerforschungszentrum Bad Saulgau

^ahttps://www.lieder-archiv.de/alle_meine_entchen-notenblatt_100055.html

^bhttps://www.lieder-archiv.de/o_tannenbaum-notenblatt_200078.html

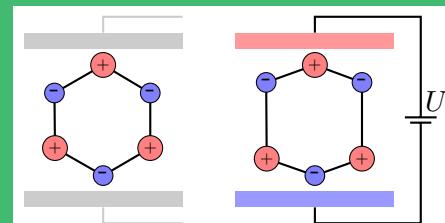
Piezo-Summer

Mit einem Piezo-Summer lassen sich Töne erzeugen, wenn man eine Spannung anschließt. Das lange Bein muss dabei an ein positives Potential angeschlossen werden; das kurze an ein negatives Potenzial bzw. GND. Ein Vorwiderstand ist dabei nicht notwendig, hilft aber die Lautstärke zu reduzieren.



Funktionsweise:

In einem Piezo-Summer befindet sich ein Kristall mit unterschiedlichen Ladungsschwerpunkten, der von einem Kondensator umgeben ist. Wenn von außen an den Kristall eine Spannung angelegt wird, dann verformt sich die Kristallstruktur durch die Anziehung zwischen den Ladungsschwerpunkten und den Kondensatorplatten (*inverser piezo-elektrischer Effekt*). Wenn keine Spannung anliegt, verformt sich der Kristall zurück. Durch diese Verformungen entstehen Druckwellen in der Luft, die wir als Ton wahrnehmen können.



3.8. Vermischte Übungen

Aufgabe 9: Reihenschaltung

Eine rote LED soll an Pin 13 des Arduino betrieben werden. Durch die LED soll eine Stromstärke von 10 mA fließen, was bei einer Spannung von 2,1 V an der LED der Fall ist.

- Zeichne den zugehörigen Schaltplan.
- Berechne, wie groß der Vorwiderstand gewählt werden muss, damit diese Werte erreicht werden.

Aufgabe 10: Parallelschaltung

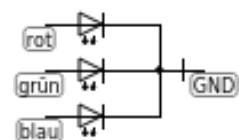
Drei grüne LEDs sollen parallel geschaltet an Pin 13 des Arduino angeschlossen und mit einem gemeinsamen Vorwiderstand betrieben werden. Die LEDs halten eine Stromstärke von maximal 20 mA bei einer Spannung von 3,3 V aus.

- Zeichne den zugehörigen Schaltplan.
- Ein Digitalpin am Arduino darf maximal mit einer Stromstärke von 40 mA belastet werden. Berechne, welche Stromstärke dann maximal durch die einzelnen LEDs fließen darf.
- Der Tabelle unten kannst du den zugehörigen Spannungswert an den LEDs entnehmen. Berechne, wie groß der gemeinsame Vorwiderstand der LEDs sein muss, damit die in b) berechnete Stromstärke eingehalten wird.

Spannung U	3,03 V	3,07 V	3,1 V	3,13 V	3,16 V	3,19 V
Stromstärke I	10 mA	11 mA	12 mA	13 mA	14 mA	15 mA

Aufgabe 11: Schaltung einer RGB-LED

Eine RGB-LED besteht aus drei einzelnen LEDs (rot, grün, blau), die jeweils über einen eigenen Digitalpin angesteuert werden (vgl. Schaltplan rechts). Am gemeinsamen GND-Anschluss soll ein gemeinsamer Vorwiderstand für alle LEDs angebracht werden, um die Stromstärke auf maximal 15 mA zu begrenzen. Die Spannung an den LEDs sollte dann 2,25 V nicht überschreiten.



B 3.12. Verschaltung der RGB-LED.

- Erkläre, welche Unterschiede zur Parallelschaltung von drei LEDs an *einem* Digitalpin zu beachten sind.
- Berechne, wie groß der gemeinsame Vorwiderstand mindestens sein muss.

Aufgabe 12: Farbcodierung von Widerständen

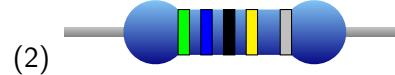
a) Gib die Farbcodierung der folgenden Widerstandsgrößen an:

$$(1) \ 330\Omega \pm 1\%$$

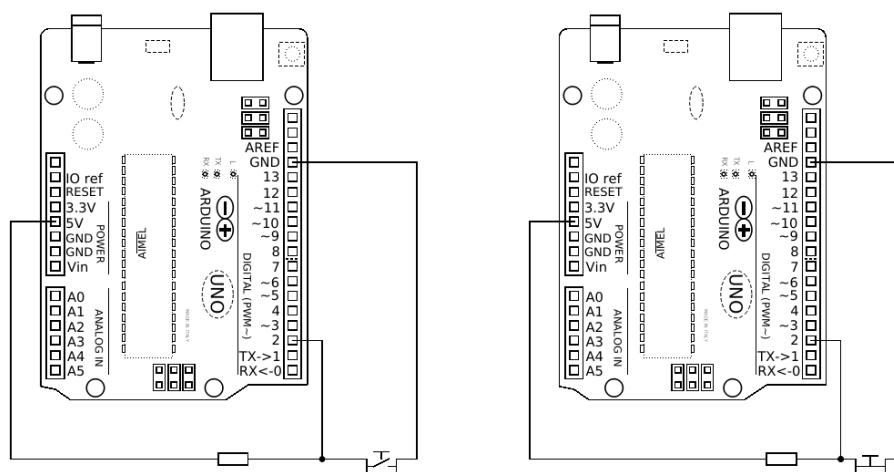
$$(2) \ 10\text{k}\Omega \pm 2\%$$

$$(3) \ 4,7\text{k}\Omega \pm 10\%$$

b) Gib die Größe der folgenden Widerstände an:

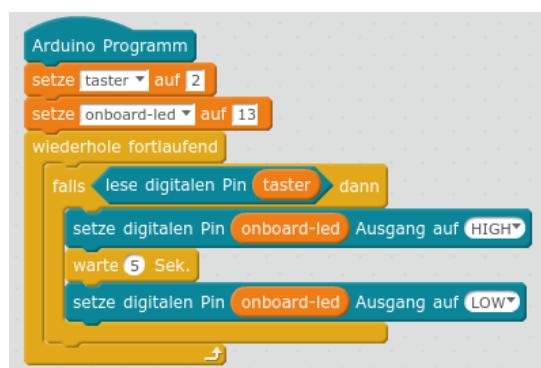


Hinweis: Als Hilfsmittel ist die Widerstandskarte aus den Boxen zugelassen.

Aufgabe 13: Pullup-Widerstand

In der Abbildung wird ein Taster mit einem sogenannten Pullup-Widerstand an den Arduino angeschlossen. Links ist der Taster offen, rechts ist der Taster geschlossen.

- a) Markiere die Kabel jeweils farbig, sodass die Kabel, die auf dem gleichen elektrischen Potential liegen, die gleiche Farbe haben. Notiere zudem den Wert des elektrischen Potentials.
- b) Erkläre, wie sich die Schaltung verhält, wenn das rechts abgebildete Programm auf dem Arduino läuft.



Motivationsquellen

> [FUTUREMAG](#)

Kurze Dokumentation des FUTUREMAG zur Arduino-Welt

> [Arduino-Wecker](#)

Mit einem selbst gebauten Wecker, der Uhrzeit und Alarmzeit getrennt voneinander anzeigt, erfüllte ein Bastler seinen Wunsch nach mehr Komfort.

> [Aquarium-Licht](#)

Der Bastler hinter diesem Projekt wollte seinen Fischen im Aquarium ein natürliches Licht einschließlich Sonnenaufgang, Sonnenuntergang und Nacht gönnen. Auf die gleiche Art und Weise kann man natürlich auch sein Terrarium beleuchten.

> [VR-Brille](#)

Drei Schüler aus Frankreich hatten kein Geld für eine Virtual Reality Brille – aber dafür das Know How, um sich mit einem Arduino und einem Gehäuse aus dem 3D-Drucker selbst eine VR-Brille zu basteln.

4. Konzepte der Informatik: Bausteine von Algorithmen

Mit den digitalen Pins, die sich als Ausgänge und Eingänge nutzen lassen, lassen sich zahlreiche Projekte umsetzen. Allerdings wird die Programmierung schnell unübersichtlich oder unnötig aufwendig, wenn man sich nicht mit algorithmischen Strukturen auskennt. Daher geht es im folgenden Kapitel um die Einführung von grundlegenden algorithmischen Bausteinen.

In diesem Kapitel lernst du...

- ... Variablen zu benutzen,
- ... mit Schleifen effizient zu programmieren,
- ... zufällige Ereignisse zu programmieren,
- ... den Arduino mit dem Computer kommunizieren zu lassen,
- ... überblicksweise, wie die USB-Verbindung funktioniert,
- ... was es mit den berühmten Einsen und Nullen auf sich hat,
- ... wie man Programme zur Planung oder zum Vergleich auf Papier einfach darstellen kann.

Projekte in diesem Kapitel:

> Auto-Blinker	24	> Reaktionszeitmesser	25
> Bombe bauen	25		

4.1. Programme mit Variablen strukturieren

Aufgabe 1: Jonas und Jana haben jeweils LEDs an die Pins 13 bis 11 angeschlossen und steuern diese mit den unten abgebildeten Programmen. Vergleiche die beiden Programme im Hinblick auf ihre Wirkung und die Art der Programmierung. Welches gefällt dir besser?

Zusatzüberlegung: Wie viel muss man ändern, wenn man zum Beispiel Probleme mit Pin 13 hat und deshalb die zugehörige LED an Pin 10 anschließen will?

Folie
öffnen

```

Arduino Programm
wiederhole fortlaufend
  setze digitalen Pin 13 Ausgang auf HIGH
  setze digitalen Pin 12 Ausgang auf LOW
  setze digitalen Pin 11 Ausgang auf LOW
  warte 1 Sek.
  setze digitalen Pin 13 Ausgang auf LOW
  setze digitalen Pin 12 Ausgang auf HIGH
  setze digitalen Pin 11 Ausgang auf LOW
  warte 1 Sek.
  setze digitalen Pin 13 Ausgang auf LOW
  setze digitalen Pin 12 Ausgang auf LOW
  setze digitalen Pin 11 Ausgang auf HIGH
  warte 1 Sek.
  setze digitalen Pin 13 Ausgang auf LOW
  setze digitalen Pin 12 Ausgang auf HIGH
  setze digitalen Pin 11 Ausgang auf LOW
  warte 1 Sek.

```

B 4.1. Jonas Programm zum Steuern der LEDs.

```

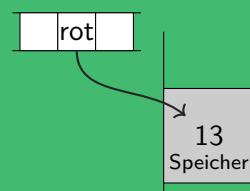
Arduino Programm
setze rot auf 13
setze gelb auf 12
setze gruen auf 11
setze wechselzeit auf 1
setze rotzeit auf 8
setze gruenzeit auf 5
wiederhole fortlaufend
  setze digitalen Pin rot Ausgang auf HIGH
  setze digitalen Pin gelb Ausgang auf LOW
  setze digitalen Pin gruen Ausgang auf LOW
  warte rotzeit Sek.
  setze digitalen Pin rot Ausgang auf LOW
  setze digitalen Pin gelb Ausgang auf HIGH
  setze digitalen Pin gruen Ausgang auf LOW
  warte wechselzeit Sek.
  setze digitalen Pin rot Ausgang auf LOW
  setze digitalen Pin gelb Ausgang auf LOW
  setze digitalen Pin gruen Ausgang auf HIGH
  warte gruenzeit Sek.
  setze digitalen Pin rot Ausgang auf LOW
  setze digitalen Pin gelb Ausgang auf HIGH
  setze digitalen Pin gruen Ausgang auf LOW
  warte wechselzeit Sek.

```

B 4.2. Jana's Programm zum Steuern der LEDs.

Variablen

Eine Variable kann man sich als Koffer vorstellen, der einen Namen bekommt und in dem man einen Zahlenwert oder ein Wort speichert. Jedes Mal, wenn der Name des Koffers aufgerufen wird, wird der abgespeicherte Wert hervorgeholt und an die Stelle des Namens gesetzt. Intern wird der Variablenname als Verweis auf einen bestimmten Speicherplatz genutzt, in dem der Wert der Variable abgelegt ist.



Wir unterscheiden drei **Datentypen**, die in Variablen abgespeichert werden: Zahlen (runde Felder), Zeichen (rechteckige Felder) und Wahrheitswerte (spitze Felder: Wahr oder falsch). Wahrheitswerte werden auch *boolsche Variablen* genannt.

4.2. Lauflichter effizient programmieren

Lauflichter findet man inzwischen überall in unserer Welt: An den Rändern von Landebahnen an Flughäfen, an Spieleanutomaten, aufdringlichen Werbeschildern, als Blinker von modernen Autos und vieles mehr.

Ziel: Ein Auto-Blinker soll auf möglichst effiziente Weise programmiert werden.

Aufgabe 2: Fange mit einem einfachen Lauflicht an und bau dieses immer weiter aus.

- Bringe vier LEDs mit Vorwiderständen von 330Ω an den Pins 10 bis 13 an. Programmiere sie so, dass sie nacheinander aufleuchten und wieder ausgehen. Nutze für die Pausenzeit eine Variable.
 - Du wirst leicht erkennen, dass sich die Schritte zum Aufleuchten und Ausstellen bei jeder LED wiederholen. Sie lassen sich effizienter mit einer Wiederholungsschleife programmieren, die aber nicht unendlich lange weiterläuft.
Lege eine Laufvariable namens `pin` an, die für die Angabe des Pins genutzt wird, der an bzw. ausgestellt werden soll. In jedem Schleifendurchlauf muss die `pin`-Variable dann entsprechend geändert werden.
- Programmiere das Lauflicht aus a) mithilfe einer Laufvariable und einer Wiederholungsschleife möglichst effizient.
- c) *Für Schnelle:* Ergänze weitere LEDs und passe dein Programm daran an.



B 4.3. Mögliche Schleifen
für Teil b.

Aufgabe 3: Programmiere ein Lauflicht, das hin- und zurückläuft.

Projekt 1: Auto-Blinker

Programmiere ein Lauflicht so, wie es auch als Blinker in modernen Autos genutzt wird.

Schleifen

Bei der Programmierung werden häufig Schleifen genutzt, die die Anweisungen in ihrem Rumpf (oder Körper) solange wiederholen, bis eine gewisse Abbruchbedingung eintritt.

Zählschleifen wiederholen die Anweisungen im Rumpf für eine festgelegte Anzahl von Wiederholungen.

Die `wiederhole bis`-Schleife wiederholt die Anweisungen solange, bis die angegebene Bedingung wahr ist. Zum Beispiel lässt sich mit dem grünen Block für den Vergleich zweier Zahlen eine WAHR/FALSCH-Aussage erzeugen, wie man an der eckigen Form erkennt. Ein weiteres Beispiel ist der Block `lese digitalen Pin`, der ebenfalls eine WAHR/FALSCH-Aussage erzeugt (vgl. S. 17). Die Überprüfung, ob die Bedingung wahr ist, erfolgt hier vor der Ausführung der Befehle im Rumpf. Daher nennt man die Schleife auch *kopfgesteuert*.

4.3. Zufällige Ereignisse programmieren

Viele Dinge werden interessanter, wenn sie sich nicht immer auf die genauso gleiche Art wiederholen. Für diese Fälle kann man im Programm den grünen Block für Zufallszahlen verwenden, der jedes Mal eine neue Zufallszahl erzeugt, wenn er aufgerufen wird. Ein einfaches Beispiel ist die „Bombe“, die man bei dem Spiel „Tick Tack Bumm“ startet und die man so lange herum geben muss, bis sie explodiert. Dabei ist die Dauer des Tickens ein zufälliger Wert zwischen ca. 5 s und 20 s.

Zufallszahl von 1 bis 10



Bomben-Beispielvideo



Programm-vergleich

Projekt 2: Bombe bauen

Baue und programmiere eine „Bombe“, die für eine zufällige Dauer zwischen 5 s und 20 s tickt und dann explodiert. Die Bombe wird über einen Taster aktiviert.

4.4. Kommunikation mit dem Arduino: Der serielle Monitor

Bisher hatte die Kommunikation mit dem Arduino stets nur eine Richtung: Vom Computer zum Arduino. Das reicht nicht mehr, wenn der Arduino etwas messen und dem Anwender dann mitteilen soll, was er gemessen hat. Für solche Fälle ist der serielle Monitor die einfachste Kommunikationsmöglichkeit zwischen Arduino und Computer.

Frage: Wie kann der Arduino mit dem Computer kommunizieren?

Mit dem Block `Sende Text an seriellen Port` kann der Arduino angewiesen werden, den eingegebenen Text über die serielle Schnittstelle an den Computer zu senden. Der Text kann auch eine Zahl sein - diese Zahl wird dann eben nicht mehr als Zahl, sondern als Text behandelt und ausgegeben. Am Computer sieht man das Ergebnis im Fenster unten rechts, in dem bisher bereits Statusnachrichten zur Übertragung des Programms auf den Arduino zu sehen waren. *Hier muss der Zeichenmodus eingestellt werden, damit die Daten für uns lesbar sind.*

sende Text an seriellen Port Hallo

Projekt 3: Reaktionszeitmesser

Baue und programmiere einen Reaktionszeitmesser.

Der Reaktionszeitmesser soll zunächst warten, bis ein Taster gedrückt wurde, der besagt, dass es losgehen kann. Dann wird eine LED angeschaltet (Vorwiderstand!) und nach einer zufälligen Zeit wieder ausgeschaltet. Nun beginnt die Zeitmessung. Die Stoppuhr läuft solange, bis der Taster gedrückt wurde.

setze Stoppuhr zurück

Stoppuhr

Die gemessene Zeit wird dann über den seriellen Monitor ausgegeben und es wird erneut gewartet, bis der Anwender bestätigt, dass es losgehen kann.

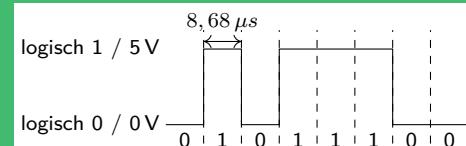
Miss mindestens zehn Mal deine Reaktionszeit und bestimme den Mittelwert. Bist du besser als dein Partner?

Für Schnelle: Nutze den `Verbinde <_> mit <_>`-Block, um der Ausgabe folgende Form zu geben: „Deine Reaktionszeit: 10s“.

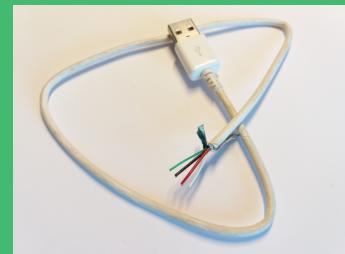
Serielle Schnittstellen

Bei seriellen Schnittstellen werden die Bits, also die Einsen und Nullen nacheinander (seriell) von einem Gerät zum nächsten geschickt. Dazu werden el. Potenzialpulse mit einer festgelegten Länge von einem Gerät gesendet und vom anderen Gerät empfangen. Dabei kann ein Puls mit 5V als eine 1 und ein Puls von 0V als eine 0 interpretiert werden. In unserem Fall ist eine Bitrate von 115200 Bit pro Sekunde festgelegt, wodurch ein Bit bzw. Potenzialpuls 8,68 µs dauert.

Die bekannteste serielle Schnittstelle ist USB - der *Universal Serial Bus*. Für die Kommunikation werden vier Kabel benötigt, die man erkennen kann, wenn man das USB-Kabel durchschneidet: Das schwarze Kabel (GND) und das rote Kabel (5 V) stellen die Spannungsversorgung sicher. Dazu kommt eine Empfängerleitung (RX) und eine Senderleitung (TX). Der Empfängerpin des Arduino ist Pin 0 und muss mit dem Senderpin am Computer verbunden sein. Umgekehrt muss der Senderpin des Arduino (Pin 1) mit dem Empfängerpin am Computer verbunden sein. Wenn etwas an Pin 0 oder 1 des Arduino angeschlossen ist, kann es passieren, dass die Übertragung eines neuen Programms auf den Arduino nicht funktioniert, weil das angeschlossene Bauteil das richtige Setzen der Potentialpulse verhindert.



B 4.4. Durch ein gepulstes elektrisches Potential wird eine Folge von Bits zwischen Arduino und Computer übertragen.



B 4.5. Ein USB-Kabel enthält vier Leitungen.

Aufgabe 4: Angenommen, du hast ein Programm erstellt, das 75 kB groß ist. Berechne, wie lange es bei einer Bitrate von 115200 Bit pro Sekunde dauert, diese Datenmenge auf den Arduino zu übertragen.

Hinweise: 1 kB = 1 kilo Byte = 1000 Byte, 1 Byte = 8 Bit

4.4.1. Exkurs: Zufallszahlen von Mikrocontrollern/-prozessoren

Aufgabe 5:

Übertrage das rechts abgebildete Programm auf den Arduino und betrachte die so erzeugten Zufallszahlen. Drücke dann auf den Reset-Taster am Arduino und betrachte die nun erzeugten Zufallszahlen. Wiederhole den Vorgang einige Male und beschreibe Auffälligkeiten.



4.5. Kurze Einführung in die Codierung von Zahlen und Zeichen

Wie wir gesehen haben, können der Computer und der Arduino nur die berühmten Einsen und Nullen austauschen.

Frage: Wie ist es möglich, mithilfe von Einsen und Nullen Zahlen und Zeichen darzustellen?

4.5.1. Ein Einblick ins Binärsystem

Aufgabe 6:

- a) Erkläre die unten dargestellte Zerlegung der Zahl 1509 im Dezimalsystem:

$$\begin{aligned} 1509 &= 1 \cdot 1000 + 5 \cdot 100 + 0 \cdot 10 + 9 \cdot 1 \\ &= 1 \cdot 10^3 + 5 \cdot 10^2 + 0 \cdot 10^1 + 9 \cdot 10^0 \end{aligned}$$

- b) Das Binärsystem arbeitet mit Zweierpotenzen statt Zehnerpotenzen. Notiere alle Zweierpotenzen von 2^0 bis 2^{10} . Erkläre kurz, wieso es bei der Verwendung von Zweierpotenzen nur die Ziffern 0 und 1 geben kann.
- c) Übersetze die Zahlen vom Binärsystem ins Dezimalsystem:
- | | | |
|----------|--------------|--------------|
| (1) 1011 | (2) 11000001 | (3) 01010101 |
|----------|--------------|--------------|
- d) Finde die Binär-Darstellungen von 12, 21 und 127.
- e) Ein Byte besteht aus acht Bit, also acht Nullen oder Einsen. Bestimme, wie viele verschiedene Zahlen man damit darstellen kann.

Für negative Zahlen und Zahlen mit Komma braucht man noch einige zusätzliche Kniffs, die an dieser Stelle zu weit führen.



Das Binärsystem

Im Gegensatz zum Dezimalsystem zur Basis 10 mit Ziffern von 0 bis 9 arbeitet das Binärsystem zur Basis 2 mit den Ziffern 0 und 1.

Im Binärsystem wird jede Zahl durch die Potenzen von 2 zusammengesetzt. Die Binärzahl 1001 bedeutet dann so viel wie $1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$. Den Wert der Binärzahl im Dezimalsystem erhält man dann durch eben diese Rechnung: $1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 8 + 0 + 0 + 1 = 9$.

Da bei Binärzahlen wie 1001 nicht sofort ersichtlich ist, ob es sich um eine Binärzahl oder eine Dezimalzahl handeln soll, wird der Zahl häufig ein 0b vorangestellt: 0b1001. Ebenfalls üblich ist es, die Basis im Index zu notieren: 1001_2 zeigt eine Binärzahl an; 1001_{10} eine Dezimalzahl.

Um die Binärdarstellung einer Dezimalzahl wie z. B. 13 zu finden, geht man folgendermaßen vor:

1. Finde die größte Zweierpotenz, die in die Dezimalzahl passt. Notiere dafür eine 1.
2. Überprüfe, wie oft die nächstkleinere Zweierpotenz in den Rest passt. Notiere entsprechend eine 0 (passt nicht) oder eine 1 (passt einmal).
3. Wiederhole den Schritt (2), bis kein Rest mehr übrig ist.

$$\begin{array}{rcl} 13 = 1 \cdot 8 + 5 & 1 \\ 5 = 1 \cdot 4 + 1 & 11 \\ 1 = 0 \cdot 2 + 1 & 110 \\ 1 = 1 \cdot 1 & 1101 \end{array}$$

4.5.2. Ein Einblick in die ASCII-Tabelle

Auf diese Codierung im Binärsystem wird dann zurückgegriffen, wenn die übertragenen (bzw. abgespeicherten) Daten vom Typ „Zahl“ sind. Wie man an der eckigen Form des Arguments im Block `Sende Text an seriellen Port` sieht, werden dabei aber Daten vom Typ „Zeichen“ übertragen. Um welches Zeichen es sich handelt, lässt sich nicht allein anhand der Bits schlussfolgern, sondern muss durch einen Standard festgelegt werden. Der einfachste Standard ist der ASCII-Standard, der später in den Unicode- und schließlich in den UTF-8-Standard eingeflossen ist.

Der ASCII-Code

Der [American Standard Code for Information Interchange \(ASCII\)](#) legt durch eine Tabelle für 128 Zeichen fest, wie diese mithilfe von sieben Bits codiert werden. Wenn man diese Bitfolge nicht als Zeichen, sondern als Zahl interpretiert, kann man jedem Zeichen eine Zahl zuordnen, die diesem entspricht. Zum Beispiel wird der Buchstabe A durch die Folge 1000001 codiert und entspricht damit der Zahl 65.

Aufgabe 7: Finde und notiere die ASCII-Bitfolgen zu den Zeichen „V“, „s“, „+“, „?“ und „0“. ASCII-Tabelle: <http://www.computer-masters.de/ascii-tabelle.php>

🔗 Recherche: Von ASCII über Unicode zu UTF-8

Der ASCII-Standard ist ausreichend, wenn man nur englische Texte darstellen muss. In anderen Sprachen ergibt sich jedoch die Herausforderung, weitere Zeichen codieren zu müssen und spätestens mit der Verbreitung des Internets und der weltweiten Kommunikation war das auch für englischsprachige Programmierer relevant. Eine locker geschriebene Einführung zur Entwicklung der unterschiedlichen Standards gibt der Artikel [The Absolute Minimum Every Software Developer Absolutely, Positively Must Know About Unicode and Character Sets \(No Excuses!\)](#).

Hilfreich für das Verständnis des Artikels ist die Kenntnis von Hexadezimalzahlen (s. S. 36 f.).

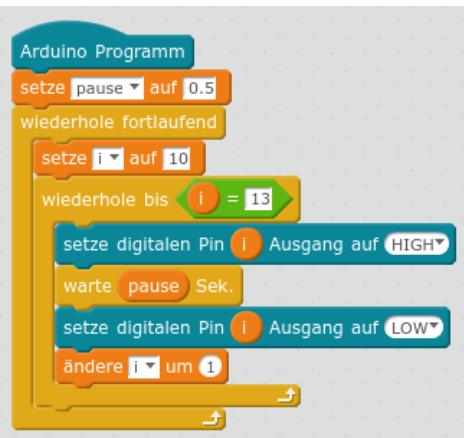
4.6. Programme mit Struktogrammen darstellen

Wenn wir uns über Programme austauschen, dann haben wir nicht immer den Computer zur Hand. In solchen Momenten wäre es viel zu aufwendig, die bunten Blöcke von mBlock zu malen. Außerdem könnte es sein, dass jemand anderes das Programm nicht mit Blöcken, sondern mit Text in der Programmiersprache C++ aufschreiben will, also so wie es in der rechten Seite von mBlock abgebildet ist.

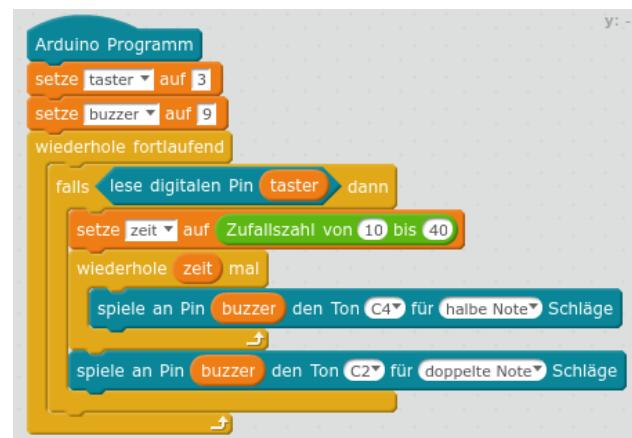
Frage: Wie kann man Programme übersichtlich zu Papier bringen?

Man nutzt zur Darstellung des Ablaufs eines Computerprogramms sogenannte **Struktogramme** (vgl. Tab. 4.1), die in den 1970er Jahren von Isaac Nassi und Ben Shneidermann entwickelt wurden. Struktogramme sollen ein Computerprogramm möglichst einfach und ohne programmiersprachen-spezifische Befehlssyntax abbilden. Auf diese Art und Weise lassen sich Programme auch einfach planen, bevor man sich damit beschäftigt, wie die Schritte im Einzelnen zu codieren sind.

Aufgabe 8: Stelle die unten abgebildeten Programme jeweils mithilfe eines Struktogramms dar. Fasse die Wirkung der Programme jeweils kurz zusammen.



B 4.6. Programm A.



B 4.7. Programm B.

Darstellung von Programmen in Struktogrammen

Lineare Struktur

Jede Anweisung wird in einen rechteckigen Block geschrieben.

Anweisung 1

Anweisung 2

Wiederholung / Schleifen

Zählergesteuerte Schleife

Die Anzahl der Schleifendurchläufe wird durch eine Zählvariable festgelegt. Im Schleifenkopf werden der Startwert der Zählvariablen, der Endwert der Zählvariablen und die Veränderung der Zählvariablen, z. B. Schrittweite 1, angegeben.

zähle Variable von Startwert bis Endwert, Schrittweite

Anweisung 1

Kopfgesteuerte Schleife

Wiederholungsschleife mit vorausgehender Prüfung der Bedingung. Der Schleifenkörper wird so lange wiederholt, *wie* oder *bis* die Bedingung wahr ist (bei uns nur der letzte Fall verfügbar).

solange, wie/bis Bedingung wahr

Anweisung 1

Fußgesteuerte Schleife

Wiederholungsschleife mit nachfolgender Prüfung der Bedingung. Der Schleifenkörper wird so lange wiederholt, *wie* oder *bis* die Bedingung wahr ist (in mBlock nicht verfügbar).

Anweisung 1

solange, wie/bis Bedingung wahr

Verzweigung

Einfache Verzweigung

Die Anweisung 1 (und ggf. weitere) wird nur ausgeführt, falls die Bedingung wahr ist. Andernfalls wird nichts gemacht.



Alternative Verzweigung

Falls die Bedingung wahr ist, wird Anweisung 1 (und ggf. weitere) ausgeführt, sonst wird Anweisung 2 (und ggf. weitere) ausgeführt.



Verschachtelte Verzweigung

Falls Bedingung 1 wahr ist, folgt eine weitere Bedingung 2.

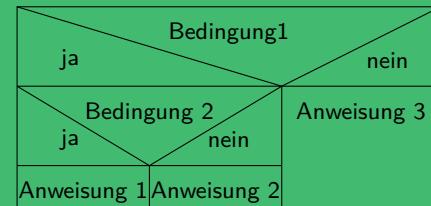


Tabelle 4.1. Tabelle zur Darstellung eines Programms als Struktogramm nach Nassi und Schneidermann.

4.7. Vermischte Übungen

Aufgabe 9: Variablen

- Nenne Vorteile, die für die Verwendung von Variablen sprechen.
- Nenne drei Datentypen, die in Variablen abgespeichert werden können.

Aufgabe 10: Schleifen

An allen Digitalpins des Arduino werden LEDs mit geeigneten Vorwiderständen angeschlossen. Dann wird das rechts abgebildete Programm ausgeführt.

- Erstelle eine Trace-Tabelle für einen Durchlauf der `Wiederhole fortlaufend`-Schleife.
- Nenne die Pin-Nummern der LEDs, die nach Durchlaufen dieses Programms einmal geleuchtet haben.
- Stelle das Programm als Struktogramm dar.



Aufgabe 11: Bitübertragung

Ein Programm ist 30 kB groß. Berechne, wie lange es bei einer Bitrate von 115200 Bit pro Sekunde dauert, das Programm auf den Arduino zu übertragen.

Aufgabe 12: Das Binärsystem und das Dezimalsystem

- Übersetze vom Binär- ins Dezimalsystem.

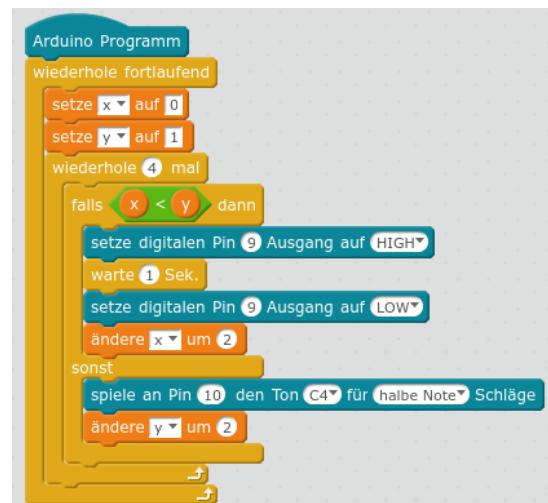
(1) 1001 ₂	(2) 1010 ₂	(3) 1111 ₂
-----------------------	-----------------------	-----------------------
- Übersetze vom Dezimal- ins Binärsystem.

(1) 11	(2) 7	(3) 14
--------	-------	--------

Aufgabe 13: Programme verstehen

Am Arduino wird an Pin 9 eine LED mit Vorwiderstand und an Pin 10 ein Piezo-Summer angebracht.

- Stelle das Programm als Struktogramm dar.
- Beschreibe die Wirkung des rechts abgebildeten Programms.
- Erkläre, wie man das Programm ändern müsste, damit die LED zwei Mal blinkt, bevor wieder der Piezo-Summer piept.



Aufgabe 14: Programm entwickeln

An allen Digitalpins des Arduino sind LEDs mit Vorwiderstand angeschlossen. Für das folgende Programm ist bereits eine Variable namens `p` angelegt.

Entwickle ein Programm, das die LEDs an Pin 1 bis 5 [2,4,6,8] nacheinander zum Leuchten bringt und nach einer Sekunde wieder ausschaltet. Es leuchtet also immer nur eine LED zur selben Zeit.

Anforderungen:

- Das Programm soll als Struktogramm dargestellt werden.
- Es sollen so wenig Code-Wiederholungen wie möglich vorkommen (*effizientes Programmieren*).

Befehle:



Motivationsquellen

> [Staubsaugerroboter](#)

Mithilfe eines Arduino haben zwei Schüler aus den Niederlanden ihren eigenen Staubsaugerroboter gebaut. Das Teile für das Gehäuse haben sie mit einem 3D-Drucker gedruckt.

> [Levitation](#)

Durch Levitation lassen sich Gegenstände scheinbar von Geisterhand in der Luft schweben. Die nötige Elektronik dafür lässt sich mit einem Arduino realisieren.

> [Arduino-kontrollierter LED-Streifen zur Visualisierung von Musik](#)

Der Arduino lässt sich zudem über ein Smartphone ansteuern.

> [Spielekonsole von Makerbuino](#)

Mithilfe eines Bausatzes lässt sich eine kleine, Arduino-basierte Spieldatenkonsole bauen.

5. Analoge Ausgänge und Eingänge

Digitale Bauteile haben ein Problem: Sie kennen nur Eins oder Null, An oder Aus, Wahr oder Falsch. Die Realität ist aber natürlich um einiges komplexer, nämlich analog: Reale Größen können durch im Prinzip beliebige reelle Zahlen ausgedrückt werden. Daher wäre es wünschenswert, auch analoge Werte angeben zu können. In diesem Kapitel lernst du, wie man mit dem Arduino quasi-analoge Werte erzeugen oder einlesen kann. Dabei ist es nie möglich, wirklich beliebig genaue Werte zu erreichen, wie man es bei analogen Werten erwarten würde, aber es gibt einige interessante Tricks, wie man mit einem digitalen Bauteil analoge Werte simulieren kann.

In diesem Kapitel lernst du...

- ... Pulsweitenmodulation und Hexadezimalzahlen zu benutzen,
- ... Spannungen zu messen und Spannungen an einem Spannungsteiler zu berechnen,
- ... logische Bedingungen mit logischen Operationen zu verfeinern,
- ... ein Potentiometer, einen LDR sowie einen NTC auszulesen, um eine Drehung, die Umgebungshelligkeit bzw. die Temperatur zu messen,
- ... wie Transistoren verwendet werden und was diese mit dem Aufbau des Arduino zu tun haben,
- ... was es mit dem EVA-Prinzip auf sich hat.

Projekte in diesem Kapitel:

> Kerzenfunkeln	34	> Dimmbare Lampe ohne μC	42
> Farbenspektakel mit RGB-Codes	36	> Straßenlampe	44
> Batterietester (kleine Spannung)	38	> Alarmanlage	44
> Batterietester (große Spannung)	38	> Thermometer	48
> Dimmbare Lampe	41		

5.1. Pulsweitenmodulation (PWM)

Ziel: Mithilfe des Arduino soll eine funkelnde LED-Kerze gebaut werden.

Der Arduino verfügt über mehrere sogenannte PWM-Pins, die mit einer Tilde (~) gekennzeichnet sind. Diese lassen sich mit dem Befehl `setze PWM-Pin <_> Ausgang auf <_>` ansteuern. Im ersten Argument muss die Nummer des Pins angegeben werden, im zweiten Argument wird ein PWM-Wert zwischen 0 und 255 angegeben.

`setze PWM-Pin 5 Ausgang auf 158`

Aufgabe 1: *Fading*

- Schließe eine LED mit Vorwiderstand an einen PWM-Pin an und verwende die PWM-Anweisung mit unterschiedlichen PWM-Werten. Beschreibe die Wirkung auf die LED in einem Je-Desto-Satz.
- Lege eine Variable `h` für den PWM-Wert an. Nutze die Variable, um systematisch alle PWM-Werte von 0 bis 255 zu durchlaufen, sodass jeder Wert kurz sichtbar wird (z. B. 0.2 Sekunden).
- Für Schnelle:* Sorge dafür, dass die PWM-Werte nach Erreichen der 255 genauso rückwärts von 255 bis 0 durchlaufen werden.

Aufgabe 2: *Pulsweitenmodulation*

Erkläre mithilfe der Zusammenfassung zur Pulsweitenmodulation, was bei der Nutzung des Befehls `Setze PWM-Pin 5 Ausgang auf 158` passiert. Berechne auch die mittlere Spannung, die am PWM-Pin ausgegeben wird.

Für Physik-Profis: Eine blaue LED hält bis zu 3,5 V aus, ohne durchzubrennen. Trotzdem darf man sie bei Verwendung dieses Befehls nicht ohne Vorwiderstand an den Pin anschließen. Begründe.

Für Informatik-Profis: Wie viel Bit stehen zur Speicherung des Tastverhältnisses mit Werten von 0 bis 255 zur Verfügung?



Neues

Werkzeug:

Servo-motor

Projekt 1: Kerzenfunkeln

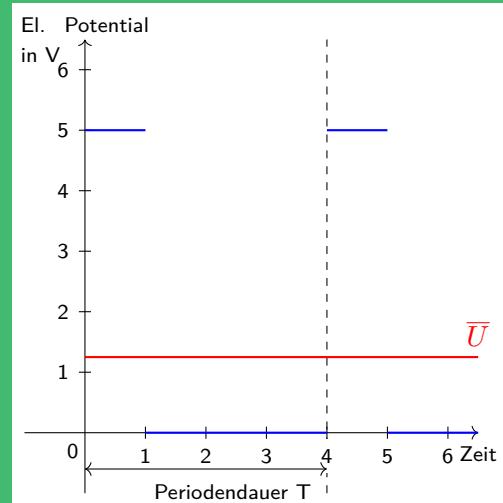
Modelliere mithilfe von drei LEDs das Funkeln von Kerzen.

Tipp: Die Verwendung des Blocks für Zufallszahlen wird dir helfen, das Funkeln natürlicher aussehen zu lassen.

Pulsweitenmodulation (PWM)

Bei der Pulsweitenmodulation wechselt der ausgewählte digitale Pin sehr schnell zwischen den elektrischen Potentialen 5 V und 0 V hin und her - es ergibt sich also ein gepulstes Signal, dessen Weite (Dauer) moduliert werden kann. Aus dem Verhältnis der Zeit, in der der Pin auf einem 5 V-Potential liegt, zu der Zeit, in der der Pin auf einem 0 V-Potential liegt, ergibt sich eine mittlere Spannung (gegenüber Ground), die scheinbar am Pin anliegt. Wenn der Pin in der Hälfte der Zeit auf 5 V und in der anderen Hälfte auf 0 V liegt, dann ergibt sich eine mittlere Spannung von $\bar{U} = 2,5 \text{ V}$. Wenn der Pin nur in einem Viertel der Zeit auf 5 V liegt, dann ergibt sich eine mittlere Spannung von $\bar{U} = 1,25 \text{ V}$ ($= 5 \text{ V} \cdot 0,25$).

Das Verhältnis der Zeit mit 5 V zu der Gesamtdauer einer Periode mit 5 V und 0 V wird als *Tastverhältnis* bezeichnet. Im Programm wird das Tastverhältnis durch einen Wert zwischen 0 und 255 angegeben. Eine 0 bedeutet, dass die Zeit mit 5 V 0% ausmacht, also liegt der Pin durchgängig auf einem 0 V-Potential. Eine 255 bedeutet, dass die Zeit mit 5 V 100% ausmacht, also liegt der Pin durchgängig auf einem 5 V-Potential. Diese beiden Werte entsprechen dem, was bei den bekannten Befehlen zur Steuerung von digitalen Pins passiert. Ein Wert von 100 bedeutet einen Anteil von $\frac{100}{255} \approx 0,39$ der Periodendauer. Daraus ergibt sich eine mittlere Spannung von $\bar{U} = 5 \text{ V} \cdot 0,39 \approx 1,96 \text{ V}$.



B 5.1. Darstellung des zeitlichen Verlaufs einer Pulsweitenmodulation mit einem Tastverhältnis von 25%.

5.1.1. RGB-Farben aus Hexadezimalzahlen erzeugen

Als wir uns das erste Mal die RGB-LED angeschaut haben (siehe S. 12), war die Anzahl der Farben, die wir erzeugen konnten, noch stark limitiert. Das ändert sich mit der Möglichkeit, die Farbanteile über PWM zu dimmen. Häufig werden die Farbanteile im RGB-Modell jedoch nicht in Dezimalzahlen zwischen 0 und 255 angegeben, sondern durch Hexadezimalzahlen - z. B. #CAFF70.

Frage: Wie kann man aus dem RGB-Hexadezimal-Code die PWM-Werte bestimmen?

Aufgabe 3: Hexadezimalzahlen

- Du kennst bereits das Dezimalsystem zur Basis 10 mit Ziffern von 0 bis 9 und das Binärsystem zur Basis 2 mit den Ziffern 0 und 1. Der RGB-Code nutzt eine Raute zu Anfang und danach das Hexadezimalsystem zur Basis 16. Wie viele Ziffern gibt es hier? Stelle anhand des Beispielcodes #CAFF70 eine Vermutung darüber auf, wie die Ziffern lauten.
- Der RGB-Code lässt sich in drei Zahlen zerlegen, die die Anteile von Rot, Grün und Blau angeben:

CA FF 70 .
Anteil Rot Anteil Grün Anteil Blau

Rechne die einzelnen Zahlen vom Hexadezimalsystem ins Dezimalsystem um.

Zur Erinnerung: Im Dezimalsystem bedeutet die Zahl 70: $7 \cdot 10^1 + 0 \cdot 10^0$. Übertrage diese Zerlegung für das Hexadezimalsystem. Übersetze die Buchstabenziffern vorher ins Dezimalsystem ($A_{16} = 10_{10}$, $B_{16} = 11_{10}, \dots$).

- Für Denker:* Wie viele verschiedene Farben kann man mithilfe des RGB-Farbcodes darstellen?

Aufgabe 4: RGB-Codes

- Bestimme anhand des RGB-Hexadezimal-Farbcodes die PWM-Werte für Rot, Grün und Blau in der folgenden Tabelle.
- Schließe eine RGB-LED mit Vorwiderstand an den Arduino an (siehe S. 12) und erzeuge nacheinander die Farben aus der Tabelle.

Farbe	Hex-Code	PWM-Werte	Farbe	Hex-Code	PWM-Werte
Blue-Violet	# 8A2BE2		OliveDrab	# 6B8E23	
Turquoise1	# 00F5FF		Khaki	# F0E68C	
Chocolate	# D2691E		DeepPink	# FF1493	

Projekt 2: Farbenspektakel mit RGB-Farbcodes

Lasse die RGB-LED mithilfe des Fadings (vgl. Aufgabe *Fading* zur PWM) möglichst viele Farbwerte durchlaufen und beobachte das Funkeln.

Das Hexadezimalsystem

Im Gegensatz zum Dezimalsystem zur Basis 10 mit Ziffern von 0 bis 9 und zum Binärsystem zur Basis 2 mit den Ziffern 0 und 1 arbeitet das Hexadezimalsystem mit der Basis 16 und den Ziffern 0, ..., 9, A, B, C, D, E, F. Die Buchstaben werden als Ziffern gewählt, weil bei der üblichen 10, 11, 12, ... nicht deutlich würde, dass es sich nur um eine Ziffer zur Basis 16 handelt.

Im Hexadezimalsystem wird jede Zahl durch die Potenzen von 16 zusammengesetzt. Die Hexadezimalzahl AF5 bedeutet dann so viel wie $A \cdot 16^2 + F \cdot 16^1 + 5 \cdot 16^0$. Den Wert der Hexadezimalzahl im Dezimalsystem erhält man dann durch eben diese Rechnung, wobei für die Buchstabenziffern die Dezimalzahlen eingesetzt werden müssen: $10 \cdot 16^2 + 15 \cdot 16^1 + 5 \cdot 16^0 = 2805$.

Da bei kleineren Ziffern wie 405 nicht ersichtlich ist, ob es sich um eine Hexadezimalzahl handeln soll, wird der Zahl häufig ein 0x vorangestellt: 0x405. Ebenfalls üblich ist es, die Basis im Index zu notieren: 405_{16} zeigt eine Hex-Zahl an; 405_{10} eine Dezimalzahl.

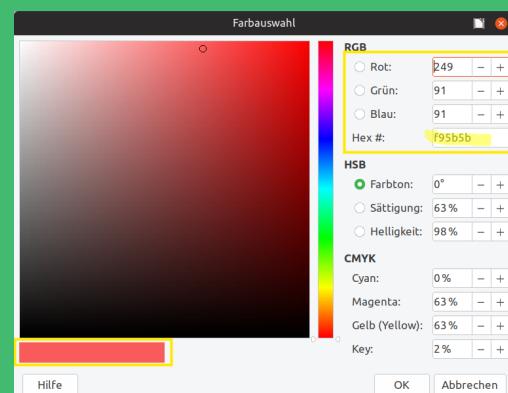
Der RGB-Farbcode

Der RGB-Farbcode setzt jede Farbe aus Rot, Grün und Blau zusammen, wobei jede Farbe einen unterschiedlichen Anteil an der Mischung haben kann. Der Anteil wird durch eine Zahl zwischen 0 und 255 ausgedrückt; es steht für jede Farbe also ein Byte (acht Bit) zur Verfügung.

Es wäre allerdings umständlich und für Menschen schlecht lesbar, für jede Farbe acht Nullen bzw. Einsen zu notieren. Die entsprechenden Dezimalzahlen wären zwar schon übersichtlicher, jedoch lassen sich Dezimalzahlen von digitalen Bauteilen schlechter verarbeiten, weil die 10 keine Zweierpotenz ist. Die Basis 16 schafft hier Abhilfe: Als vierte Potenz von Zwei ($16 = 2^4$) lassen sich die Ziffern zur Basis 16 durch genau vier Bit darstellen. Zum Beispiel ist $13_{10} = D_{16} = 1101_2$. Zweistellige Hex-Zahlen lassen sich dann durch acht Bit, also ein Byte darstellen. Der Wertebereich reicht von 00 (also 0) bis FF (also $15 \cdot 16 + 15 \cdot 1 = 255$) und passt damit genau zum RGB-Farbraum.

Um deutlich zu machen, dass die folgenden Hex-Zahlen eine Farbe im RGB-Code codieren, wird den drei Hex-Zahlen für die drei Farbanteile eine Raute vorangestellt. Aus den insgesamt sechs Ziffern, für die jeweils 16 mögliche Werte existieren, ergeben sich $16^6 = 16777216$ verschiedene Farben, die sich codieren lassen. Das sind mehr Farben als das menschliche Auge unterscheiden kann.

RGB-Farbcodes werden in zahlreichen Programmen wie zum Beispiel der Textverarbeitung LibreOffice genutzt (siehe Abbildung rechts). Im Internet findet man zudem [Farbtabellen mit der Angabe der Codes](#).



B 5.2. RGB-Farbcode in LibreOffice (rechts oben; die Angabe erfolgt über Dezimalzahlen oder Hexzahlen).

5.2. Spannung messen an analogen Eingängen

Wenn Batterien kaum noch Ladung gespeichert haben, lässt die Spannung an ihren Polen nach und sinkt unter den Wert, der auf der Batterie vermerkt ist. Mithilfe der analogen Eingänge A0 bis A5 lässt sich die Spannung messen und so entscheiden, ob die Batterie noch brauchbar ist.

Frage: Wie kann man mit dem Arduino eine Spannung messen?

Projekt 3: Batterietester (Voltmeter für $U < 5 \text{ V}$)

Für eine einfache Messung bei einer 1,5 V-Batterie wird der negative Pol der Batterie mit GND verbunden, sodass ein gemeinsames Nullpotential vorliegt. Der positive Pol der Batterie wird mit einem der analogen Eingänge A0 bis A5 verbunden. Über einen eingebauten Analog-Digital-Wandler (*engl. analog-to-digital converter, ADC*) wird der Spannungswert durch eine Zahl zwischen 0 und 1023 ausgedrückt.

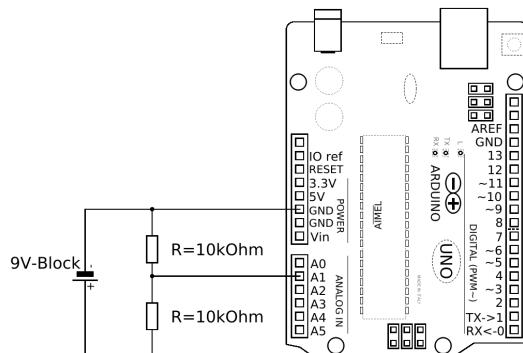
Analogwert	Spannung
0	0 V
1	
100	
1023	5 V

- Schließe eine mit 1,5 V beschriftete Batterie an den Arduino an und miss die Spannung. Berechne aus dem Analogwert die Spannung und lass sie auf dem seriellen Monitor ausgeben.
- Ergänze den Batterietester um eine Ampel, die anzeigt, ob die Batterie voll aufgeladen bzw. noch in Ordnung bzw. leer ist.
- Für Informatik-Profis:* Wie viel Bit Speicherplatz werden für den Analogwert zwischen 0 und 1023 gebraucht?

Projekt 4: Batterietester (Voltmeter für $U > 5 \text{ V}$)

Da der Arduino beim direkten Anschließen nur maximal 5V „verträgt“, muss man zum Testen von z.B. 9V-Blöcken weitere Bauteile verwenden. Mit zwei $10\text{k}\Omega$ Widerständen kann man einen einfachen *Spannungsteiler* aufbauen, der die Messung ermöglicht.

- Berechne die Stromstärke und die Spannung an den Widerständen. Warum sind große Widerstände hier sinnvoll?
- Markiere die Kabel in der Abbildung farbig, sodass die Kabel, die auf dem gleichen elektrischen Potential liegen, die gleiche Farbe haben. Notiere zudem den Wert des elektrischen Potentials.
- Gib an, wie man mit dem Arduino die Spannung am 9V-Block berechnet. Baue den Batterietester dann auf und probiere ihn mit dem 9V-Block aus der Box aus.
- Zum Nachdenken:* Wie groß darf die Spannung beim oben verwendeten Spannungsteiler maximal sein, damit am Arduino nicht mehr als 5V anliegen? Wie könnte man den Spannungsteiler bauen, sodass man Spannungen bis zu 15V messen kann?



5.2.1. Bedingungen durch logische Operationen verfeinern

Folie
öffnen

Beim Bau der Batterieampel kann die Verknüpfung von Wahrheitswerten hilfreich sein. Dazu dienen die Blöcke für die logischen Operationen **und**, **oder** und **nicht**, die aus zwei Bedingungen, die jeweils wahr oder falsch sein können, einen zusammenfassenden Wahrheitswert (wahr/falsch) machen bzw. den Wahrheitswert umkehren („NICHT“).

Frage: Wie ermittelt man das Ergebnis der logischen Operationen NICHT, UND, ODER?

Aufgabe 5: Genau draufgeschaut: Batterieampel

Wenn man die Batterieampel mit logischen Operationen realisiert, braucht man eine Bedingung der folgenden Art:

$$\text{spannung} > 1,2 \text{ V} \text{ UND } \text{spannung} < 1,4 \text{ V} \text{ ODER } \text{spannung} = 1,4 \text{ V}$$

Dies lässt sich bei mBlock auf zwei Arten realisieren, die sich in der Reihenfolge der Auswertung unterscheiden (siehe Abbildung).



Sind beide Varianten geeignet?

Zur Klärung dieser Frage eignen sich in einfachen Fällen **Wahrheitswerttabellen**, die einen Überblick über das Ergebnis der logischen Operationen bieten (siehe unten). Vergleiche die beiden Varianten mithilfe von Wahrheitswerttabellen.

Variante 1:

$U > 1,2 \text{ V}$	$U < 1,4 \text{ V}$	$U = 1,4 \text{ V}$	$U < 1,4 \text{ V ODER } U = 1,4 \text{ V}$	$U > 1,2 \text{ V UND } (U < 1,4 \text{ V ODER } U = 1,4 \text{ V})$	Bemerkung
1	1	1	1	1	phys. unmögl.
0	1	1	1
...

Variante 2:

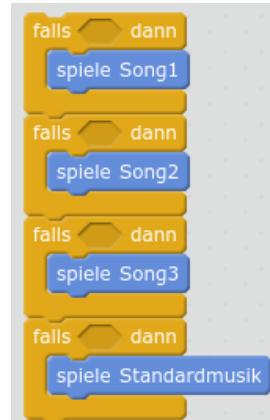
$U > 1,2 \text{ V}$	$U < 1,4 \text{ V}$	$U = 1,4 \text{ V}$	$U > 1,2 \text{ V UND } U < 1,4 \text{ V}$	$(U > 1,2 \text{ V UND } U < 1,4 \text{ V}) \text{ ODER } U = 1,4 \text{ V}$	Bemerkung
1	1	1	1	1	phys. unmögl.
0	1	1	0
...

Aufgabe 6: Juke-Box 2.0

Logische Operationen lassen sich nutzen, um die Juke-Box aus Kapitel 2 zu erweitern, ohne Hardware nachrüsten zu müssen. Zur Erinnerung: Es wurden zwei Taster und ein Piezo-Summer an den Arduino angeschlossen. Wenn Taster 1 gedrückt wurde, gibt der Befehl `lese digitalen Pin taster1` TRUE zurück und es wurde ein entsprechender Song gespielt.

Die Idee: Man kann auch beide Taster gleichzeitig oder gar keinen Taster drücken, sodass sich vier Fälle für vier Songs ergeben. Sinnvollerweise wird nur irgendeine Standardmusik gespielt, wenn gar kein Taster gedrückt wurde.

Formuliere für jeden der vier Fälle eine trennscharfe Bedingung!



Logische Operationen und Wahrheitswerttabellen

Logische Operationen dienen zum Verknüpfen von Wahrheitswerten - ganz so wie Rechenoperationen zum Verknüpfen von Zahlen dienen. Wir betrachten die logischen Operationen UND (AND), ODER (OR) sowie NICHT (NOT). Das Ergebnis dieser Operationen lässt sich anhand von Wahrheitswerttabellen übersichtlich festhalten. Darin wird festgehalten, ob zwei Aussagen bzw. Bedingungen A und B wahr (1) oder falsch (0) sind. In der rechten Spalte steht dann, ob die logische Operation wahr (1) oder falsch (0) ergibt.

		A UND B		A ODER B		NICHT A	
A	B						
1	1	1		1	1	0	
1	0	0		1	1	0	
0	1	0		1	1	1	
0	0	0		0	0	1	

Achtung: Die ODER-Operation ergibt auch dann „wahr“, wenn beide Aussagen wahr sind. Das aus dem Alltag bekannte „ENTWEDER-ODER“ (XOR) ist eine weitere logische Operation, die „falsch“ ergibt, wenn beide Aussagen wahr sind. Diese Operation ist aber nicht in mBlock enthalten.

5.3. Die Verwendung eines Potentiometers (Drehreglers)

Die Messung einer variablen, (quasi-)analogen Spannung eröffnet neue Möglichkeiten, da die Eingabewerte nun viel differenzierter sind als bei einem Taster, bei dem die Eingabe nur aus „0“ oder „1“ bestand. Zum Beispiel kann man darüber angeben, wie hell eine Lampe leuchten soll bzw. wie stark sie gedimmt werden soll. Dazu werden Potentiometer verwendet.

Frage: Wie funktioniert ein Potentiometer?

Aufgabe 7: Bleistiftpotentiometer

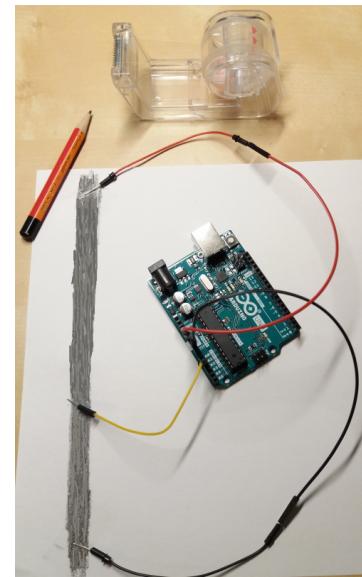
Ein einfaches Potentiometer kannst du selbst bauen.

Basteln: Markiere dafür mit Bleistift einen dicken Strich auf einem Blatt Papier und klebe am einen Ende ein Kabel fest, das mit GND verbunden ist. Klebe ans andere Ende ein Kabel, das mit 5V verbunden ist. Mit einem dritten Kabel („Sensorkabel“), das mit einem analogen Eingang verbunden ist, lässt sich nun messen, welches elektrische Potential an einer beliebigen Stelle des Bleistiftstreifens anliegt.

Experimentieren: Schreibe ein Programm, dass dir fortlaufend auf dem seriellen Monitor die Analogwerte und die umgerechneten Werte für das elektrische Potenzial bzw. die Spannung gegenüber GND anzeigt. Bewege dann das Sensorkabel über den Streifen und beobachte, wie sich die Spannungswerte verändern.

Analysieren: Der Bleistiftstreifen leitet den Strom bei einem bestimmten Gesamtwiderstand R_{ges} . Durch das Sensorkabel wird der Streifen in zwei Teile mit den Teilwiderständen R_1 und R_2 geteilt. Erläutere anhand deiner Beobachtungen, wie die drei Widerstände zusammenhängen.

Idee: Frick, Fritsch und Trick (2015): *Einführung in Mikrocontroller - Der Arduino als Steuerzentrale*, Bad Saulgau



Potentiometer

Ein **Potentiometer**, kurz: Poti, ist im Grunde nichts anderes als ein Spannungsteiler mit zwei Widerständen. Jedoch kann die Größe der Widerstände z. B. durch Drehen variiert werden. Der Gesamtwiderstand bleibt dabei immer gleich.



Beim Anschluss an den Arduino wird der mittlere Pin des Potentiometers an einen analogen Eingang angeschlossen. Die anderen beiden Pins werden mit GND und 5V verbunden.

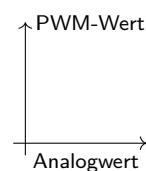
Projekt 5: Dimmbare Lampe

Baue und programmiere eine Lampe, deren Helligkeit sich durch ein Potentiometer einstellen lässt.

Hinweis: Du musst dafür sorgen, dass der eingelesene Analogwert zwischen 0 und 1023 in einen PWM-Wert zwischen 0 und 255 umgerechnet wird. Ermittle dazu eine passende Funktion.

Folie
öffnen

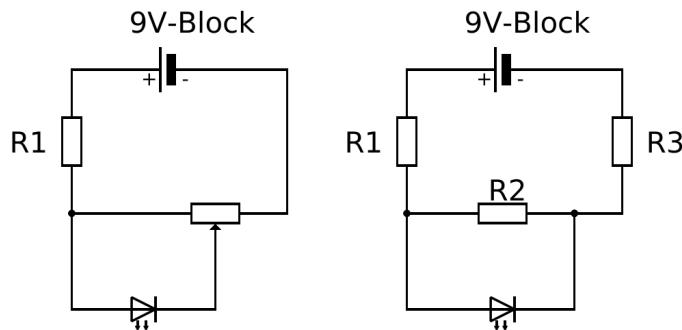
Neues
Werkzeug:
Joystick



5.3.1. Die Verwendung eines Potentiometers ohne Mikrocontroller

Für einige Projekte, wie das Dimmen einer Lampe, ist ein Mikrocontroller eigentlich überdimensioniert, weil sich die Funktion schon durch eine reine Hardwarelösung erreichen lässt.

Frage: Wie lässt sich eine Lampe ohne Mikrocontroller dimmen?



B 5.3. Auf der linken Seite ist die Anwendung eines Potentiometers ohne Mikrocontroller dargestellt. Auf der rechten Seite ist der zugehörige Ersatzschaltplan gezeichnet, der zeigt, dass das Potentiometer als Spannungsteiler mit zwei variablen Widerständen R_2 und R_3 aufgefasst werden kann.

Projekt 6: Dimmbare Lampe ohne Mikrocontroller

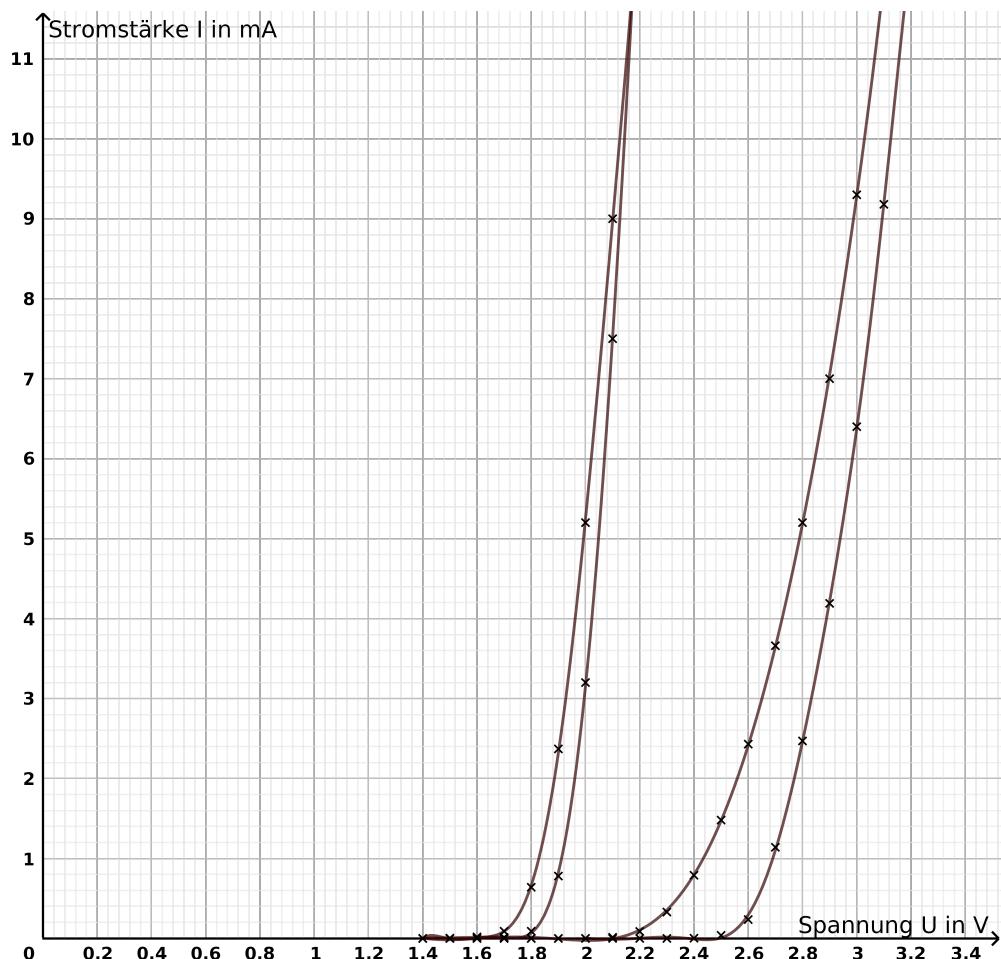
- Erkläre, wie sich die LED verhält, wenn das Potentiometer so gedreht ist, dass gilt:
 - $\dots R_2 = 0 \Omega, R_3 = 10 \text{ k}\Omega,$
 - $\dots R_2 = 10 \text{ k}\Omega, R_3 = 0 \Omega.$
- Bevor die Schaltung aufgebaut werden kann, muss die Größe des Vorwiderstands R_1 berechnet werden. Der Vorwiderstand muss den Strom auch dann noch klein genug halten, wenn das Potentiometer so gedreht ist, dass gilt: $R_3 = 0 \Omega$ (und dementsprechend $R_2 = 10 \text{ k}\Omega$). Berechne R_1 so, dass die Stromstärke durch die LED maximal 20 mA beträgt.
Hinweis: Die Stromstärke durch R_2 kann vernachlässigt werden, sodass $I_{ges} \approx I_{LED} = 20 \text{ mA}$ mit $U_{LED} = 2,3 \text{ V}$ gilt.
Begründung: Wenn $R_1 = 0 \Omega$ wäre, würde die komplette Spannung an R_2 abfallen. Dann gilt: $I_{R_2} = \frac{9 \text{ V}}{10 \text{ k}\Omega} = 0,9 \text{ mA}$. Die Stromstärke durch R_2 beträgt also nur etwa 1/20 der Stromstärke durch die LED und wenn R_1 größer wird, dann wird die Stromstärke durch R_2 sogar noch kleiner.
- Baue die Schaltung auf und beobachte das Leuchtverhalten der LED beim Drehen am Potentiometer.

Beim Experimentieren mit dem Potentiometer wirst du feststellen, dass man das Potentiometer nicht vollständig zur Seite drehen muss, damit die LED aufhört zu leuchten. Im Experiment zeigt sich, dass eine blaue LED schon bei $R_2 = 2,5 \text{ k}\Omega$ und $R_3 = 7,5 \text{ k}\Omega$ aufhört zu leuchten. Die Spannung an der LED beträgt dann $U_{L,blau} = 2,3 \text{ V}$.

Aufgabe 8: Kennlinien von Leuchtdioden

- Baue eine blaue LED in den oben dargestellten Schaltkreis und drehe das Potentiometer so, dass die blaue LED gerade nicht mehr leuchtet. Die Spannung an der blauen LED beträgt dann $U_{L,blau} = 2,3 \text{ V}$. Ersetze dann die blaue LED durch eine grüne/gelbe/rote LED. Notiere deine Beobachtungen.
- Ordne die Farben der LEDs den rechts abgebildeten Kennlinien von einer blauen, einer grünen, einer gelben und einer roten LED zu. Experimentiere dazu mit dem Potentiometer und den LEDs.

Hinweis: Das menschliche Auge ist in der Lage, bereits bei einer Stromstärke von wenigen Mikroampere ein schwaches Leuchten zu erkennen. Im Diagramm ist so eine geringe Stromstärke kaum von 0 mA zu unterscheiden.



B 5.4. U-I-Kennlinien einer roten, gelben, grünen und blauen Leuchtdiode.

5.4. Helligkeit messen

Die Helligkeit bestimmt unseren Tages- und Jahresrhythmus: Wenn es dunkel wird, schlafen wir (oder gehen feiern) und wenn es hell wird, stehen wir wieder auf und unternehmen etwas. Es ist daher nur logisch, dass es einige Anwendungen für elektrische Schaltungen gibt, die auf die Helligkeit reagieren.¹ In einfachen Fällen wird dabei auf einen Fotowiderstand, kurz: LDR (engl. *light dependent resistor*), zurückgegriffen.

Frage: Wie verwendet man einen Fotowiderstand/LDR am Arduino?

Aufgabe 9: Erste Experimente mit dem LDR

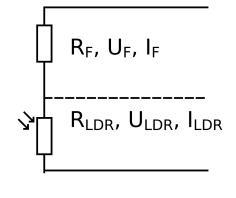
- Baue mit dem Arduino einen Spannungsteiler mit einem LDR und einem Festwiderstand von $R_1 = 10\text{ k}\Omega$ so auf, dass du die Spannung am LDR in A0 messen kannst. Lasse dir die Spannung am LDR auf dem seriellen Monitor ausgeben.
- Beschreibe, wie sich die Spannung am LDR verhält, wenn es dunkel bzw. wenn es hell wird.



Aufgabe 10: Genaue Analyse des Spannungsteilers

Die Änderung der Spannung resultiert aus der Änderung des Widerstands des LDR. Um einen Eindruck vom Wertebereich des Widerstands eines LDR zu bekommen, soll dieser nun berechnet werden.

- Leite eine Formel für den Spannungsteiler her, mit der du den Widerstand R_2 des LDR mithilfe der Spannung U_2 am LDR, dem Festwiderstand R_1 und der Spannung U_1 am Festwiderstand berechnen kannst. Tipp: Betrachte zuerst die Stromstärken I_1 und I_2 durch den Festwiderstand und den LDR. Durch das Sensorkabel fließt (näherungsweise) kein Strom.
- Berechne, welchen Widerstand der LDR hat, wenn er komplett abgedunkelt ist und wenn er mit einer Smartphone-Taschenlampe bestrahlt wird.

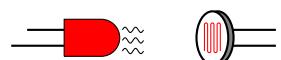


Projekt 7: Straßenlampe

Baue eine Straßenlampe, deren Licht (Vorwiderstand!) angeht, wenn es dunkel wird, und ausgeht, wenn es hell wird.

Projekt 8: Alarmanlage mit Lichtschranke

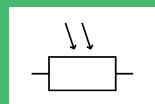
Baue eine Alarmanlage, indem du mit einer LED (Vorwiderstand!) und einem LDR eine Lichtschranke baust. Wird diese unterbrochen, soll ein akustischer Alarm ertönen.



¹Natürlich lässt er sich auch für Spielereien nutzen. Der Author hat mit einigen LDR ein Moorhuhn-Lasertag gebaut...

Fotowiderstand

Ein **Fotowiderstand**, kurz: **LDR** (*engl. light dependent resistor*), ist ein lichtabhängiger Widerstand. Wenn es dunkel wird, wird der elektrische Widerstand des LDR größer; wenn es hell wird, wird der elektrische Widerstand des LDR kleiner.



Wenn man die Helligkeit wie bisher über eine Spannung messen kann, kann man schon einige interessante Projekte umsetzen. Es wäre jedoch unbefriedigend, wenn man nicht auch die Helligkeit an sich berechnen könnte. Dies kann man zum Beispiel für eine Wetterstation nutzen.

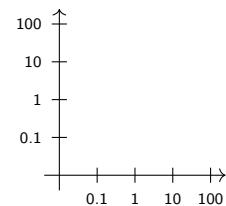
Frage: Wie kann man aus dem Widerstand des LDR die Helligkeit berechnen?

Der Zusammenhang zwischen Widerstand des LDR und der Umgebungshelligkeit ließe sich zwar theoretisch herleiten, aber das würde an dieser Stelle zu weit führen. Wir können die Formel zur Berechnung der Umgebungshelligkeit auch experimentell gewinnen. Ein erster Ansatz dazu wäre ein kontrolliertes Experiment, in dem der Widerstand des LDR bei definierter Umgebungshelligkeit gemessen wird. Leider bräuchten wir zur Festlegung der Umgebungshelligkeit aber bereits ein Messgerät für die Helligkeit. Daher entnehmen wir die zusammengehörigen Messwerte von Widerstand und Helligkeit nicht einem Experiment, sondern dem Datenblatt des LDR. Ein Datenblatt enthält in tabellarischer und oft graphischer Form die wesentlichen Kenndaten eines Bauteils. Es ist zu jedem Bauteil kostenlos im Internet zu finden.

Aufgabe 11: Datenblatt lesen

Suche im **Datenblatt des LDR** den Graphen, der den Zusammenhang von Helligkeit in der Einheit Lux und Widerstand des LDR in der Einheit $k\Omega$ abbildet. Entnimm dem Graphen fünf zusammengehörige Werte von Helligkeit und Widerstand und halte diese tabellarisch fest.

Achtung: Die Achsen im Graphen sind logarithmisch skaliert. Das bedeutet, dass die Werte an den Achsen nicht gleichmäßig zunehmen, sondern exponentiell (von 0,1 zu 1 zu 10 zu 100 zu 1000). Diese Skalierung ermöglicht erst das Ablesen der Werte, aber es muss berücksichtigt werden, dass die Werte nur sehr ungenau abzulesen sind.



Das verlinkte Datenblatt ist evtl. nicht das korrekte Datenblatt zu dem LDR. Da die Bauteilnummer bei dem verwendeten Starter Kit nicht angegeben wird, ist eine Zuordnung leider nicht mehr möglich.

Aufgabe 12: Regression durchführen

Unabhängig davon, ob man die Daten aus einem Experiment oder dem Datenblatt gewonnen hat, lässt sich nun eine Regression durchführen, um den allgemeinen Zusammenhang zwischen Widerstand des LDR und Helligkeit herauszufinden.

Führe mit den ermittelten Werten eine Regression durch, indem du die unten abgebildete Anleitung befolgst. Bestimme mit der erhaltenen Funktion die Umgebungshelligkeit im Raum sowie die Helligkeit deiner Smartphone-Taschenlampe auf niedrigster und höchster Stufe.

Eine Regression durchführen

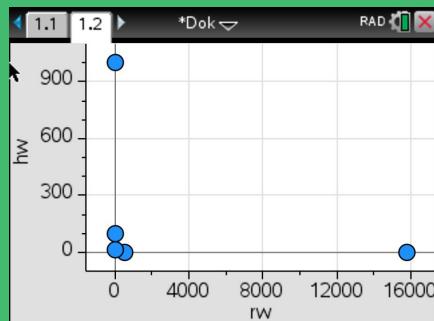
Beim Durchführen einer Regression wird diejenige Funktion(sgleichung) ermittelt, die am besten zu den gegebenen Daten passt. Die Art der Funktion muss jedoch vom Anwender sinnvoll festgelegt werden.

Regression mit TI Nspire

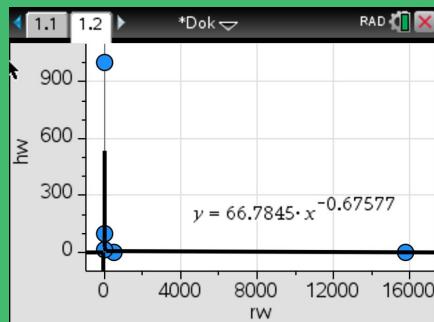
1. Erstelle ein neues Dokument mit einer Seite „Lists & Spreadsheet“. Benenne eine Spalte als *rw* (Werte für R, also der Widerstand) und eine Spalte als *hw* (Werte für die Helligkeit). Ergänze die Werte.

	A rw	B hw	C	D
1	15800	0.1		
2	500	1		
3	16	10		
4	0.5	100		
5	0.02	1000		
B5	1000			

2. Füge eine neue Seite „Data & Statistics“ hinzu (mit *ctrl → +page*). Da wir die Helligkeit in Abhängigkeit vom Widerstand berechnen wollen, kommt der Widerstand auf die Rechtsachse und die Helligkeit auf die Hochachse.



3. Führe eine Regression durch (*menu → 4: Analysieren → 6: Regression*). Welche Funktionsklasse könnte zu der Verteilung der Werte passen? Falls die ausprobierte Funktionsklasse nicht zu den Werten passt, mache die Regression rückgängig (*ctrl → esc*) und probiere eine andere Funktionsklasse.
Achtung: Durch die geringe Auflösung des Taschenrechners können auch passende Funktionen ggf. falsch aussehen.



4. Übersetze die Funktionsgleichung in den physikalischen Zusammenhang. Für den Widerstand ist das Formelzeichen *R* festgelegt. Für die Helligkeit wählen wir an dieser Stelle *H*.

$$y = 66,78 \cdot x^{-0,66}$$

$$\downarrow \qquad \qquad \downarrow$$

$$H = 66,78 \cdot R^{-0,66}$$

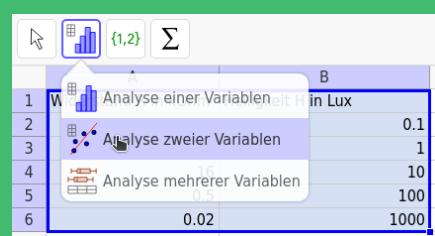
$$R \text{ in } k\Omega, H \text{ in Lux}$$

Regression mit Geogebra Classic

1. Starte Geogebra Classic und wähle als Perspektive die Tabellenkalkulation. Übertrage die Daten in die Tabellenkalkulation.

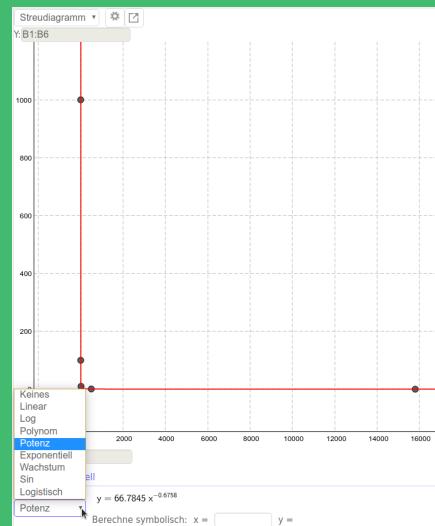
	A	B
1	Widerstand R in kOhm	Helligkeit H in Lux
2	15800	0.1
3	500	1
4	16	10
5	0.5	100
6	0.02	1000
7		

2. Markiere alle Daten und wähle das Werkzeug Analyse zweier Variablen. Die Daten aus Spalte A werden automatisch als x-Koordinate gewählt, die aus Spalte B als y-Koordinate. Bei Bedarf kann dies mit $X \rightleftarrows Y$ vertauscht werden (oben rechts).



3. Führe eine Regression durch, indem du unten links ein passendes Regressionsmodell wählst. Welche Funktionsklasse könnte zu der Verteilung der Werte passen? Falls die ausprobierte Funktionsklasse nicht zu den Werten passt, probiere eine andere Funktionsklasse.

Hinweis: Die Anzahl der Nachkommastellen lässt sich in den Einstellungen unter „Runden“ ändern.



4. Übersetze die Funktionsgleichung in den physikalischen Zusammenhang. Für den Widerstand ist das Formelzeichen R festgelegt. Für die Helligkeit wählen wir an dieser Stelle H .

$$y = 66,78 \cdot x^{-0,66}$$

$$\downarrow \qquad \qquad \downarrow$$

$$H = 66,78 \cdot R^{-0,66}$$

R in $k\Omega$, H in Lux

5.5. Temperatur messen

Nicht nur die Helligkeit beeinflusst unseren Alltag, sondern auch die Temperatur. Ganz allgemein ist die Temperatur eine wichtige Größe, die bei vielen Anwendungen eine Rolle spielt und daher erfasst und automatisiert in die Anwendung einfließen sollte: Thermostate regeln die Temperatur im Raum, Wetterstationen geben die Temperatur an und 3D-Drucker regeln die Temperatur der Düse auf eine festgelegte Temperatur, damit der Kunststoff flüssig wird, aber immer noch zäh genug bleibt, um die Figur zu bilden. Häufig wird dabei ein Heißleiter (kurz: NTC, von engl. *negative temperature coefficient*) verwendet - ein elektrischer Widerstand, der auf die Temperatur reagiert.



B 5.5. Ein NTC.

Frage: Wie verwendet man einen NTC am Arduino?

Aufgabe 13: Erste Experimente mit dem NTC

- Baue mithilfe eines Festwiderstands $R_F = 10\text{ k}\Omega$ und dem NTC einen Spannungsteiler und lies die Spannung am NTC in A0 aus (genau wie beim LDR). Erwärme den NTC, indem du ihn zwischen Daumen und Zeigerfinger hälst. Beschreibe, wie sich die Spannung am NTC ändert, wenn dieser wärmer wird.
- Begründe, dass auch hier gilt:

$$\frac{R_{NTC}}{R_F} = \frac{U_{NTC}}{U_F}.$$

Begründe anhand der Formel, wie sich der Widerstand am NTC ändert, wenn dieser wärmer wird.

Neues Werkzeug:
Temperatur- und Luftfeuchtigkeitssensor

Projekt 9: Digitales Thermometer

Baue ein digitales Thermometer, das die Lufttemperatur im Raum auf dem seriellen Monitor anzeigt!

Führe dazu mithilfe des rechts abgebildeten Ausschnitts aus einem Datenblatt eine Regression durch.

Das verlinkte Datenblatt ist evtl. nicht das korrekte Datenblatt zu dem NTC. Da die Bauteilnummer bei dem verwendeten Starter Kit nicht angegeben wird, ist eine Zuordnung leider nicht mehr möglich.

R/T No. 7003

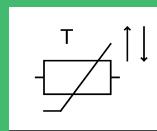
Widerstand bei 25° :

$R_{25} = 10\text{ k}\Omega$.

T (C)	R_T/R_{25}	(%/K)
5.0	2.3311	4.5
10.0	1.8684	4.4
15.0	1.5075	4.2
20.0	1.224	4.1
25.0	1.0000	4.0
30.0	0.82176	3.9

Heißleiter

Ein **Heißleiter**, kurz: **NTC** (*engl. negative temperature coefficient*), ist ein temperaturabhängiger Widerstand. Wenn es wärmer wird, wird der elektrische Widerstand des NTC kleiner; wenn es kälter wird, wird der elektrische Widerstand des NTC größer.



Anmerkung:

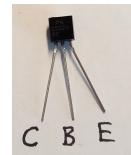
Es gibt auch Kaltleiter, kurz: **PTC** (*engl. positive temperature coefficient*), die ihren Widerstand verringern, wenn es kälter wird, und erhöhen, wenn es wärmer wird. Zusammen genommen bezeichnet man NTC's und PTC's auch als Thermistoren, also als temperaturabhängige Widerstände (*engl. thermally sensitive resistor*).

5.6. Exkurs: Schaltungen mit Transistoren vereinfachen

Insbesondere die „Straßenlampe“ benötigt nur ein sehr simples Programm in der Form WENN - DANN - SONST. Für solche Fälle ist der Arduino eigentlich eine überdimensionierte Lösung - viel einfacher, jedenfalls in Bezug auf die Anzahl der Bauteile, ist die Umsetzung dieser Schaltung mithilfe eines Transistors. Dieser ist (unter anderem) ein elektronischer Schalter, mit dem sich das WENN - DANN - SONST - Verhalten ganz ohne Programm umsetzen lässt.

Frage: Wie verwendet man einen Transistor?

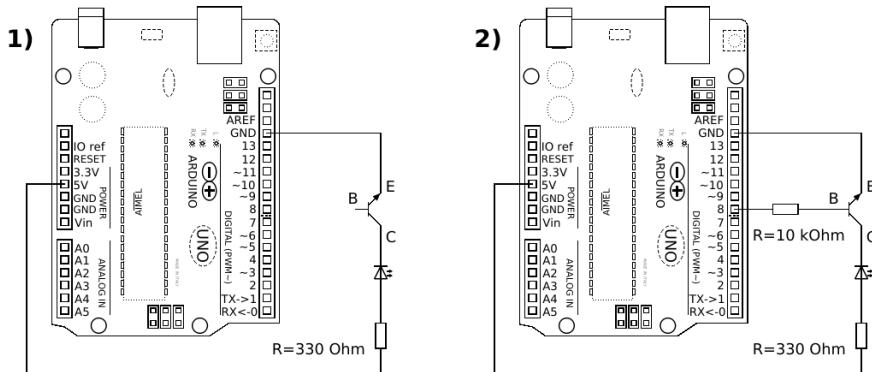
Ein Transistor hat drei Anschlüsse, die als Kollektor (**C** von engl. *collector*), Basis (**B**) und Emitter (**E**) bezeichnet werden. Wenn man auf die abgeflachte Seite des Transistors schaut, sind die drei Pins in der genannten Reihenfolge angeordnet. Im Folgenden geht es zunächst um deren Grundfunktionen.



Aufgabe 14: Digitalpins verstehen

Befolge die unten angegebenen Schritte und stelle Schlussfolgerungen über die Funktionsweise eines Transistors an.

- Baue die unten abgebildeten Schaltungen nacheinander auf. Spiele für die zweite Schaltung ein einfaches Blink-Programm auf den Arduino.

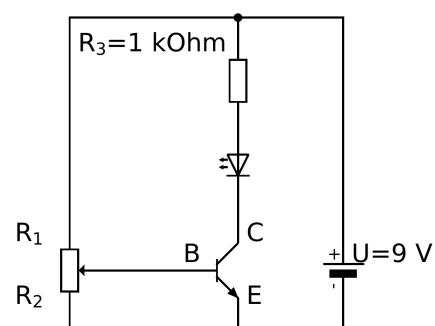


- Ersetze den $10\text{ k}\Omega$ Widerstand durch einen $100\text{ k}\Omega$ Widerstand.

Aufgabe 15: Vermessung

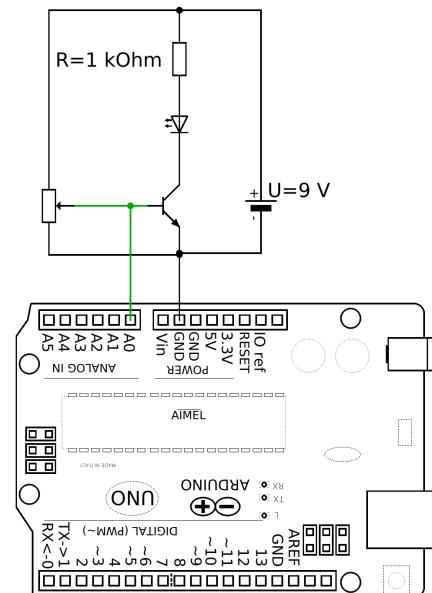
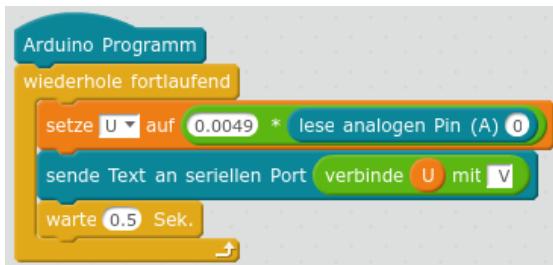
Um den Transistor zielgerichtet nutzen zu können, muss man die Spannung U_{BE} zwischen Basis und Emitter kennen, bei der der Transistor anfängt, durchzuschalten. Dazu dient die rechts abgebildete Schaltung.

Das Potentiometer lässt sich wieder in zwei Teilwiderstände R_1 und R_2 zerlegen, an denen die Spannung U_1 bzw. U_2 abfällt. Erkläre, wie der Widerstand R_2 und die Spannung U_{BE} zusammenhängen.



Baue die Schaltung nun auf. Um die Spannung U_{BE} messen zu können, wird ein Arduino ergänzt, der die Spannung in A0 ausliest und auf dem seriellen Monitor ausgibt.

Bestimme so die Grenzspannung U_{BE} , ab der der Transistor anfängt zu schalten, sodass die LED leuchtet.



Projekt 10: Straßenlampe ohne Mikrocontroller

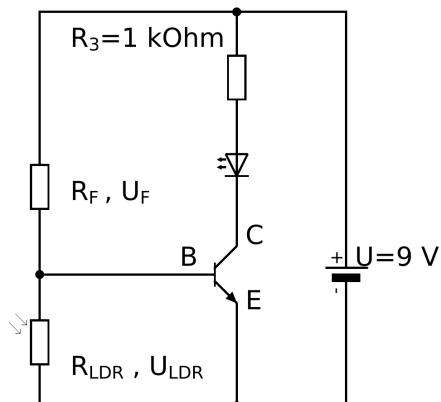
Nun kann die Straßenlampe auch ohne Arduino realisiert werden. Dazu wird statt des Potentiometers ein Spannungsteiler mit einem LDR und einem Festwiderstand aufgebaut (siehe Schaltplan rechts).

Bestimme die Größe des Festwiderstands R_F so, dass der Transistor schaltet, wenn die Größe des LDR $R_{LDR} = 7\text{ k}\Omega$ beträgt.

Tipps:

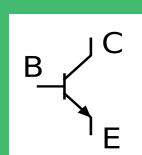
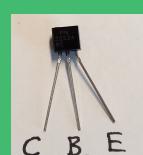
- Nutze die Spannungsteilerformel.
- Nutze $U_{LDR} = U_{BE}$. Ab welcher Spannung schaltet der Transistor?

Baue die Schaltung danach auf und teste sie.



Der Transistor

Ein Transistor hat drei Anschlüsse, die als Kollektor (**C** von engl. *collector*), Basis (**B**) und Emitter (**E**) bezeichnet werden. Wenn man auf die abgeflachte Seite des Transistors schaut, sind die drei Pins in der genannten Reihenfolge angeordnet.



Transistoren dienen als elektronische Schalter oder Verstärker (letzteres wird im Kapitel 7 zu Motoren genutzt). Als Schalter lassen sie sich nutzen, weil die Strecke vom Kollektor zum Emitter ohne Weiteres nicht leitet. Erst wenn zwischen Basis und Emitter eine Spannung $U_{BE} \approx 0.6\text{ V}$ anliegt, fließt zwischen Basis und Emitter ein schwacher Strom, der den Transistor mit Elektronen flutet und es dadurch ermöglicht, dass zwischen Kollektor und Emitter ein starker Strom fließen kann.

Die Möglichkeit, mit Transistoren automatisierte Schalter herzustellen und dadurch Programme physikalisch abzubilden, macht Transistoren zur Grundlage von Mikrocontrollern und Computern und damit zu einer der wichtigsten Erfindungen des 20. Jahrhunderts. Schon auf dem kleinen integrierten Schaltkreis des Arduino, dem ATMEGA328P, sind Millionen von Transistoren verbaut. Wenn ein Digitalpin des Arduino auf HIGH gestellt wird, dann wird intern ein Transistor geschaltet.

Es gibt verschiedene Bauarten für Transistoren. Im hier verwendeten Starter Kit sind zwei npn-Transistoren (PN2222) vorhanden, was bedeutet, dass darin zwei n-dotierte und eine p-dotierte Schicht in der Mitte verbaut sind. npn-Transistoren müssen mit einer n-Schicht (normalerweise der Emitter) mit GND verbunden sein.

5.7. Das EVA-Prinzip

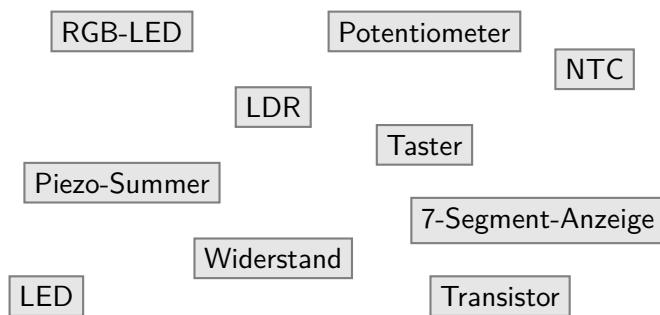
Allmählich hat sich eine Vielzahl an elektrischen Bauteilen angesammelt, die in diesem Skript genutzt wurden. Um nicht den Überblick zu verlieren, wären Kategorien praktisch, mit denen man Bauteile und technische Systeme im Allgemeinen einordnen kann.

Frage: Wie lassen sich elektrische Bauteile und technische Systeme kategorisieren?

 Folie

öffnen

Aufgabe 16: Du hast bisher (mindestens) folgende Bauteile verwendet:



Ordne sie nach ihrer Funktion in vier Gruppen.

Bauteilkategorien

Elektrische Bauteile lassen sich grob in vier Gruppen einordnen:

- **Aktive Bauteile** übernehmen in der Schaltung eine steuernde Funktion; sie können elektrische Signale umwandeln oder modifizieren. Unter den aktiven Bauteilen haben zwei Gruppen eine besondere Bedeutung:
 - **Sensoren** (auch Fühler genannt) sind elektrische Bauteile, die eine physikalische

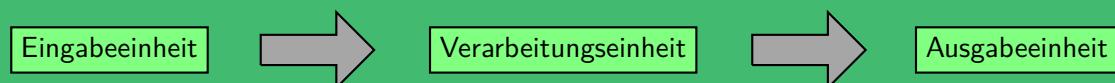
Größe aus der Umwelt (Temperatur, Helligkeit, Luftdruck oder auch ein mechanischer Druck mit dem Finger) in eine elektrische Größe (Widerstand, Spannung, elektrisches Potential, Stromstärke) umwandeln. Dadurch werden die physikalischen Größen aus der Umwelt einer elektronischen Verarbeitung zugänglich.

- **Aktoren** (auch Aktuatoren genannt) sind elektrische Bauteile, die eine elektrische Größe in eine mechanische (Bewegung, Schallwellen) oder andere Größe (Temperatur, Licht, ...) umwandeln. Sie ermöglichen, dass die elektronische Verarbeitung zu Handlungen bzw. Konsequenzen führen kann.
- **Passive Bauteile** haben eine feste Größe, die sich nicht ändert (z. B. ein ohmscher Widerstand). Sie sind nicht nur unabhängig von physikalischen Größen in der Umgebung, sondern auch von den elektrischen Größen (z. B. ein ohmscher Widerstand hat unabhängig von der anliegenden Spannung den gleichen Wert).

Abgesehen von der bisweilen nicht ganz eindeutigen Zuordnung zu einer Bauteilgruppe wird hier ein allgemeines Prinzip deutlich, das sich eignet, um technische, aber auch biologische Systeme zu beschreiben.

Das EVA-Prinzip

Technische Systeme lassen sich nach ihrer Funktion in drei Einheiten zerlegen: Eingabeeinheit, Verarbeitungseinheit, Ausgabeeinheit.



Mit dem EVA-Prinzip wird die grundlegende Reihenfolge der Verarbeitung von Daten charakterisiert. Die Einheiten bestehen dabei nicht nur aus den Bauteilen, sondern beinhalten auch die Art der Verarbeitung, also zum Beispiel das Programm auf dem Arduino. Insbesondere sind die Einheiten nicht gleichbedeutend mit den oben genannten Bauteilgruppen!

Beispiel Dämmerungsschaltung:

Der Spannungsteiler von Festwiderstand und LDR ermöglicht die Eingabe von Daten zur Helligkeit, also kann dies als Eingabeeinheit angesehen werden. Auf dem Arduino werden die elektrischen Signale entsprechend des laufenden Programms verarbeitet; dies ist die Verarbeitungseinheit. Letztlich erfolgt die Ausgabe durch das Leuchten einer LED, wenn es dunkel ist, bzw. durch das Nicht-Leuchten der LED. Die LED mit zugehörigem Vorwiderstand kann als Ausgabeeinheit angesehen werden.

Beispiel Mensch:

Unsere Sinne (Augen zum Sehen, Ohren zum Hören, ...) bilden die Eingabeeinheiten des Systems Mensch. Im Gehirn und den weiteren Nervenbahnen im Körper werden die Signale verarbeitet. Dies ist die Verarbeitungseinheit des Menschen. Schließlich kommt es zum Beispiel zu einer Bewegung

(Musik leiser drehen, Augen zukneifen, sprechen mit dem Mund ...) - dies gehört zu den Ausgabenheiten des Menschen.

Aufgabe 17: *Kleines Begriffstraining*

- a) Kategorisiere die *Dimmbare Lampe* (s. S. 41) nach dem EVA-Prinzip.
- b) Kategorisiere das *Digitalthermometer* (s. S. 48) nach dem EVA-Prinzip.



Neues

Werkzeug:

Infrarot-

Fernbedienung

5.8. Vermischte Übungen

Aufgabe 18: Pulsweitenmodulation

- Berechne die mittlere Spannung, die mit dem Befehl `setze PWM-Pin 5 Ausgang auf 138` ausgegeben wird.
- Mit dem in a) genannten Befehl wird eine Pulsweitenmodulation durchgeführt. Erkläre, was darunter zu verstehen ist.
- Jannik meint: „Mit dem Befehl in a) kann ich eine blaue LED auch ohne Vorwiderstand betreiben, denn die halten die berechnete Spannung aus.“ Nimm dazu Stellung.

Aufgabe 19: Hexadezimalzahlen und RGB-Code

- Eine Farbe lässt sich im RGB-Farbcodes zum Beispiel durch #10FFC7 codieren. Erläutere, wie dieser Code aufgebaut ist.
- Berechne die Dezimalzahlen zu dem RGB-Code aus a).
- Die Dezimalzahlen lassen sich als PWM-Werte nutzen, wenn man eine RGB-LED am Arduino anschließt. Bestimme anhand des RGB-Hexadezimal-Farbodes die PWM-Werte für Rot, Grün und Blau in der folgenden Tabelle.

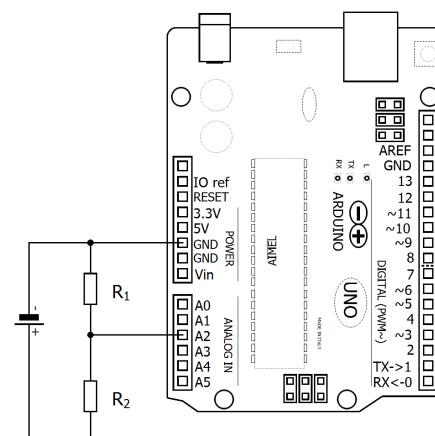
Farbe	Hex-Code	PWM-Werte	Farbe	Hex-Code	PWM-Werte
NavyBlue	# 000080		Aquamarine	# 7FFF00	
SandyBrown	# F4A460		SeaGreen	# 2E8B57	
Coral	# FF7F50		DarkOrchid	# 9932CC	

Zur Kontrolle: www.farb-tabelle.de

Aufgabe 20: Spannungsmessung

Mit der rechts abgebildeten Schaltung sollen am Arduino Spannungen an der Batterie bis zu 15 V gemessen werden.

- Nenne mögliche, sinnvolle Größen für die Widerstände R_1 und R_2 .
- Im analogen Eingang A2 wird ein Wert von 789 gemessen. Berechne die Spannung an der Batterie.

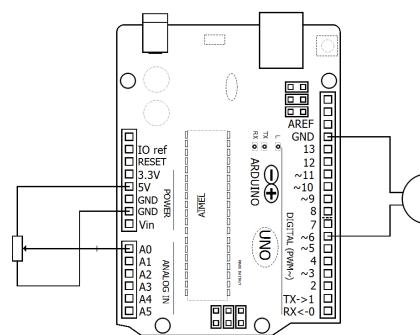


Aufgabe 21: *Potentiometer*

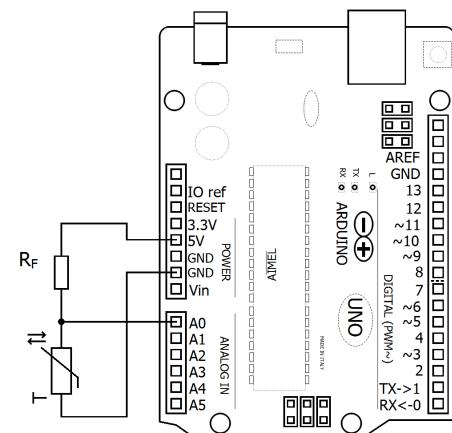
- Erläutere die Funktionsweise eines Potentiometers und nenne ein Einsatzbeispiel.
- Skizziere, wie man ein Potentiometer am Arduino anschließt.
- Ein Potentiometer hat einen Gesamtwiderstand von $R_{ges} = 10\text{k}\Omega$. Der mittlere Kontakt wird im analogen Eingang A0 ausgelesen und liefert einen Analogwert von 824. Berechne, wie groß die Teilwiderstände sind.

Aufgabe 22: *Dimmbarer Lautsprecher*

Der Schaltplan rechts zeigt ein Potentiometer, dessen mittlerer Kontakt am analogen Eingang A0 eines Arduino angeschlossen ist. Auf der anderen Seite ist ein Piezo-Summer an Digitalpin 6 des Arduino angeschlossen. Entwickle mit den unten abgebildeten Befehlen ein Programm, das dafür sorgt, dass die Lautstärke des Piezo-Summers durch das Potentiometer gedimmt werden kann. Das Programm soll in einem Struktogramm dokumentiert werden.

**Aufgabe 23:** *LDR und NTC - Basics*

- Nenne jeweils einen Einsatzzweck für einen LDR und einen NTC.
- Beschreibe das Widerstandsverhalten eines LDR (eines NTC), wenn sich die Helligkeit (die Temperatur) verringert.
- Ein NTC ist in einem Spannungsteiler mit einem Festwiderstand mit $R_F = 10\text{k}\Omega$ am Arduino angeschlossen (s. Schaltplan rechts). Im analogen Eingang A0 wird ein Wert von 643 gemessen. Berechne die Größe des Widerstands des NTC.



- d) Die Tabelle unten zeigt für den verwendeten NTC, welche Widerstandswerte R zu welcher Temperatur T gehören. Bestimme mit Hilfe einer quadratischen Regression einen funktionalen Zusammenhang zwischen R und T und berechne damit die Temperatur, die zum Widerstandswert aus Aufgabenteil c) gehört.

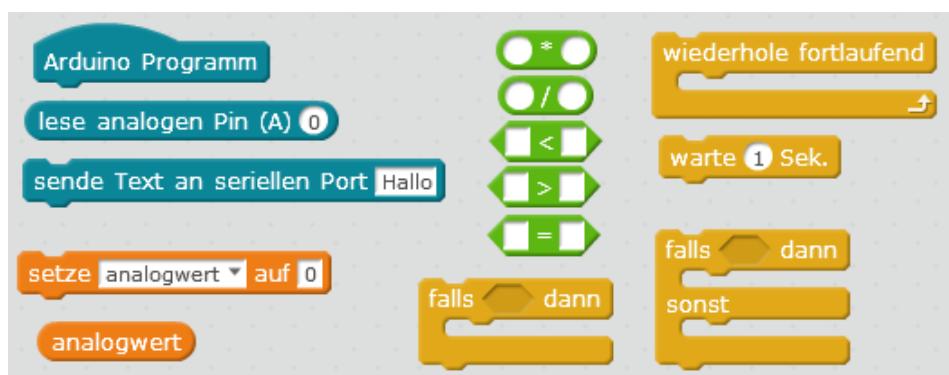
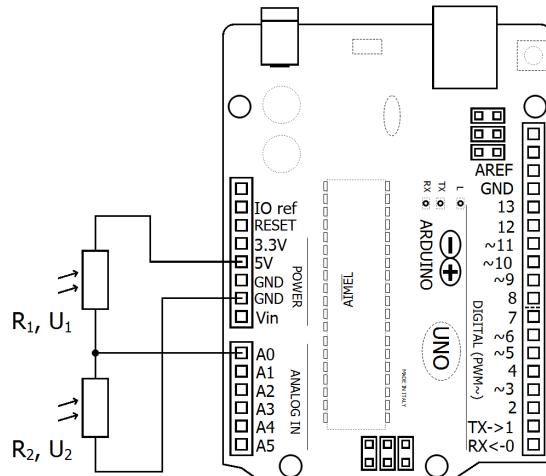
R/T No. 8307

	T (C)	R_T/R_{25}
Widerstand bei 25°:	5.0	2.252
	10.0	1.8216
	15.0	1.4827
$R_{25} = 10\text{ k}\Omega$.	20.0	1.2142
	25.0	1.0000
	30.0	0.82818

Aufgabe 24: LDR komplex

Für ein Moorhuhn-Lasertag kann man zwei gleichartige LDR in Reihe schalten und wie abgebildet am Arduino anschließen. Jeder LDR soll zu einem Moorhuhn gehören. Durch Einlesen des Wertes in A0 soll ermittelt werden, welches Moorhuhn vom Laser getroffen wurde.

- Erläutere, welche Auswirkung der Laser beim Treffen eines LDR auf die Widerstände und die Spannungen hat.
- Erkläre, welcher Wert sich in A0 näherungsweise einstellen sollte, wenn gerade keiner der beiden LDR getroffen ist.
- Entwickle mithilfe der unten abgebildeten Befehle ein Programm, das auf dem seriellen Monitor ausgibt, welches Moorhuhn (welcher LDR) getroffen wurde. Das Programm soll als Struktogramm dargestellt werden.



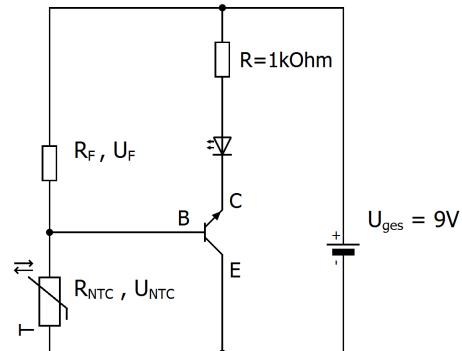
Aufgabe 25: Transistor

Der Schaltplan rechts zeigt eine Transistor-Grundschaltung, in der ein Spannungsteiler mit einem Festwiderstand R_F und ein NTC mit Widerstand R_{NTC} verbaut ist. In der folgenden Tabelle ist festgehalten, bei welcher Temperatur der NTC welchen Widerstand hat.

T in °C	25	20	15
R in kΩ	10	12,1	14,8

Bestimme die Größe von R_F so, dass der Transistor bei 25 °C (20 °C, 15 °C) schaltet.

Hinweis: Der Transistor schaltet bei einer Spannung $U_{BE} = 0,7\text{V}$.



Motivationsquellen

> [Laser-Game](#)

Ein kleines Spiel, das sich auf einfache Weise nachbauen lässt.

> [Arduino Garden Controller](#)

Gartenarbeit muss heute nicht mehr aufwendig sein: Mit einem Arduino lassen sich die Pflanzen automatisch bewässern, wenn die Erde nicht mehr feucht genug ist. Die erhobenen Daten lassen sich außerdem schön visualisieren.

> [Wetterstation von bitluni](#)

[Das Problem mit Wettervorhersagen \(Dr. Whatson, Youtube\)](#)

Selbst gebaute Wetterstationen sind beliebte Anfängerprojekte, bei denen meist ein WLAN-fähiger Mikrocontroller auf Basis des ESP8266 zum Einsatz kommt. Dieser lässt sich ebenfalls über die Arduino IDE programmieren. Wer etwas mehr Hintergrundwissen dazu haben will, schaut sich das Video von [Dr. Whatson](#) an, der außerdem das Projekt [SenseBox](#) vorstellt.

> „[Use the force or your brainwaves](#)“ (Youtube)

„[Use the force or your brainwaves](#)“ (Projektseite)

Der Schüler Imets Tamás hat es mithilfe mehrerer Arduinos geschafft, seine Gehirnwellen einzulesen und zu nutzen, um einen Roboter zu steuern!

6. Erweiterung des Werkzeugkastens: Bauteilkunde

Nach Erarbeitung der bisherigen Kapitel sind nun die wichtigsten Grundlagen zum Arduino gelegt. Die folgenden Seiten bieten jeweils eine Einführung in neue Bauteile, für die unterschiedliche Grundlagen aus den bisherigen Kapiteln benötigt werden. Dazu waren bisher jeweils Hinweise mit Links zu den entsprechenden Abschnitten eingestreut, ab wann die nötigen Grundlagen für die Bauteile und deren Projekte gelegt waren. Daher finden sich neben den Bauteilen in den folgenden Abschnitten jeweils Links, die an die entsprechenden Stellen im Skript zurückführen und auch zur Wiederholung der Grundlagen dienen können.

Überlegt euch zu zweit ein Arduino-Projekt, das ihr gerne umsetzen würdet, und erarbeitet euch die fehlenden Grundlagen dazu. Ihr könnt alle Bauteile aus eurem Arduino-Kit nutzen. Die folgenden Abschnitte geben euch eine Einführung zu den einigen Bauteilen und ggf. bieten sie euch auch Ideen, was man mit dem Arduino umsetzen kann. Am Ende solltet ihr euer Projekt mit einem funktionierenden Prototypen auf einem Steckbrett vorstellen können. Zudem sollt ihr eine kurze Ausarbeitung mit folgenden Abschnitten anfertigen:

- Projektziel
- Materialliste
- Schaltplan
- Programm (als Screenshot)
- Erläuterung des Schaltplans und des Programms
- Ausblick: Mögliche Erweiterungen oder Anwendungen

Projekte in diesem Kapitel:

> Alarmanlage	60	> Joystick-RGB-Steuerung	69
> Carport-Lampe	61	> Fernsteuerung eines RGB-LED-Streifens	71
> Einparkhilfe für ein Auto	63	> Wetterstation	72
> Uhr	66		
> Fensterheber	66		

6.1. Neigungsschalter

Mit sogenannten Neigungsschaltern (engl. *tilt switch*) lässt sich eine Neigung, aber auch eine Erschütterung oder der Beginn einer Beschleunigung messen. So lässt sich zum Beispiel feststellen, ob ein Gegenstand angehoben wird.



Ziel: Es soll eine Alarmanlage gebaut werden, die auslöst, wenn das Steckbrett angehoben wird.

B 6.1. Neigungsschalter.

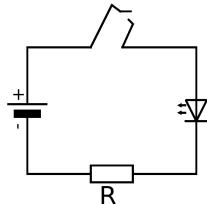
Aufgabe 1:

Die Abbildung rechts zeigt den Aufbau eines Neigungsschalters im geschlossenen und geöffneten Fall. Beschreibe den Aufbau des Schalters und erkläre, wie es in Abhängigkeit der Neigung des Neigungsschalters zum Leuchten der LED in Abbildung 6.2 kommt.



B 6.3. Neigungsschalter (geöffnet).

Neigungsschalter



B 6.4. Neigungsschalter (geschlossen).

B 6.2. Einfacher Aufbau zum Test eines Neigungsschalters ohne Arduino.

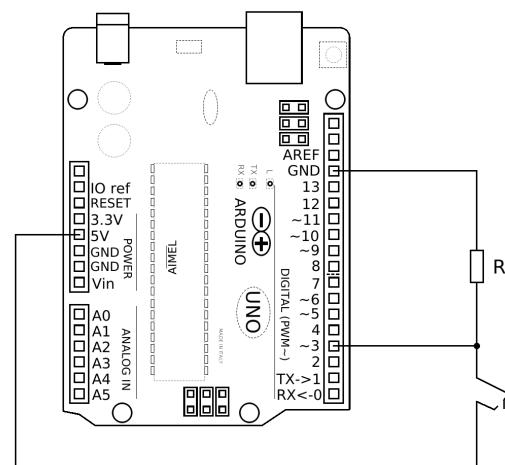
Digitale
Eingänge

Projekt 1: Alarmanlage

Baue eine Alarmanlage, die auslöst, wenn das Steckbrett angehoben wird.

Hinweis: Wenn der Neigungsschalter wie rechts abgebildet am Arduino angeschlossen wird, kann sein Zustand in Digitalpin 3 ausgelesen werden (vgl. das Auslesen von Tastern).

Zusatz: Erkläre, warum es sinnvoll ist, den Piezo-Summer nicht so wie die LED in Abb. 6.2 direkt mit dem Neigungsschalter zu verbinden, sondern das Auslösen des Tons im Programm zu regeln.



6.2. Bewegungsmelder

Ziel: Es soll eine Carport-Lampe gebaut werden, die für einige Zeit leuchtet, wenn sie eine Bewegung registriert, und ansonsten dunkel bleibt.

Bewegungsmelder verfügen über drei Pins, deren Beschriftung man lesen kann, wenn man die Kunststofflinse vorsichtig abzieht (*Vorsicht: Nach Abziehen der Linse nicht den Sensor berühren!*). Vcc und GND dienen der Stromversorgung der elektronischen Komponenten und müssen mit 5 V und GND am Arduino verbunden werden.

Der mittlere OUT-Pin ist der Signal-Pin: Wenn eine Bewegung registriert wurde, liegt er auf einem hohen elektrischen Potential (HIGH); wenn keine Bewegung registriert wurde, liegt er auf einem niedrigen elektrischen Potential (LOW). Zum Einlesen des Signals wird dieser Pin mit einem digitalen Pin des Arduino verbunden, der dann digitaler Eingang heißt.

Hinter befinden sich zwei Drehregler („Potentiometer“), mit denen sich die Dauer des Bewegungssignals (links) und die Empfindlichkeit (rechts) einstellen lassen. Zusätzlich befindet sich auf der rechten Seite ein sogenannter Jumper, mit dem auf einfache Weise eine Steckverbindung zwischen benachbarten Pins hergestellt werden kann. Wenn sich der Jumper ganz außen befindet, dann bleibt das Bewegungssignal nach dem Erkennen einer Bewegung eine Weile aktiv und wird dann auf jeden Fall deaktiviert. Eine neue Bewegung kann erst nach einer gewissen Zeit wieder registriert werden. Wenn der Jumper hingegen leicht nach innen versetzt ist, bleibt das Bewegungssignal so lange erhalten, wie eine Bewegung erkannt wird (siehe Funduino¹).

Projekt 1: Carport-Lampe

Baue und programmiere eine Carport-Lampe. Experimentiere mit den Drehreglern, um die Empfindlichkeit und Dauer des Signals richtig einzustellen.

🔍 Recherche: Wie funktioniert eigentlich ein Bewegungsmelder?

Das zentrale Bauteil eines Bewegungsmelders ist ein sogenannter *Passiver Infrarot Sensor (PIR)*, auch *Pyroelektrischer Sensor*. Recherchiere im Internet, wie solche Sensoren funktionieren und fasst zusammen, wie es zur Registrierung einer Bewegung kommt.

Digitale
Eingänge

¹<https://funduino.de/nr-8-bewegungsmelder>

6.3. Ultraschallsensor

Schleifen

Serieller Monitor

Ultraschallsensoren ermöglichen die berührungslose Messung eines Abstands zwischen dem Sensor und dem nächstgelegenen Gegenstand. Dies macht sie zu einer interessanten Ausrüstung für Staubsaugerroboter, die nicht gegen die Wand fahren sollen, oder Einparkhilfen im Auto, die dem Fahrer anzeigen sollen, wie viel Platz er noch hat.

Ziel: Es soll eine Einparkhilfe für ein Auto gebaut werden.

Die wichtigsten Bestandteile des Ultraschallsensors sind der „Transducer“ (**T**) und der „Receiver“ (**R**). Der Transducer ist praktisch ein Lautsprecher, der für uns nicht hörbare Schallwellen aussendet. Der Receiver entspricht einem Mikrofon für Schallwellen. Die Schallwellen werden also vom Transducer ausgesendet, an einem Hindernis reflektiert und vom Receiver empfangen.

Der Ultraschallsensor verfügt über vier Pins. GND und VCC (5V) sind wie üblich zu belegen und dienen der Energieversorgung. Der Trigger-Pin dient dazu, einen Ultraschallpuls auszusenden - wird er für 10 Mikrosekunden auf ein HIGH-Potential gebracht, wird der Ultraschallpuls getriggert. Wenn dies geschieht, wird der Echo-Pin von der Elektronik des Sensors auf ein HIGH-Potential gebracht, das so lange anhält, bis der Receiver die reflektierte Schallwelle empfängt.

Die Zeit, die der Echo-Pin auf HIGH liegt, gibt also an, wie lange der Schall braucht, um vom Sensor zum Hindernis und zurück zu gelangen. Sie kann vom Arduino mithilfe des Befehls

`lese Puls-Pin <_> Timeout <_>` gemessen werden. Der Timeout gibt dabei an, nach wie vielen Mikrosekunden der Arduino aufhört, auf eine reflektierte Schallwelle zu warten.

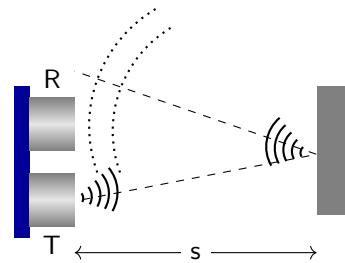
Aufgabe 1: Entferungen messen

Das rechts abgebildete Programm zeigt, wie der Trigger, der mit D9 am Arduino verbunden ist, für genau 10 Mikrosekunden auf HIGH gestellt und dann der Echo-Pin, der mit D8 am Arduino verbunden ist, zur Zeitmessung verwendet wird. In der Variable `zeit` ist am Ende also die *Zeit in Mikrosekunden* gespeichert, die die Schallwellen vom Sensor zum Hindernis und zurück gebraucht haben.

Erweitere das Programm so, dass die Entfernung vom Mikrocontroller zum Hindernis berechnet und auf dem seriellen Monitor ausgegeben wird. Überprüfe deine Ultraschall-Messung mit dem Geodreieck.

Hinweise:

- Schallwellen breiten sich in Luft mit Schallgeschwindigkeit aus - diese beträgt etwa $330 \frac{\text{m}}{\text{s}}$.
- $1 \text{ ms} = 1000 \mu\text{s}$ (Mikrosekunden)



Aufgabe 2: *Kalibrierung*

Die Schallgeschwindigkeit hängt auch von der Temperatur der Luft ab. Dementsprechend kann es sein, dass die mit dem Wert $330 \frac{\text{m}}{\text{s}}$ ermittelten Entferungen (zu) ungenau sind.

Um einen besseren Wert der Schallgeschwindigkeit zu ermitteln, kann man die Strecke zum Hindernis mit einem Geodreieck oder Maßband genau festlegen und die Zeit, die der Schall braucht wie oben beschrieben messen. Bestimme daraus einen besseren Wert für die Schallgeschwindigkeit.

Projekt 1: Einparkhilfe für ein Auto

Baue eine Einparkhilfe für ein Auto, die umso schneller piepst, je näher man dem Hindernis kommt. Ab einer Entfernung von 30 cm soll der Ton durchgängig ertönen.

 **Recherche: Wie wird Ultraschall erzeugt und gemessen?**

Die Erzeugung des Ultraschalls beruht wie beim Piezo-Summer auf dem inversen piezo-elektrischen Effekt (vgl. S. 18); die Messung des Ultraschalls beruht auf dem piezo-elektrischen Effekt. Recherchiere im Internet die Hintergründe dieser Effekte und fasse sie zusammen.

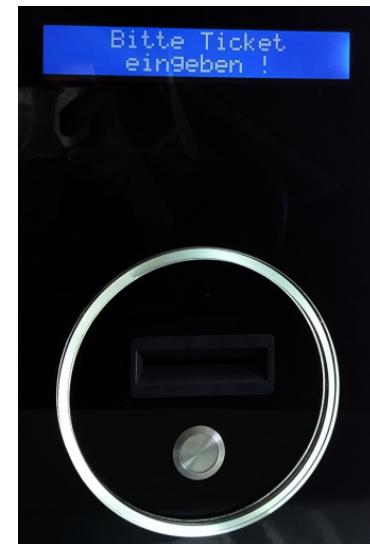
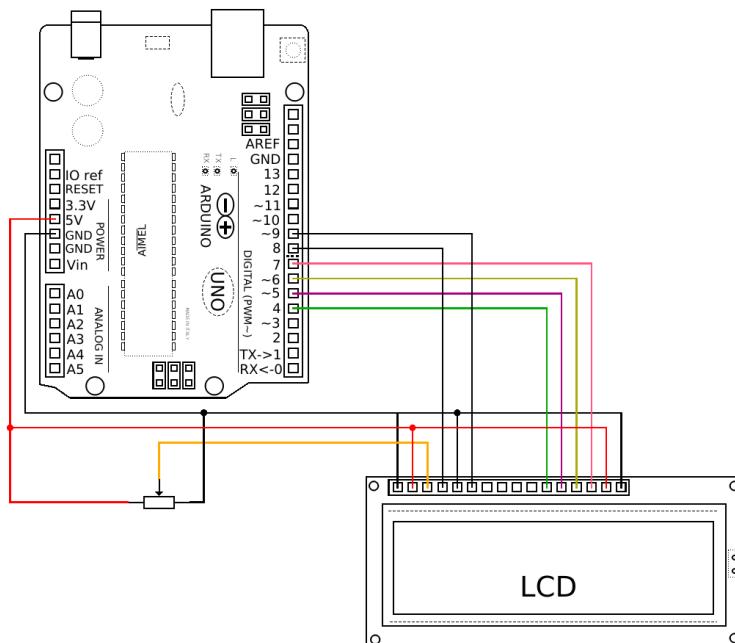
6.4. Liquid Crystal Display (LCD)

In vielen Projekten genügt es nicht, Messwerte, Statusanzeigen oder Menüs über den seriellen Monitor am Computer anzeigen zu lassen - man benötigt stattdessen ein Display, das sich direkt an den Arduino anschließen und mit ihm verbauen lässt. Ein günstige Möglichkeit dafür bieten sogenannte **Liquid Crystal Displays (LCD)**, die man zum Beispiel in Kaffeemaschinen oder Parkautomaten finden kann (siehe B6.5). Modernere LCD werden in Laptops, Fernsehern und Tablets verbaut.

Frage: Wie verwendet man ein LC-Display am Arduino?

Bevor das LC-Display mit mBlock angesteuert werden kann, muss eine Erweiterung installiert werden, die entsprechende Befehle bereitstellt. Die Erweiterung findest du unter [Extensions](#) → [Extensions verwalten](#). Gib im Suchfeld „lcd“ ein und installiere die Erweiterung „LCD“ von Heine Ravnholz.

Die Erweiterung legt fest, an welchen Pins das LC-Display angeschlossen werden muss (siehe unten). Es empfiehlt sich, die zahlreichen 5V- und GND-Anschlüsse auf den Längsseiten des Stecksbretts zu sammeln.



B 6.5. LC-Display an einem Parkhaus-Automaten.

LCD	Arduino
VSS	GND
VDD	5V
V0	Drehregler (Mitte)
RS	8
RW	GND
E	9
D0 - D3	-
D4	4
D5	5
D6	6
D7	7
A	5V
K	GND

Aufgabe 1: Funktionstest

Schließe das LC-Display wie beschrieben an den Arduino an und erstelle mithilfe der LCD-Extension ein Programm, das „Hello World!“ auf dem LC-Display anzeigt.

Hinweis: Falls alle Pixel weiß oder blau bleiben, kann es sein, dass der Drehregler falsch eingestellt ist. Drehe in diesem Fall an dem Drehregler, um den Kontrast zu verbessern.

Aufgabe 2: *Messwertanzeige*

In vielen Anwendungen soll auf dem LC-Display ein Messwert o. ä. angezeigt werden, der sich mit der Zeit ändern kann. Diese Anzeige soll aber schön formatiert sein.

Erstelle ein Programm, das alle drei Sekunden eine Zufallszahl z zwischen 0 und 200 erzeugt und auf dem Display folgende Anzeige ausgibt:

Messwert: z E

Hinweise:

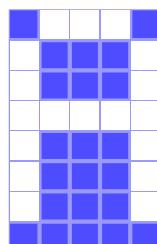
- E soll für eine beliebige Einheit stehen.
- Achte darauf, dass der vorherige Wert von z gelöscht wird (`lcd clear`).
- Die Ausgabe der Zahl z soll immer rechtsbündig erfolgen, sodass zwischen den Einern von z und der Einheit genau ein Leerzeichen steht.
- Du benötigst den Befehl `LCD set cursor (line <_> position <_>)`. Darin steht line für die Zeile, deren Nummer entweder 0 (oben) oder 1 (unten) ist. position steht für die Spalte, deren Nummer sich von 0 bis 15 erstrecken kann. (In der Informatik beginnt das Zählen stets mit der Null!)

Binär-
system

Aufgabe 3: *Codierung von Zeichen auf dem LCD*

In der eingefügten Extension muss im Hintergrund geklärt werden, wie die eingegebenen Zeichen auf dem LCD dargestellt werden. Dazu lohnt ein genauer Blick auf die Zellen des LCD.

Jede Zelle besteht aus 5×8 Pixeln, von denen manche weiß und manche blau sind. Wenn man für jedes Pixel eine 1 (weiß) oder eine 0 (blau) notiert, dann erhält man Bitfolgen (gekennzeichnet durch das vorstehende 0b), die sich in einer Reihung notieren lassen.



0	1	1	1	0		0b01110
1	0	0	0	1		0b10001
1	0	0	0	1		0b10001
1	1	1	1	1		0b11111
1	0	0	0	1		0b10001
1	0	0	0	1		0b10001
1	0	0	0	1		0b10001
0	0	0	0	0		0b00000

`A={0b01110, 0b10001, 0b10001, 0b11111, 0b10001, 0b10001, 0b10001, 0b00000}`

B 6.6. Codierung des Buchstabens A auf einem LC-Display.

Man könnte die Reihung von Bitfolgen auch als Reihung von Dezimalzahlen notieren und käme auf das gleiche Ergebnis. Das macht den Code zwar kürzer, jedoch leidet die Lesbarkeit des Codes deutlich.

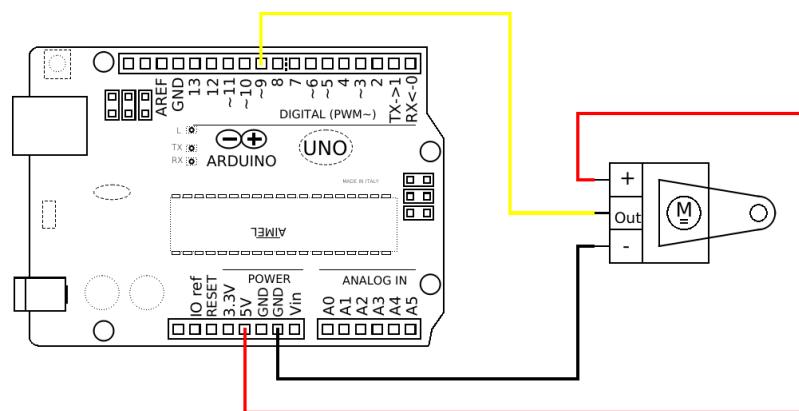
Entwerfe einen Smiley und ein eigenes Symbol auf 5×8 Pixeln und notiere die zugehörige Reihung von Bitfolgen, die dieses Zeichen codiert.

Hinweis: Mit der textbasierten Arduino-IDE lassen sich nach dem oben beschriebenen Prinzip auch eigene Zeichen für das LC-Display codieren. Ein Beispiel findet sich unter Datei → Beispiele → LiquidCrystal → CustomCharacter.

6.5. Servo

 Pulsweitenmodulation

Ein Servo ist in der Regel ein kleiner Elektromotor zusammen mit einer elektronischen Steuereinheit, die dazu dient, den Motor auf einen bestimmten Winkel einzustellen. Häufig wird beides zusammen als Servomotor bezeichnet. Angewendet werden Servos in vielen Bereichen, zum Beispiel im Modellbau, aber auch in den Fensterhebern von Autos.



B 6.7. Verschaltung eines Servo am Arduino.

Der Servo wird mit drei Anschlüssen an den Arduino angeschlossen:

- VCC (rot): Die Stromversorgung des Servo wird mit dem 5 V-Pin des Arduino verbunden. Dabei ist zu beachten, dass ein Servomotor relativ große Stromstärken „ziehen“ kann. Der 5 V-Pin des Arduino kann bis zu 200 mA ausgeben, bevor er durchbrennt. Das ist für den Servo genug. Ein normaler Digitalpin verträgt dagegen nur 40 mA, was deutlich zu wenig für den Servo ist. Die Stromversorgung des Servo kann also nicht über einen normalen Digitalpin sichergestellt werden.
- GND (schwarz/braun): Die Stromversorgung ist nur komplett, wenn auch das GND-Niveau auf das GND-Niveau des Arduino festgelegt wird.
- Signalleitung (gelb): Die Einstellung des Winkels erfolgt über ein PWM-Signal, deshalb muss das gelbe Kabel mit einem PWM-Pin am Arduino verbunden werden.

Projekt 1: Uhr

Baue mit einem Servo den Sekundenzeiger einer Uhr.

setze Servo-Pin 9 Winkel auf 90 Grad

Projekt 2: Fensterheber

Der Fensterheber eines Autos hat üblicherweise zwei Funktionen:

- (a) Wenn man den Schalter halb hochzieht bzw. herunterdrückt, wird das Fenster so lange angehoben bzw. abgesenkt, wie der Schalter in dieser Position verbleibt.
- (b) Wenn der Schalter ganz hochgezogen bzw. heruntergedrückt wird, wird das Fenster auch

ganz hochgehoben bzw. heruntergefahren.

Modelliere eine Funktion des Fensterhebers mit zwei Tastern (mit Widerstand! - vgl. Abschnitt [3.7.2](#)) und einem Servo. Wenn du beide Funktionen modellieren willst, brauchst du vier Taster (mit Widerstand!).

🔍 **Recherche: Wie funktioniert die Steuerung eines Servos?**

Der Winkel, auf den sich die Ausgangswelle des Servo drehen soll, wird über ein PWM-Signal geregelt. Recherchiere im Internet, wie dies realisiert wird und fasste es zusammen.

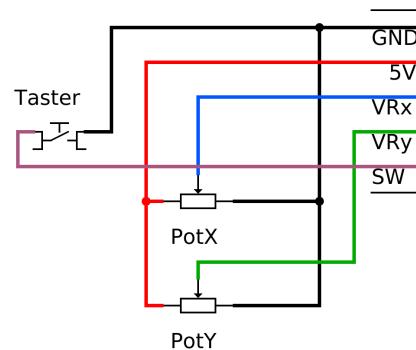
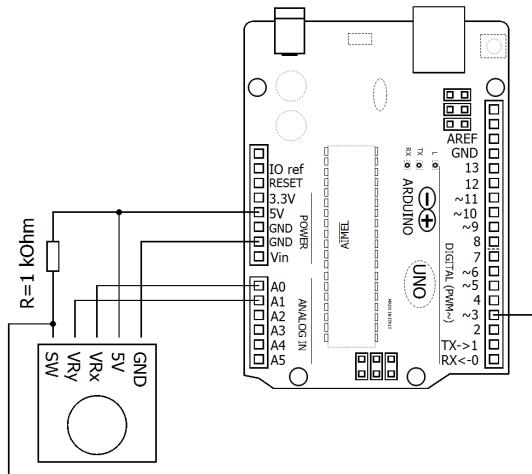
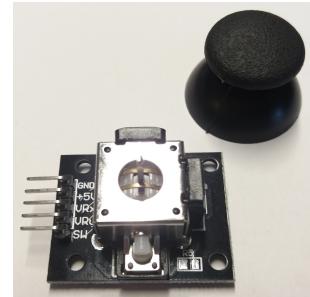
6.6. Joystick

 **Potentiometer**

Joysticks werden bekanntermaßen für Spielecontroller oder auch zur Steuerung von Maschinen genutzt. Mit dem Arduino lassen sich einfache Versionen davon nachbauen.

Frage: Wie funktioniert ein Joystick und wie verwendet man ein Joystick-Modul am Arduino?

Ein Joystick besteht im Wesentlichen aus zwei Potentiometern, die über einen gemeinsamen Hebel variiert werden können. Wie im Schaltbild zu sehen, teilen sich beide den 5V- und GND-Anschluss; der mittlere Anschluss muss natürlich jeweils einzeln ausgelesen werden. Zusätzlich wird durch Drücken des Joysticks ein angebrachter Taster gedrückt, dessen Status am SW-Pin ausgelesen werden kann (*sw* von engl. „switch“). Da das elektrische Potential am SW-Pin normalerweise schwankt, sollte ein *Pullup*-Widerstand mit $R = 1\text{ k}\Omega$ angebracht werden (vgl. Schaltbild).



B 6.9. Ersatzschaltplan für das Joystick-Modul.

B 6.8. Anschluss des Joystick-Moduls an den Arduino.

Aufgabe 1: Erste Experimente

- Bewege den Hebel des Joystick-Moduls und beobachte, wie sich dabei die Potentiometer an den Seiten mitbewegen. Bringe auch den Plastikdeckel an, drücke den Joystick herunter und beobachte dabei das Verhalten des Tasters.
- Schließe das Joystick-Modul wie oben beschrieben an den Arduino an. Lies die Werte der Potentiometer aus, während du sie bewegst. Notiere, welche Bewegungsrichtung die X-Richtung und welche die Y-Richtung darstellt. Notiere außerdem, welches der beiden Potentiometer (ggü. von Taster oder ggü. der Pins) für die X-Richtung bzw. Y-Richtung verantwortlich ist.
- Mit dem Pullup-Widerstand wird eine sogenannte Active-Low Schaltung aufgebaut. Teste die Funktionsweise des Tasters, indem du das elektrische Potential in D3 ausliest und beschreibe, was mit dem Begriff Active-Low gemeint ist.

Projekt 1: Joystick-RGB-Steuerung

Steuere mit dem Joystick-Modul die Farbe einer RGB-LED!

Da man nur zwei Freiheitsgrade bzw. Bewegungsrichtungen hat, lassen sich leider nicht alle Farbanteile mit dem Joystick-Modul kontrollieren. Interessant sieht das Ergebnis trotzdem aus.

Achtung bei Verwendung von Motoren: Es wäre natürlich reizvoll, mit dem Joystick-Modul und zwei Servos oder Motoren einen Roboter-Arm zu steuern. Der Arduino allein kann aber weder zwei Servos noch zwei einfache Elektromotoren mit Strom versorgen. Wie eine solche Schaltung zu realisieren wäre, zeigt Kapitel 7.



6.7. Infrarot-Fernbedienung

 Helligkeit

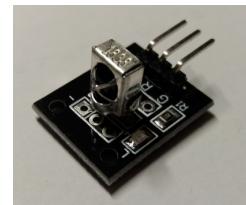
messen

Jeder weiß, wie angenehm es ist, wenn man ein Gerät fernsteuern kann statt aufzustehen zu müssen, um die angebrachten Knöpfe zu drücken. Eine einfache Möglichkeit dafür bietet eine Infrarot(IR)-Fernbedienung.

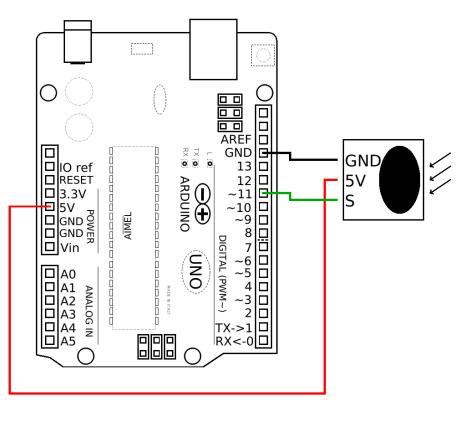
Frage: Wie verwendet man eine IR-Fernbedienung mit dem Arduino?

Wie am Namen zu erkennen, verwendet eine IR-Fernbedienung Infrarotstrahlen, die mit dem bloßen Auge nicht sichtbar sind. Hält man jedoch eine Digitalkamera, z. B. vom Smartphone, auf die Infrarot-LED der Fernbedienung und drückt eine Taste, dann kann man ein schnelles Aufblitzen erkennen. Am besten probierst du es selbst einmal aus oder schaust dir ein kurzes  [Video der IR-Strahlen](#) an. Das Aufblitzen zeigt, dass die Strahlen in einem bestimmten Rhythmus gesendet werden, aus dem sich entschlüsseln lässt, welche Taste gedrückt wurde.

Empfangen werden die Infrarotstrahlen von einem IR-Empfängermodul, das unter anderem eine Fotozelle enthält, die im Gegensatz zum LDR nicht auf das sichtbare Licht reagiert, sondern auf Infrarotlicht bei einer Frequenz von 38 kHz. Der Anschluss an den Arduino ist einfach: GND und 5V dienen wie üblich der Stromversorgung. Der Signal-Pin S muss mit einem beliebigen PWM-Pin (mit ~) verbunden werden.



B 6.10. IR-Empfängermodul.



B 6.11. Schaltplan zum Anschluss des IR-Empfängermoduls am Arduino.

Zum Einlesen und Entschlüsseln des Signals sollte die Erweiterung „IR Remote“ von AbbadoN installiert werden. Dazu wählt man [Extensions](#) → [Extensions verwalten](#) und sucht nach „ir remote“.

Mithilfe der Befehle aus der Erweiterung lässt sich das Drücken der Fernbedienung wie unten abgebildet auslesen:

- Mit `IRremote pin(<_>)` wird ein IR-Empfänger initialisiert, dessen Signalpin im rechts abgebildeten Beispiel mit Pin 11 verbunden ist.

- Der Befehl `bulean result receive` (Achtung: „boolean“ wurde dort falsch geschrieben) gibt TRUE zurück, wenn ein neues Signal empfangen wurde, ansonsten FALSE.
- Das decodierte Signal ist eine Zahl, auf deren Wert man mit `Value Results` zugreifen kann. Dies kann zum Beispiel auf dem seriellen Monitor angezeigt werden. Alternativ oder zusätzlich kann durch eine Abfrage des Codes die LED an Pin 13 an bzw. ausgeschaltet werden (im Beispiel rechts mit den Tasten 1 und 2).
- Mit `Resume IR receive` wird der Wert von `Value Results` wieder zurückgesetzt und das IR-Empfängermodul wieder in den Empfangsmodus versetzt. Dieser Befehl sollte daher immer als Letztes nach der Verarbeitung des Signals erfolgen.



B 6.12. Einfaches Beispielprogramm zur Verwendung einer IR-Fernbedienung.

Aufgabe 1: Codes kennen lernen

- Übertrage das oben abgebildete Programm auf den Arduino und probiere es aus.
- Erstelle eine Tabelle, in der du den Zahlencode für jede Taste festhältst. Probiere auch aus, was passiert, wenn du die Tasten länger gedrückt hältst.

Projekt 1: Fernsteuerung eines RGB-LED-Streifens

In vielen Bereichen werden RGB-LED-Streifen genutzt, um einen Raum mit passendem, indirektem Licht auszustatten. Die meisten RGB-Streifen lassen sich über eine kleine Infrarot-Fernbedienung steuern, wodurch sich die Farbe, aber auch der Modus einstellen lässt - zum Beispiel eine einzelne Farbe, **Fading**, **Strobe**, ...

Die einzelnen LEDs machen dabei alle dasselbe. Es reicht also für einen Prototypen, das Muster mit einer RGB-LED nachzubauen. Programmieren die verschiedenen Modi so, dass sie sich über die Fernbedienung steuern lassen.



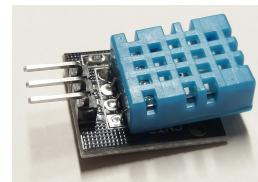
Lichterkette-Beispielvideo

6.8. Temperatur- und Luftfeuchtigkeitssensor DHT-11

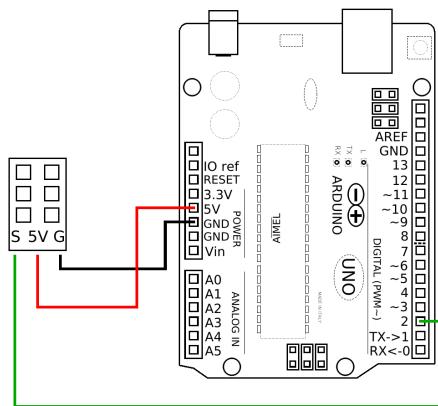
Temperatur messen

Bei vielen Umweltmessungen interessiert nicht nur die Temperatur, sondern auch die Luftfeuchtigkeit. Der Sensor DHT-11 ist ein einfaches, kleines Bauteil, mit dem sich beides messen lässt.

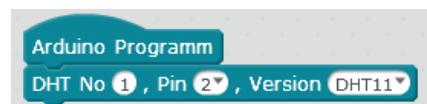
Frage: Wie verwendet man den DHT-11 am Arduino?



Der DHT-11 verfügt über drei Pins - 5V und GND dienen der Stromversorgung, während das Signal zu den Messdaten über den Signalpin ausgegeben wird. Für die Temperaturmessung ist auf dem DHT-11 ein NTC verbaut.



Das Auslesen des Signalpins ist einfach, weil es von einer Erweiterung erledigt wird, die die Analyse des Signals übernimmt und Befehle bereitstellt, mit denen sich direkt auf die Temperatur und die Luftfeuchtigkeit zugreifen lässt.



Zum Einbinden der Erweiterung wählt man [Extensions](#) →

[Extensions verwalten](#) und sucht nach „dht“. Es sollte die Erweiterung „DHT Extensions“ installiert werden. Diese bietet neben zwei Befehlen zum Auslesen von Temperatur und Luftfeuchtigkeit einen Befehl zum Initialisieren des DHT-Objekts im Programm. Dieser Befehl muss direkt nach dem Programmstart eingefügt werden.

Projekt 1: Wetterstation

Baue eine kleine Wetterstation, die alle zehn Minuten Temperatur und Luftfeuchtigkeit misst und auf dem seriellen Monitor ausgibt.

 **Recherche:** Wie wird die Luftfeuchtigkeit gemessen?

Mit dem DHT-11 lässt sich die relative Luftfeuchtigkeit bestimmen. Recherchiere, was darunter zu verstehen ist, und wie diese durch ein elektrisches Bauteil gemessen wird.

7. Elektromotoren steuern

Erst mit Motoren werden elektrischen Schaltungen dazu befähigt, mechanisch in ihre Umgebung einzugreifen, *etwas zu tun*. Die bereits kennengelernten Servos haben dies auch schon ermöglicht, dienen aber eher für Steuerungen, die eine Feinjustierung benötigen. Größere Aufgaben bewältigt der Elektromotor.

In diesem Kapitel lernst du...

- ... wie man einen Elektromotor richtig am Arduino anschließt,
- ... wie man einen Elektromotor mit einem Transistor steuert,
- ... wie man einen Elektromotor mit einem Relais steuert,
- ... wie man einen Elektromotor mit einem Motortreiber-IC steuert,
- ... welche Vorteile und Nachteile die jeweiligen Steuerungsarten bieten.

Projekte in diesem Kapitel:

> Badezimmerlüfter 76

> Waschmaschinensteuerung 79

7.1. Elektromotor und Diode

⌚ Mit Transistoren steuern

Bei vielen Projekten soll sich etwas bewegen - dies lässt sich mit Elektromotoren realisieren. Die Ansteuerung eines Elektromotors erfordert auf der Hardware-Seite ein wenig Vorbereitung, denn aufgrund der hohen Ströme, die Elektromotoren ziehen, sollte man sie nicht direkt an den Digitalpins des Arduino anschließen. Für die Steuerung greift man meistens auf einen Transistor zurück; eine brauchbare Alternative ist aber auch das Relais. Für beide Steuerungsmöglichkeiten sollte aber zuerst geklärt werden, wie ein Motor grundsätzlich aufgebaut und betrieben wird.

Frage: Wie betreibt man einen Elektromotor am Arduino?

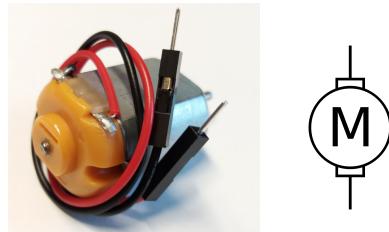
Ein **Elektromotor** besteht aus mehreren Spulen und Magneten. Wenn Strom durch die Spulen fließt, baut sich um die Spulen ein Magnetfeld auf, das mit dem Magnetfeld der eingebauten Magneten wechselwirkt (Anziehung/Abstoßung), sodass es zu einer Drehung des Motors kommt. Ein sogenannter Kommutator sorgt dafür, dass der Strom durch die Spulen ständig seine Richtung wechselt, sodass es immer wieder von Neuem zu Anziehung bzw. Abstoßung der Magnetfelder kommt und die Drehung nicht aufhört, solange eine Spannung anliegt.

Wenn keine Spannung mehr am Motor anliegt, wird sich der Motor aufgrund seiner Trägheit immer noch ein wenig weiterdrehen. Durch das Drehen der Spulen im Magnetfeld der eingebauten Permanentmagneten wird in den Spulen ein Strom induziert, dessen Richtung entgegengesetzt zur vorherigen Richtung ist. Dieser „falsch“ gerichtete Strom würde den Arduino zerstören. Aus diesem Grund schaltet man eine *Diode* parallel zum Motor.

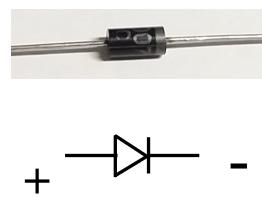
Eine **Diode** ist wie ein elektrisches Ventil: Sie lässt den Strom nur in eine Richtung durch. Im Gegensatz zu Leuchtdioden wandeln „normale“ Dioden die elektrische Energie in Wärme um. In *Durchlassrichtung* wird der negative Pol (bzw. GND) mit der Seite verbunden, an der der Ring angebracht ist, und der positive Pol mit der anderen Seite. Die Diode wird jedoch *Sperrrichtung* eingebaut, also so, dass der Ring mit 5V und die andere Seite mit GND verbunden ist. Dadurch fließt im Normalbetrieb kein Strom durch die Diode. Wenn jedoch der entgegengerichtete Induktionsstrom des Motors auftritt, kann dieser durch die Diode abfließen, bis die verbleibende elektrische Energie vollständig in Wärme umgewandelt wurde.

⌚ Recherche: Verpolungsschutz

LEDs leuchten nicht, wenn man sie falsch herum anschließt. Andere Bauteile wie Elektrolytkondensatoren explodieren sogar, wenn man sie falsch herum anschließt. Um zu vermeiden, dass solche Schäden entstehen, wenn man eine Batterie falsch herum anschließt, werden in einigen Fällen Dioden



B 7.1. Gleichstrom-Elektromotor in real und als Schaltsymbol.



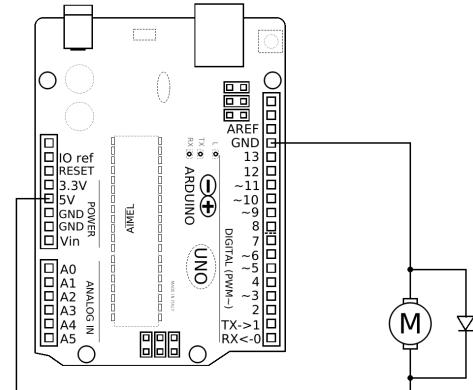
B 7.2. Diode in real und als Schaltsymbol.

genutzt. Recherchiere im [Elektronik-Kompendium](#),¹ wie dies funktioniert.

Aufgabe 1: Erkläre die Funktion der Diode parallel zum E-Motor in eigenen Worten.

Baue die rechts abgebildete Schaltung zum Betrieb eines Gleichstrom-Elektromotors am Arduino auf.

! *Es ist sehr wichtig, dass die Diode richtig, also in Sperrrichtung, eingebaut wird, da sonst der Arduino zerstört werden könnte!*



B 7.3. Anschluss eines Gleichstrom-Elektromotors mit dem Arduino als Spannungsquelle.

🔍 Recherche: Aufbau von Gleichstrom-Elektromotoren

Oben wurde die Funktionsweise von Gleichstrom-Elektromotoren bereits angedeutet. Recherchiere im Internet den genauen Aufbau und Ablauf der Drehbewegung.

7.2. Steuerung mit Transistor

Der 5 V-Pin des Arduino liefert zwar in vielen Fällen genügend Strom für den Motor, jedoch lässt er sich nicht programmieren. Dazu lässt sich ein Transistor einbauen.

Frage: Wie steuert man einen Elektromotor mit einem Transistor am Arduino?

¹<https://www.elektronik-kompendium.de/sites/slt/1206251.htm>

Die rechts abgebildete Schaltung zeigt, wie ein npn-Transistor eingebaut werden kann, um den Motor mithilfe von Digitalpin 9 zu schalten. Der Transistor lässt den Strom zwischen Emitter (E) und Kollektor (C) passieren, wenn die Spannung zwischen Basis (B) und Emitter (E) mehr als 0,7 V beträgt, anderenfalls sperrt er. Der Vorwiderstand mit $R = 1\text{k}\Omega$ sorgt dafür, dass der Basisstrom nicht zu groß wird.

Es ist ratsam, die Basis mit einem PWM-Pin (gekennzeichnet durch \sim) zu verbinden, da sich dadurch die Geschwindigkeit des Motors steuern lässt.

Aufgabe 2:

Baue die oben abgebildete Schaltung auf und probiere die Steuerung des Motors mittels

setze PWM-Pin <_> auf <_> aus.

Simuliere mit dem Motor eine konstant beschleunigende Bewegung (vgl. *Fading*), gefolgt von einer abrupten Bremsung.

Projekt 1: Badezimmerlüfter

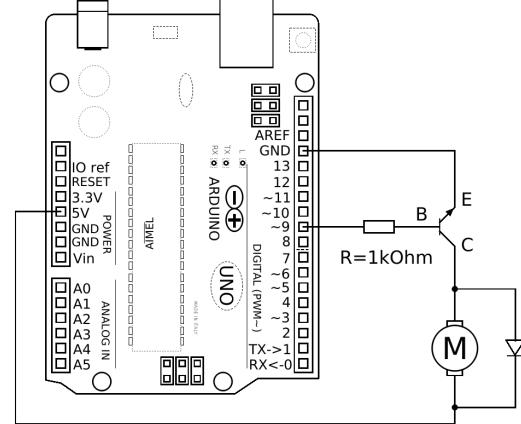
Baue einen Badezimmerlüfter, der anspringt, wenn die Luftfeuchtigkeit größer als 65% oder die Temperatur größer als 30 °C wird. Probiere deine Schaltung durch Anhauchen des Luftfeuchtigkeitssensors aus.

Erweiterung: Programmiere die Schaltung so, dass der Lüfter sich umso schneller dreht, je höher die Luftfeuchtigkeit ist.

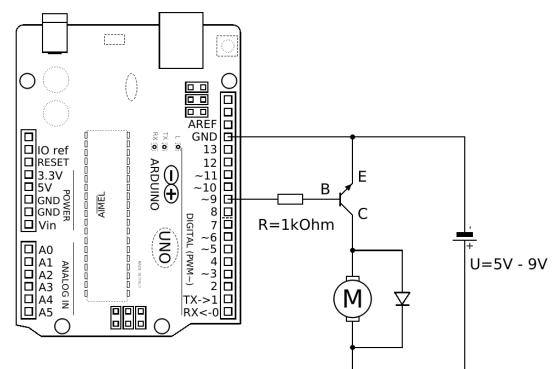
Schaltung mit externer Spannungsquelle

Wenn der verwendete Elektromotor größer ist und mehr Strom zieht bzw. größere Spannungen benötigt, muss für den Elektromotor eine eigene Spannungsquelle verwendet werden, die genügend Spannung und Strom bieten kann. Der rechts abgebildete Schaltplan zeigt, wie der Aufbau dann vorzunehmen ist. Wichtig ist dabei, dass ein gemeinsames GND-Niveau hergestellt wird - vergleichbar einem „Normalnull“ für die Höhenangaben, hier allerdings als „Normalnull“ für das elektrische Potenzial.

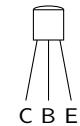
Der Arduino kann über USB oder eine zweite Batterie mit Strom versorgt werden.



B 7.4. Anschluss eines Gleichstrom-Elektromotors am Arduino mit Steuerung über einen Transistor an Digitalpin 9.



B 7.5. Anschluss eines Gleichstrom-Elektromotors am Arduino mit Steuerung über einen Transistor und mit externer Spannungsquelle für den Motor.



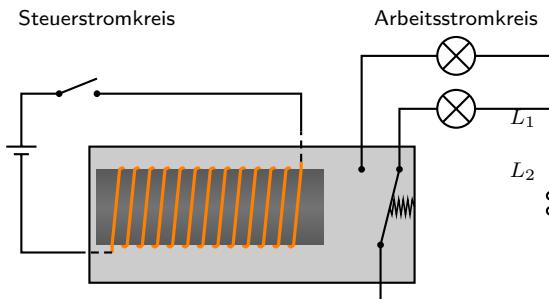
npn-Transistor;
Blick auf flache
Seite

7.3. Steuerung mit Relais

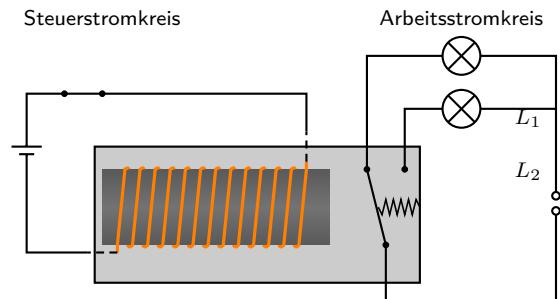
Wie oben zu sehen, muss der Arbeitsstromkreis mit dem Motor und der Steuerstromkreis mit dem Arduino bei Verwendung eines Transistors immer miteinander verbunden bleiben - auch bei sehr großen Stromstärken. Damit verbleibt immer eine gewisse Gefahr, dass eine Spannungsspitze auf den Arduino durchschlägt und ihn zerstört. Mit einem Relais lässt sich dieses Risiko vermeiden. Ein weiterer Vorteil ist, dass der Arbeitsstromkreis auch mit Wechselstrom betrieben werden kann, wenn ein Relais als Schalter genutzt wird.

Frage: Wie verwendet man ein Relais am Arduino?

Ein Relais (siehe unten, grau unterlegt) besteht im Wesentlichen aus einer Spule mit Eisenkern und einem Wechselschalter, an dem eine Feder angebracht ist.



B 7.6. Aufbau eines Relais (grau unterlegt) einschließlich der Stellung des Wechselschalters, wenn im Steuerstromkreis kein Strom fließt.



B 7.7. Aufbau eines Relais (grau unterlegt) einschließlich der Stellung des Wechselschalters, wenn im Steuerstromkreis Strom fließt.

Aufgabe 3: Physikalischer Hintergrund zum Relais

Erkläre die Stellung des Wechselschalters in Abhängigkeit des Steuerstromkreises in den beiden abgebildeten Situationen. Welche Lampe leuchtet?

Die Kontakte am Wechselschalter werden mit *NO* (*normally open*), *NC* (*normally closed*) und *C* (*common ground*) bezeichnet. Ordne die Bezeichnungen den Kontakten in der Skizze zu.

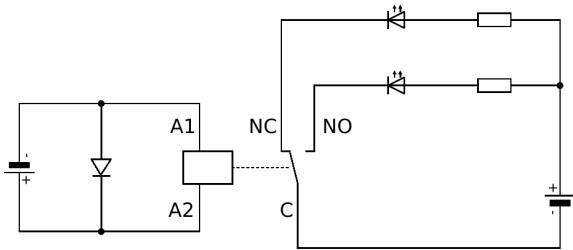
Aufgabe 4: Anschluss eines Relais

- a) Wie aus dem obigen Schaltplan ersichtlich wird, hat ein Relais fünf Anschlüsse, die jedoch nicht beschriftet sind. Suche im Internet nach „Datasheet <Gerätebezeichnung des Relais>“ (Bezeichnung vom Relais ablesen) und entnimm dem Datenblatt, welche Anschlüsse zum Steuer- bzw. Arbeitsstromkreis gehören.



⚠️ **Achtung:** Auf dem Relais ist angegeben, dass damit bis zu 250V Wechselspannung und 10A geschaltet werden können. Das sollte man mit solch billigen Bastelmodulen aber **niemals machen!** Generell gilt: **Nur ausgebildete Fachleute sollten mit Spannungen von mehr als 24V hantieren!**

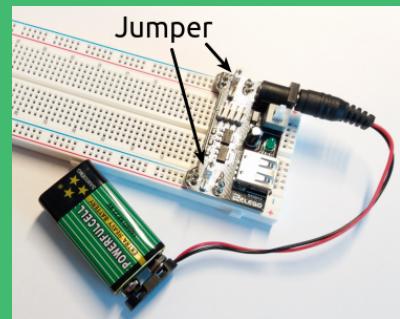
- b) Baue die Schaltung entsprechend des rechts abgebildeten Schaltplans auf. Nutze dazu das „Power Module“ auf dem Steckbrett (Erklärung unten). Probiere die Schaltung des Relais aus, indem du die Stromzufuhr der Spule unterbrichst und wieder herstelltst.



- c) Ordne in einer Skizze des Relais die Anschlüsse ihrer Bezeichnung (A1, A2, C, NO, NC) zu.

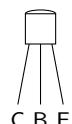
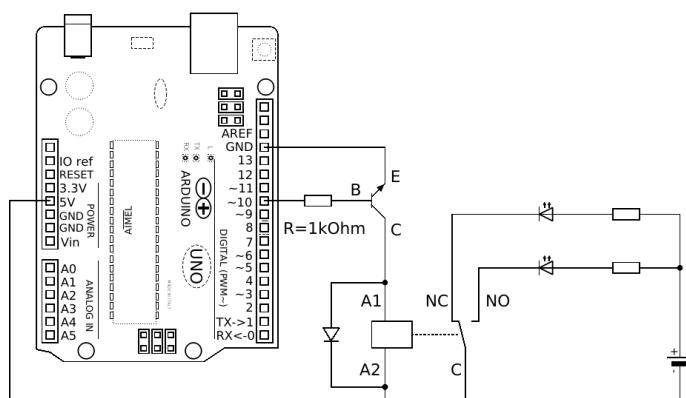
Das „Power Supply Module“

Das Power Supply Module dient zur Spannungsversorgung auf einem Steckbrett. Dazu kann eine Batterie mit 6,5 V bis 12 V oder ein USB-Kabel angeschlossen werden. Die Spannung wird auf dem Modul je nach Einstellung der *Jumper* auf 5 V oder 3,3 V heruntergeregt. Dazu verbindet man mithilfe der Jumper die Anschlüsse 5V und OFF bzw. 3.3 und OFF.



Die Spannung kann entlang der langen äußeren Leisten abgegriffen werden, wenn der Taster neben der Hohlbuchse gedrückt ist. Die Zuordnung zu Pluspol und Minuspol ist auf dem Power Supply Module mit + bzw. – markiert.

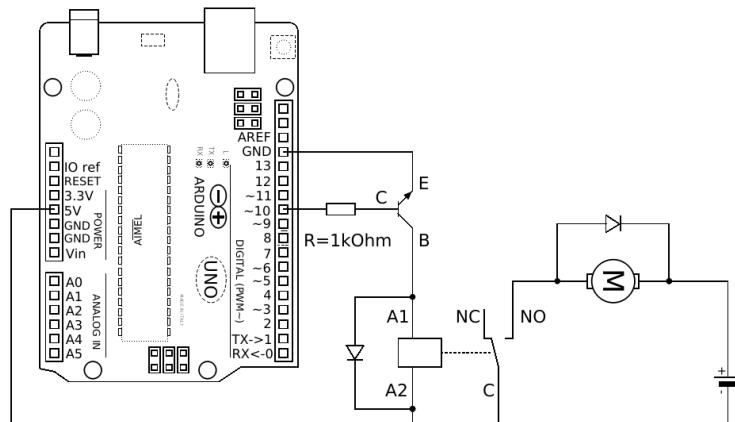
Aufgabe 5: Anschluss eines Relais am Arduino



npn-Transistor;
Blick auf flache Seite

- Der Schaltplan oben zeigt, wie man ein Relais mit dem Arduino steuert.
- Erkläre die Funktion der in Sperrrichtung geschalteten Diode parallel zur Spule des Relais.
 - Erkläre die Funktion des Transistors.
- Hinweis:* Der 5V-Pin und der GND-Pin vertragen bis zu 200 mA. Die Digitalpins vertragen dagegen maximal 40 mA; normalerweise sollten 20 mA nicht überschritten werden (s. [Pin Current Limitations](#)).
- Baue die Schaltung auf und teste sie mit einem Blink-Programm.

Projekt 2: Waschmaschinensteuerung



Baue die oben abgebildete Schaltung zur Steuerung eines Elektromotors mit einem Relais am Arduino auf. Achte auf die in Sperrrichtung geschaltete Diode parallel zum Motor.

Schließe dann drei Taster an (mit Widerstand! - vgl. Abschnitt 3.7.2).

Programmiere nun einen einfachen steuerbaren Waschmaschinenprototypen!

Dieser gibt solange auf dem seriellen Monitor die aktuell gesetzte Waschzeit aus, bis der mittlere Taster gedrückt wurde, was bedeutet, dass der Waschvorgang startet (der Motor dreht sich für die angegebene Zeit). Wenn der linke Taster gedrückt wird, wird die Waschzeit verringert (aber nicht niedriger als eine Sekunde). Wenn der rechte Taster gedrückt wird, wird die Waschzeit vergrößert (aber nicht größer als 30 Sekunden). Nach dem Waschen fragt der Waschmaschinenprototyp wieder nach der Waschzeit.

🔍 Recherche: Anwendungen von Relais

Recherchiere einige Anwendungen von Relais. Einen guten Startpunkt bietet die [Seite von Leiphy-sik](#).

Aufgabe 6: Vergleich von Transistor und Relais

Transistoren und Relais erfüllen im Wesentlichen die gleiche Funktion: Sie sind elektronisch steuerbare Schalter, bei denen die zu steuernde Stromstärke größer sein kann als die Stromstärke im Steuerstromkreis. Dennoch gibt es einige Unterschiede, weshalb sie für unterschiedliche Aufgaben geeignet sind.

Vergleiche die Steuerung mit einem Transistor und mit einem Relais hinsichtlich der Vor- und Nachteile.

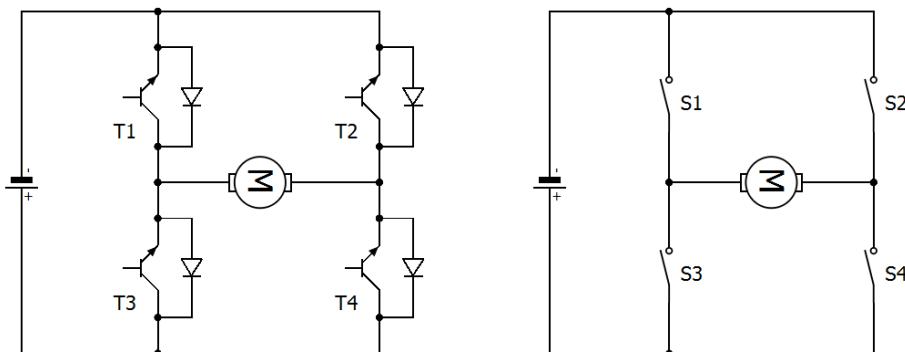
7.4. Steuerung mit dem Motortreiber-IC L293D (inkl. Drehrichtung)

Die Steuerung von Motoren erfordert in den oben beschriebenen Fällen stets mehrere Bauteile und einige Überlegungen zum Aufbau der Schaltung. Außerdem kann dabei nicht die Drehrichtung geändert werden. Der integrierte Schaltkreis L293D vereinfacht den Aufbau der Schaltung für gleich zwei Motoren und ermöglicht zusätzlich die flexible Steuerung der Drehrichtung.

Frage: Wie steuert man einen Motor mit dem L293D?

Aufgabe 7: Aufbau des L293D - der Vierquadrantensteller

Um die Drehrichtung des Motors kontrollieren zu können, braucht man eine spezielle Anordnung von Transistoren, die als *H-Brücke* oder *Vierquadrantensteller* bezeichnet wird. Dieser Aufbau befindet sich auch im L293D.

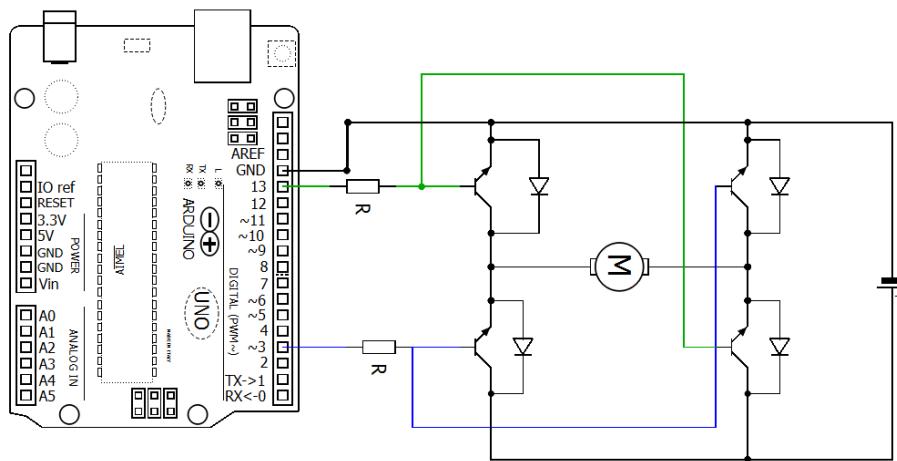


B 7.8. Vereinfachter Aufbau eines Vierquadrantenstellers mit Transistoren und zugehörigen Freilaufdioden (links) sowie die noch einmal vereinfachte Ersatzschaltung mit Schaltern.

- Die Drehrichtung des Motors hängt davon ab, in welcher Richtung der Strom durch den Motor fließt. Notiere, welche Transistoren / Schalter eingeschaltet und welche Transistoren / Schalter ausgeschaltet sein müssen, damit der Strom von links nach rechts durch den Motor fließt. Notiere danach die Kombination für die Stromrichtung von rechts nach links.
- Erkläre, wie sich der Motor mithilfe der vier Transistoren bzw. Schalter bremsen lässt.
- Welche Schaltkombinationen der Transistoren müssen unbedingt vermieden werden?

Hinweis: Die Freilaufdioden dienen dazu, die vom Motor induzierten Ströme abfließen zu lassen.

Da stets zwei Transistoren gemeinsam eingeschaltet werden müssen, könnten diese beim Anschluss an den Arduino über einen gemeinsamen Digitalpin gesteuert werden. Zudem ist es im Allgemeinen sinnvoll, für den Motor und den Arduino verschiedene Spannungsquellen zu verwenden, die über einen gemeinsamen GND-Anschluss geerdet werden, damit die möglicherweise hohen Ströme des Motors den Arduino nicht zerstören.



B 7.9. Steuerung eines Motors mit einem Vierquadrantensteller am Arduino.

Bei der oben dargestellten Schaltung muss jedoch immer noch darauf geachtet werden, dass nicht versehentlich alle vier Transistoren leitend geschaltet werden. Daher ist die Steuerung mit dem L293D noch ein wenig komplexer - die oben angestellten Überlegungen verdeutlichen aber gut den prinzipiellen Aufbau.

Der Motortreiber L293D

Der L293D ist ein integrierter Schaltkreis (*IC* von engl. *integrated circuit*), das heißt, in das schwarze Gehäuse sind Schaltkreise mit Transistoren, Widerständen, Dioden etc. integriert. Genauer gesagt, enthält der L293D zwei H-Brücken oder Vierquadrantensteller, die sich mit den Pins an beiden Seiten steuern lassen. Bei der Nummerierung der Pins ist darauf zu achten, dass die kleine Kerbe nach oben gehalten wird.



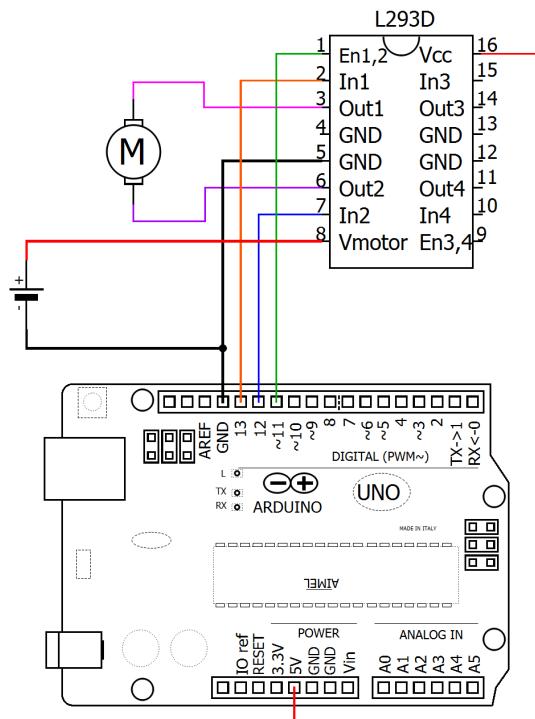
L293D	
1	En1,2
2	In1
3	Out1
4	GND
5	GND
6	Out2
7	In2
8	Vmotor
9	En3,4
10	In4
11	Out3
12	GND
13	GND
14	Out4
15	In3
16	Vcc

Im Folgenden wird die Belegung der Pins für die linke Seite beschrieben (vgl. Abbildung 7.10). Die Belegung auf der rechten Seite verläuft analog.

Der Motor wird an Pin 3 und 6 (Out1 und Out2) angeschlossen. Der jeweilige Zustand der Out-Pins kann über Pin 2 und 7 (In1 und In2) geregelt werden. Wenn an In1 der Zustand HIGH und an In2 LOW anliegt, wird das auf Out1 und Out2 übertragen, sodass durch den Motor ein Strom fließen kann. Diese Übertragung wird jedoch durch Pin 1 (En1,2 für enable pin 1, 2) gesteuert. Wenn an En1,2 HIGH anliegt, wird die Input-Konfiguration übertragen, bei LOW nicht. Durch ein PWM-Signal an En1,2 kann die Leistung des Motors entsprechend gedrosselt werden.

Die vier GND-Anschlüsse dienen zur Stromversorgung und zur Wärmeableitung, falls hohe Ströme auftreten. An Vmotor wird der Pluspol der Versorgungsspannung für den Motor angeschlossen; an

Vcc der Logik-Pegel von 5V für die Schaltung des IC.



B 7.10. Steuerung eines Motors mit dem L293D.

Aufgabe 8: Betrieb des L293D

- a) Baue die oben beschriebene Schaltung auf.

Nutze dazu das *Power Supply Module* (siehe S. 78).

- b) Experimentiere mit verschiedenen Input-Konfigurationen und PWM-Werten für den En1,2-Pin.

- c) Halte die Wirkung auf den Motor tabellarisch fest. Hier genügt es, wenn für den En1,2-Pin nur zwischen *ein* / 1 und *aus* / 0 unterschieden wird.

In1	In 2	En1,2	Wirkung
1	0	1	...



🔍 Recherche: Wie stark darf der L293D belastet werden?

Bei Motoren ist immer genau darauf zu achten, welche Stromstärken und Spannungen die verwendeten Bauteile aushalten. Suche nach dem Datenblatt (*data sheet*) des L293D und notiere die Maximalwerte zu Versorgungsspannung, Stromstärke und kurzfristige Spitzenstromstärke, die der IC aushält (*absolute maximum ratings*).

Motivationsquellen

> [Autonomes Auto](#)

Der Bastler hinter diesem Projekt hat einen Arduino-basierten Prototypen für ein autonomes Auto entworfen.

> [Pong-Bot](#)

Ein kleines witziges Spiel hat dieser Bastler mit einem Arduino automatisiert.

> [Snack-Automat](#)

Ein Arduino-basierter Snack-Automat!

8. Konzepte der Informatik II

Mit den vielen Bauteilen, die nun zur Verfügung stehen, lassen sich bereits große und komplexe Projekte realisieren. Um dabei nicht den Überblick zu verlieren, hilft es, sich mit einigen weiteren Konzepten aus der Informatik zu beschäftigen, die das Programm und die Vorgehensweise zur Erstellung des Programms besser strukturieren.

In diesem Kapitel lernst du...

- ... Programme mit Funktionen (eigenen Blöcken) aussagekräftiger und kürzer zu gestalten,
- ... die Funktionsweise von Programmen mit einem Zustandsdiagramm zu planen und zu veranschaulichen,
- ... Programme durch eine zustandsbasierte Modellierung als endlicher Automat zu flexibilisieren.

Projekte in diesem Kapitel:

> Fußgängerampel	88	> Parkplatzschanke	90
------------------------	----	--------------------------	----

8.1. Funktionen

Aufgabe 1: Funktionen verstehen

Theresa und Thorsten haben jeweils ein kleines Programm erstellt, durch das sich ein E-Motor, der über einen L293D-Motortreiber am Arduino angeschlossen ist, für drei Sekunden vorwärts drehen und dann stoppen soll. Vergleiche beide Programme.



B 8.1. Theresas Programm zum Steuern des Motors.



B 8.2. Thorstens Programm zum Steuern des Motors.

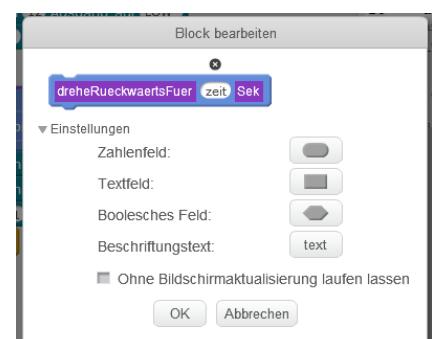
Aufgabe 2: Funktionen erstellen

Erstelle das Programm von Thorsten und ergänze eine weitere Funktion (einen neuen Block) namens

`dreheRueckwaertsFuer <zeit> Sek`. Integriere diese in die `wiederhole fortlaufend`-Schleife.

Funktionen bzw. eigene Blöcke kannst du im Bereich „Daten&Blöcke“ erstellen.

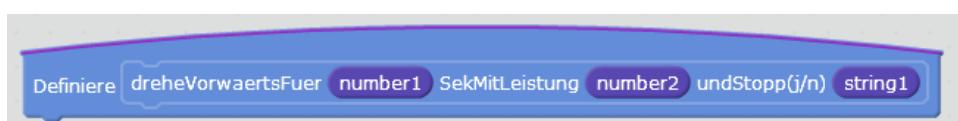
Passe die Funktionen (eigenen Blöcke) so an, dass auch die Leistung des Motors als Argument eingegeben werden kann.



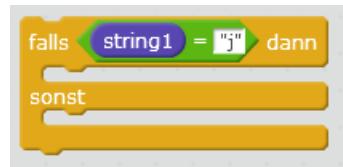
! Die Namen von Funktionen sollten keine Leerzeichen oder Umlaute enthalten!

Aufgabe 3: Argumente von Funktionen

Eine mögliche Erweiterung der Funktion wäre eine Möglichkeit, angeben zu können, ob der Motor am Ende des Durchgangs stoppen soll oder nicht. Diese Möglichkeit wird zum Beispiel bei den Blöcken von Lego Mindstorms angeboten.



Im Grunde genommen handelt es sich bei dieser Auswahlmöglichkeit um eine Variable mit zwei Werten, nämlich wahr oder falsch. mBlock lässt auch eine boolsche Variable als Argument zu, allerdings kann diese nicht direkt im Programm als wahr oder falsch angegeben werden. Stattdessen kann man eine „Ja“ / „Nein“ - Abfrage (j/n) über ein Textfeld integrieren. Bei der Abfrage ist zu beachten, dass auf beiden Seiten eine String-Variable stehen muss - dies wird durch die Anführungszeichen kenntlich gemacht.



Programmiere die oben abgebildete Funktion.

Für Schnelle: Wenn `number2` direkt als Argument für die Leistung gewählt wird (`setze PWM-Pin ... auf number2`), variieren die Werte zwischen 0 und 255. Für den Anwender wäre es aber einleuchtender, die Leistung in Prozent, also mit Werten zwischen 0 und 100, anzugeben. Ändere den Block so, dass die Leistung in Prozent angegeben werden muss.

Erweiterung: Programmiere die Funktion so, dass man auswählen kann, ob sich der Motor vorwärts oder rückwärts (v/r) dreht.

Funktionen (Eigene Blöcke)

Funktionen fassen mehrere Befehle zusammen. Häufig werden Funktionen genutzt, um ein Programm lesbarer zu machen, indem der Name der Funktion die Wirkung der Befehle zusammenfasst.

Funktionen können *Argumente* erhalten. Diese werden im Körper der Funktion verarbeitet (eingesetzt, umgerechnet, ...).

Aufgabe 4: Eine Ampel zum Üben

Baue auf dem Steckbrett eine Ampelschaltung auf und programmiere Funktionen mit einem sinnvollen Argument für die jeweiligen Phasen.

8.2. Automaten

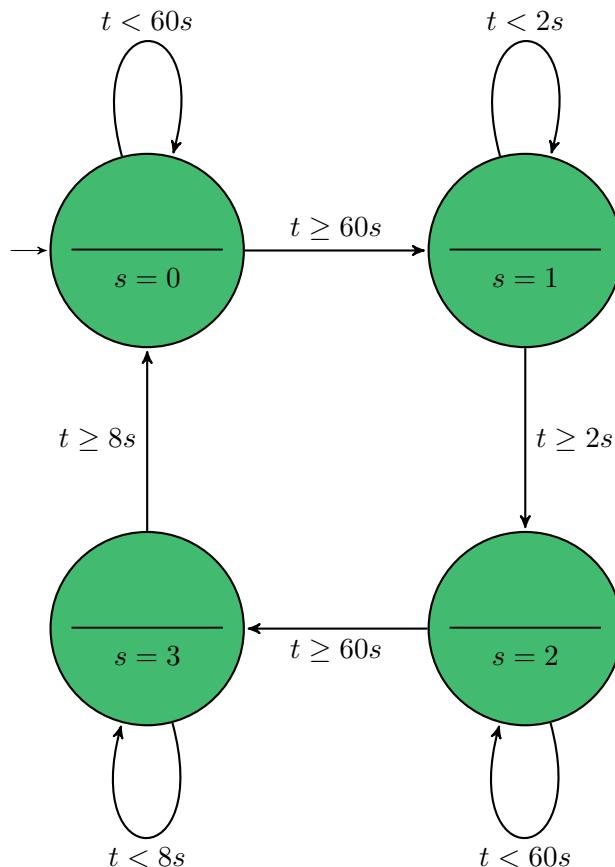
Bisher wurden Algorithmen rezeptartig als feste Handlungsabfolgen beschrieben, die nacheinander durchlaufen werden und dabei ggf. einfache Fallunterscheidungen berücksichtigen. In der Automatentheorie steht weniger das Befolgen eines Rezeptes im Mittelpunkt als das Einnehmen verschiedener Zustände, zwischen denen unter festgelegten Bedingungen Übergänge stattfinden. Dies macht die Algorithmen flexibler im Umgang mit Eingaben aus der Umwelt, wodurch sie zudem leichter zu erweitern sind.

Ziel: Durch eine zustandsbasierte Modellierung sollen Algorithmen flexibler werden und einfacher zu erweitern.

Aufgabe 5: Ampelzustände

- Nenne die vier Zustände (*engl. states*) einer Ampel. Beschreibe die Bedingung(en), unter denen der Wechsel vom einen in den nächsten Zustand stattfindet.
- Unten ist ein sogenanntes Zustandsdiagramm einer Ampel zu sehen. Erläutere, wie es aufgebaut ist und ordne ihnen die vier Ampelzustände zu.

Hinweis: s : state (*engl. für Zustand*), t : time (*engl. für Zeit*)



- Unten ist abgebildet, wie ein Programm aussehen könnte, das das Zustandsdiagramm modelliert. Jeder Übergang (jede Bedingung) wird durch eine falls-Abfrage in das Programm

integriert. Vervollständige das Programm zur Modellierung einer einfachen Ampel und bau die Ampel auf.

Erweiterung: Programmiere jeden Ampelzustand als eigene Funktion, sodass das Programm lesbarer wird.



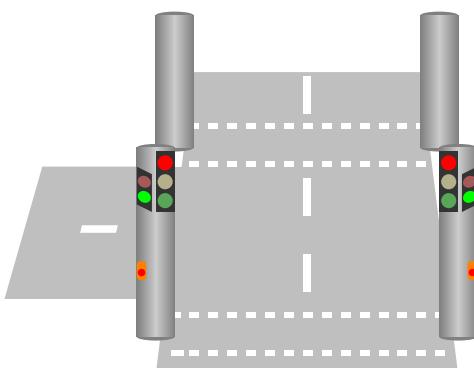
Idee: Materialien zum Kerncurriculum Informatik im Sekundarbereich I, Niedersächsisches Kultusministerium

Bei einer klassischen Programmierung mit warte-Blöcken wäre eine Erweiterung um eine Fußgängerampel, die zu jedem beliebigen Zeitpunkt aktiviert werden kann, unmöglich, da das Programm während des Wartens das Drücken der Fußgängerampel gar nicht mitbekommen würde. Durch die zustandsbasierte Modellierung mithilfe einer Stoppuhr lässt sich die Fußgängerampel jedoch integrieren.

Projekt 1:

Erweiterung um eine Fußgängerampel

Eine Fußgängerampel ist zunächst aus oder deaktiviert. Sie kann jederzeit durch einen Taster aktiviert werden, das heißt, sie zeigt rot. Sie bleibt solange rot, bis die normale Ampel ebenfalls rot zeigt und eine gewisse Mindestdauer vergangen ist. Die normale Ampel bleibt nun rot, während die Fußgängerampel auf grün springt. Nach einer gewissen Zeit schaltet die Fußgängerampel wieder auf rot. Nach ein paar Sekunden Rotphase für die letzten Fußgänger, die ihren Weg über die Straße noch beenden müssen, ist die Fußgängerampel wieder aus und die normale Ampel springt in die Rot-Gelb-Phase, mit der sie ihren normalen Rhythmus fortsetzt.



- Entnimm der obigen Beschreibung die vier Zustände einer Fußgängerampel und die Bedin-

gungen, unter denen vom einen Zustand in den nächsten gewechselt wird. Entwickle daraus ein Zustandsdiagramm für die Fußgängerampel. Ergänze die Übergänge von der Rotphase der normalen Ampel zur Rotphase der Fußgängerampel und von der (wieder) deaktivierten Fußgängerampel zur Rot-Gelb-Phase der normalen Ampel.

Achtung: Hier liegen zwei Automaten vor, nämlich die normale Ampel und die Fußgängerampel, daher gibt es auch zwei Startzustände.

- b) Ergänze die Fußgängerampel inkl. Taster auf dem Steckbrett und ergänze das Programm um die Zustände der Fußgängerampel.

Tipp: Nutze eine Variable, um den Tasterstatus zu speichern, nachdem der Taster einmal gedrückt wurde.

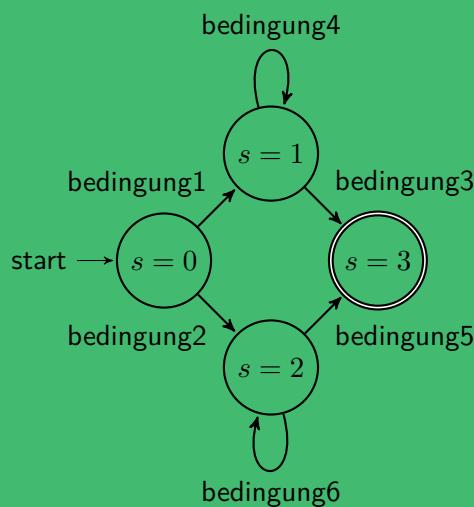
Idee: Materialien zum Kerncurriculum Informatik im Sekundarbereich I, Niedersächsisches Kultusministerium



Endlicher Automat und Zustandsdiagramm

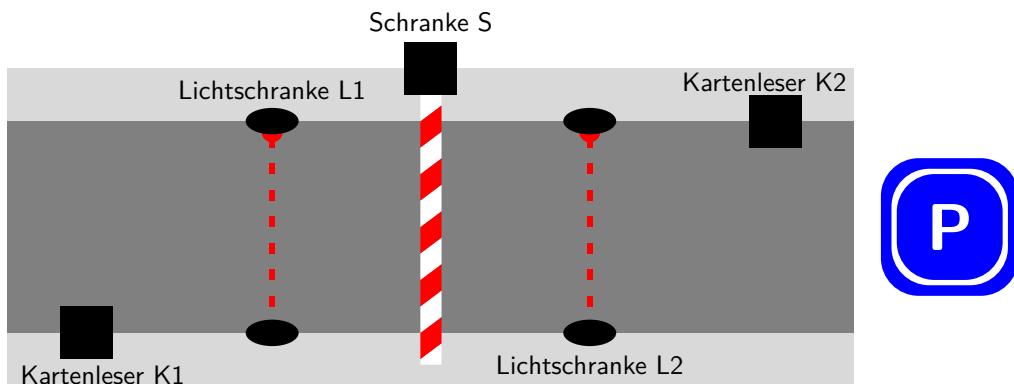
Ein Automat (auch: abstrakte Maschine) ist in der Informatik ein Modell zur Beschreibung einer Datenverarbeitung. Er befindet sich anfangs in seinem Startzustand. Abhängig von der nächsten Eingabe (z.B. das Ablaufen einer Zeit oder das Drücken eines Tasters) und des aktuellen Zustands erfolgt der Übergang in einen Folgezustand. Auch zu diesem gibt es wiederum einen Folgezustand, der abhängig von der nächsten Eingabe und dem aktuellen Zustand erreicht wird. Es kann einen oder mehrere Endzustände geben, die keinen Folgezustand haben, wenn der Ablauf des Automaten vollständig ausgeführt wurde. Wenn der Automat endlich viele Zustände hat, spricht man von einem **endlichen Automaten**.

Ein Zustandsdiagramm veranschaulicht das Verhalten eines Automaten. Die Zustände werden mit einem Kreis oder abgerundeten Rechteck dargestellt. Die Übergänge werden mit Pfeilen dargestellt, an denen die Bedingung für den Übergang steht. Der Startzustand wird mit einem zusätzlichen Pfeil markiert, an den manchmal zusätzlich `start` geschrieben wird. Der ggf. vorhandene Endzustand wird durch einen doppelten Rahmen markiert.



Projekt 2:**Parkplatzschranke**

Auf einen Parkplatz kommt man häufig erst, wenn man einen Parkschein gezogen hat. Danach öffnet sich die Schranke und man kann auf den Parkplatz fahren. Die Schranke schließt sich automatisch wieder, nachdem das Auto hindurchgefahren ist. Beim Verlassen des Parkplatzes muss man wiederum zuerst den bezahlten Parkschein einlesen lassen, bevor sich die Schranke öffnet und automatisch wieder schließt, nachdem ein Auto hindurchgefahren ist.



Das Verhalten der Parkplatzschranke soll auf dem Steckbrett simuliert werden. Dazu wird aus Pappe ein „Auto“ geschnitten, das durch die mittlere Spalte des Steckbretts „fährt“.

- Erläutere, wie man die Kartenleser und das Durchfahren der Schranke mithilfe von zwei Tastern und zwei Lichtschranken simulieren kann. Notiere alle benötigten Bauteile.
- Baue die Parkplatzschranke mit allen benötigten Teilen auf dem Steckbrett auf.
- Entwickle ein Automatenmodell für die Parkplatzschranke. Überlege dazu, in welchen Zuständen sich die einzelnen Elemente beim Einfahren und beim Ausfahren befinden.
- Implementiere das Automatenmodell.

Idee: Materialien zum Kerncurriculum Informatik im Sekundarbereich I, Niedersächsisches Kultusministerium

Motivationsquellen

> [Cocktail Mixer](#)

Dieser Automat mixt automatisch einen aus sechs wählbaren Drinks!

> [Roboterhand](#)

Mit diesem Aufbau wird die eigene Hand auf eine Roboterhand gespiegelt, wodurch sich der Roboter viel besser steuern lässt.

> [One | Aerospace](#)

Diese Gruppe von Studenten versucht u. a. mit Hilfe eines Arduino eine Rakete zu bauen!

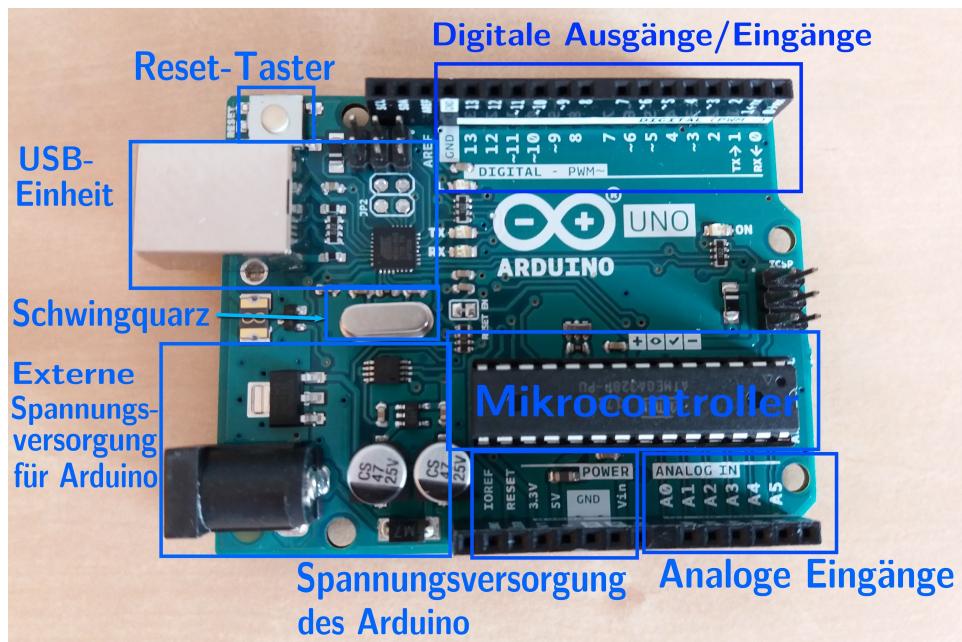
> [Ziegen-Tracker](#)

Mit dieser Kombination aus Arduino und GPS-Antenne wird es Ziegen unmöglich von ihrer Farm auszubrechen.

A. Anhang

A.1. Ein kurzer Überblick - die Funktionseinheiten des Arduino Uno

Im Folgenden wird ein vertiefter Überblick über die wichtigsten Komponenten des Arduino Uno, den sogenannten Funktionseinheiten, gegeben (vgl. Abb. A.1).



B A.1. Der Arduino Uno lässt sich in wenige Funktionseinheiten aufteilen (weitere Erklärungen im Text).

Mikrocontroller: Wenn der Arduino Uno als Mikrocontroller bezeichnet wird, dann stimmt das eigentlich nicht ganz. Der eigentliche Mikrocontroller, auf dem das Programm läuft, ist das markierte schwarze Bauteil, bei dem es sich um einen Atmega328 handelt. Die Arbeit von Massimo Banzi und David Cuartielles unter anderem bestand darin, die restlichen Bauteile, die zur Verwendung des Mikrocontrollers benötigt werden, zusammen mit dem Atmega328 auf einem Board zu befestigen. Dadurch wurde die Handhabung wesentlich vereinfacht! Auf dem Mikrocontroller befindet sich ein sogenannter Bootloader, der das Programm vom Computer in den Speicher des Mikrocontrollers lädt.

Schwingquarz: Der Atmega328P arbeitet mit einer Taktfrequenz von 16 MHz, die von dem Schwingquarz vorgegeben wird. Das heißt, es werden pro Sekunde 16 000 000 „Arbeitsschritte“ vollzogen! Natürlich sind moderne Computer noch viel schneller, aber für unsere Zwecke reicht das völlig aus.

Stromversorgungseinheit: Natürlich benötigt ein Mikrocontroller Strom oder besser gesagt eine

gewisse Spannung, um funktionieren zu können. An der schwarzen Buchse können Gleichspannungsquellen zwischen 7 V und 12 V angeschlossen werden, um den Mikrocontroller mit Energie zu versorgen. Die anderen Bauteile regeln diese Spannung dann auf die 5 V herunter, die der Mikrocontroller zum Arbeiten benötigt.

USB-Einheit: An der silbernen Buchse kann das USB-Kabel angeschlossen und der Arduino Uno somit mit dem Computer verbunden werden. Dies ermöglicht die Übertragung eines Programms auf den Arduino Uno. Außerdem lassen sich dadurch weitere Daten, z.B. von Sensoren am Arduino Uno, mit dem Computer austauschen. Der USB-Anschluss dient außerdem bereits als Spannungsversorgung. Solange der Arduino Uno am Computer angeschlossen ist, braucht man in der Regel keine Batterie als zusätzliche Spannungsversorgung. Eine Ausnahme sind Motoren, denn der USB-Anschluss stellt *maximal 500 mA* bereit und Motoren benötigen häufig mehr. Glücklicherweise ist die USB-Buchse gegen Überstrom geschützt und schaltet sich bei einem Kurzschluss automatisch ab.

Digitale Eingänge und Ausgänge: Die digitalen Pins lassen sich entweder aus Ausgänge oder als Eingänge benutzen. Wenn sie als Ausgang festgelegt werden, dann kann dort eine Spannung von 0 V oder 5 V angelegt werden. Wenn sie als Eingang festgelegt werden, können dort Spannungen von 0 V oder 5 V eingelesen werden. Einige Pins sind mit einer Tilde (~) gekennzeichnet, was bedeutet, dass sie über eine Pulsweitenmodulation verfügen. Der GND-Pin stellt den negativen Pol dar (auch Erdung oder GrouND genannt).

Hinweise:

- Die **maximale Stromstärke**, die die Digitalpins vertragen, ist **40 mA**. Empfohlen sind eher 20 mA.¹
- Pin 0 und Pin 1 sind bei angeschlossenem USB-Kabel für die Kommunikation mit dem Computer belegt und können daher nicht benutzt werden, wenn der Arduino Uno mit dem Computer kommunizieren soll!

Spannungsversorgung: Hiermit ist die Spannungsversorgung für die Bauteile gemeint, die man an den Arduino anschließt. Diese Pins lassen sich nicht über das Programm an- und ausstellen, sondern liefern permanent eine gewisse Spannung (5 V bzw. 3,3 V), die mit den GND-Pins geerdet wird. Der RESET-Pin wird in diesem Skript nicht genutzt, weil man den Reset über die Reset-Taste durchführen kann (siehe unten). Mit VIN kann eine Referenzspannung (Vergleichsspannung) eingelesen werden. Der IOREF-Pin legt fest, ob der Mikrocontroller mit 5 V oder mit 3,3 V operiert. Standardmäßig sind 5 V eingestellt.

Hinweis: Die **maximale Stromstärke**, die der 5V-Pin sowie die GND-Pins vertragen, ist 200 mA.

Analoge Eingänge: An analogen Eingängen kann eine Spannung eingelesen werden. Dabei sind nicht nur zwei Werte möglich (wie bei den digitalen Eingängen), sondern auch viele Werte zwischen 0 V und 5 V.

Reset-Taste: Mit der Reset-Taste lässt sich der Arduino neu starten. Dabei wird das Programm, das auf den Arduino geladen wurde, nicht gelöscht, sondern lediglich neu gestartet.

¹Weitere Infos finden sich unter <https://playground.arduino.cc/Main/ArduinoPinCurrentLimitations>.

A.2. Das Elegoo Starter-Kit

Bei Erstellung dieses Skripts wurde ein günstiges Starter-Kit von Elegoo genutzt, das alle grundlegenden elektronischen Bauteile enthält. Der darin enthaltene Mikrocontroller „Elegoo Uno R3“ wurde nach dem frei verfügbaren Aufbau des Arduino nachgebaut, darf aber nicht Arduino genannt werden, weil die Namensrechte bei der Firma tinker.it liegen. Der Nachsatz *Uno R3* spielt auf die aktuelle Version des am weitesten verbreiteten Arduino Uno R3 an. Es folgt eine Übersicht aller Bauteile des Starter-Kits:

- 1 x UNO R3 Mikrocontroller,
- 1 x LCD1602 Display (mit fertig verlötem Pin Header),
- 1 x DHT11 Modul (Temperatur- und Feuchtigkeitssensor),
- 1 x Joystick-Modul,
- 1 x 5V Relais,
- 1 x MB-102 Breadboard (Steckbrett),
- 1 x Servomechanismus(SG90),
- 1 x Schrittmotor,
- 1 x ULN2003 Schrittmotor-Treibermodul,
- 1 x Prototyp-Erweiterungsplatine,
- 1 x Power Supply Module,
- 1 x Ultraschall-Sensor-Modul,
- 1 x IR-Empfängermodul,
- 1 x IR-Fernbedienung,
- 1 x 3V Gleichstrommotor,
- 1 x USB Kabel,
- 1 x 65 M-M Kabel,
- 1 x 10 Female-to-Male Kabel,
- 1 x 9 V Akku mit DC,
- 1 x Kugelschalter,
- 1 x Segmentanzeige,
- 1 x 4x7-Segment-Anzeige,
- 1 x IC 74HC595 (Schieberegister),
- 1 x Aktiver Summer,
- 1 x Passiver Summer,
- 1 x Potentiometer,
- 1 x Thermistor,
- 1 x Diode Rectifier (1N4007),
- 2 x NPN Transistor (pn2222),
- 2 x LDR (Fotowiderstand),
- 1 x RGB LED,
- 5 x weiße LED,
- 5 x gelbe LED,
- 5 x Blaue LED,
- 5 x grüne LED,
- 5 x rote LED,
- 5 x Druckschalter,
- 10 x 10 Ohm-Widerstand,
- 10 x 100 Ohm-Widerstand,
- 30 x 220 Ohm-Widerstand,
- 10 x 330 Ohm-Widerstand,
- 10 x 1k Ohm-Widerstand,
- 10 x 2k Ohm-Widerstand,

- 10 x 5k1 Ohm-Widerstand,
- 10 x 10k Ohm-Widerstand,
- 10 x 100k Ohm-Widerstand,
- 10 x 1m Ohm-Widerstand,
- 1 x Fan,
- 1 x L293D (4 Kanal Treiber mit Diode).

Zusätzlich wurden Bewegungsmelder angeschafft.

A.3. Installation der integrierten Entwicklungsumgebung (IDE) für den Arduino

Üblicherweise wird der Arduino mit der integrierten Entwicklungsumgebung (IDE - engl. *integrated development environment*), die ebenfalls Arduino heißt, programmiert. Diese verfügt über einen Texteditor, in dem das Programm geschrieben werden kann. Zusätzlich lassen sich hier einfach die Bibliotheken (bei mBlock „Extensions“) und weiteren Boards verwalten und einbinden. Wichtig ist zudem die Syntax-Prüfung sowie letztendlich die Kompilation des Programms. Das Programm wird in der maschinennahen Programmiersprache C++ geschrieben, für die jedoch einige vorgefertigte Befehle für den Arduino bereitgestellt werden, damit der Einstieg leichter fällt. Das Kompilieren des Programms macht aus dem Programmcode einen maschinenlesbaren Code (von Einsen und Nullen), der dann auf den Arduino übertragen wird.

Die IDE lässt sich im Internet auf der Projekthomepage für jedes Betriebssystem herunterladen:

<https://www.arduino.cc/en/Main/Software>.

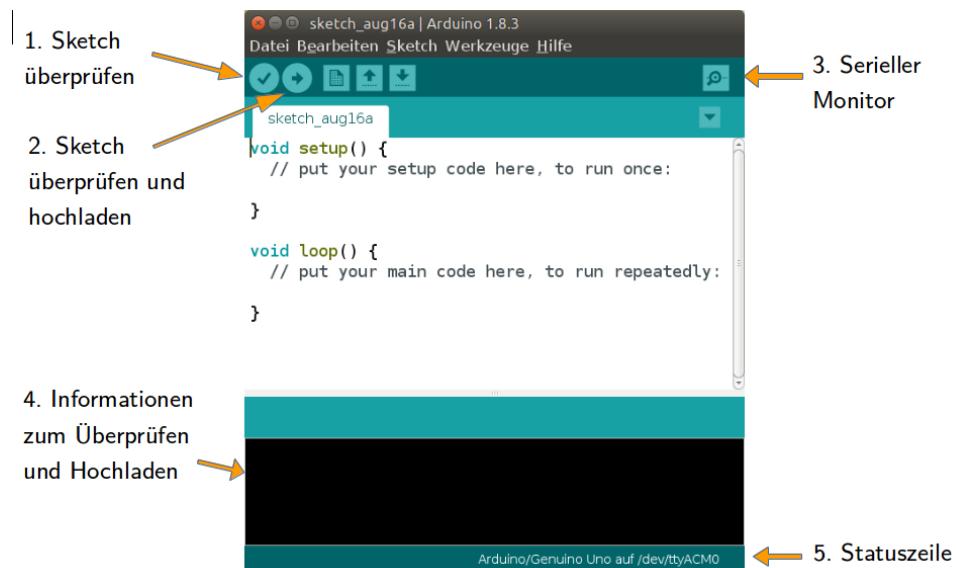
Auf derselben Seite gibt es zudem ausführliche Anleitungen zur Installation (englisch):

<https://www.arduino.cc/en/Guide/HomePage>.

Eine ebenfalls ausführliche und deutschsprachige Installationsanleitung wird auf der CD des Starter Kits mitgeliefert.

A.4. Mit der Arduino IDE ein Programm auf den Arduino Uno übertragen

Nach Installation der Entwicklungsumgebung kann das erste, noch leere Programm auf den Arduino Uno übertragen werden, um zu testen, ob und wie das Ganze funktioniert. Dazu wird der Arduino Uno mit dem USB-Kabel an den PC angeschlossen und die Arduino-Entwicklungsumgebung geöffnet. Wenn der Arduino Uno zum ersten Mal an den PC angeschlossen wird, muss evtl. kurz gewartet werden, damit die Treiber für die Kommunikation von PC und Arduino Uno installiert werden. Das passiert aber vollautomatisch.



B A.2. Ein Überblick über die Arduino-Entwicklungsumgebung.

In der Mitte des Fensters der Entwicklungsumgebung wird das Programm geschrieben. Innerhalb der geschweiften Klammern von `void setup()` werden Anweisungen geschrieben, die ganz am Anfang genau einmal ausgeführt werden. Im Setup befindet sich bereits ein Kommentar, der sich daran erkennen lässt, dass er eine graue Farbe hat. Kommentare werden im Programm dadurch gekennzeichnet, dass am Anfang zwei Schrägstriche stehen. Sie sind nützlich, um in komplizierteren Programmen auch später noch den Überblick zu behalten und um anderen, die das Programm nutzen wollen, die Möglichkeit zu geben, es zu verstehen.

Innerhalb der geschweiften Klammern von `void loop()` stehen Anweisungen, die immer wieder wiederholt werden. Die Anweisungen werden der Reihe nach abgearbeitet und nach dem letzten Befehl beginnt der Arduino Uno wieder mit dem ersten Befehl innerhalb des Loops, der auf Deutsch entsprechend Schleife heißt. In dieser unendlichen Schleife wird der Hauptteil des Programms stehen. Die wesentlichen anderen Bestandteile der Entwicklungsumgebung werden im Folgenden erklärt.

1. Sketch überprüfen: Mit dem Haken kann überprüft werden, ob das Programm, das in der Arduino-Welt auch Sketch genannt wird, richtig geschrieben ist. Damit ist natürlich nicht der Inhalt gemeint, denn die Entwicklungsumgebung weiß natürlich nicht, was das Programm bewirken soll. Es wird nur überprüft, ob die „Grammatik“ (Fachwort: die Syntax) des Programms richtig ist. So wie in einem Satz die Wörter in einer bestimmten Reihenfolge stehen und ggf. mit Kommata getrennt werden müssen, müssen in einer Programmiersprache Befehle korrekt geschrieben, geöffnete Klammern wieder geschlossen und das Ende von Befehlen mit einem Semikolon deutlich gemacht werden. Genau das wird mit dem Haken überprüft.

2. Sketch überprüfen und hochladen: Mit dem Pfeil wird der Sketch nicht nur überprüft, sondern auch auf den Arduino Uno hochgeladen.

3. Serieller Monitor: Mit dem seriellen Monitor kann man Daten vom Arduino Uno abfragen und am Computer anzeigen lassen. Man kann auch umgekehrt Daten am Computer eingeben und vom Arduino auslesen lassen. Wenn man ihn entsprechend programmiert, kann man dem Arduino auf

diese Weise auch während der Durchführung des Programms noch Befehle geben.

4. *Informationen zum Überprüfen und Hochladen:* Beim Überprüfen und Hochladen zeigt die Entwicklungsumgebung in dem schwarzen Bereich Informationen zum Fortschritt und ggf. zu Fehlern an.

5. *Statuszeile:* In der Statuszeile sieht man, welcher Arduino-Typ ausgewählt wurde (in diesem Skript ist das immer der Arduino Uno) und an welchem USB-Port er sich befindet. Dargestellt ist der Pfad unter Linux - unter Windows werden die Ports mit COM1, COM2, ... bezeichnet.

Fehlerursachen

Eine häufige Fehlerursache, die das Hochladen verhindert, ist, dass kein oder ein falscher USB-Port ausgewählt wurde. Der USB-Port muss unter Werkzeuge -> Port ausgewählt werden. In der Regel ist dort nur der eine Port auswählbar, an dem sich der Arduino Uno befindet. Falls kein Port auswählbar ist, fehlt der Treiber für die Kommunikation von Arduino und Computer. Er muss dann manuell nachinstalliert werden.

B. Didaktisch-Methodische Bemerkungen

Vor etwa drei Jahren, also im Jahr 2016, bin ich das erste Mal in einem Youtube-Video auf den Arduino aufmerksam geworden. Darin wurde er für ein Heimwerker-Projekt genutzt und als einfache Möglichkeit vorgestellt, programmieren zu lernen und Projekte umzusetzen. Ich war schnell begeistert und schaffte mir selbst ein Arduino-Starter-Kit an. Ich arbeitete mich durch einige Tutorials und lernte schnell, immer komplexere Projekte aufzubauen und zu programmieren - dabei war mein Studium in Mathematik und Physik sicherlich hilfreich. Aber endlich war das ganze Wissen, das ich mir angeeignet hatte, nicht nur theoretisch wertvoll, sondern auch ganz praktisch anwendbar! Diese Mischung aus theoretischem Wissen, praktischem Aufbau, Verständnis für den technischen Alltag sowie insbesondere der Möglichkeit, *eigene Projekte anzugehen und kreativ zu werden*, war es, die mich tief begeisterte und die ich bald weitergeben wollte.

Der Bezug zur Schule ist offensichtlich. Nachdem man ein wenig in die Arduino-Welt eingetaucht ist, erscheint es fast fahrlässig, dass man im Physikunterricht Halbleiterbauelemente behandelt, ohne gleichzeitig zu thematisieren, wie man sie über ein Programm anspricht und ausliest, sodass sie zu etwas Praktischem beitragen können.¹ Zudem werden sie in Plastikkästen versteckt, die zwar verhindern, dass etwas kaputt geht und für wenige Cent neu beschafft werden muss, aber auch dazu führt, dass die meisten Schülerinnen und Schüler kaum wissen, wie die elektrischen Bauteile, die sie im Bändermodell erklären sollen, aussehen. Auf diese Art und Weise kann die Elektronik nur furchtbar theoretisch und dementsprechend wenig motivierend werden. Und so versuchte ich mich schon bald an der Integration des Arduino in den Unterricht - zunächst ein kleines Projekt im Physikunterricht, dann eine Arduino-AG und schließlich ein Arduino-Wahlpflichtfach im MINT-Profil.

Was mir für den Unterricht noch fehlte, war ein Skript wie dieses, welches das theoretische Wissen, das sich rund um den Arduino vermitteln lässt, didaktisch aufbereitet und strukturiert, sodass es sich im Unterricht anwenden lässt. Im Gegensatz zu den vielen Internet-Tutorials sollte zum Einen eine graphische Programmiersprache im Vordergrund stehen, weil meine Erfahrungen im Unterricht zeigten, dass die Schüler bei ihrer ersten Begegnung mit dem Programmieren größere Probleme mit der komplexen Syntax von C++ bekamen. Zum Anderen sollte das Wissen mit den Schülern erarbeitet werden statt es einmal zu erklären und dann nachmachen zu lassen.

So verwendete ich im Schuljahr 2018/19 sehr viel Zeit und Arbeit in dieses Skript. Mein Ziel ist, meine Begeisterung für die Arduino-Welt mit der beschriebenen Mischung aus theoretischem Wissen, praktischem Aufbau, Verständnis für den technischen Alltag sowie der Möglichkeit eigene Projekte umzusetzen an möglichst viele Schülerinnen und Schüler weiter zu geben. Aus diesem

¹Dass man stattdessen auch eine Transistorschaltung o. ä. aufbauen kann, ist mir bewusst, aber da diese recht kompliziert werden, werden Transistorschaltungen wenn überhaupt erst ganz am Ende der Halbleiterelektronik behandelt.

Grund veröffentliche ich dieses Skript auch unter einer CC-Lizenz ([BY-NC-SA 4.0](#)), die es jedem erlaubt, dieses Skript zu vervielfältigen und anzupassen, solange mein Name genannt und mögliche Anpassungen unter der gleichen Lizenz weitergegeben werden, die zudem enthält, dass dieses Skript (von Verlagen etc.) nicht kommerziell vertrieben werden darf.

B.1. Zielgruppe

Das Skript wurde für das MINT-Profil im Jahrgang 10 geschrieben. Die Schülerinnen und Schüler haben bis dahin den gewöhnlichen Physikunterricht zu Stromstärke, Spannung und Widerstand gehabt. Die Behandlung von Halbleitern erfolgt bei uns ebenfalls im Jahrgang 10, also parallel zu diesem Kurs. Dies ist sicherlich hilfreich, aber nicht notwendig für den Einsatz dieses Skripts. Dieses Skript behandelt nur die elektrotechnische Anwendung von Halbleiterelementen, nicht jedoch die Funktionsweise im Bänder- oder Teilchenmodell, die im Physikunterricht erläutert wird. Viele der teilnehmenden Schülerinnen und Schüler haben vorher bereits im MINT-Profil mit LegoMindstorms-Robotern gearbeitet und dadurch etwas Programmiererfahrung gewonnen. Dies ist ebenfalls hilfreich, aber nicht notwendig für den Einsatz dieses Skripts. Es sind keine vorherigen informatischen Kenntnisse notwendig!

Neben Jahrgang 10 erscheint es auch denkbar, dieses Skript in Jahrgang 9 oder höheren Jahrgangsstufen einzusetzen, da dort bereits (fast) alle physikalischen und mathematischen Grundlagen gelegt sind - ggf. ist bei der Regression zum Messen von Helligkeit oder Temperatur eine Anpassung notwendig.

B.2. Organisatorisches: Arduino Starter Kits, Raum, Zusammenarbeit

Für den Wahlpflichtkurs wurden 15 Arduino Starter Kits angeschafft, die neben einem Arduino-Klon auch fast alle hier beschriebenen Bauteile enthielten. Zusätzlich wurden Bewegungsmelder und USB-Kabel mit 2 m Länge angeschafft. Da die mitgelieferten Boxen sehr eng gepackt sind und nie wieder ordentlich zusammengepackt werden, wurden zusätzlich größere Aufbewahrungsboxen angeschafft, in die später auch die aufgebauten Schaltungen gesteckt werden konnten, sodass sie am Anfang der nächsten Stunde direkt wieder zur Verfügung standen.

Die Schüler bekamen jeweils zu zweit eine Box zugewiesen, auf die eine Nummer geschrieben wurde, sodass die Namen der Schüler und die zugehörige Box festgehalten werden konnten. Damit waren die Schüler auch für den Inhalt der Box verantwortlich. Im Unterricht arbeiteten die Schüler in den Praxisübungen dann auch stets zu zweit zusammen. Die Arbeit zu zweit kann gut dazu genutzt werden, dass einer hauptsächlich für die Schaltung und einer hauptsächlich für das Programmieren zuständig ist. Der Unterricht fand stets im Computerraum statt, da Computer für die Arbeit mit dem Arduino unerlässlich sind. Aus diesem Grund ist dieses Skript auch nicht zum Ausdrucken gedacht.

B.3. Zum Aufbau dieses Skripts

B.3.1. Aufbau der Kapitel

Die Einführung von Arduino, Steckbrett, LEDs mit kurzem und langem Bein, Widerständen mit Ablesen der Ringe und Programmierung kann am Anfang recht komplex sein, daher erfolgt nach der Einleitung in Kapitel 1 eine kurze Einführung in die Programmierumgebung mBlock ohne Arduino. Erst danach beginnt das Kapitel zu digitalen Ausgängen und Eingängen. In diesem stehen die physikalischen Grundlagen zu Widerstand, Spannung und Stromstärke im Vordergrund, die ein grundlegendes Verständnis der Schaltungen und der zu beachtenden Vorsichtsmaßnahmen (kein Kurzschluss, keine zu hohen Stromstärken) erzeugen sollen. Auf Programmierebene werden zunächst lediglich Sequenzen von Befehlen in einer Endlosschleife benötigt. Gegen Ende des Kapitels taucht zudem die erste Verzweigung auf.

Die Programmierebene wird im Kapitel zu Bausteinen von Algorithmen sukzessive erweitert. Hier steht das Kennenlernen von verschiedenen Programmierstrukturen, die mBlock anbietet im Vordergrund. Nach der Einführung von Variablen und Schleifen erfolgt die Einführung von Zufallselementen sowie des seriellen Monitors, der die Kommunikation von Arduino zum Computer ermöglicht. Der umgekehrte Weg von Computer zu Arduino wird von mBlock zwar auch angeboten, jedoch habe ich diesen Weg nicht ans Laufen gekriegt - hier darf sich jeder gerne noch einmal ausprobieren und sich an mich wenden, wenn es geklappt hat. Es folgt ein theoretischer Exkurs zu Binärzahlen und zur ASCII-Tabelle, der einen Einblick bieten soll, was es mit den Einsen und Nullen auf sich hat, und zudem eine Grundlage für die Behandlung von Hexadezimalzahlen im folgenden Kapitel bietet. Der letzte Abschnitt zu Struktogrammen ist inhaltlich von den anderen Abschnitten losgelöst und kann an beliebiger Stelle des Kapitels eingestreut werden. Er wurde hauptsächlich im Hinblick auf eine Klausur eingefügt und kann dementsprechend je nach Klausurtermin eingeführt werden.

Das Kapitel zu analogen Ausgängen und Eingängen erweitert die bisherigen Grundlagen sowohl bzgl. der Informatik (RGB-Farbmodell, Hexadezimalzahlen, logische Operationen) als auch bzgl. der Physik (Spannungsteiler, Potentiometer, LDR, NTC, Transistor). Der Schwerpunkt liegt hier eher auf der physikalischen Seite, insbesondere auf dem Verständnis des Spannungsteilers, der in verschiedenster Form immer wieder auftaucht. Die Formel zum Spannungsteiler wird dabei immer in der Form

$$\frac{U_1}{U_2} = \frac{R_1}{R_2} \quad \text{oder} \quad \frac{U_1}{R_1} = \frac{U_2}{R_2} = \frac{U_{ges}}{R_{ges}}$$

belassen, ohne bereits $U_1 = U_{ges} - U_2$ o. ä. einzusetzen, weil sich gezeigt hat, dass die Schüler dies mit konkreten Zahlen sehr gut schrittweise hinkriegen und flexibel an die Situation anpassen können, was bei einer allgemeinen Formel nicht der Fall war.

Kapitel 3 (Digitale Ausgänge und Eingänge) bis 5 (Analoge Ausgänge und Eingänge) bilden zusammen eine Art „Grundkurs Arduino“, der aufeinander aufbaut und nach dessen Abschluss alle wesentlichen Grundlagen zur Arbeit mit den Arduino sowie zum Kennenlernen neuer Bauteile gelegt sind.

Das Kapitel zur Erweiterung des Werkzeugkastens nimmt eine Sonderrolle ein. Hierin werden zahlreiche neue Bauteile aus dem verwendeten Starter Kit eingeführt, für die es jedoch häufig nicht

notwendig ist, den ganzen „Grundkurs Arduino“ durchlaufen zu haben. Daher ist das Skript modular aufgebaut. Es enthält an vielen Stellen in Kapitel 3 bis 5 Links, die auf Bauteile aus diesem Kapitel verweisen, wenn die dafür notwendigen Grundlagen gelegt sind. Auf diese Weise kann der „Grundkurs“ immer wieder aufgelockert werden, weil die Bauteilkunde meist weniger Theorie, aber dafür fast immer ein neues Projekt enthält.

Das Kapitel zu Elektromotoren wiederum führt zwar ebenfalls neue Bauteile ein, enthält aber wieder mehr Theorie aufgrund der auftretenden Induktion, wobei diese nur im praxisrelevanten Ausmaß behandelt wird.

Als (bisher) letztes Kapitel folgen weitere Konzepte aus der Informatik, nämlich die Einführung von Funktionen (eigenen Blöcken) und Automaten. Hier wurde versucht, das Informatik-KC (in Niedersachsen) zu berücksichtigen, das die Thematisierung von Automaten fordert. Es ist geplant, einen weiteren Abschnitt zum Steuern und Regeln hinzuzufügen.

B.3.2. Erklärung zu Links, Symbolen und eingebetteten Arbeitsblättern

Dieses Skript enthält verschiedenfarbige Links, die teilweise mit Symbolen ergänzt sind und teilweise zu eingebetteten Arbeitsblättern führen.

Links

Dunkelroter Link	Link innerhalb des Dokuments, z. B. zu einem anderen Abschnitt
Dunkelvioletter Link	Link ins Internet
Dunkelblauer Link	Link zu einer eingebetteten Datei, die als Arbeitsblatt oder Folie verwendet werden kann

Symbole

- ⌚ kennzeichnet Links, die an eine frühere Stelle im Dokument führen
- 🔧 kennzeichnet Links, die zu einem späteren Abschnitt führen, an denen ein Bauteil eingeführt wird, für das nun alle Grundlagen gelegt wurden
- 🖨 kennzeichnet Links zu eingebetteten Arbeitsblättern, die ausgedruckt werden können
- ▶ kennzeichnet Internet-Links, die zu einem passenden Video führen
- 📽 kennzeichnet Links zu eingebetteten Folien, die per Beamer gezeigt werden können und die Aufmerksamkeit auf das Wesentliche richten sollen
- ❗ Hinweis auf ein Arbeitsblatt, das die ganze oder Teile der Lösung enthält. Daher ist das Ausrufezeichen selbst der Link zu dem Arbeitsblatt, sodass man den Link nicht so leicht als solchen erkennt. Wenn kein Link vorliegt, handelt es um einen wichtigen Hinweis zum Text.

B.4. Erfahrungsbericht

Ich habe dieses Skript im Laufe des Schuljahres 2018/19 geschrieben und in meinem Kurs mit 16 Schülern eingesetzt und ausprobiert. Dadurch sind bereits viele kleine Anpassungen in das Skript eingeflossen, um kleine Fehler zu beseitigen, Missverständnisse zu umgehen oder Hinweise besser zu platzieren.

Im ersten Halbjahr habe ich Kapitel 1 bis 4 (Bausteine von Algorithmen) behandelt und darüber auch eine Arbeit schreiben lassen. Dies hat insgesamt gut geklappt. Die vermischten Übungen am Ende von Kapitel 3 und 4 dienten zur Vorbereitung auf die Arbeit. Die Lösungen dazu, die auch in diesem Paket enthalten sind, habe ich den Schülern vor der Arbeit ebenfalls zur Selbstkontrolle zur Verfügung gestellt. Auf die Verweise zu zusätzlichen Bauteilen in Kapitel 6 habe ich in der letzten Doppelstunde vor den Weihnachtsferien zurückgegriffen.

Ich bin bereits im ersten Halbjahr mit Kapitel 5 zu analogen Ausgängen und Eingängen angefangen, welches sich über einen recht langen Zeitraum bis Anfang März zog. Über Kapitel 5 wurde auch die Klausur geschrieben, weshalb hier wiederum vermischte Übungen (und eine Extradatei mit Lösungen) bereitstehen. Weitere Bauteile aus Kapitel 6 habe ich selbstgesteuert erarbeiten lassen, um Freiraum zum Experimentieren zu geben. Die Recherche-Aufträge kamen dabei letztlich nicht zum Einsatz. Erst in Kapitel 7 zu Elektromotoren habe ich den Unterricht wieder stärker kontrolliert und für einen gemeinsamen Vergleich gesorgt. Zum Ende des Schuljahres hin haben wir in einem gemeinsamen Projekt eine Wasserrakete mit Sensoren bestückt, sodass wir letztlich nur noch eine kurze Einführung zu Funktionen hatten. Den Abschnitt zu Automaten konnte ich dadurch noch nicht im Unterricht testen.

Insgesamt haben die teilnehmenden Schüler den Unterricht zum Arduino interessiert mitgemacht. Es kommen immer wieder Fragen auf, ob man dies oder das nachbauen könnte oder wie etwas genau funktioniert. Zudem ist immer wieder zu bemerken, dass sich einzelne Schüler selbst ein Arduino-Kit zulegen und damit zu Hause experimentieren. So macht Unterricht Spaß!

B.5. Ausblick

Ein Schuljahr ist mit diesem Skript zwar schon gut gefüllt, aber ich habe schon ein paar Ideen zur Erweiterung des Skripts, die entweder von interessierten Schülern zu Hause gelesen oder im Unterricht genutzt werden können, wenn vorherige Abschnitte übersprungen oder kurz gehalten werden. Dazu gehört ein Abschnitt zu „Steuern und Regeln“ im letzten Kapitel „Konzepte der Informatik II“ sowie ein Kapitel zur Spannungsversorgung des Arduino (Batterie, Spannungsregler, Akku, Laderegelung, Solarzellen) und schließlich ein Einstieg in das textbasierte Programmieren.

Außerdem plane ich dieses Skript von mBlock 3 auf das neuere mBlock 5 umzustellen. Dies soll Ende des Schuljahres 2019/20 abgeschlossen werden.

B.6. Anpassung / Weiterentwicklung

Wer das Skript für seine Zwecke anpassen oder gar weiterentwickeln will, ist herzlich dazu eingeladen, benötigt allerdings einige Kenntnisse zu L^AT_EX. Dazu muss man sich außerdem die Entwicklerversion des Arduino-Skripts, das alle tex-Dateien, Bilddateien etc enthält, herunterladen.

Ich habe dieses Skript mit der TeX Live - Distribution und Texstudio als Editor verfasst. Es ist in mehrere tex-Dateien aufgeteilt, die sich im Hauptordner sowie im Ordner `src` befinden. Als Erstes muss die Präampel vorkompiliert werden. Dazu wählt man in Texstudio `Options - Configure Texstudio`. Beim Untermenü `Commands` gibt man folgenden Befehl für pdflatex ein:

```
pdflatex -ini -jobname="preamble" "&pdflatex preamble.tex\dump"
```

Nun öffnet man die Datei `preamble.tex` und kompiliert diese mit pdflatex (standardmäßig drückt man dazu F5). Danach ändert man den Befehl von pdflatex wieder zurück auf:

```
pdflatex -synctex=1 -interaction=nonstopmode --shell-escape %.tex
```

Nun kann die Hauptdatei `arduino-skript.tex` oder auch jede beliebige Teildatei mit pdflatex kompiliert werden (standardmäßig F5). Nach dem Vorkomplizieren der Präambel muss das Hauptdokument ggf. zwei Mal kompiliert werden, damit alles funktioniert.

Weitere Programme:

- Zeichnungen habe ich in der Regel mit TikZ erstellt, für das es ein praktisches kleines Programm namens *KTikz* (Linux) bzw. *QTikz* (Windows) gibt, mit dem man die Zeichnung programmieren kann und dabei stets sieht, wie der aktuelle Code aussieht.
- Schaltpläne habe ich mit *QElectroTech* erstellt. Die vorhandenen Schaltpläne können damit geöffnet und bearbeitet werden. Dazu sollten auch die selbst erstellten Bauteile im Ordner `QET-Bauteile` importiert werden.