Universität Stuttgart
Institut für Erziehungswissenschaft

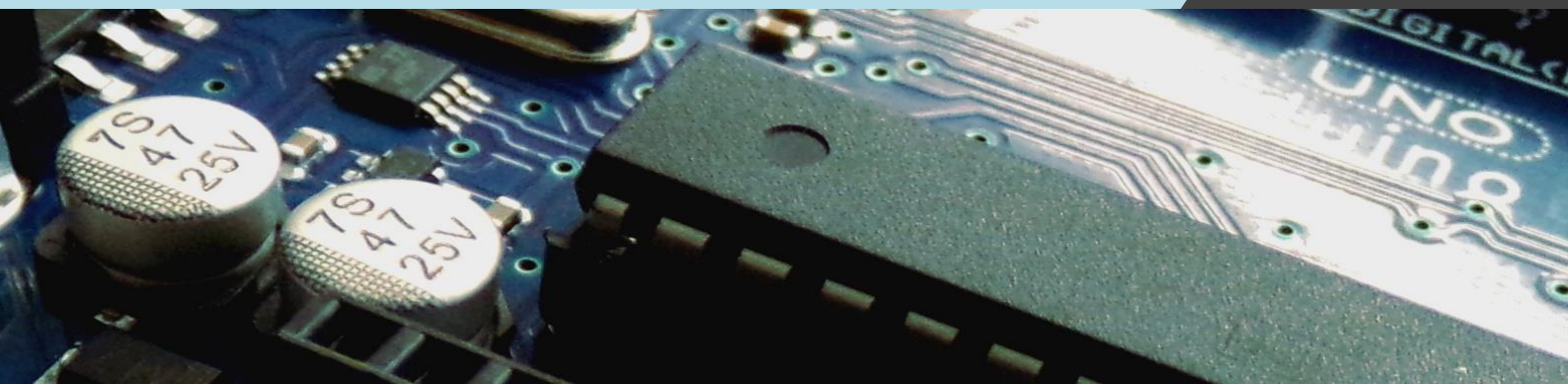
MINT Teacher Lab

Lehr- und Lernmaterialien für
Naturwissenschaft und Technik (NwT)

Rieck | Winter

Mikrocontroller

Stuttgart, Juli 2019



Redaktionelle Bearbeitung

Wissenschaftliche Leitung	Prof. Dr. Bernd Zinn, Universität Stuttgart
Autoren	Ann-Kathrin Winter und Lisa Rieck
Inhaltliche / fachliche Unterstützung	Mira Latzel und Matthias Hedrich
Hilfskräfte	Marcus Brändle
Lektorat	Mira Latzel, Matthias Hedrich und Marcus Brändle

Die vorliegenden Lehr- und Lernmaterialien zum Themenbereich *Mikrocontroller* wurden am Institut für Erziehungswissenschaft der Universität Stuttgart entwickelt und fokussieren die Weiterbildung von Lehrkräften im gymnasialen Unterrichtsfach Naturwissenschaft und Technik (NwT). Die Materialentwicklung erfolgte im Rahmen der Qualitätsoffensive Lehrerbildung im Projekt *Lehrerbildung Plus* mit einer Förderung durch das Bundesministerium für Bildung und Forschung (FKZ: 01JA1607A). Die Weiterbildung von Lehrkräften erfolgte im Projekt *MINT Teacher Lab* an der Universität Stuttgart. Das *MINT Teacher Lab* wird von der Vector Stiftung gefördert und sieht neben der Konzeptionierung eines modernen schulischen Klassenraums die Unterstützung der professionsorientierten Lehreraus- und Weiterbildung in den MINT-Lehramtsfächern im Großraum Stuttgart-Ludwigsburg vor.

Die Lehr- und Lernmaterialien umfassen eine grundlegende Einführung in die Arbeit mit dem Mikrocontroller-Board *funduino UNO R3*. Auf diesem Board befindet sich ein gängiger Mikrocontroller, der kompatibel mit der *Arduino* Software ist. Das Board ist sowohl für Einsteiger als auch Fortgeschrittene geeignet. Ohne spezielles Vorwissen und einer kurzen Einführung in die Programmierung des Mikrocontroller kann dieser im Unterricht eingesetzt werden. Innerhalb der zweitägigen Weiterbildung werden exemplarisch zwei unterrichtspraktische Projekte rund um den Mikrocontroller vorgestellt und mit den Teilnehmerinnen und Teilnehmern praktisch umgesetzt. Das erste Projekt *Ampelanlage* ist vom Anspruch primär für den Mittelstufenunterricht geeignet und beinhaltet die grundlegende Vorgehensweisen zur Ansteuerung von Schaltungsanlagen. Das zweite Projekt *Intelligentes Haus* ist eher für Schülerinnen und Schüler in der Kursstufe (Oberstufe) konzipiert und beinhaltet komplexere Programmierarbeiten. Die Lernenden können hier vertiefte Kenntnisse und Fähigkeiten zur Nutzung von unterschiedlichen Sensoren und deren Signalübertragung auf diverse Aktoren entwickeln. Inhaltliche Bezugspunkte umfassen zentrale Inhalte des Bildungsplans NwT und dabei insbesondere auch die physikalischen Aspekte zu Schaltkreisen und elektrischen Bauteilen sowie deren Anwendung im Alltag.

Die Informationen, welche in diesem Skript zusammengetragen wurden, sind sorgfältig erarbeitet worden. Jedoch können wir Fehler nicht komplett ausschließen. Wir als Autoren und Herausgeber übernehmen keine juristische Haftung und Verantwortung für eventuelle Fehler und deren Folgen. Die Bildrechte liegen bei den Autoren, außer die Abbildungen welche mit *fritzing* vermerkt sind.

Stand: August 2018

Impressum

Herausgeber:

Prof. Dr. Bernd Zinn und Mira Latzel
MINT-Teacher Lab
Universität Stuttgart
Institut für Erziehungswissenschaft
Berufspädagogik mit Schwerpunkt Technikdidaktik (BPT)
Azenbergstraße 12
70174 Stuttgart
Internetseite: http://www.uni-stuttgart.de/bpt/
E-Mail: mtl@ife.uni-stuttgart.de

Druck und Vertrieb:

MINT-Teacher Lab
Universität Stuttgart
Institut für Erziehungswissenschaft
Berufspädagogik mit Schwerpunkt Technikdidaktik (BPT)
Azenbergstraße 12
70174 Stuttgart
Internetseite: http://www.uni-stuttgart.de/bpt/
E-Mail: mtl@ife.uni-stuttgart.de

Lehrerbildung PLUS

MINT Teacher Lab

Gefördert vom

Bundesministerium für Bildung und Forschung

Gefördert von

vector Stiftung

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Abgleich Bildungsplan 2016

In diesem Kapitel wird der Bezug der Weiterbildungsinhalte zum aktuellen Bildungsplan 2016 im gymnasialen Fach Naturwissenschaft und Technik (NwT) des Bundeslands Baden-Württemberg aufgeführt. Die Inhalte der Weiterbildung *Mikrocontroller* wurden dafür mit den im Bildungsplan aufgelisteten inhaltsbezogenen Kompetenzen abgeglichen. Markiert ist, zu welchen Themen in der Weiterbildung Aufgaben entwickelt wurden (blau hinterlegt, kursiv und unterstrichen). Bei der Kennzeichnung handelt es sich nur um eine Orientierungshilfe. Je nach Klassenstufe müssen Lerninhalte gegebenenfalls intensiver oder weniger intensiv unterrichtet werden.

1.1 Denk- und Arbeitsweisen in Naturwissenschaft und Technik: Systeme und Prozesse

1) Systeme analysieren und durch Systemgrenzen und Teilsysteme beschreiben (zum Beispiel Lebewesen, Maschinen, Sonnensystem)
2) Energie-, Stoff- und Informationsströme zwischen Teilsystemen erklären (zum Beispiel Treibhauseffekt, Stoffwechsel, GPS)
3) Wechselwirkungen (positive und negative Rückkopplung) zwischen Teilsystemen beschreiben (zum Beispiel Atemfrequenzanpassung, chemisches Gleichgewicht, Drehzahlregelung, Klimawandel)
4) *Veränderungen in Systemen als Prozesse beschreiben* (Prozessschritt, Teilprozess, *Eingabe-Verarbeitung-Ausgabe-Prinzip)*
5) Teilsysteme durch ihre äußeren Funktionen beschreiben (Black-Box-Denken; zum Beispiel Sinneszelle, Batterie)

1.2 Informationsaufnahme und –verarbeitung

Informationsaufnahme durch Sinne und Sensoren

1) *Die Verwendungsmöglichkeiten von Sensoren beschreiben* (zum Beispiel Blutdruckmessgerät, Hygrometer, Anemometer)
2) Bau und Funktionsweise eines Sinnesorgans mit einem entsprechenden technischen Sensor vergleichen (zum Beispiel Auge mit Digitalkamera, Ohr mit Mikrofon)
3) die Gefährdung von Auge oder Ohr durch Überlastung beschreiben und persönliches Handeln von gesundheitlichen Grenzwerten ableiten
4) die Gesetzmäßigkeit zwischen subjektivem Erleben und Intensität des physikalischen Reizes erläutern (zum Beispiel Lichtintensität, Lautstärke, Schwereempfinden)
5) die Erweiterung menschlicher Sinnesleistungen durch Sensoren erläutern (zum Beispiel IR-Sensor, Hörgerät, Wärmebildkamera, Barometer)

Informationsverarbeitung

1) Beispiele der analogen oder digitalen Informationscodierung aus Natur und Technik beschreiben (zum Beispiel digitale Dateiformate, maschinenlesbare Code-Systeme, DNA)
2) die *Funktionsweise gesteuerter oder geregelter Systeme analysieren* und dazu Energie-, Stoff- und *Informationsströme untersuchen* (zum Beispiel effiziente Energienutzung, Entwicklung eines Objekts mit Antrieb, Herstellung eines Produkts in einem chemisch-technischen Verfahren, physiologischer Regelkreis)
3) *das Prinzip der Steuerung darstellen und erklären* (zum Beispiel Robotik)
4) das Prinzip der Regelung auch unter Verwendung der Begriffe Sollwert, Istwert, Regelgröße und Störgröße darstellen und an Beispielen aus der Natur und der Technik erklären (zum Beispiel Körpertemperatur des Menschen, chemisches Gleichgewicht, Klimawandel: Mittlere Oberflächentemperatur der Erde, Oberflächentemperatur von Himmelskörpern)
5) *Elemente einer Programmiersprache beschreiben* (zum Beispiel Bedingung, Verzweigung, Schleife, Zähler, Zeitglied, Unterprogramm, Programmbausteine)
6) *Algorithmen für zeit- und sensorgesteuerte Prozesse in einer Programmiersprache darstellen und damit Steuerungsabläufe realisieren* (zum Beispiel Ampelsteuerung, Robotik)

7) *Algorithmen für zeit- und sensorgesteuerte Prozesse entwickeln, beschreiben und darstellen*
8) Chancen und Risiken der Informationstechnik für Individuum und Gesellschaft erläutern (zum Beispiel Simulation, Datenschutz, Internet of Things, Geoinformationssysteme, autonomes Fahren)

Elektronische Schaltungen

1) die *Funktion von Bauteilen elektrischer oder elektronischer Schaltungen* beschreiben (Schalter, Widerstand, Leuchtdiode, Transistor)
2) *Schaltungen entwickeln, Bauteile dimensionieren und auswählen* (Schaltplan, Datenblatt, Vorwiderstand, Spannungsteiler)
3) *elektrische oder elektronische Schaltpläne analysieren und in einfachen Fällen entwickeln*
4) *elektrische oder elektronische Schaltungen realisieren und ihre Funktionsfähigkeit untersuchen*

2. Zeichenerklärung

Ziel	
Übung	
Frage	
Buch	
Tipp / Hinweis Allgemein, führt nicht direkt zur Aufgabenlösung	
Fachwissen	 © Winter & Rieck
Kopfnussaufgabe: Zusatzaufgabe, die freiwillig gemacht werden kann	 © Winter & Rieck

3. Übersicht der Programmierbefehle

Tabelle 1: Programmierbefehle in der Weiterbildung

Befehl	**Bedeutung**
void setup () {…}	Programmbereich: wird einmal aufgerufen.
pinMode(Pin, Modus);	Pin gibt an, wo etwas angeschlossen ist und welchen Modus der Pin hat (Ein- oder Ausgang).
void loop () {…}	Programmbereich: wird in einer Schleife durchlaufen.
digitalWrite(Pin, Pegel);	Dem Pin kann ein Pegel zugeordnet werden: - HIGH: es liegt ein hohes Potential von 5 V an. - LOW: es liegt ein Potential von 0 V an.
digitalRead(Pin);	Einlesen eines digitalen Eingangs an einem Pin.
delay(ms);	Programmieren einer Verzögerungszeit, die in Millisekunden (ms) anzugeben ist.
delayMicroseconds(µs);	Programmieren einer Verzögerungszeit, die in Mikrosekunden (µs) anzugeben ist.
tone(Pin, Tonhöhe, Dauer);	Ausgabe von Tönen.
noTone(Pin);	Ton verstummt.
if(Bedingung) {…} else {…}	Ist die Bedingung in if(…) wahr, werden die Anweisungen in {…} ausgeführt. Ist die Bedingung falsch, werden die Anweisungen in else {…} ausgeführt.
Serial.begin(9600);	Die Datenübertragung wird initialisiert, wobei 9600 die Übertragungsrate darstellt.
analogRead(Pin);	Einlesen eines analogen Eingangs.
Serial.print(…);	Die Daten können mit diesem Befehl auf dem seriellen Monitor angezeigt werden.
Serial.println(…);	Die Daten können mit diesem Befehl auf dem seriellen Monitor angezeigt werden und nach jeder Ausgabe erfolgt ein Zeilenumbruch.
map(a,b,c,d,e);	Um Werte in andere Werte umwandeln zu können: map(Wert, von Tief, von Hoch, nach Tief, nach Hoch).

Merke:

Kommentare bzw. eigene Notizen werden hinter zwei Slash-Striche geschrieben: //….

Sollen Kommentare über mehrere Zeilen gehen, so werden diese wie folgt eingerahmt: /*…*/. Diese eigenen Notizen sind nicht Teil des Programmcodes und werden nicht kompiliert.

4. Der Mikrocontroller

In alltäglichen Geräten verstecken sich viele elektronische und logische Bauteile, welche einen ausgeprägten Blackbox-Charakter aufweisen. Dazu gehören Signal aufnehmende Bauteile - wie Sensoren - und ausführende Bauteile - wie Aktoren -. Zwischen diesen Bauteilen befindet sich eine Einheit, welche die ankommenden Signale verarbeitet, an die ausführenden Bauteile weitergibt und diese dadurch ansteuert. Ein Mikrocontroller stellt eine solche Einheit dar. Er arbeitet wie ein kleiner **Computer**, der Sensordaten **aufnehmen**, diese **verarbeiten** und über Aktoren **ausgeben** kann.

Durch Programmierung bestimmter Abfolgen und Befehle kann der Mikrocontroller Anweisungen umsetzen. Er befolgt somit Programmierbefehle. Hierbei stellt sich die Frage, was ein programmierbarer Gegenstand benötigt, um die Anweisungen umzusetzen? Ein Computer oder ein Laptop besitzt dazu eine Recheneinheit, einen Speicher, verschiedene Schnittstellen (zum Beispiel einen USB-Anschluss), einen Controller und verschiedene Wandler. All dies besitzt auch das Mikrocontroller-Board.

4.1 Mikrocontroller im Alltag

Wie schon erwähnt, zeigen sich Mikrocontroller oftmals versteckt in vielen technischen Alltagsgegenständen. Sie können dabei Mess-, Steuer- und Regelungsaufgaben übernehmen. Zu den Alltagsgegenständen zählen zum Beispiel MP3-Player oder ferngesteuerte Flugzeuge aus der Hobbyelektronik, als auch verschiedene Haushaltgeräte, wie zum Beispiel eine Kaffeemaschine. In vielen Haushalten werden heutzutage ganze Abläufe automatisiert, zum Beispiel zur optimalen Beheizung der eigenen vier Wände.

Der Mikrocontroller ist ein Halbleiterchip und wird in der Fachsprache auch als **µC** verkürzt angegeben. Der griechische Buchstabe µ (gesprochen: mü) steht für den Begriff „Mikro". In Abbildung 1 ist das Schaltzeichen des Mikrocontrollers ATmega 328, welcher auf dem Mikrocontroller-Board verbaut ist, abgebildet. Hierbei sind auch die verschiedenen Belegungen vermerkt. Das Board auf dem der Mikrocontroller verbaut ist, wird im folgenden Kapitel genauer betrachtet.

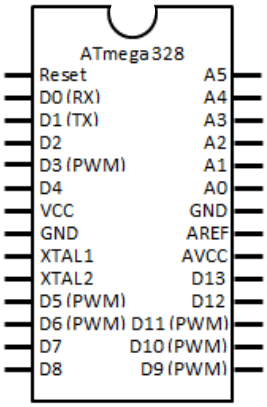


Abbildung 1: Schaltzeichen des Mikrocontroller ATmega 328

4.2 Das Mikrocontroller-Board - Die Hardware

Der Begriff Hardware umfasst in der Programmiersprache alles, was in die Hand genommen werden kann. An das Mikrocontroller-Board können verschiedene elektrische und elektronische Bauteile angeschlossen werden.

Das Mikrocontroller-Board Arduino wurde 2005 entwickelt und nach der Lieblingskneipe der beiden Erfinder benannt. Es stellt den Anfang der Mikrocontroller-Boards dar. Ein weiteres bekanntes Familienmitglied ist das Mikrocontroller-Board Arduino MEGA, welches auch häufig in Schulen verwendet wird und mehr Pins zur Verfügung stellt. Die Arduino-Familie ist groß. Dabei gibt es bestimmte Baureihen, welche nur für bestimmte Zwecke entwickelt worden sind (beispielsweise Arduino Lillypad, der für Textilien konzipiert wurde).

In dieser Weiterbildung wird das Mikrocontroller-Board funduino UNO R3 verwendet, das zu einer großen Familie der Mikrocontroller basierenden Platinen gehört. Dieses ist baugleich mit dem Arduino UNO R3.

Um einen Überblick über das Mikrocontroller-Board und seinen Aufbau zu bekommen, ist nachfolgend ein mit dem Programm *fritzing* erstelltes Board abgebildet.

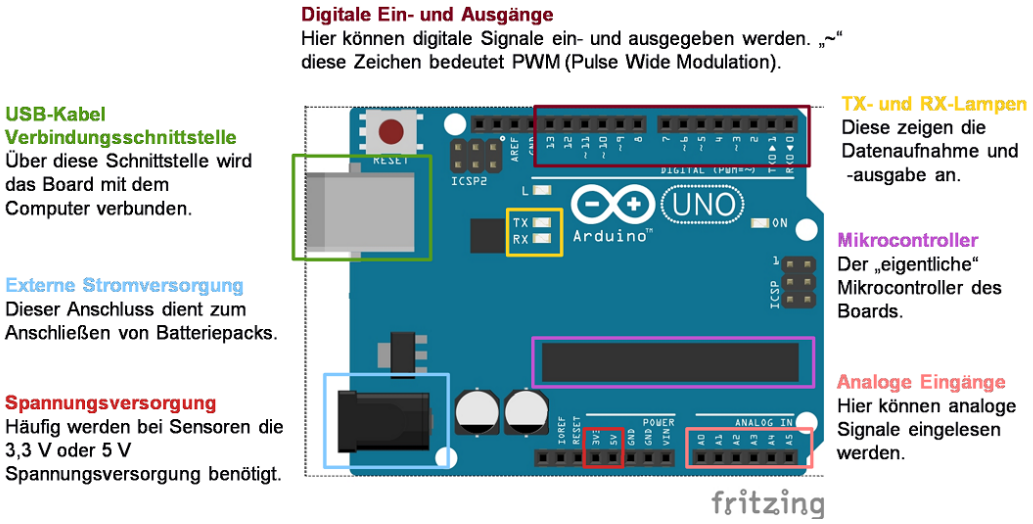


Abbildung 2: Das Mikrocontroller-Board, (in abgeänderter Form nach *fritzing*)

Für die Arbeit mit dem Mikrocontroller-Board sind einige technische Daten unverzichtbar. Gerade bei der Verwendung mit Sensoren müssen externe Spannungsquellen angepasst werden. Die technischen Daten werden in Tabelle 2 zusammengefasst.

Tabelle 2: Technische Daten des Mikrocontroller-Boards

Mikrocontroller	ATmega 328
Betriebsspannung	5 V
Eingangsspannungsbereich (empfohlen)	7-12 V
Digitale Ein- und Ausgänge	14
PWM-Kanäle	6
Analogeingänge	6
Max. Strombelastung pro Pin	40 mA
Max. Strombelastung des 3,3 V- Ausgangs	50 mA

4.3 Hardware Komponenten

Um das Mikrocontroller-Board in einen Schaltkreis zu integrieren, wird ein **Steckbrett** (**Breadboard**) benötigt. Das verwendete Breadboard stellt mehrere Buchsen zur Verfügung, in denen die Bauteile einfach hineingesteckt werden können. Dadurch lassen sich elektrische Schaltkreise aufbauen und verwirklichen.

Die Buchsen in der untenstehenden Abbildung bei Punkt 1 sind durch Leiterbahnen horizontal leitend miteinander verbunden. Die Buchsen in Punkt 2 sind durch Leiterbahnen vertikal leitend miteinander verbunden. Dies ist in Abbildung 3 an den blauen Markierungen zu erkennen.

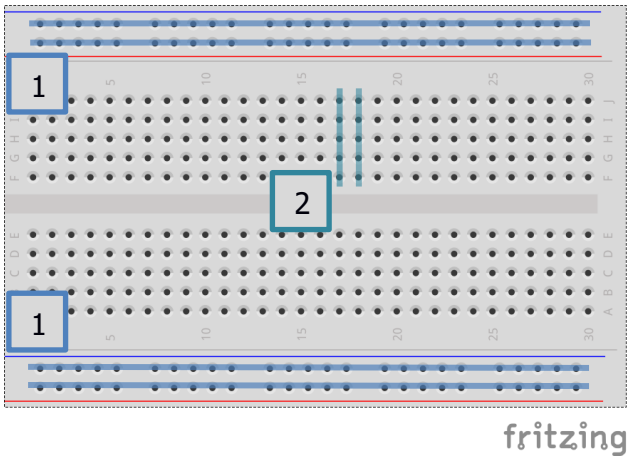


Abbildung 3: Breadboard (in abgeänderter Form nach *fritzing*)

Bei neuen Breadboards kann das Einstecken der elektrischen Bauteile einigen Kraftaufwand benötigen. Trotzdem ist **Vorsicht** geboten, denn die Anschlüsse können beim Herausnehmen und Einstecken der Bauteile und elektrischen Bauelemente verbiegen oder auch abbrechen.

Das Breadboard hat den Vorteil, dass kleinere Projekte immer wieder neu gestaltet werden können. Zur Umsetzung dauerhafter oder größerer Projekte eignen sich separate **Platinen**, die es in verschiedenen Ausführungen und Größen gibt. Diese können individuell gestaltet und dann einfach mit dem Mikrocontroller-Board verbunden werden. Für die Arbeit mit Platinen ist dabei ein genaues und gründliches Löten erforderlich.

Bei der Verwendung des Steckbretts (Breadboards) eigenen sich **Jumperkabel** (Verbindungskabel) (siehe Abbildung 4) zum Stecken leitender Verbindungen.

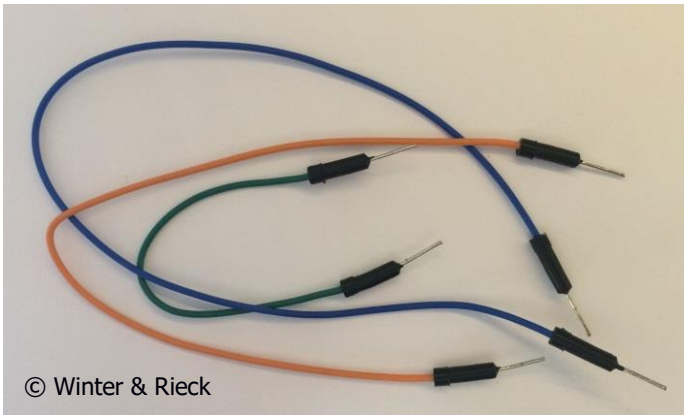


Abbildung 4: Jumperkabel (Verbindungskabel)

Auf unterschiedliche elektrische und elektronische Bauteile wird in den kommenden Kapiteln eingegangen.

5. Installation der Software

Im **Gegensatz zur Hardware** kann die Software nicht mit den Händen angefasst werden. Sie umfasst Programme und die darin geschriebenen Befehle. Die Programmierung einer Software bestimmt, wie ein Gerät - in dem Fall die Hardware – gesteuert werden soll und wie die dazugehörigen Befehle ausgeführt werden.

Bevor mit dem Programmieren mit der Arduino Software gestartet wird, erfolgt die Installation der Software. Um mit dem Mikrocontroller-Board zu arbeiten, wird die kostenlose Software auf der Internetseite

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

heruntergeladen. Es wird die zum Betriebssystem des Rechners passende Datei ausgewählt. Im Zuge des Downloads wird nach einer Spende an das Arduino-Team gefragt, der Download bindet jedoch nicht an die Bezahlung eines Geldbetrages. Durch das Auswählen des Buttons „JUST DOWNLOAD" kann die Software ohne Spende heruntergeladen werden. Ist der Download abgeschlossen und das Programm installiert, kann es auch schon fast losgehen.

Bevor mit dem Programm gearbeitet werden kann, müssen zwei Einstellungen durchgeführt werden. Bis jetzt sollte das Mikrocontroller-Board **noch nicht** mit dem Computer verbunden werden.

1. **Wählen des richtigen Boards**

   Nach Öffnen der Software wird zunächst das geeignete Board (hier: Arduino Uno) ausgewählt:

   Menüleiste → „Werkzeuge" oder „Tools" → „Platine" oder „Boards" → „Arduino UNO"

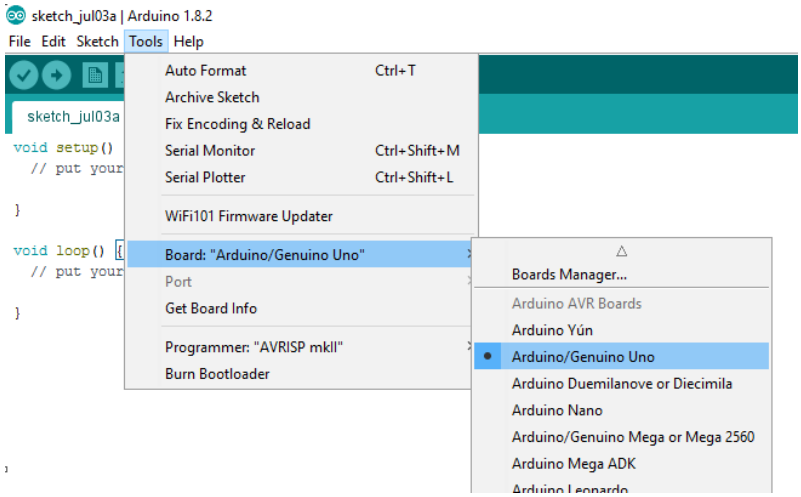


Abbildung 5: Inbetriebnahme, Auswahl des Boards

2. **Das Mikrocontroller-Board mit dem Computer verbinden**
   Das Mikrocontroller-Board wird über das USB-Kabel mit dem Computer verbunden. Dieser Schritt muss **vorsichtig** erfolgen, da es sonst zu Beschädigungen am Mikrocontroller-Board kommt. Anschließend wird in der Software der geeignete Port ausgewählt:

Menüleiste → „Werkzeuge" oder „Tools" → „Pin" → „COM 5"

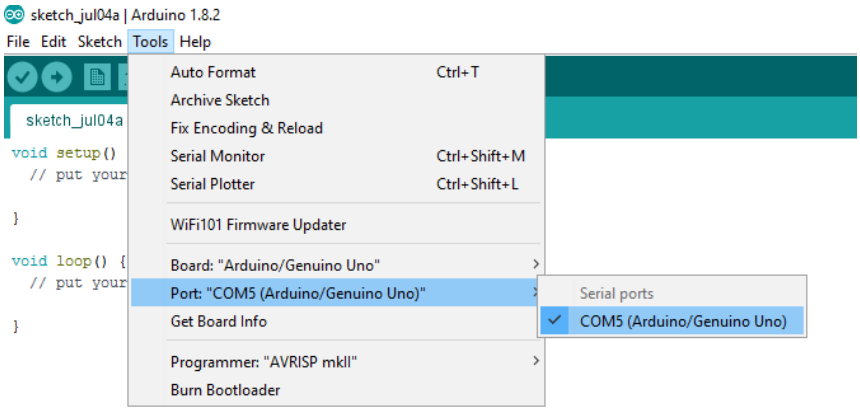


Abbildung 6: Mikrocontroller-Board mit dem Computer verbinden

Es gilt zu beachten, dass das Mikrocontroller-Board **immer an denselben USB-Port** am Computer oder Laptop angeschlossen werden sollte, um Fehlermeldungen zu vermeiden.

6. Das Programmieren – die Entwicklungsumgebung

Der Mikrocontroller auf dem Board arbeitet nach dem **EVA-Prinzip** (siehe Abbildung 7).

Abbildung 7: EVA-Prinzip

Bei der **Eingabe** kann es sich entweder um einen Sensor oder einen Taster handeln, welche mit der Außenwelt in Kontakt stehen. Diese sind unter anderem in der Lage verschiedene Größen zu messen, wie zum Beispiel die Umgebungstemperatur. Dabei liest der Mikrocontroller Signale in Form einer Spannung ein. Hinter der **Verarbeitung** steckt der „Controller" im Mikrocontroller, welcher mit Hilfe der geschriebenen Befehle bzw. Programmierung die Eingabe verarbeitet. Die verarbeiteten Eingangssignale werden über die **Ausgabe** als Spannungssignal an die Umwelt ausgegeben. Mit dieser Ausgabe lässt sich ein Aktor betätigen, zum Beispiel ein Motor oder ein Summer.

Die Verarbeitung erfolgt beim Mikrocontroller durch den programmierten Sketch. Bei der Arbeit mit dem Mikrocontroller-Board werden die Befehle in einer **Entwicklungsumgebung** programmiert. In einigen Büchern wird dies auch **Arduino-IDE** (IDE= Integrated Development Environment) genannt. In dieser Umgebung wird mit einer auf C++ basierenden Programmiersprache geschrieben. Dabei ist wichtig, dass der Mikrocontroller die programmierten Befehle **nacheinander** abarbeitet und diese nach Beendigung stets wiederholt. Wird das Mikrocontroller-Board über den USB-Anschluss an den Computer angeschlossen und der Sketch zum Mikrocontroller übertragen, so wird das geschriebene Programm in die Maschinensprache des Mikrocontrollers übersetzt. Dieser Vorgang wird **Kompilieren (= Übersetzen)** genannt.

Zur Steuerung des Mikrocontrollers werden die Befehle in der Software innerhalb einer Entwicklungsumgebung in eine Eingabemaske getippt. Die Programme werden in Arduino Sketche genannt. In Abbildung 8 ist diese Entwicklungsumgebung mit den jeweiligen Funktionen nachfolgend aufgeführt:

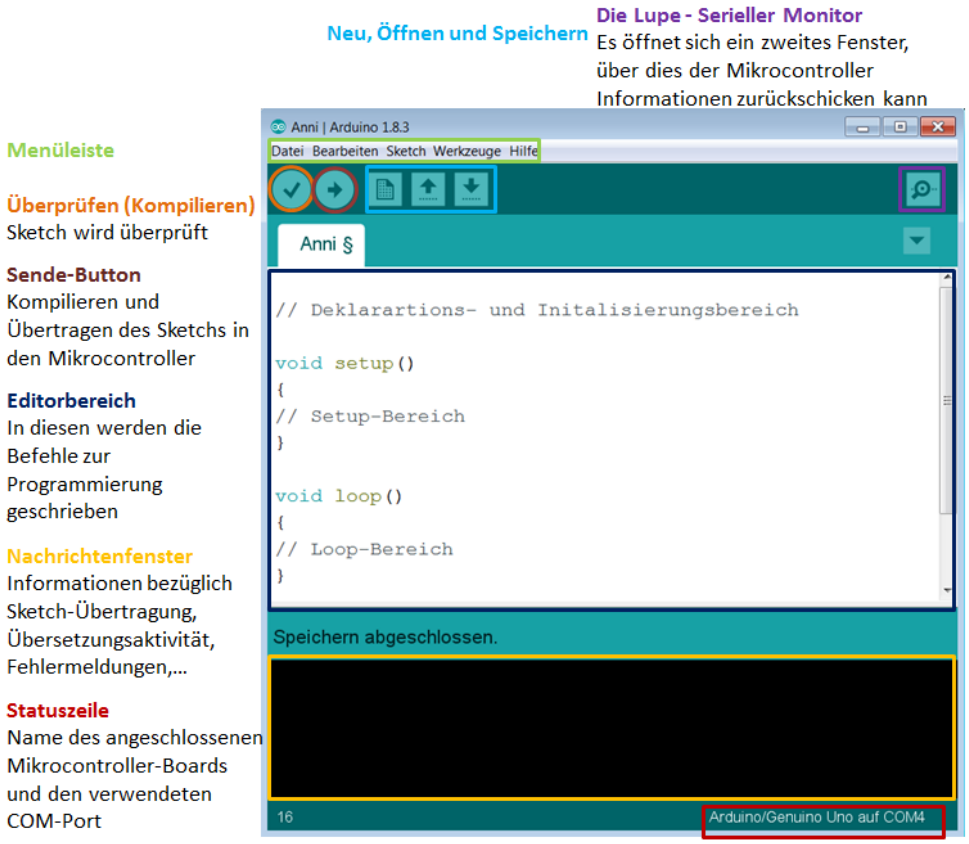


Abbildung 8: Entwicklungsumgebung der Arduino Software mit Erklärung

Die Entwicklungsumgebung wird in drei Bereiche unterteilt. In den Deklarations- und Initialisierungsbereich, den Setup Bereich und den Loop-Bereich. In den verschiedenen Bereichen werden unterschiedliche Programmteile geschrieben.

Der erste Block (wird nicht direkt angezeigt) ist der **Deklarations- und Initialisierungsbereich**. Dieser wird ganz am Anfang vor dem Setup-Bereich geschrieben. In diesen Bereich können externe Bibliotheken hinzugefügt oder es können Variablen deklariert und initialisiert werden.

Im **Setup-Bereich** werden die Grundinformationen festgelegt. Beispielsweise wird einem gewählten Pin eine Funktion zugeordnet. Der Pin kann entweder als Ausgang oder Eingang festgelegt werden. Diese Anweisung wird nur einmal durchlaufen und abgearbeitet.

Im **Loop-Bereich** (loop = Schleife) werden die Anweisungen für die einzelnen Ein- und Ausgänge geschrieben. Dieser Bereich wird dauerhaft wiederholt, er bildet daher eine endlose Schleife.

Die beiden geschweiften Klammern begrenzen den Setup- und den Loop-Bereich.
Das Wort **„void"** bedeutet so viel wie „Leerstelle" und wird hier vor dem Setup oder Loop geschrieben.

An den Stellen vor den Slash-Strichen

```
// Grundinformationen und // Anweisungen
```

steht dann der eigentliche Programmcode. Nach den beiden // sind kurze Kommentare eingefügt, um sich Notizen zu machen, das Programm übersichtlicher zu gestalten oder dem Programmcode eigene Informationen bzw. Anmerkungen zuzuteilen. Diese werden beim Kompilieren nicht beachtet und vom Mikrocontroller „überlesen".

Sollen über mehrere Zeilen Kommentare eingefügt werden, so werden diese zwischen den beiden Zeichen /* */ geschrieben.

```
/* Es können hier längere, mehrzeilige
Kommentare geschrieben werden */
```

7. Die Fehlermeldungen - und deren Behebung

Fehlermeldungen können das ein oder andere Mal zu Beginn auftreten. Jeder Befehl im Sketch wird vom Mikrocontroller ausgeführt, außer er ist falsch oder nicht sinnhaft geschrieben. Für fehlerfreies Programmieren sollte die „Sprache des Mikrocontrollers" gelernt werden und die programmierten Befehle sollten häufig geübt und stetig wiederholt werden.

Wird ein Befehl falsch oder nicht sinnhaft geschrieben, so zeigt die Arduino Entwicklungsumgebung eine Fehlermeldung im schwarzen Fenster an. Diese Fehler werden syntaktische Fehler genannt. Syntaktische Fehler beziehen sich meist darauf, dass die **Groß- und Kleinschreibung** nicht beachtet oder das **Semikolon ";"** vergessen wurde.

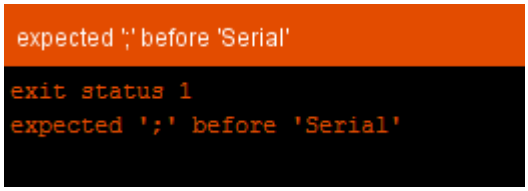


Abbildung 9: Fehlermeldung, Semikolon

In diesem Beispiel wurde vergessen, das Semikolon nach einem Befehl zu setzen.

Abbildung 11: Fehlermeldung, falscher USB-Port

In diesem Beispiel ist das Mikrocontroller-Board entweder nicht an den Computer angeschlossen oder dem falschen USB-Port zugeordnet.

Im Sketch selbst wird immer die Zeile unterhalb des Fehlers farblich markiert, der Fehler befindet sich also in der Zeile darüber.

8. Auf Fehlersuche

Wenn der Aufbau auf dem Breadboard vollständig ist, dieses mit dem Mikrocontroller-Board und dem Computer verbunden wurde und trotz der erfolgreichen Übertragung des Sketchs auf das Mikrocontroller-Board etwas nicht funktioniert, muss man sich auf Fehlersuche begeben.

Dazu ist es zunächst wichtig, das Mikrocontroller-Board vom Computer und somit von der Stromversorgung zu trennen. Diese Schritte können helfen, den Fehler schnell aufzufinden.

1. **Überprüfen des Aufbaus**

   Es sollte überprüft werden, ob alle Bauteile richtig eingesteckt sind. Gerade bei einer LED kann es dabei schnell zu Problemen kommen. So muss bei einer LED die Kathode - das ist der kürzere der beiden Anschlüsse - an den Ground (0 V) des Mikrocontrollers angeschlossen werden.

2. **Der richtige Widerstand**

   In der Eile wird dann doch danebengegriffen und es wird der falsche Widerstand in die Schaltung integriert. Es sollte überprüft werden, ob der richtige Widerstand korrekt in Reihe geschalten ist. Dabei kann die Farbcode-Tabelle (Tabelle 3, S. 39) der Widerstände weiterhelfen. Auch viele Sensoren benötigen unterschiedlich hohe Widerstände, die ebenso in Reihe geschalten werden müssen. Der richtige Aufbau des Breadboards kann mit Hilfe der Bilder aus dem Skript überprüft werden.

3. **Spannungsversorgung**

   Taster und Sensoren benötigen häufig eine externe Spannungsversorgung. In vorliegenden Fall stellt der *funduino* zwei Spannungsversorgungen zur Verfügung, zum einen 3,3 V und zum anderen 5 V. Hier kann es zum Vertauschen der beiden Anschlüsse GND und der zusätzlichen Spannungsversorgung kommen.

4. **Defekte Bauteile**

   Nach einiger Zeit können Bauteile durch unsachgemäße Handhabung im Experiment, zum Beispiel durch Überspannung, defekt gehen. Ob das Bauteil noch funktioniert, kann beispielsweise mit einem Multimeter durch Messung des ohmschen Widerstands oder durch eine Durchgangsmessung überprüft werden.

5. **Den Sketch überprüfen**

   Möglicherweise wurde ein Semikolon „;" vergessen, eine Klammer nicht geschlossen oder ein Befehl im Sketch nicht korrekt geschrieben. Diese Fehler werden angezeigt und können behoben werden. Manchmal kann aber auch einfach eine komplette Zeile fehlen, wie zum Beispiel ein wichtiger delay-Befehl.

   Es sollte überprüft werden, ob bei den Befehlen die richtigen Pins angesprochen werden. Werden die richtigen Befehle für die gewollte Funktion verwendet?

Mit ein wenig Übung werden die Fehler weniger und die Fehlersuche und -behebung gelingt präziser und schneller.

9. Wichtige Ausrüstung und Werkzeuge

Für erfolgreiche und professionelle Projekte werden oftmals noch weitere Hilfsmittel benötigt. Um sich selbst und die Bauteile zu schützen, sollte der Umgang mit diesen Hilfsmitteln vorher geübt werden.

9.1 Der Schraubendreher

Zum Entfernen und Befestigen von Bauteilen wird häufig ein Schraubendreher verwendet. Am geschicktesten sind kleine Schlitz-Schraubendreher. Wichtig ist darauf zu achten, dass an einer Schaltung nur dann gearbeitet wird, wenn **keine Spannung anliegt!**

9.2 Ein helfendes Händchen

Abbildung 12: Helfendes Händchen

Eine dritte Hand ist immer hilfreich. Wer schon einmal gelötet hat, weiß dass dieses Werkzeug immer gebraucht wird. Helfende Händchen (Abbildung 12) sind oft mit einer Lupe versehen und so für filigrane Arbeiten geeignet.

9.3 Der Lötkolben

Ein Lötkolben gehört bei vielen Arbeiten mit dem Mikrocontroller-Board zur Pflichtausstattung. Aufgrund des relativ geringen Platzverhältnisses muss **präzises und sauberes Löten** im Vorfeld geübt werden. Zudem können schlechte oder unsaubere Lötstellen entweder die angeschlossenen Sensoren und Bauteile oder den ganzen Mikrocontroller zerstören.

9.4 Lötzinn und Entlötlitze

Das Lötzinn benötigt man, um Bauteile oder Kabel leitend miteinander zu verbinden. Bei hohen Temperaturen wird dieses flüssig und beim Erkalten wieder fest. Bei der Auswahl des richtigen Lötzinns ist die DGUV Regel 113-018 zu beachten.

Wurde eine fehlerhafte Lötstelle erzeugt, kann das Lötzinn mit Hilfe einer Entlötlitze wieder entfernt werden.

Beim Löten sind zudem relevante Sicherheitsaspekte zu beachten. So heißt es in der DGUV Regel 113-018 der Deutschen Gesetzlichen Unfallversicherung unter I – 4.4 zu Schweißen und Löten: „Beim Schweißen oder Löten ist dafür zu sorgen, dass die Konzentration an gesundheitsgefährdenden Stoffen in der Atemluft minimiert wird." und „Bleihaltiges Lot darf nach der EG-Richtlinie 2002/95 (RoHS-Richtlinie) nicht verwendet werden."

Allgemeine Sicherheitshinweise für den NwT-Unterricht, sowie den Link zur DGUV Regel 113-018 findet man auf der Homepage des Landesbildungsservers:

https://www.ls-bw.de/,Lde/Startseite/Service/NwT

10. Hilfreiche Buch- und Internet-Tipps

Für ein umfangreiches Verständnis, bei der Findung von Projektideen oder bei Problemen bei der Umsetzung sind einige Internetseiten und Bücher sehr hilfreich. Diese werden hier nachfolgend aufgelistet.

10.1 Bücherauswahl

1. E. Bartmann, Die elektronische Welt mit Arduino entdecken.
2. T. Brühlmann, Arduino: Praxiseinstieg
3. P. und C. Caroli, Arduino Handbuch.
4. M. Margolis übersetzt von P. Klicman, Arduino Kochbuch.

(Eine ausführliche Aufführung der Bücher mit Jahr und Verlag ist im Literaturverzeichnis zu finden.)

Viele der Bücher enthalten nicht nur Erklärungen und physikalische Hintergrundinformationen, sondern auch kleinere Projekte mit vollständigem Aufbau auf dem Breadboard und den dazugehörigen Sketchen. Diese können geübt und in den eigenen NwT-Unterricht integriert werden.

10.2 Hilfe-Foren im Internet

Um bei Problemen hilfreiche Profi-Tipps zu bekommen, eignen sich Foren hervorragend. Das umfangreichste Forum ist das von Arduino - hier kann eine große und hilfsbereite Community angesprochen werden.

https://www.arduino.cc/

10.3 Zur Übung

Um neue elektrische Bauteile kennen zu lernen oder Erklärungen zur Funktionsweise eines Bauteils zu erhalten, gibt es viele Tutorials und Videos auf diversen Internetseiten. Diese erklären sehr ausführlich die verschiedenen Bauteile und deren Handhabung mit dem Mikrocontroller-Board.

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

www.funduino.de/vorwort

www.netzmafia.de/skripten/hardware/Arduino/programmierung.html

10.4 Bauteile kaufen

Verschiedene Internetseiten haben sich auf die Arbeit mit dem Mikrocontroller-Board spezialisiert. Dort können verschiedene Bausätze, Starter-Kits oder auch einzelne Bauteile bestellt werden. Diese Internetseiten stellen auch ausführliche Erklärungen zu den Bauteilen und deren Anwendungsmöglichkeiten zur Verfügung. In der Weiterbildung wird unter anderem mit den Bauteilen von *funduino* gearbeitet:

www.funduinoshop.com

11. *fritzing*

Wird mit dem Mikrocontroller-Board gearbeitet, ist das von der FH Potsdam initiierte Open-Source-Programm *fritzing* sehr nützlich. Während auf den realen Breadboards Schaltungen für Projekte nur temporär zur Projektumsetzung aufgebaut werden, unterstützt *fritzing* das Abspeichern der Schaltungen als digitale Version in einem virtuellen Layout.

fritzing kann einem bei der Unterrichtsvorbereitung und -durchführung dabei helfen, die geplanten Projekte auf virtuellen Breadboards zu stecken, diese Projekte anschließend abzuspeichern und bei Bedarf im Unterricht zu visualisieren.

Einige der in diesem Heft dargestellten Abbildungen sind mit Hilfe dieses Programms entstanden. *fritzing* ist sehr bedienerfreundlich und anschaulich und ermöglicht die Planung von Schaltungen, die Dokumentation eines Projektes oder die Entwicklung fertiger Schaltpläne.

Das Programm kann man kostenlos auf der Homepage von *fritzing* herunterladen:

http://fritzing.org/home/

Der Phantasie sind hier keine Grenzen gesetzt und man kann das Stecken der Breadboards üben, ohne dass dazu ein Mikrocontroller benötigt wird. In Abbildung 13 ist das Programm von *fritzing* dargestellt.

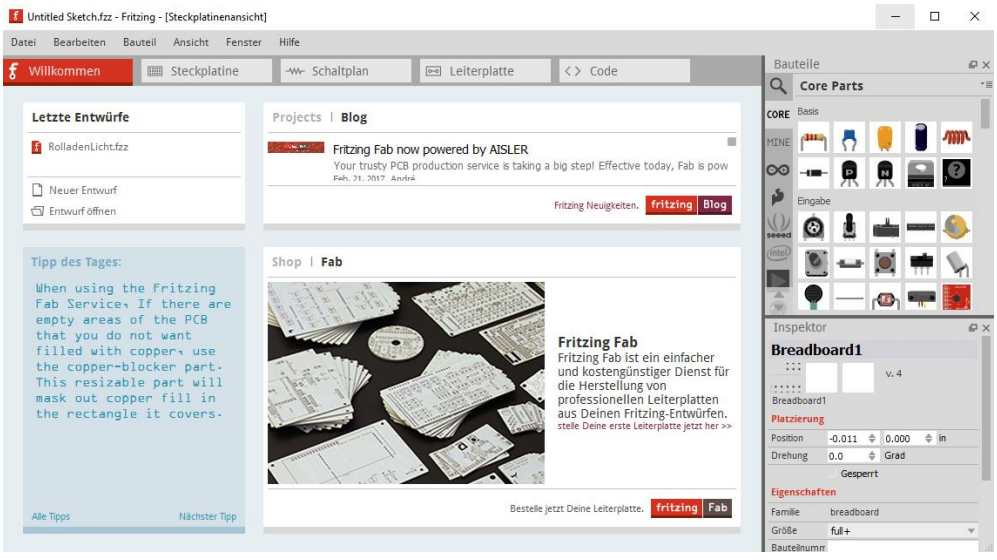


Abbildung 13: Bedienoberfläche des Programms *fritzing*

12. Die Leuchtdioden

Abbildung 14: Verschiedenfarbige LEDs

Leuchtdioden, auch bekannt als LEDs, werden im Alltag als Leuchtmittel z. B. zum Beleuchten von Wohnräumen genutzt. LEDs gibt es, wie in Abbildung 14 ersichtlich, in den unterschiedlichsten Farben. Je nach verwendetem Halbleitermaterial besitzen die LEDs eine spezifische Leuchtfarbe. Dadurch können Räume in verschiedenen Farbtönen akzentuiert werden.

LEDs werden auch in verschiedenen Alltagsgegenständen eingesetzt, wie zum Beispiel in einer Waschmaschine, um das eingestellte Waschprogramm mit Hilfe einer Leuchtanzeige anzuzeigen.

Damit eine LED, die in einem Stromkreis integriert wurde, leuchten kann und nicht zerstört wird, müssen die spezifischen Eigenschaften einer LED beachtet werden.

12.1 Der Aufbau und die Eigenschaften einer LED

Eine LED (**L**ight **E**mitting **D**iode) gehört zu den Halbleiterbauelementen. Diese hat die Eigenschaft, wie der Name schon vermuten lässt, als Diode Licht auszusenden. Der Halbleiterkristall befindet sich im „Kopf" der LED. Die beiden Anschlüsse sind mit dem Inneren der LED verbunden.

Halbleiter in der Chemie und Physik

Halbleiter haben die Eigenschaft, dass sie besser leiten wie Isolatoren, aber schlechter leiten wie Metalle.

Wird an einer Diode, wie der LED, eine äußere Spannung angelegt, wandern die Elektronen in einen energetisch günstigeren Zustand (Rekombination von Elektronen und Löchern).
Die dabei freiwerdende Energie wird in Form von Licht abgegeben, wobei die emittierte Wellenlänge des Lichts vom Material des Halbleiters abhängig ist.

Bei einer Diode muss darauf geachtet werden, wie man sie in den Stromkreis integriert, da diese nur in eine Richtung, die **Durchlassrichtung**, betrieben werden kann.

Am realen Bauteil gibt es einige Merkmale, die Auskunft über die richtige Integration innerhalb einer Schaltung geben.

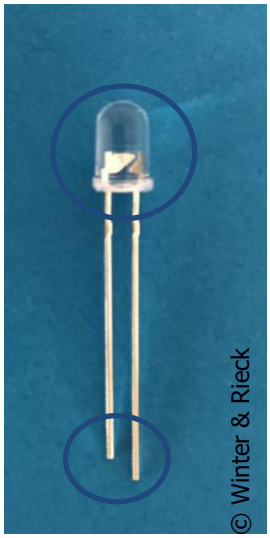


Abbildung 15: Merkmale einer LED

Werden z. B. die Anschlüsse einer LED (Abbildung 15) betrachtet, so sieht man, dass die beiden Anschlüsse unterschiedlich lang sind. Der längere Anschluss ist die Anode, der **k**ürzere die **K**athode der LED. Um eine LED in einen Stromkreis in Durchlassrichtung zu integrieren, muss die Kathode an das niedrigere Potential (Massepotential) und die Anode an das höhere Potential angeschlossen werden. Sollten die beiden Anschlüsse gleich lang sein, kann der „Kopf" der LED zu Rate gezogen werden. Erkennbar sind zwei unterschiedlich große „Metallplättchen". Der Anschluss an dem kleinen „Plättchen" entspricht der Anode und der Anschluss an dem größeren „Plättchen" der Kathode.

Das Schaltzeichen einer LED ist in Abbildung 16 dargestellt.

Abbildung 16: Schaltzeichen einer LED

Die LED wird in einem Schaltplan als Pfeil mit Querstrich dargestellt. Das heißt, dass sie nur in Pfeilrichtung stromdurchlässig ist. Die **Spitze** des Pfeils (Kathode) zeigt immer zum **niedrigeren Potential** (Massepotential) und die **andere Seite des Pfeils** (Anode) zum **höheren Potential**.

Potential in der Physik

Damit ein Stromfluss in einem Stromkreis entstehen kann, wird ein Potentialunterschied (Ladungsunterschied) benötigt.

Dabei existiert an einem Pol ein Ladungsüberschuss und an einem anderen Pol ein Ladungsmangel. Besteht eine elektrische Verbindung zwischen diesen, so fließt Strom.

Technische und physikalische Stromrichtung

Die technische Stromrichtung beruht auf einer früheren Festlegung: die Elektronen wandern vom höheren zum niedrigeren Potential.

Da der elektrische Strom jedoch eine Elektronenbewegung darstellt, fließt er vom negativeren (niedrigeren) Potential zum positiveren (höheren) Potential und wird als physikalische Stromrichtung bezeichnet. Die technische Stromrichtung wird in der Elektrotechnik vorzugsweise verwendet.

12.2 Der Schaltkreis mit einer LED

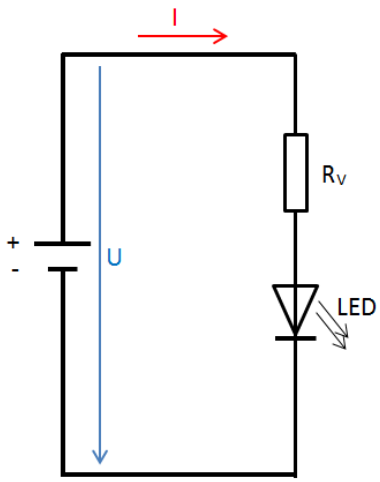


Abbildung 17: Schaltkreis mit einer LED

In Abbildung 17 ist eine **Reihenschaltung** dargestellt. Die Bauteile sind hier nacheinander in einem Stromkreis integriert.

Zu erkennen ist zum einen eine LED, welche an eine Spannungsquelle angeschlossen ist. Ein weiteres Schaltzeichen verdeutlicht einen Widerstand. Dieser wird als Vorwiderstand R_v bezeichnet.

In den Kapiteln 12.3 und Kapitel 12.4 erfolgt eine nähere Betrachtung von (Vor-)widerständen.

Reihen- und Parallelschaltung

In der Elektrotechnik werden zwei Arten von Schaltungen unterschieden.

Reihenschaltung:

In Abbildung 18 ist eine beispielhafte Reihenschaltung aus drei Widerständen abgebildet. Die Bauteile sind alle nacheinander in Reihe geschaltet. In einer Reihenschaltung fließt durch jedes Bauteil derselbe Strom.

Abbildung 18: Beispiel einer Reihenschaltung

Parallelschaltung:

Bei einer Parallelschaltung, wie in Abbildung 19, sind die drei Widerstände nebeneinander und daher parallel geschaltet. In einer Parallelschaltung herrscht an jedem Bauteil dieselbe angelegte äußere Spannung. Die Stromstärke ist dabei an jedem Bauteil unterschiedlich, wenn deren Widerstände unterschiedlich sind.

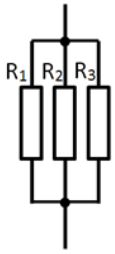


Abbildung 19: Beispiel einer Parallelschaltung

12.3 Die Berechnung des passenden Vorwiderstands

Es ist wichtig, dass eine LED nie ohne Vorwiderstand betrieben wird. Der Vorwiderstand hat die Aufgabe, die LED vor einem zu hohen Spannungsabfall zu schützen, sowie den Strom, welcher durch die LED fließt, zu begrenzen.
Das Schaltzeichen für einen Widerstand (Abbildung 20) ist nach europäischer Norm ein Rechteck. Bei *fritzing* wird dieser allerdings nach der amerikanischen Norm dargestellt.

Abbildung 20: Schaltzeichen (Widerstand nach europäischer Norm sowie nach amerikanischer Norm in abgeänderter Form nach *fritzing*)

Die Spannung, welche an einer LED abfällt, unterscheidet sich je nach Farbe der LED und kann den zugehörigen Datenblättern entnommen werden. In Abbildung 21 ist beispielhaft ein Auszug aus einem Datenblatt für eine rote LED abgebildet. Dabei zeigt die blaue Markierung auf, welche Kenngrößen für die Berechnung relevant sind. „VF" steht dabei für „Forward Voltage" und entspricht der Spannung U_F, welche an der Diode abfällt. Der typische Spannungswert ist mit U_F = 2,3 V angegeben, der Strom mit I_F = 20 mA.

PART SELECTION AND APPLICATION INFORMATION(RATINGS AT 25°C AMBIENT)

PART NO.	CHIP			LENS COLOR	Absolute Maximun Ratungs				Electrical-Optcal Characferis						Viewing Angle (deg)
									VF(V)			Rec if(mA)	Iv(mcd)		
	Material	PEAK WAVE Lenght λp(nm)	Emitting Color		Δ λ (nm)	pd (nW)	IF (mA)	Peak if(mA)	Mim.	Typ.	Max.	Max.	Mim.	Typ.	
513HD (5132D)	GaP	697	**Red**	Red Diffused	90	45	15	50	1.7	2.3	2.8	20	2.5	4.5	60

Quelle Datenblatt: http://funduino.de/DL/LEDrot.pdf, abgerufen am 20.10.2017

Abbildung 21: Auszug Datenblatt einer roten LED

Um den passenden Vorwiderstand für eine LED zu bestimmen, muss dieser berechnet werden. Am Beispiel wird dies für eine rote LED mit den Kennwerten aus Abbildung 21 durchgeführt. Die angelegte Spannung U_a am Stromkreis soll dabei 5 V betragen. Der Strom I_F, welcher durch die LED fließt, ist derselbe, wie der Strom, der durch den Widerstand fließt, da eine Reihenschaltung vorliegt.

Die Berechnung des Vorwiderstands R_V

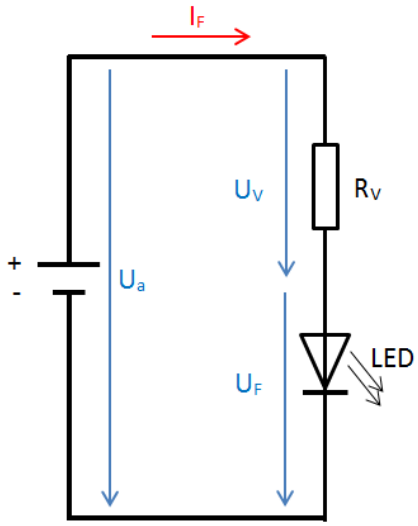


Abbildung 22: Dimensionierung am Schaltkreis der LED

Der Vorwiderstand R_V kann mit folgender Formel berechnet werden:

$$R_V = \frac{U_V}{I_F}$$

Der Strom I_F der Schaltung ist bekannt. Die Spannung U_V, welche am Vorwiderstand abfällt, berechnet sich mit Hilfe der äußeren Spannung U_a abzüglich der Diodenspannung U_F.

$$U_V = U_a - U_F = 5V - 2{,}3\ V = 2{,}7\ V$$

Mit Hilfe der obigen Gleichung für den Vorwiderstand kann dieser berechnet werden.

$$R_V = \frac{U_V}{I_F}$$

$$R_V = \frac{2{,}7\ V}{20\ mA} = \frac{2{,}7\ V}{0{,}020\ A}$$

$$R_V = 135\ \Omega.$$

Um die rote LED in die Schaltung zu integrieren, geht aus der Berechnung hervor, dass diese mit einem Vorwiderstand von R_V = 135 Ω in Reihe geschalten werden muss. Zur zusätzlichen Sicherheit des Mikrocontroller-Boards und der LED kann ein höherer Widerstand (in der Lehreinheit: R_V = 220 Ω) verwendet werden. Die LEDs leuchten dann etwas dunkler. Der Vorwiderstand kann entgegen seines Namens vor oder nach der LED platziert werden.

12.4 Die Farbcodes der Widerstände

Um die Größe des Widerstandswerts eines Widerstands kenntlich zu machen, wird ein **Farbcode** verwendet. Der Farbcode besteht in der Regel aus vier Ringen. Der Ring mit dem größeren Abstand zu den anderen Ringen muss immer rechts sein, dann kann der Farbcode links beginnend entschlüsselt werden.

Abbildung 23: Beispiel für den Farbcode eines Widerstands

Mit Hilfe von Tabelle 3 kann der Farbcode in die Größe des Widerstandes „übersetzt" werden.

Tabelle 3: Farbcodetabelle

	Ring 1 & 2	Ring 3
	0	-
Braun	1	0
Rot	2	00
Orange	3	000
Gelb	4	0000
Grün	5	00000
Blau	6	000000
Lila	7	0000000
Grau	8	00000000
Weiß	9	000000000

	Ring 4
Grau	0,05%
Lila	0,10%
Blau	0,25%
Grün	0,50%
Braun	1,00%
Rot	2,00%
Gold	5,00%
Silber	10,00%

Beispielhaft wird der Farbcode des Widerstands in Abbildung 23 entschlüsselt:

Die **ersten zwei Ringe** geben jeweils eine Zahl an. Ein jeder der beiden Ringe steht für eine Zahl gemäß dem Code. Die beiden roten Ringe stehen je für eine 2 und somit für 22.

Der **dritte Ring vor dem größeren Abstand** gibt an, wie viele „Nullen" hinzugefügt werden müssen. Durch den braunen Ring muss an das bisherige Ergebnis eine 0 angehängt werden und es ergibt sich der Wert 220.

Der **letzte Ring** gibt die Genauigkeit des Widerstands an. Der Widerstand im Beispiel ist auf 5 % genau gefertigt.

Der Widerstand im Beispiel hat somit einen Wert von 220 Ω ± 5 %.

13. Die digitalen Ein- und Ausgänge – Interaktion mit dem Board

Bei der Einführung in den Aufbau des Mikrocontroller-Boards sind die digitalen Ein- und Ausgänge (Pins) angesprochen worden. Im Folgenden sollen die Eigenschaften digitaler Signale betrachtet werden.

Digitale Signale

In der Digitaltechnik (lat. digitus = Finger) wird zwischen zwei festen Zuständen unterschieden und gearbeitet:

- **HIGH**- Pegel
- **LOW**- Pegel.

In der Digitaltechnik wird einem Zustand ein Spannungspegel zugeteilt. Der High-Pegel stellt das höhere Potential dar und beträgt +5 V. Der Low-Pegel ist der des niedrigeren Potentials, 0 V.
Ein digitaler Signalverlauf ist in Abbildung 24 zu erkennen.

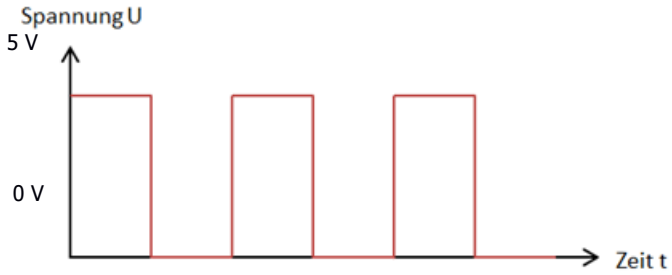


Abbildung 24: Rechtecksignal

Alle vorhandenen digitalen Pins des Mikrocontroller-Boards können bei der Programmierung zum einen als Eingang und zum anderen als Ausgang definiert werden. Die digitalen Ein- und Ausgänge 0 und 1 des Mikrocontroller-Boards sollten in der Regel nicht verwendet werden, da diese von der seriellen Schnittstelle des Mikrocontrollers selbst genutzt werden.

13.1 Festlegen und Programmieren der digitalen Ein- und Ausgänge

Prinzipiell ist zu beachten, dass die im nachfolgenden Abschnitt aufgeführten Befehle mit einem Semikolon abgeschlossen werden.

Im **Setup-**Bereich werden die einzelnen Pins des Mikrocontroller-Boards entweder als Eingang oder als Ausgang definiert. Dies erfolgt mit Hilfe des Befehls:

```
pinMode(Pin, Modus);
```

Der Modus gibt die Informationsrichtung an. Soll an diesem Pin ein Signal ausgeben werden, so wird **OUTPUT** geschrieben. Wenn ein Signal an diesem Pin eingelesen werden soll, muss an Stelle dessen **INPUT** geschrieben werden. Ein Beispiel für einen digitalen Input ist ein Taster.

Im **Loop-Bereich** wird über folgenden Befehl:

```
digitalWrite(Pin, Pegel);
```

festgelegt, welcher Pin welchen Pegel haben soll. Soll an einem Pin ein **HIGH**-Pegel von 5 V anliegen, kann zum Beispiel eine LED zum Leuchten gebracht werden. Soll die LED nicht leuchten, so wird stattdessen **LOW** geschrieben und es liegt das niedrige Potential von 0 V an.

Um ein Signal an einem digitalen Eingang einzulesen, zum Beispiel von einem Taster, wird folgender Befehl geschrieben:

```
digitalRead(Pin);
```

Die Befehle in der Programmierumgebung im Loop-Bereich werden vom Mikrocontroller nacheinander abgearbeitet und beim Erreichen des Endes wiederholt.

Soll ein Befehl für eine definierte Zeit ausgeführt werden, kann dies mit dem Befehl

```
delay(ms);
```

festgelegt werden.

Die Ausführung des Sketchs wird dabei für eine gewisse Zeitdauer unterbrochen. In die Klammer wird die Zeitdauer in Millisekunden angegeben. Wird zum Beispiel „5000" in die Klammer geschrieben, so entspricht das einer Dauer von 5 Sekunden.

Sollen zum Beispiel verschiedene **Befehle gleichzeitig und gleich lang** abgearbeitet werden, so werden diese nacheinander geschrieben und danach folgt der delay-Befehl.

```
digitalWrite(Pin1, HIGH);
digitalWrite(Pin2, LOW);
digitalWrite(Pin3, LOW);
delay(ms);
```

14. Die LED soll leuchten

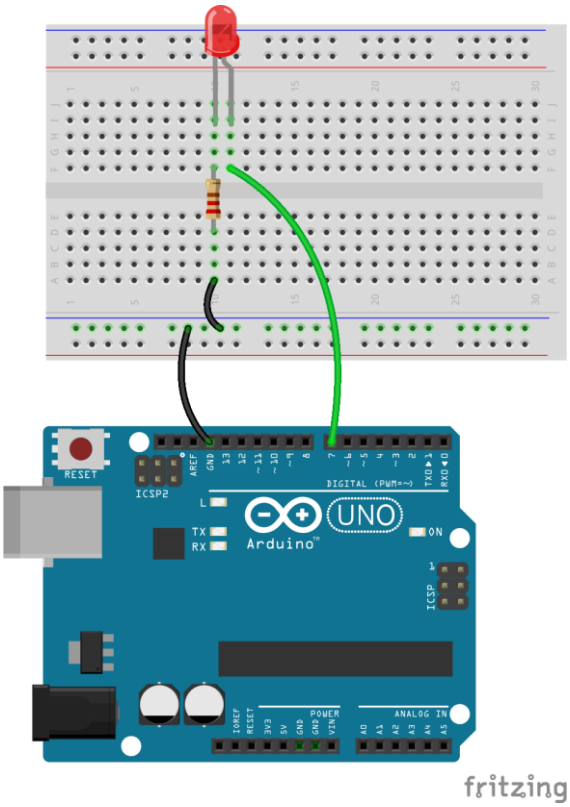


Abbildung 25: Schaltung mit einer LED auf dem Breadboard

Um eine LED mit Hilfe des Mikrocontroller-Boards anzusteuern und zum Leuchten zu bringen, kann der dargestellte Aufbau (Abbildung 25) auf das Breadboard übertragen und anschließend mit dem Mikrocontroller-Board verbunden werden.

Dabei wird die Anode der LED mit dem Pin verbunden und die Kathode der LED über den Vorwiderstand mit dem Ground (GND) des Mikrocontroller-Boards. Es wird so gewährleistet, dass der Strom durch alle Bauteile fließt.

Dies ist in Abbildung 26 in einem Schaltplan dargestellt.

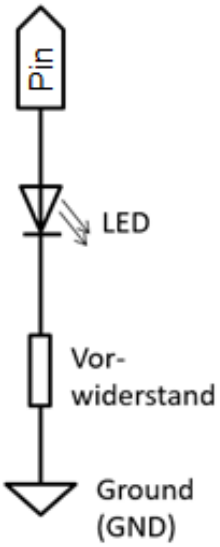


Abbildung 26: Schaltplan mit einer LED

Der Pin kann bei der Programmierung entweder auf den HIGH-Pegel (+5 V = HIGH) festgelegt werden, so dass die LED leuchtet. Oder der Pin wird auf den LOW-Pegel (0 V = LOW) festgelegt, so dass die LED nicht leuchtet.

14.1 Sketch

```
void setup()
{
  pinMode(7, OUTPUT);
}

void loop()
{
  digitalWrite(7, HIGH);
}
```

Im Sketch wird im **Setup-Bereich** über den Befehl:

pinMode(7, OUTPUT);

der Pin 7 als **OUTPUT** definiert. Der Pin fungiert daher als Ausgang.

Im **Loop-Bereich** wird an dem Pin, an dem die LED angeschlossen ist, über den Befehl:

digitalWrite(7, HIGH);

festgelegt, dass an Pin 7 der HIGH-Pegel anliegt, so dass die LED leuchtet.

15. Die If-else-Kontrollstruktur

Kontrollstrukturen werden zum Beispiel verwendet, wenn der Mikrocontroller auf äußere Einflüsse reagieren soll. Äußere Einflüsse können durch Sensoren (z. B. Regensensor) empfangen und an das Mikrocontroller-Board weitergeleitet werden. Ein INPUT kann aber auch zum Beispiel ein Taster sein, der die Zustände „gedrückt" oder „nicht gedrückt" einnehmen kann. Im Folgenden wird die If-else-Kontrollstruktur betrachtet, diese stellt daher eine „**Wenn-dann-sonst**"-Entscheidung dar.

Die If-else-Kontrollstruktur prüft, ob eine vorgegebene Bedingung **wahr oder falsch** ist. **Wenn** die Bedingung im if-Bereich wahr ist, **dann** werden die Anweisungen innerhalb des if-Bereichs ausgeführt. **Sonst**, also wenn die Bedingung falsch ist, werden die Anweisungen im else-Bereich ausgeführt.
Der Sketch besteht aus zwei Teilen im Loop-Bereich, zum einem wird der if-Bereich definiert und zum anderen der else-Bereich. Die Veranschaulichung der If-else-Kontrollstruktur erfolgt durch das Flussdiagramm in Abbildung 27.

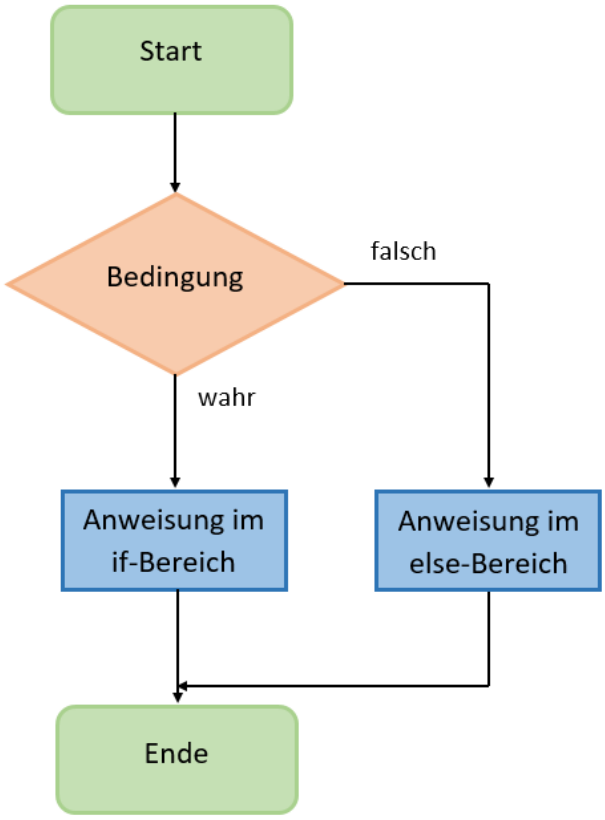


Abbildung 27: Flussdiagramm der If-else-Kontrollstruktur

15.1 Sketch

```
void setup()
{
pinMode(12, INPUT);
}

void loop()
{
if(digitalRead(12)== HIGH)
{
/* wenn der Taster gedrückt ist, soll eine Anweisung
 ausgeführt werden.*/
}
else
{
/* wenn der Taster nicht gedrückt ist, so soll eine
 andere Anweisung ausgeführt werden*/
}
}
```

Im obigen Sketch ist beispielsweise ein Taster an den digitalen Eingang (Pin 12) des Mikrocontroller-Boards angeschlossen. Dabei gilt es folgendes zu beachten: **wenn der Taster gedrückt ist**, der Schaltkreis dadurch geschlossen wurde und am Pin ein hohes Spannungssignal (**HIGH**) empfangen wird, soll eine bestimmte Anweisung ausgeführt werden. Dies wird zwischen die geschweiften Klammern des if-Bereichs geschrieben, wenn zum Beispiel eine LED leuchten soll.

```
if(digitalRead(12) == HIGH)
{
// Anweisung
}
```

Ist der Taster nicht gedrückt, liegt der niedrige Spannungspegel am Pin an. Die Bedingung im if-Bereich ist daher falsch und die Anweisung in der geschweiften Klammer des else-Bereichs wird ausgeführt:

```
else
{
// andere Anweisung
}
```

Dann leuchtet zum Beispiel die LED nicht auf.
Das doppelte Gleichheitszeichen in der If-else-Kontrollstruktur ist ein Vergleichsoperator und bedeutet „Gleichheit".

Vergleichsoperatoren

Bei der Programmierung haben sie die Aufgabe logische Verknüpfungen zu erstellen.

Weitere Vergleichsoperatoren und deren Bedeutung sind nachfolgend dargestellt:

==	Gleichheit
!=	Ungleichheit
<	kleiner als
>	größer als
<=	kleiner gleich
>=	größer gleich

16. Der Taster – Leuchten auf Knopfdruck

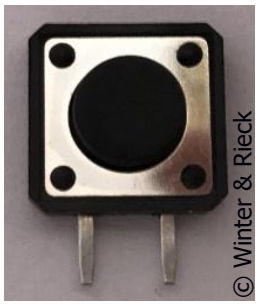


Abbildung 28: Taster

Im alltäglichen Gebrauch sind Taster zum Beispiel bei Taschenlampen, Kaffeemaschinen oder Türklingeln zu finden. Durch Betätigung eines Tasters können LEDs ein- und ausgeschalten werden. In Abbildung 28 ist ein Taster abgebildet.

16.1 Der Aufbau und die Funktion eines Tasters

Ein Taster kann zwei Zustände annehmen. Wird er gedrückt, ist der Stromkreis geschlossen und es fließt ein Strom durch diesen. Wird der Taster wieder losgelassen, geht er in seinen Grundzustand zurück. Der Stromkreis ist offen und es fließt kein Strom.
In Abbildung 29 ist der Taster als Schaltzeichen in einem Schaltplan dargestellt.

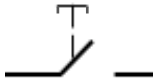


Abbildung 29: Schaltzeichen eines Tasters

16.2 Der Schaltkreis mit einem Taster

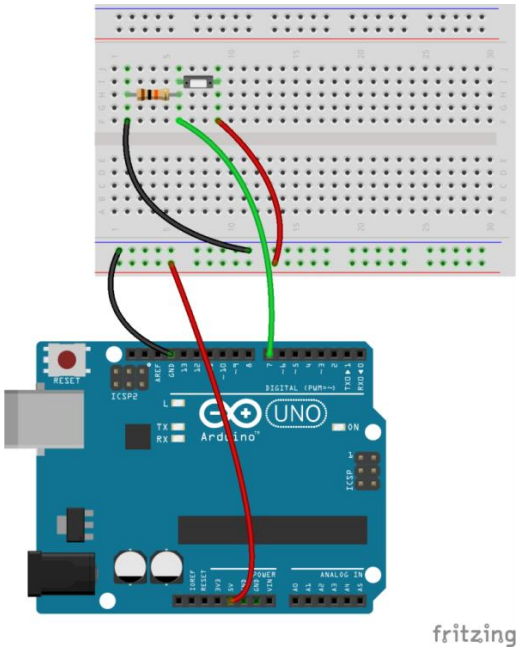


Abbildung 30: Taster auf dem Breadboard

Wie in Abbildung 30 ersichtlich, wird der Taster auf dem Breadboard mit einem Widerstand in Reihe geschaltet. Die Funktion des Widerstands in der Schaltung wird im Folgenden dargestellt.
Der Mikrocontroller kann an einem digitalen Eingang entweder den definierten Spannungspegel HIGH oder LOW einlesen. Das bedeutet, um auf einen Tasterdruck zu reagieren, benötigt der Pin einen eindeutigen Pegel. Liegt an einem Eingang kein definierter Pegel vor, so kann durch äußere Einflüsse, wie zum Beispiel durch ein stromdurchflossenes Kabel oder auch durch die Luftfeuchtigkeit, das Verhalten am Pin beeinflusst werden.

16.3 Der Pulldown-Widerstand

Um einen Taster mit dem Mikrocontroller-Board zu verbinden, so dass ein eindeutiger Spannungspegel am Pin anliegt, werden zwei Möglichkeiten unterschieden:

1. Soll am Eingangsspin des Mikrocontroller-Boards ein hohes Spannungssignal (HIGH-Pegel) anliegen, wenn der Taster gedrückt wird, so wird von einem **Pulldown-Widerstand** gesprochen.
2. Soll bei Tastendruck hingegen ein niedriges Spannungssignal eingelesen werden, so nennt sich dies **Pullup-Widerstand**.

Im Folgenden wird der **Pulldown-Widerstand** betrachtet. Wie der Name sagt, steht **„pull" für ziehen** und **„down" für runter**. In Abbildung 31 ist ein Pulldown-Widerstand mit einem Taster in einem Schaltplan dargestellt.

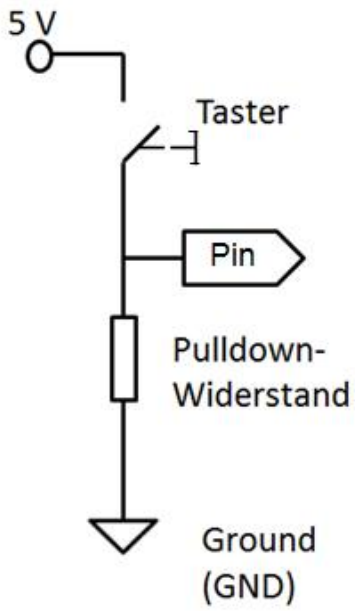


Abbildung 31: Schaltplan mit einem Pulldown-Widerstand und einem Taster

Der Pulldown-Widerstand hat die Aufgabe, das Potential am Pin „nach unten zu ziehen", so dass Ströme, die von außen einwirken, am Widerstand abfallen. Das bedeutet, ist der Taster offen, so fließt definitiv kein Strom durch den Stromkreis und an dem Pin, an dem der Taster angeschlossen ist, liegt ein Spannungspegel von 0 V (LOW-Pegel) an. Wird der Taster gedrückt, so wird der Pin direkt mit einer Betriebsspannung von +5 V (HIGH-Pegel) verbunden.

Für die Anwendung ist ein **10 kΩ Widerstand** (Erfahrungswert, häufige Verwendung in der Literatur) ausreichend, dieser wird mit dem Taster in Reihe geschaltet.

16.4 Das Anschließen eines Tasters

Es gilt zu beachten, dass der Taster und der Widerstand immer so angeschlossen werden, wie es in Abbildung 30 nach *fritzing* dargestellt ist (vgl. Kapitel 16.2).
Der Taster ist zum einen über den Pulldown-Widerstand mit Ground (GND) verbunden und zum anderen mit der Betriebsspannung von +5 V des Mikrocontroller-Boards.

16.5 Sketch

```
void setup()
{
pinMode(12, INPUT);
}

void loop()
{
if(digitalRead(12)== HIGH)
{
/* wenn der Taster gedrückt ist, soll eine Anweisung
 ausgeführt werden.*/
}
else
{
/* wenn der Taster nicht gedrückt ist, so soll eine
 andere Anweisung ausgeführt werden*/
}
}
```

Im **Setup-Bereich** wird Pin 12, an dem der Taster angeschlossen ist, der Modus **INPUT** zugewiesen. Das bedeutet dieser Pin wird als Eingang definiert.

```
pinMode(12, INPUT);
```

An diesem Pin wird der Mikrocontroller ein Signal einlesen. Der Taster hat in diesem Fall die Eigenschaft eines Sensors, welcher die Zustände gedrückt oder nicht gedrückt einnehmen kann.

Im **Loop-Bereich** wird die If-else-Kontrollstruktur geschrieben.
Im if-Bereich wird der Pegel des Pins, an dem der Taster angeschlossen ist, eingelesen und definiert, ob er den vorgegebenen **HIGH**-Pegel besitzt:

```
if(digitalRead(12) == HIGH)
      {
  // Anweisung
      }
```

Ist der Taster nicht gedrückt und die Bedingung im if-Bereich daher falsch, wird die Anweisung im else-Bereich durchlaufen.

```
else
{
  // andere Anweisung
 }
```

Wichtig ist dabei, dass die Bedingung bei einer Schaltung mit einem Pulldown-Widerstand mit „HIGH" festgelegt wird.

17. Der Piezospeaker – Der Mikrocontroller macht Geräusche

Durch das Ansteuern von kleinen Piezolautsprechern kann mit Hilfe der geeigneten Programmierung ein Geräuschsignal am Mikrocontroller-Board ausgegeben werden. In Abbildung 32 ist ein auch als Piezospeaker bezeichneter Piezolautsprecher dargestellt.

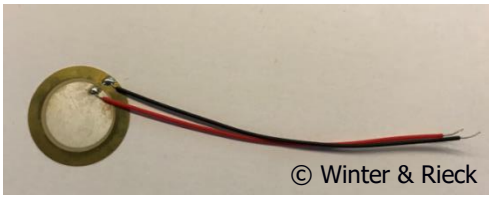


Abbildung 32: Piezospeaker

Piezospeaker können zum Beispiel dazu verwendet werden, um ein Warnsignal auszugeben oder eine Alarmanlage mit Hilfe eines Mikrocontrollers zu realisieren.

17.1 Der Aufbau und die Funktion eines Piezospeakers

Der Piezospeaker bzw. das Piezoelement besteht unter anderem aus einem Kristall. Wird an den Piezospeaker eine äußere Spannung angelegt, verformt sich der Kristall im Inneren. Durch die Verformung des Kristalls wird die Luft um ihn herum zusammengedrückt. Es entsteht eine Druckwelle, die sich um den Lautsprecher im Raum ausbreitet.

Das Schaltzeichen eines Piezoelements ist in Abbildung 33 dargestellt.

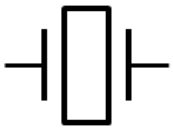


Abbildung 33: Schaltzeichen eines Piezoelements

Wahrnehmung von Tönen in der Biologie

Das Ohr kann Frequenzen, je nach Alter, zwischen 16 Hz und 20 kHz wahrnehmen.

Die Schwingungen werden über die Gehörknöchelchen im Mittelohr an die Hörschnecke im Innenohr weitergeleitet. Dort werden Sinneszellen (=Haarzellen) mechanisch stimuliert und die Information wird in ein neurologisches Signal übersetzt. Die gebildeten Nervenimpulse werden über die Synapsen an die Nervenzellen im Gehirn übermittelt und es wird ein Sinneseindruck wahrgenommen.

Der Zusammenhang zwischen Tonhöhe und Frequenz kann aus der Tastatur in Abbildung 34 entnommen werden.

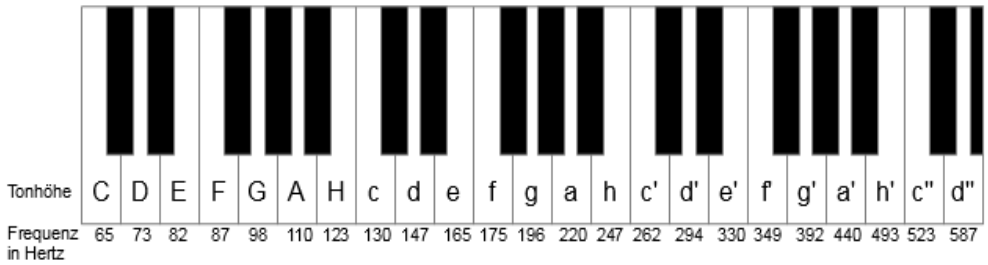


Abbildung 34: Zusammenhang Tonhöhe und Frequenz

17.2 Der Schaltkreis mit einem Piezospeaker

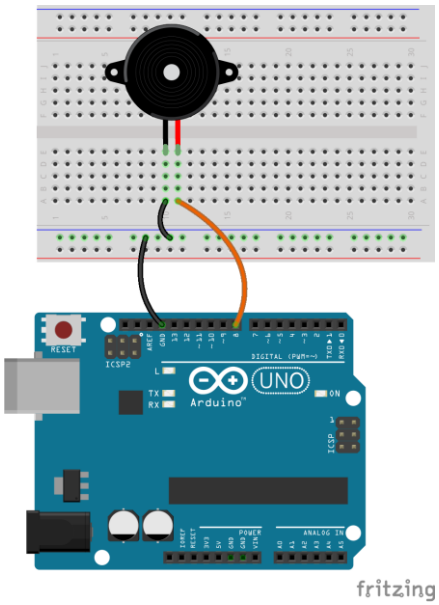


Abbildung 35: Piezospeaker auf dem Breadboard.

Der Piezospeaker hat zwei Anschlüsse. In Abbildung 35 wird der rote Anschluss (+) an den Pin des Mikrocontroller-Boards verbunden. Der schwarze Anschluss (-) wird mit dem Ground (GND) verbunden.

Mit dem Piezospeaker können kleinere Melodien und auch Signaltöne ausgegeben werden. Zur Verstärkung des Tongeräuschs kann der Piezospeaker auf eine Unterlage befestigt werden, welche frei schwingen kann.

Der Schaltplan, mit Piezospeaker (Piezoelement) und Verbindung zum Mikrocontroller-Board am Ground, ist in Abbildung 36 dargestellt.

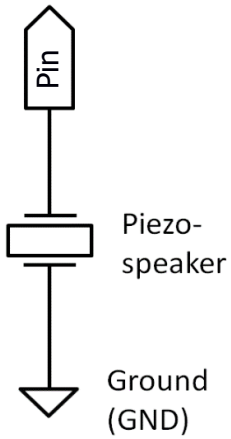


Abbildung 36: Schaltplan eines Piezoelements

17.3 Sketch

Der Piezospeaker kann über zwei unterschiedliche Programmierbefehle ein Ton- oder Geräuschsignal ausgeben.

1. Mit dem Befehl „digitalWrite(…);" und „delayMicroseconds(…);":

```
void setup()
{
  pinMode(8, OUTPUT);
}

void loop()
{
digitalWrite(8, HIGH);
delayMicroseconds(500);
digitalWrite(8,LOW);
delayMicroseconds(500);
}
```

Im **Setup-Bereich** wird der Pin, an dem der Piezospeaker angeschlossen ist, als **OUTPUT** festgelegt.

Die Tonausgabe mit dem Piezospeaker kann im **Loop-Bereich** über folgenden bekannten Befehl erfolgen:

digitalWrite(8, Pegel);

Der Befehl:

delayMicroseconds(500);

ist vergleichbar wie der Befehl „delay(ms);", wobei die Zeitdauer in Mikrosekunden angegeben wird.

Die Umrechnung: 1 ms entspricht 1000 µs.

2. Mit dem Befehl „tone(…);" und „noTone(…);"

```
void setup()
{
  pinMode(8, OUTPUT);
}

void loop()
{
  tone(8, 440, 200);
  delay(500);
  noTone(8);
  delay(500);
}
```

Im **Setup-Bereich** wird der Pin ebenso als **OUTPUT** festgelegt.

Im **Loop-Bereich** wird über den Befehl:

tone(Pin, Tonhöhe, Dauer);

festgelegt, an welchen Pin (hier Pin 8) der Piezospeaker angeschlossen ist, welche Frequenz der ausgegebene Ton haben soll und die Zeitdauer, wie lange das Tonsignal ertönen soll. Mit dem Befehl:

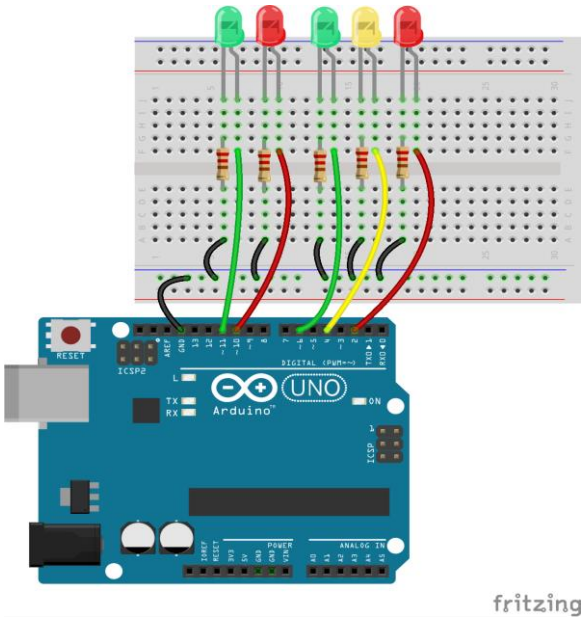
noTone(8);

verstummt der Piezospeaker.

18. Festlegung von Variablen

Zu jedem Pin gehört in späteren Projekten ein Bauteil. Werden zum Beispiel mehrere LEDs an verschiedene Pins angeschlossen, verliert man schnell den Überblick beim Schreiben eines Sketchs. Abhilfe schafft dabei das Festlegen von bestimmten Variablen. Um einen übersichtlicheren Sketch zu generieren, können den Pins Namen vergeben werden. Die Namen sollten am besten das angeschlossene Bauteil oder deren Funktion beschreiben.
Das Verteilen von Namen an den Pins nennt man **Deklaration**.

Als Beispiel eine kleine Ampelschaltung:




```
int gruen1 = 11;
int rot1 = 10;

int gruen2 = 6;
int gelb2 = 4;
int rot2 = 2;

void setup() {

pinMode(gruen1, OUTPUT);
pinMode(rot1, OUTPUT);

pinMode(gruen2, OUTPUT);
```

Abbildung 37: Kleine Ampelschaltung auf dem Breadboard mit Auszug eines Sketchs

Beim Schreiben eines Sketchs wird diese Namensverteilung, wie in Kapitel 6 schon erwähnt, **vor dem Setup-Bereich** im Deklarations- und Initialisierungsbereich vorgenommen. Beim Programmieren wird dann anstelle der Pin-Zahl nur der Name verwendet.

Über den Befehl **int** wird im obigen abgebildeten Sketch z.B. der Variablenname **„gruen1"** (die Umlaute ä, ö, ü werden beim Programmieren immer als ae, oe, ue geschrieben) eingeführt und mit **Pin 11** festgelegt. Taucht nun beim Abarbeiten des Sketchs der Befehl „gruen1" auf, so wird immer Pin 11 angesprochen.

Variablen in der Computersprache

In der Computersprache muss eine Variable immer einem festen Typ entsprechen.

Es muss unterschieden werden zwischen ganzen positiven Zahlen, ganzen negativen Zahlen und Gleitkommazahlen. Außerdem können die Zahlen nur einen bestimmten Bereich annehmen, denn je größer die Zahlen, desto mehr Speicherplatz wird benötigt. Mit „int" kann die Variable ganze Zahlen im Bereich von -32768 bis +32767 annehmen. Dies entspricht einem Speicherplatz von 2 Bytes.

Es existieren noch andere Variablentypen, wie zum Beispiel byte, long, etc..

19. Die analogen Eingänge – Der Mikrocontroller kommuniziert

Bisher wurden nur die digitalen Ein- und Ausgänge betrachtet. Um die analogen Eingänge zu verstehen und diese von den digitalen zu unterscheiden, werden nachfolgend deren Eigenschaften genauer betrachten.

Analoge Signale

Wie schon vorab besprochen, wird zwischen den zwei digitalen Zuständen

- HIGH- Pegel und
- LOW- Pegel

unterschieden.

Im Vergleich dazu weisen analoge Signale eine andere Form auf. Diese können über den zeitlichen Verlauf jeden beliebigen Wert zwischen den Pegeln HIGH und LOW annehmen. Sie können zwischen den zwei Zuständen stufenlos hin- und herpendeln. Ein Beispiel für eine Sonderform eines analogen Signals ist ein sinusförmiger Signalverlauf, der an einem Oszilloskop betrachtet werden kann (siehe Abbildung 38). Jedoch sind auch völlig unregelmäßige Signalverläufe denkbar, die sich beispielsweise unter äußeren Einflüssen verändern.

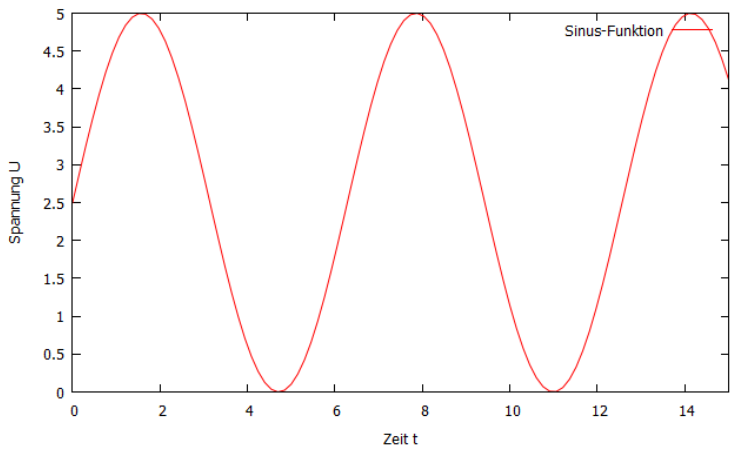


Abbildung 38: Sinus-Signal

Die analogen Eingänge des Mikrocontrollers werden unter anderem dazu verwendet, um Sensoren auszulesen. Wird zum Beispiel ein Photowiderstand als Sensor verwendet, welcher je nach Helligkeit seinen Widerstand verändert, gibt dieser Sensor eine unterschiedlich hohe Spannung am Pin aus. Dieser Wert wird vom Controller umgerechnet und auf eine reale Helligkeit bezogen.

Vorsicht! Wenn mit externen Spannungsquellen gearbeitet wird, muss darauf geachtet werden, für welche Spannungen die angeschlossenen Sensoren ausgelegt sind. Informationen dazu findet man in den Datenblättern der Sensoren (erhältlich auf der *funduino*-Homepage). Die Datenblätter enthalten die wichtigsten Eckdaten zum jeweiligen Sensor.

Wichtig! Da der Mikrocontroller nur mit einem 10-Bit-System arbeitet, ist die Auflösung der gemessenen Signale nicht unendlich klein. Es stehen 1024 Zahlen zur Verfügung, die einem bestimmten Spannungswert zwischen 0 V und +5 V zugeordnet werden können. Damit ergibt sich eine Messgenauigkeit M von:

$$M = \text{Referenzspannung}/\text{Auflösung}$$

$$M = 5\ V/\ 1024 = 4{,}883\ mV \approx 5\ mV$$

Die Zuordnung findet über einen A/D-Wandler statt. Er wandelt das analoge Signal in ein digitales um, bzw. das Eingangssignal in eine Bitkombination. Das digitale Signal ist somit stufenförmig in 4,833 mV-Schritte unterteilt. Das bedeutet, dass ein externes analoges Signal seinen Wert um mindestens 4,833 mV verändern muss, um im Arduino als Signalveränderung erfasst zu werden.

Manche Sensoren arbeiten aber auch nur mit einer geringeren Referenzspannung, dadurch verändert sich die Messgenauigkeit!

19.1 Das Signal eines Sensors einlesen

Ist ein Sensor an einem analogen Pin angeschlossen, ist es sinnvoll, die gemessenen Werte des Sensors über den seriellen Monitor oder den seriellen Plotter anzeigen zu lassen. Diese findet man in der Menüleiste der Arduino IDE unter „Werkzeuge" bzw. „Tools". Dadurch bekommt man ein Gefühl für den Wertebereich des Sensors.

Wird weiterhin das Beispiel Helligkeitssensor betrachtet, ist es wichtig zu wissen, ab welchem Sensorsignal ein Aktor reagieren soll. Soll sich ein Rollladen ab einem bestimmten Helligkeitswert schließen, muss der entsprechende Wert des Helligkeitssensors ermittelt werden. Dies ist mit dem seriellen Monitor möglich.

Um einen Sensor mit dem Mikrocontroller einlesen zu können, sind verschiedene Befehle nötig. An einem Beispiel wird dies genauer betrachtet.

```
int Sensor=A0;
int wert;

void setup()
{
Serial.begin(9600);
}

void loop()
{
wert=analogRead(Sensor);

Serial.print("Der Messwert lautet=");
Serial.println(wert);
}
```

Der erste neue Befehl befindet sich im **Setup-Bereich** und wird einmal abgearbeitet. Um die Daten des Sensors später anzeigen zu lassen, muss im Setup-Bereich die Datenübertragungsrate festgelegt werden. Das geschieht mit dem Befehl:

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

Die Zahl 9600 beschreibt die Datenübertragungsrate in der Einheit **„baud"**. Die Einheit baud bedeutet Symbole pro Sekunde.

Im **Loop-Bereich** wird über den Befehl

```
analogRead(Sensor);
```

der Messwert des Sensors eingelesen. In diesem Fall wird die Ausgabe unter der Variablen „wert" gespeichert. Das vereinfacht die Programmierung für verschiedene spätere Projekte.

19.2 Werte anzeigen – Der serielle Monitor

Im Loop-Bereich sind noch weitere Befehle zu erkennen. Diese werden benötigt, um die Werte im **seriellen Monitor** anzeigen zu lassen.

Abbildung 39: Symbol, serieller Monitor.

Der serielle Monitor kann über das Symbol der **kleinen Lupe** in der Arduino-IDE geöffnet werden. Hierbei öffnet sich ein weiteres Fenster, worin die Werte angezeigt werden, die am Sensor eingelesen wurden. Ebenfalls kann der Sensor abgerufen werden, wenn in der Menüleiste auf „Tools" geklickt wird.

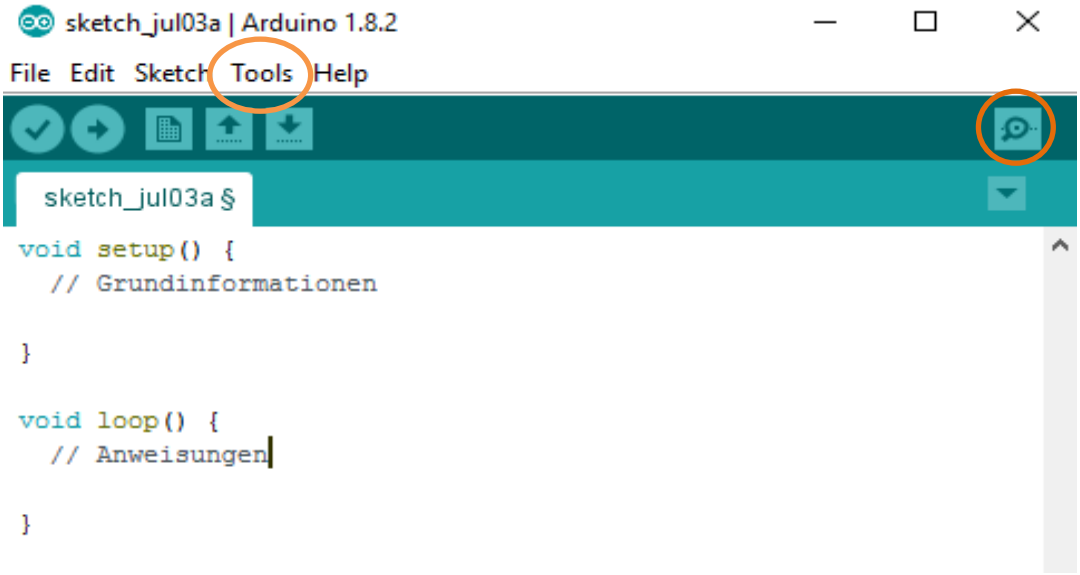


Abbildung 40: Seriellen Monitor in der Arduino-IDE öffnen

Die Befehle, welche für die Ausgabe benötigen werden, sind:

Serial.println(); oder Serial.print();

Der Unterschied der beiden Befehle ergibt sich in der Ausgabe des seriellen Monitors.

Soll ein Text und darunter der Messwert abgebildet werden, so wird folgendes geschrieben:

- für die Anzeige des **Textes**, welcher in Anführungszeichen geschrieben werden muss:

```
Serial.println(„Der Messwert lautet= “);
```

- für die Anzeige des **Messwerts**:

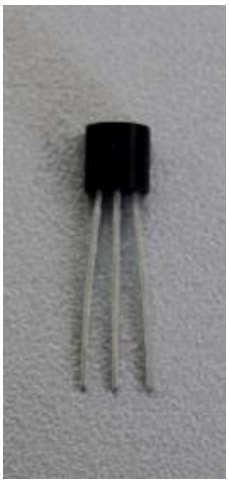
```
Serial.println(wert);
```

Das „ln“ hinter dem Befehl Serial.print bedeutet **line**. Im seriellen Monitor wird zwischen beiden Befehlen ein Zeilenumbruch eingefügt.

Sollen Text und der Wert der Messung **in eine Zeile**, so schreibt man:

```
Serial.print(„Der Messwert lautet= “);

Serial.println(wert);.
```

20. Der Temperatursensor



Um in einem modernen Haus für ein optimales Wohngefühl zu sorgen, empfiehlt es sich einige Parameter ständig zu überwachen. Dazu zählt unter anderem die Raumtemperatur, die nicht im ganzen Haus gleich sein soll. Mit Hilfe eines Temperatursensors ist die Steuerung von Heizung oder Klimaanlage sehr einfach.

Abbildung 41: Temperatursensor tmp36

20.1 Der Aufbau eines Temperatursensors

Der Sensor hat drei Anschlüsse sowie eine flache und eine bauchige Seite. Zeigt die flache Seite zum Betrachter, ist links der Anschluss für die +5 V und rechts GND. Der mittlere Anschluss dient zur Verbindung mit dem Mikrocontroller-Board. Der Sensor gibt an den Mikrocontroller eine Spannung aus, die einer bestimmten Temperatur entspricht. Der Temperatursensor tmp36 von *funduino* arbeitet in einem Wertebereich von 0 V => -50 °C bis 2,0 V => 150 °C. Ist der Temperatursensor falsch angeschlossen, besteht die Gefahr, dass er defekt geht. Mit einer externen Stromversorgung von 9 V kann die Sensorgenauigkeit für diesen Sensor erhöht werden. Die Messwerte können mit Hilfe des seriellen Monitors oder dem seriellen Plotter dargestellt werden.

Das Schaltzeichen eines Temperatursensors tmp36 ist in Abbildung 42 dargestellt.

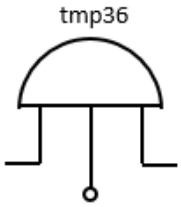


Abbildung 42: Schaltzeichen, Temperatursensor

Der Temperatursensor tmp36 ist ein Bauteil, das aus **Transistoren** aufgebaut ist. Transistoren weisen ein temperaturproportionales Spannungsverhalten auf.

20.2 Der Schaltkreis mit einem Temperatursensor

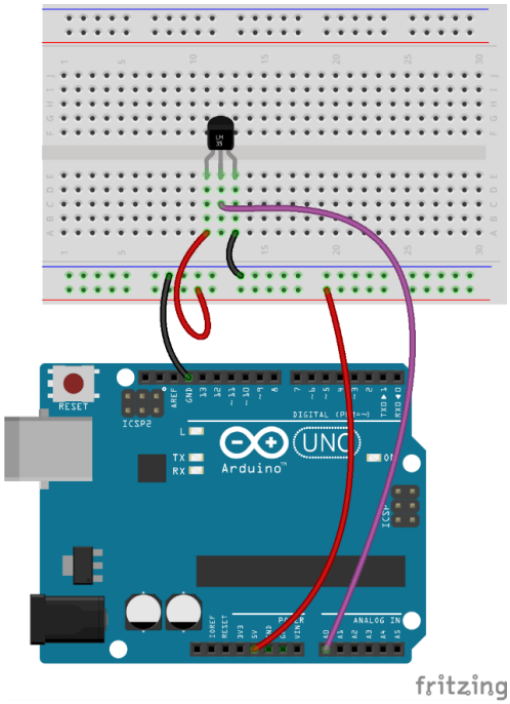


Abbildung 43: Aufbau Breadboard, Temperatursensor

Wie in Abbildung 43 gut zu erkennen ist, wird an den mittleren Anschluss der **Pin** des Mikrocontroller-Boards angeschlossen. An den rechten Anschluss der **GND** und an den linken Anschluss das **höhere Potential** mit +5 V.

Der oben beschriebene Aufbau wird im Schaltplan (Abbildung 44) nochmals sichtbar. Für das Einlesen des Temperatursensors ist darauf zu achten, dass ein analoger Pin (A0-A5) am Mikrocontroller-Board verwendet wird.

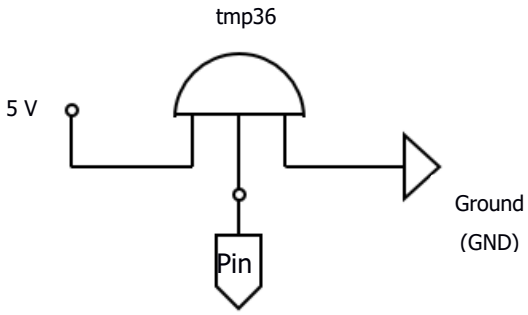


Abbildung 44: Schaltplan eines Temperatursensors

Damit am seriellen Monitor die Messwerte des Temperatursensors ausgegeben werden können, muss im Sketch ein neuer Befehl eingeführt werden.

20.3 Sketch

```
int Temperatursensor=A0;
int Wert;
int Temperatur=0;

void setup() {

pinMode(Temperatursensor, INPUT);
Serial.begin (9600);
}

void loop() {

Wert=analogRead (Temperatursensor);
Temperatur=map(Wert, 0, 410, -50, 150);
Serial.println (Temperatur);
}
```

Zu Beginn werden drei Variablen festgelegt. Am analogen Eingang A0 ist der Sensor mit dem Mikrocontroller-Board verbunden. Diesem wird die Variable „Temperatursensor" zugewiesen. Zusätzlich wird eine Variable „Wert" und eine Variable „Temperatur = 0" eingeführt.

Da die Temperatur gemessen werden soll, wird der Pin, an dem der Temperatursensor angeschlossen ist, in diesem Fall als INPUT verwendet (analoge Pins müssen nicht unbedingt als INPUT festgelegt werden). Mit

Serial.begin (9600);

wird die Datenübertragung initialisiert.

Anschließend liest der Mikrocontroller die Spannungswerte des Temperatursensors aus und ordnet ihnen mit dem Befehl:

Variable=map(a, b, c, d, e);

eine Temperatur zu.

a … eingelesener Analogwert
b … untere Grenze des Wertebereichs des digitalen Signals, welches aus dem analogen Signal gebildet wurde
c … obere Grenze des Wertebereichs des digitalen Signals, welches aus dem analogen Signal gebildet wurde
d … untere Analoggrenze, die der Sensor liefern kann
e … obere Analoggrenze, die der Sensor liefern kann

Dies sieht am Beispiel wie folgt aus:

```
Temperatur=map(Wert, 0, 410, -50, 150);
```

- Sensor misst zwischen -50°C und 150°C
- Sensor gibt Spannung zwischen 0 V – 2 V aus
- Spannung wird einem Wert von 0 - 410 zugeordnet
- Diese Zahlenwerte müssen dann einer Temperatur zugeordnet werden

Mit dem Befehl

```
Serial.println();
```

werden die zugeordneten Temperaturwerte am seriellen Monitor angezeigt.

21. Der Tropfensensor

Der Tropfensensor kann bei einem Regenwarnsystem eingesetzt werden, z. B. bei einem Autoscheibenwischer, der automatisch bei Regen aktiviert wird oder bei einer Markise, welche sich bei Regen von alleine einrollt.

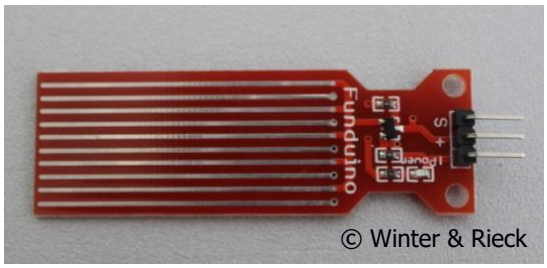


Abbildung 45: Tropfensensor

Es gibt viele unterschiedliche Sensoren, die Wasser registrieren können. Hier wird als Beispiel ein Tropfensensor vorgestellt, der aufgrund von Wasser leitfähig wird und dadurch eine Spannung ausgibt.

Das Schaltsymbol eines Tropfensensors ist in Abbildung 46 dargestellt.

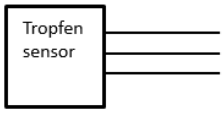


Abbildung 46: Schaltsymbol für einen Tropfensensor

21.1 Der Aufbau eines Tropfensensors

Der Sensor besteht aus langen Metallkontakten, an denen eine Spannung anliegt. Fällt ein Tropfen Wasser auf die Kontakte, kann ein Strom von einem Kontakt zum anderen fließen.

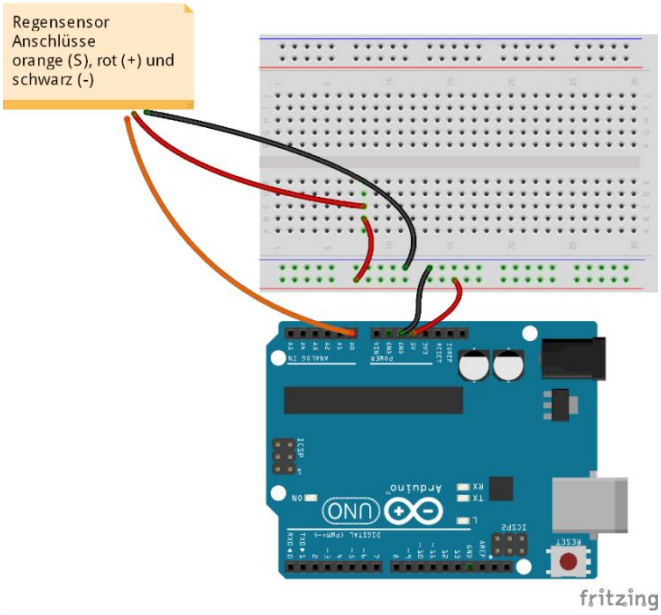
Die Leitfähigkeit von Wasser

Die Leitfähigkeit von Wasser beruht hauptsächlich auf den darin gelösten Ionen, welche den elektrischen Strom transportieren. Reines Wasser ist kein besonders guter Leiter. Je höher die Ionenkonzentration im Wasser ist, desto größer ist auch die Leitfähigkeit

In Wasser findet auch eine Protolyse-Reaktion statt, wenn sich das CO_2 aus der Luft als Kohlensäure im Wasser löst. Dies trägt ebenfalls – wenn auch gering –zur Leitfähigkeit des Wassers bei. Aber auch durch den Grotthuß-Mechanismus, also dem Aufbrechen und Neubilden von Wasserstoffbrückenbindungen, wird Wasser leitfähig.

Der Sensor gibt unterschiedliche Spannungen aus, die einer bestimmten Tropfenmenge zugeordnet werden können. Jeder anliegenden Spannung zwischen 0 V und 5 V wird ein Zahlenwert zwischen 0 und 1023 zugeordnet. Befindet sich kein Wasser auf dem Sensor, wird der Wert 0 zugeordnet. Je mehr Tropfen sich auf dem Sensor befinden, desto höher wird der zugeordnete Wert. Der Mikrocontroller bereitet die Spannung in ein für ihn verständliches Signal auf (Maschinensprache) und es können anschließend in der Arduino-IDE über den seriellen Monitor die Messwerte angezeigt werden.

21.2 Der Schaltkreis mit einem Tropfensensor



Der Tropfensensor besitzt drei Anschlüsse: einen orangenen Anschluss, einen roten Anschluss und einen schwarzen Anschluss. An den roten Anschluss werden die 5 V angeschlossen und an den schwarzen Anschluss der **GND**. Den orangenen Anschluss muss man mit einem analogen **Pin** verbinden, damit sein Signal als INPUT vom Mikrocontroller eingelesen werden kann.

Abbildung 47: Aufbau Breadboard, Tropfensensor

In Abbildung 48 ist ein Tropfensensor in einem Schaltplan dargestellt.

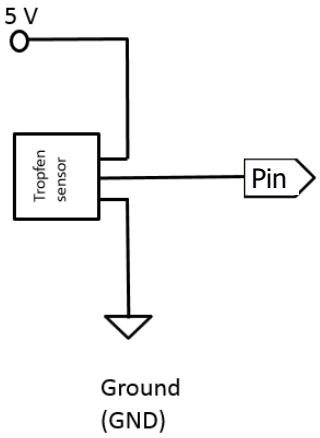


Abbildung 48: Schaltplan eines Tropfensensors

Beim Einsatz des Tropfensensors ist allerdings zu beachten, dass bei sekündlicher Messung durch direkten und durchgängigen Wasserkontakt Schäden am Sensor aufgrund der Elektrolyse entstehen können.

21.3 Sketch

```
int Tropfensensor=A1;
int Wert=0;

void setup() {

Serial.begin(9600);

}

void loop() {

Wert=analogRead(Tropfensensor);
Serial.println(Wert);
delay(1000);
}
```

Neue Befehle sind in diesem Sketch nicht notwendig. Zuerst werden wie gewohnt die Variablen festgelegt. Um die gemessenen Werte im seriellen Monitor darstellen zu können, wird im Setup-Bereich der Befehl:

Serial.begin(9600);

und im Loop-Bereich der Befehl:

Serial.println();

benötigt.

Eingelesen werden die Daten mit dem Befehl:

analogRead(Pin);.

Die Pause ist sinnvoll, um die gemessenen Werte darstellen und erkennen zu können, sowie dem oben beschriebenen Schädigungsprozess entgegenzuwirken.

22. Der Helligkeitssensor

Abbildung 49: Helligkeitssensor LDR

An heißen Sommertagen sorgt eine Verdunklung von Wohn- und Arbeitsräumen für angenehmere Temperaturen. Vielleicht soll aber auch ein Licht angehen, wenn es zu dunkel wird. Mit Helligkeitssensoren können gezielt Helligkeitswerte eingelesen werden und durch die Programmierung andere Bauteile gesteuert werden, die anschließend automatisch für eine Verdunklung oder für Licht sorgen. Ein Beispiel für einen Helligkeitssensor ist ein Photowiderstand (LDR: Light Dependent Resistor). Dieser verändert seinen Kennwert entsprechend der registrierten Lichtverhältnisänderung in der Umgebung.

22.1 Der Aufbau und die Funktion eines Photowiderstandes

Der Photowiderstand ist ein Halbleiterbauelement. Dieser besteht, wie in Abbildung 49 ersichtlich, aus zwei Kupferdrähten, welche auf einer weißen Unterlage befestigt sind. Das „rote Band" ist der Halbleiter. Das Schaltzeichen eines Photowiderstandes ist in Abbildung 50 dargestellt.

Abbildung 50: Schaltzeichen, Helligkeitssensor bei *fritzing* und allgemein

Fällt auf einen Photowiderstand Licht unterschiedlicher Intensität, verändert das Bauteil seinen Widerstand.

Der Photowiderstand

Das Material, aus dem ein Photowiderstand besteht, ist ein Halbleitermaterial wie z B. Cadmiumsulfid (CdS).

Trifft Licht einer bestimmten Wellenlänge auf den Photowiderstand, werden Elektronen aus dem Kristallverband herausgelöst und es entstehen Elektron-Loch-Paare. Diese diffundieren aufgrund der von außen angelegten Spannung in unterschiedliche Richtungen. Es entsteht ein Strom und der Widerstand wird geringer. Außerdem spielen Elektronen von Fremdatomen eine wichtige Rolle. Sie können ebenfalls leicht durch Photonen angeregt werden und zur Leitfähigkeit beitragen. Je mehr Photonen auf das Material treffen, umso stärker wird dieser Effekt.

22.2 Der Schaltkreis mit einem Photowiderstand

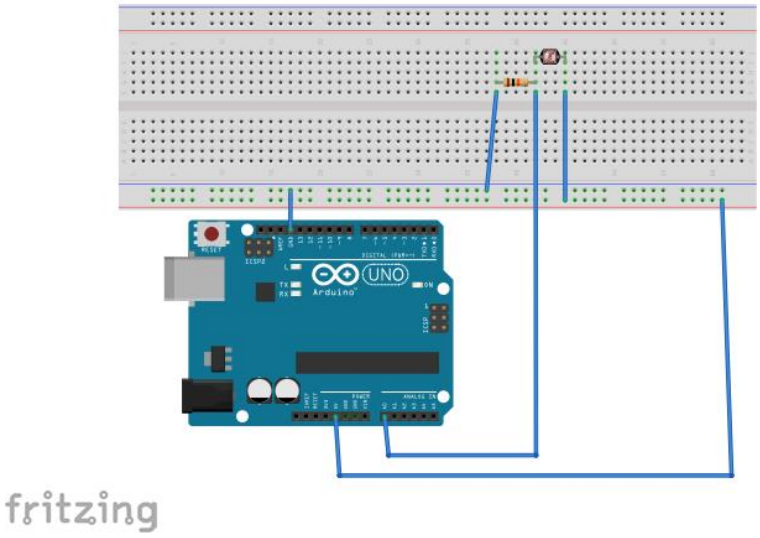


Abbildung 51: Aufbau Breadboard, Photowiderstand

In Abbildung 52 ist ein Helligkeitssensor in einem Schaltplan dargestellt. Der erste Anschluss des Photowiderstandes muss mit dem **höheren Potential** verbunden werden. Der zweite Anschluss wird an den Widerstand angeschlossen und dieser anschließend wiederum mit dem **GND** verbunden. Zwischen den beiden Widerständen wird das Spannungssignal abgegriffen und dem analogen Pin des Mikrocontrollers zugeführt.

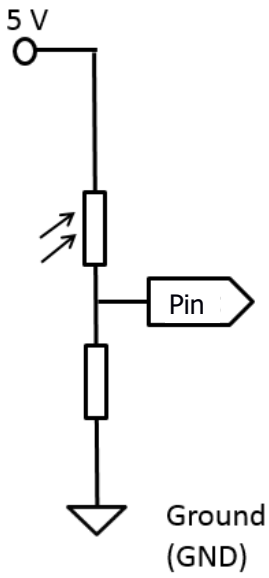


Abbildung 52: Schaltplan eines Helligkeitssensors

Die Aufgabe des zweiten Widerstands im Schaltkreis mit einem LDR:

Benötigt wird der zweite Widerstand einerseits, um einen Kurzschluss bei zu geringem Widerstand am Photowiderstand zu vermeiden.
Denn je mehr Licht auf den Photowiderstand fällt, desto kleiner wird dessen Widerstand und umso größer die Leitfähigkeit. Andererseits dient der zweite Widerstand insbesondere dafür, um sich die Gesetzmäßigkeiten der Reihenschaltung im Spezialfall (unbelasteter) Spannungsteiler zunutze zu machen. Ohne ihn könnte das Signal des Photowiderstandes überhaupt nicht verarbeitet werden.

Dies muss man sich wie folgt vorstellen:
Die an den Photowiderstand und den Widerstand angelegte Spannung von 5 V teilt sich entsprechend den Gesetzmäßigkeiten der Reihenschaltung auf beide Widerstände auf. Dabei fällt am größeren Widerstand die größere Spannung ab und am kleineren Widerstand die kleinere Spannung. In Summe ergeben beide Spannungen aber immer 5 V. Durch Lichteinfall auf den Photowiderstand verändert sich auch dessen Eigenwiderstand, der kleiner wird. Dadurch verteilt sich der größere Teil der 5 V Spannung auf den Widerstand der in Reihe geschaltet ist und einen festen Widerstandswert besitzt.

Da der zweite Widerstand eine ähnliche Größe wie der Photowiderstand besitzen soll, hängt es vom Modellaufbau ab, wie groß er dimensioniert wird. Hierfür ist in den Datenblättern des verwendeten Sensors zunächst dessen Nennwiderstand zu ermitteln. Meist liegt der Widerstand zwischen 1 kΩ und 10 kΩ.

22.3 Sketch

```
int Helligkeitssensor=A2;
int Helligkeitswert=0;

void setup() {

Serial.begin(9600);
}

void loop() {

Helligkeitswert=analogRead(Helligkeitssensor);
Serial.println(Helligkeitswert);
}
```

Für diesen Sensor sind keine neuen Befehle erforderlich. Alle Erklärungen sind den vorherigen Sketchen zu entnehmen.

23. Die Library

Um manche Bauteile ohne viel Aufwand betreiben zu können, gibt es vorgefertigte Programmteile, die durch Einfügen leicht in den Sketch integriert werden können.

Die Library wird bei den meisten Motoren und einigen Sensoren verwendet. Die Library dient dazu, dass unnötig große Programme im Sketch geschrieben werden müssen. Der Sketch gestaltet sich dadurch übersichtlicher und die Ansteuerung ist durch neue Befehle vereinfacht.

Das **Inkludieren der Library** erfolgt am Beispiel eines Servomotors. Zu finden ist diese über das Menü

Sketch → Library importieren → Servo

Die Erklärungen zur Funktion der **Ansteuerung** können unter

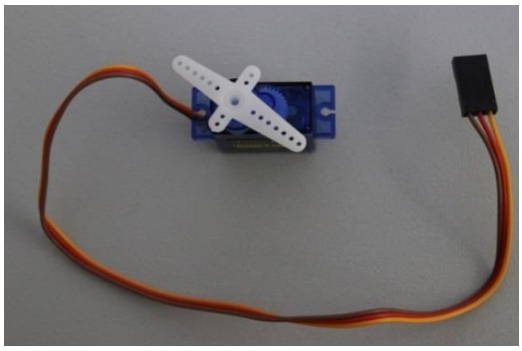
Hilfe → Referenz → Library → Servo

eingesehen werden. Es öffnet sich dabei eine Internetseite, auf der alle nötigen Hinweise in englischer Sprache zu finden sind.

Wird ein Library für den Servomotor eingebunden, so wird dies im Initialisierungs-und Deklarationsbereich angezeigt:

```
#include<Servo.h>
```

24. Der Servomotor

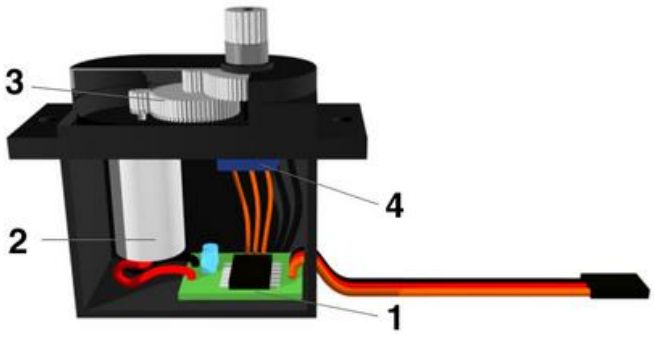


Im Alltag sind Servomotoren häufig in Autos verbaut, um beispielsweise ein Fenster zu öffnen oder zu schließen. Der Servomotor ist in der Lage, seine Position zu messen und dies als Spannungssignal weiterzugeben. Deshalb kommt er besonders oft im Modellbau (z.B. Flugzeuge) zum Einsatz.

Abbildung 53: Servomotor

24.1 Der Aufbau eines Servomotors

Servomotoren können verschiedene Elektromotoren als Antrieb nutzen. Im Modellbau wird meistens ein Gleichspannungsmotor (DC-Servomotor) verwendet. Der Servomotor von *funduino* besteht aus einem DC-Elektromotor, der mit Hilfe eines Potentiometers eine Positionsbestimmung der Motorwelle durchführen kann (siehe Abbildung 54). Das Potentiometer ändert seinen Widerstand je nach Winkeleinstellung. Durch diese Information können das vom Mikrocontroller übermittelte Spannungssignal und das Potentiometersignal in der Motorsteuerungseinheit miteinander abgeglichen werden. Die gewünschte Winkelposition kann anschließend mit dem Elektromotor eingestellt werden.



1: Motorsteuerung
2: Elektromotor
3: Getriebe
4: Potentiometer

Quelle: http;//arduino-tutorial.de/servo/, Zugriff am: 20.10.2017

Abbildung 54: Aufbau Servomotor

Das Schaltzeichen eines Servomotors ist in Abbildung 55 dargestellt.

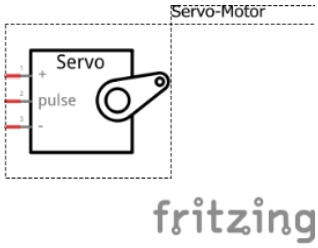


Abbildung 55: Schaltzeichen, Servomotor

24.2 Der Schaltkreis mit einem Servomotor

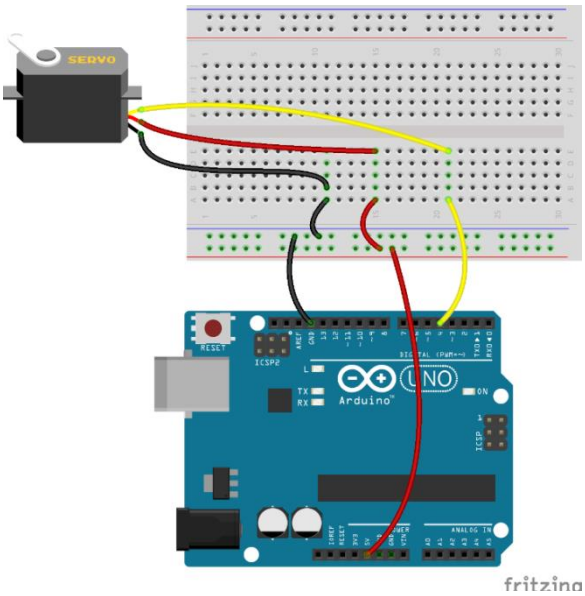


Abbildung 56: Aufbau Breadboard, Servomotor

In Abbildung 56 ist der Servomotor mit seinen drei Anschlüssen dargestellt. Der rote Anschluss wird mit dem **hohen Potential** verbunden und der schwarze Anschluss mit dem **GND**. Der gelbe Anschluss wird an einen digitalen **Pin** angeschlossen.

In Abbildung 57 ist ein Servomotor in einem Schaltplan dargestellt.

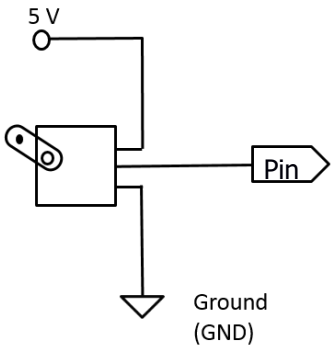


Abbildung 57: Schaltung des Servomotors

24.3 Sketch

```
#include <Servo.h>
Servo myservo;

void setup() {

myservo.attach(13);

}

void loop() {

myservo.write (90);
delay(1000);
myservo.write (0);
delay (1000);
}
```

Wie bereits im vorherigen Kapitel „Library" ausführlich erklärt, wird in den Sketch eine Bibliothek für den Servomotor eingefügt.

#include<Servo.h>

Anschließend muss der Servomotor im **Initialisierungs- und Deklarationsbereich** benannt werden, beispielsweise:

Servo myservo;

Als nächstes muss im **Setup-Bereich** definiert werden, an welchem Pin sich der Servomotor befindet:

myservo.attach(13);

Zuletzt wird im **Loop-Bereich** festgelegt, wie weit sich der Servomotor drehen soll:

myservo.write(Grad);

Im oben abgebildeten Sketch fährt der Hebel des Servomotors auf die Winkelposition 90°, verharrt 1 Sekunde und fährt dann auf 0° weiter.

Allgemein können in den write(…); -Befehl ganze Gradzahlen zwischen 0° und 180° eingesetzt werden.

24.4 PWM Signal

Wird beim Servomotor ohne Library gearbeitet, muss beachtet werden, dass dieses Bauteil ein PWM-Signal (Pulsweitenmodulation) benötigt, um an die richtige Winkelposition gefahren werden zu können.
Der Servomotor dient als Aktor, ist allerdings ein analoges Bauteil. Wenn der Servomotor an das Mikrocontroller-Board angeschlossen wird, muss darauf geachtet werden, dass ein Ausgang gewählt wird, der mit einer Tilde (~) versehen ist. Dieser Ausgang befindet sich bei den digitalen Ein- und Ausgängen. Die Tilde weist darauf hin, dass es sich beim ausgegebenen Signal um ein Pulsweiten-Moduliertes-Signal handelt. Was unter einem solchen Signal zu verstehen ist, ist in Abbildung 58 dargestellt.

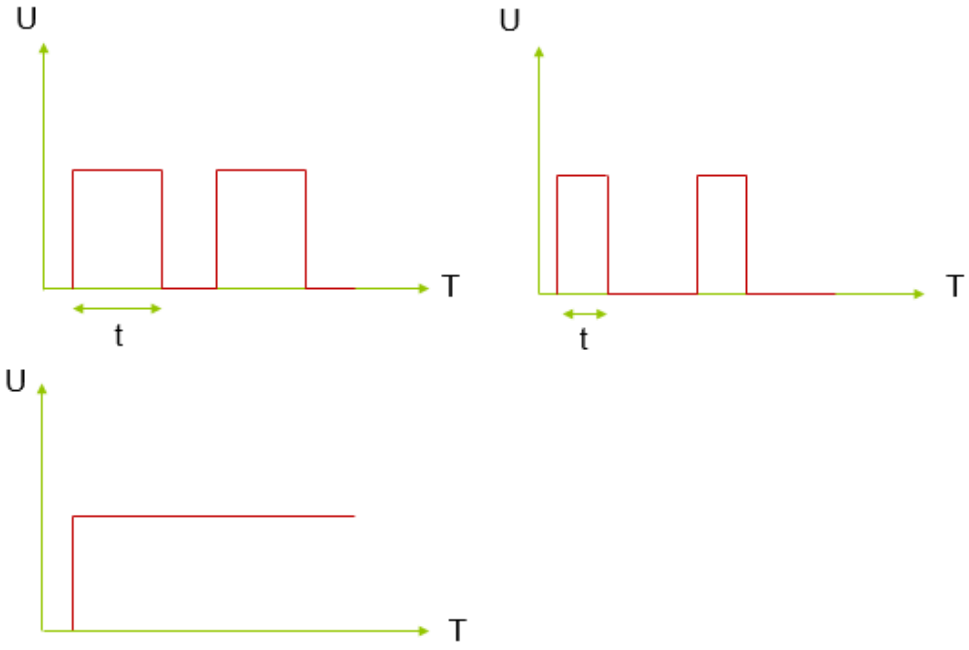


Abbildung 58: Pulsweitenmodulation

Die Pulsweite ist durch den grünen Pfeil verdeutlicht. Im rechten oberen Bild ist die Länge des Pulses am geringsten. Die Pulsweite im unteren Bild ist am längsten, das Signal ist durchgängig auf das Maximum eingestellt.
Die Fläche, die sich unter einem solchen Rechteck befindet, entspricht der übermittelten Energie. Beim Servomotor entspricht eine Pulslänge von 1 ms einer Winkeleinstellung von 0°, 2 ms entsprechen einer Winkeleinstellung von 180°. Alle Werte dazwischen werden dementsprechend zugeordnet. Durch das Einbinden der Library wird diese Zuordnung übernommen.

25. Arbeitsblätter

Blatt 1: Eine LED soll leuchten (ohne Programmierung)

Übertragung eines einfachen Schaltplans auf das Breadboard.

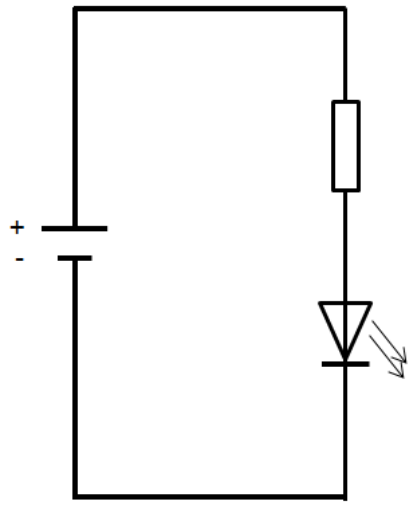


Abbildung 59: Schaltplan, Übung 1

Aufgabe 1:

a) Beschrifte in der obigen Abbildung die Schaltzeichen, zeichne die technische und physikalische Stromrichtung ein und erkläre, um welche Art von Schaltung es sich hierbei handelt.

b) Überführe den Schaltplan auf das Breadboard. Schließe erst dann an das Breadboard die 4,5 V Flachbatterie an und kontrolliere, ob die LED aufleuchtet.

c) Begründe, warum es notwendig ist, einen Vorwiderstand mit in die Schaltung einzubauen.

d) Berechne mit Hilfe des unten abgebildeten Datenblatts die Größe des Vorwiderstands für eine blaue LED, wenn eine äußere Spannung von 5 V anliegt (Verwende zur Berechnung den Mittelwert der Stromstärke).

ABSOLUTE MAXIMUM RATINGS:(Ta=25°C)

Reverse Voltage	:	5 Volt
Reverse Current(Vr=5V)	:	10uA
Operating Temperature Range	:	-40°C to+85°C
Storage Temperature Range	:	-40°C to+100°C
Lead Soldering Temperature	:	260°C for 5 Seconds

PART SELECTION AND APPLICATION INFORMATION(RATINGS AT 25°C AMBIENT)

PART NO.	CHIP			LENS COLOR	Absolute Maximun Ratungs				Electrical-Optcal Characferis						Viewing Angle (deg)
	Material	PEAK WAVE Lenght λp(nm)	Emitting Color						VF(V)			Rec If(mA)	Iv(mcd)		
					Δλ (nm)	pd (nW)	IF (mA)	Peak If(mA)	Mim.	Typ.	Max.	Max.	Mim.	Typ.	
513BD	InGaN	465	Blue	Blue Diffused	20	120	30	100	2.9	3.3	3.6	10-20	300	800	60

Quelle: Kenngrößen für die blaue LED: http://funduino.de/DL/LEDblau.pdf, Zugriff am 20.10.2017

Abbildung 60: Auszug Datenblatt für eine blaue LED

e) Nenne den Farbcode des berechneten Vorwiderstands (Genauigkeit 5%).

Abbildung 61: Undefinierter Widerstand

Kopfnussaufgabe 1:

Erkläre, wie sich die Größe des Widerstands aus Aufgabenteil 1d) verändern würde, wenn anstatt einer 4,5 V-Batterie eine 9 V-Batterie angeschlossen wird. Was würde passieren, wenn der Widerstand nicht angepasst wird?

Für die Berechnung kann die Gleichung U = R · I hilfreich sein. Zur Erinnerung: 1 mA = 0,001 A.

Alles bearbeitet? ☐ Ergebnis kontrolliert? ☐

Blatt 2: Eine LED soll leuchten (mit Programmierung)

Die Verwendung des Pins als hohes Potential.

Aufgabe 2:

a) Verwende den Aufbau vom Breadboard aus Blatt 1 und ersetze die Batterie durch das Mikrocontroller-Board.

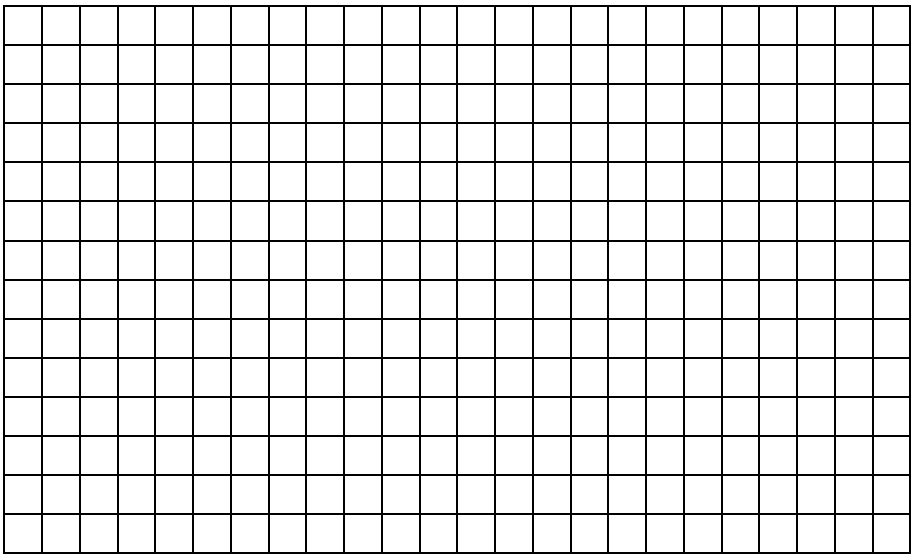
b) Übertrage den folgenden Sketch in die Arduino-IDE.

```
void setup() {
pinMode(5, OUTPUT);

}
void loop() {
digitalWrite(5, HIGH);


}
```

c) Beschreibe in eigenen Worten die einzelnen Befehle im Programmcode.

Kopfnussaufgabe 2:

Erstelle einen Schaltplan aus dem Aufbau am Breadboard mit der Verbindung zum Mikrocontroller-Board aus Aufgabe 2a) und beschrifte die einzelnen Bauteile.



Der Pin an dem die LED angeschlossen wird, wird als OUTPUT definiert und kann zwei verschiedene Pegel haben.

Alles bearbeitet? ☐ Ergebnis kontrolliert? ☐

Blatt 3: Blinklicht

Verschiedene LEDs sollen abwechselnd blinken.

Aufgabe 3:

a) Eine rote LED soll zum Blinken gebracht werden.
Hierfür soll sie für 2 s leuchten und für 2 s nicht leuchten. Schreibe den dafür benötigten Sketch.

b) Die LED soll schneller blinken. Erkläre was verändert werden muss und schreibe den zugehörigen Sketch. Überprüfe, ob die LED schneller blinkt.

c) Ergänze das Breadboard mit einer grünen und einer gelben LED und jeweils einem Widerstand. Schreibe einen Sketch, der die folgende Anweisung erfüllen soll:

Die rote LED soll für drei Sekunden leuchten, die anderen beiden LEDs sollen aus sein. Anschließend soll die gelbe LED für drei Sekunden aufleuchten und die rote und die grüne LED beide aus sein. Zum Schluss soll dann nur die grüne LED für 3 Sekunden alleine aufleuchten.

d) Der folgende Sketch hat die Aufgabe eine rote LED für zwei Sekunden zum Leuchten zu bringen, danach soll sie für fünf Sekunden aus sein. Überprüfe den Sketch und erkläre, weshalb der Sketch nicht wie geplant funktioniert. Verbessere die Fehler auf dem Arbeitsblatt.

```
void setup() {

pinMode(5, INPUT);

}

void loop() {

digitalWrite(5, HIGH);
delay(200);
digitalWrite(5,LOW);
delay(5000);

}
```

Kopfnussaufgabe 3:

Erstelle einen Sketch für ein kleines Lauflicht zu dem Aufbau aus Aufgabenteil 3c). Ein Lauflicht hat die Eigenschaften, dass die LEDs schnell hintereinander aufleuchten und sobald das eine Ende erreicht ist, kehrt sich die Laufrichtung um.

Leuchtet die LED, so wird HIGH geschrieben, leuchtet sie nicht, so wird LOW geschrieben.

Alles bearbeitet? ☐	Ergebnis kontrolliert? ☐

Blatt 4: Die LED leuchtet auf Knopfdruck

Wenn der Taster gedrückt ist, soll eine LED leuchten.

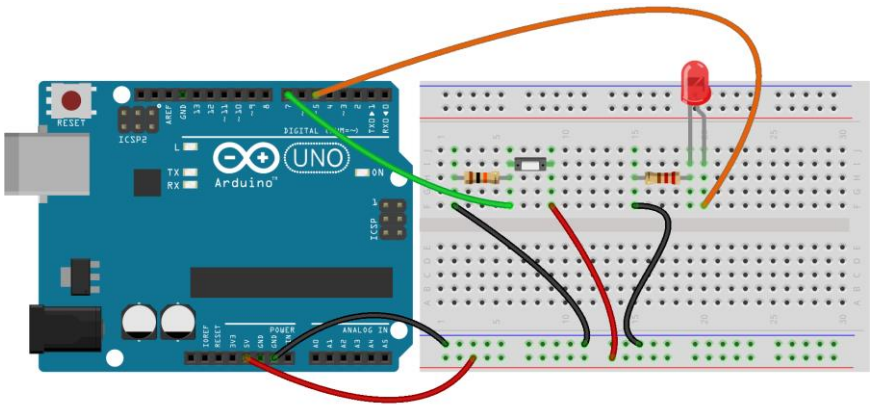


Abbildung 62: Aufbau Breadboard, Leuchten auf Knopfdruck

Aufgabe 4:

a) Übertrage den oben abgebildeten Aufbau auf das Breadboard.

b) Übertrage den folgenden Sketch in die Arduino-IDE.

```
void setup()
{
pinMode(5,OUTPUT);//LED
pinMode(7,INPUT);//Taster
}

void loop()
{
if(digitalRead(7) == HIGH)
{
/*wenn der Taster gedrückt ist, soll eine Anweisung
 ausgeführt werden */
}
else
{
/*wenn der Taster gedrückt ist, soll eine
 andere Anweisung ausgeführt werden */
}
}
```

c) Vervollständige den Sketch aus Aufgabenteil 4b), dieser soll die folgenden Anweisungen umsetzen:

Wenn der Taster gedrückt wird, soll danach die LED für 2 s leuchten. Ist er nicht gedrückt, soll die LED nicht leuchten.

Kopfnussaufgabe 4:
Ergänze das Breadboard aus Aufgabenteil 4c) um eine weitere LED und einen Widerstand. Schreibe einen Sketch, der die folgenden Anweisungen enthält:

Wenn der Taster gedrückt wird, soll nur die erste LED fünfmal blinken. Ist er nicht gedrückt, so soll nur die zweite LED leuchten.

Die if() {...}, else {...} Kontrollstruktur kann weiterhelfen.

Alles bearbeitet? ☐ Ergebnis kontrolliert? ☐

Blatt 5: Der Mikrocontroller macht Töne

Auf Knopfdruck soll das Mikrocontroller-Board Töne mit Hilfe eines Piezospeakers ausgeben, dabei soll eine LED leuchten.

Aufgabe 5:

a) Baue am Breadboard eine Schaltung mit dem Piezospeaker auf. Schreibe einen Sketch für eine Sirene. Hierbei soll der Piezospeaker für 3 s eine Tonhöhe nach Wahl und danach für 3 s keinen Ton ausgeben.

b) Baue am Breadboard eine Schaltung mit den folgenden Elementen auf: Taster, Piezospeaker, LED und den zugehörigen Widerständen.

c) Schreibe zu Aufgabenteil 5b) einen Sketch, der die folgenden Anweisungen erfüllt:

Wenn der Taster gedrückt wird, soll danach die LED für 2 s leuchten. Gleichzeitig und gleich lang soll der Piezospeaker dabei einen Ton von 440 Hz ausgeben. Wenn der Taster nicht gedrückt ist, soll weder der Piezospeaker ertönen, noch die LED aufleuchten.

Kopfnussaufgabe 5:

Überprüfe den folgenden Schaltplan und arbeite die Fehler heraus. Erstelle einen korrekten Schaltplan.

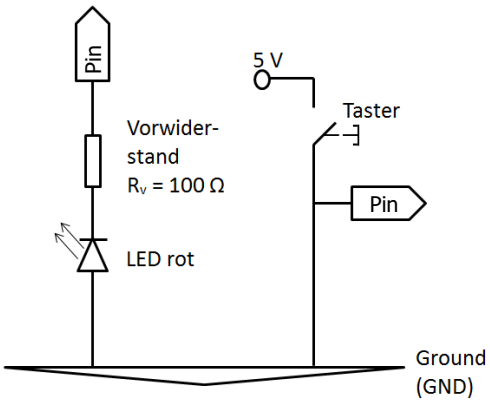


Abbildung 63: Fehlerhafter Schaltplan

Der Befehl tone(…); und noTone(…); kann weiterhelfen.

Alles bearbeitet? ☐ Ergebnis kontrolliert? ☐

Blatt 6: Die Verkehrsampel I

Realisierung eines Modells einer Verkehrsampel.

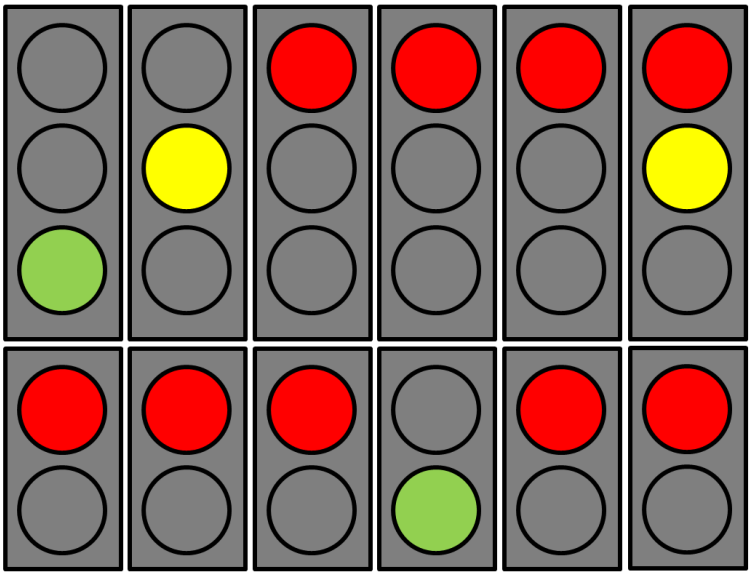


Abbildung 64: Ampelphasen

Aufgabe 6:

Entwickle und programmiere ein Modell einer Verkehrsampel. Diese soll folgende Randbedingungen erfüllen:

Die Autoampel ist grün, die Fußgängerampel ist dabei rot.
Wenn der Taster durch einen Fußgänger gedrückt wird, ändern sich die Ampelphasen. Die Autoampel schaltet von Grün auf Gelb und dann weiter auf Rot. Anschließend ist eine Sicherheitszeit einzubeziehen, bevor die Fußgängerampel wieder auf Grün schaltet und ein Tonsignal ertönt. Nach einer kurzen Grünphase sollen die Ampeln wieder in die Ausgangssituation zurückschalten.

Hilfreich kann hierbei das Erstellen eines kurzen Projektplans sein. Die Blätter 2 - 5 beinhalten die notwendigen Hilfestellungen.

Alles bearbeitet? ☐	Ergebnis kontrolliert? ☐

Blatt 7: Die Verkehrsampel II

Fehlerhaften Sketch analysieren.

Aufgabe 7:

Beim Schreiben des folgenden Sketchs sind vier Fehler unterlaufen. Finde die Fehler heraus.

```
int RAuto=12;
int GAuto=11;
int GrAuto=10;
int RFuss=7;
int GrFuss=6;
int Pieps=2;
int Button=4;
int Druck

void setup()
{
pinMode(RAuto,OUTPUT);
pinMode(GAuto,OUTPUT);
pinMode(GrAuto,OUTPUT);
pinMode(RFuss,OUTPUT);
pinMode(GrFuss,OUTPUT);
pinMode(Pieps,INPUT);
pinMode(Button,INPUT);
}
void loop() {
Druck = Digitalread(Button);

 if (Druck == HIGH)
 {
 digitalWrite(RAuto, LOW);
 digitalWrite(GAuto, LOW);
 digitalWrite(GrAuto, HIGH);
 digitalWrite(RFuss, HIGH);
 digitalWrite(GrFuss, LOW);
  delay(200);
```

```
digitalWrite(RAuto, LOW);
digitalWrite(GAuto, HIGH);
digitalWrite(GrAuto, LOW);
digitalWrite(RFuss, HIGH);
digitalWrite(GrFuss, LOW);
 delay(2000);

digitalWrite(RAuto, HIGH);
digitalWrite(GAuto, LOW);
digitalWrite(GrAuto, LOW);
digitalWrite(RFuss, HIGH);
digitalWrite(GrFuss, LOW);
 delay(2000);


digitalWrite(RAuto, HIGH);
digitalWrite(GAuto, LOW);
digitalWrite(GrAuto, LOW);
digitalWrite(RFuss, LOW);
digitalWrite(GrFuss, HIGH);
tone(Pieps,450);
 delay(6000);

digitalWrite(RAuto, HIGH);
digitalWrite(GAuto, LOW);
digitalWrite(GrAuto, LOW);
digitalWrite(RFuss, HIGH);
digitalWrite(GrFuss, LOW);
noTone(Pieps);
 delay(2000);

digitalWrite(RAuto, HIGH);
digitalWrite(GAuto, HIGH);
digitalWrite(GrAuto, LOW);
digitalWrite(RFuss, HIGH);
digitalWrite(GrFuss, LOW);
 delay(2000);


else
{
digitalWrite(RAuto, LOW);
digitalWrite(GAuto, LOW);
digitalWrite(GrAuto, HIGH);
digitalWrite(RFuss, HIGH);
digitalWrite(GrFuss, LOW);
}
}
```

Alles bearbeitet? ☐	Ergebnis kontrolliert? ☐

Blatt 8: Variablen definieren

Durch die Einführung von Variablen soll mehr Übersichtlichkeit im Sketch geschaffen werden.

Aufgabe 8:

Im Aufgabenblatt 6 wurde eine Ampelschaltung programmiert. Ersetze dort die LEDs durch Variablen, die vor dem void setup definiert werden. Die Variablen können frei benannt werden. Achte bei der Benennung der Variablen auf Eindeutigkeit und klare Zuordnung.

Es ist darauf zu achten, dass es durch ungenaues Benennen zu keinen Verwechslungen kommt.

Alles bearbeitet? ☐	Ergebnis kontrolliert? ☐

Blatt 9: Der Temperatursensor

Der Temperatursensor soll an das Mikrocontroller-Board angeschlossen werden, um am seriellen Monitor eingelesen werden zu können. Durch die erhaltenen Messwerte soll ein Lichtsignal ausgegeben werden, welches die Temperatur verdeutlicht.

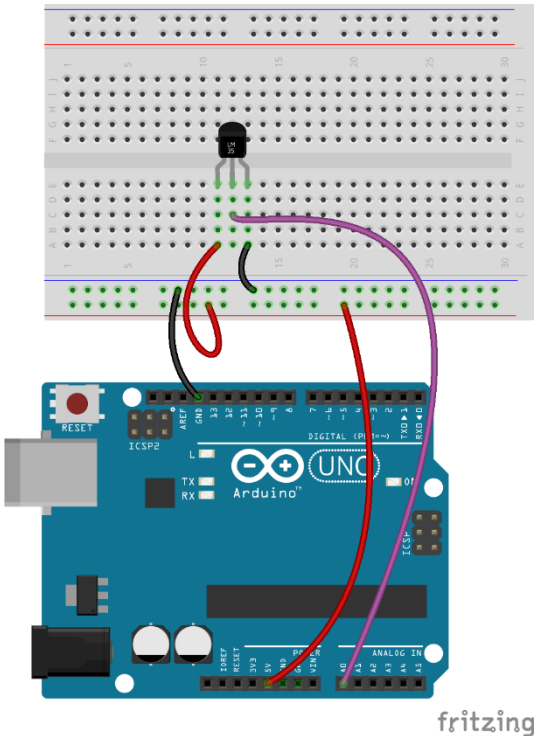


Abbildung 65: Aufbau Breadboard, Temperatursensor.

Aufgabe 9:

a) Überführe den Aufbau aus der oberen Abbildung auf das Breadboard.

b) Übertrage den Sketch und ließ die Werte am seriellen Monitor aus. Variiere die Temperatur durch Erwärmen bzw. Kühlen.

```
int TempSensor=A0;
int Wert;
int temperatur=0;

void setup()
{
Serial.begin(9600);
}

void loop()
{
Wert=analogRead(TempSensor);
temperatur=map(Wert,0,410,-50,150);
delay(500);
Serial.print(temperatur);
Serial.println(" Grad Celcius!");
}
```

c) Erkläre den oberen Sketch in eigenen Worten.

Kopfnussaufgabe 9:

Ergänze den Sketch und das Breadboard wie folgt:

1. Bei optimaler Raumtemperatur von 20 °C bis 22 °C soll eine grüne LED aufleuchten. Ist es zu warm soll eine rote LED aufleuchten und ist es zu kalt eine blaue LED.
2. Um die Bewohner eines Hauses vor einer starken Hitzeentwicklung zu warnen (z.B. durch ein Feuer), soll eine Alarmanlage angehen.

Soll eine Bedingung zwischen zwei Werten liegen, verwende den Befehl: „&&".

Alles bearbeitet? ☐ Ergebnis kontrolliert? ☐

Blatt 10: Der Tropfensensor

Der Tropfensensor soll an das Mikrocontroller-Board angeschlossen werden, um am seriellen Monitor eingelesen werden zu können. Durch die erhaltenen Messwerte soll ein Signal ausgegeben werden, das vor Regen warnt.

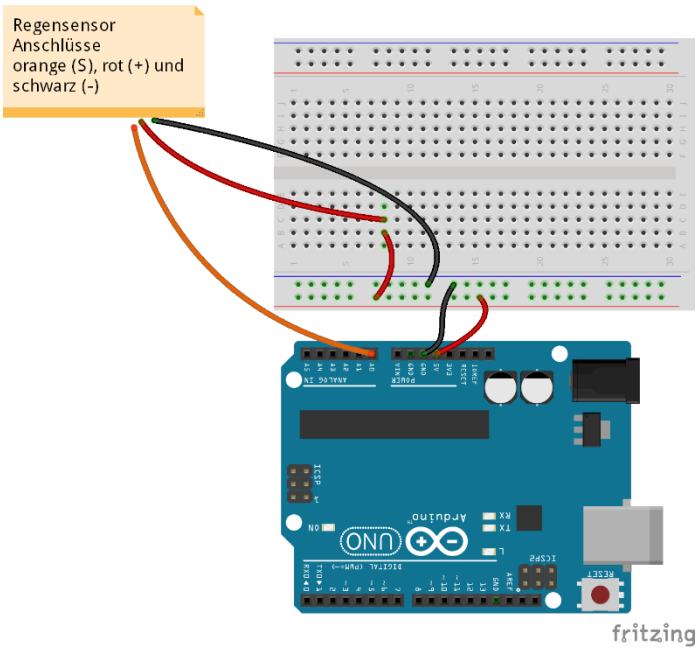


Abbildung 66: Aufbau Breaboard, Tropfensensor.

Aufgabe 10:

a) Erkläre anhand des EVA-Prinzips, wie der Regensensor mit Hilfe des Mikrocontrollers ausgelesen wird und ein Aktor angesprochen werden kann.

b) Überführe den Aufbau aus der oberen Abbildung auf das Breadboard.

c) Übertrage den Sketch und ließ die Werte am seriellen Monitor bei unterschiedlicher Tropfenanzahl aus.

```
int Tropfensensor = A0;
int Wert = 0;

void setup()
{
  Serial.begin(9600);
}

void loop()
{
 Wert = analogRead(Tropfensensor);
 Serial.println(Wert);
 delay(1000);
}
```

Kopfnussaufgabe 10:

Ergänze den Sketch und das Breadboard wie folgt:

Wenn es regnet soll eine LED leuchten und der Piezospeaker ein Tonsignal ausgeben. Überschreitet die Tropfenmenge einen gewissen Wert, so soll die Tonhöhe des Piezospeaker steigen.

Alles bearbeitet? ☐	Ergebnis kontrolliert? ☐

Blatt 11: Der Helligkeitssensor

Der Helligkeitssensor soll am Mikrocontroller-Board angeschlossen werden, um am seriellen Monitor eingelesen werden zu können. Durch die erhaltenen Messwerte soll ein Servomotor angesteuert werden.

Aufgabe 11:

a) Schließe den Helligkeitssensor an das Mikrocontroller-Board an und lies die Werte am seriellen Monitor aus.

b) Wenn ein bestimmter Helligkeitswert überschritten wird, soll sich ein Servomotor um 180° drehen.

Um den Servomotor anzusteuern, verwende die Library: **Servo**.

Danach muss der Servo benannt werden → z.B.: Servo servolein;

Den Pin legt man mit dem Befehl
→ servolein.attach(Pin); fest

Um den Servomotor drehen zu lassen, benötigt man den Befehl:
→ servolein.write(Grad);

Der Servomotor wird wie folgt angeschlossen: Orange mit dem Pin, rot mit den 5 V und braun mit dem GND.

Alles bearbeitet? ☐	Ergebnis kontrolliert? ☐

Literaturverzeichnis

Für die Erstellung der Texte, Abbildungen und Tabellen im vorliegenden Schulungsheft wurden die nachfolgend aufgeführten Literaturquellen, Bücher und Internetseiten verwendet. Falls Bedarf zum Nachlesen besteht, Ideen und Anregungen für den Schulunterricht benötigt werden oder auch nur für den eigenen Spaß beim Programmieren wird man hier sicherlich fündig.

Kapitel 1: Abgleich Bildungsplan 2016

Bildungsplan (2016): Bildungsplan des Gymnasiums. Bildungsplan 2016. Naturwissenschaft und Technik (NwT). Profilfach. Ministerium für Kultus, Jugend und Sport Baden-Württemberg (Hrsg.).

Kapitel 4: Der Mikrocontroller

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Brinkschulte, Prof. Dr. Uwe/ Ungerer, Prof. Dr. Theo (2010): Mikrocontroller und Mikroprozessoren. 3. Auflage, Heidelberg u. a.: Verlag Springer, 2010.

Caroli, Philip und Christian (2015): Arduino Handbuch. Haar bei München: Franzis Verlag, 2015.

Sommer, Ulli (2010): Arduino. Mikrocontroller-Programmierung mit Arduino/Freeduino. Poing: Franzis Verlag, 2010.

Kapitel 5: Installation der Software

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Kapitel 6: Das Programmieren – die Entwicklungsumgebung

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Kapitel 7: Die Fehlermeldungen – und deren Behebung

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Kapitel 8: Auf Fehlersuche

Eigene Erstellung – aus vielen verzweifelten Erfahrungen.

Kapitel 9: Wichtige Ausrüstung und Werkzeuge

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Kapitel 11: *fritzing*

fritzing: Unter der Internetseite: http://fritzing.org/home/, Zugriff am: 14.09.2017.

Kapitel 12: Die Leuchtdiode

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Brühlmann, Thomas (2012): Arduino. Praxiseinstieg. Behandelt Arduino 1.0. 2. Auflage, Heidelberg u.a.: mtip, 2012.

Schnabel, Patrick (2017): Elektronik-Fibel. Grundlagen, Bauelemente, Schaltungstechnik, Digitaltechnik. Unter der Internetadresse: http://www.elektronik-fibel.de/, Zugriff am: 24.12.2017 (PDF-Kaufexemplar).

Suter, Dieter: Einführung in die Festkörperphysik. Halbleiter, S. 136-155.
Unter der Internetadresse:
https://e3.physik.uni-dortmund.de/~suter/Vorlesung/Festkoerperphysik_WS12/7_Halbleiter.pdf, Zugriff am: 20.10.2017.

Sommer, Ulli (2010): Arduino. Mikrocontroller-Programmierung mit Arduino/Freeduino. Poing: Franzis Verlag, 2010.

Kapitel 13: Die digitalen Ein- und Ausgänge – Interaktion mit dem Mikrocontroller

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Magolis, Michael (2012): Arduino Kochbuch. Aus dem englischen von Peter Klicman. Köln u.a.: O´Reilly Verlag, 2012.

Kapitel 14: Die LED soll leuchten

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Kappel, Benjamin (2016): Arduino. Elektronik, Programmierung, Basteln. 1. Auflage, Bonn: Rheinwerk Verlag, 2016.

Kapitel 15: Die If-else-Kontrollstruktur

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Kappel, Benjamin (2016): Arduino. Elektronik, Programmierung, Basteln. 1. Auflage, Bonn: Rheinwerk Verlag, 2016.

Sommer, Ulli (2010): Arduino. Mikrocontroller-Programmierung mit Arduino/Freeduino. Poing: Franzis Verlag, 2010.

Kapitel 16: Der Taster – Leuchten auf Knopfdruck

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Kapitel 17: Der Piezospeaker – Der Mikrocontroller macht Geräusche

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Müller, Werner et al. (2015): Tier- und Humanphysiologie. Eine Einführung. 5. Auflage, Berlin, Heidelberg: Springer Verlag, 2015.

Kapitel 18: Festlegung von Variablen

Magolis, Michael (2012): Arduino Kochbuch. Aus dem englischen von Peter Klicman. Köln u.a.: O´Reilly Verlag, 2012.

Kapitel 19: Die analogen Eingänge – Der Mikrocontroller kommuniziert

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Kappel, Benjamin (2016): Arduino. Elektronik, Programmierung, Basteln. 1. Auflage, Bonn: Rheinwerk Verlag, 2016.

Sommer, Ulli (2010): Arduino. Mikrocontroller-Programmierung mit Arduino/Freeduino. Poing: Franzis Verlag, 2010.

Kapitel 20: Der Temperatursensor

Funduino.de: https://funduino.de/nr-9-temperatur-messen, abgerufen am 20.10.2017.

Datenblatt und Informationen zum Temperatursensor: http://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf, abgerufen am 20.10.2017.

Kapitel 21: Der Tropfensensor

Funduino.de: https://funduino.de/nr-17-tropfensensor, abgerufen am 20.10.2017.

Johannes Gutenberg Universität Mainz: http://www.uni-mainz.de/presse/57094.php, abgerufen am 20.10.2017.

Kapitel 22: Der Helligkeitssensor

Funduino.de: https://funduino.de/nr-6-fotowiderstand, abgerufen am 20.10.20017.

Sommer, Ulli (2010): Arduino. Mikrocontroller-Programmierung mit Arduino/Freeduino. Poing: Franzis Verlag, 2010.

Kapitel 23: Die Library

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Kapitel 24: Der Servomotor

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Lönarz, Dominik (2012): http://www.dloenarz.de/wp-content/uploads/2012/11/Servomotoren.pdf, abgerufen am 20.10.2017.

Datenblätter

Datenblatt der blauen LED: Unter der Internetadresse: http://funduino.de/DL/LEDblau.pdf, Zugriff am 20.10.2017.

Datenblatt der roten LED: Unter der Internetadresse: http://funduino.de/DL/LEDrot.pdf, Zugriff am: 20.10.2017.

Schaltzeichen des Mikrocontroller ATmega 328 (Abbildung 1)

Unter der Internetadresse: https://www.arduino.cc/en/Hacking/PinMapping168, Zugriff am: 20.10.2017.

Technische Daten Mikrocontroller-Board (Tabelle 2)

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.

Farbcodetabelle (Tabelle 4)

Bartmann, Erik (2014): Die elektronische Welt mit Arduino entdecken. 2. Auflage, Köln u.a.: O'Reilly Verlag, 2014.