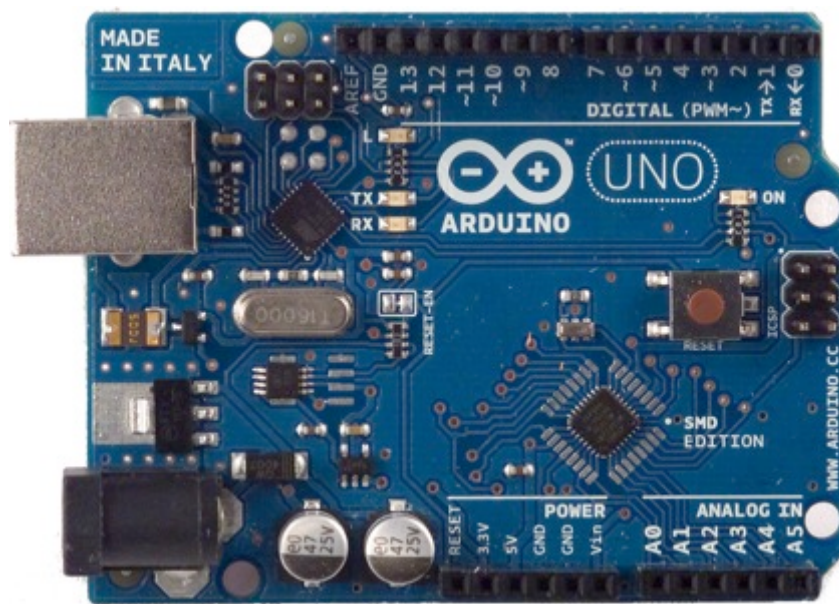


ARDUINO Handbuch



Einführung

Name: _____

Nummer von Box und Computer: _____

8. Klassen
Technisches Gestalten
OSZ Rapperswil

Inhalt

1	Einführung	4
1.1	Arbeitsweise	4
1.2	Was ist ein Mikrocontroller?	5
2	Arduino	6
2.1	Hardware	6
2.2	Software	7
3	Programmieren	8
3.1	Programmieren mit Arduino	8
3.2	Zeichen und Klammern	8
3.3	Aufbau eines Sketchs	9
3.4	Sketch unter der Lupe	11
3.5	Sketch speichern	12
4	Bauteile 1	13
4.1	Steckplatine/Breadboard	13
4.2	Jumperkabel	14
4.3	Verbindungskabel	14
4.4	USB-Kabel	14
4.5	LED	15
4.6	Widerstände	16
5	Übung 1: LEDs leuchten und blinken lassen	17
5.1	1 LED leuchten lassen	18
5.2	1 LED blinken lassen	21
5.3	5 LEDs blinken lassen	24
6	Bauteile 2	27
6.1	Schalter	27
6.2	Taster	27
7	Übung 2: LED mit einem Taster einschalten	28
8	Bauteile 3	30
8.1	Lautsprecher	30
8.2	Piezo-Summer/Buzzer	30
8.3	Lautsprecher vorbereiten	31
9	Übung 3: Töne erzeugen	32

9.1	Piezosummer summen lassen	32
9.2	Melodie aus Lautsprecher	34
10	Beurteilungspunkte	37

1 Einführung

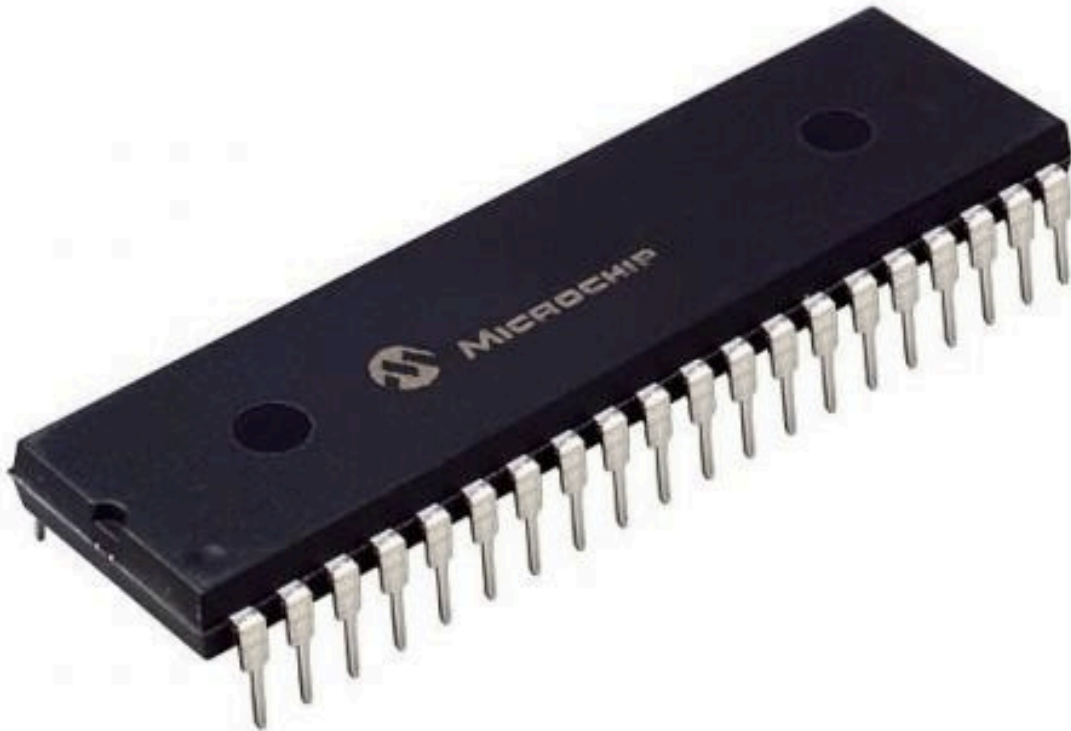
Woher weiss das Smartphone was es tun muss, wenn du ein App anklickst? Woher kommen die Inhalte im App? Wie weiss ein Auto, dass es piepsen muss, wenn deine Eltern beim Einparkieren zu nahe ans nächste Auto fahren? Wie wird eine Ampelanlage gesteuert? Wie weiss eine Maschine was sie machen muss? Geräten und Maschinen wird „gesagt“, was ihre Aufgabe ist. Da sie dir beim Sprechen aber nicht zuhören, muss man ihnen via Computer Befehle erteilen. Man programmiert die Geräte und Maschinen, damit man sie steuern kann. Damit man aber nicht zu jedem Gerät einen Computer herumtragen muss, speichert man die Befehle auf Mikrocontrollern und diese geben die Befehle weiter. Menschen erteilen den Geräten und Maschinen mit einem Programm Befehle. Dies nennt man Programmieren.

Im nächsten Semester wirst du lernen, Dinge mit einem Programm zu steuern. Dabei lernst du die Grundlagen des Programmierens kennen und machst erste Schritte in der Elektrotechnik. Du wirst mit einem Arduino und elektrischen Bauteilen arbeiten. Was das genau ist, erfährst du auf den nächsten Seiten. Diese Art von Arbeit nennt man „Physical Computing“ und ist die ideale Kombination von Informatik und Technischem Gestalten, da du einerseits am Computer die Programme schreibst und andererseits im Werkraum die Teile zusammenbaust.

1.1 Arbeitsweise

- ⇒ Arbeite selbständig nach dem Handbuch
- ⇒ Alles Material das du benötigst, findest du in deiner Kiste
- ⇒ Verwende immer Computer und Kiste mit gleicher Nummer, welche dir zu Beginn zugeteilt wird
- ⇒ Speichere alle Dateien auf Educanet
- ⇒ Fotografiere und/oder filme die Schaltungen und speichere auch dies auf Educanet
- ⇒ Trage Sorge zum Material!

1.2 Was ist ein Mikrocontroller?



Ein Mikrocontroller ist ein kleiner Computer, der fast alles beinhaltet, was einen richtigen Computer ausmacht: Recheneinheit, Speicher, Ein- und Ausgabeschnittstellen, Uhr und vieles mehr, nur einfach ohne Gehäuse, Bildschirm, Tastatur und Maus. Auf dem Mikrocontroller werden Programme und Daten gespeichert, die dieser wie ein App auf einem Handy ausführen kann. Die Programme werden auf einem Computer, z.B. auf deinem Notebook erstellt und auf den Microcontroller kopiert.

**Ein oder mehrere Mikrocontroller befinden sich
in allen elektronisch gesteuerten Geräten.**

2 Arduino

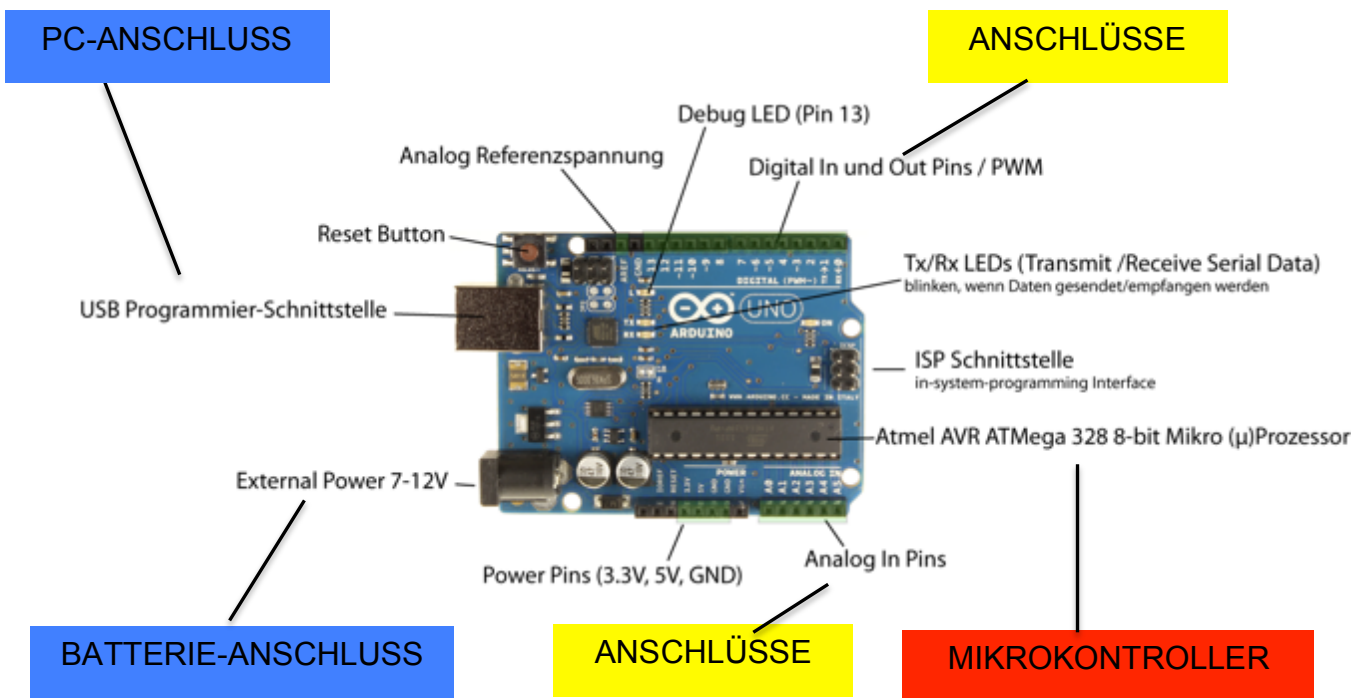
Ein Arduino ist eine Platine, auf dem sich ein Mikrocontroller befindet, aber kein Bildschirm, dafür kleine elektrische Bauteile und Anschlüsse. Die Bauteile sind bereits elektrisch miteinander verbunden, damit du einfacher arbeiten kannst.

An den Anschlüssen kannst du weitere elektrische Bauteile anschliessen, sogenannte Sensoren und Aktoren, welche der Microcontroller mit deinem Programm ansteuern kann.

2.1 Hardware

Als Hardware wird der Teil genannt, den du berühren kannst, also den eigentlichen Arduino.

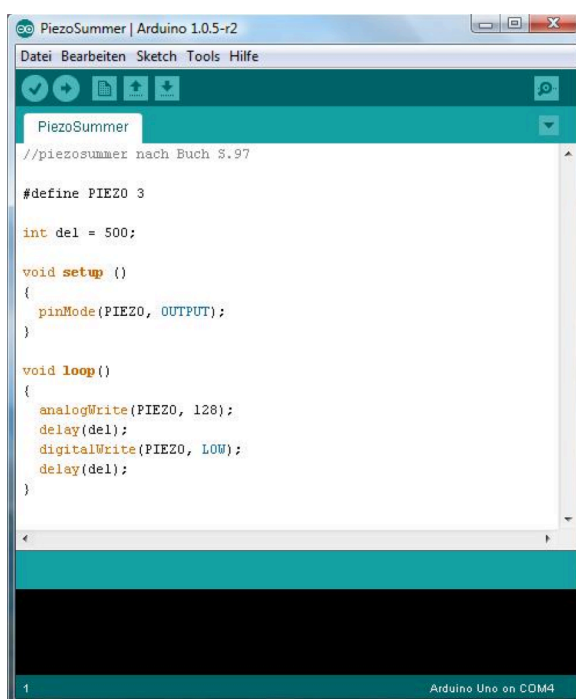
ARDUINO UNO



http://wiki.sgmk-ssam.ch/index.php?title=Kubulu_Arduino_Workshop






2.2 Software

Die Software steuert die Hardware. Mit Programmen wird der Hardware „gesagt“, was sie tun soll, dies erfolgt mit der **Programmierung**. Dazu nutzt man eine Programmiersprache. Beim Arduino erfolgt die Programmierung auf einem Computer oder Notebook in einer integrierten Entwicklungsumgebung, der sogenannten **IDE** (Integrated Development Environment). Ein Programm wird ein **Sketch** genannt.



1 Beispiel eines Sketches in der IDE

2.2.1 Symbole in der IDE

	Überprüfen	Hiermit kannst du deinen Sketch überprüfen.
	Upload	Bei diesem Symbol wird der Sketch auf den Arduino geladen.
	Neu	Beim Klicken auf dieses Symbol öffnet sich ein neues Eingabefeld.
	Öffnen	Hier kann ein abgespeicherter oder ein Beispielcode geöffnet werden.
	Speichern	Zum Speichern des Sketch auf dieses Symbol klicken.

3 Programmieren

Schliesst man die verschiedenen Bauteile an einen Arduino an, muss man dem Arduino noch sagen, was dieser mit den Bauteilen machen soll. Dies geschieht durch Programmieren. Aber der Computer versteht unsere Sprache nicht, dafür verwendet man eine spezielle Programmiersprache. Mit Programmiersprachen, die aus Buchstaben, Zeichen und Zahlen bestehen, kann man dem Computer dann „sagen“, was er tun soll.

Das Programm muss aber zuerst von einem Compiler, welcher in der IDE integriert ist, in die Maschinensprache übersetzt werden. Diese besteht nur noch aus Nullen und Einsen.

Beim Arduino wird die Programmiersprache C++ (ausgesprochen "C plus plus") verwendet.

Programmieren = Das Erstellen von Computerprogrammen mit speziellen Programmiersprachen, z.B. C++, Python, Java, usw.

3.1 Programmieren mit Arduino

Programmiert wird in der **IDE**.

Ein Programm wird **Sketch** genannt.

Als Programmiersprache wird **C++** verwendet

3.2 Zeichen und Klammern

Jede Zeile wird mit einem Semikolon ; abgeschlossen.

Die Befehle in setup () und in loop() werden in eine geschweifte Blockklammer {} gesetzt. Geschweifte Klammern auf dem MAC: { = alt 8, } = alt 9.

auf MAC = alt 3.

[= alt 5,] = alt 6

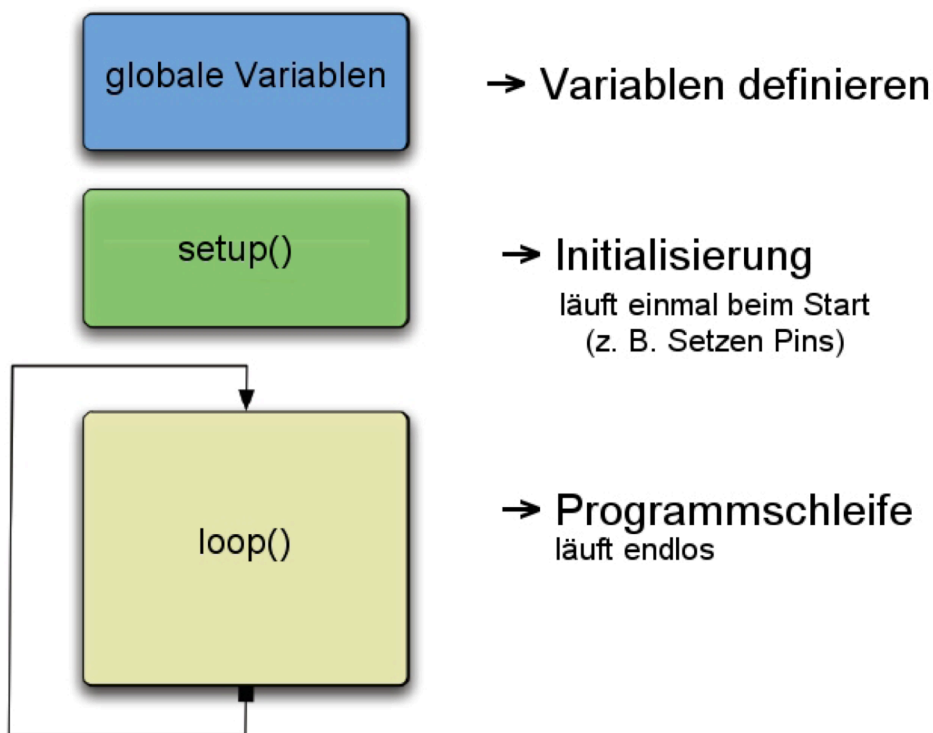
Mac = alt 3



1 Ausschnitt aus einer Programmierung

3.3 Aufbau eines Sketchs

Der Aufbau eines Sketchs ist immer gleichartig und gliedert sich in drei Teile. Der erste Teil kann ausgelassen werden, der zweite, das `setup ()` muss aufgeführt werden, auch wenn es manchmal leer ist.



1 Aufbau Standard Sketch mit drei Blöcken

http://www.netzmafia.de/skripten/hardware/Arduino/arduino_ffg.pdf

Definitionsbereich/globale Variablen

Aktoren und Sensoren werden einem Pin/Anschluss zugewiesen.
Variablen werden gesetzt.

setup()

Hier werden die Grundeinstellungen vorgenommen. Z.B. ob ein Pin ein Ein- oder Ausgang ist (Input oder Output). Das `setup ()` wird einmal zu Beginn eines Programms ausgeführt.

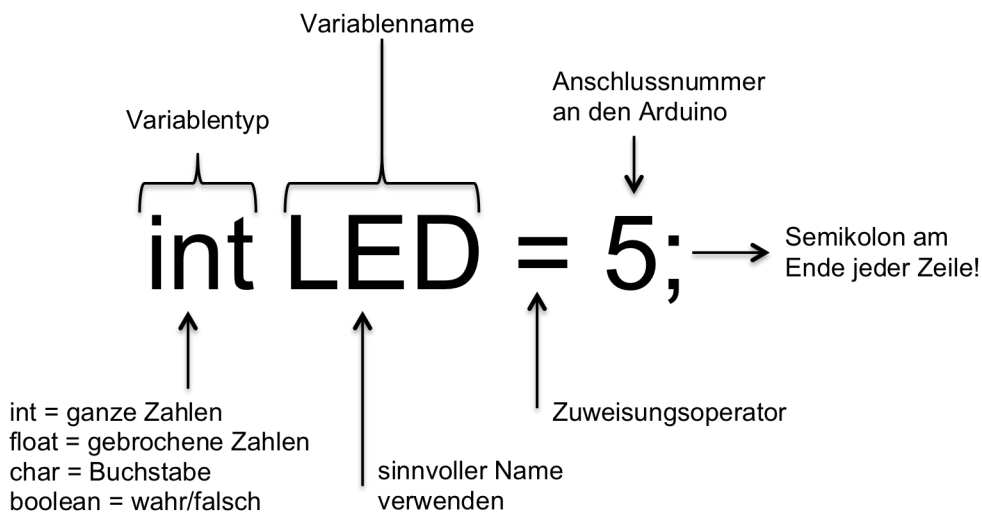
loop ()

Dies ist der eigentliche Programmablauf und wird immer wieder wiederholt.
Loop = eng. Schleife

3.3.1 Definitionsbereich: globale Variablen

Als erster Schritt beim Programmieren werden die Variablen definiert und festgelegt, welche Pins als Input oder Output verwendet werden. Das heisst, es wird festgelegt, was an den Arduino angeschlossen wird. Dabei wählt man je nach Art des elektronischen Bauteils einen anderen Variablentyp und gibt diesem einen Namen. Den Namen kannst du selber definieren, achte darauf, dass du dabei erkennst, um was es sich dabei handelt.

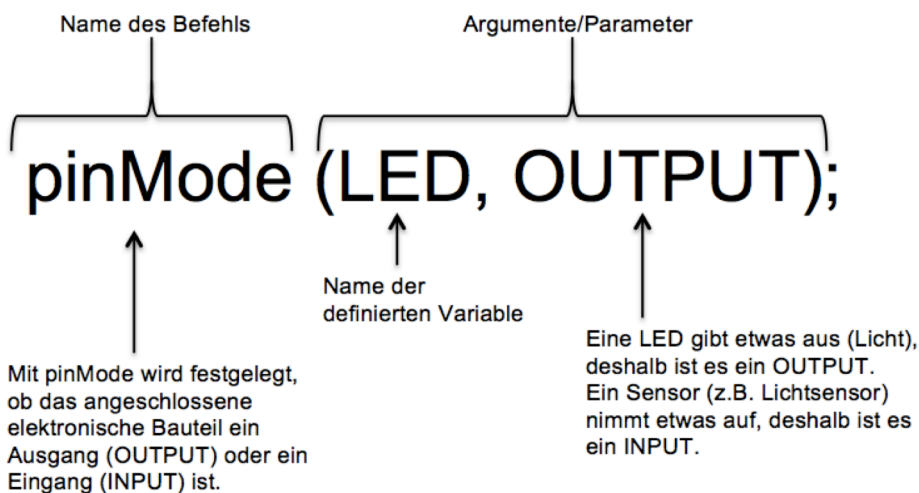
Weiter gibst du hier an, an welchen Anschluss des Arduino du das Bauteil ansteckst.



1 Beispiel einer globalen Variable mit einer LED

3.3.2 Initialisierung: void setup ()

Dieser Programmteil wird einmalig ausgeführt. Unter anderem wird dabei den Variablen eine Aufgabe zugewiesen.



2 Beispiel eines void setup () mit einer LED

3.3.3 void loop ()

Auf deutsch wird der loop () auch Schleife oder Endlos-Schleife genannt.

Im loop () wird gesagt, was die vorher definierten Teile immer wieder tun sollen. In einer Endlosschleife wird das Programm immer wieder ausgeführt, die Vorgänge wiederholen sich.

3.4 Sketch unter der Lupe

Sketch aus Übung 5.2

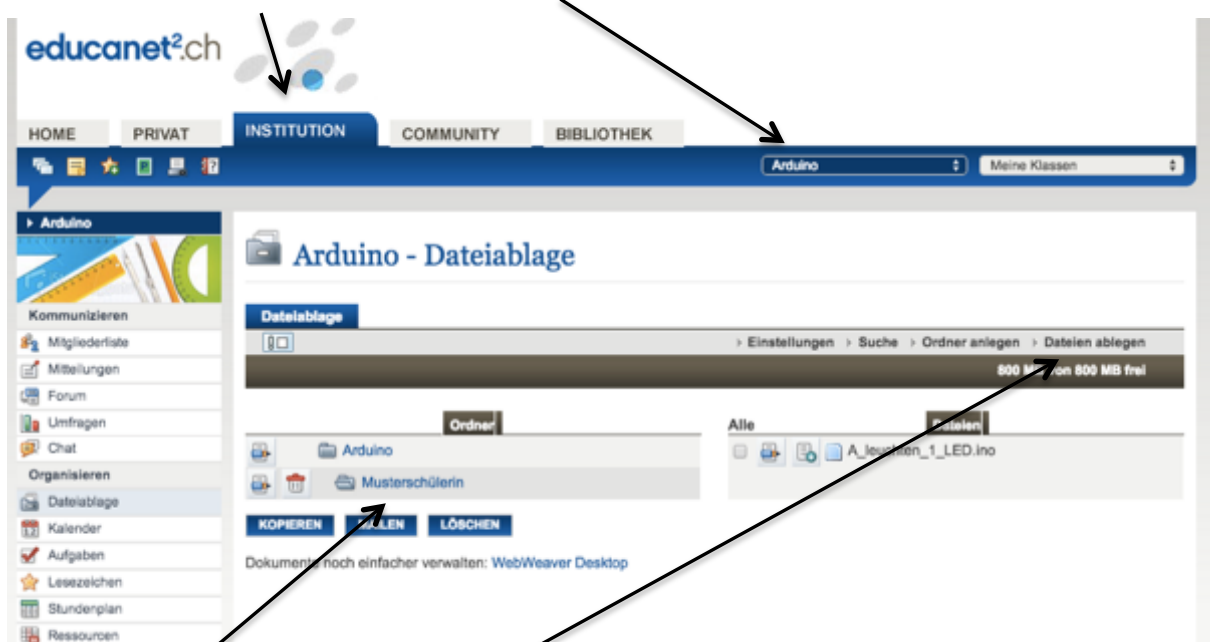
Titel	//1. LED anschliessen und leuchten lassen	Beschreiben, worum es bei diesem Sketch geht. Dies hat nichts mit dem Programm zu tun
	//	Nach einem doppelten Schrägstrich (engl. Slash) stehen Kommentare
	/* */	Kommentare über mehr als eine Zeile, werden zwischen Slash und Stern geschrieben
Definitionsbereich/ Globale Variablen	int LED = 5; //LED +Pol bei Pin 5 anschliessen	Dem Computer sagen, dass eine LED bei Pin 5 angeschlossen wird, damit er weiss, wo etwas passieren soll.
	;	Jede Eingabe wird mit einem Semikolon abgeschlossen
void setup()	void setup () { pinMode (LED, OUTPUT); //LED wird als Ausgabe definiert }	Definieren, was angeschlossen wird und was ausgegeben wird.
	{ }	Befehle, welche zusammen gehören, stehen zwischen einem geschweiften Klammerpaar.
void loop ()	void loop () { digitalWrite (LED, HIGH); delay(1000); digitalWrite (LED, LOW); delay(1000); }	Hauptteil Loop engl. = Schleife Was im Loop steht, wiederholt sich fortlaufend LED schaltet ein (HIGH) 1000 Millisekunden warten (delay) LED schaltet aus (LOW) 1000 Millisekunden warten (delay)

3.5 Sketch speichern

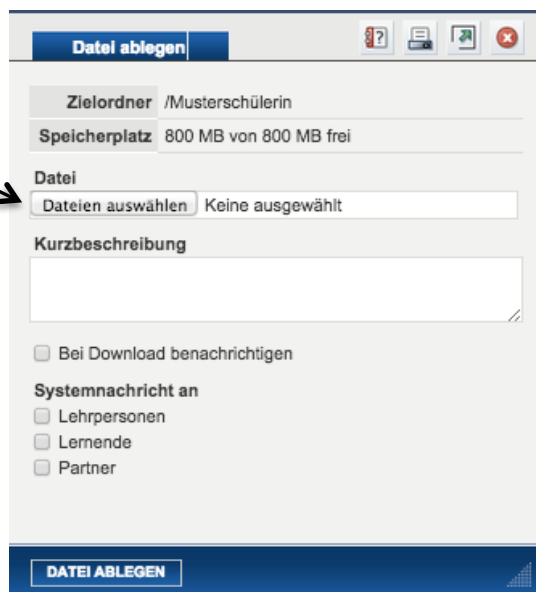
Alle Sketche werden auf www.educanet2.ch in der Gruppe „Arduino“ in einem Ordner mit deinem Namen gespeichert.

Vorgehen:

1. Speichere den Sketch mit dem vorgegebenen Titel und deinem Namen auf dem Desktop/Schreibtisch
2. Melde dich auf educanet2.ch an
3. Wähle unter Institution die Gruppe Arduino



4. Klicke deinen Ordner an
5. Wähle Dateien ablegen
6. Es öffnet sich ein Fenster
7. Klicke auf Dateien auswählen und in einem neuen Fenster kannst du deinen Sketch auswählen
8. Lade den Sketch hoch, in dem du auf DATEI ABLEGEN klickst



9. Lösche deinen Sketch auf dem Computer

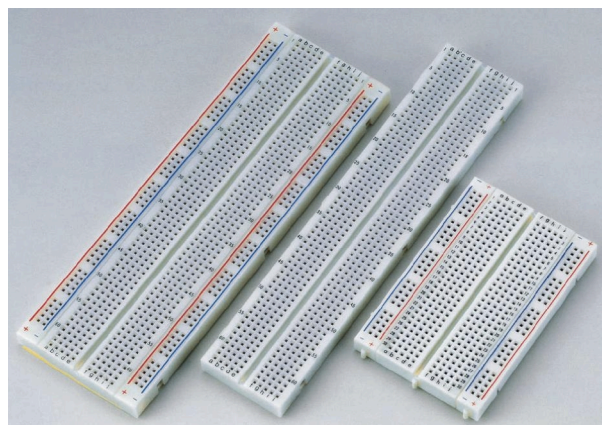
4 Bauteile 1

An den Arduino können elektronische Bauteile angeschlossen werden. Will man etwas austesten, so verwendet man eine Steckplatine, steckt dort die Teile ein und verbindet sie mit Kabel unter sich und mit dem Arduino.

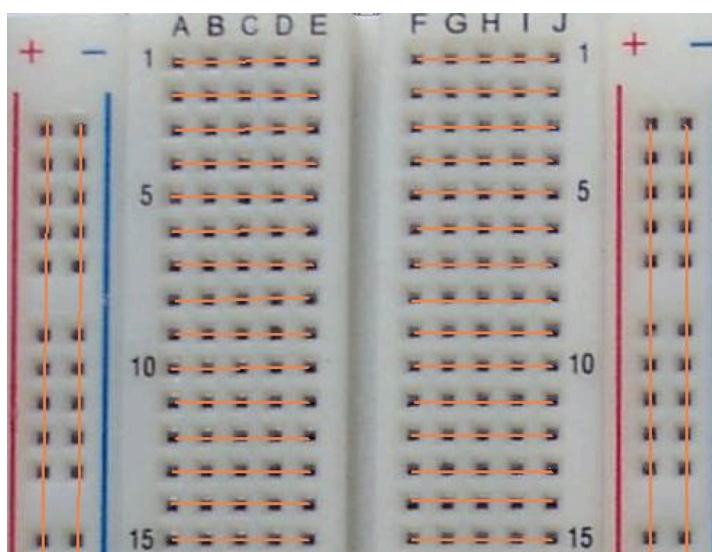
Als elektronische Bauteile werden Teile bezeichnet, die mit Strom betrieben werden können. Dabei unterscheidet man zwischen Aktoren und Sensoren. Aktoren geben etwas (Licht, Bewegung, Ton) aus, Sensoren nehmen etwas (Helligkeit, Bewegung, Temperatur) auf.

4.1 Steckplatine/Breadboard

Eine Steckplatine, engl. Breadboard, ist ein Kunststoffbrett durch welches Strom fließen kann. Darauf können elektronische Bauteile direkt oder mit einem Jumperkabel elektrisch verbunden werden ohne dass die Teile verlötet werden müssen. Die Löcher auf der Steckplatine sind in einer bestimmten Richtung miteinander verbunden. Studiere dazu die Abbildung:



1 Steckplatine in verschiedenen Grössen



2 So sind die Löcher miteinander verbunden.

4.2 Jumperkabel

Ein Jumperkabel ist ein Kabel mit einem „männlichen“ oder „weiblichen“ Stecker zur einfachen Verbindung von elektronischen Bauteilen. Jumperkabel gibt es in der Kombination m/m, f/f oder m/f.



1 männlein/male m



2 weiblein/female f

4.3 Verbindungskabel

Ähnlich wie die männlichen Jumperkabel, kann man die Verbindungskabel auf den Arduino oder die Steckplatine stecken. Sie eignen sich besonders für kurze Verbindungen auf der Steckplatine, da sie kürzer als die Jumperkabel sind.



3 Verbindungskabel
in verschiedenen

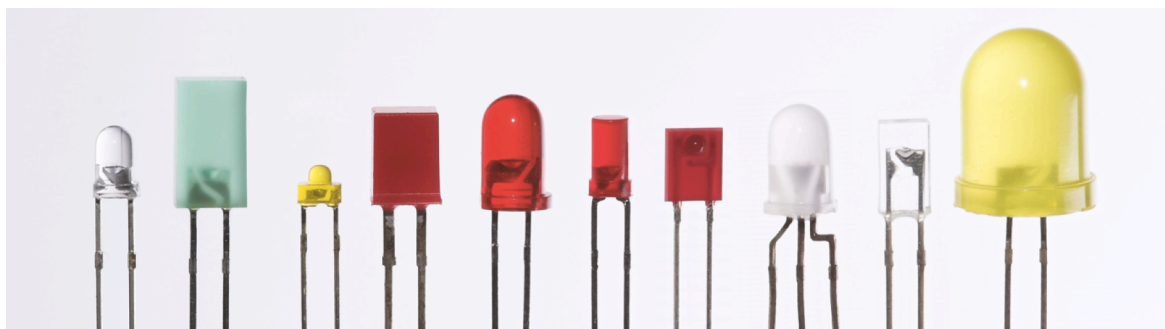
4.4 USB-Kabel

Um den Arduino mit dem Computer zu verbinden, braucht es ein USB-Kabel. Ein USB-Kabel mit einem A-Typ und B-Typ Stecker kann man auf der einen Seite an einem Computer und an der an einem Arduino anstecken.



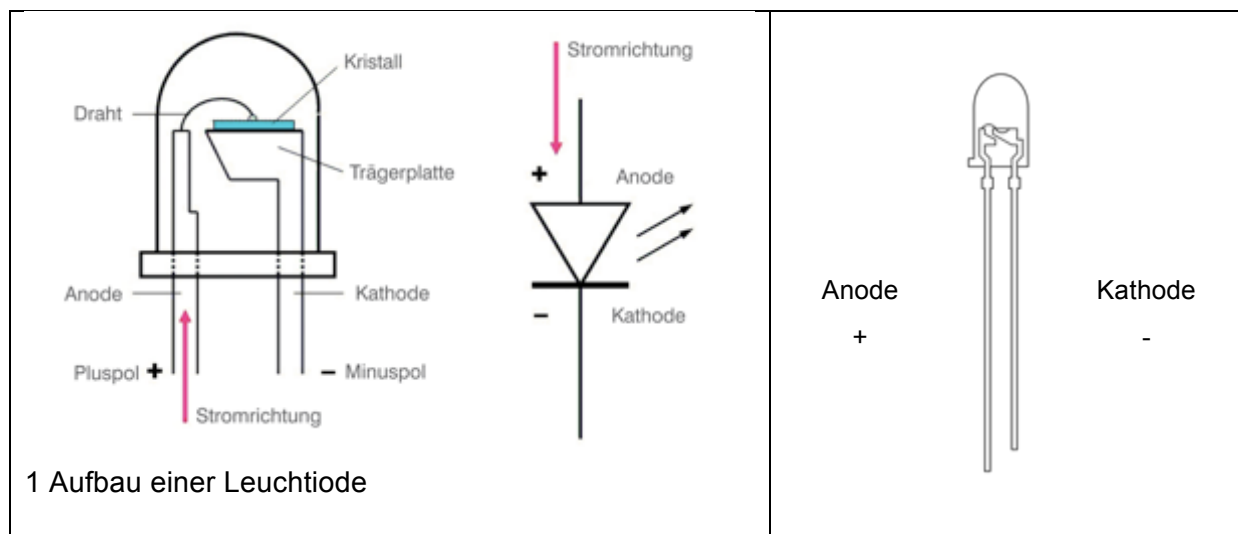
4 USB-Kabel 2.0 mit Stecker A und B

4.5 LED



4 Verschiedene LEDs

Leuchtdioden werden aus Halbleitermaterialien hergestellt, die eine Diode bilden. Eine Diode ist ein elektrisches Bauteil, das Strom nur in eine Richtung fließen lässt. Fließt ein Strom, so leuchtet die Leuchtdiode. Darum muss die Leuchtdiode richtig gepolt sein. Ist sie aber falsch gepolt, so leuchtet sie nicht, wird aber nicht zerstört.



Der Strom fließt vom Pluspol, der Anode, zum Minuspol, der Kathode. Werden die LEDs falsch angeschlossen, brennen sie nicht.

Willst du mehr über LEDs wissen, dann studiere die folgende Internetseite:

http://www.rc-electronic.com/downloads/pdf/bedienungsanleitungen/kompendium_leds_DE.pdf

(Du findest den Link auch auf Educanet unter Lesezeichen)

4.6 Widerstände

LEDs und andere elektronische Bauteile brauchen einen Vorwiderstand, damit nicht zu viel Strom fließt und die Teile nicht kaputt gehen.

Mit Widerständen lassen sich Spannungen und Stromstärken verändern und regulieren.

Anhand der Farbringe kannst du den Wert des Widerstandes bestimmen. Für die Berechnung des passenden

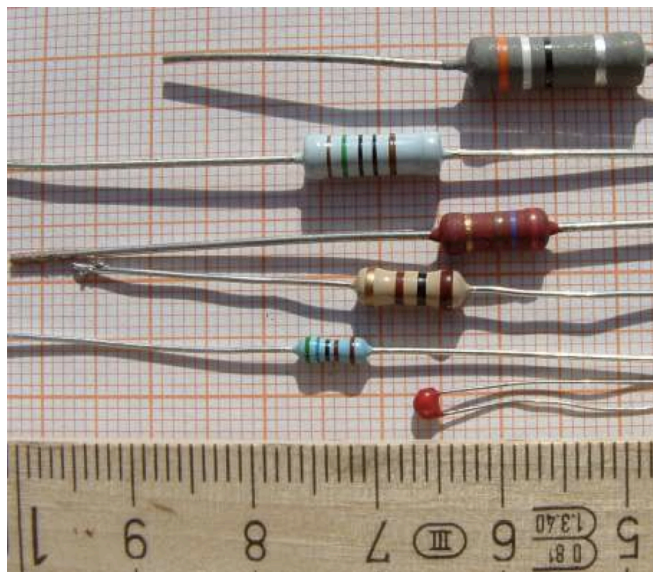
Widerstands verwendest du das Ohmsche Gesetz (NMM 7. Klasse). Es

gibt aber die Faustregel, dass eine LED mit einem Vorwiderstand von 150, 220, 270 oder 390 Ohm betrieben wird. Schliesse also eine LED nie ohne Vorwiderstand an.

Willst du mehr über Widerstände wissen, dann studiere die folgende Internetseite:

<http://www.kids-and-science.de/wie-funktioniert/detailansicht/datum/2009/12/04/was-ist-elektrischer-widerstand.html>

(Du findest den Link auch auf Educanet unter Lesezeichen)



2 Abbildung verschiedener Widerstände

5 Übung 1: LEDs leuchten und blinken lassen

Auf den nächsten Seiten lernst du, wie du zuerst eine LED, danach mehrere LEDs am Arduino anschliessen und sie in einem Programm steuern kannst.

Bei jeder Übung sagst du zuerst dem Computer, was der Arduino tun soll, in dem du ein Programm (Sketch) schreibst. Danach steckst du die Bauteile ineinander. Wenn du beides erledigt hast, kannst du den Sketch auf den Arduino laden.

Material für jede Übung:
Arduino
Steckplatine
USB-Kabel
Jumperkabel
Verbindungskabel
Das Material zu den Übungen steht jeweils im orangen Kasten

Ein Programm hat immer einen bestimmten zeitlichen Ablauf, den man auch Algorithmus nennt. Es handelt sich beim Algorithmus also um eine von dir verfasste Vorschrift, was der Microcontroller zeitlich hintereinander tun soll.

Abschreiben der Sketche

Bei den Übungen musst du die Sketche abschreiben. Bei den ersten beiden Übungen (LED leuchten lassen und LED blinken lassen), siehst du jeweils den vollständigen Sketch vor dir. Ab der dritten Übungen werden die Sketche in vier Teile (Titel, Definitionsbereich, void setup (), void loop ()) aufgeteilt und du schreibst die vier Teile nacheinander ab.

Wenn du die IDE öffnest, ist das void setup () und das void loop () bereits geschrieben. Bei den Übungen ist dies **Türkis** markiert und musst du nicht abschreiben.

Anschliessen des Arduino am Computer

Manchmal kann ein Sketch nicht hochgeladen werden, da der Arduino nicht erkannt wird. Führe folgendes aus:

WERKZEUGE → PORT → „/dev./cu.usbmodem1411 (Arduino Uno)“

5.1 1 LED leuchten lassen

Algorithmus: LED einschalten.

1 LED
150 Ω Widerstand

1. Öffne auf dem Computer die Arduino IDE, indem du auf das Symbol doppelklickst.
2. Speichere den Sketch auf dem Desktop/Schreibtisch mit deinem Namen und der Nummer 5.1. (Musterschülerin_5.1.)
3. Schreibe den Sketch ab und speichere ihn



```
//1. LED anschliessen und leuchten lassen

int LED = 5; //LED bei Pin 5 anschliessen

void setup () {
  pinMode (LED, OUTPUT); //LED wird als Ausgabe definiert
}

void loop () {
  digitalWrite (LED, HIGH); //HIGH = die Spannung ist eingeschaltet, die LED leuchtet
}
```

Achte darauf, dass du alle Zeichen machst und die Abstände einhältst!

In der IDE erscheint dein Text farbig.

Geschweifte Klammern auf dem MAC: { = alt 8, } = alt 9

A screenshot of the Arduino IDE interface. The window title is "A_leuchten_1_LED | Arduino 1.0.5". The code editor shows the same code as in the previous block, with syntax highlighting: comments in grey, keywords in blue, and function names in orange. The status bar at the bottom indicates "1" and "Arduino Uno on /dev/tty.usbmodem1411".

```
A_leuchten_1_LED


//1. LED anschliessen und leuchten lassen

int LED = 5; //LED bei Pin 5 anschliessen

void setup () {
  pinMode (LED, OUTPUT); //LED wird als Ausgabe definiert
}

void loop () {
  digitalWrite (LED, HIGH); //HIGH = etwas wird ausgegeben
}
```

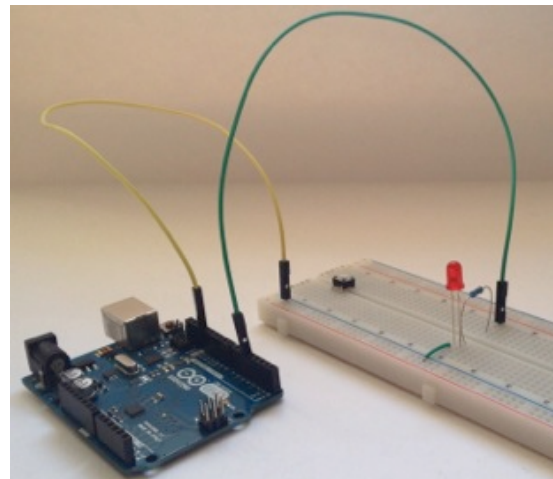
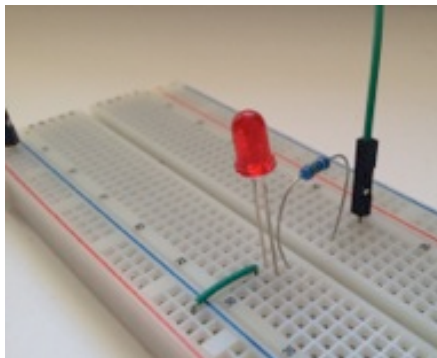
So sieht der Sketch in der IDE aus.

4. Überprüfe deinen Sketch in dem du auf  klickst.

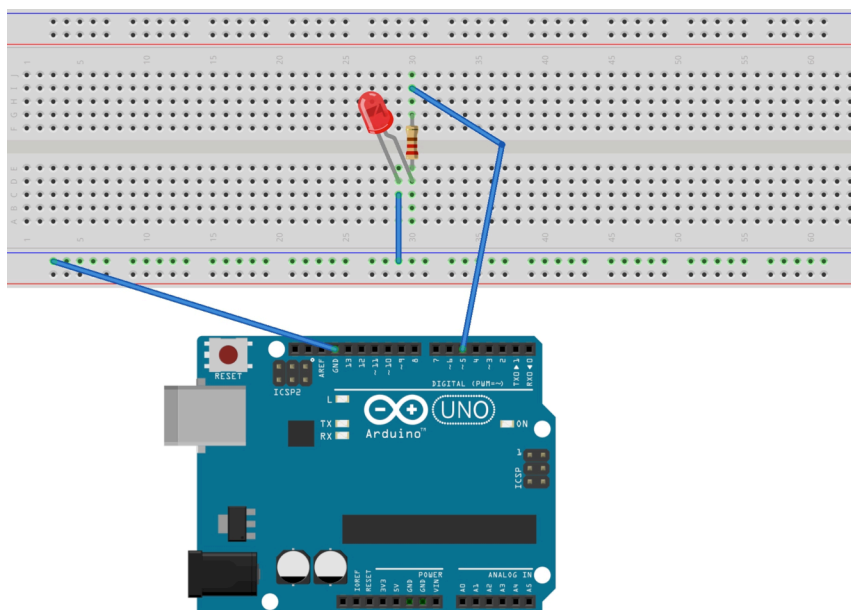
```
Kompilierung abgeschlossen.  
  
Binäre Sketchgröße: 872 Bytes (von einem Maximum von 32.256 Bytes)  
1 Arduino Uno on /dev/tty.usbmodem1411
```

Erscheint dies im schwarzen Balken, ist dein Code korrekt. Fahre bei 5. mit dem Aufbau der Schaltung weiter. Erscheint etwas anderes, musst du den Fehler suchen. Speichere den Sketch.

5. Wähle eine LED aus und stecke sie auf die Steckplatine
Die Anode (+, langes Ende) soll dabei im zweit hintersten Loch der Linie 30 stecken, die Kathode (-) in der Linie 29.
6. Nimm einen 390 Ohm Widerstand. Stecke ein Ende des Widerstands ins hinterste Loch der Linie 30 und das andere Ende auf die andere Seite des Spaltes.
7. Stecke ein kurzes Kabel von der Linie 29 aufs – (GND)
8. Stecke ein Kabel vom – auf GND (Ground) des Arduino
9. Stecke ein Kabel vom Ende des Widerstands auf den Pin 5 des Arduino
- So sollte das aussehen:




Vielleicht erkennst du den Aufbau der Schaltung besser auf dem fotografischen Schaltplan:



fritzing

10. Verbinde den Arduino mit dem Computer via USB-Kabel.

11. Lade den Sketch hoch in dem du auf  klickst.

Oft erkennt der Computer den Arduino nicht auf Anhieb. Dann musst du den Arduino auswählen wie auf Seite 16 unter „Anschliessen des Arduino an den Computer“ beschrieben.

12. Deine LED leuchtet: Aufgabe erfüllt.

LED leuchtet nicht: Ist alles korrekt zusammengeschlossen? Stimmt der Sketch? Suche den Fehler bis deine LED brennt.

13. Speichere den Sketch in deinem Educanet-Ordner und lösche die Datei auf dem Computer¹

Die LED leuchtet. Das ist aber noch keine grosse Sache, das kann man auch ohne Arduino und Computer. Leuchtet auf der Stereoanlage ein Lämpchen, sobald sie eingeschaltet wird, funktioniert dies genau so.

5.2 1 LED blinken lassen

Algorithmus: Eine LED einschalten, LED 1 Sekunde leuchten lassen, LED ausschalten und dies immer wieder wiederholen.

Ohne Arduino und Computer eine LED blinken zu lassen wird schon schwieriger, aber hier geht es ganz einfach. Dafür musst du an der Steckplatine und am Arduino nichts umstecken, sondern nur den *loop ()* im Sketch umschreiben. Folge dafür den folgenden Schritten:



1. Löse das USB-Kabel vom Computer.
2. Öffne den Sketch 5.1.
3. Speichere den Sketch neu mit deinem Namen und der Nummer 5.2 auf den Schreibtisch:
Datei → Speichern unter... → Name und Nummer eingeben
4. Ändere den Titel und den loop (). (Der grau markierte Teil steht bereits in deinem Sketch und musst du nicht abschreiben)

```
/*LED blinken
1. LED anschliessen und blinken lassen
2. Blinkgeschwindigkeit ändern
*/

int LED = 5; //LED bei Pin 5 anschliessen

void setup () {
  pinMode (LED, OUTPUT);
}

void loop () {
  digitalWrite (LED, HIGH); //HIGH = Spannung eingeschaltet, die LED leuchtet
  delay(1000); // 1000 Millisekunden warten
  digitalWrite (LED, LOW); //LOW = Spannung ausgeschaltet, die LED erlischt
  delay(1000); // 1000 Millisekunden warten
}
```

5. Überprüfe deinen Sketch mit 
6. Ist der Sketch korrekt, verbinde den Arduino mit dem Computer und lade den Sketch hoch mit 
7. LED blinkt.
8. Speichere den Sketch in deinem Educanet-Ordner und lösche die Datei auf dem Computer


5.2.1 Zusatz 1 : Blinkgeschwindigkeit ändern

```
delay(1000); //1000 Millisekunden
```

delay engl. = Verzögerung/Wartezeit

Bei einem *delay* () macht das Programm eine Pause. In Klammer wird in Millisekunden angegeben, wie lange diese Pause sein soll. 1000 Millisekunden (ms) sind 1 Sekunde.

Indem du die Zahl in Klammer änderst, ändert sich die Leuchtzeit, resp. die Zeit in der die LED nicht leuchtet.

1. Öffne den Sketch 5.2 und speichere ihn unter 5.2.1
2. Ersetze bei void loop () die Zahlen in den Klammern. Z.B. *delay(1000)* = *delay(300)*
3. Der Aufbau der Schaltung ist immer noch der Selbe wie bei 5.1. Musst du die Schaltung neu aufbauen, schaue dort nach.
4. Lade den Sketch auf den Arduino (Arduino mit Computer verbinden, danach auf  klicken)
5. Beobachte die LED. Was stellst du fest?
6. Probiere andere Zahlen aus. Setze nicht an beiden Stellen die gleiche Zahl ein. Was kannst du beobachten?
7. Speichere den Sketch in deinem Educanet-Ordner und lösche die Datei auf dem Computer

5.2.2 Zusatz 2: Blinkgeschwindigkeit mit einer Variable definieren

Du kannst für die Blinkgeschwindigkeit auch eine Variable einsetzen, damit du nur einmal das *delay()* ändern musst und es überall ausgeführt wird.

1. Öffne den Sketch 5.2 und speichere ihn unter 5.2.2
2. Die Blinkgeschwindigkeit soll mit einer Variablen definiert werden.
Dazu musst du dem Programm sagen, dass er die Variable *d* verwenden soll.
Das heisst, im Definitionsbereich wird der Typ und der Wert der Variablen festgelegt:

```
int d= 500;
```

int sagt, dass es sich um eine ganze Zahl (einen Integer) handelt. d ist der Name der Variable. Du könntest auch einen beliebigen anderen Namen eingeben, wichtig ist, dass du später immer den gleichen Namen verwendest. Die 500 steht für die Dauer des *delays*. Du könntest auch eine andere Zahl einsetzen.

Im *loop ()* musst du überall die Variable einsetzen, wo sie gebraucht wird, das heisst, beim *delay*: `delay(d)`;

Im Sketch musst du ändern, was rot markiert ist:

```
/*LED blinken
1. LED anschliessen und blinken lassen
2. Blinkgeschwindigkeit ändern
*/

int LED = 5;
int d = 500; //d ist die Variable, 500 die dafür eingesetzte Zeit

void setup () {
  pinMode (LED, OUTPUT);
}

void loop () {
  digitalWrite (LED, HIGH); //HIGH = Spannung eingeschaltet
  delay(d); //die LED leuchtet 1000 Millisekunden
  digitalWrite (LED, LOW); //LOW = Spannung ausgeschaltet
  delay(d); //die LED leuchtet nicht während 500 Millisekunden
}
```

- Überprüfe deinen Sketch und lade ihn anschliessend hoch.
- Setze verschiedene Zahlen als Variable ein und teste es.

```
int d = 500;
```

- Willst du nicht beide *delay* gleich lang, kannst du zwei Variablen einsetzen. Definiere dazu eine weitere Variable (z.B. `int d2 = 100;`) und schreibe die Variable dort in den *loop ()* wo du sie haben willst.
- Teste deinen Sketch.
- Speichere den Sketch in deinem Educanet-Ordner und lösche die Datei auf dem Computer

5.3 5 LEDs blinken lassen

Algorithmus: LED1 einschalten, leuchten lassen, ausschalten.
LED2 einschalten, leuchten lassen, ausschalten.
LED3 einschalten, leuchten lassen, ausschalten.
LED4 einschalten, leuchten lassen, ausschalten.
LED5 einschalten, leuchten lassen, ausschalten.

1. Öffne den Sketch 5.2 und speichere ihn unter 5.3 und deinem Namen
2. Ändere den Titel:

```
/* 5 LEDs blinken  
1. Sketch anpassen  
2. LEDs anschliessen  
3. Sketch uploaden  
*/
```

5 LEDs
5x passender Widerstand

3. Definitionsbereich

Da mehrere LEDs angeschlossen werden, müssen wir diese Nummerieren (man könnte ihnen auch einen unterschiedlichen Namen geben).

Ändere LED in LED1.

Füge die 2. LED hinzu:

```
int LED2 = 6;
```

Fahre mit den weiteren LEDs wie bei LED2 weiter. Definiere LED 3-5 auf die Anschlüsse 7-9.

4. void setup ()

Ändere LED in LED1

Gib die Funktion der 2. LED an:

```
pinMode (LED2, OUTPUT);
```

Fahre mit den weiteren LEDs wie bei LED2 weiter.

5. void loop ()

Ändere LED in LED1 (2x ändern)

Füge die 2. LED in die Schleife ein:

```
digitalWrite (LED2, HIGH); //LED2 wird eingeschaltet  
delay(1000); //LED2 leuchtet 1000ms  
digitalWrite (LED2, LOW); //LED2 wird ausgeschaltet  
delay(1000); //1000ms LED leuchtet nicht
```

Fahre bei den anderen LEDs wie bei LED2 weiter

6. Prüfe deinen Sketch. Stimmt dein Sketch?

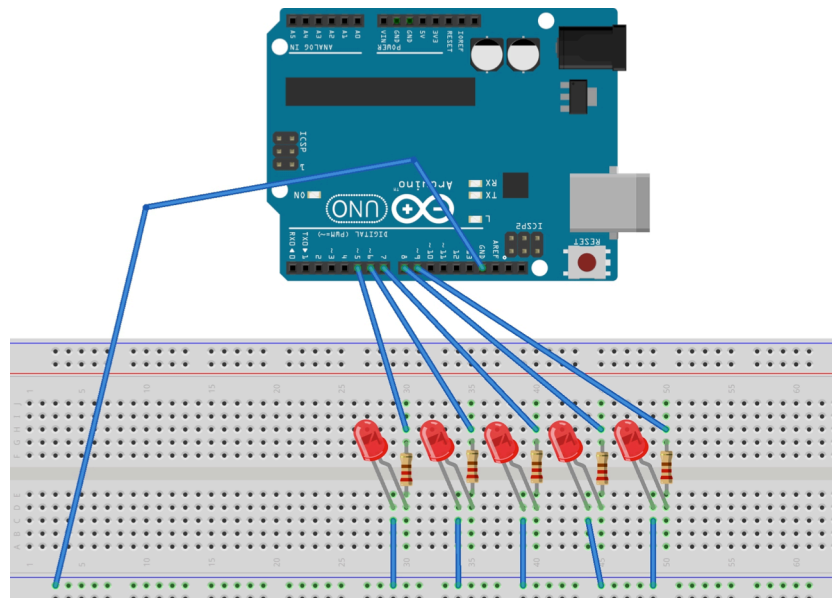
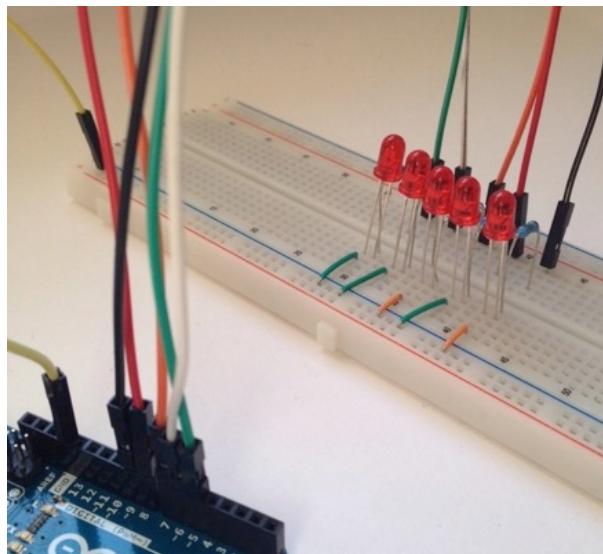
Ja = weiter bei Punkt 7. Nein = Suche den Fehler

7. Baue den Schaltkreis auf:

Stecke die LEDs 2-5 wie LED1 (Übung 5.1) mit dem Widerstand auf die Steckplatine. Wo genau du die LED auf die Steckplatine steckst, kannst du frei wählen. Wichtig ist, dass nach der Anode (+) ein Widerstand einbaust.

Verbinde das – der LEDs mit dem – der Steckplatine. Verbinde die LEDs nach dem Widerstand mit den Anschlüssen des Arduino.

Achtung! Beachte die Reihenfolge. Verwende die Anschlüsse 5-9.



8. Verbinde den Arduino mit dem Computer.

9. Lade den Sketch hoch und Teste deine Lichterkette.

10. Speichere den Sketch in deinem Educanet-Ordner und lösche die Datei auf dem Computer

5.3.1 Zusatz 1: Blinkreihenfolge und Blinkgeschwindigkeit ändern

1. Öffne den Sketch 5.3 und speichere ihn unter 5.3.1
2. Programmiere deine eigene Blinkreihenfolge in dem du den loop () änderst.

Wie bei 5.2.1 oder 5.2.2 kannst du auch die Blinkgeschwindigkeit ändern.

Kannst du die Lichter von Knight Rider nachmachen?

Beispiel einer Blinkabfolge:

```
void loop () {  
  digitalWrite (LED1, HIGH);  
  delay(2000);  
  digitalWrite (LED1, LOW);  
  delay(300);  
  digitalWrite (LED4, HIGH);  
  delay(300);  
  digitalWrite (LED4, LOW);  
  delay(600);  
  digitalWrite (LED3, HIGH);  
  delay(1000);  
  digitalWrite (LED3, LOW);  
  delay(500);  
  digitalWrite (LED5, HIGH);  
  delay(1200);  
  digitalWrite (LED5, LOW);  
  delay(1000);  
  digitalWrite (LED2, HIGH);  
  delay(200);  
  digitalWrite (LED2, LOW);  
  delay(1000);  
}
```

3. Teste deinen Sketch.
4. Der Aufbau der Schaltung ist wie bei 5.3
5. Lade den Sketch hoch, nachdem du die Schaltung aufgebaut hast.
6. Wähle eine Blinkreihenfolge aus, die du der Klasse präsentieren willst.
Melde dies der Lehrperson.
7. Speichere den Sketch in deinem Educanet-Ordner und lösche die Datei auf dem Computer

6 Bauteile 2

6.1 Schalter

Schalter schalten einen Stromkreis ein oder aus. Sie haben nach der Betätigung einen stabilen Zustand (im Gegensatz zu einem Taster).

Es gibt normale Ein-Aus-Schalter, Umschalter, Drehschalter u.v.m.



1 Verschiedene Schalter

<http://www.elovbecker.com/52.0.html?&L=1>

6.2 Taster

Taster schliessen einen Stromkreis, solange sie gedrückt werden. Lässt man sie los, öffnet sich der Stromkreis wieder.



2 Verschiedene Taster

http://www.nh-technology.de/de_beleuchtete_taster.php

7 Übung 2: LED mit einem Taster einschalten

Algorithmus: Taster drücken, LED leuchtet 1s (1000ms) und löscht dann wieder

1 LED
1 Taster

1. Öffne einen neuen Sketch und speichere ihn mit deinem Namen und der Nummer 7 auf dem Desktop/Schreibtisch.
2. Gib im Sketch zuerst den Titel ein:

```
/*  
LED mit Taster einschalten  
*/
```

3. Definitionsbereich

Definieren von LED, Taster und Tasterstatus. Genau wie die LED wird auch der Taster definiert und auf einen Pin festgelegt. Ebenfalls muss der Tasterstatus definiert werden, um später dem Taster zwei Positionen (on/off) zuzuweisen kann.

```
int LED = 6; //Die LED wird an Pin 6 angeschlossen (sie erhält deren Wert)  
int taster = 7; //Der Taster wird an Pin 7 angeschlossen (er erhält deren Wert)  
int tasterstatus = 0; //Unter diesem Wert wird später gespeichert, ob der  
Taster aedrückt ist oder nicht.
```

4. void setup ()

(Dem Programm sagen, welche Funktion die Bauteile haben)

Die LED ist ein Output, etwas (Licht) wird ausgegeben.

Der Taster ein INPUT, etwas (Druck) wird aufgenommen.

Mit dem digitalWrite wird ein PULL-UP-Widerstand eingeschaltet. Dieser Widerstand macht, dass der Strom ganz oder gar nicht fließt.

```
void setup ()  
{  
  pinMode (LED, OUTPUT); //LED gibt etwas aus (Ausgang)  
  pinMode (taster, INPUT); //Taster nimmt etwas auf (Eingang)  
  digitalWrite(taster, HIGH); //schaltet den PULL-UP-Widerstand ein  
}
```

5. void loop ()

(Dem Programm sagen, was ausgeführt werden soll.)

Der Taster hat zwei Positionen: gedrückt oder nicht gedrückt.

Das heisst, du musst dem Programm für beide Positionen sagen, was es tun soll. Dazu brauchst du den Befehl if/else (wenn/sonst).

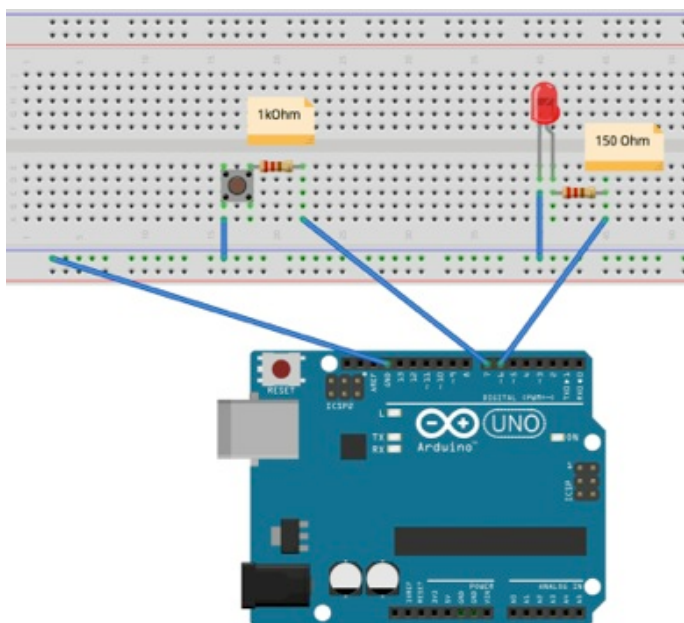
Das Programm soll Folgendes tun:

Wenn der Taster nicht gedrückt ist, leuchtet die LED nicht. → if

Wenn der Taster gedrückt ist (sonst) soll die LED brennen. → else

```
void loop ()  
{  
  tasterstatus = digitalRead(taster); //Hier wird der Taster ausgelesen. Das  
  //Ergebnis wird unter der Variable „tasterstatus“ mit dem Wert „HIGH“ für „an“  
  //oder „LOW“ für „aus“ gespeichert.  
  
  if (tasterstatus == HIGH) //Taster ist nicht gedrückt  
  {  
    digitalWrite (LED, LOW); //Taster nicht gedrückt, LED brennt nicht  
  }  
  
  else  
  {  
    digitalWrite (LED, HIGH); //Wenn der Taster gedrückt ist, brennt die LED  
    delay (1000); //LED leuchtet 1000ms  
    digitalWrite(LED, LOW); //LED löscht wieder aus  
  }  
}
```

- Überprüfe und speichere deinen Sketch.
- Baue mit Hilfe des fotografischen Schaltplans die Schaltung zusammen:



- Verbinde den Arduino mit dem Computer und lade den Sketch hoch
- Funktioniert alles? Bravo. Und sonst such den Fehler
- Wenn du willst, kannst du verschiedene *delays* eingeben. Das heisst das delay ändern wie bei 5.2.1
- Speichere den Sketch in deinem Educanet-Ordner und lösche die Datei auf dem Computer

8 Bauteile 3

8.1 Lautsprecher

Ein Lautsprecher ist ein Gerät, das elektrische Schwingungen in Töne umwandelt und diese wiedergibt. Dabei wird eine Membran im Rhythmus des Tons hin und her bewegt. Je nach Leistung, Grösse und Qualität ist die Klangqualität besser.

Wenn du mehr über Lautsprecher wissen willst, schaue hier:

<http://www.elektronik-kompodium.de/sites/bau/1207071.htm>



1 Verschiedene Lautsprecher
<https://www.etsy.com/de/listing/170844150/lautsprecher-laptop-lautsprecher-art>

8.2 Piezo-Summer/Buzzer

Ein Piezo-Summer oder Buzzer kann auch Töne wiedergeben und wird oft als Piepser verwendet und nicht um Melodien wiederzugeben.




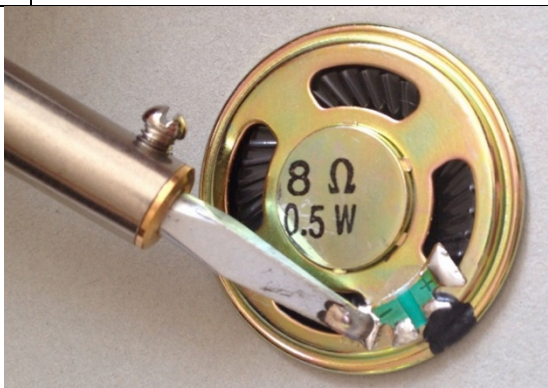
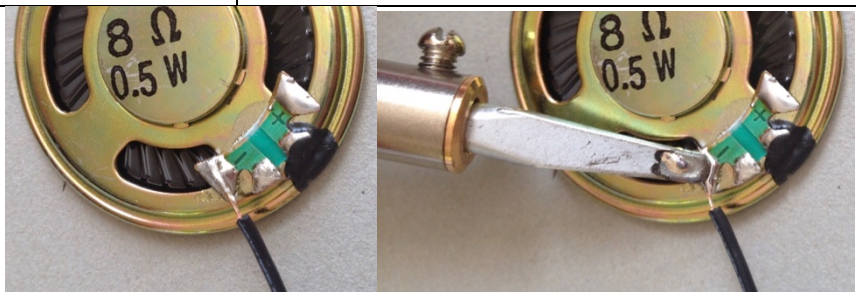
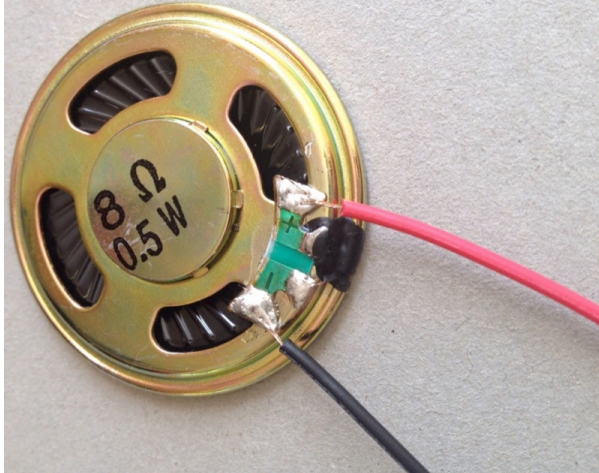
2 Verschiedene Piezosummer
<http://www.ecvv.com/product/102442.html>

8.3 Lautsprecher vorbereiten

Damit du den Lautsprecher an den Arduino anschliessen kannst, musst du zuerst die Kabel anlöten.

(In einigen Boxen ist der Lautsprecher bereits verkabelt und du kannst diesen Schritt überspringen)

- | |
|---|
| <p>Material</p> <ul style="list-style-type: none"> - 1 Lautsprecher - 1 Jumperkabel rot - 1 Jumperkabel schwarz - Lötstation - Lötzinn - Abisolierzange |
|---|

1	Einschalten der Lötstation (300°)	
2	Kabelenden abisolieren (0.5cm)	
3	Lötstelle auf dem Lautsprecher erhitzen und Lötzinn darauf geben	
4	Abisoliertes Kabelende auf die Lötstelle legen und mit dem LötKolben erhitzen bis das Kabel hineinschmilzt.	
5	Mit dem zweiten Kabel ebenso verfahren	
6	So sollte es am Ende aussehen	

9 Übung 3: Töne erzeugen

9.1 Piezosummer summen lassen

1 Piezosummer

Algorithmus: Piezosummer ertönt während 500ms. Pause während 500ms.

1. Öffne einen neuen Sketch und speichere ihn mit deinem Namen und der Nummer 9.1 auf dem Desktop/Schreibtisch.
2. Gib im Sketch zuerst den Titel ein:

```
/*  
  Piepsen mit Piezosummer  
*/
```

3. Definitionsbereich

Den Piezosummer definieren

```
int PIEZO = 3; //Der Summer wird an Pin 3 gesteckt  
int d = 500; //Mit der Variabel d den Abstand zwischen den Piepstönen definieren
```

4. void setup ()

(Dem Programm sagen, welche Funktion die Bauteile haben)

```
void setup ()  
{  
  pinMode(PIEZO, OUTPUT); //Den Piezosummer als Ausgabe definieren  
}
```

5. void loop ()

Der loop macht, dass der Piezosummer immer wieder piepst.

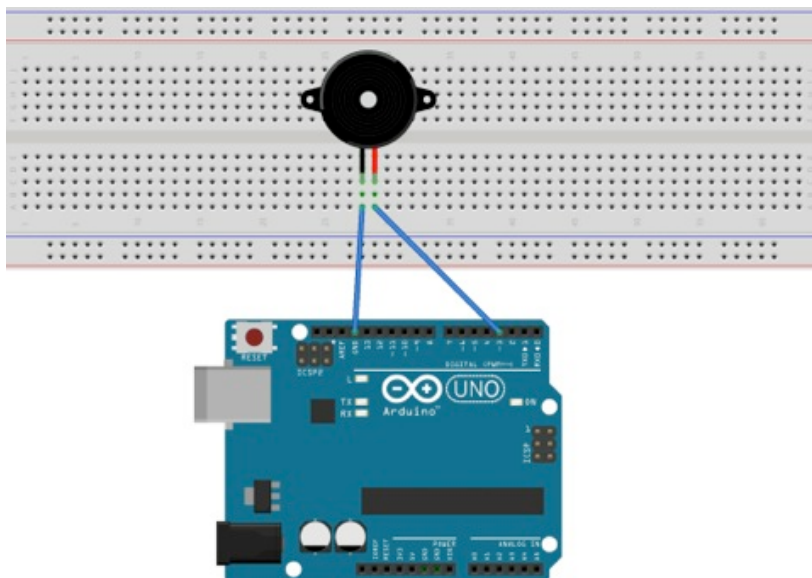
Bisher verwendeten wir im loop () den Befehl *digitalWrite*. Dieser Befehl hat nur zwei Werte: 0 oder 1, INPUT oder OUTPUT, HIGH oder LOW.

Ein Ton hat verschiedene Tonhöhen, so dass wir den Befehl *analogWrite* brauchen. Bei diesem Befehl wird eine Wechselspannung erzeugt. Je nach übergebener Zahl zwischen 0 und 255 ertönt ein anderer Ton.

Soll der Piezosummer nicht tönen, wird der Befehl *digitalWrite* verwendet, da es nur um an oder aus geht (HIGH/LOW).

```
void loop()  
{  
  analogWrite(PIEZO, 128); //Zahl zwischen 0 und 1023  
  delay(d); //Summer tönt während 500ms  
  digitalWrite(PIEZO, LOW);  
  delay(d); //Summer ertönt nicht während 500ms  
}
```

- Überprüfe den Sketch und speichere ihn
- Der Aufbau der Schaltung ist ganz einfach, da nur der Piezosummer mit dem Arduino verbunden werden muss:



- Verbinde den Arduino mit dem Computer und lade den Sketch hoch
- Piepstst? JA = weiter, NEIN = Fehler beheben
- Speichere den Sketch in deinem Educanet-Ordner und lösche die Datei auf dem Computer

9.1.1 Zusatz: Ton und Rhythmus verändern

- Ton ändern: In dem du die rot markierte Zahl veränderst, ändert sich das Piepsen. Probiere aus! (Veränderter Sketch hochladen)

```
void loop()
{
  analogWrite(PIEZO, 128); //Zahl zwischen 0 und 1023
  delay(d); //Summer tönt während 500ms
  digitalWrite(PIEZO, LOW);
  delay(d); //Summer ertönt nicht während 500ms
}
```

Was stellst du fest? Wie verändert sich der Ton, wenn du eine tiefere Zahl eingibst? Wie verändert sich der Ton, wenn du eine höhere Zahl eingibst?

- Rhythmus ändern: In dem du die gelb markierte Zahl veränderst, ändert sich der Abstand zwischen dem Piepsen. Probiere aus!

```
int PIEZO = 3; //Der Summer wird an Pin 3 gesteckt
int d = 500; //Mit der Variabel d den Abstand zwischen den Piepstönen definieren
```

Du kannst das *delay* auch direkt im *loop ()* festlegen (Zahl anstelle der Variable in Klammer).

9.2 Melodie aus Lautsprecher

1 Lautsprecher

Bevor du mit dieser Übung beginnen kannst, musst du die Lautsprecher vorbereiten. Die Anleitung dazu findest du unter Punkt 8.3 auf Seite 31.

Algorithmus: Lautsprecher spielt während 100ms einen Ton, Pause.

Nächster Ton während 100ms abspielen, Pause.

7x wiederholen.

1. Öffne einen neuen Sketch und speichere ihn mit deinem Namen und der Nummer 9.2 auf dem Desktop/Schreibtisch.
2. Gib im Sketch zuerst den Titel ein:

```
/*  
Melodie mit Lautsprecher  
*/
```

3. Definitionsbereich

Bibliothek hinzufügen und Lautsprecher definieren

In einer Bibliothek (engl. Library) werden Befehle und Variablentypen vorgespeichert. Im Variablentyp *Tone*, den du hier verwendest, wurden alle Töne vorprogrammiert, so dass nur noch der Ton angegeben werden muss und nicht die Frequenz des Tones. Die Bibliothek wurde bereits auf deinem Computer installiert, so dass du sie nur noch einfügen musst.

Zudem musst du dem Programm sagen, dass eine Variable *speaker* vom Typ *Tone* verwendest.

Bibliothek einfügen: SKETCH → INCLUDE LIBRARY → TONE

Die erste Zeile im Kästchen erscheint in der IDE. Schreibe die zweite Zeile ab.

```
#include <Tone.h> //Bibliothek(Library) Tone hinzufügen  
Tone speaker; //Den Lautsprecher als Speaker festlegen
```

4. void setup ()

Da bisher der Pin noch nicht festgelegt wurde, wird dies im setup () gemacht.

Der Lautsprecher wird auf Pin 12 gesetzt.

```
void setup ()  
{  
  speaker.begin(12); //Den Lautsprecher auf Pin 12 definieren  
}
```

5. void loop ()

Die Melodie wird immer wieder abgespielt. (Soll die Melodie nur einmal ertönen, play()-Befehl ins setup () schreiben)

Zum Abspielen eines Tons verwendet man den Befehl *play*.

In Klammern wird die Tonhöhe und die Tonlänge angegeben.

Tonhöhe: Name des Tons in der Tonleiter (Bsp. NOTE_G5) vorprogrammiert in der Library.

Tonlänge: Dauer des Tons in Millisekunden. Wird die Tonlänge nicht angegeben, wird der Ton immer wieder gespielt. Der Ton kann auch mit *stop* () beendet werden.

```
void loop()
{
  speaker.play( NOTE_G5, 100 ); //In Klammer: (Tonart,
  Tonlänge)
  delay (250);
  speaker.play( NOTE_C6, 100 );
  delay (250);
  speaker.play( NOTE_E6, 100 );
  delay (300);
  speaker.play( NOTE_DS6, 100 );
  delay (150);
  speaker.play( NOTE_DS6, 100 );
  delay (220);
  speaker.play( NOTE_DS6, 100 );
  delay (250);
  speaker.play( NOTE_G5, 100 );
  delay (100);
  speaker.play( NOTE_G5, 100 );
  delay (450);
  speaker.play( NOTE_G5, 100 );
  delay (1000);
}
```

- Überprüfe deinen Sketch. Wenn alles stimmt, speichere den Sketch und fahre mit dem Bau der Schaltung weiter
- Da der Aufbau der Schaltung ganz einfach ist, brauchst du keinen Schaltplan. Stecke das Kabel vom – des Lautsprechers auf GND des Arduino. Stecke das Kabel vom + des Lautsprechers auf Pin 12.
- Verbinde den Arduino mit dem Computer und lade den Sketch hoch.
- Melodie wird abgespielt.
- Speichere den Sketch in deinem Educanet-Ordner und lösche die Datei auf dem Computer

9.2.1 Zusatz 1: Melodie ändern

1. Öffne den Sketch 9.2 und speichere ihn unter 9.2.1
2. Programmiere eine eigene Melodie

Folgendes kannst du ändern:

```
speaker.play( NOTE_C6, 100 );  
delay (250);
```

Rot: hier kannst du einen anderen Ton eingeben. Hier

<https://code.google.com/p/rogue-code/wiki/ToneLibraryDocumentation>

findest du eine Liste der Töne.

Gelb: Wie lange soll der Ton abgespielt werden? Zeit in Millisekunden angeben.

Grün: Wie lange dauert die Pause zwischen den einzelnen Tönen? Zeit in Millisekunden angeben.

3. Wie bei Übung 9.2 wird das – des Lautsprecher auf GND des Arduino angeschlossen und das + des Lautsprecher auf Pin 12.
4. Schreibe einen neuen Sketch, überprüfe und lade ihn hoch.
5. Teste verschiedene Melodien und wähle eine aus.
6. Melde dich bei der Lehrerin, damit du deine Melodie der ganzen Klasse vorspielen kannst.
7. Speichere den Sketch in deinem Educanet-Ordner und lösche die Datei auf dem Computer

9.2.2 Zusatz 2: Taster, LED und Lautsprecher kombinieren

Algorithmus: Wenn ein Taster gedrückt wird, leuchtet die LED auf, danach ertönt ein Ton.

Du hast bereits alle Bauteile eingesetzt. Versuche selber, einen Sketch zu schreiben.

Tipp: Schaue bei den Übungen 7 und 9.2 ab.

Als Erweiterung kannst du mehrere Taster, LEDs oder Lautsprecher anhängen.

Speichere deinen Sketch am Schluss auf Educanet mit deinem Namen und der Nummer 9.2.2.

10 Beurteilungspunkte

Übungen	4P	3P	2P	1P
Zusammenbau der Schaltung Stecken der Bauteile auf der Steckplatine, nach Vorgabe, logisch aufgebaut				
Schreiben der Sketche Fehlerfreies abschreiben, Fehler selber finden				
Arbeitseinsatz/ Selbstständigkeit Zügiges, selbstständiges Arbeiten				
Umgang mit Material Bauteile nicht mutwillig zerrstören, Teile nicht verlieren, Materialbox stets komplett				
Zusatzaufgaben Aufgaben gut erledigt (ja/nein)				
19=6, 17/18=5.5, 15/16=5, 13/14=4.5, 11/12=4, 9/10=3-5, -8=3 /19P				

Aufgabe 1:	4P	3P	2P	1P
Löten Bauteile sauber verlötet, feste Verbindungen				
Plan zeichnen Gerade Linien, mit Lineal, Masse stimmen, Massangaben ersichtlich				
Box vorbereiten Löcher sauber gebohrt/gesägt und geschliffen, Sicherer Einsatz der Maschinen und Geräte				
Bauteile und Box zusammenfügen Einbau der Bauteile, Teile sind stabil, Box kann geschlossen werden				
Gesamteindruck Wie sieht deine Spiel-Box von aussen und innen aus? Anordnung der Bauteile				
Arbeitsverhalten Selbstständigkeit, Arbeitseinsatz, Umgang mit Material				
Sketch abändern Sketch sinnvoll abgeändert (ja/nein)				
26/27=6, 23-25=5.5, 20-22=5, 17-19=4.5, 14-16=4, 12/13=3-5, -11=3 /27P				

Sensor/Aktor Werkstatt	4P	3P	2P	1P
Zusammenbau der Schaltung Stecken der Bauteile auf der Steckplatine, nach Vorgabe, logisch aufgebaut				
Selbstständigkeit Übungen selbstständig nach Anleitung durchgearbeitet				
Arbeitseinsatz Zügiges Arbeiten, bei der Sache sein, andere nicht ablenken				
Umgang mit Material Bauteile nicht mutwillig zerstören, Teile nicht verlieren, Materialbox stets komplett				
Zusatzaufgaben Aufgaben gut erledigt (ja/nein)				
19=6, 17/18=5.5, 15/16=5, 13/14=4.5, 11/12=4, 9/10=3-5, -8=3 /19P				

In Bewegung	4P	3P	2P	1P
Bau Einfacher, stabiler Aufbau Objekt kann benutzt werden				
Bau Dein Objekt ist ein- oder aufgebaut, einfacher Auf-/Abbau				
Funktion Dein Objekt funktioniert immer, teilweise				
Umsetzung Wie hast du dein Objekt gebaut? Material sinnvoll ausgewählt und eingesetzt				
Arbeitsverhalten Selbstständigkeit, Arbeitseinsatz, Umgang mit Material				
Sketch Sketch angepasst				
23/24=6, 21/22=5.5, 19/20=5, 16-18=4.5, 13-15=4, 11/12=3-5, -10=3 /24P				

Bemerkungen:

Zeugnis

