



Was ist ein ARDUINO?

Eine Quelloffene Elektronik-Prototypen-Entwicklungs-Plattform.
Aber was soll das bedeuten?

Quelloffene - Die Hard- und Softwarekomponenten eines Produkts dürfen kostenlos genutzt, verändert und weiter verbreitet werden.

Elektronik - Technologie, welche sich die physikalischen Eigenschaften von Elektronen in Leiterbahnen zu Nutze macht

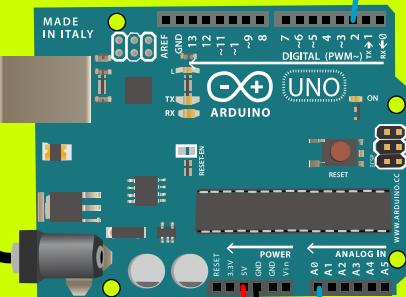
Prototypen - Probemodell eines Geräts für die Serienproduktion.

Plattform - Eine Ansammlung von passender Hardware und Software die einem das Laufenlassen eigener Programme ermöglicht.

Microcontroller

Photozelle

LED



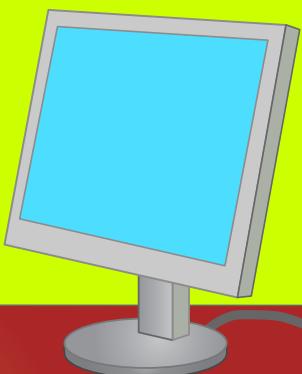
Steckbrett

Ein Arduino ist ein **Mikrocontroller**, ein sehr kleiner Computer, den man dazu bringen kann auf Geschehnisse zu reagieren. Er kann Sachen messen (z.B. wie hell ein Raum gerade ist). Und er kann dann andere Dinge dazu bringen auf diese Veränderungen zu reagieren. (z.B. Der Raum wird dunkel und eine LED geht an).

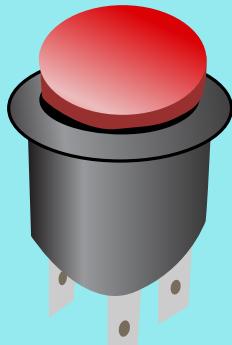


Oder er könnte auf so einfache Sachen wie das Umlegen eines Schalters reagieren.

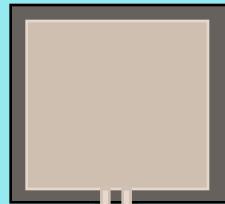
Bei PCs sind Maus und Tastatur die klassischen Eingabegeräte. Die Bildschirmanzeige ist eine Ausgabemethode.



Mikrocontroller arbeiten wie Computer mit **Ein- und Ausgaben**. Eingaben können vom Benutzer gemacht werden oder aus der Umgebung abgelesen werden. Ausgaben können Warntöne sein, mit der auf die gesammelten Informationen reagiert wird.



Tastschalter

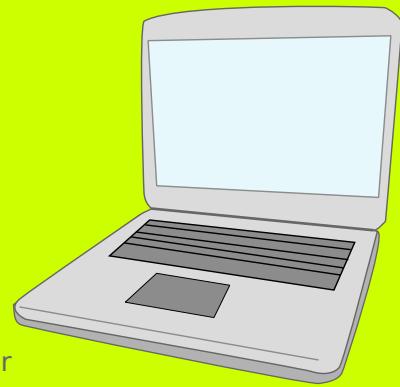


Auf Druck
reagierender
Widerstand

Einen Schalter oder einen Messfühler könnte man als Eingabegerät für den Arduino nutzen.



Gleichstrom-Motor



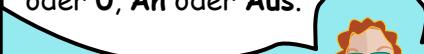
Jedes Gerät, welches man An- und Abstellen kann wäre als Ausgabegerät geeignet. Man könnte einen Motor oder sogar einen Computer verwenden.



Was ist der Unterschied zwischen digitalen und analogen Ein- und Ausgängen?

Eingänge und Ausgänge können **digital** oder **analog** sein. Digitale Informationen sind binär. Wahr oder Falsch. Analoge Informationen sind kontinuierlich und haben unendlich viele Zwischenstufen

Digitale Informationen sind **diskret** und **endlich**. Sie können sich nur in zwei Zuständen befinden: **1** oder **0**, **An** oder **Aus**.



Analoge Informationen sind **kontinuierlich**. Sie können eine **unendlich** große Anzahl verschiedener Werte annehmen.

Ein Tastschalter ist digital. Ein Messfühler ist analog. Die Anzahl der Zwischenstufen des Fühlers wird jedoch durch die Umwandlung in digitale Daten endlich.



Spannung?
Stromstärke?
Widerstand?
Das Ohmsche
Gesetz?

Bevor wir den Arduino anschließen
sollten wir uns noch ein paar
Begriffe und Prinzipien zuführen,
die uns zeigen, wie Elektrizität und
somit Elektronik funktioniert.

Spannung (U)

Ist eine
Masseeinheit für
elektrisches
Potenzial.
Gemessen wird
sie in der
Einheit **Volt** [V].

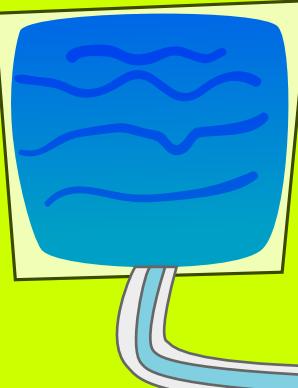
Stromstärke (I)

Stromstärke ist
die Flussmenge
durch ein
leitendes
Material. Sie
wird in **Ampere**
[A] gemessen.

Widerstand (R)

Misst wie stark
sich ein
leitendendes
Material dem
elektrischen Fluss
entgegenstellt.
Gemessen in
Ohm [Ω].

Elektrizität ist der Fluss von Energie durch ein leitendes Material.



Die Flussgeschwindigkeit wird durch
die Spannung bestimmt

Widerstand lässt Fluss zu
oder behindert ihn



Die Menge des Fluxes durch die
Leitung ist die Stromstärke

Die Wasser-Analogie wird häufig genutzt um die Begriffe zu erklären.

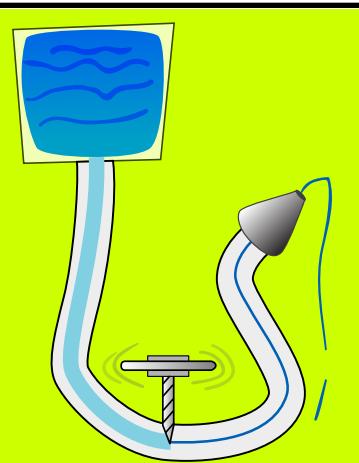
Das Ohmsche Gesetz

Stromstärke = Spannung / Widerstand
 $(I = U / R)$
oder

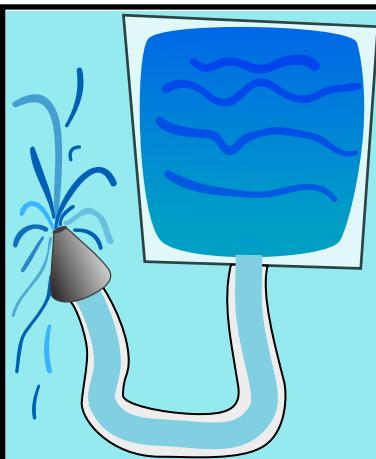
Widerstand = Spannung / Stromstärke
 $(R = U / I)$
oder

Spannung = Widerstand * Stromstärke
 $(U = R * I)$

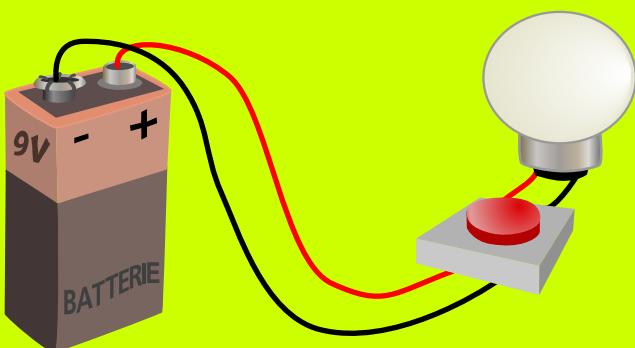
Es gibt einige Zusammenhänge zwischen Spannung, Stromstärke und Widerstand, die vom deutschen Physiker Georg Ohm entdeckt wurden.



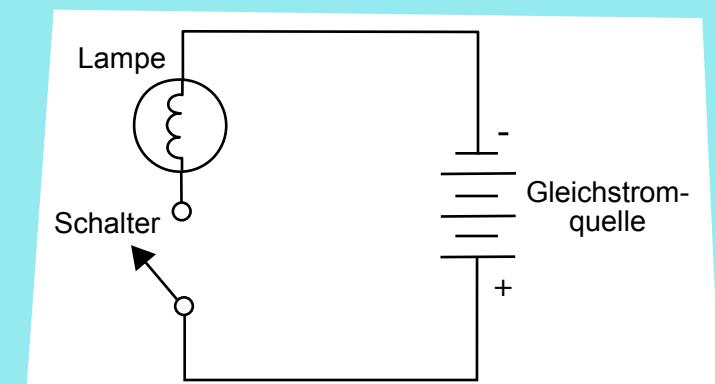
Ein Beispiel: Erhöht man den Widerstand so fliesst weniger Wasser



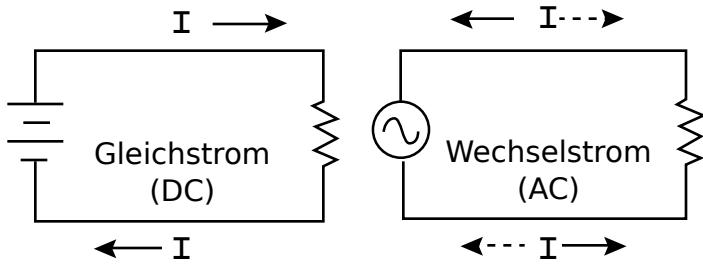
Erhöht man das Potenzial, so fliesst mehr Wasser.



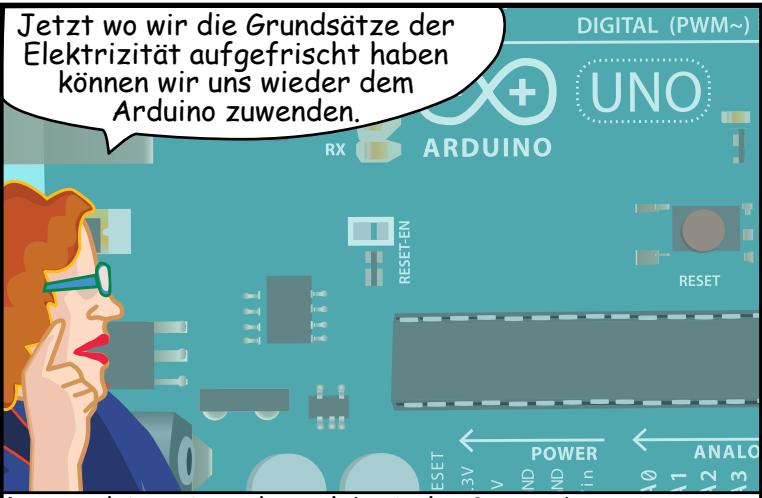
Schauen wir uns nun eine einfache **Schaltung** an. Jede Schaltung ist ein geschlossener Kreislauf, der eine **Energiequelle** (Batterie) und eine **Last** (Lampe) beinhaltet. Die Last wandelt die Energie der Batterie um und verbraucht sie. Ein Schalter ist hier auch verbaut.



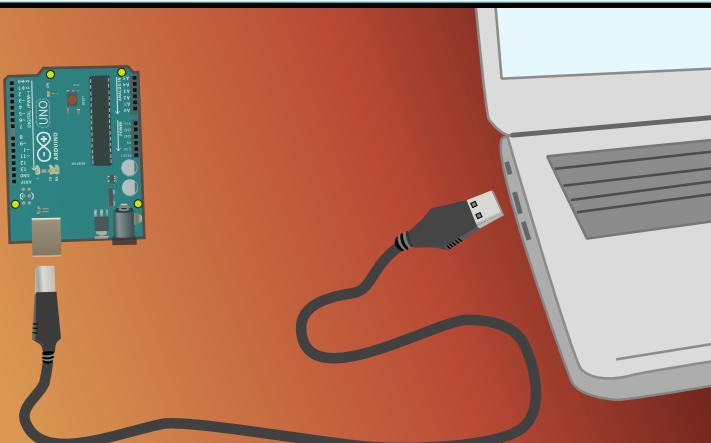
Dies ist der **Schaltplan** der einfachen Schaltung. Die Schaltung wird hier repräsentiert, indem Symbole anstatt der Komponenten gezeigt werden. Wenn der Schalter geschlossen wird fliesst Strom zwischen den Polen der Energiequelle und erleuchtet die Lampe.



Es gibt zwei Arten von Schaltungen. **Gleichstrom-** und **Wechselstromschaltungen**. Bei einer Gleichstrom-Schaltung fliesst der Strom immer in eine Richtung. Bei einem Wechselstromkreis kehrt sich der Stromfluss regelmaessig in die entgegengesetzte Richtung. Wir interessieren uns nur fuer Gleichstromschaltungen.



Der Arduino wie andere elektrische Geräte benötigt Spannung zum Arbeiten und da wir ihn jetzt programmieren möchten schliessen wir ihn an den Computer an.



Wenn wir den Arduino mit einem USB-Kabel an einen Computer anschliessen wird er über das Kabel mit den benötigten 5 Volt versorgt und wir können loslegen.



Vorher musst du jedoch noch die kostenlose Software von obiger Adresse herunterladen und installieren. Die Arduino Software läuft auf den folgenden gängigen Plattformen: Mac OSX, Windows und Linux

Wenn du Hilfe benötigst bei der Installation auf einem Mac:

<http://www.arduino.cc/en/Guide/MacOSX>

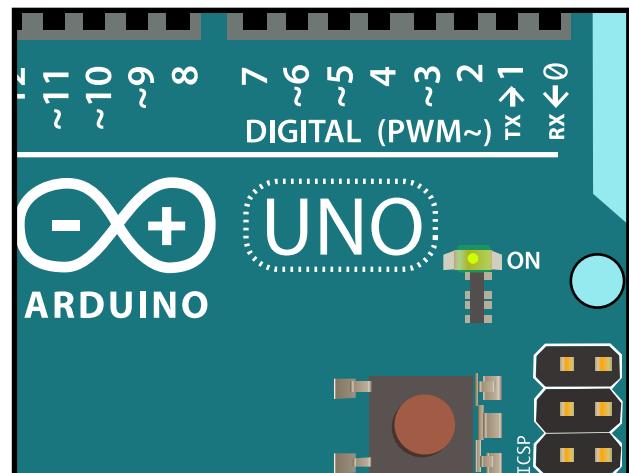
Wenn du Hilfe benötigst bei der Installation auf einem Windows-PC:

<http://www.arduino.cc/en/Guide/Windows>

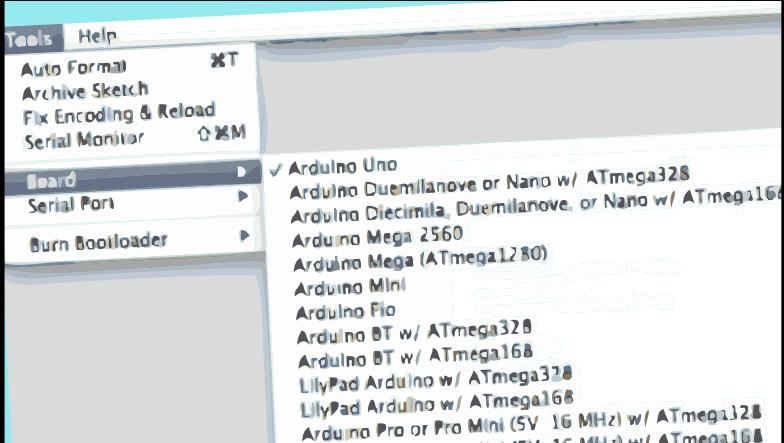
Wenn du Hilfe benötigst bei der Installation auf einem Linux-System:

<http://www.arduino.cc/playground/Learning/Linux>

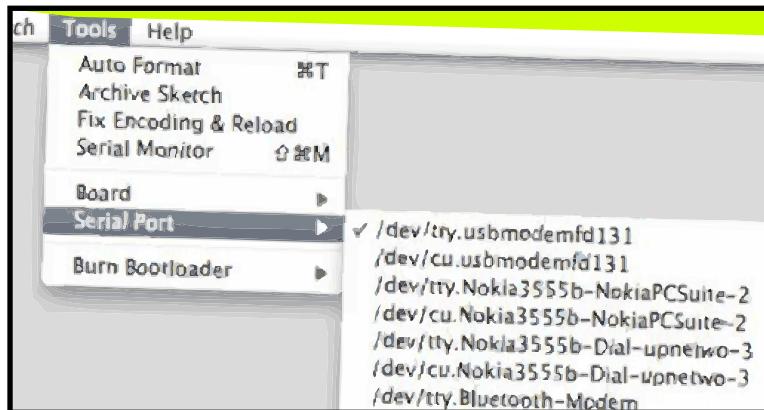
Besuche die obigen Adressen, wenn du detaillierte Anleitungen für die Software-Installation benötigst.



Wenn du die Software installiert hast und der Arduino angeschlossen wurde sollte eine mit ON beschriftete LED aufleuchten.



Nun Starte die Arduino-Software. Im Menü Tools wähle aus welches Board du benutzt (Tools > Board). Zum Beispiel **Arduino UNO**.



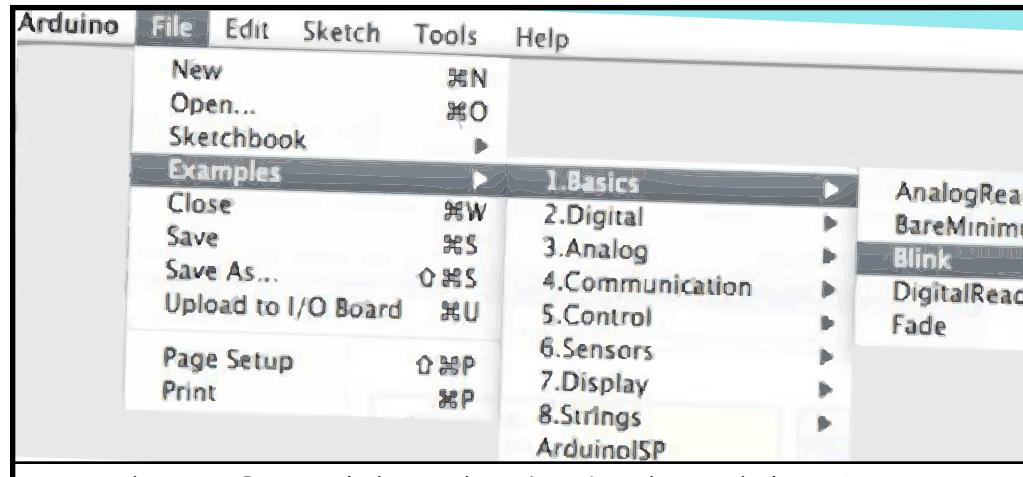
Als nächstes muss man den Seriellen Port auswählen (**Tools > Serial Port**). Auf einem Mac sieht der Eintrag in etwa wie `/dev/tty.usbmodem` aus. Auf einer Windows-Maschine vielleicht `COM3`. Häufig der Eintrag der nach Einstecken des Kabels hinzukommt.

Was ist eine
Integrierte
Entwicklungs
Umgebung ?

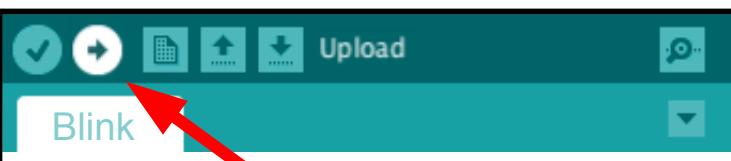
Englisch:
Integrated
Development
Environment



Die Software, die du heruntergeladen hast ist eine **IDE**. Es handelt sich um einen Texteditor, verbunden mit einem Kompilierer und weiteren Funktionen, die es Programmierern erleichtert Software zu entwickeln.



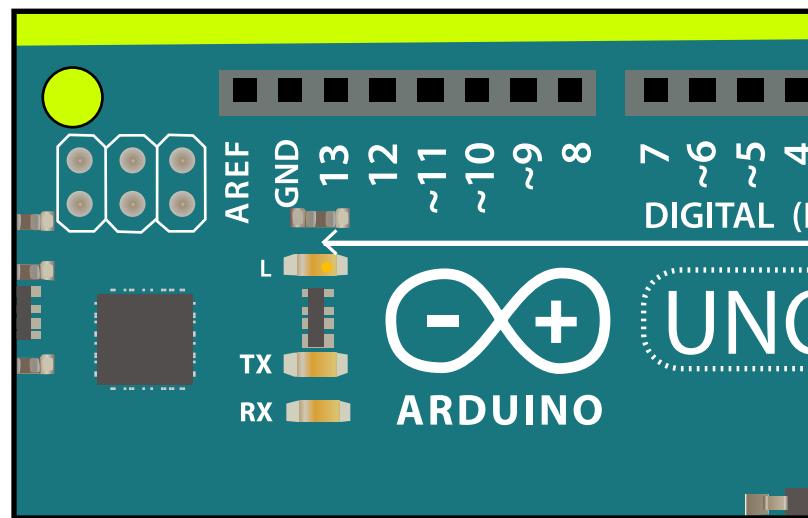
Die Arduino IDE ermöglicht es dir, **Sketche**, das sind kleine **Programme**, zu schreiben und sie auf den Arduino hochzuladen. Öffne das **Blink** - Beispielprogramm im File-Menü. File > Examples > 1.Basics > Blink.



Upload Knopf

```
int ledPin = 13;  
  
void setup() {  
  pinMode(ledPin, OUTPUT);  
}  
  
void loop() {
```

Um einen Sketch zum Arduino Mikrocontroller hochzuladen klick den **Upload Knopf** in der Knopfleiste im oberen Teil des Fensters. Unten werden einige Nachrichten erscheinen. Schlussendlich sollte **Done Uploading** zu lesen sein.



Die LED am Pin 13 vom Arduino fängt an zu blinken.

```

void setup() {
    // initialize the digital pin as an output.
    // Pin 13 has LED connected on most Arduino boards
    pinMode(13, OUTPUT);
}

void loop() {
    digitalWrite(13, HIGH);      // set the LED on
    delay(1000);                // wait for a second
    digitalWrite(13, LOW);       // set the LED off
    delay(1000);                // wait for a second
}

```

Ein Sketch, genauso wie jedes andere Programm auch, ist eine Menge von Anweisungen für den Computer. Wenn wir uns den Blink-Sketch genauer ansehen so können wir feststellen, dass er aus 2 größeren Teilen besteht: **SETUP** und **LOOP**.

<http://arduino.cc/en/Reference/HomePage>



Besuch später mal die Arduino Webseite um eine Referenz der Sprache zu sehen sowie viele andere Ressourcen zum Erlernen der Sprache.

SETUP: Wird einmal ausgeführt beim Programmstart

LOOP: Wird immer wieder und wieder ausgeführt

Diese Code-Blöcke heissen **Funktionen** und jeder Sketch hat welche. Sie werden durch geschweifte Klammern {} begrenzt.

```

void setup() {           // Deklariert einen Code-Block
    pinMode(13, OUTPUT); // Setzt Pin 13 auf Ausgabe
}                         // Ende des Code-Blocks

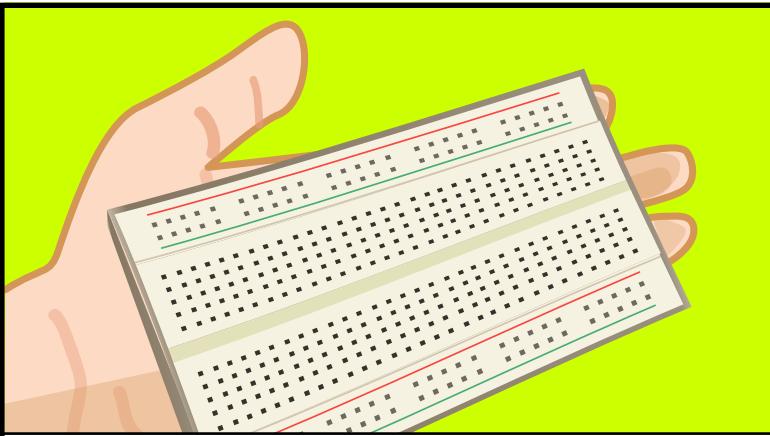
```

```

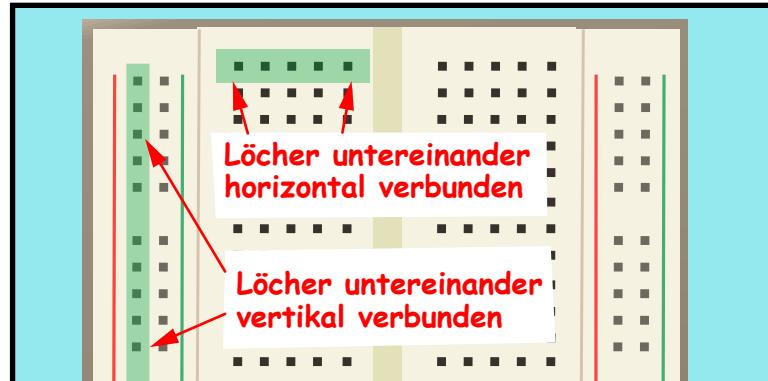
void loop() {           // Deklariert einen Code-Block
    digitalWrite(13, HIGH); // Schaltet Pin 13 an
    delay(1000);           // Pause für eine Sekunde
    digitalWrite(13, LOW);  // Schaltet Pin 13 ab
    delay(1000);           // Pause für eine Sekunde
}                         // Ende des Code-Blocks

```

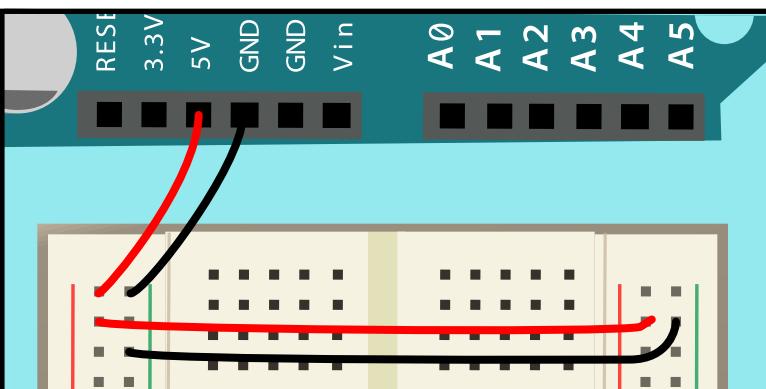
Jetzt aber schau dir Zeile für Zeile das Skript an und sehe in den Kommentaren was jede Zeile macht.



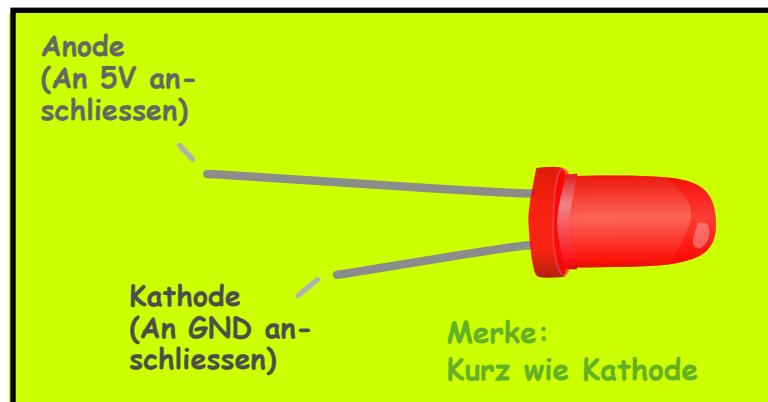
Aber wie können wir Objekte steuern, die nicht auf dem Arduino sind? Dafür verbinden wir den Arduino mit einem **lötfreien Experimentier-Steckbrett**. Damit können wir blitzschnell Testschaltungen aufbauen.



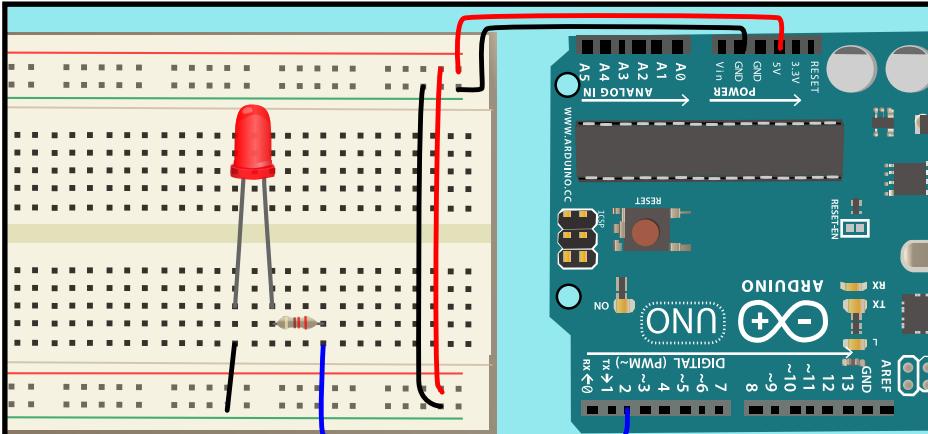
Dieses Steckbrett hat je 2 Reihen mit Löchern auf der linken und rechten Seite, die **vertikal** miteinander verbunden sind sowie 5 Spalten links und rechts der kleinen Vertiefung in der Mitte. Diese sind **horizontal** miteinander verbunden.



Wir verbinden 5V (Potential/+Pol) und GND (Masse/
Erde) vom Arduino mit den vertikal verbundenen
Streifen links und rechts des Brettes mit 0,7mm
dickem Draht. Elektronische Bauelemente können jetzt
an die Löcher in der Mitte sowie an 5V und GND
angeschlossen werden.



Wenn Strom durch eine LED (Licht emittierende
Diode) in der richtigen Richtung fliest, dann leuchtet
sie auf. Wir stecken eine LED auf das Experimentier-
brett. Dann legen wir Leitungen zum Arduino, so dass
wir die LED mit Code steuern können.

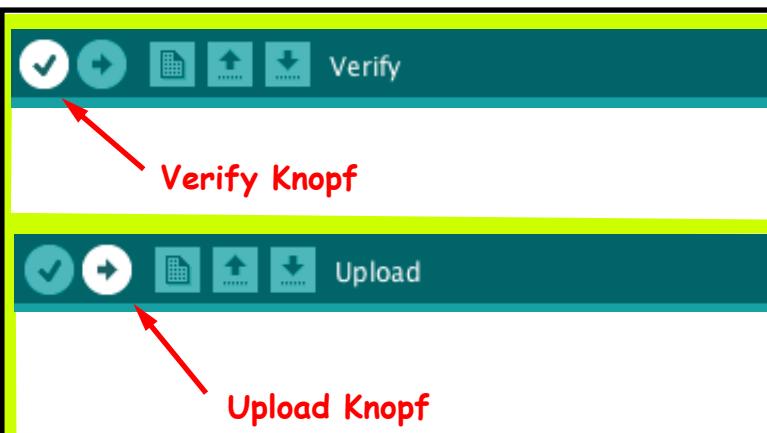


Die Anode wird mit Pin 2 am Arduino über einen 220 Ohm Widerstand verbunden. Die Kathode wird mit GND verbunden. Die Pins 2 bis 13 können als digitale Ein- oder Ausgänge konfiguriert werden. Klick auf den Neu Knopf um einen neuen Sketch zu machen.

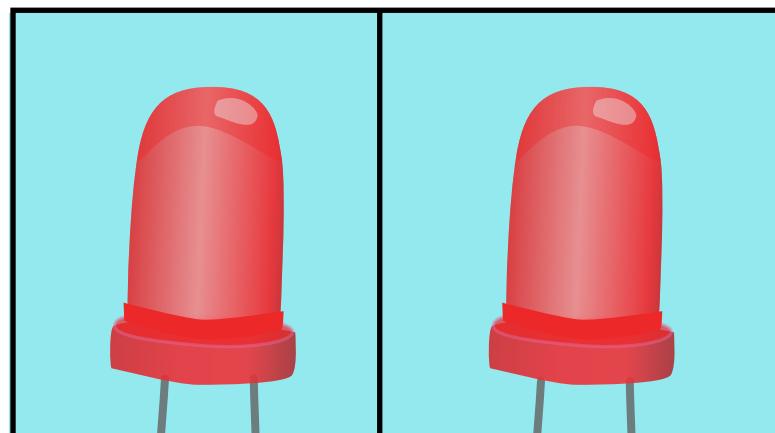
```
void setup() {
    pinMode(2, OUTPUT);
}
```

```
void loop() {
    digitalWrite(2, HIGH);
    delay(500);
    digitalWrite(2, LOW);
    delay(500); }
```

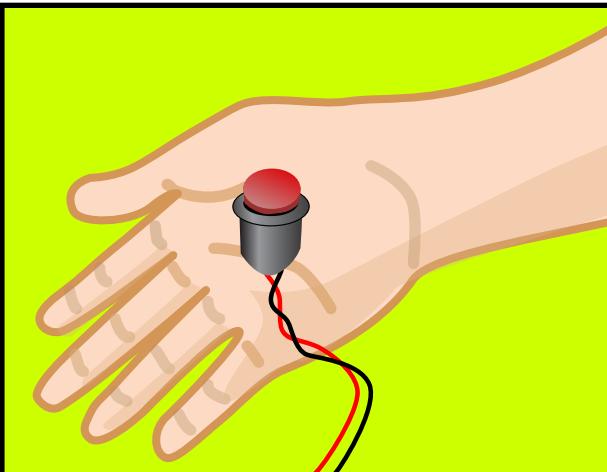
In **Setup** konfigurieren wir Pin 2 als Ausgabe. In **Loop** setzen wir Pin 2 auf HIGH, was die LED leuchten lässt. Delay sorgt für eine Pause von 500 Millisekunden ($\frac{1}{2}$ Sek.). Wenn Pin 2 auf LOW gesetzt wird geht die LED aus. Dann eine weitere $\frac{1}{2}$ Sek. Pause.



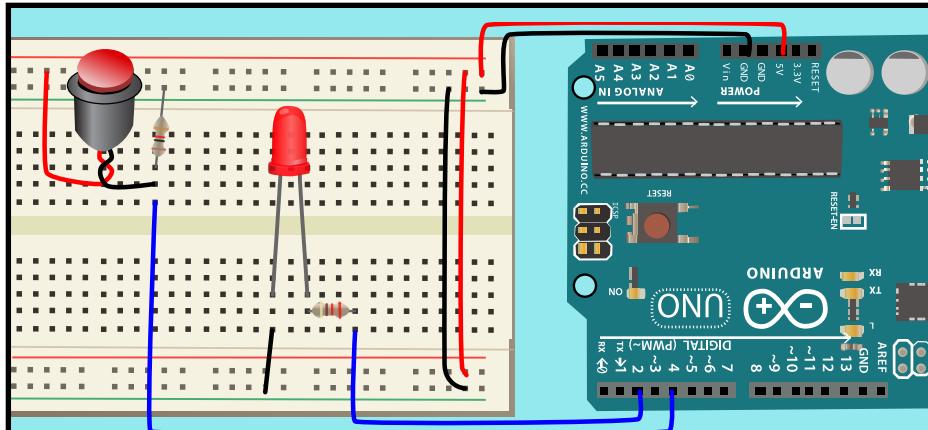
Klicke den Verify Knopf im Menü um deinen Code prüfen zu lassen. Wenn es keine Fehlermeldung gab, klicke auf Upload um dein Programm auf den Arduino zu befördern.



Die LED leuchtet für eine halbe Sekunde, dann ist sie für eine halbe Sekunde aus. So blinkt sie immer wieder und wieder.



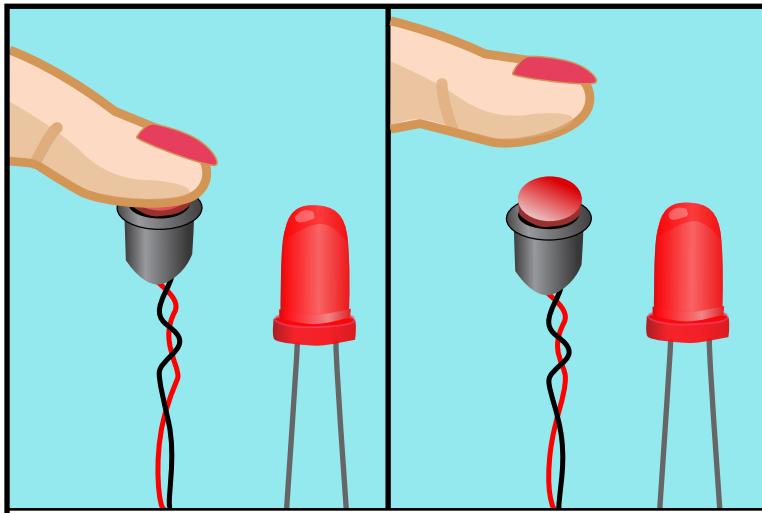
Als nächstes fügen wir dem Ganzen einen digitalen Schalter hinzu, so dass wir die LED An- und Ausschalten können.



Verbinde den einen Kontakt des Tasters mit Pin 4 am Arduino. Den selben Kontakt des Schalters schliessen wir an Erde über einem 10 kiloOhm Widerstand an. Die andere Seite des Schalters soll mit 5V verbunden werden. Die LED lassen wir wo sie ist.

```
void setup() {  
    pinMode(2, OUTPUT);  
    pinMode(4, INPUT);  
}  
  
void loop() {  
    if(digitalRead(4)){  
        digitalWrite(2, HIGH);  
    }else{  
        digitalWrite(2, LOW);  
    }  
}
```

Jetzt schreiben wir den Code. In setup definieren wir Pin 2 als Output und Pin 4 als Input. Im Loop benutzen wir eine if-Anweisung, eine Bedingung. Wenn das Signal an 4 HIGH ist setzen wir den LED-Pin HIGH, ansonsten setzen wir ihn auf LOW und schalten die LED somit ab.

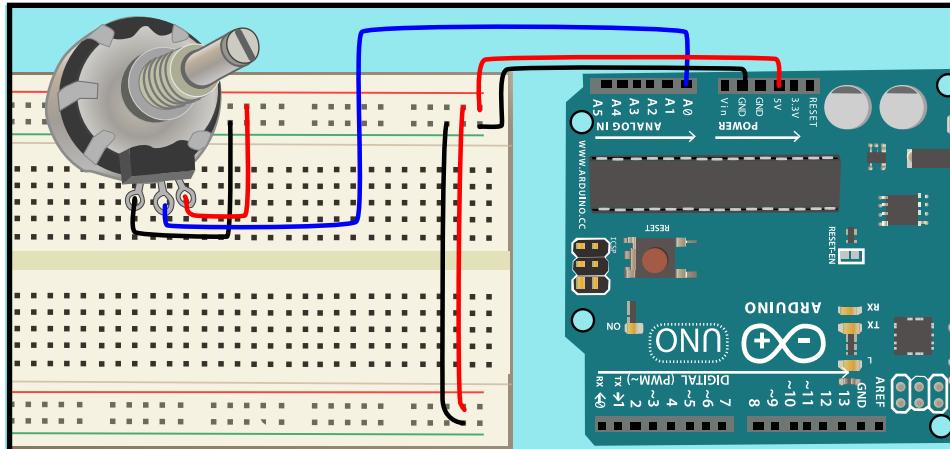


Die LED leuchtet wenn der Schalter gedrückt wird.

Ein Potentiometer, auch Poti genannt, ist ein veränderlicher Widerstand. Die Größe des Widerstandes wird größer oder kleiner je nachdem in welche Richtung er gedreht wird.



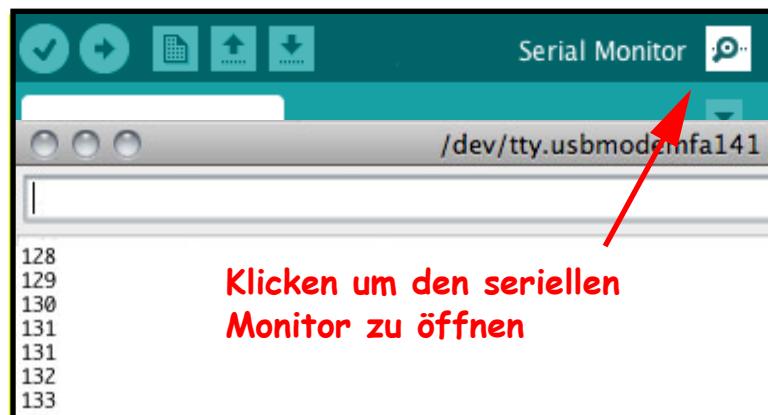
Lass uns mal ein analoges Eingabegerät ausprobieren, den Potentiometer.



Schliesse den mittleren Pin des Potis an den **Analogpin A0** an. Die anderen Anschlüsse des Potentiometers müssen auf GND und 5V gelegt werden. So gibt der Mittelpin Teilspannungen zwischen 0 und 5V aus.

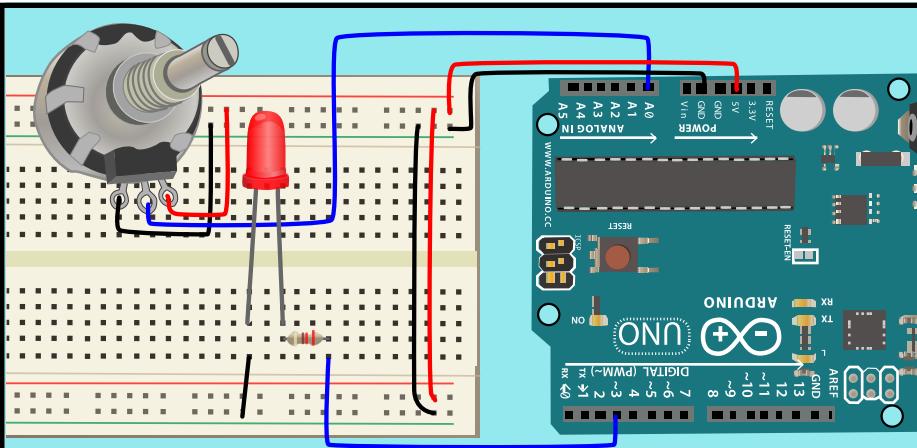
```
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    Serial.println(analogRead(A0));  
}
```

Erstmal sehen wir uns die unterschiedlichen Werte, die wir durch das Drehen des Potis erhalten, mit Hilfe des **Seriellen Monitors** an. Im Code initialisieren wir dafür die serielle Schnittstelle und setzen sie auf eine Geschwindigkeit von 9600 Baud. Im loop lesen wir den Wert von Analog Pin A0 und geben ihn zur Ausgabe an die Serielle Schnittstelle mit der `println`-Funktion.

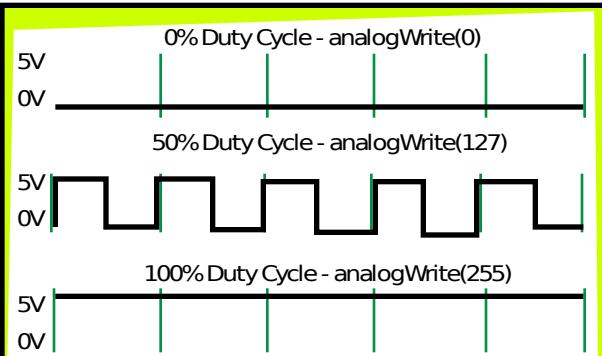


Klicken um den seriellen Monitor zu öffnen

Nachdem du den Sketch auf den Arduino hochgeladen hast, klicke auf den Knopf für den **Seriellen Monitor** um die Werte sehen zu können. Ein Fenster wird sich öffnen und du wirst Werte von 0 bis 1023 sehen, je nachdem wohin du den Potentiometer gedreht hast.



Wir könnten die wechselnden Werte, die wir vom Poti empfangen, nutzen um eine LED zu dimmen. Steck die LED zurück ins Brett und verbinde sie mit dem Arduino über Pin 3.



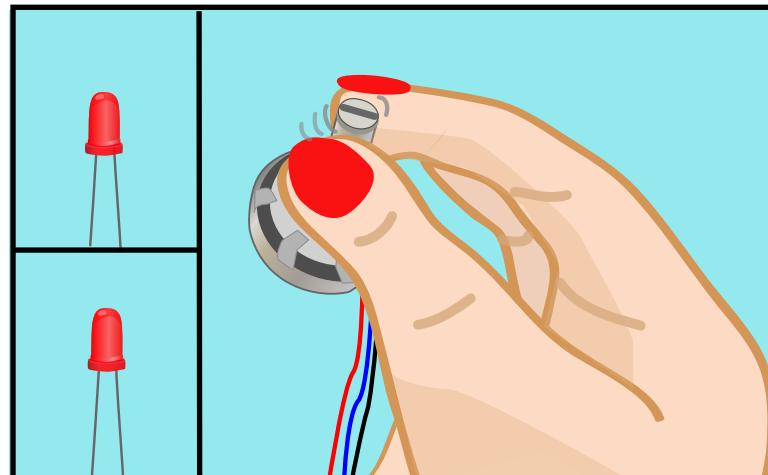
Wir werden die **PulsWeitenModulation** nutzen um Analogspannungen zu simulieren. Durch das An- und Abschalten der Spannung mit unterschiedlichen Perioden kann die LED 0-100% der Zeit an sein. PWM funktioniert auf Pins 3, 5, 6, 9, 10 und 11

```
int sensorValue = 0;

void setup() {
  pinMode(3,OUTPUT);
}

void loop() {
  sensorValue = analogRead(A0);
  analogWrite(3, sensorValue/4);
}
```

Zunächst erstellen wir eine Variable um den Wert des Potis zu speichern. In setup konfigurieren wir Pin 3 für Ausgabe. In loop speichern wir den Wert den wir von A0 gelesen haben in unserer Variable. Dann schreiben wir den Wert an Pin 3 (LED), müssen ihn vorher aber noch durch 4 teilen damit er zwischen 0 und 255 liegt.



Die Helligkeit der LED lässt sich jetzt von komplett aus bis sehr hell ändern, je nachdem wie du den Poti einstellst.



Das war eine kurze Einführung in die Mikrocontroller-programmierung mit Arduino. Besuch die Links in den nächsten Kästen und du wirst noch viel mehr erfahren.

Links

Software

Software Download

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

Referenz der Sprache

<http://arduino.cc/en/Reference/HomePage>

Bezugsquellen

Distributorenliste

<http://arduino.cc/en/Main/Buy>

TinkerSoup

<http://www.tinkersoup.de/index.php?cPath=22>

Watterott

<http://www.watterott.com/de/Boards-Kits/Arduino>

Elektronikladen

<http://elmicro.com/de/arduino.html>

Tutorials

Arduino Tutorials

www.arduino.cc/playground/Deutsch/HomePage

Lady Ada

www.freeduino.de/books/arduino-tutorial-lady-ada

Mehr Deutsche Tutorials

www.freeduino.de/books/

Bücher

Die elektronische Welt mit Arduino entdecken

Erik Bartmannn

Arduino - Physical Computing für Bastler, Designer und Geeks

Verlag: O'Reilly

Make: Elektronik: Lernen durch Entdecken

Charles Platt

TEXT UND ZEICHNUNGEN VON JODY CULKIN LUST AUF MEHR? BESUCHE JODYCULKIN.COM

Besonderen Dank an Tom Igoe, Marianne Petit, Calvin Reid, die Fakultät und Bediensteten des Programms Interaktive Telekommunikation an der New Yorker Universität, besonderen Dank an Dan O'Sullivan, Danny Rozin und Red Burns. Dank auch an Cindy Karasek, Chris Stein, Sarah Teitler, Kathy Goncharov & Zannah Marsh.

Vielen herzlichen Dank an das Arduino-Team, dass uns diese robuste und Flexible Open Source Plattform gebracht hat.

Genausoviel Dank auch an die lebendige, aktive und stetig wachsende Arduino-Gemeinschaft.

Introduction to Arduino by Jody Culkin
steht unter der Creative Commons
Attribution-NonCommercial-ShareAlike 3.0
Unported Lizenz.