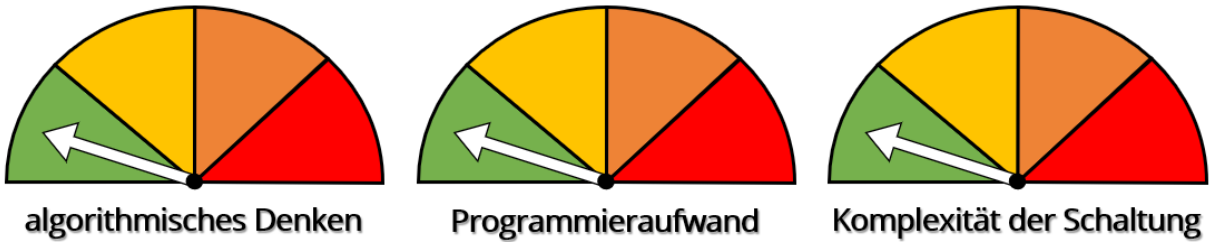
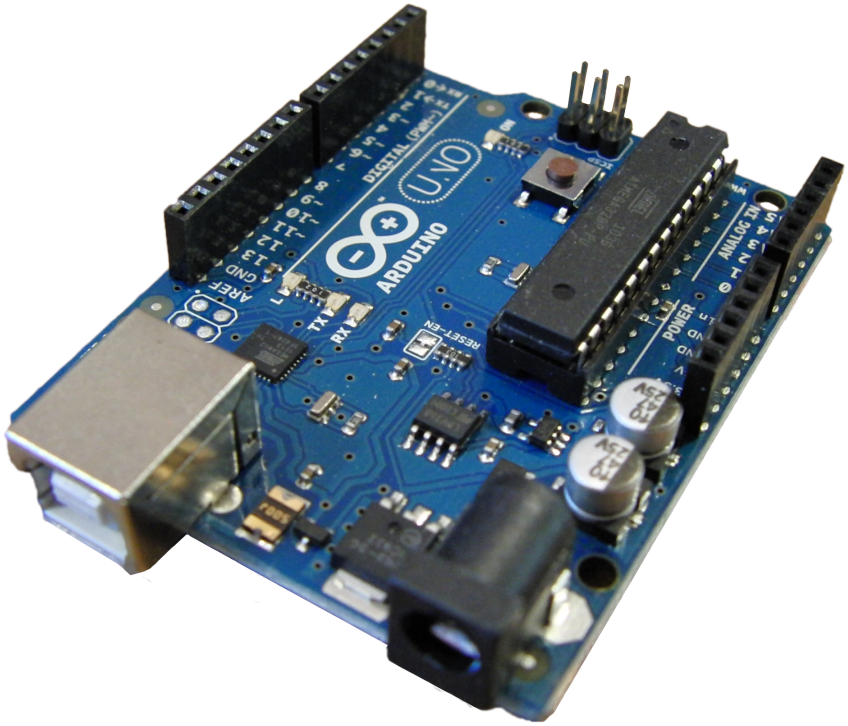
TECHNISCHE UNIVERSITÄT DRESDEN

Arduino Uno

Einstieg | Blinkende Lichter





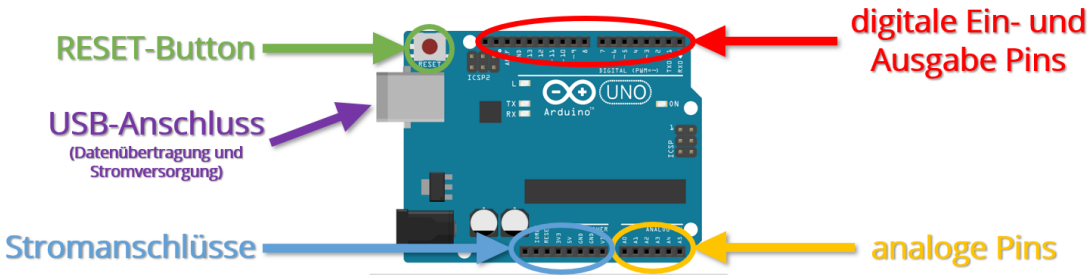
A Arduino Uno board, JotaCartas, CC BY-SA 2.0 , https://commons.wikimedia.org/

Professur für Didaktik der Informatik

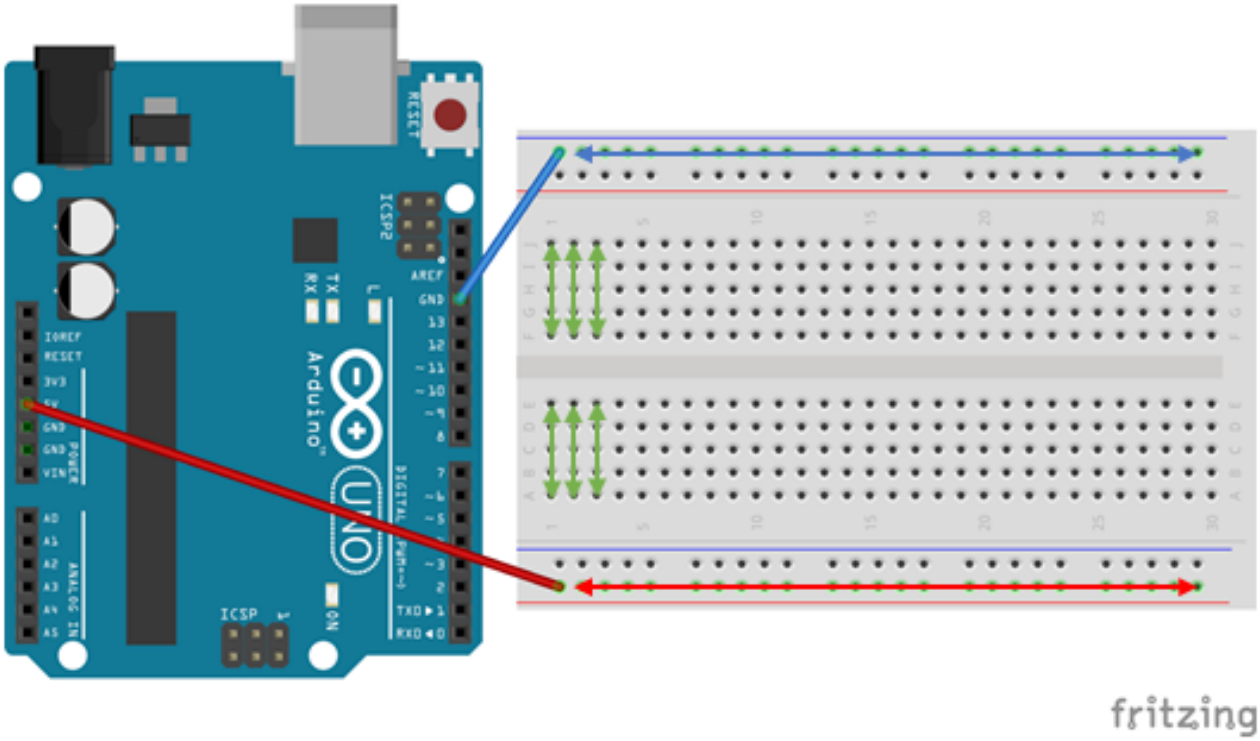
UM WAS WIRD ES IN DIESER STATION GEHEN?

Ob zu Hause, oder in der Schule, jeden Tag siehst du Lichter sich automatisch ein- und ausschalten. Du siehst Blinklichter an Autos, Ampeln und der Baustelle. In dieser Station wollen wir genau diese Lichter nachbauen und programmieren. Doch zuerst lasst uns einen Blick auf den Arduino werfen.

DER ARDUINO



USB-Anschluss	Der USB-Anschluss liefert den Strom für den Arduino. Außerdem wird über ihn dein geschriebenes Programm auf dem Arduino installiert.
Stromanschlüsse	Deine Bauteile auf dem Steckbrett brauchen Strom. Deswegen musst du sie mit den Stromanschlüssen verbinden. 
digitale Ein- und Ausgabepins	Diese Pins können Strom an- und ausschalten. Sie können aber auch lesen, ob am Ende der Verbindung Strom fließt.
analoge Pins	Diese Pins können das gleiche, wie digitale Pins. Sie können außerdem bei fließendem Strom verschiedene Spannungswerte einstellen. Durch sie kannst du zum Beispiel eine LED dimmen.



Auf dem Steckbrett bringst du deine Bauteile an. Die äußeren Leisten (rot und blau) sind durchgehend miteinander verbunden. Verbindest du also ein Loch der äußeren blauen Leiste mit dem Minuspol, so sind auch die anderen Löcher der Leiste mit ihm verbunden. Für die mittleren Leisten (grün) gilt das gleiche. Hier ist die Verbindung aber vertikal.

AUFGABE 1 – VERSORGE DAS STECKBRETT MIT STROM

Verbinde die äußeren Leisten mit den Stromanschlüssen, wie im oberen Bild. Der +Pol sollte immer mit der roten Leiste verbunden werden, der -Pol mit der blauen Leiste. Verwende dafür zwei beliebige Kabel.

NUN LASSEN WIR UNS EIN LICHT AUFGEHEN

Als nächsten Schritt, bringen wir eine LED zum Leuchten.

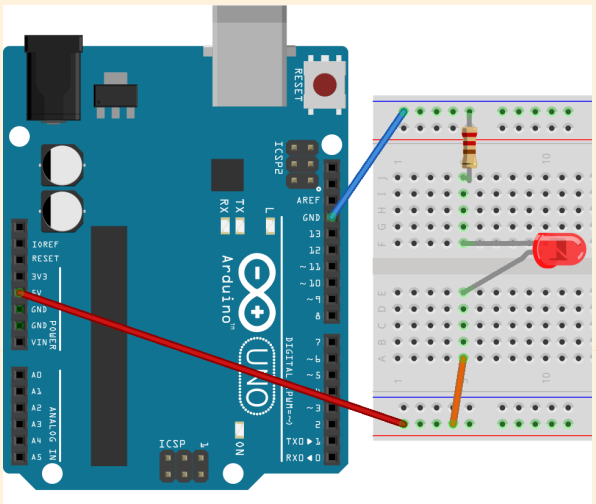
WAS IST EINE LED?

Eine **LED** ist eine Art kleine Lampe. Diese funktioniert jedoch nur, wenn du sie richtig herum anschließt. Bei genauerem Hinschauen siehst du, dass ein Beinchen länger ist, als das andere. Das lange muss mit dem Pluspol verbunden werden!
Würdest du nun die LED anschließen, würde sie kaputt gehen. Das liegt daran, dass sie alles an Strom aufnimmt, auch wenn sie dadurch überhitzt. Um das zu verhindern, brauchst du einen **Widerstand**. Baue diesen immer in deinen Stromkreis ein.

AUFGABE 2 – BRINGE DIE LED ZUM LEUCHTEN

Baue die LED und den Widerstand auf dem Steckbrett an. Verbinde sie wie auf dem Bild mit den Stromleisten.

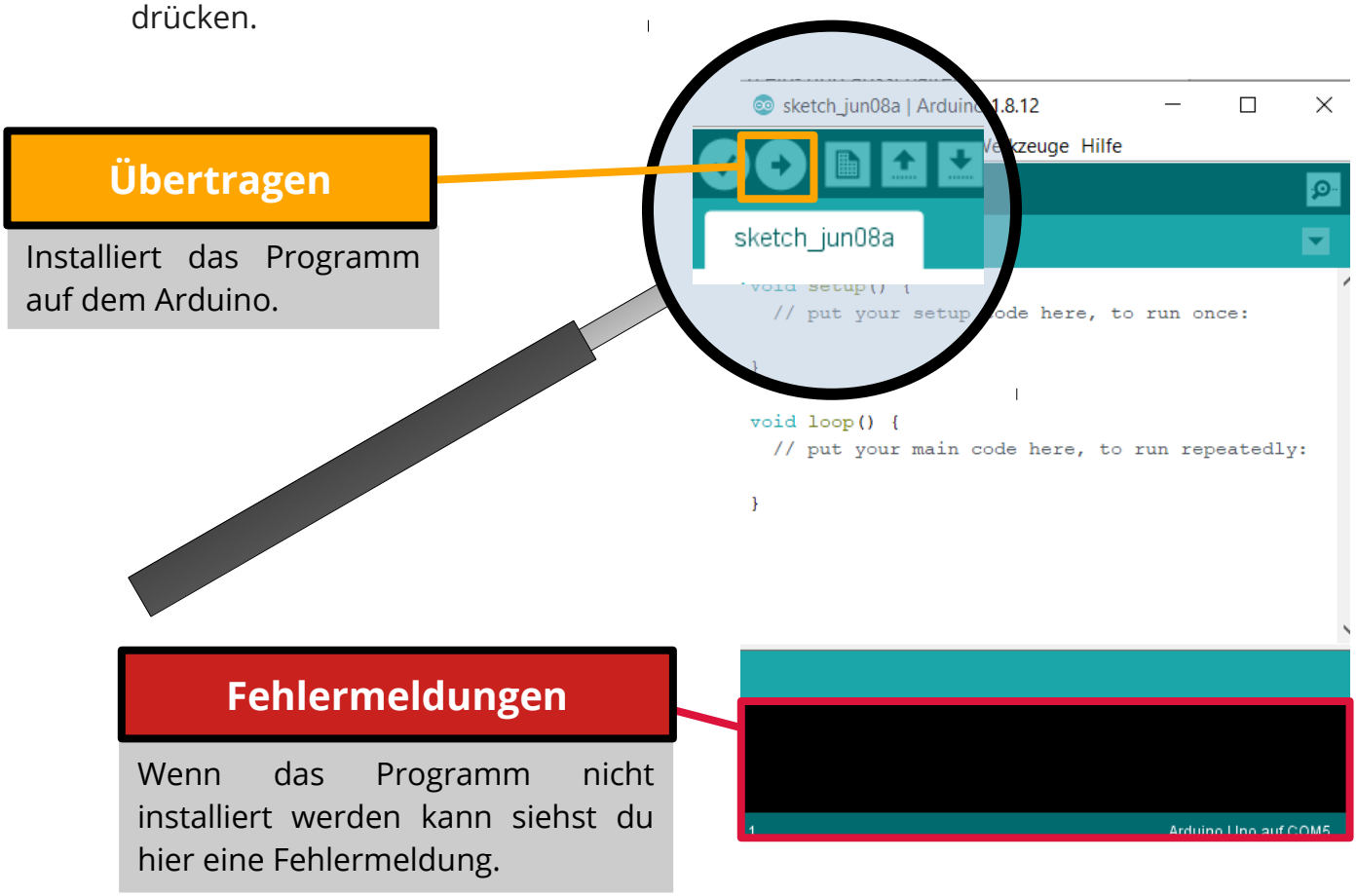
Alles verbunden? Dann verbinde jetzt das USB-Kabel mit dem Computer!



Die LED sollte jetzt leuchten! 👍 😊

DIE LED DURCH DEN ARDUINO STEUERN

Bisher haben wir die LED durch die Versorgung mit Strom zum Leuchten gebracht. Wir wollen aber nicht bei jedem Ein- und Ausschalten der LED die Kabel umstecken müssen. Der Arduino soll den Strom automatisch ein- und ausschalten.
Deswegen brauchen wir die Programmierumgebung, um Code zu schreiben. Diese siehst du auf dem Bild unten. Wenn du dein Programm ausprobieren möchtest, musst du es auf dem Arduino installieren. Dazu musst du den Button Übertragen drücken.



Das Programm besteht aus zwei Teilen:

```
void setup() {
  ...
}
```

Der Code im `setup()`-Teil wird nur ein Mal am Anfang des Programms ausgeführt. Hier legst du die Einstellungen für den Arduino fest. Zum Beispiel, wo eine LED angeschlossen ist.

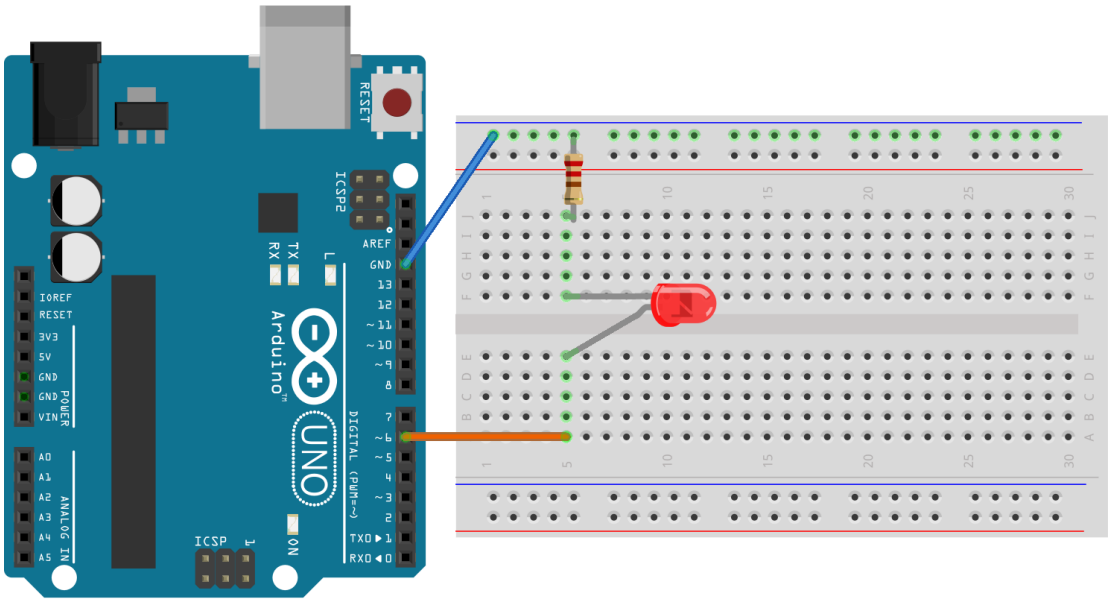
```
void loop() {
  ...
}
```

Der Code im `loop()`–Teil wird unendlich oft wiederholt. Hier schreibst du dein tatsächliches Programm. Zum Beispiel, wann eine LED angeschaltet wird und wie lange sie leuchten soll.

AUFGABE 3 – LED GESTEUERT ZUM LEUCHTEN BRINGEN

Bringe die LED zum Leuchten, indem du sie mit dem Arduino steuerst. Schreibe dafür einen passenden Code. Beachte dabei die folgenden Schritte.

1. Die Schaltung abändern
   Nehme das Kabel, das zuvor mit dem Pluspol verbunden war und stecke es in einen digitalen Pin, zum Beispiel Pin 6. Das lange Beinchen der LED sollte jetzt also mit einem Pin verbunden sein.

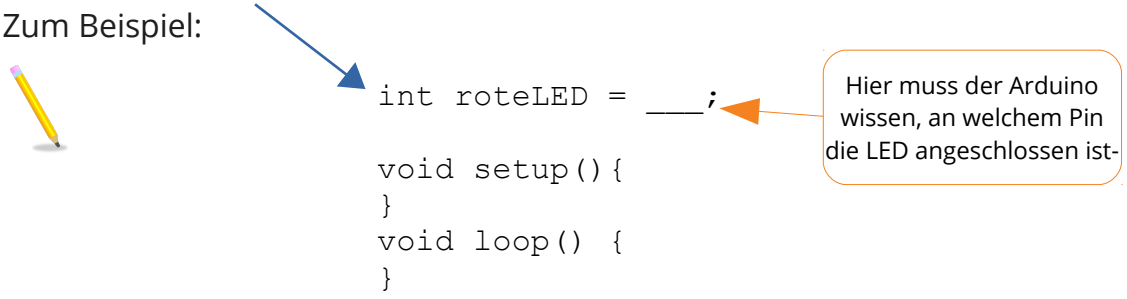


2. Variable definieren
   Als nächstes musst du dem Arduino sagen, an welchem digitalen Pin die LED angeschlossen ist. Dafür musst du eine Variable definieren.

WAS IST EINE VARIABLE?

Eine **Variable** speichert eine Zahl unter einem bestimmten Namen, dem **Variablennamen**. Beim Programmieren ist das nützlich, weil man sich Namen besser merken kann. Außerdem kann es vorkommen, dass sich Zahlen verändern, zum Beispiel wenn der Pin, an dem eine LED angeschlossen ist, geändert wird. Dann muss nur die Zahl in der Variablendefinition geändert werden und nicht jedes Mal die Zahl im Code ausgetauscht werden.

Die Variablendefinition schreibst du direkt am Anfang des Codes, also noch vor dem `setup()`-Teil.
Zum Beispiel:



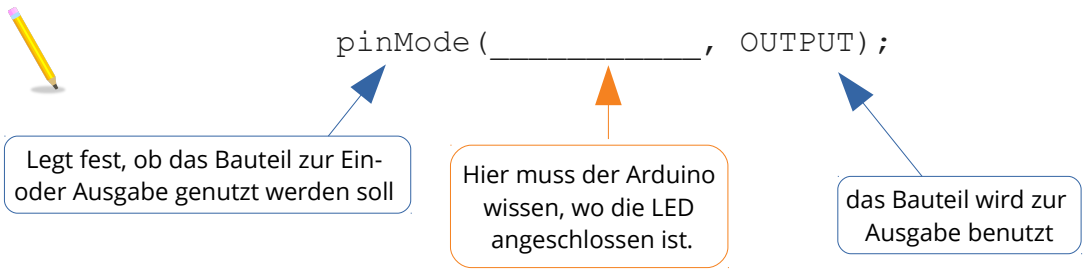
Das `int` bedeutet, dass unter dem Namen roteLED `eine Zahl` gespeichert werden soll. `roteLED` ist der Variablenname. Mit der Zahl 6 gibst du an, dass die LED an Pin 6 angeschlossen ist. Immer dann wenn der Arduino im Code wissen muss, wo die LED angeschlossen ist, kannst du ab jetzt `roteLED` schreiben.

GRAMMATIK VON PROGRAMMIERSPRACHEN

Wie in unserer Sprache, so hat auch eine Programmiersprache eine eigene Grammatik. Es ist wichtig die Regeln zu befolgen, sonst versteht dich der Computer vielleicht nicht. In der Programmiersprache des Arduino ist es wichtig, dass du nach jedem Befehl ein Semikolon schreibst. Sollte dir ein Fehler angezeigt werden, liegt es oft daran, dass du das Semikolon vergessen hast.

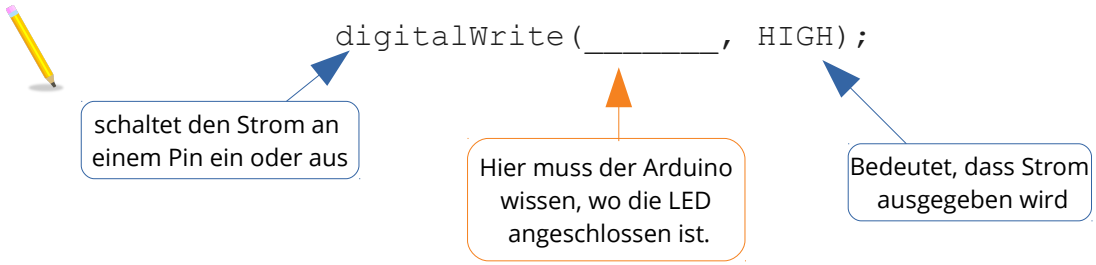
3. Digitalen Pin verbinden

   Der Arduino muss wissen, ob an einem Pin etwas angeschlossen ist, das angeschaltet werden soll. Damit der Arduino das weiß, musst du ihm dies mit den Befehlen `pinMode` und `OUTPUT` mitteilen.



4. LED anschalten

   Jetzt muss die LED noch angeschaltet werden. Der Befehl dafür wird in den `loop()`- Teil geschrieben. Er lautet:



5. Programm installieren

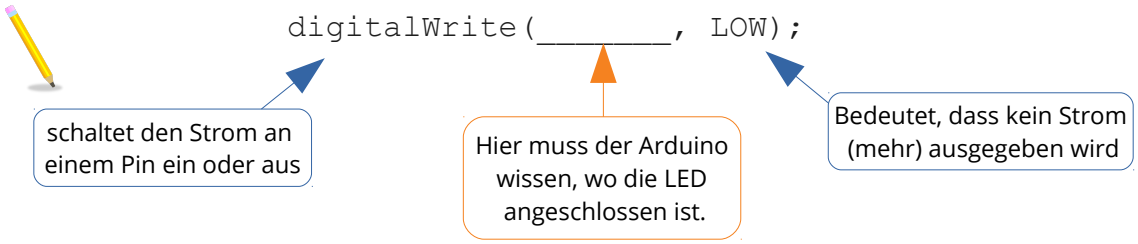
   Installiere das fertige Programm auf den Arduino. Klicke dafür auf den Übertragen-Button. Die LED sollte jetzt leuchten!

AUFGABE 4 – LASSE DIE LED BLINKEN

Ändere deinen Code, damit die LED blinkt. Füge die nötigen Befehle an der richtigen Stelle im Code hinzu. Installiere dein Programm auf dem Arduino und teste, ob die LED blinkt.

1. Ausschalten

   Um eine blinkende LED zu programmieren, müssen wir sie an- und wieder ausschalten. Und dies muss dann immer wiederholt werden. In unserem Code wird die LED bisher nur angeschaltet. Aber auch den Befehl zum Ausschalten kennst du schon:



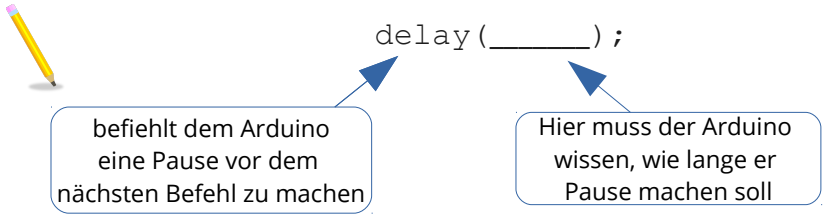
   Nach welchem Befehl muss das Ausschalten passieren?

   _____

Was stellst du fest, nachdem du den Befehl in deinen Code geschrieben hast und das Programm installiert hast?

Richtig! Irgendwie blinkt die LED gar nicht…
Schuld daran ist die Schnelligkeit des Arduino. Die LED wird sofort nach dem Anschalten wieder ausgeschaltet. Das menschliche Auge kann das Anschalten nicht sehen.

2. Verzögerung
   Deswegen musst du den Arduino dazu bringen, dass er eine Pause zwischen dem Ein- und Ausschalten macht. Dazu brauchen wir einen Befehl:



Welche Zahl gehört in die Klammer?

Der Arduino macht immer so lange Pause, wie du ihm in der Klammer befiehlst. Die Zeit misst er in Millisekunden. Füge die passenden Zahlen in die Klammern ein. Der Arduino soll Pause machen für:

- Eine Sekunde `delay(_____);`
- Zehn Sekunden `delay(_____);`
- Eine halbe Sekunde `delay(_____);`

Überlege auch hier, an welchen Stellen der Befehl eingefügt werden muss. Die Pause sollte etwa eine Sekunde lang sein.

AUFGABE 5 - WAHRNEHMUNG

Wie lange kann das Auge noch sehen? Verkleinere die Zahl im `delay(1000)` – Befehl solange, bis die LED nicht mehr blinkt. Erhöhe dann die Zahl langsam wieder, bis du das Blinken wieder sehen kannst. So bekommst du den genauen Wert heraus. Ab welcher Zahl können deine Augen das Blinken nicht mehr sehen? Schreibe dein Ergebnis hier auf:

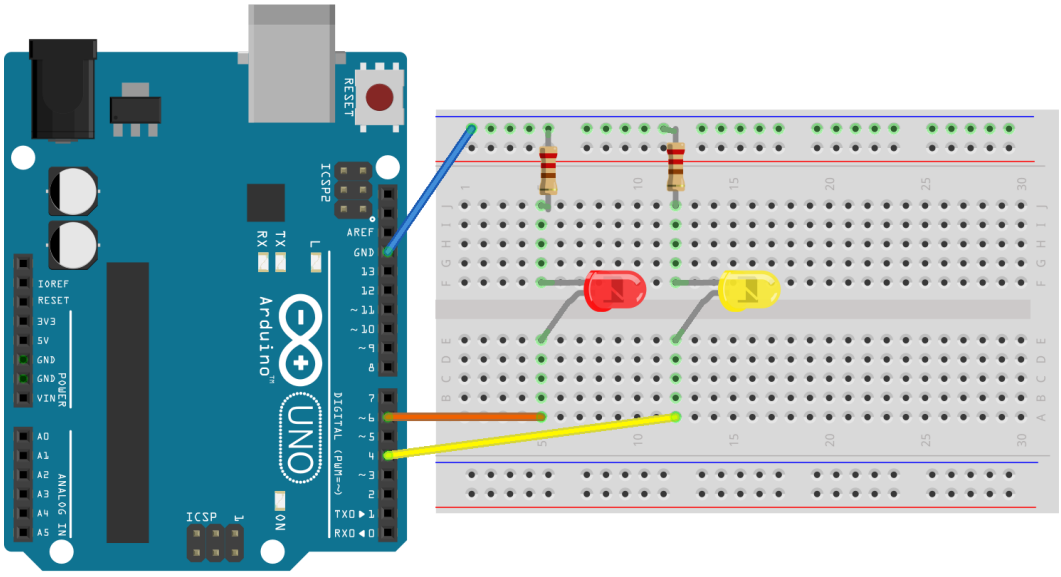
_____

AUFGABE 6 – WARNBLINKLICHT

Als nächstes wollen wir ein Warnlicht programmieren, wie man es von einer Baustelle kennt. Es besteht aus zwei Lichtern, die abwechselnd blinken. Folge dafür den folgenden Schritten.

distelAPPArath, pixabay.com , Pixabay Lizenz

1. Die Schaltung abändern
   Verbinde eine zweite LED mit dem Minuspol und einem digitalen Pin. Vergiss dabei nicht den Widerstand!



2. Variable definieren und digitalen Pin verbinden
   Auch für die gelbe LED muss eine Variable definiert werden. Du kannst sie `gelbeLED` nennen. Auch der digitale Pin der gelben LED soll Strom geben. Für Näheres schau dir noch einmal Aufgabe 3 Punkt 2 und 3 an, oder deinen geschriebenen Code.

3. Absprechen der LEDs
   Die rote LED geht bisher automatisch an und aus. Jetzt aber sollen sich die rote und die gelbe LED aber „absprechen“ und nacheinander leuchten. Damit sie wissen, wer an der Reihe ist, gibt es eine Art Schiedsrichter. Der Schiedsrichter wird durch eine Variable programmiert. Diese muss definiert werden. Zum Beispiel:

```
int anderreihe = roteLED;
```

   Wir legen dadurch fest, dass die rote LED zuerst an der Reihe ist.

Wie muss der Befehl später aussehen, wenn die gelbe LED an der Reihe ist?

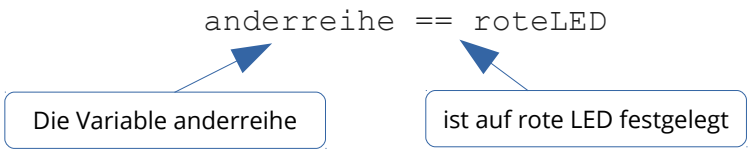
```
int anderreihe = __________;
```

4. Wenn und sonst
   Wenn die rote LED an der Reihe ist, soll die rote LED leuchten. **Sonst** soll die gelbe LED leuchten.
   Beim Programmieren gibt es dafür diesen Ausdruck:

```
if ( roteLED ist an der Reihe ){
roteLED leuchtet ; }
else {
gelbeLED leuchtet;}
```

   Der Ausdruck gehört in den `loop` - Teil. Achte auch hier unbedingt auf die Grammatik und verwende die richtigen Klammern!

5. Befehle
   Den Text in den Klammern kann der Arduino natürlich noch nicht verstehen. Wir müssen ihn noch abändern.
   Die rote LED ist an der Reihe:



Die rote LED haben wir schon zum Leuchten gebracht. Auch hier soll die LED eine Sekunde lang leuchten und sich dann wieder ausschalten. Die gleichen Befehle kannst du auch benutzen, um die gelbe LED eine Sekunde lang leuchten zu lassen.

Bringe die Codeteile in die richtige Reihenfolge

```
if(naechsteLED == _____) {
      _____
      _____
      _____
}
else {
      _____
      _____
      _____
}
```

A `digitalWrite(roteLED, LOW);`

B `roteLED`

C `digitalWrite(gelbeLED, LOW);`

D `delay(1000);`

E `digitalWrite(gelbeLED, HIGH);`

F `delay(1000);`

G `digitalWrite(roteLED, HIGH);`

Welche LEDs leuchten, nachdem du den Code eingefügt hast?

_____

6. <u>Schiedsrichterentscheidung</u>
   Dass nur die rote LED leuchtet liegt daran, dass der Schiedsrichter nie angesagt hat, dass die gelbe LED an der Reihe ist zu leuchten. Füge deswegen nach dem Blinken der roten LED einen Befehl ein, der sagt, dass die gelbe LED an der Reihe ist. Bedenke auch, dass der Schiedsrichter auch ansagen muss, wann die rote LED wieder leuchten soll. Füge hierfür einen weiteren Befehl nach dem Leuchten der gelben LED ein.

DU HAST NOCH NICHT GENUG VON DIESER STATION?

Kein Problem - Frage die Betreuer einfach nach dem Zusatzblatt! Dort wirst du das Warnlicht zu einer Ampel umprogrammiern.

Screenshots: *fritzing electronics made by easy und Arduino IDE 1.8.12 (windows)*
Alle weiteren Grafiken: *Patrick Binkert,* EduInf@TUD