DILIN
Didaktiklabor Informatik

Anleitung
Arduino-Experimentiersets

Ziel dieses Dokuments ist es, die selbstständige Nutzung und Erprobung der Arduino-Experimentiersets durch Erläuterungen, praktische Beispiele und die Angabe weitere Informationsquellen zu erleichtern.

Einsatzmöglichkeiten

Der Schwerpunkt kann auf der Realisierung von Schaltungen unterschiedlicher Komplexität (mit Sensoren und Aktoren) oder auf der Programmierung des Mikrocontrollers liegen. Insbesondere sind größere, fächerübergreifende Projekte an der Schnittstelle zwischen Physik und Informatik machbar. Auf Seiten der Informatik können insbesondere folgende Bereiche bzw. Kompetenzen angesprochen werden:

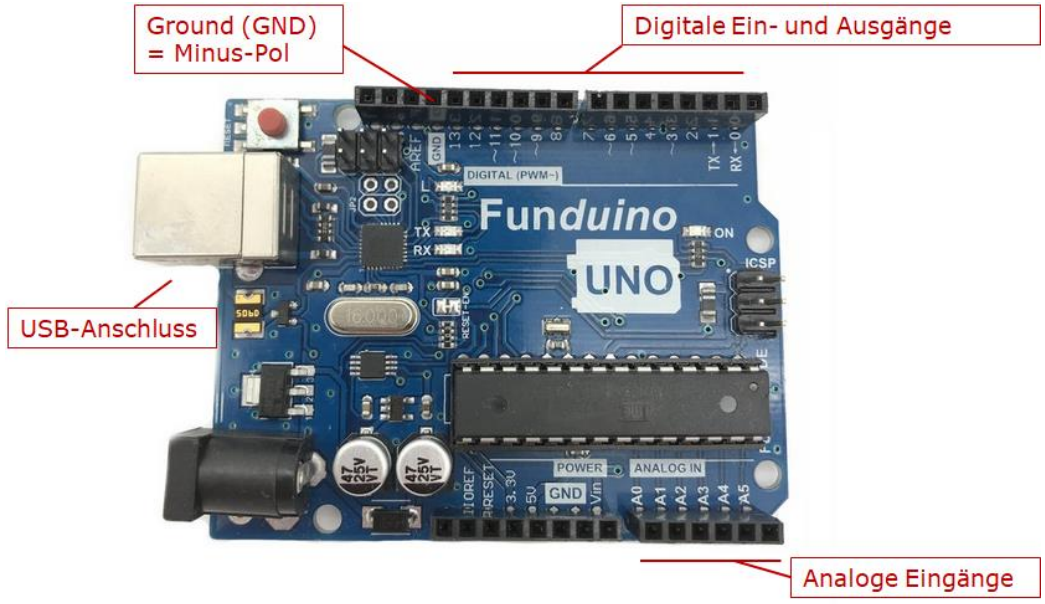
- Modellieren unter Verwendung von Soft- *und* Hardwarekomponenten
- Verstehen, Entwickeln, Implementieren, Testen und Debuggen von Algorithmen
- Verständnis von reaktiven/eingebetteten Informatiksystemen
- Technische Grundlagen informationsverarbeitender Systeme

Informatiksystem Arduino

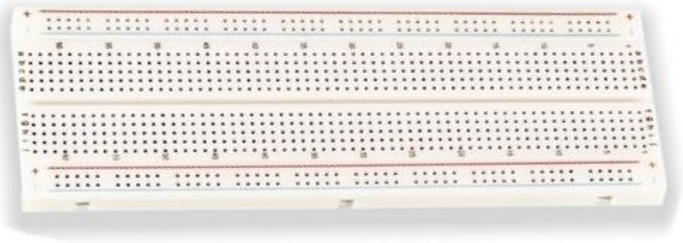
Die erforderliche ***Hardware*** besteht aus der Platine mit Mikrocontroller, Anschlusskabeln und weiterem Zubehör zum Aufbau von Schaltungen sowie zum Anschließen von Sensoren (z.B. Taster) bzw. Aktoren (z.B. LEDs). Softwareseitig können verschiedene ***Entwicklungsumgebungen*** genutzt werden, in diesem Dokument verwenden wir die unter https://www.arduino.cc/ kostenlos zur Verfügung gestellte IDE. Programmiert wird dort in C++. SchülerInnen, die Java beherrschen, kommen mit der Syntax gut zurecht.

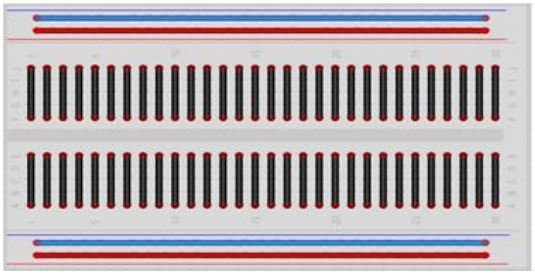
Hardware *(wichtigste Komponenten)***:**

- **Platine mit Mikrocontroller**



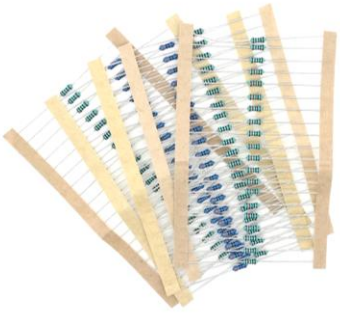
- **Steckbrett**





In dieser schematischen Darstellung sind leitende Verbindungen zwischen den Kontakten des Steckbretts markiert.

- **LEDs und Widerstände**

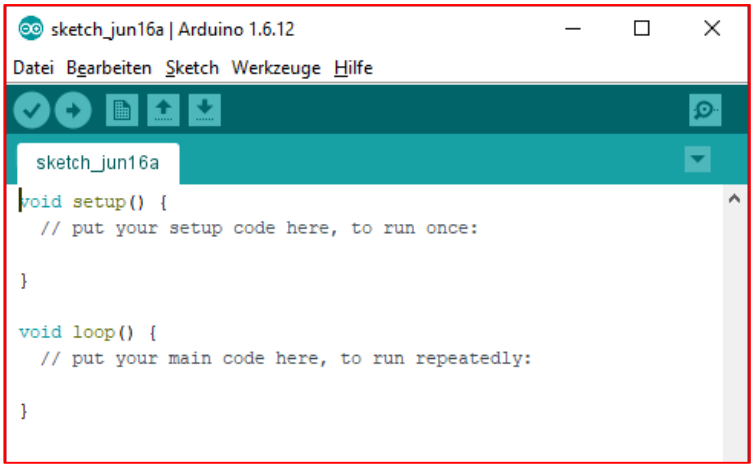


Die Widerstandswerte sind durch Folgen farbiger Ringe codiert.
Bei der Verwendung von LEDs immer auf den Einbau eines 200-Ohm-Widerstands (Farbcodierung: *rot-schwarz-schwarz-schwarz-braun*) Widerstands achten. Ohne entsprechenden Widerstand wird die LED zerstört.

Wird die LED versehentlich in falscher Richtung eingebaut (längeres Bein zum Minus-Pol), so ist dies unproblematisch– sie leuchtet dann nicht („Sperrrichtung“), es geht aber nichts kaputt.

Software:

Die hier dargestellte IDE kann unter arduino.cc kostenfrei heruntergeladen werden.



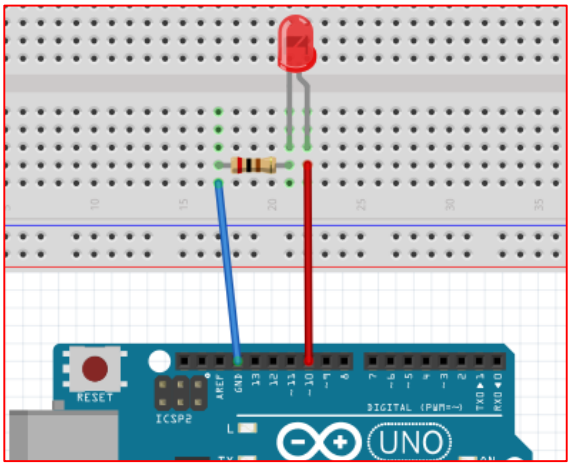
Programme werden hier als „Sketches“ bezeichnet. Code, der in den setup-Block geschrieben wird, wird am Anfang einmal ausgeführt. Der Code im loop-Block wird in Endlosschleife wiederholt.

Beispielprojekt:

- **Aufgabenstellung**:
  Realisierung einer Schaltung, in der eine rote LED im Sekundentakt blinkt.

- **Schaltung** und **Programmcode**

  Die LED wird über einen 200-Ohm-Widerstand mit GND (Minus-Pol) und dem Pin 10 verbunden, der hier die Rolle des Pluspols übernimmt.



Pin 10 wird zunächst als Output-Pin deklariert. Dann wird wiederholend eine Sekunde Spannung auf den Pin gegeben und für eine Sekunde wieder weggenommen.

(Bedeutung der Befehle: siehe unten)

Blinkende_LED

```
void setup() {
  pinMode(10, OUTPUT);
}

void loop() {
  digitalWrite(10, HIGH);
  delay(1000);
  digitalWrite(10, LOW);
  delay(1000);
}
```

Erforderliche Grundkenntnisse:

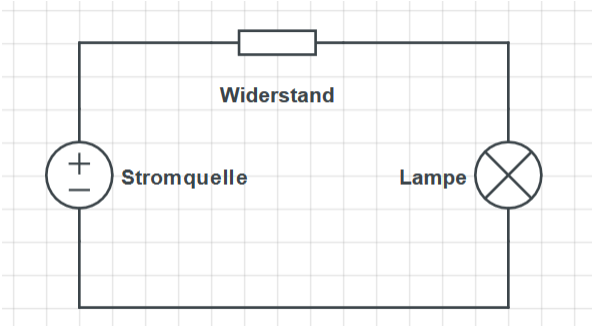
Zur erfolgreichen Nutzung des Informatiksystems sind grundlegende physikalische Kenntnisse über Aufbau und Kenngrößen eines Stromkreislaufs erforderlich. Eine Gesundheitsgefährung ist jedoch in jedem Fall ausgeschlossen, da die maximale Spannung von 5V für Menschen ungefährlich ist.
Für die Programmierung werden am Anfang – abgesehen von „normalen“ Sprachstrukturen – nur wenige spezielle Befehle benötigt, die nachfolgend aufgelistet werden.

Grundlegende Befehle

Befehl	Bedeutung
`pinMode(<pin-Name>, <Pin-Typ>)`	definiert einen Anschluss des Arduino als INPUT bzw. OUTPUT
`digitalWrite(<pin-Name>, <pin-Zustand>)`	gibt Strom (HIGH / 1) bzw. keinen Strom (LOW / 0) auf das angegebene Pin
`delay(<Zeitdauer>)`	verzögert die Programmausführung um die angegebene Zeit in Millisekunden
`analogRead(<pin-Name>)`	Liest einen Wert am analogen Pin `<pin-Name>` ein [Wertebereich: 0 bis 1023]
`pulseIn(<pin-Name>, HIGH)`	Misst die Zeit (in Mikrosekunden), bis am entsprechenden Pin ein Wechsel von HIGH zu LOW registriert wird.

Aufbau eines Stromkreislaufs

Ein Stromkreislauf besteht in der Regel aus einer Stromquelle, einem (oder mehreren) Verbraucher(n), z.B. Lampen, ggf. zusätzlichen Widerständen und Draht, der die Objekte miteinander verbindet.

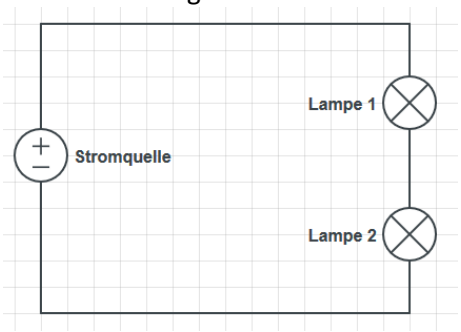


Kenngrößen eines Stromkreislaufs

- Stromstärke I (in **A**mpere)
- Spannung U (in **V**olt)
- Widerstand R (in **Ohm**)
- Zusammenhang zwischen diesen Größen (*ohmsches Gesetz*): R * I = U

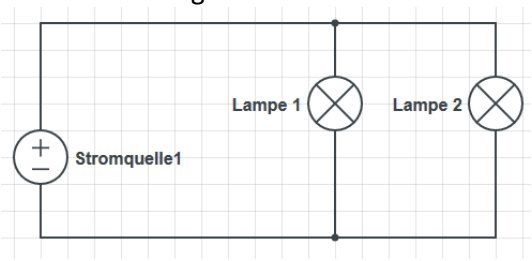
Besondere Schaltungsformen

Reihenschaltung



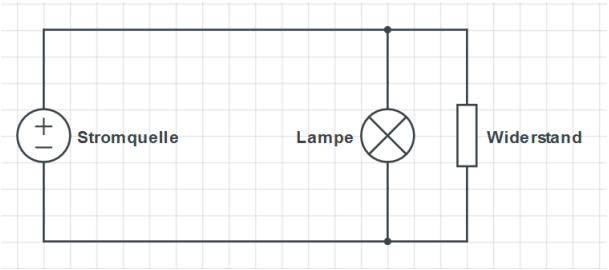
$I_{ges} = I_1 = I_2$
$U_{ges} = U_1 + U_2$

Parallelschaltung



$I_{ges} = I_1 + I_2$
$U_{ges} = U_1 = U_2$

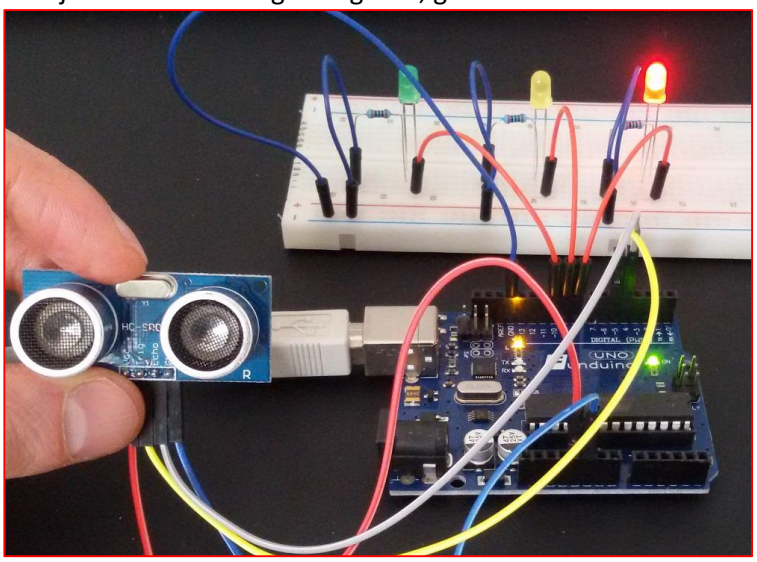
!! Vorsicht !!



Ein zum Verbraucher **parallel** geschalteter Widerstand reduziert nicht die am Verbraucher anliegende Spannung (s. Parallelschaltung). LED und Widerstand daher immer in Reihe schalten!

Beispielhafte Folgeprojekte:

- Simulation des **Blaulichts** eines Polizeifahrzeuges mithilfe von 2 LEDs.
  - Erweiterung: Dauer des Aufleuchtens der LEDs soll von außen (mithilfe eines Drehreglers/Potentiometers) steuerbar sein.
  - Modifikation: Abhängig von der Position eines Drehreglers soll die erste, die zweite oder gar keine LED leuchtet.
- Simulation eines **Abstandswarners**: Mithilfe des Ultraschallsensors soll der Abstand von Objekten ermittelt werden und je nach Entfernung eine grüne, gelbe oder rote LED leuchten. (siehe folgende Abbildung)



Ausstattung des Didaktiklabors

- 13 Sets mit Grundausrüstung
  Dazu zählen u.a.:
  - Mikrocontroller, USB-Kabel, Steckbrett, Anschlusskabel, Widerstände
  - **Aktoren**: LEDs, RGB-LED, Piezospeaker
  - **Sensoren**: Fotowiderstand, Taster, Drehpotentiometer, Ultraschallsensor, Neigungssensor

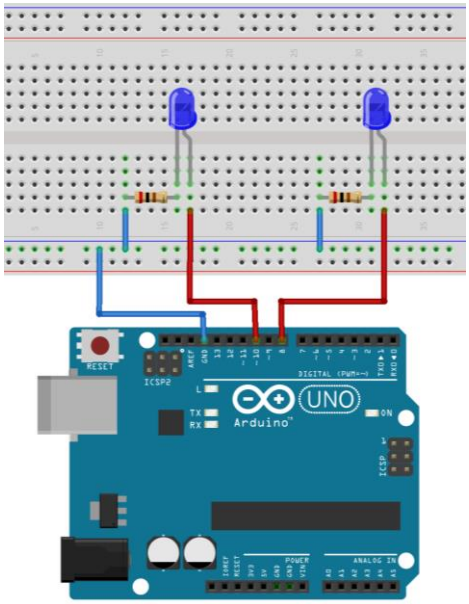
- Umfangreiches, sortiertes Spezialzubehör

Weitere Informationen (Spannende Projekte, Anleitungen zu Bauteilen)

- *Funduino*: https://funduino.de/anleitungen
- Erik Bartmann: *„Die elektronische Welt mit Arduino entdecken“* (in Campusbibliothek verfügbar)
- Schülerlabor RWTH Aachen: http://schuelerlabor.informatik.rwth-aachen.de/modulmaterialien/informatik-enlightened

Lösungen zu den Beispielprojekten

- Simulation Blaulicht

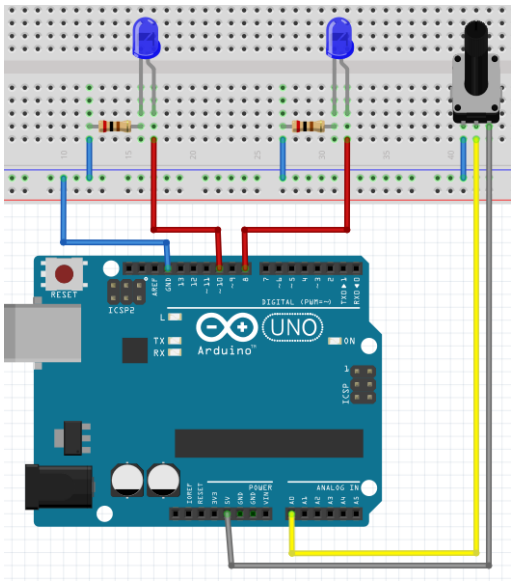


Blinken:

```
int led1 = 10;
int led2 = 8;

void setup() {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
}

void loop() {
  digitalWrite(led1, HIGH);
  digitalWrite(led2, LOW);
  delay(200);
  digitalWrite(led1, LOW);
  digitalWrite(led2, HIGH);
  delay(200);
}
```



Mit Drehregler – Variante 1:

```
int led1 = 10; int led2 = 8;
int sensorwert = 0;

void setup() {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
}

void loop() {
  sensorwert = analogRead(A0);
  digitalWrite(led1, HIGH);
  digitalWrite(led2, LOW);
  delay(sensorwert);
  digitalWrite(led1, LOW);
  digitalWrite(led2, HIGH);
  delay(sensorwert);
}
```

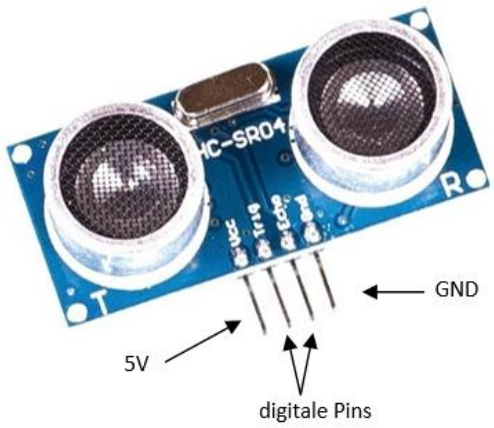
Variante 2:

```
int led1 = 10; int led2 = 8;
int sensor = 0;

void setup() {
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
}

void loop() {
  sensor = analogRead(A0);
  if (sensor < 200) {
    digitalWrite(led1, HIGH);
    digitalWrite(led2, LOW);
    delay(500);}
  if (sensor > 800) {
    digitalWrite(led1, LOW);
    digitalWrite(led2, HIGH);
    delay(500); }
  if (sensor >= 200 && sensor <= 800) {
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    delay(500);
  }
}
```

- Simulation Abstandswarner:




```
int ledGruen = 10;
int ledGelb = 9;
int ledRot = 8;
int trigger = 5;
int echo = 4;

long dauer = 0;
long entfernung = 0;

void setup() {
  pinMode(ledGruen, OUTPUT);
  pinMode(ledGelb, OUTPUT);
  pinMode(ledRot, OUTPUT);
  pinMode(trigger, OUTPUT);
  pinMode(echo, INPUT);
}

void loop() {
  digitalWrite(trigger, LOW);
  delay(100);
  digitalWrite(trigger, HIGH);
  delay(10);
  digitalWrite(trigger, LOW);
  dauer = pulseIn(echo, HIGH);
  entfernung = (dauer/2) * 0.03432;

  if (entfernung < 50) {
    digitalWrite(ledGruen, LOW);
    digitalWrite(ledGelb, LOW);
    digitalWrite(ledRot, HIGH);
    delay(500);
  }
  if (entfernung > 200) {
    digitalWrite(ledGruen, HIGH);
    digitalWrite(ledGelb, LOW);
    digitalWrite(ledRot, LOW);
    delay(500);
  }
  if (entfernung >= 50 && entfernung <=
200) {
    digitalWrite(ledGruen, LOW);
    digitalWrite(ledGelb, HIGH);
    digitalWrite(ledRot, LOW);
    delay(500);
  }
}
```

Alle schematischen Schaltungsdarstellungen wurden mit dem kostenlos Tool FRITZING erstellt.

Autor: Pascal Schmidt
pascal.schmidt@uni-saarland.de

Stand: 07.09.2018

GEFÖRDERT VOM
Bundesministerium für Bildung und Forschung
SaLUt Saarland