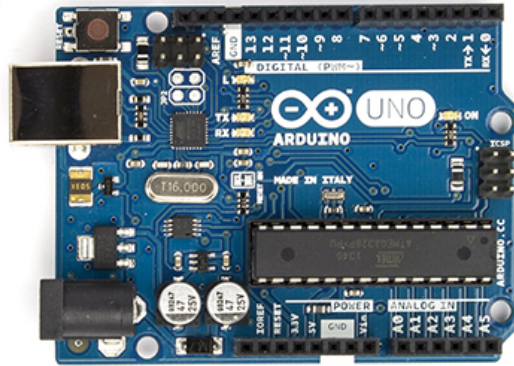


# Arduino I – Einführungskurs



28. April 2018

Christian Walther  
Matthias Hofer

# Agenda

Was ist ein Mikrocontroller, was ist Arduino?

Loslegen mit Software-Installation

Refresher Elektrizitätslehre

Digitale und analoge Inputs und Outputs

Kommunikation mit dem Computer

Töne erzeugen, Servo ansteuern

Ausblick

# Was ist ein Mikrocontroller?

Einsatz von programmierbaren digitalen Rechnern:

## **Personal Computer oder Mobilgerät**

Desktop, Laptop, Spielkonsole, Tablet,  
Smartphone

Allzweck-Gerät  
viel Rechenleistung und Speicherplatz

## **Mikroprozessor**

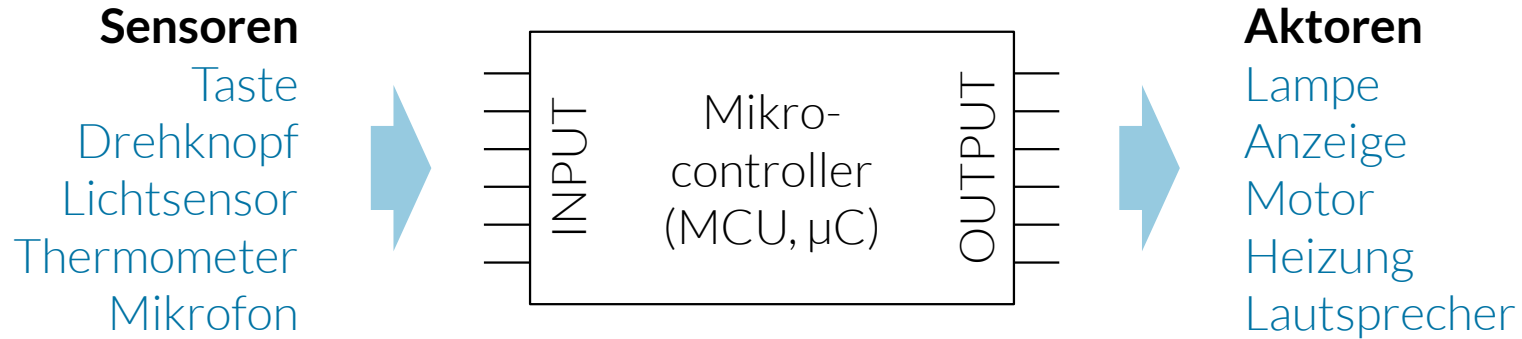
## **Embedded-System**

Digitaluhr, Kaffeemaschine,  
Verkehrsampel, Zahnbürste

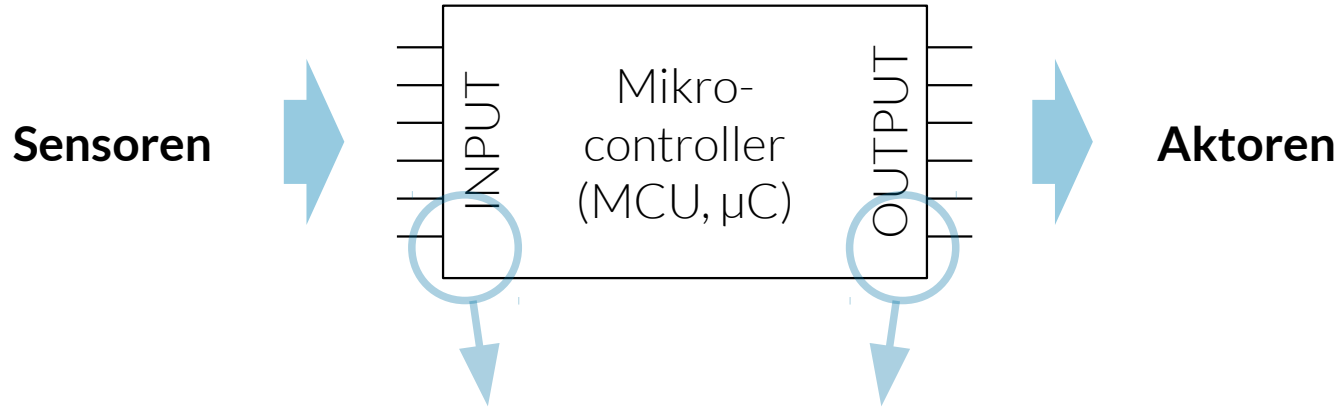
optimiert für einen Zweck  
klein und billig  
alles nötige in einem Baustein

## **Mikrocontroller**

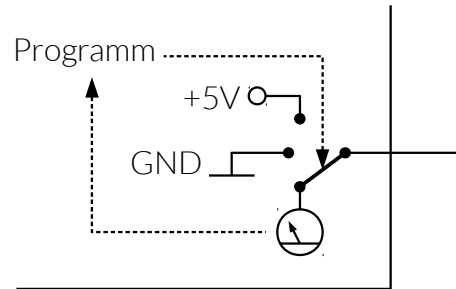
# Mikrocontroller interagieren mit der Welt



# Mikrocontroller interagieren mit der Welt



**GPIO Pin**  
General Purpose  
Input/Output



# Was ist Arduino?

Ein Projekt mit dem Ziel, Bastlern, Künstlern und anderen Nicht-Ingenieuren den Einsatz von Mikrocontrollern zu ermöglichen.

## **Produkte** (open source)

- Boards
- Entwicklungsumgebung (IDE, Integrated Development Environment)

## **Community**

- Foren, Tutorials etc.
  - <http://forum.arduino.cc>
  - <http://playground.arduino.cc>
- Software-Erweiterungen: Libraries
- Hardware-Erweiterungen: Shields

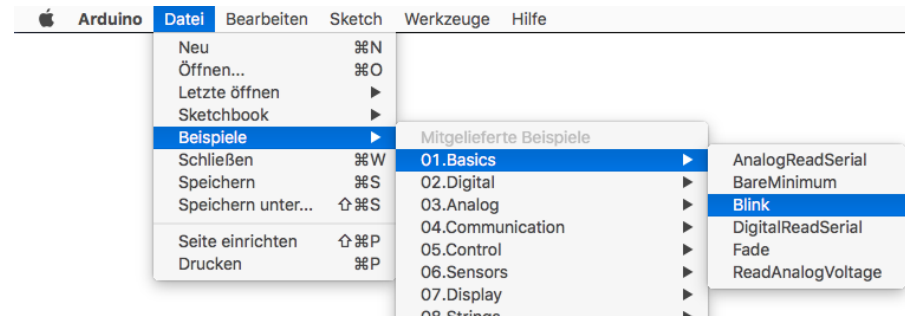
# Entwicklungsumgebung (IDE) installieren

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

Projektbuch S. 16

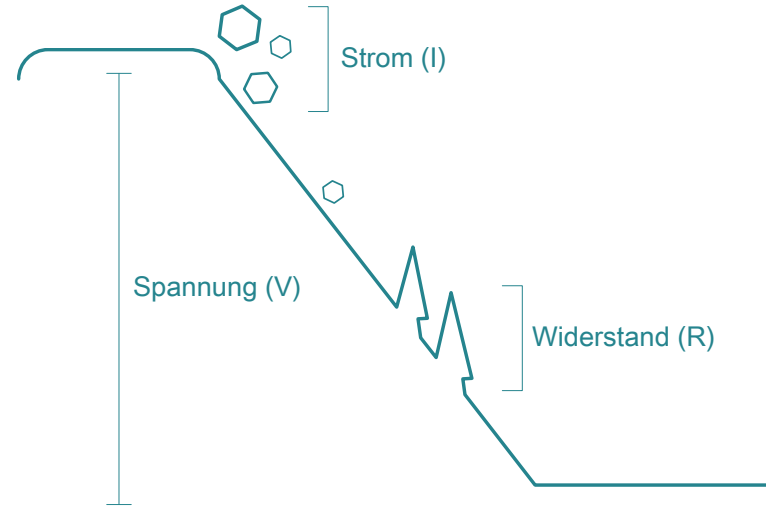
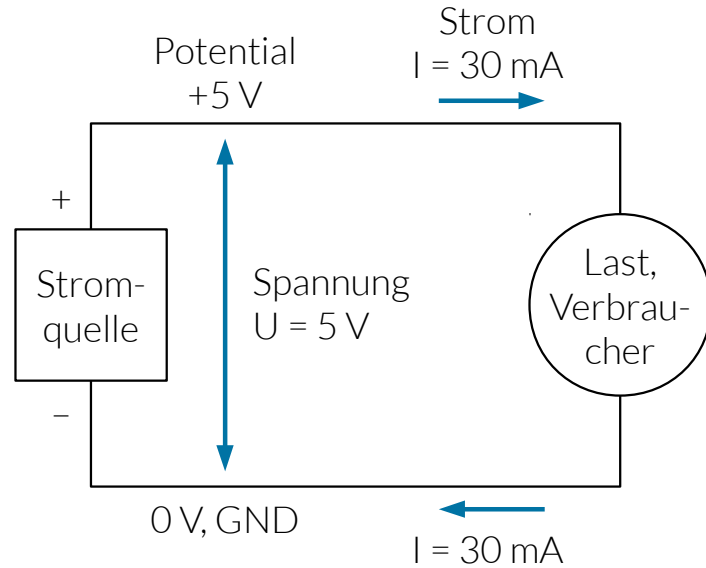
Sprache umschalten: File ► Preferences (Windows)  
Arduino ► Preferences (Mac)

## Beispielprogramm «Blink» laden



Projektbuch S. 18

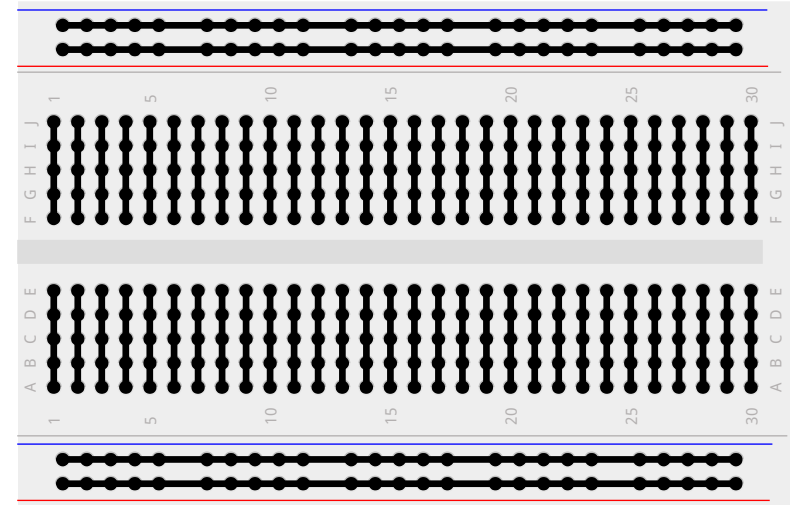
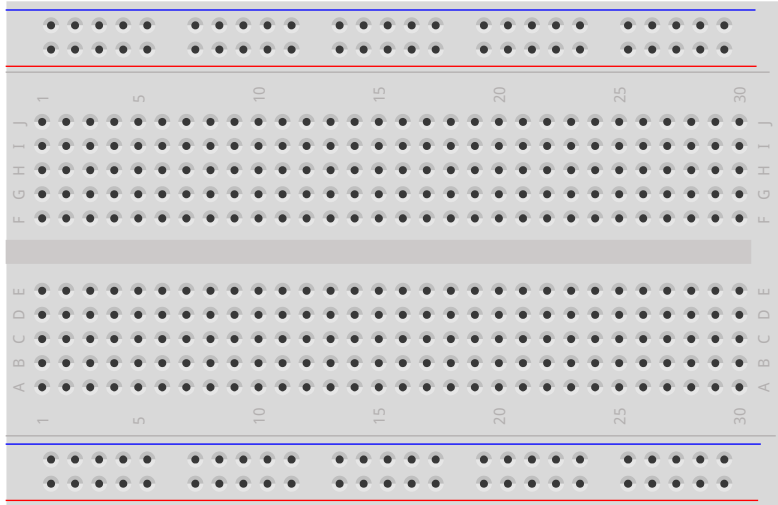
# Stromkreise



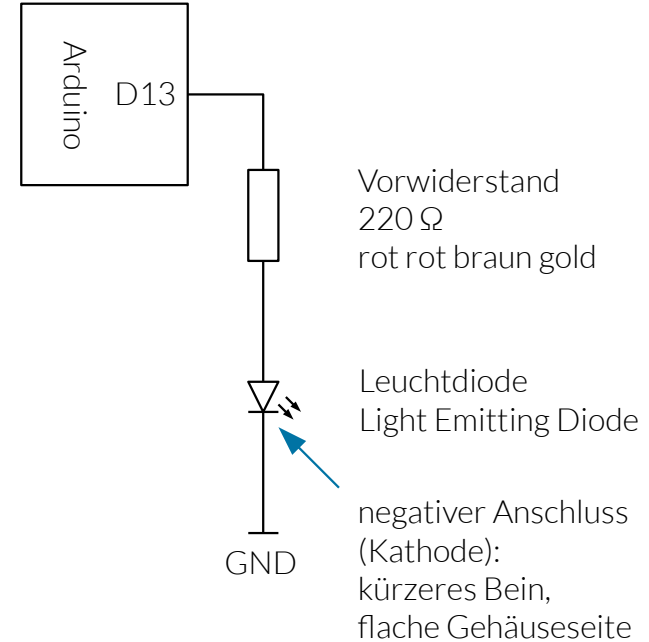
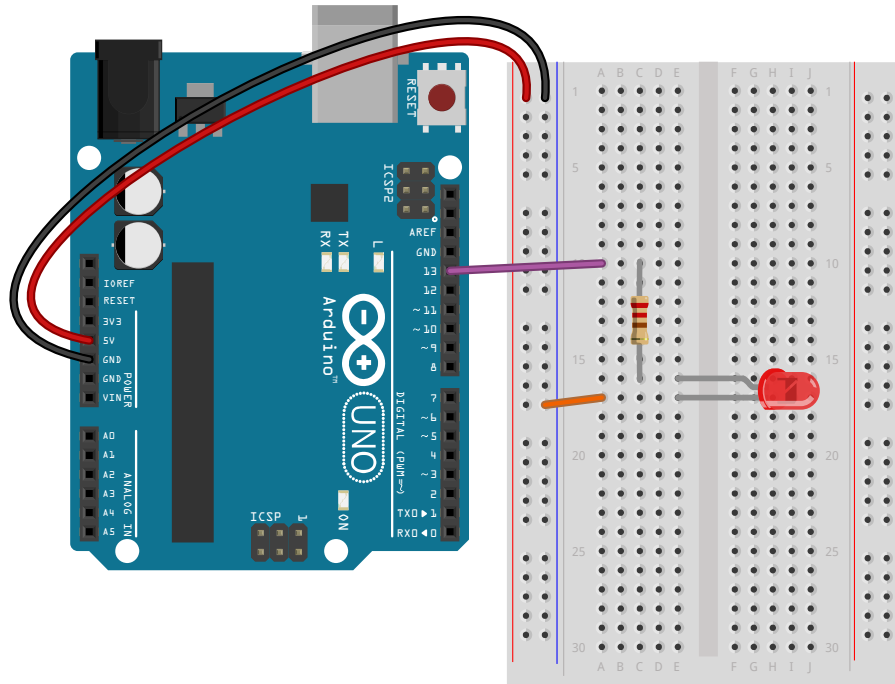
Projektbuch S. 21



# Steckbrett (Breadboard)

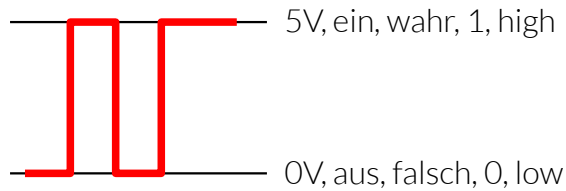


# Blink mit externer LED: Digitaler Output

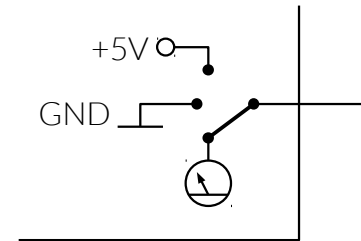
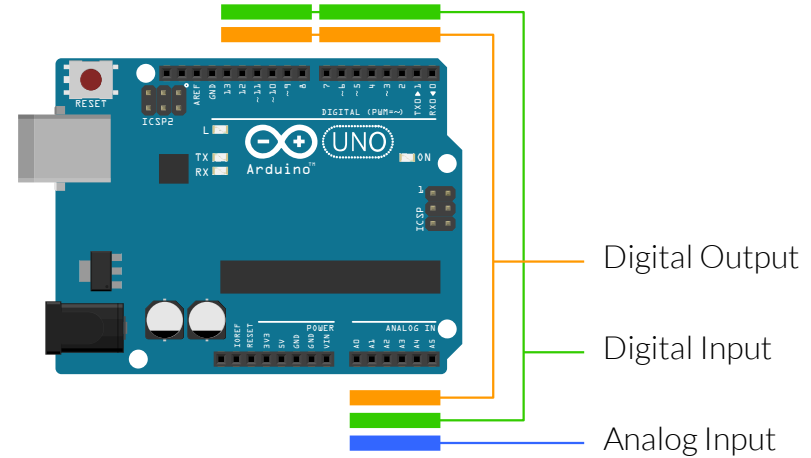
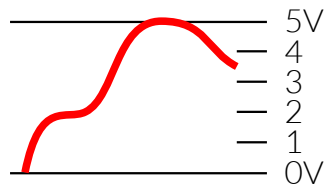


# Digital und Analog

## Digitales Signal



## Analoges Signal



# Programmstruktur

Funktion → `void setup() {`  
// the setup function runs once when you press reset or power the board  
// initialize digital pin LED\_BUILTIN as an output.  
`pinMode(LED_BUILTIN, OUTPUT);`  
`}`

Kommentar → `// the loop function runs over and over again forever`

Funktionsaufruf → `void loop() {`  
`digitalWrite(LED_BUILTIN, HIGH);` // turn the LED on (HIGH is the voltage level)  
`delay(1000);` // wait for a second  
`digitalWrite(LED_BUILTIN, LOW);` // turn the LED off by making the voltage LOW  
`delay(1000);` // wait for a second  
`}`

`{}` fasst Befehle zu Block zusammen

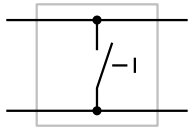
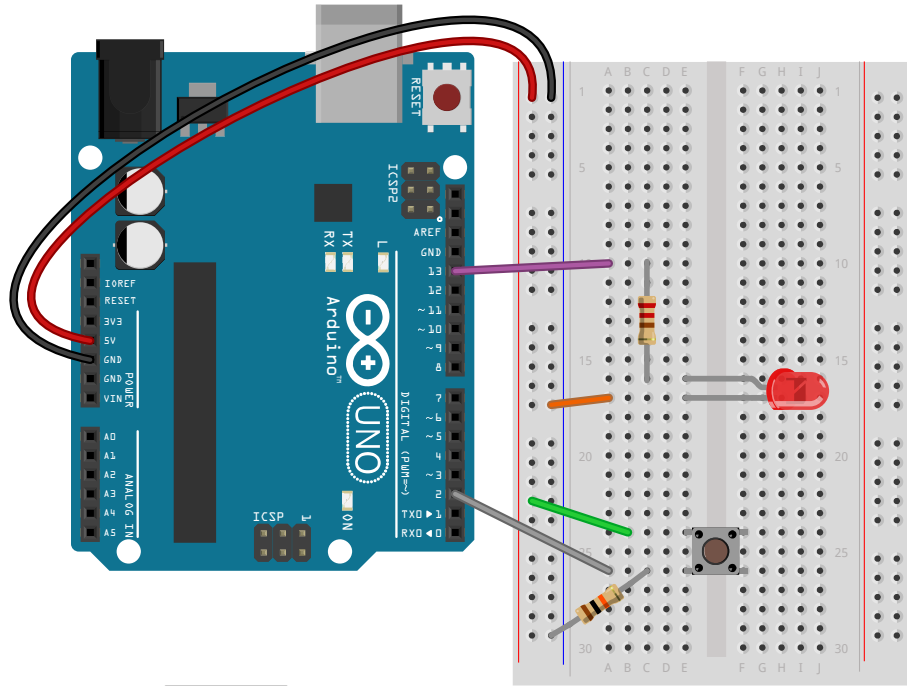
`;` schliesst Befehl ab

Leerzeichen und Einrückung sind optional

<https://www.arduino.cc/en/Reference>

Projektbuch S. 36

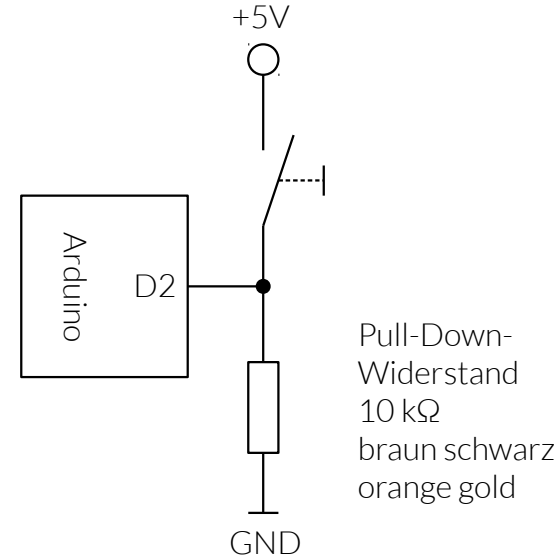
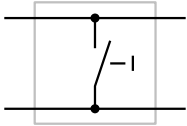
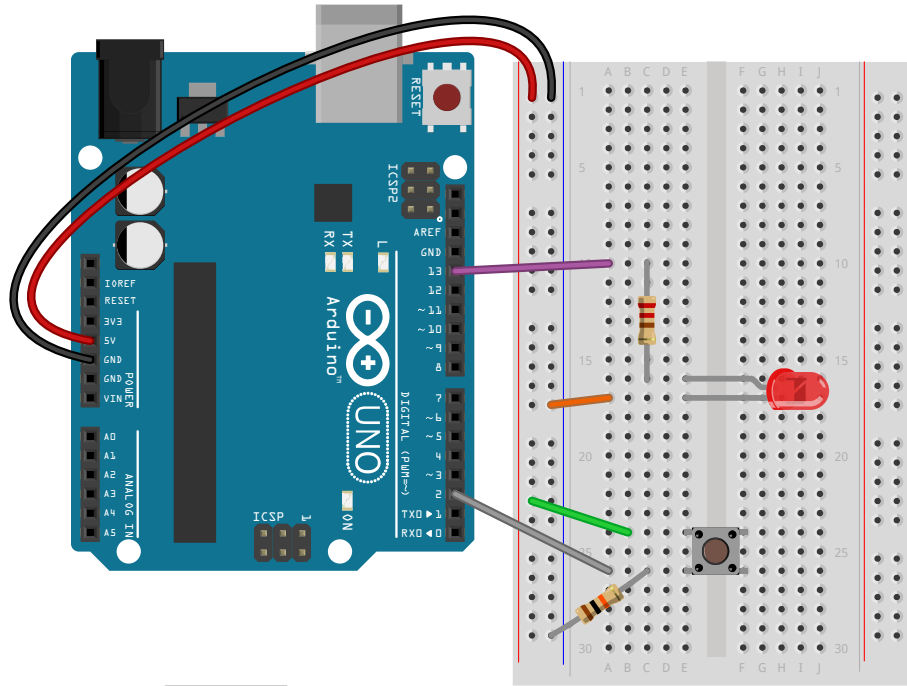
# Digitaler Input



braun schwarz orange gold (10 kΩ)

```
void setup() {  
  pinMode(2, INPUT);  
  pinMode(13, OUTPUT);  
}  
  
void loop() {  
  int switchState = digitalRead(2);  
  if (switchState == HIGH) {  
    digitalWrite(13, HIGH);  
    delay(200);  
    digitalWrite(13, LOW);  
    delay(200);  
  }  
  else {  
    digitalWrite(13, LOW);  
  }  
}
```

# Digitaler Input



Input-Pin darf nicht unverbunden sein, sonst liest er undefinierte Werte. Projektbuch S. 35

# Digitaler Input

Variable  
Datentyp: `int` = Ganzzahl  
gültig innerhalb des nächstäußeren `{}`-Blocks  
=: Zuweisung

Entscheidung  
wenn/dann/sonst  
==: Vergleich

Gross-/Kleinschreibung beachten

```
void setup() {  
  pinMode(2, INPUT);  
  pinMode(13, OUTPUT);  
}  
  
void loop() {  
  int switchState = digitalRead(2);  
  if (switchState == HIGH) {  
    digitalWrite(13, HIGH);  
    delay(200);  
    digitalWrite(13, LOW);  
    delay(200);  
  }  
  else {  
    digitalWrite(13, LOW);  
  }  
}
```

Projektbuch S. 36–38

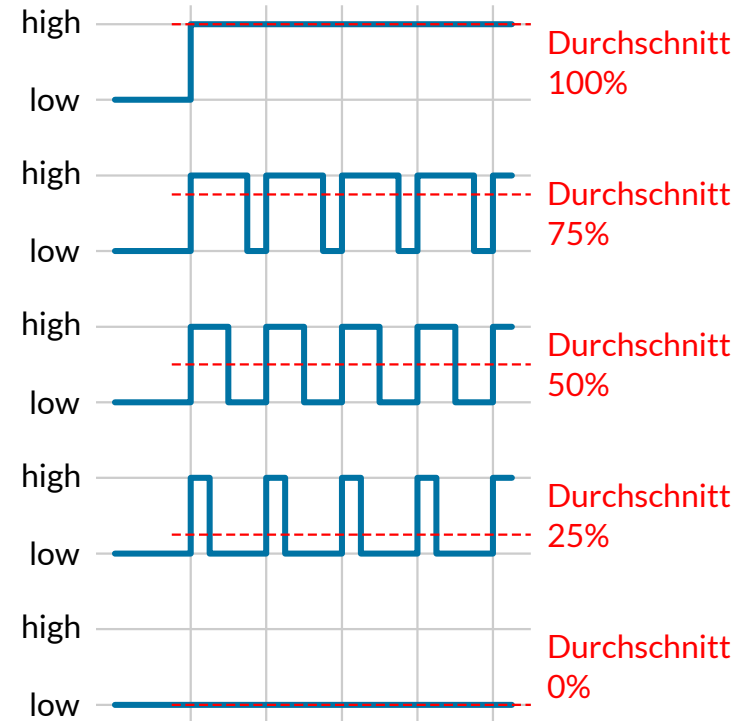
# PWM-Output

Pulse-Width Modulation:  
Spannung zwischen low und high simuliert  
durch schnelles Blinken  
feste Frequenz (~490 Hz), variable Pulsbreite

vom Mikrocontroller in Hardware  
implementiert auf mit ~ markierten Pins

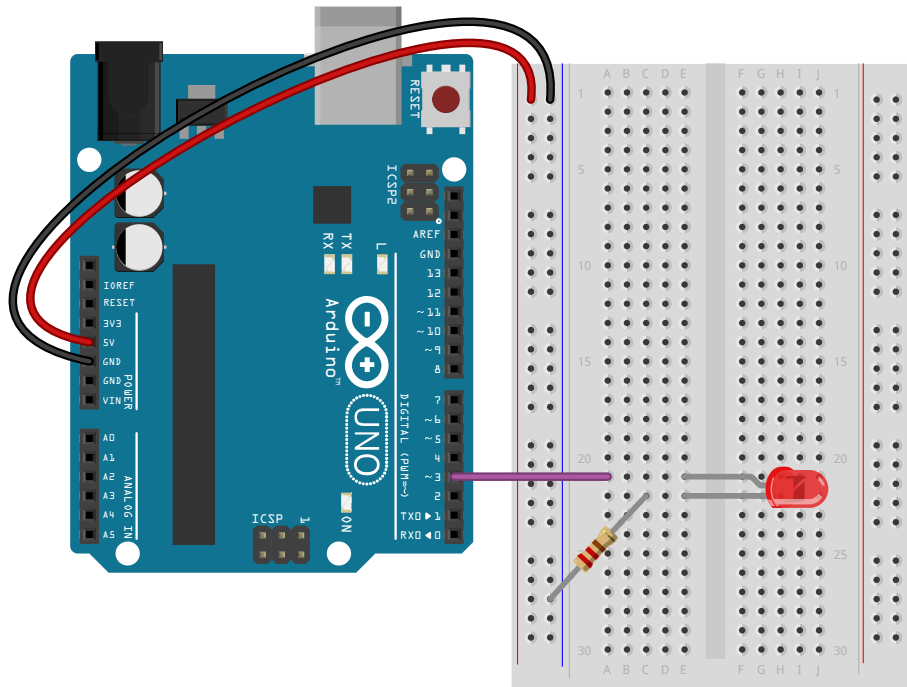
`analogWrite(pin, duty)` nimmt Wert  
zwischen 0 (immer aus) und 255 (immer ein),  
128 = 50%

Projektbuch S. 53



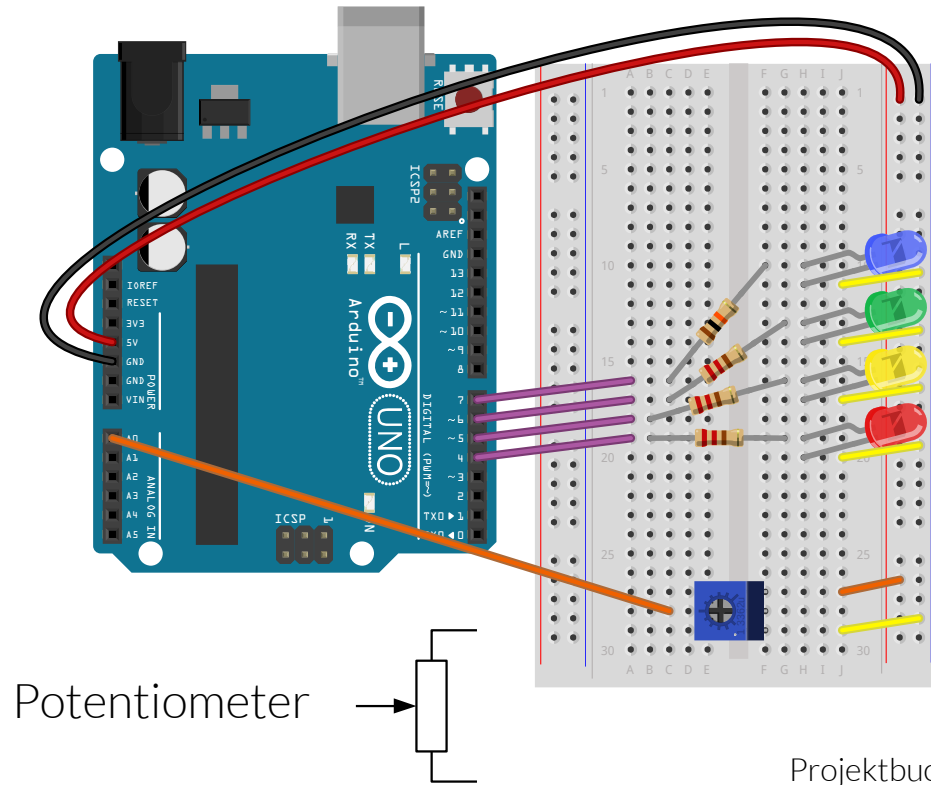


# PWM-Output

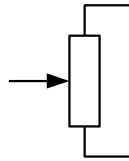


```
void setup() {  
  pinMode(3, OUTPUT);  
}  
  
void loop() {  
  int brightness = 0;  
  while (brightness < 255) {  
    brightness += 51;  
    analogWrite(3, brightness);  
    delay(200);  
  }  
  while (brightness > 0) {  
    brightness -= 51;  
    analogWrite(3, brightness);  
    delay(200);  
  }  
}
```

# Analog Input



Potentiometer



Projektbuch S. 43, 65

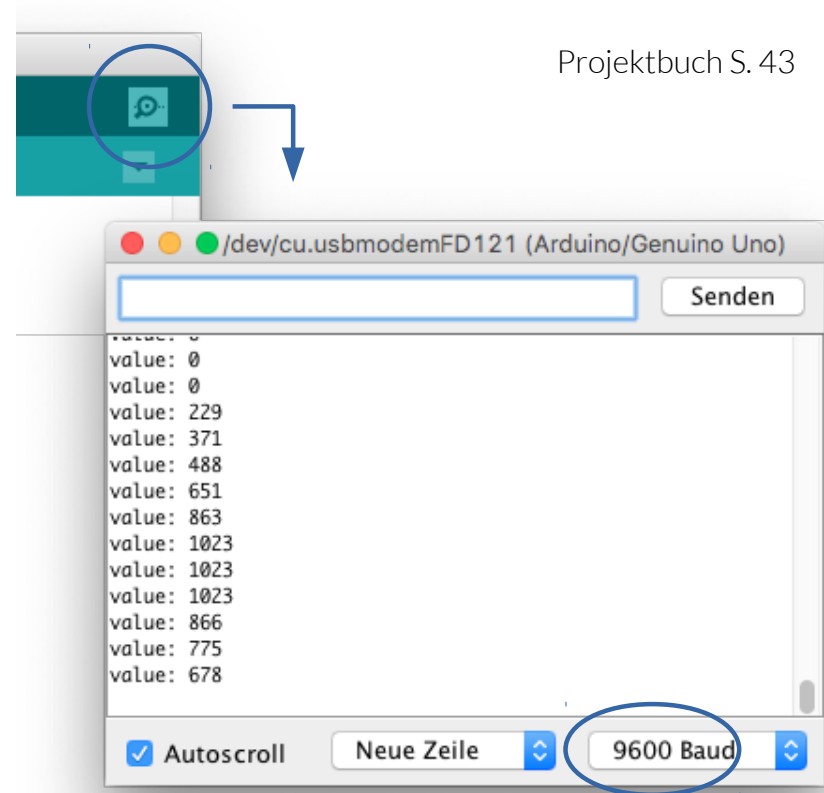
```
void setup() {  
  pinMode(A0, INPUT);  
  pinMode(4, OUTPUT);  
  pinMode(5, OUTPUT);  
  pinMode(6, OUTPUT);  
  pinMode(7, OUTPUT);  
}  
  
void loop() {  
  int value = analogRead(A0);  
  digitalWrite(4, value > 200);  
  digitalWrite(5, value > 400);  
  digitalWrite(6, value > 600);  
  digitalWrite(7, value > 800);  
}
```

value = 0 ... 1023  
0V ... 5V

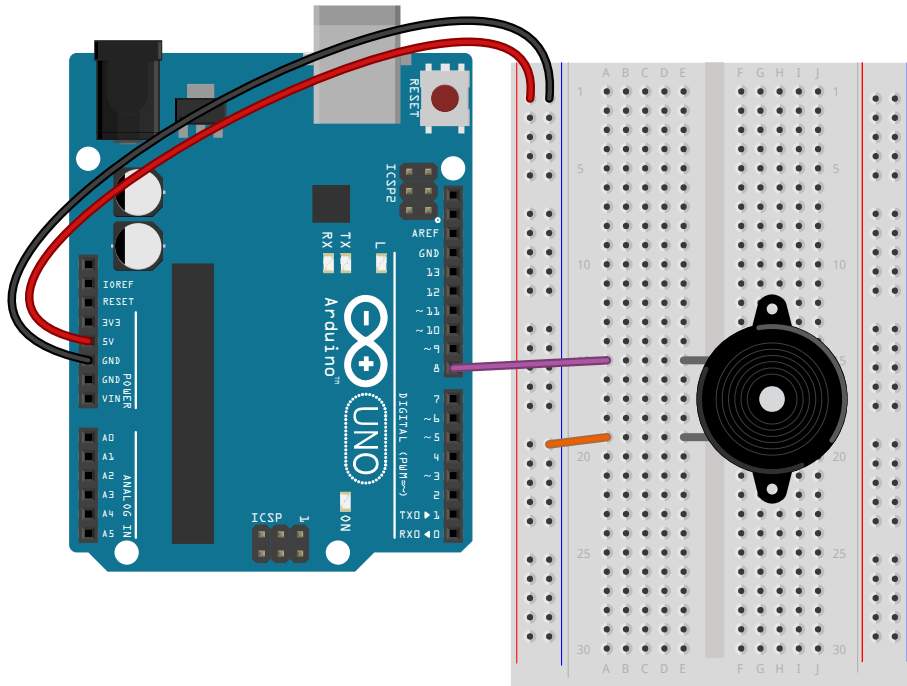
# Serieller Monitor

Projektbuch S. 43

```
void setup() {  
  pinMode(A0, INPUT);  
  pinMode(4, OUTPUT);  
  pinMode(5, OUTPUT);  
  pinMode(6, OUTPUT);  
  pinMode(7, OUTPUT);  
  Serial.begin(9600);  
}  
  
void loop() {  
  int value = analogRead(A0);  
  Serial.print("value: ");  
  Serial.println(value);  
  digitalWrite(4, value > 200);  
  digitalWrite(5, value > 400);  
  digitalWrite(6, value > 600);  
  digitalWrite(7, value > 800);  
  delay(200);  
}
```



# Töne erzeugen

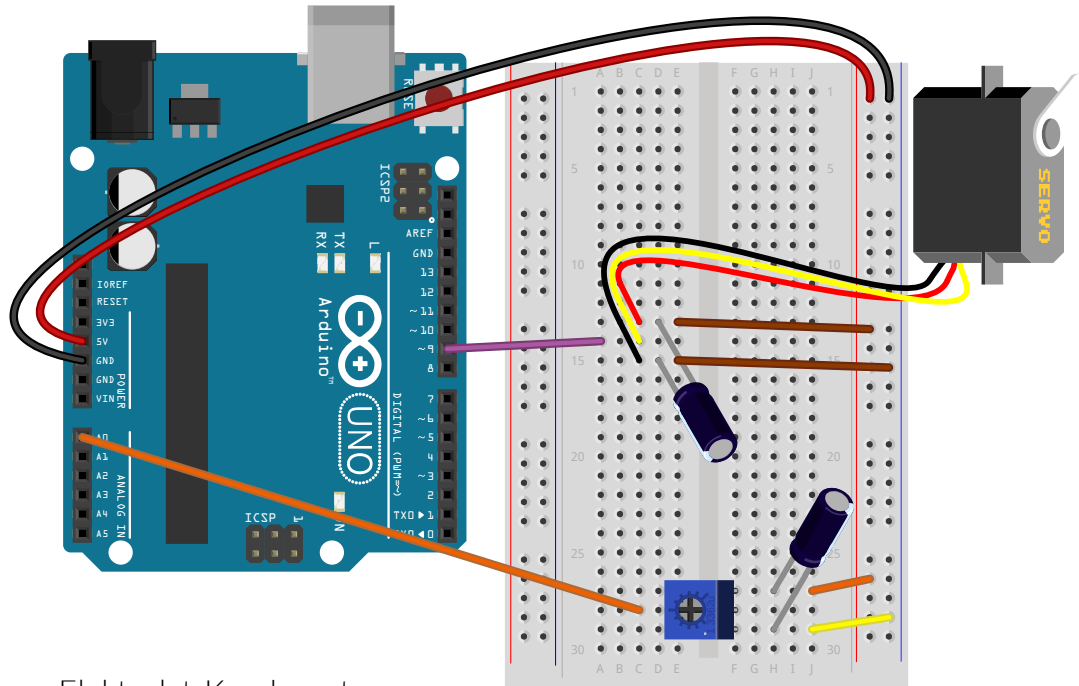


```
void setup() {  
}  
  
void loop() {  
  tone(8, 440, 200);  
  delay(200);  
  tone(8, 550, 100);  
  delay(100);  
  tone(8, 587, 100);  
  delay(100);  
  tone(8, 660, 100);  
  delay(1600);  
}
```

`tone(pin, frequency, [duration])`

<https://www.arduino.cc/en/Reference/Tone>  
Projektbuch S. 71, 79

# Servo ansteuern (Buch Projekt 05)



Programm:

Datei ▶ Beispiele

▶ 10.StarterKit\_BasicKit

▶ p05\_ServoMoodIndicator

Elektrolyt-Kondensator  
richtig herum anschliessen!  
Minus-Markierung beachten.

Projektbuch S. 64–67

# Ausblick

- Kontrollstrukturen: mit **if** und **while** geht alles, manchmal ist **for**, **switch**, **do while** bequemer
- Logische Verknüpfungen: Und, Oder, Nicht
- Datentypen: was sind die Limiten von **int**, was gibts sonst noch? Ganzzahlen, Kommazahlen, Zeichenketten
- Mathe: Modulo, Bitmanipulation, Zufallszahlen, Trigonometrie
- Arrays: mehrere Werte in einer Variable
- Funktionen schreiben und verwenden
- Libraries verwenden

*Im Übrigen: die Sprache ist (fast) C++, es gilt jede C- oder C++-Referenz*

- Mehr Sensoren: Licht, Temperatur (analog/digital), Feuchte, Distanz, Beschleunigung, Magnetfeld
- Mehr Aktoren: Servo, DC-Motor, Schrittmotor, LCD-Anzeige, LED-Strips, Relais
- Mit anderer Hardware sprechen: Real-Time Clock, RFID, SD-Karte, GPS, Internet

Quellennachweis:

Arduino-Uno-Foto von [www.arduino.cc](http://www.arduino.cc), CC BY-SA 3.0

Felssturz-Illustration aus dem Arduino-Projektbuch, © Arduino Srl., CC BY-NC-SA 3.0

Schaltungsskizzen erstellt mit Fritzing (<http://fritzing.org>), CC BY-SA 3.0

© 2017–2018 Christian Walther <[cwalther@gmx.ch](mailto:cwalther@gmx.ch)>, FabLab Winti ([www.fablabwinti.ch](http://www.fablabwinti.ch)), CC BY-SA 4.0

