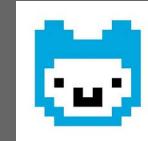


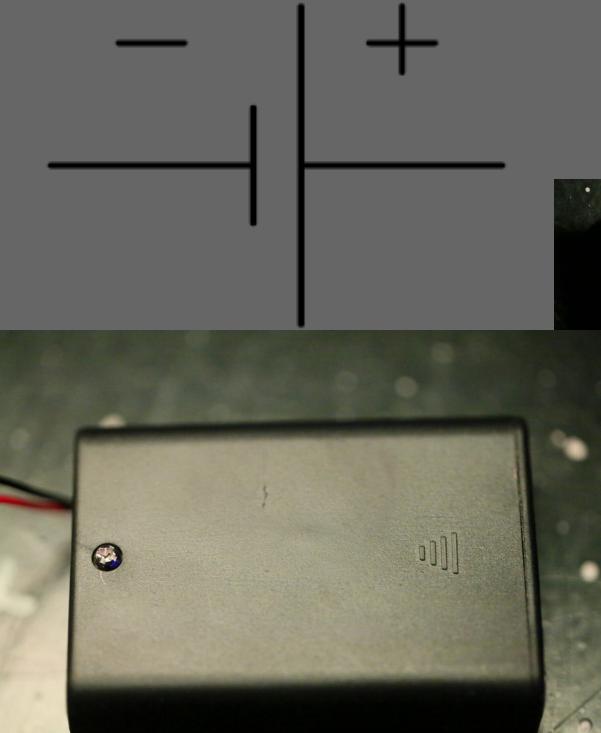
Foliensatz

- bengoshi.crew.c-base.org/CMS



CODERDOJO.NEO

Batterie einsetzen

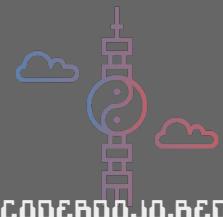
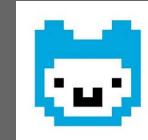
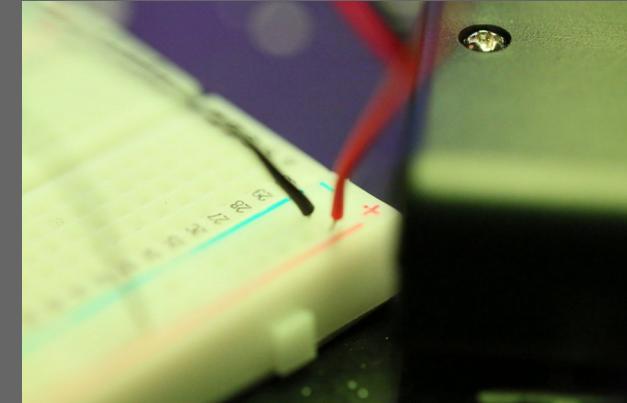
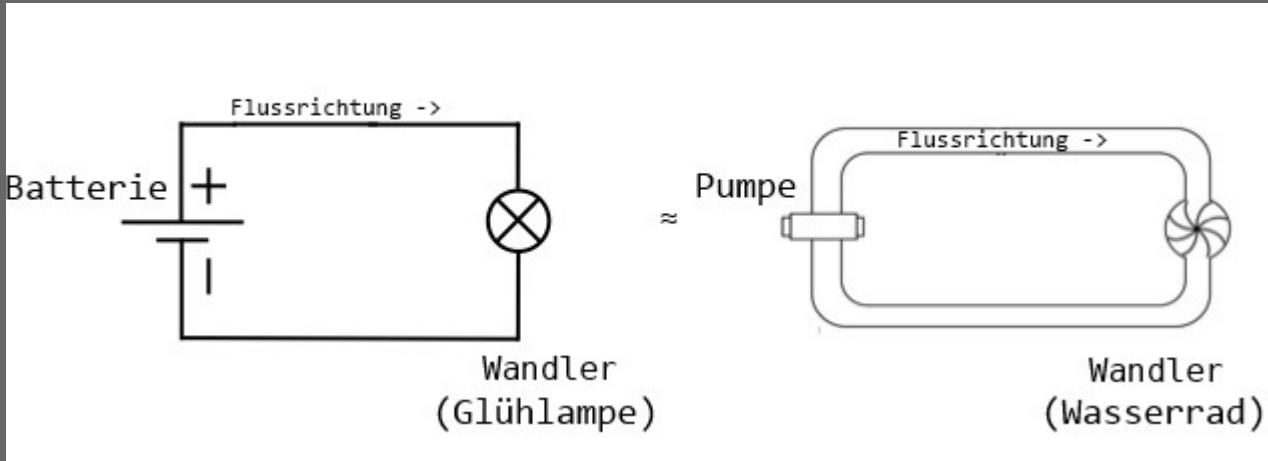


- 1) Schraube auf
- 2) aufschieben
- 3) Batterien gepolt einsetzen
- 4) schließen
- 5) Schraube rein



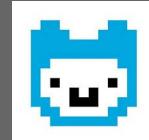
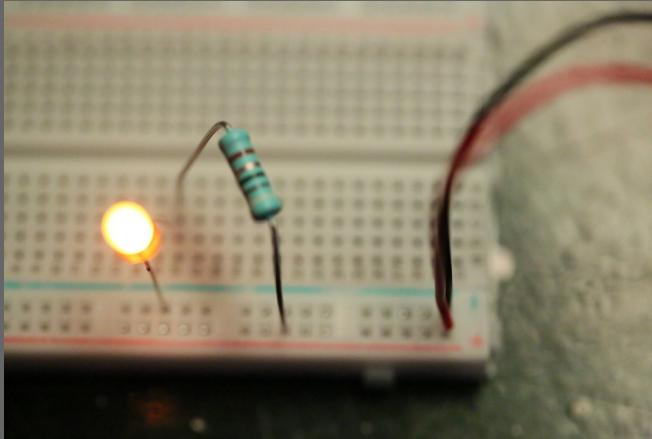
Schaltung aufbauen

- Stromkreis aufbauen



Schaltung aufbauen

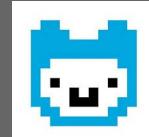
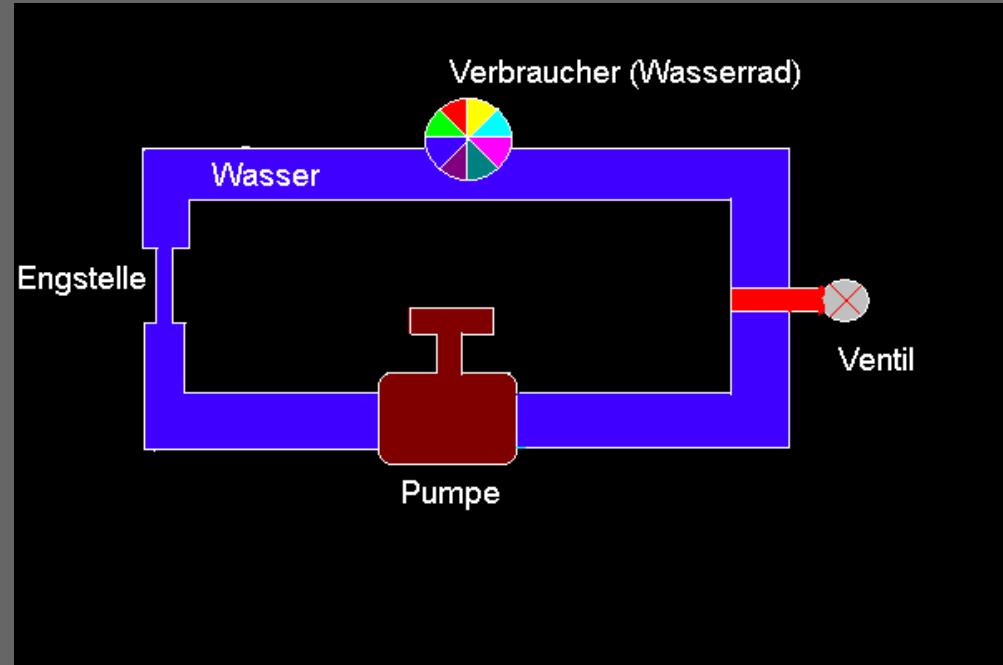
- Widerstand 220 Ω



CODERODOJO.EDU

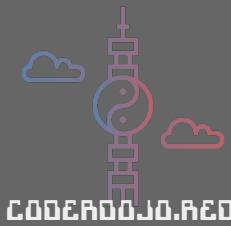
Schaltung aufbauen

- Widerstand
 - reduziert die Strom / Spannung



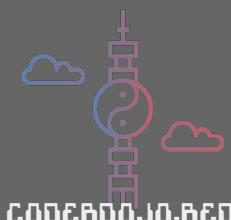
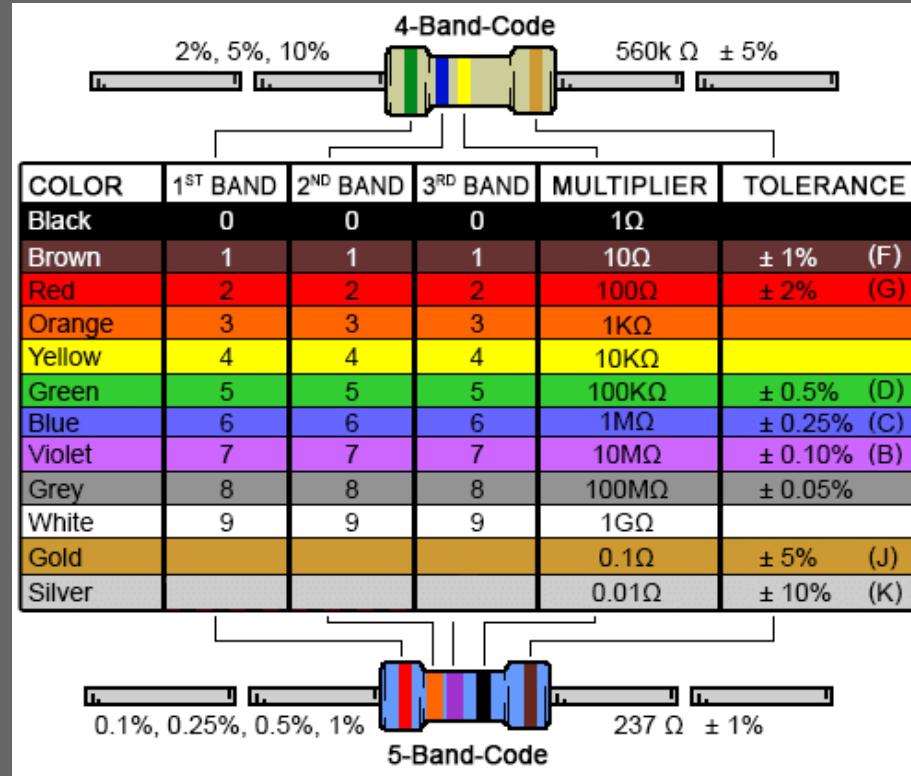
Schaltung aufbauen

- Widerstand
 - reduziert die Strom / Spannung



Schaltung aufbauen

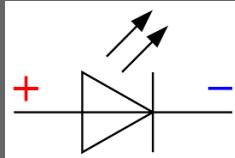
- Widerstand



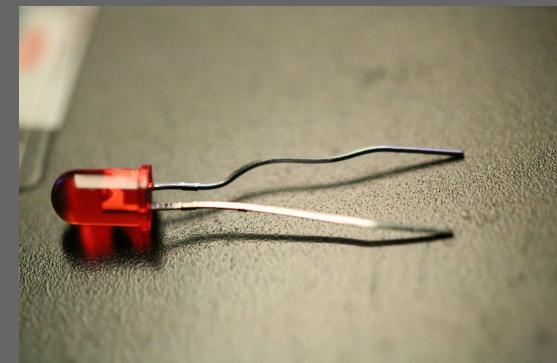
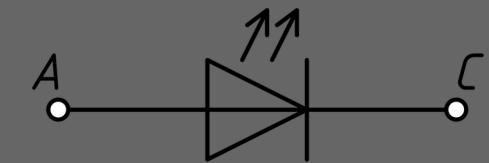
CODERDOJO.RED

Schaltung aufbauen

- LED



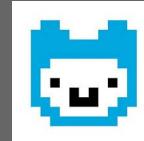
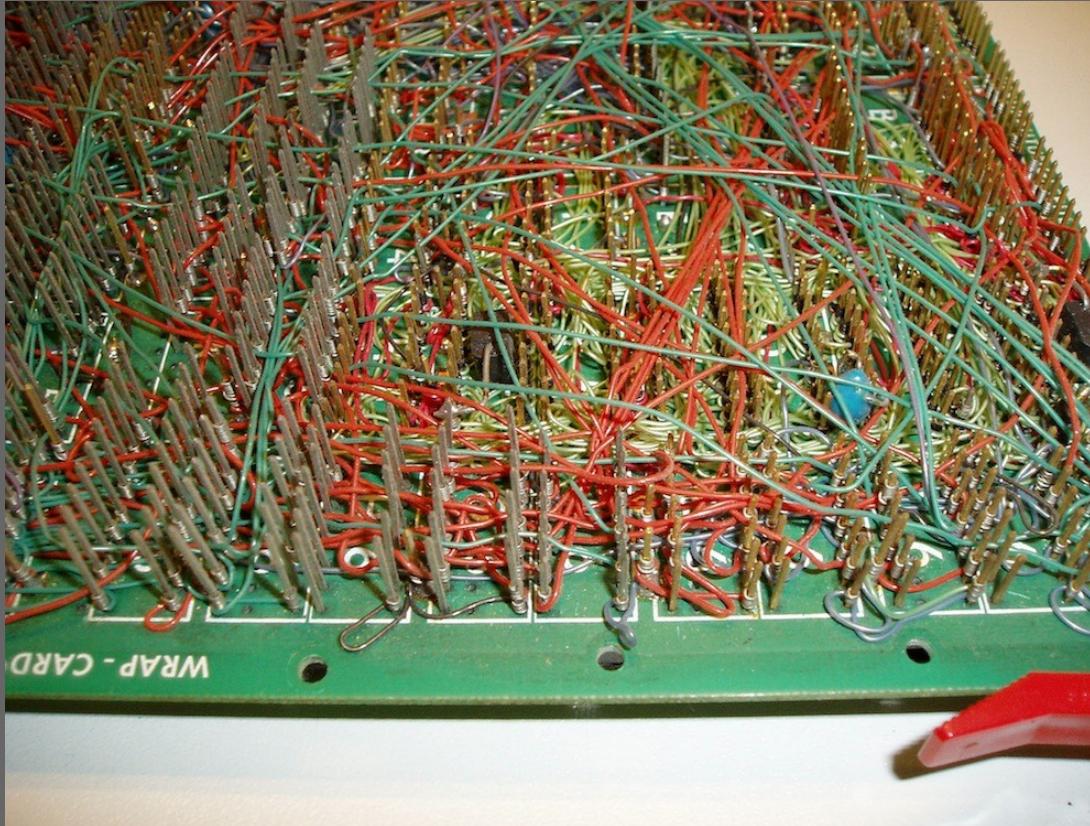
- kurzes Bein → Kathode → - - Pol
- langes Bein → Anode → + - Pol
- LEDs mögen keine hohe Spannung,
deshalb brauchen sie in aller Regel
einen Widerstand!



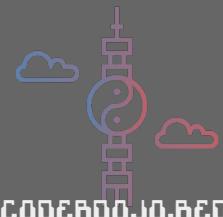
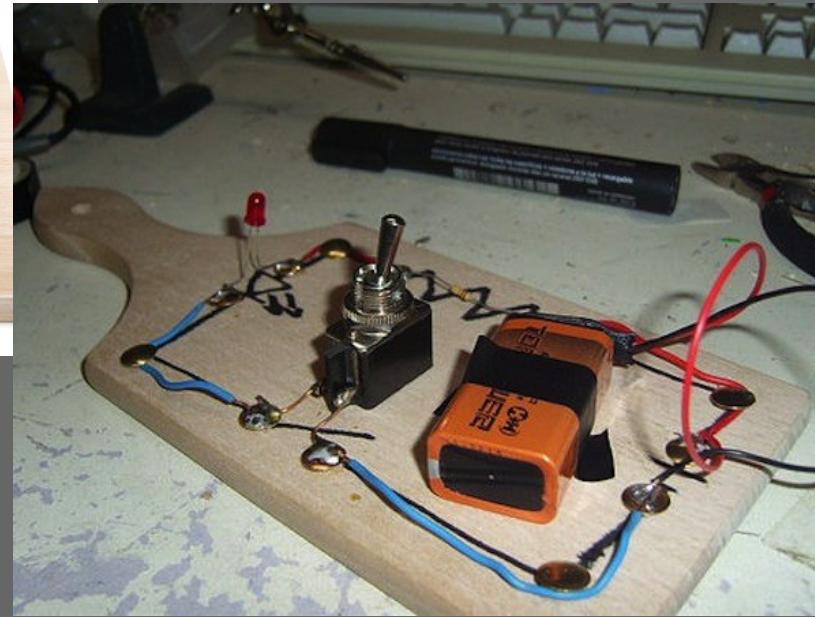
Warum Breadboard?



Darum Breadboard?

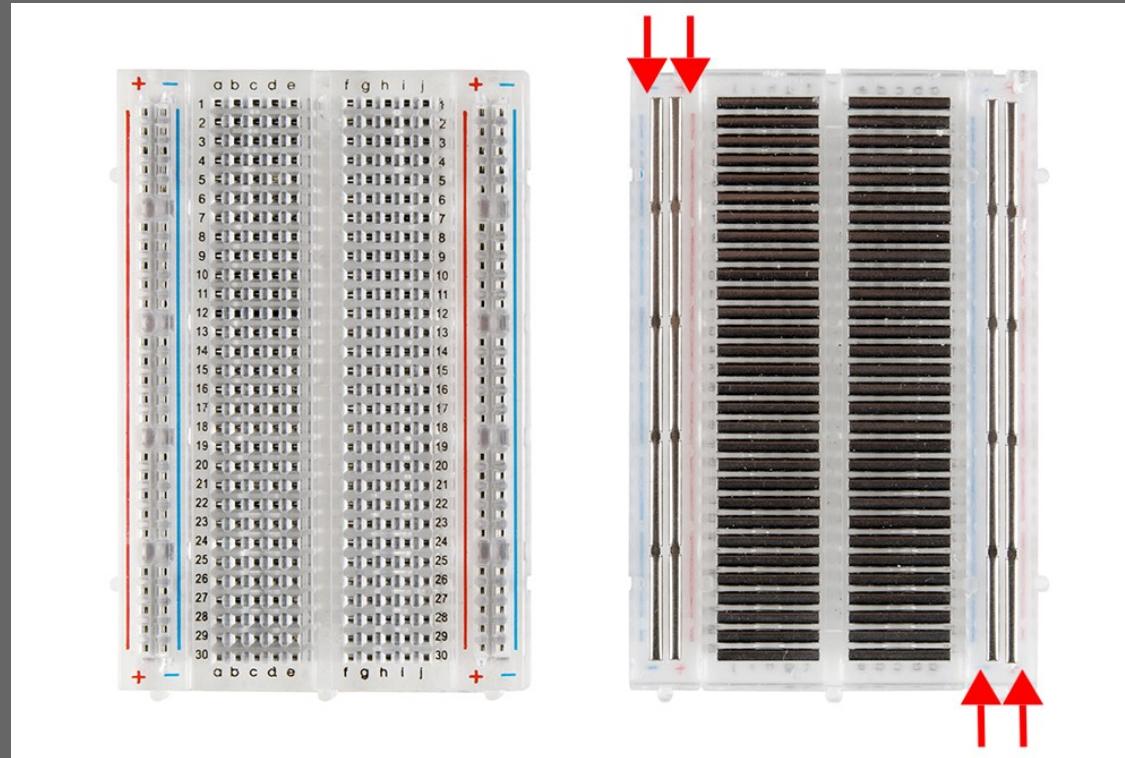


Erste Versuche der Besserung...

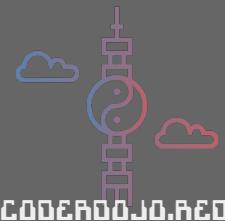
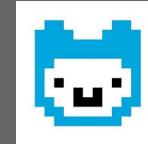
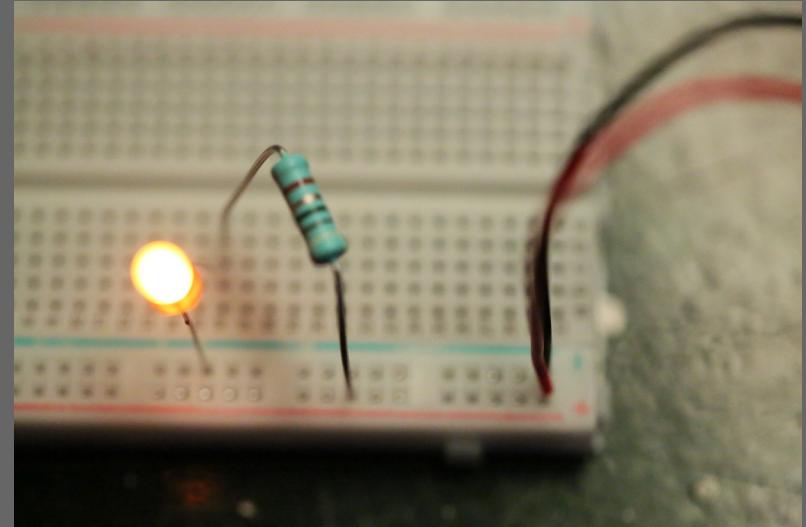
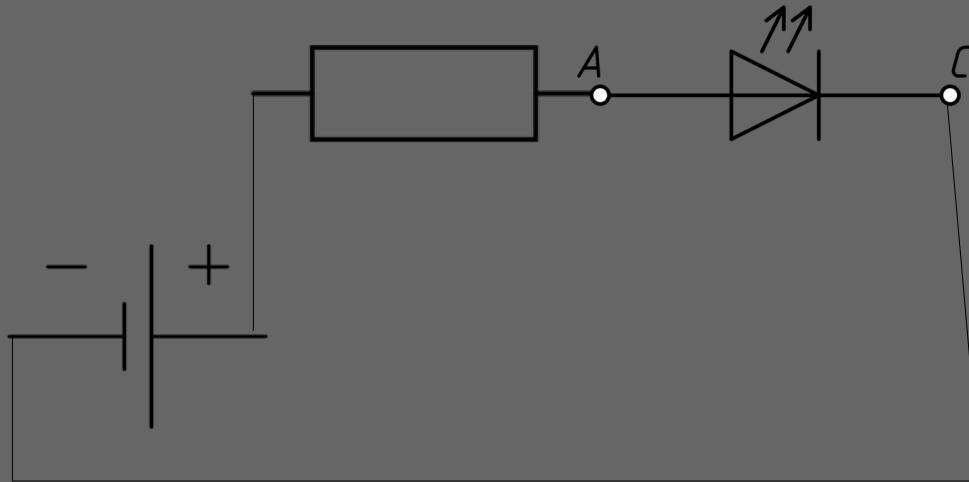


CODERDOJO.NEO

Wie funktioniert ein Breadboard?



Schaltung aufbauen



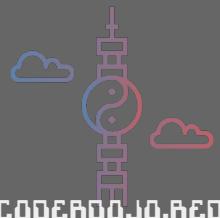
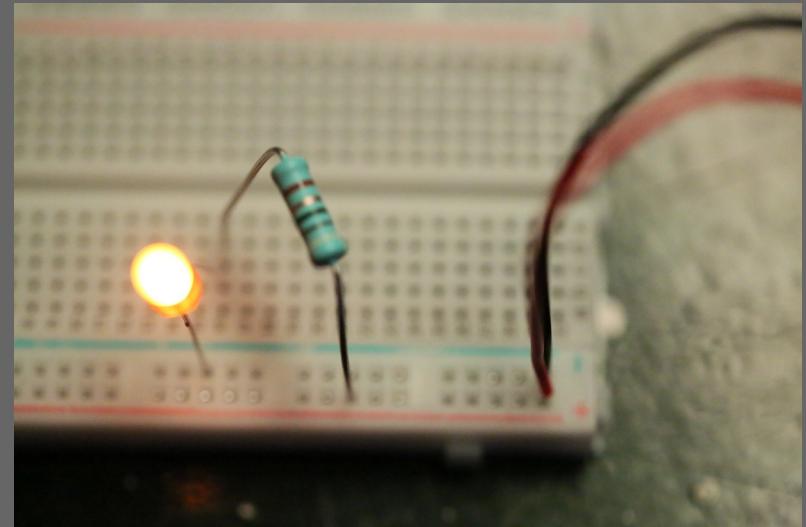
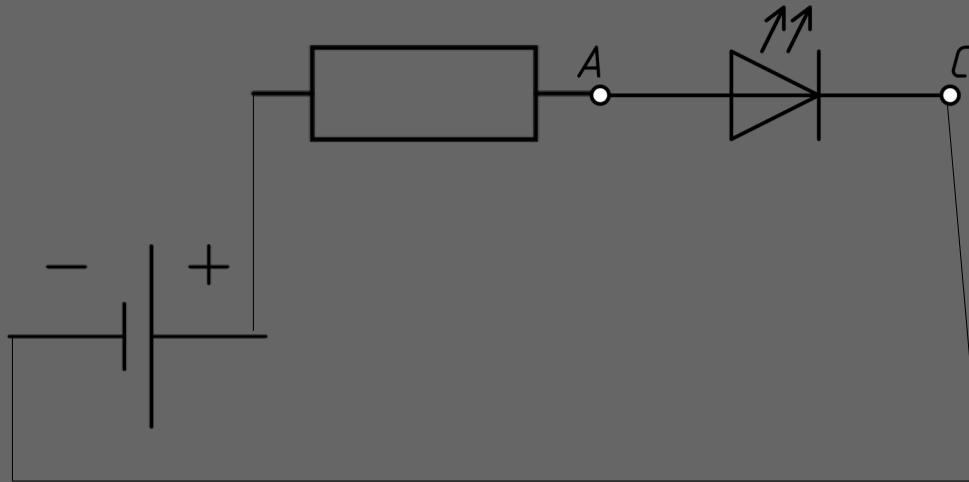
CODERDOJO.NE

Schaltung aufbauen

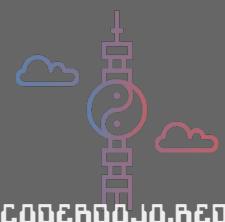


CODERDOJO.RO

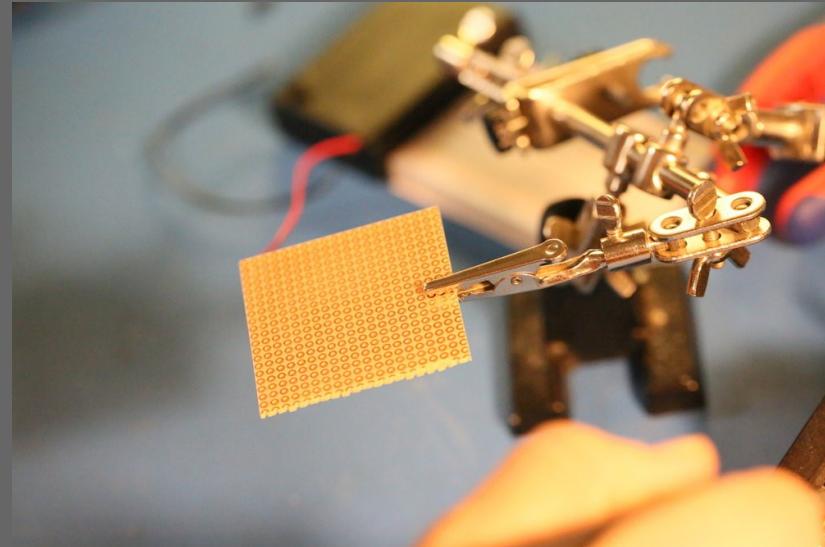
Schaltung aufbauen



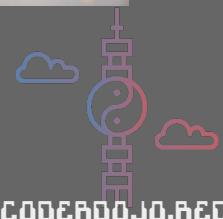
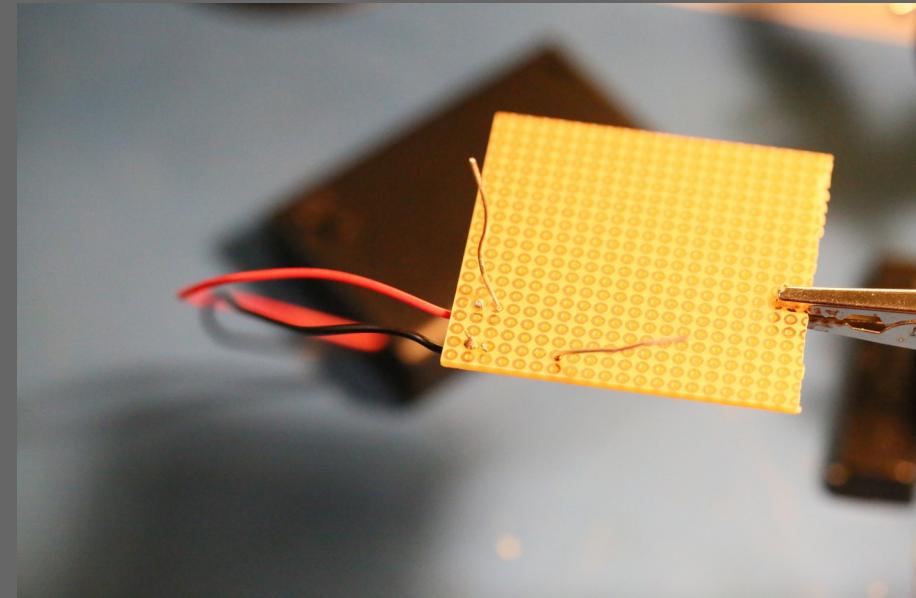
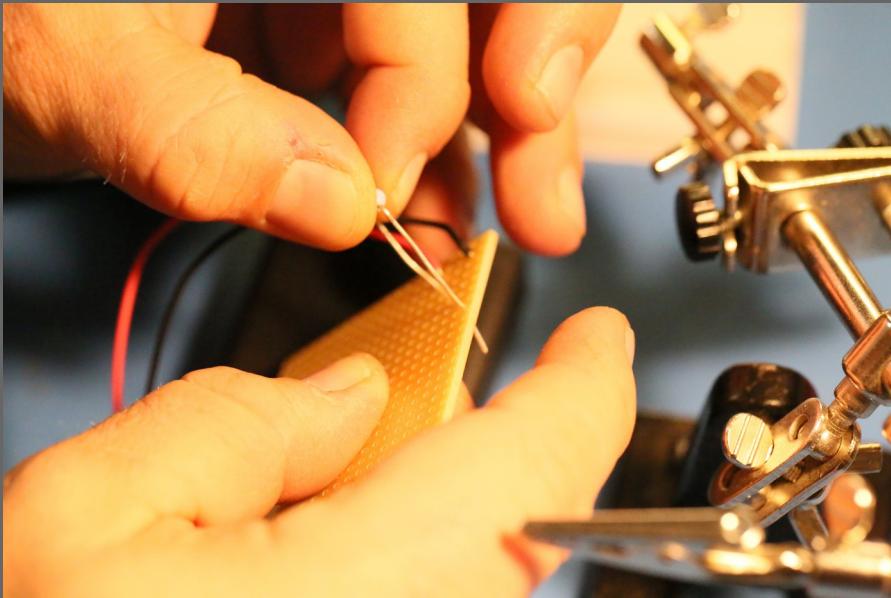
Löten



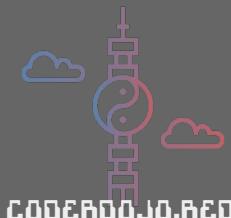
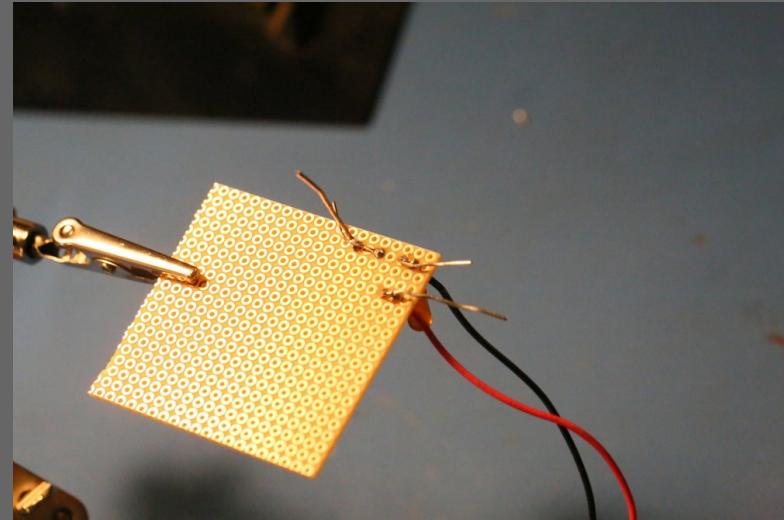
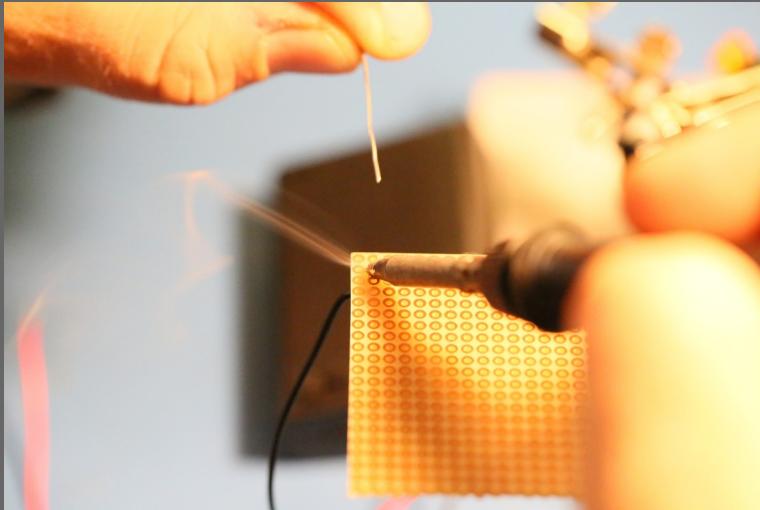
Löten



Löten

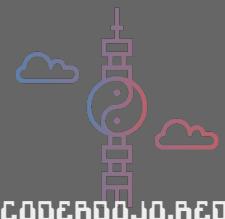
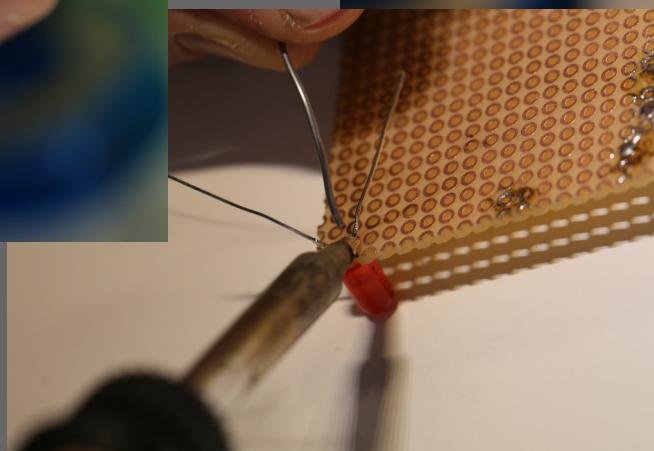
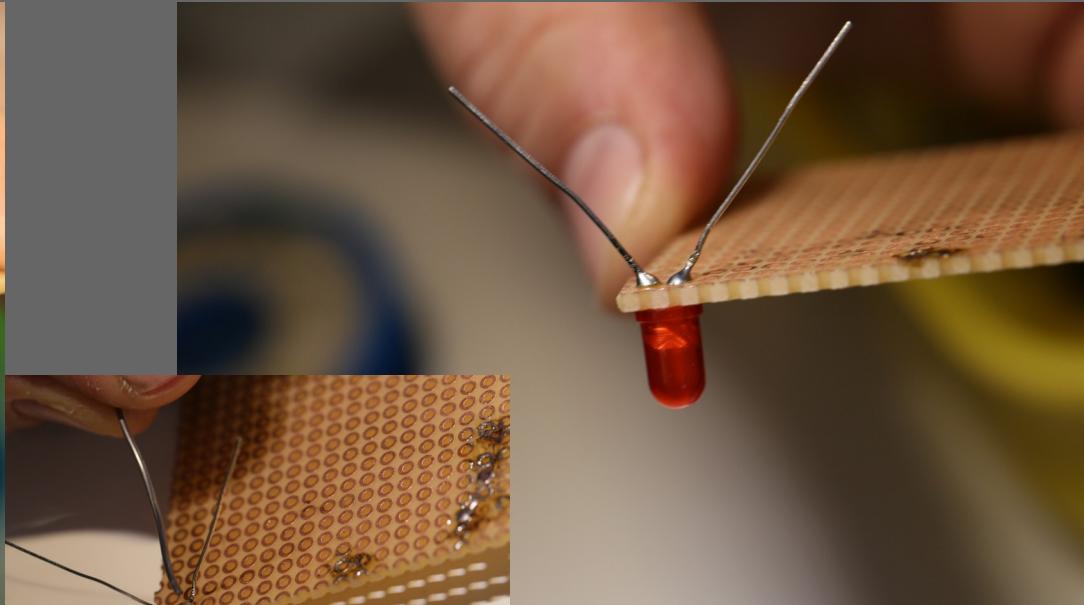
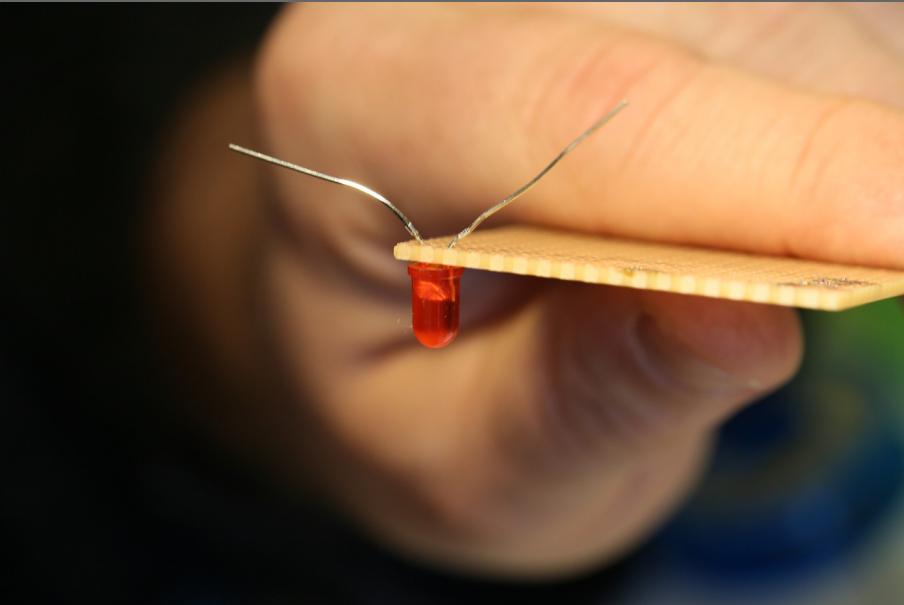


Löten



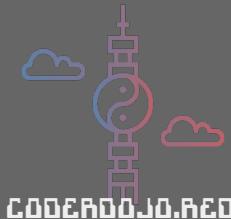
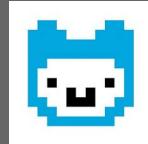
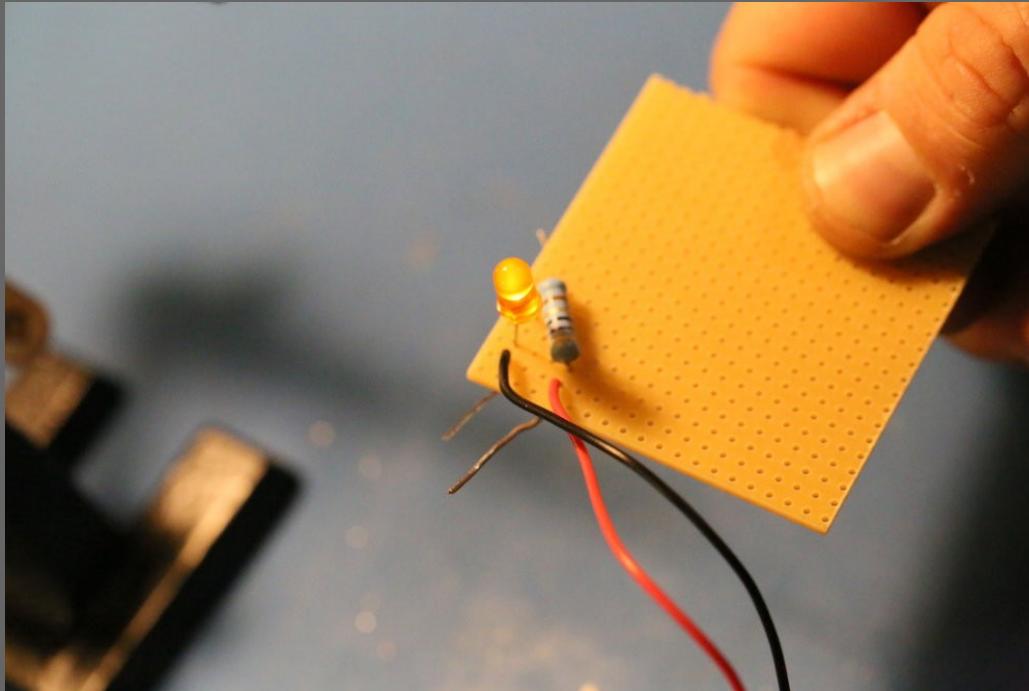
CODEcademy

Löten

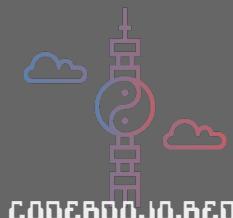


CODEROOJO.RED

Löten



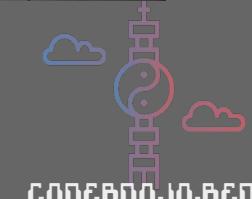
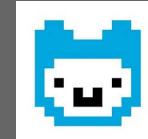
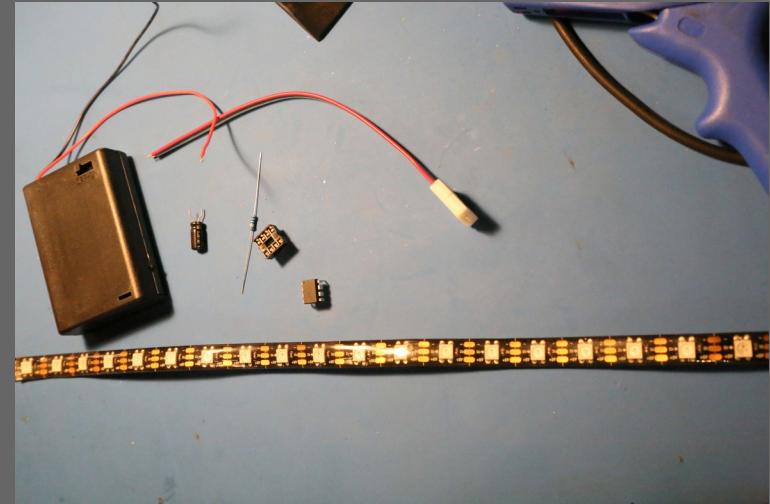
Strom an – leuchtet die LED?



CODERDOJO.RED

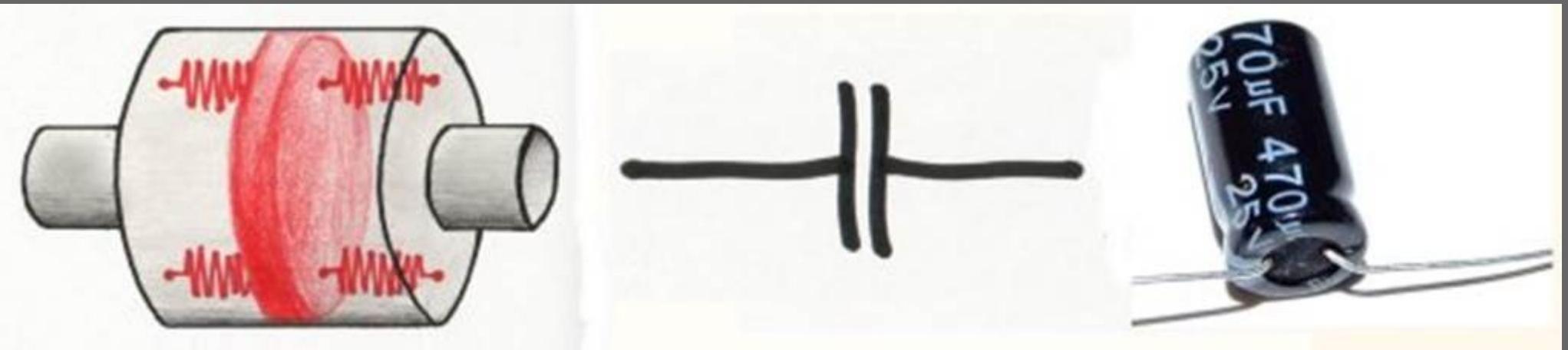
benötigte Bauteile – stückchenweise

- 220 Ω Vorschalt-Widerstand
- 10 μF Kondensator
- IC-Sockelhalter mit IC
- LED-Stripe
- Stripe-Verbinder
- plus Teile von eben



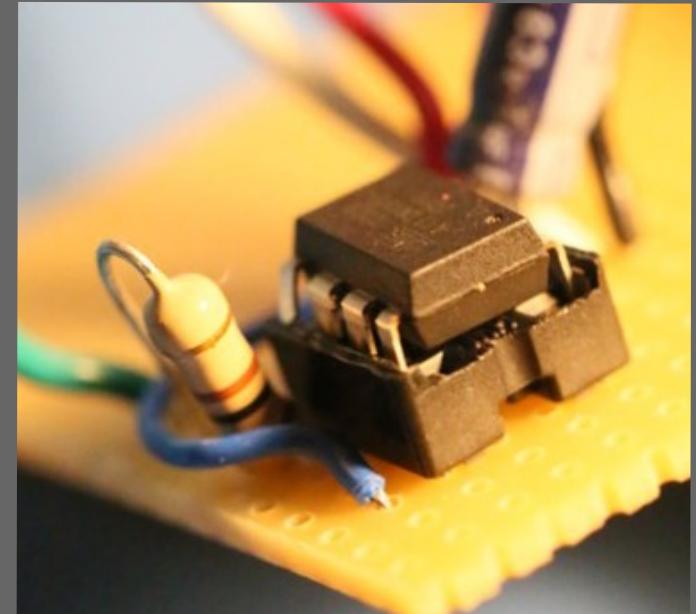
Neue Bauteile – kurz erklärt

- Kondensator → F – Farad (Faraday → Käfig)
- fängt Spannungsspitzen ab (⚡ Wasserkreislauf)



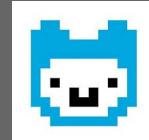
Neue Bauteile – kurz erklärt

- IC → integrated circuit
(integrierter Schaltkreis)
- enthalten vor allem Transistoren,
aber auch Widerstände und
Kondensatoren (und teilweise mehr)
- Transistoren sind vereinfacht Schalter
- ATTiny85 → Mikrocontroller, enthält
noch Prozessor, also Recheneinheit und Speicher



Blinklicht mit Microcontroller

- vorher kurze Pause!



CODEROOJO.RED

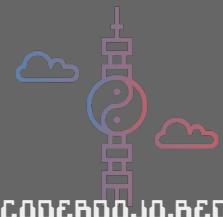
1. Schritt

- Stromzufuhr, LED und Widerstand auslöten

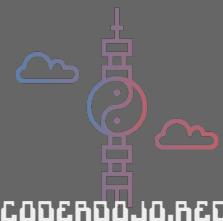
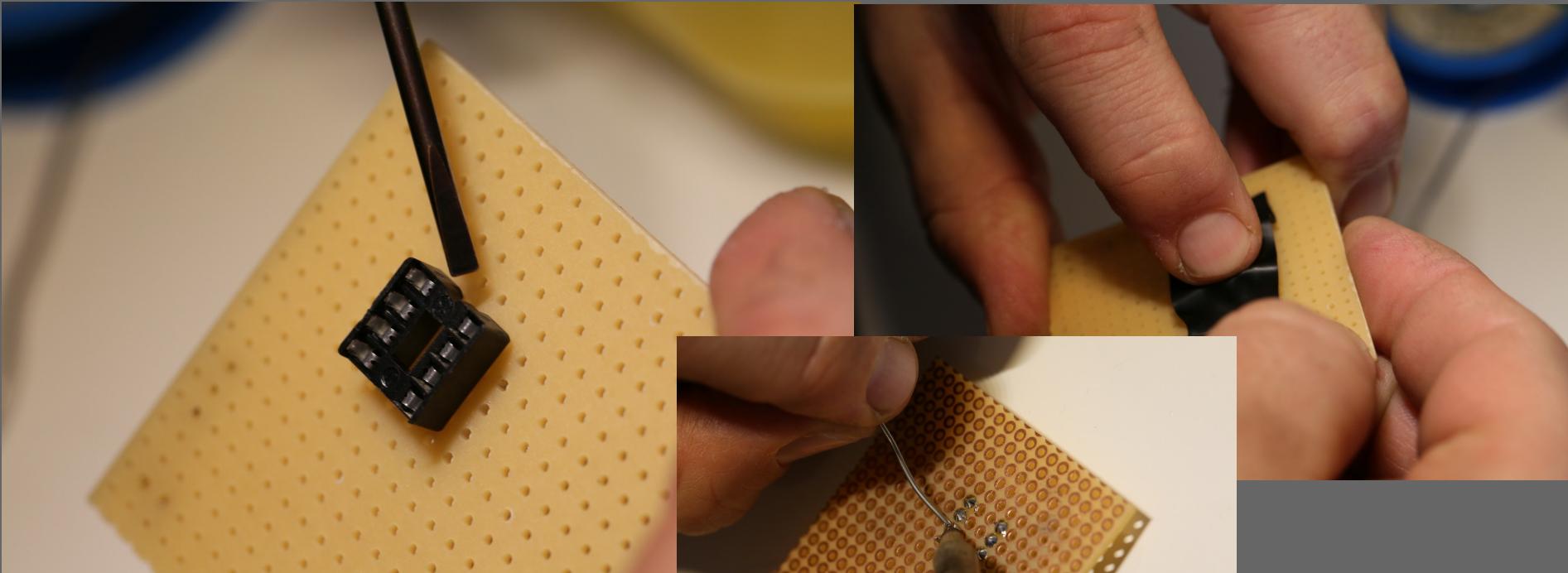


1. Schritt

- IC-Sockelhalter anlöten
 - daran denken: Lötspitze an den Pin, dann Lötzinn drauf; nicht das Lötzinn an die Spitze halten und es “klecksen” lassen

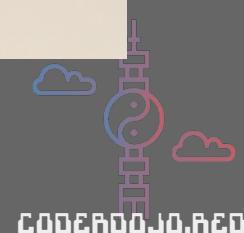
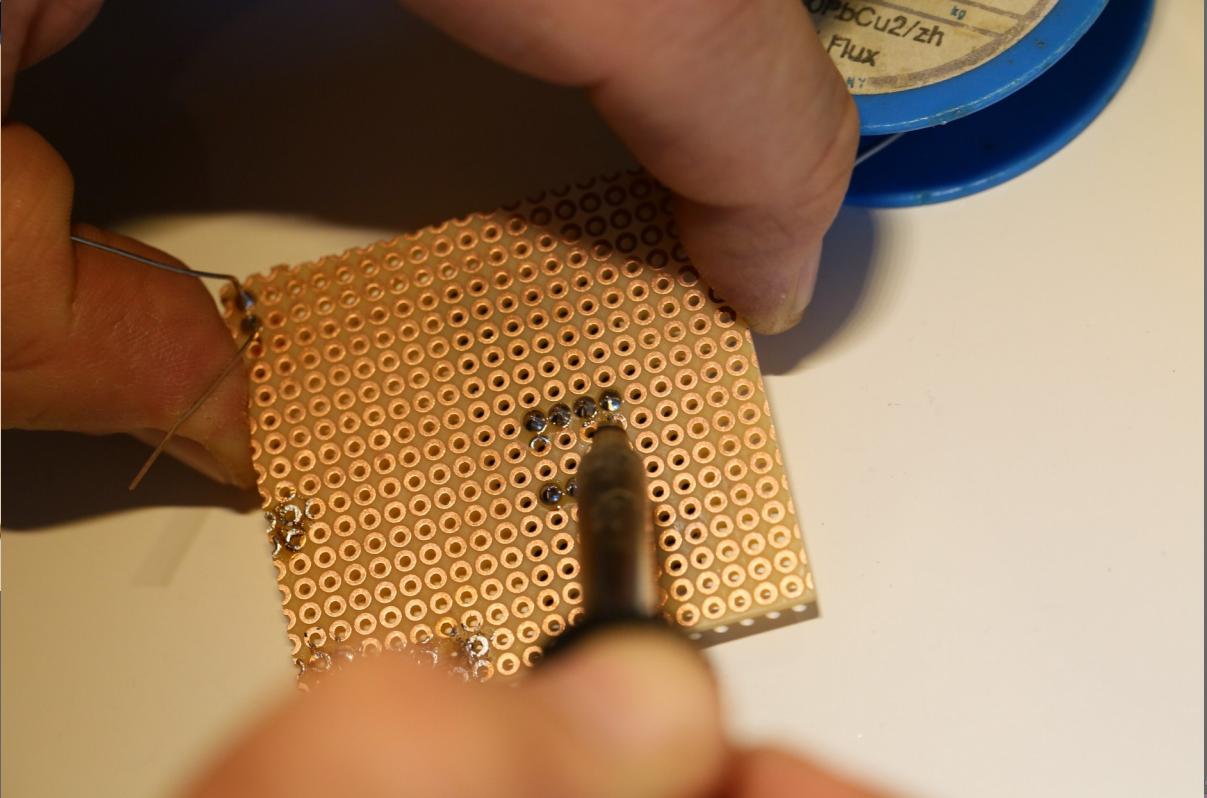
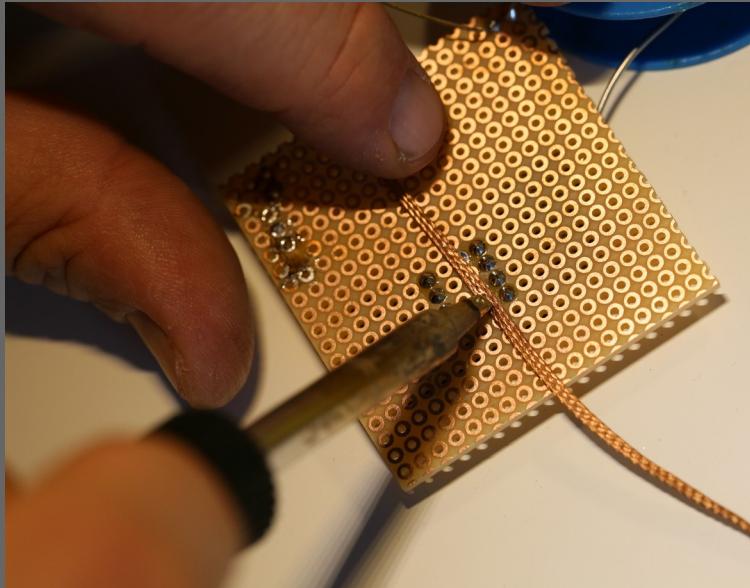


1. Schritt



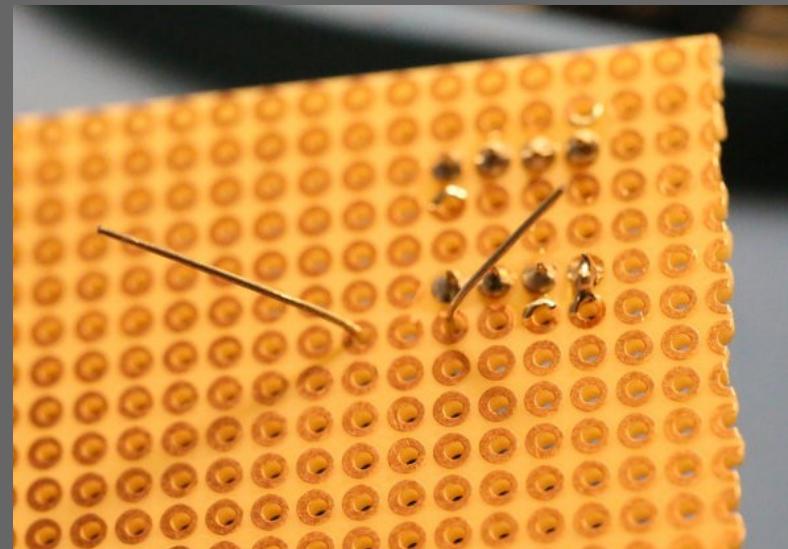
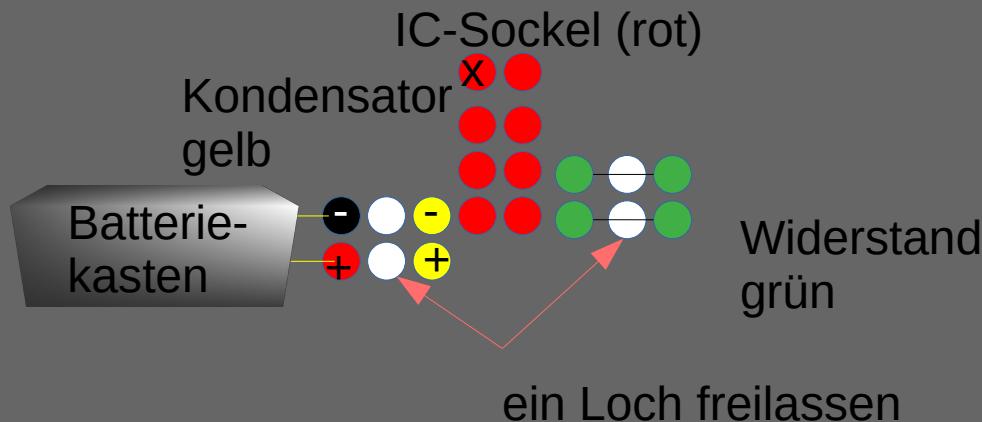
CODERDOJO.NEO

1. Schritt

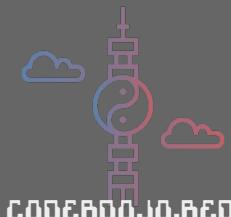


1. Schritt

- Sockel, Kondensator und Widerstände einlöten

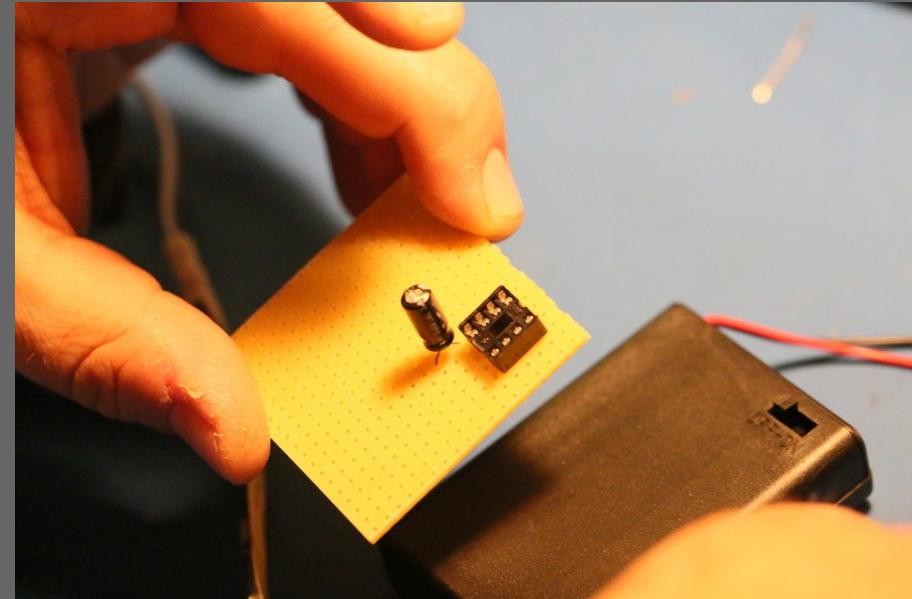
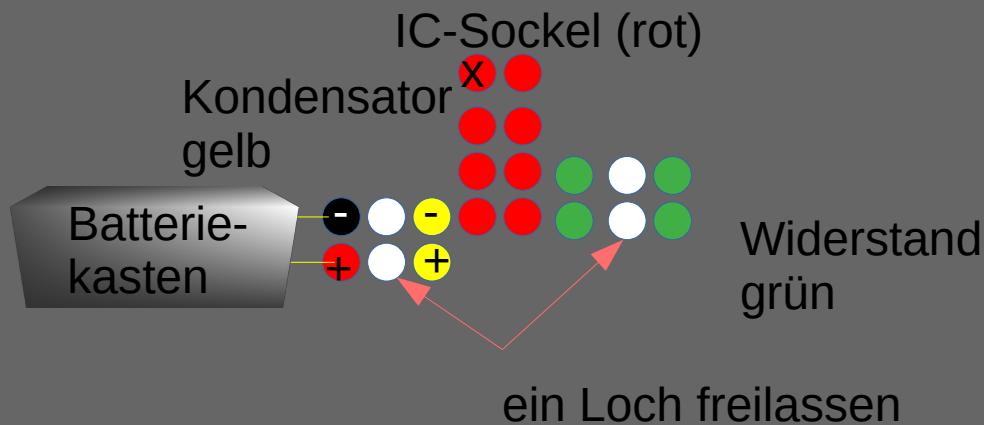


Aufsicht

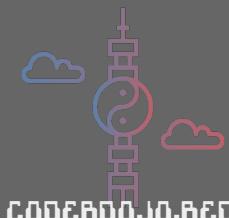


2. Schritt

- Batteriekontakte einlöten

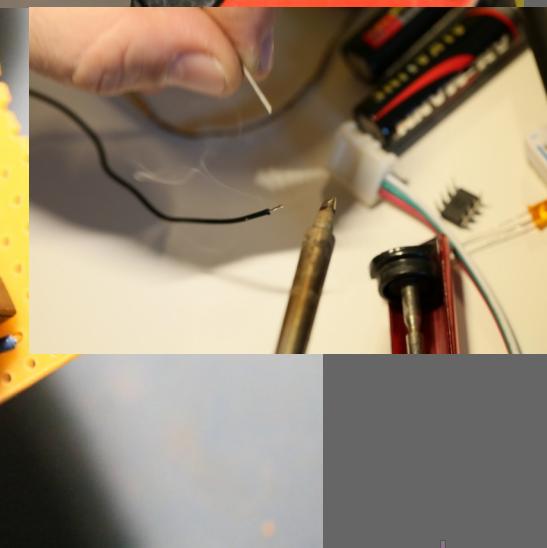
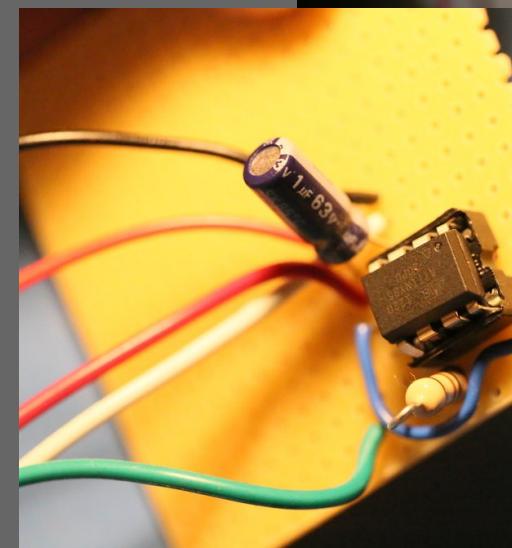
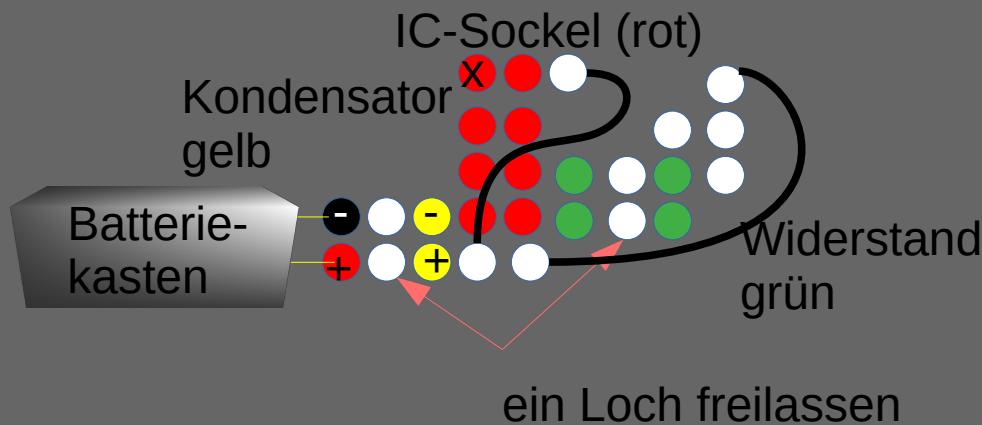


Aufsicht

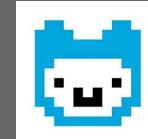


3. Schritt

- Drahtbrücke löten



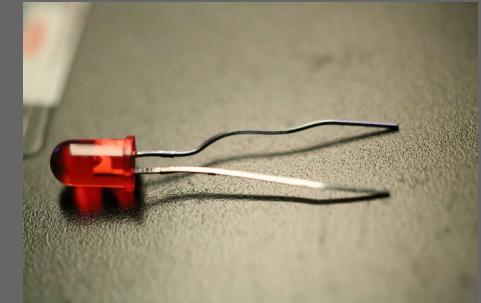
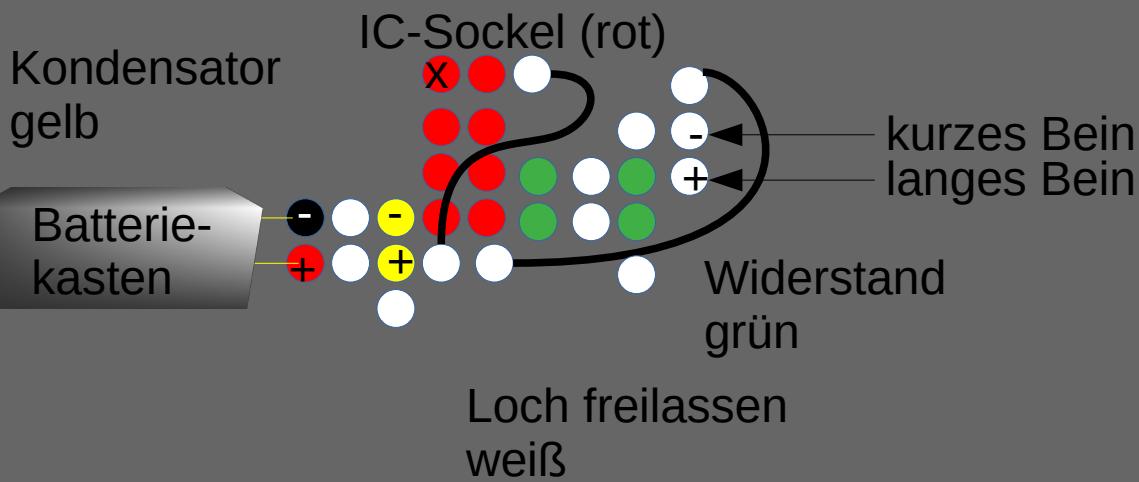
Aufsicht



CODERDOJO.RO

4. Schritt

- LED einlöten – volle Konzentration bei der Polung!



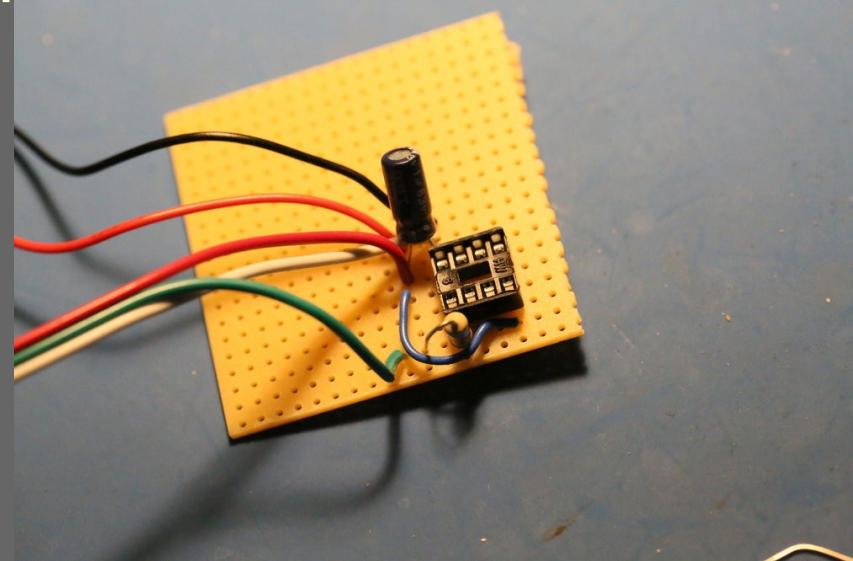
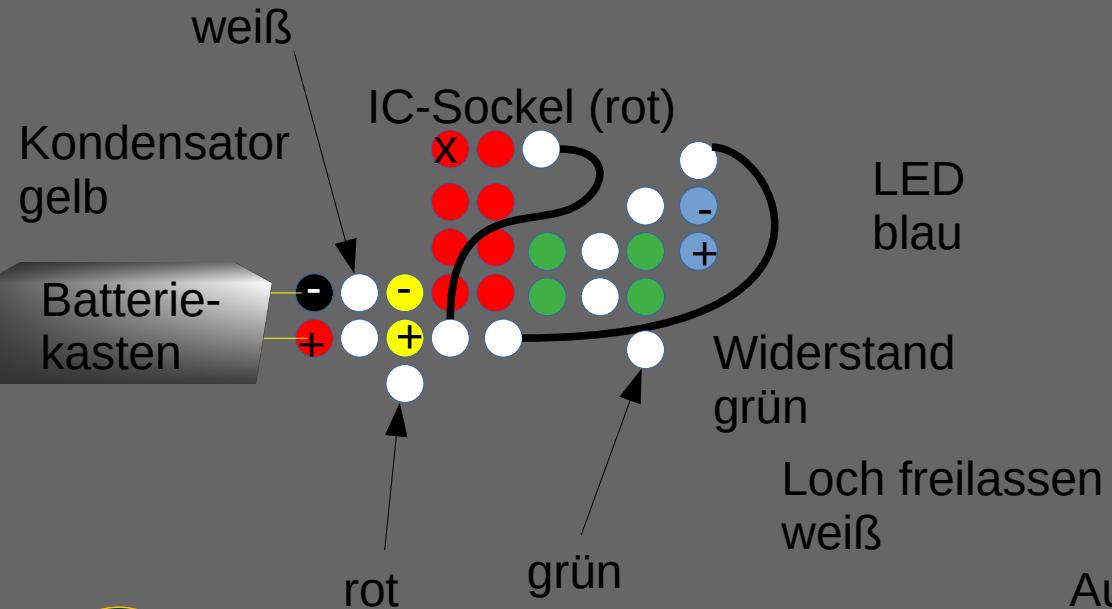
Aufsicht



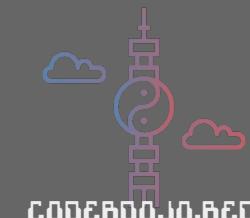
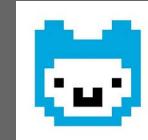
CODERDOJO.NEO

5. Schritt

- LED-Stripe-Connector einlöten – Löten nach Farben!

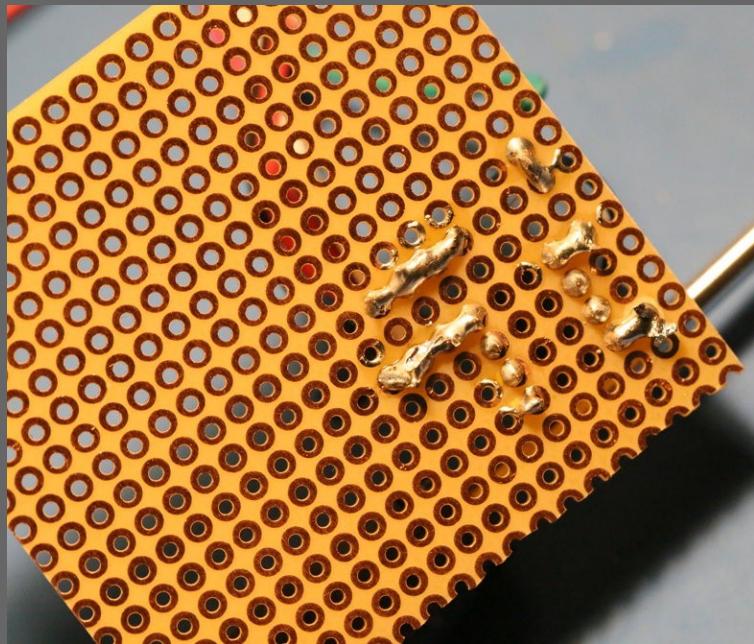
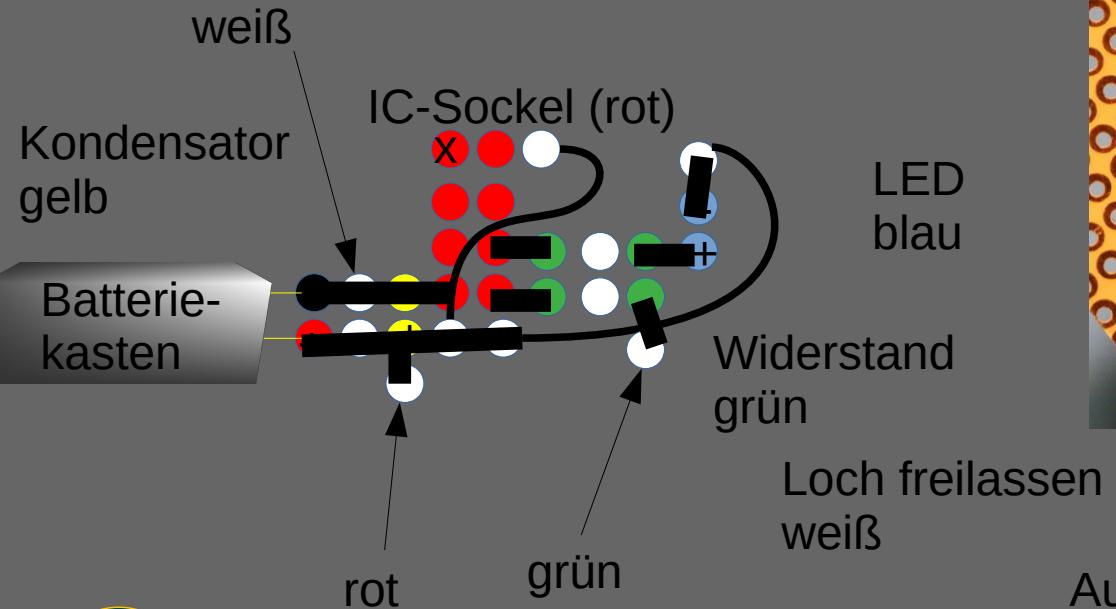


Aufsicht

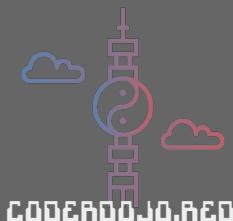
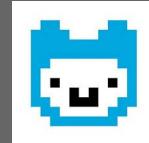


6. Schritt

- Brücken löten

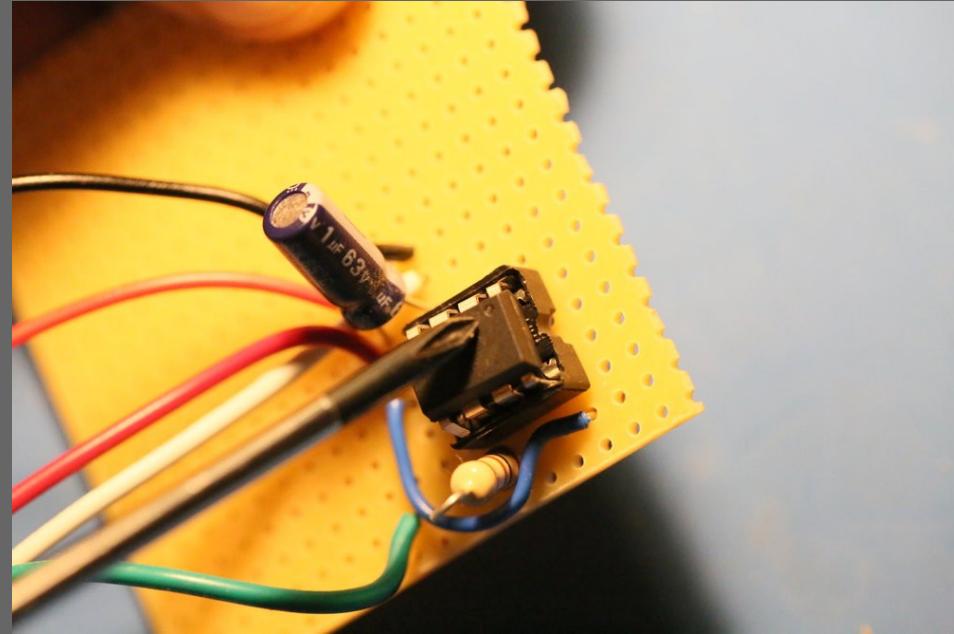
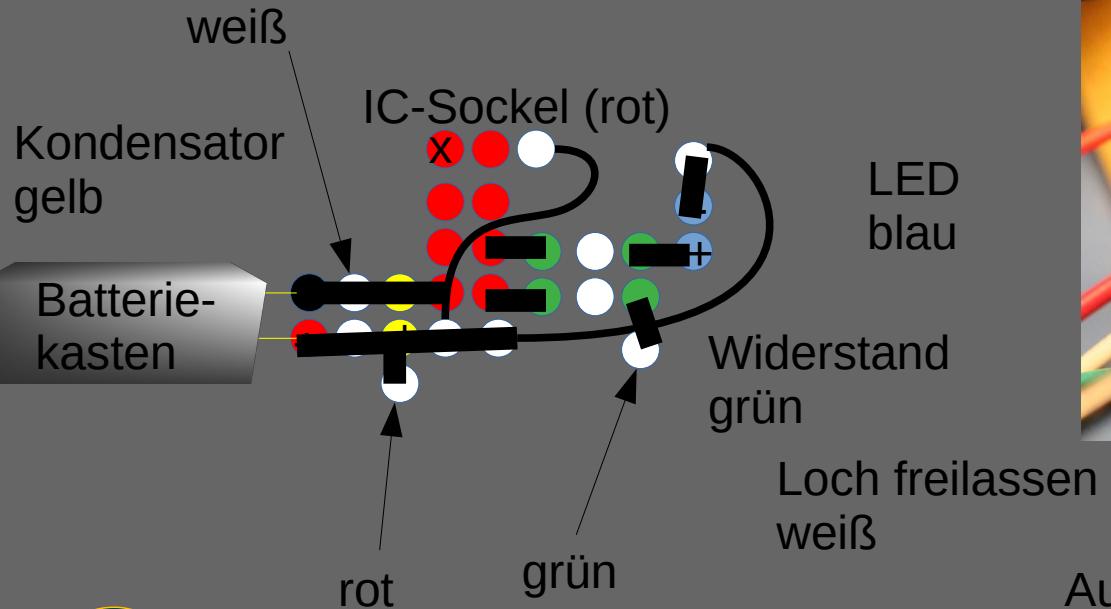


Aufsicht

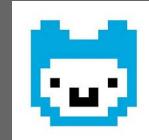


7. Schritt

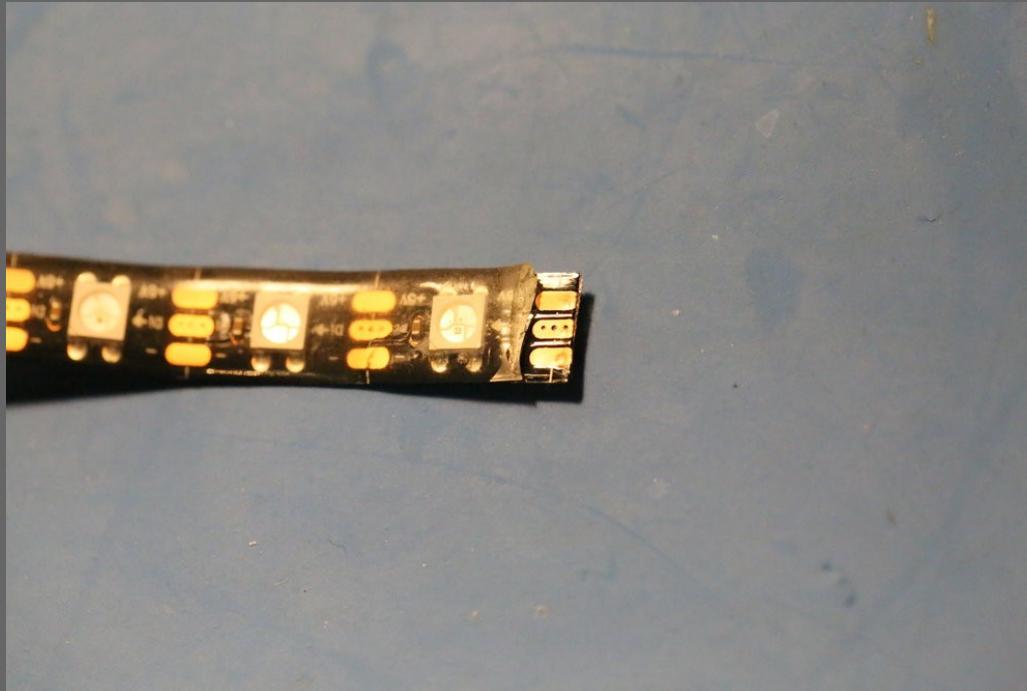
- ATTiny85 einsetzen



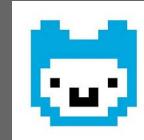
Aufsicht



LED-Stripe verbinden

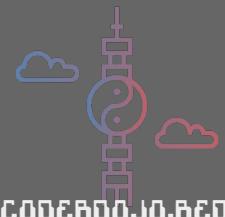


- Stripe in Klemme legen
- Klemme schließen

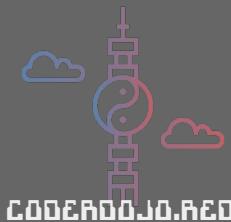


Fruststop

- Solange Du keinen programmierten IC hast, kann es nicht laufen!
- Zum Testen nimmst Du Dir am besten einen vorbereiteten IC.
Danach beginnst Du mit der Programmierung.

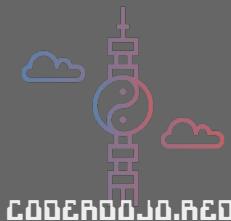
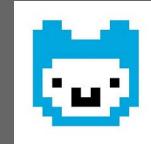


Pause



CODEEROJO.RO

auf zum Code



CODEROOJO.RED

Tiny-Referenz aufrufen

https://cdn.sparkfun.com/assets/0/4/1/4/a/Tiny_QuickRef_v2_2_1.png

over as long as the board is red.

is a single line comment
is
line
t */

pinNum, INPUT/OUTPUT/INPUT_PULLUP);
the mode of the digital I/O pin.
1.700V to 3.600V

```
analogWrite(pin, val);
/* Writes an analog voltage (using PWM) to a pin. val = integer value from 0 to 255 */
```

```
sensorVal = analogRead(pin);
/* Reads the voltage from the specified analog pin. 0V returns 0; Vcc returns 1023*/
```

TIME

```
delay(time_ms);
/* Pauses the program for the amount of time (in milliseconds). */
millis();
```

```
using analogWrite();
pins A1, A2, A3 setup for sensor input with analogRead
```

DATA TYPES

```
void // nothing is returned
boolean // 0, 1, false, true
char // 8 bits: -128 to 127
byte // 8 bits: 0 to 255
int // 16 bits: -32,768 to 32,767
unsigned int // 16 bits (0 to 65,535)
long /* 32 bits: -2,147,483,648 to 2,147,483,647 */
unsigned long // 32 bits (0 to 4,294,967,295)
```



Arduino-IDE aufrufen

Activities Processing IDE

Mon 23:47

sketch_nov04a | Arduino 1.8.10

File Edit Sketch Tools Help

sketch_nov04a

```
void setup() {
  // put your setup code here, to run once:
}

void loop() {
  // put your main code here, to run repeatedly:
}
```

1

ATtiny25/45/85, ATtiny85, Internal 8 MHz on /dev/ttyUSB0

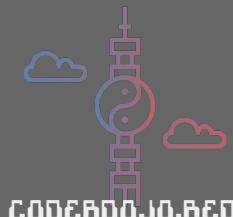


CODERDOJO.NEO

Code

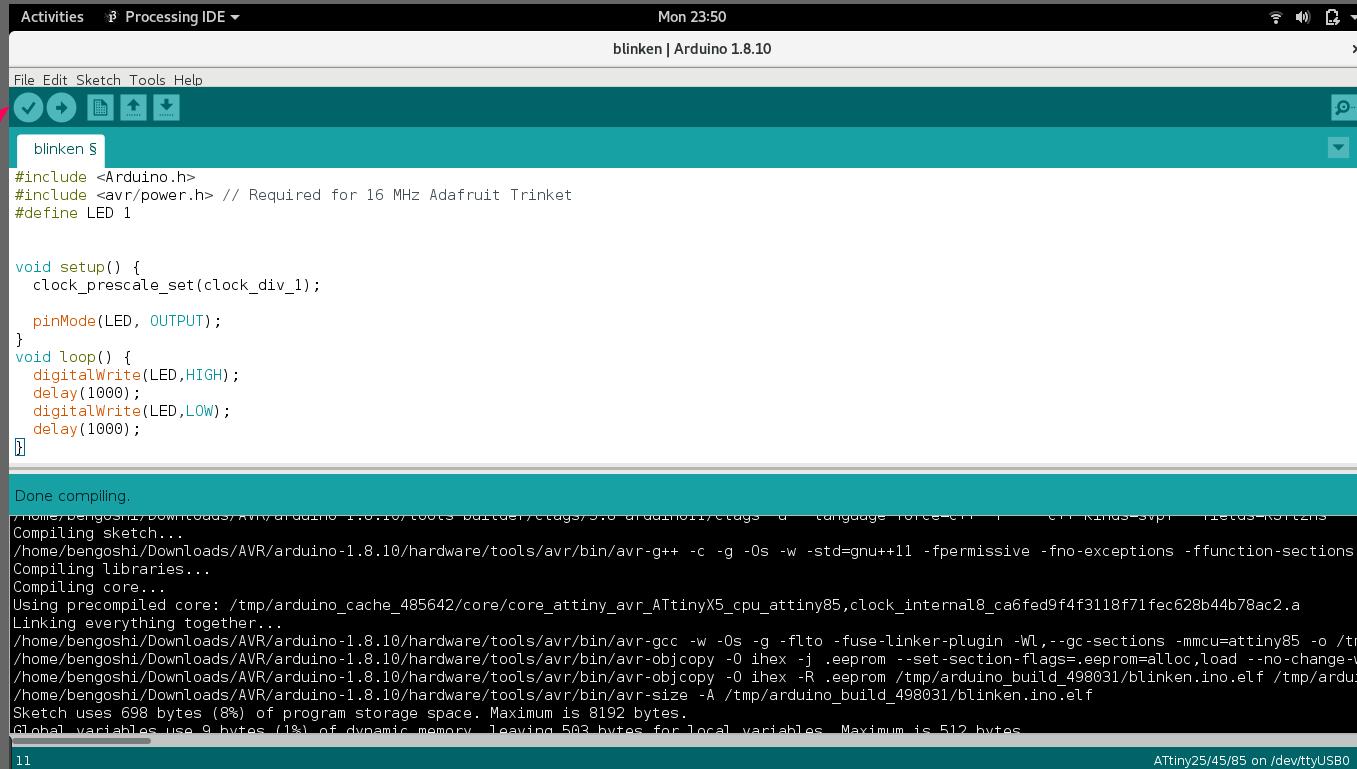
```
#include <Arduino.h>  
  
#include <avr/power.h> // Required  
for 16 MHz Adafruit Trinket  
  
#define LED 1  
  
void setup() {  
    clock_prescale_set(clock_div_1);  
    pinMode(LED, OUTPUT);  
}  
}
```

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



Code testen

hier klicken



```
#include <Arduino.h>
#include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#define LED 1

void setup() {
  clock_prescale_set(clock_div_1);

  pinMode(LED, OUTPUT);
}

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

Done compiling.

```
Compiling sketch...
/home/bengoshi/Downloads/AVR/arduino-1.8.10/hardware/tools/avr/bin/avr-g++ -c -g -Os -w -std=gnu++11 -fpermissive -fno-exceptions -ffunction-sections
Compiling libraries...
Compiling core...
Using precompiled core: /tmp/arduino_cache_485642/core/core_attiny_avr_ATtinyX5_cpu_attiny85,clock_internal8_ca6fed9f4f3l18f71fec628b44b78ac2.a
Linking everything together...
/home/bengoshi/Downloads/AVR/arduino-1.8.10/hardware/tools/avr/bin/avr-gcc -w -Os -g -flto -fuse-linker-plugin -Wl,--gc-sections -mmcu=attiny85 -o /tmp/arduino_build_498031/blinker.ino.elf
/home/bengoshi/Downloads/AVR/arduino-1.8.10/hardware/tools/avr/bin/avr-objcopy -O ihex -j .eeprom --set-section-flags=.eeprom=alloc,load -no-change-warnings -R .eeprom /tmp/arduino_build_498031/blinker.ino.elf /tmp/arduino_build_498031/blinker.elf
Sketch uses 698 bytes (8%) of program storage space. Maximum is 8192 bytes.
Global variables use 9 bytes (1%) of dynamic memory, leaving 503 bytes for local variables. Maximum is 512 bytes.
```

ATTINY25/45/85 on /dev/ttyUSB0



Code testen

Activities Processing IDE Mon 23:52
blinker | Arduino 1.8.10

File Edit Sketch Tools Help

blinker

```
#include <Arduino.h>
#include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#define LED 1

void setup() {
  clock_prescale_set(clock_div_1);

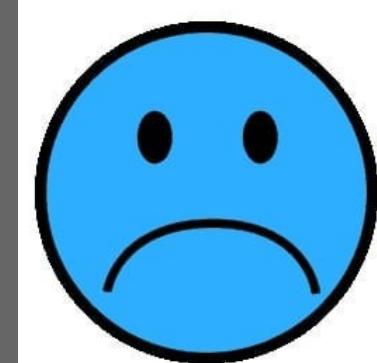
  pinMode(LED, OUTPUT)
}

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

expected ';' before ')' token
```

Generating function prototypes...
/home/bengoshi/Downloads/AVR/arduino-1.8.10/hardware/tools/avr/bin/avr-g++ -c -g -Os -w -std=gnu++11 -fpermissive -fno-exceptions -ffunction-sections
/home/bengoshi/Downloads/AVR/arduino-1.8.10/tools-builder/ctags/5.8-arduino1/ctags -u --language-force=c++ -f - --c++-kinds=svpf --fields=KSTtzns --l
Compiling sketch...
/home/bengoshi/Downloads/AVR/arduino-1.8.10/hardware/tools/avr/bin/avr-g++ -c -g -Os -w -std=gnu++11 -fpermissive -fno-exceptions -ffunction-sections
/home/bengoshi/Arduino/blinker/blinker.ino: In function 'void setup()':
blinker:10:1: error: expected ';' before ')' token
 ^
exit status 1
expected ';' before ')' token

10 ATtiny25/45/85 on /dev/ttyUSB0



Code testen

Activities Processing IDE ▾ Mon 23:58
blinker | Arduino 1.8.10

File Edit Sketch Tools Help

blinker

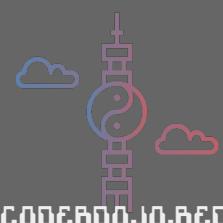
```
#include <Arduino.h>
#include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#define LED 1

void setup() {
  clock_prescale_set(clock_div_1);
  pinMode(LED, OUTPUT);
}
void loop() {
  digitalWrite(LED,HIGH);
  delay(1000);
  digitalWrite(LED,LOW);
  delay(1000);
}
```

Done compiling.

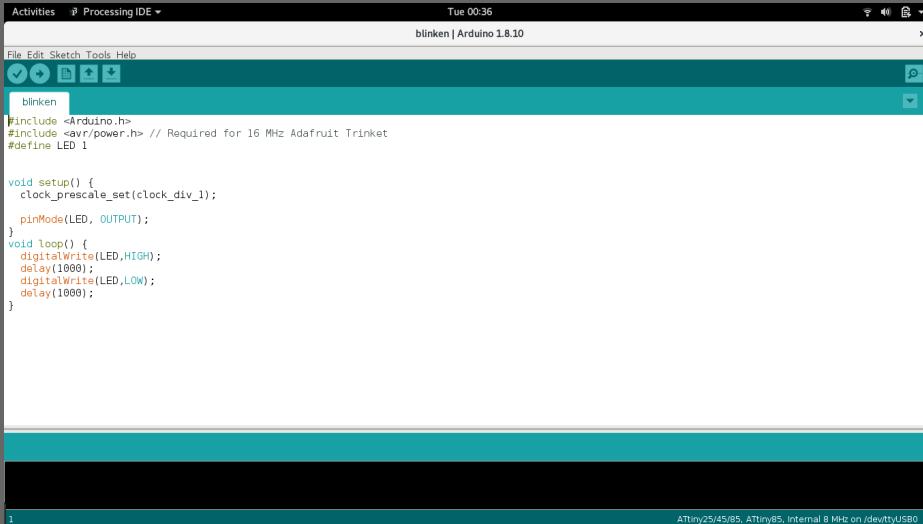
```
/home/bengoshi/Downloads/AVR/arduino-1.8.10/hardware/tools/avr/bin/avr-g++ -c -g -Os -w -std=gnu++11 -fpermissive -fno-exceptions -ffunction-sections
Compiling libraries...
Compiling core...
Using precompiled core: /tmp/arduino_cache_485642/core/core_attiny_avr_ATtinyX5_cpu_attiny85_clock_internal8_ca6fed9f4f3118f71fec628b44b78ac2.a
Linking everything together...
/home/bengoshi/Downloads/AVR/arduino-1.8.10/hardware/tools/avr/bin/avr-gcc -w -Os -g -flto -fuse-linker-plugin -WL,--gc-sections -mmcu=attiny85 -o /tmp/arduino_build_498031/blinker.ino.elf
/home/bengoshi/Downloads/AVR/arduino-1.8.10/hardware/tools/avr/bin/avr-objcopy -O ihex -j .eeprom --set-section-flags=.eeprom=alloc,load --no-change-w
/home/bengoshi/Downloads/AVR/arduino-1.8.10/hardware/tools/avr/bin/avr-objcopy -O ihex -R .eeprom /tmp/arduino_build_498031/blinker.ino.elf /tmp/arduino_build_498031/blinker.ino.elf
Sketch uses 698 bytes (8%) of program storage space. Maximum is 8192 bytes.
Global variables use 9 bytes (1%) of dynamic memory, leaving 503 bytes for local variables. Maximum is 512 bytes.
```

9 ATtiny25/45/85 on /dev/ttyUSB0



Einstellungen

- Tools →
- Board:
ATTiny25/45/85
- Prozessor:
ATTiny85
- Clock:
Internal 8MHz
- Port → testen
- Programmer:
USBasp

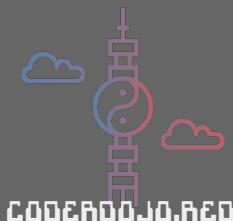
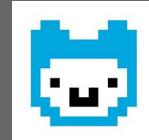


The screenshot shows the Arduino IDE interface with the following details:

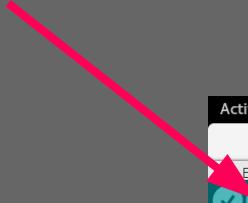
- Top bar: Activities, Processing IDE, Tue 00:36, blinken | Arduino 1.8.10
- File menu: File, Edit, Sketch, Tools, Help
- Sketch name: blinken
- Code area:

```
#include <Arduino.h>
#include <avr/power.h> // Required for 16 MHz Adafruit Trinket
#define LED 1

void setup() {
  clock_prescale_set(clock_div_1);
  pinMode(LED, OUTPUT);
}
void loop() {
  digitalWrite(LED,HIGH);
  delay(1000);
  digitalWrite(LED,LOW);
  delay(1000);
}
```
- Bottom status bar: ATTiny25/45/85, ATTiny85, Internal 8 MHz on /dev/ttyUSB0



ATTiny in den Programmer stecken und schreiben



```
Activities  i Processing IDE Mon 23:58
          Edit Sketch Tools Help
          blinken | Arduino 1.8.10
          x
          File Edit Sketch Tools Help
          blinken
          #include <Arduino.h>
          #include <avr/power.h> // Required for 16 MHz Adafruit Trinket
          #define LED 1

          void setup() {
            clock_prescale_set(clock_div_1);

            pinMode(LED, OUTPUT);
          }
          void loop() {
            digitalWrite(LED,HIGH);
            delay(1000);
            digitalWrite(LED,LOW);
            delay(1000);
          }

          Done compiling.

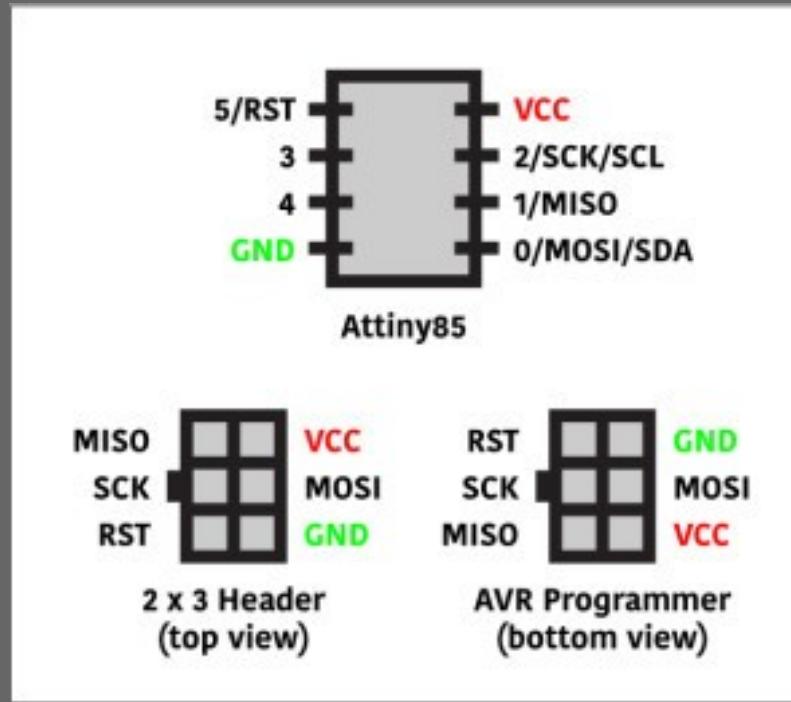
          /home/bengoshi/Downloads/AVR/arduino-1.8.10/hardware/tools/avr/bin/avr-g++ -c -g -Os -w -std=gnu++11 -fpermissive -fno-exceptions -ffunction-sections
          Compiling libraries...
          Compiling core...
          Using precompiled core: /tmp/arduino_cache_485642/core/core_attiny_avr_ATtinyX5_cpu_attiny85,clock_internal8_ca6fed9f4f3118f71fec628b44b78ac2.a
          Linking everything together...
          /home/bengoshi/Downloads/AVR/arduino-1.8.10/hardware/tools/avr/bin/avr-gcc -w -Os -g -flto -fuse-linker-plugin -WL,--gc-sections -mmcu=attiny85 -o /tmp/arduino_build_498031/blinden.elf
          /home/bengoshi/Downloads/AVR/arduino-1.8.10/hardware/tools/avr/bin/avr-objcopy -O ihex -j .eeprom --set-section-flags=.eeprom=alloc,load --no-change-w
          /home/bengoshi/Downloads/AVR/arduino-1.8.10/hardware/tools/avr/bin/avr-objcopy -O ihex -R .eeprom /tmp/arduino_build_498031/blinden.elf /tmp/arduino_
          /home/bengoshi/Downloads/AVR/arduino-1.8.10/hardware/tools/avr/bin/avr-size -A /tmp/arduino_build_498031/blinden.elf
          Sketch uses 698 bytes (8%) of program storage space. Maximum is 8192 bytes.
          Global variables use 9 bytes (1%) of dynamic memory, leaving 503 bytes for local variables. Maximum is 512 bytes.

          9
          ATtiny25/45/85 on /dev/ttyUSB0
```

Er bedankt sich, wenn alles richtig ist!



Löt-Freak? Hinweis für den eigenen Programmer



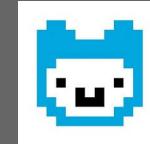
ATTiny einsetzen

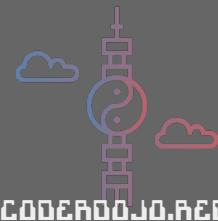
Batterie einschalten



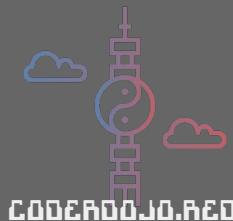
CODEcademy

Blinkt die LED? - wenn ja, dann





Mit Code spielen



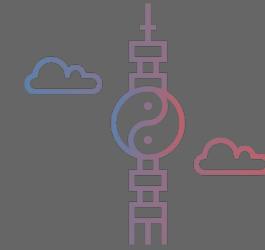
„großen“ Code implementieren

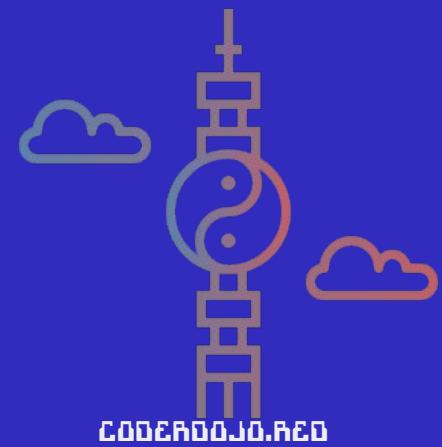
- siehe
<https://github.com/dermicha/tinyPixelFun/>



Und jetzt?

- wöchentliches Online-CoderDojo
→ coderdojo.red
- Jugend hackt
 - offenes Lab oder Workshops einmal im Monat dienstags
 - Workshops einmal im Monat samstags





CODEROOJO.RED