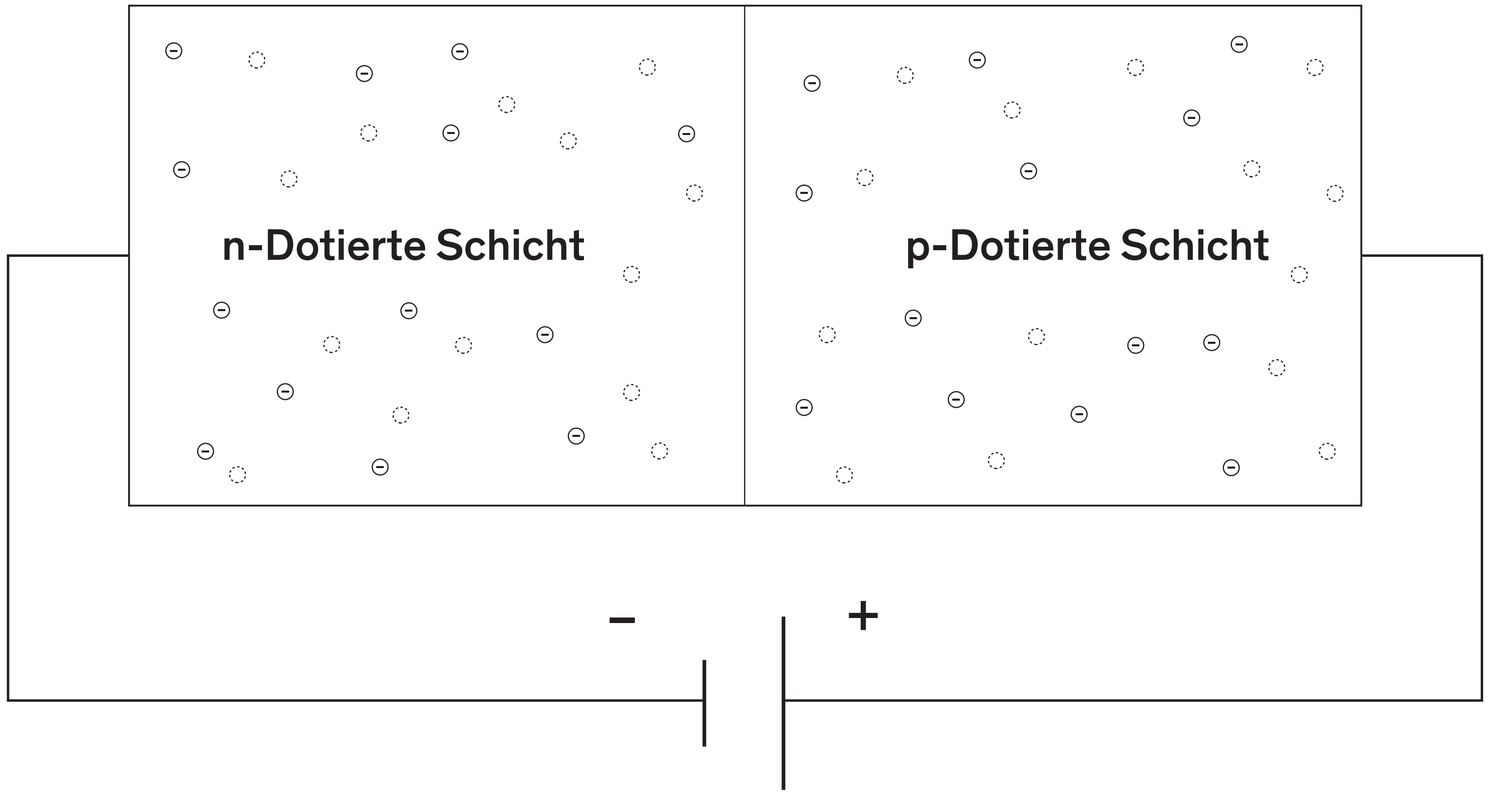
Individuelles Feedback folgt

Aufgabe 4

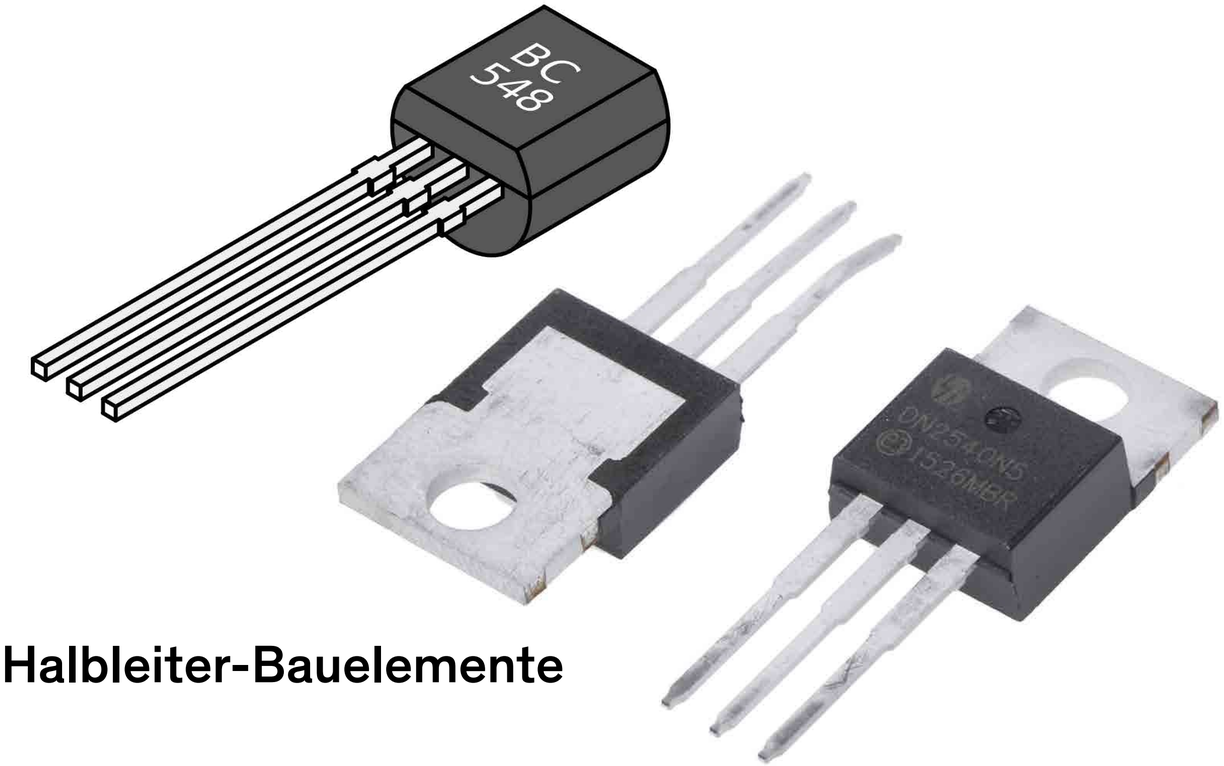
Recap
Multimeter
Serielle Kommunikation
ISP & ICSP
UART
I2C
SPI
Arduino Kommunikation
MQTT + Sensoren
API / Datenbankaufrufe
Special Topics

Recap

Halbleiter



Halbleiter

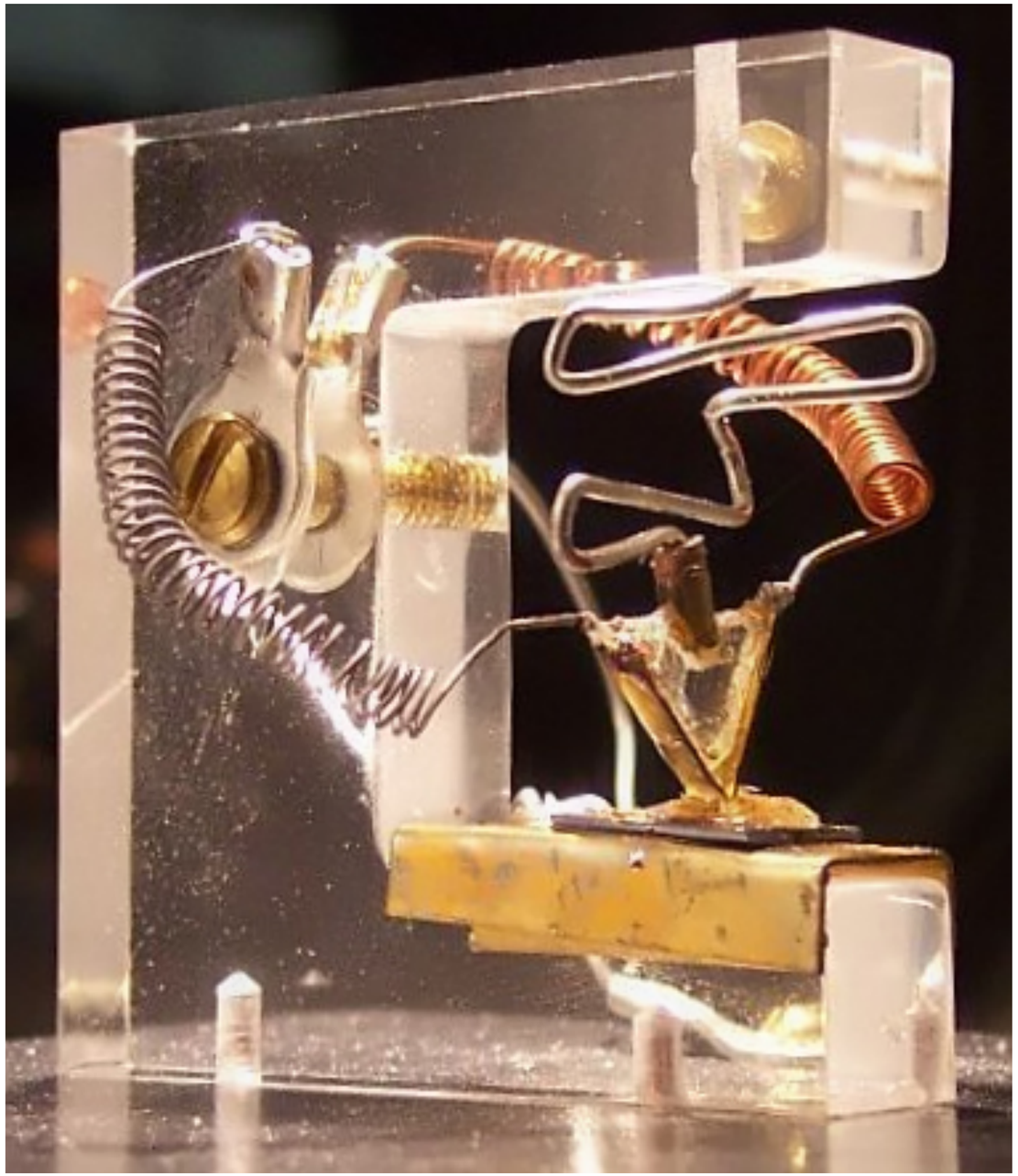


Halbleiter-Bauelemente

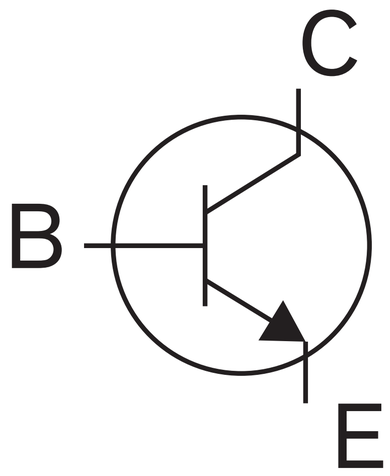
"Elektronischer Schalter"

Steuern und Verstärken von Strömen

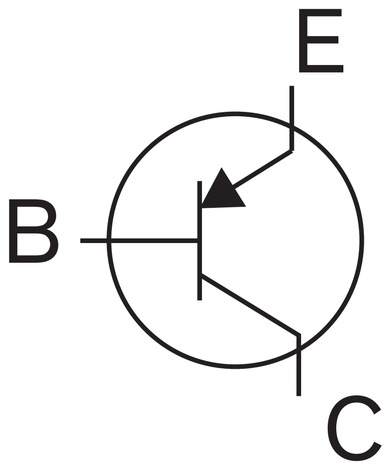
Verschiedene Arten:
– Bipolare Transistoren
– Feldeffekttransistoren



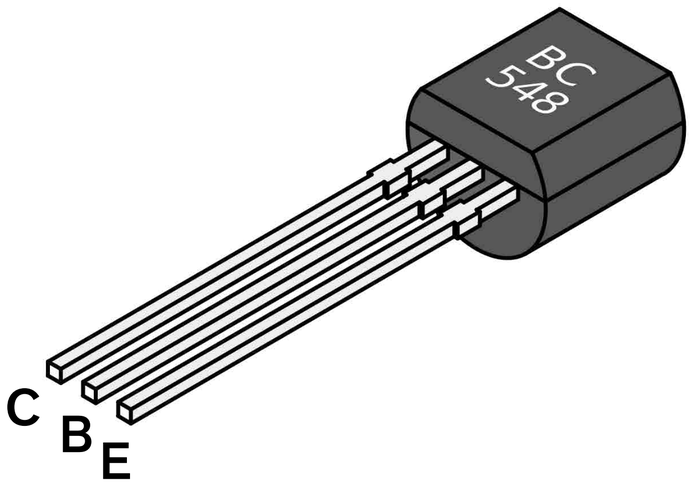
Transistoren



npn-Transistor



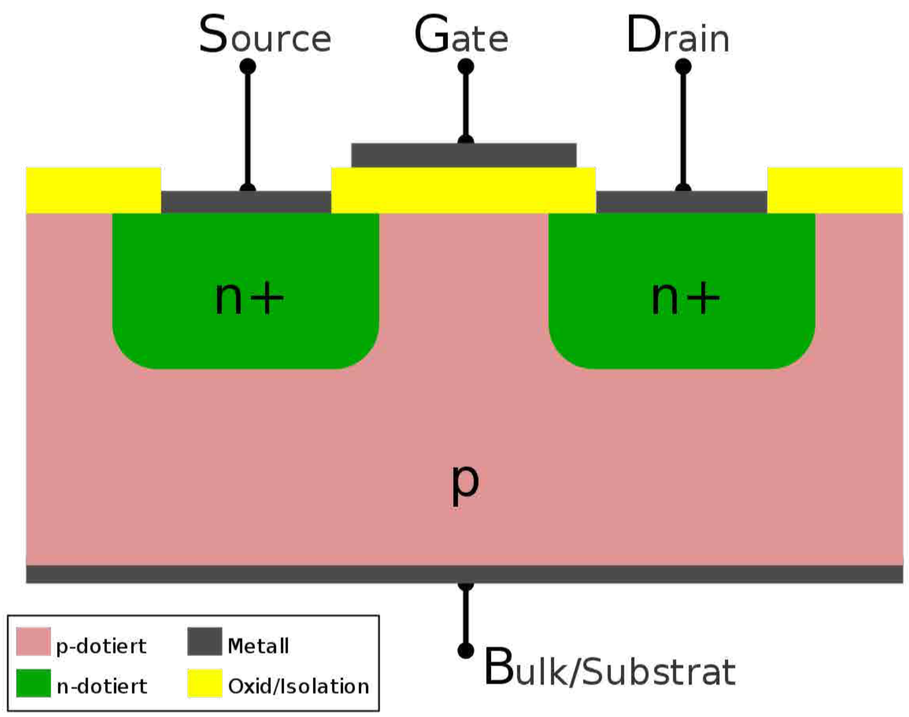
pnp-Transistor

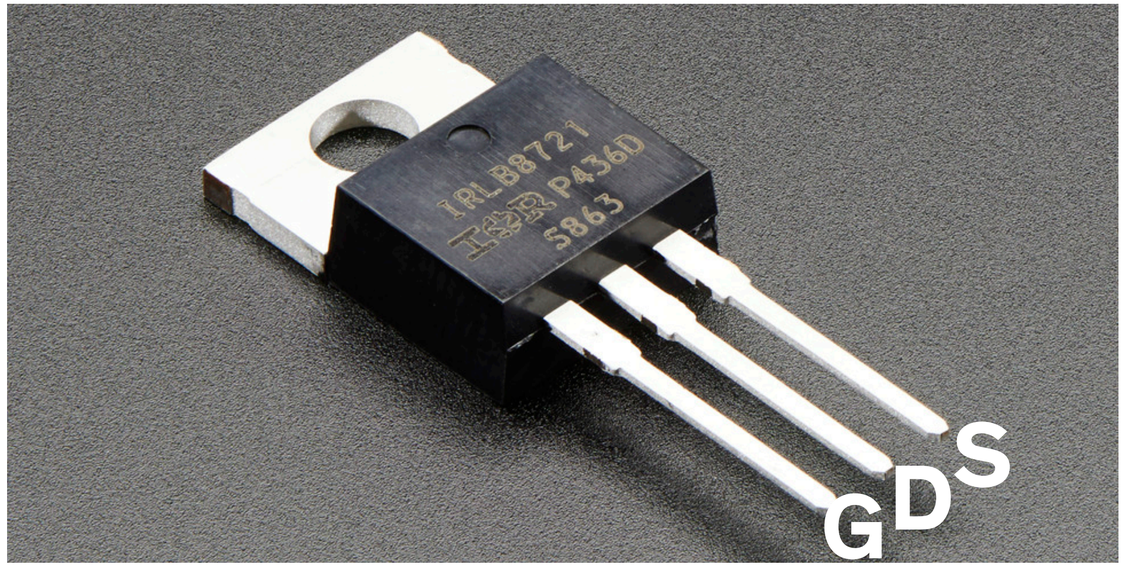


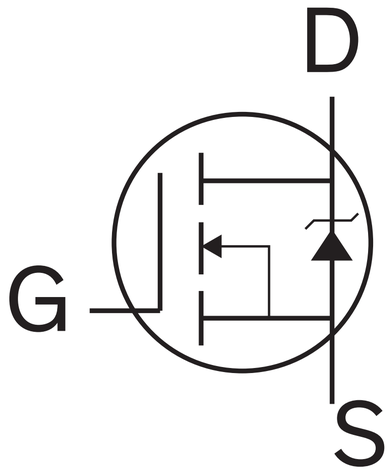
Kein Strom an Basis B → Transistor sperrt = Unendlich großer Widerstand

Strom an Basis B → Transistor leitet = Elektrisch regelbar, sehr kleiner Widerstand

Bipolartransistoren





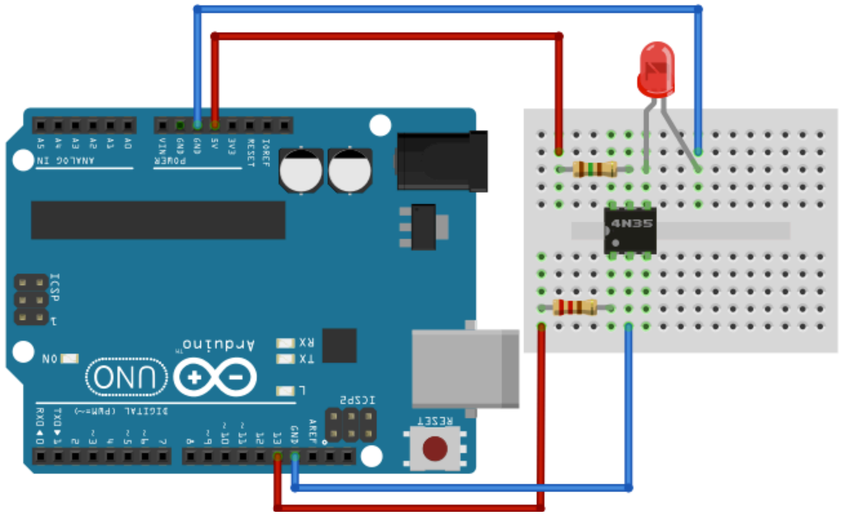


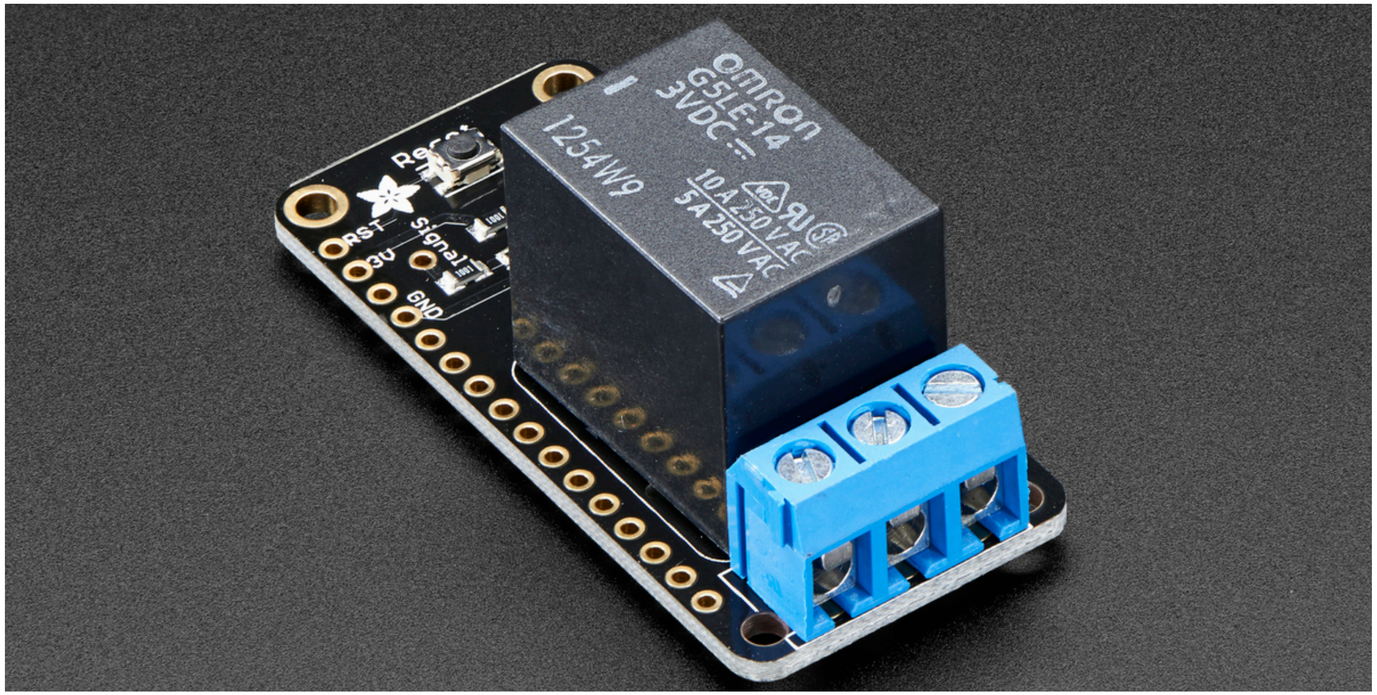
Kein Spannung am Gate G → Transistor sperrt = Unendlich großer Widerstand

Spannung am Gate G → Transistor leitet = Elektrisch regelbar, sehr kleiner Widerstand

Feldeffekt-Transistoren

Masseschleifen / "Brummschleifen"
Sicherheit
Schutzmaßnahme
Vermeidung von Potentialverschiebungen





Galvanische Trennung

Vereinfacht die Darstellung komplexer Abläufe

Verhindert "Spaghetti-Code"

Definierte Programmzustände

Einfache Nachvollziehbarkeit

(Endlicher) Zustandsautomat

```
void loop() {
  switch(state) {
    case EINGABE:
      // unser code hier
    break;
    case GRUENE_LED:
      // code für grüne led
    break;
    case GELBE_LED:
      // code für gelbe led
    break;
    case ROTE_LED:
      // code für rote led
   break;
  }
}
```

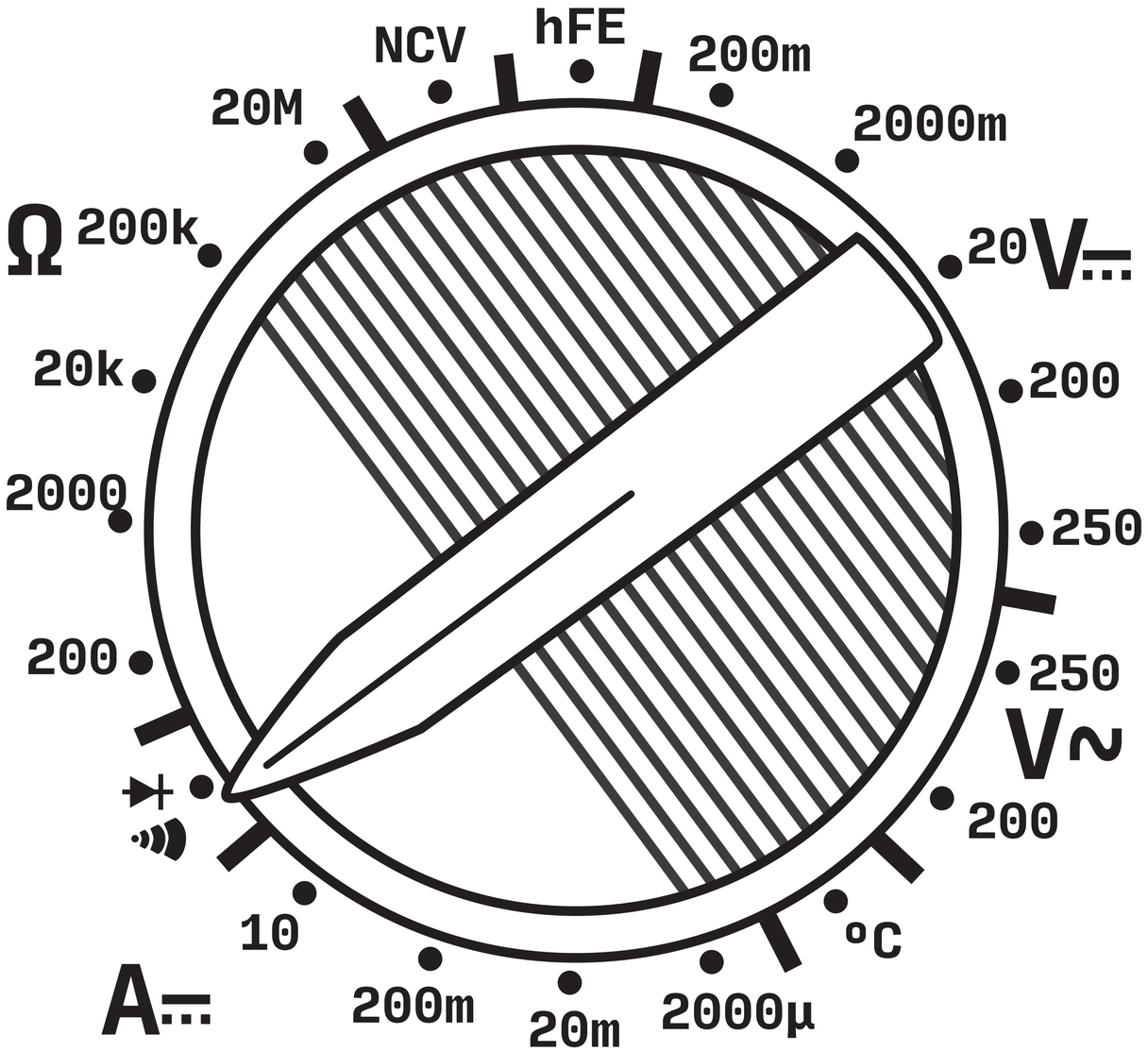
(Endlicher) Zustandsautomat

VOLTCRAFT
VC150-1
HOLD
POWER
NCV
hFE
200m
2000m
20M
20 V
Ω 200k
200
20k
250
2000
250
V~
200
200
°C
10
200m
20m
2000µ
A
FUSED
10A/250V
MAX10sec
EACH15min
COM
FUSED
200mA/250V
10A
CE

VOLTCRAFT
POWER
HOLD
NCV
hFE
200m
20M
2000m
Ω 200k
20 V
20k
200
2000
250
200
250
V~
200
10
°C
A
200m
20m
2000µ
V
CATIII 250V
FUSED
hFE
°C
mAΩ
10A
CE

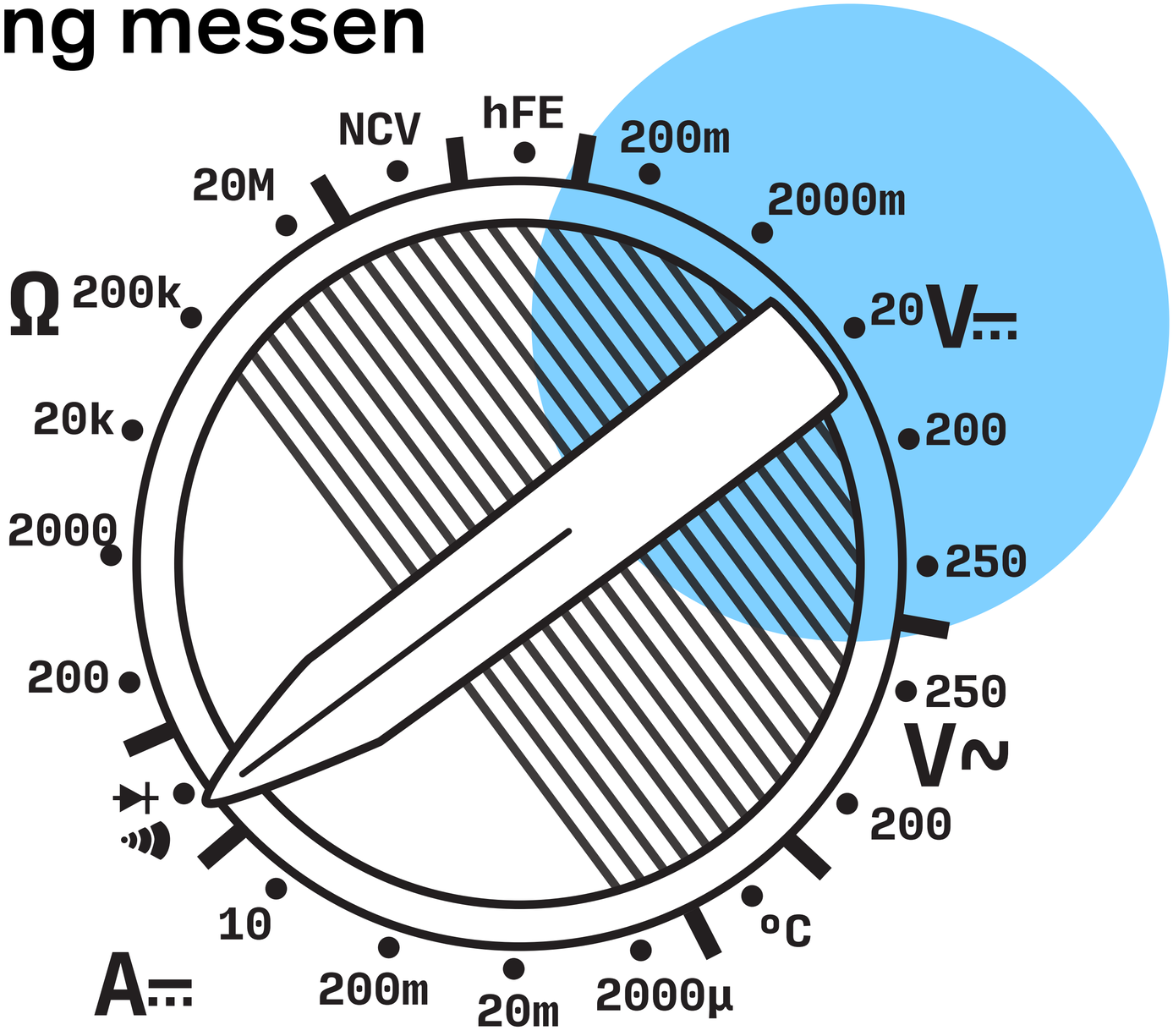
Multimeter

Übersicht



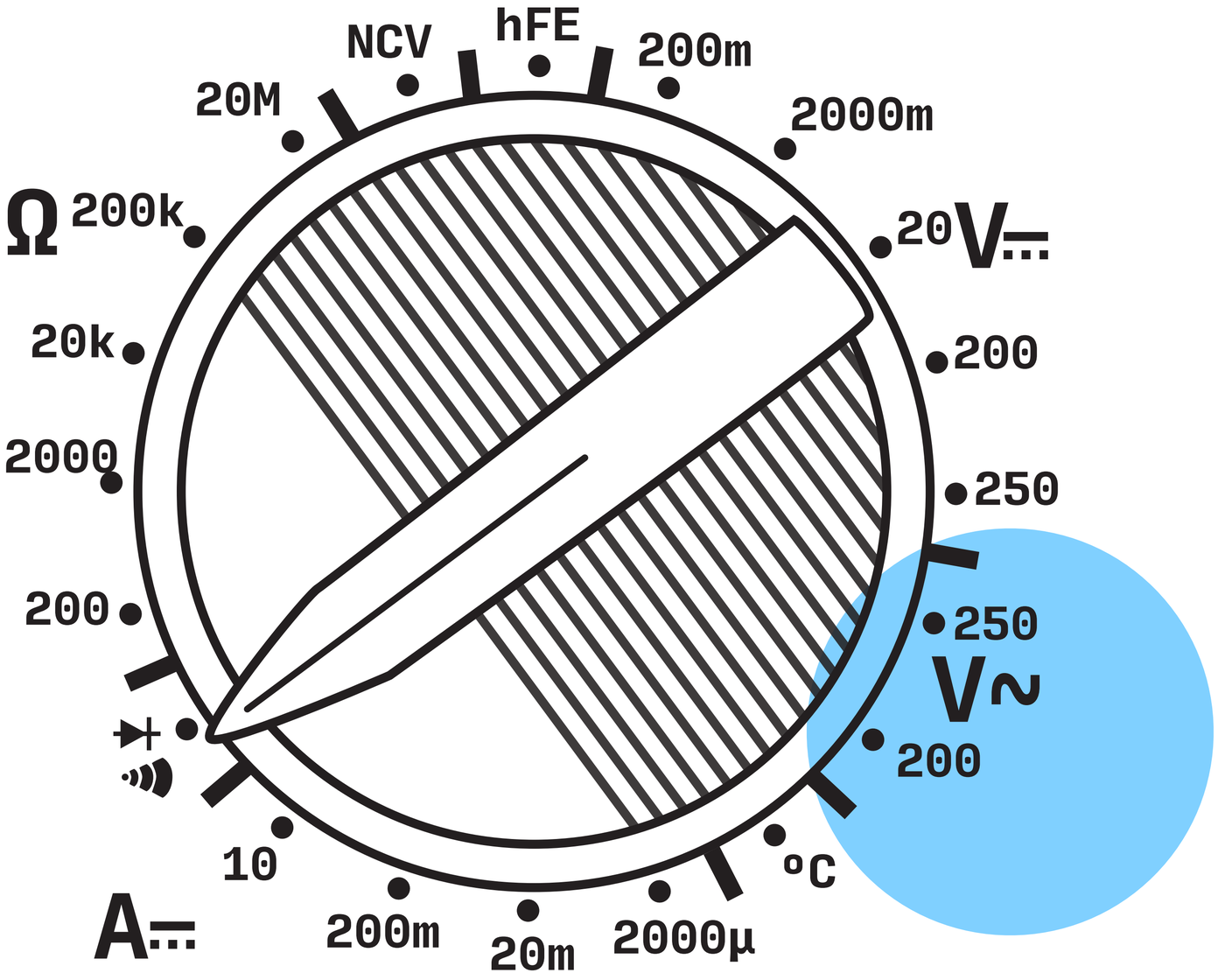
Multimeter

Gleichspannung messen



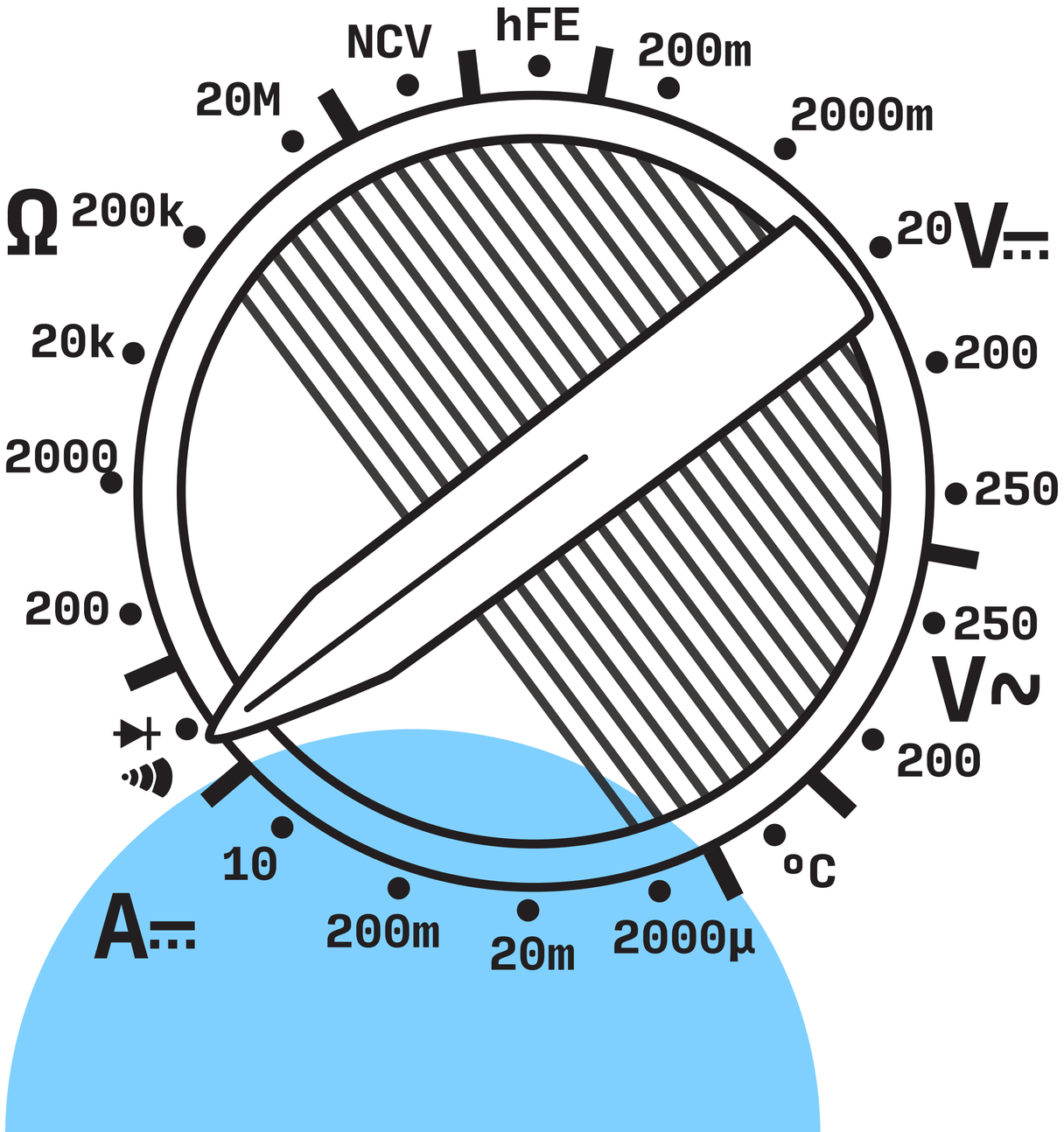
Multimeter

Wechselspannung messen

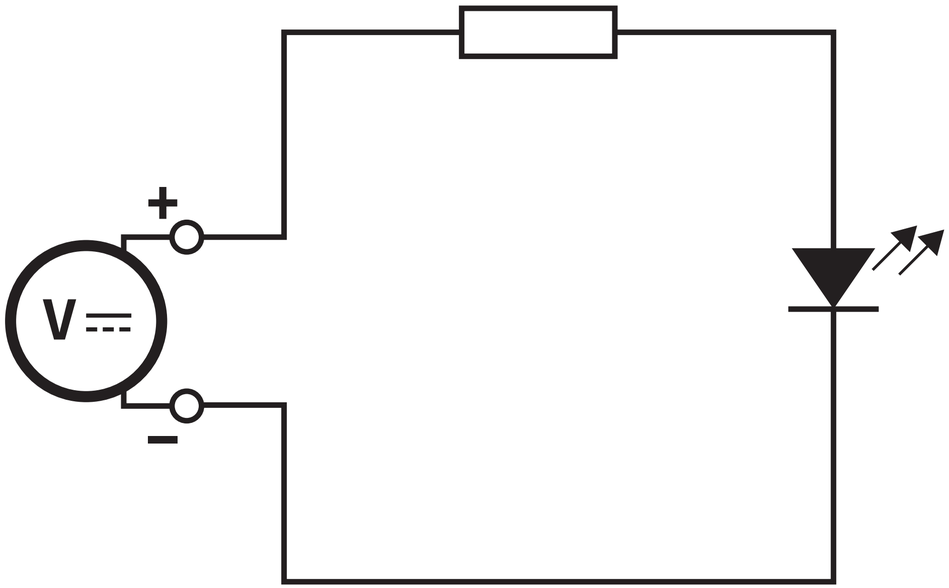


Multimeter

Stromintensität messen

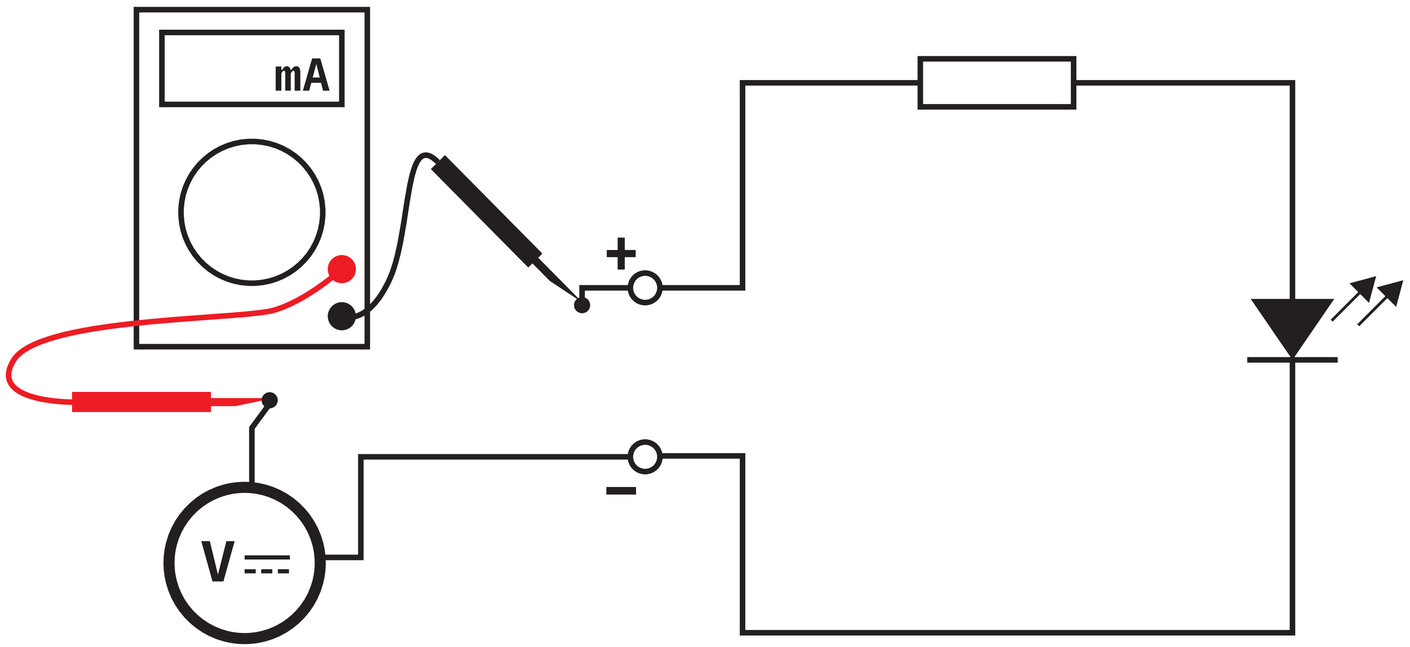


Multimeter



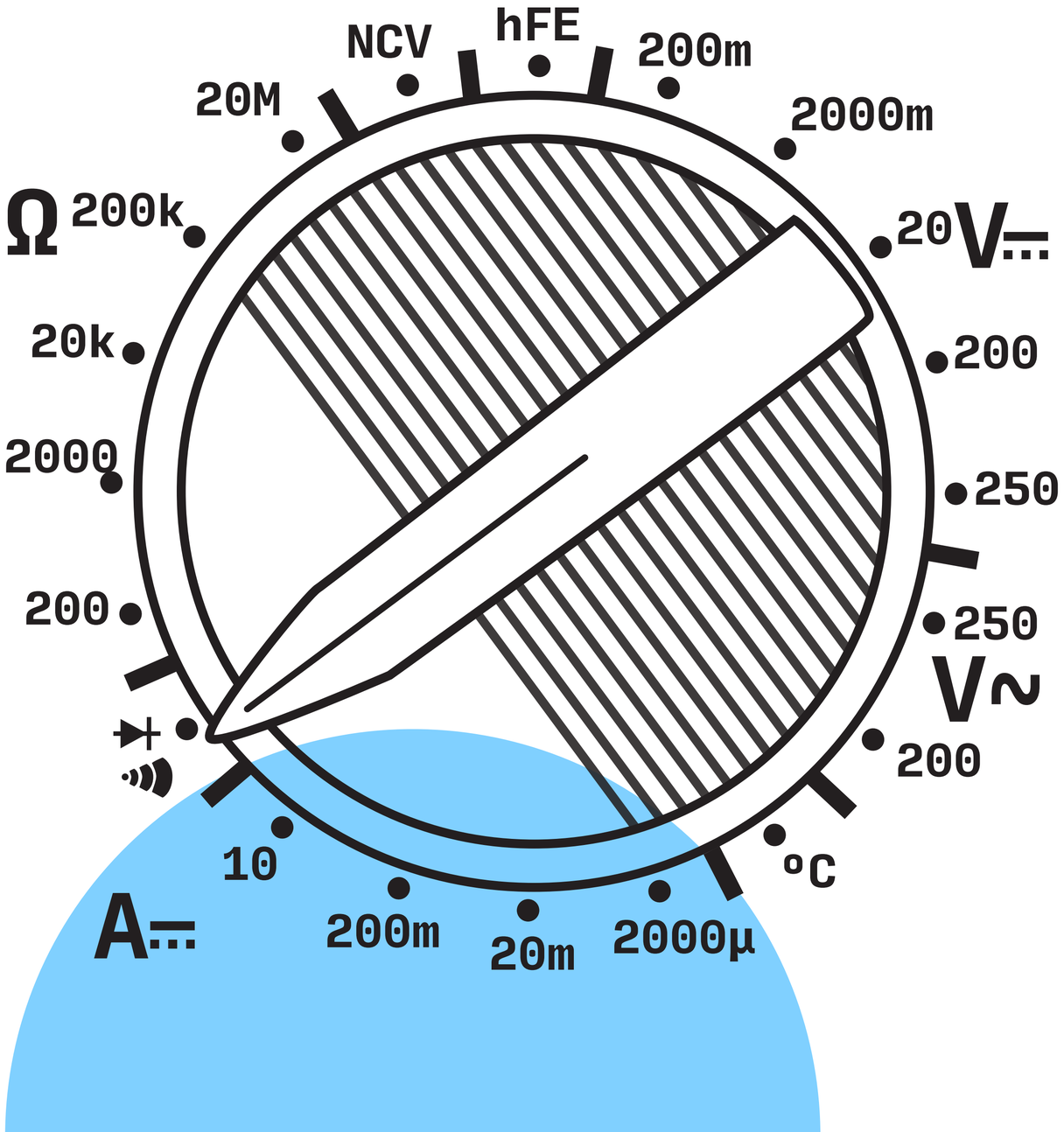
Multimeter

Multimeter "In-Line" mit dem Schaltkreis



Multimeter

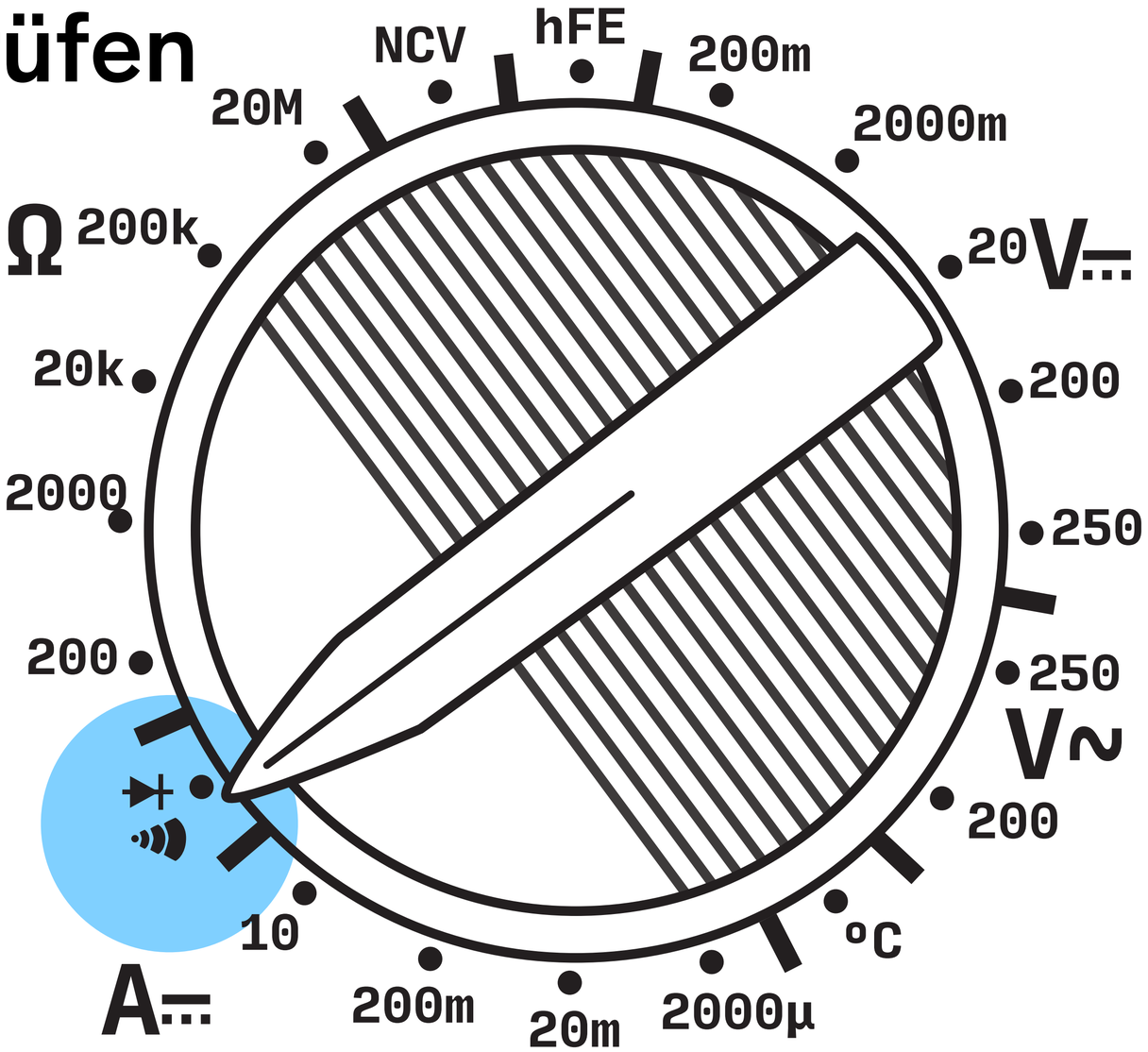
ab 5 Ampere nicht länger als 1 Minute am Stück messen



Multimeter

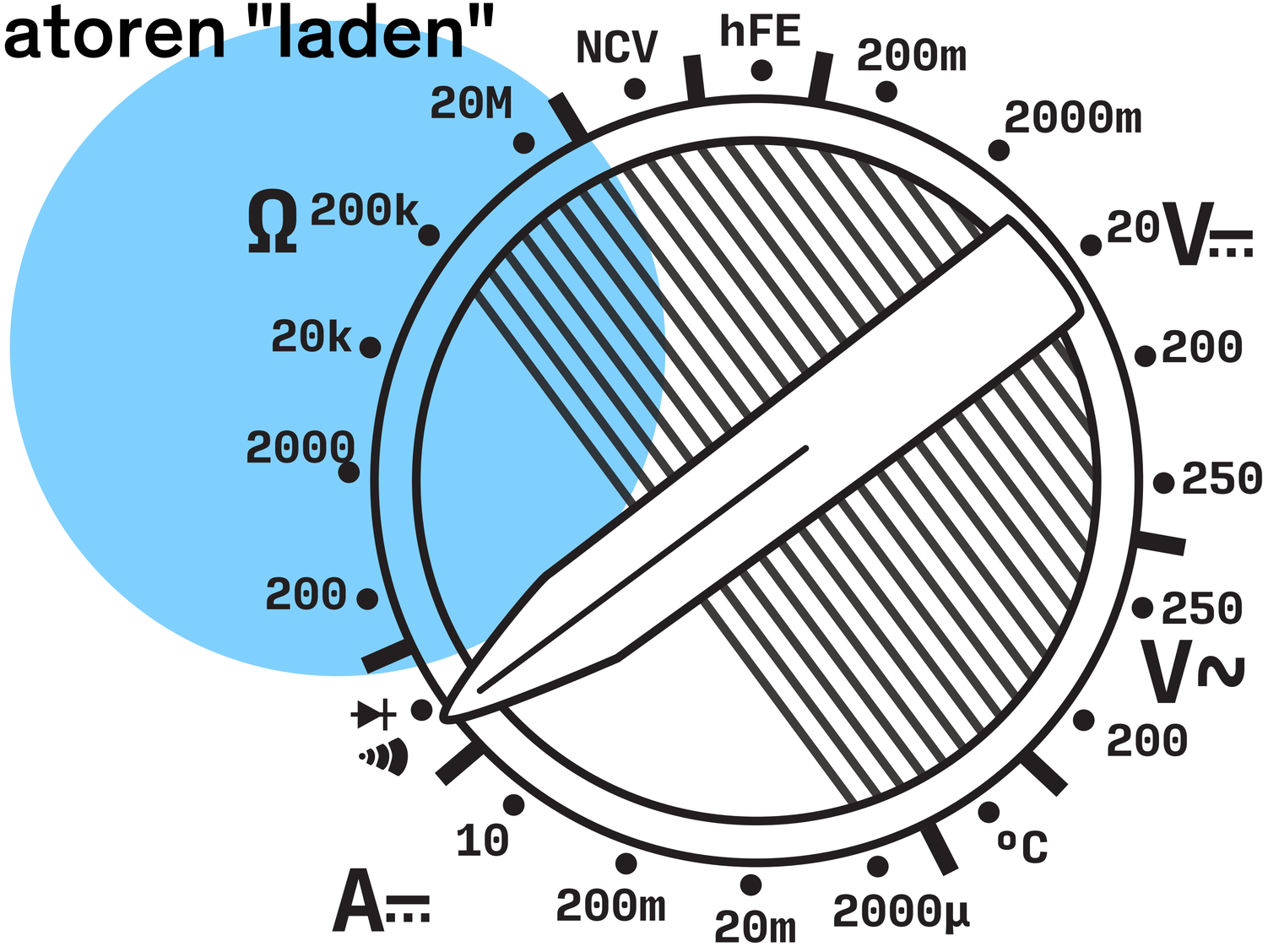
Physical Computing

Durchmessen
Kontinuität prüfen



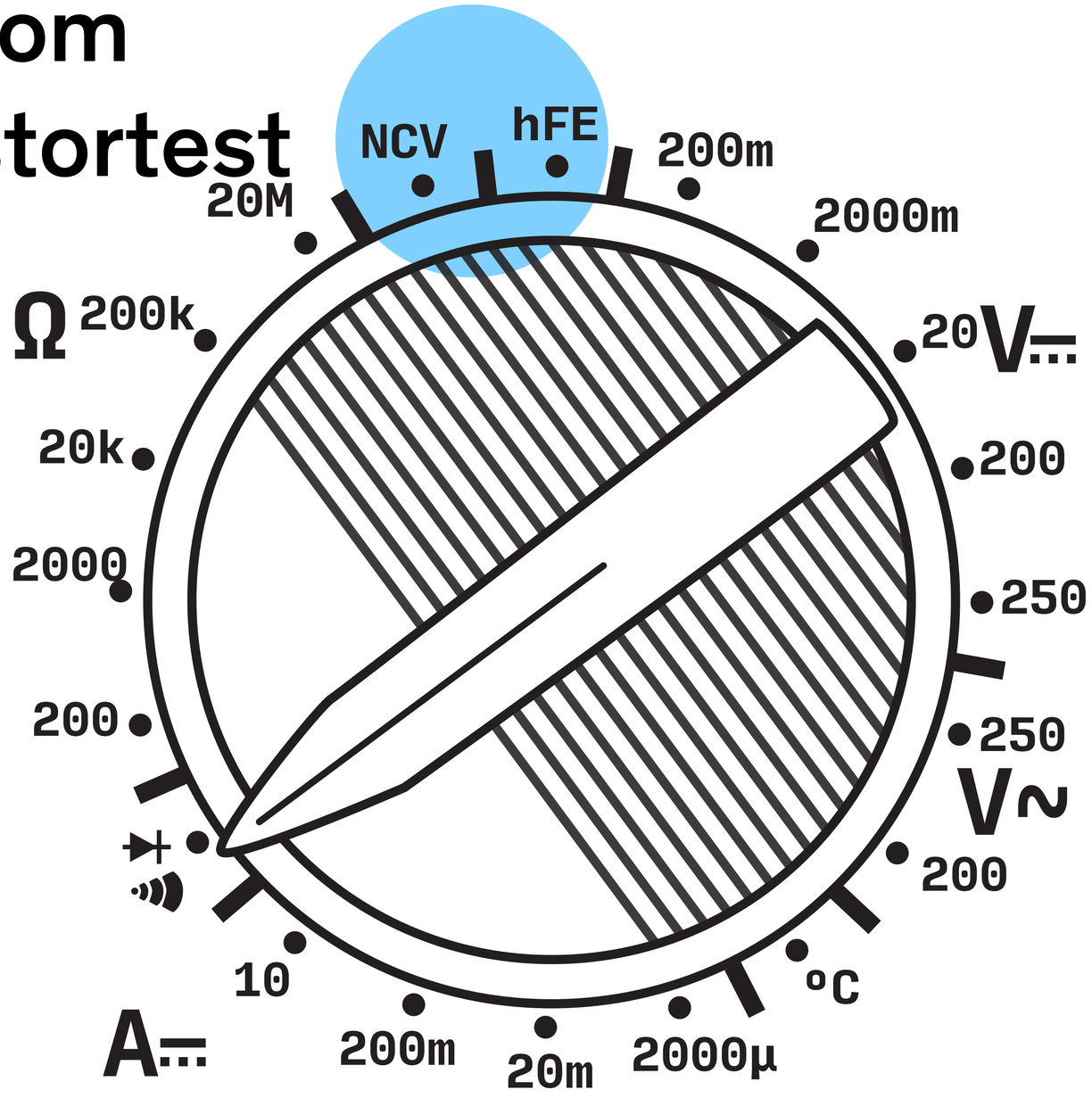
Multimeter

Widerstand testen
Kondensatoren "laden"

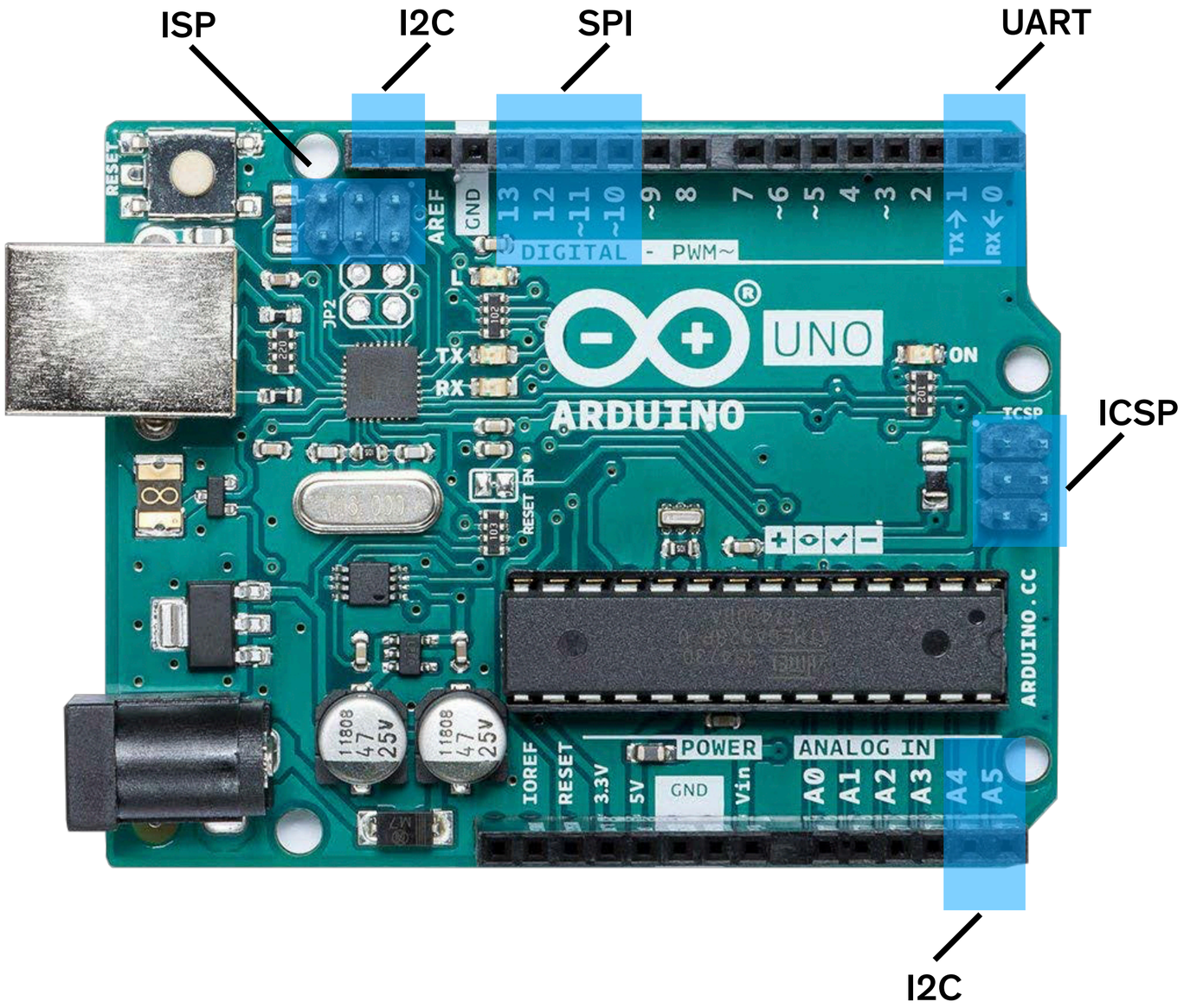


Multimeter

NCV = AC Strom
hFE = Transistortest



Multimeter



Arduino beherrscht verschiede Kommunikationsprotokolle

– UART
– SPI
– I2C

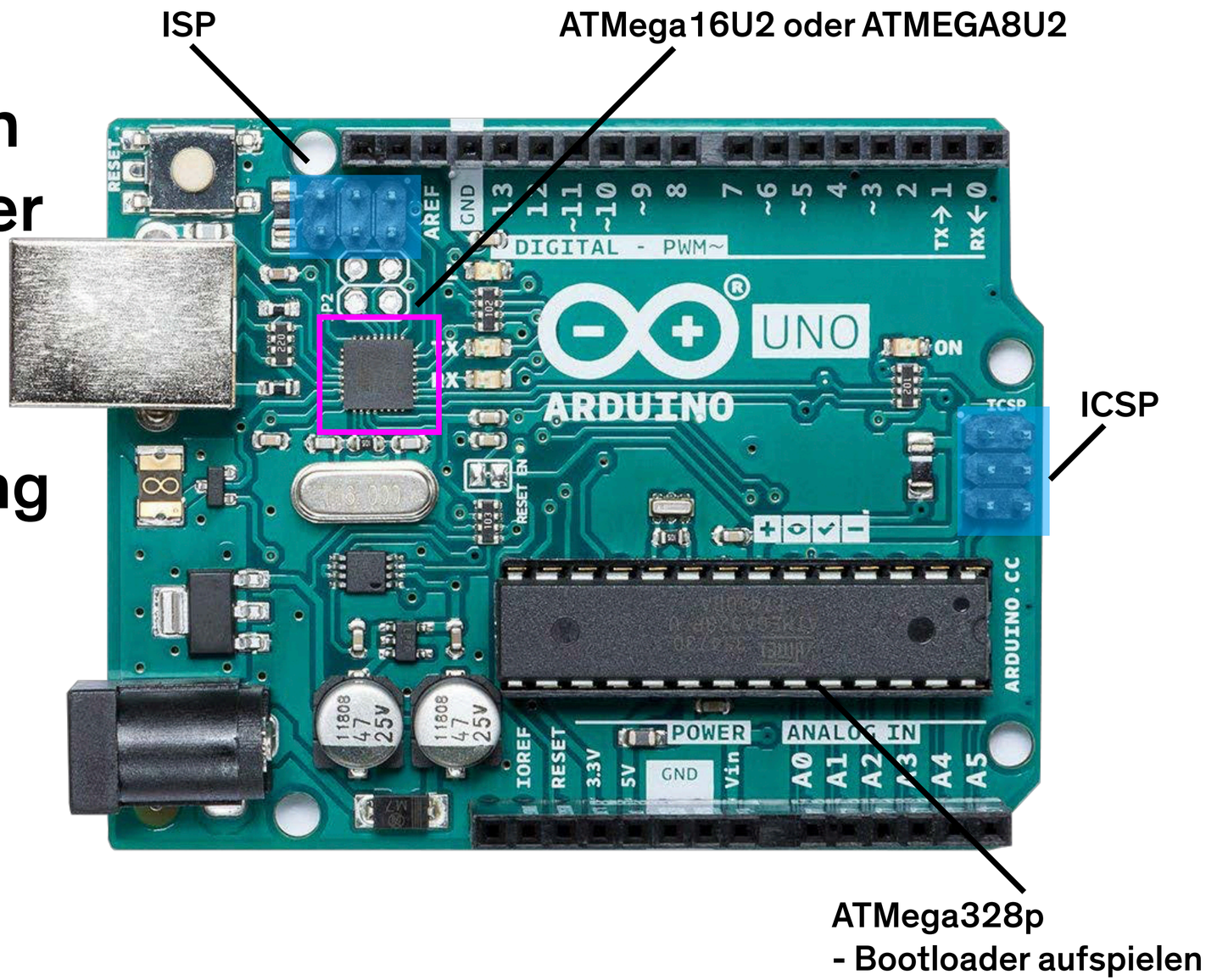
Serielle Kommunikation

In-System Programming

Aufspielen von Programmen
z.B. via USB oder ISP-Header

In-Circuit Serial Programming

Ändern des Bootloaders
Flash-Device erforderlich



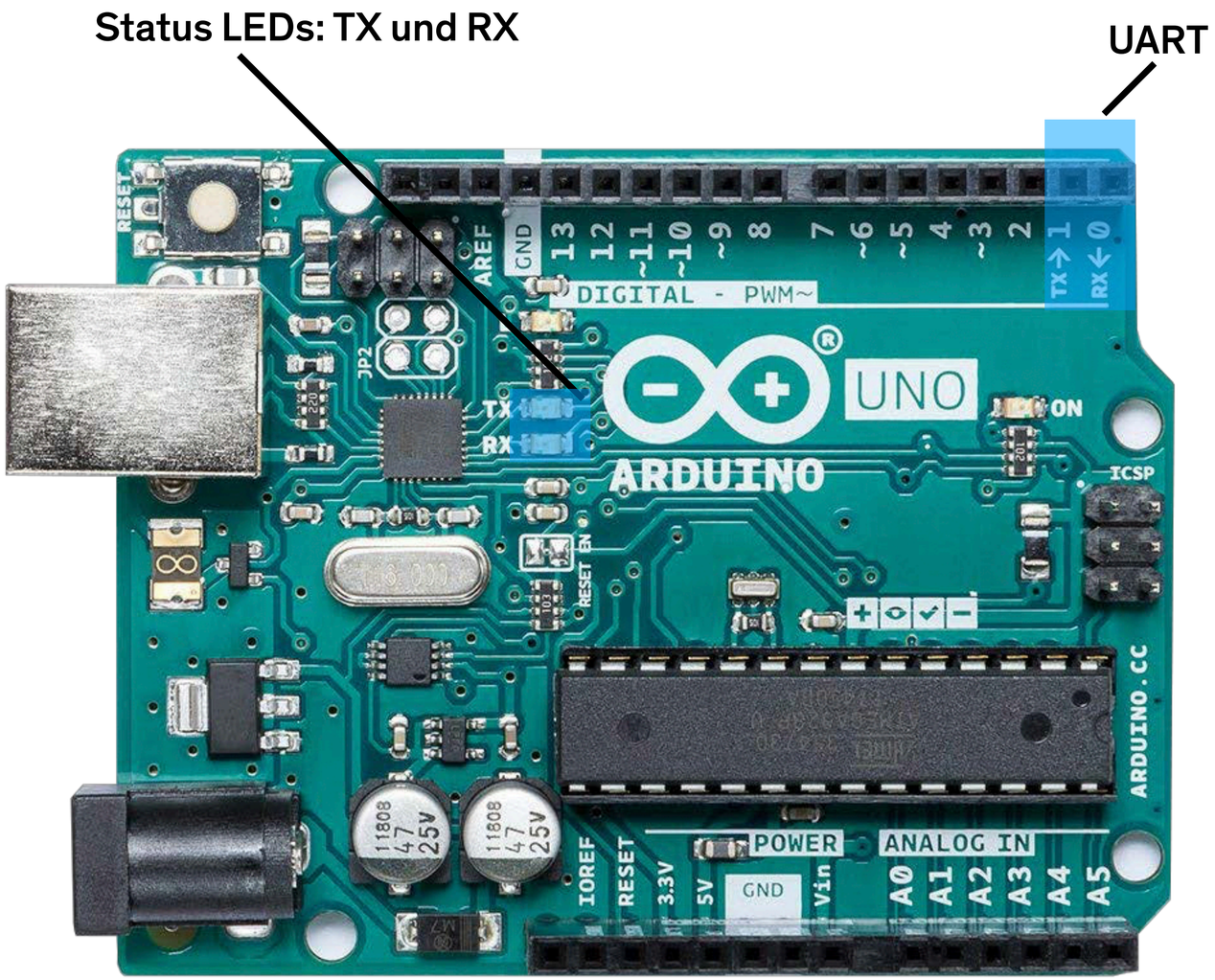
ISP & ICSP

Universal Asynchronous Recept and Transmission

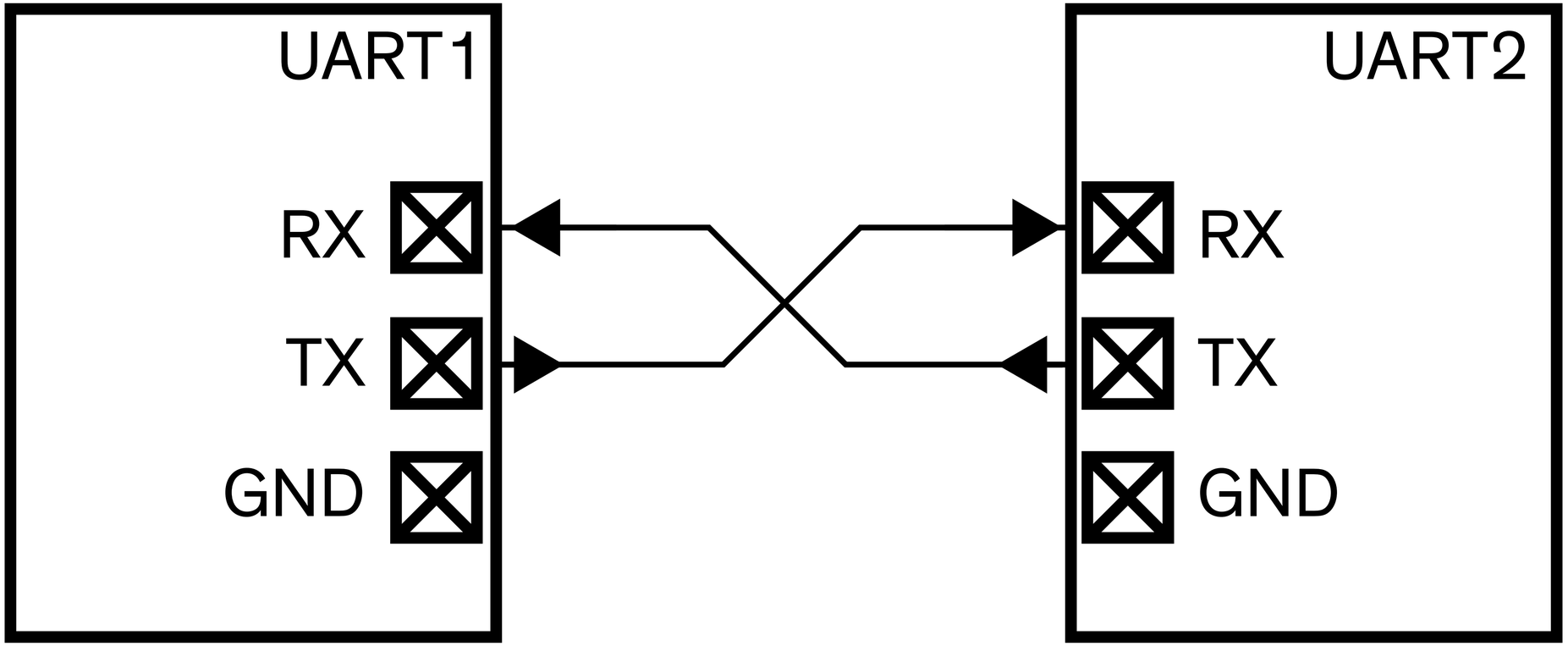
TX = Transmission
RX = Receive

Simple Kommunikation ohne Clock-Signal

Wird auch für serielle Kommunikation über USB verwendet

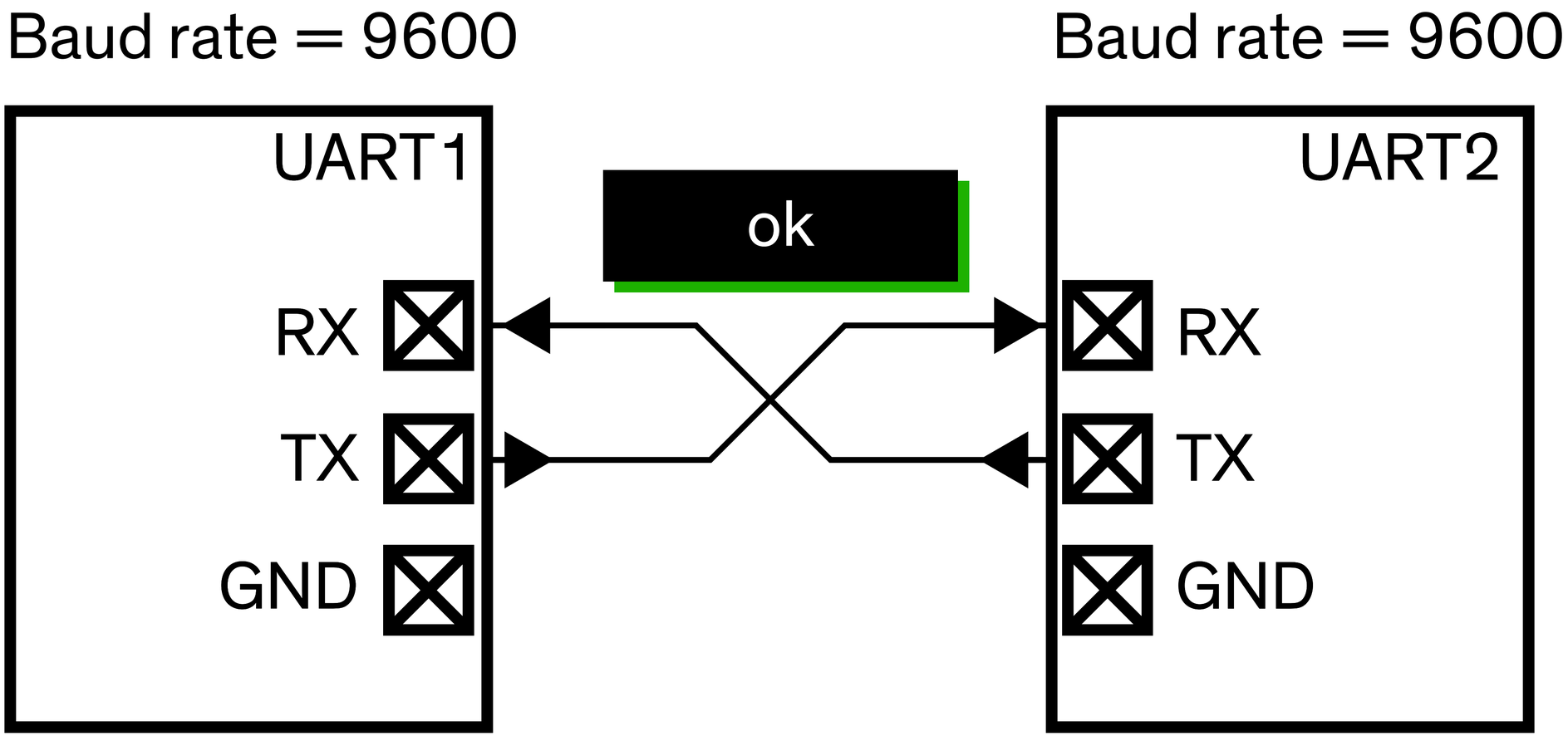


UART



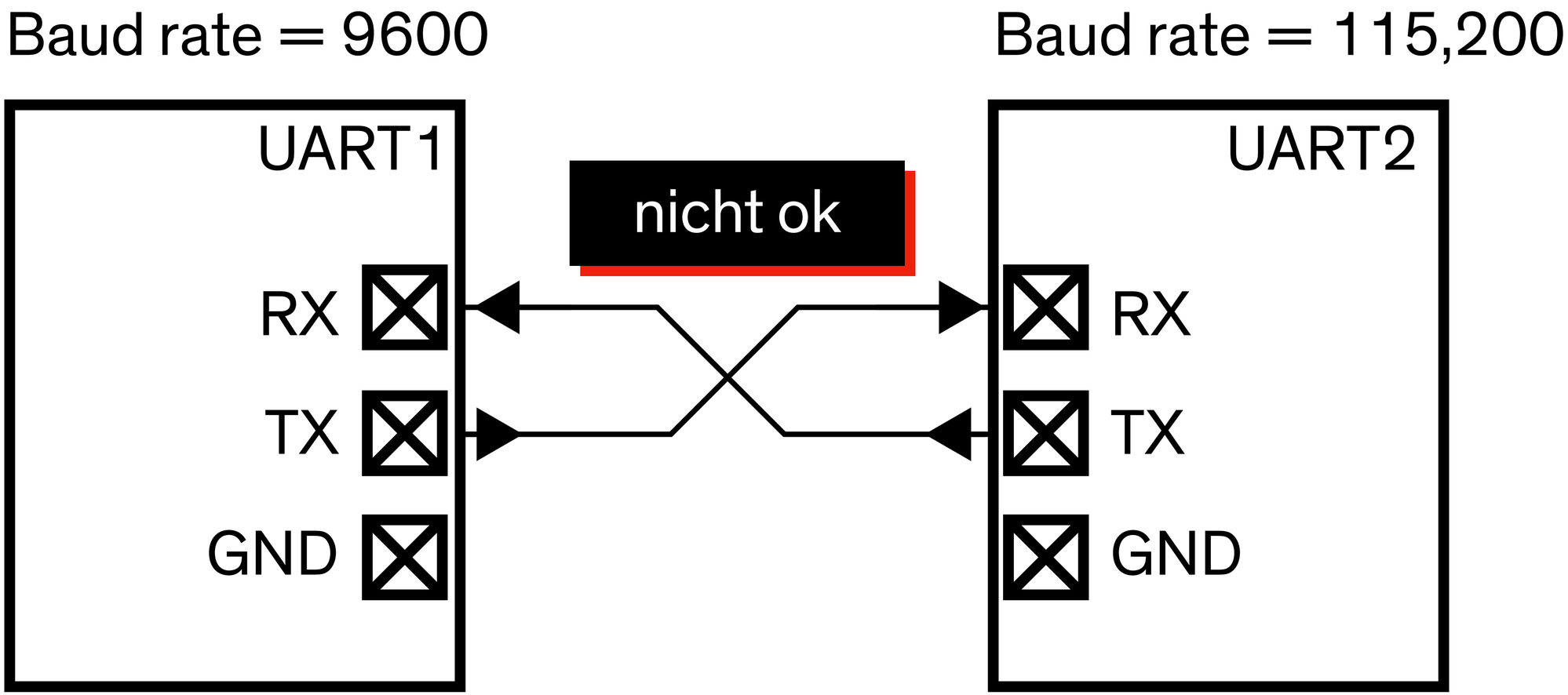
UART

Bits (Symbole) pro Sekunde

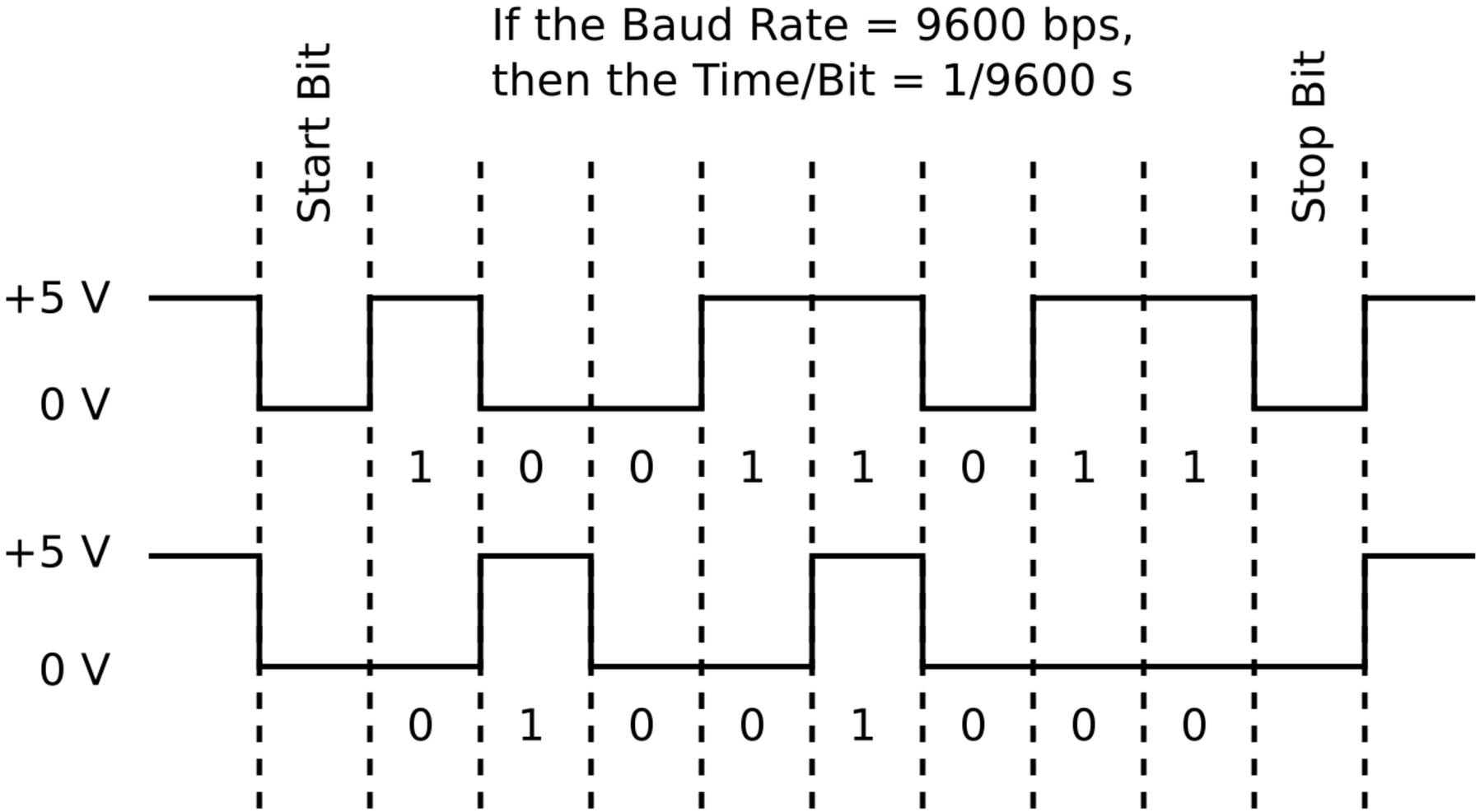


UART

Bits (Symbole) pro Sekunde



UART


If the Baud Rate = 9600 bps,
then the Time/Bit = 1/9600 s
Start Bit
Stop Bit
+5 V
0 V
1 0 0 1 1 0 1 1
+5 V
0 V
0 1 0 0 1 0 0 0

UART

Vorteile
– Simpel aufzubauen (z.B. zwei Arduinos miteinander kommunizieren lassen)
– Online gut dokumentiert
– Braucht keine Clock

Nachteile
– Ein Datenpaket (Frame) kann nur 9 Bits lang sein
– Overhead durch Start/Stopp Bits
– Nur eine *Master / Slave* Konfiguration
→ Besser: *Controller* und *Peripherie*
– Baud rates müssen gleich sein (10% Abweichung ok)
– Langsam

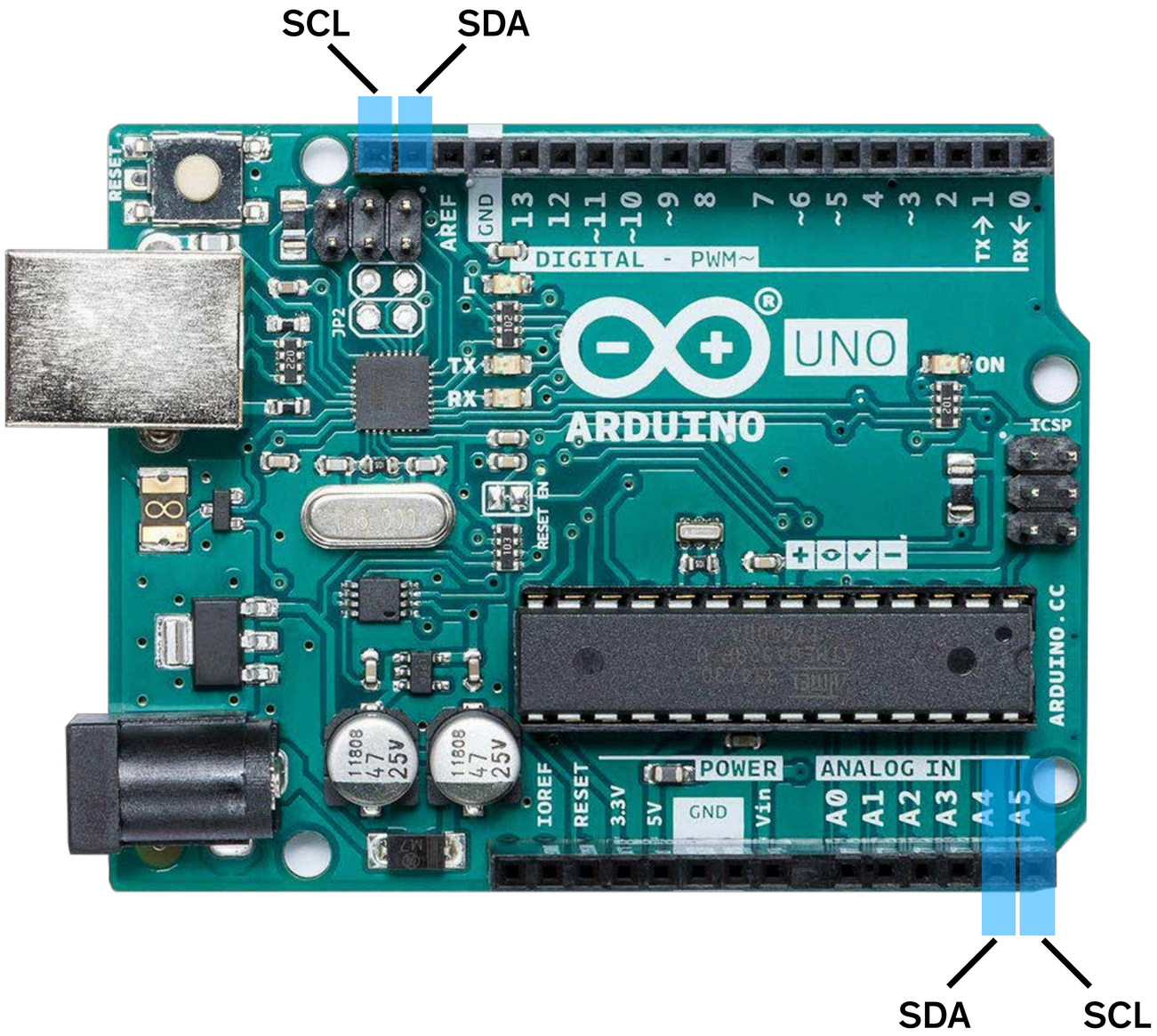
UART

I2C oder I²C
Inter-Integrated Circuit

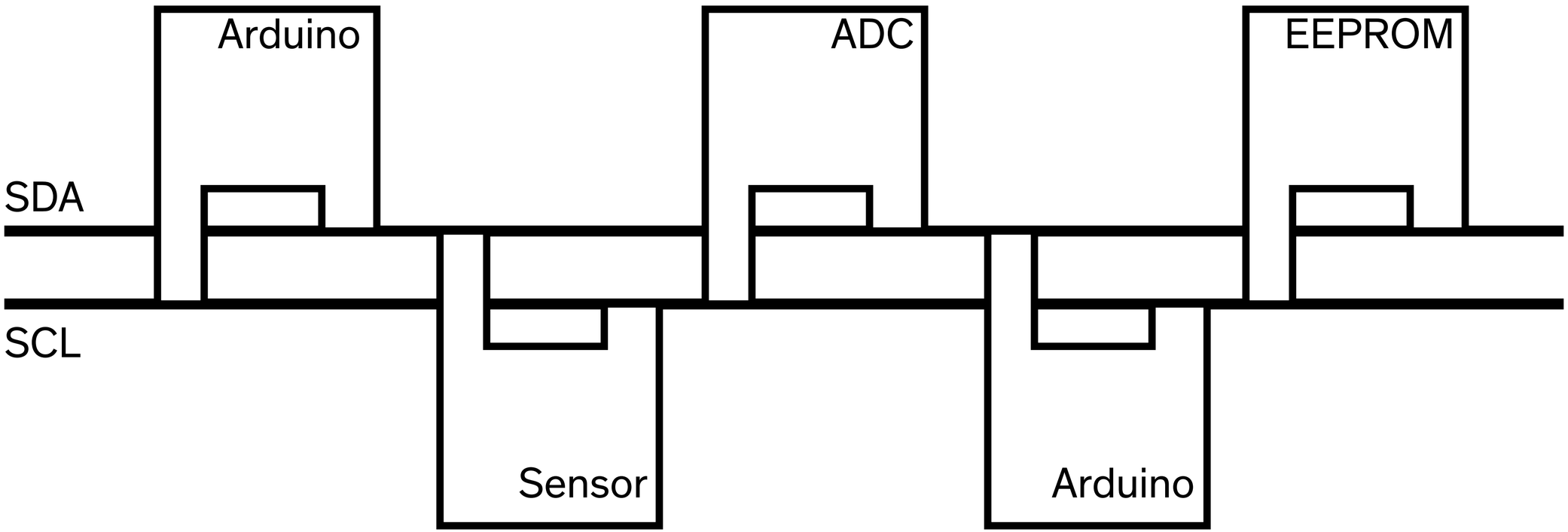
SCL = Signal Clock
SDA = Signal Data

Bis zu 128 Peripherien pro Controller

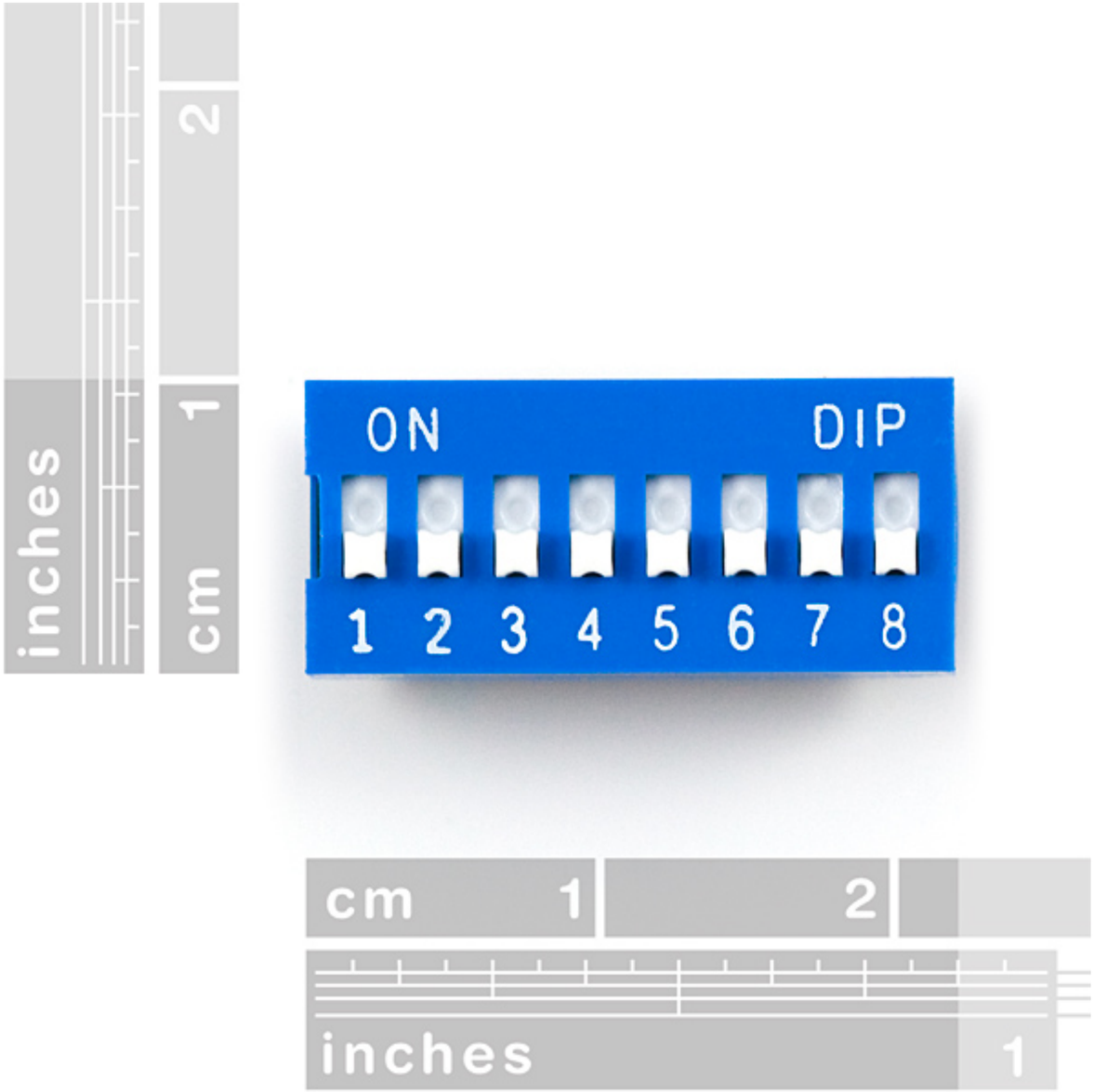
Address-System und Shared Bus



I2C



I2C

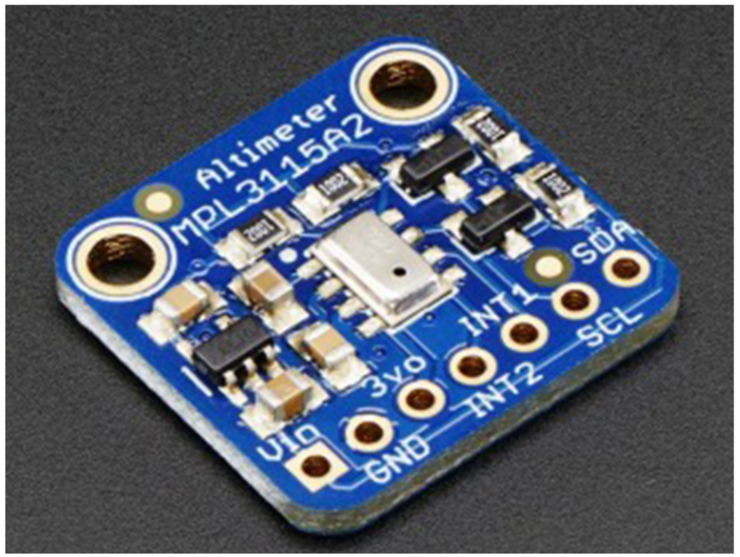


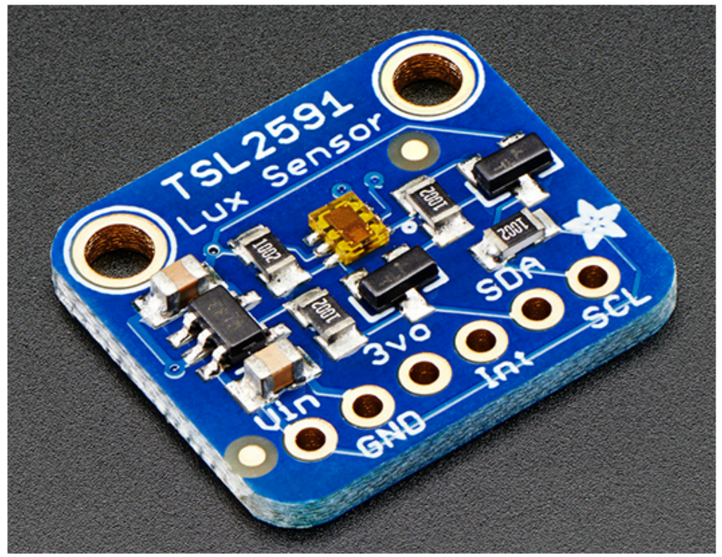
I2C

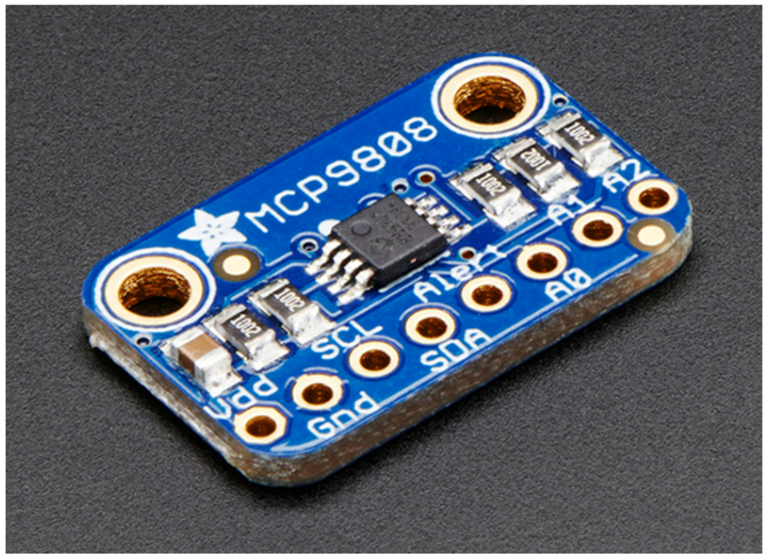
Vorteile
– Simpler Aufbau
– Nur zwei Pins benötigt
– Adressierungssystem (7-Bit, 8-Bit oder 10-Bit)
– Bis zu 128 Geräte addressierbar
– Klarer Kommunikationsweg
– Multiple Controller

Nachteile
– Langsamer als SPI
– Geschwindigkeit hängt von Wahl/Qualität der Kabel, Datenvolumen und externem Rauschen ab
– Komplexität nimmt mit steigender Geräte Anzahl zu

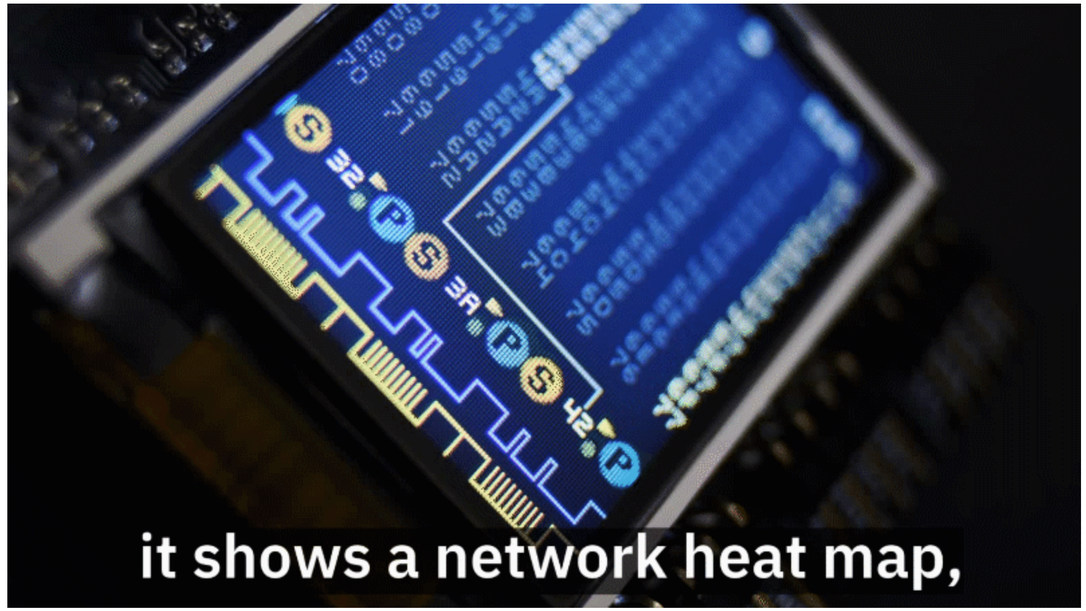
I2C

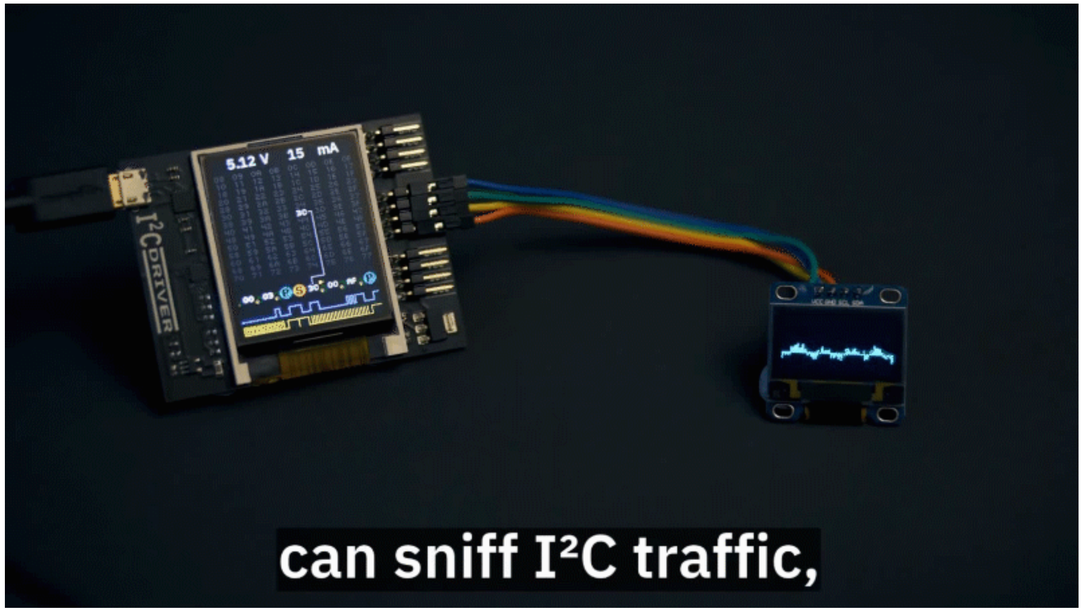

Altimeter
MPL3115A2
SDA
INT1
SCL
3vo
INT2
Vin
GND


TSL2591
Lux Sensor
SDA
SCL
3vo
IRQ
Vin
GND


MCP9808
A2
A1
Alert
A0
SCL
SDA
Vdd
Gnd

I2C


it shows a network heat map,


5.12 V 15 mA
I²C DRIVER
can sniff I²C traffic,

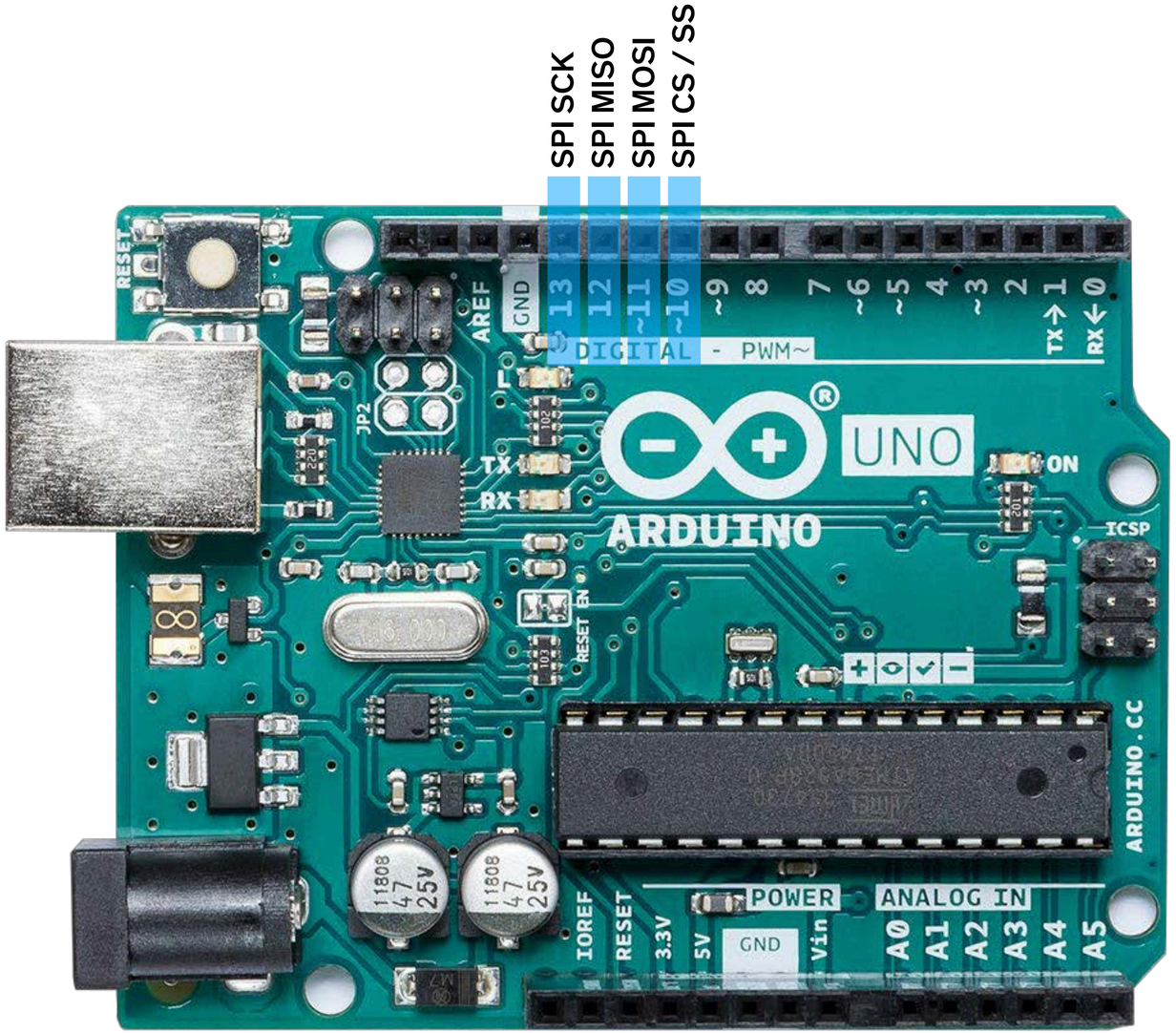
I2C

Serial Peripheral Interface

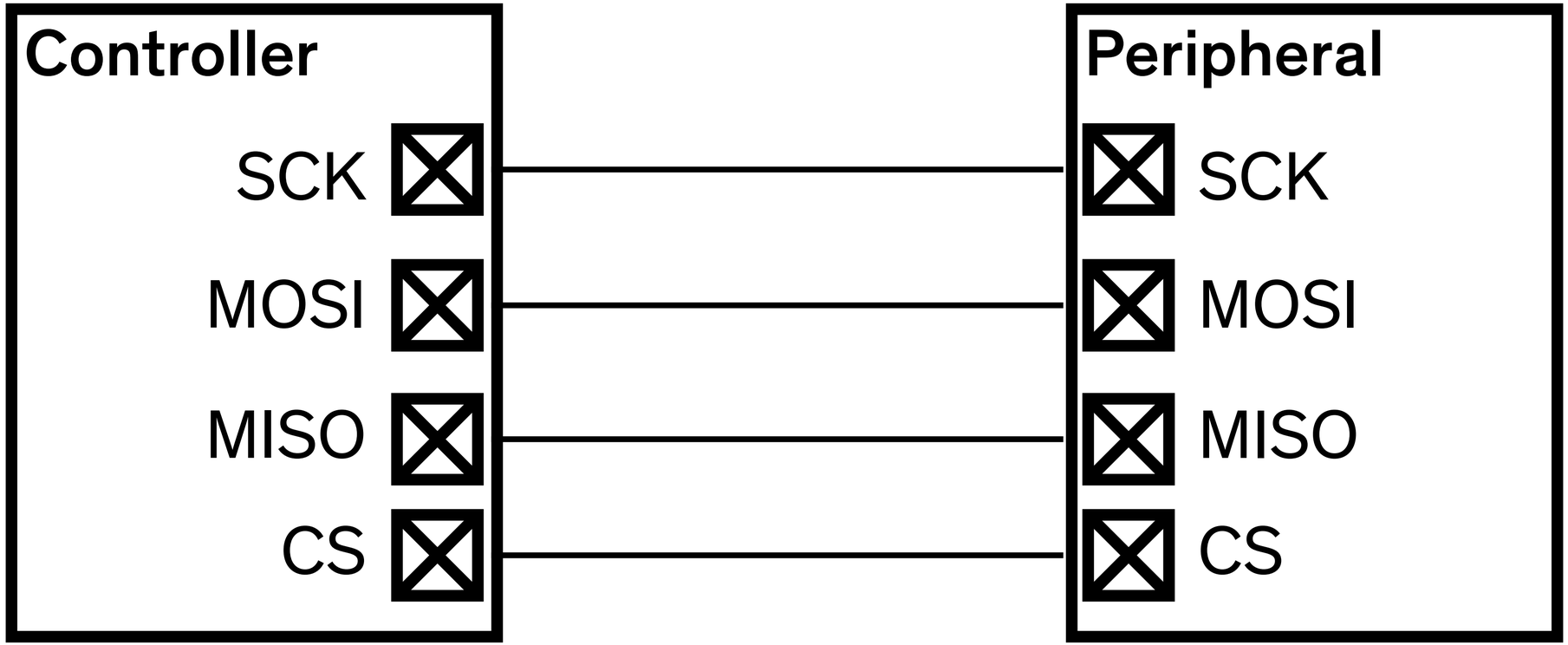
SCK = Serial Clock
MISO = Master (Data) Input, Slave (Data) Output
MOSI = Master (Data) Output, Slave (Data) Input
CS / SS = Chips Select / Slave Select

Ein Controller mit max. 4 Peripherien

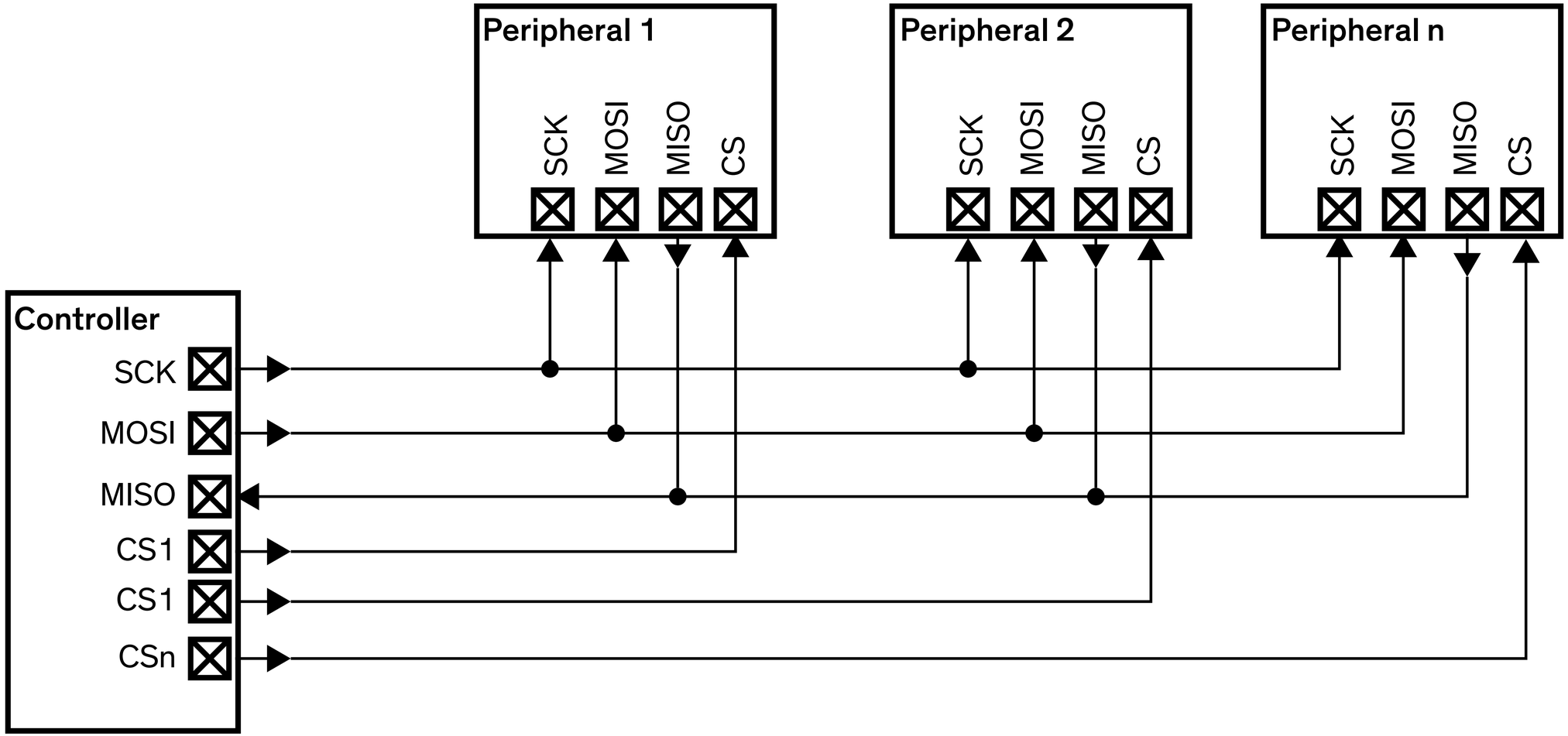
Deutlich schneller als I2C und UART



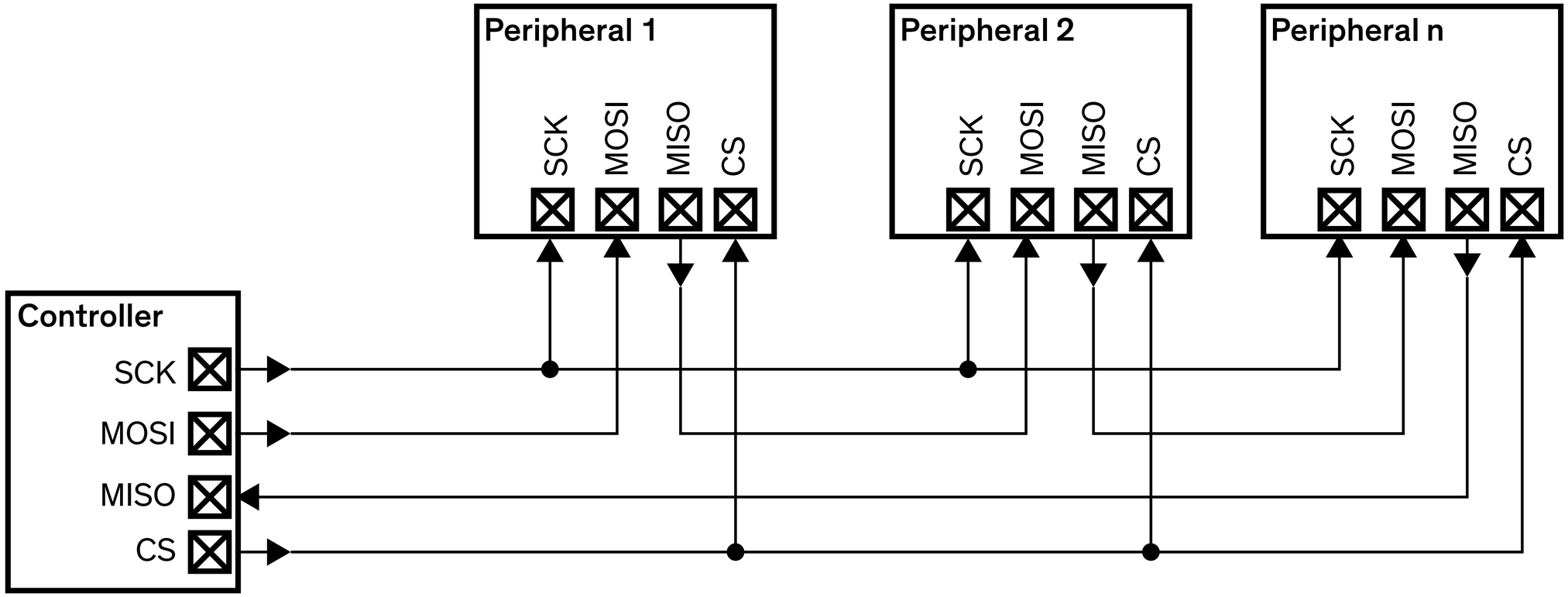
SPI



SPI https://www.oshwa.org/a-resolution-to-redefine-spi-signal-names/


Peripheral 1
SCK
MOSI
MISO
CS
Peripheral 2
SCK
MOSI
MISO
CS
Peripheral n
SCK
MOSI
MISO
CS
Controller
SCK
MOSI
MISO
CS1
CS1
CSn

SPI

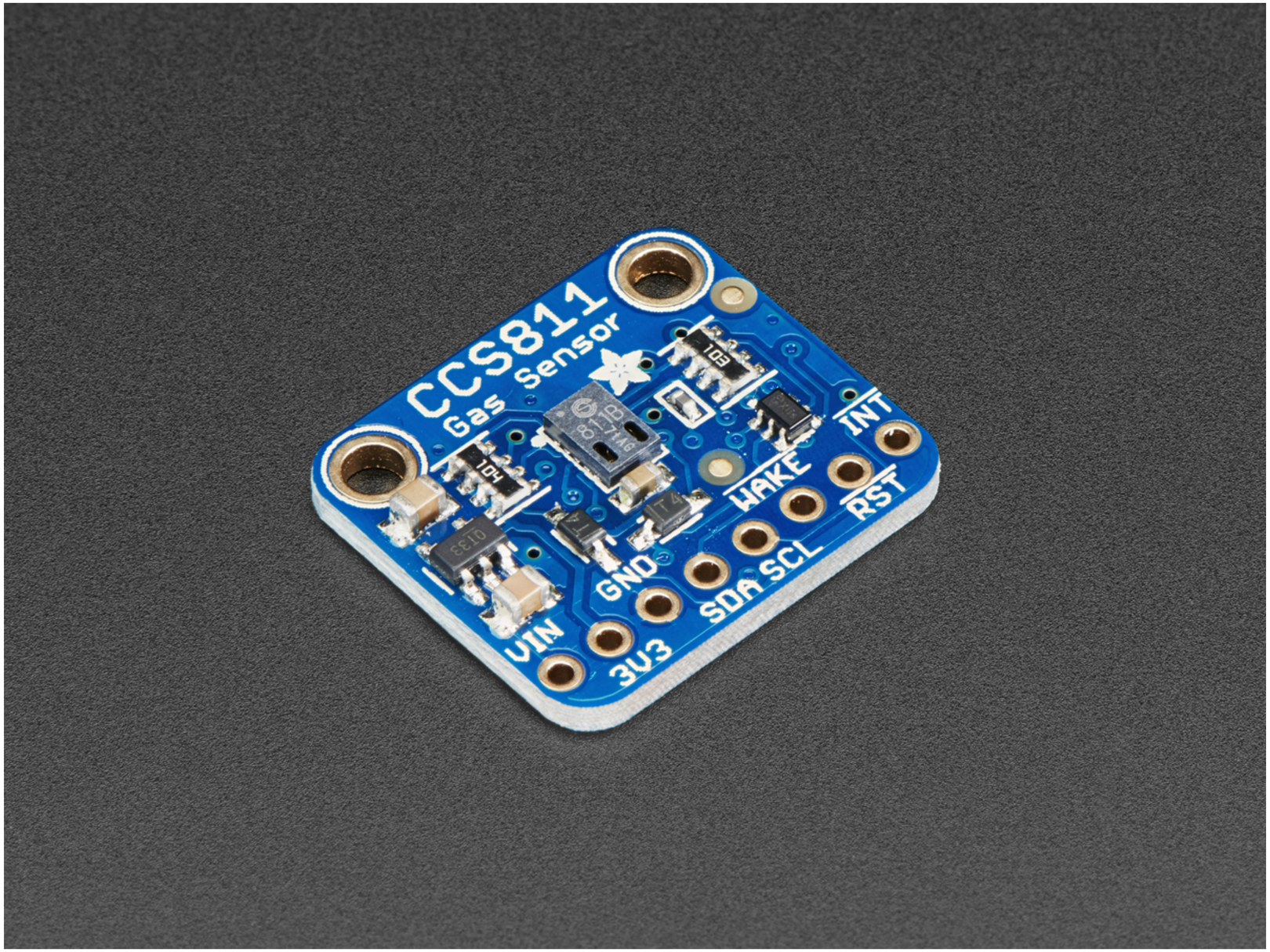

Peripheral 1
SCK
MOSI
MISO
CS
Peripheral 2
SCK
MOSI
MISO
CS
Peripheral n
SCK
MOSI
MISO
CS
Controller
SCK
MOSI
MISO
CS

SPI

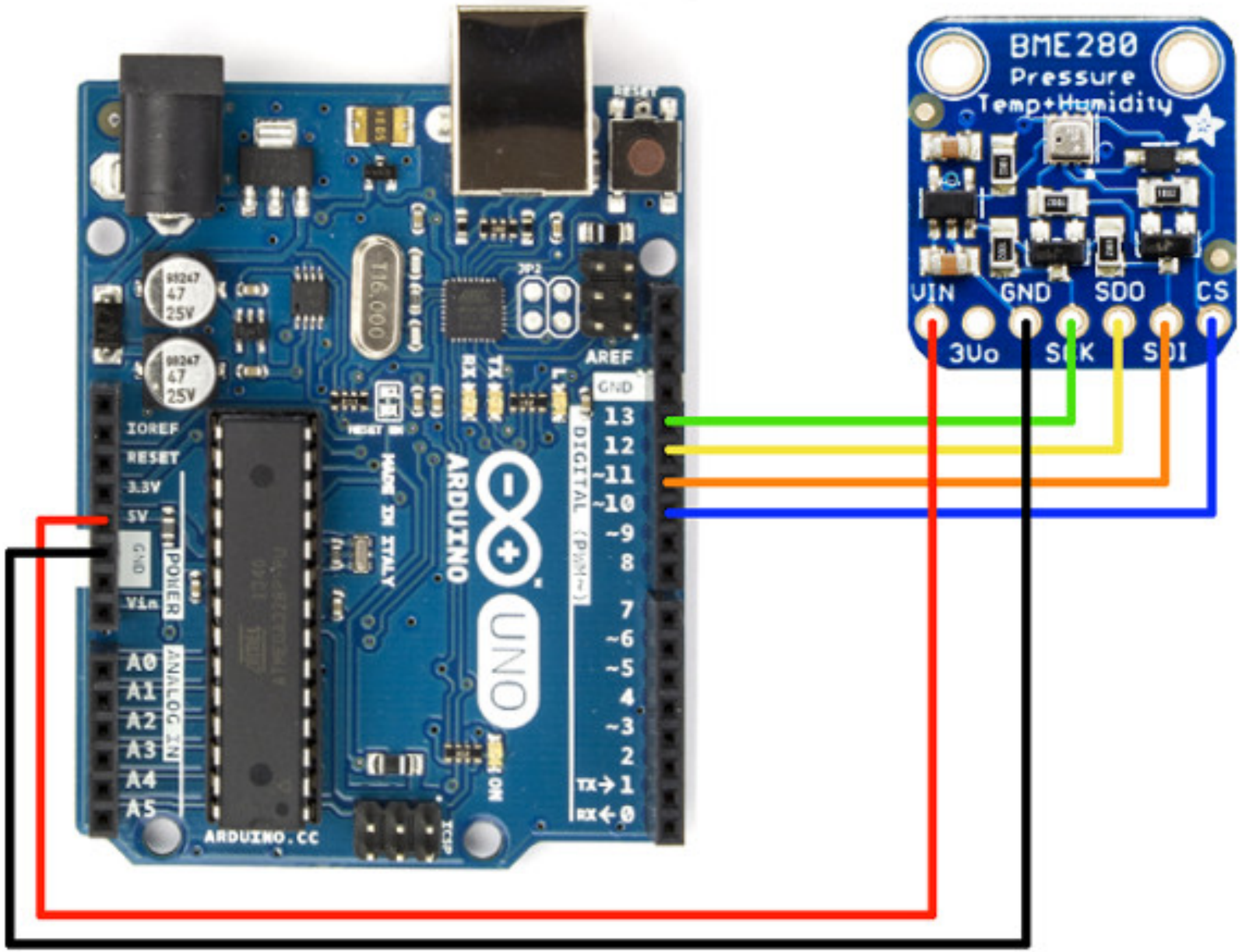
Vorteile
– Simples Protokoll
– Kein Adressierungssystem wie bei I2C
– Kein Limit an Geräten im Bus
– Wenig Pins in Benutzung
– Schnell: 8Mbits und mehr (Schneller als UART oder I2C)
– MISO und MOSI Leitungen sind separat (Gleichzeitig Daten empfangen und senden)

Nachteile
– Limitierung durch CS Pins (Außer bei Daisy-Chaining)
– min. 4 Pins benötigt / mehr Pins bei mehren CS Pins
– Single Controller

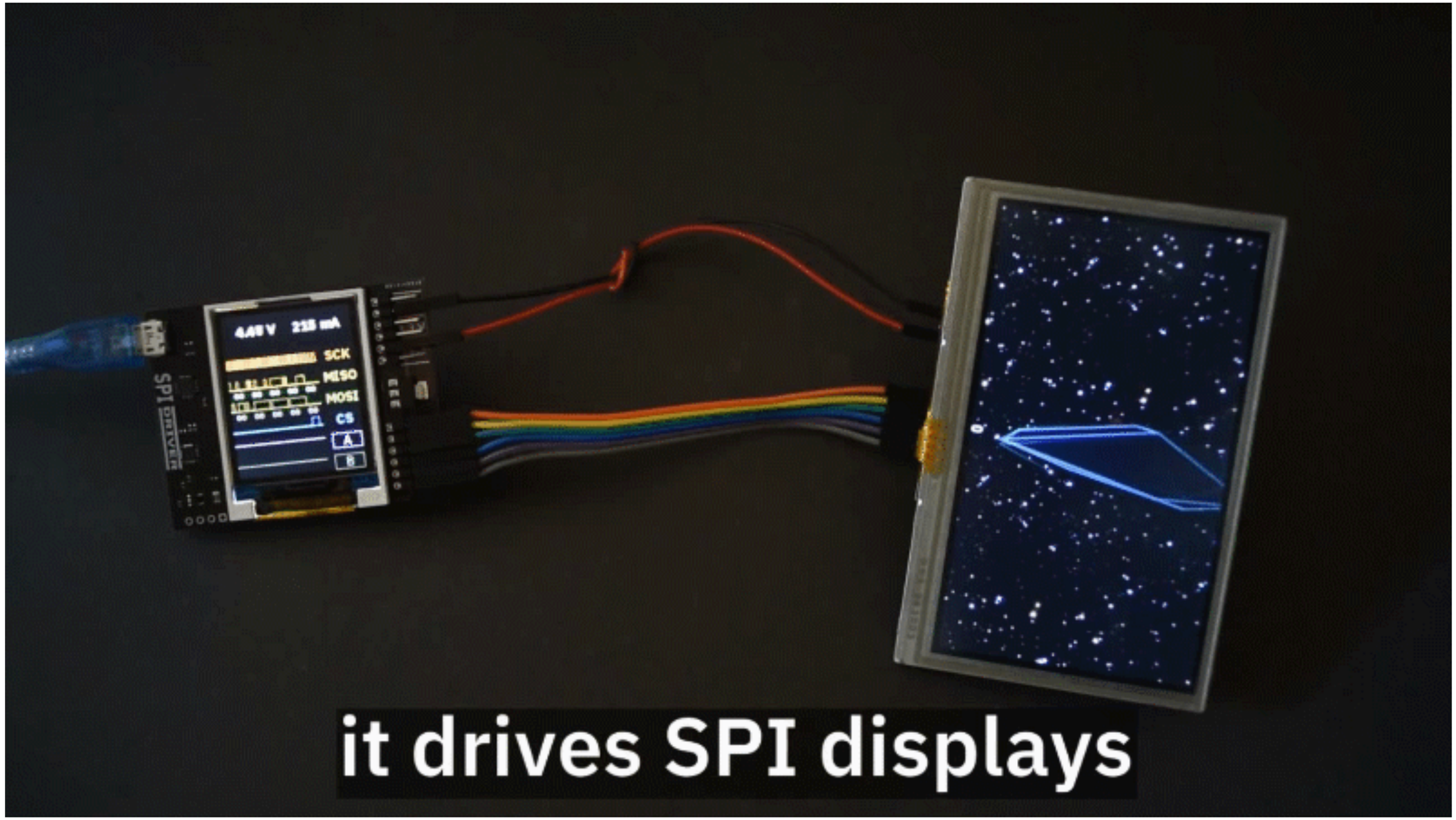
SPI



SPI


BME280
Pressure
Temp+Humidity
VIN
GND
SDO
CS
3Vo
SCK
SDI
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
IOREF
RESET
3.3V
5V
GND
Vin
POWER
A0
A1
A2
A3
A4
A5
ANALOG IN
ARDUINO
UNO
ARDUINO.CC

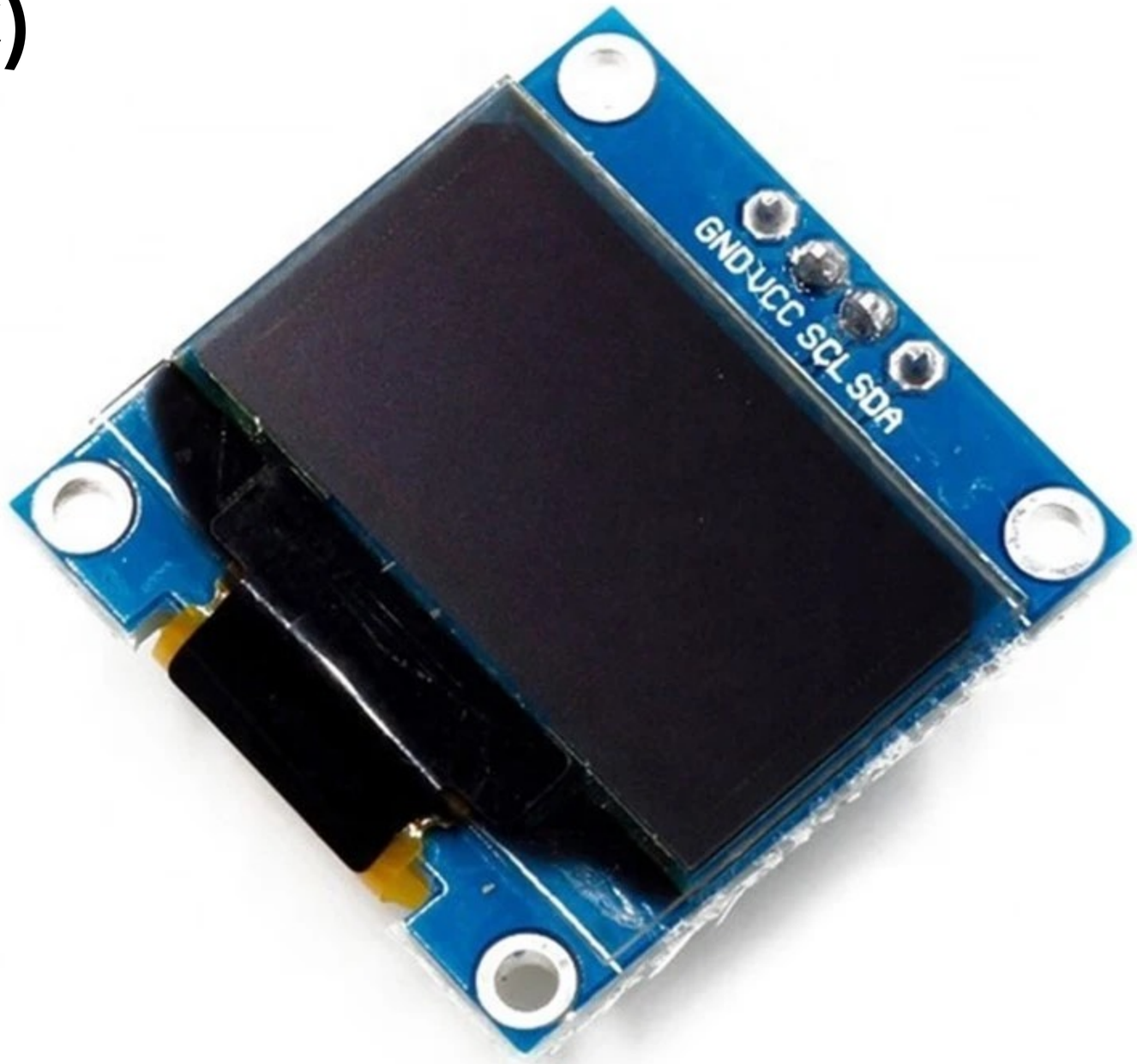
SPI



SPI

UART vs I2C vs SPI

2 Pins: SCL, SDA (I2C)



OLED Display (Demo)

Arduino Öko-System (Uno, Nano, Mega etc.)
→ UART / I2C / SPI
→ WLAN & Bluetooth über Shields
→ TCP / UDP über Shields

ESP8266 / ESP32 (Arduino IDE kompatibel)
→ UART / I2C / SPI
→ WLAN & Bluetooth (LE) integriert
→ TCP / UDP

Raspberry Pi (eigenes Betriebssystem)
→ UART / I2C / SPI
→ WLAN & Bluetooth integriert
→ TCP / UDP

Arduino Kommunikation

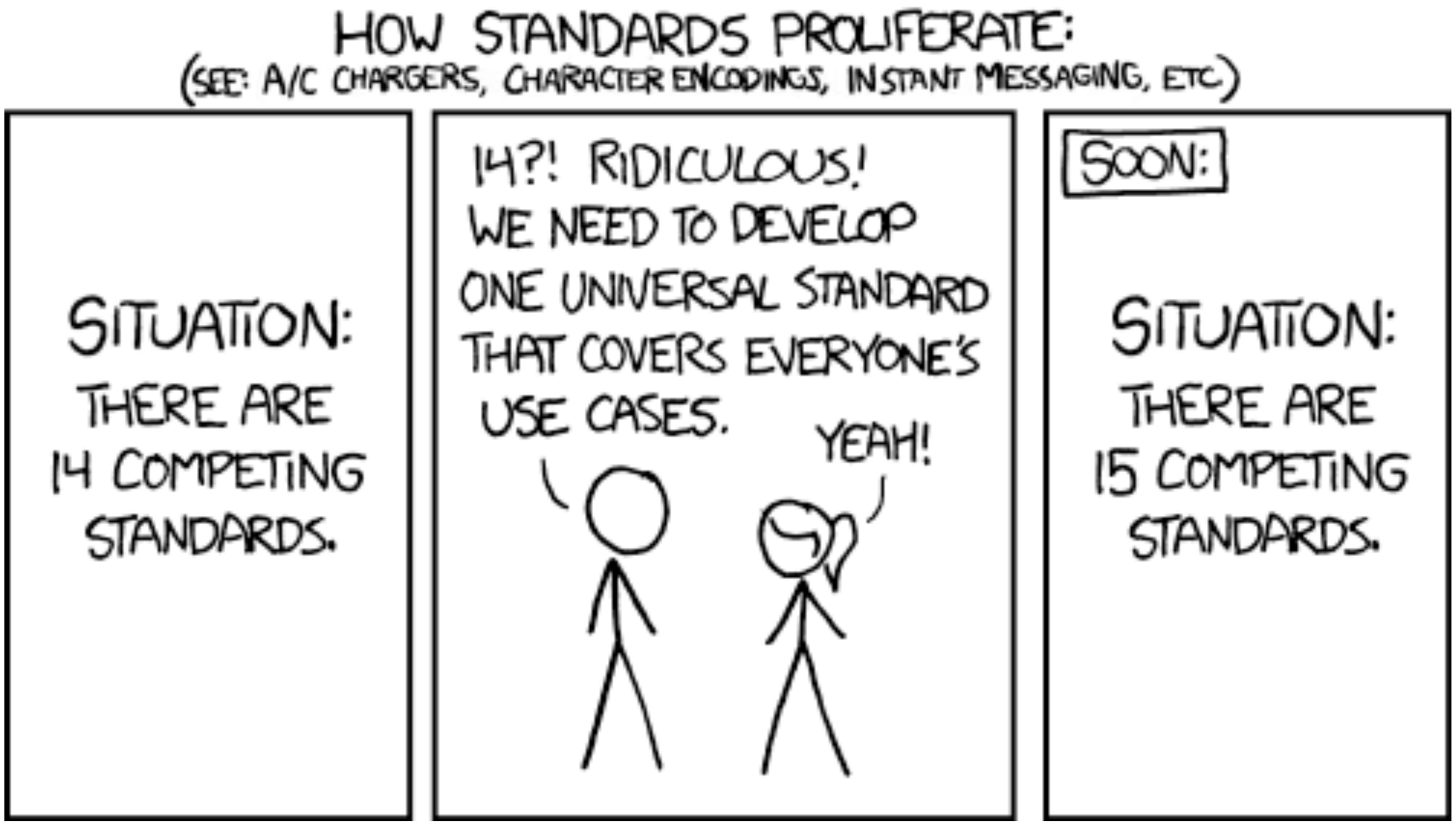
```
// PROCESSING
import processing.serial.*;
Serial myPort;
int value = 0;
void setup() {
 size(400, 400);
 // wir wählen das erste Gerät -> [0] aus
 // sollte unser Arduino nicht das erste Gerät, muss das
 // an dieser Steller angepasst werden
 myPort = new Serial(this, Serial.list()[0], 9600);
}

void loop() {
  //Wenn Daten vom Arduino übertragen werden,
  //lesen wir diese hier aus
  while(myPort.available() > 0) {
    value = myPort.read();
    println(value);
  }
}
```

```
// ARDUINO
void setup() {
  //Seriellen Kommunikationskanal mit 9600 Baud
  //pro Sekunde starten
  Serial.begin(9600);
}

void loop() {
  //Zeichen via Serial verschicken
  for(int i = 0; i<255; i++) {
    Serial.println(char(i));
    delay(500);
  }
}
```

Serielle Kommunikation mit Processing



Kommunikation über MQTT

Kommunikation über MQTT

Message Queuing Telemetry Transport

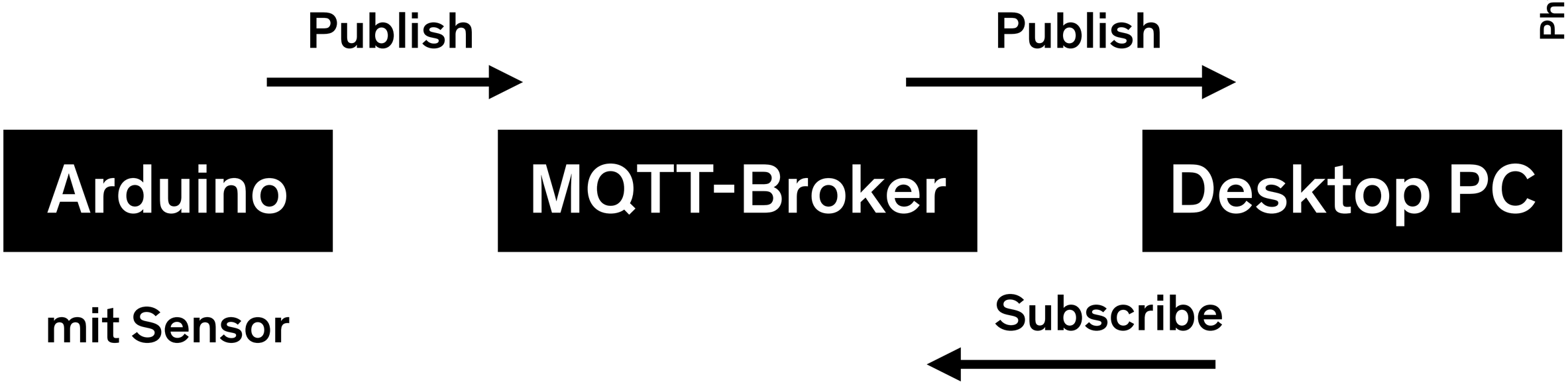
Machine-to-Machine-Kommunikation (M2M)

Simple System übertragen Sensor / Aktor Daten an komplexe(re) System bis zu voll entwickelten Rechnern weiter

Clients und Broker (Server)

Publish + Subscribe von Topics, z.B. mqtt://broker/topic/message

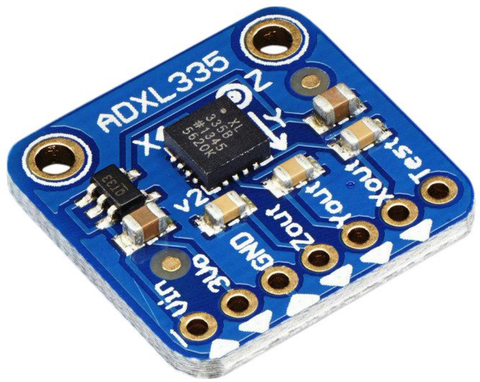
Quality of Service: at most once, at least once, exactly once



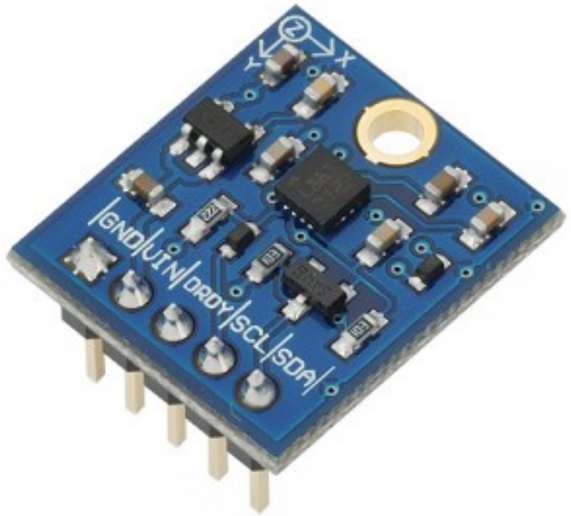
Kommunikation über MQTT

Sensoren höherer Komplexität

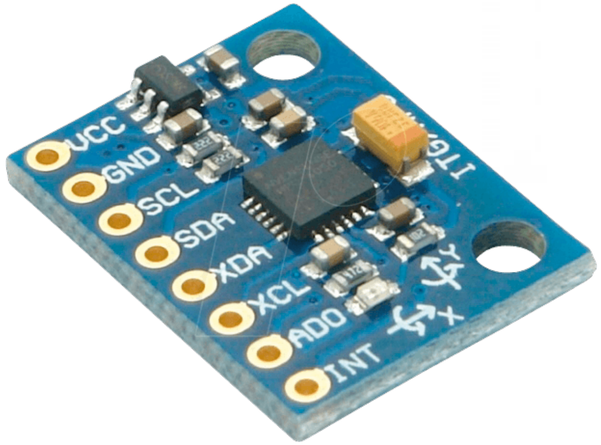
ADXL335
- Beschleunigungssensor
- Gibt X, Y und Z-Beschleunigung über drei separate analoge Ausgänge aus



HMC5883L
- 3-Achsen Lagesensor
- Sendet Positionsdaten über den I2C-Bus



GY-521
- 3-Achsen-Gyroskop
- Sendet Positionsdaten über den I2C-Bus

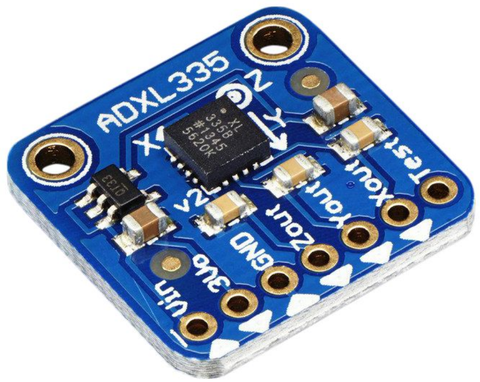


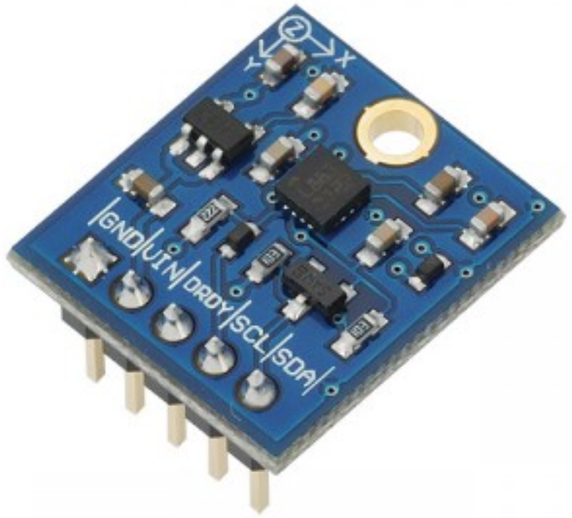
Lage- und Beschleunigungssensoren

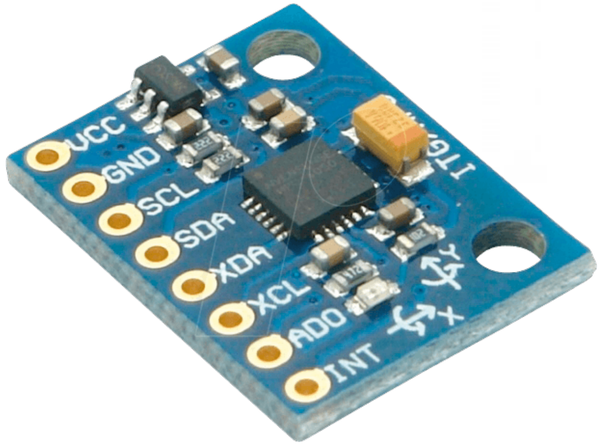
3 / 6 / 9 Degrees of Freedom

Inertiale Messeinheit

IMU: Inertial Mess Unit
MPU: Motion Processing Unit

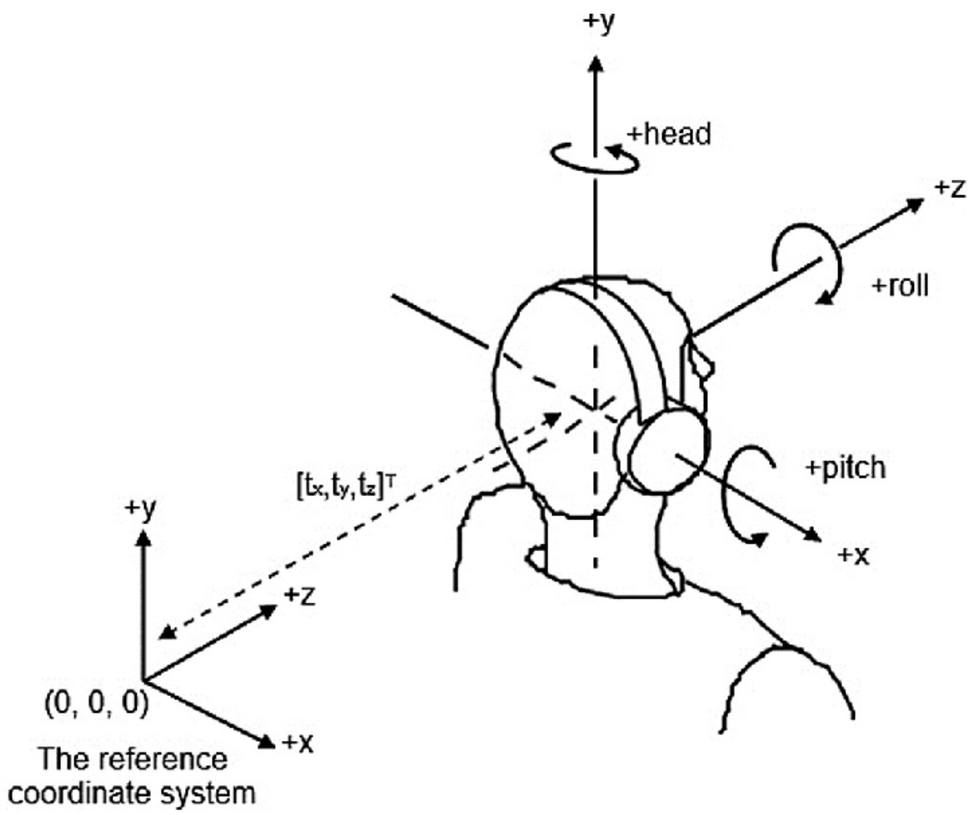


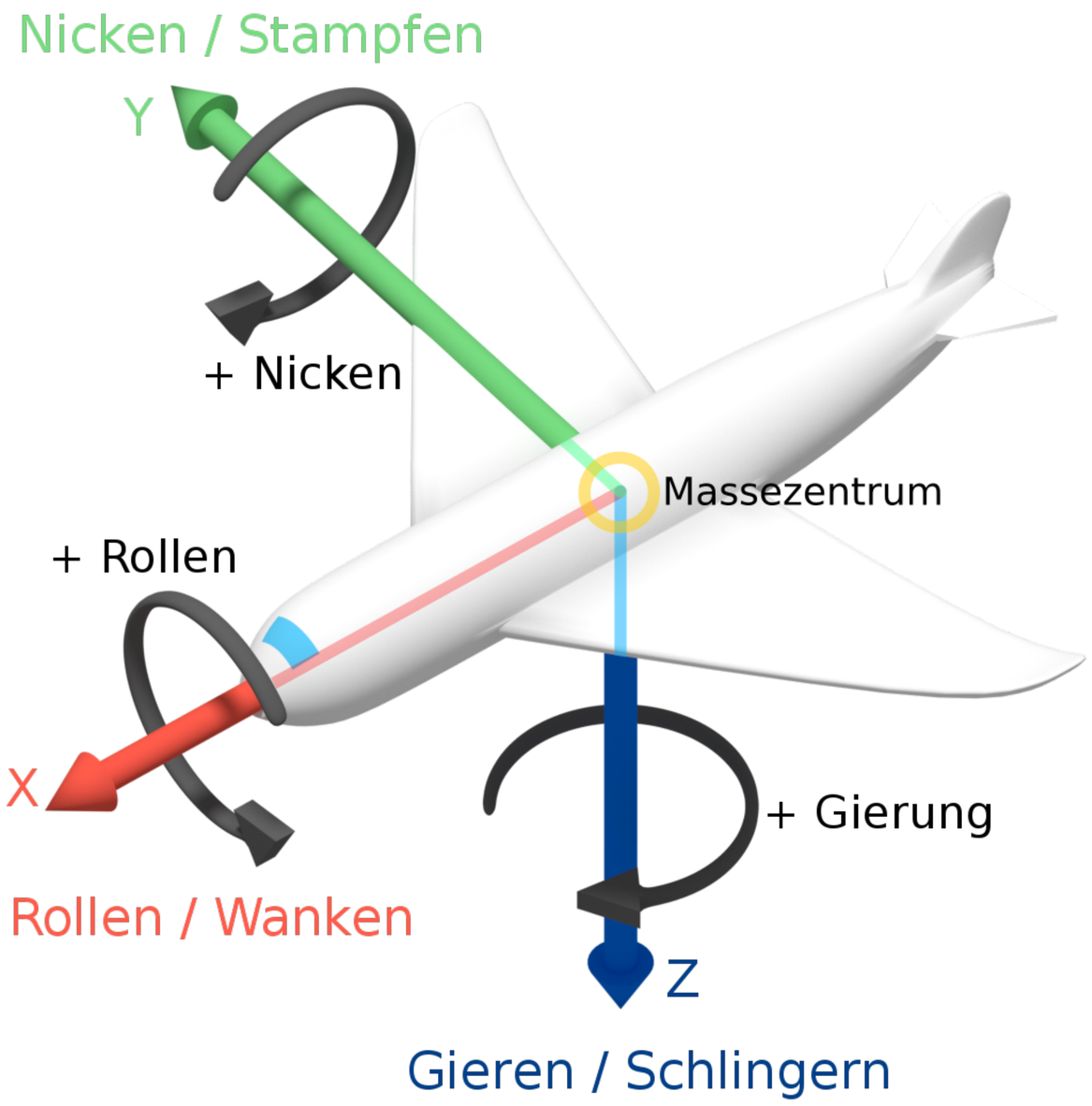




Lage- und Beschleunigungssensoren

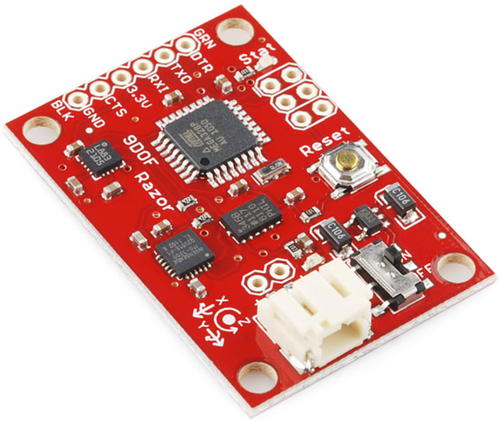
Degrees of Freedom

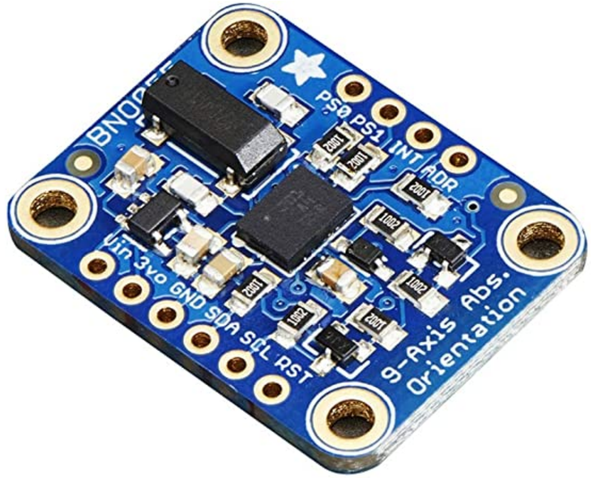


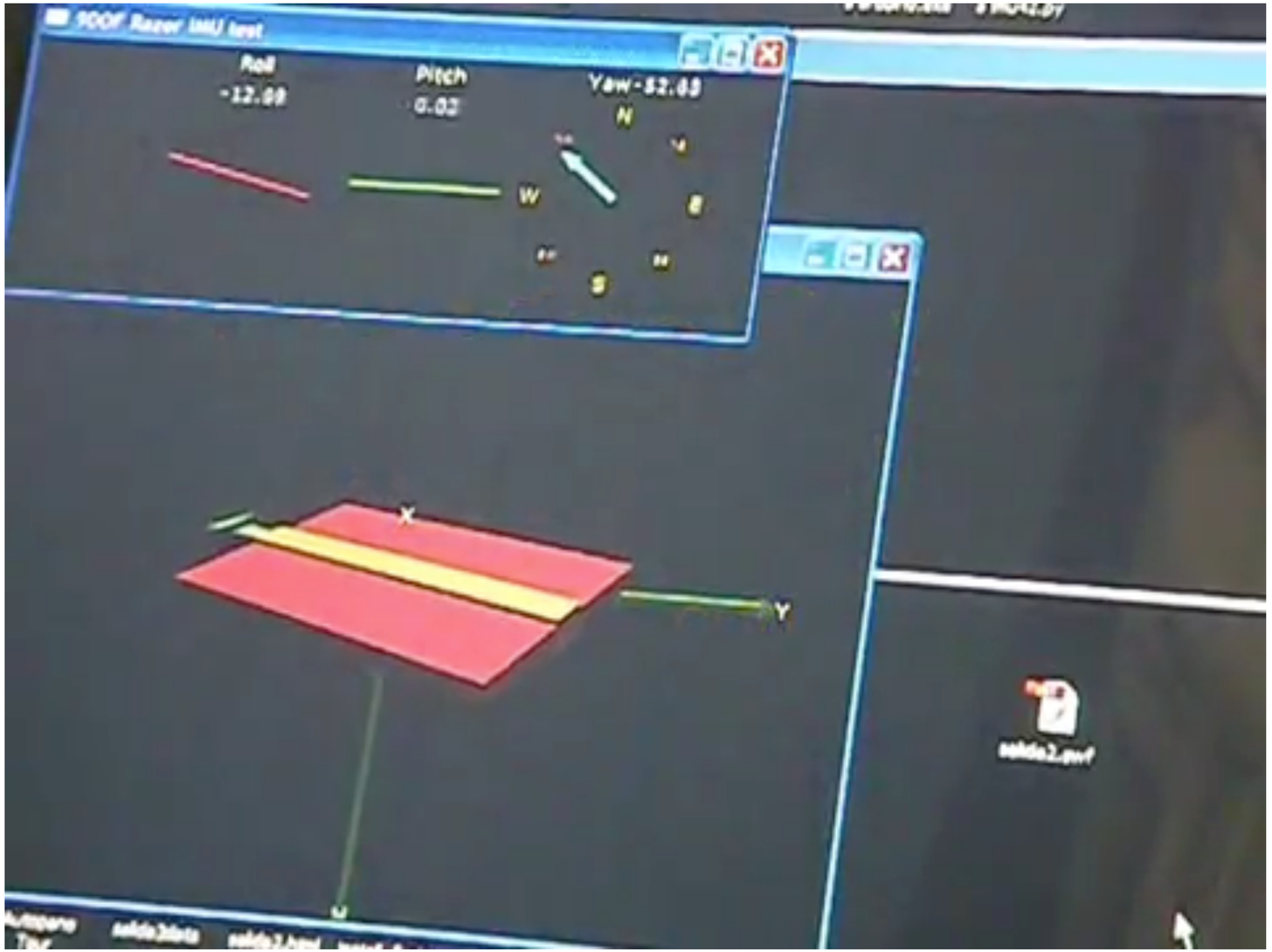


Lage- und Beschleunigungssensoren

Degrees of Freedom







https://www.sparkfun.com/products/retired/10736

Lage- und Beschleunigungssensoren

— Viele Online-Tools bieten eine sogenannte API an
— API → Application Programming Interface
— Bietet eine Schnittstelle zur entsprechenden Anwendung
– Man benötigt für gewöhnlich einen API Key
— Integration ist für Fortgeschrittene Benutzer
— Meistens HTTP-Zugriffe auf Server via Representational State Transfer (REST)
— Rückgriff auf Github-Repositories lohnt sich oft, um sich das Leben zu erleichtern, z.B. Twitter API: Sensordaten posten oder Tweets lesen
https://github.com/debsahu/TwitterWebAPI

oder Google Calendar auslesen:
https://www.arduino.cc/en/Tutorial/GoogleCalendarWiFi101

API / Datenbankaufrufe

Kommunikation über Arduino

- https://www.makerblog.at/2014/11/arduino-und-processing-datenuebertragung-leicht-gemacht/
- http://www.iotsharing.com/2017/05/tcp-udp-ip-with-esp32.html
- https://randomnerdtutorials.com/esp32-bluetooth-low-energy-ble-arduino-ide/
- https://www.arduino.cc/en/Reference/SPI
- https://learn.sparkfun.com/tutorials/serial-communication/all
- https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi

API / Datenbankaufrufe

- https://www.arduino.cc/en/Tutorial/GoogleCalendarWiFi101
- https://tutorials-raspberrypi.de/esp8266-google-maps-api-routenplaner-lcd-display/
- https://github.com/doctormord/ESP8266_EPD_Weather_Google_Calendar

Linksammlung

Special Topics 1
Einführung in EDA mit KiCad https://www.kicad.org/download/

Themenwünsche für Special Topics 2 + 3

Motor Workshop: Stepper, DC Motor, Servos
LED Matrix + LCD Display
Processing und Arduino (Datenvisualisierung)
Projekte Dekonstruieren (Unter die Haube schauen)

weitere Wünsche?

Special Topics

Arbeitet eigenständig an euren Ideen weiter.

Versucht auszuloten und/oder neue zu entwickeln.

Ich versuche im Laufe der Woche euch via E-Mail Feedback zu geben.

Aufgabe - Ausarbeitung

miro Board

Fragen?