Grundlagen der Programmierung von Microcontrollern

Michael Reichart, GFU Cyrus AG, Köln

2021 Michael Reichart, copyleft CC BY-NC-SA

Das Arduino Board

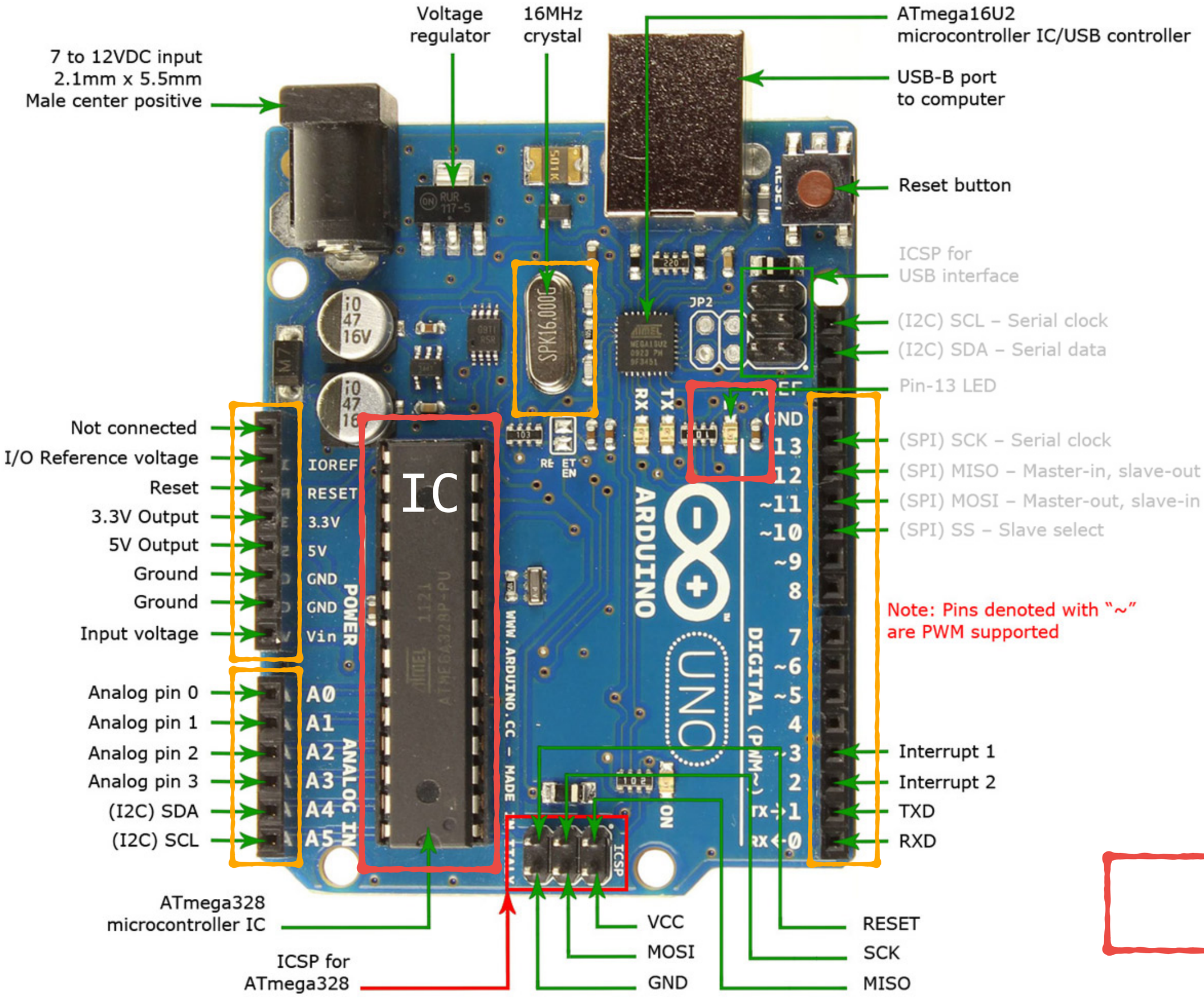
2021 Michael Reichart, copyleft CC BY-NC-SA

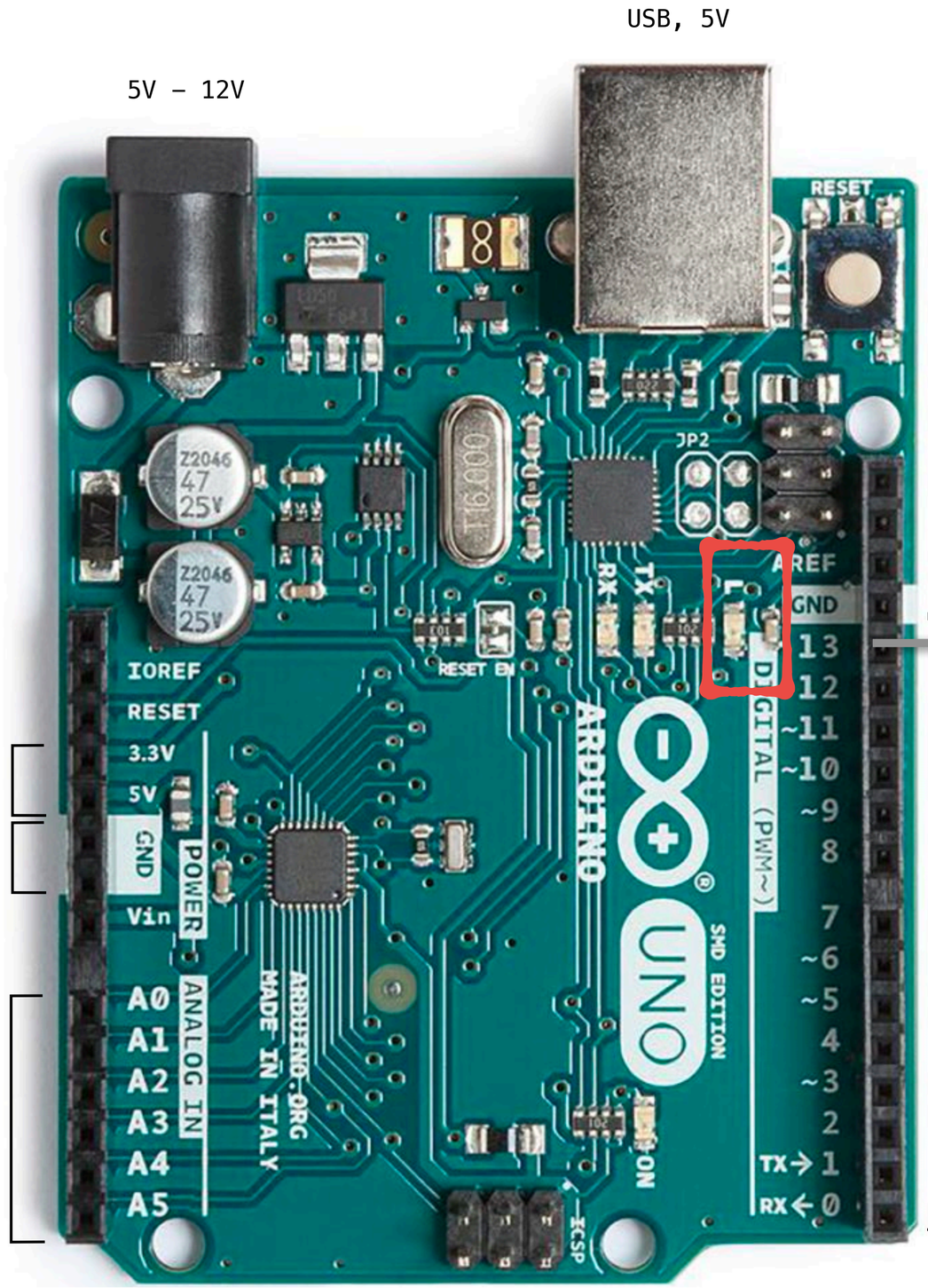
Geschichte und Herkunft

- Hernando Barragán entwickelt "Wiring" im Jahr 2003 als Masterarbeit am Interaction Design Institute Ivrea (IDII), betreut durch Casey Reas und Massimo Banzi.

- 2005 erstes "Arduino Uno" Board durch Massimo Banzi

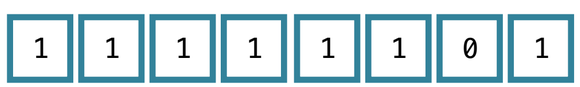
- Der Name „Arduino“ wurde von einer Bar in Ivrea übernommen, in der sich einige der Projektgründer gewöhnlich trafen. (Die Bar selbst wurde nach Arduin von Ivrea benannt, der von 1002 bis 1014 auch König von Italien war.)



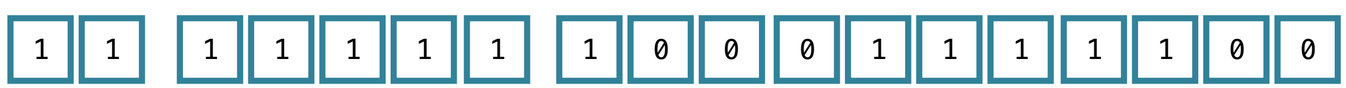


Binärsystem

1 byte
Register



253.0



3 252 , 625

3.1415298....

Hexadezimale Codes

0x AF 23 5C 6D 55 ...

Prozessor, Speicher, Stromversorgung

- Mikrokontroller: **ATmega328 P**
  SRAM: 2 KB, EEPROM: 1 KB (ATmega328)
  Prozessortaktung: **16 MHz**

- **Flash Memory: 32 KB** (ATmega328)
  davon 0.5 KB für den Bootloader

- Maße und Gewicht: 68.6 mm x 53.4 mm, 25 g

- **Versorgungsspannung: 5V**
  Eingabesspannung (empfohlen): 7-12V
  Eingabespannung (Grenzen): 6-20V

- Digitale I/O Pins: 14 (6 mit PWM Ausgabe)
  Analoge Input Pins: 6
  DC pro I/O Pin: 40 mA
  DC für den 3.3V Pin: 50 mA

Software

2021 Michael Reichart, copyleft CC BY-NC-SA

ARDUINO | Genuino

Search the Arduino Website

Home | Buy | Download | Products | Learning | Forum | Support | Blog | LOG IN | SIGN UP

DOWNLOAD

ENGLISH

Download the Arduino Software

ARDUINO 1.8.0

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board.
Refer to the Getting Started page for Installation instructions.

Windows Installer
Windows ZIP file for non admin install

Windows app Get

Mac OS X 10.7 Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM

Release Notes
Source Code
Checksums (sha512)

Blink

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct LED pin independent of which board is used.
  If you want to know what pin the on-board LED is connected to on your Arduino model, check
  the Technical Specs of your board  at https://www.arduino.cc/en/Main/Products

  This example code is in the public domain.

  modified 8 May 2014
  by Scott Fitzgerald

  modified 2 Sep 2016
  by Arturo Guadalupi

  modified 8 Sep 2016
  by Colby Newman
*/
```

Arduino/Genuino Uno auf /dev/cu.HC-05-DevB

fritzing

electronics
made easy

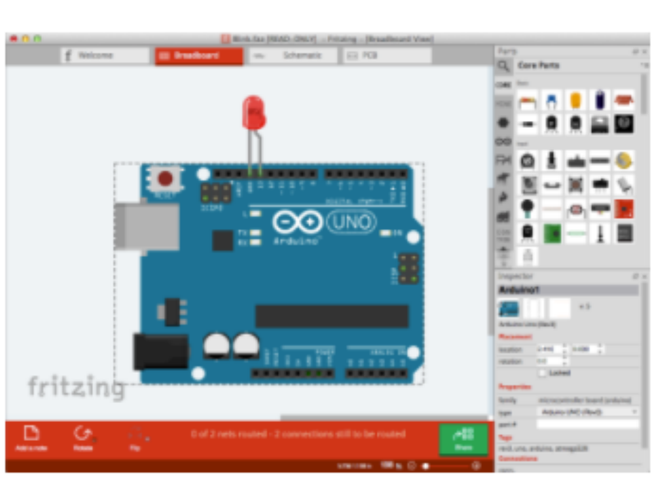
Projects Parts **Download** Learning Services Contribute FORUM FAB SIGN UP LOGIN

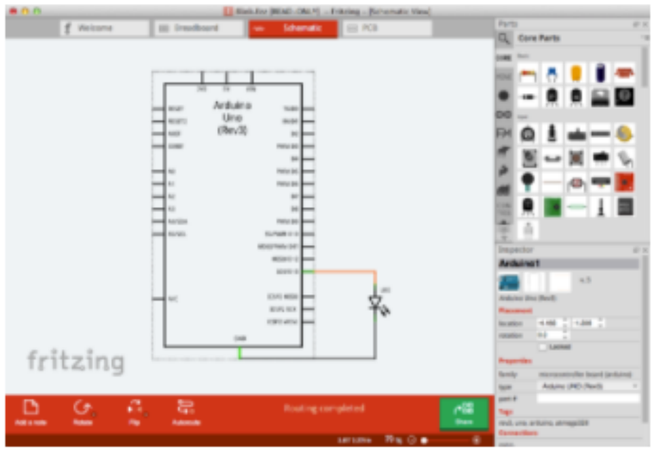
Fritzing is open source, free software. Please consider donating to Friends-of-Fritzing e.V. before downloading the app.
Fritzing is a non-profit organization devoted to making creative use of electronics accessible to everyone.

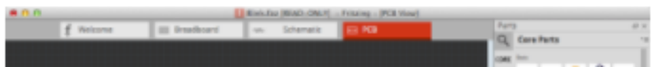
- ○ **No Donation**
- ○ **€ 10**
- ○ **€ 25**
- ○ **€** 50

Donate & Download

Version 0.9.3b was released on **Juni 2, 2016**.
Downloaded 908401 times.
See what's new and the known issues.







GO

FAQ ABOUT CONTACT

Blog

New fritzing release 0.9.3b!
Jun. 3, 2016

New Book: "Fritzing for Inventors"
Dec. 6, 2015

A new fritzing discussion forum
Nov. 8, 2015

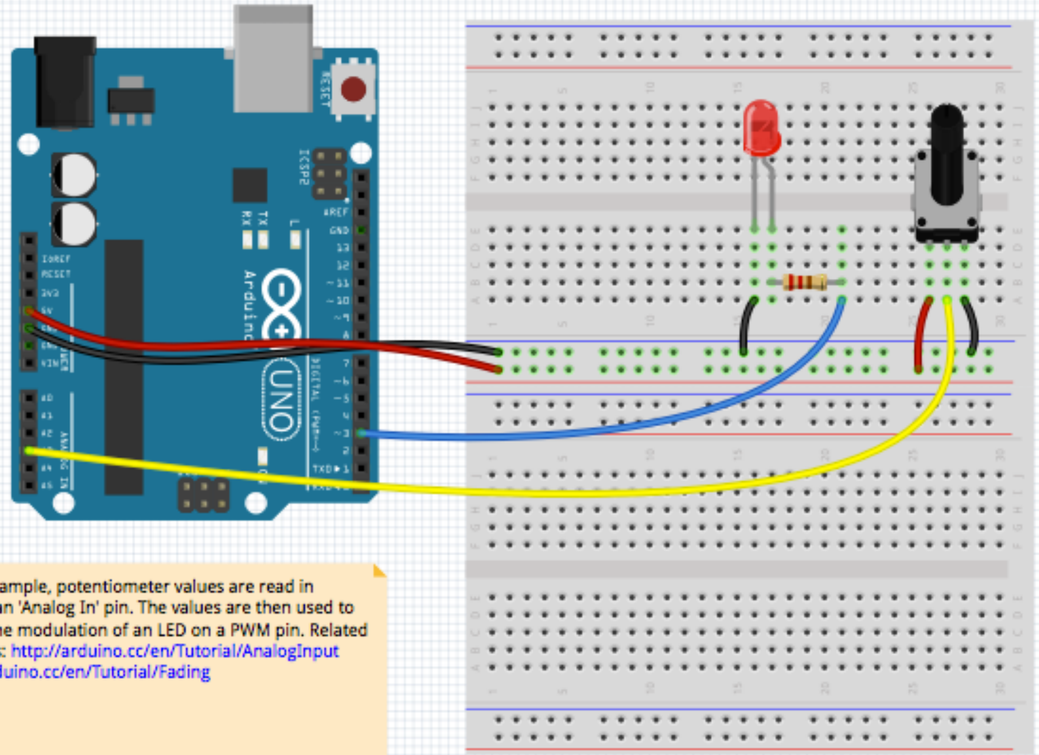
More posts...

Projects

3-axis CNC GRBL Setup
pless84

Laser Tag

Willkommen | Steckplatine | Schaltplan | Leiterplatte | Code



In this example, potentiometer values are read in through an 'Analog In' pin. The values are then used to control the modulation of an LED on a PWM pin. Related examples: http://arduino.cc/en/Tutorial/AnalogInput http://arduino.cc/en/Tutorial/Fading

fritzing

Notiz | Drehen | Umdrehen

Routing fertiggestellt

Veröffentlichen

Fertig

(x,y)=(1.442, -3.340) in 102 %

Bauteile

Core Parts

CORE | MINE | seeed | intel

Basis

Eingabe

Inspektor

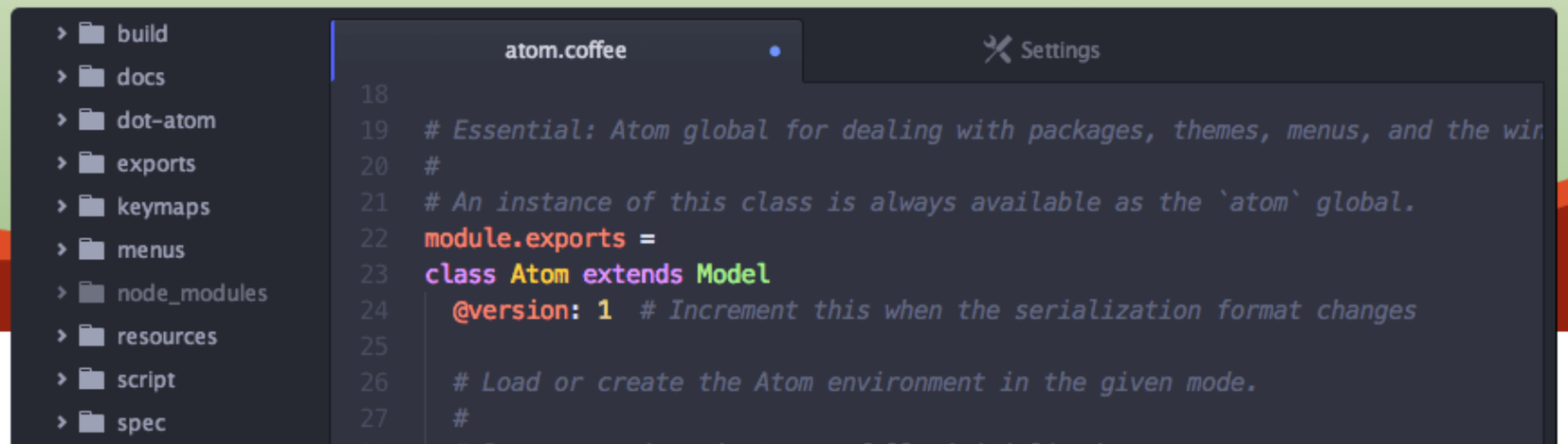
Packages Themes Documentation Blog Discuss Sign In

ATOM

A hackable text editor
for the 21st Century

Download For Mac

For macOS 10.8 or later. Other platforms - Beta releases



Entwicklungsumgebungen

- Visual Studio
- Atom
- Eclipse
- ...

Websites

2021 Michael Reichart, copyleft CC BY-NC-SA

Arduino und Fritzing

- https://www.arduino.cc/
- http://www.arduino.org/
- http://fritzing.org/
- https://arduinohistory.github.io/de

Maker Szene

- http://makezine.com/
- http://www.instructables.com/

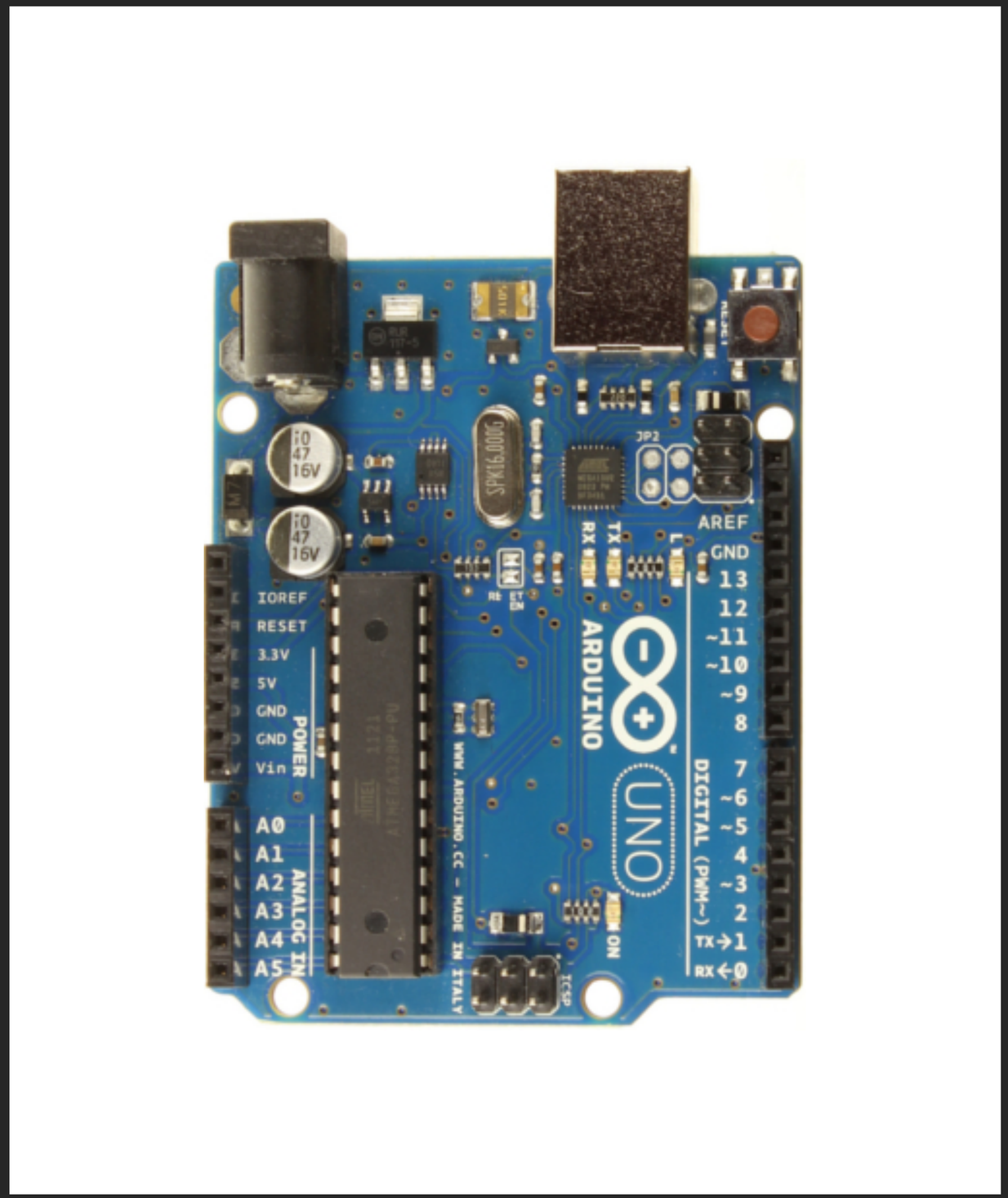
Bezugsquellen

- https://www.sparkfun.com/
- https://www.adafruit.com/
- https://www.amazon.de
- http://www.exp-tech.de/
- https://www.sunfounder.com/
- https://www.alibaba.com

Arduino Boards

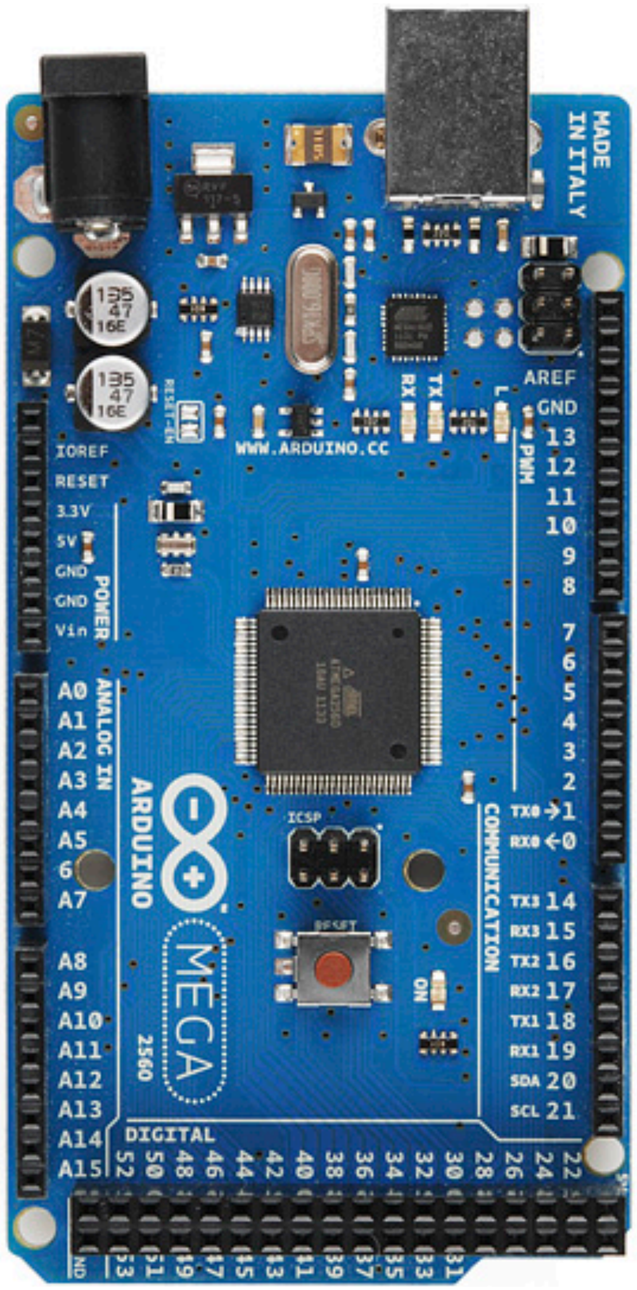
Modelle

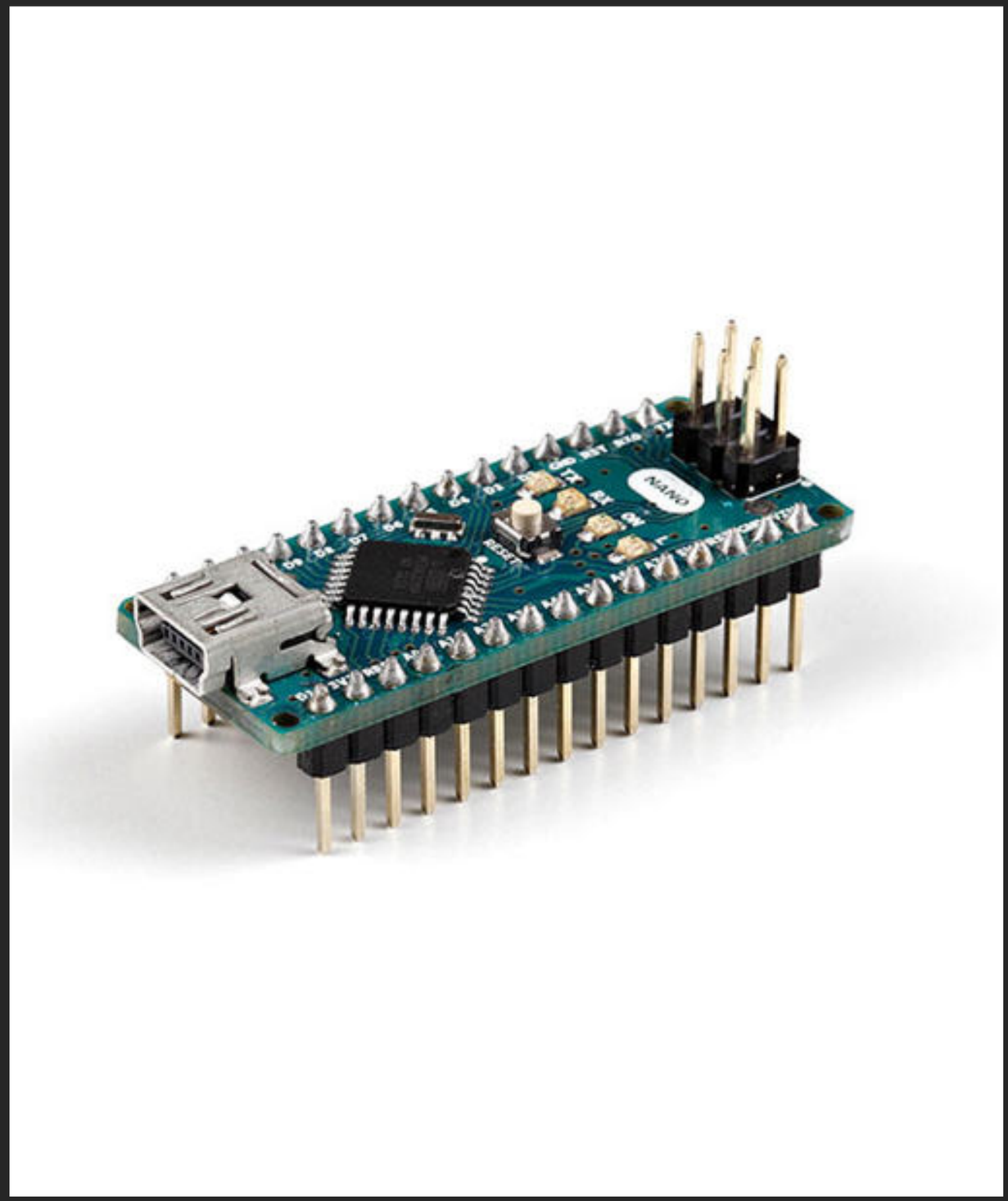
2021 Michael Reichart, copyleft CC BY-NC-SA



Arduino Uno

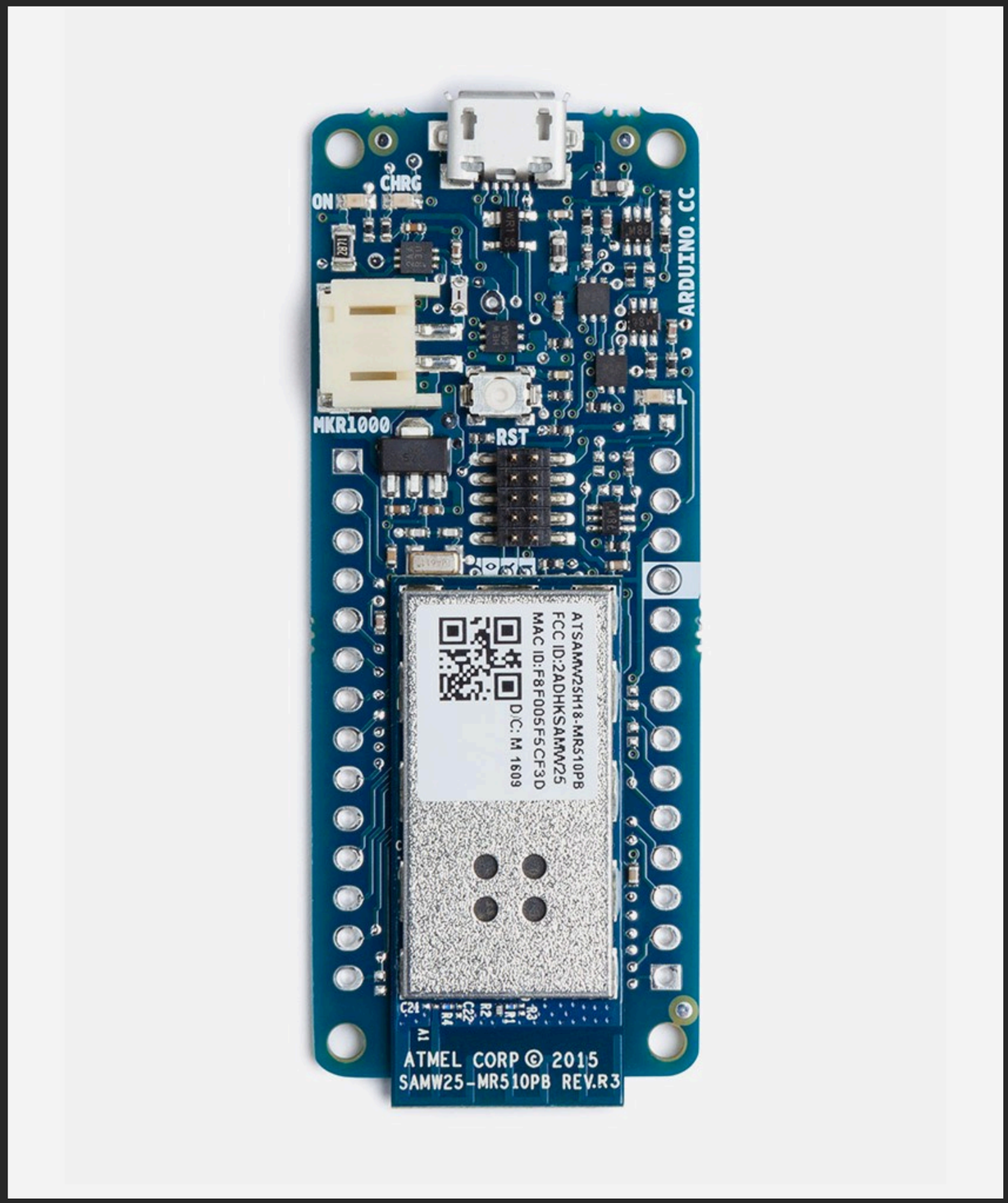
Arduino Mega



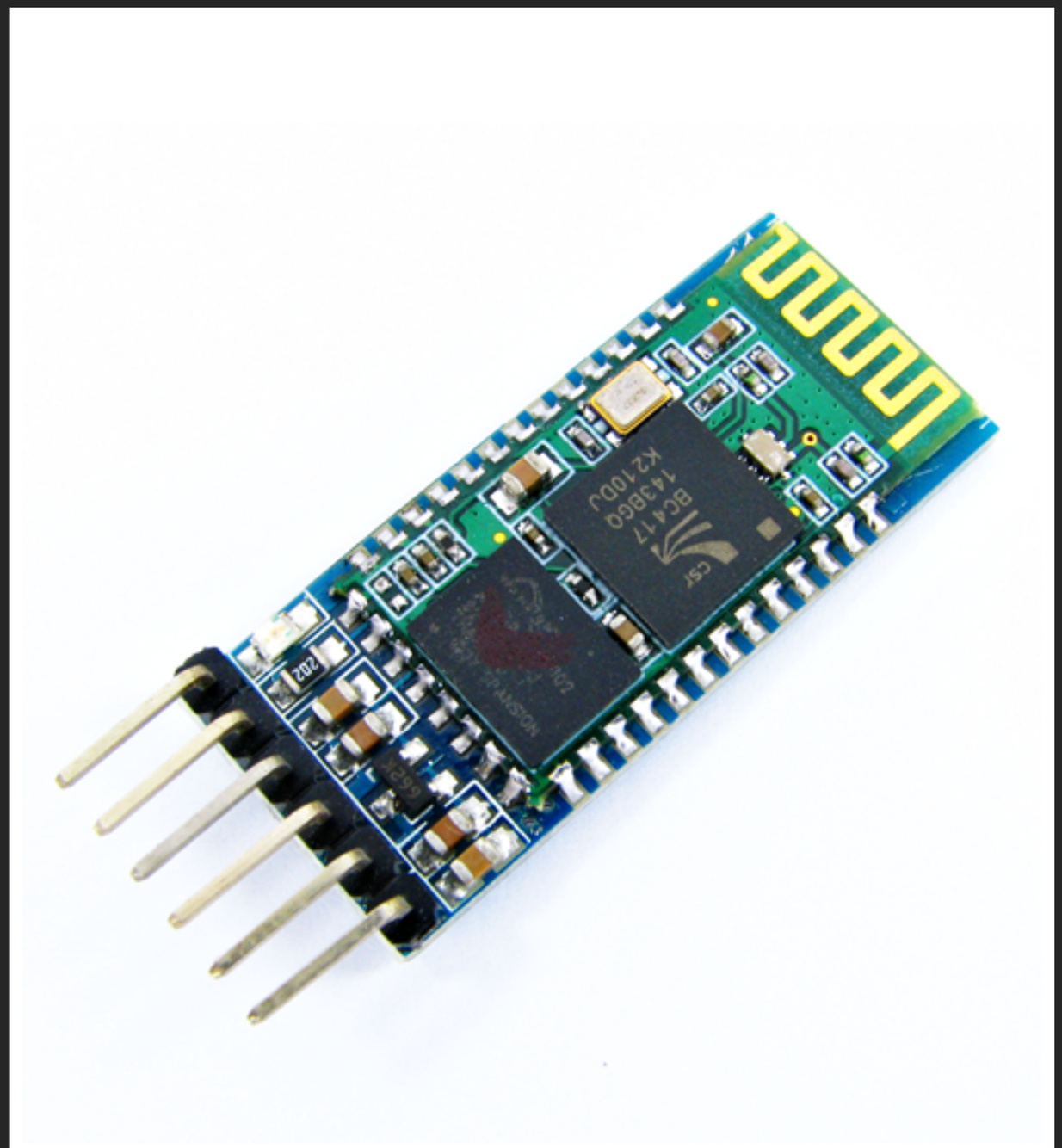


Arduino Nano

Arduino MKR1000



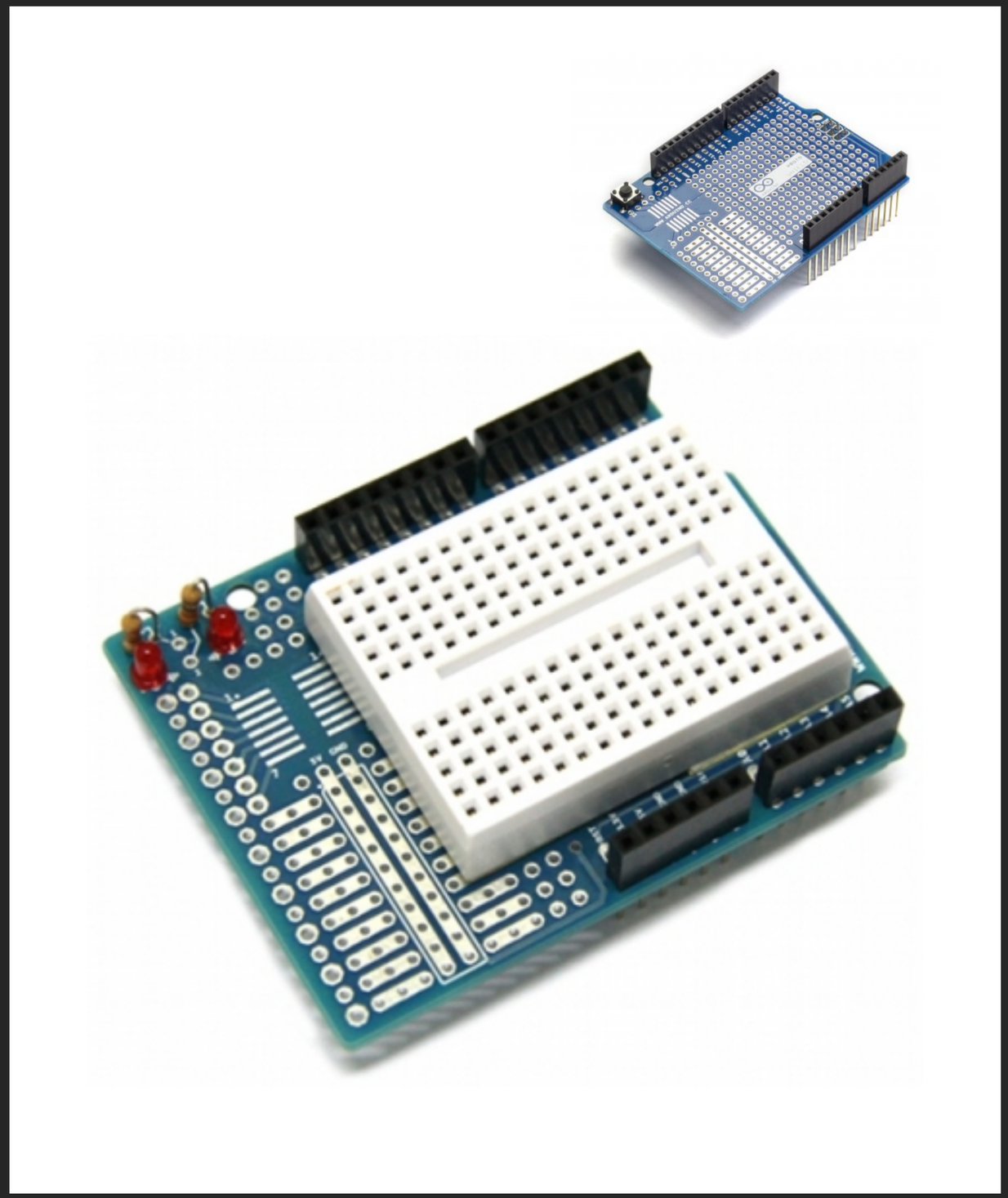
Bluetooth Modul HC05



Lilypad

Shields

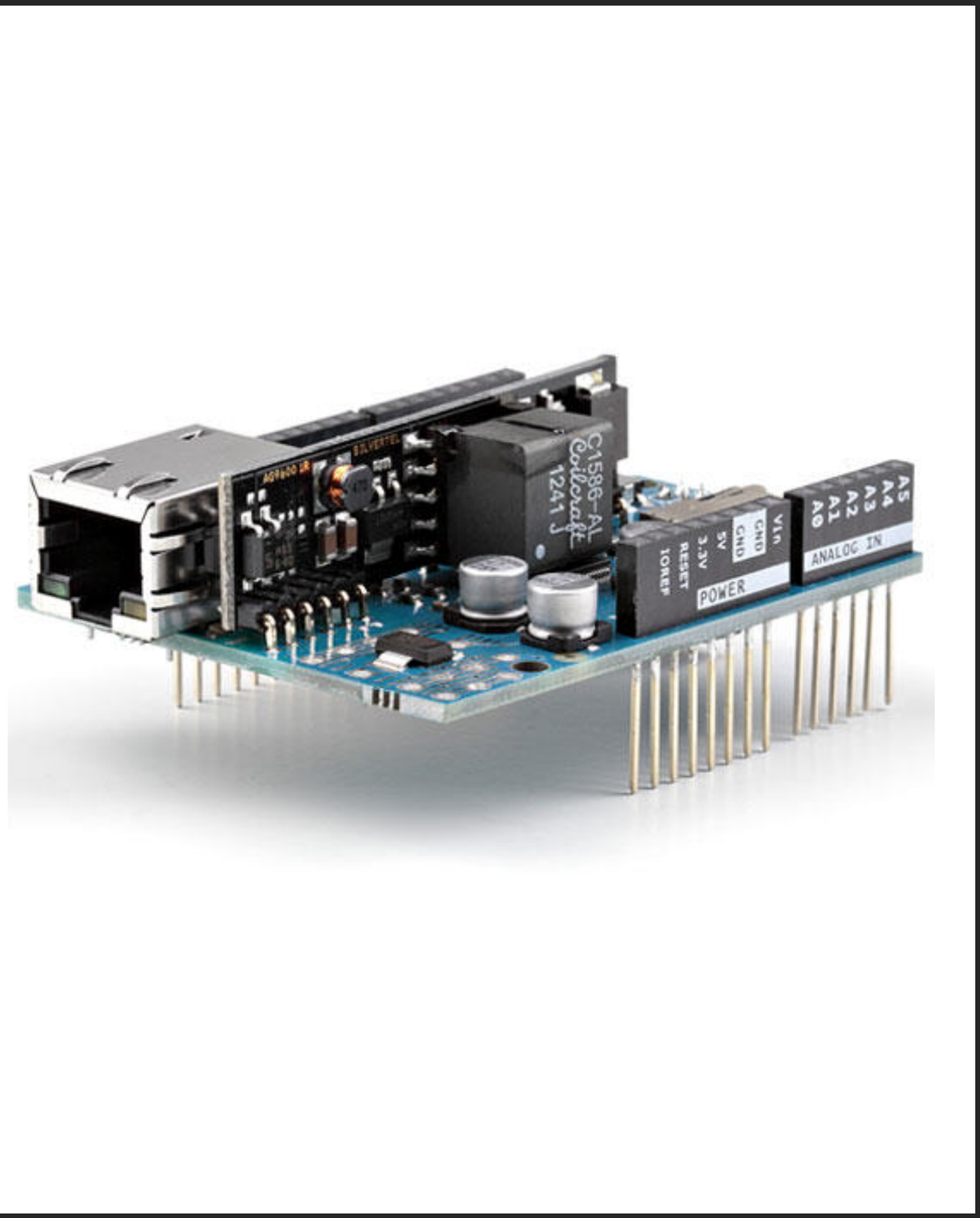
2021 Michael Reichart, copyleft CC BY-NC-SA



Protoshield

Motor Stepper Shie

Bluetooth Shield

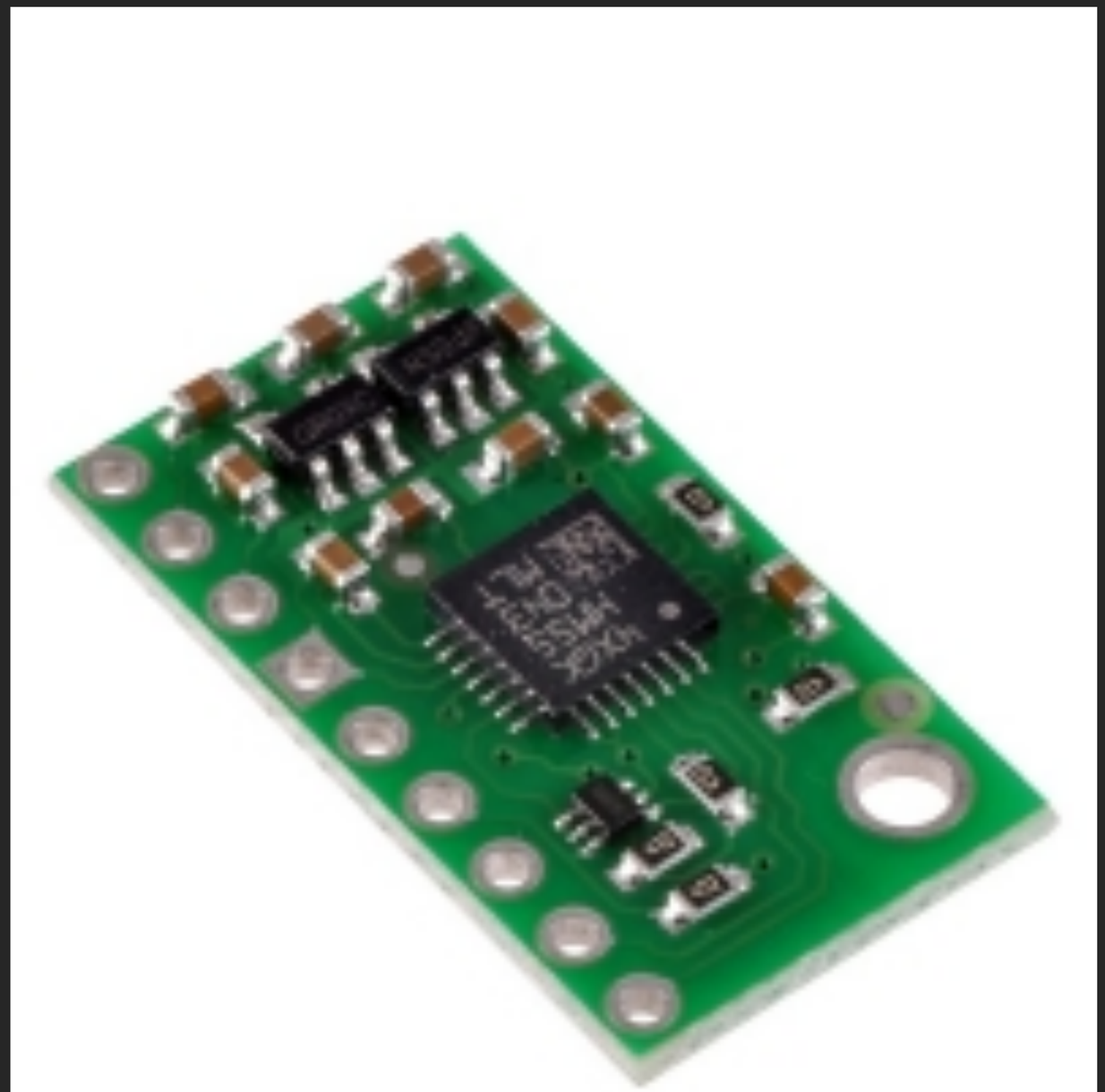


Ethernet Shield

Sensoren

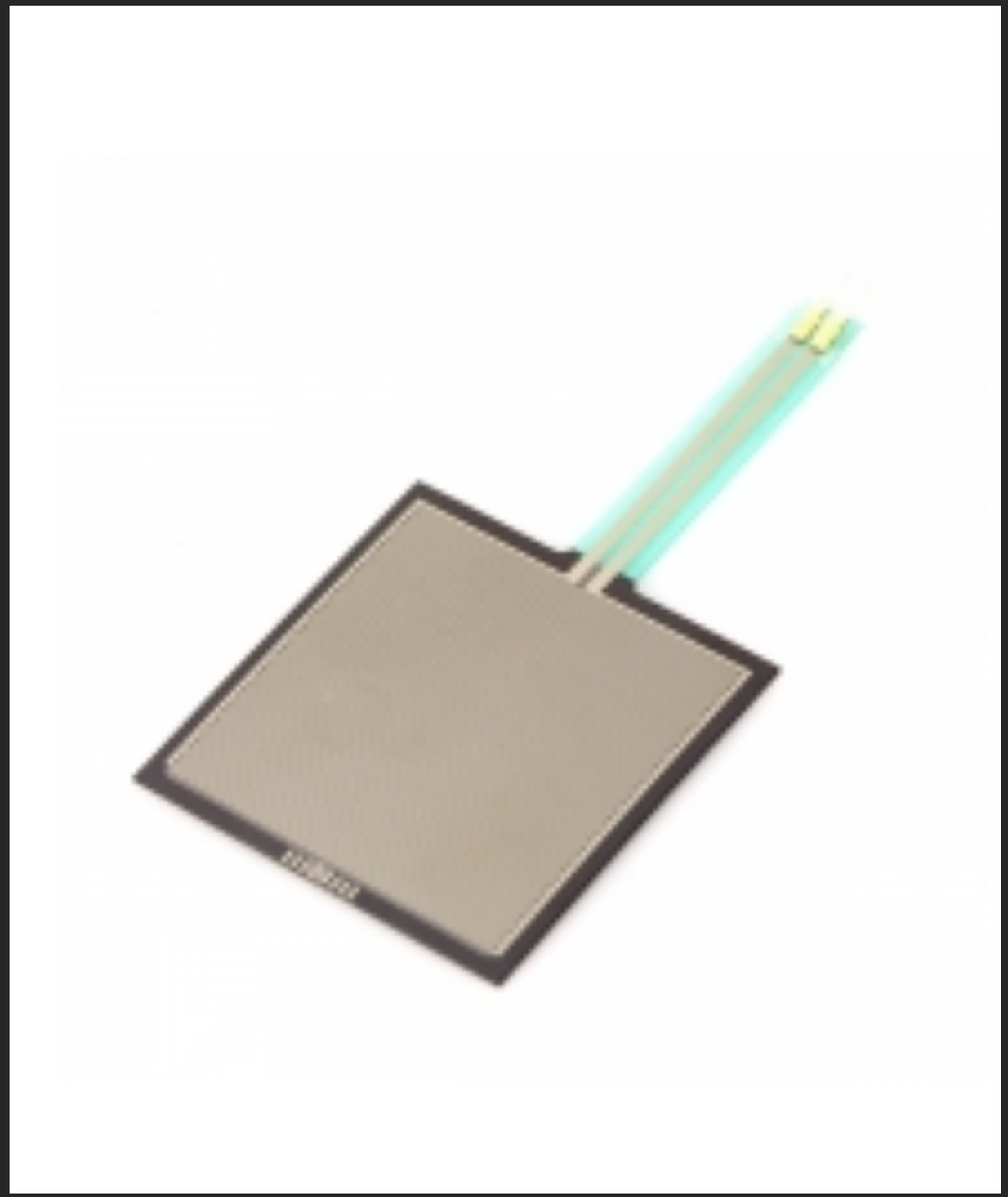
2021 Michael Reichart, copyleft CC BY-NC-SA

Kondensator micro



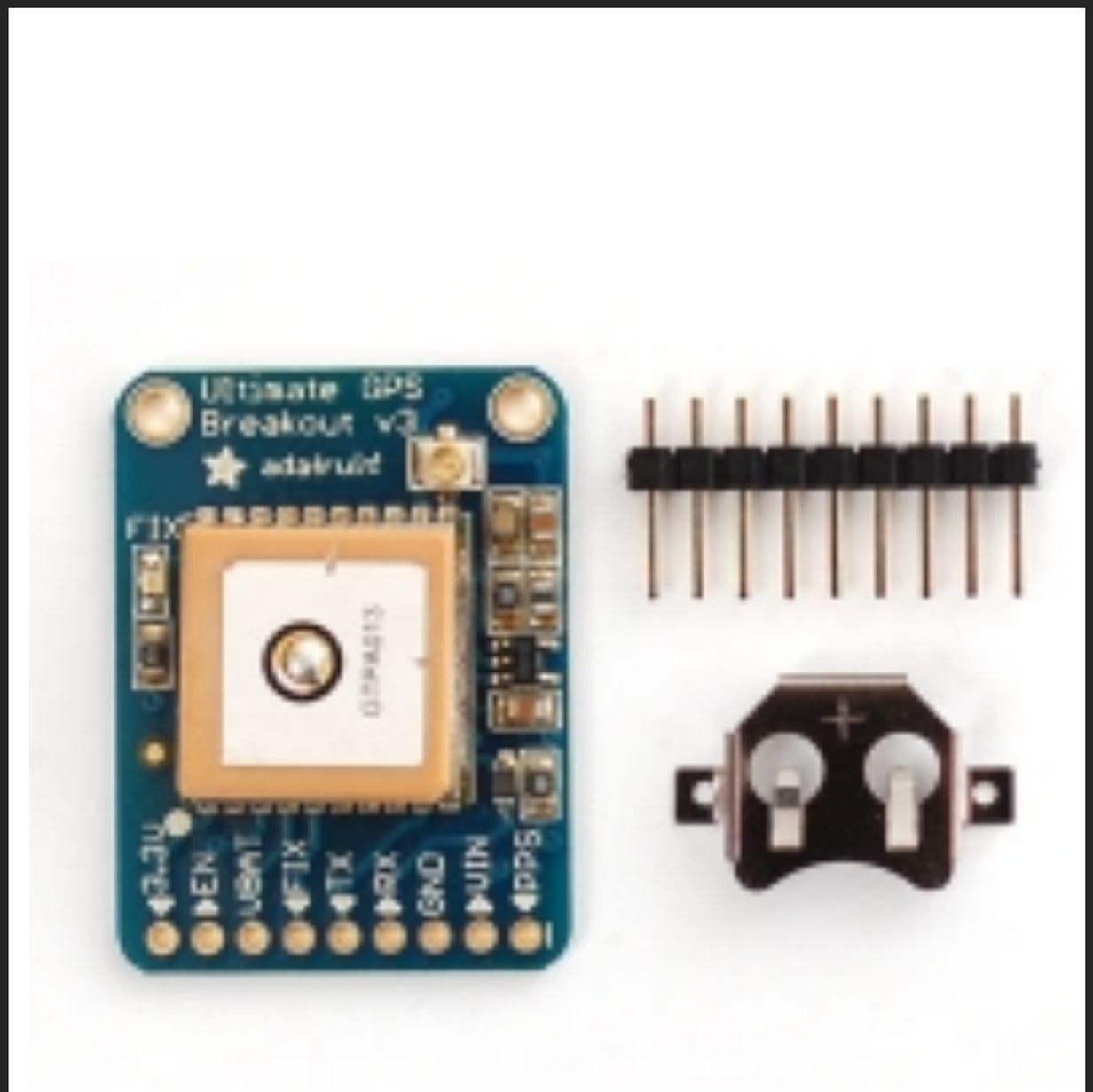
3D Kompass und Bes

Barometric Pressur



Drucksensor

Fingerprint Scanne



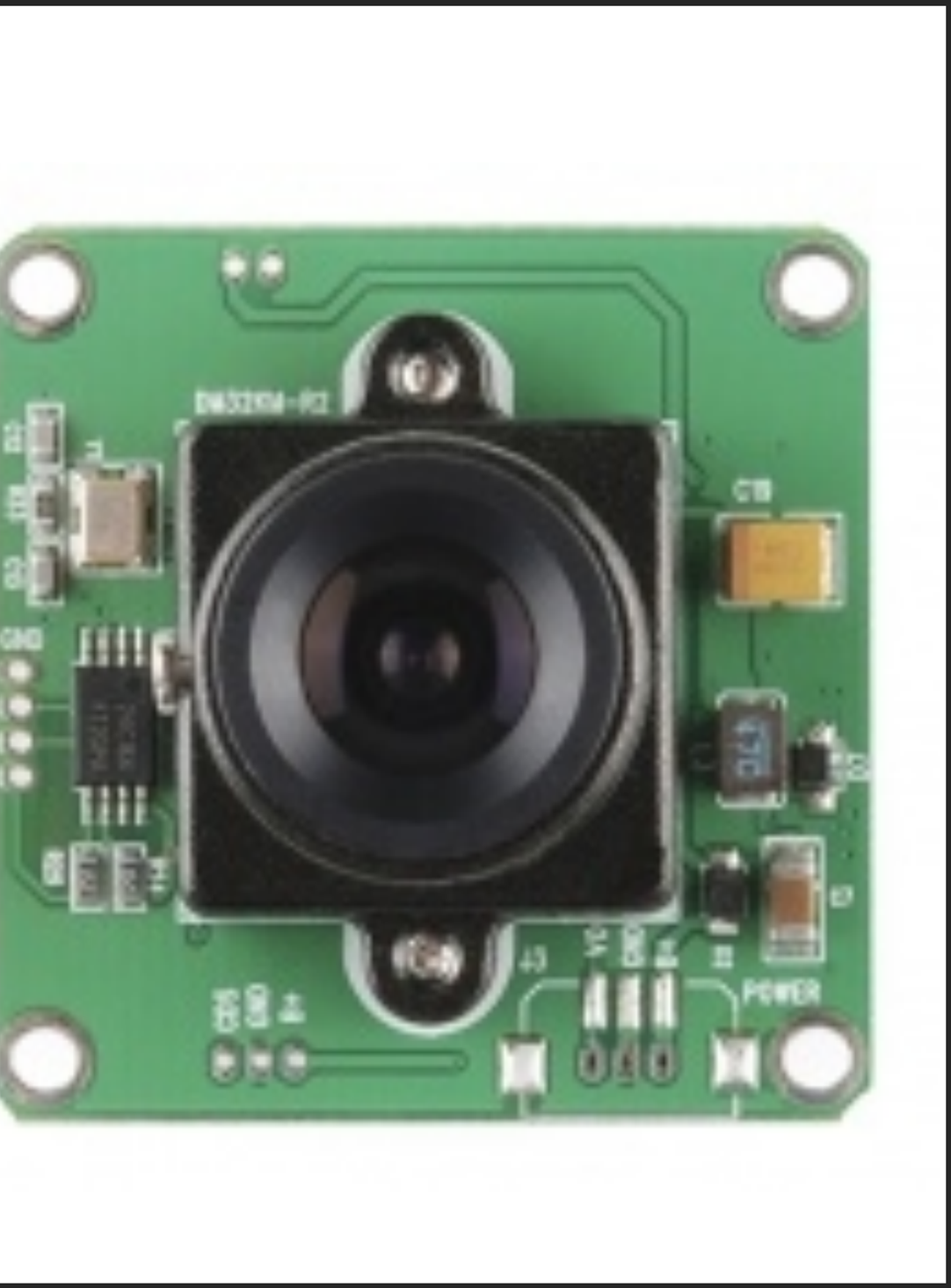
GPS Receiver

Kompass mit Neigu

Sonar

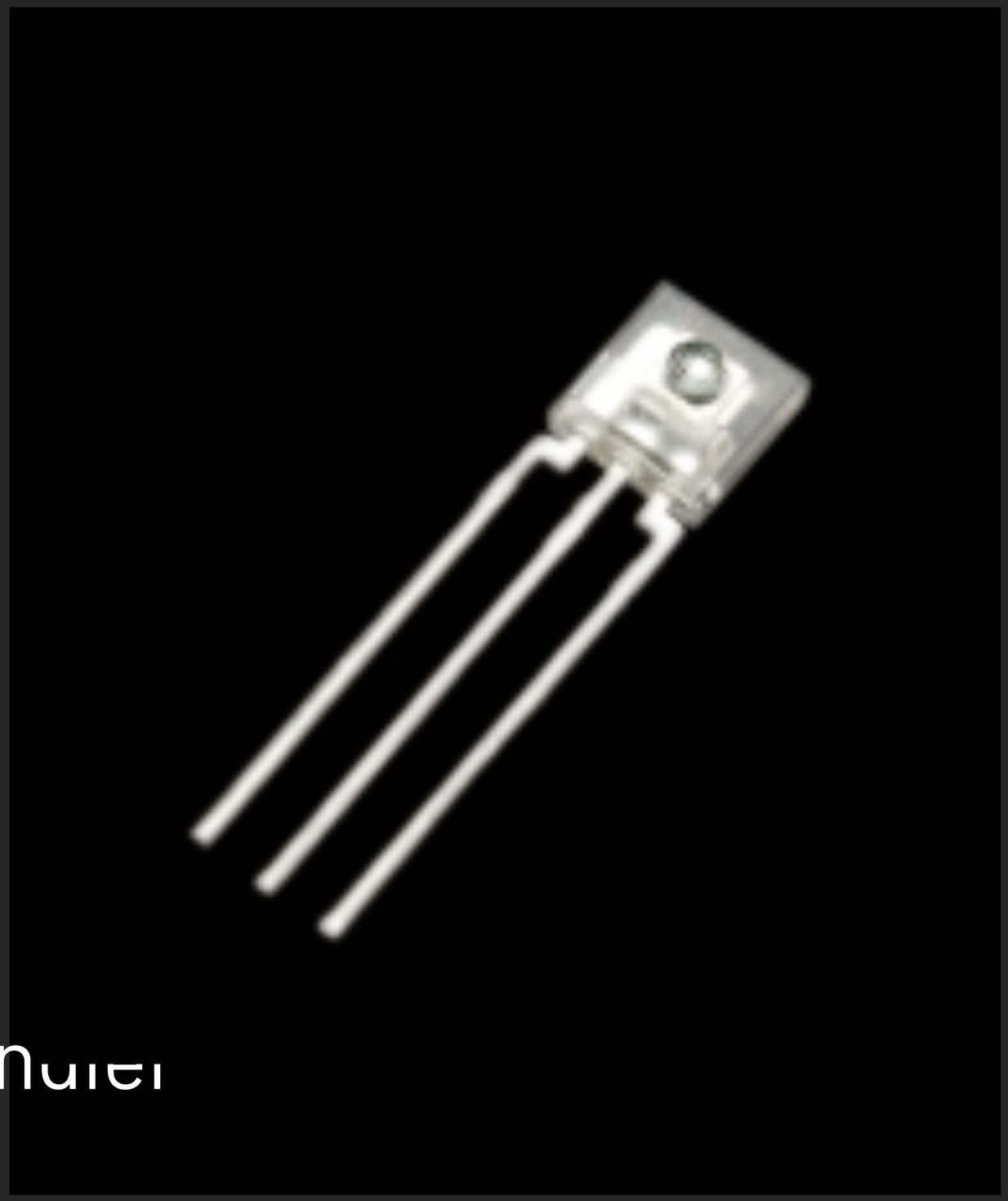
Ultraschall Entfernu

Infrarot Entfernung



CMOS Kamera-mod

Fotowider-stand



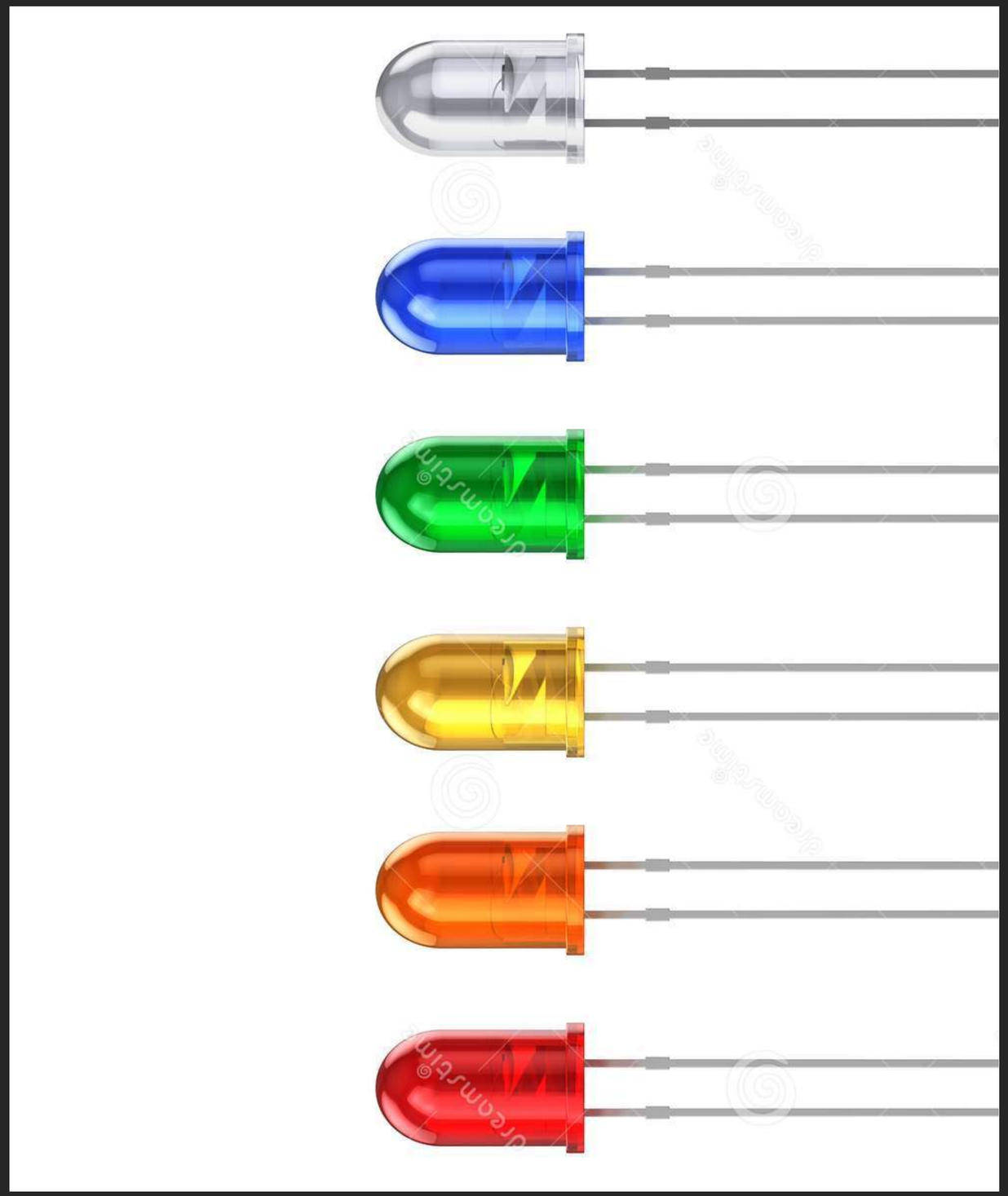
Licht Frequenz Wandler

Optischer Staubsen

Aktoren

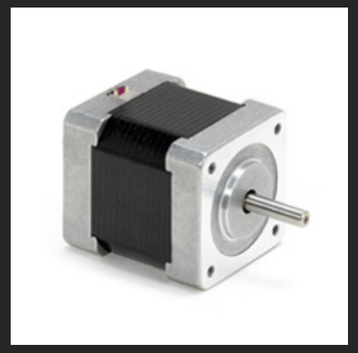
2021 Michael Reichart, copyleft CC BY-NC-SA

LED



Servo

Elektro-motor



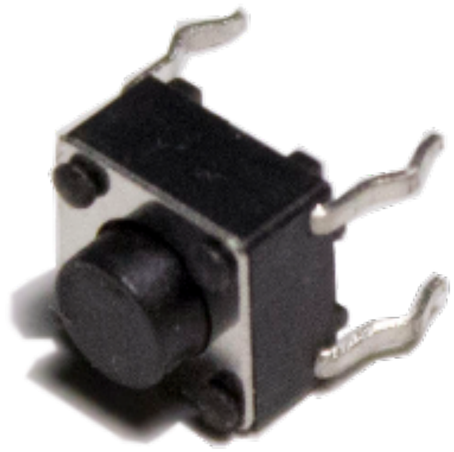
Vibrations-motor

Elektronische Bauteile

2021 Michael Reichart, copyleft CC BY-NC-SA

Widerstände

Potentiometer



Taster

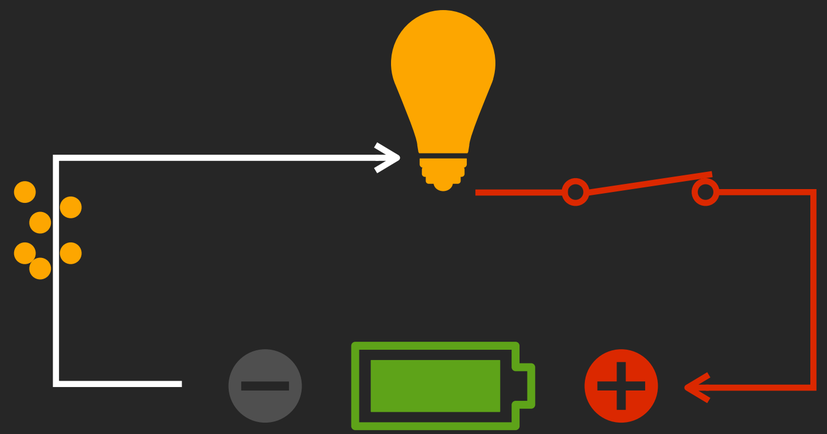
Strom, Spannung, Widerstand

2021 Michael Reichart, copyleft CC BY-NC-SA

Stromstärke

$6{,}24 \times 10^{18}$
(1 Coulomb) Elektronen
fließen pro Sekunde
= 1 Ampere

Anzahl der
Elektronen



Einheiten der Stromstärke

1 Ampere = 1 A
1 Milliampere = 1 mA
1 Mikroampere = 1 µA

In Formelgleichungen wird die Stromstärke durch das Formelzeichen I (Intensität) ausgedrückt.

$$\frac{U}{R \cdot I} = 1$$

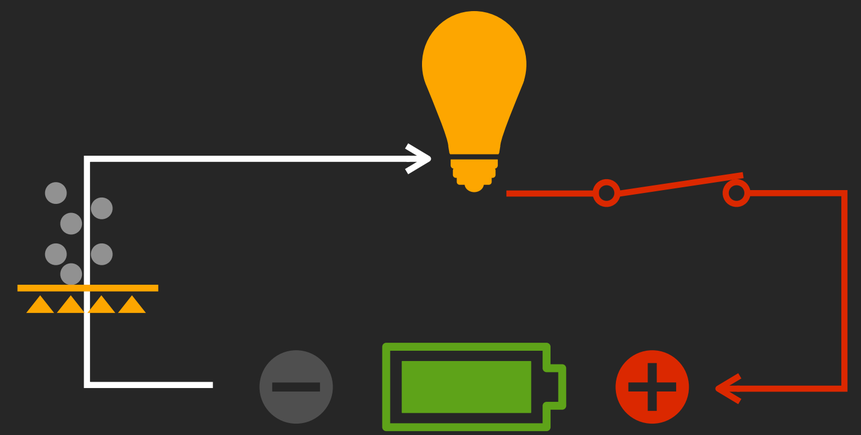
Ohmsches Gesetz

Symbol für
Wechselstrom

Symbol für
Gleichstrom

Elektrische Spannung

Kraft, mit der die Elektronenbewegung vorangetrieben wird,



Einheiten der Spannung

1 Volt = 1 V
1 Millivolt = 1 mV

In Formelgleichungen wird die Spannung durch das Formelzeichen U (Intensität) ausgedrückt. Im Angelsächsischen wird V (Voltage) verwendet.

$$\frac{U}{R \cdot I} = 1$$

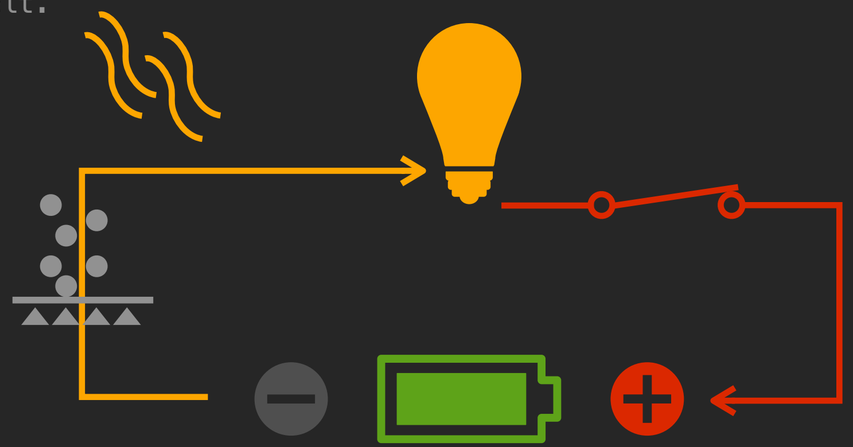
Ohmsches Gesetz

V_{cc} V_{IN}

Beschriftungen für Spannung

Widerstand

Der Widerstand eines Leitermaterials bewirkt einen Spannungsabfall. Spannung wird in thermische Energie (Wärme) umgewandelt.



Maß dafür, welche Spannung erforderlich ist, um eine Stromstärke durch einen Leiter fließen zu lassen

Einheiten der Spannung

1 Ohm
1 kOhm = 1 Kiloohm

In Formelgleichungen wird der Widerstand durch das Formelzeichen R (Resistor) ausgedrückt.

$$\frac{U}{R \cdot I} = 1$$

Ohmsches Gesetz

Ω

Omega als Symbol für Ohm

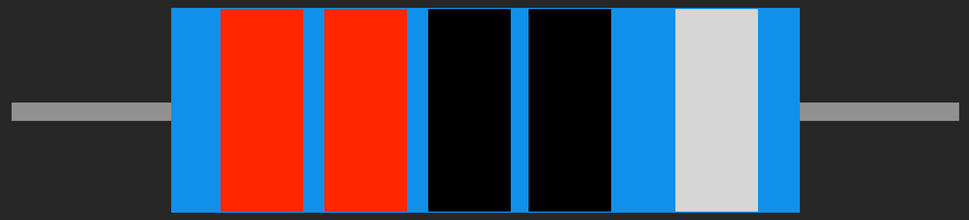
Widerstände

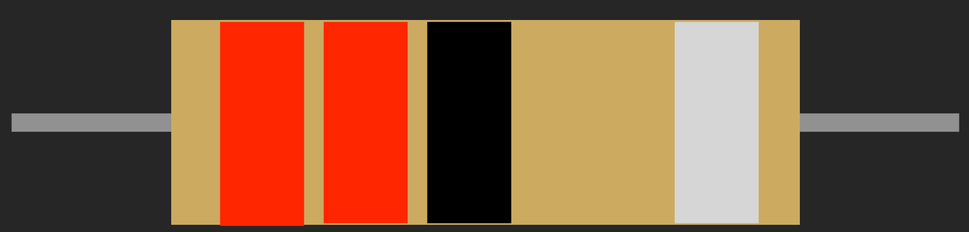
2021 Michael Reichart, copyleft CC BY-NC-SA

	Zahl	Zahl	Faktor	Toleranz
			x 0.01	± 10%
			x 0.1	± 5%
	0	0	x 1	
	1	1	x 10	± 1 %
	2	2	x 100	± 2 %
	3	3	x 1 000	
	4	4	x 10 000	
	5	5	x 100 000	± 0,5 %
	6	6	x 1 000 000	± 0,25 %
	7	7		± 0,1 %
	8	8		
	9	9		

	Zahl	Zahl	Zahl	Faktor	Toleranz	
				x 0.01	± 10%	
				x 0.1	± 5%	
	0	0	0	x 1		
	1	1	1	x 10	± 1 %	
	2	2	2	x 100	± 2 %	
	3	3	3	x 1 000		
	4	4	4	x 10 000		
	5	5	5	x 100 000	± 0,5 %	
	6	6	6	x 1 000 000	± 0,25 %	
	7	7	7		± 0,1 %	
	8	8	8			
	9	9	9			

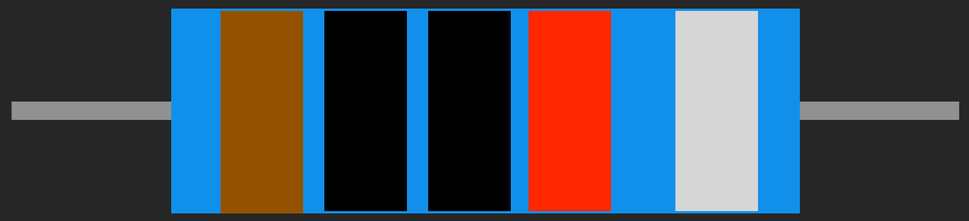
220 Ohm Metallschicht-Widerstand

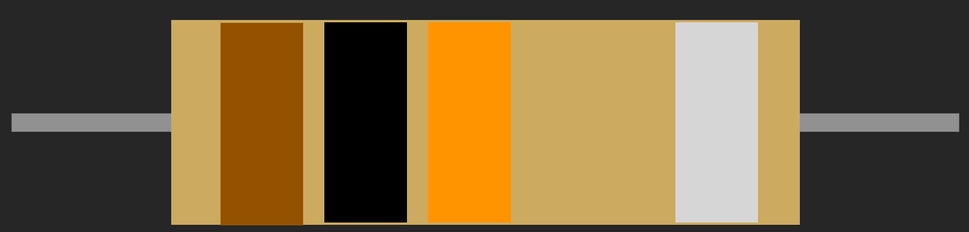




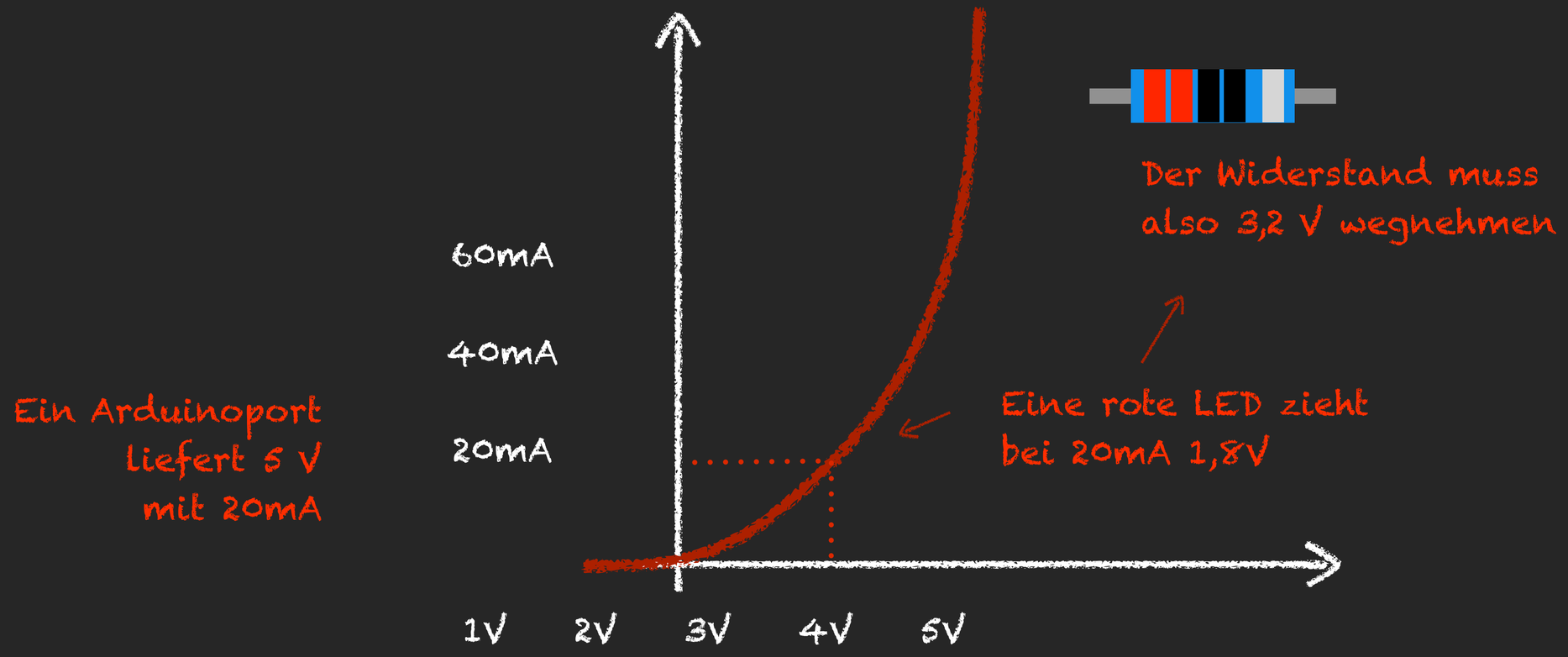
220 Ohm Kohleschicht-Widerstand

10 KOhm Metallschicht-Widerstand





10 KOhm Kohleschicht-Widerstand



Widerstand = Spannung / Stromstärke

R = U / I

Ohm = Volt / Ampere

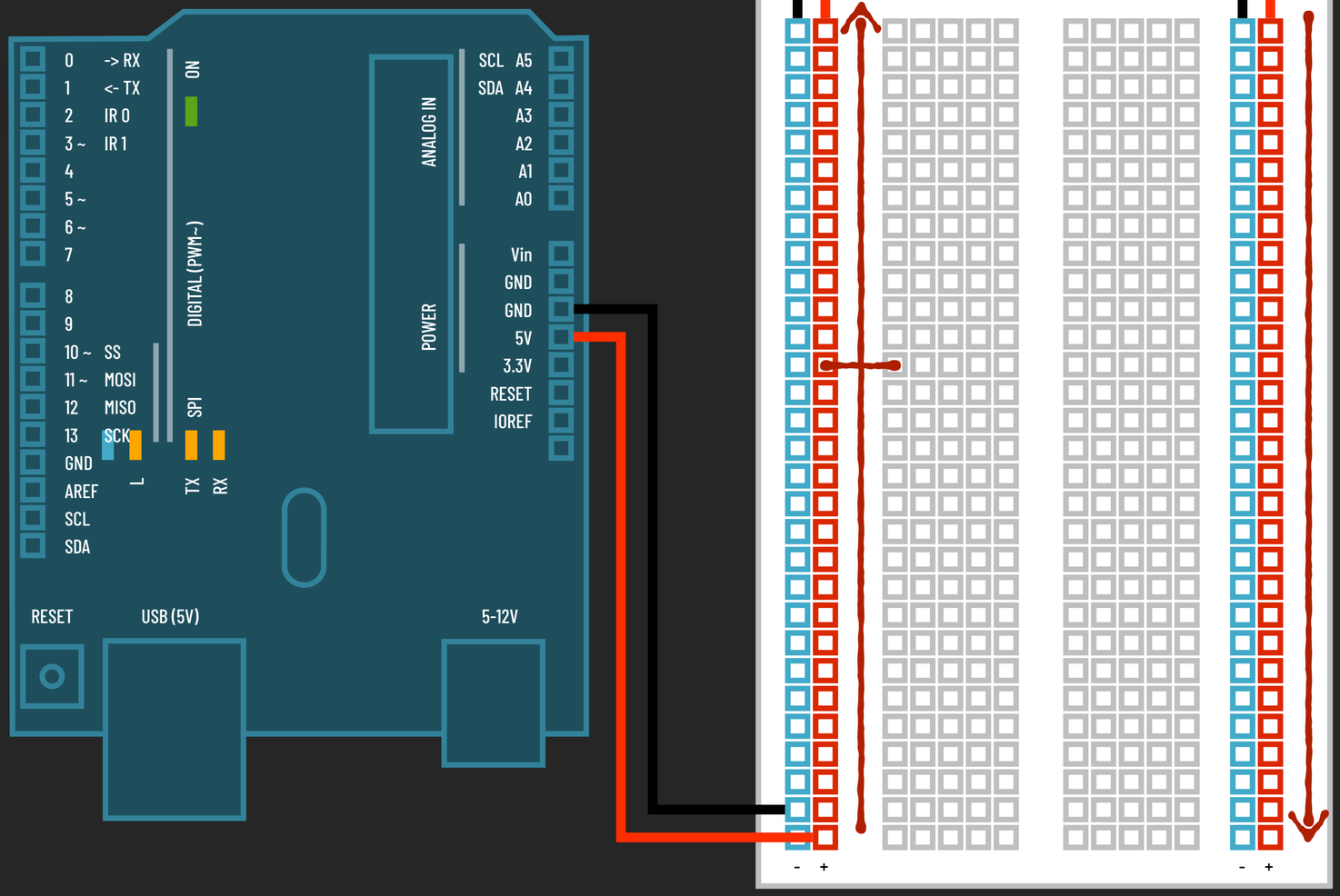
3,2 V / 20 mA (0,02A) = **160 Ohm**

Der Widerstand muss
also 3,2 V wegnehmen

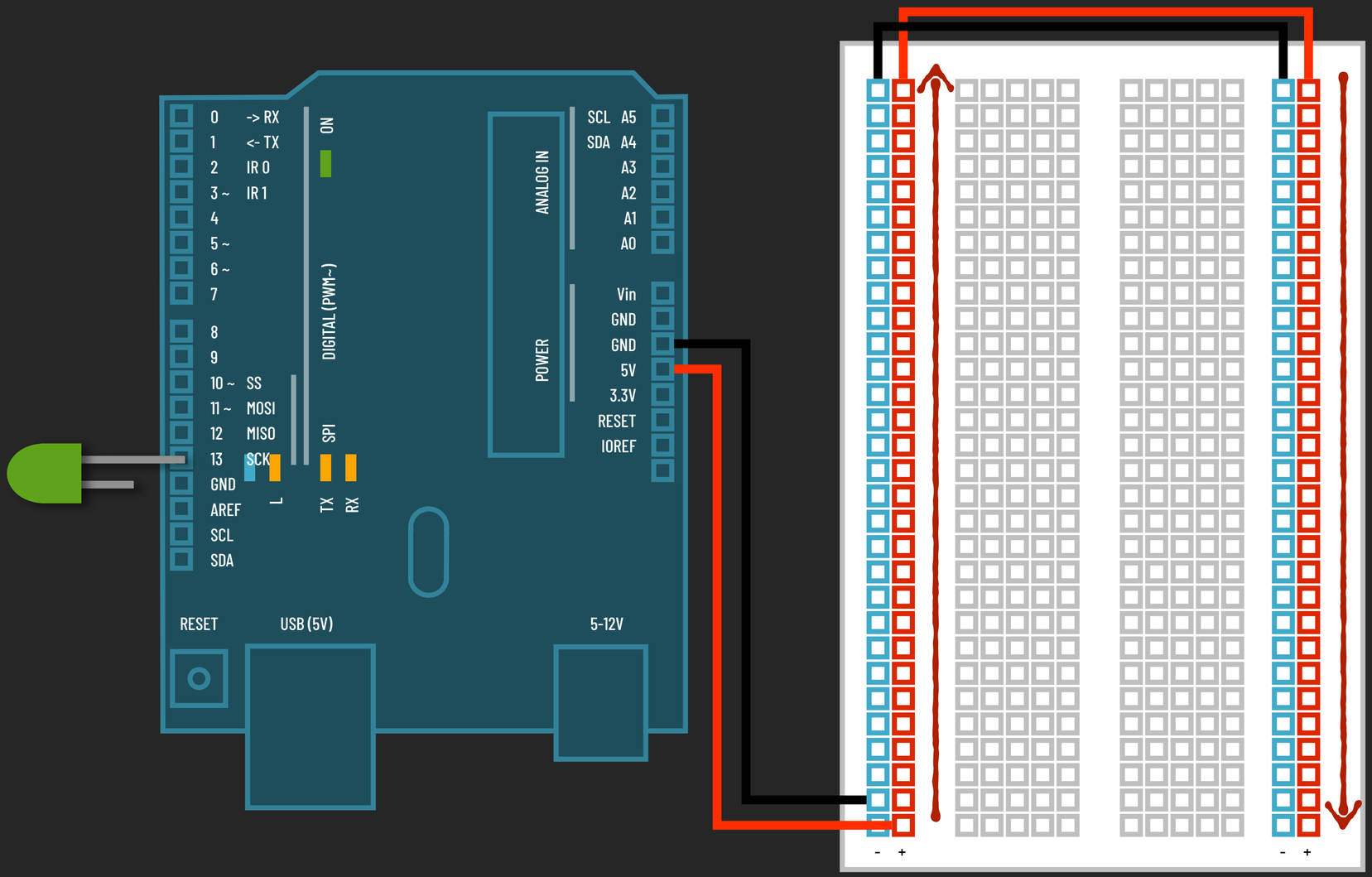
Layouts und Programmierung

2021 Michael Reichart, copyleft CC BY-NC-SA

Anschluß und
Aufbau eines Breadboards



Anschluß und
Aufbau eines Breadboards



Digitale und analoge Pins konfigurieren, lesen und schreiben

Digitale Pins 0 - 13

Werte: 0 (LOW), 1 (HIGH)
Spannung/keine Spannung

Pin als Eingabe nutzen:

```
pinMode(pin, INPUT)
digitalRead()
```

-> zum Lesen von Sensorwerten

Pin zur Ausgabe nutzen:

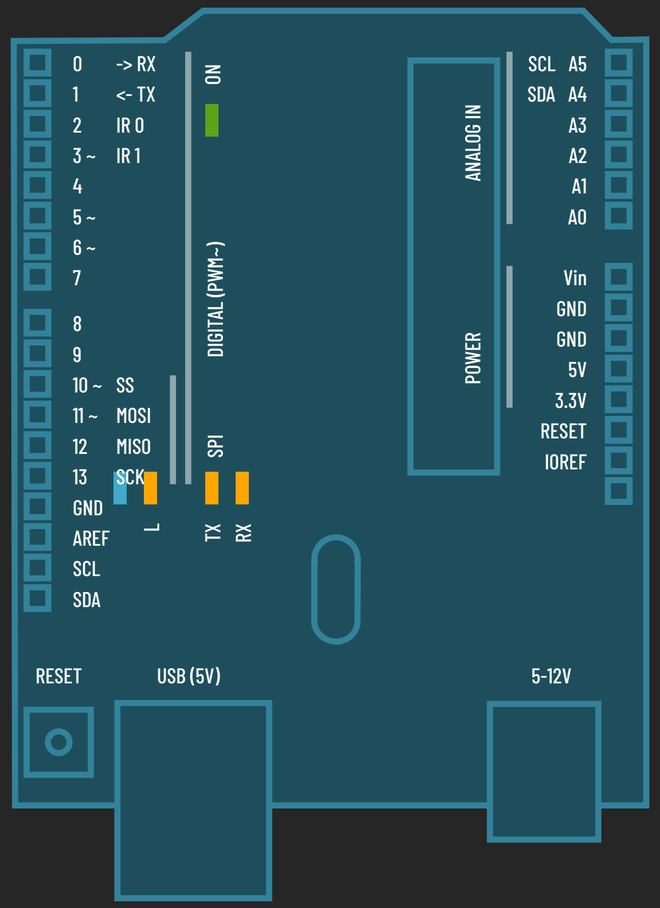
```
pinMode(pin, OUTPUT)
digitalWrite(pin, LOW|HIGH)
```

-> zum Steuern von Aktoren,
z. B. einer LED

PWM Pins (3, 5, 6, 9, 10, 11)

```
analogWrite(pin, 0–255)
```

-> zum Steuern von regelbaren Aktoren, z. B. zum "Dimmen" einer LED.



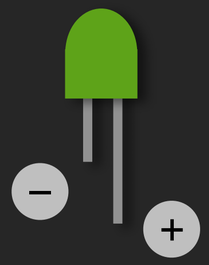
Analoge Pins A0 - A5

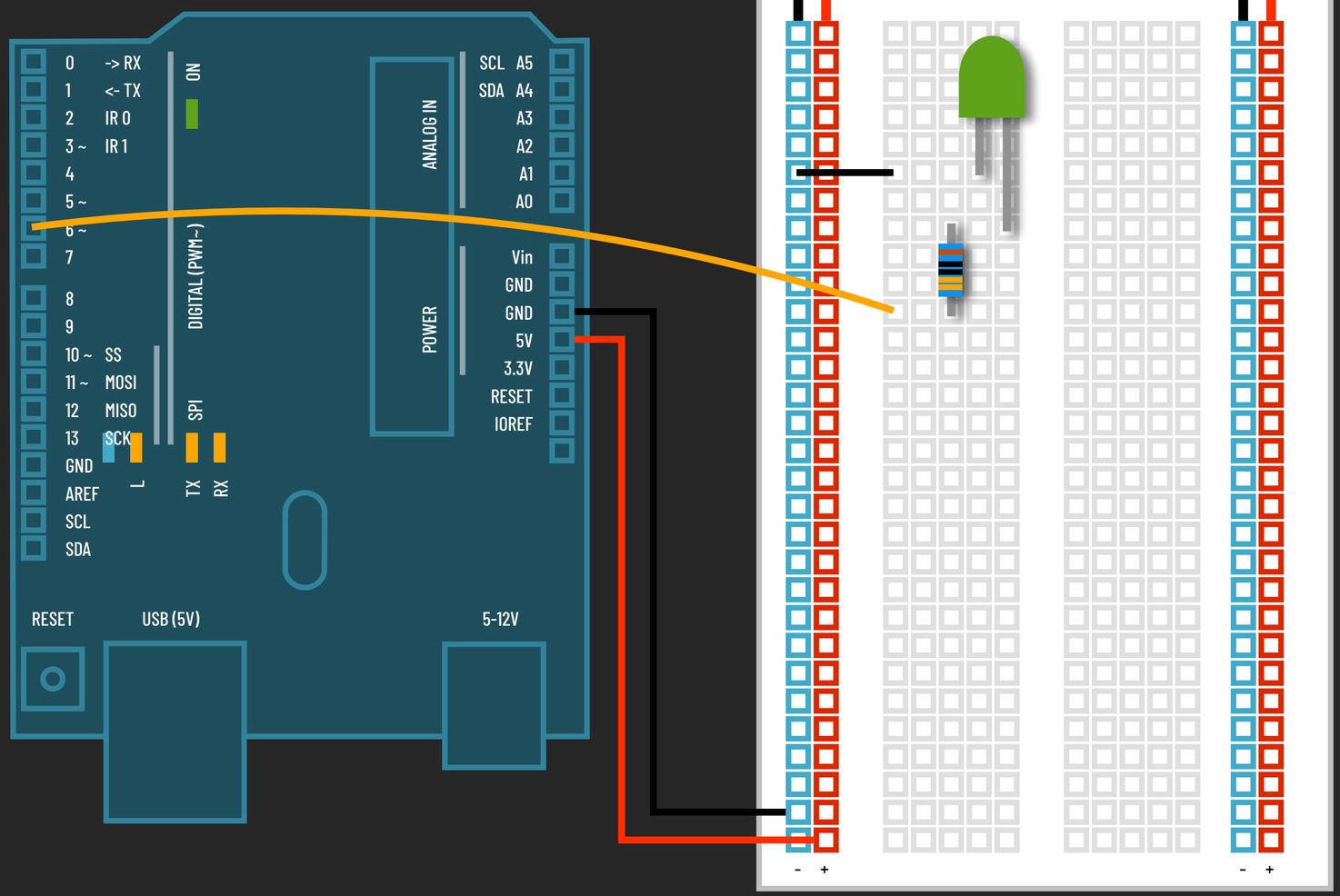
Werte von 0 - 1023 für verschiedene eingehende Spannungen, z. B. von einem Potentiometer oder einer Fotodiode. (10 bit)

KEIN pinMode() nötig.

```
analogRead(pin)
```

Ansteuern einer LED mit einem Pullup-Widerstand (330 Ohm)





Für den Betrieb einer LED sind ~3V ausreichend.

D. h. 5V sind zuviel und können die LED zerstören.

Deshalb wird ein Widerstand dazwischengesteckt, der den "zuvielen" Strom wegfrisst, so dass nur noch ca. 2-3V bei der LED ankommen.

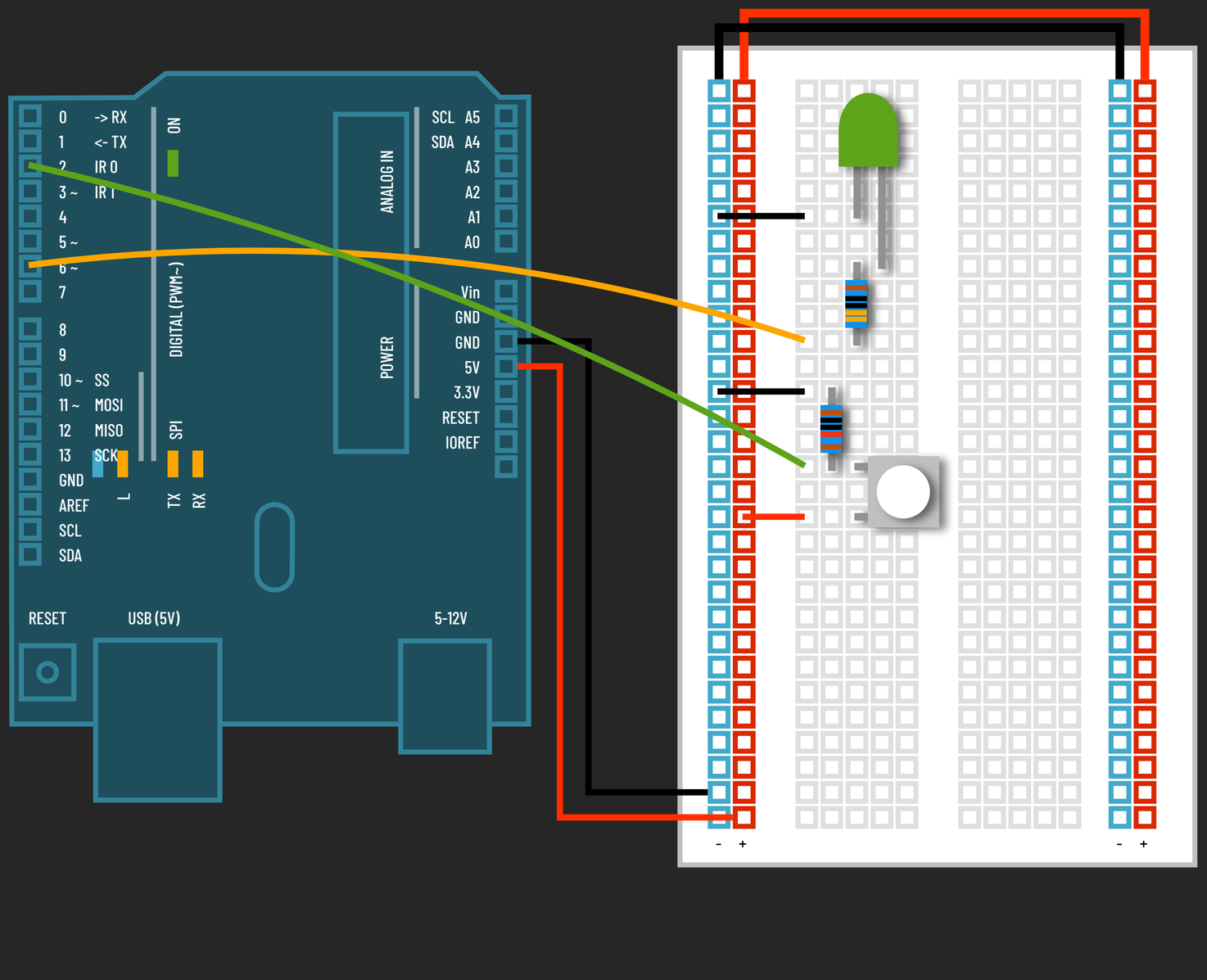
*Tipp: Der **Pin 13** hat einen eingebauten Vorwiderstand, so dass die LED korrekt mit Spannung versorgt wird.*

```
void setup(){
  pinMode(6, OUTPUT);
}

void loop(){
  digitalWrite(6, HIGH);
  delay(1000);
  digitalWrite(6, LOW);
  delay(1000);
}
```

Blinking LED

```
const int PIN = 6;

void setup() {
 pinMode(PIN, OUTPUT);
}

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

LED mit Taster schalten und mit dem Potentiometer dimmen



Ein Taster wir mit einem 10kOhm Widerstand auf "Null runtergezogen" - Wenn er nicht gedrückt wird, ist die Restspannung so klein, dass sie nichts mehr auslöst. Wird der Taster gedrückt, werden 5V durchgeleitet.

```
void setup(){
  pinMode(5, INPUT)
}

void loop(){
  int state = digitalRead(5);
}
```

Das Potentiometer ist ein regelbarer Widerstand mit 10 kOhm. Er liefert je nach Stellung eine Spannung zwischen 0 und 5 V. Die Spannung wird über einen analogen Input-Pin abgegriffen und in einer 10 Bit-Auflösung in Werten zwischen 0 und 1023 ausgegeben. Die LED lässt sich mit einem PWM-Wert zwischen 0 und 255 dimmen.

```
void setup(){}

void loop(){
  int value = analogRead(A1);
  analogWrite(map(value, 0,1023, 0,255));
}
```

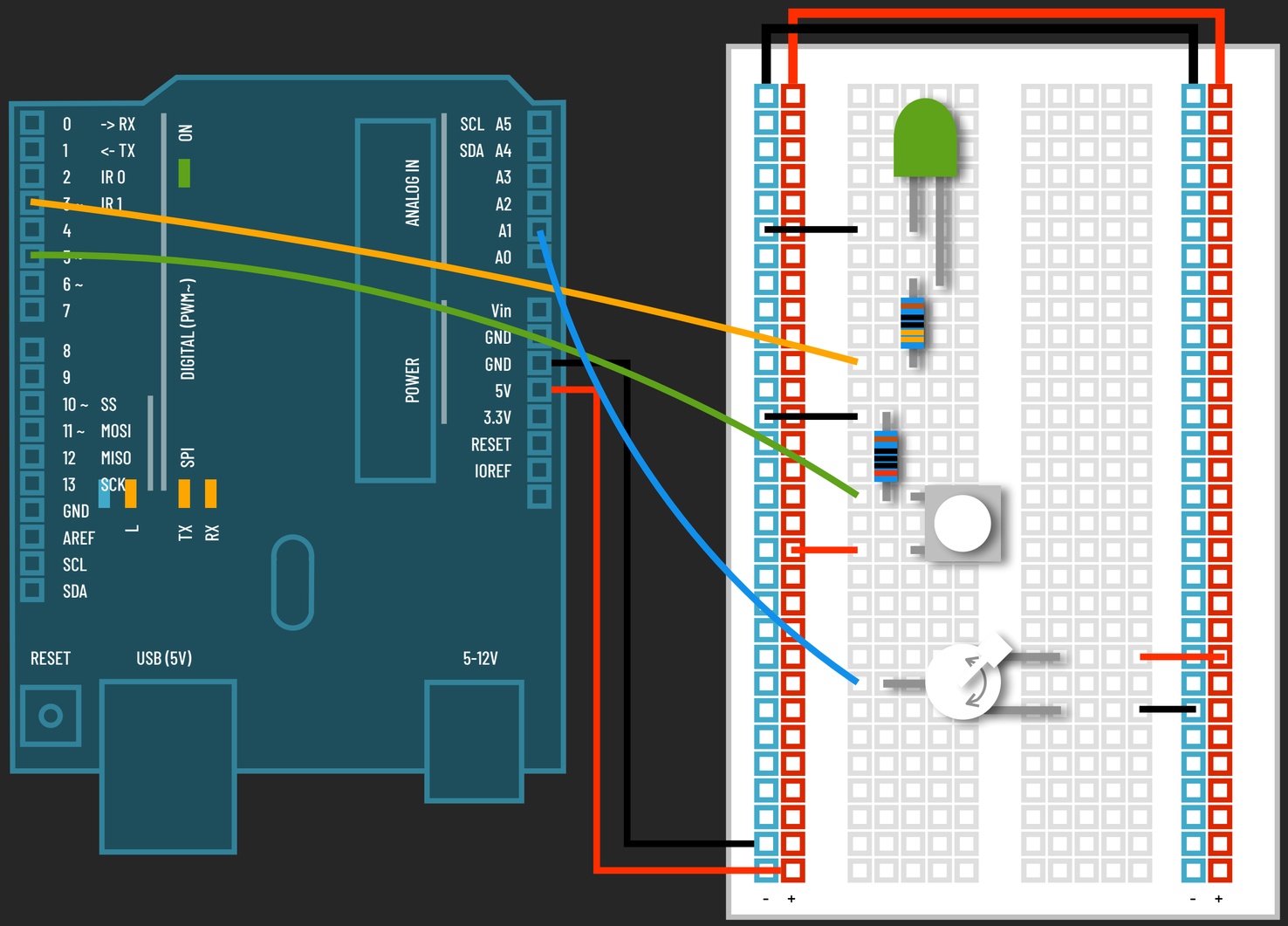
Programmsteuerung mit `if`

```
if (btnState == HIGH) {
    // Der Taster ist gedrückt.;
  } else {
    // Der Taster ist NICHT gedrückt;
  }

// BEREICH ABFRAGEN
if (potValue >= 0 && potValue <= 100) {
    // Zwischen 0 und 100;
} else if (potValue > 100 && potValue <= 400) {
    // Zwischen 101 und 400.;
} else if (potValue > 401 && potValue <= 800) {
    // Zwischen 401 und 800.;
} else {
    // Über 801.;
}
```

LED mit Taster und Interrupt

```
const int LED_PIN = 6;
const int INTERRUPT_PIN = 2;
volatile bool ledState = LOW;

void setup() {
    pinMode(LED_PIN, OUTPUT);
    pinMode(INTERRUPT_PIN, INPUT);
    attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), myISR, FALLING);
// trigger when button pressed, but not when released.
}

void loop() {
    digitalWrite(LED_PIN, ledState);
}

void myISR() {
    ledState = !ledState;
    // note: LOW == false == 0, HIGH == true == 1, so inverting the
boolean is the same as switching between LOW and HIGH.
}
```

LED mit Taster schalten und mit dem Potentiometer dimmen



Ein Taster wir mit einem 10kOhm Widerstand auf "Null runtergezogen" - Wenn er nicht gedrückt wird, ist die Restspannung so klein, dass sie nichts mehr auslöst. Wird der Taster gedrückt, werden 5V durchgeleitet.

```
void setup(){
  pinMode(5, INPUT)
}

void loop(){
  int state = digitalRead(5);
}
```

Das Potentiometer ist ein regelbarer Widerstand mit 10 kOhm. Er liefert je nach Stellung eine Spannung zwischen 0 und 5 V. Die Spannung wird über einen analogen Input-Pin abgegriffen und in einer 10 Bit-Auflösung in Werten zwischen 0 und 1023 ausgegeben. Die LED lässt sich mit einem PWM-Wert zwischen 0 und 255 dimmen.

```
void setup(){}

void loop(){
  int value = analogRead(A1);
  analogWrite(map(value, 0,1023, 0,255));
}
```

Potentiometer auf dem seriellen Monitor

```
int potPin = A0;
int potValue;

void setup() {
  Serial.begin(9600);
  Serial.println("Ready.");


  pinMode(potPin, INPUT);
  potValue = analogRead(potPin);
}

void loop() {
   potValue = analogRead(potPin);
   Serial.println(potValue);
   delay(100);
 }
```

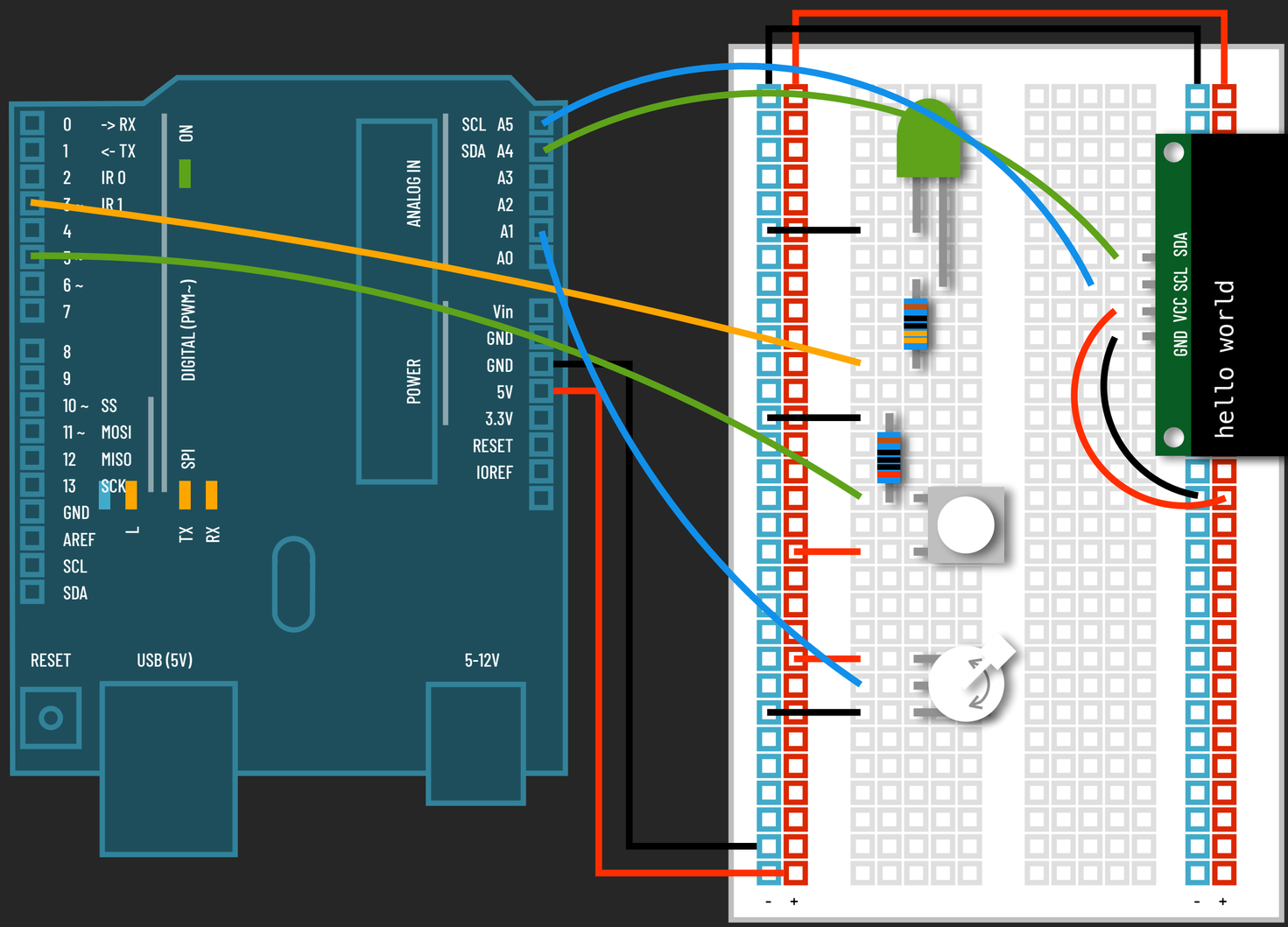
Potentiometer mit Threshold

```
const byte POTENTIOMETER_PIN = A0;
const byte LED_PIN = 13;
const byte LED_DIMMER_PIN = 11;

int potentiometerValue = 0;
int treshold = 512;
float voltage = 0;
byte pwmValue = 0;

void setup(){
 Serial.begin(9600);

  pinMode(POTENTIOMETER_PIN, INPUT);
  pinMode(LED_PIN, OUTPUT);

 potentiometerValue = analogRead(POTENTIOMETER_PIN);
}

void loop(){
  // read the analog value
  potentiometerValue = analogRead(POTENTIOMETER_PIN);
  Serial.print(potentiometerValue);
  Serial.println();

  // Spannungwert ausrechnen
  voltage = (potentiometerValue / 1023.0) * 5;
  Serial.print(voltage);
  Serial.print(" V");
  Serial.println();

  // LED fade in/out
  pwmValue = map(potentiometerValue, 0,1023, 0,255);
  analogWrite(LED_DIMMER_PIN, pwmValue);

  // HIGH the led, when sensor value is over threshold
  if (potentiometerValue >= treshold) {
    digitalWrite(LED_PIN, HIGH);
  } else {
    digitalWrite(LED_PIN, LOW);
  }
  delay(50);
}
```

Potentiometer für RGB LED

```
int button = 8;
int buttonstate = 0;

//potty value 0 - 1023

int pottyR = A0;
int pottyG = A1;
int pottyB = A2;

int valR = 0;
int valG = 0;
int valB = 0;


int ledR = 9;
int ledG = 10;
int ledB = 11;

int LedValR = 0;
int LedValG = 0;
int LedValB = 0;

void setup() {

    Serial.begin(9600);

    pinMode(pottyR, INPUT);
    pinMode(pottyG, INPUT);
    pinMode(pottyB, INPUT);

    pinMode(ledR, OUTPUT);
    pinMode(ledG, OUTPUT);
    pinMode(ledB, OUTPUT);

    pinMode(button, INPUT);
}

void loop() {
  valR = analogRead(pottyR);
  valG = analogRead(pottyG);
  valB = analogRead(pottyB);

  LedValR = map(LedValR, 0, 1023, 0, 255);
  LedValG = map(LedValG, 0, 1023, 0, 255);
  LedValB = map(LedValB, 0, 1023, 0, 255);

buttonstate = digitalRead(button);

analogWrite(ledR, LedValR);
analogWrite(ledG, LedValG);
analogWrite(ledB, LedValB);

if (buttonstate == HIGH){
  Serial.println(LedValR);
  Serial.println(LedValG);
  Serial.println(LedValB);
  }
}
```

OLED SSD1306 Display

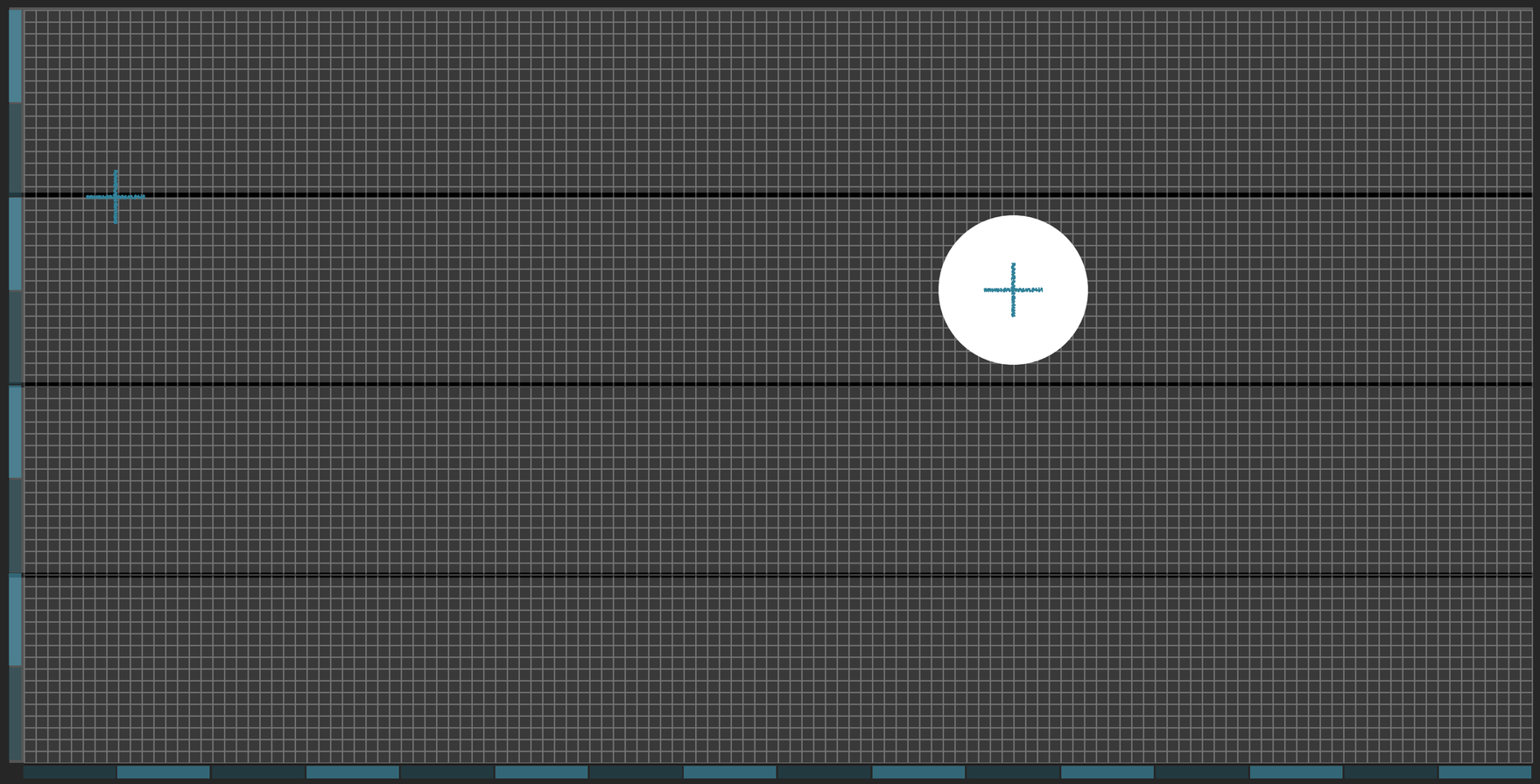


Das OLED SSD1306 wird über den I2C Bus angesteuert. Als Pins werden A5 - "Serial Clock" und A4 - "Serial Data" verwendet.

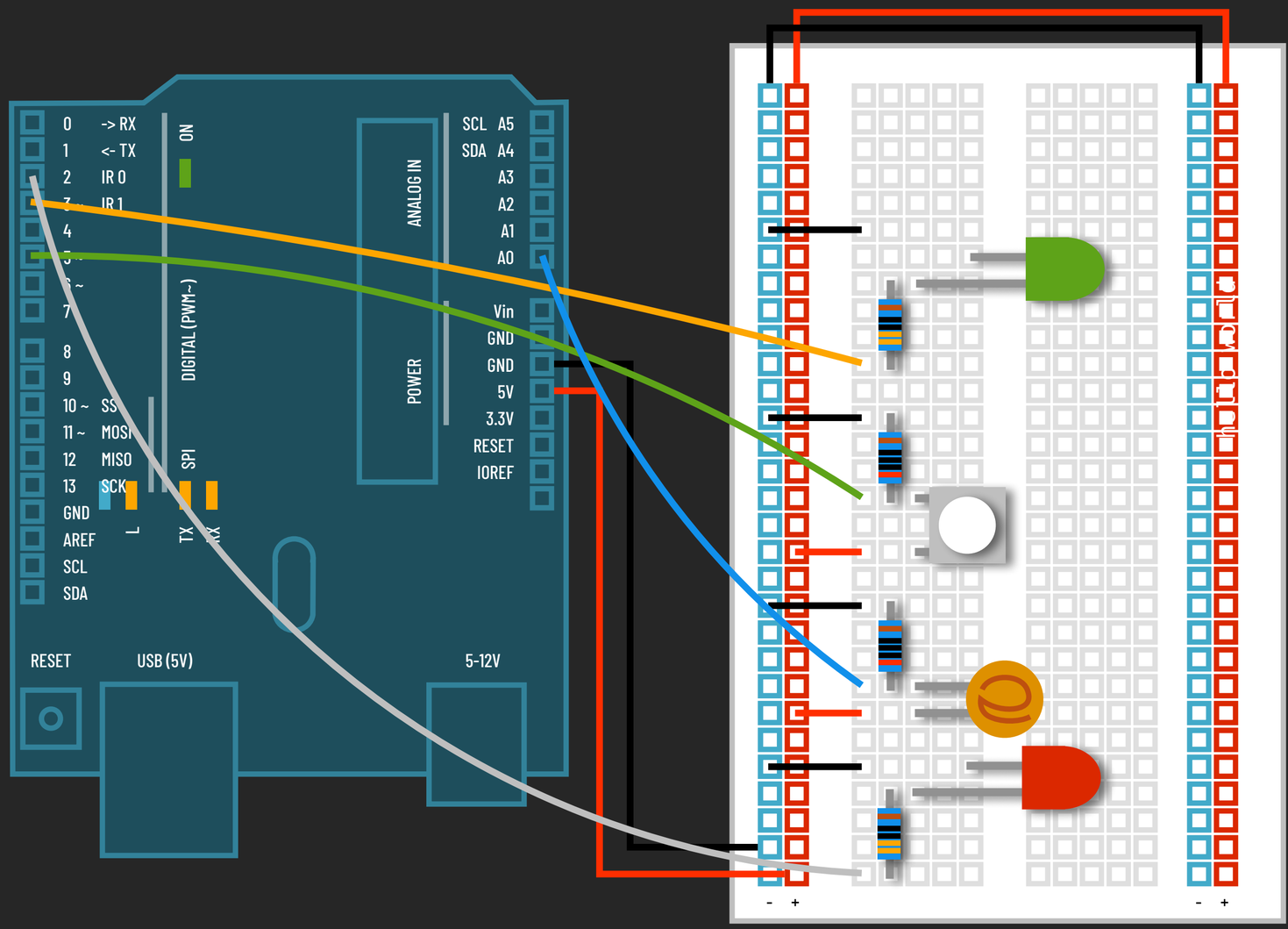
```
#include <U8g2lib.h>
#include <Wire.h>

U8G2_SSD1306_128X64_NONAME_1_HW_I2C ¬
    oled(U8G2_R0, U8X8_PIN_NONE);


void setup(){
  u8g2.begin();
}

void loop(){
 u8g2.firstPage();
  do {
    // Write the pot value
    oled.setFont(u8g2_font_resoledmedium_tr);
    oled.setCursor(0, 15);
    oled.print("hello world");
  } while ( u8g2.nextPage() );
}
```

OLED Bildschirm
128 x 64 Pixel

(0,0)



Photowiderstand




```
const int LDR_PIN = A0;

const int ldrValuelistSize = 100;
int ldrValuelist[ldrValuelistSize];

int ldrValuesSum = 0;
int ldrValuesMedium = 0;
int index = 0;

int ldrValue;

void setup() {
  ldrValue = analogRead(LDR_PIN);
}

void loop() {
  ldrValue = analogRead(LDR_PIN);
  ldrValuelist[index] = ldrValue;

  index = index + 1;
  if (index >= ldrValuelistSize) {
    index = 0;
  }

  ldrValuesSum = 0;

  for (int i = 0; i < ldrValuelistSize; i++)
  {
    ldrValuesSum  = ldrValuesSum + ldrValuelist[i];
  }

  ldrValuesMedium = int(ldrValuesSum / ldrValuelistSize);
}
```

bit

2^1
= 2

byte (8bit)

2^8
= 256

0 – 255

2 byte (16bit)

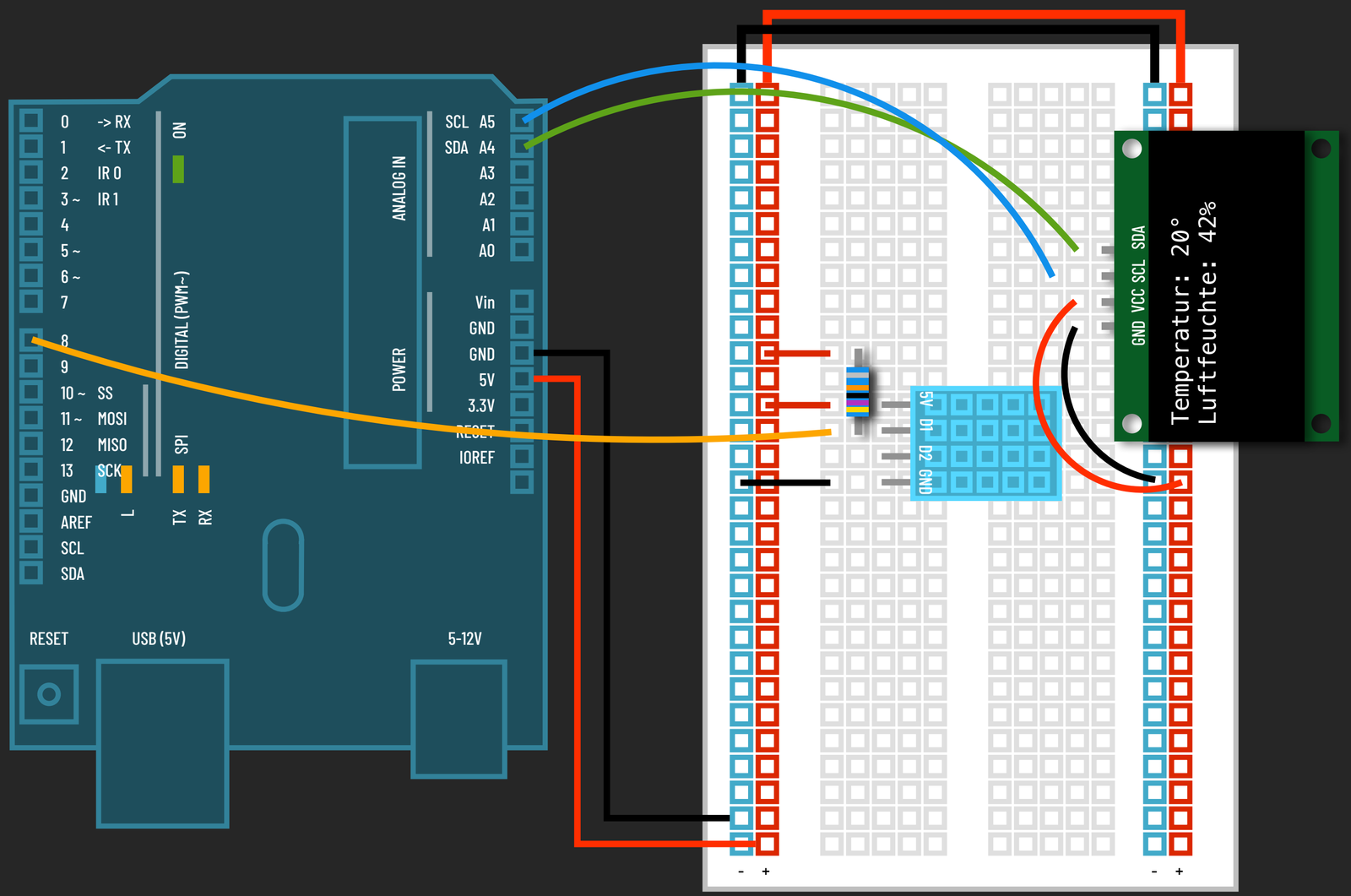
bit

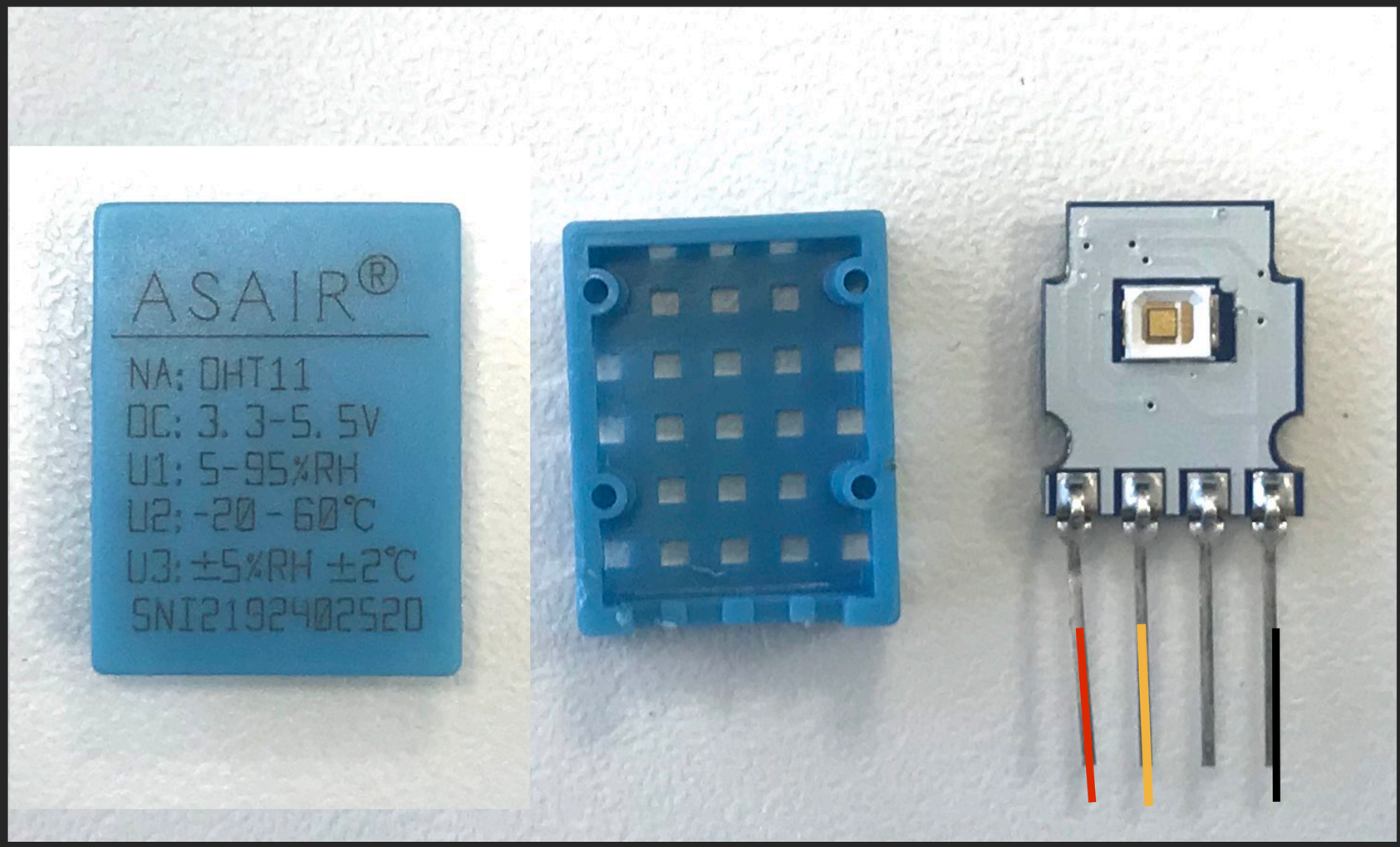
2^1
= 2

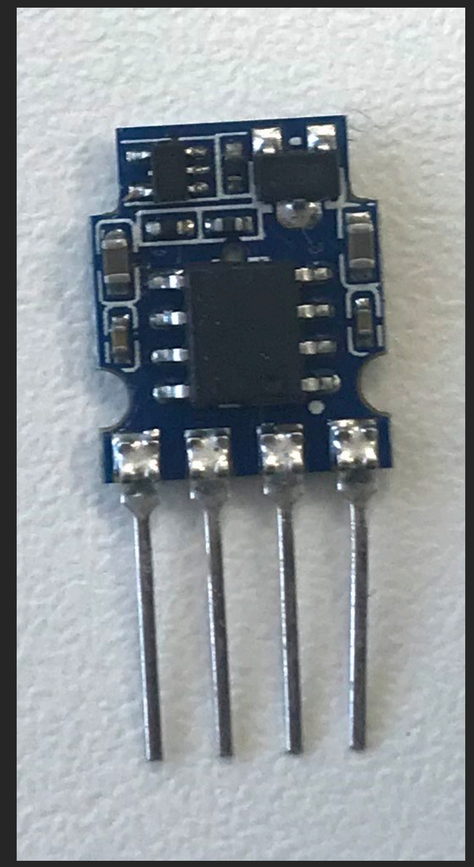
2^6
= 65535

unsigned: 0 – 65534
signed: –32168 – 32167

DHT11 – Temperatur und Luftfeuchte-Sensor (mit Widerstand 4,7–10 kOhm) OLED über i²c Bus







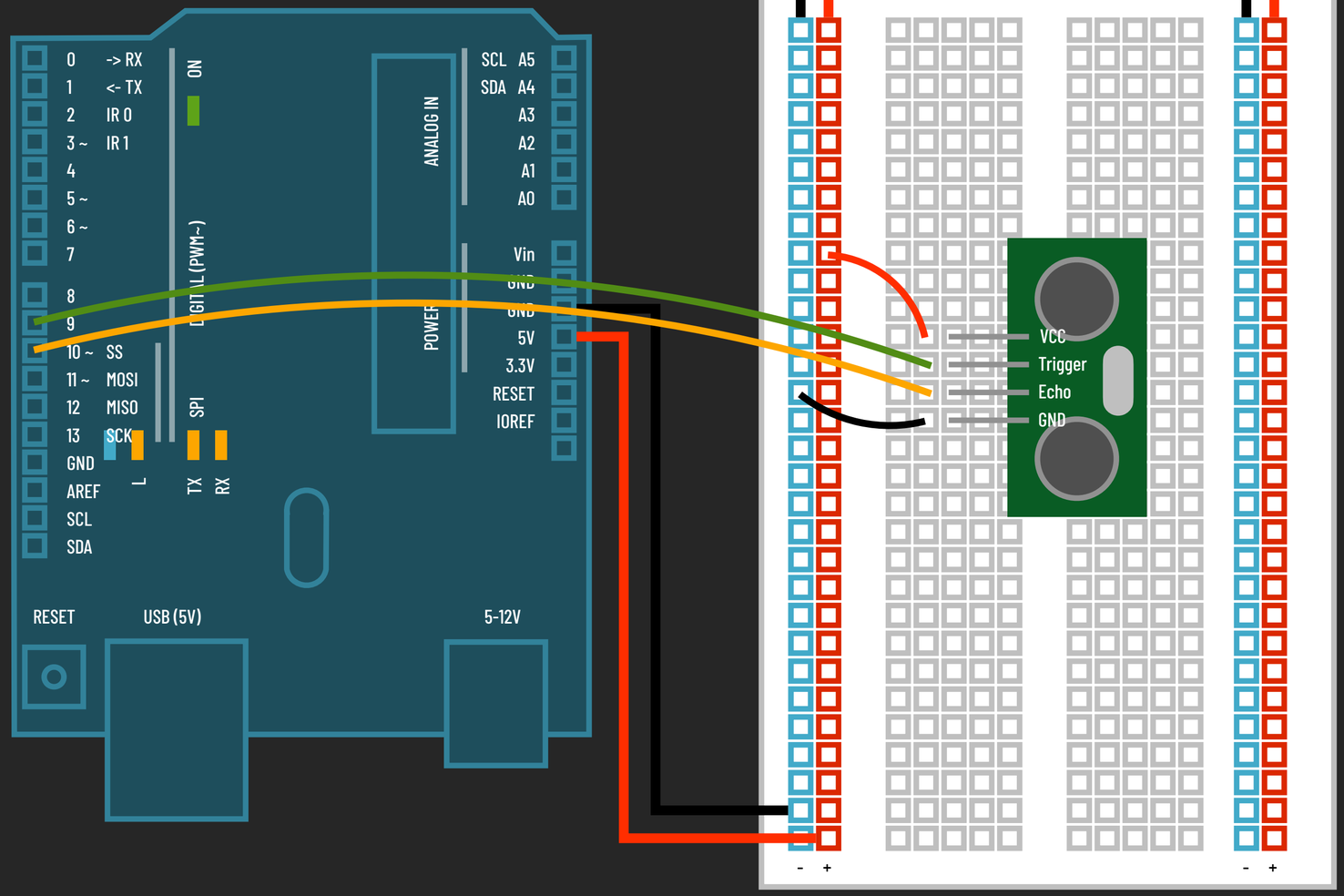
DHT11 MIT OLED

```
#include <Arduino.h>
#include <U8g2lib.h>
#include "DHT.h"
#include <Wire.h>

#define DHTTYPE DHT11   // DHT 11

// Initalize the SSED1306 OLED
U8G2_SSD1306_128X64_NONAME_1_HW_I2C oled(U8G2_R0, /* reset=*/ U8X8_PIN_NONE);

const byte DHTPIN = 8;

// Initialize DHT sensor.
DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);

  dht.begin();

  oled.begin();
  oled.enableUTF8Print();
  oled.setFont(u8g2_font_6x12_me);
}
```

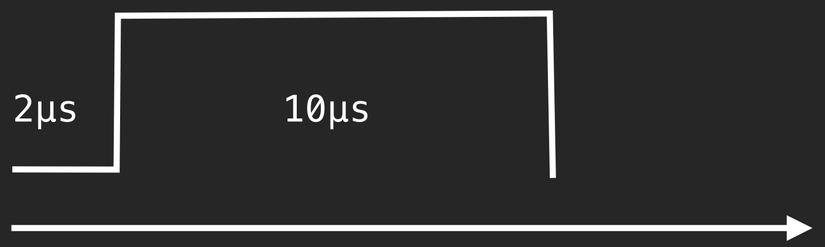
DHT11 MIT OLED

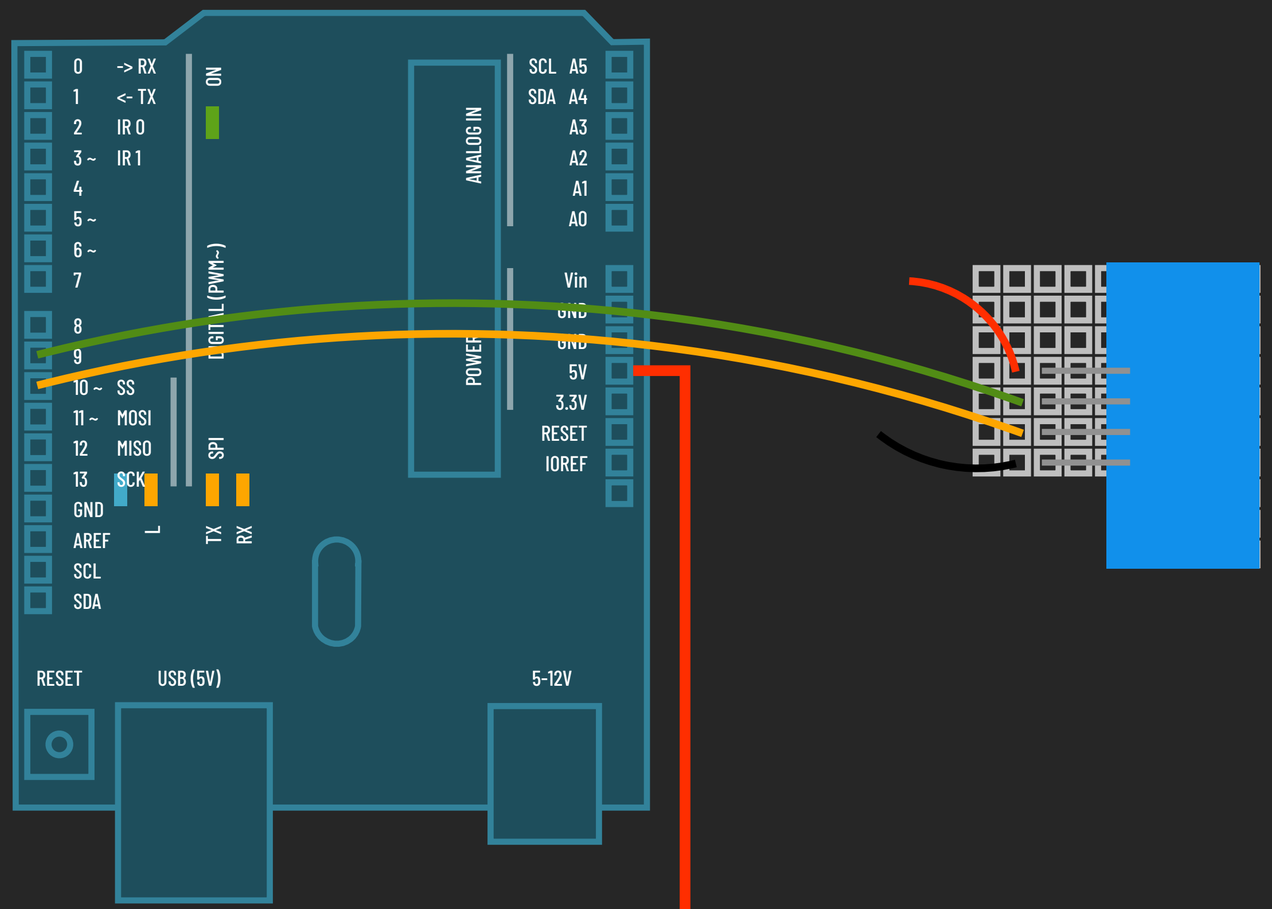
```
void loop() {
  // Wait a few seconds between
  // measurements.
  delay(1000);

  // Reading temperature or humidity
  // takes about 250 milliseconds!
  // Sensor readings may also be up to 2
  // seconds 'old'
  float h = dht.readHumidity();

  // Read temperature as Celsius
  // (the default)
  float t = dht.readTemperature();

  // Check if any reads failed and
  // exit early (to try again).
  if (isnan(h) || isnan(t)) {
    Serial.println(F("Failed to read
from DHT sensor!"));
    return;
  }

  // Serial output
  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.print(F("°C "));
  Serial.println();
  // - - - - - - - - - - - -
  float f = round(t*10)/10;
    Serial.print(t);
    Serial.print(" ");
    Serial.println(f);

  oled.firstPage();
  do {
    oled.setCursor(0,32);
    oled.print(f);
    oled.print("°C");
  } while ( oled.nextPage() );
  // - - - - - - - - - - - -
}
```

Anschluß und
Aufbau eines Breadboards

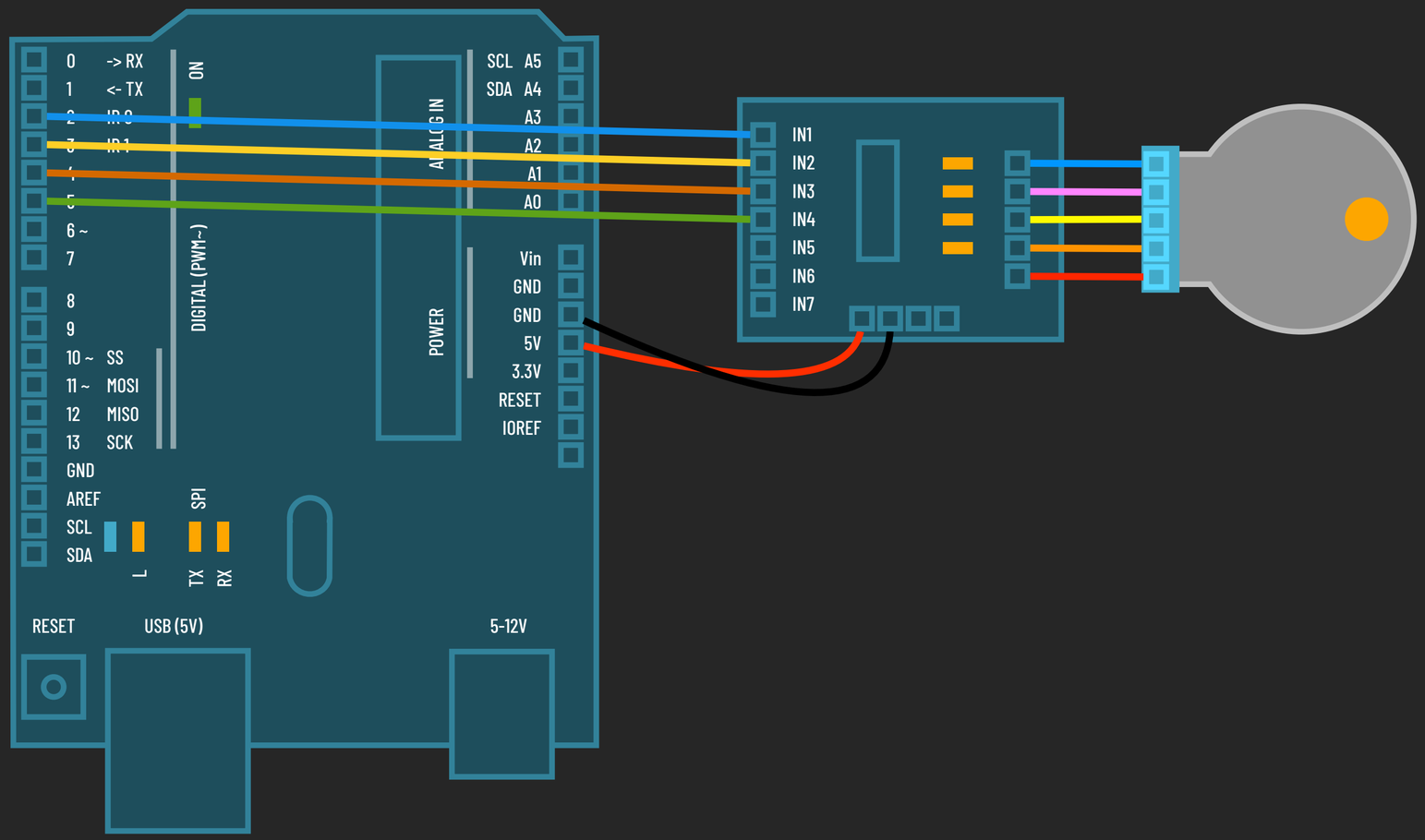


Anschluß und
Aufbau eines Breadboards

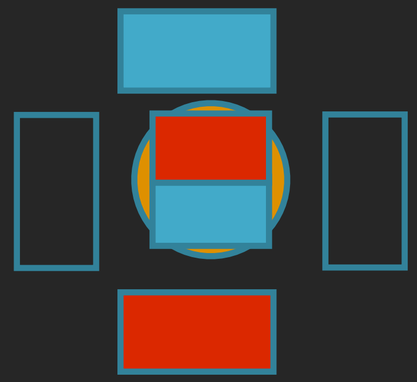


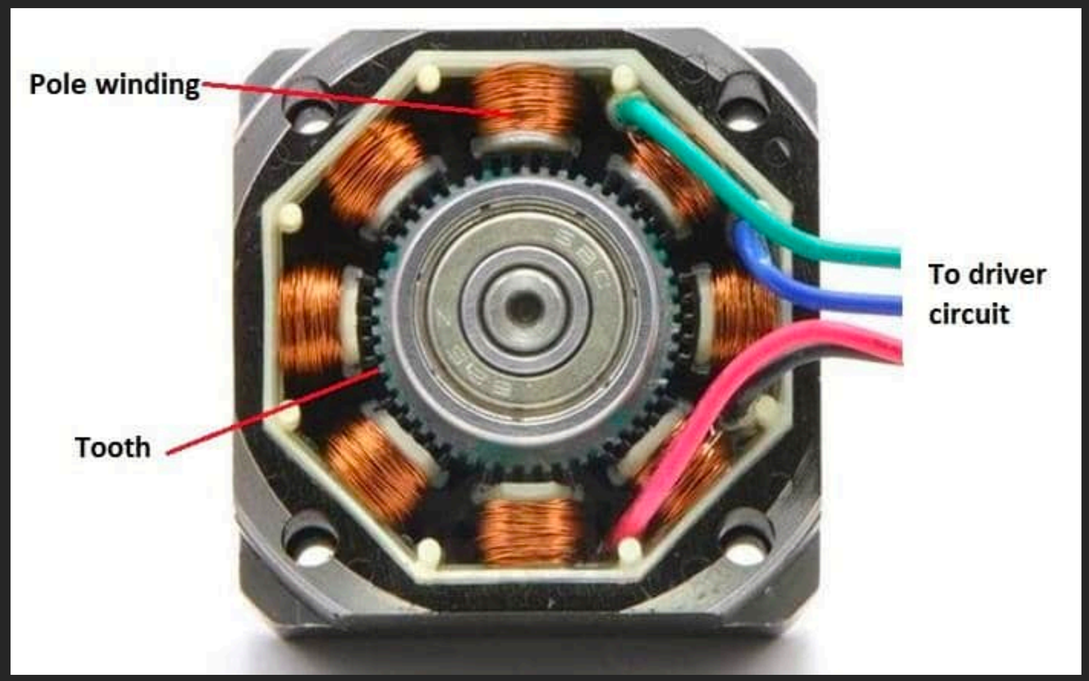


Schrittmotor 28BYJ-48 und ULN-2003 Motorshield



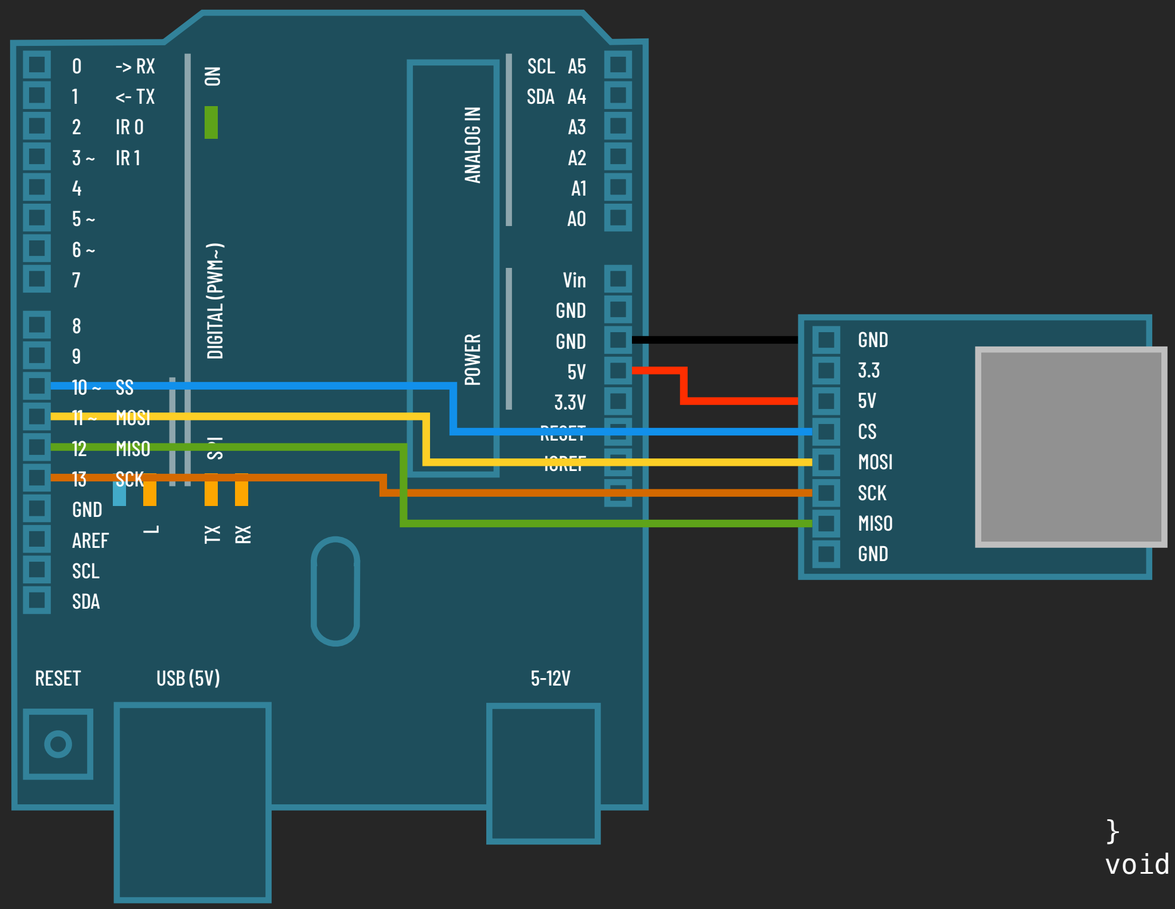

```
#define A 2
#define B 3
#define C 4
#define D 5

#define NUMBER_OF_STEPS_PER_REV 512

void setup(){
    pinMode(A,OUTPUT);
    pinMode(B,OUTPUT);
    pinMode(C,OUTPUT);
    pinMode(D,OUTPUT);
}

void write(int a,int b,int c,int d){
    digitalWrite(A,a);
    digitalWrite(B,b);
    digitalWrite(C,c);
    digitalWrite(D,d);
}

void onestep(){
    write(1,0,0,0);
    delay(5);
    write(1,1,0,0);
    delay(5);
    write(0,1,0,0);
    delay(5);
    write(0,1,1,0);
    delay(5);
    write(0,0,1,0);
    delay(5);
    write(0,0,1,1);
    delay(5);
    write(0,0,0,1);
    delay(5);
    write(1,0,0,1);
    delay(5);
}

void loop(){
    int i;
    i=0;
    while(i<NUMBER_OF_STEPS_PER_REV){
        onestep();
        i++;
    }
}
```





Innenleben eines Schrittmotors

SD Card Module




```
#include <SPI.h>

#include <SD.h>
File myFile;

void setup() {
	Serial.begin(9600);
	while (!Serial) {;}
	Serial.print("Initializing SD card...");
	if (!SD.begin(10)) {
		Serial.println("initialization failed!");
		while (1);
	}
	Serial.println("initialization done.");

	// open the file. note that only one file can be open at a time,
	// so you have to close this one before opening another.
	myFile = SD.open("test.txt", FILE_WRITE);

	// if the file opened okay, write to it:
	if (myFile) {
		Serial.print("Writing to test.txt...");
		myFile.println("This is a test file :)");
		myFile.println("testing 1, 2, 3.");

		for (int i = 0; i < 20; i++) {
			myFile.print(i);
			myFile.print(",");
			myFile.print(j);
			myFile.println();
		}

		// close the file:
		myFile.close();
		Serial.println("done.");
	} else {
		// if the file didn't open, print an error:
		Serial.println("error opening test.txt");
	}
}
void loop() {
	// nothing happens after setup
}
```

Serial Interface

2021 Michael Reichart, copyleft CC BY-NC-SA

Serial IO

Arduino sendet und empfängt über einen seriellen Port Daten. Die Pins sind RX und TX, aber auch der USB Port kann zur seriellen Kommunikation genutzt werden.

Wenn der Arduino Daten am seriellen Port empfängt, wird ein Interrupt ausgelöst. Interrupts sind Ereignisse, auf die der Microcontroller sofort reagiert. Er unterbricht seine aktuellen Aktionen und führt eine für diesen Interrupt definierte Routine aus.

Den seriellen Port starten

```
// Serial communication on pins TX/RX
// uses TTL logic levels (5V or 3.3V)

void setup() {
    // opens serial port,
    // sets data rate to 9600 bps
    Serial.begin(9600);}

void loop() {}
```

Serielle Daten senden

```
void setup() {
    // opens serial port, sets data rate to 9600 bps
    Serial.begin(9600);
}

void loop() {
    // send data
    Serial.print("I send a message");
    Serial.println(incomingByte, DEC);
}
```

Serielle Daten empfangen

```
int incomingByte = 0;   // for incoming serial data

void setup() {
    // opens serial port, sets data rate to 9600 bps
    Serial.begin(9600);
}

void loop() {
    // send data only when you receive data:
    if (Serial.available() > 0) {
        // read the incoming byte:
        incomingByte = Serial.read();

        // say what you got:
        Serial.print("I received: ");
        Serial.println(incomingByte, DEC);
    }
}
```

Ausgaben zusammenfassen

```
float fDeg = 21;
char sUnit[] = "Celsius";

char buffer[40];
sprintf(buffer, "%d degree %s", fDeg, sUnit);
Serial.println(buffer);
```

OLED Bildschirm
128 x 64 Pixel


```
#include <U8g2lib.h>
U8G2_SSD1306_128X64_NONAME_1_HW_I2C myOled(U8G2_R0);

int potPin = A3, potValue = 0;
int screenW = 128, screenH = 64;
```