

Aufgabe 2

```
// Musterlösung für Aufgabe 3
// Pins werden sauber im Header definiert
// Eingangswert wird auf einen Bereich für die LED
umgeschrieben
// Mit Serial.print und Serial.println geben wir uns für
das Debugging die Werte am Seriellen Monitor aus
```

```
int ledPin = 6;
int ldrPin = A0;

void setup() {
    Serial.begin(9600);
    pinMode(ldrPin, INPUT);
    pinMode(ledPin, OUTPUT);

}

void loop() {
    int val = analogRead(ldrPin);
    int mapped = map(val, 0, 1024, 0, 255);
    Serial.print("ldrPin= ");
    Serial.print(val);
    Serial.print(" / mapped= ");
    Serial.println(mapped);
    analogWrite(ledPin, mapped);

}
```

Aufgabe 2

- Was passiert wenn ich 2 Photowiderstände einbaue und 2 LEDs parallel steuern will?

Aufgabe 2

```
int ledPin = 6;
int ldrPin = A0;

void setup() {
    Serial.begin(9600);
    pinMode(ldrPin, INPUT);
    pinMode(ledPin, OUTPUT);
}

void loop() {
    int val = analogRead(ldrPin);
    int mapped = map(val, 0, 1024, 0, 255);
    Serial.print("ldrPin= ");
    Serial.print(val);
    Serial.print(" / mapped= ");
    Serial.println(mapped);
    analogWrite(ledPin, mapped);
}

void loop() {
    fadeOurLED(A0, 6);
    fadeOurLED(A1, 5); // eine zweite LDR + LED
}

void fadeOurLED(int ldrPin, int ledPin) {
    int val = analogRead(ldrPin);
    int mapped = map(val, 0, 1024, 0, 255);
    Serial.print("ldrPin= ");
    Serial.print(val);
    Serial.print(" / mapped= ");
    Serial.println(mapped);
    analogWrite(ledPin, mapped);
}
```

Aufgabe 2

```
int[] ldrs = {A0, A1};  
int[] leds = {6, 5};  
void loop() {  
    for(int i = 0; i<10; i++) {  
        fadeOurLED(ldrs[i], leds[i]);  
    }  
}  
  
void fadeOurLED(int ldrPin, int ledPin) {  
    int val = analogRead(ldrPin);  
    int mapped = map(val, 0, 1024, 0, 255);  
    Serial.print("ldrPin= ");  
    Serial.print(val);  
    Serial.print(" / mapped= ");  
    Serial.println(mapped);  
    analogWrite(ledPin, mapped);  
}
```

Aufgabe 2

- Wie würde man vorgehen um "gleichzeitig" beide Inputs abzugreifen bzw. was verhindert momentan diesen Moment der Synchronisierung?

Aufgabe 2

Recap

Arduino Bauteile 2

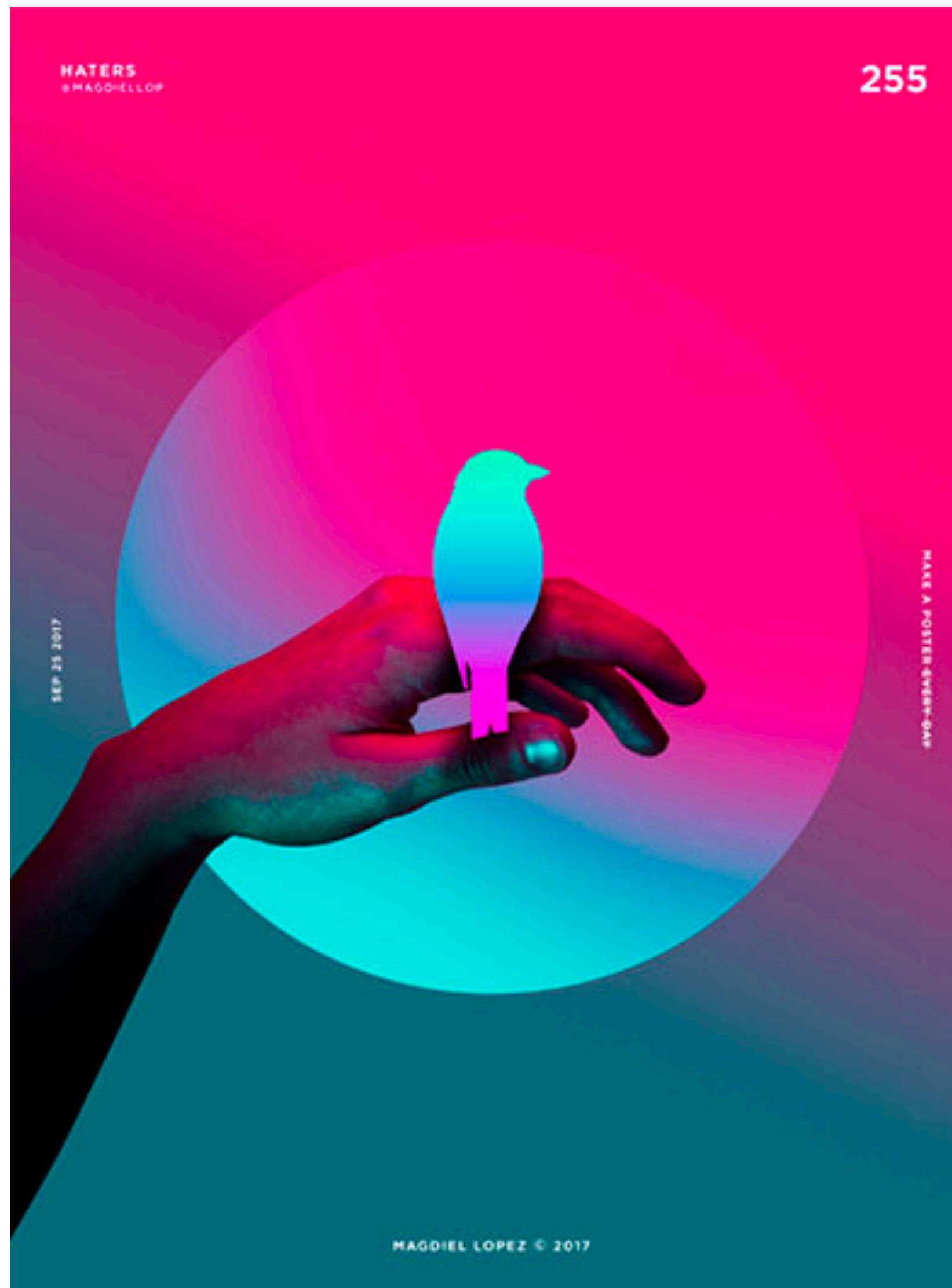
Shields

Bibliotheken

Multitasking

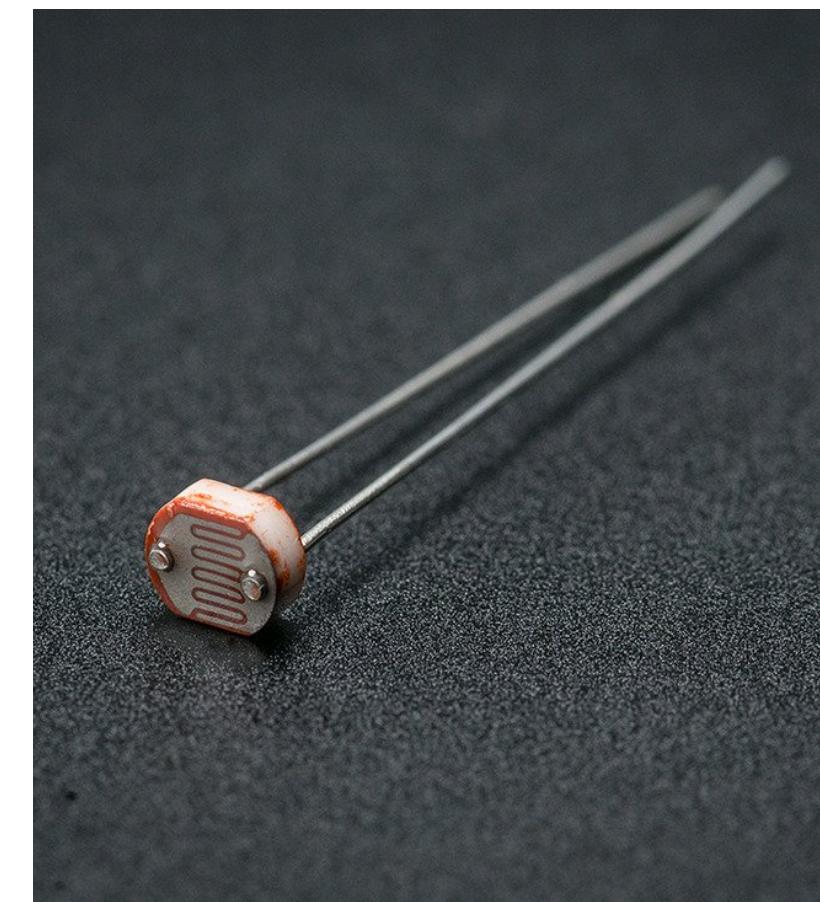
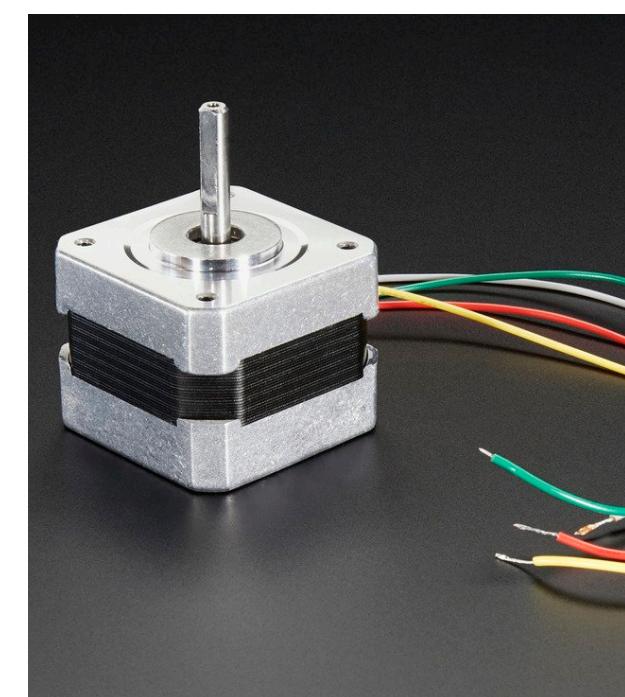
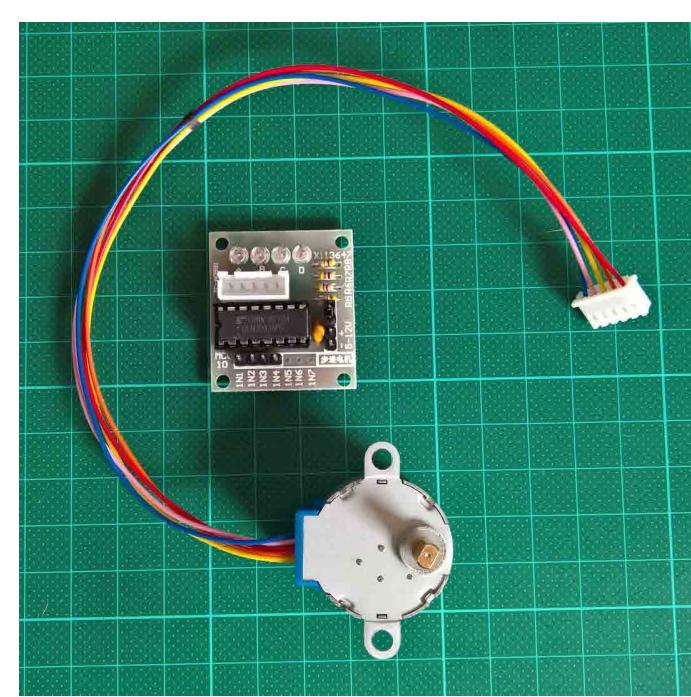
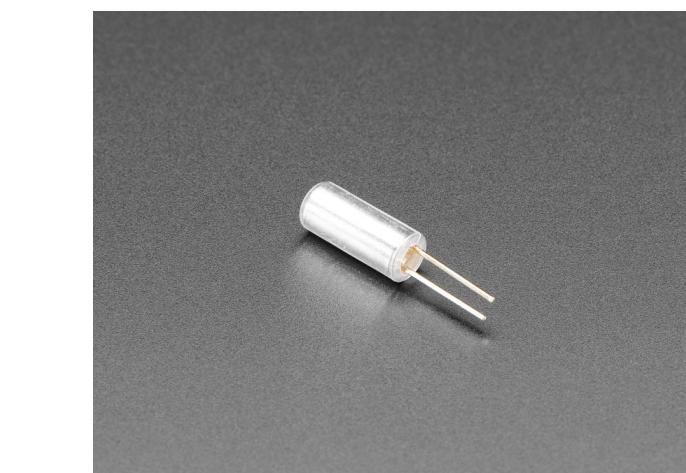
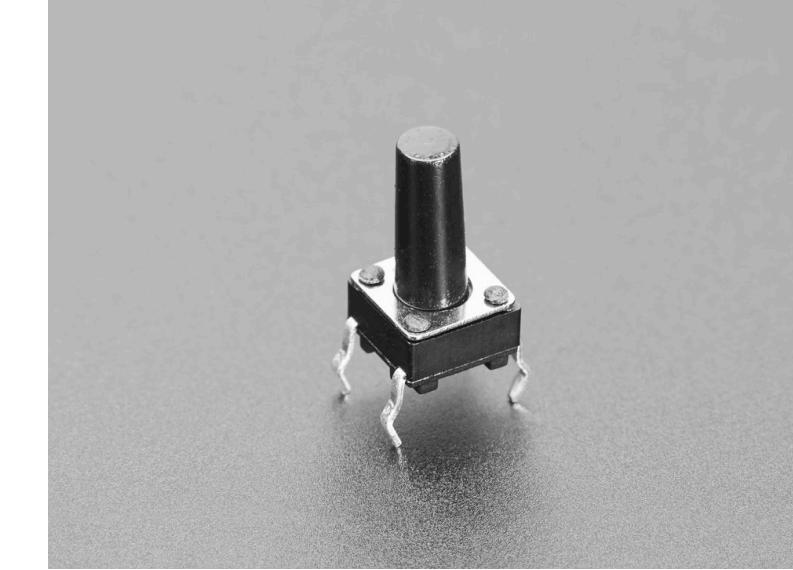
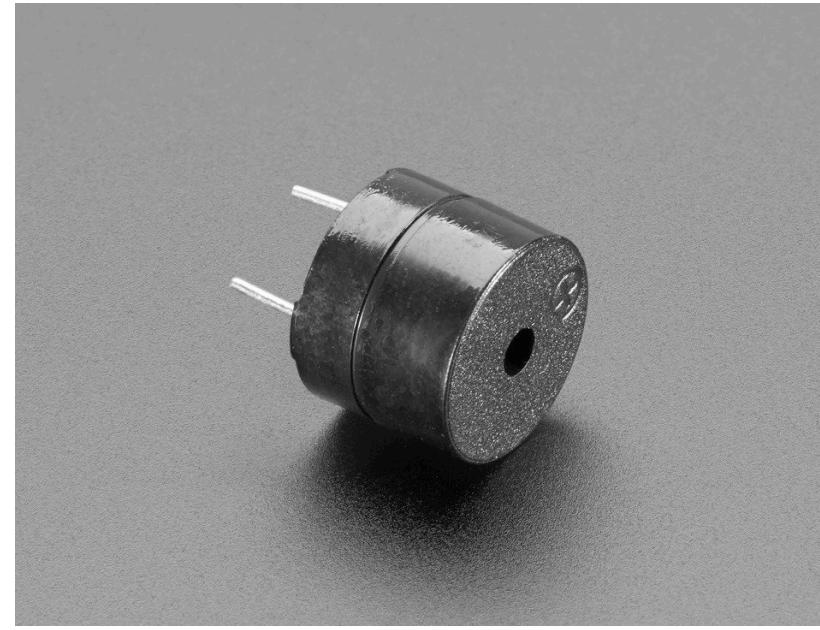
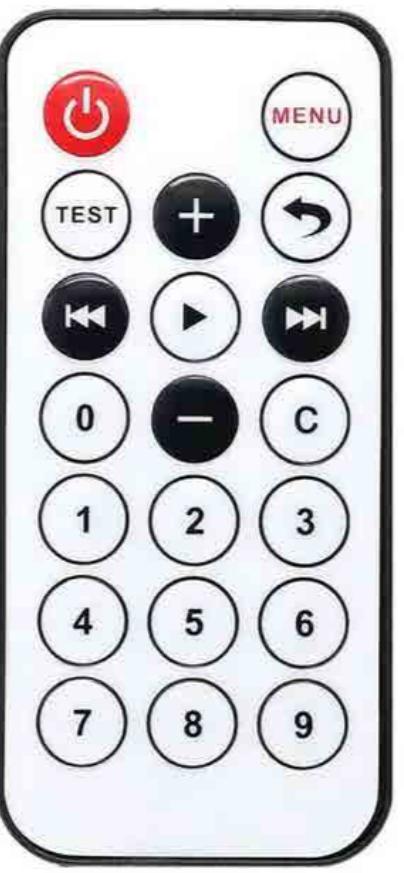
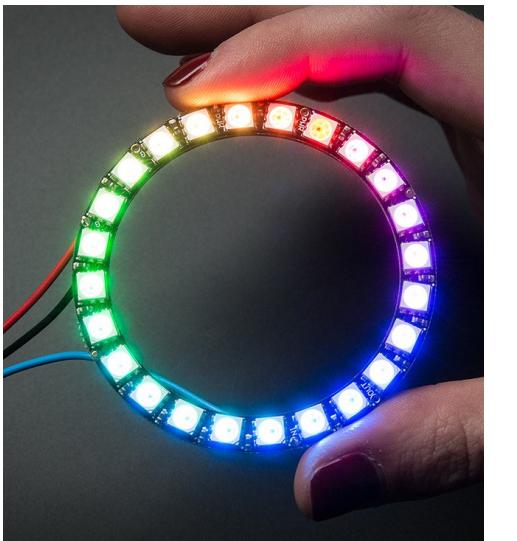
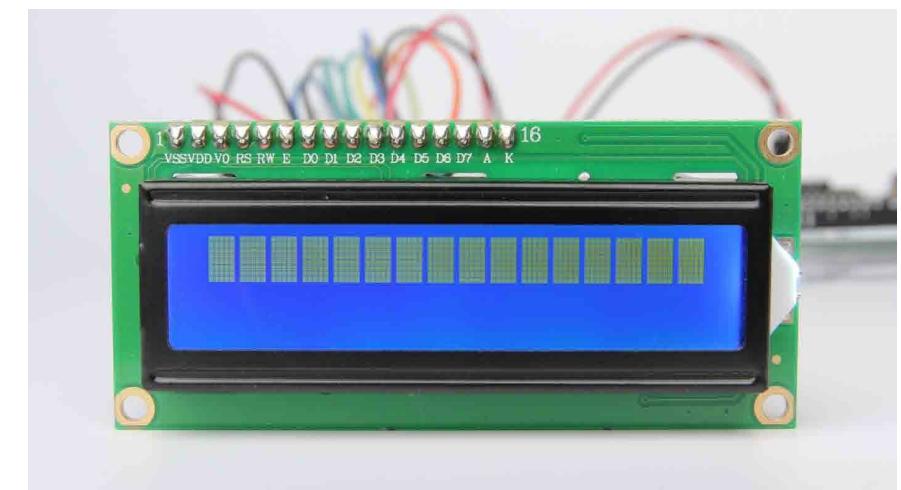
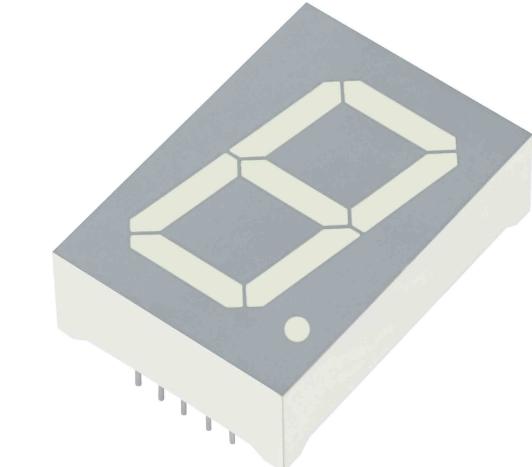
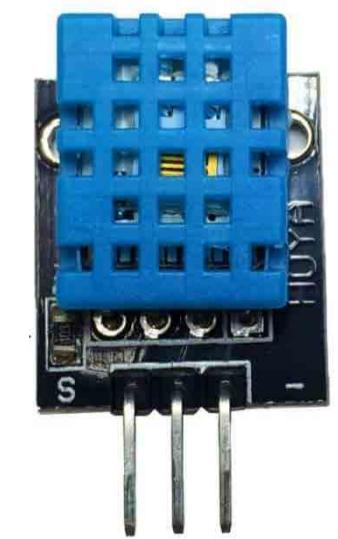
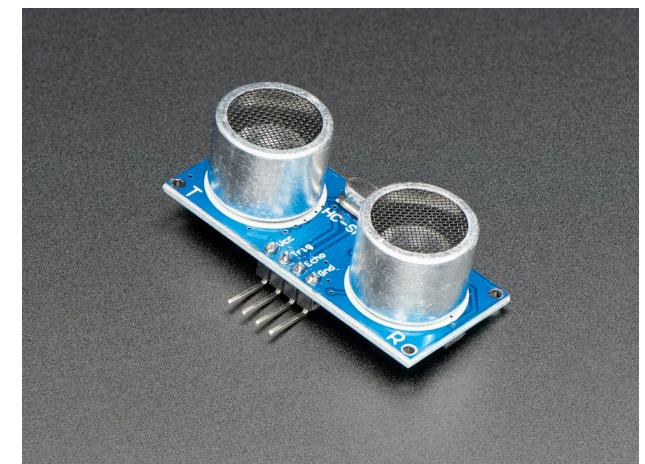
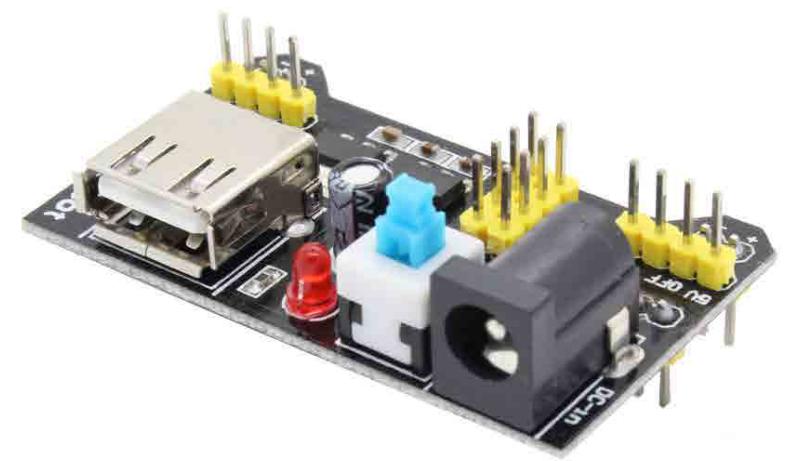
Workshop Servos

Recap

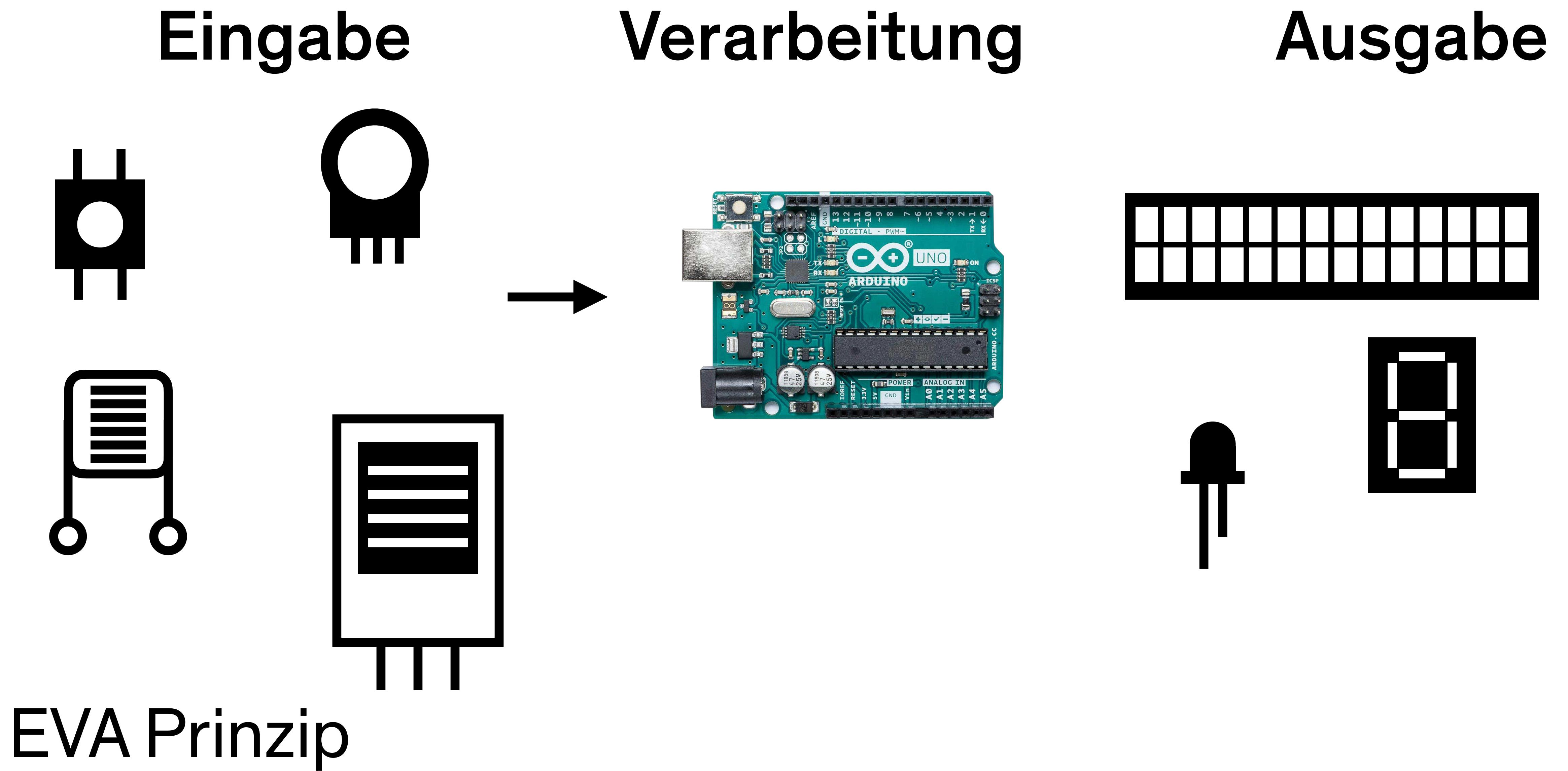


Warum Physical Computing?

Physical Computing



Arduino Bauteile



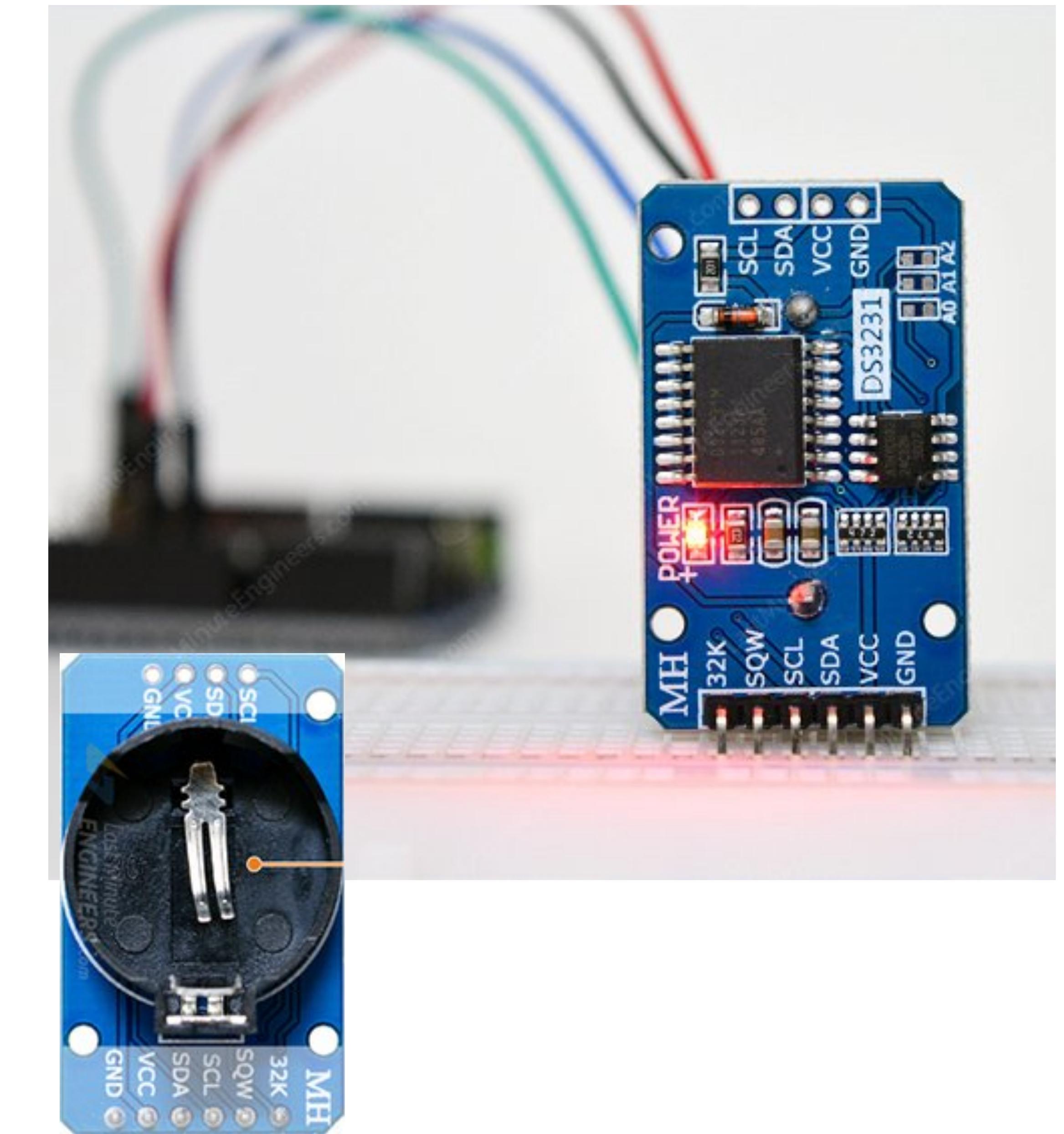
setup()	HIGH	INPUT	INPUT_PULLP	true
loop()	LOW	OUTPUT	LED_BULTIN	false
bool	byte	integer	float	
char	short	long	double	
Array	String		unsigned int	void
Objekt	unsigned char		unsigned long	
pinMode()	digitalRead()		analogRead()	
	digitalWrite()		analogWrite()	
Serial.begin()		Serial.print()	Serial.println()	
delay()	millis()	delayMicroseconds()	micros()	
map()				Kommentare → // oder /* */

Programmier Grundlagen

Arduino Bauteile 2

Real Time Clock Module

DS1307 DS3231 →



Echtzeituhr

Konsistente und genaue Zeitmessung

Unabhängig vom Programmstart oder Reset

Arduino Bauteile 2 <https://funduino.de/anleitung-dht11-dht22>

Real Time Clock Module

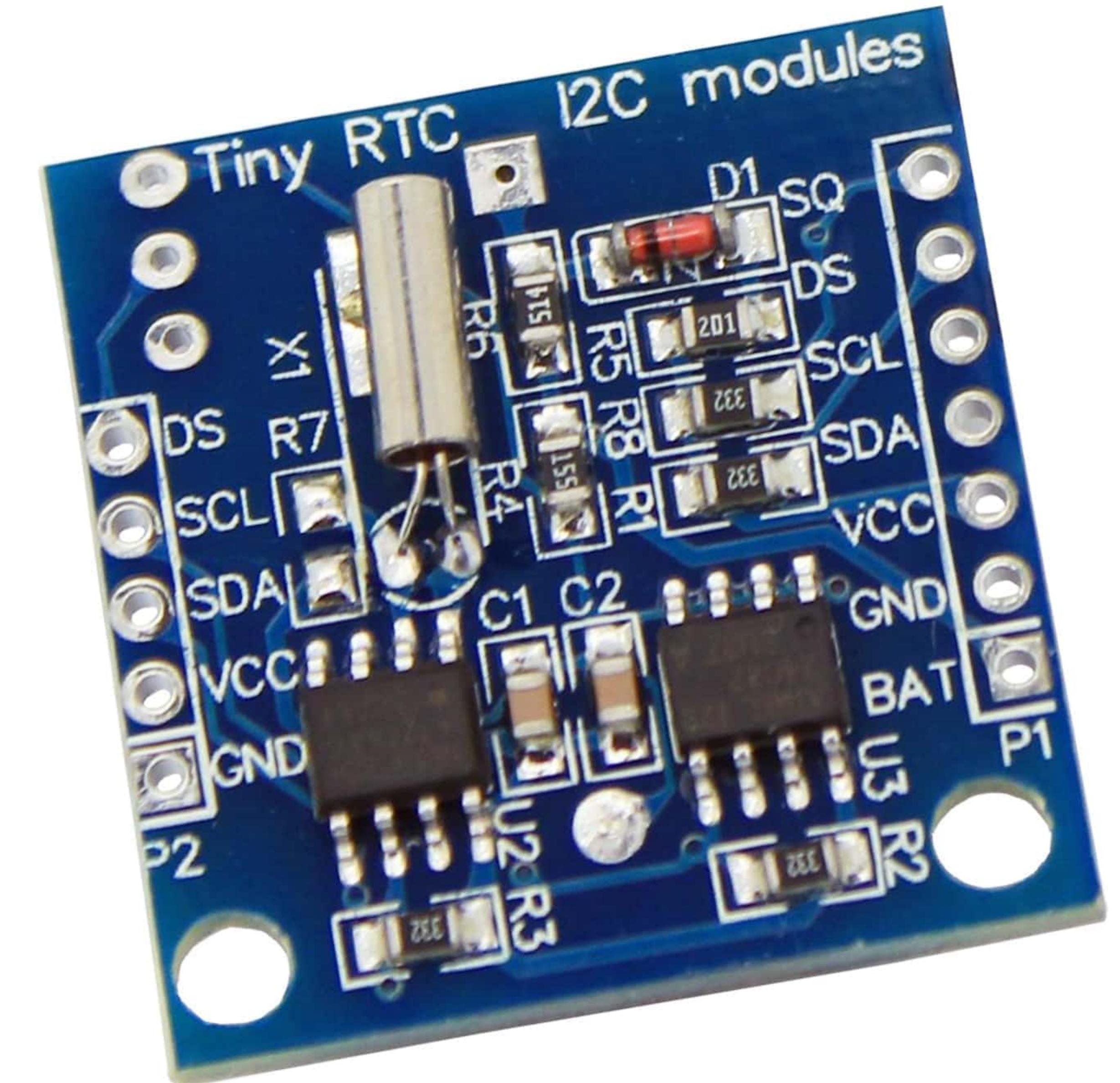
DS1307 →

DS3231

Echtzeituhr

Konsistente und genaue Zeitmessung

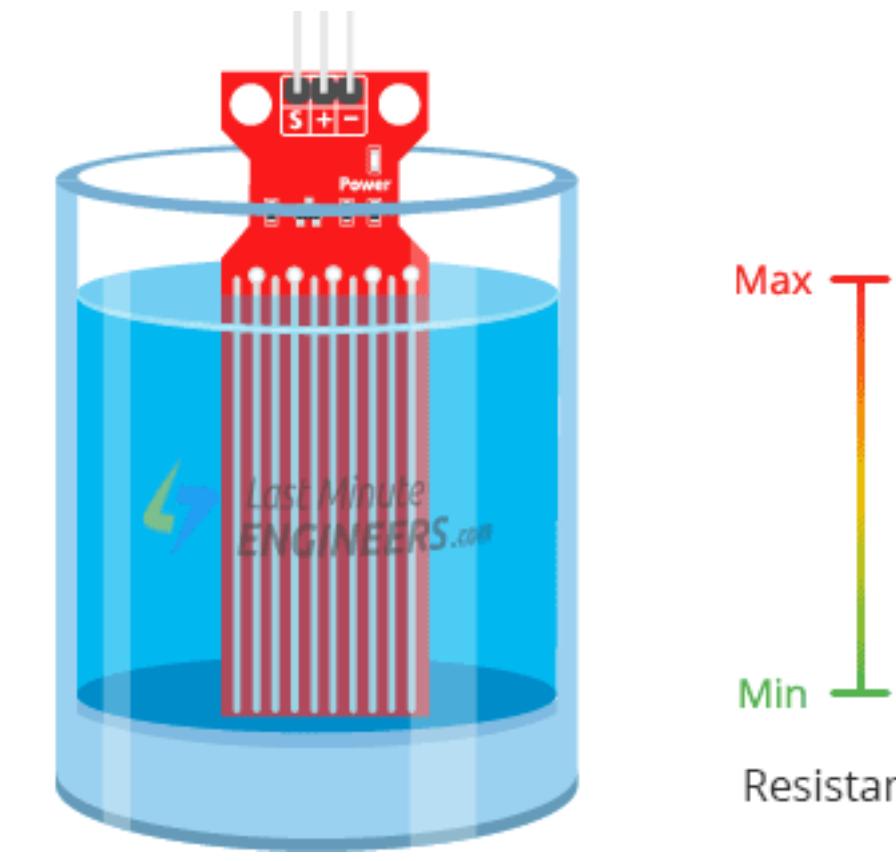
Unabhängig vom Programmstart oder Reset



Arduino Bauteile 2

<https://funduino.de/anleitung-dht11-dht22>

Wasserstandsensor



Messen des Füllstands von Flüssigkeiten

Ausgabe des Füllstandes über einen Analogwert

Arduino Bauteile 2 <https://www.aeq-web.com/arduino-water-tank-level-sensor/>

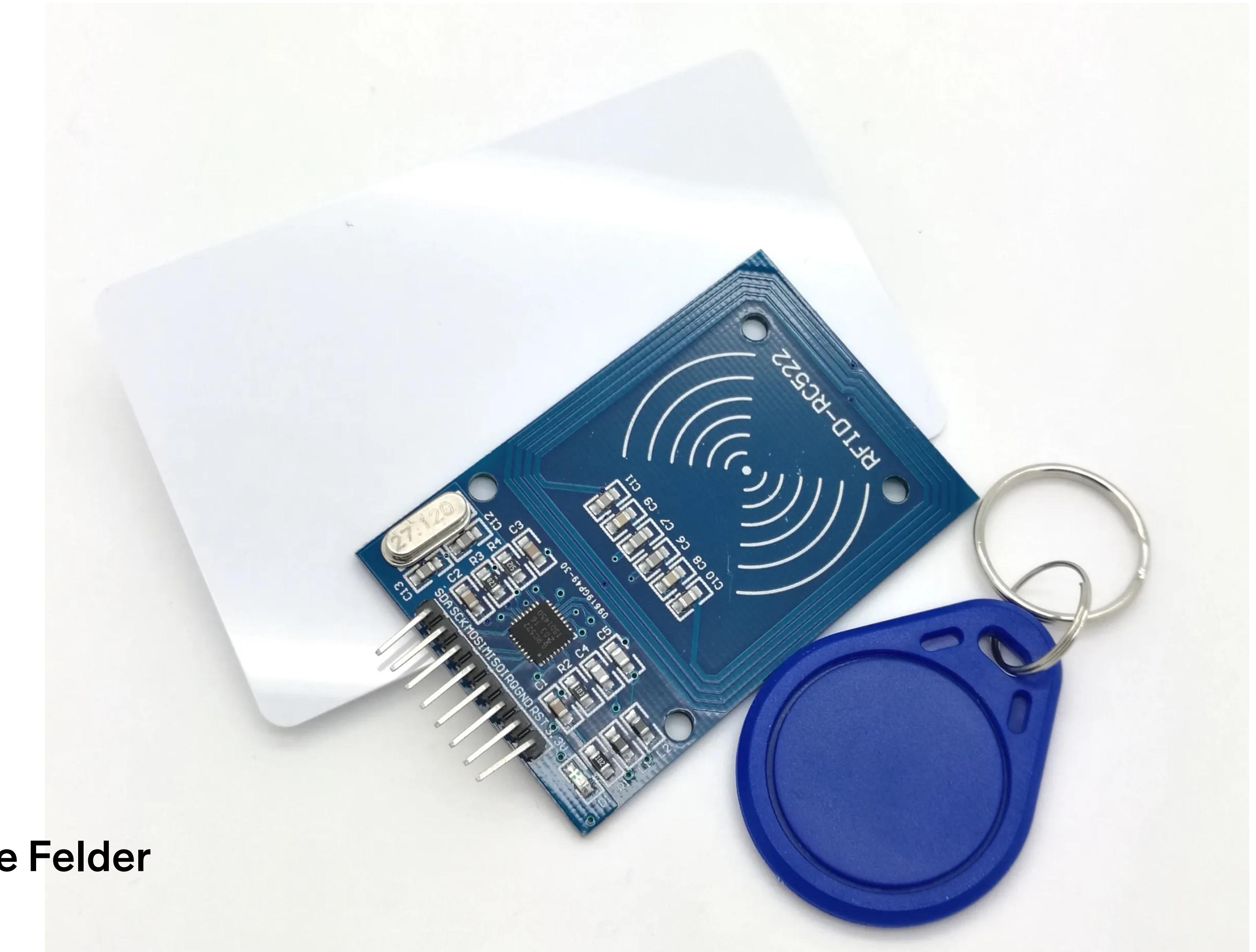
RFID Reader

RFID Reader + RFID Karten u. RFID Chip

Beschreibbar mit eindeutigen IDs

Kommunikation über elektromagnetische Felder

Einsatz z.B. Erkennungssysteme



Arduino Bauteile 2

<https://funduino.de/nr-18-rfid-kit>

PIR Infrarotsensor

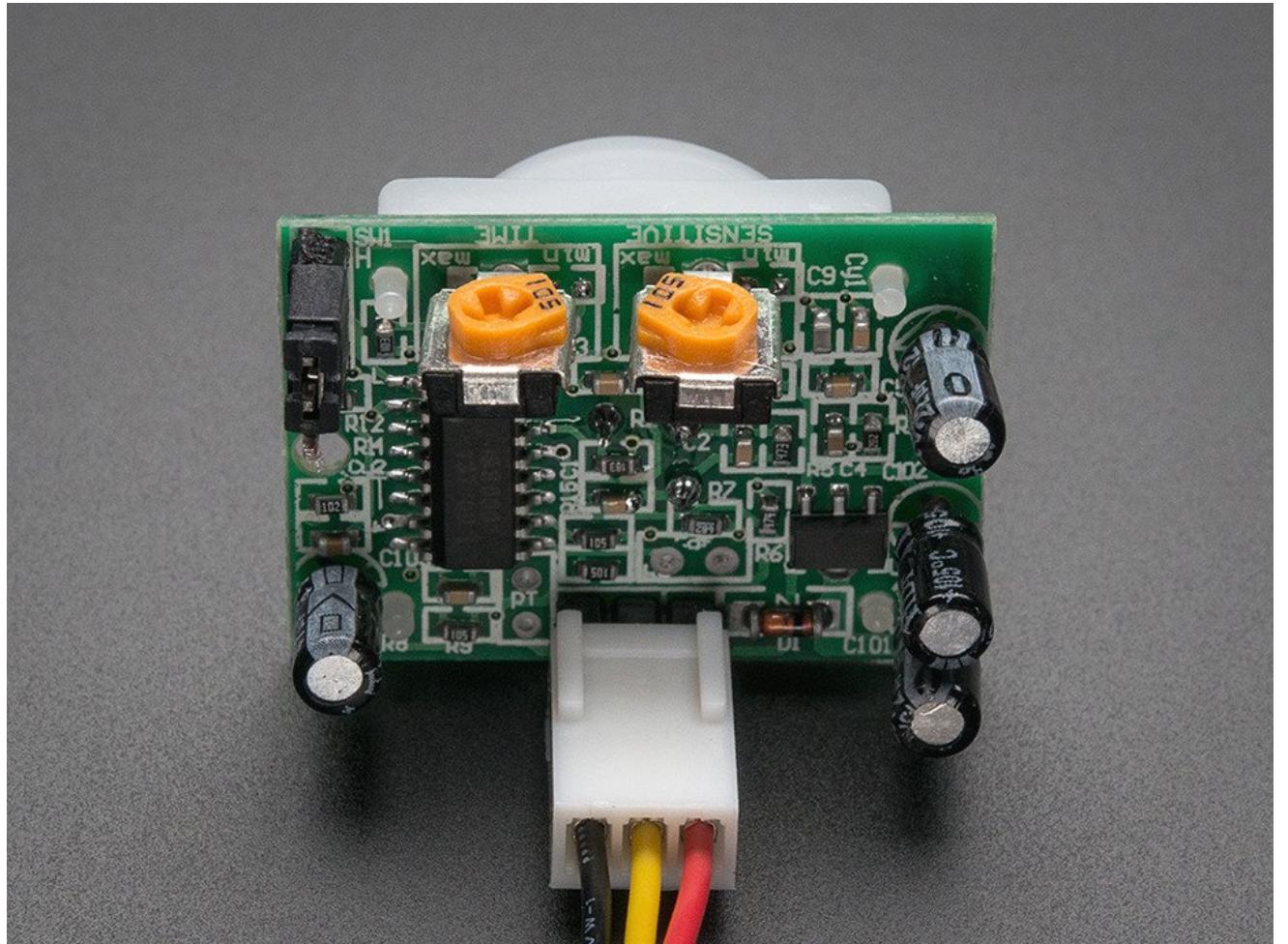
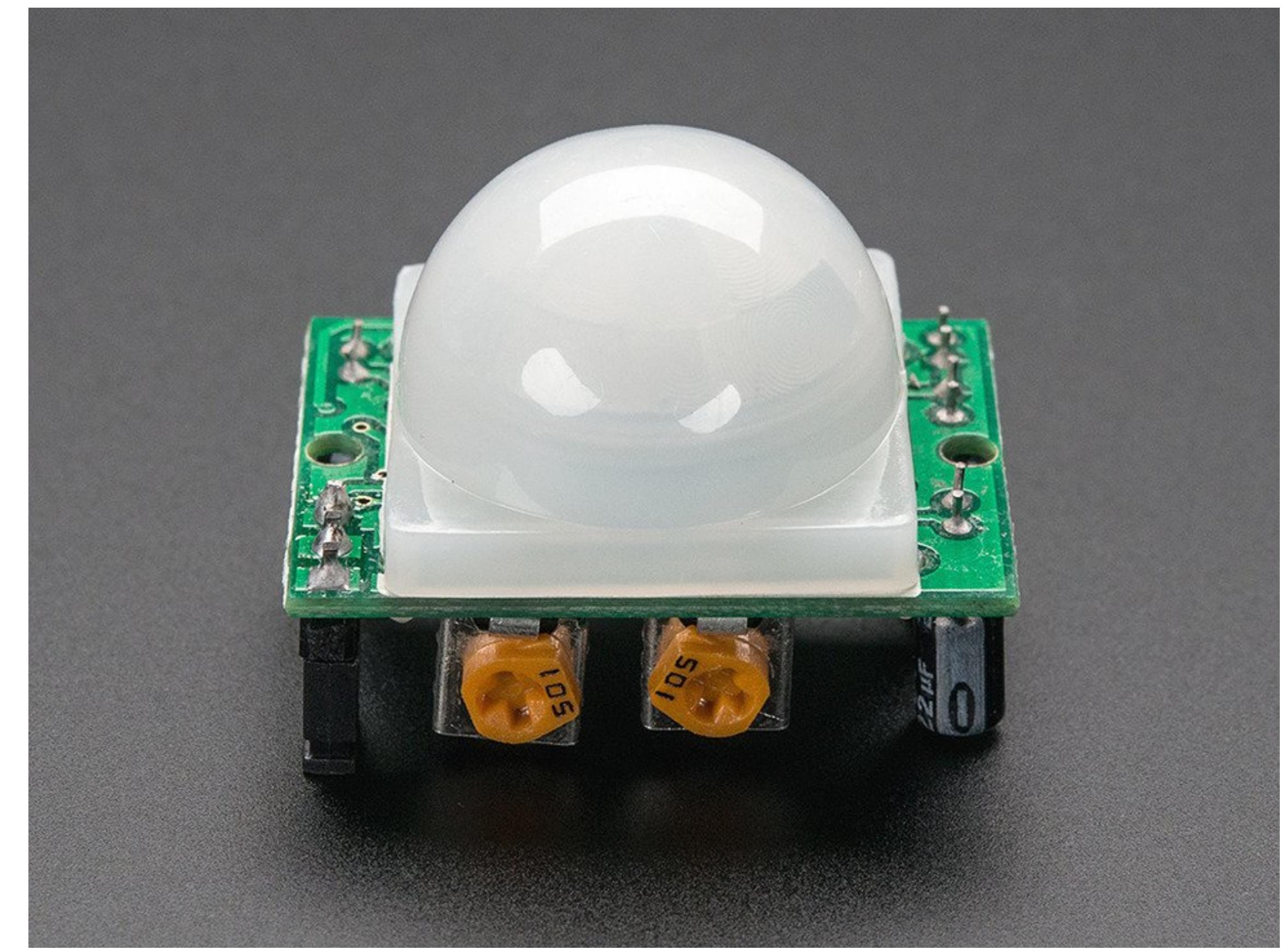
Bewegungsmelder

Trim-Potentiometer um die Reichweite und das Signal einzustellen

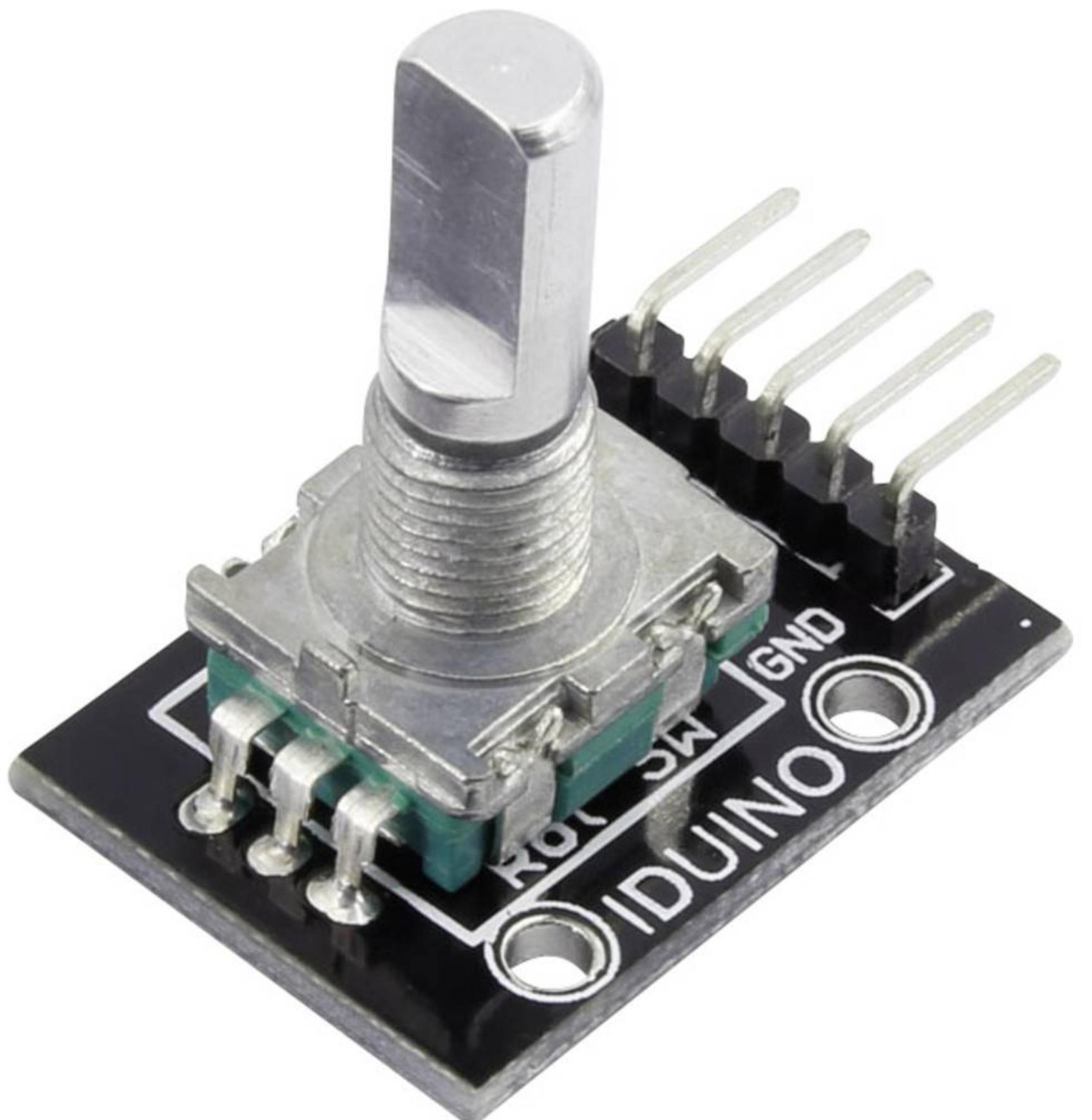
Erkennt warme Objekte im Erkennungsbereich

Arduino Bauteile 2

<https://funduino.de/nr-8-bewegungsmelder>



Rotary Encoder Drehwinkelgeber



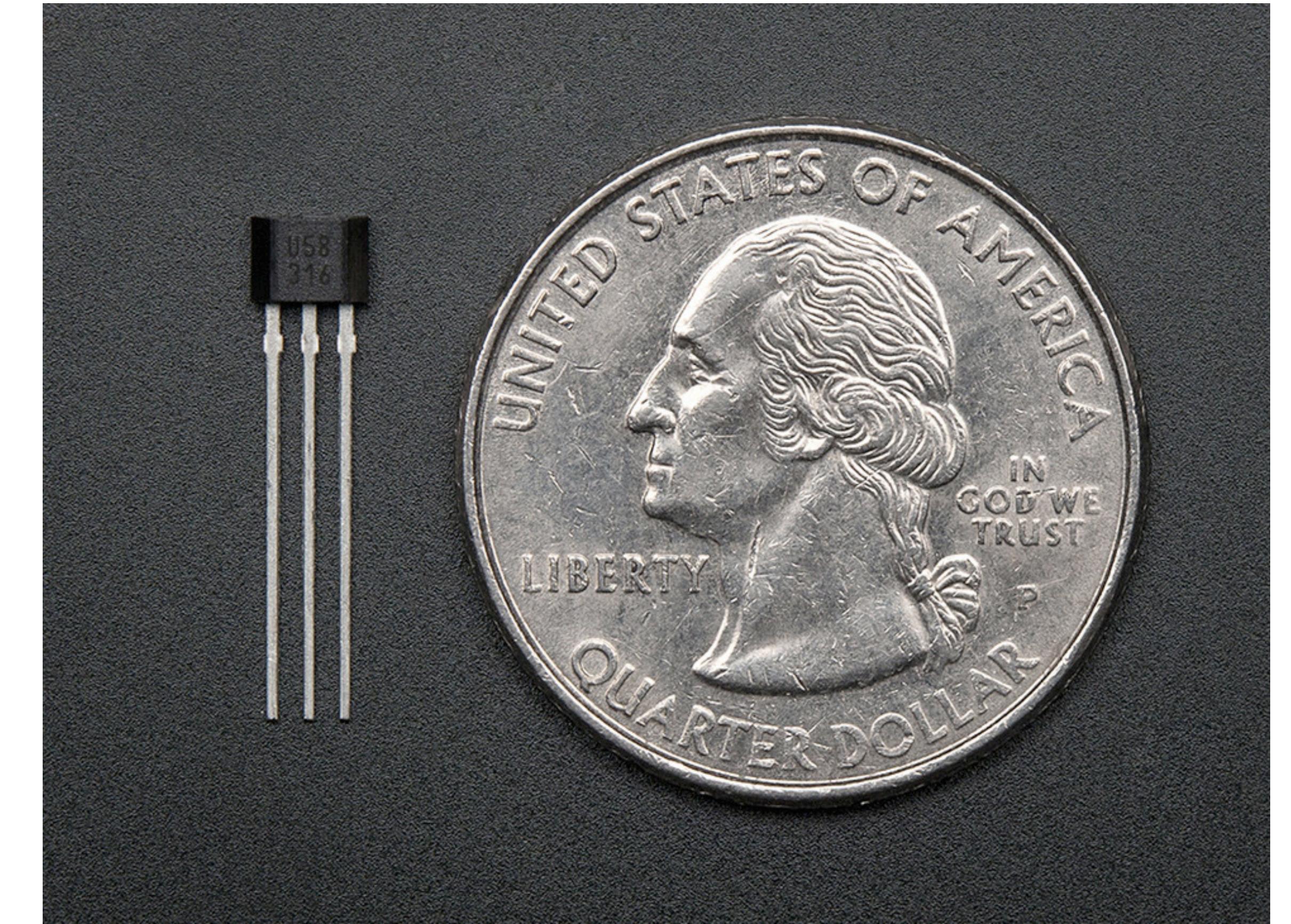
Funktioniert ähnlich wie ein Potentiometer

Angabe der Drehrichtung und der dabei zurückgelegten Schritte

Kein Endanschlag. 360° Rotation möglich

Arduino Bauteile 2 <https://funduino.de/nr-32-der-rotary-encoder-ky-040>

Hall-Effekt Sensor



Erkennt Magnetfelder

Berührungsloser Schalter

Gibt ein Signal aus wenn ein Magnetfeld erkannt wird

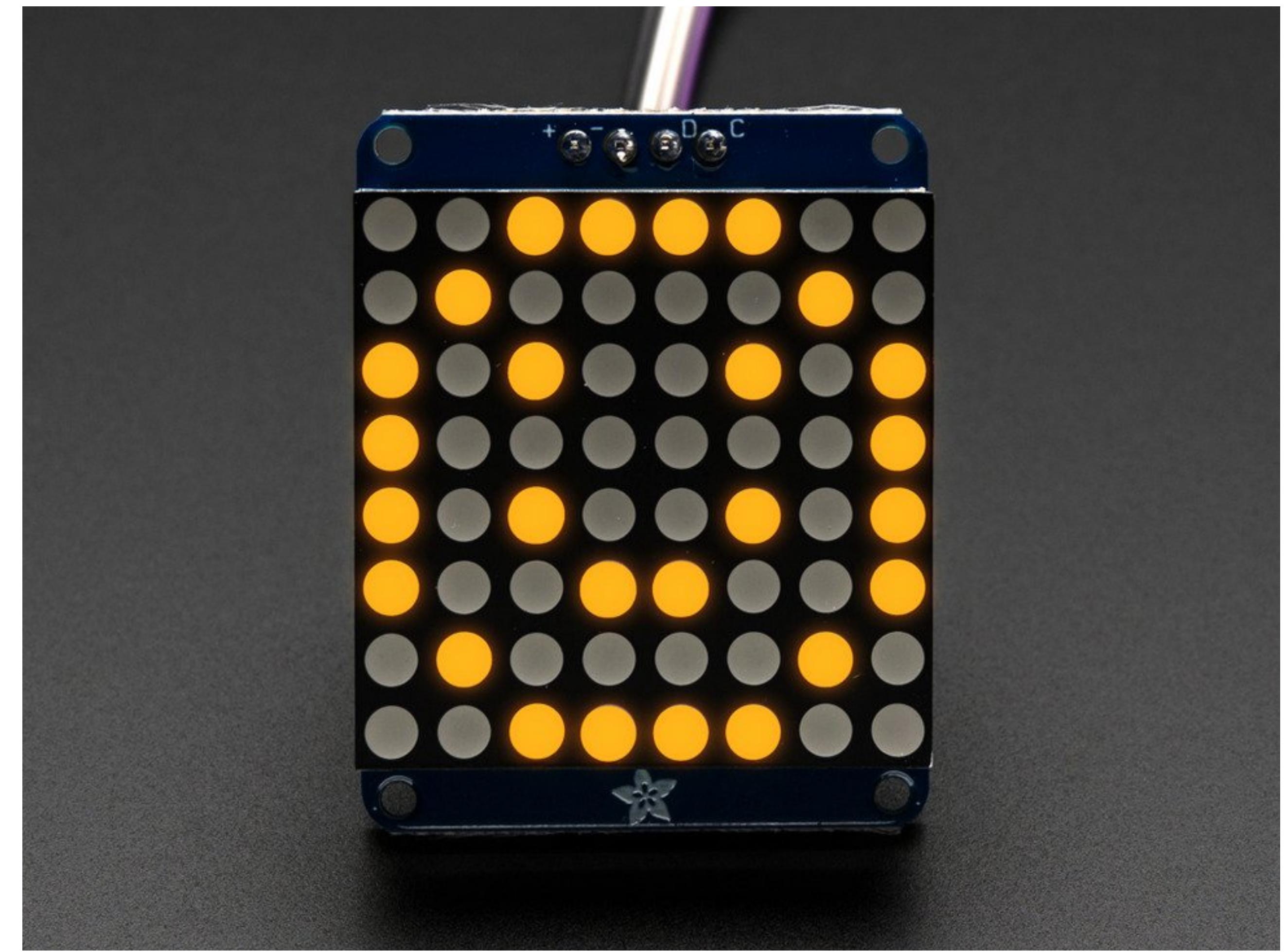
Arduino Bauteile 2 http://bollwerk-essen.bplaced.net/?hall_sensor

LED Matrix

8x8 Matrix

Benötigt drei Pins des Arduino (SPI Schnittstelle)

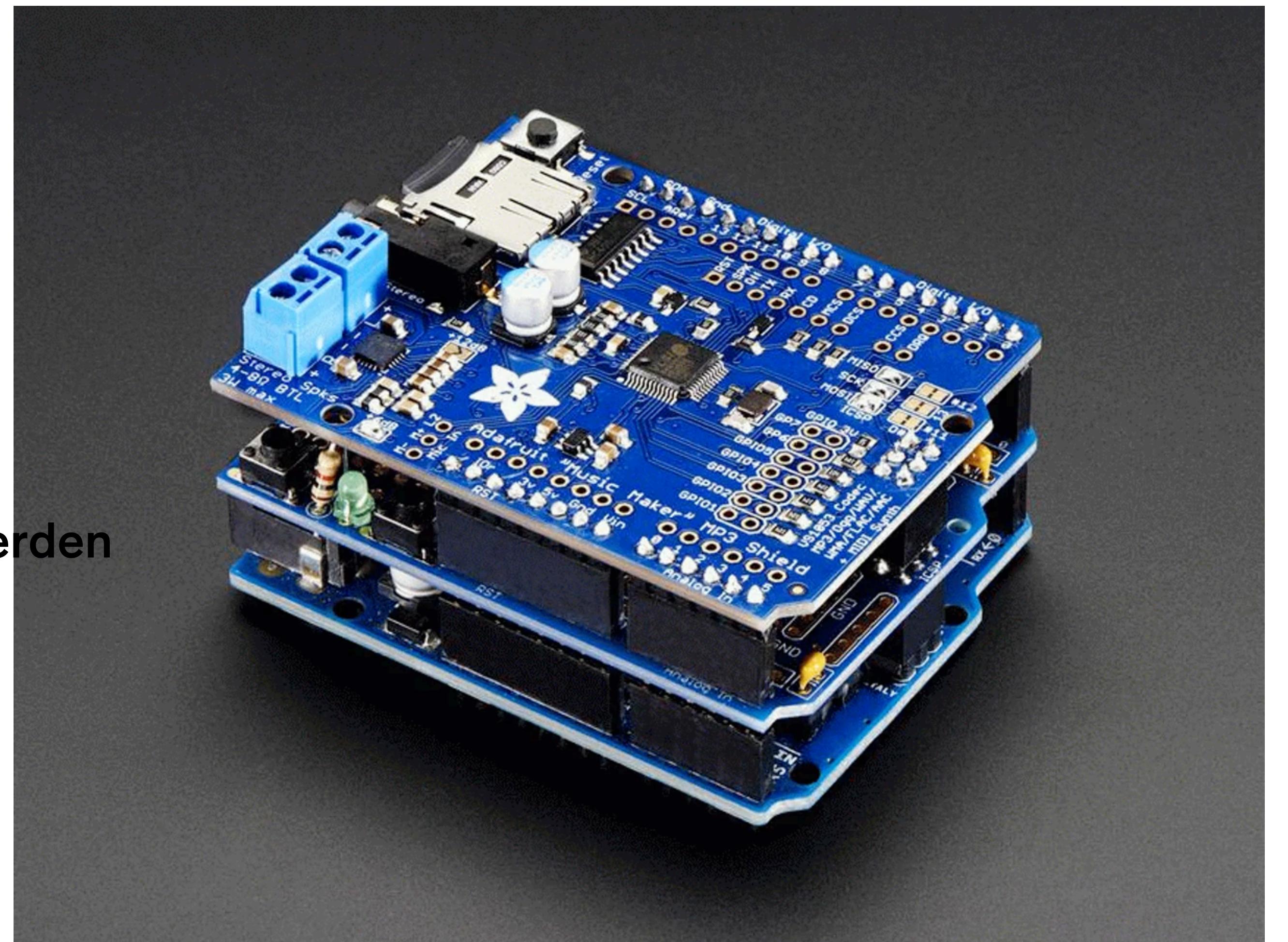
Anzeigen beliebiger Muster, Buchstaben und
Informationen



Arduino Bauteile 2

<https://howtomechatronics.com/tutorials/arduino/8x8-led-matrix-max7219-tutorial-with-bluetooth/>

Shields



Platinen die auf das Arduino Board gesteckt werden können

Erweitern die Funktionalität des Arduinos

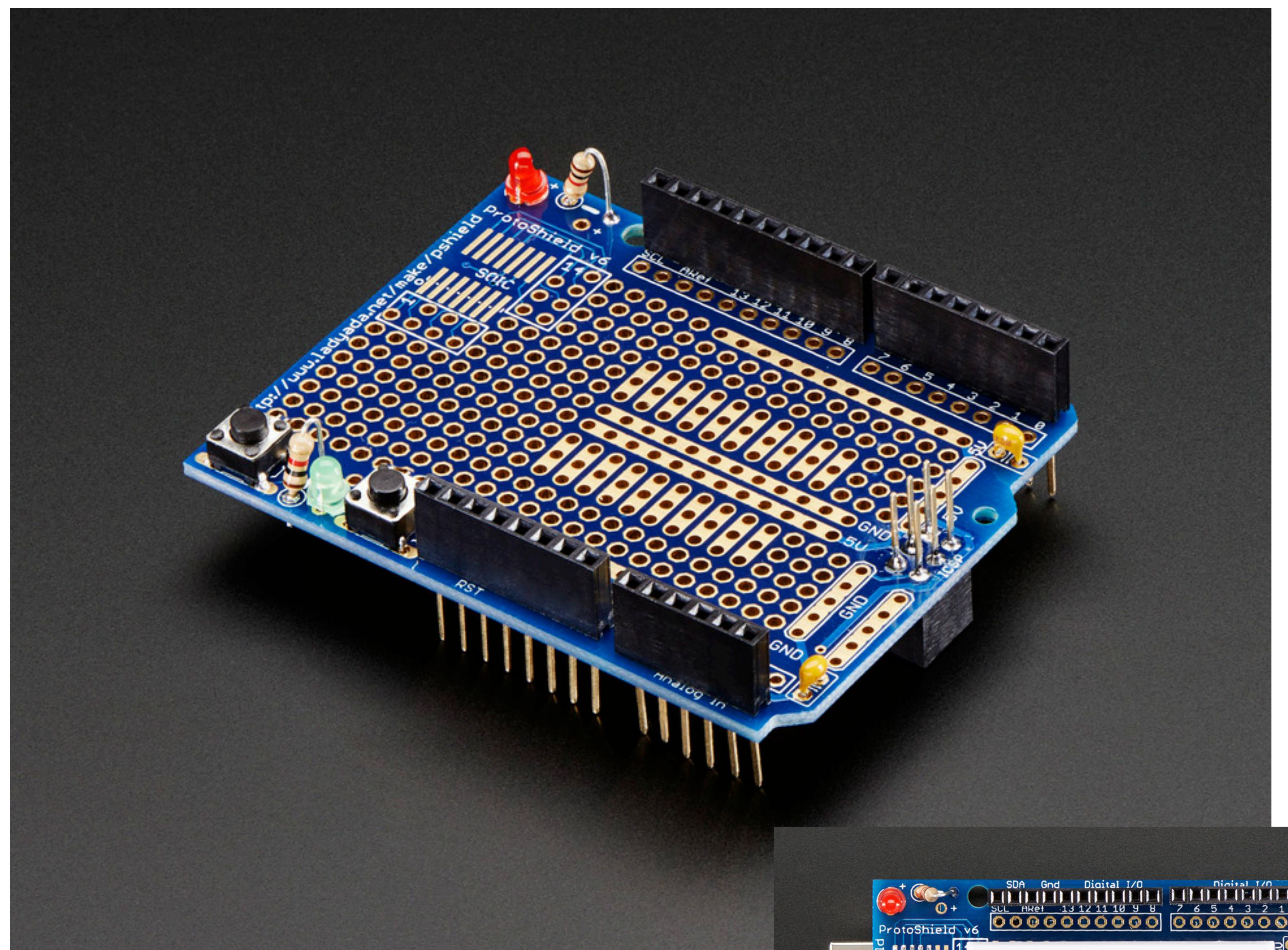
Normiertes System

Große Varianz an Shields vorhanden

Achtung: Auf das Pinout des Shields achten um Kompatibilität zu gewährleisten

Shields

Prototyping Shield



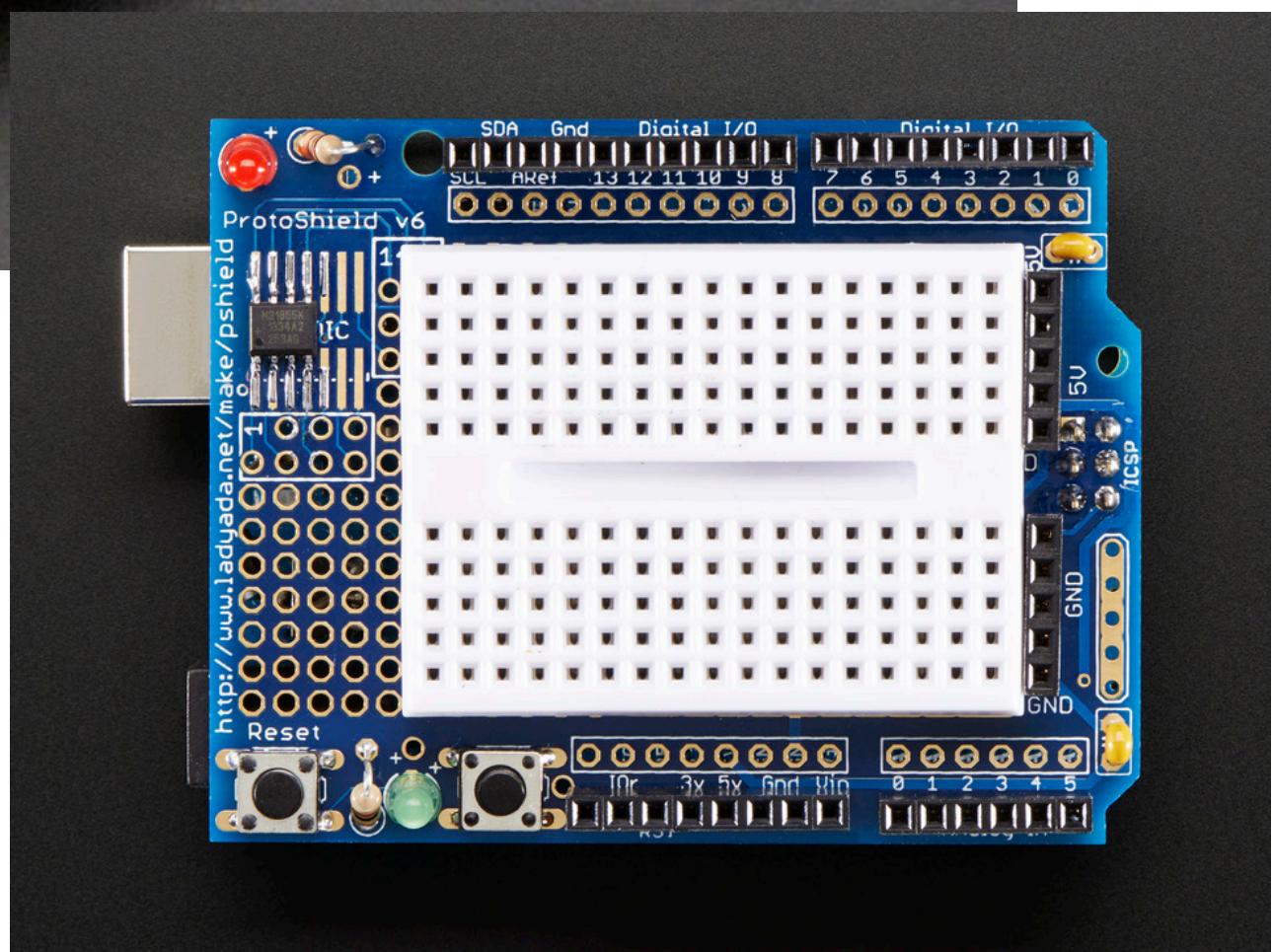
Führt die Arduino Pins weiter

Führ den Reset Pin weiter

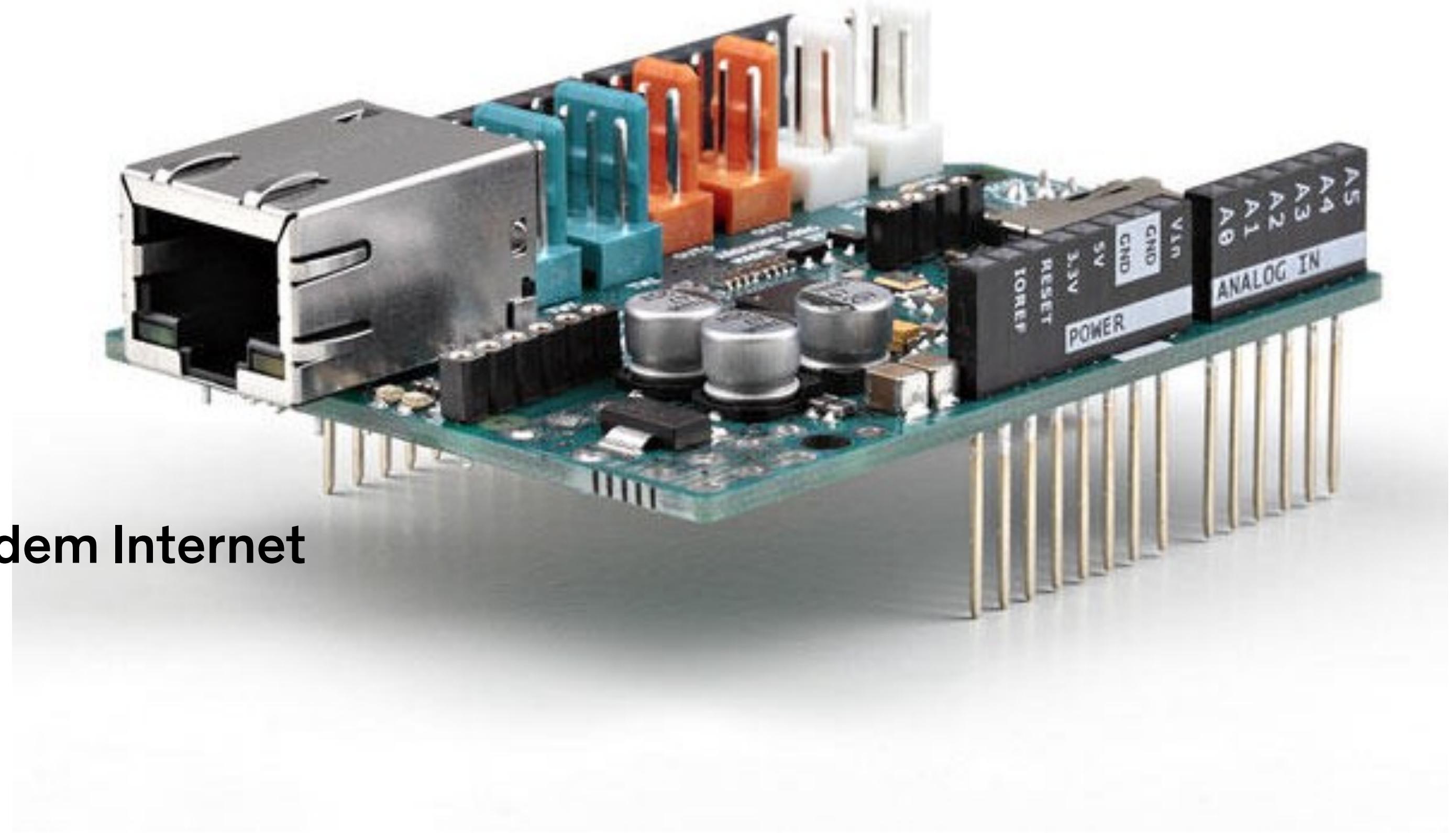
In die Mitte kann ein Breadboard "geklebt" werden

Auf dem Shield kann direkt gelötet werden

Shields



Ethernet Shield



RJ-45 Anschlußbuchse

Erlaubt die Verbindung des Arduino mit dem Internet

Micro SD Kartenslot

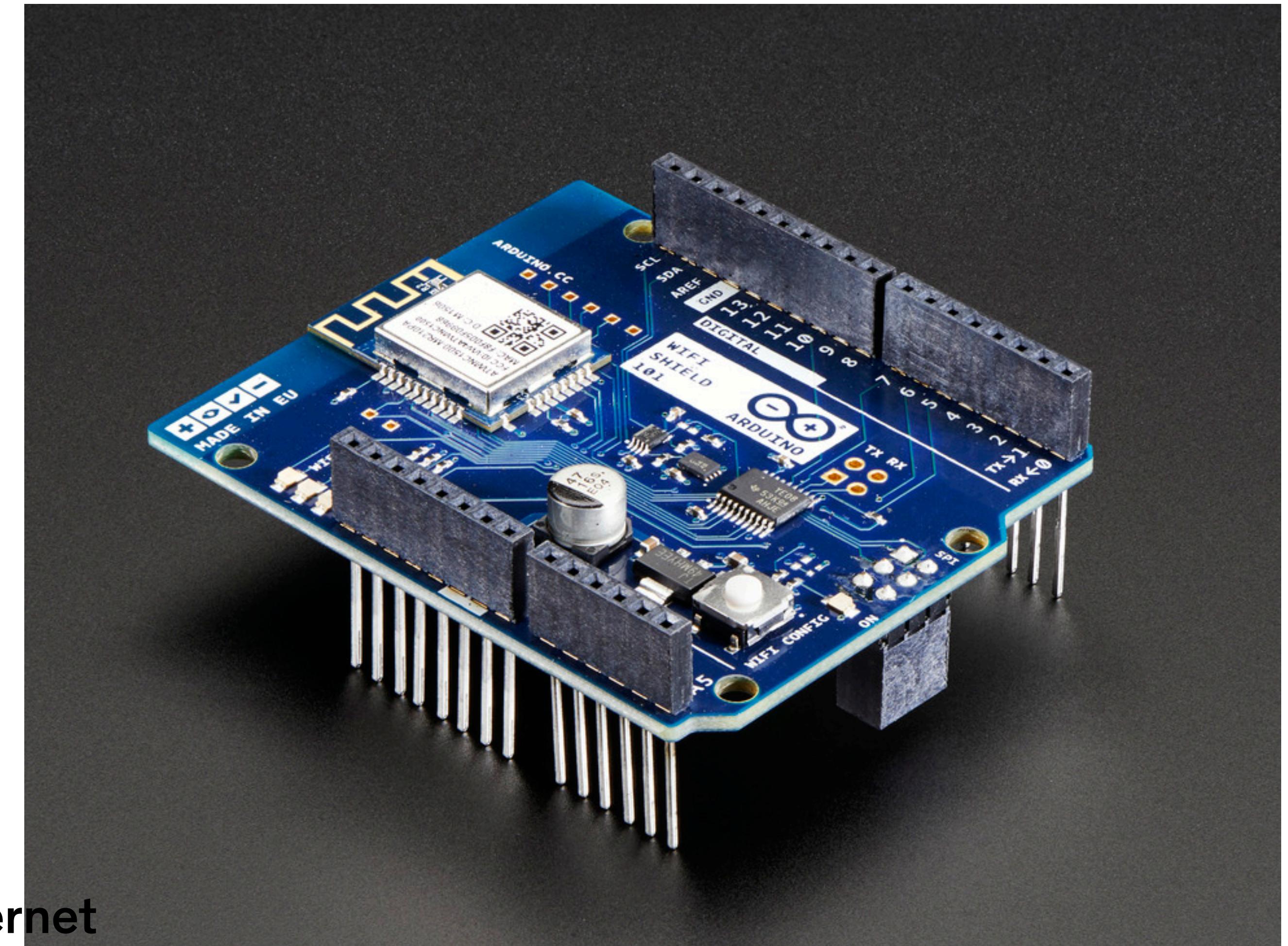
Führt die Arduino Pins weiter

Kein POE!

8 gleichzeitige Socket Verbindungen

Shields

WiFi Shield



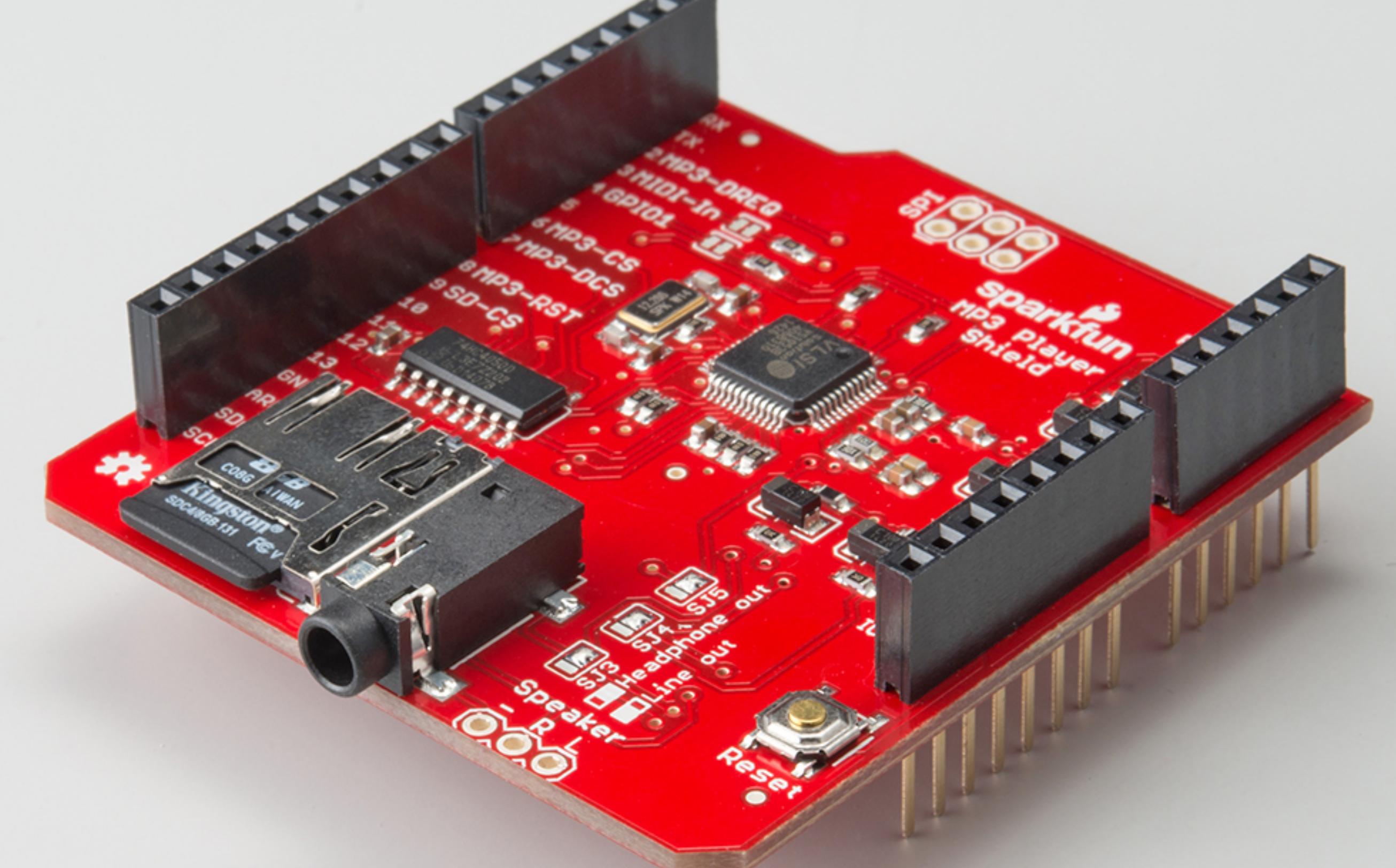
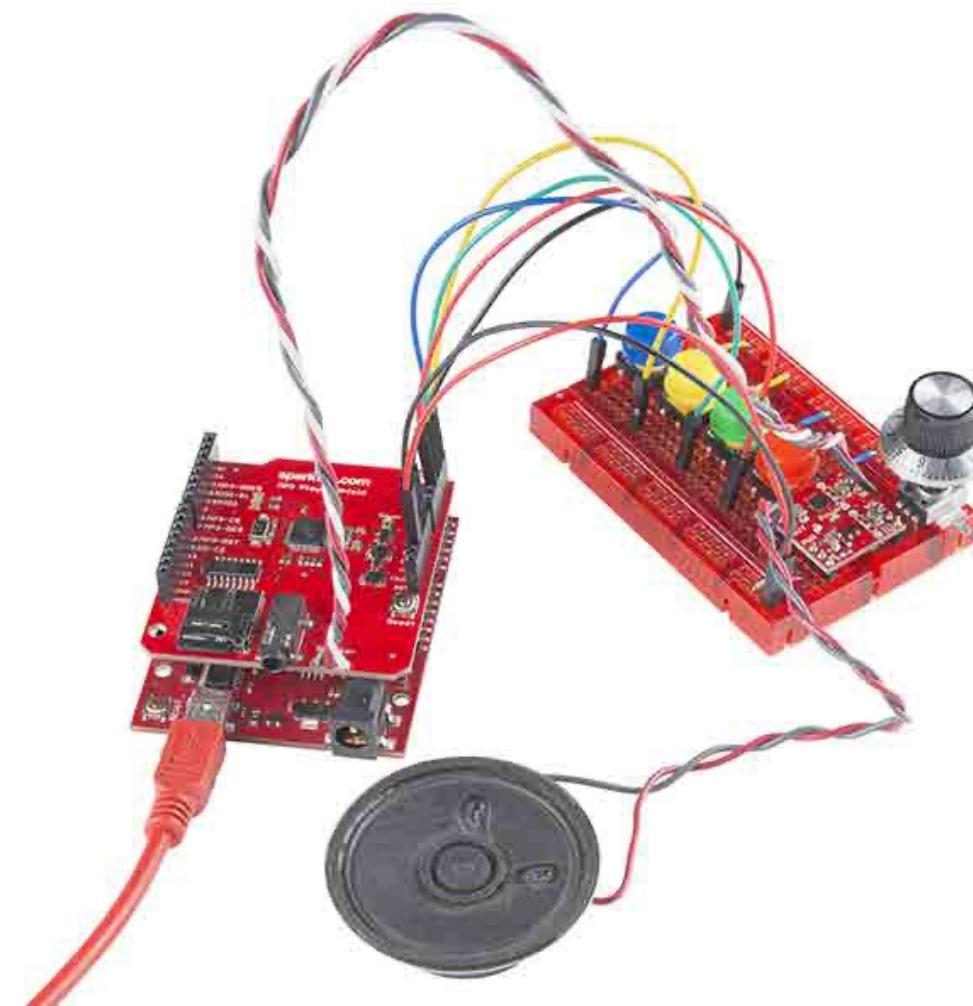
Erlaubt die Verbindung des Arduino mit dem Internet

Micro SD Kartenslot

Führt die Arduino Pins weiter

Shields

MP3 Player Shield



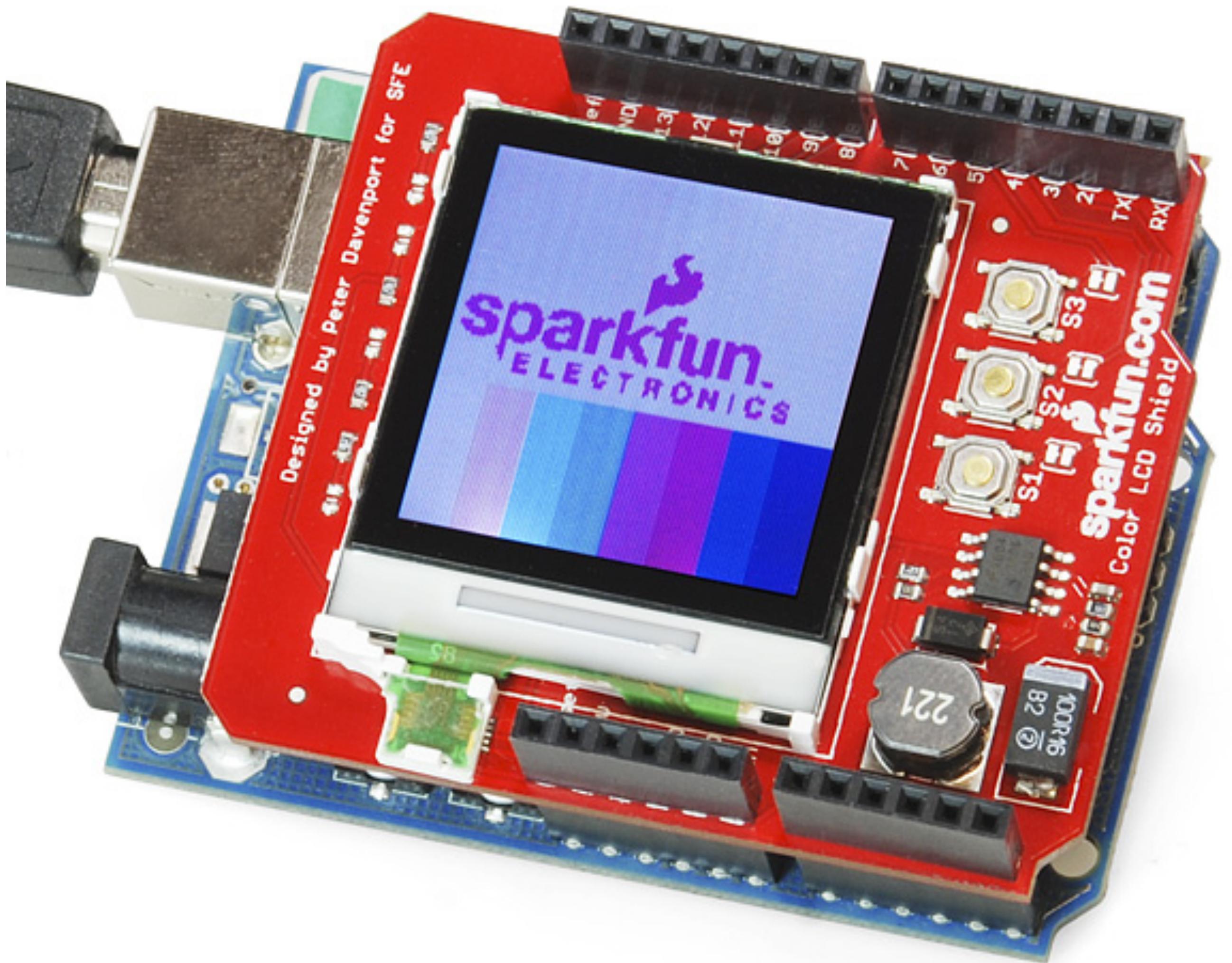
Kann Musik von Micro SD Kartenslot abspielen

Führt die Arduino Pins weiter

Klinke / Aux Anschluß

Shields

Color LCD Shield



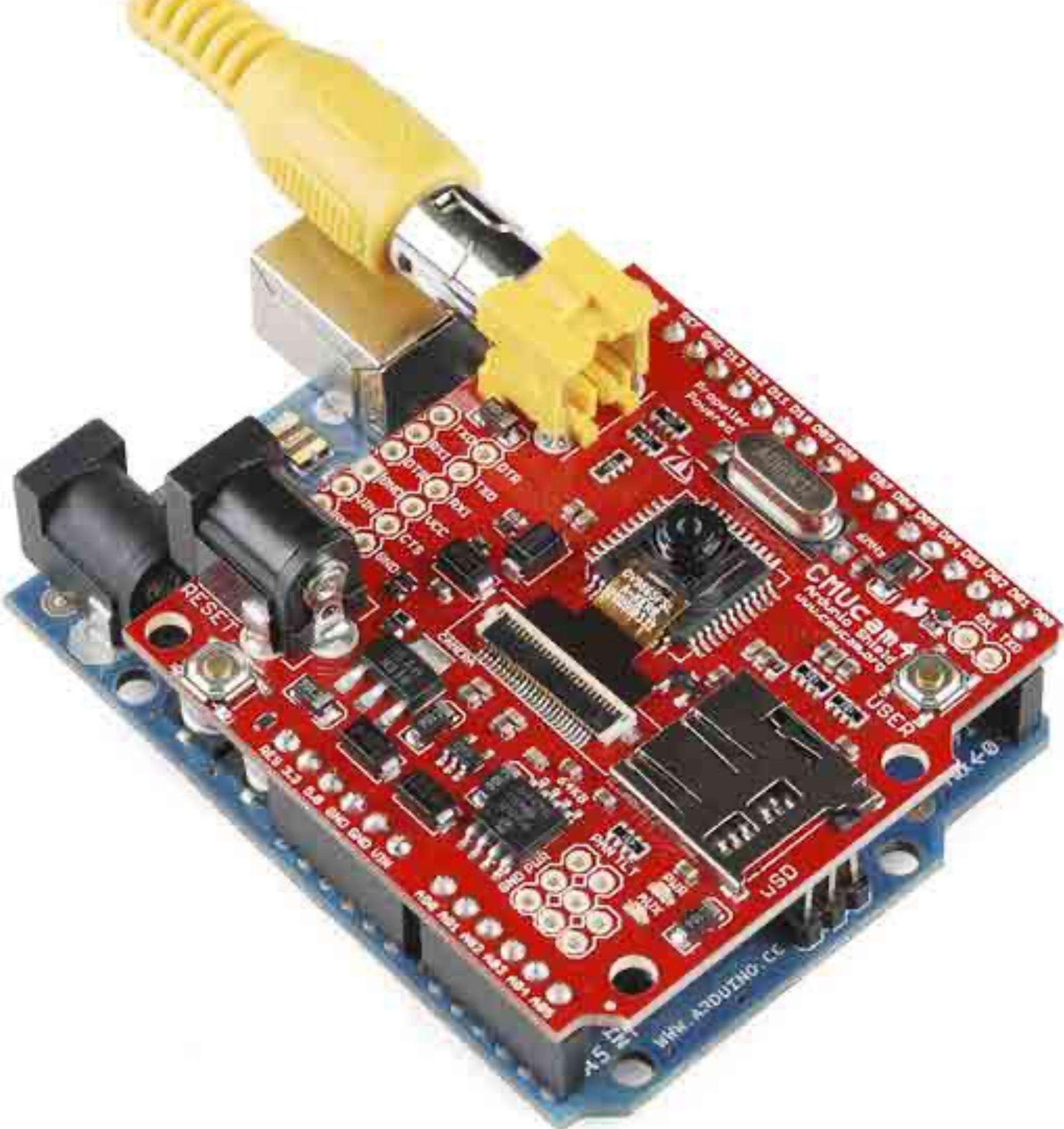
Hintergrundbeleuchtetes LCD-Display

Farbdisplay

Führt die Arduino Pins weiter

Shields

Camera Shield

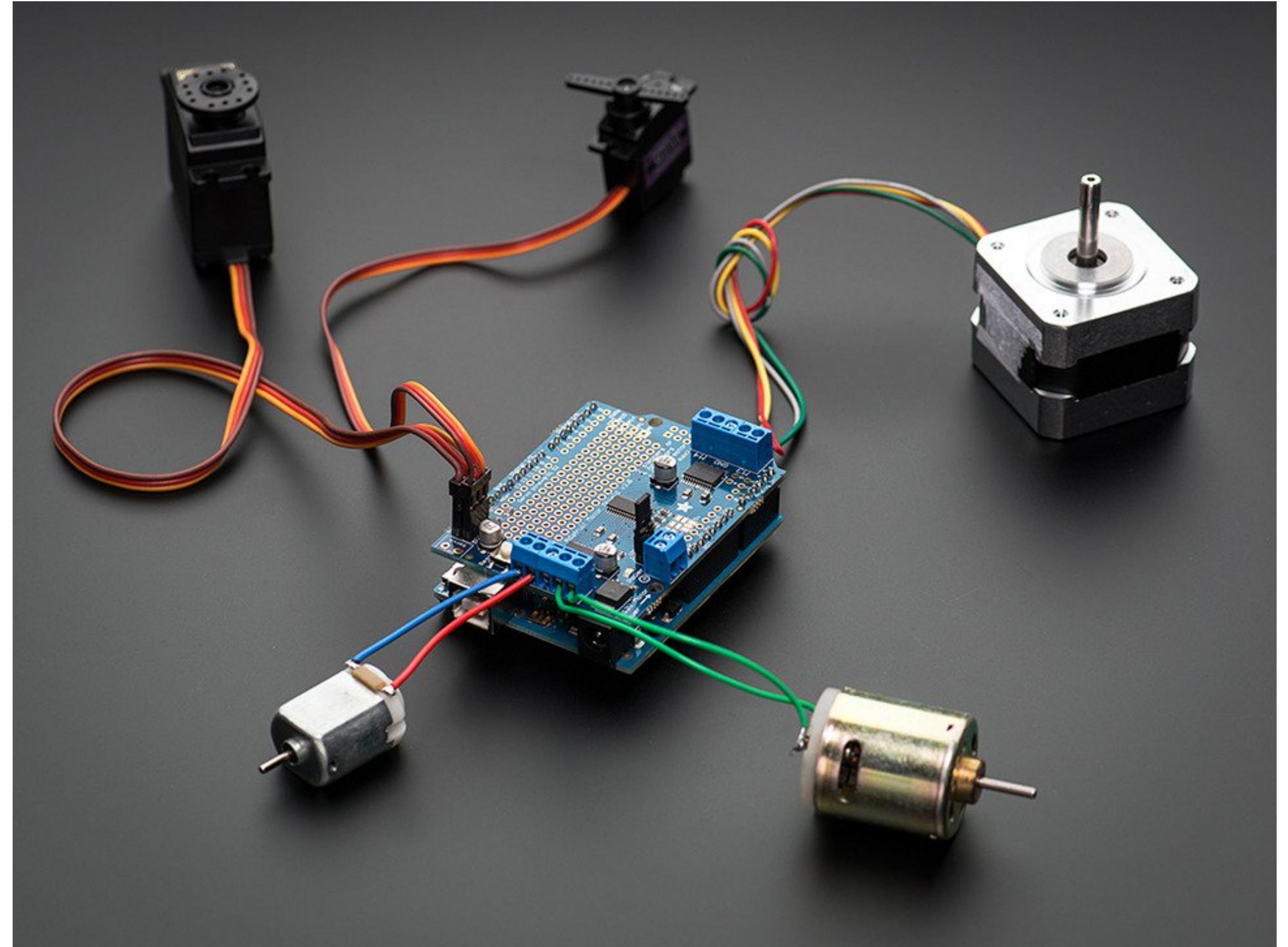


Kameramodul für den Arduino

CMOS Sensor

Shields

Motor Shield



Unterstützt DCs, Servos, Stepper

Schützt das Arduino vor Beschädigung

2A pro Motor

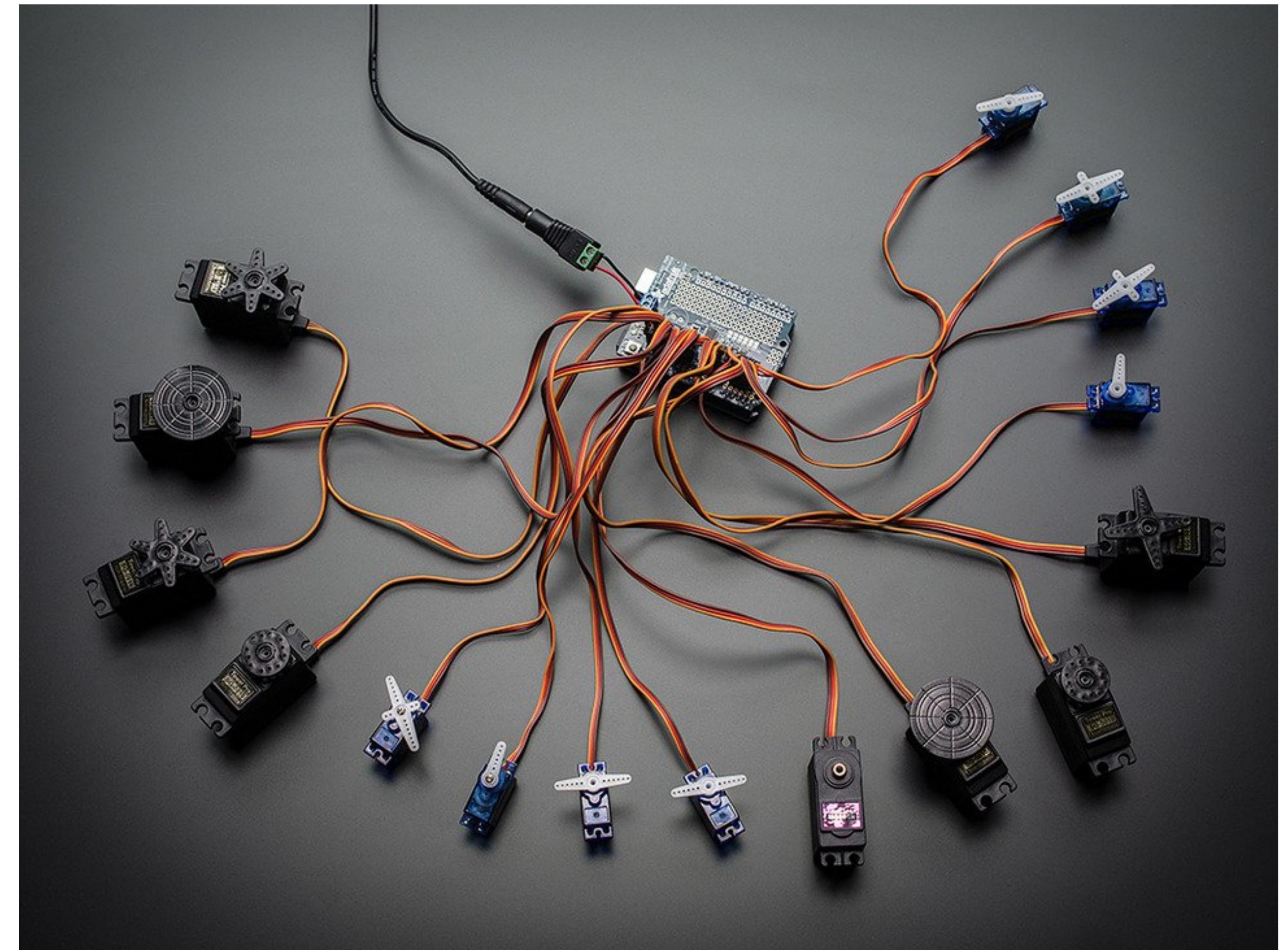
Shields

Servo Shield

Bis zu 16 Servos über PWM

I2C Interface

Shields





Bibliotheken

Arduino-IDE kann über Bibliotheken erweitert werden

Bieten zusätzliche Funktionalität für Sketches

Sind in C/C++ geschrieben

**Erleichtern das Arbeiten, da viele Funktionalitäten nicht selbst geschrieben werden müssen,
z.B. für Motoren o.ä.**

Unterteilung in Standard-Libraries und Libraries von Dritten

<https://www.arduinolibraries.info/>

<https://www.arduino.cc/en/reference/libraries>

Bibliotheken

Überprüfen/Kompilieren ⌘R
Hochladen ⌘U
Hochladen mit Programmer ⌘⌘U
Kompilierte Binärdatei exportieren ⌘⌘S

Sketch-Ordner anzeigen ⌘K
Bibliothek einbinden ►
Datei hinzufügen...

Sweep §
1 #include <Servo.h>
2
3 Servo myservo;
4 // twelve servo pins
5
6 int pos = 0;
7
8 void setup() {
9 myservo.attach(9);
10 }
11
12 void loop() {
13 for (pos = 0; pos < 180; pos += 15) {
14 // in steps of 15 degrees
15 myservo.write(pos);
16 delay(15);
17 }
18 for (pos = 180; pos >= 0; pos -= 15) {
19 myservo.write(pos);
20 delay(15);
21 }
22 }
23

Bibliotheken verwalten... ⌘I
.ZIP-Bibliothek hinzufügen...

Sweep | Arduino 1.8.9

Arduinobibliotheken

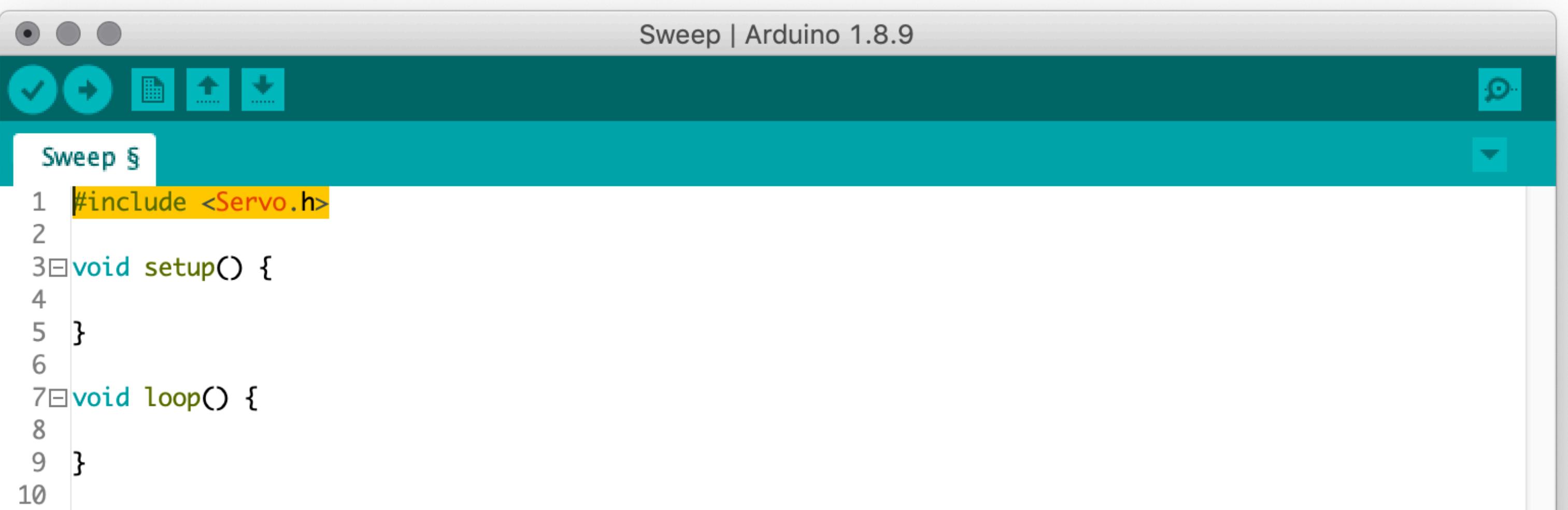
- Bridge
- Esplora
- Ethernet
- Firmata
- GSM
- Keyboard
- LiquidCrystal
- Mouse
- Robot Control
- Robot IR Remote
- Robot Motor
- SD
- Servo
- SpacebrewYun
- Stepper
- TFT
- Temboo
- WiFi

Beigetragene Bibliotheken

- Artnet
- EEPROM
- HID
- MFRC522
- SPI
- SoftwareSerial
- Wire

Empfohlene Bibliotheken

- Adafruit Circuit Playground
- Adafruit MPR121
- Adafruit PWM Servo Driver Library



The screenshot shows the Arduino IDE interface with a sketch titled "Sweep". The code includes an include statement for the Servo library and empty setup and loop functions.

```
#include <Servo.h>
void setup() {
}
void loop() {
}
```

Stichwort: #include

<NameDerBibliothek.h> // Durchsucht den Standard-Pfad

"NameDerBibliothek.h" → Library im Sketch-Ordner

Bibliotheken

Überprüfen/Kompilieren ⌘R
Hochladen ⌘U
Hochladen mit Programmer ⌘⌘U
Kompilierte Binärdatei exportieren ⌘⌘S

Sketch-Ordner anzeigen ⌘K
Bibliothek einbinden ►
Datei hinzufügen...

```
Sweep §
1 #include <Servo.h>
2
3 Servo myservo;
4 // twelve servo pins
5
6 int pos = 0;
7
8 void setup() {
9   myservo.attach(7);
10 }
11
12 void loop() {
13   for (pos = 0; pos < 180; pos += 15) {
14     // in steps of 15 degrees
15     myservo.write(pos);
16     delay(15);
17   }
18   for (pos = 180; pos >= 0; pos -= 15) {
19     myservo.write(pos);
20     delay(15);
21   }
22 }
```

Bibliotheken verwalten... ⌘⌘I

.ZIP-Bibliothek hinzufügen...

Sweep 1.8.9

Arduino Bibliotheken

- Bridge
- Esplora
- Ethernet
- Firmata
- GSM
- Keyboard
- LiquidCrystal
- Mouse
- Robot Control
- Robot IR Remote
- Robot Motor
- SD
- Servo
- SpacebrewYun
- Stepper
- TFT
- Temboo
- WiFi

Beigetragen Bibliotheken

- Artnet
- EEPROM
- HID
- MFRC522
- SPI
- SoftwareSerial
- Wire

Empfohlen Bibliotheken

- Adafruit Circuit Playground
- Adafruit MPR121
- Adafruit PWM Servo Driver Library

Sweep | Arduino 1.8.9

```
1 #include <Servo.h>
2
3 Servo myservo; // create servo object to control a servo
4 // twelve servo objects can be created on most boards
5
6 int pos = 0; // variable to store the servo position
7
```

Bibliotheksverwalter

Typ Alle Thema Alle Grenzen Sie Ihre Suche ein...

[More info](#) Version 1.0.2 [Installieren](#)

Arduino_ConnectionHandler by Ubi de Feo, Cristian Maglie, Andrea Catozzi, Alexander Entinger et al.
Arduino Library for network connection management (WiFi, GSM, [Ethernet]) Originally part of ArduinoIoTCloud
[More info](#)

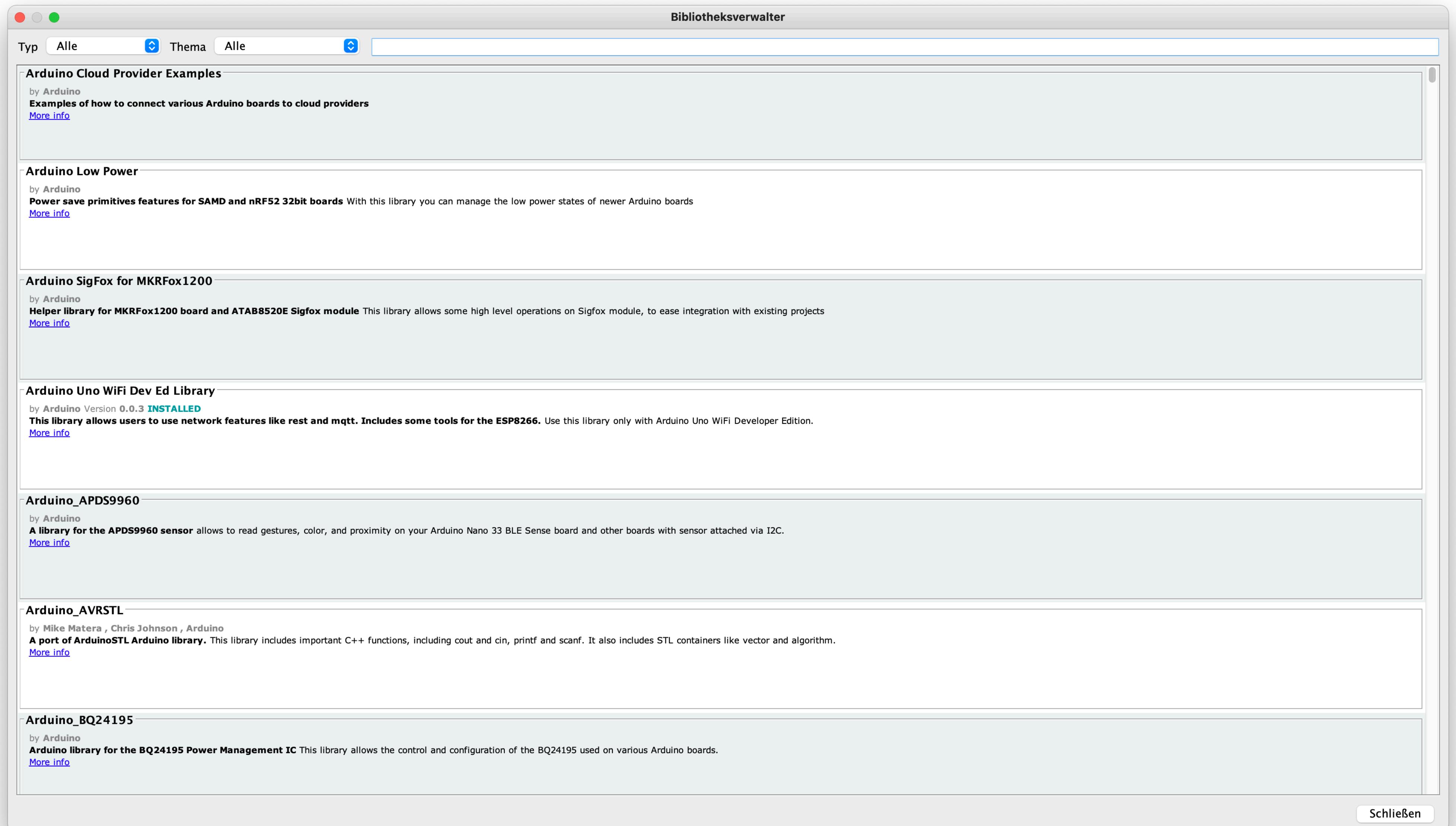
Arduino_DebugUtils by Arduino
Debugging module with different debug levels, timestamps and printf-style output. This class provides functionality useful for debugging sketches via printf-style statements.
[More info](#)

Arduino_HTS221 by Arduino
Allows you to read the temperature and humidity sensors of your Nano 33 BLE Sense.
[More info](#)

Schließen

20 Arduino/Genuino Uno auf /dev/cu.usbmodem14101

Bibliotheksverwalter



Bibliotheken

Multitasking

delay()

Pro

- Einfach zu verstehen (für Anfänger)
- Schnell zu benutzen

Contra

- Es blockiert das komplette Board
- Arduino "schläft" → Kriegt von der Außenwelt in der delay() Zeit nichts mit
- Komplexe Programme nicht umsetzbar



```

Blink | Arduino 1.8.13

Blink
1 /*
2  *  Blink
3  *
4  *  Turns an LED on for one second, then off for one second, repeatedly.
5  *
6  *  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
7  *  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
8  *  the correct LED pin independent of which board is used.
9  *  If you want to know what pin the on-board LED is connected to on your Arduino
10 *  model, check the Technical Specs of your board at:
11 *  https://www.arduino.cc/en/Main/Products
12 *
13 *  modified 8 May 2014
14 *  by Scott Fitzgerald
15 *  modified 2 Sep 2016
16 *  by Arturo Guadalupi
17 *  modified 8 Sep 2016
18 *  by Colby Newman
19 *
20 *  This example code is in the public domain.
21 *
22 *  http://www.arduino.cc/en/Tutorial/Blink
23 */
24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27     // initialize digital pin LED_BUILTIN as an output.
28     pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33     digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on (HIGH is the voltage level)
34     delay(1000);                      // wait for a second
35     digitalWrite(LED_BUILTIN, LOW);     // turn the LED off by making the voltage LOW
36     delay(1000);                      // wait for a second
37 }

```

The screenshot shows the Arduino IDE interface with the title bar "Blink | Arduino 1.8.13". The code editor displays the "Blink" example sketch. The code is annotated with comments explaining its functionality. Lines 34 and 36, which contain the `delay(1000)` statements, are highlighted in yellow.

Multitasking

```
Blink §  
1 int ledPin = 9;  
2 int sensorPin = A0;  
3 int value = 0;  
4  
5 void setup() {  
6   pinMode(ledPin, OUTPUT);  
7   pinMode(sensorPin, INPUT);  
8 }  
9  
10 void loop() {  
11   value = analogRead(sensorPin);  
12   value = map(value, 800, 1023, 0, 255);  
13   analogWrite(ledPin, value);  
14   delay(50); // unser arduino wird von hier an für 50ms in den *schlaf* gelegt  
15  
16   // anderen code "parallel" ausführen? nicht möglich. bei zeitkritischem code schlecht  
17 }  
18
```

- LED blinkt
- Nachfolgende Prozesse werden aber immer 50ms versetzt zur "echten Zeit" ausgeführt
- In zeitsensiblen Projekten baut sich hier automatisch ein Zeitrückstand auf

Multitasking

The screenshot shows the Arduino IDE interface with a sketch titled "Blink". The code is as follows:

```
1 int ledPin1 = 9;
2 int ledPin2 = 10;
3 int sensorPin = A0;
4 int value = 0;
5
6 void setup() {
7     pinMode(ledPin1, OUTPUT);
8     pinMode(ledPin2, OUTPUT);
9     pinMode(sensorPin, INPUT);
10}
11
12 void loop() {
13     value = analogRead(sensorPin);
14     value = map(value, 800, 1023, 0, 255);
15     analogWrite(ledPin1, value);
16     delay(1000); // << block
17     analogWrite(ledPin2, value);
18     delay(1000); // << block
19
20     // anderen code "parallel" ausführen? nicht möglich. bei zeitkritischem code schlecht
21}
22
```

Multitasking

Funktionen – Time

delay()

- Pausiert das Programm (das *ganze* Board)
- Wert in Millisekunden
- Während der Pause kann / wird nichts ausgeführt
- Schlecht für Synchronisierungsprozesse
- `delay(ms)`

millis()

- Gibt die Anzahl der Millisekunden seit dem Programmstart zurück
- `long timestamp = millis()`

delayMicroseconds()

- Pausiert das Programm (das *ganze* Board)
- Wert in Mikrosekunden
- Während der Pause kann / wird nichts ausgeführt
- Schlecht für Synchronisierungsprozesse
- `delayMicroseconds(us)`

micros()

- Gibt die Anzahl der Mikrosekunden seit dem Programmstart zurück
- `long timestamp = micros()`

Multitasking

millis() für Multitasking

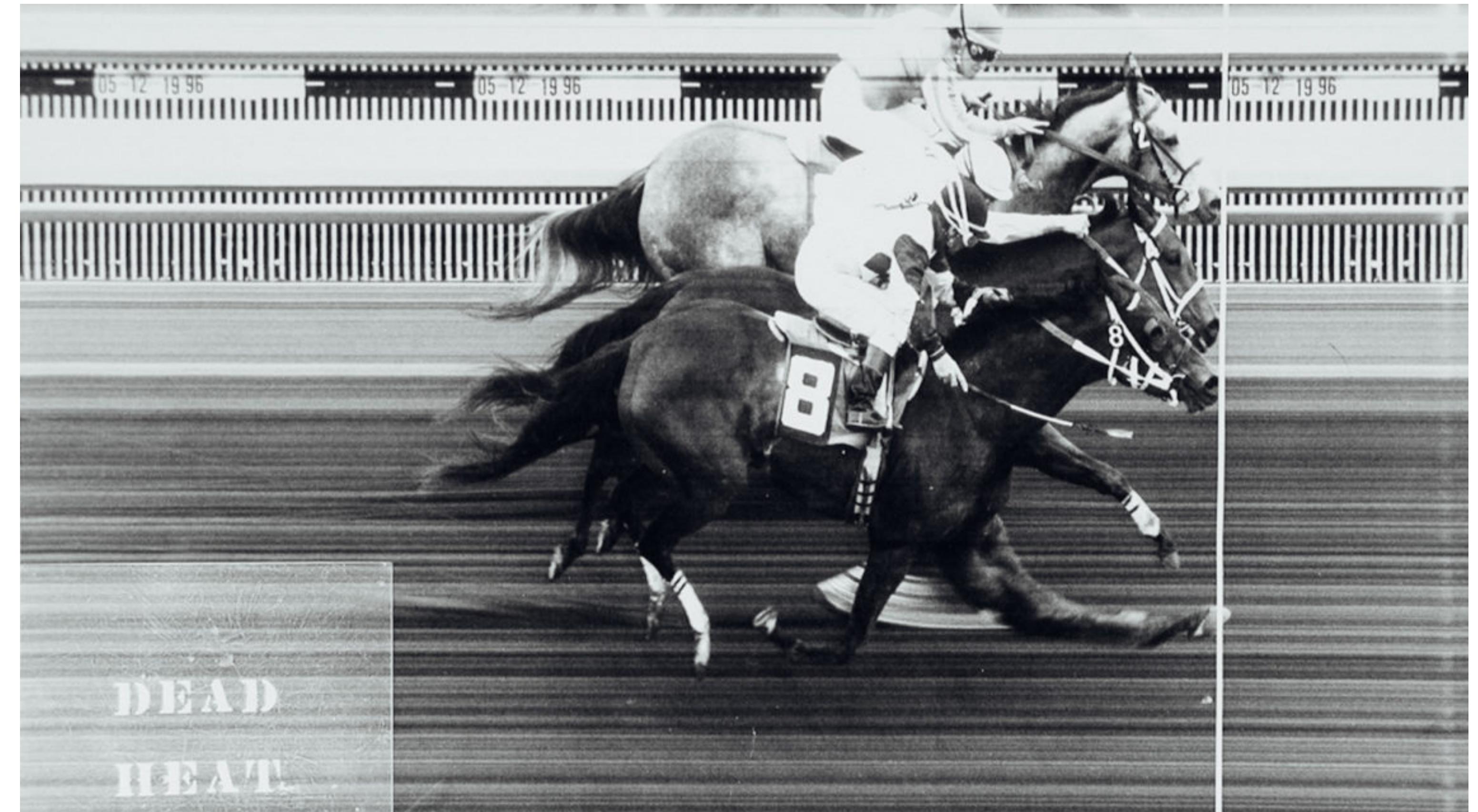
Zeitdifferenz errechnen

Hardwareunabhängig

Zwei Messpunkte

Ein Mess-Interval

Multitasking



millis() für Multitasking

```
void setup() {  
}  
  
void loop() {  
}
```

Multitasking

millis() für Multitasking

```
long timestamp = 0;  
long interval = 1000;  
  
void setup() {  
}  
  
void loop() {  
}
```

Multitasking

millis() für Multitasking

```
long timestamp = 0;  
long interval = 1000;  
  
void setup() {  
}  
  
void loop() {  
    if(millis() - timestamp > interval) {  
    }  
}
```

Multitasking

millis() für Multitasking

```
long timestamp = 0;  
long interval = 1000;  
  
void setup() {  
}  
  
void loop() {  
    if(millis() - timestamp > interval) {  
        timestamp = millis();  
        // unser zeitsensibler code  
    }  
}
```

Multitasking

Cycle n	millis()	timestamp	interval
0	0	0	1000
1	1	0	1000
2	2	0	1000
3	3	0	1000
4	4	0	1000
...
10	10	0	1000
...
1001	1001	0	1000

Multitasking

millis() für Multitasking

```
long timestamp = 0;  
long interval = 1000;  
  
void setup() {  
}  
  
void loop() {  
    if(millis() - timestamp > interval) {  
        timestamp = millis();  
        // unser zeitsensibler code  
    }  
}
```

Cycle 0

```
if( 0 - 0 > 1000) {  
    // unser code  
}
```

Cycle 1001

```
if( 1001 - 0 > 1000) {  
    // unser code  
}
```

Multitasking

```
long timestamp0 = 0;  
long interval0 = 1000;  
long timestamp1 = 0;  
long interval1 = 2500;  
  
void setup() {  
    Serial.begin(9600);  
}  
  
void loop() {  
    if(millis() - timestamp0 > interval0) {  
        timestamp0 = millis();  
        Serial.println("jede sekunde");  
    }  
    if(millis() - timestamp1 > interval1) {  
        timestamp1 = millis();  
        Serial.println("alle 2.5 sekunden");  
    }  
}
```

Multitasking

```

Blink | Arduino 1.8.13

Blink §

1 int ledPin1 = 9;
2 int ledPin2 = 10;
3 int sensorPin = A0;
4 int value = 0;
5
6 void setup() {
7   pinMode(ledPin1, OUTPUT);
8   pinMode(ledPin2, OUTPUT);
9   pinMode(sensorPin, INPUT);
10 }
11
12 void loop() {
13   value = analogRead(sensorPin);
14   value = map(value, 800, 1023, 0, 255);
15   analogWrite(ledPin1, value);
16   delay(1000); // << block
17   analogWrite(ledPin2, value);
18   delay(1000); // << block
19
20   // anderen code "parallel" ausführen? nicht möglich. bei zeitkritischem code schlecht
21 }
22

```

Arduino Mega or Mega 2560, ATmega2560 (Mega 2560) auf /dev/cu.usbmodem141401

Multitasking

```

int ledPin1 = 9;
int ledPin2 = 9;
int sensorPin = A0;
int value = 0;
long timestamp0 = 0;
long interval0 = 1000;
long timestamp1 = 0;
long interval1 = 2500;
int ledState = LOW; // für unsere zweite LED

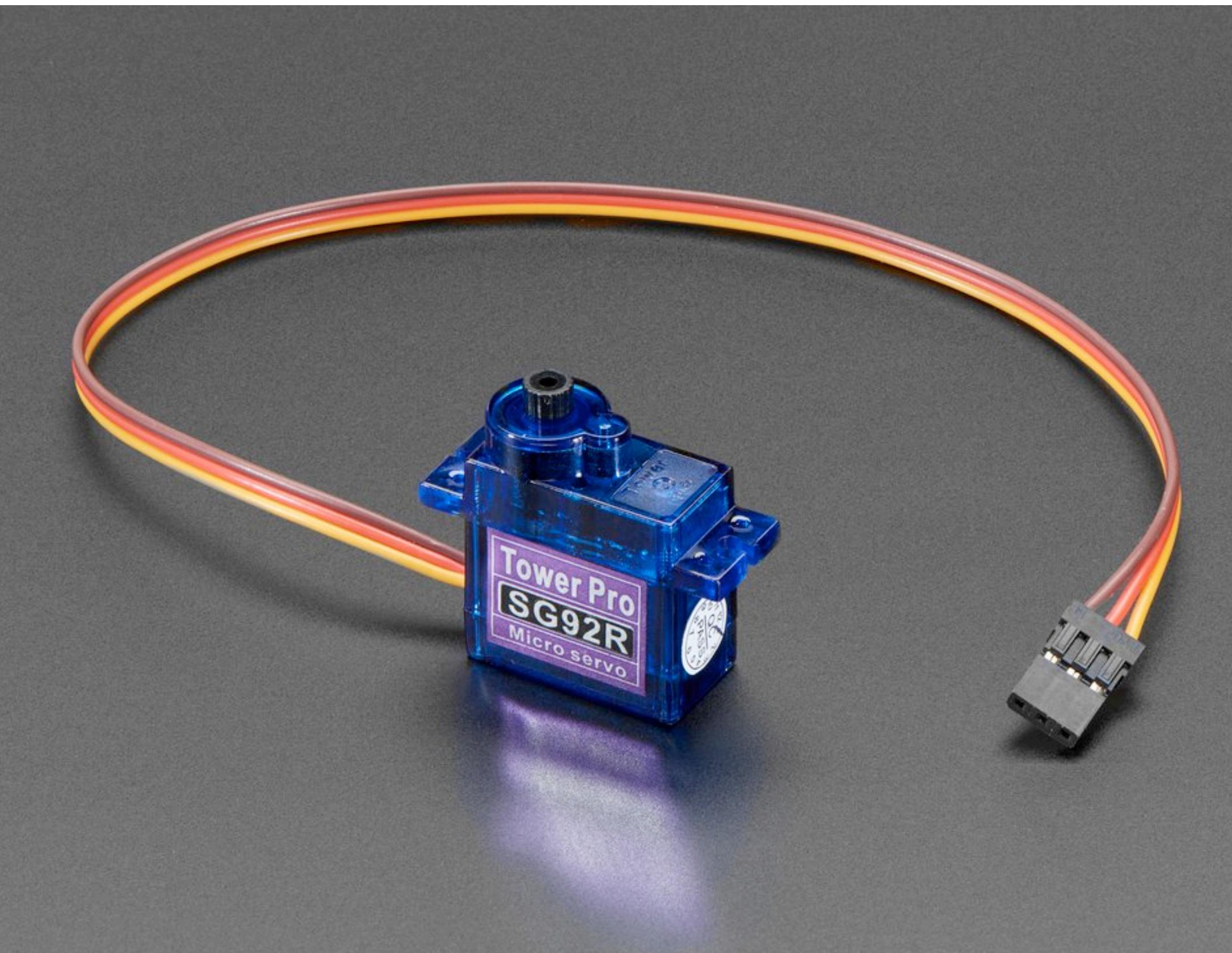
void setup() {
  pinMode(ledPin1, OUTPUT);
  pinMode(ledPin2, OUTPUT);
  pinMode(sensorPin, INPUT);
}

void loop() {
  value = analogRead(sensorPin);
  value = map(value, 800, 1023, 0, 255);
  if(millis() - timestamp0 > interval0) {
    timestamp0 = millis();
    analogWrite(ledPin1, value);
  }
  if(millis() - timestamp1 > interval1) {
    timestamp1 = millis();
    // Blinkt immernoch alle 2.5 Sekunden!
    if(ledState == LOW) {
      ledState == HIGH;
    } else if(ledState == HIGH) {
      ledState == LOW;
    }
    digitalWrite(ledPin2, ledState);
  }
}

```

Workshop Servos

Servomotor SG90



2 Drehrichtungen

Rotation zwischen 0° und 180°

Enthält Zahnräder und eine Welle

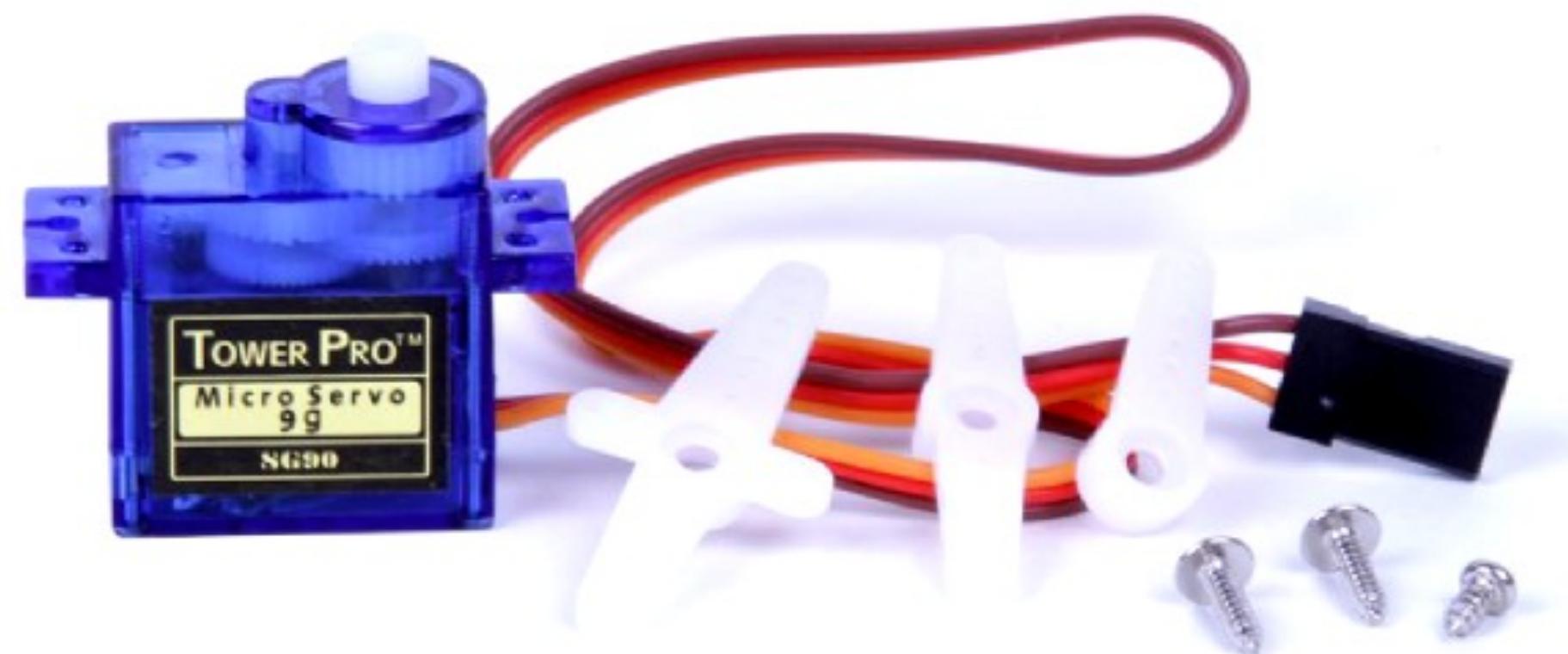
Genaue Positionierung über IC + Potentiometer

Kann direkt an den Arduino angeschlossen werden

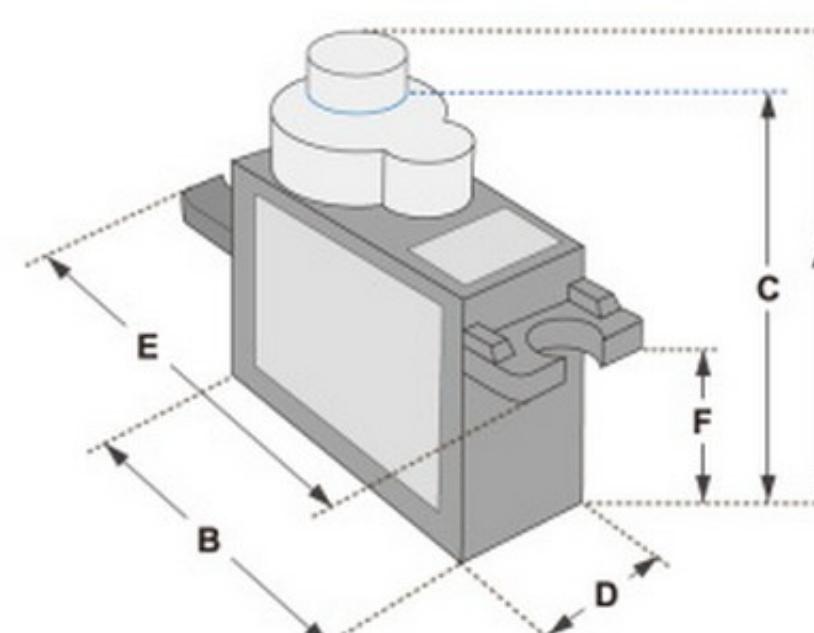
Besser: Servomotor an separate Spannungsversorgung

SERVO MOTOR SG90

DATA SHEET

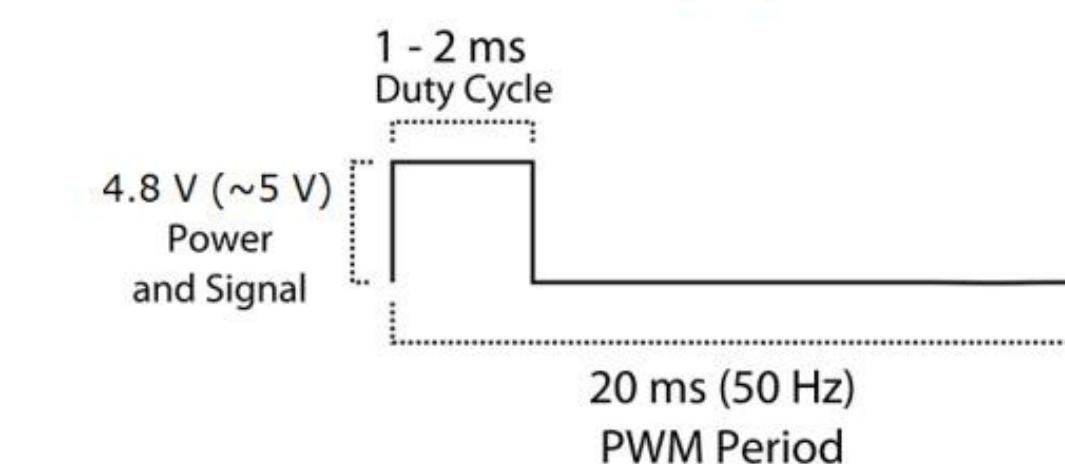
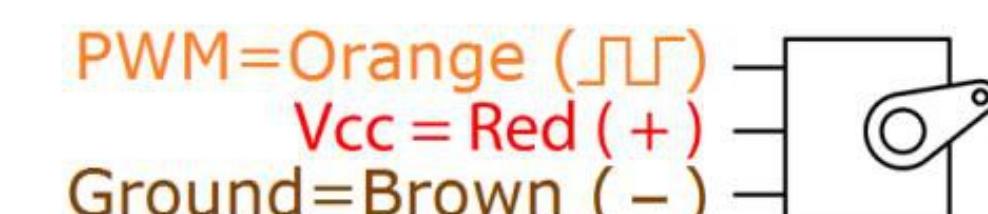


Tiny and lightweight with high output power. Servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller. You can use any servo code, hardware or library to control these servos. Good for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. It comes with a 3 horns (arms) and hardware.



Dimensions & Specifications	
A (mm)	: 32
B (mm)	: 23
C (mm)	: 28.5
D (mm)	: 12
E (mm)	: 32
F (mm)	: 19.5
Speed (sec)	: 0.1
Torque (kg-cm)	: 2.5
Weight (g)	: 14.7
Voltage	: 4.8 - 6

Position "0" (1.5 ms pulse) is middle, "90" (~2ms pulse) is middle, is all the way to the right, "-90" (~1ms pulse) is all the way to the left.

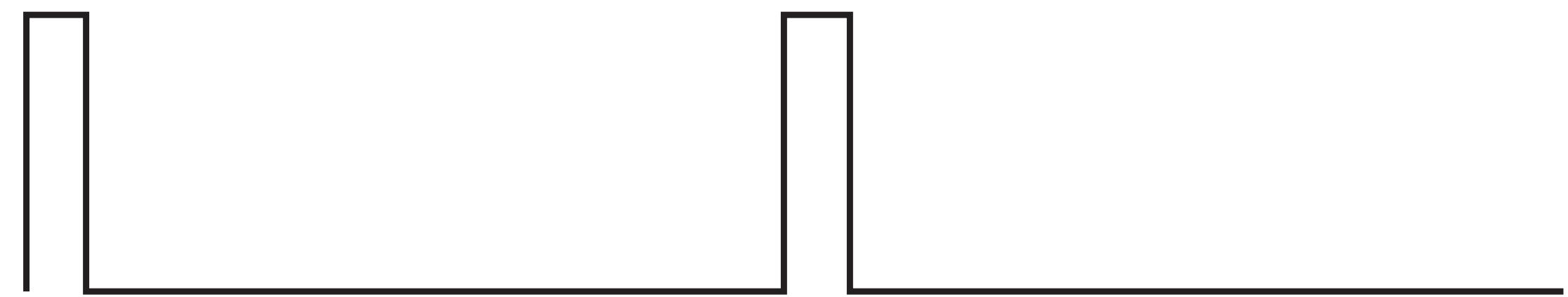


Workshop Servos

Workshop Servos

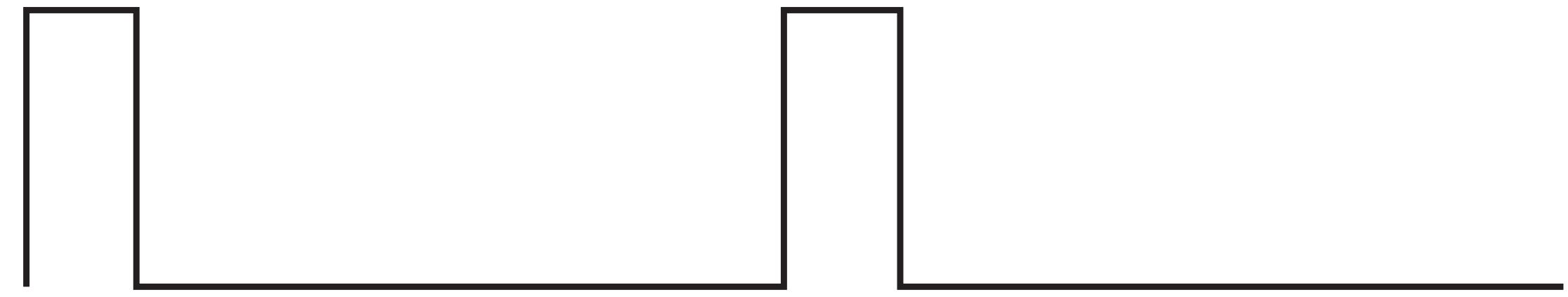
20ms Periode

Minimalpuls



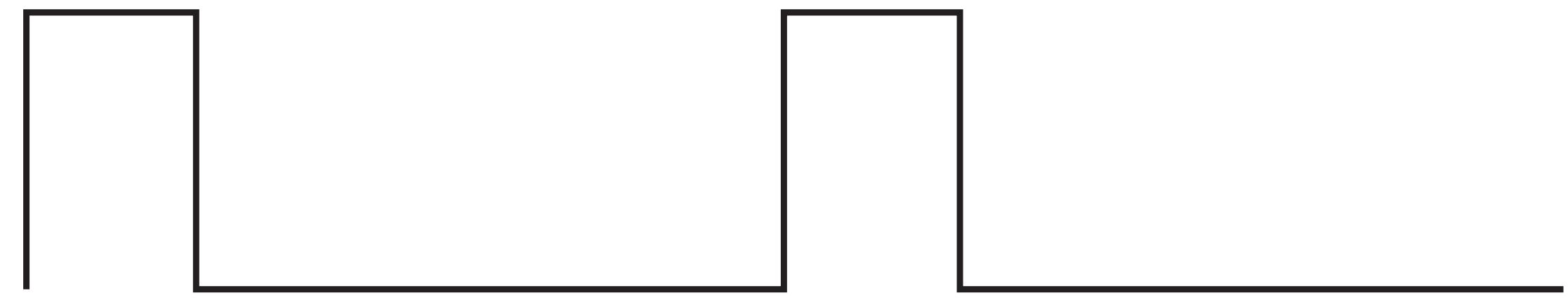
0°

Neutral



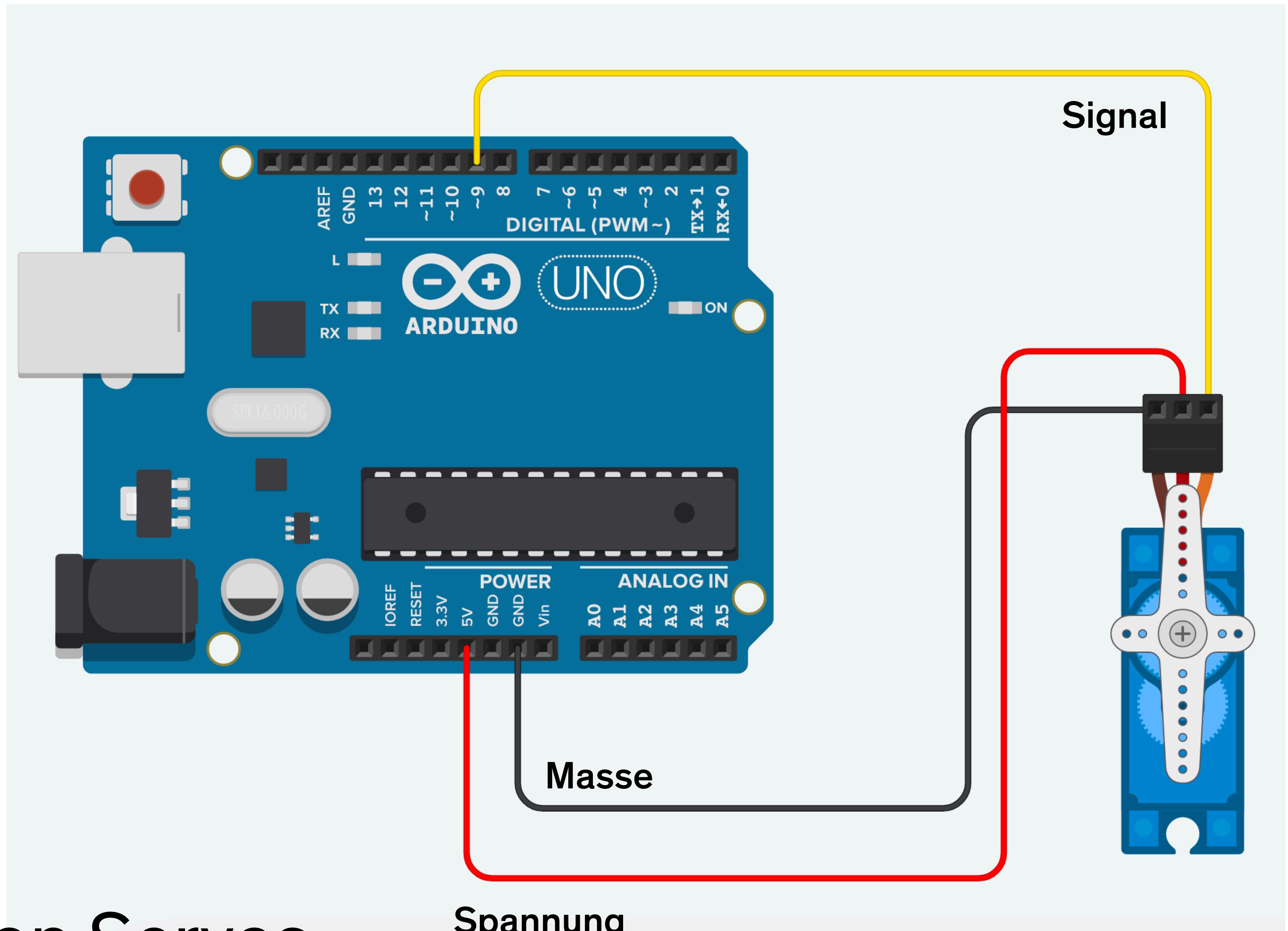
90°

Maximalpuls



180°

2ms



Workshop Servos

Pin Nummer	Signaltyp	Schema 1	Schema 2	Schema 3
1	GND (Erde)	Schwarz	Braun	Schwarz
2	VCC	Rot	Rot	Rot oder Braun
3	Steuerung	Weiß	Orange	Gelb oder Weiß

Workshop Servos

Servomotor Bibliothek

Usage

This library allows an Arduino board to control RC (hobby) servo motors. Servos have integrated gears and a shaft that can be precisely controlled. Standard servos allow the shaft to be positioned at various angles, usually between 0 and 180 degrees. Continuous rotation servos allow the rotation of the shaft to be set to various speeds.

The Servo library supports up to 12 motors on most Arduino boards and 48 on the Arduino Mega. On boards other than the Mega, use of the library disables `analogWrite()` (PWM) functionality on pins 9 and 10, whether or not there is a Servo on those pins. On the Mega, up to 12 servos can be used without interfering with PWM functionality; use of 12 to 23 motors will disable PWM on pins 11 and 12.

To use this library:

```
#include <Servo.h>
```

Examples

- **Knob**: control the shaft of a servo motor by turning a potentiometer
- **Sweep**: sweeps the shaft of a servo motor back and forth

Methods

- `attach()`
- `write()`
- `writeMicroseconds()`
- `read()`
- `attached()`
- `detach()`

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

<https://funduino.de/>

Code Referenz + Link für Übungen

Coding Challenge

Verwendet 3 Taster um einen Servomotor in die Position 0°, 90° und 180° stellen zu können.

Keine Abgabe

Aufgabe 3

Baut ein System, welches nach dem EVA-Prinzip funktioniert. Dabei könnt ihr beliebige Hardware für die Ein- und Ausgabe verwenden.

Aufgabe 3 – Bis 28.05.2021

→ Baut ein System, welches nach dem EVA-Prinzip funktioniert. Dabei könnt ihr beliebige Hardware für die Ein- und Ausgabe verwenden.

Nutzt dazu:

- Arduino
- Breadboard
- Eingabemodul (E-Teil)
- Informationsverarbeitung im Arduino (V-Teil)
- Ausgabemodul (A-teil)
- Arduino-Referenz
- Min. eine Bibliothek
- Passender Programmcode

Aufgabe 3 – Bis 28.05.2021

- 1. Plant eure Aufbauten in TinkerCad (nutzt gerne die "Simulation" Funktion)**
- 2. Zeichnet die Schaltpläne sauber mit der Hand**
- 3. Dokumentiert:**
 - Fotos: Echter Aufbau + Schaltplan**
 - Screenshots: TinkerCad**
 - Kurze Beschreibung (Bauteile, Vorgehensweise)**
 - Quellcode**
 - Name, Matrikelnummer**
 - Aufgabe, Datum**
- 4. Abgabe auf Incom als PDF im "Abgabe" Ordner**

Nutzt die Vorlesungen und das Tutorium mit Christoph Schubert (s. Incom)

Aufgabe 3 – Bis 28.05.2021

Fragen?