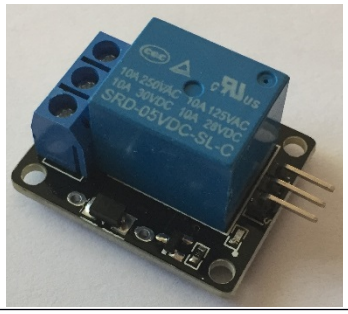
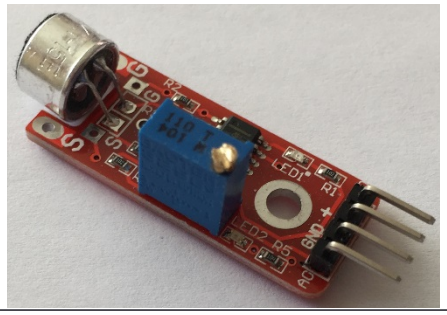
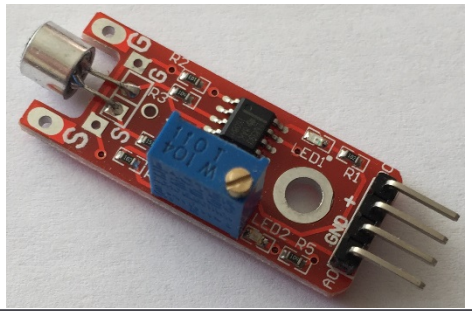
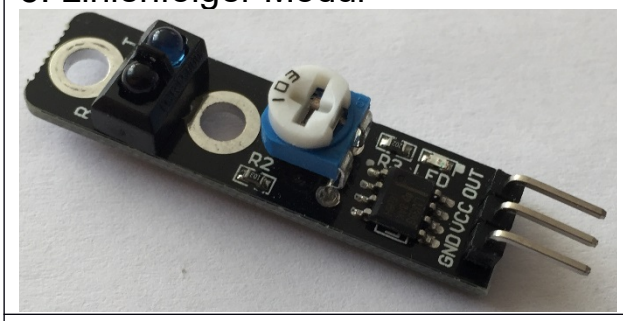
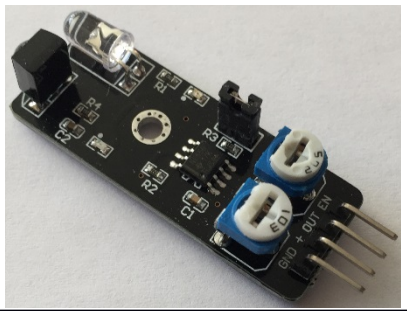
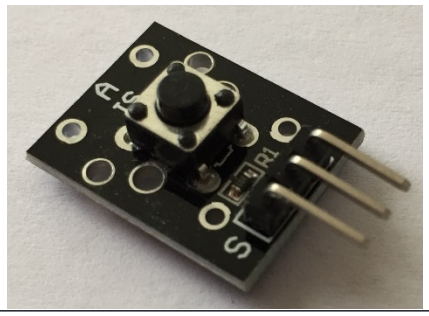
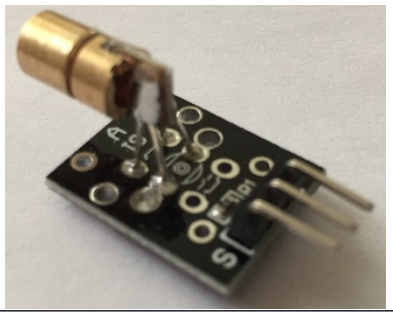
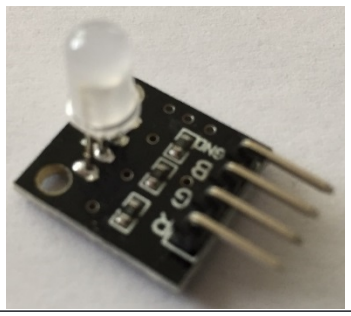
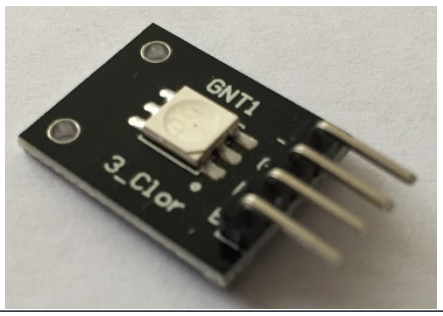
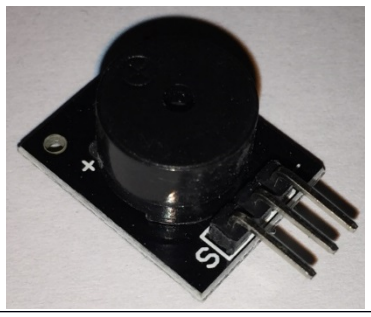
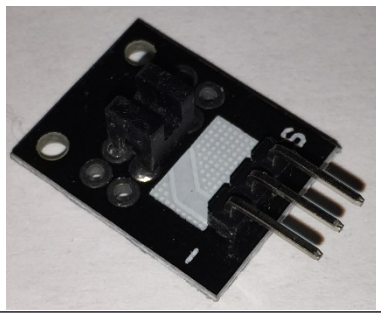
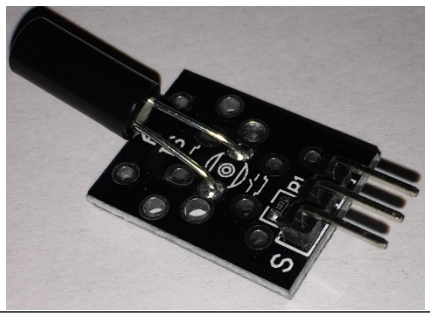
Willkommen!

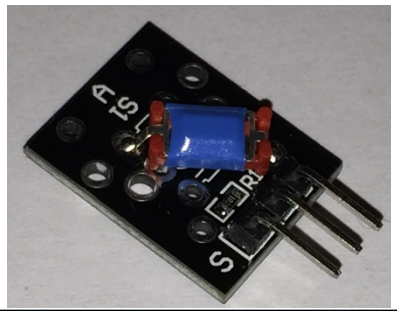
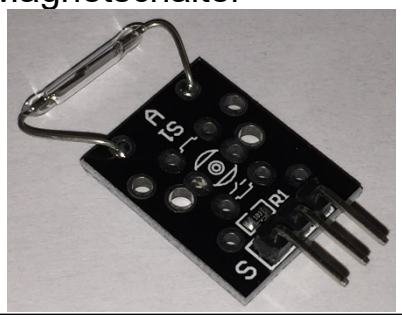
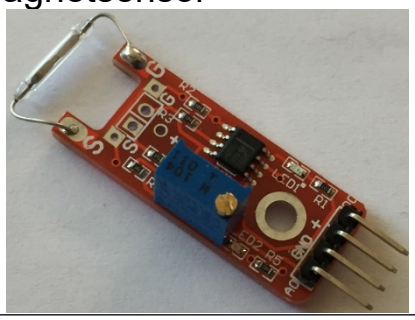
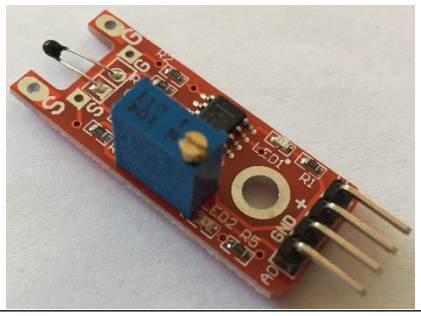
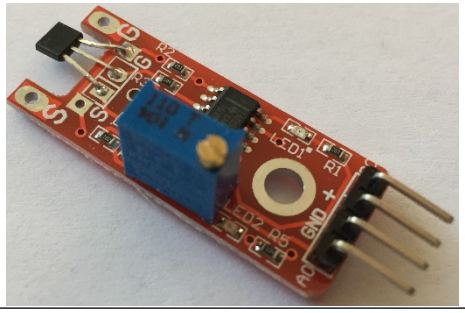
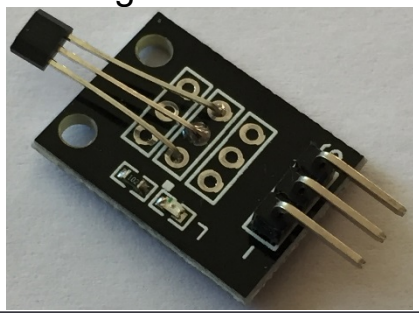
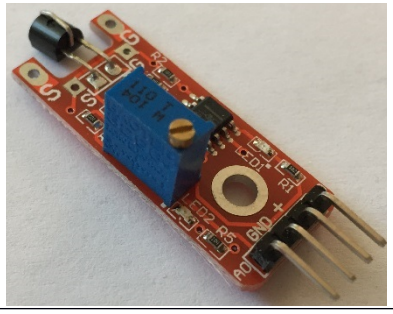
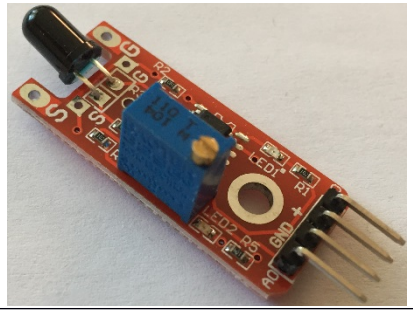
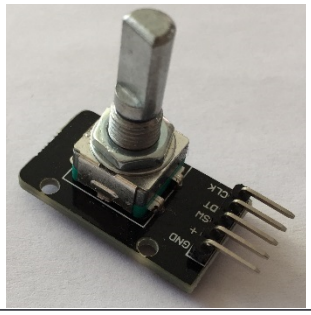
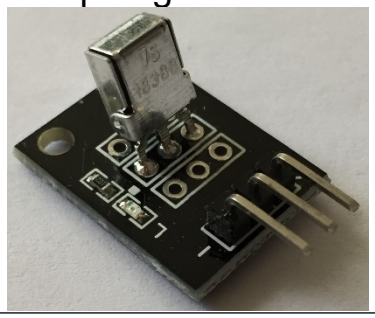
Und herzlichen Dank für den Kauf unseres AZ-Delivery 35 in 1 Arduino-Sensoren- und Zubehörkit! Auf den folgenden Seiten gehen wir mit dir gemeinsam die einzelnen Sensoren von der Einrichtung bis hin zum Programmieren durch.
Viel Spaß!

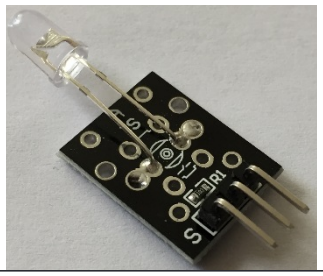
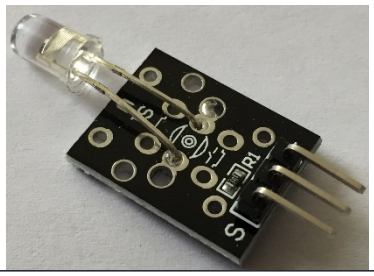
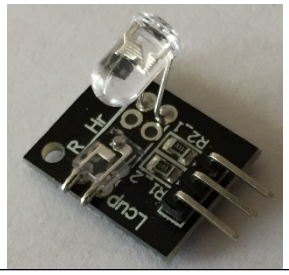
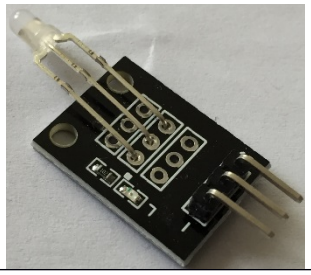
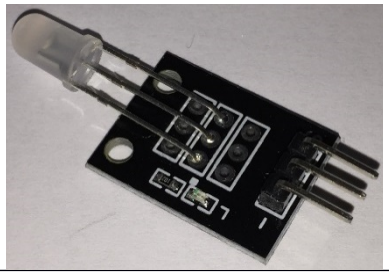
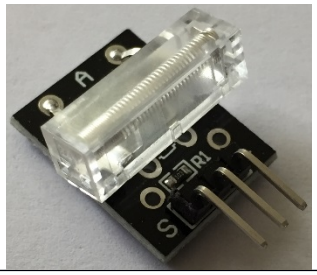
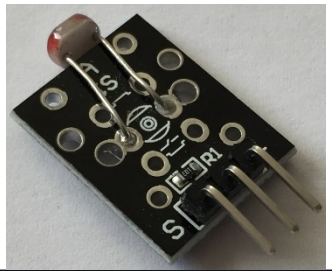
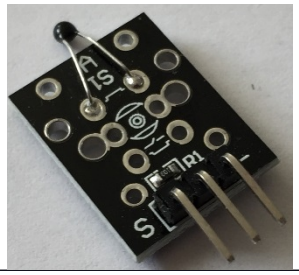
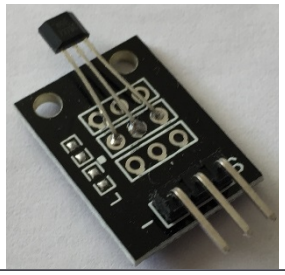
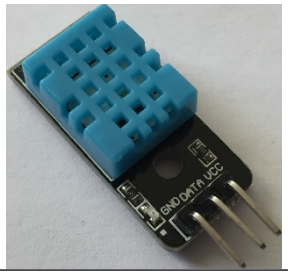
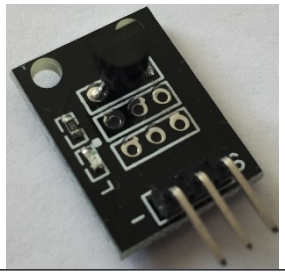
Dein Set enthält:

1. Joystick 	2. Relais 
3. Großes Mikrofonmodul 	4. Kleines Mikrofonmodul 

AZ-Delivery

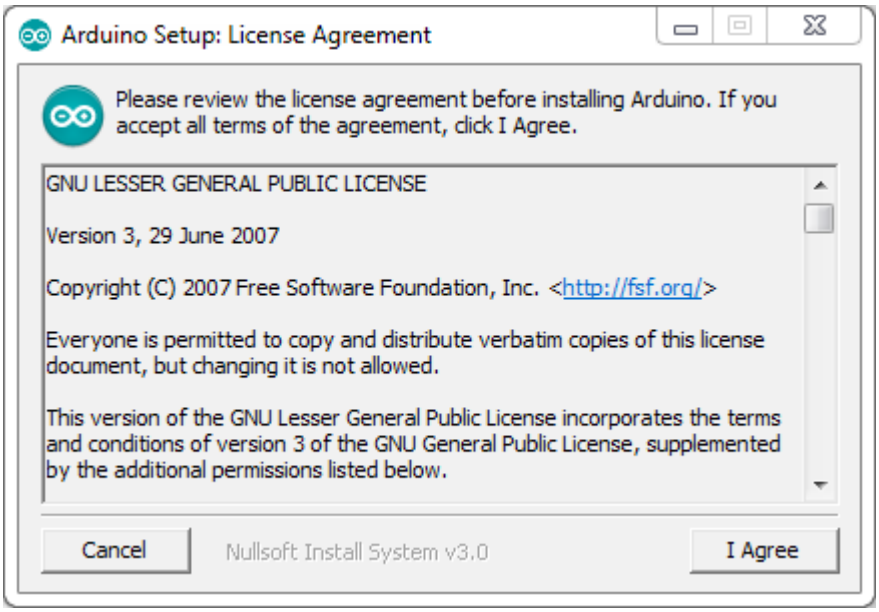
5. Linienfolger Modul 	6. Hindernissensor 
7. Taster 	8. Laser 
9. RGB-LED 	10. SMD RGB-LED 
11. Aktiver Buzzer 	12. Passiver Buzzer 
13. Lichtschranke 	14. Schocksensor 

15. Schüttelsensor 	16. Magnetschalter 
17. Magnetsensor 	18. Thermistor 
19. Hallsensor 	20. Digitaler Hallsensor 
21. Touchsensor 	22. Flammensensor 
23. Drehschalter 	24. IR-Empfänger 

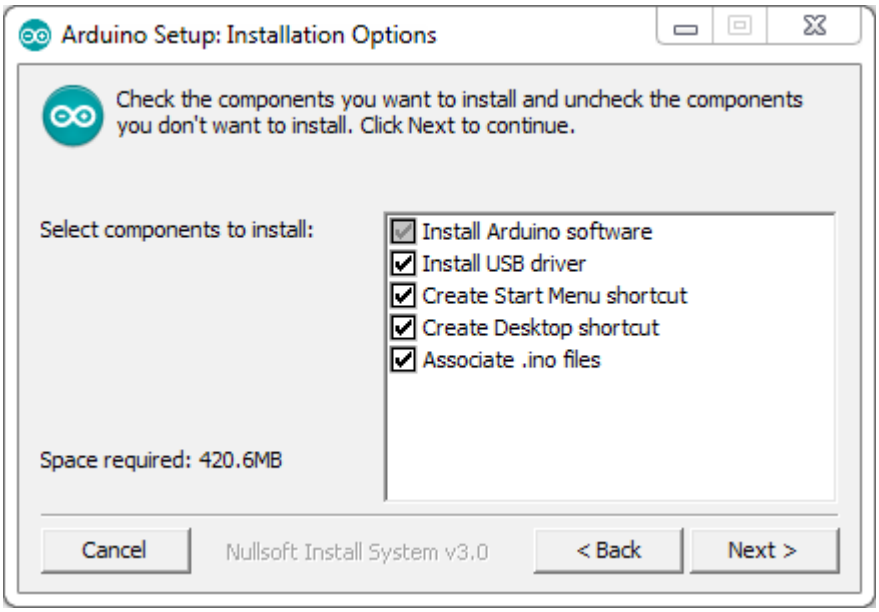
25. IR-Sender 	26. Farbwechsel-LED 
27. IR-Lichtschranke 	28. Bicolor LED 3mm 
29. Bicolor LED 5mm 	30. Schocksensor 
31. LDR Widerstand 	32. Thermistor 
33. Hallsensor 	34. DHT11 
35. DS18B20 	

Installation der Software:

Bevor wir mit dem Programmieren beginnen können, müssen wir uns die Arduino Software von https://www.arduino.cc/en/Main/Software# herunterladen. Nach dem Download und starten wir den Installer und es erscheint folgender Bildschirm:

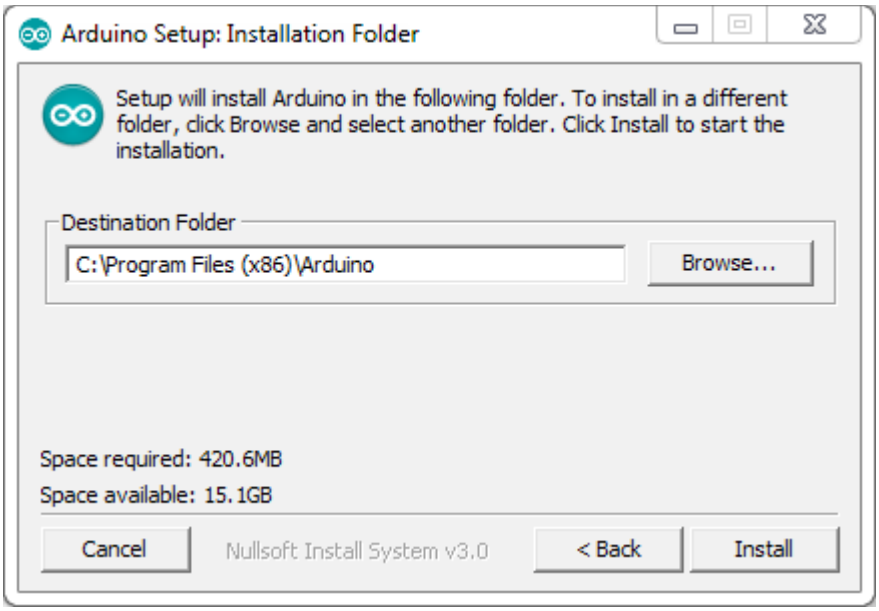


Dieses Fenster bestätigen wir mit „I Agree“ sofern du die Lizenzbestimmungen akzeptierst.

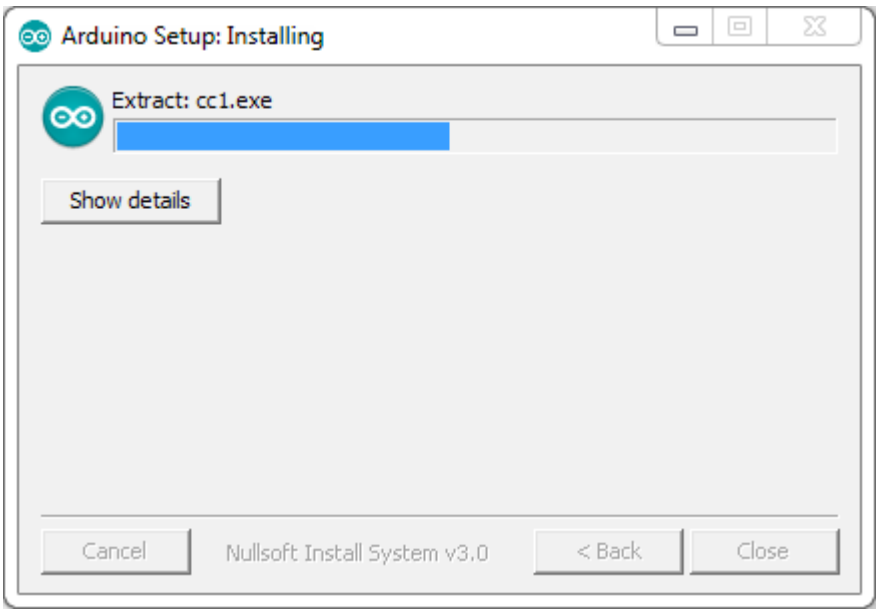


Im nächsten Fenster können wir auswählen, von wo aus wir die Arduino Software starten können und ob wir auch die USB-Treiber mit installieren möchten. Am besten man setzt die Häkchen wie im Bild oben zu sehen ist.

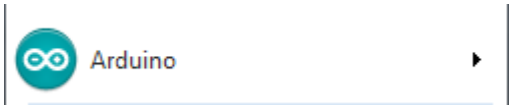
Als nächsten Schritt geben wir das Installationsverzeichnis an, das Standard-Verzeichnis sollte in der Regel stimmen:



Und schon wird die Arduino Software installiert.

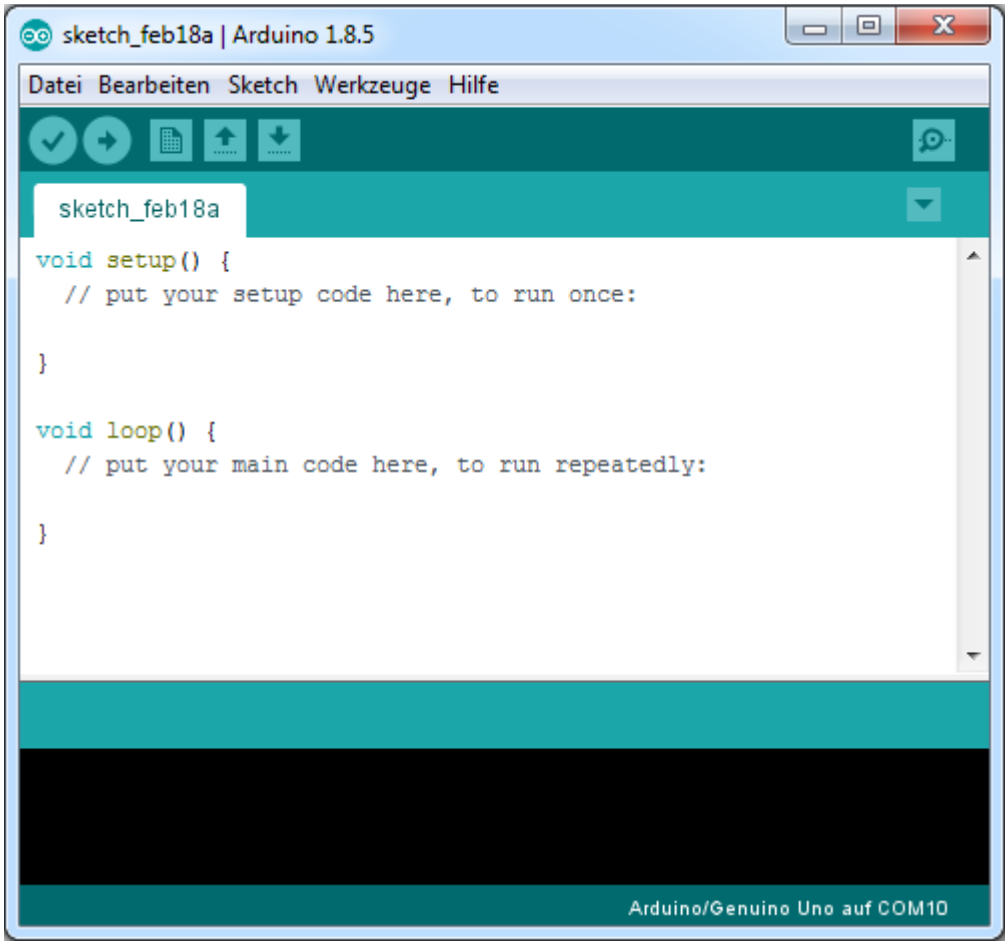


Mit Close wird der Installer anschließend beendet und im Startmenü und Desktop befindet sich ein neues Symbol. Dieses starten wir jetzt:



Es startet die Arduino Software:

Und das Programmierfenster erscheint:



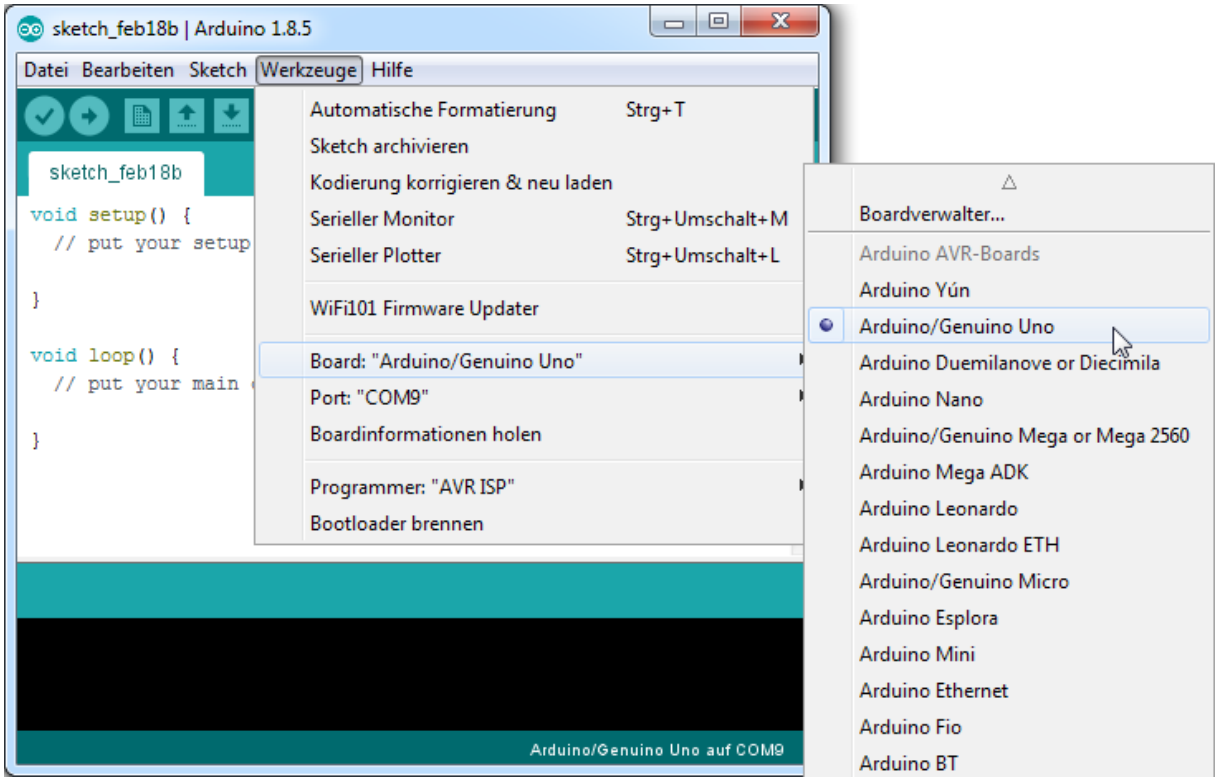
Jetzt können wir mit dem Programmieren beginnen.

Erste Schritte in der Arduino Programmiersoftware

Bevor wir mit dem Sensorkit beginnen können, müssen wir in der Software auch unseren Arduino (den du separat bei uns bestellen kannst) definieren.

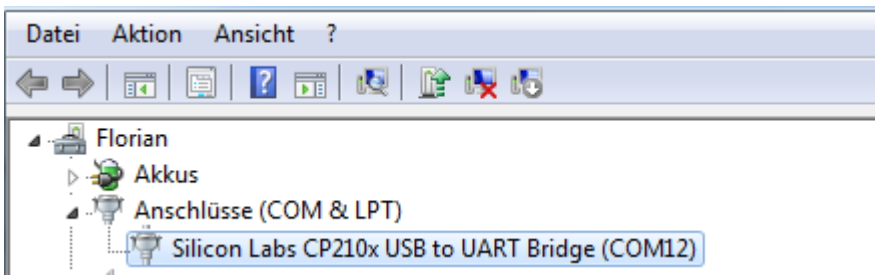
Dazu wählen wir in der Software:

Werkzeuge > Board: > {Hier deinen Arduino auswählen} Arduino Uno



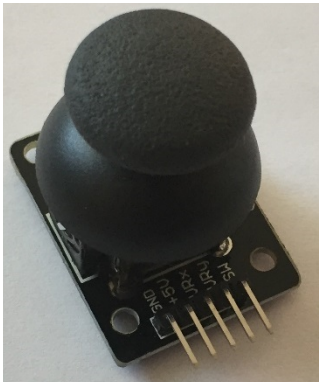
In der Anleitung verwenden wir einen Arduino Uno. Aber auch andere Arduino funktionieren.

Bei Port musst du nur noch den Com-Port deines Arduino eintragen, diesen kannst du beim Gerätemanager auslesen und ggf. auch abändern.

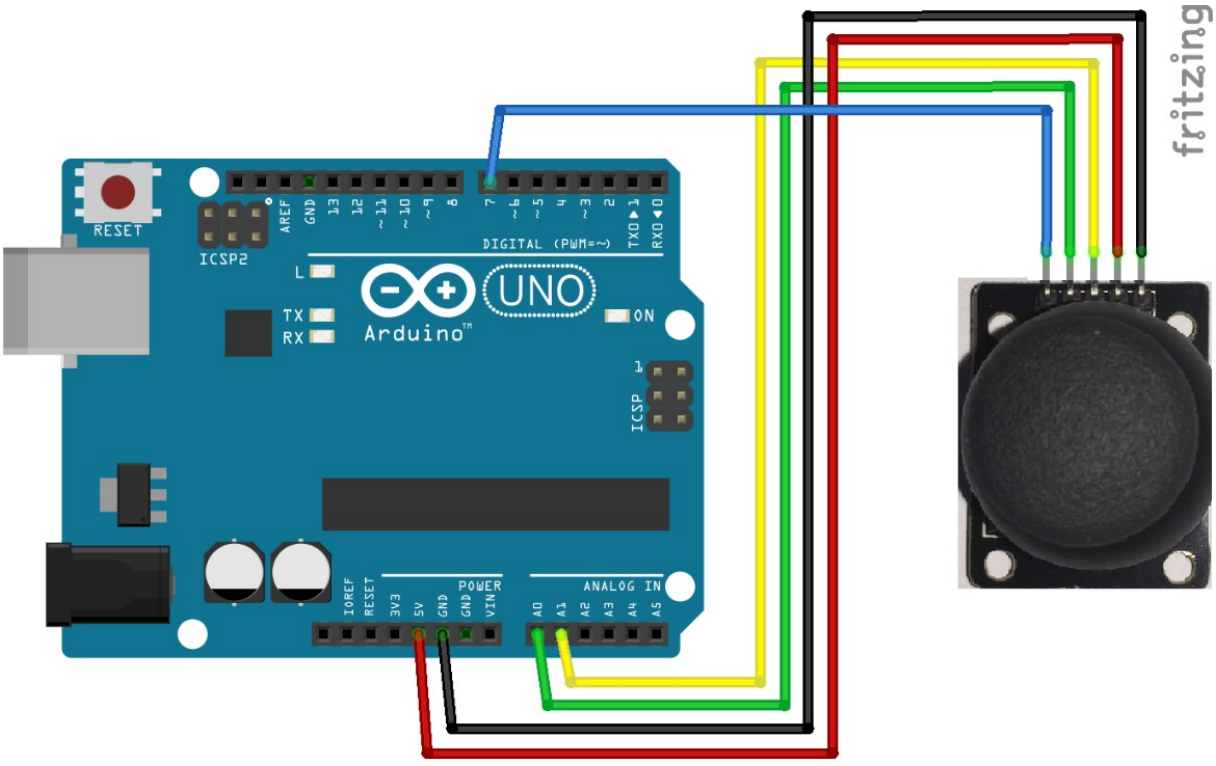


Das waren die ersten Grundeinstellungen, nun können wir mit dem Programmieren beginnen.

1. Joystick



Verdrahten des Joysticks

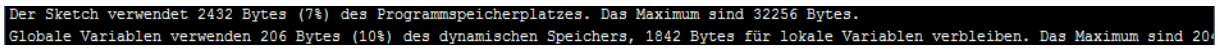


+5V wird mit **5V** am Arduino verbunden	Rote Leitung
GND wird mit **GND** verbunden	Schwarze Leitung
VRx wird mit **A1** verbunden	Gelbe Leitung
VRy wird mit **A0** verbunden	Grüne Leitung
MS wird mit **D7** verbunden	Blaue Leitung

Software für den Joystick

```
int value = 0;

void setup() {
pinMode(7, INPUT_PULLUP);
Serial.begin(9600);
}

void loop() {
value = analogRead(0);
Serial.print("X:");
Serial.print(value, DEC);
value = analogRead(1);
Serial.print(" | Y:");
Serial.print(value, DEC);
value = !digitalRead(7);       // ! Invertiert wegen Pullup
Serial.print(" | Z: ");
Serial.println(value, DEC);
delay(100);
}
```

Nachdem wir den Code geschrieben haben klicken wir oben auf  und Verifizieren unser Programm:

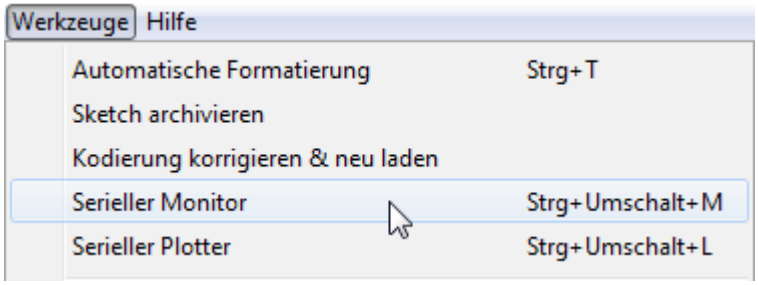


Wenn alles stimmt und unser Programm keine Fehler enthält können wir es auf den Arduino hochladen. Dazu klicken wir oben auf  .

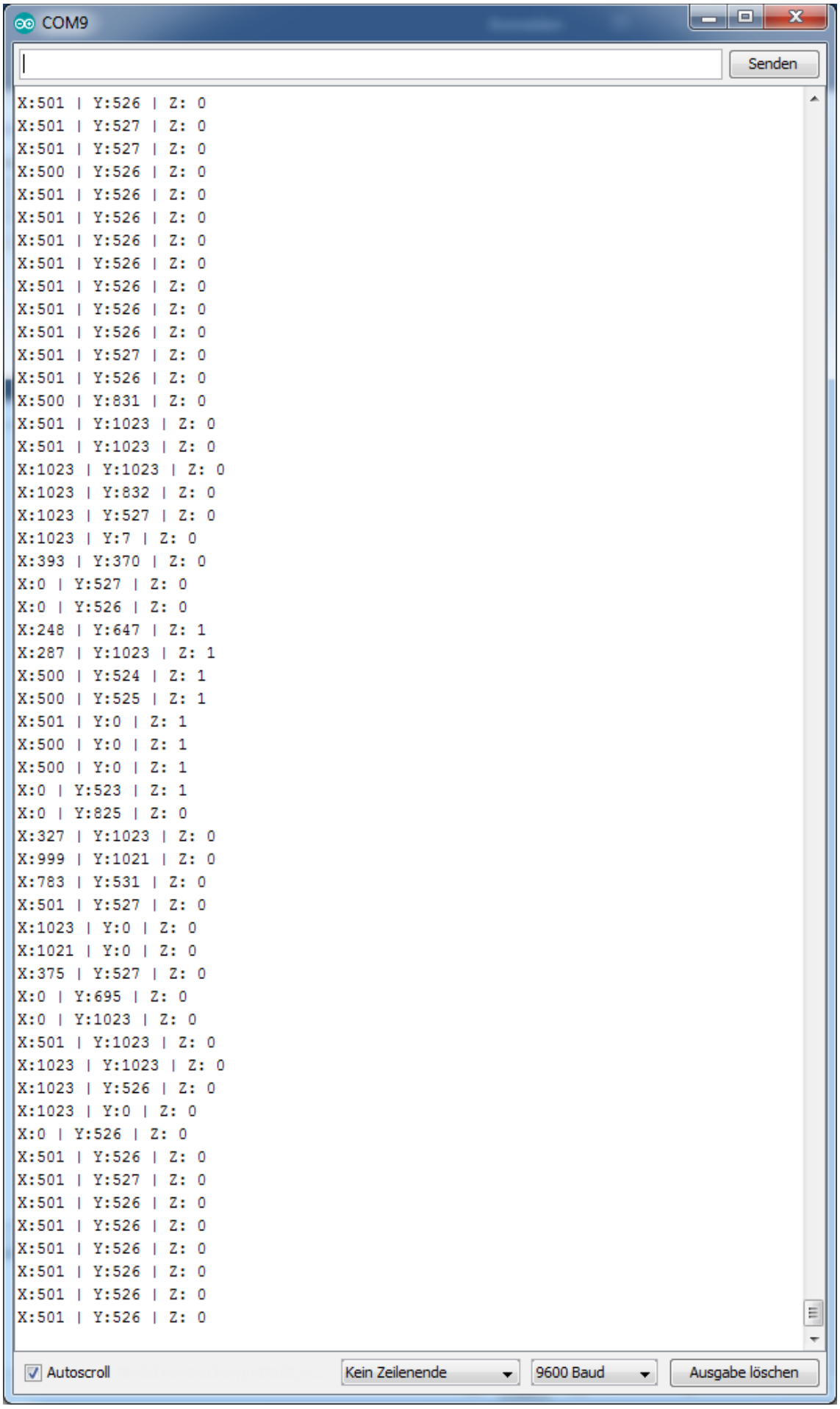
Kurz darauf kommt dann: 

Jetzt starten wir den Serial Monitor in der Arduino Software:

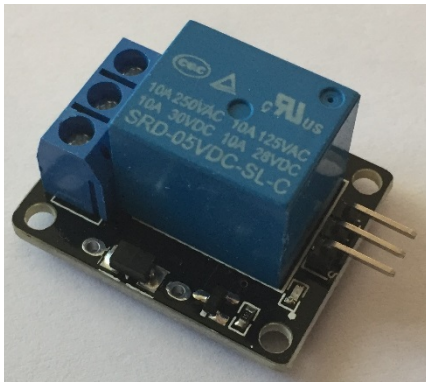
Werkzeuge > Serial Monitor



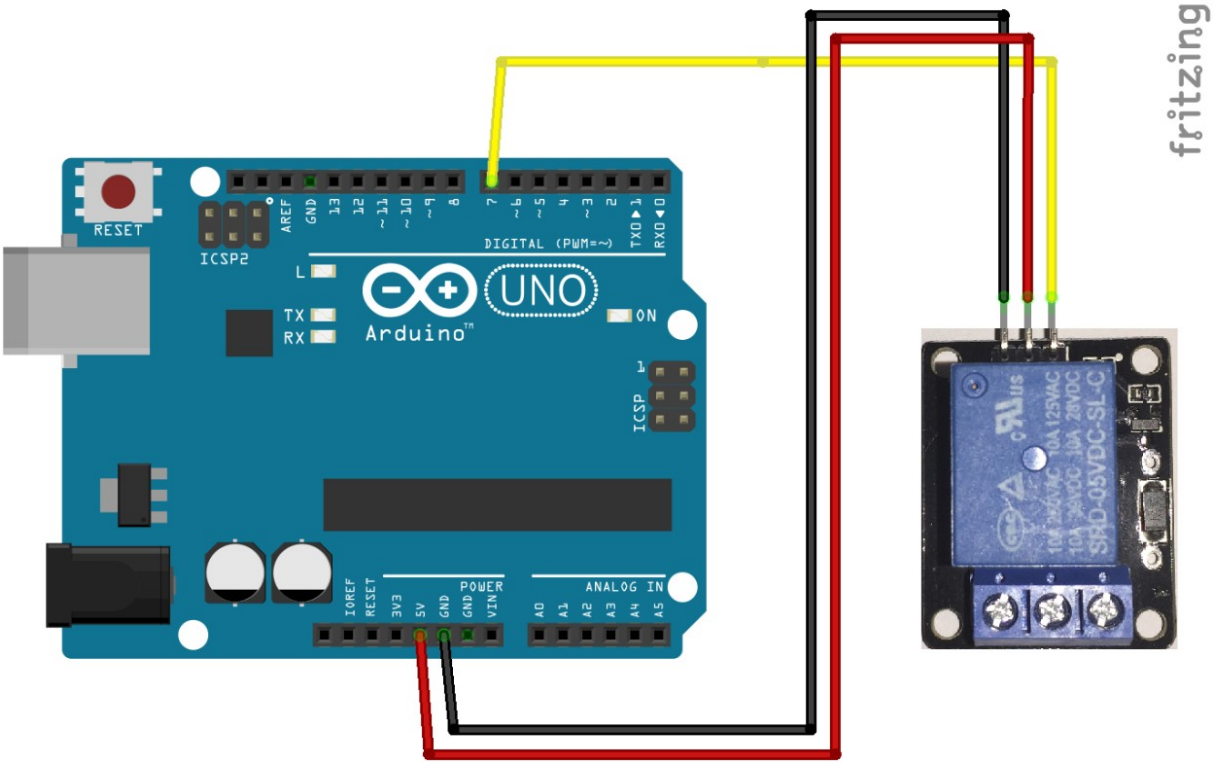
Nach dem öffnen muss evtl. unten rechts noch die Baudrate auf 9600 Baud umgestellt werden und schon bekommen wir die Werte unseres Joysticks:


COM9
Senden
X:501 | Y:526 | Z: 0
X:501 | Y:527 | Z: 0
X:501 | Y:527 | Z: 0
X:500 | Y:526 | Z: 0
X:501 | Y:526 | Z: 0
X:501 | Y:526 | Z: 0
X:501 | Y:526 | Z: 0
X:501 | Y:526 | Z: 0
X:501 | Y:526 | Z: 0
X:501 | Y:526 | Z: 0
X:501 | Y:526 | Z: 0
X:501 | Y:526 | Z: 0
X:501 | Y:527 | Z: 0
X:501 | Y:526 | Z: 0
X:500 | Y:831 | Z: 0
X:501 | Y:1023 | Z: 0
X:501 | Y:1023 | Z: 0
X:1023 | Y:1023 | Z: 0
X:1023 | Y:832 | Z: 0
X:1023 | Y:527 | Z: 0
X:1023 | Y:7 | Z: 0
X:393 | Y:370 | Z: 0
X:0 | Y:527 | Z: 0
X:0 | Y:526 | Z: 0
X:248 | Y:647 | Z: 1
X:287 | Y:1023 | Z: 1
X:500 | Y:524 | Z: 1
X:500 | Y:525 | Z: 1
X:501 | Y:0 | Z: 1
X:500 | Y:0 | Z: 1
X:500 | Y:0 | Z: 1
X:0 | Y:523 | Z: 1
X:0 | Y:825 | Z: 0
X:327 | Y:1023 | Z: 0
X:999 | Y:1021 | Z: 0
X:783 | Y:531 | Z: 0
X:501 | Y:527 | Z: 0
X:1023 | Y:0 | Z: 0
X:1021 | Y:0 | Z: 0
X:375 | Y:527 | Z: 0
X:0 | Y:695 | Z: 0
X:0 | Y:1023 | Z: 0
X:501 | Y:1023 | Z: 0
X:1023 | Y:1023 | Z: 0
X:1023 | Y:526 | Z: 0
X:1023 | Y:0 | Z: 0
X:0 | Y:526 | Z: 0
X:501 | Y:526 | Z: 0
X:501 | Y:527 | Z: 0
X:501 | Y:526 | Z: 0
X:501 | Y:526 | Z: 0
X:501 | Y:526 | Z: 0
X:501 | Y:526 | Z: 0
X:501 | Y:526 | Z: 0
X:501 | Y:526 | Z: 0
Autoscroll
Kein Zeilenende
9600 Baud
Ausgabe löschen

2. Relais



Verdrahten des Relais



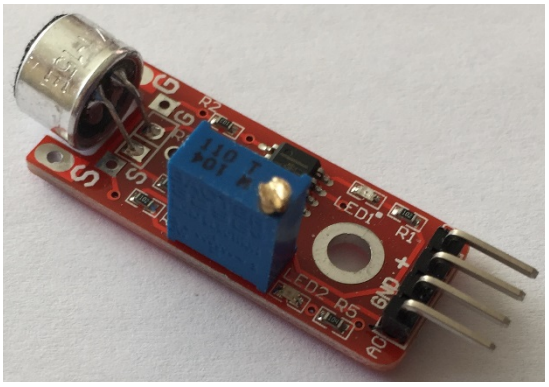
+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
S wird mit **D7** verbunden — Gelbe Leitung

Software für das Relais

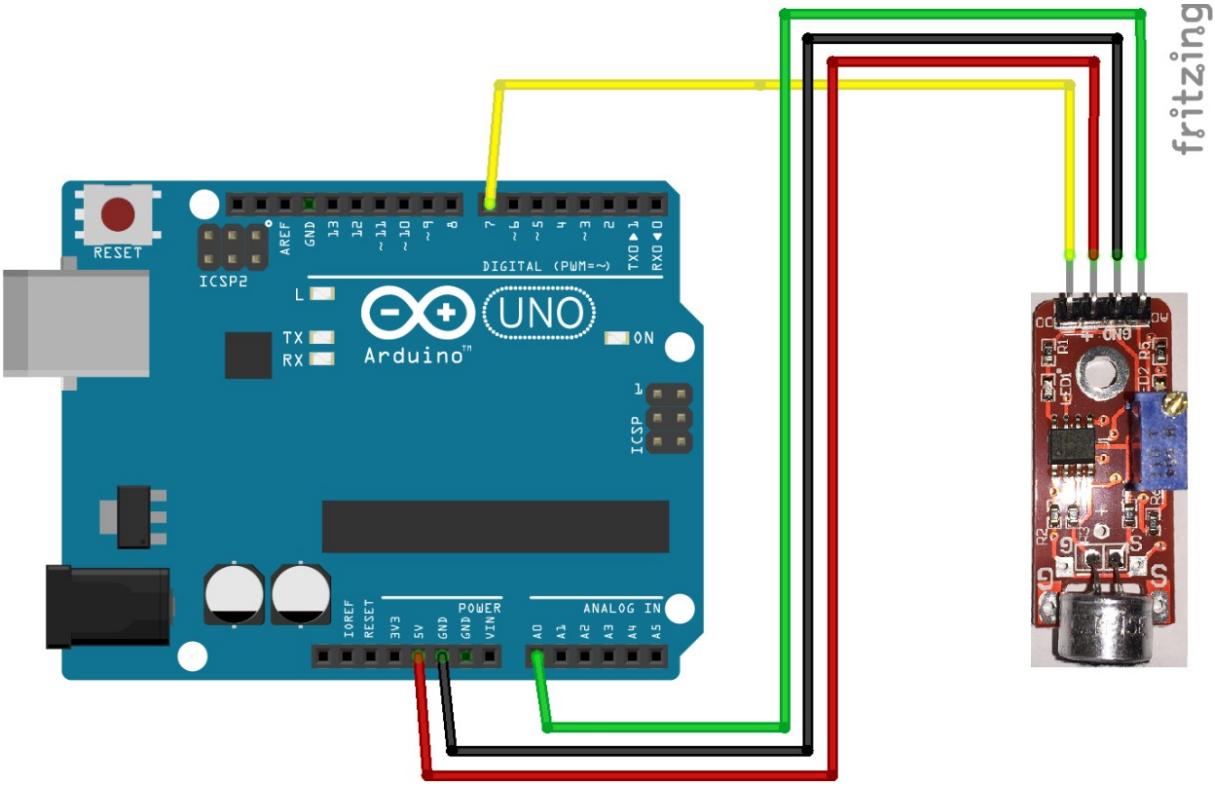
```
int Relais = 7;

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

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

Der Code wird wieder Verifiziert  und Hochgeladen  . Das Relais beginnt nun im Sekundentakt ein und auszuschalten.

3. Großes Mikrofonmodul



Verdrahten des Moduls

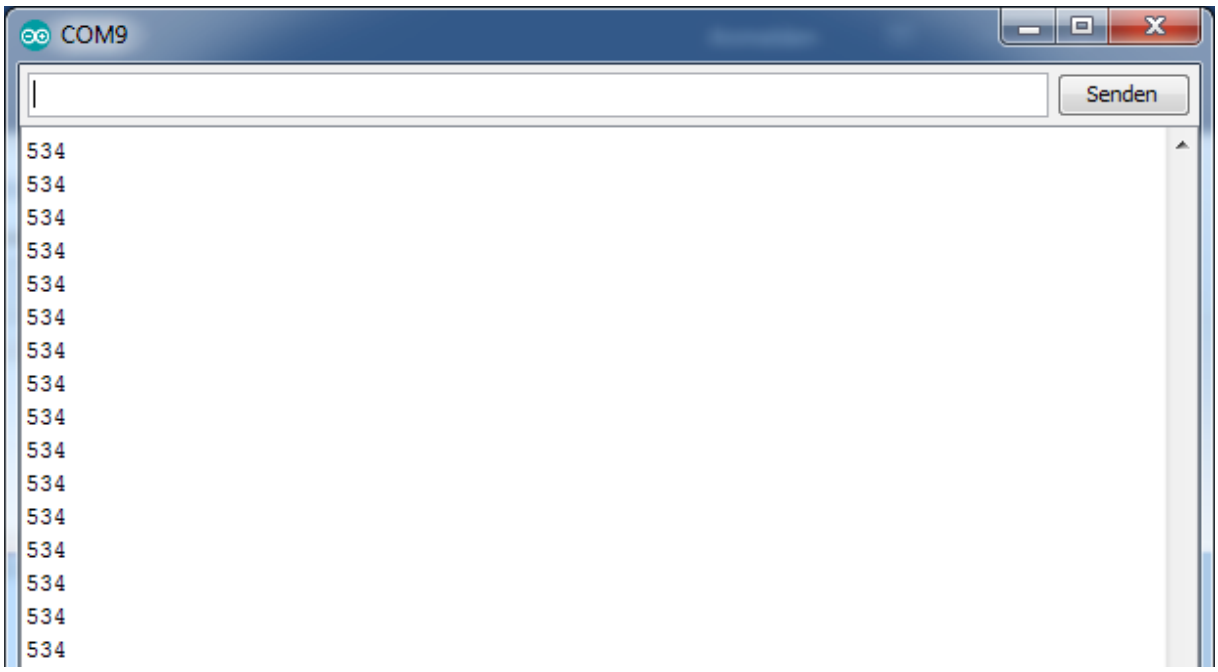


+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
DO wird mit **D7** verbunden — Gelbe Leitung
AO wird mit **A0** verbunden — Grüne Leitung

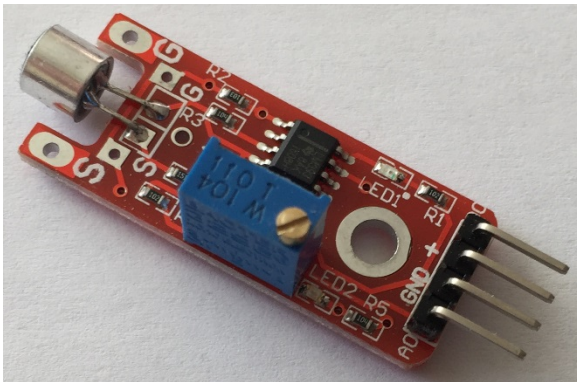
Software für das Große Mikrofon:

```
int Led=LED_BUILTIN;
int Mikrofon=7;
int PotiPin = A0;
int PotiValue = 0;
int val;

void setup()
{
Serial.begin(9600);
pinMode(Led,OUTPUT);
pinMode(Mikrofon,INPUT);
}

void loop()
{
PotiValue = analogRead(PotiPin);
Serial.println(PotiValue, DEC);
val=digitalRead(Mikrofon);
if(val==HIGH)
  {
    digitalWrite(Led,HIGH);
  }
  else
  {
    digitalWrite(Led,LOW);
  }
}
```

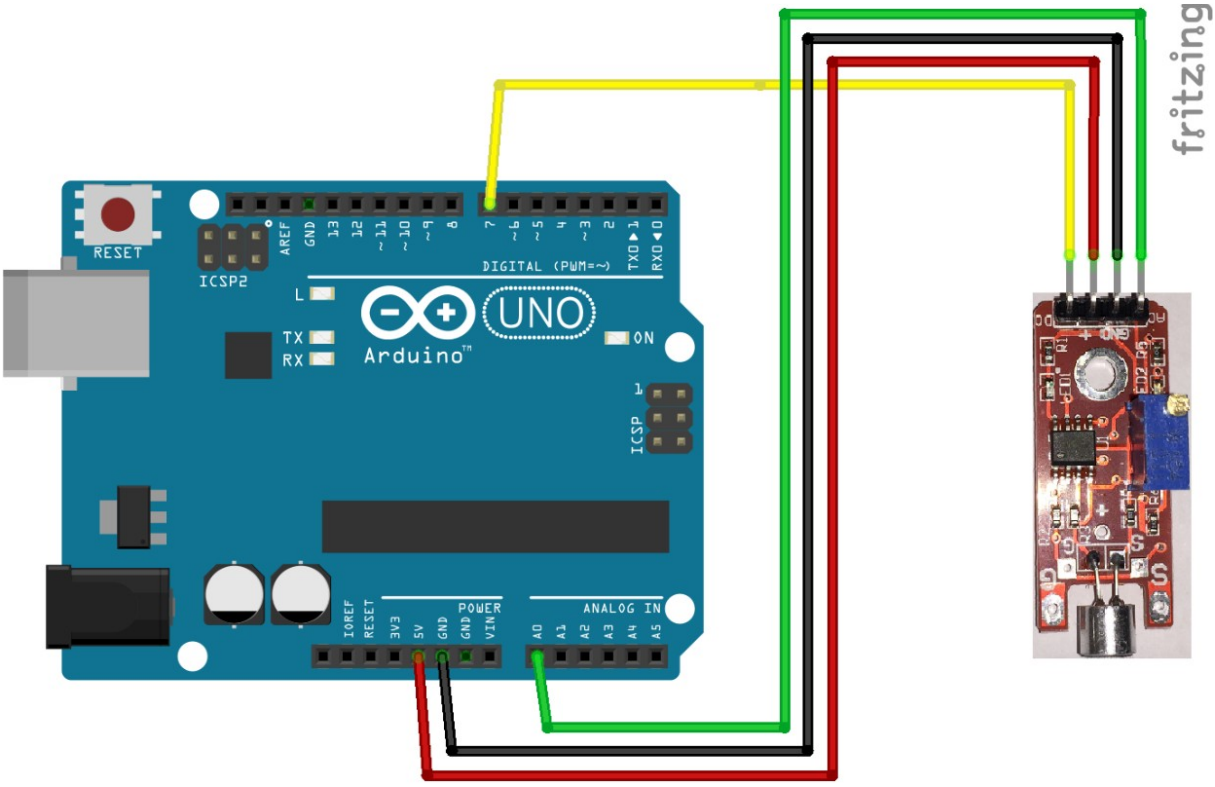
Der Code wird wieder Verifiziert und Hochgeladen. Im SerialMonitor wird nun der Wert des Potentiometers ausgegeben. Wird am Mikrofon ein stärkeres Signal aufgenommen, so leuchtet die eingebaute LED am Arduino. Mit dem Mikrofon kann man nun einen Klatschsensor bauen.



4. Kleines Mikrofonmodul



Verdrahten des Moduls

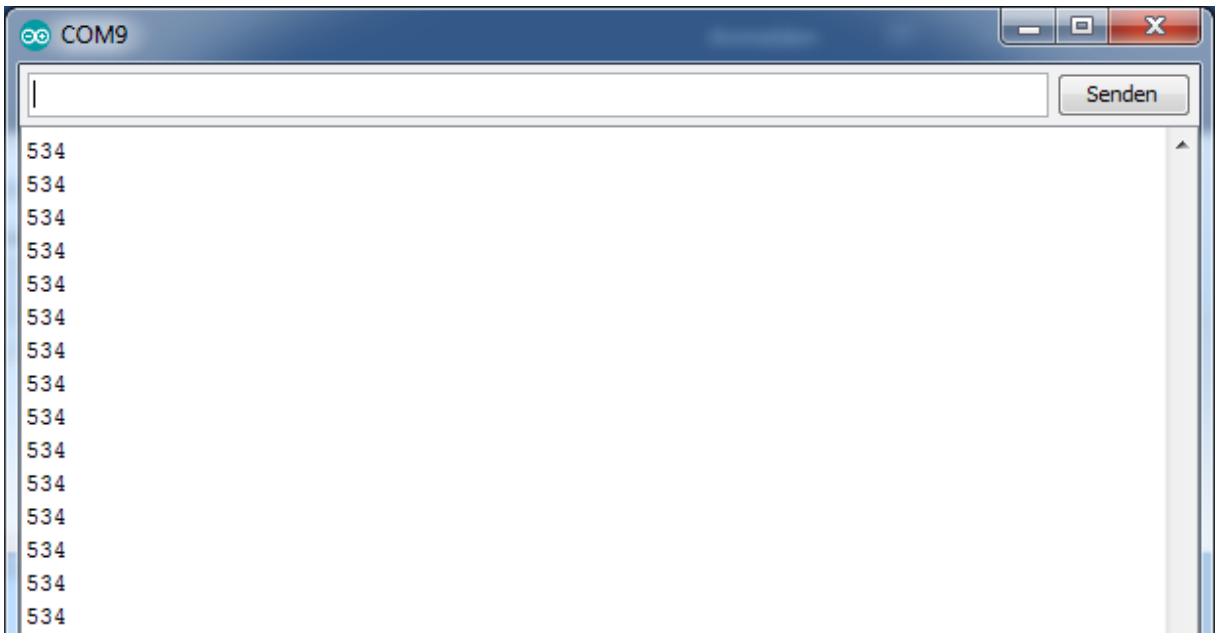


+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
DO wird mit **D7** verbunden — Gelbe Leitung
AO wird mit **A0** verbunden — Grüne Leitung

Software für das kleine Mikrofon:

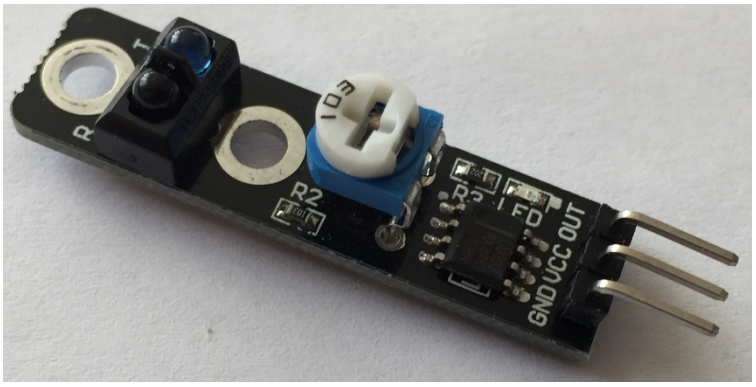
```
int Led=LED_BUILTIN;
int Mikrofon=7;
int PotiPin = A0;
int PotiValue = 0;
int val;

void setup()
{
Serial.begin(9600);
pinMode(Led,OUTPUT);
pinMode(Mikrofon,INPUT);
}

void loop()
{
PotiValue = analogRead(PotiPin);
Serial.println(PotiValue, DEC);
val=digitalRead(Mikrofon);
if(val==HIGH)
  {
    digitalWrite(Led,HIGH);
  }
  else
  {
    digitalWrite(Led,LOW);
  }
}
```

Der Code wird wieder Verifiziert und Hochgeladen. Im SerialMonitor wird nun der Wert des Potentiometers ausgegeben. Wird am Mikrofon ein stärkeres Signal aufgenommen, so leuchtet die eingebaute LED am Arduino. Mit dem Mikrofon kann man nun einen Klatschsensor bauen.

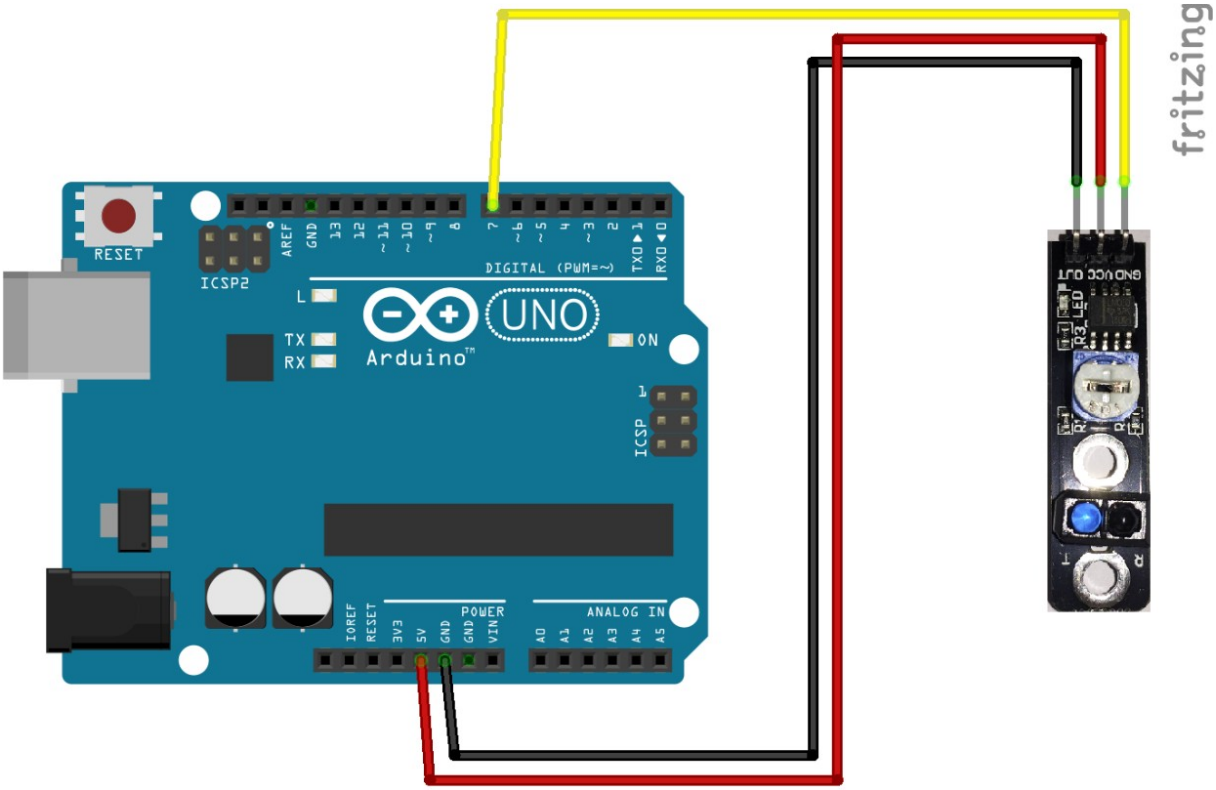


Das große und kleine Mikrofon sind identisch, sie unterscheiden sich nur von der Mikrofonkapsel, das große hat eine größere Empfindlichkeit (electret microphone).

5. Linienfolger Modul



Verdrahten des Moduls



VCC wird mit **5V** am Arduino verbunden — Rote Leitung
GND wird mit **GND** verbunden — Schwarze Leitung
OUT wird mit **D7** verbunden — Gelbe Leitung

Software für den Linienfolger:

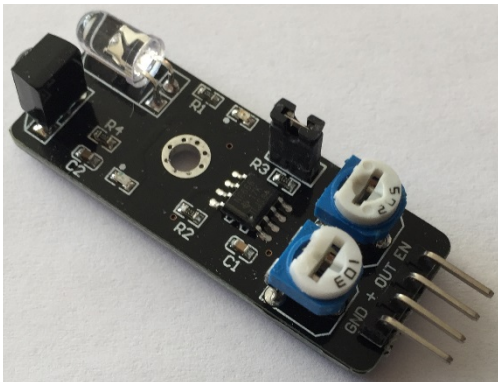
```
int Led=LED_BUILTIN;
int Sensor=7;
int val;

void setup()
{
pinMode(Led,OUTPUT);
pinMode(Sensor,INPUT);
}

void loop()
{
val=digitalRead(Sensor);
digitalWrite(Led,val);
}
```

Der Code wird wieder Verifiziert  und Hochgeladen  .

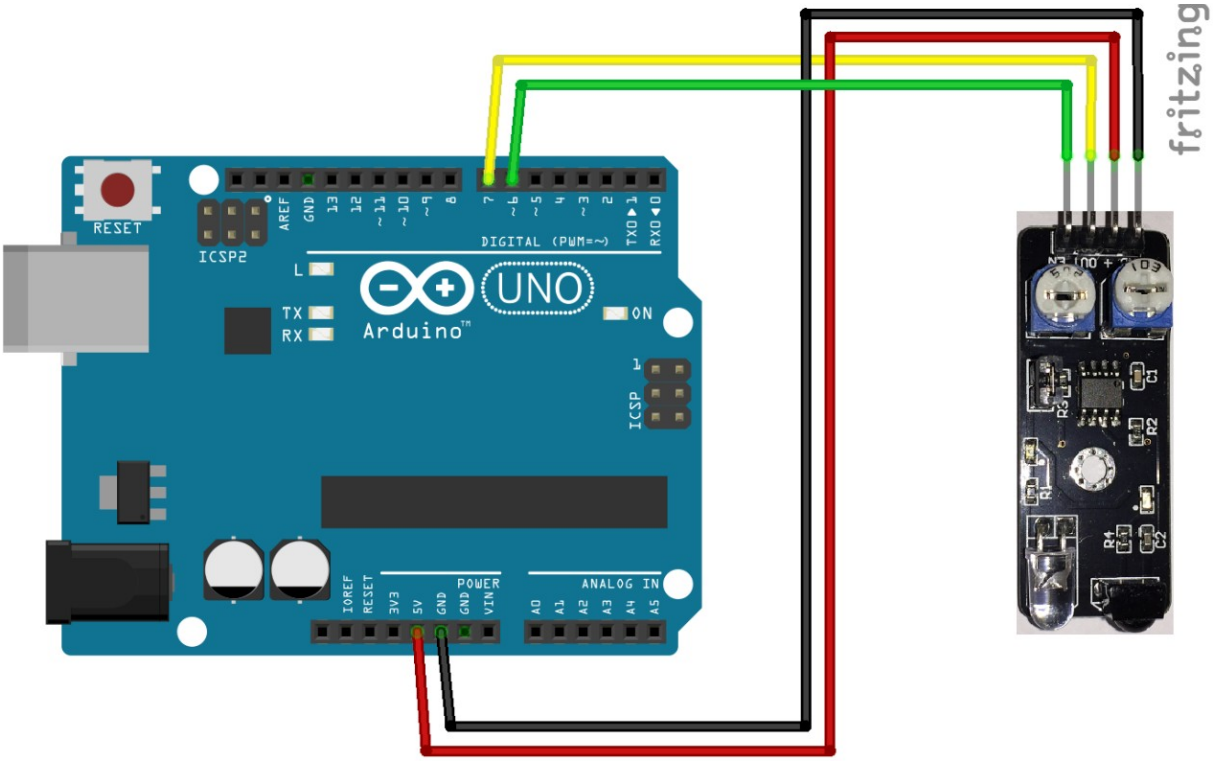
Der Linienfolger folgt einer dunklen (schwarzen) Linie. Der Sensor gibt immer dann ein Signal aus, wenn er die Linie verlassen hat.

Diesen Sensor kann man in einen Roboter einbauen.

6. Hindernissensor



Verdrahten des Moduls



VCC wird mit **5V** am Arduino verbunden — Rote Leitung
GND wird mit **GND** verbunden — Schwarze Leitung
OUT wird mit **D7** verbunden — Gelbe Leitung
EN wird mit **D6** verbunden — Grüne Leitung

Software für den Hindernissensor:

```
int Led=LED_BUILTIN;
int Sensor=7;
int Enable=6;
int val;

void setup()
{
pinMode(Led,OUTPUT);
pinMode(Sensor,INPUT);
pinMode(Enable,OUTPUT);
digitalWrite(Enable,HIGH);      //ohne Funktion
}

void loop()
{
val=digitalRead(Sensor);
digitalWrite(Led,val);
}
```

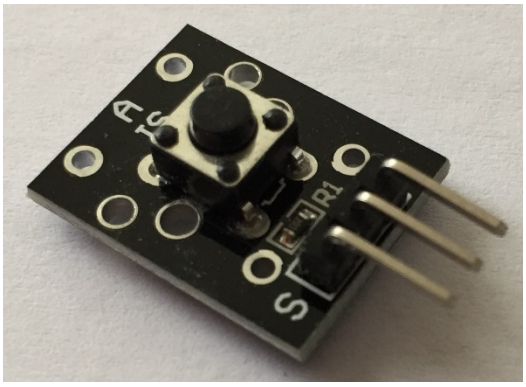
Der Code wird wieder Verifiziert  und Hochgeladen  .

Der Hindernissensor gibt ein Signal, wenn sich ein Gegenstand nähert.

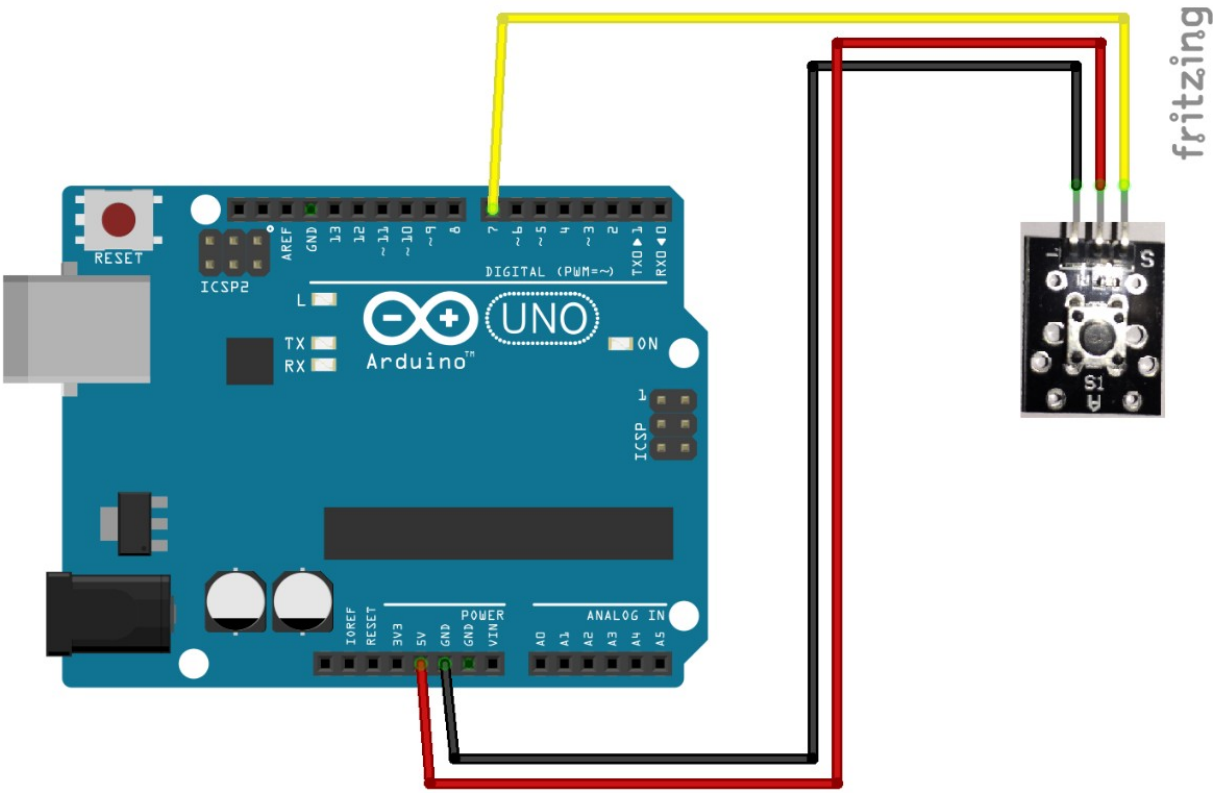
Diesen Sensor kann man in einen Roboter einbauen.

Der Pin EN (Enable) kann auch nicht angeschlossen werden, dieser hat keinerlei Auswirkung auf den Sensor.

7. Taster



Verdrahten des Moduls



+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
S wird mit **D7** verbunden — Gelbe Leitung

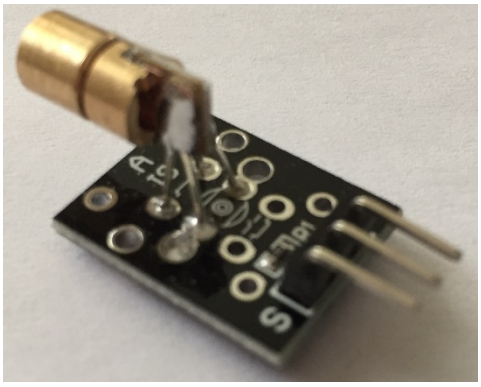
Software für den Taster:

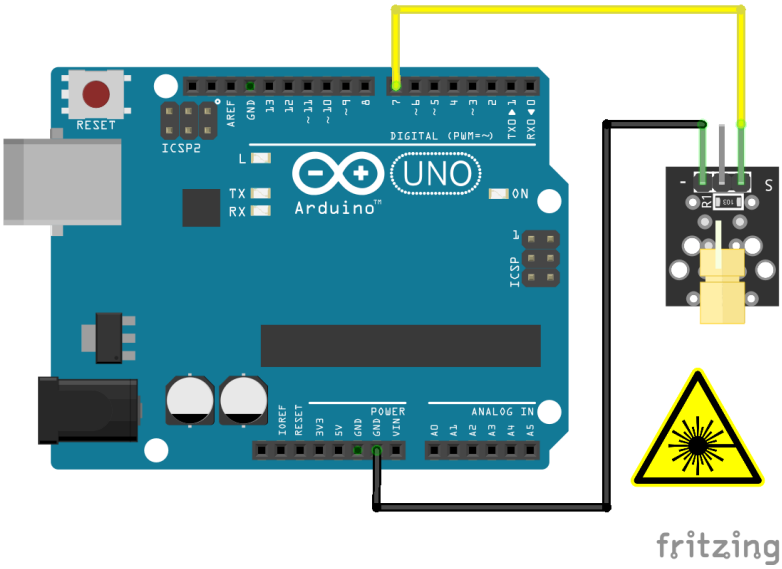
```
int Led = LED_BUILTIN;
int Taster = 7;
int value;

void setup ()
{
pinMode (Led, OUTPUT);
pinMode (Taster, INPUT);
}

void loop ()
{
value=digitalRead(Taster);
digitalWrite (Led, !value);
}
```

Der Code wird wieder Verifiziert  und Hochgeladen  .

Sobald der Taster betätigt wird, beginnt die LED am Arduino zu leuchten.

8. Laser





Verdrahten des Moduls

- wird mit **GND** verbunden — Schwarze Leitung
S wird mit **D7** verbunden — Gelbe Leitung

ACHTUNG:

Nicht direkt in den Laser sehen!!!

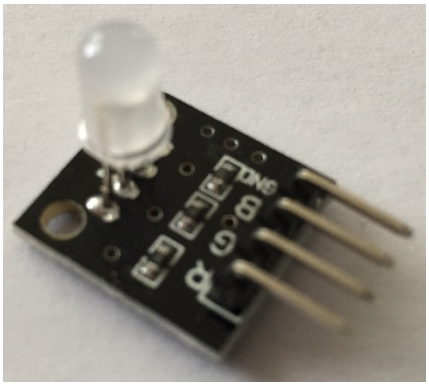
Software für den Laser:

```
int Laser = 7;

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

void loop() {
digitalWrite(Laser, HIGH);
delay(100);
digitalWrite(Laser, LOW);
delay(100);
}
```

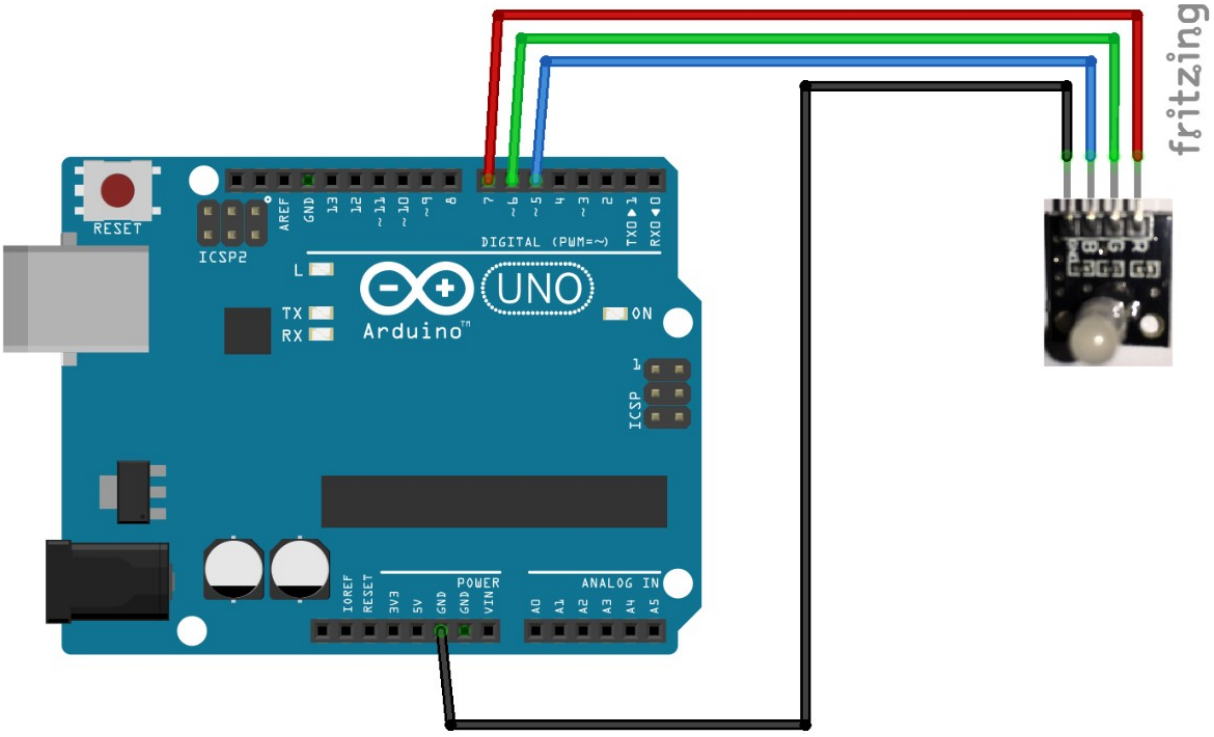
Der Code wird wieder Verifiziert  und Hochgeladen  .

Die Laserdiode blinkt nun mit 5Hz.

9. RGB-LED



Verdrahten des Moduls



GND wird mit **GND** verbunden	Schwarze Leitung
R wird mit **D7** verbunden	Rote Leitung
G wird mit **D6** verbunden	Grüne Leitung
B wird mit **D5** verbunden	Blaue Leitung

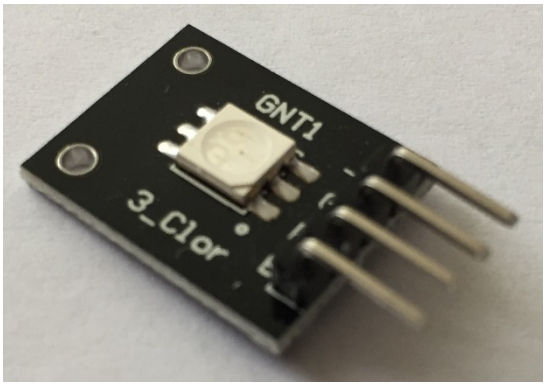
Software für die RGB-LED:

```
int ROT =  7;
int GRUEN = 6;
int BLAU = 5;

void setup(){
  pinMode(ROT, OUTPUT);
  pinMode(GRUEN, OUTPUT);
  pinMode(BLAU, OUTPUT);
}

void loop(){
  digitalWrite(ROT, HIGH);
  digitalWrite(GRUEN, LOW);
  digitalWrite(BLAU, LOW);
delay(500);
  digitalWrite(ROT, LOW);
  digitalWrite(GRUEN, HIGH);
  digitalWrite(BLAU, LOW);
delay(500);
  digitalWrite(ROT, LOW);
  digitalWrite(GRUEN, LOW);
  digitalWrite(BLAU, HIGH);
delay(500);

  digitalWrite(ROT, HIGH);
  digitalWrite(GRUEN, HIGH);
  digitalWrite(BLAU, LOW);
delay(500);
  digitalWrite(ROT, LOW);
  digitalWrite(GRUEN, HIGH);
  digitalWrite(BLAU, HIGH);
delay(500);
  digitalWrite(ROT, HIGH);
  digitalWrite(GRUEN, HIGH);
  digitalWrite(BLAU, HIGH);
delay(500);
  digitalWrite(ROT, HIGH);
  digitalWrite(GRUEN, LOW);
  digitalWrite(BLAU, HIGH);
delay(500);
}
```

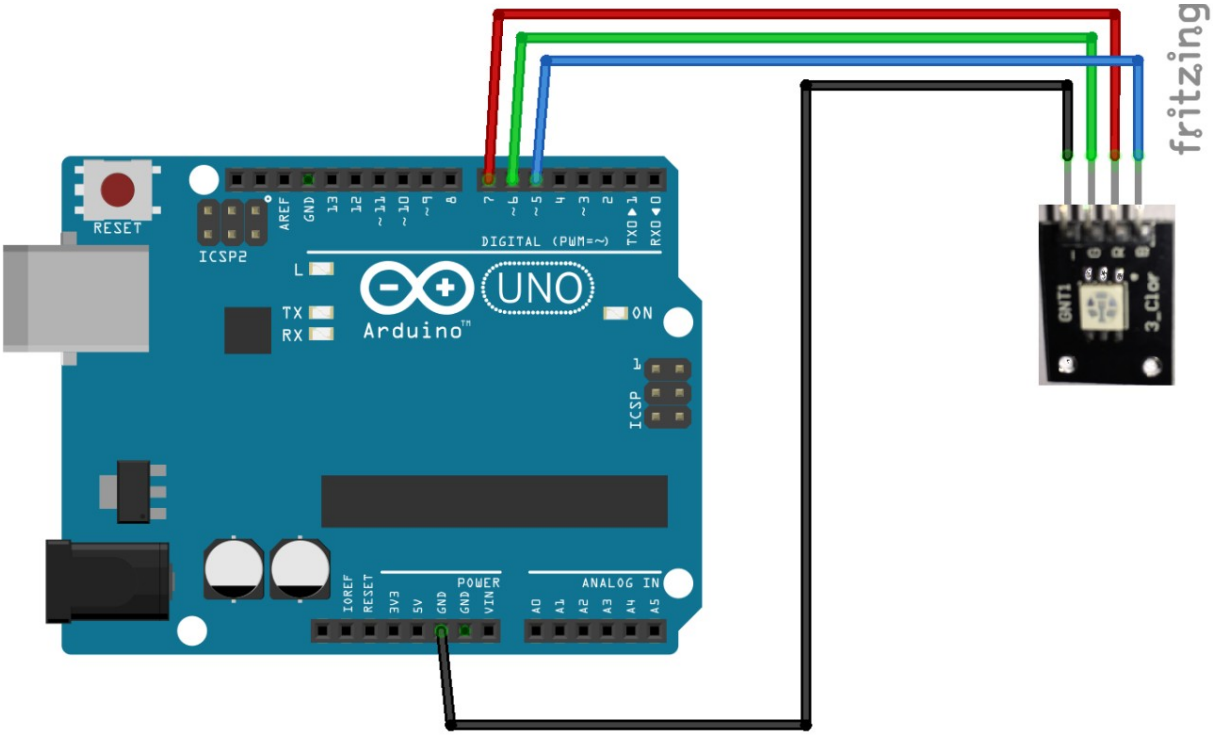
Der Code wird wieder Verifiziert  und Hochgeladen  .

Die RGB-LED wird nun durch 3 Pins angesteuert und zeigt verschiedene Farben abwechselnd an.

10. SMD RGB-LED



Verdrahten des Moduls



GND wird mit **GND** verbunden — Schwarze Leitung
R wird mit **D7** verbunden — Rote Leitung
G wird mit **D6** verbunden — Grüne Leitung
B wird mit **D5** verbunden — Blaue Leitung

Software für die RGB-LED:

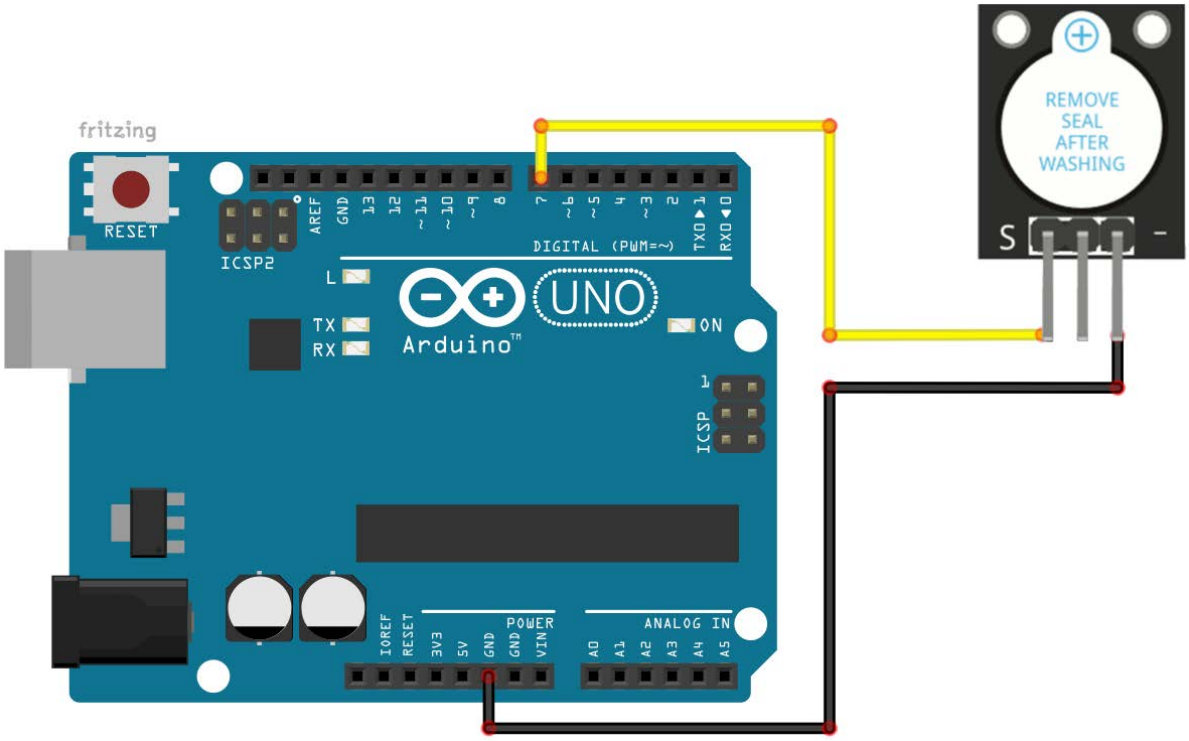
```
int ROT =  7;
int GRUEN = 6;
int BLAU = 5;

void setup(){
  pinMode(ROT, OUTPUT);
  pinMode(GRUEN, OUTPUT);
  pinMode(BLAU, OUTPUT);
}

void loop(){
  digitalWrite(ROT, HIGH);
  digitalWrite(GRUEN, LOW);
  digitalWrite(BLAU, LOW);
delay(500);
  digitalWrite(ROT, LOW);
  digitalWrite(GRUEN, HIGH);
  digitalWrite(BLAU, LOW);
delay(500);
  digitalWrite(ROT, LOW);
  digitalWrite(GRUEN, LOW);
  digitalWrite(BLAU, HIGH);
delay(500);

  digitalWrite(ROT, HIGH);
  digitalWrite(GRUEN, HIGH);
  digitalWrite(BLAU, LOW);
delay(500);
  digitalWrite(ROT, LOW);
  digitalWrite(GRUEN, HIGH);
  digitalWrite(BLAU, HIGH);
delay(500);
  digitalWrite(ROT, HIGH);
  digitalWrite(GRUEN, HIGH);
  digitalWrite(BLAU, HIGH);
delay(500);
  digitalWrite(ROT, HIGH);
  digitalWrite(GRUEN, LOW);
  digitalWrite(BLAU, HIGH);
delay(500);
}
```

Der Code wird wieder Verifiziert  und Hochgeladen  .

Die RGB-LED wird nun durch 3 Pins angesteuert und zeigt verschiedene Farben abwechselnd an.

11. Aktiver Buzzer

Verdrahten des Moduls



- wird mit **GND** verbunden — Schwarze Leitung
S wird mit **D7** verbunden — Gelbe Leitung

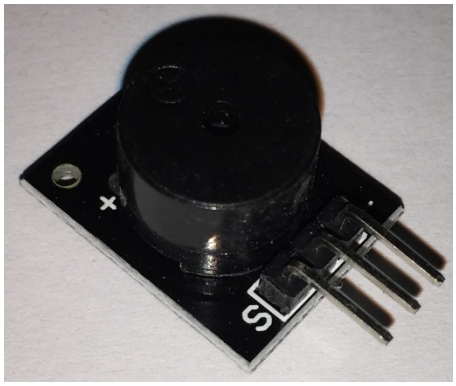
Software für den Aktiven Buzzer:

```
int Buzzer = 7;

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

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

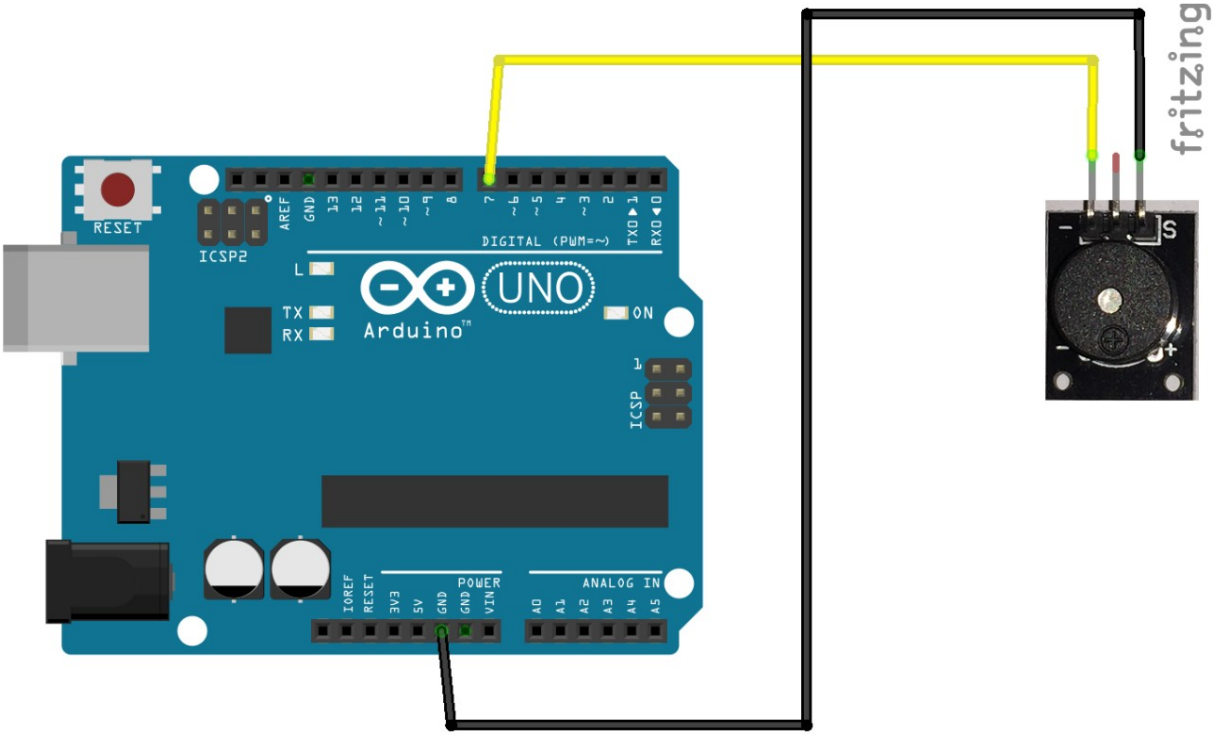
Der Code wird wieder Verifiziert  und Hochgeladen  .

Der Buzzer Piept immer für 1 Sekunde und macht dann eine Pause. Du kannst diesen Buzzer als Bestätigungstongeber verwenden.

12. Passiver Buzzer



Verdrahten des Moduls



- wird mit **GND** verbunden — Schwarze Leitung
S wird mit **D10** verbunden — Gelbe Leitung

Software für den Passiven Buzzer:

```
int Buzzer = 7;

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

void loop() {
digitalWrite(Buzzer, HIGH);
delay(1);
digitalWrite(Buzzer, LOW);
delay(1);
}
```

Der Code wird wieder Verifiziert  und Hochgeladen  .

Der Buzzer gibt nun einen Ton mit ca. 500Hz aus. Dieser Buzzer ist ähnlich wie ein Lautsprecher.

Musik mit dem Passiven Buzzer:

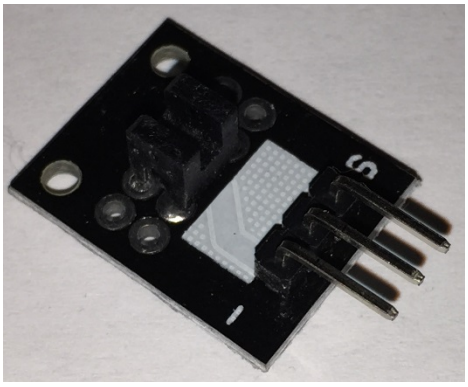
Der Arduino kann auch mit PWM Musik abspielen, anbei ein kleines Beispiel mit Pirates of the Caribbean Song:

https://github.com/xitangg/-Pirates-of-the-Caribbean-Theme-Song

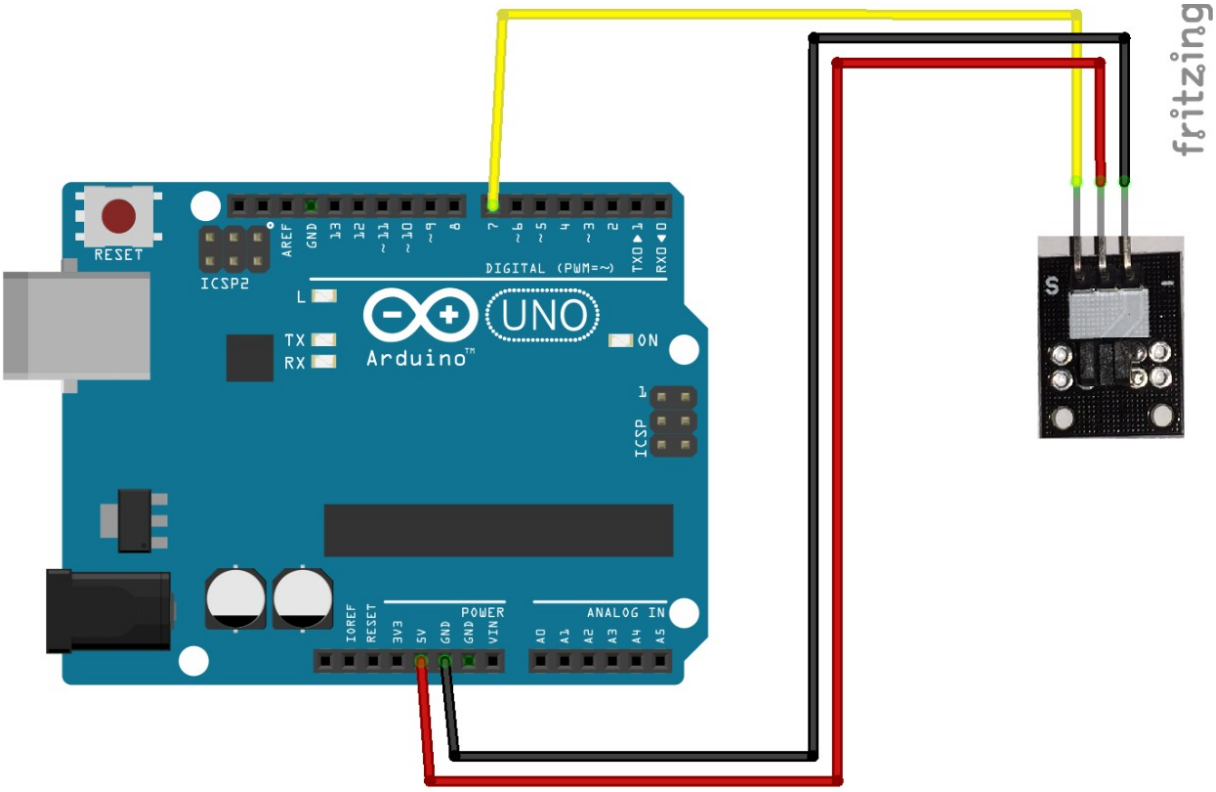
```
const int buzzer = 10;
const int songspeed = 1.0;
//*****************************************
#define NOTE_C4  262
#define NOTE_D4  294
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_G4  392
#define NOTE_A4  440
#define NOTE_B4  494
#define NOTE_C5  523
#define NOTE_D5  587
#define NOTE_E5  659
#define NOTE_F5  698
#define NOTE_G5  784
#define NOTE_A5  880
#define NOTE_B5  988
//*****************************************
int notes[] = {
   NOTE_E4, NOTE_G4, NOTE_A4, NOTE_A4, 0,
   NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
   NOTE_C5, NOTE_D5, NOTE_B4, NOTE_B4, 0,
   NOTE_A4, NOTE_G4, NOTE_A4, 0,
   NOTE_E4, NOTE_G4, NOTE_A4, NOTE_A4, 0,
   NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
   NOTE_C5, NOTE_D5, NOTE_B4, NOTE_B4, 0,
   NOTE_A4, NOTE_G4, NOTE_A4, 0,
   NOTE_E4, NOTE_G4, NOTE_A4, NOTE_A4, 0,
   NOTE_A4, NOTE_C5, NOTE_D5, NOTE_D5, 0,
   NOTE_D5, NOTE_E5, NOTE_F5, NOTE_F5, 0,
   NOTE_E5, NOTE_D5, NOTE_E5, NOTE_A4, 0,
   NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
   NOTE_D5, NOTE_E5, NOTE_A4, 0,
   NOTE_A4, NOTE_C5, NOTE_B4, NOTE_B4, 0,
   NOTE_C5, NOTE_A4, NOTE_B4, 0,
```

```
   NOTE_A4, NOTE_A4,
   NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
   NOTE_C5, NOTE_D5, NOTE_B4, NOTE_B4, 0,
   NOTE_A4, NOTE_G4, NOTE_A4, 0,
   NOTE_E4, NOTE_G4, NOTE_A4, NOTE_A4, 0,
   NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
   NOTE_C5, NOTE_D5, NOTE_B4, NOTE_B4, 0,
   NOTE_A4, NOTE_G4, NOTE_A4, 0,
   NOTE_E4, NOTE_G4, NOTE_A4, NOTE_A4, 0,
   NOTE_A4, NOTE_C5, NOTE_D5, NOTE_D5, 0,
   NOTE_D5, NOTE_E5, NOTE_F5, NOTE_F5, 0,
   NOTE_E5, NOTE_D5, NOTE_E5, NOTE_A4, 0,
   NOTE_A4, NOTE_B4, NOTE_C5, NOTE_C5, 0,
   NOTE_D5, NOTE_E5, NOTE_A4, 0,
   NOTE_A4, NOTE_C5, NOTE_B4, NOTE_B4, 0,
   NOTE_C5, NOTE_A4, NOTE_B4, 0,
   NOTE_E5, 0, 0, NOTE_F5, 0, 0,
   NOTE_E5, NOTE_E5, 0, NOTE_G5, 0, NOTE_E5, NOTE_D5, 0, 0,
   NOTE_D5, 0, 0, NOTE_C5, 0, 0,
   NOTE_B4, NOTE_C5, 0, NOTE_B4, 0, NOTE_A4,
   NOTE_E5, 0, 0, NOTE_F5, 0, 0,
   NOTE_E5, NOTE_E5, 0, NOTE_G5, 0, NOTE_E5, NOTE_D5, 0, 0,
   NOTE_D5, 0, 0, NOTE_C5, 0, 0,
   NOTE_B4, NOTE_C5, 0, NOTE_B4, 0, NOTE_A4
};
int duration[] = {
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 375, 125,
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 375, 125,
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 125, 250, 125,
  125, 125, 250, 125, 125,
  250, 125, 250, 125,
  125, 125, 250, 125, 125,
  125, 125, 375, 375,
  250, 125,
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 375, 125,
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 375, 125,
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 250, 125, 125,
  125, 125, 125, 250, 125,
  125, 125, 250, 125, 125,
  250, 125, 250, 125,
  125, 125, 250, 125, 125,
  125, 125, 375, 375,
  250, 125, 375, 250, 125, 375,
  125, 125, 125, 125, 125, 125, 125, 125, 375,
  250, 125, 375, 250, 125, 375,
  125, 125, 125, 125, 125, 500,
  250, 125, 375, 250, 125, 375,
  125, 125, 125, 125, 125, 125, 125, 125, 375,
  250, 125, 375, 250, 125, 375,
  125, 125, 125, 125, 125, 500
};

void setup() {
}

void loop() {
  for (int i=0;i<203;i++){
  int wait = duration[i] * songspeed;
  tone(buzzer,notes[i],wait);
  delay(wait);}
  delay(10000);
}
```

13. Lichtschranke



Verdrahten des Moduls



- wird mit **GND** verbunden — Schwarze Leitung
\+ wird mit **5V** verbunden — Rote Leitung
S wird mit **D7** verbunden — Gelbe Leitung

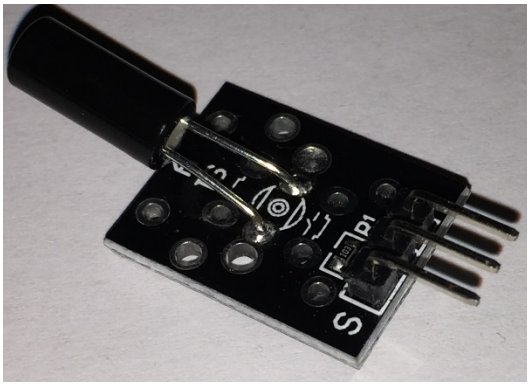
Software für die Lichtschranke:

```
int Led = LED_BUILTIN;
int Eingang = 7;
int value;

void setup ()
{
pinMode (Led, OUTPUT);
pinMode (Eingang, INPUT);
}

void loop ()
{
value=digitalRead(Eingang);
digitalWrite (Led, !value);
}
```

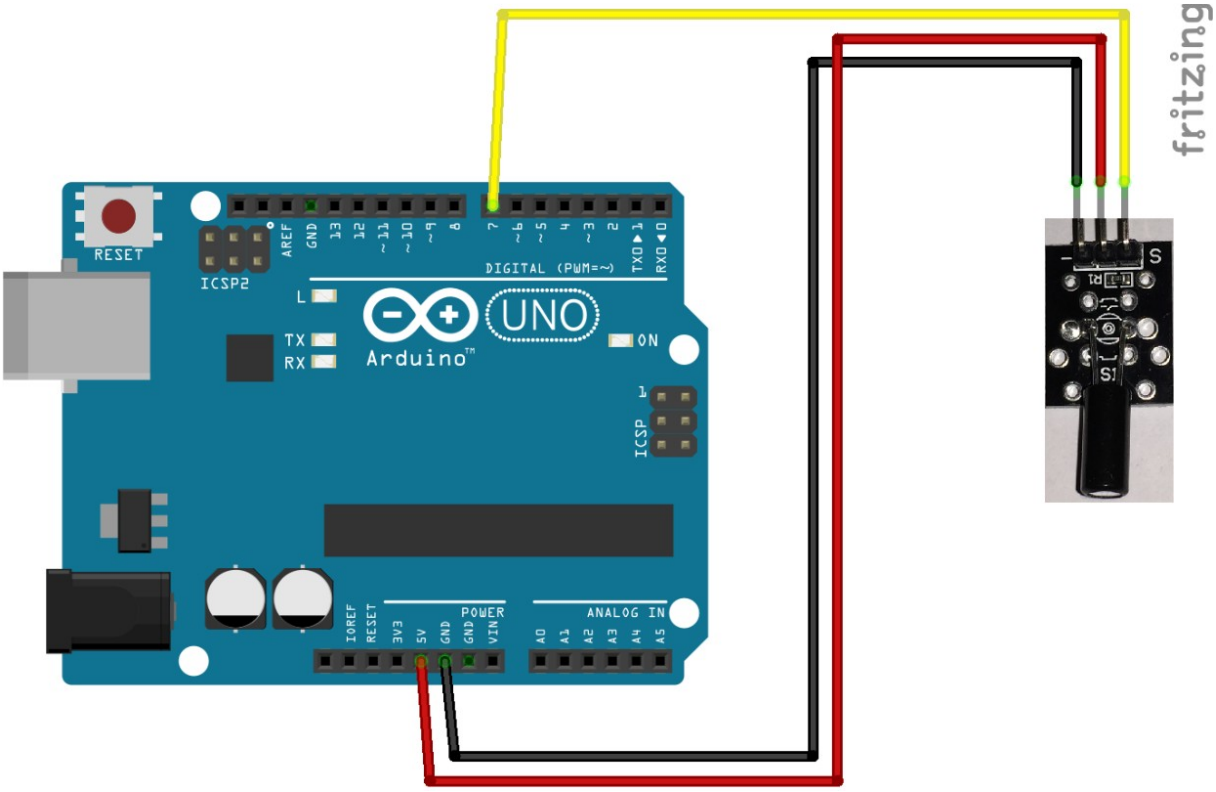
Der Code wird wieder Verifiziert  und Hochgeladen  .

Die LED auf dem Arduino leuchtet immer, wenn die Lichtschranke keinen Gegenstand erkennt. Sobald ein Gegenstand zwischen die Lichtschranke gehalten wird, leuchtet die LED nicht mehr.

14. Schocksensor



Verdrahten des Moduls



- wird mit **GND** verbunden — Schwarze Leitung
\+ wird mit **5V** verbunden — Rote Leitung
S wird mit **D7** verbunden — Gelbe Leitung

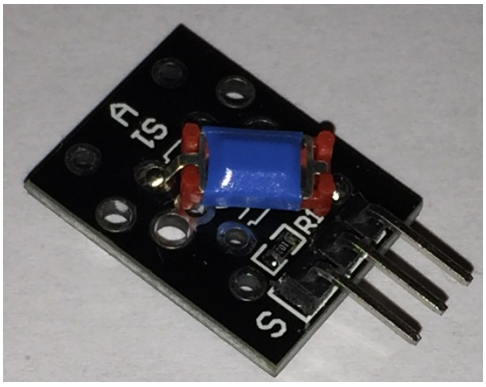
Software für den Schocksensor:

```
int Led = LED_BUILTIN;
int Eingang = 7;
int value;

void setup ()
{
pinMode (Led, OUTPUT);
pinMode (Eingang, INPUT);
}

void loop ()
{
value=digitalRead(Eingang);
digitalWrite (Led, value);
}
```

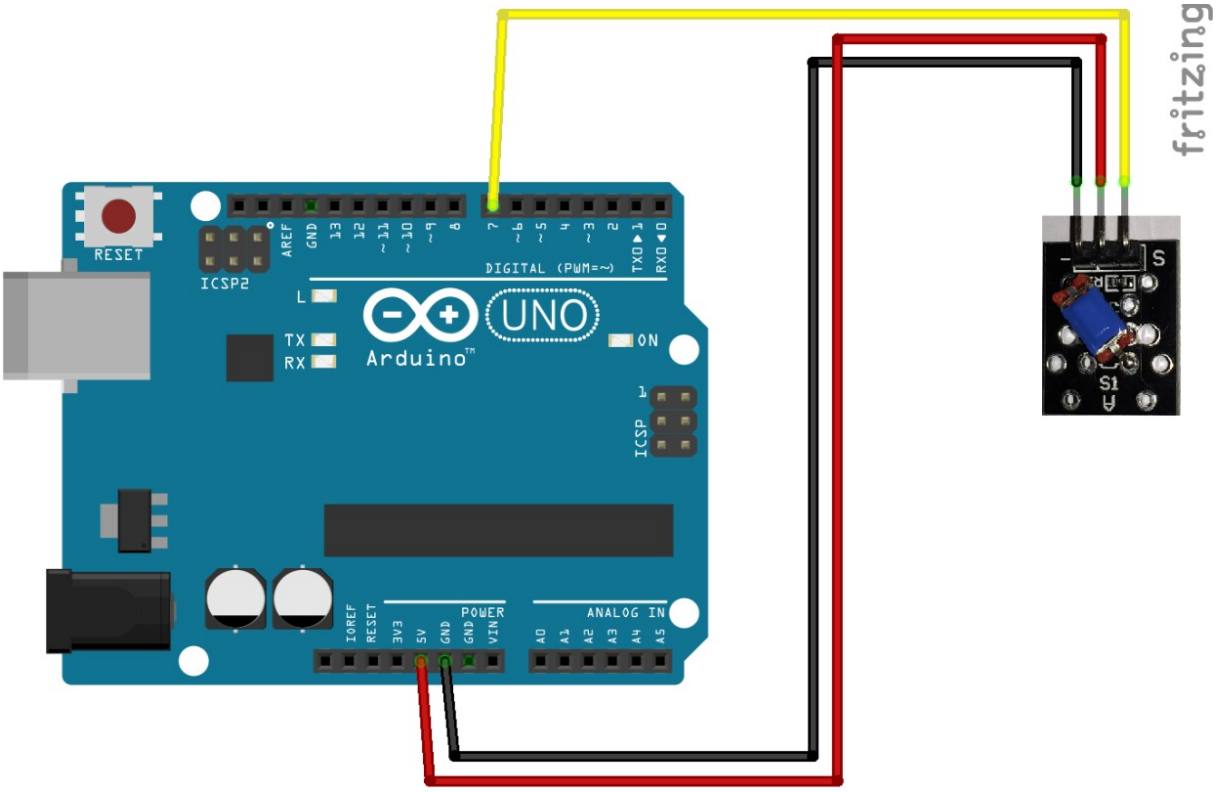
Der Code wird wieder Verifiziert  und Hochgeladen  .

Die LED auf dem Arduino leuchtet immer, wenn Schocksensor bewegt wird, je stärker desto mehr blinkt die LED am Arduino. Ist der Sensor in Ruhelage, leuchtet die LED entweder dauerhaft oder gar nicht.

15. Schüttelsensor



Verdrahten des Moduls



- wird mit **GND** verbunden — Schwarze Leitung
\+ wird mit **5V** verbunden — Rote Leitung
S wird mit **D7** verbunden — Gelbe Leitung

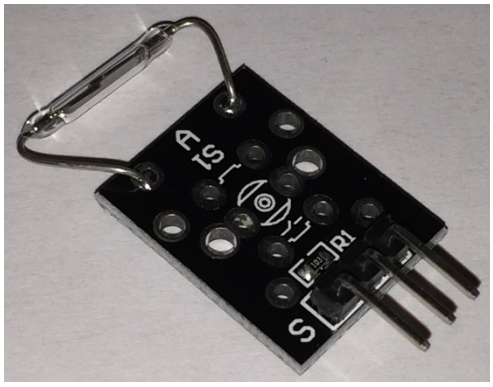
Software für den Schüttelsensor:

```
int Led = LED_BUILTIN;
int Eingang = 7;
int value;

void setup ()
{
pinMode (Led, OUTPUT);
pinMode (Eingang, INPUT);
}

void loop ()
{
value=digitalRead(Eingang);
digitalWrite (Led, value);
}
```

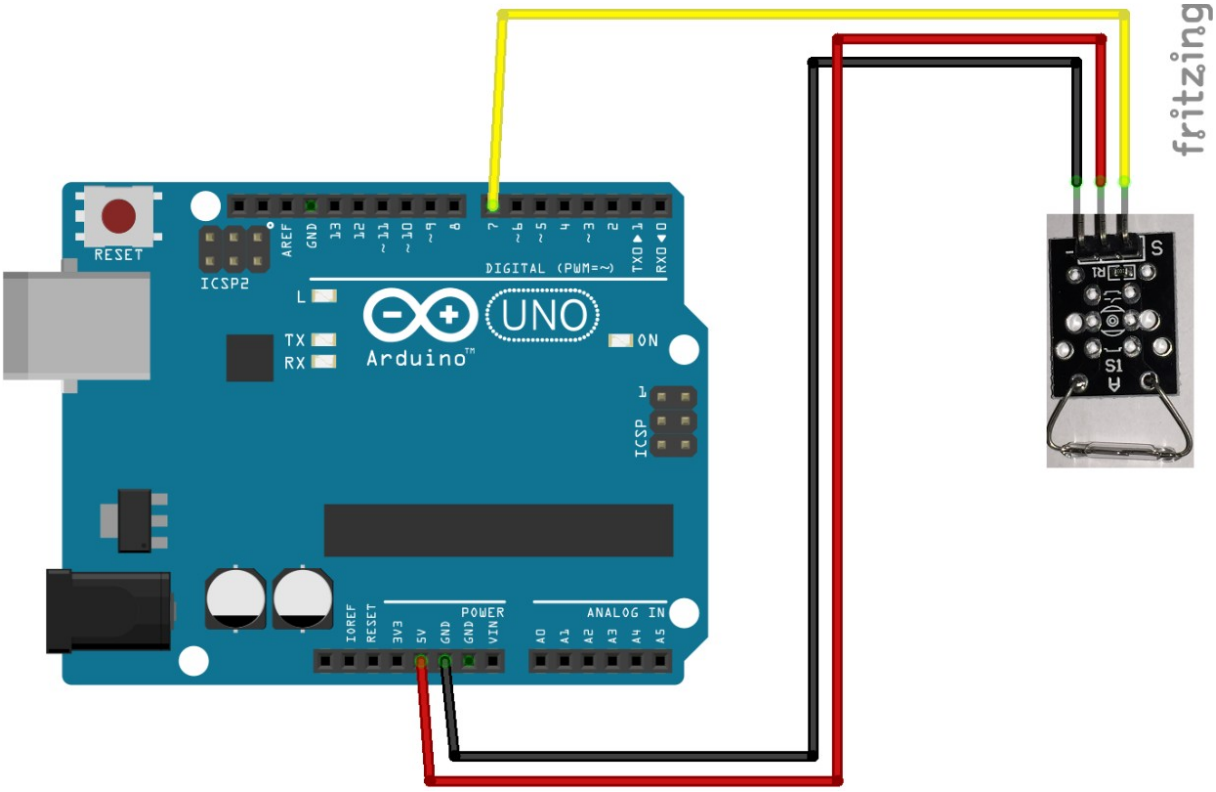
Der Code wird wieder Verifiziert  und Hochgeladen  .

Die LED auf dem Arduino blinkt immer, wenn der Schüttelsensor geschüttelt wird. Der Sensor ist ähnlich wie der Schocksensor.

16. Magnetschalter (Reedrelais)



Verdrahten des Moduls



- wird mit **GND** verbunden — Schwarze Leitung
\+ wird mit **5V** verbunden — Rote Leitung
S wird mit **D7** verbunden — Gelbe Leitung

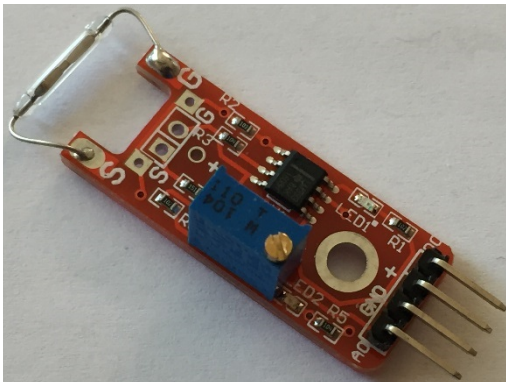
Software für den Magnetschalter:

```
int Led = LED_BUILTIN;
int Eingang = 7;
int value;

void setup ()
{
pinMode (Led, OUTPUT);
pinMode (Eingang, INPUT);
}

void loop ()
{
value=digitalRead(Eingang);
digitalWrite (Led, value);
}
```

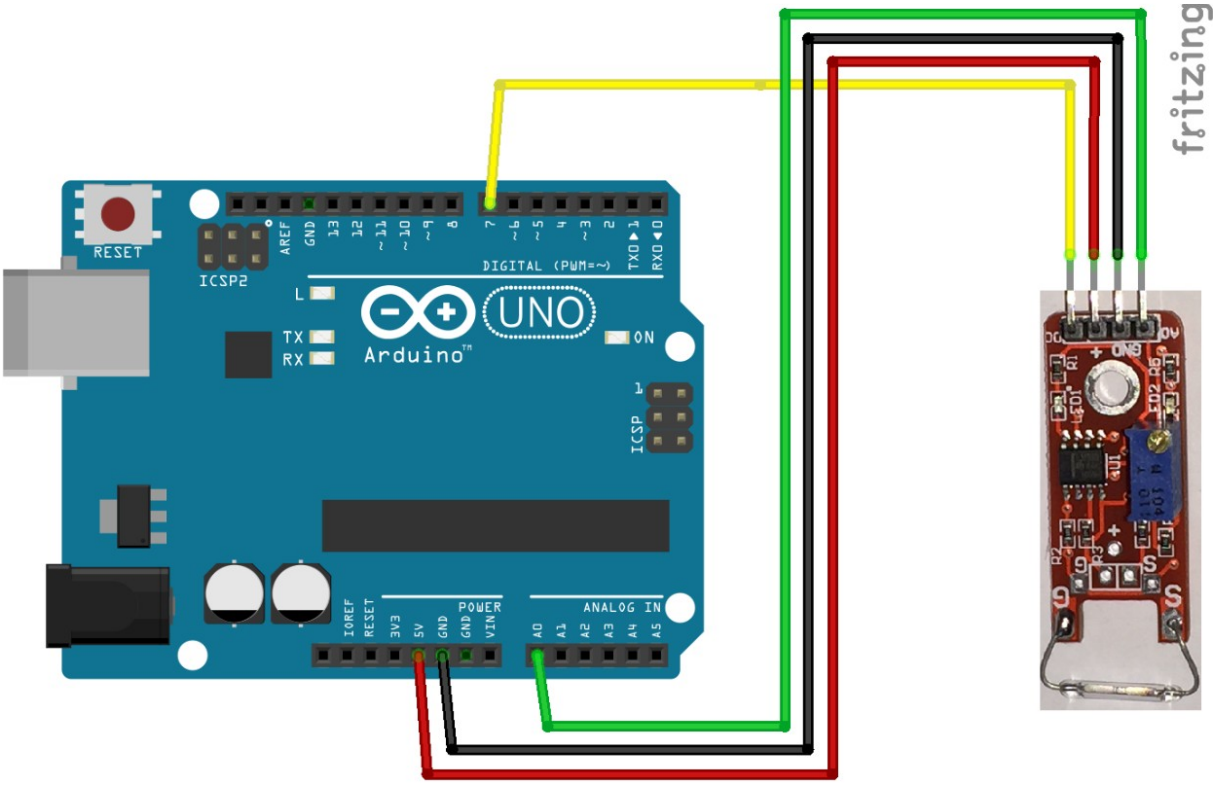
Der Code wird wieder Verifiziert  und Hochgeladen  .

Die LED auf dem Arduino leuchtet, sobald sich ein Magnet dem Reedrelais nähert.
Die 2 Metallplättchen ziehen sich in einem Magnetfeld an und schließen den Kontakt.

17. Magnetsensor



Verdrahten des Moduls

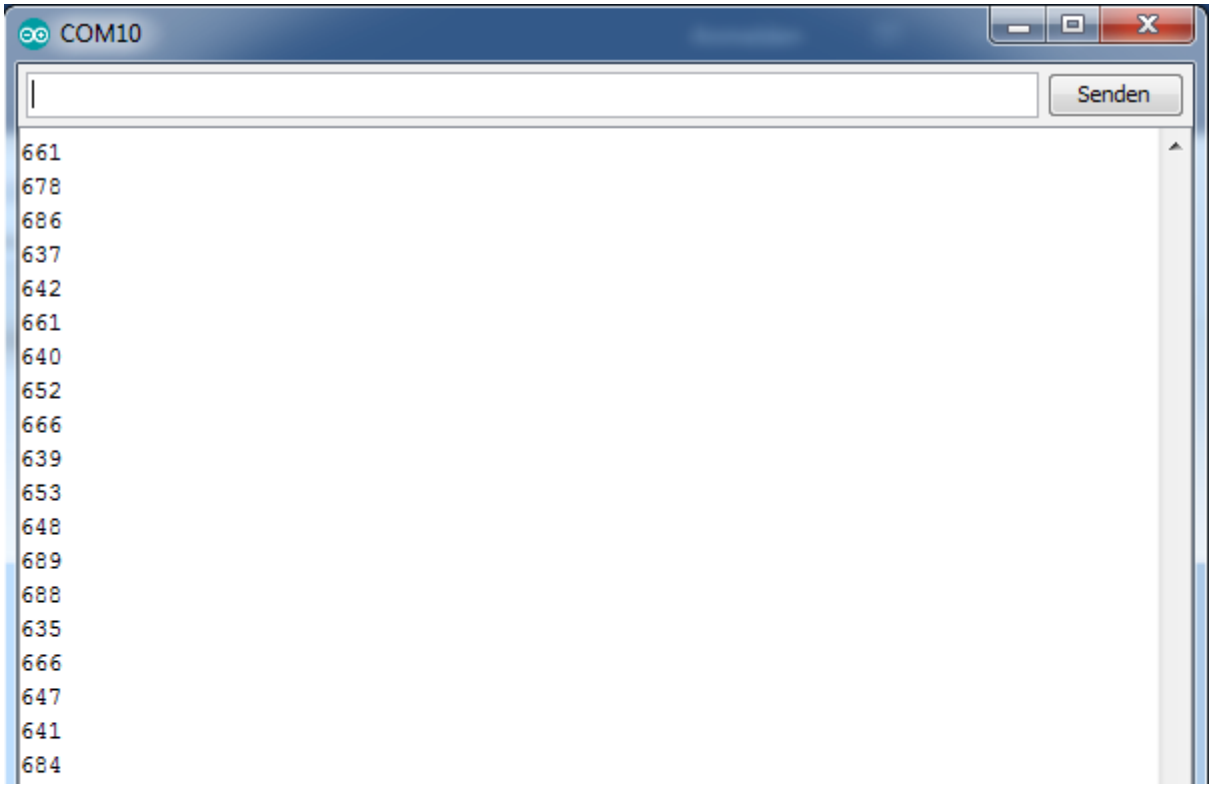


+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
DO wird mit **D7** verbunden — Gelbe Leitung
AO wird mit **A0** verbunden — Grüne Leitung

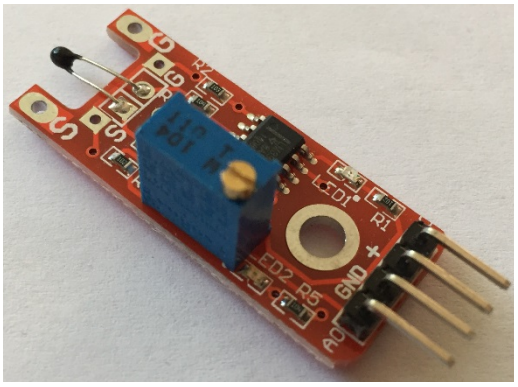
Software für den Magnetsensor:

```
int Led=LED_BUILTIN;
int Reed=7;
int PotiPin = A0;
int PotiValue = 0;
int val;

void setup()
{
Serial.begin(9600);
pinMode(Led,OUTPUT);
pinMode(Reed,INPUT);
}

void loop()
{
PotiValue = analogRead(PotiPin);
Serial.println(PotiValue, DEC);
val=digitalRead(Reed);
if(val==HIGH)
  {
    digitalWrite(Led,HIGH);
  }
  else
  {
    digitalWrite(Led,LOW);
  }
}
```

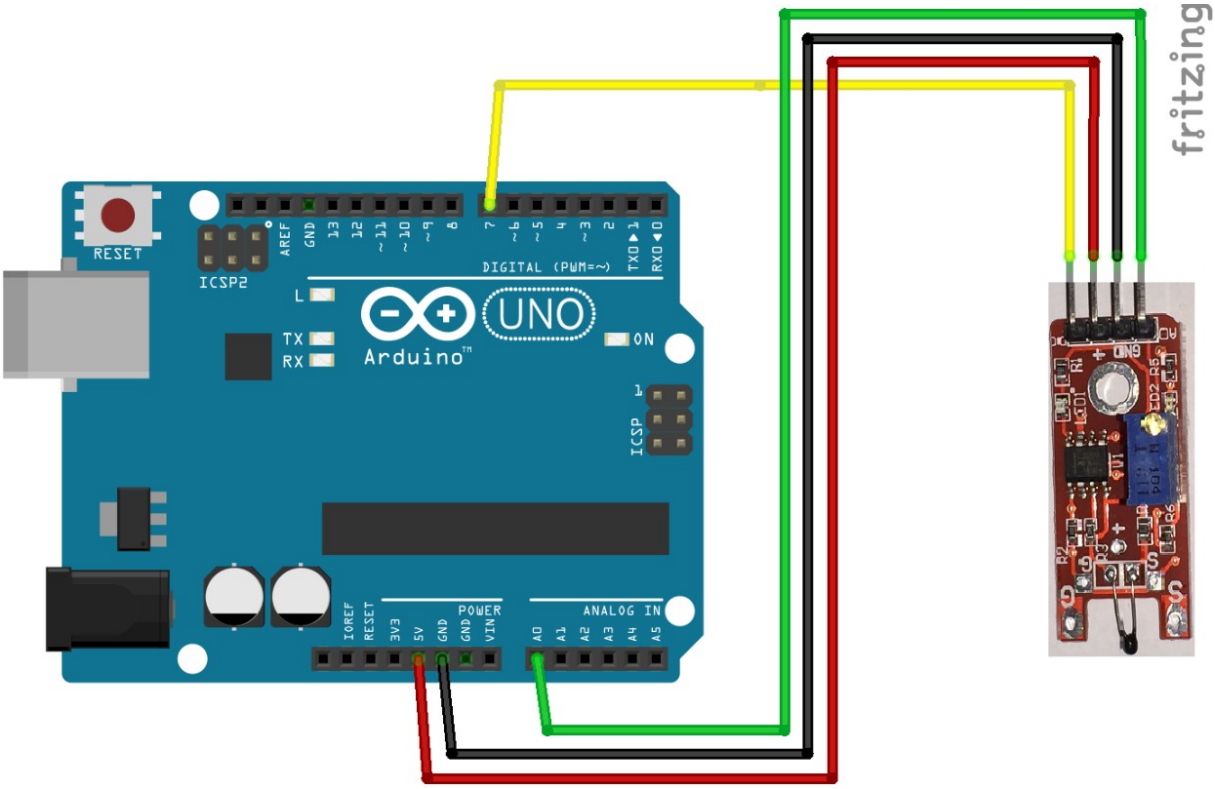
Der Code wird wieder Verifiziert und Hochgeladen. Im SerialMonitor wird nun der Analoge Wert des Potentiometers ausgegeben, wenn kein Magnet in der Nähe ist, Ist ein Magnet in der Nähe, wird nur eine 0 oder 1 am Analogeingang gemessen.



18. Thermistor (Temperaturschalter)



Verdrahten des Moduls

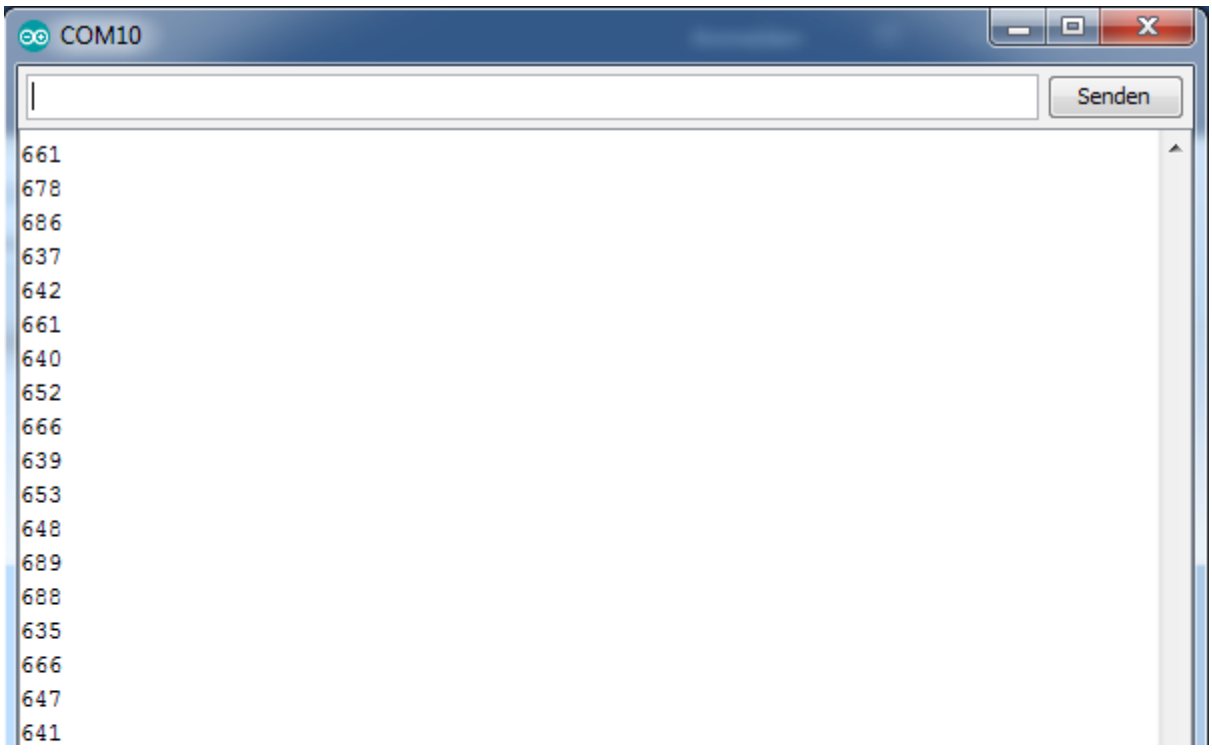


+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
DO wird mit **D7** verbunden — Gelbe Leitung
AO wird mit **A0** verbunden — Grüne Leitung

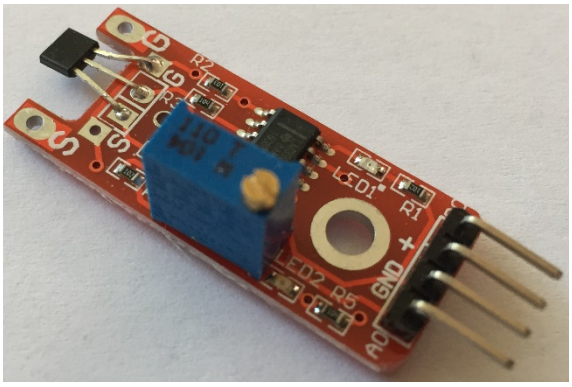
Software für den Thermistor:

```
int Led=LED_BUILTIN;
int Sensor=7;
int PotiPin = A0;
int PotiValue = 0;
int val;

void setup()
{
Serial.begin(9600);
pinMode(Led,OUTPUT);
pinMode(Sensor,INPUT);
}

void loop()
{
PotiValue = analogRead(PotiPin);
Serial.println(PotiValue, DEC);
val=digitalRead(Sensor);
if(val==HIGH)
  {
    digitalWrite(Led,HIGH);
  }
  else
  {
    digitalWrite(Led,LOW);
  }
}
```

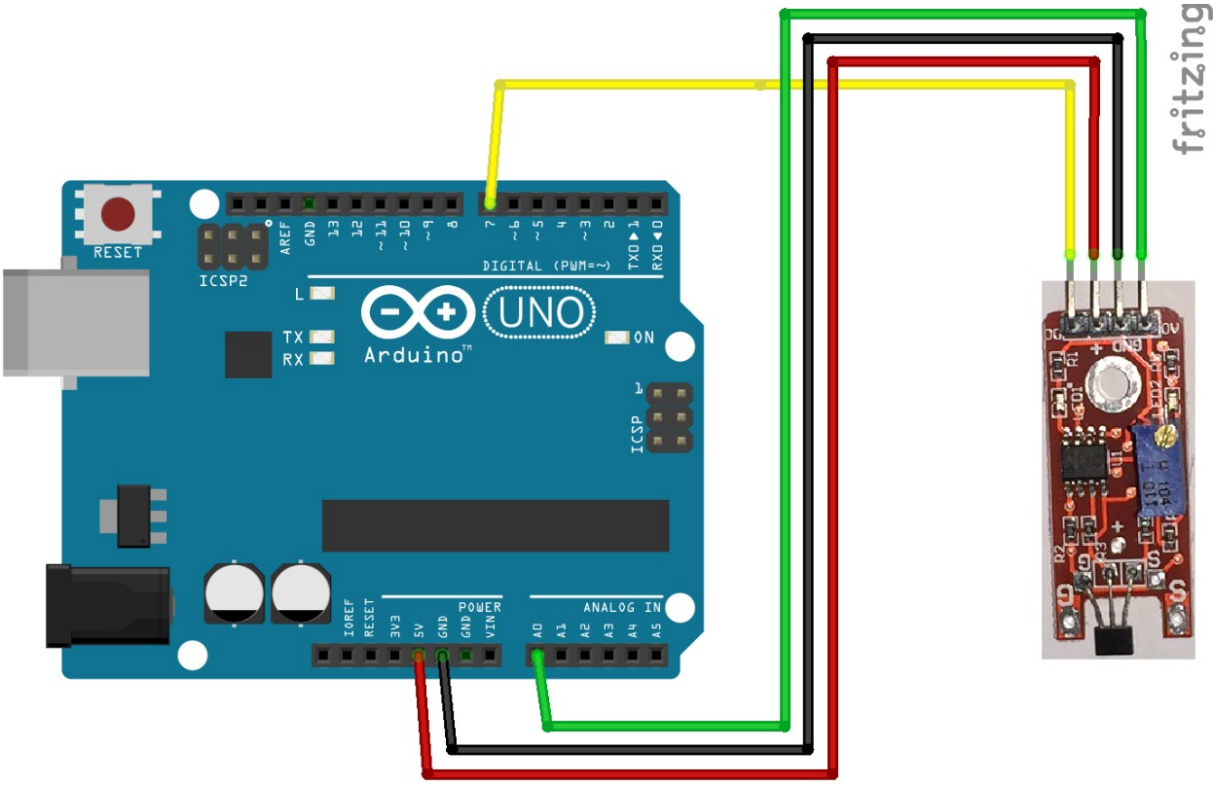
Der Code wird wieder Verifiziert und Hochgeladen. Im SerialMonitor wird nun der Analoge Wert des Potentiometers ausgegeben, wenn die Temperatur über den eingestellten Analogwert geht, wird der Digitale Ausgang geschalten (OnBoard LED leuchtet).



19. Hallsensensor



Verdrahten des Moduls

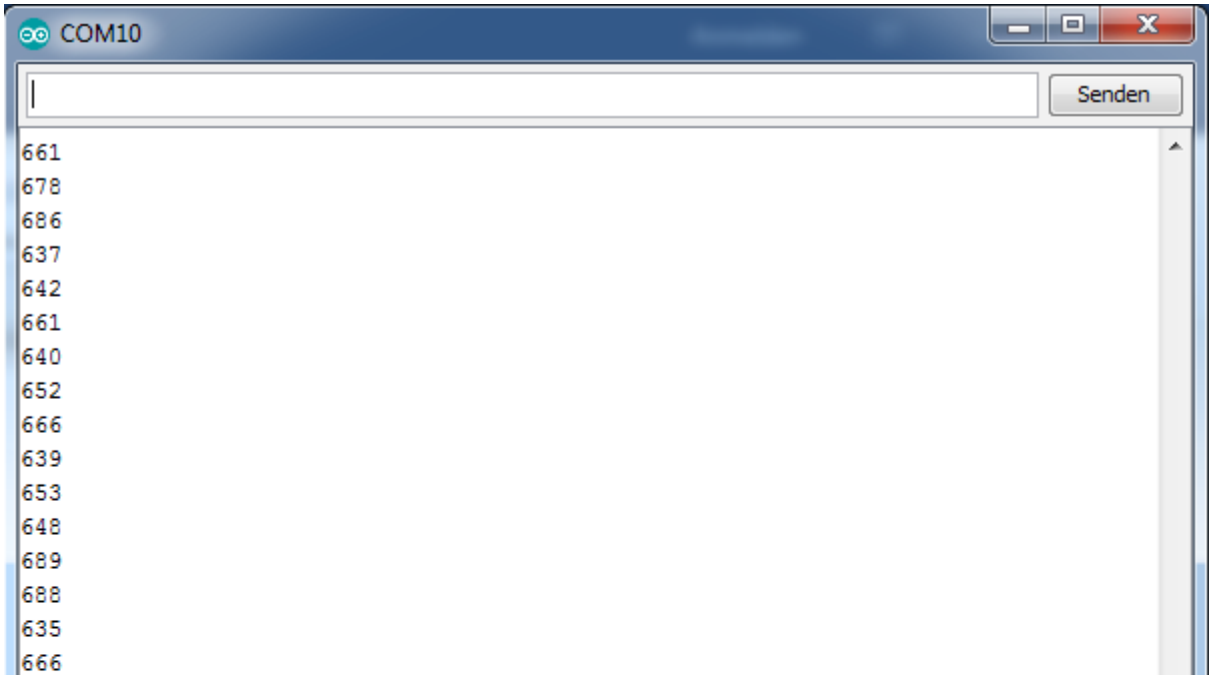


+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
DO wird mit **D7** verbunden — Gelbe Leitung
AO wird mit **A0** verbunden — Grüne Leitung

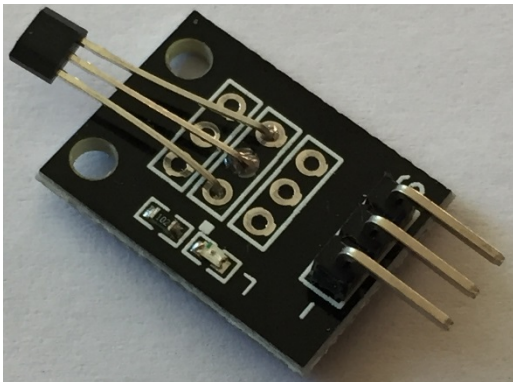
Software für den Hallsensor:

```
int Led=LED_BUILTIN;
int Sensor=7;
int PotiPin = A0;
int PotiValue = 0;
int val;

void setup()
{
Serial.begin(9600);
pinMode(Led,OUTPUT);
pinMode(Sensor,INPUT);
}

void loop()
{
PotiValue = analogRead(PotiPin);
Serial.println(PotiValue, DEC);
val=digitalRead(Sensor);
if(val==HIGH)
  {
    digitalWrite(Led,HIGH);
  }
  else
  {
    digitalWrite(Led,LOW);
  }
}
```

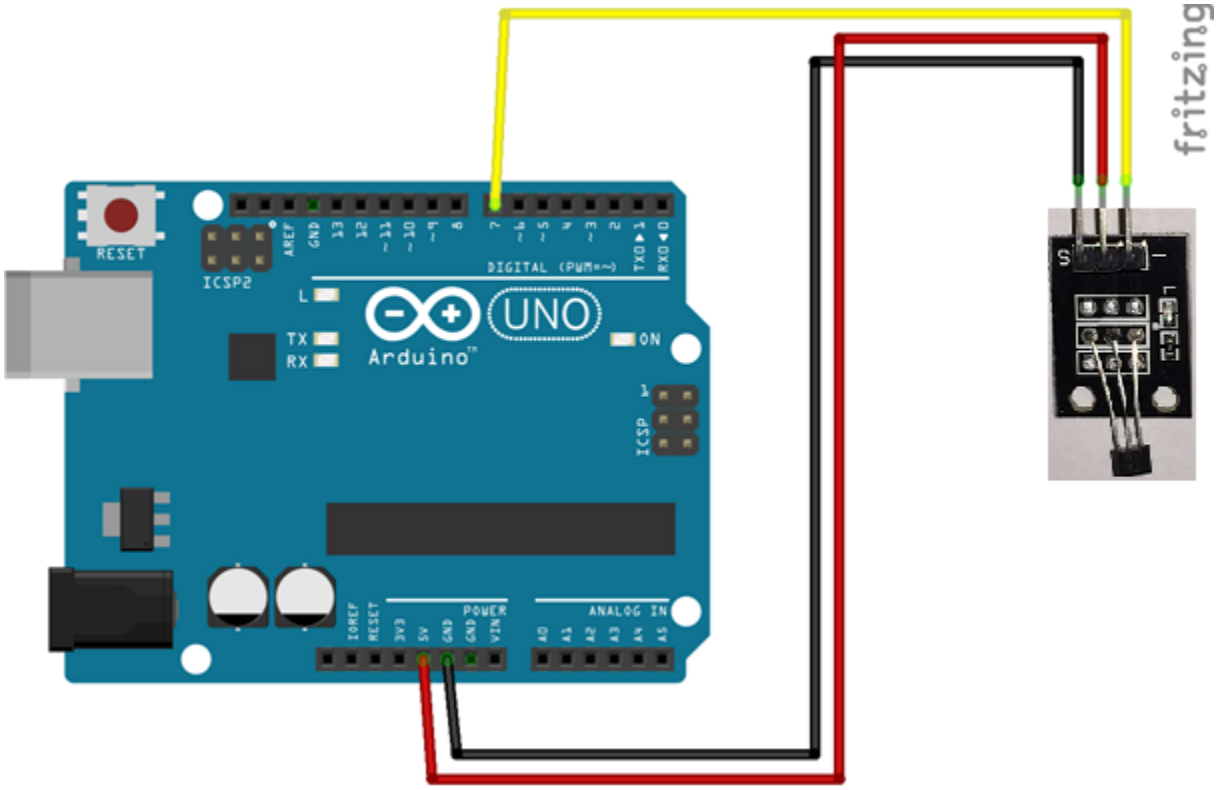
Der Code wird wieder Verifiziert und Hochgeladen. Im SerialMonitor wird nun der Analoge Wert des Potentiometers ausgegeben, wenn das erkannte Magnetfeld über den eingestellten Schwellwert geht, wird der Digitale Ausgang geschalten (OnBoard LED leuchtet).



20. Digitaler Hallsensor



Verdrahten des Moduls



+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
S wird mit **D7** verbunden — Gelbe Leitung

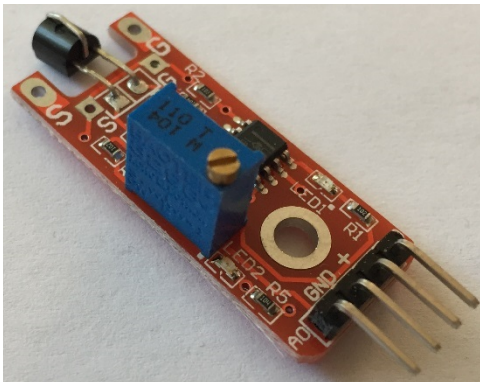
Software für den Hallsensor:

```
int Led = LED_BUILTIN;
int Eingang = 7;
int value;

void setup ()
{
pinMode (Led, OUTPUT);
pinMode (Eingang, INPUT);
}

void loop ()
{
value=digitalRead(Eingang);
digitalWrite (Led, value);
}
```

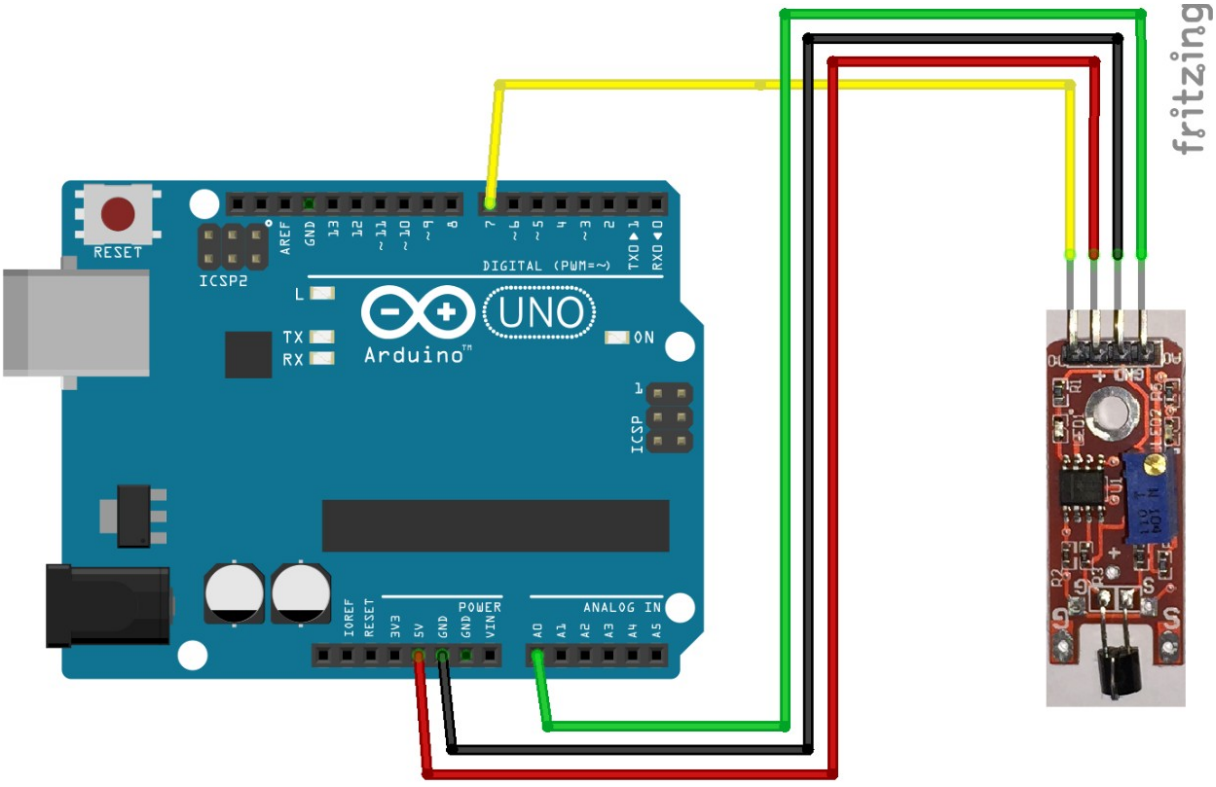
Der Code wird wieder Verifiziert und Hochgeladen .

Nähert sich dem Hallsensor ein Magnetfeld, wird der Ausgang am Sensor geschalten. Dies geht an einen Eingang des Arduinos, dieser Schaltet die OnBoard LED ein.

21. Touchsensor



Verdrahten des Moduls

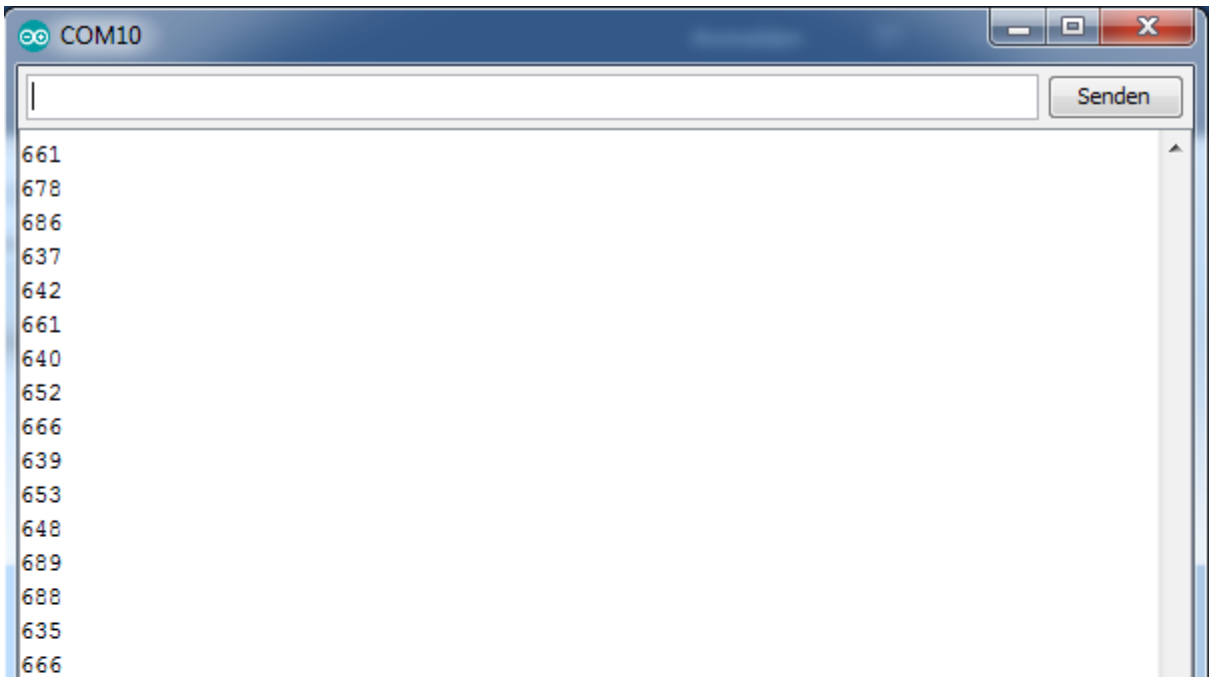


+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
DO wird mit **D7** verbunden — Gelbe Leitung
AO wird mit **A0** verbunden — Grüne Leitung

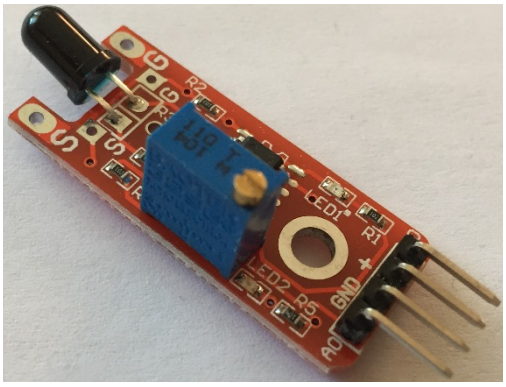
Software für den Touchsensor:

```
int Led=LED_BUILTIN;
int Sensor=7;
int PotiPin = A0;
int PotiValue = 0;
int val;

void setup()
{
Serial.begin(9600);
pinMode(Led,OUTPUT);
pinMode(Sensor,INPUT);
}

void loop()
{
PotiValue = analogRead(PotiPin);
Serial.println(PotiValue, DEC);
val=digitalRead(Sensor);
if(val==HIGH)
  {
    digitalWrite(Led,HIGH);
  }
  else
  {
    digitalWrite(Led,LOW);
  }
}
```

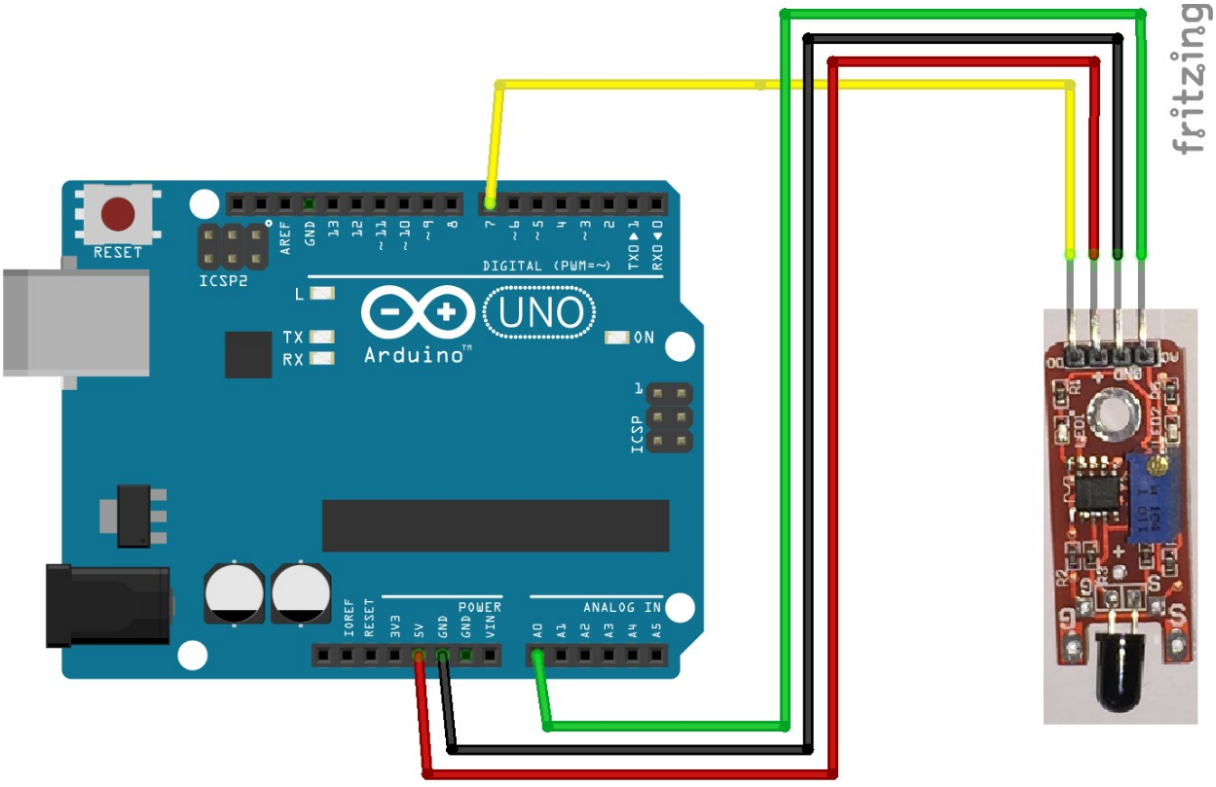
Der Code wird wieder Verifiziert und Hochgeladen. Im SerialMonitor wird nun der Analoge Wert des Potentiometers ausgegeben, berührt man den Touchsensor, wird ein Digitales Signal ausgegeben. Die Empfindlichkeit kann man mit dem Potentiometer einstellen.



22. Flammensensor



Verdrahten des Moduls

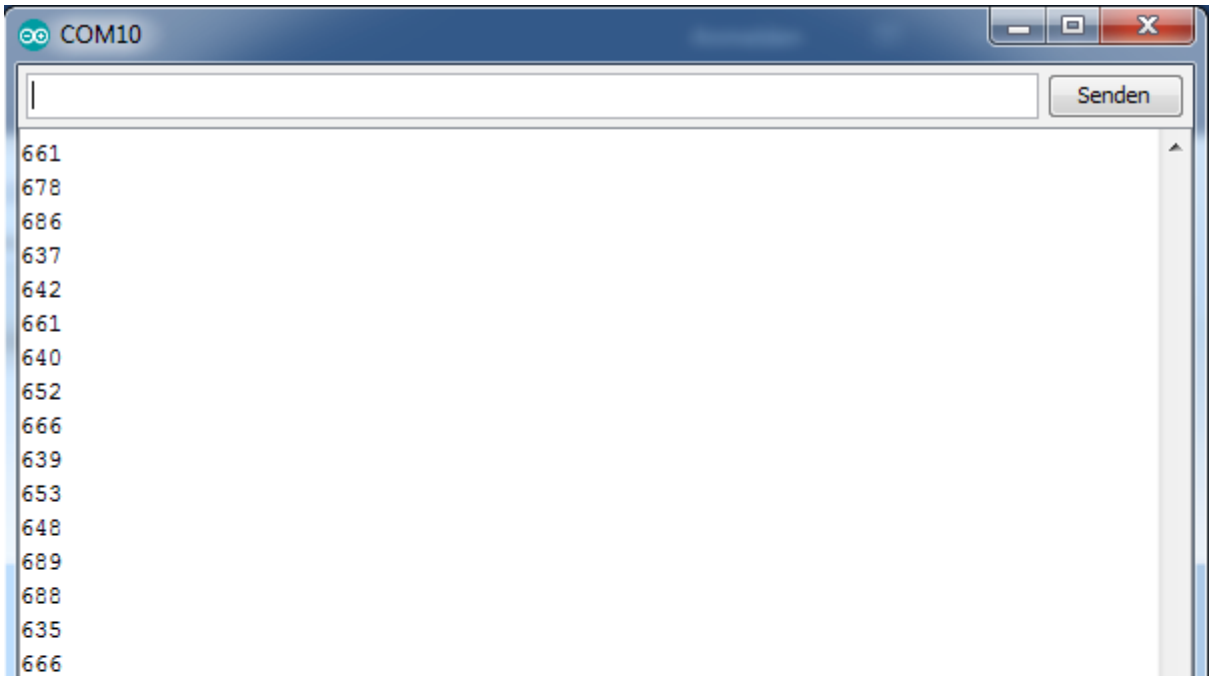


+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
DO wird mit **D7** verbunden — Gelbe Leitung
AO wird mit **A0** verbunden — Grüne Leitung

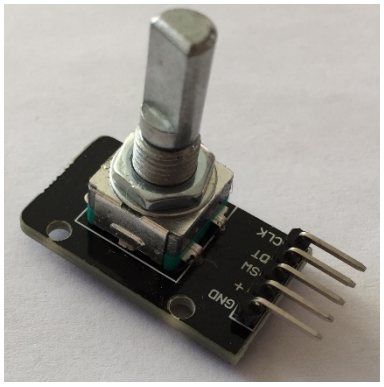
Software für den Flammensensor:

```
int Led=LED_BUILTIN;
int Sensor=7;
int PotiPin = A0;
int PotiValue = 0;
int val;

void setup()
{
Serial.begin(9600);
pinMode(Led,OUTPUT);
pinMode(Sensor,INPUT);
}

void loop()
{
PotiValue = analogRead(PotiPin);
Serial.println(PotiValue, DEC);
val=digitalRead(Sensor);
if(val==HIGH)
  {
    digitalWrite(Led,HIGH);
  }
  else
  {
    digitalWrite(Led,LOW);
  }
}
```

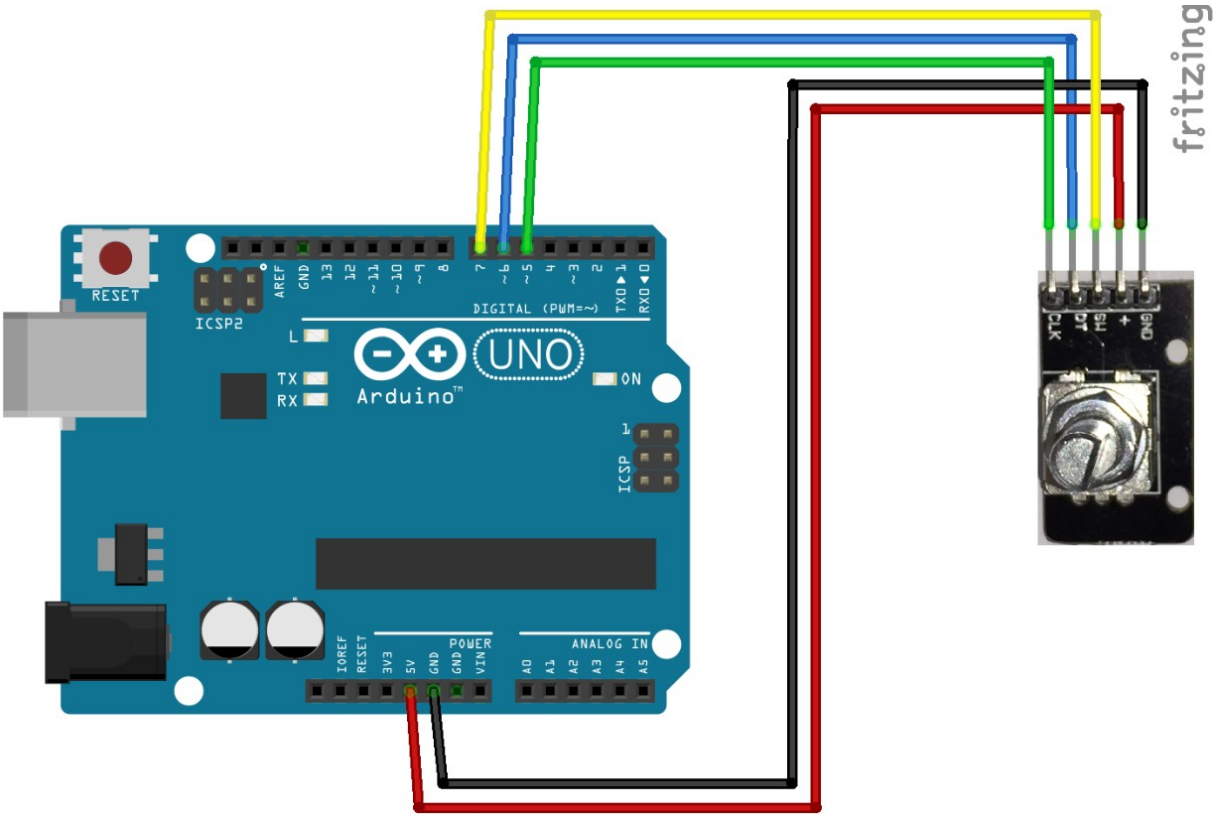
Der Code wird wieder Verifiziert und Hochgeladen. Im SerialMonitor wird nun der Analoge Wert des Sensors ausgegeben. Der Sensor reagiert auf Feuer. Wird eine Flamme eines offenen Feuers detektiert, so wird der Digitalausgang geschalten und LED auf dem Arduino beginnt zu leuchten. Die Sensitivität kann wieder mit dem Potentiometer eingestellt werden.



23. Drehschalter



Verdrahten des Schalters

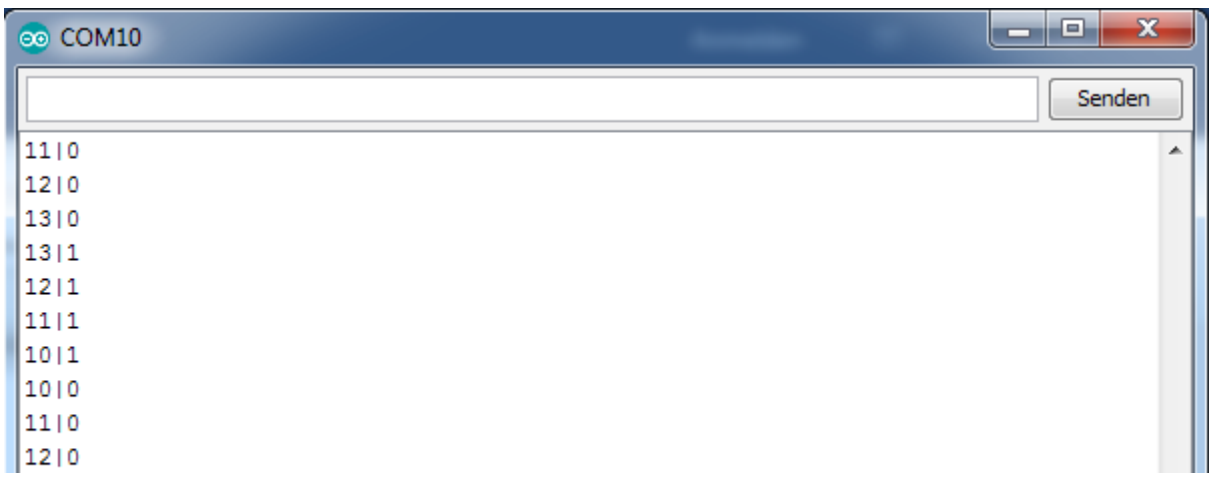


+ wird mit **5V** am Arduino verbunden	Rote Leitung
GND wird mit **GND** verbunden	Schwarze Leitung
SW wird mit **D7** verbunden	Gelbe Leitung
DT wird mit **D6** verbunden	Blaue Leitung
CLK wird mit **D5** verbunden	Grüne Leitung

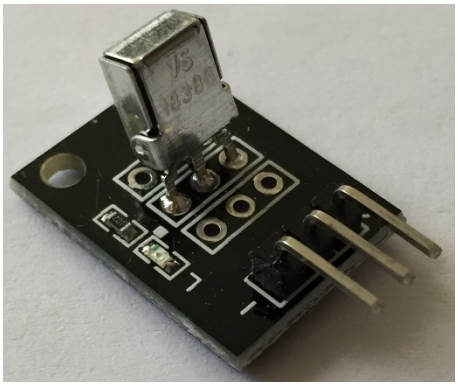
Software für den Drehschalter:

```
int DT = 6;
int CLK = 5;
int SW = 7;
int Position = 0;
int Letzte_Position = LOW;

int n = LOW;
int Taster = LOW;
int Letzte_Taster = LOW;
void setup() {
  pinMode (CLK, INPUT_PULLUP);
  pinMode (DT, INPUT_PULLUP);
  pinMode (SW, INPUT_PULLUP);
  Serial.begin (9600);
}

void loop() {
  n = digitalRead(CLK);
  Taster = !digitalRead(SW);
  if (Taster != Letzte_Taster){
    Serial.print (Position);
    Serial.print ("|");
    Serial.println (Taster);
    delay(10);
    Letzte_Taster = Taster;
  }
  if ((Letzte_Position == LOW) && (n == HIGH)) {
    if (digitalRead(DT) == LOW) {
      Position++;
    } else {
      Position--;
    }

    Serial.print (Position);
    Serial.print ("|");
    Serial.println (Taster);
  }
  Letzte_Position = n;
}
```

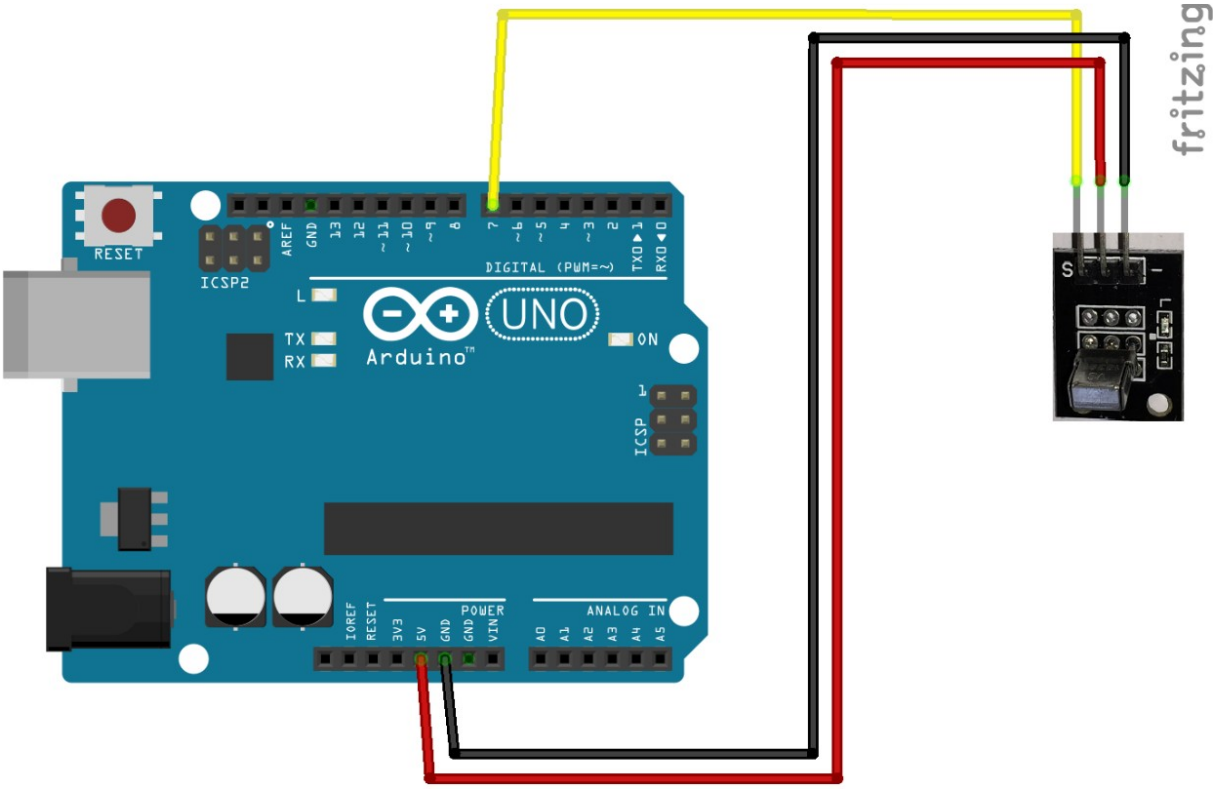
Der Code wird wieder Verifiziert  und Hochgeladen  . Im SerialMonitor wird nun die aktuelle Drehposition und ob der Taster gedrückt ist ausgegeben.



24. IR-Empfänger



Verdrahten des Moduls



+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
S wird mit **D7** verbunden — Gelbe Leitung

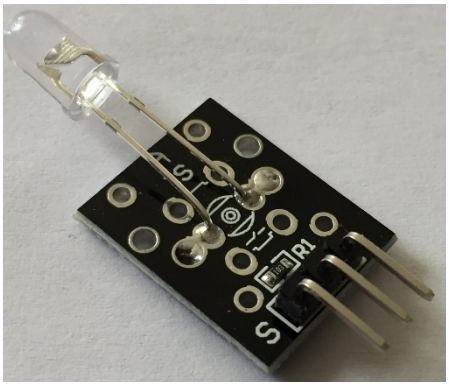
Software für den IR-Sensor:

```
int Led = LED_BUILTIN;
int Eingang = 7;
int value;

void setup ()
{
pinMode (Led, OUTPUT);
pinMode (Eingang, INPUT);
}

void loop ()
{
value=digitalRead(Eingang);
digitalWrite (Led, value);
}
```

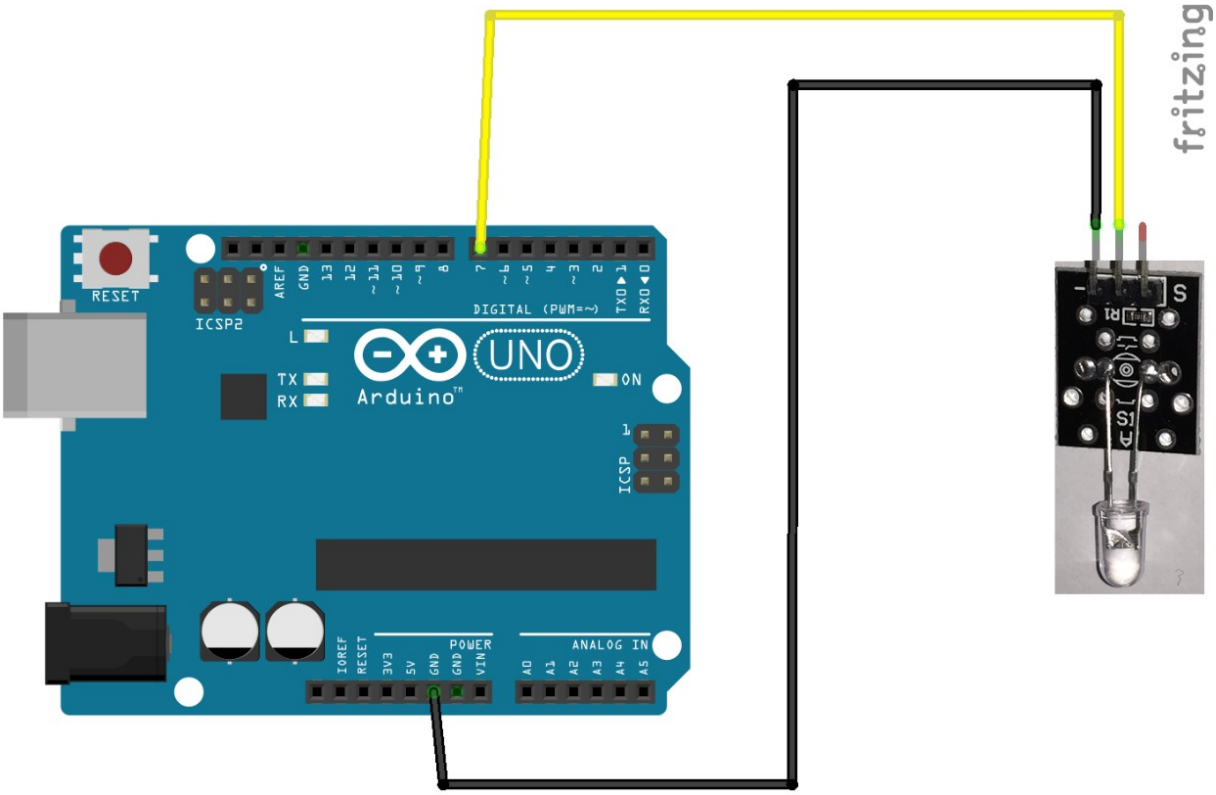
Der Code wird wieder Verifiziert  und Hochgeladen  .

Sobald ein IR-Licht empfangen wird, leuchtet die LED am Arduino. Je nachdem wie lange die Signale anliegen, kann daraus ein Code berechnet werden, dieser kann für Fernbedienungen verwendet werden.

25. IR-Sender



Verdrahten des Moduls



- wird mit **GND** verbunden — Schwarze Leitung
+ wird mit **D7** verbunden — Gelbe Leitung

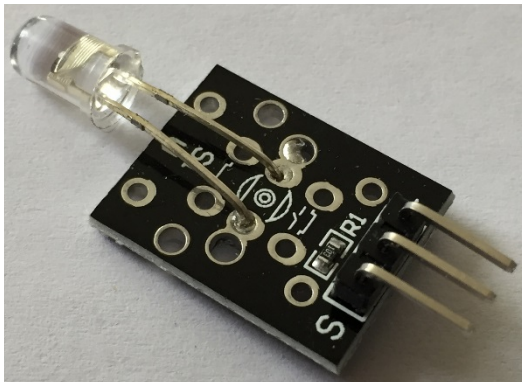
Software für den IR-Sender:

```
int IR = 7;

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

void loop() {
digitalWrite(IR, HIGH);
delay(100);
digitalWrite(IR, LOW);
delay(100);
}
```

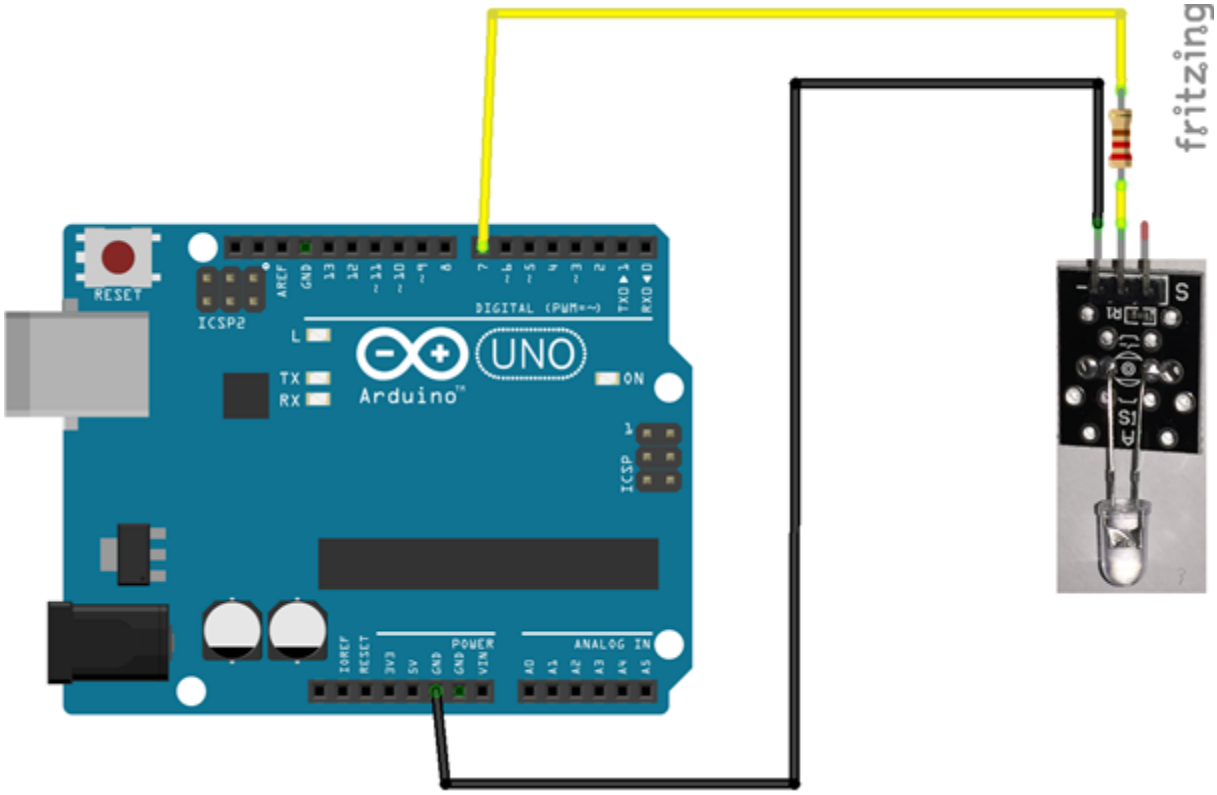
Der Code wird wieder Verifiziert  und Hochgeladen  .

Die IR-Diode blinkt nun mit 5Hz. Lässt man diese einen entsprechenden Code senden, können diverse Geräte geschalten werden. Das IR-Licht ist nicht sichtbar!

26. Farbwechsel-LED



Verdrahten des Moduls



- wird mit **GND** verbunden — Schwarze Leitung
+ wird mit **D7** verbunden — Gelbe Leitung

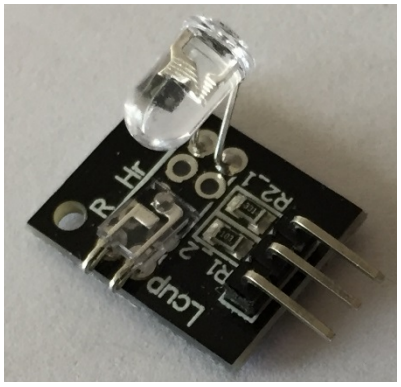
Software für die Farbwechsel-LED:

```
int LED = 7;

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

void loop() {
digitalWrite(LED, HIGH);
delay(10000);
digitalWrite(LED, LOW);
delay(1000);
}
```

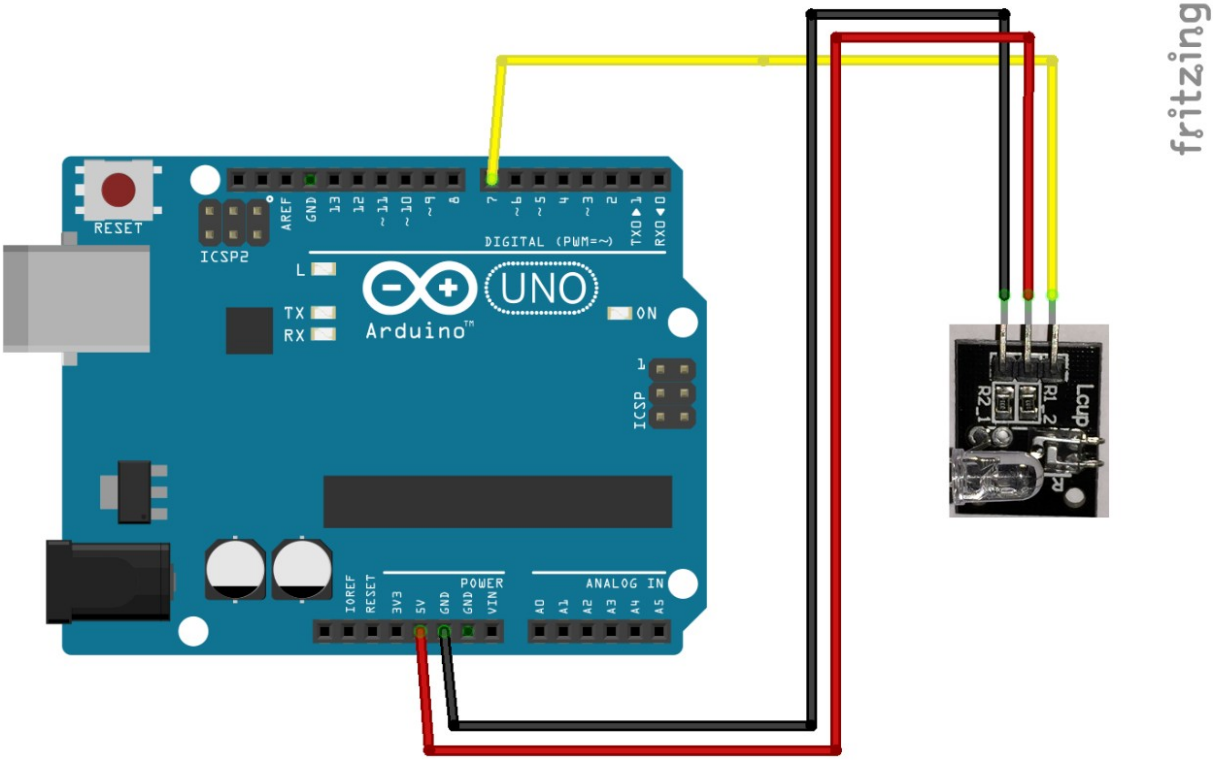
Der Code wird wieder Verifiziert  und Hochgeladen  .

Die Farbwechsel-LED wird für 10 Sekunden eingeschalten und dann für 1 Sekunde ausgeschalten. Während den 10 Sekunden leuchtet und blinkt die LED in verschieden farben.

27. IR-Lichtschranke



Verdrahten des Moduls



+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
S wird mit **D7** verbunden — Gelbe Leitung

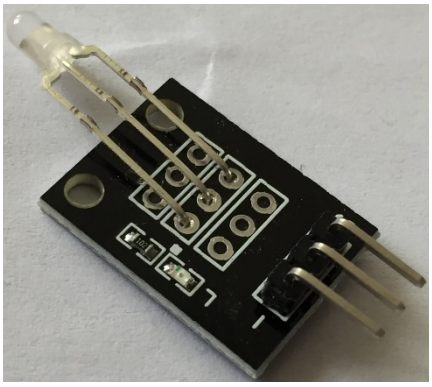
Software für den IR-Lichtschranke:

```
int Led = LED_BUILTIN;
int Eingang = 7;
int value;

void setup ()
{
pinMode (Led, OUTPUT);
pinMode (Eingang, INPUT);
}

void loop ()
{
value=digitalRead(Eingang);
digitalWrite (Led, value);
}
```

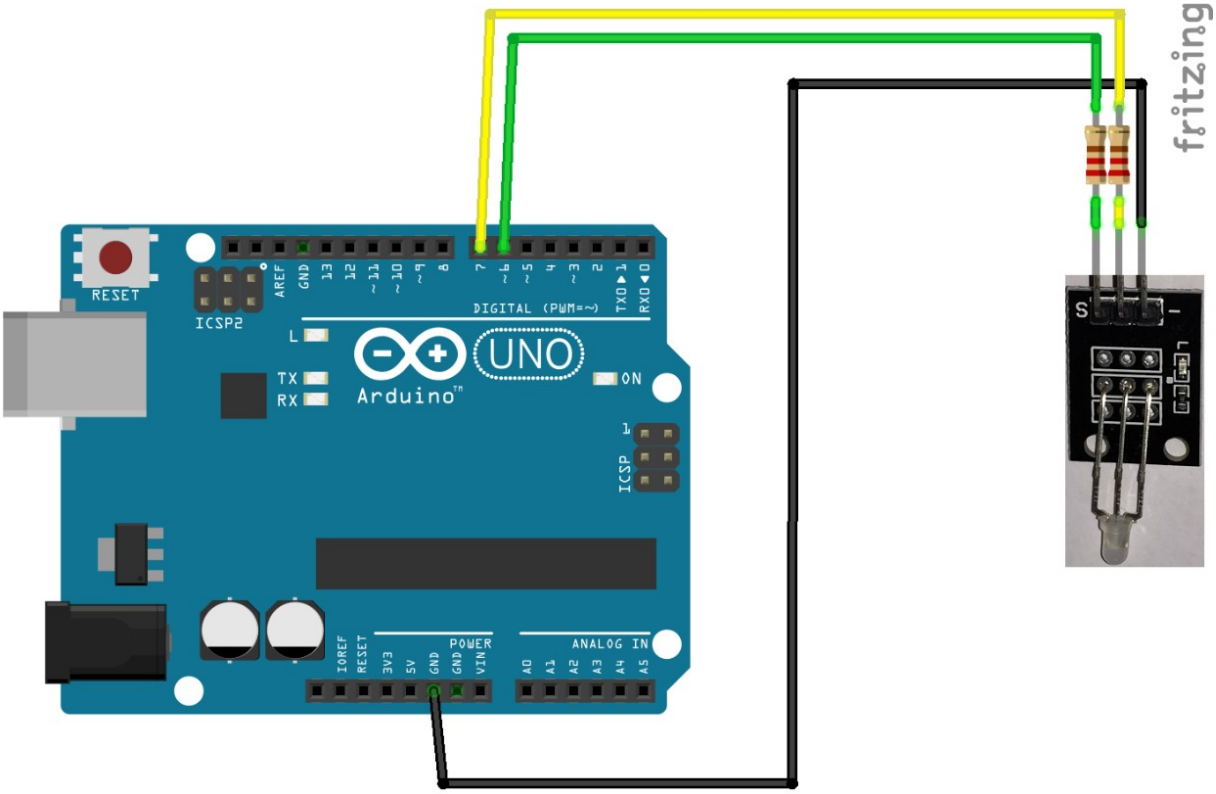
Der Code wird wieder Verifiziert  und Hochgeladen  .

Der Sensor besteht auf einer IR-LED und einem IR-Sensor. Legt man einen Finger oder Gegenstand dazwischen, kommt kein Licht mehr auf den Sensor und dieser schaltet ab.

28. Bicolor-LED (2 Farben-LED) 3mm



Verdrahten des Moduls



+ wird mit **D7** verbunden	220Ohm	Gelbe Leitung
- wird mit **GND** verbunden		Schwarze Leitung
S wird mit **D6** verbunden	220Ohm	Grüne Leitung

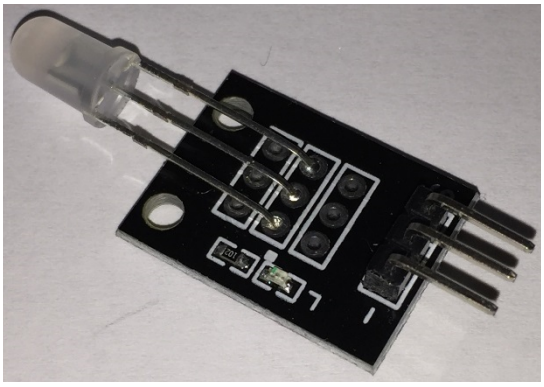
Software für die Bicolor-LED:

```
int LED1 = 7;
int LED2 = 6;

void setup(){
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
}

void loop(){
  digitalWrite(LED1, HIGH);
  digitalWrite(LED2, LOW);
delay(500);
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, HIGH);
delay(500);
  digitalWrite(LED1, HIGH);
  digitalWrite(LED2, HIGH);
delay(500);
}
```

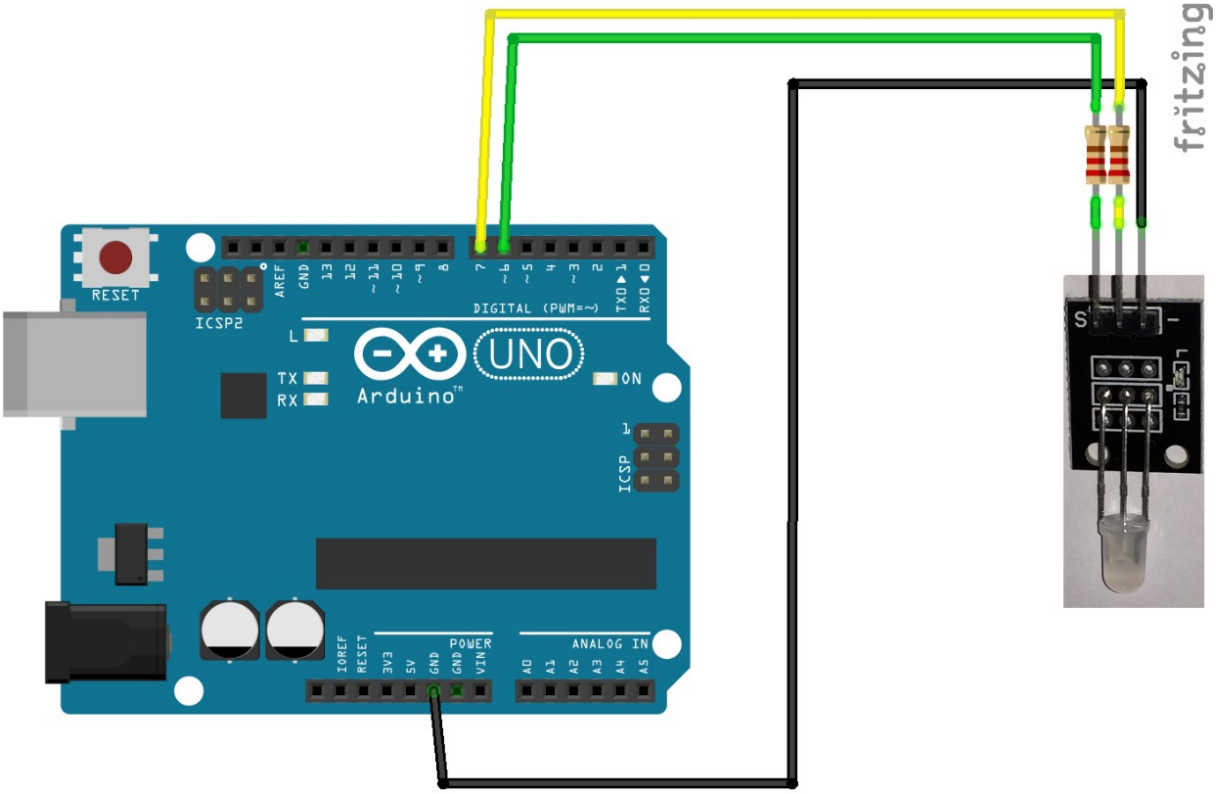
Der Code wird wieder Verifiziert  und Hochgeladen  .

Die 2-Farben-LED benötigt Vorwiderstände. 220 Ohm sind ein guter Wert. Die LED blinkt nun in allen 3 möglichen Farben. ROT – GRÜN - ORANGE

29. Bicolor-LED (2 Farben-LED) 5mm



Verdrahten des Moduls



+ wird mit **D7** verbunden	220Ohm	Gelbe Leitung
- wird mit **GND** verbunden		Schwarze Leitung
S wird mit **D6** verbunden	220Ohm	Grüne Leitung

Software für die Bicolor-LED:

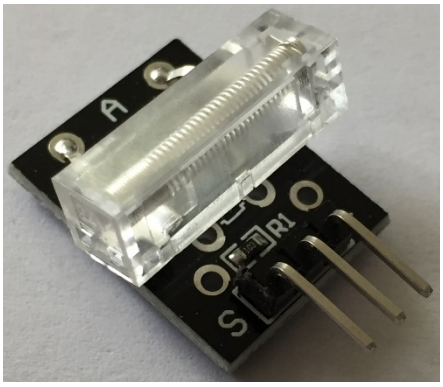
```
int LED1 = 7;
int LED2 = 6;

void setup(){
  pinMode(LED1, OUTPUT);
  pinMode(LED2, OUTPUT);
}

void loop(){
  digitalWrite(LED1, HIGH);
  digitalWrite(LED2, LOW);
delay(500);
  digitalWrite(LED1, LOW);
  digitalWrite(LED2, HIGH);
delay(500);
  digitalWrite(LED1, HIGH);
  digitalWrite(LED2, HIGH);
delay(500);
}
```

Der Code wird wieder Verifiziert  und Hochgeladen  .

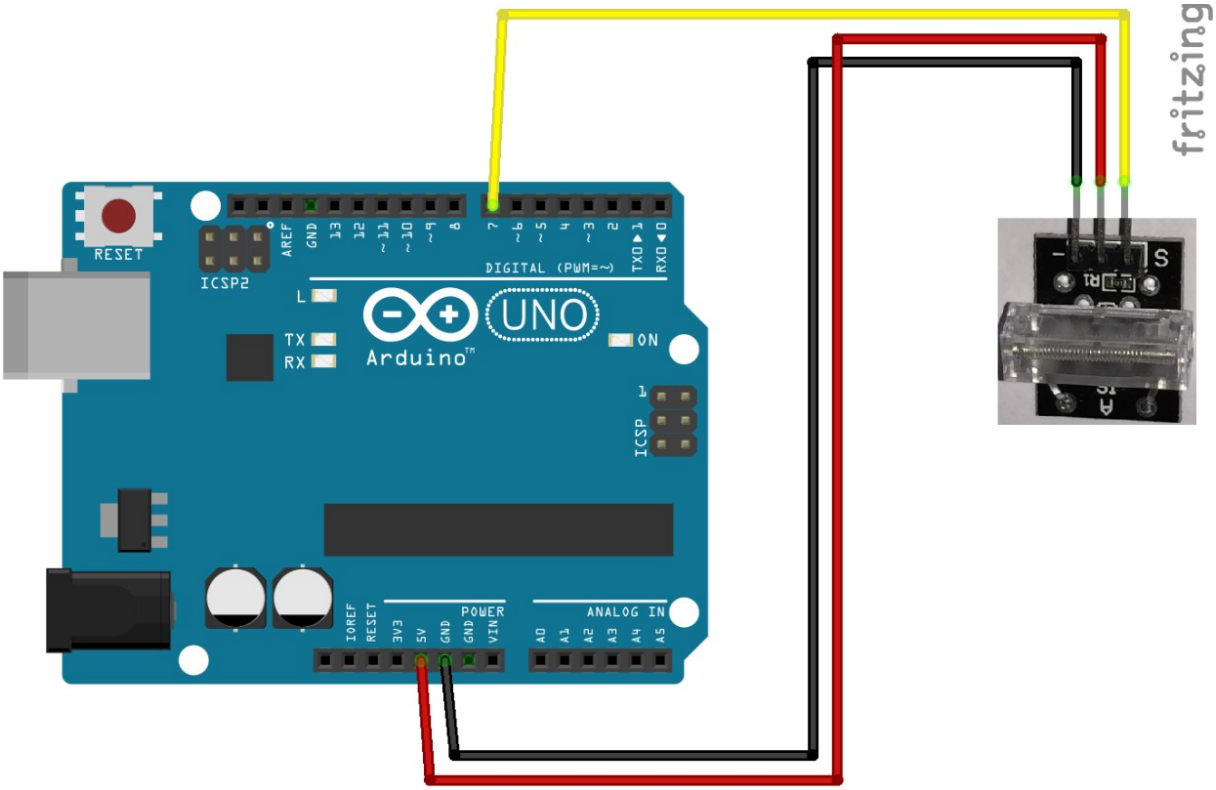
Die 2-Farben-LED mit 5mm Durchmesser benötigt auch Vorwiderstände. 220 Ohm sind auch hier ein guter Wert. Die LED blinkt nun wieder in allen 3 möglichen Farben.

ROT – GRÜN - ORANGE

30. Tap-Sensor



Verdrahten des Moduls



+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
S wird mit **D7** verbunden — Gelbe Leitung

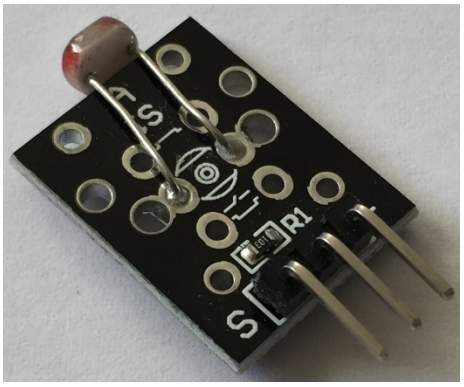
Software für den IR-Lichtschranke:

```
int Led = LED_BUILTIN;
int Eingang = 7;
int value;

void setup ()
{
pinMode (Led, OUTPUT);
pinMode (Eingang, INPUT);
}

void loop ()
{
value=digitalRead(Eingang);
digitalWrite (Led, value);
}
```

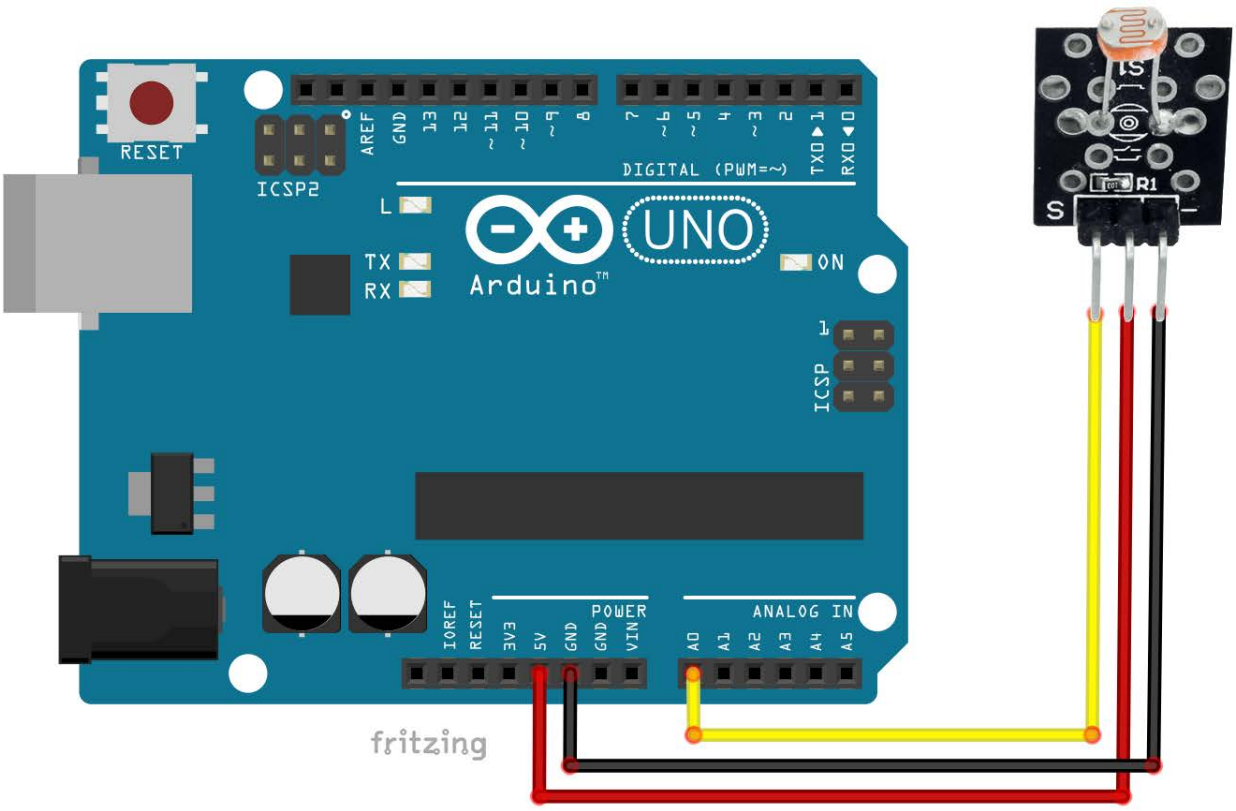
Der Code wird wieder Verifiziert  und Hochgeladen  .

Der Tap-Sensor reagiert auf sehr harte Stöße, er ist ähnlich dem Schocksensor, aber die Schläge müssen um einiges Stärker sein, damit dieser Sensor ein Signal gibt.

31. Lichtabhängiger Widerstand (LDR)



Verdrahten des Moduls



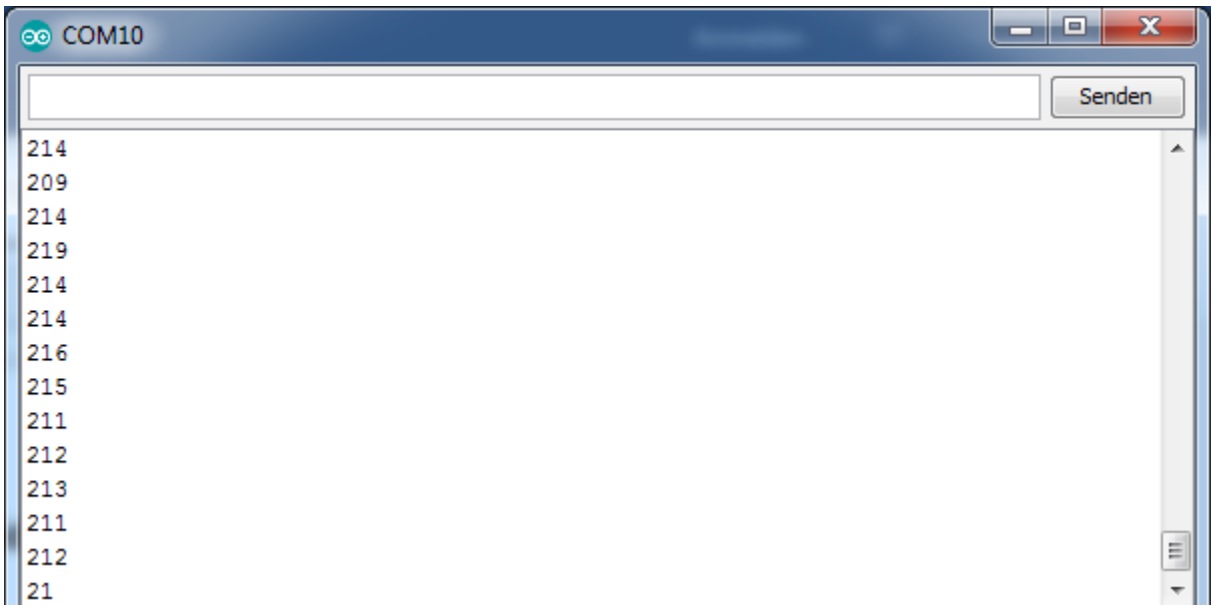
+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
S wird mit **A0** verbunden — Gelbe Leitung

Software für den LDR-Widerstand:

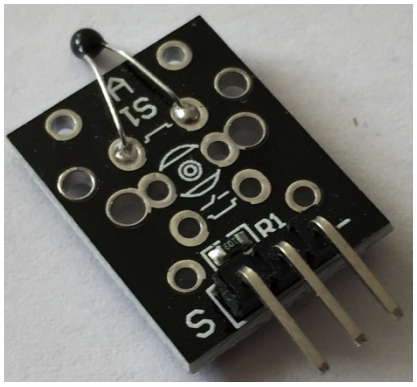
```
void setup() {
  Serial.begin(9600);
  pinMode(A0, INPUT);
}

void loop() {
  int LDR = analogRead(A0);
  Serial.println(LDR);
}
```

Der Code wird wieder Verifiziert und Hochgeladen.

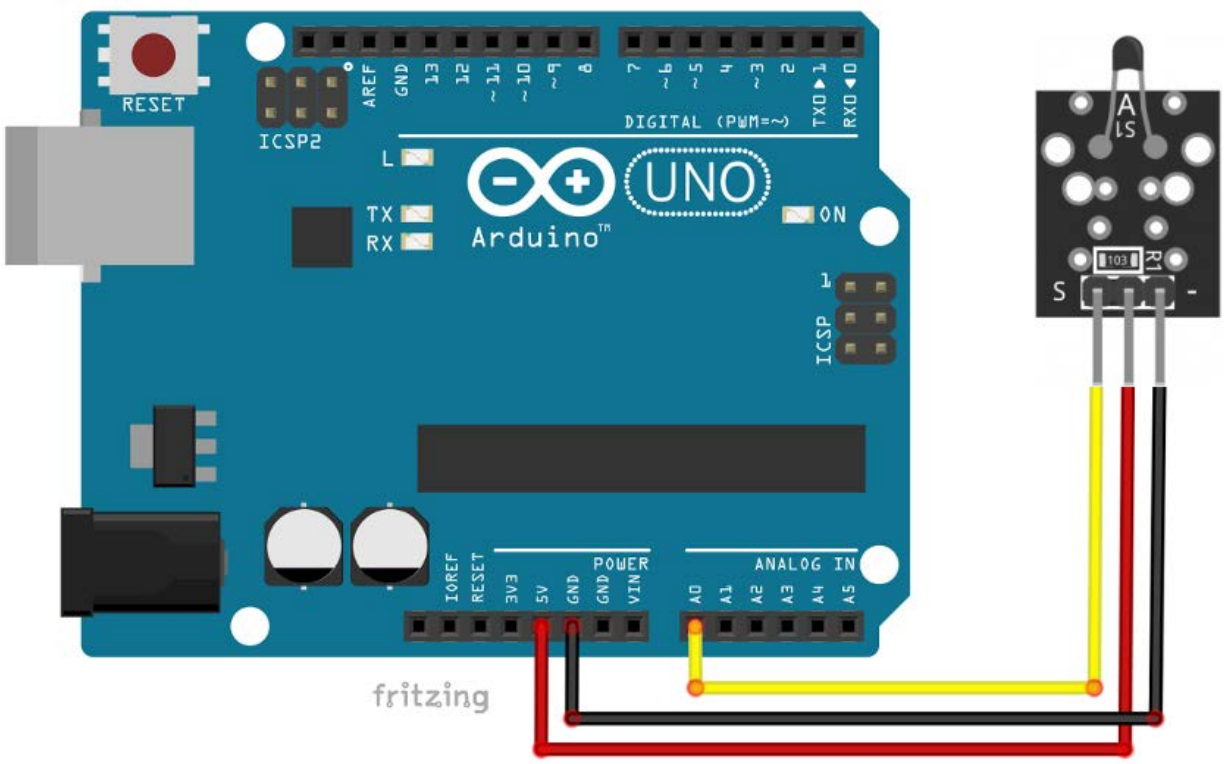
Der LDR-Widerstand ändert sich mit der Helligkeit, je dunkler es wird, desto größer ist der Widerstand. Im SerialMonitor wird der aktuelle Analogwert angezeigt.



32. Temperaturabhängiger Widerstand (Thermistor)



Verdrahten des Moduls



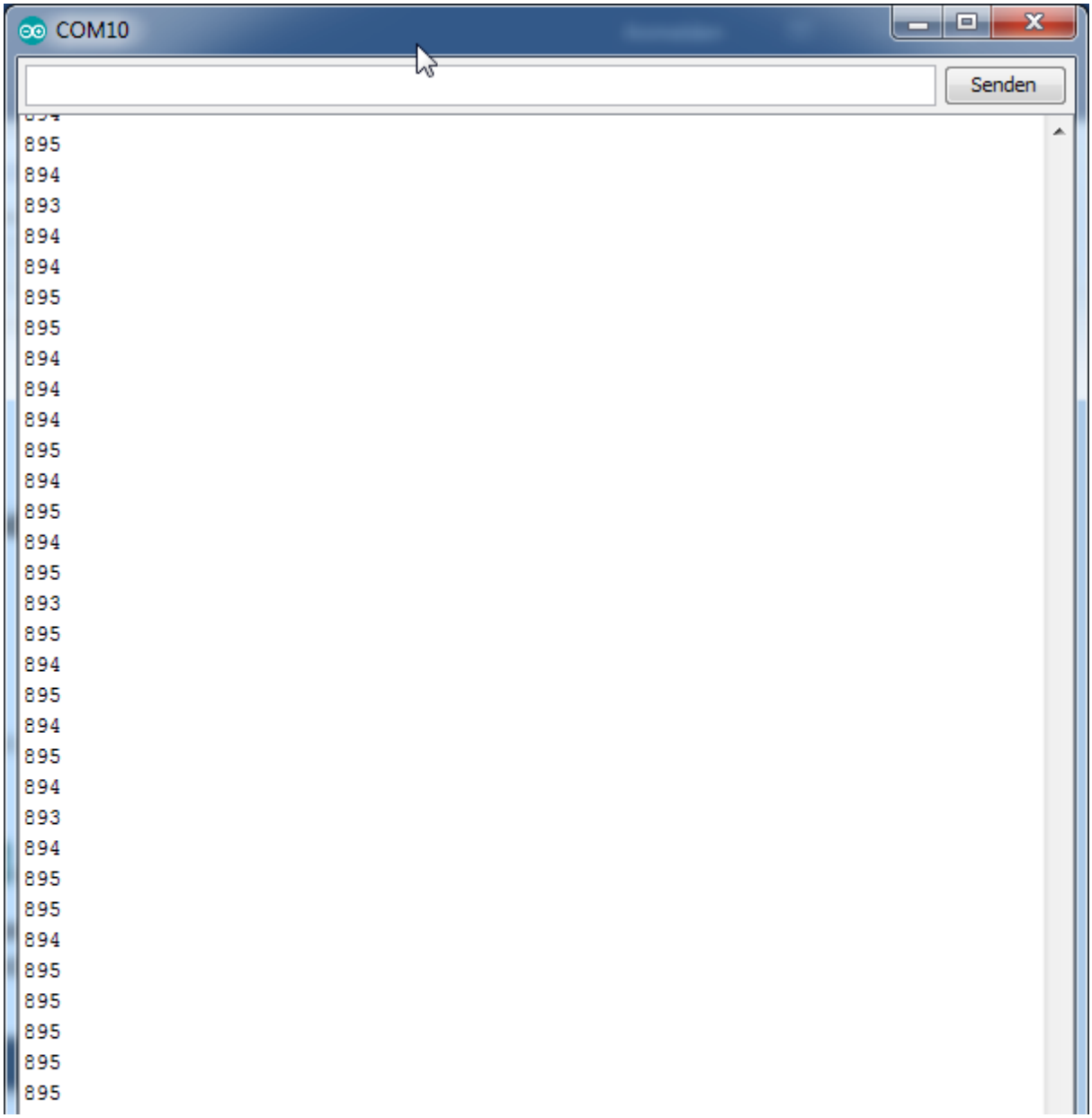
+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
S wird mit **A0** verbunden — Gelbe Leitung

Software für den Thermistor:

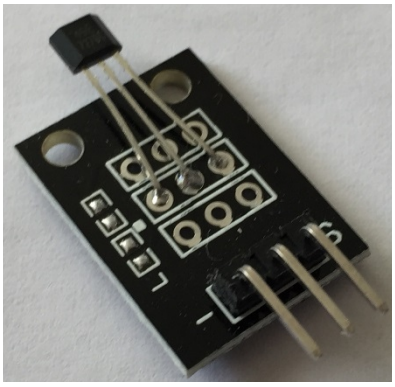
```
void setup() {
  Serial.begin(9600);
  pinMode(A0, INPUT);
}

void loop() {
  int Thermistor = analogRead(A0);
  Serial.println(Thermistor);
}
```

Der Code wird wieder Verifiziert und Hochgeladen .

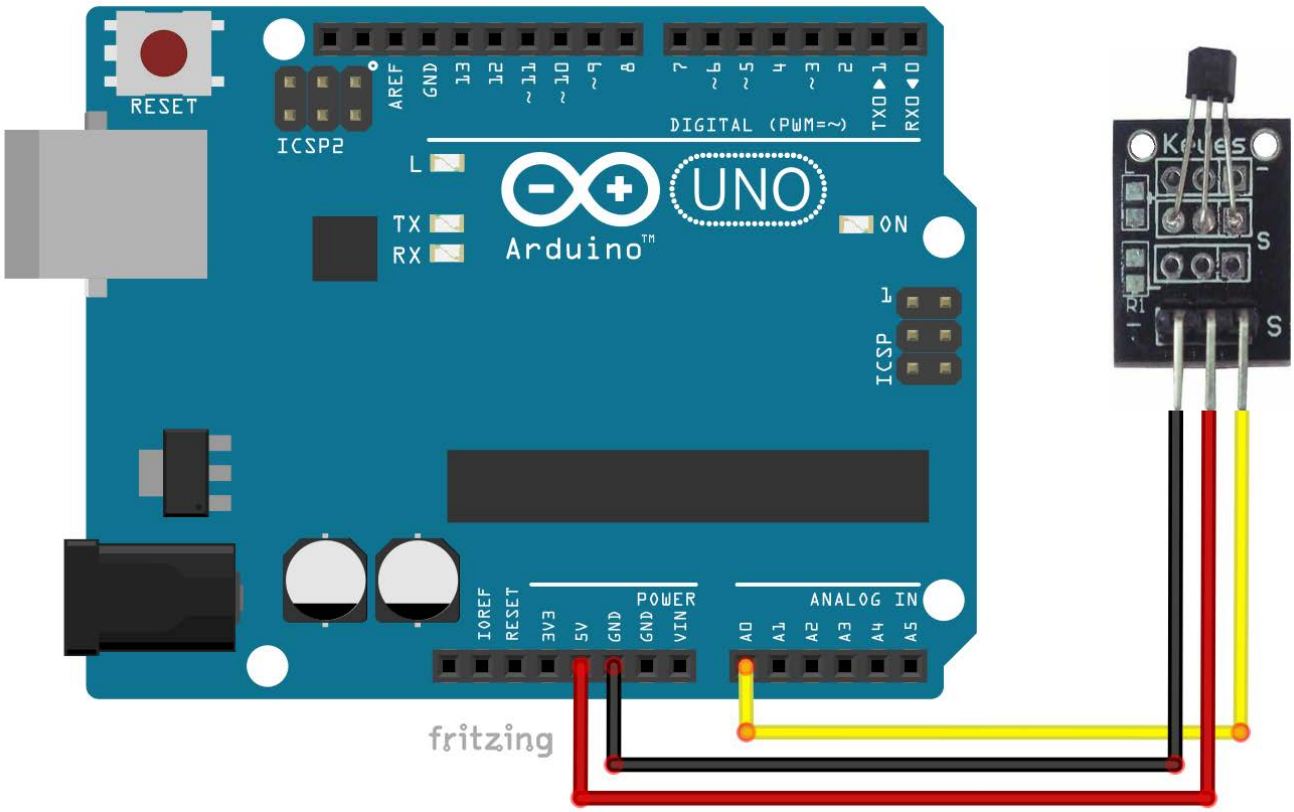
Der Temperaturabhängige-Widerstand ändert sich mit der Temperatur, je wärmer es wird, desto kleiner ist der Widerstand. Im SerialMonitor wird der aktuelle Analogwert angezeigt.



33. Hallsensor



Verdrahten des Moduls



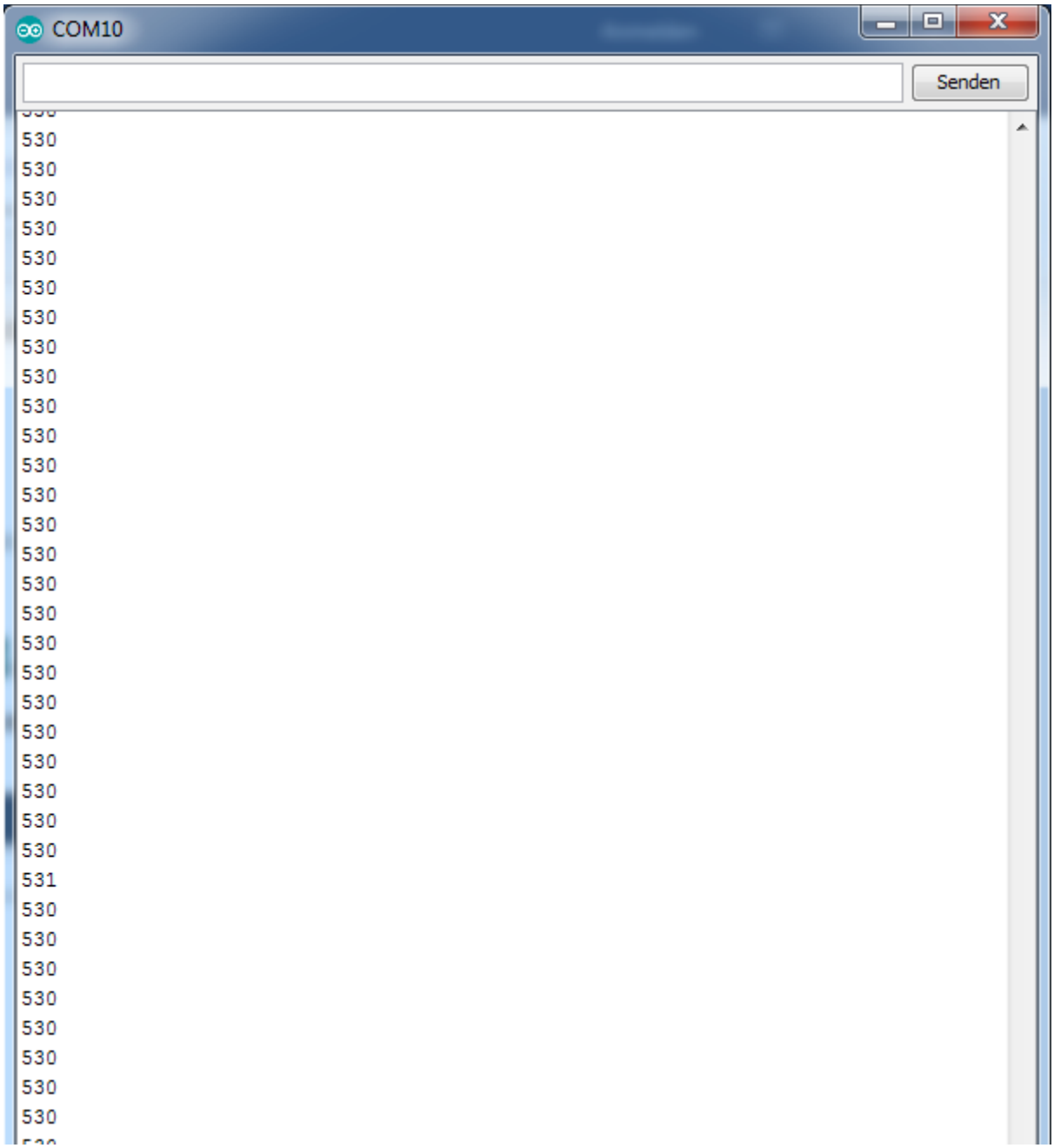
+ wird mit **5V** am Arduino verbunden — Rote Leitung
- wird mit **GND** verbunden — Schwarze Leitung
S wird mit **A0** verbunden — Gelbe Leitung

Software für den Hallsensor:

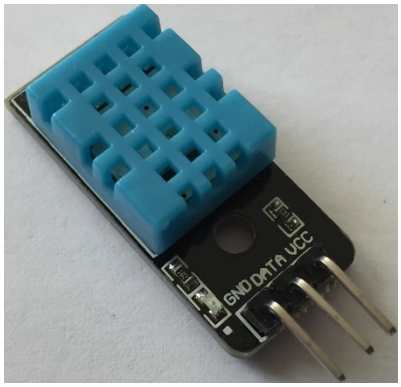
```
void setup() {
  Serial.begin(9600);
  pinMode(A0, INPUT);
}

void loop() {
  int Hallsensor = analogRead(A0);
  Serial.println(Hallsensor);
}
```

Der Code wird wieder Verifiziert  und Hochgeladen  .

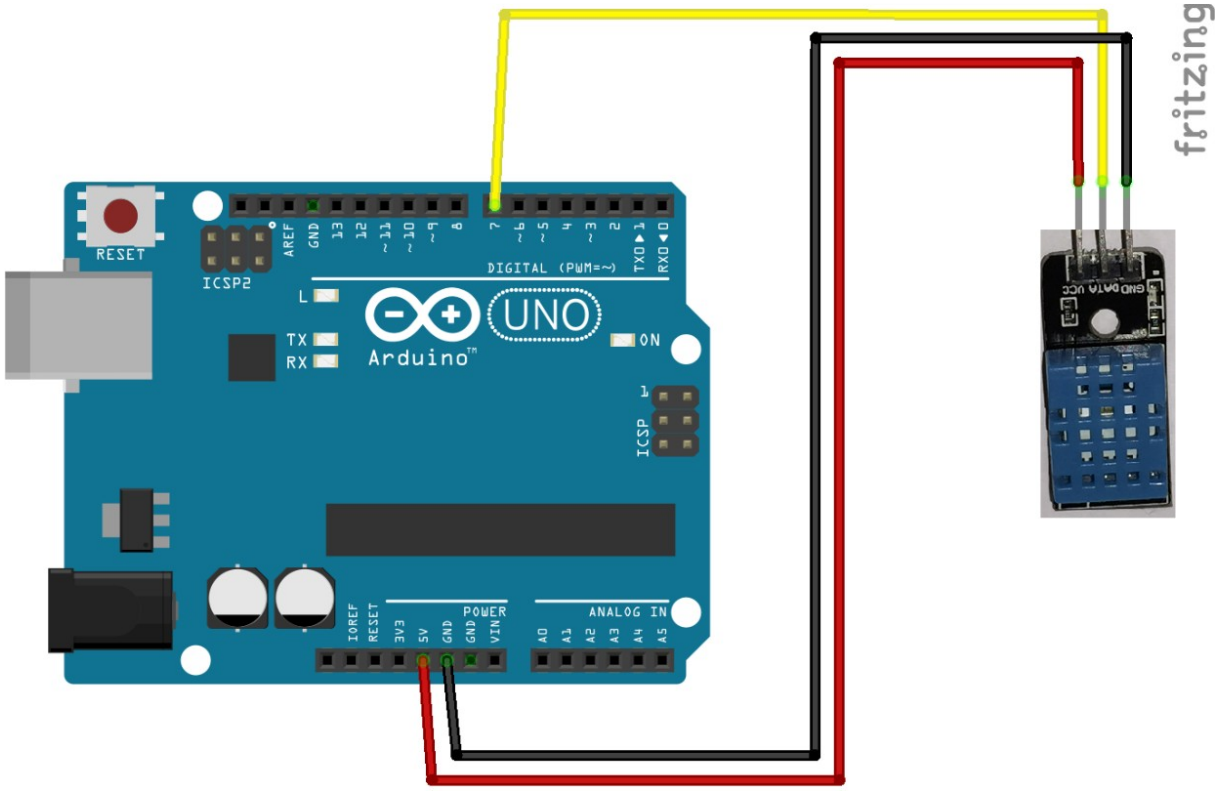
Der Hallsensor ändert seinen Wert in einem Magnetfeld. Im SerialMonitor wird der aktuelle Analogwert angezeit.



34. DHT11 Luftfeuchtigkeit und Temperatur



Verdrahten des Moduls

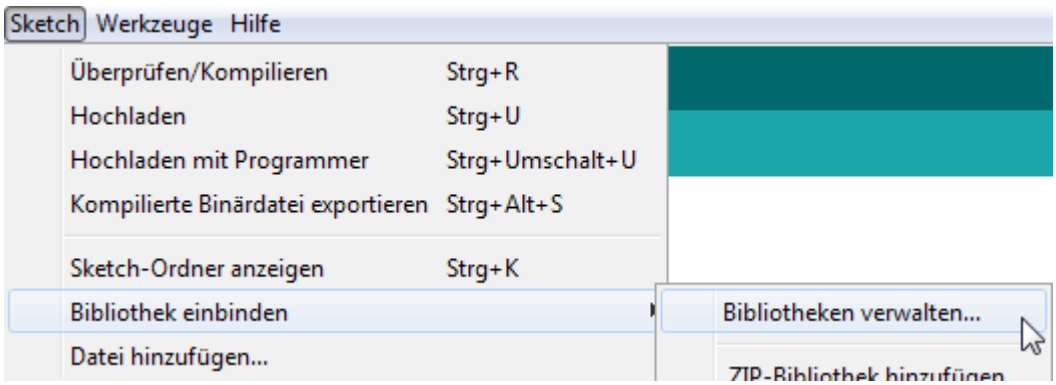


VCC wird mit **5V** am Arduino verbunden — Rote Leitung
GND wird mit **GND** verbunden — Schwarze Leitung
DATA wird mit **D7** verbunden — Gelbe Leitung

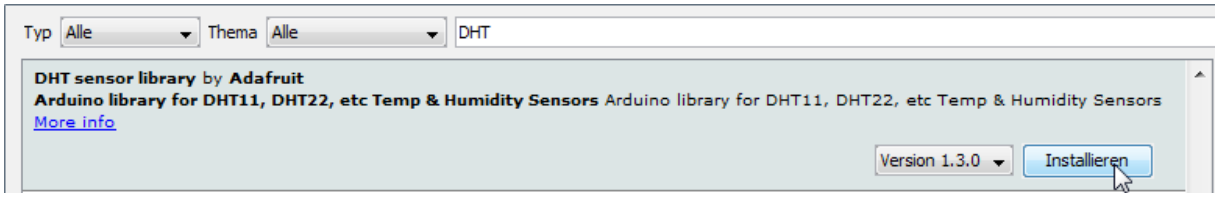
Software für den DHT11:

Für den Sensor benötigen wir noch eine Bibliothek. Diese installieren wir über die Bibliotheksverwaltung:

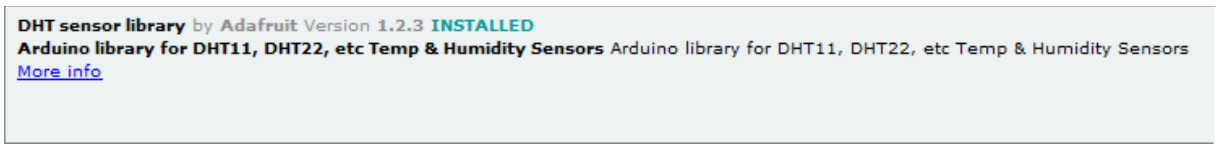
Sketch > Bibliothek einbinden > Bibliotheken verwalten



Darin suchen wir nach „DHT“ und wählen das DHT sensor library Paket von Adafruit aus und installieren es.

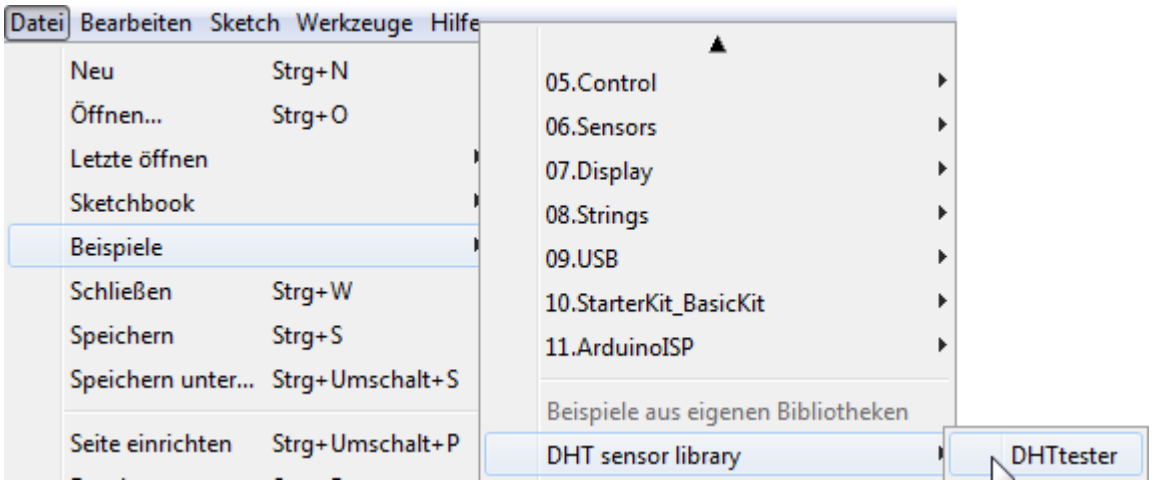


Anschließend steht neben dem Paket INSTALLED.



Nun können wir unseren Code schreiben:

Dazu starten wir unter Datei > Beispiele > DHT sensor library > DHTtester



Den Code müssen wir nur noch an unseren Sensor anpassen:

```
#define DHTPIN 7
#define DHTTYPE DHT11
```

Ansonsten kann der Code unverändert bleiben:

Hier der Code ohne Kommentare

```
#include "DHT.h"

#define DHTPIN 7
#define DHTTYPE DHT11   // DHT 11
//#define DHTTYPE DHT22   // DHT 22  (AM2302), AM2321

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  Serial.println("DHTxx test!");

  dht.begin();
}

void loop() {
  delay(2000);
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);
  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  float hif = dht.computeHeatIndex(f, h);
  float hic = dht.computeHeatIndex(t, h, false);

  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
  Serial.print(f);
  Serial.print(" *F\t");
  Serial.print("Heat index: ");
  Serial.print(hic);
  Serial.print(" *C ");
  Serial.print(hif);
  Serial.println(" *F");
}
```

Der Code wird wieder Verifiziert  und Hochgeladen  .

Nun werden über den SerialMonitor alle 2 Sekunden die Temperatur und Luftfeuchtigkeit ausgegeben.

COM10

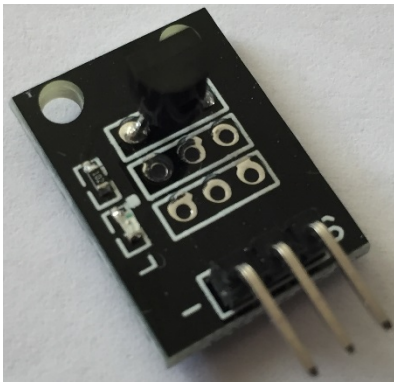
```
DHTxx test!
Humidity: 28.00 %	Temperature: 24.00 *C 75.20 *F	Heat index: 23.19 *C 73.74 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 29.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.11 *C 71.80 *F
Humidity: 29.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.11 *C 71.80 *F
Humidity: 29.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.11 *C 71.80 *F
Humidity: 29.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.11 *C 71.80 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 29.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.11 *C 71.80 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
```

Ist der Sensor falsch angeschlossen oder verliert die Verbindung, dann kommt die Fehlermeldung: Failed to read from DHT sensor.

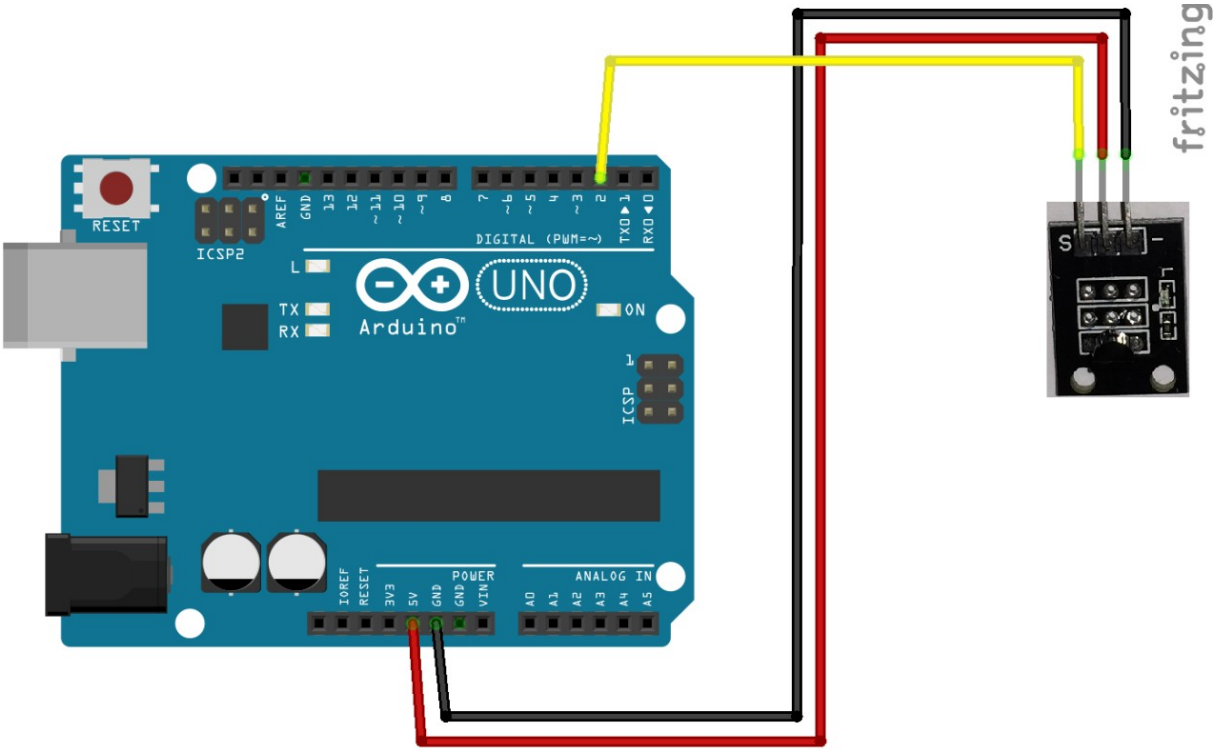
COM10

```
Humidity: 29.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.11 *C 71.80 *F
Humidity: 29.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.11 *C 71.80 *F
Humidity: 29.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.11 *C 71.80 *F
Humidity: 28.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.09 *C 71.76 *F
Humidity: 31.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.16 *C 71.90 *F
Humidity: 30.00 %	Temperature: 23.00 *C 73.40 *F	Heat index: 22.14 *C 71.85 *F
Failed to read from DHT sensor!
Failed to read from DHT sensor!
Failed to read from DHT sensor!
Failed to read from DHT sensor!
Failed to read from DHT sensor!
```

35. DALLAS 18B20 (DS18B20)



Verdrahten des Moduls

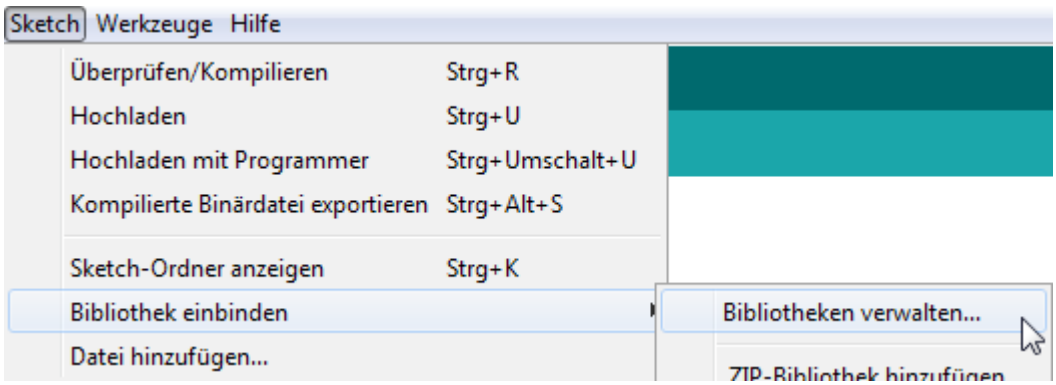


VCC wird mit **5V** am Arduino verbunden — Rote Leitung
GND wird mit **GND** verbunden — Schwarze Leitung
DATA wird mit **D7** verbunden — Gelbe Leitung

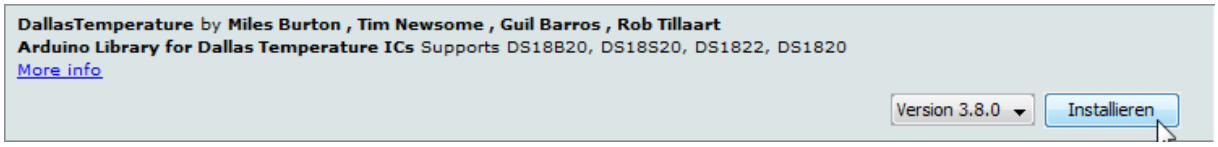
Software für den DS18B20:

Für den Sensor benötigen wir noch eine Bibliothek. Diese installieren wir über die Bibliotheksverwaltung:

Sketch > Bibliothek einbinden > Bibliotheken verwalten



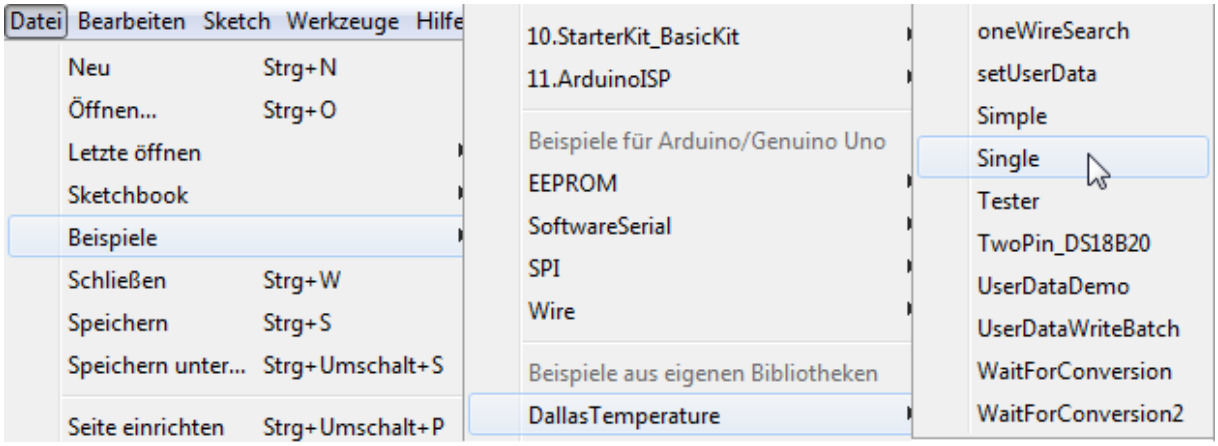
Darin suchen wir nach „DS18B20“ und wählen das DallasTemperature Paket aus und installieren es.



Anschließend steht neben dem Paket INSTALLED.

Nun können wir unseren Code schreiben,

dazu starten wir unter Datei > Beispiele >DallasTemperatur > Single



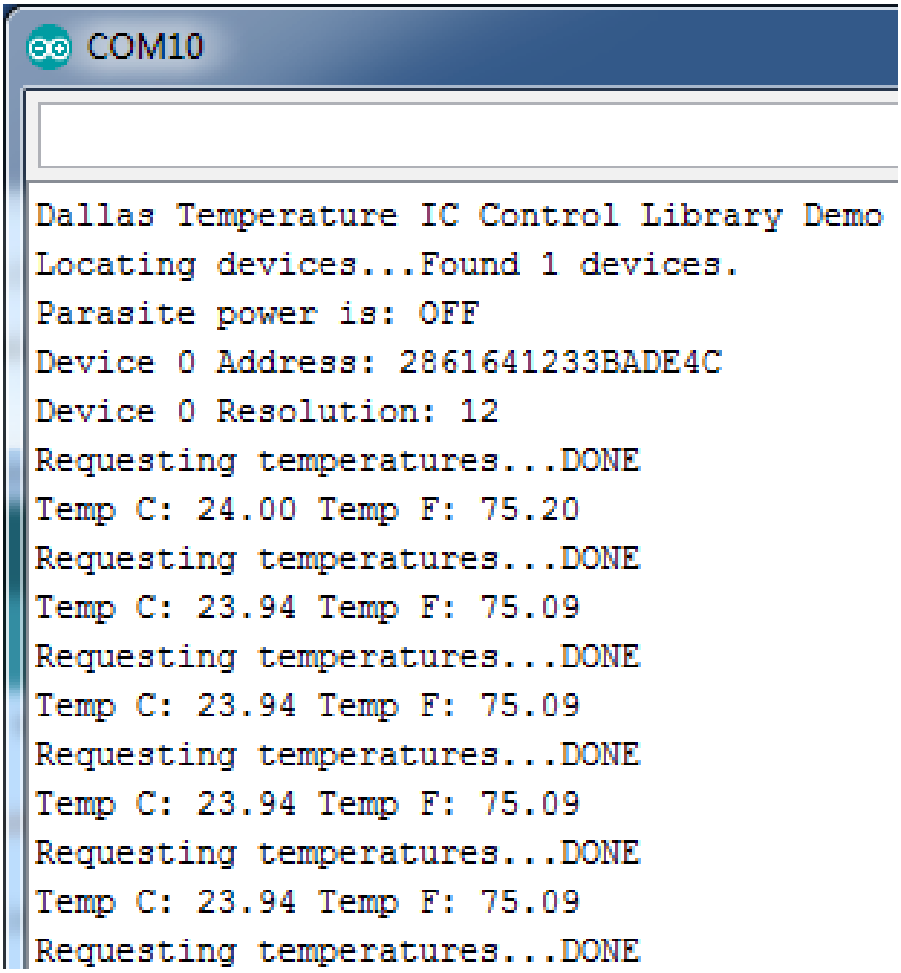
Den Code den wir bekommen können wir ohne Änderungen auf den Arduino laden.

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
DeviceAddress insideThermometer;

void setup(void)
{
  Serial.begin(9600);
  Serial.println("Dallas Temperature IC Control Library Demo");
  Serial.print("Locating devices...");
  sensors.begin();
  Serial.print("Found ");
  Serial.print(sensors.getDeviceCount(), DEC);
  Serial.println(" devices.");
  Serial.print("Parasite power is: ");
  if (sensors.isParasitePowerMode()) Serial.println("ON");
  else Serial.println("OFF");
  Serial.print("Device 0 Address: ");
  printAddress(insideThermometer);
  Serial.println();
  sensors.setResolution(insideThermometer, 9);
  Serial.print("Device 0 Resolution: ");
  Serial.print(sensors.getResolution(insideThermometer), DEC);
  Serial.println();
}

void printTemperature(DeviceAddress deviceAddress)
{
  float tempC = sensors.getTempC(deviceAddress);
  Serial.print("Temp C: ");
  Serial.print(tempC);
  Serial.print(" Temp F: ");
  Serial.println(DallasTemperature::toFahrenheit(tempC));
}

void loop(void)
{
  Serial.print("Requesting temperatures...");
  sensors.requestTemperatures();
  Serial.println("DONE");
  printTemperature(insideThermometer);
}

void printAddress(DeviceAddress deviceAddress)
{
  for (uint8_t i = 0; i < 8; i++)
  {
    if (deviceAddress[i] < 16) Serial.print("0");
    Serial.print(deviceAddress[i], HEX);
  }
}
```

Der Code wird wieder Verifiziert  und Hochgeladen  .

Nun wird über den SerialMonitor die Temperatur ausgegeben.

Zu Beginn werden noch die Adresse und ein paar Informationen über den Sensor ausgegeben.



Du hast es geschafft, deine Sensoren und Aktoren kannst du in deinen Projekten einsetzen und programmieren.

Ab jetzt heißt es lernen und eigene Projekte verwirklichen.

Und für mehr Hardware sorgt natürlich dein Online-Shop auf:

https://az-delivery.de

Viel Spaß!

Impressum
https://az-delivery.de/pages/about-us