Willkommen!

Und herzlichen Dank für den Kauf unseres DS18B20 Sensors! Auf den folgenden Seiten gehen wir mit dir gemeinsam die ersten Schritte von der Einrichtung bis zur Ausgabe der Werte.
Viel Spaß!

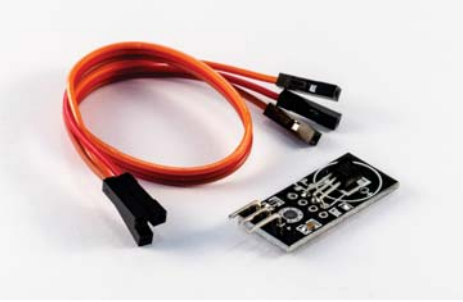
Der DS18B20 ist ein digitaler Temperatursensor, der über das proprietäre oneWire-Protokoll kommuniziert. Jeder Sensor hat eine einmalige Adresse (64-Bit) weshalb Sie mehrere Sensoren problemlos an einem Bus betreiben können.

Als der Sensor noch von Dallas kam gab es nur eine Variante, diese unterstütze sowohl den parasitären Modus, als auch den Normalbetrieb. Mittlerweile produziert Maxim die Sensoren in den Ausführungen DS18B20 und DS18B20-PAR. Beide unterstützen das 1-Wire Protokoll:
Das 1-Wire Protokoll hat seinen Namen daher, dass nur eine Leitung zur Kommunikation erforderlich ist anstatt wie sonst üblich mindestens zwei Datenleitungen, der PAR-Modus bleibt davon unberührt.

Wir haben also im Normalbetrieb drei Pins, oder Kabel: Data, VCC und GND. Über VCC und GND wird der Sensor mit Strom versorgt, über Data läuft die bidirektionale Kommunikation.

Der DS18B20-PAR unterstützt den parasitären Modus. Dieser wird so genannt, da der Sensor den Strom aus der Datenleitung schmarotzen kann. D.h. in diesem Betriebsmodus sind sogar nur 2 Leitungen erforderlich: GND und Data. Der VCC des ICs ist über einen Widerstand mit Data verbunden.

Da der DS18B20 ein weit verbreiteter Sensor ist stehen im Internet viele Anwendungsbeispiele zur Verfügung.



Die wichtigsten Informationen in Kürze

» **Abmessungen:** je nach Ausführung

» **Verbindung:**

VCC	3.1V-5.5V
GND	Masse
Data	Bidirektionale Datenleitung

» **Temperaturbereich:** -55 - +125 °C

» **Maximale Leistungsaufnahme:** 1,5mA

» **Programmierung über One-Wire Protokoll**

Auf den nächsten Seiten findest du Informationen zur
» ***Einrichtung der Hardware***
und eine Anleitung für
» ***das Auslesen der Sensordaten***.

Diese Anleitung setzt voraus, dass du weißt, wie du Sketche auf einen Arduino hochlädst und den Serial Monitor verwendest!

Alle Links im Überblick

»**Datenblatt:**
https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

» **Datenblatt:**
https://datasheets.maximintegrated.com/en/ds/DS18B20-PAR.pdf

» **Bibliothek Arduino:**
https://github.com/milesburton/Arduino-Temperature-Control-Library

Programmieroberflächen:

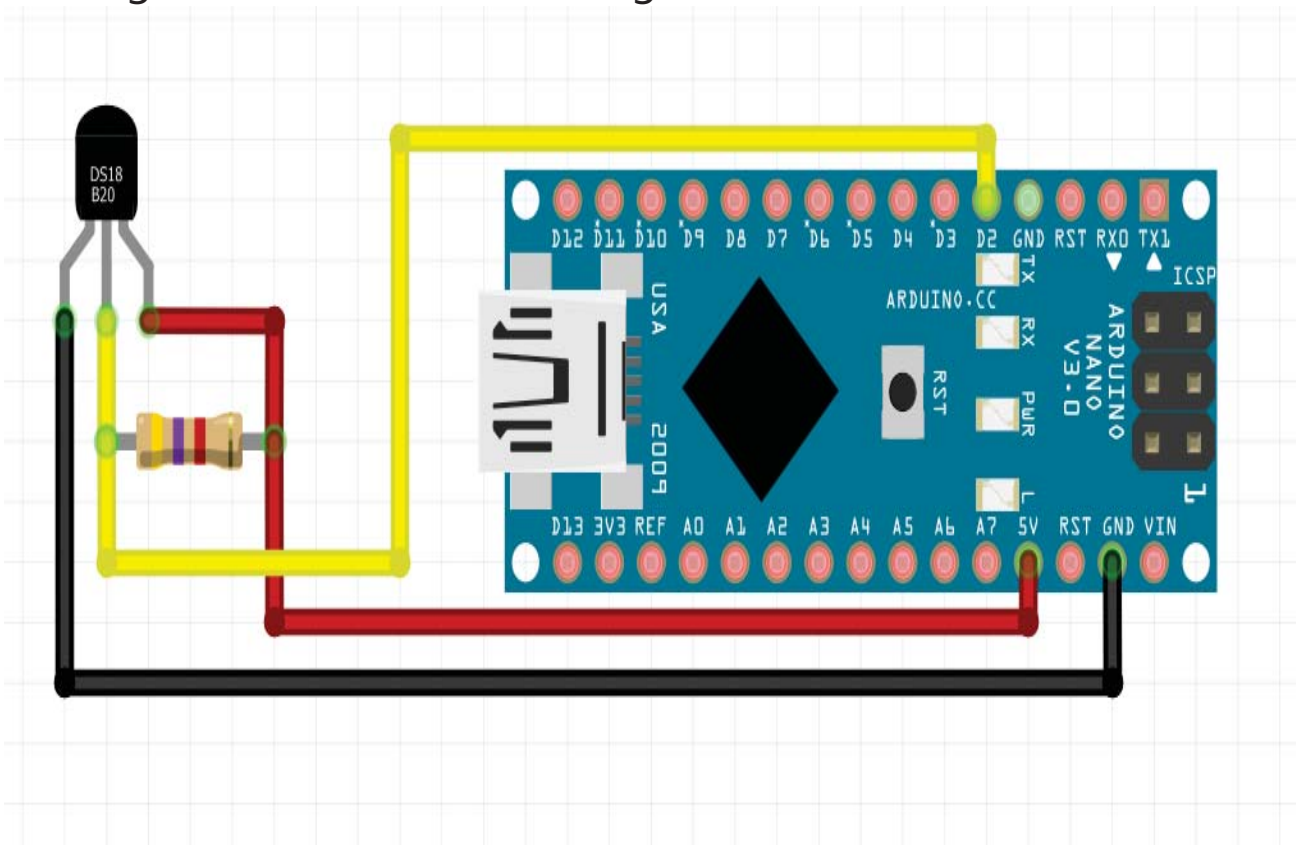
» Arduino IDE: *https://www.arduino.cc/en/Main/Software*
» Web-Editor: *https://create.arduino.cc/editor*
» Arduino-Erweiterung für SublimeText:
https://github.com/Robot-Will/Stino
» Arduino-Erweiterung "Visual Micro" für Atmel Studio oder Microsoft Visual Studio:
http://www.visualmicro.com/page/Arduino-for-Atmel-Studio.aspx
» *PlatformIO:* https://platformio.org/

Arduino Tutorials, Beispiele, Referenz, Community:

» *https://www.arduino.cc/en/Tutorial/HomePage*
» *https://www.arduino.cc/en/Reference/HomePage*

Einrichtung der Moduls am Arduino

Wir beginnen mit der Einrichtung der Hardware:



In unserem Anschlussbeispiel verbinden wir die Datenleitung über einen 4.7k Widerstand mit Pin D2 und an den 5V-Pin des Arduinos.

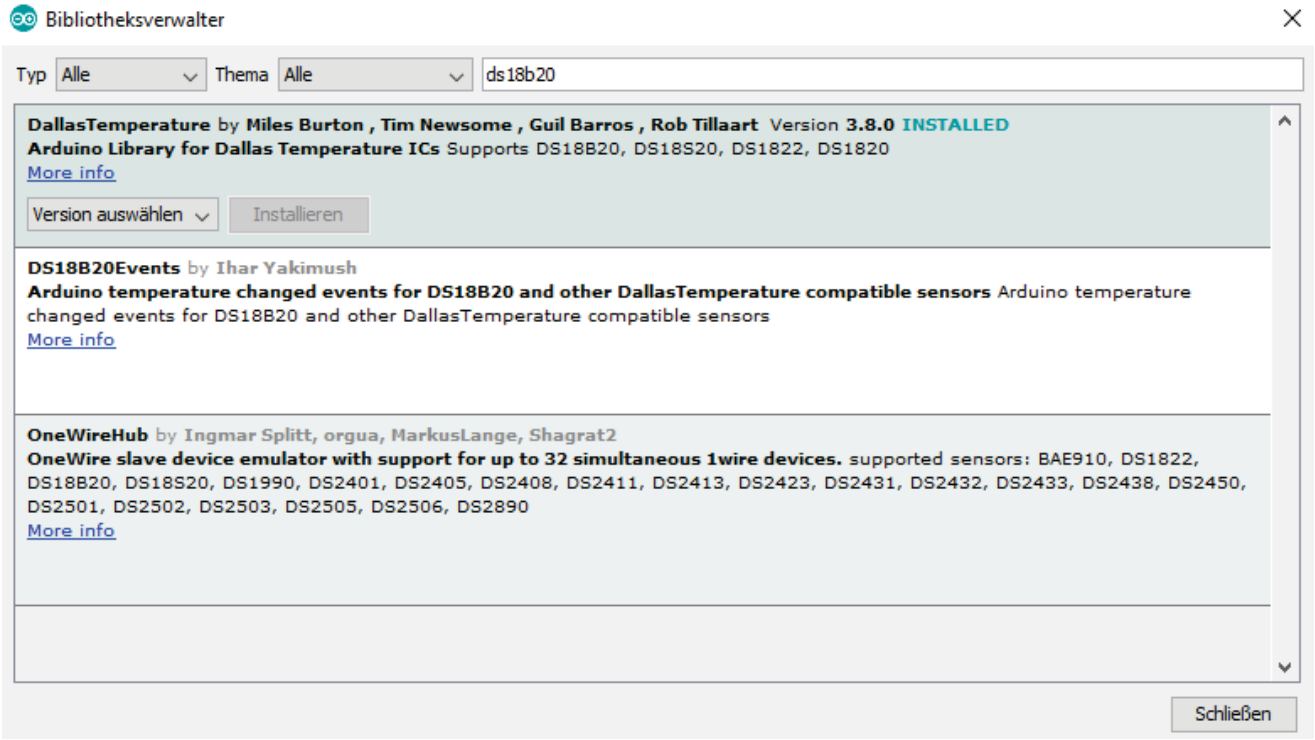
Haben Sie unseren DS18B20-Sensor auf einer Platine erworben, sind die Pins mit - / + / S beschriftet.

Bei unserer Industrieausführung des Sensors in Edelstahlhülse sind die Anschlusskabel (1m, 3m) in den üblichen Farben wie oben im Anschlussdiagramm zu sehen: GND = schwarz, VCC = rot, Data = gelb.

Die Installation der Libarys:

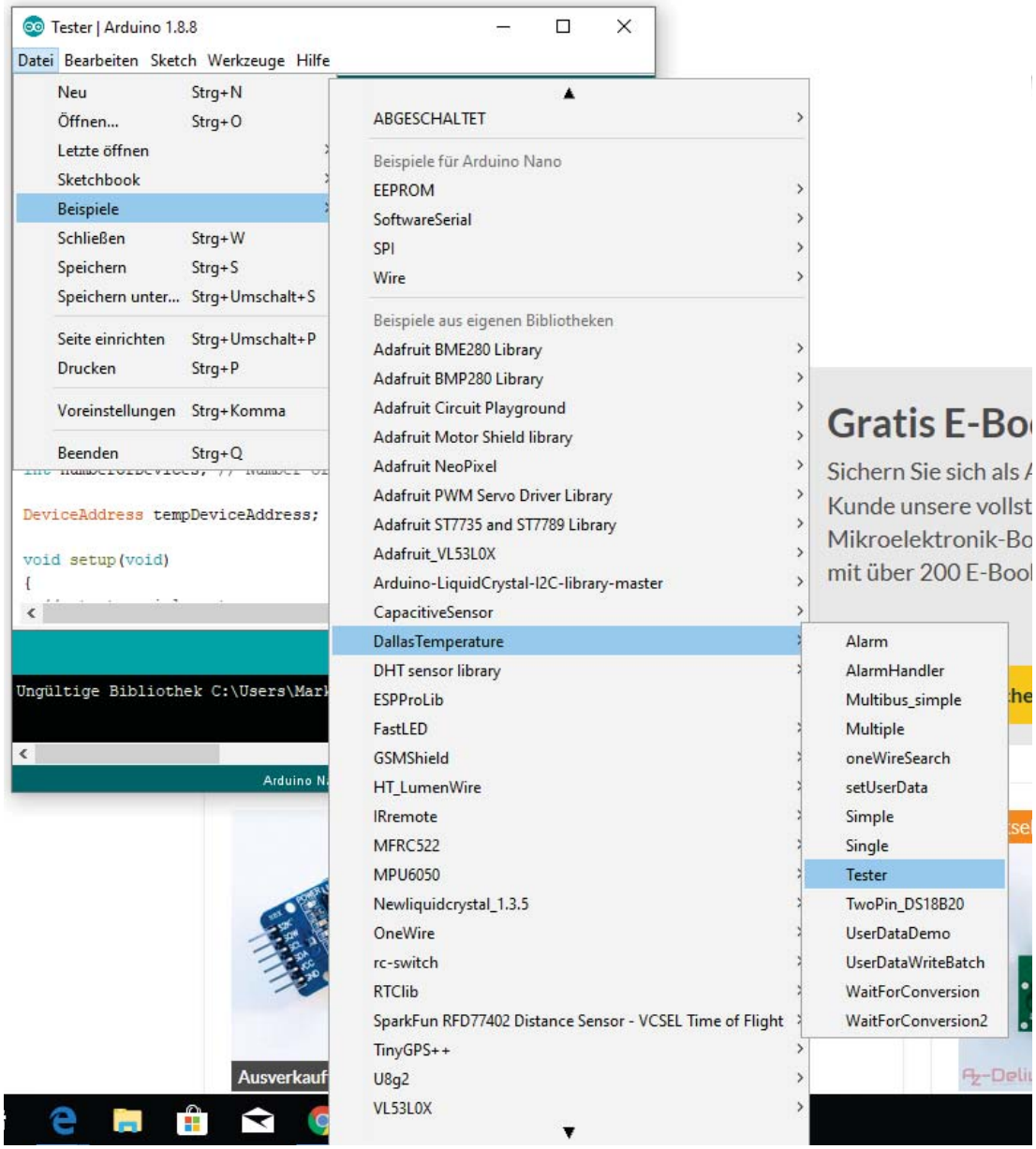
Libarys können auf verschiedene Arten in die ArduinoIDE importiert werden. Neben der Option die Libarys über den Bibliotheksverwalter oder einem Import über die Zip-Datei können wir diese auch direkt in das Arduino-Verzeichniss kopieren.

Da der DS18B20 jedoch ein recht verbreiteter Sensor ist steht das passende Paket im Bibliotheksverwalter zur Verfügung.



Nach der Installation starten wir die Arduino-IDE neu.

Im Anschluss stehen uns Beispiele zur Verfügung.



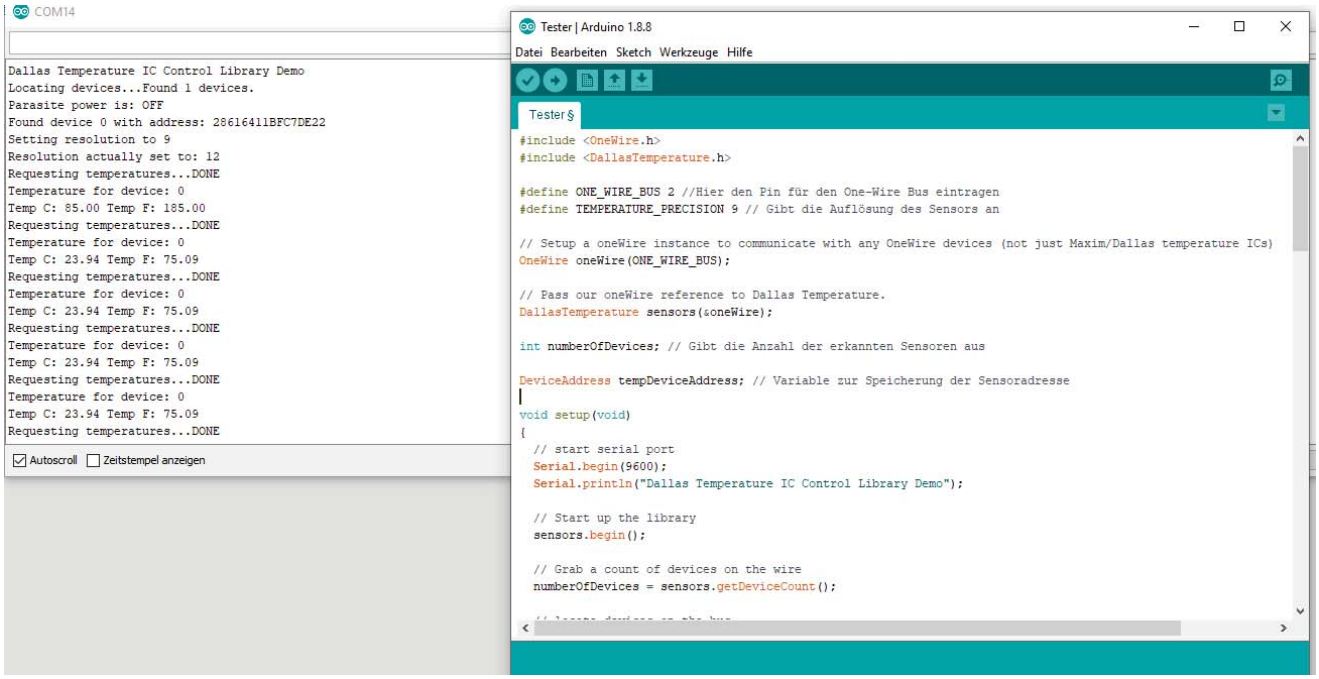
Das Auslesen der Sensordaten:

Der mitgelieferte Sketch „Tester“ liest die Anzahl angeschlossener Sensoren sowie die Auflösung, Adresse und Temperatur unseres Sensor aus und stellt die Informationen im seriellen Monitor dar. Weshalb der Sketch gut zur Überprüfung der Funktionsfähigkeit des Sensors und der Verdrahtung geeignet ist.

Sie finden diesen unter Datei → Beispiele → DallasTemperature → Tester

Im Header wird der Sensor konfiguriert:

```
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 2 //Hier den Pin für den One-Wire Bus eintragen
#define TEMPERATURE_PRECISION 9 // Gibt die Auflösung des Sensors an

// Setup a oneWire instance to communicate with any OneWire devices (not just Maxim/Dallas temperature ICs)
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);

int numberOfDevices; // Gibt die Anzahl der erkannten Sensoren aus

DeviceAddress tempDeviceAddress; // Variable zur Speicherung der Sensoradresse
```

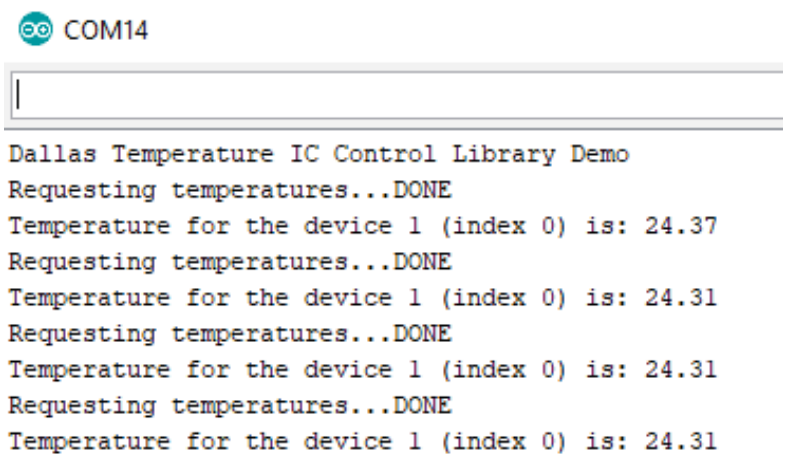


Der gezeigte Sketch eignet sich gut, weil darin der Abruf aller relevanter Daten gezeigt wird.

Der Minimalsketch um die Temperaturdaten im seriellen Monitor auszugeben sieht wie folgt aus:

```
#include <OneWire.h>
#include <DallasTemperature.h>

#define ONE_WIRE_BUS 2


OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

void setup(void)
{
  Serial.begin(9600);
  Serial.println("Dallas Temperature IC Control Library Demo");
  sensors.begin();
}


void loop(void)
{
  Serial.print("Requesting temperatures...");
  sensors.requestTemperatures(); // Send the command to get temperatures
  Serial.println("DONE");
  Serial.print("Temperature for the device 1 (index 0) is: ");
  Serial.println(sensors.getTempCByIndex(0));
}
```

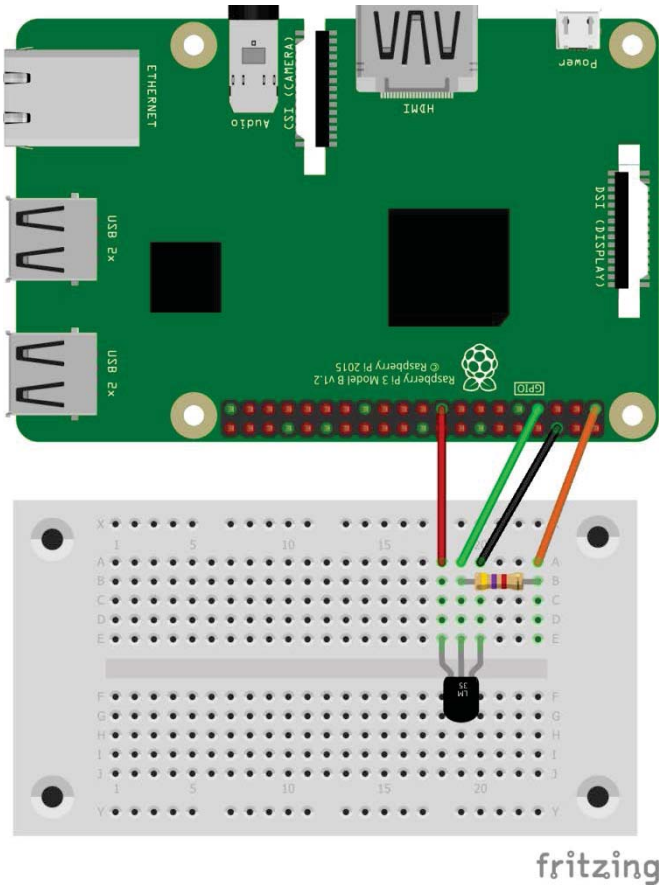
Im Anschluss sehen wir die Ausgabe der Temperaturwerte im seriellen Monitor:



Nachdem wir den DS18B20 nun am Arduino zum laufen gebracht haben, zeigen wir euch nun hier das gleiche mit dem Raspberry.

Das 1-Wire Protokoll wird am Raspberry nur am GPIO4 unterstützt. Deswegen muss der 1-Wire-Anschluss des Temperatursensors hier angeschlossen werden.

Bauen wir einmal die Schaltung auf.



Wenn alles entsprechend verkabelt ist, können wir das 1-Wire Protokoll damit aktivieren:

```
sudo modprobe w1-gpio
```

```
sudo modprobe w1-therm
```

Ob es geklappt hat, können wir herausfinden, indem wir folgendes eingeben:

```
lsmod
```

Die Module und müssten nun aufgelistet sein, falls nicht wird ein anderer GPIO benutzt oder es trat ein Fehler beim Aktivieren auf.

Damit nicht bei jedem Start die Module geladen werden, tragen wir sie in die Datei /etc/modules ein:

```
sudo nano /etc/modules
```

und fügen als letztes die folgenden zwei Zeilen ein:

```
w1_gpio
```

```
w1_therm
```

Außerdem muss man noch in der boot.txt den GPIO4 für 1-Wire reservieren:

```
sudo nano /boot/config.txt
```

und folgende Zeile am Ende ergänzen:

```
dtoverlay=w1-gpio,gpiopin=4
```

Nach einem Neustart wechseln wir in das W1-Bus Verzeichnis und lassen uns alle Sensoren anzeigen.

```
cd /sys/bus/w1/devices/
```

```
ls
```

Es sollte in der Ausgabe in etwa so ein Eintrag enthalten sein: **28-02162eb1fbee**

Diese Nummer muss man sich merken, das ist die ID des Temperatur Sensor. Als nächstes lesen wir den Sensor aus, wir geben diesen Befehl in die Konsole:

```
cat /sys/bus/w1/devices/28-02162eb1fbee/w1_slave
```

Als Antwort bekommen wir folgendes:

```
b8 01 4b 46 7f ff 0c 10 b1 : crc=b1 YES
b8 01 4b 46 7f ff 0c 10 b1 t=27500
```

in der 2. Zeile t=27500 ist unser Temperaturwert in Milligrad. Dieser Wert muss nur durch 1000 geteilt werden, 27000 : 1000 = 27,5. Die gemessene Temperatur ist 27,5°C.

Da diese Anzeige und Umrechnung umständlich ist, schreiben wir ein kleines Skript um nun noch den Befehl „temperatur" eingeben müssen und diesen dann sofort in Klartext angezeigt bekommen.

```
sudo nano /usr/bin/temperatur
```

Mit dem Inhalt:

```
#! /bin/bash
# Temperatur auslesen
tempread=`cat /sys/bus/w1/devices/28-02162eb1fbee/w1_slave`
# Wert Formatieren
temp=`echo "scale=2; "\`echo ${tempread##*=}\`" / 1000" | bc`
#Ausgabe
echo "Die gemessene Temperatur beträgt" $temp "°C"
```

Nun noch entsprechende Rechte geben

```
sudo chmod +x /usr/bin/temperatur
```

und wenn man nun in die Konsole

```
temperatur
```

eingibt, so erscheint die aktuell gemessene Temperatur. Falls eine Fehlermeldung kommt, so musst du wohl bc nachinstallieren.

```
sudo apt-get install bc
```

Wenn alles funktioniert dann sollte bei "temperatur" folgendes Ergebnis kommen:

```
Die gemessene Temperatur beträgt 27.5 °C
```

Im nächsten Schritt geben wir die Temperatur noch in einer Webseite aus.

Dazu erstellen wir eine neue PHP Datei:

```
touch temperatur.php
```

```
nano temperatur.php
```

Und geben folgenden Inhalt ein:

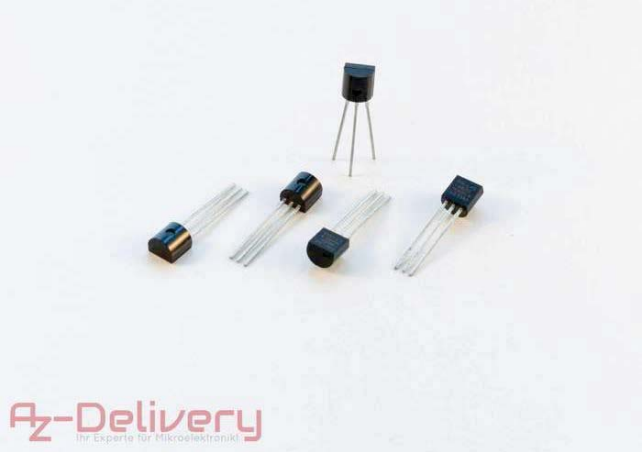
```
<?php
$handle = fopen("/sys/bus/w1/devices/w1_bus_master1/w1_master_slaves",
"r");
if ($handle) {
    while (($sensors = fgets($handle)) !== false) {
            $sensor = "/sys/bus/w1/devices/".trim($sensors)."/w1_slave";
            $sensorhandle = fopen($sensor, "r");
              if ($sensorhandle) {
                  $thermometerReading = fread($sensorhandle,
filesize($sensor));
                  fclose($sensorhandle);
                  // Auslesen der Temperatur nach dem t= in der 2. Zeile
                  preg_match("/t=(.+)/", preg_split("/\n/",
$thermometerReading)[1], $matches);
                  $celsius = $matches[1] / 1000; //umrechnen
                  $fahrenheit = $celsius*9/5+32;
                  print "$sensors = <b>$celsius &deg;C</b> / $fahrenheit
&deg;F<br>";
                  $sensors++;
              } else {
                 print "Sensor kann nicht gelesen werden";
              }
    }
    fclose($handle);
} else {
    print "Kein Sensor gefunden";
}
?>
```

Die Webseite zeigt nun beim Aufrufen:

28-02162eb1fbee = **27.437 °C** / 83.1866 °F

Und jetzt noch als Python Programm: temperatur.py

```
#!/usr/bin/python
# coding=utf-8
#import os, sys, time
from time import sleep
def aktuelleTemperatur():
    file = open('/sys/bus/w1/devices/28-02162eb1fbee/w1_slave')
    filecontent = file.read()                                   #1-wire Slave Datei
lesen
    file.close()
    stringvalue = filecontent.split("\n")[1].split(" ")[9]      #Temperaturwerte
    temperature = float(stringvalue[2:]) / 1000                 #auslesen und
konvertieren
    rueckgabewert = '%6.3f' % temperature                       #Temperatur ausgeben
    return(rueckgabewert)
while True:
        temperatur = aktuelleTemperatur()
        print "Aktuelle Temperatur : ", temperatur, "°C"
        sleep(2)
```



Du hast es geschafft! Herzlichen Glückwunsch!

Ab jetzt heißt es lernen und ausprobieren. Du weißt nun wie ein Mikrocontroller Temperaturen lesen kann. Jetzt kannst du versuchen die Werte praktisch einzusetzen.

Diesen Sensor und noch mehr Hardware findest du natürlich in deinem Online-Shop auf:

https://az-delivery.de

Viel Spaß!

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