

AZ-Delivery

Willkommen!

Vielen Dank, dass sie sich für unseren Temperatursensorsonde vom Typ "DS18B20" mit 3m Kabel von AZ-Delivery entschieden haben. In den nachfolgenden Seiten werden wir Ihnen erklären wie Sie das Gerät einrichten und nutzen können.

Have fun!



Az-Delivery

Der Temperatursensor DS18B20 ist in der Sonde hermetisch eingeschlossen. Die Sondenspitze besteht aus Edelstahl und eignet sich am besten für die Temperaturmessung in feuchten Umgebungen.

Der DS18B20 ist ein digitaler Temperatursensor, der digitale 9- bis 12-Bit-Temperaturmessungen liefert und über eine Alarmfunktion mit nichtflüchtigen, vom Benutzer programmierbaren oberen und unteren Triggerpunkten verfügt. Der Sensor kommuniziert über einen One-Wire-Bus, der nur einen Daten-, Stromversorgungs- und Masse-Pin für die Kommunikation mit einem Mikrocontroller benötigt. Darüber hinaus kann der Sensor direkt von der Datenleitung Strom beziehen (als "Parasite Power"-Modus bezeichnet), was eine externe Stromversorgung substituiert.

Jeder Sensor hat eine einzigartige, serielle 64-Bit-Adresse, wodurch mehrere Sensoren am selben One-Wire-Bus arbeiten können. So ist es uns möglich mit einem Mikrocontroller viele über einen großen Bereich verteilte Sensoren zu steuern. Anwendungen, die von dieser Funktion profitieren können, sind Temperaturüberwachungssysteme innerhalb von Gebäuden, Anlagen oder Maschinen, sowie Prozessüberwachungs- und Steuerungssysteme.

Der Sensor benötigt keine Standby-Stromversorgung, d.h. wenn keine Temperaturdaten gelesen werden, wird kein Strom verbraucht.

Der Messtemperaturbereich reicht von -55°C bis $+125^{\circ}\text{C}$ (67°F bis 257°F), mit einer Genauigkeit von $\pm 0,5^{\circ}\text{C}$ (9 Bit); $\pm 0,25^{\circ}\text{C}$ (10 Bit); $\pm 0,125^{\circ}\text{C}$ (11 Bit) und einer Auflösung von $\pm 0,0625^{\circ}\text{C}$ (12 Bit).



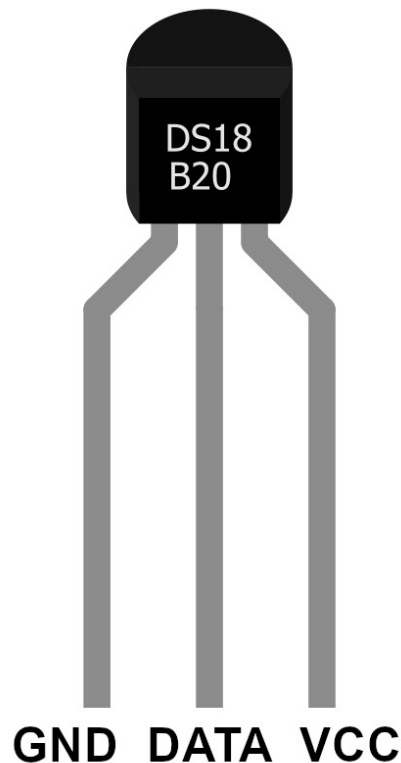
HINWEIS: Falls Kommunikationsprobleme auftreten, versuchen Sie, einen $4.7k\Omega$ Pull-up-Widerstand an den DATA-Pin anzuschließen.

HINWEIS: In den relevanten technischen Daten der One-Wire-Schnittstelle ist die maximale Anzahl der Sonden, die auf derselben Schnittstelle angeschlossen werden können, nicht erwähnt, aber in der praktischen Anwendung ist diese Anzahl nicht so hoch, und Sie sollten darauf achten.

HINWEIS: Es gibt eine Begrenzung der Kabellänge, die bei der Verwendung von Fernkommunikationsmitteln berücksichtigt werden sollte. Sie sollten auf die verteilte Kapazität und den Widerstand des Kabels achten.

HINWEIS: Der DS18B20 sieht gewöhnlichen Transistoren ähnlich, achten Sie also darauf, ihn nicht als Transistor zu verwenden, um Schäden zu vermeiden!

Pinbelegung



"VCC" pin - liefert Strom für die Sonde. Obwohl die Stromversorgung zwischen 3,3V und 5,5V liegen kann, wird eine 5V-Versorgung empfohlen. Im Falle einer 5V-Stromversorgung können Sie bis zu 20 Meter lange Kabel verwenden, um die Sonde und Mikrocontroller zu verbinden. Bei einer Stromversorgung von 3,3V darf die Kabellänge jedoch nicht größer als ein Meter sein. Andernfalls führt der Netzspannungsabfall zu Messfehlern.

"DATA" pin - ist ein Daten-Pin und wird für die Kommunikation zwischen der Sonde und dem Mikrocontroller verwendet (kann an die One-Wire-Schnittstelle angeschlossen werden).

"GND" pin - ist der Masse-Pin und sollte mit der gemeinsamen Masse oder 0V (bei Arduino oder Raspberry Pi) verbunden werden.

Einrichten der Arduino IDE

Falls Sie die Arduino IDE noch nicht installiert haben, gehen Sie auf den Link: <https://www.arduino.cc/en/Main/Software> und laden Sie die Installationsdatei für Ihre Betriebssystemplattform herunter.

Download the Arduino IDE



The screenshot shows the Arduino IDE download page. On the left, there is a teal circle with a white infinity symbol containing a minus and a plus sign. To its right, the text reads: **ARDUINO 1.8.9**, followed by a description of the IDE as open-source software written in Java, and a note that it can be used with any Arduino board. On the right side, there is a teal sidebar with links for different operating systems: Windows (installer and ZIP file), Windows app (with a 'Get' button), Mac OS X, and Linux (32 bits, 64 bits, ARM 32 bits, and ARM 64 bits). At the bottom of the sidebar are links for Release Notes, Source Code, and Checksums (sha512).

ARDUINO 1.8.9
The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software.
This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions.

Windows Installer, for Windows XP and up
Windows ZIP file for non admin install

Windows app Requires Win 8.1 or 10
[Get](#)

Mac OS X 10.8 Mountain Lion or newer

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

[Release Notes](#)
[Source Code](#)
[Checksums \(sha512\)](#)

Für *Windows* doppelklicken die Benutzer auf die heruntergeladene ".exe"-Datei und folgen den Anweisungen im Installationsfenster.

Az-Delivery

Für *Linux*-User, laden Sie eine Datei mit der Erweiterung *".tar.xz"* herunter, die Sie entpacken müssen. Wenn Sie sie extrahieren, gehen Sie in das extrahierte Verzeichnis und öffnen Sie das Terminal in diesem Verzeichnis. Sie müssen zwei *".sh"*-Skripte ausführen, das erste heißt *"arduino-linux-setup.sh"* und das zweite *"install.sh"*.

Um das Skript im Terminal auszuführen, führen Sie folgenden Befehl aus:

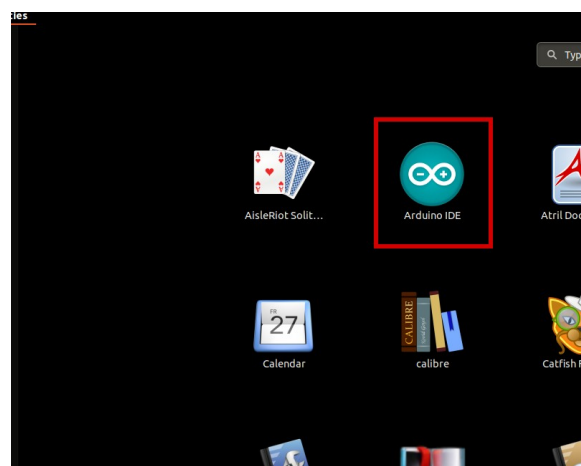
sh arduino-linux-setup.sh user_name

user_name - ist der Name des Super-Users im Linux-Betriebssystem. Sie werden aufgefordert, das Passwort für den Super-User einzugeben. Warten Sie ein paar Minuten, bis das Skript alles abgeschlossen hat.

Nach der Installation des ersten Skripts müssen Sie das zweite Skript namens *"install.sh"*-Skript ausführen. Führen Sie folgenden Befehl aus:

sh install.sh

Nach der Installation dieser Skripte gehen Sie zu "All Apps", wo Sie die installierte Arduino IDE finden.



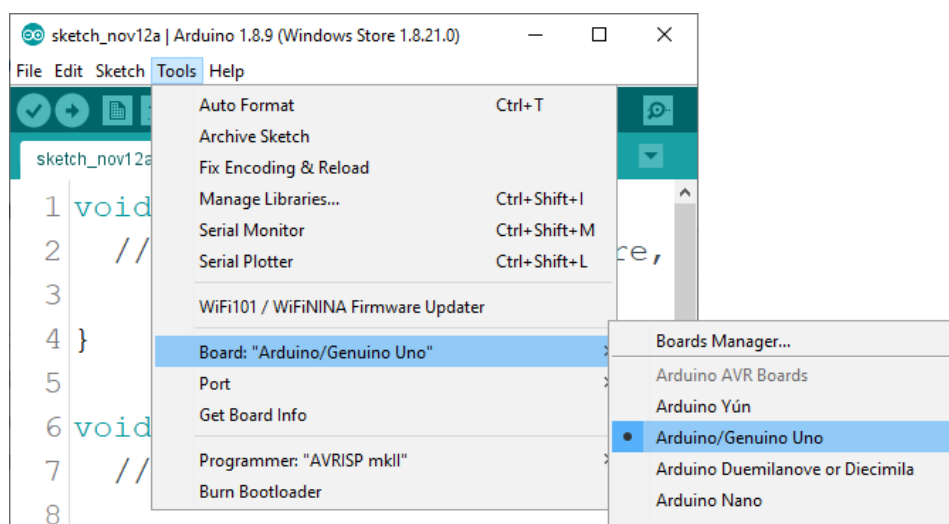
Az-Delivery

Auf fast allen Betriebssystemen ist ein Texteditor vorinstalliert (*Windows* kommt mit *Notepad*, *Linux/Ubuntu* mit *Gedit*, *Linux/Raspbian* mit *Leafpad* usw.). Alle diese Texteditoren sind für den Zweck des eBooks vollkommen in Ordnung.

Als nächstes sollten Sie überprüfen, ob Ihr PC Ihr Arduino-Board erkennen kann. Öffnen Sie die Arduino IDE, und gehen Sie zu:

Tools > Board > {your board name here}

{your board name here} sollte der *Arduino/Genuino Uno* sein, wie Sie auf dem Bild unten sehen können:



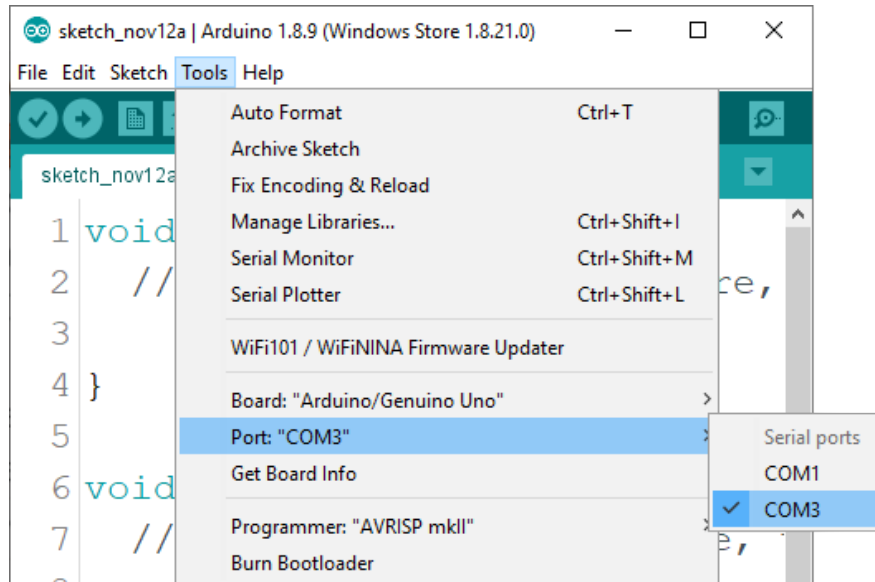
Sie müssen den Port auswählen, an dem das Arduino-Board angeschlossen ist. Gehen Sie zu

Tools > Port > {port name goes here}

und wenn Sie das Arduino-Board an den USB-Port angeschlossen haben, sollten Sie einen Portnamen sehen.

Az-Delivery

Wenn Sie die Arduino IDE in Windows verwenden, lauten die Portnamen wie folgt:



Für Linux-User lautet der Portname beispielsweise `"/dev/ttyUSBx"`, „x“ kann eine ganze Zahl zwischen 0 und 9 repräsentieren.



Wie Sie den Raspberry Pi und Python einrichten

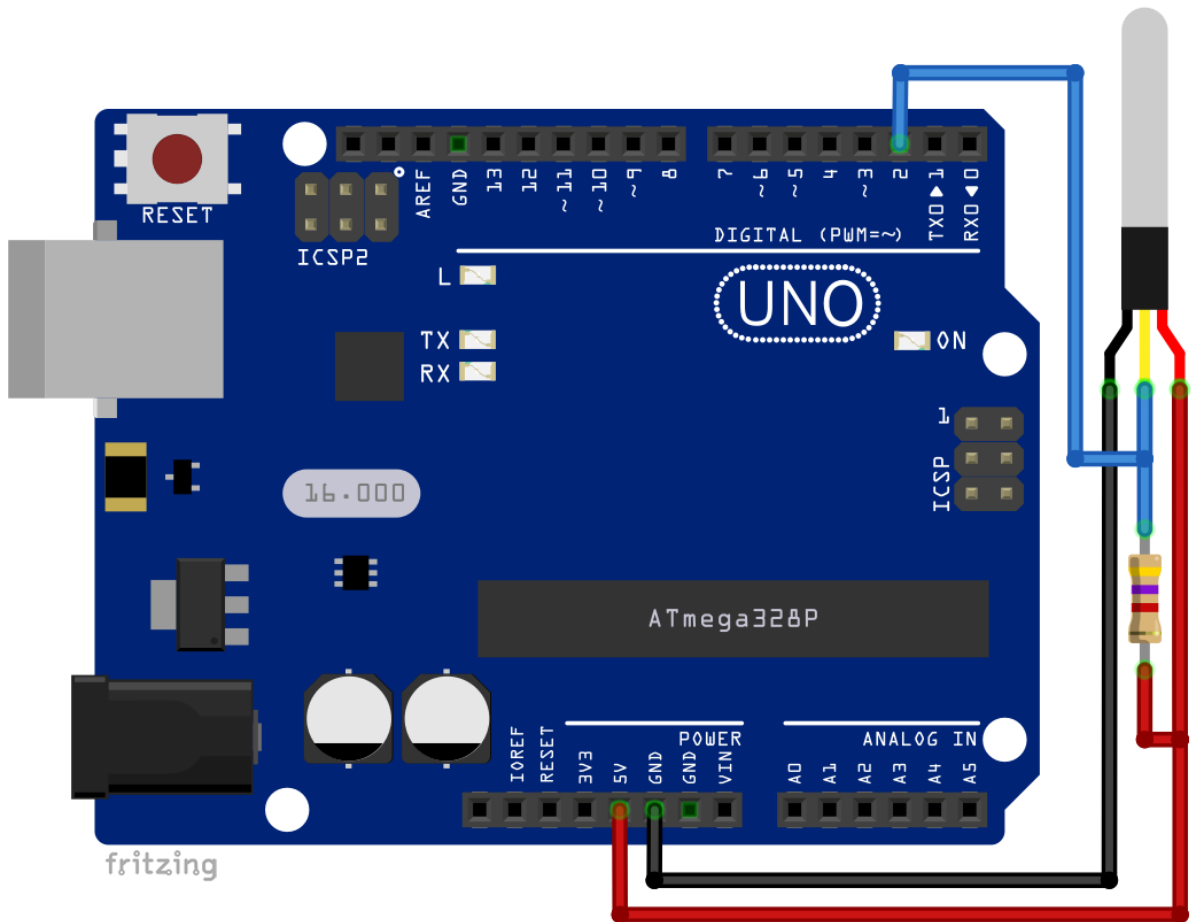
Für den Raspberry Pi müssen Sie zuerst das Betriebssystem installieren und dann alles so einrichten, dass Sie es im "*Headless*"-Modus verwenden können. Im "*Headless*"-Modus können Sie eine Verbindung zur Raspberry Pi herstellen, ohne dass Sie einen PC-Bildschirm, eine Maus oder eine Tastatur benötigen. Das Einzige, was Sie für diesen Modus benötigen, sind ein Raspberry Pi, eine Stromversorgung und eine Internetverbindung. All dies wird ausführlich in dem kostenlosen eBook "Raspberry Pi Quick Startup Guide" erklärt, das Sie auf unserer Website finden:

<https://www.az-delivery.de/products/raspberry-pi-kostenfreies-e-book?ls=en>

Das Betriebssystem *Rasbian* wird vorinstalliert mit Python ausgeliefert.

Verbindung der Sonde mit dem Uno

Verbinden Sie die DS18B20-Sonde mit der Uno-Platine wie unten dargestellt:



DS18B20 pin	>	Uno pin
DATA	>	D2
VCC	>	5V
GND	>	GND

Blauer Draht

Roter Draht

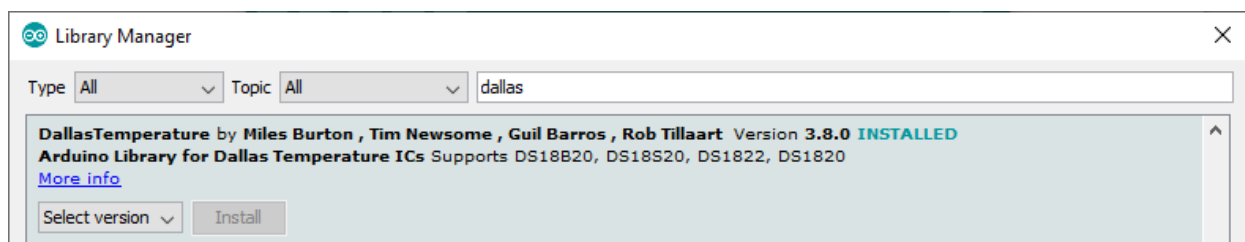
Schwarzer Draht

Hinweis: Ein Pull-Up-Widerstand mit $4.7k\Omega$ ist zwischen dem *DATA*-Pin und *VCC* angeschlossen.

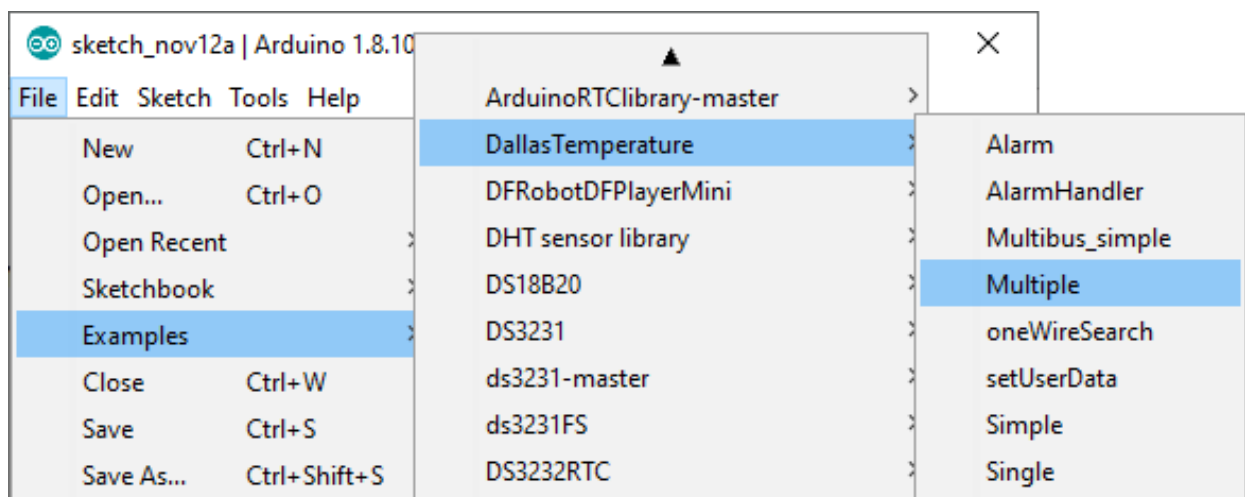
Az-Delivery

Um die Sonde DS18B20 mit Arduino-Boards verwenden zu können, müssen wir zunächst eine Bibliothek dafür herunterladen. Gehen Sie dazu: Werkzeuge > Bibliotheken verwalten, es erscheint ein neues Fenster. Geben Sie "Dallas" in das Suchfeld ein und laden Sie die Bibliothek "DallasTemperature" von "Miles Burton, Tim Newsome Guil Barros und Rob Tillaart" herunter, wie auf dem folgenden Bild zu sehen ist:

Die Version der benutzten Bibliothek ist **3.7.5**



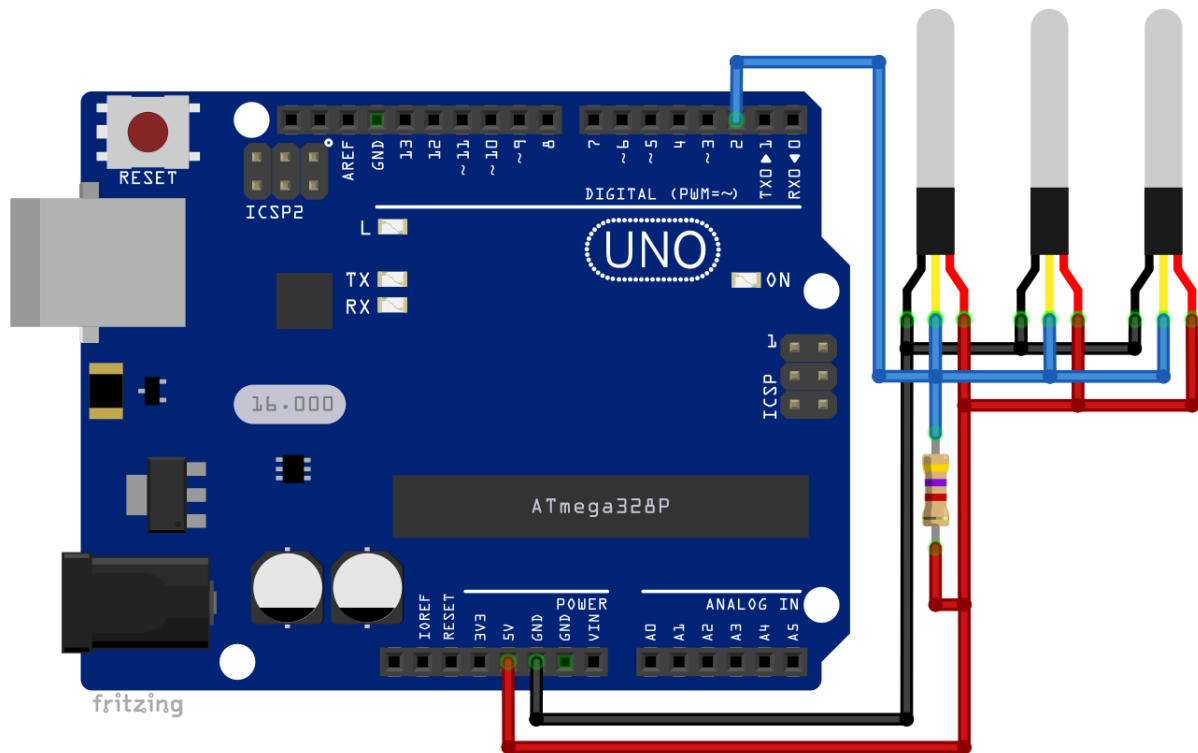
Gehen Sie zu *File > Examples > DallasTemperature > ...* dort finden Sie viele Sketch-Beispiele. Wir werden einen Sketch namens "*Multiple*" verwenden und modifizieren, um Temperaturdaten von drei verschiedenen DS18B20-Sonden zu lesen.



Beim Verwenden einer DS18B20-Sonde, ist der „*Simple*“-Sketch genügend.

Az-Delivery

Schließen Sie drei DS18B20-Sonden an die Uno-Platine an, wie unten dargestellt:



Nachfolgend finden Sie ein Sketch-Beispiel für drei DS18B20-Sonden auf derselben One-Wire-Schnittstelle:

```
#include <OneWire.h>
#include <DallasTemperature.h>
#define ONE_WIRE_BUS 2 // Data wire is plugged into D2 pin
#define TEMPERATURE_PRECISION 12
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);
DeviceAddress one, two, three;
```

Az-Delivery

```
void setup() {
  Serial.begin(9600);
  Serial.println("Dallas Temperature IC Control Library Demo");
  sensors.begin();
  Serial.print("Locating devices...");
  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");
  }
  if(!sensors.getAddress(one, 0)) {
    Serial.println("Unable to find address for Device 0"); }
  if(!sensors.getAddress(two, 2)) {
    Serial.println("Unable to find address for Device 2"); }
  if(!sensors.getAddress(three, 1)) {
    Serial.println("Unable to find address for Device 1"); }

  Serial.print("Device 0 Address: ");
  printAddress(one);
  Serial.println(); Serial.print("Device 1 Address: ");
  printAddress(two);
  Serial.println(); Serial.print("Device 2 Address: ");
  printAddress(three); Serial.println();
  sensors.setResolution(one, TEMPERATURE_PRECISION);
  sensors.setResolution(two, TEMPERATURE_PRECISION);
  sensors.setResolution(three, TEMPERATURE_PRECISION);
}
```

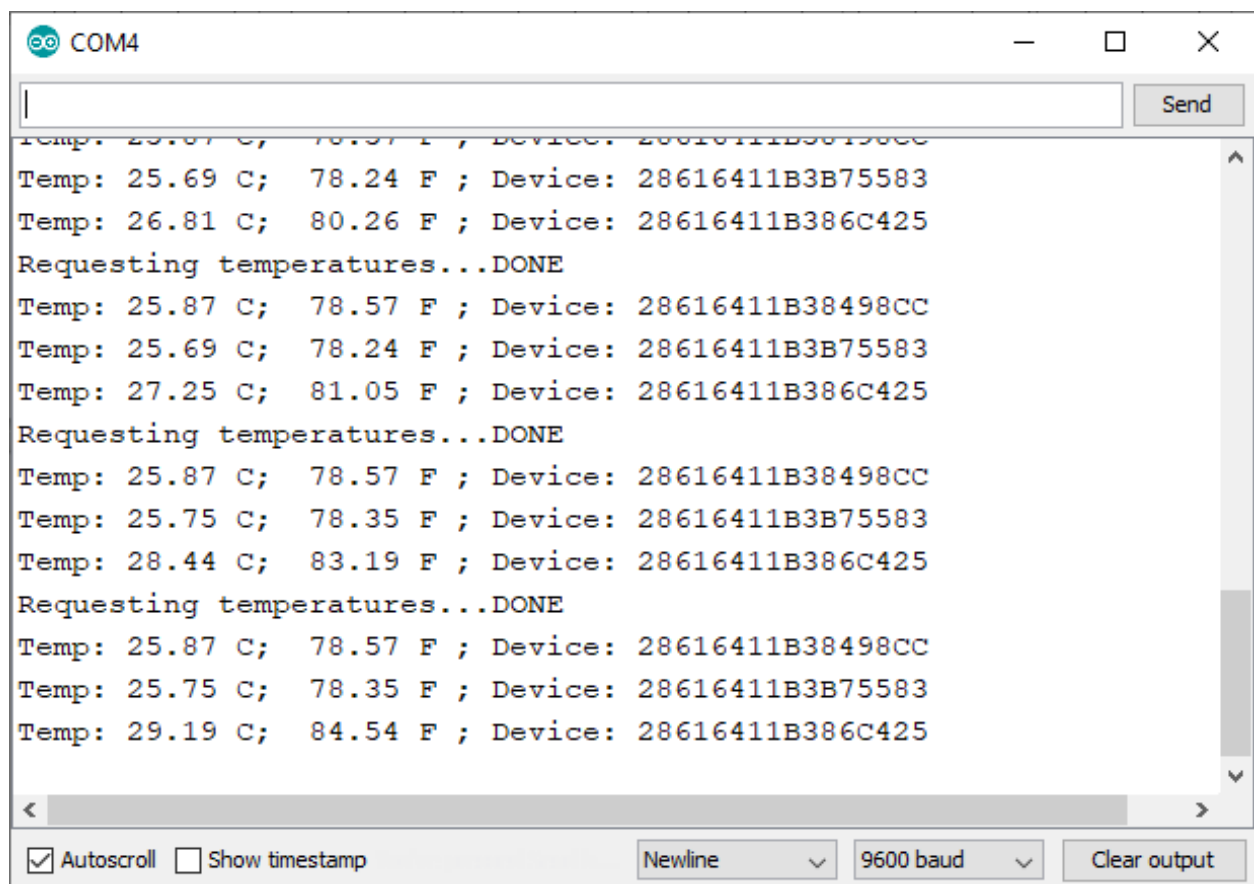
Az-Delivery

```
// one tab
    Serial.print("Device 0 Resolution: ");
    Serial.print(sensors.getResolution(one), DEC);
    Serial.println(); Serial.print("Device 1 Resolution: ");
    Serial.print(sensors.getResolution(two), DEC);
    Serial.println(); Serial.print("Device 2 Resolution: ");
    Serial.print(sensors.getResolution(three), DEC);
    Serial.println();
}
void printAddress(DeviceAddress deviceAddress) {
    for(uint8_t i = 0; i < 8; i++) {
        if(deviceAddress[i] < 16) Serial.print("0");
        Serial.print(deviceAddress[i], HEX);
    }
}
void printTemperature(DeviceAddress deviceAddress) {
    float tempC = sensors.getTempC(deviceAddress);
    Serial.print("Temp: "); Serial.print(tempC);
    Serial.print(" C; ");
    Serial.print(DallasTemperature::toFahrenheit(tempC));
    Serial.print(" F");
}
void printResolution(DeviceAddress deviceAddress) {
    Serial.print("Resolution: ");
    Serial.println(sensors.getResolution(deviceAddress));
}
void printData(DeviceAddress deviceAddress) {
    printTemperature(deviceAddress); Serial.print(" ; Device: ");
    printAddress(deviceAddress); Serial.println();
}
```

Az-Delivery

```
void loop() {  
  Serial.print("Requesting temperatures...");  
  sensors.requestTemperatures(); Serial.println("DONE");  
  printData(one);  
  printData(two);  
  printData(three);  
  delay(1000);  
}
```

Laden Sie den Sketch in den Uno hoch und öffnen Sie den Serial Monitor (*Tools > Serial Monitor*). Die Ausgabe sollte wie folgt aussehen:





Im Folgenden wird der Sketch erklärt.

Wir verwenden die Variable `ONE_WIRE_BUS`, um festzulegen, an welchem digitalen Pin wir die One-Wire-Schnittstelle anschließen werden. Für die Zwecke dieses eBooks wird der Wert der `ONE_WIRE_BUS`-Variablen auf `D2` gesetzt. Aber Sie können jeden anderen digitalen Pin des Uno verwenden, mit Ausnahme der in der seriellen Schnittstelle verwendeten Pins `D0` und `D1` (es ist eine Empfehlung, Sie können sie verwenden, aber Sie müssen sicherstellen, dass diese Pins beim Hochladen von Sketchen getrennt sind).

Wir haben die Variable `TEMPERATURE_PRECISION` verwendet, um die Präzision für DS18B20-Sonden einzustellen. Die in dieser Variablen gespeicherte Zahl ist eine digitale Konvertierungszahl in Bits und kann im Bereich von 9 bis 12 liegen. Jede andere Zahl führt zu einem Fehler. Für die Zwecke dieses eBooks setzen wir sie auf den maximalen Wert 12.

Wir haben die folgende Codezeile verwendet:

```
DeviceAddress one, two, three
```

um Variablen für Sondenadressen zu erstellen; in unserem Beispiel haben wir drei erstellt.

Wir haben ein *One-Wire*-Objekt definiert und erstellt, das für die One-Wire-Schnittstelle verwendet wird: `OneWire oneWire(ONE_WIRE_BUS);`

Dann verwenden wir ein *OneWire*-Objekt zur Definition und Erstellung eines Sensor-Objekts, das für alle angeschlossenen Sonden verwendet wird: `DallasTemperature sensors(&oneWire)`

Az-Delivery

Zur Initialisierung des *Sensor*-Objekts verwenden wir die Codezeile:

```
sensors.begin()
```

Mit dieser Zeile erkennt das *Sensoren*-Objekt alle an der One-Wire-Schnittstelle angeschlossenen Sonden. Es erkennt auch deren Adressen.

Jetzt können wir überprüfen, ob die Sonden einwandfrei funktionieren, indem wir die folgenden Codezeilen für jede Sonde verwenden, die wir an die One-Wire-Schnittstelle anschließen:

```
if(!sensors.getAddress(one, 0)) {  
    Serial.println("Unable to find address: Device 0"); }
```

wobei eins die Adresse der ersten Sonde ist.

Um die Präzision der Analog-Digital-Umwandlung der spezifischen Sonde einzustellen, haben wir die folgende Codezeile verwendet:

```
sensors.setResolution(one, TEMPERATURE_PRECISION)
```

Wenn Sie die Präzision der Analog-Digital-Umwandlung der spezifischen Sonde lesen möchten, können Sie die folgende Codezeile verwenden:

```
sensors.getResolution(one)
```

Die Funktion gibt einen hexadezimalen Wert zurück, und um ihn in einen dezimalen Wert umzuwandeln, verwenden wir die folgende Codezeile:

```
Serial.print(sensors.getResolution(one), DEC);
```

Um die Temperaturdaten auszulesen, müssen wir zuerst alle Daten der Sonden anfordern, indem wir die folgende Codezeile verwenden:

```
sensors.requestTemperatures();
```

Az-Delivery

Erst nach dieser Codezeile können wir Daten einer bestimmten Sonde lesen, indem wir die folgende Codezeile verwenden:

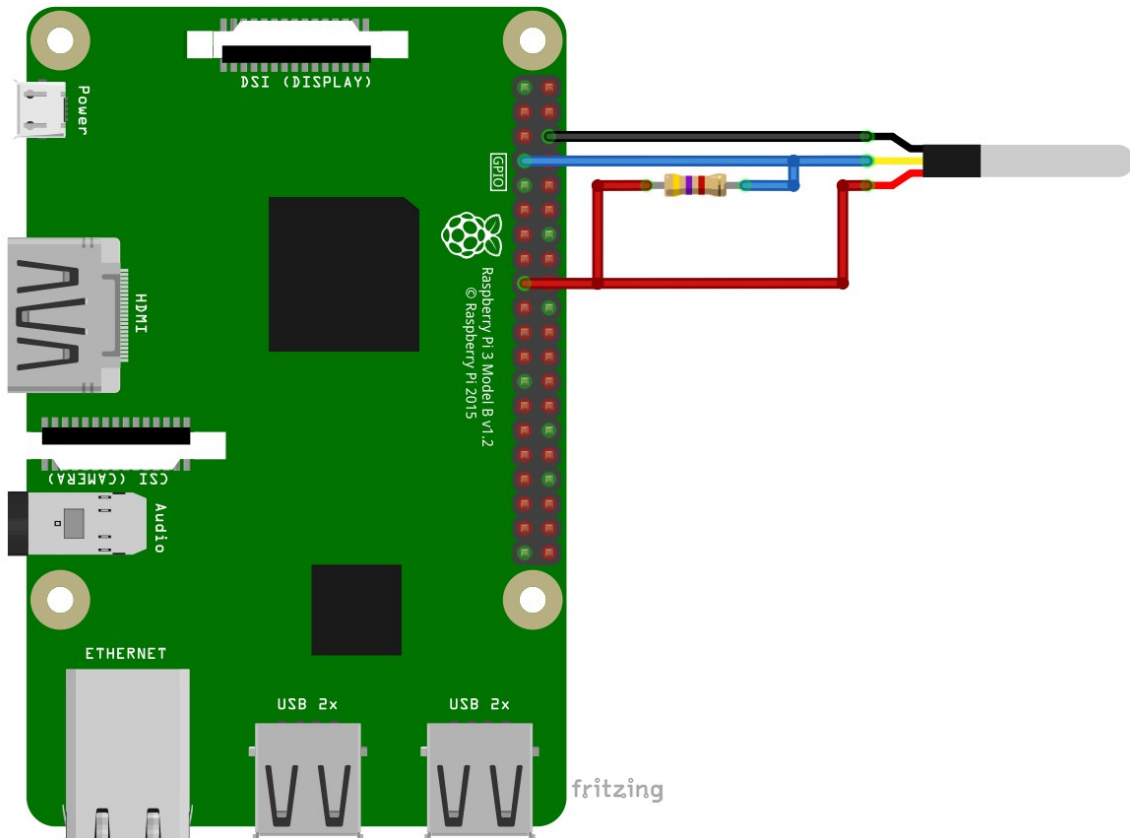
```
float tempC = sensors.getTempC(deviceAddress);
```

wobei wir das Argument *deviceAddress* in die Funktion integrieren, um Temperaturdaten von einer bestimmten Sonde zu lesen. Bei diesen Daten handelt es sich um Temperaturwerte in Celsius. Um sie in Fahrenheit umzuwandeln, haben wir die folgende Codezeile verwendet:

```
DallasTemperature::toFahrenheit(tempC)
```

Verbindung der Sonde mit dem Raspberry Pi

Verbinden Sie die DS18B20-Sonde mit dem Raspberry Pi, wie unten abgebildet:



DS18B20 pin	>	Raspberry Pi pin
GND	>	GND [pin 4]
DATA	>	GPIO4 [pin 7]
VCC	>	3V3 [pin 17]

Schwarzer Draht

Blauer Draht

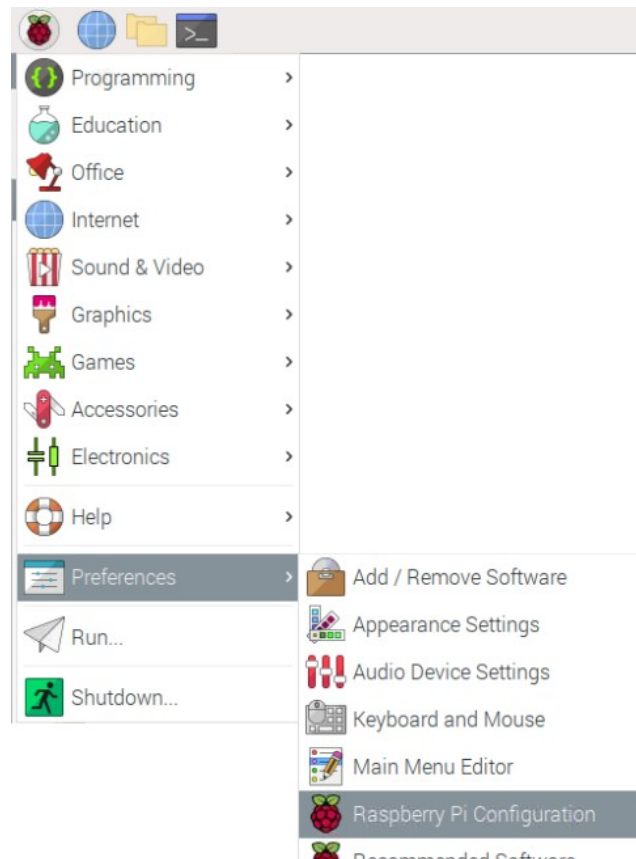
Roter Draht

Hinweis: Ein Pull-Up-Widerstand mit $4.7k\Omega$ ist zwischen dem *OUT-Pin* und dem 3V3-Pin angeschlossen.

Az-Delivery

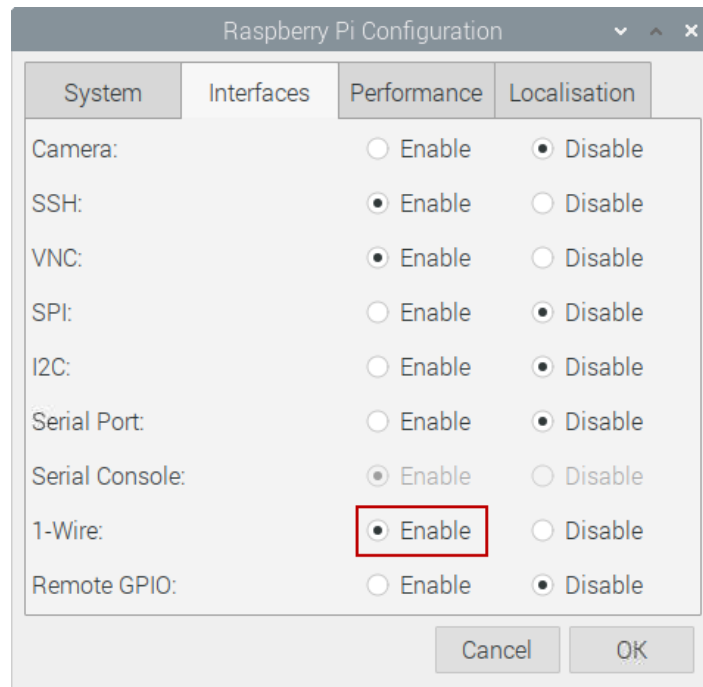
Aktivierung des One-Wire-Interface

Bevor wir die Sonde DS18B20 mit dem Raspberry Pi verwenden können, müssen wir zunächst die One-Wire-Schnittstelle in Raspbian aktivieren. Standardmäßig befindet sich die Hardware-One-Wire-Schnittstelle auf Pin GPIO4 (Pin 7), aber wir müssen sie erst aktivieren. Um die One-Wire-Schnittstelle zu aktivieren, öffnen Sie *All Apps* und gehen Sie zu: *Preferences > Raspberry Pi Configuration*, as shown below:

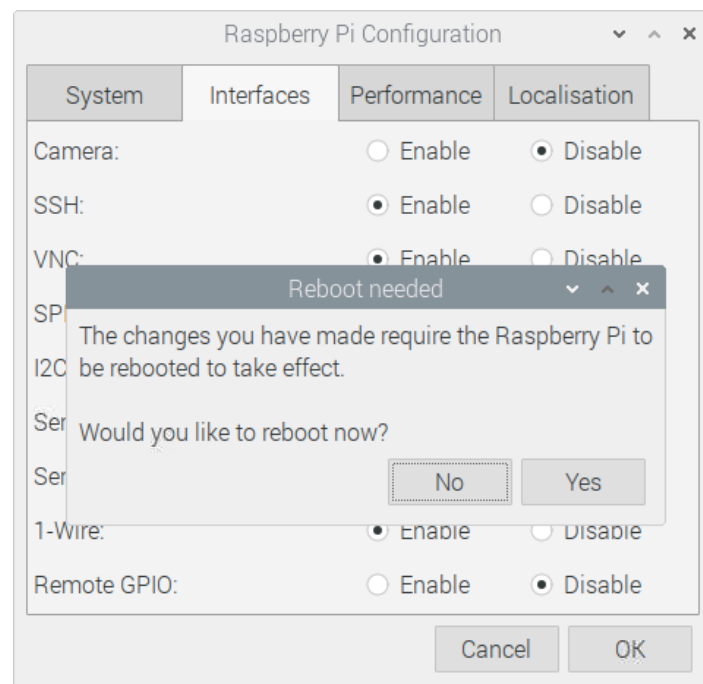


Az-Delivery

Wenn das neue Fenster erscheint, öffnen Sie die Registerkarte "Interfaces", suchen Sie nach Optionsfeldern mit der Bezeichnung "1-Wire" und wählen Sie das Optionsfeld "Enable", wie auf dem folgenden Bild dargestellt:



Sie werden aufgefordert, das System neu zu starten.



Az-Delivery

Wenn der Raspbian wieder hochgefahren ist, öffnen Sie das Terminal und führen Sie nacheinander folgenden Befehle aus:

```
sudo modprobe w1-gpio
sudo modprobe w1-therm
cd /sys/bus/w1/devices/
```

Und wenn Sie folgenden Befehl ausführen:

```
ls
```


Sollte die Ausgabe im Terminal, wie folgt, aussehen:

```
28-7285b3116461 and w1_bus_master1
```

die erste Nummer (28-7285b3116461) wird für Sie unterschiedlich sein, da dies die Serienadresse der jeweiligen Sonde ist, und jede Sonde hat ihre eigene eindeutige Serienadresse. Um zu testen, ob alles funktioniert, führen Sie diese beiden Befehle aus:

```
cd 28-7285b3116461 - eine Nummer/Serienadresse der letzten Seite
cat w1_slave
```

Die Ausgabe sollte wie folgt aussehen:



```
pi@raspberrypi:~ $ sudo modprobe w1-gpio
pi@raspberrypi:~ $ sudo modprobe w1-therm
pi@raspberrypi:~ $ cd /sys/bus/w1/devices/
pi@raspberrypi:/sys/bus/w1/devices $ ls
28-7285b3116461  w1_bus_master1
pi@raspberrypi:/sys/bus/w1/devices $ cd 28-7285b3116461
pi@raspberrypi:/sys/bus/w1/devices/28-7285b3116461 $ cat w1_slave
9e 01 ff ff 7f ff ff 89 : crc=89 YES
9e 01 ff ff 7f ff ff 89 t=25875
pi@raspberrypi:/sys/bus/w1/devices/28-7285b3116461 $
```

t=25875 - das sind die Temperaturdaten in °C (Celsius) = 25,875°C.



Aktivieren mehrerer One-Wire-Interfaces

Um die One-Wire-Schnittstelle ohne grafische Benutzeroberfläche (GUI) zu aktivieren, müssen Sie vor dem Neustart Ihres Raspberry Pi in der Datei `"/boot/config.txt"` die folgende Zeile hinzufügen:

```
dtoverlay=w1-gpio
```

oder

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

wobei *x* ein benutzerdefinierter Pin ist, falls Sie diesen verwenden möchten (Standard ist GPIO4 [Pin 7], wie wir im vorherigen Kapitel erwähnt haben).

Neuere Kernel (4.9.28 und später) erlauben Ihnen stattdessen das dynamische Laden von Overlays, einschließlich der Erstellung mehrerer One-Wire-Schnittstellen, die gleichzeitig verwendet werden können:

```
sudo dtoverlay w1-gpio gpiopin=4 pullup=0 # header pin 7  
sudo dtoverlay w1-gpio gpiopin=17 pullup=0 # header pin 11  
sudo dtoverlay w1-gpio gpiopin=27 pullup=0 # header pin 13
```

Sobald einer der obigen Schritte ausgeführt wurde und die Erkennung abgeschlossen ist, können Sie die Geräte auflisten, die der Raspberry Pi über die One-Wire-Schnittstellen entdeckt hat, indem Sie folgenden Befehl im Terminal ausführen:

```
ls /sys/bus/w1/devices/
```

Hinweis: Um **w1-gpio** auf dem Raspberry Pi zu verwenden, sollte ein Pull-Up-Widerstand mit $4.7k\Omega$ zwischen dem *GPIO-Pin* und der 3.3V-Stromversorgung angeschlossen werden.



Python-Skript zum Lesen mehrerer DS18B20-Sonden

Wir haben uns aufgrund der besseren Verständlichkeit dafür entschieden, den Code in zwei Skripte aufzuteilen. Es folgt ein Code für den Klassenindex:

```
import os
import glob
import time

class DS18B20:

    def __init__(self):
        os.system('modprobe w1-gpio')
        os.system('modprobe w1-therm')
        base_dir = '/sys/bus/w1/devices/'
        device_folder = glob.glob(base_dir + '28*')
        self._count_devices = len(device_folder)
        self._devices = list()
        i = 0
        while i < self._count_devices:
            self._devices.append(device_folder[i] + '/w1_slave')
            i += 1

    def device_names(self):
        names = list()
        for i in range(self._count_devices):
            names.append(self._devices[i])
            temp = names[i][20:35]
            names[i] = temp

        return names
```


Az-Delivery

```
# (one tab)
def _read_temp(self, index):
    f = open(self._devices[index], 'r')
    lines = f.readlines()
    f.close()
    return lines

def tempC(self, index = 0):
    lines = self._read_temp(index)
    retries = 5
    while (lines[0].strip()[-3:] != 'YES') and (retries > 0):
        time.sleep(0.1)
        lines = self._read_temp(index)
        retries -= 1

    if retries == 0:
        return 998

    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp = lines[1][equals_pos + 2:]
        return float(temp) / 1000
    else:
        return 999 # error

def device_count(self):
    return self._count_devices
```

(most of code in the script is modified from script on the adafruit site)

Speichern Sie das Skript unter dem Namen “DS18B20classfile.py”.

Az-Delivery

Im Anschluss folgt der Code für das Hauptskript:

```
import time
from DS18B20classFile import DS18B20

degree_sign = u'\xb0' # degree sign
devices = DS18B20()
count = devices.device_count()
names = devices.device_names()

print('[press ctrl+c to end the script]')
try: # Main program loop
    while True:
        i = 0
        print('\nReading temperature, number of sensors: {}'.format(count))

        while i < count:
            container = devices.tempC(i)
            print('{} Temp: {:.3f}{}C, {:.3f}{}F of the device {}'.format(i+1, container, degree_sign,
                container * 9.0 / 5.0 + 32.0, degree_sign,
                names[i]))

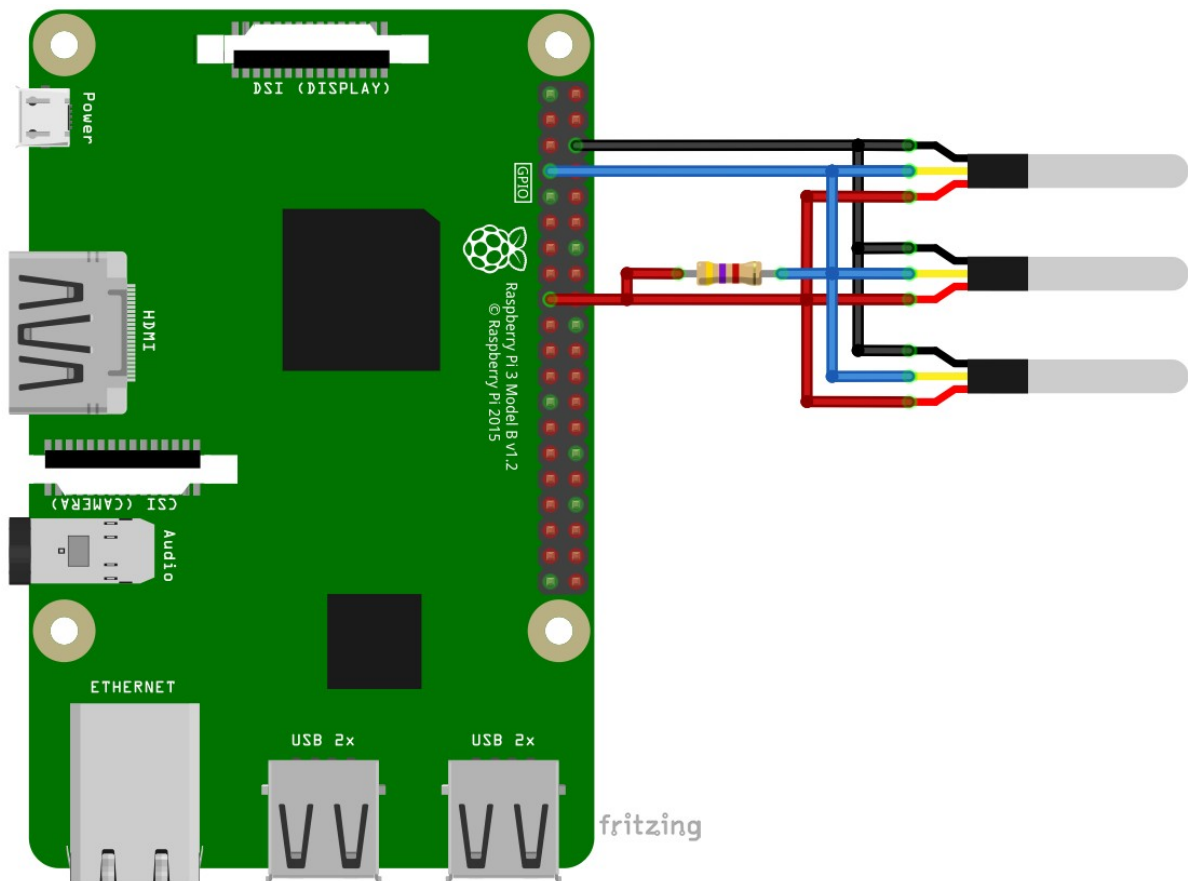
            i = i + 1

        time.sleep(1)

# Scavenging work after the end of the program
except KeyboardInterrupt:
    print('Script end!')
```

Speichern Sie das Skript unter dem Namen "*DS18B20multiple.py*" im selben Verzeichnis, in dem Sie das erste Skript gespeichert haben.

Wir haben zum Beispiel drei DS18B20-Sonden an dieselbe One-Wire-Schnittstelle des Raspberry Pi angeschlossen, wie unten abgebildet:

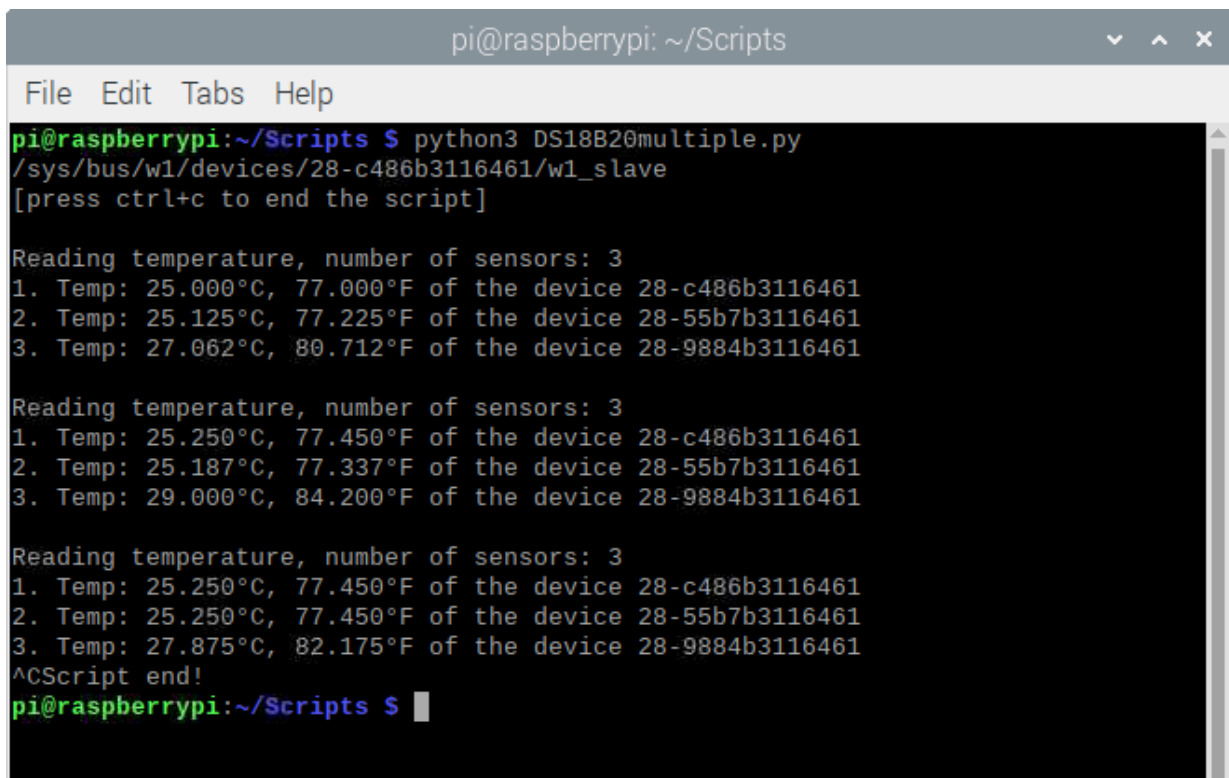


Az-Delivery

Um das Hauptskript auszuführen, öffnen Sie das Terminal in dem Verzeichnis, in dem Sie beide Skripte gespeichert haben, und führen Sie den folgenden Befehl aus:

python3 DS18B20multiple.py

Die Ausgabe sollte wie unten abgebildet aussehen:



```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ python3 DS18B20multiple.py
/sys/bus/w1/devices/28-c486b3116461/w1_slave
[press ctrl+c to end the script]

Reading temperature, number of sensors: 3
1. Temp: 25.000°C, 77.000°F of the device 28-c486b3116461
2. Temp: 25.125°C, 77.225°F of the device 28-55b7b3116461
3. Temp: 27.062°C, 80.712°F of the device 28-9884b3116461

Reading temperature, number of sensors: 3
1. Temp: 25.250°C, 77.450°F of the device 28-c486b3116461
2. Temp: 25.187°C, 77.337°F of the device 28-55b7b3116461
3. Temp: 29.000°C, 84.200°F of the device 28-9884b3116461

Reading temperature, number of sensors: 3
1. Temp: 25.250°C, 77.450°F of the device 28-c486b3116461
2. Temp: 25.250°C, 77.450°F of the device 28-55b7b3116461
3. Temp: 27.875°C, 82.175°F of the device 28-9884b3116461
^CScript end!
pi@raspberrypi:~/Scripts $
```

Um das Skript zu stoppen, drücken Sie "STRG + C" auf der Tastatur.

Sie können das Skript problemlos für eine oder mehrere DS18B20-Sonden verwenden.

**Sie haben es geschafft. Sie können jetzt unser Modul
nun für Ihre Projekte nutzen.**



Jetzt sind Sie dran! Entwickeln Sie Ihre eigenen Projekte und Smart-Home Installationen. Wie Sie das bewerkstelligen können, zeigen wir Ihnen unkompliziert und verständlich auf unserem Blog. Dort bieten wir Ihnen Beispielskripte und Tutorials mit interessanten kleinen Projekten an, um schnell in die Welt der Mikroelektronik einzusteigen. Zusätzlich bietet Ihnen auch das Internet unzählige Möglichkeiten, um sich in Sachen Mikroelektronik weiterzubilden.

Falls Sie nach noch weiteren hochwertigen Produkten für Arduino und Raspberry Pi suchen, sind Sie bei AZ-Delivery Vertriebs GmbH goldrichtig. Wir bieten Ihnen zahlreiche Anwendungsbeispiele, ausführliche Installationsanleitungen, E-Books, Bibliotheken und natürlich die Unterstützung unserer technischen Experten.

<https://az-delivery.de>

Viel Spaß!

Impressum

<https://az-delivery.de/pages/about-us>