

Part 1 - Sensoren auslesen

In diesem Tutorial wollen wir eine Temperaturüberwachung aufbauen die über mehrere Temperaturfühler die Temperaturen an verschiedenen Stellen in einem Raum überwacht. Dazu verwenden wir den DS18B20 der in einer vergossenen Variante erhältlich ist und sehr genau die Temperaturen messen kann. Zusätzlich lesen wir ebenfalls die Temperatur der CPU des Raspberry Pi aus und protokollieren die Daten in einer Log-Datei. In den weiteren Tutorials werden wir diese Daten dann per MQTT an einen Server zur Auswertung weiterleiten sowie eine Alarmbenachrichtigung über SMS und über einen Messenger Dienst einrichten.

Inhaltsverzeichnis

- 1 [Einleitung](#)
- 2 [Voraussetzungen](#)
- 3 [Verkablung](#)
- 4 [1-Wire aktivieren](#)
- 5 [DS18B20 auslesen](#)
- 6 [Temperatur der CPU auslesen](#)
- 7 [Temperaturen auslesen und in Log Datei schreiben](#)

Einleitung



Der Temperaturfühler DS18B20 ist ein sehr robuster und genauer Sensor der auf dem [1-Wire Bus](#) basiert. Dadurch können theoretisch bis zu 100 Sensoren, sogenannte Slaves, parallel an einen Master betrieben werden. Der Raspberry Pi unterstützt an GPIO4 (Pin 7) das 1-Wire Protokoll und fungiert in diesem Fall als Master. Der Begriff 1-Wire ist etwas verwirrend da der Temperaturfühler ja 3 Anschlüsse hat, der Begriff bezieht sich darauf das nur eine Datenleitung notwendig ist. Jeder Slave hat dabei eine eindeutige ID und ist individuell ansprechbar.

Voraussetzungen

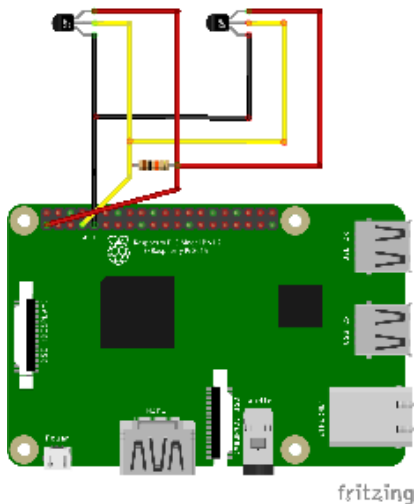
Für dieses Tutorial wird an Hardware folgendes benötigt:

- [Raspberry Pi](#) oder [BalenaFin](#)
- 1 oder mehrere [DS18B20 Temperaturfühler](#)
- 1 [10kOhm](#) oder [4,7kOhm](#) als [PullUp-Widerstand](#)

An Software verwenden wir das aktuelle Raspberry Pi OS (basierend auf Debian Buster). Das Betriebssystem sollte [grundlegend eingerichtet](#) sein und, da wir für die Auswertung der Sensoren Python3 verwenden, auf [Python3 als Standard](#) eingestellt sein. Auch sollten grundlegende Kenntnisse auf der Shell vorhanden sein sowie die Bedienung eines Editors.

Verkablung

Der elektronische Aufbau ist sehr einfach gehalten, die roten Anschlusskabel (Plus des Temperatursensors) werden alle parallel an 3,3V (Pin 1) des Raspberry Pi angeschlossen, die Masseleitungen (schwarz) werden mit GND (Pin 9) verbunden. Die Datenleitung der DS18B20 (meistens gelb, selten auch blau) wird an GPIO4 (Pin 7) angeschlossen. Zwischen 3,3V und GPIO4 (Pin 1 und Pin 7) wird dann noch der PullUp-Widerstand dazwischengeschaltet. Dieser Widerstand ist unabhängig von der Anzahl der angeschlossenen Sensoren und wird nur einmal benötigt.



Sie planen ein IoT, Automatisierungs- oder Embedded Projekt? Mit über 30 Jahren Erfahrung und zertifiziert nach DIN EN ISO 9001:2015 unterstützen wir Sie in jeder Phase Ihres Projektes. Ob Auswahl der geeigneten Hardware, Planung, Projektierung und Umsetzung, individuelle Anpassungen, Implementierung, Logistik oder Ausbau und Betreuung der Infrastruktur - sprechen Sie uns einfach an. Zu fairen Konditionen setzen wir gemeinsam mit Ihnen das Projekt erfolgreich um.

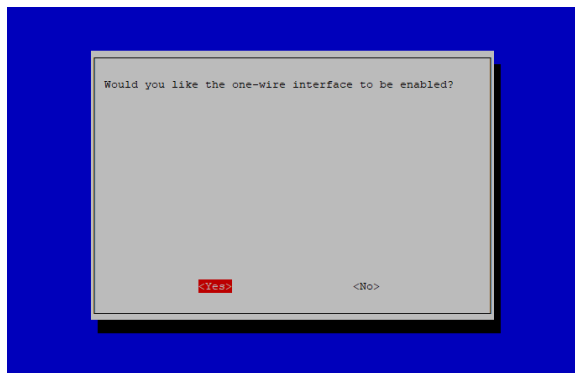
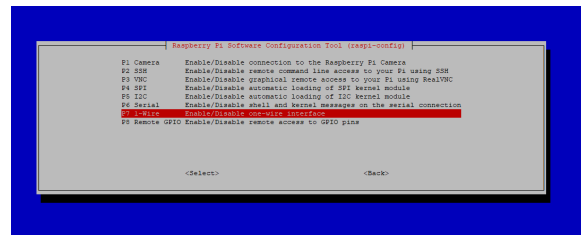
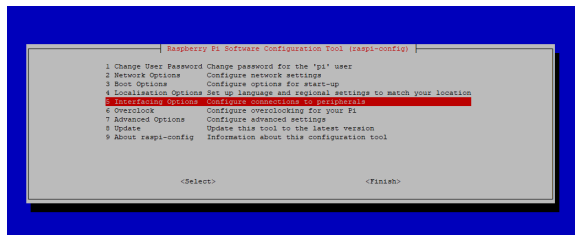
Kontakt: IoT@ico.de oder telefonisch unter 06432 / 9139 - 320

1-Wire aktivieren


Nachdem die Verkabelung abgeschlossen ist können wir nun den Raspberry Pi starten und im nächsten Schritt 1-Wire aktivieren. In der Standardeinstellung ist der 1-Wire Bus in Raspberry Pi OS nicht aktiv, lässt sich aber leicht über raspi-config aktivieren.

```
pi@raspberrypi:~ $ sudo raspi-config
```

Die Einstellungen für 1-Wire befinden sich im Menüpunkt 5 Interfacing Options und im anschließenden Untermenü P7 1-Wire. Dort muss 1-Wire dann nur aktiviert werden und wie von raspi-config empfohlen ein Neustart durchgeführt werden. Nach dem Neustart ist 1-Wire aktiviert und kann direkt benutzt werden.



Durch das Aktivieren von 1-Wire über raspi-config wird in der Datei /boot/config.txt folgender Eintrag hinzugefügt:

 dtoverlay=w1-gpio

Bei älteren Versionen von Raspberry Pi OS, noch als Raspbian bekannt, mussten noch weitere Module wie w1-therm oder auch der PullUp Modus definiert werden. Dies ist mit der aktuellen Version von Raspberry Pi OS nicht mehr notwendig.

DS18B20 auslesen

Der 1-Wire Bus Master wie auch die Slaves integrieren sich nach dem Neustart des Systems direkt in das Dateisystem. Unter /sys/bus/w1/devices/ sind dann für den Master wie auch für jeden Slave ein eigener Ordner vorhanden. Der Ordnernamen der Slaves entspricht dann der ID des jeweiligen Sensors. In diesem Beispiel sind 2 Stück DS18B20 angeschlossen.

```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ ls -l /sys/bus/w1/devices/  
total 0  
lrwxrwxrwx 1 root root 0 Aug 15 18:40 28-011927724003 -> ../../../../devices/w1_bus_master1/28-011927724003  
lrwxrwxrwx 1 root root 0 Aug 15 18:40 28-0119292513c2 -> ../../../../devices/w1_bus_master1/28-0119292513c2  
lrwxrwxrwx 1 root root 0 Aug 15 18:38 w1_bus_master1 -> ../../../../devices/w1_bus_master1  
pi@raspberrypi:~ $
```

Im Verzeichnis des Master w1_bus_master1 finden sich Informationen zum Master selbst aber auch zu den angeschlossenen Slaves. Diese befinden sich alle in separaten Dateien die einfach mit "cat" ausgelesen werden können. Root Rechte sind zum Auslesen der Dateien nicht notwendig.

Datei	Funktion
w1_master_slave_count	Enthält die Anzahl der angeschlossenen Slaves
w1_master_slaves	Enthält die Namen (IDs) der angeschlossenen Slaves
w1_master_pullup	Definiert den PullUP Widerstand
w1_master_max_slave_count	Enthält die Anzahl der Slaves die maximal angeschlossen werden können

```
pi@raspberrypi: /sys/bus/w1/devices/w1_bus_master1  
pi@raspberrypi:~ $ cd /sys/bus/w1/devices/w1_bus_master1  
pi@raspberrypi:/sys/bus/w1/devices/w1_bus_master1 $ ls -l  
total 0  
drwxr-xr-x 4 root root 0 Aug 15 18:38 28-011927724003  
drwxr-xr-x 4 root root 0 Aug 15 18:38 28-0119292513c2  
lrwxrwxrwx 1 root root 0 Aug 15 19:00 driver -> ../../bus/w1/drivers/w1_master_driver  
drwxr-xr-x 2 root root 0 Aug 15 19:00 power  
lrwxrwxrwx 1 root root 0 Aug 15 18:38 subsystem -> ../../bus/w1  
-rw-r--r-- 1 root root 4096 Aug 15 19:00 therm_bulk_read  
-rw-r--r-- 1 root root 4096 Aug 15 18:38 uevent  
-rw-rw-r-- 1 root root 4096 Aug 15 19:00 w1_master_add  
-r--r--r-- 1 root root 4096 Aug 15 19:00 w1_master_attempts  
-rw-rw-r-- 1 root root 4096 Aug 15 19:00 w1_master_max_slave_count  
-r--r--r-- 1 root root 4096 Aug 15 19:00 w1_master_name  
-r--r--r-- 1 root root 4096 Aug 15 19:00 w1_master_pointer  
-rw-rw-r-- 1 root root 4096 Aug 15 19:00 w1_master_pullup  
-rw-rw-r-- 1 root root 4096 Aug 15 19:00 w1_master_remove  
-rw-rw-r-- 1 root root 4096 Aug 15 19:00 w1_master_search  
-r--r--r-- 1 root root 4096 Aug 15 19:00 w1_master_slave_count  
-r--r--r-- 1 root root 4096 Aug 15 19:00 w1_master_slaves  
-r--r--r-- 1 root root 4096 Aug 15 19:00 w1_master_timeout  
-r--r--r-- 1 root root 4096 Aug 15 19:00 w1_master_timeout_us  
pi@raspberrypi:/sys/bus/w1/devices/w1_bus_master1 $ cat w1_master_slave_count  
2  
pi@raspberrypi:/sys/bus/w1/devices/w1_bus_master1 $ cat w1_master_slaves  
28-0119292513c2  
28-011927724003  
pi@raspberrypi:/sys/bus/w1/devices/w1_bus_master1 $ cat w1_master_pullup  
1  
pi@raspberrypi:/sys/bus/w1/devices/w1_bus_master1 $ cat w1_master_max_slave_count  
64  
pi@raspberrypi:/sys/bus/w1/devices/w1_bus_master1 $
```

In den Verzeichnissen der Slaves sind insbesondere die Dateien w1_slave und temperature interessant.

```

pi@raspberrypi:~ $ ls -l /sys/bus/wl/devices/28-01192*/
/sys/bus/wl/devices/28-011927724003/:
total 0
-rw-r--r-- 1 root root 4096 Aug 15 18:42 alarms
lrwxrwxrwx 1 root root 0 Aug 15 18:42 driver -> ../../../../bus/wl/drivers/wl_slave_driver
--w----- 1 root root 4096 Aug 15 18:42 eeprom
-r--r--r-- 1 root root 4096 Aug 15 18:42 ext_power
drwxr-xr-x 3 root root 0 Aug 15 18:38 hwmon
-r--r--r-- 1 root root 4096 Aug 15 18:42 id
-r--r--r-- 1 root root 4096 Aug 15 18:42 name
drwxr-xr-x 2 root root 0 Aug 15 18:42 power
-rw-r--r-- 1 root root 4096 Aug 15 18:42 resolution
lrwxrwxrwx 1 root root 0 Aug 15 18:38 subsystem -> ../../../../bus/wl
-r--r--r-- 1 root root 4096 Aug 15 18:42 temperature
-rw-r--r-- 1 root root 4096 Aug 15 18:38 uevent
-rw-r--r-- 1 root root 4096 Aug 15 18:42 wl_slave

/sys/bus/wl/devices/28-0119292513c2/:
total 0
-rw-r--r-- 1 root root 4096 Aug 15 18:45 alarms
lrwxrwxrwx 1 root root 0 Aug 15 18:45 driver -> ../../../../bus/wl/drivers/wl_slave_driver
--w----- 1 root root 4096 Aug 15 18:45 eeprom
-r--r--r-- 1 root root 4096 Aug 15 18:45 ext_power
drwxr-xr-x 3 root root 0 Aug 15 18:38 hwmon
-r--r--r-- 1 root root 4096 Aug 15 18:45 id
-r--r--r-- 1 root root 4096 Aug 15 18:45 name
drwxr-xr-x 2 root root 0 Aug 15 18:45 power
-rw-r--r-- 1 root root 4096 Aug 15 18:45 resolution
lrwxrwxrwx 1 root root 0 Aug 15 18:38 subsystem -> ../../../../bus/wl
-r--r--r-- 1 root root 4096 Aug 15 18:45 temperature
-rw-r--r-- 1 root root 4096 Aug 15 18:38 uevent
-rw-r--r-- 1 root root 4096 Aug 15 18:42 wl_slave
pi@raspberrypi:~ $

```

wl_slave enthält zum einen Status und die Checksumme des Devices und zum anderen die aktuelle Temperatur in der zweiten Zeile. In der ersten Zeile sollte ein YES stehen, bei Error oder No muss der Sensor und die Verkabelung nochmals überprüft werden. Die aktuelle Temperatur ist dann mit t= angegeben und entspricht Milligrad Celsius. In diesem Beispiel sind es aktuell bei beiden Sensoren 25,625°C

```

pi@raspberrypi:~ $ cat /sys/bus/wl/devices/28-011927724003/wl_slave
99 01 4b 46 7f ff 0c 10 5a : crc=5a YES
99 01 4b 46 7f ff 0c 10 5a t=25562
pi@raspberrypi:~ $ cat /sys/bus/wl/devices/28-0119292513c2/wl_slave
9a 01 4b 46 7f ff 0c 10 9f : crc=9f YES
9a 01 4b 46 7f ff 0c 10 9f t=25625
pi@raspberrypi:~ $ cat /sys/bus/wl/devices/28-01192*/wl_slave
9a 01 4b 46 7f ff 0c 10 9f : crc=9f YES
9a 01 4b 46 7f ff 0c 10 9f t=25625
9a 01 4b 46 7f ff 0c 10 9f : crc=9f YES
9a 01 4b 46 7f ff 0c 10 9f t=25625
pi@raspberrypi:~ $

```

Zusätzlich ist die Temperatur aber auch in der Datei temperature als reiner Zahlenwert in Milligrad enthalten:

```

pi@raspberrypi:~ $ cat /sys/bus/wl/devices/28-011927724003/temperature
25625

```

Temperatur der CPU auslesen

Um die Temperatur der CPU des Raspberry Pi auszulesen kann der Befehl `vcgencmd` mit dem Parameter `measure_temp` verwendet werden. Dafür sind ebenfalls keine Root Rechte erforderlich, die Temperatur wird dabei direkt in Grad Celsius angegeben.

```
pi@raspberrypi:~ $ vcgencmd measure_temp  
temp=42.4'C
```

Temperaturen auslesen und in Log Datei schreiben

Da wir nun alle Temperaturen auslesen können verwenden wir jetzt ein kleines Python Script welches die Temperaturen automatisch ausliest und in eine Log Datei schreibt.

temperature.py

```
#!/usr/bin/python  
# -*- coding: UTF-8 -*-  
  
import time  
import os  
import logging  
from logging.handlers import TimedRotatingFileHandler  
  
logger = logging.getLogger('temperature')  
logger.setLevel(logging.INFO)  
  
formatter = logging.Formatter('%(asctime)s - %(message)s',  
                              datefmt='%d-%m-%Y %H:%M:%S')  
  
loghandler = TimedRotatingFileHandler(  
    '/home/pi/temperature.log', when='D', interval=1, backupCount=10)  
loghandler.setLevel(logging.INFO)  
loghandler.setFormatter(formatter)  
  
logger.addHandler(loghandler)  
  
logger.info('Programm started...')  
  
sensors = {}  
sensors['Serverraum Oben'] = '28-011927724003'  
sensors['Serverraum Unten'] = '28-0119292513c2'  
  
interval = 5  
  
def main():  
    while True:  
        for name, sensor in sensors.items():  
            file = open('/sys/bus/wl/devices/' + str(sensor) + '/temperature')  
            filecontent = file.read()  
            file.close()  
  
            temp = float(filecontent) / 1000  
            logger.info("%s: %.2f °C", name, temp)  
  
            cpu_temp = os.popen("vcgencmd measure_temp").readline()  
            logger.info("CPU: %s °C", cpu_temp[5][:4])  
            time.sleep(interval)  
  
if __name__ == '__main__':  
    main()
```

Nachfolgend die Erläuterungen zum Script

Zeile	Funktion
1-2	Definition des Python Scripts und des Encoding
4-7	Importieren der notwendigen Bibliotheken

9-10	Erzeugen eines Logger Objekts mit dem Level INFO
12	Format des Log Eintrags: <Tag>-<Monat>-<Jahr> <Stunde>:<Minute>:<Sekunde> - <Text des Eintrags>
15	Name der Log Datei temperature.log, soll täglich neu erzeugt werden, alte Log Dateien erhalten die Endung des Tags. Maximal 10 alte Log Dateien werden behalten, ältere werden gelöscht
17-22	Log Level auf INFO gesetzt und dem Logger Objekt hinzugefügt
24-26	Erzeugen eines Dictionarys mit dem Namen des Sensors und der entsprechenden ID
28	Variable für den Intervall der Messung, hier alle 5 Sekunden soll eine Messung erfolgen
30-31	Starten der Hauptfunktion
32	Starten des Loop über die Sensoren
33	Öffnen der temperature Datei des jeweiligen Sensors
34-35	Einlesen des Sensorwertes
37	Berechnung der Temperatur von Milligrad in Grad Celsius
38	Log Eintrag mit dem Namen des Sensors und der berechneten Temperatur mit zwei Nachkommastellen
40	Nachdem die Sensoren ausgewertet sind wird nun die Temperatur der CPU eingelesen
41	Formatierung der Temperatur und Eintrag in das Logfile
42	Pause des Programms bis zur nächsten Messung
44-45	Starten der Hauptfunktion

Nach dem Starten des Scripts mit "python temperature.py" wird nun alle 5 Sekunden eine Messung gestartet und in die Logdatei geschrieben:

```
pi@raspberrypi:~ $ tail -20 temperature.log
23-08-2020 17:57:58 - Serverraum Unten: 23.44 °C
23-08-2020 17:57:58 - CPU: 44.0 °C
23-08-2020 17:58:04 - Serverraum Oben: 23.50 °C
23-08-2020 17:58:04 - Serverraum Unten: 23.44 °C
23-08-2020 17:58:04 - CPU: 44.5 °C
23-08-2020 17:58:10 - Serverraum Oben: 23.44 °C
23-08-2020 17:58:11 - Serverraum Unten: 23.44 °C
23-08-2020 17:58:11 - CPU: 44.0 °C
23-08-2020 17:58:17 - Serverraum Oben: 23.44 °C
23-08-2020 17:58:18 - Serverraum Unten: 23.44 °C
23-08-2020 17:58:18 - CPU: 44.0 °C
23-08-2020 17:58:24 - Serverraum Oben: 23.44 °C
23-08-2020 17:58:25 - Serverraum Unten: 23.44 °C
23-08-2020 17:58:25 - CPU: 44.0 °C
23-08-2020 17:58:31 - Serverraum Oben: 23.44 °C
23-08-2020 17:58:32 - Serverraum Unten: 23.44 °C
23-08-2020 17:58:32 - CPU: 44.5 °C
23-08-2020 17:58:38 - Serverraum Oben: 23.44 °C
23-08-2020 17:58:39 - Serverraum Unten: 23.44 °C
23-08-2020 17:58:39 - CPU: 44.5 °C
```

Damit das Script bei jedem Systemstart ebenfalls gestartet wird bietet sich es an folgende Befehlszeile in die Datei /etc/rc.local vor dem exit 0 einzutragen.

```
...
python /home/pi/temperature.py &

exit 0
```