



Hochschule Neubrandenburg  
University of Applied Sciences

Hochschule Neubrandenburg  
Master – Geoinformatik und Geodäsie  
Modul VMGG33 – Informatik-Projekt  
1. Semester

# Entwicklerdokumentation

## Entwicklung einer konfigurierbarer Sensorstation auf der Basis von Raspberry Pi

Autor: Tino Schuldt

Betreuer: Dipl.-Inform. Jörg Schäfer  
Prof. Dr.-Ing. Andreas Wehrenpfennig

Tag der Einreichung: 18.01.2016

# Inhaltsverzeichnis

<b>Inhaltsverzeichnis.....</b>	<b>II</b>
<b>1 Einführung.....</b>	<b>1</b>
1.1 Aufgabenstellung.....	1
1.2 Ziel .....	1
1.3 Anwendung .....	1
<b>2 Raspberry Pi.....</b>	<b>2</b>
2.1 Modelle.....	2
2.2 Aufbau.....	2
2.3 Betriebssystem.....	3
<b>3 GPIO-Schnittstelle.....</b>	<b>4</b>
3.1 GPIO-Pins .....	4
3.2 GPIO-Belegung .....	4
3.3 WiringPi.....	6
3.4 GPIO-Sonderfunktionen .....	6
<b>4 Erste Versuche.....</b>	<b>7</b>
4.1 Zugriff mit einem SSH-Client .....	7
4.2 Übertragen von Projektdateien .....	7
4.3 Ansteuern einer LED .....	7
<b>5 Ansteuern von Sensoren.....</b>	<b>9</b>
5.1 Ansteuern des DHT22-Sensor.....	9
5.2 Ansteuern des DS18B20-Sensor.....	10
5.3 Ansteuern des HC-SR04-Sensor.....	12
5.4 Ansteuern des MPU-6050-Sensor .....	13
5.5 Fazit .....	13
<b>6 Konfigurierbarkeit.....</b>	<b>14</b>
6.1 Ordnerstruktur .....	14
6.2 Konfiguration .....	17
6.3 Erster Start .....	17
6.4 Installation der Sensoren .....	17
6.5 Sensorstation nutzen.....	17
6.6 Automatisierte Aufzeichnung .....	19
<b>7 Speicherung .....</b>	<b>21</b>
7.1 SQLite-Datenbank.....	21
7.2 CSV-Daten .....	21
<b>8 Webbasierte Datenabfrage .....</b>	<b>22</b>

8.1	RESTful-Webservice .....	22
<b>9</b>	<b>Zusammenfassung und Ausblick .....</b>	<b>24</b>
9.1	Zusammenfassung .....	24
9.2	Ausblick und weiterführende Arbeiten .....	24
<b>Glossar</b> .....		<b>IV</b>
<b>Quellenverzeichnis</b> .....		<b>V</b>
<b>Abbildungsverzeichnis</b> .....		<b>VI</b>
<b>Tabellenverzeichnis</b> .....		<b>VI</b>
<b>Anhang</b> .....		<b>VII</b>
	Anhang A – Projektdateien .....	VII

# 1 Einführung

## 1.1 Aufgabenstellung

In diesem Projekt soll eine Sensorstation mithilfe des Raspberry Pi entwickelt werden. Diese Sensorstation soll für den Benutzer frei konfigurierbar sein und beliebige Sensoren unterstützen. Außerdem soll ein Konzept für die Konfigurierbarkeit und die Datenhaltung erstellt werden. Ein weiterer Punkt in diesem Projekt betrifft die Bereitstellung der Daten mithilfe einer webbasierten Schnittstelle zur Datenabfrage.

## 1.2 Ziel

Am Ende des Projektes soll ein Konzept für die Sensorstation und ein lauffähiger Prototyp erstellt werden. Hierfür wird sich zunächst auf ein Sensor je Typ beschränkt.

## 1.3 Anwendung

In der Abb. 1.1 wird das UseCase-Diagramm der Sensorstation dargestellt. Der Benutzer soll folgende Aktionen mit der Sensorstation ausführen können. Dazu gehört das Installieren der Sensoren, das Konfigurieren der Sensoren, die Aufzeichnung, Visualisierung und Bereitstellung der Daten.

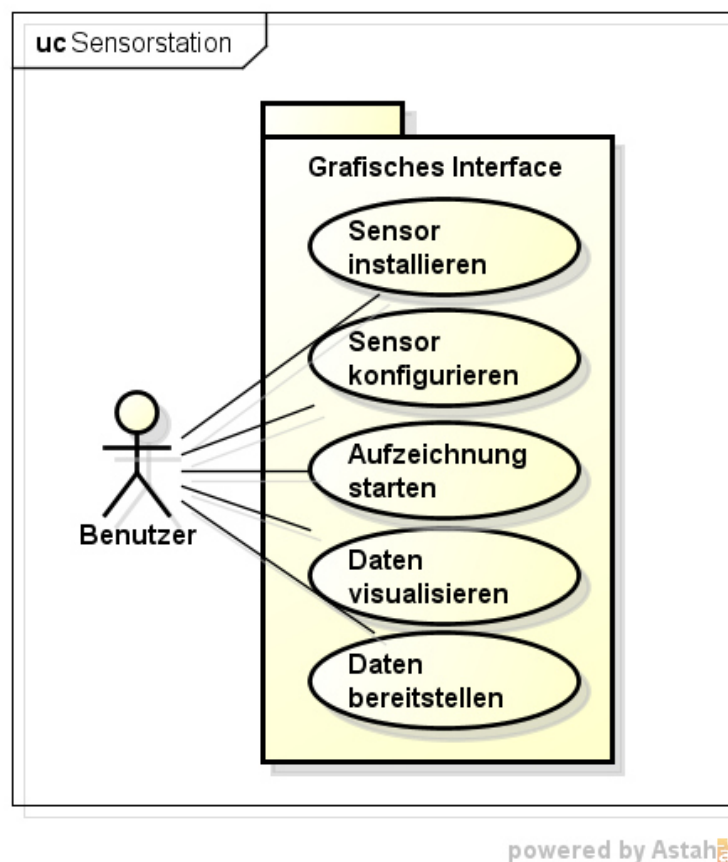


Abb. 1.1: UseCase-Diagramm zur Verwendung der Sensorstation

## 2 Raspberry Pi

### 2.1 Modelle

Beim Raspberry Pi gibt es verschiedene Modelle auf dem Markt. Diese Modelle und die Unterschiede sind in der Tabelle 2.1 dargestellt. Der Unterschied zwischen Modell A und B ist die Erhöhung des Arbeitsspeichers (RAM), zusätzlich ist beim Modell B eine Ethernet-Schnittstelle (RJ45) hinzugekommen und mehrere USB-Schnittstellen sind auf dem Board verfügbar. Das Plus hinter dem Modell A und B signalisiert, dass statt 26 GPIO-Pins hier 40 GPIO-Pins auf dem Board zur Verfügung stehen. Der Raspberry Pi 2 besteht aus einem Vierkernprozessor mit jeweils 900 MHz und der Arbeitsspeicher wurde auf 1 GByte erhöht.

Modell	RPi A	RPi A+	RPi B	RPi B+	RPi 2 B
System on Chip	Broadcom BCM 2835 (ARMv6)				BCM2836 (ARMv7)
	700 MHz				4x 900 MHz
RAM	256 MByte (400 MHz)	256 MByte (400 MHz)	512 MByte (400 MHz)	512 MByte (400 MHz)	1 GByte
Speicher	SD-Card	MicroSD-Card	SD-Card	MicroSD-Card	MicroSD-Card
USB 2.0	1	1	2	4	4
HDMI	1	1	1	1	1
Video-Ausgang	1	1	1	1	1
Fast Ethernet	-	-	1	1	1
Audio-Ausgang	1	1	1	1	1
GPIO-Pins	26 (17)	40 (26)	26 (17)	40 (26)	40 (26)
Preis	ca. 25 Euro	ca. 25 Euro	ca. 25 Euro	ca. 30 Euro	ca. 35 Euro

Tabelle 2.1: Vergleich der Modelle (vgl. [Ele151])

### 2.2 Aufbau

Für das Projekt wurde der Raspberry Pi B+ ausgewählt. Grundsätzlich ist aber jedes Modell für das Projekt geeignet. In der Abb. 2.1 sind die einzelnen Komponenten des Raspberry Pi Modell B+ dargestellt.

Wichtig für das Projekt sind die GPIO-Pins zum Ansteuern der Sensoren. Auf der Rückseite befindet sich der MicroSD-Card-Slot, in dem die MicroSD-Card mit dem darauf befindlichen Betriebssystem eingeschoben werden kann. Diese MicroSD-Card sollte eine Class 10 (10 MB/s) sein, um möglichst schnelle Schreibgeschwindigkeiten zu erreichen und mind. 8 GB groß sein, damit das Betriebssystem darauf installiert werden kann. Für die

Energieversorgung wird ein Micro-USB-Kabel mit Netzteil, welches mind. 1A liefert, verwendet. Um das Raspberry Pi mit dem Netzwerk zu verbinden, kann die Ethernet-Schnittstelle mit einem RJ45-Kabel oder mit einem WLAN-Stick in einem der freien USB-Ports verwendet werden. Die anderen freien USB-Ports können noch für weitere Eingabegeräte wie z. B. Tastatur verwendet werden. Die HDMI-Schnittstelle dient als digitale grafische Ausgabe an einem Bildschirm (vgl. [Tut151]).

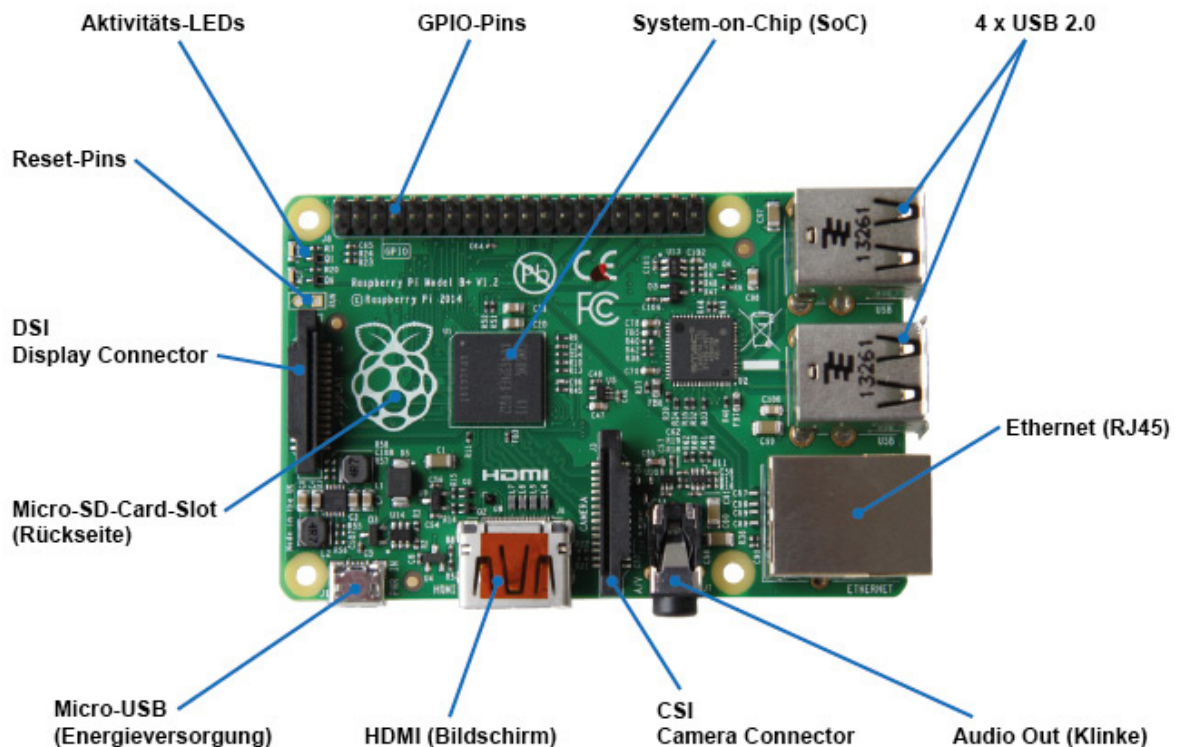


Abb. 2.1: Komponenten vom Raspberry Pi B+ (übernommen von [Ele152])

## 2.3 Betriebssystem

Für das Raspberry Pi gibt es eine Vielzahl an Betriebssystemen. Dazu gehören Raspbian, Pidora, Archlinux, OpenElec, Raspbmc, RISC OS, Kali Linux, Razdroid, RetroPie. Speziell für das Raspberry Pi 2 ist auch eine Windows 10 IoT Version von Microsoft zur Verfügung gestellt worden. (vgl. [Ras151]).

In diesem Projekt wird das Betriebssystem Raspbian OS 4.1.7+ Jessie eingesetzt.<sup>1</sup>

<sup>1</sup> Installationsanleitung um Raspbian OS auf die MicroSD-Card zu installieren: <http://basic-tutorials.de/raspbian-auf-raspberry-pi-installieren>

## 3 GPIO-Schnittstelle

### 3.1 GPIO-Pins

Die GPIO-Pins (General Purpose Input/Output) dienen als Schnittstelle zwischen externen Geräten und dem Raspberry Pi. Diese Pins können als Ein- und Ausgabe genutzt werden, um Systeme oder Schaltungen zu steuern (vgl. [Ras152], [Ele153]).

In der Abb. 3.1 wird die Anordnung der Pins von 1 bis 40 dargestellt.

### 3.2 GPIO-Belegung

In der Tabelle 3.1 wird die Pinbelegung des Raspberry Pi dargestellt. Es wird unterschieden zwischen Revision 1 (vor September 2012) und Revision 2, bei dem die Pinbelegung sich minimal verändert hat. Einige Modelle des Raspberry Pi stellen 26 Pins und andere 40 Pins zur Verfügung. Davon stehen jeweils 16 oder 26 Pins als GPIO zur Verfügung. Die Pins 1 und 17 bieten eine +3,3 V und die Pins 2 und 4 eine +5 V Spannungsversorgung. Die Pins 6, 9, 14, 20 und 25 sowie 30, 34, 39 sind für die Erdung (Ground, 0 V) zuständig. Alle anderen Pins können als GPIO genutzt werden (vgl. [Ras151], [Ras152], [Net151], [Ele154]).

Die Pins 27 (SDA 0 oder auch ID\_SD) und 28 (SCL 0 oder auch ID\_SC) erlauben den Anschluss eines ID EEPROM (Electrically Erasable Programmable Read-Only Memory). Dadurch können Erweiterungsplatinen während der Bootphase identifiziert werden. Dies ist für Hersteller von Platinen gedacht. Daher kann an diesen beiden Pins nichts anderes als ein ID EEPROM angeschlossen werden (vgl. [Ras153]).

Mit folgendem Befehl kann die Konfiguration und Beschaltung der GPIO-Pins herausgefunden werden:

- `gpio readall`

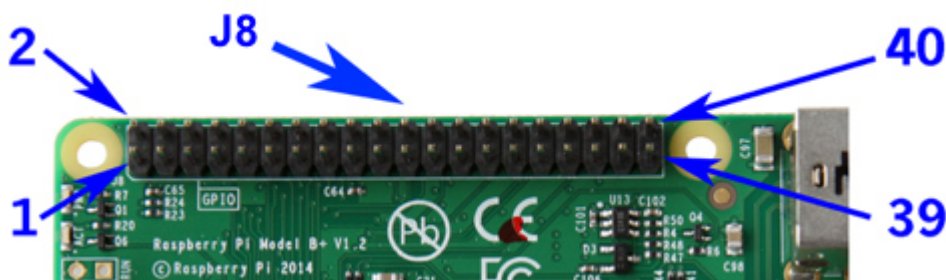


Abb. 3.1: Anordnung der Pins (übernommen von [PiJ15])

WiringPi	Rev 2	Rev 1	Pins		Rev 1	Rev 2	WiringPi
	+ 3,3 V	+ 3,3 V	1		2	+ 5 V	
8	GPIO 2	GPIO 0	3		4	+ 5 V	
9	GPIO 3	GPIO 1	5		6	GND	
7	GPIO 4	GPIO 4	7		8	GPIO 14	15
	GND	GND	9		10	GPIO 15	16
0	GPIO 17	GPIO 17	11		12	GPIO 18	1
2	GPIO 27	GPIO 21	13		14	GND	
3	GPIO 22	GPIO 22	15		16	GPIO 23	4
	+ 3,3 V	+ 3,3 V	17		18	GPIO 24	5
12	GPIO 10	GPIO 10	19		20	GND	
13	GPIO 9	GPIO 9	21		22	GPIO 25	6
14	GPIO 11	GPIO 11	23		24	GPIO 8	10
	GND	GND	25		26	GPIO 7	11
30	SDA 0 (GPIO 0)		27		28	SCL 0 (GPIO 1)	31
21	GPIO 5		29		30	GND	
22	GPIO 6		31		32	GPIO 12	26
23	GPIO 13		33		34	GND	
24	GPIO 19		35		36	GPIO 16	27
25	GPIO 26		37		38	GPIO 20	28
	GND		39		40	GPIO 21	29

Tabelle 3.1: Pin-Belegung der GPIO-Schnittstelle



### 3.3 WiringPi

Die Bibliothek WiringPi ermöglicht den einfachen Zugriff auf die GPIOs des Raspberry Pi und wird sehr oft bei Projekten mit einer Programmiersprache wie C/C++ genutzt. (vgl. [Tut152]).

### 3.4 GPIO-Sonderfunktionen

Einige GPIO-Pins des Raspberry Pi bieten zusätzliche Protokolle an. Diese können manuell aktiviert werden. Dazu gehören I2C, SPI und UART. Diese Protokolle mit den dazu gehörigen Pins sind in der Tabelle 3.2 dargestellt (vgl. [RiJ15]).

I2C steht für ein serielles Protokoll für ein zweiadriges Interface und wird verwendet um EEPROM, A/D- und D/A-Konverter, I/O Interfaces und andere Geräte zu verbinden.

SPI (Serial Peripheral Interface) ist ein synchrones Datenbus-System, um digitale Schaltungen nach dem Master-Slave-Prinzip zu verbinden. Dieser Bus findet Einsatz in Embedded Systemen, Sensoren und SD-Karten und wird für kurze Distanzen verwendet.

UART (Universal Asynchronous Receiver Transmitter) dient zur Realisierung digitaler serieller Schnittstellen. Diese bildet eine Grundlage für das Senden und Empfangen von Daten an serielle Schnittstellen mit Computern und Mikrocontrollern (vgl. [Ras154]).

Protokoll	Signale	Pins	Signale	Protokoll
I2C	SDA SCL	1	2	UART
		3	4	
		5	6	
		7	8	
SPI	MOSI MISO SCLK	9	10	
		11	12	
		13	14	
		15	16	
		17	18	
		19	20	
		21	22	
		23	24	
		25	26	
		27	28	
ID EEPROM	ID_SD	27	ID_SC	ID EEPROM

Tabelle 3.2: Pin-Belegung für die Sonderfunktionen der GPIO-Schnittstelle

## 4 Erste Versuche

### 4.1 Zugriff mit einem SSH-Client

Um mit dem Raspberry Pi zu kommunizieren, wird dieser in das Netzwerk per Netzkabel angeschlossen. Anschließend wird sich mit einem SSH-Client (beispielsweise mit dem Programm Putty) auf dem Raspberry Pi angemeldet.<sup>2</sup>

- Benutzername: pi
- Passwort: raspberry

### 4.2 Übertragen von Projektdaten

Zum Übertragen der späteren Projektdaten wird das Programm WinSCP genutzt.<sup>3</sup> Damit können Dateien sehr einfach über das Netzwerk auf dem Raspberry Pi übertragen werden.

### 4.3 Ansteuern einer LED

Der erste Versuch beschäftigt sich mit dem einfachen Ansteuern einer LED (Leuchtdiode). Zur Vorbereitung wird ein Steckbrett, sowie ein paar Verbindungskabel, eine LED und ein Vorwiderstand zwischen 330 bis 1000 Ohm benötigt. In der Abb. 4.1 wird der fertige Versuchsaufbau dargestellt.<sup>4</sup> In diesem Versuchsaufbau wird ein Vorwiderstand von 560 Ohm mit dem Farbcode Grün-Blau-Braun verwendet.<sup>5</sup> Ohne einen Vorwiderstand, der den Strom begrenzt, kann die LED zerstört werden. Anschließend wird dieser mit der LED an den Pins 11 (GPIO 17) und 14 (GND) verbunden. Die längere Verbindung der LED (Anode) muss mit dem GPIO-Pin (Pluspol) und die kürzere Verbindung (Kathode) mit GND (Minuspole) verbunden werden. Das Vertauschen der Anschlüsse würde die LED in Sperrrichtung schalten, das heißt es würde kein Strom fließen und die LED nicht leuchten. Der GPIO-Pin liefert, sobald dieser aktiviert wurde, eine Spannung von +3,3 V und kann daher in diesem Beispiel auch als Spannungsquelle genutzt werden (vgl. [Ada15], [Dev15], [Ore15]).

Zur Ansteuerung der LED wird die Programmiersprache Python (Version: 2.7.9) genutzt, da diese mit wenigen und einfachen Befehlen zum gewünschten Ergebnis führt. Auch andere Programmiersprachen und Skriptsprachen können eingesetzt werden. Zunächst wird die Python Konsole als Root geöffnet. Danach werden die benötigten Bibliotheken eingebunden. Anschließend kann die GPIO-Schnittstelle mit wenigen Befehlen angesteuert werden.

---

<sup>2</sup> Genaue Anleitung zum Anmelden mit dem Programm Putty: <http://tutorials-raspberrypi.de/erste-schritte/ssh-zugriff-einrichten-via-putty/>

<sup>3</sup> Das Programm WinSCP ist unter folgender Adresse erreichbar: <https://winscp.net/eng/docs/lang:de>

<sup>4</sup> Für den theoretischen Aufbau der Elektronischen Bauteile werden die Programme Crocodile Clips und Fritzing genutzt.

<sup>5</sup> Um anhand des Farbcodes auf dem Vorwiderstand dessen Ohm-Wert herauszufinden, wird die folgende Webseite genutzt: <https://www.ph-ludwigsburg.de/html/2f-tech-s-01/studium/Veranstaltungsmaterial/Programme/Widerstand%20Farbcode%204%20und%205%20Ring%20DIN%2041429.htm>

```
$ sudo python
```

```
>>> import RPi.GPIO as GPIO
>>> GPIO.setmode(GPIO.BCM)
>>> GPIO.setup(17, GPIO.OUT)
>>> GPIO.output(17, True)
>>> GPIO.output(17, False)
```

```
# Bibliotheken einbinden
# Auf die GPIO-Nummer beziehen
# GPIO-Pin als Ausgabe nutzen
# LED an
# LED aus
```

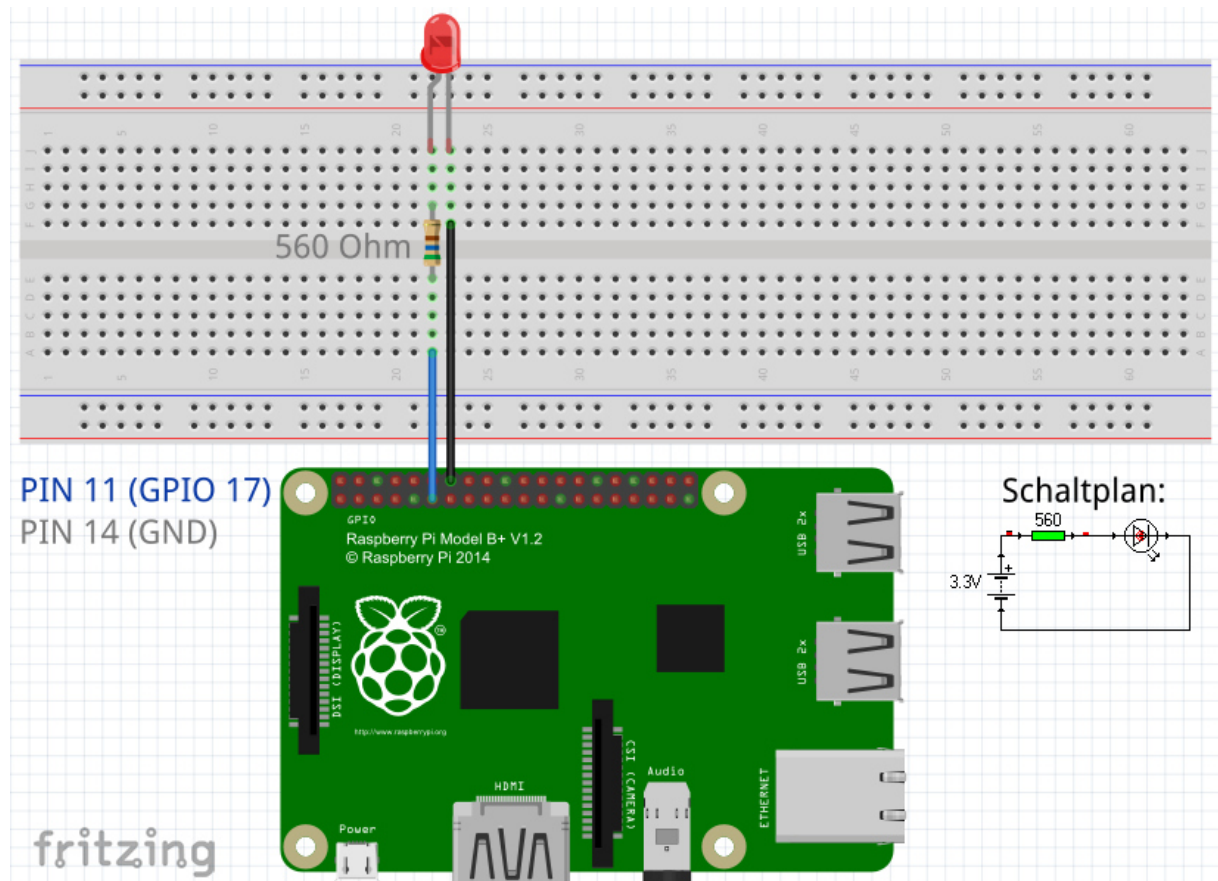


Abb. 4.1: Aufbau zur Ansteuerung einer LED mit der GPIO-Schnittstelle

## 5 Ansteuern von Sensoren

### 5.1 Ansteuern des DHT22-Sensor

Mit dem kostengünstigen Sensor DHT22 können Temperatur und Luftfeuchtigkeit gemessen werden. Der DHT22 (weiß) ist im Gegensatz zum Vorgänger DHT11 (blau) wesentlich präziser und kann auch in extremen Gebieten mit 0 - 20% oder 80 - 100% Luftfeuchtigkeit eingesetzt werden. Der Temperaturbereich des Sensors beträgt ca. -40 bis 80 Grad Celsius. Dieser Sensor hat 4 Pins, die in der Abb. 5.1 dargestellt werden. Der linke Pin 1 des Sensors wird an die Spannungsversorgung (+3,3V) des Raspberry Pi (Pin 1) angeschlossen. Pin 2 des Sensors wird an einen freien GPIO-Pin angeschlossen, in diesem Beispiel an dem Pin 11 (GPIO 17). Weiterhin wird zwischen Pin 1 und Pin 2 des Sensors ein Pullup-Widerstand von 10k Ohm (Farbcode: Braun-Schwarz-Orange) eingesetzt. Der Wert des Widerstandes sollte 4,7k Ohm bis 10k Ohm betragen. Der Pin 3 des Sensors bleibt frei und ungenutzt. Der letzte Pin 4 wird an GND angeschlossen. Dieser Versuchsaufbau ist in der Abb. 5.2 dargestellt (vgl. [Tut153], [Kle15], [Lau15]).

Für das Auslesen der Werte vom Sensor müssen zunächst einige weitere Pakete installiert werden. Danach kann die benötigte Bibliothek von der Adafruit Projektseite heruntergeladen werden. Anschließend wird in das heruntergeladene Verzeichnis gewechselt. In diesem Verzeichnis muss nun die „setup.py“-Datei aufgerufen werden und die Installation beginnt. Zum Schluss kann im Verzeichnis „examples“ das Python-Skript „Adafruit\_DHT.py“ aufgerufen werden. Als Parameter werden der Typ des Sensors (in diesem Fall die 22) und der GPIO-Pin (17) angegeben.

Für das Auslesen der Werte werden immer Root-Rechte benötigt. Nur durch das Verändern des Quellcodes von pi\_mmio.c im Verzeichnis „source/Raspberry\_Pi“, kann dieser auch mit Nutzer-Rechten ausgelesen werden. Dafür muss im Quelltext „/dev/mem“ durch „/dev/gpiomem“ ersetzt werden (vgl. [Ras155]).

```
$ sudo apt-get install build-essential python-dev python-openssl
$ cd Downloads
$ git clone https://github.com/adafruit/Adafruit_Python_DHT.git
$ cd Adafruit_Python_DHT
$ sudo python setup.py install
$ sudo examples/Adafruit_DHT.py 22 17
```

Nach dem Aufrufen erscheint die Ausgabe mit dem Temperatur- und Luftfeuchtigkeitswert:

Temp=19.8* Humidity=60.3%
---------------------------

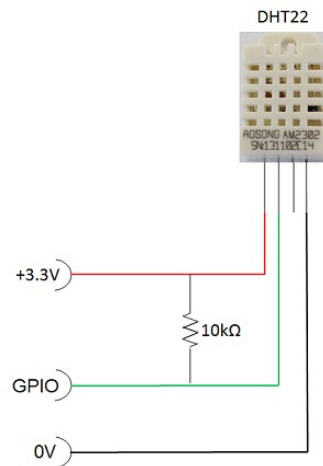


Abb. 5.1: Sensor DHT22 mit den vier Pins (übernommen von [Kle15])

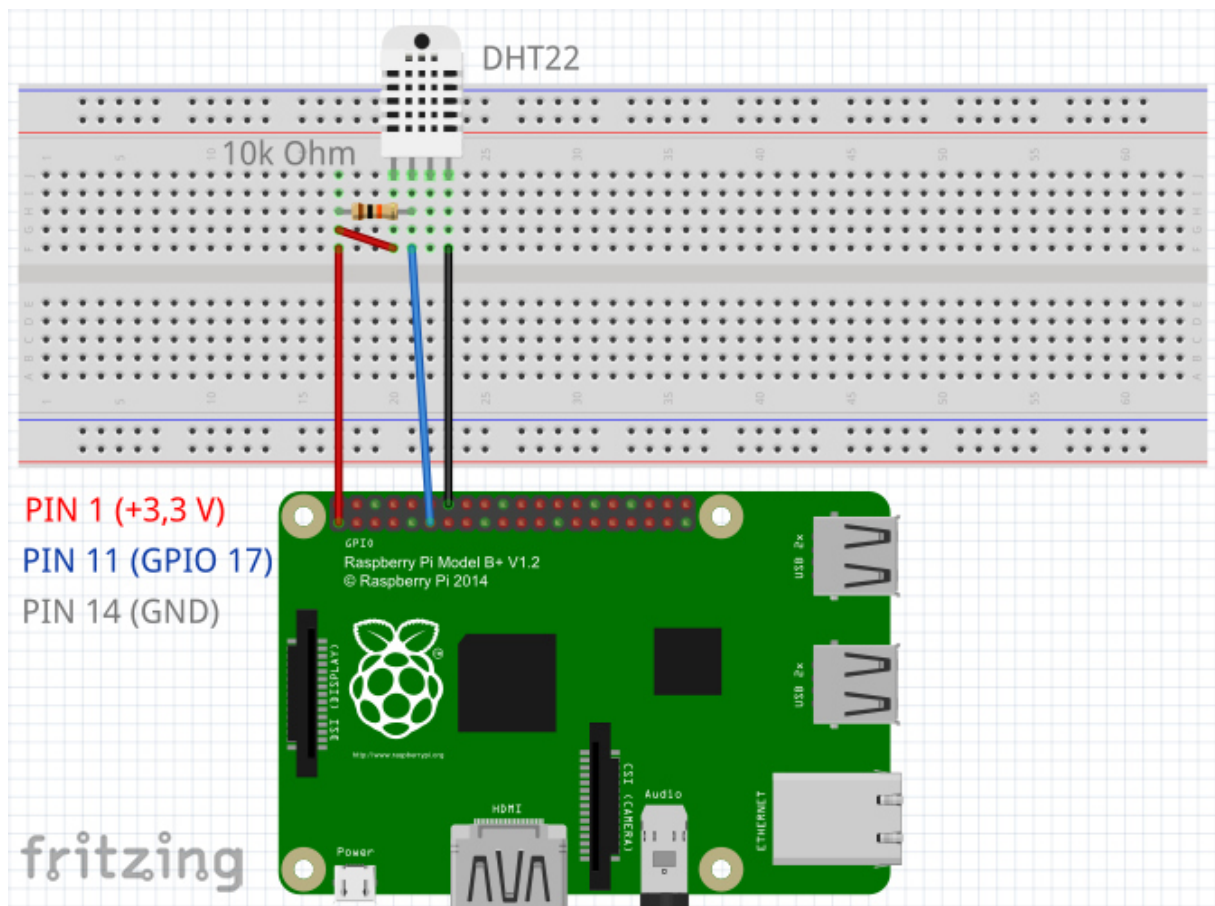


Abb. 5.2: Aufbau zur Ansteuerung des Sensors DHT22 mit der GPIO-Schnittstelle

## 5.2 Ansteuern des DS18B20-Sensor

Mit dem weitverbreiteten Sensor DS18B20 kann die Temperatur (-55 bis 125 Grad Celsius) gemessen werden und gibt einen 9 bis 12 bit langen Wert zurück. Dieser basiert auf dem 1-Wire-Protokoll und kann nur an Pin 7 (GPIO 4) des Raspberry Pi ausgelesen werden. Mehrere dieser Sensoren können jedoch in Reihe geschaltet werden.



Dieser Sensor hat 3 Pins und eine abgeflachte Seite. Von dieser abgeflachten Seite beginnt die Nummerierung der Pins von links nach rechts. Der linke Pin 1 des Sensors wird an GND angeschlossen. Der mittlere Pin 2 des Sensors wird an Pin 7 (GPIO 4) angeschlossen. Der letzte Pin 3 des Sensors wird die Spannungsversorgung (+3,3V) des Raspberry Pi (Pin 1) angeschlossen. Zusätzlich muss ein Widerstand mit 4,7k Ohm (Farbcode: Gelb-Violett-Rot) zwischen den Pins 2 und 3 des Sensors eingesetzt werden. Dieser Versuchsaufbau ist in der Abb. 5.3 dargestellt (vgl. [Tut154], [ITA15]).

Bevor der Sensor ausgelesen werden kann, muss (ab dem Kernel 3.18.3) in der Datei „/boot/config.txt“ folgende Zeile eingetragen werden:

- dtoverlay=w1-gpio-pullup

Nach einem anschließenden Neustart wird der Sensor im Verzeichnis „/sys/bus/w1/devices/“ mit einer weltweit eindeutigen Sensor-ID aufgelistet, welche sich von Sensor zu Sensor unterscheidet. Wird die folgende Datei „/sys/bus/w1/devices/28-021572306eff/w1\_slave“ geöffnet, kann der Temperaturwert direkt ausgelesen werden. Dieser Wert bei „t=“ muss nur durch den Wert 1000 geteilt werden, um den Temperaturwert in Grad Celsius zu erhalten (vgl. [Fhe15]).

```
2c 00 4b 46 ff ff 0f 10 d3 : crc=d3 YES
2c 00 4b 46 ff ff 0f 10 d3 t=21812
```

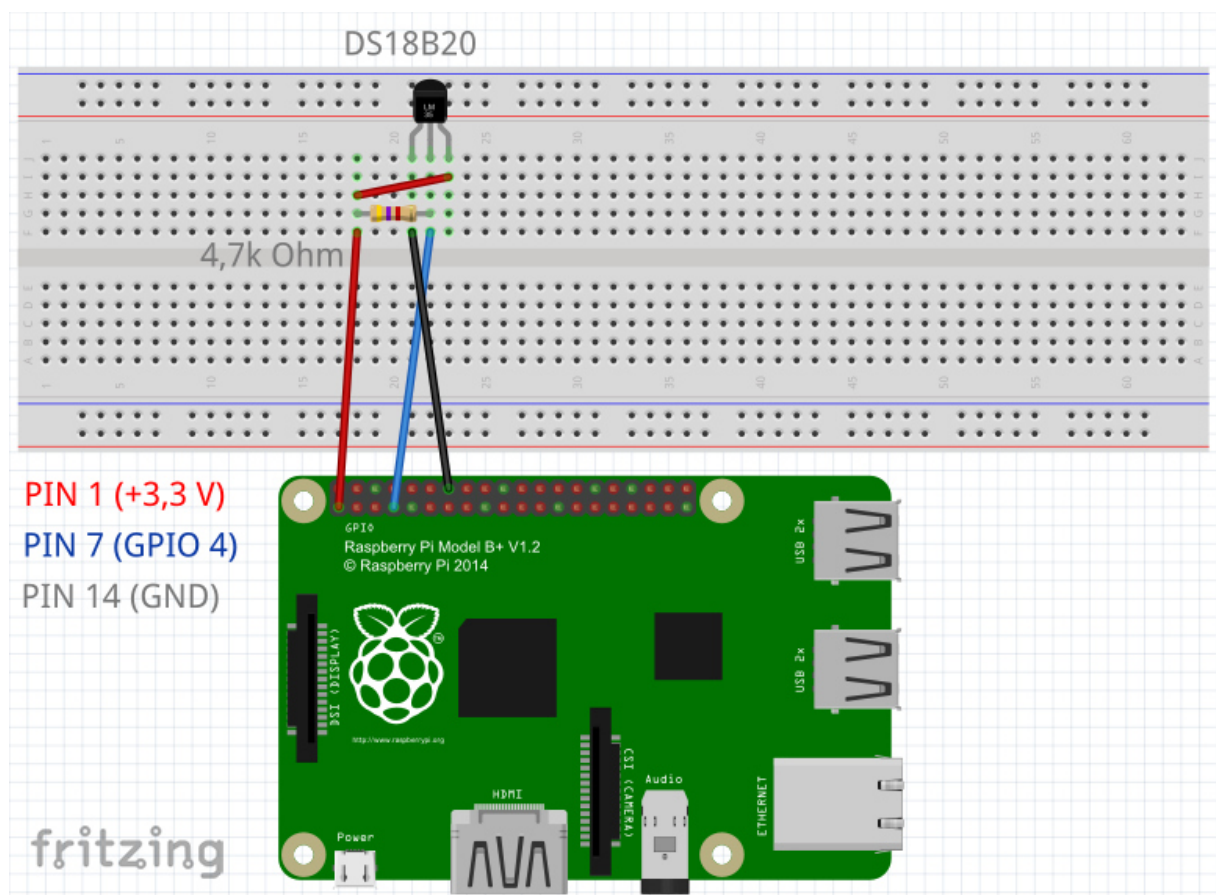


Abb. 5.3: Aufbau zur Ansteuerung des Sensors DS18B20 mit der GPIO-Schnittstelle

### 5.3 Ansteuern des HC-SR04-Sensor

Mit dem Sensor HC-SR04 kann die Entfernung zu einem Objekt vor dem Sensor mit Ultraschall gemessen werden. Aus dem Datenblatt entnommen, beträgt der Messbereich des Sensors 2 bis 400 cm.<sup>6</sup>

Der linke Pin 1 (VCC) des Sensors benötigt eine +5V Spannungsversorgung und wird deshalb mit dem Pin 2 des Raspberry Pi verbunden. Der zweite Pin (Trig) des Sensors wird an Pin 12 (GPIO 18) des Raspberry Pi angeschlossen. Zwischen den Pin 3 (Echo) des Sensors und dem Pin 18 (GPIO 24) des Raspberry Pi muss ein Widerstand von 330 Ohm (Farbcode: Orange-Orange-Braun) gesetzt werden. Der letzte Pin (GND) des Sensors wird an GND (Pin 6) des Raspberry Pi angeschlossen. Zusätzlich muss ein Widerstand von 470 Ohm (Farbcode: Gelb-Violett-Braun) zwischen der GND-Leitung und dem Pin 18 (GPIO 24) eingesetzt werden. Dieser Versuchsaufbau ist in der Abb. 5.4 dargestellt (vgl. [Tut155]).

Anstelle des Widerstandes mit 470 Ohm kann auch ein Widerstand mit 330 Ohm genutzt werden, sodass für den Aufbau lediglich nur zwei Widerstände mit gleichem Wert verwendet werden müssen.

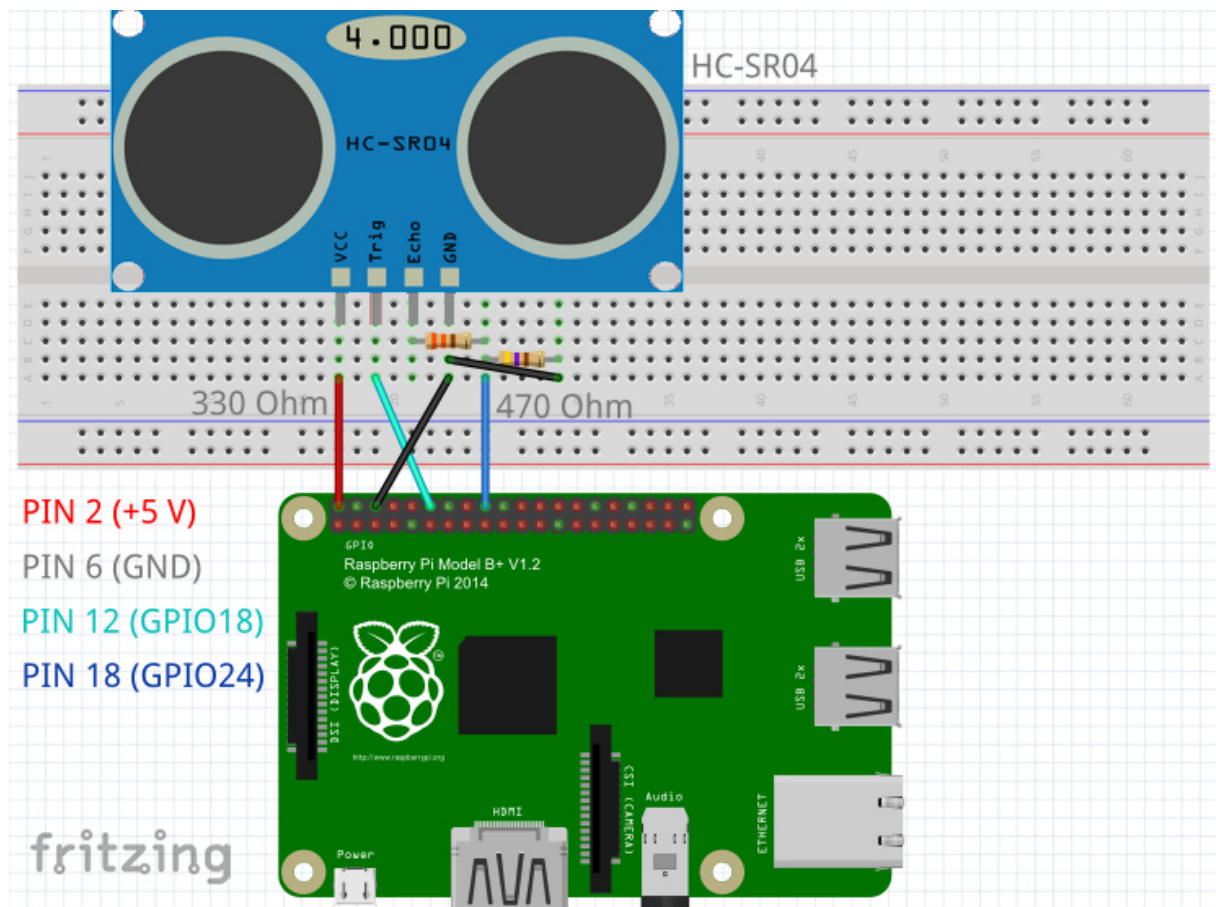


Abb. 5.4: Aufbau zur Ansteuerung des Sensors HC-SR04 mit der GPIO-Schnittstelle

<sup>6</sup> Datenblatt zum Sensor HC-SR04: <http://www.micropik.com/PDF/HCSR04.pdf>

### 5.4 Ansteuern des MPU-6050-Sensor

Der MPU-6050 ist ein Gyroskop/Beschleunigungssensor und kann die Beschleunigung und Rotation der 3 Achsen messen.

Von den 8 Pins des Sensors müssen nur 4 benutzt werden. Der erste Pin 1 (VCC) des Sensors benötigt eine +3,3V Spannungsversorgung und wird mit dem Pin 1 des Raspberry Pi verbunden. Der zweite Pin (GND) des Sensors wird an Pin 14 (GND) des Raspberry Pi angeschlossen. Der dritte Pin 3 (SCL) des Sensors muss mit dem Pin 5 (GPIO 3 / SCL) des Raspberry Pi verbunden. Der letzte benötigte Pin 4 (SDA) des Sensors muss mit dem Pin 3 (GPIO 2 / SDA) des Raspberry Pi angeschlossen werden.

Für diesen Sensor müssen die Sonderfunktionen für das I2C aktiviert werden, daher kann dieser Sensor nur an die hier vorgegebenen Pins angeschlossen werden. Dieser Versuchsaufbau ist in der Abb. 5.5 dargestellt (vgl. [Tut156]).

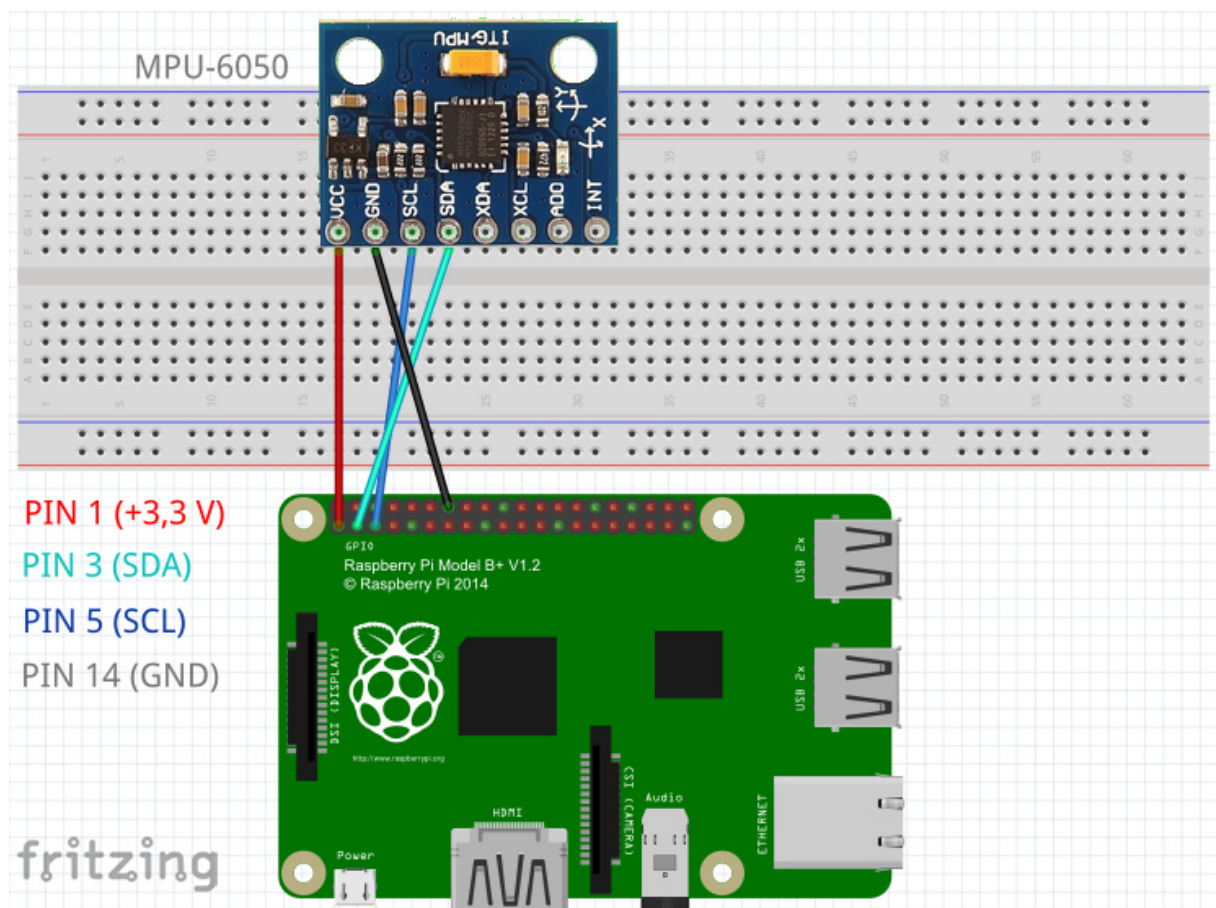


Abb. 5.5: Aufbau zur Ansteuerung des Sensors MPU-6050 mit der GPIO-Schnittstelle

### 5.5 Fazit

Die Sensoren können sehr einfach mit der GPIO-Schnittstelle des Raspberry Pi verbunden werden. Für den Aufbau von einigen Sensoren sind weitere Bauteile wie z. B. Widerstände notwendig. Bevor Messwerte ausgelesen werden können, müssen je nach Sensor zunächst Konfigurationsdateien im System angepasst oder Installationsroutinen durchlaufen werden. Erst danach können die Messwerte mit wenigen Python-Befehlen ausgelesen werden.



## 6 Konfigurierbarkeit

### 6.1 Ordnerstruktur

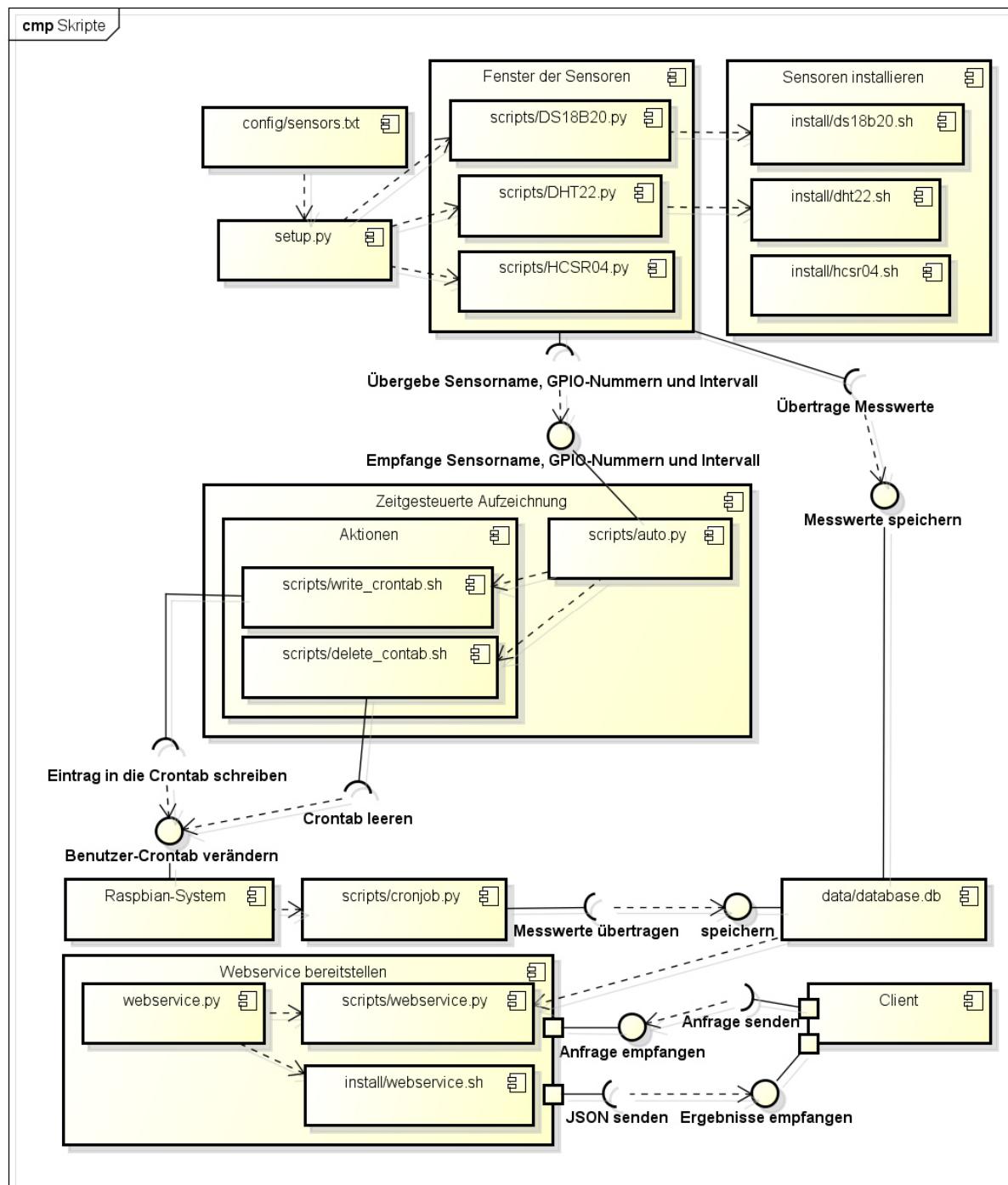
Für das Projekt wird eine Ordnerstruktur angelegt. Im Verzeichnis „config“ in der Datei „sensors.txt“ stehen die verfügbaren Sensoren. Die genutzte SQLite-Datenbank „data.db“ zum Speichern aller Messwerte wird im Verzeichnis „database“ abgelegt. Die Datenblätter zu den Sensoren, der GPIO-Schnittstelle, sowie die Entwickler- und Anwenderdokumentation befinden sich im Ordner „datasheet“. Alle benötigten Pakete, die für das Installieren der Sensoren benötigt werden, befinden sich im Verzeichnis „downloads“. Das Verzeichnis „gui“ dient als Speicherort für die Darstellung der verwendeten Bilder in der Bedienoberfläche. Einige Sensoren benötigen zusätzlich vor dem Auslesen eine Installationsroutine, diese Installationsskripte befinden sich im Verzeichnis „install“. Das letzte Verzeichnis „logfiles“ dient dazu, dass bei der Benutzung der Skripte diese mitprotokolliert und als Textdatei gespeichert werden. Alle sonstigen Skripte des Projektes befinden sich im Ordner „scripts“. Diese Skripte werden kurz in der Tabelle 6.1 erläutert.

Datei	Beschreibung
<b>auto.py</b>	Stellt das Fenster zum Konfigurieren der zeitgesteuerten Aufzeichnung dar.
<b>cronjob.py</b>	Diese Datei wird vom Benutzer-Crontab automatisch zu einer vorgegebenen Zeit ausgeführt, um die Messwerte der Sensoren auszulesen.
<b>delete_crontab.sh</b>	Löscht alle Jobs, die für die zeitgesteuerte Aufzeichnung dienen, in der Benutzer-Crontab.
<b>DHT22.py</b>	Stellt das Fenster zum Konfigurieren des Sensors dar.
<b>DS18B20.py</b>	Stellt das Fenster zum Konfigurieren des Sensors dar.
<b>HCSR04.py</b>	Stellt das Fenster zum Konfigurieren des Sensors dar.
<b>sensor.py</b>	Diese Bibliothek stellt Methoden für das Auslesen der Sensoren zur Verfügung.
<b>webservice.py</b>	Startet einen Webservice, um die Messwerte als JSON Plattformübergreifen zur Verfügung stellen zu können.
<b>write_crontab.sh</b>	Schreibt einen Job für die zeitgesteuerte Aufzeichnung in die Benutzer-Crontab.

Tabelle 6.1: Dateien im Ordner „scripts“

Im Hauptverzeichnis befinden sich die drei Dateien „README“, „setup.py“ und „webservice.py“. Die „README“-Datei sollte vor dem ersten Start durchgelesen werden. Um das Projekt zu starten, dient die Python-Datei „setup.py“. Diese wird gestartet und es öffnet sich das grafische Bedienfenster. Die Datei „webservice.py“ sollte gestartet werden, um die Daten als RESTful-Webservice bereitzustellen. In der Abb. 6.1 sind die Abhängigkeiten der einzelnen Skripte als Komponentendiagramm dargestellt. Die Ordnerstruktur wird im nachfolgenden zusammengefasst:

- **Sensorstation**
  - **config**
    - sensors.txt
  - **data**
    - database.db
    - dht22.csv
    - ds18b20.csv
    - hcsr04.csv
  - **datasheet**
    - Anwenderdokumentation.pdf
    - DHT22.pdf
    - DS18B20.pdf
    - Entwicklerdokumentation.pdf
    - GPIO.pdf
    - HC-SR04.pdf
    - MPU-6050.pdf
  - **downloads**
    - **Adafruit\_Python\_DHT**
  - **gui**
    - DHT22.png
    - DS18B20.png
    - HCSR04.png
    - LED.png
    - MPU6050.png
  - **install**
    - dht22.sh
    - ds18b20.sh
    - hcsr04.sh
    - webservice.sh
  - **logfiles**
    - sensorstation.txt
  - **scripts**
    - auto.py
    - cronjob.py
    - delete\_crontab.sh
    - DHT22.py
    - DS18B20.py
    - HCSR04.py
    - sensor.py
    - webservice.py
    - write\_crontab.sh
  - README
  - setup.py
  - webservice.py



powered by Astah

Abb. 6.1: Komponentendiagramm für die Abhängigkeiten der einzelnen Skripte

## 6.2 Konfiguration

Die Textdatei „sensors.txt“ im Verzeichnis „config“ beinhaltet die verfügbaren Sensoren für die Sensorstation. Zurzeit beschränkt sich das Projekt auf die drei Sensoren DS18B20, DHT22 und HCSR04.

Zeilenweise stehen die verfügbaren Sensoren in dieser Textdatei. Getrennt durch das erste Komma enthält eine Zeile Informationen über den Sensor. Vor dem ersten Komma wird der Sensorname und nach dem ersten Komma können weitere Metainformationen über den Sensor angegeben werden wie z. B. die Messart des Sensors.

DS18B20, Temperatur
DHT22, Temperatur, Luftfeuchtigkeit
HCSR04, Entfernung

## 6.3 Erster Start

Zunächst müssen alle Skripte die benötigten Ausführungsrechte besitzen. Dazu wird in der Kommandozeilebene im Projektverzeichnis folgender Befehl genutzt:

- `chmod +x . -R`

Anschließend kann die Datei „setup.py“ gestartet werden. Dies geschieht durch einen Doppelklick auf die Datei und „Ausführen“. Alternativ kann die Datei auch im Terminal ausgeführt werden. Nach dem Ausführen öffnet sich eine grafische Oberfläche, bei dem die entsprechenden Sensoren ausgewählt und konfiguriert werden können.

## 6.4 Installation der Sensoren

Bevor die Sensorstation vollständig genutzt werden kann, müssen einige Sensoren wie z. B. der DS18B20 und DHT22 im System installiert werden. Die Sensorstation muss dafür mit Root-Rechten ausgeführt werden. Dazu wird das Terminal aufgerufen und im Projektverzeichnis folgender Befehl ausgeführt:

- `sudo ./setup.py`

## 6.5 Sensorstation nutzen

Wie in der Abb. 6.2 dargestellt, wird nach dem Start der Sensor ausgewählt. Nach der Wahl des Sensors kann dieser durch einen Klick auf den Button „Konfigurieren“ konfiguriert werden. Anschließend erscheint wie in Abb. 6.3 dargestellt ein neues Fenster für die Konfiguration. Wie der Sensor angeschlossen werden kann, wird in diesem Fenster auf der rechten Seite angezeigt. Auf der linken Seite müssen die genutzten GPIO-Nummern und das Intervall für die Abfrage der Werte angegeben werden. In der Mitte sind alle möglichen Aktionen der Sensoren dargestellt. Diese Aktionen sind in der Tabelle 6.2 kurz erläutert.

Die Sensorstation beschränkt sich auf die Nutzung von einem einzigen Sensor pro Typ. Mehrere Sensoren des gleichen Typs können nicht unterschieden werden.

Aktion	Beschreibung
<b>Sensor installieren</b>	Starten eine Installationsroutine, die notwendig für einige Sensoren sind. Benötigt dazu Root-Rechte!
<b>Sensor testen</b>	Prüft, ob der Sensor Werte liefert, und gibt diese als Message-Box aus.
<b>Aufzeichnung starten</b>	Startet dauerhaft das Auslesen des Sensors im eingegebenen Intervall, bis dieser gestoppt wird.
<b>Zeitgesteuerte Aufzeichnung</b>	Öffnet ein Fenster, in dem ein Job für die Aufzeichnung erstellt werden kann.
<b>Export als CSV-Datei</b>	Exportiert die Daten als CSV-Datei im Ordner „data“.
<b>Beenden</b>	Schließt das Fenster.

Tabelle 6.2: Aktionen der Sensoren



Abb. 6.2: Auswählen des Sensors

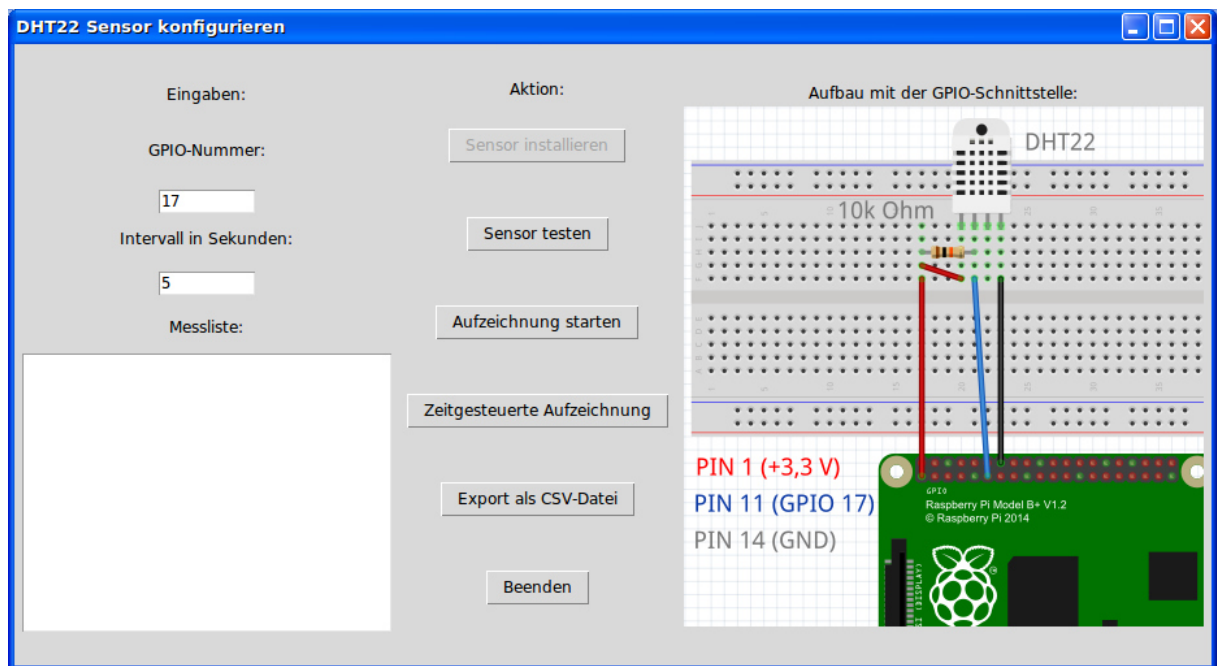


Abb. 6.3: Sensor DHT22 konfigurieren

## 6.6 Automatisierte Aufzeichnung

Für die automatisierte Aufzeichnung wird die vom System bereitgestellte Crontab genutzt. Dadurch können Skripte automatisch zu einer ausgewählten Zeit im Hintergrund gestartet werden. Um die Crontab als Benutzer zu nutzen, sind nur wenige Befehle notwendig. Mit dem Befehl „`crontab -l > crontab.txt`“ werden alle Cronjobs in eine Textdatei geschrieben und mit dem Befehl „`crontab < crontab.txt`“ kann die Textdatei dem Crontab übergeben werden.

Die verschiedenen Jobs werden zeilenweise eingetragen. Jeder Job enthält 6 Parameter.

Die Cronjobs in der Crontab sind wie folgt aufgebaut: \* \* \* \* \* Befehl

Der Stern ist ein besonderer Parameter und steht für jeden möglichen Eintrag (vgl. [Ste15]).

1. Minute von 0-59
2. Stunde von 0-23
3. Tag von 1-31
4. Monat von 1-12
5. Wochentag von 0-7 (Sonntag ist die 0 und 7)
6. Befehl der ausgeführt werden soll

Durch einen Klick auf den Button „Zeitgesteuerte Aufzeichnung“ erscheint wie in der Abb. 6.4 dargestellt ein neues Fenster, in dem Jobs erstellt werden können. Zuerst muss eine Startzeit festgelegt werden, an dem die Messung automatisch beginnt. Diese Einstellung kann durch fünf Angaben (Monat, Wochentag, Tag, Stunde und Minute) spezifiziert werden. Danach muss die Laufzeit der Messung nach dem Start in Minuten angegeben werden. Einige Beispiele für die Nutzung sind in der Tabelle 6.3 dargestellt.

Um einen eingestellten Job zu übernehmen, wird der Button „Neuen Job“ genutzt. Anschließend werden alle aktiven Jobs im unteren Feld angezeigt. Alle Jobs können mit dem Button „Alle Jobs entfernen“ entfernt werden.

Aktion	Laufzeit[Min]	Monat	Wochentag	Tag	Stunde	Min.
Jeden Abend um 20:30 Uhr bis 21:30 Uhr	60	*	*	*	20	30
Jeden Montag um 12:00 Uhr bis 14:00 Uhr	120	*	Mo.	*	12	00
Jede volle Stunde für 10 Minuten	10	*	*	*	*	00
Den ganzen Tag am 24.12	1440	12	*	24	00	00

Tabelle 6.3: Beispiele für die Benutzung der zeitgesteuerten Aufzeichnung

**Zeitgesteuerte Aufzeichnung mit dem Sensor DHT22**

Startzeit:

Monat:  Wochentag:  Tag:  Stunde:  Minute:

Laufzeit in Minuten:

Alle aktiven Jobs:

Job 1: Monat: \*, Wochentag: \*, Tag: \*, Stunde: 21, Minute: 30,  
Sensor: DHT22, Laufzeit[Min]: 30, Intervall[Sek]: 5,  
GPIO1: 17

Aktion:

Abb. 6.4: Zeitgesteuerte Aufzeichnung

## 7 Speicherung

### 7.1 SQLite-Datenbank

Die Messwerte werden in einer SQLite-Datenbank im Verzeichnis „data/database.db“ geschrieben. Diese wird automatisch vor dem ersten Aufzeichnen von Messwerten angelegt. Die Daten in der SQLite-Datenbank sind durch Tabellen strukturiert. Jeder Sensor besitzt eine eigene Tabelle, in dem die Messwerte gespeichert werden. Der Primärschlüssel der jeweiligen Daten steht im Feld „timestamp“ und dient der Speicherung des Datums und der Zeit. In den Feldern wie z. B. „temperature“, „humidity“ und „distance“ werden die Messwerte der jeweiligen Sensoren gespeichert.

Die Tabelle wird im Folgenden am Beispiel des Sensors DHT22 dargestellt:

Tabelle: dht22		
timestamp	temperature	humidity
2015-12-27 11:14:43	20.543	50.123
2015-12-27 11:14:48	20.551	50.132

### 7.2 CSV-Daten

Um die Daten beispielsweise mit Excel zu visualisieren, können die Messdaten als CSV-Datei (Comma Separated Values) gespeichert werden. Eine CSV-Datei beinhaltet eine Kopfzeile mit einer Beschreibung der Spalten und dann zeilenweise alle Messdaten, die durch ein Trennzeichen getrennt sind.

Das folgende Beispiel stellt den Inhalt einer CSV-Datei (Sensor: DHT22) dar:

```
timestamp,temperature,humidity
2015-12-27 11:14:43,20.543,50.123
2015-12-27 11:14:48,20.551,50.132
```

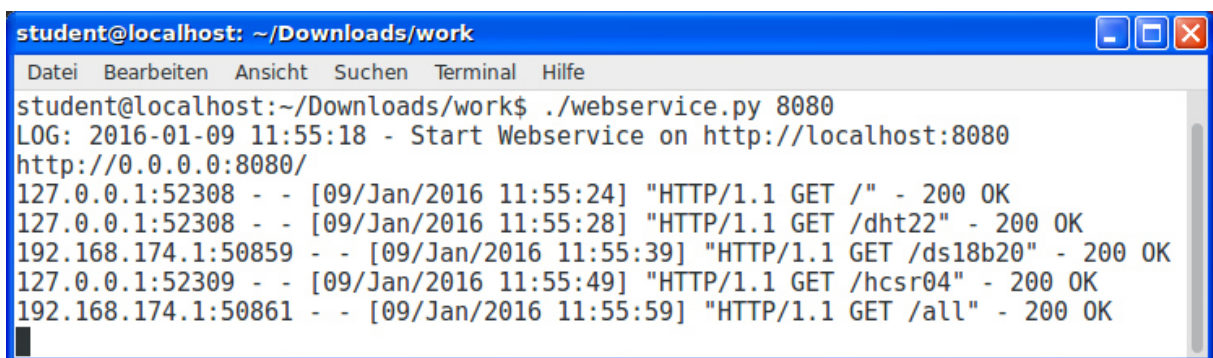


## 8 Webbasierte Datenabfrage

### 8.1 RESTful-Webservice

Alle Messdaten können als Webservice plattformübergreifend bereitgestellt werden. Hierfür muss die Datei „webservice.py“ im Terminal ausgeführt werden. Für den ersten Start muss dieses Skript mit Root-Rechten ausgeführt werden. Anschließend wird das benötigte Paket „python-webpy“ installiert. Der Standardport zum kommunizieren mit dem Webservice ist 8080. Um diesen Port zu verändern, kann das Skript mit einem Parameter gestartet werden.

- `sudo ./webservice.py 8080`
- `./webservice.py 8080`



```
student@localhost: ~/Downloads/work
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
student@localhost:~/Downloads/work$ ./webservice.py 8080
LOG: 2016-01-09 11:55:18 - Start Webservice on http://localhost:8080
http://0.0.0.0:8080/
127.0.0.1:52308 - - [09/Jan/2016 11:55:24] "HTTP/1.1 GET /" - 200 OK
127.0.0.1:52308 - - [09/Jan/2016 11:55:28] "HTTP/1.1 GET /dht22" - 200 OK
192.168.174.1:50859 - - [09/Jan/2016 11:55:39] "HTTP/1.1 GET /ds18b20" - 200 OK
127.0.0.1:52309 - - [09/Jan/2016 11:55:49] "HTTP/1.1 GET /hcsr04" - 200 OK
192.168.174.1:50861 - - [09/Jan/2016 11:55:59] "HTTP/1.1 GET /all" - 200 OK
```

Abb. 8.1: Webservice im Terminal starten

Nach dem Start kann der Client mit dem Webservice kommunizieren. Zum Testen wird der Webbrowser „Iceweasel“ genutzt und die URL „localhost:8080“ aufgerufen. Hier erscheint eine Webseite, indem die verfügbaren Sensoren aufgelistet sind. Mit dem Verzeichnis z. B. „/dht22“ werden nur die Messwerte des jeweiligen Sensors aus der SQLite-Datenbank ausgelesen. Um alle Auszulesen wird das Verzeichnis „/all“ genutzt. Weitere Parameter z. B. rows, pretty und calc können genutzt werden, um die Ausgabe zu verändern. Die Beschreibung der einzelnen Parameter sind in der Tabelle 8.1 dargestellt. Der Aufruf mit dem Webbrowser ist in der Abb. 8.2 dargestellt.

Aufruf / Parameter	Beschreibung
/dht22	Zeigt die Werte des Sensors DHT22.
/ds18b20	Zeigt die Werte des Sensors DS18B20.
/hcsr04	Zeigt die Werte des Sensors HCSR04.
/all	Zeigt die Werte aller Sensoren an.
?rows=3	Zeigt die letzten drei Werte an.
?calc	Berechnet den Mittelwert, Maximalen und Minimalen Wert über alle Messwerte und gibt diese aus.
?pretty	Formatiert das JSON in leserlicher Form (Für den Produktiven Einsatz ungeeignet)
?rows=3&calc&pretty	Kombinierte Abfrage der drei Parameter mit dem &-Zeichen

Tabelle 8.1: Parameter für den Aufruf des Webservices

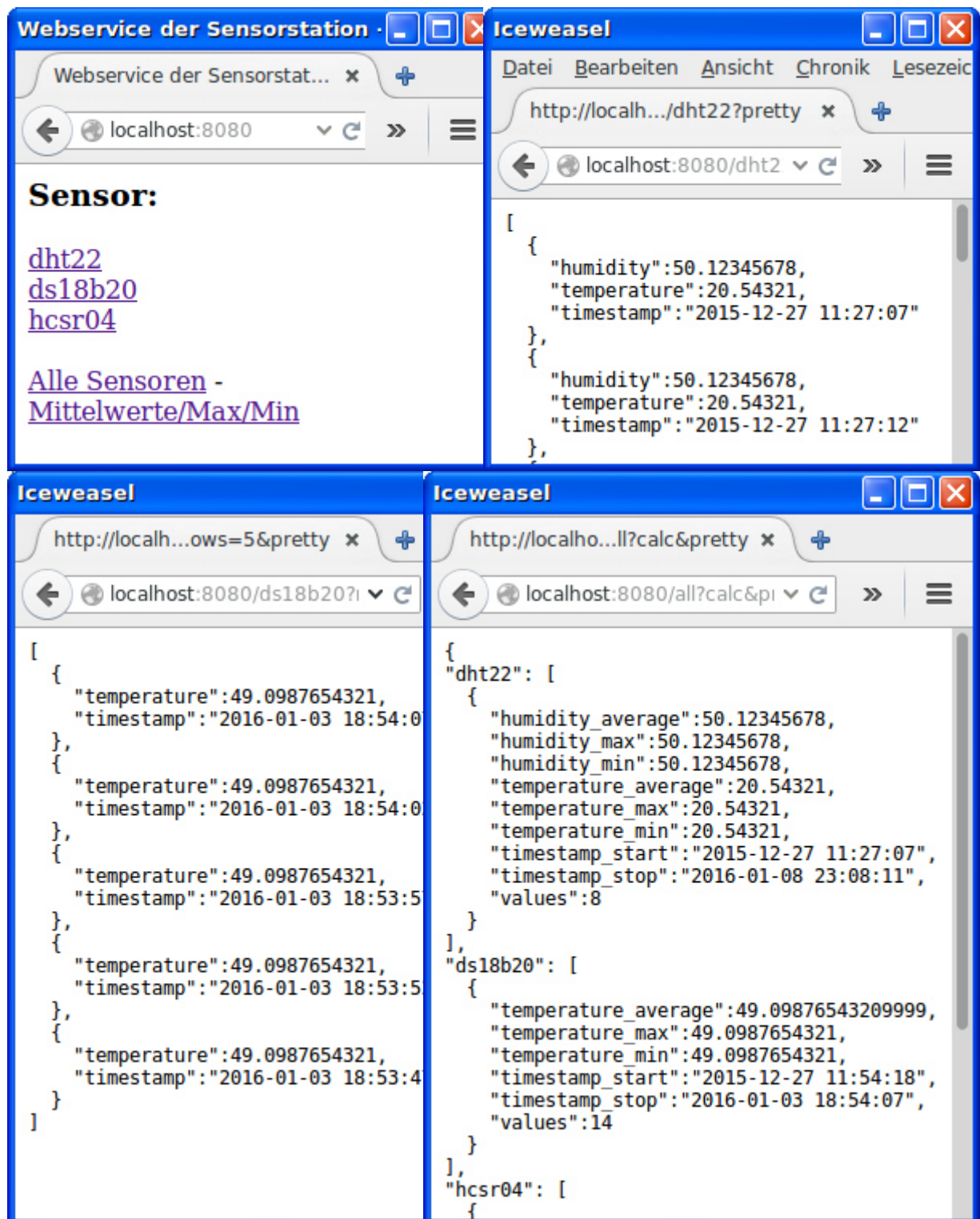


Abb. 8.2: Kommunikation mit dem Webservice

## 9 Zusammenfassung und Ausblick

### 9.1 Zusammenfassung

Der Raspberry Pi ermöglicht das Anschließen von verschiedensten Sensoren an die GPIO-Schnittstelle. Einige Sensoren können frei an die GPIO-Schnittstelle angeschlossen werden und andere wie z. B. der DS18B20 müssen an bestimmte GPIO-Pins angeschlossen werden. Alle getesteten Sensoren benötigen mind. 3 Leitungen, eine für die Spannungsversorgung von +3,3 bis +5 Volt, eine für die Erdung und eine oder mehr für die GPIO-Pins. Zusätzlich müssen einige Sensoren noch mit Widerständen beschaltet werden. Alle Sensoren funktionieren unterschiedlich und sind nicht einheitlich. Einige benötigen eine Installationsroutine oder Systemdaten müssen verändert werden, bevor diese ausgelesen werden können. Das Auslesen der Sensoren wurde in der Sensorstation als eigene Bibliothek zusammengefasst, sodass durch wenige Befehle jeder Sensor einfach ausgelesen werden kann.

Mit der entwickelten Sensorstation können die Sensoren sehr einfach im System installiert und konfiguriert werden. Der entwickelte Prototyp unterstützt derzeit nur jeweils einen Sensor pro Typ und nur 3 bestimmte Sensoren (DHT22, DS18B20 und HC-SR04). Alle diese Sensoren können gleichzeitig manuell durch Knopfdruck sowie automatisch per Zeiteinstellung ausgelesen werden. Die gespeicherten Messdaten in der SQLite-Datenbank können anschließend als Webservice und als CSV-Daten bereitgestellt werden.

### 9.2 Ausblick und weiterführende Arbeiten

Der aktuelle Prototyp unterstützt einige Sensoren zum Auslesen und könnte als mobile Sensorstation in Schulen eingesetzt werden.

Die Sensorstation unterstützt zurzeit nur eine Art pro Sensor. Eine weiterführende Arbeit wäre die Implementation mehrerer Sensoren der gleichen Art. Hierfür müsste den Sensoren eine ID oder einen eindeutigen Namen in der Sensorstation zugewiesen werden. Jeder Sensor erhält derzeit seine eigene Tabelle. Für den Betrieb von mehreren Sensoren der gleichen Art müssten mehrere Tabellen mit unterschiedlichen Namen für die Sensoren angelegt werden. Hierfür könnte ein individueller Name zur Identifikation des Sensors in der Sensorstation vergeben werden.

Eine weiterführende Arbeit würde sich mit der Unterstützung weiterer Sensoren für die Sensorstation beschäftigen. Zur Realisierung müssten folgende Dinge angepasst werden. Zuerst müsste die „config/sensors.txt“ Textdatei editiert werden und die neuen Sensoren ergänzt werden. Anschließend müssen folgende Skripte im Verzeichnis „scripts“ angepasst werden:

- cronjob.py
- sensor.py

- webservice.py

Neue Sensoren benötigt ein eigenes individuelles Fenster. Dafür können die Fenster der Sensoren DHT22.py, DS18B20.py und HCSR04.py als Vorlage genutzt werden. Diese befinden sich im Verzeichnis „scripts“. Der Name des Skripts entspricht dem eingetragenen Sensor in der Konfigurationsdatei. Auch ein PNG-Bild für die Nutzung der GPIO-Schnittstelle mit dem Sensor müsste erstellt werden. Diese Bilder zum Anzeigen der Verdrahtung könnten später auch durch andere ohne Steckbrett ersetzt werden.

Auch eine weiterführende Arbeit würde das Bereitstellen der Daten betreffen. Um nicht immer die kompletten Daten als CSV-Datei exportieren zu müssen, könnten diese durch eine Bereichsangabe mithilfe von zwei Timestamps gezielt selektiert werden. Das gleiche Prinzip könnte auch im Webservice angewendet werden, sodass gezielt nach Daten zwischen zwei Timestamps gesucht werden kann.

## Glossar

<b>Crontab</b>	Dient der zeitbasierten Ausführung von Prozessen in Unix-Betriebssystemen.
<b>EEPROM</b>	Bezeichnet einen elektronisch löschbaren programmierbaren Nur-Lese-Speicher (Erasable Programmable Read-Only Memory).
<b>GPIO</b>	Schnittstelle (General Purpose Input/Output) zum Verbinden externer Geräte.
<b>GND</b>	Ground bedeutet der Minuspol 0V (Masseleitung) eines Elektrischen Stromkreises.
<b>Job</b>	Bezeichnet einen einzelnen Auftrag, der vom Betriebssystem im Hintergrund ausgeführt wird.
<b>Primärschlüssel</b>	Besteht aus einem Attribut, der den dazugehörigen Datensatz eindeutig kennzeichnet.
<b>Pullup-Widerstand</b>	Bezeichnet einen Widerstand, der genutzt wird, um den Eingangspegel mit einem Gleichspannungspegel anzuheben. Dadurch können undefinierte Potentiale ausgeschlossen und Störsignale vermieden werden.
<b>REST</b>	„Representational State Transfer“ bezeichnet eine Schnittstelle eines Webservices.
<b>Root-Rechte</b>	Bezeichnet die höchsten Ausführungsrechte, um Änderungen am System vornehmen zu können.
<b>SQLite</b>	Bezeichnet eine Programmbibliothek, die ein relationales Datenbanksystem enthält.
<b>Webservice</b>	Ist ein Dienst, um eine Kommunikation zwischen Maschinen herzustellen.

## Quellenverzeichnis

- [Ada15] Adafruit; <https://learn.adafruit.com/debugging-with-the-raspberry-pi-webide/debug-a-blinking-led> (31.10.2015).
- [Dev151] Developer-Blog; <https://developer-blog.net/hardware/raspberry-pi-gpio-schnittstelle-teil-2/> (31.10.2015).
- [Ele151] Elektronik Kompendium; <http://www.elektronik-kompendium.de/sites/com/1904221.htm> (06.10.2015).
- [Ele152] Elektronik Kompendium; <https://www.elektronik-kompendium.de/sites/raspberry-pi/1905251.htm> (06.10.2015).
- [Ele153] Elektronik Kompendium; <https://www.elektronik-kompendium.de/sites/raspberry-pi/2002191.htm> (13.10.2015).
- [Ele154] Elektronik Kompendium; <http://www.elektronik-kompendium.de/sites/raspberry-pi/2002061.htm> (06.10.2015).
- [Fhe15] Fhem-Wiki; [http://www.fhemwiki.de/wiki/Raspberry\\_Pi\\_und\\_1-Wire#ab\\_2015\\_bzw.\\_Kernelversion\\_3.18.3](http://www.fhemwiki.de/wiki/Raspberry_Pi_und_1-Wire#ab_2015_bzw._Kernelversion_3.18.3) (07.12.2015).
- [ITA15] IT-Adviser; <http://www.it-adviser.net/raspberry-pi-temperaturmessung-mit-ds18b20-1-wire-sensor/> (26.11.2015).
- [Kle15] Klenzel; <https://klenzel.de/1827> (10.11.2015).
- [Lau15] Laubenstein-Heyden; <http://laubenstein-heyden.de/raspberry-pi-dht-22-temperatur-und-feuchtigkeits-sensor-anschliessen/> (10.11.2015).
- [Net151] Netzmafia; [http://www.netzmafia.de/skripten/hardware/RasPi/RasPi\\_GPIO.html](http://www.netzmafia.de/skripten/hardware/RasPi/RasPi_GPIO.html) (06.10.2015).
- [Ore15] O'Reilly Raspberry Pi Cookbook; <http://razzpisampler.oreilly.com/ch03.html> (31.10.2015).
- [PiJ15] The Pi4J Project; <http://pi4j.com/pins/model-b-plus.html> (13.10.2015).
- [Ras151] Raspberrypiguide; <http://raspberrypiguide.de/#Einstieg> (06.10.2015).
- [Ras152] Raspberrypiguide; <http://raspberrypiguide.de/howtos/raspberry-pi-gpio-how-to/> (06.10.2015).
- [Ras153] Raspberry Pi Lab; <https://raspilab.wordpress.com/2014/08/06/gpio-pins-bei-modell-b-ab-juli-2014/> (13.10.2015).
- [Ras154] Raspberry.Tips; <http://raspberrypi.tips/faq/raspberry-pi-spi-und-i2c-aktivieren/> (13.10.2015).
- [Ras155] Raspberry Pi Stack Exchange; <http://raspberrypi.stackexchange.com/questions/36520/raspberry-pi-2-error-accessing-gpio-with-sudo-for-adafruitdht-and-am2302-dht22-s> (14.01.2016).
- [Ste15] <https://www.stetic.com/developer/cronjob-linux-tutorial-und-crontab-syntax.html> (27.12.2015).
- [Tut151] Tutorials-Raspberrypi; <http://tutorials-raspberrypi.de/erste-schritte/raspberry-pi-einstieg-was-brauche-ich-und-wie-starte-ich/> (13.10.2015).
- [Tut152] Tutorials-Raspberrypi; <http://tutorials-raspberrypi.de/gpio/wiringpi-installieren-pinbelegung/> (31.10.2015).
- [Tut153] Tutorials-Raspberrypi; <http://tutorials-raspberrypi.de/gpio/luftfeuchtigkeit-und-temperatur-mit-dht11-dht22-mit-dem-raspberry-pi-messen/> (10.11.2015).
- [Tut154] Tutorials-Raspberrypi; <http://tutorials-raspberrypi.de/gpio/temperatur-mittels-sensor-messen/> (10.11.2015).
- [Tut155] Tutorials-Raspberrypi; <http://tutorials-raspberrypi.de/gpio/entfernung-messen-mit-ultraschallsensor-hc-sr04/> (13.12.2015).
- [Tut156] Tutorials-Raspberrypi; <http://tutorials-raspberrypi.de/gpio/rotation-und-beschleunigung-mit-dem-raspberry-pi-messen/> (22.12.2015).

## Abbildungsverzeichnis

Abb. 1.1: UseCase-Diagramm zur Verwendung der Sensorstation .....	1
Abb. 2.1: Komponenten vom Raspberry Pi B+ (übernommen von [Ele152]) .....	3
Abb. 3.1: Anordnung der Pins (übernommen von [PiJ15]) .....	4
Abb. 4.1: Aufbau zur Ansteuerung einer LED mit der GPIO-Schnittstelle .....	8
Abb. 5.1: Sensor DHT22 mit den vier Pins (übernommen von [Kle15]) .....	10
Abb. 5.2: Aufbau zur Ansteuerung des Sensors DHT22 mit der GPIO-Schnittstelle .....	10
Abb. 5.3: Aufbau zur Ansteuerung des Sensors DS18B20 mit der GPIO-Schnittstelle .....	11
Abb. 5.4: Aufbau zur Ansteuerung des Sensors HC-SR04 mit der GPIO-Schnittstelle .....	12
Abb. 5.5: Aufbau zur Ansteuerung des Sensors MPU-6050 mit der GPIO-Schnittstelle .....	13
Abb. 6.1: Komponentendiagramm für die Abhängigkeiten der einzelnen Skripte .....	16
Abb. 6.2: Auswählen des Sensors .....	18
Abb. 6.3: Sensor DHT22 konfigurieren .....	18
Abb. 6.4: Zeitgesteuerte Aufzeichnung .....	20
Abb. 8.1: Webservice im Terminal starten .....	22
Abb. 8.2: Kommunikation mit dem Webservice .....	23

## Tabellenverzeichnis

Tabelle 2.1: Vergleich der Modelle (vgl. [Ele151]) .....	2
Tabelle 3.1: Pin-Belegung der GPIO-Schnittstelle .....	5
Tabelle 3.2: Pin-Belegung für die Sonderfunktionen der GPIO-Schnittstelle .....	6
Tabelle 6.1: Dateien im Ordner „scripts“ .....	14
Tabelle 6.2: Aktionen der Sensoren .....	18
Tabelle 6.3: Beispiele für die Benutzung der zeitgesteuerten Aufzeichnung .....	19
Tabelle 8.1: Parameter für den Aufruf des Webservices .....	22

## Anhang

### ***Anhang A – Projektdateien***

- Projektdateien der Sensorstation
- Fritzing Projektdateien im Dateiformat „fzz“