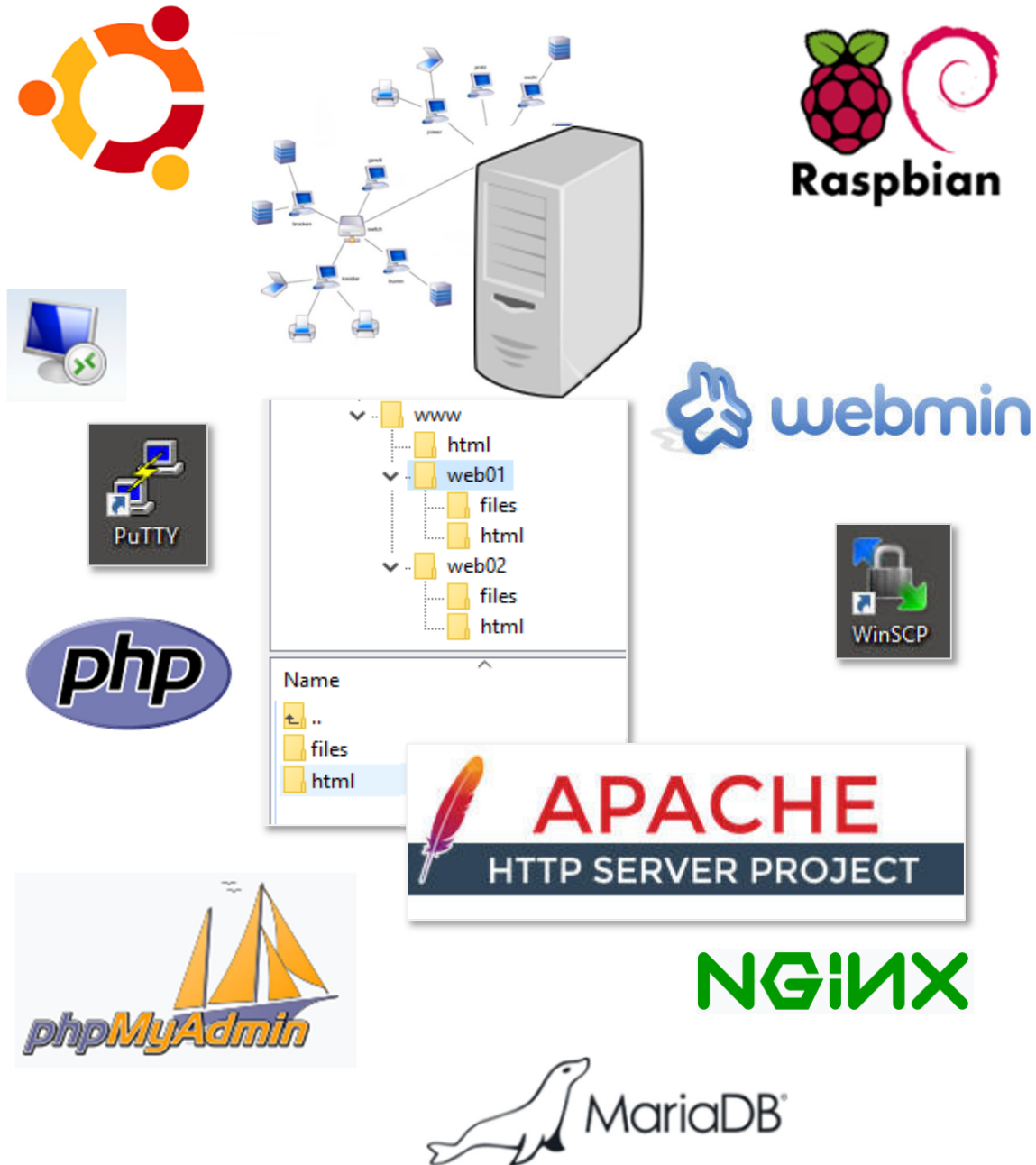


Raspberry Pi - Wetterstation

Gesamtprojekt Version3

darin enthalten sind Beispiele zur Linux-Installation und zu bash-Scripting, Remotezugriff, Linux-Serverkonfiguration, Datenbankmodellierung, Pythonscripting u.v.m.



Vorwort

Die vorliegende Dokumentation soll die Planung, Entwicklung und den Aufbau einer kleinen Wetterstation mit dem Kleincomputer Raspberry Pi Zero ausführlich darstellen. Sie erhebt jedoch keinen Anspruch auf Vollständigkeit. Eventuelle technische Änderungen können nicht laufend in der Dokumentation berücksichtigt werden.

Bei der Umsetzung des Projekts wurden sowohl auf der Hardware- wie auch auf der Softwareseite einige Versuchsinstallation durchgeföhrt, die aus mehreren Gründen wieder verworfen wurden. Die Beschreibung stützt sich auf Versionsstände ca. Sommer 2019.

Als im Juli 2019 das neue Raspberry-Image „Buster“ veröffentlicht wurde, konnte nochmals eine komplette Installation in kleinen Schritten nachvollzogen und hier endgültig dokumentiert werden.

Update_V2020.10:

Durch einen Defekt der SD-Karte wird nun die Wetterstation mit neuem Konzept aufgesetzt. Grund für den Defekt war vermutlich das sehr häufige Schreiben der Datendateien für die html-Anzeige und die Datenbank. Ein Backup der Daten soll nun regelmäßig über das Netzwerk an ein zuverlässigeres Speichermedium geschickt werden.

Inhalt

Abbildungsverzeichnis	IV
Verzeichnis der Programmcode-Ausschnitte	VI
Internetquellen	VI
Quellangabe der verwendeten Grafiken.....	VI
1 Das Projekt.....	1
2 Hardware.....	2
2.1 Raspberry Pi.....	2
2.2 Kameramodul.....	3
2.3 Temperatur- und Feuchtesensor	4
2.4 Vormontage.....	5
3 System-Installation	7
3.1 Download und Flashen.....	7
3.2 Vorbereitung des Remotezugriffs	8
3.3 Erster Startvorgang.....	9
3.4 Ersteinrichtung.....	10
4 Kamera- und Sensortest	14
4.1 Kamera.....	14
4.2 Temperatur und Feuchte-Sensor.....	15
5 Einbau des Luftdrucksensors	17
5.1 Allgemeines	17
5.2 Der BOSCH-Sensor BMP280	18
5.3 mechanischer Umbau	23
5.4 erster Funktionstest im Labor.....	24
5.5 Software zum Auslesen der BMP280-Sensordaten.....	25
6 Aufbau „vor Ort“	31
6.1 Montage	31
6.2 Verbindungstest.....	32
7 Serverinstallation.....	34
7.1 Vorbereitung	34
7.2 Datenbankserver MariaDB.....	35
7.3 Webserver Apache 2 installieren	38
7.4 Ordnerstruktur für den Webserver.....	39
7.5 PHP 7.....	40
7.6 Administrationssoftware für die Datenbank	42
8 Datenbank für Wetterdaten erstellen.....	47
8.1 Neue Datenbank anlegen.....	47
8.2 Tabelle erzeugen.....	47
8.3 Felder erstellen.....	48
8.4 Testdaten eingeben	48
8.5 User für den Datenbankzugriff	49
9 Serverüberblick	50
10 Messungen steuern	51
10.1 Planung	51
10.2 Python-Skripte programmieren.....	51
10.3 Wetterbild erstellen	55
11 Messdaten in die Datenbank ablegen	58
11.1 Zeitsteuerung der Messungen	62

12	Website vorbereiten.....	63
12.1	Planung	63
12.2	Programmierungsumgebung	64
12.3	Programmierung der Grundstruktur	65
13	Datenbankinfos aufbereiten und ausgeben.....	72
13.1	Sensordaten als Tabelle ausgeben	72
13.2	Grafische Wetterstatistik generieren.....	75
Anhang-.....		80
	Webseite – Übersicht	80

Abbildungsverzeichnis

Abbildung 1: Raspberry Pi Zero.....	2
Abbildung 2: Kamermodule-Datenblatt	3
Abbildung 3: Raspi-Kamera V2.1.....	3
Abbildung 4: Sensor DHT22	5
Abbildung 5: Datenblattauszug DHT22	5
Abbildung 6: Verdrahtungsplan Sensor	6
Abbildung 7: Sensorleitung.....	6
Abbildung 8: Sensormontage	6
Abbildung 9: Anschluss der Camera und Spannungsversorgung.....	6
Abbildung 10: Image flashen.....	7
Abbildung 11: Downloadseite (https://www.raspberrypi.org/downloads/raspbian/)	7
Abbildung 12: Image entpacken	7
Abbildung 13: Bootpartition	8
Abbildung 14: IP-Adresse ermitteln	9
Abbildung 15: feste IP-Adresse für den Raspberry Zero.....	9
Abbildung 16: erste Remoteverbindung.....	9
Abbildung 17: Anmeldung am System	10
Abbildung 18: Raspberry PI Software Configuration Tool	10
Abbildung 19: Lokale Einstellungen.....	10
Abbildung 20: raspi-config / Ländereinstellung.....	11
Abbildung 21: Dateisystem optimieren.....	11
Abbildung 22: Interface aktivieren	11
Abbildung 23: Neustart	11
Abbildung 24: Passwort für root	12
Abbildung 25: root-Anmeldung mit PuTTY und WinSCP	12
Abbildung 26: IP-Netzwerkconfiguration testen	13
Abbildung 27: Verbindungstest mit ping und traceroute.....	13
Abbildung 28: Systeminfos auslesen	13
Abbildung 29: Testbild der Raspberry Zero Kamera	14
Abbildung 30: System updaten	15
Abbildung 31: Adafruit-Bibliothek herunterladen.....	15
Abbildung 32: Messergebnisse für Temperatur und Luftfeuchtigkeit	16
Abbildung 33: Sensorabfrage	16
Abbildung 34: Datenblatt-Titel BMP280	18
Abbildung 35: BMP280 mit Trägerplatine.....	18
Abbildung 36: Blockschaltbild BMP280 (BOSCH Sensortec GmbH, 2018, S. 11).....	19
Abbildung 37: Messzyklus BMP280	19
Abbildung 38: Anschaltung der Sensorplatine	21
Abbildung 39: BMP280 Memory-Map (BOSCH Sensortec GmbH, 2018, S. 24)	21
Abbildung 40: I²C Schreibvorgang (BOSCH Sensortec GmbH, 2018, S. 29)	22
Abbildung 41: I²C Lesevorgang (BOSCH Sensortec GmbH, 2018, S. 30)	22

Abbildung 42: I ² C-Timing BMP280 (BOSCH Sensortec GmbH, 2018, S. 33).....	22
Abbildung 43: Leiterplatte für die Sensoranschaltung.....	23
Abbildung 44: Verdrahtung der Sensoren	23
Abbildung 45: Sensor- und Raspberry-Gehäuse.....	24
Abbildung 46: I ² C-Sensor ansprechen.....	24
Abbildung 47: I ² C-Dump	25
Abbildung 48: Controlregister.....	27
Abbildung 49: BMP280 Memory-Map Messregister (BOSCH Sensortec GmbH, 2018, S. 24).....	28
Abbildung 50: Entmontage der Wetterstation.....	31
Abbildung 51: Testbilder der Webcam.....	32
Abbildung 52: Bandbreitentest mit jperf.....	33
Abbildung 53: /etc/apt/sources.list erweitern	34
Abbildung 54: Systemupdate.....	34
Abbildung 55: Installationsablauf MariaDB.....	36
Abbildung 56: Anpassung MariaDB-login.....	37
Abbildung 57: Apache2-Startseite	38
Abbildung 58: Ordnerstruktur auf dem Webserver.....	39
Abbildung 59: php7.3 einrichten	40
Abbildung 60: php-Testseite (Code).....	40
Abbildung 61: php-Testseite (im Browser).....	41
Abbildung 62: Installationspfade für phpmyadmin.....	42
Abbildung 63: Anmelden an phpMyAdmin	43
Abbildung 64: root-Passwort in mysql setzen.....	43
Abbildung 65: phpMyAdmin updaten.....	44
Abbildung 66: Fehlermeldungen bei phpMyAdmin	45
Abbildung 67: Konfiguration des blowfish-Passworts.....	45
Abbildung 68: phpMyAdmin - Arbeitsbildschirm.....	46
Abbildung 69: Datenfelder anlegen	48
Abbildung 70: Datenbank-Designer	48
Abbildung 71: Daten einfügen.....	48
Abbildung 72: Datensätze anzeigen.....	48
Abbildung 73: aktuelle Serverversionen	50
Abbildung 74: Test-Wetterbild	56
Abbildung 75: Testskript für die Kamera	56
Abbildung 76: Kamertest mit Python-Skript.....	56
Abbildung 77: Integration in das Messskript.....	57
Abbildung 78: Messvorgang.....	58
Abbildung 79: Datenbankankbindung an ein Python-Script.....	59
Abbildung 80: Datenbankankbindung an ein Python-Script.....	60
Abbildung 81: Sensordaten in html-Datei	62
Abbildung 82: Datenbankeinträge.....	62
Abbildung 83: Seitenlayout - Planung.....	63
Abbildung 84: Programmierumgebung zur Webseitenentwicklung.....	64
Abbildung 85: Media-Query mit Firefox.....	67
Abbildung 86: Webseite bei kleiner Anzeigebreite.....	68
Abbildung 87: Menüleiste	69
Abbildung 88: Startseite im Browser	71
Abbildung 89: Messdaten in Tabellenansicht.....	72
Abbildung 90: jgraph im Webserverordner.....	75
Abbildung 91: Anzeige der Temperaturgrafiken	77

Verzeichnis der Programmcode-Ausschnitte



Programmcode 1: Kopfdaten des Python-Scripts	26
Programmcode 2: Kalibrierung BMP280.....	26
Programmcode 3: BMP280 - Konfigurationsdaten schreiben	27
Programmcode 4: BMP280 - Bitverschiebung.....	29
Programmcode 5: Berechnung der Messdaten	30
Programmcode 6: BMP280 - Messwerte ausgeben.....	30
Programmcode 7: Test-Messwerte in Datenbank ablegen.....	61
Programmcode 8: crontab-Konfiguration	62
Programmcode 9: index.php - Teil 1	65
Programmcode 10: index.php - Teil 2	66
Programmcode 11: index.php - Teil 3	66
Programmcode 12: css-Styledatei (Ausschnitt)	67
Programmcode 13: css-Stryledatei (Bereiche und Media-Queries)	68
Programmcode 14: Kopfbereich	69
Programmcode 15: Menü-dropdown	69
Programmcode 16: Menübereich 1	70
Programmcode 17: Menübereich 2	70
Programmcode 18: Menübereich 3	70
Programmcode 19: Logodatei	71
Programmcode 20: Inhaltsdatei beim Erstaufwurf	71
Programmcode 21: Datenbankzugriff Messdaten Zugang	73
Programmcode 22: Datenbankzugriff Messdaten SQL	73
Programmcode 23: Datenbankzugriff Messdaten Anzeige	74
Programmcode 24: Datenbankverbindung schließen.....	74
Programmcode 25: Hauptdatei für die Grafikanzeige	77
Programmcode 26: Untermenü der Grafikanzeige	77
Programmcode 27: SQL-String für die Grafikanzeige	78
Programmcode 28: Anzeigeskript	78
Programmcode 29: fertige Website (Stand 16.12.2020)	80
Programmcode 30: grafischer Temperaturverlauf auf der Webseite	80










Internetquellen

Grafik GPIOs

<https://indibit.de/raspberry-pi-die-gpio-schnittstelle-grundlagenbelegung/>

Quellangabe der verwendeten Grafiken

	<p>Das Logo des Linux-Tux wurde freigegeben von Larry Ewing unter folgenden Bedingungen: „Es ist erlaubt, diese Grafik zu verwenden und/oder zu verändern. Bedingung ist jedoch: Falls jemand fragt, muss man mich – lewing@isc.tamu.edu – als Urheber nennen und auf GIMP hinweisen.“ (Ewing, Budig, Gerwinsky, & Hagel, 2008)</p>
<p>Linux-UBUNTU</p> 	<p>Grafik gemeinfrei Datei: https://de.wikipedia.org/wiki/Ubuntu#/media/File:Ubuntu_logo.svg</p>

<p>Webserver Apache</p> 	<p>Datei: https://de.wikipedia.org/wiki/Apache_Software_Foundation#/media/File:Apache_Software_Foundation_Logo_(2016).svg</p> <p>Licensed under the Apache License, Version 2.0 (the "License");you may not use this file except in compliance with the License.You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.</p>
<p>PHP Skriptsprache</p>  <p>Datenbankserver</p>  <p>DB-Admin</p> 	<p>Datei php: https://de.wikipedia.org/wiki/PHP#/media/File:PHP-logo.svg</p> <p>Datei DB: https://de.wikipedia.org/wiki/MariaDB#/media/File:MariaDB_Logo.png</p> <p>Datei : https://de.wikipedia.org/wiki/PhpMyAdmin#/media/File:PhpMyAdmin-Logo.svg</p> <p>Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) (Creative Commons PoBox 1866, Mountain View, CA 94042)</p> <p>You are free to: Share — copy and redistribute the material in any medium or format Adapt — remix, transform, and build upon the material for any purpose, even commercially. This license is acceptable for Free Cultural Works.</p> <p>The licensor cannot revoke these freedoms as long as you follow the license terms. Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.</p> <p>ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.</p> <p>No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.</p> <p>https://creativecommons.org/licenses/by-sa/4.0/legalcode</p>
<p>WinSCP</p> 	<p>Datei: https://de.wikipedia.org/wiki/WinSCP#/media/File:WinSCP_Logo.png</p> <p>Lizenz: https://www.gnu.org/licenses/gpl-3.0.html</p>
<p>PuTTY</p> 	<p>Datei: https://de.wikipedia.org/wiki/PuTTY#/media/File:PuTTY_icon_128px.png</p> <p>Lizenz: https://opensource.org/licenses/mit-license.php</p>
<p>RaspBerryPi</p> 	<p>Lizenzen:</p> <p>https://www.raspberrypi.org/trademark-rules/</p> <p>https://www.raspberrypi.org/</p>
<p>WebAdmin</p> 	<p>Datei: http://www.webmin.com/graphics/webmin-full.jpg</p> <p>Lizenz: http://www.webmin.com/graphics.html</p>
	<p>Grafik gemeinfrei</p> <p>Datei: https://upload.wikimedia.org/wikipedia/commons/c/c5/Nginx_logo.svg</p> <p>Autor: https://commons.wikimedia.org/w/index.php?curid=24774395</p>

Weitere Abbildungen (Bilder, Grafiken) wurden selbst angefertigt und sind mit einer Fußnote gekennzeichnet.

1 Das Projekt

Ziel des Projekts soll zunächst sein, eine praktische Anwendung zur intensiven Einarbeitung in die Möglichkeiten des Raspberry Pi zu erhalten. Hierbei soll zunächst die schrittweise Beschreibung der Inbetriebnahme von Hardware und Software stehen.

Sollte die Wetterstation brauchbare Daten liefern, ist eine Veröffentlichung auf der Raspberry-Webseite im Internet geplant.

Update_V2020.10:

Nach knapp 1 Jahr Betriebsdauer hat leider die SD-Karte ihre Funktion aufgegeben, vermutlich durch die vielen Schreibzugriffe beim Speichern der Daten. Da die Wetterstation etwas unzugänglich montiert war, wurde längere Zeit kein Image des Systems erstellt. Aber egal: Nun wird die Gelegenheit ergriffen, die ganze Installation nochmals auf einer neuen Karte durchzuspielen. Dabei kann auch gecheckt werden, ob die Doku an allen Stellen korrekt ist.

Außerdem soll die Komplettinstallation bis zur Datenbankanbindung in einem Dokument beschrieben werden.

2 Hardware

Mit einem Kleincomputer Raspberry Pi Zero W (siehe Abbildung 1) soll als Wetterstation eingerichtet werden. Dabei sollen auf einer Webseite neben aktuellen Wetterbildern auch eine Wetter-Historie angezeigt werden, die neben Wetterbildern auch Temperaturdaten enthalten muss. Die Einzelkomponenten bestehen aus Raspberry-Pi-Zero-Hauptplatine, Raspberry Pi Kameramodul und diversen Sensoren. An die Hauptplatine des Raspberry Pi Zero soll im Erstausbau lediglich ein Temperatur-/Feuchtigkeitssensor, sowie die Kamera V 2.1 angeschaltet werden.

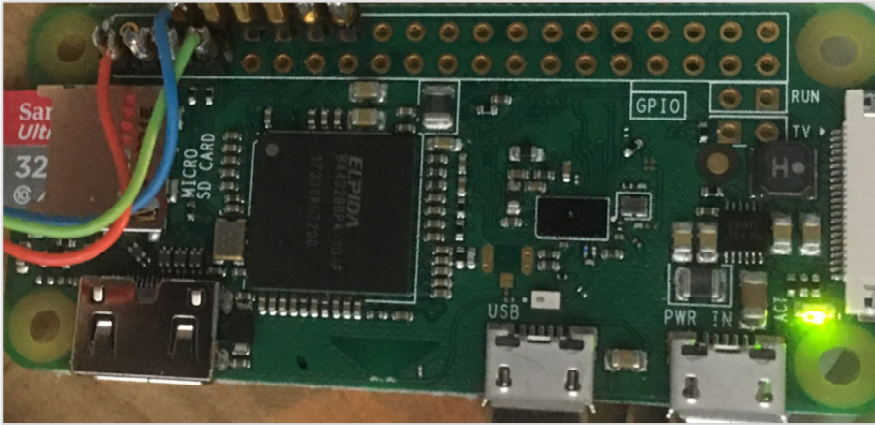


Abbildung 1: Raspberry Pi Zero

2.1 Raspberry Pi

Der Raspberry Pi ist ein Einplatinencomputer und wurde von der britischen Raspberry Pi Foundation entwickelt. Er enthält ein Ein-Chip-System von Broadcom mit einem ARM-Mikroprozessor und kam 2012 auf den Markt. Entwickelt wurde er mit dem Ziel, jungen Menschen Hardwarekenntnisse und Softwareprogrammierung einfach nahezubringen.

Der Name ist das englische Wort für Himbeerkuchen. Damit wird in der Tradition von Apple oder Acorn verfahren, die Computer nach Früchten benannt haben. Die Erweiterung „Pi“ soll für Python-Interpreter stehen, da ursprünglich gedacht war, einen fest eingebauten Interpreter für die Programmiersprache Python mitzuliefern. Das Logo des Projekts zeigt eine stilisierte Himbeere. Das Modell Zero W, das für dieses Projekt eingesetzt werden soll, wurde Anfang 2017 veröffentlicht und besitzt folgende Eigenschaften:

- CPU: ARM 11 mit 1 Kern
- Taktfrequenz: 1000 MHz
- Architektur ARMv6
- Grafik: Broadcom Full HD 1080p30 400 MHz
- Arbeitsspeicher (RAM): 512 MB
- Speicherkarten: microSDb
- Video: Mini HDMI Typ C
- Audio: HDMI

- Netzwerk: WLAN Broadcom 2,4 GHz nach IEEE 802.11 b/g/n
- Pins: 40, davon 26 GPIO
- weitere Schnittstellen: CSI, I²C
- Leistung (Stromaufnahme): 0,5-0,7 W / 100-140 mA
- Betriebsspannung: 5 V über Micro-USB-B

2.2 Kameramodul

Die Kamera soll als „Wettercam“ eingesetzt werden und wird an den Steckplatz auf der Raspi-Platine angeschlossen. Als Kameramodul wird die Raspberry Pi Camera V.2.1 angeschlossen. Es handelt sich um eine HD-Camera 1080p mit einer Auflösung von 8 Megapixel (siehe Abbildung 3). Das Hersteller-Datenblatt zeigt die technischen Daten des Moduls (siehe Abbildung 2).


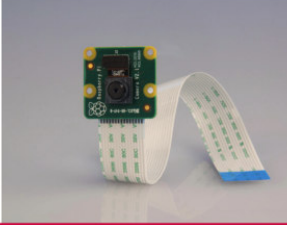


 Raspberry Pi	
	
Camera Module	
Product Name	Raspberry Pi Camera Module
Product Description	High Definition camera module compatible with all Raspberry Pi models. Provides high sensitivity, low crosstalk and low noise image capture in an ultra small and lightweight design. The camera module connects to the Raspberry Pi board via the CSI connector designed specifically for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the processor.
RS Part Number	913-2664
Specifications	
Image Sensor	Sony IMX 219 PQ CMOS image sensor in a fixed-focus module.
Resolution	8-megapixel
Still picture resolution	3280 x 2464
Max image transfer rate	1080p: 30fps (encode and decode) 720p: 60fps
Connection to Raspberry Pi	15-pin ribbon cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI-2).
Image control functions	Automatic exposure control Automatic white balance Automatic band filter Automatic 50/60 Hz luminance detection Automatic black level calibration
Temp range	Operating: -20° to 60° Stable image: -20° to 60°
Lens size	1/4"
Dimensions	23.86 x 25 x 9mm
Weight	3g
www.rs-online.com/raspberrypi  	

Abbildung 2: Kameramodul-Datenblatt¹

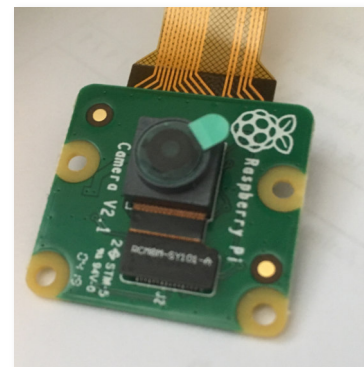


Abbildung 3: Raspi-Kamera V2.12

¹ Datenblattquelle: <http://www.rs-online.com/raspberrypi>

² Bild: HK 2019

2.3 Temperatur- und Feuchtesensor

Der Sensor muss verdrahtet und dann durch Installation weiterer Pakete in Betrieb genommen werden. Die Messdaten sollen mit Hilfe der Programmiersprache Python ausgelesen und verarbeitet werden. Der Raspberry Pi Zero unterstützt verschiedene Kommunikationsprotokolle der unterschiedlichen Hardware-Schnittstellen UART (serielle Schnittstelle), SPI, I2C. Das Verfahren **Dallas 1-Wire** nutzt die gleiche Leitung für Spannungsversorgung und Daten, daher wird dies auch **1-Wire-Protokoll** genannt.

Zur Ermittlung von Temperatur und Luftfeuchtigkeit soll der Sensor DHT22 eingesetzt werden. Der Sensor besitzt eine **1-wire-Schnittstelle** und kann somit mit drei Anschlussleitungen an die Raspi-Leiterplatte angeschlossen werden.

Sensor-Grundlagen

Laut Datenblatt des Sensorherstellers wird beim DHT22 ein ähnliches, allerdings nicht 1-Wire kompatibles Protokoll genutzt. Aus diesem Grund benötigt man einen speziellen Treiber. Es wird vom PI aus per Software gesteuert. Als Nachteil ergibt sich die etwas anfälligere Kommunikation gegenüber Störungen und sie ist softwareseitig etwas aufwändiger, aber dafür als Vorteil deutlich flexibler in der Anwendung.

Ein Pullup-Widerstand zieht den Input des DHT22 normalerweise auf V_{CC} (+3,3V). Dadurch ist der Ruhepegel auf der Datenleitung HIGH-Signal. Zum Starten der Kommunikation mit dem Sensor setzt der Raspberry Pi den Pin für eine bestimmte Zeit auf LOW (0 V), anschließend wieder auf HIGH. Der Sensor wacht aus seinem Stromsparmodus auf, und führt eine Messung durch. Anschließend sendet er 40 Bit Daten, in denen die Luftfeuchtigkeit und die Temperatur als Code enthalten sind und geht anschließend wieder in den Schlafmodus.

Beim Übertragen der Daten wechselt der Sensor den Logikzustand der Datenleitung periodisch zwischen HIGH und LOW. Welche Daten übertragen werden, ist durch die Länge der Periode, in der der Sensor HIGH sendet, definiert:

- Ist das HIGH-Signal 26-28 ms lang, dann entspricht dies einem Datenbit 0
- Ist das HIGH-Signal 70 ms lang, entspricht dies einem Datenbit 1

Durch diese Zeitabhängigkeit ist ein präzises Timing sehr wichtig, um diesen Sensor auszulesen. Dies kann nicht direkt mit Hilfe „langsamer“ Programmiersprachen, wie z.B. Python oder Java erfolgen. Diese Sprachen werden nicht nahe genug an der CPU ausgeführt. Es wird kompilierter C Code benötigt.

Der Sensor DHT22 sollte **nicht häufiger als alle drei Sekunden** abgefragt werden.

Ein Open Source Treiber für die Sensoren stammt von der Firma Adafruit³. Wie oben bereits erwähnt ist der Grundcode in C geschrieben, da der Sensor sehr genaues Timing erfordert und das Timing in Software durchgeführt werden muss. Adafruit stellt auf Basis dieses C Codes eine Python-Schnittstelle zur Verfügung, über die der Sensor in eigenen Anwendungen eingebunden werden kann.

³ Adafruit Industries – Firma für Open-Source-Hardware / New York

Datenblattauszug DHT22⁴

Aosong Electronics Co.,Ltd,

Your specialist in innovating humidity & temperature sensors

Feature & Application

Full range temperature compensated

Relative humidity and temperature measurement

Calibrated digital signal , Outstanding long-term stability

Extra components not needed. Long transmission distance

Low power consumption

4 pins packaged and fully interchangeable



Abbildung 4: Sensor DHT225

Description

DHT22 output calibrated digital signal. It utilizes exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability. Its sensing elements is connected with 8-bit single-chip computer.

Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory.

Small size & low consumption & long transmission distance(20m) enable DHT22 to be suited in all kinds of harsh application occasions.

Single-row packaged with four pins, making the connection very convenient.

Technical Specification

Model	DHT22
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +-2%RH(Max +-5%RH); temperature <+-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Sensing period	Average: 2s
Interchangeability	fully interchangeable
Dimensions	small size 14*18*5.5mm; big size 22*28*5mm

Abbildung 5: Datenblattauszug DHT22

2.4 Vormontage

Der Aufbau der Hardware erfolgt zunächst im Labor. Erst nach Inbetriebnahme der Grundversion soll die Montage an einer außenliegenden Haus-Giebelwand erfolgen. Damit die Leiterplatte des Raspberry Zero W zur Montage im Freien geschützt ist, wird ein OBO-Verteiler mit IP55- Spritzwasserschutz verwendet. Eine Litze-Anschlussleitung mit Abschirmung wird zur Verbindung des Raspberry Pi mit dem unterhalb der Verteilerdose angebrachten Sensors genutzt.

⁴ Quelle: <https://www.mikrocontroller-elektronik.de/?projekt-download=442>

⁵ Bild: HK 2019

Die Kamera soll von innen durch eine Öffnung in der Dose die Wetterbilder aufnehmen. Zur Spannungsversorgung vom Steckernetzteil wird die Leitung mit Stecker ebenfalls durch die Öffnung der Verteilerdose geführt und mit der Raspberry-Pi-Leiterplatte verbunden.

Verdrahtung des Sensors

Der Sensor wird nun an das wasserfeste Aufputzgehäuse des Raspberry PI Zero angebaut. Er soll die Daten im Freien und nicht innerhalb des Schutzgehäuses aufnehmen. Da der Sensor lediglich eine einzige Datenleitung besitzt, reicht mit VDD (3,3 V) und GND (0 V) eine dreiadrige Leitung aus.

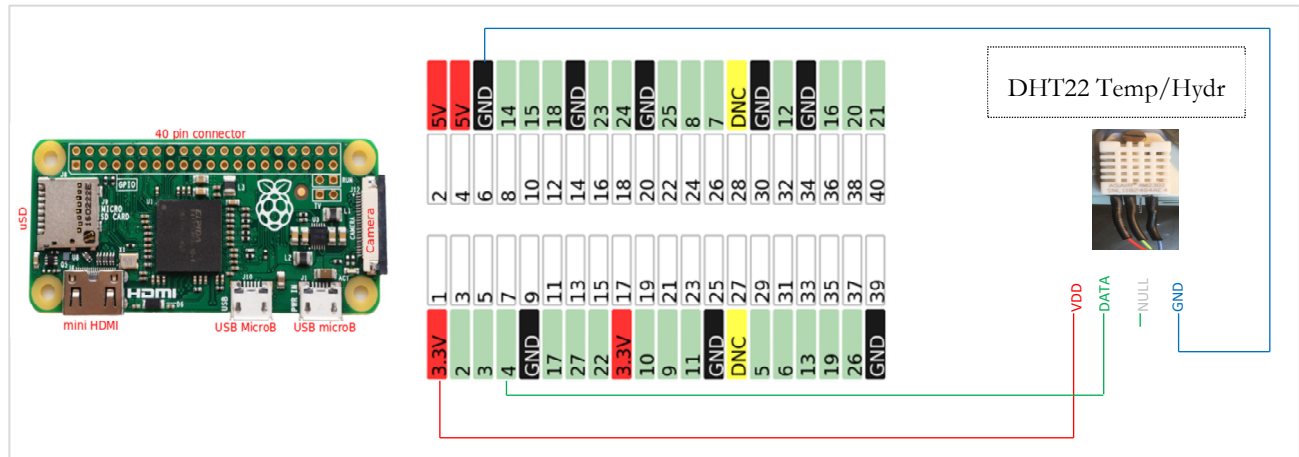


Abbildung 6: Verdrahtungsplan Sensor

Die Adern werden abisoliert und verzinnt. Ein Schrumpfschlauch soll das spätere Eindringen von Feuchtigkeit an die Lötstellen und die Berührung der Einzeldrähte verhindern. Auch bei der Einführung der Leitung in die Anschlussdose wird Schrumpfschlauch verwendet. Später wird diese Stelle noch durch Silikonkleber abgedichtet. Nun wird noch die USB-Anschlussleitung vom Netzgerät eingeführt, um erste Funktionstests durchzuführen. Zur Endmontage wird diese später nochmals entfernt und durch einen Wanddurchbruch gesteckt.

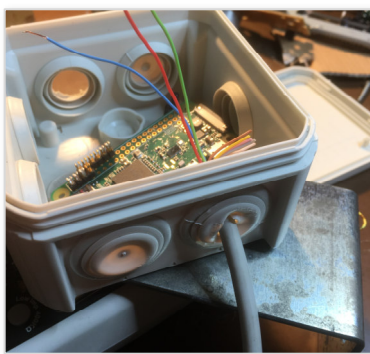


Abbildung 7: Sensorleitung⁶

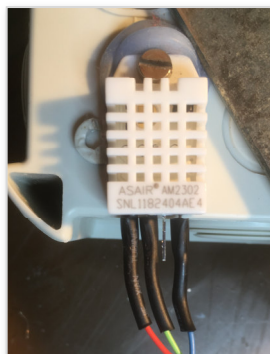


Abbildung 8: Sensormontage



Abbildung 9: Anschluss der Camera und Spannungsversorgung

⁶ Bilder: HK 2019

3 System-Installation

Als Betriebssystem für den Raspberry Pi Zero W wird Raspian Buster) mit dem Versionsstand 2019-07-10 (Juli 2019) auf der SD-Karte installiert. Nach dem Download erfolgt das Flashen der SD-Karte und die Erstkonfiguration.

3.1 Download und Flashen

Der Download des aktuellen Betriebssystems ist z.B. unter <https://www.raspberrypi.org/downloads/raspbian/> erhältlich.

Folgende Arbeitsschritte werden ausgeführt:

- Image-Download vom Windows-Rechner als **zip**-Datei.
- Image entpacken und **img**-Datei auf dem Desktop ablegen.
- Flashen des Images mit dem Tool Etcher auf die 32 GByte SD-Karte.

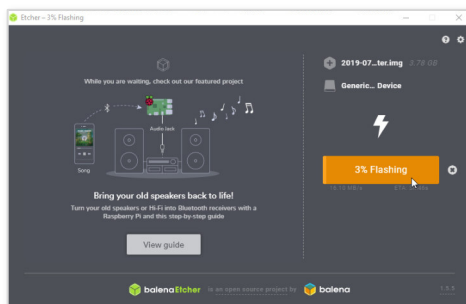
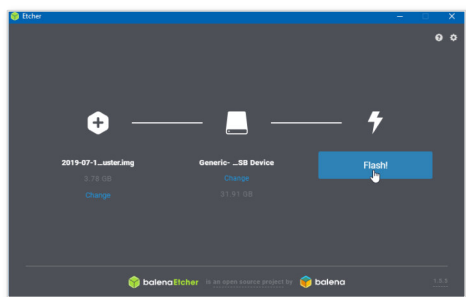


Abbildung 10: Image flashen

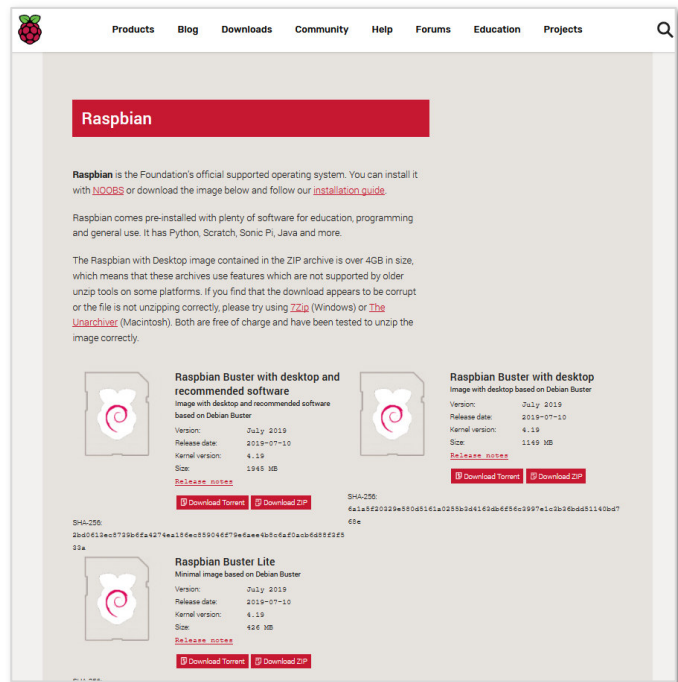


Abbildung 11: Downloadseite (<https://www.raspberrypi.org/downloads/raspbian/>)

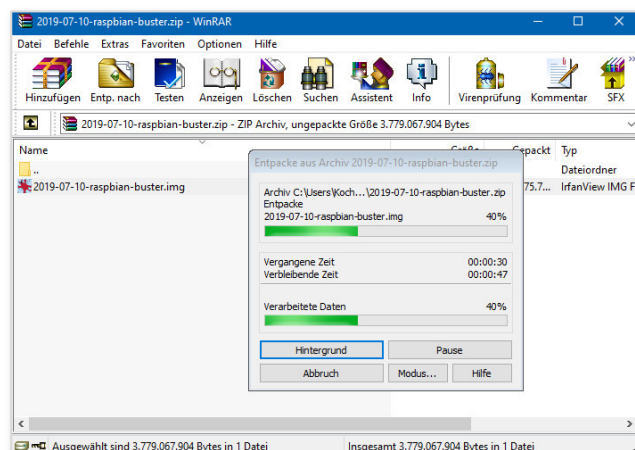


Abbildung 12: Image entpacken

Update_V2020.10:

Bei der Neuinstallation wurde das Image „Raspbian Buster Light“ (ohne Desktop) verwendet.

3.2 Vorbereitung des Remotezugriffs

Damit der Raspberry Zero über das Netzwerk erreicht wird, benötigt das System die Aktivierung des SSH-Protokolls. Außerdem muss die WLAN-Verbindung korrekt eingestellt sein. Hierzu müssen zwei Dateien in der boot-Partition der SD-Karte erstellt werden

SSH aktivieren

Zur Aktivierung des SSH-Protokolls wird auf der obersten Ebene der SD-Bootpartition eine leere Datei mit dem Namen **ssh** (!ohne Dateierweiterung!) erstellt.

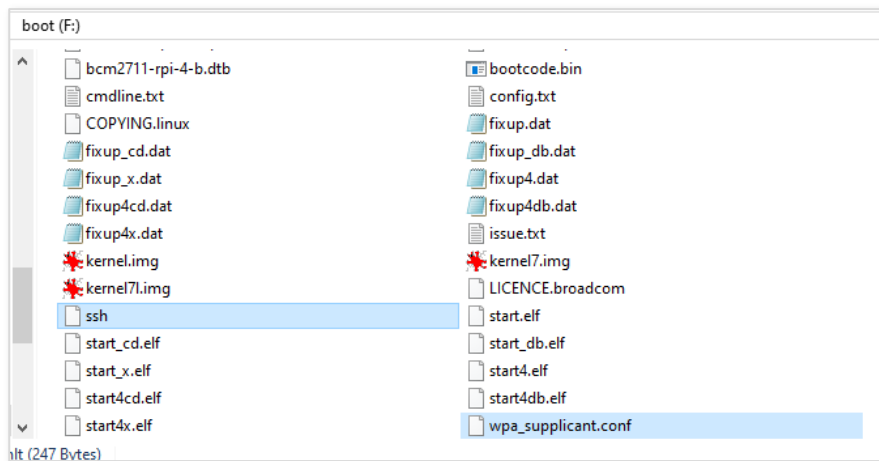


Abbildung 13: Bootpartition

WLAN konfigurieren

Damit sich der Raspberry Zero beim Booten mit dem WLAN verbinden kann, wird eine zusätzliche Datei mit dem Namen **wpa_supplicant.conf** (ebenfalls auf der Bootpartition) benötigt. Die Datei enthält neben der ssid auch das Passwort für das Netzwerk.

```
# Datei wpa_supplicant.conf in der Boot-Partition (Raspbian Buster)
country=DE
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    ssid="SSID des Netzes"
    psk="Wlan Passwort des Netzes"
    key_mgmt=WPA-PSK
}
```

3.3 Erster Startvorgang

Für den ersten Start wird SD-Karte vom Windows-PC in den Raspberry Zero gesteckt und die Spannungsversorgung angeschaltet. Der PI bootet und meldet sich „hoffentlich“ am Netzwerk per WLAN an. Um die IP-Adresse zu ermitteln bietet sich verschiedene Tools an, z.B. **PingInfoView**, mit dem die aktiven Adressen eines LAN-Segments angezeigt werden.

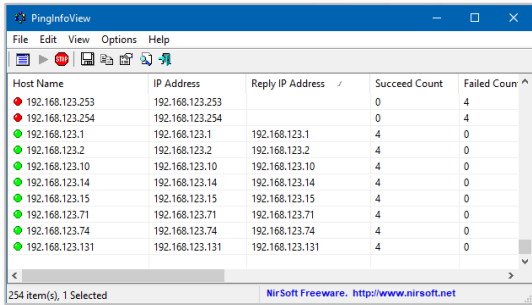


Abbildung 14: IP-Adresse ermitteln

Eine weitere Möglichkeit ist, im Protokoll des DHCP-Servers (z.B. im LAN-Router) nachzusehen, welche Adresse der neue Host **raspberrypi** erhalten hat. An dieser Stelle ist es meist auch möglich, dem Raspberry Zero eine feste, sich nicht mehr ändernde IP-Adresse zu geben. Im Beispiel der Fritz!Box 7390 ist dies in dargestellt.

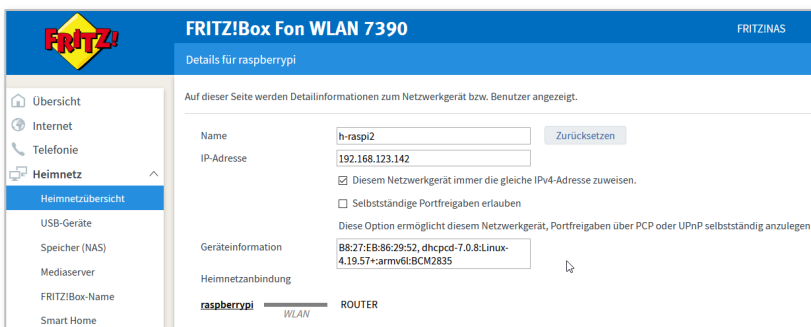


Abbildung 15: feste IP-Adresse für den Raspberry Zero

Nun ist der Raspberry Zero gestartet und eine Anmeldung per SSH-Remotezugriff kann getestet werden. In PuTTY wird die ermittelte IP-Adresse eingetragen, der Zeichensatz auf UTF-8 gestellt und die Verbindung gestartet. Beim ersten Verbinden erfolgt ein Schlüsselaustausch zwischen Host und Remote-PC.

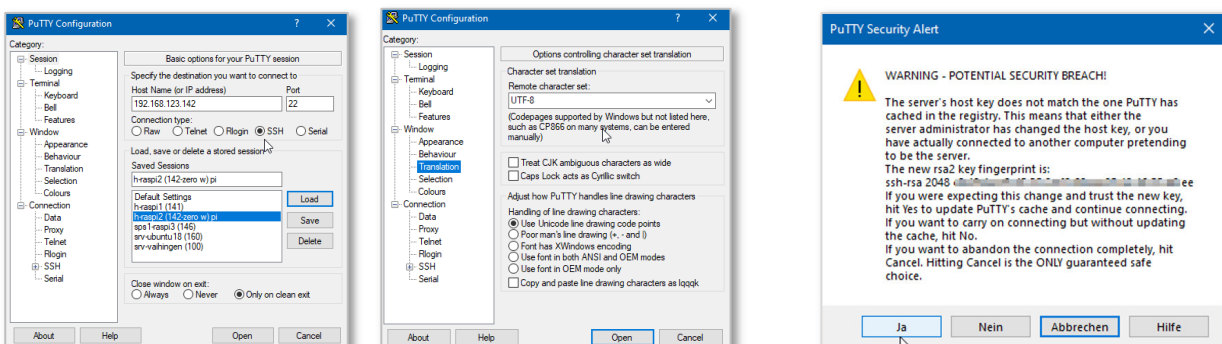


Abbildung 16: erste Remoteverbindung

3.4 Ersteinrichtung

Die Anmeldung erfolgt beim ersten Mal mit dem Benutzer **pi** und dem Passwort **raspberrypi**.

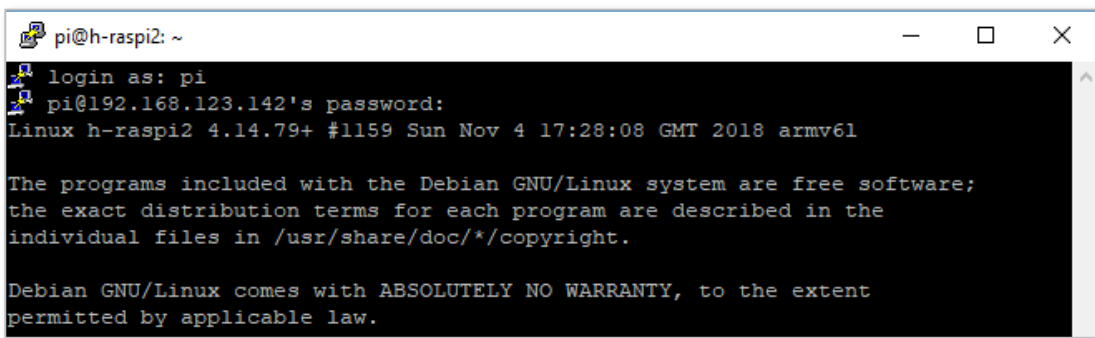


Abbildung 17: Anmeldung am System

Nun kann das System über die Kommandozeile eingerichtet werden. Nach der erfolgreichen Anmeldung wird das Konfigurationsprogramm aufgerufen.

\$ sudo raspi-config

In verschiedenen Untermenüs müssen Anpassungen vorgenommen werden. Dies ist aus den folgenden Abbildungen und den Kurzbeschreibungen ersichtlich.

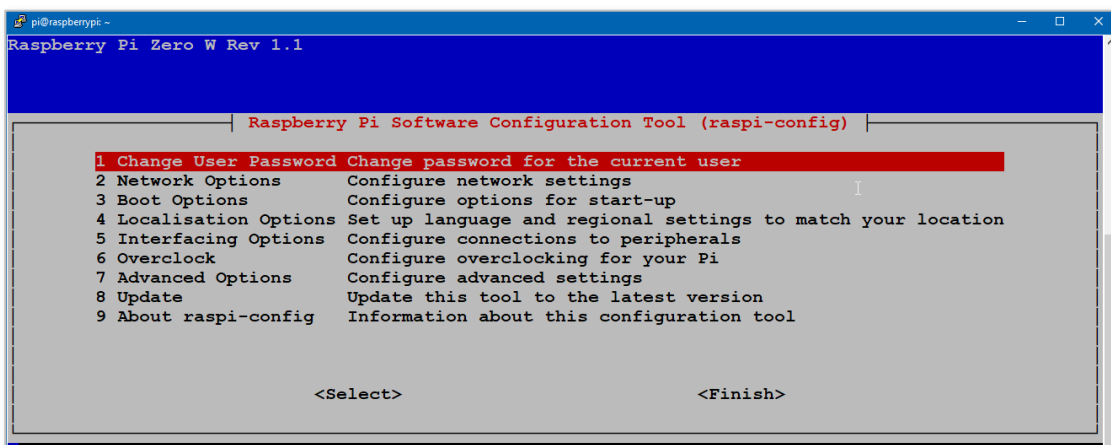


Abbildung 18 Raspberry Pi Software Configuration Tool

Passwort für aktuellen User **pi**.

Das Passwort für **pi** sollte aus Sicherheitsgründen als erste Maßnahme geändert werden.

Network Options

Die Netzwerkoptionen können in einem weiteren Untermenü den lokalen Gegebenheiten angepasst werden.

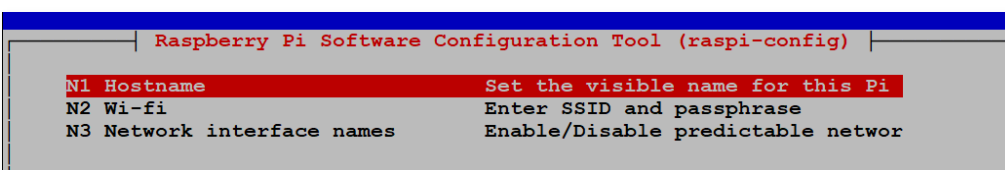


Abbildung 19: Lokale Einstellungen

- Hostname ändern, hier im Beispiel: **h-raspi2**
- Network-Interface Names – neues Verfahren aktivieren

Localisation Options (Regionale Einstellungen)

Diese Einstellungen sollten als erstes durchgeführt werden, um die Tastatur auf deutschen Zeichensatz umzustellen.

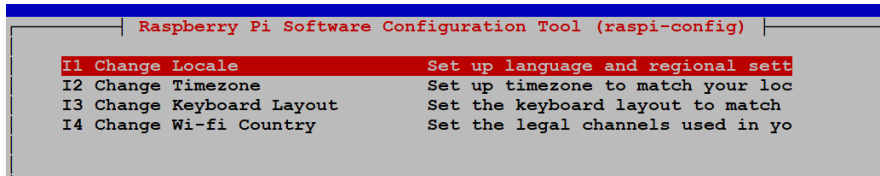


Abbildung 20: raspi-config / Ländereinstellung

- Advanced: Dateisystem SD-Karte - Das Dateisystem wird auf die gesamte SD-Karte optimiert.

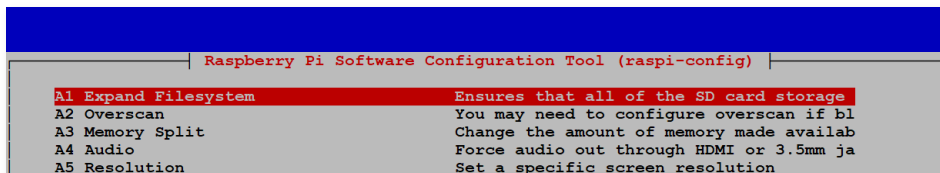


Abbildung 21: Dateisystem optimieren

Peripherals: Interface-Options

SSH-Zugang (falls noch nicht geschehen durch SSH-Boot-Datei) und das Kameramodul werden aktiviert. Außerdem wird aktiviert:

- Remote GPIO
- I²C – Bustreiber zur Ansteuerung der externen Sensoren
- **Update_V2020.10:** zusätzlich 1-Wire Interface aktivieren

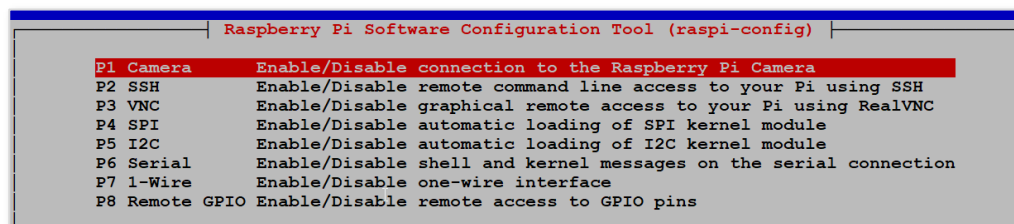


Abbildung 22: Interface aktivieren

Änderungen übernehmen und neustarten

Damit alles aktiviert wird, ist ein Neustart notwendig.

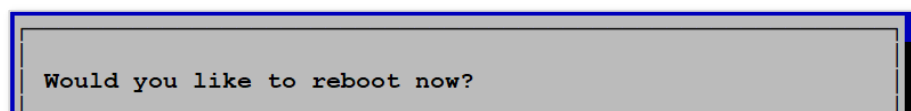


Abbildung 23: Neustart

Remoteanmeldung auch mit root

Damit auch mit dem Benutzer **root** eine SSH-Anmeldung erfolgen kann, muss eine Änderung in der **sshd_config**-Datei gemacht werden.

Vorher wird das Passwort vergeben.

```
pi@h-raspi2:~ $ sudo passwd root
Geben Sie ein neues UNIX-Passwort ein:
Geben Sie das neue UNIX-Passwort erneut ein:
passwd: Passwort erfolgreich geändert
pi@h-raspi2:~ $
```

Abbildung 24: Passwort für root

Die Änderungen in der **sshd_config** betreffen die Zeile **PermitRootLogin**. Die Zeile kann in **nano** mit **Strg+W** gesucht werden.

```
pi@h-raspi2: ~
root@h-raspi2:~# nano /etc/ssh/sshd_config
GNU nano 2.7.4 Datei: /etc/ssh/sshd_config

#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin yes
#StrictModes yes
#MaxAuthTries 6
```

Nach dem Speichern der Datei wird der Dienst neu gestartet.

```
$ sudo service ssh restart
```

Nach dem Neustart des Dienstes ist eine Anmeldung mit z.B. WinSCP oder PuTTY möglich:

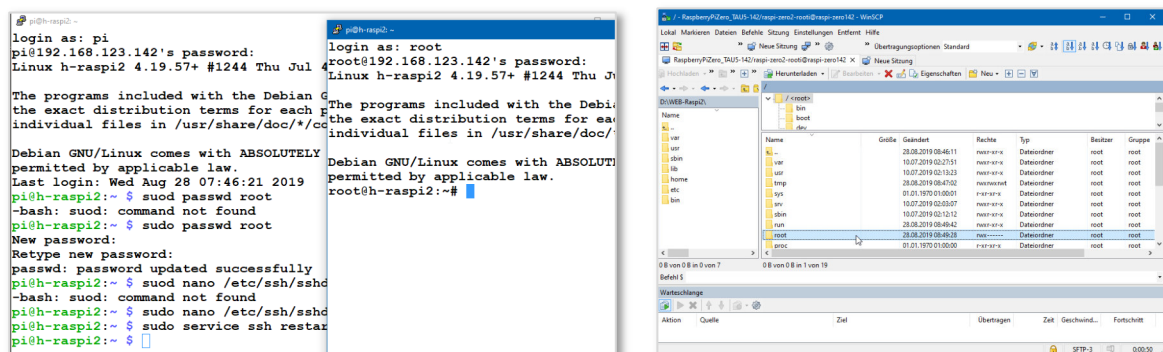


Abbildung 25: root-Anmeldung mit PuTTY und WinSCP

ACHTUNG: Die Anmeldung mit dem Systemverwalter **root** sollte aus Sicherheitsgründen nur in Ausnahmefällen vorgenommen werden.

Konfigurationstest

Nach erfolgter Remoteanmeldung werden wichtige Dienste getestet und Parameter ausgelesen. Das Auslesen der Netzwerkeinstellungen erfolgt nach neuer Methode mit **ip address** (alt: **ifconfig**).

```
pi@h-raspi2:~ $ ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
    link/ether b8:27:eb:86:29:52 brd ff:ff:ff:ff:ff:ff
    inet 192.168.123.142/24 brd 192.168.123.255 scope global noprefixroute
        valid_lft forever preferred_lft forever
    inet6 fe80::712b:5910:e221:bf78/64 scope link
        valid_lft forever preferred_lft forever
pi@h-raspi2:~ $
```

Abbildung 26: IP-Netzwerkconfiguration testen

Der Netzwerkverbindungstest wird über **ping** und **traceroute** durchgeführt.

```
pi@h-raspi2:/proc
pi@h-raspi2:/proc $ ping www.google.de
PING www.google.de (172.217.17.67) 56(84) bytes of data.
64 bytes from ams16s30-in-f3.1e100.net (172.217.17.67): icmp_seq=1 ttl=57 time=25.6 ms
64 bytes from ams16s30-in-f3.1e100.net (172.217.17.67): icmp_seq=2 ttl=57 time=42.5 ms

pi@h-raspi2:/proc
pi@h-raspi2:/proc $ traceroute www.google.de
traceroute to www.google.de (172.217.168.195), 30 hops max, 60 byte packets
 1 fritz.box (192.168.123.1) 4.656 ms 5.361 ms 5.926 ms
 2 * 62.155.245.70 (62.155.245.70) 33.927 ms 34.412 ms
 3 217.239.52.30 (217.239.52.30) 33.829 ms 35.517 ms 35.755 ms
 4 80.157.207.46 (80.157.207.46) 35.540 ms 35.318 ms 34.495 ms
 5 108.170.247.115 (108.170.247.115) 34.271 ms 108.170.247.99 (108.170.247.99)
```

Abbildung 27: Verbindungstest mit ping und traceroute

Updaten des Systems

Damit für die folgenden Einstellungen ein aktuelles System zur Verfügung steht werden die Kommandos zum Updaten und Upgraden angewendet:

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Abschließend werden die wichtigsten Systemdaten zur Dokumentation ausgelesen.

```
pi@h-raspi2:~ $ cat /sys/firmware/devicetree/base/model
Raspberry Pi Zero W Rev 1.1pi@h-raspi2:~ $
pi@h-raspi2:~ $ cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 10 (buster)"
NAME="Raspbian GNU/Linux"
VERSION_ID="10"
VERSION="10 (buster)"
VERSION_CODENAME=buster
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"
pi@h-raspi2:~ $
```

Abbildung 28: Systeminfos auslesen

4 Kamera- und Sensortest

Die Kamera ist an die interne RaspberryPI – Schnittstelle angeschlossen. In mehreren Schritten wird die Funktion der Kamera nun geprüft.

4.1 Kamera

Zunächst wird über den PuTTY-Remotezugang mit dem bereits installierten Programm **raspistill** die Funktion der Kamera getestet. Das erzeugte Bild kann mit WinSCP auf den lokalen Windowsrechner heruntergeladen und angesehen werden.

```
$ raspistill -o /home/pi/testbild.jpg
```

Die Position der Kamera im Labor ist noch nicht an der endgültigen Stelle und zeigt im Moment den Himmel über Weinstadt. Am gesamten System sollen zunächst weitere Anpassungen und Erweiterungen vorgenommen werden, bevor die endgültige Montage und Ausrichtung der Kamera erfolgt.



Abbildung 29: Testbild der Raspberry Zero Kamera

4.2 Temperatur und Feuchte-Sensor

Zur komfortablen Ansteuerung des Sensors wird das Programmpaket der Firma Adafruit benötigt. Diese Bibliotheken werden heruntergeladen und anhand eines mitgelieferten Beispielskripts getestet.

Linux-System auf den neusten Stand bringen

Damit sich alle Softwarepakete auf dem neuesten Stand befinden, wird ein Systemupdate durchgeführt.

```
$ sudo apt-get update
$ sudo apt-get install build-essential python-dev python-openssl
```

```
pi@h-raspi2:~ $ sudo apt-get update
Hit:1 http://archive.raspberrypi.org/debian buster InRelease
Get:2 http://raspbian.raspberrypi.org/raspbian buster InRelease [15.0 kB]
Fetched 15.0 kB in 2s (6,200 B/s)
Reading package lists... Done
pi@h-raspi2:~ $ sudo apt-get install build-essential python-dev python-openssl
Reading package lists... Done
Building dependency tree
Reading state information... Done
build-essential is already the newest version (12.6).
python-dev is already the newest version (2.7.16-1).
python-dev set to manually installed.
python-openssl is already the newest version (19.0.0-1).
python-openssl set to manually installed.
The following package was automatically installed and is no longer required:
  rpi.gpio-common
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 2 not upgraded.
pi@h-raspi2:~ $
```

Abbildung 30: System updaten

Adafruit Bibliotheken herunterladen und installieren

Die Bibliotheken des Herstellers müssen installiert werden. Mit der Adafruit Bibliothek ist die Ansteuerung des DHT-Sensors einfach durchzuführen.

```
$ sudo apt-get install git
$ git clone https://github.com/adafruit/Adafruit_Python_DHT.git
```

```
pi@h-raspi2:~ $ git clone https://github.com/adafruit/Adafruit_Python_DHT.git
Cloning into 'Adafruit_Python_DHT'...
remote: Enumerating objects: 317, done.
remote: Total 317 (delta 0), reused 0 (delta 0), pack-reused 317
Receiving objects: 100% (317/317), 95.66 KiB | 382.00 KiB/s, done.
Resolving deltas: 100% (171/171), done.
pi@h-raspi2:~ $
```

Abbildung 31: Adafruit-Bibliothek herunterladen

```
$ cd Adafruit_Python_DHT
$ sudo python setup.py install
```

Update_V2020.10: Fehlermeldung – Setuptools wurden nicht gefunden.
Es muss python3 verwendet werden!

```
$ sudo apt-get install python3-pip
$ sudo python3 setup.py install
```

Nun wird mit Hilfe des mitgelieferten Testprogramms die Funktionstüchtigkeit geprüft.

```
$ cd examples
$ sudo ./AdafruitDHT.py 22 4
```

Dem Script **AdafruitDHT.py** wird die Nummer des Sensors (DHT22) und die Nummer des GPIO-Pins (4) übergeben. Als Ergebnis werden die aktuelle Temperatur und die Feuchtigkeit zurückgegeben.

```
pi@h-raspi2:~/Adafruit_Python_DHT/examples $ cd /home/pi/Adafruit_Python_DHT/examples/
pi@h-raspi2:~/Adafruit_Python_DHT/examples $ sudo ./AdafruitDHT.py 22 4
Temp=26.7* Humidity=58.5%
pi@h-raspi2:~/Adafruit_Python_DHT/examples $
```

Abbildung 32: Messergebnisse für Temperatur und Luftfeuchtigkeit

Update_V2020.10: Auch hier muss die Sensorabfrage mit python3 gemacht werden!

```
pi@h-raspi2:~/Adafruit_Python_DHT/examples $ sudo python3 ./AdafruitDHT.py 22 4
Temp=18.7* Humidity=67.0%
pi@h-raspi2:~/Adafruit_Python_DHT/examples $ sudo python3 ./AdafruitDHT.py 22 4
Temp=19.3* Humidity=66.6%
```

Abbildung 33: Sensorabfrage

Ein Vergleich mit einem konventionellen Digital-Thermometer zeigt nur eine geringe Abweichung an. Für den ersten Test ist das gesamte System im Labor (Innenraum) aufgestellt.

Laut Datenblatt des Sensors dürfen alle drei Sekunden Abfragen durchgeführt werden. Kürzere Intervalle sind nicht möglich. Die geplanten stündlichen Messungsabstände sind also nicht kritisch.

5 Einbau des Luftdrucksensors

Nachdem die Grundfunktionen der Wetterstation mit Wetterbild, Temperatur und Luftfeuchtigkeit problemlos funktioniert, soll nun zusätzlich der Luftdruck gemessen, später auf der Website angezeigt und in einer Datenbank gespeichert werden.

Zur Messung wird der Sensor BMP280 der Fa. BOSCH eingesetzt.

5.1 Allgemeines

Ein Artikel aus (Plate, 2018)⁷ beschreibt die Einführung in die Messung des Luftdrucks so:

Unser Wetter wird deutlich vom Luftdruck und den damit einhergehenden Luftströmungen beeinflusst. Hoher Luftdruck steht meist für sonniges und trockenes Wetter, wogegen Tiefdruck Regen bedeutet. Starke Veränderungen des Luftdrucks innerhalb kurzer Zeit signalisieren einen bevorstehenden Wetterumschwung oder sogar einen Sturm.

Der Luftdruck an einem beliebigen Ort der Erdatmosphäre ist der hydrostatische Druck der Luft, der an diesem Ort herrscht. Dieser Druck entsteht durch die Gewichtskraft der Luftsäule, die auf der Erdoberfläche oder einem Körper steht. Ein Barometer misst den aktuellen Luftdruck. Neben dem Wetter beeinflusst auch die Höhe des Ortes den Luftdruck. Kennt man den aktuellen Luftdruck bezogen auf die Höhe des Meeresspiegels kann man mit Hilfe der barometrischen Höhenformel aus dem aktuellen Luftdruck die aktuelle Höhe, beispielsweise eines Flugzeugs, bestimmen. Um für meteorologische Zwecke vergleichbare Luftdruckwerte zu bekommen, rechnet man den gemessenen Luftdruck immer auf Meereshöhe (Normal-Null) um.

Die international verwendete Maßeinheit für den Luftdruck ist das Pascal (Pa). Um gut handhabbare Zahlenwerte zu erhalten, wird der Luftdruck meist in Hektopascal (hPa) angegeben. Dieser Wert ist dann auch identisch mit der früher verwendeten Einheit Millibar. Der mittlere Luftdruck der Atmosphäre, bezogen auf Meereshöhe, beträgt normgemäß $101325 \text{ Pa} = 1013,25 \text{ hPa} \approx 1 \text{ bar}$.

Der Luftdruck ist einer täglich wiederkehrenden Periodik unterworfen, die zwei Maximal- und zwei Minimalwerte pro Tag aufweist. Grund sind die täglichen Schwankungen der Lufttemperatur. Die Maxima finden sich gegen 10 und 22 Uhr, die Minima gegen 4 und 16 Uhr. Der Luftdruck bewegt sich in der Regel zwischen 900 und 1050 hPa (der tiefste überlieferte Wert war ca. 870 hPa, der höchste Wert lag bei 1085 hPa).

⁷ <http://www.netzmafia.de/skripten/hardware/RasPi/Projekt-BMP280/index.html>

5.2 Der BOSCH-Sensor BMP280

Der verwendete Sensor wird von BOSCH Sensortec GmbH (Sitz: Reutlingen) entwickelt und vertrieben. Verschiedene Distributoren verwenden den Sensor und vertreiben ihn, aufgebaut auf einer kleinen Leiterplatte mit Steckverbinder-Lötstiften. Ein umfangreiches Datenblatt (BOSCH Sensortec GmbH, 2018) dient zur Erläuterung der einzelnen Funktionen.

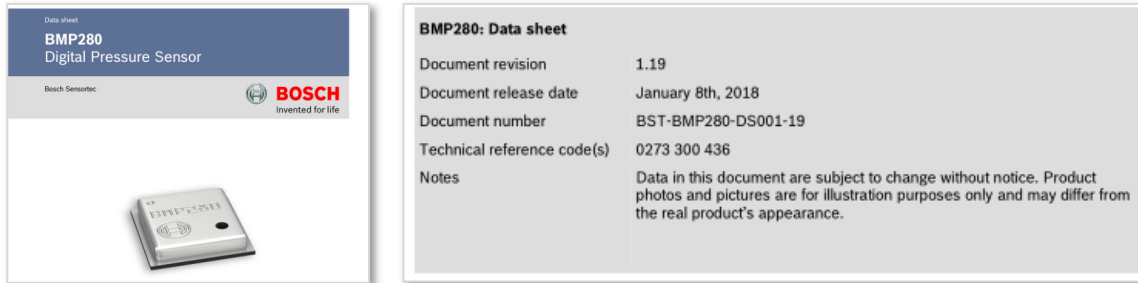


Abbildung 34: Datenblatt-Titel BMP280

Der winzige Sensor kann sowohl die Umgebungstemperatur als auch den Luftdruck messen. Er basiert auf der Piezo-resistiven Druck-Sensorik von Bosch mit hoher Genauigkeit, Linearität und Langzeitstabilität. Die Auflösung beträgt 0,12 Pa. Der Sensor allein ist nur 2,5 mm x 2,5 mm groß und weniger als 1 mm hoch.

Die Verbindung nach außen wird über ein digitales I²C Interface über die Adressen **0x76** und **0x77** gewährleistet. Es sind maximal 60 Messungen pro Minute möglich.

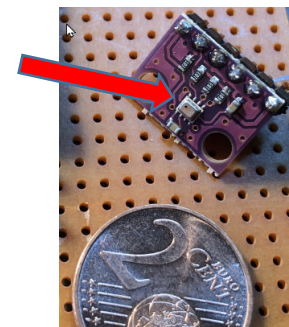


Abbildung 35: BMP280 mit Trägerplatine

Technische Daten

Messbereich	300 bis 1100 hPa (entspricht +9000 bis -500 m)
rel Genauigkeit	±0.12 hPa, (entspricht ±1 m, Bereich 950 bis 1050 hPa bei 25 °C)
absol. Genauigkeit	±1 hPa (Bereich 950 bis 1050 hPa, bei 0 bis +40 °C)
Digitalinterfaces	I ² C (max. 3.4 MHz) / SPI (3 and 4 wire, max. 10 MHz)
Stromverbrauch	2.7µA bei 1 Hz sampling rate
Temperatur-Offset	1.5 Pa/K (entspricht 12.6 cm/K, bei 25 bis 40 °C und 900 hPa)

Blockschaltbild

Abbildung 36 zeigt ein vereinfachtes Blockschaltbild (BOSCH Sensortec GmbH, 2018) des Sensors. Das Sensorelement gibt die Daten an den Analog-Digital-Wandler (ADC), der dann weiter an die Logik bis zum Ausgabeinterface.

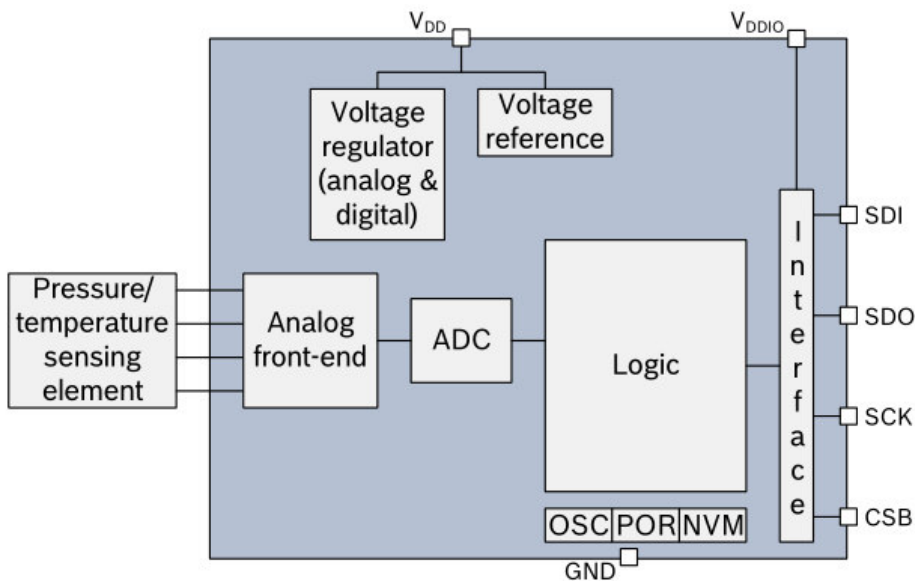


Abbildung 36: Blockschaltbild BMP280 (BOSCH Sensortec GmbH, 2018, S. 11)

Messzyklus

Der Zyklus der Messvorgänge ist in Abbildung 37 ersichtlich. Zunächst wird die Temperatur gemessen, anschließend der Luftdruck. Nach der Analog-Digital-Wandlung erfolgt das Ablegen der Werte im Speicher (output-registers), abhängig von der Einstellung des IIR-Filters. Vom Speicher können die Messwerte mit Hilfe eines I²C-Zugriffsprogramms aus einzelnen Registeradressen abgeholt werden.

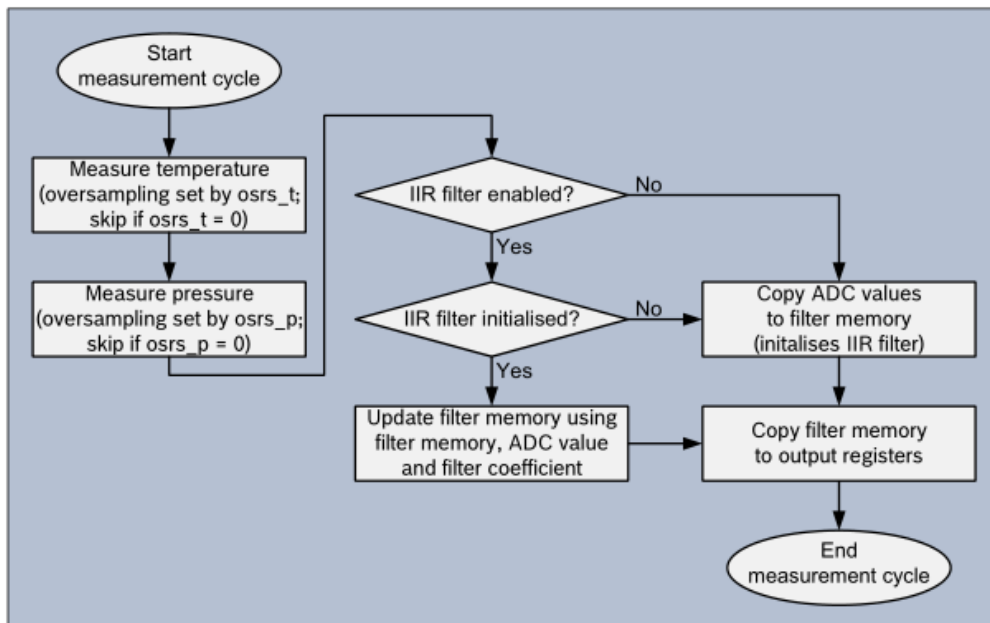


Abbildung 37: Messzyklus BMP280

Je nachdem, welche Genauigkeit gewünscht ist, kann die Samplingrate des AD-Wandlers eingestellt werden. Tabelle 1 zeigt die notwendigen Parameter für die Luftdruckmessung, in Tabelle 2 sind die Parameter für die Temperaturmessung angegeben.

Tabelle 1: Samplingrate der Luftdruckmessung BMP280 (BOSCH Sensortec GmbH, 2018, S. 12)

Oversampling setting	Pressure oversampling	Typical pressure resolution	Recommended temperature oversampling
Pressure measurement skipped	Skipped (output set to 0x80000)	–	As needed
Ultra low power	×1	16 bit / 2.62 Pa	×1
Low power	×2	17 bit / 1.31 Pa	×1
Standard resolution	×4	18 bit / 0.66 Pa	×1
High resolution	×8	19 bit / 0.33 Pa	×1
Ultra high resolution	×16	20 bit / 0.16 Pa	×2

Tabelle 2: Samplingrate der Temperaturmessung BMP280 (BOSCH Sensortec GmbH, 2018, S. 13)

osrs_t[2:0]	Temperature oversampling	Typical temperature resolution
000	Skipped (output set to 0x80000)	–
001	×1	16 bit / 0.0050 °C
010	×2	17 bit / 0.0025 °C
011	×4	18 bit / 0.0012 °C
100	×8	19 bit / 0.0006 °C
101, 110, 111	×16	20 bit / 0.0003 °C

Der Messzyklus kann durch Setzen von Filterbedingungen beeinflusst werden. Hierzu empfiehlt der Hersteller im Datenblatt von der jeweilig gewünschten Anwendung unterschiedliche Einstellungen (siehe Tabelle 3).

Tabelle 3: Filtereinstellungen BMP280 (BOSCH Sensortec GmbH, 2018, S. 14)

Use case	Mode	Over-sampling setting	osrs_p	osrs_t	IIR filter coeff. (see 3.3.3)	I _{DD} [μA] (see 3.7)	ODR [Hz] (see 3.8.2)	RMS Noise [cm] (see 3.5)
handheld device low-power (e.g. Android)	Normal	Ultra high resolution	×16	×2	4	247	10.0	4.0
handheld device dynamic (e.g. Android)	Normal	Standard resolution	×4	×1	16	577	83.3	2.4
Weather monitoring (lowest power)	Forced	Ultra low power	×1	×1	Off	0.14	1/60	26.4
Elevator / floor change detection	Normal	Standard resolution	×4	×1	4	50.9	7.3	6.4
Drop detection	Normal	Low power	×2	×1	Off	509	125	20.8
Indoor navigation	Normal	Ultra high resolution	×16	×2	16	650	26.3	1.6

Weitere Einstellwerte sind zur Vermeidung von Rauschen und thermischen Drifts vorhanden. Das Power-Management verfügt über vier Werte, die Sleep-, Normal- und Forced-Modi bieten.

Verbindungsleitungen I²C

Für die Verbindung vom Raspberry Pi zum kleinen BMP280 sind folgenden Verbindungen notwendig:

VCC: Spannungsversorgung 3,3 V

GND: Masse 0 V

SCL: Serial Clock Line (Taktsignal)

SDA: Serial Data Line (serielle Daten)

CSB und SDB der Sensorplatine bleiben frei.

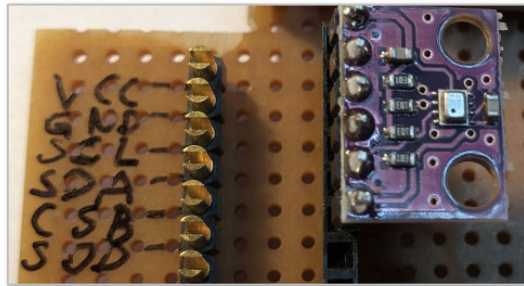


Abbildung 38: Anschaltung der Sensorplatine

Der BMP280 liefert zunächst Rohwerte für Temperatur und Luftdruck, die unkompensiert sind. Die Messdaten müssen mit den beim Herstellungsprozess im 176-Bit-EEPROM abgelegten Kalibrierungswerten umgerechnet werden. Das Auswerte-Programm liest diese Werte aus und verrechnet sie mit den rohen Messdaten. Die Software ist komplex, im Herstellerdatenblatt gibt es wertvolle Informationen und Beispielprogramme hierzu.

Memory-Map

Die Kommunikation mit dem BMT280 geht über den I2C-Bus. Register (8 Bit breit) werden gelesen und beschrieben. Mit Hilfe der Memory-Map aus dem Datenblatt können die **Registernamen** ermittelt werden. Die read-only-Datenregister mit den Messwerten für Temperatur und Druck sind gelb hinterlegt.

4.2 Memory map

The memory map is given in Table 18 below. Reserved registers are not shown.

Table 18: Memory map

Register Name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Reset state
temp_xlsb	0xFC	temp_xlsb<7:4>				0	0	0	0	0x00
temp_lsb	0xFB	temp_lsb<7:0>								0x00
temp_msb	0xFA	temp_msb<7:0>								0x80
press_xlsb	0xF9	press_xlsb<7:4>				0	0	0	0	0x00
press_lsb	0xF8	press_lsb<7:0>								0x00
press_msb	0xF7	press_msb<7:0>								0x80
config	0xF5	t_sb[2:0]			filter[2:0]			spi3w_en[0]		0x00
ctrl_meas	0xF4	osrs_t[2:0]			osrs_p[2:0]			mode[1:0]		0x00
status	0xF3					measuring[0]		im_update[0]		0x00
reset	0xE0	reset[7:0]								0x00
id	0xD0	chip_id[7:0]								0x58
calib25...calib00	0xA1...0x88	calibration data								individual

Registers:	Reserved registers	Calibration data	Control registers	Data registers	Status registers	Revision	Reset
Type:	do not write	read only	read / write	read only	read only	read only	write only

Abbildung 39: BMP280 Memory-Map (BOSCH Sensortec GmbH, 2018, S. 24)

Das Schreiben in ein Register über den I²C-Bus ist in beispielhaft in Abbildung 40 dargestellt, das Lesen aus einem Datenregister zeigt Abbildung 41. Als I²C-Master dient das Steuerprogramm des RaspberryPi, Slave ist der Sensor am Bus.

Zum Schreiben von Daten wird nach dem Start-Bit die I²C-Slave-Adresse im Schreib-Modus gesendet. ACKS ist das Acknowledge- (Bestätigungs-) Signal des Slaves. Danach sendet der Master Registeradresse und Registerdaten. Die Übertragung endet durch ein Stop-Bit.



Abbildung 40: I²C Schreibvorgang (BOSCH Sensortec GmbH, 2018, S. 29)

Der Lesevorgang beginnt genauso wie der Schreibvorgang mit dem Senden der Slaveadresse nach dem Startbit im Schreibmodus. Anschließend wird das Control-Byte gesendet. Danach wird der Sensor in den Lesemodus umgeschaltet. Nun sendet der Slave seine Daten automatisch aus mehreren Register-Bytes, bis das NOACKM-Signal und das Stop-Bit die Übertragung beendet. Im Beispiel werden die Speicherplätze 0xF6 und 0xF7 ausgelesen.

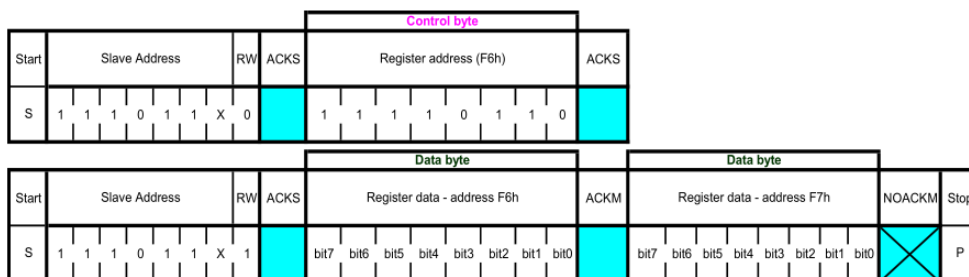


Abbildung 41: I²C Lesevorgang (BOSCH Sensortec GmbH, 2018, S. 30)

Weitere Angaben zur Steuerung der Datenübertragung über die I²C-Schnittstelle finden sich im SBMP280-Datenblatt (z.B. wie in Abbildung 42).

Neben der Steuerung mit dem I²C-Bus kann auch über das SPI-Interface mit 3-wire- oder 4-wire-mode zugegriffen werden.

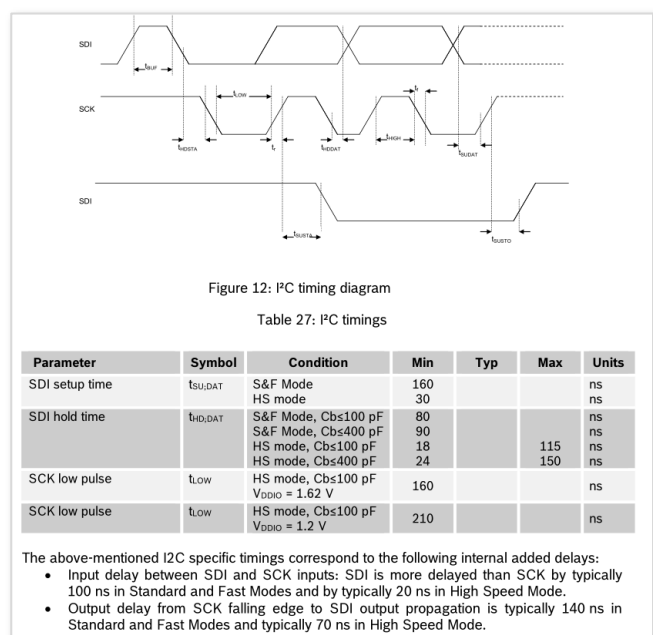


Abbildung 42: I²C-Timing BMP280 (BOSCH Sensortec GmbH, 2018, S. 33)

5.3 mechanischer Umbau

Die Wetterstation muss gegenüber dem ersten Prototypen nun auch mechanisch etwas umgebaut werden. Der bisher nicht gegen Spritzwasser (Regen) geschützte Feuchtigkeitssensor soll zusammen mit dem Luftdrucksensor in dem zusätzlichen Gehäuse untergebracht werden. Damit beide Sensoren einfach kontaktiert und gut befestigt werden können, erfolgt die Montage auf einem kleinen Leiterplattenrest. Dieser wird in seinen Außenmaßen so angepasst, dass er mit einer Schraube im Gehäuse befestigt werden kann.

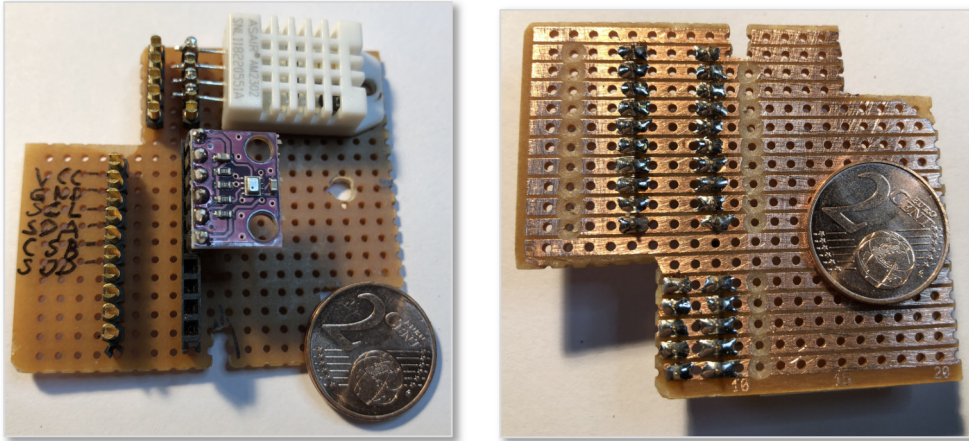


Abbildung 43: Leiterplatte für die Sensoranschlüsse

Der Stromlaufplan der anzufertigenden Verdrahtung zeigt die notwendigen Leitungen für den 1-wire-Datenkanal des DHT22 und den I²C-Bus des BMP280.

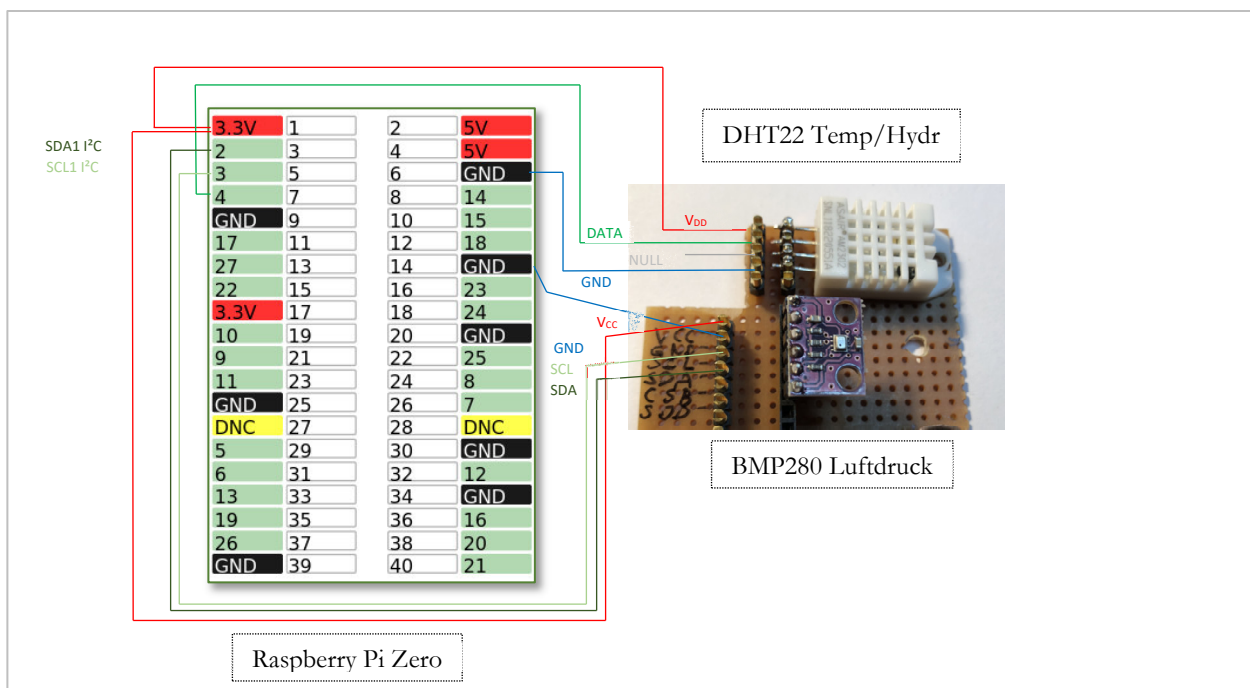


Abbildung 44: Verdrahtung der Sensoren

Nun wird die Verteilerdose ohne Deckel unten an die bestehende Halterung des Raspberry angeschraubt. Auf dem Leiterplattenrest werden Lötstützpunkte zur Verbindung der Sensor-Anschlussleitungen verlötet. Die Leiterplatte wird in die offene Dose montiert. Die Verbindungsleitungen stammen aus einer ausgedienten flexiblen Fernsprech-Anschlussleitung. Folgende Bilder zeigen den Einbau der Sensordose

an der Unterseite des Befestigungswinkes und die später geschlossene spritzwassergeschützte Verteilerdose mit der Raspberry-Platine und der Kamera. Die Kabelstopfen werden bei der Endmontage mit Silikon verschlossen.

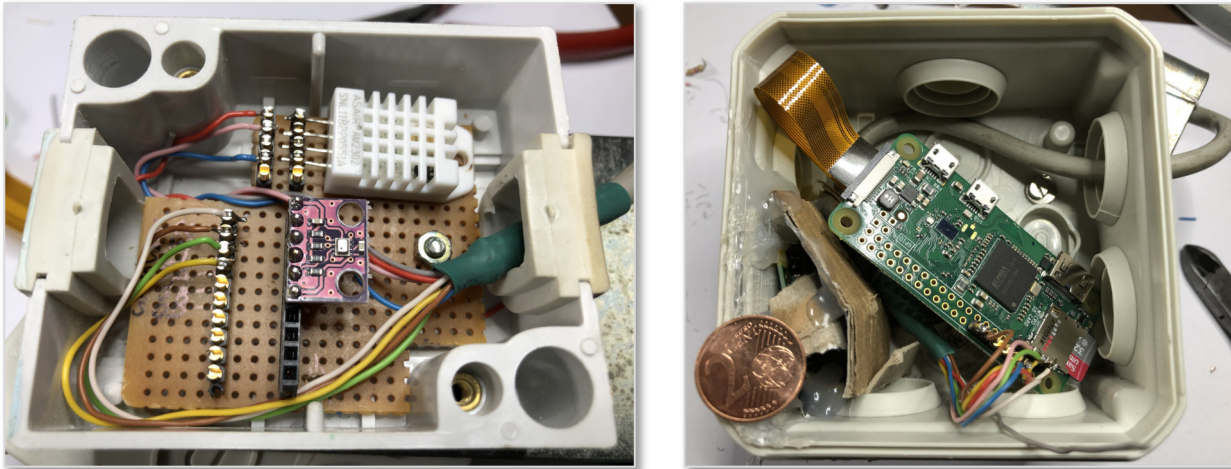


Abbildung 45: Sensor- und Raspberry-Gehäuse

5.4 erster Funktionstest im Labor

Die komplett neu verdrahtete Wetterstation muss getestet werden, so lange der Zugriff im Labor noch relativ einfach zu bewerkstelligen ist. Nach dem Einschalten der Spannungsversorgung ist der Zugriff mit **PuTTY** nach wenigen Sekunden möglich, auch aktuelle Kamerabilder sind auf der Webseite eingebunden.

Nun geht es an den Test des neuen Sensors. Da es sich um einen I²C-Sensor handelt, muss dies über die entsprechende Betriebssystemfunktion geprüft werden. Die Schnittstelle wurde bereits in der Ersteinrichtung aktiviert (siehe Abbildung 22).

```
$ i2cdetect -y 1
```

```
pi@h-raspi2:~ $ i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  76  --  --  --  --  --  --  --  --
```

Abbildung 46: I²C-Sensor ansprechen

Mit **i2cdetect** wird geprüft, ob sich ein Gerät (Slave) auf dem Bus meldet. Dies ist hier unter der Adresse **0x76** der Fall.

Mit **i2cset** können Werte zum Bus gesendet werden. In der Adresse **0xF4** muss dem BMP280 mitgeteilt werden, wie genau das Messergebnis sein soll. Einfach-Oversampling wird laut Datenblatt die Bitkombination **001 001** sein. Der Powermode wird auf normal gesetzt, ergibt komplett **001 001 11** bzw. **0x27**.

Dieser Wert wird in die Speicherzelle 0xF4 des BMP280 geschrieben.

```
pi@h-raspi2:~ $ i2cset -y 1 0x76 0xf4 0x27
```

Nun können per Kommandozeile Daten ausgelesen werden. Ab der Adresse **0xF7** stehen die 20 Bit Daten für Druck und ab **0xFA** die 20 Bit Daten für die Temperatur.

```
pi@h-raspi2:~ $ i2cdump -y 1 0x76
No size specified (using byte-data access)
   0   1   2   3   4   5   6   7   8   9   a   b   c   d   e   f      0123456789abcdef
00: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      .....
10: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      .....
20: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      .....
30: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      .....
40: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      .....
50: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      .....
60: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      .....
70: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      .....
80: 6e 74 90 ab 56 4e a7 00 14 6d d7 63 32 00 65 8f      nt??VN?.?m?c2.e?
90: 75 d6 d0 0b 47 0e 2f 00 f9 ff 8c 3c f8 c6 70 17      u???G?/.?.?<??p?
a0: 00 00 cd 00 00 00 00 00 00 00 00 00 33 00 00 c0      ..?.....3..?
b0: 00 55 00 00 00 00 60 02 00 01 ff cd 13 71 03 00      .U....`?.?.??q?.
c0: 00 00 27 ff 00 00 00 00 00 00 00 00 00 00 00 00      ..'.....
d0: 58 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00      X?.....
e0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00      .....
f0: 00 00 00 00 27 a0 00 64 78 00 81 60 00 00 00 00      ....'?.dx.?'....
```

Abbildung 47: I²C-Dump

Es handelt sich noch um Rohwerte, da die Sensoren beim Hersteller (Bosch) kalibriert werden. Jeder Sensor liefert aufgrund seiner physikalischen Eigenschaften leicht andere Messwerte. Die Kalibrierungswerte werden in den BMP280 gebrannt, müssen also von dort ausgelesen werden und von der Software in eine Berechnung einbezogen werden. So entsteht dann ein genügend genauer Wert.

5.5 Software zum Auslesen der BMP280-Sensordaten

Zum Auslesen der Sensordaten ist ein komplexes Steuerprogramm notwendig, weil die Daten nacheinander über den I²C-Bus abgeholt und dann mit einer barometrischen Höhenformel umgerechnet werden müssen. Die Fa.BOSCH bietet auf GitHub⁸ ein C-Steuerungsprogramm an, das Professor Jürgen Plate von der Hochschule München (Plate, 2018) für Python adaptiert hat. Dieser Quellcode wurde nun für dieses Projekt bearbeitet und zum besseren Verständnis mit ausführlichen Kommentaren versehen.

Beim Starten des rohen Beispiel-Scripts entstand jedoch eine Fehlermeldung. Das für die I²C-Abfrage wichtige Python-**smbus**-Modul wurde nicht gefunden.

Deshalb erfolgt als erster Schritt die Installation der fehlenden Pakete.

```
$ sudo apt-get update
$ sudo apt-get install python-smbus python3-smbus
python-dev python3-dev i2c-tools
```

⁸ https://github.com/BoschSensortec/BMP280_driver

bearbeitetes Testskript zum I²C -Sensor

Die einzelnen Teile des angepassten Testskripts werden hier beschrieben:

Kopfdaten

Das Pythonscript soll die Kommunikation mit dem Sensor, das Auslesen der Daten und die Berechnung durchführen. Anschließend werden die Werte der Datenbank abgelegt.

Kalibrierwerte aus dem Sensor lesen

Die Kompensationsparameter sind jeweils 16-Bit-Werte (unsigned int oder signed int), die jeweils in ein Array abgelegt werden.

Immer zwei Bytes müssen zu einem 16-Bit-Wert zusammengefasst werden, weil der Speicher in 8-Bit (Byte) organisiert ist. Die Werte sind an den Speicheradressen **0x88** bis **0xA1** gespeichert. In Tabelle 4 sind Speicherplätze der Datenwörter für die Temperaturkompensation bezeichnet mit **dig_Txx** und die Datenwörter für die Druckkompensation mit **dig_Pxx**.

Tabelle 4: Kompensationsregister (BOSCH Sensortec GmbH, 2018, S. 21)

Register Address LSB / MSB	Register content	Data type
0x88 / 0x89	dig_T1	unsigned short
0x8A / 0x8B	dig_T2	signed short
0x8C / 0x8D	dig_T3	signed short
0x8E / 0x8F	dig_P1	unsigned short
0x90 / 0x91	dig_P2	signed short
0x92 / 0x93	dig_P3	signed short
0x94 / 0x95	dig_P4	signed short
0x96 / 0x97	dig_P5	signed short

```

1  #!/usr/bin/env python
2  #Sensordaten vom BOSCH BMP280 auslesen
3  #erstellt 21.09.2019 (HK)
4  #-----
5  import smbus
6  import time
7  # Adresse des I2C Bus
8  BUS = 1
9  # BMP280 Adresse, 0x76 oder 0x77
10 BMP280ADDR = 0x76
11
12 # Meereshöhe der Wetterstation 239 m
13 ALTITUDE = 239
14

```

Programcode 1: Kopfdaten des Python-Skripts

```

15 # I2C Bus einlesen
16 bus = smbus.SMBus(BUS)
17
18 # Temperatur kalibrieren (Array)
19 T = [0, 0, 0];
20 # Druck kalibrieren (Array)
21 P = [0, 0, 0, 0, 0, 0, 0, 0, 0];
22
23 # Kalibrierungsdaten aus dem Sensor einlesen
24 # (Daten aus Adresse 0x88, 24 bytes)
25 data = bus.read_i2c_block_data(BMP280ADDR, 0x88, 24)
26
27 # Temperatur-Koeffizienten
28 T[0] = data[1] * 256 + data[0]
29 T[1] = data[3] * 256 + data[2]
30 if T[1] > 32767:
31     T[1] -= 65536
32 T[2] = data[5] * 256 + data[4]
33 if T[2] > 32767:
34     T[2] -= 65536
35
36 # Druck-Koeffizienten
37 P[0] = data[7] * 256 + data[6];
38 for i in range(0, 8):
39     P[i+1] = data[2*i+9]*256 + data[2*i+8];
40     if P[i+1] > 32767:
41         P[i+1] -= 65536

```

Programcode 2: Kalibrierung BMP280

Konfigurationsdaten schreiben

Der folgende Programmteil wählt das Kontroll-Messregister aus und setzt mehrere Messparameter. Anschließend wird das Konfigurationsregister mit weiteren Parameterdaten geladen.

```

42
43 # Kontroll-Messregister auswaehlen (Adresse 0xF4)
44 # In das Register 0b00100111 schreiben (=0x27)
45 # 0x27 --> pressure/temperature oversampling rate = 1, normal mode
46 bus.write_byte_data(BMP280ADDR, 0xF4, 0x27)
47
48 # Konfigurationsregister auswaehlen, (Adresse 0xF5)
49 # beschreiben mit 0xA0: standby time = 1000 ms
50 bus.write_byte_data(BMP280ADDR, 0xF5, 0xA0)
51
52 #Kurze Wartezeit
53 time.sleep(1.0)
54

```

Programcode 3: BMP280 - Konfigurationsdaten schreiben

Die folgenden Tabellen zeigen die Inhalte der beschriebenen Steuer-Register und deren Bedeutung.

Tabelle 5: BMP280 Controlregister

Controlregister (ctrl_meas)								
Adresse	Bit 7, Bit6, Bit5 (osrs_t)			Bit 4, Bit3, Bit2 (osrs_p)			Bit1, Bit0 (mode)	
0xF4	0	0	1	0	0	1	1	1
Datenwert (Hex)	2			7				

osrs_t[2:0]	Temperature oversampling	Typical temperature resolution
000	Skipped (output set to 0x80000)	–
001	×1	16 bit / 0.0050 °C
010	×2	17 bit / 0.0025 °C

osrs_p[2:0]	Pressure oversampling
000	Skipped (output set to 0x80000)
001	oversampling ×1
010	oversampling ×2

mode[1:0]	Mode
00	Sleep mode
01 and 10	Forced mode
11	Normal mode

Abbildung 48: Controlregister

Das Konfigurationsregister hat ähnliche Bitaufteilung wie das Controlregister. Die Standby-Zeit ist auf 1000 ms gesetzt, der IIR-Filter wird abgeschaltet und der Baustein auf I2C-Kommunikation eingestellt.

Tabelle 6: BMP280 Konfigurationsregister

Konfigurationsregister (config)							
Adresse 0xF5	Bit 7, Bit6, Bit5 (t_sb)			Bit 4, Bit3, Bit2 (filter)			Bit0 (spi3w_en)
	1	0	1	0	0	0	0
Datenwert (Hex)	A			0			

t_sb[1:0]	tstandby [ms]
000	0.5
001	62.5
010	125
011	250
100	500
101	1000

Messwertdaten einlesen

Für jeden Messwert sind im Arbeitsspeicherregister drei Byte, also 24 Bit reserviert.

Register Name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
temp_xlsb	0xFC	temp_xlsb<7:4>				0	0	0	0
temp_lsb	0xFB	temp_lsb<7:0>							
temp_msb	0xFA	temp_msb<7:0>							

Abbildung 49: BMP280 Memory-Map Messregister (BOSCH Sensortec GmbH, 2018, S. 24)

Die gesamte Bitverteilung ist am Beispiel eines Temperaturmesswerts folgendermaßen festgelegt:

2 ²³	2 ²²	2 ²¹	2 ²⁰	2 ¹⁹	2 ¹⁸	2 ¹⁷	2 ¹⁶	2 ¹⁵	2 ¹⁴	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
																				0	0	0	0
msb (0xFA)								lsb (0xFB)								xlsb (0xFC)							

Am Beispiel einer I²C-Dump-Ausgabe wird der aktuelle Inhalt des Temperatur-Messregisters dargestellt.

```
$ i2cdump -y 1 0x76
```

f0: 00 00 00 0c 27 00 00 62 b2 00 79 4a 00 00 00 00

2 ²³	2 ²²	2 ²¹	2 ²⁰	2 ¹⁹	2 ¹⁸	2 ¹⁷	2 ¹⁶	2 ¹⁵	2 ¹⁴	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	2 ⁷	2 ⁶		2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
7				9				4				A					0				0			
0	1	1	1	1	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0

Insgesamt kann dabei eine Auflösung von 20 Bit erreicht werden (2⁰ bis 2³ sind fest auf 0).

Messwert bilden durch Bitverschiebung

Damit die drei Speicherplätze (Register) zusammengefasst und später als zusammenhängender Messwert vom Python-Programm ausgewertet werden, muss eine Bitverschiebung vorgenommen werden.

Die Bits des Registers **0xFA** müssen um 12 Stellen nach links verschoben werden.

Bitweise daran angehängt werden dann alle 8 Bit des Registers **0xFB**. Damit sind 16 Stellen gefüllt.

Die vier Bit des Registers 0xFC müssen nach rechts verschoben werden, um an die niedrigste Stelle der Variablen zu gelangen.

Die 0-Werte des Registers **0xFC** fallen weg.

Die Variable adc_t muss von rechts „gefüllt“ werden, d.h. zunächst wird der Inhalt vom 8-Bit-Speicherplatz 0xFA um 12 Stellen nach links verschoben. Die weiteren Speicherinhalte müssen ebenfalls verschoben werden, damit ein 20-Bit-Wert entsteht.	Adresse	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	0xFA	0	1	1	1	1	0	0	1
	0xFB	0	1	0	0	1	0	1	0
	0xFC	0	0	0	0	0	0	0	0

Messwertvariable für die Temperatur adc_t	2 ¹⁹	2 ¹⁸	2 ¹⁷	2 ¹⁶	2 ¹⁵	2 ¹⁴	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
	0	1	1	1	1	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0

Im Pythonscript dient die Variable **adc_t** als Messwertspeicher. Durch die Bitverschiebung werden die drei Messbytes aneinandergefügt und gespeichert. Im Programmcode wird dies in einer einzigen Zeile über den ODER-Operator **|** und die Bitverschiebung **<<** bzw. **>>** durchgeführt.

```
adc_t = (data[3] << 12) | (data[4] << 4) | (data[5] >> 4)
```

```

54
55 # Daten lesen aus 0xF7 (Druck und Temp., 6 bytes)
56 data = bus.read_i2c_block_data(BMP280ADDR, 0xF7, 6)
57
58 # Konvertierung der Daten in je 20-Bit-Werte
59 # Speicherung in je einer Variablen
60 adc_p = (data[0] << 12) | (data[1] << 4) | (data[2] >> 4)
61 adc_t = (data[3] << 12) | (data[4] << 4) | (data[5] >> 4)
62

```

Programcode 4: BMP280 - Bitverschiebung

Anmerkung: Er reicht auch, nur die Register für **lsb** und **msb** auszulesen, wenn nicht die volle Genauigkeit benötigt wird.

Berechnung

Das Programm berechnet zunächst die Temperatur und den lokal gemessenen absoluten Luftdruck aus den Mess- und Kalibrierungsdaten. Die Ergebnisse sind abschließend in den Variablen **temperature** und **pressure** gespeichert.

```

62
63 # Temperatur Offset Berechnung aus Messdaten und Kalibrierungsdaten
64 temp1 = ((adc_t)/16384.0 - (T[0])/1024.0)*(T[1]);
65 temp3 = (adc_t)/131072.0 - (T[0])/8192.0;
66 temp2 = temp3*temp3*(T[2]);
67 temperature = (temp1 + temp2)/5120.0
68
69 # Luftdruck Offset Berechnung aus Messdaten und Kalibrierungsdaten
70 press1 = (temp1 + temp2)/2.0 - 64000.0
71 press2 = press1*press1*(P[5])/32768.0
72 press2 = press2 + press1*(P[4])*2.0
73 press2 = press2/4.0 + (P[3])*65536.0
74 press1 = ((P[2])*press1*press1/524288.0 + (P[1])*press1)/524288.0
75 press1 = (1.0 + press1/32768.0)*(P[0])
76 press3 = 1048576.0 - (adc_p)
77 if press1 != 0:
78     press3 = (press3 - press2/4096.0)*6250.0/press1
79     press1 = press3*press3*(P[8])/2147483648.0
80     press2 = press3*(P[7])/32768.0
81     pressure = (press3 + (press1 + press2 + (P[6]))/16.0)/100
82 else:
83     pressure = 0
84

```

Programmcod 5: Berechnung der Messdaten

Bei einer Wetterstation ist die Höhe der Station über NN bekannt und es soll aus dem gemessenen absoluten Luftdruck der auf Meereshöhe bezogene Luftdruck bestimmt werden.

Der lokale Luftdruck (**pressure**) in Pascal muss aus dem Berechnungsteil des Scripts übergeben werden. Die Höhe des Sensorstandpunkts über dem Meeresspiegel NN (**ALTITUDE**) in Metern stammt aus dem Kopfteil des Programms. Die Berechnung des relativen Luftdrucks erfolgt mittels einer vereinfachten Höhenformel. Ein Test auf der Kommandozeile zeigt die aktuellen Sensordaten an:

```

84
85 # Druck relativ zu Seehöhe NN
86 pressure_nn = pressure/pow(1 - ALTITUDE/44330.0, 5.255)
87
88 # Ausgabedaten
89 print "Temperatur          : %.2f C" %temperature
90 print "abs. Luftdruck      :   %.2f hPa " %pressure
91 print "rel. Luftdruck NN : %.2f hPa " %pressure_nn
92
93 # Ende des Messprogramms

```

```

Temperatur          : 23.80 C
abs. Luftdruck      :   984.82 hPa
rel. Luftdruck NN : 1013.20 hPa

```

Programmcod 6: BMP280 - Messwerte ausgeben

6 Aufbau „vor Ort“

Für die weiteren Installationsschritte und zum Test der optimalen Positionierung soll die Hardware am Einsatzortes montiert werden.

- Montage der Hardware
- Test: Weitere Erreichbarkeit per WLAN
- Ausrichtung der Wettercam
- Laufzeitmessung / Übertragungsleistung

6.1 Montage

Die wetterfeste Aufputzdose wird mit einem Winkel am Dachbodenfester montiert. Für die Spannungsversorgung des Raspberry Pi wird das Mico-USB-Kabel durch den Fensterrahmen in die Dose geführt und am angeschlossen. Die zweite Dose, die Sensoren enthält, wird unterhalb angeschraubt. In den Deckel werden einige Löcher gebohrt, um für Luftdruck und Temperatur genügend Luftaustausch zu erreichen.



Abbildung 50: Entmontage der Wetterstation⁹

⁹ Bilder: HK 2019

Nun werden einige Testbilder mit Hilfe des Tools **raspistill** angefertigt und die Ausrichtung der Kamera geringfügig abgeglichen. Es wurde darauf geachtet, dass keine benachbarten Grundstücksflächen im Bild dargestellt werden. Aufgrund des Weitwinkelobjektivs der Kamera muss das Bild nach der Aufnahme noch beschnitten werden. Dies wird später eine Funktion eines Softwarepakets erledigen.

```
$ sudo raspistill -o /home/pi/testfotos/testbild.jpg
```

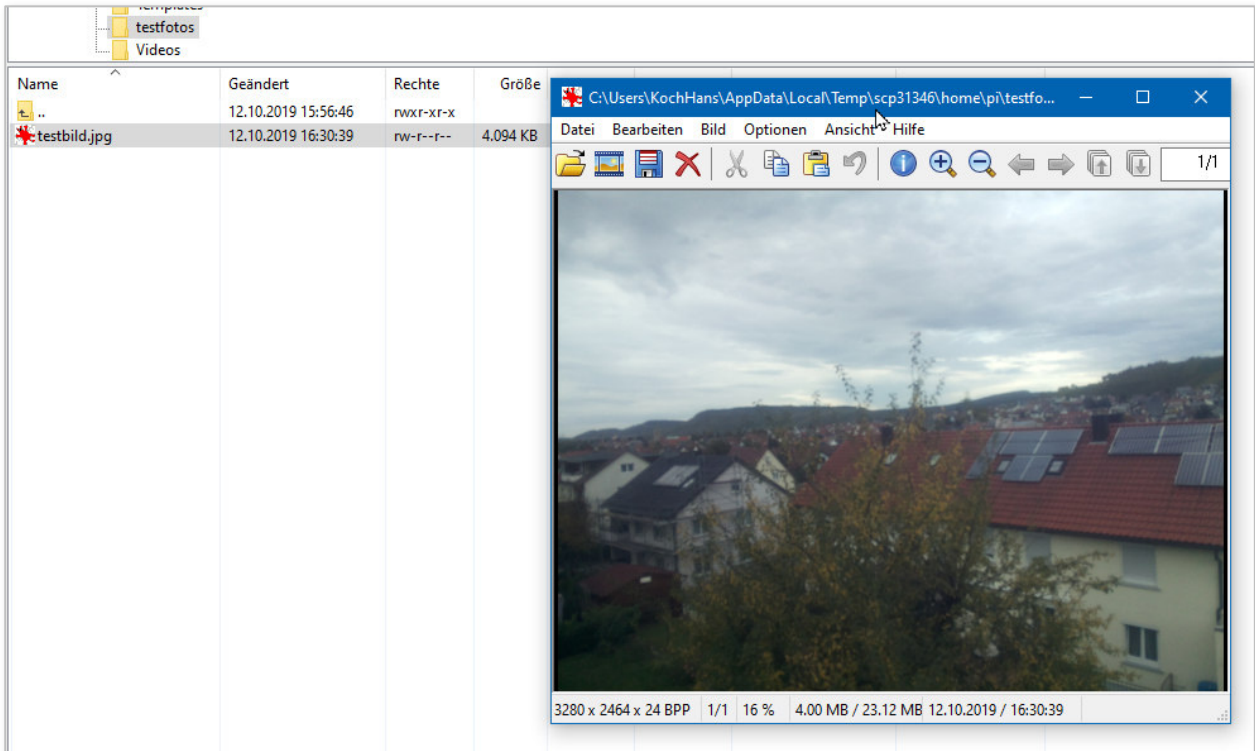


Abbildung 51: Testbilder der Webcam

6.2 Verbindungstest

Nachdem die Leiterplatte mit der Spanungsversorgung verbunden ist, bootet das Betriebssystem und es werden Verbindungstests durchgeführt. Hierzu müssen zunächst **iperf**-Pakete nachgerüstet werden.

```
pi@h-raspi2:~ $ sudo apt-get install iperf
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  iperf
0 upgraded, 1 newly installed, 0 to remove and 2 not upgraded.
Need to get 73.8 kB of archives.
```

Nun wird der **iperf**-Server auf dem Raspi gestartet. Er hört auf TCP-Port 5001.

```
$ sudo iperf -s
```

Die Messung erfolgt von einem Windows-PC aus über das grafische Tool **jperf**. Der Raspberry Pi zeigt die Verbindung zum Windowsrechner an.

Das Ergebnis mehrerer Laufzeittest brachte über die WLAN-Verbindung eine Übertragungsrate von ca. 3,8 MBit/sec. Das Tool **jperf** zeigt sowohl eine tabellarische als auch eine grafische Auswertung.

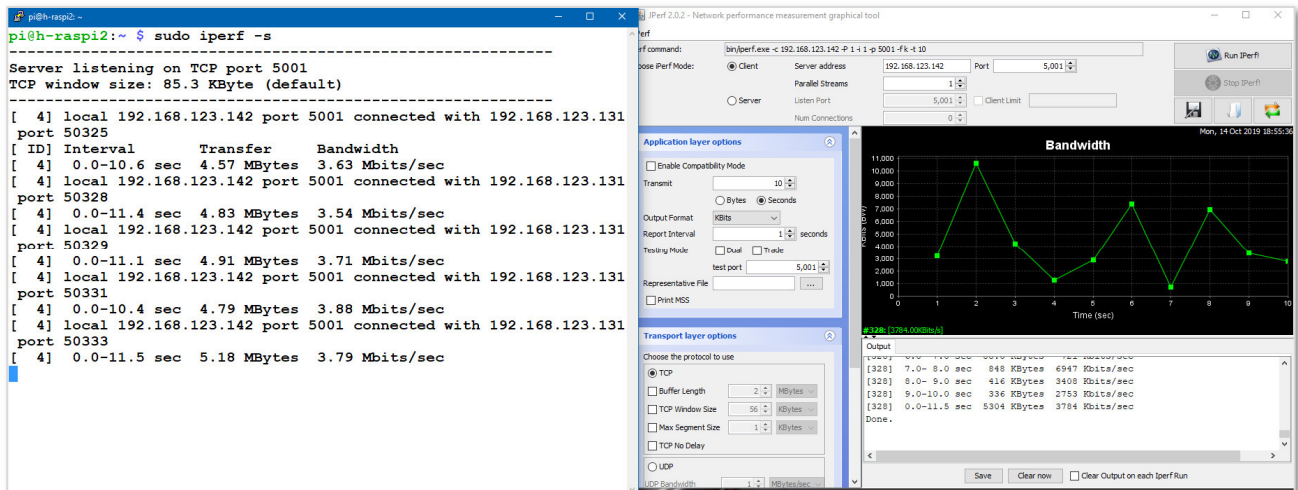


Abbildung 52: Bandbreitentest mit jperf

7 Serverinstallation

Zur Darstellung und Auswertung der gemessenen Wetterdaten sind nun weitere Serverdienste auf dem Raspberry Pi Zero nötig. Für das Hosting der Sensordaten und der Wetterbilder soll der Webserver Apache2 mit der Scriptsprache PHP7.3 und der Datenbankbindung MariaDB eingesetzt werden.

7.1 Vorbereitung

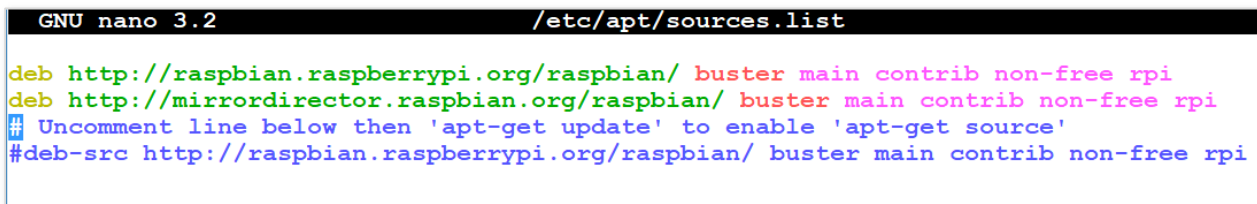
Als erstes sollte ein Systemupdate gemacht werden.

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Anschließend müssen die aktuellen Quellen dem System bekannt gemacht werden.

```
$ sudo nano /etc/apt/sources.list
```

Die zweite Zeile wird der Sources-Liste hinzugefügt.

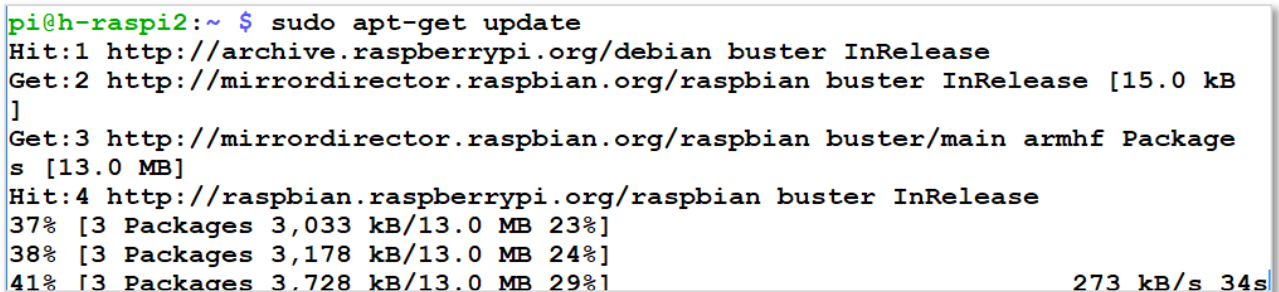


```
GNU nano 3.2 /etc/apt/sources.list
deb http://raspbian.raspberrypi.org/raspbian/ buster main contrib non-free rpi
deb http://mirrordirector.raspbian.org/raspbian/ buster main contrib non-free rpi
# Uncomment line below then 'apt-get update' to enable 'apt-get source'
#deb-src http://raspbian.raspberrypi.org/raspbian/ buster main contrib non-free rpi
```

Abbildung 53: /etc/apt/sources.list erweitern

Nun erfolgt nochmals ein Systemupdate.

```
$ sudo apt-get update
$ sudo apt-get upgrade
```



```
pi@h-raspi2:~ $ sudo apt-get update
Hit:1 http://archive.raspberrypi.org/debian buster InRelease
Get:2 http://mirrordirector.raspbian.org/raspbian buster InRelease [15.0 kB]
Get:3 http://mirrordirector.raspbian.org/raspbian buster/main armhf Packages [13.0 MB]
Hit:4 http://raspbian.raspberrypi.org/raspbian buster InRelease
37% [3 Packages 3,033 kB/13.0 MB 23%]
38% [3 Packages 3,178 kB/13.0 MB 24%]
41% [3 Packages 3.728 kB/13.0 MB 29%] 273 kB/s 34s
```

Abbildung 54: Systemupdate

7.2 Datenbankserver MariaDB

Damit später die Messdaten in einer Datenbank abgelegt und auch ausgewertet werden können, muss ein Datenbanksystem installiert werden.

Übersicht

MariaDB ist ein freies, relationales Open-Source-Datenbankmanagementsystem, das durch eine Abspaltung aus MySQL entstanden ist. Das Projekt wurde von MySQLs früherem Hauptentwickler Michael Widenius initiiert, der auch die Storage-Engine Aria entwickelte, auf welcher MariaDB ursprünglich aufbaute (das ist die Software-Schicht, welche die Basisfunktionalität der Datenbank enthält, d. h. das Erstellen, Lesen, Ändern, Löschen von Daten). Da Oracle die Markenrechte an MySQL hält, mussten neue Namen für das Datenbanksystem und dessen Storage-Engines gefunden werden. Der Name MariaDB geht auf Widenius' jüngere Tochter Maria zurück; seine andere Tochter My war bereits die Namensgeberin für MySQL.

Seit Ende 2012 haben einige Linux-Distributionen MySQL durch MariaDB als Standard-Installation ersetzt, dazu gehören Fedora, CentOS, openSUSE, Slackware und Arch Linux. Die Wikimedia Foundation, die unter anderem auch die Server für die Wikipedia bereitstellt, hat ihre Produktsysteme im April 2013 auf MariaDB umgestellt. Damit hat sich eine der weltweit größten Web-Plattformen von MySQL verabschiedet. Die MariaDB- und MySQL-Server sind keine monolithischen Datenbankserver wie z. B. PostgreSQL. Diese Server kann man sich als Framework für "pluggable engines" vorstellen. Als Standard-Engine verwenden beide seit MariaDB 10.2 die identische InnoDB-Engine, auf die in der Regel auch Applikationen zurückgreifen. Der SQL-Dialekt entspricht dem „Standard-SQL“, und zwischen MySQL und MariaDB gibt es keine essenziellen Unterschiede. Aus Sicht von Applikationen sind zwischen MariaDB Server und MySQL Server keine Inkompatibilitäten bekannt, d. h. man kann MariaDB und MySQL einfach ersetzen. Die Daten-Dateien der InnoDB sind kompatibel und damit austauschbar.¹⁰

Installation

Im folgenden Arbeitsschritt wird der Datenbankserver Maria DB installiert. Der Maria DB-Client wird für den Kommandozeilen-Zugriff auf den Server benötigt.

```
$ sudo apt-get -y install mariadb-server mariadb-client
```

Nun werden die Zugangsdaten eingetragen.

```
$ sudo mysql_secure_installation
```

- Set root password? [Y/n] Y (da bereits das root-Passwort für den LINUX-User gesetzt ist, kann hier mit [n] bestätigt werden)
- Remove anonymous users? [Y/n] Y
- Disallow root login remotely? [Y/n] n
//wird hier Y gewählt, muss der root-Account später manuell erlaubt werden
- Remove test database and access to it? [Y/n] Y
- Reload privilege tables now? [Y/n] Y

¹⁰ Quelle: Ausschnitt aus <https://de.wikipedia.org/wiki/MariaDB>

Installationsablauf von MariaDB

```

pi@h-raspi2:~ $ sudo mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user.  If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

You already have a root password set, so you can safely answer 'n'.

Change the root password? [Y/n] Y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
... Success!

By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] Y
... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] Y
... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] Y
- Dropping test database...
... Success!
- Removing privileges on test database...
... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] Y
... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
pi@h-raspi2:~ $

```

Abbildung 55: Installationsablauf MariaDB

Wurde während der Installation RemoteLogin versehentlich nicht erlaubt, muss dies nun noch erledigt werden.

```
$ sudo mysql -u root
```

```
MariaDB [(none)]> use mysql;
```

```
MariaDB [mysql]> update user set plugin='' where User='root';
```

```
MariaDB [mysql]> flush privileges;
```

```
MariaDB [mysql]> exit
```

```
pi@h-raspi2:~ $ mysql -u root
ERROR 1698 (28000): Access denied for user 'root'@'localhost'
pi@h-raspi2:~ $ sudo mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 56
Server version: 10.3.15-MariaDB-1 Raspbian testing-staging

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement
.

MariaDB [(none)]> use mysql;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [mysql]> update user set plugin='' where User='root';
Query OK, 1 row affected (0.003 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [mysql]> flush privileges;
Query OK, 0 rows affected (0.004 sec)

MariaDB [mysql]> exit
Bye
pi@h-raspi2:~ $
```

Abbildung 56: Anpassung MariaDB-login

Die Datenbank soll später durch das webbasierte Frontend PhpMyAdmin bedient werden. Hierzu ist es jedoch erforderlich einen Webserver und die Skriptsprache PHP zu installieren.

7.3 Webserver Apache 2 installieren

Der Webserverdienst als Service, um Webseiten auf dem RaspberryPI zu organisieren und auf Anforderung des Besuchers auszuliefern.

Allgemeines zur Software¹¹

Der Apache HTTP Server ist ein quelloffenes und freies Produkt der Apache Software Foundation und einer der meistbenutzten Webserver im Internet. Eine Gruppe von acht Entwicklern begann 1994 den Webserver NCSA HTTPd zu erweitern. [...] Sie gaben dem Ergebnis ihrer Arbeit den Namen Apache HTTP Server und veröffentlichten diesen im April 1995. Er war das Gründungsprojekt der Apache Software Foundation. [...]

Der Apache bietet die Möglichkeit, mittels serverseitiger Skriptsprachen Webseiten dynamisch zu erstellen. Häufig verwendete Skriptsprachen sind PHP, Perl oder Ruby. Weitere Sprachen sind Python, JavaScript (z. B. V8CGI), Lua, Tcl und .NET (mit ASP.NET oder Mono). Diese sind kein Bestandteil des Webservers, sondern müssen ebenfalls entweder als Module eingebunden werden oder über das CGI angesprochen werden, da Apache im Gegensatz zu beispielsweise nginx modulbasiert ist. Die Module können jederzeit aktiviert oder deaktiviert werden. Über das bei der Apache-Installation enthaltene `mod_include` kann Server Side Includes (SSI) ausgeführt werden. Damit ist es möglich, einfache dynamische Webseiten zu erstellen und den Verwaltungsaufwand von statischen Webseiten zu minimieren.

Der Apache HTTP Server ist, wie alle Programme der Apache Software Foundation, eine freie Software. Derzeit wird noch die stabile Version 2.4.x unterstützt und somit beispielsweise mit Sicherheitsupdates versorgt. Die Apache-Entwickler empfehlen die Version 2.4.x für den Produktiveinsatz.

Installation

Die Installation gestaltet sich einfach von der Kommandozeile aus.

```
sudo apt-get -y install apache2
```

Die Installation dauert einige Minuten, danach ist ein erster Zugriffstest möglich.

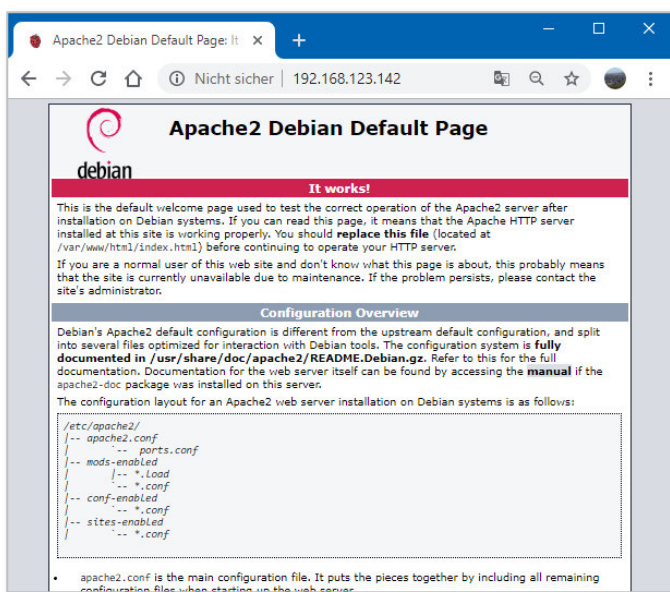


Abbildung 57: Apache2-Startseite

¹¹ Quelle: aus https://de.wikipedia.org/wiki/Apache_HTTP_Server

7.4 Ordnerstruktur für den Webserver

Um für die einzelnen Anwendungen des Webserver eine übersichtliche Struktur zu schaffen, wird diese nun erzeugt. Unterhalb des Verzeichnisses `www` sollen weitere Verzeichnisse erstellt werden.

- **`/var/www/html`**
Anfragen von außen gehen vom Apache2 in dieses Verzeichnis. Dort wird nach einer Datei mit dem Namen `index.html` oder `index.php` gesucht.
- **`/var/www/html/wetterbilder`**
Dies ist das Verzeichnis das aktuelle Wetterbild.
- **`/var/www/html/sensordaten`**
Dies ist das Verzeichnis für die aktuellen Sensordaten.
- **`/var/www/files`**
Dieses Verzeichnis dient als Ablage für Daten, die nicht vom Apache2 und damit von außen geöffnet werden können. Hier werden z.B. Zugangsdaten zur Datenbank gespeichert. Nur ein php-Skript darf darauf zugreifen.
- **`/var/www/cgi-bin`**
Hier werden Skripte abgelegt, die ebenfalls nur von einem php-Skript benutzt werden können.

Das Anlegen der Verzeichnisse geschieht entweder über die Kommandozeile mit `mkdir` oder über das Tool WinSCP. Die fertige Ordnerstruktur sieht so aus:

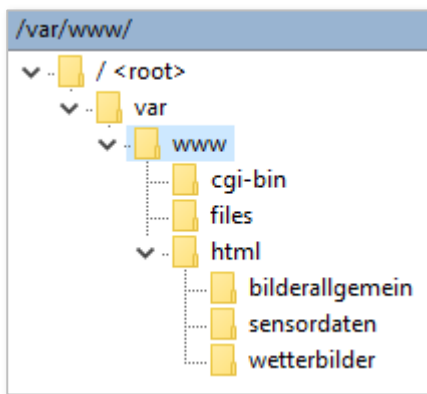


Abbildung 58: Ordnerstruktur auf dem Webserver

7.5 PHP 7

Die serverbasierte Scriptsprache PHP7.3 wird für die Programmierung der dynamischen Webseiten, vor allem bei der Datenbankanbindung benötigt.

PHP7.3 – Pakete werden ausgewählt und installiert.

```
# apt-get -t buster -y install php7.3 php7.3-mysql php7.3-curl
  php7.3-gd php7.3-zip php7.3-fpm php7.3-cli php7.3-opcache
  php7.3-json php7.3-mbstring php7.3-xml libapache2-mod-php7.3
```

Nun muss PHP 7 FPM eingerichtet werden.

```
# a2enmod proxy_fcgi setenvif
# a2enconf php7.3-fpm
```

```
pi@h-raspi2:~ $ sudo a2enmod proxy_fcgi setenvif
Considering dependency proxy for proxy_fcgi:
Enabling module proxy.
Enabling module proxy_fcgi.
Module setenvif already enabled
To activate the new configuration, you need to run:
    systemctl restart apache2
pi@h-raspi2:~ $

pi@h-raspi2:~ $ sudo a2enconf php7.3-fpm
Enabling conf php7.3-fpm.
To activate the new configuration, you need to run:
    systemctl reload apache2
pi@h-raspi2:~ $
```

Abbildung 59: php7.3 einrichten

Ein Neustart des Apachesystems schließt die Installation ab.

```
# systemctl restart apache2
```

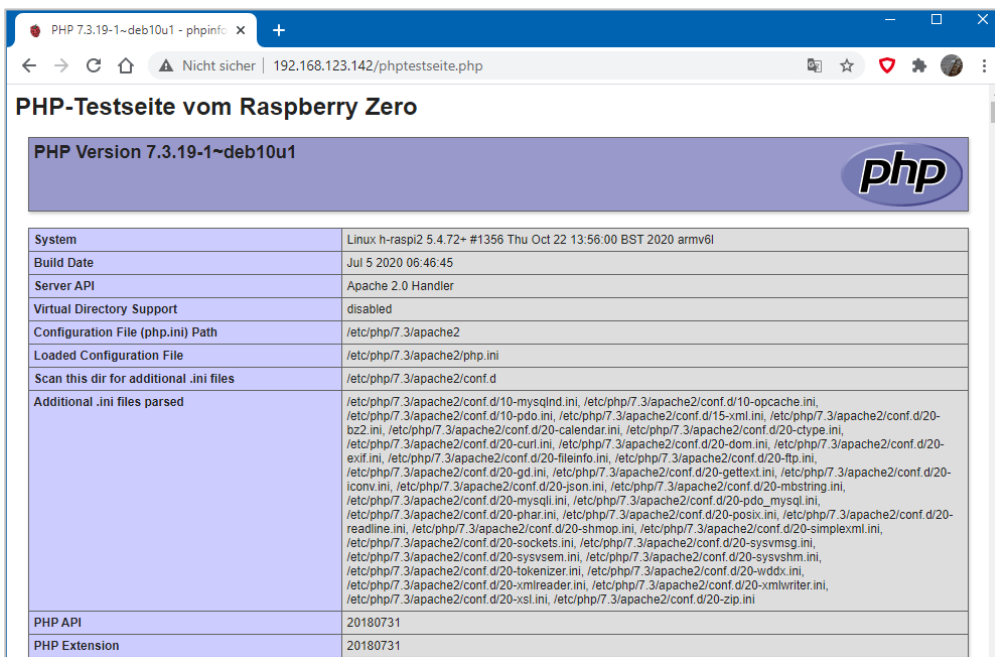
Zum Funktionstest wird eine php-Seite erstellt, die lediglich den Aufruf der Funktion `phpinfo()` enthält. Diese Seite wird im Webverzeichnis des Apache2 abgelegt (`/var/www/html`) und kann so mit einem Browser aufgerufen werden.

```
GNU nano 3.2 /var/www/html/phptestseite.php

<h1>PHP-Testseite vom Raspberry Zero</h1>
<?php
phpinfo();
?>
```

Abbildung 60: php-Testseite (Code)

Die Testseite zeigt die Konfigurationen des Raspberry Pi Zero an und wird hier in Ausschnitten wiedergegeben:



PHP-Testseite vom Raspberry Zero

PHP Version 7.3.19-1~deb10u1

System	Linux h-raspil2 5.4.72+ #1356 Thu Oct 22 13:56:00 BST 2020 armv6l
Build Date	Jul 5 2020 06:46:45
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.3/apache2
Loaded Configuration File	/etc/php/7.3/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.3/apache2/conf.d
Additional .ini files parsed	/etc/php/7.3/apache2/conf.d/10-mysqld.ini, /etc/php/7.3/apache2/conf.d/10-opcache.ini, /etc/php/7.3/apache2/conf.d/10-pdo.ini, /etc/php/7.3/apache2/conf.d/15-xml.ini, /etc/php/7.3/apache2/conf.d/20-bz2.ini, /etc/php/7.3/apache2/conf.d/20-calendar.ini, /etc/php/7.3/apache2/conf.d/20-ctype.ini, /etc/php/7.3/apache2/conf.d/20-curl.ini, /etc/php/7.3/apache2/conf.d/20-dom.ini, /etc/php/7.3/apache2/conf.d/20-exif.ini, /etc/php/7.3/apache2/conf.d/20-fileinfo.ini, /etc/php/7.3/apache2/conf.d/20-ftp.ini, /etc/php/7.3/apache2/conf.d/20-gd.ini, /etc/php/7.3/apache2/conf.d/20-gettext.ini, /etc/php/7.3/apache2/conf.d/20-iconv.ini, /etc/php/7.3/apache2/conf.d/20-json.ini, /etc/php/7.3/apache2/conf.d/20-mbstring.ini, /etc/php/7.3/apache2/conf.d/20-mysql.ini, /etc/php/7.3/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.3/apache2/conf.d/20-phar.ini, /etc/php/7.3/apache2/conf.d/20-posix.ini, /etc/php/7.3/apache2/conf.d/20-readline.ini, /etc/php/7.3/apache2/conf.d/20-shmop.ini, /etc/php/7.3/apache2/conf.d/20-simplexml.ini, /etc/php/7.3/apache2/conf.d/20-sockets.ini, /etc/php/7.3/apache2/conf.d/20-sysmsg.ini, /etc/php/7.3/apache2/conf.d/20-syssem.ini, /etc/php/7.3/apache2/conf.d/20-sysshm.ini, /etc/php/7.3/apache2/conf.d/20-tokenizer.ini, /etc/php/7.3/apache2/conf.d/20-wddx.ini, /etc/php/7.3/apache2/conf.d/20-xmlreader.ini, /etc/php/7.3/apache2/conf.d/20-xmlwriter.ini, /etc/php/7.3/apache2/conf.d/20-xsl.ini, /etc/php/7.3/apache2/conf.d/20-zip.ini
PHP API	20180731
PHP Extension	20180731

~~

Configuration
apache2handler

Apache Version	Apache/2.4.38 (Raspbian)
Apache API Version	20120211
Server Administrator	webmaster@localhost
Hostname:Port	127.0.1.1:80
User/Group	www-data(33)/33
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 300 - Keep-Alive: 5
Virtual Server	Yes
Server Root	/etc/apache2
Loaded Modules	core mod_so mod_watchdog http_core mod_log_config mod_logio mod_version mod_unixd mod_access_compat mod_alias mod_auth_basic mod_auth_core mod_authn_file mod_authn_core mod_authz_host mod_authz_user mod_autoindex mod_deflate mod_dir mod_env mod_filter mod_mime prefork mod_negotiation mod_php7 mod_proxy mod_proxy_fcgi mod_reqtimeout mod_setenvif mod_status

Directive	Local Value	Master Value
engine	1	1
last_modified	0	0
xbithack	0	0

Apache Environment

Variable	Value
HTTP_HOST	192.168.123.142
HTTP_CONNECTION	keep-alive
HTTP_UPGRADE_INSECURE_REQUESTS	1
HTTP_USER_AGENT	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/86.0.4240.198 Safari/537.36
HTTP_ACCEPT	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sign

~~

mysqli

Mysqli Support	enabled
Client API library version	mysqlnd 5.0.12-dev - 20150407 - \$Id: 7cc7cc96e675f6d72e5cf0f267f48e167c2abb23 \$
Active Persistent Links	0
Inactive Persistent Links	0
Active Links	0

Directive	Local Value	Master Value
mysqli.allow_local_infile	Off	Off
mysqli.allow_persistent	On	On
mysqli.default_host	no value	no value
mysqli.default_port	3306	3306

Abbildung 61: php-Testseite (im Browser)

7.6 Administrationssoftware für die Datenbank

Die Datenbank-Frontend **phpMyAdmin** ist eine freie Webanwendung zur Administration von MySQL-Datenbanken (MySQL und MariaDB). Die Software ist in PHP programmiert. Die meisten Funktionen können ausgeführt werden, ohne selbst SQL-Anweisungen zu schreiben. Möglich ist das Anlegen, Bearbeiten, Löschen von

- Datenbanken
- Benutzern
- Tabellen und Tabellendefinitionen
- Verknüpfungen zwischen den Tabellen
- Datensätzen

PhpMyAdmin ist unter der GNU General Public License lizenziert und ist auch in vielen Linux-Distributionen enthalten. Das Tool ist weit verbreitet und wird unter anderem von großen Webhosting-Providern verwendet.

phpMyAdmin installieren

Die Installation geschieht durch Downloaden, Entpacken und Verschieben aller Dateien in den Webordner des Servers.

Update_V2020.10: Installation mit `sudo apt install phpmyadmin`

Dadurch werden alle Parameter abgefragt und konfiguriert.

```
$ sudo apt install phpmyadmin
```

PhpMyAdmin wird im System im Pfad `/usr/share/phpmyadmin` installiert. Weitere Dateien befinden sich an anderen Stellen des Systems:

```
pi@h-raspi2:/$ sudo find / -name phpmyadmin
/var/lib/apache2/conf/enabled_by_maint/phpmyadmin
/var/lib/phpmyadmin
/var/lib/mysql/phpmyadmin
/usr/share/doc/phpmyadmin
/usr/share/lintian/overrides/phpmyadmin
/usr/share/phpmyadmin
/usr/share/dbconfig-common/data/phpmyadmin
/usr/share/dbconfig-common/scripts/phpmyadmin
/usr/share/doc-base/phpmyadmin
/etc/phpmyadmin
```

Abbildung 62: Installationspfade für phpmyadmin

Anmeldung und Fehlerkorrektur

Nun ist im Unterverzeichnis **phpmyadmin** die Startseite vom Browser aus erreichbar. Mit dem Benutzer **root** und dem bereits bei der Installation festgelegten MariaDB-Passwort ist das Einloggen an der Datenbank möglich (siehe Abbildung 63).

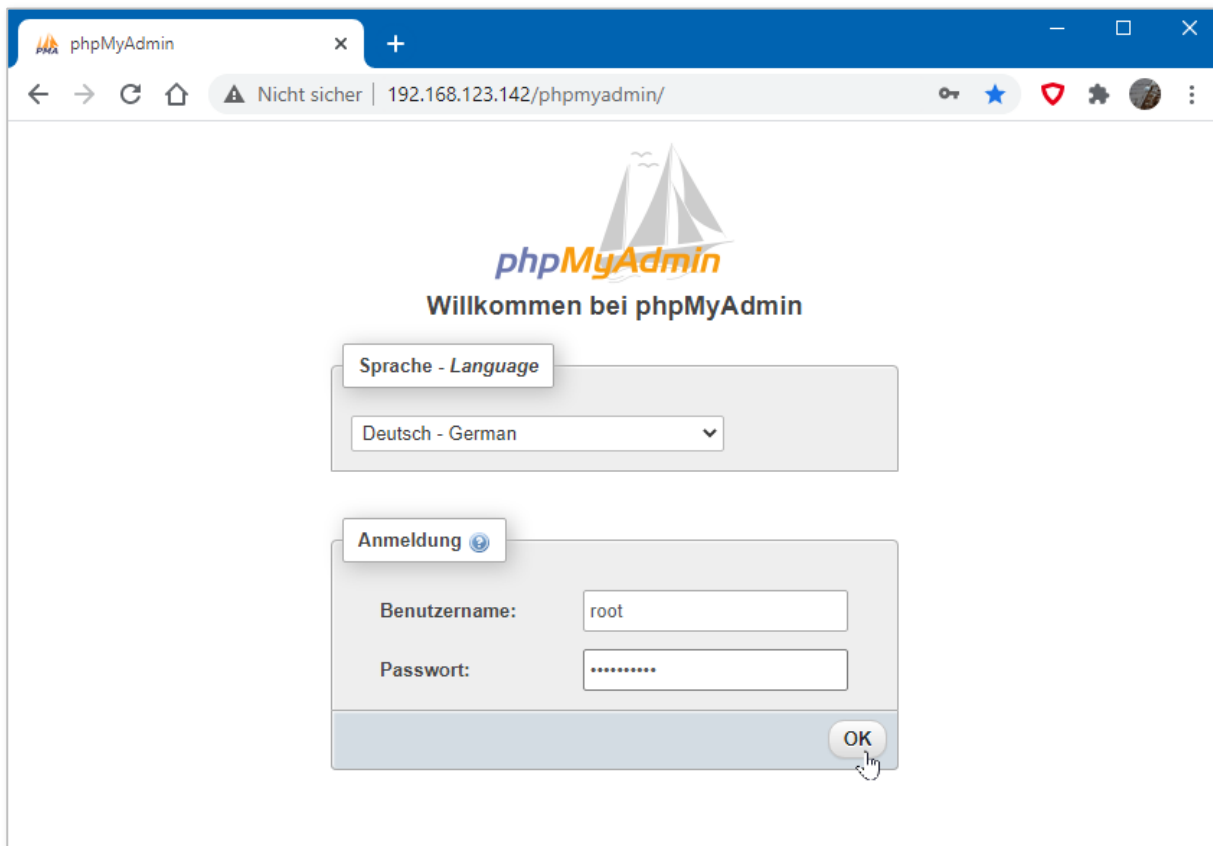


Abbildung 63: Anmelden an phpMyAdmin

Sollte ein Zugriffsfehler auftreten, liegt dies meist am fehlenden oder fehlerhaften **root**-Passwort. Dies kann durch Anmelden an der Kommandozeile und Neusetzen des Passworts in **mysql** behoben werden.

```
pi@h-raspi2:/var/www/html $ mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 57
Server version: 10.3.15-MariaDB-1 Raspbian testing-staging

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> update mysql.user set password=password('') where user='root';
Query OK, 1 row affected (0.110 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.018 sec)

MariaDB [(none)]> exit
Bye
```

Abbildung 64: root-Passwort in mysql setzen

Updaten von phpMyAdmin

Sollte phpMyAdmin in einer älteren Version installiert sein, kann folgendermaßen leicht ein Update vorgenommen werden:

- Neuste Version von der phpMyAdmin-Website herunterladen
- Verbindung zum Raspberry Pi über WinSCP aufbauen
- bestehenden Ordner umbenennen z.B. in **phpmyadmin-alt**
- neuen Ordner erstellen mit dem bisher bestehenden Namen z.B. **phpmyadmin**
- ZIP-Datei öffnen und alle enthaltenen Dateien in den neu erstellten Ordner hochladen

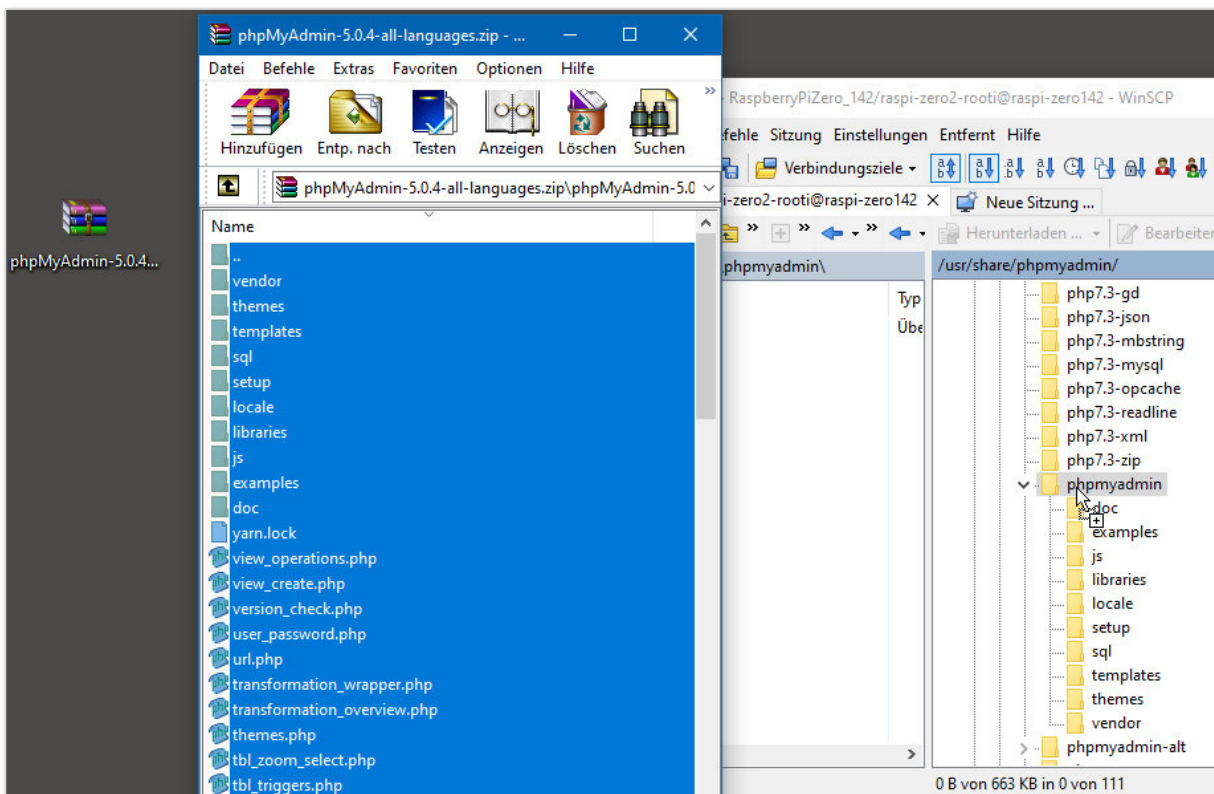


Abbildung 65: phpMyAdmin updaten

Der ehemalige Ordner sollte erst gelöscht werden, wenn die Anwendung in der neuen Version getestet ist und ordnungsgemäß funktioniert.

Weitere Einstellungen

Nachdem alle Dateien hochgeladen sind erfolgt der nächste Login. Es erscheinen aufgrund des Versionsprungs u.U. weitere Fehlermeldungen:

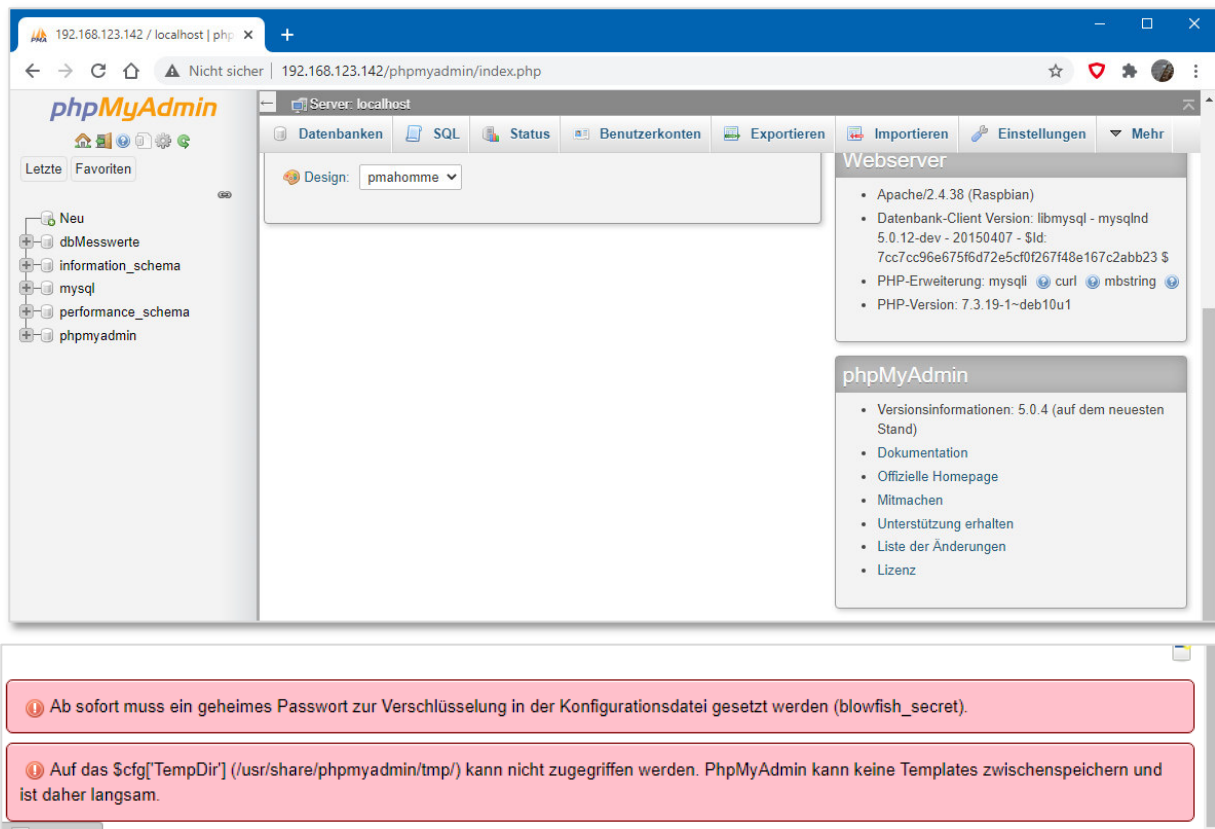


Abbildung 66: Fehlermeldungen bei phpMyAdmin

Erläuterung und Lösung

Bei neuen Versionen muss für phpMyAdmin ein geheimes Passwort gesetzt werden, das einmalig in die Konfigurationsdatei `usr/share/phpmyadmin/config.sample.inc.php` geschrieben wird. Zur Erzeugung des hash-Wertes wird ein blowfish-Hash-Generator im Internet verwendet. Das Passwort dient zur Cookie-Verwaltung, muss vom Besucher oder Programmierer nicht eingegeben werden. Im Beispiel unten erfolgt das Eintragen über WinSCP Remotezugriff und Editieren mit Notepad++.

Im Bereich `$cfg['blowfish_secret'] = '.....';` wird das 32Bit Passwort platziert (Abbildung 67).

```

8  * or at <https://docs.phpmyadmin.net/>.
9  *
10 * @package PhpMyAdmin
11 */
12 declare(strict_types=1);
13
14 /**
15  * This is needed for cookie based authentication to encrypt password in
16  * cookie. Needs to be 32 chars long.
17  */
18 $cfg['blowfish_secret'] = 'aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa';
19 /* YOU MUST FILL IN THIS FOR COOKIE AUTH! */

```

Abbildung 67: Konfiguration des blowfish-Passworts

Die Beispiel-Konfigurationsdatei muss zum Schluss umbenannt werden, um sie zu aktivieren.

```
$ sudo mv usr/share/phpmyadmin/config.sample.inc.php
usr/share/phpmyadmin/config.inc.php
```

Eine weitere Fehlermeldung betrifft den Ordner **tmp** im **phpmyadmin**-Verzeichnis. Dieser Ordner fehlte in der ZIP-Datei und muss erstellt und mit korrekten Rechten versehen werden.

```
$ sudo mkdir usr/share/phpmyadmin/tmp
$ sudo chmod 775 /usr/share/phpmyadmin/tmp
```

phpMyAdmin – Bildschirmaufbau

Die Funktionen von phpMyAdmin sind sehr umfangreich. Hier wird lediglich der Startbildschirm erläutert. Andere, datenbankspezifische Bedienungshinweise werden an anderer Stelle veröffentlicht.

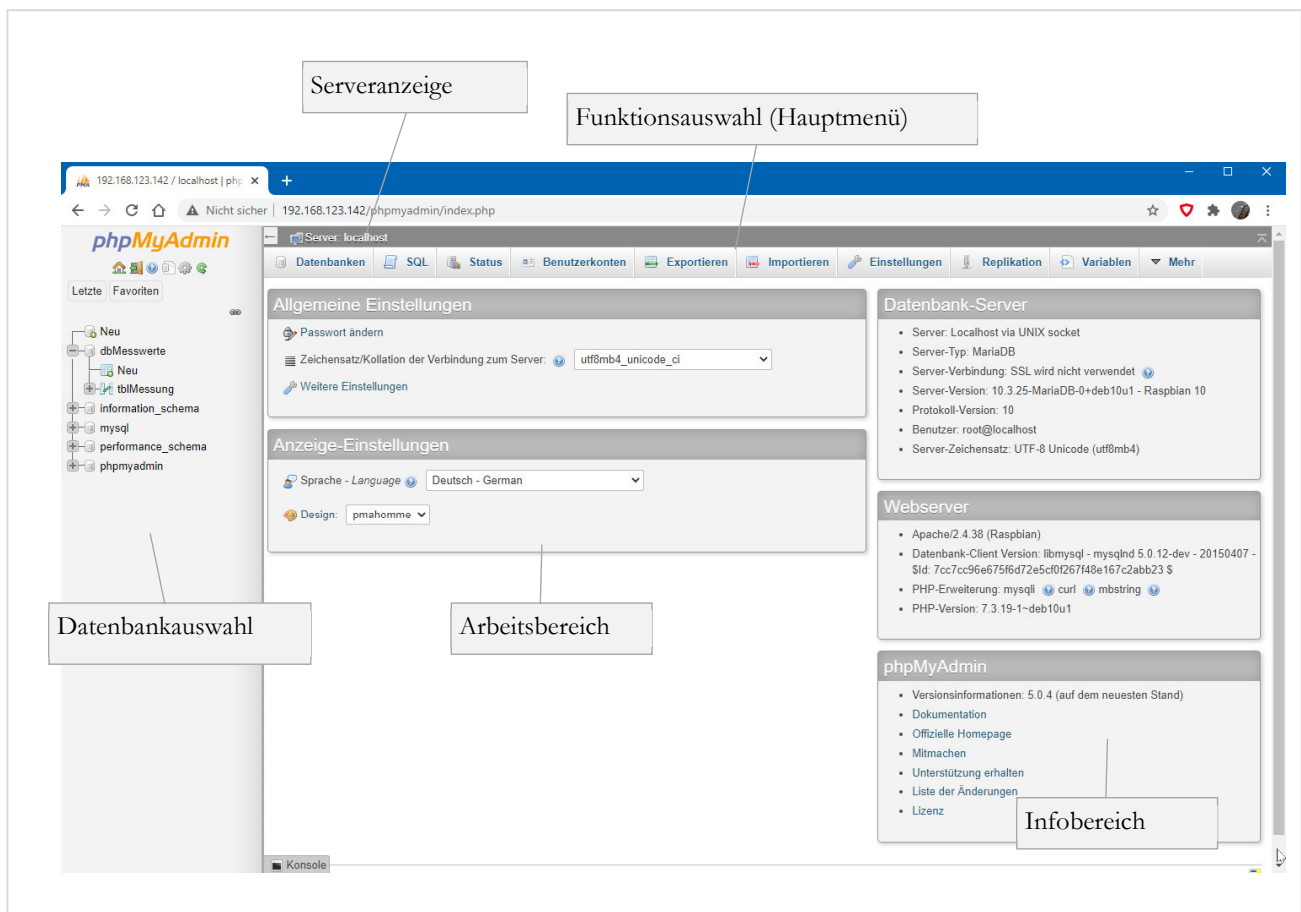


Abbildung 68: phpMyAdmin - Arbeitsbildschirm

8 Datenbank für Wetterdaten erstellen

Zur Vorbereitung der Datenablage von Sensor-Messwerten wird nun mit Hilfe von phpMyAdmin eine Datenbank auf dem Raspberry Zero erstellt. Folgende Anforderungen soll die Datenbank erfüllen:

- Speicherung von Messzeitpunkt und die Messwerte für verschiedene Werte.
- Speicherung der Besuche der Webseite
- Die Namenskonvention orientiert sich an der Ungarischen Notationsweise

Datenbanken: **dbDatenbank**

Tabellen: **tblTabelle**

Feldnamen: Kleinbuchstaben-Präfix für Tabellenanfang

Da es sich lediglich um das Speichern von Messwerten handelt, reicht eine Tabelle aus, es müssen also auch keine Schlüsselfelder und Verknüpfungen angelegt werden.

Die folgenden Bildschirmausschnitte zeigt die Vorgehensweise.

8.1 Neue Datenbank anlegen

Das Anlegen einer Datenbank wird mit phpMyAdmin unter dem Hauptreiter Datenbanken durchgeführt.

Die neue Datenbank erhält den Namen **dbMesswerte**. Die Kollation definiert die Sortierreihenfolge in Listen. Der **utf8mb4**-Zeichensatz ist hier die beste Wahl.

Alternative:

utf8mb4_german2_ci



8.2 Tabelle erzeugen

In der Datenbank muss mindestens eine Tabelle vorhanden sein. Für die Messung erhält diese den Namen **tblMessung**. Bei der Benennung wird hier immer Singular verwendet.



8.3 Felder erstellen

Die zu erstellenden Felder sind abhängig von den zu speichernden Messdaten. Das Beispiel zeigt, wie mehrere Felder in einem Arbeitsgang mit phpMyAdmin angelegt werden.



Abbildung 69: Datenfelder anlegen

m_id (Zähler, Primärschlüssel), **m_messzeitpunkt** (Datum und Zeit der Messung), **m_temp** (Temperatur), **m_druckrel** (relativer Luftdruck), **m_druckabs** (absoluter Luftdruck), **m_feucht** (relative Luftfeuchtigkeit)

Im Datenbankdesigner sieht die durchgeführte Konstruktion so aus:

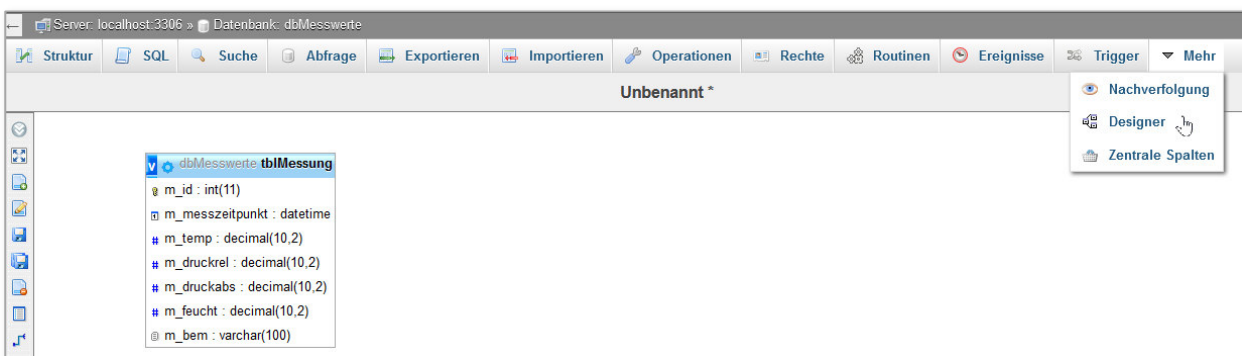


Abbildung 70: Datenbank-Designer

8.4 Testdaten eingeben

Zu Testzwecken werden zwei Daten manuell in die Felder eingegeben:



Abbildung 71: Daten einfügen

Die abschließende Anzeige ergibt folgende Liste:

		m_id	m_messzeitpunkt	m_temp	m_druckrel	m_druckabs	m_feucht
<input type="checkbox"/>	Bearbeiten	2	2019-10-12 18:00:00	21	986	1017	62
<input type="checkbox"/>	Bearbeiten	6	2019-10-12 19:30:00	19	986	1017	72

Abbildung 72: Datensätze anzeigen

8.5 User für den Datenbankzugriff

Server: localhost

Datenbanken SQL Status Benutzerkonten Exportieren Importieren

Benutzerkonto hinzufügen

Anmeldeinformation

Benutzername: Textfeld verwenden:

Hostname: Jeder Host

Passwort: Textfeld verwenden: Strength: Stark

Wiederholen:

Authentifizierungs Plugin: Native MySQL-Authentifizierung

Passwort generieren:

Global Datenbank Passwort ändern Anmeldeinformation

Rechte ändern:

Benutzerkonto

Datenbankspezifische Rechte

Datenbank Rechte GRANT Tabellenspezifische Rechte Aktion

keine

dbMesswerte
mysql
phpmyadmin

Rechte zu folgender Datenbank(en) hinzufügen:

Datenbankspezifische Rechte ☒ Alle auswählen

Hinweis: MySQL-Rechte werden auf Englisch angegeben.

☒ Daten ☒ Struktur ☐ Administration

☒ SELECT
☒ INSERT
☒ UPDATE
☒ DELETE

☒ CREATE
☒ ALTER
☒ INDEX
☒ DROP
☒ CREATE TEMPORARY TABLES
☒ SHOW VIEW
☒ CREATE ROUTINE
☒ ALTER ROUTINE
☒ EXECUTE
☒ CREATE VIEW
☒ EVENT
☒ TRIGGER

☐ GRANT
☐ LOCK TABLES
☐ REFERENCES

Datenbank dbMesswerte: Struktur 1

9 Serverüberblick

Nun sind alle wichtigen Serverdienste implementiert. PhpMyAdmin zeigt die aktuellen Versionen nach dem Login übersichtlich an.



Abbildung 73: aktuelle Serverversionen

Serverupdates durchführen

Alle Serverdienste sollten auf aktuellem Stand gehalten werden. Grundsätzlich geschieht dies über die Nutzung der Quellen.

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

10 Messungen steuern

Im nächsten Schritt wird angestrebt, eine möglichst einfach zu handhabenden Messzyklus zu erstellen, der alle Daten regelmäßig in die Datenbank schreibt und auf der Website möglichst aktuell anzeigt.

10.1 Planung

Folgende Schritte zur Umsetzung der Datenspeicherung und -aufbereitung sind vorgesehen:

- Wetterbild und alle Messdaten für die Datenbank vier Mal pro Tag erzeugen (0:00 Uhr, 6:00 Uhr, 12:00 Uhr, 18:00 Uhr)
- Messwerte für aktuelle Anzeige alle 60 Minuten einlesen und anzeigen (0:30, 1:30, 2:30 usw... bis 23:30)
- Modularisierung der verschiedenen Skripte in eine einheitliche Python-Umgebung, um den Code leichter lesbar und wartbar zu machen

10.2 Python-Skripte programmieren

Damit die Zeitsteuerung mit dem LINUX-Daemon CRON erfolgen kann, sind zwei getrennte Skripte erforderlich. Zusätzlich soll die Abfrage der Messsensoren in einem separaten Python-Modul stehen.

- **messvorgang.py** (Modul mit Sensorzugriffen)
- **sensorabfrage-stunde.py** (Stundenmessung zur Minute 30)
- **sensorabfrage-tag4x.py** (vier Tagesmessungen mit Speicherung in der Datenbank)

Skript zur Sensorabfrage

Die Sensorabfrageskripte werden von CRON zum festgelegten Zeitpunkt gestartet. Diese Skripte rufen jeweils das Messmodul auf und erhalten von diesem die angeforderten Messwerte zurück.

Das Wetterbild wird direkt im Messskript vier Mal pro Tag erstellt und als **jpg**-Datei in den vorgegebenen Pfad des Webservers gespeichert. Von dort lädt die Webseite das Bild und stellt es dar. Die aktuellen Wetterdaten werden in eine **html**-Datei geschrieben, die auf die Haupt-Webseite einfach eingebunden wird. Das Auslesen der in der Datenbank gespeicherten Wetterdaten erfolgt auf einer separaten Webseite mit einem php-Skript.

Das Messkript besteht aus mehreren Teilen, die hier nacheinander dargestellt sind. Erläuterungen befinden sich im Kommentarbereich.

Kopfbereich, Bibliotheken, Variablen

```

1  #!/usr/bin/python
2  #WETTERSTATION 2 (h-raspi2)
3  #--- Messmodul zum Einbinden V1.20.1121 -----
4  #-----
5  #Dient zur Messung, Kalibrierung und Berechnung
6  #DATEiname: var\www\cgi-bin\wetterstation2\messvorgang.py
7  #erstellt 09.11.2020 (HK) - zuletzt geaendert: 21.11.2020
8  # Rueckgabewerte der Sensoren
9  #   BOSCH-Sensor BMP280 kalibrierte Werte
10 #   Adafruit-Sensor ADT22
11 #   ---Temperatur beider Sensoren
12 #   ---BMP280: Luftdruck nach Hoehenformel berechnet (abs. und rel.)
13 #   ---ADT22 : rel. Luftfeuchtigkeit
14 #   ---aktueller Messzeitpunkt
15 #-----
16 def sensorberechnung():
17     #Adafruit-Bibliothek einbinden
18     import Adafruit_DHT
19     #I2C-Bus Bibliothek einbinden
20     import smbus
21     #Zeitfunktionen
22     import time
23     #Weitere Bibliotheken
24     import subprocess
25     import sys
26     #globale Variablen
27     global templ
28     global druckabs
29     global druckrel
30     global feuchte
31     global temp2
32     global aktuellesDatum
33     global aktuelleZeit
34
35     #Aktueller Messzeitpunkt
36     aktuellesDatum=time.strftime('%d.%m.%Y - %H:%M')
37     aktuelleZeit=time.strftime('%H')
38
39     # Adresse des I2C Bus
40     BUS = 1
41     # BMP280 Adresse, 0x76 oder 0x77
42     BMP280ADDR = 0x76
43     # Meereshoehe der Wetterstation 248m Boden -- 255 m Montageort
44     ALTITUDE = 255

```

Sensor DHT22 auslesen

```

45
46 #SensorwerteDHT22: Adafruit_DHT.DHT11, Adafruit_DHT.DHT22 oder Adafruit_DHT.AM2302
47 strSensor=Adafruit_DHT.DHT22
48 #Pin am RaspiZero
49 intPin = 4
50
51 #Sensor DHT22 auslesen
52 lngFeuchte, lngTemp = Adafruit_DHT.read_retry(strSensor, intPin)
53 #Werte Testen
54 if lngFeuchte is not None and lngTemp is not None:
55     print('Temperatur={0:0.1f}*C  Feuchtigkeit={1:0.1f}%'.format(lngTemp, lngFeuchte))
56 else:
57     print('Keine Messdaten erhalten - bitte nochmals versuchen...')
58 print(aktuellesDatum)
59

```

Sensor BMP280 auslesen

```

60 # Fuer Sensor BMP280 -- I2C Bus einlesen
61 bus = smbus.SMBus(BUS)
62
63 # Temperatur kalibrieren (Array)
64 T = [0, 0, 0];
65 # Druck kalibrieren (Array)
66 P = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0];
67
68 # Kalibrierungsdaten aus dem Sensor einlesen
69 # (Daten aus Adresse 0x88, 24 bytes)
70 data = bus.read_i2c_block_data(BMP280ADDR, 0x88, 24)
71
72 # Temperatur-Koeffizienten
73 T[0] = data[1] * 256 + data[0]
74 T[1] = data[3] * 256 + data[2]
75 if T[1] > 32767:
76     T[1] -= 65536
77 T[2] = data[5] * 256 + data[4]
78 if T[2] > 32767:
79     T[2] -= 65536
80
81 # Druck-Koeffizienten
82 P[0] = data[7] * 256 + data[6];
83 for i in range (0, 8):
84     P[i+1] = data[2*i+9]*256 + data[2*i+8];
85     if P[i+1] > 32767:
86         P[i+1] -= 65536
87
88 # Kontroll-Messregister auswaehlen (Adresse 0xF4)
89 # In das Register 0b00100111 schreiben (=0x27)
90 # 0x27 --> pressure/temperature oversampling rate = 1, normal mode
91 bus.write_byte_data(BMP280ADDR, 0xF4, 0x27)
92
93 # Konfigurationsregister auswaehlen, (Adresse 0xF5)
94 # beschreiben mit 0xA0: standby time = 1000 ms
95 bus.write_byte_data(BMP280ADDR, 0xF5, 0xA0)
96
97 #Kurze Wartezeit
98 time.sleep(1.0)
99

```

Daten konvertieren und umrechnen

```

100 # Daten lesen aus 0xF7 (Druck und Temp., 6 Bytes)
101 data = bus.read_i2c_block_data(BMP280ADDR, 0xF7, 6)
102
103 # Konvertierung der Daten in je 20-Bit-Werte
104 # Speicherung in je einer Variablen
105 adc_p = (data[0] << 12) | (data[1] << 4) | (data[2] >> 4)
106 adc_t = (data[3] << 12) | (data[4] << 4) | (data[5] >> 4)
107
108 # Temperatur Offset Berechnung aus Messdaten und Kalibrierungsdaten
109 temp1 = ((adc_t)/16384.0 - (T[0])/1024.0)*(T[1]);
110 temp3 = (adc_t)/131072.0 - (T[0])/8192.0;
111 temp2 = temp3*temp3*(T[2]);
112 temperature = (temp1 + temp2)/5120.0
113
114 # Luftdruck Offset Berechnung aus Messdaten und Kalibrierungsdaten
115 press1 = (temp1 + temp2)/2.0 - 64000.0
116 press2 = press1*press1*(P[5])/32768.0
117 press2 = press2 + press1*(P[4])*2.0
118 press2 = press2/4.0 + (P[3])*65536.0
119 press1 = ((P[2])*press1*press1/524288.0 + (P[1])*press1)/524288.0
120 press1 = (1.0 + press1/32768.0)*(P[0])
121 press3 = 1048576.0 - (adc_p)
122 if press1 != 0:
123     press3 = (press3 - press2/4096.0)*6250.0/press1
124     press1 = press3*press3*(P[8])/2147483648.0
125     press2 = press3*(P[7])/32768.0
126     pressure = (press3 + (press1 + press2 + (P[6]))/16.0)/100
127 else:
128     pressure = 0
129 # Druck relativ zu Seehoehe NN berechnen
130 pressure_nn = pressure/pow(1 - ALTITUDE/44330.0, 5.255)
131

```

Messwertewerte zurückgeben

```

131
132 # -----
133 # -- Rueckgabewerte -----
134 #BOSCH_Sensor-Messwerte
135 temp1=temperature
136 druckabs=pressure
137 druckrel=pressure_nn
138 #Adafruit-Sensor-Messwerte
139 temp2=lngTemp
140 feuchte=lngFeuchte
141 #-----

```

Skripte - Messungen starten

Zur stündlichen Messung löst das Skript **sensorabfrage-stunde.py** den Messvorgang aus und zeigt zunächst die Werte auf der LINUX-Konsole an. Außerdem erzeugt es im festgelegten Ordner **sensordaten** eine **html**-Datei, die später auf die Webseite eingebunden wird.

Messvorgang

```

1  #!/usr/bin/python
2  #WETTERSTATION 2 (h-raspi2)
3  #--- Messmodul zum Einbinden V1.20.1121 -----
4  #-----
5  #DATEINAME: var\www\cgi-bin\wetterstation2\sensorabfrage-stunde.py
6  #erstellt 09.11.2019 (HK) - zuletzt geaendert: 21.11.2020
7  #---Daten messen und speichern in html-Datei sensoraktuell.html)
8  #--laeuft von 0-24 Uhr zu Minute 30, d.h. 0:30, 1:30, ... 23:30
9  #---Dateipfad zum Speichern.: var\www\html\sensordaten\sensoraktuell.html
10 #--Wetterbild erstellen.....: var\www\html\wetterbilder\wetterbildaktuell.jpg
11 #-----
12
13 #Messmodul einbinden (ACHTUNG Dateiname ist messvorgang.py)
14 import messvorgang
15 #weitere Module einbinden
16 import picamera
17 import time
18 import os
19
20 #Messen starten und Werte abfragen
21 messvorgang.sensorberechnung()
22 print ('Zeitpunkt.....:', messvorgang.aktuellesDatum)
23 print ('Temperatur 1.....:', messvorgang.temp1)
24 print ('Temperatur 2.....:', messvorgang.temp2)
25 print ('absoluter Luftdruck.....:', messvorgang.druckabs)
26 print ('relativer Luftdruck.....:', messvorgang.druckrel)
27 print ('relative Luftfeuchtigkeit...:', messvorgang.feuchte)

```

html-Datei für die Webseite erstellen

```

28
29 #Durchschnitt der beiden Temperatursensoren beerchnen
30 tempschnitt=(messvorgang.temp1+messvorgang.temp2)/2
31
32 #Aktuelle Wetterdaten als Tabelle in eine html-Datei schreiben
33 file = open("/var/www/html/sensordaten/sensoraktuell.html","w")
34 strHtml="<p class='textklein'>Aktuelle Sensordaten:<br>" + messvorgang.
    aktuellesDatum + "</p><table><tr><td
    class='ueberschrift'>Temperatur</td></tr><tr><td
    class='wetter'>{0:.1f}*C</td></tr><tr><td
    class='ueberschrift'>Luftfeuchtigkeit</td></tr><tr><td
    class='wetter'>{1:.1f} %</td></tr><tr><td class='ueberschrift'>rel.
    Luftdruck in hPa</td></tr><tr><td
    class='wetter'>{2:.1f}</td></tr></table>".format(tempschnitt,messvorgang
    .feuchte,messvorgang.druckrel)
35 file.write(strHtml)
36 file.close()

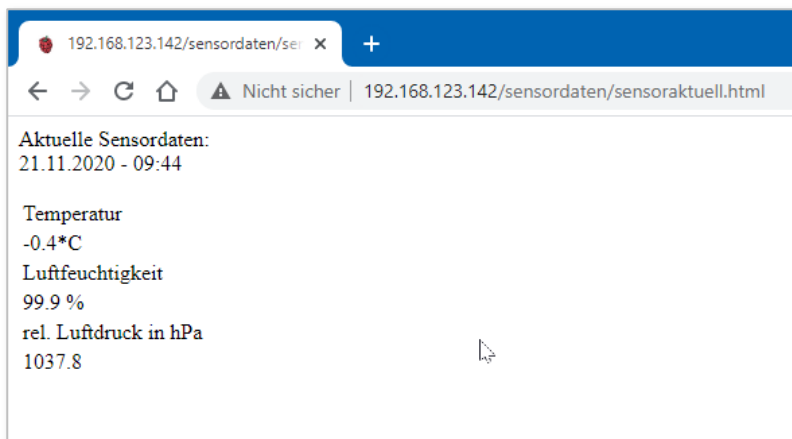
```

erster Funktionstest

Die Funktion des Skripts wird nun auf der Raspberry – Konsole getestet.

```
pi@h-raspi2:/var/www/cgi-bin/wetterstation2 $ sudo python3 sensorabfrage-stunde.py
Temperatur=-0.5*C  Feuchtigkeit=99.9%
21.11.2020 - 09:44
Zeitpunkt.....: 21.11.2020 - 09:44
Temperatur 1.....: -0.3753735497535672
Temperatur 2.....: -0.5
absoluter Luftdruck.....: 1006.7974156557439
relativer Luftdruck.....: 1037.7865154546662
relative Luftfeuchtigkeit...: 99.9000015258789
pi@h-raspi2:/var/www/cgi-bin/wetterstation2 $
```

Ermittelte Messwerte kommen von den Sensoren zurück und erscheinen in der gewünschten Form. Die vom Skript erstellte **html**-Datei wird durch Zugriff per Browser angezeigt.



10.3 Wetterbild erstellen

Mit der Raspi-Kamera soll in stündlichen Abständen das aktuelle Wetterbild erstellt werden. Zunächst erfolgt nochmals ein Test der Kamerafunktion über die bekannten Kommandozeilenkommandos (siehe 4.1, Seite 14). Anschließend soll die Kamerafunktion in das Python-Skript der Sensor-Stundenmessungen eingebunden werden.

Allgemeines

Das Raspberry-Kamera-Modul wird über die beiden Programme **raspistill** (für Bilder) und **raspivid** (für Videos) angesprochen. Hierzu gibt es zahlreiche Optionen. **raspivid** speichert Videos im H264-Format ab, meist muss die Datei in ein anderes Format konvertiert werden (z.B. mit dem Tool **gpac**).

Kamerateest

Das Kommandozeilenkommando **raspistill** erzeugt ein jpg-Bild im angegebenen Pfad. Das Wetterbild soll nun schon in den endgültigen Speicherpfad des Webserver abgelegt werden.

```
$ raspistill -o /var/www/html/wetterbilder/wetterbildaktuell.jpg
```

Zur Kontrolle erfolgt ein Zugriff auf den Raspberry Pi Zero über WinSCP und die Anzeige mit einem Bildverarbeitungsprogramm.

Das Bild muss aufgrund rechtlicher Vorschriften beschnitten werden und zeigt ausschließlich die Landschaft und den aktuellen Himmel.



Abbildung 74: Test-Wetterbild

Für die Einbindung in Python-Skripts wird das Modul Picamera benötigt. Es implementiert fast alle Optionen der Kamera.

Das Python Kameramodul

Zunächst wird das Pythonmodul auf dem Raspberry Pi Zero installiert.

```
$ sudo apt-get update
```

```
$ sudo apt-get install python-picamera python3-picamera
```

Die umfangreichen Konfigurationsmöglichkeiten der Library sind unter <https://picamera.readthedocs.org> beschrieben.

Für das Erzeugen des Bildes für die Wetterstation wird ein einfaches Skript benötigt. Das Bild muss lediglich in angepasster Auflösung erstellt und anschließend beschnitten werden.

Um die Funktion zu testen, wird zunächst eine separate Python-Datei erstellt und zunächst im Skript-Verzeichnis gespeichert.

```
1 #Module importieren
2 import picamera
3 import time
4 #neues Kameraobjekt
5 cam = picamera.PiCamera()
6 #Parameter einstellen
7 cam.resolution = (640, 480)
8 #Bild machen
9 cam.start_preview()
10 time.sleep(5)
11 cam.stop_preview()
12 cam.capture('bild.jpg')
13 cam.close()
```

Abbildung 75: Testskript für die Kamera

```
$ sudo python3 kameratest.py
```

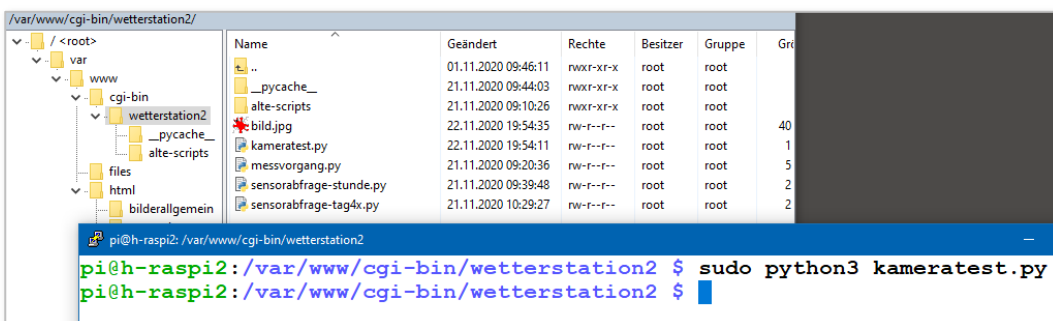


Abbildung 76: Kameratest mit Python-Skript

Um eine Fehlerbehandlung einzufügen und das Bild im gewünschten Zielverzeichnis abzulegen, wird das Skript ergänzt und in das bestehende Messskript **sensorabfrage-stunde.py** integriert. Außerdem muss das Bild beschnitten werden, um keine Rechtsverletzungen auszulösen.

```
37 #-----
38 #Wetterbild machen
39 #-----
40 zieldatei = '/var/www/html/wetterbilder/wetterbildaktuell.jpg'
41 if os.path.isfile(zieldatei): # falls Dabei existiert
42     os.remove(zieldatei)
43 #neues Kameraobjekt
44 cam = picamera.PiCamera()
45 try:
46     #Parameter einstellen
47     cam.resolution = (720, 480)
48     #Bild machen
49     cam.start_preview()
50     time.sleep(5)
51     cam.stop_preview()
52     #Bild beschneiden Breite 100%, Höhe 75%
53     cam.zoom=(0,0,1,0.75)
54     cam.capture(zieldatei)
55
56 except:
57     print("Kamera-Fehler")
58 finally:
59     cam.close()
60
```

Abbildung 77: Integration in das Messskript

11 Messdaten in die Datenbank ablegen

Das Skript zur Ablage der Messwerte in der Datenbank und zum Erstellen des Wetterbildes muss nun separat angelegt werden. Der Eintrag der Messdaten soll vier Mal pro Tag jeweils um 0:00 Uhr, 6:00 Uhr, 12:00 Uhr und 18:00 Uhr erfolgen. Dadurch wird eine kontinuierliche Mess-Statistik erreicht.

Um MariaDB (mysql) in python verwenden zu können, muss der mysql-connector nachinstalliert werden.

```
$ sudo apt-get install python-mysqldb
$ sudo apt install python3-pip
$ sudo pip3 install mysql-connector-python
```

Die folgende Abbildung zeigt den geplanten Vorgang von der Messung bis zur Anzeige im Browser.

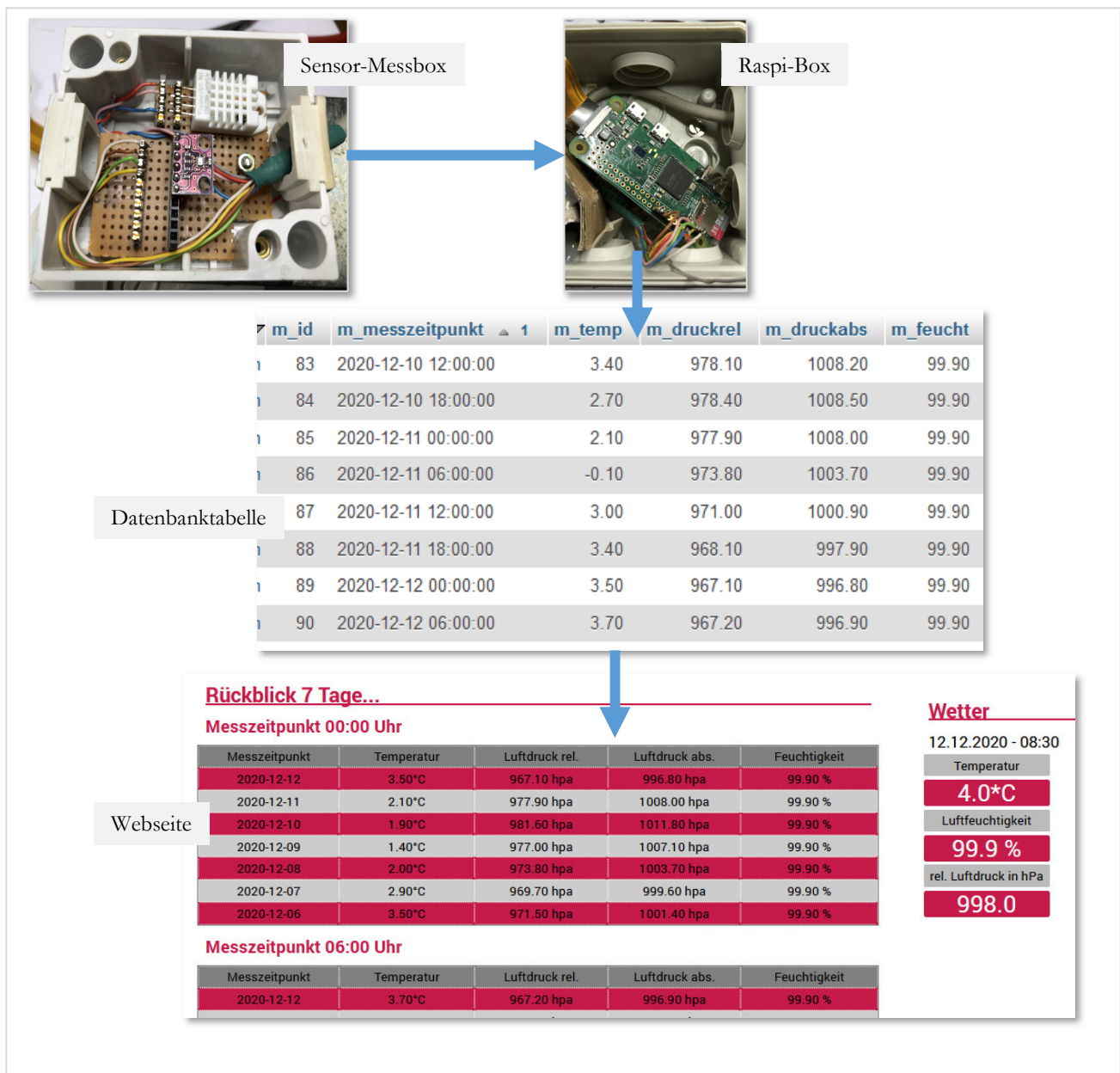


Abbildung 78: Messvorgang

Test der Datenbankverbindung

Der Zugriff auf MariaDB über ein Pythonscript muss zunächst in kleinen Schritten programmiert werden, um die einzelnen Schritte auch nachhaltig zu verstehen. Hierzu dienen kleine Testprogramme, z.B. **db_test_connect.py**. Diese Programme sollen später als Scriptquelle für das eigentliche Abfrageprogramm der Sensoren verwendet werden.

Abbildung 79 zeigt schematisch, wie aus einem Python-Script auf die Datenbank zugegriffen werden kann.

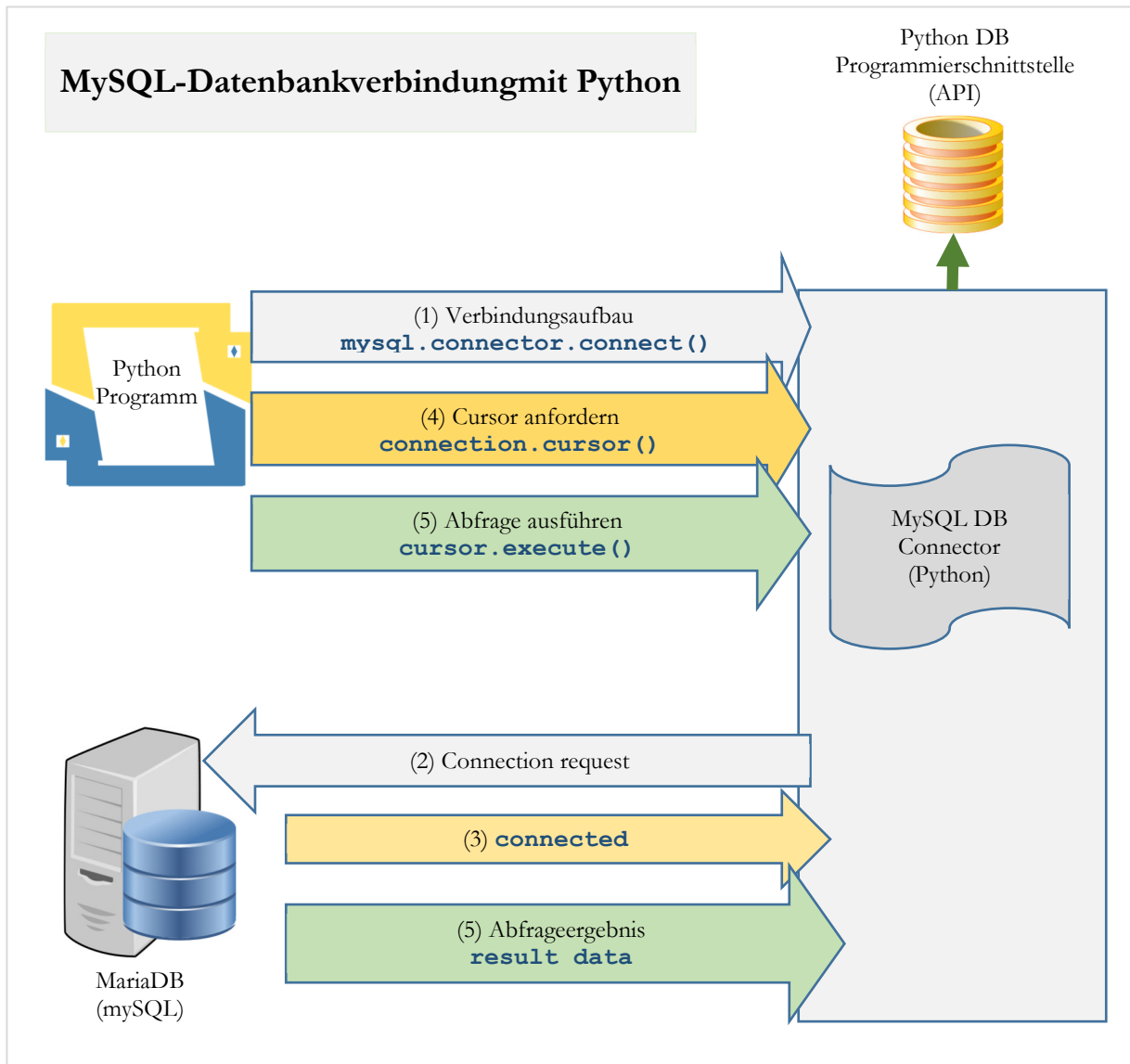


Abbildung 79: Datenbankverbindung an ein Python-Script¹²

Das Verbindungstestprogramm wird aufgerufen und zeigt die entsprechenden Meldungen an.

```
$ sudo python3 dbtest_connect.py
```

```
('Verbunden mit MySQL-Server Version: ', u'5.5.5-10.3.15-MariaDB-1')
('...mit folgender Datenbank steht die Verbindung: ', (u'dbMesswerte',))
MySQL-Verbindung geschlossen
```

¹² Grafik bearbeitet nach <https://pynative.com/python-mysql-database-connection/>

Die folgende Grafik zeigt den Vorgang der Datensatzspeicherung. Um den Datenbankzugriff zu testen, wird das vorhandene Testskript mit dem Messskript kombiniert.

Damit die Sensordaten in der Tabelle gespeichert werden, muss der SQL-Befehl **INSERT INTO** mit den Messwerten an die Datenbank gesendet werden.

Den kompletten Datenzugriff zeigt gesamte Script (siehe Folgeseiten).

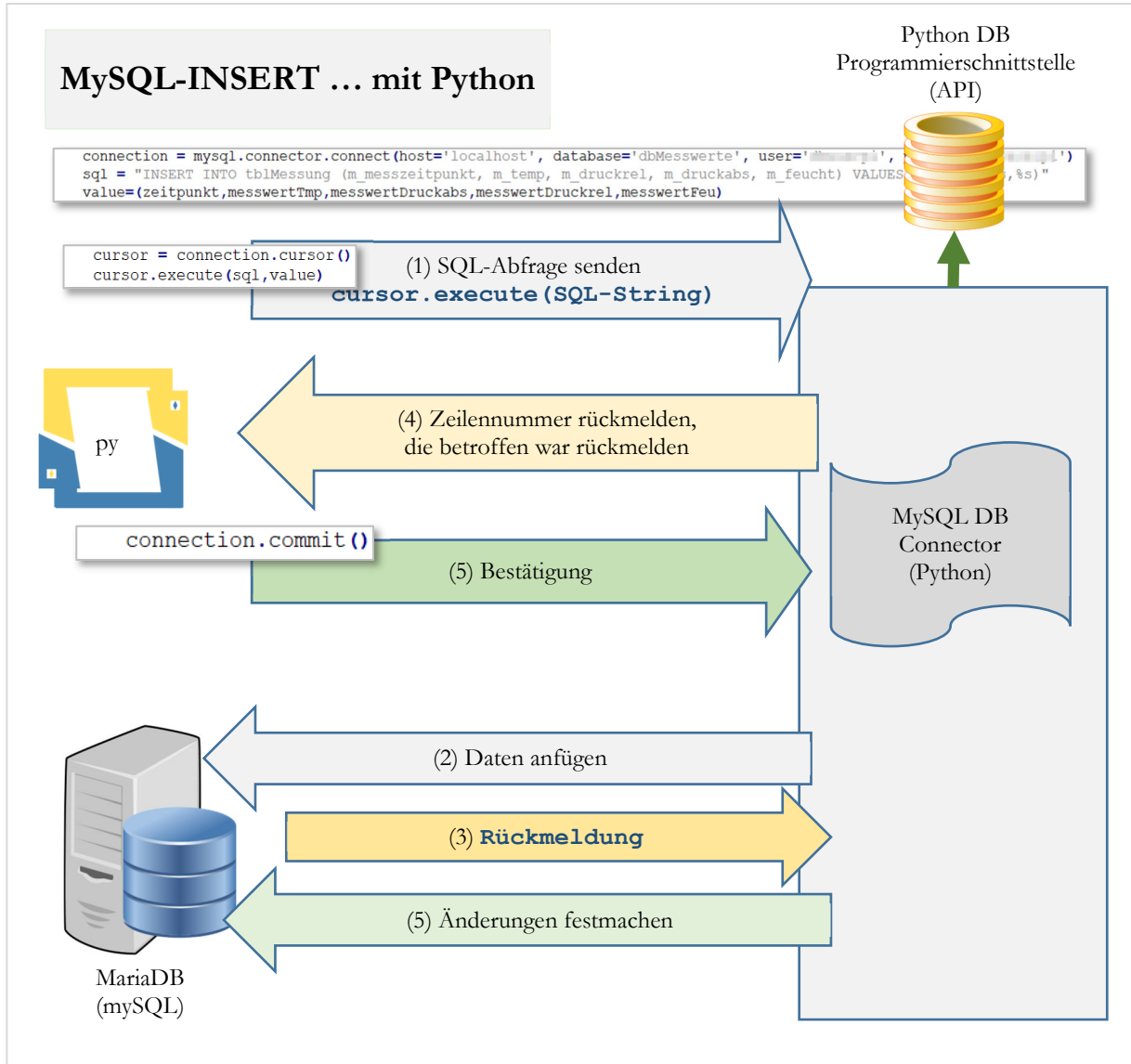


Abbildung 80: Datenbankanbindung an ein Python-Skript¹³

¹³ Grafik bearbeitet nach <https://pynative.com/python-mysql-insert-data-into-database-table/>

Gesamtes Python-Script mit Datenbankzugriff

Nun gilt es, die Erkenntnisse aus den Tests zur konkreten Abspeicherung der Messwerte der Sensoren zu nutzen. Zunächst werden die Messwerte zusammengestellt und formatiert. Der **try**-Bereich baut die Verbindung zur Datenbank auf, stellt das SQL-Statement zusammen und übergibt es MariaDB. Fehlermeldungen werden bei Bedarf auf der Console ausgegeben. Abschließend wird die Verbindung geschlossen.

```

34 try:
35     #Verbindung zur Datenbank aufbauen - bei Fehler-->exception
36     connection=mysql.connector.connect(host='localhost',database='dbMesswerte',user='root',password='')
37     sql="INSERT INTO tblMessung(m_messzeitpunkt, m_temp, m_druckrel, m_druckabs, m_feucht) VALUES (%s,%s,%s,%s,%s)"
38     value=(zeitpunkt,messwertTmp, messwertDruckabs, messwertDruckrel, messwertFeu)
39     cursor=connection.cursor()
40     cursor.execute(sql, value)
41     connection.commit()
42     cursor.close()
43     print("Datensatz erfolgreich gespeichert in der Messtabelle")
44     cursor.close()
45
46 except mysql.connector.Error as error:
47     print("Fehler beim Abspeichern in der Messdatenbanktabelle {}".format(error))
48
49 finally:
50     if(connection.is_connected()):
51         connection.close()
52         print("MySQL-Verbindung geschlossen")

```

Programmcode 7: Test-Messwerte in Datenbank ablegen

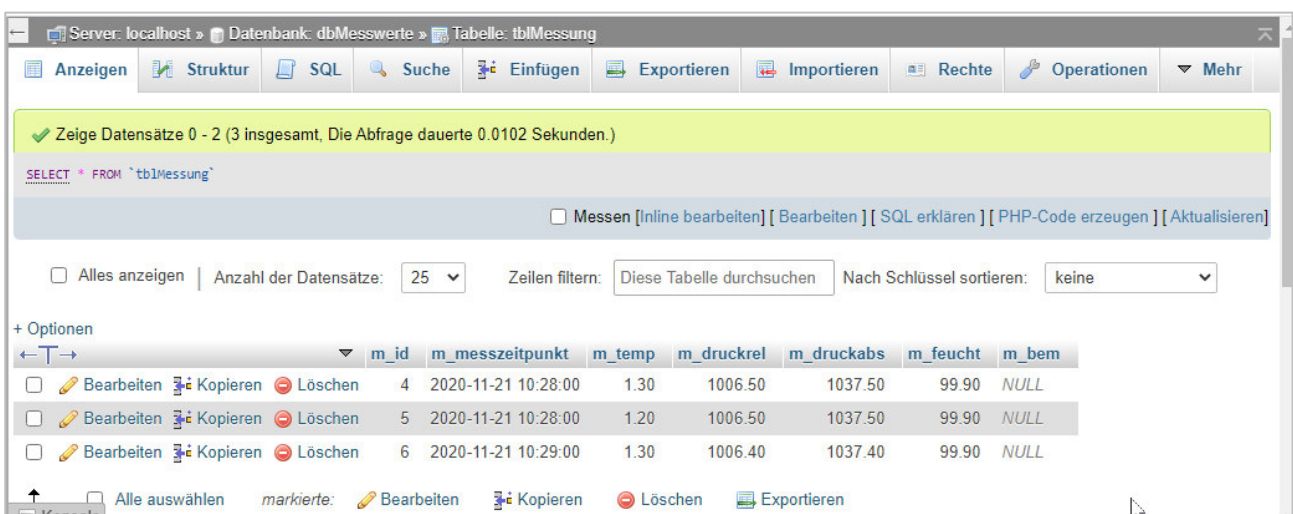
Der erste Zugriff auf MariaDB zeigt eine Fehlermeldung, die durch Übermitteln eines fehlerhaften Passworts entstand. Beim zweiten Versuch mit korrektem Passwort zeigt das Script die erfolgreiche Speicherung der Messwerte an.

```

pi@h-raspi2:/var/www/cgi-bin/wetterstation2 $ sudo python3 sensorabfrage-tag4x.py
Temperatur=1.2°C Feuchtigkeit=99.9%
21.11.2020 - 10:29
Datensatz erfolgreich gespeichert in der Messtabelle
MySQL-Verbindung geschlossen
pi@h-raspi2:/var/www/cgi-bin/wetterstation2 $

```

Eine Überprüfung der Einträge mittels phpMyAdmin zeigt ebenfalls, dass die Messwerte gespeichert sind.



The screenshot shows the phpMyAdmin interface for a database named 'dbMesswerte' and a table named 'tblMessung'. The table contains 3 records. The status bar indicates 'Zeige Datensätze 0 - 2 (3 insgesamt, Die Abfrage dauerte 0.0102 Sekunden.)'.

	m_id	m_messzeitpunkt	m_temp	m_druckrel	m_druckabs	m_feucht	m_bem
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	4	2020-11-21 10:28:00	1.30	1006.50	1037.50	99.90	NULL
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	5	2020-11-21 10:28:00	1.20	1006.50	1037.50	99.90	NULL
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	6	2020-11-21 10:29:00	1.30	1006.40	1037.40	99.90	NULL

11.1 Zeitsteuerung der Messungen

Das automatische Ausführen der Python-Skripts wird jeweils über einen **cronjob** des Linuxsystems erreicht. Die bestehende **/etc/crontab** muss hierzu lediglich durch zwei Zeilen ergänzt werden.

\$ sudo nano /etc/crontab

```

GNU nano 3.2 /etc/crontab Verändert
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .----- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# | | | | |
# * * * * * user-name command to be executed
17 * * * * root cd / && run-parts --report /etc/cron.hourly
25 6 * * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/$
47 6 * * 7 root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/$
52 6 1 * * root test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/$
# Wetterstation
00 0-23/6 * * * root python3 /var/www/cgi-bin/wetterstation2/sensorabfrage-tag4x.py
30 * * * * root python3 /var/www/cgi-bin/wetterstation2/sensorabfrage-stunde.py

^G Hilfe ^O Speichern ^W Wo ist ^K Ausschneide ^J Ausrichten ^C Cursor
^X Beenden ^R Datei öffne ^\ Ersetzen ^U Ausschn. r ^T Rechtschr. ^_ Zu Zeile

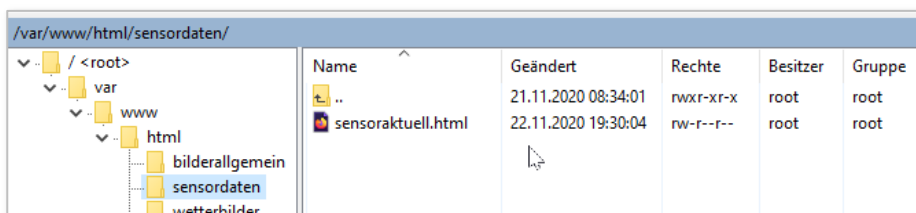
```

Programmcodex 8: crontab-Konfiguration

Um alle Eventualitäten auszuschließen, wird der **cron**-Daemon neu gestartet.

\$ sudo service cron restart

Zur Überprüfung der cron-Funktion wird nach einigen Stunden im Zielverzeichnis des Raspberry Pi überprüft, ob nach einer Stunde nach Minute 30 die gewünschte html-Datei mit den aktuellen Werten vorhanden ist.



Name	Geändert	Rechte	Besitzer	Gruppe
..	21.11.2020 08:34:01	rw-r-xr-x	root	root
sensoraktuell.html	22.11.2020 19:30:04	rw-r--r--	root	root

Abbildung 81: Sensordaten in html-Datei

Außerdem sollten nach einem Tag in der Datenbank mindestens die ersten vier automatisch erfassten Messdaten vorliegen.



	m_id	m_messzeitpunkt	m_temp	m_druckrel	m_druckabs	m_feucht	m_bem
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	8	2020-11-21 18:00:00	1.60	1003.20	1034.10	97.80	NULL
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	9	2020-11-22 00:00:00	-1.40	1001.90	1032.70	99.90	NULL
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	10	2020-11-22 06:00:00	-3.00	1001.40	1032.30	99.90	NULL
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	11	2020-11-22 12:00:00	6.40	1000.00	1030.80	81.10	NULL
<input type="checkbox"/> Bearbeiten <input type="checkbox"/> Kopieren <input type="checkbox"/> Löschen	12	2020-11-22 18:00:00	6.60	998.40	1029.20	86.40	NULL

Abbildung 82: Datenbankeinträge

12 Website vorbereiten

Auf dem Raspberry Pi wird eine Website eingerichtet. Hierzu erfolgt zunächst die Vorplanung des Site-Aufbaus und der Gestaltung. Anschließend wird zunächst der Grundaufbau programmiert und getestet. Die Wetterdaten können implementiert werden, wenn die Struktur der Seiten fixiert ist. Nach Abschluss der Vorarbeiten kann die Website über die Internetverbindung veröffentlicht werden.

12.1 Planung

Zunächst wird ein Seitenlayout vorbereitet. Dies soll in einer zentralen CSS-Datei definiert werden. Um eine möglichst responsible Darstellung zu erreichen, sollen so genannte Mediaqueries eingesetzt werden, d.h. bei Über- bzw. Unterschreiten einer festgelegten Pixelbreite des Anwendungsbrowserfensters werden die Fensterteile unterschiedlich groß und auch an anderen Positionen angezeigt.

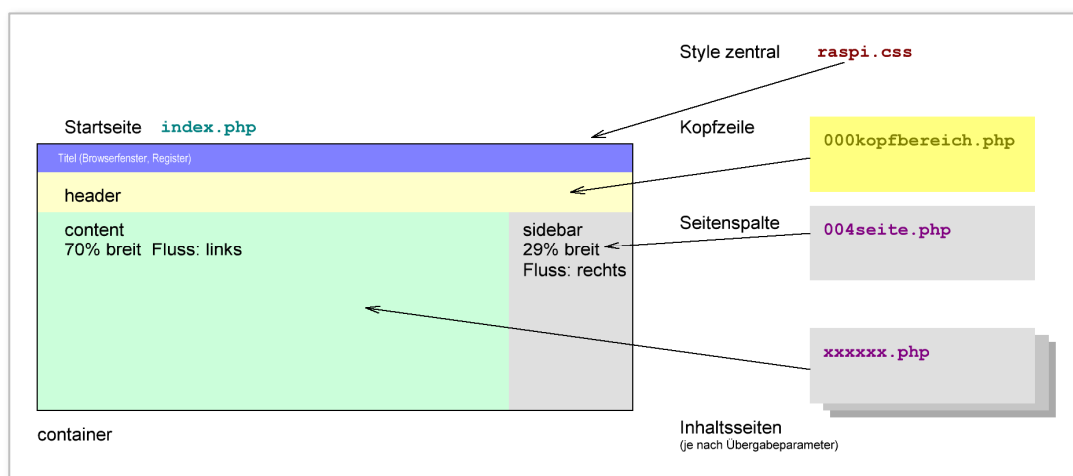


Abbildung 83: Seitenlayout - Planung

Die einzelnen Fensterteile haben folgende Bedeutung:

- **index.php**: Diese Datei wird geladen wenn die Seite zum ersten Mal betreten wird und immer dann, wenn der Benutzer einen Link anklickt, der innerhalb der Seite bleibt.
- **raspi.css**: Die **css**-Datei dient als zentrale Styledatei und enthält alle Layoutinformationen
- **000kopfbereich.php**: Sie soll als feste Siteüberschrift dienen. Außerdem wird hier die Menüleiste eingeblendet.
- **004seite.php**: Hier wird die rechte Spalte, die für Zusatzinformationen eingesetzt werden soll, gespeichert. Bei schmalen Ausgabegeräten < 350px wird dieser Bereich unterhalb des Hauptinhalts angezeigt.
- **xxxxxx.php**: Dies sind alle weiteren Inhaltsseiten, die in den Content-Bereich dynamisch geladen werden sollen.

Damit die einzelnen Dateien im Dateiverzeichnis übersichtlich sortiert werden, erhalten sie als Präfix zwei- oder dreistellige Ziffern. Es ist geplant diese Ziffern aufgrund der Anordnung des Inhalts im Hauptmenü zuzuweisen. Die folgende Abbildung zeigt schematisch das Seitenlayout.

12.2 Programmierumgebung

Zur Organisation der Programmierumgebung wird auf einem Windows-PC dieselbe Ordnerstruktur angelegt, wie sie auf dem Raspberry PI Zero vorliegt. Dadurch kann lokal gearbeitet werden und bei Fertigstellung ein Upload auf den Webserver erfolgen.

- D:\WEB-Raspi2\var\www\html
Anfragen von außen gehen vom Apache2 in dieses Verzeichnis. Dort wird nach einer Datei mit dem Namen **index.html** oder **index.php** gesucht.
- D:\WEB-Raspi2\var\www\html\wetterbilder
Dies ist das Verzeichnis das aktuelle Wetterbild.
- D:\WEB-Raspi2\var\www\html\sensordaten
Dies ist das Verzeichnis für die aktuellen Sensordaten.
- D:\WEB-Raspi2\var\www\files
Dieses Verzeichnis dient als Ablage für Daten, die nicht vom Apache2 und damit von außen geöffnet werden können. Hier werden z.B. Zugangsdaten zur Datenbank gespeichert. Nur ein php-Skript darf darauf zugreifen.
- D:\WEB-Raspi2\var\www\cgi-bin
Hier werden Skripte abgelegt, die ebenfalls nur von einem php-Skript benutzt werden können.

Mit dem Windowstool WinSCP können in Zweifensterdarstellung optimal der Windows-Entwicklerrechner und der Raspberry PI dargestellt werden. Der Datenaustausch wird damit erleichtert, zur Unterstützung wird eine Verzeichnissynchronisation eingerichtet, so dass lokales und das entfernte (Server-)Verzeichnis stets in derselben Ebene angezeigt werden.

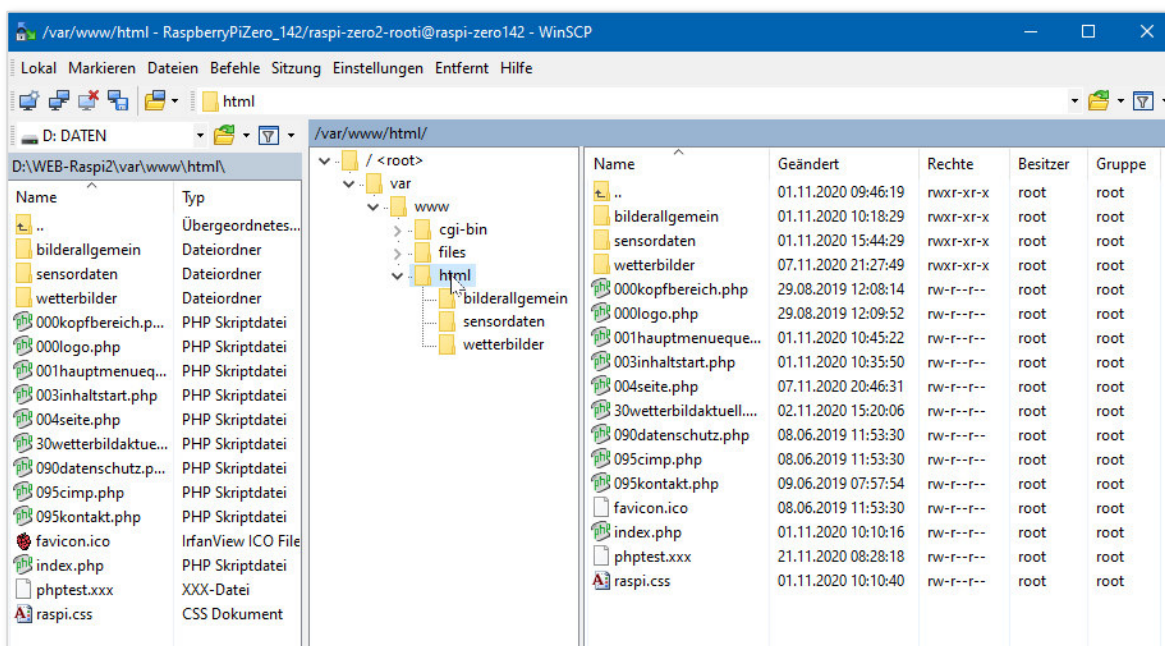


Abbildung 84: Programmierumgebung zur Webseitenentwicklung

12.3 Programmierung der Grundstruktur

Zunächst wird die Hauptseite und die zugehörige Styledatei für Layout und Gestaltung angelegt. Anschließend wird die Startseite durch Hinzufügen von Menüsteuerung und Inhaltsseite für erste Tests funktionsfähig gemacht.

Im Web-Stammverzeichnis `/var/www/html` befinden sich zusätzlich zur Startdatei `index.php` und Styledatei `raspi.css` noch das Favorite-Icon mit dem Dateinamen `favicon.ico`, das automatisch geladen und im Browser-Register angezeigt wird. In den Kopfbereich `000kopfbereich.php` wird die Datei `001hauptmenuequer.php` und die Logodatei `000logo.php` eingefügt. Das Hauptmenü sorgt für die dynamische Steuerung der Website.

Das Hauptmenü `001hauptmenuequer.php` wird so geplant, dass fünf Hauptmenüpunkte (Start, Hauptmenü, Projekt, Wetterdaten, Infos) auf der Website angezeigt werden. Bei kleinen Bildschirmauflösungen (Handys) soll nur noch ein so genanntes „Hamburger-Menü“ dargestellt werden. Durch Anklicken werden die Menüpunkte sichtbar.

Startseite

Die Datei `index.php` dient als Startseite und wird bei jedem internen Seitenwechsel aufgerufen. Lediglich der Inhaltsbereich wechselt aufgrund der Menüwahl.

Im Kopfbereich wird zunächst durch die Cookiesteuerung ein Besucherzähler vorbereitet (2), zusätzlich wird der Zeichensatz der Seite auf UTF-8 festgelegt (3). Dies ist in manchen Fällen notwendig, damit sicher der korrekte Zeichensatz dargestellt wird. Der eigentliche html5-Code beginnt in Zeile (6). In `<head>` wird nochmals UTF-8 definiert (9), dann folgt der Zugriff auf die Styledatei `raspi.css` (10) und in Zeile (11) wird der Start-Viewport eingestellt.

Die Zuweisung des Favicons (12) ist durch Verwendung des Standardnamen `favicon.ico` nicht unbedingt notwendig, dient hier lediglich zu Demonstration. Zeile (13) beschreibt den Titel des Browserfensters.

```

1 <?php
2 setcookie("raspi2","besucht",time()+(60*60*24)); //60Sekunden*60Minuten*24Stunden=1Tag
3 header('Content-Type: text/html;charset=utf-8');
4 //Das Cookie raspi2 dient zur Wiedererkennung des Benutzers
5 ?>
6 <!DOCTYPE HTML>
7 <html> <!--Kopfdaten der Site-->
8 <head>
9     <meta charset="UTF-8"> <!--Zeichensatz-->
10    <link href="./raspi.css" rel="stylesheet" type="text/css"> <!--css-Verweis-->
11    <meta name="viewport" content="width=device-width, initial-scale=0.8, user-scalable=yes">
12    <link href="favicon.ico" rel="shortcut icon"> <!--Favoriten-Icon-->
13    <title>Raspberry PI-Website</title> <!--Browser/Register/Favoriten-Titel-->
14 </head>

```

Programmcod 9: `index.php` - Teil 1

Im `<body>`-Bereich wird die Steuerung zum dynamischen Nachladen der Inhaltsteile vorbereitet. Ein Klick auf einen Menüpunkt lädt stets die Datei `index.php`, übergibt im Gegensatz zum ersten Aufruf zusätzlich in der URL die gewünschte Inhaltsdatei mit dem `$_GET`-Parameter. Hier wird dieser übergebene Parameter in die Variable `$strZiel` gespeichert (20).

Im Beispiel wird der Parameterwert **aw=20projekt.php** an die Datei **index.php** übergeben.

http://192.168.123.142/index.php?aw=20projekt.php

Beim ersten Aufruf der Startseite muss als Ziel die Seite **003inhaltstart.php** geladen werden (24).

```

15 <body> <!--darzustellende Elemente auf der Seite-->
16 <?php
17 // ---- Übergabeparameter auslesen: Zieldatei für content
18 if(isset($_GET['aw'])) //Inhaltsdatei aus der GET-Übergabe lesen
19 {
20     $strZiel=$_GET['aw'];
21 }
22 else //Startdatei einfügen, beim ersten Aufruf
23 {
24     $strZiel='003inhaltstart.php';
25 }
26 //----- Seite darstellen Kopf/Inhalt/Seitenleiste -----
27 ?>

```

Programmcode 10: index.php - Teil 2

Im nächsten Abschnitt werden die Layout-Bereiche **<div>** mit den entsprechenden Inhalten gefüllt. Dabei umschließt **container** den kompletten Inhalt (29), darin sind die weiteren Bereiche untergebracht.

- Kopfzeile **header** (32) mit der Datei 000kopfbereich.php.
- Inhaltsbereich **content** (36) mit der, in der Variablen **\$strZiel** enthaltenen, dynamisch einzufügenden Datei. Beim ersten Aufruf ist die die Datei **003inhaltstart.php** (24).
- Der Bereich **sidebar** soll nur angezeigt werden, wenn derselbe Inhalt nicht schon als Ziel ausgewählt wurde. Deshalb erfolgt die Auswahl über eine **if**-Struktur (41).

```

28 <div id="container"> <!-- Container/ Bereich für den gesamten Inhalt -->
29 <div id="header"> <!-- Kopfzeilenbereich für die Überschrift-->
30 <?php
31     include "../000kopfbereich.php";
32 <?>
33 </div>
34 <div id="content"> <!-- Inhalt wird dynamisch je nach Menüauswahl eingefügt-->
35 <?php include "." . $strZiel ;?>
36 </div>
37 <div id="sidebar"> <!--wird unten und ab 370px rechts gezeigt-->
38
39 <!--um Dopplung zu vermeiden, wird die rechte Seite tlw. ausgeblendet-->
40 <?php
41     if($strZiel=='30wetterbild.php'){
42         include "";
43     }
44     else{
45         include "../004seite.php";
46     }
47 <?>
48 </div>
49 </div>

```

Programmcode 11: index.php - Teil 3

Damit das Seitenlayout korrekt dargestellt wird, ist eine **CSS**-Datei notwendig, die im folgenden Abschnitt beschrieben wird. Der Kopfbereich mit Logo und Menüdarstellung wird im Abschnitt erläutert.

Stylesheet-Datei

In der Stylesheet-Datei wird der Ansatz „mobile-first“ verfolgt, d.h. zunächst wird die Anzeige für Geräte mit kleiner Bildschirmauflösung gut lesbar eingestellt. Alle anderen Auflösungen werden später anhand dieses Ansatzes verändert.

Als Schriftart kommt die Familie Roboto zum Einsatz. Als Maß für die Abstände und Schriftgrößen wird **rem** verwendet, das für responsive Designs und relative Schriftgrößen wesentliche Vorteile gegenüber Pixel oder Pt-Angaben besitzt (2-5).

Im html-Bereich, also auf der gesamten Seite gilt als Schriftgröße 100%. Die minimale Breite wird auf 370px beschränkt (12-13). Dies verhindert ein „Stauchen“ der Seite, wenn das Browserfenster schmal dargestellt wird.

Die Layoutdarstellung des Bildschirms wird durch die ID's (#) für Logo, Container, Content und Sidebar festgelegt (24-45). Durch so genannte Media-Queries bestimmt später der Browser abhängig von der zur Verfügung stehenden Bildschirmauflösung, in welcher Größe und Gestaltung die einzelnen Bereiche angezeigt werden.

```

1  /* CSS Document  raspi.css*/
2  *{
3      font-family : Roboto, Verdana , Arial, Helvet
4      padding:0rem; margin: 0rem;
5      box-sizing: border-box;
6  }
7  /*-----
8  mobile-first-Ansatz
9  für die kleinsten Geräte (iPhone...)
10 -----*/
11 html{
12     font-size:100%;
13     min-width:370px;
14 }
15 body {
16     background: #ffffff;
17     line-height: 1.1rem;
18     color: #000000;
19     padding: 0;
20     text-align:left;
21     word-wrap:break-word !important;
22 }
23
24 #meinlogo {
25     display: block;
26     padding: 0.1rem;
27     float:right;
28 }
29
30
31 #container {
32     margin: 0 ;
33     max-width: 1600px;
34     padding:1rem 1rem 5rem 1rem;
35 }
36
37 #content{
38     margin: 0 ;
39     padding:1rem 1rem 5rem 1rem;
40     float:left;
41     width: 100%;
42 }
43
44 #sidebar{
45     display:none;
46 }
47

```

Programmcod 12: css-Styledatei (Ausschnitt)

Beispiele für die Definition der Media-Queries befinden sich auf der folgenden Seite.

Mit dem Internet-Browser kann die Umschaltung der Media-Queries leicht getestet werden. Beim Firefox-Browser geschieht dies durch **Strg** + **Shift** + **m** (siehe Abbildung 85), hier 736px Breite.

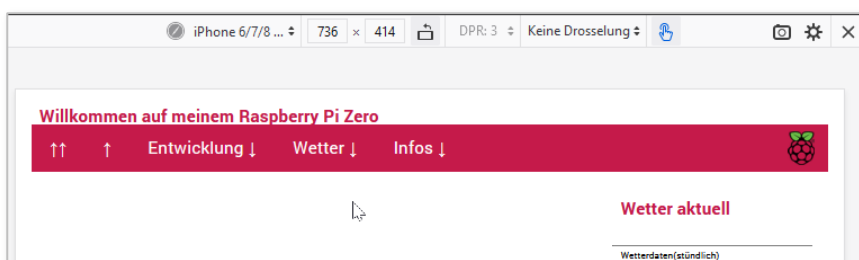


Abbildung 85: Media-Query mit Firefox

Spätere Definitionen in der Styledatei überschreiben die früheren Definitionen. Deshalb müssen unten nur noch Ergänzungen zum „Mobile-First-Ansatz“ gemacht werden.

Im Beispiel wurden folgende Media-Queries eingesetzt:

- Maximal-Bildschirmauflösung Breite 4800px (230-252)
Große Schrift 150%, Position der div-Container **content** und **sidebar** nebeneinander mit Textfluss, Breite der Bereich 70:29
- Maximal-Bildschirmauflösung Breite 1400px (255-263)
Änderung in Schriftgrößen auf 130%.
- Maximal-Bildschirmauflösung Breite 1000px (265-273)
Änderung in Schriftgrößen auf 110%.

```

227  /*-----
228  Media Queries
229  -----*/
230  @media all and (max-width : 4800px) {
231      html{
232          font-size:150%;
233      }
234      .titel
235      {
236          font-size: 1.5rem;
237      }
238      #content{
239          margin: 0 ;
240          padding:1rem 1rem 5rem 1rem;
241          float:left;
242          width: 70%;
243      }
244
245      #sidebar{
246          margin: 0 ;
247          float:right;
248          width:29%;
249          padding:1rem 1rem 5rem 1rem;
250          display:block;
251      }
252  }

255  @media all and (max-width : 1400px) {
256      html{
257          font-size:130%;
258      }
259      .titel
260      {
261          font-size: 1.3rem;
262      }
263  }
264
265  @media all and (max-width : 1000px) {
266      html{
267          font-size:110%;
268      }
269      .titel
270      {
271          font-size: 1.1rem;
272      }
273  }

```

Programmcode 13: css-Styledatei (Bereiche und Media-Queries)

Erst beim Unterschreiten der Bildschirmbreite von 768px wird das Layout umgestellt und auf das so genannte „Hamburger-Menü“. Die Menüsteuerung **nav** wird komplett unter ein einziges Symbol verlegt (siehe Abbildung 86).

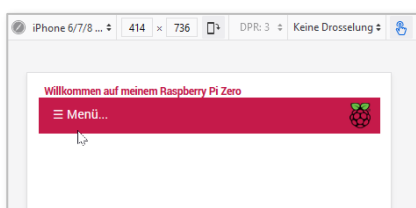


Abbildung 86: Webseite bei kleiner Anzeigebreite

```

331  @media all and (max-width : 768px) {
332      html{
333          font-size:100%;
334      }
335      .titel
336      {
337          font-size: 0.9rem;
338      }
339      nav {
340          margin: 0;
341      }
342      /* Navigation verstecken */
343      .toggle + a, .menu {
344          display: none;
345      }
346      /* Style bei der Menütabelle
347      für kleine Auflösung */
348      .toggle {
349          display: block;

```

Kopfbereich

Der Kopfbereich der Seite wird in der Datei **000kopfbereich.php** abgelegt. Hierin befindet sich der Aufruf des Logos und das Einbinden des Hauptmenüs aus der Datei

001hauptmenuequer.php. Dies ist der komplette Quellcode des Kopfbereichs:

```

1 <?php
2     echo "<p class='titel'>Willkommen auf meinem Raspberry Pi Zero</p>";
3 <?>
4 <div id="meinlogo">
5     <?php
6         include "../bilderallgemein/00logo.php";
7     <?>
8 </div>
9 <?php
10    include "../001hauptmenuequer.php";
11 <?>

```

Programmcod 14: Kopfbereich

Menübereich

Das Hauptmenü in der Datei **001hauptmenuequer.php** ist etwas komplexer aufgebaut, da hier auch die Media-Queries berücksichtigt werden müssen. Der Container **nav** schließt das gesamte Menü ein.

Bei kleinen Auflösungen soll fürs Einblenden der Menüs per dropdown-Schalter eine versteckte Check-box verwendet werden (2-4).

```

1 <nav>
2     <!-- Menü für kleine Auflösungen -->
3     <label for="drop" class="toggle">Menü...</label>
4     <input type="checkbox" id="drop" />
5     <!-- Menü für alle großen Auflösungen -->

```

Programmcod 15: Menü-dropdown

Die Auswirkung bei eingeklapptem Menü ist in Abbildung 86 zu erkennen.

Die Darstellung des Gesamtmenüs wird in den folgenden Beispielen sichtbar. Die Formatierung von **ul**-Tags (unordered lists) wird in der CSS-Datei so formatiert, dass ein so genanntes CSS-Menü entsteht.

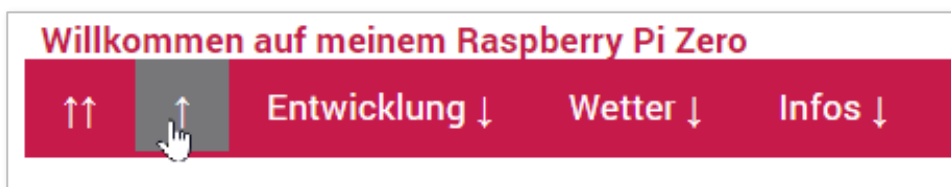


Abbildung 87: Menüleiste



Der erste und zweite Menüpunkt dient als Sprung zur Haupt-Homepage bzw. als Sprung zurück zur Startseite **index.php**. Dahinter steht jeweils kein Dropdown-Menü. Durch das html-Sonderzeichen **↑** wird ein senkrechter Pfeil erzeugt.

```

6      <ul class="menu">
7
8          <!-- Hauptmenü1 -->
9          <li><a href="https://www.hans-koch.eu/index.php">&uarr;&uarr;</a></li>
10         <li><a href="./index.php">&uarr;</a></li>

```

Programmcode 16: Menübereich 1



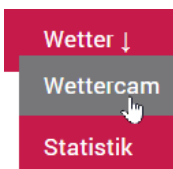
Im Menüpunkt „Entwicklung“ erfolgt ein Dropdown, die Unterpunkte klappen auf. Bei geringer Bildschirmauflösung wird durch die versteckte Checkbox das Menü offengehalten. Die Verweise zu den Untermenüpunkten sind hier noch nicht eingetragen.

```

<!-- Hauptmenü2 -->
<li>
    <!-- Toggelschalter für das Hauptmenü2 -->
    <label for="drop-2" class="toggle">Entwicklung &darr;</label><a href="#">Entwicklung
    <input type="checkbox" id="drop-2"/>
    <ul><!-- Untermenüs des Hauptmenü2 -->
        <li><a href="./index.php?aw=20projekt.php">Planung</a></li>
        <li><a href="./index.php?aw=21projekthardware.php">Hardware</a></li>
        <li><a href="./index.php?aw=22raspieinrichten.php">Raspi einrichten</a></li>
        <li><a href="./index.php?aw=23aufbau.php">Aufbau</a></li>
        <li><a href="./index.php?aw=24programm.php">Programmierung</a></li>
    </ul>
</li>

```

Programmcode 17: Menübereich 2



In gleicher Art werden alle anderen Menüpunkte programmiert. Als Linkadresse wird jeweils die Datei **index.php** mit Übergabe der Inhaltsdatei verwendet, z.B. **index.php?aw=30wetterbildaktuell.php**.

```

<!-- Hauptmenü3 -->
<li>
    <!-- Toggelschalter für das Hauptmenü3 -->
    <label for="drop-3" class="toggle">Wetter&darr;</label><a href="#">Wetter &darr;</a>
    <input type="checkbox" id="drop-3"/>
    <ul><!-- Untermenüs des Hauptmenü3 -->
        <li><a href="./index.php?aw=30wetterbildaktuell.php">Wettercam</a></li>
        <li><a href="./index.php?aw=31wetterstatistik.php">Statistik</a></li>
    </ul>
</li>

```

Programmcode 18: Menübereich 3

Logodatei

Die Datei **00logo.php** enthält als einzige Zeile den Zugriff auf das Bild mit Pfad zum Unterordner.

```
1 <p class="textklein"></p>
2
```

Programmcod 19: Logodatei

Inhaltsdateien

Jede Inhaltsdatei wird nach Anwahl des entsprechenden Menüpunkts und damit der Übergabe des gewünschten Dateinamens von der **index.php** in den **content**-Bereich geladen. Beim ersten Besuch wird automatisch die Datei **003inhaltstart.php** eingefügt. Sie enthält die Anzeige der aktuellen Wettercam und der Sensordaten.

```
003inhaltstart.php
1 <?php
2 echo "<div id='infospalterrechts'>";
3 echo "<p class='textnormal'>";
4 $strDateiname= "./sensordaten/sensoraktuell.html";
5 include ($strDateiname);
6 echo "</p>";
7 echo "</div>";
8 echo "<div id='infospaltelinks'>";
9 echo "<p>Wettercam<br><img src='./wetterbilder/wetterbildaktuell.jpg'></p>";
10 echo "</p>";
11 echo "</div>";
12
13
14 ?>
```

Programmcod 20: Inhaltsdatei beim Erstaufwurf

Im Browser sieht die Startseite so aus:

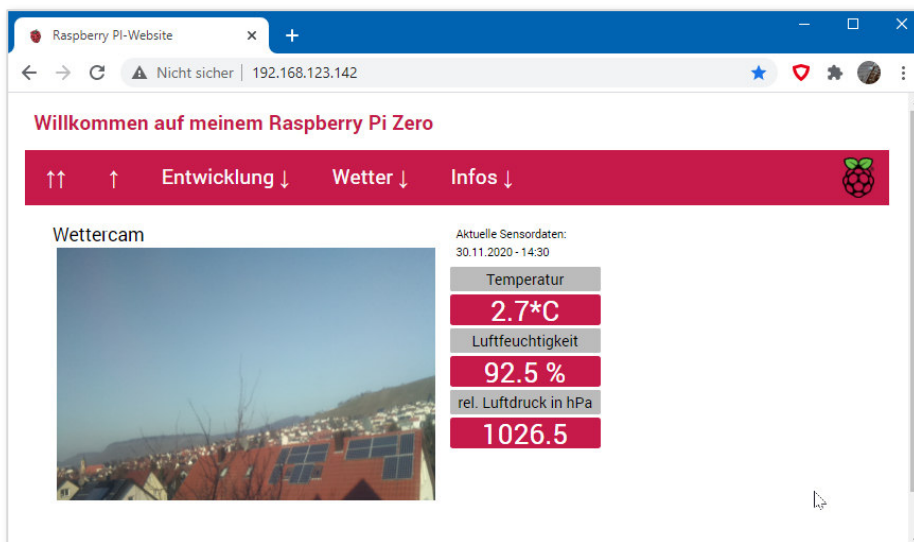


Abbildung 88: Startseite im Browser

Das Favicon wird in der Registerkarte des Browsers angezeigt, auf die Titelzeile folgt die Menüleiste quer mit fünf Menüpunkten, dem Logo rechts, danach der Inhaltsbereich in zweispaltiger Anordnung. Die Sidebar rechts ist noch leer.

13 Datenbankinfos aufbereiten und ausgeben

Im folgenden Abschnitt werden unterschiedliche Methoden zum Aufbereiten und Anzeigen der Messdaten erläutert. Dies reicht von der Darstellung als einfache Tabelle bis zu verschiedenen Grafik-Statistiken.

13.1 Sensordaten als Tabelle ausgeben

Die gemessenen Daten sollen im Hauptbereich der Seite als Tabelle dargestellt sein. Um eine sinnvolle Anzeige und gute Lesbarkeit zu erreichen, werden tägliche Messzeiten zusammengefasst und die Anzeige optisch optimiert. Abbildung 89 zeigt die gewünschte Form. Die Seite wird in der Menüleiste unter dem Punkt **Wetter-Wochenrückblick** eingebunden. Der Programmcode der Seite ist unter beschrieben.

Willkommen auf meinem Raspberry Pi Zero

↑↑ ↑ Entwicklung ↓ Wetter ↓ Infos ↓

Rückblick 7 Tage...

Messzeitpunkt 00:00 Uhr **Wochenrückblick**

Messzeitpunkt	Temperatur	Luftdruck rel.	Luftdruck abs.	Feuchtigkeit
2020-12-12	3.50°C	967.10 hpa	996.80 hpa	99.90 %
2020-12-11	2.10°C	977.90 hpa	1008.00 hpa	99.90 %
2020-12-10	1.90°C	981.60 hpa	1011.80 hpa	99.90 %
2020-12-09	1.40°C	977.00 hpa	1007.10 hpa	99.90 %
2020-12-08	2.00°C	973.80 hpa	1003.70 hpa	99.90 %
2020-12-07	2.90°C	969.70 hpa	999.60 hpa	99.90 %
2020-12-06	3.50°C	971.50 hpa	1001.40 hpa	99.90 %

Messzeitpunkt 06:00 Uhr

Messzeitpunkt	Temperatur	Luftdruck rel.	Luftdruck abs.	Feuchtigkeit
2020-12-12	3.70°C	967.20 hpa	996.90 hpa	99.90 %
2020-12-11	-0.10°C	973.80 hpa	1003.70 hpa	99.90 %
2020-12-10	1.70°C	979.00 hpa	1009.20 hpa	99.90 %
2020-12-09	1.50°C	979.10 hpa	1009.30 hpa	99.90 %
2020-12-08	-0.10°C	972.90 hpa	1002.80 hpa	99.90 %
2020-12-07	3.10°C	971.20 hpa	1001.10 hpa	99.90 %
2020-12-06	3.00°C	969.70 hpa	999.50 hpa	99.90 %

Messzeitpunkt 12:00 Uhr

Messzeitpunkt	Temperatur	Luftdruck rel.	Luftdruck abs.	Feuchtigkeit
2020-12-11	3.00°C	971.00 hpa	1000.90 hpa	99.90 %
2020-12-10	3.40°C	978.10 hpa	1008.20 hpa	99.90 %
2020-12-09	2.90°C	980.80 hpa	1011.00 hpa	99.90 %
2020-12-08	3.80°C	973.20 hpa	1003.20 hpa	98.70 %
2020-12-07	6.40°C	973.00 hpa	1002.90 hpa	99.20 %
2020-12-06	4.30°C	970.50 hpa	1000.40 hpa	99.90 %
2020-12-05	3.60°C	972.50 hpa	1002.40 hpa	99.90 %

Messzeitpunkt 18:00 Uhr

Messzeitpunkt	Temperatur	Luftdruck rel.	Luftdruck abs.	Feuchtigkeit
2020-12-11	3.40°C	968.10 hpa	997.90 hpa	99.90 %
2020-12-10	2.70°C	978.40 hpa	1008.50 hpa	99.90 %
2020-12-09	2.60°C	981.20 hpa	1011.40 hpa	99.90 %
2020-12-08	0.90°C	974.60 hpa	1004.60 hpa	99.90 %
2020-12-07	13.90°C	1004.60 hpa	1035.50 hpa	99.90 %

Abbildung 89: Messdaten in Tabellenansicht

Zum Zugriff auf die Datenbank wird zunächst der Connectionstring eingebunden. Dieser liegt auf dem Server in einem Verzeichnis außerhalb des html-Bereichs. Anschließend erfolgt die Zusammenstellung des SQL-Strings und der Abruf der Daten. Diese können dann in einer Tabelle angezeigt werden.

Connectionstring

In Zeile (3) wird der Connectionstring eingebunden, danach erfolgt die Darstellung der Hauptüberschrift auf der Seite (4).

```

1 <?php
2 //Zugangsdaten zur Datenbank einbinden
3 include ('../files/zugang/raspizero-dbverbindung.php.inc');
4 echo "<h1 class='u1'>Rückblick 7 Tage...</h1>";

```

Programmcode 21: Datenbankzugriff Messdaten Zugang

SQL-String

Damit vier Tabellen erstellt werden können, müssen die Tageszeiten gesondert in SQL-Abfragen gepackt werden. Deshalb werden in einer Schleife vier unterschiedliche SQL-Strings generiert und nacheinander an die Datenbank geschickt, das Ergebnis angezeigt usw.

Damit die Zeilen unterschiedlich gestylt sind, wird über den Modulo-Operator (%) festgestellt, ob die Zeilenzahl gerade oder ungerade ist und dabei unterschiedliche Style-Klassen der **CSS**-Datei angewendet (siehe Programmcode 23).

```

5 //Schleife für die vier Tageszeiten vorbereiten
6 $i=0;
7 for($i=0;$i<4;$i++)
8 {
9     if($i==0){ $strMesszeit="00:00"; }
10    elseif($i==1){ $strMesszeit="06:00"; }
11    elseif($i==2){ $strMesszeit="12:00"; }
12    elseif($i==3){ $strMesszeit="18:00"; }
13    echo "<h2>Messzeitpunkt " . $strMesszeit . " Uhr</h2>";
14    //Zugriff auf die Datenbank
15    //SQL zusammenstellen
16    $strSQL="SELECT * FROM `tblMessung` ";
17    $strSQL= $strSQL . "WHERE (m_messzeitpunkt LIKE '% " . $strMesszeit . "%')
18    AND (m_messzeitpunkt > DATE_SUB(NOW(), INTERVAL 7 DAY)) ";
19    $strSQL= $strSQL . "ORDER BY `m_messzeitpunkt` DESC;";
20    //SQL wegschicken und Recordset zurück
21    $recordset=mysqli_query($dbverb,$strSQL);
22    if (!$recordset)
23    { //bei Fehler
24        echo "<p class='textrot'>Fehlerhafte Abfrage der Daten! <br />".
25        mysqli_errno($dbanklink) . "<br />" . mysqli_error($dbanklink) .
26        "</p>";
27    }
28    else

```

Programmcode 22: Datenbankzugriff Messdaten SQL

Anzeige

Die Anzeige erfolgt in Tabellenform. CSS-Klassen stylen hierbei die einzelnen Zeilen und Zellen. Um eine bessere Lesbarkeit zu erhalten, werden die Zeilen abwechselnd unterschiedlich gefärbt.

Die Ausgabe erfolgt so lange, bis keine Daten mehr im Recordset-Array vorhanden sind. SO entstehen vier untereinander angeordnete Tabellen mit den Messdaten der letzten sieben Tage inklusive der Überschrift unter Angabe der Tageszeit der Messung.

```

25 |         else
26 |         {
27 |             // #Ausgabe in html-Tabelle
28 |             echo "<table class='statistik' >";
29 |             echo "<tr height='1rem' class='statzeile2'>";
30 |             echo "<td class='statzelle' >Messzeitpunkt</td>";
31 |             echo "<td class='statzelle' >Temperatur</td>";
32 |             echo "<td class='statzelle' >Luftdruck rel.</td>";
33 |             echo "<td class='statzelle' >Luftdruck abs.</td>";
34 |             echo "<td class='statzelle' >Feuchtigkeit</td>";
35 |             echo "</tr>";
36 |             $intZeilenzahl=0;
37 |             while($ds=mysqli_fetch_array($recordset))
38 |             {
39 |                 $intZeilenzahl++;
40 |                 if($intZeilenzahl%2==0)
41 |                 {
42 |                     echo "<tr class='statzeilegerade'>";
43 |                 }
44 |                 else
45 |                 {
46 |                     echo "<tr class='statzeileungerade'>";
47 |                 }
48 |                 echo "<td class='statzelle'>" . substr($ds['m_messzeitpunkt'],0,10) .
49 |                     "</td>";
50 |                 echo "<td class='statzelle'>" . $ds['m_temp'] . " °C</td>";
51 |                 echo "<td class='statzelle'>" . $ds['m_druckrel'] . " hpa</td>";
52 |                 echo "<td class='statzelle'>" . $ds['m_druckabs'] . " hpa</td>";
53 |                 echo "<td class='statzelle'>" . $ds['m_feucht'] . " %</td>";
54 |             }
55 |             echo "</table>";
56 |         }

```

Programmcode 23: Datenbankzugriff Messdaten Anzeige

Datenbankzugriff schließen

Nachdem alle Datensätze ausgegeben sind und die letzte Tabelle geschlossen ist, wird auch die Datenbankverbindung beendet.

```

54 |         echo "</table>";
55 |     }
56 |
57 | }
58 | //DB-Zugriff schliessen
59 | mysqli_close($dbverb);
60 | ?>

```

Programmcode 24: Datenbankverbindung schließen

13.2 Grafische Wetterstatistik generieren

Die im letzten Abschnitt verwendete Tabellendarstellung ist zur Aufbereitung von Messdaten eines längeren Zeitraums wenig geeignet. Die Übersicht über den Verlauf ist nur schwer zu erreichen. Es bietet sich an, alle Daten grafisch auf der Webseite darzustellen. Hierfür werden unterschiedliche Bibliotheken angeboten.

- Matplotlib
- JQuery
- JpGraph

Bis zur Einbindung auf der Raspi-Wetterstationsseite müssen einige Installations- und Testschritte durchlaufen werden.

JpGraph scheint sich am Besten in eine PHP7-/MySQL-/Apacheumgebung einzupassen. Aus diesem Grund soll dieses Paket nun in die bestehende Installation eingebunden werden.

Download und Installation

Wie immer vor der Installation neuer Pakete sollte das System auf den neuesten Stand gebracht werden.

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

Anschließend erfolgt die Installation des **jpggraph** Pakets.

```
$ sudo apt-get install libphp-jpgraph
```

Das Zielverzeichnis kann allerdings nicht vom Webserver erreicht werden. Deshalb muss es in das **html**-Unterverzeichnis verschoben werden. Zunächst wird der Installationspfad gesucht.

```
$ sudo /find -name jpgraph*
```

```
/usr/share/jpgraph/jpgraph_error.php
/usr/share/jpgraph/jpgraph_scatter.php
/usr/share/jpgraph/jpgraph_canvas.php
/usr/share/jpgraph/jpgraph_gantt.php
root@h-raspi2:/#
```

Danach kann **jpggraph** komplett in den **html**-Ordner des Webserver verschoben werden.

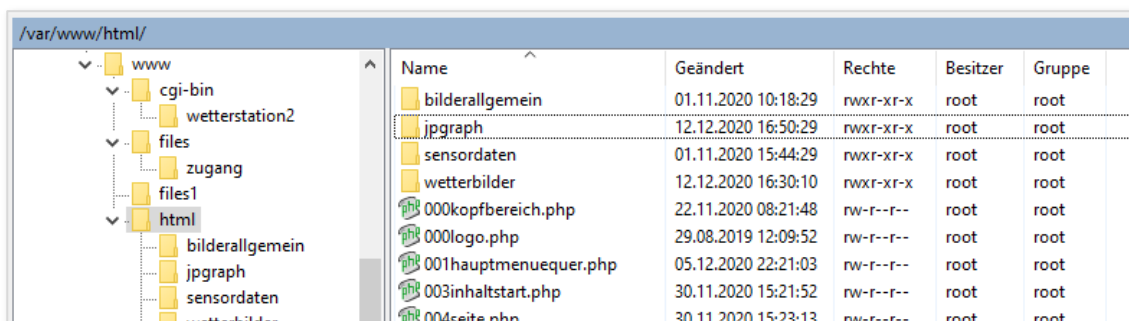


Abbildung 90: **jpggraph** im Webserverordner

Testskript erstellen

Ein kleines Testskript soll die Funktion von **jpggraph** überprüfen und dann als Grundlage für die Einbindung in die Webseite dienen.

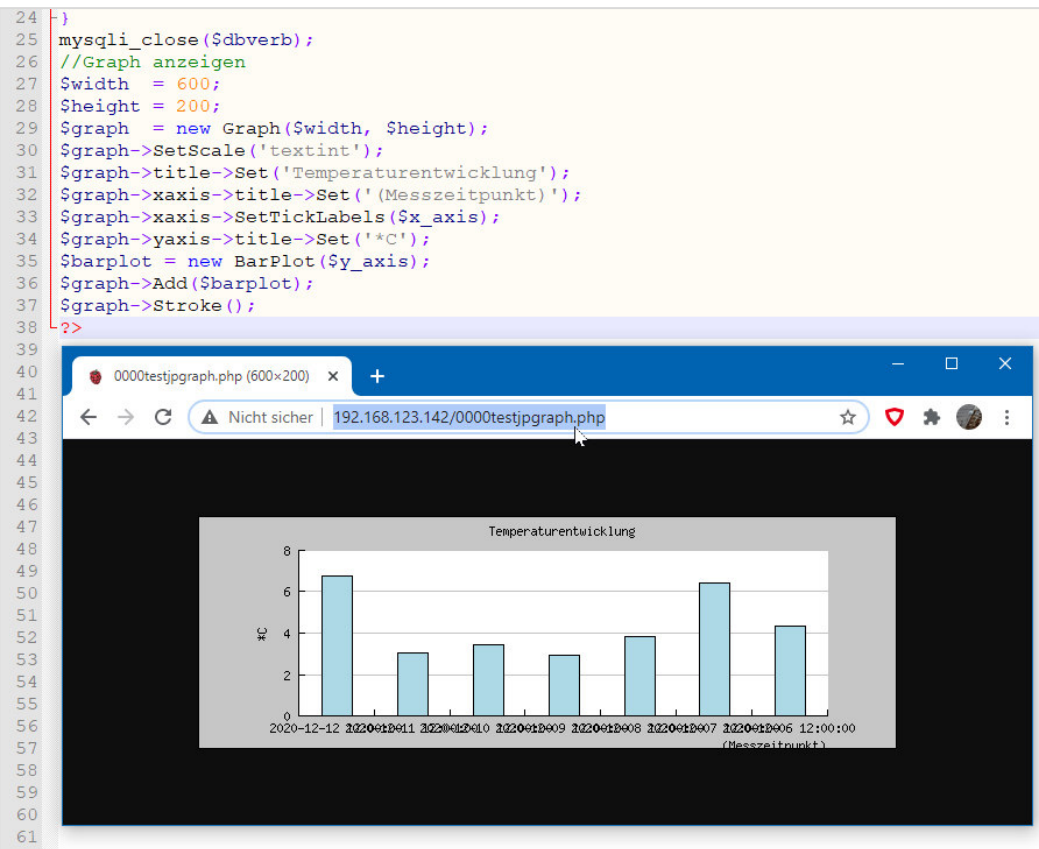
jpggraph einbinden und auf Datenbank zugreifen

```

1 <?php
2 //JPGraph einbinden
3 require_once('jpggraph/jpggraph.php');
4 require_once('jpggraph/jpggraph_line.php');
5 require_once('jpggraph/jpggraph_bar.php');
6 //Variablen vorbereiten
7 $x_axis = array();
8 $y_axis = array();
9 $i = 0;
10 $strMesszeit="12:00";
11
12 //Datenbankzugriff
13 include ('../files/zugang/raspizero-dbverbindung.php.inc');
14 $strSQL="SELECT * FROM `tblMessung` ";
15 $strSQL= $strSQL . "WHERE (m_messzeitpunkt LIKE '% ' . $strMesszeit . '%') AND
(m_messzeitpunkt > DATE_SUB(NOW(), INTERVAL 7 DAY)) ";
16 $strSQL= $strSQL . "ORDER BY `m_messzeitpunkt` DESC;";
17 $recordset=mysqli_query($dbverb,$strSQL);
18 //Daten für Graph zusammenstellen
19 while ($row = mysqli_fetch_array($recordset))
20 {
21     $x_axis[$i] = $row["m_messzeitpunkt"];
22     $y_axis[$i] = $row["m_temp"];
23     $i++;
24 }
25 mysqli_close($dbverb);

```

Aufruf der Datei im Browser



Skript in Website integrieren

Nun sollen Skripte erstellt werden, die die Wetterdaten aus der Datenbankauslesen und übersichtlich auf der Webseite darstellen. Der Programmcode wird in mehrere Teile aufgeteilt, um keine überflüssigen Dopplungen zu erhalten.

Im ersten Step ist eine Einbindung in die bestehende Umgebung geplant (siehe Abbildung 91).

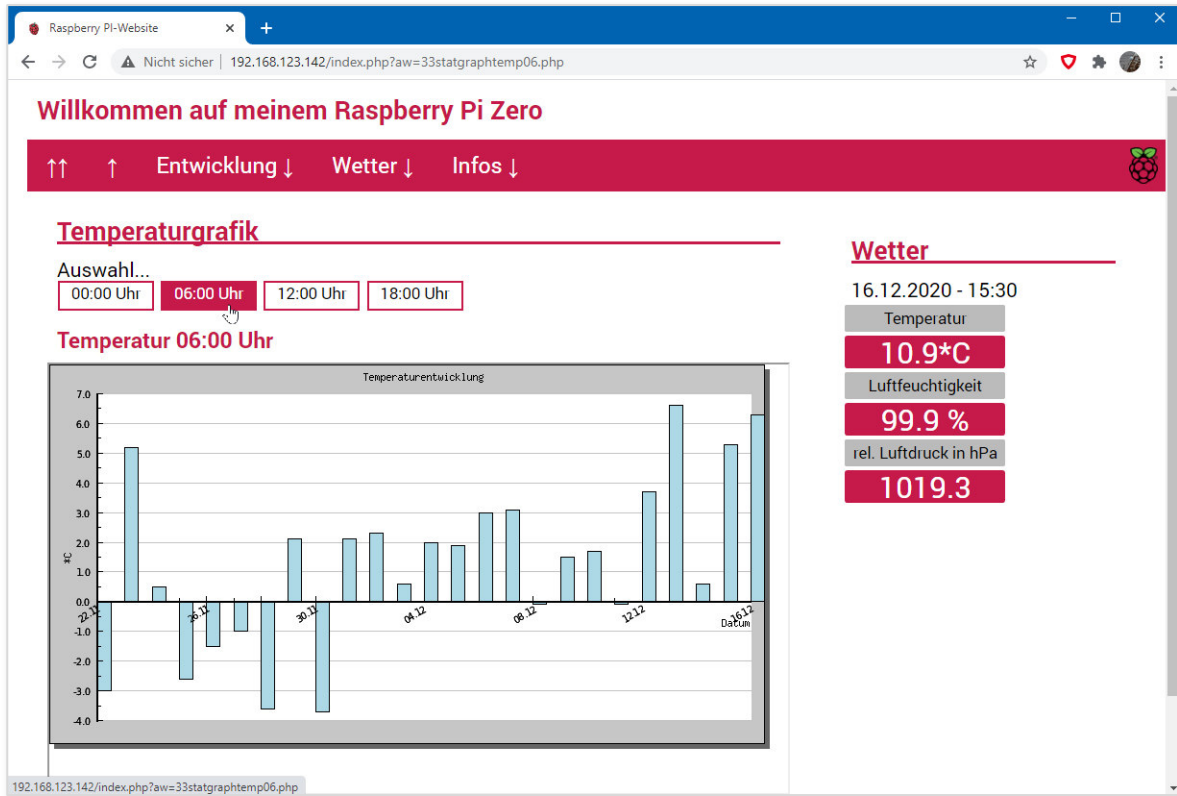


Abbildung 91: Anzeige der Temperaturgrafiken

Alle Dateien für die Aufbereitung und Anzeige der Daten sind im Hauptverzeichnis des Webserver abgelegt und nacheinander abgearbeitet.

```

1 <?php
2 include ('./33statgraphtempmen.php');
3 echo "<h2>Temperatur 00:00 Uhr</h2>";
4 echo "<iframe src='./33statgraphtempwerte00.php' width='100%' height='500px'></iframe>";
5 ?>

```

Programmcode 25: Hauptdatei für die Grafikanzeige

Über mehrere Schaltflächen erfolgt die Auswahl der gewünschten Mess-Uhrzeit.

```

1 <h1>Temperaturgrafik</h1>
2 <p class='textlinks'>Auswahl...<br>
3 <a class='meinbutton meinbutton3' href='./index.php?aw=33statgraphtemp00.php'> 00:00 Uhr </a>
4 <a class='meinbutton meinbutton3' href='./index.php?aw=33statgraphtemp06.php'> 06:00 Uhr </a>
5 <a class='meinbutton meinbutton3' href='./index.php?aw=33statgraphtemp12.php'> 12:00 Uhr </a>
6 <a class='meinbutton meinbutton3' href='./index.php?aw=33statgraphtemp18.php'> 18:00 Uhr </a> </p>

```

Programmcode 26: Untermenü der Grafikanzeige

Nun erfolgt in zwei Stufen die Aufbereitung der Datensätze. Zunächst wird die SQL-String gebildet und alle Datensätze mit der gewünschten Zeit zusammengestellt.

```

1 <?php
2 //JPGGraph einbinden
3 require_once('jpggraph/jpggraph.php');
4 require_once('jpggraph/jpggraph_line.php');
5 require_once('jpggraph/jpggraph_bar.php');
6 //Variablen vorbereiten
7 $x_axis = array();
8 $y_axis = array();
9 $i = 0;
10 $strMesszeit="00:00";
11
12 //Datenbankzugriff
13 include ('../files/zugang/raspizero-dbverbindung.php.inc');
14 $strSQL="SELECT m_messzeitpunkt, DATE_FORMAT(m_messzeitpunkt,'%d.%c') AS m_messzeit, m_temp
15 FROM `tblMessung` ";
16 $strSQL= $strSQL . "WHERE (m_messzeitpunkt LIKE '% " . $strMesszeit . "%')";
17 $strSQL= $strSQL . "ORDER BY `m_messzeitpunkt` ASC;";
18 $recordset=mysqli_query($dbverb,$strSQL);
19 include ('../33statgraphtempwertegrafik.php');
20 ?>

```

Programmcod 27: SQL-String für die Grafikanzeige

Danach kann im Anzeigeskript die Grafik über **jpggraph** – Eigenschaften und -Methoden formatiert und erstellt werden.

```

1 <?php
2 while ($row = mysqli_fetch_array($recordset))
3 {
4     $x_axis[$i] = $row["m_messzeit"];
5     $y_axis[$i] = $row["m_temp"];
6     $i++;
7 }
8 mysqli_close($dbverb);
9 //Graph anzeigen
10 $width = 750;
11 $height = 400;
12 $graph = new Graph($width, $height);
13 $graph->SetScale("lin","auto","auto","auto","auto");
14
15 $graph->title->Set('Temperaturentwicklung');
16
17 $graph->xaxis->title->Set('Datum');
18 $graph->xaxis->SetTickLabels($x_axis);
19 $graph->xaxis->SetFont(FF_ARIAL,FS_NORMAL,8);
20 $graph->xaxis->SetWeight("2");
21 $graph->xaxis->SetLabelAngle(30); // 45 degrees angle
22 $graph->xaxis->SetTextLabelInterval(2);
23 // $graph->xaxis->HideFirstTickLabel();
24
25
26 $graph->img->SetMargin(50,20,30,30); //links, rechts, oben, unten
27
28 $graph->yaxis->title->Set('*C');
29 $graph->yaxis->SetColor("black");
30 $graph->yaxis->SetWeight("2");
31 $graph->yaxis->SetFont(FF_ARIAL,FS_NORMAL,8);
32 // $graph->yaxis->HideFirstTickLabel();
33
34 $graph->SetShadow();
35 $barplot = new BarPlot($y_axis);
36 // $lineplot = new LinePlot($y_axis);
37 // $lineplot->SetColor('red');
38
39 $graph->Add($barplot);
40 $graph->Stroke();
41 ?>

```

Programmcod 28: Anzeigeskript

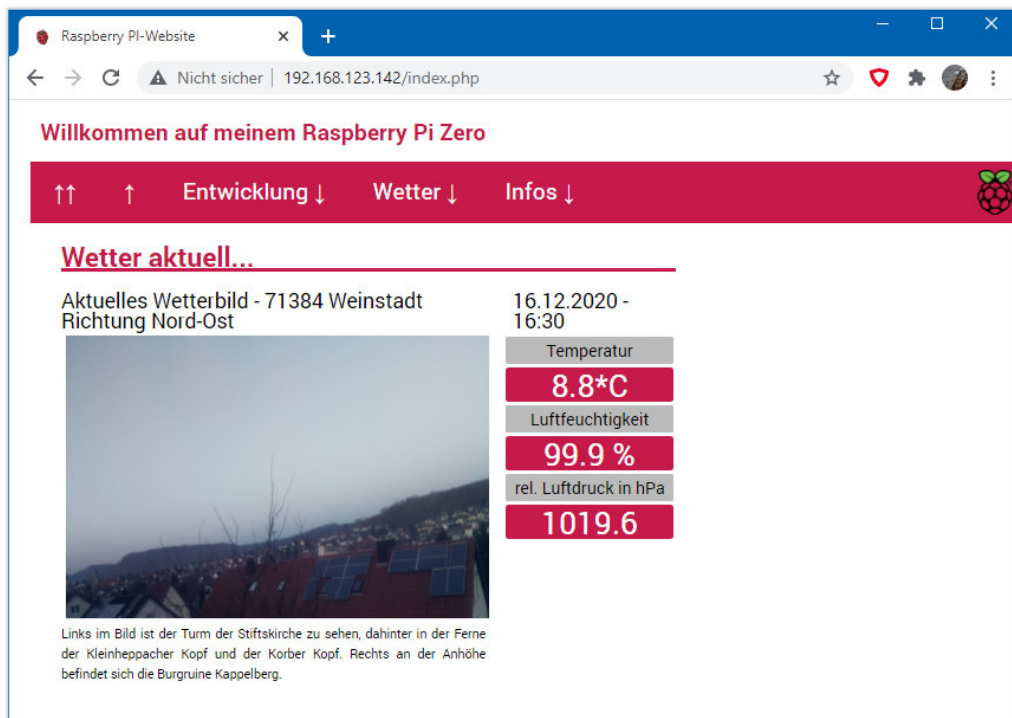
Formatierung der Grafik

Die Darstellung der nun erzeugten Grafik wird in der Anzeigedatei weiter verbessert. Hierzu gibt es im Laufe der Zeit hier noch weitere Ergänzungen.

Anhang-

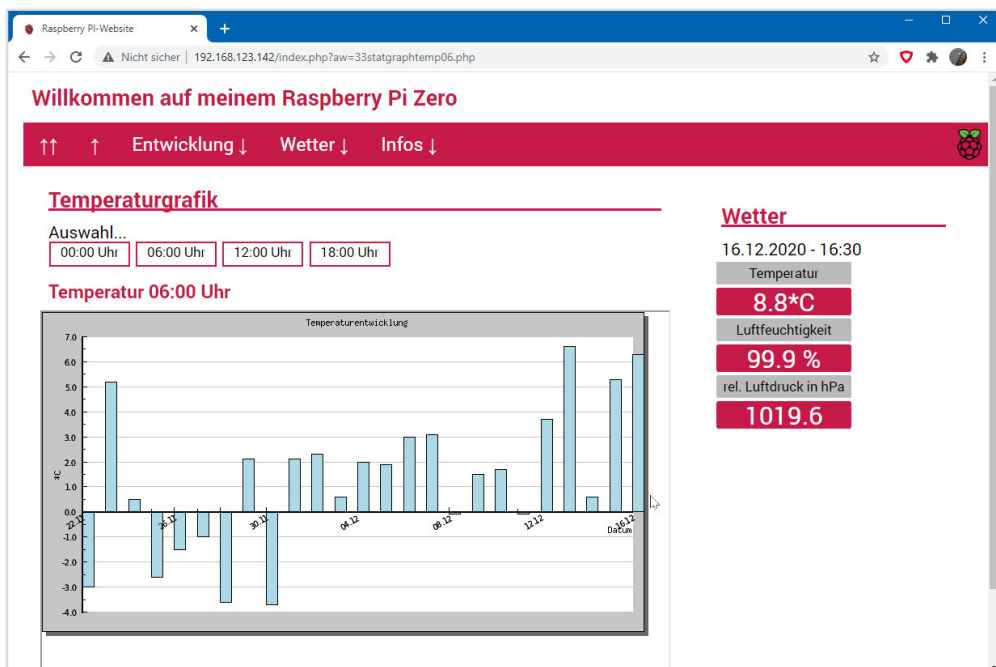
Webseite – Übersicht

Die Abbildung zeigt die fertige Startseite der Wetterstation mit dem Raspberry Pi Zero.



Programmcod 29: fertige Website (Stand 16.12.2020)

Das folgende Beispiel zeigt die Grafikauswertung der Temperaturdaten.



Programmcod 30: grafischer Temperaturverlauf auf der Webseite