Raspberry Pi - Wetterstation

Ausbaustufe 1 - Grundausbau

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

Autor: Hans Koch – Version 10/2019

Vorwort

Die vorliegende Dokumentation soll die Planung, Entwicklung und den Aufbau einer kleinen Wetterstation mit dem Kleincomputer Raspberyy 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 Versuchsinstallationen durchgeführt, 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.

Inhalt

Abbildungsverzeichnis IV
Verzeichnis der Programmcode-Ausschnitte V
Internetquellen VI
Quellangaben 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 Serverinstallation 17
5.1 Vorbereitung 17
5.2 Datenbankserver MariaDB 18
5.3 Webserver Apache 2 installieren 21
5.4 PHP 7 22
5.5 Administrationssoftware für die Datenbank 23
5.6 Datenbank für Wetterdaten erstellen 26
5.7 Entwicklungsstand des Servers 28
6 Website vorbereiten 29
6.1 Planung 29
6.2 Programmierumgebung 30
6.3 Programmierung der Grundstruktur 31
7 Wetterdaten veröffentlichen 39
7.1 Wetterbilder 39
7.2 Temperatur und Luftfeuchtigkeit anzeigen 43
8 Erweiterung der Wetterstation mit einem Luftdrucksensor 47
8.1 Allgemeines 47
8.2 Der BOSCH-Sensor BMP280 48
8.3 mechanischer Umbau 53
8.4 erster Funktionstest im Labor 54
8.5 Software zum Auslesen der BMP280-Sensordaten 56
9 Aufbau „vor Ort“ 62
9.1 Montage 62
9.2 Verbindungstest 63
10 Anpassung der Messzeiten 65
11 Messwerte in der Datenbank ablegen 67
Anhang-Beispielscripte 68

Abbildungsverzeichnis

Abbildung 1: Raspberry Pi Zero ... 2
Abbildung 2: Kamermodul-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 18Raspberry 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-Netzwerkkonfiguration testen ... 13
Abbildung 27: Verbindungstest mit ping und traceroute ... 13
Abbildung 28: Systeminfos auslesen ... 13
Abbildung 29: Kameratestbild ... 14
Abbildung 30: Testbild der Raspberry Zero Kamera ... 14
Abbildung 31: System updaten ... 15
Abbildung 32: Adafruit-Bibliothek herunterladen ... 15
Abbildung 33: Adafruit Python installieren ... 15
Abbildung 34: Messergebisse für Temperatur und Luftfeuchtigkeit ... 16
Abbildung 35: Messergebnisse mit angepasstem Python-Script ... 16
Abbildung 36: /etc/apt/sources.list erweitern ... 17
Abbildung 37: Systemupdate ... 17
Abbildung 38: Installationsablauf MariaDB ... 19
Abbildung 39: Anpassung MariaDB-login ... 20
Abbildung 40: Apache2-Startseite ... 21
Abbildung 41: php-Testseite (Code) ... 22
Abbildung 42: php-Testseite (im Browser) ... 22
Abbildung 43: Konfiguration des blowfish-Passworts ... 24
Abbildung 44: Anmelden an phpMyAdmin ... 24
Abbildung 45: root-Passwort in mysql setzen ... 24
Abbildung 46: Fehlermeldung nach dem Login ... 25
Abbildung 47: Temporären Ordner anlegen und Rechte setzen ... 25
Abbildung 48: Warnhinweis - fehlender Konfigurationsspeicher ... 25
Abbildung 49: Konfigurations-Speicher-Tabellen einrichten ... 25
Abbildung 50: phpMyAdmin - Arbeitsbildscirm ... 26
Abbildung 51: Datenfelder anlegen ... 27
Abbildung 52: Daten einfügen ... 28
Abbildung 53: Datensätze anzeigen ... 28

Abbildung 54: Datenbank-Designer ... 28
Abbildung 55: Seitenlayout - Planung ... 29
Abbildung 56: Programmierumgebung zur Webseitenentwicklung ... 30
Abbildung 57: Inhalt des html-Verzeichnisses ... 31
Abbildung 58: Website - Dateien im Erstentwurf ... 31
Abbildung 59: Media-Query mit Firefox ... 34
Abbildung 60: Webseite bei kleiner Anzeigebreite ... 35
Abbildung 61: Startseite im Browser ... 38
Abbildung 62: Aktuelles Wetterbild ... 39
Abbildung 63: Kamerabild bearbeitet ... 41
Abbildung 64: Wetterdaten auf der Startseite darstellen ... 45
Abbildung 65: Tages-Wetterdaten auf der Webseite ... 46
Abbildung 66: Datenblatt-Titel BMP280 ... 48
Abbildung 67: BMP280 mit Trägerplatine ... 48
Abbildung 68: Blockschaltbild BMP280 (BOSCH Sensortec GmbH, 2018, S. 11) ... 49
Abbildung 69: Messzyklus BMP280 ... 49
Abbildung 70: Anschaltung der Sensorplatine ... 51
Abbildung 71: BMP280 Memory-Map (BOSCH Sensortec GmbH, 2018, S. 24) ... 51
Abbildung 72: I²C Schreibvorgang (BOSCH Sensortec GmbH, 2018, S. 29) ... 52
Abbildung 73: I²C Lesevorgang (BOSCH Sensortec GmbH, 2018, S. 30) ... 52
Abbildung 74: I²C-Timing BMP280 (BOSCH Sensortec GmbH, 2018, S. 33) ... 52
Abbildung 75: Leiterplatte für die Sensoranschaltung ... 53
Abbildung 76: Verdrahtung der Sensoren ... 53
Abbildung 77: Sensor- und Raspberry-Gehäuse ... 54
Abbildung 78: I²C-Sensor ansprechen ... 54
Abbildung 79: I²C-Dump ... 55
Abbildung 80: Controlregister ... 57
Abbildung 81: BMP280 Memory-Map Messregister (BOSCH Sensortec GmbH, 2018, S. 24) ... 59
Abbildung 82: Entmontage der Wetterstation ... 62
Abbildung 83: Testbilder der Webcam ... 63
Abbildung 84: Bandbreitentest mit jperf ... 64
Abbildung 85: crontab anpassen ... 65
Abbildung 86: System-Logdatei ... 65
Abbildung 87: Fehler in der crontab ... 65
Abbildung 88: berichtigte crontab ... 65
Abbildung 89: Abfolge der crontab-Scripts ... 66

Verzeichnis der Programmcode-Ausschnitte

Programmcode 1: index.php - Teil 1 ... 32
Programmcode 2: index.php - Teil 2 ... 32
Programmcode 3: index.php - Teil 3 ... 33
Programmcode 4: css-Styledatei (Ausschnitt) ... 34
Programmcode 5: css-Stryledatei (Bereiche und Media-Queries) ... 35
Programmcode 6: css-Datei (Menüsteuerung) ... 35
Programmcode 7: Kopfbereich ... 36
Programmcode 8: Menü-dropdown ... 36
Programmcode 9: Menübereich 1 ... 37
Programmcode 10: Menübereich 2 ... 37
Programmcode 11: Menübereich 3 ... 37
Programmcode 12: Logodatei ... 38
Programmcode 13: Inhaltsdatei beim Erstaufruf ... 38
Programmcode 14: Wetterbildanzeigedatei ... 39
Programmcode 15: bash-Script zum Erzeugen des aktuellen Wetterbildes ... 40

Programmcode 16: bash-Script für die Stundenbilder 41
Programmcode 17: crontab-Konfiguration 42
Programmcode 18: Bildverzeichnis 42
Programmcode 19: bash-Script zum Auslesen der Sensordaten (Teil 1) 43
Programmcode 20: bash-Script zum Auslesen der Sensordaten (Teil 2) 44
Programmcode 21: bash-Script zum Auslesen der Sensordaten (Teil 3) 44
Programmcode 22: bash-Script zum Auslesen der Sensordaten (Teil 4) 44
Programmcode 23: Wetterdaten auf die Startseite einfügen 45
Programmcode 24: Tages-Wetterdaten anzeigen 46
Programmcode 25: Kopfdaten des Python-Scripts 56
Programmcode 26: Kalibrierung BMP280 56
Programmcode 27: BMP280 - Konfigurationsdaten schreiben 57
Programmcode 28: BMP280 - Bitverschiebung 60
Programmcode 29: Berechnung der Messdaten 61
Programmcode 30: BMP280 - Messwerte ausgeben 61

Internetquellen

Grafik GPIOs

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

Quellangaben der verwendeten Grafiken

	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)
Linux-UBUNTU 	Grafik gemeinfrei Datei: https://de.wikipedia.org/wiki/Ubuntu#/media/File:Ubuntu_logo.svg
Webserver Apache 	Datei: https://de.wikipedia.org/wiki/Apache_Software_Foundation#/media/File:Apache_Software_Foundation_Logo_(2016).svg 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.
PHP Skriptsprache  Datenbankserver 	Datei php: https://de.wikipedia.org/wiki/PHP#/media/File:PHP-logo.svg Datei DB: https://de.wikipedia.org/wiki/MariaDB#/media/File:MariaDB_Logo.png Datei : https://de.wikipedia.org/wiki/PhpMyAdmin#/media/File:PhpMyAdmin-Logo.svg Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) (Creative Commons PoBox 1866, Mountain View, CA 94042) 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.

DB-Admin 	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. ShareAlike — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original. No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits. https://creativecommons.org/licenses/by-sa/4.0/legalcode
WinSCP 	Datei: https://de.wikipedia.org/wiki/WinSCP#/media/File:WinSCP_Logo.png Lizenz: https://www.gnu.org/licenses/gpl-3.0.html
PuTTY 	Datei: https://de.wikipedia.org/wiki/PuTTY#/media/File:PuTTY_icon_128px.png Lizenz: https://opensource.org/licenses/mit-license.php
RaspBerryPi 	Lizenzen: https://www.raspberrypi.org/trademark-rules/ https://www.raspberrypi.org/
WebAdmin 	Datei: http://www.webmin.com/graphics/webmin-full.jpg Lizenz: http://www.webmin.com/graphics.html
	Grafik gemeinfrei Datei: https://upload.wikimedia.org/wikipedia/commons/c/c5/Nginx_logo.svg Autor: https://commons.wikimedia.org/w/index.php?curid=24774395

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.

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.

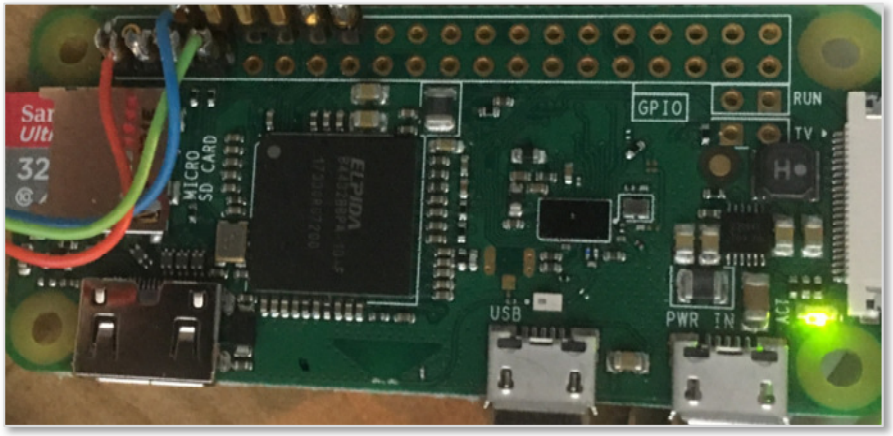


Abbildung 1: Raspberry Pi Zero

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.

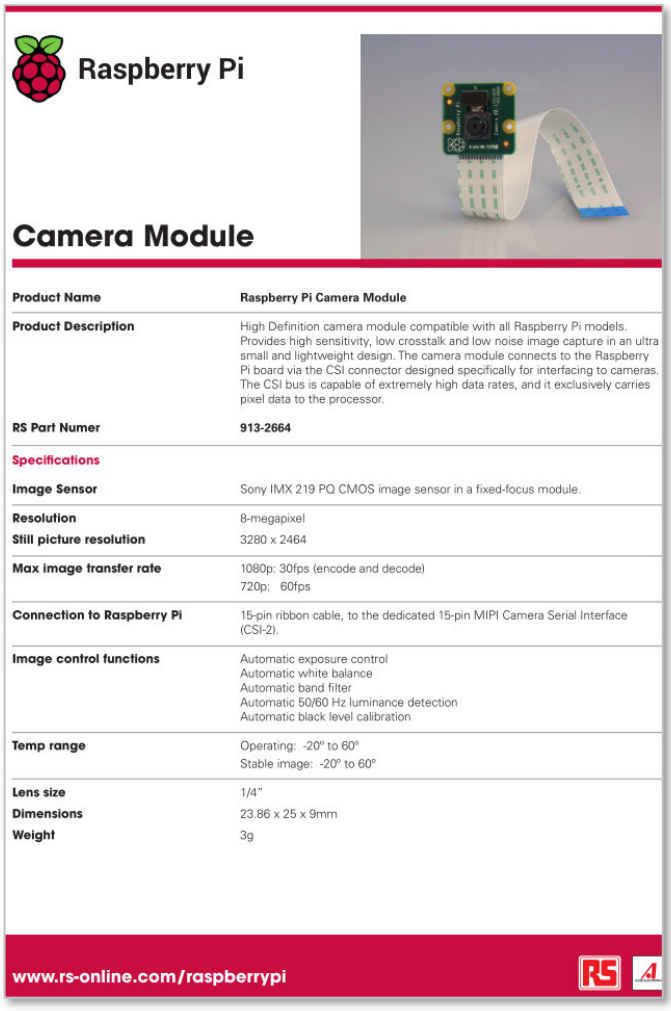
Der Name ist das englische Wort für Himbeerkuchen. Damit wird in der Tradition von Apple oder Acorn verfahren, die Computer nach Früchten benannt haben. Die Erweiterung „Pi“ soll für Python-Interpreter stehen, da ursprünglich gedacht war, einen fest eingebauten Interpreter für die Programmiersprache 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 angeschaltet. 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 Numer	**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: Kamermodul-Datenblatt[1]

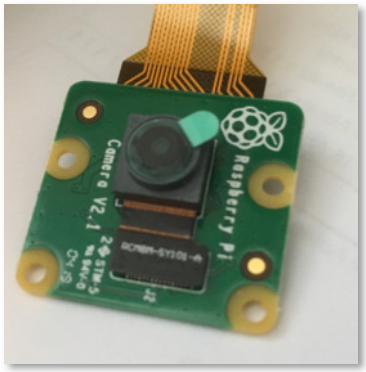


Abbildung 3: Raspi-Kamera V2.1[2]

[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-Protkoll** 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 Adafruit[3]. 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 DHT22[4]

Aosong Electronics Co.,Ltd,
Your specialist in innovating humidity & temperature sensors

Abbildung 4: Sensor DHT225

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

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.

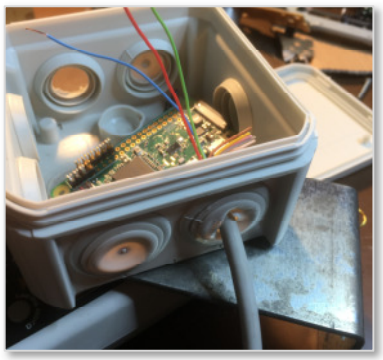


Abbildung 7: Sensorleitung[6]

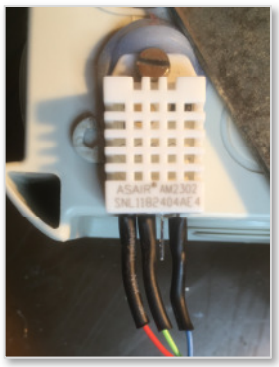


Abbildung 8: Sensormontage

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

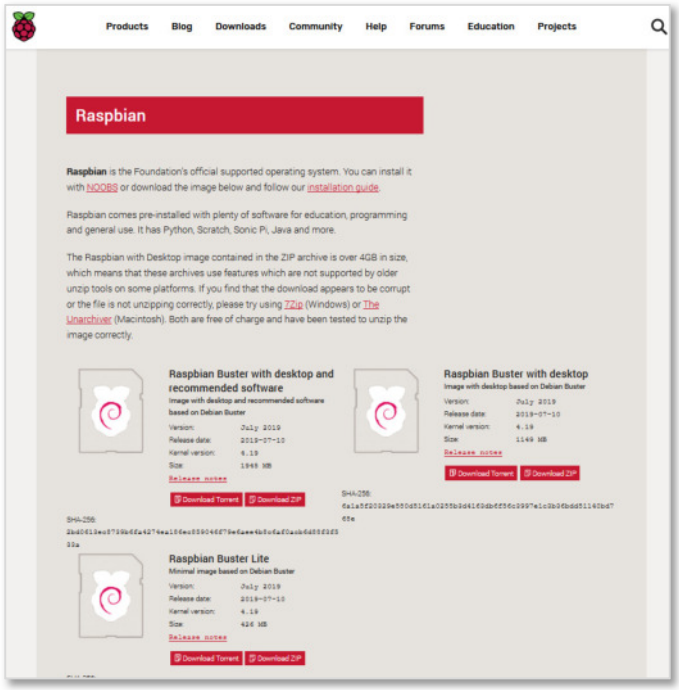


Abbildung 11: Downloadseite (https://www.raspberrypi.org/downloads/raspbian/)

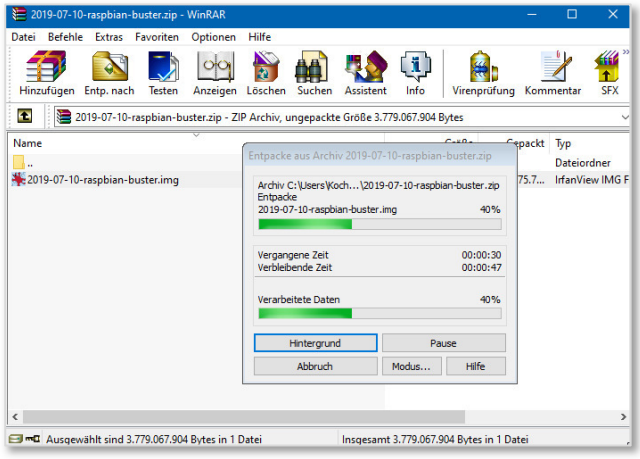


Abbildung 12: Image entpacken

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.

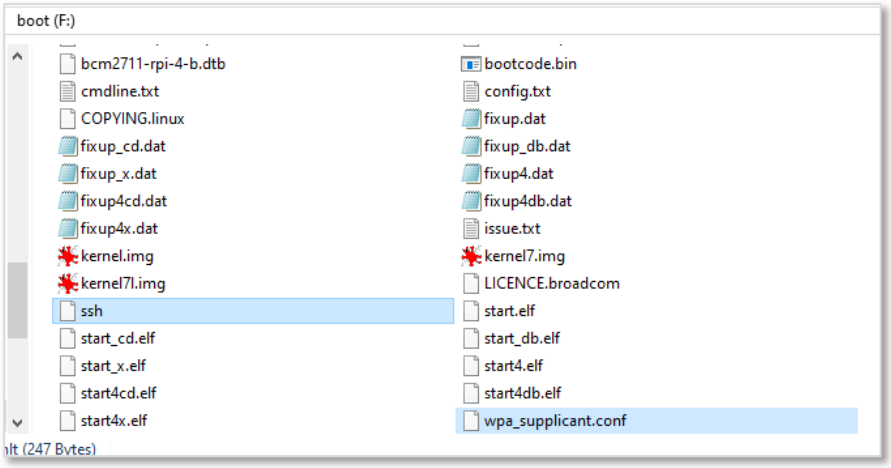


Abbildung 13: Bootpartition

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.

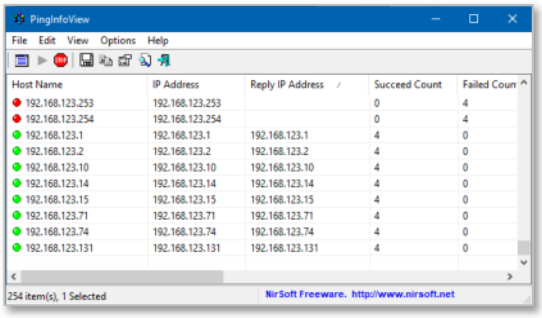


Abbildung 14: IP-Adresse ermitteln

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.

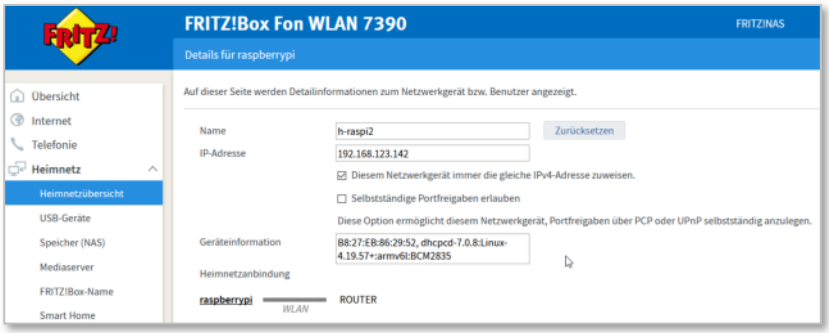


Abbildung 15: feste IP-Adresse für den Raspberry Zero

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.

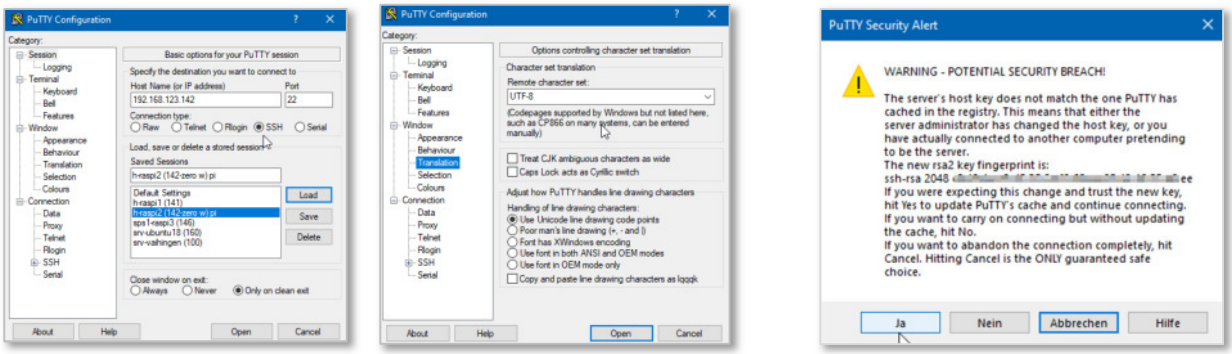


Abbildung 16: erste Remoteverbindung

3.4 Ersteinrichtung

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

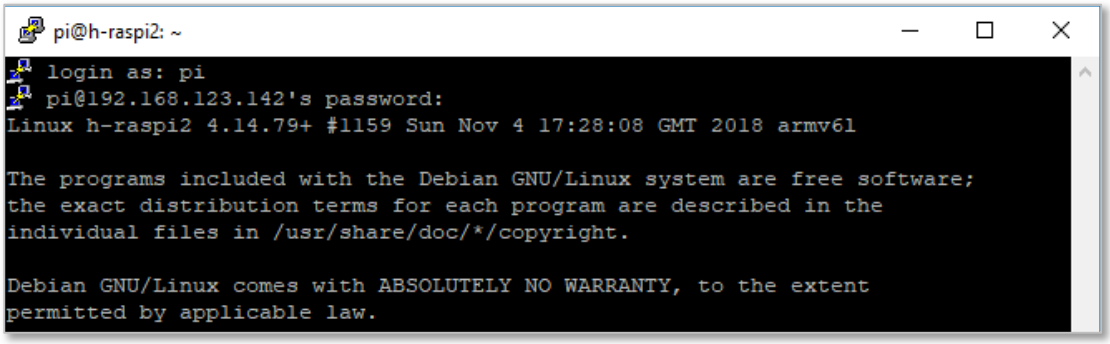


Abbildung 17: Anmeldung am System

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.

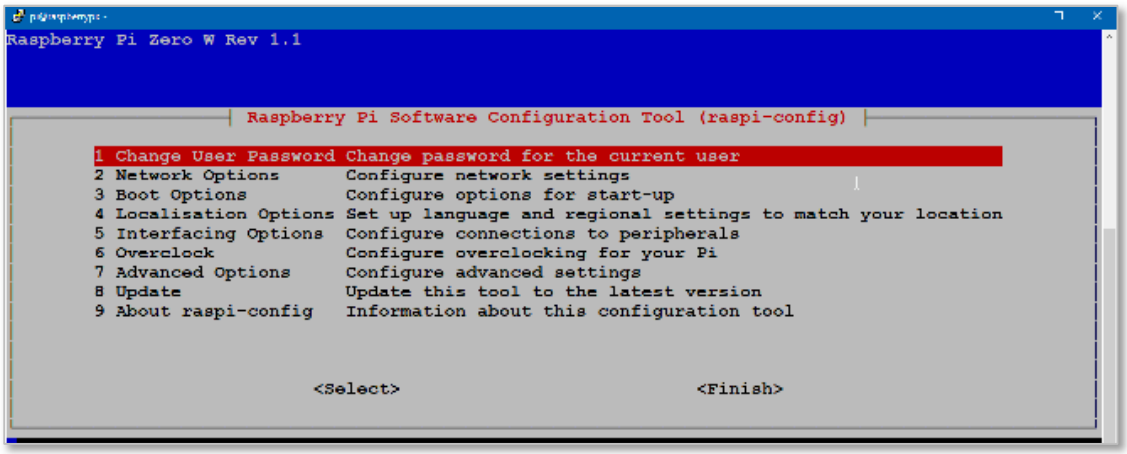


Abbildung 18Raspberry PI Software Configuration Tool

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.

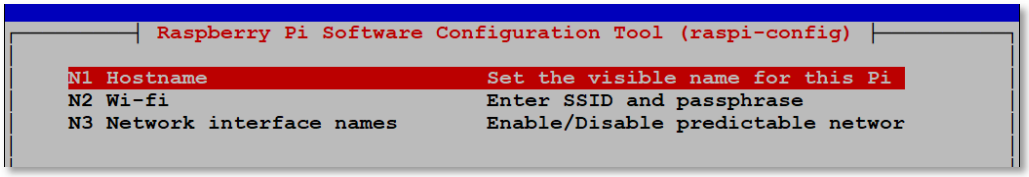


Abbildung 19: Lokale Einstellungen

- 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.

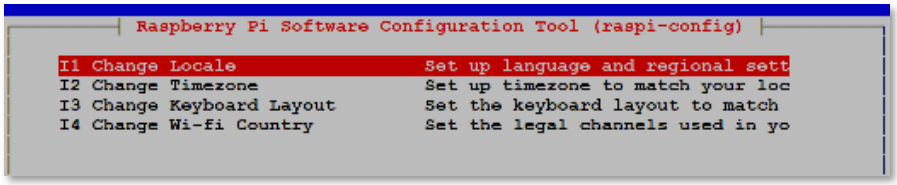


Abbildung 20: raspi-config / Ländereinstellung

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

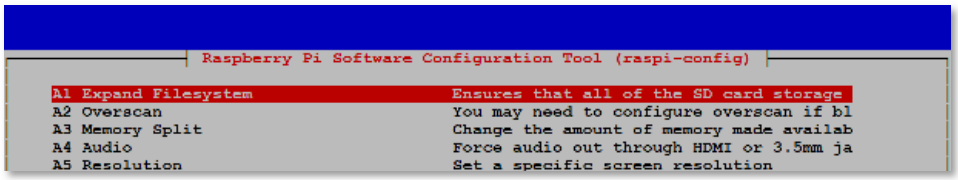


Abbildung 21: Dateisystem optimieren

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

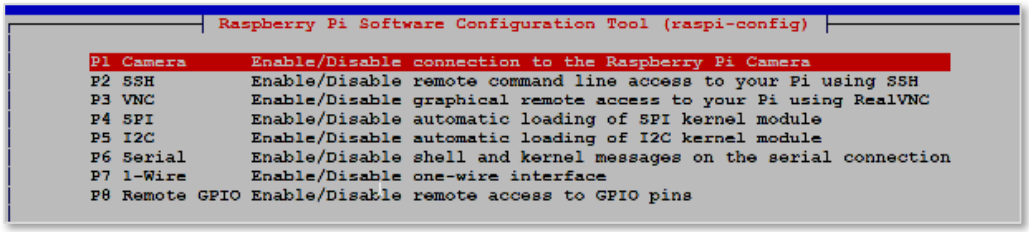


Abbildung 22: Interface aktivieren

Änderungen übernehmen und neustarten

Damit alles aktiviert wird, ist ein Neustart notwendig.

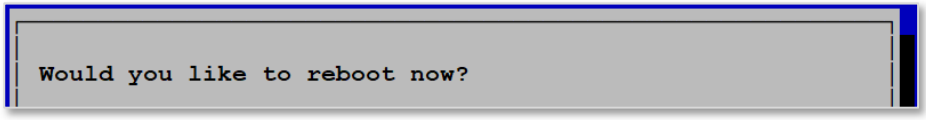


Abbildung 23: Neustart

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:

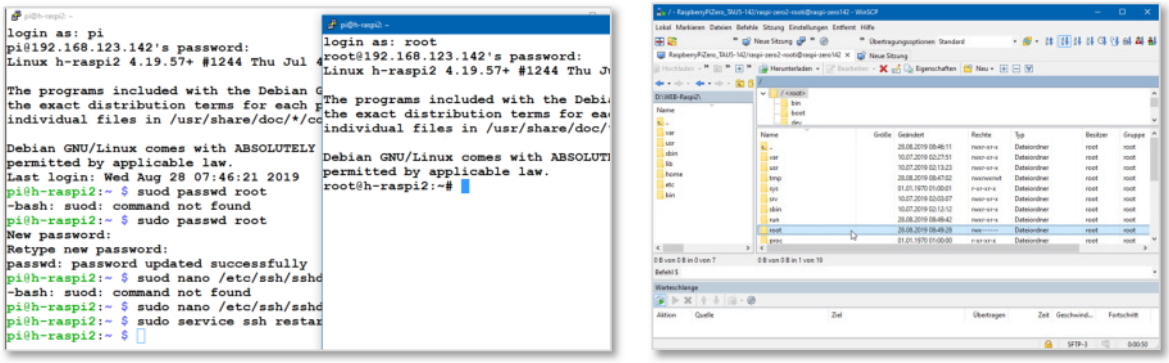


Abbildung 25: root-Anmeldung mit PuTTY und WinSCP

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 adress`** (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_fas
    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 noprefi
       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-Netzwerkkonfiguration 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

```
pi@h-raspi2:~ $ raspistill -o /home/pi/testbild.jpg
pi@h-raspi2:~ $ ls -l
total 4076
drwxr-xr-x 2 pi pi    4096 Aug 25 13:39 Desktop
drwxr-xr-x 2 pi pi    4096 Aug 25 13:39 Documents
drwxr-xr-x 2 pi pi    4096 Aug 25 13:39 Downloads
drwxr-xr-x 2 pi pi    4096 Jul 10 01:15 MagPi
drwxr-xr-x 2 pi pi    4096 Aug 25 13:39 Music
drwxr-xr-x 2 pi pi    4096 Aug 25 13:39 Pictures
drwxr-xr-x 2 pi pi    4096 Aug 25 13:39 Public
drwxr-xr-x 2 pi pi    4096 Aug 25 13:39 Templates
-rw-r--r-- 1 pi pi 4136017 Aug 26 09:11 testbild.jpg
drwxr-xr-x 2 pi pi    4096 Aug 25 13:39 Videos
pi@h-raspi2:~ $
```

Abbildung 29: Kameratestbild

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 30: 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 31: 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.

```
$ 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 32: Adafruit-Bibliothek herunterladen

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

```
creating 'dist/Adafruit_DHT-1.4.0-py2.7-linux-armv6l.egg' and adding
removing 'build/bdist.linux-armv6l/egg' (and everything under it)
Processing Adafruit_DHT-1.4.0-py2.7-linux-armv6l.egg
Copying Adafruit_DHT-1.4.0-py2.7-linux-armv6l.egg to /usr/local/lib/p
Adding Adafruit-DHT 1.4.0 to easy-install.pth file

Installed /usr/local/lib/python2.7/dist-packages/Adafruit_DHT-1.4.0-p
Processing dependencies for Adafruit-DHT==1.4.0
Finished processing dependencies for Adafruit-DHT==1.4.0
pi@h-raspi2:~/Adafruit_Python_DHT $
```

Abbildung 33: Adafruit Python installieren

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 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 34: Messergebisse für Temperatur und Luftfeuchtigkeit

Ein Vergleich mit einem konventionellen Digital-Thermometer zeigt eine geringe Abweichung an.

Laut Datenblatt des Sensors dürfen alle drei Sekunden Abfragen durchgeführt werden. Kürzere Intervalle sind nicht möglich.

Python-Skript anpassen

Mit dem folgenden Skript soll der Sensor für eigene Anwendungen ausgelesen werden. Das von Adafruit mitgelieferte Beispielskript wurde hierfür geringfügig abgeändert, kommentiert und erweitert.

sensorabfrage2019.py

```
1  #!/usr/bin/python
2  import Adafruit_DHT
3  import time
4  # Sensorwerte: Adafruit_DHT.DHT11, Adafruit_DHT.DHT22 oder Adafruit_DHT.AM2302
5  strSensor = Adafruit_DHT.DHT22
6  #Pin am RaspiZero
7  intPin = 4
8  #Sensor auslesen
9  lngFeuchte, lngTemp = Adafruit_DHT.read_retry(strSensor, intPin)
10 #Aktueller Messzeitpunkt
11 datAktuell=time.strftime('%d.%m.%Y - %H:%M')
12 datAktuellZeit=time.strftime('%H')
13 #Werte Testen
14 if lngFeuchte is not None and lngTemp is not None:
15     print('Temperatur={0:0.1f}*C  Feuchtigkeit={1:0.1f}%'.format(lngTemp, lngFeuchte))
16 else:
17     print('Keine Messdaten erhalten - bitte nochmals versuchen...')
18 print(datAktuell)
```

Der Test ergibt die Anzeige der Sensorwerte und der aktuellen Uhrzeit. Ein Vergleich mit einem konventionellen Thermometer bestätigt den Messwert des Sensors mit geringer Abweichung.

```
$ sudo python sensorabfrage2019.py
```

```
pi@h-raspi2:~ $ sudo python sensorabfrage2019.py
Temperatur=26.8*C  Feuchtigkeit=59.8%
27.08.2019 - 07:52
```

Abbildung 35: Messergebnisse mit angepasstem Python-Script

5 Serverinstallation

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

5.1 Vorbereitung

Ale 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 36: /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 Package
s [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 37: Systemupdate

5.2 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 Produktivsysteme 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.[7]

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] Y
- Remove test database and access to it? [Y/n] Y
- Reload privilege tables now? [Y/n] Y

[7] 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 38: Installationsablauf MariaDB

Das Einloggen auf der Shell funktioniert noch nicht. Es muss das Plugin-Login deaktiviert 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 39: 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. Im Erstausbau der Raspberry Pi-Wetterstation kam der ressourcenschonende Webserver Nginx zum Einsatz.

Im zweiten Ausbau soll nun der weit verbreitete Webserver Apache2 installiert werden.

5.3 Webserver Apache 2 installieren

Der Webserver dienst als Dienst, um Webseiten auf dem RaspberryPI zu organisieren und auf Anforderung des Besuchers auszuliefern.

Allgemeines zur Software[8]

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.

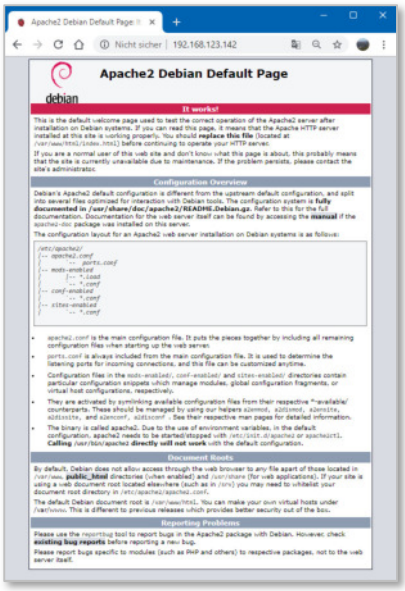


Abbildung 40: Apache2-Startseite

[8] Quelle: aus https://de.wikipedia.org/wiki/Apache_HTTP_Server

5.4 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:~ $
```

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 41: php-Testseite (Code)

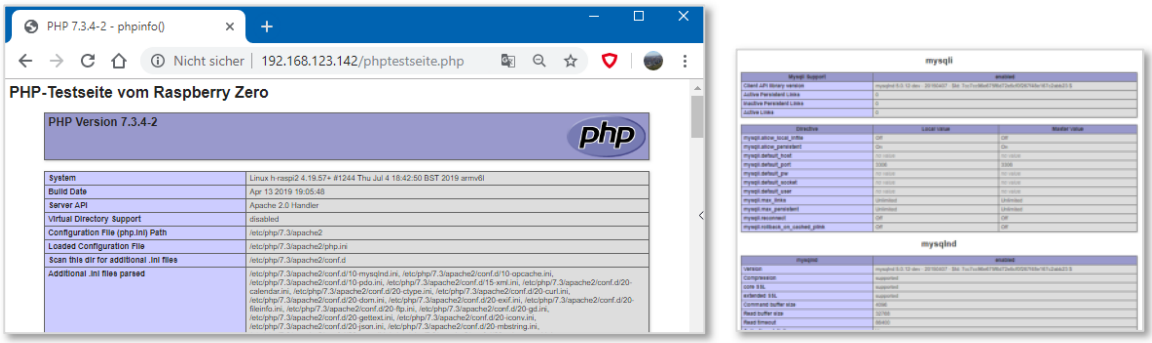


Abbildung 42: php-Testseite (im Browser)

5.5 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.

Wechseln in das Webverzeichnis

```
$ cd /var/www/html/
```

Download des ZIP-Archivs

```
$ sudo wget https://files.phpmyadmin.net/phpMyAdmin/4.9.0.1/phpMy-
   Admin-4.9.0.1-all-languages.zip
```

Entpacken

```
$ sudo unzip phpMyAdmin-4.9.0.1-all-languages.zip
```

Verzeichnis umbenennen

```
$ sudo mv phpMyAdmin-4.9.0.1-all-languages pma
```

Anschließend kann die heruntergeladene ZIP-Datei gelöscht werden.

```
$ sudo rm phpMyAdmin-4.9.0.1-all-languages.zip
```

Konfiguration

Nun wird für phpMyAdmin ein geheimes Passwort gesetzt, das einmalig in die Konfigurationsdatei 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 43).

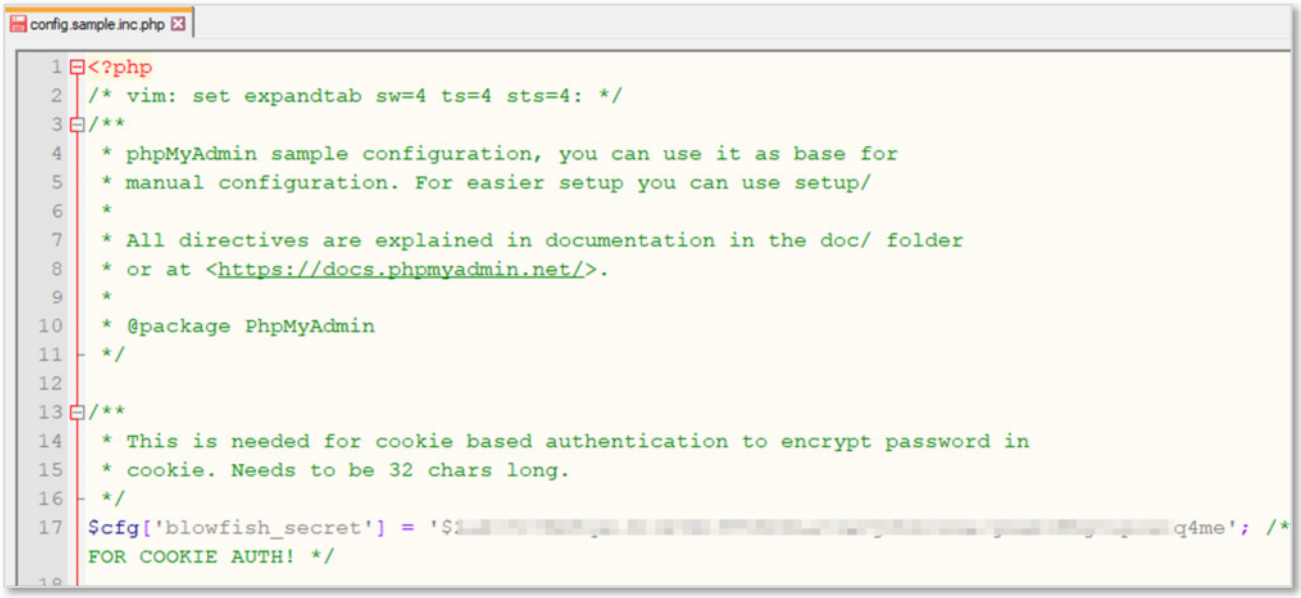


Abbildung 43: Konfiguration des blowfish-Passworts

Die Datei muss nun noch umbenannt werden.

```
$ sudo mv pma/config.sample.inc.php pma/config.inc.php
```

Nun ist im Unterverzeichnis **`pma`** die Startseite des phpMyAdmin 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 44.

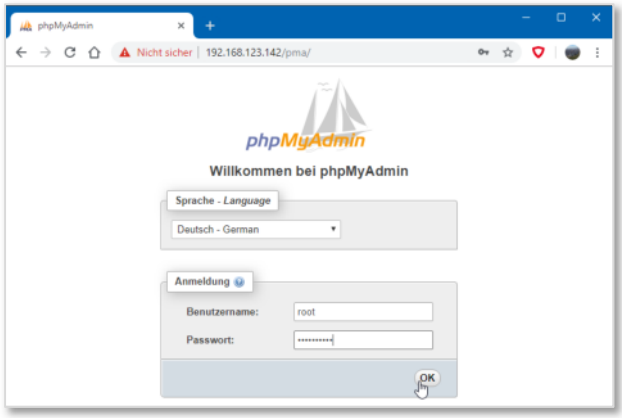


Abbildung 44: Anmelden an phpMyAdmin

Sollte ein Zugriffsfehler auftreten, liegt dies meist am fehlenden oder fehlerhaften **`root`**-Passwort. Dies kann durch Anmelden an der 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('[redacted]') 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 45: root-Passwort in mysql setzen

Anmeldung und Fehlerkorrektur

Nach der erfolgreichen Anmeldung gilt es eventuelle Fehlermeldungen zu beachten. Ein im Moment bekannter Fehler besteht darin, dass im phpMyAdmin-Verzeichnis kein **tmp**-Ordner angelegt wurde. Dies zeigt die rot unterlegte Anzeige.

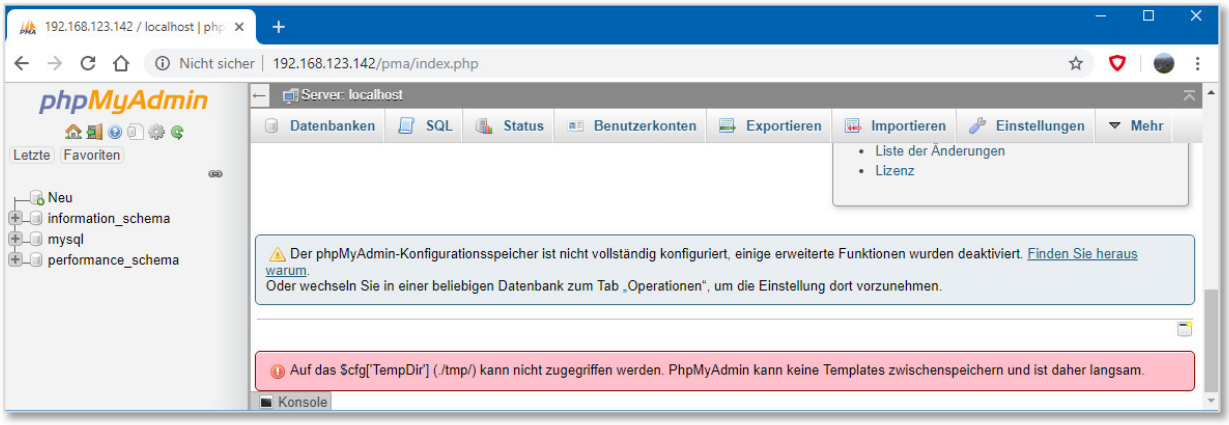


Abbildung 46: Fehlermeldung nach dem Login

Durch Anlegen eines **tmp**-Ordners und Setzen der Zugriffsrechte ist dieser Fehler behoben.

```
pi@h-raspi2:/var/www/html/pma $ cd /var/www/html/pma
pi@h-raspi2:/var/www/html/pma $ sudo mkdir tmp
pi@h-raspi2:/var/www/html/pma $ sudo chmod 777 tmp
pi@h-raspi2:/var/www/html/pma $
```

Abbildung 47: Temporären Ordner anlegen und Rechte setzen

Ein weiterer Hinweis nach der Anmeldung bezieht sich auf deaktivierte Funktionen.

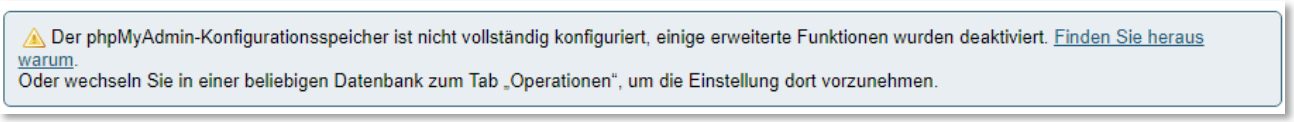


Abbildung 48: Warnhinweis - fehlender Konfigurationsspeicher

Durch Anklicken des Links und Durchführen der Datenbankaktion kann auch dieser Fehler behoben werden.

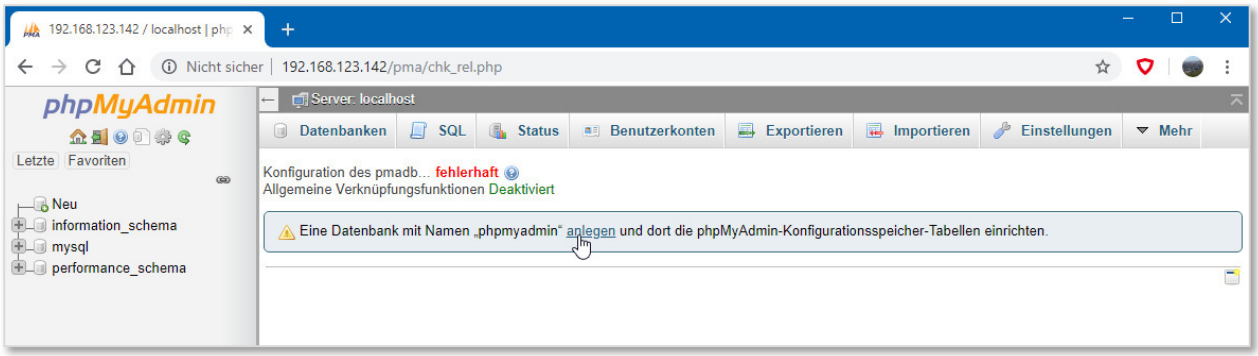


Abbildung 49: Konfigurations-Speicher-Tabellen einrichten

phpMyAdmin – Bildschirmaufbau

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

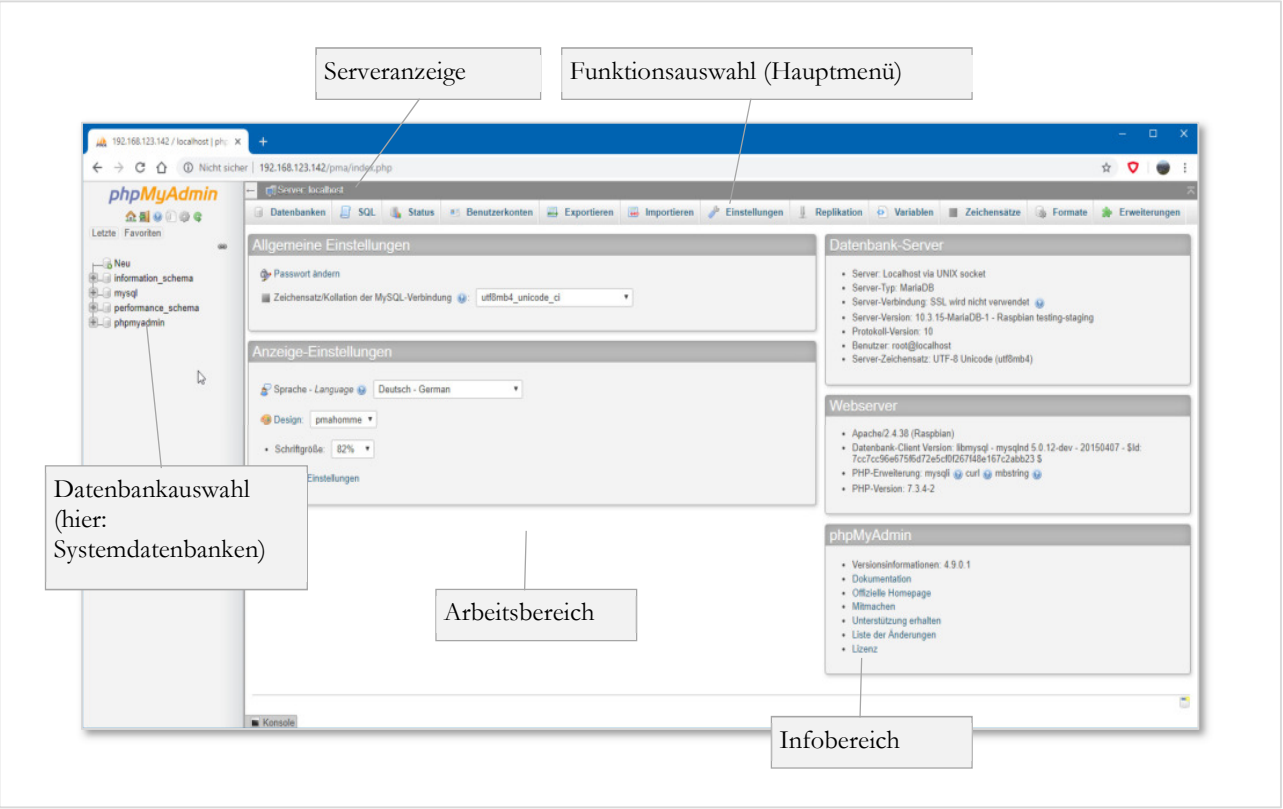


Abbildung 50: phpMyAdmin - Arbeitsbildscirm

5.6 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 Bildschirmausschnitt zeigt die Vorgehensweise.

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.

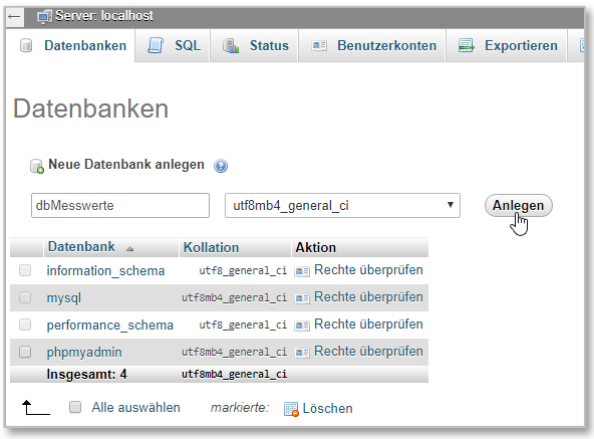


Tabelle erzeugen

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.

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.

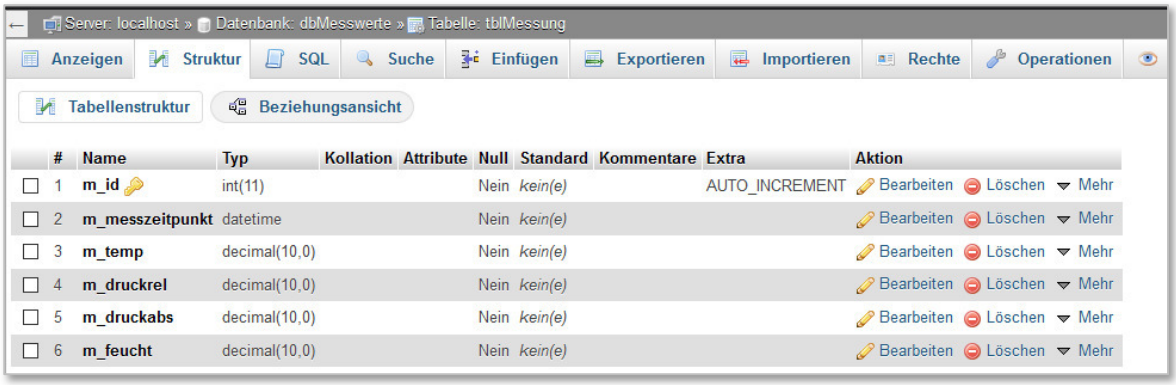


Abbildung 51: Datenfelder anlegen

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)

Testdaten eingeben

Zu Testzwecken werden zwei Daten manuell in die Felder eingegeben:

Abbildung 52: Daten einfügen

Die abschließende Anzeige ergibt folgende Liste:

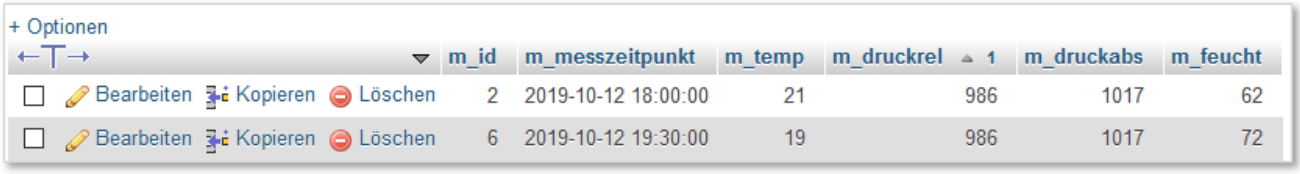


Abbildung 53: Datensätze anzeigen

Datenbank erweitern

Die Datenbank wird nun um eine zweite Tabelle erweitert, um eine Besucherstatistik speichern zu können. Beide Tabellen haben keine Verknüpfung miteinander. Der Datenbank-Designer zeigt in der Übersicht die Tabellenkonstruktion an.

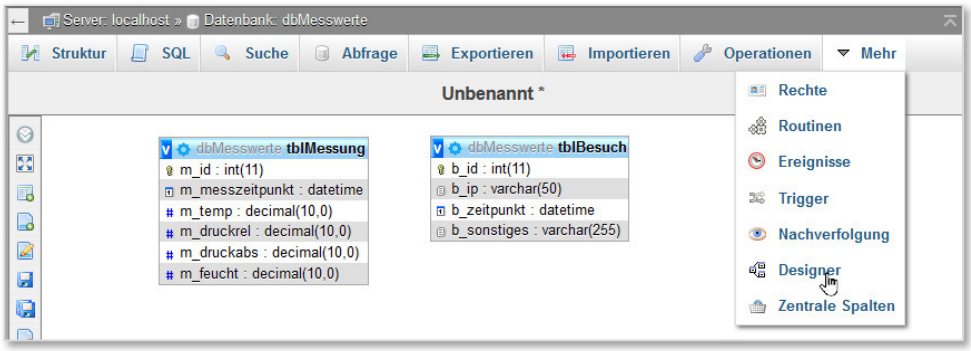


Abbildung 54: Datenbank-Designer

5.7 Entwicklungsstand des Servers

Nun sind alle wichtigen Serverdienste implementiert. Hier in der Übersicht und in der Installationsreihenfolge die einzelnen Komponenten.

- Datenbankmanagementsystem MariaDB (Version 10.3.15)
  Server Zeichensatz UTF-8 Unicode (utf8mb4)
- Webserver Apache2 (Version 2.4.38)
  PHP-Erweiterungen: mysqli, curl, mbstring
- PHP-serverbasierte Skriptsprache in Version 7.3.4-2
- Administrationstool phpMyAdmin (Version 4.9.0.1)

Im nächsten Abschnitt wird nun die Webseite zur Darstellung der Messwerte geplant und umgesetzt.

6 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.

6.1 Planung

Zunächst wird ein Seitenlayout vorbereitet. Dies soll in einer zentralen CSS-Datei definiert werden. Um eine möglichst responsable 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.

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.
- **`raspi2019.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.
- **`xxxx.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.

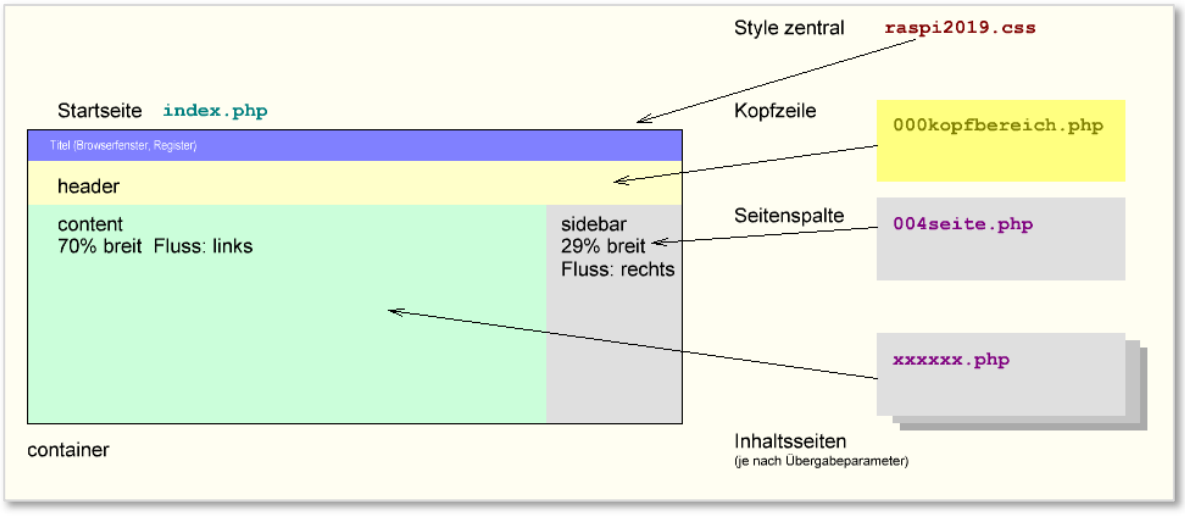


Abbildung 55: Seitenlayout - Planung

6.2 Programmierumgebung

Zur Organisation der Programmierumgebung wird auf einem Windows-PC dieselbe Ordnerstruktur angelegt, wie sie auf dem RaspberryPI 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\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-Script darf darauf zugreifen.

- D:\WEB-Raspi2**\var\www\cgi-bin**
  Hier werden Scripte abgelegt, die ebenfalls nur von einem php-Script benutzt werden können.

Mit dem Windowstool WinSCP können in Zweifensterdarstellung optimal der Windows-Entwicklerrechner und der RaspberryPI dargestellt werden. Der Datenaustausch wird damit erleichtert.

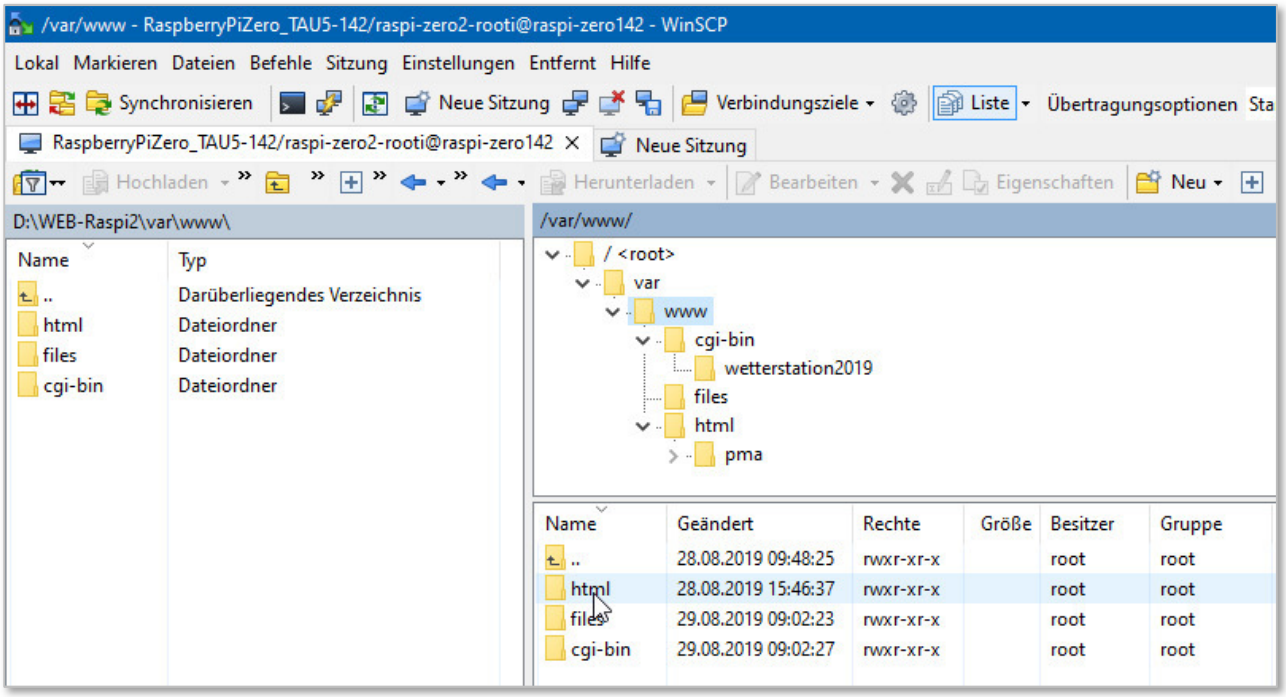


Abbildung 56: Programmierumgebung zur Webseitenentwicklung

Im Moment befinden sich im **html**-Verzeichnis lediglich das Unterverzeichnis **pma** für phpMyAdmin und die bisher benutzten Testdateien für den Webserver und PHP.

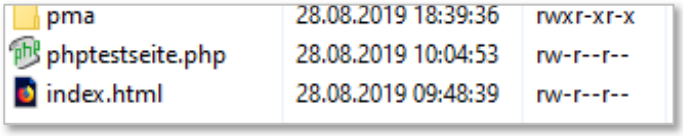


Abbildung 57: Inhalt des html-Verzeichnisses

6.3 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 **raspi2019.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.

Die folgende Abbildung 58 zeigt den aktuellen Inhalt des Webverzeichnisses mit Hilfe des WinSCP-Remotezugriffs auf das Dateisystem des RaspberryPi zero. Unten werden die einzelnen Dateien und deren Code näher erläutert.

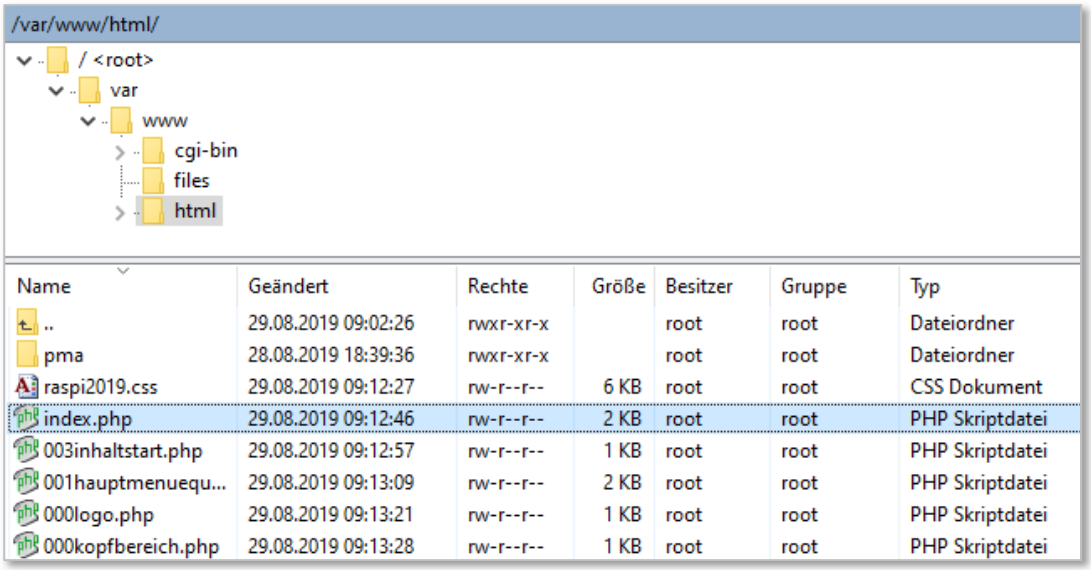


Abbildung 58: Website - Dateien im Erstentwurf

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, je nach Menüwahl.

Im Kopfbereich wird zunächst durch die Cookiesteuerung ein Besucherzähler vorbereitet (1), zusätzlich wird der Zeichensatz der Seite auf UTF-8 festgelegt (2). Dies ist in manchen Fällen notwendig, damit sicher der korrekte Zeichensatz dargestellt wird. Der eigentliche html5-Code beginnt in Zeile (5). In **<head>** wird nochmals UTF-8 definiert (8), dann folgt der Zugriff auf die Styledatei raspi2019.css (9) und in Zeile (10) wird der Start-Viewport eingestellt.

Die Zuweisung des Favicons (11) ist durch Verwendung des Standardnamen **favicon.ico** nicht unbedingt notwendig, dient hier lediglich zu Demonstration. Zeile (12) beschreibt den Titel des Browserfensters.

```
1  <?php
2  setcookie("raspi2","besucht",time()+(60*60*24));//60Sekunden*60Minuten*12Stunden
3  header('Content-Type: text/html;charset=utf-8');
4  ?>
5  <!DOCTYPE HTML>
6  <html>
7  <head>
8      <meta charset="UTF-8">
9      <link href="./raspi2019.css" rel="stylesheet" type="text/css">
10     <meta name="viewport" content="width=device-width, initial-scale=0.8, user-scalable=yes">
11     <link href="favicon.ico" rel="shortcut icon">
12     <title>Raspberry PI-Website</title>
13 </head>
```

Programmcode 1: 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 **$_GET**-Parameter. Hier wird dieser übregebene Parameter in die Variable **$strZiel** gespeichert (20). Im Beispiel wird der Parameterwert **aw=31wetterdatentag.php** an die Datei **index.php** übergeben. Beim ersten Aufruf der Startseite muss als Ziel die Seite **003inhaltstart.php** geladen werden (24).

http://192.168.123.142/index.php?aw=31wetterdatentag.php

```
16      <?php
17          //Übergabeparameter auslesen: Zieldatei für content
18          if(isset($_GET['aw']))
19              {
20              $strZiel=$_GET['aw'];
21              }
22          else
23              {
24              $strZiel='003inhaltstart.php'; //Startdatei einfügen, beim ersten Aufruf
25              }
```

Programmcode 2: index.php - Teil 2

Im nächsten Abschnitt werden die Layout-Bereiche ** ** 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**-Auswahl (42).

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

Programmcode 3: 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  raspi2019.css*/
2  *{
3      font-family : Roboto, Verdana , Aria
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     }
```

```
24 #meinlogo {
25     display: block;
26     padding: 0.1rem;
27     float: right;
28     }
29
30 #container {
31     margin: 0 ;
32     max-width: 1600px;
33     padding:1rem 1rem 5rem 1rem;
34     }
35
36 #content{
37     margin: 0 ;
38     padding:1rem 1rem 5rem 1rem;
39     float:left;
40     width: 100%;
41     }
42
43 #sidebar{
44     display:none;
45     }
```

Programmcode 4: 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 59), hier 770px Breite..

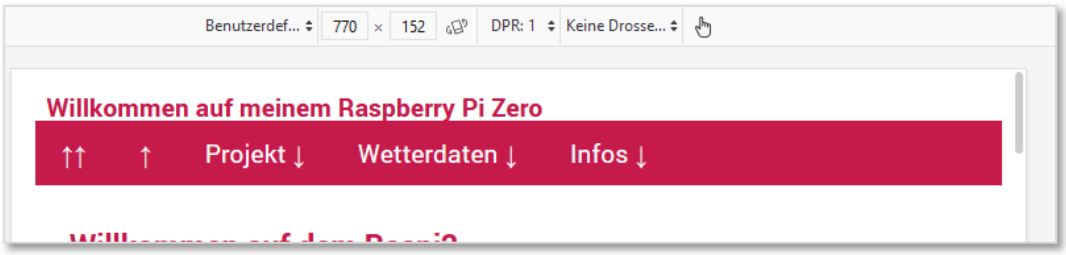


Abbildung 59: Media-Query mit Firefox

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%.

```
/*----------------------------------------
 Media Queries
----------------------------------------*/
@media all and (max-width : 4800px) {
    html{
        font-size:150%;
        }
    .titel
        {
        font-size: 1.5rem;
        }
#content{
    margin: 0 ;
    padding:1rem 1rem 5rem 1rem;
    float:left;
    width: 70%;
    }

#sidebar{
    margin: 0 ;
    float:right;
    width:29%;
    padding:1rem 1rem 5rem 1rem;
    display:block;
    }
}
```

```
@media all and (max-width : 1400px) {
    html{
        font-size:130%;
        }
    .titel
        {
        font-size: 1.3rem;
        }
}

@media all and (max-width : 1000px) {
    html{
        font-size:110%;
        }
    .titel
        {
        font-size: 1.1rem;
        }
}
```

Programmcode 5: 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 60).

Abbildung 60: Webseite bei kleiner Anzeigebreite

```
@media all and (max-width : 768px) {
    html{
        font-size:100%;
        }
    .titel
        {
        font-size: 0.9rem;
        }
nav {
        margin: 0;
    }
    /* Navigation verstecken */
    .toggle + a, .menu {
        display: none;
        }
    /* Style bei der Menütabelle
    für kleine Auflösung */
    .toggle {
        display: block;
```

Programmcode 6: css-Datei (Menüsteuerung)

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:

```
<?php
    echo "<p class='titel'>Willkommen auf meinem Raspberry Pi Zero</p>";
?>
<div id="meinlogo">
    <?php include "./bilderallgemein/00logo.php";?>
</div>
<?php
    include "./001hauptmenuequer.php";
?>
```

Programmcode 7: 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.

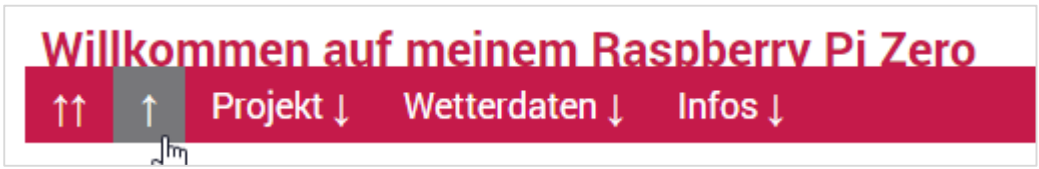
Bei kleinen Auflösungen soll fürs Einblenden der Menüs per dropdown-Schalter eine versteckte Checkbox verwendet werden (2-4).

```
<nav>
    <!-- Menü für kleine Auflösungen -->
    <label for="drop" class="toggle">&#9776; Menü...</label>
    <input type="checkbox" id="drop" />
    <!-- Menü für alle großen Auflösungen -->
```

Programmcode 8: Menü-dropdown

Die Auswirkung bei eingeklapptem Menü ist in Abbildung 60 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.



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.

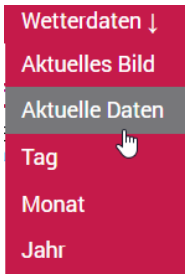
```
6     <ul class="menu">
7
8         <!-- Hauptmenü1 -->
9         <li><a href="https://www.hans-koch.eu/index.php">&uarr;&uarr;</a></li>
10        <li><a href="./index.php">&uarr;</a></li>
```

Programmcode 9: Menübereich 1

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

```
12        <!-- Hauptmenü2 -->
13        <li>
14            <!-- Toggelschalter für das Hauptmenü2 -->
15            <label for="drop-2" class="toggle">Projekt &darr;</label><a href="
16            <input type="checkbox" id="drop-2"/>
17            <ul><!-- Untermenüs des Hauptmenü2  -->
18                <li><a href="">Planung</a></li>
19                <li><a href="">Hardware</a></li>
20                <li><a href="">Raspi einrichten</a></li>
21                <li><a href="">Aufbau</a></li>
22                <li><a href="">Programmierung</a></li>
23            </ul>
24        </li>
```

Programmcode 10: 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=30wetterbild.php**.

```
<li>

    <!-- Toggelschalter für das Hauptmenü3 -->
    <label for="drop-3" class="toggle">Wetterdaten &darr;</label><a href="#">Wetterdaten &darr;</a>
    <input type="checkbox" id="drop-3"/>
    <ul><!-- Untermenüs des Hauptmenü3  -->
        <li><a href="./index.php?aw=30wetterbild.php">Aktuelles Bild</a></li>
        <li><a href="./index.php?aw=30wetterdaten.php">Aktuelle Daten</a></li>
        <li><a href="./index.php?aw=31wetterdatentag.php">Tag</a></li>
        <li><a href="./index.php?aw=32wetterdatenmonat.php">Monat</a></li>
        <li><a href="./index.php?aw=33wetterdatenjahr.php">Jahr</a></li>
    </ul>
</li>
```

Programmcode 11: Menübereich 3

Logodatei

Die Datei 000logo.php enthält als einzige Zeile den Zugriff auf das Bild mit Pfad zum Unterordner.

```
<p class="textklein"><img border="0" src="bilderallgemein/00logo.png"></p>

```

Programmcode 12: 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 einen einfachen Begrüßungstext im php-Format.

003inhaltstart.php

```
<?php
echo "<h1>Willkommen auf dem Raspi2</h1>";
echo "<p class='textnormal'>Hier wird der Aufbau und Konfiguration einer
Wetterstation mit dem RaspberryPI zero und dem Betriebssysytem Raspbian Buster
dokumentiert.<br>";
echo "Die Seite läuft vollständig auf einem RaspberryPI zero W mit
WLAN-Anbindung und einem Apache2-Webserver mit dynamischen PHP-Seiten (PHP7.3).
</p>";
?>
```

Programmcode 13: Inhaltsdatei beim Erstaufruf

Im Browser sieht die Startseite so aus:

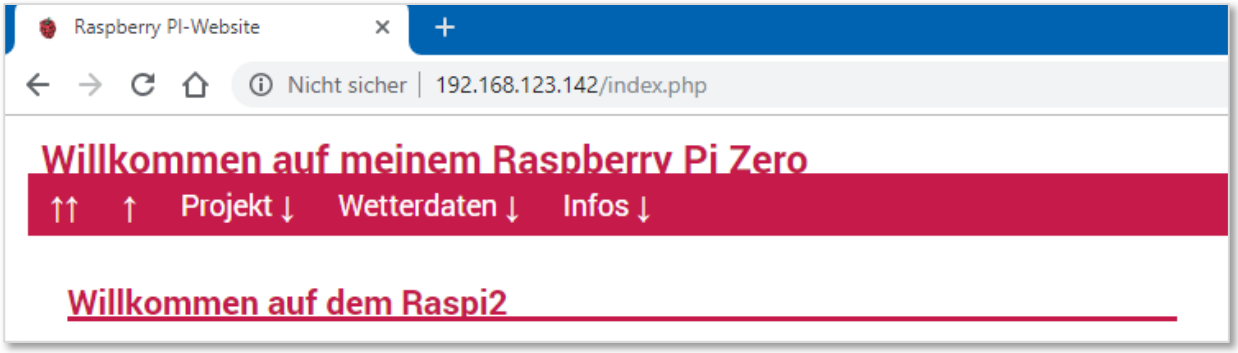


Abbildung 61: Startseite im Browser

Das Favicon wird in der Registerkarte des Browsers angezeigt, auf die Titelzeile folgt die Menüleiste quer mit fünf Menüpunkten, danach der Inhaltsbereich. Das Anzeigen des Wetterbildes, das Auslesen der Sensordaten und die Anzeige in der Sidebar wird im folgenden Abschnitt erläutert.

7 Wetterdaten veröffentlichen

Auf der bereits angelegten Webseitenstruktur sollen nun Informationen der Wetterstation veröffentlicht werden. Im ersten Ausbauschritt wird die Kamera und der Temperatur-/Feuchtesensor eingebunden. Eine Erweiterung mit einem Luftdrucksensor ist für einen späteren Zeitpunkt geplant.

7.1 Wetterbilder

Die Wetter-Standbilder sollen alle 10 Minuten aktualisiert werden und als Großbild, sowie zusätzlich alle 60 Minuten als kleinformatiges Bild gespeichert werden. Am Ende des Tages sollen alle Bilder automatisch in einem **tar**-Archiv abgelegt werden.

Webseite für Kameraanzeige erstellen

Zunächst wird eine Webseite für das Hauptbild erzeugt. Da die Webseite später dynamisch in eine Hauptseite eingebunden wird, soll noch keine optische Gestaltung vorgenommen werden. Das aktuelle Kamerabild wird mit dem Dateinamen **00wetteraktuell.jpg** im Unter-Verzeichnispfad **wetterstation2019/bilder/** abgelegt.

```
1  <h1>Webcam aktuell</h1>
2  <p class='textklein'><br>Aktuelles Wetterbild<br>
3  71384 Weinstadt Richtung Nord-Ost<br>
4  <img src='wetterstation2019/bilder/00wetteraktuell.jpg' > </p>
5
```

Programmcode 14: Wetterbildanzeigedatei

Die Webseitendatei erhält den Dateinamen **30wetterbild.php**.

Kameratest

Bevor das Script zur Erzeugung des aktuellen Wetterbildes erstellt wird, erfolgt nochmals ein Test der Kamera von der Kommandozeile des RaspberryPI aus.

```
pi@h-raspi2:~ $ pwd
/home/pi
pi@h-raspi2:~ $ sudo raspistill -o /home/pi/wetterbild.jpg
```

Der Remotezugriff mit WinSCP zeigt die **jpg**-Datei, sie kann mit einem Bildbearbeitungstool geöffnet werden (siehe Abbildung 62).

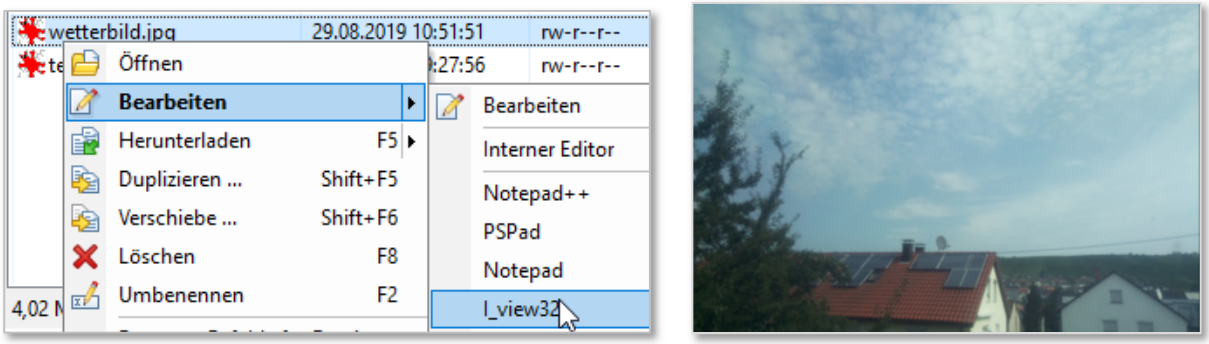


Abbildung 62: Aktuelles Wetterbild

Kamerasoftware einrichten

Zur nachträglichen Bearbeitung der Bilder muss das Paket **imagemagick** nachinstalliert werden.

```
$ sudo apt-get install imagemagick
```

Nach der Aufnahme eines Bildes können nun mit dem Kommando **convert** Bildbeschneidung und Beschriftung erledigt werden.

Bilder mit bash-Script aufnehmen

Zur automatischen Aufnahme der Wetterbilder wird ein bash-Shellscript programmiert.

Im ersten Programmteil (4-8) wird der Aufnahmezeitpunkt in einzelne Werte zerlegt, um später die Beschriftung des Bildes daraus zu generieren. Der Verzeichnispfad für die Bilder und zwei Dateinamen werden ebenfalls in Variablen angelegt (10-11).

Nun wird mit **raspistill** die Aufnahme ausgelöst und unter dem Verzeichnispfad abgespeichert.

Im nächsten Schritt soll das Bild mit dem Aufnahmezeitpunkt beschriftet werden. Hierzu wird das Tool **convert** eingesetzt. Das Bild wird zunächst unter dem temporären Dateinamen abgelegt (15) und dann kopiert und wieder unter dem Originalnamen gespeichert (17).

minutenbilder2019

```
1  #!/bin/bash
2  #Script zum Erzeugen der aktueller Bilder (minutenbilder2019)
3  #Dateiname der Bilddatei:  00wetteraktuell.jpg
4  stunde=`date +%H`
5  minute=`date +%M`
6  tag=`date +%d`
7  monat=`date +%m`
8  jahr=`date  +%Y`
9  #Speicherpfade festlegen
10 bildpfad="/var/www/html/wetterstation2019/bilder/00wetteraktuell.jpg"
11 bildpfadtmp="/var/www/html/wetterstation2019/bilder/00wetteraktuelltmp.jpg"
12 #Bild aufnehmen und speichern
13 sudo raspistill  -o $bildpfad
14 #Bild beschneiden und beschriften, als tmp-Bild ablegen
15 convert  -pointsize 50 -gravity NorthWest -fill red -draw "text 200,50 'Aufnahmezeitpunkt:
   $tag.$monat.$jahr - $stunde:$minute'" -crop 3280x1749+0+0 +repage  $bildpfad $bildpfadtmp
16 #tmp-Bild in normalen Dateienamen umbenennen
17 cp $bildpfadtmp $bildpfad
18
```

Programmcode 15: bash-Script zum Erzeugen des aktuellen Wetterbildes

Das Script wird im Ordner **/var/www/cgi-bin** außerhalb des Apache-Zugriffs gespeichert. Somit kann es nur das Linux System selbst ausführen.

Auf der Kommandozeile am RaspberryPI wird das Script getestet.

```
$ sudo bash minutenbilder2019
```

Das Foto zeigt das Wetterbild mit Beschneidung und Beschriftung.

Abbildung 63: Kamerabild bearbeitet

Das Erzeugen der Stundenbilder erfolgt ebenfalls mit einem bash-Script. Unterschied zum obigen Beispiel ist, dass die Bilder kleiner sind und nun auch der Dateiname abhängig von der aktuellen Zeit dynamisch gesetzt wird (10). Dadurch können 24 Bilder pro Tag angefertigt werden.

minutenbilder2019 stundenbilder2019

```
1  #!/bin/bash
2  #Script zum Erzeugen stündlicher Bilder (stundenbilder2019)
3  #Dateiname der Bilddatei: HHstundenbild.jpg (HH=Stunde)
4  stunde=`date +%H`
5  minute=`date +%M`
6  tag=`date +%d`
7  monat=`date +%m`
8  jahr=`date  +%Y`
9  #Speicherpfade festlegen
10 bildpfad="/var/www/html/wetterstation2019/bilder/"$stunde"stundenbild.jpg"
11 bildpfadtmp="/var/www/html/wetterstation2019/bilder/stundenbildtmp.jpg"
12 #Bild aufnehmen Kleinformat 320x240
13 sudo raspistill  -o $bildpfad -w 320 -h 240
14 #Bild beschndiden und beschriften
15 convert  -pointsize 10 -gravity NorthWest -fill red -draw "text 10,10
   '$tag.$monat.$jahr - $stunde:$minute'" -crop 320x200+0+0 $bildpfad $bildpfadtmp
16
17 cp $bildpfadtmp $bildpfad
```

Programmcode 16: bash-Script für die Stundenbilder

Zur Langzeitarchivierung von Wetterbildern wird ein weiteres Script verwendet, das täglich jeweils um 12:00 Uhr ein Tagesbild aufnimmt. Diese Datei soll 1 Jahr gespeichert bleiben.

Das automatische Ausführen der bash-Scripts wird jeweils über einen **cronjob** des Linuxsystems erreicht. Hierbei werden die Minutenbilder zunächst von 0 Uhr bis 23 Uhr in den Minuten 10, 20, 30, 40 und 50 aufgenommen. Das Stundenbild wird zwischen 4 Uhr und 23 Uhr jeweils zur Minute 00 erstellt. Für die Langzeitarchivierung über ein ganzes Jahr dienen die Tagesbilder, die jeweils um 12:05 Uhr fotografiert werden. Damit auch die Sensorabfrage regelmäßige Messwerte liefert, erfolgt der Aufruf des Pythonscriptes im Abstand von 7 Minuten.

Aufgrund der von Windows unterschiedlichen UNIX-Sonderzeichen ist es ratsam, die **crontab** unter einem Systemeditor zu bearbeiten (**nano** oder **vi**)

```
$ sudo nano /etc/crontab
```

```
  GNU nano 3.2                                    /etc/crontab

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# ------- /etc/crontab (Systemweite crontab) -----------------
# min   dow (0-6, Sonntag =0)
#  H T M   user  Kommando
10 * * * * root bash /var/www/cgi-bin/wetterstation2019/minutenbilder2019
20 * * * * root bash /var/www/cgi-bin/wetterstation2019/minutenbilder2019
30 * * * * root bash /var/www/cgi-bin/wetterstation2019/minutenbilder2019
40 * * * * root bash /var/www/cgi-bin/wetterstation2019/minutenbilder2019
50 * * * * root bash /var/www/cgi-bin/wetterstation2019/minutenbilder2019
00 4-23/1 * * * root bash /var/www/cgi-bin/wetterstation2019/stundenbilder2019
05 12 * * * root bash /var/www/cgi-bin/wetterstation2019/tagesbilder2019
1-59/7 * * * * root python /var/www/cgi-bin/wetterstation2019/sensorabfrage2019.py
#
```

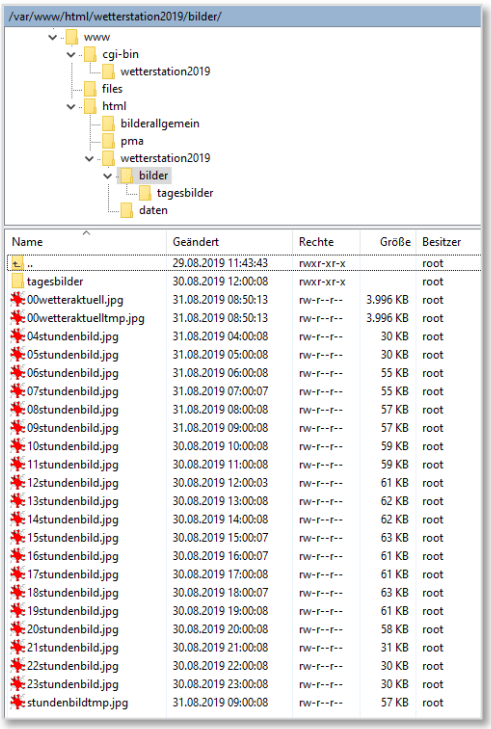
Programmcode 17: crontab-Konfiguration

Im RaspberryPI-Verzeichnis **/var/www/html/wetterstation2019/bilder** sind alle Wetterbilder gespeichert (siehe Abbildung rechts).

00wetteraktuell.jpg ist das aktuelle „Minutenbild“ und die einzelnen Stundenbilder werden durch das bash-Script von **04**... bis **23**... benannt.

Das Tagesbild erhält als Präfix Monat und Tag des Aufnahmedatums. Diese Bilder kommen in das Unterverzeichnis **tagesbilder**.

Die Ausgabe der Aufnahmen und der Messdaten erfolgt auf den entsprechenden Inhalts-Webseiten.



Programmcode 18: Bildverzeichnis

7.2 Temperatur und Luftfeuchtigkeit anzeigen

Nun sollen zusätzlich zum Wetterbild die aktuelle Temperatur und die Luftfeuchtigkeit auf der Webseite angezeigt werden.

Hierzu wird nochmals mit Hilfe des von Adafruit mitgelieferten Testprogramms die Funktionstüchtigkeit geprüft.

```
$ cd ~/downloads/Adafruit_Python_DHT/examples
$ sudo ./AdafruitDHT.py 22 4
```

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

```
pi@h-raspi2:~ $ cd ~/downloads/Adafruit_Python_DHT/examples/
pi@h-raspi2:~/downloads/Adafruit_Python_DHT/examples $ sudo ./AdafruitDHT.py 22 4
Temp=27.9*  Humidity=91.1%
pi@h-raspi2:~/downloads/Adafruit_Python_DHT/examples $ date
Fre Aug 23 09:23:26 CEST 2019
pi@h-raspi2:~/downloads/Adafruit_Python_DHT/examples $
```

Laut Datenblatt des Sensors dürfen alle drei Sekunden Abfragen durchgeführt werden. Kürzere Intervalle sind nicht möglich.

Python-Skript zur Messung und Darstellung

Das mitgelieferte Python-Beispielskript wird nun so angepasst, dass die Wetterdaten sofort im **html**-Format in das Webverzeichnis abgelegt werden. Damit können die Informationen beim Aufruf der Webseite durch den Server direkt von dort ausgelesen und angezeigt werden. Das Skript wird aus Sicherheitsgründen unter **/var/www/cgi-bin/wetterstation2019/sensorabfrage.py** außerhalb des Webserverstammpfads abgelegt, ist aber vom System ausführbar.

In Zeile (2-3) werden Module importiert, die für die Ausführung benötigt werden. Der verwendete Sensor (5) und der genutzte Pin (4) werden je in eine Variable gespeichert. Danach erfolgt das Auslesen der Sensordaten durch Übergaben der Variablenwerte. Zurückgegeben werden Feuchte und Temperatur (9).

Damit später mit sinnvollen Datums- und Zeitwerten weitergearbeitet werden kann, müssen diese von der Systemzeit in gewünschte Formate umgewandelt und in Variablen gespeichert werden (11-12).

```
1  #!/usr/bin/python
2  import Adafruit_DHT
3  import time
4  # Sensorwerte: Adafruit_DHT.DHT11, Adafruit_DHT.DHT22 oder Adafruit_DHT.AM2302
5  strSensor = Adafruit_DHT.DHT22
6  #Pin am RaspiZero
7  intPin = 4
8  #Sensor auslesen
9  lngFeuchte, lngTemp = Adafruit_DHT.read_retry(strSensor, intPin)
10 #Aktueller Messzeitpunkt
11 datAktuell=time.strftime('%d.%m.%Y - %H:%M')
12 datAktuellZeit=time.strftime('%H')
```

Programmcode 19: bash-Script zum Auslesen der Sensordaten (Teil 1)

Im nächsten Schritt müssen die zurückgegebenen Werte geprüft werden. Bei Erfolg werden die Messwerte und der Messzeitpunkt angezeigt. Diese Anzeige wird sichtbar, wenn das Skript an der Kommandozeile gestartet wird.

```
13 #Werte Testen
14 if lngFeuchte is not None and lngTemp is not None:
15     print('Temperatur={0:0.1f}*C  Feuchtigkeit={1:0.1f}%'.format(lngTemp, lngFeuchte))
16 else:
17     print('Keine Messdaten erhalten - bitte nochmals versuchen...')
18 print(datAktuell)
```

Programmcode 20: bash-Script zum Auslesen der Sensordaten (Teil 2)

Es erfolgt nun ein Zwischentest des Skripts. Die Anzeige erfolgt und das Skript wertet die Daten korrekt aus.

```
$ sudo python sensorabfrage2019.py
```

```
pi@h-raspi2:~ $ sudo python /var/www/cgi-bin/wetterstation2019/sensorabfrage2019.py
Temperatur=27.4*C  Feuchtigkeit=55.5%
29.08.2019 - 10:43
```

Nun sollen die Daten auf der **html**-Seite angezeigt werden. Hierzu erfolgt eine Ergänzung im Python Skript. Die Datei wird zum Schreiben geöffnet (20). Sie muss in einem, für den Webserver durch Lesezugriff erreichbaren Verzeichnis liegen (**/var/www/html/[…]/00sensoraktuell.html**).

Der html-Code beinhaltet neben den Standard-Tags die dynamisch hinzugefügten Variablen aus der Sensorabfrage (21) und zusätzlich ergänzende Texthinweise mit CSS-Formatierungen. Anschließend wird der html-Text in die Datei geschrieben und diese geschlossen (22+23).

```
19 #Aktuelle Wetterdaten in eine html-Datei schreiben
20 file = open("/var/www/html/wetterstation2019/daten/00sensoraktuell.html","w")
21 strHtml="<p class='textklein'>Aktuelle Sensordaten:<br>" + datAktuell +
   "</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></table>".format(lngTemp,lngFeuchte)
22 file.write(strHtml)
23 file.close()
```

Programmcode 21: bash-Script zum Auslesen der Sensordaten (Teil 3)

Zusätzlich werden die Daten in stundenweise getrennte Dateien abgelegt. Dies wird durch die Integration der aktuellen Stunde in den Dateinamen erreicht (26). Dadurch sollen die Daten eines kompletten Tages gespeichert bleiben, ohne dass ein Datenbankzugriff nötig ist.

```
25 #Aktuelle Wetterdaten in eine zweite html-Datei schreiben
26 filWetterdaten="/var/www/html/wetterstation2019/daten/"+ datAktuellZeit +".html"
27 print filWetterdaten
28 file = open(filWetterdaten,"w")
29 strHtml="<p class='textklein'>" + datAktuell + " h {0:.1f}*C   {1:.1f} % Luftf.".
   format(lngTemp,lngFeuchte)
30 file.write(strHtml)
31 file.close()
```

Programmcode 22: bash-Script zum Auslesen der Sensordaten (Teil 4)

Die Darstellung der Wetterdaten erfolgt nun durch Einbindung der erzeigten html-Dateien in die komplette Website. Als Position der aktuellen Anzeige wurde die **sidebar** ausgewählt (siehe auch 0). Zusätzlich wird dort der Besucherzähler integriert.

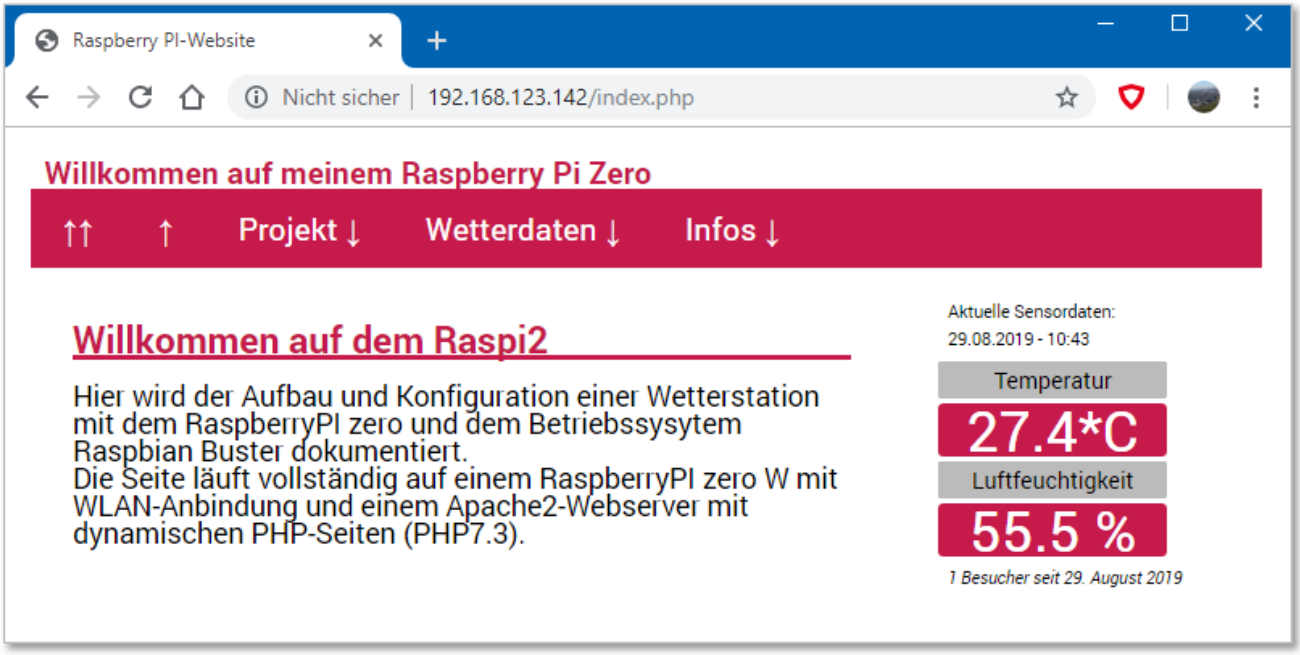


Abbildung 64: Wetterdaten auf der Startseite darstellen

Der Code der **sidebar** wird durch die Startseite **index.php** zusammengesetzt und angezeigt. Die Gestaltung der Sensoranzeige übernimmt eine CSS-Definition der zentralen Styledatei.

```php
<?php
//aktuelle Wetterdaten anzeigen
include ("./wetterstation2019/daten/00sensoraktuell.html");
//Besucherzähler auslesen
$intZeiger = fopen("../files/raspi2zaehler.txt" , "r+");
$intZaehler = fgets($intZeiger,20);
    if (isset($_COOKIE['raspi2']) && $_COOKIE['raspi2'] == "besucht")
        {
        }
    else{
        $intZaehler=$intZaehler+1;
        rewind($intZeiger);
        fputs($intZeiger,$intZaehler);
        echo "";
        }
fclose($intZeiger);
//Besucherzähler darstellen
echo "<p class='textkleinkursiv'>" . $intZaehler;
echo " Besucher seit 8. Juni 2019</p>";
?>
```

Programmcode 23: Wetterdaten auf die Startseite einfügen

Das Anzeigen der Tages-Wetterdaten erfolgt auf einer eigenen Seite, die in den Inhaltsbereich der Indexseite includiert wird. Abhängig von der Tageszeit werden die Messwerte aus den einzelnen Dateien ausgelesen und angezeigt. Ebenso wird mit den Tagesbildern (Stundenbildern) verfahren.

```php
<?php
$strDatum=date('d.m.Y');
$strStunde=date('H');
echo "<h2>Wetterdaten</h2>";
echo "<p class='textklein'>";

$intBildnummer=4;
echo "Messwerte:<br>";
while(($intBildnummer < $strStunde))
{
    $strDateiname2= "wetterstation2019/daten/" .
    sprintf('%02d', $intBildnummer) . ".html";
    include($strDateiname2);
    //
    $intBildnummer++;
}
echo "</p><hr>";
echo "<p class='textklein'>Bilder:<br>";
$intBildnummer=4;
while(($intBildnummer < $strStunde))
{
    $strDateiname= "wetterstation2019/bilder/" .
    sprintf('%02d', $intBildnummer) .
    "stundenbild.jpg";
    echo "<img src='$strDateiname' >";
    $intBildnummer++;
}

echo "</p>";

?>
```

Programmcode 24: Tages-Wetterdaten anzeigen

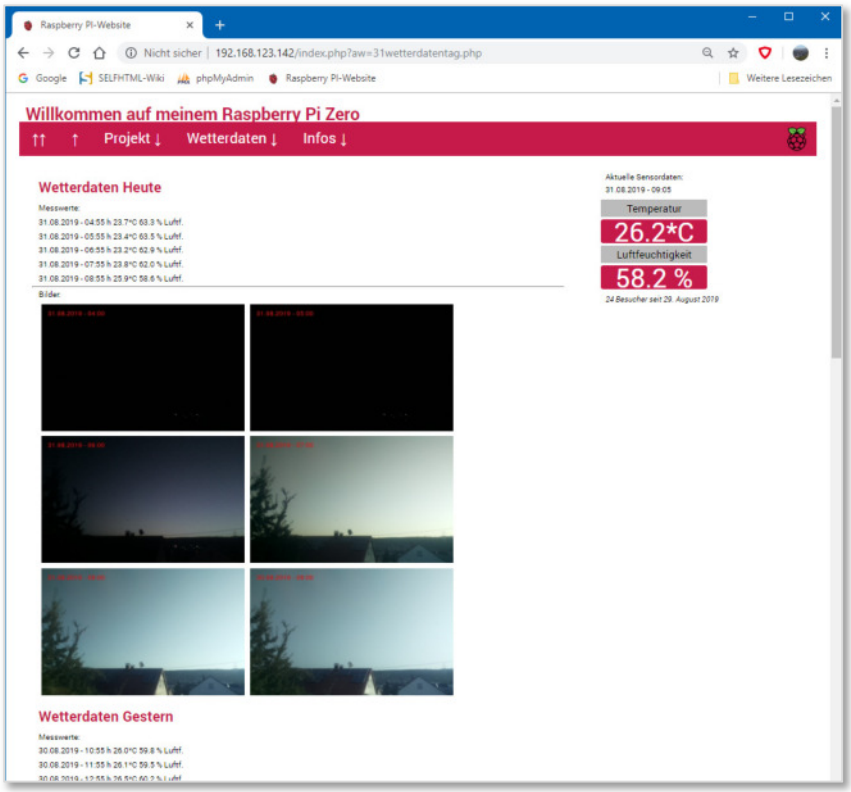


Abbildung 65: Tages-Wetterdaten auf der Webseite

8 Erweiterung der Wetterstation mit einem Luftdrucksensor

Nachdem die Grundfunktionen der Wetterstation mit Temperatur, Luftfeuchtigkeit und Wetterbild problemlos funktioniert, soll nun zusätzlich der Luftdruck gemessen, angezeigt und gespeichert werden. Hierzu wird der Sensor BMP280 der Fa. BOSCH eingesetzt. Zusätzlich kann dieser Sensor auch die Temperatur messen, so dass dann hierzu zwei Messwerte vorliegen.

8.1 Allgemeines

Ein Artikel aus (Plate, 2018)[9] 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 Pa = 1013,25 hPa ≈ 1 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).

[9] http://www.netzmafia.de/skripten/hardware/RasPi/Projekt-BMP280/index.html

8.2 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.

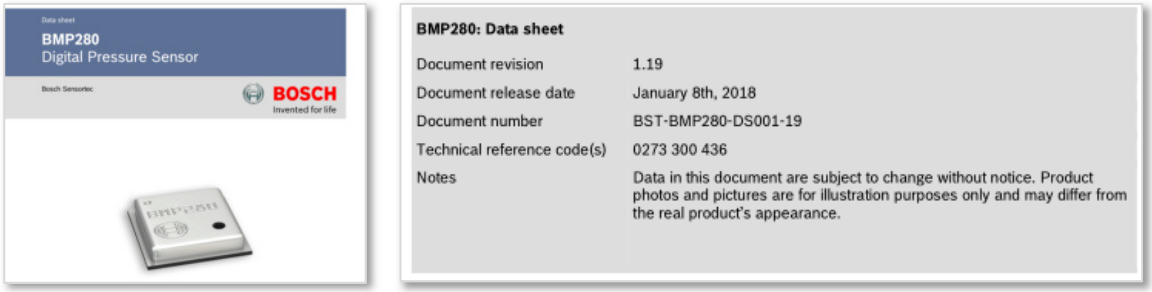


Abbildung 66: Datenblatt-Titel BMP280

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.

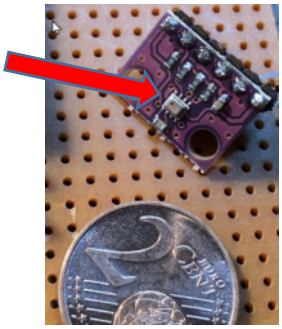


Abbildung 67: BMP280 mit Trägerplatine

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	I²C (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 68 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.

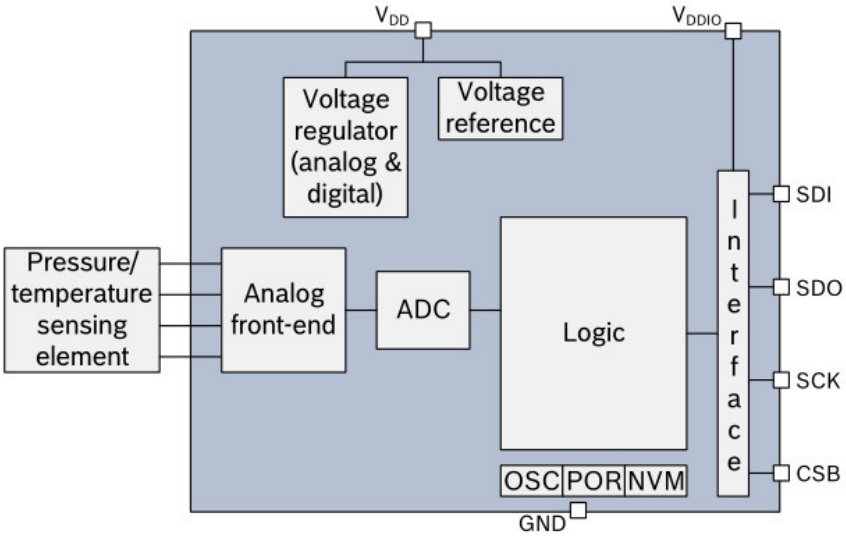


Abbildung 68: Blockschaltbild BMP280 (BOSCH Sensortec GmbH, 2018, S. 11)

Messzyklus

Der Zyklus der Messvorgänge ist in Abbildung 69 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.

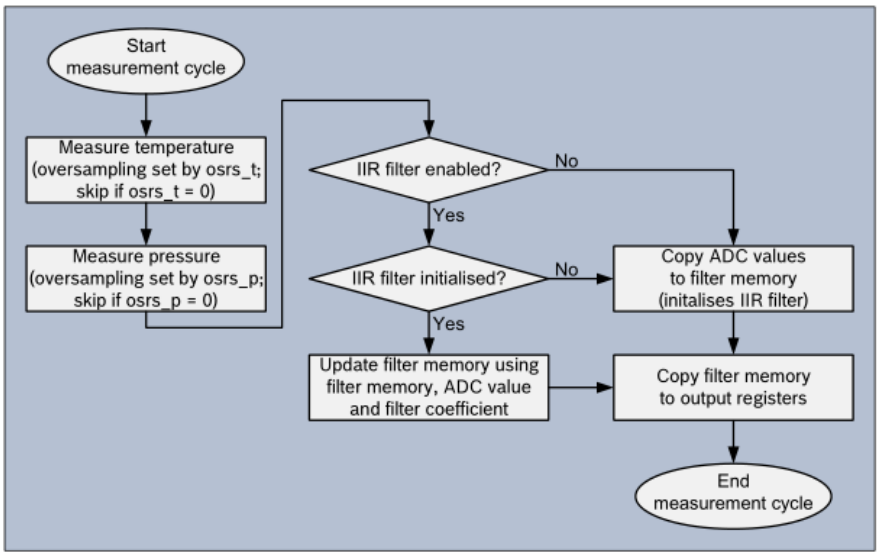


Abbildung 69: Messzyklus BMP280

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.

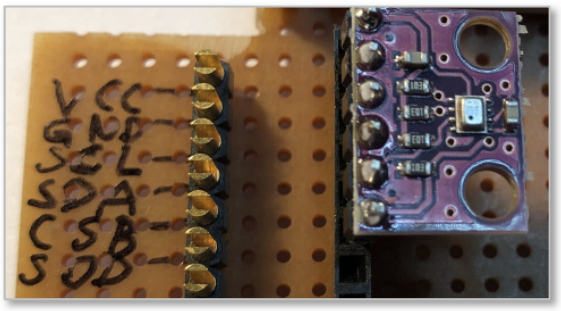


Abbildung 70: Anschaltung der Sensorplatine

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

Das Schreiben in ein Register über den I²C-Bus ist in beispielhaft in Abbildung 72 dargestellt, das Lesen aus einem Datenregister zeigt Abbildung 73. 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 72: I²C Schreibvorgang (BOSCH Sensortec GmbH, 2018, S. 29)

Der Lesevorgang beginnt genauso wir 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.

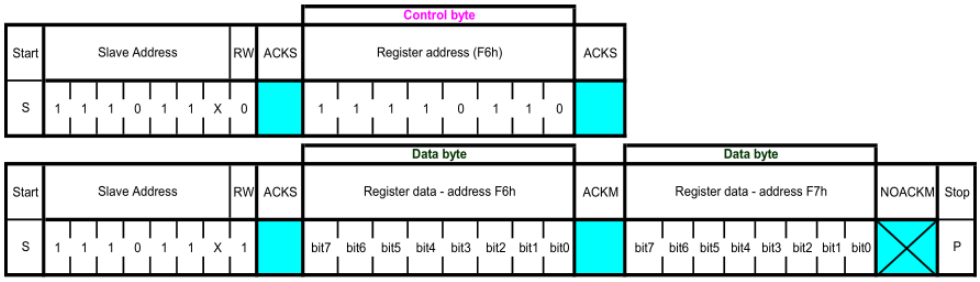


Abbildung 73: I²C Lesevorgang (BOSCH Sensortec GmbH, 2018, S. 30)

Weitere Angaben zur Steuerung der Datenübertragung über die I²C-Schnittstelle finden sich im SBMP280-Datenblatt (z.B. wie in Abbildung 74).

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

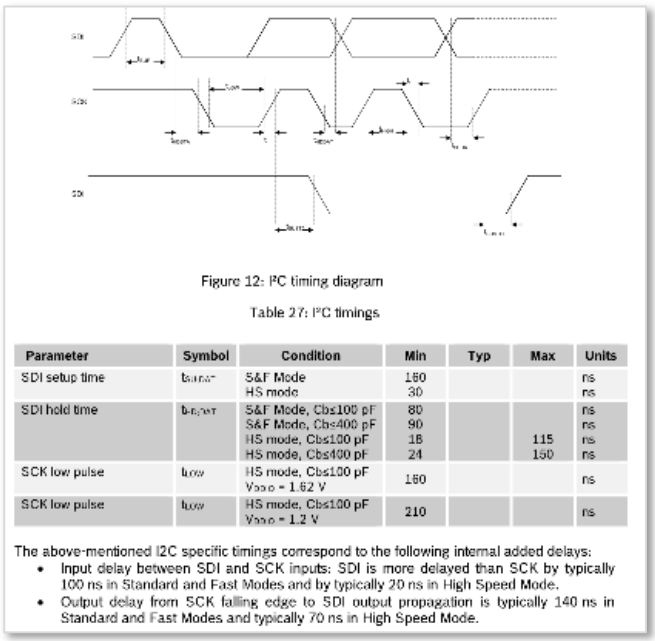


Figure 12: I²C timing diagram

Table 27: I²C timings

Parameter	Symbol	Condition	Min	Typ	Max	Units
SDI setup time		S&F Mode HS mode	160 30			ns ns
SDI hold time		S&F Mode, Cb≤100 pF S&F Mode, Cb≤400 pF HS mode, Cb≤100 pF HS mode, Cb≤400 pF	80 90 18 24		 115 150	ns ns ns ns
SCK low pulse		HS mode, Cb≤100 pF Vddio = 1.62 V	160			ns
SCK low pulse		HS mode, Cb≤100 pF Vddio = 1.2 V	210			ns

The above-mentioned I2C specific timings correspond to the following internal added delays:

- Input delay between SDI and SCK inputs: SDI is more delayed than SCK by typically 100 ns in Standard and Fast Modes and by typically 20 ns in High Speed Mode.
- Output delay from SCK falling edge to SDI output propagation is typically 140 ns in Standard and Fast Modes and typically 70 ns in High Speed Mode.

Abbildung 74: I²C-Timing BMP280 (BOSCH Sensortec GmbH, 2018, S. 33)

8.3 mechanischer Umbau

Die Wetterstation muss nun auch mechanisch etwas umgebaut werden. Der bisher nicht gegen Spritzwasser (Regen) geschützte Feuchtigkeitssensor soll nun 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.

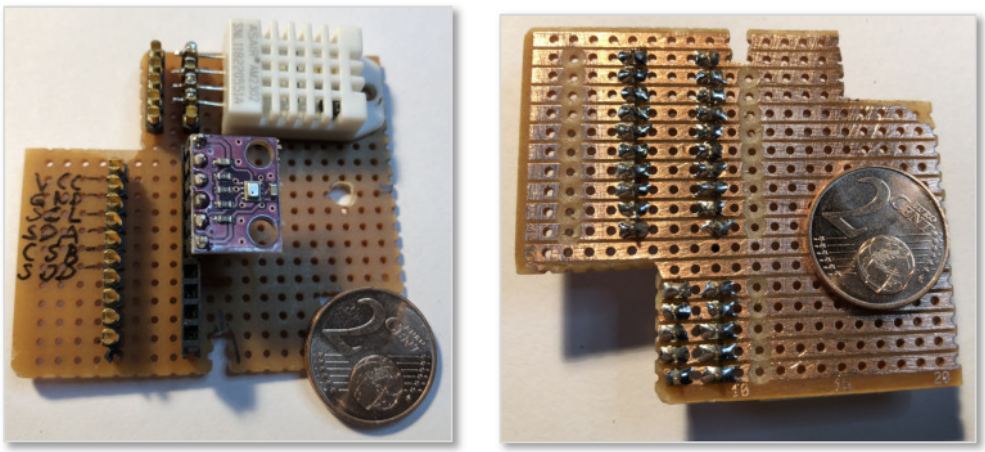


Abbildung 75: Leiterplatte für die Sensoranschaltung

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

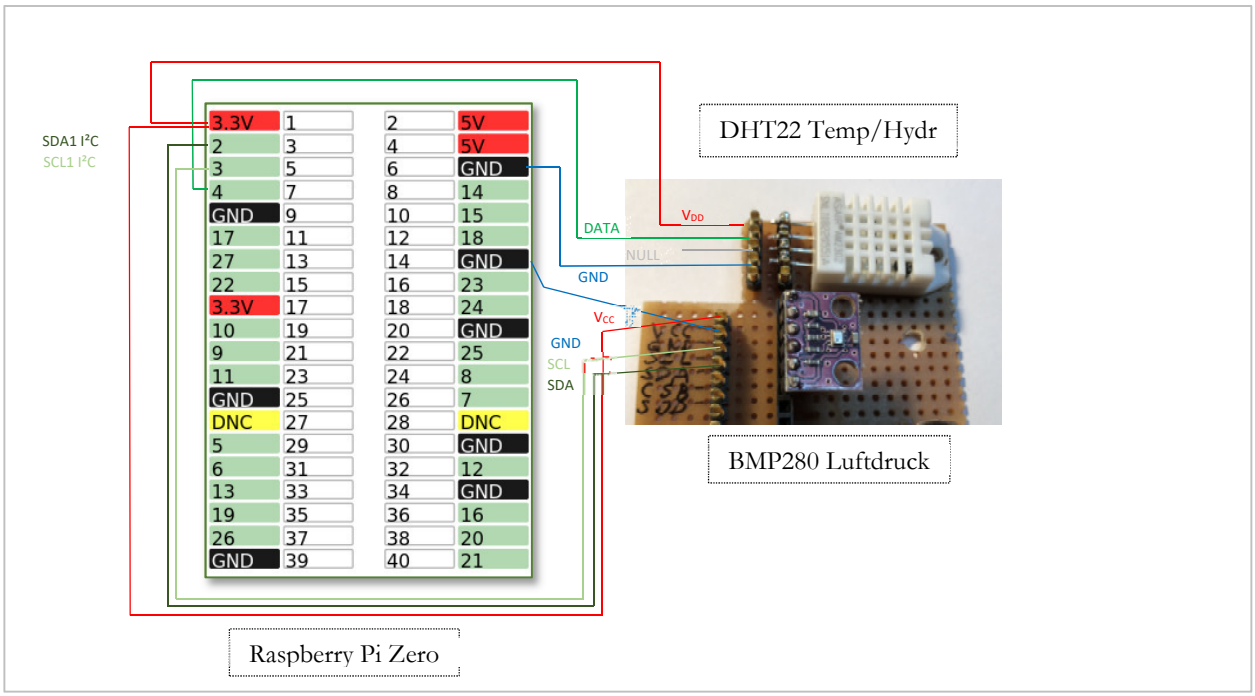


Abbildung 76: Verdrahtung der Sensoren

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.

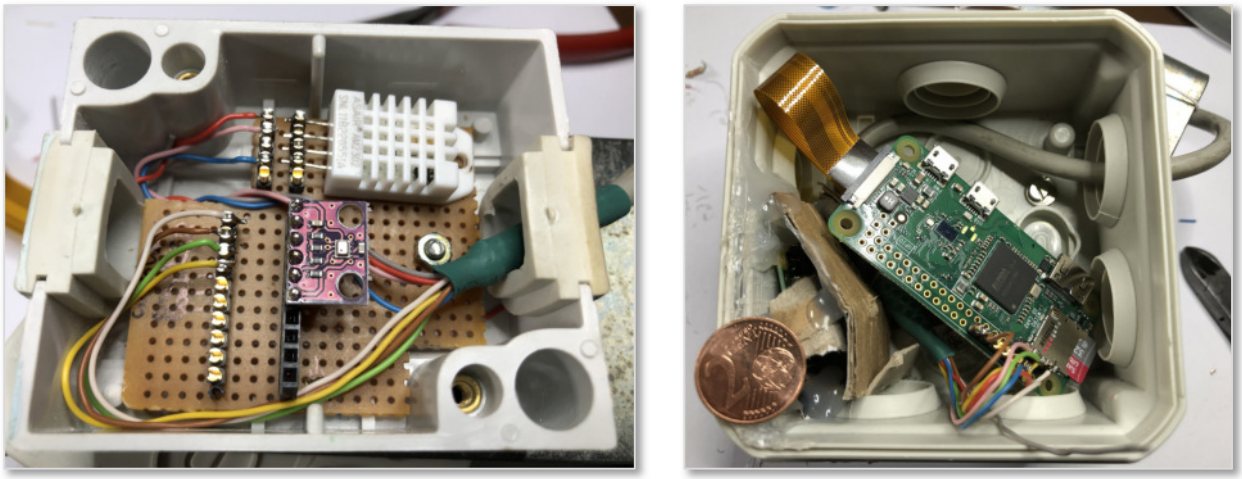


Abbildung 77: Sensor- und Raspberry-Gehäuse

8.4 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 --
pi@h-raspi2:~ $
```

Abbildung 78: 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 mitteilt 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 79: 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.

8.5 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. Der vorliegende Python Quellcode (Plate, 2018) wurde ein wenig bearbeitet und zum besseren Verständnis mit deutschen Kommentaren versehen.

Python-Script - Kopfdaten

Das Pythonscript soll die Kommunikation mit dem Sensor, das Auslesen der Daten und die Berechnung durchführen. Später soll eine zur Datenspeicherung in der Datenbank MariaDB erfolgen.

```
#!/usr/bin/env python
#Sensordaten vom BOSCH BMP280 auslesen
#erstellt 21.09.2019 (HK)
#---------------------------------------
import smbus
import time
# Adresse des I2C Bus
BUS = 1
# BMP280 Adresse, 0x76 oder 0x77
BMP280ADDR = 0x76

# Meereshoehe der Wetterstation 239 m
ALTITUDE = 239

```

Programmcode 25: Kopfdaten des Python-Scripts

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

```
# I2C Bus einlesen
bus = smbus.SMBus(BUS)

# Temperatur kalibrieren (Array)
T = [0, 0, 0];
# Druck kalibrieren (Array)
P = [0, 0, 0, 0, 0, 0, 0, 0, 0];

# Kalibrierungsdaten aus dem Sensor einlesen
#(Daten aus Adresse 0x88, 24 bytes)
data = bus.read_i2c_block_data(BMP280ADDR, 0x88, 24)

# Temperatur-Koeffizienten
T[0] = data[1] * 256 + data[0]
T[1] = data[3] * 256 + data[2]
if T[1] > 32767:
  T[1] -= 65536
T[2] = data[5] * 256 + data[4]
if T[2] > 32767:
  T[2] -= 65536

# Druck-Koeffizienten
P[0] = data[7] * 256 + data[6];
for i in range (0, 8):
  P[i+1] = data[2*i+9]*256 + data[2*i+8];
  if P[i+1] > 32767:
    P[i+1] -= 65536
```

Programmcode 26: Kalibrierung BMP280

Konfigurationsdaten in Register 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
```

Programmcode 27: BMP280 - Konfigurationsdaten schreiben

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

Tabelle 5: BMP280 Controlregister

Controlregister (**ctrl_meas**)								
Adresse **0xF4**	Bit 7, Bit6, Bit5 (**osrs_t**)			Bit 4, Bit3, Bit2 (**osrs_p**)			Bi1, Bit0 (**mode**)	
	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 80: 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]	$t_{standby}$ [ms]
000	0.5
001	62.5
010	125
011	250
100	500
101	1000

Messwertedaten 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 81: 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

Insgesamt kann dabei eine Auflösung von 20 Bit erreicht werden (2^0 bis 2^3 sind fest auf 0).

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
```

Programmcode 28: 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
```

Programmcode 29: 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**) muss in Pascal 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.

```
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
```

Programmcode 30: BMP280 - Messwerte ausgeben

Ein Test auf der Kommandozeile zeigt die aktuellen Sensordaten an:

```
pi@h-raspi2:/var/www/cgi-bin/wetterstation2019 $ sudo python sensorluftdruck2019.py
Temperatur        : 23.80 C
abs. Luftdruck    :    984.82 hPa
rel. Luftdruck NN : 1013.20 hPa
```

9 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

9.1 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 82: Entmontage der Wetterstation[10]

[10] 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.

```
pi@h-raspi2:~/testfotos $ raspistill -o /home/pi/testfotos/testbild.jpg
pi@h-raspi2:~/testfotos $ ls -l
total 4280
-rw-r--r-- 1 pi pi 4380400 Oct 12 15:57 testbild.jpg
pi@h-raspi2:~/testfotos $
```

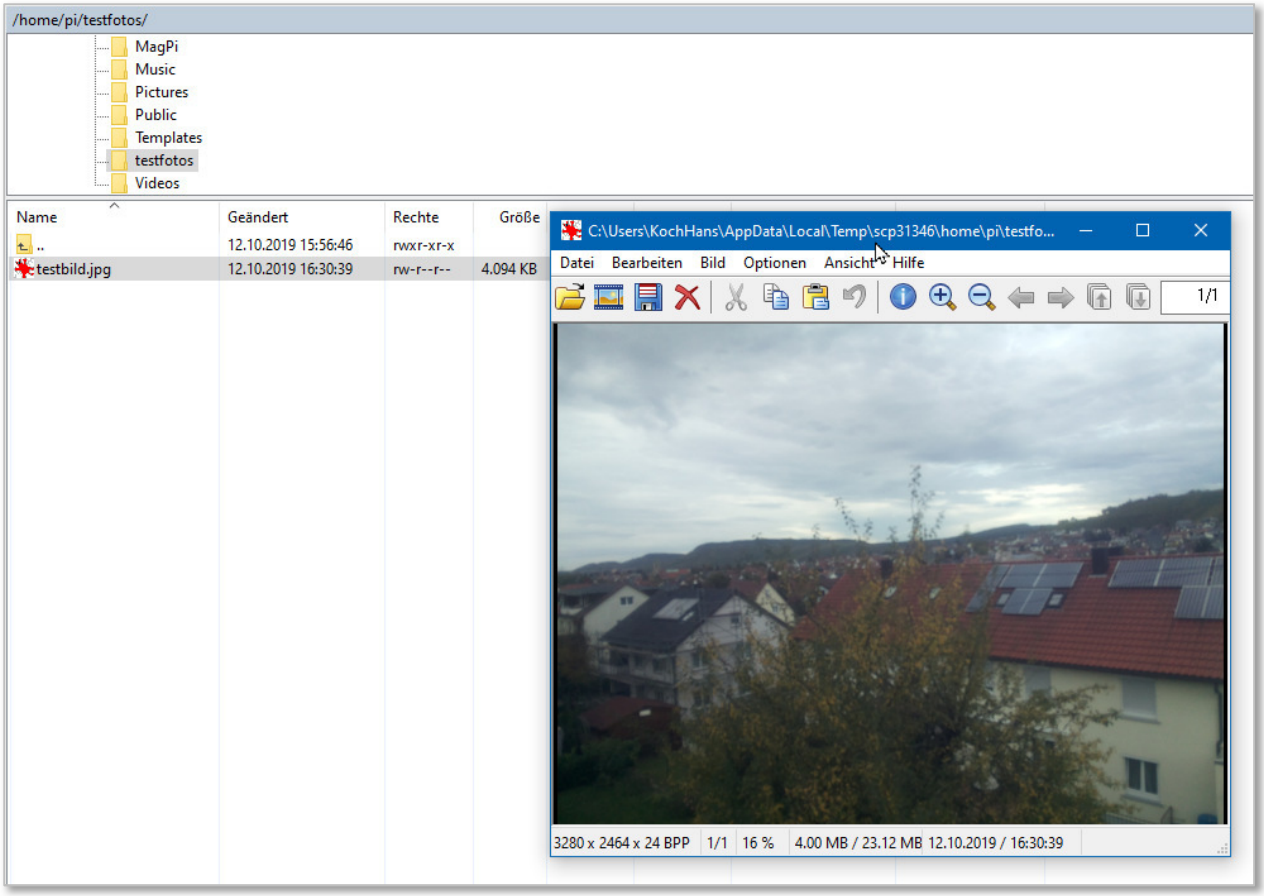


Abbildung 83: Testbilder der Webcam

9.2 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.

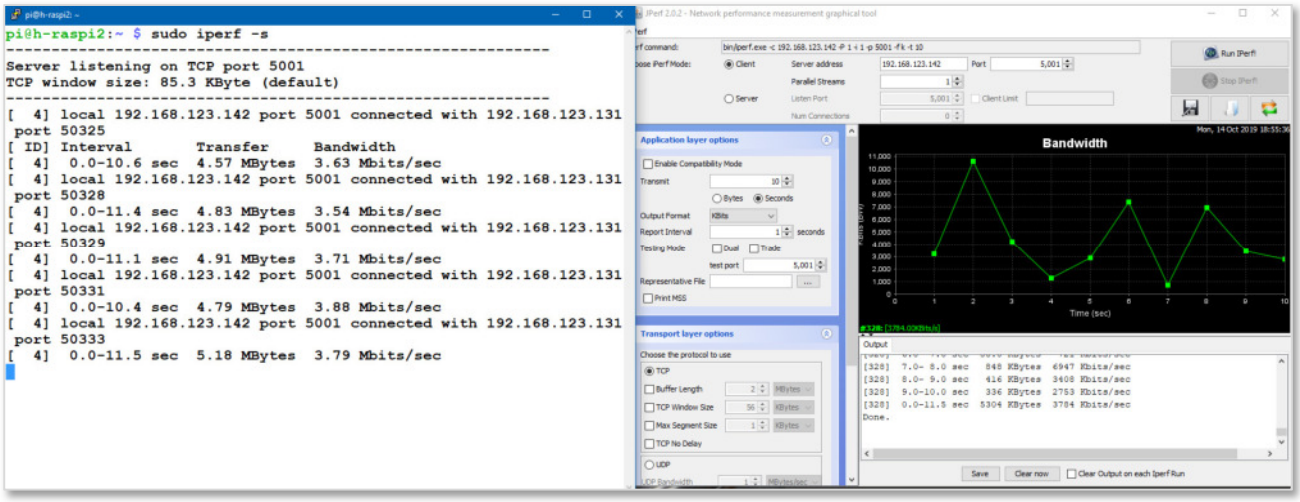


Abbildung 84: Bandbreitentest mit jperf

10 Anpassung der Messzeiten

Die Auswertung der ersten Messzyklen zeigte, dass die Messungen nicht so kurz aufeinanderfolgend gewählt werden müssen, um eine aussagekräftige und angemessen genaue Statistik zu erhalten. Durch Vergrößerung der Messabstände wird die anfallende Datenmenge merklich reduziert. Hierzu wurde die **crontab** und auch die Python-Scripte nochmals angepasst.

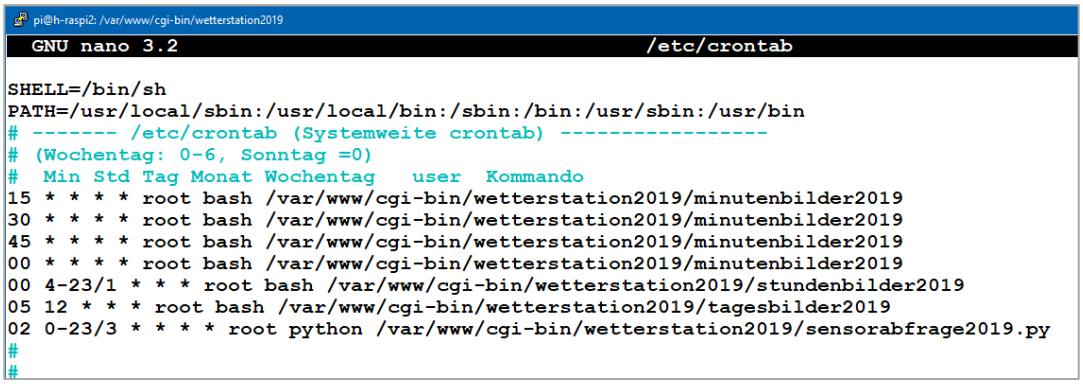


Abbildung 85: crontab anpassen

Die Ergänzung der **crontab** bewirkte zunächst, dass keinerlei Messung mehr durchgeführt wurde. Die Fehlersuche durch die Anzeige der Logdatei zeigte, dass ein Syntaxfehler die komplette **cron**-Funktion stillsetzte. Alle durchgeführten Änderungen wurden verworfen.

$ cat /var/log/syslog

```
Oct 25 19:31:01 h-raspi2 cron[270]: (*system*) ERROR (Syntax error, this crontab file will be ignored)
Oct 25 19:32:59 h-raspi2 systemd[1]: Started Session c5 of user root.
Oct 25 19:33:01 h-raspi2 cron[270]: (*system*) RELOAD (/etc/crontab)
Oct 25 19:33:01 h-raspi2 cron[270]: Error: bad username; while reading /etc/crontab
Oct 25 19:33:01 h-raspi2 cron[270]: (*system*) ERROR (Syntax error, this crontab file will be ignored)
pi@h-raspi2:~ $ cat /var/log/syslog
```

Abbildung 86: System-Logdatei

Der Fehler konnte so lokalisiert werden. Es handelte sich um eine überzählige Spalte (*) in der python-Zeile der **crontab**.

```
02 0-23/3 * * * * root python /var/www/
#
```

Abbildung 87: Fehler in der crontab

Der Fehler wurde berichtigt und die **crontab** funktionierte wieder. Weitere Anpassungen und Vereinfachungen führten zu folgender Zeitsteuertabelle:

```
SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# ------- /etc/crontab (Systemweite crontab) -----------------
# (0-6, Sonntag =0)
# Minute - Stunde - Tag - Monat - Wochentag - User - Kommandoscript
8-58/10 5-22 * * * root bash /var/www/cgi-bin/wetterstation2019/minutenbilder2019
05 5-22/1 * * * root bash /var/www/cgi-bin/wetterstation2019/stundenbilder2019
30 12 * * * root bash /var/www/cgi-bin/wetterstation2019/tagesbilder2019
00 0-21/3 * * * root python /var/www/cgi-bin/wetterstation2019/sensorabfrage2019.py
```

Abbildung 88: berichtigte crontab

Eine weitere Verbesserung der automatischen Messscripts wurde vorgenommen, indem die zur selben Zeit ablaufende Scripte aus einem Hauptscript aufgerufen werden und nicht parallel in der **crontab**. Gleichzeitig aufgerufene Prozesse in der **crontab** führen teilweise zu Fehlern.

Die folgende Grafik zeigt die Abfolge der **cron**-Scripts.

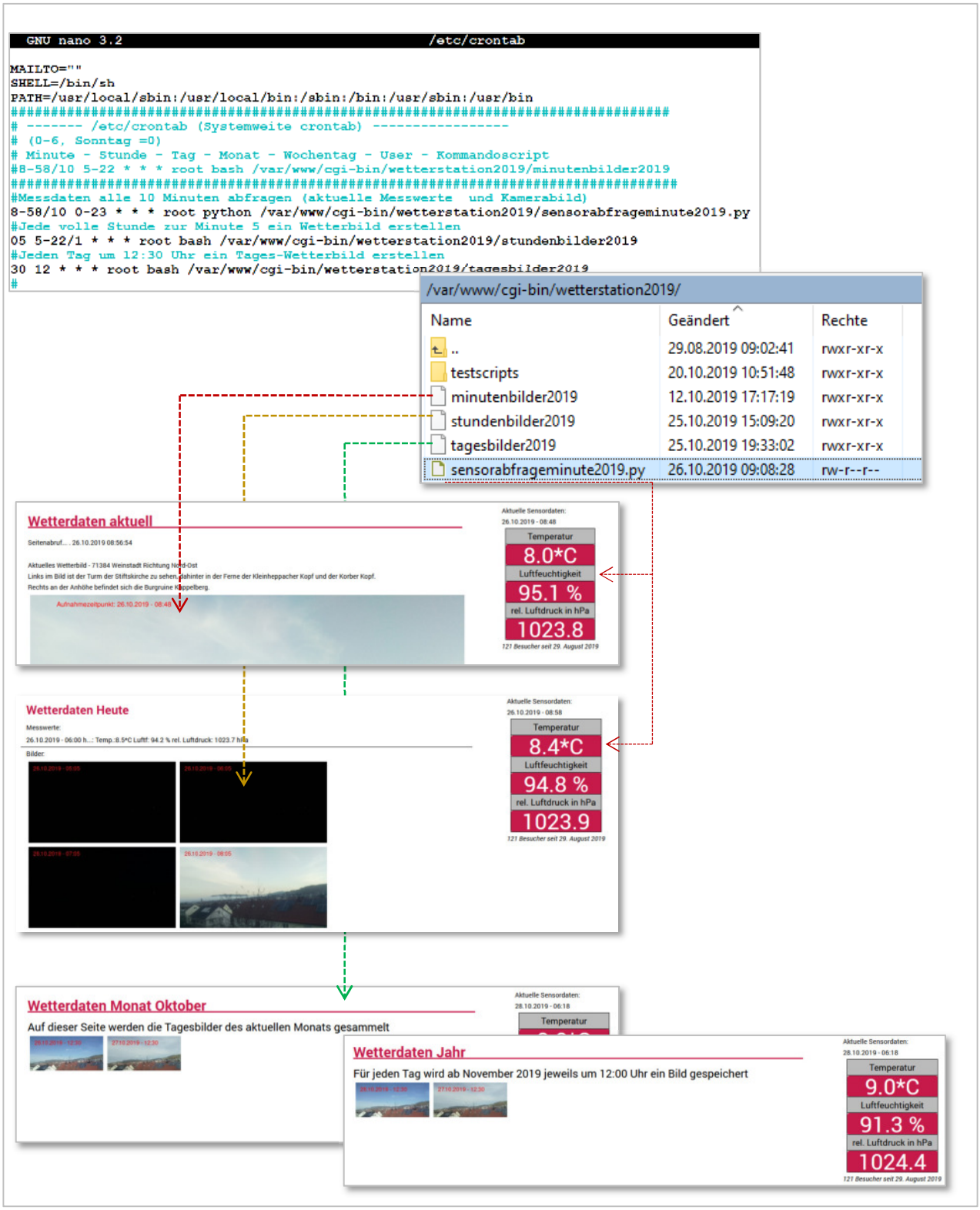


Abbildung 89: Abfolge der crontab-Scripts

11 Messwerte in der Datenbank ablegen

Bisher wurden die Messwerte der Sensoren in einzelnen **`html`**-Dateien, Kamerabilder in **`jpg`**-Dateien gespeichert und dann wieder auf der Webseite dargestellt. Diese Vorgehensweise ist relativ unflexibel und umständlich. Damit die Wetterdaten längere Zeit aufbewahrt und für statistische Zwecke ausgewertet werden können, sollen sie nun in einer Datenbank abgespeichert werden.

Zusätzlich ist angedacht, Grafiken und Auswertestatistiken auf die Website zu integrieren.

Die Dokumentation dieser Erweiterungen wird in einem separaten Dokument erfolgen.

Anhang-Beispielscripte

Im Anhang werden die wichtigsten Scripts aus der Dokumentation zusammengefasst dargestellt.

index.php

```
<?php
setcookie("raspi2","besucht",time()+(60*60*24)); //60Sekunden*60Minuten*24Stunden=1Tag
header('Content-Type: text/html;charset=utf-8');
//Das Cookie raspi2 dient zur Wiedererkennung des Benutzers
?>
<!DOCTYPE HTML>
<html>
<head>
<meta charset="UTF-8">
        <link href="./raspi2019.css" rel="stylesheet" type="text/css">
        <meta name="viewport" content="width=device-width, initial-scale=0.8, user-scalable=yes">
        <link href="favicon.ico" rel="shortcut icon">
        <title>Raspberry PI-Website</title>
</head>
<body>

        <?php
        // ---- Übergabeparameter auslesen: Zieldatei für content
                if(isset($_GET['aw'])) //Inhaltsdatei aus der GET-Übergabe lesen
                        {
                        $strZiel=$_GET['aw'];
                        }
                else    //Startdatei einfügen, beim ersten Aufruf
                        {
                        $strZiel='003inhaltstart.php';
                        }
        //-------Beginn Kopfbereich -------------------
        ?>
        <div id="container"> <!-- Container/ Bereich für den gesamten Inhalt -->

                <div id="header"> <!-- Kopfzeilenbereich für die Überschrift-->
                        <?php
                                include "./000kopfbereich.php";
                        ?>
                </div>

                <div id="content"> <!-- Inhalt wird dynamisch je nach Menüauswahl eingefügt-->
                        <?php include "./" . $strZiel ;?>
                </div>

                <div id="sidebar">
                <!--wird unten und ab 370px rechts gezeigt-->
                <!--um Dopplung zu vermeiden, wird die rechte Seite tlw. ausgeblendet-->
                                <?php
                                        if($strZiel=='30wetterbild.php'){
                                                include "";
                                                                }
                                        else{
                                                include "./004seite.php";
                                                }
                                        ?>
                </div>
        </div>
</body>
</html>
```

000kopfbereich.php

```
<?php
        echo "<p class='titel'>Willkommen auf meinem Raspberry Pi Zero</p>";
?>
<div id="meinlogo">
        <?php
                include "./000logo.php";
        ?>
</div>
<?php
        include "./001hauptmenuequer.php";
?>
```

000logo.php

```
<p class="textklein"><img border="0" height="30px" width="30px" src="./bilderallge-
mein/00logo.png"></p>
```

000seite.php

```
<?php
//aktuelle Wetterdaten anzeigen
include ("./wetterstation2019/daten/00sensoraktuell.html");
//Besucherzähler auslesen
$intZeiger = fopen("../files/raspi2zaehler.txt" , "r+");
$intZaehler = fgets($intZeiger,20);
        if (isset($_COOKIE['raspi2']) && $_COOKIE['raspi2'] == "besucht")
                {
                }
        else{
                $intZaehler=$intZaehler+1;
                rewind($intZeiger);
                fputs($intZeiger,$intZaehler);
                echo "";
        }
fclose($intZeiger);
//Besucherzähler darstellen
echo "<p class='textkleinkursiv'>" . $intZaehler;
echo " Besucher seit 28. November 2019</p>";
?>
```

003inhaltstart.php

```
<?php
echo "<h1>Willkommen auf dem Raspi2</h1>";
echo "<p class='textnormal'>Hier wird der Aufbau und Konfiguration einer Wetterstation mit dem Rasp-
berryPI zero und dem Betriebssysytem Raspbian Buster dokumentiert.<br>";
echo "Die Seite läuft vollständig auf einem RaspberryPI zero W mit WLAN-Anbindung und einem Apache2-
Webserver mit dynamischen PHP-Seiten (PHP7.3). </p>";
$strDatum=date('d.m.Y');
$strStunde=date('H');
$intBildnummer=$strStunde;
echo "<h2>Wetter aktuell</h2>";
echo "<p class='textklein'>";
echo "Messwerte:<br>";
$strDateiname2= "wetterstation2019/daten/" . sprintf('%02d', $intBildnummer) . ".html";
include($strDateiname2);
echo "<br><br>Wetterbild zur ganzen Stunde:<br>";
$strDateiname= "wetterstation2019/bilder/" . sprintf('%02d', $intBildnummer) . "stundenbild.jpg";
echo "<img src='$strDateiname' ></p>";
?>
```

001hauptmenuequer.php

```
<nav>
<!-- Menü für kleine Auflösungen -->
<label for="drop" class="toggle">&#9776; Menü...</label>
<input type="checkbox" id="drop" />
<!-- Menü für alle großen Auflösungen -->
<ul class="menu">
        <!-- Hauptmenü1 -->
        <li><a href="https://www.hans-koch.eu/index.php">&uarr;&uarr;</a></li>
        <li><a href="./index.php">&uarr;</a></li>

        <!-- Hauptmenü2 -->
        <li>
                <!-- Toggelschalter für das Hauptmenü2 -->
                <label for="drop-2" class="toggle">Entwicklung</label><a href="#">Entwicklung </a>
                <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>

                </ul>
        </li>
        <!-- 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  -->
                        <hr>
                        <li><a href="./index.php?aw=300ausbaustufe1.php"><i>Ausbaustufe1</i></a></li>
                        <hr>
                        <li><a href="./index.php?aw=30wetterdaten.php">Aktuelle Wetterdaten</a></li>
                        <li><a href="./index.php?aw=31wetterdatentag.php">Tagesansicht</a></li>
                        <li><a href="./index.php?aw=32wetterdatenmonat.php">Monatsansicht</a></li>
                        <li><a href="./index.php?aw=33wetterdatenjahr.php">Jahresansicht</a></li>
                        <hr>
                        <li><a href="./index.php?aw=340ausbaustufe2.php"><i>Ausbaustufe2</i></a></li>
                        <hr>
                        <li><a href="./index.php?aw=341tag.php">Tagestabelle</a></li>
                        <li><a href="./index.php?aw=342monatstat.php">Statistik 30 Tage</a></li>
                        </ul>
                </li>

                <!-- Hauptmenü2 -->
                <li>
                <!-- Toggelschalter für das Hauptmenü2 -->
                <label for="drop-1" class="toggle">Infos &darr;</label><a href="#">Infos &darr;</a>
                <input type="checkbox" id="drop-1"/>
                <ul> <!-- Untermenüs des Hauptmenü2  -->
                        <li><a href="./index.php?aw=095kontakt.php">Kontakt</a></li>
                        <li><a href="./index.php?aw=095cimp.php">Impressum</a></li>
                </ul>
                </li>
        </ul>
</nav>
```