

Hochschule Bremerhaven

Bachelorarbeit

Thema

**Programmierung einer grafischen Benutzeroberfläche zur
Visualisierung von Luftqualitäts-/Sensormesswerten**

Akademischer Abschlussgrad: Grad, Fachrichtung (Abkürzung)

Bachelor of Engineering (B.Eng.)

Vorname Nachname, Matrikelnummer, Ort

Jan-Ole Wiebusch, 32592, Bülkau(21782)

Studiengang

Gebäudeenergietechnik (GET)

Erstprüfer

Prof. Dr. Mathias Lindemann

Zweitprüfer

Prof. Dr. Thomas Juch

Abgabedatum

05.09.2022

Eidesstattliche Erklärung

Hiermit versichere ich eidesstattlich, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Schriften entnommen sind, wurden als solche kenntlich gemacht. Die Arbeit wurde in gleicher oder ähnlicher Form keinem anderen Prüfungsamt vorgelegt. Mir ist bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

Bülkau, 28.08.2022

Ort, Datum

Jan-Ole Wiebusch

Unterschrift

Abstract

Das Ziel dieser Bachelorarbeit ist die Umsetzung einer geeigneten Visualisierung sensorgestützter Messwerte. In der Hochschule Bremerhaven sollen zukünftig die Raumluftparameter in den Vorlesungsräumen und die Feinstaubbelastung im Außenbereich erfasst und abgespeichert werden.

Der Fokus liegt auf einer kostengünstigen Implementierung mit kostenloser Open-Source Software auf einem Raspberry Pi 4 und preiswerten Sensoren die über ESP32 Mikrocontroller ausgelesen werden können. Die in JavaScript programmierte grafische Benutzeroberfläche soll die Zeitreihendaten übersichtlich zugänglich machen und eine Beurteilung des Raumluftklimas für eine eventuelle Anpassung des Lüftungsverhaltens erleichtern.

Für die Verwaltung und Konfiguration der Sensoren wird die Software **Home Assistant** mit der **ESPHome** Erweiterung genutzt. Als Zeitreihendatenbank wird **InfluxDB** verwendet, die besonders für den Umgang mit mit einem Zeitstempel versehenen Informationen ausgelegt ist.

Inhaltsverzeichnis

Abbildungsverzeichnis	IV
Quellcodeverzeichnis	VII
Abkürzungsverzeichnis	VIII
1 Einleitung	1
2 Auswahl der Software	2
3 Einrichtung der Hard- und Software	4
3.1 Auswahl des Betriebssystems	4
3.2 Installationen unter Linux	6
3.3 Docker Installation	6
3.4 Docker Home Assistant Supervised	7
3.4.1 OS-Agent	7
3.4.2 Installation Home Assistant Supervised	8
3.4.3 Docker Commands	8
3.4.4 IP-Adresse herausfinden	9
3.5 Home Assistant Add-Ons	10
3.5.1 Add-On File editor	10
3.5.2 ESPHome	11
3.6 NodeJs Installieren	12
3.6.1 SSH aktivieren	12
3.7 nginx	14
3.7.1 Installation	14
3.7.2 Terminal Commands	15
3.7.3 nginx Konfiguration	15
3.8 InfluxDB Installation	18
4 Sensormesswerte	22
4.1 ESP32	22
4.2 DHT22 - Temperatur- und Luftfeuchtigkeitssensor	23
4.3 SDS011 - Feinstaubsensor	24
4.4 MH-Z19 - Kohlenstoffdioxid-Sensor	25
4.5 BME280 - Temperatur-, Luftfeuchtigkeit- und Luftdrucksensor	26

4.6	Verschaltung aller Sensoren	26
4.7	Konfiguration in ESPHome	28
4.7.1	Neues Gerät hinzufügen	28
4.7.2	Board, Wifi und Gerätenamen festlegen	30
4.7.3	DHT22 Konfiguration	31
4.7.4	SDS011 Konfiguration	31
4.7.5	BME280 Konfiguration	32
4.7.6	MH-Z19 Konfiguration	32
4.7.7	Vollständige YAML Konfiguration	33
4.8	Home Assistant InfluxDB Integration	34
5	JavaScript Grundlagen	36
5.1	Kommentare	36
5.2	Variablen	36
5.3	Scope	36
5.4	Datentypen	38
5.4.1	Arrays	38
5.4.2	Objects	40
5.4.3	Functions	41
5.5	ReactJs	41
5.5.1	Styling in React	42
5.5.2	React Props	44
5.5.3	Fractions	44
5.5.4	useState Hook	45
5.5.5	useEffect Hook	45
5.5.6	Conditional Rendering	46
5.5.7	Mapping von Arrays/Objects	47
6	Programmaufbau	49
6.1	Layout	49
6.2	Authentication	55
6.2.1	Implementierung	56
6.3	InfluxDB Queries	62
6.3.1	Integration Influx Query	63
6.4	Routing	66
6.5	Aufbereitung der Home Assistant Daten	68

6.6	Hinweise zum React Quellcode	70
6.6.1	Aufbau und Ordnerstrukturen	72
6.6.2	Installation der React Application	72
7	Diskussion	75
8	Fazit	77
	Literaturverzeichnis	78

Abbildungsverzeichnis

1	Raspberry Pi Prozessor Architekturen	4
2	Raspberry Pi Imager	6
3	Docker Installation on Ubuntu	7
4	Docker Container im Terminal anzeigen	9
5	IP-Adresse im Terminal anzeigen lassen	9
6	Home Assistant Benutzerkonto erstellen	10
7	Home Assistant Add-On Store	11
8	File Editor Installation	11
9	Windows SSH	13
10	Windows SSH Neuen Benutzer anlegen	14
11	nginx Configuration	15
12	Nginx Homepage	18
13	InfluxDB Einrichtung	18
14	InfluxDB User Registrieren	19
15	InfluxDB User Registrieren ausgefüllt	19
16	InfluxDB Bucket Retention Zeit verändern	20
17	Edit Bucket	20
18	Bucket Retention Time Optionen	20
19	InfluxDB API Token	21
20	InfluxDB neuen Ready Only Token erstellen	21
21	ESP32 Pinout AZ-Delivery	23
22	ESP32 mit DHT22 Steckplatine	24
23	ESP32 mit DHT22 Schaltplan	24
24	ESP32 mit SDS011	25
25	ESP32 mit SDS011 Schaltplan	25
26	ESP32 mit MH-Z19	25
27	ESP32 mit MH-Z19 Schaltplan	25
28	ESP32 mit BME280	26
29	ESP32 mit BME280 Schaltplan	26
30	Mögliche Verschaltung von DHT22, MH-Z19 und SDS011 an einem ESP32	27
31	DHT22, MH-Z19 und SDS011 an einem ESP32 - Schaltplan	27
32	ESPHome neues Gerät hinzufügen	28
33	ESP Version Auswahl	29
34	Geräte in ESPHome	29

35	Page Layout	49
36	Layout Body - Login Page	49
37	Layout Body - Authenticated	49
38	Page Layout Responsive	50
39	Login Page	50
40	Tabellenübersicht	51
41	Grenzwerte für Feinstaubbelastung	51
42	Menü geschlossen	52
43	Menü geöffnet	52
44	einzelne Sensor Seite	52
45	Responsive Design für Tablets/Phone	53
46	Graphen anzeigen lassen	54
47	Graphen Messpunkte anzeigen	54
48	Behaglichkeitsfeld für Temperatur und Luftfeuchte nach Frank	55
49	Microsoft OAuth2.0 Protokolldiagramm	56
50	WebSocket Entities Object	62
51	Graph - Query mit createEmpty: false	63
52	Graph - Query mit createEmpty: true	63
53	Entities Object Gruppierung	69
54	Entities Array	70
55	Ordnerstruktur Code	72
56	React Komponenten Ordner Struktur	72
57	Git Bash deploy.sh	73
58	Deploying Files to Server	74

Quellcodeverzeichnis

1	nginx Cofiguration File	16
2	Nginx home.html	17
3	YAML Configuration File	30
4	YAML Configuration DHT22	31
5	YAML Configuration SDS011	31
6	YAML Configuration BME280	32
7	YAML Configuration MH-Z19	32
8	YAML Configuration Kombination	33
9	Home Assistant Configuration File	34
10	Kommentare	36
11	Scope	37
12	Function and Block Scope	37
13	Arrays	39
14	let und const mit Arrays	39
15	Array Spread-Operator	39
16	Array Destructuring	40
17	Objects	40
18	Verschachtelung von Objekten	40
19	Function	41
20	React	42
21	Styling CSS-File	42
22	Styling mit Inline-Styles	43
23	Styling mit Styled-Components	43
24	React Props	44
25	React Fractions	44
26	React useState Hook	45
27	React useEffect Hook	45
28	Conditional Rendering - && Operator	46
29	Conditional Rendering - Ternary Operator	46
30	Array.map()	47
31	Object.map()	47
32	Koordinaten Behaglichkeitsfelder	55
33	AuthContext	56
34	Login Function	57

35	Authentifizierung überprüfen	58
36	Logout Function	59
37	Authentication Implementierung	59
38	Beispiel Influx Query	62
39	Influx Client	63
40	Influx Query Integration	63
41	State Initialisation	64
42	Update Query Parameter	65
43	Dynamic Influx Query	65
44	React Router Implementierung	66
45	React Router Layout	67
46	Custom Hook useAuth	68
47	converEntities Funktion	68
48	objectToArray Funktion	70
49	package.json File	71
50	deploy.sh - Bash-Script zur Installation der React Application	73

Abkürzungsverzeichnis

DOM Document Object Model

OS Operating System

YAML Yet Another Markup Language

ota over the air

npm Node Package Manager

SSH Secure Shell

URL Uniform Resource Locator

PM Particular Matter

UART Universal Asynchronous Receiver Transmitter

BUS Binary Unit System

UI User Interface

CSS Cascading Style Sheets

HTML Hypertext Markup Language

API Application Programming Interface

GUI Graphical User Interface

1 Einleitung

In den meisten Bereichen stellt Deutschland, was das Thema Digitalisierung oder das Internet der Dinge angeht, eher das Schlusslicht im internationalen Ranking dar. Was einerseits für Ernüchterung sorgt, bietet andererseits noch genügend Potential für Verbesserung, die es zu realisieren gilt[1].

Gerade in der Gebäudetechnik ist die Überwachung des Raumluftklimas durch die Auswirkungen der Coronapandemie stark in den Fokus gerückt. Bei zu niedriger Temperatur und geringer Luftfeuchtigkeit unterkühlt der Körper und die Schleimhäute trocknen aus, was wiederum die Anfälligkeit gegenüber Viren und Bakterien erhöht. Bei zu hoher Temperatur und Luftfeuchtigkeit wird die Luft als stickig empfunden, Müdigkeit und Unkonzentriertheit nehmen zu. Eine unzureichende Belüftung in Innenräumen mit vielen Menschen erhöht das Ansteckungsrisiko zudem außerordentlich[2].

Im Rahmen dieser Arbeit soll eine grafische Benutzeroberfläche zur Darstellung der Raumluftparameter programmiert werden, die auf sensorgestützte Messwerte zurückgreift. Als Vorbereitung ist zu ermitteln wie Sensoren über eine geeignete Software-Lösung eingebunden und verwaltet, sowie die gesammelten Daten über eine zeitreihenbasierte Datenbank abgespeichert werden können. Erst eine immer größer werdende Community für Open Source Software (auch im Bereich der IoT-Anwendungen) in Kombination mit preiswerter Hardware, ermöglicht eine kostengünstige Umsetzung einer Smart Home ähnlichen Raumluftüberwachung.

Die Erfassung der Luftparameter soll über einen ESP32-Mikrocontroller realisiert werden, an den Sensoren für Temperatur, Luftfeuchtigkeit, Luftdruck und Kohlenstoffdioxid, sowie Feinstaub angeschlossen werden. Als Host und lokaler Webserver dient ein Raspberry Pi 4, auf dem die Daten abgespeichert werden. Die Messwerte sollen dem Benutzer im lokalen Netzwerk über ein Interface im Browser zugänglich gemacht werden.

2 Auswahl der Software

Ausgangspunkt der Arbeit ist eine geeignete Software für die Einbindung und Verwaltung von selbst verkabelten Sensoren, die über einen ESP32 Mikrocontroller angesteuert und ausgelesen werden können.

Von Herrn Prof. Dr. Lindemann wurde das Tool **Home Assistant** an mich herangetragen. Home Assistant ist eine Open-Source Software für das Smart Home, die lokal ohne Clouddanbindung das Monitoring und die Automation vieler Geräte und Services unterstützt [3]. Der Funktionsumfang kann durch communitybasierte Add-Ons noch erweitert werden. Außerdem ist Home Assistant genau für die Anwendung auf Einplatinencomputern, wie dem Raspberry Pi ausgelegt.

Ein hilfreiches Add-On ist **ESPHome**. Dieses wurde speziell für die Konfiguration von ESP32 und ESP8266 Mikrocontrollern entworfen und ermöglicht die Integration von Sensoren ohne Code, über in Yet Another Markup Language (YAML) verfasste Configuration Files [4].

Als zeitreihenbasierte Datenbank bietet sich **InfluxDB** an. InfluxDB ist nicht nur eine skalierbare und effiziente Datenbank im Umgang mit Zeitreihendaten, sondern lässt sich auch einfach in Home Assistant integrieren. Die Daten werden in so genannten **Buckets** gespeichert, für die ein Zeitraum festgelegt werden kann, nach dem die alten Daten automatisch gelöscht werden, um einen Überlauf zu verhindern.

Für die Visualisierung gibt es in Home Assistant und InfluxDB Dashboards oder zusätzliche Software wie z.B. Grafana, die die aktuellen Messwerte und Graphen über vorgefertigte Templates anzeigen lassen können. Home Assistant und InfluxDB ermöglichen allerdings auch die Abfrage von Daten über eine Web-Socket Connection oder HTTP-Requests.

Aus persönlicher Präferenz wird die Visualisierung mit JavaScript und der Library **ReactJS** realisiert. Die eigene Programmierung bietet die Möglichkeit den Funktionsumfang der Anwendung auf das Wesentliche zu beschränken und die Darstellung flexibler, sowie das User-Interface anpassbarer zu gestalten. Außerdem ist der Mehrwert aus der Umsetzung eines Projektes, in einer Programmiersprache die so nicht zum Umfang des Studiums gehört, größer und lässt sich auch später in anderen Bereichen noch anwenden, wo hingegen der Umgang mit einem spezifischen Programm vielleicht später nie wieder Anwendung findet.

ReactJS ist eine Open-Source Library, die von Meta (ehemals Facebook) entwickelt wurde, und über State-Management das Document Object Model (DOM) des Browsers updaten kann [5].

Das DOM ist eine Repräsentation aller HTML-Elemente einer Seite in Baum-Struktur

ähnlicher Form mit Parent-Child Beziehungen [6]. React ist geeignet für Einsatz mit sich ständig ändernden Messwerten, da das Updaten des Bildinhaltes ohne das Neuladen der gesamten Seite erfolgen kann. React updated nur die Elemente des DOM's, bei denen State Änderungen stattgefunden haben. Dies ermöglicht zudem eine gute Performance.

3 Einrichtung der Hard- und Software

Dieses Kapitel beschäftigt sich mit der Auswahl eines passenden Betriebssystems für den Raspberry Pi 4 und der Installation der benötigten Software. Die aufgezeigten Schritte basieren auf der erneuten Einrichtung eines zweiten Raspberry Pi's, nach der erstmaligen Umsetzung des Projektes und können gleichzeitig als Anleitung aufgefasst werden. Es sei gesagt, dass Software ständig Änderungen unterliegt und dass durch Updates die Installationsmethoden abweichen können oder sich gänzlich ändern. Deshalb beziehen sich die folgenden Abschnitte in erster Linie auf die verwendete Softwareversion und es sollte gegebenenfalls immer die offizielle Webseite des Herstellers oder die Dokumentation des Programms zu Hilfe genommen werden, um Abweichungen zu erkennen oder Unklarheiten zu beseitigen.

3.1 Auswahl des Betriebssystems

Für die Auswahl eines funktionsfähigen Betriebssystems ist in erster Linie die Prozessor-Architektur und die unterstützte Bandbreite entscheidend. Der Raspberry Pi 4 basiert auf der 64-bit arm64, bzw. aarch64 Architektur (siehe Abbildung 1).

Product	Processor	ARM core	Debian/Raspbian ARM port (maximum)	Architecture width
Raspberry Pi 1	BCM2835	ARM1176	arm6hf	32 bit
Raspberry Pi 2	BCM2836	Cortex-A7	armhf	32 bit
Raspberry Pi Zero	BCM2835	ARM1176	arm6hf	32 bit
Raspberry Pi Zero 2	BCM2710	Cortex- A53	arm64	64 bit
Raspberry Pi 3	BCM2710	Cortex- A53	arm64	64 bit
Raspberry Pi 4	BCM2711	Cortex- A72	arm64	64 bit

Abbildung 1: Raspberry Pi Prozessor Architekturen
[7]

Außerdem muss das Betriebssystem für den Raspberry Pi die Installationen folgender Programme unterstützen:

- InfluxDB OSS 2.3 [8]
- Docker mit Home Assistant (supervised) [9]
- Nginx 1.18 [10]
- Nodejs 16.16.0 LTS [11]

Ein geeignetes Betriebssystem ist z.B. Ubuntu Desktop 22.04 LTS, welches für die Verwendung eines Raspberry Pi mit mindestens 2 GB Arbeitsspeicher zertifiziert ist und alle obigen Programme unterstützt. Die Desktopversion verfügt gegenüber der Serverversion über eine grafische Benutzeroberfläche, was die Bedienung für Benutzer, die noch keine Erfahrung im Umgang mit Linux und SSH haben, deutlich vereinfacht - bei remoteverwalteten Servern allerdings nicht benötigt wird. Beide Versionen wären für den geforderten Anwendungsfall denkbar.

Der Download ist über die offizielle Ubuntu Seite verfügbar [12]. Eine detaillierte Anleitung für die Installation auf einem Raspberry Pi 4 stellt Ubuntu ebenfalls bereit [13].

Über den Raspberry Pi Imager (siehe Abbildung 2) wird ein bootfähiges Betriebssystem auf ein externes Speichermedium, z.B. einen USB-Stick oder eine MikroSD Karte überspielt. Der Datenträger sollte, falls notwendig, vor der Installation noch einmal formatiert werden, um eventuellen Problemen beim Bootvorgang vorzubeugen. Nach erfolgreicher Erstellung des Images kann das Speichermedium entnommen, in den Raspberry Pi übertragen und die Einrichtung des Betriebssystems abgeschlossen werden.



Abbildung 2: Raspberry Pi Imager
[12]

3.2 Installationen unter Linux

Die Installation von neuen Programmen auf einer Debian-basierten Linux-Distribution wird über das Command Line Interface(cli) bzw. Terminal vorgenommen. Ein Download eines Pakets erfolgt dabei meist aus einer in einem Software-Repository hinterlegten Liste mit gängigen Programmen. Vor der Installation eines oder mehrerer Programme ist es sinnvoll mit dem Command `sudo apt-get update` oder `sudo apt update` die Paket Liste zu aktualisieren, um auch die neusten Versionen und Abhängigkeiten installieren zu können. Mit `sudo apt upgrade` lässt sich bereits installierte Software auf den neusten Stand bringen. Das Word `sudo` impliziert die Ausführung als Admin und erfordert bei erstmaligem Einsatz die Eingabe des Admin-Passworts[14].

3.3 Docker Installation

Für die Ubuntu Version 22.04(LTS) existiert eine Installationsanleitung in der Docker Dokumentation [15]. In Abbildung 3 sind alle notwendigen Terminal Commands noch

einmal abgebildet.

Set up the repository

1. Update the `apt` package index and install packages to allow `apt` to use a repository over HTTPS:

```
$ sudo apt-get update

$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release
```

2. Add Docker's official GPG key:

```
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
```

3. Use the following command to set up the repository:

```
$ echo \
    "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Install Docker Engine

1. Update the `apt` package index, and install the *latest version* of Docker Engine, containerd, and Docker Compose, or go to the next step to install a specific version:

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Abbildung 3: Docker Installation on Ubuntu
[15]

Die Installation kann mit dem Command `sudo docker run hello-world` oder `docker --version` überprüft werden.

3.4 Docker Home Assistant Supervised

Für die Verwendung von Open-Source Plug-Ins wie ESPHome mit Home Assistant, ist zwingend die supervised Version von Home Assistant erforderlich.

3.4.1 OS-Agent

Der Home Assistant Operating System (OS)-Agent ermöglicht die Kommunikation mit dem Host Betriebssystem. Die Installation für Linux aarch64 Debian erfolgt wieder im Terminal über

```
wget https://github.com/home-assistant/os-agent/releases/tag/1.2.2/os-agent_1.2.2_linux_aarch64.deb
, gefolgt von sudo dpkg -i os-agent_1.2.2_linux_aarch64.deb.
```

Die Installation kann mit dem Command

`gdbus introspect -system -dest io.hass.os -object-path /io/hass/os` überprüft werden, dieser sollte keinen Error zurück geben[16].

3.4.2 Installation Home Assistant Supervised

Die Installation von Home Assistant Supervised ermöglicht die Benutzung der gesamten Home Assistant Anwendung auf einem regulären Betriebssystem, ohne das native Home Assistant Operating System. Es wird darauf hingewiesen, dass verwendete Komponenten und Pakete sich mit der Zeit ändern können und neuere Versionen erscheinen. Für die Installation und Wartung dieser Komponenten ist der Verwender selbst verantwortlich. Mit folgendem Command werden alle für Home Assistant Supervised benötigten Dependencies installiert `apt-get install jq wget curl udisks2 libglib2.0-bin network-manager dbus -y`. Bei dem erforderlichen jq Package kann es zu Problemen bei der Installation kommen, die sich mit `sudo apt -fix-broken install` allerdings beheben lassen. Die Installation des Home Assistant Supervised Debian Packages erfolgt mit folgenden Commands:

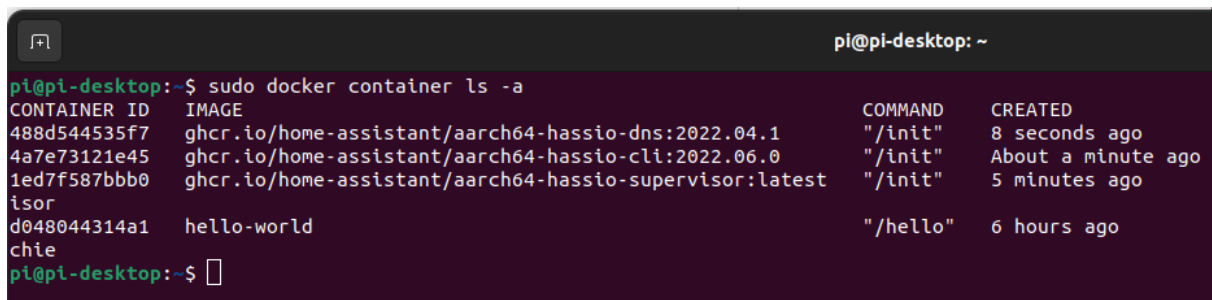
1. `wget https://github.com/home-assistant/supervised-installer/releases/latest/download/homeassistant-supervised.deb`
2. `dpkg -i homeassistant-supervised.deb`

Anschließend wird aufgefordert den verwendeten Gerätetyp auszuwählen, in diesem Fall `Raspberrypi4-64`. Die Anleitung ist auf der home-assistant Github Seite unter supervised-installer zu finden[9].

3.4.3 Docker Commands

Folgende Befehle sind hilfreich um Docker Container zu verwalten:

- `sudo docker container ls -a` - Zeigt alle Docker Container an (siehe Abbildung 4)
- `sudo docker container rm <ID>` - Container entfernen mithilfe der Container ID
- `sudo docker stop <name>` - Container Stoppen
- `sudo docker restart <name>` - Container Restart

A terminal window on a Raspberry Pi desktop. The prompt is 'pi@pi-desktop: ~'. The command 'sudo docker container ls -a' has been executed. The output is a table with four columns: CONTAINER ID, IMAGE, COMMAND, and CREATED. There are five containers listed: three from ghcr.io/home-assistant (aarch64-hassio-dns, aarch64-hassio-cli, aarch64-hassio-supervisor) and two others (isor, chie).

CONTAINER ID	IMAGE	COMMAND	CREATED
488d544535f7	ghcr.io/home-assistant/aarch64-hassio-dns:2022.04.1	"/init"	8 seconds ago
4a7e73121e45	ghcr.io/home-assistant/aarch64-hassio-cli:2022.06.0	"/init"	About a minute ago
1ed7f587bbb0	ghcr.io/home-assistant/aarch64-hassio-supervisor:latest	"/init"	5 minutes ago
isor			
d048044314a1	hello-world	"/hello"	6 hours ago
chie			

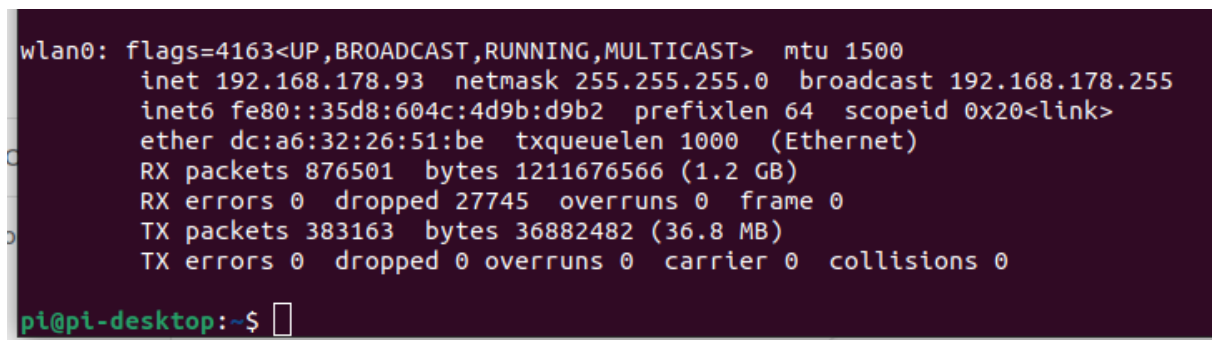
Abbildung 4: Docker Container im Terminal anzeigen

3.4.4 IP-Adresse herausfinden

Die IP-Adresse wird für die Einrichtung noch häufiger benötigt. Über die IP-Adresse kann auf die Home Assistant Instanz und später auch InfluxDB, sowie die eigenen Sensordaten Seite zugegriffen werden. Um die IP-Adresse zu bestimmen wird das Package `net-tools` benötigt.

1. `sudo apt install net-tools`
2. `ifconfig`

Die IP-Adresse kann unter wlan0 — inet: <ip> gefunden werden (siehe Abbildung 5).

A terminal window on a Raspberry Pi desktop. The prompt is 'pi@pi-desktop: ~'. The command 'ifconfig' has been executed. The output shows details for the wlan0 interface, including flags, mtu, inet address (192.168.178.93), netmask, broadcast, inet6 address, ether address, and RX/TX statistics.

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.178.93 netmask 255.255.255.0 broadcast 192.168.178.255
    inet6 fe80::35d8:604c:4d9b:d9b2 prefixlen 64 scopeid 0x20<link>
    ether dc:a6:32:26:51:be txqueuelen 1000 (Ethernet)
    RX packets 876501 bytes 1211676566 (1.2 GB)
    RX errors 0 dropped 27745 overruns 0 frame 0
    TX packets 383163 bytes 36882482 (36.8 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

pi@pi-desktop:~$
```

Abbildung 5: IP-Adresse im Terminal anzeigen lassen

Dem verwendeten Raspberry Pi wurde beispielsweise die IP-Adresse `192.168.178.93` zugewiesen.

Home Assistant kann nach einer Einrichtungsdauer von ca. 20 Minuten auf einem anderen Computer über die herausgefundene IP-Adresse auf port `<ip>:8123` erreicht werden. Ein Benutzerkonto sollte aus Gründen der IT-Sicherheit zeitnah erstellt werden, da jeder im gleichen Netzwerk auf die Seite zugreifen kann und das Admin Konto übernehmen könnte (siehe Abbildung 6).

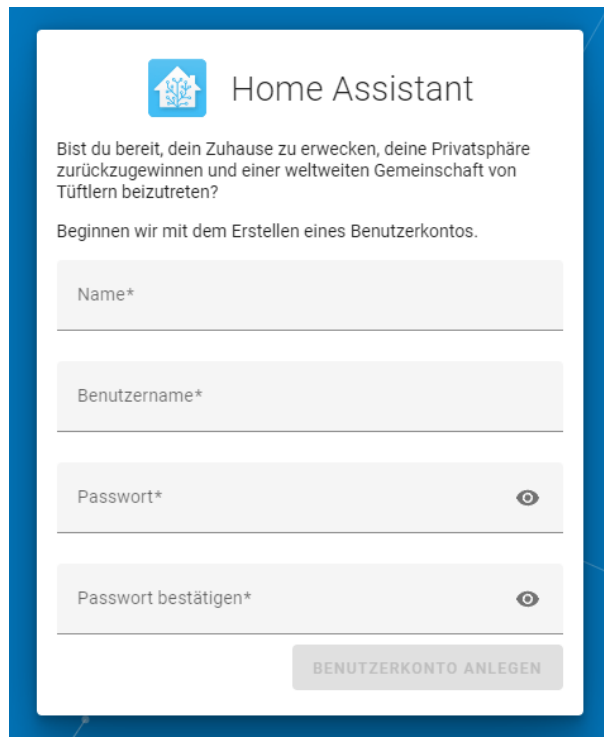
The image shows a web form for creating a Home Assistant user account. At the top, there is a Home Assistant logo (a house with a tree) and the text 'Home Assistant'. Below this, a paragraph asks if the user is ready to wake up their home, regain their privacy, and join a global community of tinkerers. It then says 'Beginnen wir mit dem Erstellen eines Benutzerkontos.' (We start with creating a user account). The form contains four input fields: 'Name*', 'Benutzername*', 'Passwort*' (with an eye icon to toggle visibility), and 'Passwort bestätigen*' (also with an eye icon). At the bottom right of the form is a button labeled 'BENUTZERKONTO ANLEGEN'.

Abbildung 6: Home Assistant Benutzerkonto erstellen

3.5 Home Assistant Add-Ons

Home Assistant bietet zahlreiche Add-Ons an, die den Funktionsumfang der Home Assistant Anwendung erweitern. Dazu gehören auch von der Community erstellte Add-Ons, die allerdings nur mit der Home Assistant Supervised Version verfügbar sind. Der Add-On Store ist über `Einstellungen -> Add-ons -> ADD-ON STORE` erreichbar. In diesem Projekt werden zwei Add-Ons verwendet, `File editor` und `ESPHome` (siehe Abbildung 7).

3.5.1 Add-On File editor

Der `File editor` übernimmt die Funktion eines File-Explorers mit integriertem Text-Editor. Dieser ist später hilfreich, um das Configuration File für die Übertragung der Sensordaten an die Datenbank einzurichten.

Jedes Add-On besitzt Einstellungen für den Autostart und Neustart bei Abstürzen (siehe Abbildung 8). Außerdem kann jedes Add-On ganz gestoppt oder deinstalliert werden, wenn es nicht mehr benötigt wird.

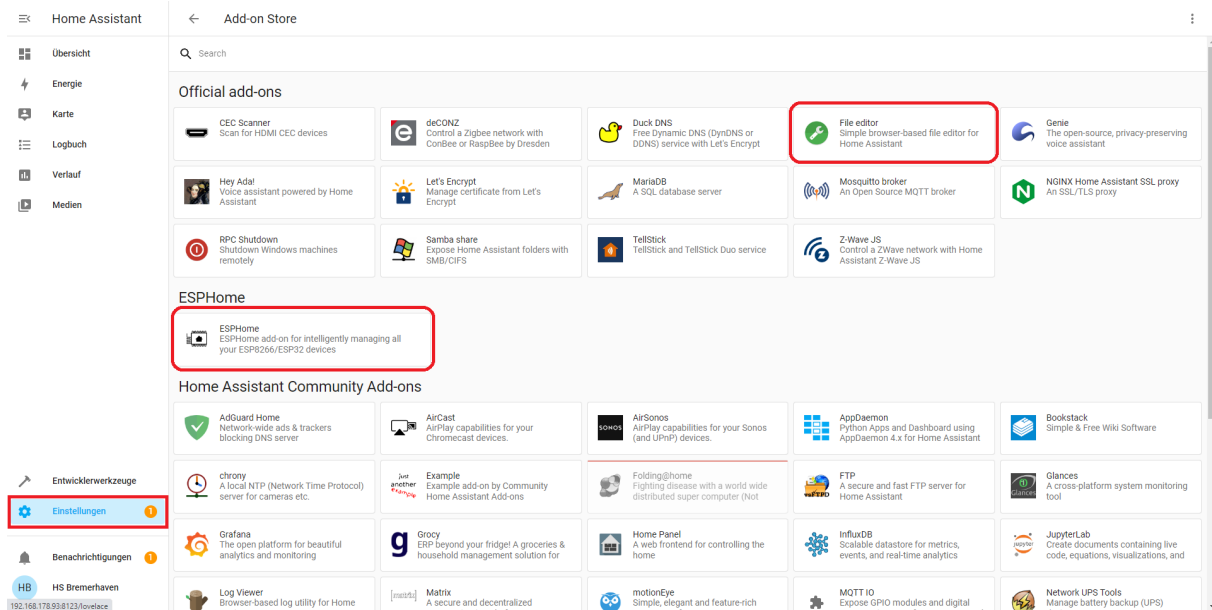


Abbildung 7: Home Assistant Add-On Store

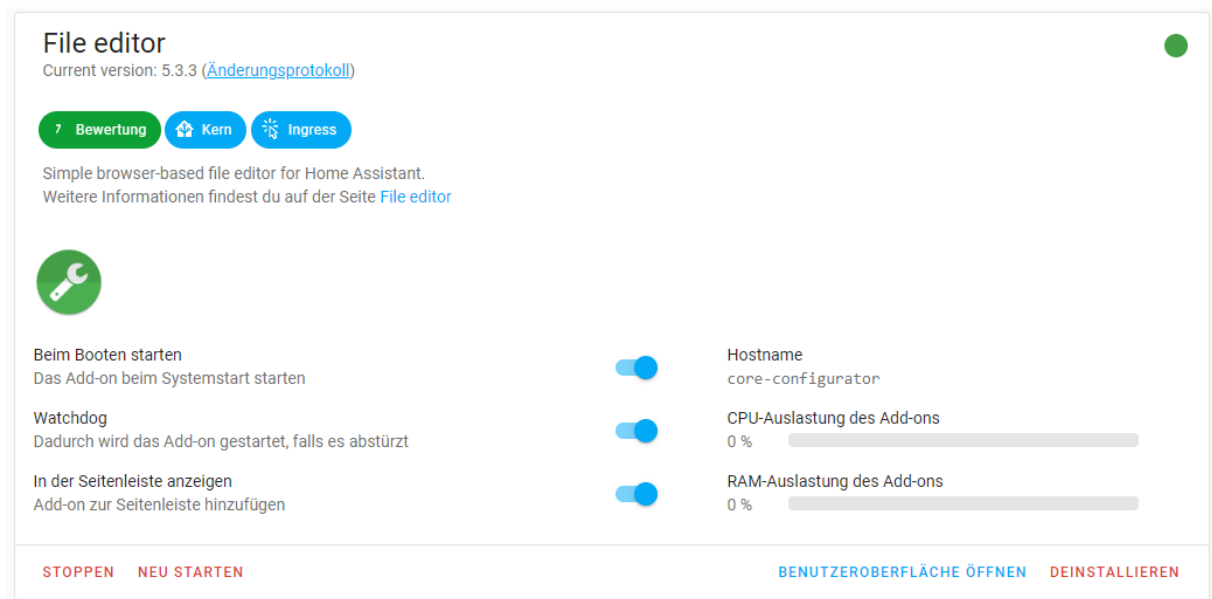


Abbildung 8: File Editor Installation

3.5.2 ESPHome

ESPHome ist für die Einbindung von ESP32 basierten Mikrocontrollern gedacht, aber auch für ältere Modelle wie den ESP8266 [4]. Die Erweiterung ermöglicht die Einrichtung und Verwaltung der ESP Geräte über Configuration Files im YAML-Format. ESPHome generiert den Quellcode aus der YAML-Datei, bevor dieser auf den Mikrocontroller

überspielt wird. Die Datenübertragung ist über den Raspberry Pi direkt, aber auch mit jedem anderen Computer, der über das lokale Netzwerk auf die Home Assistant Webseite zugreifen kann, möglich. Nach der erstmaligen Installation der ESP Firmware auf dem ESP Mikrocontroller wird außerdem die Übertragung over the air (ota) mittels W-Lan unterstützt. Dies erleichtert die Installation von Updates oder die nachträgliche Bearbeitung des Configuration Files immens.

3.6 NodeJs Installieren

NodeJs wird für die Verwendung des Node Package Manager (npm)s benötigt, welcher wiederum für die Installation von JavaScript Libraries und Paketen notwendig ist. NodeJs und npm können wie folgt installiert werden:

1. `sudo apt install nodejs`
2. `node -v`
3. `sudo apt install npm`

Das Update auf die neuste, stabile Version erfolgt durch die folgenden Befehle:

4. `sudo npm cache clean -f`
5. `sudo npm install -g n`
6. `sudo n stable`
7. `node -v`

3.6.1 SSH aktivieren

Secure Shell (SSH) ist ein Netzwerkprotokoll, das die Bedienung eines anderen Computers über das Terminal ermöglicht. Mit `ssh` können Befehle von einem Computer aus auf dem Raspberry Pi ausgeführt werden. Folgende Schritte sind für die Installation notwendig:

1. `sudo apt install openssh-server`
2. `sudo service ssh status`

Der Zugriff auf den Raspberry Pi von einem anderen Computer erfolgt mit folgenden Befehlen (siehe auch Abbildung 9):

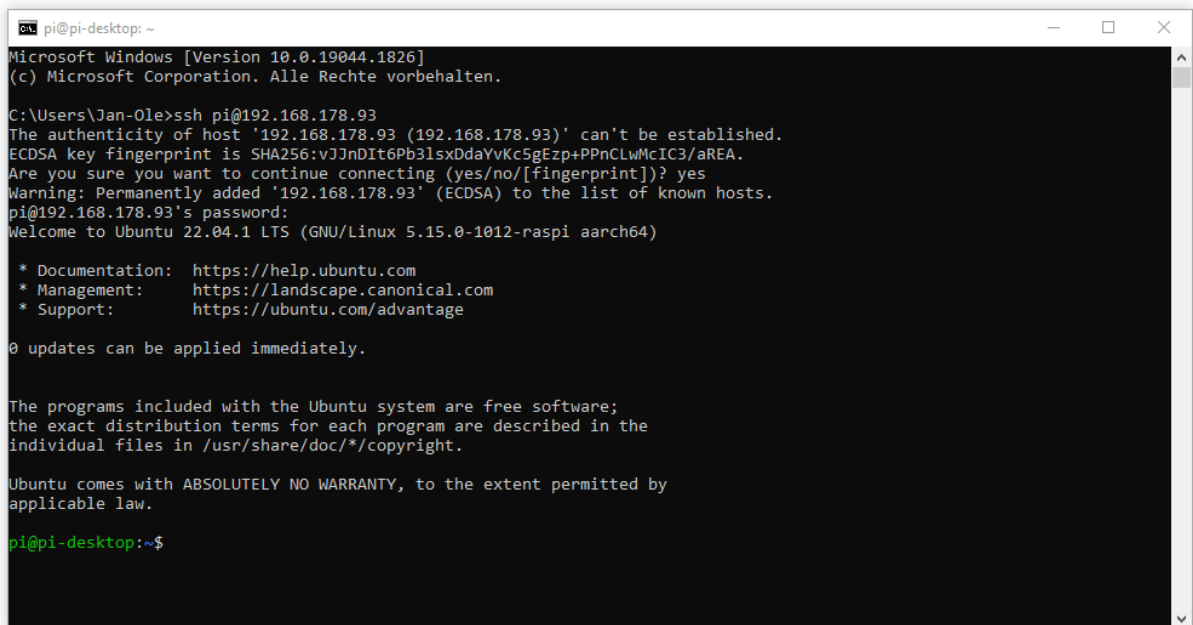


Abbildung 9: Windows SSH

- `ssh <name>@<ip-address>`
- `ssh pi@192.168.178.93`

Für den SSH-Zugang kann außerdem ein neuer Benutzer angelegt werden. Mit der Ergänzung `sudo` am Ende wird der neue Benutzer der Gruppe `sudo` mit Adminrechten hinzugefügt. Der Befehl um einen neuen User mit dem Namen `pireact` anzulegen lautet:

- `sudo adduser pireact sudo`

Anschließend muss ein Passwort definiert werden (siehe Abbildung 10).

Mit `ctrl + d` kann die Verbindung wieder getrennt werden.

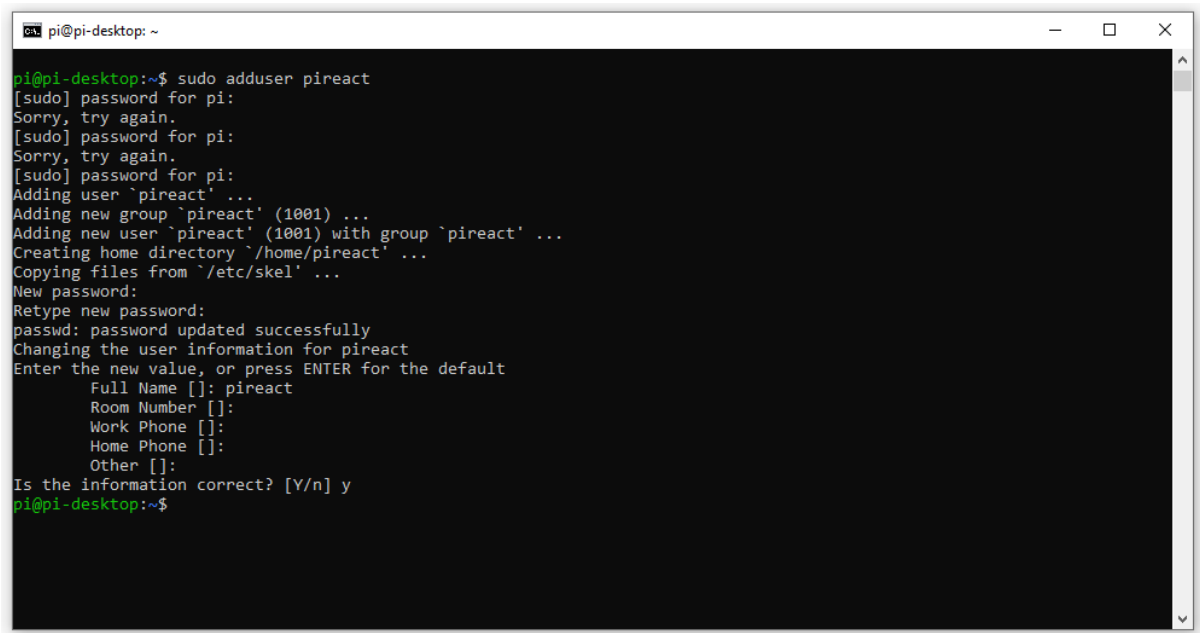


Abbildung 10: Windows SSH Neuen Benutzer anlegen

3.7 nginx

Nginx ist eine Webserver-Software, die modular aufgebaut ist und auch Techniken wie Lastverteilung, Netzwerksicherheit und Verschlüsselung unterstützt[10]. Für diesen Anwendungszweck soll nginx als Webserver die Uniform Resource Locator (URL)-Anfragen überwachen und dementsprechend an die `index.html` Datei der React Application verweisen.

3.7.1 Installation

Die Installation erfolgt über:

1. `sudo apt install nginx`
2. `sudo systemctl status nginx`

Folgende Befehle sind hilfreich im Umgang mit nginx:

- `sudo systemctl stop nginx`
- `sudo systemctl start nginx`
- `sudo systemctl restart nginx`
- `sudo systemctl reload nginx`

- `sudo systemctl disable nginx`
- `sudo systemctl enable nginx`

3.7.2 Terminal Commands

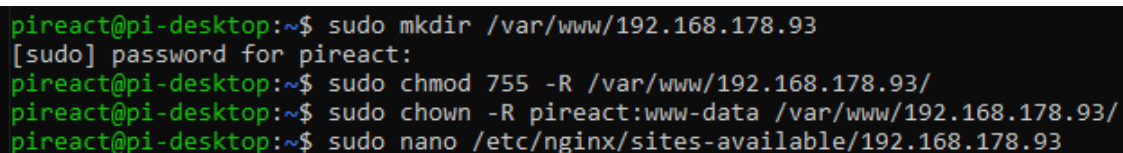
Mit Hilfe von `cd` (change directory) lässt sich innerhalb der Ordnerstrukturen navigieren. Durch das Drücken der Tab-Taste kann ein angefangener Datei- oder Ordnername automatisch vervollständigt werden, falls nur noch eine passende Bezeichnung zutrifft. Weitere nützliche Commands sind:

- `cd <folder>`
- `cd ..` vorheriger Ordner
- `cd ~` root directory
- `ls` zeigt alle Ordner und Dateien im aktuellen Ordner an

3.7.3 nginx Konfiguration

In einem Configuration File muss festgelegt werden, welche Hypertext Markup Language (HTML)-Dateien bei bestimmten URL-Anfragen ausgeführt werden sollen[17]. Der Einrichtungsprozess lautet wie folgt (siehe auch 11):

1. `sudo mkdir /var/www/<ip>` erstellt einen Ordner
2. `sudo chmod 755 -R /var/www/<ip>/` ermöglicht Allen den Inhalt des Ordners zu öffnen
3. `sudo chown -R <user>:www-data /var/www/<ip>/` definiert den Benutzer als Besitzer des Ordners
4. `sudo nano /etc/nginx/sites-available/<ip>` öffnet die Datei



```
pireact@pi-desktop:~$ sudo mkdir /var/www/192.168.178.93
[sudo] password for pireact:
pireact@pi-desktop:~$ sudo chmod 755 -R /var/www/192.168.178.93/
pireact@pi-desktop:~$ sudo chown -R pireact:www-data /var/www/192.168.178.93/
pireact@pi-desktop:~$ sudo nano /etc/nginx/sites-available/192.168.178.93
```

Abbildung 11: nginx Configuration

Im Configuration File wird die root-directory festgelegt, in der nach den index-Files gesucht wird. Alle URL-Anfragen mit `192.168.178.93/` verweisen auf die `home.html` Datei im Ordner `/var/www/192.168.178.93`. Beginnt die URL allerdings mit `192.168.178.93/sensordaten`, dann soll das `index.html` File im Ordner `/var/www/192.168.178.93/sensordaten` verwendet werden. Der Zusatz `try_files $uri $uri/ /sensordaten/index.html` verweist auch alle anderen Anfragen, die nach `sensordaten/` noch weitere URL-Segmente aufweisen, auf das gleich `index.html` File (siehe Code 1).

```
1 server {
2     listen 80;
3     listen [::]:80;
4
5     root /var/www/192.168.178.93;
6
7     server_name sensordaten;
8
9     location /sensordaten {
10         alias /var/www/192.168.178.93/sensordaten;
11         index index.html;
12         try_files $uri $uri/ /sensordaten/index.html;
13     }
14
15
16     location / {
17         index home.html;
18     }
19 }
```

Code 1: nginx Configuration File

5. `sudo nginx -t` testet das Configuration File
6. `sudo unlink /etc/nginx/sites-enabled/default` trennt die Verbindung zum default Configuration File
7. `sudo ln -s /etc/nginx/sites-available/192.168.178.93 /etc/nginx/sites-enabled/` setzt das neue Configuration File als default ein
8. `sudo nginx -t` testet das Configuration File
9. `sudo systemctl restart nginx` startet nginx neu um die Änderungen zu übernehmen
10. `sudo mkdir /var/www/<ip>/sensordaten` in diesem neuen Ordner wird die React Application installiert

Im Ordner `/var/www/<ip>` kann nun das HTML-File `home.html` mit dem Befehl `touch home.html` erstellt werden. Mit `sudo nano home.html` kann die Datei geöffnet und folgender Code abgespeichert werden (siehe Code 2):

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Welcome to nginx!</title>
5 <style>
6     body {
7         width: 35em;
8         margin: 0 auto;
9         font-family: Tahoma, Verdana, Arial, sans-serif;
10    }
11 </style>
12 </head>
13 <body>
14 <h1>Welcome to nginx!</h1>
15 <p>If you see this page, the nginx web server is successfully installed and
16 working.</p>
17
18
19 <p>Continue with:</p>
20
21 <a href="http://192.168.178.93:8086">InfluxDB</a><br/>
22 <a href="http://192.168.178.93:8123">Home Assistant</a><br/>
23 <a href="http://192.168.178.93/sensordaten">Sensordaten</a>
24
25 </body>
26 </html>
```

Code 2: Nginx home.html

Die `<a>` Tags verlinken zu der URL von der Home Assistant Instanz mit dem Port `:8123` und InfluxDB mit dem Port `:8086`. Die Seite mit den Sensordaten wird unter dem Standard Port `:80` mit dem Anhängsel `/sensordaten` zu finden sein (siehe Abbildung 12).

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working.

Continue with:

[InfluxDB](#)
[Home Assistant](#)
[Sensordaten](#)

Abbildung 12: Nginx Homepage

3.8 InfluxDB Installation

Bei der Installation von InfluxDB unter Linux ist mit folgenden Befehlen möglich[18]:

1. `wget https://dl.influxdata.com/influxdb/releases/influxdb2-2.3.0-arm64.deb`
2. `sudo dpkg -i influxdb2-2.3.0-arm64.deb`
3. `sudo service influxdb start`
4. nach einem System Neustart `sudo service influxdb status`

InfluxDB kann jetzt über `<ip>:8086` eingerichtet werden (siehe Abbildung 13 - 15).

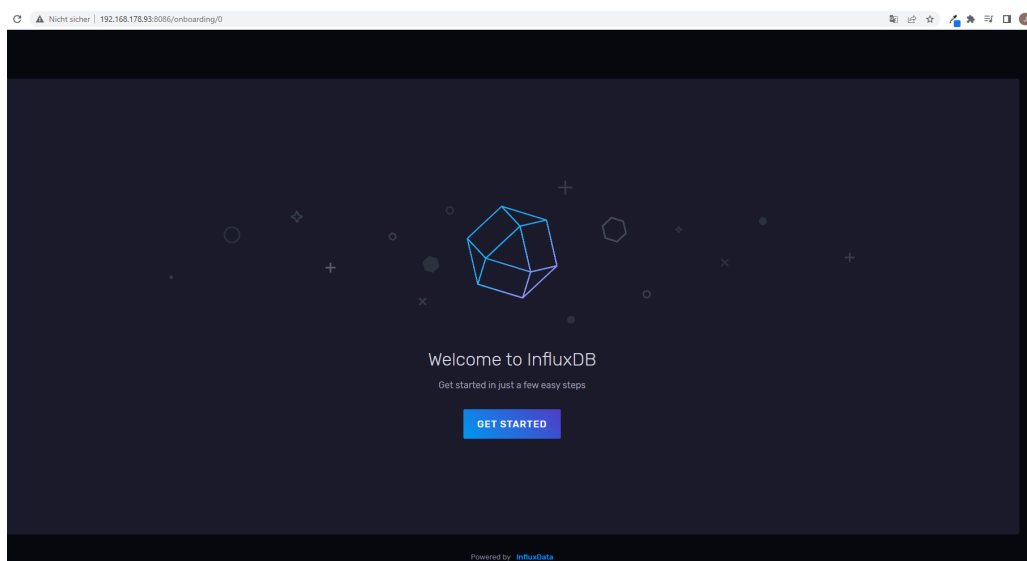
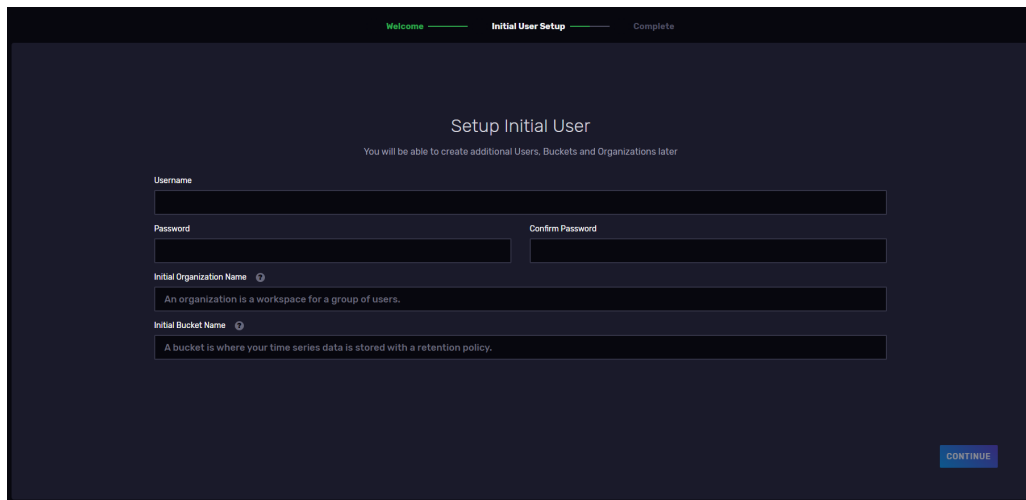
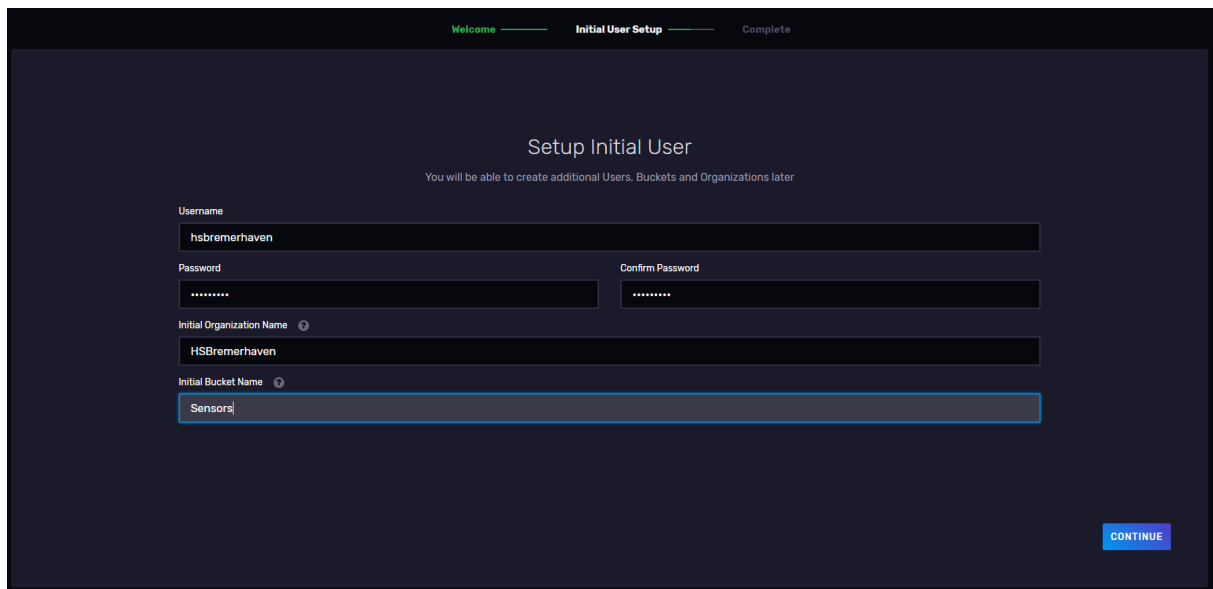


Abbildung 13: InfluxDB Einrichtung



The screenshot shows the 'Setup Initial User' form in the InfluxDB web interface. At the top, there is a progress bar with three steps: 'Welcome' (completed), 'Initial User Setup' (current step), and 'Complete'. The form title is 'Setup Initial User' with a subtitle 'You will be able to create additional Users, Buckets and Organizations later'. The form contains five input fields: 'Username', 'Password', 'Confirm Password', 'Initial Organization Name', and 'Initial Bucket Name'. Each field has a small help icon to its right. Below the 'Initial Organization Name' field, there is a text box explaining: 'An organization is a workspace for a group of users.' Below the 'Initial Bucket Name' field, there is a text box explaining: 'A bucket is where your time series data is stored with a retention policy.' A blue 'CONTINUE' button is located at the bottom right of the form.

Abbildung 14: InfluxDB User Registrieren



This screenshot shows the same 'Setup Initial User' form as in the previous image, but with the fields filled out. The 'Username' field contains 'hsbremerhaven'. The 'Password' and 'Confirm Password' fields contain masked text (dots). The 'Initial Organization Name' field contains 'HSBremerhaven'. The 'Initial Bucket Name' field contains 'Sensors'. The 'CONTINUE' button remains at the bottom right.

Abbildung 15: InfluxDB User Registrieren ausgefüllt

Unter Settings lässt sich die Retention-Time des Buckets einstellen (siehe Abbildung 16 - 18). Die Retention-Time legt den Zeitraum fest, wie lange Daten in der Datenbank gespeichert werden sollen, bevor diese automatisch gelöscht werden.

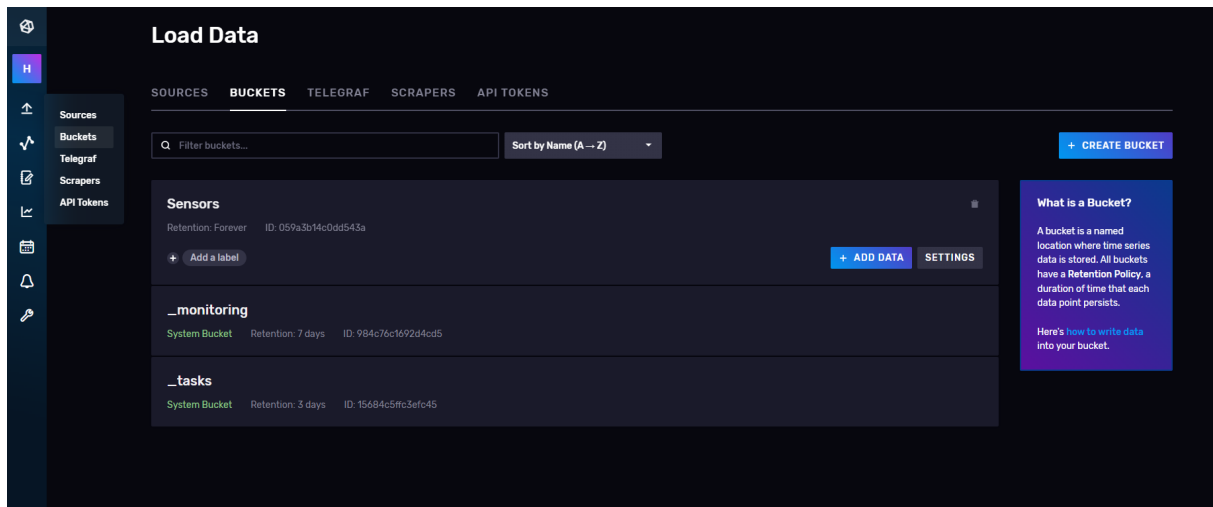


Abbildung 16: InfluxDB Bucket Retention Zeit verändern

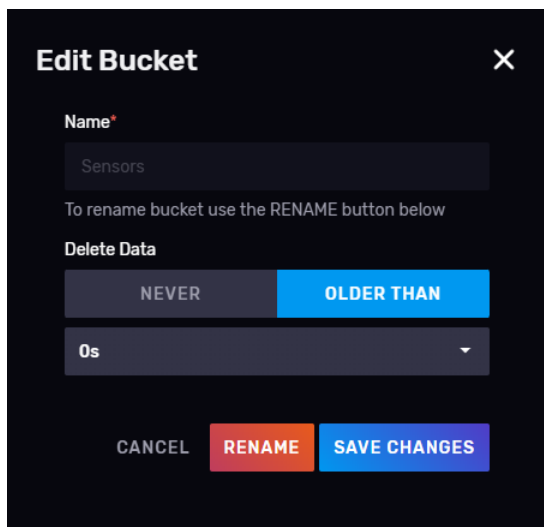


Abbildung 17: Edit Bucket

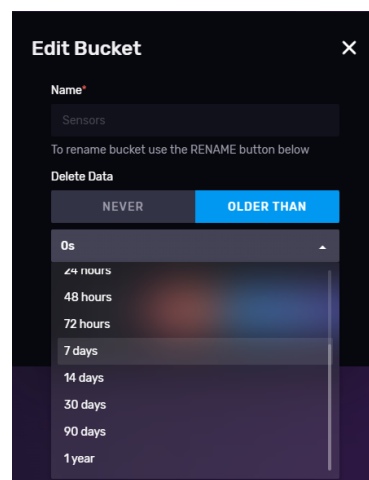


Abbildung 18: Bucket Retention Time Optionen

Für die das spätere Anfragen der Daten wird ein Read-Only API Token benötigt, dieser kann unter **Load Data -> API TOKENS** erstellt werden (siehe Abbildung 19 und 20). Der automatisch generierte Token mit vollen Schreib- und Leserechten wird für das Übertragen der Daten aus der Home Assistant Instanz benötigt.

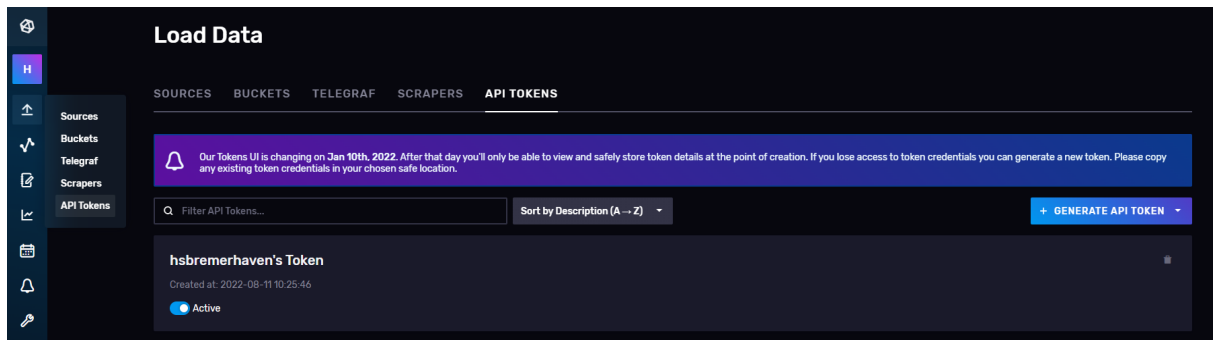


Abbildung 19: InfluxDB API Token

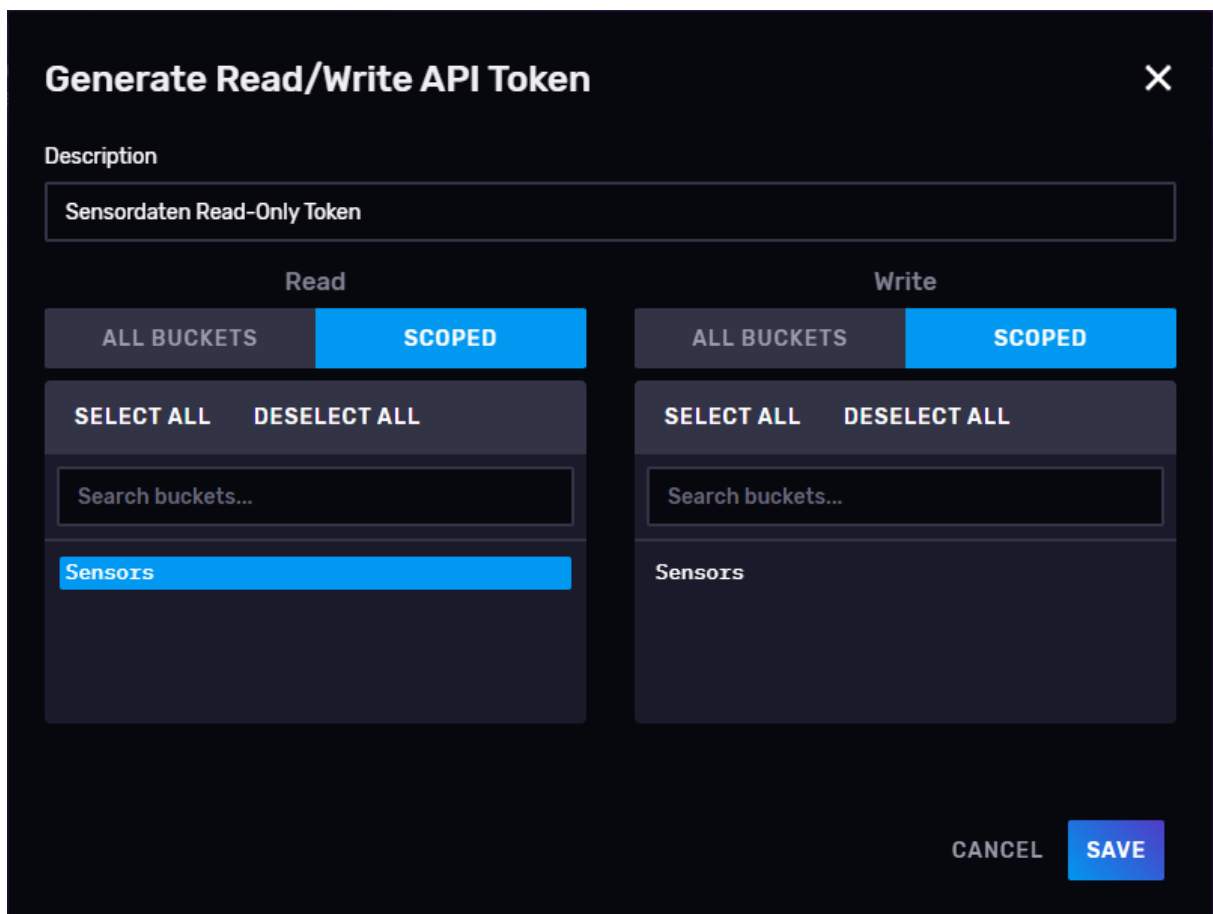


Abbildung 20: InfluxDB neuen Ready Only Token erstellen

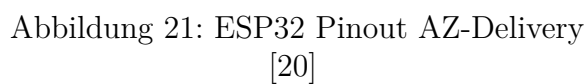
Nachdem alle vorher aufgeführten Installationsschritte abgeschlossen wurden, sind das Betriebssystem auf dem Raspberry PI, sowie Home Assistant, InfluxDB mit Add-Ons und nginx erfolgreich eingerichtet und es kann mit dem eigentlichen Erfassen und Speichern der Sensormesswerte, sowie der Konfiguration der Sensoren begonnen werden.

4 Sensormesswerte

Bevor die Software für das Auslesen der Sensordaten mit Hilfe von ESPHome auf die ESP32-Microcontroller überspielt werden kann, muss zuerst ermittelt werden, welches Pin-Layout das verwendete ESP32-Modul nutzt und welche Pins für den jeweiligen Sensortyp benötigt werden. Die Schaltbilder für die einzelnen Sensoren wurden mit der Software von **fritzing** erstellt[19].

4.1 ESP32

Das ESP32-Board der Marke AZ-Delivery nutzt ein 38-Pin-Layout (siehe Abbildung 21). Gleiche Modelle von unterschiedlichen Herstellern haben häufig ähnliche, aber nicht exakt gleiche Layouts. Deswegen sollte vor der Verkabelung immer das Datenblatt des Herstellers für die Zuweisung der Pins zu Rate gezogen werden. Die Pins werden auch häufig mit GPIO, IO oder G, gefolgt von der Pin-Nummer, bezeichnet. Das Layout des ESP32 in fritzing entspricht dem gleichen Layout des ESP32 von AZ-Delivery, die Pin-Bezeichnungen sind allerdings leicht unterschiedlich.



Der DHT22 Sensor misst sowohl die Temperatur, als auch die Luftfeuchtigkeit. Für die Spannungsversorgung werden 3 - 6V DC benötigt. Die Datenübertragung erfolgt über einen

digitalen Pin, dafür eignen sich die GPIO's G4 und G13 - G33[21]. Außerdem ist ein 10k Ohm pull-up Widerstand zwischen dem Datenpin und der Spannungsversorgung notwendig [22]. Die Verkabelung auf einem Breadboard und der Schaltplan sind in Abbildung 22 und 23 zu sehen.

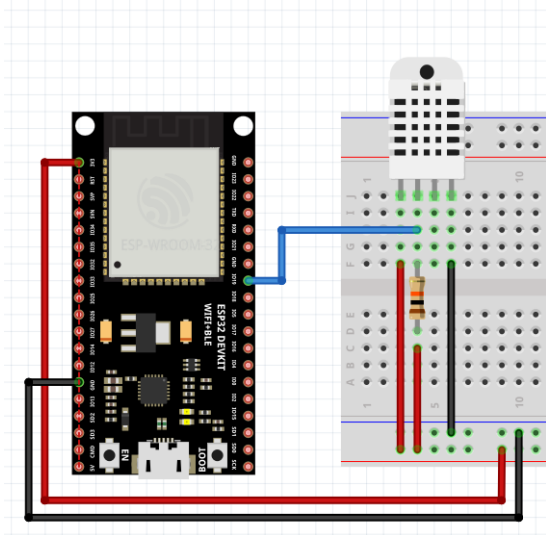


Abbildung 22: ESP32 mit DHT22 Steckplatine

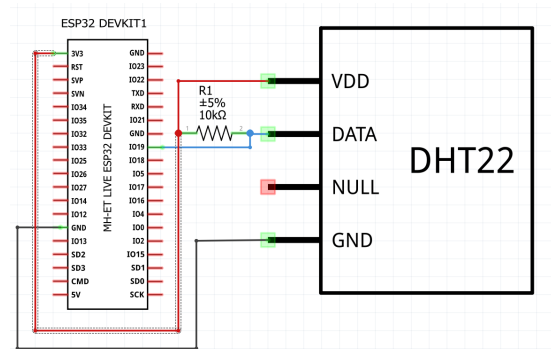


Abbildung 23: ESP32 mit DHT22 Schaltplan

4.3 SDS011 - Feinstaubsensor

Der SDS011 ist ein Feinstaubsensor, der Particular Matter (PM) Werte für PM2.5 und PM10 erfasst. Dies bezeichnet Feinstaubpartikel mit einem aerodynamischen Durchmesser kleiner als 10, respektive 2.5 Mikrometer [23]. Für die Kommunikation mit dem SDS011 wird das Universal Asynchronous Receiver Transmitter (UART)-Protokoll benutzt. Dafür kommen die Pins mit der Bezeichnung TXD/GPIO1/GPIO23 und RXD/GPIO3/GPIO22 oder die Pins GPIO17 und GPIO16 in Frage. Die Pins sind im Datenblatt meist extra mit UART gekennzeichnet. Die Spannungsversorgung erfolgt über den 5V DC Pin. Es ist anzumerken, dass der TXD-Pin des Sensors mit dem RXD-Pin des ESP verbunden werden muss und umgekehrt (siehe Abbildung 24 und 25).

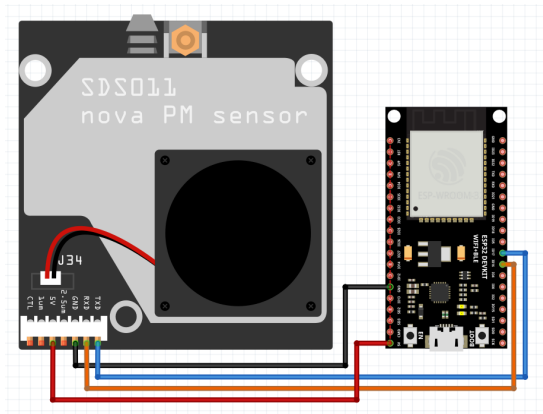


Abbildung 24: ESP32 mit SDS011

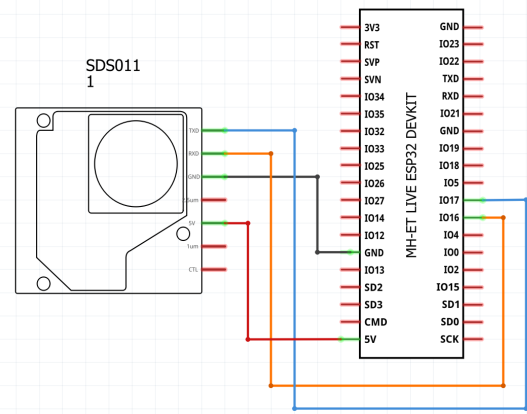


Abbildung 25: ESP32 mit SDS011 Schaltplan

4.4 MH-Z19 - Kohlenstoffdioxid-Sensor

Der MH-Z19 misst die Kohlenstoffdioxidkonzentration in der Luft und kommuniziert ebenfalls über die UART-Pins. Um diesen Sensor mit dem Feinstaubsensor zu kombinieren, werden dieses Mal die anderen, noch unbelegten UART-Pins verwendet (siehe Abbildung 26 und 27). Die Spannungsversorgung erfolgt mit 5V DC.

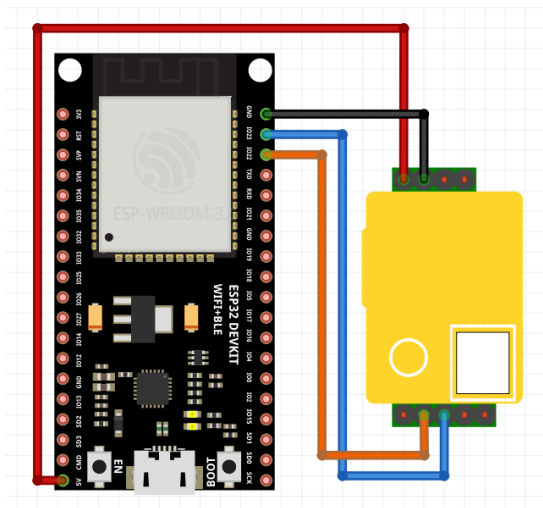


Abbildung 26: ESP32 mit MH-Z19

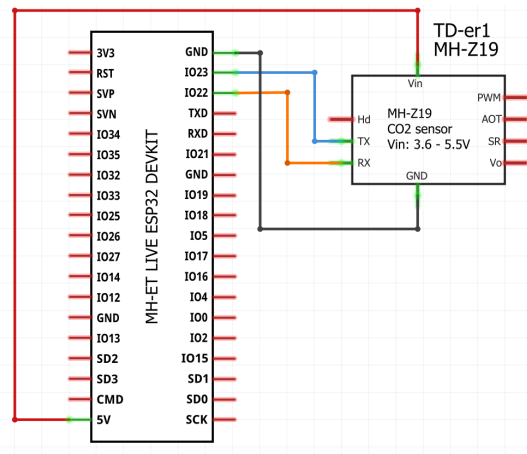


Abbildung 27: ESP32 mit MH-Z19 Schaltplan

4.5 BME280 - Temperatur-, Luftfeuchtigkeit- und Luftdrucksensor

Der BME280 Sensor ist eine kostengünstigere Alternative zum DHT22 und misst zudem noch den Luftdruck. Die Kommunikation erfolgt über das I2C oder SPI Kommunikationsprotokoll. Die Standard Pins für I2C sind GPIO22 und GPIO21 und die Spannungsversorgung läuft über 3.3V DC (siehe Abbildung 28 und 29) [24].

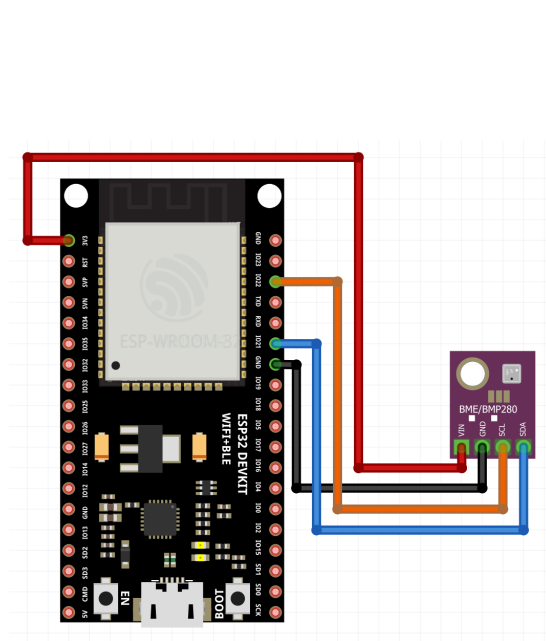


Abbildung 28: ESP32 mit BME280

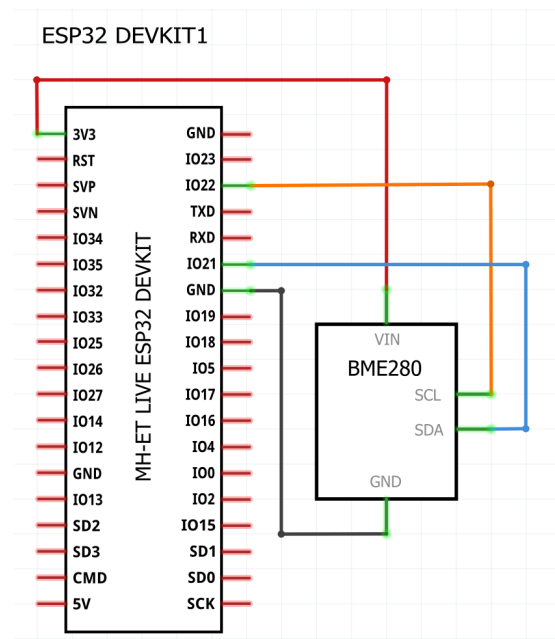


Abbildung 29: ESP32 mit BME280 Schaltplan

4.6 Verschaltung aller Sensoren

In diesem Abschnitt wird noch einmal eine vollständige Verschaltung von einem DHT22, MH-Z19 und dem SDS011 Sensor an einem ESP32-Mikrocontroller gezeigt (siehe Abbildung 30 und 31). Für die Verwendung mit einem BME280 anstelle des DHT22 Sensors, müssten softwareseitig andere I2C Pins festgelegt werden, da der default GPIO22 schon als UART Pin in Benutzung ist. ESPHome unterstützt diese Neuzuteilung.

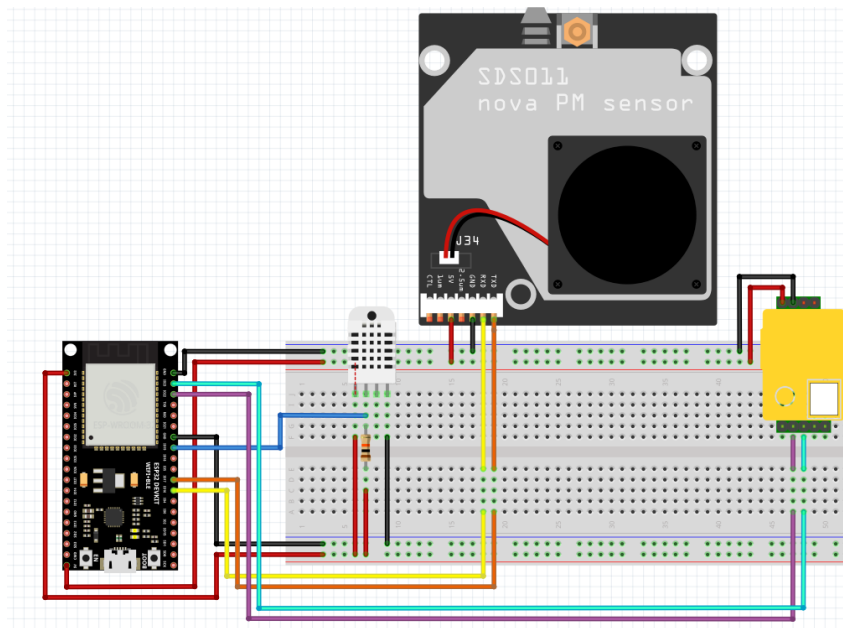


Abbildung 30: Mögliche Verschaltung von DHT22, MH-Z19 und SDS011 an einem ESP32

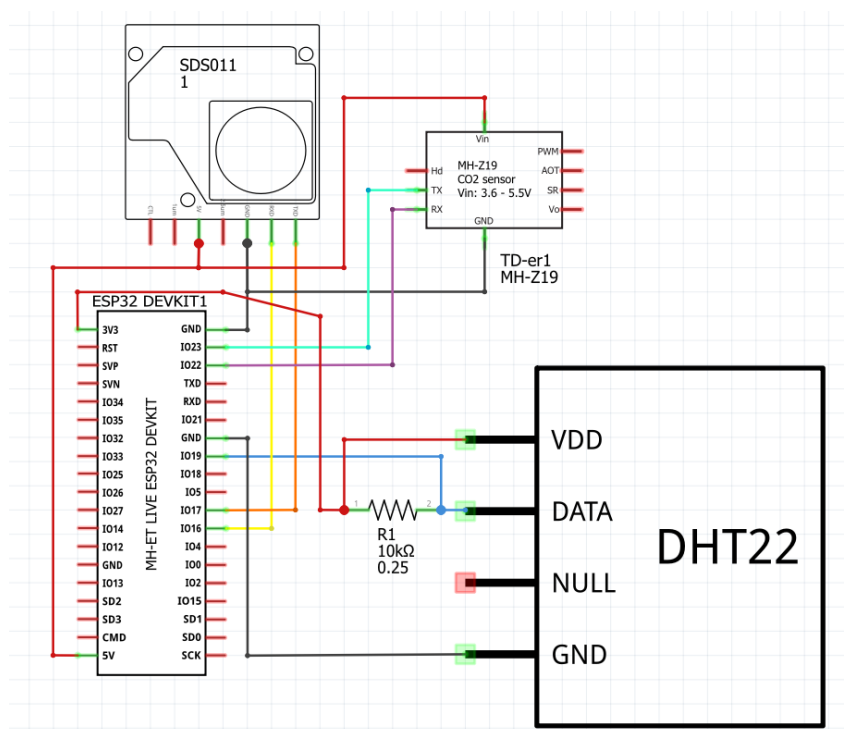


Abbildung 31: DHT22, MH-Z19 und SDS011 an einem ESP32 - Schaltplan

4.7 Konfiguration in ESPHome

ESPHome unterstützt mittlerweile eine Vielzahl an gängigen Smart Home Devices und wird durch regelmäßige Updates erweitert. Auf der Homepage sind die Geräte nach Kategorien eingeteilt zu finden, mit passenden Beispielen für die Einrichtung des Configuration Files in der ESPHome Home Assistant Erweiterung[4].

4.7.1 Neues Gerät hinzufügen

Als erstes sollte das zu konfigurierende ESP32-Modul über ein Mikro-USB Kabel mit dem Computer verbunden sein, über den die Home Assistant Seite aufgerufen wurde. In der Home Assistant Software unter dem Reiter ESPHome lässt sich mit dem Button in der Mitte **NEW DEVICE** (falls noch kein Gerät hinzugefügt wurde - siehe Abbildung 32) oder ansonsten rechts unten am Bildschirmrand, ein neues Gerät hinzufügen.

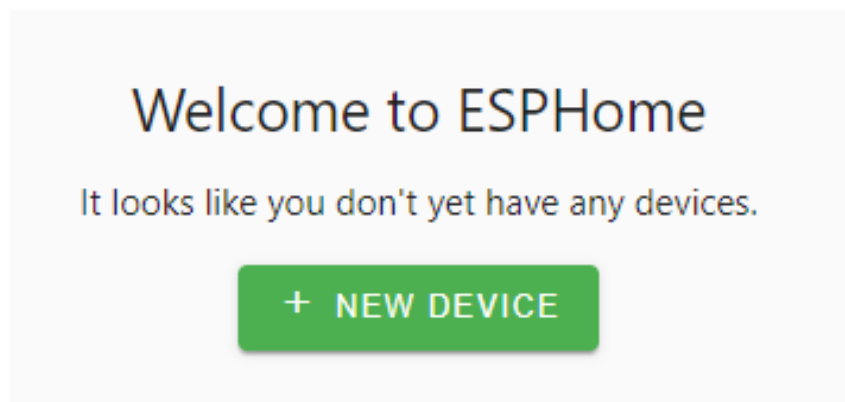


Abbildung 32: ESPHome neues Gerät hinzufügen

Mit **Continue** beginnt die Erstellung und es kann ein Name für das Gerät festgelegt und die WiFi-Login Daten angegeben werden. Diese werden unter **secrets** gespeichert und müssen nach erstmaliger Eingabe nicht erneut angegeben werden. Danach ist die Version des verwendeten ESP Boards auszuwählen (siehe Abbildung 33).

Anschließend kann mit **skip** die Installation des Configuration Files erstmal übersprungen werden. Das Gerät ist nun erstellt und wird unter ESPHome angezeigt (siehe Abbildung 34).

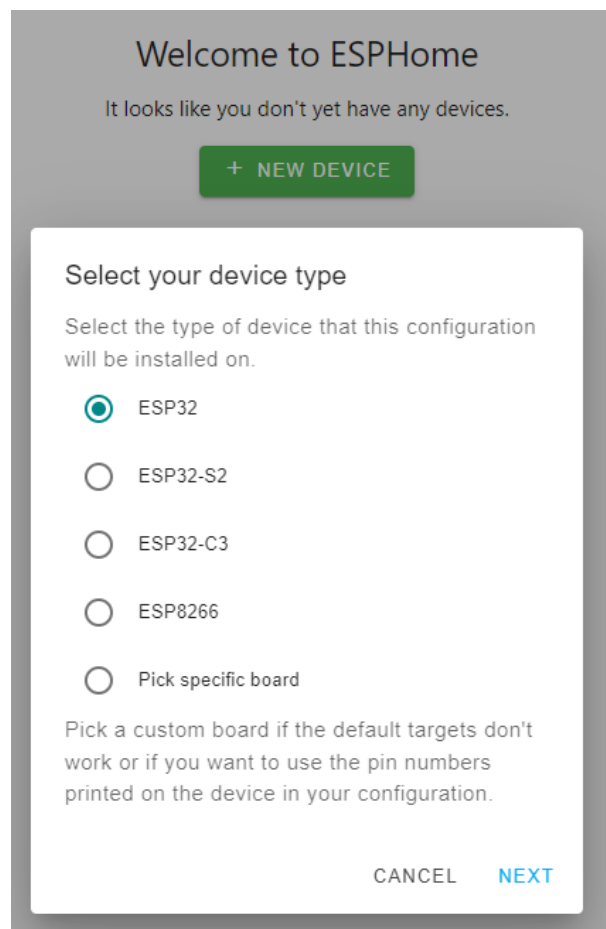


Abbildung 33: ESP Version Auswahl

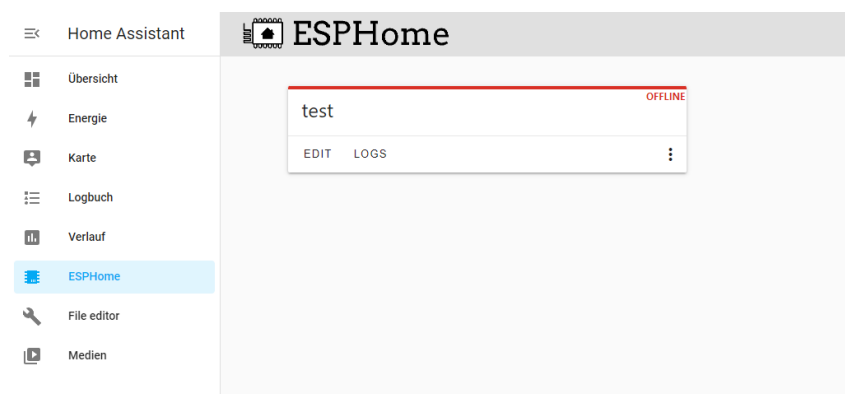


Abbildung 34: Geräte in ESPHome

Mit dem Button **Edit** lässt sich das Configuration File für das jeweilige Gerät öffnen und anschließend auch installieren.

4.7.2 Board, Wifi und Gerätenamen festlegen

In YAML lassen sich direkt keine Variablen definieren, deswegen kann sich mit `substitutions:` beholfen werden. Unter `substitutions` kann z.B. der `devicename` definiert werden, da dieser später für jeden Sensor benötigt wird und so vermieden wird, dass bei Umbenennung der Name an mehreren Stellen geändert werden muss. Der Geräte name ist deswegen so entscheidend, weil aus diesem die Entitybezeichnung abgeleitet wird und für alle Sensoren des gleichen ESP32 identisch sein muss, damit diese später auch wieder eindeutig zugeordnet werden können. Entities sind keinem Gerät zugeordnet, sondern für jeden Messwert wird eine eigene Entity erstellt.

In YAML ist das Einrücken des Codes mit der `Tab-Taste` notwendig, damit dieser richtig kompiliert werden kann. Unter `esphome:` kann separat der angezeigte Gerätenamen in ESPHome definiert werden (allerdings nur in Kleinbuchstaben).

Mit `esp32/esp8266` kann das genaue Modell des verwendeten Mikrocontrollers festgelegt werden. Die W-Lan Daten werden mit `wifi:` über das Secrets File eingebunden (siehe Code 3)[25].

```
1 substitutions:
2   devicename: Wohnzimmer
3   lower_devicename: wohnzimmer
4
5
6 esphome:
7   name: $lower_devicename
8
9 esp32:
10  board: esp32dev
11  framework:
12    type: arduino
13
14 # Enable logging
15 logger:
16
17 # Enable Home Assistant API
18 api:
19
20 ota:
21  password: "84cbdc6ac348457886b1a2233072a10996"
22
23 wifi:
24  ssid: !secret wifi_ssid
25  password: !secret wifi_password
26
27 # Enable fallback hotspot (captive portal) in case wifi connection fails
28 ap:
29  ssid: "Esp32-Dht22-Sds011-1"
30  password: "Ck9ViJKSFBHwXm"
31
```

```
32 captive_portal:
```

Code 3: YAML Configuration File

4.7.3 DHT22 Konfiguration

Alle Sensoren werden unter dem Punkt `sensor:` zusammengefasst. Bei mehreren Sensoren an einem Gerät wird durch das vorangehende `-` eine Liste von Sensoren erwartet. Unter `platform: <sensortyp>` wird festgelegt, welcher Sensor verwendet wird und auch welche Messwerte zulässig sind. Für alle Sensoren kann ein eigenständiges `update_interval` festgelegt werden. [26]

```
1 sensor:
2   - platform: dht
3     pin: GPIO19
4     temperature:
5       name: Temperatur ${devicename}
6     humidity:
7       name: Luftfeuchtigkeit ${devicename}
8     update_interval: 5min
```

Code 4: YAML Configuration DHT22

4.7.4 SDS011 Konfiguration

Unter `uart:` können ein oder mehrere UART-Binary Unit System (BUS)ES definiert werden, dafür müssen die jeweiligen GPIO Pins dem `rx_pin` bzw. `tx_pin` zugewiesen werden. Außerdem muss die `baud_rate` auf 9600 festgelegt werden. Bei mehreren Sensoren die ein UART-BUS benutzen, kann zusätzlich eine `uart_id` festgelegt werden, die dann einem Sensor zugewiesen werden kann (siehe vollständiges Codebeispiel - Code 8)[27]. Aus dem `name` wird automatische die Entity Bezeichnung abgeleitet. Aus `name: PM10 Wohnzimmer` würde beispielsweise `sensor.pm10_wohnzimmer`, dabei werden alle Buchstaben zu Lowercase umgewandelt und Leerzeichen durch einen Unterstrich ersetzt.

```
1 uart:
2   rx_pin: GPIO16
3   tx_pin: GPIO17
4   baud_rate: 9600
5   debug:
6
7
8 sensor:
9   - platform: sds011
10     pm_2_5:
11       name: PM25 ${devicename}
12     pm_10_0:
```

```
13     name: PM10 ${devicename}  
14     update_interval: 5min
```

Code 5: YAML Configuration SDS011

4.7.5 BME280 Konfiguration

Der BME280 nutzt das I²C-BUS für die Datenübertragung. Ähnlich wie beim UART-BUS werden mit `sda` und `scl` die verwendeten Pins definiert, bei mehreren BUSES kann wieder eine ID hinzugefügt werden. Außerdem muss die I²C Adresse des Sensors angegeben werden, diese ist standardmäßig `0x77`, kann aber auch `0x76` sein (siehe Code 6) [28].

```
1 i2c:  
2   sda: 21  
3   scl: 22  
4   scan: true  
5  
6 sensor:  
7   - platform: bme280  
8     temperature:  
9       name: Temperatur ${devicename}  
10      oversampling: 16x  
11      pressure:  
12        name: Luftdruck ${devicename}  
13      humidity:  
14        name: Luftfeuchtigkeit ${devicename}  
15      address: 0x76  
16      update_interval: 5min
```

Code 6: YAML Configuration BME280

4.7.6 MH-Z19 Konfiguration

Der MH-Z19 Sensor benutzt ebenfalls das UART-BUS und kann neben der Kohlenstoffdioxid-Konzentration auch die Temperatur ausgeben. In Kombination mit einem BME280 oder DHT22 kann diese jedoch auskommentiert werden (siehe Code 7) [29].

```
1 uart:  
2   - rx_pin: GPIO23  
3     tx_pin: GPIO22  
4     baud_rate: 9600  
5     id: uart2  
6  
7 sensor:  
8   - platform: mhz19  
9     co2:  
10      name: CO2 ${devicename}  
11      #temperature:  
12        #name: "MH-Z19 Temperature"
```

```
13   update_interval: 5min
14   automatic_baseline_calibration: false
15   uart_id: uart2
```

Code 7: YAML Configuration MH-Z19

4.7.7 Vollständige YAML Konfiguration

Das folgende Codebeispiel zeigt das vollständige Configuration-File für einen DHT22, SDS011 und MH-Z19 Sensor einem ESP32 mit zwei UART-BUSES und einem I²C-BUS (siehe Code 8) .

```
1 substitutions:
2   devicename: Arbeitszimmer HausC
3   lower_devicename: arbeitszimmer_hausc
4
5
6 esphome:
7   name: $lower_devicename
8
9 esp32:
10  board: esp32dev
11  framework:
12    type: arduino
13
14 # Enable logging
15 logger:
16
17 # Enable Home Assistant API
18 api:
19
20 ota:
21
22
23 wifi:
24   ssid: !secret wifi_ssid
25   password: !secret wifi_password
26
27 # Enable fallback hotspot (captive portal) in case wifi connection fails
28 ap:
29   ssid: "Esphome-Web-C88788"
30   password: "cPbSbuusdDewA8zTfv"
31
32 captive_portal:
33
34 uart:
35   - rx_pin: GPIO17
36     tx_pin: GPIO16
37     baud_rate: 9600
38     id: uart1
39   - rx_pin: GPIO23
40     tx_pin: GPIO22
```

```

41     baud_rate: 9600
42     id: uart2
43
44   sensor:
45     - platform: dht
46       pin: GPIO19
47       temperature:
48         name: Temperatur ${devicename}
49       humidity:
50         name: Luftfeuchtigkeit ${devicename}
51       update_interval: 5min
52     - platform: mhcz19
53       co2:
54         name: CO2 ${devicename}
55       #temperature:
56         #name: "MH-Z19 Temperature"
57       update_interval: 5min
58       automatic_baseline_calibration: false
59       uart_id: uart2
60     - platform: sds011
61       pm_2_5:
62         name: PM25 ${devicename}
63       pm_10_0:
64         name: PM10 ${devicename}
65       update_interval: 5min
66       uart_id: uart1

```

Code 8: YAML Configuration Kombination

4.8 Home Assistant InfluxDB Integration

Über den `File editor` kann über das Icon mit dem Ordner `Browse Filesystem` die Datei unter `/config/configuration.yaml` geöffnet werden. Unter `influxdb:` können nun die Daten zur InfluxDB Instanz eingefügt werden, der verwendete Token muss zwingend Schreibrechte aufweisen. Unter `entity_globs:` können die Bezeichnungen der Entities festgelegt werden, deren Daten übertragen werden sollen. Mit `- sensor.temperatur_*` werden alle Entities vom Typ Sensor mit der Anfangsbezeichnung `temperatur_` ausgewählt. Das `*` steht als Platzhalter für alle darauffolgenden Zeichen (siehe Code 9).

```

1  # Loads default set of integrations. Do not remove.
2  default_config:
3
4  # influxDB integration
5  influxdb:
6    api_version: 2
7    ssl: false
8    host: 192.168.178.75
9    port: 8086
10   token: 296ISoBvhix0Ujsdfuijh324uisdfjoXzOpMbZFEggiH8-eZbWJAPifob57FCKU9-xPw==

```

```
11  organization: HSBremerhaven
12  bucket: Sensors
13  # tags:
14  #   source: HA
15  # tags_attributes:
16  #   - friendly_name
17  default_measurement: units
18  include:
19    domains:
20      - sensor
21  entity_globs:
22    - sensor.temperatur_*
23    - sensor.luftfeuchtigkeit_*
24    - sensor.co2_*
25    - sensor.pm25_*
26    - sensor.pm10_*
```

Code 9: Home Assistant Configuration File

5 JavaScript Grundlagen

Dieses Kapitel soll eine Einführung und Überblick über den für diese Arbeit relevanten Funktionsumfang von JavaScript liefern und die Lesbarkeit, sowie das Verständnis des Quellcodes erleichtern. Außerdem werden die Grundlegenden Konzepte von ReactJS und ReactRouter erklärt. Die Erläuterung erfolgt in einem kurzen beschreibenden Text zu einem passenden, eingebetteten Code-Beispiel.

5.1 Kommentare

Bei Kommentaren lässt sich zwischen Single-Line und Multi-Line Kommentaren unterscheiden. Ersteres wird durch `//` erreicht und Kommentare über mehrere Zeilen werden mit `/* */` eingefasst.

```
1 // Kommentar - eine Zeile
2
3 /* Kommentar -
4 mehrere
5 Zeilen
6 */
```

Code 10: Kommentare

5.2 Variablen

In JavaScript gibt es drei Möglichkeiten, Variablen zu deklarieren: `var`, `let` und `const`. `var` findet mittlerweile in modernem JavaScript kaum noch Anwendung.

5.3 Scope

Variablen unterliegen einem Scope (zu Deutsch: Anwendungsbereich), in dem diese Gültigkeit besitzen. Es gibt unter anderem Block- und Functionscope. Eine Variable, die innerhalb eines Blockes (z.B. einer Schleife oder einem If-Statement) oder innerhalb einer Funktion deklariert wurde, ist nur innerhalb dieser definiert und kann nicht von außerhalb verwendet werden. Ein Block kann außerdem einfach mit geschweiften Klammern `{}` dargestellt werden. Außerdem können Variablen innerhalb des gleichen Scopes nur einmal deklariert werden. Variablen, die durch `let` deklariert wurden, lassen sich hierbei aber trotzdem noch neue Werte zuweisen. Für `const` gilt dieses für primitive Werte nicht und für Objekte nur bedingt. Eine genaue Erläuterung mit Beispiel erfolgt im weiteren Verlauf dieses Kapitels.

Wird die gleiche Variable eines höher liegenden Scopes in einem tieferliegenden Scope neu deklariert, wird diese neu angelegt und es kann erst nach der Deklaration auf diese zugegriffen werden.

```
1 let a = 1;
2 const b = 1;
3 {
4   console.log({ a, b }); // Theoretisch: { a: 1, b: 1 } , aber Error: Variable kann nicht vor
                           // der Initialisierung verwendet werden
5
6   let a = 2; // die Variable a kann innerhalb eines neuen Blockes auch neu deklariert
               // werden
7   let a = 3; // Error: Variable wurde schon deklariert
8   a = 3;     // neuer Wert fuer a
9
10  const b = 2;
11  const b = 3; // Error: Variable wurde schon deklariert
12  b = 3;      // Error: b ist konstant
13 }
```

Code 11: Scope

Bei verschachtelten Blöcken, z.B. bei einer Schleife in einer Funktion, kann eine Variable außerhalb der Schleife trotzdem in der Schleife verwendet werden, da der Scope mit vererbt wird. Andersherum ist dies nicht möglich.

```
1
2 function scopeTest() {
3   let number = 1;
4
5   for(let i = 1; i < 2; i++) {
6     let newNumber = 2;
7     number += 1;
8   }
9
10  console.log({number}); // {number: 2}
11  console.log({newNumber}); // Error: newNumber is not defined
12 }
```

Code 12: Function and Block Scope

Mit `console.log()` lassen sich Werte in der Konsole ausgeben. Ein nützlicher Trick ist es, die auszugebende Variable in geschweiften Klammern unterzubringen `{ variable }`. Dadurch wird der Wert als JavaScript Object geloggt und ein Object besteht immer aus Key-Value-Paaren. Dadurch lassen sich auch mehrere Variablen in einem `console.log()` Statement unterbringen und einfach voneinander unterscheiden. Würde man auf das Beispiel oben bezogen lediglich `console.log(number)` eingeben, wäre das Ergebnis nur `2`.

5.4 Datentypen

Bei JavaScript handelt es sich um eine schwach typisierte Programmiersprache, d.h. es lassen sich sowohl Variablen von einem in einen anderen Datentyp umwandeln, als auch Operationen zwischen zwei unterschiedlichen Datentypen durchführen. Folgende primitive Datentypen (die nur einen einfachen Wert besitzen) gibt es:

- boolean - `true` oder `false`
- null - mit Absicht fehlender Wert
- undefined - Variable wurde noch kein Wert zugewiesen
- number - Zahl `1`, `1.5`, `100000`, `100_000_000`
- string - Zeichenabfolge `"test"`, `'test'`, ``test``
- symbol - einzigartiger Identifier, zwei Symbole mit dem gleichen String Wert sind nicht identisch

Alle weiteren Werte die nicht primitiv sind, werden unter dem Datentyp object zusammengefasst. Dazu gehören Arrays, Funktionen und das JavaScript object an sich[30].

5.4.1 Arrays

Arrays und Objects in JavaScript werden nicht nach Wert gespeichert, sondern mit einer Referenz. Das ist insofern relevant, weil obwohl sich die Werte in einem Array oder einem Object ändern können, die Referenz zu diesem Array/Object gleich bleibt. ReactJS hingegen ist dafür ausgelegt User Interface Elemente basierend auf so genannten State- oder Property-Changes zu updaten. Intern wird ein oberflächlicher Vergleich mit dem vorherigen State durchgeführt. Oberflächlich bedeutet in diesem Fall, dass bei einem Array oder Objekt nur die Referenz verglichen wird, aber keine Betrachtung der tatsächlichen Werte innerhalb erfolgt. Deshalb ist es auch möglich, ein Array/Object mit `const` zu deklarieren und nachträglich trotzdem Änderungen vorzunehmen. Ein Array wird durch eckige Klammern gekennzeichnet `let array = []`. Die Werte eines Arrays sind über den Index beginnend bei 0 zu erreichen. Das erste Element des Arrays kann mit `array[0]` ausgelesen werden. Die Länge eines Arrays kann mit `array.length` bestimmt werden, es ist allerdings zu beachten, dass der letzte Index in einem Array immer `array.length - 1` ist.

```
1  const array = ["a"];
2  array.push(1);    // füegt 1 an das Ende des Arrays, ["a",1]
3  array[3] = 3;     // setzt den Wert mit dem Index 3 auf 3, sind dis dahin noch nicht alle
                     // Werte definiert, werden diese mit undefined aufgefüllt , ["a",1,undefined,3]
4
5  const newArray = array; // definiert ein neues Array mit der gleichen Referenz
6
7  newArray.push(4, "b"); // ["a",1,undefined,3,4,"b"]
8
9  console.log({ array, newArray });
10 /*
11 {
12   array: ["a",1,undefined,3,4,"b"],
13   newArray: ["a",1,undefined,3,4,"b"]
14 }
15 */
16 console.log(array === newArray); // true
17
18 let firstElement = array[0]; // "a"
19 let arrayLength = array.length; // 6
```

Code 13: Arrays

Da `array` und `newArray` sich die gleiche Referenz teilen, führen Veränderungen an dem einen auch zu Veränderungen an dem anderen Array.

Versucht man einem bestehenden Array mit eckigen Klammern neue Werte zuzuweisen, dann funktioniert dieses nur mit `let`, da durch die eckigen Klammern auch ein neues Array mit einer anderen Referenz erzeugt wird.

```
1  const constArray = [];
2  constArray = [1]; //Error: Assignment to constant variable
3
4  let letArray = [];
5  letArray = [1];  // funktioniert
```

Code 14: let und const mit Arrays

Möchte man eine Kopie eines Arrays mit einer neuen Referenz erstellen, kommt häufig der Spread-Operator zum Einsatz. Mit `[...array]` werden alle Elemente des ursprünglichen Arrays in ein neues Array übernommen.

```
1  const array = [1,2,3];
2
3  const newArray = [...array];
4
5  console.log({array, newArray}); // {array: [1,2,3], newArray: [1,2,3]}
6
7  console.log(array === newArray); // false
```

Code 15: Array Spread-Operator

Oft möchte man nur ganz bestimmte Werte eines Arrays, welche z.B. von einer Funktion zurückgegeben werden, nutzen. Durch Array-Destructuring können neuen Variablen die Werte des Arrays, beginnend bei dem ersten Eintrag, zugewiesen werden. Dabei gibt es, ähnlich wie beim Spread-Operator, den Rest-Operator, durch den alle übrigen Werte in einem neuen Array untergebracht werden. Dies erfolgt genauso durch das Hinzufügen von `...` vor dem Variablennamen.

```
1  const array = [1,2,3,4,5];
2  const [firstElement, secondElement, ...rest] = array;
3
4  console.log({firstElement,secondElement,rest}); // {firstElement: 1, secondElement: 2, rest:
    [3,4,5]}
```

Code 16: Array Destructuring

5.4.2 Objects

JavaScript Objects verhalten sich in Bezug auf Referenz genauso wie Arrays. Der Unterschied ist, dass ein Object keine Liste von Werten mit einem Index beginnen bei 0 darstellt, sondern eine Ansammlung von Key-Value-Paaren enthält, die über den Key des Objects zu erreichen sind. Ein Object wird außerdem durch geschweifte Klammern erzeugt `{ key: value }`. Der Key kann entweder mit der Dot-Notation oder über das String-Equivalent in eckigen Klammern angegeben werden. Letzteres ist hilfreich, falls der Key ein Leerzeichen enthält.

```
1  const airCondition = { temperature: 10, humidity: 50, "unit of measurement": "%" };
2
3  let temperature1 = airCondition.temperature;
4  let temperature2 = airCondition["temperature"];
5  let unitOfMeasurement = airCondition["unit of measurement"];
6
7  console.log({ temperature1, temperature2, unitOfMeasurement }); // { temperature1: 10,
    temperature2: 10, unitOfMeasurement: "%" }
```

Code 17: Objects

Objekte, aber auch Arrays, können mehrere Verschachtelungen aufweisen. Diese werden als nested Objects/Arrays bezeichnet. Das ist insofern sinnvoll, da z.B. jeder Messwert immer seinen eigenen Wert und eine eigene Einheit aufweist. Auch hier kann dieses Mal auf Object-Destructuring zurückgegriffen werden.

```
1  const airCondition = {
2    temperature: { value: 10, unitOfMeasurement: "C" },
3    humidity: { value: 50, unitOfMeasurement: "%" }
4  };
5
```

```
6  let temperatureValue = airCondition.temperature.value;    // 10
7
8
9  airCondition.airPressure = { value: 1012, unitOfMeasurement: "hPa" };
10
11  const { temperature, humidity, airPressure } = airCondition;
12  const {
13    value: airPressureValue,
14    unitOfMeasurement: airPressureUnit
15  } = airCondition.airPressure;
16
17  console.log({ temperature, humidity, airPressure }); // { temperature : { value: 10,
18    unitOfMeasurement: "C" }, humidity: {value: 50, unitOfMeasurement: "%" }, airPressure:
19    { value: 1012, unitOfMeasurement: "hPa" }}
20
21  console.log({ airPressureValue, airPressureUnit }); // { airPressureValue: 1012,
22    airPressureUnit: "hPa" }
```

Code 18: Verschachtelung von Objekten

5.4.3 Functions

Es gibt zwei Möglichkeiten eine Funktion zu definieren, einmal mit dem `function` Keyword, oder mit `const` als Arrow-Function.

```
1  let number = 0;
2
3  const square = (x) => {
4    return x * x;
5  };
6
7  function updateNumber() {
8    number += 1;
9  }
10
11  updateNumber();
12  let squaredNumber = square(2);
13  console.log({squaredNumber, number}); // { squaredNumber: 4, number: 1}
```

Code 19: Function

5.5 ReactJs

React verwendet Functional Components und HTML-ähnliche Syntax, um User Interface (UI)-Elemente darzustellen. Dazu wird eine Root-Node an ein bestimmtes Element (meistens ein `<div id='root'></div>` Element mit der ID `root`) in dem `index.html` angefügt, in der dann die ganze React Applikation ausgeführt wird. Um ein neues React Projekt anzulegen, gibt es den Command `npm create-react-app <name>`, mit dem ohne viel Aufwand

ein boilerplate (Rohmodell) Projekt mit den nötigsten Dateien angelegt wird. Eine Installation von NodeJS ist erforderlich, um sowohl React, als auch später andere npm Libraries nutzen zu können.

```
1 import React from "react";
2 import ReactDOM from "react-dom/client";
3
4 const App = () => {
5
6     return(
7         <div>React App</div>
8     )
9 }
10
11 const root = ReactDOM.createRoot(document.getElementById("root"));
12 root.render(
13     <App />
14 );
```

Code 20: React

5.5.1 Styling in React

Das Styling von HTML-Elementen ist in React nicht nur auf die Importierung von CSS-Files limitiert, es gibt außerdem die Möglichkeit mit Hilfe von Inline-Styles oder Styled-Components zu arbeiten.

In React lassen sich standardmäßig ein oder mehrere Cascading Style Sheets (CSS)-Files einbinden, am einfachsten ist die Importierung direkt in der Datei, in der die Root Node definiert wird. Das Hinzufügen von CSS-Klassen erfolgt über die Property `className=`.

```
1 import "../index.css";
2
3 const App = () => {
4
5     return(
6         <div className="textClass1 testClass2">React App</div>
7     )
8 }
```

Code 21: Styling CSS-File

Der Vorteil von Inline-Styles ist, dass der CSS-Code direkt im JavaScript File untergebracht ist und somit auch die Verwendung von Variablen erleichtert. Ein HTML-Element kann durch das Hinzufügen des `style={}` Attributes beeinflusst werden. Eine Besonderheit bei Inline-Styles ist, dass bei CSS-Attributen, die aus mehrere Wörtern bestehen oder durch Bindestriche getrennt sind, diese durch Camel Case ersetzt werden. Camel Case ist eine Form der Wort Bezeichnung von Variablen/Attributen, dabei wird das erste

Wort immer klein geschrieben und alle darauf folgenden Worte ohne Leerzeichen mit einem Großbuchstaben angefügt. Die CSS-Attributes müssen außerdem als Object, mit Kommata anstelle von Semikolons getrennt, an das style Attribute weitergegeben werden.

```
1 let fontSizeDiv = "2em";
2
3 <div style={{
4   color: "white",
5   backgroundColor: "blue",
6   fontSize: fontSizeDiv
7 }}>
8 Test
9 </div>
```

Code 22: Styling mit Inline-Styles

Die Verwendung von Styled-Components ermöglicht die Importierung aus anderen Files, ohne auf den Vorteil von Variabel verzichten zu müssen. Außerdem können HTML-Elementen eigene Bezeichnungen zugewiesen werden, was unter anderem die Lesbarkeit erhöhen kann. Um Styled-Components verwenden zu können, ist die Installation des zugehörigen Node-Packages notwendig.

```
1 import Container from "../styledComponents.jsx"
2
3 <Container color={"blue"}>
4   <Container.Heading>
5     Überschrift
6   </Container.Heading>
7 </Container>
8
9 // --- neue Datei mit der Bezeichnung styledComponents.jsx
10 import styled from "styled-components";
11
12 const Heading = styled.h1`
13   border: 1px solid black;
14   `
15
16 const Container = styled.div`
17   width: 100%;
18   height: 50px;
19   //falls eine color Property spezifiziert wurde, stelle dies dar, ansonsten "white" als default
20   color: ${({color}) => color ? color : "white" }
21   `
22
23 Container.Heading = Heading;
24
25 export default Container;
```

Code 23: Styling mit Styled-Components

5.5.2 React Props

Properties oder kurz Props, sind Variablen, die in eine React-Komponente weitergegeben werden können. Ändert sich der Wert einer Property, führt dies zu einem Rerendering dieser Komponente und allen weiteren Child-Komponenten, die in dieser beherbergt sind.

```
1 import Text from "../Text.jsx"
2
3 const App = () => {
4
5     return <Text content={"Beispieltext"} active={true}/>
6 }
7
8
9 % ----- Text.jsx
10 const Text = ({content, active}) => {
11     return (
12         <div style={{color: active ? "green" : "red"}}>
13             { content ? content : "kein Text vorhanden" }
14         </div>
15     )
16 }
17
18 export default Text;
```

Code 24: React Props

5.5.3 Fractions

Jede React-Komponente kann nur ein einziges HTML-Element zurückgeben, bzw. ein Parent-Element, in dem weitere Child-Element untergebracht sind. Entscheidend ist, dass im Return-Statement ein Element existieren muss, dass alle anderen umschließt. Häufig reicht dafür ein simples `<div>` Element aus, aber in manchen Fällen möchte man gar kein Element zurückgeben, da dies eventuell zu Komplikationen im Styling führt. Für diesen Fall gibt es Fractions `<> </>`. Fractions können als Wrapper verwendet werden, ohne ein HTML-Element zurückzugeben.

```
1 const App = () => {
2
3     return (
4         <>
5             <span>Überschrift</span>
6             <div>Text</div>
7         </>
8     )
9 }
```

Code 25: React Fractions

5.5.4 UseState Hook

Möchte man ein Rerendering innerhalb einer Komponente erreichen, kann man den `useState()` Hook verwenden. Der `useState`-Hook gibt einen State-Wert und eine Update-funktion in einem Array zurück. In den runden Klammern kann ein Startwert festgelegt werden, der eingesetzt wird, wenn eine Komponente das erste Mal gerendert wird.

```
1 import React , { useState } from "react";
2
3 const App = () => {
4
5   const [state, setState] = useState(1);
6
7
8
9   return (
10     <div>
11       {state}
12     </div>
13   )
14 }
```

Code 26: React `useState` Hook

5.5.5 UseEffect Hook

Der `useEffect` Hook ist vergleichbar mit einer Setup-Funktion in anderen Programmiersprachen. Diese wird ausgeführt, wenn eine Komponente das erste Mal gerendert wird. Allerdings kann die Funktion auch später erneut ausgeführt werden, wenn sich bestimmte Variablen oder Parameter ändern. Dazu hat `useEffect` ein Dependency-Array. Veränderungen an Werten in dem Dependency-Array führen zur erneuten Ausführung der Funktionen im `useEffect`-Hook. Soll `useEffect` nur einmalig ausgeführt werden, kann das Dependency-Array auch leer gelassen werden. Außerdem können in dem `return`-Statement Cleanup-Functions untergebracht werden, die ausgeführt werden, bevor eine Komponente nicht mehr gerendert wird oder `useEffect` erneut durchläuft. Das `return`-Statement kann auch weggelassen werden. Mehrere `useEffect` Funktionen sind in einer Komponente möglich.

```
1 import React , { useState, useEffect } from "react";
2
3 const App = ({number, text}) => {
4
5   const [state, setState] = useState(1);
6
7   // wird einmal am Anfang ausgeführt und jedes mal, wenn sich number ändert
8   useEffect(() =>
9
10     setState(number + 10);
```



```
11
12     return ();
13     ,[number]);
14
15 return (
16     <div>
17         {state}
18     </div>
19 )
20 }
```

Code 27: React useEffect Hook

5.5.6 Conditional Rendering

Oft soll ein Element nur dargestellt werden, wenn bestimmte Voraussetzungen erfüllt sind. Dafür lässt sich der `&&` oder Ternary-Operator verwenden. Beide Operanden können auch im regulären JavaScript für Variablen angewendet werden.

Der logische UND Operator (angewendet auf UI-Elemente), zeigt diese dann an, wenn die Bedingung erfüllt ist, also wie im Beispiel - `active = true` ist.

```
1 const App = ({active}) => {
2
3     return (
4         <div>
5             {active &&
6                 (
7                     <p> Active is true </p>
8                 )
9             }
10        </div>
11    )
12 }
```

Code 28: Conditional Rendering - `&&` Operator

Der Ternary-Operator ist eine kürzere Version eines if-else-Statements und kann sowohl bei erfüllter Bedingung, als auch unerfüllter Bedingung, ein Ergebnis anzeigen.

```
1 const App = ({active}) => {
2
3     return (
4         <div>
5             {active ?
6                 (
7                     <p> active ist true </p>
8                 ) :
9                 (
10                    <p> active ist false </p>
11                )
12            }
13        </div>
14    )
15 }
```

```
12     }  
13   </div>  
14 )  
15 }
```

Code 29: Conditional Rendering - Ternary Operator

5.5.7 Mapping von Arrays/Objects

Um Text oder Daten dynamisch aus einem Array/Object anzuzeigen, kann die `.map()` Methode auf das jeweilige Element angewendet werden. Einzelne Werte können als Props an Child-Komponenten weitergegeben werden (beispielhaft die Komponente `<Device>`). Alle Komponenten, die einer `.map()` Funktion entstammen, muss durch die property `key` ein einzigartiger Identifier zugewiesen werden, damit React die Elemente eindeutig zuordnen kann.

```
1 import Device from "../Device";  
2  
3 const App = () => {  
4  
5   const data = [{name: "arbeitszimmer"}, {name: "flur"}];  
6  
7   return (  
8     <div>  
9       {data.map((value, index) =>  
10         return <Device name={value.name} key={index} />  
11       )}  
12     </div>  
13   )  
14 }  
15 }
```

Code 30: Array.map()

```
1 import Device from "../Device";  
2  
3 const App = () => {  
4  
5   const names = { arbeitszimmer: { sensoren: ["dht", "sds011"] }, flur: { sensoren: ["bme280", "mh-z19"] } }  
6  
7   return (  
8     <>  
9       {Object.keys(names).map((key, i) => {  
10         return (  
11           <Device  
12             key={i}  
13             name={key}  
14             sensoren={names[key].sensoren}  
15           />  
16         )  
17       }  
18     )  
19   )  
20 }
```

```
16         );  
17     }  
18 </>  
19 );  
20 };  
21 export default App;
```

Code 31: Object.map()

6 Programmaufbau

In diesem Kapitel wird der Programmaufbau der React Application, sowie die Implementierung der wichtigsten Programmfunktionen erläutert.

6.1 Layout

Die Application besteht aus mehreren verschachtelten Komponenten. Auf oberster Ebene gibt es nur die Header-Komponente, in der auf der linken Seite ein Logo angezeigt wird und auf der rechten Seite ein Logout-Button, falls ein Benutzer eingeloggt ist (siehe Abbildung 35).

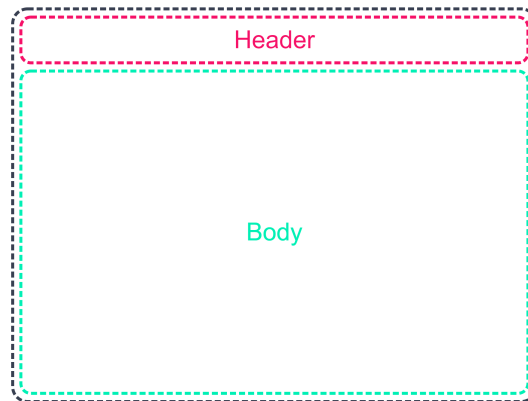


Abbildung 35: Page Layout

Der Body zeigt entweder die Login-Page, wenn noch kein Benutzer angemeldet ist, oder ein Menü zur Navigation und einen Bereich zu den Sensordaten (siehe Abbildung 36 - 37).

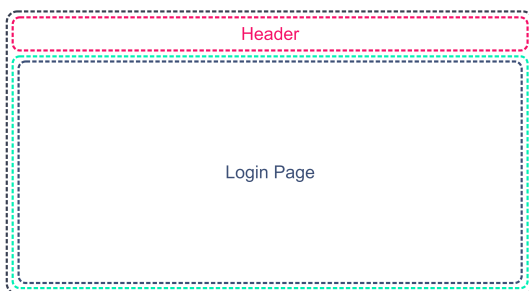


Abbildung 36: Layout Body - Login Page

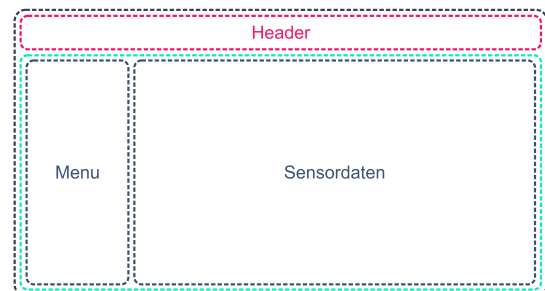


Abbildung 37: Layout Body - Authenticated

Für die Ansicht auf einem Smartphone oder einem Tablet ist die Darstellung untereinander geeigneter (siehe Abbildung 38). Die unterschiedlichen Ansichten werden über Media Queries im Mobile-First Approach erreicht.

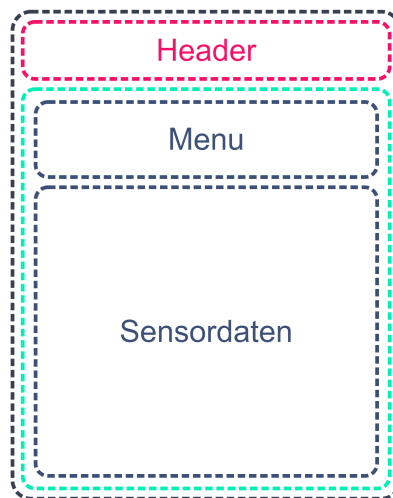


Abbildung 38: Page Layout Responsive

Über den Button `Home Assistant Login` wird der Benutzer auf die Home Assistant Instanz weitergeleitet, um sich mit den Benutzerdaten zu verifizieren (siehe Abbildung 39).

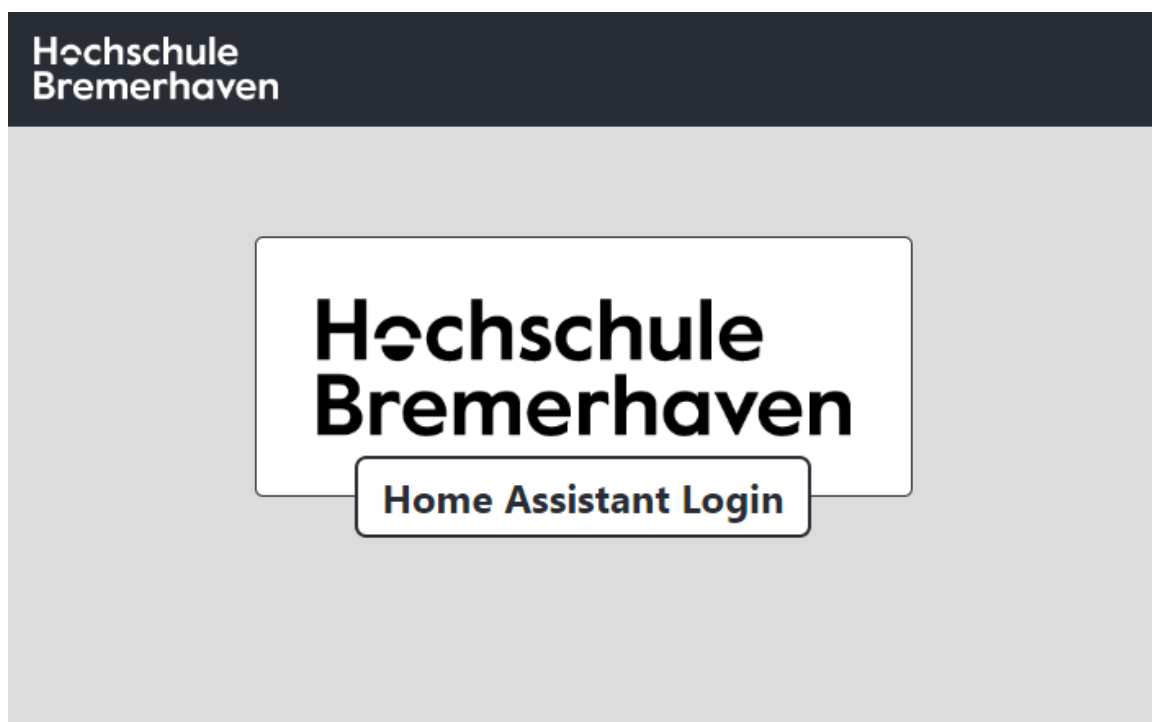


Abbildung 39: Login Page

Nach erfolgreicher Anmeldung wird die Seite standardmäßig immer zu dem Pfad `sensordaten/overview` zurückgeleitet. Dort wird eine Tabelle mit allen Sensoren und den aktuellen Messwerten angezeigt (siehe Abbildung 40). Über das Häkchen `farbige Grenzwerte`

lassen sich die kategorisierten Messwerte hervorheben. Bei der Temperatur und Luftfeuchtigkeit beziehen sich die Farben Grün, Gelb und Rot auf das Behaglichkeitsfeld nach Frank[31] - behaglich, noch behaglich und nicht behaglich. Bei dem Kohlenstoffdioxidgehalt auf die Grenzwerte von unbedenklich (bis 1000 ppm), bedenklich (1000 bis 1500 ppm) und kritisch (ab 1500 ppm). Für die Feinstaubbelastung sind Grenzwerte der WHO aus 2021 und der EU für Jahresmittelwerte zu Grunde gelegt worden (siehe Abbildung 41). Für PM10 gelten 15 und 40, für PM2.5 5 und 25 Mikrogramm pro Kubikmeter Staubpartikel.

Gerät	↑ Gruppierung	Temperatur [°C]	Luftfeuchtigkeit [%]	Luftdruck [hPa]	CO ₂ [ppm]	PM 10 [µg/m³]	PM 2.5 [µg/m³]
arbeitszimmer	Hausc	27.9	50	-	552	6.1	1.2
flur	-	27.6	47	-	-	-	-
wohnzimmer	-	26.1	55	-	-	5.4	1.8
hochschule	-	23.9	57.3	1014.5	-	5.5	2.5

Abbildung 40: Tabellenübersicht

Neue WHO Leitlinien (Jahresmittel-Werte)

Luftschadstoff	WHO 2005	WHO 2021	EU-Grenzwert
Stickstoffdioxid	40 µg/m³	10 µg/m³	40 µg/m³
PM2,5	10 µg/m³	5 µg/m³	25 µg/m³
PM10	20 µg/m³	15 µg/m³	40 µg/m³

Abbildung 41: Grenzwerte für Feinstaubbelastung
[32]

Die Gerätegruppen lassen sich im Menü öffnen und schließen. Über einen Klick lässt sich zu den Sensordaten des jeweiligen ESP32 oder zurück zur Tabellenübersicht navigieren

(siehe Abbildung 42 und 43).



Abbildung 42: Menü geschlossen

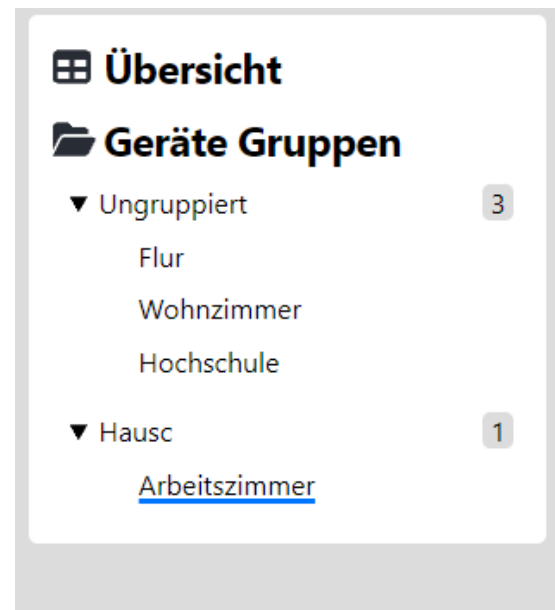



Abbildung 43: Menü geöffnet

Die Seite zu dem jeweiligen Sensor zeigt oben mittig den Namen mit zugehöriger Gruppe (falls vorhanden) und links und rechts Pfeile zur Navigation zum vorherigen und nächsten Gerät. Darunter befindet sich einmal das Behaglichkeitsfeld von Temperatur und Luftfeuchtigkeit, der Luftdruck und dann zwei Balkendiagramme für CO₂ und Feinstaub. Falls der jeweilige Mikrocontroller keinen Sensor für einen Messwert angeschlossen hat, wird dieser durch ein  ersetzt (siehe Abbildung 44).

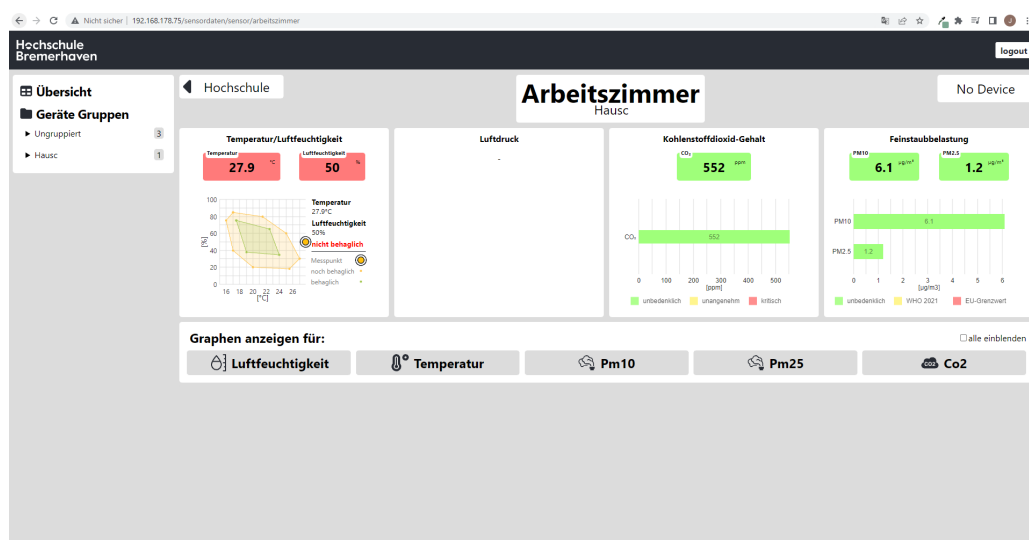


Abbildung 44: einzelne Sensor Seite

Die Darstellung der einzelnen Kacheln erfolgt für schmale Bildschirme vertikal (siehe Abbildung 45).

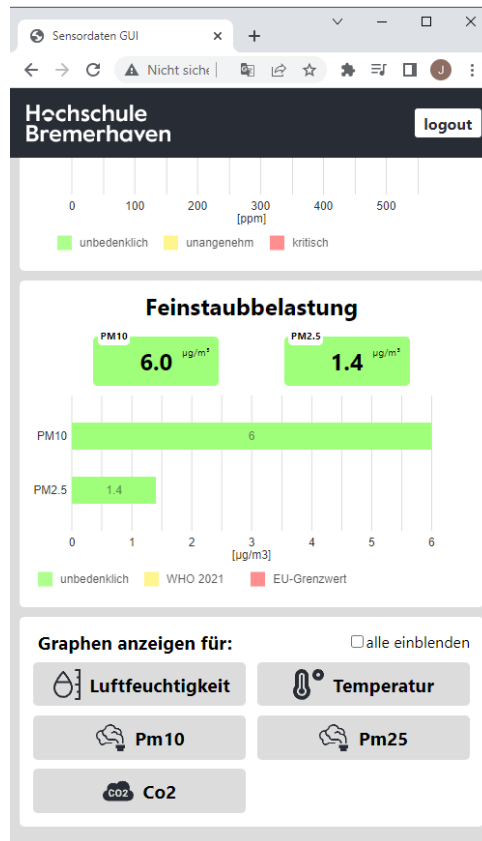


Abbildung 45: Responsive Design für Tablets/Phone

In dem Feld mit **Graphen anzeigen für:** befindet sich ein Button zu jedem verfügbaren Messwert, über den einzeln die Graphen eingeblendet und ausgeblendet werden können. Standardmäßig werden die Diagramme nicht eingeblendet, damit beim Durchklicken durch die einzelnen Devices nicht wiederholt Anfragen an die InfluxDB erfolgen. Über das Häkchen **alle einblenden** können gleichzeitig alle Graphen angezeigt oder ausgeblendet werden (siehe Abbildung 46). Zur Darstellung der Diagramme wurde die Library **Nivo** verwendet [33]. Diese ist für die Verwendung mit React gedacht und basiert auf einer der größten und verbreitetsten Libraries für die Darstellung von Daten in JavaScript - **D3** (Data Driven Documents).

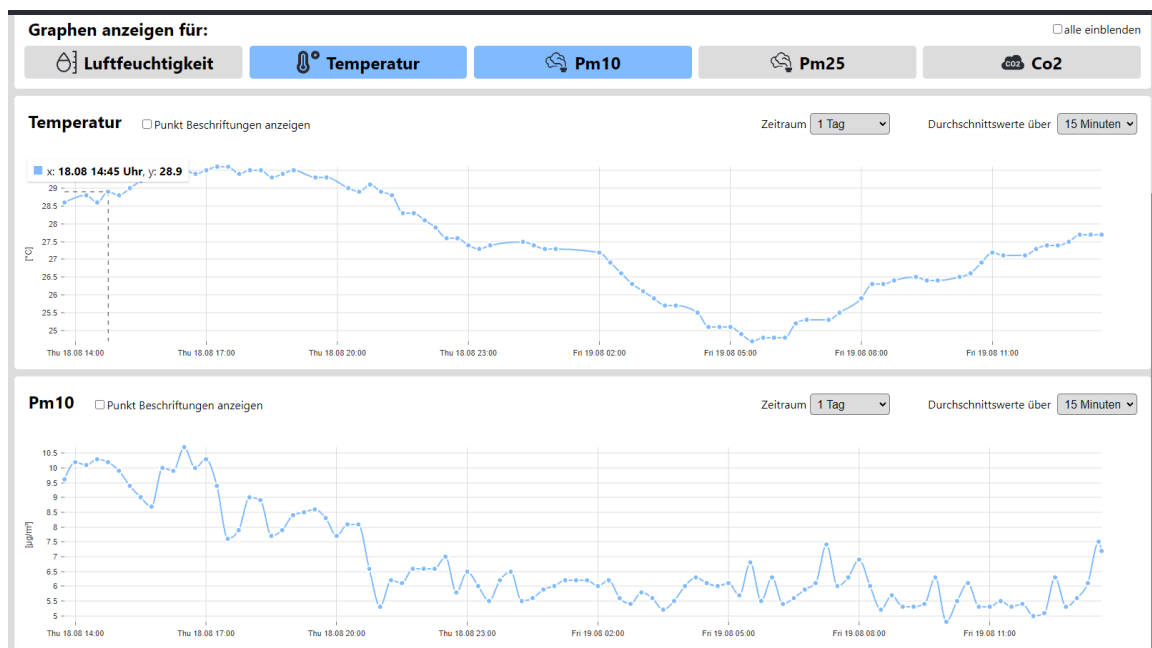


Abbildung 46: Graphen anzeigen lassen

In dem Feld zum Graphen selbst lässt sich der betrachtete Zeitraum und der Zeitbereich für die gemittelten Messpunkte aus einer vordefinierten Liste verändern. Außerdem lassen sich die Punktbeschriftungen für die einzelnen Punkte ein und ausstellen. Standardmäßig sind auch diese ausgeschaltet, da bei sehr vielen Messpunkten bei kleinen Durchschnittsintervallen sich die Texte der Messpunkte überlagern würden. Bei wenigen Messpunkten führen die Beschriftungen aber zu einer besseren Lesbarkeit des Diagramms (siehe Abbildung 47).

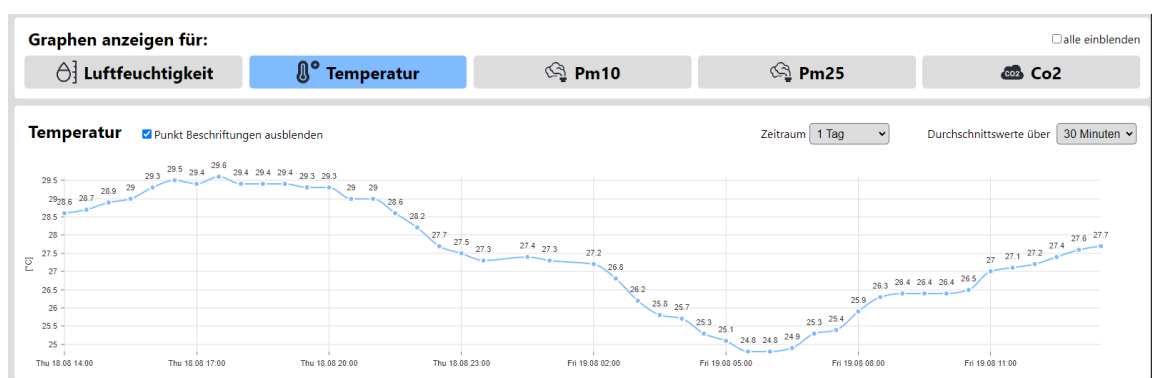


Abbildung 47: Graphen Messpunkte anzeigen

Die Punkte für die Behaglichkeitsfelder können bei Bedarf angepasst werden, müssen aber am Anfang und am Ende den gleichen Wert aufweisen, damit eine geschlossene Fläche

angezeigt wird (siehe Abbildung 48 und Code 32).

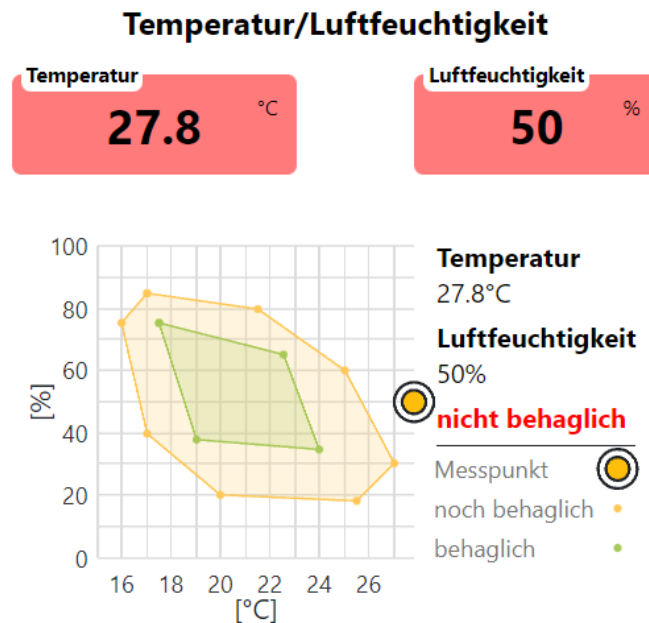


Abbildung 48: Behaglichkeitsfeld für Temperatur und Luftfeuchte nach Frank [31]

```

1  const polygonBehaglich = [
2    [17.5, 75],[22.5, 65],[24, 35],[19, 38],[17.5, 75],
3    ];
4
5  const polygonNochBehaglich = [
6    [17, 85],[21.5, 80],[25, 60],[27, 30],[25.5, 18],[20, 20],[17, 40],
7    [16, 75],[17, 85],
8    ];

```

Code 32: Koordinaten Behaglichkeitsfelder

6.2 Authentication

Home Assistant ermöglicht die Authentifizierung mittels des OAuth2.0 Protokoll-Ablaufs. Dies erlaubt eine Anmeldung auf der Sensordaten-Webseite ohne einen eigenen Anmeldevorgang und die Verwaltung von Benutzern. Um sich anzumelden, wird der Benutzer auf die Home Assistant Webseite umgeleitet, wo dieser sich mit den Benutzerdaten, die schon in Home Assistant eingerichtet wurden, einloggen kann. Nach erfolgreichem Login wird der Benutzer auf die ursprüngliche Webseite zurückgeleitet und erhält einen einmalig gültigen Authentifizierungs-Token, der für die Anfrage eines Access-/ und Refresh-Tokens benötigt

wird. Der Access-Token wird bei einer Application Programming Interface (API)-Anfrage oder einer WebSocket-Connection mitgeschickt und bestätigt bei Gültigkeit, dass der Benutzer berechtigt ist, die angefragten Daten zu erhalten. Der Access-Token hat meist nur eine kurze Gültigkeit und kann nach Ablauf durch einen Refresh-Token erneuert werden, ohne dass der Benutzer sich erneut einloggen muss (vergleiche Abbildung 49).

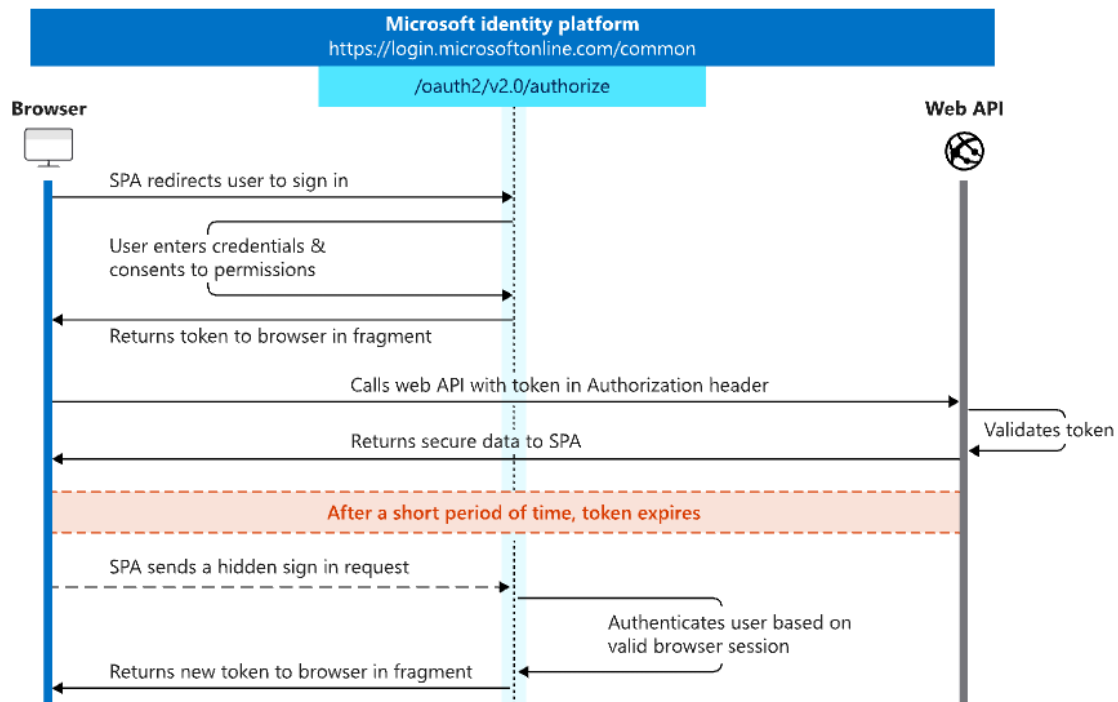


Abbildung 49: Microsoft OAuth2.0 Protokolldiagramm [34]

6.2.1 Implementierung

In React gibt es durch `useState()` Variablen, die in einer Komponente und allen Child-Komponenten zur Verfügung stehen. Bei der Authentifizierung benötigt man allerdings einen State, der die ganze Applikation umfasst und sagt, ob ein Benutzer eingeloggt ist oder nicht. Durch `createContext()` kann ein `AuthContext` geschaffen werden, der am besten im Top-Level der Applikation mit Hilfe des `Context.Provider` 's eine Art global State schafft, der einzeln in jeder Komponente importiert werden kann (siehe Code 33).

```

1 import React, { createContext } from "react";
2
3 const AuthContext = createContext(null);
4
5 const AuthProvider = ({ children }) => {

```

```

6
7   ...
8
9   // Objekt, dass via des AuthProviders verfuegbar sein soll
10  const value = {
11    token,
12    onLogin: handleLogin,
13    onLogout: handleLogout,
14    entities,
15  };
16
17  return <AuthContext.Provider value={value}>{children}</AuthContext.Provider>;
18 };
19
20 export { AuthProvider, AuthContext };
21
22
23
24 // ----- index.js File
25 import { AuthProvider } from "../components/AuthProvider";
26
27 ...
28
29 root.render(
30   <AuthProvider>
31     <App />
32   </AuthProvider>
33 );

```

Code 33: AuthContext

Der gesamte OAuth2.0 Flow wird von der `getAuth()` Funktion der `"home-assistant-js-websocket"` Library übernommen und wird durch die an die Funktion weitergegebenen Parameter beeinflusst. Das erhaltene Auth-Object mit den Access-/ und Refresh-Token wird im lokalen Session-Storage gespeichert und ist so lange verfügbar, bis der Benutzer sich wieder ausloggt oder den Browser, bzw. den offenen Tab schließt.

Die `handleLogin()` Function soll beim Klicken des Login-Buttons den Benutzer auf die Home Assistant Seite umleiten (siehe Code 34). Der Authentifizierungs-Token wird beim Zurückleiten an die URL angefügt.

```

1  import { url } from "../utils/settings";
2
3  const hassUrl = url;    // const url = "http://192.168.178.75:8123";
4
5  ...
6
7  const handleLogin = async () => {
8    try {
9      // Redirect user to log in on their instance
10     let auth = await getAuth({ hassUrl });
11   } catch (err) {

```

```

12     console.log('Unknown error: ${err}');
13   }
14 };

```

Code 34: Login Function

Nachdem der Benutzer sich erfolgreich angemeldet hat, gelangt dieser wieder auf die ursprüngliche Webseite. Für die React Application kommt dieser Vorgang dem erstmaligen Rendern der Seite gleich, es muss also in dem `useEffect()` Hook beim erstmaligen Rendern überprüft werden, ob der Benutzer gerade von Home Assistant redirected wurde, der Benutzer bereits angemeldet war oder gar nicht angemeldet ist. Da der Authentifizierungstoken nach einem Redirect nur einmalig gültig ist, soll der Benutzer nach dem Login immer zu der Übersichtstabelle weitergeleitet werden, damit `getAuth()` bei einem Seiten-Refresh nicht erneut versucht, sich mit dem gleichen Authentifizierungstoken Zugang zu verschaffen (siehe Code 35). Dies führt zu einem Fehler.

```

1  // speichert das Auth-Object in sessionStorage unter "auth"
2  const storeTokenInSessionStorage = (auth) => {
3    sessionStorage.setItem("auth", JSON.stringify(auth));
4  };
5
6
7  // laedt das Auth-Object aus dem sessionStorage , erwartet ein Promise , dass das Object oder
8  // undefined zurueckgibt.
9  const loadTokenInformationFromSessionStorage = () => {
10    let authFromStorage = JSON.parse(sessionStorage.getItem("auth"));
11    let promise = new Promise(function (resolve, reject) {
12      resolve(authFromStorage);
13      reject(undefined);
14    });
15    return promise;
16  };
17
18  // States
19  const [authState, setAuthState] = useState(null);
20  const [token, setToken] = useState(null);
21  const [entities, setEntities] = useState(null);
22
23  const checkForExistingAuth = useCallback(async () => {
24    let auth;
25    try {
26      // versucht zuerst die Tokens aus dem sessionStorage zu laden,
27      // falls keine Session -> checks URL nach Authentication-Token und speichert diesen dann
28      // mit saveTokens
29      auth = await getAuth({
30        saveTokens: storeTokenInSessionStorage,
31        loadTokens: loadTokenInformationFromSessionStorage,
32      });

```

```

33 // bei erfolgreicher Authentifizierung
34 if (auth) {
35     setAuthState(auth);
36     // Umleitung zur Tabelle
37     navigate("sensordaten/overview", { replace: true });
38 }
39 // bei nicht erfolgreicher Authentifizierung
40 } catch (err) {
41     console.log(
42         'Error Number: ${err} --- Click the login button to authenticate'
43     );
44 }
45
46 // bei erfolgreicher Authentifizierung - Websocket Connection mit Home Assistant
47 if (auth) {
48     const connection = await createConnection({ auth });
49     subscribeEntities(connection, (ent) => {
50         // speichere Ergebnis in entities State
51         setEntities(ent);
52     });
53     // Token state, ob Benutzer eingeloggt ist
54     setToken(true);
55 } else {
56     setToken(false);
57 }
58 }, []);
59
60
61 useEffect(() => {
62     checkForExistingAuth();
63 }, [checkForExistingAuth]);

```

Code 35: Authentifizierung überprüfen

Durch Drücken des Logout-Buttons wird die `handleLogout()` Function ausgelöst und der Refresh-Token widerrufen, alle States gelöscht und das Token-Object aus dem Session-Storage entfernt (siehe Code 36).

```

1 const handleLogout = () => {
2     authState.revoke();
3     sessionStorage.removeItem("auth");
4     setToken(false);
5     setAuthState(null);
6     setEntities(null);
7 };

```

Code 36: Logout Function

Vollständiger Code des `AuthProvider`s (siehe Code 37).

```

1 import React, { useState, createContext, useEffect, useCallback } from "react";
2 import {
3     getAuth,
4     createConnection,

```

```

5   subscribeEntities,
6 } from "home-assistant-js-websocket";
7 import { useNavigate } from "react-router-dom";
8
9 import { url } from "../utils/settings";
10
11 const hassUrl = url;
12
13 const AuthContext = createContext(null);
14
15 const AuthProvider = ({ children }) => {
16   const [token, setToken] = useState(null);
17   const [authState, setAuthState] = useState(null);
18   const [entities, setEntities] = useState(null);
19   let navigate = useNavigate();
20
21   const storeTokenInSessionStorage = (auth) => {
22     sessionStorage.setItem("auth", JSON.stringify(auth));
23   };
24
25   const loadTokenInformationFromSessionStorage = () => {
26     let authFromStorage = JSON.parse(sessionStorage.getItem("auth"));
27     let promise = new Promise(function (resolve, reject) {
28       resolve(authFromStorage);
29       reject(undefined);
30     });
31
32     return promise;
33   };
34
35   const checkForExistingAuth = useCallback(async () => {
36     let auth;
37     try {
38       auth = await getAuth({
39         saveTokens: storeTokenInSessionStorage,
40         loadTokens: loadTokenInformationFromSessionStorage,
41       });
42
43       if (auth) {
44         setAuthState(auth);
45         navigate("sensordaten/overview", { replace: true });
46       }
47     } catch (err) {
48       console.log(
49         'Error Number: ${err} --- Click the login button to authenticate'
50       );
51     }
52
53     if (auth) {
54       const connection = await createConnection({ auth });
55       subscribeEntities(connection, (ent) => {
56         setEntities(ent);
57       });
58       setToken(true);

```

```

59     } else {
60         setToken(false);
61     }
62 }, []);
63
64 const handleLogin = async () => {
65     try {
66         // Redirect user to log in on their instance
67         let auth = await getAuth({ hassUrl });
68         console.log("auth", auth);
69     } catch (err) {
70         console.log('Unknown error: ${err}');
71     }
72 };
73
74 const handleLogout = () => {
75     authState.revoke();
76     sessionStorage.removeItem("auth");
77     setToken(false);
78     setAuthState(null);
79     setEntities(null);
80 };
81
82 const value = {
83     token,
84     onLogin: handleLogin,
85     onLogout: handleLogout,
86     error,
87     entities,
88 };
89
90 useEffect(() => {
91     checkForExistingAuth();
92 }, [checkForExistingAuth]);
93
94 return <AuthContext.Provider value={value}>{children}</AuthContext.Provider>;
95 };
96
97 export { AuthProvider, AuthContext };

```

Code 37: Authentication Implementierung

Die Authentifizierung ist ein Schwerpunkt der gesamten Arbeit, da diese den Zugriff durch unbefugte Personen verhindert und gleichzeitig das Empfangen aller Home Assistant Entities mit den aktuellen Sensormesswerten ermöglicht. Alles weitere beschränkt sich, mit Ausnahme der Aufbereitung der erhaltenen Daten, auf vergleichsweise übersichtliche UI-Logik.

Das Entities Object enthält alle Home Assistant Entities als eigenes Object. Benötigt werden von jedem Sensor der `state` für den aktuellen Messwert, die `entity_id` (Bezeichnung) und `attributes.unit_of_measurement` die Einheit des Messwerts (siehe Abbildung 50).


```

▶ sensor.pm10_hochschule: {entity_id: 'sensor.pm10_hochschule', state: '2.4', attributes:
▼ sensor.pm10_wohnzimmer:
  ▼ attributes:
    device_class: "pm10"
    friendly_name: "PM10 Wohnzimmer"
    icon: "mdi:chemical-weapon"
    state_class: "measurement"
    unit_of_measurement: "µg/m³"
  ▶ [[Prototype]]: Object
▶ context: {id: '01GAVM98SN09RHZJCT0NY17WSH', parent_id: null, user_id: null}
  entity_id: "sensor.pm10_wohnzimmer"
  last_changed: "2022-08-19T18:01:45.013Z"
  last_updated: "2022-08-19T18:01:45.013Z"
  state: "3.6"
  ▶ [[Prototype]]: Object
▶ sensor.pm25_arbeitszimmer_hausc: {entity_id: 'sensor.pm25_arbeitszimmer_hausc', state:
▶ sensor.pm25_hochschule: {entity_id: 'sensor.pm25_hochschule', state: '1.0', attributes:
▶ sensor.pm25_wohnzimmer: {entity_id: 'sensor.pm25_wohnzimmer', state: '0.8', attributes:
▶ sensor.temperatur_arbeitszimmer_hausc: {entity_id: 'sensor.temperatur_arbeitszimmer_hau

```

Abbildung 50: WebSocket Entities Object

6.3 InfluxDB Queries

Der Aufbau einer Influx Query (Datenabfrage) lässt sich am besten anhand eines Beispiels aus dem InfluxDB Query Builder erklären (siehe Code 38).

Die Anfrage filtert nach allen Daten im Bucket `Sensors` in einer range von vor einem Tag (`start: -1d`) bis jetzt (es kann auch mit `stop:` ein Endzeitpunkt festgelegt werden) nach Werten mit der domain `sensor` und der `entity_id` von `luftfeuchtigkeit_arbeitszimmer_hausc` mit der Einheit `%` im Feld `value`. Die `aggregateWindow()` Funktion kann genutzt werden, um die Ausgabe der Daten zu modifizieren, z.B. können die Werte aufintegriert werden, der Median oder Mittelwert gebildet werden .

Mit `aggregateWindow(every: 15m, fn: mean, createEmpty: false)` wird der Mittelwerte aller Messwerte in einem Zeitraum von 15 Minuten als ein Datenpunkt ausgegeben und Zeiträume in denen keine Daten vorliegen, werden nicht berücksichtigt. Mit `createEmpty: true` würden auch alle 15 Minuten Werte mit 0 aufgenommen werden (vergleiche Abbildung 51 und 52).

```

1 from(bucket: "Sensors")
2   |> range(start: -1d)
3   |> filter(fn: (r) => r["_measurement"] == "%")
4   |> filter(fn: (r) => r["_field"] == "value")
5   |> filter(fn: (r) => r["domain"] == "sensor")
6   |> filter(fn: (r) => r["entity_id"] == "luftfeuchtigkeit_arbeitszimmer_hausc")
7   |> aggregateWindow(every: 15m, fn: mean, createEmpty: false)

```

Code 38: Beispiel Influx Query

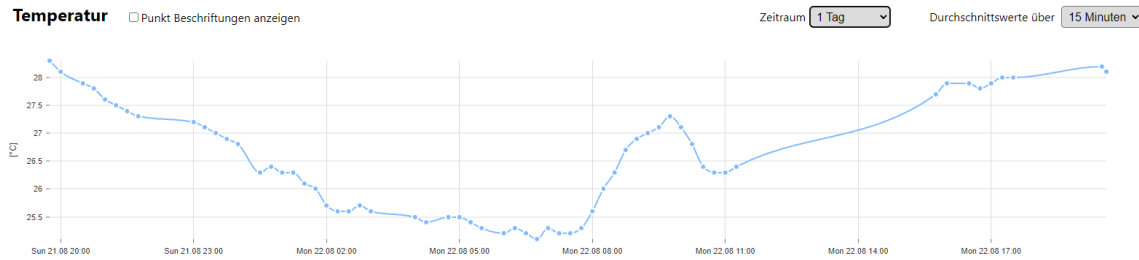


Abbildung 51: Graph - Query mit createEmpty: false

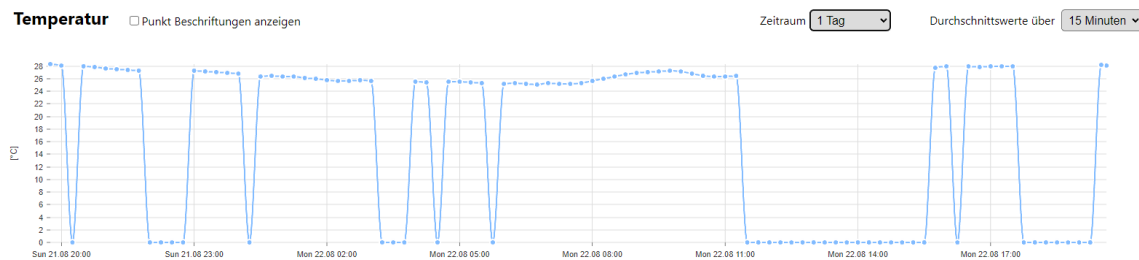


Abbildung 52: Graph - Query mit createEmpty: true

6.3.1 Integration Influx Query

Die Informationen zu der InfluxDB sowie die Bezeichnungen und Werte der Drop-Down Menüs für die Veränderung der betrachteten Zeiträume werde aus dem Order `/utils/settings` importiert.

```
1 import { org, influxToken, influxUrl, bucket, influxQuerySettings } from "../../utils/
  settings";
2
3 import { InfluxDB } from "@influxdata/influxdb-client-browser";
4
5 const client = new InfluxDB({
6   url: influxUrl,      // "http://192.168.178.75:8086";
7   token: influxToken,  // "X8nBrX1zQ0Hp...";
8 });
```

Code 39: Influx Client

Jeder Graph soll nur dann eine Datenabfrage an die InfluxDB senden, wenn eine `influxQuery` vorliegt und der Graph angezeigt wird und nicht vorher. Die `useEffect()` Funktion soll immer dann ausgeführt werden, wenn sich der `influxQuery` oder `show` State ändert. Die erhaltenen Daten werden dann im State `influxData` gespeichert.

```
1 useEffect(() => {
2   if (show && influxQuery) {
3     let res = [];
4     const queryApi = client.getQueryApi(org);
5   }
```

```

6      queryApi.queryRows(influxQuery, {
7          next(row, tableMeta) {
8              const o = tableMeta.toObject(row);
9              res.push(o);
10         },
11         error(error) {
12             console.error(error);
13             console.log("Finished ERROR");
14         },
15         complete() {
16             console.log("Finished SUCCESS");
17             console.log("result", res);
18             let result = dataToPoints(res);
19             setInfluxData(result);
20         },
21     });
22 }
23 }, [influxQuery, show]);

```

Code 40: Influx Query Integration

Für die Query-Parameter, die vollständige Query und die Influx-Daten werden verschiedene States initialisiert.

```

1      const [influxQueryParams, setInfluxQueryParams] = useState({
2          unit_of_measurement: "%",
3          field: "value",
4          domain: "sensor",
5          entity_id: null,
6          aggregate_timeframe: influxQuerySettings[0].aggregate_array[0].value, // "15m"
7          aggregate_timeframes_for_period: influxQuerySettings[0].aggregate_array, // { value:
            "15m", label: "15 Minuten" }
8          start: influxQuerySettings[0].value, // "1d"
9          arrayIndex: 0,
10     });
11
12     const [influxData, setInfluxData] = useState(null);
13     const [influxQuery, setInfluxQuery] = useState(null);

```

Code 41: State Initialisation

Beim erstmaligen Rendern oder jedes mal, wenn sich die Daten des ESP32 in `device` ändern oder ein neues Gerät ausgewählt wird und sich der `name` ändert, sollen alle vorherigen Query Parameter übernommen werden und nur die `entity_id` und `unit_of_measurement` geupdated werden.

Das return-Statement in dem `useEffect`-Hook wird immer dann ausgeführt, wenn eine Komponente nicht mehr dargestellt wird oder bevor der `useEffect`-Hook erneut ausgeführt wird, als eine Art Clean-Up Funktion. Dadurch kann erreicht werden, dass beim Wechseln des Gerätes die Label der Drop-Down-Menüs und die Daten beim Einblenden der Graphen wieder den gleichen Zeitraum anzeigen, da ansonsten die Label wieder auf den ersten Wert

im Array zurückgesetzt würden, die Daten im Graphen aber noch immer auf einer Query mit einem anderen Zeitraum beruhen.

```

1  useEffect(() => {
2    setInfluxQueryParams((prevQueryParams) => ({
3      ...prevQueryParams,
4      entity_id: device[name]?.entity_id,
5      unit_of_measurement: device[name]?.unit_of_measurement,
6    }));
7  }, [device, name]);
8
9  useEffect(() => {
10   return () => {
11     return setInfluxQueryParams((prevQueryParams) => ({
12       ...prevQueryParams,
13       aggregate_timeframe: influxQuerySettings[0].aggregate_array[0].value,
14       aggregate_timeframes_for_period: influxQuerySettings[0].aggregate_array,
15       start: influxQuerySettings[0].value,
16     }));
17   };
18 }, [show]);

```

Code 42: Update Query Parameter

Wenn der Graph angezeigt wird, soll die fertige `influxQuery` aus den Query-Parametern gebildet werden und im `influxQuery`-State gespeichert werden, sofern die gebildete Query nicht schon mit der Query im State übereinstimmt.

```

1  useEffect(() => {
2    if (show) {
3      let query = `from(bucket: "${bucket}")
4      |> range(start: -${influxQueryParams.start})
5      |> filter(fn: (r) => r["_measurement"] == "${influxQueryParams.unit_of_measurement}")
6      |> filter(fn: (r) => r["_field"] == "${influxQueryParams.field}")
7      |> filter(fn: (r) => r["domain"] == "${influxQueryParams.domain}")
8      |> filter(fn: (r) => r["entity_id"] == "${influxQueryParams.entity_id}")
9      |> aggregateWindow(every: ${influxQueryParams.aggregate_timeframe}, fn: mean,
10        createEmpty: false)
11      |> yield(name: "mean")`;
12
13      setInfluxQuery((prevQuery) => (prevQuery === query ? prevQuery : query));
14    }
15  }, [
16    influxQueryParams.start, influxQueryParams.unit_of_measurement,
17    influxQueryParams.field, influxQueryParams.domain,
18    influxQueryParams.entity_id, influxQueryParams.aggregate_timeframe,
19    show,
20  ]);

```

Code 43: Dynamic Influx Query

6.4 Routing

Routing in Bezug auf die Browser-URL wird in dieser Applikation von der JS-Library **reactrouter** übernommen [35]. Routes ermöglichen den Zugriff auf unterschiedliche Teile des Programms basierend auf der momentanen URL mit Hilfe von **Segments** die durch `/` getrennt sind. **Dynamic Routes** wie z.B. `/sensordaten/sensor/:deviceId` -> `/sensordaten/sensor/arbeitszimmer` können genutzt werden um sich mit verändernden Elementen der URL, wie hier die Bezeichnung des ESP32, zu matchen. Durch **Links** kann der Benutzer beim Klicken eines Buttons auf eine bestimmte URL weitergeleitet werden. Es ist auch möglich, auf den History Stack des Browsers zurückzugreifen (Verlauf der zuletzt aufgerufenen URL's) und so den Benutzer gegebenenfalls nach einem bestimmten Ereignis wieder zurückzuleiten. Die URL Parameter können in einer React-Komponente mit `useParams()` aus "react-router-dom" ausgelesen werden und es ist somit möglich aus der URL die Gerätenamen zu extrahieren und dann in den Daten nach dem richtigen Gerät zu filtern.

Ähnlich wie beim `<AuthProvider>` benötigt man auch einen `<BrowserRouter>`, der die gesamte Applikation einfasst und somit Zugriff auf die reactrouter Komponenten gewährt. Unter `<Routes>` können alle verschiedenen Segmente der Applikation eingefügt werden. Man benötigt nur ein einziges Segment mit dem `path='/sensordaten/'`, in dem noch weitere Nested Routes untergebracht werden. Der Pfad der Nested Routes wird an den Pfad der Parent-Route angefügt und muss nicht komplett neu aufgeführt werden. Die Nested Routes wären einmal `overview`, die die Übersichtstabelle `<OverviewTabelle>` rendert und die dynamic Route `sensor/:deviceId`, die auf die Messwerte des jeweiligen Geräts verlinkt und die `<Device>` Komponente rendert. Für den Fall, dass die URL mit keiner der obigen Pfade übereinstimmt, wird noch eine so genannte No-Match-Route benötigt, die garantiert, dass bei falscher URL kein Fehler auftritt.

```

1 // index.js
2 import { BrowserRouter } from "react-router-dom";
3
4 root.render(
5   <BrowserRouter>
6     <AuthProvider>
7       <App />
8     </AuthProvider>
9   </BrowserRouter>
10 );
11
12 // ----- App.js
13 import { Routes, Route } from "react-router-dom";
14
15 return (

```

```

16   <Routes>
17     <Route path="/sensordaten/" element={<Layout devices={devices} />}>
18       <Route
19         path="overview"
20         element={<OverviewTabelle devices={devices} />}
21       />
22     <Route path="sensor/:deviceId" element={<Device devices={devices} />} />
23     <Route path="*" element={<p>Falsche URL</p>} />
24   </Routes>
25 </Routes>
26 );

```

Code 44: React Router Implementierung

Die Layout-Komponente ist das oberste UI-Element der Applikation. Hier kann mit dem Custom-Hook `useAuth()` auf den AuthContext der Applikation zugegriffen werden, um zu überprüfen, ob der Benutzer eingeloggt ist oder nicht, und dem entsprechend die Sensordaten oder die Login Seite angezeigt wird. Die Komponente `<Outlet>` dient als Platzhalter für Nested Routes, diese wird später durch die Komponente der jeweiligen aktiven Route ersetzt.

```

1  import React from "react";
2
3  import { Outlet } from "react-router-dom";
4  import useAuth from "../../hooks/useAuth";
5
6  import DeviceSelectMenu from "../deviceSelectMenu/DeviceSelectMenu";
7  import LoginPage from "../loginPage/LoginPage";
8
9  import LayoutSC from "../styledComponents/StyledComponents";
10 import Header from "../header/Header";
11
12 const Layout = ({ devices }) => {
13   const { token } = useAuth(); // true or false
14
15   return (
16     <LayoutSC>
17       <Header />
18       <LayoutSC.Body>
19         {token ? (
20           <>
21             <DeviceSelectMenu devices={devices} />
22             <Outlet />
23           </>
24         ) : (
25           <LoginPage />
26         )}
27       </LayoutSC.Body>
28     </LayoutSC>
29   );
30 };
31

```

```
32 export default Layout;
```

Code 45: React Router Layout

`useAuth()` ist eine Funktion die auf den Context aus dem Abschnitt Authentication zugreift und den Token bzw. Auth-State und die Login/Logout-Funktionen zurückgibt.

```
1 import { useContext } from "react";
2 import { AuthContext } from "../components/AuthProvider";
3
4 const useAuth = () => {
5   return useContext(AuthContext);
6 };
7
8 export default useAuth;
```

Code 46: Custom Hook useAuth

6.5 Aufbereitung der Home Assistant Daten

Das Object, dass über die Web-Socket Verbindung mit Home Assistant übertragen wird, enthält alle Entitäten, die in Home Assistant verwendet werden, als weiteres Object, dass auch viele Werte enthält, die gar nicht weiter benötigt werden. Außerdem sind darunter auch einige Entities, die nicht zum Typ Sensor gehören. Desweiteren müssen die Entitäten mit unterschiedlichen Messwerten, die aber zum selben ESP32 gehören, gruppiert (siehe Code 47 und Abbildung 53) und dann in Array umgewandelt werden (siehe Code 48 und Abbildung 54), um die Weiterverwendung zu erleichtern.

```
1 const convertEntities = (entities) => {
2   // es soll ein neues Object angelegt werden, mit einem Key fuer jeden ESP32
3   let sortedEntitiesObject = {};
4   //for loop - fuer jede Entity
5   for (const [entity, values] of Object.entries(entities)) {
6     // check if entity is a sensor -- starts with sensor.
7     // bsp: sensor.pm10_arbeitszimmer_hausc -> splitName = ["sensor", "
8       pm10_arbeitszimmer_hausc"]
9     let splitName = entity.split(".");
10    let entityTypeIsSensor = splitName[0] === "sensor" ? true : false;
11
12    if (entityTypeIsSensor) {
13      let [measurement, name, group] = splitName[1].split("_"); // ["pm10", "arbeitszimmer", "
14        hausc"]
15
16      if (name) {
17        //schau ob das Object schon eine Key fuer das jeweilige Device hat
18        let deviceAlreadyExists = sortedEntitiesObject.hasOwnProperty(name);
19
20        // Aufbau des Object eines einzelnen Messwerts
21        let newEntityObject = {
```

```

20     state: values.state,
21     unit_of_measurement: values.attributes.unit_of_measurement,
22     last_updated: values.last_updated,
23     entity_id: splitName[1],
24     friendly_name: values.attributes.friendly_name,
25 };
26
27 // wenn das device schon existiert
28 if (deviceAlreadyExists) {
29     let existingProperties = sortedEntitiesObject[name];
30     sortedEntitiesObject[name] = {
31         ...existingProperties,
32         [measurement]: newEntityObject,
33     };
34 } else {
35     // ansonsten erstelle eine neuen Key mit dem device namen
36     sortedEntitiesObject[name] = {
37         [measurement]: newEntityObject,
38         group: group ? group : null,
39     };
40 }
41 }
42 }
43 }
44
45 return sortedEntitiesObject;
46 };

```

Code 47: converEntities Funktion

```

▼ Object ⓘ
  ▼ arbeitszimmer:
    ▼ co2:
      entity_id: "co2_arbeitszimmer_hausc"
      friendly_name: "CO2 Arbeitszimmer HausC"
      last_updated: "2022-08-23T07:10:49.595Z"
      state: "556"
      unit_of_measurement: "ppm"
      ► [[Prototype]]: Object
      group: "hausc"
      ► luftfeuchtigkeit: {state: '48', unit_of_measurement: '%', last_updated: '2022-08-23T06:
      ► pm10: {state: '6.8', unit_of_measurement: 'µg/m³', last_updated: '2022-08-23T07:10:44.8
      ► pm25: {state: '1.4', unit_of_measurement: 'µg/m³', last_updated: '2022-08-23T07:10:44.8
      ► temperatur: {state: '23.5', unit_of_measurement: '°C', last_updated: '2022-08-23T07:07:
      ► [[Prototype]]: Object
      ► flur: {temperatur: {...}, group: null, luftfeuchtigkeit: {...}}
      ► hochschule: {pm25: {...}, group: null, pm10: {...}, temperatur: {...}, luftdruck: {...}, ...}
      ► wohnzimmer: {pm25: {...}, group: null, pm10: {...}, temperatur: {...}, luftfeuchtigkeit: {...}}
      ► [[Prototype]]: Object

```

Abbildung 53: Entities Object Gruppierung


```

1  const objectToArray = (object) => {
2    let array = [];
3
4    if (object) {
5      for (const [device, properties] of Object.entries(object)) {
6
7        array.push({ device, ...properties });
8      }
9    }
10
11    return array;
12  };

```

Code 48: objectToArray Funktion

```

array
▼ (4) [{...}, {...}, {...}, {...}] ⓘ
  ▼ 0:
    device: "hochschule"
    group: null
    ▼ luftdruck:
      entity_id: "luftdruck_hochschule"
      friendly_name: "Luftdruck Hochschule"
      last_updated: "2022-08-23T06:47:03.989Z"
      state: "1018.3"
      unit_of_measurement: "hPa"
      ▶ [[Prototype]]: Object
    ▶ luftfeuchtigkeit: {state: '46.7', unit_of_measurement: '%', last_updated: '2022-08-23T07:07:04.105'}
    ▶ pm10: {state: '4.4', unit_of_measurement: 'µg/m³', last_updated: '2022-08-23T07:11:02.727Z', entity}
    ▶ pm25: {state: '2.0', unit_of_measurement: 'µg/m³', last_updated: '2022-08-23T07:11:02.720Z', entity}
    ▶ temperatur: {state: '23.5', unit_of_measurement: '°C', last_updated: '2022-08-23T06:32:04.010Z', er}
    ▶ [[Prototype]]: Object
    ▶ 1: {device: 'flur', group: null, luftfeuchtigkeit: {...}, temperatur: {...}}
    ▶ 2: {device: 'wohnzimmer', group: null, luftfeuchtigkeit: {...}, temperatur: {...}, pm10: {...}, ...}
    ▶ 3: {device: 'arbeitszimmer', group: 'hausc', luftfeuchtigkeit: {...}, temperatur: {...}, pm10: {...}, ...}
    length: 4
    ▶ [[Prototype]]: Array(0)

```

Abbildung 54: Entities Array

6.6 Hinweise zum React Quellcode

Das `package.json` File enthält alle Dependencies, also JavaScript Libraries, die benötigt werden, um den Code ausführen zu können (siehe Code 49). Mit dem Command `npm install` im Terminal werden alle Libraries automatisch aus dem `package.json` File installiert. Das Package `"home-assistant-js-websocket"` benötigt den Zusatz `npm install --legacy-peer-deps`, da einige Dependencies der Library wohl im Konflikt mit anderen stehen und somit ein Installationsfehler auftreten würde.

```
1 {
2   "name": "influxgui",
3   "version": "0.1.0",
4   "private": true,
5   "homepage": "/sensordaten",
6   "dependencies": {
7     "@influxdata/influxdb-client-browser": "^1.26.0",
8     "@nivo/bar": "^0.79.1",
9     "@nivo/core": "^0.79.0",
10    "@nivo/line": "^0.79.1",
11    "@testing-library/jest-dom": "^5.16.4",
12    "@testing-library/react": "^13.3.0",
13    "@testing-library/user-event": "^13.5.0",
14    "home-assistant-js-websocket": "^7.1.0",
15    "react": "^18.1.0",
16    "react-dom": "^18.1.0",
17    "react-router-dom": "^6.3.0",
18    "react-scripts": "5.0.1",
19    "robust-point-in-polygon": "^1.0.3",
20    "styled-components": "^5.3.5"
21  },
22  "scripts": {
23    "start": "react-scripts start",
24    "build": "react-scripts build",
25    "test": "react-scripts test",
26    "eject": "react-scripts eject"
27  },
28  "eslintConfig": {
29    "extends": [
30      "react-app",
31      "react-app/jest"
32    ]
33  },
34  "browserslist": {
35    "production": [
36      ">0.2%",
37      "not dead",
38      "not op_mini all"
39    ],
40    "development": [
41      "last 1 chrome version",
42      "last 1 firefox version",
43      "last 1 safari version"
44    ]
45  }
46 }
```

Code 49: package.json File

6.6.1 Aufbau und Ordnerstrukturen

Das index.html File befindet sich im Order `public`, der Code der React Application ist im Order `src` untergebracht und einige Utility Functions und Settings sind unter `utils` zu finden (siehe Abbildung 55). Da die React Functional Components genau wie das DOM des Browsers eine Baum-Struktur aufweisen, in der eine Parent Component meist eine oder mehrere Child Components aufweist, sind die Order für die Dateien ähnlich strukturiert. Die Komponenten sind in dem Order `components` zu finden, die Top-Level Komponenten haben jeweils einen eigenen Ordern. In dem zugehörigen Order befindet sich das gleichnamige `jsx` File und ein `styledComponents.jsx` File für das CSS-Styling. Alle Child-Components sind in dem Order `childComponents` untergebracht (siehe Abbildung 56).

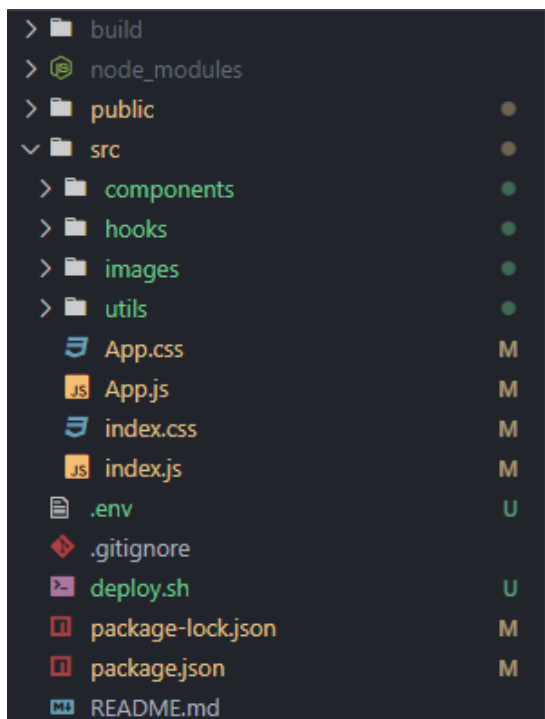


Abbildung 55: Ordnerstruktur Code

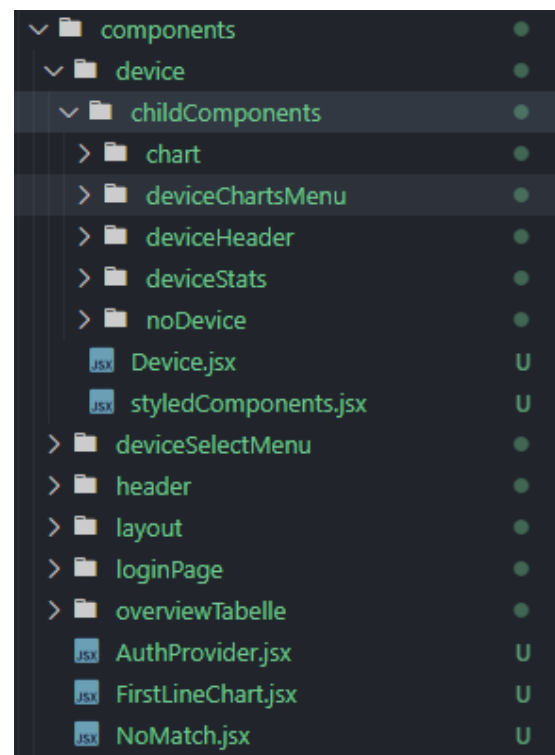


Abbildung 56: React Komponenten Ordner Struktur

6.6.2 Installation der React Application

Der Build Command für die React Application kann über ein Bash-Skript ausgeführt werden. Um ein Bash-Skript ausführen zu können, wird ein Terminal benötigt, das Bash

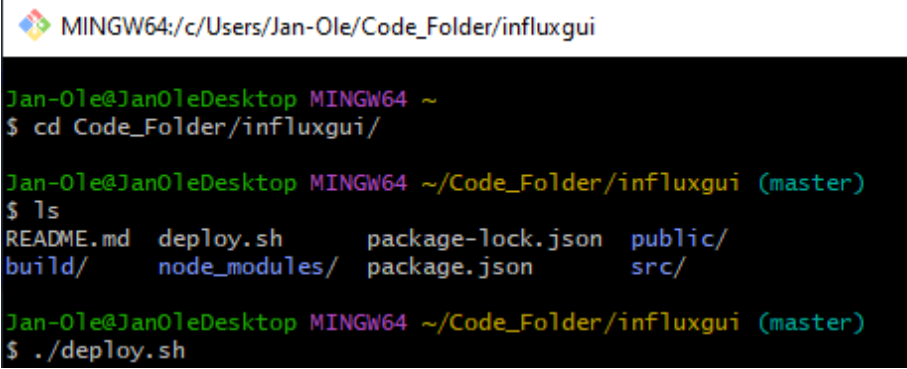
ausführen kann. Dazu eignet sich z.B. git-BASH. Git-BASH ist im Tool `git for windows` enthalten [36].

Mit `scp -r build/* pireact@192.168.178.75:/var/www/192.168.178.75/sensordaten/` wird der Inhalt des build-Folders in den Ordner `/var/www/192.168.178.75/sensordaten/` des Raspberry Pi's mit der IP-Adresse `192.168.178.75` unter dem Benutzer `pireact` übertragen (siehe Code 50). Bei Ausführung muss der Vorgang auch mit dem Passwort des jeweiligen Benutzers bestätigt werden. Bei eigener Einrichtung ist die IP-Adresse natürlich zu ersetzen.

```
1 echo "Switching to branch master"
2 git checkout master
3
4 echo "Building Site..."
5 npm run build
6
7 echo "Deploying files so server..."
8 scp -r build/* pireact@192.168.178.75:/var/www/192.168.178.75/sensordaten/
9
10 echo "Done."
```

Code 50: `deploy.sh` - Bash-Script zur Installation der React Application

Der Befehl `./deploy.sh` muss in dem Ordner ausgeführt werden, in dem das `deploy.sh` File abgespeichert ist (siehe Abbildung 57).



```
MINGW64/c/Users/Jan-Ole/Code_Folder/influxgui

Jan-Ole@JanOleDesktop MINGW64 ~
$ cd Code_Folder/influxgui/

Jan-Ole@JanOleDesktop MINGW64 ~/Code_Folder/influxgui (master)
$ ls
README.md  deploy.sh      package-lock.json  public/
build/     node_modules/  package.json       src/

Jan-Ole@JanOleDesktop MINGW64 ~/Code_Folder/influxgui (master)
$ ./deploy.sh
```

Abbildung 57: Git Bash `deploy.sh`

Nach dem Build-Vorgang muss die Übertragung mit dem Benutzerpasswort bestätigt werden. Der abgeschlossene Vorgang sieht wie folgt aus (siehe Abbildung 58).

```
The project was built assuming it is hosted at /sensordaten/.
You can control this with the homepage field in your package.json.

The build folder is ready to be deployed.

Find out more about deployment here:

  https://cra.link/deployment

Deploying files to server...
pireact@192.168.178.75's password:
asset-manifest.json          100% 429    95.0KB/s  00:00
index.html                  100% 647   166.1KB/s  00:00
manifest.json               100% 492   110.0KB/s  00:00
robots.txt                  100% 67     5.9KB/s   00:00
main.131d38ce.css           100% 640   154.8KB/s  00:00
main.131d38ce.css.map       100% 1259   264.5KB/s  00:00
main.3e3af69f.js            100% 617KB 478.3KB/s  00:01
main.3e3af69f.js.LICENSE.txt 100% 2873  391.8KB/s  00:00
main.3e3af69f.js.map        100% 2245KB 457.0KB/s  00:04
Done.

Jan-01e@Jan01eDesktop MINGW64 ~/Code_Folder/influxgui (master)
$ |
```

Abbildung 58: Deploying Files to Server

7 Diskussion

Motivation der Arbeit ist es, dass die Raumluftparameter der Hochschule zukünftig messtechnisch erfasst werden sollen. Die gesammelten Daten sollen als Grundlage für die Beurteilung des Raumluftklimas in den Vorlesungsräumen genutzt werden können. Zusätzlich soll aber auch die Erfassung der Feinstaubbelastung im Außenbereich möglich sein. Die Mikrocontroller können dort nicht kabelgebunden betrieben werden können, weshalb eine Kommunikation über W-Lan möglich sein muss.

Der Fokus liegt dabei auf einer Low-Cost Alternative zu herkömmlichen Sensorgeräten, die, gerade in hoher Stückzahl, bei einem Einzelpreis von über 100-200 Euro ein großes Investitionsvolumen annehmen.

Home Assistant, als Softwaregrundlage in Kombination mit der ESPHome Erweiterung, ermöglicht die Verwaltung von ESP32 Mikrocontrollern und Einbindung von Sensoren durch die Markup Language YAML. Außerdem lässt sich die Datenübertragung an InfluxDB ebenfalls über ein Configuration-File einrichten oder direkt durch ein Community-Plugin in Home Assistant integrieren. Die Verwendung der Open-Source Software ist kostenlos. Für den dauerhaften Betrieb der Messeinrichtung sollte der ESP32 in ein Gehäuse eingefasst werden, auf/in dem auch die einzelnen Sensoren mit genügend Luftzufuhr fixiert werden können. Die Steckverbindungen der Pins sind in der Regel etwas locker, was Wackelkontakte begünstigt und in Kombination mit Störungen im W-Lan gelegentlich zu Ausfällen führen kann.

Für den Raspberry Pi ist eine MicroSD-Karte mit vielen Lese- und Schreibzyklen oder eine externe SSD notwendig und eine Kühlung durch einen kleinen Lüfter, da die Wärmeentwicklung im 24 Stunden Betrieb nicht unerheblich ist.

Der Vorteil einer selbst programmierten Applikation gegenüber einer vorgefertigten Software ist, dass die angezeigten Daten und verwendeten Grafiken selbst gewählt und genau auf die Bedürfnisse zugeschnitten werden können. Der Nachteil ist hingegen, dass der Aufwand um ein vielfaches größer ist und auch schon kleine Änderungen auch neue Programmierarbeit bedeuten. Andere Programme haben meist einen größeren Funktionsumfang der auch Modifikationen von z.B. Grenzwerten, Farben oder Zeiträumen durch Menüs in dem Graphical User Interface (GUI) ermöglicht und die Änderungen im Backend speichert und beim nächsten Öffnen automatisch anwendet.

Die Websocket Connection mit Home Assistant achtet auf Veränderungen der Sensormesswerte und überträgt jedes Mal alle Entitäten, sobald sich ein einzelner Messwert ändert. Das bedeutet allerdings, dass mit mehreren Sensoren nicht nur mehr Daten übertragen werden, sondern auch die Häufigkeit, mit der sich die Daten ändern, immer weiter zu-

nimmt. Hier bestünde noch Klärungsbedarf, ob und wie Home Assistant bei erstmaligem Anmelden alle aktuellen Sensormesswerte übermittelt und danach nur noch die einzelnen Entitäten, die sich kürzlich verändert haben. Das Testen der Belastbarkeit und Stabilität bei gleichzeitigem Zugriff von mehreren Benutzern konnte im Rahmen dieser Arbeit nicht getestet werden. Es ist auch möglich, die Seite gar nicht über den Raspberry Pi bereitzustellen, sodass diese nur über eine Entwicklungsumgebung wie z.B. Visual Studio Code, lokal auf dem Rechner läuft.

Mit einer Lüftungsanlage kann durch ein Kühl-/Heizregister und einem Be-/Entfeuchter nur direkter Einfluss auf die Temperatur und Luftfeuchtigkeit der Raumluft genommen werden. Die Verringerung der Kohlenstoffdioxidkonzentration im Raum ist nur durch die Zufuhr frischer Außenluft möglich. Die Feinstaubpartikel sind in der Regel zu klein, um in der Lüftungsanlage effektiv herausgefiltert zu werden.

Da im Moment jedes Gerät theoretisch alle Messwerte für Temperatur, Luftfeuchtigkeit, Luftdruck, CO₂ und Feinstaub erfassen kann und dafür ein eigenes Feld in der Visualisierung besitzt, wäre eine Unterteilung von Außensensoren zur Erfassung des Feinstaubes und Innensensoren zur Erfassung von Temperatur, Luftfeuchtigkeit und CO₂ denkbar. Eine zweite Übersichtstabelle könnte für die Trennung der zwei Sensorgruppen und eine bessere Vergleichbarkeit sorgen.

Zum jetzigen Zeitpunkt ist die grafische Benutzeroberfläche ein reines Tool zur Darstellung der aktuellen Messwerte, die über Home Assistant bereit gestellt werden und der Zeitreihendaten aus der InfluxDB, die in Graphenform dargestellt sind. Die Betrachtung der Daten im Diagramm ist zurzeit nur vom aktuellen Datum bis zu einem Zeitpunkt (aus einer definierten Liste an Zeiträumen) von einem Jahr in der Vergangenheit möglich. Die Auflösung der Daten nimmt für einen Zeitpunkt, der schon länger vergangen liegt, ab. Über das InfluxDB Interface kann der Zeitraum aber auch konkret noch einmal betrachtet werden. Der Unterschied zur Visualisierung in InfluxDB ist, dass die eigene Sensordatenseite einen schnelleren Überblick über alle Messwerte ermöglicht und die Navigation zwischen verschiedenen Geräten deutlich erleichtert.

Nach Ausfällen oder Neustarts kann es vorkommen, dass Sensoren kurzzeitig falsche Daten übermitteln, die dann trotzdem an die Datenbank übertragen werden. Eine Filterung dieser falschen Messwerte erfolgt momentan nicht.

Diese Arbeit bietet eine solide Grundlage für die Automation eines Lüftungssignals, basierend auf den gesammelten Daten. Home Assistant unterstützt auch viele Funktionen der Home Automation, die genaue Umsetzung übersteigt den Rahmen dieser Bachelorarbeit und wäre ein Thema für eine zukünftige Ausarbeitung.

8 Fazit

Das Ziel dieser Bachelorarbeit war es, ein geeignetes Tool zur Visualisierung von Sensormesswerten zu schaffen, dass die Beurteilung der Raumluftparameter in den Vorlesungsräumen und die Feinstaubkonzentration im Außenbereich der Hochschule Bremerhaven erleichtern soll. Der Schwerpunkt lag dabei auf einer kostengünstigen Lösung. Die Umsetzung erfolgte durch die Kombination von kostenloser Open-Source Software mit preiswerter Sensortechnik. Das Ergebnis hat gezeigt, dass es durch ausgereifte Open-Source Projekte wie Home Assistant und InfluxDB, möglich ist, eigene Projekte im Bereich Messtechnik, Smart Home und Zeitreihendaten umzusetzen. Code-Libraries in den gängigsten Programmiersprachen wie z.B. Python, Javascript, Java, C++, usw. ermöglichen zudem den Export/Import der Daten, um auch außerhalb der Programme mit den Information arbeiten zu können.

Literatur

- [1] Katharina Aganina. Digitalisierung in deutschland: Wie ist der aktuelle stand?, 2022. URL <https://www.ratbacher.de/blog/digitalisierung-stand-deutschland/>. Zuletzt besucht: 2022.08.03.
- [2] Lungenaerzte im Netz. Schlechte belüftung in innenräumen erhöht das corona-infektionsrisiko, 2022. URL <https://www.lungenaerzte-im-netz.de/news-archiv/meldung/article/schlechte-belueftung-in-innenraeumen-erhoeht-das-corona-infektionsrisiko>. Zuletzt besucht: 2022.08.03.
- [3] homeassistant. Home assistant, 2022. URL <https://www.home-assistant.io/>. Zuletzt besucht: 2022.08.24.
- [4] ESPHome. Getting started with esphome and home assistant, 2022. URL https://www.esphome.io/guides/getting_started_hassio.html. Zuletzt besucht: 2022.08.04.
- [5] React. React - a javascript library for building user interfaces, 2022. URL <https://reactjs.org/>. Zuletzt besucht: 2022.08.24.
- [6] Wikipedia. Document object model, 2022. URL https://de.wikipedia.org/wiki/Document_Object_Model. Zuletzt besucht: 2022.08.24.
- [7] Gordon Hollingworth. Raspberry pi os (64-bit), 2022. URL <https://www.raspberrypi.com/news/raspberry-pi-os-64-bit/>. Zuletzt besucht: 2022.08.01.
- [8] InfluxDB. Get started with influxdb oss 2.3, 2022. URL <https://docs.influxdata.com/influxdb/v2.3/>. Zuletzt besucht: 2022.08.13.
- [9] Home Assistant. Install home assistant supervised, 2022. URL <https://github.com/home-assistant/supervised-installer>. Zuletzt besucht: 2022.08.01.
- [10] nginx. nginx webserver-software, 2022. URL <https://www.nginx.com/>. Zuletzt besucht: 2022.08.09.
- [11] NodeJs. Nodejs website, 2022. URL <https://nodejs.org/en/>. Zuletzt besucht: 2022.08.13.
- [12] Ubuntu. Ubuntu betriebssystem für raspberry pi 4, 2022. URL <https://ubuntu.com/download/raspberry-pi>. Zuletzt besucht: 2022.07.31.

- [13] Ubuntu. How to install ubuntu desktop on raspberry pi 4, 2022. URL <https://ubuntu.com/tutorials/how-to-install-ubuntu-desktop-on-raspberry-pi-4#1-overview>. Zuletzt besucht: 2022.07.31.
- [14] Kris Koishigawa. sudo apt-get update vs upgrade – what is the difference?, 2022. URL <https://www.freecodecamp.org/news/sudo-apt-get-update-vs-upgrade-what-is-the-difference/>. Zuletzt besucht: 2022.08.01.
- [15] Docker. Docker install on ubuntu, 2022. URL <https://docs.docker.com/engine/install/ubuntu/>. Zuletzt besucht: 2022.08.01.
- [16] Home Assistant. Using home assistant supervised os-agent, 2022. URL <https://github.com/home-assistant/os-agent>. Zuletzt besucht: 2022.08.01.
- [17] Tech With Tim. How to deploy a react app - using nginx and linux, 2022. URL <https://www.youtube.com/watch?v=KFwFDZpEzXY>. Zuletzt besucht: 2022.08.09.
- [18] InfluxDB. Install influxdb - linux, 2022. URL <https://docs.influxdata.com/influxdb/v2.3/install/?t=Linux>. Zuletzt besucht: 2022.08.11.
- [19] Fritzling. Fritzling software, 2022. URL <https://fritzling.org/>. Zuletzt besucht: 2022.08.14.
- [20] AZ-Delivery. Esp-32 pinout az-delivery, 2022. URL https://cdn.shopify.com/s/files/1/1509/1638/files/ESP-32_NodeMCU_Developmentboard_Pinout.pdf?v=1609851295. Zuletzt besucht: 2022.08.02.
- [21] randomnerdtutorials. Esp32 pinout reference: Which gpio pins should you use?, 2022. URL <https://randomnerdtutorials.com/esp32-pinout-reference-gpios/>. Zuletzt besucht: 2022.08.14.
- [22] Az-Delivery. Dht22 adruion schematics 1, 2022. URL https://cdn.shopify.com/s/files/1/1509/1638/files/DHT_22_-_AM2302_Temperatur-_und_Luftfeuchtigkeitssensor_Arduino_Schematics_1.pdf?9648280433239966886. Zuletzt besucht: 2022.08.14.
- [23] Umweltbundesamt. Emission von feinstaub der partikelgröße pm2,5, 2022. URL <https://www.umweltbundesamt.de/daten/luft/luftschadstoff-emissionen-in-deutschland/>

- emission-von-feinstaub-der-partikelgroesse-pm25#emissionsentwicklung.
Zuletzt besucht: 2022.08.14.
- [24] randomnerdtutorials. Esp32 with bme280 sensor using arduino ide (pressure, temperature, humidity), 2022. URL <https://randomnerdtutorials.com/esp32-bme280-arduino-ide-pressure-temperature-humidity/>. Zuletzt besucht: 2022.08.14.
- [25] ESPHome. Generic esp32, 2022. URL <https://www.esphome.io/devices/esp32.html>. Zuletzt besucht: 2022.08.15.
- [26] ESPHome. Dht temperature+humidity sensor, 2022. URL <https://www.esphome.io/components/sensor/dht.html>. Zuletzt besucht: 2022.08.15.
- [27] ESPHome. Sds011 particulate matter sensor, 2022. URL <https://www.esphome.io/components/sensor/sds011.html>. Zuletzt besucht: 2022.08.15.
- [28] ESPHome. Bme280 temperature+pressure+humidity sensor, 2022. URL <https://www.esphome.io/components/sensor/bme280.html>. Zuletzt besucht: 2022.08.15.
- [29] ESPHome. Mh-z19 co2 and temperature sensor, 2022. URL <https://www.esphome.io/components/sensor/mhz19.html>. Zuletzt besucht: 2022.08.15.
- [30] mdn. Javascript datentypen und datenstrukturen, 2022. URL https://developer.mozilla.org/de/docs/Web/JavaScript/Data_structures. Zuletzt besucht: 2022.08.05.
- [31] Frank W. *Berichte aus der Bauforschung – Raumklima und Thermische Behaglichkeit*. Ernst und Sohn KG, Berlin-München-Düsseldorf, 1975.
- [32] Christian Baars. Who verschärft empfehlungen massiv, 2022. URL <https://www.tagesschau.de/investigativ/ndr/who-luftverschmutzung-111.html>. Zuletzt besucht: 2022.08.21.
- [33] Nivo. nivo provides a rich set of dataviz components, built on top of d3 and react, 2022. URL <https://nivo.rocks/>. Zuletzt besucht: 2022.08.24.
- [34] Microsoft. Impliziter oauth 2.0-gewährungsablauf, 2022. URL <https://docs.microsoft.com/de-de/azure/active-directory/develop/v2-oauth2-implicit-grant-flow>. Zuletzt besucht: 2022.08.19.

- [35] Reactrouter. Main concepts - routs, 2022. URL <https://reactrouter.com/docs/en/v6/getting-started/concepts#main-concepts>. Zuletzt besucht: 2022.08.23.
- [36] gitforwindows. Git bash, 2022. URL <https://gitforwindows.org/>. Zuletzt besucht: 2022.08.24.