Fachhochschule Nordwestschweiz
Hochschule für Technik

cde2 - Smart Classroom Challenge Bericht

Etienne Roulet, Florin Barbisch, Gabriel Torres Gamez, Yvo Keller

GitHub: https://github.com/Smart-Classroom-Challenge/Abgabe

System Demo Video: https://www.youtube.com/watch?v=7WmZn0B6uVw

Fachhochschule Nordwestschweiz, Hochschule für Technik, BSc Data Science
Windisch, FS22

Abstract

Schlagwörter: CO_2, Klassenzimmer, Schule, Luftqualität, Luftfeuchtigkeit, IoT, MQTT, Django, Bluetooth, Sensoren, Sensordaten, Datenanalyse, Data Science, Gesundheit, Konzentration

Seit des Beginns der COVID-19 Pandemie wird von Experten empfohlen, regelmässig zu Lüften. Für eine Einordnung der Lage an öffentlichen Volksschulen hinsichtlich der Einwirkung des Lüftens auf die Luftqualität in den Klassenzimmern, greifen wir auf einen Sensor gestützte Datenanalyse zu.

Diese Arbeit dediziert sich der Beantwortung der folgenden Fragen:

- Wie ist der Zusammenhang zwischen Luftqualität und Frequenz bzw. Dauer des Lüftens?
- Lässt sich anhand der Luftqualität etwas über die Anzahl Personen im Raum aussagen?
- Lässt sich aus den Messdaten der optimale Zeitpunkt und Dauer zum Lüften (laufend) ermitteln?
- Welche weiteren Erkenntnisse lassen sich aus den erfassten Messdaten ableiten?

Um die Daten zu sammeln, die für die Beantwortung der Forschungsfragen benötigt werden, wurden in der Primar- und Sekundarschule Bettlach im Kanton Solothurn (CH) in drei Klassenzimmern Sensoren montiert, welche verschiedene Sensordaten in regelmässigen Zeitabständen von 10 Sekunden messen. Resultat ist eine Serverlandschaft mit IoT Messgeräten, Backend System, Datenbank und Frontend Konsumenten für Monitoring und Analytics.

Die Auswertung zeigt, dass Stosslüften (3 bis 5 Minuten) effektiver als kontinuierliches Lüften ist, um einen tiefen CO2 Wert in den Klassenzimmern zu erhalten. Das Öffnen der Tür verbessert den Effekt, den das Lüften hat, zusätzlich. Auf Basis der Anzahl der Personen und der Zimmergrösse lässt sich laufend berechnen, wann gelüftet werden muss. Bei einer Klasse mit 20 Schülern in einem Klassenzimmer mit 201 m^3 Volumen wird das Lüften beispielsweise alle 30 Minuten notwendig.

Sensibilisierung der Schulen auf die Luftqualität in den Schulzimmern ist zu empfehlen, da der CO_2 Wert in den Schulzimmern öfters den empfohlenen Wert von 1400 ppm übersteigt.

Zusätzliche Arbeiten

Diese Arbeit wurde anstatt als zweiter oder dreier Gruppe, als vierer Gruppe geschrieben und umgesetzt. Dazu wurde der folgende Mehraufwand betrieben: Es wurde ein Django MQTT Backend implementiert, dass es verschiedenen Datenproduzenten ermöglicht, Nachrichten bidirektional auf einer Zeitreihen Datenbank zu persistieren. Wir haben weiter Grafana für Monitoring sowie Alerts eingerichtet, und darin Dashboards für alle Messgeräte konfiguriert. Mit dem Ziel, dass auch Datenkonsumenten mit demselben Protokoll kommunizieren. Um diesen Mehrwert zu verdeutlichen, wurde ein clientseitiges Frontend geschrieben, das mit MQTT die Daten konsumiert und das Management der IoT Messstationen ermöglicht. Die Serverlandschaft wurde opportun mit Docker virtualisiert. Dabei wurde Wert auf Performance und Erreichbarkeit gelegt.

Bilder

Die Bilder sind jeweils in der Bildbeschriftung für eine höhere Auflösung verlinkt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Systemarchitektur

1.1 System Architektur Diagramm

Die Architektur unseres Systems ist dem untenstehenden Diagramm zu entnehmen. Die einzelnen Komponenten werden auf den nachfolgenden Seiten des Berichts genauer erklärt.

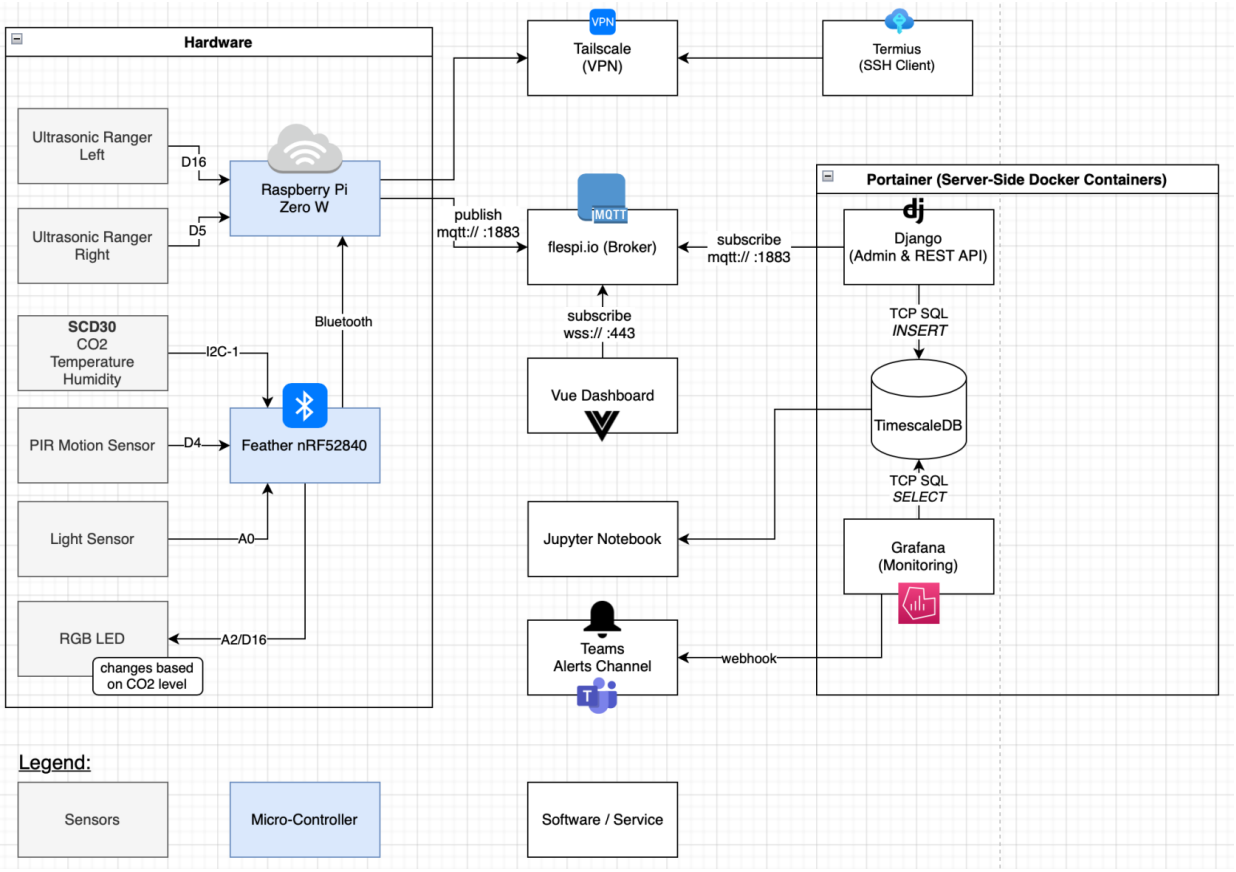


Abbildung 1 System Architektur Diagramm

1.2 Hosting

Als Hoster wurde Netcup.de evaluiert. Die Evaluationskriterien waren (geordnet nach Priorität): Root/VServer, Preis-Leistung, Erreichbarkeit. Entschieden haben wir uns für einen Root Server mit 4 AMD EPYC™ 7702 Prozessorkerne und 16 GB Arbeitsspeicher. Durch die zukünftig breite Systemlandschaft entschieden wir uns für eine Docker Containerisierung. Diese ist über einen Reverse Proxy (Nginx) per TLS ansprechbar. Die Docker Container werden jeweils mit dem Container Management GUI «Portainer» (https://www.portainer.io/) verwaltet.

1.3 Docker

Die untenstehende Tabelle beschreibt unsere Docker System Landschaft.

Table 1: Docker System Landschaft

Name	Docker Hub Image	Docker Port / Port	Public Domain
Grafana	grafana/grafana-oss:latest	3000:3000	grafana.roulet.dev
TimeScale	timescale/timescaledb:latest-pg14	5432:5432	timescale.roulet.dev
smc-django-backend	itsfrdm/smc-django-backend:latest	8888:8000	django.roulet.dev
Portainer	portainer/portainer-ce:2.11.0	8000:8000 9000:9000 9443:9443	portainer.roulet.dev
nginxproxy2_app_1 & nginxproxy2_db_1	jc21/nginx-proxy-manager:latest	443:443 80:80 81:81	-

1.4 MQTT

Sequenzielle Darstellung der MQTT Kommunikation zwischen den Komponenten mit Persistenzschicht.

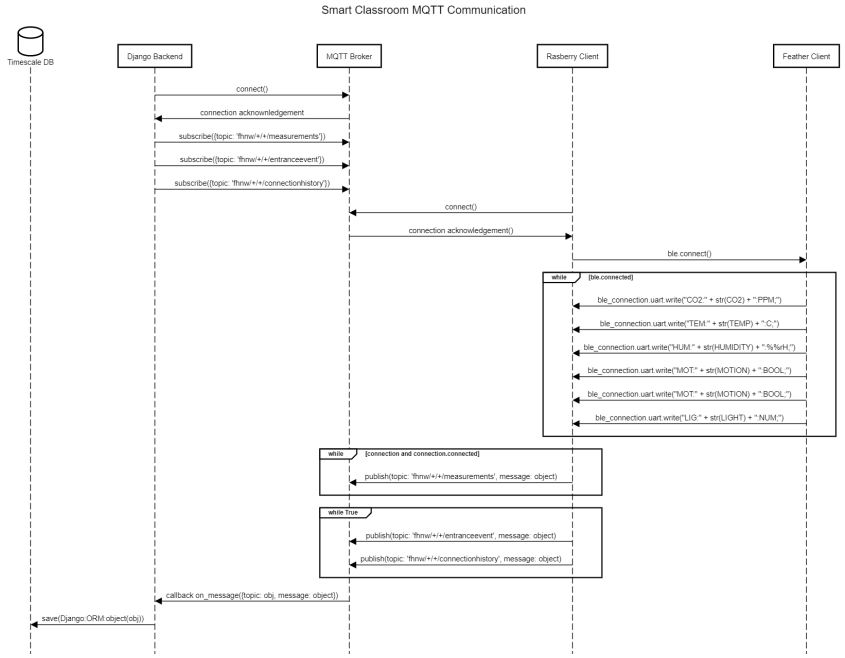


Abbildung 2 MQTT Kommunikation Sequenzdiagramm

2 Hardware

In diesem Kapitel zeigen wir, welche Instrumente wir vor Ort nutzen, um unsere Daten zu erheben.

2.1 Sensoren

Um möglichst viele Daten zu sammeln, verwenden wir verschiedene Sensoren.

2.1.1 Feather

Auf unseren Feathers haben wir ein CO_2-Messer, ein Temperaturmesser, ein Luftfeuchtigkeitsmesser, ein Bewegungsmelder, ein Lichtsensor und eine Status LED. Die Status LED wird verwendet, um auf den CO_2-Gehalt der Luft aufmerksam zu machen. Die anderen Sensoren erheben nur Daten.

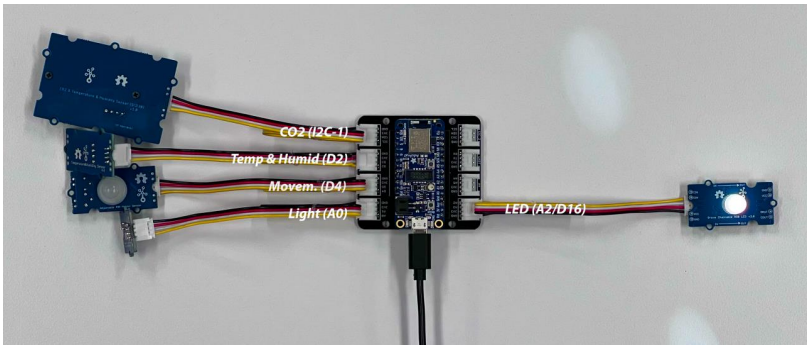


Abbildung 3 Feather mit verbundenen Sensoren

2.1.2 Raspberry Pi

Auf unseren Raspberry Pis haben wir zwei Ultrasonic Range Sensoren. Diese messen die Distanz zwischen einem Objekt und dem Sensor mit Schallwellen. Diese verwenden wir, um herauszufinden, ob eine Person das Schulzimmer betreten oder verlassen hat. Sie werden im Gehäuse (unten erklärt) mit ca. 12 cm Abstand montiert. Falls der Sensor näher am Ausgang als Erstes erkennt, dass eine Person davorsteht und danach das Gleiche mit dem anderen Sensor passiert, zählen wir, dass jemand das Zimmer betreten hat. Falls der andere Sensor aber zuerst bemerkt, dass eine Person davorsteht, und dann der andere, dann zählen wir, dass jemand das Zimmer verlassen hat.

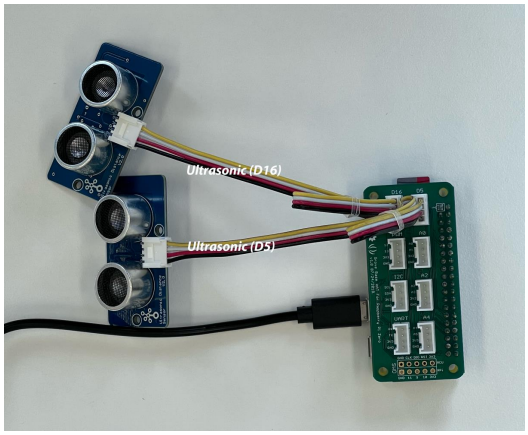


Abbildung 4 Raspberry Pi mit Ultraschall Sensoren

2.2 Stromversorgung

Um unsere Raspberry Pi's und unsere Feathers mit Strom zu versorgen, benutzen wir die integrierten Micro-USB Ports. Diese Kabel hängen wir an alte Handy-Netzteile, da diese mehr als genug Strom liefern können. Mit der Hilfe von Verlängerungskabel sind unsere Geräte an den geeigneten Stellen montiert worden.

2.3 Module

Um unsere Sensoren mit unseren Raspberry Pis und unseren Feathers zu verbinden, verwenden wir Shields. Für den Raspberry verwenden wir den «Grove Base Hat for Raspberry Pi Zero» und für die Feathers verwenden wir den «Grove Shield for Particle Mesh».

2.4 Gehäuse

Damit die Sensoren während dem Feldeinsatz keinen Schaden nehmen, haben wir zwei Gehäuse mit Fusion360 konstruiert und mit einem 3D Drucker ausgedruckt. Die Gehäuse lassen sich mithilfe eines Klickmechanismus schliessen. Für die Sensoren haben wir Aussparungen gemacht, sodass die Sensoren im Gehäuse festgeklemmt werden können. Auch für die Kabel gibt es im Gehäuse Löcher. Die Gehäuse haben wir mithilfe von Klebestreifen an der Wand oder dem Türrahmen befestigt. Der Klebestreifen von Tesa lässt sich ohne Spuren problemlos wieder entfernen.

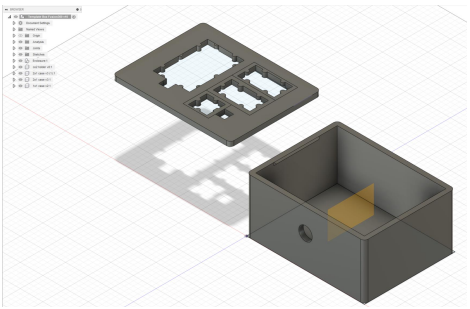


Abbildung 5 3D Modell des Gehäuses des Featherwings

3 Software

In diesem Kapitel betrachten wir die verschiedenen Softwarekomponenten, auf der unser Smart Classroom System basiert. Grundsätzlich ist zu unterscheiden zwischen Software, die wir selbst entwickelt haben, und Open Source/Free Plan Tools, die wir einsetzen. Diese wiederum laufen entweder auf unserem Server in einem Docker Container, oder werden direkt in der Cloud genutzt.

3.1 Feather Client

Unsere Feathers haben als Aufgabe, Sensordaten an den Raspberry Pi zu senden und gleichzeitig anhand einer RGB-LED auf schlechte Luftqualität (hoher CO_2 Wert) zu alarmieren. Wir nutzen Bluetooth Low Energy mit UART, um unsere Daten an den Raspberry zu senden.

3.2 Raspberry Pi Client

Unsere Raspberry Pis haben als auf Aufgabe, Personeneintritte und Austritte am Eingang zu messen und diese Daten mit den Daten der Feather Clients an den MQTT Broker zu publizieren. Damit wir alles gleichzeitig machen können und keine Sensoren blockieren, benutzen wir Multi Threading. Um den Absturz eines Threads zu verhindern, stellen wir das Error Handling sicher und schreiben jeden Fehler in ein Log-File.

3.3 Django Backend

Ein mit Python implementiertes Django Backend (https://django.roulet.dev/) erfüllt in unserem System mehrere Aufgaben:

- Wildcard Subscription auf die relevanten MQTT Topics, wobei neue Daten laufend in die TimescaleDB geschrieben werden
- Django ORM - Das Datenmodell ist hier spezifiziert und wird in der Datenbank über das Django CLI Tool mittels der Migration-Anwendung aktuell gehalten
- Das Django Admin Panel erlaubt per Browser Endpunkt /admin die Verwaltung der persistenten Daten
- Eine RESTful API stellt CRUD Endpoints für alle unsere Entitäten zur Verfügung (Classroom, MeasurementStation, Measurement, ConnectionHistory, EntranceEvent)

3.4 Flespi.io

Wir setzen den Free Plan des MQTT Cloud Broker Flespi (https://flespi.io/) ein, um Sensordaten nach dem Publish/Subscribe Verfahren mittels Topics real-time zur Verfügung zu stellen. Die Verwendung von MQTT erlaubt allen unseren Raspberries Daten zu senden (publish), und dem Django Backend sowie dem Vue Dashboard diese zu empfangen (subscribe).

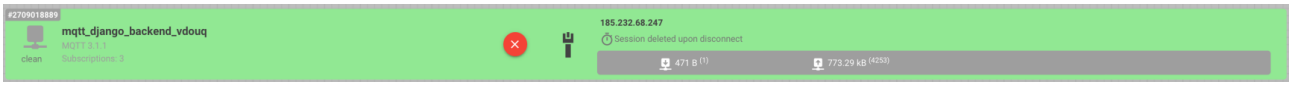


Abbildung 6 Die Subscriptions des Django Backend auf drei Topics auf dem MQTT Broker

3.5 TimescaleDB

Wir verwenden als Datenbank TimescaleDB. Diese auf PostgreSQL basierende Time Series Datenbank wurde speziell dafür entwickelt, grosse Volumen von Time Series Daten effizient zu verwalten, und unterstützt auch horizontale Skalierung. Sie wird von vielen grossen Tech-Konzernen wie Apple und Tesla eingesetzt. Die TimescaleDB läuft auf unserem Server und basiert auf dem offiziellen Docker Image: timescale/timescaledb

3.6 Grafana

Grafana (https://grafana.roulet.dev/) ist unsere Monitoring Software, und kann auch als das Herz unseres Systems betrachtet werden. Alle Daten werden direkt aus der TimescaleDB bezogen. Die Dashboards, die wir hier konfiguriert haben, erlauben es uns, die Time Series Daten von unseren Messgeräten in Echtzeit zu überwachen und auszuwerten. Wird ein Problem festgestellt, kommen zum Beispiel keine neuen Daten mehr, wird eine entsprechende Meldung an einen Teams Channel gesendet, und wir erhalten alle eine Push-Benachrichtigung. Wir können dann entsprechend reagieren und das Problem identifizieren & beheben.

Abbildung 7 Die Daten im Primarschul-Klassenzimmer auf dem Grafana Dashboard

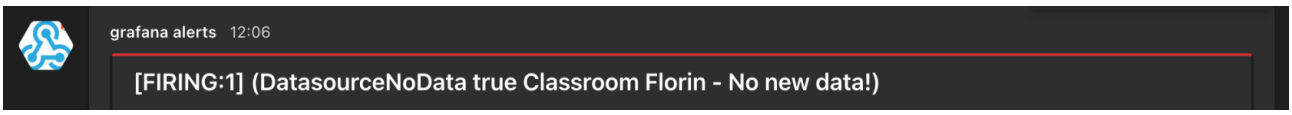


Abbildung 8 Ein Grafana Alert im Teams Channel bei ausbleibenden Daten

Grafana läuft auf unserem Server und basiert auf dem offiziellen Docker Image: «grafana/grafana»

3.7 Vue.js MQTT Frontend

Um die Einsatzzwecke von MQTT aufzuzeigen, wurde zusätzlich ein JavaScript Frontend gebaut, welches auf netlify.com gehostet wird und die Daten live von unserem MQTT Broker bezieht.
Die Applikation ist hier verfügbar: https://smartclassroom.netlify.app.

Table 2: Frontend Routes

Route	Beschreibung	Controller
./classrooms	Listet alle Klassenräume auf	Pages >> Classrooms >> index.vue
./classrooms/all/	Listet alle Messstationen auf	Pages >> Classrooms >> all.vue
./classrooms/<classroom>/	Listet alle Messstationen des Klassenraums auf	Pages >> Classrooms >> id >> index.vue
./classrooms/all/DetailsView/<id> **./classrooms/<id>/DetailsView/<id>**	MQTT Nachrichten Diagramme und Message Protokoll	Pages >> Classrooms >> index.vue & >> DetailsView >> [id].vue

3.8 Uptime Robot

Uptime Robot ist ein Monitoring Service, der die Uptime unseres zentralen Servers https://roulet.dev laufend überprüft. Sollte unser Server Offline gehen und dadurch auch die Grafana Alerts nicht mehr funktionieren, erhalten wir einen Alert in einem entsprechenden Teams Channel mit Push Benachrichtigung.

3.9 Tailscale

Tailscale (https://tailscale.com) stellt sicher, dass die Verbindung zu den Raspberry Pi's gewährleistet ist, auch wenn diese sich in einem anderen Netzwerk, wie z.B. an einer Schule, befinden. Die Software ist auf jedem Raspberry Pi installiert, und diese verbinden sich beim Start automatisch mit dem VPN Service. So ist es möglich, jederzeit eine Verbindung aufzubauen und Fehleranalysen zu machen oder einen Neustart auszulösen.

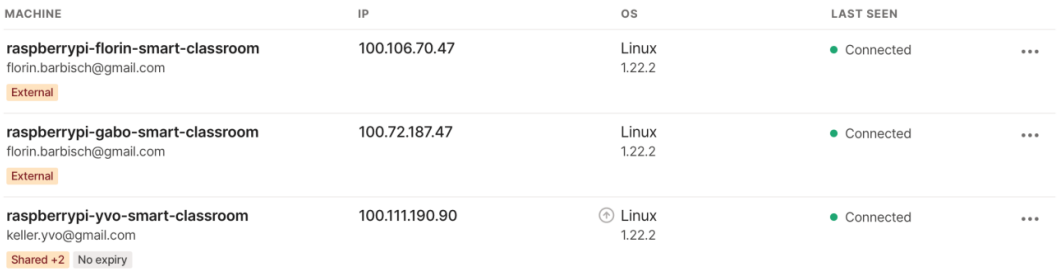


Abbildung 9 Die mit Tailscale verbundenen Raspberry Pi

3.10 Termius

Termius erleichtert uns als SSH Client die Verbindung mit unseren Raspberry Pi. Besteht eine Verbindung zum Tailscale VPN vom Endgerät aus (Handy/Laptop), kann eine SSH Session gestartet werden. Hinterlegte Kurzbefehle ermöglichen das schnelle Ausführen von immer wieder verwendeten Befehlen wie einem Neustart oder der Anzeige von Log Dateien.

Abbildung 10 Die Raspberry Hosts in Termius

4 Datenbank

Wir haben uns aus mehreren Gründen für die Verwendung von TimescaleDB entschieden:

- Baut auf PostgreSQL auf
- Ist speziell für riesige Time Series Datenmengen ausgelegt – wir werden das Potenzial mit unseren Datenmengen nicht mal ansatzweise vollständig ausnutzen können.
- Horizontal skalierbar und distributierbar mit der Nutzung von Hypertables
- ORM Integration zu Django verfügbar

In der Konfiguration haben wir uns an die Standardeinstellungen gehalten. Spezielle Beachtung beim Einsatz von TimescaleDB braucht die Einstellung des Time Intervalls auf Hypertables. TimescaleDB ermöglicht die Verwendung von normalen PostgreSQL Tabellen für relationale Daten, und sogenannte Hypertables für alle Time Series Daten. Hypertables werden in Table Partitions aufgeteilt, nach dem definierten Time Intervall. Wir haben uns bei allen Hypertables für einen Time Intervall von 7 Tagen entschieden. Das bedeutet, alle 7 Tage wird eine neue Paritioned-Table erstellt, in der die Daten dann während einem Zeitraum von 7 Tagen abgelegt werden, bevor eine neue Partition erstellt wird. Diese Einstellung macht sich bei der Performance mit grossen Datenvolumen erkennbar, hat aber bei Abfragen keinen Nachteil, da die Paritioned Tables automatisch aggregiert werden, wenn eine Abfrage über einen Zeitraum von mehreren Partitioned Tables geht.

Unsere Entitäten sind in Django durch Model Klassen wie diese spezifiziert.

```
class EntranceEvent(TimescaleModel):
    fk_measurement_station = models.ForeignKey(MeasurementStation, on_delete=models.CASCADE)
    change = models.IntegerField()
    insert_time = TimescaleDateTimeField(interval="7 days")
```

Abbildung 11 Das Model für "EntranceEvent"

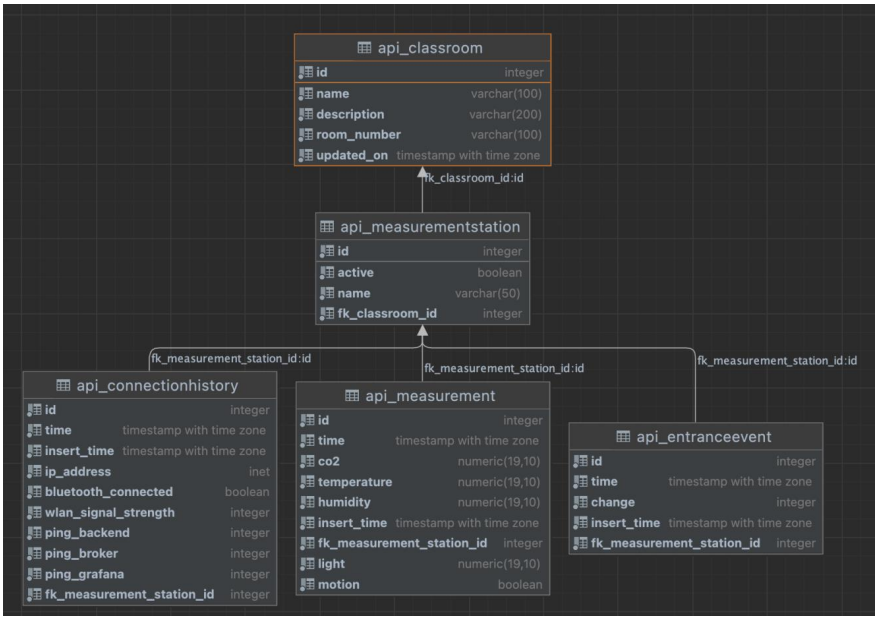


Abbildung 12 TimescaleDB Datenmodell für die Smart Classroom Challenge

5 Methode der Datenerhebung

Die Daten wurden in drei verschiedenen Klassenzimmern erhoben. Zwei davon waren Zimmer von Primarklassen, eines davon eine Oberstufe, bei der sich nicht immer die ganze Klasse im Zimmer befand. Die Daten wurden teils manuell erhoben und teils automatisch.

Zu den **manuell** erhobenen Daten zählen Daten wie die Raumgrösse, oder wie viel Personen zu einem bestimmten Zeitpunkt anwesend waren. Die Raumgrösse wurde von Hand vermessen und zur Berechnung der Anzahl Personen wurde der Stundenplan und die Klassengrösse der Schule verwendet.

Die **automatisch** erfassten Daten lassen sich in zwei Kategorien unterteilen:

- Es wurden Daten zu den Eigenschaften des Zimmers erfasst. Dazu gehören der CO_2-Wert, Luftfeuchtigkeit, Temperatur, Bewegung und Lichtstärke. Diese Daten wurden mithilfe von Sensoren an einem Mikrokontroller erfasst und direkt via Bluetooth an einen Raspberry Pi weitergeleitet. Auf dem Raspberry Pi wurden die Daten geloggt und via MQTT weiter an einen Broker versandt.
- Auf dem Raspberry Pi wurden des weiteren Metadaten (WLAN-Stärke, Ping-Time, Bluetooth-Verbindung, IP-Adresse) auch geloggt und via MQTT versandt.

Experimentell haben wir versucht, mit Hilfe von zwei Ultraschallsensoren die Personenzahlveränderung im Zimmer zu registrieren und auch via MQTT zu versenden. Das Zählen hat wegen Ausreissern bei den Sensormessungen leider nicht so verlässlich funktioniert, wie wir uns das vorgestellt hatten. Da wir vor der Installation wussten, dass wir die Anzahl Personen via Stundenpläne ermitteln werden können, haben wir diesen Versuch nicht weiterverfolgt. Der aktuelle Stand ist, dass sobald eine Person den Raum betritt oder verlässt, dies geloggt wird, und via MQTT an den Broker versendet.

Beim Broker hat sich unser Backend (MQTT-Client) registriert. Sobald das Backend neue Daten erhält, speichert es diese in unsere TimescaleDB. Auf die TimescaleDB, macht unser Monitoring- und Visualisierungs-Tool Grafana regelmässig Abfragen und stellt so innert kürzester Zeit fest, wenn Daten nicht gesendet werden. Merkt Grafana, dass keine Daten mehr ankommen, so wird via einem Teams-Channel eine Benachrichtigung an alle Mitglieder dieser Challengegruppe gesendet. Damit nicht jeder das Gefühl hat, die anderen würden sich schon um das Problem kümmern, haben wir eine Person definiert, welche den Unterbruch untersucht und weitere Massnahmen einleitet. Wenn bei der Untersuchung zum Beispiel herauskam, dass die Bluetooth Verbindung unterbrochen ist, so konnten wir eine Person vor Ort informieren, welche die Geräte durch ein und ausstecken neustarten konnte. Diese Person konnte uns auch mitteilen, ob unsere Geräte nicht laufen oder das WLAN ausgefallen ist.

Es wurde wöchentlich ein komplettes Backup der Docker Volumen erstellt. Ausserdem wurde im Falle eines Server Verlustes oder eines Brandes, innerhalb des Schulzimmers, ein PostgreSQL-Dump erstellt. Zusätzlich wurden die Log-Files der MQTT Kommunikation aller Messstationen auf OneDrive hochgeladen.

6 Beantwortung der Forschungsfragen

In diesem Kapitel gehen wir auf die gestellten Forschungsfragen ein. Die detaillierten Methoden, Analysen und Resultate können unseren Jupyter Notebooks entnommen werden.

6.1 Wie ist der Zusammenhang zwischen Luftqualität und Frequenz bzw. Dauer des Lüftens?

Es zeigt sich, dass die Frequenz des Lüftens um einiges wichtiger ist als die Dauer des Lüftens. Je mehr gelüftet wird, desto kleiner ist der Bereich, in dem sich der CO_2-Gehalt bewegt. Was sich ausserdem abzeichnet: Es ist unvorteilhaft, das Fenster länger offen zu halten, anstatt die Fenster in einer schnelleren Frequenz zu öffnen. Beispielsweise in einer Messung sank der CO_2-Gehalt in 173 Sekunden lüften um 917 ppm, von 1792 ppm auf 875 ppm. Innerhalb 1498 Sekunden jedoch nur von 1514 ppm auf 598 ppm. Somit lässt sich sagen, die Luftqualität hängt enorm von der Frequenz des Lüftens ab. Die Auswertung zeigt auch, dass nicht nur das Lüften den CO_2-Gehalt reduziert, sondern auch das Öffnen der Türe zum Schulgang. Eine gute Art zu Lüften ist somit für kurze Zeit (zwischen 3-5 Minuten) alle Türen und Fenster weit aufzureissen. Zur genauen Frequenz respektive zum optimalen Zeitpunkt des Lüftens sind im Kapitel 6.3 konkrete Zahlen zu entnehmen. So wurde der CO_2-Gehalt der Luft im Durchschnitt innerhalb einer Unterrichtsstunde durch das Atmen der Personen im Raum von 615 ppm auf 1009 ppm erhöht.

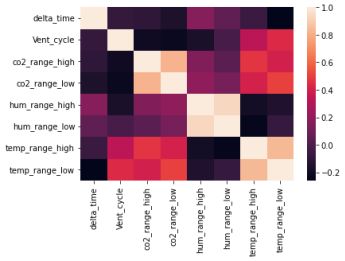


Abbildung 14 Korrelation von Delta Werten bei der Lüftung

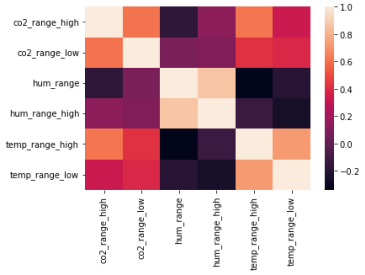


Abbildung 13 Korrelation von Delta Werten zwischen Lüftungen

6.2 Lässt sich anhand der Luftqualität etwas über die Anzahl Personen im Raum aussagen?

Um diese Frage zu beantworten, haben wir bei allen Lektionen die Daten in 5-Minuten Buckets unterteilt. Da wir nicht wissen, wann die Türe/Fenster offen sind, haben wir nur Lektionen untersucht, welche über die Buckets hinweg immer einen Anstieg von CO_2 aufweisen. Mithilfe des Raumvolumens, den Anzahl Personen und unseren CO_2 Werten, konnten wir für jedes Klassenzimmer ausrechnen, wie viel Milligramm CO_2 eine Person pro Minute ausstosst. Die Resultate liegen nahe beieinander, im arithmetischen Mittel sind wir damit auf ein Resultat von 347 mg/Person/Minute gekommen. Das sind am Tag etwa 0.5 kg, und ist die Hälfte davon, was (Palmer, 2015) für erwachsene Personen herausgefunden hat. Wir führen diesen Unterschied auf die körperlich weniger anstrengende Aktivität, sowie das kleinere Lungenvolumen der Kinder zurück. Die folgende Formel lässt sich aus diesem Resultat herleiten, um die Anzahl Personen in einem Raum auszurechnen:

$$Anzahl\ Personen = \frac{CO_2\ Erhöhung\ [ppm] * 1.8 * Volumen[m^3]}{Zeit\ [Minuten] * 347}$$

Als Parameter sind die CO_2 Erhöhung in ppm, das Volumen des Raumes im Kubikmeter sowie die Dauer während der gemessenen Erhöhung in Minuten notwendig.

6.3 Lässt sich aus den Messdaten der optimale Zeitpunkt zum Lüften (laufend) ermitteln?

Aus den Erkenntnissen der letzten Forschungsfrage lässt sich auch der optimale Zeitpunkt zum Lüften in einem Klassenzimmer laufend ermitteln. Wenn das Klassenzimmer gut gelüftet ist, hat es einen CO_2 Wert von 400 ppm. Da bei 1000 ppm die Luft schlecht ist [Pettenkoferzahl (Dentel & Dietrich, 2008)], müssen wir nur ausrechnen, wie lange es dauert, bis wir 600 ppm CO_2 produziert haben. An diesem Zeitpunkt sollte dann wieder gelüftet werden. Wenn wir die Formel von oben umformen, ergibt das diese neue Formel:

$$Zeit\ [Minuten] = \frac{600 * 1.8 * Volumen[m^3]}{Anzahl\ Personen * 347}$$

Für die Klassenzimmer an der Schule in Bettlach gibt das folgende **Faustregeln**:
In der Klasse 5a (unteres Primarzimmer) sollte circa alle 30 Minuten gelüftet werden, in der 4a reicht alle 45 Minuten. Bei Halbklassenunterricht kann diese Zeit verdoppelt werden, und es muss nur noch halb so viel gelüftet werden. Da es im Oberstufenzimmer viele verschiedene Klassengrössen gibt, haben wir hier den optimalen Zeitpunkt zum Lüften für die Klassengrössen 10 Personen (alle 60 Minuten), 15 Personen (alle 40 Minuten) und 20 Personen (alle 30 Minuten) ausgerechnet.

6.4 Welche weiteren Erkenntnisse lassen sich aus den erfassten Messdaten ableiten?

Soll man Lüften, um eine höhere Luftfeuchtigkeit in den Klassenzimmern zu erhalten?

«A RH below 30% can cause irritation of the nasal and bronchial passages.» (Grandjean, 1988) Um Grandjeans (1988) idealen Luftfeuchtigkeitswert zu folgen und somit einen zu tiefen Luftfeuchtigkeitswert zu vermeiden, wollen wir analysieren, wie sich der Luftfeuchtigkeitswert beim Lüften verhaltet, wenn die Luftfeuchtigkeit draussen hoch ist. Bei unseren Observationen konnten wir feststellen, dass die Luftfeuchtigkeit im Klassenzimmer beim Lüften sinkt, obwohl draussen die Luftfeuchtigkeit viel höher ist. Also können wir die Luftfeuchtigkeit nicht durch das Lüften steigen lassen. Den Grund für dieses Verhalten konnten wir nicht ermitteln.

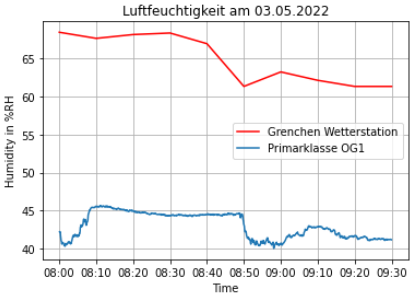


Abbildung 15 Beispiel des Luftfeuchtigkeitsverhältnis beim Lüften um 08:50

Sind Luftfeuchtigkeit und Temperatur in den Klassenzimmern überhaupt gut?

Diese Frage können wir nur in einem bestimmten Rahmen beantworten, da unsere Datenerhebung nur für 3 Wochen im Frühling stattgefunden hat. «A RH above 70% can produce a feeling of “stuffiness.”» (Grandjean, 1988) Nach Grandjean (1988) sollte der RH Wert nicht über 70% und nicht unter 30% liegen. «It was shown that ambient relative humidity is a major factor influencing lifetime of droplets and the distance they may travel. As a consequence, and independently of any other health consideration linked to ambient humidity, it is seen that a dry air is a favourable factor for limiting risk of contamination from COVID19.» (Carlotti et al., 2022) Nach Carlotti et al. (2022) sollte aber für die Verbreitung von COVID so tief wie möglich liegen, also können wir in unserem Fall einschätzen, dass der ideale Luftfeuchtigkeitswert (RH) zwischen 40% und 50% liegen sollte. Wenn wir die Verteilung der Luftfeuchtigkeit in den Klassenzimmern betrachten, sehen wir, dass die Werte der drei Klassenzimmer in einem guten Bereich sind. In der Primarklasse im Obergeschoss war die Luftfeuchtigkeit ab und zu ein wenig tief, jedoch liegt das im akzeptablen Rahmen.

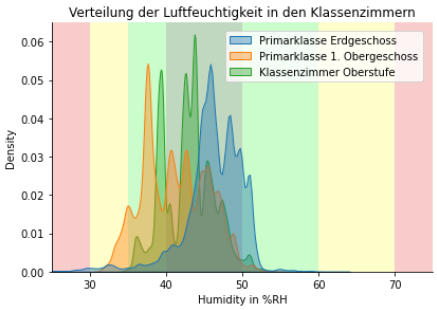


Abbildung 16 Verteilung der Luftfeuchtigkeit in den Klassenzimmern

«The effect of indoor room temperature has more influences than the effect of illumination. The effect of indoor temperature has 38.56% of contribution on the performance. The optimum levels of indoor temperature at 21°C and illumination at 1000 lux have improved the work performance and health of office workers. […] The productivity is one of the most important factors which can affect overall performance of any organization either small or entire nation. The Performance of call center workers has less Performance, when the temperature was above 25°C. Performance has been reduced to 2.4% per degree temperature increase between 21.9°C to 28.5°C.» (Vimalanathan & Ramesh Babu, 2014) Nach Vimalanthan & Ramesh Babu (2014) liegt die ideale Temperatur für die Produktivität bei 21°C. Die Produktivität nimmt dann in Korrelation mit der steigenden Temperatur ab. In den Klassenzimmern lag die Temperatur nach unserer Analyse selten bei 21°C, sondern eher zwischen 22°C und 24°C. Das sollte aber nach Vimalanthan & Ramesh Babu (2014) keinen grossen Einfluss auf die Produktivität der Schüler haben.

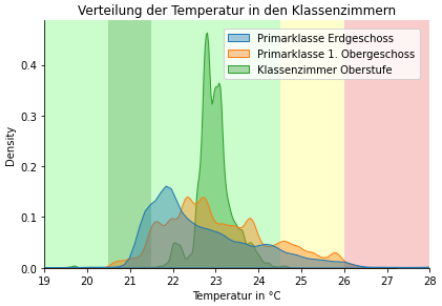


Abbildung 17 Verteilung der Temperatur in den Klassenzimmern

7 Lessons Learned

7.1 Hypertables in TimescaleDB

Zu Beginn der Challenge hatten wir starke Performance Probleme mit der Datenbank. Abfragen auf Messdaten über einen Bereich von einer Stunde dauerten schnell mehrere Minuten, und der ganze Server stiess ressourcentechnisch an die Grenzen. Dies hat uns sehr überrascht, da TimescaleDB doch als Time Series Datenbank speziell performant sein sollte. Nach einiger Nachforschung begannen wir zu verstehen, wie die TimescaleDB Hypertables funktionieren, und fanden das Problem: Wir hatten in unseren Time Series Entitäten einen Time Intervall von 1 Millisekunde gewählt, in der falschen Annahme, dass wir damit die Genauigkeit festlegen. Das hat dazu geführt, dass die Datenbank jeden neuen Measurement Datensatz in einer neu angelegten, eigenen Partitioned Table angelegt hat. Das hat für ganz einfache Queries bedeutet, dass Daten über mehrere tausend Partitioned Tables ausgelesen werden müssen – was das Performance Problem plötzlich offensichtlich erscheinen lässt. Wir haben das Zeitintervall auf 7 Tage angepasst, und seither ist die Datenbank Performance herausragend und wie erwünscht.

7.2 Spannungsabfall beim CO_2 Sensor im Batteriebetrieb führt zu falschen Messungen

Ein Interessantes Phänomen konnten wir beobachten, als wir den Feather im Batteriebetrieb getestet haben. Mit unserer Verkabelung und 3 vollgeladenen AAA-Batterien hat der Strom für ca. 36 Stunden gereicht. Ab diesem Zeitpunkt lieferte der Feather zwar weiter Daten, aber der CO_2 Sensor begann grosse Sprünge in den Messungen zu machen. Wir gehen davon aus, dass das mit einem Spannungsabfall in der Stromversorgung des Sensors zusammenhängt, wodurch dieser nicht mehr richtig funktioniert hat.

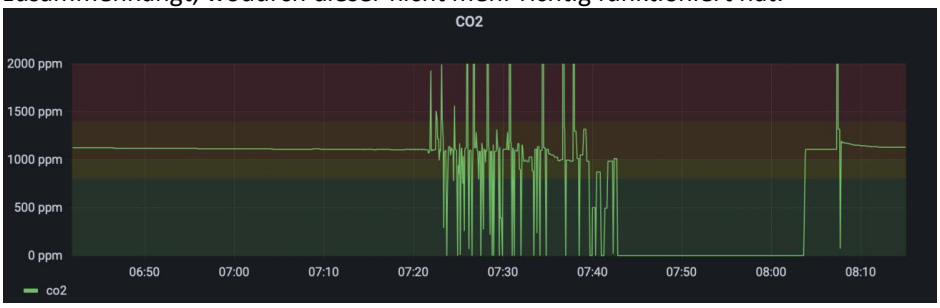


Abbildung 18 Spannungsabfall im CO2 Sensor führt zu falschen Messungen

7.3 Django Channels

Es wird in den Fachkreisen und Fachliteraturen empfohlen, mit Django Channels die MQTT Kommunikation per Socket zu implementieren. Dafür gibt es auch diverse Bibliotheken, unter anderem https://pypi.org/project/django-mqtt-bridge/. Jedoch keine einzige Bibliothek konnte funktionsfähig implementiert werden, weil die meistens Bibliotheken nicht für unsere Versionen kompatibel waren.

7.4 Personenzähler mit Ultraschallsensoren

Wir hatten zuerst die geniale Idee, Personen, welche das Zimmer betreten oder verlassen mithilfe von zwei Ultraschalldistanzsensoren zu registrieren. Schon der erste Prototyp hatte Personen registriert, welche gar nicht da waren. Wir haben dann den Algorithmus zum Personen erkennen mehrmals angepasst. Die Geisterpersonen wurden zwar weniger, aber das Resultat war dennoch nicht brauchbar. Beim Suchen nach einer Lösung sind wir im Wiki des Herstellers fündig geworden. Damit die Distanz gemessen werden kann, braucht der Sensor eine glatte Reflektionsfläche von mindestens 0.5 Quadratmeter (Seeed Technology Co.,Ltd, 2017). Da dies bei Personen leider nicht der Fall ist, ist uns auch klar, wieso unsere Methode nicht funktionieren konnte. Nächstes Mal sollten wir uns früher mit der offiziellen Dokumentation auseinandersetzen.
Da im Verlauf der Challenge klar wurde, dass wir Stundenpläne und Klassengrösse zur Verfügen gestellt bekommen, ist der Personenzähler überflüssig.

7.5 Luftqualität an der Fachhochschule

Bei unseren Testversuchen an der Fachhochschule mussten wir feststellen, dass sich der CO_2-Wert nicht stark verändert. Das ist auf die Lüftung zurückzuführen, welche ununterbrochen läuft und so für tiefe CO_2-Werte sorgt. Wir konnten auch feststellen, dass die Zimmer sehr trocken sind (tiefe Luftfeuchtigkeit).

8 Zitate und Quellen

Carlotti, P., Massoulié, B., Morez, A., Villaret, A., Jing, L., Vrignaud, T., & Pfister, A. (2022). Respiratory pandemic and indoor aeraulics of classrooms. *Building and Environment*, *212*, 108756. https://doi.org/10.1016/j.buildenv.2022.108756

Dentel, A., & Dietrich, U. (2008). *Thermische Behaglichkeit – Komfort in Gebäuden*. 37.

Grandjean, E. (1988). *Fitting the task to the Man: A textbook of occupational ergonomics (4th ed.)*. Taylor & Francis. https://swisscovery.slsp.ch/permalink/41SLSP_FNW/sonqgj/alma991047384749705501

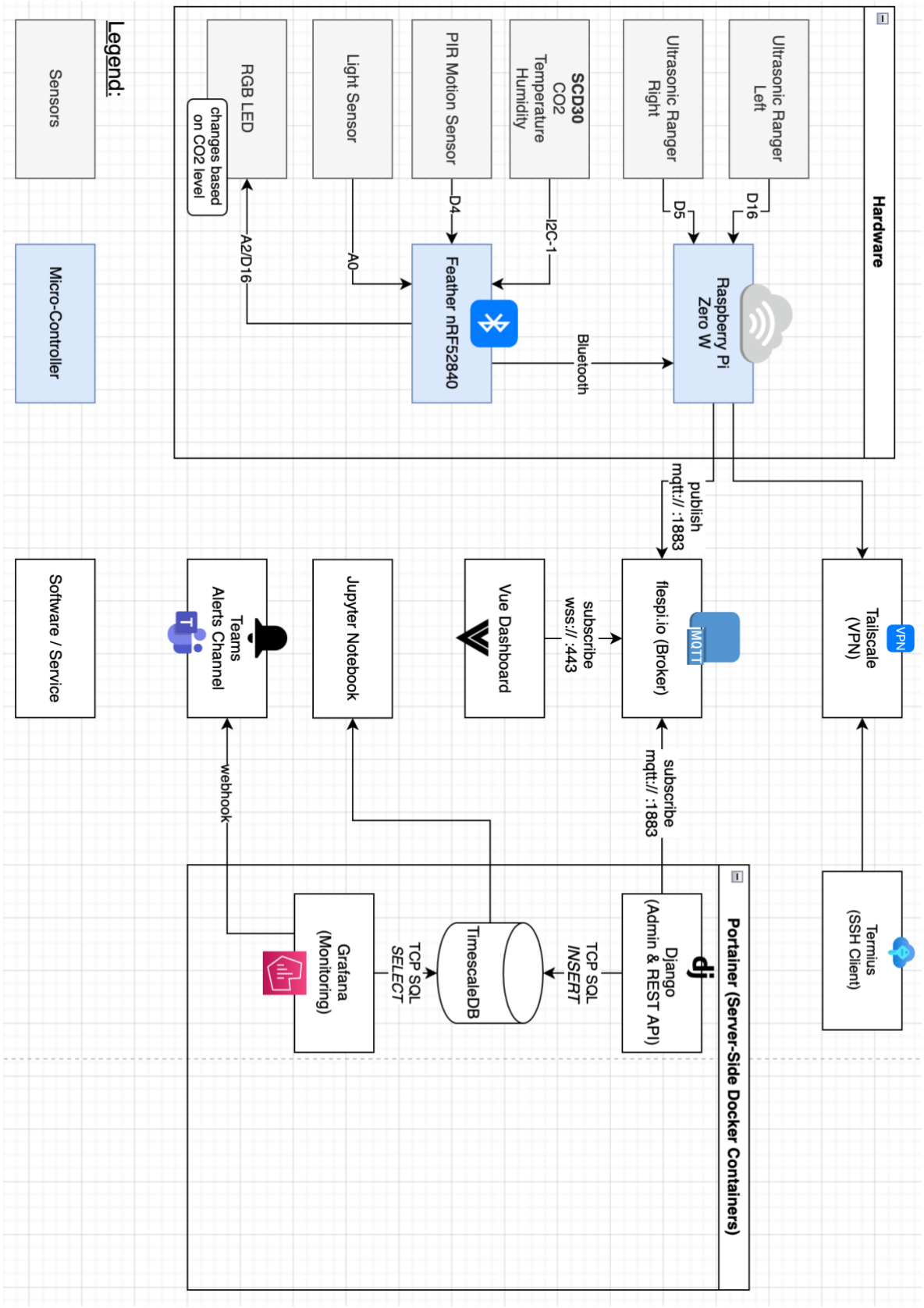
Palmer, B. (2015, May 19). *Do We Exhale Carbon?* NRDC. https://www.nrdc.org/stories/do-we-exhale-carbon

Seeed Technology Co.,Ltd. (2017, July). *Grove—Ultrasonic Ranger* [Wiki]. Wiki.Seeedstudio.Com. https://wiki.seeedstudio.com/Grove-Ultrasonic_Ranger/

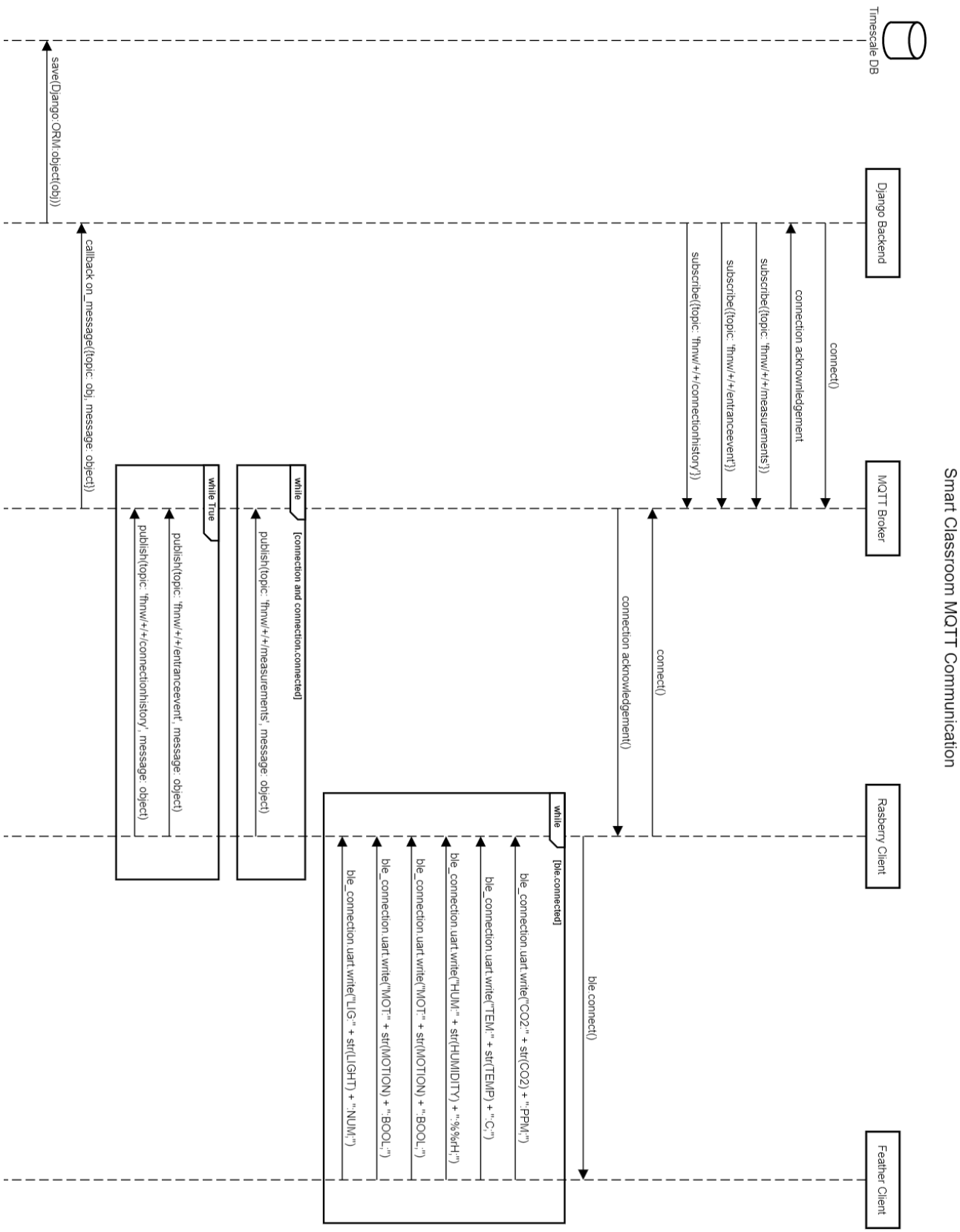
Vimalanathan, K., & Ramesh Babu, T. (2014). The effect of indoor office environment on the work performance, health and well-being of office workers. *Journal of Environmental Health Science and Engineering*, *12*(1), 113. https://doi.org/10.1186/s40201-014-0113-7

Anhang

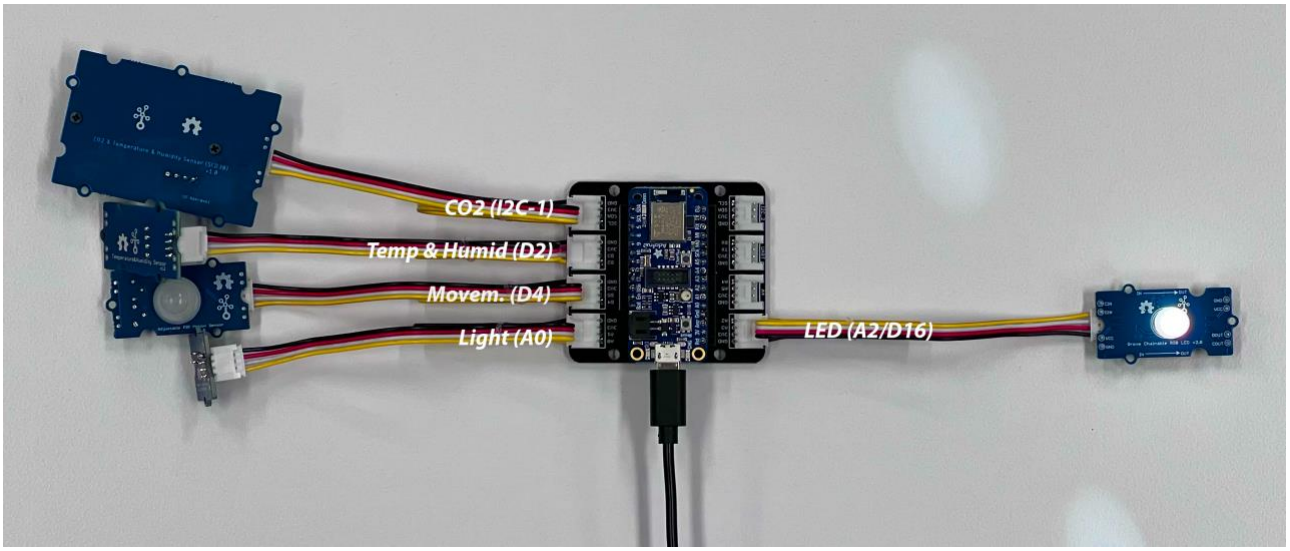
8.1 Abbildung 1 System Architektur Diagramm



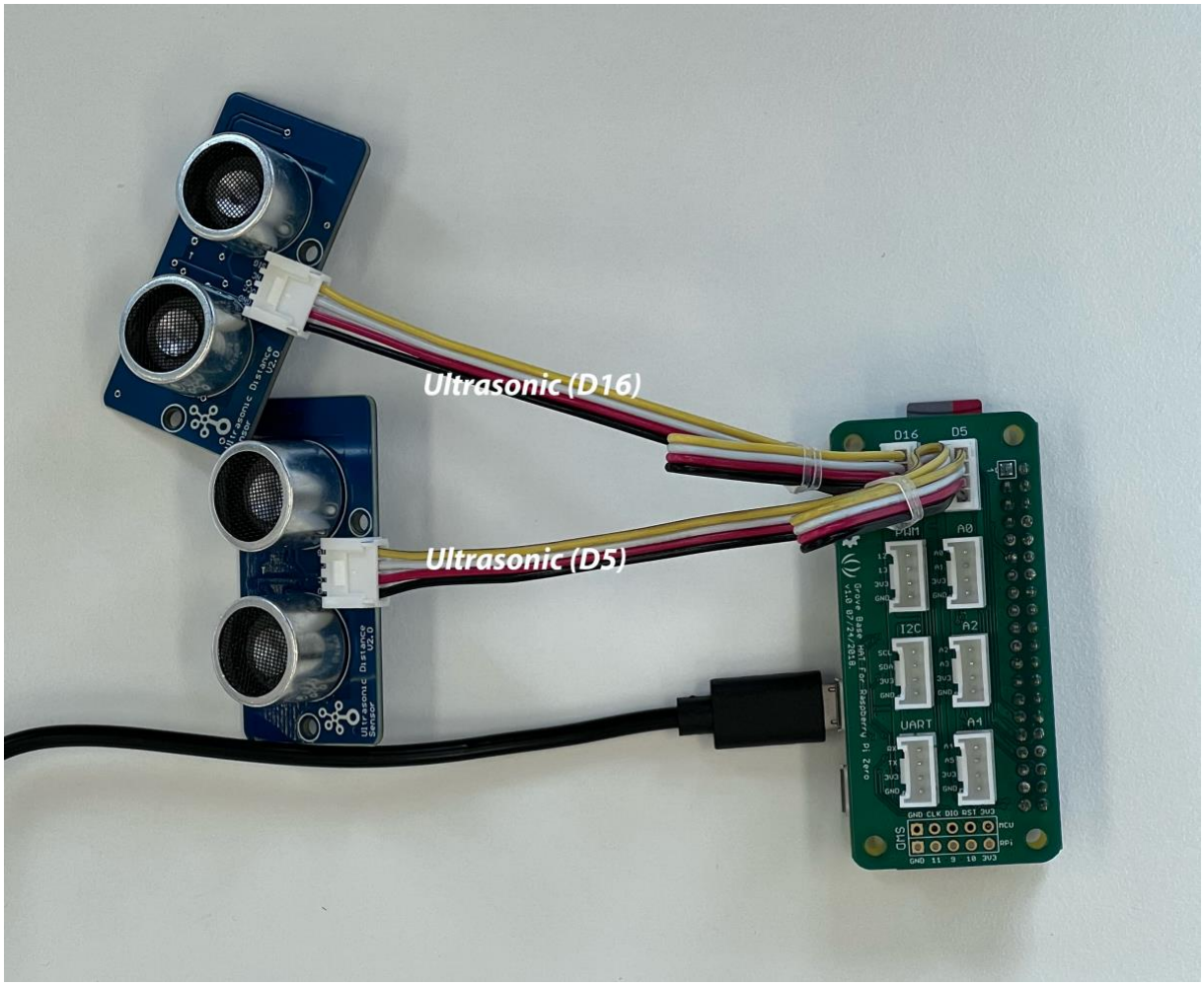
8.2 Abbildung 2 Sequenzdiagram



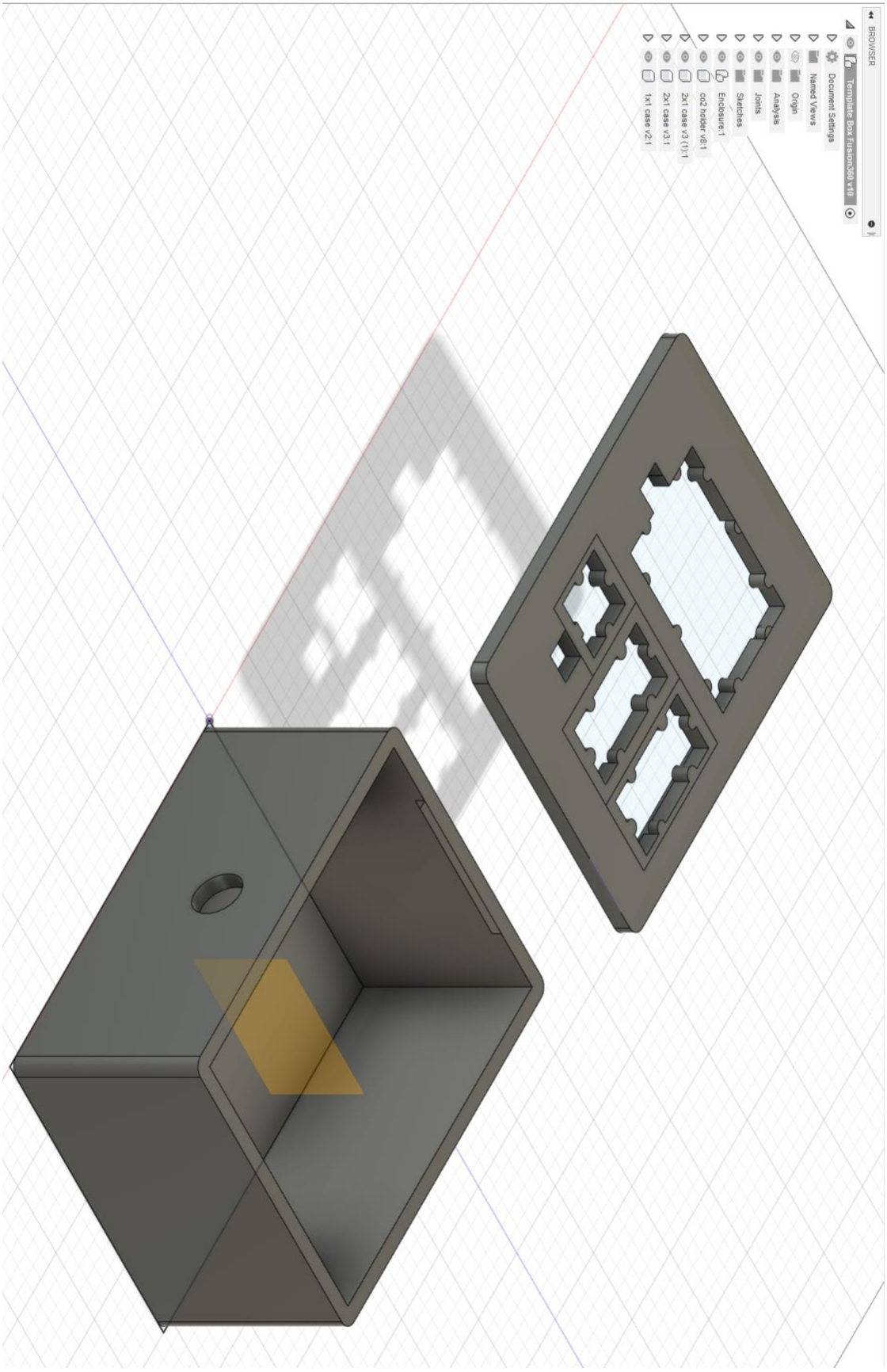
8.3 Abbildung 3 Feather



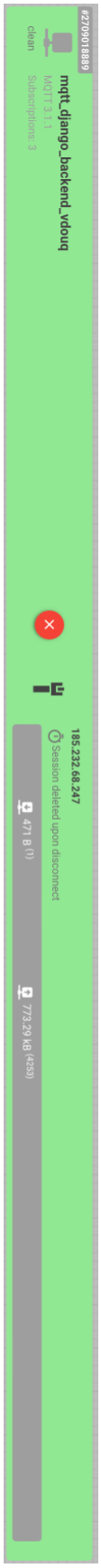
8.4 Abbildung 4 Raspberry Pi



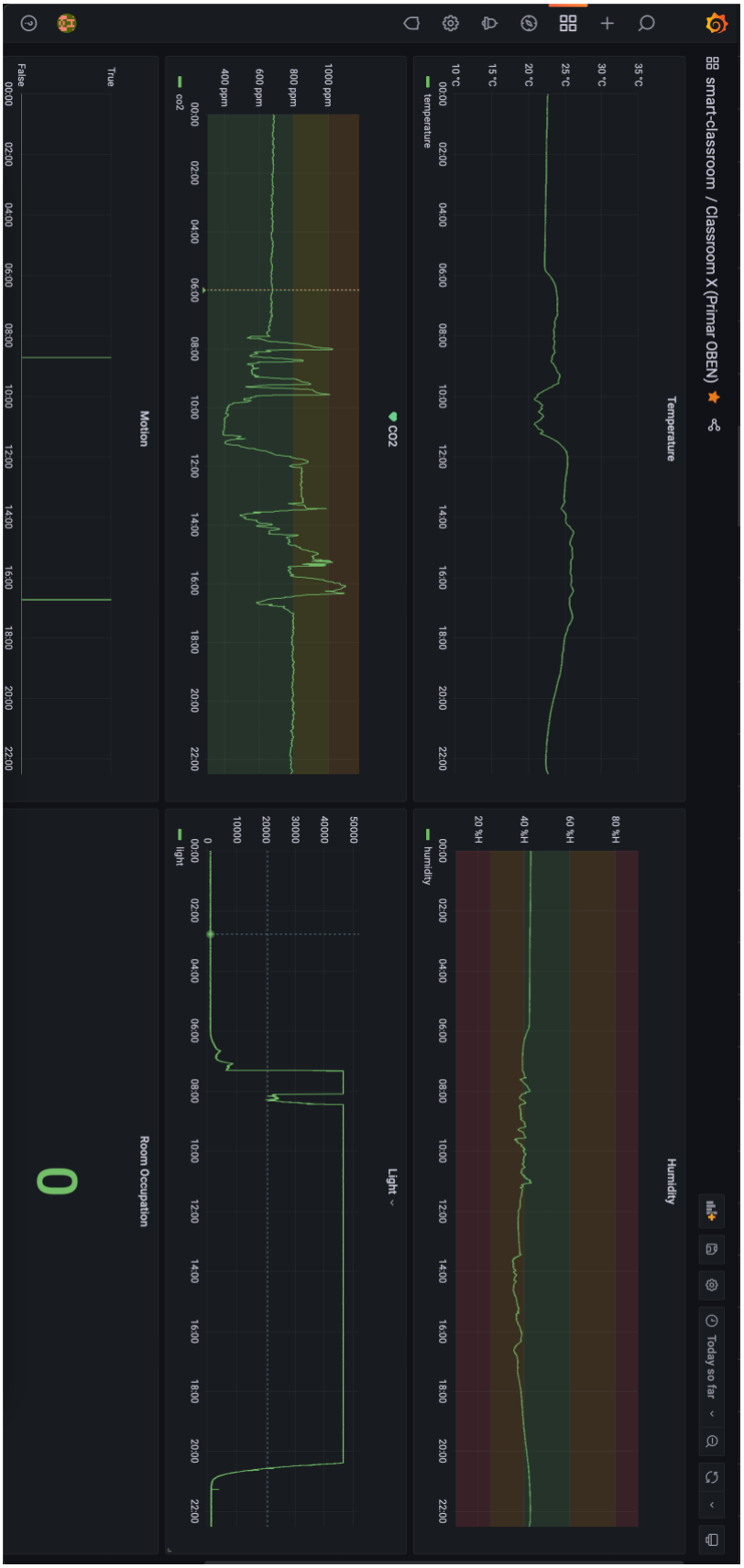
8.5 Abbildung 5 Gehäuse



8.6 Abbildung 6 Flespi.io

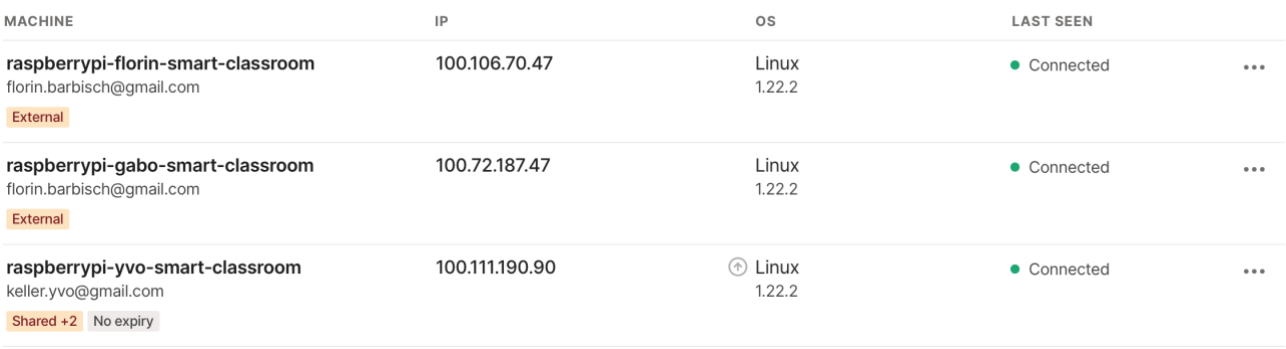


8.7 Abbildung 7 Grafana Dashboard



8.8 Abbildung 8 Grafana Meldung DatasourceNoData

8.9 Abbildung 9 Tailscale



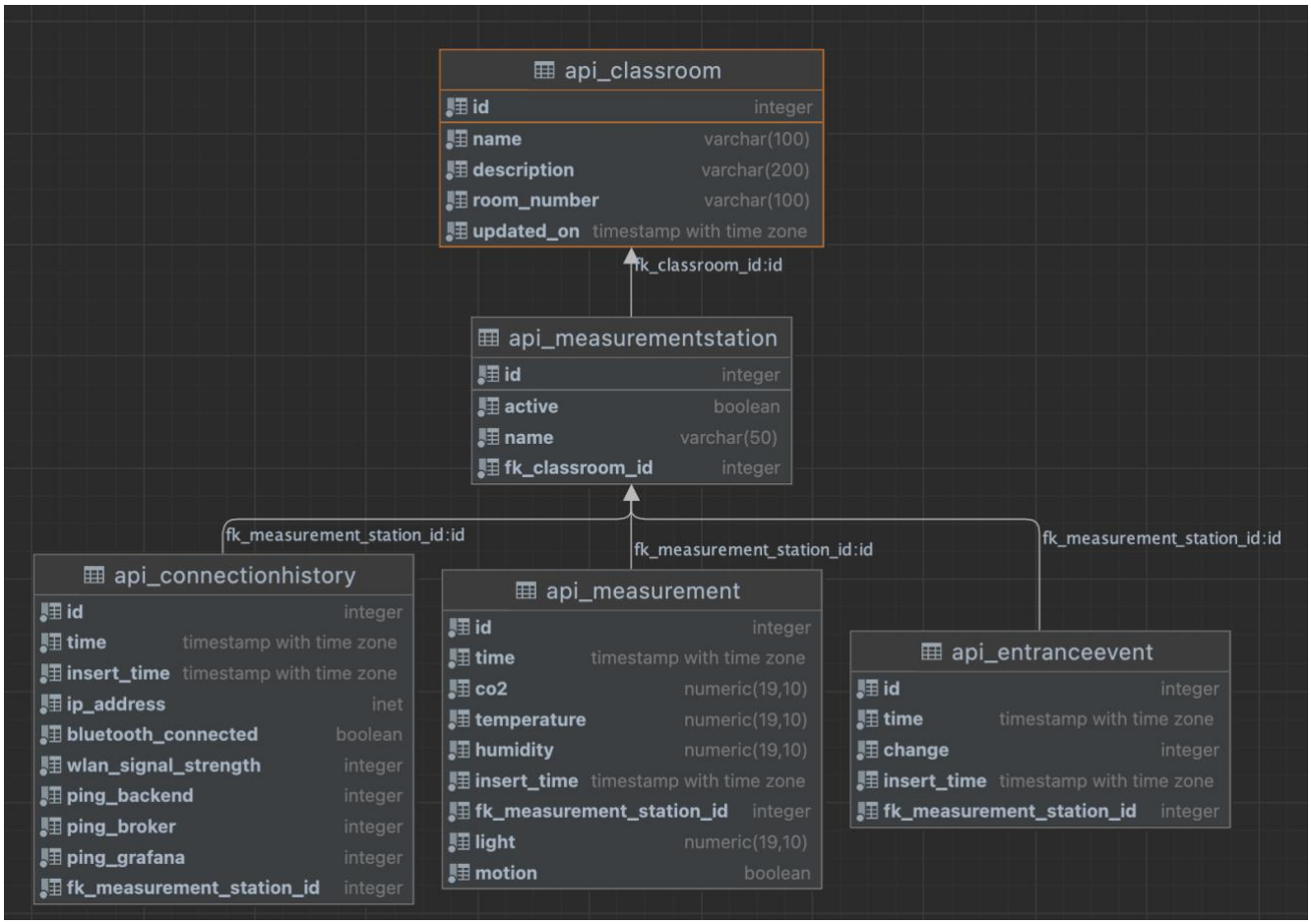
MACHINE	IP	OS	LAST SEEN
raspberrypi-florin-smart-classroom florin.barbisch@gmail.com External	100.106.70.47	Linux 1.22.2	Connected •••
raspberrypi-gabo-smart-classroom florin.barbisch@gmail.com External	100.72.187.47	Linux 1.22.2	Connected •••
raspberrypi-yvo-smart-classroom keller.yvo@gmail.com Shared +2 No expiry	100.111.190.90	Linux 1.22.2	Connected •••

8.10 Abbildung 10 Termius

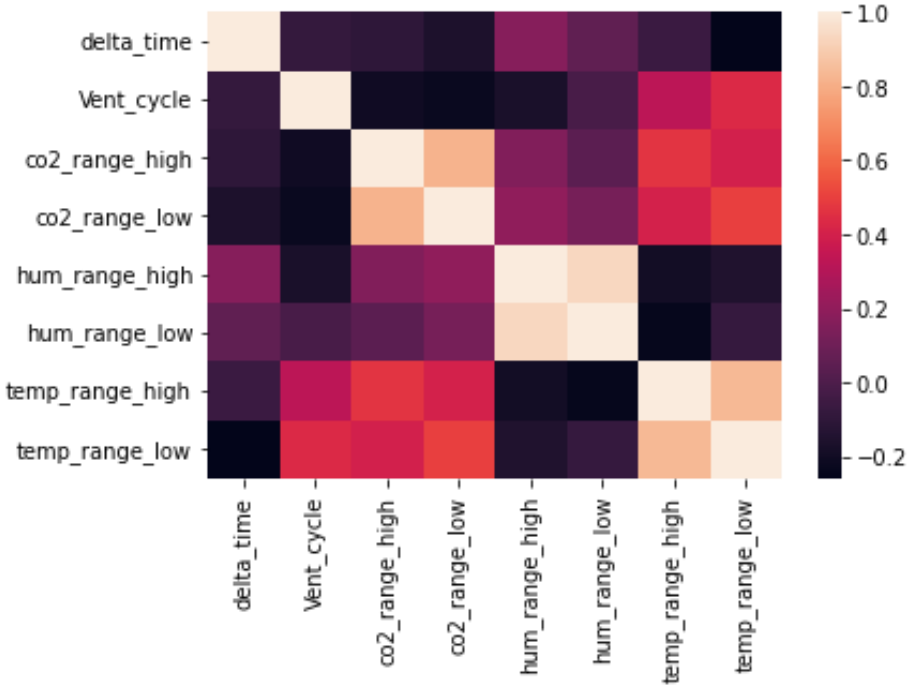
8.11 Abbildung 11 EntranceEvent

```
class EntranceEvent(TimescaleModel):
    fk_measurement_station = models.ForeignKey(MeasurementStation, on_delete=models.CASCADE)
    change = models.IntegerField()
    insert_time = TimescaleDateTimeField(interval="7 days")
```

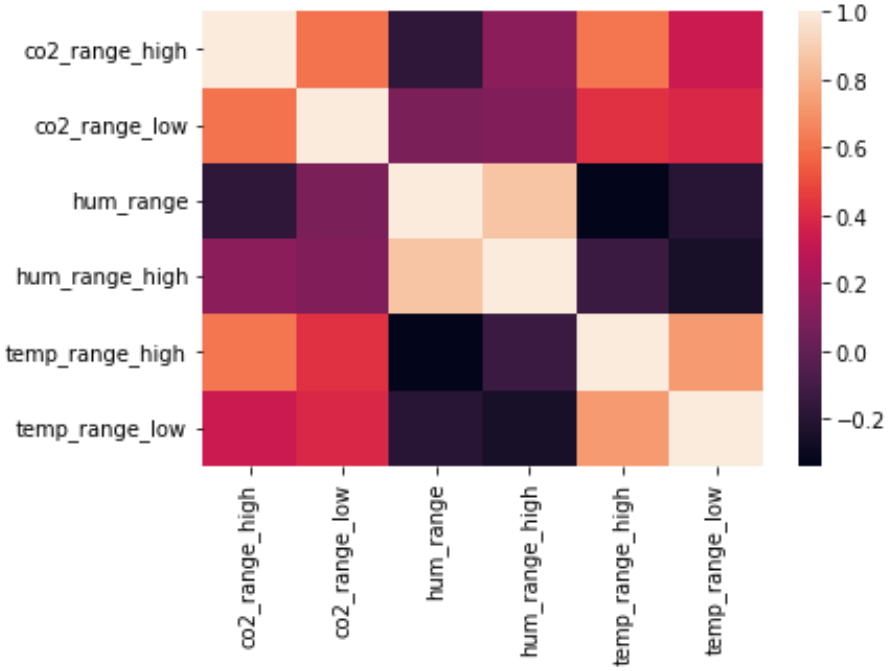
8.12 Abbildung 12 Datenbank Model TimeScale DB



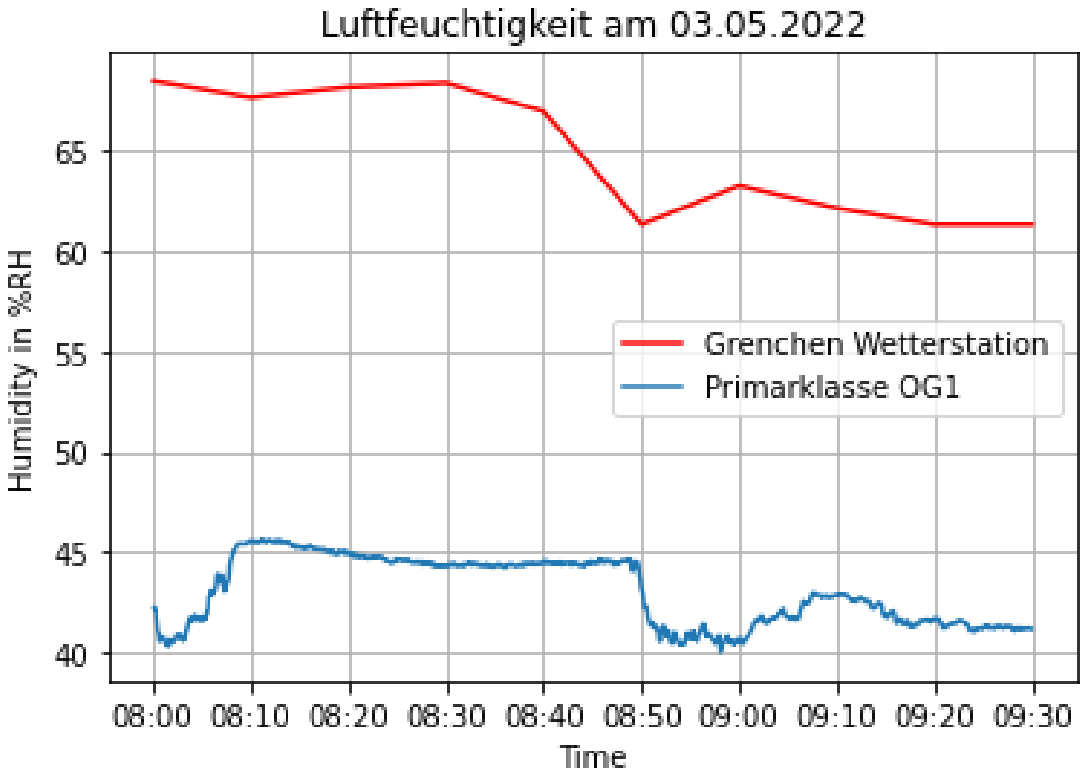
8.13 Abbildung 13 Korrelation Delta Lüften



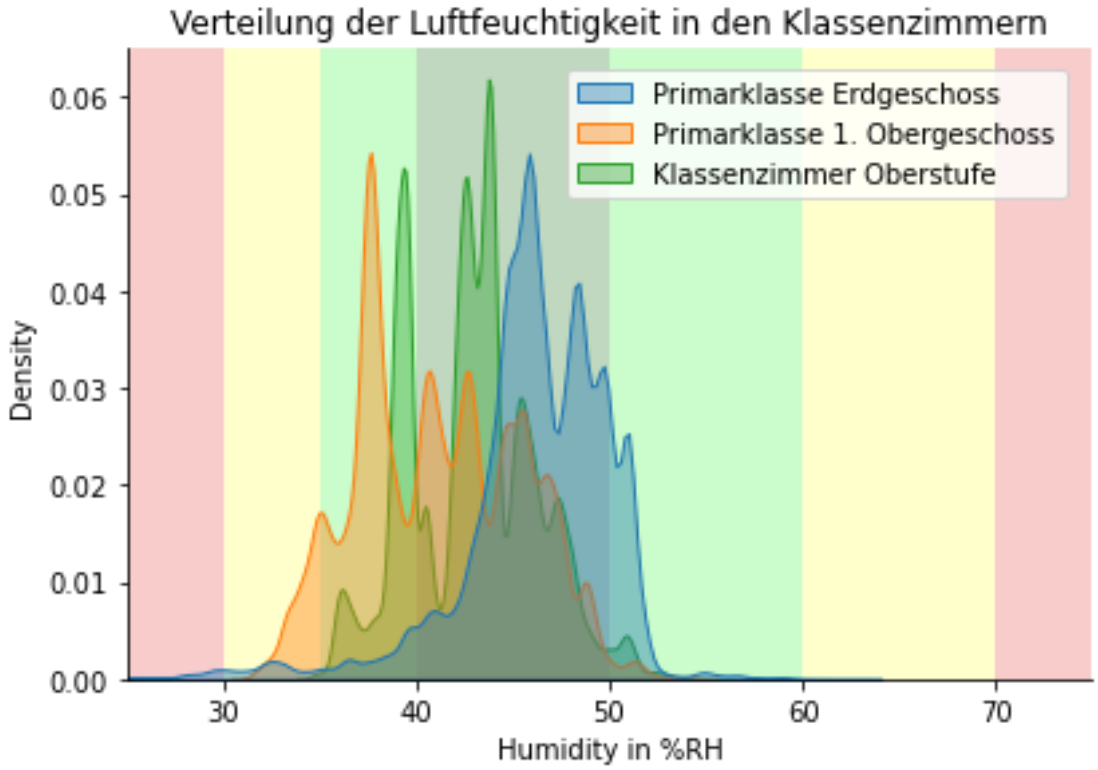
8.14 Abbildung 14 Korrelation Delta Zwischen dem Lüften



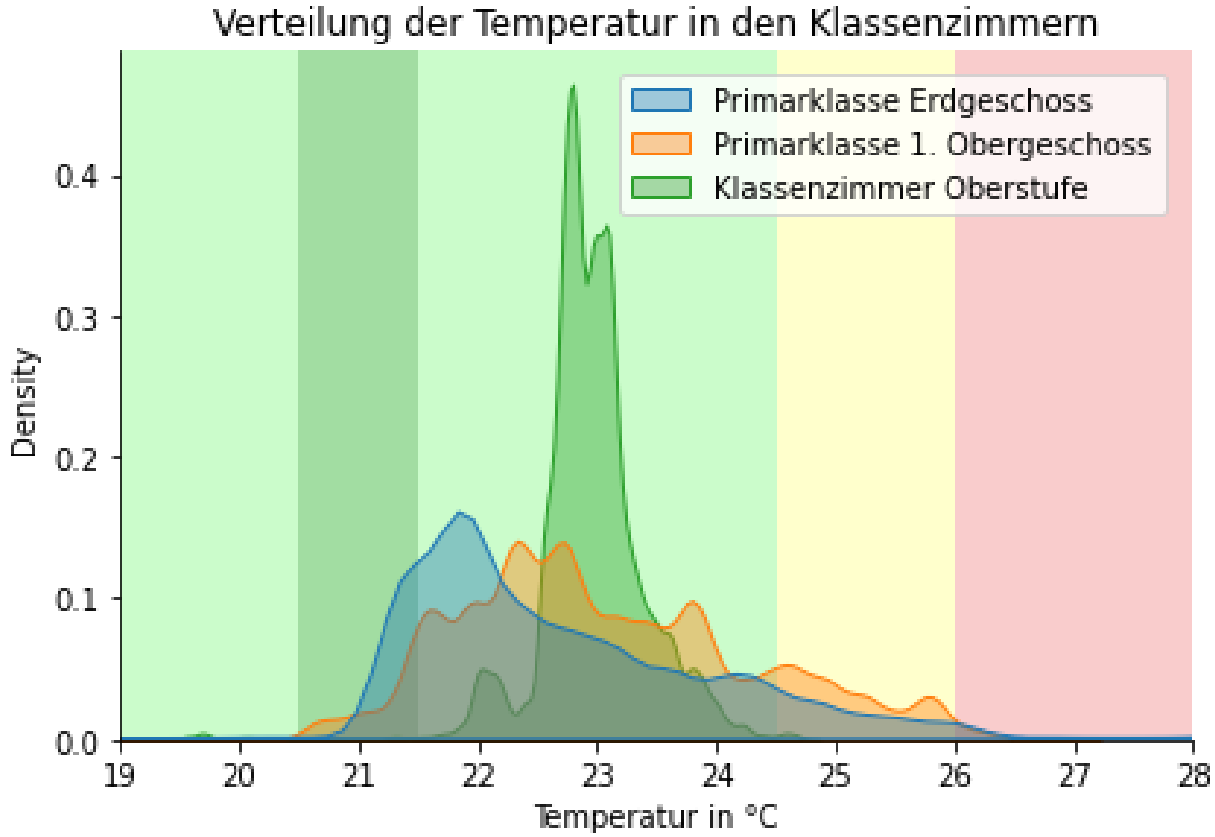
8.15 Abbildung 15 Luftfeuchtigkeit am 03.05.2022



8.16 Abbildung 16 Verteilung der Luftfeuchtigkeit in den Klassenzimmern



8.17 Abbildung 17 Verteilung der Temperatur in den Klassenzimmern



8.18 Abbildung 18 CO_2 Grafana