**Dokumentation**

# Aufgabenstellung

## Allgemein

Es wird eine Digitale Raumluftstation zum Messen von Temperatur, Luftfeuchtigkeit und Luftdruck designed und aufgebaut.

## Vorgaben

Für den Microcontroller soll ein ESP-32-C3 verwendet werden.

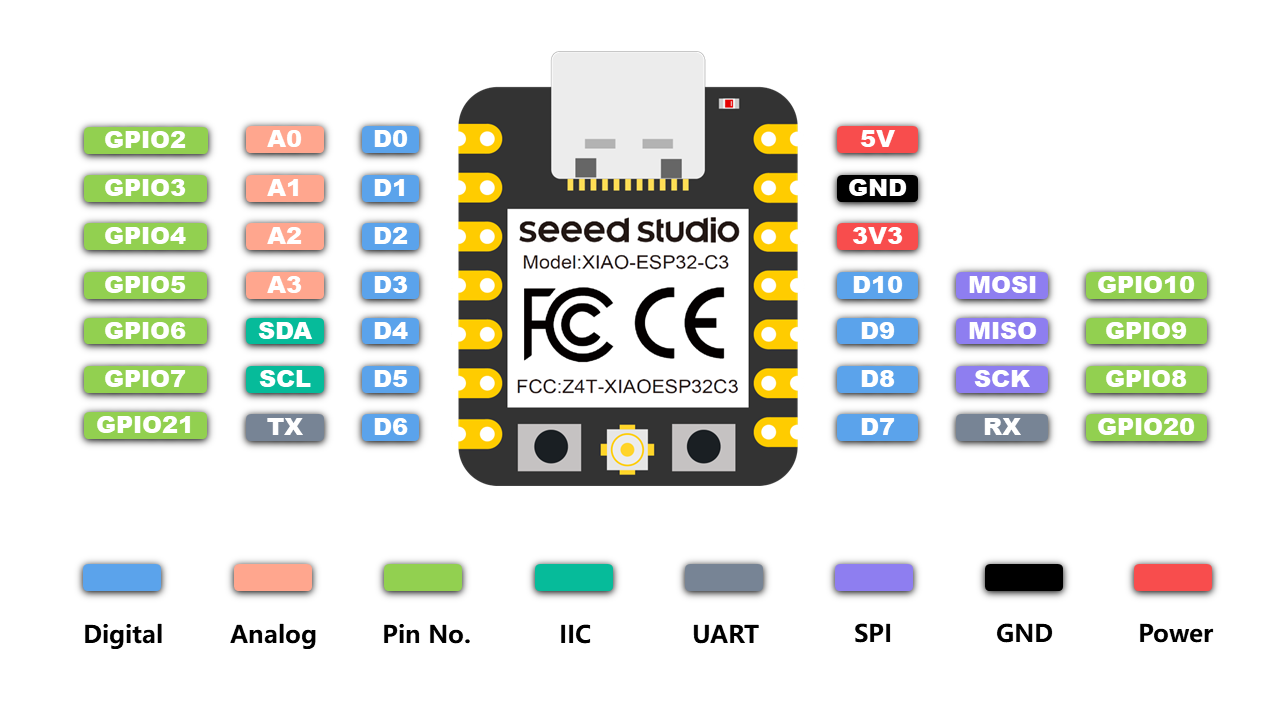
Als Sensor wird der BME280 verwendet. Dieser ist sehr genau und hat alle benötigten Funktionen integriert.

## Hauptziele

* Aufnahme der Sensorwerte
* Ausgabe auf dem Display
* Webserver
* Platine
* Gehäuse

# Ressourcen

### Pinout des ESPs:



# 1. Aufnahme der Sensorwerte

### 1.1 Library

Für den BME280 gibt es eigene Libraries nur für diesen Baustein. Wir haben uns für folgende entschieden:

#include <BME280\_t.h>

Diese ist eine Leichtgewichtige Version der gesamten Library. Diese hat alle gewünschten Funktionen für das Projekt.

Angesteuert wird der Sensor mittels I2C.

### 1.2 Code

Zuerst muss ein Objekt erstellt werden



Mittels eingebauten Funktionen, werden die Sensorwerte dann ausgelesen und für Debugzwecke auf den Serial Monitor ausgegeben.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

# 2. Ausgabe auf dem Display

Da diese Funktion bei uns etwas umfangreicher war haben wir diesen Teil in einem eigenen Header File gemacht.

Verwendete Library:  


### 2.1 Ausgabe

Ein Bild, das Text, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

An die Main werden hier mehrere Funktionen gegeben, die unterschiedliche Ausgaben auf dem Display zu bewirken.

In der Main:





### 2.2 Display verhalten

Unser Display hat mehrere Funktionen die mit zwei Tastern durchgeswitched werden können. Hierzu werden zwei Taster als Interrupt hinzugefügt welche beim Druck ausgelöst werden.

Ein Bild, das Text, Schrift, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Im Handler wird dann je nach Buttondruck eine Variable hoch oder runtergezählt.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Im Loop wird dann festgelegt, welche Display Ausgabe je nach BoardNum angezeigt werden sollte.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

### 3. Webserver

Da nun eine Ausgabe auf dem Display möglich ist, ist noch gewünscht, dass man sich seine Sensordaten Online auch noch anschauen kann. Um auf dem ESP einen Webserver laufen zu lassen, gibt es wieder Libraries mit denen man einen Server starten kann.

Ein Bild, das Text, Schrift, Screenshot, Grafiken enthält.

KI-generierte Inhalte können fehlerhaft sein.

Um sich mit dem WIFI zu verbinden gibt es hier zwei Möglichkeiten, entweder man verbindet sich mit einem bereits bestehenden WLAN Netzwerk, wie zum Beispiel dem HTL\_IoT oder man erstellt sich mit dem ESP sein eigenes Netz.

Wie wir später dann mitbekommen haben ist die zweite Option nicht von Vorteil, da hierbei keine Graphen für den Webserver verwendet werden können.

Deshalb wird mit einem bereits bestehenden Netz gearbeitet.

Normalerweise würden wir das IoT Netz der HTL verwenden, da das aber teilweise ewig gebraucht hat bis der ESP sich verbunden hat, haben wir einen Hotspot erstellt.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Mit der Asynch-Weberver-Library kann ganz einfach ein Webserver zum laufen gebracht werden.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Oben wird der Server mit der index.html (welche sich in einer Header Datei befindet um den Code Strukturiert zu halten) geladen.

Mit dem unteren Server Aufruf werden die Werte an die Javascript übergeben.



Gestartet wird der Server über den http Port 80.

# 3. Altium Design

## 3.1 Schematic

Die Altium Schematic ist eher schlicht gehalten. Das Display (P1) und der Sensor (MOD1) hängen direkt am ESP und dessen Versorgung. Zum Weiterschalten des Interfaces am Display, werden zwei Taster S1 & S2 verwendet.



## 3.2 PCB

Ein Bild, das Text, Screenshot, Farbigkeit, Grafikdesign enthält.

KI-generierte Inhalte können fehlerhaft sein.

# 4. Das Gehäuse

Das Gehäuse besteht aus einer Unterseite, einem Deckel, sowie Tasterelemente zum Bedienen von den Pushbuttons. Der Gesamtaufbau sieht so aus:

Ein Bild, das Stecker enthält.

KI-generierte Inhalte können fehlerhaft sein.

## 4.1 Unterboden

Der Unterboden fixiert die Platine, besitzt aber auch Ausschnitte für den USB-Anschluss des ESPs, sowie ein Gitternetz für den Sensor. Zum Fixieren des Deckels gibt ein Snap-Enclosure System.

Ein Bild, das Elektronik, Elektronisches Bauteil, Design enthält.

KI-generierte Inhalte können fehlerhaft sein.

## 4.2 Deckel

Der Deckel hat einen Ausschnitt und eine Halterung für das OLED-Display. Außerdem gibt es zwei Löcher in welchen die Tasterelemente eingesetzt werden. Auf dem zweiten Bild lässt sich das Snap-Enclosure System ziemlich gut erkennen.

Ein Bild, das Rechteck, Quadrat, Bilderrahmen, Design enthält.

KI-generierte Inhalte können fehlerhaft sein.Ein Bild, das Himmel, Rechteck, Screenshot, Design enthält.

KI-generierte Inhalte können fehlerhaft sein.

## 4.3 Tasterelemente

Die Tasterelemente (Verlängerungen) sind einfache Klötze, die bis zu den Pushbuttons auf der Platine reichen, um diese zu aktivieren. Dadurch das der Unterkörper der Tasterverlängerung breiter ist, kann dieser auch nicht aus dem Deckel herausrutschen.

