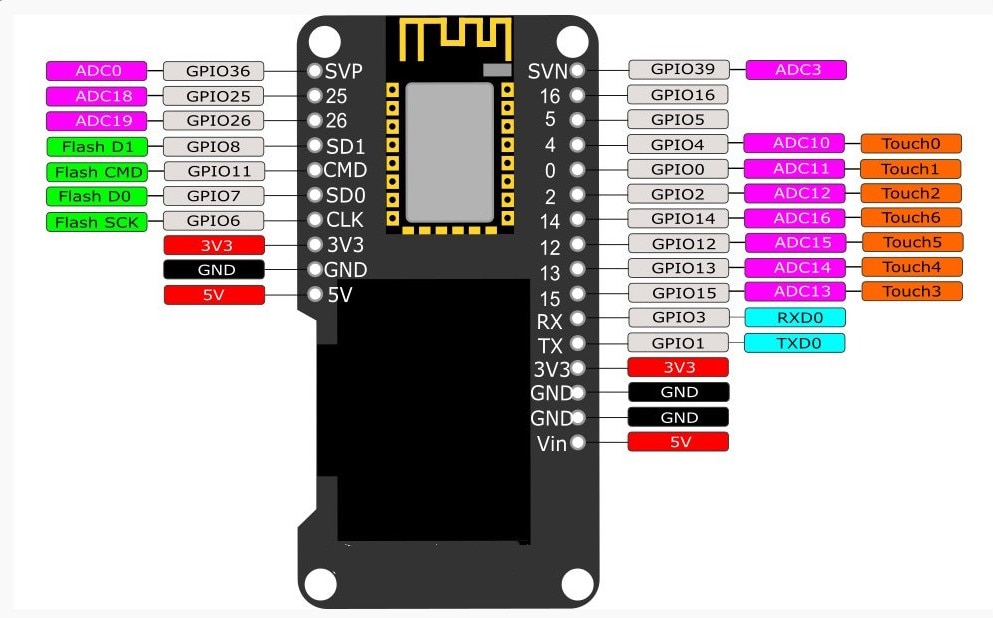
Wemos Board SCD30 CO2 Sensor

# Pins des Wemos



Die Pins 2 und 14 können problemlos für ein bit-banging I2C verwendet werden. 13 und 15 ebenso; allerdings wurden 13 und 15 für ein SoftSerial genutzt, daher brauchte ich andere.

Pin 12 funktionierte nicht – legte ich dort SDA von meinem Sensor an, bootete der ESP32 immer wieder neu.

Das OLED ist intern an GPIO4 und GPIO5 (also Pins 4 und 5) angeschlossen (SCL ist GPIO4, SDA ist GPIO5), die I2C-Adresse ist 0x3c.

# Sketche

Um den SCD30 (von Sensirion) und ggf. Parallel dazu noch den MHZ19 (aus China) anschließen zu können, brauche ich Ansteuerung von 4 Geräten:

1. Das OLED SSD1306 (128x64 pixel monochrome OLED) – ein I2C device
2. Den SCD30, ebenfalls ein I2C device
3. Den MHZ19, seriell mit 9600 baud
4. Den MiCS6814, ebenfalls ein I2C device.

Ein gewisses Problem entsteht dadurch, dass I2C dreimal verwendet wird, und dass dafür nicht derselbe I2C Port verwendet werden kann: das SCD30 kann allerhöchstens 100 kHz I2C Frequenz und macht dabei noch massives Clock Stretching (bis zu 150 ms!!!); das OLED hingegen kann viel schneller angesprochen werden und sollte das auch, weil recht viele Daten übertragen werden müssen.

Der ESP32 hat mit so langem Clock Stretching ein Problem. Github-User paulvha behauptet, der ESP8266 könnte es, und der ESP32 nicht, und verwendet daher in seiner Library ein SoftWire (eine bitbanging-Implementierung für I2C).

## OLED SSD1306

Die Library die verwendet wird ist die in der Standard-Arduino-Library-Sammlung befindliche Library „ESP8266 and ESP32 OLED driver for SSD1306 displays“ von ThingPulse, Fabrice Weinberg. Ich verwende die 4.1.0. Weil es eine Standardlibrary ist, befindet sie sich im Standardpfad:

C:\Users\Familie\Documents\Arduino\libraries\ESP8266\_and\_ESP32\_OLED\_driver\_for\_SSD1306\_displays

Der Aufbau der Library ist wie folgt. Die Hauptklasse its OLEDDisplay, darin werden die Sachen wie setPixel, drawString usw. definiert. Von OLEDDisplay erben drei verschiedene Klassen, von denen der Benutzer eine aussucht: SSD1306Wire, SSD1306Brzo (eine alternative I2C Implementierung in Assembler die bis zu 1 MHz I2C-Geschwindigkeit unterstützt) und SSD1306Spi (es gibt auch SSD1306 die per SPI angeschlossen sind, das Wemos nicht, diese ist also nicht nutzbar).

## MH-Z19

Es gibt da 2 Libraries bei den Standard-Arduino-Libraries; ich verwende die „MH-Z19“ von Jonathan Dempsey in Version 1.5.1 (die andere von T. Schürg unterstützt nur Arduino, kein ESP32).

Mein erster Versuch verwendete SoftSerial mit Pins 13 und 15 als Tx/Rx und das tat auch sogleich; man kann allerdings auf dem ESP32 auch HardwareSerial verwenden (sagt die Library). Ich hatte mich nicht getraut weil ich befürchtet habe, der USB-Port der zum Flashen verwendet wird verwendet ebenso HardwareSerial und darum könnte es einen Konflikt geben bzw. das Debuggen mit SerialMonitor wäre nicht möglich.

Schlussendlich habe ich dann doch HardwareSerial (vom Standard-ESP32-Arduino-Code) verwendet, und zwar mit Parameter 1 im Konstruktor (also nicht SERIAL0, sondern wohl SERIAL1).

## SCD30

Es gibt hier keine Standard-Arduino-Library. Ich habe was Funktionierendes bei github gefunden und es zu mir (Lecostarius) auf github geklont: paulvha/scd30. Das ist nun bei mir auch scd30. Das habe ich ins Library-Verzeichnis geklont, nach

C:\Users\Familie\Documents\Arduino\libraries\scd30

Und dann einen Bug in paulvha\_SCD30.cpp korrigiert – da war ein \_i2cPort->begin() drin, das hat die Standardpins für SDA und SCL benutzt, auch wenn der User andere beim Aufruf angegeben hat. Mit dieser Library funktioniert – mit SoftI2C – der SCD30. Allerdings verträgt sich die Library nicht mit der SSD1306 Library! Warum? Weil die Funktionen von „Wire.h“ doppelt definiert werden, einmal als Teil der Soft-I2C Implementierung in der SCD30 Library von paulvha, und einmal in der OLED Library, die auf die Espressif-Implementierung von Wire zurückgreift. Damit schlägt das Linken fehl.

## Versuch, SCD30 und OLED SSD1306 gleichzeitig zu nutzen

Die Idee wäre, die Brzo-Implementierung von I2C in der SSD1306-Library zu nehmen. Allerdings gibt es die Brzo-Implementierung ausschließlich für den 8266.

Die andere Möglichkeit: die Soft-I2C-Implementierung in der paulvha\_SCD30 Library so ändern, dass sie nicht mehr TwoWire implementiert, sondern SoftWire.

Gemacht – nun tut es. Achtung: aus irgendeinem Grund muss display.init() vor dem swi.begin() kommen, scheint mir... außerdem funktioniert das Flashen des Boards nicht wenn SDA und SCL an Pins 2 und 14 dran sind (13,15 sind tolerant!).

Der Sketch der jetzt läuft heißt wemos\_scd30.

Es tut aber nur wenn das SCD30 Modul 3.3V Betriebsspannung von extern bekommt.

## PMS7003 Feinstaubsensor

Der Sensor schickt regelmäßig über UART seriell Datenpakete. Er braucht keinerlei bidirektionale Verbindung; es genügt, das TX anzuschließen.

Um seriell auf dem ESP32 kommunizieren zu können, kann man SoftSerial oder HardwareSerial verwenden. Außerdem gibt es noch unter der Arduino IDE die Klasse „Serial“, die vom Arduino her kommt, und die HardwareSerial(0) entspricht. Der ESP32 hat drei UARTs die quasi beliebig auf I/O Ports gemappt werden können. Die ESP32 Arduino Library implementiert dazu die Klasse HardwareSerial und die drei UARTs kann man aussuchen mit

HardwareSerial ser(0); // identisch zu Serial ser

HardwareSerial sef(1);

HardwareSerial seg(2);

Man kann das nutzen wie folgt:

include <HardwareSerial.h>

HardwareSerial MySerial(1);

void setup() {

MySerial.begin(9600, SERIAL\_8N1, 16, 17); // RX,TX

}

void loop() {

while (MySerial.available() > 0) {

uint8\_t byteFromSerial = MySerial.read();

// Do something

}

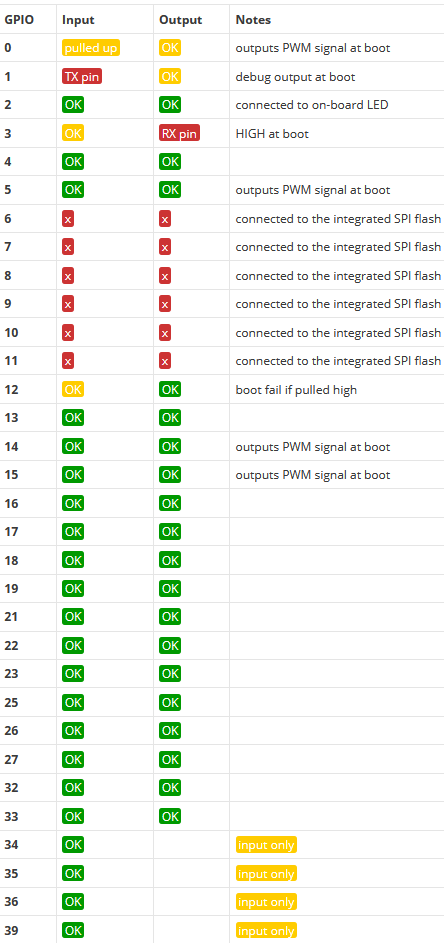
//Write something like that

MySerial.write(rand(0, 255));

}

Bei Arduino gibt es für die Boards, die 2 UARTs haben, Serial1 und Serial2.

Ich habe gesehen dass UART0 auf Pins 1 (TX) und 3 (RX), sowie UART2 auf Pins 17 (TX)/16 (RX) gemappt wurden. Pins 34 bis 39 sind übrigens ausschließlich Eingänge. Komplett freie Pins für I/O sind 4,5, und 13-33 (von https://randomnerdtutorials.com/esp32-pinout-reference-gpios/):



## MiCS-6814

Der Sensor hat 3 Sensoren und kann entweder mit 3.3 oder mit 5V betrieben werden. Die Heizleistung beträgt 90 mW und die maximale Leistung 150 mW (wären 50 mA bei 3.3V), I2C geht mit 100 kHz typisch. Die I2C-Adresse ist (7 bit) 1110000 oder 0x70.

Ich habe ein Seeed-Breakout und das hat I2C-Adresse 0x04 (kann umgestellt werden).

Das einfachste Kommando ist CMD\_CONTROL\_LED, das ist Register 10 (0x0A), und dort schreibt man 1 rein für an und 0 für aus.

Nach langem Rumspielen kann ich nun die LED ein und ausschalten. Grundsätzlich funktioniert also die (schreibende) I2C-Kommunikation. Der Sketch heißt mics6814.ino und ist in Documents\Arduino\mics6814.

Die Original-Library von Seeed gefällt mir aus mehreren Gründen nicht. Ich habe daher meine eigene Version davon abgeleitet, die heißt mics6814. Die Kalibrierung erwartet keine so winzigen Sensorabweichungen (die Originalkalibrierung hat nicht terminiert), die LED wird nicht bei jedem Lesevorgang eingeschaltet, und das Wire.begin() ist nicht mehr in der Library.