Hands-on Environmental Sensing I

Praktikum - Aufbau und Charakterisierung eines Low-Cost CO2 Sensors

Daniel Kühbacher

Version: 0.2

Zuletzt bearbeitet: 14. Mai 2025

Autor: Daniel Kühbacher

Inhalt

[Vorbereitung 3](#_Toc193370145)

[Einführung 4](#_Toc193370146)

[Ablauf 4](#_Toc193370147)

[Aufgabe 1: Verbindung zum Microcontroller 5](#_Toc193370148)

[Aufgabe 2: Serieller Monitor 5](#_Toc193370149)

[Aufgabe 3: Temperatur und Luftfeuchtigkeit messen. 5](#_Toc193370150)

[Aufgabe 4: Inbetriebnahme der SD-Karte 6](#_Toc193370151)

[Aufgabe 5: Sensordaten auf SD-Karte speichern 6](#_Toc193370152)

[Aufgabe 6: Inbetriebnahme des CO2 Sensors 6](#_Toc193370153)

[Aufgabe 7: CO2 Beacon 7](#_Toc193370154)

[Zusatzaufgabe 7.1: Zeitstempel 7](#_Toc193370155)

[Aufgabe 8: Datenauswertung mit Python 7](#_Toc193370156)

# Vorbereitung

Zur Vorbereitung für das Praktikum sind zwischen 30 min bis 2 h erforderlich, je nach Vorerfahrung mit der **Arduino Microcontroller Programmierung** und **Python**.

Das Praktikum wird in 2er-Gruppen durchgeführt, und jede Gruppe benötigt einen eigenen Laptop. Ihr könnt euch im Voraus absprechen und Gruppen bilden. Falls ihr möchtet, könnt ihr euch einen Laptop teilen und nur einen mitbringen.

* Installiere vor dem Praktikum die **Arduino IDE**  
  Link zum Download: <https://www.arduino.cc/en/software>  
    
  Zur weiteren Vorbereitung, insbesondere wenn noch keine Erfahrung mit Arduino Programmierung hast sehe dir das folgende Video an:   
  **Arduino Crash Course in 15 Min**  
  <https://youtu.be/nL34zDTPkcs?si=MqPz2jDbrG39jyJ->
* Installiere **Visual Studio Code** und die **Erweiterung für Jupyter Notebooks** (erforderlich für die Datenauswertung in Python)   
  Link zum Download: <https://code.visualstudio.com/Download>  
    
  Zur Einführung und Installation von Jupyter sehe dir das folgende Video an:  
  <https://youtu.be/suAkMeWJ1yE?si=K90X1-_5LGSxWY5A>
* Lade und entpacke das **GitHub-Repository** zum Kurs auf deinen Laptop: <https://github.com/tum-esm/HES-Praktikum.git>

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

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 1 - Herunterladen des Github Code Repositories

# Einführung

Kohlendioxid (CO₂) macht mit etwa 0,04 % nur einen geringen Anteil der Erdatmosphäre aus. Dennoch hat es eine erhebliche Klimawirkung, da es die von der Erde abgegebene Wärmestrahlung im nahen Infrarotbereich absorbiert und teilweise zurück auf die Erdoberfläche reflektiert – ein Effekt, der als Treibhauseffekt bekannt ist. Ohne diesen natürlichen Treibhauseffekt wäre Leben auf der Erde nicht möglich, da die Durchschnittstemperatur der Erdoberfläche bei etwa -18 °C läge. Durch menschliche Aktivitäten, insbesondere die Verbrennung fossiler Brennstoffe, steigt jedoch die CO₂-Konzentration kontinuierlich an und verstärkt den Treibhauseffekt. Dies führt zu einer globalen Erwärmung mit weitreichenden Folgen für Klima, Wetter und Ökosysteme.

Die präzise Messung von CO₂ ist essenziell, um natürliche und anthropogene Quellen sowie Senken des Treibhausgases besser zu verstehen. Nur so lassen sich gezielte Maßnahmen zur Reduktion von Emissionen entwickeln und die Auswirkungen des Klimawandels effektiver begrenzen. Eine zentrale Methode zur Bestimmung der CO₂-Konzentration basiert auf dem Lambert-Beer-Gesetz, das die Abschwächung von Lichtintensität durch Absorption beschreibt. CO₂ absorbiert infrarote Strahlung bei spezifischen Wellenlängen, wodurch seine Konzentration in der Luft durch nicht-dispersive Infrarotspektroskopie (NDIR) genau bestimmt werden kann. Diese Methode wird in Laboranalysen, Umweltmessstationen und sogar Satelliten eingesetzt, um den globalen CO₂-Gehalt kontinuierlich zu überwachen.

Im Praktikum “Hands-On Environmental Sensing” werden wir einen Low-Cost CO₂ Sensor in Betrieb nehmen, Daten erfassen und den Sensor charakterisieren. Dabei wird ein Microcontroller (ESP32-C3) so programmiert, dass er Daten von einem Temperatur- und Luftfeuchtigkeitssensors sowie einem CO₂ Sensor empfangen und an einen Seriellen Monitor senden sowie auf eine SD-Karte speichern kann. Daraufhin wird eine Messkampagne mit Referenzgasen durchgeführt und damit der Sensor charakterisiert.

## Ablauf

* *(15 min)* Einführung in die Zusammensetzung der Atmosphäre, Treibhauseffekt, physikalische Grundlagen und Methoden der CO₂ Gasmesstechnik
* *(15 min)* Inbetriebnahme des Microcontrollers
* *(30 min)* Inbetriebnahme des Temperatur- & Luftfeuchtigkeitssensors sowie der SD-Karte
* *(45 min)* Inbetriebnahme des CO2 Sensors und Implementierung der Messroutine
* *(30 min)* Unterschiedliche Messungen (Referenzgas, Innenluft, Außenluft)
* *(45 min)* Auswertung und Visualisierung der Daten in Python

# Aufgabe 1: Verbindung zum Microcontroller

Öffne die Arduino IDE. Gehe zu *Tools >Board >Boards Manager* und installiere „**esp32** by Espressif Systems**“.** Gehe zu *File >Examples >01.Basics >Blink.* Ein neues Fenster mit dem Blink Beispiel wird sich öffnen.

Verbinde nun den Microcontroller mit dem Computer. Wenn der Microcontroller erkannt wird, kannst du nun den entsprechenden PORT und das verwendete BOARD *„ESP32C3 Dev Module“* auswählen. Lade das Blink Beispiel hoch und überprüfe, ob die Verbindung zum Microcontroller erfolgreich hergestellt wurde.

Problembehandlung 👉 Sollte der Microcontroller nicht erkannt oder kein Port in der Arduino IDE angezeigt werden, ist es sehr wahrscheinlich, dass der notwendige Treiber fehlt.

Unter dem folgenden Link findest du den richtigen Treiber zur Installation und Fehlerbehebung.

<https://www.silabs.com/developer-tools/usb-to-uart-bridge-vcp-drivers?tab=downloads>

## Aufgabe 1.2: RGB LED

Da der ESP32 zusätzlich eine RGB Led verbaut hat, kann die Lichtfarbe verändert werden.

ToDo ✅ Verändere das Blink Beispiel so, dass die LED abwechselnd rot und grün blinkt. Die Variablen <*red/green/blue\_val*> können einen Wert von 0-255 annehmen, was der Intensität der entsprechenden Farbe entspricht.



# Aufgabe 2: Serieller Monitor

Um Daten vom ESP an den Laptop zu senden und anzuzeigen, verwenden wir den Seriellen Monitor. Dieser muss in der Setup Routine mit einer bestimmten Übertragungsgeschwindigkeit (Baud-Rate) initialisiert werden. Die Baud-Rate gibt die Anzahl der übertragenen Symbole pro Sekunde an und entspricht der Bitrate, wenn ein Symbol genau einem Bit entspricht (z.B. bei binärer Übertragung. Werden mit einem Symbol mehrere Bits übertragen (z.B. bei komplexeren Modulationen), ist die Bitrate entsprechend höher als die Baudrate.

ToDo ✅ Füge diese Funktion im Blink-Beispiel hinzu und sende „LED rot“ und LED grün“ entsprechend dem LED-Zustand.



# Aufgabe 3: Temperatur und Luftfeuchtigkeit messen.

Verbinde nun den Microcontroller mit dem Temperatur- und Luftfeuchtigkeitssensor AHT20. Die Datenübertragung erfolgt mit I2C, sodass neben der Spannungsversorgung (3V) die Datenleitung SDA (Serial Data) und ein Takt SCL (Serial Clock) verbunden werden müssen. Verbinden **SDA mit GPIO 8** und **SCL mit GPIO 9** des Microcontrollers (Default Pins für I2C).

Gehe zu *Sketch >Include Library >Manage Libraries* u^nd installiere „**DFRobot\_AHT20** by DFRobot“.

Nachdem die Library erfolgreich installiert wurde, gehe zu *File >Examples >DFRobot\_AHT20 >readAHT20.* Ein neuer Sketch mit Beispielcode zum Auslesen des Sensors öffnet sich. Lade den Beispielcode hoch, um die Verbindung zum Sensor zu testen.

Nun würden wir die Messwerte gerne als Liniengraph am Computer darstellen. Hierzu bietet die Arduino IDE einen Serial Plotter unter *Tools> Serial Plotter.* Dieser erfordert jedoch ein spezielles Tabellenformat, um die Daten zu visualisieren. Mit einem tabspace („\t“) werden die mehrere Variablen in einer Zeile getrennt. Mit dem zeilenumbruch („\n“) wird ein neuer Schritt auf der x-Achse erzeugt.

ToDo ✅ Verändere den Beispielcode so, dass die Temperatur (in °C) und die Luftfeuchtigkeit (in %rH) im Serial Plotter dargestellt werden kann. Überprüfe wie sich der Temperatur- und Luftfeuchtigkeitswert verändert, wenn du z.B.: auf den Sensor atmest.

Speichere den Code zu dieser Aufgabe als neuen Sketch im Code-Ordner des Praktikums.



Information 👉 GPIO 8 wird auch für den Betrieb der internen RGB LED verwendet (siehe Abbildung 2), wodurch diese beim Betrieb des Sensors hell leuchtet. Wir können alternativ GPIO 0 als SDA und GPIO 1 als SCL für I2C definieren. Dazu müssen wir die Verdrahtung entsprechend ändern und im Code ist eine Definition der I2C Pins und eine Re-Initialisierung des I2C Interfaces erforderlich.

# 

ToDo ✅ Ändere die Verdrahtung der I2C Pins auf GPIO 0&1 und ändere den Code wie beschrieben.

# Aufgabe 4: Inbetriebnahme der SD-Karte

Verbinde nun die SD-Karte mit dem Microcontroller. Sie benötigt wiederum eine Spannungsversorgung (3V) und wird mittels SPI-Bus (Serial Peripherial Interface) verbunden. Dieser benötigt jedoch im Gegensatz zu I2C zwei Datenleitungen und eine Taktleitung. Verbinden wie folgt CS (Chip Select) mit GPIO 7, SCK (Serial Clock) mit GPIO 4, SO/MISO (Master In Slave Out) mit GPIO 5 und SI/MOSI (Master Out Slave In) mit GPIO 6.

Öffne den Beispielcode „02\_SDCard.ino“ aus dem *code* Ordner.

Gehe zu *Sketch >Include Library >Manage Libraries* und installiere „**ArduinoJson** by Benoit Blanchon“.

ToDo ✅ Lade den Beispielsketch hoch und überprüfe, ob die Verbindung mit der SD-Karte funktioniert.

# Aufgabe 5: Sensordaten auf SD-Karte speichern

Nun wollen wir echte Temperatur- und Luftfeuchtigkeitsdaten auf der SD-Karte speichern. Als Grundlage nutzen wir hierzu den Code aus Aufgabe 4 („02\_SDCard.ino“) und fügen die notwendigen Funktionen aus Aufgabe 3 hinzu.

ToDo ✅ Integriere den Code von Aufgabe 3 in den code von Aufgabe 4 sodass „echte“ Temperatur und Luftfeuchtigkeits-Messdaten auf der SD-Karte gespeichert werden.

Plotte die Daten zusätzlich im Seriellen Monitor.

# Aufgabe 6: Inbetriebnahme des CO2 Sensors

Als letzten Baustein des Systems nehmen wir nun den CO2 Sensor in Betrieb. Verbinde hierzu den den EN (Enable) Pin mit GPIO 3 und den RDY (Ready) Pin mit GPIO 2. Der Sensor wir auch über I2C ausgelesen und kann einfach mit den SCL- und SDA-Pins des Temperatur- und Luftfeuchtigkeitssensors verbunden werden (Bus-Topologie).

Abbildung 1 ist der Microcontroller und die verwendeten Pins dargestellt.

ToDo ✅ Öffne nun den Sketch *„03\_co2\_sensor.ino“,* lade das Skript hoch und überprüfe die Verbindung zum CO2 Sensor.

Ein Bild, das Screenshot, Diagramm, Pixel enthält.

KI-generierte Inhalte können fehlerhaft sein.

Abbildung 2 – Übersicht über alle GPIOs und ihre Verwendung am ESP32C3 Devkit

# Aufgabe 7: CO2 Beacon

Da nun auch die CO2 Konzentration gemessen werden kann, sollen diese Daten auch auf der SD-Karte gespeichert und im Serial Plotter angezeigt werden.

ToDo ✅ Kombiniere den Code aus Aufgabe 5 mit dem von Aufgabe 6. Nutze als Grundlage hierzu den Code aus Aufgabe 6 (*„03\_co2\_sensor.ino“*)

## Zusatzaufgabe 7.1: Zeitstempel

Der ESP verfügt über ein WIFI-Modul, wodurch die Möglichkeit besteht, online die aktuelle Zeit abzufragen. Unter *File >Examples >ESP32 >Time >SimpleTime* findest du ein Beispiel dafür, wie diese Zeitabfrage über einen NTP-Server (Network Time Protocol) funktioniert. Öffne den Sketch und gib die SSID und das Password für ein WLAN-Netzwerk, beispielsweise vom Handy, an. Nachdem sich das Gerät mit dem Internet verbunden hat, sollte die aktuelle Zeit im Seriellen Monitor dargestellt werden.

ToDo ✅ Integriere die Funktion zum Abrufen von Zeitstempel aus dem Beispielcode in Aufgabe 7. Die Zeitstempel sollen lediglich auf der SD-Karte zu jedem Datenpaket gespeichert werden, nicht aber im seriellen monitor angezeigt. Zur Hilfestellung befinden sich nachfolgend noch einige Code-Ausschnitte.



# Aufgabe 8: Datenauswertung mit Python

Die erhobenen Daten können nun eingelesen, geplottet und weiterverarbeitet werden. Hierzu nutzen wir Python und ein Jupyter Notebook. Lege nun die SD-Karte in den Computer ein und kopiere das Datenfile in den Ordner *notebooks/data*.

Öffne mit Hilfe von Visual Studio Code den Ordner *notebooks*und öffnedie Datei *analysis\_notebook.ipynb* aus diesem Ordner.

ToDo ✅ Die weiteren Schritte zur Datenanalyse sind im analysis notebook beschrieben. Führe die Aufgaben dort aus.

# Prüfungsfragen

**Welches physikalische Gesetz wird bei der optischen Messung von Spurengasen in der Atmosphäre angewendet? Beschreibe dieses kurz.**Lösung: Beer-Lambert Gesetz; Bei atmosphärischen Messungen ermöglicht dieses Gesetz die Berechnung der Konzentration von Spurengasen anhand der gemessenen Lichtabschwächung bei charakteristischen Wellenlängen.

**Nenne mindestens 3 Bus-Protokolle, die typischerweise zur Kommunikation zwischen Mikrocontrollern und Peripheriegeräten eigesetzt werden (Abkürzung ist ausreichend).**  
Lösung:

* SPI (Serial Peripheral Interface)
* I2C (Inter-Integrated Circuit)
* UART (Universal Asynchronous Receiver Transceiver)
* CAN (Controller Area Network)

**Nenne 3 Herausforderungen von Low-Cost CO2 Sensoren im Vergleich zu wesentlich teureren Gas-Analysatoren (z.B.: Picarro G2401).**  
Lösung:

* Geringere Messgenauigkeit 🡪 Höhere Streuung in den Messwerten (Rauschen)
* Zeitlicher Drift des Mess-Signals 🡪 Regelmäßige Neukalibrierung erforderlich
* Empfindlichkeit gegen Umwelteinflüsse 🡪 Temperatur und Luftfeuchtigkeit haben oft einen großen Einfluss auf das gemessene Signal und müssen mitbetrachtet werden.

**Beschreibe kurze den Nutzen und die Unterschiede der Funktionen *void setup()* und *void loop()* bei der Arduino Microcontroller Programmierung?**

Lösung: Die Funktion void setup() wird nur einmal zum Beginn des Programmablaufs durchgeführt. Sie dient zur Initialisierung von z.B.: Pin Modi, der seriellen Kommunikation oder zur Initialisierung von Sensoren. In der Funktion void loop() befindet sich das Hauptprogramm und sie wird nach Abschluss der Setup-Routine endlos wiederholt.