Vergleich zweier Low-Cost NDIR CO2-Sensoren im Hinblick auf die Verwendung in einer CO2-Luftgüte Ampel

Bernd Laquai, 1.11.2020

Low-Cost CO2 Sensoren, die nach dem NDIR Prinzip arbeiten und sich für CO2-Ampeln eignen, gibt es einige. Aufgrund der Handelsstreitigkeiten zwischen USA und China entsteht allerdings eine gewisse Wettbewerbsverzerrung. In der Zwischenzeit ist die Beschaffung von Sensoren chinesischer Hersteller meist nur über die chinesische Handelsplattform AliExpress.com möglich. Europäische Distributoren scheuen sich, chinesische Hersteller im Direktvertrieb in ihr Sortiment zu nehmen. Wer die höhere Unsicherheit was die Absicherung beim Bezahlen und bei der Möglichkeit der Retoure im Defektfall nicht scheut, und auch sonst keine moralischen Bedenken hat, der kann dort chinesische Sensoren zu deutlich günstigeren Preisen bekommen. Wie es um die Produkt-Qualität bestellt ist, das ist schwer zu sagen, oft fällt die Qualitätssicherung in China zu Gunsten eines niedrigeren Preises nicht so umfangreich aus. Aber es gibt dabei sicher erhebliche Unterschiede zwischen den einzelnen Firmen.

Eine chinesische Firma, welche sich bisher mit Low-Cost Partikelsensoren für PM2.5 sehr hervorgetan hat (Serie PMS) ist die Firma Plantower. Sie hat auch einen NDIR CO2 Sensor im Programm, der unter der Bezeichnung DS-CO2-20 erhältlich ist. Dieser Sensor hat einen Messbereich von 400-5000 ppm, und die Genauigkeit ist mit ± （50ppm+5% des Messwerts） spezifiziert. Diese Genauigkeit wird vermutlich nur zwischen 400 und 3000ppm erreicht (effektiver Messbereich) was aber ausreichend für eine CO2-Ampel ist. Per Default nutzt er ein I2C Interface, kann aber auch über eine herkömmliche UART kommunizieren. Die Kommunikation über das I2C-Interface ist aber deutlich einfacher und erfolgt identisch zum SCD30 über die SDA und SCL Signale. Auch er muss mit 5V versorgt werden, und hat nur 3.3V Signalpegel an den IO Pins. Daher sollte auch wieder ein 3.3V Arduino Entwicklungsboard, wie z.B. der DUE verwendet werden, wenn man sich separate das Umsetzen der Pegel ersparen will.

Abb. 1: Der DS-CO2-20 NDIR CO2-Sensor von Plantower

Die Pinbelegung kann Tabelle 1 entnommen werden. Um also das I2C Interface zu nutzen, sollte man den Pin 6 auf Ground legen. Den Reset (Pin5) sollte man vorsichtshalber über einen 4.7kOhm an VCC festklemmen. Der Sensor erzeugt intern zwar alle Sekunde kontinuierlich einen neuen Messwert, dieser wird allerdings stark gefiltert, und hängt damit auch von den vorigen Messwerten ab. Unabhängig sind die Messwerte erst nach etwa 25 Sekunden.

PIN1	VCC	Positive power 5V
PIN2	GND	Negative power
PIN3	SDA/TX	I2C SDA with drive mode of OD
		Serial port sending pin/TTL level@3.3V
PIN4	SCL/RX	I2C SCL with drive mode of OD
		Serial port receiving pin/TTL level@3.3V
PIN5	RESET	Module reset signal /TTL level@3.3V, low reset.
PIN6	SELECT	Interface Mode Select/TTL level@3.3V
		High or Float: Uart Low: I2C
PIN7	PWM	1Hz, 200us high level per 1PPM

Tabelle 1: Pin-Out des DS-CO2-20 Sensors

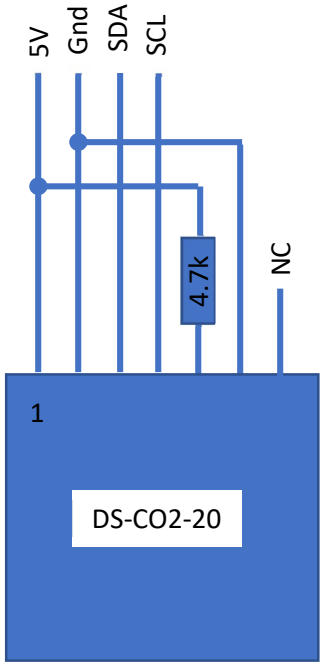

Abb. 2: Anschluss des DS-CO2-20 Sensors an die Stromversorgung und an den I2C-Bus

Für einen Vergleich des Sensirion SDC30 mit dem Plantower DS-CO2-20 bietet es sich an, das Adafruit Assembled Datalogging Shield (Product Id. 1141) auf den Arduino DUE aufzustecken (versehen mit Stackable Headers), so dass man die Messwerte des Sensors über längere Zeit auf eine SD-Karte aufzeichnen kann. Verwendet man gleichzeitig die Real Time Clock (RTC) auf dem Datalogging Shield, kann man zu jedem Messwert einen Zeitstempel mit auf die SD-Karte schreiben. Damit wird später ein Vergleich der Messwerte der Sensoren zu gleichen Zeitpunkten möglich, sofern man zuvor die RTC auf die richtige Uhrzeit justiert hat. Der Programmcode für jedes DUE Board ist im Anhang zu finden.

In dieser Untersuchung wurden zwei Arduino DUE Boards mit einen Datalogging Shield versehen und einmal der Sensirion SCD30 mit den I2C Anschlüssen des Arduino verbunden und das andere Mal der Plantower DS-CO2-20. Beim Sensirion Sensor war zusätzlich noch die Ampel mit 3 LEDs (grün, gelb, rot) zur Signalisierung an den Pins 5 6 und 7 angeschlossen. Für die Messung wurden beide Boards nebeneinander eine Nacht, einen Tag und wieder eine Nacht in einem Wohnraum betrieben. Während der Nacht stand der Wohnraum über eine offene Tür mit dem Schlafraum in Verbindung, ein Fenster

war nicht geöffnet. Nach dem Schlafen wurden Schlaf- und Wohnraum gelüftet, am Tag wurde zusätzlich gelüftet.

Die Analyse der Messdaten zeigt zunächst einmal eine sehr gute Übereinstimmung der Messwerte sowohl qualitativ wie quantitativ und liegt im Bereich der angegebenen Genauigkeiten. Man sieht deutlich, dass der DS-CO2-20 dem SCD30 um nichts nachsteht. Die Nacht mit gleichmäßig ansteigenden CO2-Werten durch das Schlafen sowie das Lüften bilden sich sehr deutlich bei beiden Sensoren ab. Tagsüber sind höhere Fluktuationen durch die Aktivitäten im Wohnraum und durch das Lüften zu erkennen. Dabei schwankt die CO2-Konzentration zwischen etwa 500ppm nach ausgiebigem, zweimaligem Lüften und etwa 1300ppm. Ohne Lüften würde der Maximalwert deutlich höher liegen, soviel kann man erkennen.

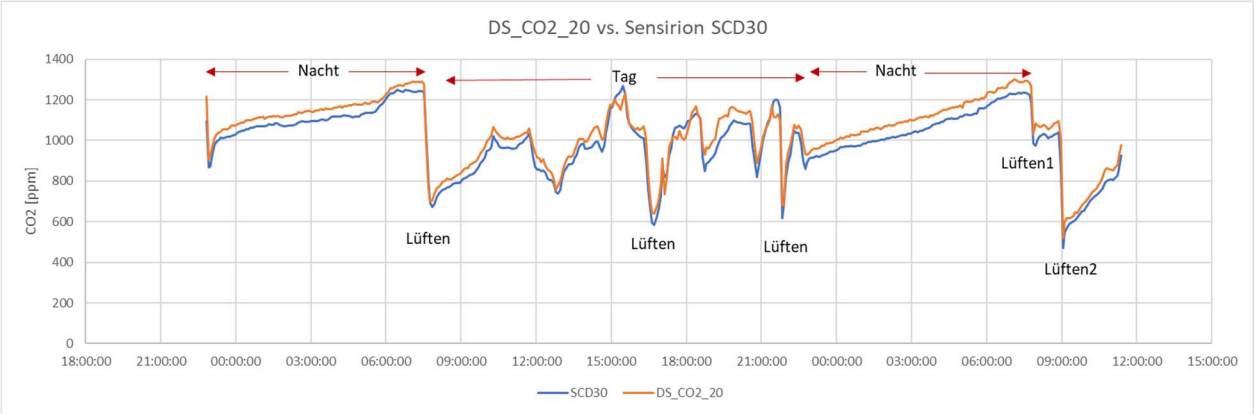

Abb. 3: Vergleich der Messergebnisse für CO2 (blau Sensirion SCD30, orange Plantower DS-CO2-20)

Insgesamt kann man daher sagen, dass auch der billigere chinesische Sensor genauso für eine CO2-Ampel geeignet ist, wie der Sensor der schweizerischen Firma (der vermutlich auch in China gefertigt wird). Allerdings hat der Sensirion Sensor noch zusätzlich einen Sensor für Temperatur und Feuchte auf dem PCB montiert, was auch ganz hilfreich ist und Kosten spart. Ein kleiner Nachteil dabei ist aber, dass Temperatur und Feuchte spürbar von der Wärmeenergie der übrigen Elektronik auf dem Board beeinflusst werden. Die Werte stimmen also nicht ganz mit denen eines guten, externen Sensors überein, sondern müssen über eine zusätzliche Kalibration korrigiert werden. Dies ließe sich aber mit zusätzlicher Software sicher bewerkstelligen.

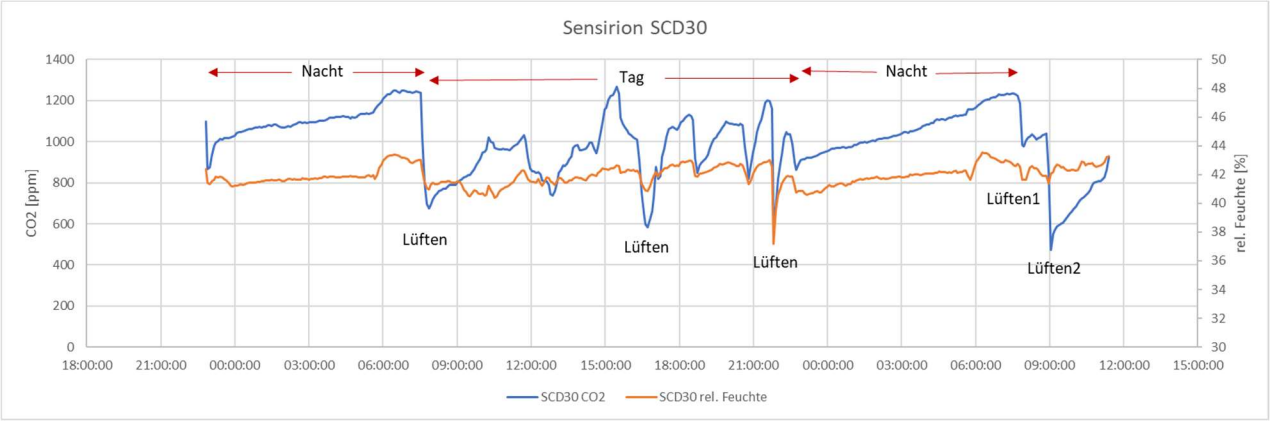

Abb. 4: Verlauf der CO2-Konzentration (blau) und der relativen Luftfeuchte (orange) beim Sensirion Sensor

Literatur

Eine Arduino-basierte CO2-Luftgüte-Ampel unter Verwendung eines kostengünstigen NDIR-Sensors zur Covid-19 Prävention; Bernd Laquai; 31.10.2020

Anhang

Arduino Listings

Listing 1: SCD30 mit Datalogging Shield und Ampel

```
#define gn 5
#define ge 6
#define rt 7
#include <SPI.h>
#include <SD.h>
#include <RTClib.h>
#include <Wire.h>

#include "SparkFun_SCD30_Arduino_Library.h" //Click here to get the library:
http://librarymanager/All#SparkFun_SCD30
SCD30 airSensor;
int co2ppm;
float temp, rh;
RTC_PCF8523 rtc;
//RTC_DS1307 rtc;
DateTime now1;

char fileName[15] = "datalog.txt";
File myFile;

void setup()
{
  pinMode(gn, OUTPUT);    // sets the digital pin 13 as output
  pinMode(ge, OUTPUT);    // sets the digital pin 13 as output
  pinMode(rt, OUTPUT);    // sets the digital pin 13 as output
  Serial.begin(9600);
  Serial.println("SCD30 Example");
  Wire.begin();

  if (airSensor.begin() == false)
  {
    Serial.println("Air sensor not detected. Please check wiring. Freezing...");
    while (1)
      ;
  }

  pinMode(10, OUTPUT); // SD Card CS
  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    Serial.flush();
    abort();
  }

  if (! rtc.initialized()) {  //PCF8523
  //if (! rtc.isrunning()) {    //DS1307
    Serial.println("RTC is NOT running!");
    // following line sets the RTC to the date & time this sketch was compiled
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    // This line sets the RTC with an explicit date & time, for example to set
    // January 21, 2014 at 3am you would call:
    //rtc.adjust(DateTime(2017, 1, 21, 3, 0, 0));
  }
  //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

  Serial.begin(9600);          // start serial communication at 9600bps
  if (!SD.begin(10)) {
    Serial.println("SDcard not ready");
    delay(10000);
  }
  else
    Serial.println("SDcard ok");
```

```
  if (!SD.exists(fileName)) {
    myFile = SD.open(fileName, FILE_WRITE);
    myFile.println("###");
    myFile.flush();
  }
  else {
    myFile = SD.open(fileName, FILE_WRITE);
    myFile.println("-----");
    myFile.flush();
  }


  Serial.println("Setup done, now logging...");

}

void loop()
{
  if (airSensor.dataAvailable())
  {
    co2ppm = airSensor.getCO2();
    temp = airSensor.getTemperature();
    rh = airSensor.getHumidity();

    if (co2ppm < 1000) {
      digitalWrite(gn, HIGH);
      digitalWrite(ge, LOW);
      digitalWrite(rt, LOW);
    }
    else if ((co2ppm>=1000) && (co2ppm < 1250)) {
      digitalWrite(gn, HIGH);
      digitalWrite(ge, HIGH);
      digitalWrite(rt, LOW);
    }
    else if ((co2ppm>=1250) && (co2ppm < 1750)) {
      digitalWrite(gn, LOW);
      digitalWrite(ge, HIGH);
      digitalWrite(rt, LOW);
    }
    else if ((co2ppm>=1750) && (co2ppm < 2000)) {
      digitalWrite(gn, LOW);
      digitalWrite(ge, HIGH);
      digitalWrite(rt, HIGH);
    }
    else if (co2ppm>=2000) {
      digitalWrite(gn, LOW);
      digitalWrite(ge, LOW);
      digitalWrite(rt, HIGH);
    }

    //logging
    DateTime now = rtc.now();

    Serial.print(now.day());
    Serial.print('.');
    Serial.print(now.month());
    Serial.print('.');
    Serial.print(now.year());
    Serial.print(' ');
    Serial.print(now.hour());
    Serial.print(':');
    Serial.print(now.minute());
    Serial.print(':');
    Serial.print(now.second());
    Serial.print(' ');
    Serial.print(co2ppm);
    Serial.print(' ');
    Serial.print(temp);
    Serial.print(' ');
    Serial.print(rh);
    Serial.println();

    myFile.print(now.day(), DEC);
    myFile.print('.');
    myFile.print(now.month(), DEC);
    myFile.print('.');
    myFile.print(now.year(), DEC);
```

```
    myFile.print(' ');
    myFile.print(now.hour(), DEC);
    myFile.print(':');
    myFile.print(now.minute(), DEC);
    myFile.print(':');
    myFile.print(now.second(), DEC);
    myFile.print('\t');
    myFile.print(co2ppm);
    myFile.print('\t');
    myFile.print(temp);
    myFile.print('\t');
    myFile.print(rh);

    myFile.println();
    myFile.flush();

  }
  delay(500);
}
```

Listing 2: DS_CO2_20 mit Datalogging Shield

```
#include <SPI.h>
#include <SD.h>
#include <Wire.h>
#include <RTClib.h>

RTC_PCF8523 rtc;
//RTC_DS1307 rtc;
DateTime now1;

char fileName[15] = "datalog.txt";
File myFile;


int i=0;
int data[12];
void setup() {

  Wire.begin();                // join i2c bus (address optional for master)
  pinMode(10, OUTPUT); // SD Card CS
  if (! rtc.begin()) {
    Serial.println("Couldn't find RTC");
    Serial.flush();
    abort();
  }


  if (! rtc.initialized()) {  //PCF8523
  //if (! rtc.isrunning()) {    //DS1307
    Serial.println("RTC is NOT running!");
    // following line sets the RTC to the date & time this sketch was compiled
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    // This line sets the RTC with an explicit date & time, for example to set
    // January 21, 2014 at 3am you would call:
    //rtc.adjust(DateTime(2017, 1, 21, 3, 0, 0));
  }
  //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));

  Serial.begin(9600);          // start serial communication at 9600bps
  if (!SD.begin(10)) {
    Serial.println("SDcard not ready");
    delay(10000);
  }
  else
    Serial.println("SDcard ok");

  if (!SD.exists(fileName)) {
    myFile = SD.open(fileName, FILE_WRITE);
    myFile.println("###");
    myFile.flush();
  }
  else {
    myFile = SD.open(fileName, FILE_WRITE);
    myFile.println("-----");
    myFile.flush();
```

```
  }


  Serial.println("Setup done, now logging...");
}

int reading = 0;

void loop() {
  int j;
  int csum = 0;
  int CO2ppm = 0;
  //Serial.println(i);
  //request reading from sensor
  Wire.requestFrom(0x08, 12);    // request 2 bytes from slave device #8
  //reading from sensor
  if (12 <= Wire.available()) { // if two bytes were received
    for (j = 0; j <12; j++) {
      data[j] = Wire.read();
    }
    //Serial.println(data[0],HEX); // start char 1
    //Serial.println(data[1],HEX); // start char 2

    reading = data[2];  // Frame length high
    reading = reading << 8;
    reading |= data[3]; // Frame length low
    //Serial.println(reading);

    CO2ppm = data[4];  // CO2 ppm high
    CO2ppm = CO2ppm << 8;
    CO2ppm |= data[5]; // CO2 ppm low
    //Serial.println(CO2ppm);

    reading = data[10];  // Check high
    reading = reading << 8;
    reading |= data[11]; // Check low
    //Serial.println(reading);

    for (j = 0; j <10; j++) {
      csum = csum+data[j];
    }
    if (reading != csum) {
      Serial.println("frame check error");
    }
    //logging
    DateTime now = rtc.now();

    Serial.print(now.day());
    Serial.print('.');
    Serial.print(now.month());
    Serial.print('.');
    Serial.print(now.year());
    Serial.print(' ');
    Serial.print(now.hour());
    Serial.print(':');
    Serial.print(now.minute());
    Serial.print(':');
    Serial.print(now.second());
    Serial.print(' ');
    Serial.print(CO2ppm);
    Serial.println();

    myFile.print(now.day(), DEC);
    myFile.print('.');
    myFile.print(now.month(), DEC);
    myFile.print('.');
    myFile.print(now.year(), DEC);
    myFile.print(' ');
    myFile.print(now.hour(), DEC);
    myFile.print(':');
    myFile.print(now.minute(), DEC);
    myFile.print(':');
    myFile.print(now.second(), DEC);
    myFile.print('\t');
    myFile.print(CO2ppm);

    myFile.println();
    myFile.flush();
```

```
  }
  else {
    Serial.println("nothing received");
  }
  i = i+1;
  delay(1500);
}
```