Eine Arduino-basierte CO2-Luftgüte-Ampel mit Datenaufzeichnung auf SD-Karte

Bernd Laquai, 12.12.20

Die in /1/ beschriebene CO2-Lüftgüte-Ampel liefert anhand der 3 LEDs grün-rot-gelb eine hilfreiche Information über den CO2-Anteil in der Raumluft auf den ersten Blick und damit über die Notwendigkeit zu Lüften. Da der CO2-Anteil nachgewiesenermaßen deutlich zur Aerosolkonzentration in der Luft korreliert, wobei die vom Menschen ausgeatmeten Aerosole mit Viren beladen sein können, stellt die CO2-Luftgüte-Ampel einen wichtigen Beitrag zur Corona-Prophylaxe dar. Allerdings kann auch der zeitliche Verlauf der CO2-Konzentration zusammen mit Temperatur und Feuchte eine interessante Information sein, dann z.B., wenn bestimmte Zusammenhänge zwischen der CO2-Konzentration und gewissen Varianten des Lüftungsverhaltens analysiert werden sollen oder zwischen der CO2-Konzentration und anderen Raumluftqualitäts-Faktoren ein Bezug vermutet wird. So liegt es nahe, dass in einem Gebiet mit hohem Radon-Potential in der Bodenluft die Radonkonzentration und die CO2-Konzentration in der Raumluft ebenfalls gut korreliert sind, und damit das von einer CO2-Ampel beeinflusste Lüftungsverhalten gleichzeitig auch sicherstellen kann, dass die Radon-Konzentration in der Raumluft nicht auf kritische Werte ansteigt.

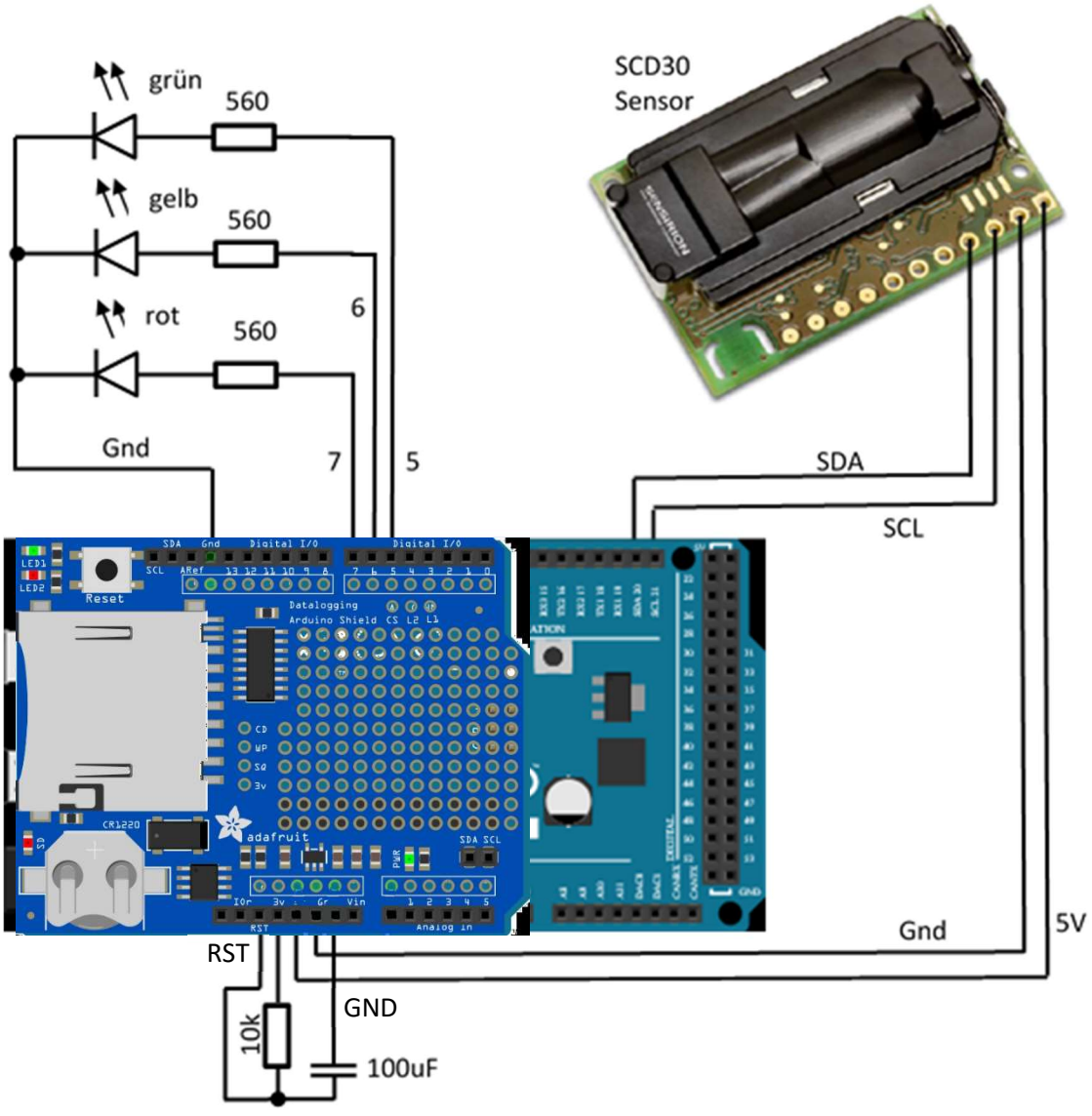

Abb. 1: Der schematische Aufbau der CO2-Luftgüteampel mit Daten-Aufzeichnungsfunktion

Für solche Fälle kann die in /1/ beschriebene CO2-Luftqualitäts-Ampel so modifiziert werden, dass sie zusätzlich die vom Sensor gelieferten CO2 Konzentrationswerte sowie die Temperatur- und Luftfeuchtewerte zusammen mit einem Zeitstempel auf eine SD-Karte aufzeichnet. Dazu kann ein Data Logger Shield verwendet werden. Dies kann direkt auf den Arduino aufgesteckt werden und ermöglicht nicht nur die Aufzeichnung der Daten auf SD-Karte, sondern beinhaltet auch eine Real Time Clock (RTC), mit welcher der Zeitstempel generiert werden kann. Es handelt sich dabei um eine Quarzuhr mit Stützbatterie, welche das Datum und die Uhrzeit bereitstellt und welche die Zeit- und Datums-information auch fortschreibt, wenn die Versorgungsspannung abgetrennt wird. Diese RTC wird einmal beim Kompilieren des Programms auf die aktuelle Uhrzeit eingestellt, danach wird noch einmal ohne das Einstell-Kommando (rtc.adjust) kompiliert, so dass die Uhr von da an mit der RTC-internen Zeit läuft. Für den Betrieb der SD-Karte wird die SPI-Schnittstelle des Arduino genutzt und die in der Arduino Entwicklungsumgebung bereits vorhandene Bibliothek SD verwendet, welche die notwendigen IO-Befehle zur Verfügung stellt. Die RTC benötigt zum Kompilieren eine zusätzliche vom Hersteller bereitgestellte Bibliothek.

Sehr populär ist die neue Variante des Data Logger Shield von Adafruit (Product ID 1141), welches die RTC vom Typ PCF8523 beinhaltet. Es funktioniert auch auf den 3.3V Arduino Boards wie dem Arduino DUE und macht damit weitere Bauteile unnötig. Es kann statt mit den normalen Stiftleisten mit sogenannten „Stackable Headers“ ausgestattet werden, so dass die Buchsen des Arduino oben auch wieder zur Verfügung stehen. Dann kann dieses Shield einfach auf den Arduino DUE aufgesteckt werden und der Sensor, die LEDs und die Bauteile für den verzögerten Reset genauso wieder eingesteckt werden, wie bei der CO2-Ampel ohne Daten-Aufzeichnungsfunktion in /1/. In Abb. 1. Ist der Aufbau schematisch und in Abb. 2 nach dem Einbau ins Gehäuse gezeigt. Von außen sehen beide CO2-Ampeln, mit und ohne Aufzeichnungsfunktion identisch aus.

Abb. 2: Der Aufbau der Komponenten im Innern des Gehäuses

Das Arduino Programm legt beim Start, sofern noch nicht vorhanden, dann eine Datei mit dem Namen DATALOG.TXT an, in die das Datum und die Zeit durch ein Leerzeichen getrennt und dann CO2-, Temperatur und relative Feuchte-Werte durch Tabs getrennt bei jedem Schleifendurchlauf in einer Zeile abgespeichert werden. Damit können die Daten sehr einfach in Excel in durch Tabs getrennte Spalten importiert und dort ausgewertet werden.

```
###
29.10.2020 21:20:52   1336  22.70 51.83
29.10.2020 21:20:55   1482  22.81 52.91
29.10.2020 21:20:57   1544  22.84 52.39
29.10.2020 21:20:59   1560  22.87 51.68
29.10.2020 21:21:2    1572  22.91 51.30
29.10.2020 21:21:4    1555  22.97 50.96
29.10.2020 21:21:6    1540  23.00 50.71
29.10.2020 21:21:9    1511  23.05 50.46
29.10.2020 21:21:11   1470  23.09 50.22
```

Tabelle 1: Dateiformat für die Daten, welche auf die SD-Karte geschrieben werden, die Spalten enthalten Datum und Zeit, CO2 in ppm, Temperatur in °C und relative Luftfeuchte in %

Literatur

/1/ Eine Arduino-basierte CO2-Luftgüte-Ampel unter Verwendung eines kostengünstigen NDIR-Sensors zur Covid-19 Prävention, http://opengeiger.de/Corona/CO2LuftgueteAmpel.pdf

/2/ Adafruit Assembled Data Logging shield for Arduino, https://www.adafruit.com/product/1141

Anhang

Listing mit Logger und Twinkle

```
#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"
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);
  pinMode(ge, OUTPUT);
  pinMode(rt, OUTPUT);
  Serial.begin(9600);
  Serial.println("SCD30 CO2 Logger");
  Wire.begin();
  Wire.setClock(50000L);

  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(3000);
    delay(3000);
  }
  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) {
      twinkle();
      digitalWrite(gn, HIGH);
      digitalWrite(ge, LOW);
      digitalWrite(rt, LOW);
    }
    else if ((co2ppm>=1000) && (co2ppm < 1250)) {
      twinkle();
      digitalWrite(gn, HIGH);
      digitalWrite(ge, HIGH);
      digitalWrite(rt, LOW);
    }
    else if ((co2ppm>=1250) && (co2ppm < 1750)) {
      twinkle();
      digitalWrite(gn, LOW);
      digitalWrite(ge, HIGH);
      digitalWrite(rt, LOW);
    }
    else if ((co2ppm>=1750) && (co2ppm < 2000)) {
      twinkle();
      digitalWrite(gn, LOW);
      digitalWrite(ge, HIGH);
      digitalWrite(rt, HIGH);
    }
    else if (co2ppm>=2000) {
```

```
      twinkle();
      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(10000);
  digitalWrite(gn, LOW);
  digitalWrite(ge, LOW);
  digitalWrite(rt, LOW);
}

void twinkle() //shows activity
{
      digitalWrite(gn, LOW);
      digitalWrite(ge, LOW);
      digitalWrite(rt, LOW);
      delay(100);
      return;
}
```