A CO_2 monitor as an introductory microelectronics project helping to slow-down the spread of the corona virus and ensuring a healthy learning and working environment

A. Köhn-Seemann, alf.koehn@gmail.com

2020-09-20

Abstract

This paper describes the setup of a simple yet reliable CO_2 monitor which is based on open-source microelectronics hardware. The monitor is intended to be used in class rooms, lecture halls or offices and can be constructed as a joint students project. It was motivated by recent discussions on the role of aerosols being part of exhaled air to spread the corona virus. The aerosol concentration in air is correlated with the CO_2 concentration. Measuring the latter can thus help to slow-down the spread of the corona-virus. The program code used for the CO_2 monitor and this documentation is available as a GitHub repository to allow to updates and improvements.

1 Introduction

It is generally accepted that the CO_2 concentration in a class room has an influence on students' activities, their ability to study and learn [1, 2], or on their health and thus attendance [3]. The same applies of course to office environments [4]. The major source of CO_2 in a class room is the exhaled air of the students (and teachers). It thus increases over time but can also be relatively easy controlled by proper ventilation. Monitoring the CO_2 concentration over time provides thus a simple way to ensure an productive and healthy learning environment.

In addition to CO_2 , exhaled air consists of aerosols (among other things). In preliminary studies, it has been recently discovered that the aerosols of patients being infected with Sars-CoV-2, might contain viable virus concentrations which are large enough to cause further infections if somebody else inhales those aerosols [5–7]. With half-life periods of the virus on aerosols on the order of 1 hour [8], it becomes evident that proper ventilation, strongly reducing the aerosol concentration, can help to prevent hidden infections, i.e. infections where the infected person is not (yet) aware of their infection but already contagious. Since aerosols and CO_2 are both parts of exhaled air, measuring the CO_2 concentration in a room provides an easy accessible approximation for the aerosol concentration [9].

Here we present a simple and cost effective, yet reliable way to monitor the CO_2 concentration. Widely available microelectronic components are used which can be easily programmed via open source software platforms allowing to modify and extend the example presented in this paper. Students can build the detectors in class as a joint project which might serve to raise interest in electronics or the underlying physical and chemical processes [10].

This work was inspired by a project of the *Hochschule Trier* [11], where the design and construction of a CO_2 measuring device is suggested as a students' project, allowing to discuss a variety of scientific topics during the course of the project. In addition, a few posts from different forums served as an inspiration [12–15]. Furthermore, a small number of GitHub repositories using the same CO_2 are available [16–18] (we would like to recommend the interested reader in particular to the repository by paulvha [18] as it contains a rather large number of examples).

2 The CO_2 monitor

The CO_2 monitor is based on the microelectronic sensor SCD30 which measures the CO_2 concentration and also provides measurements of the ambient temperature and relative humidity [19]. Using Arduino as a programing language and some microcontroller, it is straightforward to get the sensor running and outputting data, thanks to the examples available in the libraries provided by SparkFun [20]. Using the Arduino IDE [21], which is available for all major operating systems, the corresponding libraries can be simply included via the library manager.

To make the CO_2 monitor visually appealing, we decided to output the measurement to an OLED display (which is also very cheap and available in a large variety of sizes and configurations). Due to the widespread usage of such displays, they can also be directly included via the library manager in the Arduino IDE. In addition to just showing some numbers, we have included a red LED which lights up as soon as some threshold value of the CO_2 concentration is reached, indicating the need for ventilation. One could also think of a traffic light design, where first a yellow LED lights up at a slightly lower threshold value. The *Federation of European Heating, Ventilation and Air Conditioning associations (REHVA)* recommend to issue a warning, corresponding to an orange light, when a value of 800 ppm is reached and prompt to trigger some action like ventilation, corresponding to a red light, when 1000 ppm are reached [22]. The *Federal Ministry of Labour and Social Affairs* of Germany also states a threshold value of 1000 ppm that should not be passed [23]. Note that a value of 400 ppm is the typical CO_2 concentration of air.

As controller we decided to use the low-cost open source NodeMCU board [24], as it offers enough flexibility to further extend the functionality of the CO_2 monitor. Of particular interest might be the WiFi capability allowing for example to write the measured values to a web-server where they can then accessed via a web-browser or an app on a smartphone.

A prototype of the CO_2 monitor is shown in Fig. 1. As one can see, it is not enclosed in some box to still allow easy access for modifications. The idea of this prototype was rather to show that the general principle of the CO_2 monitor is working and not to provide a polished final product. The prototype is ready to be used in a class room or lecture hall, although it might be worth to mount everything into a box which is not only visually more appealing but provides also some protection.

3 Required parts

The CO_2 monitor as presented here consists of a number of parts for which it is not important to use the exact same model. The only component which should not be replaced is the CO_2 measuring device, the SCD30. Note that the program code discussed in Sec. 6 is tailored for the NodeMCU ESP8266, replacing that component would thus require small adjustments to the code.

The parts used for the prototype of the CO_2 monitor are listed in Table 3. The display can be easily replaced by an OLED of larger size. One could also use multiple displays, which would require to take care of proper addressing the display and thus add a little bit of complexity to the code (and the assembly).

The usage of a breadboard was motivated by educational purposes as this allows very easy assembly without the need to solder anything. It can, however, directly be replaced by a stripboard or completely omitted and use

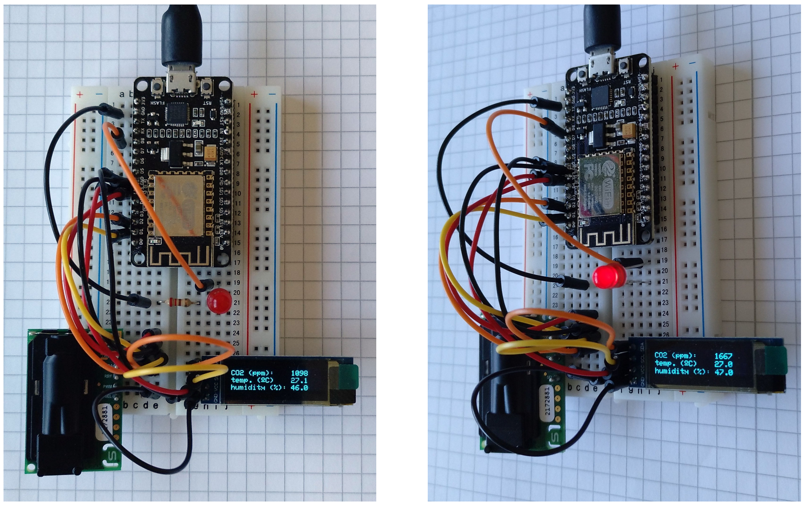

Figure 1: Assembled and working prototype of the CO_2 monitor, *(left)* with a measured CO_2 concentration below the threshold and *(right)* above it (note the red LED).

only cables or pin headers (which would require some soldering).

Note that the prices as listed in the table can be pushed down (significantly for some of the components) when ordering larger quantities.

For the prototype design of the CO_2 monitor we have decided to leave out a proper casing. One could either use a standard-sized case, or design one and print it for example on a 3D printer or re-use/recycle some old boxes. It is however important to correctly position the SCD30 inside the box: as described in a manufacturer's document [25], the sensor is ideally placed as close as possible to the box's outer shell and to a large opening to be properly exposed to the ambient. The box should be as small as possible to get fast response times to changes in the ambient air. The SCD30 should also be isolated from direct air flow, as the corresponding changes in pressure (due to the air flow) would lead to increased noise and thus reduced accuracy in the measurements. It is also recommended to not directly place the sensor above heat sources like for example microcontrollers.

4 The CO_2 sensor

The SCD30 has been chosen because it performs direct measurements of the CO_2 concentration. Cheaper sensors often measure the concentration of volatile organic compounds (VOC) and then assume a correlation between the two quantities. This can, however, lead to wrong values of the

Element	Quantity	Price
SCD30 (CO_2 sensor)	1	60 €
NodeMCU EPS8266	1	8 €
0.91″ OLED display	1	5 €
red LED	1	0.2 €
220 Omega resistor	1	.1 €
mini breadboard	1	4 €
breadboard cables	10	4 €
pin header	1	0.5 €
micro USB cable	1	3 €

Table 1: Components used for the CO_2 monitor as presented in this paper (note that the prices were obtained in 09/2020 and may vary).

CO_2 concentration since VOC can be emitted from a variety of chemicals. Although VOCs are also known to cause health problems, here we are explicitly interested in the CO_2 concentration, as discussed in Sec. 1. For a discussion about monitoring VOC and CO_2 concentration with self-assembled devices we would like to point the interested reader to e.g. Ref. [26].

4.1 Technical specifications

According to the datasheet of the SCD30 [19], the CO_2 sensor has a measurement range of $0 - 40,000$ ppm with an accuracy of ± 30 ppm. The supply voltage needs to be between 3.3 and 5 V which allows to use a variety of microcontrollers. The drawn current is specified to be on average 19 mA with a maximum value of 75 mA. With a sensor lifetime of 15 years, the SCD30 offers a reliable system to permanently monitor the CO_2 concentration.

4.2 Nondispersive infrared technique

The CO_2 concentration is measured using the so-called *nondispersive infrared* technique (NDIR). It is the most common sensor type used in industry to measure the CO_2 concentration. It's principle is sketched in Fig. 2. A light source emits infrared light which travels through a tube filled with a sample of the surrounding air. The spectrum of the emitted light includes the 4.26 μm absorption band of CO_2 which is unique to the typical components of air and the light is absorbed by them. At the end of the tube, the remaining light hits an optical filter that allows only that specific wavelength of 4.26 μm to pass. A detector then collects the remaining light. The difference between

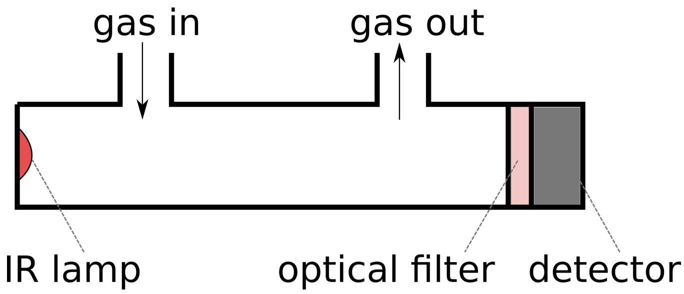

Figure 2: Sketch of a sensor using the nondispersive infrared technique to measure CO_2 concentration.

the amount of light emitted by the source and received by the detector is due to the CO_2 molecules in the tube which then allows to calculate the CO_2 concentration.

Using folded optics, i.e. waveguides, for the tube and diodes for the infrared source and detector, allows to small sizes of the overall sensor on the order of just a few centimeters.

5 Assembly

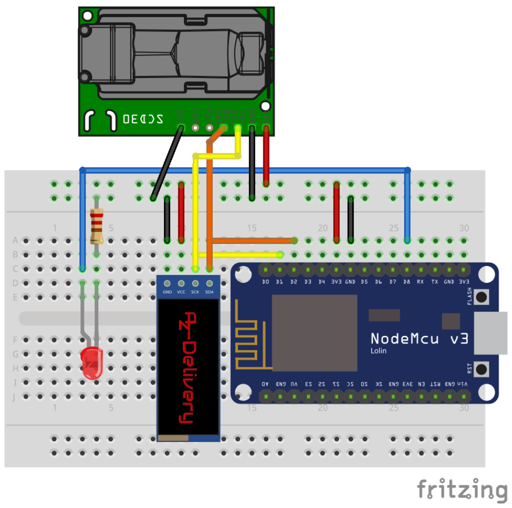

Figure 3: Schematic of a prototype of the CO_2 monitor.

The CO_2 monitor can be assembled in various ways, here we will restrict ourselves to the case of a simple prototype design on a breadboard as shown in Fig. 3. The connection between the NodeMCU (with the ESP8266) and the SCD30 sensor is as follows:

NodeMCU		SCD30
GND	⟶	GND
3.3 V	⟶	VIN
D2/GPIO4	⟶	RX/SDA
D1/GPIO5	⟶	TX/SCL
GND	⟶	SEL

The NodeMCU then needs to be connected to the OLED display as follows:

NodeMCU		OLED display
GND	⟶	GND
3.3 V	⟶	VCC
D2/GPIO4	⟶	SDA
D1/GPIO5	⟶	SCL

It is of course also possible to directly connect the respective `SDA` and `SCL` pins of the OLED and the SCD30, as shown in Fig. 3, instead of connecting those pins between the SCD30 and the NodeMCU. The red LED is connected with its anode, the longer leg, to pin `D8/GPIO15` of the NodeMCU and with its cathode, the shorter leg, via a 220 Omega resistor (to limit the current) to ground.

6 The program code

Arduino is used as programming language in this project due to its widespread usage and large numbers of libraries available for various hardware components. The Arduino IDE library manager allows to directly install a proper Arduino library for the SCD30, alternatively the library is available as a GitHub repository [20]. For a tutorial on how to install libraries within the Arduino IDE, see Ref. [27]. As for the NodeMCU and the OLED display, the Arduino IDE library manager is able to provide the required libraries.

The source code for the CO_2 monitor as described in this paper is available on GitHub [28], in order to be able to update and extend it. Nevertheless, we have also included the code in this paper, to provide a complete description of the project. The include statements of the code are listed in Listing 1. The `Adafruit_GFX.h` and `Adafruit_SSD1306.h` libraries are used for the

OLED display and are required to be installed via the library manager of the Arduino IDE beforehand (alternatively, they are also available on GitHub [29] for manual installation). Note that the display size in pixels needs to be set correctly and can vary. The `SparkFun_SCD30_Arduino_Library.h` also needs to be installed via the library managed (or manually from the GitHub repository [20]).

```
#include <Wire.h>                  // for I2C communication
#include <Adafruit_GFX.h>          // for writing to display
#include <Adafruit_SSD1306.h>      // for writing to display
#include "SparkFun_SCD30_Arduino_Library.h"

// activate debugging
//   true:  print info + data to serial monitor
//   false: serial monitor is not used
#define DEBUG true

#define CO2_CRITICAL 1500          // threshold for warning
#define WARNING_DIODE_PIN D8       // NodeMCU pin for red LED

#define MEASURE_INTERVAL 2         // seconds, minimum: 2

#define SCREEN_WIDTH 128           // OLED display width in pixels
#define SCREEN_HEIGHT 32           // OLED display height in pixels

// OLDE reset pin, 4 is default (-1 if sharing Arduino reset pin)
// using NodeMCU, we have to use LED_BUILTIN
#define OLED_RESET LED_BUILTIN
// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

SCD30 airSensor;
```

Listing 1: Loading the required libraries.

Listing 2 shows the setup of the code, where the serial monitor is initialized, followed by the diode, the OLED display, and finally the SCD30.

```
void setup() {
  if (DEBUG == true) {
    // initialize serial monitor at baud rate of 115200
    Serial.begin(115200);
    Serial.println("Using SCD30 to get: CO2 concentration, temperature, humidity");
  }

  Wire.begin();

  // initialize LED pin as an output
  pinMode(WARNING_DIODE_PIN, OUTPUT);

  // SSD1306_SWITCHCAPVCC: generate display voltage from 3.3V internally
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) { // Address 0x3C for 128x32
    if (DEBUG == true)
      Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

```

```
  // Show initial display buffer contents on the screen --
  // the library initializes this with an Adafruit splash screen.
  display.display();              // initialize display
                                  // library will show Adafruit logo
  delay(2000);                    // pause for 2 seconds
  display.clearDisplay();         // clear the buffer
  display.setTextSize(1);         // has to be set initially
  display.setTextColor(WHITE);    // has to be set initially

  // move cursor to position and print text there
  display.setCursor(2,5);
  display.println("SCD30 test program");
  display.println("twitter.com/formbar");

  display.display();              // write display buffer to display

  // turn warning LED on and off to test it
  digitalWrite(WARNING_DIODE_PIN, HIGH);
  delay(2000);
  digitalWrite(WARNING_DIODE_PIN, LOW);

  // initialize SCD30
  // SCD30 has data ready every two seconds
  if (airSensor.begin() == false) {
    if (DEBUG == true)
      Serial.println("Air sensor not detected. Please check wiring. Freezing...");
    while (1)
      ;
  }
}
```

Listing 2: Setup code.

The main code is given in Listing 3. First, the data is obtained from the SCD30 sensor and then passed to a function outputting it to the serial monitor and then to another function, printing it on the OLED display. Listings 4 and 5 show the code for the two latter functions.

```
void loop() {

  float
    co2,
    temperature,
    humidity;

  if (airSensor.dataAvailable()) {
    // get data from SCD30 sensor
    co2         = airSensor.getCO2();
    temperature = airSensor.getTemperature();
    humidity    = airSensor.getHumidity();

    // print data to serial console
    if (DEBUG == true)
      printToSerial(co2, temperature, humidity);

    // print data to OLED display
    printToOLED(co2, temperature, humidity);
  }
  else
```

```
    if (DEBUG == true)
      Serial.println("Waiting for new data");

  // if CO2-value is too high, issue a warning
  if (co2 >= CO2_CRITICAL) {
    digitalWrite(WARNING_DIODE_PIN, HIGH);
  } else {
    digitalWrite(WARNING_DIODE_PIN, LOW);
  }

  // SCD30 has new data every 2 seconds
  delay(MEASURE_INTERVAL*1000);
}
```

Listing 3: Main loop which is executed repeatedly.

```
void printToSerial( float co2, float temperature, float humidity) {
  Serial.print("co2(ppm):");
  Serial.print(co2, 1);
  Serial.print(" temp(C):");
  Serial.print(temperature, 1);
  Serial.print(" humidity(%):");
  Serial.print(humidity, 1);
  Serial.println();
}
```

Listing 4: Function which prints data to the serial console.

```
void printToOLED( float co2, float temperature, float humidity) {
  int
    x0, x1;            // to align output on OLED display vertically

  x0  = 2;
  x1  = 86;

  display.clearDisplay();
  display.setCursor(x0,5);
  display.print("CO2 (ppm):");
  display.setCursor(x1,5);
  // for floats, 2nd parameter in display.print sets number of decimals
  display.print(co2, 0);

  display.setCursor(x0,15);
  display.print("temp. ( C)");
  display.setCursor(x0+7*6,15);
  display.write(167);
  display.setCursor(x1,15);
  display.print(temperature, 1);

  display.setCursor(x0,25);
  display.print("humidity (%):");
  display.setCursor(x1,25);
  display.print(humidity, 1);

  display.display();
}
```

Listing 5: Function which prints data to the OLED display.

Acknowledgments

The author is indebted to the efforts of the open-source software community.

References

[1] D. Twardella, W. Matzen, T. Lahrz, R. Burghardt, H. Spegel, L. Hendrowarsito, A. C. Frenzel, and H. Fromme. Effect of classroom air quality on students' concentration: results of a cluster-randomized cross-over experimental study. *Indoor Air*, 22:378, 2012, doi:10.1111/j.1600-0668.2012.00774.x.

[2] S. Gaihre, S. Semple, J. Miller, S. Fielding, and S. Turner. Classroom Carbon Dioxide Concentration, School Attendance, and Educational Attainment. *J. Sch. Health*, 85:569–574, 2014, doi:10.1111/josh.12183.

[3] D. G. Shendell, R. Prill, W. J. Fisk, M. G. Apte, D. Blake, and D. Faulkner. Associations between classroom co_2 concentrations and student attendance in washington and idaho. *Indoor Air*, 14:333, 2004, doi:10.1111/j.1600-0668.2004.00251.x.

[4] J. G. Allen, P. MacNaughton, U. Satish, S. Santanam, J. Vallarino, and J. D. Spengler. Associations of Cognitive Function Scores with Carbon Dioxide, Ventilation, and Volatile Organic Compound Exposures in Office Workers: A Controlled Exposure Study of Green and Conventional Office Environments. *Environmental Health Perspectives*, 6:805, 2016, doi:10.1289/ehp.1510037.

[5] J. A. Lednicky, M. Lauzardo, Z. Hugh Fan, A. S. Jutla, T. B. Tilly, M. Gangwar, M. Usmani, S. N. Shankar, K. Mohamed, A. Eiguren-Fernandez, C. J. Stephenson, Alam. Md. M., M. A. Elbadry, J. C. Loeb, K. Subramaniam, T. B. Waltzek, K. Cherabuddi, J. G. Morris Jr., and C.-Y. Wu. Viable sars-cov-2 in the air of a hospital room with covid-19 patients. *medRxiv*, 2020, doi:10.1101/2020.08.03.20167395.

[6] L. Morawska and D. K. Milton. It is time to address airborne transmission of covid-19. *Clinical Infectious Diseases*, 2020, doi:10.1093/cid/ciaa939.

[7] M. A. Kohanski, L. J. Lo, and M. S. Waring. Review of indoor aerosol generation, transport, and control in the context of covid19. *International Forum of Allergy & Rhinology*, 2020, doi:10.1002/alr.22661.

[8] N. v. Doremalen, T. Bushmaker, D. H. Morris, M. G. Holbrook, A. Gamble, B. N. Williamson, A. Tamin, J. L. Harcourt, N. J. Thornburg, S. I. Gerber, J. O. Lloyd-Smith, E. de Witt, and V. J. Munster. Aerosol and Surface Stability of SARS-CoV-2 as Compared with SARS-CoV-1. *N. Engl. J. Med.*, 382:1564–1567, 2020, doi:10.1056/NEJMc2004973.

[9] A. Hartmann and M. Kriegel. Risk assessment of aerosols loaded with virus based on co_2-concentration. *Preprint server of the Technische Universität Berlin*, 2020, doi:10.14279/depositonce-10362.

[10] B. Watzka and R. Girwidz. Ist die Luft zu schlecht zum Lernen? Nichtdispersive IR-CO2-Gassensoren im Physikunterricht. *Physik und Didaktik in Schule und Hochschule*, 10:22–33, 2011.

[11] Hochschule Trier Umwelt-Campus Birkenfeld. Website: Covid-19 Prävention: CO2-Messung und bedarfsorientierte Lüftung (accessed: 2020-09-13). `https://www.umwelt-campus.de/forschung/projekte/iot-werkstatt/ideen-zur-corona-krise`.

[12] hackster.io (user Brian Boyles). Website: Homemade CO2 Sensor Unit (accessed: 2020-09-13). `https://www.hackster.io/bfboyles/homemade-co2-sensor-unit-22a9d8`.

[13] Arduino Forum (user metropol). Website: Arduino forum: Co2 level classroom sensor (accessed: 2020-09-13). `https://forum.arduino.cc/index.php?topic=702131`.

[14] Home Assistant Forum (user Ombra). Website: Co2 monitoring with scd30 sensor and ha integration (accessed: 2020-09-13). `https://community.home-assistant.io/t/co2-monitoring-with-scd30-sensor-and-ha-integration/216708`.

[15] Raspberry Pi Forum (user Zentris). Website: In Entwicklung: Co2-Sensor (SCD30) mit ESP8266 und Display SSD1306 (kurze Vorstellung) (accessed: 2020-09-13). `https://forum-raspberrypi.de/forum/thread/44717-in-entwicklung-co2-sensor-scd30-mit-esp8266-und-display-ssd1306-kurze-`

[16] Kaspar Metz. Coro$_2$sens. `https://github.com/kmetz/coro2sens`, 2020.

[17] Netzbasteln. Co$_2$narienvogel. `https://github.com/netzbasteln/co2narienvogel`, 2020.

[18] paulvha. paulvha scd30 library. https://github.com/paulvha/scd30, 2020.

[19] Sensirion: The Sensor Company. Datasheet sensirion scd30 sensor module (version 0.94). https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/9.5_CO2/Sensirion_CO2_Sensors_SCD30_Datasheet.pdf.

[20] SparkFun Electronics. SparkFun SCD30 CO2 Sensor Library. https://github.com/sparkfun/SparkFun_SCD30_Arduino_Library, 2020.

[21] The Arduino team. Software: The open-source arduino ide. https://www.arduino.cc/en/main/software.

[22] Ventilation The Federation of European Heating and Air Conditioning associations (REHVA). Website: Rehva covid-19 guidance document (version 2020-08-03). hhttps://www.rehva.eu/activities/covid-19-guidance/rehva-covid-19-guidance.

[23] Federal Ministry of Labour and Social Affairs Germany. Website: SARS-CoV-2-Arbeitsschutzregel (version 2020-08-20). https://www.bmas.de/DE/Presse/Meldungen/2020/neue-sars-cov-2-arbeitsschutzregel.html.

[24] NodeMCU. Website: Nodemcu documentation (accessed: 2020-09-18). https://nodemcu.readthedocs.io/en/master/.

[25] Sensirion: The Sensor Company. Design-in guidelines for scd30 (version 0.2). https://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/9.5_CO2/Sensirion_CO2_Sensors_SCD30_Assembly_Guideline.pdf.

[26] G. Chiesa, S. Cesari, M. Garcia, M. Issa, and S. Li. Multisensor IoT Platform for Optimising IAQ Levels in Buildings through a Smart Ventilation System. *Sustainability*, 1:5777, 2019, doi:10.3390/su11205777.

[27] SparkFun Electronic. Website: Installing an Arduino Library. https://learn.sparkfun.com/tutorials/installing-an-arduino-library.

[28] Alf Köhn-Seemann. Co_2 monitor. https://github.com/alfkoehn/CO2_monitor, 2020.

[29] Adafruit Industries. Adafruit_ssd1306. https://github.com/adafruit/Adafruit_SSD1306, 2020.