Professorship of Environmental Sensing and Modeling
School of Computation, Information and Technology
Technical University of Munich

Air-quality Sensor Station

Course script - Practical course **Air-quality Sensor Station**

Prof. Dr.-Ing Jia Chen

Version: 1.5

Last modified: 08.05.2024

Autors: Daniel Kühbacher/ Julian Bärtschi

Content

Prerequisites and Course Schedule

Please watch our introduction video before joining the course, since it provides a comprehensive overview of the topic. Apart from that, no further preparation is necessary. The course is held on Monday and lasts from **10:00 to 13:00**.

We will cover the following topics in the practical course:

- Introduction to environmental sensing; Recap of the introduction video
- Commissioning of the sensor boxes
- Air quality measurement
- Evaluation and assessment of the collected data

Sensor Station

A measurement station was developed specifically for this course. Figure 1 depicts the basic structure of the measurement station. It uses a Raspberry Pi processing unit and implements a 7'' touchscreen display on the front. It has four sensors that measure NO2 (Nitrogen Dioxide), CO (Carbon Monoxide), O3 (ozone), and CO_2 (carbon dioxide). These gases are the main pollutants and greenhouse gases in cities next to PM_{10} (Particulate Matter) and are also officially measured by the LfU (Bayrisches Landesamt für Umwelt). As suggested by the manufacturer of the sensors, they are installed in a dedicated, ventilated measuring channel. The ventilator can also be controlled by the Raspberry and sucks fresh air into the channel on demand. During measurement, the ventilator is turned off to avoid fluctuations of the signal due to pressure fluctuations and to save energy.

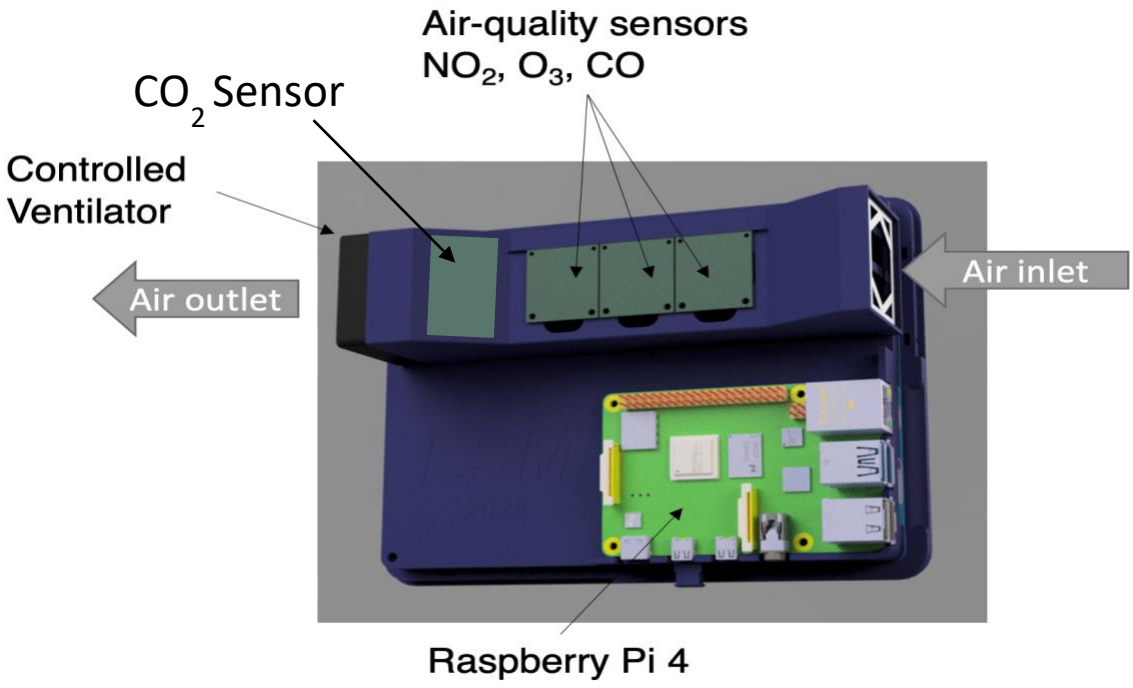

Figure 1 - Basic structure of the sensing station.

EC-Sense air-quality sensors

"TB600B series modules (see Figure 2) combine high-precision sensing technologies. The gas sensor is a small-in-size Solid Polymer Electrochemical Sensor from EC Sense, detecting very low concentrations of gases accurately and reliably. The module serves a UART digital output for ease of use and comes pre-calibrated eliminating the tedious work of calibration.
Solid polymer electrochemical technology is based on the principle of electrochemical catalytic reaction, detecting the output signals of the electrochemical reactions of different gases and accurately measuring the gas concentration through the signal.
The sensor is composed of three electrodes in contact with the electrolyte. A typical electrode consists of a large surface area of noble metal and other materials. The electrode, electrolyte, and the surrounding air are in contact, and the gas diffuses into the working electrode. Here, the gas will be oxidized; this causes a current, which is proportional to the gas concentration and can be measured."[1]

In Table 1 the main characteristics of the implemented sensors are listed. The rightmost column gives you a conversion factor between the measured value in ppm (parts per million) and the mass/ volume. This conversion depends on the molecular weight (g/mol) of the compared substances, the surrounding temperature, and the air pressure. Considering the ideal gas law $PV_m = RT$ at a temperature of 25°C at a pressure of 1 atm, we can derive a molar Volume $V_m = 0.02479\ m^3/mol$. Using the molar mass of, e.g. NO_2 $M_{NO2} = 46{,}0055\ g/mol$, we can derive the following equation:

$$concentration\left[\frac{\mu g}{m^3}\right] = \frac{M_{NO2}\left[\frac{g}{mol}\right] * concentration[ppm]}{V_m\left[\frac{m^3}{mol}\right]} * 10^3$$

Figure 2 - EC-Sense TB600B sensor modules

[1] TB600B-O3-5 Technical Specification – EC Sense

Table 1 - Overview of sensor characteristics

Sensor type	*Range [ppm]*	*Det-Limit [ppm]*	*Resolution [ppm]*	*FS-accuracy error*	*Conversion factor @ 25 °C/ 1 atm*
O_3	0-5	0,01	0.001	± 5% FS	1ppb = 1,96 µg/m^3
CO	0-10	0,1	0.01	0 – 5 ppm ± 2-5% 5 – 10 ppm ± 5%	1ppb = 1,15 µg/m^3
NO_2	0-5	0,05	0.001	± 5% FS	1ppb = 1,88 µg/m^3
	Temperature error: ± 0.2 °C Humidity error: ± 2%				

Cozir Low Power CO2 Sensor

" The CozIR®-LP3 is a low-power NDIR CO2 sensor using state-of-the-art solid-state LED optical technology. The low-power LEDs are manufactured in-house, giving GSS complete control of the CO2 sensor signal chain. The CozIR®-LP3's low power consumption is compatible with battery-powered operation, allowing the sensor to be used in a wide variety of applications, including wirelessly connected equipment. The CozIR®-LP3 operation is configurable depending on user requirements. On powerup, the CozIR®-LP3 automatically starts taking measurements. Measurements can be streamed or output on request. The LP3 also features a built-in autozero function that maintains CO2 measurement accuracy over the lifetime of the product." [2]

Figure 3 - Cozir-LP3-5000 sensor module

Sensor type	*Range [ppm]*	*Resolution [ppm]*	*FS-accuracy error*	*Conversion factor @ 25 °C/ 1 atm*
CO_2	0-5000	1	± (30 +3%rdg)	1ppm = 1,8 mg/m^3
	Pressure Dependence: ± 0.14% per mbar deviation from 1013 mbar			

[2] CozIR®-LP3 Data Sheet Rev 4.9 – Gas Sensing Solutions

Raspberry Pi 4 Model B - Peripherals

The Raspberry Pi 4 has 28 general-purpose input/output (GPIO) pins available.
These pins can be used as simple input or output pins but can also be used with other communication protocols, such as I2C, SPI, or UART. A GPIO can be used either as a digital input or digital output.

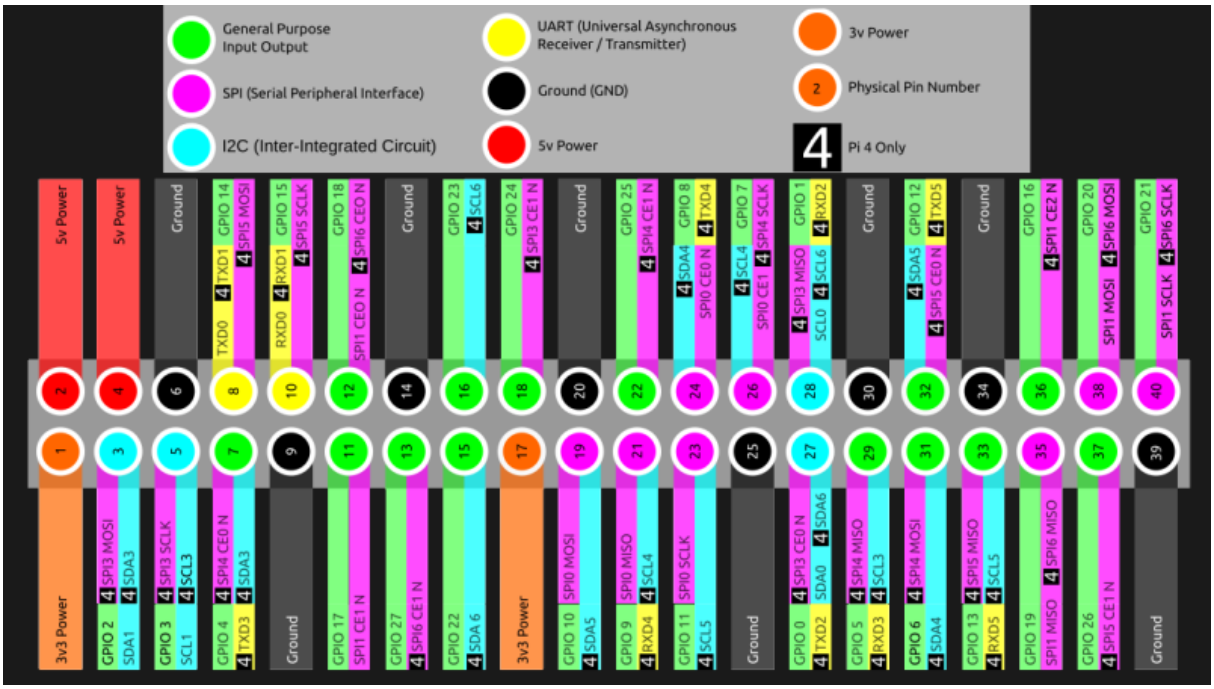

Figure 3 - RPi4 Pinout (https://hackaday.com/2022/02/01/did-you-know-that-the-raspberry-pi-4-has-more-spi-i2c-uart-ports/)

The following section explains the basic steps necessary to use a **GPIO** pin.

1. Import the Raspberry Pi Python module to control GPIOs
> import RPi.GPIO as GPIO

2. Set the pinout numbering. BOARD refers to the physical pinout of the Board (circled numbers in Figure 3). BCM refers to logical GPIO pins (GPIO X in Figure 3).
Example: Physical Pin 13 (BOARD) corresponds to GPIO 27 (BCM)
> GPIO.setmode(GPIO.BCM)

3. Set the direction of the pin (example with GPIO 27).
Output | Input:
> GPIO.setup(27, GPIO.OUT) > GPIO.setup(27, GPIO.IN)

4. Use the pin.
> GPIO.output(27, GPIO.HIGH) > GPIO.output(27, GPIO.LOW)

5. In case the pins are no longer needed, there is a function to reset **all** used ports.
> GPIO.cleanup( )

Further information can be found in the official documentation and the datasheet:
https://www.raspberrypi.com/documentation/

https://datasheets.raspberrypi.com/rpi4/raspberry-pi-4-datasheet.pdf

UART

Universal Asynchronous Receiver-Transmitter (UART) is a serial communication protocol. It allows bi-directional data transfer using only two wires (plus one for common GND). UART works by sending data as a stream of bits. One wire is used for transmitting (TX), and the other is used for receiving (RX). Beware that one device's RX is the other device's TX and vice versa.

The data Is transmitted as a series of pulses, where each pulse represents a single bit. A standard UART data frame usually consists of a start bit, 8 data bits, a parity bit, and a stop bit. Each of these bits Is sent sequentially, where the level (high or low) represents the value of the corresponding bit (1 or 0). A common baud rate is 9600 bit/s, but faster rates can be configured. These rates must match between the receiver and the transmitter.
Fortunately, it Is not necessary to implement this protocol by ourselves, but we can use the

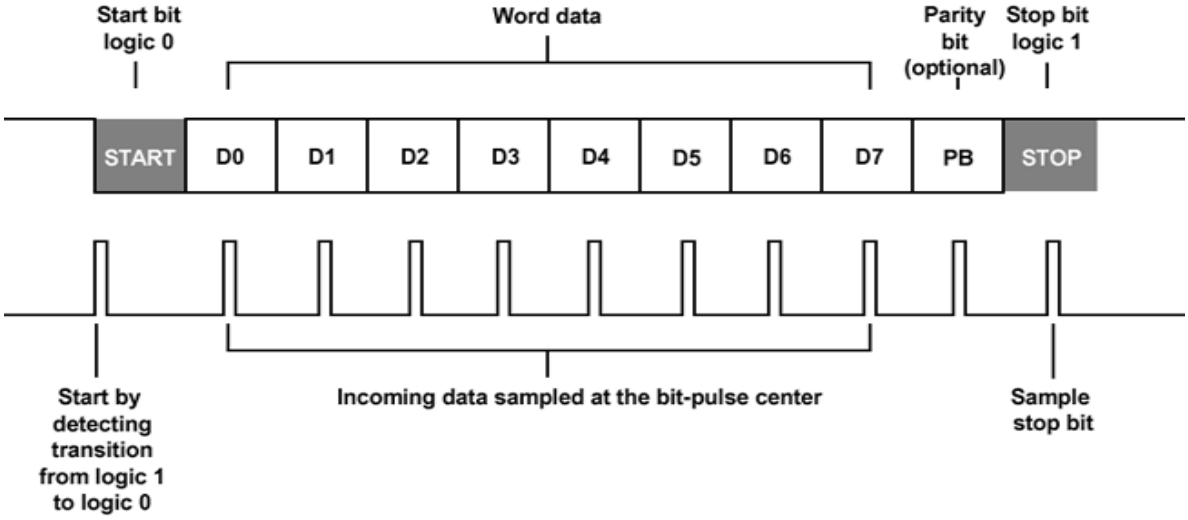

Figure 4 - A standard UART dataframe consists of 11 bits but these configurations may vary. (Source: https://www.seeedstudio.com/blog/2022/09/08/uart-communication-protocol-and-how-it-works/)

"serial" Python module. Here Is a code snipped of a simple serial configuration for basic usage:

```
> import serial
> ser = serial.Serial(port, baudrate = 9600, parity = serial.PARITY_NONE,
                      stopbits = serial.STOPBITS_ONE, bytesize = serial.EIGHTBITS, timeout = 1)
> reading = ser.read( )
```

Task 1 – Commissioning the sensor station.

In this task, you will set up the sensing station and test its functionalities. You should be able to read out the sensor values and control the ventilator with a simple Python script. Follow the steps below:

1. Open *ecsense.py* and have a look at the already-implemented class that controls the sensor. Since all sensors are from the same producer, they are identical to use. When you run the file, the content of the function `if __name__ == "__main__"` is executed.
    a. Have a look at the class and try to understand what each method does.
    b. In the constructor of the class, one connection setting is missing. Have a look into the datasheet of the sensor (folder documents) and find the missing baud-rate setting.
    c. The read command in the read method is missing. Look for the right command in the datasheet.
    d. Test the class by running it in the IDE. You should now get back some sensor values.
2. Now, since we can control and read one sensor, we want to apply this functionality to all three of them and control the ventilator. Task b is a bonus task, in case of extra time.
    a. Open the file *sensor_test.py* and implement the ventilator test as described in the code comments.
    b. (BONUS TASK) Use the classes EcSensor and CozirSensor to implement a readout for all sensors connected to the Raspberry Pi. Print sensor measurements to the console. Hint: Have a look at the files *cozir.py and ecsense.py to see the available functions.*
3. Finally, have a look at the file *local_db.py.* This runs what you did in step 3 in a loop and saves the data in a sqlite3 database. The image below shows a flowchart of the main measurement cycle.

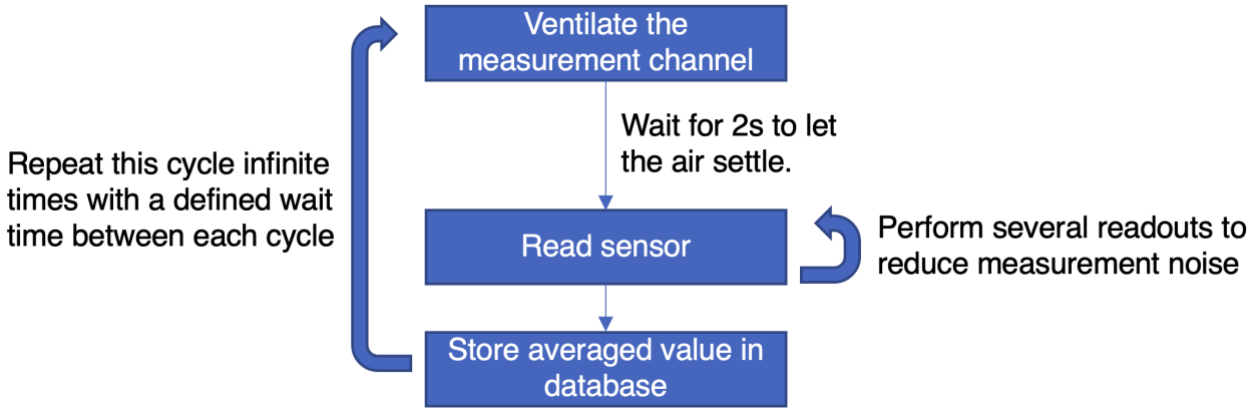

Figure 5 - Measurement cycle of the main program

Task 2 - Influence of the Location on the measured air quality

Recently, the topic of nitrogen oxide in major German cities has been a constant theme in the media. The public interest in precise measurements with high temporal and spatial resolution has therefore increased considerably. So far, however, there is no optimal method that combines all these requirements.

In the following experiment, we will evaluate the performance of our low-cost station and compare the acquired values at different distances to the street. In Figure 6 you can find three different positions with the following characteristics:

- Position 1 is close to a crossing, where cars and trucks tend to wait and accelerate when the light turns green.
- Position 2 is like Position 1 but on a spot that complies with the official regulation “Anlage 3C of the BImSchV”: height of about 1,5 – 4 m; > 25 m away from an intersection; < 10 m distance to the street.
- Position 3 is far off the street.

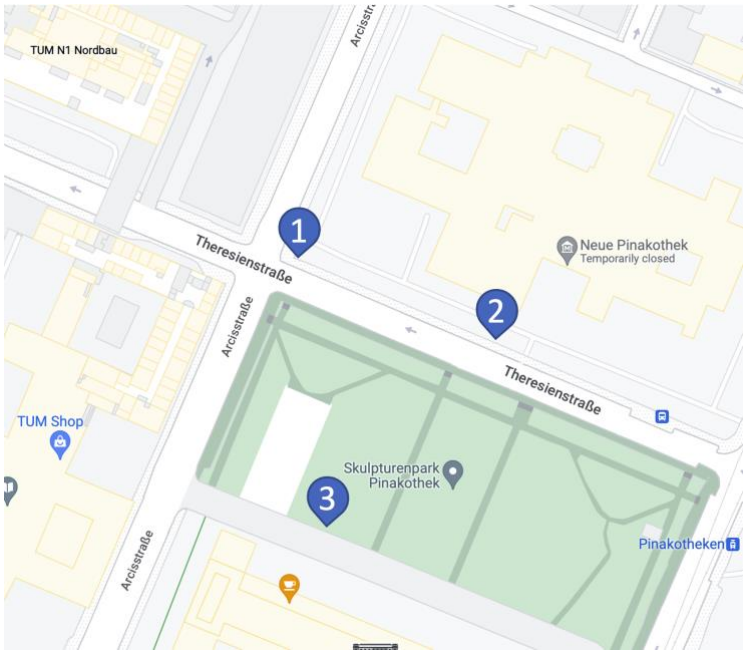

Figure 6 - Measurement positions for three sensors

Task 3 – Measure the air quality

In this task, we will measure the air quality in different locations next to the university.
Follow the steps below:

1. Mount the sensor on the tripod and prepare it for field measurements. Check if the sensor works by running your test script *sensor_test.py.*
2. Choose a location and start the measurement. Measure for at least 30 minutes.
3. During measurement, discuss the following questions with your partner:
    a. Do you expect different values at different locations? Why and what conclusion would you draw from it?
    b. Are the sensors a good choice for the measurements conducted? Have a look at Table 1 (Detection limit, Error) and compare the values to typical expected values in urban environments.
    c. Think about other use cases besides cities where air-quality measurements are necessary.

Task 4 - Evaluation and assessment of the acquired data

Finally, it is time to evaluate the acquired data and calculate some representative statistical values to compare. You will use Jupyter Notebook for this. Jupyter is an IDE used by many data scientists and in many labs around the world. It can be used with a variety of programming languages (we use Python) and runs in your browser. To start the notebook, open a terminal window and navigate to the *device_data* folder. Type in *jupyter notebook* to start the application. A browser window will open and you can start the notebook *analysis_notebook.ipynb.*

Task 4.1 – Analysis and comparison of the acquired data

Follow the steps described in the notebook.

Extra task – Analysis of LfU data (Landshuter Allee)

Start jupyter notebook in the folder *lfu_analysis.*
Follow the steps described in the notebook.