1. Workflow

1.1. Setting Up the Raspberry Pi and connecting the sensor

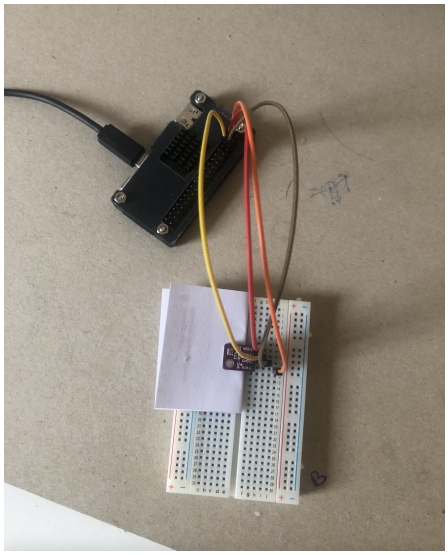
The raspberry pi was booted with the Raspberry Pi OS Lite software provided by raspberrypi.org. Therefore no initial software was installed and no GUI was provided. In order for the raspberry pi to recognize the i2c client, some configurations of the pi needed to be changed.

```
$ sudo raspi-config
```

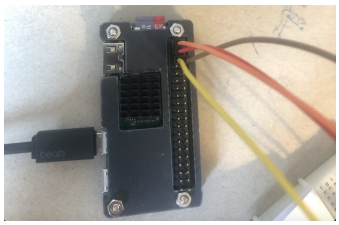
Choose 7 interface options, choose i2c and enable i2c interface.After this a reboot of the raspberry pi is neccessary. To work with the i2c bus the packages python-smbus and i2c-tools are needed.

```
$ sudo apt-get update
$ sudo apt-get install i2c-tools
$ sudo apt-get install python-smbus
```

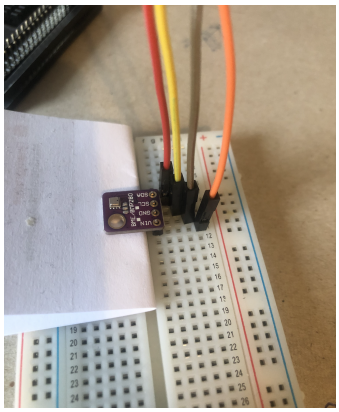
With the command lsmod it can be checked wheter the activation of the i2c has worked. The temperature and pressure sensor BME280 was connected to the raspberry pi using the breadboard as seen in figure 1a The sensor was not soldered to the pins, making the whole construction quite instable. The piece of paper was needed to hold the sensor in place so it could be recognized by the raspberry pi. The slightes movement of the cables or the sensor would cause the raspberry pi to not detect the sensor anymore (1.



(a) General set up of the Raspberry Pi connected to the BME280 sensor



(b) Set up of the cables at the Raspberry Pi



(c) Set up of the BME280 sensor on the breadboard

Figure 1: Set up of the hardware

With the i2c-tools function i2cdetect the connected sensor could be detected. The sensor should be detected on port 0x76 by default (see figure 2, if it does not show 76 it means that the sensor was not detected by the pi. To check whether the sensor is not only detected but can also pass data to the pi one can type the command i2cget. If the output is an error then something is wrong.

```
$ i2cdetect -y 1
$ i2cget -y 1 0x76
```

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                 --  --  --  --  --  --  --  --  --  --  --  --  --
10: --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20: --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30: --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40: --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50: --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60: --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70: --  --  --  --  --  --  76  --
```

Figure 2: Detection of the sensor at port 0x76

I decided to not read the data from scratch but to use one of the many

different provided libraries for this specific sensor. As there are many different libraries out it was in the beginning difficult to decide which one works best. The library that I decided was the best is from adafruit. The documentation can be found here: https://circuitpython.readthedocs.io/projects/bme280/en/latest/api.html .

2. Plotting Data

I wrote two programs that can be run with python3. One monitors the temperature data from the sensor continously creates a plot and only ends when the program is aborted by typing control+C. The other one monitors temperature, pressure, humidity and altitude for a specific amount of time and then produces a figure of all four datasets.

2.1. Continously monitoring temperature

The code is executed by calling python3 ¡filename.py¿. The code runs until it is aborted, during the running time it collects the temperature measured by the sensor and draws a graph from all the previously measured temperatures and the current temperatures, saves the graph and waits for 10 seconds.

Listing 1: Python example

```
import board
import busio
import adafruit_bme280
import matplotlib.pyplot as plt
import time

# Create library object using our Bus I2C port
i2c = busio.I2C(board.SCL, board.SDA)
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

xs= []
ys= []

#monitor continously
while True:
    xs.append(time.strftime("%Y,%m,%d,%H,%M,%S"))
    ys.append(bme280.temperature)

    #draw graph
    plt.plot(xs, ys)
    plt.xticks(rotation=45, ha='right')
    plt.title('BME280 temperature over time')
    plt.ylabel('Temperature (deg C)')
    plt.savefig('Temperature_cont.pdf')

    time.sleep(10)
```

The code could definitely be optimized, as it is probably not optimal that the program creates a new graph every time and overwrites the old graph. This code only continously monitors the temperature. By adapting the line ys.append(bme280.temperature) other information from the sensor could be read and monitored. Additionally the simple abort of the program sometimes, depending on when the program is ended with control+C, the figure created is damaged, as it might happen that the saving process is interrupted.

2.2. Monitoring Temperature, Humidity, Pressure, Altitude

The code below asks the user for how long the temperature, pressure, humidity and altitude should be monitored. Only integer values can be given as input. The program monitors the data and draws a plot of them.

Listing 2: Python example

```
import time
import board
import busio
import adafruit_bme280
import matplotlib.pyplot as plt

# Create library object using our Bus I2C port
i2c = busio.I2C(board.SCL, board.SDA)
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c)

# change this to match the location's pressure (hPa) at sea level
bme280.sea_level_pressure = 1013.25
temperature = []
humidity = []
pressure = []
altitude = []
current_time = []

mon_time = input("For how many minutes do you want to measure temperature,
pressure, humidity and altitude? Type only a number: ")
mon_time = int(mon_time)*60
counter = 0
while counter <(mon_time/2):
    temperature.append(bme280.temperature)
    humidity.append(bme280.relative_humidity)
    pressure.append(bme280.pressure)
    altitude.append(bme280.altitude)
    current_time.append(time.strftime("%H,%M,%S"))
    print("\nTemperature: %0.1f C" % bme280.temperature)
    print("Humidity: %0.1f %%" % bme280.relative_humidity)
    print("Pressure: %0.1f hPa" % bme280.pressure)
    print("Altitude = %0.2f meters" % bme280.altitude)
    counter = counter + 1
    time.sleep(2)
```

```
#draw the plot
fig, axs = plt.subplots(2, 2, sharex=True)
axs[0, 0].plot(current_time, temperature)
axs[0, 0].set_title('Temperature')
axs[0, 1].plot(current_time, humidity, 'tab:orange')
axs[0, 1].set_title('Humidity')
axs[1, 0].plot(current_time, pressure, 'tab:green')
axs[1, 0].set_title('Pressure')
axs[1, 1].plot(current_time, altitude, 'tab:red')
axs[1, 1].set_title('Altitude')

for ax in axs.flat:
    ax.set_xlabel('Timepoints of measurement', fontsize=5)
    ax.tick_params(labelrotation=45)

fig.savefig('Measurements.pdf')
```

Improvements of the code could be done by adapting the input so that one could also enter other numbers than just integers, for example 0.5 minutes of monitoring. Also the labelling of the x axis could be adapted, by adapting the fontsize.

3. Results

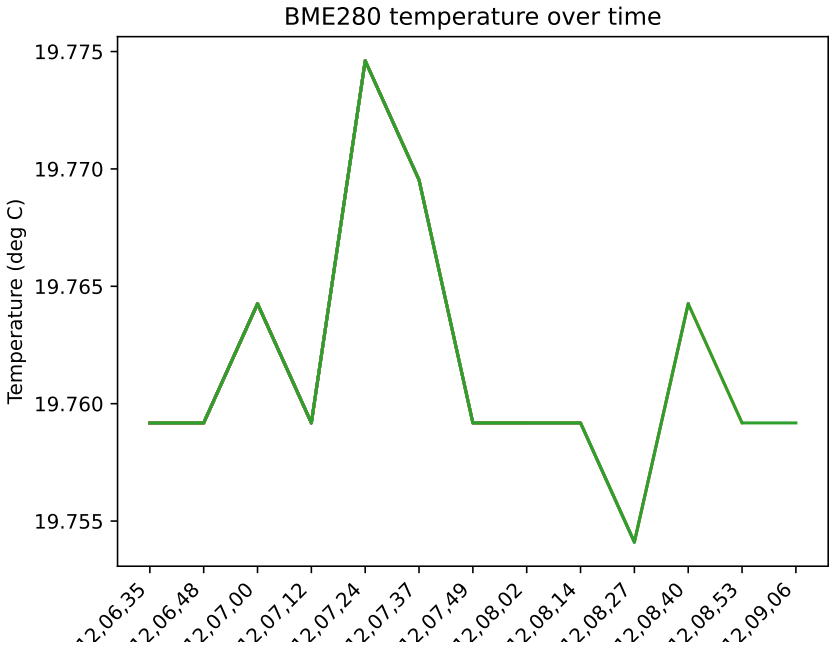


Figure 3: Continous monitoring of the temperature with the BME280 sensor

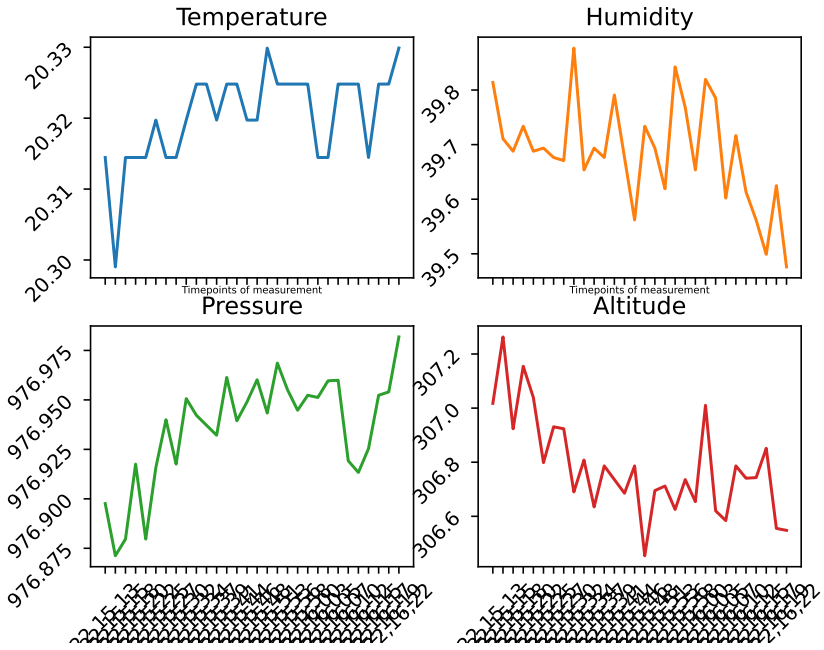


Figure 4: Plotting of temperature, humidity, pressure and altitude measured by the BME280 sensor

4. Learnings, Take Home Message

I installed the Raspian lite version as software on the Raspberry Pi. Therefore I had to manually install everything. In the beginning I also did not think about using conda during the whole process to manage my python packages. During the installation I ran into dependency problems that conda would have solved automatically, however I had to manually install all missing packages. This was kind of tedious so next time I would probably spend more time thinking about how to set up the Pi in the best way to avoid problems in the future. During my research how to access and process the data from the i2c bus, I realized that processing the raw data is quite hard and complex. Therefore I then decided to use a provided library that converts the raw data into readable and processed data. If the project would be continued with other students I would suggest to solder the sensor to the pins. I haven't done this and it took me a long time to adjust the sensor so that it is connected to the Pi and also submits data. The slightest movement deconnected the sensor and I had to reconnect it.