Finanziert von der Europäischen Union
Funded by the European Union

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.

Erasmus+
„Strato-Ballon Measurement an Environmental Consciousness“
2024-1-DE02-KA210-VET-000243591

Material for students and teachers allowing for the preparation of measuring equipment. Developed and used as part of the project implementation

https://euballon.zslp.edu.pl/

1)BME 280 Pressure, humidity and temperature sensor

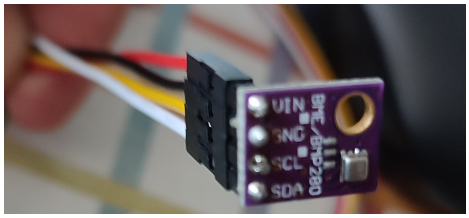
Experiment:
Measurement of atmospheric pressure, air temperature and air humidity

Required equipment:

-PicoBox (Raspberry Pi Pico 2 with 2.4" SSD1306 OLED screen)
-BME280 3.3V sensor
-Grove cable with female ends

Software:

MicroPython system installed on Pico in version 1.25 or newer,
Thonny Editor v.4 or newer,
MicroPython script presented below.



Wiring:

BME280 sensor pin	Grove cable colors
VCC	red
GND	black
SCL	yellow
SDA	white

Connect Grove cable plug to PicoBox socket GP6,GP7

MicroPython scripts for Raspberry Pi Pico
Simple version

The second line imports the BME280 sensor library.
The third line imports the sleep method from the module

```
1  from machine import I2C, Pin
2  from bme280 import BME280
3  from time import sleep
4  bus = I2C(1, scl=Pin(7), sda=Pin(6))
5  bme = BME280(i2c=bus, address=0x76)
6  print("temp. C, pressure [hPa],  humidity %")
7  while True:
8      #You can select which parameters are plotted by Plotter
9      print(bme.values[0],bme.values[1],bme.values[2])
10     #time between measurements (2 seconds)
11     sleep(2)
```

The fourth line defines the I2C bus to which we connect the sensor (GP6, GP7)

The fifth line creates the **bme** variable, which will contain the results of the temperature measurement (bme.values[0]), pressure (bme.values[1]) and humidity (bme.values[2])

The sleep method allows us to set the interval between measurements. In the presented example, sleep(2) is 2 seconds, but usually this interval should be longer, also due to the size of the recording file and the limited capacity of Pico memory

Version with logger and OLED SSD1306 screen

This version saves the measurement results in a CSV text file), which can be a source of data for further analysis, for example using Excel. In our case, we used Python with the MatPlotlib library installed

In line 8 You can set filename for data log, in our case this is **pressuretemp.CSV**

This file is open in mode "a"- append. If PicoBox will be restarted, then new lines are added to log file.

This part is executed only once

```
from machine import I2C, Pin
from bme280 import BME280
from time import sleep
import ssd1306

bus =I2C(1, scl=Pin(7),sda=Pin(6), freq=400000)
bme = BME280(i2c=bus, address=0x76)
file=open("presssuretemp.CSV","a")
file.write("Measurement EUBallon BME280 sensor \n")
file.write("No"+ "\t"+"temp.C"+"\t"+"pressure[hPa]"+"\t"+"hum.[%]"+"\n")
file.flush()
no=1
#oled screen resolution
WIDTH = 128
HEIGHT = 64
#Pinout for Raspberry Pi Pico-Oled screen and BME280 sensor
#connected to pin GP6(SDA) and GP7(SCL)
#both connected to the same BUS I2C
#variable oled assigned to oled screen
oled =ssd1306.SSD1306_I2C(WIDTH, HEIGHT,bus)
#clear screen
oled.fill(0)
oled.contrast(200)
print("temp. C, pressure [hPa],  humidity %")

```

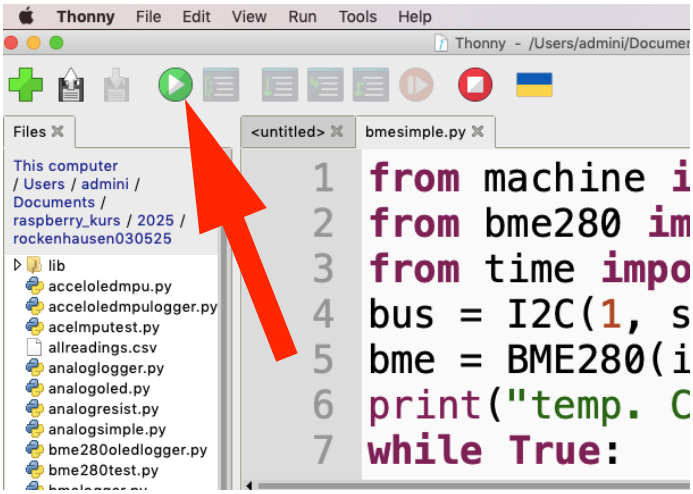
Infinite loop

```
while True:
    #You can select which parameters are plotted by Plotter
    print(bme.values[0],bme.values[1],bme.values[2])
    file.write(str(no)+ "\t"+str(bme.values[0])+"\t"\
    +bme.values[1]+"\t"+str(bme.values[2])+"\n")
    no=no+1
    #save to the file without closing file:
    file.flush()

    oled.text("EUballon",30,0)
    oled.text("Temp C:"+bme.values[0],0,15)
    oled.text("Press[hPa]:"+bme.values[1],0,30)
    oled.text("Hum.[%]:"+bme.values[2],0,45)
    #and now drawing above lines- must use show method!
    oled.show()
    #time between measurements (2 seconds)
    sleep(2)
    oled.fill(0)
```

If You want use this system without PC, than save the script on Pico with filname: **main.py**.

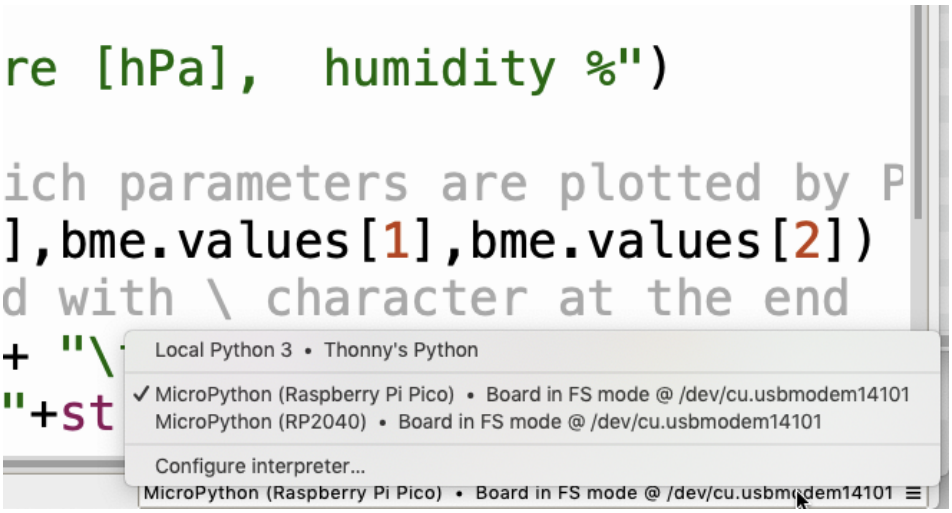
In this case Your script will be automatically executed when PicoBox is turned on.

Running the script in the Thonny environment



Before running the script, make sure you have selected the correct MicroPython interpreter.

In the lower right corner of the window, select "Micropython Raspberry Pi Pico"
If this interpreter does not appear, then the problem is that there is no connection between Pico and the computer, see below:



On our website we share our Python3 scripts which produces graphs using Matplotlib based on files with measurement results taken during a stratospheric balloon flight on May 7, 2025 in Rockenhausen

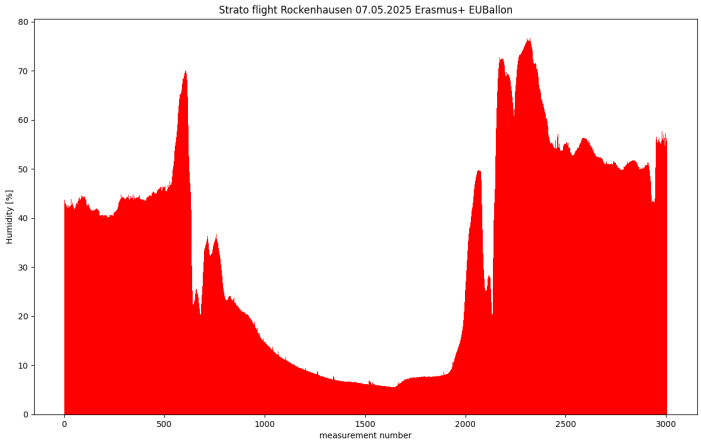
In a separate document we discuss these results in detail.

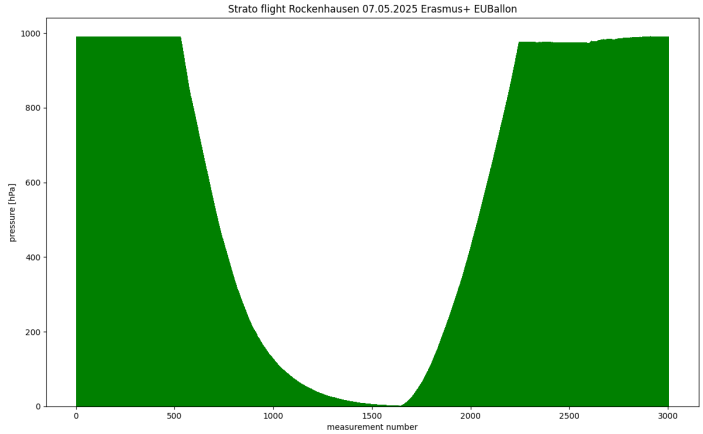
"Analysis of measurement results taken during a stratospheric balloon flight. Recorded data and analysis methods." We share also our

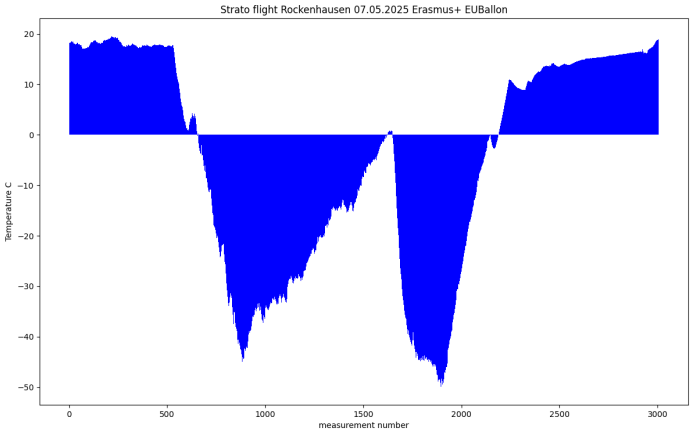
Below is a CSV file with measurement results made by Pico and BME280 using the presented script

1839	-44.77	165.34	7.75
1840	-44.68	166.83	7.66
1841	-44.92	168.25	7.77
1842	-44.21	169.53	7.74
1843	-45.01	171.16	7.76

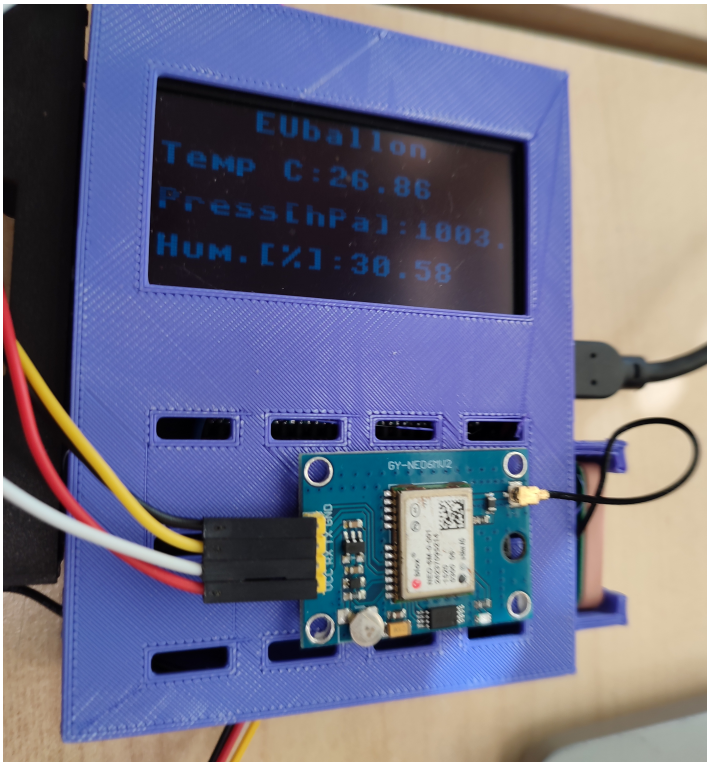
Graphs generated by our own Python3 scripts based on CSV files saved in Pico







Our PicoBox with Raspberry Pi Pico 2 and BME280 sensor

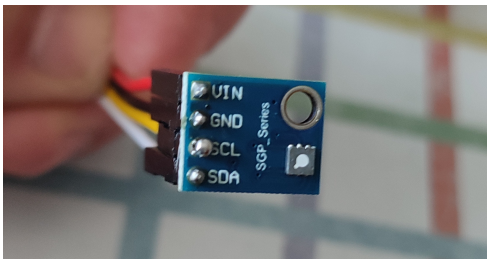


2)SGP30 CO2 TVOC organic pollution

Experiment:
Testing the content of CO2 and volatile organic compounds in the air

Required equipment:
-PicoBox (Raspberry Pi Pico 2 with 2.4" SSD1306 OLED screen)
-SGP30 I2C 3.3V sensor
-Grove cable with female ends

Software:
MicroPython system installed on Pico in version 1.25 or newer,
Thonny Editor v.4 or newer,
MicroPython script presented below.



Wiring:

SGP30 sensor pin	Grove cable colors
VCC	red
GND	black
SCL	yellow
SDA	white

Connect Grove cable plug to PicoBox socket GP6,GP7

CO2 ppm (particle per milion)

TVOC ppb (particle per bilion)

Note: SGP30 sensor requires several minutes for initialization and is not ready to measure immediately

MIcroPython scripts for Raspberry Pi Pico

Simple version

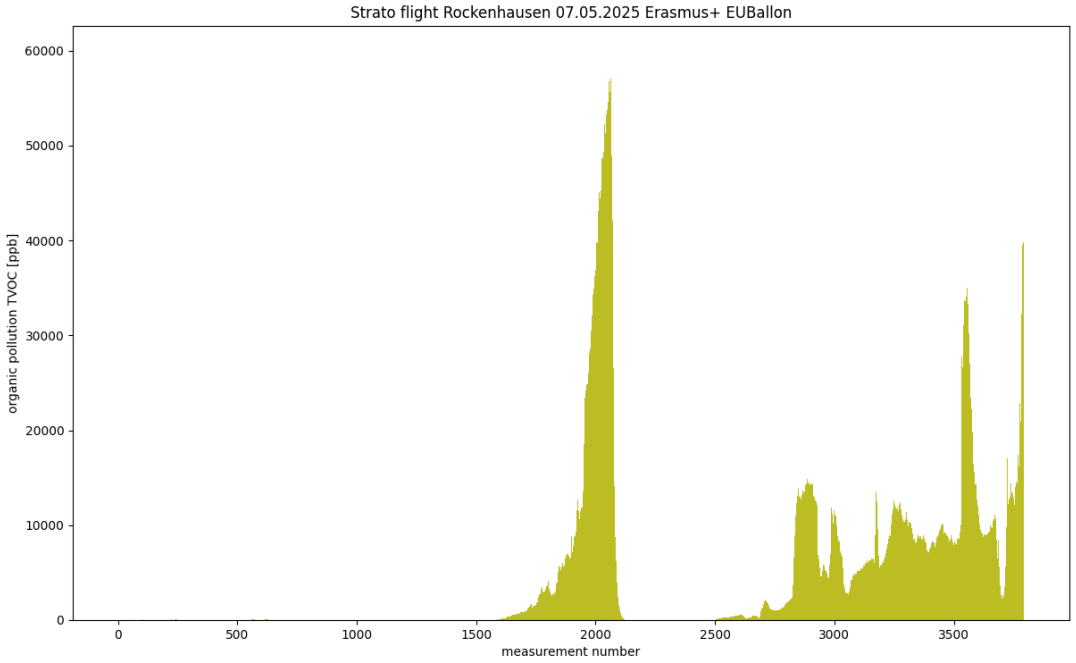
```
#Adafruit library for SGP30 CO2 and TVOC sensor
import adafruit_sgp30
from machine import I2C, Pin
from time import sleep

bus =I2C(1, scl=Pin(7),sda=Pin(6), freq=400000)
#SGP sensor connected to GP6(SDA Pin) and GP7(SCL Pin)
sgp = adafruit_sgp30.Adafruit_SGP30(bus)

while True:
    co2eq, tvoc = sgp.iaq_measure()
    if co2eq == 400:
        print("initializing mdodule, please wait!")
    else:
        while True:
            try:
                co2eq, tvoc = sgp.iaq_measure()
                print("CO2 concentration = %d ppm \t TVOC = %d ppb" % (co2eq, tvoc))
                #delay before next reading in seconds:
                sleep(0.5)
            except OSError as e:
                pass


```

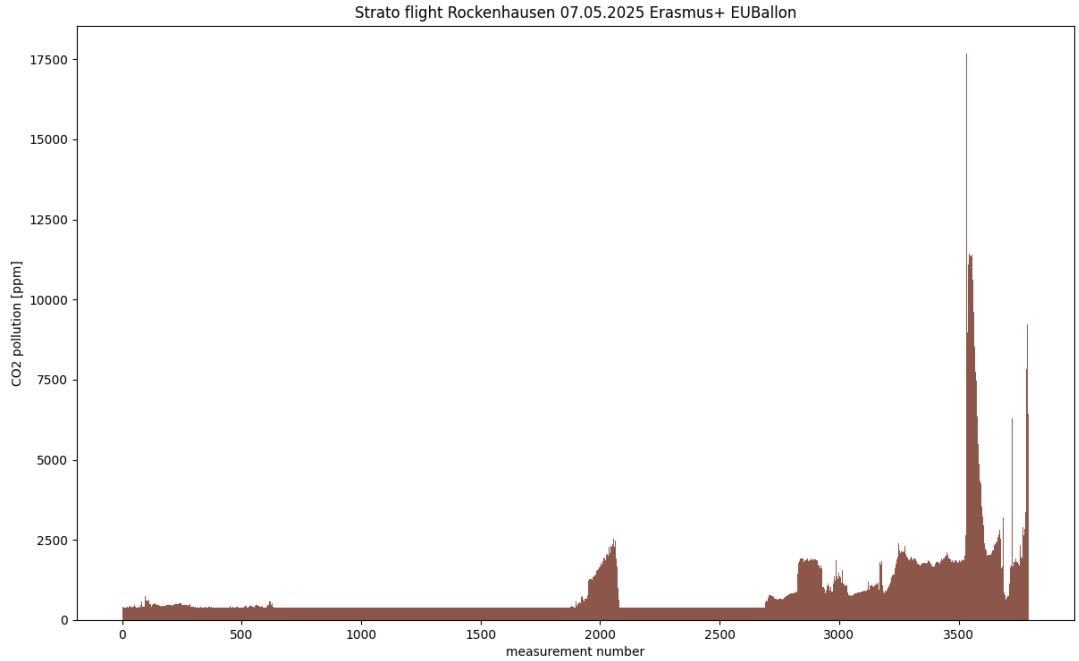
Version with OLED screen and data logger (result saved inside Raspberry Pi Pico)

```
#Adafruit library for SGP30 CO2 and TVOC sensor
#oled screen library:
import ssd1306
import adafruit_sgp30
from machine import I2C, Pin
from time import sleep
#screen resolution
WIDTH = 128
HEIGHT = 64

bus =I2C(1, scl=Pin(7),sda=Pin(6), freq=400000)
#SGP sensor and OLED screen connected to GP6(SDA Pin) and GP7(SCL Pin)
# variable sgp assigned to SGP30 sensor
sgp = adafruit_sgp30.Adafruit_SGP30(bus)
# variable oled assigned to OLED screen, initialisation
oled=ssd1306.SSD1306_I2C(WIDTH, HEIGHT,bus)
file=open("CO2TVOCresult.CSV","a")
file.write("Measurement EU-Ballon CO2 and TVOC sensor \n")
file.write("No"+ "\t"+"CO2"+"\t"+"TVOC"+"\n")
file.flush()
no=1
#clear screen:
oled.fill(0)
#maximum brightness contrast(255), usually more then 100
oled.contrast(200)
oled.text("EU-ballon",30,0)
oled.text("Air Quality",0,15)
oled.text("Measurement",0,30)
oled.show()
sleep(3)
```

```
oled.fill(0)
while True:
    co2eq, tvoc = sgp.iaq_measure()
    if co2eq == 400:
        print("initializing module, please wait!")
        oled.text("EUballon",30,0)
        oled.text("initializing...",0,15)
        oled.text("please wait!",0,30)
        oled.show()
        sleep(1)
        oled.fill(0)
    else:
        while True:
            try:
                co2eq, tvoc = sgp.iaq_measure()
                print("CO2 concentration = %d ppm \t TVOC = %d ppb" % (co2eq, tvoc))
                oled.text("EUballon",0,0)
                oled.text("org. pollution",0,15)
                oled.text("CO2 ppm:"+str(co2eq),0,30)
                oled.text("TVOC ppb:"+str(tvoc),0,42)
                oled.show()
                file.write(str(no)+"\t"+str(co2eq)+"\t"+str(tvoc)+"\n")
                no=no+1
                #save to the file without closing file:
                file.flush()
                #delay before next reading in seconds:
                sleep(1)
                oled.fill(0)
            except OSError as e:
                pass
```

Charts generated from a csv file with measurement data taken during a balloon flight in Rockenhausen on 07.05.2025



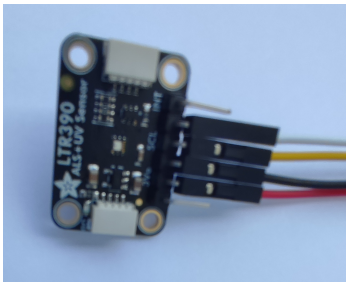


3)UV Ltr390 sensor simple version

Experiment:
UV radiation measurement

Required equipment:
-PicoBox (Raspberry Pi Pico 2 with 2.4" SSD1306 OLED screen)
-LTR390 UV Adafruit 3.3V sensor
-Grove cable with female ends

Software:
MicroPython system installed on Pico in version 1.25 or newer,
Thonny Editor v.4 or newer,
MicroPython script presented below.



Wiring:

LTR390 UV sensor pin	Grove cable colors
VCC	red
GND	black
SCL	yellow
SDA	white

Connect Grove cable plug to PicoBox socket GP6,GP7

Simple test

```
# UV LTR390 Adafruit sensor with library modified by Pawel Boryczka
# based on library from Waveshare (Pico Enviromental Sensor).
from time import sleep
from ltr390 import LTR390
from machine import Pin, I2C
i2c=I2C(1, scl=Pin(7), sda=Pin(6), freq=100000)

sensor = LTR390(i2c)
sleep(1)
while True:
    UVS = sensor.UVS()
    #print in consola
    print(" UVS:",UVS )
    sleep(1)
```

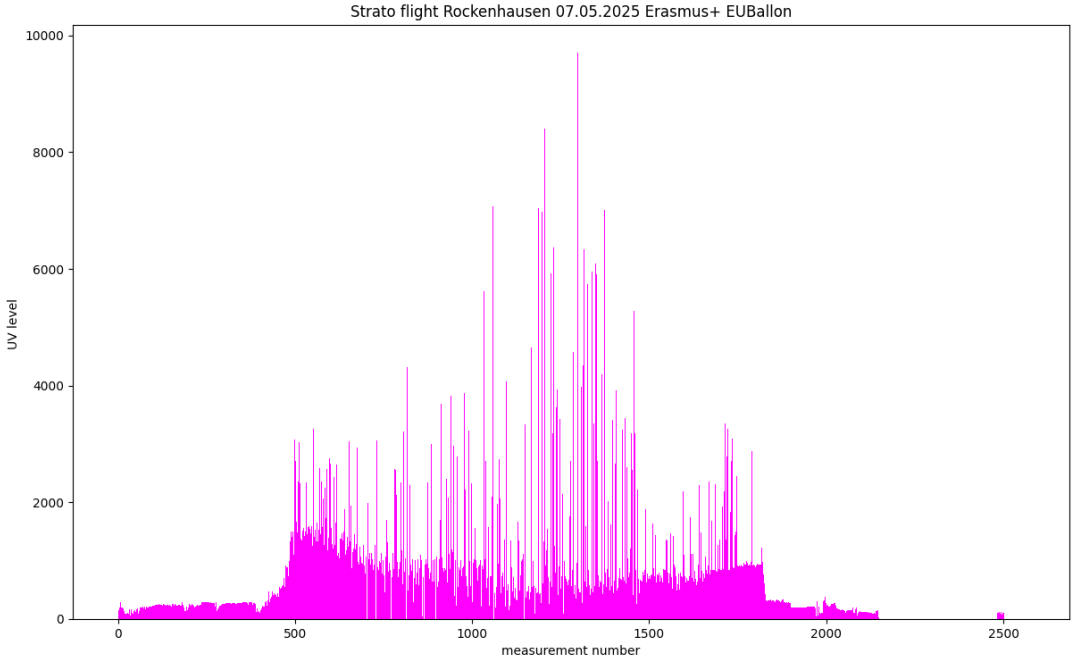
Version with OLED screen and logger

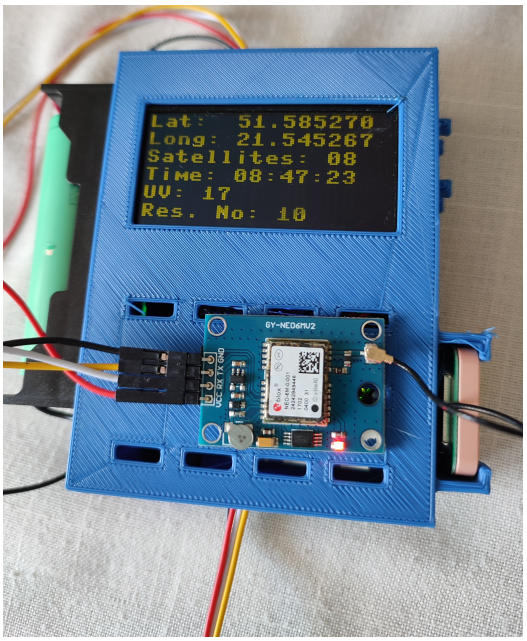
```
# UV LTR390 Adafruit sensor with library modified by Pawel Boryczka
# based on library fromWaveshare (Pico Enviromental Sensor).
from time import sleep
from ltr390 import LTR390
from machine import Pin, I2C
import ssd1306
#screen resolution
WIDTH = 128
HEIGHT = 64
#Pinout for Raspberry Pi Pico-Oled screen and LTR390
#I2C Screen and sensor connected to GP6(SDA) and GP7(SCL)
i2c=I2C(1, scl=Pin(7), sda=Pin(6), freq=400000)
oled =ssd1306.SSD1306_I2C(WIDTH, HEIGHT,i2c)
#clear screen:
oled.fill(0)
#maximum brightness contrast(255), usually more then 100
oled.contrast(250)
#variable assigned to LTR390
sensor = LTR390(i2c)
sleep(1)
file=open("UVreadings.CSV","a")
file.write("Measurement EUBallon UV LTR390 sensor \n")
file.write("No"+ "\t"+"result"+"\n")
file.flush()
no=1

```

infinite loop:

```
while True:
    UVS = sensor.UVS()
    #print in consola:
    print(" UVS:",UVS )
    oled.text("EU-ballon",30,0)
    oled.text("UV index:",0,16)
    oled.text(str(UVS),20,35)
    oled.text("No:"+str(no),0,50)
    oled.show()
    file.write(str(no)+ "\t"+str(UVS)+"\n")
    no=no+1
    #save to the file without closing file:
    file.flush()
    sleep(1)
    oled.fill(0)
```

Example of a graph generated based on data recorded in a file during a stratospheric balloon flight in Rockenhausen on May 7, 2025





4)Measuring the voltage from a 3V solar panel using an analog pin

Experiment:
solar panel voltage

Required equipment:
-PicoBox (Raspberry Pi Pico 2 with 2.4" SSD1306 OLED screen)
-solar panel 3V
-Grove cable with female ends

Software:
MicroPython system installed on Pico in version 1.25 or newer,
Thonny Editor v.4 or newer,
MicroPython script presented below.

Wiring:

```
while True:
    sensor_raw = analog_value.read_u16()
    print("raw:", sensor_raw)
    minivolt= sensor_raw/65535*3300
    oled.text("Analog sensor",0,0)
    oled.text("readings",0,15)
    oled.text("ADC level: "+str(sensor_raw),0,30)
    oled.text(str(round(minivolt,0))+" mV",0,45)
    oled.show()
    print("mV:", minivolt)
    sleep(2)
    oled.fill(0)
```

Solar panel 3V sensor pin	Grove cable colors
VCC	red
GND	black
not used	yellow
not used	white

Connect Grove cable plug to PicoBox socket GP27

Simple version

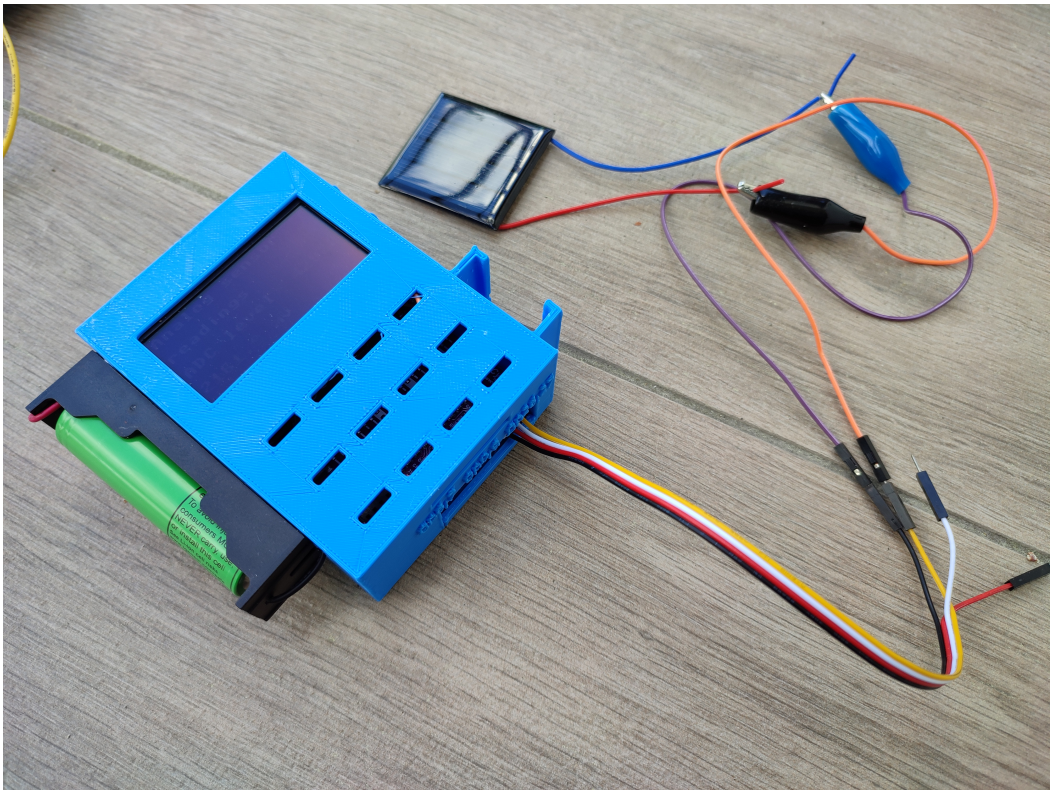
```
from time import sleep
from machine import ADC
#sensor signal pin (OUT) connected to GP27
analog_value = ADC(1)

while True:
    sensor_raw = analog_value.read_u16()
    print(sensor_raw)
    sleep(2)
    print("Voltage", sensor_raw/65535*3300, "mV")
```

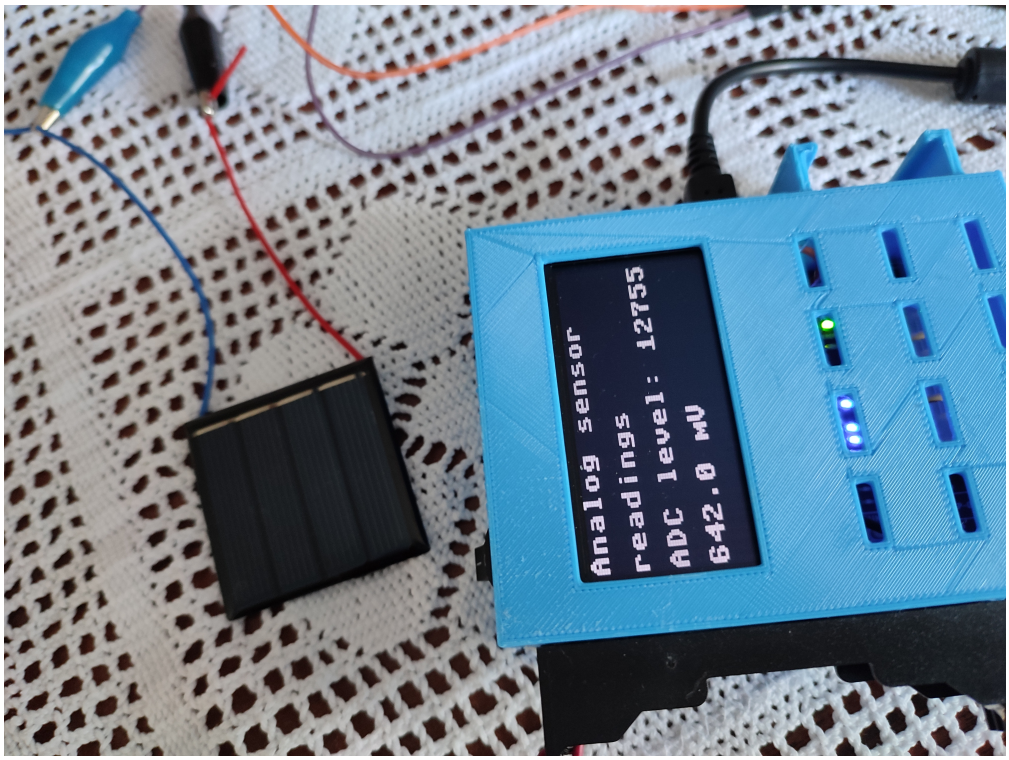
Version with OLED screen

```
from time import sleep
from machine import I2C, Pin, ADC
#oled screen library:
import ssd1306
#screen resolution
WIDTH = 128
HEIGHT = 64

#Pinout for Raspberry Pi Pico-Oled screen
#connected to pin SDA to GP6, pin SCL to GP7
#screen connected to BUS I2C

bus =I2C(1, scl=Pin(7),sda=Pin(6), freq=400000)
oled =ssd1306.SSD1306_I2C(WIDTH, HEIGHT,bus)
oled.fill(0)
#sensor signal pin (OUT) connected to GP27 analog Pin
analog_value = ADC(1)
oled.contrast(250)
```

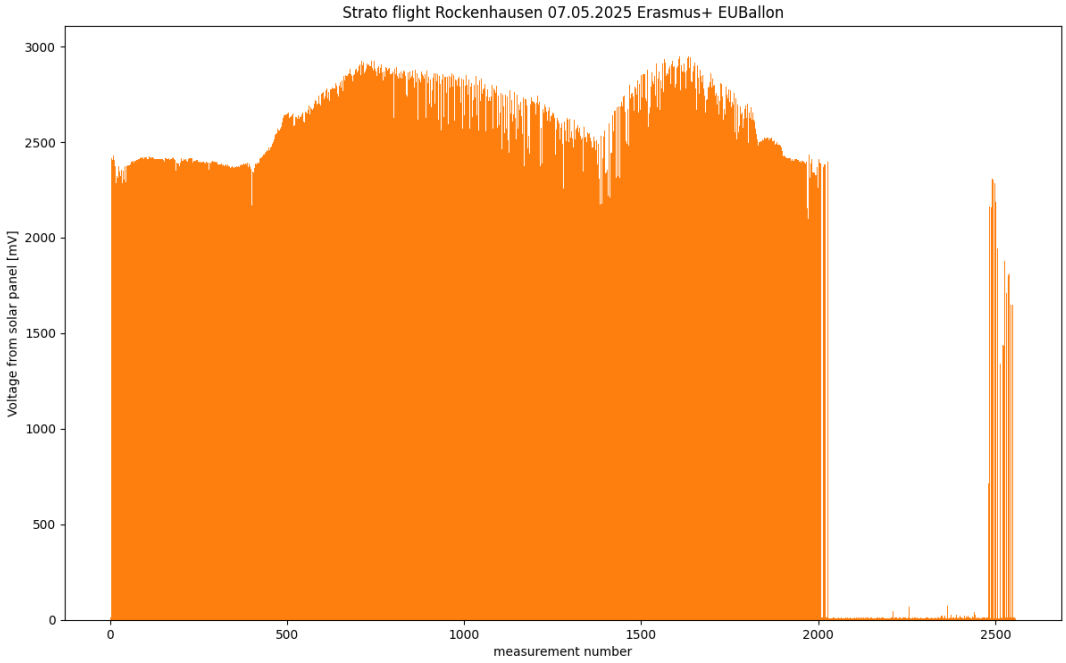
Data logger

```
from machine import  ADC
from time import sleep
analog_value=ADC(1)
file=open("readings.CSV","a")
file.write("Measurement EU-Ballon Analog sensor \n")
file.write("No"+ "\t"+"result"+"\n")
file.flush()
no=1
while True:
    result=analog_value.read_u16()
    file.write(str(no)+ "\t"+str(result)+"\n")
    #print in consola
    print(str(nr)+ "\t"+str(result))
    no=no+1
    #save to the file without closing file:
    file.flush()
    #measurement every 5 seconds:
    sleep(5)
```





Example of a chart generated based on data recorded in a file during a stratospheric balloon flight in Rockenhausen on May 7, 2025



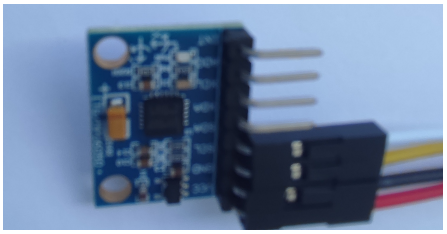
5)Accelerometer

MPU-6050
simple version
This example use library

https://github.com/harishragul/mpu6050-raspberry-pi-pico-library
sample.py is a sample code to find the raw values
Author of the used library: Harish Ragul

Required equipment:

-PicoBox (Raspberry Pi Pico 2 with 2.4" SSD1306 OLED screen)
-MPU-6050 accelerometer with thermometer (I2C 3.3V sensor)
-Grove cable with female ends



Software:

Micropython system installed on Pico in version 1.25 or newer,
Thonny Editor v.4 or newer,
MicroPython script presented below.

Wiring:

MPU-6050 sensor pin	Grove cable colors
VCC	red
GND	black
SCL	yellow
SDA	white

Connect Grove cable plug to PicoBox socket GP6,GP7

Simple version

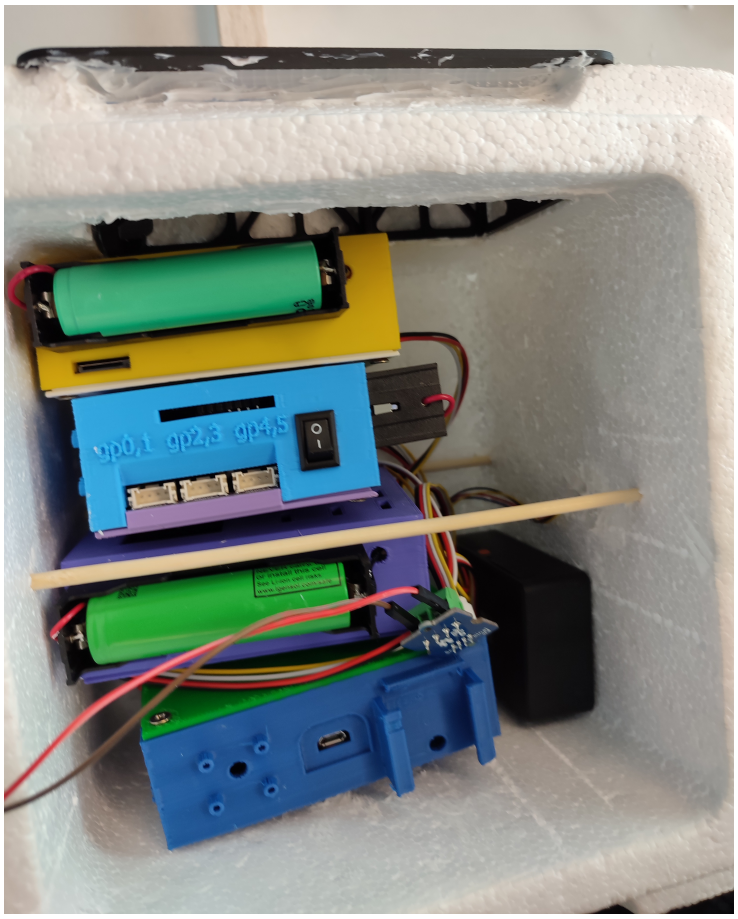
```
#vector3d.py and imu.py inside lib folder
from imu import MPU6050
from time import sleep
from machine import Pin, I2C
import ssd1306
#MPU6050 connected to GP6, GP7
i2c = I2C(1, sda=Pin(6), scl=Pin(7), freq=400000)
imu = MPU6050(i2c)
#oled screen resolution
WIDTH = 128
HEIGHT = 64
oled=ssd1306.SSD1306_I2C(WIDTH, HEIGHT,i2c)
oled.fill(0)
oled.text("EUBallon ", 0, 0)
oled.text("Acceleration", 0, 15)
oled.text("Temperature ", 0, 30)
oled.text("Measurement ", 0, 40)
oled.text("Please wait.. ", 0, 50)
oled.show()
file=open("Accellog.CSV","a")
file.write("Measurement EU-Ballon MPU6050 sensor \n")
file.write("No"+ "\t"+"ax"+"\t"+"ay"+"\t"+"az"+"\t"+"temp C"+"\n")
file.flush()
no=1
sleep(2)
```

Version with OLED screen and logger

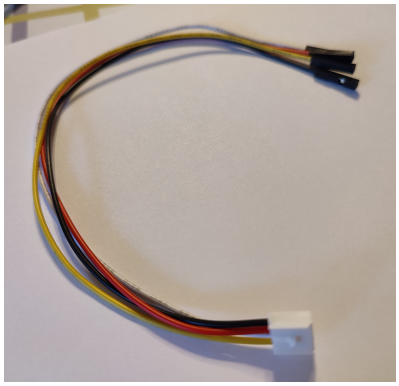
```
"""https://github.com/harishragul/mpu6050-raspberry-pi-pico-library
#sample.py is a sample code to find the raw values
# Author Harish Ragul"""
#vector3d.py and imu.py inside lib folder
from imu import MPU6050
from time import sleep
from machine import Pin, I2C
#MPU6050 connected to GP6, GP7
i2c = I2C(1, sda=Pin(6), scl=Pin(7), freq=400000)
imu = MPU6050(i2c)

while True:
    #Read Accelerometer raw value
    accX = round(imu.accel.x,2)
    accY = round(imu.accel.y,2)
    accZ = round(imu.accel.z,2)
    temp=round(imu.temperature,1)
    print(accX,"\t",accY,"\t",accZ,"\t", temp)
    sleep(0.2)
```

```
while True:
    #Read Accelerometer raw value
    accX = round(imu.accel.x,2)
    accY = round(imu.accel.y,2)
    accZ = round(imu.accel.z,2)
    temp=round(imu.temperature,1)
    print(accX,"\t",accY,"\t",accZ,"\t", temp)
    #long line splitted with \ character at the end
    file.write(str(no)+ "\t"+str(accX)+"\t"\
    +str(accY)+"\t"+str(accZ)+"\t"+str(temp)+"\n")
    #save to the file without closing file:
    file.flush()
    #time between measurements (2 seconds)
    oled.fill(0)
    oled.text("EU-Ballon logger", 0, 0)
    oled.text("AX:"+str(accX), 0, 10)
    oled.text("AY:"+str(accY), 0, 20)
    oled.text("AZ:"+str(accZ), 0, 30)
    oled.text("Temp:"+str(temp)+"C", 0, 40)
    oled.text("No:"+str(no), 0, 50)
    oled.show()
    no=no+1
    sleep(0.5)
```

Our equipment inside thermal box- preparation to ballon flight (07.05.2025)



Grove to female cable-for connection between sensors and Grove socket in PicoBox:



From the website https://euballon.zslp.edu.pl/downloads/ You can download our MicroPython scripts for Raspberry Pi Pico, libraries, our Python tools for data analysis and other materials.

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Culture Executive Agency (EACEA). Neither the European Union nor EACEA can be held responsible for them.