



Co-funded by the
Erasmus+ Programme
of the European Union



BIRGIT – training on Building InfoRmation
models integrated with Geographical
InformaTion

With the support of the Erasmus+ Program of the European Union Strategic Partnerships N° 2021-1-SE01-KA220-VET-000028000

Sensor Alarms

Assignment with solution

Author(s)/Organisation(s):

Anders Östman (Novogit AB)

License



<https://creativecommons.org/licenses/by-sa/4.0/>

Version

Version 1.0

Date: 2024-04-08

Learning outcomes

At the end of this assignment, the learner is expected to be able to

- Access sensor measurements in a sensor network.
- Develop a small Python script for handling sensor measurements.



Co-funded by the
Erasmus+ Programme
of the European Union



BIRGIT – training on Building InfoRmation
models integrated with Geographical
InformaTion

With the support of the Erasmus+ Program of the European Union Strategic Partnerships N° 2021-1-SE01-KA220-VET-000028000

Expected competences when entering the lecture

- Basic knowledge in Python.
- Basic knowledge of sensor networks.
- Basic knowledge of JSON format

Summary

This assignment is about reading sensor data from an air quality sensor network, The sensor network can be accessed using the OGC Sensor Observation Standard (SOS). If the air pollution exceeds a certain threshold, an alarm is to be issued.

Expected workload

One assignment in Python. 8 classroom hours and homework, 0.8 ECTS (ECVET)

The European Commission support for the production of this publication does not constitute endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



Revision History:

Revision	Date	Author(s)	Status	Description
0.1	2023-10-20	A. Östman	Final Draft	Assignment ready for review
0.2	2024-02-16	A Östman	Final Draft	Update based on reviewers' comments
1.0	2024-04-08	A Östman	First reviewed version	Update based on reviewers' comments



Co-funded by the
Erasmus+ Programme
of the European Union



Contents

Assignment task	5
Preparation	6
Normative solution	7
Links	8



Assignment task

Data on air quality are continuously monitored through a network of monitoring stations. In the Swedish context, the Swedish Environmental Protection Agency is responsible for reporting air quality data to the European Environmental Agency. The operations of the Swedish monitoring stations are subcontracted to different data hosts. All air quality observations are managed by the Swedish Meteorological and Hydrological Institute (SMHI).

Air quality data are entered hourly or daily and made available through various web services and API's. This assignment deals particular with the measurements of PM2.5 (aerosols), which are small particles having a maximum diameter of 2.5 micrometres. According to the Swedish norm for clean air, the concentration of PM2.5 cannot exceed 25 microgram per cubic meter of air, calculated as the mean value per day. However, the World Health Organisation (WHO) recommendations, which soon also will be the European norm, that the daily average should not exceed 15 microgram per cubic meter of air.

The stations measuring the PM2.5 content report this on an hourly basis. Your task is to develop a piece of software, which read the PM2.5 content at a monitoring station and issues an alert if the PM2.5 content the last recorded hour exceeds 15 micrograms per cubic meter air. Please note that the alert should be issued at an hourly basis, although the air quality norm refers to daily average.



Preparation

Prepare the software development environment. In case you wish to develop your software in Python, outside the QGIS environment, you need to download IDLE.

1. Download and install Python IDLE development platform from <https://www.python.org/downloads/>.
2. Prepare your python environment by installing classes required to access web resources. Open the command console and enter the following command
`$ python3 -m pip install requests`

Find the ID of the monitoring station you wish to access

3. Open QGIS and open a background map. For Swedish users, the topowebb map from Lantmäteriet may be a suitable WMS to use (<https://www.lantmateriet.se/sv/geodata/vara-produkter/produktlista/topografisk-webbkarta-visning/>). You need to apply for a token in order to have access to this map. Another option is to use Open Street Map and QuickOSM plugin as a background map.
4. The positions of all monitoring stations are provided by the Swedish EPA through a WFS. Connect to this service using the URL <https://datavardluft.smhi.se/shair/wfs>.
5. To find out the ID of our monitoring station, use the “Identify object” button and click on a suitable station. First check that the station measures PM2.5, by assuring that the attribute `sampling_point_property_notation` is set to PM2.5. Be aware that each monitoring station may have several different sensors, so you may have to look at more than one sensor. The ID to be used when querying the SOS is given by the attribute `samplingpoint_process_inspireid`. In my example, I used a monitoring station in Uppsala, having the ID “SPP-SE159404_06001_100_100”.
6. REST-API’s are based on json format while the SOS’s are originally based on XML. XML provides greater flexibility, but it is easier to work with json. Fortunately, the SOS we are using is also able to respond in json format, which simplifies our task.

Open the Python development platform and start working on the assignment.

Hint 1: One problem dealing with external data is finding out the meaning of the terms being used. In the Swedish example, some hints are given above in the preparation text. Otherwise, it may be difficult to find the meaning of the other terms being used. More general terms are specified in the OGC standards, in this case the OGC SOS Interface Standard.

Hint 2: import also the json extension. This allows you to work with json files in a convenient way.

Hint 3: Do printouts during the development, so you can see the data you are working with. Use for instance `json.dump` for formatting data as json.



Normative solution

```
# Reading sensor observations

import requests
import json

sppInspireId = "SPP-SE159404_06001_100_100"
Threshold = 15

print ("Reading SMHI sensor observations - SOS standard")
sosEndpoint = "https://datavardluft.smhi.se/52North/service?"
sosCommon = "service=SOS&version=2.0.0&"
getObservationString = "REQUEST=GetObservation&procedure=" + sppInspireId + "&responseFormat=application/json"

# Get all observations for this station
print ("Getting Observations")
outputfile = 'Observations.json'
sosUrl = sosEndpoint + sosCommon + getObservationString
response = requests.get(sosUrl)
json_formatted_str = json.dumps(response.json(), indent=2)
print(json_formatted_str,file=open(outputfile, 'w'))

# Find last observation and print date/time and value
print ("Extracting most recent observation")
Obs = response.json()["observations"] # Strip off the header information
lastObs = Obs[0]["phenomenonTime"][1] # Last observation is the first one

for sensorObs in Obs : # Check if later observations
    thisObsTime = sensorObs["phenomenonTime"][1]
    if thisObsTime > lastObs :
        lastObs = thisObsTime
        lastValue = sensorObs["result"]["value"]

print ("\n\nMost recent observation")
print ("Time: ", lastObs, "\nValue: ", lastValue, "\n\n")

if lastValue > Threshold :
    print ("\nWARNING: Threshold exceeded")
else :
    print ("\nAir Quality OK")

print ("Reading sensor observations finished")
```



Co-funded by the
Erasmus+ Programme
of the European Union



Links

General description of air quality data and reporting

<https://www.naturvardsverket.se/amnesomraden/luft/statistik--utslapp-och-halter/luftkvaliteten-i-realtid-och-preliminar-statistik/>

Description of sensor data and access.

<https://www.naturvardsverket.se/amnesomraden/luft/statistik--utslapp-och-halter/luftkvaliteten-i-realtid-och-preliminar-statistik/webbtjanster-luftkvalitetsdata/>

OGC, Sensor Observation Service Standard. <https://www.ogc.org/standard/sos/>

Link to GeoCOVID Watch Sensor Things API: <http://covidsta.hft-stuttgart.de/server/v1.1>

Basics of REST API programming. <https://realpython.com/api-integration-in-python/>

<https://dd.eionet.europa.eu/vocabularyconcept/aq/pollutant/5/view>

<https://realpython.com/api-integration-in-python/>