

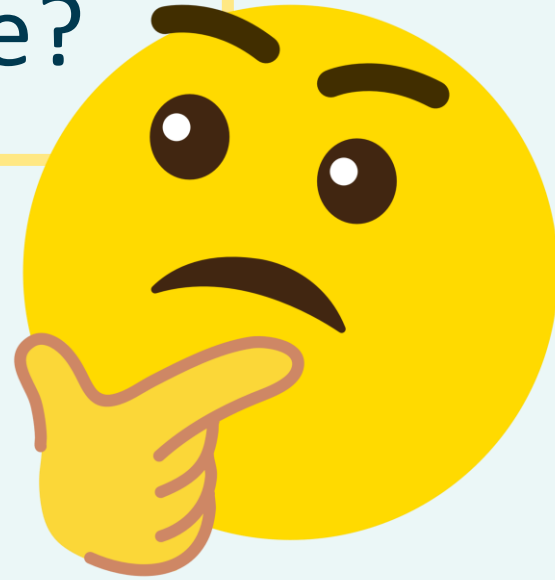
02 – JavaScript-based web dashboards for STA

Luca Giovannini



This project has received funding from the European Union's HE research and innovation programme under the grant agreement No. 101057497

Where were we?



OGC SensorThingsAPI

Advantages of OGC SensorThings API

Most IoT devices today have **proprietary software interfaces** defined by manufacturers and used selectively.



New APIs are therefore requested and developed as needed, often in an environment with **limited resources** and associated **risks**.

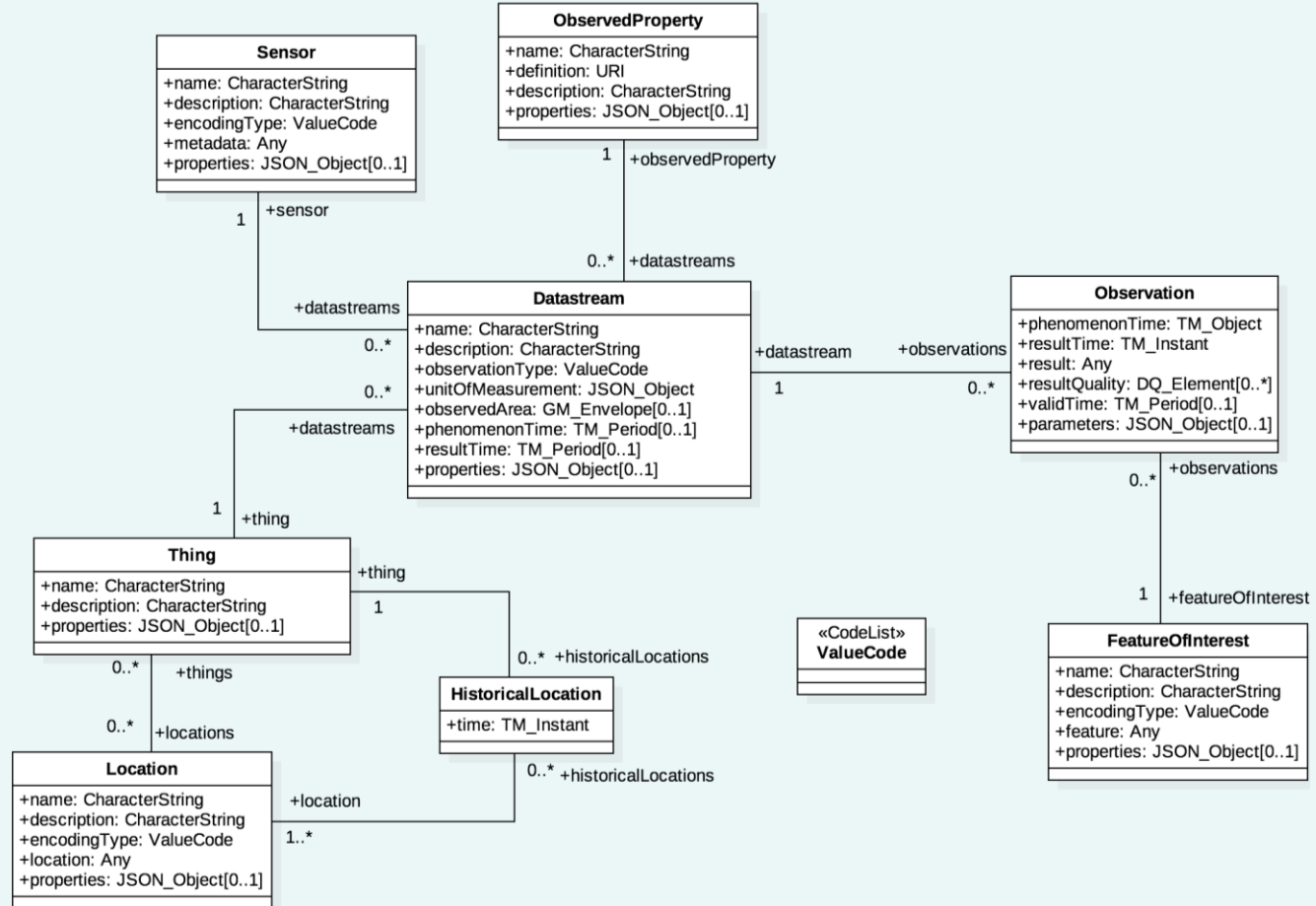


Relevant investments for each new sensor or project involving multiple systems.

As a standardized data model and interface for IoT sensors, the OGC SensorThings API offers the following benefits:

1. **enables** the proliferation of **new high-value services** with lower development overhead and broader reach
2. **reduces risk, time and cost** in a complete IoT product cycle
3. **simplifies connections** between device-devices and device-applications.

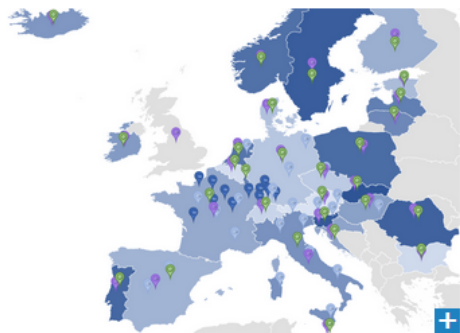
STA Data Model



STA Data Access

- **\$top**: specify the maximum number of objects to be returned. The usual default setting for \$top is 100.
- **\$skip**: used for paging, skip over the first n records and provide records from the n + 1 on.
- **\$count**: return the total number of objects in the response. The usual default setting for \$count is false.
- **\$orderBy**: used to specify that the returned objects should be ordered by a specific attribute, either ascending or descending.
- **\$select**: specify exactly which attributes are to be provided in the response.
- **\$filter**: specify filters that control which entities are returned. See Filtering
- **\$expand**: create a response returning multiple object types nested within each other.

FROST[®]-Server - An open source implementation of OGC SensorThings API




Locations of the data sources connected in API4INSPIRE

Description of the project

Against the background of the requirements from numerous applications in the mega trend topic "Internet of Things", the Fraunhofer IOSB decided to develop a server for the Standard SensorThingsAPI. The objective was to achieve high performance with low resource consumption and an openness that facilitates usability both in the research environment and in commercial applications.

Performance with low resource consumption and an openness that facilitates usability in both the research environment and commercial applications. The desire for openness led to the decision to design the implementation as open source software right from the start, in order to make the way as free as possible for innovation. The result is the FROST[®]-Server. The name is an acronym and stands for "Fraunhofer Open Source SensorThings API Server". But the name is also intended to suggest that your data is kept "fresh and available"

Meanwhile, the "SensorThings API" is recommended by the European Commission as "Good Practice" for the deployment of measurement data according to the INSPIRE guidelines: ["OGC SensorThings API as an INSPIRE download service"](#) 

<https://www.iosb.fraunhofer.de/en/projects-and-products/frost-server.html>

FROST Server deployment

The FROST-Server comes in three packages, and thus also in three docker images:

- [fraunhoferiosb/frost-server](https://github.com/fraunhoferiosb/frost-server) The all-in-one package
- [fraunhoferiosb/frost-server-http](https://github.com/fraunhoferiosb/frost-server-http) The HTTP-only package
- [fraunhoferiosb/frost-server-mqtt](https://github.com/fraunhoferiosb/frost-server-mqtt) The MQTT-only package

The FROST-Server can be run in **Tomcat** or as a [docker image](#)
You will also need a **PostgreSQL** server with **PostGIS** extensions

<https://fraunhoferiosb.github.io/FROST-Server/deployment/architecture-packages.html>



STA Endpoint

SensorThings API

Thanks to the way SensorThings API was built...



Web browser navigation

...data from the API can be easily viewed using a **normal Web Browser**. One can simply navigate from one object to the next by clicking the **URLs** provided within the data.

How?

ENDPOINT

<https://airquality-frost.k8s.ilt-dmz.iosb.fraunhofer.de/v1.1/Observations>

STA Endpoint: examples

- 1 BRGM Water**
Water data by OGC
<https://sensorthings-wq.brgm-rec.fr/FROST-Server/v1.0>
- 2 Fraunhofer Stats**
Demographic statistics by Fraunhofer Institute, Germany
<https://demography.k8s.ilt-dmz.iosb.fraunhofer.de/v1.1>
Covid data by Fraunhofer Institute, Germany
<http://covidsta.hft-stuttgart.de/server/v1.1/>
- 3 Municipality of Ferrara**
Data about air quality, bike transits, traffic by Municipality of Ferrara, Italy
<https://iot.comune.fe.it/FROST-Server/v1.1/>
- 4 Fraunhofer Air Quality**
Data about air quality from AQ stations in Europe, by Fraunhofer Institute, Germany (EEA)
<https://airquality-frost.k8s.ilt-dmz.iosb.fraunhofer.de/v1.1/>

Now...how to go from
a STA Endpoint
to a web dashboard?

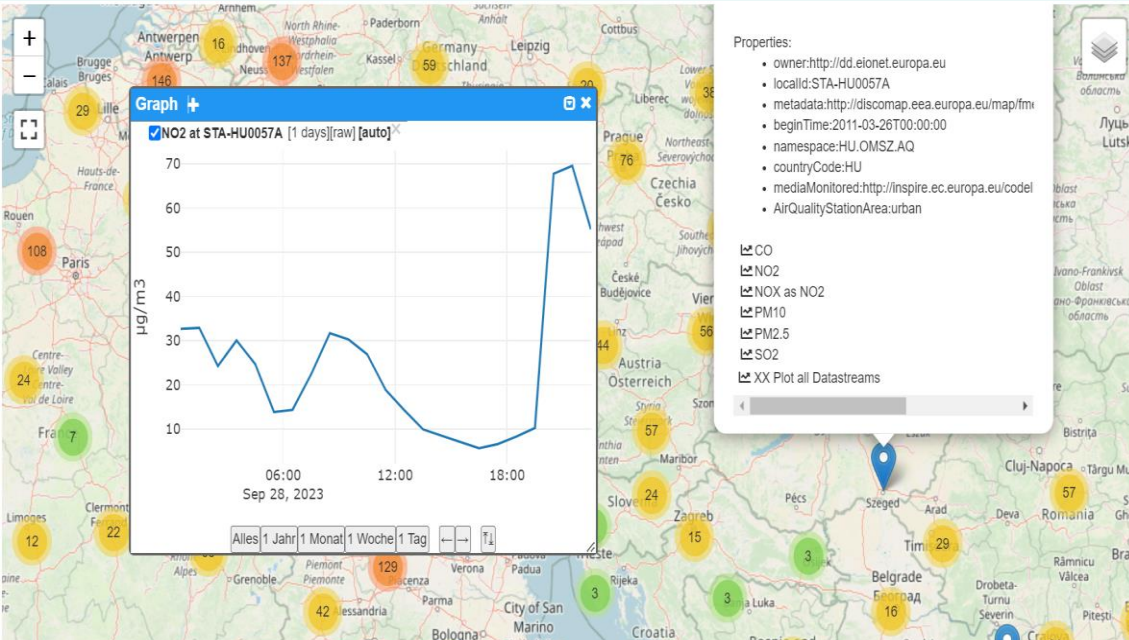


Solution 1: STAM (by DataCove)



STAM (SensorThings API Map) is a JavaScript library for showing the Things/Features of interest of a SensorThings server on a Leaflet/OpenLayers map.

Based for plots on open-source library <https://plotly.com/>



Example 1: without plot



Example 2: simple plot



Example 3: complex plot



Solution 2: SensorUp SDK



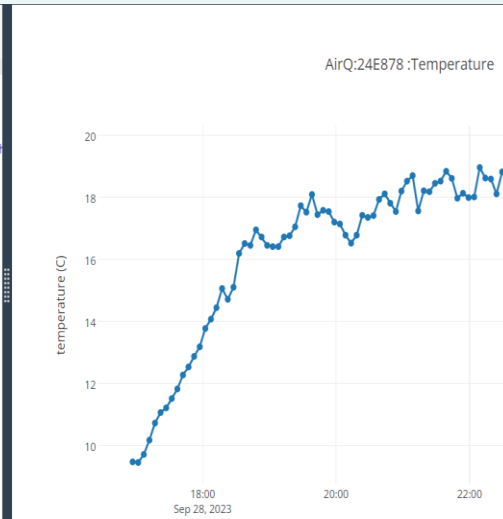
Several reusable JS-based examples for maps, charts and gauges.

Based on Leaflet and Highcharts <https://www.highcharts.com/>

```

Templates ▾ Tools ▾ Share ▾ Run
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset='utf-8' />
5   <meta name='viewport' content='initial-scale=1,maximum-scale=1,user-scalable=no' />
6   <link href='https://fonts.googleapis.com/css?family=Open+Sans:400,300,700' rel='stylesheet' />
7   <script src='https://cdn.plot.ly/plotly-1.22.0.min.js'></script>
8   <script src='https://sdk.sensorup.com/sta-chart/sta-chart-0.0.13.min.js'></script>
9 </head>
10 <body>
11   font-family: 'Open Sans', sans-serif, Arial;
12   font-size: 12px;
13 </body>
14 </style>
15 </head>
16 <body>
17 <div id='vis'></div>
18 <script type='text/javascript'>
19   stachart.generateChart({
20     'targetId': 'vis',
21     'staBaseUrl': 'https://calgary-aq-sta.sensorup.com/v1.0',
22     'datastreamId': '15708486'
23   });
24 </script>
25 </body>
26 </html>
27

```



Example 1: maps



Example 2: charts



Example 3: sandbox



Solution 3: Do-It-Yourself...

Open-source libraries for **mapping**:

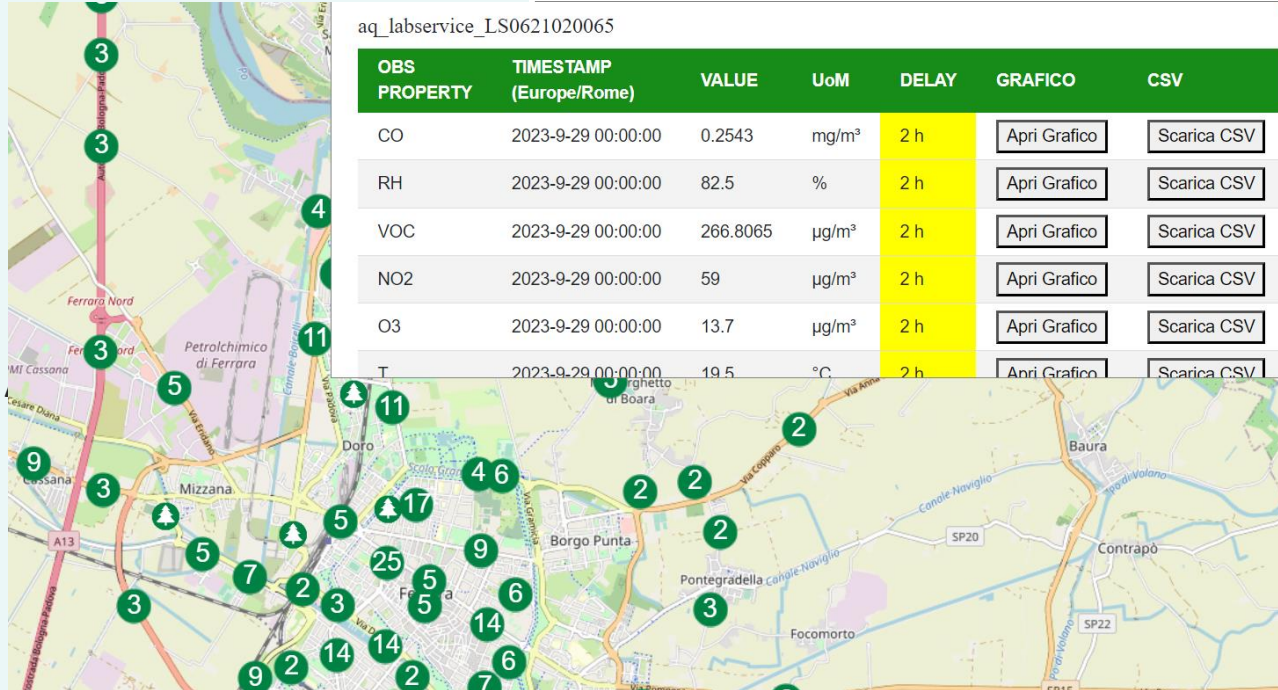
<https://leafletjs.com/>

<https://openlayers.org/>

Open-source libraries for **plots**:

<https://plotly.com/>

<https://www.chartjs.org/>



AirBreak STA Explore

