openSenseMap

Dokumentation

Inhaltsverzeichnis

openSenseMap

Die openSenseMap (OSeM) ist eine Webplattform, auf welcher diverse standortbezogene Sensordaten hochgeladen und visualisiert werden können. Auf der Plattform lassen sich Stationen registrieren, welche die Daten eines oder mehrerer Sensoren übertragen.

Neben einer Zeitreihenvisualisierung der Daten, ist es auch möglich diese nach verschiedenen Kriterien zu Filtern und räumliche Interpolationen zu errechnen.

Sämtliche Sensordaten stehen unter der Public Domain Dedication and License 1.0 zum Download zur Verfügung, und können frei verwendet werden.

Sowohl die openSenseMap als auch die zugehörige API ist Open Source Software. Quellcode und Issuetracker sind hier zu finden:

- openSenseMap
- openSenseMap API

Registrierung auf der OSeM

Bearbeiten einer Station

Von bereits registrierten Stationen lassen sich sämtliche Angaben nachträglich ändern.

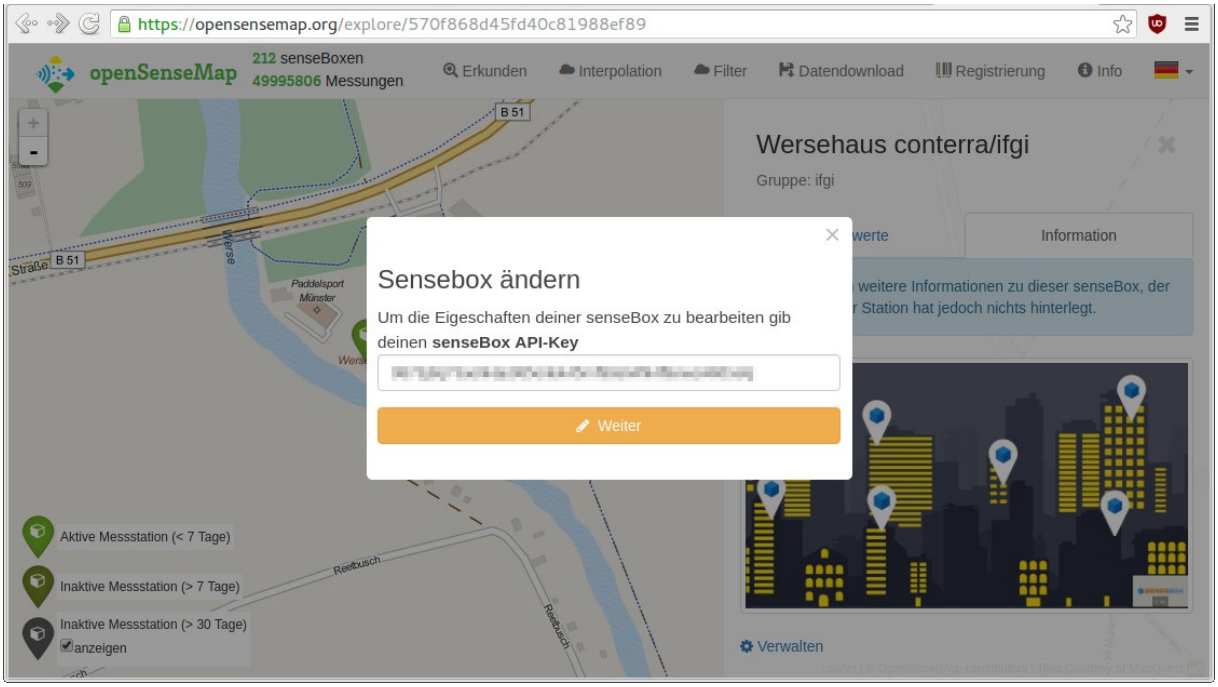
Hierzu wird der bei der Registrierung erhaltene API-Key benötigt!

1. Wähle deine Station auf der openSenseMap durch Klick auf den entsprechenden Marker auf der Karte aus.

   Alternativ kannst du auch in der folgenden URL `<senseBox-ID>` durch deine senseBox-ID (nicht der API-Key!) ersetzen: `https://opensensemap.org/explore/<senseBox-ID>`

2. Wähle in der rechten Sidebar den Tab "Information", und klicke unten den Button "Verwalten".

3. Gebe deinen API-Key im erscheinenden Dialog ein.



4. Gebe deine Änderungen im sich öffnenden Formular ein. Neben Änderungen an Metadaten, Standort und Foto ist es auch möglich die Sensor-Konfiguration zu ändern.

   > Hinweis: Falls du einen neuen Sensor hinzugefügt hast und den aktualisierten Arduino-Sketch herunterladen willst, ist dies erst möglich, wenn die Seite nach dem Speichern neu geladen, und der API-Key erneut eingegeben wurde.

5. Klicke im oberen Teil des Dialogs auf "Speichern" um deine Änderungen zu speichern, oder "Abbrechen" um sie zu verwerfen.

Löschen einer Station

Folge den Schritten unter "Bearbeiten einer Station" und gebe unter "Sensebox löschen" `DELETE` in das Textfeld ein.

> Achtung: Hierdurch werden neben deiner senseBox alle hinterlegten Sensordaten unwiderruflich entfernt!

Datendownload

Unter dem Reiter "Datendownload" sind Funktionen zum Herunterladen der Sensordaten zu finden.

Der Datendownload bezieht sich immer auf ein ausgewähltes Phänomen (z.B. Lufttemperatur), einen Zeitraum und eine Boundingbox. Die Boundingbox bezeichnet die räumliche Auswahl der Stationen, und wird automatisch durch den aktuell sichtbaren Kartenausschnitt bestimmt.

> Achtung: Je nach Auswahl der Filterparameter kann der Download sehr groß werden (mehrere 100mb)!

Erweitertes Filtern

Zusätzlich zu den zeitlichen und räumlichen Filtern unter "Datendownload" lässt sich die Stationsauswahl weiter unter dem Reiter "Filter" einschränken. Wie das geht ist im Kapitel Datenanalyse beschrieben.

Formate

Derzeit wird nur das Datenformat CSV unterstützt, welches problemlos mit Tabellenkalkulations-Tools wie Excel verarbeitet werden kann.

Jede Zeile enthält eine Messung einer senseBox mit dem ausgewählten Phänomen. Der Messwert (`value`), Standort des Sensors (`lat` , `lng` , Referenzsystem WGS84) und ein Zeitstempel (`createdAt`) sind in je einer Spalte angegeben:

```
createdAt;value;lat;lng
2016-09-20T10:05:49.581Z;18.70;7.64568;51.962372
2016-09-20T10:00:52.689Z;18.62;7.64568;51.962372
2016-09-20T09:55:54.282Z;18.47;7.64568;51.962372
....
```

Datenanalyse

Filter

Die angezeigten senseBoxen lassen sich nach verschiedenen Kriterien auswählen.

Hierzu können unter dem Reiter "Filter" die entsprechenden Angaben in der Sidebar gemacht werden. Nach einem Klick auf "Filter anwenden" werden die Boxen gefiltert (dies kann je nach Auswahl einen Augenblick dauern).

Anschließend wird unterhalb der Filtereinstellungen eine Auflistung der auf die Kriterien zutreffenden senseBoxen angezeigt.

Dieser Filter bezieht sich auch auf die anderen Datenanalyse-Funktionen Interpolation und Datendownload!

Interpolation

Im Reiter "Interpolation" lassen sich die Daten mehrerer senseBoxen räumlich interpolieren.

Dies ist nützlich um die räumlichen Unterschiede eines Phänomens auf der Karte sichtbar zu machen, oder um ungefähre (!) Werte in Regionen ableiten zu können, in welchen keine Sensoren vorhanden sind.

Vorraussetzung für diese Funktion ist, dass zuvor ein Filter auf einen Zeitraum und ein bestimmtes Mess-Phänomen gewählt wurde.

Es stehen zwei Interpolationsverfahren zur Verfügung: Inverse Distance Weighting (IDW) und Thiessen Polygone.

Nach Einstellung der Parameter (s.U.) wird die Interpolation auf unserem Server berechnet. Wenn die Kalkulation abgeschlossen ist, wird das Ergebnis als Heatmap in der Karte angezeigt.

Inverse Distance Weighting

`TODO`

Thiessen Polygone

`TODO`

The openSenseMap provides a REST API, which can be used to query & post senseBox metadata & measurements. The endpoint is `https://api.opensensemap.org/` .

This documentation can also be found here with an improved layout.

---

openSenseMap API documentation

methods to manage senseBoxes and get/post measurements

Boxes

Validate authorization method GET

> Validate authorization through API key and senseBoxId. Will return status code 403 if invalid, 200 if valid.

```
GET /users/:senseBoxId
```

Headers

Name	Type	Description
x-APIKey	String	the secret API key which corresponds to the `senseBoxId` parameter.

Parameter

Name	Type	Description
:senseBoxId	String	the ID of the senseBox you are referring to.

Success 200

Name	Type	Description
Response	String	ApiKey is valid

Error Response

Error-Response:

```
HTTP/1.1 403 Forbidden
{"code":"NotAuthorized","message":"ApiKey is invalid or missing"}
```

Get one senseBox method GET

```
GET /boxes/:boxId
```

Parameter

Name	Type	Description
format	String	the format the sensor data is returned in. Default value: json Allowed values: "json","geojson"
:senseBoxId	String	the ID of the senseBox you are referring to.

Success Response

Example data on success:

```
{
  "_id": "5386e44d5f08822009b8b614",
  "name": "PHOBOS",
  "boxType": "fixed",
  "sensors": [
    {
      "_id": "5386e44d5f08822009b8b615",
      "boxes_id": "5386e44d5f08822009b8b614",
      "lastMeasurement": {
        "_id": "5388d07f5f08822009b937b7",
        "createdAt": "2014-05-30T18:39:59.353Z",
        "updatedAt": "2014-05-30T18:39:59.353Z",
        "value": "584",
        "sensor_id": "5386e44d5f08822009b8b615",
      },
      "sensorType": "GL5528",
      "title": "Helligkeit",
      "unit": "Pegel"
    }
  ],
  "loc": [
    {
      "_id": "5386e44d5f08822009b8b61a",
      "geometry": {
        "coordinates": [
          10.54555893642828,
          49.61361673283691
        ],
        "type": "Point"
      },
      "type": "feature"
    }
  ]
}
```


Get all senseBoxes method GET

> With the optional `date` and `phenomenon` parameters you can find senseBoxes that have submitted data around that time, +/- 2 hours, or specify two dates separated by a comma.

```
GET /boxes?date=:date&phenomenon=:phenomenon&format=:format
```

Parameter

Name	Type	Description
date	String	A date or datetime (UTC) where a station should provide measurements. Use in combination with `phenomenon`.
phenomenon	String	A sensor phenomenon (determined by sensor name) such as temperature, humidity or UV intensity. Use in combination with `date`.
format	String	the format the sensor data is returned in. Default value: json Allowed values: "json","geojson"

Post new senseBox method POST

> Create a new senseBox. This method allows you to submit a new senseBox.
>
> Along with the senseBox, an user is created which then owns the senseBox.
>
> If you specify `mqtt` parameters, the openSenseMap API will try to connect to the MQTT broker specified by you. The parameter `messageFormat` tells the API in which format you are sending measurements in.
>
> For `json`, the format is:
>
> ```
> {   "sensorId": <value>,   "sensorId": [<value>,<createdAt>]   ... }
> ```
>
> For `csv`, the format is:
>
> ```
> sensorId,value sensorId,value,createdAt ...
> ```

```
POST /boxes
```

JSON request body

Name	Type	Description
firstname	String	firstname of the user for the senseBox.
lastname	String	lastname of the user for the senseBox.
email	String	email of the user for the senseBox.
orderID	String	the apiKey of the user for the senseBox.
name	String	the name of this senseBox.
grouptag	String	the grouptag of this senseBox.
exposure	String	the exposure of this senseBox. Allowed values: "indoor","outdoor"
boxType	String	the type of the senseBox. Currently only 'fixed' is supported. Allowed values: "fixed"
sensors	Sensor[]	an array containing the sensors of this senseBox.
loc	Location	the location of this senseBox. Must be a GeoJSON Point Feature. (RFC7946)

A single sensor for the nested Sensor parameter

Name	Type	Description
title	String	the title of the phenomenon the sensor observes.
unit	String	the unit of the phenomenon the sensor observes.
sensorType	String	the type of the sensor.
icon	String	the visual representation for the openSenseMap of this sensor.

Settings for a senseBox connected through MQTT

Name	Type	Description
url	String	the url to the mqtt server.
topic	String	the topic to subscribe to.
messageFormat	String	the format the mqtt messages are in. Allowed values: "json","csv"
decodeOptions	String	a json encoded string with options for decoding the message. 'jsonPath' for 'json' messageFormat.
connectionOptions	String	a json encoded string with options to supply to the mqtt client (https://github.com/mqttjs/MQTT.js#client)

Update a senseBox: Image and sensor names method PUT

> Modify the specified senseBox.

```
PUT /boxes/:senseBoxId
```

Headers

Name	Type	Description
x-APIKey	String	the secret API key which corresponds to the `senseBoxId` parameter.

JSON request body

Name	Type	Description
description	String	the updated description of this senseBox.
image	String	the updated image of this senseBox encoded as base64 data uri.
name	String	the name of this senseBox.
grouptag	String	the grouptag of this senseBox.
exposure	String	the exposure of this senseBox. Allowed values: "indoor","outdoor"
boxType	String	the type of the senseBox. Currently only 'fixed' is supported. Allowed values: "fixed"
sensors	Sensor[]	an array containing the sensors of this senseBox.
loc	Location	the location of this senseBox. Must be a GeoJSON Point Feature. (RFC7946)

A single sensor for the nested Sensor parameter

Name	Type	Description
title	String	the title of the phenomenon the sensor observes.
unit	String	the unit of the phenomenon the sensor observes.
sensorType	String	the type of the sensor.
icon	String	the visual representation for the openSenseMap of this sensor.

Settings for a senseBox connected through MQTT

Name	Type	Description
url	String	the url to the mqtt server.
topic	String	the topic to subscribe to.
messageFormat	String	the format the mqtt messages are in. Allowed values: "json","csv"
decodeOptions	String	a json encoded string with options for decoding the message. 'jsonPath' for 'json' messageFormat.
connectionOptions	String	a json encoded string with options to supply to the mqtt client (https://github.com/mqttjs/MQTT.js#client)

Parameter

Name	Type	Description
:senseBoxId	String	the ID of the senseBox you are referring to.

Error Response

Error-Response:

```
HTTP/1.1 403 Forbidden
{"code":"NotAuthorized","message":"ApiKey is invalid or missing"}
```

Delete a senseBox and its measurements method DELETE

```
DELETE /boxes/:senseBoxId
```

Headers

Name	Type	Description
x-APIKey	String	the secret API key which corresponds to the `senseBoxId` parameter.

Parameter

Name	Type	Description
:senseBoxId	String	the ID of the senseBox you are referring to.

Error Response

Error-Response:

```
HTTP/1.1 403 Forbidden
{"code":"NotAuthorized","message":"ApiKey is invalid or missing"}
```

Download the Arduino script for your senseBox method GET

```
GET /boxes/:senseBoxId/script
```

Headers

Name	Type	Description
x-APIKey	String	the secret API key which corresponds to the `senseBoxId` parameter.

Parameter

Name	Type	Description
:senseBoxId	String	the ID of the senseBox you are referring to.

Error Response

Error-Response:

```
HTTP/1.1 403 Forbidden
{"code":"NotAuthorized","message":"ApiKey is invalid or missing"}
```

Measurements

Post new measurement method POST

> Posts a new measurement to a specific sensor of a box.

```
POST /boxes/:senseBoxId/:sensorId
```

JSON request body

Name	Type	Description
value	String	the measured value of the sensor. Also accepts JSON float numbers.
createdAt	String	the timestamp of the measurement. Should be parseable by JavaScript.

Parameter

Name	Type	Description
:senseBoxId	String	the ID of the senseBox you are referring to.
:sensorId	String	the ID of the sensor you are referring to.

Get latest measurements of a senseBox method GET

> Get the latest measurements of all sensors of the specified senseBox.

```
GET /boxes/:senseBoxId/sensors
```

Parameter

Name	Type	Description
:senseBoxId	String	the ID of the senseBox you are referring to.

Get the 10000 latest measurements for a sensor method GET

> Get up to 10000 measurements from a sensor for a specific time frame, parameters `from-date` and `to-date` are optional. If not set, the last 48 hours are used. The maximum time frame is 1 month. If `download=true` `Content-disposition` headers will be set. Allows for JSON or CSV format.

```
GET /boxes/:senseBoxId/data/:sensorId?from-date=fromDate&to-datetoDate&download=true&format=json
```

Parameter

Name	Type	Description
from-date	String	Beginning date of measurement data (default: 48 hours ago from now)
to-date	String	End date of measurement data (default: now)
download	String	If set, offer download to the user (default: false, always on if CSV is used) Allowed values: "true","false"
format	String	Can be 'json' (default) or 'csv' (default: json) Default value: json Allowed values: "json","csv"
:senseBoxId	String	the ID of the senseBox you are referring to.
:sensorId	String	the ID of the sensor you are referring to.
separator	String	Only for csv: the separator for csv. Possible values: `comma` for comma as separator, everything else: semicolon. Per default a semicolon is used.

Get latest measurements for a phenomenon as CSV method GET,POST

> Download data of a given phenomenon from multiple selected senseBoxes as CSV

```
GET,POST /boxes/data?boxid=:senseBoxIds&from-date=:fromDate&to-date:toDate&phenomenon=:phenomenon
```

Parameter

Name	Type	Description
senseBoxIds	String	Comma separated list of senseBox IDs.
phenomenon	String	the name of the phenomenon you want to download the data for.
from-date	String	Beginning date of measurement data (default: 15 days ago from now)
to-date	String	End date of measurement data (default: now)
separator	String	Only for csv: the separator for csv. Possible values: `comma` for comma as separator, everything else: semicolon. Per default a semicolon is used.

Misc

Get some statistics about the database method GET

> 8 boxes, 13 measurements in the database, 2 measurements in the last minute

```
GET /stats
```

Success Response

[8,13, 2]

```
[8,13, 2]
```