

# openSenseMap

## Dokumentation



---

# Inhaltsverzeichnis

Einleitung	1.1
Registrierung	1.2
Luftdaten.info Feinstaubsensor	1.2.1
Andere Plattformen	1.2.2
Verwalten von senseBoxen	1.3
Datendownload	1.4
Datenanalyse	1.5
MQTT Client	2.1
TheThingNetwork Integration	2.2
REST API	3.1
Community Anwendungen	3.2
HTML Widget	3.3



## 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](https://creativecommons.org/licenses/by/4.0/)<sup>1</sup> 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](https://github.com/sensebox/OpenSenseMap)<sup>2</sup>
- [openSenseMap API](https://github.com/sensebox/OpenSenseMap-API)<sup>3</sup>

---

<sup>1</sup>. <https://opendatacommons.org/licenses/pddl/summary/> ↩

<sup>2</sup>. <https://github.com/sensebox/OpenSenseMap> ↩

<sup>3</sup>. <https://github.com/sensebox/OpenSenseMap-API> ↩

## Registrierung auf der openSenseMap

Um eine neue senseBox auf der openSenseMap zu registrieren, muss zunächst unter Anmelden ein Nutzer-Account erstellt werden. Anschließend ist unter dem User-Logo oben rechts das Dashboard zu finden, über welches senseBoxen hinzugefügt und [verwaltet](#)<sup>1</sup> werden können.

Unter Neue senseBox kann nun eine neue senseBox registriert werden. Die folgenden Angaben sind notwendig:

- Name der senseBox: z.B. der Standort
- Aufstellungsort: dient der Filterung von Boxen
- Standort: kann über die Karte ausgewählt werden
- Modell: bestimmt die Sensorkonfiguration

Es gibt eine Vorauswahl für verschiedene Modelle. Falls eine nicht vorhandene Sensorkonfiguration vorliegt, können einzelne Sensoren unter Manuelle Konfiguration von Hand hinzugefügt werden. Wie dies im Detail funktioniert ist unter [Andere Plattformen](#)<sup>2</sup> beschrieben.

Nachdem die Registrierung abgeschlossen wurde, wird ein Arduino-Sketch angezeigt, welcher die angegebenen Sensoren ausliest und deren Daten regelmäßig zur openSenseMap überträgt. Um diesen auf die senseBox zu übertragen, wird die [Arduino IDE](#)<sup>1</sup> benötigt, eine exemplarische Installations-Anleitung für die senseBox:home ist [hier](#)<sup>2</sup> zu finden.

## Erweiterte Konfiguration

Es besteht die Möglichkeit neben der [HTTP REST API](#)<sup>3</sup> auch andere Schnittstellen zur Datenübertragung zu nutzen. Einstellungen hierfür müssen unter dem entsprechenden Reiter im Abschnitt Erweitert vorgenommen werden. Detaillierte Anleitungen dazu sind hier zu finden:

- [MQTT](#)<sup>4</sup>
- [TheThingsNetwork](#)<sup>5</sup>

---

<sup>1</sup>. Siehe [1.3 Verwalten von senseBoxen](#) ↩

<sup>2</sup>. Siehe [1.2.2 Andere Plattformen](#) ↩

<sup>1</sup>. <https://www.arduino.cc/en/Main/Software> ↩

<sup>2</sup>. [https://home.books.sensebox.de/de/software\\_installation.html](https://home.books.sensebox.de/de/software_installation.html) ↩

<sup>3</sup>. Siehe [3.1 REST API](#) ↩

<sup>4</sup>. Siehe [2.1 MQTT Client](#) ↩

<sup>5</sup>. Siehe [2.2 TheThingNetwork Integration](#) ↩

# Stuttgart Luftdaten.info Daten an openSenseMap.org senden

Die Feinstaub Sensoren des OK Lab Stuttgart (Luftdaten.info) erlauben es, die gemessenen Daten auch an die openSenseMap zu senden. Um die Daten an die openSenseMap zu senden muss zu allererst herausgefunden werden, welche Sensoren am Feinstaubsensor verwendet werden. Dies kann man am besten in dem Webinterface des Feinstaubsensors nachsehen. (Abb. 1<sup>1</sup>)

Danach muss eine senseBox auf der openSenseMap mit der gerade nachgesehenen Konfiguration registriert werden. Sollten bei der Registrierung die falschen Sensoren ausgewählt worden sein, ist es am einfachsten die Box einfach wieder zu löschen und mit der korrekten Sensorkonfiguration neu zu registrieren.

## 1. Neue senseBox registrieren<sup>1</sup> (<https://opensensemap.org/register>)

- User, Standort, Aufstellungsort und Namen ausfüllen. Gruppenkennzeichnung könnte z.B.: Luftdaten sein.
- Unter dem Punkt "Hardware" im Schritt "meine senseBox" das Feld "luftdaten.info" ausklappen und die passende Sensorkonfiguration auswählen. (Abb. 2<sup>2</sup>)
- Registrierung abschließen.
- Wichtig: senseBox ID kopieren. Dies ist eine 24 Zeichen lange Zeichenkette die ungefähr so aussieht:  
58a88c6b650831d8a3625e01
- Wenn eine korrekte E-Mailadresse angegeben wurde, kommt die senseBox ID auch nochmal per Mail. (zum Beispiel: Deine senseBox-ID lautet: 58a88c6b650831d8a3625e01)

## 2. Feinstaub Sensor konfigurieren

Der Feinstaub Sensor von Luftdaten.info lässt sich bequem über eine Webseite konfigurieren. Hierfür muss zuerst die IP des Geräts im WLAN ausfindig gemacht werden. Dies gelingt am besten entweder durch ablesen im WLAN-Router.

- Mit dem Browser die Konfigurationsseite des Feinstaubsensors aufrufen.
- Unter dem Punkt Weitere APIs einen Haken bei An openSenseMap senden machen. In das Feld senseBox-ID die eigene senseBox-ID eintragen.
- Ganz unten auf der Seite auf Speichern klicken

## Fertig

Der Feinstaubsensor sollte nun seine Daten an die openSenseMap senden.

## Abbildung 1: Webinterface Feinstaubsensor

**Sensoren**

- ☒ SDS011 (Feinstaub)
- ☒ DHT22 (Temp., Luftfeuchte)
- ☐ PPD42NS
- ☐ BMP180
- ☐ BME280
- ☐ GPS (NEO 6M)

Die Auswahl der Sensoren an dieser Stelle ist entscheidend für die Registrierung auf <https://openSenseMap.org/>  
Sollte vor der Registrierung nachgesehen werden.

**Weitere Einstellungen**

- ☐ Auto Update
- ☐ Display

Debug Level

**Weitere APIs**


☒ An OpenSenseMap senden  
senseBox-ID:

Nach der Registrierung auf <https://openSenseMap.org/> hier die zugewiesene senseBox ID angeben

☐ An eigene API senden  
Server:   
Pfad:   
Port:   
Benutzer:   
Passwort:

☐ Senden an InfluxDB  
Server:   
Pfad:   
Port:   
Benutzer:   
Passwort:

Abbildung 2: Registrierung openSenseMap

 **openSenseMap** 391 senseBoxen  
174352668 Messungen

Suche nach Boxen und Orten

---

### Hardware

Wähle den Typ deiner senseBox aus.

senseBox:home

luftdaten.info

☐ Luftdaten.info Feinstaubsensor ohne Temperatur/Feuchtesensor

☐ Luftdaten.info Feinstaubsensor mit DHT11

☒ Luftdaten.info Feinstaubsensor mit DHT22

☐ Luftdaten.info Feinstaubsensor mit BMP180

☐ Luftdaten.info Feinstaubsensor mit BME280

Manuelle Konfiguration

### Erweitert

MQTT

Zurück

Weiter

<sup>1</sup>. Siehe [1.2.1 Luftdaten.info Feinstaubsensor](#) > [abbildung-1-webinterface-feinstaubsensor](#) ↩

<sup>1</sup>. <https://opensensemap.org/register> ↩

<sup>2</sup>. Siehe [1.2.1 Luftdaten.info Feinstaubsensor](#) > [abbildung-2-registrierung-opensensemap](#) ↩

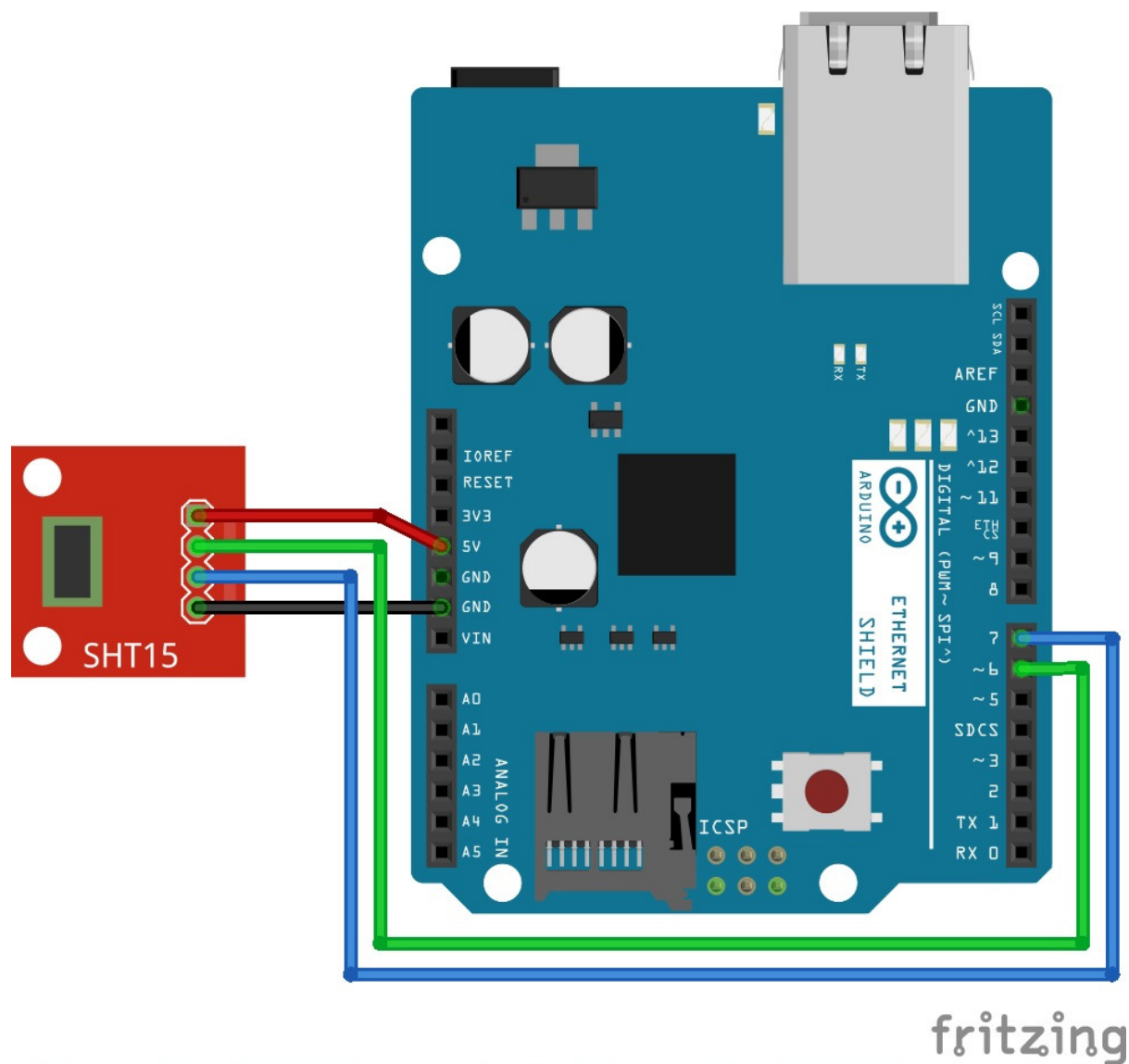
## Manuelle Konfiguration eines Sensors auf der OSeM

In dieser Anleitung wird beispielhaft die Anbindung eines SHT15 Sensors in das OSeM Netzwerk beschrieben. Für die Datenverarbeitung bzw. -übertragung nutzen wir einen Arduino Uno mit Arduino Ethernet Shield. Die REST Schnittstelle bietet aber auch jeder anderen programmierbaren und mit dem Internet verbundenen Messstation die Möglichkeit, Messungen auf der OSeM zu veröffentlichen und zu teilen.

### Materialien

- Arduino Uno R3
- Arduino Ethernet Shield R3
- [Sparkfun SHT15 Breakout](#)<sup>1</sup>

### Aufbau



- VCC zu 5V



- DATA zu Pin 6
- SCK zu Pin 7
- GND zu Arduino GND

## SHT1x Bibliothek

Für Sensoren der SHT1x Serie von Sensirion gibt es bereits eine einfache Arduino-Schnittstelle. [Lade die Bibliothek von Github herunter](#)<sup>2</sup> und entpacke sie in deinen `Arduino/libraries` Ordner. Importiert die Bibliothek wie gehabt in deinen Arduino Sketch, und definiere zusätzlich den Daten- und Taktpin entsprechend der Verkabelung in obiger Abbildung. Danach kannst du eine Verbindung zum Sensor herstellen:

```
#include<sht1x.h>
#define dataPin 6
#define clockPin 7
SHT1x sht1x(dataPin, clockPin);
```

Nun lassen sich über zwei Funktionen die Temperatur in Grad Celsius, sowie die relative Luftfeuchte in Prozent als Gleitkommazahl abspeichern:

```
float temp = sht1x.readTemperatureC();
float humi = sht1x.readHumidity();
```

## Manuelle Registrierung

Um einen Sensor mit der OSeM zu verbinden, musst du ihn [zuerst dort registrieren](#)<sup>3</sup>. Falls du dabei keinen der senseBox-Bausätze nutzt, muss in Schritt 4 der Registrierung die manuelle Konfiguration gewählt werden. Wie unten in der Abbildung dargestellt, wird dort für jedes gemessene Phänomen ein neuer Sensor angelegt:

SenseBox Home

SenseBox Photonik

Manuelle Konfiguration

☒ Manuelle Konfiguration

Phänomen	Einheit	Typ	Ändern
Temperatur	°C	SHT15	
<div> <div></div> <ul style="list-style-type: none"> <li>Temperatur</li> <li>Luftfeuchtigkeit</li> <li>Luftdruck</li> <li>Schall</li> <li>Licht</li> <li>Licht (digital)</li> <li>UV</li> <li>Kamera</li> </ul> </div>			

+ Sensor hinzufügen

## openSenseMap API

Eine REST Schnittstelle regelt den Zugang zur Datenbank auf dem OSeM Server. Intern ist jede Messstation mit ihren Sensoren (bzw. Phänomenen) verknüpft, die bei der Registrierung angegeben wurden. In unserem Falle haben wir eine senseBox ID für die Station, sowie jeweils eine Sensor ID für Temperatur- und Luftfeuchtheitsmessungen bei der Registrierung generiert. Die IDs werden dir nach der Registrierung per Mail zugeschickt. Jede Messung wird dann über das HTTP Protokoll mit der `POST` Operation an den Server gesendet. Dazu muss eine eindeutige URI angegeben werden die wie folgt aufgebaut ist:

```
https://api.opensensemap.org/senseBoxID/SensorID
```

Hinweis: Sollte der verwendete Microcontroller nicht HTTPS-kompatibel sein, gibt es derzeit noch eine HTTP Schnittstelle: `http://opensensemap.org:8000/senseBoxID/sensorID`

Jede Messung wird einzeln im JSON Format über das `value` -Attribut an den Server gesendet. Angenommen, wir wollen von unserer Station (ID 1234) einen Messwert des Thermometers (ID abcd) von `22,5` an den OSeM Server schicken, dann sähe der vollständige HTTP POST Request folgendermaßen aus:

```
POST /boxes/1234/abcd HTTP/1.1
Host:opensensemap.org
Content-Type: application/json
Connection: close
Content-Length: 14

{"value":22.5}
```

Achtung: Ab Zeile 7 werden die JSON-Daten gesendet. Der Zeilenumbruch ( `\n` ) in Zeile 6 ist notwendig um die Operation korrekt auszuführen.

## Arduino OSeM Client

Nach der Registrierung wird ein Arduino Sketch generiert, den du als Anhang in einer Bestätigungsmail zugeschickt bekommst. Diesen Sketch musst du noch anpassen, indem die SHT1x Bibliothek eingefügt, sowie die benötigten Variablen und eine Sensorinstanz erstellt werden:

```
#include <SPI.h>
#include <Ethernet.h>

#include<sht1x.h>
#define dataPin 6
#define clockPin 7
SHT1x sht1x(dataPin, clockPin);

//senseBox ID
#define senseBox_ID "1234"
//Sensor IDs
#define TEMPERATURESENSOR_ID "abcd"
#define HUMIDITYSENSOR_ID "efgh"
```

Innerhalb der `if`-Anweisung in der `loop` -Funktion, musst du nacheinander die Sensoren auslesen und mit der Hilfsfunktion `postFloatValue()` hochladen.

```
void loop()
{
  //Upload der Daten mit konstanter Frequenz
  if (millis() - oldTime >= postInterval)
  {
    oldTime = millis();
    temperature = sht1x.readTemperatureC();
```

```
postFloatValue(temperature, 1, temperatureSensorID);  
humidity = sht1x.readHumidity();  
postFloatValue(humidity, 0, humiditySensorID);  
}  
}
```

Falls du ein Ethernet Modul nutzt, welches nicht mit der Ethernet Bibliothek kompatibel ist muss der Sketch entsprechend angepasst werden. Solltest du weitere Fragen dazu haben, kannst du dich auch direkt [an unseren Support wenden](#)<sup>4</sup>.

---

<sup>1</sup>. <https://www.sparkfun.com/products/8257> ↩

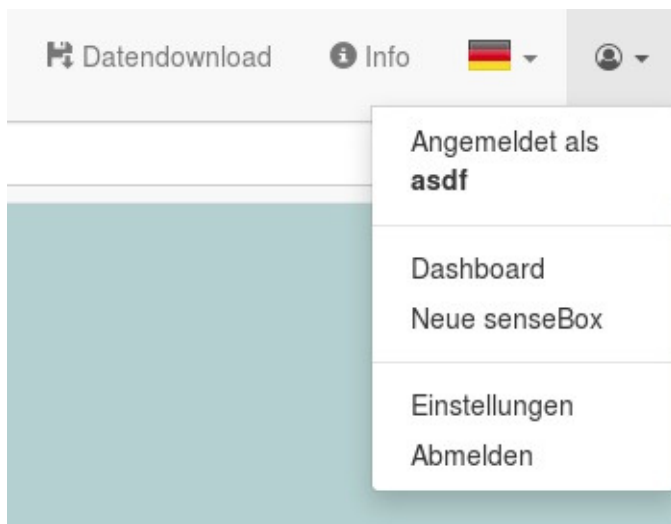
<sup>2</sup>. <https://github.com/practicalarduino/SHT1x> ↩

<sup>3</sup>. <https://opensensemap.org/register> ↩

<sup>4</sup>. <mailto:support@sensebox.de> ↩

## Boxen verwalten

Jeder registrierte Nutzer kann beliebig viele Boxen auf der openSenseMap verwalten. Wenn die Einstellungen einer Box nachträglich geändert oder der Sketch heruntergeladen werden sollen ist dies über das Dashboard möglich. Dieses ist in der Menüleiste unter dem User-Icon verfügbbar, sobald ein Nutzer eingeloggt ist:



Im Dashboard können...

- [neue Boxen registriert werden](#)<sup>1</sup>,
- bestehende Boxen angepasst oder entfernt werden,
- Sketches zur Programmierung einer Box heruntergeladen werden.

## senseBox anpassen

Durch Klick auf den ÄNDERN-Button einer Box im Dashboard können beliebige Eigenschaften dieser Box nachträglich verändert werden. Nachdem in einem der Abschnitte Änderungen vorgenommen wurden, werden diese durch Klick auf das Diskettensymbol oben rechts übernommen.

Hinweis: Wenn die Sensorkonfiguration geändert wurde, muss der Programmcode der senseBox in den allermeisten Fällen ebenfalls aktualisiert werden. Dieser ist unter dem Reiter Skript zu finden, um ihn in die Arduino IDE zu kopieren. Falls die WiFi-Version der senseBox verwendet wird, muss erneut die SSID und das WiFi Passwort im Sketch ersetzt werden!

## senseBox löschen

Falls eine senseBox nicht mehr verwendet wird, oder die Messungen dieser Box von der openSenseMap entfernt werden sollen, kann diese entfernt werden. Dazu muss im Bearbeitungsmodus (s.o.) im Reiter Allgemein unter dem Feld "senseBox löschen" der Wert `DELETE` eingetragen werden. Anschließend erscheint unter dem Feld ein Button, durch welchen die senseBox und ihre Messungen gelöscht werden.

Achtung: Hierdurch werden neben der senseBox alle hinterlegten Sensordaten unwiderruflich entfernt! Da die Messungen auch für eine Nachträgliche Datenauswertung wertvoll sein können, sollte abgewägt werden ob die senseBox gelöscht werden sollte.

<sup>1</sup>. Siehe [1.2 Registrierung](#) ↩

# Datendownload

Es bestehen mehrere Möglichkeiten Sensordaten von der openSenseMap herunterzuladen. Abhängig von der Fragestellung bietet sich je eine Option an.

## Daten zu einer Box

Unter <https://archive.opensensemap.org> wird ein Archiv für sämtliche Messungen in der openSenseMap Datenbank geführt. Hier sind nach Tag und Box gegliederte Messwerte als CSV beziehungsweise ZIP-Archiv verfügbar.

## Daten zu einem Phänomen

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](#)<sup>1</sup> 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
....
```

## API-Download

Falls die beiden genannten Möglichkeiten nicht flexibel genug sind, können über die [REST API unter /boxes/data](#)<sup>2</sup> auch komplexe Anfragen gestellt werden.

Für solche anfragen bietet sich das Kommandozeilenwerkzeug `curl` an. Unter Linux ein Terminal öffnen und beispielsweise folgenden Befehl eingeben, um sämtliche Temperatur-Messungen im geographischen Bereich 51°N - 52°N, 7°E - 8°E in die Datei `measurements.csv` herunterzuladen:

```
curl "https://api.opensensemap.org/boxes/data?phenomenon=Temperatur&bbox=7,51,8,52" > measurements.csv
```

Andere geeignete Parameter (Zeitraum, Box-IDs, ...) lassen sich der verlinkten API-Dokumentation entnehmen.

---

<sup>1</sup>. Siehe [1.5 Datenanalyse](#) [↩](#)

<sup>2</sup>. Siehe [3.1 REST API > get-latest-measurements-for-a-phenomenon-as-csv-](#) [↩](#)

# 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).

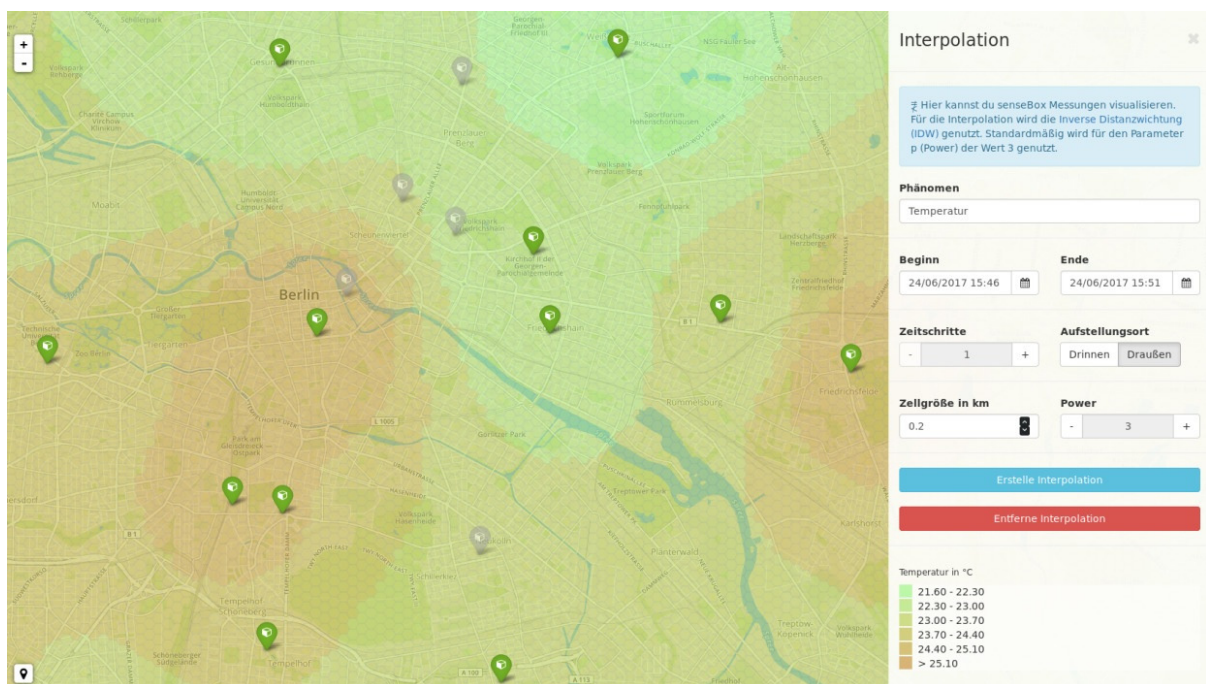
Anschließend wird unterhalb der Filtereinstellungen eine Auflistung der auf die Kriterien zutreffenden senseBoxen angezeigt.

Dieser Filter bezieht sich auch auf die anderen Datenanalyse-Funktionen [Interpolation](#)<sup>1</sup> und [Datendownload](#)<sup>2</sup>!

## Interpolation

Im Reiter Interpolation lassen sich die Daten mehrerer senseBoxen zu einem Phänomen 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.



Es wird das Interpolationsverfahren IDW (Inverse Distance Weighting) verwendet, welches als Parameter einen power-Wert hat. Dieser ist der Exponent, mit welchem die Distanz eines Messwerts zu einem Ort der Interpoliert wird gewichtet wird. Ein niedriger Wert für power bezieht also Werte aus größeren Distanzen ähnlich stark wie aus der Nähe ein, während ein hoher power-Wert insbesondere Werte aus unmittelbarer Entfernung betrachtet.

Nach Einstellung der Parameter wird die Interpolation auf unserem Server berechnet. Wenn die Kalkulation abgeschlossen ist, wird das Ergebnis als Heatmap in der Karte angezeigt. Die Zellgröße der Kalkulation kann auch eingestellt werden um detailliertere Ergebnisse zu erhalten; hierbei ist aber zu beachten dass die Berechnungszeit bei kleineren Zellen sehr stark ansteigt.



1. Siehe [1.5 Datenanalyse > interpolation](#) ↩
2. Siehe [1.4 Datendownload](#) ↩

# Datenübertragung über MQTT

Diese Anleitung beschreibt die Möglichkeit, Messwerte über MQTT an die openSenseMap zu senden. Die openSenseMap ist in der Lage, sich als MQTT Client mit einem öffentlichen MQTT Broker zu verbinden. Einen eigenen MQTT Broker bietet die openSenseMap nicht an. Der openSenseMap MQTT Client verbindet sich, wenn nicht anders in den Verbindungseinstellungen angegeben, mit einer 13 Stelligen Id mit prefix `osem_` gefolgt von 8 zufälligen Ziffern und Buchstaben von A bis F.

Je registrierter senseBox müssen separate MQTT Einstellungen vorgenommen werden. Für eine Verbindung mit einem Broker können die folgenden Parameter angegeben werden. Alle angegebenen Einstellungen werden in der Datenbank der openSenseMap gespeichert. Es bietet sich also an, eigene Zugangsdaten einzurichten.

## URL

Die Adresse zum MQTT Broker. Sollte mit `mqtt://` oder `ws://` beginnen. Sollte der MQTT Broker Authentifizierung mittels Nutzernamen und Passwort benötigen, kann dieser in der URL kodiert werden. Die URL sollte dann wie folgt aussehen: `mqtt://username:password@hostname.of.mqtt.broker`

## Topic

Das MQTT Topic unter dem die openSenseMap Nachrichten empfangen soll. Zum Beispiel `home/temperatures/outside`

## Nachrichtenformat

Hier sollte zwischen `json` und `csv` ausgewählt werden. Die Formate entsprechen JSON-Array und csv dokumentiert in [docs.opensensemap.org](https://docs.opensensemap.org)<sup>1</sup>.

## Dekodierungsoptionen

Erwartet ein JSON Objekt. Nur für Nachrichtenformat `json`: Erlaubt es, unter dem Schlüssel `jsonPath` einen JSONPath Ausdruck anzugeben, welches die Position der JSON kodierten Daten angibt. Beispiel: `{"jsonPath": "$.payload_fields"}`

## Verbindungsoptionen

Erwartet ein JSON Objekt. Erlaubt es, dem MQTT Client Verbindungsoptionen zu übergeben. Die Schlüssel `keepAlive`, `reschedulePings`, `clientId`, `username` und `password` von <https://github.com/mqttjs/MQTT.js#client><sup>2</sup> sind erlaubt.

---

<sup>1</sup>. <https://docs.opensensemap.org/#api-Measurements-postNewMeasurements> ←

<sup>2</sup>. <https://github.com/mqttjs/MQTT.js#client> ←

## Upload über LoRaWAN

Es ist möglich Sensordaten per LoRaWAN™ durch das [TheThingsNetwork](#)<sup>1</sup> (TTN) auf die openSenseMap zu laden. LoRa ist ein zunehmend Verbreitung findender Funkstandard, welcher ähnlich wie WiFi digitale Datenübertragung in einem IP-Netzwerk erlaubt, jedoch deutlich andere Features bietet:

- Datendurchsatz: 300 - 3000 Bit/s
- Reichweite: bis zu 15km

TTN ist eins von mehreren Projekten, welches die zur Funk-Hardware zugehörige Infrastruktur für das IP-Netzwerk implementiert, wodurch registrierte Geräte mit dem Internet verbunden werden können.

Nutzer können Gateways sowie Nodes zu dem Netzwerk hinzufügen. Mit dem LoRa Sender auf dem [Dragino Shield](#)<sup>2</sup> haben wir bisher gute Erfahrungen gemacht.

## TTN openSenseMap Integration

Die openSenseMap bietet eine direkte Integration in das TTN Netzwerk, was die Konfiguration stark vereinfacht.


### Registrierung in TTN Console

Um ein Gerät in das TTN einzubinden, muss für dieses zunächst unter [thethingsnetwork.org](#)<sup>3</sup> eine Application und ein Device registriert werden. Hierbei erhält man eine `app_id` und eine `dev_id`.

Für die registrierte Application muss die HTTP Integration unter [https://console.thethingsnetwork.org/applications/DEINE\\_APPID/integrations/create/http-ttn](https://console.thethingsnetwork.org/applications/DEINE_APPID/integrations/create/http-ttn) aktiviert werden. Diese muss konfiguriert werden, dass sie die Nachrichten von Devices per `POST` an `https://ttn.opensensemap.org/v1.1` weiterleitet. Das Authorization-Feld kann leer bleiben!

[Overview](#) [Devices](#) [Payload Formats](#) [Integrations](#) [Data](#) [Settings](#)

**ADD INTEGRATION**



**HTTP Integration** (v2.5.1)  
The Things Industries B.V.  
Sends uplink data to an endpoint and receives downlink data over HTTP.  
[documentation](#)

**Process ID**  
The unique identifier of the new integration process

**Access Key**  
The access key used for downlink  
 [devices](#) [messages](#)

**URL**  
The URL of the endpoint

**Method**  
The HTTP method to use

Für die Datenübertragung zur openSenseMap müssen die `app_id` und `dev_id` bei der Registrierung auf der openSenseMap in der TTN-Konfiguration angegeben werden. Darüber hinaus muss ein passendes Decoding-Profil konfiguriert werden, welches bestimmt wie die - wegen der geringen Bandbreite als rohe Bytes übertragenen - Daten als Messungen interpretiert werden sollen.

## Erweitert

MQTT >

TheThingsNetwork - TTN >

Die openSenseMap bietet eine Integration mit [TheThingsNetwork](#) an. Für eine Erklärung der Parameter siehe [hier](#)

☒ **TheThingsNetwork**

**Dekodierungs-Profil**

senseBox:home

**TTN Application-ID**

my-osem-app

**TTN Device-ID**

my-osem-device

**Dekodierungsoptionen**

[]

**Port**

Optional kann im Feld `port` noch der Port angegeben werden, auf welchem der Sender seine Daten an das TTN schickt. So lassen sich die selbe `app_id` und `dev_id` für mehrere Sensorstationen verwenden.

## Decoding Profile

Für eine Box muss passend zu den übertragenen Messdaten ein Decoding-Profil ausgewählt oder definiert werden. Die Auswahl des Decoding-Profiles ist von dem Encoding der Nachrichten auf dem Mikrocontroller, und ob im TTN eine Payload-Function eingestellt wurde abhängig.

- Für die `senseBox:home` (ohne Erweiterungen) kann das `senseBox:home` Profil verwendet werden.
- Werden die Messungen auf der LoRa-Node mit der `lora-serialization` -Library encodiert, sollte das `lora-serialization` Profil verwendet werden.
- Mit dem `json` Profil werden beliebige andere Encodings unterstützt, falls eine Payload-Function in der TTN Console die Nachrichten passend decodiert.

Im Folgenden wird erklärt wie die unterstützten Profile konfiguriert werden:

### sensebox/home

Dieses Profil ist zugeschnitten auf die mit der `senseBox:home` gelieferten Sensoren, in Verwendung mit [diesen Arduino Sketch](#)<sup>4</sup>. Neben der Angabe `sensebox/home` unter `profile` ist keine weitere Konfiguration notwendig.

### lora-serialization

Für Sensorstationen, welche eine spezielle Sensorkonfiguration haben, können durch das `lora-serialization` Profil nahezu beliebige Daten annehmen. Hierzu nutzen wir die `lora-serialization`<sup>5</sup> Bibliothek, welche ein einheitliches Encoding auf dem Microcontroller, und Decoding am anderen Ende der Leitung erlaubt.

Es werden die Encodings `temperature`, `humidity`, `unixtime`, `uint8` und `uint16` unterstützt, welche pro Sensor unter Dekodierungsoptionen angegeben werden müssen. Die Zuordnung des Sensors kann über eine der Properties `sensor_id`, `sensor_title`, `sensor_unit`, `sensor_type` erfolgen.

Ein Beispiel für zwei Sensoren sähe so aus:

```
[
  { "decoder": "temperature", "sensor_title": "Temperatur" },
  { "decoder": "humidity", "sensor_unit": "%" }
]
```

Hinweis: Die Reihenfolge der Sensoren muss hier beim Arduino und der `openSenseMap` identisch sein!

Wenn ein `unixtime` Decoder angegeben wird, wird dessen Zeitstempel für alle im Folgenden angegebenen Messungen verwendet. Andernfalls wird der Moment verwendet, in dem das erste Gateway die Nachricht erhält. Beispiel:

```
[
  { "decoder": "unixtime" },
  { "decoder": "temperature", "sensor_title": "Temperatur" }
]
```

## json - Decoding mit TTN Payload Function

Falls die `lora-serialization` Library nicht zur Wahl steht, können Messungen schon auf Seite des TTN mittels einer Payload Function dekodiert werden, sodass hier beliebige Datenformate unterstützt werden.

Applications > wala

decoder converter validator encoder Remove decoder

```

1 function Decoder(bytes, port) {
2   return decode(bytes, [unixtime, uint16, uint16], ['time', 'sensor1', 'sensor2']);
3 }
4
5
6 // ---- contents of src/decoder.js ----
7 var bytesToInt = function(bytes) {
8   var i = 0;
9   for (var x = 0; x < bytes.length; x++) {
10    i |= +(bytes[x] << (x * 8));
11  }
12 }

```

decoder has unsaved changes – [undo changes](#)

**Payload**

00 00 00 00 91 04 8B 04 8 bytes 1 Test

```

{
  "sensor1": 1169,
  "sensor2": 1163,
  "time": 0
}

```

Das resultierende JSON muss kompatibel mit den von der `openSenseMap-API` verstandenen Measurement Formaten sein. Ein einfaches Beispiel:

```
{ "sensor_id1": "value1, "sensor_id2": "value2" }
```

Auf Seiten der openSenseMap ist keine Konfiguration notwendig.

---

1. <https://thethingsnetwork.org> ↩
2. [http://wiki.dragino.com/index.php?title=Lora\\_Shield](http://wiki.dragino.com/index.php?title=Lora_Shield) ↩
3. <https://console.thethingsnetwork.org/> ↩
4. <https://github.com/sensebox/random-sketches/tree/master/lora/dragino> ↩
5. <https://github.com/thesolarnomad/lora-serialization> ↩

## openSenseMap RESTful API

Die openSenseMap stellt eine API zur Abfrage von Daten zu Boxen und Messungen sowie zum Upload von Messungen unter der Adresse <https://api.opensensemap.org/> zur Verfügung.

Die Dokumentation der API-Routen ist [hier](https://docs.opensensemap.org)<sup>1</sup> zu finden.

---

<sup>1</sup>. <https://docs.opensensemap.org> ↩



# Anwendungen für die openSenseMap API

In der Community sind bereits einige Tools und Anwendungen entstanden, welche die openSenseMap integrieren oder als Schnittstelle nutzen.

Falls dir weitere Integrationen bekannt sind, lass es uns wissen!

## Visualisierung

- [DevelopmentSeed Dashboard<sup>1</sup>](#): elegantes Dashboard welches die Messungen einer senseBox anzeigt
- [HPI Makerclub Dashboard<sup>2</sup>](#) ([demo<sup>3</sup>](#))
- [senseBox Dashboard<sup>4</sup>](#): zeigt aktuelle Messwerte einer senseBox an
- [senseBox Widget<sup>1</sup>](#)

## Datenanalyse

- [opensensmap R client<sup>7</sup>](#): openSenseMap Messwerte und Boxen in der statistischen Umgebung R
- [senseBox openSenseMap R client<sup>8</sup>](#): An R API for the senseBox project. Download and analyse environmental data provided by <https://sensebox.de/en/>.

## Sensor Firmware

- [RasPi Wolkenbedeckungs-Sensor<sup>9</sup>](#)
- [GSM Wassersensor<sup>10</sup>](#)
- [mobile GPS senseBox<sup>11</sup>](#)
- [BigGIS LoRa senseBox<sup>12</sup>](#)

## Sonstige

- [Alexa senseBox-Skill<sup>13</sup>](#)
- [openhab2 smart home senseBox Integration<sup>14</sup>](#)
- [openSenseMap Game<sup>15</sup>](#)

---

<sup>1</sup>. <https://github.com/developmentseed/sense> ↩

<sup>2</sup>. <https://github.com/HPIMakerKlub/sensebox> ↩

<sup>3</sup>. <http://rawgit.com/HPIMakerKlub/sensebox/master/statistics/sensor.html?senseBoxID=5719c4037514d05c121e317c> ↩

<sup>4</sup>. <https://github.com/sensebox/sensebox-dashboard> ↩

<sup>1</sup>. Siehe [3.3 HTML Widget](#) ↩

<sup>5</sup>. <https://github.com/Avipsa1/Sensebox> ↩

<sup>6</sup>. <https://github.com/sensebox/Innotruck> ↩

<sup>7</sup>. <https://github.com/noerw/opensensmapR> ↩

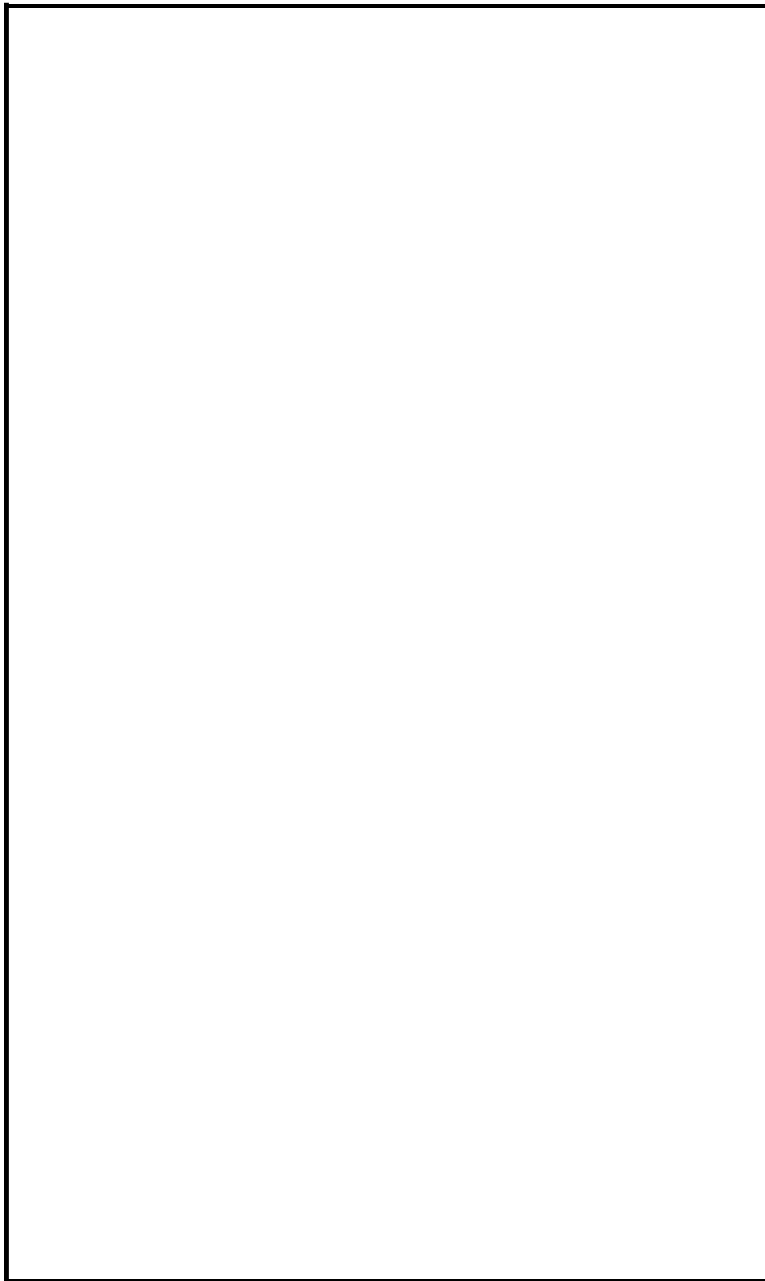
~

8. <https://github.com/JohannesFriedrich/senseBox> ↩
9. <https://github.com/felixerdy/senseBox-cloud> ↩
10. <https://github.com/felixerdy/GSM-Temperature-senseBox> ↩
11. <https://github.com/noerw/mobile-sensebox> ↩
12. <https://github.com/biggis-project/sensebox-station> ↩
13. <https://github.com/Zeygon/alexa-sensebox> ↩
14. <https://github.com/hakan42/openhab2-addons/tree/sensebox-binding/addons/binding/org.openhab.binding.sensebox> ↩
15. <https://github.com/MaxMoody/Senseboxgamification> ↩

## openSenseMap Widget

Um die Sensordaten einer senseBox nicht nur auf `opensensemap.org` anzuzeigen, sondern beispielsweise auch in eine eigene Website einzubinden kann das openSenseMap Widget verwendet werden.

### Beispiel



### Anleitung

Dieses Widget kann in eine Seite eingebunden werden und zeigt dann dort Informationen zu einer ausgewählten, auf der openSenseMap registrierten senseBox an. Um das Widget einzubinden, muss man einfach folgende Codezeilen in sein HTML-Element einfügen:

```
<iframe
  src="https://sensebox.github.io/opensensemap-widget/iframe.html?senseboxId=DEINE-SENSEBOXID"
  width="400"
  height="600"
  marginwidth="8" marginheight="8"
  hspace="0" vspace="0"
  frameborder="0"
  scrolling="no"
></iframe>
```

In der URL muss `DEINE-SENSEBOXID` durch die ID deiner Box ersetzt werden, wie sie auch in der OpenSenseMap gespeichert ist. Anschließend ist das Widget sofort auf der Seite verfügbar. Die Größe lässt sich über die `width` und `height` Attribute anpassen.

Der Quellcode ist auf GitHub unter [sensebox/opensensemap-widget](https://github.com/sensebox/opensensemap-widget)<sup>1</sup> zu finden.

---

<sup>1</sup> <https://github.com/sensebox/opensensemap-widget> ↩