Lärm-API

rxf 2023-11-29

Über diese API können Lärm-Daten für alle Sensoren direkt aus der Datenbank ausgelesen werden. Es existieren verschiedene Aufrufe.
Im folgenden wird der HOST, auf dem das Programm läuft (der Docker-Container *noise*) als **<host>** dargestellt.

Live - Messdaten

Aufruf

https://<host>/api/getsensordata?sensorid=xxx&csv=true&span=xxx&datetime=xxxx&pretty
oder ohne Parameter
https://<host>/api/getsensordata

Parameter

Alle Parameter sind optional und werden mit sinnvollen Defaults belegt, wenn sie fehlen.

- ***sensorid:*** Nummer des Sensors, für den die Daten abgerufen werden sollen. Ist der Sensor nicht in der Datenbank, wird ein leeres Werte-Array zurückgegeben. Default = 1 (Sensor 1 existiert nicht, also wird bei fehlender Sensornummer ein leeres Array zurück gegeben)
- ***span:*** Zeitspanne in Tagen, für die die Daten ausgegeben werden. Default = 1, max = 30.
- ***datetime*** Zeitpunkt, ab dem die Ausgabe beginnen soll. Die Zeitangabe erfolgt im ISO8601-Format (also YYY-MM-HHThh:mmZ).
    - Ist dieser Wert angegeben, so gilt für die zu berechnende Zeitspanne:
      Startzeitpunkt = ***datetime***, Endzeitpunkt = ***datetime + span (Tage)***
    - Ist dieser Wert **nicht** angegeben, so berechnet sich die Zeitspanne zu:
      Startzeitpunkt = ***aktuelle Zeit - span (Tage)***, Endzeitpunkt = ***aktuelle Zeit***
- ***csv:*** Ist ***csv=true*** angegeben, so werden die Daten als CSV-Datei ausgegeben. Ohne das ***csv*** werden die Daten als JSON ausgegeben. Wird bei JSON-Ausgabe noch der Parameter **pretty** angegeben, so werden die JSON-Daten in einem gut lesbaren Format ausgegeben.

Ausgabe

Das Ausgabe-Format ist entweder ein JSON-Dokument oder eine CSV-Datei.

Das JSON-Dokument hat z.B. folgenden Aufbau:

```
{
  "err": null,
  "options": {
    "sid": 37833,
    "indoor": 0,
    "span": 1,
    "start": "2023-11-28T14:21:21Z",
    "data": "live",
    "count": 73
  },
  "values": [
    {
      "datetime": "2023-11-29T10:55:56Z",
      "LA_min": 35.94,
      "LA_max": 47.12,
      "LAeq": 39.74,
      "E10tel_eq": 9418.895965228417
    },
    {
      "datetime": "2023-11-29T11:07:47Z",
      "LA_min": 36.45,
      "LA_max": 52,
      "LAeq": 41.53,
      "E10tel_eq": 14223.287871228213
    },
    ...
    ]
}
```

und als CSV-Datei:

```
datetime,LAeq,LAmax,LAmin,"10^(LAeq/10)"
2020-08-10T09:37:29,52.5,68.75,39.58,177827.94100389228
2020-08-10T09:39:59,44.84,67.48,39.21,30478.94989627983
2020-08-10T09:42:31,51.68,64.79,39.58,147231.25024327193
2020-08-10T09:45:02,44.43,56.95,39.46,27733.20104651838
2020-08-10T09:47:32,50.15,64.87,38.95,103514.2166679343
2020-08-10T09:50:03,47.17,58.74,39.78,52119.47111050811
...
2020-08-11T09:31:00,55.05,71.6,40.13,319889.51096913975
2020-08-11T09:33:31,53.91,66.21,40.62,246036.76041476274
```

Auswertungen der Messdaten

Aufruf

https://<host>/api/getsensordata?sensorid=xxx&data=xxx&peak=70&csv=true&span=xxx&datetime=xxxx&pretty

Parameter

Die Parameter sind die gleichen wie oben und optional. Als zusätzlicher Parameter kommt hier ***data*** hinzu. Darüber wird ausgewählt, welche Auswertung verwendet werden soll.

Die Bedeutung der einzelnen Werte für ***data***:

- ***data=live***: Ergibt die identische Ausgabe wie oben, d.h. data=live kann auch weggelassen werden.
- ***data=havg***: Ausgabe der Stunden-Mittelwerte. In der Augabe wird die Anzahl der Überschreitungen (als ***peakcount***) des Lärm-Pegels über einen bestimmten db-Wert (default = 70) angegeben. Mit dem Parameter ***peak=xxx*** kan dieser db-Wert eingestellt werden.
- ***data=davg***: Ausgabe der Tages-Mittelwerte. Für die Spitzenwert-Überschreitungen (***peakcount***) gilt das Gleiche wie bei Stundenmittel.
- ***data=daynight***: Ausgabe der Tages- und der Nacht-Mittelwerte. Die Tagesmittel werden von 6:00 bis 22:00 Uhr und die Nachtmittel von 22:00 bis 6:00 Uhr berechnet.
- ***data=lden***: Ausgabe des LDEN-Mittelwertes. Hier wird der Tag weiter aufgeteilt: Tag (D) ist von 6:00 bit 18:00 Uhr, Abend (E) ist von 18:00 bis 22:00 Uhr und Nacht (N) geht von 22:00 bis 6:00 Uhr. Die einzelnen Mittelwerte werden noch unterschiedlich bewertet und daraus dann der gesamte Mittelwert (der LDEN) berechnet.

Ausgabe

Hie sind die Augaben ebenfalls als JSON-Dokument oder als CSV-Datei (mit der Option ***csv=true***) abrufbar.

Stundenmittel (JSON):

```
{
  "err": null,
  "options": {
    "sid": 37833,
    "indoor": 0,
    "span": 7,
    "start": "2023-11-22T14:30:08Z",
    "data": "havg",
    "peak": 70,
    "count": 168
  },
  "values": [
    {
      "peakcount": 0,
      "datetime": "2023-11-24T00:00:00Z",
      "n_AVG": 38.42796079826682
    },
    {
      "peakcount": 0,
      "datetime": "2023-11-24T01:00:00Z",
      "n_AVG": 37.946929500486355
    },
    ....
    ]
}
```


Stundenmittel (CSV):


```
datetime,n_AVG,peakcount
2020-07-11T22:00:00Z,42.760062147998624,0
2020-07-11T23:00:00Z,40.712802683893074,0
2020-07-12T00:00:00Z,40.71753006698157,1
2020-07-12T01:00:00Z,39.25606411190053,0
```


Tagesmittel (JSON):

```
{
  "err": null,
  "options": {
    "sid": 37833,
    "indoor": 0,
    "span": 30,
    "start": "2023-10-30T00:00:00Z",
    "data": "davg",
    "peak": 70,
    "count": 30
  },
  "values": [
    {
      "datetime": "2023-11-24T00:00:00Z",
      "n_AVG": 43.21691983032072,
      "peakcount": 39
    },
    {
      "datetime": "2023-11-25T00:00:00Z",
      "n_AVG": null,
      "peakcount": 7
    },
    ...
  ]
}
```


Tagesmittel (CSV):


```
datetime,n_AVG,peakcount
2020-07-11T00:00:00Z,41.698517715727675,0
2020-07-12T00:00:00Z,47.33575272178947,32
2020-07-13T00:00:00Z,54.63937702461399,94
```


Tag-Nacht-Mittel (JSON)

```
{
  "err": null,
  "options": {
    "sid": 37833,
    "indoor": 0,
    "span": 30,
    "start": "2023-10-30T00:00:00Z",
    "data": "daynight",
    "count": 30
  },
  "values": [
    {
      "datetime": "2023-11-24T00:00:00Z",
      "n_dayAVG": 43.900464786300105,
      "n_nightAVG": 40.851023645531406
    },
    {
      "datetime": "2023-11-25T00:00:00Z",
      "n_dayAVG": null,
      "n_nightAVG": null
    },
    ...
  ]
}
```


Tag-Nacht-Mittel (CSV)


```
datetime,n_dayAVG,n_nightAVG
2020-07-11T22:00:00Z,0,40.93231330612532
2020-07-12T06:00:00Z,49.14770564689205,46.99391595352652
2020-07-13T06:00:00Z,56.305828057753864,64.57989823641728
2020-07-14T06:00:00Z,63.62474952627255,43.17808436755339
...
```


LDEN (JSON)

```
{
  "err": null,
  "options": {
    "sid": 37833,
    "indoor": 0,
    "span": 30,
    "start": "2023-10-30T00:00:00Z",
    "data": "lden",
    "count": 30
  },
  "values": [
    {
      "datetime": "2023-11-24T00:00:00Z",
      "lden": null
    },
    {
      "datetime": "2023-11-25T00:00:00Z",
      "lden": null
    },
    ...
  ]
}
```


LDEN (CSV)


```
datetime,lden
2020-07-11T22:00:00Z,46.161208717927906
2020-07-12T06:00:00Z,53.79509744837196
2020-07-13T06:00:00Z,69.9515379149476
2020-07-14T06:00:00Z,62.02707812443635
2020-07-15T07:00:00Z,56.98105132364213
```


Daten abspeichern

Um die ausgelesenen Messdaten in eine Datei abzuspeichern, kann am Einfachsten das Programm **curl** verwendet werden. Sollen z.B. die aktuellen Daten des Sensors *37833* als JSON in die Datei **sensor_37833.json** gespeichert werden, so sieht der Aufruf folgendermaßen aus:

```
curl -o sensor_37833.json "https://<host>/api/getsensordata?sensorid=37833"
```

Entprechendes gilt für die anderen Auswertungen.