The complexity of the uRADMonitor system stretches from a multitude of compact hardware detectors capable of sensing the environment, to the **big data software solutions** that can handle the huge amounts of data in real time. With the network spreading at a fast pace, periodic upgrades on the server side are a must, in order to provide a high quality, uninterrupted service.

Backend features

- 99.9% uptime guarantee
- RESTFul API for data access
- designed for linear scalability

Frontend features

- modern user interface
- powerful data visualization, stats and charting
- dashboard for user access to uRADMonitor hardware management
- web data access as CSV or JSON
- notifications and alerts

Applications

- Home monitoring
- Office and production space monitoring
- CBRN Monitoring
- Smart cities
- Internet of things
- Automation systems

New features

The continuous improvements to the uRADMonitor network are demanding more changes both to the units themselves but also to the central infrastructure. While we saw considerable progress on the hardware side with the model D and the new model A3 and model CITY units, the server was also been in the attention with important new additions like the Dashboard or the Dynamic ID system used in the Open Source KIT1 hardware.

To summarise the new v5.0 frontend in a word, that would be **"dynamic"**. Both the maps and the charts are interactive, and this translates to better access to the data. With the new uRADMonitor detectors (like the model D that runs on battery and has a built in GPS), the frontend had to support a new class of uRADMonitor devices, the mobile units.

Summary

1. The backend

This is a separate server, in charge of the system database and the uRADMonitor RESTful APIs. Its purpose is to provide input/output real time data operations via a mature API interface. It receives data from the distributed detectors, and provides data to the frontend, mobile apps and other parties, all via API calls. The data is stored in a big-data ready database.

1.1 Specifications

At the moment of documenting this datasheet, the backend is projected to support a network of up to 1000 units in size and uses the following specifications:

CPU	MEMORY	STORAGE	NETWORK	OS
Intel(R) Xeon(R) CPU E5-2680 v3 @ 2.50GHz, 4 cores	8.0GB	160GB SSD	40Gbps network in 5Gbps network out	CentOS 7

The server is designed for linear scalability, to cope with the increasing network size.

1.2 System diagram

A simplified overview of the backend architecture with its major components can be seen in the diagram below:

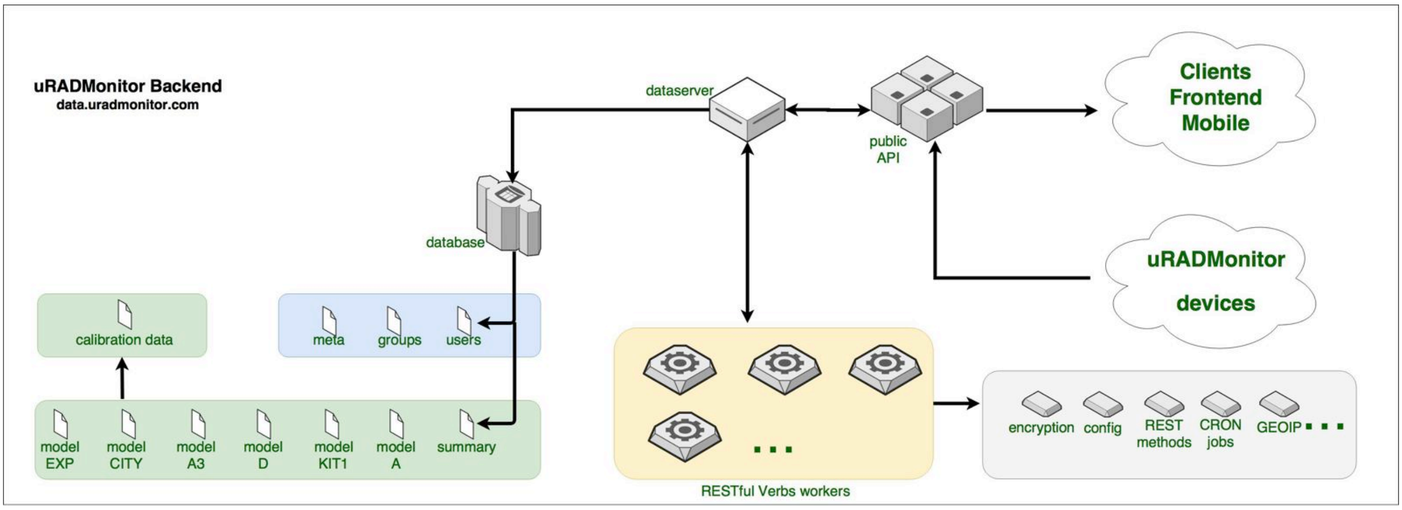


Figure 1: Backend server diagram

1.3 User database

The uRADMonitor backend implements user accounts in a separate database. This is used to permit login to the system, access data, manage devices and set automated notifications. Each user can be owner of a number of hardware devices, access their data and manage various settings for them. User hardware management is implemented via the Dashboard. Each user account can be used to authenticate for API calls using a pair of credentials formed from the user ID and the corresponding user Key. The user ID is associated to the username, while the user Key must be kept secret and is generated from the user password.

Each user has access to a profile page, where they can change personal information or password.

1.4 Owner vs global access

Each uRADMonitor unit can have maximum one owner. The owner is the user that can manage the unit, including changing its settings in the Dashboard or accessing the data.

A second user can have the meta global access data configured for the same unit (including others) and so will have access to the data, but only the owner can manage the unit.

1.5 RESTful API interface

REST API does not require the client to know anything about the structure of the API. Rather, the server needs to provide whatever information the client needs to interact with the service. An HTML form is an example of this: The server specifies the location of the resource, and the required fields. The browser doesn't know in advance where to submit the information, and it doesn't know in advance what information to submit. Both forms of information are entirely supplied by the server. Lookups should use GET requests. PUT, POST, and DELETE requests should be used for creation, mutation, and deletion.
The API is called for both directions of data transfer (upload and download). The uRADMonitor devices use the API to upload their measurements to the server, for further processing and storage in the database. The API is then used to access data by the frontend, the mobile app or third party systems using the uRADMonitor data.

1.6 API Authentication

Some API calls require authentication with user ID and user Key and will return results depending on the privileges and settings of the given user. To authenticate a call, the HTTP GET header must contain two custom fields, defined as follows:

X-User-id	Will contain the user ID.
X-User-hash	Will contain the user Key.

Both the user ID and the user Key are displayed in the Dashboard. Here is call example, using the authentication headers:

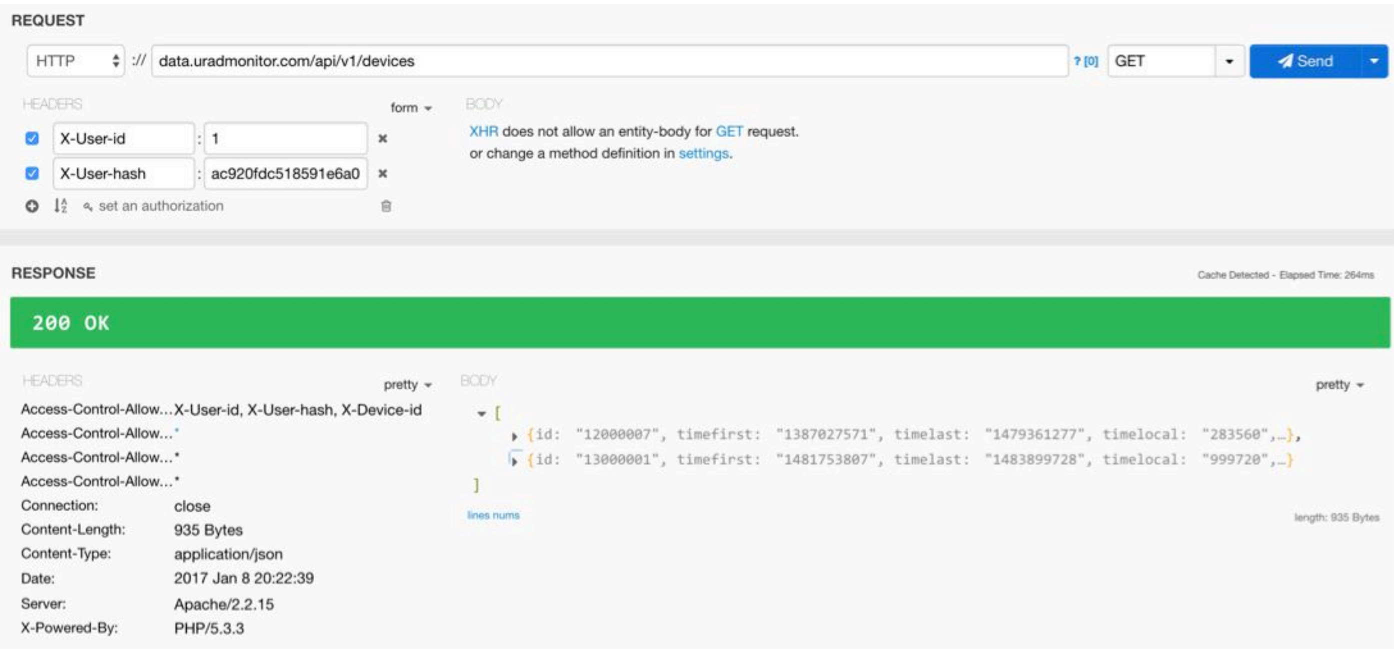


Figure 2: Authenticated API call

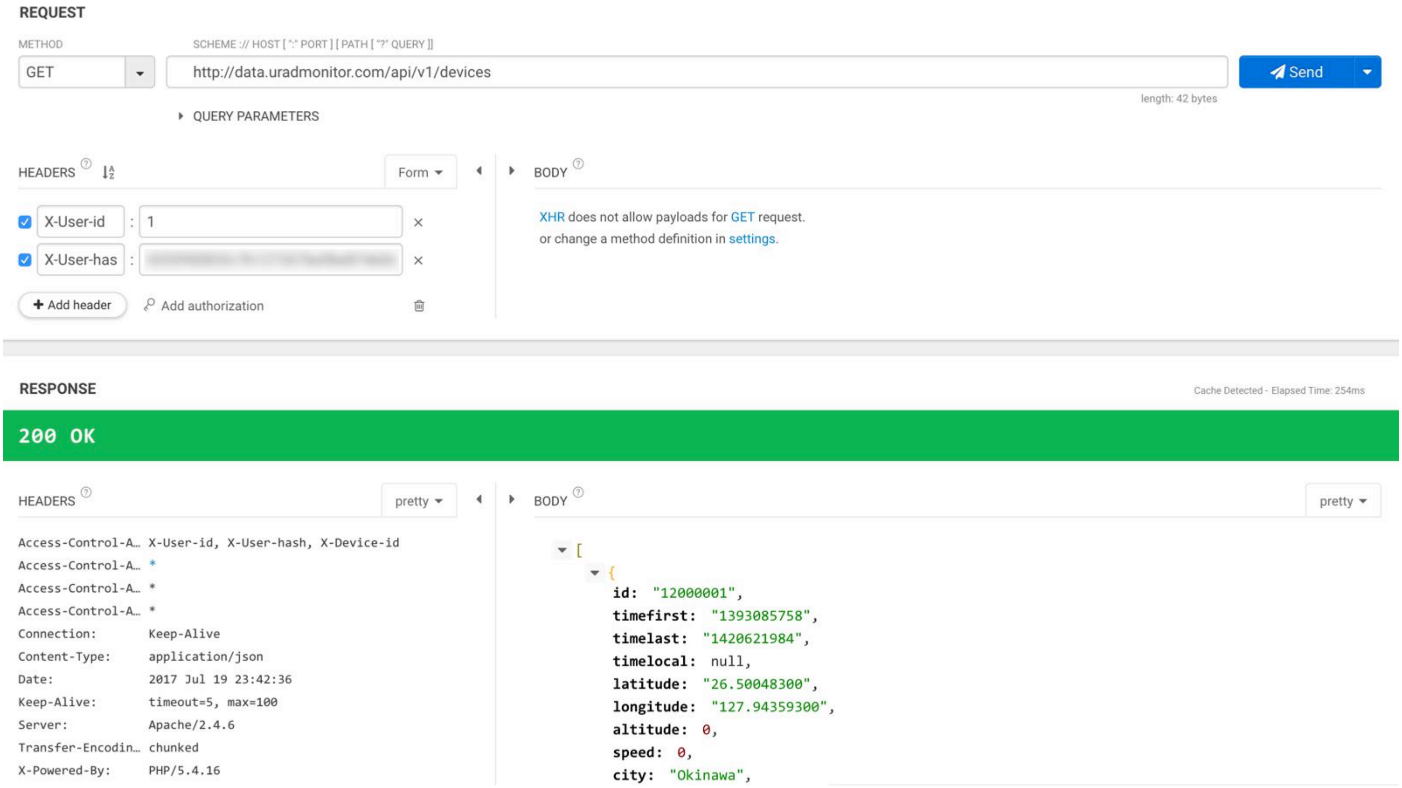
Below the list of API calls is presented. Those that require authentication will be marked accordingly.

1.7 API Calls for data access

For the uRADMonitor RESTful API, there is a common base url, defined as http://data.uradmonitor.com/api/v1/ followed by the following verbs:

1	*devices*	**Full URL**: http://data.uradmonitor.com/api/v1/devices
Method: HTTP GET		**Purpose**: data access
Description		Used to retrieve the list of uRADMonitor units assigned to the user account. The list includes the units the user is either set as owner or has global access to them.
Authentication		yes, using X-User-id and X-User-hash in HTTP Get header

Call example:



Return: summary array of uRADMonitor units in JSON format.

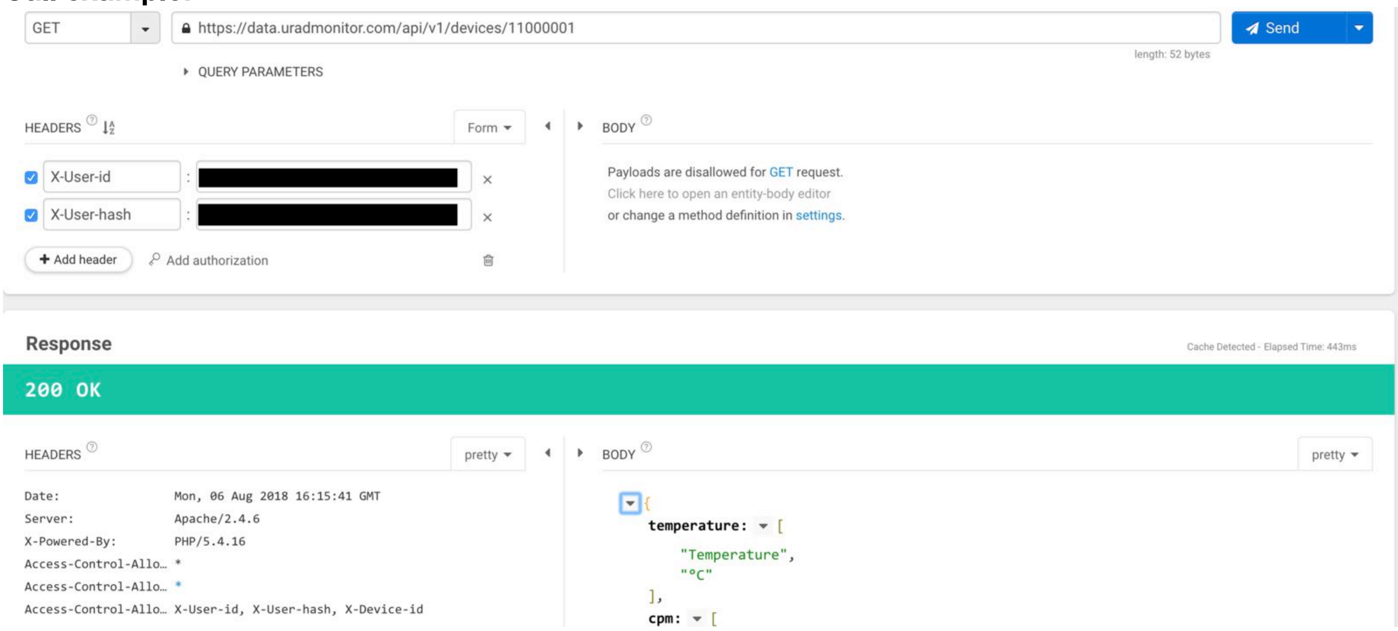

```
[
{ id : "82000050", timefirst : "1476801965", timelast : "1499877474", timelocal : "120", latitude : "37.46906600", longitude : "-79.21035800", altitude : 213, speed : 0, city : "Lynchburg", country : "US",
versionsw : "122", versionhw : "103", status : null, mobile : null, detector : "SI29BG",
factor : 0.01, avg_temperature : "25.39", avg_pressure : "99268", avg_humidity : "67.13",
avg_voc : "2669238", min_voc : "73049", max_voc : "11818108", avg_co2 : "514", avg_ch2o : "0.00",
avg_pm25 : "950", avg_noise : "0.00", avg_cpm : "11.40", avg_voltage : "380.97",
avg_duty : "219.68"}, {...}]
```


Each result in the array contains the following information:

id	the unique uRADMonitor unit ID
timefirst	unix timestamp containing the moment in time the unit first transmitted data
timelast	unix timestamp containing the moment in time of the last data transmission
timelocal	timestamp containing the number of seconds elapsed since the unit was last rebooted
latitude	latitude coordinate in decimal format
longitude	longitude coordinate in decimal format
altitude	altitude coordinate in meters
speed	unit speed in km/h
city	define base city for this unit
country	2 letter country code for the location of this unit
versionsw	firmware version
versionhw	hardware iteration version
status	1 if the unit is online, NULL if it is offline
mobile	1 if the unit is a mobile unit (eg. Model-D units or A3 units installed in buses)
detector	name of radiation detector sensor if the unit has such capabilities (only for Model A, KIT1, D and A3)
factor	CPM to Eq Dose Rate linear approximation conversion factor (dependent on "detector")
avg_XX	last 24hours average of the given sensor. Each unit model has a different number of avg_XX values returned, depending on its capabilities and the number of parameters it measures

2	*devices/[ID]*	**Full URL**: http://data.uradmonitor.com/api/v1/devices/[ID]
Method: HTTP GET		**Purpose**: data access
Description		ID is a unique uRADMonitor unit ID (eg. 110000AB) . This call is used to return the list of sensors of the specified unit.
Authentication		yes, using X-User-id and X-User-hash in HTTP Get header

Call example:

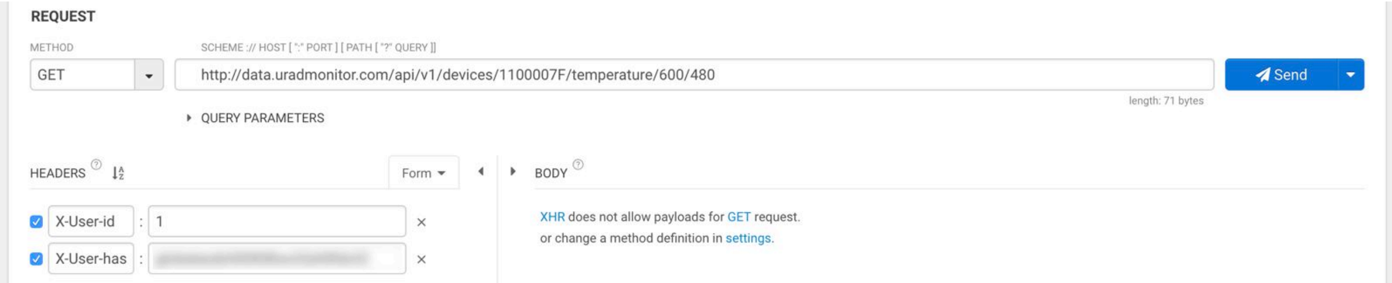


Return: list of supported sensors as an array in JSON format, including the unit of measure:

{temperature: ["Temperature","°C"],cpm: ["Radiation","cpm"],voltage: ["Voltage","V"],duty: ["Duty cycle","‰"],all: ["All",""]}

3	*devices/[ID]/[sensor]/[startinterval]/[stopinterval]*	**Full URL**: http://data.uradmonitor.com/api/v1/devices/[ID]/[sensor]/[startinterval]/[stopinterval]
Method: HTTP GET		**Purpose**: data access
Description		ID is a unique uRADMonitor unit ID (eg. 110000AB) . Sensor is a sensor name (eg. temperature) or you can also use the special keyword "all" to access data from all sensors installed on the unit. Startinterval is the the number of seconds from the moment of the present to get data from; "stopinterval" is optional and it represents the number of seconds from the moment of present to get data to. If "stopinterval" is not specified, the moment of present is used as the stop point. If there is no data for the query specified, you will receive an empty JSON array.
Authentication		yes, using X-User-id and X-User-hash in HTTP Get header

Call example:



Return: For the previous example call, we receive two temperature measurements, because we specified an interval of 120 seconds and the unit resolution was 1 minute:

[{ time : "1500498412", latitude : "61.11200000", longitude : "-149.90440000", altitude : "250.00", temperature : "22.00"}, { time : "1500498472", latitude : "61.11200000", longitude : "-149.90440000", altitude : "250.00", temperature : "21.93"}]

Additional information is presented under the API tab in the uRADMonitor dashboard: https://www.uradmonitor.com/dashboard/

1.8 Private API Calls

There are additional API calls used for Data upload, Dashboard, Hardware management, but these are not of public relevance - on the contrary they are private mechanisms comprising the uRADMonitor system.

2. The frontend

This is the visible side of the centralized server system, as it is in charge of generating the webpage showing all radiation and air quality readings. The webpage is a modern implementation using the powerful OpenLayers 3 mapping library. Thanks to JQuery and Dygraphs the data is shown in interactive charts, and it is generated on the client side to allow features like local timezone mapping and zooming, highly requested among uRADMonitor users.

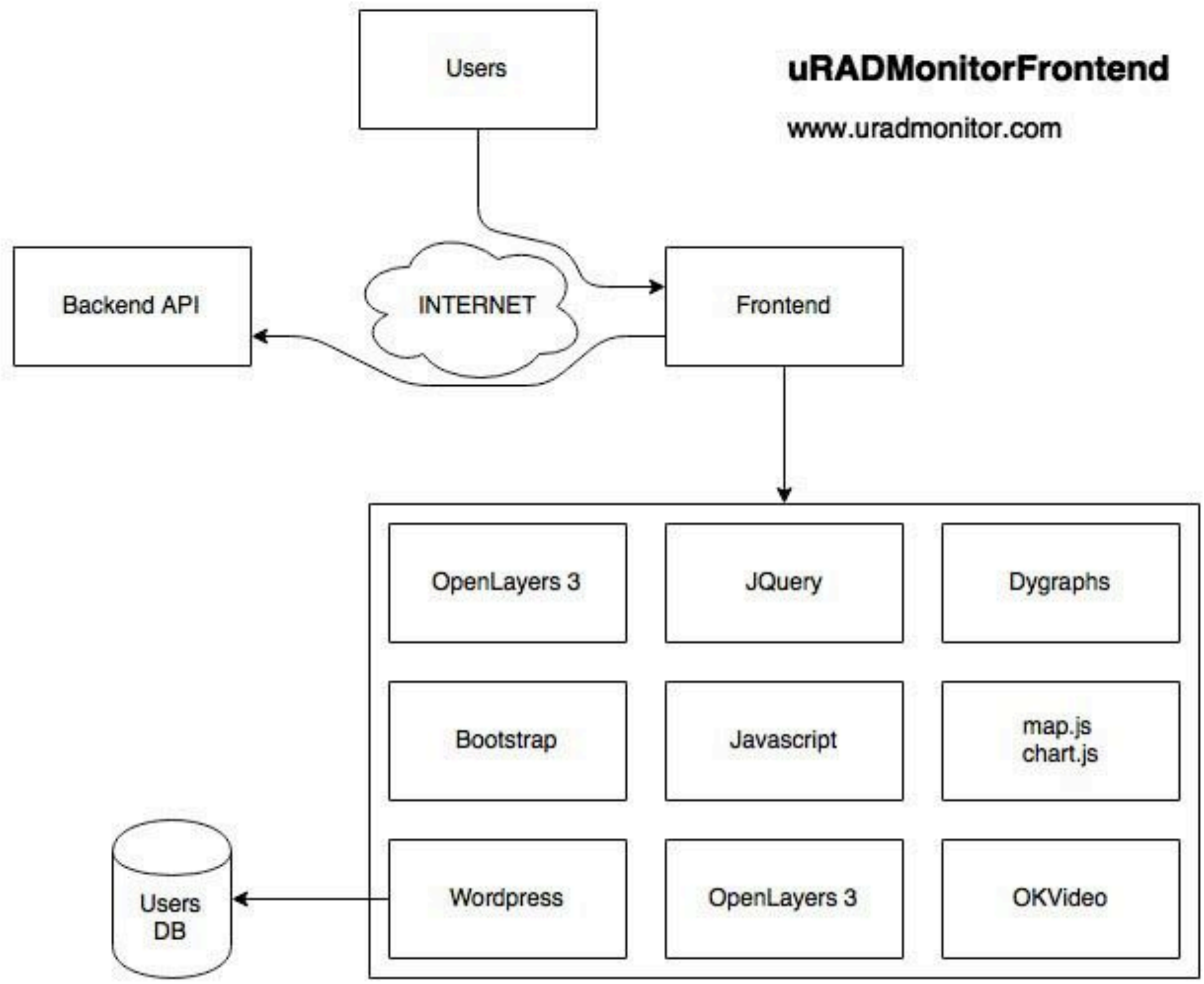


Figure 3: Major frontend components

2.1 Mobile units support

The map will refresh automatically to update the position of mobile units in real time. Speed and altitude are displayed, while the readings in the chart will update accordingly. Make sure the “Automated refresh” option is enabled, and your unit is selected (Blue dot). In “Cluster” and “Gradient” visualization modes, the mobile units are displayed as triangles, while the fixed units are represented as squares. A cluster of units is a circle, with its size proportional to the number of units contained. Clicking a cluster will unveil its comprising units automatically, by zooming the map to that particular location.

2.2 History view

If History view is enabled in the left menu while a mobile unit is selected, you will see the history chart at the bottom for the time interval you’ve selected, but also the corresponding path that unit covered on the map. This powerful feature will quickly identify various measurements to their exact location on the map.

2.3 The left menu

Includes two new selectors: one for the sensor parameter and one for the time interval. You can use them to see uniform global data on the map. If a particular unit is equipped with the sensor you’ve selected (eg. Temperature), its last 24hours average of the particular measurement will appear on the map. The default visualisation view is set for “Clusters”, meaning nearby units will be joined in clusters, depending on the zoom level, to make the vizualization cleaner and easier to follow. The clusters will be labeled with an average of all the units included, and the circle symbol will have its size proportional to the number of units included.

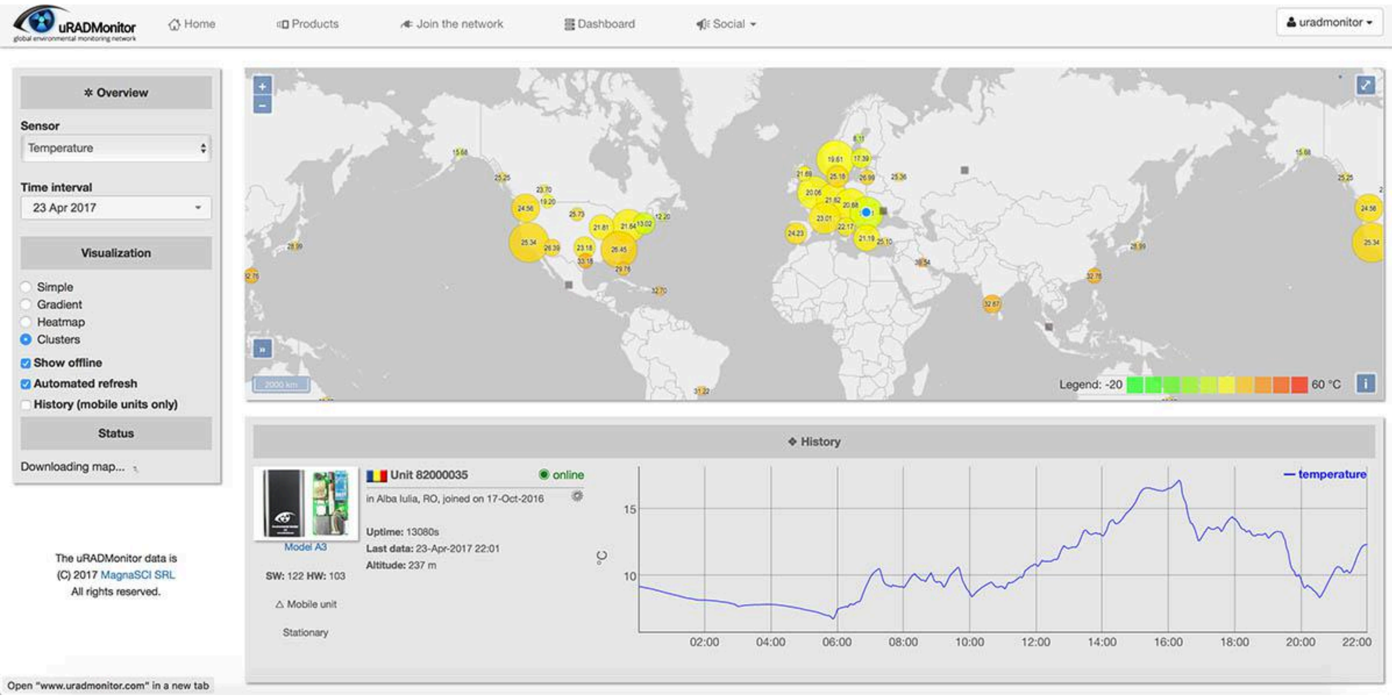


Figure 4: uRADMonitor frontend v5.0

Clicking a cluster will zoom in to its comprising area, while also showing the contained units at the bottom.

2.4 Direct ID access

When you click a cluster you see its comprising units. You can click these IDs to open a particular unit and see the readings history. If you want to open a unit directly, you can still use the previous syntax:
https://www.uradmonitor.com/?open=ID
Here is a quick example to that: https://www.uradmonitor.com/?open=11000017

2.5 Visualization options

The other visualization options include Simple, Gradient, Heatmap and Cluster, use them as needed. Change the Sensor selector to the value you are interested in.
The heatmap presents a color function that takes both the weight and the density as arguments. Keep that in mind when interpreting the visual representations. Zooming in will be needed to remove the density factor if color interpretation is required.

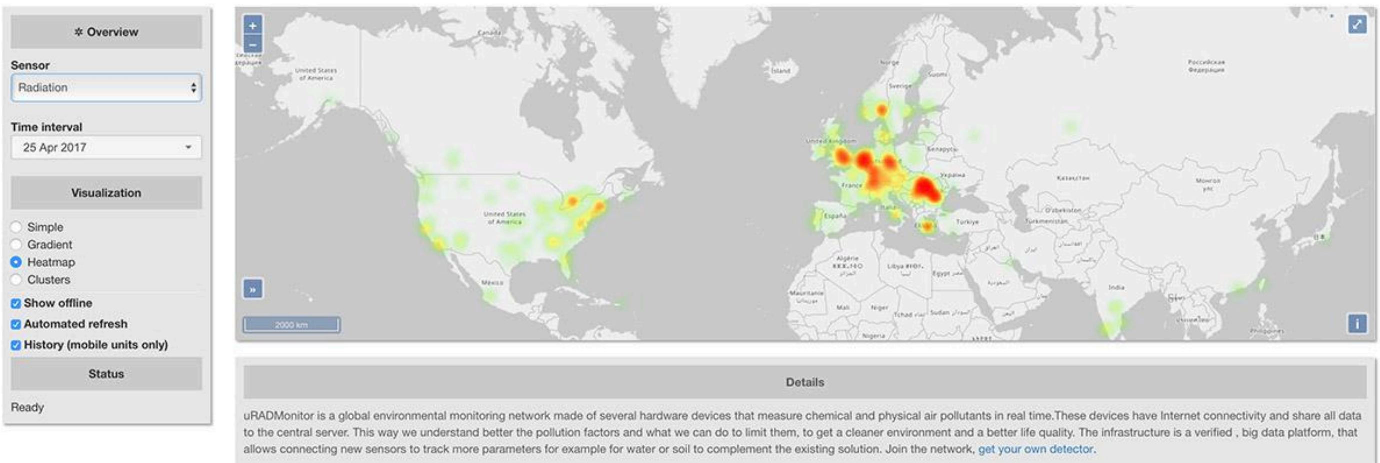


Figure 5: heatmap view

2.6 Legend

For the “Clusters” and “Gradient” visualization methods, a Legend is displayed at the right-bottom corner of the map, to provide a quick indication on the scale of the values represented on the map. The legend will show a minimum and a maximum value, and the Unit of Measure of the sensor you selected. For example, PM2.5 will use micrograms per cubic meter, while CO2 will display in ppm (parts per million):

Figure 6: map color legend

2.7 More detailed unit view

When selecting a unit, the bottom part of the screen will load a chart showing readings for the selected time interval, and for the selected sensor:

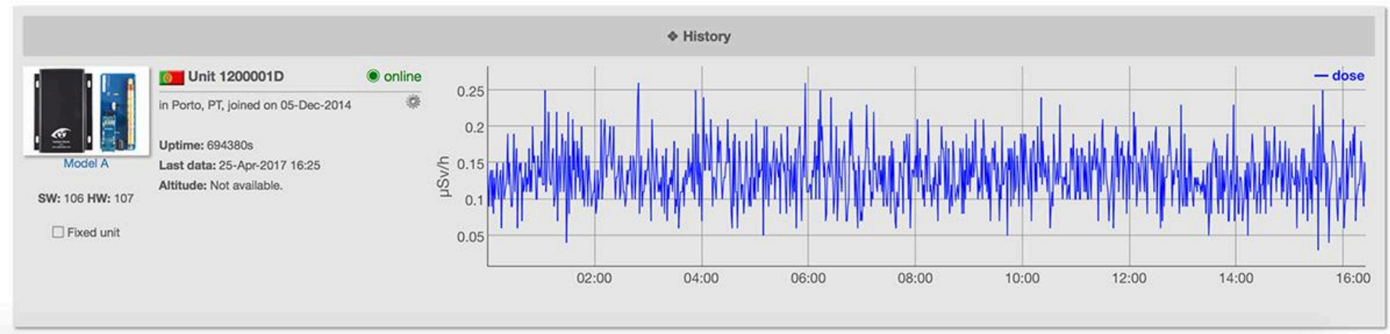


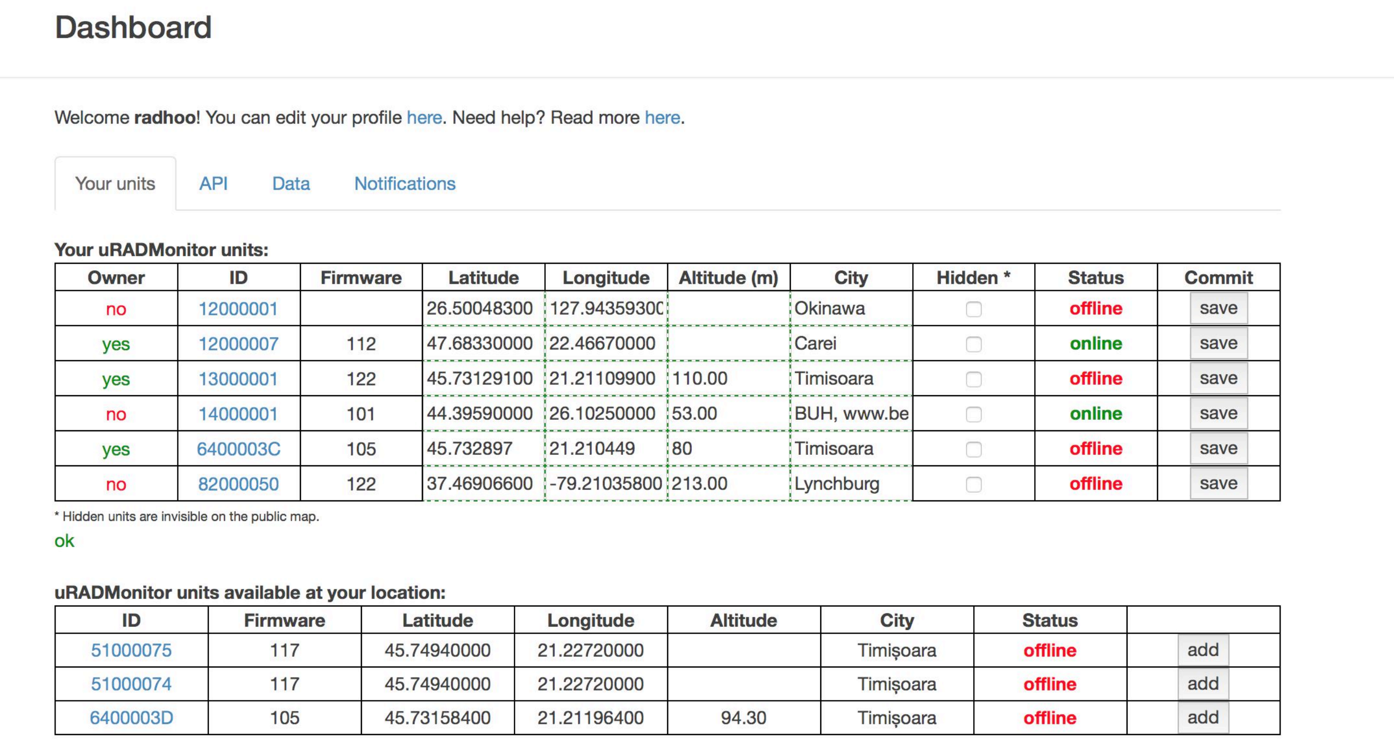
Figure 7: data chart

This part of the screen was also designed to pack more information, like type of hardware detector, version, time it joined the network, and more.

Like before, these charts support zooming by drag and drop, both on the horizontal and the vertical axis. This mechanism allows to zoom in on the time factor, or the actual value being measured, to analyze various pulses and trends in the data. To revert to default zoom level, simply double click the chart.

2.8 The dashboard

The Dashboard can be used to access raw data and measurements from the uRADMonitor cloud, both in CSV/JSON format, but also via the RESTful API as direct HTTP Calls. If you own a uRADMonitor hardware unit, you can use the Dashboard to configure your unit. Finally, the dashboard is the place to set notifications and alarms to be informed when a particular unit reaches a given threshold eg. excessive PM2.5 in your area to avoid any outdoor walks, to protect your health.



Dashboard

Welcome **radhoo**! You can edit your profile here. Need help? Read more here.

Your units | API | Data | Notifications

Your uRADMonitor units:

Owner	ID	Firmware	Latitude	Longitude	Altitude (m)	City	Hidden *	Status	Commit
no	12000001		26.50048300	127.9435930		Okinawa	☐	offline	save
yes	12000007	112	47.68330000	22.46670000		Carei	☐	online	save
yes	13000001	122	45.73129100	21.21109900	110.00	Timisoara	☐	offline	save
no	14000001	101	44.39590000	26.10250000	53.00	BUH, www.be	☐	online	save
yes	6400003C	105	45.732897	21.210449	80	Timisoara	☐	offline	save
no	82000050	122	37.46906600	-79.21035800	213.00	Lynchburg	☐	offline	save

* Hidden units are invisible on the public map.

ok

uRADMonitor units available at your location:

ID	Firmware	Latitude	Longitude	Altitude	City	Status	
51000075	117	45.74940000	21.22720000		Timișoara	offline	add
51000074	117	45.74940000	21.22720000		Timișoara	offline	add
6400003D	105	45.73158400	21.21196400	94.30	Timișoara	offline	add

Figure 8: dashboard main view

The dashboard is available on https://www.uradmonitor.com/dashboard , you will need to login before being able to use it.

Your units

This is the first screen of the dashboard, showing the available units. uRADMonitor hardware that is not yet assigned to your account, will appear here if it shares the same external IP with the computer you use to access the Dashboard. You can press Add to become owner of the respective unit. This mechanism was design to allow convenient control and configuration of your units. The external IP requirement is needed only the very first time because once assigned to your account, the units will be manageable from any other IP or location, assuming you can login to your account.
For the units already assigned, you will see them on the top side, marked with "Your uRADMonitor units". Here you can change the latitude, longitude, altitude, city name or mark the units as hidden. Hidden units will not appear on the global map, but you will still be able to access their data via the API calls. For any modifications to take place, press the "Save" button, an "ok" in green, under the list of units, will confirm operation was successful.

API

The second dashboard TAB offers information on API access. Most of that has been presented in this document already.

Dashboard

Welcome **radhoo**! You can edit your profile here. Need help? Read more here.

Your units | API | Data | Notifications

Authentification	API access
user-id : 1	units : 6
user-key: [illegible]	api-credit: 42120
user-ip : 84.232.160.78	global-access : 82000050,14000001,12000001

Get devices list
`//data.uradmonitor.com/api/v1/devices`
Credits used: ~~equals the number of units returned.~~ None in beta.

Get device sensors list
`//data.uradmonitor.com/api/v1/devices/[ID]`
Credits used: none

Get device data
`//data.uradmonitor.com/api/v1/devices/[ID]/[sensor]/[interval]`
Credits used: ~~equals the number of values returned.~~ None in beta.

Method: GET

Authentification: All calls must be authenticated with user-id and user-key sent in the headers section of the GET call as key-value pairs of the following format: `X-User-id:1, X-User-hash:`[illegible]
Response: JSON formatted answer, containing error message in case of failure or the requested data.

Note: Each API call consumes credit. Your credit balance must be positive to call any APIs. The APIs return data of your own units, unless your account is set for global access. Uploading data to the uRADMonitor network increases your credit. Data returned from the server cache is not counted.

Figure 9: dashboard API access details

The bottom of the page comes with examples to get you started integrating the uRADMonitor API to your custom apps.

Data tab

The third dashboard tab provides an interface to uRADMonitor datasets.

Select time interval
20 Jul 2017

Format
JSON CSV

Select unit and sensor

	ID	Firmware	City	Status	Sensor	Download
ok	12000001		Okinawa	offline	Temperature	go
ok	12000007	112	Carei	online	Temperature	go
ok	13000001	122	Timisoara	offline	Temperature	go
ok	14000001	101	BUH, www.beia.ro	online	Temperature	go
ok	6400003C	105	Timisoara	offline	Temperature	go
ok	82000050	122	Lynchburg	offline	Temperature	go

You can select the unit, the sensor and the time interval, as well as the output format: both CSV and JSON are supported. The downloaded data can then be used with your third party software for statistics, calculations or graphical representation.

Notifications

This is the last Dashboard tab and is used to configure automated notifications for any of your uRADMonitor units, in case a certain sensor returns values over a predefined threshold.

2.9 Open Layers 4.1.0

Not visible to the naked eye, but important from a programatic point of view and directly impacting performance, the new frontend was upgraded with the latest version of OpenLayers map library, for the data map representations.

3.0 Animations

Not to leave aesthetics aside, certain animation effects where added to the frontend, to make navigation more appealing, like when opening a unit by direct ID access or by clicking a node on the map, the map will zoom and pan to bring the unit in the center.