@City IoT Cloud Platform

iSys – Intelligent Systems IoT Solutions

@City

@AirQ @Bin

@Light @Metering @Trace

EN.iSys.PL

Table of Contents

1. Introduction.

@City IoT Cloud Platform is dedicated "micro-cloud" system for individual customers. Platform is not shareable and only one customer have access to physical or virtual server (VPS or dedicated servers). Customer may select one of dozens data-centers in Europe or in the world.

1.1 Supported device types.

@City IoT platform is dedicated to following iSys.PL products

- IoT - RF/LoRaWAN (Internet of Things)
- CIoT – GSM/2G/3G/4G/CATM1/NBIoT (Cell Internet of Things)
- WiFi
- Ethernet

1.2. Supported products types.

@City (eCity) Cloud IoT Platform is various size system for IP IoT products (called together as **@City Hardware** or **CioT Devices**):

- @City
- @Light
- @Metering
- @Trace
- @AirQ
- @Bin

1.3. Supported communication protocols

@City IoT platform supports following protocols for communication:

- UDP - suggested for CIoT sensors/devices (especially NBIoT) – lowest data utilization
- TCPIP - suggested for devices with bidirectional communication – handshaking/confirmation
- HTTP - suggested only for data access/visualization/export/ "cloud to cloud"
- HTTP Webhooks – for LoRaWAN communication between LoRaWAN Network/Application Server and @City Cloud.

1.4. Supported Communication Technology of the Devices

@City IoT platform supports:

- GSM: 2G, 3G, 4G (LTE), CATM1 (LTEM1), NBIoT – devices (UDP / TCPIP communication)
- LoRaWAN devices (HTTP Webhooks) – via LoRaWAN gateway and LoRaWAN network/application servers

- Ethernet Controllers (UDP / TCP communication)
- WiFi Controllers (UDP / TCP communication)
- Future IP products
- Non IP products via additional local @City/eHouse.PRO Hardware Gateway

@City IoT Platform is dedicated to devices/nodes:

- Native iSys products:
  - CIoT (GSM/2G/3G/4G/CATM1/NBIoT)
  - IoT (LoRaWAN)
  - WiFi - dedicated products for @City Platform
  - Ethernet - dedicated products for @City Platform
- Co-Production products (manufactured by third parties under iSys license and external brand)
- Franchise products (manufactured by third parties under iSys license with iSys's logos)

1.5. @City Cloud Server

@City software works on Linux based VPS (Virtual Private Server) or Dedicated Server on internet side, depending on requested performance of the **Server (called later Server):**

- private/public access
- overall devices count
- update of devices status frequency
- data refresh rate

Several variants of VPS exists depending on:

- Price
- data-center geo-localization
- Virtual processor cores (1-8)
- Virtual RAM (1-32GB)
- SSD disk (20GB-1TB)

Dozens of Dedicated server exists depending on:

- Price
- data-center geo-localization
- Processor cores (4 .. 32)
- RAM (16 .. 512GB)
- SSD/HDD (256GB .. 8TB)

The @City IoT platform is dedicated to single customer:

- local governments and authorities (City, Community, Country)
- B2B (for third party owner)

Because it is not shareable Server between customers, it simplify security access and performance issues. **Due to this reason only customer is responsible for effective security, stability, efficiency, data throughput, etc. In case of insufficient performance, customer may purchase higher plan (VPS or Dedicated Server), more optimal to expected functionality and performance.**
In special cases “Cloud to cloud” communication might be implement for globalization and centralization of data to bigger areas instead of multi-customer cloud.

1.5.1. Server and communication gateways

Communication of @City Server is realized based on low level application for performance maximalization.
Main features of @City Server application are:

- asynchronously receive data from devices (via protocols : UDP, TCP IP, HTTP)
- minimize data utilization and its costs between devices and Cloud (by using low level communication protocols UDP, TCP IP)
- receive encrypted status of devices (via any communication media)
- authorization and validating data from devices by decrypting their statuses
- decoding devices status and updating it into MariaDB/MySQL tables directly (in raw data format)
  - current data table (contains only newest status of each devices)
  - historical data table (contains all statuses for single device)
- send pending commands (Events) to devices
- assuring handshaking, verification and confirmations

@City Server software is the same for each user and can't be customized for different customers.

1.5.2 HTTP LoRaWAN integration

LoRaWAN controllers are integrated with the @City cloud via the HTTP interface (webhooks) available on the LoRaWAN network/application server.

Several types of network/application server are supported:

TTN (limited time "On The Air" and the maximum number of commands sent to the driver and do not support firmware upgrade)

LoraWAN-Stack (Requires hosting on a physical device with internet access).

LoraServer.Io (Requires hosting on a physical device with internet access - only sending data to the server and do not support firmware upgrade)

The @City Cloud for LoRaWAN controllers is divided in the same way as for other interfaces. It is discussed in the previous chapter.

1.5.3. Front-end interface

Front-end interface is realized with PHP scripts for extracting customized data from @City Cloud

Database. It uses very elastic search mechanism, based on original SQL queries to limit desired data. Interface supplies query results in JSON format for further decoding and processing by JavaScript Front-end Web 'application'.
Original front-end interface is the same for each user and can't be customized for different customers. Overlay interface may be created by our staff or in the cooperation to assure customization for the customer.

1.5.3. Server access rights

Customer access rights (to physical **Server)** are limited.
File access for 'templates' directory only (native text files – .txt, .js, .css, .html) :

- adding, removing, modifying pure HTML files (Front-End GUI/Interface Development)
- adding, removing, modifying pure JavaScript files (Front-End GUI/Interface Development and Algorithms)
- adding, removing, modifying pure CSS Files (Front-End – Custom views / Themes)
- adding, removing, modifying text files of templates (Front-End GUI)
- adding, removing, modifying tabs, shortcuts, links to data results

Other Access rights:

- Full access to @City Cloud Database MySQL/MariaDB where all devices data are stored
- Access to Web Services are defined (credentials) for customer.
- Customer is not allowed to pass credentials to third parties (multiple access of results might affect performance, stability and security of overall system)
- Under special circumstances and usage of high performance Dedicated Server, public account might be set up to observe most current data (not historical).
- User may duplicate data to his own MySQL server and perform own data analyze and processing, in order don't affect performance of @City Server

iSys – Intelligent Systems staff – have unlimited access to whole server including root account and full DB access for maintenance.
Under certain circumstances iSys might grant additional limited rights to customer (PHP scripts, files) after checking source code, running tests, if it not affect overall system security, stability and performance.

1.6. Smart Devices

1.6.1. CIoT - GSM devices

Our devices contains microcontroller and GSM/GPS/GNSS module (2G..4G, NBIoT, CATM1) for communication. Microcontroller contains encrypted bootloader for secure OTA firmware upgrade. This enables creating many system variants based on the same “CIoT Smart device”.

1.6.3. BAS, BMS, IoT – Ethernet and WiFi devices

Ethernet and WiFi controllers allows IP based communication to the system (without charging for data transfer to the GSM operator). This devices have also encrypted bootloader and devices might be updated via its native interface. For WiFi it have OTA firmware upgrade from main server

1.6.2. IoT - LoRaWAN devices

LoRaWAN enables data transmission over very long distances (up to approx. 15km). This range depends on the speed of data transmission, the amount of data, urbanization of the area and the efficiency of the radio paths of the devices.

Our devices include a microcontroller and LoRaWAN module for communication. The microcontroller contains an encrypted bootloader for secure OTA software update. This allows you to create multiple system variants based on the same "IoT smart device". The devices operate in the ISM open band without additional subscription fees. It is necessary to use LoRaWAN Gateways to cover the whole area with access to the Internet. In the case of existing LoRaWAN gates within the range of devices (configured for TTN server), it is possible to send information through them. Firmware upgrade require own network/application LoRaWAN server and good range for communication.

1.7. Business to Business (B2B) options

There are several options for business and cooperation:

- import/export ready products (switchboards, devices) – final products
- import/export OEM products (PCBs, controllers, etc) – intermediate products, spare parts
- franchise – production based on our license for local markets (we supply only programmed microcontrollers) under iSys – Intelligent Systems Brands
- Co-Production – (as above) but under external Brands

2. @City IoT Platform Functionality

@City platform supports customizable Front-End template for data visualization, query, limiting and processing (Current/history data):

- visualize selected data on the map (geo-localization)
- shows selected data and results in the tables
- shows selected data in bar charts
- shows selected data in historical charts
- export data to third party applications and further processing
- other functionality will be updated on individual requests.

User's Front-End is accessible via static IP or DNS redirection domain/subdomain/file if available.

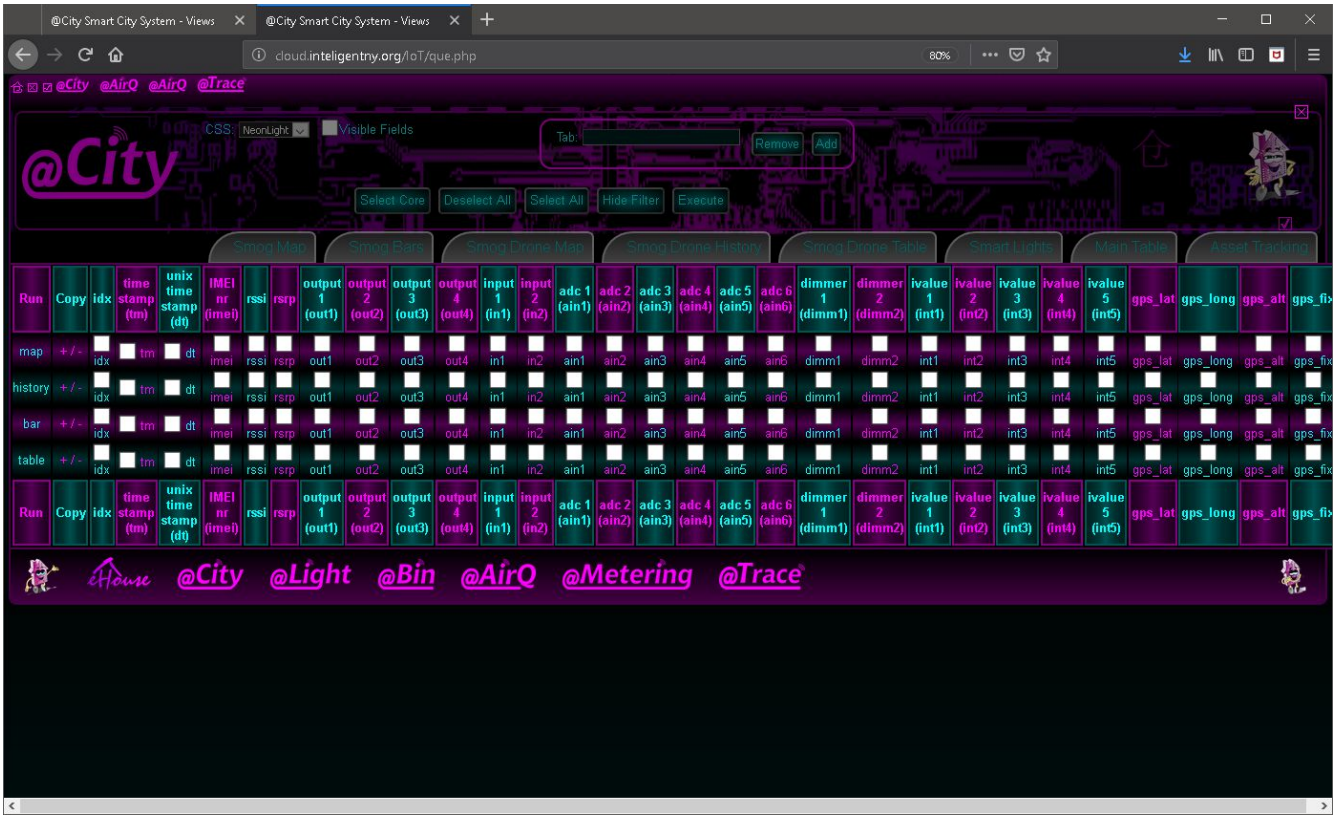
Exemplary & Demo installation (It is enabled only for prospective customers).
Please inform us when you want to test it – to enable public access to the platform.
It could require static IP of remote computer to enable communication to @City platform.

3. Main Page

Main page is left empty intentionally for security reasons: **http://%YourIP%/IoT/**
It might be individually enabled and edited and contain links to all available services of **@City IoT Platform** if it is required

4. Main Form

Main Form is intended to create new presets and tabs: http://iot.inteligentny.org/IoT/que.php
This is initial form for creating results, views and tabs for each configuration



Descriptions (From top and left to right direction)

4.1. Header

4.1.1. Home Link - (opens actual results table)

4.1.2. 'X' checkbox - opens/closes Query Form

4.1.3. 'V' checkbox - opens/closes Fields Form

4.1.4. Graphical icons - links to visualization results (editable)

4.2. Form:

4.2.1. 'X' checkbox - opens/closes whole Query Form

4.2.2. CSS - Select Visualization Theme

Modify Visualization Theme CSS file must exist in “templates/css/” directory - listed automatically.

4.2.3.Visible Fields checkbox – shows/hides Field Filter List

4.2.4. Tab: Tab Name to add or remove

4.2.5. Add / Remove Buttons - Add or remove tabs with the name in Tab field

4.2.6. Select Core Button

Select main fields visible on the table. It is updated automatically.

4.2.7. Deselect All Button

Deselect all fields (must be followed by selecting some of them manually)

4.2.7. Select All Button

Select all fields (must be followed with deselect some of them manually)

4.2.8. Hide Filter – Hide whole Form

This is equivalent of all (X) checkbox

4.2.9. Execute Button – Change parameters settings

4.2.10. 'V' checkbox – show/high filter fields.

4.3. Tabs

Individually created tabs with names and presets (stored in **cfg/tabs.cfg** file).
The file actually contains name and URL (separated by tab char).

4.4. Table Contents

Displays all fields limited by Field Filter.

Fields in the table:

4.4.1. Run – views result type

map – mapping results on the map (one or more field may be selected)
history – historical charts (one or more field may be selected)
tab – displays table (any combination of fields may be selected)
bar – only one field is displayed on the bar chart
On pressing one of its value it will open new results with selected fields (for current row).

4.4.2. Copy (+/- links)

Adding/removing a Tab with the name set in **Tab** field. It use only fields selected in the same row of the table.

4.4.3. Table Cell Links

Pressing any other field name will initiate Data Visualization of selected field for selected row.

4.5. Data Order

Order of displayed fields are as its order in fields form (however **tm** field is always send to the end of text). This order can only be changed with direct editing of URL parameters (fields order part).

4.6. Example

For example: Setting Tab with **Asset Tracking** name and contains map with time and speed on the map

All description referring to row where "**Map**" text is in "Run" column.

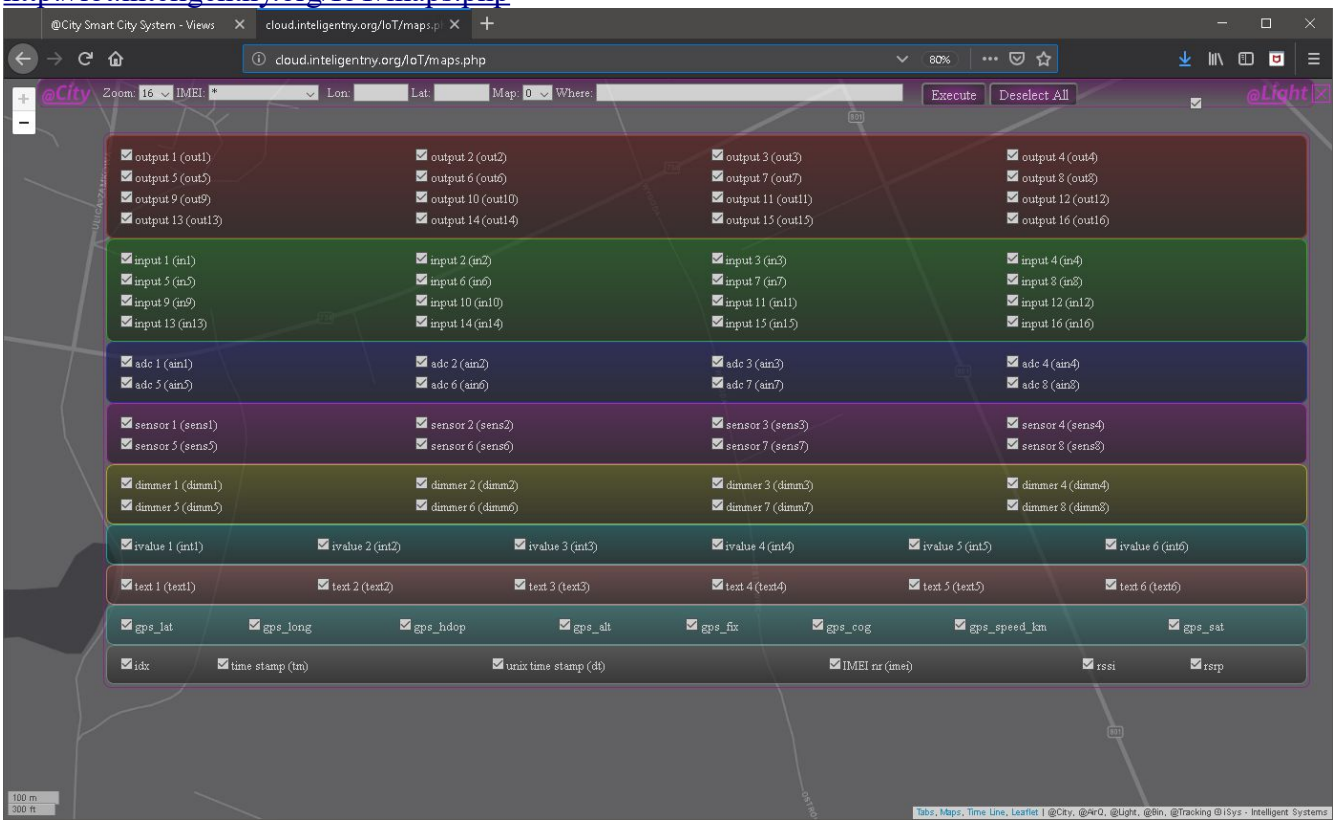
1) Enter name "**Asset Tracking**" in **Tab** field (without quotation marks)
2) Ensure all columns are unselected in the row
3) select **tm**, **gps_speed_km** only in the row
4) press + button where in the row

5. Maps

Maps can be launched from MainForm with pre-configuration

5.1. Map Initialization

Map initialization is performed manually when executed directly with link:
http://iot.inteligentny.org/IoT/maps.php



1. User should Deselect all fields (Press **Deselect** Button)
2. Press some checkbox for displayed fields (eg. **Ain5** (for Smog level) and **tm** (for measurement date/time)
3. press **'V'** checkbox to hide fields form
4. press **Execute** button to run DB query and display current information from all sensors/devices
5. Map with data is updated after 30 seconds or more.

5.2. Optional Settings for query

Settings described from left to right (on above screenshot).

5.2.1. Modify MAP scale (Zoom Level)

1) Zoom level might be modified using (+/-) buttons for scale (current_scale*2 or current_scale/2 respectively) . Pressing one of this buttons will automatically modify scale.
2) Another way is select Zoom Level in **Zoom** Combo Box field and press **Execute** button. In this case whole View/Map is reload and refreshed (takes a while during initialization).

5.2.2. IMEI (Select Device Field)

IMEI field contains device unique ID or Unique alias for a device. Default setting is * (asterisk) which shows most recent values and geolocation for each device.
Setting IMEI to any other value, will show historical data of selected device. It have sense only for mobile and moving sensors, otherwise results will overlap on the map at the same position.

5.2.3. Lon, Lat (Longitude, Latitude coordinate fields)

Set center position of the map. This field is set to cursor position when mouse button is pressed on the map.

5.2.4. Modify MAP Style (Theme)

Map style/theme can be selected from **Map** ComboBox field (eg. Dark, Grey, Topographic) .
Various map themes might have different maximal zoom levels so it might enforce proper Theme to increase map scale.

5.2.5. WHERE Clause

Where Clause is used for additional query string {WHERE part} for MySQL/MariaDB.
This clause is taken into account for construct complete QUERY string for database result. It may limit data, time and any other values by limiting results count. Original table field names (not alias) must be used in this field. Eg.

1. gps_speed_km>10 //speed is more than 10km/h
2. ain5>3 //ain5 is greater than 3 (holding 2.5um particles count – smog level)
3. gps_speed_km>10 and ain6>5 //speed is more than 10km/h and ain6 is greater than 5 (holding 10um particles count – smog level)

5.2.6. Execute (Run Query Button)

Pressing this button is required to change any settings, parameters (except pressing +/- buttons).
Map is loaded from the beginning with new presets.
Map is not loaded at all, when no data is available for current query.

5.2.7. **Deselect All** (Remove all fields from query)

After pressing this button at least one field must be selected manually to display results on the map.

5.2.8. **'V'** Checkbox (Open/Close Field Form)

This checkbox is used to show/hide selector of fields to display.

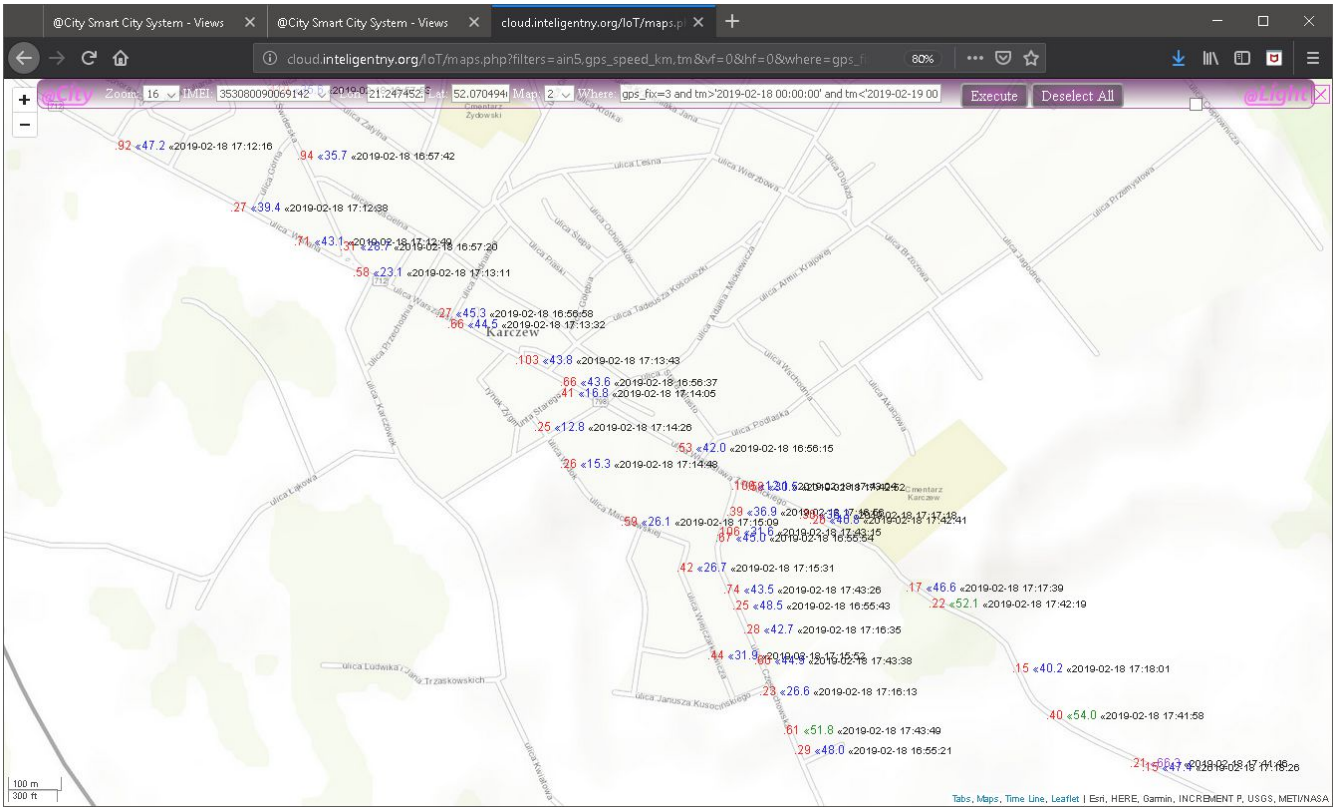
5.2.9. **'X'** Checkbox (Show/Hide Query Form)

This checkbox enables hide whole Form except (+/- buttons)

The results on the map are continuously refreshed and updated with new values

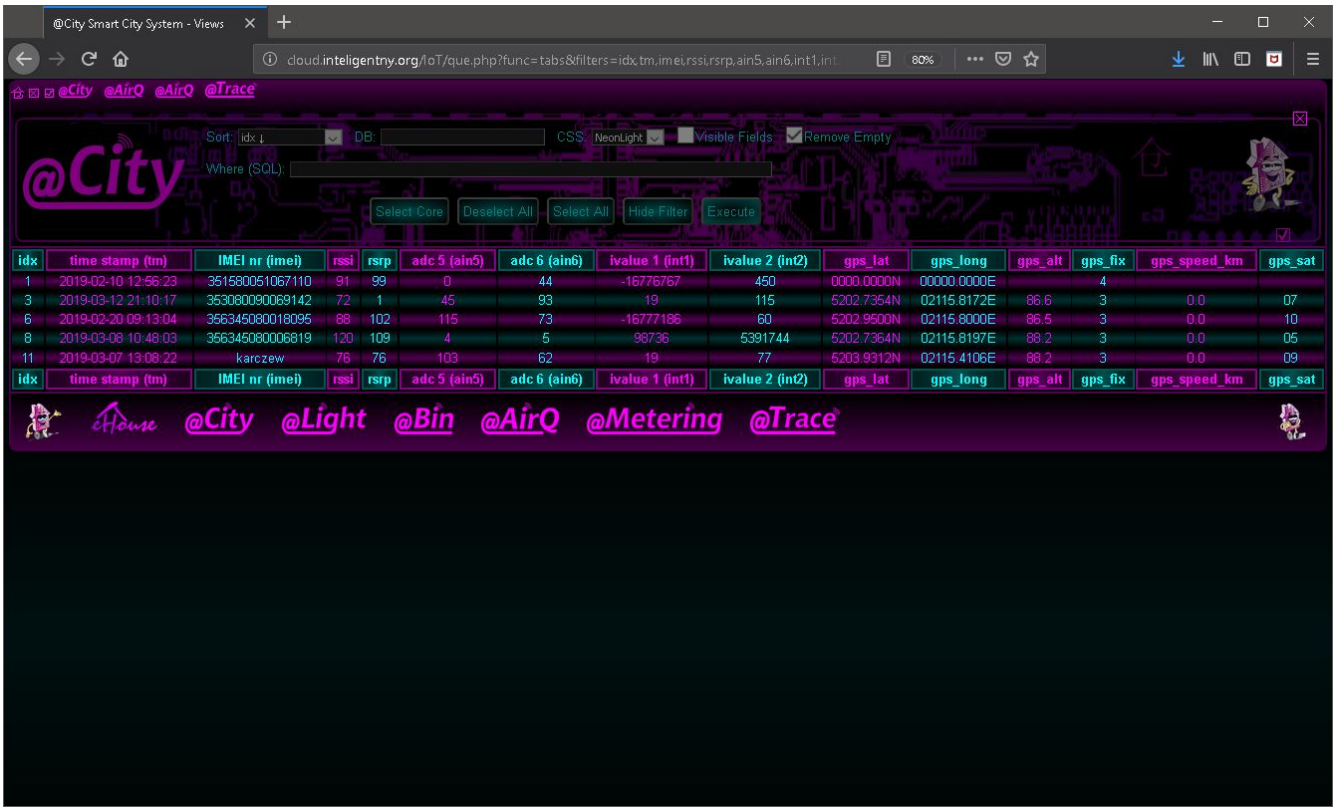
5.3. Example

Eg Smog results (Sensor installed on the car): Smog level 2.5um particles (Ain5), Speed (gps_speed_km), Date/Time (tm), map (2 - topographic), zoom level 16,
Where clause:
"**gps_fix=3 and tm>'2019-02-18 00:00:00' and tm<'2019-02-19 00:00:00' and gps_speed_km>0**".
//GPS= valid 3D results & date=2019-02-18 & speed > 0 km/h



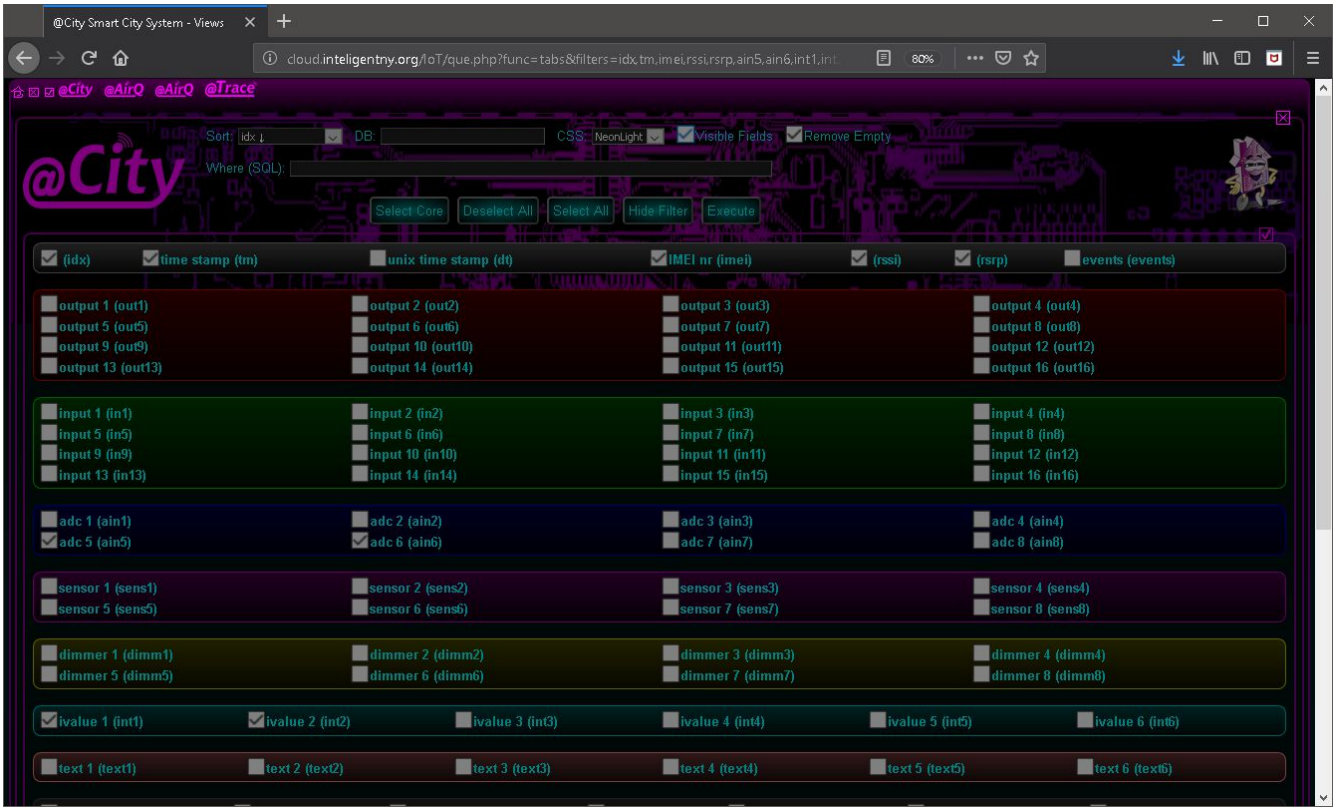
6. Show Results in the Table

Show results in the table.
On "**Main Form**" press "table" item, after selecting some fields to display pre-configured table



6.1. Initialization of table

When table is open from link http://iot.inteligentny.org/IoT/que.php?func=tabs it requires prior initialization of settings.
You can select visible fields (by pressing "Visible Fields") checkbox.



1. Press all required checkbox for displayed fields
2. Press checkbox “Visible Fields” to hide fields form
3. Press Execute button to run DB query and display table

6.2. Optional Settings for query

Settings are described from left to right (on the screenshot).

6.2.1. Sort - sort field and order ascending/descending

Sort field is equivalent of pressing column header.

6.2.2. DB / IMEI - Select Device

IMEI field contains device unique ID or Unique alias for a device. With empty value it shows table of most recent values.
Setting IMEI to any other value, will show historical data of selected device.

6.2.3. CSS – select style (Visualization Theme)

6.2.4. Visible Fields – Show/Hide Fields Form

6.2.5. Remove Empty – Don't display empty columns

6.2.6. 'X' Checkbox (Show/Hide Query Form)

6.2.7. Where Clause (for data limitation)

This is sufix for MySQL/MariaDB additional query string {WHERE part}
This clause is taken into account to construct complete QUERY string for database result. It may limit data, time and any other values by limiting results count. Original table field names (not alias) must be used in this field. Eg.

1. gps_speed_km>10 //speed is more than 10km/h
2. ain5>3 //ain5 is greater than 3 (holding 2.5um particles count – smog level)
3. gps_speed_km>10 and ain6>5 //speed is more than 10km/h and ain6 is greater than 5 (holding 10um particles count – smog level)

6.2.8. Select Core Button (Enable most common fields)

6.2.9. Deselect All Button (Remove all fields from query)

After pressing this button at least one field must be selected manually to display results on the map.

6.2.10. Execute (Run Query Button)

Pressing this button is required to change any settings, parameters (except pressing +/- buttons).
Table is reloaded from the beginning with new presets.

6.2.11. 'V' Checkbox (Open/Close Field Form)

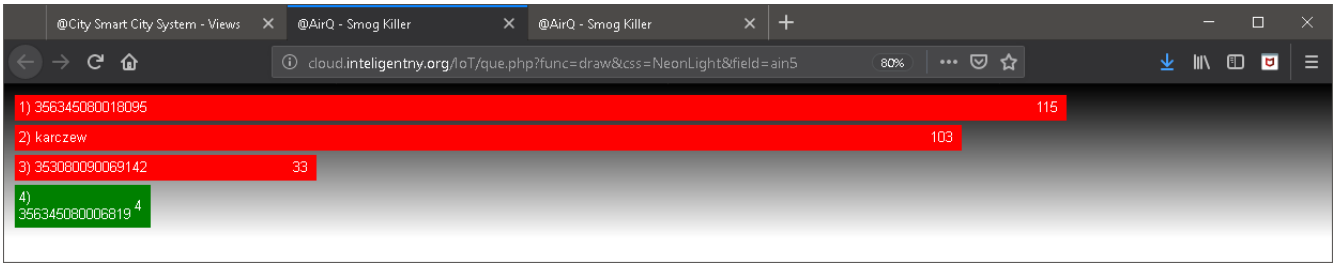
This checkbox is used to show/hide selector of fields to display.

Results in the table are sorted according to **Sort** field setting. Sorting order can be changed by pressing row header (once for one direction twice for another direction).
Some results in columns link to further visualization screens (hard-coded).

When displaying historical data for device it should be limited in order not to display whole history information because it may lead to performance or out of memory issues.

7. Bar Charts.

Bar charts should be executed from Main Form by pressing single field in 'Bar' row.
It displays sorted bars normalized to maximal value, showing from highest to lowest order.
It is useful for fast checking extreme results and take some actions.



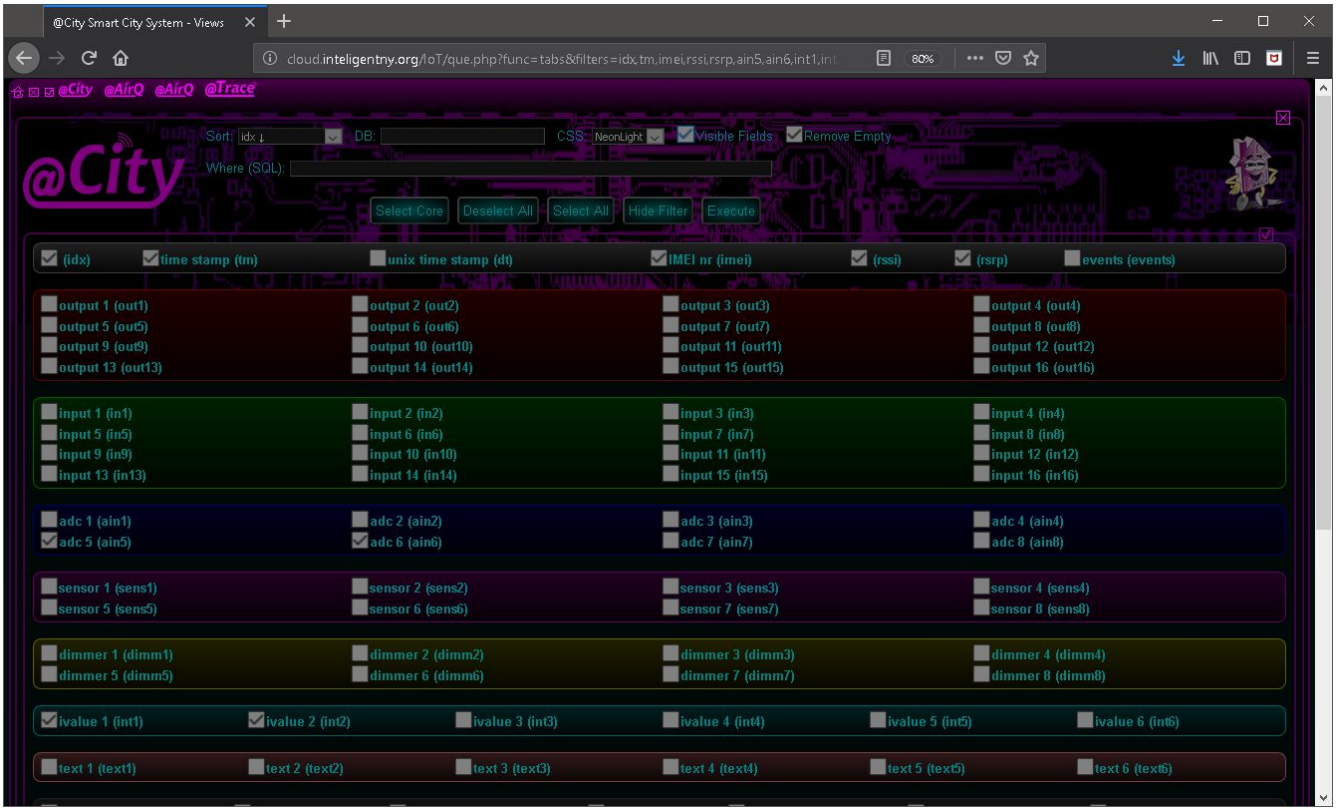
Mouse Over event will display additional information for the device.

8. Historical Charts.

Historical charts can be initiated from the MainForm when pressing selected column in 'History' row (for single field).
For Multiple fields in 'History' row desired fields must be checked and 'History' link must be pressed in 'Run' column.
Historical results is limited to last 24 hours + next 24 hours (for eventual refreshing charts), when no limits were set up.

8.1. Initialization of Historical charts

Historical charts when opened from main link require initialization as other results, when open from link without preferences parameters.
Multiple fields may be selected to display various items. It can also be set in Field Filter Form.



1. Press all required checkbox for displayed fields
2. Press checkbox “Visible Fields” to hide fields form
3. Press Execute button to run DB query and display the table

8.2. Optional Settings of Historical Charts

Items described from top and from left to right (on the screenshot).

8.2.1. **IMEI** - (Select Device to display historical data)

IMEI field contains device unique ID or Unique alias for a device. With * (asterix) value it shows table of most recent values which has no sense.
Setting IMEI to any other value, will show historical data of selected device.

8.2.2. **Min -** limit minimal value of first field

8.2.3. **Max -** limit maximal value of first field

8.2.4. **'V'** – Show/Hide Fields Form

8.2.5. **From**: set minimal date/time (*)

8.2.6. **To**: set maximal date date/time (*)

8.2.7. '**X**' Checkbox (Show/Hide Query Form)

8.2.8. **“Where”** Clause

Clause for limiting data results MySQL/MariaDB additional query string {WHERE part}.
This clause is taken into account for construct complete QUERY string for database result. It may limit data, time and any other values by limiting results count. Original table field names (not alias) must be used in this field and valid SQL syntax. Eg.

1. gps_speed_km>10 //speed is more than 10km/h
2. ain5>3 //ain5 is greater than 3 (holding 2.5um particles count – smog level)
3. gps_speed_km>10 and ain6>5 //speed is more than 10km/h and ain6 is greater than 5 (holding 10um particles count – smog level)

8.2.9. **Deselect All** Button (Remove all fields from query)

After pressing this button at least one field must be selected manually to display historical results.

8.2.10. **Execute** (Run Query Button)

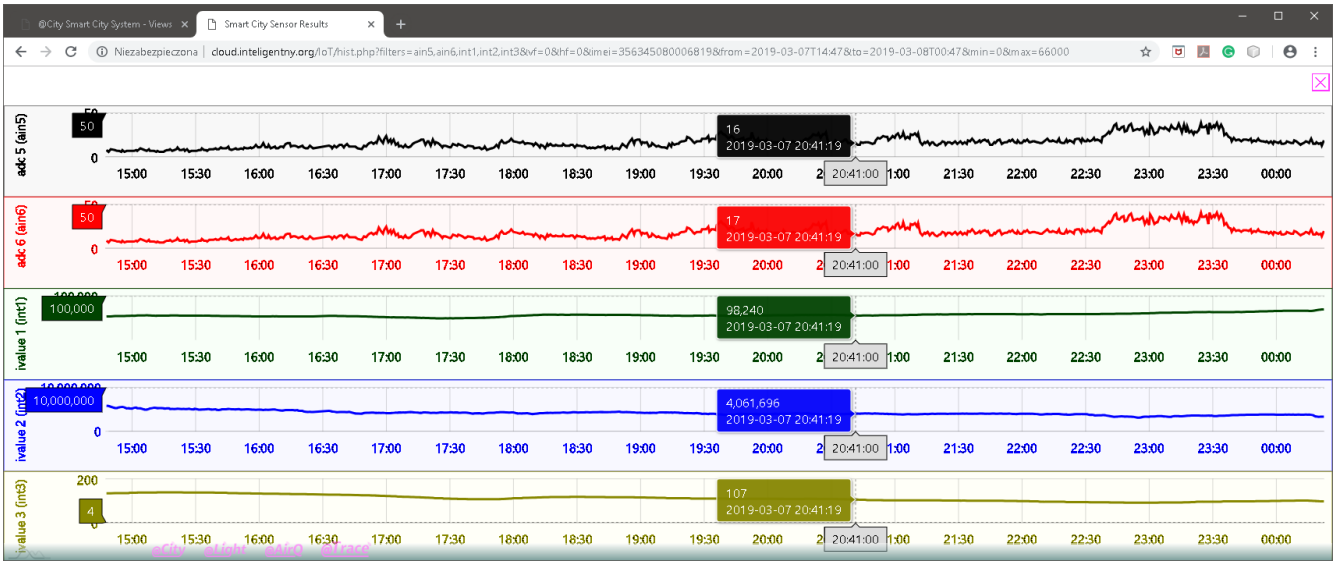
Pressing this button is required to change any settings, parameters (except showing fields or query panel). Table is reloaded from the beginning with new presets.

8.2.11. 'V' Checkbox (Open/Close Field Form)

This checkbox is used to show/hide selector of fields to display.

*8.3. **Bars** Variant: (displays only available data)*

*8.4. **Continuous** variant (with the same data):*



Mouse pointer display values of measurements and date/time.

9. Web browser Compatibility

Function / WWW Browser	Chrome 72	FireFox 65	Edge	O
Maps	+	+	+	+
Historical	+	+ (*)	+	+
Bars	+	+	+	+
Tabs	+	+	+	+

* - Firefox do not support date/time picker (text field must be manually edited using proper date time format).
Internet Explorer is unsupported (use **Edge** instead)
Other web browsers were not tested.

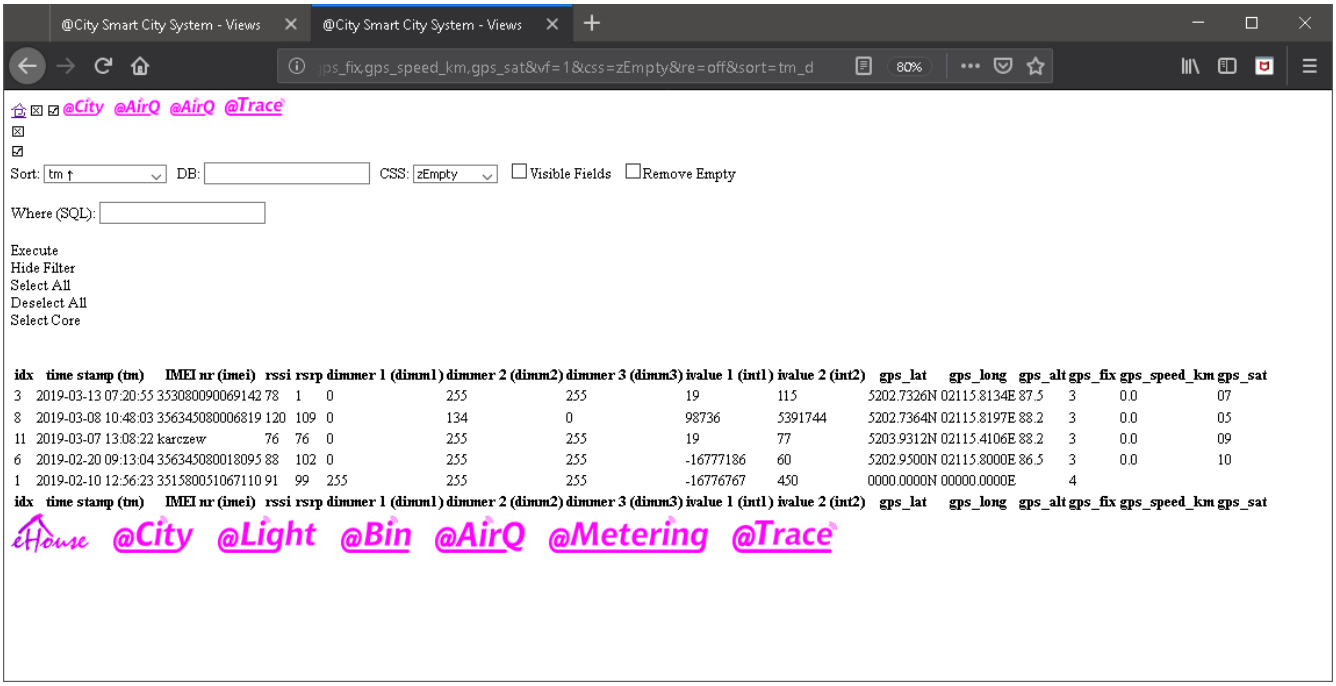
10. Themes Customization

Web pages are based on general template file located at '**templates**' directory '*.template'. Additionally each page type contains:

1) '*.head' file which stores header of the page (links, imported CSS, JavaScript Files, etc.)
2) '*.foot' files which stores footer of the page (links, etc.)

Visualization Theme can be changed according to user preferences by coping and modifying CSS files. CSS files are located in '**templates/css**' directory. Different Web Page Themes might be used to create optimized for eg. printing , SmartPhones, PADs templates.

Table views - have selectable field for choosing CSS file for complete modification of theme (stored in '**templates/css/tabs**' directory).



Map views **-** general theme is selected by '**map**' type combo box. Additionally there is default CSS file '**templates/css/map.css**' which contains some additional functionality like hiding/coloring results based on its values.The rest of this CSS file is practically limited to query and field forms.

Most of **@City Platform** PHP files for visualization accept **css** parameter with value of file name for the Theme (without extension). File must be located in 'templates/css' directory and the name is case sensitive.

Some elements of Theme display is located directly in JavaScript file located in '**template/js**' directory. Main **@City** script '**@City.js**' is located in upper directory. There is no modification possibility in this

location, however script may be copied to '**templates/js**' directory and modified there. Usage of individual script requires update all header files.

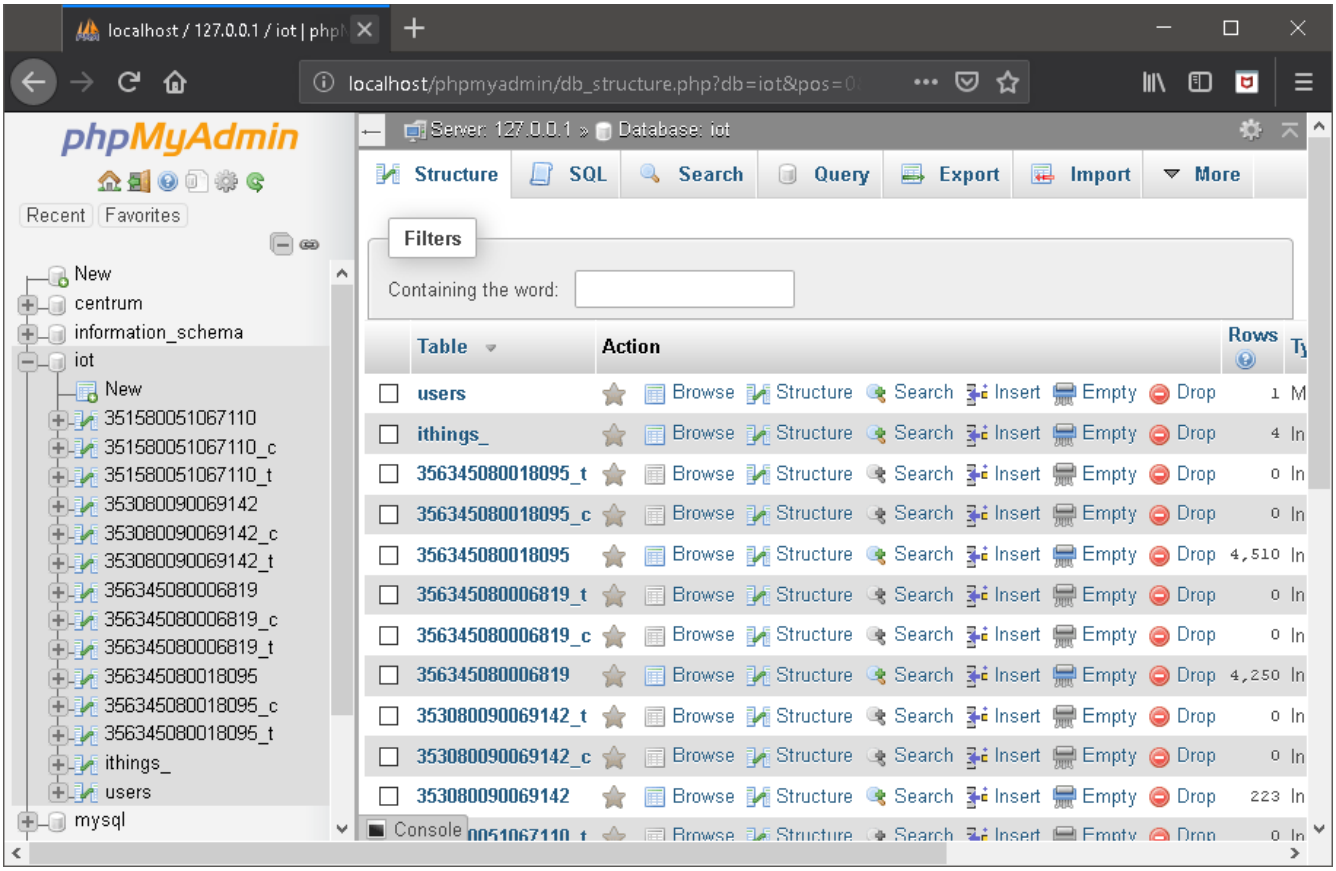
11. Algorithms Update

Some unique sensors may require dedicated calculation functions.
There is no possibility to update and maintain multiple variants of **@City Server Software**, **Front-end PHP interface**, which would cause a lot of issues, versions, errors.
Best and easiest way to achieve it, is updating JavaScript 'overlay' files for proper display of the value/description.
Original JS scripts are open text file and might be adopted to customer needs. As stated in previous chapter they have to be copied to '**templates/js**' directory where customer have access rights for modification.

Technical aspect on programming of **@City** system is not a subject of this document, however Web developer with basic knowledge of HTML and JS may customize Front-end Web application to individual customer needs.

12. Database Structure

@City Database with a name '**IoT**' or '***IoT**' is divided in tables (where asterix is prefix depending on hosting server – if required). DataBase might be observed in PHPAdmin (web application) at link http://iot.inteligentny.org/phpmyadmin



Tables Set for each Device (where * {asterix} is IMEI address – unique ID):

- '***_t**' – tokens for current users authorization (not used for Single-Customer clouds)
- '***_c**' – commands (Events) queue - to be sent to the device
- '*' – all decoded status results.

Other tables:

- '**ithings_**' table - contains decoded current status of all devices (it is copied during update any of '*' tables). Structure of '**ithings_**' table is almost the same to the '*' tables. '**ithings_**' have additional fields for Postal Address and descriptions.
- 'users' table is restricted and should be modified.

12.1. 'ithings_' and '*' tables structure

- **`idx`** - index
- **`tm`** - automatic timestamp
- **`dt`** - unix timestamp
- **`imei`** - unique device address
- **`rssi`** - RSSI signal level
- **`rsrp`** - RSRP signal
- **`events`** - waiting events to be sent to controller
- **`out1`** .. **`out16`** - decoded outputs status
- **`in1`** .. **`in16`** - decoded inputs status
- **`ain1`** .. **`ain8`** - decoded ADC values (RAW)
- **`sens1`** .. **`sens8`** - converted sensor values (depends on device type)
- **`dimm1`** .. **`dimm8`** - decoded dimmers values (depends on device type)
- **`int1`** .. **`int6`** - decoded counters values (depends on device type)
- **`text1`** .. **`text6`** - decoded text values (depends on device type)
- **`creation`** - device creation date/time
- **`last`** - last date/time
- `user` - future use
- `pass` - future use
- `sn` - gsm serial nr
- **`status`** - current controller status not decoded in hex format
- `hash_code` - future use
- `addr` - short device address
- `fwnr` - firmware nr
- `disabled` - disabled device (not performed)
- `gsm_nr` - CIoT gsm number
- `vendor` - vendor code (for franchise/co-production)
- `timezone` - Time Zone offset
- `dst` - Use Daily savings settings
- **`gps_lat`** - GPS Latitude
- **`gps_long`** - GPS Longitude
- **`gps_hdop`** - GPS HDOP
- **`gps_alt`** - GPS Altitude
- **`gps_fix`** - GPS Fixation (must be 3 – for valid results)
- **`gps_cog`** - GPS Cog
- **`gps_speed_km`** - GPS Speed in [km/h]
- **`gps_sat`** - GPS satellites found
- `continent` ,`country` , `region`, `subregion`, `subsubregion` , `city` , `district`, `street`, `street_nr` , `item_nr` - **Customer address and description fields (!!!!Not available for '*' historical tables)**

- `log` - log data

  The fields names are important for creating SQL queries for increasing search functionality because original name must be used in SQL Statement.

*12.2. Device commands (Events) queue '*_c' table – structure*

This table is event/commands queue for each device and have following structure:

- `command` - command to be send or already sent to the controller
- `confirmed` - confirmation flag if it was already sent and confirmed
- `date` - unix timestamp of event
- `updated` - auto update flag (timestamp date/time)

12.3. Accessing results from databases - Mid-Level (Reading Data)

Data can be accessible without Front-end Web application. @City system contains script with mid-level functions. Results are returned in JSON format.

12.3.1. Get current statuses of all devices

http://%IP%/IoT/que.php?func=devsjson

Query returns whole '**_ithings**' table (current statuses of all devices) in JSON Format:


```
[ { "country":"", "city":"", "continent":"", "country":"", "region":"",
"subregion":"", "subsubregion":"", "city":"", "district":"", "street":"",
"street_nr":"", "item_nr":"", "gps_lat":"0000.0000N", "gps_long":"00000.0000E",
"tm":"2019-02-10 12:56:23", "creation":"2019-02-09 18:12:38", "last":"0000-00-00
00:00:00", "events":"", "user":"", "pass":"", "imei":"351580051067110",
"sn":"",
"status":"73000200000f360033026800240000002c002c002dffffffffffffffff5b63000001c1000
001c2000000000000000009250a4f0a760a7a0a750a780a7e0000031d032205fc34029b025c02560046
0eb305320000", "hash_code":"", "addr":"", "fwnr":"", "disabled":"",
"gsm_nr":"", "vendor":"", "timezone":"", "dst":"", "rssi":"91", "rsrp":"99",
"gps_lat":"0000.0000N", "gps_long":"00000.0000E", "gps_hdop":"", "gps_alt":"",
"gps_fix":"4", "gps_cog":"", "gps_speed_km":"", "gps_sat":"", "events":"",
"out1":"0", "out2":"0", "out3":"0", "out4":"0", "out5":"0", "out6":"0",
"out7":"0", "out8":"0", "out9":"0", "out10":"1", "out11":"0", "out12":"0",
"out13":"0", "out14":"0", "out15":"0", "out16":"0", "in1":"0", "in2":"0",
"in3":"0", "in4":"0", "in5":"0", "in6":"0", "in7":"0", "in8":"0", "in9":"0",
"in10":"0", "in11":"0", "in12":"0", "in13":"0", "in14":"0", "in15":"0",
"in16":"0", "ain1":"3894", "ain2":"51", "ain3":"616", "ain4":"36", "ain5":"0",
"ain6":"44", "ain7":"44", "ain8":"45", "sens1":"0", "sens2":"0", "sens3":"0",
"sens4":"0", "sens5":"0", "sens6":"0", "sens7":"0", "sens8":"0",
```

"dimm1":"255", "dimm2":"255", "dimm3":"255", "dimm4":"255", "dimm5":"255", "dimm6":"255", "dimm7":"255", "dimm8":"255", "int1":"-16776767", "int2":"450", "int3":"", "int4":"", "int5":"", "int6":"0", "text1":"", "text2":"", "text3":"", "text4":"", "text5":"", "text6":"" }]

12.3.2. Get Historical data for the Device

Query historical data of single device by IMEI nr:
http://%IP%/IoT/que.php?func=imeijson&imei=356345080018095

Because whole table might contain millions of rows it should be limited with WHERE clause in order not to hang-up server.
Additional parameters url parameters:
func – imeijson
imei – IMEI of device
field – fields to be displayed in the results (coma separated list)
min – minimum value for the first field from the list
max – maximum value for the first field from the list
sort – field for sort
tm – field is automatically added to the results.
where – where clause to limt data

Example:
We want get following result
for device with **imei=356345080018095**
show fields: **ain5, ain6, gps_lat, gps_long**
and limit **ain5** in range **(1, 10000)** - must be first field in list
and **gps** have valid data **(gps_fix=3)**
and date/time **(tm)** from **2019-02-14 23:00:19** to **2019-02-15 00:00:00**

Constructed URL string:
http://iot.inteligentny.org/IoT/que.php?
func=**imeijson**&imei=**356345080018095**&field=**ain5,ain6,gps_lat,gps_long**&min=**1**&max=**1000**&where=**gps_fix=3 and tm>'2019-02-14 23:00:19' and tm<'2019-02-15 00:00:00'**

Query Results:

[{ "ain5":"66","ain6":"68","gps_lat":"5202.7326N","gps_long":"02115.8073E","tm":"2019-02-14 23:04:31" },
{ "ain5":"67","ain6":"76","gps_lat":"5202.7328N","gps_long":"02115.8075E","tm":"2019-02-14 23:05:42" },
{ "ain5":"63","ain6":"77","gps_lat":"5202.7328N","gps_long":"02115.8074E","tm":"2019-02-14 23:06:05" },
{ "ain5":"58","ain6":"77","gps_lat":"5202.7328N","gps_long":"02115.8075E","tm":"2019-02-14 23:06:32" },
{ "ain5":"58","ain6":"68","gps_lat":"5202.7328N","gps_long":"02115.8076E","tm":"2019-02-14 23:06:55" }]

12.3.3. Get list of devices - single field from current statuses with limitation

This function returns limited data from '_ithings' table

http://cloud.inteligentny.org/IoT/que.php?func=fieldjson&field=ain5&min=13&max=5000

Parameters:

func – fieldjson
field – field to be displayed in the results - **imei** and **tm** are automatically added
min – minimum value for the field
max – maximum value for the field

For above query string it returns results of **ain5, imei, tm** fields:
if **ain5** is in range **(13,5000)**

Query Results:


```
[ {"imei":"353080090069142", "tm":"2019-03-14 11:51:01", "ain5":"14" },
  {"imei":"356345080018095", "tm":"2019-02-20 09:13:04", "ain5":"115" },
  {"imei":"karczew", "tm":"2019-03-07 13:08:22", "ain5":"103" } ]
```