Water -M Project

D 4.2 Report of System Integrations and Pilots V1.2

**History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version & Status | Author | Modifications |
| 21/11/2017 | V1.0 | Eero Antikainen (Savonia) | A first document |
| 23/11/2017 | V1.01 | Teemu Räsänen (Savonia) | Structure of a document |
| 30/11/2017 | V1.02 | Jean-Hugues Lauret  (Cityzen data) | Water Indoor farming pilots and integration |
| 12/12/2017 | V1.03 | Ali KAFALI (ACD) | Waste water integration and Pilots |
| 12/12/2017  12/12/2017 | V1.04  V1.05 | Güven Köse(MANTIS)  Aydın Can POLATKAN(MANTIS) | Waste water integration and Pilots |
| 18/12/2017 | V1.06 | George Suciu (BEIA) | Contributions for Romanian use case |
| 22/12/2017 | V1.07 | Teemu Räsänen (Savonia) | Description of Savonia’s pilot study |
| 22/12/2017 | V1.08 | J-J Busson (Eolane)  J. Bousquieres (Evolution Energie) | Water distribution network Monitoring in St-Etienne & Hierarchical description of meter network |
| 22/12/2017 | V1.2 | Laura Antikainen (Savonia) | Integration of the pilot reports |

Table of content

[1 Introduction 8](#_Toc503355245)

[2 Water Distribution Network Monitoring in Kuopio 9](#_Toc503355246)

[2.1 General description of the pilot-case 9](#_Toc503355247)

[2.1.1 Aim of the Pilot-Case 9](#_Toc503355248)

[2.1.2 Actors involved and end-users 9](#_Toc503355249)

[2.1.3 Description of the object for monitoring 9](#_Toc503355250)

[2.2 Hardware Devices and their installation approaches 10](#_Toc503355251)

[2.2.1 Sensors and measuring devices 10](#_Toc503355252)

[2.2.2 Installation of sensors and measuring devices 10](#_Toc503355253)

[2.2.3 Powering of sensors 11](#_Toc503355254)

[2.3 Data Transfer Technologies Implemented in Pilot 11](#_Toc503355255)

[2.3.1 Description of the implemented data transfer protocols 11](#_Toc503355256)

[2.4 Measurements and data 11](#_Toc503355257)

[2.4.1 Parameters 11](#_Toc503355258)

[2.4.2 Locations of parameters (sensors) 11](#_Toc503355259)

[2.4.3 Measurement period, interval 12](#_Toc503355260)

[2.4.4 Example figures or other visualization 12](#_Toc503355261)

[2.5 Software Modules and Components 12](#_Toc503355262)

[2.5.1 Interfaces 12](#_Toc503355263)

[2.6 Database 13](#_Toc503355264)

[2.6.1 Data processing, analysing and modelling software’s 13](#_Toc503355265)

[2.7 Data processing and visualization methods 14](#_Toc503355266)

[2.8 Execution of the pilot testing 14](#_Toc503355267)

[2.8.1 Description of the execution and building up the pilot test 14](#_Toc503355268)

[2.9 Main results of the piloting 15](#_Toc503355269)

[3 Water Network Monitoring in Saint Etienne 16](#_Toc503355270)

[3.1 General description of the pilot-case 16](#_Toc503355271)

[3.1.1 Aim of the Pilot-Case 16](#_Toc503355272)

[3.1.2 Actors involved and end-users 16](#_Toc503355273)

[3.1.3 Description of the object for monitoring 16](#_Toc503355274)

[3.2 Hardware Devices and their installation approaches 19](#_Toc503355275)

[3.2.1 Sensors and measuring devices 19](#_Toc503355276)

[3.2.2 Installation of sensors and measuring devices 20](#_Toc503355277)

[3.2.3 Powering of sensors 21](#_Toc503355278)

[3.3 Data Transfer Technologies Implemented in Pilot 21](#_Toc503355279)

[3.3.1 Description of the implemented data transfer protocols 21](#_Toc503355280)

[3.3.2 Description of the gateway performing data concentration 22](#_Toc503355281)

[3.3.3 Installation of the LoRa gateway 23](#_Toc503355282)

[3.4 Measurements and data 23](#_Toc503355283)

[3.4.1 Parameters 23](#_Toc503355284)

[3.4.2 Locations of parameters (sensors) 24](#_Toc503355285)

[3.4.3 Measurement period, interval 24](#_Toc503355286)

[3.4.4 Example figures or other visualization 24](#_Toc503355287)

[3.5 Software Modules and Components 26](#_Toc503355288)

[3.5.1 Interfaces 27](#_Toc503355289)

[3.5.2 Database 27](#_Toc503355290)

[3.5.3 Data processing, analysing and modelling software’s 29](#_Toc503355291)

[3.6 Data processing and visualization methods 29](#_Toc503355292)

[3.6.1 Data pre-processing methods 29](#_Toc503355293)

[3.6.2 Data analysing methods 30](#_Toc503355294)

[3.6.3 Modelling methods 30](#_Toc503355295)

[3.6.4 Data visualization 30](#_Toc503355296)

[3.7 Execution of the pilot testing 33](#_Toc503355297)

[3.7.1 Description of the execution and building up the pilot test 33](#_Toc503355298)

[3.7.2 General results 33](#_Toc503355299)

[3.8 Main results of the piloting 33](#_Toc503355300)

[3.8.1 Dynamic Limit Values 33](#_Toc503355301)

[3.8.2 Leakage Detection 33](#_Toc503355302)

[3.8.3 Malfunction Detection 34](#_Toc503355303)

[3.9 Conclusions and Further Actions 34](#_Toc503355304)

[4 Waste Water Plant Monitoring 35](#_Toc503355305)

[4.1 General description of the pilot-case 35](#_Toc503355306)

[4.1.1 Aim of the Pilot-Case 36](#_Toc503355307)

[4.1.2 Actors involved and end-users 36](#_Toc503355308)

[4.1.3 Description of the object for monitoring 37](#_Toc503355309)

[4.2 Hardware Devices and their installation approaches 37](#_Toc503355310)

[4.2.1 Sensors and measuring devices 37](#_Toc503355311)

[4.3 Data Transfer Technologies Implemented in Pilot 37](#_Toc503355312)

[4.3.1 Description of the implemented data transfer protocols 37](#_Toc503355313)

[4.4 Measurements and data 38](#_Toc503355314)

[4.4.1 Parameters 38](#_Toc503355315)

[4.4.2 Locations of parameters (sensors) 39](#_Toc503355316)

[4.4.3 Measurement period, interval 40](#_Toc503355317)

[4.4.4 Example figures or other visualization 41](#_Toc503355318)

[4.5 Software Modules and Components 41](#_Toc503355319)

[4.5.1 Interfaces 41](#_Toc503355320)

[4.5.2 Database 44](#_Toc503355321)

[4.5.3 Data processing, analysing and modelling software’s 46](#_Toc503355322)

[4.6 Data processing and visualization methods 47](#_Toc503355323)

[4.7 Execution of the pilot testing 54](#_Toc503355324)

[4.7.1 Description of the execution and building up the pilot test 54](#_Toc503355325)

[4.7.2 General results 54](#_Toc503355326)

[4.8 Main results of the piloting 54](#_Toc503355327)

[4.9 Conclusions and Further Actions 54](#_Toc503355328)

[5 River Monitoring 55](#_Toc503355329)

[5.1 General description of the pilot-case 55](#_Toc503355330)

[5.1.1 Aim of the Pilot-Case 55](#_Toc503355331)

[5.1.2 Actors involved and end-users 55](#_Toc503355332)

[5.1.3 Description of the object for monitoring 55](#_Toc503355333)

[5.2 Hardware Devices and their installation 56](#_Toc503355334)

[5.2.1 Sensors and measuring devices 56](#_Toc503355335)

[5.2.2 Installation of sensors and measuring devices 57](#_Toc503355336)

[5.2.3 Powering of sensors 57](#_Toc503355337)

[5.2.4 Powering of the complete system 58](#_Toc503355338)

[5.2.5 Gateway 59](#_Toc503355339)

[5.3 Data Transfer Technologies Implemented in Pilot 59](#_Toc503355340)

[5.3.1 Description of the implemented data transfer protocols 59](#_Toc503355341)

[5.4 Measurements and data 60](#_Toc503355342)

[5.4.1 Parameters 60](#_Toc503355343)

[5.4.2 Locations of parameters (sensors) 60](#_Toc503355344)

[5.4.3 Measurement period, interval 62](#_Toc503355345)

[5.4.4 Example figures or other visualization 62](#_Toc503355346)

[5.5 Software Modules and Components 63](#_Toc503355347)

[5.5.1 Interfaces 63](#_Toc503355348)

[5.5.2 Database 63](#_Toc503355349)

[5.5.3 Data processing, analysing and modelling software’s 63](#_Toc503355350)

[5.6 Data processing and visualization methods 63](#_Toc503355351)

[5.6.1 Data pre-processing methods 63](#_Toc503355352)

[5.6.2 Data analysing methods 64](#_Toc503355353)

[5.6.3 Data visualization 64](#_Toc503355354)

[5.7 Execution of the pilot testing 65](#_Toc503355355)

[5.7.1 Description of the execution and building up the pilot test 65](#_Toc503355356)

[5.7.2 General results 66](#_Toc503355357)

[5.8 Main results of the piloting 66](#_Toc503355358)

[5.8.1 Malfunction Detection 66](#_Toc503355359)

[5.9 Conclusions and Further Actions 66](#_Toc503355360)

[6 Urban Farming 67](#_Toc503355361)

[6.1 General description of the pilot-case 67](#_Toc503355362)

[6.1.1 Aim of the Pilot-Case 67](#_Toc503355363)

[6.1.2 Actors involved and end-users 72](#_Toc503355364)

[6.2 Hardware Devices and their installation approaches 73](#_Toc503355365)

[6.2.1 Sensors and measuring devices 73](#_Toc503355366)

[6.2.2 Actuator (actionneurs ) 73](#_Toc503355367)

[6.2.3 Installation of sensors and measuring devices 73](#_Toc503355368)

[6.2.4 Powering of sensors 76](#_Toc503355369)

[6.2.5 Powering of the complete system 76](#_Toc503355370)

[6.2.6 LoRA gateway solution 78](#_Toc503355371)

[6.3 Data Transfer Technologies Implemented in Pilot 81](#_Toc503355372)

[6.3.1 Description of the implemented data transfer protocols 81](#_Toc503355373)

[6.4 Measurements and data 81](#_Toc503355374)

[6.4.1 Parameters 81](#_Toc503355375)

[6.4.2 Locations of parameters (sensors) 81](#_Toc503355376)

[6.4.3 Measurement period, interval 81](#_Toc503355377)

[6.4.4 Example figures or other visualization 82](#_Toc503355378)

[6.5 Software Modules and Components 82](#_Toc503355379)

[6.5.1 Interfaces 82](#_Toc503355380)

[6.5.2 Open source Database 83](#_Toc503355381)

[6.5.3 Data processing, analysing and modelling software’s 84](#_Toc503355382)

[6.6 Data processing and visualization methods 84](#_Toc503355383)

[6.6.1 Data pre-processing methods 84](#_Toc503355384)

[6.6.2 Data analysing methods 84](#_Toc503355385)

[6.6.3 Modelling methods 85](#_Toc503355386)

[6.6.4 Data visualization time series 86](#_Toc503355387)

[6.6.5 Geographic Data visualization 87](#_Toc503355388)

[6.7 Execution of the pilot testing 87](#_Toc503355389)

[6.7.1 Description of the execution and building up the pilot test 87](#_Toc503355390)

[6.7.2 General results 89](#_Toc503355391)

[6.8 Main results of the piloting 89](#_Toc503355392)

[6.8.1 Dynamic Limit Values 90](#_Toc503355393)

[6.9 Demos 90](#_Toc503355394)

[6.10 Conclusions and Further Actions 93](#_Toc503355395)

[7 Hierarchical description of meter network 94](#_Toc503355396)

[7.1 General description of the pilot-case 94](#_Toc503355397)

[7.1.1 Aim of the Pilot-Case 94](#_Toc503355398)

[7.1.2 Actors involved and end-users 94](#_Toc503355399)

[7.2 Hardware Devices and their installation approaches 94](#_Toc503355400)

[7.2.1 Sensors and measuring devices 94](#_Toc503355401)

[7.2.2 Installation of sensors and measuring devices 94](#_Toc503355402)

[7.2.3 Powering of sensors 95](#_Toc503355403)

[7.3 Data Transfer Technologies Implemented in Pilot 95](#_Toc503355404)

[7.3.1 Description of the implemented data transfer protocols 95](#_Toc503355405)

[7.4 Software Modules and Components 100](#_Toc503355406)

[7.4.1 Graph Database 100](#_Toc503355407)

[7.5 Execution of the pilot testing 110](#_Toc503355408)

[7.5.1 Description of the execution and building up the pilot test 110](#_Toc503355409)

[7.5.2 General results 111](#_Toc503355410)

[7.6 Conclusions and Further Actions 113](#_Toc503355411)

Abstract

This document describes the overall testing and validation of the technical components as integrated solutions. Systems’ hardware/software technological solutions integrated into several infrastructures.

# 1 Introduction

This document describes the overall testing and validation of the technical components as integrated solutions. It includes several pilots that implemented in different countries, namely Turkey, Romania, France and Finland. Content of the pilots described by used hardware and software solutions as well as the integration of the systems in different purposes.

# Water Distribution Network Monitoring in Kuopio

## General description of the pilot-case

### Aim of the Pilot-Case

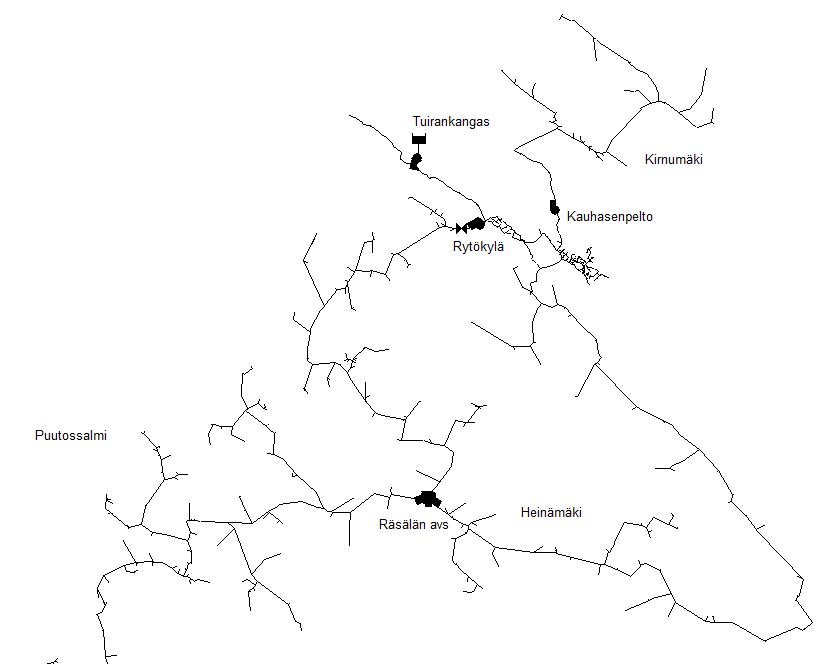
The aim of pilot was to test Savonia’s SAMI –platform in a real world environment by creation of application, which is capable to identify automatically abnormal hydraulic situations (leaks, power failure, equipment failure and blockage) of water distribution network by monitoring pressure and discharge measurements. Water use is time-dependent and dependency based on the type of the water users (residential real estate, industrial users, commercial, nursery school etc.). This kind of dynamic behaviour poses challenges that handled using automated data processing and calculation methods. Moreover, continuous near real time data processing is challenging and set’s demands for data transfer and processing.

### Actors involved and end-users

Pilot operated in Kuopio Waterworks water distribution network, which is located to Vehmersalmi. Savonia University of Applied Sciences provided monitoring as service to Kuopio Waterworks. Savonia managed data transfer, data storage, data analyses and provided refined information using web based application. Moreover, provider of automation system (SCADA) gave technical instructions and opened the data interface to Savonia. In this pilot case, most important end-user of refined information was operators (employees of Kuopio Waterworks) who are controlling the water distribution in Vehmersalmi.

### Description of the object for monitoring

The monitoring system was designed for Kuopio Waterworks district measuring area (DMA) (Figure 1), which was located to Vehmersalmi. For example, the measuring chamber Kauhasenpelto (Figure 2) contains continuous discharge and pressure sensors, which were connected to Siemens automation system (Kuopio Waterworks). Water is provided to approximately 300 residential customers through this measurement chamber. All the measurement chambers utilized within this pilot. These residential customers (water users) belong actually to Itä-Kallaveden Vesiosuuskunta ry (small water co-operative) and Kuopio Waterworks sells water to them. Vehmersalmi is a typical Finnish sparsely populated area.



Water Intake

DMA

Measurement well

Figure 1. Schematic illustration of District Metering Area (DMA) and location of measurement chamber.

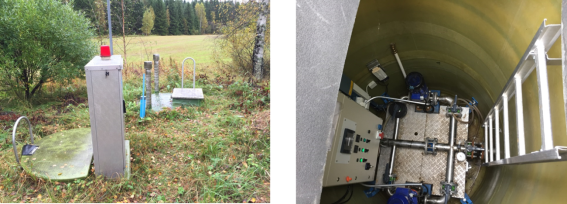


Figure 2. Kauhasenpelto pressure boosting station, where some of the sensors was located in Savonia’s Water-M pilot study. In this pilot, other same kind of chambers used as an installation places of sensors.

## Hardware Devices and their installation approaches

### Sensors and measuring devices

Vehmersalmi water distribution network monitored using following sensors:

* Pressure sensors in 11 different locations
* Flow sensors in 6 different locations
* Water level sensor in one water tank

All of these sensors were included in our monitoring pilot.

### Installation of sensors and measuring devices

Sensors are located into a pressure boosting stations or measuring chambers. The pressure sensors mounted on the pressurized pipe using measurement connectors. Flow measurements are magnetic flow measuring devices.

### Powering of sensors

The pressure booster and the measurement chambers had normal AC power available.

## Data Transfer Technologies Implemented in Pilot

### Description of the implemented data transfer protocols

The sensor in a network are connected to Siemens automation systems PLC’s (like S7-300 in Figure 3) and measurement data directed from Siemens automation system’s WinCC server to SaMi –platform (Savonia) via Internet (JSON –interface).

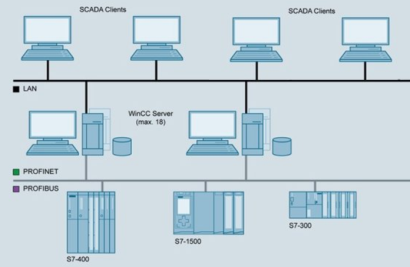


Figure 3. Simple architecture of Siemens WinCC SCADA system.

## Measurements and data

### Parameters

In this pilot, hydraulic measurements discharge (l/s, m3/h) and pressure (kPa, Bar) are used to monitor state of the distribution network. One water level measurement was utilized also.

### Locations of parameters (sensors)

Sensors where located mainly on input pipe of each district metering area (DMA). See example in Figure 1. New measurements were not installed during this project, existing ones were utilized for monitoring. During the project, some ideas for new measurement locations was raised in order to have more comprehensive monitoring system.

### Measurement period, interval

The data collection started 8th of January 2016 and 3 minutes measuring interval was used for each sensor. The data collection and the pilot testing still continues.

### Example figures or other visualization

In this case, the collected data visualized using typical methods suitable for time-series data. The daily variation of water use was easily seen on data, also data quality and outliers analysed using these visualizations. Example of data is presented in Figure 4.

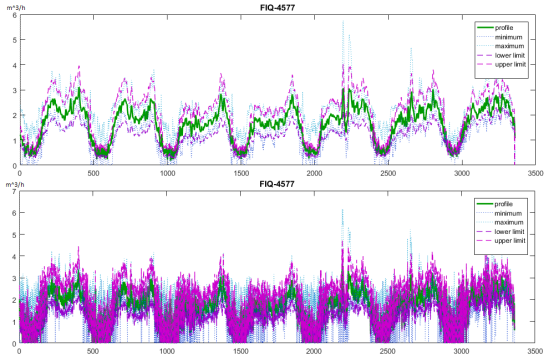


Figure 4. Example of time-series data collected in this pilot.

´

## Software Modules and Components

### Interfaces

Savonia’s SAMI –platform contains open interfaces for getting data. There are two main ways to add data (measurements) to SAMI:

1) Client (= sensor, application, device etc.) sends data to SAMI via SOAP1.1 or SOAP1.2 web service or via HTTP POST request. Client needs to have an internet connection.

2) SAMI reads data from the client. This requires that the client is accessible from the internet and a plugin developed to SAMI knows how to read data from the client.

Plugins tested and/or currently running:

* OPC UA client
* HTTP GET client to read CSV data
* SOAP1.1 client to read data from web service

Description of interfaces and detailed instructions for using them are provided SAMI –website:

<https://sami.savonia.fi/Manage/Home/Help>

## Database

Savonia’s SAMI –platform has open source web interface to save and import all measurement data from the database. It is a .Net Framework 4.5 application and can be run on IIS and it uses MS SQL database as data storage. It was published on the online project hosting service GitHub (can be found at <https://github.com/SavoniaUAS/SaMi>). Third party applications, like created monitoring service, interacts SAMI measurements database by using provided interfaces.

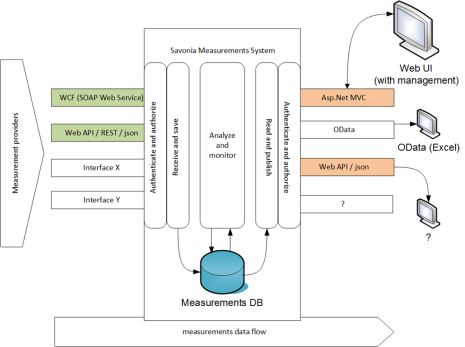


Figure 5. Architecture of Savonia's SAMI -platform.

### Data processing, analysing and modelling software’s

In this pilot, time-series data was processed and analysed first using Matlab –software. It was used to create proper data pre-processing and analysing methods. After that, selected data processing chain was implemented into an online web service using C# programming language (.Net environment).

## Data processing and visualization methods

In this pilot, advanced monitoring application was created and it includes different kind of data processing methods. Application contains following parts which implemented by using various methods:

* Importing sensor data from the database
* Pre-analyzing the collected data
* Forming data profiles and models
* Automatically generating dynamic limit values from the measured data
* Compare live feed to the dynamic limit values
* Alert the user if the limits are exceeded

Used data processing methods designed especially for continuous monitoring and online applications.

## Execution of the pilot testing

### Description of the execution and building up the pilot test

Structure of created monitoring pilot is presented in Figure 6. Savonia University of Applied Sciences carried out all the phases in creation of online monitoring pilot.

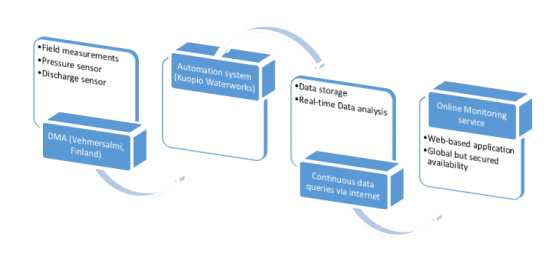


Figure 6. Structure of created monitoring pilot.

Pilot started in January 2016 and user experiences collected during the piloting period until December 2017. Feedback was used to development of monitoring service.

## Main results of the piloting

In this pilot, online monitoring application were created and it was used to observe hydraulic state of Vehmersalmi water distribution network. Main results of piloting are:

* Creation and finishing the SAMI –platform and it’s open interfaces
* Publishing SAMI –platform as open source software in Github
* Testing SAMI –platform with real world application
* Creation of third party application using SAMI –platform. This is dynamic and self-learning monitoring web application for observing hydraulic state of water distribution network.

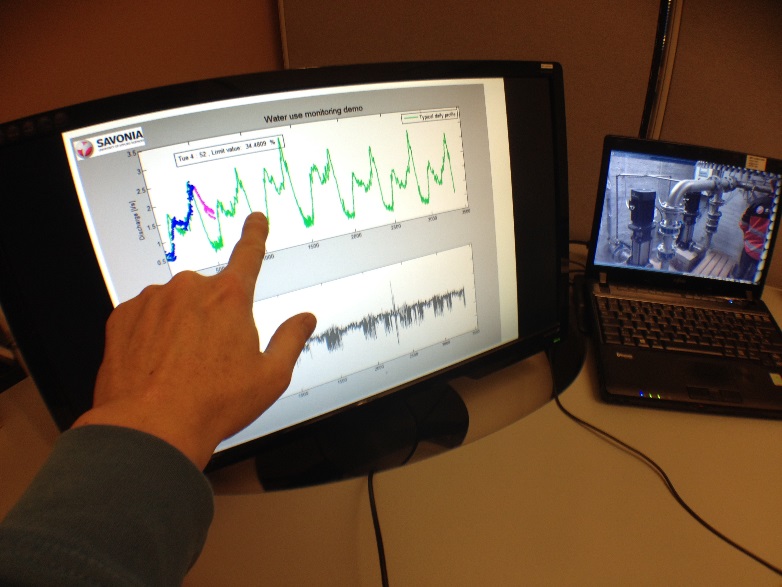
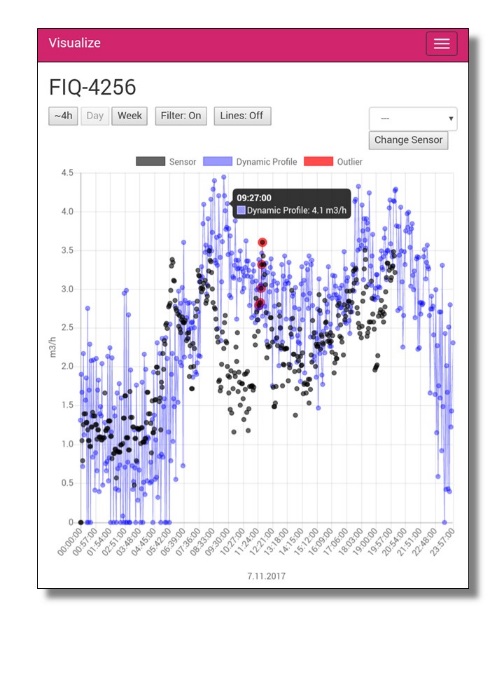
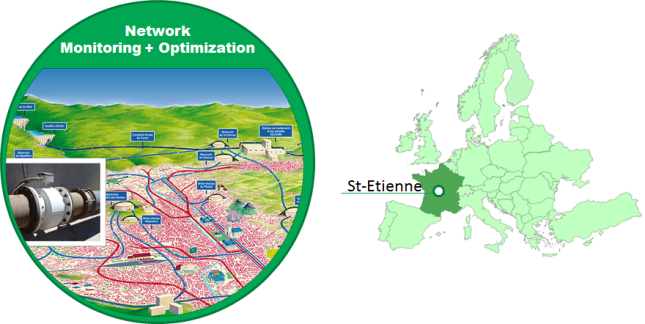


Figure 7. Examples of graphical user interfaces for created monitoring applications (demo version).

Demonstration version of monitoring application was evaluated together with actors in water management industry and feedback was used to develop, for example design of graphical user interfaces. In figure 7, examples of early phase GUI’s are presented. Moreover, networking with actors in water supply industry can be seen as a result of piloting. National and international co-operation will surely be a cornerstone in the future as new applications are established to develop water management in Finland and Europe.

# Water Network Monitoring in Saint Etienne

## General description of the pilot-case



### Aim of the Pilot-Case

This pilot is intended to meet some of the requirements that arise from the use case concerning the monitoring of a water distribution network as described in Work Package 1.

In this context, it allows implementing and validating the architectures at the telecommunications network level and at the storage level as well as the processing and display of the data as defined in the Work Packages 2 and 3.

### Actors involved and end-users

The project was carried out in collaboration with the agglomeration of Saint-Etienne, which represents the customer and the “Stéphanoise des eaux ” company, which is mendaté to ensure the operation, maintenance and evolution of the water network.

EOLANE has deployed the sensors as well as the telecommunication network for the data. UJM and TSE have been implemented the technical IT architecture for storing, processing and displaying data with the support of Cityzen Data on Warp-IO solution.

### Description of the object for monitoring

The water network of the city of Saint-Etienne supplies 28 municipalities for a population of 300,000 inhabitants. Water production can go up 100,000 M3 / day and water reaches the consumer through a 580 km pipeline network.

A closed area was selected as part of the demonstration. Closed area means a subset of the network for which it is possible to control all water volumes entering and leaving the area.

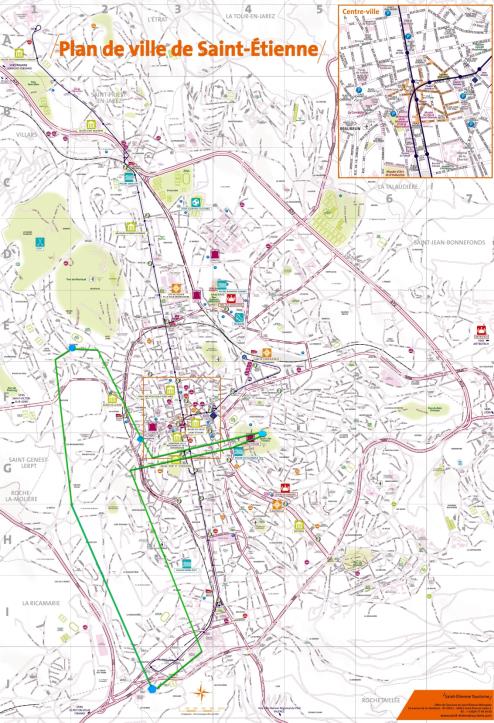
On the other hand, given the very high cost (> 40 k €) of the creation of a measuring point (measuring equipment, earthworks ...), it was decided with the agglomeration to use existing facilities.

The selected area (in green color on the map) consists of 48 km of pipe, is located in urban area and includes 2 entry points and 3 exit points (in blue color on the map). Within the selected area, the distribution network feeds several thousand inhabitants.

Each measurement point in the zone is equipped with a flowmeter and two of the points include a pressure sensor. On the other hand, to complete the experimentation and to check the capabilities of radio technology to cross buildings with a location most often in the basement, public building have been selected to be instrumented with water meters and to allow remote reading.

Some measurement points located in roadside boxes are equipped with temperature probes.

In total, data from five flow meters, two pressure sensors, two temperature probes and three water meters are instrumented.



Selected Area Map

## Hardware Devices and their installation approaches

### Sensors and measuring devices

At the measurement points of the distribution network are implemented using the existing physical sensors. The photo below shows a flowmeter that is mounted on one of the pipes corresponding to an exit point (diameter of 800mm).



For information, the diameter of the pipes of the entry points is of the order of 800mm. The pipe diameter of one of the exit points is also 800mm and the other two have a diameter of about 300mm

The sensors are wired to measuring heads, which are also wired to radio transmitters using LoRa technology. Depending on the information you want to provide the transmitters have two types of interfaces:  
- Dry loop type interface (relay) for supplying an internal pulse counter to the transmitter;  
- 4-20mA current loop type interface, the transmitter performing the current measurement which is proportional to the physical value measured by the sensor.

Transmitters used to equip meters in public buildings are identical to those used with distribution system meters (Dry loop type interface). The transmitter must be connected to a pulse generator that mounts to the meter head. The pulse generator is specific to the type of counter (brand, model).

### Installation of sensors and measuring devices

As indicated the transmitters are either connected to measuring heads (pressure sensor, flowmeter) through a wired interface (4-20mA current loop, Dry loop) or connected to a pulse generator in the case of volumetric meters for individuals or communities.



Transmitter interconnected to measuring heads



Transmitter interconnected to pulse generator

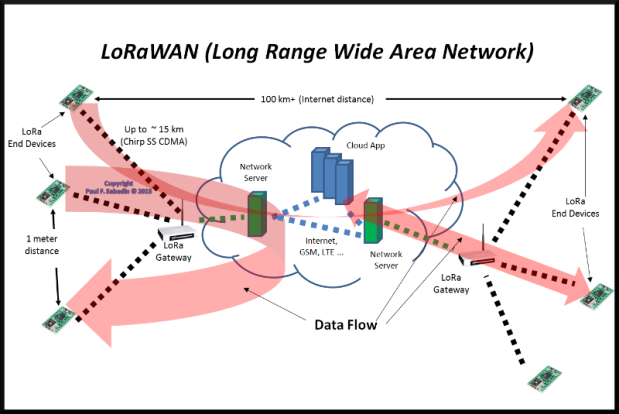
### Powering of sensors

The LoRa transmitters are waterproof and dustproof (IP 68). A battery giving them a 15-year operating life (4 measurements and 4 transmissions per day) powers the transmitters.

## Data Transfer Technologies Implemented in Pilot

### Description of the implemented data transfer protocols

The transmitters use the LoRaWan protocol.

****

**LoRaWan Network**

**LoRa** is a cellular style wireless system which allows for private deployments. LoRa refers to PHY layer technology and LoRaWAN includes the communications protocols. It enables low data rate, ultra-low power and long-range communications IoT applications and uses the ISM bands (868 MHz for Europe, 915 MHz for North America, and 433 MHz for Asia). As shown in Figure 3, LoRa uses a star topology in which sensors with LoRa chip send encrypted data to the LoRa gateways. Then LoRa gateways relay messages to the servers at the backend that decrypt the data and process it.

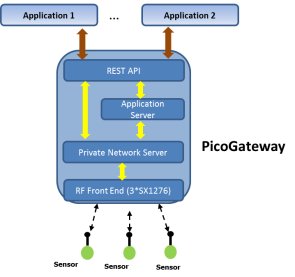
Here are the characteristics of LoRa technology:

* A range of 15 - 20 km in environments with less distortions and around 2 km in urban environments
* Data rates of 0.3 kbps to 50 kbps.
* Energy efficient communication with end device batteries lasting up to 10 yrs
* Secure communication

### Description of the gateway performing data concentration

We implemented the gateway developed by Éolane Company as part of the project. This gateway has some peculiarities.

Dedicated to Private wireless network based on LoRa™ technology, the **eoPicoGW** (gateway commercial name) is a low-cost indoor platform which embeds all the required hardware & software components for servicing a complete application.



**eoPicoGW Gateway**

In that way, the **eoPicoGW** is rather a complete Wireless Network Platform than a “simple” gateway.

It includes:

* A low cost **Radio Board** able to manage up to three concurrent LoRaWAN channel in SF12.
* A **Core Board** which embeds following Software Layer :
  + **HAL Layer** (RF\_Engine) in charge of LoRa™ radio reception & transmission
  + **Network Server Layer**: device management on the network point of view (allocation, integrity, network authentication…)

* + **Application Server Layer** (optional) intended to manage device of a given application and to expose the associated / collected data to the final user using API based on REST web services. It can be seen as the container of the virtualized sensors of the application

All these software components embedded on a single platform will contribute to reduce system & integration effort.

The **eoPicoGW** proposes different application interfaces on the application / Data layer:

* Rest API
* Local visualization GUI (Graphical User Interface)

The embedded GUI (future release) allows easily visualizing the data recovered from the sensors connected to the gateway, directly from a local network or distant network.

The Rest API allows developing applications around the data recovered from the sensors in either a simple local network or any other network configurations.

NB: the **eoPicoGW** is LoRaWAN End Node Compatible

### Installation of the LoRa gateway



Four LoRa gateways were needed to cover the selected area. For reasons of convenience, each gateway is associated with a cellular router, which allows easy access to the gateway to manage it. Gateways are installed in public buildings.

## Measurements and data

### Parameters

The sensors can be used to obtain water volume, pressure and temperature parameters at different points of the network within the experimental zone.

### Locations of parameters (sensors)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Location** | **Sensor Type** | **LoRa Id** | **Data conversion** | **Network Location** |
| Courteline | Flowmeter | 70B3D580A010095A | 1 pulse / m3 | Entry point |
| Courteline | Flowmeter | 70B3D580A010095D | 1 pulse / m3 | Entry point |
| Courteline | Pressure | 70B3D580A010104F | 4mA = 0 Bar 20mA = 16 Bar | - |
| Jardin des plantes | Flowmeter | 70B3D580A010095F | 1 pulse / m3 | Exit point |
| Jardin des plantes | Pressure | 70B3D580A01034C8 | 4mA = 0 Bar 20mA = 16 Bar | - |
| Chavassieux | Flowmeter | 70B3D580A0100865 | 1 Pulse / 100 l | Exit point |
| Chavassieux | Temperature | 70B3D580A0100599 | °C | - |
| Descours | Flowmeter | 70B3D580A01008AD | 1 Pulse / 100 l | Exit point |
| Descours | Flowmeter | 70B3D580A01008F6 | 1 Pulse / 100 l | Exit point |
| Descours | Temperature | 70B3D580A0100594 | °C | - |
| Médiathèque | Meter | 70B3D580A0100B20 | 1 Pulse / 1 l | - |
| Mairie annexe | Meter | 70B3D580A0100B55 | 1 Pulse / 1 l | - |
| Mairie annexe | Temperature | 70B3D580A010055B | °C | - |
| Police municipale | Meter | 70B3D580A0100B8D | 1 Pulse / 1 l | - |

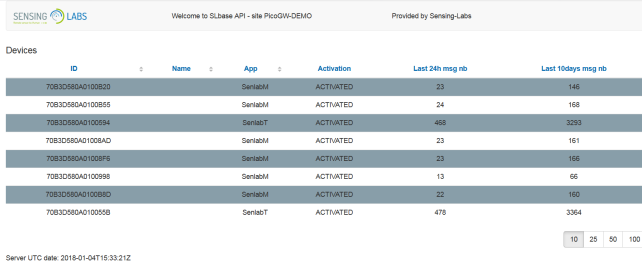
### Measurement period, interval

The transmitters can be parameterized as needed at the factory. The life of the battery depends on the number of acquisitions and the number of transmissions made during the day.

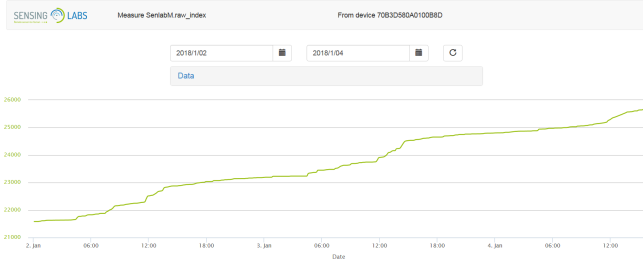
In the case of the experiment, the transmitters associated with the flow and pressure sensors are configured to perform 6 measurements per hour and 24 transmissions per day.

### Example figures or other visualization

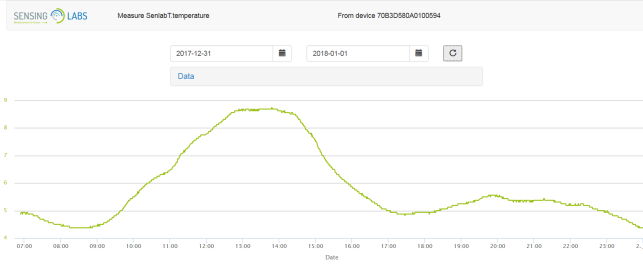
The gateway embeds a Web server allowing visualizing the raw data of the sensors which depend on it.



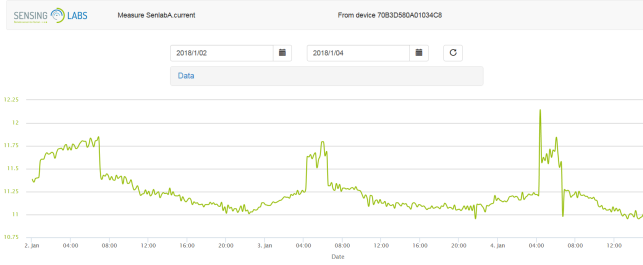
**List of devices (PicoGW N° 4)**



Pulse sensor (Police – PicoGW N° 4)



**Temperature (Descours – PicoGW N° 4)**



**Pressure (Jardin des Plantes – PicoGW N° 2)**

## Software Modules and Components



**Platform Architecture**

The platform architecture is shown in the figure above. The sensors connect to a LORA gateway and the gateway then regularly sends the data to the WarpIO[[1]](#footnote-1) platform. When the data arrives to the platform, the Node API interface is used for parsing and converting the raw data to the WarpIO database format. WarpIO then stores the data in the form of time series. Another interface is managed by Grafana[[2]](#footnote-2) that allows to visualize the real-time data using a web browser.

### Interfaces

**WarpIO API :**

WarpIO listens on port 8080 and its exposes several APIs:

Ingress (/api/vX/update): for sending data to the WarpIO platform

WarpScript (/api/vX/exec): for executing analysis expressed in the WarpScript language (WarpIO proposes its own functions and language)

Fetch (/api/vX/fetch): for retrieving raw Geo Time Series (GTS) data in an extremely quick an efficient way

Delete (/api/vX/delete): for erasing data from the Warp 10 platform

Meta (/api/vX/meta): for setting Geo Time Series attributes

**Node API:**

Node API is based on NodeJS + Express framework. It listens on port 3000. The following steps take place:

1. LORA gateway sends the data with other information such as device id and timestamp using HTTP POST.
2. Node API then parses the data using bitwise operations to extract the useful values.
3. These values, timestamp along with other metadata are stored into the warp database using HTTP POST and WarpIO interface: /api/v0/update.
4. Node API sends back HTTP 204 ok No content acknowledgement to the LoRA Gateway.

**Grafana:**

The data visualisation is based on Grafana tool. It allows to create dashboards and graphs which are fed by the real-time data from the WarpIO platform. Grafana listens on port 4000 and its dashboards use the WarpIO interface /api/v0/exec to execute scripts for data processing and obtaining the processed data. This in turn is displayed in the graphs.

### Database

The data is stored in WarpIO platform and the metadata or attributes can be defined by the application. Refering to WarpIO documentation, the data is stored in the format:

TS/LAT:LON/ELEV NAME{LABELS} VALUE

where each element has the following syntax and meaning:

TS Timestamp of the reading, in microseconds since the Unix Epoch.

LAT:LON \*Optional\* geographic coordinates of the reading, using

ELEV \*Optional\* elevation of the reading, in millimeters

NAME Class name or the database table name of the reading. For this pilot the name “Sensor” is used.

The encoding of character `{` (Unicode \*LEFT CURLY BRACKET\*, \*0x007B\*) is MANDATORY.

LABELS Comma separated list of labels, using the syntax `key=value` where both `key` and `value` are URL encoded UTF-8 character strings. If a key or value contains `,` (Unicode COMMA, 0x002C),`}` (Unicode RIGHT CURLY BRACKET, 0x007D) or `=` (Unicode EQUALS SIGN, 0x003D), those characters MUST be encoded.

VALUE The value of the reading. It can be of one of four types: \*LONG\*, \*DOUBLE\*, \*BOOLEAN\*, \*STRING\*

For the pilot in St-Etienne the metadata stored is as follows and WaterM ontology (deliverable D4.1) was used to define this metadata:

**ID, Name, Location, Gateway, Port, Type, Latitude, Longitude, Unit, MeasureType, MeasureToImpulseRatio.**

Where ID is device ID, Name is device name, Location is name of location, gateway is the id of gateway to which this device is connected, Port is the gateway port, Type is type of device, Latitude and Longitude are the coordinates, Unit is unit of measure, MeasureType is what quantity is measured using the device, MeasureToImpluseRatio is the ratio of measure actual value per measure provided by the device (For example if 1 impulse means 100 litres of volume then this value is 100 and unit is litres)

Here are the examples of metadata stored for different sensors. This metadata allows to search for different sensors in the query. For example to search for all ImpulseMeters the query will ask for { ‘Type’ ‘ImpulseMeter’ }.

["70B3D580A010095A", "ImpulseMeter", "Courteline", "waterm01", "8080", "ImpulseMeter", "45.4042909", "4.3806775", "Litre", "Volume", "1000"],

["70B3D580A010095D", "ImpulseMeter", "Courteline", "waterm01", "8080", "ImpulseMeter", "45.4042909", "4.3806775", "Litre", "Volume", "1000"],

["70B3D580A010104F", "AmpereMeter", "Courteline", "waterm01", "8080", "AmpereMeter", "45.4042909", "4.3806775", "Bar", "Pressure", "0.005"],

["70B3D580A01034C8", "AmpereMeter", "Jardin des Plants", "waterm02", "8080", "AmpereMeter", "45.4341786", "4.3976615", "Bar", "Pressure", "0.005"],

["70B3D580A010095F", "ImpulseMeter", "Jardin des Plants", "waterm02", "8080", "ImpulseMeter", "45.4341786", "4.3976615", "Litre", "Volume", "1000"],

["70B3D580A0100865", "ImpulseMeter", "Chavassieux", "waterm03", "8080", "ImpulseMeter", "45.446657", "4.3650832", "Litre", "Volume", "100"],

["70B3D580A0100599", "Temperature", "Chavassieux", "waterm03", "8080", "Temperature", "45.446657", "4.3650832", "Celsius", "Temperature", "0.0625"],

["70B3D580A0100B20", "WaterMeter", "Mediatheque Descours", "waterm04", "8080", "ImpulseMeter", "45.4363582", "4.3713645", "Litre", "Volume", "1"],

["70B3D580A0100B55", "WaterMeter", "Mairie Tarantaize", "waterm04", "8080", "ImpulseMeter", "45.4363398", "4.3779305", "Litre", "Volume", "1"],

["70B3D580A0100594", "Temperature", "Tarantaize", "waterm04", "8080", "Temperature", "45.4373148", "4.3796956", "Celsius", "Temperature", "0.0625"],

["70B3D580A01008AD", "ImpulseMeter", "Tarantaize", "waterm04", "8080", "ImpulseMeter", "45.4373148", "4.3796956", "Litre", "Volume", "100"],

["70B3D580A01008F6", "ImpulseMeter", "Tarantaize", "waterm04", "8080", "ImpulseMeter", "45.4373148", "4.3796956", "Litre", "Volume", "100"],

["70B3D580A0100998", "WaterMeter", "Primary School Tarantaize", "waterm04", "8080", "ImpulseMeter", "45.4373223", "4.3796956", "Litre", "Volume", "1"],

["70B3D580A0100B8D", "WaterMeter", "Police Tarantaize", "waterm04", "8080", "ImpulseMeter", "45.4379022", "4.3779435", "Litre", "Volume", "1"],

["70B3D580A010055B", "Temperature", "Mairie Tarantaize", "waterm04", "8080", "Temperature", "45.4363398", "4.3779305", "Celsius", "Temperature", "0.0625"]

### Data processing, analysing and modelling software’s

Data processing is done by the WarpIO platform as it provides the warpscript[[3]](#footnote-3) functionality which provides the option of in-platform processing of the data. The WarpScript data manipulation environment allows the retrieval and analysis of data stored in the WarpIO storage platform.

WarpScript is an extensible stack oriented programming language which offers more than 600 functions and several high-level frameworks to ease and speed the data analysis. The idea is to create scripts containing data analysis code and submit them to the platform, they will execute close to where the data resides and in the end, one gets the result of that analysis as a JSON object that can be easily integrated with any application.

## Data processing and visualization methods

### Data pre-processing methods

Data pre-processing is done using the functions[[4]](#footnote-4) provided by warpscript. Pre-processing is needed due to unreliable nature of the sensor data. Sometimes sensor data is missing and most of the times due to asynchronous nature of sensor data arrival, the timestamps of different time series are not synchronised, for example sensor A may send data every 10s, but at 19h01m31s – 19h01m41s and sensor B may send at 19h01m29s – 19h01m39s and so one. Sometimes, when working with multiple time series together for analyses it may be desirable to shift some time series to bring them all together at a single common time-axis.

Some pre-processing functions used by the Pilot in St-Etienne are bucketizer.mean and INTERPOLATE. The first function allows to calculate a mean value and can also be used to bring the sensor data to common time-axis. INTERPOLATE allows to interpolate missing sensor values.

Here is an example of Grafana query plus a warpscript. Note that read token below is an authentication token generated by WarpIO and provided offline to the application so that is can use it for authentication in its HTTP POST queries. FETCH function simply fetches the time series, the variable $end is the last desired timestamp to be displayed in the graph, 600000000 is the sliding window of 10 minutes that will be used to compute the mean and 0 is default option for this function:

'\*\*\*readtoken\*\*' 'Sensor' { 'id' '70B3D580A010095D' } $startISO $endISO ] FETCH

[ SWAP bucketizer.mean $end 600000000 0 ] BUCKETIZE

INTERPOLATE

### Data analysing methods

Warpscript functions are again used for analysing the data.

The function mapper.delta is used to calculate the water flow rata as the installed ImpulseMeters only provide cumulative volumes and not the flow rate. The mapper.delta function is used to calculate the delta between consecutive values. Here is an example:

[ SWAP mapper.delta 1 0 0 ] MAP

The REDUCE function is used to merge multiple time series into one after applying functions such as reducer.sum (it calculates the sum of values of different time series at same timestamp). Here is an example:

(Note that before these lines we assume that 2 time-series have already been fetched)

[] // list of labels to define the equivalence classes. For example: 'label0' 1 ->LIST

reducer.sum

4 ->LIST

REDUCE

The MAP function is also used to apply different functions to a given time series. This is useful for example to multiply -1 to the values of a given time series, which can then be used to subtract its values from other time series with reducer.sum function. It can also be used to for example multiply 0.1 to the delta values to have a flow rate per minute (as delta corresponds to 10 minutes). These functions were used to calculate the overall consumption of a zone by adding the flow rates of input pipes and subtracting the flow rates of the output pipes. Here is an example:

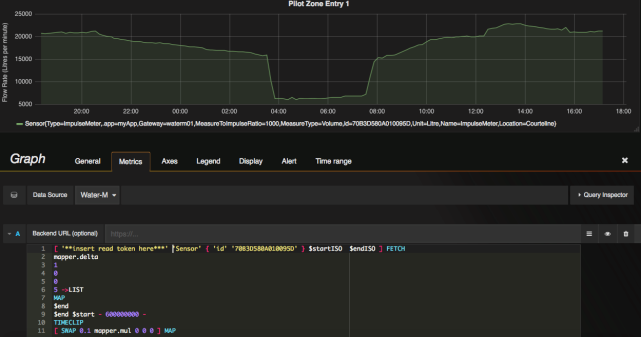
[ SWAP -1 mapper.mul 0 0 0 ] MAP

### Modelling methods

Data modelling was done by creating an ontology and it is described in the data model deliverable (D4.1)

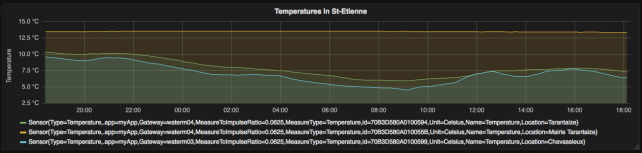
### Data visualization

Data visualization is done with the help of Grafana tool. It allows one to create different dashboards showing monitoring graphs. Grafana has a query builder for each graph that can be used to write queries including warpscripts that will be executed by WarpIO platform. An example is shown below. The graph shows the flow rate of an entry pipe. Below in the query builder the query to fetch a time series along with the warpscript code is also shown. Note that the TIMECLIP function below is used to clip the last and the first values of delta as they may be unreliable in case the latest data did not arrive.

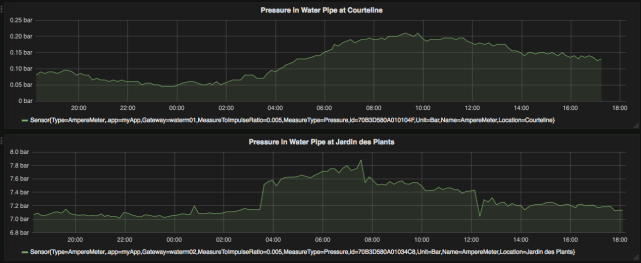


**Grafana Query builder**

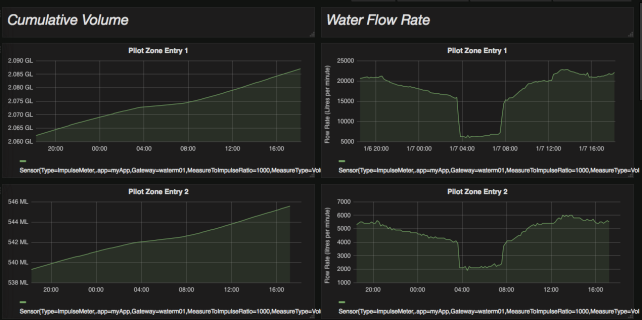
For the Water distribution monitoring application, we created dashboards to show Temperatures, Pressures, Volumes, WaterMeters data, etc as below:



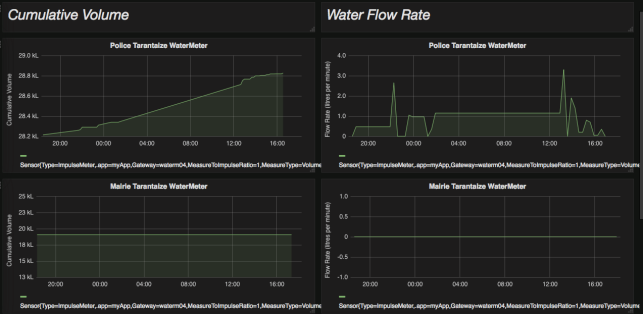
**Dashboard Temperatures**



**Dashboard Pressures**



**Dashboard Volumes and Flow Rates of the zone**



**Dashboard WaterMeters**

## Execution of the pilot testing

### Description of the execution and building up the pilot test

As a first step, different candidate zones were shortlisted in St-Etienne for the pilot test. After initial analyses and visits, a zone was identified having 2 entries and 3 exits (3rd exit had 2 sub exits). After that the installation of sensors (ImpulseMeters, WaterMeters, AmpereMeters) and Lora gateway took place. Then the WarpIO platform was configured and APIs were created to arrive at the final Water Monitoring application.

### General results

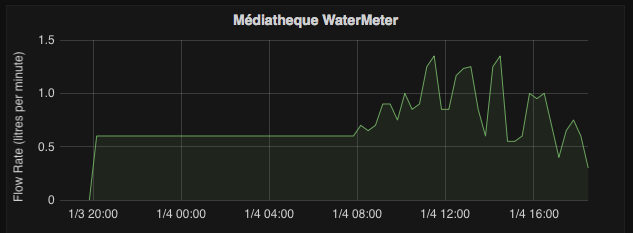
A water monitoring application was created which allows to monitor the water distribution in real time with a fine granularity. The information provided by the application is very useful to optimise the water distribution, identify some problems such as leaks and eventually optimise the performance.

## Main results of the piloting

### Dynamic Limit Values

The monitoring of some parameters is limited to every 10 minutes for some devices or sensors. This is done to preserve battery life of these sensors.

### Leakage Detection

During the course of pilot testing a leak was detected in one of the locations. As seen in the figure below a minimum flow rate of around 0.6 litres per minute is present even during closing hours of Mediatheque. This indicates a leak; which in fact was repaired around 16h indicating a flow rate of zero during closed hours afterwards. 

### Malfunction Detection

No Malfunction was detected during the pilot testing except missing values of some sensors due to bad radio conditions and the leak detected above.

## Conclusions and Further Actions

Wireless LoRa Technology is well adapted for sensor installation in the basement (The water meter of the media library is located on the second basement of a reinforced concrete building at a distance of approximately 500 meters from the LoRa Gateway).

LoRa Network Server (LoRa MAC) integrated into the gateway simplifies the network architecture (no additional external server to deploy)

Thanks to WarpIO, the computing solution is easily deployable and the use of these components allows an efficient management of a set of sensors. This implementation allows effective monitoring of the water network.

The pilot testing and demonstration in St-Etienne is an important milestone towards the goal of the city to become smart city.

In the future, it would be necessary to instrument the network with a much larger number of sensors that would allow the development of indicators to optimize the water network. However, the installation of numerous sensors throughout the network is a long and costly operation that must be budgeted and planned very early in advance by agglomerations, in particular for existing networks.

# Waste Water Plant Monitoring

## General description of the pilot-case



Pilot is located Eskisehir industrial zone area.



Threatment Pools



Threatment Pools



Threatment Pools



Control Unit Center



Management Building

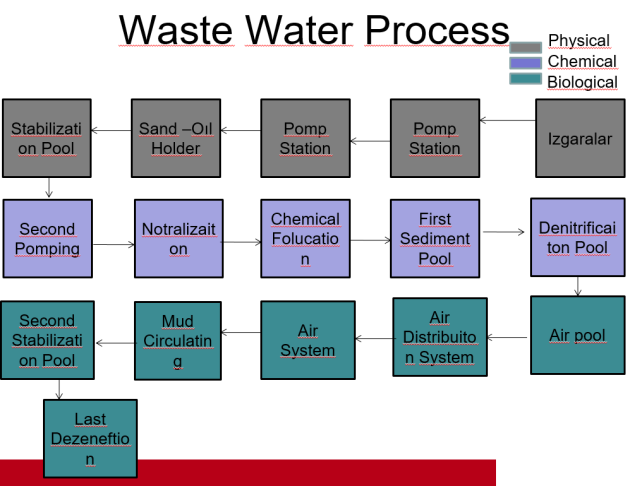
### Aim of the Pilot-Case

The aim of the Pilot case is to monitor waste water process and in case of rule events fired according to rule system will provide alarms warnings. Plant is located in Industrial zone area and more than 600 factories send their waste water to this plant.

### Actors involved and end-users

Waste water plant managerial stuff involved in the project.

### Description of the object for monitoring



## Hardware Devices and their installation approaches

### Sensors and measuring devices

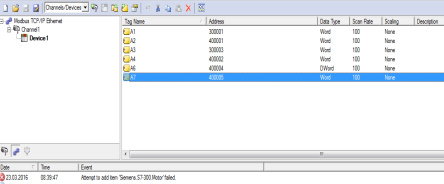
Plant has a scada system to control whole system .Sensors controlled by scada system.

## Data Transfer Technologies Implemented in Pilot

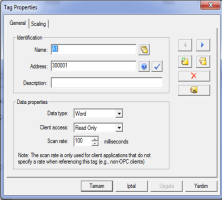
### Description of the implemented data transfer protocols

For the data transfer system must integrated with scada system and should communicate plc parameter by using a valid protocol which system involved.

Modbus is protocol used by existing scada system.Modbus TCP/IP Ethernet protocol is used and figures shown below protocol and related tags description below.



Tag definition screen shown below.



## Measurements and data

### Parameters

**Parameters Used For Waste Water Processes**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PARAMETER** | **INPUT** | | **OUTPUT** | | **FREQUENCY** |
| **MİN** | **MAX** | **MİN** | **MAX** |
| Temperature | 8 | 20 | 6 | 20 | 3 per day |
| Ph | 6 | 9 | 6 | 9 | 3 per day |
| Solid | 700 | 800 | 50 | 100 | 3 per day |
| Oil | 400 | 500 | 5 | 10 | 3 per day |
| Con | 1000 | 1200 | 200 | 300 | 3 per day |
| Sulfate | 900 | 1000 | 1400 | 1500 | 3 per day |
| Total Nitrogen | 120 | 130 | 10 | 15 | 3 per day |
| Total Phosphorus | 15 | 20 | 1 | 3 | 3 per day |
| Total Cyanide | 10 | 15 | 1 | 2 | 3 per day |
| Total Plumbum | 3 | 5 | 1 | 2 | 3 per day |
| Total Chromium | 3 | 5 | 1 | 2 | 3 per day |
|  | DESCRIPTON | | | |  |
| Phase1Level | Level of the waste water level for Phase1 | | | | 1 per minute |
| Phase3Level | Level of the waste water level for Phase3 | | | | 1 per minute |
| Phase3O2 | Phase3 oxygen level | | | | 1 per minute |
| Phase4FlowRate | Phase4 flowrate | | | | 1 per minute |
| Phase5Ph | Phase5 ph level | | | | 1 per minute |
| Phase6O2 | Phase6 oxygen level | | | | 1 per minute |
| Phase7O2 | Phase7 oxygen level | | | | 1 per minute |

Parameter : Name of the parameter used for process of the waste water analysis.

Input : Valid input value range for parameter.

Output : Valid output vale range for parameter.

Frequency : Sample frequency (per day).

**Parameters Used For Sensor Events**

|  |  |  |
| --- | --- | --- |
| **PARAMETER** | **DESCRIPTION** | **VALUES** |
| sensorID | Sensor identification number. | any integer value |
| eventType | Type of the current event. | input, error, Warning, alarm, action |
| sensorName | Name of the curent sensor. | temperature, ph, solid, etc. |
| frequency | Sample collection frequency for current sensor. | any integer value |
| value | Measured value for current sensor. | any float value |
| typeOfMeasure | Current state of the measurement. | input, output, state |
| timeStamp | Time stamp of the current measurement. | any time value |
| description | Short description of the measurement. | “Phase 1 waste Water level”, etc. |
| coordinates | Geographical location (lat,lng) of the current sensor. | “39.756731, 30.626215” |

### Locations of parameters (sensors)

**Phase 1:Waste Water input ,Rule1:if the input water more than 1000 m3 start to work second grid**

**Level control sensor located in Phase1 rule control parameter Phase1level**

**Phase 2:Stabilization Pool ,**

**Rule2:if the water level in the stabilization pool more than 10 m start to work second pomp,**

**if the water level in the stabilization pool more than 15 m start to work third pomp**

**Rule3:O2 quantity level less than 2 mg/liter start to work Blower unit in the stabilization pool.**

**Phase 4: Second Pumping**

**Control parameter name Phase4Flowrate**

**Rule4: If Flow rate 700 liter /minutes second pumping start to work for discharging the waste water in this phase**

**Phase5: Neutralization Phase**

**Rule control parameter named Phase5Ph**

**In this phase pH parameter is important because pH sensor data bigger than 9 Lime thank pumping start to work to push the lime into neutralization pool**

**Phase6 :Nitrification Pool**

**Rule control named Phase6O2 controlling O2 level**

**Phase6O2 :if O2 probe sensor data smaller than 1,5 mg/L Nitrogen(Azote) Nitrogen**

**and phosphorus pumping motor start to work**

**Phase7 :Biological Pool there are blower unit is used for to send oxygen into the pool**

**Rule control named Phase7O2 controlling O2 level**

**If the O2 level 2 mg o2/liter less start Blower 2 motor**

**If the o2 level 0.5 mg O2/liter less start Blower 3 motor**

|  |  |  |
| --- | --- | --- |
| Phase1Level | Level of the waste water level for Phase1 | 1 per minute |
| Phase3Level | Level of the waste water level for Phase3 | 1 per minute |
| Phase3O2 | Phase3 oxygen level | 1 per minute |
| Phase4FlowRate | Phase4 flowrate | 1 per minute |
| Phase5Ph | Phase5 ph level | 1 per minute |
| Phase6O2 | Phase6 oxygen level | 1 per minute |
| Phase7O2 | Phase7 oxygen level | 1 per minute |

### Measurement period, interval

Measurement period and interval can be changed depending on the parameters that are measured. Some parameters must be measured per minute and some others can be measured on 30 minutes.

Following parameters measured 3 times per day

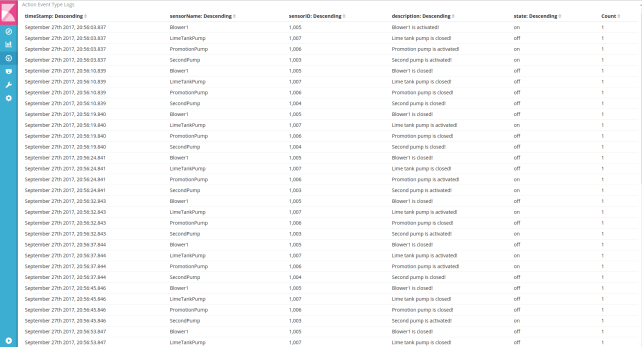
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PARAMETER** | **INPUT** | | **OUTPUT** | | **FREQUENCY** |
| **MİN** | **MAX** | **MİN** | **MAX** |
| Temperature | 8 | 20 | 6 | 20 | 3 per day |
| Ph | 6 | 9 | 6 | 9 | 3 per day |
| Solid | 700 | 800 | 50 | 100 | 3 per day |
| Oil | 400 | 500 | 5 | 10 | 3 per day |
| Con | 1000 | 1200 | 200 | 300 | 3 per day |
| Sulfate | 900 | 1000 | 1400 | 1500 | 3 per day |
| Total Nitrogen | 120 | 130 | 10 | 15 | 3 per day |
| Total Phosphorus | 15 | 20 | 1 | 3 | 3 per day |
| Total Cyanide | 10 | 15 | 1 | 2 | 3 per day |
| Total Plumbum | 3 | 5 | 1 | 2 | 3 per day |
| Total Chromium | 3 | 5 | 1 | 2 | 3 per day |
|  |  |  |  |  |  |

In the rules working among process to awake the alarms and events measurement period per minute.

|  |  |  |
| --- | --- | --- |
| Phase1Level | Level of the waste water level for Phase1 | 1 per minute |
| Phase3Level | Level of the waste water level for Phase3 | 1 per minute |
| Phase3O2 | Phase3 oxygen level | 1 per minute |
| Phase4FlowRate | Phase4 flowrate | 1 per minute |
| Phase5Ph | Phase5 ph level | 1 per minute |
| Phase6O2 | Phase6 oxygen level | 1 per minute |
| Phase7O2 | Phase7 oxygen level | 1 per minute |

### Example figures or other visualization

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PARAMETER** | **INPUT** | | **OUTPUT** | | **FREQUENCY** |
| **MİN** | **MAX** | **MİN** | **MAX** |
| Temperature | 8 | 20 | 6 | 20 | 3 per day |
| Ph | 6 | 9 | 6 | 9 | 3 per day |
| Solid | 700 | 800 | 50 | 100 | 3 per day |
| Oil | 400 | 500 | 5 | 10 | 3 per day |
| Con | 1000 | 1200 | 200 | 300 | 3 per day |
| Sulfate | 900 | 1000 | 1400 | 1500 | 3 per day |
| Total Nitrogen | 120 | 130 | 10 | 15 | 3 per day |
| Total Phosphorus | 15 | 20 | 1 | 3 | 3 per day |
| Total Cyanide | 10 | 15 | 1 | 2 | 3 per day |
| Total Plumbum | 3 | 5 | 1 | 2 | 3 per day |
| Total Chromium | 3 | 5 | 1 | 2 | 3 per day |



* Data Table For Analysed Results

## Software Modules and Components

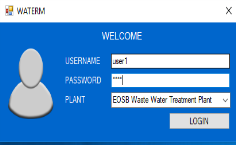
### Interfaces

There are 3 interfaces;

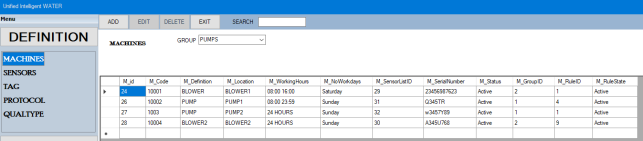
1-Communication software for Modbus TCP/IP OPC SDK.

2-User interface which is used for sensor parameters and data relation combination and rules created screens.

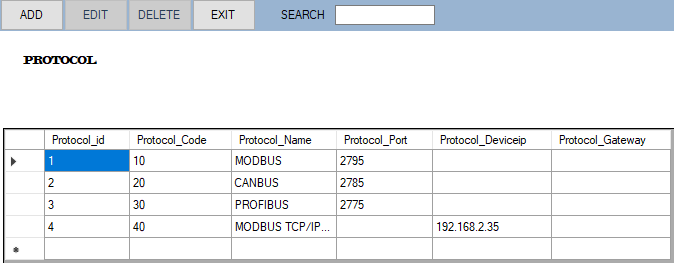
3-Measurement data sending Apache/Kafka by using API.



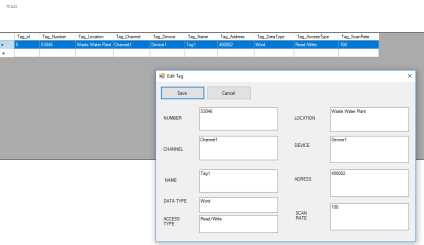
Definition Screen of the Plant assets and combined related sensors and rules.



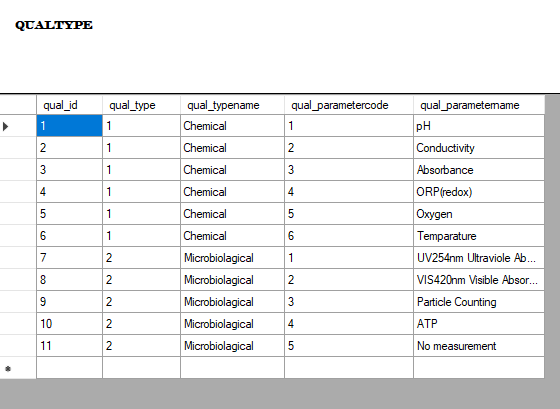
Deifinition of the protocols supported more than one protocols.



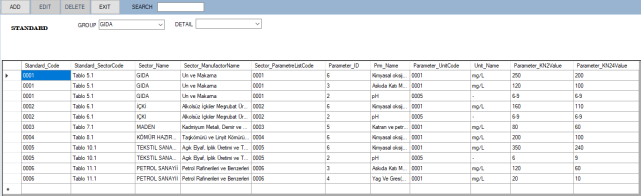
Tag Definition Screen



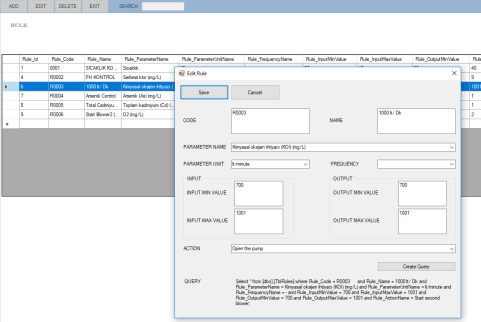
Quality Type Definition



Industrial Standards Definition (These definition given by Authorized Management Govermental Data)



Rule Definition Screen



### Database

Waste Water use case has two type of database structure. Local and Global

Local definition written below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Purpose** | **Structure** |
| **Machines** | Definition of the Machines which works in the plant and can control by the system. | Aim of this definition is to known by the water-m platform and what duty in the system and how it is control in which cases. | M\_id,  M\_Definition,  M\_Location,  M\_Workinghours,  M\_NoWorkdays,  M\_Sensorid,  M\_SerialNumber |
| **Parameters** | Definition of the parameters which values will be controlled and in which cases besides upper and lower limits. | Controlling the parameters with their upper and lower limits according to EU standards. | P\_id,  P\_Name,  P\_GlobeName,  P\_Frequency,  P\_inputvalue1,  P\_inputvalue2,  P\_SampleValue1,  P\_SampleValue2,  P\_SampleOutputValue1,  P\_SampleOutputValue2 |
| **QualityType** | Definition of the Quality measurement type with respect to EU standards with their needs and not needs and grouping of the quality type like Chemical,Microbiological ,Biological. | Aim of this table is to define and grouping control parameters i.e.  Chemical–Conductivity  Chemical–ORP(Redox)  Microbio–UV254nm Ultr.  Microbio –VIS420nm Vis.Absorbence | q\_id,  q\_type,  q\_typename,  q\_controlparametercode,  q\_controlpname |
| **Plant** | Definition of the plant | Aim of this table is to define Location,country,City | Plnt\_id,  Plnt\_name,  Plnt\_Location,(GEO),  Plnt\_Country,  Plnt\_City |
| **ProtocolType** | Definition of the protocol for the communication with local scada or other systems. | Aim of this table is to define which protocol,what is ip address,gateways etc.. | Protocol\_id,  Protocol\_Name,  Protocol\_port,  Protocol\_ip,  Protocol\_gateway |
| **Sensors** | Definition of the sensors which data comes from which sensors with its technical properties | Aim of this table is to define which sensor is send which data i.e.  pH sensor – sensor id  Redox sensor-sensor id | s\_id,  s\_name,  s\_location,  s\_plcbrand,  s\_plcip,  s\_procolid,  s\_plcipnumber,  s\_plcgateway,  s\_plcport,  s\_plcscanrate |
| **TagDefinition** | Definition of the Tag means address of the PLC in which data comes from which address from the existing SCADA system | Aim of this table is to define address of the sensor data in PLCs which is running in the scada system. | Tag\_id,  Tag\_number,  Tag\_location,  Tag\_channel,  Tag\_device,  Tag\_name,  Tag\_address,  Tag\_datatype,  Tag\_accesstype,  Tag\_scanrate |
| **Rules** | This definition can be changed according to their requirements all controlable rules can be stored in this structure. | Aim of this table is to define in which case parameters will be controlled and what action will be taken and in which situation data will communicate with global structure and only active rules will be valid for this structure | Rule\_id,  Rule\_name,  Rule\_type,  Rule\_typenumber,  Rule\_Control1,  Rule\_Control2,  Rule\_action,  Rule\_actionnumber,  Rule\_active |

Global Structure

Global Database structure can be changed into one structure according to their requirements but must be one structure with communicates API in order to send data to Enterprise level.This structure will be message base in order to send data other relevant system like Enterprise Bus service and CEP engine.

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Purpose** | **Structure** |
| **SensorMeasureMessage** | This table will define structure of the sensor measure data. | Aim of this structure is to communicate with Global structure and Local ,Local data will be send by using this structure via API to the enterprise level. | Source,  Scope,  Temporality,  Value,  Type of measure,  Sensor ID,  Date,  Resource |

API service reads local data and sent by using Apache/Kafka platform with configuration written below.SSL security protocol is used.

security.protocol=SSL

ssl.keystore.password=xxxxxx

ssl.truststore.location=kafka.client.truststore.jks

bootstrap.servers=kafka.mantam.com.tr\:9093

ssl.truststore.password=xxxxx

value.serializer=org.apache.kafka.common.serialization.StringSerializer

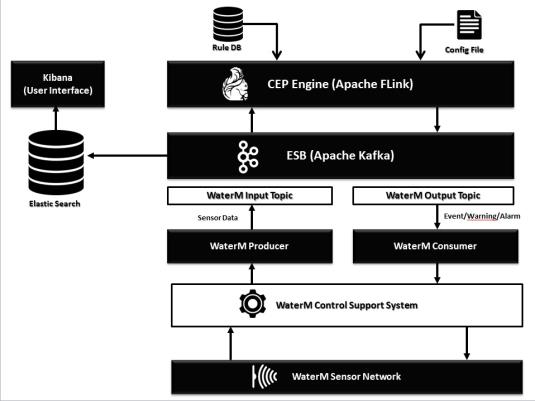
ssl.keystore.location=kafka.client.keystore.jks

key.serializer=org.apache.kafka.common.serialization.ByteArraySerializer

ssl.key.password=xxxxxx

### Data processing, analysing and modelling software’s

Water-M Analytics Platform is software which includes flexible, extensible, scalable, fast and high permanced, fault tolerance open source applications that has been accepted all around the world on big data processes.



Within the scope of project, Water-M Analytics Platform has been developed according to the functional architecture which is represented before.

On the functional architecture, All real time data has been collected by “ WaterMPrducer” which is defined on Apache Kafka that has been using as ESB ( Enterprise Service Bus ). and then the real time data is send to “WaterM Input Topic” which has all sensor data. This “WaterM Input Topic” provides concurrent multi access platform to share the data for any other services. All these data flow processes happen on “WaterM Sensor Network” layer.

Also these collected real time data is transmitted to CEP (Complex Event Processing) Module which are developed with Apache Flink. CEP Module processes the coming data packages according to rules which are determined by system user and produces necessary actions.

## Data processing and visualization methods

We are using Kibana to visualize the data in ElasticSearch. With easy connection between ElasticSearch and Kibana, Kibana creates dashboards, charts, tables etc. while using the data stored in ElasticSearch. Also this provides the near-real time monitoring for all data in WaterM system. Advantages of the Water-M Analytcis Platform System Architecture;

* Scalability : All system components of Mind4.0 can be run on any amount of distributed system.
* Low Latency : Mind4.0 has a capacity to process real-time very high volume of data with minimal delay.
* Fault Tolerant: With strong and intelligent architecture, Mind4.0 can provide minimum interruption on any system component failure.

High Availability: Mind4.0 uses durable hardwares and produce confidential software for all projects.

It is the visualization and analysis component which provides to search on the data stored in Water-M Analytics platform, to determine the results of analysis, to determine the data with the charts, to create dashboards and reports and these whole actions to be processed dinamically by users. Apache Kibana (<https://www.elastic.co/products/kibana>) has been used for visualization layer on the Water-M project.

Kibana is an open source analytics and visualization platform designed to work with Elasticsearch. Kibana can be used to search, view, and interact with data stored in Elasticsearch indices. It can be easily performed advanced data analysing and visualizing data in a variety of charts, tables, and maps.

Kibana makes it easy to understand large volumes of data. Its simple, browser-based interface enables users to quickly create and share dynamic dashboards that display changes to Elasticsearch queries in real time.

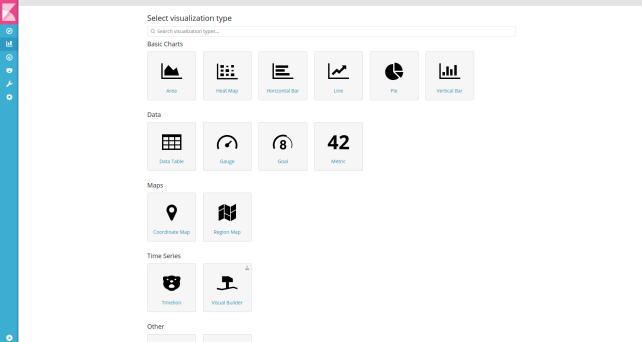
Water-M Big Data Analysis Platform's Visualization Service component provides the abilities below:

* It is possible to search, to display and to interact with the stored data in Water-M Big Data Storage via this component,
* It includes the enhanced data analysis instruments to visualise the data in various graphics, charts and maps.
* It enables the dinamic indicator boards which are able to display the revisions in index queries; to be formed and shared summarily via its interface on simple browser basis.
* It displays the search results' numbers and records detailedly.
* The temporal distribution of the records provided from the search results is able to be filtered in time by being displayed in a histogram bar.
* Relative time intervals are able to be descripted such as certain range between now and 30 min. 25 sec. before.
* Renewal intervals are able to be chosen to update the data on the visualisation interface. For example; updating the data in every 5 seconds.
* Search queries can be saved, then be reopened and also the searchings saved in the visualisation process can be used.
* Dinamic filters, such as is, is not, is one of, is not one of, is between, is not between, exsist, does not exsist can be used on search screen.
* It provides visualisation instruments in which the data can be analyzed by creating dinamic grap-hics on the selected data.
* It provides various graphics such as area, line, bar, pie, heat map, data board, data table, word cloud, time series etc. can be formed dinamically.
* It can create marks on the map bases for the areas which includes spatial information and also can form heat maps.
* Visualization can be used on bar and pie charts with the subrefractions. For example; first five message types produced in a management and first three devices producing these messages can appear on the same chart.
* Time sequence analyses and charts can be formed by using the data of distinct data resources.
* All of the images can be saved to be used for creating the dashboard later on.
* Dinamic dashboards can be formed by using the saved charts.
* The location and size of the dashboards formed before, can be personalized with drag-drop.
* Dinamic filtrations can be formed on dashboards by clicking on the related parts. For example in an ABC Business, the types of messages can be displayed immediately by clicking on the device which produce most number of messages.
* All graphics are automatically updated when any filter is applied from Dashboard.
* It provides to make analysis that based on Machine Learning techniques.
* It detect the abnormal data changes via applying the anomaly detection methods. With this way, system can warn the user for sensors which produce wrong data.
* It provides the user management and security services
* It provides the automated report system for all created analysis and gives the ability to export these report in many different format.

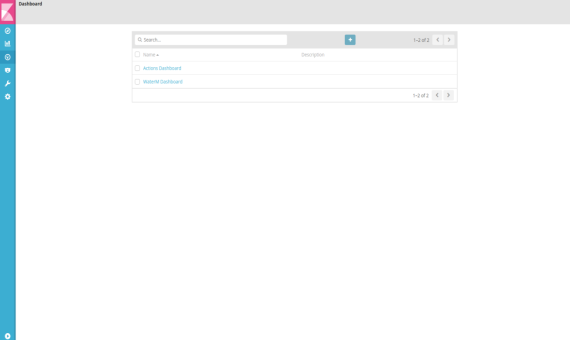
Screenshots From WaterM Analytics Patform



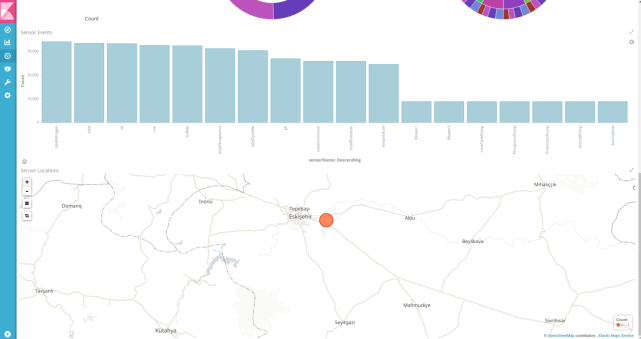
* Search Results For Sensor Events



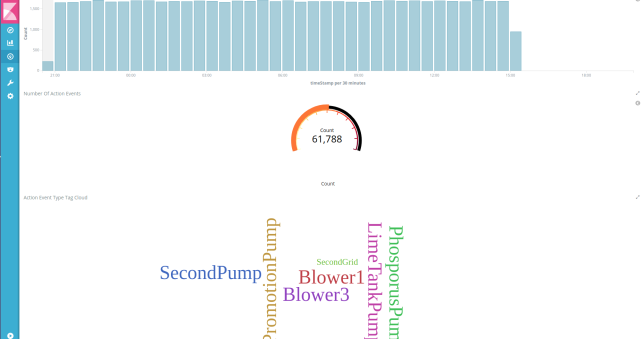
* Creating Visualization From Sensor Data



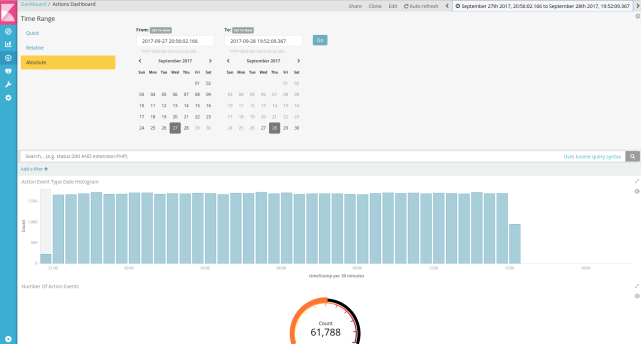
* Stored Dashboards For Water-M Analytics



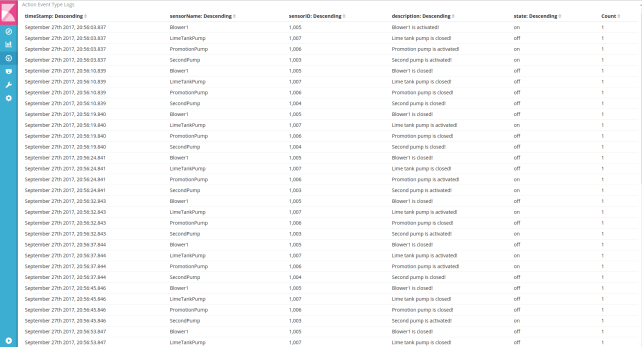
* Dashboard Graphs For Analysed Results



* Dashboard Graphs For Analysed Results



* Select Date/Time Range Screen



* Data Table For Analysed Results

## Execution of the pilot testing

### Description of the execution and building up the pilot test

During the Pilot test consortium use only existing sensors and scada system because management of the Plant could not let us to put new sensor thus we create rules and controlling only existing parameters. However parameters was not controlling online and no warning and alarms produced.

Control parameters have some minimum and maximum values these values can be changed by the end user depending on their request.

On the other side for all this process waste water system use some chemical materials these materials also taken under control and system will not let waste of these materials in terms of cost manner.

### General results

System gives alarms in case of unwanted situation

User can create their own rules depending on the request.

Materials used in the system taken under control in cost manner.

Cep engine system works 7/24 thus there is no human fault.

Invisible parameters of the system became more visible to give more effective decisions.

Flexible ,technological system makes more helpful for the industrial manner this system can be applied for new and legacy systems waste water plants.

Within this project also leakage detection and waste water plant simulation created.

## Main results of the piloting

One of the most important events in water distribution systems is that the water reaches the target correctly. This demonstrates the importance of leak detection in water distribution systems. The Reynold's coefficient is used to predict leak detection in the system. The Reynolds number is dimensionless and describes the ratio of inertial forces to viscous forces in a flowing fluid. It is used in many fluid flow correlations and describes the boundaries of fluid flow regimes (laminar, transitional and turbulent). The formula used to calculate the Reynold's coefficient is given below [5].

(1)

## Conclusions and Further Actions

System need more tested and comparing the results with the management of the plant.

# River Monitoring

## General description of the pilot-case

### Aim of the Pilot-Case

The primary goal of this use case is to develop an automatic system that can allow to continuously monitor the level and water temperature along a river and some of its tributaries.

According to a market research report [[5]](#footnote-5), the total value of global telemetry market was estimated at $109.54 Billion in 2014 and is expected to reach $243 billion by 2020, registering a CAGR of 14.20%.

With the alarming rise in water scarcity level, the global market for automatic water level controllers is experiencing high growth in its valuations.

Around 1.2 billion people, or almost one-fifth of the world's population, live in areas of physical scarcity, and 500 million people are approaching this situation. Another 1.6 billion people, or almost one-quarter of the world's population, face economic water shortage (where countries lack the necessary infrastructure to take water from rivers and aquifers). The world's population, face water shortage (where countries lack the infrastructure required to take water from rivers and aquifers).[[6]](#footnote-6)

The need to develop various methods without human intervention is fueling the demand for automatic water level controllers, which is also reflecting significantly on this market.

Beia Consult International has installed and is further developing for the National Administration “Romanian Waters” (ANAR) an automatic system able to continuously monitor the level and water temperature along the Danube and some of its tributary rivers.[[7]](#footnote-7)

### Actors involved and end-users

Actors involved are: Beia development team.

Table 1: End-users

|  |  |
| --- | --- |
| **Actor name** | **Actor responsibilities** |
| Administrator | Add/remove/edit the users access to the interface and maintain the system |
| Operators | Generate reports and are responsible for the maintenance of the web interface |
| Final users | Final users Receive the alarms generated by the system. (ANAR users ) |

### Description of the object for monitoring

**Danube river**

The Danube, flowing through 10 countries, is Western Europe’s largest river. Over a third of its total length of 2857 km flows through Romania, placing almost 98% of the country within the Danube Basin.

Not only is the Danube a major contributor to Romania’s water resources (44 %), it also is a major source of income through shipping and fishing, a carrier of wastewater discharges, a major contributor to irrigation schemes – and a permanent threat through flooding.[[8]](#footnote-8)

Water Monitoring on the Danube river is crucial for ensuring an efficient management of the river. The previous situation of monitoring efforts has resulted in experimental data logging equipment that does not fulfil requirements of reliable GSM transmission and sustainable power supply. [[9]](#footnote-9)



Figure 1. Danube River

## Hardware Devices and their installation

### Sensors and measuring devices

The type of sensors installed for river monitoring are presented in the next table.

Table 2 – Type of intalled sensors

|  |  |
| --- | --- |
| **Hardware** | Water level sensors;  Outdoor thermometer;  Potentiometric divider;  Rain gauge;  Inclinometer;  Opening sensor;  Power switch;  Pressure transducer;  Ultrasonic altimeter;  Temperature and Relative Humidity Sensor;  Rain Gauge sensors;  Dual-Radar Water Flow Monitoring System;  Multi-Parameter water quality sensor;  Remote Terminal Unit. |

The Remote Telemetry Unit (RTU) installed on-site are presented in the next figure.



Figure 2. Water tele-monitoring station

### Installation of sensors and measuring devices

The central elements of the water telemonitoring system are:

– Data concentrator (**Gateway**) performs communication with the remote telemetry units (RTUs) and also allows the configuration and management of all RTUs and sensors;

– Data presentation **server** is hosted on a computer with strong server features (such as safe unattended running 24/24 and 7/7).

### Powering of sensors

Key Performance Indicators (KPI) of the Beia sensors networks are presented in the table 3.

Table 3 - Beia Sensors Network KPIs

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Achieved Value** |
| Pressure range | Pressure range of water level sensor. | 0 - at least 200 mbar |
| Overpressure | Overpressure of water level sensor. | 1000 mbar |
| Operating temperature | Operating temperature of water level sensor. | 0 - + 55 ° C |
| Measured temperature range | Measured temperature range for outdoor thermometer. | - 40 ° C - + 50 ° C |
| Resolution | Resolution of rain gauge. | 0.2 mm |
| Accuracy | Accuracy of potentiometric divider. | 0.98% |
| Temperature stability ratio division | Temperature stability ratio division of potentiometric divider. | 0.998% |
| Measured inclinations range | Measured inclinations range of inclinometer. | 10 degrees |
| Resistance to physical shocks | Resistance to physical shocks of inclinometer. | 30g, 11 ms |
| Maximum permissible contacts | Maximum permissible contacts of the opening sensor. | 24 V / 5 W / 50 mA |
| Maximum permissible switch | Maximum permissible switch of the power switch. | 0 - at least 200 mbar |
| Measured pressure range | Measured pressure range of pressure transducer. | 1000 mbar |
| Maximum permissible overpressure | Maximum permissible overpressure of the pressure transducer. | 2000 mbar |
| DC Resistance | DC Resistance of the pressure transducer. | 3500 ohm |
| Measured depth range | Measured depth range of the ultrasonic altimeter. | 0 - more than 10 m |
| Measuring range | Measuring range of the humidity sensor. | 0 % ... 100 %. |
| Response Time | Response Time of the Temperature and Relative Humidity Sensor. | 1 sec |
| Capacity per Minute | Capacity per Minute of the Rain Gauge sensors. | 15 tips (1,6 resp. 3mm) |
| Measurement range flow speed | Measurement range flow speed of the Dual-Radar Water Flow Monitoring System. | 0,3 … 15 m/s |
| Memory | Memory of the Multi-Parameter water quality sensor. | 120.000 |
| Transmission Distance | Transmission Distance of the Remote Terminal Unit. | 20km (max. 12 miles) |
| Memory | Memory of the Remote Terminal Unit. | 32KB for up to 20.000 readings |

### Powering of the complete system

Each remote monitoring installation consists of an RTU, a water level and temperature sensor which is connected to the RTU through an atmospheric pressure relief box and the solar panel that powers the RTU and the sensor. A closer view of the A753 RTU, solar panel and pressure relief box (from up to down) is shown in Figure 3.



Figure 3. General structure of the water monitoring system

### Gateway

Data concentrator (Gateway) performs communication with the remote telemetry units (RTUs) and also allows the configuration and management of all RTUs and sensors.

The gateway ensures the communication with the non-GSM remote telemetry units and provides for resource management. Communication between the RTU and the Gateway is done through GSM-GPRS and Internet:

* If the telemetry unit is situated in a place with no GSM coverage, remote stations in the UHF 430-440MHzband will be used;
* The station communicates with the data concentrator (gateway) through a bridge station which ensures the data conversion for UHF to GPRS and GPRS to UHF.

The initially GW (Gateway) KPIs can be found in Table 4 below.

Table 4 - Gateway KPIs

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Achieved Value** |
| Dimensions | Dimensions of the gateway | 182 x 260x 52 mm |
| Connectors | Connectors of the gateway | 1 x 100Base-T Ethernet 1 x RS-232 (ext. modems) 1 x RS-232 (Console) 1 x RS-485 (Radio Modem) 2 x USB (ext. modems) 1 x Mains Power |
| Micro Processor | Micro Processor of the GW | Cirrus Logic 32-bit ARM |
| Memory | Memory of the gateway | 1GB Flash, 32MB RAM |
| Operating System | Operating System of the GW | Linux OS 2.4 Kernel |
| Operating Temperature | Operating Temp. of the GW | -10°C ... +55°C |

## Data Transfer Technologies Implemented in Pilot

### Description of the implemented data transfer protocols

Figure 4 shows the general structure of the water monitoring system.

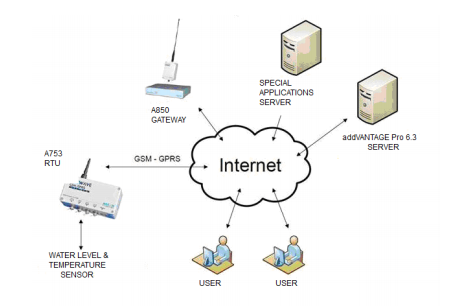


Figure 4. General structure of the water monitoring system

The RTUs network rely on a proprietary communication protocol named addUPI.

A typical session between a client and a server communicating over addUPI follows the steps described below:

1. The client discovers the server: the name of the server (or its IP address) must be known;

2. The client authenticates itself;

3. The client requests the server to return its capabilities: in this context, capability can be a list of nodes that are proxy-ed;

4. Inquire each node about its capabilities: this means getting the list of attributes and/or functions the node has and/or is able to execute (this step can be done simultaneously with step 3 in that the whole configuration of a node is requested, see “getconfig” on page 16 for more details);

5. Further, the client can act upon attributes (get/set), call functions or retrieve data (which is in fact also a particular case of a function call). Conversely, nodes may issue notifications on pending events to the client.

A request formulated by a client has a generic form as given below:

http://hostname:port/addUPI?function=fn&session-id=nnnn&param1=p1&param2=p2&...paramn=pn

where

• hostname is the address of the server

• port is the port number to the server listens upon (if not specified, it is standard 80; it is recommended to use the port 80 for Telemetry Gateways and 8080 for addVANTAGE servers)

• addUPI is the name of the handler on the server that will be invoked to handle the request; this may be a servlet, a cgi, etc.

• function is the name of the invoked method

• session-id is a number identifying a certain client (obtained after authentication)

• param*=pn are the parameters requested by a particular function.*

The response will return an XML document. There are two main ways to code the result: in plain ASCII, or binary (compressed).

## Measurements and data

### Parameters

Each RTU can monitor the following parameters, depending on the sensor’s configuration:

* Temperature;
* Air Humidity;
* Atmospheric pressure;
* Water level;
* Water temperature;
* Wind speed;
* Rainfall;
* Solar radiation.

### Locations of parameters (sensors)

At each monitoring location, a telemonitoring installation was built-up. It mainly consists of a wireless remote telemetry unit, always mounted on an aluminum mast, and a water level sensor. At some of the installations, the water level sensor is a pressure probe (Figure 6), immersed in water through a stably-mounted aluminum & plastic pipe.

The cable going through the pipe to the pressure probe always contains an air tube ensuring atmospheric pressure compensation. At other installations, a radar water level sensor (Figure 5) hanging above water surface was preferred. The pressure sensor immersed in water offers the advantage of always being accompanied by a water temperature sensor. The radar sensor has, in turn, the advantage of being immune to problems like being clogged by dirt, sand, ice and other such things.



Figure 5. Radar Water Sensor monitoring station



Figure 6. Pressure probe imerssed

### Measurement period, interval

All sensors attached to a particular RTU are powered and read during short time intervals.

### Example figures or other visualization

 "LiveData" is an application for addVANTAGE Pro 6.4. Measured data can be published on any website by simply connecting an RTU station to LiveData and embedding a frame on the desired page of the website. An easy to use navigation system allows browsing through the data of this station.

An example of the ”LiveData” implementation on BEIA’s website[[10]](#footnote-10) is available in Figure 7.

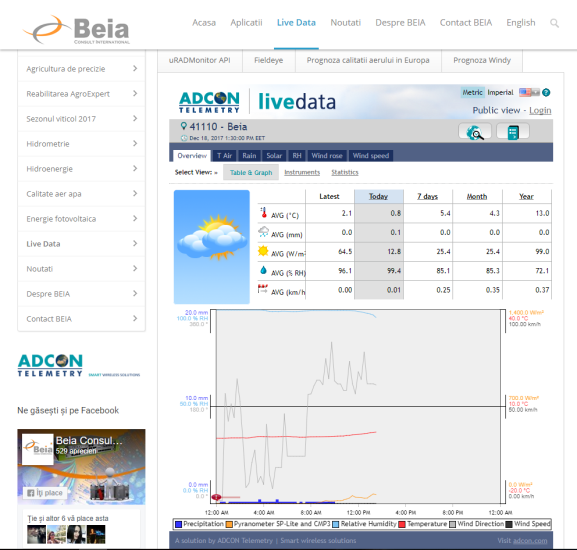


Figure 7. LiveData application

## Software Modules and Components

### Interfaces

addVANTAGE Pro is the web interface which allows the access for data visualization. Its main characteristics are: a browser-based, fully Internet-enabled,

* data visualization,
* data processing and data distribution platform,
* offering customizable trends, tables, statistics,
* alarms and events,
* disease and irrigation models,
* for all kinds of environmental and industrial data.

The interface is a software package to collect, store, process and graphically display data. The inherent flexibility of the software makes it an ideal tool for many applications, for weather and environmental data as much as for hydrographic, for leakage detection, frost warning and pump monitoring to countless other applications.   
addVANTAGE not only displays data graphically but also as text in a table, informs and/or alarms the user about specific events, provides several tools to process the data according to the needs of the clients, be it a simple report.

### Database

addVANTAGE Professional is a universal data visualization, processing, and distribution platform. It is fully web-based, runs on a fast and reliable PostgreSQL database engine, and is fully scalable from a single user version for 5 RTU's to a super server, serving thousands of clients and thousands of RTU's.

### Data processing, analysing and modelling software’s

At every 15 minutes, the RTU computes results an average value from a periodic measurement. At every hour, the four average values for each of the monitored parameters are sent by the RTU to the central gateway.

## Data processing and visualization methods

### Data pre-processing methods

The A850 Telemetry Gateway (shown in Figure 8) forms the heart of a water monitoring system. The A850 acts as a **mid-term storage device for data collected** from UHF radio and mobile phone (GPRS) based remote telemetry units (RTUs).



Figure 8. A850 Gateway

The unit is built around an embedded Linux operating system for reliable 24/7 operation. An integrated UPS ensures interruption-free operation throughout power outages.

### Data analysing methods

The server analyzes the measured data, and through some predefined scenarios allows to verify if the measured data corresponds to a registered pattern, or it registered an anomaly measurement.

### Data visualization

Data presentation server is hosted on a computer with reliable server features (such as safe unattended running 24/24 and 7/7. Server software is mainly focused on the presentation of data in various formats (tables and diagrams, for instance), which are entirely at the free choice of users.

Changing data presentation formats is however allowed to those users only who have the right to make such changes; a very sophisticated rights hierarchy system does exist in the software.

Figure 9 and Figure 10 shows several data diagrams from 2 site locations.

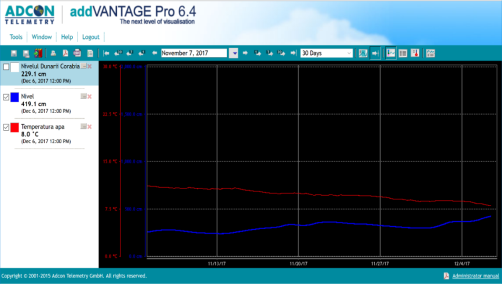


Figure 9. Water level (Probe sensor)

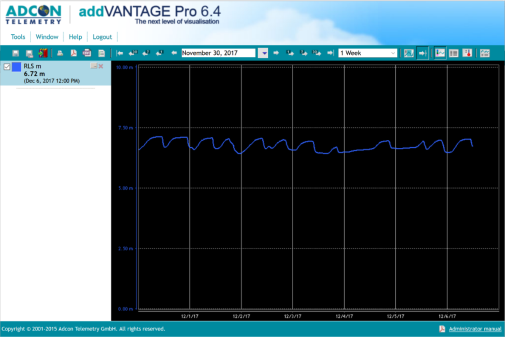


Figure 10. Water level (Radar sensor)

## Execution of the pilot testing

### Description of the execution and building up the pilot test

A server for a central database has been hosted on a cloud platform which delivers elastic resources (CPU, storage, memory, etc.) and is equipped with software focused mainly on data presented in various forms, entirely available to users, but also on particular tasks (data mining, etc.).

Table 5 - BeiaIoT Server specifications

|  |  |
| --- | --- |
| Hardware | * 2 vCPU 2GHz E5-2620 * 8GB DDR3 * 50GB HDD 10K |
| Software | * VMware ESXi 5.5.0 * Microsoft Windows 7 64-bit * Database: PostgreSQL |

The initially specified KPIs and the finally achieved KPIs can be found in the table below.

Table 6 - BeiaIoT Server KPIs

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Description** | **Specified Value** | **Achieved Value** |
| Number of RPS (Requests per Second) | The average number of requests to be processed per second. | 500 | 500 |
| Error Rates | Naturally some errors may occur when processing requests, especially when under a big load. | Less 1% | 0.95% |
| ART (Average Response Time) | Time that the application use to generate a response. | 0.5 sec | 0.5 sec |
| Uptime | The amount of time that a server has stayed up and running properly. | 99% | 99% |
| Max Live Sensors | Maximum number of sensors that connected to Beia IoT Server and populates data. | 500-1000 sensors | 500-1000 sensors |
| Availability | Percentage of actual uptime (in hours) of equipment relative to the total numbers of planned uptime (in hours). | %95 | %95 |

### General results

Figure 11 below shows the remote monitoring stations installed across the Danube River, and the most western can be found at Lugoj (on the Timis river), while the most eastern finds itself on the Danube’s Chilia arm, not far from the Black Sea.

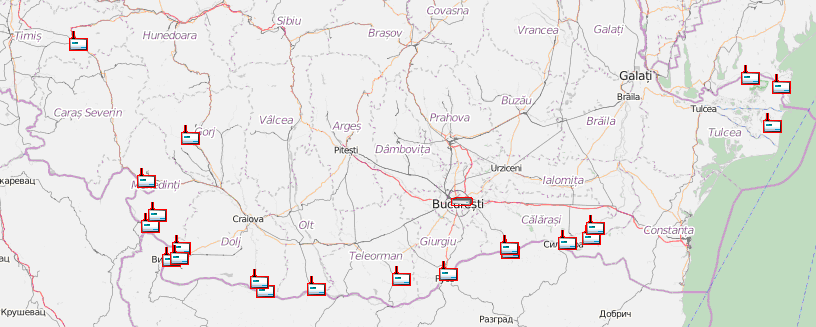


Figure 11. Remote monitoring stations across the Danube River

## Main results of the piloting

### Malfunction Detection

The primary value that the pressure probe water level sensor delivers is the height of water column above the sensor. When the water level sensor is radar type, the primary value is the distance between radar and water surface. In both cases, the system allows alignment calculations, so that what is being finally delivered is the water level officially recognized by the “Romanian Waters” National Administration.

## Conclusions and Further Actions

The system was piloted for 3 months and data was gathered for visualization and further processing. No anomalies were detected, as there have not been any severe weather or environmental changes (heavy rain, drought, etc.)

# Urban Farming

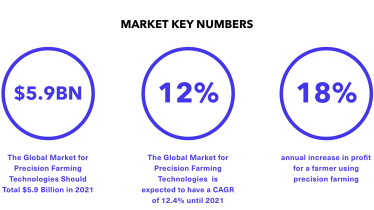
## General description of the pilot-case

### Aim of the Pilot-Case

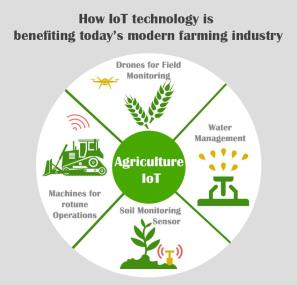
**To develop a prototype: sensors + server + datavisualization with : LoRa, Warp10, Grafana, and SEAS ontology.**

The entire system is declined on a "Smart indoor farming" module: automated cultivation of plantations, monitoring of surrounding environment, management of water, light, and nutriment supply.

Precision Farming will be a significant market:



Both outdoor and Indoor farming need Water Management + Soil Monitoring.



This present pilot will be initially focus on indoor farming with both Soil monitoring and water Management question. Along with Evolution Energie, a WaterM partner, we will also consider the electricity cost Managment. Indeed for the Indoor type of farming this OpEX cost management is critical.

Our focus is to demonstrate the WARP open source Big data platform scalability and capacity to run in very memory and CPU constraints factors for indoor precision farming.

So in End 2017, we start a initial indoor farming demonstration that was been demonstrate to a paris event at the La Villette ( A). During the summer 2017, we downsized the indoor system with water management within a Primary School of a south west paris Region local authorities (LePerray-en-Yvelines (B)).



After discussions many local authorities around ask us to propose a BigData / iot pedagogical kit at a very **reasonable pricing acquisition cost** for being able to deploy in many sites. In primary and secondary schools but also in Higher Education. So we work on R&D for downsizing solutions to a under 500 € pedagogical kit ( possible due to low price IKEA Small Indoor farming solution that occur on the market H2 2017 ).

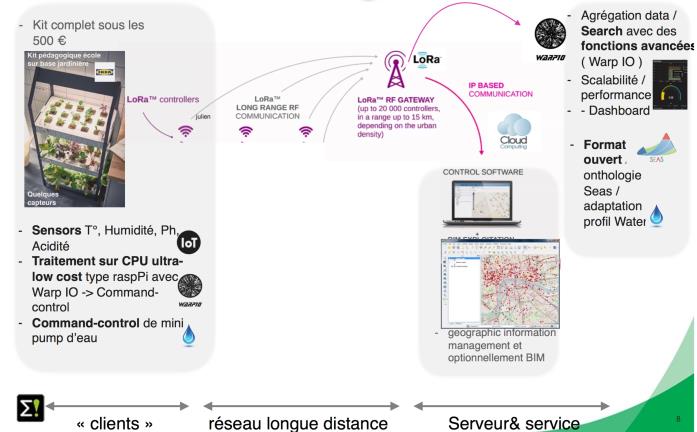
|  |
| --- |
|  |

A second consideration during H2 2017, we to reduce also the IoT Data collect solution for both Water and Soil data acquisition but also Water actionner and light management. For IoT data collection, we choose the **LoRA protocol** that could be deployed on a specific territory without Telco licence and opex cost problematics. The LoRA gateway pricing resquest was also should be under 500 € for the complete pedagogical kit.

A third request, was to integer dashboards that will enable supervision for various Kit deployment within territories, so we work on an additional modules. So

1. We developed a first entry level solution with an open source GIS solution, “open street Map”.
2. We provide APIs and data to waterM partner ( Evolution Energie could the data and make treatments and could provide their professionnel information system solution.

The under diagram expose the global aim of our UC contribution.



A technical aim was also to propose open data format for Water Management and energy questions (métadata, …). For that, considering the existing Data format and data frameworks from ITEA3 SEAS project and HayStack open initiative, we did implement those data format within open source Warp IO platform.

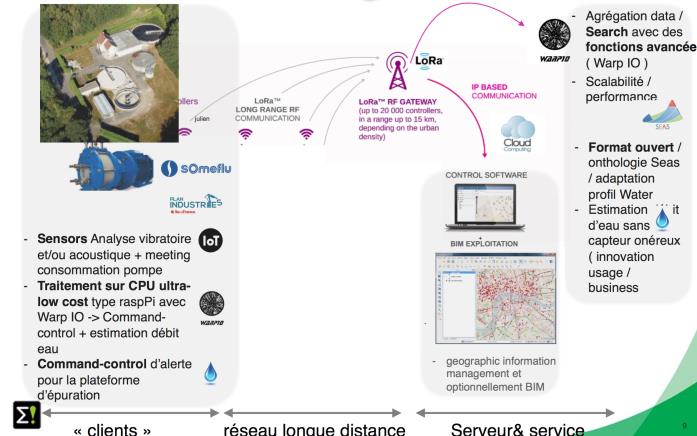
**2018 pilot**

After those pilots, the lePerray Local authorities request cityzen Data to consider an adaptation of the Cityzen data WaterM data acquisition and IOT solutions for their local Water treatment infrastructure

|  |
| --- |
|  |

That adaptation and deployment will take place for a Sept 2018 timing. A water-pump partner will integer those technologies and initial WaterM UC results.

Under the diagram

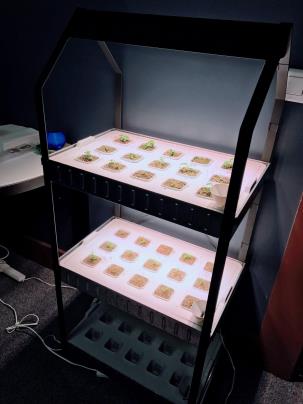


WaterM partner water quality sensor could then be deployed.

For short term aim, we describes under the mains Pedagogical kit components

|  |
| --- |
|  |

The IKEA packaging is well suited for education distribution.



Objective: Record the plantation environment variables: Brightness, Humidity, Temperature, Localisation.

Sensors are connected to a Raspberry Pi, sending data via HTTP protocol and LoRa protocol.

All the recorded data are sent by the Raspberry to the centralized server (API nodeJS).



### Actors involved and end-users

The actors involved are:

CitizenData, Warp10 development teams,

with its WARP 10 open source solution Capture d’écran 2017-11-03 à 09.59.43.png

## Hardware Devices and their installation approaches

### Sensors and measuring devices

Installed environmental monitoring sensors:

* Temperature sensor
* Brightness sensor
* Humidity sensor
* Location sensor:
* Raspberry Pi LoRa/GPS HAT card - 868M frequency support
* A GPS/Lora sensor ( for Kit tracking position among various public places ( the kit is easily movable ).

### Actuator (actionneurs )

* Water acruator within small water pump



* On-off relay for on-off of electricity feed for light management



### Installation of sensors and measuring devices

The environmental sensors are connected to a first Raspberry Pi 3 system via GPIO connectors.



The sensors are placed on the metallic strcture of the system.

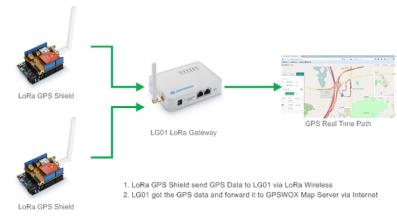


**Figure 1 - Light sensor**

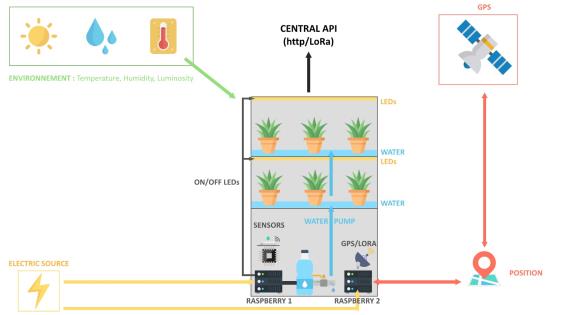


**Figure 2 - Humidity sensor**

The GPS/Lora sensor is connected to a second raspberry Pi 3 via the GPIO connectors.



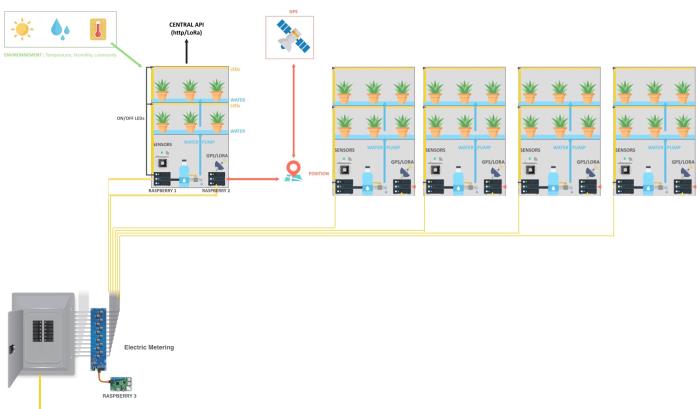
**Figure 2 - GPS LORA module and principle**

* GPS X,Y data collect will eanble to position each of the Indoor MicroFarming solution. 

### Powering of sensors

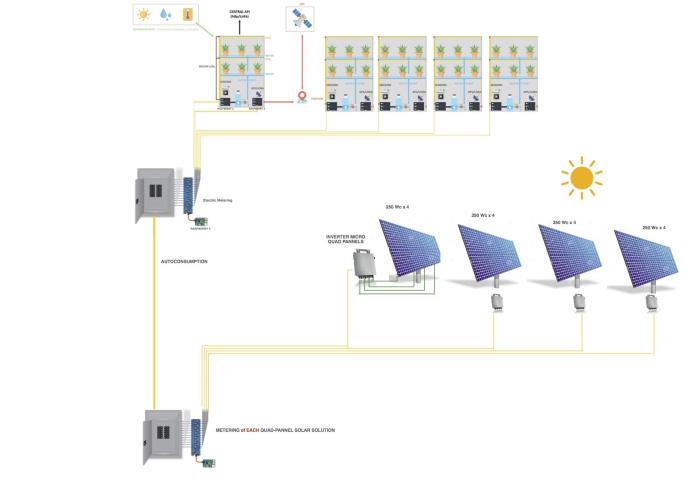
The systems are connected to the raspberry Pi 3

### Powering of the complete system

We consider that the targeted possible market ( micro pousse aka Micro Shoot which are above 60 € per Kg ) need several indoor Gardening system.

SubMetering Sensors will be used for the pilot. 12 subMetering mesures could be perform.

With associated LED ernegy feeded, the electricity cost could be a OpeX factor declining rentability. For solving that, a Solar Panel Electric Panels CaPEX could be consider.



Today PV Capex reduce price + Micro-Inverter reduce price tendency enable, in Germany, Swiss, Italie, a return investment price based on the actual grid electric price ( above 200 €:Mwh ) is under 8 to 6 years.

The deployment phase will use Micro-Inverter managing each 4 solar panels ( 300 W each ), so the above schema is a 4 PV x 300 watt x 4 systems= 4,8 KWpeak solution.



Figure = Micro -inverter managing 4 solar panels

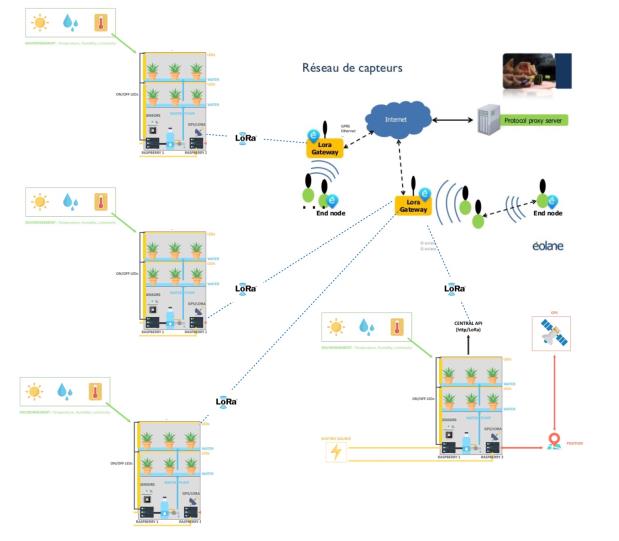
That could feed hundreds of Square meter of Indoor Gardening needs. Dropping Battery price, 14 kWh powerwall tesla at 7 K€, or 5 kWh ion lithium 2 k€ battery give also solution for high cost grid markets ( Germany, Danemark are above 280 €:MWh ). Then 24h:Day indoor gardening solution could be pursue.

### LoRA gateway solution

The LoRA GateWAY solution is compose

* the LoRA station telecom component
* a Local Network server ( EOLANE solution ) and / or distance LoRA Netwrok server
* a GSM data backend module and / or a Ethernet and:or wifi solution

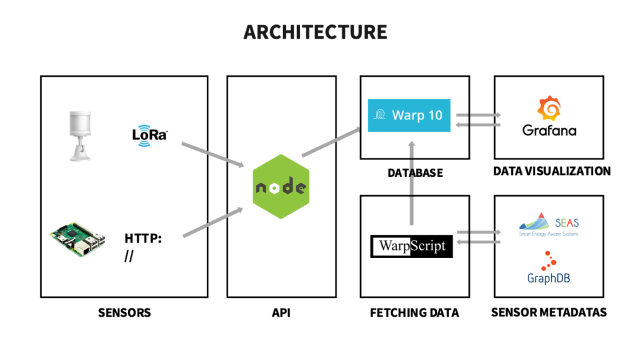
|  |
| --- |
|  |

* GPS X,Y data collect + LoRA Transmission will server for production localisation of hundreds of micro / Nano Production sites within cities or suburban zones without Telco OpEX costs.

## Data Transfer Technologies Implemented in Pilot

### Description of the implemented data transfer protocols

General architecture:



## Measurements and data

### Parameters

* Humidity: Measures the percentage of humidity in the air using a DHT11 sensor connected to a raspberry Pi. It is placed as close as possible to the plants. Check the condition of the air (too humid, not wet enough) => Monitor the perfect humidity for plants, Returns a float
* Temperature: Measures the air temperature (°C). Located closest to the plants. Checks the air temperature => Monitor the heat input, Returns a float
* Brightness: Measures the luminous intensity in a binary way (0 or 1). Check light intensity => Monitor day/night cycles of plants
* GPS: Retrieve the geographical position of the system, Positioned at the bottom of the system to not generate plant growth, Transmitted by LoRa network to the server => Monitor the position of the sensor, returns latitude/longitude

### Locations of parameters (sensors)

All sensors are placed on the metal hydroponic structure.

### Measurement period, interval

Variable measurement period depending on the sensor.

* Humidity: 1 measurement per minute
* Temperature: 1 measurement per minute
* Brightness: 1 bar / 30 seconds
* GPS: 1 measurement per day

### Example figures or other visualization

Sample of data in Warp10 database :

1510059478842000// sensor{id=2,.app=io.warp10.bootstrap} 42

=1510059473813000// 42

=1510059461225000// 42

=1510059448626000// 39

=1510059436035000// 41

=1510059431013000// 42

1510059488130000// sensor{id=42,.app=io.warp10.bootstrap} 42

=1510058846082000// 22

=1510058845903000// 22

=1510058845732000// 22

=1510058845491000// 22

=1510058844968000// 22

=1510058569822000// 22

1510059486381000// sensor{id=1,.app=io.warp10.bootstrap} 21

=1510059476275000// 21

=1510059468700000// 21

=1510059456128000// 21

=1510059446074000// 21

=1510059438546000// 21

## Software Modules and Components

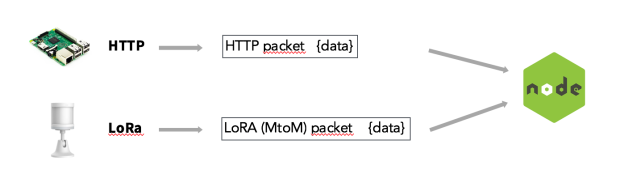
### Interfaces

Interface : How data are gathered

Internet of things sensors ("IoT"): reading of information and sending on the NodeJS API.

1) Sending data in HTTP

2) Sending MtoM network data: LoRa (See https://www.lora-alliance.org)



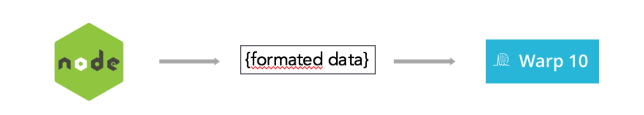
Software interface : Centralized API

Centralized server.

Technology: Node. JS + Express framework

Receives sensor data and formats a WARP10 compatible frame.

Sends the frame to the WARP10 database.



### Open source Database

Capture d’écran 2017-11-03 à 09.59.43.png

WARP10 DATABASE

Geo Time Series Engine ("GTS").

The recorded data are stored and time stamped + localized (timestamp + geolocation)

See <http://www.warp10.io>

And <https://www.warp10.io/apis/gts-input-format>

Structure, data sample :

TIMESTAMP/LAT:LON/ELEV TABLENAME{LABELS} VALUE

1510662893/48.813995:2.392448/10.22 sensors{id=0} 1.45

[…]

### Data processing, analysing and modelling software’s

../Desktop/Capture%20d’écran%202017-12-04%20à%2016.26.59.png

WARPSCRIPT

Requesting language on the Warp10 layer.

2 objectives:

The first: add a description of the sensors in the Warp Engine to perform searches based on SEAS ontology.

See the warpscript documentation for macros. (see GitHub rendering)

Example of a description of the brightness sensor inserted via a macro in the Warp database:

Structure, data sample :

[…]

'0' { 'type' 'photosensor' 'label' 'analog photosensor' 'unit' 'bool' 'lat' '48.813917' 'long' '2.392437' }

## Data processing and visualization methods

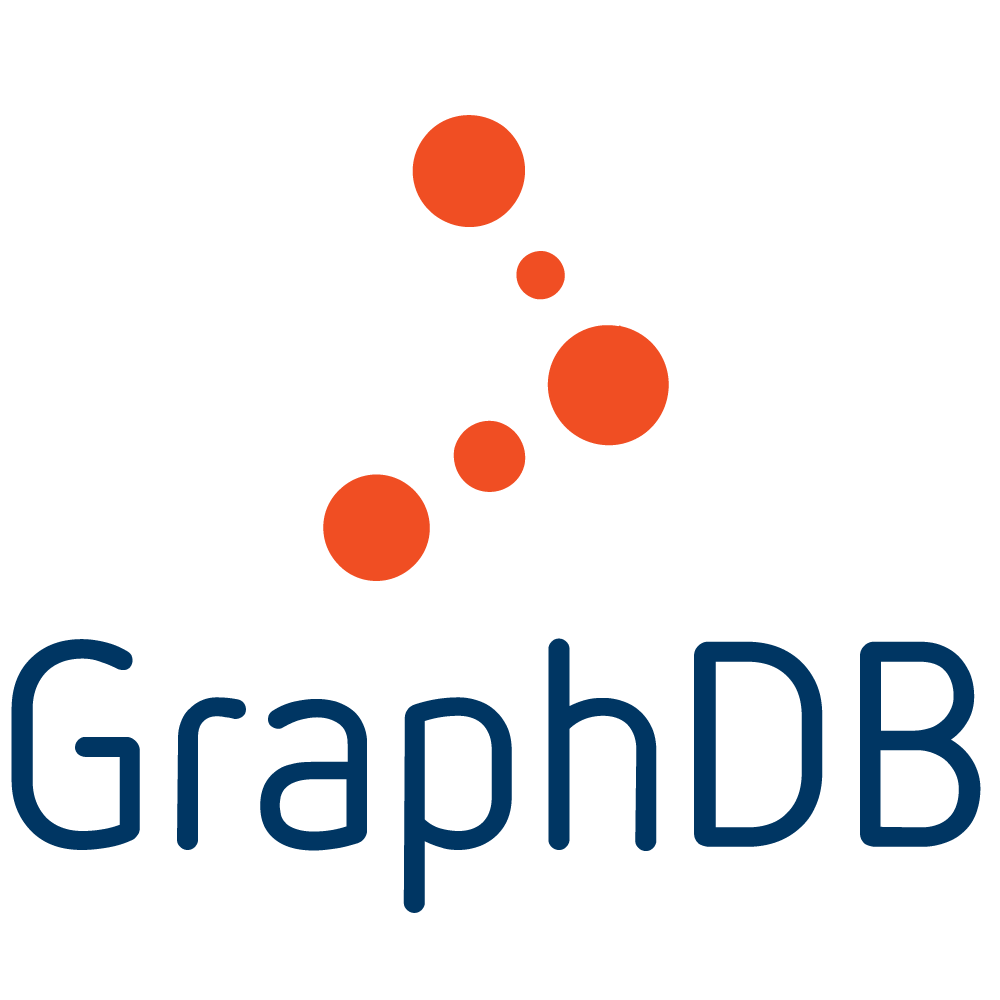
### Data pre-processing methods

Via Waprscript: search, filter data. Searches and sorting are indeed carried out via WarpScript queries in the Grafana tool: the data we are interested in is then retrieved, with the possibility of sorting according to the elements inserted in the macro.

### Data analysing methods

The measured data is analyzed by the central server. Scenarios are set up on the API. If the analyzed data corresponds to a registered pattern (e. g.: sudden change of brightness, temperature), the server communicates with the sensor (Raspberry) and activates a response (e. g.: switch on the LEDs of the hydroponics system)

### Modelling methods

logo.png

Ontology SEAS

The sensors are described according to SEAS ontology.

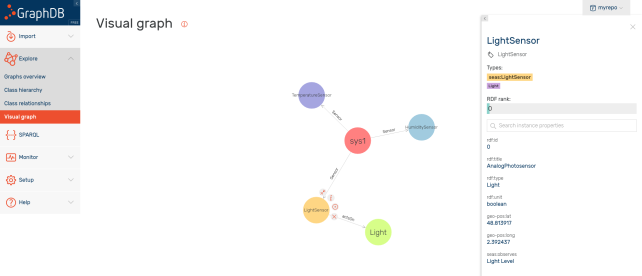
The visualization of the characteristics and relationships between the sensors is done using GraphDB.

See: <https://w3id.org/seas/>

See: <http://graphdb.ontotext.com>



You can see all sensors, and their relations with GraphDB :



### Data visualization time series

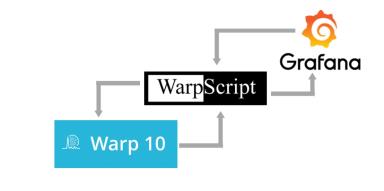
Data visualization engine. Interface with the « Grafana-Warp10 module ».

See: <http://grafana.com>

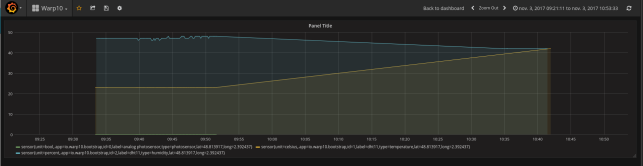
See: <https://github.com/cityzendata/grafana-warp10>

Allows visualization of all collected datas by the sensors.

Modular queries thru Warpscript in Grafana console.

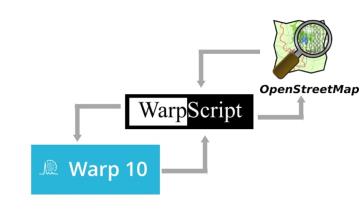


Grafana visualization :



### Geographic Data visualization

GPS Sensor Lora transmission positioning and other sensors positioning will use open street Map framework solution for data visualization.



## Execution of the pilot testing

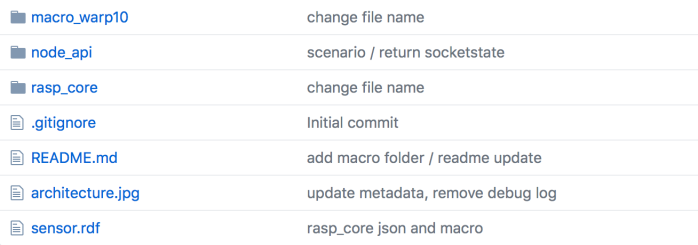
### Description of the execution and building up the pilot test

All sources are available on :

<https://github.com/etna-alternance/Co-Labs-WaterM.git>

There is 2 folders :

* Rasp\_core\_macro contains the raspberry pi code to manage all sensors as described in this document
* Node\_api contains central server API in nodeJS
* Macro\_warp10 contains the macro file ”macro.mc2” to include in the warp10 engine to register the sensors metadatas.



How to install (Described in README.md):

Install NodeJS

*Arch Linux*

pacman -S nodejs npm

*Ubuntu & Debian*

sudo apt-get install -y nodejs

*Windows & MacOS*

HYPERLINK "https://nodejs.org/en/"Download link

Install source

*Install*

git clone HYPERLINK "https://github.com/etna-alternance/Co-Labs-WaterM.git"https://github.com/etna-alternance/Co-Labs-WaterM.git

cd Co-Labs-WaterM

npm install

*Run*

**First you need to setup config.js and sensor.json**

*Config.js*

// Write/Read token Warp10

exports.write\_token = "WRITE\_TOKEN"

exports.read\_token = "READ\_TOKEN"

// Warp10 ip/port

exports.Warp10\_port = "WARP10\_PORT"

exports.Warp10\_hostname = "WARP10\_HOSTNAME"

//Metadata config file

exports.config = "PATH\_TO\_CONFIG\_FILE"

In source folder

node .

### General results

The sensors send the data continuously to the central server.

The central server analyzes the data, triggers parameter scenarios if the results match registered patterns.

The sensors are described using SEAS ontology. Their description is embedded in a macro in the engine of the warp10 database.

Thus, we can perform Warpscript queries to obtain data using the metadata of each sensor.

The visualization of the sensor descriptions is done via GraphDH.

Data visualization is done via Grafana and its Grafana-WarpDB module.

## Main results of the piloting

The results for the prototype are as follows:

The data is well collected and fed back to a centralized system that links the data with the corresponding sensor.

The system makes it possible to perform intelligent actions thanks to scenarios programmed for this specific system (use case hydroponics/culture).

The system allows data visualization.

### Dynamic Limit Values

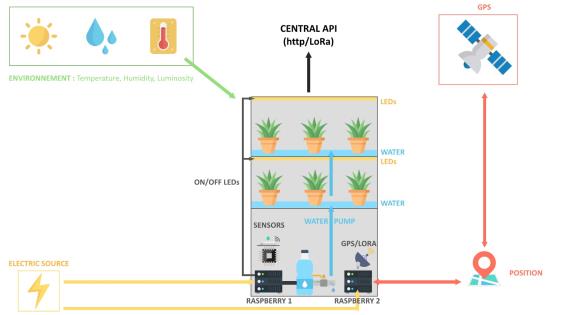
The limitations of the system are as follows:

Scenarios are statically programmed on the central server. They are further developed to interact specifically with our Raspberry transmitter to manage the light and water input of the system.

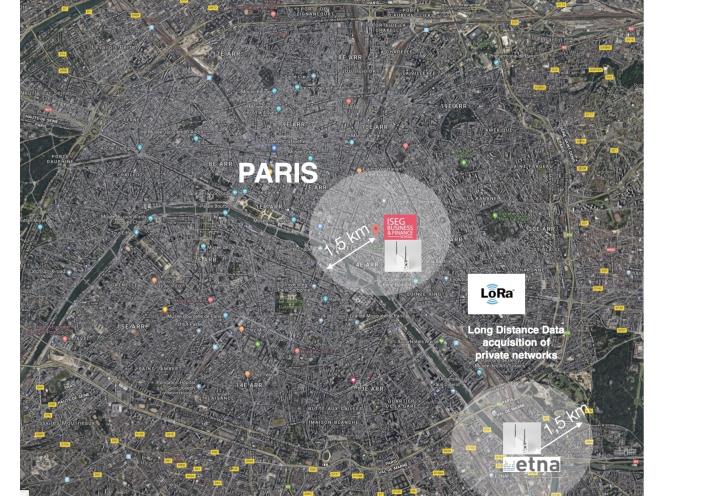
For a more complex and larger system, it would be necessary to review how to interact with the sensors in return, and on what level of architecture to place the data analysis and scenario code.

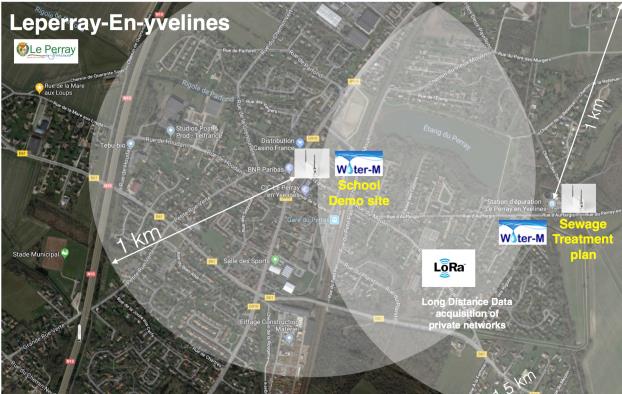
## Demos

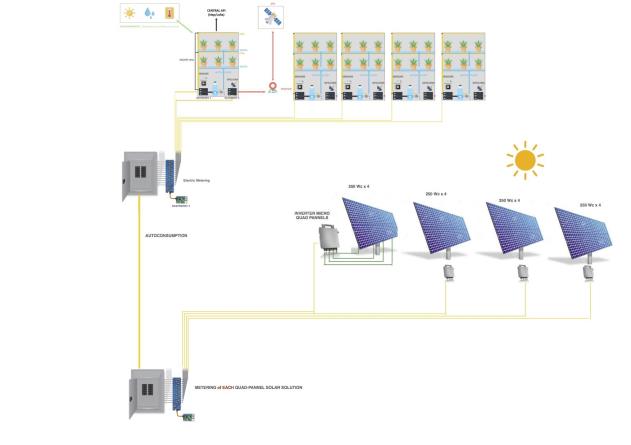
For the january 2018 WaterM démo , two of the under pedagogical system will be deployed

* 

Two pedagogical Kits in Paris and Surrounding , ISEG 28 Rue des Francs Bourgeois, 75003 Paris and ETNA engineering School in Ivry sur Seine



On pedagogical Kit in a School + one scaled version, in the sewage treatment plan in south west of Paris Region place, called Leperray en YvelinesThe Leperray-Water-Sewage treatment plant has two mid sized greenhouse Building in each will deployed the following solution the autocomsumption energy site system



One advantage of the Sewage water Treatment sites is that some local treatment generate CO2 that is nicely used by vegetables growth.



## Conclusions and Further Actions

The use of WarpIO technologies, LoRA MtoM sensors or conventional HTTP sensors, as well as the ontology SEAS, waterM specific Onthology and open source HAYSTACK format in a single package is quite feasible.

The synergy of these elements allows an efficient management of a set of sensors and an extremely fast visualization on large quantities of data.

1. Scenario management, however, needs to be further researched, as no software layer currently stands as an interface between the received data and the return to sensors.
2. In addition, the integration of sensor descriptions (metadata) is quite complex if you want to gain in performance by including them directly in a macro to pre-process the data in the Warp engine.

# Hierarchical description of meter network

## General description of the pilot-case

### Aim of the Pilot-Case

The aim of this document is to present a specific data collection method from a given data source which could be embedded in small and portable systems. This data is pushed towards a Data Center deployed in a different location, via the network. In other words, the idea is to collect raw data and pass it from a rough DB to an SQL DB to make them available for a suitable and user-friendly Interface. A *Raspberry Pi* is employed as bridge between raw data and the Data Center. As source, we use a device called *Spoony* which let to measure the energy consumption in *Evolution Energie* premises. Moreover, we want to visualize and manage this data in our software, *FLEXINERGY*. The connectors and the methods presented in the sections that follow are implemented within the *Water-M project*. The *FLEXINERGY* solution is the *Evolution Energie* software, which is not implemented or coded for this project, but is already widely used as user-interface for different costumers.

### Actors involved and end-users

The actors involved is the Evolution Energie team.

## Hardware Devices and their installation approaches

### Sensors and measuring devices

The objective presented in this document has been reached by making use of the following tools:

* Energy-Gateway Spoony:



* Raspberry PI 3 (Model B, 1GB RAM memory, Debian OS which is a Linux like OS)



* Database cache and message broker REDIS
* Client Web service in JAVA 7 for input /output data: RPiConnectorIN.jar and RPiConnectorOUT.jar.
* SQL server as Data Center.

### Installation of sensors and measuring devices

### Powering of sensors

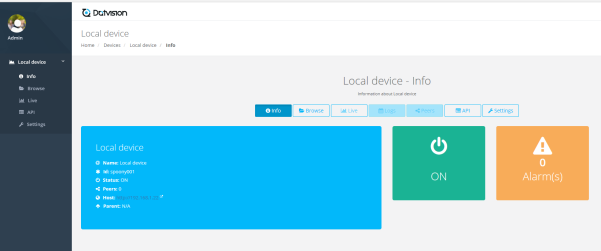
Spoony and Raspberry equipment require a normal 220V power supply.

## Data Transfer Technologies Implemented in Pilot

### Description of the implemented data transfer protocols

For our preliminary tests, we use an Energy-Gateway, called **Spoony**, which is a product of the company *DotVision* (<https://energy.dotvision.com/Products/Spoony>).

The Spoony is a power measuring device for single or three phase installations. In our case, it is a three-phase installation. The Spoony measures the power consumption and sends data to a server with a restful API. The user interface looks as follow.



Through the user interface we could visualize the real-time data, but we cannot store it. The server does not manage the historical data.

The data coming from the Spoony are temporary stored in a **Raspberry** **Pi**. The Raspberry (<https://www.raspberrypi.org/help/faqs/#introWhatIs>) is a capable little computer which can be used in electronics projects, spreadsheets, word processing, browsing internet, or playing games. The Raspberry Pi is slower than a normal computer but is still a complete Linux computer and can provide all the expected abilities that implies, at a low-power consumption level. We use a Raspberry Pi 3, Model B with 1GB of RAM memory, this version measures 85.60mm x 56mm x 21mm. It weighs 45g.



Furnished of an alimentation cable, an HDMI cable, 4 USB ports and an Ethernet cable. A 16GB microSD card is used as well. A Windows 10 IoT Core could be installed on the SD card. However, it does not include the user interface or the desktop operating system. This is an entirely new version of the operating system designed exclusively for embedded use and is not recommended for our purposes. Therefore, we use Raspbian as OS, which is specifically designed for the Raspberry Pi. It’s based-on Debian, a Unix-like computer operating system.

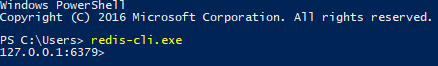


To temporary store the raw data coming from the Spoony, we deployed **REDIS** (<https://redis.io/>) on the Raspberry.

The word REDIS stands for REmote DIctionary Server. It is an open source in-memory software which stores data structure. It is used as a database, cache and message broker. It supports data structures such as strings, maps, lists, sets, sorted sets.

Developed in C under open-source, REDIS was initially released under Linux. Windows version has been released by MS Open Technology. REDIS supports the ARM processor in general, and the RPi specifically.

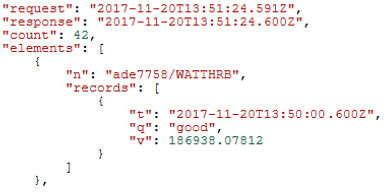
With simple installation, easily customizable via *.conf* file in installation folder, REDIS comes with a batch client *redis-cli.exe* that let a quick use :



We employ REDIS more like a key-value type database management system. With the ”SET” and ”GET” commands we could obtain a fast write/read operations: good for storing fast-updating and simple structured/unstructured data. As REDIS keeps all data in RAM memory, it avoids disk access and achieves excellent performance. Regular clean of the DB with the FLUSHALL command must be performed.

Two Connectors**, RPiConnectorIN** and **RPiConnectorOUT**, are also deployed on the Raspberry under .jar format. They are employed to implement the data transfert between Client DB (the Spoony in this case) and local DB. A connector is a specific code, in this case written in JAVA 7.

The **connector IN** is employed to get the data from the Spoony and set them into REDIS database. The data coming from the Spoony are in *json* format. They have the following structure:



|  |  |
| --- | --- |
| “n” | This filed of the *json* stands for the unique name of meter |
| “t” | In this filed we can found the date and the exact time in which the value is taken. The time is expressed in UTC. |
| “v” | This field specifies the value of the meter. |

The Java code uses the Maven library *org.apache.httpcomponents* in order to connect to the Spoony, the *org.jason* library to read the json and the *jedis* library is used to write in REDIS DB.

A configuration file, in xml format, defines the main features needed for the collection of the data:

|  |  |
| --- | --- |
| <InterfaceName> | This is the name of the interface. |
| <LogLevel> | 5 options are possible: **DEBUG, INFO, WARN, ERROR, FATAL**. By default, please leave this parameter set to INFO. |
| <spoonyUrl> | This field specifies the web address where retrieve the data coming from the Spoony. |
| <redisIp> | In this field, the host which identifies the local machine where REDIS is installed. |
| <meter> | This field indicates the list of the meters that the connector takes into account from the Spoony. It has different sub-fields listed below. |
| extIdSpoony | This is the unique ID as it appears to into the Spoony. |
| extIdFlex | This field represents the unique ID as we want it appears in our solution. If it is left empty then the *estIdSpoony* is taken by default. |
| m, d | These two fields are conversion factor that could be applied to the meter data value. To sake of simplicity the letter m indicates a multiplication factor as well as the letter d indicates a division factor. If one or both are left empty, 1 is taken as default value. |
| isIndex | This is a Boolean, true or false, which indicates if the meter data values are computed as delta between two different date/time. It is false by default or if the field is left empty |

The data are downloaded with a Rest library. The Meter name, the date and the values are stored in REDIS under the key-value format:



The key is made by the meter name (e.g. ade7758/WATTHRB) and the date-time (e.g. 2017-11-20T13:50:00.600Z). The value is the value of that meter at that time (e.g. 186938.07812). Key and value are inserted with the command SET and they are separated by an empty space during their insertion.

The **connector OUT** is used to get the key and values set in REDIS, to arrange them in a suitable format and then to push them to the SQL DB (Data Base). The corresponding command in the redis-client.exe batch is:

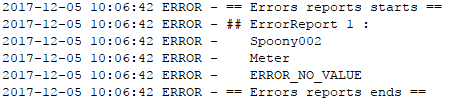


A configuration file, in xml format, defines the main features needed for this connector:

|  |  |
| --- | --- |
| <InterfaceName> | This is the name of the interface. |
| <LogLevel> | 5 options are possible: **DEBUG, INFO, WARN, ERROR, FATAL**. By default, please leave this parameter set to INFO. |
| <endpoint> | This is the address which let to use the *WebService* for store data in SQL DB. |
| <redisIp> | In this field, the host which identifies the local machine where REDIS is installed. |

The *jedis* library is used to get data from REDIS. A Web Service implemented by Evolution Energie outside the Water-M project is employed to send the data outside the Raspberry Pi, towards the SQL DB.

It worth to be notice that the characteristics of the meters (i.e. Meters Name, frequency or Units) should be previously inserted in the SQL DB. The task of the connector out is to fill the meters with their data at a specific time. An error could be occurring during data insertion in SQL DB. For example, we are trying to insert some data that not correspond to a meter already present in the database. In this case an error is sent back to the Connector OUT. The error points out the name of the meter whose data are not stored in the SQL DB. For example, the following error could appear:

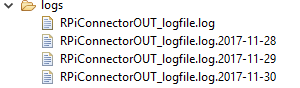


Otherwise, if all the collected data from the Spoony are all correctly inserted in the local database, the REDIS DB is cleaned. The corresponding command in the redis-client.exe batch is:

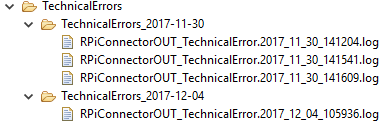


Both connectors IN and OUT are furnished with specific error logs to follow the result (errors or not) obtained. There could be 2 types of logs, by the two connectors:

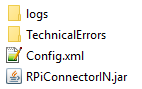
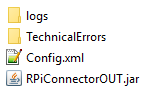
1. **Connector Monitoring log**: this log consists in a text files with .log extension, which appears in a folder, called *logs*, in the same folder where the .jar is deployed. It tells if the connectors detected errors or the file has been treated without problems. An history of the logs is held. The old logs have the date in their name:

****

1. **Technical errors log**: this kind of logs appears if a technical error not associated to the data is detected. If this log is present, the connector has stopped. The name of the technical errors log is *<Name of the connectors>\_TechnicalError.yyyy\_MM\_dd\_hhmmss.log* where the last part of the name is the date and time when the errors has been produced. They are written in a folder named *TechnicalErrors\_yyyy-MM-dd* where the last part is only the date when the error is produced. These folders are contained in a general folder called *TechnicalErrors*, which is in the same location of the .jar. Example:

****

The two connectors are deployed as two .jar executable, each one with is specific Configuration file and their proper logs folders. They appear like:

## Software Modules and Components

### Graph Database

The objective of this section is to design a tool for hierarchical description of meter/submeter network relevant to our aims. The main idea is to represent the meter hierarchy by means of a tree-graph using C# GraphEngine framework. A specific graph server will be developed and will expose SOAP webservice contracts that will enable third-party clients to insert, delete and consult nodes stored in the graph.

Graphs are information representation focused on relations between information chunks. Entities are stored as nodes, and relations between these objects by edges. Nodes can be people, meters, physical or abstract objects; while edges may be relation between people (family bonds, relation between accounts on social networks, etc.), logical or hierarchical bonds between objects.

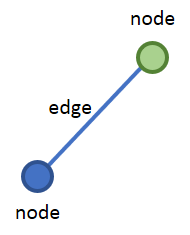


Figure Simple graph with only 2 nodes and one edge.

Hierarchical relations (such as in military organizations) may be represented as a tree-graph, i.e. a graph where there is exactly one path (without return) between every node pair. In our case, the representation will correspond to geographical spaces (city>building>floor).

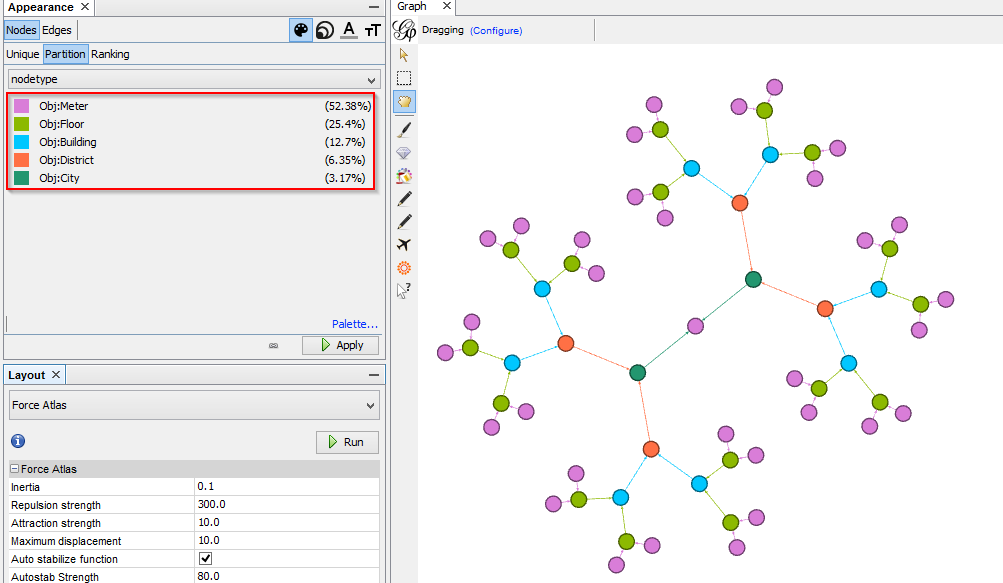


Figure Example of a graph-tree on GePhi (free software written in java under GNU GPL licence). Colors depend on the node type.

#### GraphEngine Framework

GraphEngine is a microsoft framework available for C# 4.5. Nodes are registered in cache for quick read and write. Data can also be stored on disk for persistence.

#### Application for WaterM

The goal of the graph tool will be to store the tree description of geographical objects (such as site, building, floor, etc.). These objects may carry meter, and lower level geographical objects. The structure is that of a nested doll principle: one may carry only lower level objects, and can be carried by one and only one parent, which must be of a higher level (not necessarily the next higher though).

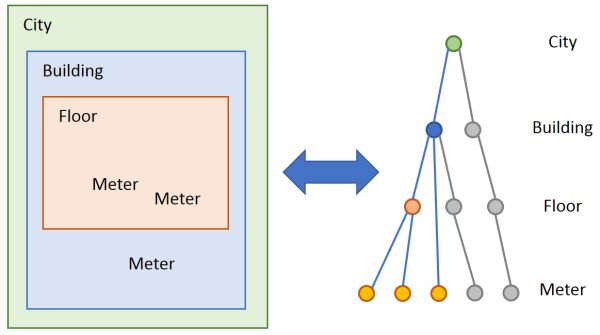


Figure Equivalence between "nested doll representation" and tree graph representation. Note that a Meter may be carried by any higher-level object, and not only Floor.

Meters may be linked to one and only one geographical entity. The graph base will store the geographical representation, while the SQL database will hold data registered by meters. The meters reference (id and properties) will be stored in both SQL and graph databases. This will allow logical connection between both datasets: it is sufficient to know the meter id to recover the meter properties and registered data from the SQL database, and its parent locations from the graph database.

Geographical aggregates will enable the reduction of meter data for meters bound to a given node. For example, it may be possible to compute the total flux of water through a given building, by aggregating consumption recorder by meters bound to this building, plus meters bound to the building floors.

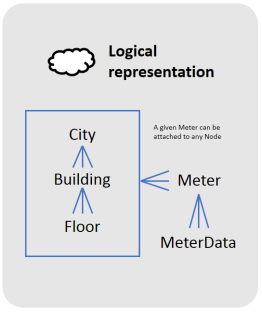
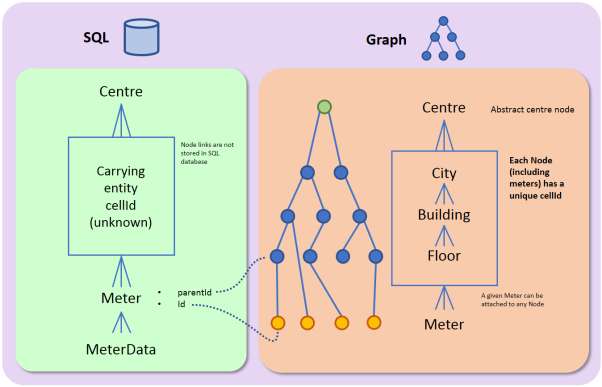
 

Figure Graphical description of the expected logical structure (top) and software distribution between SQL database and graph-oriented database. Dashed lines are relations between databases.

On demand, the graph server will deliver the graph structure as a recursive object.

Queries can be performed :

* Get total cell count,
* get node by id (the SQL-stored id can be used, as well as the graph CellId),
* get central node as a structured recursive object (holding all other nodes as well),
* get all nodes as a structured recusive object,
* get nodes with given type,
* get a json description of the given node (from cellId),
* get full parent line for a given node (incl. Meter).

Node pool can also be manipulated using those services, such as:

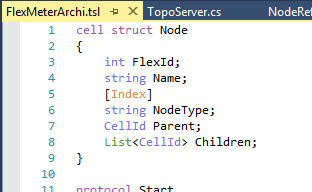
* Add a new node (that can carry subnodes in a recursive manner),
* Add several nodes,
* Delete a node, or an enumeration of nodes.

Those services should be accessible via WCF web services.

#### Technical use of GraphEngine for this purpose

Both meters and geographical nodes will be stored in graph database as Nodes. Edges will be oriented through a parent/child relation: a higher rank geographical node will be parent of children nodes; meters will only be children of other non-Meter nodes and won’t carry any children (ex: City>District>Building>Floor>Meter).

Nodes are described in a TSL file:



Compiling TSL generate classes that will enable manipulation of optimized Node objects.

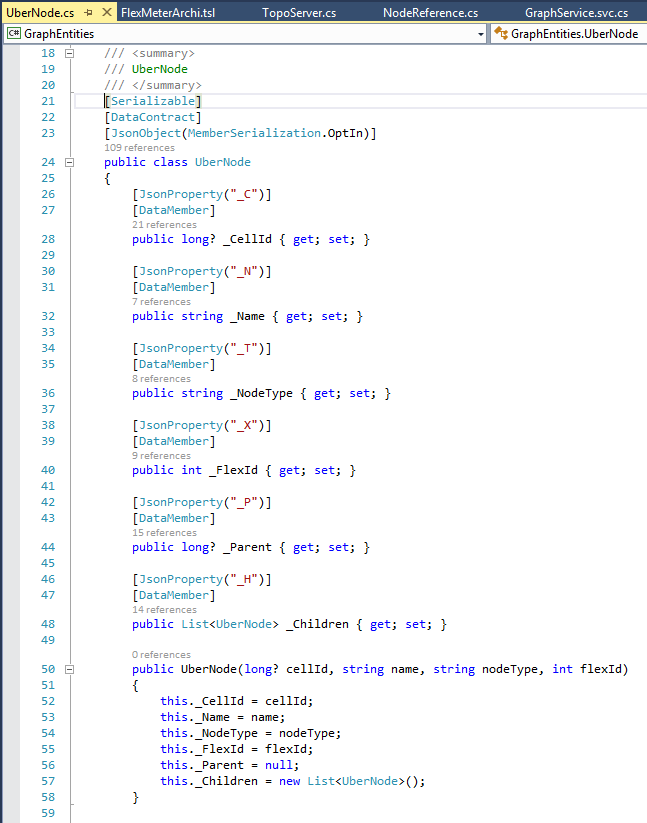


Figure Excerpt of the Node class description, as a recursive object (note that \_Children object is a Node collection).

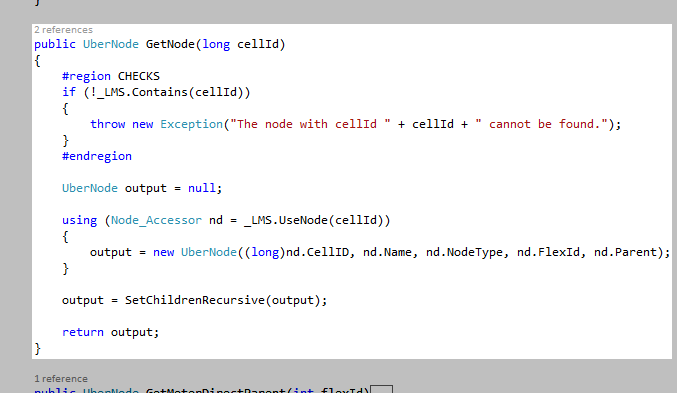
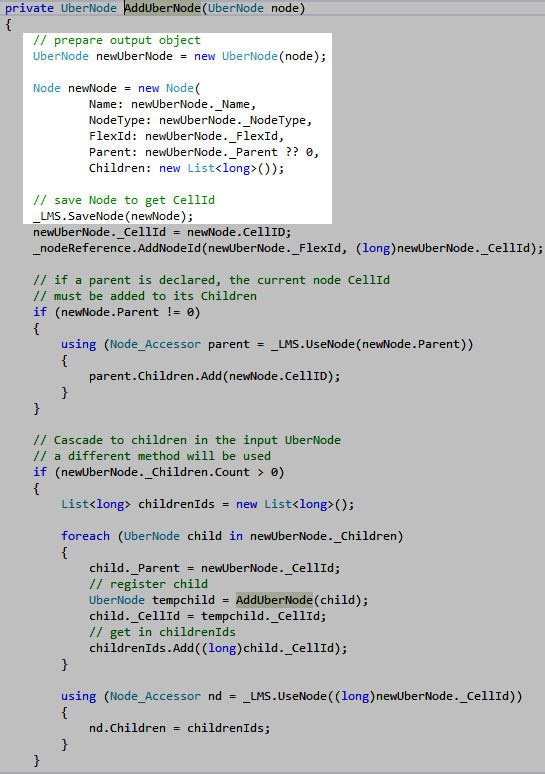


Figure Example of a GraphEngine query on Nodes

Adding a new Node:



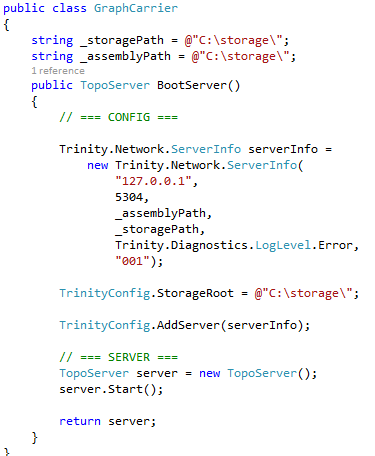


Figure GraphEngine server configuration (note StorageRoot description, that allow data persistence).

#### Tests

A simple test program *fullServerTest* has been developed. The aim is to test the result of queries which are processed on client side and server side, and which are supposed to return the same result.

First, the node set is provided by a json file which is parsed to a *UberNode* object, and registered to the graph database through the correct webservice.



Figure 8 Json node description (excerpt).

Queries on the client side are rewritten to reproduce the server-side queries behaviour. The query results are both performed, and their results are compared.

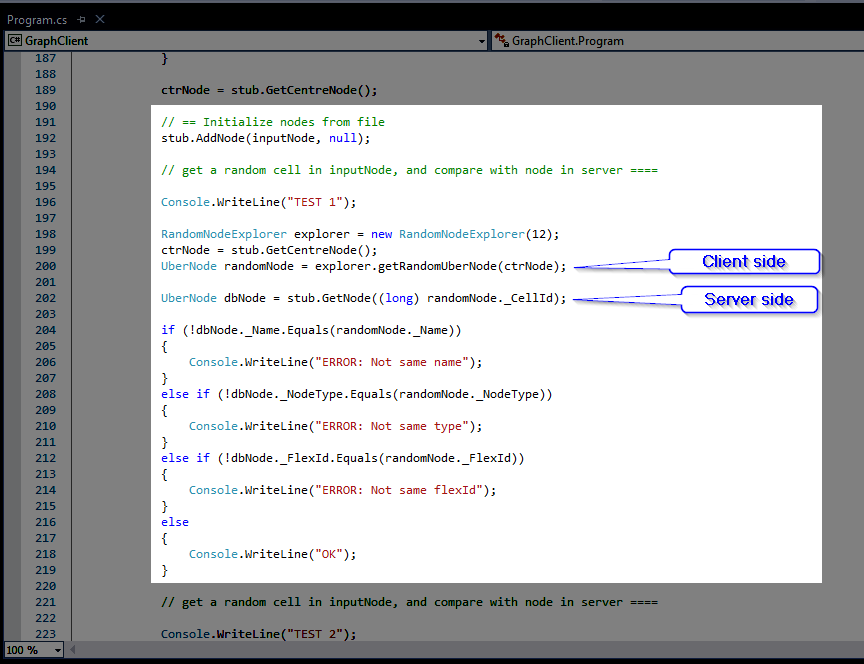


Figure One example test code.

Tests are proving correct:

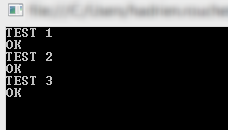


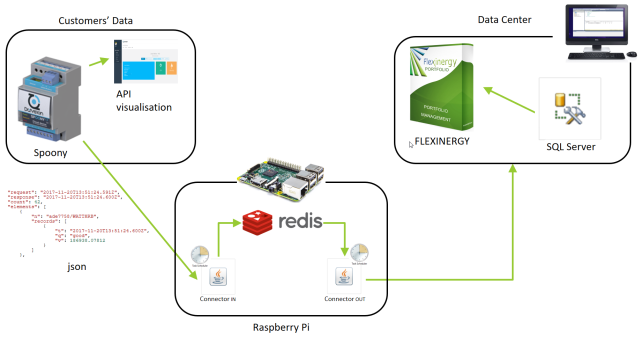
Figure Tests results (first 3)

## Execution of the pilot testing

### Description of the execution and building up the pilot test

As we already have said, the data from the Spoony is transformed into a suitable format for their insertion into REDIS. The two connectors IN and OUT are in .jar format. A scheduler (cron for DEBIAN) launches every 5 minutes first the RPiConnectorIN.jar to set Data into REDIS, after the task is accomplished the scheduler launch the RPiConnectorOUT to get the data from REDIS and send them to the Data Center. If the Connection between the Raspberry and the Data Center is shut down or lost for any reasons, REDIS assures that the data are not lost and they are stored. When the connection is re-established the RpiConnectorOUT retrieves the current data and the historical data. When the Data Center responds that the data are correctly written into SQL DB, the connector out flushes REDIS DB.

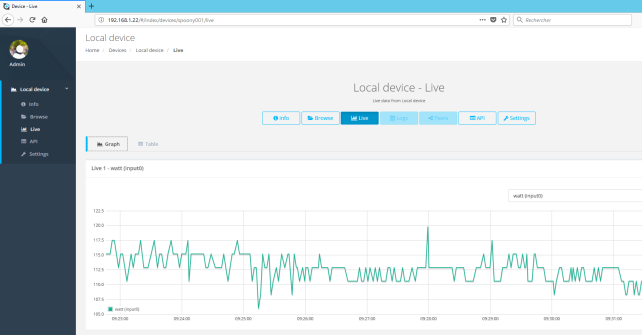
In what follows, a simplified scheme of the transfer between client DB (e.g. Spoony) and the DATA CENTER (SQL and FLEXINERGY) via the Raspberry Pi (Connectors IN/OUT and REDIS) is presented:



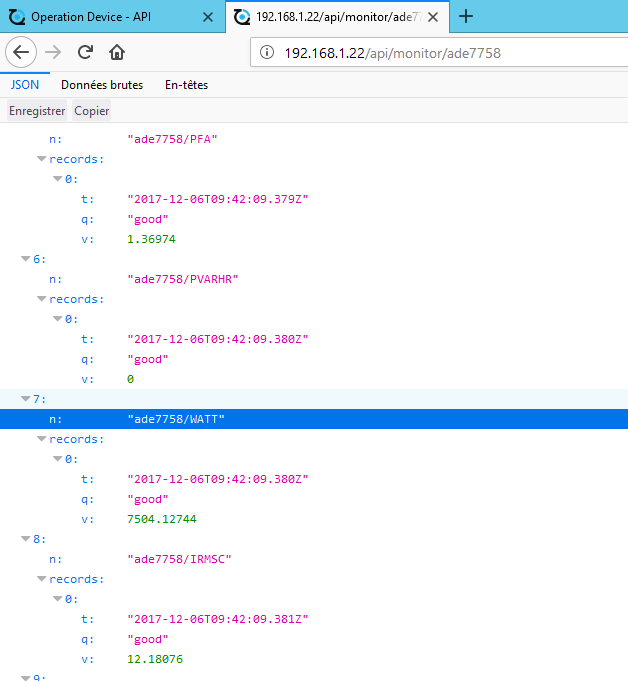
### General results

The protocol we made up let us to show some preliminary results. They are presented in the follow.

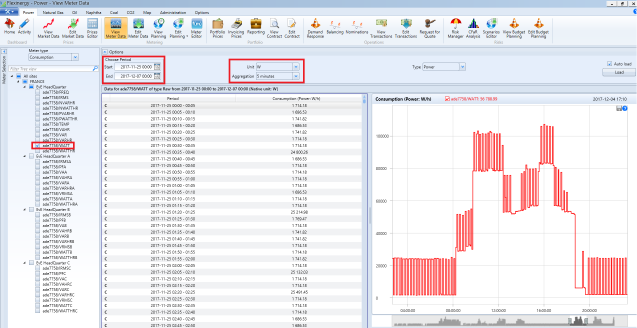
As example, we take the consumption of *watt* in the Evolution Energie premises from the Spoony in json format. They are visible in the Spoony API. We remind that with this API no storage of data is possible and the API server is available on the local network of where the Spoony is installed. In fact, only live data could be visualised as shown in the picture below. Here the consumption in watt is shown at the time where the application is open. In this example, Wednesday 06 December 2017 between around 9:20 a.m.



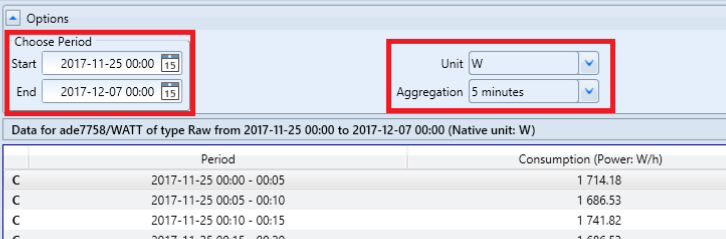
We could also have access to the data in *json* format. In the picture below different data are present. Highlighted the consumption in watt taken at the 09:42:09 a.m.



Thanks to the protocol we have presented in the previous sections (Connectors IN/OUT on Raspberry Pi), we could store the date in a SQL server and we use FLEXINERGY, the *Evolution Energie* solution, to show the data. In the next picture, we show the data in Watt, with a frequency of 5 minutes. In the “Meter data View” unit, we could see on the left part, the list of all the Meters present in Th SQL DB, like the watt consumption or the temperature (ade7758/FREQ, ade7758/TEMP or ade7758/WATT). In the central part, we can see two columns. The first column represents the date and the period in which a measurement has been taken. The second column represent the value of that measure at that time. In the right part of the unit a 2D graph showing the behavior of the consumption depicted.



We choose to point out on the data between midnight of the 25th November 2017 and midnight of the 07th December 2017.

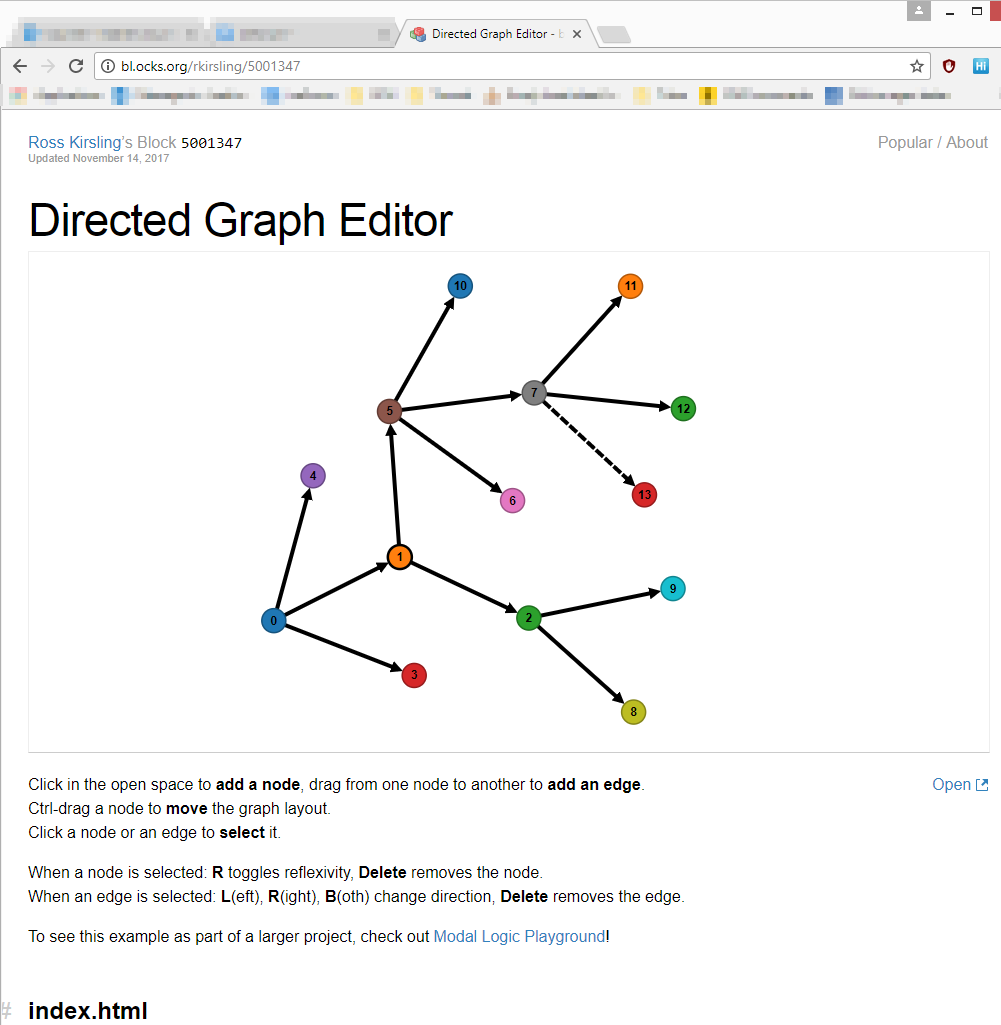


Changing the start date or the end date we could handle different periods.

## Conclusions and Further Actions

The objective was to have a local processing and storage of data on Raspberry that avoid data loss in case of connection interruption. The advantage is to have a bridge between the client DB (the Spoony in this example) and the DB SQL, without having to install this DB with the User Interface directly in the client environment. It’s important to notice that, the connector IN is specific to the Client DB, the Spoony in this case. In case we have to deal with different way to get client data, the connector should be recoded to fit that specific data.

Proposed graph database strategy will enable to store metering architecture with a flexible tool, instead of relying on SQL database. As the graph database server provides structured objects, those could be graphically represented on a web interface, using various graph frameworks. For example: [bl.ocks.org](http://bl.ocks.org/rkirsling/5001347):



Finally, this infrasctruture has the advantage that it could be applied to any type of use case such water networks which need a small and portable system to collect, store, analyze and visualize the data.

1. http://www.warp10.io/introduction/platform/ [↑](#footnote-ref-1)
2. https://grafana.com [↑](#footnote-ref-2)
3. <http://www.warp10.io/reference/> [↑](#footnote-ref-3)
4. http://www.warp10.io/reference/reference/ [↑](#footnote-ref-4)
5. <https://www.marketsandmarkets.com/Market-Reports/global-telemetry-market-88563522.html> [↑](#footnote-ref-5)
6. Human Development Report 2006. UNDP, 2006, Coping with water scarcity. Challenge of the twenty-first century. UN-Water, FAO, 2007 [↑](#footnote-ref-6)
7. <http://www.adcon.com/projects/romania-monitoring-the-danube/> [↑](#footnote-ref-7)
8. <http://www.danube-river.com/maps/danube> [↑](#footnote-ref-8)
9. <http://www.limnology.ro/water2014/proceedings/29_Vasilescu.pdf> [↑](#footnote-ref-9)
10. <http://www.beia-telemetrie.ro/> [↑](#footnote-ref-10)