# Visual Development Environment for Semantically Interoperable Smart Cities Applications

Aikaterini Roukounaki, John Soldatos, Riccardo Petrolo, Valeria Loscri, Nathalie Mitton, Martin Serrano

# Visual Development Environment for Semantically Interoperable Smart Cities Applications

Aikaterini Roukounaki, John Soldatos

Athens Information Technology,
Kifisias Ave. 44, 11525 Maroussi, Greece
{arou,jsol}@ait.edu.gr

Riccardo Petrolo, Valeria Loscri, Nathalie Mitton
Inria Lille-Nord Europe
avenue Halley 40, 59650 VILLENEUVE D'ASCQ, France
{ riccardo.petrolo, valeria.loscri, nathalie.mitton}@inria.fr

Martin Serrano
National University of Ireland, Galway,
O'Brien Centre for Science, Science Centre East, Belfield, Dublin 04, Ireland
martin.serrano@insight-centre.org

**Abstract.** This paper presents an IoT architecture for the semantic interoperability of diverse IoT systems and applications in smart cities. The architecture virtualizes diverse IoT systems and ensures their modelling and representation according to common standards-based IoT ontologies. Furthermore, based on this architecture, the paper introduces a first-of-a-kind visual development environment which eases the development of semantically interoperable applications in smart citites. The development environment comes with a range of visual tools, which enable the assembly of non-trivial data-driven applications in smart cities, including applications that leverage data streams from diverse IoT systems. Moreover, these tools allow developers to leverage the functionalities and building blocks of the presented architecture. Overall, the introduced visual environment advances the state of the art in IoT developments for smart cities towards the direction of semantic interoperability for data driven applications.

**Keywords:** Smart Cities, Internet-of-Things, Semantic Interoperability, Development Tools

# 1 Introduction

## 1.1 IoT Application silos in Smart Cities

The Internet-of-Things (IoT) [1] is a key enabler of software applications for smart cities, given that it deals with several integral elements of smart city applications such as the communication across heterogeneous devices, the dynamic discovery of sensors and data streams, the collection and filtering of information from multiple data streams, as well as the analysis of data stemming from multiple data sources within the city. As a result of the proliferating IoT deployments in smart cities in different sectors (such as energy, transport, e-government), most cities have nowadays to manage multiple smart city deployments and applications. In most cases these applications have been designed and developed independently from each other, even when they have been deployed as part of a unified strategic plan. Furthermore, the various applications tend to evolve independently of each other, given that updates and enhancements to existing smart city applications rely on heterogeneous platforms and architectures. Also, in most cases the various applications have been designed and purchased by different departments of the cities. Therefore, the interconnection of different smart city systems and applications is very difficult, which is a set-back to the integration of these applications towards pursuing city-wide (instead of application-specific) goals. Hence, smart cities are nowadays characterized by the presence of a set of disaggregated heterogeneous application silos [2].

Recent advances in IoT architectures and semantic web technologies enable the convergence of these silos and facilitate the development of added-value integrated cross-context applications. In particular, the advent of IoT ontologies that can represent IoT data streams and services in a semantically interoperable way provides the means for integrated semantically unified IoT applications. The later integrated applications refer typically to applications that combine data and/or services from different legacy deployments within the smart city. Along with IoT ontologies, platforms supporting IoT ontologies for semantically interoperable modelling data/services have emerged. Such platforms provide a foundation for the implementation and operation of semantically interoperable IoT applications in the smart cities. In this paper, we present an IoT architecture for semantically unified applications in smart cities, which leverages existing ontologies (such as W3C SSN ontology [3]) and middleware platforms that support them (such as OpenIoT) [10].

## 1.2 Development Environment and Tools

Based on the close affiliation of IoT with smart cities, tools and techniques for developing IoT applications can act as a catalyst for the development of smart cities applications. Early frameworks for programming IoT applications have focused on the areas of WSN (Wireless Sensor Networks)[4] and RFID applications [5]. Most of these frameworks provide the means for virtualizing sensors and actuators, thereby

alleviating the heterogeneity of the various devices [6]. Recently, we have also witnessed the emergence of software engineering solutions for the integrated development of more general IoT applications (e.g., [7], [8], [9]). These solutions leverage semantic models that capture the main elements of IoT applications, including standard semantic models and ontologies, such as the W3C Semantic Sensor Networks (SSN) [11], which provides the means for abstracting/virtualising sensors. Such software engineering solutions are appropriate for supporting the development of applications that rely on the semantic unifications of the legacy disaggregated silos, which comprise the digital mature smart cities.

In addition to introducing an IoT architecture for semantic interoperability, the present paper presents also a development environments for integrated cross-silo applications in smart cities. This environment leverages the IoT architectures and its common IoT semantic models (ontologies) in order to provide the means for combining, aggregating and processing (in a unified way) data streams stemming from diverse IoT systems and application silos within a smart city. The environment comes with a range of visual drag-and-drop tools, which boosts developers' productivity. It also takes advantage of the IoT architecture towards providing added-value data processing functionalities such as data analytics.

### 1.3 Structure of the Paper

Following this introductory section, Section 2 introduces an IoT architecture for semantic interoperability, which provides a wide range of added-value data processing functionalities. The latter operates over multiple semantically unified systems, rather than over a single legacy IoT system. Section 3 is devoted to the presentation of the development environment and of the visual tools that it comprises. Section 4 illustrates a range of sample data processing applications, which demonstrated the added-value of the development tools. Finally, section 5 concludes the paper.

## 2 IoT Architecture for Semantic Interoperability

An overview of the IoT architecture for semantic interoperability is provided in Fig. 1. It is structured according to the following layers:

- **IoT Systems/Platforms and Data Sources Layer**: This is the lowest layer of the architecture, where various IoT systems (including IoT platforms, applications and data sources) reside. The architecture integrates and processes information from multiple IoT systems. Therefore, this layer comprises the various IoT systems that are virtualized and integrated as part of the architecture. In order to validate the architectural concept, we have integrated and processed data from several IoT platforms. However, the architecture is flexible in the integration of additional platforms and/or applications, as soon as these platforms (or applications) expose a well defined PPI (Platform Provider Interface).

- **Platforms Access and Data Acquisition Layer**: This layer enables access to data and services of the underlying platforms/systems in a secure and authenticated way. Its role is to access the low-level capabilities of the IoT systems (through the PPI) and accordingly to transform the acquired data and metadata into a common data model (i.e. expressed based on an IoT ontology comprising sensors, IoT and smart cities concept). Hence, the functionality of the layer involves translating the low-level semantics of the individual platforms to the high-level and richer semantics of the presented platform. At this layer we also specify the notion of the adapter module, which undertakes the transformation of the lower-level PPI semantics to the high-level ontology.
- **Platform Agnostic Data Management Layer**: This layer provides cloud-based functionalities for managing data and metadata that comply with the VITAL ontology. At this layer, the various data streams (and their metadata) are modelled and formatted according to the common ontology. The offered services include data and metadata persistence, creation of new data, discovery of data subject to various criteria and more. Access to the data of the platform will be based on semantic web technologies and techniques, given the modelling of the data according to the ontology. The platform agnostic data management layer offers a wide range of services to higher level applications, which reside in the added-value functionalities layer. These services will be accessible in a virtualized platform and location agnostic manner, through VUAIs (Virtualized Unified Access Interfaces). VUAIs are abstract interfaces residing at the top layer of the architecture, thereby enabling access to added-value data processing and process management functionalities, including CEP, service discovery, filtering, and other functionalities. The platform agnostic data management layer includes also a Service Discovery (SD) module, which enables the look-up and resolution of IoT resources.
- **Added Value Functionalities Layer**: This layer comprises a set of complete services and tools, which leverage data and services from the platform agnostic management layer. The virtualized services of this layer will be provided by the following modules (depicted in Fig. 1): (A) **Filtering Module**: The filtering module of the architecture provides the means for reducing the information associated with individual data streams persisted in the platform agnostic data management layer. It therefore reduces unwanted information, thereby optimizing processing performance and economizing on network bandwidth; (B) **Complex Event Processing (CEP) Module**: This module enables the processing of data streams for multiple sources in order to identify patterns and/or infer events; (C) **Orchestration Module**: This module combines and manages multiple services from the above-listed modules, in order to deliver new added-value services. The combination of the various services is based on a workflow of service oriented components and interactions, which may be specified on the basis of rules.
- **Smart City Applications and Tools Layer**: The architecture supports the development, integration, deployment and operation of smart city applications, notably applications leveraging data from multiple IoT platforms and applications (including legacy IoT applications in urban environments). All smart city applications are (during their operation) accessing data and services of

the platform agnostic data management layer, including the discovery, filtering and CEP services. This layer includes management tools that enable the monitoring and configuration of IoT platforms, applications and services at various granularities. It also includes development tools enabling the development of smart city applications and services. The various applications and tools will be able to access data from the platform agnostic data management layer, as well as services offered from this layer.

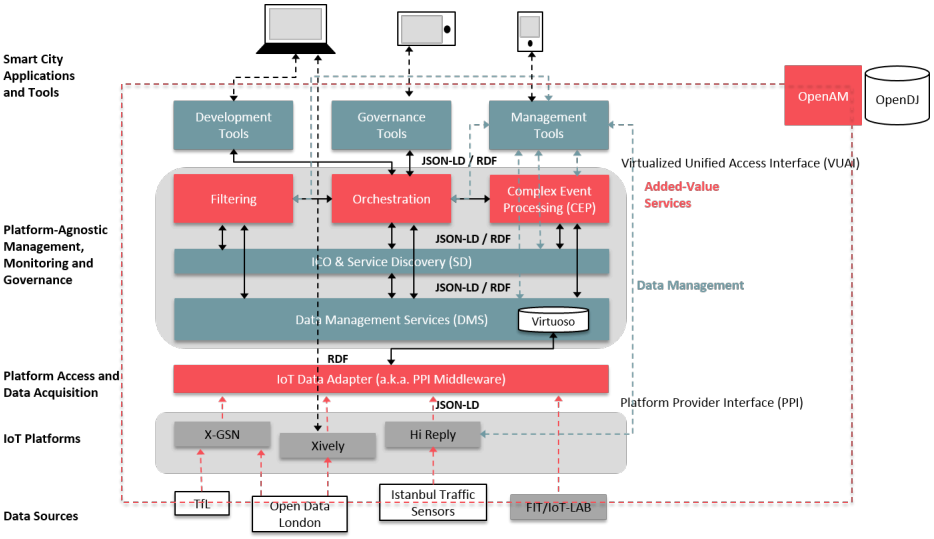


**Fig. 1.** Overview of IoT Architecture for Semantic Interoperability

In the scope of the FP7 VITAL project (www.vital-iot.eu) a prototype implementation of the presented architecture is provided. This prototype implementation constitutes a middleware platform (i.e. the VITAL platform) that enables the development, deployment and operation of semantically interoperable applications in smart cities. Due to the implementation of the VITAL platform in-line with the above-presented architecture, we also conveniently refer to the architecture as the VITAL project architecture.

## 3 Development and Deployment Environment

Based on the architecture described above we have implemented an environment enabling the assembly of IoT applications that take advantage of the functionalities of the architecture. The environment exploits the semantic modelling and added-value functionalities of the architecture towards enabling the development of semantically interoperable applications. It is built over the popular Node-RED open-source tool for wiring the Internet of Things [12]. Nodes and flows are the two fundamental concepts

in Node-RED. A node is a well-defined piece of functionality. Based on the number of its input and output ports, a node can be of one of the following types: (A) An input node, which is a node that has one or more output ports; (B) An output node, which is a node that has one input port; (C) A function node, which is a node that has one input port and one or more output ports. Nodes can be wired together into what is called a flow. Input nodes sit at the start of a flow, output nodes sit at the end of a flow, and finally function nodes sit in the middle of a flow. Flows can be considered as programs, and nodes as the blocks that can be used to build them. The Node-RED platform comprises two components: (1) a browser-based editor that allows us to design flows (i.e. programs), and (2) a light-weight runtime where we can deploy and execute our flows. The browser-based flow editor is used for creating flows and deploying them on the Node-RED runtime. It comprises a palette, which can be extended with new nodes.

### 3.1 Development and Deployment Environment Overview

The development and deployment environment allows developers to access and compose the various capabilities offered by the architecture, in order to implement smart city applications. Part of the development and deployment environment is the VITAL development tool that serves as a single entry point to developers that want to build and deploy IoT applications that transcend multiple IoT platforms, architectures and business contexts by leveraging the capabilities provided by the VITAL framework. All these capabilities are exposed through VUAIs. All mechanisms to be integrated into that tool are exposed through VUAIs, which are implemented as RESTful web services, thus rendering Node-RED ideal as the basis for the implementation of the VITAL development tool.

The development tool has been also enhanced with a number of nodes relating to the architecture, as well as with functionalities provided by the R project [13], a programming language and an environment for statistical computing and graphics. The result is an easy-to-use tool that also enables its users to perform a number of task (e.g. retrieval of IoT system metadata) relating to the semantic interoperability architecture, along with data analysis (e.g. data value prediction or clustering) tasks.

Based on the above-mentioned nodes and extensions, the palette of the development tool, offers, apart from the nodes already present in Node-RED, one node for each piece of functionality offered by a VITAL component. For example, Fig. 2 depicts the node that corresponds to the sensor-resource discovery functionality. In a similar way developers can exploit any VITAL functionality by just adding the corresponding node in their flow and setting its properties. While the flow is running, the node interacts with the appropriate components of the VITAL platform in order to perform its tasks. Thus, VITAL nodes facilitate the use of VITAL capabilities by hiding (from the developers) low level implementation and formatting details.

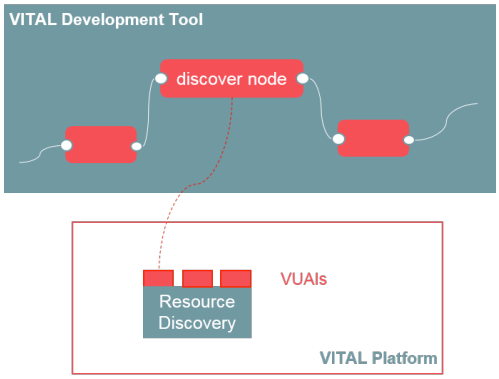


**Fig. 2.** Concept of VITAL Resource Discovery Node within the Development Tool

### 3.2 Authentication and Multi-User Environment

In order to be able to direct each user to a dedicated Node-RED instance that has been created for him/her, a router component has been developed. The router component is a web application that is responsible for the user authentication, as well as for re-directing an authenticated user to the Node-RED instance that is associated with them. Hence, the VITAL platform users register to the router. Whenever a registered user wants to manage their workflows, they log into the router. Once the user has been successfully authenticated, the router looks up into its local repository (1) the host where the Node-RED instance that is dedicated to the user is running and (2) the port where the Node-RED editor of that instance can be accessed. In case there is no running instance for the user, the router creates one. This effectively means that the router creates a clone of the Node-RED codebase, sets all necessary settings (see below), and starts it. Once the new Node-RED instance has been successfully created, the router saves the new mapping into its local repository for future reference.

In order to allow only to the user that owns the Node-RED instance to access it, but without having to log in again (after they have been successfully authenticated by the router), we must (1) enable user authentication in all Node-RED instances and (2) define a user in each Node-RED instance that has the same username and password with the corresponding VITAL user, to whom this Node-RED instance belongs. Both these requirements can be satisfied by setting the appropriate values in the settings file of each Node-RED instance that the router creates.

Based on the above, before the router re-directs an authenticated user to the appropriate Node-RED editor, it needs to contact the Node-RED instance (i.e. perform an HTTP POST to /auth/token), and exchange the credentials of the user with an access token. Once the router has obtained the access token, it can finally re-direct the user to the Node-RED editor with the access token set in the Authorization header. The authentication process is depicted in Fig. 3. It is a process that overall alleviates

the single-user nature of the Node-RED environment, thus enabling the VITAL platform to operate as a multi-user platform.

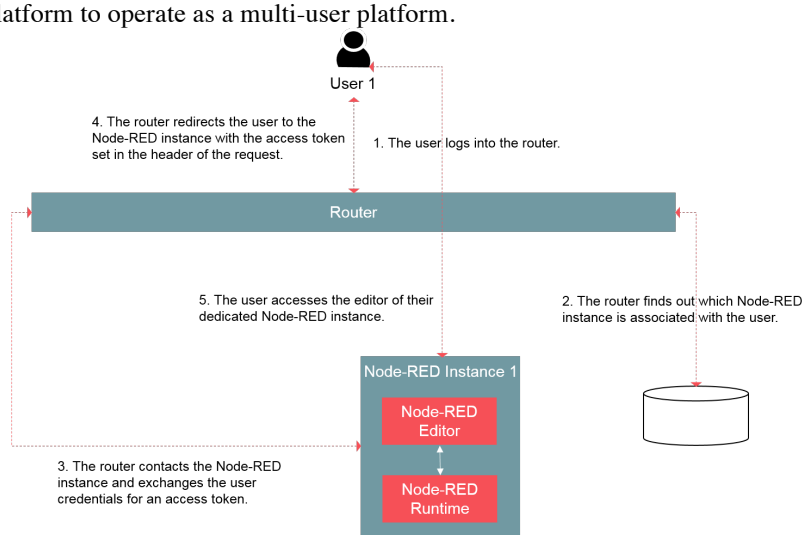


**Fig. 3.** User Authentication in the scope of the VITAL multi-user Node-RED environment

### 3.3 VITAL Nodes

In following paragraphs we illustrate two of the new nodes that the VITAL platform has added to the core palette of Node-RED, notably nodes corresponding to PPI implementations and Resource Discovery.

#### 3.3.1 Platform Provider Interface Implementation

The Platform Provider Interface (PPI) is defined as a set of primitives, marked as either mandatory or optional. All IoT systems that are compliant to the VITAL platform are expected to implement and expose (as RESTful web services) at least those primitives that are designated as mandatory. The following nodes have been specified and implemented:

- **PPI nodes**: These are configuration nodes that store information about how to access the PPI implementation of a VITAL compliant IoT system (e.g. its base URL). Configuration nodes are a special category of nodes in Node-RED that can be used for sharing configuration among nodes. For example, PPI nodes can be used by system and services nodes (see below) in order to select the PPI implementation they want to access, without having to specify more than once details on how to access it. The properties of these nodes include the base URL of the PPI implementation, as well as whether basic authentication is required.

- **System Node**: System nodes can be used to retrieve metadata about a PPI compliant IoT system. When a system node receives a message, the node makes an HTTP POST request to BASE_URL/metadata, and then sends out a message that contains the response to that request. The properties of these nodes include the PPI implementation to retrieve metadata about.
- **Sensor Nodes**: Sensors nodes are function nodes that retrieve metadata about sensors that a VITAL compliant IoT system manages. The messages sent to these nodes can be used to filter the sensors to retrieve metadata for (based on their ID and type), whereas the messages sent by these nodes contain the retrieved sensor metadata.
- **Service Nodes**: Services nodes can be used to retrieve metadata about IoT services that a PPI compliant IoT system provides. The message that a services node receives may contain information, which can be used to filter the services to retrieve metadata for (based on their ID and type), whereas the message that a services node sends contains the retrieved service metadata. The services node is responsible for making an HTTP POST request to BASE_URL/sensor/metadata, in order to fetch the requested metadata
- **Data Nodes**: Data nodes are function nodes that pull observations made by sensors managed by a PPI compliant IoT system. Input messages may contain information, which can be used to filter the observations to fetch (based on the corresponding sensor, property and time), whereas output messages contain the retrieved observations in JSON-LD format.

#### 3.3.2 Resource Discovery

The following extra nodes have been added to the node palette in order to enable access to Resource Discovery functionalities:

- **Resource discovery**: Resource discovery nodes are configuration nodes that store information about how to access the Resource Discovery component of an instance of the VITAL platform. The properties of these nodes include the base URL of the component, as well as whether basic authentication is required (in which case a username and a password can also be provided).
- **Discover sensors**: These are function nodes that can be used to discover sensors that reside within a certain radius from a certain point (latitude-longitude).The messages received by these nodes contain the radius, the latitude and the longitude. The messages sent by these nodes include metadata about the discovered sensors.

## 4 Sample Validating Applications

In this section we present a set of sample workflows which have been implemented using the VITAL development tool.

### 4.1 Sample workflow #1: Find the last observation made by a specific traffic sensor

We have used the development tool to implement a web service that accepts HTTP GET requests, which contain the URI of a traffic sensor in a sensor parameter in the query string, and responds with the last observation made by that sensor. For the implementation of the web service, we make use of an implementation of the PPI specification provided for the traffic sensors in Istanbul. The flow consists of the following nodes: (A) An http in node that accepts HTTP GET requests at /last-observation; (B) A function node that extracts the sensor URI from the query string of the request, and uses http://vital-iot.eu/ontology/ns/Speed as the property URI; (C) A data node that retrieves the last observation made by that sensor for that property; (D) An http response node that responds to the initial HTTP request with the observation value.

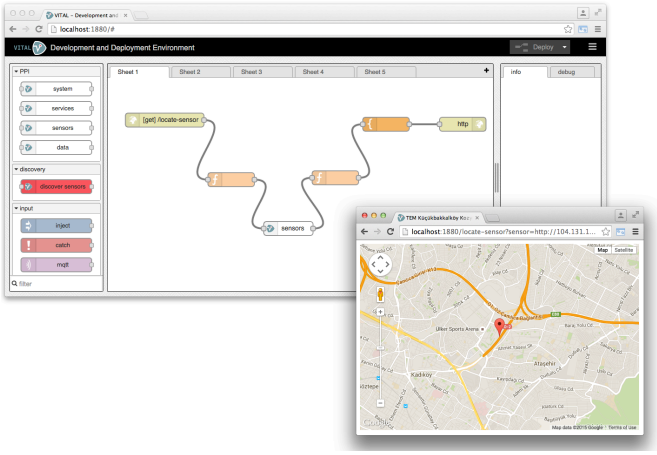


**Fig. 4.** Screenshot for sample workflow #2 - Show the location of a specific traffic sensor

### 4.2 Sample workflow #2: Show the location of a specific traffic sensor

This workflow implements a web service that accepts HTTP GET requests, which contain the URI of a traffic sensor in a sensor parameter in the query string, and responds with a static HTML page that contains a map with a marker set on the location of that sensor. For the purposes of this web service, we use again the PPI implementation for Istanbul traffic sensors. The result of using that flow is depicted in Fig. 4. The flow comprises six nodes: (A) An http in node that accepts HTTP GET requests at /locate-sensor; (B) A function node that extracts the sensor URI from the query string of the request; (C) A sensors node that retrieves metadata about the sensor with that URI; (D) A function node that extracts the name, the longitude and the latitude of the sensor from the retrieved metadata; (E) A template node that creates an HTML page with the name of the sensor in the title, and a map with a

marker at the location of the sensor; (F) An http response node that responds to the initial HTTP request with the that HTML page.

### 4.3 Sample workflow #3: Predict the next observation that a specific traffic sensor will make

The implementation of this workflow is based on the rstats which is an interface to R project. In particular, the development tool and rstats are used in order to implement a web service that accepts HTTP GET requests, which contain the URI of a traffic sensor in a sensor parameter in the query string, and responds with a prediction for the next observation that that sensor will make. The flow contains five nodes: (A) An http in node that accepts HTTP GET requests at /predict-next-observation; (B) A function node that that extracts the sensor URI from the query string of the request, and uses http://vital-iot.eu/ontology/ns/Speed as the property URI; (C) A data node that retrieves all observations made by that sensor for that property since a given (earlier) time instant; (C) A function node that retrieves the values from the returned observations, and uses R to predict the value of the next observation according to the best ARIMA model (using the forecast R package); (D) An http response node that responds to the initial HTTP request with that value.

## 5 Conclusions

Existing IoT ontologies and semantic models can serve as a basis for semantically unifying data sources and data streams from a variety of legacy IoT systems within a smart city. This paper has introduced an IoT architecture which collects and processes information from multiple IoT systems in a smart city environment. Based on this architecture it has also introduced a toolset for IoT application developers that: (A) Ensures virtualized access to diverse IoT systems, including the sensors they manage and the services they provide; and (B) Supports the functionalities provided by the VITAL platform as the latter are specified in the above-mentioned architecture. The toolset has been implemented on the basis of extensions to the popular Node-RED visual environment for IoT. The resulting tools provide the means for implementing semantically interoperable data driven applications in smart cities environment with only minimal programming effort, thus boosting IoT application developers' productivity.

## Acknowledgements

This work was funded in part by the European Community in the framework of the VITAL FP7 project (Virtualized programmable InTerfAces for smart, secure and cost-effective IoT depLoyments in smart cities) under contract number FP7-SMARTCITIES-608662. The authors acknowledge help from all partners of the project.

## References


1. Atzori, L., Iera A., Morabito G.. The internet of things: A survey. Computer Networks, 54(15):2787-2805, (2010).
2. Schiele G., Soldatos J. and Mitton N., Moving Towards Interoperable Internet-of-Things Deployments in Smart Cities. ERCIM News, Special Theme: Smart Cities, 98, (2014).
3. Taylor K. Semantic Sensor Networks: The W3C SSN-XG Ontology and How to Semantically Enable Real Time Sensor Feeds. Semantic Technology Conference, June 5-9, San Francisco CA, USA (2011)
4. Chatzigiannakis I., Mylonas G., Nikoletseas S. 50 ways to build your application: A survey of middleware and systems for Wireless Sensor Networks. ETFA, 466-473, (2007)
5. Anagnostopoulos A., Soldatos J., Michalakos S. REFiLL: A lightweight programmable middleware platform for cost effective RFID application development. Pervasive and Mobile Computing 5(1): 49-63 (2009).
6. Aberer K., Hauswirth M., Salehi A. Infrastructure for Data Processing in Large-Scale Interconnected Sensor Networks. MDM, 198-205 (2007).
7. Dimakis N., Soldatos J., Polymenakos L., Fleury P., Curín J., Kleindienst J. Integrated Development of Context-Aware Applications in Smart Spaces. IEEE Pervasive Computing 7(4): 71-79 (2008)
8. Patel P., Pathak A., Cassou D., Issarny V., Enabling High-Level Application Development in the Internet of Things, Sensor Systems and Software, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering Volume 122, 111-126, (2013)
9. Cassou D., Bruneau J., Mercadal J., Enard Q., Balland E., Loriant N., Consel C.. Towards a tool-based development methodology for sense/compute/control applications. ACM international conference companion on Object oriented programming systems languages and applications companion, 247-248, (2010)
10. Serrano M., Quoc H., Le Phuoc D., Hauswirth M., Soldatos J., Kefalakis N., Jayaraman P., Zaslavsky A. Defining the Stack for Service Delivery Models and Interoperability in the Internet of Things: A Practical Case With OpenIoT-VDK. IEEE Journal on Selected Areas in Communications 33(4): 676-689 (2015)
11. Compton M., Barnaghi P., Bermudez L., Castro R. G., Corcho O., Cox S., et. al.. The SSN Ontology of the Semantic Sensor Networks Incubator Group. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, ISSN 1570-8268, Elsevier, (2012).
12. Node-RED, A visual tool for wiring the Internet-of-Things, http://nodered.org/
13. R, The R Project for Statistical Computing, http://www.r-project.org/