OpenIoT: Open Source Internet-of-Things in the Cloud

John Soldatos[1], Nikos Kefalakis[1], Manfred Hauswirth[2], Martin Serrano[2], Jean-Paul Calbimonte[3], Mehdi Riahi[3], Karl Aberer[3], Prem Prakash Jayaraman[4], Arkady Zaslavsky[4], Ivana Podnar Žarko[5(✉)], Lea Skorin-Kapov[5], and Reinhard Herzog[6]

[1] Athens Information Technology, 0.8 Km Markopoulo Ave., P.O. Box 68, 19002 Peania, Greece
{jsol,nkef}@ait.gr
[2] INSIGHT@ National University of Ireland, Galway, IDA Business Park, Lower Dangan, Galway, Ireland
{manfred.hauswirth,serrano}@deri.org
[3] EPFL IC LSIR, École Polytechnique Fédérale de Lausanne, Station 14, 1015 Lausanne, Switzerland
{jean-paul.calbimonte,mehdi.riahi,karl.aberer}@epfl.ch
[4] CSIRO Digital Productivity Flagship, Building 108 North Road, Acton, Canberra 2617, Australia
{prem.jayaraman,arkady.zaslavsky}@csiro.au
[5] Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia
{ivana.podnar,lea.skorin-kapov}@fer.hr
[6] Fraunhofer IOSB, Fraunhoferstr. 1, 76131 Karlsruhe, Germany
Reinhard.Herzog@iosb.fraunhofer.de

Abstract. Despite the proliferation of Internet-of-Things (IoT) platforms for building and deploying IoT applications in the cloud, there is still no easy way to integrate heterogeneous geographically and administratively dispersed sensors and IoT services in a semantically interoperable fashion. In this paper we provide an overview of the OpenIoT project, which has developed and provided a first-of-kind open source IoT platform enabling the semantic interoperability of IoT services in the cloud. At the heart of OpenIoT lies the W3C Semantic Sensor Networks (SSN) ontology, which provides a common standards-based model for representing physical and virtual sensors. OpenIoT includes also sensor middleware that eases the collection of data from virtually any sensor, while at the same time ensuring their proper semantic annotation. Furthermore, it offers a wide range of visual tools that enable the development and deployment of IoT applications with almost zero programming. Another key feature of OpenIoT is its ability to handle mobile sensors, thereby enabling the emerging wave of mobile crowd sensing applications. OpenIoT is currently supported by an active community of IoT researchers, while being extensively used for the development of IoT applications in areas where semantic interoperability is a major concern.

Keywords: Internet-of-Things · Open source · Semantic interoperability

I. Podnar Žarko et al. (Eds.): FP7 OpenIoT Project Workshop 2014, LNCS 9001, pp. 13–25, 2015.
DOI: 10.1007/978-3-319-16546-2_3

1 Introduction

We are nowadays witnessing the convergence of the Internet-of-Things (IoT) and the cloud computing paradigms, which is largely motivated by the need of IoT applications to leverage the scalability, performance and pay-as-you-go capabilities of the cloud. During recent years several efforts towards IoT/cloud convergence have been undertaken both in the research community (e.g., [1]) and in the enterprise (e.g., Xively.com). A common characteristic of these efforts is their ability to stream data to the cloud in a scalable and high performance way, while at the same time providing the means for managing applications and data streams. Nevertheless, these architectures do not essentially provide semantic interoperability [2] across IoT applications which have been developed/deployed independently from each other. Therefore, there is still no easy way to combine data streams and services from diverse IoT applications that feature incompatible semantics (e.g., units of measurement, raw sensor values and points of interest).

This paper presents an overview of the FP7-287305 OpenIoT project (co-funded by the European Commission), which has provided a middleware platform enabling the semantic unification of diverse IoT applications in the cloud. OpenIoT uses the W3C Semantic Sensor Networks (SSN) ontology [3] as a common standards-based model for semantic unification of diverse IoT systems. OpenIoT offers a versatile infrastructure for collecting and semantically annotating data from virtually any sensor available. OpenIoT exploits also the Linked Data concept [4] towards linking related sensor data sets. Furthermore, OpenIoT provides functionalities for dynamically filtering and selecting data streams, as well as for dealing with mobile sensors. It comes with a wide range of visual tools, which enable the development of cloud based IoT applications through minimal programming effort.

OpenIoT is currently available as an open source project (https://github.com/OpenIotOrg/openiot/). As of June 2014, it consists of nearly 400.000 lines of code, while it also integrates libraries of the popular Global Sensor Networks (GSN) open source project [5]. Recently, OpenIoT received an award from Black Duck, as being one of the top ten open source project that emerged in 2013 [6]. The rest of the paper is devoted to the presentation of the main technical developments of the project. The structure of the paper is as follows: Sect. 2 provides an overview of the OpenIoT platform, including an illustration of its architecture. Section 3 is devoted to the presentation of the main functionalities of the platform and how they can be used towards developing IoT applications. Section 4 provides an overview of real-life IoT applications, which have been developed based on OpenIoT. Section 5 concludes the paper.

2 OpenIoT Platform Overview

2.1 Achitecture Overview

The OpenIoT architecture comprises seven main elements [7] as depicted in Fig. 1.

- **The Sensor Middleware** (Extended Global Sensor Networks, X-GSN) collects, filters and combines data streams from virtual sensors or physical devices. The Sensor Middleware is deployed on the basis of one or more distributed instances

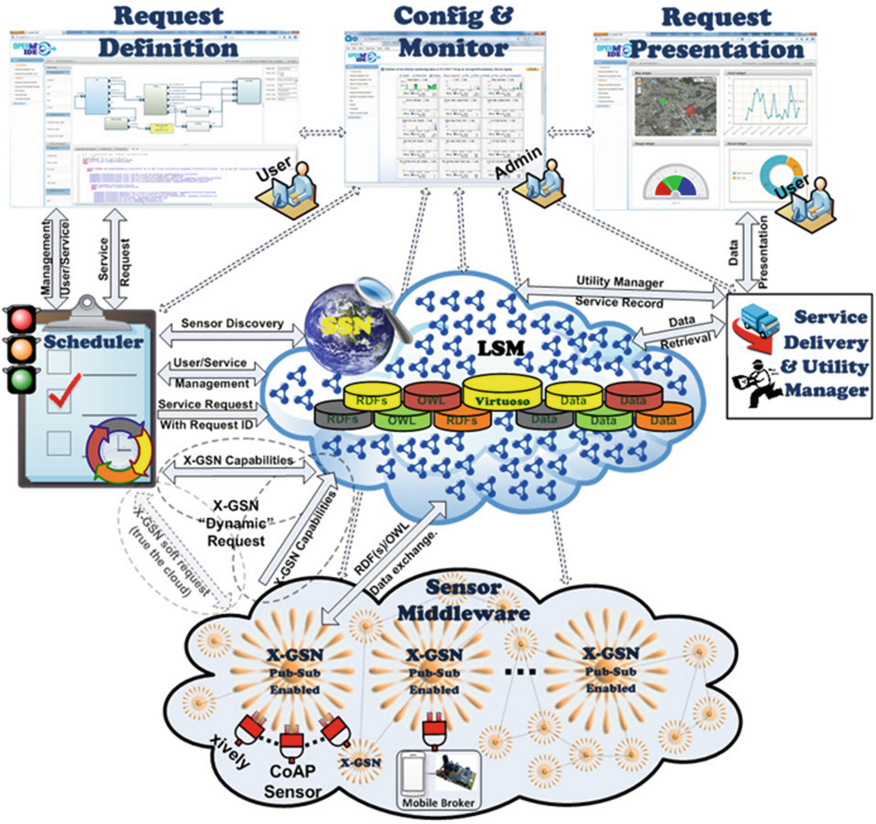

Fig. 1. Overview of OpenIoT Architecture and Main Components

(nodes), which may belong to different administrative entities. The OpenIoT prototype implementation uses X-GSN (Extended GSN), an extended version of the GSN middleware [5]. Furthermore, a mobile broker (publish/subscribe middleware) is used for the integration of mobile sensors.

- **The Cloud Data Storage** (Linked Stream Middleware Light, LSM-Light) acts as a cloud database which enables storage of data streams stemming from the sensor middleware. The cloud infrastructure stores also metadata required for the operation of OpenIoT. The OpenIoT prototype implementation uses the Linked Stream Middleware (LSM) [8], which has been re-designed with push-pull data functionality and cloud interfaces.
- **The Scheduler** processes requests for on-demand deployment of services and ensures their proper access to the resources (e.g. data streams) that they require. It discovers sensors and associated data streams that can contribute to a given service. It also manages a service and activates the resources involved in its provision.

- **The Service Delivery & Utility Manager** (SD&UM) combines data streams as indicated by service workflows within the OpenIoT system in order to deliver the requested service (typically expressed as an SPARQL query). The SD&UM acts also as a service metering facility which keeps track of utility metrics for each service.
- **The Request Definition** component enables on-the-fly specification of service requests to the OpenIoT platform. It comprises a set of services for specifying and formulating such requests, while also submitting them to the Scheduler. This component is supported by a GUI (Graphical User Interface).
- **The Request Presentation** component is in charge of the visualization of the outputs of a service. This component selects mash-ups from an appropriate library in order to facilitate service presentation.
- **The Configuration and Monitoring** component enables visual management and configuration of functionalities over sensors and services that are deployed within the OpenIoT platform.

2.2 OpenIoT Ontology for Semantic Interoperability and Linked Data Integration

The OpenIoT ontology represents a universally adopted terminology for the convergence of sensed data with the semantic web. It enhances existing vocabularies for sensors and Internet Connected Objects (ICOs), with additional concepts relevant to IoT/cloud integration such as terms to annotate units of measurement, raw sensor values and points of interest at some specific levels of granularity. In particular, the OpenIoT ontology is extending the W3C SSN ontology, which supports the description of the physical and processing structure of sensors. Sensors are not constrained to physical sensing devices: rather a sensor is anything that can estimate/calculate the value of a phenomenon. Thus, either a device or computational process or a combination of them could play the role of a sensor. The representation of a sensor in the ontology links together what it measures (the domain phenomena), the physical sensor (the device) and its functions and processing (the models).

The OpenoT ontology is available as a single OWL file, and provides the means for a semi-automatically generated documentation. Additional annotations have been added to split the ontology into thematic modules. The implementation of the ontology and its integration in the OpenIoT architecture are realized through the LSM middleware. LSM transforms the data from virtual sensors into Linked Data stored in RDF (Resource Description Format), which is de facto queried using SPARQL. In the context of IoT applications in general and LSM in particular, such queries refer typically to sensor metadata and historical sensor readings. The SPARQL endpoint of LSM provides the interface to issue these types of queries. The RDF triple store deployed by LSM is based on OpenLink Virtuoso and provides a Linked Data query processor that supports the SPARQL 1.1 standard. While SPARQL queries are executed once over the entire collection and discarded after the results are produced, queries over Linked

Stream Data are continuous. Continuous queries are first registered in the system, and continuously executed as new data arrives, with new results being output as soon as they are produced. LSM provides a wide range of interfaces (wrappers) for accessing sensor readings such as physical connections, middleware APIs, and database connections. Each wrapper is pluggable at runtime so that wrappers can be developed to connect new types of sensors into a live system when the system is running. The wrappers output the data in a unified format, following the data layout described in the OpenIoT ontology.

2.3 Mobile Broker and Publish/Subscribe Middleware

OpenIoT offers support for discovering and collecting data from mobile sensors (e.g., wearable sensors, sensors built-in mobile devices). This is achieved through a publish/subscribe middleware titled CloUd-based Publish/Subscribe middleware for the IoT (CUPUS) which integrates: (1) A cloud-based processing engine for sensor data streams based on the publish/subscribe principles and (2) A mobile broker running on mobile devices for flexible data acquisition from mobile ICOs. In the OpenIoT architecture, CUPUS interfaces to the Cloud Database via X-GSN which annotates the data collected from mobile devices. Hence, data streams from mobile ICOs are annotated and stored in the OpenIoT cloud via X-GSN, similar to the way data streams from stationary sensors are announced via the X-GSN sensor middleware.

CUPUS supports content-based publish/subscribe processing, i.e., stateless Boolean subscriptions with an expressive set of operators for the most common data types (relational and set operators, prefix and suffix operators on strings, and the SQL BETWEEN operator), and continuous top-k processing over sliding windows i.e. a novel publish/subscribe operator which identifies k best-ranked data objects with respect to a given scoring function over a sliding window [9]. It facilitates pre-filtering of sensor data streams close to data sources, so that only data objects of interest, value and relevance to users are pushed into the cloud. The filtering process is not guided locally on mobile devices, but rather from the cloud based on global requirements. Moreover, CUPUS distributes in near real-time push-based notifications from the cloud to largely distributed destinations, e.g., mobile devices, based on user information needs.

As depicted in Fig. 2, a Mobile Broker (MB) running on a mobile device can connect to and disconnect from a publish/subscribe processing engine running within the cloud. On the one hand, a device with attached sensors acts as a data source: The MB announces the type of data it is able to contribute to the platform and adds the sensor to the Cloud Data Storage. On the other hand, mobile phone users can define continuous requests for data in the form of subscriptions. Based on existing requests for sensor data expressed through subscriptions by either mobile device users or the OpenIoT platform, the MB receives subscriptions from the publish/subscribe processing engine which become data filters to prevent potential data overload within the cloud. This mechanism ensures that only relevant data is transmitted from mobile

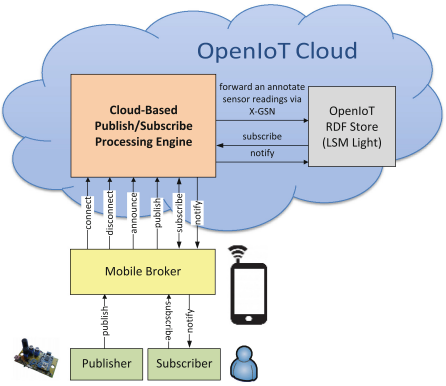

Fig. 2. High-level OpenIoT Publish/Subscribe Architecture

devices into the platform to be annotated and stored in the RDF repository, and subsequently to be transmitted in near real-time to adequate mobile devices.

Since the load of the publish/subscribe processing engine is generated by a varying number of publishers and subscribers with changing joint publication rate, the engine offers elastic real-time computation. It processes many subscriptions in parallel, which minimizes the processing overhead and optimizes the usage of cloud resources under varying load.

3 OpenIoT Platform Capabilities

3.1 Sensors and Data Streams Registration, Deployment and Discovery

OpenIoT manages the registration, data acquisition and deployment of sensors and interconnected objects, through X-GSN. X-GSN is an extension of the GSN that supports semantic annotation of both sensor data and metadata. The core fundamental concept in X-GSN is the virtual sensor, which can represent not only physical devices, but in general any abstract or concrete entity that observes features of any kind. A virtual sensor can also be an aggregation or computation over other virtual sensors, or even represent a mathematical model of a sensing environment.

In order to propagate its data to the rest of the OpenIoT platform, each virtual sensor needs to register within the LSM, so that other applications and users can discover them and get access to their data. The sensor is registered through X-GSN by posting a semantically annotated representation of its metadata. In order to associate metadata with a virtual sensor, a simple metadata descriptor is used. X-GSN takes care of creating the semantic annotations in RDF, according to the OpenIoT ontology, and posting them to the LSM cloud store repository.

```
sensorName=weather
source="Brussels netatmo"
sourceType=weather
sensorType=weatherType
information=Weather sensors in Brussels
author=openiot
feature="http://lsm.deri.ie/OpenIoT/BrusselsFeature"
fields="airtemperature"
field.airtemperature.propertyName="http://lsm.deri.ie/OpenIoT/AirTemperature"
field.airtemperature.unit=C
latitude=46.529838
longitude=6.596818
```

Listing 1. Sample metadata file for a X-GSN virtual sensor

Listing 1 illustrates the descriptor of a virtual sensor, which contains the location and the fields exposed by the virtual sensor. The descriptor includes the mapping between a sensor field (e.g., airtemperature) and the corresponding high-level concept defined in the ontology (e.g. the URI http://lsm.deri.ie/OpenIoT/AirTemperature).

```
@prefix : <http://sensordb.csiro.au/id/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix DUL: <http://www.loa-cnr.it/ontologies/DUL.owl#> .
@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
@prefix aws: <http://purl.oclc.org/NET/ssnx/meteo/aws#> .
@prefix prov:<http://purl.org/net/provenance/ns#> .
@prefix wgs84: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@base <http://sensordb.csiro.au/id/> .
<sensor/5010> rdf:type aws:CapacitiveBead,ssn:Sensor;
              rdfs:label "weather";
              ssn:observes aws:air_temperature ;
              ssn:onPlatform <site/narrabri/Pweather> ;
              ssn:ofFeature <site/narrabri/sf/sf_narrabri> ;
              prov:PerformedBy "SensorSource";
              DUL:hasLocation <place/location1>.
<place/location1> rdf:type DUL:Place;
                  wgs84:lat 52.3;
                  wgs84:long 98.2.
<site/narrabri/Pweather>  rdf:type ssn:Platform ;
                          ssn:inDeployment <site/narrabri/deployment/2013> .
```

Listing 2. Sample RDF snippet of a sensor annotation, in Turtle format

After the sensor has been registered, the corresponding RDF triples (Listing 2) are stored in LSM, and the sensor is available for discovery and querying from the upper layers of the OpenIoT architecture. Data acquisition for each virtual sensor is achieved based on wrappers that collect data through serial port communication, UDP

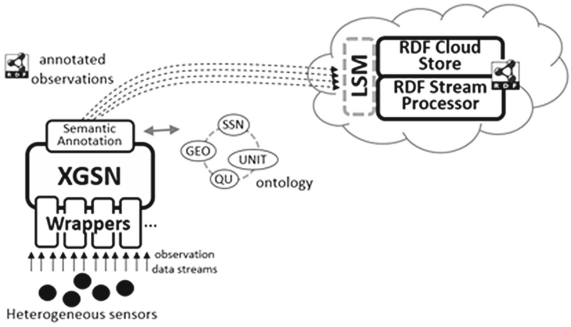

Fig. 3. Semantic annotation of observations in X-GSN

connections, HTTP requests, JDBC database queries, and more. X-GSN implements wrappers for these data providers, and allows users to develop custom ones. Virtual sensors and wrapper settings are specified in configuration files, which provide internal details of the data to be exposed. Data are represented as streams of data tuples which can be consumed, queried or analyzed on-line. In OpenIoT this processing includes the annotation of sensor observations as soon as they arrive to X-GSN, as depicted in Fig. 3. Note that virtual sensors can be built on top of other virtual sensors, providing different layers of information. For example, one can imagine a set of thermometers that send their data into X-GSN. Then all those data streams can feed an aggregating virtual sensor that averages received values over predefined time windows, annotates average values semantically and stores them in the LSM cloud store. The described example is realized by editing only a few XML files. In general, the effort needed to deploy a new sensor in OpenIoT is typically in the range of few man-hours.

3.2 Authenticated and Authorized Access to Resources

The diversity of applications interacting in an IoT ecosystem calls for non-trivial security and access-rights schemes. Conventional approaches (e.g., creating distinct user accounts for each application and granting access rights to each user) are not scalable as the number of applications and user accounts grows. OpenIoT adopts a flexible and generic approach for authentication and authorization. User management, authentication, and authorization are performed by the privacy & security module and its CAS (Central Authentication Service) service. Users are redirected to a centric login page the first time they try to access a restricted resource where they provide their username and password to the central authentication entity. If authentication is successful, the CAS redirects the user to the original web page and returns a token to the web application. Tokens represent authenticated users, have a predefined expiration time and are valid only before they expire. The token is forwarded from a service to the next one in a request chain, e.g., from the user interface to LSM. Services can check if the token is valid, or use the token to check if the user represented by this token has the necessary access rights.

In terms of implementation, OAuth2.0 enabled Jasig CAS has been extended for the OpenIoT needs. In particular, we added the end point permissions for retrieving

authorization information from CAS. Authorization information includes user roles/permissions. Permissions are textual values that define actions or behaviors and are defined per service. A wildcard permission format (Apache Shiro) is used. Permissions can consist of multiple levels delimited by colons, and levels can be defined by each application following a predefined pattern. For example, the permission string "admin:delete_role:SERVICE_NAME" has three levels: "admin" means that the permission is for administrative tasks, "delete_role" is the action, and "SERVICE_NAME" is the name of the service for which the action is permitted.

3.3 Zero-Programming Application Development

OpenIoT provides an integrated environment (i.e. OpenIoT IDE (Integrated Development Environment)) for building/deploying and managing IoT applications. OpenIoT IDE comprises a range of visual tools (Fig. 4) enabling: (a) Visual definition of IoT services in a way that obviates the need to master the details of the SPARQL language; (b) Visual discovery of sensors according to their location and type; (c) Configuration of sensor metadata as needed for their integration within the X-GSN middleware; (d) Monitoring of the status of the various IoT services, including the volumes of data that they produce and the status of the sensors that they comprise; (e) Visualization of IoT services on the basis of Web2.0 mashups (i.e. maps, line/bar charts, dashboards and more). These tools accelerate the process of developing IoT applications. In several cases simple applications can be developed with virtually zero programming.

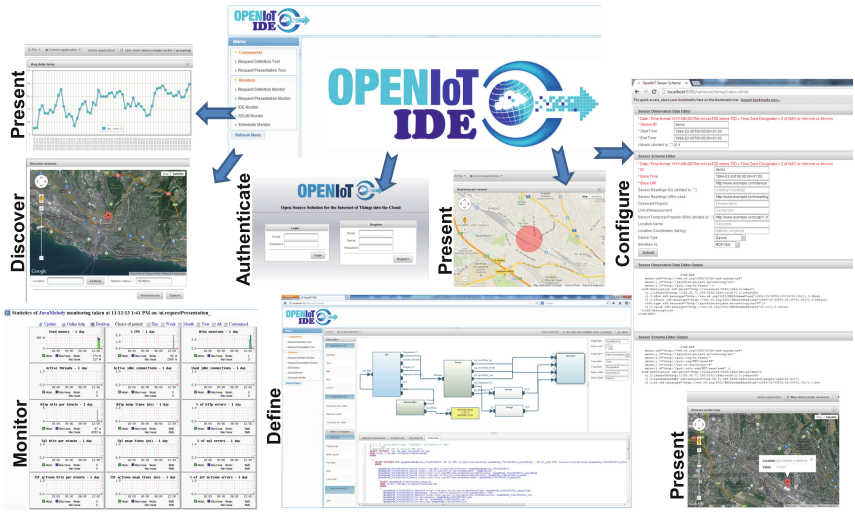

Fig. 4. Overview of the OpenIoT Integrated Development Environment (OpenIoT IDE)

3.4 Handling of Mobility with Quality Driven Sensor Management

As mobile crowd sensing applications generate large volumes of data with varying sensing coverage and density, there is a need to offer mobility management of ICOs and quality-driven mobile sensor data collection to satisfy global sensing coverage requirements while taking into account data redundancy and varying sensor accuracy [10]. CUPUS provides the means for collecting data from mobile ICOs, whose geographical location potentially changes while providing data to the cloud. As mobile brokers running on mobile devices announce the type of data that can be provided by their currently available publishers, they are configured so as to announce their available data sources each time they enter a new geographic area. Moreover, an X-GSN virtual sensor is created on demand for each new geographic area and is used to both push and annotate the data generated by all mobile sensors currently residing within its geographical area.

CUPUS addresses quality requirements (e.g., energy efficiency, sensing data quality, network resource consumption, latency), through smart data acquisition mechanisms. Firstly, by deploying mobile brokers on mobile devices, data can be selectively collected from external data sources attached to the mobile device and transmitted to the cloud only when required. Mobile brokers running in geographical areas where there are no currently active subscriptions will suppress data collection and refrain from sending unnecessary data into the cloud. Secondly, CUPUS is integrated with a centralized quality-driven sensor management function, designed to manage and acquire sensor readings to satisfy global sensing coverage requirements, while obviating redundant sensor activity and consequently reducing overall system energy consumption. Assuming redundant data sources in a certain geographic area, a decision-making engine is invoked to determine an optimal subset of sensors which to keep active in order to meet data requests while considering parameters such as sensor accuracy, trustworthiness, and battery level.

4 Proof-of-Concept Applications

4.1 Phenonet Experiment

Phenonet uses sensor networks to gather environmental data for crop variety trials at a far higher resolution than conventional methods and provides high performance real-time online data analysis platform that allows scientists and farmers to visualize, process and extract both real time and long-term crop performance information from the acquired sensor measurements. Figure 5 provides an example of a Phenonet experiment with two types of sensors (1) Gypsum block soil moisture sensors (GBHeavy) at various depths (e.g., 20, 30 and 40 cm) and (2) Canopy temperature measurement sensor.

The goal of the experiment is to monitor the growth and yield of a specific variety of crop by analyzing the impact of root activity, water use (soil moisture) and temperature. Information about crop growth obtained in real time effectively helps plant scientist researchers to provide estimates on the potential yield of a variety. OpenIoT facilitates the processes of real-time data collection, on-the-fly annotation of sensed data, data cleaning, data discovery, storage and visualization.

4.2 Urban Crowdsensing Application

This is a mobile application for community sensing where sensors and mobile devices jointly collect and share data of interest to observe and measure air quality in real-time. Volunteers carrying wearable air quality sensors contribute sensed data to the OpenIoT platform while moving through the city. Citizens are able to consume air quality information of interest to observe it typically in their close vicinity. Figure 6 shows air quality sensors measuring temperature, humidity, pressure, CO, NO2 and SO2 levels which communicate with the mobile application running on an Android phone via a Bluetooth connection. Users can declare interest to receive environmental data (e.g., temperature, CO levels) in their close vicinity and in near real-time. Moreover, they can express interest to receive the readings portraying poorest air quality for an area over time, or average readings within specific areas as soon as they are available.

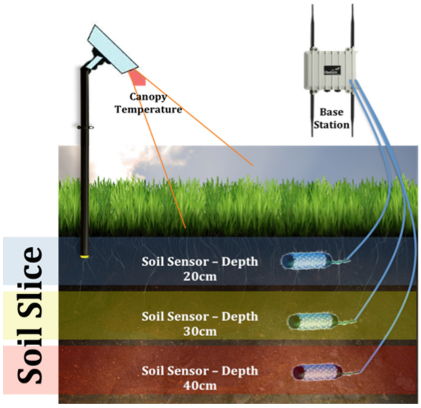

Fig. 5. Phenonet Experiment Illustration

Fig. 6. Air Quality Sensors and Mobile Application

4.3 Smart Campus Application

The smart campus application brings information about interactions among people and things within typical campus situations into one Common Information Model (CIM). This model combines observations from sensors with mobile applications and static structural information into one cyber-physical context managed by OpenIoT. In the prototype the used sensors are QR-code or NFC based scanners to detect and confirm the presence of persons and to identify assets and topics. The mobile applications are used for booking workplaces and for discussions. The structural information describes campus assets like buildings, rooms and workplaces, as well as teaching material. OpenIoT supports the stream oriented processing of events as well as context reasoning on the CIM.

5 Conclusions

OpenIoT has provided an innovative platform for IoT/cloud convergence which enables: (a) Integration of IoT data and applications within cloud computing infrastructures; (b) Deployment of and secure access to semantically interoperable applications; (c) Handling of mobile sensors and associated QoS parameters. The semantic interoperability functionalities of OpenIoT are a key differentiating factor of the project when compared to the wide range of other IoT/cloud platforms. These functionalities provide a basis for the development of novel applications in the areas of smart cities and mobile crowd sensing, while also enabling large scale IoT experimentation.

Acknowledgments. This work has been carried out in the scope of the OpenIoT project (http://openiot.eu). The authors acknowledge contributions from all partners of the project.

References

1. Hassan, M.M., Song, B., Huh, E.-N.: A framework of sensor-cloud integration opportunities and challenges. In: Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication (ICUIMC 2009), pp. 618–626. ACM, New York (2009)
2. Blair, G.S., Paolucci, M., Grace, P., Georgantas, N.: Interoperability in complex distributed systems. In: Bernardo, M., Issarny, V. (eds.) SFM 2011. LNCS, vol. 6659, pp. 1–26. Springer, Heidelberg (2011)
3. Compton, M., et al.: The SSN ontology of the W3C semantic sensor network incubator group. J. Web Semant. **17**, 25–32 (2012). Elsevier
4. Heath, T.: Linked data - welcome to the data network. IEEE Internet Comput. **15**(6), 70–73 (2011). IEEE Press, New York
5. Aberer, K., Hauswirth, M., Salehi, A.: Infrastructure for data processing in large-scale interconnected sensor networks. In: International Conference on Mobile Data Management (MDM 2007), pp. 198–205. IEEE Press, New York (2007)
6. Black Duck Software: Black Duck Announces Open Source Rookies of the Year Winners, Press Release, January 2013

7. Serrano, M., Hauswirth, M., Soldatos, J.: Design principles for utility-driven services and cloud-based computing modelling for the internet of things. Int. J. Web Grid Serv. **10**(2–3), 139–167 (2014). Inderscience Publishers, Geneva
8. Le Phuoc, D., Nguyen-Mau, H.Q., Parreira, J.X., Hauswirth, M.: A middleware framework for scalable management of linked streams. J. Web Semant. **16**, 42–51 (2012). Elsevier
9. Pripužić, K., Podnar Žarko, I., Aberer, K.: Top-k/w publish/subscribe: a publish/subscribe model for continuous top-k processing over data streams. Inf. Syst. **39**, 256–276 (2014). Elsevier
10. Antonić, A., Rožanković, K., Marjanović, M., Pripužić, K., Podnar Žarko, I.: A mobile crowdsensing ecosystem enabled by a cloud-based publish/subscribe middleware. In: 2014 International Conference on Future Internet of Things and Cloud (FiCloud), pp. 107–114. IEEE Press, New York (2014)