*Sensors & Transducers, Vol. 185, Issue 2, February 2015, pp. 100-112*

www.sensorsportal.com

**Sensors & Transducers**
© 2015 by IFSA Publishing, S. L.
*http://www.sensorsportal.com*

# Contemporary Analysis and Architecture for a Generic Cloud-based Sensor Data Management Platform

**Vincent C. Emeakaroha, Kaniz Fatema, Philip Healy, John P. Morrison**
Irish Centre for Cloud Computing and Commerce (IC4)
University College Cork, Ireland
E-mail: vc.emeakaroha@cs.ucc.ie, k.fatema@cs.ucc.ie, p.healy@cs.ucc.ie, j.morrison@cs.ucc.ie

*Received: 14 November 2014 /Accepted: 15 January 2015 /Published: 28 February 2015*

**Abstract:** An increasing volume of data is being generated by sensors and smart devices deployed in different areas, often far from computing facilities such as data centres. These data can be difficult to gather and process using local computing infrastructure. This is due to cost and limited resources. Cloud computing provides scalable resources that are capable of addressing such problems. However, platform-independent methods of gathering and transmitting sensor data to Clouds are not widely available. This paper presents a state-of-the-art analysis of Cloud-based sensor monitoring and data gathering platforms. It discusses their strengths and weaknesses and reviews the current trends in this area. Informed by the analysis, the paper further proposes a generic conceptual architecture for achieving a platform-neutral Cloud-based sensor monitoring and data gathering platform. We also discuss the objectives, design decisions and the implementation considerations for the conceptual architecture. *Copyright © 2015 IFSA Publishing, S. L.*

**Keywords:** Sensor Monitoring Platforms, Sensor Data Gathering, Cloud Computing, Generic Sensor Cloud Platform, Interoperable Communication.

## 1. Introduction

With the increase in the number of sensors and smart devices that are capable of generating data from their usage, data gathering IT techniques such as those supporting control systems are showing commercial potentials in controlling at real-time data transfer from real-world devices. Based on this, more generalized applications for sensor devices are becoming significant in real world usages. Sensors enable access to remote objects and environmental information to provide the raw materials, through simple monitoring, for the next generation applications such as smart cities. The procedures of gathering, storing and processing of sensor generated data, using local computing infrastructures are becoming very costly due to scalability constraints resulting from limited resources and the volume of the generated data.

In a similar fashion, the Internet of Things (IoT), [1] that promises to connect objects, devices and humans, generates large volumes of data. Harnessing these data by organisations can be a complex process due to heterogeneous operating systems, varying connectivity protocols and legacy application compatibility. Furthermore, the ability to draw meaningful insights from the voluminous data unleashed by these technologies is another challenge especially in terms of compute power and analytic tool availability.

Cloud computing platforms offer scalable compute and storage resources to support the management of such sensor data [2]. However, solutions available today [3-6] are mostly customised for particular usages. To unlock the business potential in this area, generic solutions are required to address the core challenges such as communication bottleneck, data interchange formats, security, energy efficiency and interoperability.

In this paper, we present a state-of-the-art analysis of Cloud-based sensor monitoring platforms and data gathering techniques. This work is an extension of our initial approach presented in [7]. Informed by the analysis, the paper further proposes a platform-neutral architecture providing a generic Cloud interface that is focused on addressing the identified challenges. The main contributions are

1) A review of industrial, commercial and open source sensor monitoring and data management platforms;

2) Analysis of the platforms to identify trends, issues and challenges;

3) The proposal of a generic architecture utilizing standardized data interchange formats and interoperable communication mechanisms to achieve interoperability among heterogeneous platforms.

The rest of the paper is organized as follows: Section 2 presents the related work. The state-of-the-art analysis is performed in Section 3. Section 4 presents an overview summary of the analyzed platform features and discusses the challenges and trends derived from the analysis. In Section 5, we describe the relevant challenges and objectives for a platform-neutral architecture while Section 6 presents the proposed architecture and its implementation considerations. In Section 7, we conclude the paper and discuss follow-up work.

## 2. Related Work

In this area, previous efforts have been focused mostly on particular issues like the virtualization of physical sensors on Clouds, data privacy and security or the provision of efficient platforms for sensor data storage and processing. Catrein, *et al.* [8] propose a Cloud design for user-controlled storage and processing of sensor data to ensure privacy. They identified the importance of using the Cloud for sensor data processing. However, their approach focuses on security-related issues such as data privacy and access control. Aoki, *et al.* [9] present a Cloud architecture to enable fast response to real world applications despite the flood of sensor data. The authors used the strategy of reducing network latency to achieve this goal but they did not consider creating a generic interface for diverse sensor data gathering.

Piyare, *et al.* [10] propose an architecture for integrating Wireless Sensor Networks (WSN) into Cloud services for real time data collection. In their approach, WSN is considered an important paradigm for Internet of Things since they consist of smart sensing nodes with embedded Central Processing Units (CPU) and sensors for monitoring different environments. This work concentrated on connecting WSNs to Clouds and does not consider other sensor types. The authors in [3-4] discuss particular approaches for gathering medical sensor data. Alamri, *et al.* [11] present a survey on sensor-Cloud, architecture, applications and approaches. The survey analyses the current efforts and challenges in this area. It shows that many of the existing efforts are geared towards creating virtual sensors from physical ones on Clouds.

Salehi, *et al.* [12] present Global Sensor Network (GSN), which is a middleware to interconnect diverse sensor network technologies. This work focus on providing a mechanism for easy integration of existing sensor networks. With this approach, the management of sensor networks is simplified. For example, changing or updating components within a sensor network does not interfere or hinder communications with other sensor networks. However, it does not consider the gathering of single sensor data. Other efforts like the Sensor Observation Service (SOS) [13] provide standards in accordance with the Open Geospatial Consortium (OGC) for discovery and retrieval of real-time data from diverse sensors in the context of geospatial data processing and sharing.

Thus, to the best of our knowledge, none of the existing work provides a generic Cloud-based monitoring platform for gathering sensor data. In the next section, we present our analysis of the state of the art.

## 3. State-of-the-Art Analysis

This section details the review of the existing tools and platforms for monitoring, gathering and processing sensor data. It highlights their characteristic features and practical use cases.

### 3.1. IBM Mote Runner

Mote Runner is IBM's infrastructure for WSNs [14]. It is based on a virtual machine targeted to resource-constrained hardware environments and consists of two parts: a run time for mote-class hardware, such as Libelium Waspmote or MEMSIC Iris motes, and a development environment for WSN applications.

At its core, Mote Runner is designed to run on very small, standard, embedded controllers, including low-power 8-bit processors, thereby reducing initial investments as well as post-deployment and maintenance costs. It has been shown to be light in energy consumption [15]. It provides a high-level, language-friendly, resource-efficient and high-performance Virtual Machine (VM) that shields portable applications from hardware specifics. In addition, Mote Runner allows programmers to use object-oriented programming languages such as Java and a development environment based on Eclipse to develop portable WSN applications that may be dynamically distributed. Its features include [16]:

- Low-power;
- Support for harvesting solar power as a source of energy;
- Wirelessly connected embedded systems;
- Provides Software Development Kits (SDK) for developers;
- Supports multiple high-level languages such as Java and C#;
- Runs on 8 bit micro-controllers with as little as 4 kB of Random Access Memory (RAM) and 32 kB of flash memory;
- Leverages integrated development environments such as Eclipse, Visual Studio and MonoDevelop.

This tool focuses on WSN related applications and mote actuators. However, little or no information was available about its support for wired or other sensor types. The details about the used data interchange formats are not publicly available.

### 3.2. SensorCloud Platform

MicroStrains SensorCloud offers a sensor data storage, visualization and remote management platform that leverage Cloud computing technologies to provide data scalability and rapid visualization [17]. It was initially designed to support long-term deployments of MicroStrain wireless sensors. However, it now supports any web-connected third party devices, sensors, or sensor networks through a simple OpenData Application Programming Interface (API). It aims to provide virtually unlimited storage for the sensor data [18]. Its features include:

- Presumably unlimited data storage with triple-redundant reliability,
- A time series visualization and graphing tool;
- A MathEngine analytic tool facilitating quick user application development using their data on the Cloud;
- Provides Short Message Service (SMS) and Electronic mail (E-mail) alerting capabilities;
- Provides OpenData API and Representational State Transfer (REST) API for data transport;
- Supports only eXternal Data Representation (XDR) and Comma Separated Value (CSV) data interchange formats.

Some of its usage scenarios include structural health monitoring and condition based monitoring of high value assets where commonly available data tools are often not capable in terms of accessibility, data scalability, programmability, or performance. All communication with the SensorCloud is performed over Secure HyperText Transfer Protocol (HTTPS), which is secure. It restricts however, other forms of interactions such as low-level communication thereby forcing the use of HyperText Transfer Protocol (HTTP)-capable middleware to connect the devices and the platform. Furthermore, XDR and CSV are the only data interchange format types currently supported.

### 3.3. Ostia Portus Platform

Ostia Portus is designed to mediate between multiple vendor technologies where each vendor has particular platforms, databases and programming languages. It achieves this by taking the isolated data sets from individual systems and packaging them into a standard service [19]. This platform connects a variety of devices including sensors, networks and platforms. Its features include:

- Support for relational databases;
- Uses Secure Socket Layer (SSL) over HTTP to achieve security;
- Supports Simple Object Access Protocol (SOAP), REST, Publication/Subscription (PUB/SUB) proto- cols;
- Supports Java Message Service (JMS), RabbitMQ, Transmission Control Protocol (TCP)/Internet Protocol (IP), Message Queue (MQ) transport protocols;
- Easy installation and use.

Portus is built using open technologies and exploits open standards in accessing organizations data and presenting them using business defined views. Its core components are:

1) Server written in C and C++ and hosted by Apache;

2) Control centre for administration that is written in Java and built to run with the Eclipse Framework and (ii) front-end providing web services.

### 3.4. TempoIQ Platform

This platform is formerly known as TempoDB. It provides a real-time sensor monitoring service aiming to store and analyze time series data from sensors, smart meters, servers and automotive telematics. It is a commercial tool consisting of the following features [20]:

- Flexible monitoring and alerting;
- Powerful and custom analytics;
- Simple REST API for data storage and retrieval;
- Allows data storage at full resolution (no down sampling);
- Guarantees data availability with its triple data replication;
- Offers SSL encryption for all data transfer;
- API clients available in multiple programming languages like Java, .NET, Python, Ruby;
- Supports Internet of things.

The primary aim of this platform is the management of time series data sets with timestamps in ISO8601 format [21]. In querying data from this platform storage using a client API, the returned data is formatted only as JavaScript Object Notation (JSON). Other data interchange formats are currently not available. The platform has been upgraded to offer flexible sensor data monitoring and alerting mechanism. The alerts are based on thresholds and inform the user about the status of its applications that are using the sensor data. It offers also analytic tools to support user applications.

### 3.5. FreshTemp Temperature Monitor

FreshTemp is a Cloud-based monitoring system for perishable goods [5]. It automates temperature collection during production, transportation and storage of any perishable product by providing the capability of integrating with the different temperature sensors monitoring such products. It offers real-time data logs, configurable alerts and an online Dashboard. It is a commercial tool aiming to provide solutions for food services, transportation, health care and industrial usages. Its core features include:

- Wireless temperature monitoring;
- Bluetooth food probes;
- Bluetooth data loggers;
- Real time data logs;
- Alerting mechanism with SMS E-mail and phone;
- Online dashboard.

This tool focuses solely on temperature sensors, which limits its usability for managing and controlling other sensor device types. Furthermore, it does not offer any programming API for application development or any means of accessing the data stored on the platform by developers.

### 3.6. SensaTrack Monitor

SensaTrack is a multi-platform independent monitoring service that is ideal for Machine-to-Machine (M2M) sensor monitoring. It is developed by Cannon Water Technology Inc. and released in November 2012 [6]. The SensaTrack Cloud- based monitoring software solution allows users to visualize sensor data from any web-enabled device. It is designed for enterprises with distributed assets like chemical storage facilities, bulk storage tanks, bins and silos. Its features include:

- Ability to quickly scan multiple locations;
- Secure data servers;
- Wireless data gateways;
- Track trends and find problems early;
- Easy installation by non-technical personnel.

SensaTrack uses wireless communications equipment based on Zigbee protocols developed by the Digi Corporation. It also supports hybrid networks of wired and wireless sensors. This tool has no support for application developments and does not provide an API for external access.

### 3.7. Bluwired S-Cloud Platform

Bluwired S-Cloud provides a platform for sensor data exploration, interaction and analysis [22]. It facilitates the management of sensor and device data from any web enabled location in the world, and to deploy data processing and analysis applications that rely on gathered data on the Cloud. Its features include:

- A platform for sensor data exploration, interaction and analysis;
- Management of wireless sensor data from any web enabled location in the world;
- Support for user development and deployment of data processing and analytic applications using their data on the cloud;
- Facilitates real-time data storage and retrieval;
- Provides alerting mechanism for abnormal events;
- Provides visual management interface.

Bluwired S-Cloud promises to offer reliable storage, tracking and analysis of sensor data that comes from monitoring and control solutions for most applications, including: Factory Automation, Process Control, Agriculture and Irrigation monitoring, Patient Monitoring Systems, Oil and Gas. It is a commercial tool and does not offer an open source API for accessing the platform.

### 3.8. Xively Platform

Xively is a Cloud Service platform that harnesses the power of IoT to quickly and easily transform connected product vision into market reality. It was formerly known as “pachube” and later as “COSM” [23]. Xively is currently a division of LogMeln Inc. and strives to provide business solutions through the IoT. It offers a platform that connects devices and products with applications to provide real-time control, management and storage. Its features include:

- Secure real-time messaging;
- Time series data storage;
- Selective data sharing;
- Provides easy connection to external Cloud services like Twitter and Facebook;
- Encryption with Transport Layer Security (TLS) and SSL;
- Real-time message bus based on Message Queue Telemetry Transport (MQTT).

An example use case for Xively is the “Park-A-Lot” project [24], which is designed to support an automated parking management system. Although a commercial product, it provides open APIs and libraries for easy usage by developers to create smart applications and interacts with the platform. Xively provides effective means to reason about devices and actuators at high level but fails to provide detailed context information within which all these devices are being placed, especially when it comes to small-scale setups such as individual houses.

## 3.9. Nimbits Platform

Nimbits is a platform as a service (PaaS) for developing software and hardware solutions that seamlessly connect to the Cloud and each other. It has the ability to record and share sensor data on the Cloud [25]. Within Nimbits, sensor data are stored as data points using textual, JSON or eXtensible Markup Language (XML) formats. It provides REST web services for logging and retrieving time and geo stamped data (such as a reading from a temperature sensor). Nimbits servers can run on both powerful Cloud platforms like Google App Engine and on the smallest Raspberry Pi device. Its graphic user interface is tree structured having parent and child structures, which allow user content to be organized according to a parent-child structure and could be dragged and dropped as desired [26]. Its features include:

- Ability for recording and sharing data;
- Data storage as data points;
- Easy connection of data to analytic tools;
- Graphic user interface for visualization;
- Ability to generate alerts based on defined thresholds or events.

Nimbits is an open source platform for the Internet of things. It is freely available and provides libraries, APIs and documentations for different programming languages.

## 3.10. ThingSpeak Platform

ThingSpeak is an open source Internet of things platform that provides API to store and retrieve data from things using HTTP over the Internet or via a Local Area Network [25]. It has the ability to facilitate the creation of sensor logging applications, location-tracking applications, or a social network of things providing status updates.

In excess to its ability to store and retrieve numeric data and alphanumeric data, its API allows for numeric data processing such as time scaling, averaging, median, summing and rounding. ThingSpeak is organized in Channels, which is where a user application can store and retrieve data. Each Channel supports data entries of up to 8 data fields. The channel feeds support JSON, XML, and CSV data formats for integration into applications. Its features include [27]:

- Open API for developers;
- Real-time data collection;
- Geolocation data gathering;
- Data processing and analytic tools;
- Data visualizations on web and mobile devices;
- Device status messages and event alerting;
- Supports diverse programming languages like Java;
- JavaScript, .Net, Ruby;
- Allows easy plugins integrations.

ThingSpeak offers also a hosted service that is different to the open source version. Open.Sen.se [10] is another Internet of Things tool that is very similar in characteristics and features to ThingSpeak.

## 3.11. Microsoft Azure Intelligent System Service

The Microsoft Azure Intelligent Systems Service aims to securely connect, manage and capture machine-generated data from industry devices, sensors and other line-of-business (LoB) assets across a range of operating system platforms. The intelligent service represents the efforts of Microsoft to address the challenges of IoT and to help businesses utilize its potentials.

This tool promises to offer enterprises the ability to extend their Microsoft Azure Cloud across connected devices and sensors in order to capture vital data, analyze them with familiar Microsoft tools like HD Insight and Power BI for Office 365 in order to facilitate taking quick and appropriate actions that drive impact. Its features include [28]:

- Secure connection and management of devices and data;
- Support for real-time control of heterogeneous environments and accelerated implementations;
- Supports efficient capture, store, join, analyze, visualize and share data;
- Provide a trusted platform that can be extended easily for industrial specific requirements;
- Improve operations and unlock new business opportunities by harnessing machine-generated data from connected sensors and actuators.

The intelligent service is fully commercial software as the other Microsoft services however, it is not yet production ready. A limited preview version was released in April 2014.

## 3.12. Paho Platform

The Paho project aims to provide scalable open-source implementations of open and standard messaging protocols to facilitate new, existing, and emerging applications to enable machine-to-machine and Internet of things usage scenarios. It is a part of the Eclipse foundation and its features include:

- Enables levels decoupling between devices and applications;

- Encourages the growth of scalable web and enterprise middleware and applications;
- Supports resource constrained embedded platforms;
- Based on MQTT;
- Provide MQTT client implementations in Java, Python, C, C++;
- Provide open libraries, API and client implementations;
- Enables integration of wide range of middleware, programming languages and messaging models.

Paho strives to be software for constrained networks, devices with limited processing resources and embedded platforms [29].

### 3.13. SicsthSense

SicsthSense is an open Cloud platform for the Internet of Things that enables low power devices such as sensor nodes and smartphones to easily store their generated data in the Cloud. SicsthSense incorporates real data collection and a means of pushing them to a data store, which is then visualized and made available for sharing between users of the platform. It possesses the following features [30]:

- Designed to ease the interconnection of the billions of sensors and actuators to the Cloud;
- Devices are registered, discovered, configured, and programmed from the Cloud, either through a Web interface or an open machine-to-machine API;
- Stores/retrieves sensor data in the Cloud and makes decisions using Big Data analysis techniques and visualizes data;
- SicsthSense supports the HTTP and Cap proto- cols (a specialized web transfer protocol to use with constrained nodes and constrained networks in the Internet of Things);
- Communication with the API is performed using JSON;
- Storage is provided by two subsystems: in a normal RDBMS, which is specifically designed to enable easy searching for relevant feeds and in a highly scalable asynchronous master less replication database system.

SicsthSense also provides features for basic actuation, allowing the system to affect the physical device when the incoming data streams meet certain conditions. For instance, when the average temperature of a user’s house goes below 15C, SicsthSense can signal the heater to switch on. It enables centralization of computation, monitoring, control and redistribution of collected data.

### 3.14. Relay

Relay provides Open Sensor Cloud platform [31] that collects data from sensor devices called Underbars. A Wonderbra contains six detachable Beacons sensors, which can be monitored and controlled by smart phones. These sensor modules communicate with the Master Module by BLE (Bluetooth Low Energy) and the Master Module connects with the Internet by Wi-Fi. The modules are designed to make programming of sensor devices very easy. The composed sensors include: light, color, distance, temperature, humidity, remote control etc. Relay platform’s features include:

- It provides a platform enabling users to use the data gathered by sensors and other devices regardless of the data format and data gathering method;
- It enables an indirect connection of sensors and applications and it ensures data accessibility regardless of location and distance;
- The platform receives data from the devices, stores it and distributes it only to the applications which incorporate the Relay SDK and meet the authorization criteria established when first registering on the platform;
- It also translates BLE data into JSON/MQTT - enabling both the SDKs and the Cloud Platform to communicate with the sensor modules;
- The relayr cloud platform allows data normalization.

The Relay Open Sensor Cloud Platform and the Wonderbra provide means to easily develop applications for the physical world. In its operation, each Master Module is authenticated during the registration process to enable it to receive data from the specific sensors. The Master Module is the only module capable of connecting to Wi-Fi and is responsible for delivering data from the sensors to the cloud and from the cloud to the sensors. Relay also provides credential and rule based authentication via OAuth and SSL. However, it currently does not emphasize much on the analysis methods of the gathered data.

In the next section, we summary the features of the above described platforms and discuss the identified challenges and trends.

## 4. Result Summary and Comparison

In this section, we present an overview summary of the identified platform features and drawbacks.

This facilitates a quick comparison between them. Furthermore, this section offers us also the opportunity to identify and discuss trends, and the gaps in technological advancements in this area. A goal of this analysis is to organize our thoughts and to see what is available in order to design a conceptual architecture.

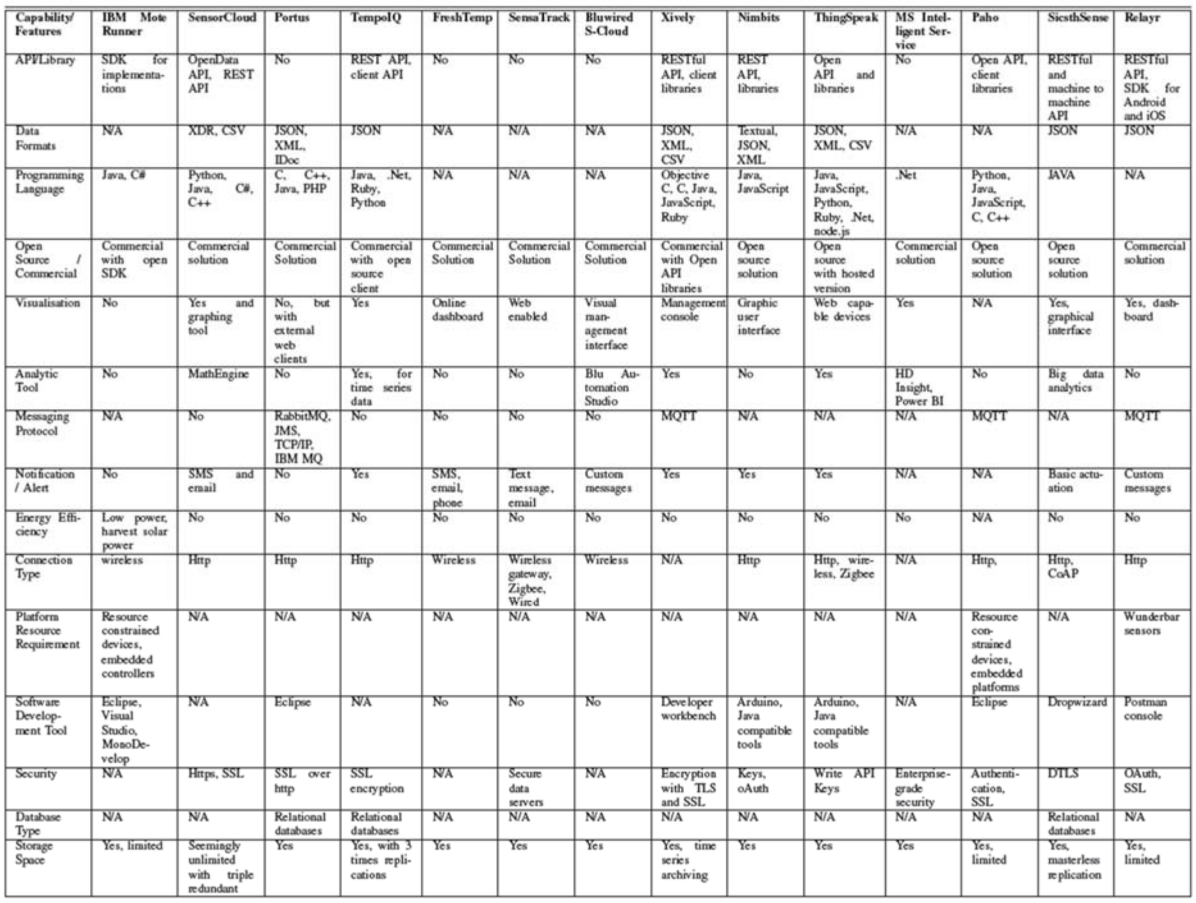
The described and analyzed platforms represent the current efforts towards addressing the challenges of Cloud-based sensor monitoring and data gathering.

As can be observed from the platform descriptions most of these challenges are not yet adequately addressed to facilitate informed decision making and support smart applications. Table 1 presents a summarized version of the platform analysis.

As shown in Table 1, the analyzed platforms exhibit some similar characteristics in terms of data storage, support for web technologies and availability of RESTful APIs. The commercial platforms are closed source, thus, details regarding programming languages, API, data interchange formats were not accessible. It can be observed that the platforms lack support for resource-constrained deployments, energy efficiency, messaging protocol implementation, software development environments and data interchange formats. These issues represent the challenges facing the existing platforms.

In the following sections, we highlight and discuss some of them in detail.

**Table 1.** Analyzed Platform Features.



| Capability/ Features | IBM Mote Runner | SensorCloud | Portus | TempoIQ | FreshTemp | SensaTrack | Bluwired S-Cloud | Xively | Nimbits | ThingSpeak | MS Intelligent Service | Paho | SicsthSense | Relayr |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| API/Library | SDK for implementations | OpenData API, REST API | No | REST API, client API | No | No | No | RESTful API, client libraries | REST API, libraries | Open API and libraries | No | Open API, client libraries | RESTful and machine to machine API | RESTful API, SDK for Android and iOS |
| Data Formats | N/A | XDR, CSV | JSON, XML, IDoc | JSON | N/A | N/A | N/A | JSON, XML, CSV | Textual, JSON, XML | JSON, XML, CSV | N/A | N/A | JSON | JSON |
| Programming Language | Java, C# | Python, Java, C#, C++ | C, C++, Java, PHP | Java, .Net, Ruby, Python | N/A | N/A | N/A | Objective C, C, Java, JavaScript, Ruby | Java, JavaScript | Java, JavaScript, Python, Ruby, .Net, node.js | .Net | Python, Java, JavaScript, C, C++ | JAVA | N/A |
| Open Source / Commercial | Commercial with open SDK | Commercial solution | Commercial Solution | Commercial with open source client | Commercial Solution | Commercial Solution | Commercial Solution | Commercial with Open API libraries | Open source solution | Open source with hosted version | Commercial solution | Open source solution | Open source solution | Commercial solution |
| Visualisation | No | Yes and graphing tool | No, but with external web clients | Yes | Online dashboard | Web enabled | Visual management interface | Management console | Graphic user interface | Web capable devices | Yes | N/A | Yes, graphical interface | Yes, dashboard |
| Analytic Tool | No | MathEngine | No | Yes, for time series data | No | No | Blu Automation Studio | Yes | No | Yes | HD Insight, Power BI | No | Big data analytics | No |
| Messaging Protocol | N/A | No | RabbitMQ, JMS, TCP/IP, IBM MQ | No | No | No | No | MQTT | N/A | N/A | N/A | MQTT | N/A | MQTT |
| Notification / Alert | No | SMS and email | No | Yes | SMS, email, phone | Text message, email | Custom messages | Yes | Yes | Yes | N/A | N/A | Basic actuation | Custom messages |
| Energy Efficiency | Low power, harvest solar power | No | No | No | No | No | No | No | No | No | No | N/A | No | No |
| Connection Type | wireless | Http | Http | Http | Wireless | Wireless gateway, Zigbee, Wired | Wireless | N/A | Http | Http, wireless, Zigbee | N/A | Http, | Http, CoAP | Http |
| Platform Resource Requirement | Resource constrained devices, embedded controllers | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | Resource constrained devices, embedded platforms | N/A | Wunderbar sensors |
| Software Development Tool | Eclipse, Visual Studio, MonoDevelop | N/A | Eclipse | N/A | No | No | No | Developer workbench | Arduino, Java compatible tools | Arduino, Java compatible tools | N/A | Eclipse | Dropwizard | Postman console |
| Security | N/A | Https, SSL | SSL over http | SSL encryption | N/A | Secure data servers | N/A | Encryption with TLS and SSL | Keys, oAuth | Write API Keys | Enterprise-grade security | Authentication, SSL | DTLS | OAuth, SSL |
| Database Type | N/A | N/A | Relational databases | Relational databases | N/A | N/A | N/A | N/A | N/A | N/A | N/A | N/A | Relational databases | N/A |
| Storage Space | Yes, limited | Seemingly unlimited with triple redundant | Yes | Yes, with 3 times replications | Yes | Yes | Yes | Yes, time series archiving | Yes | Yes | Yes | Yes, limited | Yes, masterless replication | Yes, limited |

## 4.1. Openness, API and Library Implementation

The characteristic feature of being open source and providing API or libraries is essential to facilitate rapid development. As can be observed in Table 1, the commercial tools promise many features and functionalities, but details of their implementations are completely hidden from the public, which slows down technical know-how establishment and thereby makes it difficult for developers to leverage such functionalities when creating applications. This problem obstructs market growth and poses challenges to technology adoption.

Fig. 1 presents a graphical description of those feature analyses. The figure combines the analysis of the open source and API/Library features of the identified existing platforms. According to the figure, only 28 % of the analyzed platforms are fully open source projects, which show the growing interest in this area to provide standardized software stacks for gathering and processing sensor data. The analysis also shows 36 % for both partially open and closed source. Partially open represents situation where the platform provides only open API for interfacing. These results indicate the need to strive for more openness especially regarding the completely closed platforms. We are not proposing that everything should be open source but platforms should provide open and well-defined APIs. Providing open source APIs and interfaces enables easy creation of software and assures adherence to defined standards.

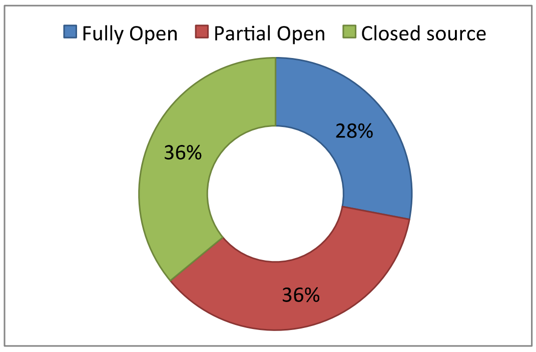


**Fig. 1.** Opennes, API and Library Analysis.

### 4.2. Interoperability

There is currently a broad spectrum of sensors and smart devices. This constitutes heterogeneous environments with different systems and data types. The heterogeneity of this environment has proven to hinder the rate of accessing and gathering sensor data. To disentangle this issue, requires the development of generic solutions to facilitate interoperability. A means to achieve this is by applying appropriate data interchange formats to realize a common data format that could be used by multiple platforms.

As shown in Table 1, most of the open source analyzed tools are using textual data interchange formats, which have the advantage of being human readable and easy to understand. However, the serialized data in those formats are not very compact in size for efficient transportation without consuming large amounts of bandwidth. This implies that further efforts are required to improve the state-of-the-art. For example, by integrating standardized binary data interchange formats that have the ability to achieve compact serialization of data.

Fig. 2 illustrates a graphical analysis of the interoperability levels supported by the analyzed platforms. This graphic is generated by merging the analysis for data interchange formats and programming languages since these two features contributes highly to the realization of interoperability and portability of a software solution.

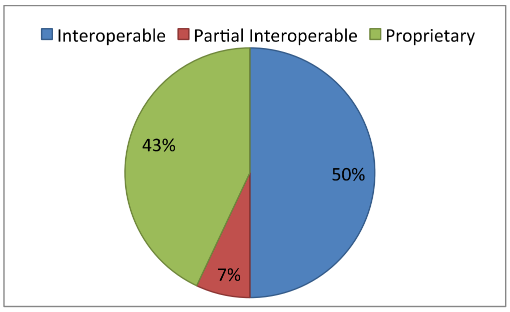


**Fig. 2.** Interoperability Analysis.

As depicted in Fig. 2, 50 % of the analyzed platforms are categorized as interoperable, which means that they are applying a standard data interchange format in formatting and serializing their data. This is an encouraging result that highlights the interest and importance of this feature for the industry. However, Fig. 2 also shows that 43 % of the platforms are using proprietary data interchange formats and therefore, are not interoperable. Thus, there are requirements for further improvements in this area to achieve higher levels of interoperability in gathering and processing sensor generated data.

### 4.3. Messaging Protocol

The ability to notify or pass control information as messages from Cloud platforms back to sensors and smart devices is still a challenge, as can be observed in the overview summary of Table 1. This brings to light that most of the existing platforms supports only one-way communication and do not provide feedback to the underlying sensors generating the data. It is gradually becoming an essential requirement by applications to facilitate dynamic actuation of sensors in order to update or re-configure them.

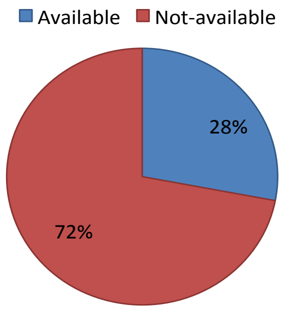


**Fig. 3.** Messaging Protocol Analysis.

Fig. 3 presents a graphical illustration of the level of messaging protocol implementations by the analyzed platforms. Informed by this figure, 78 % of the analyzed tools do not implement any messaging protocol or have a hidden one not mentioned in their documentations. This is a big surprise to us with regards to the high importance of this feature. Most of the modern applications using multiple sensors re-quire the ability to dynamically perform configurations at run-time. The fulfillment of this requirement is lacking in the existing platforms thereby necessitating further developments in this direction.

### 4.4. Energy Efficiency

Energy efficiency considerations are important aspect of sensors and smart devices management since most of them rely on batteries and small energy sources. Much research efforts [32-35] have been focused at the infrastructure levels on sustaining sensor battery life to achieve longer durations and maintain consistent performance.

However, at the application levels where sensor generated data are being gathered, processed and utilized, energy efficiency considerations are treated with less priority. This is evident in Table 1. Fig. 4 depicts a graphical representation of this analysis.

According to Fig. 4, 93 % of the analyzed platforms do not consider energy efficiency in their operations. This underlines the importance for more research effort in this aspect since the number of smart applications utilizing multiple sensors is continuously increasing. It has been shown in data center energy management approaches [36-37] that the behaviour of applications in terms of resource consumptions affects the amount of energy consumed by the underlying hardware. Thus, energy efficiency requirements should be considered at the application layers in order to appropriately control and manage the energy consumptions, which could help reduce the total IT energy consumption levels that are presently high.

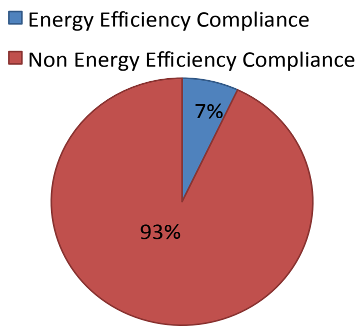


**Fig. 4.** Energy Efficiency Analysis.

### 4.5. Analysis Summary

This section provides a short summary to the analysis.

In general, the analyzed commercial platforms tend to surface many features. But, they are mainly closed source and implement proprietary technologies instead of the standardized ones. Most of them also do not provide open-source API, which hinders interoperability and ease of adoption.

On the side of the open source solutions, they provide basic features and are not quite advanced. Another interesting point is that most of the open source tools are developed in the context of Internet of Things. This means that they are designed for multiplicity and could be easily used in designing generic solutions.

Furthermore, the existing platforms provide poor support for resource-constrained devices, software development kits, energy efficiency and messaging protocol. These challenges call for further efforts and development of standardized architectures. We view the open source tools as providing a good basis for this purpose.

In the next section, we further highlight the identified challenges and discuss the aims of our proposed generic architecture.

## 5. Issues and Objectives

In this section, we discuss the challenges informed from the state-of-the-art analysis and present our objectives for the proposed architecture.

### 5.1. Challenges and API Requirements

As can be observed in Table 1, none of the analyzed platforms provide a complete set of features representing a generic and open solution. The implementation of such a generic platform is complex and difficult. Many challenges exist in this spectrum especially regarding the heterogeneity of the targeted hardware and software components. At the lower levels, there is a plethora of sensor devices out there, which gather data in different formats. To design a generic interface for these data types requires a comprehensive study of the existing data types and how they could be aggregated or adapted/converted to a standard format.

At the higher levels, another challenging factor is the security and data access control on the Cloud platform. The authentication of a gateway server before allowing the forwarding of the sensor data to the Cloud platform is not enough. Many customers worry more about the privacy of their data and what the Cloud providers might do with them on the Cloud. To address these issues, different levels of security assurances are required for securing the data on the Cloud and controlling their usage.

Furthermore, the communication mechanism for transfer- ring the sensor data and the actuator control information is challenging as shown in Table 1. Such a mechanism must be simple, pluggable and reliable in order to ensure, robust and reliable communications.

### 5.2. Objectives

Currently, the ability to gather data from different sensor devices and feed them into Cloud platforms in a unified manner is lacking. This hinders the availability of raw data on computational capable platforms that can quickly and efficiently analyze the data to derive knowledge, which could support nu-merous usage scenarios such as real-time critical applications in health and medicine. Therefore, the goals of the generic Cloud-based sensor monitoring and data processing platform are to provide standard and open mechanism for collecting diverse remote sensor data and processing them irrespective of the sensor devices or usage platform. Furthermore, we aim to achieve:

1) Easy setup and portability;

2) Platform-neutral data formatting and serialization;

3) Interoperability among heterogeneous Cloud platforms;

4) Simple configurations and ease of usage.

In the next section, we describe our proposed architecture.

## 6. Generic Cloud-based Sensor Monitoring and Data Gathering Platform

The details of our proposed generic Cloud-based sensor monitoring and data gathering platform are presented in this section. We discuss the conceptual architecture and its implementation considerations.

### 6.1. Conceptual Architecture

The conceptual architecture is designed to address the identified challenges posed by the existing platforms. Ease of use, portability, openness and interoperability are the key factors driving this design. This architecture is further motivated by what we think could be of value to customers with a system of distributed sensors and actuators.

Fig. 5 presents the proposed architecture. As shown on that figure, it is a distributed environment designed to accommodate efficient sensor monitoring, data gathering and sending of control information based on analyzed data to actuators. Different usage scenarios are considered in this conceptual architecture such as:

**Dual interactions:** In this case, a local or public server is attached to both sensors and actuators thereby making it capable of forwarding the captured sensor data and as well receiving the control information to be passed on to the actuators.

**Data gathering:** This scenario includes situations where the aim is to store and analyze the sensor captured data in the Cloud. This could also be to make the data public for applications to use or to use the analyzed data results to derive control decisions for actuators.

**Controller actuating:** This represents the cases where the stored sensor data are directly being used by applications or to control an actuator.

As shown in Fig. 5, the conceptual architecture consists of different components. However, we focus on the key ones due to space constraints.

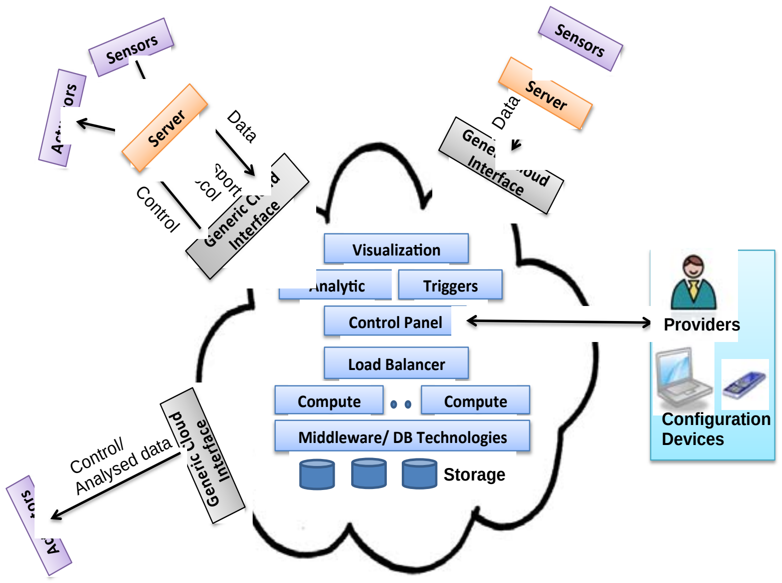


**Fig. 5.** Conceptual Architecture.

**Database Technologies**

The storage of the sensor data on a Cloud platform requires efficient management for inputting and querying the data. This process is supported by appropriate database technologies. In our design, we aim to make this component pluggable so as to support diverse DB types depending on the usage platform. This enables the use of both SQL and NoSQL databases. The relational SQL database types would provide easy compatibility since most of the existing applications, according to our review and experiences, are based on this technology. The NoSQL databases are aimed for quick and less complex scaling of the data storage system.

**Generic Cloud Interface**

The generic Cloud interface shown in Fig. 6 is responsible for mediating between the sensor servers and the Cloud platform. It provides a simple interface and supports numerous formats for time series and alphanumeric data coming from the sensor servers.

On the side of the Cloud platforms, it uses platform-neutral data interchange formats to achieve interoperability among heterogeneous Cloud environments.

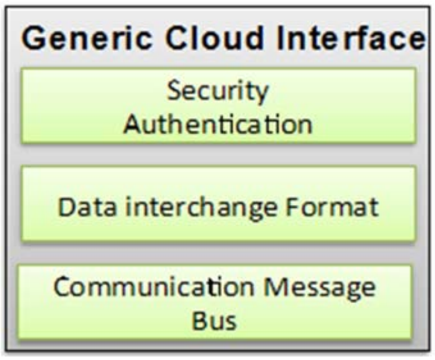


**Fig. 6.** Generic Cloud Interface.

As shown in Fig. 6 the generic Cloud interface includes three essential components –

1) Security;
2) Data interchange format;
3) Communication message bus.

These components provide the core functionalities of this interface and therefore deserve more explanations.

**Security**

This component ensures the privacy of the data interactions and guarantees their authenticity. It provides authentication mechanism to validate the access credentials of the sensor servers to access and transfer data to the Cloud platform. It also enforces data location constraints specified as a policy by the sensor data owners. This policy controls the location of the sensor data storage in the Cloud. It also informs the data owner whenever their data are being copied to some other location.

**Data Interchange Format**

The format of the sensor data impacts how they can be transported and analyzed. This component plays the role of formatting the sensor data into a platform-neutral data interchange format enabling portability, interoperability and efficient transportation. The data interchange formats can be categorized into two groups: (i) self-describing data interchange formats such as XML and JSON and (ii) binary (schema-based) data interchange formats like MessagePack & Protocol Buffers. Based on our previous findings [38], the two format groups have their advantages and drawbacks. The self-describing data interchange format group has the strength of being human readable and easy to understand. But, from the transmission perspective, they contain redundant components, which affect the size of data transfers. The binary data interchange format group is not human readable. But, they are more efficient for transmission according to the performance results in [38]. For this generic interface, we propose a hybrid data interchange format combining the strength of self-describing and binary data interchange formats.

**Communication Message Bus**

This component provides the communication mechanism for transferring the sensor data to the Cloud platform and also for sending control information to the actuators. It uses the data interchange formats for formatting and serializing the data. The message bus consist of three internal components:

1) Producer;
2) Messaging infrastructure;
3) Consumer.

The producer feeds in the data to be transmitted by connecting to the external data producing devices. It integrates the data interchange formats to appropriately prepare the data for transmission. The messaging infrastructure provides the functions of a message broker by asynchronously delivering messages from the producer to the consumers (synchronization decoupling). The producer does not need to know the nature or location of a consumer. It simply delivers its messages to the broker, which in turn routes them to the appropriate consumer (space decoupling). The broker therefore enables space, time and synchronization decoupling [39]. This feature facilitates the necessarily loose relationship between a producer and a consumer, which is essential in distributed systems like Clouds. To realize the messaging infrastructure, we base our implementation on the well-established Advanced Message Queuing Protocol (AMQP) [40]. The consumer connects the receiving end of the communication. It ensures that the data is appropriately reserialized for the end target platform.

## 6.2. Implementation Considerations

For a prototype implementation of this proposed architecture, we suppose the following considerations to be important.

**Software artifacts:** We envision the use of well established open source software components and standardized technologies as the basis for the implementation. As shown by our survey, there are some promising open source projects that could be considered. This would support the vision for an open and generic solution. Furthermore, it would avoid unnecessary re-implementations.

**Component interactions:** The generic interface is designed to interact with sensors, actuators and servers mediating between remote sensors and actuators in different fashions. In implementing these interactions, care has to be taken in choosing the data interchange format and communication mechanism since these two factors are very relevant in achieving a wide portability and interoperability of a solution.

**Openness:** Efforts should be made to make any implemented solution accessible to the general public. This would support quick adoptions and provide demos to the industry to encourage uptake among startup enterprises.

## 7. Conclusions and Future Work

This paper presented a survey of the current Cloud-based sensor monitoring and data gathering platforms. It analyzed their state-of-the-art and identified open challenges and trends. Both open source and commercial solutions were reviewed to provide a comprehensive analysis. We used a table to compile all the features of the analyzed platforms, which created a basis for quick comparisons and identification of trends. Furthermore, we presented a detailed description of some identified challenges regarding openness, interoperability, messaging protocol and energy efficiency.

According to our analysis, the commercial solutions are generally closed source and implement proprietary technologies, which hinders portability and interoperability among the platforms. Furthermore, the analyzed platforms provide weak support for messaging protocols, energy efficiency and resource-constrained environments. Also, the platforms use mostly textual data interchange formats, which are not very compact in size for efficient transportation. To address these identified challenges, we proposed a generic Cloud-based sensor monitoring and data gathering platform. The goal is to provide an open solution implementing standardized technologies to promote fast adoptions.

In our next steps, we intend to implement a prototype of our proposed conceptual architecture as a proof-of-concept. We aim to provide it to the general public as open source software to enable fast testing, deployment and adoption by the market. This contributes to our vision of facilitating interoperable data management in Clouds.

## Acknowledgements

This work is supported by the Irish Centre for Cloud Computing and Commerce (IC4), an Irish National Technology Centre funded by Enterprise Ireland and the Irish Industrial Development Authority.

## References

[1]. L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, *Computer Networks*, Vol. 54, No. 15, 2010, pp. 2787–2805.

[2]. B. Rao, P. Saluia, N. Sharma, A. Mittal, S. Sharma, Cloud computing for internet of things and sensing based applications, in *Proceedings of the 6th International Conference on Sensing Technology (ICST)*, Dec. 2012, pp. 374–380.

[3]. J.-C. Liu, K.-Y. Chuang, Y.-L. Wen, An efficient data gathering system for home medical treatment, in *Proceedings of the 6th International Conference on Genetic and Evolutionary Computing (ICGEC)*, Aug. 2012, pp. 460–463.

[4]. C. Rolim, F. Koch, C. Westphall, J. Werner, A. Fracalossi, G. Salvador, A cloud computing solution for patient's data collection in health care institutions, in *Proceedings of the 2nd International Conference on eHealth, Telemedicine, and Social Medicine (ETELEMED'10)*, Feb. 2010, pp. 95–99.

[5]. FreshTemp, Cloud-based temperature monitoring tool, https://freshtemp.com/[retrieved: 09-01-2015].

[6]. SensaTrack, Online sensor monitoring platform, http://www.sensatrack.com/index.html [retrieved: 09-01-2015].

[7]. V. C. Emeakaroha, K. Fatema, P. Healy, J. P. Morrison, Towards a generic cloud-based sensor data management platform: A survey and conceptual architecture, in *Proceedings of the 8th International Conference on Sensor Technologies and Applications (SENSORCOMM'2014)*, 2014, pp. 88-95.

[8]. D. Catrein, M. Henze, K. Wehrle, R. Hummen, A cloud design for user-controlled storage and processing of sensor data, in *Proceedings of the IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom), (CLOUDCOM'12)*, 2012, pp. 232–240.

[9]. Aoki H., *et al.*, Cloud architecture for tight interaction with the real world and deep sensor-data aggregation mechanism, in *Proceedings of the International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*, September 2010, pp. 280–284.

[10]. Piyare R., *et al.*, Integrating wireless sensor network into cloud services for real-time data collection, in *Proceedings of the International Conference on ICT Convergence (ICTC)*, Oct. 2013, pp. 752–756.

[11]. A. Alamri, W. S. Ansari, M. M. Hassan, M. S. Hossain, A. Alelaiwi, M. A. Hossain, A survey on sensor-cloud: Architecture, applications, and approaches, *International Journal of Distributed Sensor Networks*, 2013, Vol. 2013, 2013, Article ID 917923.

[12]. A. Salehi, K. Aberer, GSN, quick and simple sensor network deployment, in *Proceedings of the European Conference on Wireless Sensor Networks (EWSN)*, Netherlands, January 2007, http://lsir.epfl.ch /aberer/files/PAPERS/EWSN%202007.pdf

[13]. Open Geospatial Consortium, Sensor observation service, http://www.ogcnetwork.net/SOS_Intro [retrieved: 09-01-2015].

[14]. IBM, Ibm mote runner for wireless sensor platform, http://www.zurich.ibm.com/moterunner/ [retrieved: 09-01-2015].

[15]. Caracas A., *et al.*, Energy-efficiency through micro-managing communication and optimizing sleep, in *Proceedings of the 8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, June 2011, pp. 55–63.

[16]. A. Caracas, T. Kramp, M. Baentsch, M. Oestreicher, T. Eirich, I. Romanov, Mote runner: A multi-language virtual machine for small embedded devices, in *Proceedings of the 3rd International*

*Conference on Sensor Technologies and Applications (SENSORCOMM'09)*, June 2009, pp. 117–125.

[17]. MicroStrain, Microstrain sensorcloud platform, http://www.sensorcloud.com/[retrieved: 09-01-2015].

[18]. V. K. Sehgal, A. Patrick, L. Rajpoot, A comparative study of cyber physical cloud, cloud of sensors and internet of things: Their ideology, similarities and differences, in *Proceedings of the IEEE International Advance Computing Conference (IACC)*, 2014, pp. 708–716.

[19]. Ostia, Portus Platform, http://www.ostiasolutions.com /index.php/ product/platform-overview [retrieved: 09-01-2015].

[20]. TempoIQ, Sensor Analytics for the Measured World, https://www. tempoiq.com/ [retrieved: 09-01-2015].

[21]. Zhang Jia, *et al.*, Sensor data as a service–a federated platform for mobile data-centric service development and sharing, in *Proceedings of the IEEE International Conference on Services Computing (SCC)*, 2013, pp. 446–453.

[22]. Bluwird, BLUWIRED S-CLOUD Platform, http://bluwired.com/ Pages/Discover/BluwiredCloud.aspx [retrieved: 09-01-2015].

[23]. J. Hong, M. Baker, Interaction platforms, energy conservation, behavior change research, and more, *IEEE Pervasive Computing*, Vol. 12, No. 3, 2013, pp. 10–13.

[24]. Yang Kuo-Pao, *et al*., Park-a-lot: An automated parking management system, *Computer Science and Information Technology*, Vol. 1, No. 4, 2013, pp. 276–279.

[25]. C. Doukas, I. Maglogiannis, Bringing IoT and cloud computing towards pervasive healthcare, in *Proceedings of the IEEE 6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, 2012, pp. 922–926.

[26]. A. A. Chandra, Y. Lee, B. M. Kim, S. Y. Maeng, S. H. Park, S. R. Lee, Review on sensor cloud and its integration with arduino based sensor network, in *Proceedings of the IEEE International Conference on IT Convergence and Security (ICITCS)*, 2013, pp. 1–4.

[27]. ThingSpeak, ThingSpeak Platform, https://thingspeak.com/ [retrieved: 09-01-2015].

[28]. Microsoft, Microsoft azure intelligent system service, http://www.microsoft.com/windowsembedded/en-us/intelligent-systems-service.aspx [retrieved: 09-01-2015].

[29]. M. Prihodko, Energy consumption in location sharing protocols for android applications, Master Thesis, *Institute of Technology, Linkoeping University*, 2012,.

[30]. L. McNamara, B. A. Nahas, S. Duquennoy, J. Eriksson, T. Voigt, Demo Abstract:SicsthSense - Dispersing the Cloud, http://docs.sense.sics.se/wp-content/uploads/2014/09/SicsthSense-tutorial.pdf [retrieved: 06-01-2015].

[31]. Relay Platform, https://developer.relayr.io/documents/Welcome/Introduction [retrieved: 06-01-2015].

[32]. L. Diakite, L. Yu, Energy and bandwidth efficient wireless sensor communications for improving the energy efficiency of the air interface for wireless sensor networks, in *Proceedings of the International Conference on Information Science and Technology (ICIST)*, March 2013, pp. 1426–1429.

[33]. B. Han, D. Zhang, T. Yang, Energy consumption analysis and energy management strategy for sensor node, in *Proceedings of the International Conference on Information and Automation (ICIA'08)*, June 2008, pp. 211–214.

[34]. K. J. Kim, F. Cottone, S. Goyal, J. Punch, Energy scavenging for energy efficiency in networks and applications, *Bell Labs Technical Journal*, Vol. 15, No. 2, Sept. 2010, pp. 7–29.

[35]. A. Mammu, A. Sharma, U. Hernandez-Jayo, N. Sainz, A novel cluster-based energy efficient routing in wireless sensor networks, in *Proceedings of the IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, March 2013, pp. 41–47.

[36]. T. Wilde, A. Auweter, M. Patterson, H. Shoukourian, H. Huber, A. Bode, D. Labrenz, C. Cavazzoni, Dwpe, a new data center energy-efficiency metric bridging the gap between infrastructure and workload, in *Proceedings of the International Conference on High Performance Computing Simulation (HPCS)*, July 2014, pp. 893–901.

[37]. H. Hamann, T. van Kessel, M. Iyengar, J.-Y. Chung, W. Hirt, M. Schappert, A. Claassen, J. Cook, W. Min, Y. Amemiya, V. Lopez, J. Lacey, M. O'Boyle, Uncovering energy-efficiency opportunities in data centers, *IBM Journal of Research and Development*, Vol. 53, No. 3, May 2009, pp. 10:1–10:12.

[38]. V. C. Emeakaroha, P. Healy, K. Fatema, J. P. Morrison, Analysis of data interchange formats for interoperable and efficient data communication in clouds, in *Proceedings of the IEEE/ACM 6th International Conference on Utility and Cloud Computing (UCC'13)*, 2013, pp. 393–398.

[39]. N.-L. Tran, S. Skhiri, E. Zimanyi, EQS: An elastic and scalable message queue for the cloud, in *Proceedings of the IEEE 3rd International Conference on Cloud Computing Technology and Science (CloudCom)*, 2011, pp. 391–398.

[40]. S. Vinoski, Advanced message queuing protocol, *IEEE Internet Computing*, Vol. 10, No. 6, 2006, pp. 87–89.

___

2015 Copyright ©, International Frequency Sensor Association (IFSA) Publishing, S. L. All rights reserved.
(http://www.sensorsportal.com)