Metadata Management for the Web of Things: a Practical Perspective

Carolina Fortuna, Patricia Oniga[1],
Zoltan Padrah, Mihael Mohorcic
Department of Communication Systems,
Jozef Stefan Institute
Jamova 39, Ljubljana, Slovenia
+ 386 1 477 3528
carolina.fortuna@ijs.si,
patricia.oniga@yahoo.com,
zoltan.padrah@ijs.si, mihael.mohorcic@ijs.si

Alexandra Moraru
Artificial Intelligence Laboratory,
Jozef Stefan Institute
Jamova 39, Ljubljana, Slovenia
+ 386 1 477 3528
alexandra.moraru@ijs.si

ABSTRACT
Motivated by the importance of metadata for WoT systems, in this paper, we describe building a metadata management system which is scalable and rich in semantics. We describe two implementation approaches and discuss advantages and disadvantages of each: the embedded approach and the middleware approach. We also identify three components relevant to managing the metadata: the storage, the representation and the access. Based on our experience with implementation, we conclude that: (i) both the embedded and the middleware solutions can already be prototyped, but some critical technologies for the embedded approach are still in early development and require considerable improvements, (ii) XML like syntax is not well suited for storing and transmission of metadata due to sensor device constraints with respect to available storage and link datarate; and (iii) the middleware approach proved more convenient from the web application developer's point of view.

Categories and Subject Descriptors
H.3.4 **[Information Systems]**: Systems and Software: *distributed systems* H.3.5 **[Information Storage and Retrieval]**: On-line Information Services: *web-based services* D.2.12 **[Software]**: Interoperability: data mapping.

General Terms
Design, Experimentation.

Keywords
Web of Things, metadata, sensors, embedded, middleware.

WoT 2012, June 2012, Newcastle, UK.

1. INTRODUCTION
One of the essential components enabling the Web of Things (WoT) applications is the existence of metadata. Properly collected and annotated metadata permit search over sensors attached to or embedded in various things and their deployments, context inference as well as interoperability between systems. This problem has been recognized and researched in the literature and several collection, annotation and management systems have been proposed [1]-[4],[6] and [13]. While the existing body of work addresses aspects such as vocabulary and representation [10][6], collection [7], [3] and management [2], embedded [6][15] and middleware [2][14] approaches, to the best of our knowledge a comparative assessment based on implementations is still missing.

In this work we describe building a metadata management system which is scalable and rich in semantics based on our implementation experience. We describe two implementation approaches and discuss advantages and disadvantages of each: the embedded approach and the middleware approach. We also identify three components relevant to managing the metadata: the storage, the representation and the access.

The contribution of this work is that it compares the middleware based and embedded metadata management and describes reference implementations for each along with their advantages and disadvantages. Another contribution is our on device automatic metadata assembly mechanism.

The rest of this paper is structured as follows. Section 2 introduces alternative metadata management architectures: the embedded and the middleware approach. Sections 3, 4 and 5 present the implementation of the metadata storage, representation and access components for the embedded and middleware approaches and discuss their relative merits. Section 6 presents related work and Section 7 concludes the paper.

2. ARCHITECTURE ALTERNATIVES
The two architectures, embedded and middleware based are depicted in Figure 1. The embedded approach, on the left side of

[1] Currently a MSc student at the Technical University of Cluj-Napoca, Romania. The work was conducted during the summer internship at the Jozef Stefan Institute.

the picture presents things which are accessed directly from the web without any intermediates. In the middleware based approach depicted on the right side, applications communicate with a middleware that intermediates to the things. The middleware is responsible to collect metadata from the things, store it, process it when necessary and respond to clients' requests.

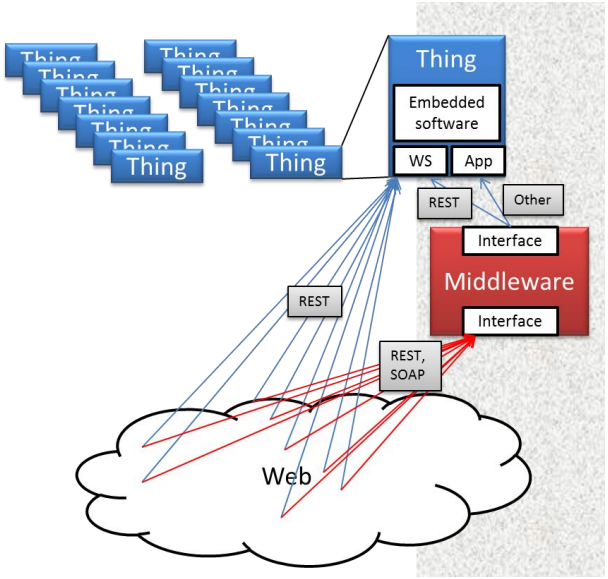

Figure 1 Schematic illustration of two WoT systems

2.1 The Embedded Approach

From the Internet community's point of view, the technological solution for creating the Internet of Things (IoT) is to assign an IP address to every device, therefore connecting the things directly to the current Internet. As a result, simplified implementations of the IPv6 protocol emerged, as well as the 6LoWPAN family of standards which are essentially used for the efficient utilization of IPv6 over constrained communication links [5]. Furthermore, to support the WoT and Machine-to-Machine (M2M) communication, embedded web services implemented using the emerging IETF Constrained Application Protocol (CoAP)[2] standard are being proposed in [6]. From the WoT application developer's point of view, the proposed technology is very friendly as it allows REST access to any device and it does not require writing custom code to communicate with the thing.

2.2 The Middleware Approach

From the web community's point of view, the distinction between the IoT/WoT and Internet/Web protocol stack architectures is not so relevant. Typically web application developers would expect similar quality of experience in accessing resources on the things as they get when accessing any traditional web resource. This involves using a gateway solution which would, on one side, interface with the applications and on the other side interface with a network of things. This is illustrated with red in Figure 1. From the implementation point of view, the requests from web applications for resources would employ the typical HTTP/TCP/IP protocol stack using RESTful and/or SOAP calls. Then, from the middleware to a particular thing, REST or proprietary protocols can be used.

2.3 The Information about the Things

Depending on the two approaches, the metadata can be stored, represented and accessed in several ways. The metadata can be stored in large repositories such as enterprise servers and databases which are part of the middleware or they can reside on the things themselves. From the representation point of view, the metadata can be represented via a custom approach which is system specific or a standardized approach which allows compatibility among systems. Among the standardized representations, several solutions exist; typically these depend on which community was involved in their development and how much emphasis was put to expressivity and interoperability. The metadata access needs to be discussed from two points of view: from the data consumer's point of view and from the things' point of view. While the data consumer may expect certain quality of experience from the device, the latter may not be able to provide it due to hardware constraints. While direct access to the device is what ultimately the WoT is all about, access through a middleware may be more pragmatic given the current state of the art.

3. THE METADATA STORAGE

The middleware storage solutions can choose between a wide variety of available technologies. The classic approach is the implementation using a relational database solution, a document oriented database solution or a knowledge base. In this implementation, the collection of metadata has to be addressed as well. Here we distinguish the manual approach of inserting the metadata through some kind of UI or an automated approach which can scale to large number of types of devices. In the case when the metadata resides on the things we may be dealing with a hypermedia database implemented by the World Wide Web of Things or as a distributed database where each thing holds a piece of data (i.e. TinyDB). With respect to linked data and semantic web technologies, the preferred solutions are provided under the form of triple stores or knowledge bases.

3.1 The Embedded Metadata Storage

In our implementation, the embedded metadata has been added to the embedded software (firmware) of the devices, in order to facilitate the automatic gathering of information about the things. This metadata is stored in the flash memory of the devices, just as the program code, and it is transmitted upon request. The metadata has been designed to be added in the C source files of the embedded device. This way, when the program code is changed, the metadata can be edited in the same file. In our implementation, metadata declaration can exist in multiple files.

When building the firmware image, all the metadata declarations are collected in a single continuous block inside the firmware. The order of these blocks is not predefined so they may appear in random order in the final message. In order to overcome the problem of not having a predefined order of declarations, a header and a footer declaration have been created. It is guaranteed that the header is the first, and the footer is the last part of the metadata. The metadata first contains the header, then the metadata as declared in the modules, and finally the footer.

[2] http://tools.ietf.org/html/draft-shelby-core-coap-req-04

For implementation, the linker script has been modified, a new C header and source files have been created and finally the metadata has been added. The implementation uses VESNA sensor node [8] devices running Contiki 2.5 OS[3].

3.2 The Middleware Metadata Storage

In our middleware implementation we have used two types of storage systems: relational database and knowledge bases. The messages containing the metadata are processed by the server and then sent to the storage systems. For the relational database we used the MySQL database server. The database schema has been designed so as to reflect the hardware configuration and has a direct mapping to the information collected from the things using a custom format and custom developed protocol, Discovery and Identification Protocol (DIP). DIP is an application layer protocol which consists of three separate parts: node discovery, data collection and node identification. When a node is discovered, but not identified (no corresponding metadata available on the server side), the server sends request for identification of that node. For the implementation of DIP we used VESNA sensor node platform with Contiki 2.5 OS. The metadata can be delivered encoded in custom or JSON format. The server implements a custom parser for the first case.

For the knowledge bases we used two systems Sesame[4] triple store and ResearchCyc KB[5]. The triple store is populated with rich metadata retrieved from the things using REST calls and JSON for Linking Data (JSON-LD) representation. The Research CycKB is currently filled in using a custom mapper between the MySQL tables and the ResearchCyc Ontology.

3.3 Discussion

In the case of the embedded approach, the size of the metadata string (i.e. JSON-LD in our implementation) is limited by the size of the device's flash memory which is opposed to the data consumer's requirements for high expressivity.

In the case of VESNA platform, which is powered by a stronger than average microcontroller with relatively larger flash memory, the full JSON-LD message occupies 15KB out of the 512KB of flash memory (~3%). For comparison, the CoAP application code takes about 25 KB and the full system image of the operating system with networking stack and application takes 216KB without optimization and roughly 150KB with compiler optimizations. Therefore the metadata uses 10% of the entire system image; this is arguably expensive for the current state of the art in embedded devices. An alternative implementation can store the metadata as string on dedicated external storage which are typically larger (i.e. SD cards), but require hand editing of the final JSON-LD string as in this case it cannot be automatically assembled.

In the middleware approach, based on a more mature field, the database type of storage systems are more accessible for application developers and also have well-defined methods for handling large amounts of structured data. However, a triple store gives more flexibility in terms of redesigning the structure of the data stored, or adding new features.

For instance, let us consider the following simplified example: we have a sensor node that has two types of sensors attached to it, measuring temperature and pressure in degree Celsius and millibars respectively. A database schema could be formed by three tables: the first one (SN) for sensor nodes identification and their MAC addresses, the second one (ST) for the sensor types, containing the observed phenomenon and the unit of measurement and the third one (SA) for specifying the sensors attached to a sensor node (illustration in Figure 2 Database Schema for Metadata StorageFigure 2).

SN	
ID	MAC
1	403AB8F9

ST		
ID	phenomenon	uom
1	temperature	degC
2	pressure	mbar

SA	
sn	st
1	1
1	2

Figure 2 Database Schema for Metadata Storage

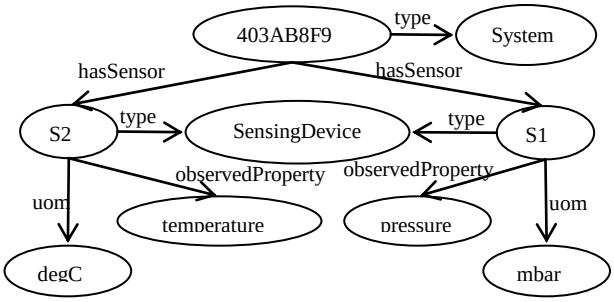

Figure 3 Metadata Triples from the Store

The same metadata is represented in Figure 3 in a graph-based model of triples of the form subject-predicate-object. The arrows represent the predicates and objects are the ones to which the arrows are pointing to.

If additional metadata is needed at a later time it would require the extension or even redesign of the database, which would affect the other applications using these data. For example if we need to specify also the radio board attached to the sensor node, it would probably require a new table in the database, while in the triple based representation we simply add these data and we can just specify the predicate and object types later in an ontology.

A combination of the two seems however to work best [9]. A relational database can be used to store the basic information, providing fast and scalable access to essential sensor metadata, while the triple store can be used for richer descriptions, offering more flexibility.

4. THE METADATA REPRESENTATION

For satisfying application requirements the metadata needs to be represented in a structured and standardized manner which also allows interoperability. Solutions with high expressivity and good interoperability characteristics are typically based on XML syntax, domain relevant examples including: SensorML, ZigBee SmartEnergy and RDF. Recently, JSON is becoming increasingly popular as low overhead alternative to XML. Solutions typically use a publicly shared schema, vocabulary or ontology to annotate data. For efficient storage or transmission of metadata, some

[3] http://www.contiki-os.org/

[4] http://www.openrdf.org/

[5] http://research.cyc.com/

implementations may use encoding techniques which perform compression. For XML syntax Efficient XML Interchange (EXI), Binary XML (BXML) and Fast Infoset have been considered in the literature [6] while for JSON the emerging approach is Binary JSON (BSON).

4.1 Embedded Metadata Representation

We implemented two versions of the metadata representation: a custom, non-standard representation and a standard JSON-LD representation as can be seen in Figure 4. The description is for the same example introduced in Section 3.

```
p01=403AB8F9&p02=VSCv1.2.1&p03=temperature
&p04=degC&p05=pressure&p06=mbar                    (a)
```

```
"@context":{
"base":" http://hostname/semsense/resource/",
"uri": "@subject",
"isa" : "@type",
"sensorNode" : "http://purl.oclc.org/NET/ssnx/ssn#System",
"modules" : "http://purl.oclc.org/NET/ssnx/ssn#hasSubSystem",
     ...
}
"uri":"base:403AB8FC",
      "isa": "sensorNode",
      "modules" : [ {
           "uri":"base:403AB8FC/s1",
                "isa": "sensor",
                "observedProperty" : {
                          "uri" : "temperature",
                          "uom" : "degC",
                          }
                 },
                {
           "uri":"base:403AB8FC/s2",
                "isa": "sensor",
                "observedProperty" : {
                          "uri" : "pressure",
                          "uom" : "mbar",
                          }
                 }]                                (b)
```

Figure 4 Proprietary metadata message (a) vs JSON-LD format (b)

The semantic vocabulary used in our implementation comprises several ontologies. On the upper layer we use the DOLCE Ultra Lite Ontology, to which the Semantic Sensor Network (SSN) ontology [10] is aligned. The majority of the concepts used are part of the SSN ontology, which is a sensor network specific ontology. Other external resources used are the Basic GeoWGS84 vocabulary, which provides the namespace for representing the coordinates, and GeoNames database in RDF format for the names of geographical location. Finally, for the domain specific concepts we have used the Measurement Units Ontology (MUO) and an extension of the SSN ontology that we created consisting mainly of subclasses of the SensingDevice class and individuals of the FeatureOfInterest class.

For comparison, the custom message is less human readable than the equivalent JSON, it follows an implicit convention and sequence of parameters and necessitates a custom parser and several transformation steps on the middleware side to transform it into an interoperable format.

4.2 Middleware Metadata Representation

Our implementation for the middleware metadata storage uses two approaches: relational database and triple store for the two types of formats described in the embedded approach. Figure 5 illustrates the workflows for two approaches, one based on the relational database and the second using the triple store. The workflow presents the main steps for publication of sensor metadata as linked data on the web, starting from the sensor metadata collected at the server. The process of creating semantic descriptions of sensor metadata can be applied in different steps, depending on the storage solution adopted and format of the metadata. For the relational database case, first a parser for the custom metadata format illustrated in Figure 4 (a) is used to fill the database tables. The database schema used is similar to the one introduced in the example from Section 3.3, just having more fields. Next, a wrapper that translates the database content to RDF is used. The wrapper provides a RDF representation of the metadata based on declarative mappings between the database schema and the semantic vocabulary. The D2R Server can be used to manage the URIs created for the RDF representation of the database content [11].

For the triple storage solution, we used the JSON-LD metadata format in which the semantics of the description are explicitly stated by the ontology concepts defined in the section tagged with the "@context" keyword. The full message is omitted due to space constraints. A preparation step is required for translation from JSON-LD to the XML/RDF syntax supported by the triplestore.

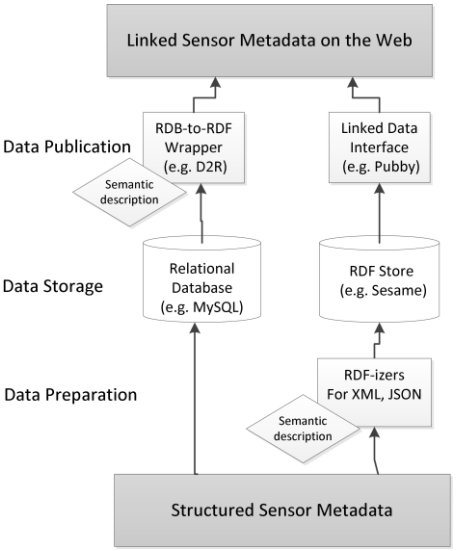

Figure 5 Workflow of semantic enrichment and publication of sensor metadata (adapted from Figure 5.1 from [12])

4.3 Discussion

In the embedded approach, a structured representation of the data has the disadvantage of occupying a notable proportion of the device's memory resources, however it supports immediate integration into WoT. For comparison, the custom format equivalent to the full 15KB JSON-LD message takes only 1KB. Recently, research activities are looking into more efficient

representation of metadata and data for constrained devices and if followed by appropriate standardization efforts this may lead to a widely accepted solution [6].

On the middleware side, the transformation of metadata can be straightforward if properly represented in a structured manner. If in custom format, however it needs to be transformed in a structured form and this requires custom parsers and adapters. These may work well for a small system, but are likely becoming and integration nightmare on longer term. The middleware approach also permits users to send SPARQL queries and receive the corresponding answer, while this is currently not supported in the embedded approach (but some efforts in the this direction are underway). In [13], a SPARQL-enabled applications for mobile devices are described, able to expose via an SPARQL-endpoint the data created by sensors or other applications running on an Android device.

5. THE METADATA ACCESS

In our implementation, of the embedded approach, the access to the metadata is based on a RESTful API to the resource which returns a structured JSON-LD message. On the middleware side we provide full access according to linked data principles and support SPARQL clients.

5.1 Embedded Metadata Access

We made several resources available for access on the thing: one of them being the metadata which provides a JSON-LD message part of which is presented in Figure 4 Proprietary metadata message (a) vs JSON-LD format (b)

. As an interchange language for linked data, JSON-LD can be retrieved from the node by calling the resource at coap://[2001:470:55:0:212:4b00:6:a03c]:5683/metadata. The application which generated the call can convert the message using a JSON-LD processor into the desired triple format using the information from @context.

The implementation on the sensor node imposes constraints as discussed in the storage section, hence the metadata access on the node is done through the resource and not by querying the sensor node URI. We chose to implement this resource in such way as to return the message in JSON-LD format, which is more compact compared to RDF/XML and HTML representations. The URIs from the description provided are then made dereferenceable in the middleware approach for exposing the metadata as linked data.

5.2 Middleware Metadata Access

For providing access to the metadata in the middleware approach we have followed the linked data principles. For the publication of the metadata from the relational database, the D2R Server used for adding semantic descriptions provides also a SPARQL end-point for querying and an interface for RDF and HTML browsers.

Regarding the triple store, which provides only a SPARQL end-point, an additional tool for creating a linked data interface was used. In our implementation, the URIs from the RDF store are dereferenced by applying a linked data frontend for SPARQL end-points, implemented by the Pubby web application. Therefore a sensor node identified by the URI listed in (1) will respond with an HTML or XML/RDF message (URIs listed in (2) and (3)), depending on the type of requested content.

(1) http://hostname/semsense/resource/403AB8FC

(2) http://hostname/semsense/page/resource/403AB8FC

(3) http://hostname/semsense/data/resource/403AB8FC

5.3 Discussion

The fact that the constrained communication links typically have high packet loss rate (5-10%) and low datarate (10s kb/s) [6] will result in lack of or slow response from the thing. This may be inconvenient for the web application developer, especially from the semantic web and linked data communities which typically work with large XML and JSON messages. In our experience, in order to retrieve the JSON-LD message containing the metadata of the device, approximately 15 KB had to be transferred as application data. The technologies of the WoT stack impose fragmentation of such large messages. Since the CoAP specification does not recommend any fragmentation, we are using block-wise transfer, by sending the metadata in chuncks of 64 B each. Thus it takes over 200 such chunks to transfer the entire message which for a 100 kb/s link results in roughly 12 seconds transfer time. Finally, it may be that some of these chunks will be lost; therefore a new request for the resource may need to be triggered.

Another inconvenience related to constrained devices is the inability to process multiple requests at the same time. If it takes seconds to transfer a message from a thing, it is unable during this period to handle alternative calls for resources.

We found that the embedded approach to supporting the WoT can be implemented already with current devices and technologies but for the time being, and probably in the near future as well, it will be difficult to scale with respect to numbers of users/consumers of data. Eventually some kind of overlay network which would perform caching on less constrained devices will need to be deployed.

Currently, the middleware approach appears more practical as it gives the possibility of handling more requests and larger amounts of data, which otherwise is constrained by the resources available on the embedded device. This provides the quality of experience expected by the web application developer, bypassing the limitations imposed by the device. As a tradeoff, implementing the middleware approach may be more complex as it requires more components and more design decisions.

6. RELATED WORK

In the following we briefly summarize some related research from other groups and compare it to our work.

In [2], the authors propose a framework for metadata management in federated sensor networks. Centralized repositories for metadata interact with highly distributed data stream processors allowing each user to autonomously obtain metadata from streaming sensor data, but human users can also insert it. Such metadata is then sent, stored, and managed at a centralized metadata repository, and can also be transmitted back in real-time to the distributed sensor networks and joined with sensor data streams for online processing and visualization.

In [14], the SensorMap system includes the GeoDB component which stores user generated sensor metadata. The latter allows users to search and is also used in the GUI as context information.

In [6], the authors introduce the Constrained Application Protocol (CoAP) and discuss technology for implementing embedded web services. Mechanisms and standards for encoding and annotation of resources are also discussed. In [15], the author introduces architecture for integrating things in the web and advocates the embedded approach.

Our work is different from the above as it presents a comparison between the two architectural approaches (i.e. embedded vs middleware) based on the actual experience we gained by implementing the two systems. We discuss advantages and disadvantages of the approaches from the application developer's perspective as well as from the perspective of constrained device embedded in things.

7. CONCLUSIONS

In this work we described our experience in building a metadata management system which is scalable and rich in semantics. We described the implementation of the embedded and the middleware approach and discussed advantages and disadvantages of each. As a result, we reached the following conclusions:

- Both the embedded and the middleware solutions can be prototyped, however for the embedded solution critical technologies such as application level protocols and data access mechanisms are still in early development.

- The current state of the art in sensor devices limits the amount of stored data, therefore XML like syntax is not well suited for storing. Similar conclusion can be drawn for accessing the data, where the constraints are imposed by the communication links.

- The middleware approach is more convenient from the perspective of web application developer; however technological developments empowering the embedded approach may be catching up soon.

8. ACKNOWLEDGEMENTS

We would like to thank all our colleagues in SensorLab who contributed indirectly to this work. This work was supported by the Slovenian Research Agency through the programme P2-0016 and project J2-4197, the competence center KC OPCOMM, and the FP7 ICT projects, ENVISION (ICT-2009-249120) and PlanetData (ICT-NoE-257641).

9. REFERENCES

[1] Guinard, D., Trifa, V., Karnouskos, S., Spiess, P., & Savio, D. (2010, July). Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services. IEEE Transactions on Services Computing, 3(3), 223-235.

[2] Jeung, H. et al, Effective Metadata Management in Federated Sensor Networks, In Proc. of the Third IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC2010), Newport Beach, California, USA, June 7-9, 2010.

[3] De Ipina, D. L., Vazquez, I., Abaitua, J. (2007). A Context-Aware Mobile Mashup for Ubiquitous Web. In Proceedings of IET International Conference on Intelligent Environments (University of Ulm, Ulm, Germany, Sept. 24-25, 2007). IE2007.

[4] Le-Phuoc, D., Hauswirth, M. (2009). Linked open data in sensor data mashups. In Proceedings of the 2nd International Workshop on Semantic Sensor Networks (Washington DC, USA, Oct. 25-29, 2009). SSN'09.

[5] Vasseur, J.-P., & Dunkels, A. (2010). Interconnecting Smart Objects with IP: The Next Internet, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. ISBN:0123751659 9780123751652.

[6] Shelby, Z. (2010, December). Embedded Web Services. IEEE Wireless Communications, 17(6), 52-57.

[7] Jammes, F., Mensch, A. Smit, H., Service-oriented device communications using the devices profile for web services, Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing, 2005, New York, NY, USA. DOI: 10.1145/1101480.1101496.

[8] Smolnikar, M., Fortuna, C., Vučnik, M., Mihelin, M., Mohorčič, M. (2011). Wireless sensor network testbed on public lighting infrastructure. In EcoSense 2011 The Second International Workshop on Sensing Technologies in Agriculture, Forestry and Environment, 2011, April 6-7, Belgrade, Serbia.

[9] Aasman, J. (2011, March). NoSQL Option: Triplestore Databases. Retrieved 2011, September 30, from http://www.dbta.com/Articles/Editorial/Trends-and-Applications/NoSQL-Option-Triplestore-Databases-74251.aspx.

[10] Barnaghi, P., & Presser, M. (2010). Publishing Linked Sensor Data. In Proceedings of the 3rd International Workshop on Semantic Sensor Networks , at 9th International Semantic Web Conference, 2010, November 7-11, Shanghai, China.

[11] Moraru, A., Mladenic, D., Vucnik, M., Porcius, M., Fortuna, C. & Mohorcic, M. (2011, March). Exposing real world information for the web of things. In Proceedings of the 8th International Workshop on Information Integration on the Web: in conjunction with WWW 2011, Hyderabad, India.

[12] Heath, T. & Bizer, C. (2011). Linked Data: Evolving the Web into a Global Data Space (1st edition). Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool.

[13] d'Aquin, M., Nikolov, A., & Motta, E. (2011). Building SPARQL-Enabled Applications with Android Devices. (To appear) Demo at 10th International Semantic Web Conference (ISWC 2011), Bonn, Germany.

[14] Nath, S., Liu, J., Zhao, F. 2007. SensorMap for Wide-Area Sensor Webs, IEEE Computer, Vol. 40, Issue 7. July 2007. DOI=http://doi.acm.org/10.1109/MC.2007.250

[15] Guinard, D. 2011. A Web of Things Application Architecture – Integrating the Real-World into the Web. PhD thesis No. 19891, ETH Zurich, Zurich, Switzerland.