# Generic sensor network gateway architecture for plug and play data management in smart laboratory environments

Elmar Zeeb, Ralf Behnke, Christian Hess,
Dirk Timmermann
Institute of Applied Microelectronics
and Computer Engineering
University of Rostock
18057 Rostock, Germany
{elmar.zeeb, ralf.behnke, christian.hess,
dirk.timmermann}@uni-rostock.de

Frank Golatowski, Kerstin Thurow
Center for Life Science and Automation
18119 Rostock, Germany
{frank.golatowski, kerstin.thurow}@celisca.de

## Abstract

*During the last years, recent technological advances enabled the development of tiny devices equipped with radio, micro controller and sensors, called sensor nodes. Collaborating networks of such devices, known as Wireless Sensor Networks (WSNs), are subject of many researches. A couple of network centric tasks like data transmission and aggregation, self-organisation, localization and energy awareness have been focused in various publications. Recent researches, e.g. IPv6 over Low power WPAN (6LoWPAN), are going to standardize the connection of sensor nodes with common internet protocols. Nevertheless, until now proprietary techniques are used within most networks, especially for the coupling towards the outside world as well as the data management. Depending on WSN hardware and software, the gateway software as well as storage and presentation of data differs for most sensor networks. In this work we present an infrastructure which bases upon a generic gateway. Standardized dataset will be stored by this device using Sensor Web Enablement (SWE) and standard protocols. By use of Devices Profile for Web Services (DPWS) plug and play abilities as well as a comfortable query interface for presentation is realized.*

## 1 Introduction

Recent technological advances enabled the development of tiny devices equipped with radio, micro controller and sensors, called sensor nodes. Collaborating networks of such devices, known as WSNs, are still subject of many researches. Although the vision of sugar cube sized, long living sensor nodes is still a vision, a large amount of WSNs is in use in many different projects. A couple of network centric tasks like data transmission and aggregation, self-organization, localization and energy awareness have been focused in various publications. Recent researches, namely 6LoWPAN [6], are going to standardize the connection of sensor nodes with common internet protocols. Nevertheless, until now proprietary techniques are used within most networks. This includes also integration into IT systems where sensor data is used. Depending on the applications domain, those systems can be very complex and consist of dozens of components. WSN hardware, WSN software, gateway software as well as storage and presentation of data differs for most sensor network technologies. This leads to systems that are hard to extend or change. For example, for integration of a new WSN technology in an existing network you often have to modify gateway software and other components like database. When changing the whole WSN, e.g. hardware platform or node software, also a large part of data consuming components have to be modified. One of the most important drawbacks of data handling in existing WSNs is the lack of coexistence. Most WSN concepts assume a single large network connected to data consuming software. But in many applications it would be very useful to connect disjunct networks within a single data representation. This may be needed when spatial separated areas have to be monitored or different WSN hardware is used due to special demands. This can be hardly achieved in existing WSN approaches.

In Live Science Automation (LSA), which is one of our main applications of WSN, as described in [9], connecting multiple networks to a large virtual network is as much a demand as the integration of various sensors for measuring for phenomena like temperature, gas concentrations, light intensity, vibration and so on. For further use, gathered data has to be stored along with its time and place of origin, which can gathered from a global positioning system or a WSN specific localization like AWCL [10]. The process of data shipping and data management has been sparsely investigated in the past. In this paper we describe an architectural approach that offers the flexibility

to cope with the increasing complexity of WSNs and IT infrastructure used in LSA.

The following demands are extracted from our experiences with WSNs in LSA systems:

*Software and Hardware Independence:* Collected data should be accessible for data consuming components independent of the used WSN hardware, software and data coding.

*Multi Network Support:* Multiple WSNs should be coupled to the data management and should be merged.

*Multi Data Support:* The data management should be able to handle any kind of sensor readings and store it beside with meta data like date and position.

*Multi Client Support:* In contrast to existing approaches without data management, which often only allows to connect one data consumer directly to the network, multiple data consumers should be enabled to receive measurement data from our data management solution.

Most of this demands match features of Service-Oriented Architectures (SOAs). With the ongoing spread of common Internet protocols in many application domains the appliance of SOAs becomes obvious. Of course it is not always clear how to apply SOA concepts to new application domains. Systems in the LSA domain are on the one hand device centric when it comes to sensors and actors and on the other hand enterprise centric when it comes to data management and composition of components. Here it is difficult to integrate device infrastructures into SOAs in a way that the devices are part of the architecture and not only integrated. In the Web Services for Devices (WS4D) initiative [8] we work exactly on this issue and offer standardized technologies for device centric SOAs. In this paper we introduce a SOA based System that integrates WSNs in a flexible way into a LSA system to show how these technologies can be converged.

In the following a brief overview of technologies and standards in the area of sensor networks, data management, data acquisition, SOA and device centric SOAs is given in Section 2. Then in Section 3 the composition of technologies used for our approach is described. The SOA based architecture is shown in Section 4, followed by the implementation, discussed in Section 5. The paper closes with a conclusion in Section 6.

## 2 State of the art

The integration of WSNs covers several research topics such as WSNs itself, data management in sensor networks, data acquisition of sensor data and device centric SOAs that will be described in this section.

### 2.1 Wireless Sensor Network Technologies

WSNs became feasible due to the ongoing miniaturization of electronic components. Main components of a sensor node are a battery as the most often used energy source, a micro controller with attached memory to process data and control the nodes activity as well as the communication, a radio for communication and at least one sensor. This basic device architecture is illustrated in figure 1. One of the most important constraints and also the driving force for most researches on WSNs are limited resources in computation and energy. To meet energy constraints, various low power communication technologies have been established. While early implementations of WSNs used proprietary 868 MHz radio chips, newer devices are using IEEE 802.15.4 communication standard at 2.4 GHz. Many platforms are using ZigBee [3] or 6LoWPAN [6] on top of IEEE 802.15.4. Beside this process of standardization many protocols for Medium Access Control (MAC), specialized for WSNs have been developed, e.g. S-MAC [25], X-MAC [13] and Z-MAC [20].

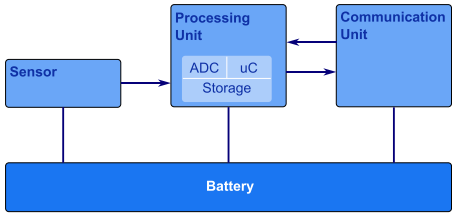


**Figure 1. Basic architecture of a non actuating wireless sensor node**

### 2.2 Data Management in Sensor Networks and Applications

Although we mentioned that the task of data management for WSNs has been sparsely investigated in the past, there are existing approaches. In this section we briefly review the most important players in this field and point out the drawbacks.

One of the best known middleware for data aquisition in WSNs is TinyDB [18]. It offers the possibility to inject SQL like queries directly into the sensor network. This makes the network act like a virtual database. It is possible to formulate queries selecting any conditions in measurements. The formulated query can be performed once or in defined intervals. The TinyDB middleware resides on the network gateway as well as on the nodes itself. The gateway hosts a query interpreter which transforms the query into instructions which can be processed in the network. The nodes are hosting a program which accepts these instructions and reply an appropriate sensor reading. The node part of TinyDB needs TinyOS [17] to work. Therefore TinyDB is not independent from the sensor network, because only WSNs, running TinyOS are addressed by TinyDB. In addition TinyDB covers only the network side. It provides only snapshot like queries. A history of past readings is not supported. In the same way only readings of active nodes can be requested.

A similar and also well known approach is Cougar [24]. Cougar consists of a node specific program, which performs a clustering process and data aggregation within the cluster, and a gateway program like TinyDB. An additional component of cougar is a graphical user interface (GUI) which allows to formulate queries.

In contrast to our approach TinyDB as well as Cougar are mainly located at the network side and the gateway. Both concepts strongly depend on the used sensor network hardware as well as the software. Neither handling of multiple WSNs nor provision of historic data is addressed. Therefore these concepts do not cover functionality of our work but can be an integral part, to retrieve data out of the network. Since these concepts of data management result from WSN research community, they have been designed with energy efficiency in mind. As already mentioned our work prescinds form the network and focuses on interoperability and thus data acquisition technologies tries to solve the interoperability problems when acquiring data form sensors.

### 2.3 Data Acquisition of Sensor Data

To evaluate data management approaches figure 2 shows an extended illustration of a typical WSN architecture. On the right sensor nodes are illustrated with at least one sink node which collects network readings to hand it to the gateway. The gateways main responsibility is to bridge the communication between WSN and the outside world, e.g. intranet. On the other hand any kind of client units can be connected to the gateway including monitoring applications as well as user clients or data storage. For better understanding we divide the given architecture into network side, gateway and client side.

#### 2.3.1 Sensor Web Enablement

The Open Geospatial Consortium (OGC), which drives the development of standards for geospatial content and

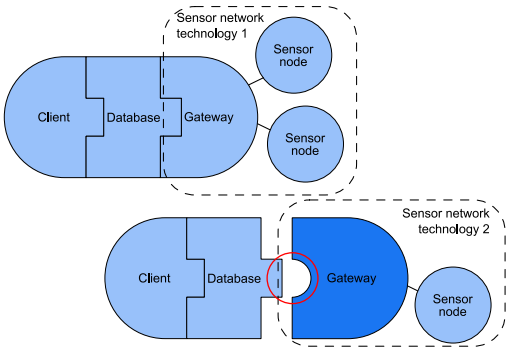


**Figure 2. Typical architecture for applications that use sensor data and the problems with this architecture**

services, defined SWE as a framework of open standards. SWE defines a standard framework for Sensor Webs. A Sensor Web refers to web accessible sensor networks and archived sensor data that can be discovered and accessed using standard protocols and application program interfaces (APIs) [12]. As further described in [12], SWE supports discovery of sensor systems, observations and observation processes, determination of a sensor's capabilities and quality of measurements, access to sensor parameters, retrieval of real-time or time-series observations and coverages in standard encodings, tasking of sensors and subscription to and publishing of alerts to be issued by sensors. These features are defined with several models, encodings and services for Sensor Webs as shown in figure 3.

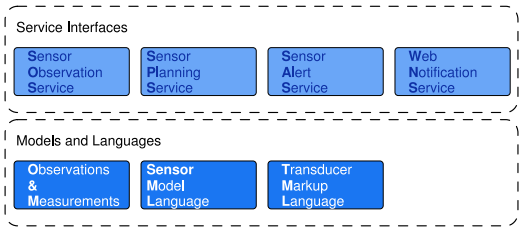


**Figure 3. SWE standards framework**

The Observation & Measurement (O&M) specification defines models and XML Schemas to encode observations and measurements of sensors. The specification is targeted to support a wide range of sensor specific data formats. It provides formats to represent observations, measurements, procedures and metadata of sensor systems.

Sensor Model Language (SensorML) provides standardized models and XML Schemas to describe sensors and processes. This description can be used for discovery of sensors, processing of low-level sensor observations, and get the taskable properties of sensors.

Transducers can be considered as the interface between the physical and the digital realm of sensors and actuators. With Transducer Markup Language (TML) it is possible to describe transducers. TML defines models to describe hardware response characteristics of a transducer and provides methods for transporting sensor data and preparing it for fusion through spatial and temporal association.

SWE defines the Sensor Observation Service (SOS) as a service interface to provide standardized access to observations, measurements and sensor description data. The data formats used herein are consistent for all different kind of sensor systems. The SOS interface is based on OGC web services and provides web based access to sensor data. The service interface defines operations to register new sensors, insert new observations or retrieve archived observations. A big advantage is that the sensor data is stored in a consistent and standardized format with the help of O&M and SensorML.

These standards already cover the typical data management scenario described before but offers additional flex-

ibility and a clean service oriented approach. To cover more complex scenarios SWE defines further service interfaces. Sensor Planing Service (SPS) provides methods to define tasks for sensor networks. Sensor Alert Service (SAS) provides publishing and subscribing to alerts from sensors. Web Notification Service (WNS) has a service interface for asynchronous delivery of messages or alerts from SAS, SPS or other endpoints. All this service interfaces use data formats that are consistent for different kinds of sensor systems.

#### 2.3.2 IEEE 1451

As shown in [21] the IEEE 1451 family of standards is another technology that can be used to integrate sensor data with standardized formats into applications. IEEE concentrats on communication in between sensors, actors and processors. It doesn't specify data storeage or publish subscribe mechanisms. In [21] the IEEE 1451 standards are combined with a SOAP web service called Smart Transducer Web Services (STWSs). This service provides the operations TimDiscovery, TransducerDiscovery, ReadTransducerData, and ReadTimMetaIDTeds to aquire the data of sensors in a consistent way for different sensor systems. So IEEE 1451 could also be used to integrate sensor data into applications but at a very low abstraction level.

### 2.4 SOAs and Device Communication

The increasing complexity of device networks consisting of up to thousands of devices demands new technologies for simple device interaction and interoperability. SOAs [14] firstly addressed this issue for software components. The probably most popular implementation of SOA are W3C Web services. For device to device communication, especially for resource-constraint devices, the Web services protocols often need too much resources and computing power. The Web services technology also lacks features like ad-hoc device discovery, device description and eventing channels needed in device networks. Thus, DPWS [19] was defined. It uses some specific Web services protocols and restricts their usage because of resource limitations in embedded systems. So DPWS enables the usage of Web services based technologies to implement device centric SOAs and thus offers the same modular and clearly defined software architectures in device networks ([11], [16]).

DPWS bases on well known protocols and Web service specifications. It uses similar messaging mechanisms as the Web Services Architecture (WSA) [23] with restrictions to complexity and message size ([19], [26]). On top of the low level communication foundations it uses Extensible Markup Language (XML), SOAP and XML-Schema for actual information exchanges. DPWS specifies mechanisms for ad-hoc device discovery based on WS-Discovery, device and service description based on WS-MetadataExchange and WS-Transfer and a publish-subscribe mechanism using WS-Eventing.

As a partner of the WS4D initiative, the University of Rostock has developed the WS4D-gSOAP toolkit, which includes a DPWS protocol stack implementation and some software tools to create devices and clients. Other toolkits for different platforms and programming languages are available from WS4D [8], Service-Oriented Architecture for Devices (SOA4D) [7] or as integral part of Windows Vista and the .Net Framework [4].

In difference to DPWS, SWE bases on OGC web services. These web services don't use the SOAP standard to encode messages on top of Hypertext Transfer Protocol (HTTP) but uses XML messages in combination with HTTP/Get and HTTP/Post methods.

## 3 Combination of DPWS with SWE

In the last section we described several technologies that can be part of an overall service oriented architecture for applications based on sensor data. We see the best opportunities in the combination of DPWS and SWE and use the synergies of both technologies.

The DPWS specification defines a base technology for device communication that can be easily composed with and extended by other specifications and technologies. DPWS has an architectural concept that is different to WSA to fit better into device scenarios. The main difference is the multicast service discovery with WS-Discovery that requires no central service registry such as UDDI that is used in WSA. But the service usage of services on devices is the same as the service usage in WSA whereby DPWS devices can be directly integrated WSA based enterprise systems.

SOAs are often used to improve flexibility and reusability of components in complex distributed applications. This is done by modelling functional blocks as independent services. But often there are still dependencies in the used data formats as it is more complicated to define generic data formats than generic service interfaces. Applications that make use of sensor data can easily be divided in functional independent building blocks but it is hard to define a common data format for different sensor systems to not depend on a specific sensor network technology. SWE solves this problem by offering data format and meta data formats to represent all information about observations, sensors, etc. in a way that is independent of specific sensor network technologies.

So one of the main opportunities for the combination of DPWS with SWE is a reduced administration overhead through real plug and play on the sensor application level and at the device communication level. Furthermore this combination reduces the sensor technology dependent part of a SOA to a minimal extend and eases the integration of new sensor network technologies into applications based on this architecture.

Of course there are several challenges when combining these technologies. As SWE uses a web service ap-

proach different from DPWS, which uses SOAP, there is the requirement for a proxy component. As the data formats defined in SWE are quite complex, it is challenging to implement these formats on small embedded systems that are often used as sensor network gateways and sensor data acquisition systems.

In the next sections we will describe first the resulting architecture and then the implementation of a generic sensor gateway for SWE based applications.

# 4 Generic Gateway Architecture

This section describes the generic gateway architecture resulting of the combination of SWE with DPWS. Main components are the SOS, the SWE-WS-Proxy which bridges from OGC to W3C web services and vice versa, and the generic gateway itself. The overall architecture including the mentioned components is shown in figure 4. In particular the SWE-WS-Proxy component enables to use the SOS service via a SOAP web service based interface.

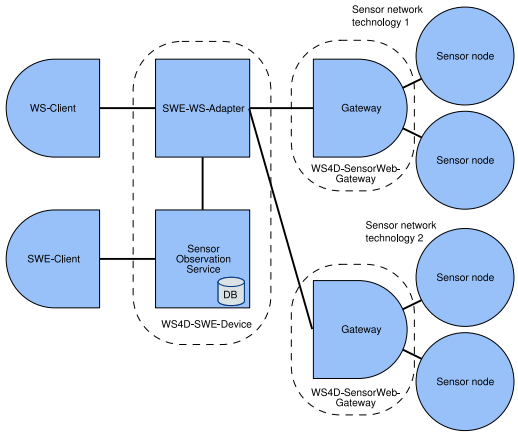


**Figure 4. Sensor web architecture overview**

At the moment this architecture does not cover all aspects of SWE. We keep the focus on the SOS and add the other services of SWE later.

## 4.1 SWE-WS-Adapter

As DPWS is based on SOAP web services, a SOAP web service interface for the SOS service is required. OGC web services have similar conventions to SOAP web services. Both have methods, parameters and a request response message exchange pattern. The data formats in SWE are defined with XML Schema which is also used to define data formats for SOAP web services. So it was quite easy to define a WSDL interface description that is similar to the interface of the SOS. The SOAP web service or the proxy, respectively, created in this way, has the same methods as SWE, i.e. *GetCapabilities*, *GetObservation*, *RegisterSensor*, *InsertObservation*, *DescribeSensor*, and uses the SWE data formats.

To announce the SWE-WS-Proxy in a network with DPWS, we defined the device type *WS4D-SWE-Device*. This device type indicates that the device hosts an SWE-WS-Adapter service that can be used by the generic gateway to register sensors and insert observations.

## 4.2 Generic Gateway

The generic gateway architecture focuses on keeping the sensor network dependent part as small as possible by using the SWE specifications and integrate into the service oriented aspects of DPWS. It consists of several components as shown in figure 5.

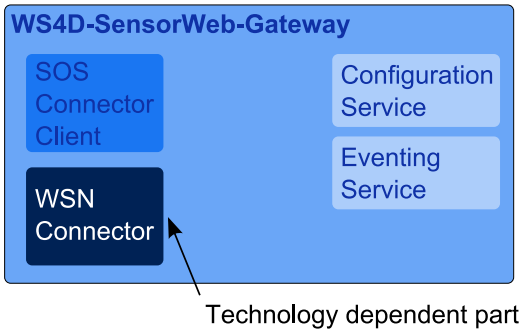


**Figure 5. Generic Gateway Architecture**

The SOS Connector component is a SOAP web service client component that uses an SWE-WS-Proxy service to register sensors and insert observations. This component is fed with sensor data by the WSN connector component. The WSN connector component is the only component in this architecture that depends on the sensor network technology. This component can be exchanged to integrate new sensor network technologies easily. In most cases implementation of this component should be a simple adaptation of the exist gateway code of the sensor network which should be integrated. The generic gateway provides a configuration service that can be used to configure the settings of the gateway. These settings include which *WS4D-SWE-Device* the gateway should use. If the WSN doesn't provide sensor data with location, the configuration service can be used to define the location for sensors. The gateway will then provide the sensor data with this configured location. If the gateway should feed other systems with sensor data, these systems can use the eventing service to get the sensor data with a push mechanism.

To indicate that a device supports the services and components described above, it announces the *WS4D Sensor Web Gateway* device type. This can be used by configuration and management tools to supervise these devices.

# 5 WS4D Sensor Web Gateway

## 5.1 Example Scenario

The architecture described above is implemented for a concrete scenario. Before going into detail of the implementation itself, this scenario is shortly described in this

subsection. Our scenarios' data source is a WSN consisting of ez430-RF2480 motes [1]. This ZigBee based network contains one network coordinator which is used as data sink. For data transmission, this special node is connected to the serial port of our gateway.

The gateway transforms the given data into an SOS compatible soap message and sends it to the SWE-WS-Proxy which removes the soap envelope, which is needed for DPWS, and hands the data to a 52°North SOS server. The use of DPWS allows dynamic finding of the SOS or the proxy, respectively by the gateway. Then a data consumer, e.g. a user client, connects to SOS which can be found dynamically to receive data. All mentioned components are illustrated in figure 6.

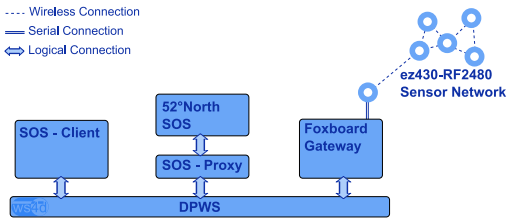


**Figure 6. Scenario**

## 5.2 Implementation

As mentioned in our architecture description, our data management solution consists of three main modules, i.e. an SOS Server, an SWE-WS-Proxy and a gateway for coupling WSNs. In this section we describe main aspects of implementation, starting with SOS server.

### 5.2.1 SOS Server

For realization of the SOS component, which is part of SWE, we studied actual open source implementations of SWE. We identified the current implementation of 52°North project [5] as the implementation which fits best for our use and is also one of the most complete implementations of SWE. SOS is realized as a web service application, using Java servlets. Therefore it has to be deployed in a servlet container. In our implementation we use Apache Jakarta Tomcat 6.0.16 in conjunction Java Runtime Environment (JRE) 1.6.0. In our case both are installed on a WindowsXP Operating System (OS) with Service Pack 2 (SP2). For data storage a PostgreSQL 8.2.6 database with PostGIS 1.3.2 Geographic Information System (GIS) extension is used. For compiling and deployment purposes Apache Ant 1.6.5 is used. A data model for use in PostgreSQL is defined by SOS which provides an SQL file to build the base structure. Database name and address as well as username, password and some additional data base related settings have to be edited in SOS configuration. In a similar way properties concerning runtime environment, i.e. tomcat and java, have to be set before deploying the web application.

### 5.2.2 SWE-WS-Proxy

When the described installation process is done, a 52°North SOS is running. In the next step this service has to be adjusted to work as a DPWS Service. As mentioned earlier, SOS only provides OGC web service access using HTTP/Post and HTTP/Get without any discovery mechanism. There are two possibilities to achieve DPWS access to SOS. The first one is to adapt SOS by changing the servlet itself. The second solution, which we used, consists of a proxy service that is capable of transforming DPWS SOAP messages into HTTP and vise versa. The most important advantage of this solution is reusability. Due to the fact that SOS is still under development of 52°North, it is highly expected that newer versions will be available in future. By using a DPWS proxy service, it is most likely that a new version can be operated without modifying the proxy. It is even possible to change the SOS implementation itself without having to change the proxy.

To achieve DPWS capabilities, WS4D axis2 DPWS implementation [8] is used for proxy implementation. WS4D enhances axis2 by DPWS, while the use of axis2 enables us to run our proxy within the same servlet container as SOS. The axis2 code generator was used for creating java code from a WSDL file, which was built upon the OGC specification which is given as xml schema files. Due to the extent of these schema files, only an essential subset of the specification was chosen. For data binding we had to choose XMLBeans due to the extensive use of special schema elements which are not supported by axis standard binding. Our proxy implements the following SOS operations: *GetCapabilities*, *GetObservation*, *RegisterSensor* and *InsertObservation*. The proxy extracts all necessary information from incoming soap messages and generates OGC compatible SOS requests. Replay messages from SOS become encapsulated in soap messages and are send back to the initiator.

### 5.2.3 Generic WSN Gateway

Third part of our architecture is a gateway to connect various WSN implementations to SOS. This is physically realized by an embedded system, i.e. Foxboard [22]. This

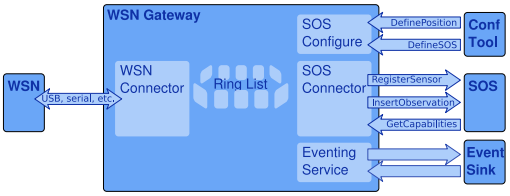


**Figure 7. WS4D Sensor Web Gateway implementation**

platform is based on a 100 MHz RISC SOC processor called ETRAX 100 LX and has 8MB of flash memory and 16 MB of RAM. It hosts an embedded linux and provides various input and output interfaces, e.g. ethernet, serial port Universal Serial Bus (USB) and General Purpose I/O (GPIO). The software part of the gateway consists of several parts as depicted in figure 7 implemented in C. On the one hand there are SOS specific parts like an SOS connector, which transmits data to SOS, and a configuration service, which provides the possibility to configure the gateway, i.e. specify SOS to connect to and optionally position information to be added to sensor measurements. These parts need DPWS capabilities, that are achieved by using WS4D-gSOAP. WS4D-gSOAP bases on gSOAP. Both are optimized for small embedded systems und thus small memory and cpu requirements. gSOAP uses a code generator to generate to C code that directly marshalls and demarshalls XML messages to C structures. Thus gSOAP offers high throughput and has low memory requirements as shown in [15].

The SOS connector is implemented as a DPWS client that connects to the SOS. Like it is done for proxy implementation, connector code was widely generated with the help of the before mentioned WSDL. As the data types used in SWE are based on GML and are quite complex the marshalling and demashalling code is a big part of the resulting executable.

The SOS connector is implemented as a threat, running concurrently with the WSN connector. While the SOS connector is consuming data to send it to the SOS, the WSN connector produces data from WSN inputs. Data exchange between both threads is realized through a list of measurements.

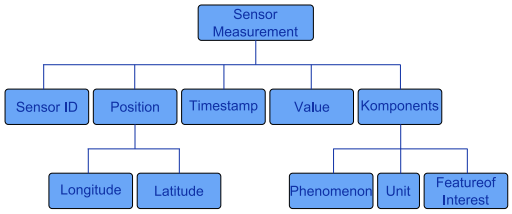


**Figure 8. Data structure for sensor data of WSN connectors**

The WSN connector is either listening to a serial port, a USB port or any other interface which is connected to a WSN sink node, depending on the specific WSN. Connectors task is to decode the incoming data stream, to extract the necessary data as depicted in figure 8 and to place the result in the connecting ring list. It is the only WSN specific part within the presented architecture. Therefore only this part, which is easy to exchange, has to be adapted when changing the WSN. As an example we implemented two WSN connectors. The first one connects a WSN of ez430-RF2480 nodes via serial port. The second connector is for exchanging data with a Meshbean [2] based WSN which is connected via USB. Phenomena measured by these WSNs include temperature, concentration of carbon monoxide and hydrogen, light intensity, acceleration and battery power, which are specific parameters of experiments performed in LSA. To ease the extension with new WSN connectors the Gateway has an API to add new measurements to the measurements list and to provide the sensor data description.

## 6 Conclusion

In this work we presented a generic gateway architecture, which enables access to Wireless Sensor Network (WSN) data, fully transparent and independent form hardware, software, data format and data collection paradigm of the used network. Main components are generic gateway, Sensor Observation Service (SOS) and the SWE-WS-Proxy. The describes architecture is logically located between WSN and data consumer, replacing the gateway of generally used WSNs. Collected data is enriched with spatial and temporal information which allows spatial data request as well as data series, which is important for LSA application. Sensor Web Enablement (SWE) was combined with Devices Profile for Web Services (DPWS) to reduce the administration overhead through real plug and play support for gateways and to improved the flexibility and the reusability of components and services. The architecture demonstrates how this two technologies can be combined to leverage the advantages of both technologies. The concept was implemented with tools of the WS4D initiative and the SWE implementation of the 52°North project to demonstrate a scenario that fits into the Live Science Automation (LSA) domain.

## References

[1] ez430 RF2480 - Texas Instruments, manufacturer home page. http://focus.ti.com/docs/toolsw/folders/print/ez430-rf2480.html.

[2] ZigBit Developement Kit by Meshnetics. http://www.meshnetics.com/dev-tools/zdk/.

[3] ZigBee Specification version 1.0, ZigBee Alliance, 2004. http://www.zigbee.org.

[4] Web Services on Devices, 2008. http://msdn.microsoft.com/en-us/library/aa826001(VS.85).aspx.

[5] 52°North Initiative for Geospatial Open Source Software, 2009. http://52north.org/.

[6] 6LoWPAN working group, 2009. http://tools.ietf.org/wg/6lowpan/.

[7] Service-oriented Architectures for Devices, 2009. http://www.soa4d.org.

[8] Web Services for Devices, 2009. http://www.ws4d.org.

[9] R. Behnke, F. Golatowski, K. Thurow, and D. Timmermann. Wireless Sensor Networks for Life Science Automation. In *International Forum Life Science Automation*, 2007.

[10] R. Behnke and D. Timmermann. AWCL: Adaptive Weighted Centroid Localization as an efficient Improvement of Coarse Grained Localization. *Positioning, Navigation and Communication, 2008. WPNC 2008. 5th Workshop on*, pages 243–250, March 2008.

[11] H. Bohn, A. Bobek, and F. Golatowski. SIRENA - Service Infrastructure for Real-time Embedded Networked Devices: A service oriented framework for different domains. In *International Conference on Networking (ICN)*, 2006.

[12] M. Botts, G. Percivall, C. Reed, and J. Davidson. OGC Sensor Web Enablement: Overview And High Level Architecture, 2007.

[13] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-MAC: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *SenSys '06: Proceedings of the 4th international conference on Embedded networked sensor systems*, pages 307–320, New York, NY, USA, 2006. ACM.

[14] W. Dostal, M. Jeckle, I. Melzer, and B. Zengler. *Service-orientierte Architekturen mit Web Services*. Elsevier, 2005.

[15] M. Govindaraju, A. Slominski, K. Chiu, P. Liu, R. van Engelen, and M. J. Lewis. Toward characterizing the performance of soap toolkits. *Grid Computing, IEEE/ACM International Workshop on*, 0:365–372, 2004.

[16] F. Jammes, A. Mensch, and H. Smit. Service-oriented device communications using the devices profile for web services. In *3rd International Workshop on Middleware for Pervasive and Ad-Hoc Computing (MPAC05) at the 6th International Middleware Conference*, 2005.

[17] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An Operating System for Sensor Networks. pages 115–148. 2005.

[18] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

[19] Microsoft, Intel, Ricoh, Lexmark. *Devices Profile for Web Services*, 2006. `http://schemas.xmlsoap.org/ws/2006/02/devprof/`.

[20] I. Rhee, A. Warrier, M. Aia, J. Min, and M. Sichitiu. Z-MAC: A Hybrid MAC for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 16(3):511–524, June 2008.

[21] E. Y. Song and K. B. Lee. STWS: A Unified Web Service for IEEE 1451 Smart Transducers. In *IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT, VOL. 57, NO. 8*, 2008.

[22] A. Systems. FOXLX home page - Embedded Linux single board computer. `http://foxlx.acmesystems.it/?id=4`.

[23] World Wide Web Consortium (W3C). *Web Services Architecture*, 2004.

[24] Y. Yao and J. Gehrke. The cougar approach to in-network query processing in sensor networks. *SIGMOD Rec.*, 31(3):9–18, 2002.

[25] W. Ye, J. Heidemann, and D. Estrin. An energy-efficient MAC protocol for wireless sensor networks. *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 3:1567–1576 vol.3, 2002.

[26] E. Zeeb, A. Bobek, H. Bohn, and F. Golatowski. Service-Oriented Architectures for Embedded Systems Using Devices Profile for Web Services. In *2nd International IEEE Workshop on SOCNE 07*, 2007.