# WEB-OF-THINGS GATEWAYS FOR KNX AND ENOCEAN NETWORKS

Gérôme Bovet[1,2]; Jean Hennebert[2,3]

[1]*LTCI, Telecom ParisTech, 46 rue Barrault, 75013 Paris, France*
[2]*CoSI, HES-SO//Fribourg, Bd. de Pérolles 80, 1700 Fribourg, Switzerland*
[3]*DIUF, University of Fribourg, Bd. de Pérolles 90, 1700 Fribourg, Switzerland*

## ABSTRACT

Smart buildings tend to democratize both in new and renovated constructions aiming at minimizing energy consumption and maximizing comfort. They rely on dedicated networks of sensors and actuators orchestrated by management systems. Those systems tend to migrate from simple reactive control to complex predictive systems using self-learning algorithms requiring access to history data. The underlying building networks are often heterogeneous, leading to complex software systems having to implement all the available protocols and resulting in low system integration and heavy maintenance efforts. Typical building networks offer no common standardized application layer for building applications. This is not only true for data access but also for functionality discovery. They base on specific protocols for each technology, that are requiring expert knowledge when building software applications on top of them. The emerging Web-of-Things (WoT) framework, using well-known technologies like HTTP and RESTful APIs to offer a simple and homogeneous application layer must be considered as a strong candidate for standardization purposes. In this work, we defend the position that the WoT framework is an excellent candidate to elaborate next generation BMS systems, mainly due to the simplicity and universality of the telecommunication and application protocols. Further to this, we investigate the possibility to implement a gateway allowing access to devices connected to KNX and EnOcean networks in a Web-of-Things manner. By taking advantage of the bests practices of the WoT, we show the possibility of a fast integration of KNX in every control system. The elaboration of WoT gateways for EnOcean network presents further challenges that are described in the paper, essentially due to optimization of the underlying communication protocol.

*Keywords: Smart Buildings, Web-of-Things, RESTful, KNX, EnOcean, Gateways*

## INTRODUCTION

In recent years, building management systems (BMS) have become very common in various types of buildings, such as offices, manufactures or even private households. Motivated by raising energy costs and by the importance of the comfort, complex management strategies have been developed. Modern BMS include many kinds of sensing and actuating devices, managing the HVAC (Heating, Ventilation and Air Conditioning), the lightening, doors opening, windows and blinds control, and also security access systems. Buildings have become "smart" and are now including complete information systems using dedicated building management networks for communication, as for example KNX, BACnet, or LonWorks. KNX is actually the most used network in Europe. Another emerging standard for interconnecting sensors and actuators in buildings is EnOcean, principally

based on energy harvesting wireless technologies. Unfortunately, such building management networks do not offer a standardised way to interact with devices connected to them from an application point of view. Due to this, it becomes difficult to build BMS combining multiple networks. This situation can be found in buildings where the network should evolve with new devices that are not compatible with the actual one, or where extending the wiring is not feasible because of physical constraints [1]. This is leading to heterogeneous building management networks. While it exists gateways encapsulating the specific telegrams of the building management network in IP packets, there is actually no standard at the application level, resulting in the BMS having to understand and to implement every network protocol.

Looking now at Internet and Web technologies, so called Web services are nowadays widespread, able to make heterogeneous information systems (IS) interoperable. They are platform independent and use well-known standards for structured data exchange. The Simple Object Access Protocol (SOAP) is an example of Web service protocol specification relying on Extensible Markup Language (XML) for its message format and Hypertext Transfer Protocol (HTTP) for message negotiation and transmission. Unfortunately SOAP is not well suited for accessing sensors and actuators that present severe constraints in terms of memory and computing capacities. On the other hands, so-called Internet Of Things (IoT) paradigms are now emerging to qualify small IP based communicating devices. The latest development of IoT includes applications layers defining somehow how programming interfaces can be elaborated on top of the HTTP protocol. This extension of IoT principles is called Web-of-Things (WoT), offering new ways for accessing things in a resource-oriented architecture (ROA) [3].

Trying to ease the development of applications using KNX devices has been explored in different works. A first attempt was realized with the BCU SDK [4], which consists of a script generating C++ classes representing devices capabilities. A more Web oriented approach has been realized in [5]. The principle was to expose KNX functionalities as Web services by using the oBIX (Open Building Information Exchange) standard, which is a special XML schema for representing building data and operations. Unfortunately, oBIX is not at all widespread in BMS, probably because of its relatively complex XML schema. In addition to this, the proposed implementation does not allow an easy integration of the gateway in an existing environment, requiring an important configuration effort for large networks.

## THE WEB-OF-THINGS

The Web-of-Things framework fills the gap left by the Internet-of-Things regarding the application layer [2]. It is leveraging on well-accepted standards of the Web to build Application Programming Interfaces (APIs) to things. In this framework, things are representing resources identified by URLs and manageable using the verbs of the HTTP protocol to form the so-called RESTful APIs. In the WoT, every capability or property of a device is considered as a resource. For example, a temperature sensor could return the measured value both in Celsius and in Fahrenheit. More precisely, some resources can allow multiple operations as read and write. So, we first need to be able to identify and address those resources in a simple way before we can interact with them. This is realized by using URLs in the same way as for retrieving Web pages on servers. An advantage of this approach is in its hierarchical way to organize resources reflecting the physical world. This principle is

shown in figure 1. For uld use the following
URL: *http://<DOM. erature/celsius.*

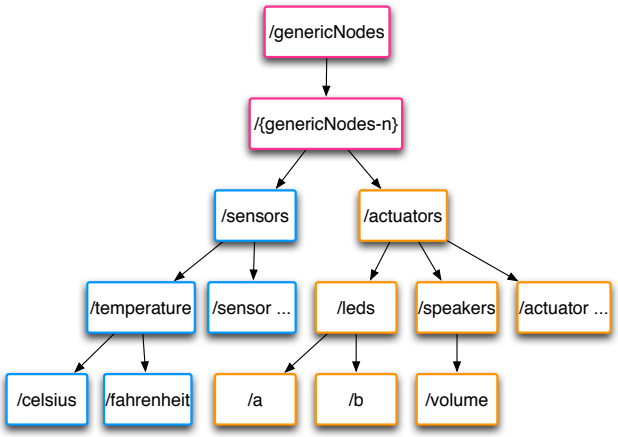


*Figure 1: Resource hierarchy example of an abstract node*

The domain part of the URL also allows to be hierarchically structured to match a virtual abstract structure or a real physical organization. In the case of buildings, URLs can give insights on the location of devices inside the organization by decomposing it into sub-domains according to buildings, floors and rooms. For communicating with endpoints, RESTful APIs are really the communication and application layers in the WoT by leveraging the HTTP protocol. Unlike SOAP, HTTP is used as application protocol and not only for transport. REST has several advantages over SOAP by having less overhead and being resource oriented, which fits naturally with physical objects. With the WoT paradigm, every object or thing is embedding a Web server exposing an API for acting with its sensing, actuating and configuration capabilities. Those services are located through the URLs as explained previously. The interaction with the resources is achieved by sending HTTP requests containing a so-called HTTP verb that can be one as follows: GET, POST, PUT and DELETE. These verbs reflect actions that can be performed on resources. The GET is for retrieving information (e.g. read a sensor's value) and POST to modify information (e.g. actuating a relay) on a resource. They are in the context of WoT the most used ones.

## MAPPINGS TO RESTFUL APIS

As previously outlined with WoT paradigms, every object is expected to embed a REST server offering an API located through URLs for interaction. Unfortunately this approach can not be applied as such to most building networks. Devices connected to KNX or EnOcean network have no IP address and therefore will not be accessible by using URLs. A way of filling this gap is to propose a gateway exposing devices functionalities in the form of RESTful APIs. The gateway will hide the complexity of the building networks and allow clients to interact with attached devices in a Web-of-Things manner. In other words, the devices will appear to other participants of the WoT as they would be embedding the API on themselves.

### From KNX

KNX describes device capabilities in terms of datapoints (communication endpoints of devices, standardized data type and size) that are located inside group objects involved in group communications between producers and consumers, basing on a multicast approach. All this information is stored inside the ETS archive file coming from the ETS configuration software. This archive contains several XML files describing the topology of the network, device datapoints and all related group objects including addresses. In order for the gateway to work with a more appropriate and smaller file, a XSL transformation is performed when the archive is loaded to filter unnecessary information. From the resulting XML file, we are now able to map KNX group endpoints to REST services. The URL of each group object is composed as follows: http://<GROUP_NAME>.<LOCATION>. <ORGANIZATION_DOMAIN>/<DATAPOINT>. Here is an example when applying this scheme for an archive file issuing from the KNX network of the EPFL's LESO building for controlling lighting: *http://light.office005.ground.leso/dpt_switch*. By emitting an HTTP GET request, one will read the actual status of the light (on or off), while a HTTP POST will allow to turn it on or off according to the payload data.

### From EnOcean

Altough EnOcean is very easy to install and configure by simply pairing devices, the mapping to REST services is more complicated than KNX. This is due to the fact that the EnOcean network is not configured or managed by a central application as ETS. The pairing of devices to form groups is done by users putting actuators in a teach-in mode, while triggering a learn telegram on sensors that have to drive the actuator. All the knowledge is stored inside the devices and there exists no possibility to retrieve it from the devices. As a consequence of this, it is not possible to automate the mapping of the EnOcean network. The user has to reproduce the configuration inside the gateway through a Web interface. However the gateway can automatically detect unknown sensors having sent a learn telegram. The user can edit the related information, set the device type, add actuators and eventually associate them together to build groups. At the time of writing this paper, only the reading of values has been explored. Unlike KNX, EnOcean sensors are not addressable so that it is not possible to read the actual value. To bypass this problematic, the gateway will respond to a read request with the latest value sent by the sensor. The URL for each sensor is composed as follows: http://<SENSOR_NAME>.<LOCATION>. <ORGANIZATION_DOMAIN>/<SHORTCUT>. The shortcut designates the data to read, as EnOcean sensors can report various kind of data like temperature, humidity in one telegram. Shortcuts are defined in the EnOcean Equipment Profile (EEP) specification. EEP are similar to KNX endpoints. Here is an example of an URL mapping to a temperature and humidity sensor: *http://air.office005.ground.leso/tmp*.

### Common functionalities

For each gateway, the REST APIs are extended with common functionalities, especially thought for reactive and proactive BMS. The first extension is the discovery of groups and device capabilities. Clients can do GET request on URLs only pointing to a location. The gateway will answer with all sub-locations or devices available in the specified location. One can also ask about the available datapoints/shortcuts for a specific device by putting the .../* placeholder at the end of the URL.

The notification paradigm is used to inform clients as soon as a value of a sensor changes.

This is achieved by a client registering on a resource and furnishing the callback that have to be called by the gateway. One has to put the *.../[un]register* keyword at the end of an URL pointing to an endpoint (e.g. *http://air.office005.ground.leso/tmp/register*).

At last, and specific for proactive BMS, clients can announce their need for storing history data on the gateway, and retrieve it later. For doing this, a client will interact with the *.../storage* sub-resource of an endpoint. It can then decide to add or remove the storage by putting the *add/remove* keywords and indicating the history size in days in the payload (e.g. *http://air.office005.ground.leso/tmp/storage/add*). Clients have then two ways for retrieving the history data. The first is by indicating the number of days one wants to go back in the history with the URL: *.../storage?days=X*. The second one is by specifying a period of time with a start and end date as follows: *.../storage?from=X&to=Y*.

## IMPLEMENTATION AND EVALUATION

We implemented both gateways on a Raspberry Pi Model B micro-controller with 512MB of memory. This tiny computer offers several ports like RJ45, HDMI and two USB. For the KNX gateway, our implementation relies on the *Calimero 2.0* Java library, providing classes and methods for KNXnet/IP tunnel communications, and datapoint object representation. The Web part of our application is composed of a Java servlet running on a *Jetty* server, known for being lightweight and optimized for constrained devices. The database is running on MySQL. As shown in figure 2, we base our implementation for KNX on several logical modules shared in different scenarios of use. The first one is the configuration of the gateway, where the administrator will provide all the necessary information for proper running. Once configured, the gateway enters in its normal operation where it can serve requests for manipulating group objects. The architecture and working of the EnOcean gateway is very similar as for KNX.

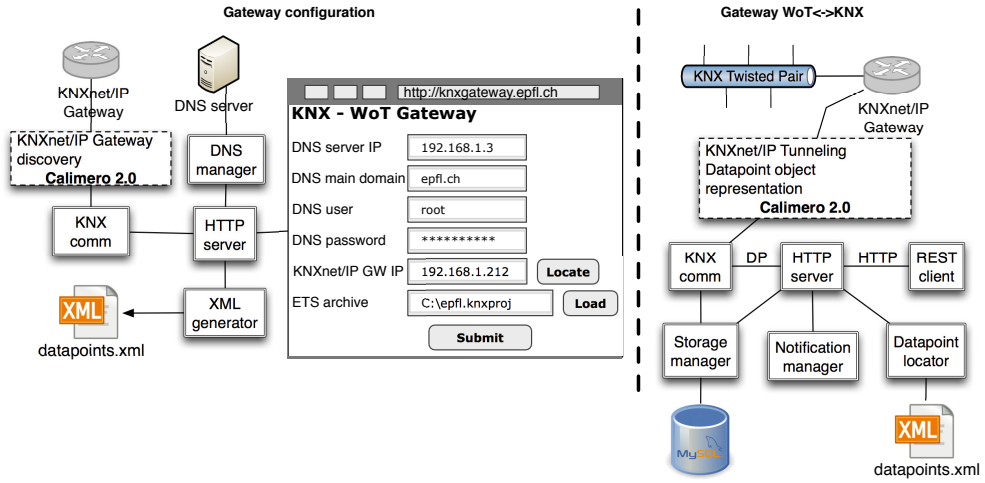


*Figure 2: Overall KNX-WoT gateway architecture illustrating the logical modules for two scenarios: gateway configuration (left part) and gateway normal operation (right part).*

At the time of writing this paper, the EnOcean gateway not yet being terminated, only the KNX one has been evaluated. We proceeded two categories of evaluations: relative to performance and the usability of the gateway. We evaluated the performance of the KNX gateway with the network of the LESO building composed of 265 devices distributed in 765 groups. Table 1 shows the results we obtained, and that can be considered as

totally acceptable for such an installation. At last, three developers tested the ease of integration of KNX using our gateway. Their feedback was very positive by admitting that our gateway offering RESTful APIs allows to significantly reduce the development time of an application as it would be the case if they would need to implement the KNX protocol.

| Measure type | Result |
|---|---|
| Maximum HTTP requests per second | 45 |
| Maximum simultaneous HTTP requests | 620 |
| Average event reaction time | 33 [ms] |

*Table 1: Gateway performance measured on a KNX installation running 265 devices*

## CONCLUSION

Inspired by Web-of-Things paradigms, we explored the feasibility and benefits of using well-known web standards like HTTP and RESTful APIs to interface KNX/EnOcean networks and building management systems. We proposed an architecture for a KNX-WoT gateway that has been validated through an implementation on a low-cost Raspberry Pi and validated on a realistic KNX configuration. Positive feedbacks were also returned by developers of building management systems thanks to the simplicity of use of WoT APIs. Generally speaking, we believe that WoT approaches are good candidates to facilitate the integration of heterogeneous networks. We also believe that building management systems will have to dialogue with various networks in a near future as new technologies are emerging, such as for example EnOcean. Our future works will cover security aspects of the gateway through authentication and encryption of data to prevent misuse.

## ACKNOWLEDGEMENTS

This work was supported by the research grant Green-Mod of the Hasler Foundation and the research grant EE-WoT of the HES-SO.

## REFERENCES

1. G. Bovet and J. Hennebert. The web-of-things conquering smart buildings. volume 10s/2012, pages 15–19. ElectroSuisse, 2012.
2. D. Guinard. *A Web of Things Application Architecture – Integrating the Real-World into the Web*. PhD thesis, ETHZ, 2011.
3. D. Guinard, V. Trifa, F. Mattern, and E. Wilde. From the internet of things to the web of things : Resource oriented architecture and best practices. In D. Uckelmann, M. Harrison, and F. Michahelles, editors, *Architecting the Internet of Things*, pages 97–129. Springer Berlin Heidelberg, 2011.
4. W. Kastner, G. Neugschwandtner, and M. Kögler. An open approach to eib/knx software development. In *Fieldbus Systems and their Applications*, pages 255–262, 2005.
5. M. Neugschwandtner, G. Neugschwandtner, and W. Kastner. Web services in building automation: Mapping knx to obix. In *Proc. of the 5th IEEE International Conference on Industrial Informatics*, volume 1, pages 87–92, 2007.