Developing Smart Homes Using the Internet of Things: How to demonstrate Your System

Ioannis Chatzigiannakis[1], Jan Philipp Drude[2], Henning Hasemann[3], and Alexander Kröller[3]

[1] Computer Technology Institute and Press "Diophantus", Patras, Greece
ichatz@cti.gr
[2] Leibniz Universität Hannover, Germany
jpdrude@googlemail.com
[3] Technische Universität Braunschweig, Germany
{h.hasemann,a.kroeller}@tu-bs.de

Abstract. The Internet of Things (IoT) currently grows with great momentum offering the potential of virtually endless opportunities for new applications and services in the "Smart Home" context. Yet, regardless of the tremendous efforts made by the relevant research & development community, application development is still a very complex and error prone process as the large range of IoT devices and smart appliances often result to complex systems-of-systems interactions. In addition, we need to factor in the human behavior and interaction goals thus making it more difficult to understand and analyzing the operating principles of the new applications. It is therefore imperative to conduct experiments verifying the complex interactions of those systems, as well as to be able to demonstrate and showcase them; to give users clear evidence how the system around them will behave. In this work we present two demonstrators that we have developed during the past years in order to provide a generic environment for showcasing new applications and services in a "Smart Home" context. We have displayed these demonstrators at several occasions, which gave us numerous opportunities to receive feedback from spectators of different backgrounds. We discuss the design choices of each demonstrator, the benefits of each approach and the experience gained from each one.

1 Introduction

The Internet of Things (IoT) currently grows with great momentum. One major driving application for this is building automation, especially in the "Smart Home" context. Smart homes combine several active and passive appliances: HVAC, entertainment media, lighting control, energy profiling, and automation of mechanical tasks are just some examples. This constitutes a broad range of appliances with diverse requirements and interaction patterns making Smart Homes a challenging environment for the design of user experiences. It is therefore important to be able to design systems that detect, analyze and react to the behavior of the users and the general environment.

N. Streitz and P. Markopoulos (Eds.): DAPI 2014, LNCS 8530, pp. 415–426, 2014.

We observe two effects: while the number of interactive embedded components grows dramatically, they increasingly penetrate the lives of their users. Their widespread deployment enable significant technical achievements, and the data they deliver is capable of supporting an almost unlimited set of high value proposition applications. A central problem hampering success is that sensors are typically locked into unimodal closed systems [PRB+11]. For example, motion detection sensors in a building may be exclusively controlled by the intrusion detection system. Yet the information they provide could be used by many other applications, for example, placing empty buildings into an energy-conserving sleep mode or locating empty meeting rooms.

Recently, new means of coping with this increase of complexity from a data integration perspective have been proposed, many of them targeted at the idea of describing elements of the IoT network in a universal way. The SPITFIRE project [1] has connected the IoT with the *Semantic Web*, a vast distributed knowledge database existing in the current World Wide Web. The unique approach of SPITFIRE goes beyond a mere linkage but elevates the embedded devices of the IoT to fully self-describing smart objects. In the final abstraction step, SPITFIRE combines the descriptions of several embedded devices and form so-called *Semantic Entities* [HKK+13], to enable reasoning about descriptions of the actual observed real-world objects. This allows for a novel style of application development: An application developer does not need to address devices individually at a low-level networking layer but can rather define interactions based on semantic descriptions available from the devices or the Semantic Web. Findings of applications then can be pushed back to devices and used as input knowledge for other applications, thus providing a universal platform that allows communication about arbitrary facts between components.

Consider a smart home application that might be defined close to natural language and read "turn off all heaters that are in the same room as an open window". Additionally, the application might enhance the devices description so the according heaters hold the information why they have been turned off, which can in turn be used by a different application, which might or might not be related to home-automation. In order to fully exploit this new connectivity on a data layer, it is imperative to be able to simulate and experiment with different states of devices and real-world objects in order to evaluate the correct behavior of the application.

Application development that takes advantage of such large range of embedded devices and smart appliances have strong dependencies on IoT systems that often result to complex systems-of-systems interactions. In addition, the need to include the human behavior and interaction in the loop further complicates this situation as the applications usually need to be compliant with a plethora of often contradicting requirements. It is therefore important to test and optimize applications in a controllable environment such as real networks of limited size [BCF+11,GKN+11]. Designers need to conduct experiments verifying the complex interactions of those systems, as well as to be able to demonstrate and

[1] `http://www.spitfire-project.eu`

showcase them; to give users an idea how the system around them will behave. As Mark Weiser announced in the nineties [Wei93],

> "the research method for ubiquitous computing is standard experimental computer science: the construction of working prototypes of the necessary infrastructure in sufficient quantity to debug the viability of the systems in everyday use".

In this work we present two demonstrators that combine of IoT devices in a controllable Smart Home environment. Their focus is on allowing for interactions that make complex processes easily understandable. These demonstrators rely on testbed structures that utilize real-world networks. They provide a rich environment for interacting with the IoT network at device level, and offer visual feedback on the complex interaction patterns triggered by user actions. A wide variety of scenarios can be accommodated so that we can carefully monitor the operation of the resulting system under different settings. The user can also interact with the demonstrator in real-time by introducing external events and control the operation of the resulting system. Both demonstrators are independent of the high-level applications thus they can be used to reliably reproduce the same series of event scenarios and test how the applications react to set of external events.

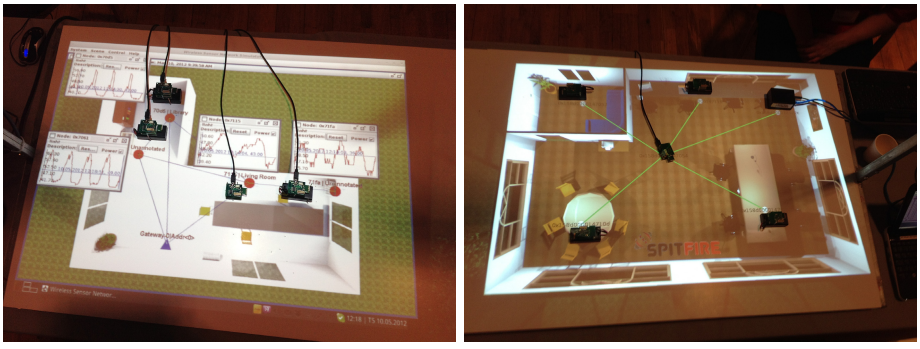

Fig. 1. The Projected House demonstrator

The first demonstrator consists of 10 iSense nodes [2] equipped with an environmental module responsible for reading the light and temperature conditions and a actuator module that can control a small fan and an LED lamp. The nodes are placed on top of a two neoFoam surfaces on designated positions. In order to monitor the status of the running demonstrator, each iSense node is connected via USB to a laptop. This connection is required in order to collect all the debug output from the nodes so that the visualization can be done at a later point. The neoFoam has been drilled so that the USB cables can be put through. The two

[2] `http://www.coalesenses.com/`

surfaces are placed side-by-side creating a combined surface of 200 x 70 cm. The laptop is also responsible for the visualization of the demonstrator. Two video projectors are connected to this machine that project the visualization on the neoFoam surface. In order to cover the whole 200 x 70 cm surface, the projectors are attached to a metal frame that allows their elevation up to 190 cm from the neoFoam, while at the same time, provides stability for the whole structure (see Fig. 1). For logistic and portability reasons, the frame consists of many smaller parts that allow its disassembly.

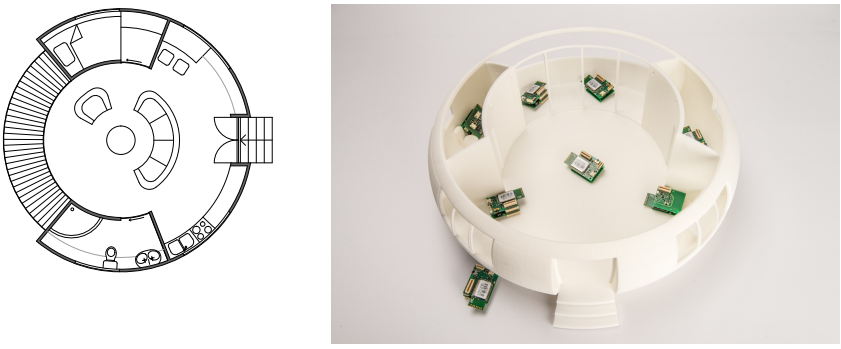

Fig. 2. The House model demonstrator: Floor plan and photo

The second demonstrator follows a different approach. Instead of having a projected visualization of the environment where the application is executed, we offer a realistic 3D environment where we the position of the IoT devices is more flexible while the interaction is done in a much more natural way. The demonstrator is composed of a physical home model (see Fig. 2) equipped with various actuators and sensors. They are attached to embedded devices and can be relocated within the model for simulating different deployment scenarios.

These demonstrators have been setup for display at various conferences in the past years, as part of a demonstration session. We have used them to show case different applications that allow the automatic control of home appliances and their configuration based on the needs and behavior of the user. Throughout all these events many visitors have had the opportunity to interact with the demonstrators in order to acquire a better understanding of the smart home applications. While both demonstrators were very valuable tools to explain the concepts of our applications and highlight the main technological characteristics, each one had different benefits and drawbacks. In this paper we present the software and hardware design of the demonstrators that we have developed and summarize the experiences gained.

2 A Complete Protocol Stack for Semantic Sensors

The generic demonstrators that we have developed offer the ability to showcase high-level user applications that utilize IoT resources connected via the Web. The actual development makes minimal assumptions on the connectivity of the IoT devices with the Web and uses well accepted standards for interacting with the devices. In this section we describe in details the software infrastructure that is used for interacting with the physical domain.

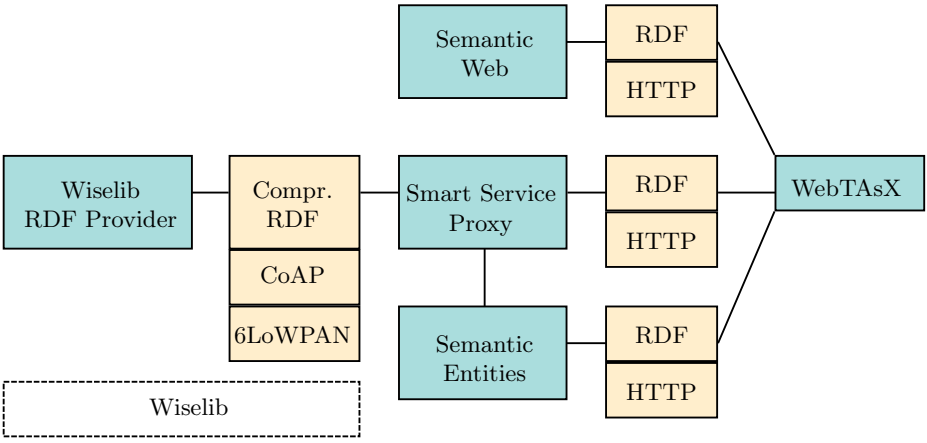

Fig. 3. Simplified software- and protocol architecture of our system. Blue boxes represent software components while khaki boxes represent communication protocols.

All of our embedded software builds on the Wiselib [BCF+10], a multi-platform embedded algorithms library. On top of that we provide the Wiselib RDF Provider [HKP12] which is responsible for managing semantically annotated data on the device such as sensor data, sensor meta data or high-level application knowledge. In order to provide these descriptions and other services (such as actuation) to the Internet, we provide a Wiselib-implementation of the standardized IPv6 interface 6LoWPAN [KMS07]. On the application layer we provide an implementation of the Constrained Application Protocol (CoAP) [FSHB11] which allows RESTful service provisioning. Together with the usage of the RDF standard for universally valid fact descriptions this allows self-describing and auto-configuring devices which connect to the Semantic Web.

In order to cache device descriptions and translate between the lightweight embedded networks protocols and Semantic Web clients that rely on protocols like HTTP, we introduce the Smart Service Proxy (SSP) [HKK+13]. The SSP collects and caches semantic descriptions produced by the devices and provides a web service endpoint for interaction with the system. An overview of these components and their interactions is given by Figure 3.

2.1 Wiselib RDF Provider

A main challenge in interacting with IoT devices is the vast variety in terms of employed hardware and available system resources. It is common to have devices with limited amount of available memory, processing power or energy – also devices have different peripheral and memory configurations (RAM, internal flash, SD cards and so forth). To overcome this heterogeneity, we adopt a storage mechanism for semantic (RDF) data that is resource-efficient, portable and provides means of grouping semantic statements into documents. This allows to make knowledge addressable and provides fine-granular control over what information is communicated for a successful retrieval of knowledge.

The Wiselib RDF provider [HKP12] provides a portable, modular and resource-efficient embedded database for the storage of RDF data. It builds on the Wiselib [BCF+10], an embedded algorithms library that focusses on modularization, resource efficiency and portability. Thanks to the Wiselib, the RDF Provider can be compiled for platforms like Contiki, TinyOS, Arduino or Android and many more.

The Wiselib RDF Provider represents the stored RDF data as a set of potentially overlapping documents. This way, statements that change on a regular basis (such as those describing the current sensor value) can be requested independently from those that change rarely or never (such as the owner of the embedded device). The documents are exported over a configurable and extensible set of encodings and protocols such as the combination of a lightweight RDF compression scheme on top of CoAP and 6LoWPAN.

2.2 Constrained Application Protocol (CoAP)

The Constrained Application Protocol (CoAP) is a IETF CoRE working group draft dealing with Constrained Restful Environments [FSHB11]. It presents a web transfer protocol suitable for machine-to-machine applications such as smart energy and building automation. The protocol is designed to operate effectively in erroneous low bandwidth environments while providing a subset of HTTP's methods (GET, PUT, POST, DELETE). By this means it offers the ability to provide RESTful web services in IoT deployments.

Using CoAP makes interaction with the IoT device as simple as invoking an HTTP resource. The communication between endpoints is based on a lightweight request/response model. The message exchange is asynchronous and is based on UDP and thus connectionless. Essentially the development of the high-level application is completely decoupled from the way the IoT devices communicate and organize their network. Furthermore, by avoiding the use of any middleware to provide access to the devices, the developer does not need to acquire additional technical knowledge in order to test and evaluate a new application using the demonstrators.

2.3 Smart Service Proxy

While our architecture provides devices that are self-describing and usable without a central authority, at some point an application developer might want to access the semantic descriptions of objects such as the deployed devices or the observed real-world objects in the very same way they use other resources on the Semantic Web, namely by issuing HTTP requests. In such a scenario, user and/or application developer will usually not want to be concerned with the peculiarities of efficient access to the embedded devices that is, request congestion, checking for presence of the device, caching, configuring push- versus pull-based mechanisms and conversion between different RDF serialization formats.

The Smart Service Proxy (SSP) fills this role and more: It provides a service endpoint to the Web which can be used to access up-to-date device descriptions and control device actuation via a RESTful HTTP interface. To the embedded network, the Smart Service Proxy provides a registration service and can negotiate observe mechanisms with the embedded devices in order to save energy by only communicating when descriptions change in a way that is considered relevant. While on the "front" side, the SSP provides different RDF encodings, on the "back" side it can communicate with embedded devices using an exchangeable and extensible protocol stack, including CoAP and 6LoWPAN, but also allowing for non-standard communication protocols to ease adaption.

2.4 Semantic Entities

While a sound, accessible semantic description of embedded devices already provides a certain degree of abstraction and ensures connectivity on a data layer, an application programmer or user will usually want to be able to describe observations, state and actions in terms of real-world objects and not be concerned with the devices monitoring them.

The *Semantic Entities Service* offers that: It can access any number of Smart Service Proxies or other data sources on the Semantic Web and combine the found semantic descriptions into a semantic description of real-world objects, the Semantic Entities (SEs). Which descriptions are to be assembled into what kind of Semantic Entities can be configured by the user using a set of rules which can be injected into the system. Examples of these rules are *"Construct one SE for each object being observed"* or *"Construct one SE for each type of sensor in each room"*.

Constructed Semantic Entities are exposed in the same standardized Semantic Web format as the device descriptions or static data on the Semantic Web and can thus be seamlessly integrated.

2.5 The WebTAsX User Interface

For these situations we provide the WebTAsX User Inferface (see Fig. 4). WebTAsX provides user-friendly web frontend that can connect to Smart Service Proxies, Semantic Entities and/or static data on the Semantic Web or provided

locally by a user. WebTAsX offers the user a list of entities (that is devices, or abstract Semantic Entities), as well as other available semantic data sources and allows the straight forward creation of actuation rules.

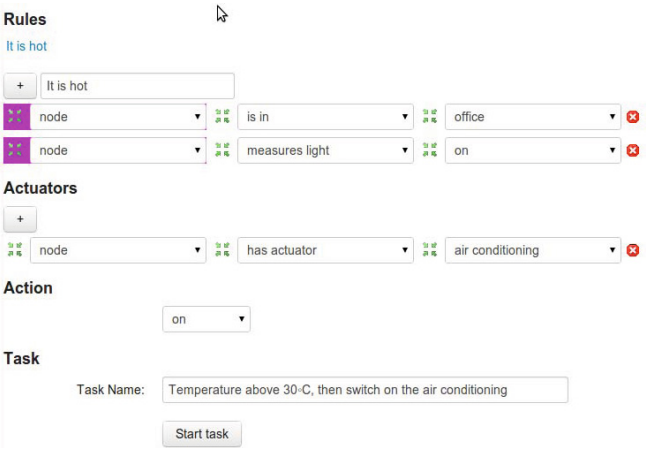

Fig. 4. Setting automation rules with the WebTAsX tool

3 Demonstrators

3.1 The Projected House Demonstrator

Our first approach to demonstrate the system was based on projection of house shapes to a planar surface. For this, two projectors were mounted on a steel framework, projecting downwards the images of two houses onto a plane where several smart objects were placed. The projectors where connected to a PC and could thus dynamically alter the projection to reflect the state of the simulation.

The projection shows two rooms, which have been rendered by a consumer interior design software. Furniture suggests an office environment, however there was no emphasis on believability during construction. One room covers most of the available space, to account for demos where many objects have to be placed in the same room. The other room is tiny, just sufficient to have some object space that does not belong to the main room.

We provided a set of sensors and actuators based on iSense nodes that were placed at different positions the virtual building. These included a switchable fan and radio as well as several light- and temperature sensors. Via additional screens visitors could observe the semantic descriptions of objects and create rules using WebTAsX.

This approach of using projectors coving the demonstration surface completely had the following advantages:

Visulazation of Abstract Data. In addition to providing the imagy of a house as substitute for a physical model, the projectors allowed us to provide several abstract visulations, such as packet flow between objects or curves plotting the recent sensor data history. This made the state and otherwisely invisible actions of the system very visible and allowed the visitor to observe live what influence on the internal state of the system certain actions have.

Daylight Cycle Simulation. The projected house model was not just presented as a static image but rather its lighting conditions over the course of a day were visualized in form of a full-sized animation. In order to be able to discuss long-term sensor value analysis in a descriptive way, the simulated day was configured to pass in a matter of minutes.

As our smart objects were equipped with light sensors they could pick up the simulated lighting condition produced by the projector.. This allowed an interesting way of presenting: While the processed sensor values were indirectly controlled by the daylight simulation, they were still produced by actual light (from the projector) being picked up by light sensors, a process that could easily be influenced by the demonstrator or a spectator to observe the effect on the descriptions of the nodes and auto annotation system.

Tangible Objects. The smart objects could be freely moved, switched and replaced by the user. This allowed for example to demonstrate the semantic auto annotation by bringing in a new, unconfigured device, placing it in the virtual building and switching in it on. Objects equipped with light sensors could thus be moved when to a place with different light situation or influenced directly using a flashlight. This way, the user could directly observe the effects of his actions on the system.

3.2 The Model House Demonstrator

The central piece of the demonstrator is a model of a house, developed with easy interaction in mind. There is a large main room, which can hold about five smart objects. This is sufficient for complex demonstration scenarios of pervasive systems; the user can put both sensors and actuators into it, and define operational rules for them. LED lighting in the walls can be used to showcase home automation applications, where the light level automatically changes based on user-induced sensor input. There are three additional rooms. These are smaller, capable of holding one or two smart objects, with the purpose to show how a system can use location information to provide separate services for different parts of the home.

The smart objects are bare-bone sensor and actuator hardware, powered by batteries or USB connections. They are used without casing, to let the user obtain a feeling for the hardware. We used different products, including Arduino and iSense devices, as well as most Contiki- or TinyOS-driven sensor nodes.

The architectural design of the model has been extracted from futuristic architecture of the 1960s. The ideas behind those designs having always been a mechanization in living, the model is rather suitable for the topic of home automation. The model illustrates a small, independent housing unit, consisting of

a bathroom, a bedroom, which is not much more than a bunk, a terrace and a larger living room with kitchenette.

In contrast to the designs from the 1960s, which came mostly out of student designs with low budgets and defective constructions, this model can actually be imagined in a technological context. Building automation and a technological construction process with prefabricated parts in small numbers are economically reasonable nowadays and could lead to inexpensive, mobile housing units.Nonetheless the simple floor-plan and design ask for adaption by a human being. In its cold self the design does not seem to inviting, but it has the big advantage, that it can be customized. In color, in material, in facilities, in furniture. This is actually an important way of thinking in contemporary architecture, that most architects lacked in the 1960s. In that time, architects would invent completely new ways of living, in new structures, taken from science-fiction novels. This would certainly lead to a synchronized society, where cities would look very much alike, containing similar housing units, which would tend to peoples every needs; a society that appears to be like Aldous Huxley's Brave New World. Having overcome this kind of thinking for good and for bad, architects nowadays think more individualized. Structures like our model are still thinkable, but only if they can be customized, very much like buying a car.

The design was very much affected by its final outcome being a model. While a simple cubic building would have been more economic, if it would have been a real building, nonetheless that was not what we were looking for, which is easy interaction and vividness. This allowed us to trade some preferrable properties for real houses for a more appealing style. The model fits its purpose: It can arouse peoples interest by looking different, however fitting.

Interactions. Through the provided hardware/software stack, our system offers a unique user experience: A device can be installed by just switching it on and placing it in one of the rooms of the building model. Through self-description and auto-configuration mechanisms, the description of the device becomes available to the system so the device is considered in future events.

Using a frontend, the user can define rules like "When the weather report says it is above 24 degrees and it is between 4pm and 8pm, turn on the air condition". All available devices will be used to fulfill this request. This way the impression of a "smart" home is completed: Without any need for technicalities, the system will try to fulfill the request of the user, incorporating available sensors and actuators as well as the knowledge available in the Semantic Web.

4 Experiences

The projected house was used at several occasions which gave us numerous opportunities to receive feedback from spectators of different backgrounds. In October 2012, the demonstrator was used to demonstrate "True Self-Configuration for the Internet of Things" [CHK+12] at the Internet of Things Challenge

Competition [3] in Wuxi, China. In a process that combined votes from a jury and the audience, the demo was awarded third prize.

In 2013, the yearly Future Internet Assembly [4] was held in Dublin. Again, the projected house demonstrator was used to show and discuss the latest advances of SPITFIRE, in this case in the *Hands on FIRE* demo session. In contrast to the former setups, for the FIA we decided to split the demo into two parts: One focussing on the specifics of our CoAP implementation, the other one – using the projecting demonstrator – focussed more on the interaction of all SPITFIRE components. This demo was again a great success, as it received the best demo award, chosen out of a total of 17 demonstrations.

More importantly, on all occasions we received valuable feedback from spectators: While the tangibility and the possibility to influence the simulation in such a direct way were positively noted. The visitors noted however that the smart objects placed on a plane surface did not really convey the message of a smart building but rather of a collection of objects. While ths is sufficient to demonstrate a protocol stack it missed out on sufficiently illustrating the amount of ubiquity and integration, SPITFIRE provides. By the use of USB cabling and because of the not location-aware visualization, displacing the objects yielded a slightly inconsistent experience: While the sensor values would adapt to the potentially changed lighting situation, the visual aids for packet flow and annotations would still be placed at the devices' original position.

From a exhibitors point of view we have to mention a notable amount of overhead in transporting (often via airplane) and assembling several bars of metal framework and two projectors.

The Model House demonstrator addresses most of these issues: By the use of RFID we can locate devices and thus have them freely movable with a consistent experience. Consisting of a single piece, it is almost trivial to transport and set up and, first and foremost provides a much more natural and believable model of a smart home: Location mechanisms are integrated into the floor, light is produced by LEDs that simulate light-bulbs, rooms can have individual (natural) lighting situation; the house itself is three-dimensional and tangible.

Acknowledgments. This work was partially supported by the European Union under contract number ICT-2009-258885 (SPITFIRE).

References

BCF+10. Baumgartner, T., Chatzigiannakis, I., Fekete, S., Koninis, C., Kröller, A., Pyrgelis, A.: Wiselib: A generic algorithm library for heterogeneous sensor networks. In: Silva, J.S., Krishnamachari, B., Boavida, F. (eds.) EWSN 2010. LNCS, vol. 5970, pp. 162–177. Springer, Heidelberg (2010)

BCF+11. Baumgartner, T., Chatzigiannakis, I., Fekete, S.P., Fischer, S., Koninis, C., Kröller, A., Krüger, D., Mylonas, G., Pfisterer, D.: Distributed algorithm engineering for networks of tiny artifacts. Computer Science Review 5(1), 85–102 (2011)

[3] `http://iot2012.org/`

[4] `http://fi-dublin.eu`

CHK+12. Chatzigiannakis, I., Hasemann, H., Karnstedt, M., Kleine, O., Kröller, A., Leggieri, M., Pfisterer, D., Römer, K., Truong, C.: Demo: True self-configuration for the IoT. In: IoT Challenge Competition, Internet of Things International Conference for Industry and Academia, IEEE (2012) (to appear)

FSHB11. Frank, B., Shelby, Z., Hartke, K., Bormann, C.: Constrained application protocol (CoAP). IETF draft (July 2011)

GKN+11. Gluhak, A., Krco, S., Nati, M., Pfisterer, D., Mitton, N., Razafindralambo, T.: A survey on facilities for experimental internet of things research. IEEE Communications Magazine 49(11), 58–67 (2011)

HKK+13. Hasemann, H., Kleine, O., Kröller, A., Leggieri, M., Pfisterer, D.: Annotating Real-World Objects Using Semantic Entities. In: Demeester, P., Moerman, I., Terzis, A. (eds.) EWSN 2013. LNCS, vol. 7772, pp. 67–82. Springer, Heidelberg (2013)

HKP12. Hasemann, H., Kröller, A., Pagel, M.: RDF provisioning for the Internet of Things. In: 3rd IEEE International Conference on the Internet of Things, pp. 143–150 (2012)

KMS07. Kushalnagar, N., Montenegro, G., Schumacher, C.: IPv6 over low-power wireless personal area networks (6LoWPANs): Overview, assumptions, problem statement, and goals. Technical report, IETF Secretariat (2007)

PRB+11. Pfisterer, D., Römer, K., Bimschas, D., Kleine, O., Mietz, R., Truong, C., Hasemann, H., Kröller, A., Pagel, M., Hauswirth, M., Karnstedt, M., Leggieri, M., Passant, A., Richardson, R.: Spitfire: Toward a semantic web of things. IEEE Communications Magazine 49(11), 40–48 (2011)

Wei93. Weiser, M.: Some computer science issues in ubiquitous computing. Communications of the ACM 36(7), 75–84 (1993)