From Heterogeneous Sensor Networks to Integrated Software Services: Design and Implementation of a Semantic Architecture for the Internet of Things at ARCES@UNIBO

Cristiano Aguzzi*, Francesco Antoniazzi*, Paolo Azzoni‡, Luciano Bononi*, Francesco Brasini*, Roberto Canegallo§, Alfredo D'Elia*, Angelo De Lisa*, Marco Di Felice*, Eleonora Franchi*, Luca Perilli*, Luca Roffia*, Luca Sciullo* Roberto Siagri‡, Martina Verardi*, Tullio Salmon Cinotti*
* University of Bologna, Via Zamboni 33, Bologna, Italy
‡ Eurotech S.p.a., Via Fratelli Solari 3/a, 33020 Amaro (Udine), Italy
§ STMicroelectronics, Agrate Brianza, Italy

***Abstract*—The Internet of Things (IoTs) is growing fast both in terms of number of devices connected and of complexity of deployments and applications. Several research studies analyzing the economical impact of the IoT worldwide identify the interoperability as one of the main boosting factor for its growth, thanks to the possibility to unlock novel commercial opportunities derived from the integration of heterogeneous systems which are currently not interconnected. However, at present, interoperability constitutes a relevant practical issue on any IoT deployments that is composed of sensor platforms mapped on different wireless technologies, network protocols or data formats. The paper addresses such issue, and investigates how to achieve effective data interoperability and data reuse on complex IoT deployments, where multiple users/applications need to consume sensor data produced by heterogeneous sensor networks. We propose a generic three-tier IoT architecture, which decouples the sensor data producers from the sensor data consumers, thanks to the intermediation of a semantic broker which is in charge of translating the sensor data into a shared ontology, and of providing publish-subscribe facilities to the producers/consumers. Then, we describe the real-world implementation of such architecture devised at the Advanced Research Center on Electronic System (ARCES) of the University of Bologna. The actual system collects the data produced by three different sensor networks, integrates them through a SPARQL Event Processing Architecture (SEPA), and supports two front-end applications for the data access, i.e. a web dashboard and an Amazon Alexa voice service.**

I. Introduction

Context-awareness is a key ingredient of virtuous behaviors over Internet of Things (IoT) scenarios, both by humans and machines [1]. For example, in the energy domain, energy-aware users can significantly reduce their power consumption macroscopically, by accessing metering data and dynamic price signals, and acting consequentially [2]. Similarly, several Machine-to-Machine systems traditionally employed in smart agriculture and home/industrial automation rely on context understanding to decide what data need to be processed, and the proper actions to perform [3][4]. Awareness originates from the information concerning the environment, which are often made available by networked sensors; the pervasiveness of such devices, with more than 30 billion expected to be connected to the Internet by 2025 (Source: IDC, https://www.idc.com), gives the idea of the IoT growth worldwide. At the same time, the deployment of large-scale, context-aware IoT systems is hindered by the heterogeneity of software platforms and hardware devices available on the market, which translates into the need of interconnecting systems using different communication technologies, network protocols and data format. The problem becomes even more challenging if we consider that -in several IoT scenarios- the same sensor data must be accessed by multiple consumers: for example, the information provided by a light sensor can be used -in combination both with other sensor data and with profile information- to set the optimal ambient light, but it can influence the local energy management system of a power-autonomous wireless sensor and actuator network with nodes capable of harvesting energy from light. Hence, proper data-reuse mechanisms must be designed. However, beside constituting a research problem, interoperability has been often identified as one of the main boosting factor for the IoT growth worldwide: among others, the McKinsey report quantifies in 40% the additional IoT market value which might be provided by achieving full interoperabilty among IoT ecosystems [5].

In this paper, we investigate how to achieve efficient data-reuse and data integration on heterogeneous, large-scale IoT systems, in order to ease the deployment of next-generation context-aware services and applications. Both theoretical and practical contributions are provided: hence, the paper can be read as a scientific proposal, advancing the state-of-art of IoT software architectures, as well as a tutorial, providing details on the implementation, installation and seamless integration of heterogeneous IoT sensing platforms. Regarding the theoretical contributions, we present a generic software architecture which can fit the requirements and characteristics of most of IoT scenarios. The proposed architecture features three complementary layers, i.e. the *connectivity* layer, the *data interoperability* layer, and the *software service* layer. The system core is constituted by a semantic end-point, which

ISSN 2305-7254

is in charge of providing a common semantic description to the sensor data originated from the sensor networks, and of offering content-based publish-subscribe facilities to the upper-layer applications, including front-end dashboards and back-end automation services. Differently from other IoT platforms presented so far in the literature, our proposal does not require any architectural change to the sensing platforms: indeed, these latter can be abstracted as independent *silos* that - according to the technology currently in use- convey data from sensors to clouds/repositories via gateways. Data fusion is performed at the semantic end-point, by converting the sensor data into a domain-specific ontology. Regarding the practical contributions, we describe how the abstract architecture has been made concrete through a proof-of-concept implementation and installation at the Advanced Research Center on Electronic System (ARCES) of the University of Bologna. The sensing layer is constituted by three wireless sensor networks, using different communication technologies (e.g. Dash-7 [6], 6LoWPAN [7] on SPIRIT and LoRa [8]), each connected to a specific data repository, constituted by a cloud service (e.g. The Things Network, https://www.thethingsnetwork.org for LoRa) or by a local database instance (e.g. Apache Cassandra, http://cassandra.apache.org). From here, data are transferred to the SEPA platform [18], a semantic end-point supporting SPARQL language, and offering publish-subscribe mechanisms to the upper layer services, which can thus be notified about any change in the query result due to a sensor data update. Finally, two different applications have been implemented at service layer in order to provide user-friendly data access, one based on a Web dashboard, and another one employing speech technologies (e.g. Amazon Alexa, https://developer.amazon.com/it/alexa) for voice-based interactions.
The paper is structured as follows. Section II reviews the IoT literature regarding the data interoperability issue and approaches based on the Semantic Web. Section III describes the proposed IoT abstract architecture; the current implementation available at ARCES@UNIBO is discussed in Section IV. Conclusions and future works can be found in Section V.

II. Related Works

The SPARQL Event Processing Architecture (SEPA) presented in this paper can be framed within the research topics known as stream reasoning [22], linked stream data processing [21] and content-based publish-subscribe [23]. In particular, SEPA enables the detection and notification of events (i.e., changes over the Web of Data) by means of a content-based publish-subscribe mechanism where the W3C SPARQL 1.1 Update (https://www.w3.org/TR/sparql11-update/) and Query languages (https://www.w3.org/TR/sparql11-query/) are fully supported and used respectively by publishers and subscribers. The architecture is built on top of the W3C SPARQL 1.1 Protocol (https://www.w3.org/TR/sparql11-protocol/) and introduces the SPARQL 1.1 Secure Event protocol (http://mml.arces.unibo.it/TR/sparql11-se-protocol.html) and the SPARQL 1.1 Subscribe Language (http://mml.arces.unibo.it/TR/sparql11-subscribe.html) as means for conveying and expressing subscription requests and notifications. Linked Data changes would have an impact on the development of real applications as much as Web standards will be adopted and promoted. The main Web players, including W3C, are pushing in this direction by promoting standards and defining ontologies and vocabularies. To this purpose, the Linked Data concept and relative technologies were formalized for the first time in [9]. Regarding Linked Data dynamics, [10] provides a deep insight about aspects like discovery, granularity level, description of changes, detection algorithms and notification mechanisms. To the best of our knowledge, a first attempt to use SPARQL as the subscription language is presented in [25]. Other research works focusing on using SPARQL as a subscription language include the one by Groppe et al. [26], EventCloud [27], [28],INSTANS [13], [14], semantic event notification s ervice (SENS) [29], [30], [15], [31] a nd SmartM3 [32] (i.e., Suomalainen et al. [33] proposed a secure broker, called RIBS. Galov et al. [34] developed the CuteSIB, focusing on extensibility, dependability and portability. Viola et al. [17] proposed pySIB, targeted especially at resource-constrained computing platforms).

Other early SPARQL-based RDF stream processing approaches, including continuous SPARQL (C-SPARQL) [35], SPARQLStream [36], event processing SPARQL (EP-SPARQL) [37], continuous query evaluation over Linked Data streams (CQELS) [38] and Sparkwave [39], propose to extend SPARQL with time-windows. A time-window specifies the triples for which the query is executed; it can be defined either by the number of triples (last triples from the stream) or the time (e.g., the last 15 minutes). The window specification defines also how often the window is updated and consequently the frequency of query evaluation.

There are four notable aspects that differentiate SEPA from a window-based approach. First, the SEPA does not use windows to define the triples for which the query is evaluated (i.e., we concentrate on real-time evaluation of events within the whole system). Second, SEPA fully supports SPARQL 1.1 both to generate (i.e., update) and subscribe (i.e., query) to events, and it is based on the W3C SPARQL 1.1 Protocol (i.e., it would be transparent to clients performing SPARQL 1.1 Updates and SPARQL 1.1 Queries). Third, instead of processing individual RDF triples coming from specific RDF streams, SEPA is based on an interaction model where any agent can trigger events by modifying the context of the system with SPARQL 1.1 Update Language operations. Fourth, the SEPA detects how the results have changed from the initial query results, whereas the window-based approaches provide the whole result set whenever it is modified in any way. Because of these fundamental differences in the SPARQL event processing approaches, also the implementations of the event processing mechanisms and algorithms are totally different, as detailed in Section IV.
Last but not least, security plays also a crucial role in IoT systems. For example, the approach presented in [16] focused on providing a secure publish-subscribe mechanism for the management of complex supply chains in enterprises based on RDF. The solution proposed employed the same authorization framework used by SEPA (i.e., the OAuth 2.0 Authorization Framework (https://tools.ietf.org/html/rfc6749), suggesting also other solutions, like WebID (https://www.w3.org/2005/Incubator/webid/spec/). They adopted WebSub (https://www.w3.org/TR/2018/REC-websub-20180123/) (formerly PubSubHubBub) as a mean for conveying notifications; the same notification mechanism is also mentioned in [40].

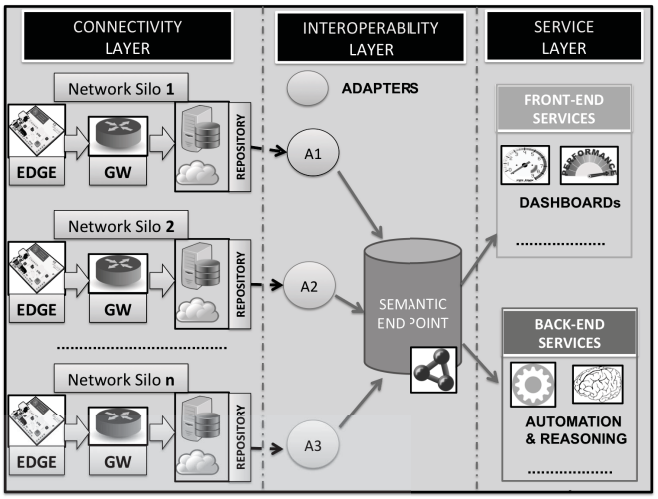


Fig. 1. The (abstract) IoT architecture for data-reuse and integration proposed in this paper

III. Architecture

Figure 1 depicts the proposed IoT architecture; we focus here on the logic data flow, while details on the implementation of the single components are provided in Section IV. The proposed architecture includes three main layers: the *connectivity* layer, the *data interoperability* layer, and the *software service* layer. The connectivity layer is composed of a set of so-called *network silos*, i.e. network architectures that are specific of a sensing/network technology. Each silo might include sensing edge devices, gateways and public/proprietary clouds or local repositories, where the sensing data are stored via any communication link. For instance, a LoraWAN silo includes: wireless sensors equipped with LoRa transceivers, LoRaWAN gateways, and the Network/Application servers (e.g. provided by the Things Network). Similarly, a short-range LR-PAN silo might include: 802.15.4/6LoWPAN sensing devices, a 802.15.4/6LoWPAN gateway and a local/remote DataBase Management System (DBMS). All the network silos are assumed to work independently, i.e. no data exchange occurs among them. The data integration among the available silos is performed on the semantic end-point (at the interoperability layer): this latter works as a semantic repository, and implements a publish-subscribe paradigm to allow sensor data access from the upper layer applications. In the semantic end-point, data are stored as RDF triples, according to a domain-specific ontology. Since the sensing data gathered by the edge-devices might have different file formats, a Semantic Adapter is needed for each silos, implementing the data conversion into the target RDF. The data transfer from each silo toward the adapter (i.e. the dotted line in Figure 1) is managed through any IoT protocol (e.g. MQTT, CoAP, HTTP, ...). Finally, the applications operating on the session layer can retrieve sensor data by interacting with the semantic end-point; the same data can be made available to multiple applications (data-reuse), moreover, the same application can consume sensor data from multiple silos in a seamless way, i.e. abstracting from the data source and from the original data format. In Fig. 1, we distinguish between two kinds of applications at the service layer, i.e.: (*i*) *front-end services*, mainly offering data-access to users or to external applications, and (*ii*) *back-end* services implementing further data processing, e.g employing semantic reasoning for autonomic, context-aware behaviours.

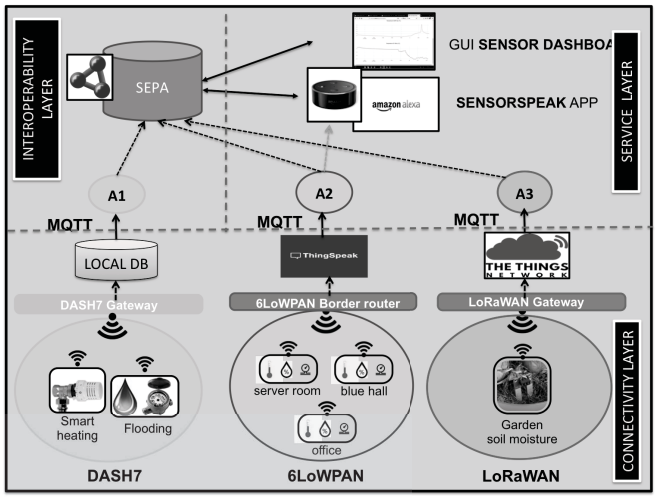


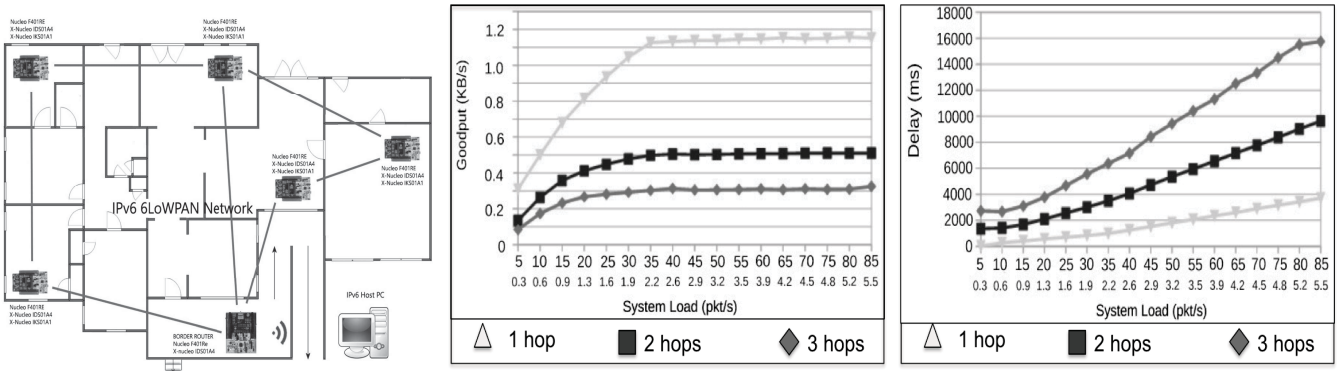
Fig. 2. The concrete IoT architecture currently implemented at ARCES@UNIBO

IV. Proof-of-concept Implementation at ARCES@UNIBO

Fig. 2 depicts the concrete version of the IoT ar-chitecture proposed in this paper, as currently deployed at ARCES@DISI. The connectivity layer is composed of three wireless sensor networks, all operating in the SubGHz bands, i.e. a Dash-7 network, a 6LoWPAN/Spirit network, and a LoRA network. Each sensor node measures a context-specific characteristic of the environment where it is placed, as better explained in the following. The sensor data are hence transferred (via the MQTT protocol) to the corresponding Adapter, where they are converted into semantic RDF triples. To this aim, the W3C Semantic Sensor Network ontology is used. The RDF triples are hence managed by the SEPA tool, which plays a twofold role: (*i*) it serves as a semantic broker between the connectivity and service layers, i.e. it allows the service application to be notified when any change is observed on the sensing data (details are provided below) and a (*ii*) i t can work as a semantic cache of the sensor data gathered by the network silos. At service layer, two front-end applications have been implemented so far: i.e. (*i*) *Sensor Dashboard*, a Web application providing a graphical visualization of the sensor data (for each silo) in both the spatial and temporal domain; and (*ii*) *Sensor Speak*, an Amazon Alexa application, allowing to read the sensor values and to tune the current network configuration through a vocal interface. The actual implementation of SensorSpeak does not involve the utilization of the SEPA, i.e. the application is retrieving data directly from the ThingSpeak cloud (https://thingspeak.com) (dotted line in Fig. 2); we are currently implementing the missing link. Similarly, no back-end services are currently available, but we plan to integrate them as future work. In the following Sections, we provide details for each of the component mentioned so far. Scalability issues are discussed in Section IV-D.

TABLE I. DAS7 WSAN NETWORK (433 MHZ). AVG. CURRENT VALUES FOR INDOOR SMART-HOME APPLICATIONS

Node type	#Nodes	Communication Mode	TR	Avg. current Beacon Mode	Avg. current R/R Mode
Temperature sensor	3	beacon	10 s	14 μA	NA
Water flooding sensor	1	beacon	10 s	14 μA	NA
Actuator (brushed DC motor)	2	request/ response	$\geq$ 5 s	NA	2mA (R/R phase) 35 μA (stand-by)



(a) (b) (c)

Fig. 3. The 6LoWPAN installation is depicted in Figure 3(a). Average network Goodput and Delay for different hop length and traffic loads are reported in Figure 3(b) and Figure 3(c), respectively

A. Wireless Sensor Platforms

1) Dash-7 Network: The prototype wireless sensor and actuator network (WSAN) based on DASH7 [6] protocol performs two tasks representative of indoor smart-home applications, i.e controlling radiator head distribution and water flooding sensing [41][42]. The network architecture has a star structure, where all the nodes communicate with the DASH7 gateway. The nodes are composed of a low-power microcontroller (STM32L1), a sub-GHz radio for data communication (SPIRIT1) and sensor and/or actuator devices. A real-time Operating System runs on STM32L1 microcontroller and manages the DASH7 protocol stack, the radio activity and the sensors and actuators operations. The smart heating control system is composed of three temperature-sensing nodes and one actuating node that drives a brushed DC motor connected to a radiator valve. In the second application, sensing is done measuring impedance changes between two electrodes in order to detect the presence of water whereas one actuating node drives a brushed DC motor connected to the water supply system to cut off water flux. DASH7 communication protocol is a low-power open source WSAN protocol which operates in the 433 MHz (the one chosen in the prototype WSAN network), 868 MHz and 915 MHz unlicensed ISM band. It can achieve a data rate of 200 kbps and an outdoor range up to 2 km. It supports two communication models that can coexist in the same network and in the same node: a pull model for a query method (request-response) and a push model for data transfer initiated from the nodes (beaconing). In request-response mode the communication between the nodes and the gateway is bidirectional but the nodes can send data only after a gateway request. In this communication mode, the node periodically wakes-up from a sleep state to an active one to perform RX scan phases needed for network synchronization (stand-by phase). In the proposed implementation, a latency equal to 5 s has been chosen as reasonable both for the smart heating application and for the water flooding control. This results in an average current equal to 35 μA. An average current of 2 mA is absorbed during a 5 s data request-response (R/R) phase. In beaconing mode the node periodically initiates a communication sending a message to the gateway with a programmable frequency while for the rest of the time it remains in a sleep state. This involves lower average current consumption than the request-response method, but the communication is unidirectional which is not suitable for applications that need data or commands sent on request. On the contrary, this communication model is necessary for sending alarm (e.g. flooding sensor nodes) or periodic alive messages.

A summary of the DASH7-based network is given in Table I. In Table, TR is the period between two successive data/command transmissions.

2) 6LoWPAN Network: 6LoWPAN (IPv6 over Low-Power Wireless Personal Area Networks) [7] [43] is a recent set of IETF standards enabling IPv6 networking over low-power devices with limited processing capabilities. This is achieved by means of lossy compression mechanisms reducing the IPv6 packet header, in order to meet the maximum frame size imposed by some MAC/PHY wireless technologies (e.g. the IEEE 802.15.4). Among others, the 6LoWPAN standard defines a default multi-hop network architecture, composed of hosts, i.e. edge sensing devices, relays, i.e. forwarding nodes, and edge routers that connects the 6LoWPAN network to the Internet. The 6LoWPAN network currently implemented is composed of five edge/relay nodes and one edge router, all located within the first floor of the ARCES building in Via Pepoli (Bologna). Each host device is constituted by a Nucleo ST32F401RE board, equipped with a SPIRIT X-NUCLEO-IDS01A4 radio transceiver operating on the 868 MhZ ISM

TABLE II. LoRaWAN WSN Network (868 MHz). Avg. Current Values for Outdoor applications

Node type	#Nodes	Communication Mode.	TR	Avg. current
Soil moisture sensor	5	Beacon (class A type)	30 min	10.5 μA

band, and some environmental sensors (temperature, humidity, etc); the edge router is also equipped with a Wi-Fi module, through which it can connect to the Internet. Each device is located in a different room, and connects to the router through a direct or a multi-hop connection, as shown in Fig. 3(a); the RPL protocol is used as routing scheme, in mesh-under mode [44].

Fig. 4. LoRa device used to measure the soil moisture

Fig. 3(b) and 3(c) depict some performance metrics of the installed 6LoWPAN network. More specifically, Figure 3(b) shows the network goodput, when varying the packet load produced by an edge device; each packet has a constant size of 65 bytes. Different curves are drawn based on the hop distance between the sender device and the edge router. As expected, the throughput increases with the network load till the saturation point is achieved; however, the hop count has a considerable impact on the system performance. Similar behaviours can be observed on the delay curves in Fig. 3(c).

3) LoRa Network: An outdoor prototype wireless sensor network (WSN) based on LoRaWAN protocol [8] has been designed to measure the soil moisture. The nodes are composed of STM32-Nucleo board (based on low-power microcontroller STM32L0) equipped with a shield with LoRa radio (SX1272) and a soil moisture sensor based on impedance measurement. The radio operates in the 868 MHz unlicensed ISM band. The network is composed of five battery-powered nodes and data transfer is initiated from the nodes (LoRaWAN class A type nodes). The gateway is placed inside a building and the nodes are placed outdoor in a range of about 20-30 meters. A summary of the LoRa-based network is given in Table II. TR = 30 min is the period between two successive data acquisition. Fig. 4 depicts one of the LoRa device used to measure the soil moisture.

B. The SPARQL Event Processing Architecture (SEPA)

The interoperability layer has been developed on top of the SPARQL Event Processing Architecture (SEPA) (https://github.com/arces-wot/SEPA, http://mml.arces.unibo.it/TR/sepa.html) [18]. SEPA is the result of researches in smart spaces [20] and Internet of Things technologies [19] and it has been designed to support the development of interoperable, context aware, secure and distributed Web based applications and services. In particular, as depicted in Figure 5, SEPA is built on top of Linked Data [45] technologies (https://www.w3.org/DesignIssues/LinkedData.html) and standards like URI (https://tools.ietf.org/html/rfc3986), HTTP (https://tools.ietf.org/html/rfc2616), RDF (https://www.w3.org/TR/rdf11-concepts/), SPARQL (https://www.w3.org/TR/sparql11-overview/), JSON-LD (https://json-ld.org/spec/latest/json-ld/), Linked Data Platform (https://www.w3.org/TR/ldp/) and Linked Data Notifications (https://www.w3.org/TR/ldn/) and it extends the SPARQL 1.1 protocol (https://www.w3.org/TR/sparql11-protocol/) with the SPARQL 1.1 Secure Event (SE) protocol (http://mml.arces.unibo.it/TR/sparql11-se-protocol.html) offering a publish-subscribe mechanism to its clients. The SPARQL 1.1 SE protocol transparently conveys SPARQL 1.1 Query (https://www.w3.org/TR/sparql11-query/) and Update (https://www.w3.org/TR/sparql11-update/) so that a generic SPARQL protocol client can interact with a SEPA broker like with a common SPARQL protocol service (also knows as SPARQL endpoint). Furthermore, the SPARQL 1.1 SE protocol introduces the SPARQL 1.1 Subscribe as a language to describe subscription requests and notifications (http://mml.arces.unibo.it/TR/sparql11-subscribe.html). A subscription request acts as a persistent SPARQL 1.1 Query and the subscriber is notified of any changes in the query results due to any incoming update. SEPA also introduces an application design pattern (http://mml.arces.unibo.it/TR/jsap.html) where each client can assume one of the following roles: producer (i.e., it acts as a publisher), consumer (i.e., it acts as a subscriber) and aggregator (i.e., it publishes aggregated data driven by the received notifications). Eventually, SEPA is built around a security layer (e.g.,TLS https://tools.ietf.org/html/rfc5246) and it allows clients authentication by means of JSON Web Tokens (https://tools.ietf.org/html/rfc7519).

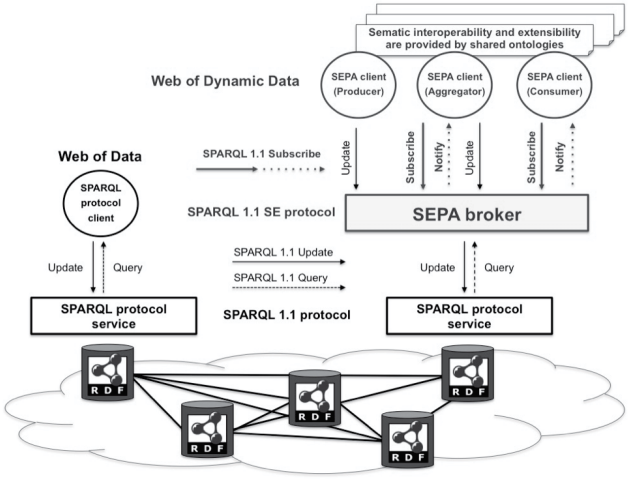


Fig. 5. SEPA enables the development of Dynamic Linked Data applications

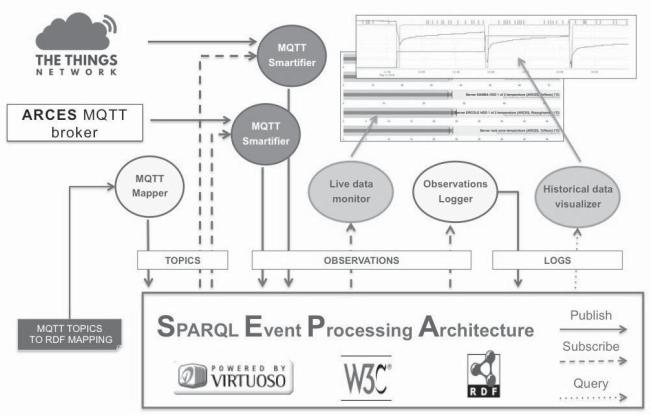


Fig. 6. The SEPA based sensor dashboard allows to visualize heterogeneous data in real time (live data monitor) and to plot the evolution over a time interval (historical data visualizer)

C. Smart Services

1) Sensor Dashboard: The sensor dashboard service provides two main functions (see Fig. 6): the real time mon-itoring of data and the plotting of data trends within a time interval. The proof-of-concept implementation allows to gather data from a generic MQTT broker. An MQTT broker can be interfaced with the SEPA broker through an adapter (see *MQTT Smartifier* in Figure 6). As an example two different MQTT brokers have been considered to validate the approach: the *ARCES MQTT broker* who provides the temperatures of a set of servers used by ARCES (e.g., CPU cores and HDD) and environmental sensors (e.g., temperature and hu-midity of the server rooms) and the *The Things Network* MQTT broker used to collect data from the LoRa networks. Data collected by the MQTT Smartifiers are published on a SEPA broker according to the W3C Semantic Sensor Network ontology (https://www.w3.org/TR/vocab-ssn/). The ontology has been extended by adding two properties: one to link an observation with the corresponding MQTT topic (e.g., arces-monitor:hasMqttTopic) and one to link an observation with its historical data (e.g., arces-monitor:refersTo). Timestamps are expressed according to the W3C Time ontology in OWL (https://www.w3.org/TR/owl-time/). Data are contextualized according to Schema.org. In particular, the observations are linked to their physical location (e.g., schema:Place). An overview of the adopted ontology is drawn in Fig. 7.

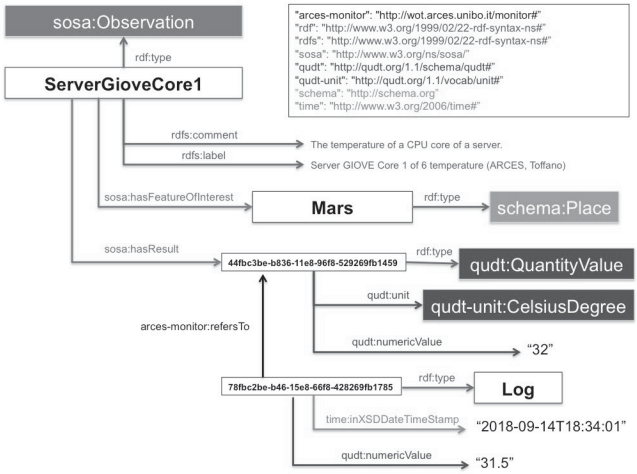


Fig. 7. The ontology of the IoT dashboard service

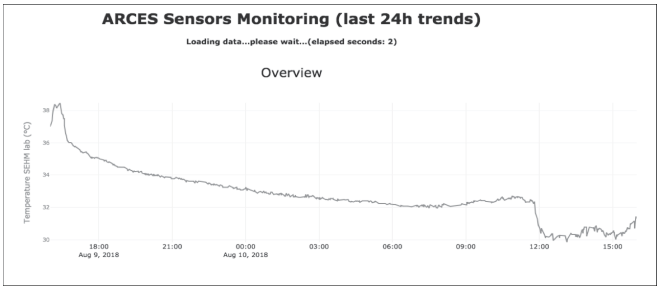


Fig. 8. Temperature with air conditioner off in a closed lab in summertime: somebody enters at around 11:30

The underneath Fig. 8 shows the temperature of a laboratory (located at the ARCES building, in Bologna) in a day in August. It clearly shows that the air conditioning system was off or down, and that the windows where closed during that day. It is also interesting to highlight that -through the sensor data monitoring- we can infer that somebody entered the laboratory at around 11:30 and opened a window, as the temperature quite rapidly dropped from 32 to 30 degrees (i.e,. the outdoor temperature). The temperature sensor acts as a presence sensor in this case, very relevant at summertime, when the lab is supposed to be closed.

Fig. 9 shows the temperature in two ARCES laboratories on July 30, when the outdoor temperature was approximately 32 C and the air conditioning system was down. The figure shows that, while the SEHM laboratory (above, brown profile) is kept closed, with all its windows closed, somebody entered in the ST office at around 8:00 am, and opened a window. Then the air conditioning system was repaired at 10:30 and it took approximately 3 hours for the temperature to stabilize at around 28C. The 44C peak in the SEHM lab is simply

due to the location of the sensor, i.e. the peak is due to the sun, filtering through the opening of two tends and hitting the sensor at mid afternoon.

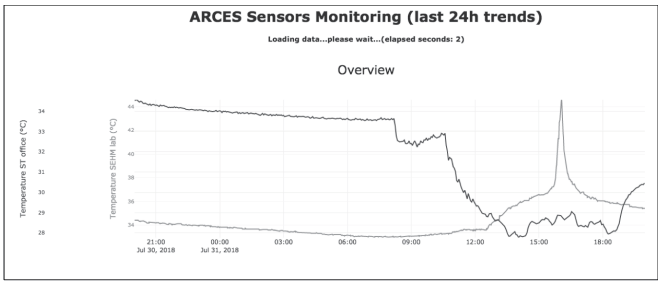


Fig. 9. Temperature in two labs of the same building in a summer day

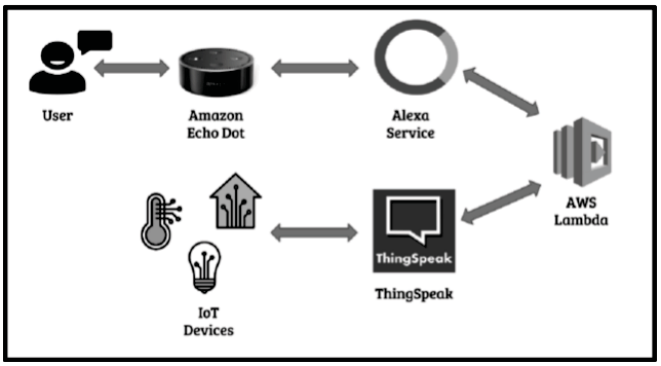


Fig. 10. Current implementation of the SensorSpeak front-end service

2) SensorSpeak service: Humans expect to access data in the most natural and unobtrusive way, and, in many circumstances, the human voice is the appropriate media for this User-Environment Interaction (UEI). Many voice based smart personal assistants already exist on the market. They leverage on speech recognition, voice synthesis and artificial Intelligence technologies and they are mostly provided by big players like Apple (e.g. Siri), Microsoft (e.g. Cortana) Google (e.g. Google Assistant), and Amazon (e.g. Alexa). In this research, Amazon Alexa was selected and integrated in the architecture of the voice service, depicted in Figure 10. This latter refines the general architecture of Fig. 2, however introducing a fundamental difference, since the SensorSpeak application has not been integrated yet with the SEPA platform, and hence with the interoperability layer. The integration is currenly under development. As a result, at present, ThingSpeak is the cloud storage platform where all data collected by the heterogeneous networks are conveyed through MQTT messages. SensorSpeak main contribution is to introduce the man in the loop via voice commands: users, through their interaction with Alexa, monitor and control all devices reflected in the ThingSpeak cloud. This interaction is achieved by implementing an Alexa skill (https://developer.amazon.com/docs/ask-overviews/build-skills-with-the-alexa-skills-kit.html) which maps commands invoked by the users to specific requests to ThingSpeak. Such skill is implemented in the blocks Alexa Service and AWS Lambda, and it consists essentially of the following two components:

1) A *Voice interaction model*, defining the features offered by the skill (intents) and all the related sentences or keywords (utterances).
2) A *Lambda Function*, which is the logic driving the Amazon Lambda Service which handles all user requests and interacts with the cloud using a RESTful API.

With this set up, numerous functions are performed through voice commands, such as adding or removing devices from the cloud, listing all devices that meet a particular property, getting a specific measurement or a measurement aggregated by multiple sensors, turning on and off devices and setting their reconfigurable parameters.

As shown in the demo (https://www.youtube.com/watch?v=EvYwDVXVWI8) (a screenshot is reported in Fig. 11), the proposed solution allows the interaction with individual sensors (e.g. asking for the temperature in a specific lab), as well as the simultaneous interaction with many of them (e.g. turning off all devices in the cloud). This feature allows Sensor-Speak to be used in many contexts where sensor networks are deployed, both indoor and outdoor. With SensorSpeak there are no restrictions on the types of devices included in the framework: the only requirement is that the device must be connected to the ThingSpeak cloud through an active Internet connection. The procedure to add a new device starts with the voice command *Add new device*: Alexa reacts asking for various parameters regarding the new sensor to be connected to the cloud. At the end of the dialogue, a virtual entity is created in the cloud with an associated ID. Such ID is then used by the real device to properly connect to its own virtual entity: we remark that this is the only information required to integrate the sensor in the voice system. From the users point of view, it is not practical to identify the devices by means of a numeric ID consisting of several digits. For this reason, it is easier and more intuitive to associate names to the various devices. Such name is defined by the user during the process of adding the new sensor and must be unique within the cloud to avoid any ambiguity. The mapping between name and ID of the device is performed by several RESTful API provided by ThingSpeak and it is hidden to the user. Comparing Fig. 2 and 10, the voice service does not share its data model with the SEPA that supports the cooperative automation services enabled by the proposed macroarchitecture. Consequently, once a new sensor is added to the edge side of the ecosystem, this needs to be made visible independently to all data models of the interoperability layer shown in Figure 2. Furthermore, the proposed voice service is able to interact with the private clouds and stores shown in Figure 10, but it is not able to interact in SPARQL with the semantic end point, or, in other words, it does not support speech2sparql transformations yet. By providing this capability, a major and ultimate step towards the data-service convergence will be taken: seamless interaction both with the environment and with context dependent semantic services shaped according to unanticipated patterns will be enabled from a single voice port.

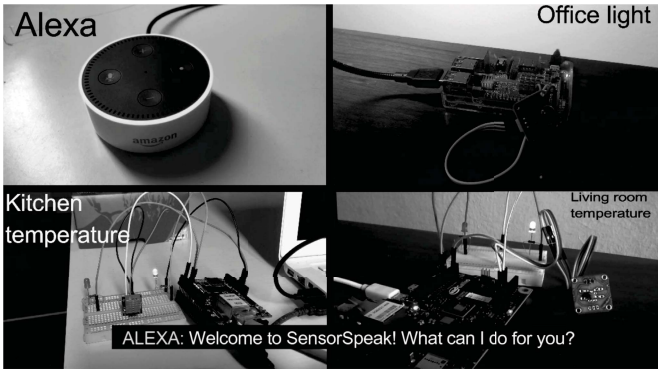


Fig. 11. SensorSpeak service in action

D. Scalability Issues

The need to scale up is a major requirement of the IoT domain, which relies on software infrastructures that will be asked to manage a huge number of data sources and to transport a massive quantity of data and process it efficiently. To ensure that the interoperability solution proposed in this paper will not be limited by scalability issues and could be effectively adopted in real applications belonging to different vertical domains, the next step of our research will consist in the integration of an industrial platform within our platform. In this respect, an interesting option is the Eclipse Kapua IoT platform (https://www.eclipse.org/kapua/index.php). This platform provides all services required for the management of IoT gateways and devices in the field, including configuration management, application life-cycle management and remote access. It also connects the data collected by field-deployed devices to enterprise applications and analytics, leveraging reliable and open protocols. With respect to the architecture depicted in Fig. 2, the platform could potentially act both (*i*) as a standard IoT cloud platform, providing an industrial-grade solution for data collection, storage and first level analysis, and (*ii*) as an integration platform that complements SEPA capabilities, providing advanced management features for the entire IoT ecosystem (remote control, device management, diagnostics, lifecycle support, etc.). This cloud platform focuses also on the inclusion of the enterprise level, providing integration flows that connect premises application, services, processes and data. These services allow to see the IoT infrastructure as any other enterprise application: the network of distributed IoT devices and sensors is one extremity of the integration flow, while the enterprise application is the other extremity. In this way the general requirements of a cloud computing platform, i.e. the ability to collect, control, transport and analyze data coming from the field, together with the functionalities required to deploy and maintain IoT solutions, will be achieved.

V. Conclusions and Future Works

In this paper, we addressed data-reuse and interoperability issues as the key for overcoming the current IoT fragmentation and for opening the way to new IoT applications and market opportunities. We presented an abstract reference application, which decouples the sensor data producers from the data consumers, thanks to the introduction of a semantic end-point. Then, we discussed its implementation within a a framework -currently installed at the ARCES centre of the University of Bologna, Italy- resulting from the interplay of several research projects and from the activity of several research teams. More specifically, the research team in ARCES (leaded by prof. Tullio Salmon Cinotti) is active on fields on data interoperability and semantic architecture for the IoT, and contributed to the development of the SEPA tool and of the dashboard. The research team in DISI (leaded by prof. Marco Di Felice and Luciano Bononi) is working on wireless sensor networks and mobile application, and contributed to the deployment and evaluation of the 6LowPAN network and of the SensorSpeak application. The research team at the DEI@UNIBO department (team leaded by prof. Eleonora Franchi Scarselli) collaborates with the STMicroelectronics on sensor design, and, in this paper, worked on the LoRa network deployment and analysis. Finally, the Eurotech group provided valuable insights on how to extend the current framework with an industrial IoT vision. We remark that the framework is a research result, and a habitat for new studies at the same time. With respect to the Fig. 2, there are still several challenges that need to be faced in near future works, both in terms of novel software contributions, integration with industrial IoT platforms, and performance analysis. First of all, as already mentioned in the previous Section, the integration of SensorSpeak with the SEPA has to be completed. Second, we plan to devise context-aware, back-end services providing semantic reasoning mechanisms, in order to infer new knowledge about the environment from the triples available in the SEPA, and to decide the proper actions to perform according to users' defined policies. Regarding the integration with industrial IoT platforms, the adoption of Eclipse Kapua will provide a solid baseline to develop an industrial-grade interoperability solution, capable to ensure scalability, efficiency, reliability, security and enterprise level seamless integration. Regarding the analysis, an extensive performance evaluation of the proposed system must be conducted, in order to identify possible bottlenecks, quantify the overhead introduced by the SEPA, and test the scalability when increasing the number of devices connected, and the applications supported.

Acknowledgment

Part of this project was funded by the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737434 (CONNECT). This Joint Undertaking receives support from the European Unions Horizon 2020 research and innovation programme and Germany, Slovakia, Netherlands, Spain, Italy. Moreover, part of the project was supported by the AlmaIdea Senior Project "BeeDrones: Environmental monitoring systems based on ultra low-power sensors and unmanned aerial vehicles". Finally, the authors would like to thank Dr. Giovanni Modica and Dr. Giovanni Pisana for their valuable contribution in the design and implementation of the 6LoWPAN sensor network platform.

References

[1] C. Perera, A. Zaslavsky, P. Christen and D. Georgakopoulos Context Aware Computing for The Internet of Things: A Survey *IEEE Communications Surveys & Tutorials*, 16(1), pp. 414-454, 2014

[2] FLEXIENCY H2020 Project. http://www.flexiciency-h2020.eu

[3] P. Corista, D. Ferreira, J. Giao, J. Sarraipa and R. J. Gonalves. An IoT Agriculture System Using FIWARE. *Proc. of IEEE ICE/ITMC*, Stuttgart, Germany, 2018.

[4] A. Akbar, F. Carrez, K. Moessner, J. Sancho and J. Rico. Context-aware stream processing for distributed IoT applications. *Proc. of IEEE WF-IoT*, Milan, Italy, 2015.

[5] McKInsey Global Institute. The Internet of Things: Mapping the value beyond the hype. Executive Summary. 2015.

[6] DASH7 Alliance Protocol Specification v1.1. http://www.dash7-alliance.org/product/d7ap1-1/.

[7] IETF RFC 4919. IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals. https://tools.ietf.org/html/rfc4919.

[8] LoRa Alliance. https://lora-alliance.org/

[9] C. Bizer, T. Heath, and T. Berners-Lee. Linked Data—The Story So Far. *Int. J. Semant. Web Inf. Syst. 5*, 1–22, 2009..

[10] J. Umbrich, B. Villazön-Terrazas and M. Hausenblas. Dataset Dynamics Compendium: A Comparative Study. *Proc. of the First International Conference on Consuming Linked Data*, Shanghai, China, 2010.

[11] Capadisli, S.; Guy, A.; Lange, C.; Auer, S.; Sambra, A.; Berners-Lee, T. Linked Data Notifications: A Resource-Centric Communication Protocol. In ESWC 2017 - The Semantic Web, Switzerland, 2017.

[12] Bhide, M.; Deolasee, P.; Katkar, A.; Panchbudhe, A.; Ramamritham, K.; Shenoy, P. Adaptive push-pull: Disseminating dynamic Web data. *IEEE Trans. Comput.*, *51*, 652–668, 2002.

[13] Abdullah, H.; Rinne, M.; Törmä, S.; Nuutila, E. Efficient Matching of SPARQL Subscriptions Using Rete. In Prof. of the 27th Annual ACM Symposium on Applied Computing, Trento, Italy, 26–30 March 2012.

[14] Rinne, M.; Abdullah, H.; Törmä, S.; Nuutila, E. Processing Heterogeneous RDF Events with Standing SPARQL Update Rules. In On the Move to Meaningful Internet Systems: OTM 2012.

[15] Murth, M.; Kühn, E. A heuristics framework for semantic subscription processing. In *The Semantic Web: Research and Applications*, 2009, pp. 96–110.

[16] Frommhold, M.; Arndt, N.; Tramp, S.; Petersen, N. Publish and Subscribe for RDF in Enterprise Value Networks. In Proc. of the Workshop on Linked Data on the Web, Montreal, Canada, 11–15 April 2016.

[17] Viola, F.; D'Elia, A.; Roffia, L.; Salmon Cinotti, T. A modular lightweight implementation of the Smart-M3 semantic information broker. In Proc. of the 2016 18th Conference of Open Innovations Association and Seminar on Information Security and Protection of Information Technology (FRUCT-ISPIT), St. Petersburg, Russia, 18–22 April 2016.

[18] Roffia, L.; Azzoni, P.; Aguzzi, C.; Viola, F.; Antoniazzi, F.; Salmon Cinotti, T. Dynamic Linked Data: A SPARQL Event Processing Architecture. Future Internet 2018, 10, 36.

[19] L. Roffia et al., "A Semantic Publish-Subscribe Architecture for the Internet of Things," in IEEE Internet of Things Journal, vol. 3, no. 6, pp. 1274-1296, Dec. 2016.

[20] Morandi, F.; Roffia, L.; DElia, A.; Vergari, F.; Salmon Cinotti, T. RedSib: A Smart-M3 semantic information broker implementation. In Proc. of the 12th FRUCT Conference, Oulu, Finland, 59 November 2012; pp. 8698.

[21] Le-phuoc, D.; Parreira, J.X.; Hauswirth, M. Linked Stream Data Processing. In Proc. of the Reasoning Web. Semantic Technologies for Advanced Query Answering: 8th International Summer School 2012, Vienna, Austria, 3–8 September 2012, pp. 245–289.

[22] Della Valle, E.; Ceri, S.; Harmelen, F.V.; Fensel, D. It's a Streaming World! Reasoning upon Rapidly Changing Information. *IEEE Intell. Syst.*, *24*, 83–89, 2009.

[23] Eugster, P.T.; Felber, P.A.; Guerraoui, R.; Kermarrec, A.M. The many faces of publish/subscribe. *ACM Comput. Surv.*, *35*, 114–131, 2003.

[24] Dell'Aglio, D.; Le Phuoc, D.; Le-Tuan, A.; Ali, M.; Calbimonte, J.P. On a Web of data streams. In Proc. of the ISWC2017 workshop on Decentralizing the Semantic Web, Vienna, Austria, 21–22 October 2017.

[25] Skovronski, J. An Ontology-Based Publish-Subscribe Framework. Master's Thesis, State University of New York at Binghamton, Vestal, NY, USA, 2006.

[26] Groppe, S.; Groppe, J.; Kukulenz, D.; Linnemann, V. A SPARQL Engine for Streaming RDF Data. In Proc. of the 2007 3rd International IEEE Conference on Signal-Image Technologies and Internet-Based System, Shanghai, China, 16–18 December 2007; pp. 167–174.

[27] Pellegrino, L.; Baude, F.; Alshabani, I. Towards a scalable cloud- based RDF storage offering a pub/sub query service. In Proc. of the 3rd International Conference on Cloud Computing and GRIDs Virtualization, Nice, France, 22–27 July 2012.

[28] Pellegrino, L.; Huet, F.; Baude, F.; Alshabani, A. A Distributed Publish/Subscribe System for RDF Data. In Data Management in Cloud, Grid and P2P Systems, 2013; pp. 39–50.

[29] Murth, M. A Semantic Event Notification Service for Knowledge-Driven Coordination. In Proc. of the1st Int'l. workshop on emergent semantics and cooperation in open systems (ESTEEM), cooperation with the 2nd Int'l. Conf. on Distributed Event-Based Systems (DEBS 2008), Rome, Italy, 1 July 2008.

[30] Murth, M.; Kühn, E. Knowledge-based coordination with a reliable semantic subscription mechanism. In Proc. of the 2009 ACM Symposium on Applied Computing, Honolulu, HI, USA, 8–12 March 2009.

[31] Murth, M.; Kühn, E. Knowledge-based interaction patterns for semantic spaces. In Proc. of the 4th International Conference on Complex, Intelligent and Software Intensive Systems, Krakow, Poland, 15–18 February 2010; pp. 1036–1043.

[32] Honkola, J.; Laine, H.; Brown, R.; Tyrkko, O. Smart-M3 information sharing platform. In Proc. of the IEEE symposium on Computers and Communications, Riccione, Italy, 22–25 June 2010; pp. 1041–1046.

[33] Suomalainen, J.; Hyttinen, P.; Tarvainen, P. Secure Information Sharing Between Heterogeneous Embedded Devices. In Proc. of the Fourth European Conference on Software Architecture, Copenhagen, Denmark, 23–26 August 2010.

[34] Galov, I.V.; Lomov, A.A.; Korzun, D.G. Design of semantic information broker for localized computing environments in the internet of things. In Proc. of the 2015 17th Conference of Open Innovations Association (FRUCT), Yaroslavl, Russia, 20–24 April 2015; pp. 36–43.

[35] Barbieri, D.F.; Braga, D.; Ceri, S.; Grossniklaus, M. An Execution Environment for C-SPARQL Queries. In Proc. of the 13th International Conference on Extending Database Technology, Lausanne, Switzerland, 22–26 March 2010.

[36] Calbimonte, J.P.; Corcho, O.; Gray, A.J.G. Enabling Ontology-Based Access to Streaming Data Sources. In The Semantic Web—ISWC 2010; pp. 96–111.

[37] Anicic, D.; Fodor, P.; Rudolph, S.; Stojanovic, N. EP-SPARQL: A Unified Language for Event Processing and Stream Reasoning. In Proc. of the 20th International Conference on World Wide Web, Hyderabad, India, 28 March–1April 2011.

[38] Le-Phuoc, D.; Dao-Tran, M.; Xavier Parreira, J.; Hauswirth, M. A Native and Adaptive Approach for Unified Processing of Linked Streams and Linked Data. In The Semantic Web—ISWC 2011, pp. 370–388.

[39] Komazec, S.; Cerri, D.; Fensel, D. Sparkwave: Continuous Schema-enhanced Pattern Matching over RDF Data Streams. In Proc. of the 6th ACM International Conference on Distributed Event-Based Systems, Berlin, Germany, 16–20 July 2012.

[40] Passant, A.; Mendes, P.N. SparqlPuSH: Proactive Notification of Data Updates in RDF Stores Using PubSubHubbub. In Proc. of the Sixth Workshop on Scripting and Development for the Semantic Web (ESWC 2010), Crete, Greece, 31 May 2010.

[41] A. D'Elia, L. Perilli, F. Viola, L. Roffia, F. Antoniazzi, R. Canegallo, T. Salmon Cinotti. A self-powered WSAN for energy efficient heat distribution. *Proc. of IEEE SAS*, Catania, Italy, 2016. (ESWC 2010), Crete, Greece, 31 May2010; Volume 699. Applications Symposium (SAS), Catania, 2016, pp. 1-6.

[42] L. Perilli et al. Wake-up radio impact in self-sustainability of sebsir and actuator wireless nodes in smart-home applications. to appear in *Proc. of IGSC*, Pittsburgh, USA, 2018.

[43] C. Bormann and Z. Shelby. 6LoWPAN: The Wireless Embedded Internet. Wiley Press. 2009.

[44] IETF RFC 6550. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. https://tools.ietf.org/html/rfc6550.

[45] Bizer, C.; Heath, T.; Berners-Lee, T. Linked DataThe Story So Far. Int. J. Semant. Web Inf. Syst. 2009, 5, 122.