International Journal of Control and Automation
Vol. 11, No. 7 (2018), pp.1-10
http//dx.doi.org/10.14257/ijca.2018.11.7.01

Design and Implementation of IoT Object Virtualization for Physical Devices in Smart Home

Lei Hang[1], Muhammad Sohail Khan[2] and Do-Hyeun Kim[3*]

[1,3]Computer Engineering Department, Jeju National University, South Korea
[2]Department of Computer Software Engineering, University of Engineering&Technology, Pakistan
[1]hanglei112233@hotmail.com, [2]sohail.khan@uetpeshawar.edu.pk, [3]kimdh@jejunu.ac.kr

Abstract

Recently, IoT (Internet of Things) refers to the networking of everyday objects, which are often equipped with ubiquitous intelligence. IoT will increase the ubiquity of the Internet by integrating all objects for interaction via embedded systems, leading to a highly distributed network of devices communicating with human beings as well as other devices. In recent years, the development of the IoT application is a complex task requiring a wide range of expertise. And this is an important issue for inexperienced developers, especially non-programmable skills, in order to quickly build IoT prototypes. To solve this problem, this paper proposes improved IoT object virtualization scheme. The proposed method uses the existed concept of virtual objects to allow end-users to create virtual objects through simple actions such as drag-and-drop and clicks, and so on. We implement the prototype of IoT object virtualization based on IETF CoAP protocol using Intel Edison boards with different sensors and actuators. The proposed IoT object virtualization scheme support to connect a virtual objects with visual components and physical IoT devices. Accordingly, we access physical IoT devices using virtual objects.

Keywords: *Internet of things, CoAP protocol, Virtual objects, Service composition, Smart home*

1. Introduction

The Internet of Things (IOT) is becoming an increasingly important subject of conversation both on and off the workplace [1]. IoT promises to enable building novel applications in areas such as building and home automation, smart environment, agriculture, intelligent transportation, and healthcare. IoT is the basic concept of basically enabling a global connectivity of any devices with an on and off switch to the Internet between the real world and a virtual world. These IoT devices detect and interact with their environment to take and give operation commands. These devices allow a growing number of applications to provide people with digital aids for their daily activities. In this ever-changing landscape of IoT, recent researches imply that the number of intelligent connected objects will increase dramatically over the next few years. With this flourishing of smart devices, the challenge arises, how to put devices together to make IoT applications? This challenging issue manifests that most IoT infrastructures lack of interoperability in system integration such as data transmission, device management, and service composition and application deployment.

With the rapid increase in the dynamic market of the IoT, it is inefficient for developers to meet all these requirements because new products and techniques arise all the time. The development of communication protocols and IoT technologies strongly

Received (January 4, 2018), Review Result (March 8, 2018), Accepted (March 12, 2018)

ISSN: 2005-4297 IJCA

favors and a growing number of APIs and data models are issued for a specific domain and product. IoT mashup is a new solution to facilitate the development of IoT applications using the usual web development tools and technologies. Recently, the representative architecture of state transfer (REST) appeared, leading to the development of the Web of Things [2]. URIs are used to identify Web things and the HTTP protocol is used for stateless reciprocation between clients and servers. The use of Web-based protocols allows mashups to be created that allow developers to combine data from physical data sources and virtual sources on the Web [3].

Many upmarket IoT platforms or toolkits have been developed, platform such as GeoThings and Fitbit providing Application Programming Interfaces (APIs) based on RESTful for easy device and app integration. openHAB is an open source platform designed for smart home environment which provides Integrated Development Environments (IDEs), proprietary APIs, Software Development Kits (SDKs) and script engines.

However, with the development of web front-end technologies such as JavaScripts and HTML5, these IoT platforms moving in a Web-centric direction with REST APIs and Web-based dashboards helping users to quickly set up and monitor the platform. Platforms such as Kaa, Predix or Carriots providing dashboards for data management of connected objects, SDKs for integration between device and server, web-based APIs for integration with product-specific services. However, the point is that all these solutions are perfectly valid for their specific application domains but limit to user customized domains. These platforms provide their own programming environment therefore users must be able to know programming languages such as Java or C to construct their applications.

In this paper, we propose an enhanced IoT object virtualization scheme for users who have no programming experience to design and develop their own IoT prototypes. In different systems, these virtual objects may be represented using different encoding technologies but nevertheless, the main objective of the virtual objects remains the same. The client application development with application logic at the client side is again a developer's job and people with no programming skills may not be able to create their own custom applications. At the end such application development is time consuming and the resulting applications cannot be modified easily.

The rest of the paper is organized as follows, Section 2 presents related work. Section 3 illustrates the proposed IoT object virtualization. In Section 4, we describe the implementation of IoT object virtualization. Towards the end, we conclude this paper.

2. Related Work

IoT's vision is the realization of a global network of connected and interoperable intelligent devices that provides various services to people. Although individual communication and device technologies have improved considerably, the implementation and implementation of such a complex application is still at an early stage. A mashup tool is a solution that allows software developers to combine data flows and then data viewing via the internet or a mobile application using REST APIs and a Web description language such as WSDL.

At the same time, the user interface is capable of producing complex applications by integrating the user interface into component-oriented style and oriented service [6]. Examples well known for such tools are Mnubo [7], Swarm [8] and Axeda [9]. These commercial companies provide solutions to developers to create and deploy powerful IoT products to bring these innovations to market. Glue.things [10] is a recent mashup platform for wiring web peripherals and web services. It implements the concepts of real-time peripheral integration and communication using recent technologies such as Web Sockets, MQTT and CoAP [11].

These protocols are applied on real-time data stream networks to perform data streams. The Glue.things also supports the development process with features for device and data management, aggregation, and distributed deployment. The mashup interface is based on the Node-RED application, including triggers, actions, and authentication. Glue.things provides a RESTful API for developers to deploy their mashups and a browser-based dashboard for end users to handle all mashup applications.

IoT-MAP [12], a mobile application platform aiding flexible interoperability between mobile devices and surrounding smart things. IoT Map decouples the development of mobile application from the static model chosen by the designer/developer of the application which is a great hurdle in the way of application adaptability to the user's needs and the surrounding environment. IoT-MAP utilizes the concept of abstracted service objects for the development of IoT applications. This is achieved through a set of APIs exposed for functions like discovery and retrieval of the service objects etc. The business logic is written in POJO while abstracting out the details of connectivity and implementation of smart things. Some other platforms focus on the perspective of reducing the difficulty of IoT application development.

IoTLink [13] is a typical example based on a model-driven approach that allows inexperienced developers to compose mashup applications using a domain-specific graphic language. Through visual components, IoTLink encapsulates the complexity of communicating with devices and services over the Internet and summarizes them as virtual objects that can be easily configured and wired together and can be accessed using different communication technologies to create an IoT application. Therefore, it resolves interoperability between heterogeneous IoT components. The project utilized IBM post-study system usability questionnaire to get user feedback regarding the system usability. Although these projects consider the utilization of existing web development technologies, it does not consider the users' abilities to using these technologies. These systems themselves are composed of various open source components which makes the employment of the system complex, even for skilled users. Glue.things utilizes Node-RED application as the mashup interface. Node-RED [14] is a powerful and promising tool for IoT mashups but it still requires some level of programming skills as the composed services must be utilized in a web application. Users also need to know basic programming structures to define condition statements, triggers and actions for these mashups of data streams.

3. Proposed IoT Object Virtualization

Figure 1 illustrates the concept of configuring the proposed architecture. The physical domain consists of various IoT devices, and each device can use one or more sensors. The elements of the virtual domain are abstract models of physical devices. Each device in the physical domain corresponds to a clone element, noted as a virtual device, in the virtual domain in a unique way. When the given situation of the device is changed, a sensor acquires located data and transfers them as events to the corresponding clone of the virtual domain, respectively. The proposed IoT object virtualization system consists of virtual and physical domain. In the virtual domain, multiple clones, if necessary that contain newly updated events from a physical domain can be composed as an associated aggregation. The associated aggregation also includes simulated elements, if the opportunity arises. The associated aggregation then derives a state from the device or domain with a set of events.

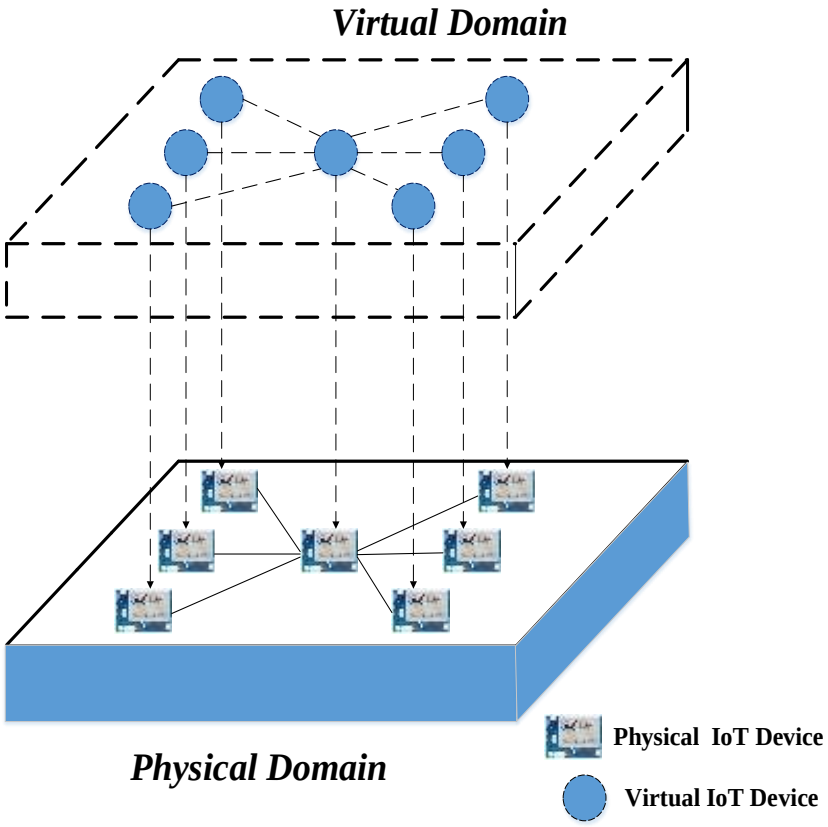

Figure.1. IoT Object Virtualization Concept

The proposed Virtual Object Manager represents the physical things in the form of VO Behavior, VO attributes and VO visualization. The VO Behavior is the services or functions which can be utilized by the system to interact with the physical thing represented by the VO. A simple example would be the name of CoAP service which can be called remotely to interact with the physical thing. The VO Attributes are the other information in the form of complete URI and Location etc. which collectively describes the existence of the physical thing through its virtual representation. Finally, VO Visualization is the graphical representation of VO depicting the type of the physical thing represented through it. It is an icon or string of characters visually representing the underlying physical entity such as thermometer icon to represent a temperature sensor.

The VO information acquisition interface is used to register physical things as virtual objects. This interface can be a local data entry interface for a user or administrator to register the available physical things or it can be exposed as an online service to enable users to remotely access and register their devices. After the registration, the information related to VOs is stored at the VO Repository. VO Repository holds the information about virtual objects in XML document for so that it can be transferred easily over the Internet.

The Virtual Object Manager is in collaboration with other classes such as the File Manager, Communication Manager and Parser etc., provides the implementation of all the functionality associated with the VONL. The main interface provides a view for all the existing VOs so it requests the File Manager through the VOM to read the VO data from the XML repository. The data is parsed by the XML parser and the virtual object information is provided to the VOM in order to display it through the interface. Now the user is set to interact with the VOs through the interface. The VO related interactions that user can perform have been shown in the sequence model. The user selects a VO graphical representation and the VO information is displayed to the user through the view interface. The user can then choose to Edit and Update the VO if needed be. To save an

edited VO, the information from the view interface is sent to the parser to convert it into proper format and the File Manager writes it to the XML Repository. The Delete Operation also works in the same fashion. The Communication Manager acts as a server thread which listens for incoming connections from the remote client (SCM). Once is connection request is received, the Communication Manager requests the VOM for VO information which is sent to the client.

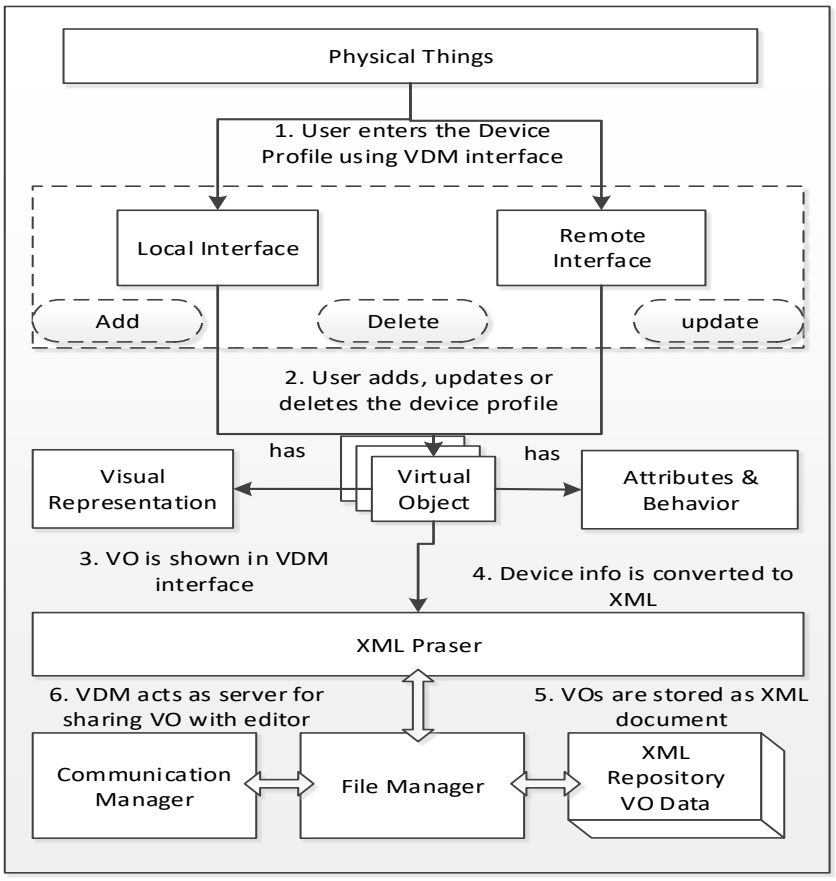

Figure 2. Virtual Device Manager Operation Configuration

4. Implementation of IoT Virtual Object

Figure 3 shows the screen shot of the Virtual Device Manager module at the Virtual Object Network Layer. This module helps the users to encapsulate the attributes of their IoT resources. It provides an intuitive way to bind the IoT resources with a virtual representation so that inexperienced users are able to interact and manipulate. User manually provides the details with respect to their IoT resources in the form of URI, location, type and properties. Any device which supports the proposed protocols can be added as a resource to the system via this approach. URI specifies the protocol and address through which the device can be uniquely identified. Device type tag specifies the type of the specific IoT devices. Location tag is used for specifying the location of the remote IoT resource. Properties tag specifies the available services which can be executed via the specific resource.

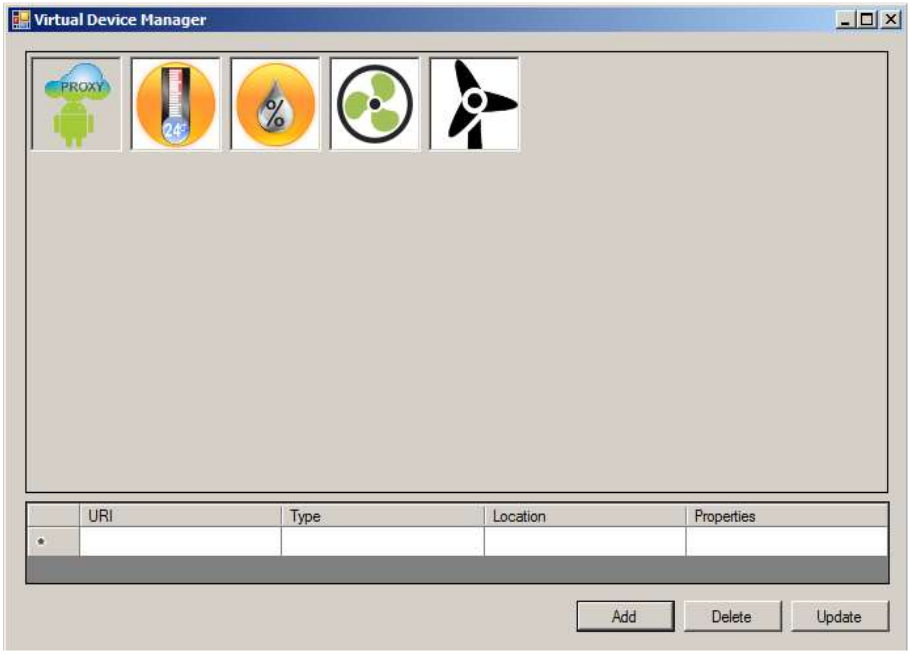

Figure 3. Virtual Object Manager Main Interface

Figure 4 represents the screen shot for creating virtual device in the Virtual Device Manager. Users need to specify the device URI, device type, location and properties. After that, users can save the input information and the Virtual Device Manager will automatically generate the virtual device information using XML representation as shown in Figure 5.

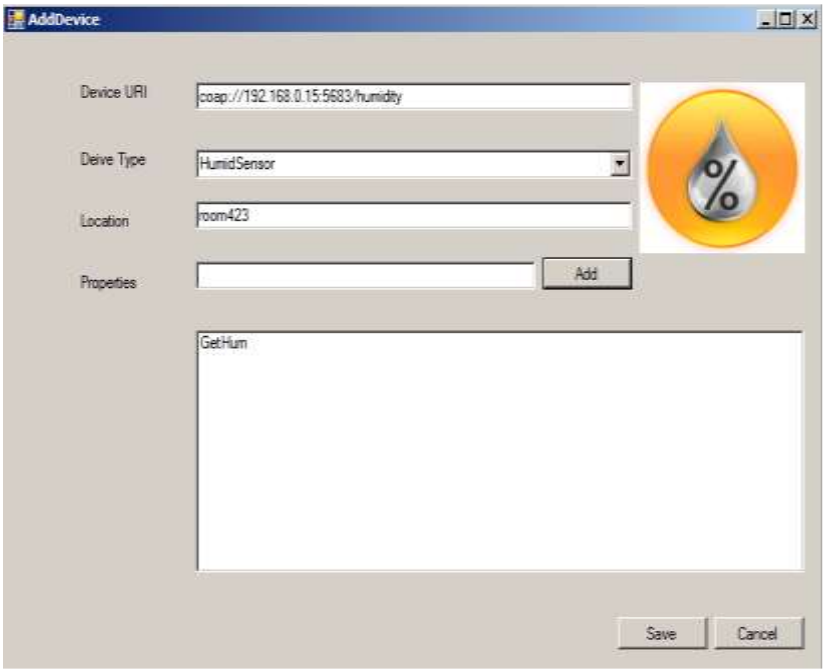

Figure 4. Device Interface for Creating Virtual Objects

Figure 5 presents the XML representation for virtual objects in the form of device nodes. Four nodes have been expanded in the figure which shows the stored information for a temperature sensor, humidity sensor, wind sensor and a fan device. The URI

specifies the protocol and address through which the device can be uniquely identified. The Properties tag specifies the available services which can be executed via the specific resource. A resource can also have more than one property (executable functions) which is specified by the sub-tags <P> in the Properties tag. Both URI and an instance of the Properties tag can be utilized to provide a uniquely addressable function of the remote resource. The Location tag is used for specifying the location of the remote IoT resource. This tag and more information regarding the owner or allowed users etc. can be considered for future studies related to the security of the system.

```
<?xml version="1.0" encoding="UTF-8"?>
- <Devices>
  - <Device>
      <Uri>coap://192.168.0.14:5683/temperature</Uri>
      <Type>TempSensor</Type>
      <Location>room423</Location>
    - <Properties>
        <P>GetTemp</P>
      </Properties>
    </Device>
  - <Device>
      <Uri>coap://192.168.0.15:5683/humidity</Uri>
      <Type>HumidSensor</Type>
      <Location>room423</Location>
    - <Properties>
        <P>GetHum</P>
      </Properties>
    </Device>
  - <Device>
      <Uri>coap://192.168.0.10:5683/fan</Uri>
      <Type>Fan</Type>
      <Location>room423</Location>
    - <Properties>
        <P>SetFan</P>
      </Properties>
    </Device>
  - <Device>
      <Uri>coap://192.168.0.11:5683/windspeed</Uri>
      <Type>WindSensor</Type>
      <Location>room423</Location>
    - <Properties>
        <P>GetWind</P>
      </Properties>
    </Device>
```

Figure. 5. XML Representation of Created Virtual Devices

Table 1 illustrates device in the smart indoor space which are temperature sensor module, humidity sensor module, wind sensor module, fan module and proxy module. TinkerKit T000200 thermistor for temperature sensor module is specifically designed to be used with the TinkerKit development toolkit for Arduino. The Thermistor is a resistor whose resistance varies significantly (more than in standard resistors) with temperature. This module's output approaches 5v as the temperature increases. As the temperature decreases, it approaches 0V. The module has been utilized to implement Intel Edison based CoAP services for providing temperature values in Centigrade as well as Fahrenheit scales. Wind sensor module is used to measure the wind speed. The SEN1070 is a three-cup anemometer that monitors wind speed for the range of 0 to 30m/s. The wind sensor module has been utilized to implement Intel Edison based CoAP services for providing the air speed values by level. The HIH-4030/4031 series humidity sensors are designed specifically for high volume OEM (Original Equipment Manufacturer) users. The direct input of a controller or other device is made possible by the near linear voltage output of this sensor. With a typical output of only 200 μA, the HIH-4030/4031 series is often ideal for low-flow and battery-powered systems. The Moisture Sensor Module was used to implement Intel Edison-based CoAP services to measure moisture values. Arduino L9110 for fan module is used as an IoT actuator in the proposed prototype. L9110 drive which can control the positive & negative turning with mounting holes, high quality and high

efficiency. The fan actuator module has been utilized to implement Intel Edison based CoAP services for setting fan coil speed. The IoT proxy establishes the connection between the virtual domain and physical domain. The IoT proxy module has been utilized to implement Intel Edison based CoAP services for transferring sensing data from physical domain to virtual domain and providing prediction service to maintain indoor comfort index.

Table 1. Table Label

Devices	Temperature Sensor	Wind Sensor	Humidity Sensor	Fan	Proxy
Visual Representation					
Servo Model	TinkerKit T000200 thermistor module	Wind Speed Sensor (SKU:SEN0170)	Humidity Sensor (HIH-4030)	Arduino L9110 Fan Module	None
CoAP Server	Intel Edison with libcoap	Intel Edison with libcoap	Intel Edison with libcoap	Intel Edison with libcoap	Intel Edison with CoAP
CoAP Services	GetTempC, GetTempF	GetAir	GetHumidity	SetFan	GetPMV, MLR, Fuzzy

Figure 6. Intel Edison Based IoT Prototype for Smart Home

Figure 6 presents finalized form of the IoT prototype. This prototype has been developed as a miniature representation of a smart space scenario where multiple sensing devices are deployed to capture the contextual data and an actuating device is deployed to modify the surroundings in the indoor environment. The prototype consists of the

following sensing and actuating devices which are used by users to customize the behavior of the smart space based on the contextual situations.

5. Conclusion

In this paper, we propose an IoT object virtualization architecture, which is methodically applicable to the design of intelligent home service system. The proposed architecture can meet methodological requirements such as combination, adoption, integration, delivery, etc. The architecture consists of two domains: physical, virtual. The paper applied the architecture and methodology for designing a smart space system for home virtualization and functional collaboration, intending to form a relative comfort climate in the indoor environment in low-energy construction, saves energy. This paper has only proposed the device virtualization in smart home and it requires service composition based on these virtualized devices. In the future, we plan to design the service composition approach to generate IoT service process. The scalability of our approach has not yet been tested. We plan to test it in a large-scale and real-world environment using a wireless sensor test bench.

Acknowledgement

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(2014-1-00743) supervised by the IITP(Institute for Information & communications Technology Promotion), and this This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2017-0-00756, Development of interoperability and management technology of IoT system with heterogeneous ID mechanism). Any correspondence related to this paper should be addressed to DoHyeun Kim; kimdh@jejunu.ac.kr. This paper is a revised and expanded version of a paper entitled "A Physical Device Virtualization for Service Composition in IoT Environment" presented at CRTT 2017 [15].

References

[1] Gartner, "Predicts 2015: The Internet of Things", **(2014)** December.
[2] L. Richardson and S. Ruby, "RESTful Web Services", O'Reilly Media, Inc., **(2007)**.
[3] D. Zhiquan, Y. Nan, C. Bo and C. Junliang, "Data Mashup in the Internet of Things", Proc. International Conference on Computer Science and Network Technology (ICCSNT), **(2011)** December, pp. 948-952.
[4] W. S., "Business process modeling improves administrative control", Automation, **(1967)**, pp. 44-50.
[5] ASHRAE. 2010. ANSI/ASHRAE Standard 55-2010. Atlanta: American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc.
[6] S. Pietschmann, M. Voigt, A. Rumpel and K. Meißner, "CRUISe: Composition of Rich User Interface Services", Proceedings of the 9th International Conference on Web Engineering (ICWE 2009), Springer, **(2009)** June, pp. 473-476.
[7] "Munbo", http://mnubo.com/.
[8] "Swarm", http://buglabs.net/products/swarm.
[9] "Axeda", http://www.ptc.com/axeda.
[10] R. Kleinfeld, S. Steglich, L. Radziwonowicz and C. Doukas, "Glue.Things: A Mashup Platform for wiring the Internet of Things with the Internet of Services", Proceedings of the 5th International Workshop on Web of Things, **(2014)**, pp. 16-21.
[11] M. Kovatsch, "CoAP for the Web of Things: From Tiny Resource-constrained Devices to the Web Browser", 4th International Workshop on the Web of Things (WoT 2013), **(2013)**.
[12] S. Heo, S. Woo, J. Lm and D. Kim, "IoT-MAP: IoT mashup application platform for the flexible IoT ecosystem", Internet of Things (IOT), 2015 5th International Conference on the Internet of Things (IOT), **(2015)**, pp. 163-170.
[13] F. Pramudianto, C. A. Kamienski, E. Souto, F. Borelli and L. L. Gomes, "IoT Link: An Internet of Things Prototyping Toolkit", 11th International Conference on Ubiquitous Intelligence & Computing, **(2014)**, pp. 1-9.

[14] Roberto, "Introduction to Node RED | Sensetecnic," 2015.[Online].Available:http://developers.sensetecnic.com/article/introduction-to-node-red/. [Accessed: 19-Apr-2016].
[15] L. Hang and D. H. Kim, "A Physical Device Virtualization for Service Composition in IoT Environment", The 1st International Conference on Convergence Research Theory and Technology CRTT 2017, **(2017)**, pp. 25-26.

Authors

Lei Hang, he received the B.S. degree in computer engineering in 2015 and the M.S. degree from the Jejunu National University, Korea, in 2017. He is currently a Ph.D student at Jeju National University, South Korea. His area of interest is sensor networks, M2M/IOT, intelligent service, and mobile computing.

Do-Hyeun Kim, he received the B.S. degree in electronics engineering from the Kyungpook National University, Korea, in 1988, and the M.S. and Ph.D. degrees in information telecommunication the Kyungpook National University, Korea, in 1990 and 2000, respectively. He joined the Agency of Defense Development (ADD), from Match 1990 to April 1995. Since 2004, he has been with the Jeju National University, Korea, where he is currently a Professor of Department of Computer Engineering. From 2008 to 2009, he has been at the Queensland University of Technology, Australia, as a visiting researcher. His research interests include sensor networks, M2M/IOT, energy optimization and prediction, intelligent service, and mobile computing.

