HAL
open science

IDE-OM2M: A framework for the development of IoT applications using the OM2M platform

Karima Khadir, Thierry Monteil, Samir Medjiah

To cite this version:

Karima Khadir, Thierry Monteil, Samir Medjiah. IDE-OM2M: A framework for the development of IoT applications using the OM2M platform. 19th International Conference on Internet Computing and Internet of Things (ICOMP' 18), Jul 2018, Las Vegas, NV, United States. pp.76-82. hal-01874998

HAL Id: hal-01874998

https://laas.hal.science/hal-01874998

Submitted on 11 Jul 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

IDE-OM2M: A framework for the development of IoT applications using the OM2M platform

Karima Khadir, Thierry Monteil and Samir Medjiah

LAAS-CNRS, Universit de Toulouse, CNRS, INSA, UPS, Toulouse, France.
Email: {kkhadir, medjiah, monteil}@laas.fr

***Abstract*— The Internet of Things (IoT) has emerged strongly in recent years thanks to the proliferation of wireless communication devices and networks. It has spawned an explosion of new uses and services such as supply chain management, property and person monitoring, residential or business home automation, etc. However, several challenges must be overcome in order to develop IoT applications such as interoperability and interaction with and between different connected objects. In this paper, we propose a high-level Integrated Development Environment (IDE) that provides end-to-end solution for writing and deploying IoT applications. This IDE uses the Node-RED graphical environment to interact with the OM2M standard which serves as a middle-ware between our Node-RED module and the various heterogeneous devices. It also offers semantic support for an intuitive interaction with these devices.**

***Index Terms*—IoT, Node-RED, OM2M, oneM2M standard, semantic.**

INTRODUCTION

The Internet of Things is a concept newly emerged in the jargon of new technologies. It touches almost all the domains in a very sensitive way making possible the interconnection of multiple heterogeneous objects (sensors, actuators, smartphones, etc). The frequent and intensive use of these connected objects in daily life gave rise to the emergence of several tools and languages for the development of IoT applications. However, several challenges such as: heterogeneity, collecting information from sensors, extracting useful data and selecting devices participating in the application to be developed must be overcome before implementing an end-to-end system for the realization of innovative IoT applications both intuitively and flexibly.

In this article, we will cite some tools and methodologies for the development of IoT applications, ranging from methods based on traditional programming languages as JAVA, which require a lot of technical knowledge and arriving at Mashup tools such as: COMPOSE, WoTKit, ThingWorx and Node-RED, which allow a visual and interactive modeling of data flows exchanged between connected objects. Next, we describe in detail our system, which aims to offer an Integrated Development Environment for the development of IoT applications through a visual approach hiding all technical difficulties involved in the implementation of this type of applications. Our system is characterized by an innovative end-to-end architecture based on our OM2M platform implementing the global standards SmartM2M and oneM2M to address the problem of heterogeneity, discovery approaches based on the abstract names of sensors and actuators, Labels describing these devices and even an approach based on semantics. The end of this architecture on the user side is determined by a set of Node-RED nodes allowing the user to create his applications easily by dragging and dropping these nodes. Then, we compare our solution with the aforementioned tools to extract the strong points that characterize it. Finally, to demonstrate the value of our system and evaluate the benefits of using the various provided nodes, we propose a scenario for creating IoT applications easily within the ADREAM intelligent building.

I. STATE OF THE ART

There are several works in the literature that are focused on the problem of developing IoT applications, we present in this section some tools and specific languages.

A. Eclipse IoT

Eclipse IoT [1] is a set of frameworks and open source technologies as well as a wide range of APIs (RESTful Application Programming Interface) to enable Java developers to implement IoT solutions easily. Among them we find the Open IoT Stack for JAVA that gathers JAVA components and OSGI (Open Services Gateway initiative) services. However, the user must overcome some standards such as MQTT and CoAP as well as technical knowledge allowing the interaction with the devices, the discovery of the services they expose, etc.

B. PyoT

PyoT [2] is an IOT application programming tool that can manipulate the various sensors and actuators represented as virtual CoAP resources and exposed via RESTful interfaces. It is based on the Python language.
The users can create their applications using a set of predefined functions grouped in the Python API to retrieve the list of resources (get_actuator_list()), to retrieve data from a sensor (get_setpoint()), or to send commands to the actuators (set_actuator()). A wide range of IoT applications can be created by using these functions combined with notions of traditional programming such as loops and conditions.

C. Reactive Blocks

Reactive Blocks [3] is a tool based on the model-driven development model dedicated to the development of event and simultaneous applications. It uses both JAVA programming and UML modeling where the user can program his own blocks or use pre-existing blocks and then interconnect them.
UML's role is to coordinate the behavior of blocks in the form of activity diagrams or state machines. For JAVA, it is used to implement the operations performed by the block. Once the application consisting of several blocks is built, it will be analyzed automatically and finally generate code in the form of OSGI packets or JAVA applications.

D. IoT-MAP

IoT-MAP [4] is an IoT Mashup Application Platform that enables mobile devices to interact with intelligent objects in a flexible and abstract way, and which facilitates the implementation of mobile IoT applications by offering a set of intuitive APIs. These APIs contain functions for devices discovery, retrieval of services offered by them and their abstract use as a real objects.
The manufacturers of connected objects use name servers based on the ONS (Object Name Service) and the bundle of drivers bundles to facilitate the discovery of devices by end users. These latter can then choose which drivers will participate in the desired service or create it using a tool provided by the platform. Finally, they build the desired application by gathering the services chosen and the logical modules allowing their interoperability. The platform made available to users of the generalized functional abstraction interfaces via the IoT-App API for the development of their applications using the Plain Old JAVA Object (POJO) style.

E. BETaaS

BETaaS [3] is a platform dedicated to the execution of distributed applications that runs on gateways to which objects are connected. This platform exposes the contents of connected objects as an interface called Things-as-a-Service (TaaS) independently of the technologies on which the sensors/actuators are based or their locations. Therefore, it reduces the execution time and complexity of code development of IoT applications that can also be reused.
It is a modular platform based on layered architecture to enable the integration of existing M2M technologies by using plugins that provide a common interface for access to the functionality offered by M2M systems. It also gives developers the ability to develop new services and applications and deploy them on the service layer.

F. ThingWorx

ThingWorx [5] [6] is a complete platform for the design and execution of end-to-end IoT applications. Thanks to ThingWorx [3] Composer, it allows to model the components required by the application to be developed, such as devices and business logic. It also provides developers with a drag-and-drop tool called Mashup Builder to create IoT applications using platform data via a user-friendly interface and supports multiple protocols, cloud services, and social services.

G. COMPOSE

COMPOSE [7] is a platform as a service that provides a complete infrastructure for developing end-to-end IoT applications that intelligently detect or interact with physical devices and external information resources. It uses Node.js to create the flows describing these applications via Node-RED.
It also provides a set of tools (SDK, IDE, etc.) that allow the user to discover the existing services in the platform, compose them to create new applications and store them in the COMPOSE infrastructure. This platform integrates Web 2.0 based communication technologies and M2M protocols such as CoAP and MQTT.
Compose implements a data storage and exposure framework called servIoTicy. The latter makes it possible to interact with the user applications and the connected objects to store their data. The end users can retrieve this data and be notified in the event that new measurements are produced.
It offers a web interface available at http://www.gluethings.com that allow to create a virtual intelligent object on the servIoTicy platform, to test the deployment of the applications and to receive their data in real time. COMPOSE uses the IServe service warehouse to unify the publication of these services and their discovery. The latter functionality is achieved by using the IServe API and requests to provide a list of available services or software components to be used in the application to be developed.

H. WoTKit Processor

WoTKit Processor [8] is a data flow-based system implemented using JAVA. It provides a graphical tool to process the data from the different sensors and react in real time according to the received data.
It allows to create a set of interconnected modules called *pipes* managed by a secure management Web page that restrict access to the user concerned in the platform. Once the user executes his pipe, the system compiles the different modules to verify them, then instantiates them on the server and fills a routing table describing the existing interconnections within the pipe. Thereafter the input modules can connect to external systems such as WoTKit sensor aggregation platforms or subscribe to data streams to receive real-time notifications.

I. Node-RED

Node-RED [8] [1] is open source software on GitHub, under the Apache 2 license, developed by the IBM community. It is a tool for building IoT applications in a simple way using a visual drag-and-drop approach that allows developers to wire predefined or self-developed code blocks called *nodes* to create flows that perform a specific task (processing data, controlling devices, or even sending alerts, etc).
This graphical environment works as a Web server. It allows users to create and manipulate their IoT applications via

[1] https://nodered.org/

cabling of hardware devices, APIs, and Web services from a Web or Mobile browser, local or remote, as part of the Internet of Things. It is built on Node.js which makes it ideal for running on restricted objects with limited capacity like Raspberry Pi.
Node-RED provides a flow editor that consists of three panels basically:

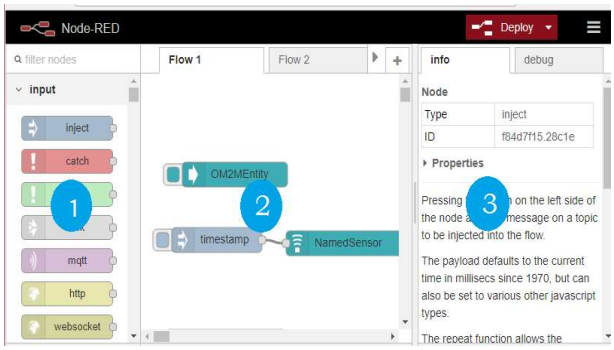

Fig. 1: Node-RED Stream Editor

1) Node Panel: contains a list of nodes that will be used by the user to create their feeds. Each node belongs to a specific category depending on the functionality it offers.
2) Flux panel: This is the main work area of the user where he can create his flows by dragging and dropping nodes from the first panel.
3) Information and debug panel: it consists of two tabs:
   a) An info tab: has the role to display the information related to a given node (its type, its properties, the features it provides, etc).
   b) A debug tab: allows users to view the results and outputs of the deployed flows and the errors encountered during execution.

II. IDE-OM2M

Our contribution is the development of a new Node-RED module that use the services offered by the OM2M platform to build useful IoT applications in a simple and fast way.
In this section, we begin by presenting the proposed end-to-end architecture, and then we explain the role of each part of this architecture by addressing the OM2M platform and the different nodes of our IDE-OM2M module.

A. Physical architecture

the figure 2 illustrates the physical architecture on which our IDE is based. This architecture includes a set of devices based on heterogeneous technologies from different brands, linked to several gateways that are connected and registered with a server that gives access to the system. IDE-OM2M interacts with this server to retrieve the data generated by the sensors to obtain the various measurements produced by the sensors such as, for example, the temperature of a room or/and act on the actuators for effecting Changes to the environment by sending specific commands to the relevant actuators according to the specific conditions in the context of IoT applications.

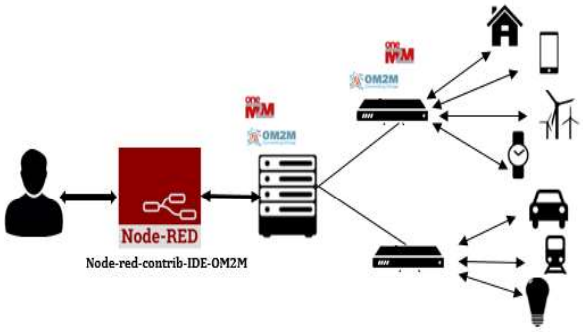

Fig. 2: Physical architecture of IDE-OM2M

1) OM2M: Eclipse OM2M [2] [9] is an open source project developed by by ourselves in LAAS-CNRS and distributed by the Eclipse foundation. It has created a standardized horizontal M2M services platform that implements ETSI SmartM2M and oneM2M standards.
OM2M aims to reduce the complexity of the development process of vertical M2M applications that can work on a wide range of devices, protocols and networks to facilitate their deployment. To do this, a *horizontal* service layer (Common Service Entity (CSE)) is introduced between the network layer and the application layer. This layer can be deployed on the network server, gateways and devices to enable the M2M interoperability in terms of communication and data exchange between the various heterogeneous devices, which will allow a large-scale deployment of M2M.
This platform is composed of a set of JAVA services defined as an Eclipse product using Maven and Tycho running on an OSGi (Open Services Gateway Initiative) platform which makes it highly extensible via plugins. These services are implemented independently of the network and the underlying hardware layer and are exposed through a RESTful API in the form of identifiable resources and accessible in a unique way via URIs (Uniform Resource Identify) specific [10].

These resources can be manipulated via HTTP queries (GET, POST, etc.) which can be very useful for providing IoT applications easily.
Several types of resources exist, of which we quote:
1- The AE resource: which is used to represent an application of an IoT device registered with OM2M in order to be able to manipulate it, for example a temperature sensor.
2- The Container resource: this is a data buffer. It must be created under a given AE resource, two types of Container resources are possible:

- A Descriptive Container: to contain Content Instances that describe the application concerned.
- A Data Container: to contain Content Instances containing the data of the application.

3- The Content Instance resource: represents an instance of data in the Container resource. These instances of data will be

[2] http://www.eclipse.org/om2m/

retrieved by our system to extract the value of certain attributes such as the state of a given device (off/on) or the level of luminosity in a room for example.
4- The Subscription resource: to allow subscribers to receive notifications during a new event on the resource to which they subscribed.

Thanks to the OM2M platform, communication interoperability has been achieved, allowing automatic discovery of connected devices, notification of subscribers to the arrival of new events and efficient routing involving various protocols and network technologies. However, applications did not have the ability to understand the meaning of content exchanged between devices without human intervention. For this, an expressive ontology for IoT called *IoT-O* (IoT Ontology) [11] is proposed taking full advantage of ontologies already defined in other domains such as sensors, observation, service, type of quantity, units, or time.
To integrate semantics into OM2M operations, two options are available:

- An *"ontologyRef"* attribute: which contains a unique reference that describes the semantic meaning of the resource. It can be used to find a specific resource based on a semantic concept. But this option is very limited because it does not provide any semantic meda-data on the relationships that the concept possesses.
- A new *"SemanticDescriptor"* sub-resource: which contains a complete semantic description of the resource defined using the RDF (Resource Description Framework) triplet. It is exposed and shared between the different applications.

2) IDE-OM2M: IDE-OM2M uses three approaches:

- The first uses the names of the applications offered by the devices as stored in the OM2M resources tree.
- The second solicits various actuators and sensors using a list of labels describing these objects, for example: Type/sensor, Category/temperature, Location/home, etc.
- The third one is a semantic approach that simplifies the designation of the devices using a language close to the natural human language via SPARQL (SPARQL Protocol and RDF Query Language) queries. This approach also avoids system reconfiguration problems when another that performs the same function replaces one device.

Our module includes a set of Node-RED nodes interacting with the resources mentioned above via REST calls. We shall cite in the following the list of nodes that compose it: - *OM2MEntity node*: allows the user to enter the parameters (URL, login, etc) describing the OM2M platform requested in a global way. The user can also use objects connected to different OM2M platforms using multiple OM2MEntity nodes.
- *Discovery node*: lists the devices connected to the OM2M platform designated by the user.
- *Sensor*: implements the methods used to retrieve the measurements produced by the various sensors recorded on an OM2M platform described by an OM2MEntity node (luminosity level and temperature). It uses the HTTP protocol to send its GET request as well as JSON (JavaScript Object Notation) as an exchange format between this module and the OM2M standard.
- *NotificationsHandler node*: implements an HTTP server that allows the user to receive notifications when a change occurs in an OM2M resource to which he subscribed via the *Subscription* node. It also enables him to retrieve the new state of this resource after the update.
- *Actuator node*: implements methods for sending simple commands to actuators to act on the state of a device (Off/On) or to modify the value of some measures of a given device, such as temperature.

III. Features and Benefits

Several works, which aimed to simplify the development of IoT applications have been proposed, but most of these solutions focus on particular problems (devices discovery, interoperability, etc) and require technical skills and knowledge in the field of IoT. Hence, the need for the implementation of an end-to-end, high-level solution that provides syntactic and semantic interoperability of communications with/among devices and that abstracts the various concepts required in the development of IoT applications.
The table I lists the different tools mentioned in this article by comparing them and our IDE-OM2M module according to some fundamental characteristics to meet the expectations of the developers of IoT applications such as: architecture, service discovery, semantics, etc. Through this table, it is clear that IDE-OM2M provides more advantage over other tools in terms of decentralization of its architecture, the discovery of appropriate services and the use of semantics.
The main features of IDE-OM2M are:

1 *Syntactic and semantic interoperability:* we used the OM2M standard as a syntactic and semantic middle-ware to deal with the problem of the horizontal fragmentation of the IoT market and allow users to interact with their devices with a language close to natural language.

2 *Distributed applications:* the devices included in the IoT application to be developed can be connected to different OM2M platforms (different servers).

3 *A drag-and-drop graphical environment:* thanks to Node-RED, we were able to set up a set of nodes that the user will need to develop their own applications easily. He only has to take these nodes, introduce the necessary information for their operation via simple and explicit forms and connect these nodes by arcs which serve as channels for the transport of data flows exchanged between them.

	Model	Type	Service discovery	Architecture	REST	Open source	Support for heterogeneity	Semantic
Java	Textual	-	-	-	-	-	-	-
PyoT	Textual	-	-	-	-	-	-	-
Reactive Blocks	Mashup	Server	No	Cloud service	No	Yes	Yes	No
IoT-MAP	Mashup	Server	Yes	Cloud-based	No	Yes	Yes	No
BETaaS	Mashup	TaaS	Yes	Cloud-based	No	Yes	Yes	No
ThingWorx	Mashup	PaaS	Yes	Cloud-based	Yes	No	Yes	No
COMPOSE	Mashup	PaaS	Yes	Cloud-based	Yes	Yes	Yes	No
WoTKit Processor	Mashup	PaaS	Yes	Cloud-based	Yes	No	Yes	No
Node-RED	Mashup	Server	No	Centralized	No	Yes	Yes	No
IDE-OM2M	Mashup	Server	Yes	Decentralized	Yes	Yes	Yes	Yes

TABLE I: Some available IoT tools

4 *Devices discovering:* to identify the sensors producing useful measurements that must be used by the IoT application to be developed and the actuators on which it is necessary to act to change the state of a device or a property of the environment, we have implemented nodes that allow the users to designate these devices. The users can do that either by using their exact names if they are known by them, by keywords such as the location and category that characterize them or by using a SPARQL request that allows an abstraction of high level for the discovery of these objects.

5 *Data processing:* after receiving the data instances produced by the sensors of the OM2M platform, the user can retrieve the value of a given property.

6 *Subscription to sensor data:* the user can subscribe to specific sensors to receive notifications in real time when new measurements are produced while retrieving the data. The user can trigger his applications either manually or on specific dates or at well-defined time intervals and thanks to this functionality, processing can be started as soon as new sensors data are received.

7 *Application of simple conditions:* the user can apply simple or complex conditions to the data received from the sensors in order to make intelligent decisions about his environment.

IV. Examples of application

In this section, we will begin by presenting the ADREAM building on which we will carry out our experiments. After wards, we will illustrate the functionalities offered by our module through real use cases.

A. Intelligent Building ADREAM

The ADREAM project (Reconfigurable Dynamic Architecture for Autonomous Mobile Embedded Systems) is an experimental research support funded by the European Union and the Midi-Pyrenees region.

It is inaugurated by the LAAS-CNRS in 2012. This program is devoted to energy optimization and ambient intelligence including various heterogeneous objects such as: vehicles, clothing, buildings, etc.
It includes an intelligent building capable of linking the virtual world of computing to the real world of physical objects. It is carried out by a multitude of sensors (temperature, presence, humidity, etc.) used to evaluate the condition of the building and the devices located there, and several actuators acting on the condition of the building according to the needs of the users. These devices are connected to two gateways, both are also connected to a central server allowing access to the system assembly.
To enable communication interoperability and integration of existing technologies such as ZigBee and 6lowpan, M2M gateways have a map module that translates them into a generic protocol independent of the transport protocol or network access.

B. Demonstration scenarios: Lights management

This scenario aims to optimize the energy consumption of the scaling system within the ADREAM building. It ensures a minimum brightness of 200 lux in case of presence of a person. Our system must benefit from daylight by opening the shutters. If the brightness outside is too low, the lights in the building should come on gradually according to their energy consumption (starting with the lamp that consumes less energy) until reaching the expected brightness level.
We give in the following a small abstract algorithm to facilitate the understanding of the scenario.

1) If a presence is detected then:
   a) Measure the brightness level inside the building via inLuminositySensor
   b) If the level of brightness is less than 200 lux then:
      i) Measure the external brightness via outLuminositySensor
      ii) if the external brightness is greater than 200 lux then: open the shutters via the shutter_actuator.
      iii) Else: as long as the brightness is less than 200 lux and there is a lamp off:

A) choose the lamp that consumes the least energy via a SPARQL discovery request.
B) Turn on the chosen lamp.

2) Else: turn off all lights and close the shutters of the building

The presence detection is performed by the notficationsHandler node which receives notifications directly from the presence sensor to which the node subscribes. While indoor and outdoor brightness is measured via LabeledSensor nodes via simple keywords.
Regarding the discovery of the lamps that consumes the least energy is achieved via the node SemanticSensor. The latter offers flexibility and ease to discover the devices with specific characteristics such as QoS attributes that can only be done with semantics.The figure 3 shows the node-red workflow for the chosen scenario

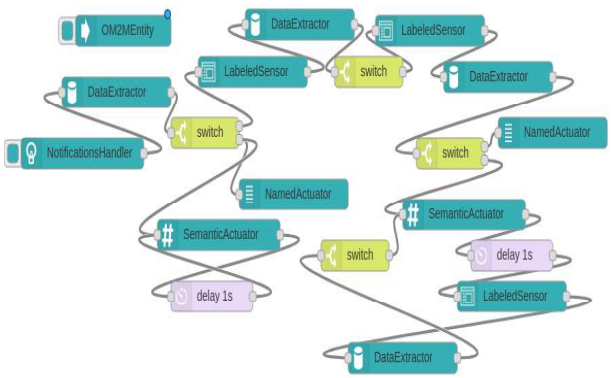

Fig. 3: Light management scenario

In the following, an illustration of the configuration of the most important nodes.
OM2MEntity: to name our platform by OM2M and specify the URL of the gateway to which the devices are connected as well as the login and the password.

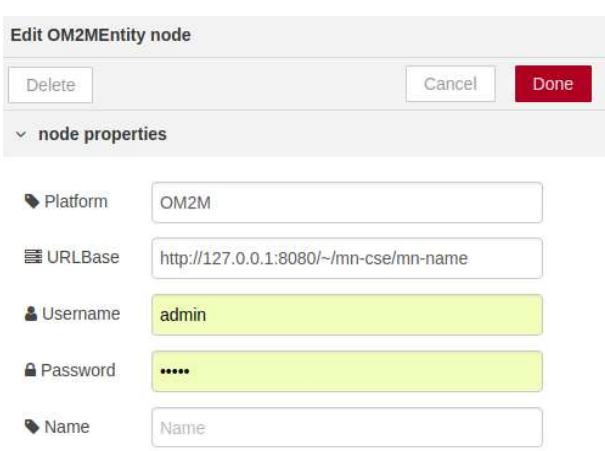

Fig. 4: OM2MEntity node configuration

LabeledSensor: to discover an appropriate sensor to measure the brightness inside the building via keywords. With regard to external brightness, just put the label Location/inADream.

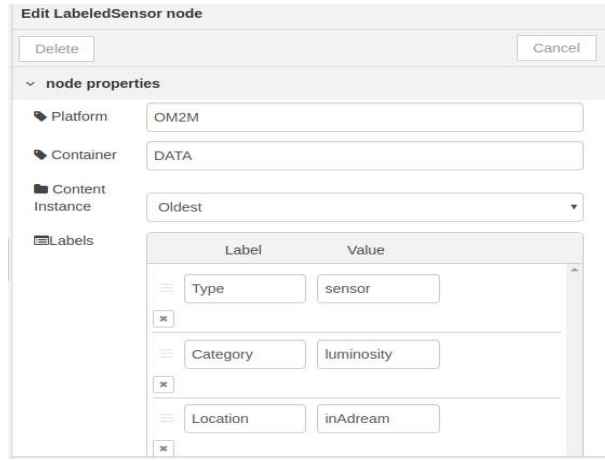

Fig. 5: LabeledSensor node configuration

SemanticActuator: to write a SPARQL query that searches for the lamp that consumes the least energy and is not lit to light it.

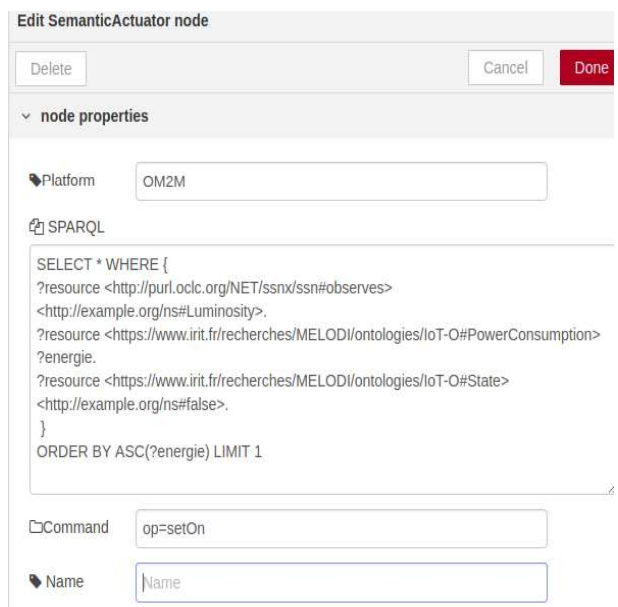

Fig. 6: semanticActuator node configuration

SemanticActuator: to turn off the lights on via a simple SPARQL query.

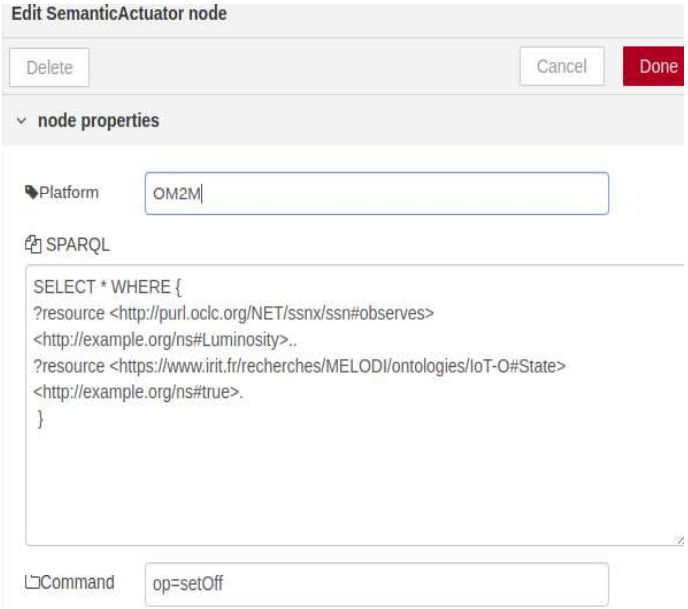

Fig. 7: semanticActuator node configuration

We see through this scenario the flexibility and simplicity of our IDE to set up this small application by connecting a set of nodes and configure them. it also shows the benefits of using semantics for resource discovery (here to find the lamp.

V. Conclusion

In this article we present a high-level environment using Node-RED which aims to provide the user with a graphical tool for the development of IoT applications easily. It operates the services offered by various IoT devices and exposed as RESTful resources in the OM2M platform to manipulate them via HTTP operations (GET, POST, etc). In addition, this environment integrates semantic support to provide an abstraction in the identification of sensors and actuators involved in the IoT applications to be developed.

In future work, we aim demonstrate more sophisticated scenarios with large set of sensors and actuators. IDE-OM2M will be published as a new open source IDE an eclipse OM2M and a Node-RED repository.

We will also develop a partnership to extend this IDE specifically with NCTU University.

References

[1] I. Eclipse. (2016) Open iot stack for java developers. https://iot.eclipse.org/java/ retrieved June 2017. [Online]. Available: https://iot.eclipse.org/java/

[2] A. Azzar and L. Mottola, "Virtual resources for the internet of things," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Dec 2015, pp. 245–250.

[3] E. Baccelli and D. Raggett, *The Internet of Things and The Web of Things*, ser. ERCIM News Special Issue on The Internet of Things and The Web of Things. ERCIM, Apr. 2015, no. 101. [Online]. Available: https://hal.inria.fr/hal-01244735

[4] S. Heo, S. Woo, J. Im, and D. Kim, "Iot-map: Iot mashup application platform for the flexible iot ecosystem," in *2015 5th International Conference on the Internet of Things (IOT)*, Oct 2015, pp. 163–170.

[5] Y. J. Heo, S. M. Oh, W. S. Chin, and J. W. Jang, "A lightweight platform implementation for internet of things," in *2015 3rd International Conference on Future Internet of Things and Cloud*, Aug 2015, pp. 526–531.

[6] thingworx. (2016) thingworx. https://www.thingworx.com/ retrieved June 2017. [Online]. Available: https://www.thingworx.com/

[7] C. Doukas and F. Antonelli, "A full end-to-end platform as a service for smart city applications," in *2014 IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2014, pp. 181–186.

[8] M. Blackstock and R. Lea, "Toward a distributed data flow platform for the web of things (distributed node-red)," in *Proceedings of the 5th International Workshop on Web of Things*, ser. WoT '14. New York, NY, USA: ACM, 2014, pp. 34–39. [Online]. Available: http://doi.acm.org/10.1145/2684432.2684439

[9] M. Ben-Alaya and al., "Om2m: Extensible etsi-compliant m2m service platform with self-configuration capability," *Procedia Computer Science*, vol. 32, pp. 1079–1086, 2014.

[10] Ptcek, Cackovic, and al., "Architecture and functionality in m2m standards," in *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2015, pp. 413–418.

[11] N. Seydoux, K. Drira, N. Hernandez, and T. Monteil, *IoT-O, a Core-Domain IoT Ontology to Represent Connected Devices Networks*. Cham: Springer International Publishing, 2016, pp. 561–576. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-49004-5_36