FIWARE: A WEB OF THINGS DEVELOPMENT PLATFORM

Ivan A. Tot[a], *Dušan* Lj. Bogićević[b], *Mladen* B. Trikoš[c], *Komlen* G. Lalović[d]

[a] University of Defence in Belgrade, Military Academy, Department for information systems and telecommunication engineering, Belgrade, Republic of Serbia,
e-mail: ivan.tot@va.mod.gov.rs,
ORCID iD: http://orcid.org/0000-0002-5862-9042

[b] Serbian Armed Forces, General Staff, Department for Telecommunication and Informatics (J-6), Command Information Systems and IT Support Centre, Belgrade +
University of Niš, Faculty of Electronic Engineering, Niš, Republic of Serbia,
e-mail: dusan.bogicevic@gmail.com,
ORCID iD: http://orcid.org/0000-0002-4300-2490

[c] University of Defence in Belgrade, Military Academy, Department for information systems and telecommunication engineering, Belgrade, Republic of Serbia,
e-mail: mladen.trikos@va.mod.gov.rs,
ORCID iD: http://orcid.org/0000-0002-5243-1326

[d] Faculty of Management, Economy and Finance, Belgrade, Republic of Serbia,
e-mail: komlen@mef.edu.rs,
ORCID iD: http://orcid.org/0000-0002-4590-2185

DOI: 10.5937/vojtehg66-17063; https://doi.org/10.5937/vojtehg66-17063

FIELD: Computer Sciences, IT
ARTICLE TYPE: Professional Paper
ARTICLE LANGUAGE: English

Abstract:

As an extension to the concept of the Internet of Things (IoT), Web of Things (WoT) represents a step towards connecting smart things to the existing web environment while considering issues such as heterogeneity, scalability, and usability. This paper is dedicated to current opportunities as well as challenges for development in the concept of WoT. The theoretical foundations of the Internet of Things concept, such as architecture, protocols, services, and things themselves, which form the basis of both concepts, are described in the paper. The paper deals with the necessary preconditions for developing the concept of Web of Things. The main contribution of the paper is a proposal of architecture based on the FIWARE platform as the basis for the development of Web of Things.The demonstration of the proposed architecture is described by a real case scenario.

Key words*: Internet of Things (IoT), Web of Things (WoT), FIWARE.*

ACKNOWLEDGMENT: The authors thank the Ministry of Defence of the Republic of Serbia through projects VA-TT/3/18-20 and VA-TT/1/17-19.

Introduction

Although the term Internet of Things (IoT) was quite accidentally conceived in 1999 as an idea that, at the time, the new technology Radio-Frequency IDentification (RFID) was presented as something new and needed, Kevin Ashton called his lecture Internet of Things, when even the Internet was also considered as new technology. The name he gave is considered to be the beginning of a new era in the Internet world. That era is characterized by communication between devices, that is machine to machine (M2M) communication (Internet of Things History).

With the development of computer networks, we have come to the era where the social network of devices is created. With the IoT concept, Things are able to use the Internet as a communication medium with services for exchanging data. When a large number of Things are connected, which will be able to be part of the World Wide Web (WWW), we come to the term Web of Things (WoT), which is the next step in the development of IoT, that is the successor to the IoT concept.

The development of IoT did not stop with M2M communication. It is still developing in different directions such as smart cities, agro-culture, animals, etc. Some of the research papers go a step further, where the big data and the web of things are connected, thus achieving the integration of humans, Things, and computers (Zhong et al, 2016).

IoT architecture

The IoT architecture is not based on one device. It is about sets of devices that collect information in different ways. When talking about IoT, the most considered topic are environments. The prefix “smart” is often found like e.g. smart homes, smart streets, smart parking lots, smart garbage cans, smart cities, etc. Smart environments can be defined as sets (federations) of sensors and actuators designed for house, building, city, transport etc. (Gubbi et al, 2013). Mark Weiser, who is considered as the founder of ubiquitous computing, has defined smart environments as a world of physical objects that are connected with sensors, actuators, displays, other environments over a network that allows interlaced connectivity (Perera et al, 2013).

From the highest level, IoT consists of three parts as shown in Figure 1 (Gubbi et al, 2013):

Part of devices: Devices or actuators with their communication components integrated with them.

Middleware part: The most complex part that implements data processing logic, stores data and provides access to data to users in

such a way that they do not care about the architecture of individual devices (actuators). This layer is made up of several parts (Object Abstraction, Service Management, Service Composition) (Botta et al, 2015).

Presentation part: This part adjusts to a specific application and performs data display, data management, etc.

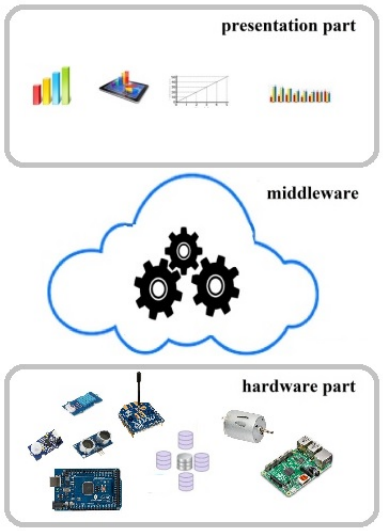

Figure 1 – IoT architecture
Рис. 1 – Архитектура интернета вещей
Слика 1 – Архитектура интернета ствари

The purpose of the device layer is to process collected mostly analogue data and to send it in the digital form over the network layer to the server layer.

The number of connected devices exceeds the number of human population. In 2010, the number of devices was almost twice the number of human population.

The architecture of the device (Things) layer should consist of three parts as shown in Figure 2 (Wortmann & Flüchter, 2015):

Middleware component, parts of software that will allow device management,

IoT component, which will collect data such as sensors (actuators), and the components through which communication with the server will occur (communication modules such as Ethernet, Wi-Fi, Bluetooth, ZigBee etc.),

Hardware part, on which the software will be executed and to which the sensors (actuators) will be connected.

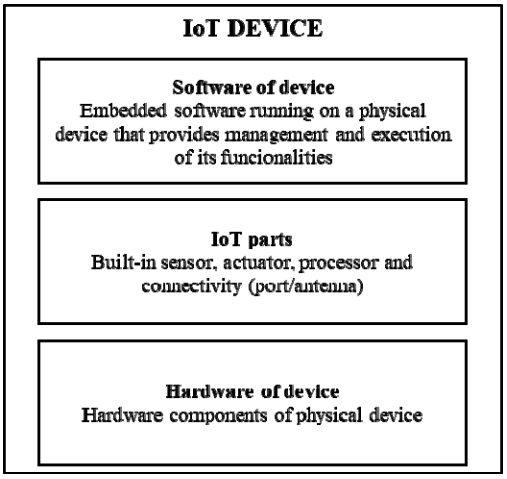

Figure 2 – Architecture of IoT device
Рис. 2 – Архитектура устройства интернета вещей
Слика 2 – Архитектура уређаја интернета ствари

As the IoT concept evolved, the protocols that were used for M2M communication were designed and improved. In addition to the Hypertext Transfer Protocol (HTTP) protocol that found application in the IoT concept, other protocols have been also designed, such as:

Extensible Messaging and Presence Protocol (XMPP):

The first version of the XMPP protocol appeared in 2000. This protocol uses a Transmission Control Protocol(TCP) protocol on the transport layer, and supports two message exchange models, both a request/response (synchronous) model and an advertiser/subscriber (asynchronous) model. The protocol is designed for short messages that are used in fast real-time applications that require small delay, or in presence-based applications. The disadvantage of this protocol is a relatively large overhead (Saint-Andre et al, 2009).

Advanced Message Queuing Protocol (AMQP Advanced Message Queuing Protocol Specification, Cisco Systems, 2018):

AMQP is an application protocol developed in 2003 by John O'Hara, for the needs of banks, and it was officially adopted in 2012 by OASIS (Organization for the Advancement of Structured Information Standards).

AMQP uses the TCP protocol on the transport layer, and has an overhead of 8 bytes. It is based on message exchange on the principle of

an advertiser/subscriber. This protocol achieves great reliability and ensures that the message is delivered even when the network breaks down. There are three mechanisms that can be used to send messages: at most once, at least once and exactly once (AMQP Advanced Message Queuing Protocol Specification, Cisco Systems, 2018).

Message Queuing Telemetry Transport (MQTT)

MQTT is an application protocol designed in 1999 by IBM and standardized in 2013, which has a relatively small overhead, thus providing a possible application on devices with limited resources (memory, processor etc.) such as IoT devices. This protocol, like the HTTP protocol, uses TCP on the transport layer, but has a smaller overhead of 2 or 4 bytes. The protocol uses the principle of advertisers and subscribers. Facebook Messenger application uses this protocol. In terms of security, this protocol uses Transport Layer Security(TLS). Brokers may require a username and a password (Stanford-Clark & Truong, 2013).

Constrained Application Protocol (CoAP)

The main goal of this protocol is to reduce the overhead to a minimum and provide a mechanism that would be used on a large number of devices that have limitations in terms of power, resources and network considerations (low-range networks such as IEEE 802.15.4, Bluetooth, Low Power Wi-Fi). The HTTP protocol was uses as a model for development. It is important to note that CoAP is not a reduced HTTP, but it is a protocol optimized for M2M communication, which supports basic REpresentational State Transfer (REST) functionalities, common with the HTTP protocol. Also, CoAP in some things represents a step forward in comparison to HTTP. It supports multicast, asynchronous messaging and has a mechanism for finding resources (Sharma, 2014), (Shelby, 2014).

CoAP is an application protocol that uses two messaging models. It supports the request/response model, as well as the advertiser/subscriber model. Unlike HTTP, it relies on the User Datagram Protocol (UDP) protocol, but above UDP the DTPLS protocol can also be used to increase the security.

The most important part of the IoT architecture is the middleware part, because in that part there are services that mediate the communication with devices. In addition, the services provide the necessary data abstraction, storing and exchanging data with other services.

In order to provide the ability of communication between different platforms (hardware, computer), Web services use platform independent

data formats such as EXtensible Mark-up Language (XML) and JavaScript Object Notation (JSON).

Two basic types of services are:

SOAP (Simple Object Access Protocol) is a standard for exchanging structured information using HTTP and XML. It is platform independent, so it can be used on all computer platforms, as well as in all programming languages. The most common way to call the service method is through RPC (Remote Procedure Call) messages, where the client calls the server method and awaits its response. The SOAP protocol is based on WSDL (Web Services Description Language) and UDDI (Universal Description, Discovery, and Integration) technologies.

REST (REpresentational State Transfer) is an architectural style that was designed by Roy Thomas Fielding. Although not a protocol, it uses multiple protocols in its work, such as: HTTP, Uniform Resource Identifier(URI), JSON, and XML. The basis of this style is the HTTP protocol, which is used as a data transfer mechanism, with a limited number of commands such as: GET, POST, PUT, DELETE, etc. It is RESTfull in sense of performance, scalability, simplicity, modularity, visibility, portability and reliability.

Regarding the applicability in the IoT architecture, although both types provide similar services, in the specific constraints that are present in the IoT concept, primarily in device constraints, it is more convenient to use the RESTfull architecture (Zeng et al, 2011).

Web of Things

The foundation of Web (World Wide Web) is the Internet. The initial idea Tim Berners-Lee had for the development of the Web was to enable the interrelationships mechanism between documents in order to exploit and improve the Internet. The Web developed very rapidly and became much more than the exchange of documents. Now the Web is the largest platform that allows the development of Web applications in different domains. It has evolved from static pages, through applications, social networks to the latest stage, which is the social network of the devices, or Web of Things (Raggett, 2015).

WoT is a continuation of the IoT concept, which represents its starting point. IoT deals with access modes and communication protocols between Things and services, that is, IoT deals with the vertical structure. Unlike IoT where it is possible to send data from Things to the server and vice versa, WoT deals with the integration of Things into Web, that is, with the horizontal structure (Figure 3).

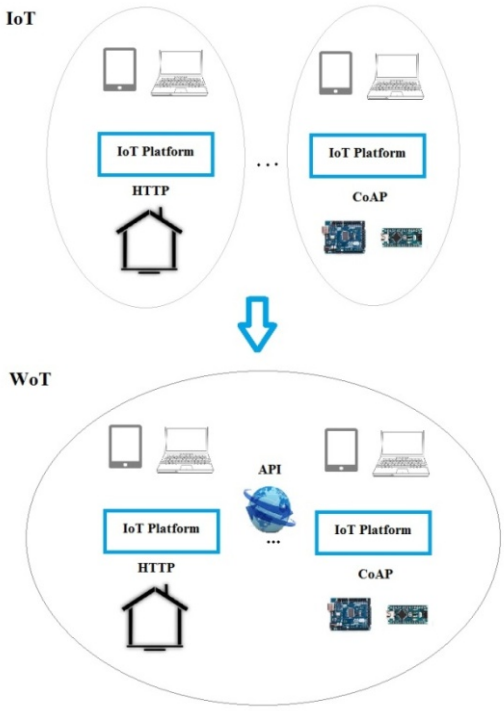

Figure 3 – Moving from IoT to WoT
Рис. 3 – Переход от интернета вещей к веб вещей
Слика 3 – Прелаз са интернета ствари на веб ствари

From Things, we now expect to be accessible through a web protocol such as HTTP, which represents a ticket for the web. The next step that needs to be done is to define a mechanism for using the IoT resources. This mechanism should be Open Source, it must take care of security and needs to be API (Application Programming Interfaces). Things will represent virtual objects that machines and people will be able to access and communicate with them.

With the development of the IoT platform, where each platform has its own mechanism, without interaction with other platforms, we will take a step backwards compared to the initial idea that Tim Berners-Lee had when designing the Web. The problem of the development of royalty-free and platform-independent standards has been discovered and there are currently working groups under the World Wide Web Consortium (W3C) that deal with the standardization and development of the WoT API since 2015 (White Paper for the Web of Things, 2016).

In terms of WoT architecture, the idea is that devices are directly linked to Cloud or through their proxies (Figure 4).

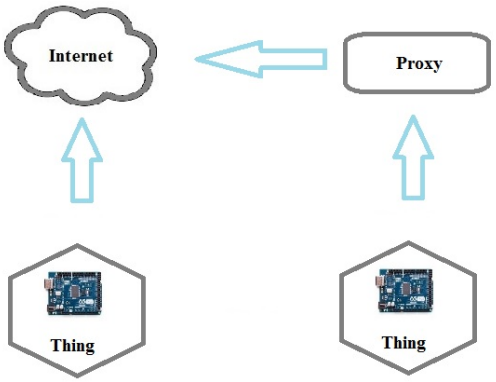

Figure 4 – Connecting Things to Web
Рис. 4 – Связывание вещей на веб
Слика 4 – Повезивање ствари на веб

Challenges in interconnection IoT

It is necessary to create a service architecture that will not be too much demanding for Things, so it would be possible to implement such services with the existing restrictions on Things. Such services should be able to work without an operating system, and with a limit of 8KB of memory. Such restrictions exclude the use of heavyweight Web services such as, for example, SOAP/WSDL (Guinard et al, 2011).

A protocol that meets the mentioned restrictions is the HTTP protocol, that is RESTfull architecture, which has basic access methods such as: GET, POST, PUT, DELETE. In addition to this protocol, it is also possible to use the CoAP protocol, which is similar to the HTTP protocol.

In order to make the data understandable for machine processing, they need to be in one of the languages that are understandable from the machines perspective, such as XML or JSON.

In most existing scenarios, IoT applications coincide, but are not interoperable among themselves. The goal is to harmonize the used data models as well as frameworks to minimize the human interaction in their communication. It is necessary to find an approach so that the data collected from the sensor will be reusable. It is also necessary to ensure that data can be shared (Gyrard, 2015).

In order to make Things capable of building complex systems, we need to enable certain APIs that will allow communication with each

other through HTTP or CoAP protocols or other protocols, and in this way we will create the WoT (Jara et al, 2014).

Currently in Europe, according to Google statistics, most countries have IPv6 present less than 34%, which means that it is still not possible to use all the IPv6 addressing capability (Google IpV6 Statistics). In addition, it is still not possible to expect Things to work as servers that will be able to process all the requests that come to them in terms of resources. These constraints imply the need to think about other ways to access the resources of Things (Barnaghi et al, 2013).

In terms of architecture used in the integration of Things, we encounter two types of architectures. The first is Direct Integration that for every Thing assigns an address, and has its own small embedded service. Another architecture is Indirect Integration that refers to Things that are not able to run their own services, but use other services as their proxy, often referred to as the smart gateway (Zeng et al, 2011).

Using the CoAP protocol, it is possible to reduce overhead, but the problem of finding resources using the web browser remains. This problem can be solved in several ways (Castron et al, 2016).

For Emmanuel Baccelli and Dave Raggett, there are two basic challenges to be solved in the development of the IoT concept. The first is the implementation of the IPv6 protocol at the hardware level of the IoT device. The second problem is providing end-to-end security (Baccelli & Raggett, 2015).

In summary, we can conclude that WoT's software architecture should enable:

- development of services on devices with limited resources,
- M2M communication,
- the creation of a data model that will be suitable for processing and reuse,
- an API that will allow resources to be visible from other services,
- secure communication.

In order to be able to implement these requirements independently (without using the IoT platform), we need to have a team of at least several people, who will work for several years (IoT Platforms The central backbone for the Internet of Things, 2015). This points to the need to use existing platforms, or middleware. Using middleware, which implements the mentioned requirements, we can save the time it takes to create our own applications.

There are currently a large number of IoT platforms, and most of them are commercial. Large companies such as Amazon, Google,

Microsoft, IBM etc. are present in the field of IoT platforms. One of the Open Source platforms that provides these requirements is FIWARE, which can be used in different scenarios, using open and free APIs.

Open Source FIWARE platform

FIWARE is a platform that has a layer of infrastructure, and provides the IaaS service. In addition to IaaS, the platform also provides PaaS service features. These service functionalities that the platform offers enable the development of our own IoT applications. The platform is also possible to host on your own hardware, but this feature is out of scope of this paper.

The FIWARE platform allows the creation of virtual machines in which we can create our own architecture for the IoT application. Also, the platform frees us from taking care of hardware, which makes it easier for developers to create a service for a specific application. In terms of security, it is possible to define the security police, which will take care of security and access to the application. Security of frontend and backend parts is achieved using the OAuth2 protocol.

The basis for the functioning of the FIWARE platform is the Open API Next Generation Service Interface (NGSI). NGSI defines data model, context data interface, context availability interface. Currently, two versions of this interface have been implemented, NGSI9 and NGSI10. (FIWARE-NGSI v2 Specification).

A particularly important part for the functioning of the interface is the data model. It consists of three parts:

Context Entities: It is an entity that can represent any Thing like a sensor, an actuator, a human, an animal, a car, etc. Each entity must have its own ID. To enable later search by type, you need to assign a type attribute.

Context Attributes: These are attributes that make up an entity. These attributes, besides their name, have additional attributes that describe the entity. These attributes are value, type and metadata that further describe the attribute.

Context metadata: As mentioned before, it is an optional attribute. It consists of the basic three parts (name, type, value).

The NGSI interface allows the data to be accessed through the REST method, whereby the data can be displayed in the JSON or XML format. The developers are free of concern about the communication protocols IoT uses, thanks to the IoT Generic Enablers (GE), which

translate all data that comes from Things into the NGSI form, with which developers can then work.

The software architecture of the platform is based on Generic Enablers (GE), which are elements that build the FIWARE platform. GEs can be added depending on the needs of a specific application, and a list of all GEs is available through the FIWARE catalogue.

The FIWARE platform enables integration of more services. Data can be exchanged using the NGSI API. The service architecture is based on the REST style, where resources are accessed based on their names, using URI. In this way, the horizontal interaction with other systems is achieved, hence we come to the infrastructure that can provide the development of Web of Things. Also, the infrastructure has mechanisms for generating and using Big Data. By using additional metadata attributes, we can achieve the infrastructure for semantic web development (Start Using FIWARE Right Now, 2018).

Example of using the FIWARE platform

An example that will serve us for testing the FIWARE platform is based on reading the values from the sensor, and the actuation of the DC motor depending on the values from the sensor. The goal we want to achieve by this example is to connect the hardware of some Thing with the service, which will be used for management, and that part will represent the IoT domain. The next functionality we want to explore is the ability to access Things over the Web browser and gain value in order to check the possibility to integrate into the Web. In this way, we want to test the functionalities that the FIWARE platform provides in the WoT domain.

In the testing stage, we will use the ATMega2560 microcontroller, through which we will collect data from the sensors, as well as starting the activity to check the communication on both sides. Communication between hardware and services will be achieved over the Internet. To connect the hardware to the Internet, we will use the Wi-Fi module ESP8266 (Figure 5).

Both virtual sensor, as well as a virtual actuator, will be described using the JSON format. We will describe the sensor with the basic attributes such as ID, type, owner, hardware, location, temperature, humidity. When creating a virtual sensor, we can enter the default values. The sensor description is shown in Figure 6.

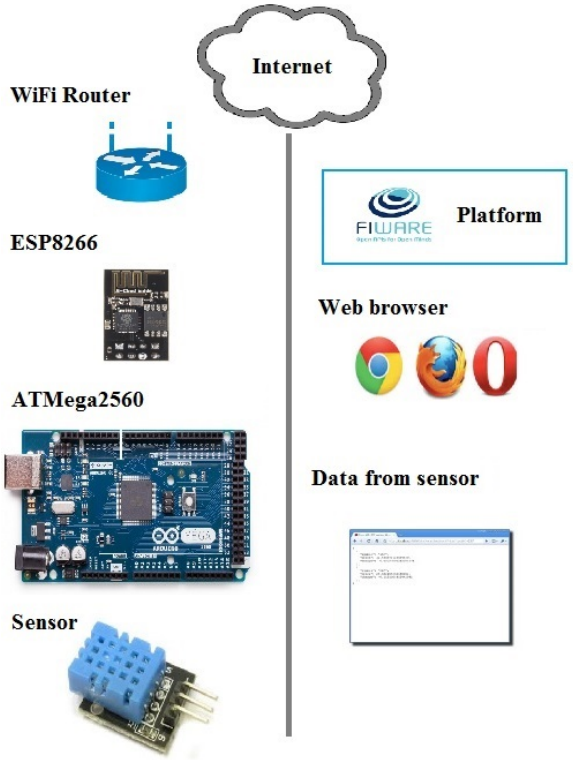

Figure 5 – Components of the example
Рис. 5 – Компоненты примера
Слика 5 – Компоненте примера

The state attribute refers to the status of the actuator, whether it is turned on or not, while the speed attribute represents the current motor speed. In the later physical realization, for easier display, the speed attribute will be represented by the number of LEDs that will signal the speed at which the motor is currently running.

The listed attributes will allow us to map real Things into virtual ones that will live on the Web as separate entities (Figure 7).

When the entities are created, using the NGSI API, we have to connect hardware (Things) to them, and then use the virtual sensor or actuator.

A physical sensor should send data to a virtual sensor in order to have an image of the device in real time. The virtual sensor is located at the following address: http://130.206.xxx.xxx:1026/v2/entities/Sensor-01/.

```
{
    "id": "Sensor-01",
    "type": "DHT11",
    "owner": {
        "value": "User1",
        "type": "String"
    },
    "hardware": {
        "value": "ArduinoDHT11",
        "type": "String"
    },
    "location": {
        "value": "44.7866, 20.4489",
        "type": "geo:point"
    },
    "temperature_c": {
        "value": 0,
        "type": "Float"
    },
    "humidity": {
        "value": 0,
        "type": "Float"
    }
}
```


Figure 6 – Model of a sensor
Рис. 6 – Модель датчика
Слика 6 – Модел сензора

Unlike the sensor that updates the virtual sensor, the actuator monitors the state of its virtual entity, and, depending on its condition, adjusts its state. The virtual actuator is located at the following address: http://130.206.xxx.xxx:1026/v2/entities/Actuator-01/.

After connecting, the data on the server represent the real state of the device, and are available through the Web browser in the JSON format. Figure 8 shows the virtual actuator data. Based on the data, we can conclude that our motor is started and running at speed 2.

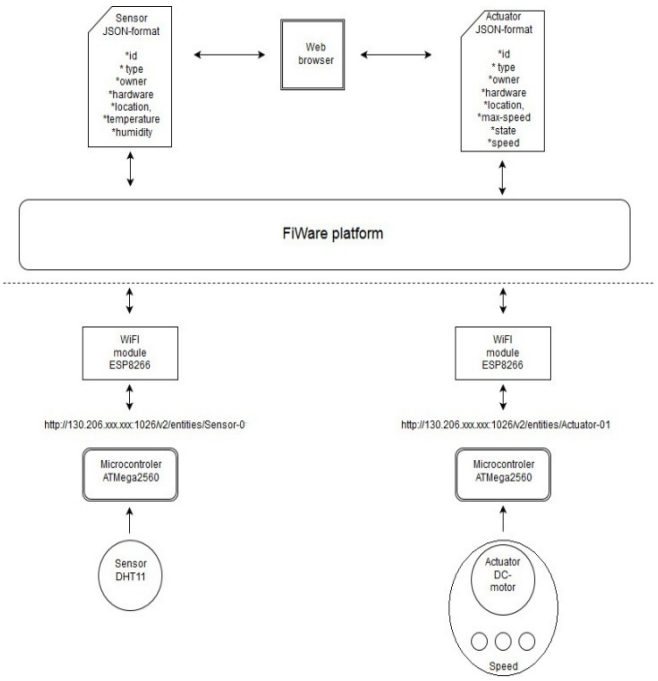

Figure 7 – Architecture of the example
Рис. 7 – Архитектура примера
Слика 7 – Архитектура примера

As the data is available via the Web browser, it is possible to manage actuators, or read values from the sensors using our own applications, and to connect with other sources.

In our case, the idea is that when the temperature exceeds a certain value, the actuation of a small motor at a certain speed is carried out. Another advantage of the FIWARE platform is the use of GE, such as the CEP (Complex Event Processing)., We will use this GE to monitor the state of the sensor, and to start the actuator. A practical example is given in Figure 9.

```
id:           "Actuator-01"
type:         "DCmotor"
▼ hardware:
    type:     "String"
    value:    "ArduinoDCmotor"
    metadata:
▼ location:
    type:     "geo:point"
    value:    "44.7866, 20.4489"
    metadata:
▼ max-speed:
    type:     "Integer"
    value:    3
    metadata:
▼ owner:
    type:     "String"
    value:    "User2"
    metadata:
▼ speed:
    type:     "Integer"
    value:    2
    metadata:
▼ state:
    type:     "Boolean"
    value:    true
    metadata:
```


Figure 8 – JSON model of the actuator
Рис. 8 – JCOH модель актуатора
Слика 8 – JCOH модел актуатора

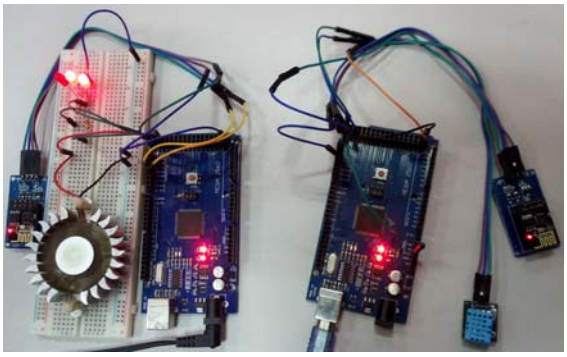

Figure 9 – Practical example
Рис. 9 – Пример из практики
Слика 9 – Практичан пример

Conclusion

Based on the presented theory and a practical example, we can conclude that the FIWARE platform allows us to create virtual devices (Things) that will reflect the state of the physical devices (Things), which have computational constraint (8KB memory). This virtual representation of the devices (Things) is displayed in a platform independent JSON format that allows later processing, and is available using the REST methods.

This system architecture allows us to manage devices (Things), that is, to use their resources by looking at them as services. The most important feature is that these resources are available using the Web protocol. The developers are free from concern about the complexity of device management (Things), using a virtual representation of the devices (Things). This functionality enables easier use in future applications.

The FIWARE platform is based on an open API, so when WoT API is created by the W3C some time in the future, it will probably be implemented on the FIWARE platform as well. Until then, the FIWARE platform provides the ability to create and test WoT applications, using the APIs that provide the necessary functionality for implementing the WoT concept.

References

AMQP Advanced Message Queuing Protocol Specification, Cisco Systems. 2018. [Internet]. Available at: https://www.rabbitmq.com/resources/specs/amqp0-9-1.pdf. Accessed: 2018 Mar 23.

Baccelli, E., & Raggett, D. 2015. The Promise of the Internet of Things and the Web of Things. In *Special theme The Internet of Things and The Web of Thing, ERCIM,* pp.8–11. [Internet]. Available at: https://ercim-news.ercim.eu/images/stories/EN101/EN101-web.pdf. Accessed: 2018 Mar 28.

Barnaghi, P., Sheth, A., & Henson, C. 2013. From Data to Actionable Knowledge: Big Data Challenges in the Web of Things [Guest Editors' Introduction]. *IEEE Intelligent Systems*, 28(6), pp.6-11. Available at: https://doi.org/10.1109/MIS.2013.142.

Botta, A., Donato, W., Persico, V., & Pescapé, A. 2015. Integration of Cloud computing and Internet of Things: A survey. *Future Generation Computer Systems*, 56, pp.684-700. Available at: https://doi.org/10.1016/j.future.2015.09.021.

Castron, M., Jara, A., & Skarmeta, A. 2016. Enabling end-to-end CoAP-based communications for the Web of Things. *Journal of Network and Computer Applications*, 59, pp.230-236. Available at: https://doi.org/10.1016/j.jnca.2014.09.019.

FIWARE-NGSI v2 Specification. [Internet]. Available at: http://docs.orioncontextbroker.apiary.io/#introduction/specification/introduction, Accessed: 2018 Mar 12.

Google IpV6 Statistics 2018. [Internet]. Available at: https://www.google.com/intl/en/ipv6/statistics.html#tab=per-country-ipv6-adoption&tab=per-country-ipv6-adoption, Accessed: 2018 Mar 12.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), pp.1645-1660. Available at: https://doi.org/10.1016/j.future.2013.01.010.

Guinard, D., Trifa, V., Mattern, F., & Wilde, E. 2011. From the Internet of Things to the Web of Things: Resource Oriented Architecture and Best Practices. *Architecting the Internet of Things*, pp.97-129. Available at: https://doi.org/10.1007/978-3-642-19157-2_5.

Gyrard, A., Bonnet, C., Boudaoud, K., & Serrano, M. 2015. Assisting IoT Projects and Developers in Designing Interoperable Semantic Web of Things Applications. In *2015 IEEE International Conference on Data Science and Data Intensive Systems*. Institute of Electrical and Electronics Engineers (IEEE), pp.659-666. Available at: https://doi.org/10.1109/DSDIS.2015.60.

Internet of Things (IoT) History. [Internet]. Available at: https://www.postscapes.com/internet-of-things-history. Accessed: 2018 Feb 5.

IoT Platforms The central backbone for the Internet of Things. IoT Analytics GmbH. 2015. [Internet]. Available at: http://iot-analytics.com/wp/wp-content/uploads/2016/01/White-paper-IoT-platforms-The-central-backbone-for-the-Internet-of-Things-Nov-2015-vfi5.pdf. Accessed: 2017 Dec 23.

Jara, A.J., Olivieri, A.C., Bocchi, Y., Jung, M., Kastner, W., & Skarmeta, A.F. 2014. Semantic Web of Things: an analysis of the application semantics for the IoT moving towards the IoT convergence. *International Journal of Web and Grid Services*, 10(2/3), p.244. Available at: https://doi.org/10.1504/IJWGS.2014.060260.

Perera, C., Zaslavsky, A., Christen, P., & Georgakopoulos, D. 2014. Context Aware Computing for The Internet of Things: A Survey. *IEEE Communications Surveys & Tutorials*, 16(1), pp.414-454. Available at: https://doi.org/10.1109/SURV.2013.042313.00197.

Raggett, D. 2015. The Web of Things: Challenges and Opportunities. *Computer*, 48(5), pp.26-32. Available at: https://doi.org/10.1109/MC.2015.149.

Saint-Andre, P., Smith, K., & Tronçon, R. 2009. *XMPP: The Definitive Guide*. O'Reilly Media, pp.3-27.

Sharma, V. 2014. *Understanding Constrained Application Protocol*. Exilant Technologies Pvt, pp.7-18. [Internet]. Available at: www.coapsharp.com/wordpress/?wpdmdl=504, Accessed: 2017 Dec 11.

Shelby, Z. 2014. *Constrained Application Protocol (CoAP), Internet Engineering Task Force*. Internet Engineering Task Force (IETF). [Internet]. Available at: https://tools.ietf.org/html/rfc7252, Accessed: 2017 Nov 10.

Stanford-Clark, A., & Truong, H.L. 2013. *MQTT for Sensor Networks (MQTT-SN) Protocol*. International Business Machines Corporation (IBM). [Internet]. Available at: http://mqtt.org/new/wp-content/uploads/2009/06/mqtt-sn_spec_v1.2.pdf. Accessed: 2016 Aug 20.

Start Using FIWARE Right Now. 2018. [Internet]. Available at: http://fiwaretourguide.readthedocs.io/en/latest/, Accessed: 2018 Jan 31.

White Paper for the Web of Things. 2016. [Internet], Available at: http://w3c.github.io/wot/charters/wot-white-paper-2016.html, Accessed: 2017 Nov 14.

Wortmann, F., & Flüchter, K. 2015. Internet of Things. *Business & Information Systems Engineering*, 57(3), pp.221-224. Available at: https://doi.org/10.1007/s12599-015-0383-3.

Zeng, D., Guo, S., & Cheng, Z. 2011. The Web of Things: A Survey (Invited Paper). *Journal of Communications*, 6(6). Available at: https://doi.org/10.4304/jcm.6.6.424-438.

Zhong, N., Yau, S.S., Ma, J., Shimojo, S., Just, M., Hu, B., Wang, G., Oiwa, K., Anzai, Y. 2016. Brain Big Data in Wisdom Web of Things. In N. Zhong, J. Ma, J. Liu, R. Huang, & X. Tao Eds., *Wisdom Web of Things*. Cham: Springer Nature., pp.339-349. Available at: https://doi.org/10.1007/978-3-319-44198-6_15.

FIWARE: РАЗВИТИЕ ПЛАТФОРМЫ ДЛЯ ИНТЕРНЕТА ВЕЩЕЙ

Иван А. Тот[а], *Душан* Л. Богичевич[б], *Младен* Б. Трикош[а], *Комлен* Г. Лалович[в]

[а] Университет обороны в г. Белград, Военная академия,
Кафедра информационных систем и телекомумникационной инженерии,
г. Белград, Республика Сербия
[б] Вооружённые Силы Республики Сербия, Генеральный штаб,
Управление информатики и телекоммуникаций (J-6),
Центр командно-информационных систем, г. Белград +
Университет в г. Ниш, Факультет электроники,
г. Ниш, Республика Сербия
[в] Факультет прикладного менеджмента, экономики и финансов,
г. Белград, Республика Сербия

ОБЛАСТЬ: компьютерные науки, информационные технологии
ВИД СТАТЬИ: профессиональная статья
ЯЗЫК СТАТЬИ: английский

Резюме:

Предметом данной статьи является Веб вещей, как продолжение развития концепта Интернета вещей. Веб вещей представляет собой шаг к связыванию умных вещей с существующим Веб

окружением, при рассмотрении актуальных проблем, таких как: гетерогенность, масштабируемость и применяемость, то есть удобство использования. Данная статья посвящена современным возможностям и вызовам для развития концепта Веб вещей. В статье представлены теоретические постулаты концепта Интернета вещей, в частности: архитектура, протоколы, услуги и собственно вещи, которые и являются основой обоих концептов. В работе описаны необходимые условия для развития концепта Веб вещей. Главным научным вкладом настоящей статьи является предложение разработанной архитектуры, основанной на платформе FIWARE, в качестве основы для развития Веб вещей. Предлагаемая архитектура основана на реальном примере.

Ключевые слова: Интернет вещей (IoT), Веб вещей (WoT), FIWARE.

FIWARE: РАЗВОЈНА ПЛАТФОРМА ЗА ВЕБ СТВАРИ

Иван А. Тот[а], *Душан* Љ. Богићевић[б], *Младен* Б. Трикош[а], *Комлен* Г. Лаловић[в]

[а] Универзитет одбране у Београду, Војна академија,
Катедра информационих система и телекомуникационог инжењерства,
Београд, Република Србија
[б] Војска Србије, Генералштаб,
Управа за телекомуникације и информатику (J-6),
Центар за командно-информационе системе, Београд +
Универзитет у Нишу, Електронски факултет, Ниш, Република Србија
[в] Факултет за примењени менаџмент, економију и финансије,
Београд, Република Србија

ОБЛАСТ: информатика, ИТ
ВРСТА ЧЛАНКА: стручни чланак
ЈЕЗИК ЧЛАНКА: енглески

Сажетак:

Као наставак концепта интернет ствари, веб ствари представљају корак ка повезивању интелигентних ствари са постојећим веб окружењем, уз разматрање проблема као што су хетерогеност, скалабилност и употребљивост. Овај рад је посвећен тренутним могућностима, као и изазовима за развој у концепту веб ствари. У раду су описане теоријске основе концепта интернет ствари, као што су архитектура, протоколи, услуге и саме ствари, које су основа оба концепта. Рад се бави потребним предусловима за развој концепта веб ствари. Главни допринос рада је предлог архитектуре заснован на FIWARE

платформи као основи за развој веб ствари. Демонстрација предложене архитектуре описана је реалним случајем.

Кључне речи: интернет ствари (IoT), веб ствари (WoT), FIWARE.

Paper received on / Дата получения работы / Датум пријема чланка: 04.04.2018.
Manuscript corrections submitted on / Дата получения исправленной версии работы / Датум достављања исправки рукописа: 03.07.2018.
Paper accepted for publishing on / Дата окончательного согласования работы / Датум коначног прихватања чланка за објављивање: 05.07.2018.