The 8th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
24-26 September 2015, Warsaw, Poland

Implementation of Standardized 6LoWPAN Based Application Layer Protocols

Fesseha Tsegaye Mamo, Axel Sikora
Laboratory Embedded Systems and Communication Electronics
University of Applied Science Offenburg, 77652 Offenburg, Germany
{fesseha.mamo, axel.sikora}@hs-offenburg.de

***Abstract*—6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks) is gaining more and more attraction for the seamless connectivity of embedded devices for the Internet of Things (IoT). Whereas the lower layers (IEEE802.15.4 and 6LoWPAN) are already well defined and consolidated with regard to frame formats, header compression, routing protocols and commissioning procedures, there is still an abundant choice of possibilities on the application layer.**

Currently, various groups are working towards standardization of the application layer, i.e. the ETSI Technical Committee on M2M, the IP for Smart Objects (IPSO) Alliance, Lightweight M2M (LWM2M) protocol of the Open Mobile Alliance (OMA), and OneM2M. This multitude of approaches leaves the system developer with the agony of choice.

This paper selects, presents and explains one of the promising solutions, discusses its strengths and weaknesses, and demonstrates its implementation.

Keywords—6LoWPAN; ETSI M2M; IPSO; LWM2M; OMA; ZigBee

I. INTRODUCTION

The Internet of Things (IoT) is becoming more and more real with the advances of small, reasonably powerful, energy and cost efficient embedded device that communicate not only with each other but can connect seamlessly with the existing Internet. On one side, these devices have interfaces to the physical world, whereas on the other side, they connect to the virtual world of databases and servers in the Internet. Thus they are a cornerstone of a Cyber-Physical System (CPS). Due to the still restricted resources of the embedded devices, but even more due to the scalability effects of the Internet of Things with the potential advent of billions of devices in the Internet and of thousands or even millions of devices in a single network, it is necessary to keep the complexity of the solution as low as possible.

Although sometimes still disputed, it can be observed that the lower layers of the protocol stacks are already well defined with IEEE802.15.4 [1] and 6LoWPAN [2]. 6LoWPAN is developed to enable IPv6 connectivity for constrained embedded devices that use 802.15.4 low-power wireless communication. Although issues remain open with regard to the selection of physical layer, unified commissioning procedures, routing functions and parameters, and security; a reasonable level of interoperability has already been achieved that is comparable with other more homogeneous protocol stacks.

However, the situation is still completely different on the application layer level, where a good number of diverse application frameworks, protocols, profiles, and semantic descriptions has emerged and continues to see the light of the world. [3] gives an overview on the most important candidates. However, at this moment, it cannot be said that one solution will outperform all others. IoT will be the next big thing like the world-wide web, but unlike there, it cannot be anticipated that one protocol or solution will rule all IoT. The best fitting solution still strongly depends on the concrete applications' requirements.

In ch. II, this paper describes the proposed approach that addresses each aspect of the application layer in a modular way. The pros and cons of this approach are also described. In ch. III, a demonstration implementation targeting different embedded hardware is described. Finally, the testing of the demonstrator is briefly described in ch. V along with a short summary and outlook in ch. VI.

II. ARCHITECTURE

A. Overview

To have an architecture that provides a solution for a horizontal implementation of applications without the limitations of being dependent on a specific protocol, a wider view towards what the application constitutes is required. An application can be seen as a solution composed of the following components:

- Application framework
- Application layer protocol
- Application semantics
- Data formats / Data semantics

Application frameworks define a general understanding of which resources or services are provided by an application, how to identify these resources, and how to manage them. Different application frameworks exist originating from various interest groups

based on region, application sector and so on. In the work discussed here in this paper, analysis of Zigbee [4], ETSI M2M [5], LWM2M [6], EEBus [7], and OPC-UA [8] has been done for use as an application framework solution.

The application layer protocol enables the actual transfer of data from one WSN to another utilizing the functionality provided by lower layers. An application protocol for the Internet of Things shall have small overhead as it should run on constrained devices. The use of TCP/IP as transport mechanism and relatively large overhead makes HTTP unsuitable for use in constrained devices. MQTT-SN and CoAP are the potentially capable application protocols for the Internet of Things.

The application semantics can be seen as what a specific WSN does or what applications the WSN runs, e.g a light or a thermostat. Assuming all the communication and resource access mechanisms are provided, the application semantics helps in identifying what kind of specific application is running on a WSN and in turn enables the user (client) to retrieve resources and manipulate the characteristics (features) provided by the WSN.

Different application protocols and implementations result in vertical solutions that are not able to interoperate with each other. Various problems arise at various levels when it comes to finding a universal solution for the compatibility issues in the application layer of a 6LoWPAN based communication stack for WSN. Therefore a more detailed look on what an application or application layer constitutes and where the problems arise is necessary. Fig. 0 shows one possible architecture for an application layer for the IPv6-based Internet of Things, which was selected for the given implementation and will be discussed in more detail in the remainder of this paper.

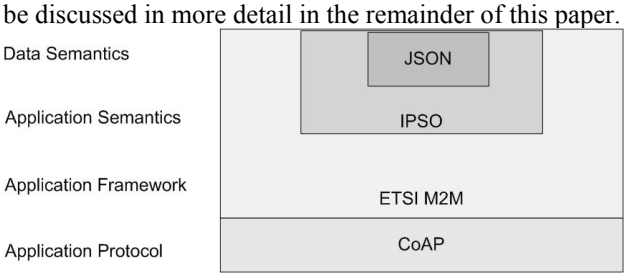


Figure 1. Application layer architecture for the IPv6 based Internet of Things.

ETSI M2M [5] provides a framework for an interoperable M2M network. Even though ETSI M2M standard is coming from the European Telecommunication Standardization Institute, an international effort is underway to come up with a global solution for interoperability of M2M networks from a global standards initiative for M2M communications and Internet of Things, OneM2M [9]. The standards from OneM2M are similar in architecture as the ETSI M2M standard but differ in the details.

Internet Protocol for Smart Objects (IPSO) is defined according to OMA LWM2M resource model. It should be used within an LWM2M engine. However since the resource model follows a RESTful architecture, it can also be used for application semantics in ETSI M2M. In addition LWM2M could potentially be also used as a management entity in ETSI M2M. Currently ETSI M2M suggests the use of OMA DM [10] and BBF TR-69 [11] for device management. OneM2M proposes LWM2M as the ultimate management option for constrained devices.

B. *Strengths and weaknesses*

The application layer approach discussed in this paper tackles all important components of the application layer by addressing them separately. None of existing solution tries to address all four components; application framework, application layer protocol, application semantics and data format. ESTI M2M defines an application framework that is independent of application protocol, which is a very important thing. But it does not define application semantics methods and even representation of resources in the application framework needs more work to be used with the data formats which the standard proposes.

LWM2M provides an application framework and an application layer protocol, which are both specifically tied with CoAP. Application semantics is provided through IPSO, where multiple data formats are also supported. This analogy continues for all other solutions. The main point is that all solutions lack one design aspect or another. This work proposes an application layer architecture which is believed to solve the problem discussed above through the combination of different approaches, i.e. the combination of a separate application layer architecture approach and specific protocols. While it is believed that the architecture is a generic solution, the specific protocols selected can be and should be replaceable with other equivalent solutions. This is useful in reusing all existing solutions as necessary.

The use of IPSO objects in an ETSI M2M framework can be seen as a proprietary solution because it is unlikely that any other device manufacturer uses the same approach. On the other side it is not a proprietary solution because ETSI M2M specification remains to address application semantics issues. Therefore, a device with this architecture can seamlessly fit in an ETSI M2M network but it is up to applications on the other side to figure out how the applications are interpreted.

III. IMPLEMENTATION

A. *Hardware*

A prototype implementation was targeted in our project to demonstrate the proposed application layer architecture. The setup is shown in Fig. 0 and contains the most important components. The Network and Gateway Service Capability Layers (NSCL & GSCL) from the ETSI M2M part are executed on a standalone computer and a Raspberry Pi respectively. Communication between these two components is facilitated through LAN and WLAN connections. The

Raspberry Pi is connected with the embedded devices and their test applications through an IEEE 802.15.4 physical link and IPv6 network link using 6LoWPAN. The Network and Gateway Service Capability Layers of the ETSI M2M specification provide interfaces for standardized communication between device, gateway and network components.

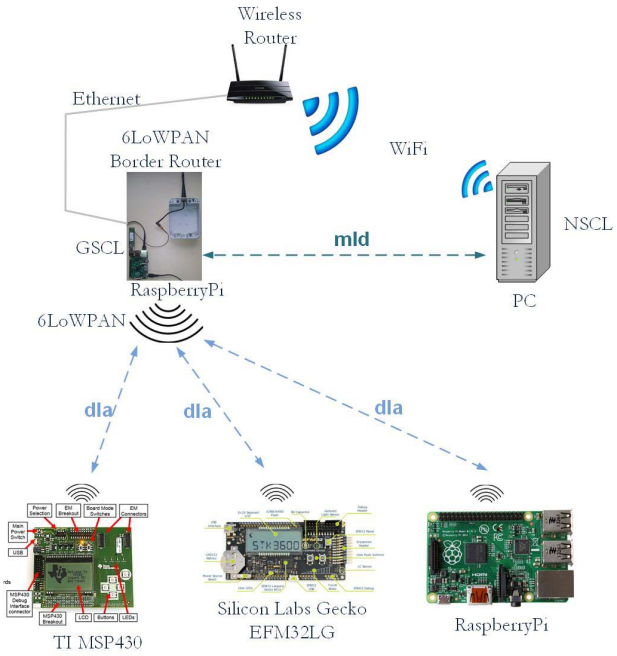


Figure 2. Hardware setup.

Three classes of embedded hardware devices have been chosen for running the sample application. These are a Raspberry Pi, a Texas Instruments MSP430 development kit, and a Silicon Labs Gecko EFM32LG development kit. TABLE I gives an overview on the specifications of the three device classes.

TABLE I HARDWARE SPECIFICATION

Device name	Flash	ROM	CPU
Raspberry Pi	512 MB	7.2 GB	700 MHz
TI MSP430 Dev Kit	256 KB	16 KB	25 MHz
SiLabs Gecko EFM32LG	256 KB	32 KB	48 MHz

The Raspberry Pi, acting as a border router, connects to a 6LoWPAN and LAN network. A USB pluggable @any-900-2 dongle with external antenna connector is serving as an RF module for the Raspberry Pi. The TI MSP430 development kit has a TI CC1120 transceiver module. For development purposes the Silicon Labs Gecko EFM32LG development board (Fig. 0) is used. The board is used together with an Atmel ATZB-RF-212B transceiver module.

B. *Software*

Since this work mainly focused on developing a sample application, it was required to use multiple open source software projects. OM2M [12], Wakaama [13], and Erbium [14] projects are open source implementations of ETSI M2M, LWM2M and CoAP respectively. OM2M is an open source service platform for M2M interoperability based on the ETSI-M2M standard. The Eclipse Wakaama project is an open source LWM2M implementation. Erbium is a CoAP implementation for the Contiki operating system. These software components were used in this work.

Major tasks of the software development included integration of the different open source components and developing a library that provides APIs for interaction with the M2M network. This library provides the following functionalities:

- Creating and registering a device application in the ETSI M2M network
- Initiating the LWM2M engine that is used for managing IPSO application objects.

The software structure is composed of existing implementations of the lower layers and the newly developed application layer which is the main focus of this paper. The software structure is shown in Fig. 4.

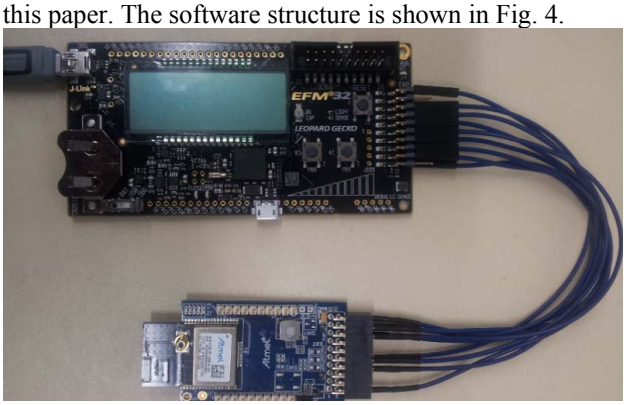


Figure 3. Silicon Labs Gecko EFM32LG board setup.

IV. IMPLEMENTATION

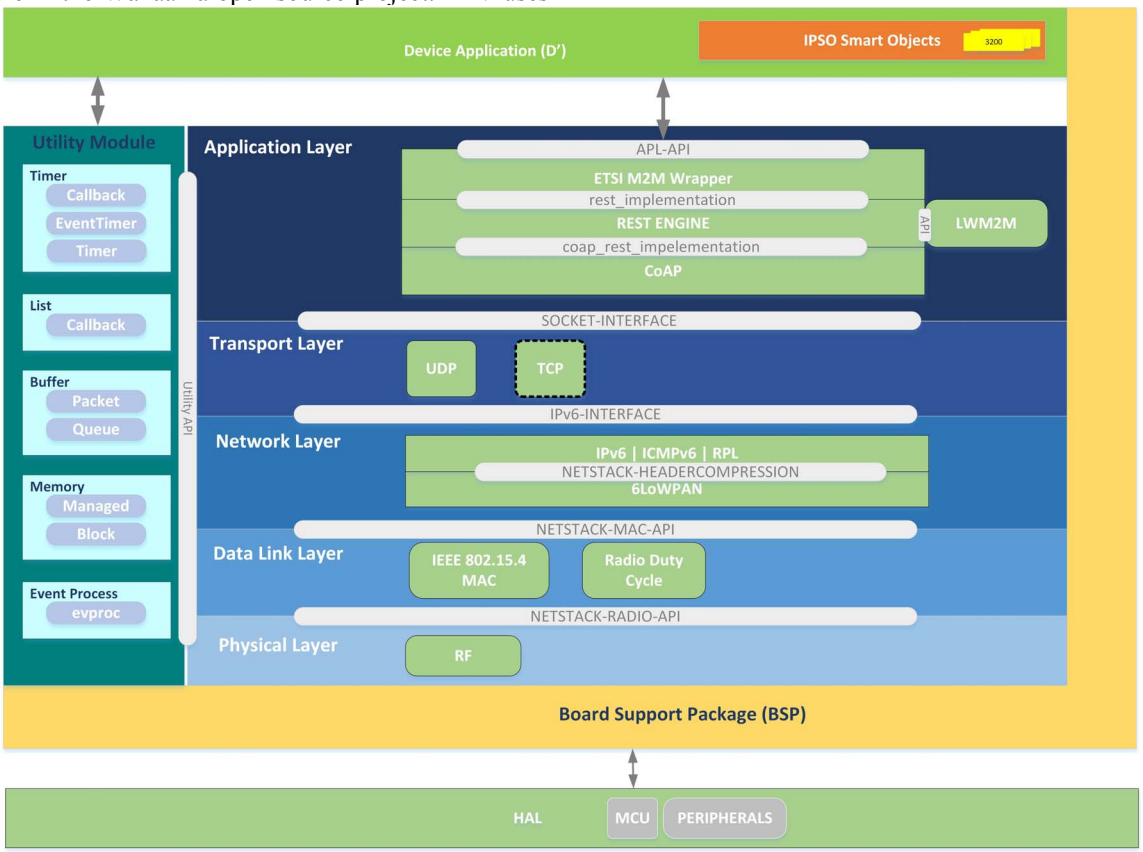
As described in the previous sub-chapter the device application has registration, authentication, creating resources, and manages application objects (IPSO) features. The application life cycle is described in a state machine, which represents all possible states of the device application. First the application goes through a registration process which involves creating resources on the service capability layer, later it initializes the IPSO objects and listens for requests to access these objects.

The first task attempts to register the user created application(s). This is done by sending an `applicationCreateRequestIndication` primitive to the local SCL (GSCL). This request uses, depending on the protocol supported by the requesting client, a CoAP POST method. In this POST request the representation of the application resource, which shall be created, is also sent. The representation holds important information about the application to be created. These are the application id and the application point of contact path (aPoc). The application id uniquely identifies the application in the SCL. The application point of contact

path is used by the SCL to contact the application. It includes the IP address of the device and listening port. If the creation of the application succeeds, the response (`applicationCreateResponseConfirm` primitive) will hold the representation of the created application in the agreed data format. XML is the only supported data format in OM2M and therefore XML representation of the application is sent in the response.

After the application is successfully registered the device application becomes accessible through the locally connected SCL (GSCL) and an authenticated network application can discover it. In addition, the device application capabilities such as embedded IPSO objects can be published as a '`searchString`' attribute to facilitate easy discovery. In this work, '`searchString`'s are not implemented because it requires long strings, which in turn demands larger memory which is a luxury on constrained sensor nodes. But it has been tested using the Copper [15] Firefox plugin by updating an already created device application.

LWM2M suggests the use of JSON, Type-Length-Value (TLV) or plain text for data representation format. In the Wakaama open source project TLV is used. Even though JSON is the recommended data format as a result of the study conducted in this work, TLV has been used in the demo implementation because it is already available in the Wakaama open source project. TLV uses binary representation of data property and results in smaller data sizes than JSON. It is not the recommended mechanism in this work because of the easy availability of JSON in web technologies as compared to TLV. A TLV encoder and decoder is provided by the Wakaama project. This TLV parser is available only in C language, which makes it difficult to use directly in a web browser. In order to create a network application that can retrieve data from device applications in TLV format, a CGI script has been created using the Wakaama TLV parser. This CGI script is executed on a web server that is running on a RaspberryPi which is the same device that runs the GSCL. Therefore requests from network applications can be made in two ways. For discovery direct requests should be made to the GSCL which is available at port 8181. In order to retrieve complex resources of IPSO objects (eg. /3311/) after the discovery process is finished, the request should be sent to the CGI script with the requested IPSO resource as a query string. The web server where the CGI script is located (on port 80) then makes a request to the GSCL, which is running on the same device, but listens to a different port. It then sends back a parsed representation of the retrieved data. Requests to simple objects (eg /3311/0/5850) should be made directly to the SCL as the response is not TLV encoded. This procedure is shown in Fig. 5.



* ICMPv6 is used to transport RPL messages.

Figure 4. Software structure.

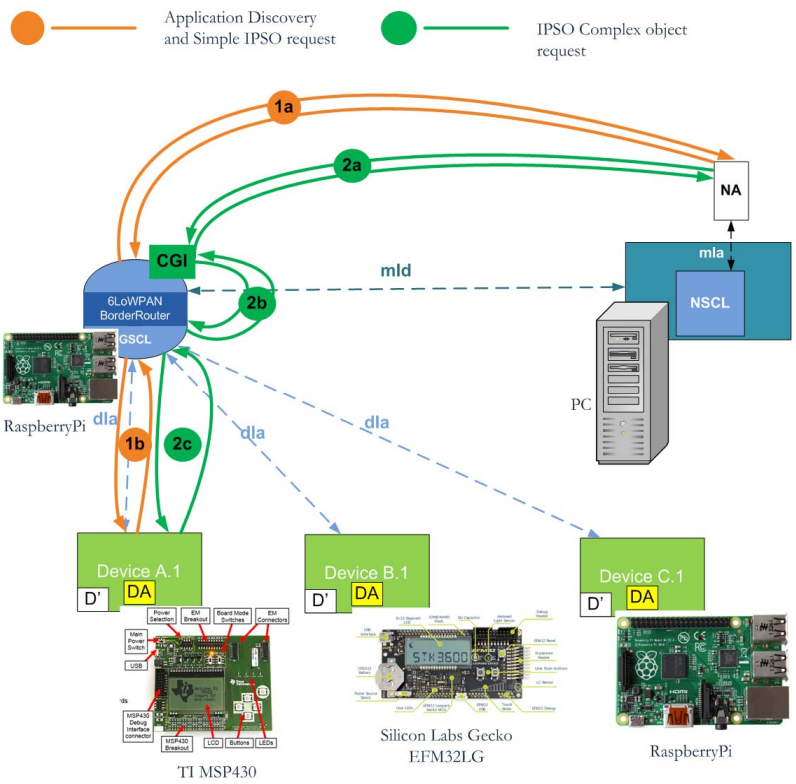


Figure 5. Application discovery and IPSO object retrieval.

IPSO Light Control object (3311) is created for demonstration purpose. Table I shows the 3311 IPSO object. The 5850 On/Off resource is implemented in the application demonstrator.

Sending a POST request without payload toggles an LED on the Silicon Labs Gecko EFM32LG development board.

POST `http://gsclAddress:8181/om2m/gscl/applications/Light/3311/0/5850`

A GET request to this same resource returns the status of the light it controls.

GET `http://gsclAddress:8181/om2m/gscl/applications/Light/3311/0/5850`

TABLE II IPSO 3311 LIGHT CONTROL OBJECT

Resource	Resource ID	Multiple instance	Type
On/Off	5850	No	Boolean
Dimmer	5851	No	Integer
On Time	5852	No	Integer
Cumulative active power	5805	No	Float
Power factor	5820	No	Float

V. TEST

Software implementations can be tested through different procedures and in multiple ways. Functional tests prove the correctness of the output with respect to the input defined in the specification.

Testing of software functionalities has been done using different tools to make sure that the initially planned features are implemented and functional. Positive and negative scenarios of the functionalities have been tested. Tools such as Wireshark and Postman have been used. Table II summarizes the tools used to test each features.

TABLE III FUNCTIONAL TESTS AND TOOLS

Feature	Test tool/mechanism
Authentication and registration	Wireshark [16] and web interface of OM2M
Creating container resource	Wireshark and web interface of OM2M
Respond to requests coming from the GSCL	Wireshark and Postman [17] Google Chrome plugin

By monitoring the traffic exchange between the GSCL and the device application, the registration procedure has been observed to correspond to the expected result. Traffic monitoring is done using Wireshark. A block wise transfer is necessary to POST the application resource which is larger than the maximum CoAP block size configured on the sensor device. The successful registration of the application can also be verified from the web interface of the OM2M as shown in Fig. 6.

What is described above can be seen as the positive scenario of the application registration, where there are no other applications registered on the GSCL with the same application id. In conflicting situation where an application with the same id exists the registration fails with the status code of '5.03' STATUS_CONFLICT.

Managing this situation needs thorough design and implementation. The application again needs to send a registration request with an empty application id. Then the GSCL will register the application with a system generated application id. This application id is returned in the response application representation. The device application then has to retrieve the application id from the response message for further communication with its own representation on the SCL. Because this procedure requires storing the response message, which can reach a size of up to 2000 Bytes, the response message is ignored and only the status code is checked. If the status code shows an application id conflict the application will consider it as it is already registered.

Figure 6. GSCL resource tree

The discovery feature of the SCL is also demonstrated using the POSTMAN Google chrome plug-in. This plug-in used to send different types of web requests to a web service and response is displayed in XML or raw data format. Discovery is done by sending a GET request to the discovery resource of the SCL. The discovery result can also be narrowed by adding a 'searchString' query parameter on the request.

VI. Summary and Outlook

With the implementation of the application demonstrator it has been possible to show the possibility to integrate different protocols for a unified solution. In this work, IPSO objects were used with ETSI M2M SCLs to provide application semantics. This approach is acceptable when looked at from the perspective of an ETSI M2M standard because it is not concerned with the content of applications. But if this approach were to be used with an LWM2M server the necessary management procedures of LWM2M has to be fulfilled.

The main challenge in this work was not to come up with a direct solution but to come up with a solution that is acceptable in the industry and backed up by a standard which doesn't lead to another proprietary solution. All existing solutions fail to address each and every aspect of the application layer. To mention few examples; ETSI M2M, at this moment, lacks supporting resources to implement the standard for constrained devices (eg. JSON representations) and application semantics and LWM2M is short of supporting horizontal integration of solutions and independence from the underlying transport protocol, i.e CoAP.

LWM2M could potentially be also used as a management entity in ETSI M2M. Currently ETSI M2M suggests the use of OMA DM and BBF TR-69 for device management. OneM2M proposes LWM2M as a potential management option for constrained devices.

In this work main focus has been given on introducing an application layer approach and integration of different application and management protocols. Security, commissioning and autonomous operation are areas to further extend this work. Further work can also be done to study OneM2M standards according to the approach of this work.

VII. Acknowledgment

The work of this paper was executed in close cooperation with and co-financed by KundoXT GmbH, St. Georgen, Germany, in the framework of an "Innovationsgutschein" of the State Baden-Wuerttemberg. The authors are grateful for the opportunity and the intense and fruitful discussions in the project. They look forward to the joint future activities.

References

[1] IEEE Std. 802.15.4 – 2003: Standard for Telecommunications and Information Exchange Between Systems – Local Area Metropolitan Area Networks – Specific Requirements – Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPAN), http://www.ieee802.org/15/pub/TG4.html
[2] IPv6 over Low power WPAN WG, https://datatracker.ietf.org/wg/6lowpan/charter
[3] F. T. Mamo, Specification, *Implementation and Test of a 6LoWPAN Based Application Demonstrator for Distributed Home and Building Automation*, Master thesis, March 2015.
[4] http://www.zigbee.org
[5] http://www.etsi.org/technologies-clusters/technologies/m2m
[6] http://openmobilealliance.hs-sites.com/lightweight-m2m-specification-from-oma
[7] http://www.eebus.org
[8] https://opcfoundation.org/
[9] http://www.onem2m.org
[10] http://openmobilealliance.org/about-oma/work-program/device-management/
[11] http://www.broadband-forum.org
[12] http://eclipse.org/om2m
[13] https://projects.eclipse.org/projects/technology.wakaama
[14] http://people.inf.ethz.ch/mkovatsc/erbium.php
[15] http://people.inf.ethz.ch/mkovatsc/copper.php
[16] https://www.wireshark.org
[17] http://www.getpostman.com