Remote Networking Technology for IoT: Cloud-based Access for AllJoyn-enabled Devices

Pavel Masek, Radek Fujdiak, Krystof Zeman, Jiri Hosek
Brno University of Technology
Brno, Czech Republic
{masekpavel, fujdiak, hosek}@feec.vutbr.cz, krystof.zeman@phd.feec.vutbr.cz

Ammar Muthanna
State University of Telecommunication
St. Petersburg, Russia
ammarexpress@gmail.com

***Abstract*—The Internet of Things (IoT) represents a vision of a future communication between users, systems, and daily objects performing sensing and actuating capabilities with the goal to bring unprecedented convenience and economical benefits. Today, a wide variety of developed solutions for IoT can be seen through the all industry fields. Each of the developed systems is based on the proprietary SW implementation unable (in most cases) to share collected data with others. Trying to offer common communication platform for IoT, AllSeen Alliance introduced AllJoyn framework – interoperable platform for devices (sensors, actuators, etc.) and applications to communicate among themselves regardless of brands, transport technologies, and operating systems. In this paper, we discuss an application for remote management of light systems built as an extension of AllJoyn Framework – developed application is independent on communication technologies (e.g., ZigBee or WiFi). Besides provided communication independence, the presented framework can run on both major SoC architectures ARM and MIPS. To this end, we believe that our application (available as open source on GitHub) can serve as building block in future IoT / Smart home implementations.**

I. Introduction

The Internet of Things (IoT) is currently a widely used term, which comprises large amount of concepts – Wireless Sensor Networks (WSN), Machine-to-Machine (M2M) communications or Low power Wireless Personal Area Networks (LoWPAN) [1]. According to forecast from leading telecommunication companies as Cisco [2] and Juniper [3], the M2M connections will grow from 495 million in 2014 to more than 3 billion in 2019. Today, we can see a wide variety of smart devices (e.g., smart meters, sensors, actuators) coming on the market in waves and trying to bring intelligent behavior into today's households. Each device is equipped with proprietary software implementation which makes difficult to activate cooperation between group of devices. Hence, there is a crucial need to develop communication platform merging majority of smart devices and so providing desired interoperability.

Question of developing universal platform integrating all sensors and actuators under one system can be further divided into two parts: (i) hardware and (ii) software. Development of universal device called *smart home gateway* (SH-GW) or *Machine-type Communication Gateway* (MTCG) is described in detail in literature [4], [5] – HW board solution capable to control information from sensors and actuators in home is discussed. Dealing with the software implementation, several application frameworks were developed during last few years. Over this time, two frameworks emerged as major platforms (i) *OSGi Framework* [6] and (ii) *AllJoyn Framework* [7]. To provide insight into the main principles of these frameworks, description of those two representatives is given in the following section.

In this paper, we focus on the remote control system for light sources from the viewpoint of developing the extension module for AllJoyn Framework, see Fig. 1 representing the AllSeen Alliance vision of Smart Home [8].

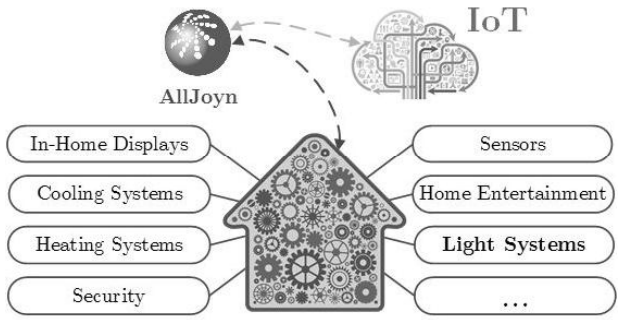

Fig. 1. AllSeen Alliance vision for Internet of Things using the AllJoyn framework – provides a special AllJoyn routing node for secure and manageable connections with external network devices and services in IoT domain

Particularly challenging task in this context, powered by the recent progress in affordable CPU and memory capacity of home IP routers [9], brings the unified functionality of remote controlling for wide spectrum of use-cases in (smart)home / *IoT domain*. Therefore, we introduce the extension for AllJoyn Framework suitable for various communication technologies used in today's households – on the top of it, the functionality was tested on the IP router TP-LINK TL-WDR4300 running OpenWRT 14.07 operating system which can be taken into account as a classical representative of home IP routers. Further, we expand our vision of control system on remote management by using *cloud* entity – enabling the communication between the public network (Internet) and smart devices located behind the edge IP device (creating LAN (Local Are Network) with NAT (Network Address Translation) service).

The rest of the paper is organized as follows. Section II provides the description of available frameworks (including AllJoyn, OSGi) dealing with the application "middle-ware" layer enabling communication between different devices. Further, in Section III we discuss in detail the communication

chain for remote management using the AllJoyn Gateway Agent. Created module (application) for Gateway Agent together with the description of realized testbed scenario are provided in Section IV. Finally, the lessons learned and conclusions are summarized in Section V.

II. Frameworks for IoT domain

It becomes more and more obvious that to meet the *IoT vision*, provided solution capable to act as IoT framework has to enable more than visualization platform or database for obtained M2M data. To provide a comprehensive package for IoT, additional requirements as (i) type o communication (one-directional, bi-directional), (ii) communication protocols (on application and transport layer), and (iii) security (encrypted communication) should be implemented. Today's MTCGs support variety of communication protocols, so called IoT protocols (Constrained Application Protocol (CoAP), Message Queue Telemetry Transport (MQTT), Data Distribution Service (DDS), Session Initiation Protocol (SIP), etc.), which differ in many aspects from legacy transport protocol like a TCP/UDP. In addition to providing basic connectivity, IoT protocols should be able to inter-connect and handle communication between smart devices. Following this fact, two cornerstones enabling the development of applications have been formed over the last decade (i) *OSGi framework* and (ii) *AllJoyn framework* – for a better overview, Table I categorizes frameworks and protocols by most representative TCP/IP layer, where specific platform operates [1].

TABLE I. FRAMEWORKS AND PROTOCOLS CATEGORIZED BY COMMUNICATION TCP/IP LAYERS [1], [10]

Layer	Frameworks/Protocols
Application	IPSO Alliance, Xively, Cumulocity, Thing-Worx, Smart Energy Profile (SEP) 2.0, OSGi Alliance, **AllSeen (AllJoyn)**, Microsoft, IBM, IoTivity, IzoT
Transport	TCP, UDP, WirelessHART, SMS (IMS)
Network	IP, ZigBee

The centralized frameworks mentioned in Table I implement protocol flexibility and usually support application protocols MQTT, REpresentational State Transfer (REST), CoAP and Extensible Messaging and Presence Protocol (XMPP). Table II contains the comparison of the frameworks against common (application) protocols supported.

TABLE II. COMMUNICATION PROTOCOLS SUPPORTED BY THE STUDIED FRAMEWORKS [1], [10]

	MQTT	XMPP	CoAP	REST	Other	Modular
IPSO Alliance			✓	✓		
Xively	✓	✓	✓	✓	✓	
Cumulocity				✓		
Thing-Worx	✓	✓	✓	✓	✓	
SEP 2.0					✓	
OSGi All.					✓	✓
AllSeen All.	✓	✓	✓	✓	✓	✓
Microsoft					✓	
IBM					✓	
IzoT				✓	✓	

Under the "Other column", proprietary protocols (i.e., DDS) are included. Column "Modular" means that the real implementation is left to the developers (selection of application/transport protocol, etc.).

Many of the frameworks are proprietary. On the other hand, most of them support at least one open source messaging protocol. IPSO Alliance, AllJoyn, OSGi and IoTivity are based on open-source technologies. IBM and Microsoft are proprietary. The platforms such as Thing-Worx, Xively and Cumulocity are proprietary and so porting an application from one to another would be a costly exercise.

Some of the frameworks such as, IoTivity, OSGi and AllJoyn support a dual stack implementation, enabling a reduced functionality stack for constrained (embedded) devices. However this is not always the case, such as Xively, Cumulocity and Thing-Worx do not support constrained devices and rely on intermediary agents or gateways - therefore these frameworks do not represent the vision of the comprehensive IoT framework.

Rapid application development, (re-)configurability, scalability and deployment considerations are important characteristics. It is difficult to make evaluation on such aspects, but it is worthy to mention frameworks with comparative strengths. IBM and Microsoft's strong background in enterprise service bus means that they have a good advantage of upscaling as business needs grow. Thing-Worx, Cumulocity and Xively demonstrate strength in rapid application development and focus on value added work. IoTivity, OSGi and AllJoyn tend to focus on customers using commercial *off the shelf* devices and therefore simplify the deployment. Arrowheads strength lies in its re-configurability through the utilization of dynamic orchestration of services and systems.

A. OSGi framework

The Open Services Gateway Initiative (OSGi) [6] attempts to meet requirements as platform and vendor independence as well as architecture openness by providing a managed, extensible framework to connect various devices in a local environments such as home, office, or car. By defining a standard execution environment and service interfaces (framework), OSGi promotes the dynamic discovery and collaboration of devices and services from different sources.

OSGi service platform is an instantiation of a Java virtual machine (JVM) complemented by a set of bundles (packages providing different functionality), see Fig. 2. Running on top of JVM, the framework provides a shared execution environment that installs, updates, and uninstalls bundles without a need to restart the system. Bundles can collaborate by providing other bundles with application components called services [10].

Since frameworks based on OSGi API are highly emerging especially in specific area of IoT (M2M communication), the number of OSGi Frameworks is growing as well. The list of most popular OSGi frameworks contains: (i) Apache Felix (Open source) [11], (ii) Eclipse Equinox (Open source) [12], (iii) Hitachi (Commercial), (iv) Knopflerfish (Open source) [13], and (v) ProSyst (Commercial) [14] – detailed evaluation of those frameworks is provided in [10].

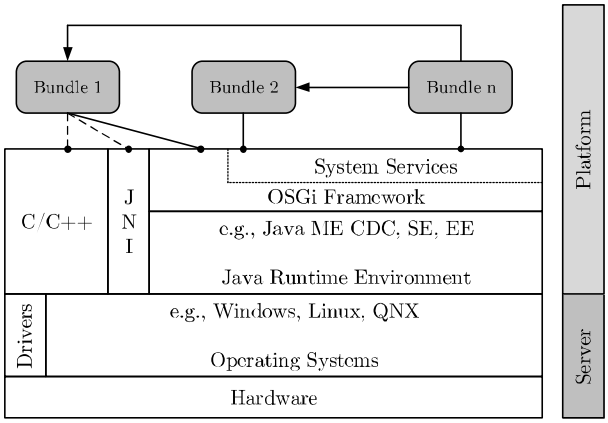

Fig. 2. OSGi Service Gateway Architecture [6]

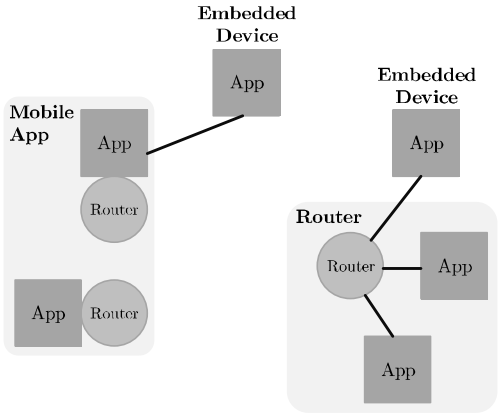

Fig. 3. Network arcitecture of AllJoyn framework [8]

B. AllJoyn framework

AllJoyn represents an open-source software system that provides an environment for distributed applications running across different type of devices and operating systems [7]. AllJoyn is "platform-neutral" framework, meaning it was designed to be as independent as possible of the specifics of the hardware configuration and operating system of the targeted device – framework was developed to run on Windows, Linux, Android, iOS, OS X, and OpenWRT.

1) Apps and Routers: The AllJoyn framework comprises AllJoyn Apps and AllJoyn Routers including mutual communication between both components. Apps can only communicate with other Apps by going through a Router. In real implementation, Apps and Routers can be implemented on the same physical device, or on different devices – three possible topologies exist, see Fig. 3 [7]:

- **App uses its own Router -** in this case, the Router is called a "Bundled Router" as it is bundled with the App. AllJoyn Apps on mobile OS like Android or iOS and desktop OS like Mac OS X and Windows generally fall in this group.
- **Multiple Apps on the same device use one Router -** the Router is called a "Standalone Router" and it typically runs in a background / service process. This is common on Linux systems where the AllJoyn Router runs as a daemon process and other AllJoyn apps connect to the Standalone Router. In case of multiple apps on the same device, the common AllJoyn Router is utilized and, as the result, such device consumes less overall resources.
- **App uses a Router on a different device -** embedded devices (which use the thin variant of the AllJoyn framework, more on this later) typically fall in this camp as the embedded device which typically does not have enough CPU and memory to run the AllJoyn router.

2) Transport technologies: The AllJoyn framework runs on the local network (behind the NAT). It currently supports WiFi, Ethernet, serial, and Power Line (PLC) communication technologies, but since the AllJoyn software was written to be transport-agnostic and since the AllJoyn system is an evolving open-source project, the support for more transport means is scheduled to be added in the future by community.

Additionally, complementary software can be created to bridge the AllJoyn framework to other systems like ZigBee, Z-Wave, or the *cloud*. In fact, AllSeen Alliance is currently working on adding a Gateway Agent as a default AllJoyn service – see Section III for detail description.

C. Software architecture

The AllJoyn Service Frameworks implement a set of common services, like notification, or control panel [7]. By using the common AllJoyn service frameworks, apps and devices can properly inter-operate with each other to perform a specific functionality. The AllJoyn framework provides two possible variants, see Fig. 4:

- **Standard core -** targeted for non-embedded devices (e.g., Android, iOS or Linux).
- **Thin core -** for resource constrained (embedded) devices (e.g., Arduino, Intel Edison, ThreadX).

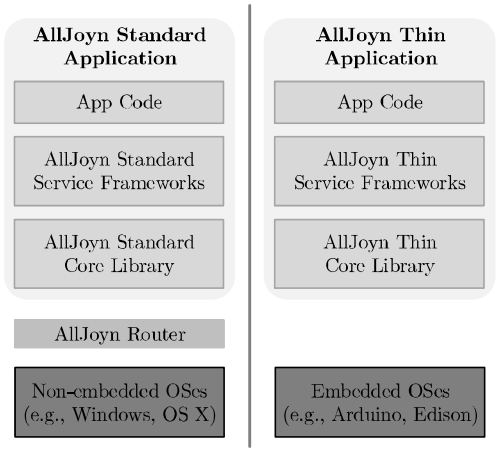

Fig. 4. SW architectures of AllJoyn framework [8]

Typically, developed applications will be written using the AllJoyn Service Framework APIs so that the applications can be compatible with devices using the same Service Frameworks. Only by using AllJoyn Service Frameworks developed by AllSeen Working Groups [15] the application will be compatible with other applications and devices in the AllSeen ecosystem.

III. Communication logic using gateway agent

In our created application, we have focused our attention to extend the functionality of AllJoyn framework. The *connector* for communication between internal distributed communication bus (DBus) and remote devices located outside the home (local) network was developed.

To meet the required remote and secure connection, the node called *Gateway Agent* was defined as a special AllJoyn routing node for secure and manageable connections with external network devices and services, see Fig. 7. AllSeen Working Group [15] is currently working on adding a Gateway Agent as a standard AllJoyn service. However, since the implementation is still in progress, we have created our own instance of Gateway Agent – source codes are available on GitHub [16]. Key features of the Gateway Agent are listed bellow:

- **Remote access and management -** provides a standard and secure method for connecting local AllJoyn devices and applications to one or more external services through which users and service providers can access and manage AllJoyn-enabled devices and applications.
- **Security and data privacy -** AllJoyn's end-to-end encryption keeps communications secure. Fine-grained controls let users decide which AllJoyn enabled devices and applications have access to and from user-approved cloud-based services.
- **Interoperability -** a standard API extends the interoperability already provided by AllJoyn to fully embrace external networks and remote services, by supporting plug-in protocol connectors into a standard AllJoyn Gateway Agent API enabling secure inter-networking to cloud services, PAN and wireless technologies (including Bluetooth, ZigBee and Z-Wave) and other networks.
- **Support for open standards -** connectors provide connectivity, interaction and integration over a variety of protocols including REST, XMPP, MQTT and TR-069.

Communication scheme of created solution is depicted in Fig. 7. We have selected the IP router TP-Link TL-WDR 4300 as a representative of "low-cost" solution for households capable to run OpenWRT operating system - in our work, OpenWRT Barrier Breaker 14.07 was used. For purpose of compilation of various packages, we extended the internal storage by FLASH storage (using the extroot command). The installed software part provided by AllJoyn framework is represented by (i) *AllJoyn daemon* (basic shared bus; run on startup), (ii) *Lightning controller service* (management, discovery and control of light sources), and *AllJoyn Gateway Management App* (management of connectors) – the selected modules during the compilation process (*make menuconfig* command) on the TP-Link router are displayed in Fig. 5.

```
--- alljoyn.............................. AllJoyn Peer-to-Peer networking
-*-   alljoyn-about.............. AllJoyn - About service library (deprecated)
< >   alljoyn-c.................................... AllJoyn - C binding
-*-   alljoyn-config.......................... AllJoyn Config service library
< >   alljoyn-controlpanel................. AllJoyn ControlPanel service library
<*>   alljoyn-gwagent.............................. AllJoyn Gateway Management
<*>   alljoyn-lighting..................................... AllJoyn Lighting
< >   alljoyn-non-gw-config.............. AllJoyn - alternate non-Gateway Config
-*-   alljoyn-notification.............. AllJoyn Notification service library
< >   alljoyn-sample_apps... AllJoyn services sample_apps - sample applications
< >   alljoyn-samples.............................. AllJoyn - testing samples
-*-   alljoyn-services_common......... AllJoyn Services Common service library
```

Fig. 5. Selection of AllJoyn modules during the compilation of OpenWRT OS for IP router TP-Link TL-WDR 4300

Mentioned AllJoyn modules enable the space for development of new services. In our project, we focus on development of connector which allows remote management of light sources. This connector listens on distributed AllJoyn DBus and in parallel is connected to remote control application – in our case represented by Twitter API.

IV. Implementation of remote management

A. AllJoyn-enabled light bulbs application

We have decided not to test the created connector with real devices (rely on specific communication technology) but we have verified all functions with our software application (for Android version 5.0 (API 21) and higher; tested on Samsung Galaxy S4) in role of virtual light bulb. Created tool simulates a lamp running the AllSeen Alliance's Open Source AllJoyn Lighting Service Framework (LSF) including an embedded instance of Lighting Controller Service, see Fig. 6. Application exposes Lighting Controller Service API's that can be controlled via the AllJoyn-based Lighting SDK [7].

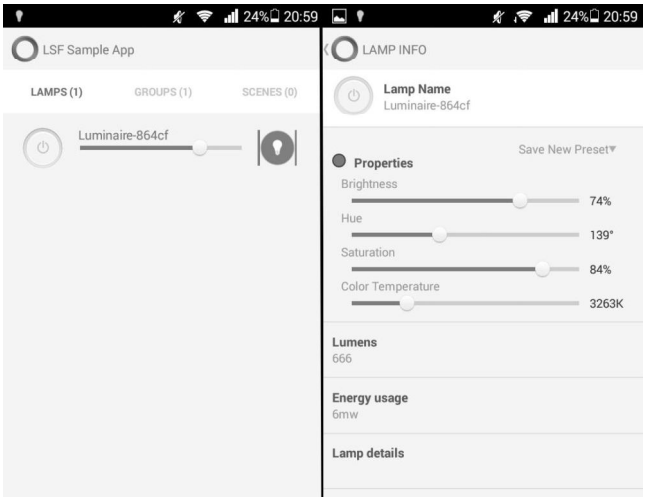

Fig. 6. Updated LSF application by adding capabilities to communicate with device located outside the home network. Initial functionality allows internal communication only. Source codes for simple LSF application are provided by AllSeen Alliance

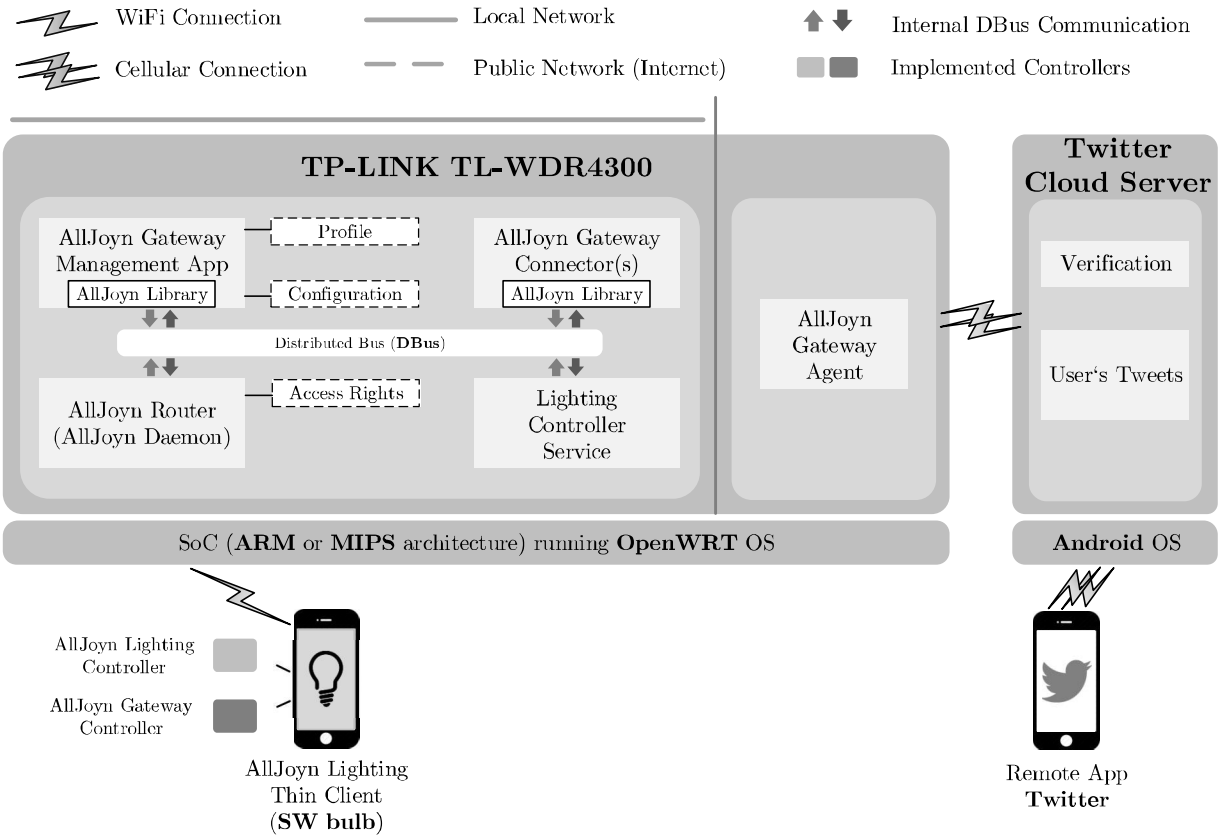

Fig. 7. Considered communication structure using the AllJoyn Gateway Agent

B. Gateway agent connector

Connector for Gateway Agent was written in *C++* programing language together with additional scripts created in *Bash*. As the first step, new distributed bus (DBus) for the purpose of connector was created – this allows the interconnection with the main DBus defined by AllJoyn framework.

Listing 1: New DBus

```
BusAttachment bus ("ConnectorApp", true);
QStatus status = bus.Start();
status = bus.Connect();
```

Event handling is managed by the method "LampStateChangedCB" in class "LampManagerCallbackHandler". This method is activated when the bulb's status is changed.

Listing 2: Control Class

```
...
ControllerClientCallbackHandler controllerClientCBHandler;
LampManagerCallbackHandler lampManagerCBHandler;

ControllerClient client (bus, controllerClientCBHandler);
LampManager lampManager (client, lampManagerCBHandler);
ControllerClientStatus cstatus = client.Start();
   while (!connectedToControllerService) {
      printf ("waiting");
      sleep (1);
   }

cstatus = lampManager.GetAllLampIDs();
cstatus = CONTROLLER_CLIENT_OK;
...
```

C. Application for remote control

As an application platform for remote controlling, we have decided to use one of the most popular social application *Twitter*. Following the requirements given by the Twitter REST API [17], the message length is restricted to 140 characters. To meet this condition, we have created the new message format for our application:

Listing 3: Message Format

```
AllJoynL<command> lampID=<id> onOff=<state> hue=<state>
saturation=<state> colorTemp=<state> brightness=<state>
```

- AllJyounL - determinates whether it is an incoming command to get status (info) of bulb(s) (AllJoynLS) or command to change parameters (AllJoynLC).
- lampID - defines the bulb ID (uint32_t).
- onOff - bulb status (i) On (1), (ii) Off (0).
- hue - hue definition in range 0 to 360.
- saturation - bulb saturation in percents.
- colorTemp - light temperature defined in Kelvin in range 2700 to 5500.
- brightness - key parameter of brightness, defined in percents.

The example of running Twitter application is depicted in Fig. 8. Messages are sent to the Gateway Agent where the created connector is processing the received data. Further, data

is sent via the DBus to Alljoyn router daemon. In the final phase, data is sent over the used communication technology (in our case WiFi) to destination node (lamp represented by SW tool), see Fig. 9.

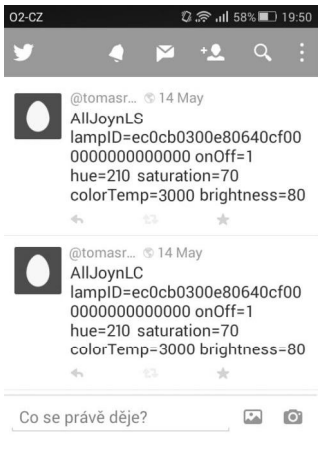

Fig. 8. Example of "tweets" with created message format

Fig. 9. Processed "tweets" on the side of SW bulb

V. Lessons Learned and Conclusions

In this final section, we discuss the important aspects that we have been facing during the prototype implementation, as well as outline our conclusions. Within the process of development of the connector for AllJoyn Gateway Agent, we have addressed a number of challenges – mostly due to the fact, that structure (modules and classes) of AllJoyn framework was changing frequently (development procedure on monthly basis). Therefore, we had to create our code as general as possible.

In particular, we had to adapt to the fact that core AllJoyn framework is written in C++ programing language. On the other hand, the thin client (our created software bulb) has to be written, due to several limitations from the side of AllJoyn framework, in C. Although the AllJoyn framework and two additional modules (AllJoyn Gateway Agent and AllJoyn Lighting Service Framework (LSP)) are available for compilation on IP routers running OpenWRT this is not sufficient for our use case. Due to the missing integration between mentioned parts, there is a need to create an extension of AllJoyn Gateway Agent – communication bridge for internal and external network. Our developed extension comprises all mentioned parts and enables possibility to build our created Gateway Agent connector with libraries provided as a part of AllJoyn framework.

Our main and the most essential learning while working with the AllJoyn framework is such that source codes, pre-compiled and distributed as AllJoyn framework and provided directly by AllSeen, are ready to be operated on today's IP routers running OpenWRT operating system. Further, we proved that framework is truly platform independent – both major architectures of IP routers available on market (ARM and MIPS) are supported. In our case, we have used the TL-WDR4300 router form TP-LINK running latest OpenWRT 14.07 (Barrier Breaker) - MIPS architecture.

To prove the functionality of developed *connector*, we have created the software tool that simulates a lamp running the AllSeen Alliance's Open Source AllJoyn Lighting Service Framework. Application was controlled remotely (from external network) by commands sent via Twitter. To this end, we implemented new message format for sending commands "tweets" (similar to *getters* and *setters*) – limited messages to length 140 characters. Since the created solution is network-independent, our connector / SW lamp can run over Wi-Fi, Wi-Fi Direct, Ethernet, PLC, Bluetooth, and other communication technologies.

Finally, we can conclude that today's "low-cost" IP routers (within the price range 50 - 70€) have already reached the computational threshold required to run both widespread platforms for IoT – *OSGi framework* and *AllJoyn framework*. As sponsored member of AllSeen Alliance [8], we believe our created extension of Gateway Agent within the AllJoyn framework can be used (and modified) by community members.

Acknowledgment

The described research was supported by the National Sustainability Program under grant LO1401. For the research, infrastructure of the SIX Center was used.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications." *Communications Surveys & Tutorials*, IEEE, 17(4), 2347-2376.

[2] Cisco, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update", 2014-2019, February 2015.

[3] M. S. Whitcup and K. LaMattina, "Juniper What is Inhibiting Growth in the Medical Device Wearable Market?", September 2014.

[4] P. Masek, J. Hosek, D. Kovac, F. Kropfl, "M2M Gateway: The Centerpiece of Future Home." *In 2014 6th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. St. Petersburg, Russia (pp. 286-293).

[5] J. Hosek, P. Masek, D. Kovac, M. Ries, F. Kropfl, Universal Smart Energy Communication Platform. *In 2014 International Conference on Intelligent Green Building and Smart Grid (IGBSG)*. 1. Taipei, Taiwan: IEEE, 2014. pp. 1-4. ISBN: 9781467361217.

[6] OSGi Alliance [Online]. Available from: http://www.osgi.org.

[7] AllJoyn Framework [Online]. Available from: http://bit.ly/1OA9nNG.

[8] AllSeen Alliance [Online]. Available from: https://allseenalliance.org/.

[9] J. Hosek, P. Masek, D. Kovac, M. Ries, F. Kropfl, "IP Home Gateway as Universal Multi- Purpose Enabler for Smart Home Services." *Elektrotechnik und Informationstechnik VE - Verbandszeitschrift*, 2014, roc. 131, c. 5, pp. 1-6. ISSN: 0932-383X.

[10] M. Stusek, J. Hosek, D. Kovac, P. Masek, P. Cika, F. Kropfl, "Performance Analysis of the OSGi-based IoT Frameworks on Restricted Devices as Enablers for Connected-Home." *In 2015 7th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. Brno, Czech Republic: 2015. pp. 211-216. ISBN: 978-1-4673-9282-2.

[11] Apache Felix: OSGi Framework and Service Platform [Online]. Available from: http://felix.apache.org/

[12] Eclipse Equinox: OSGi core framework [Online]. Available from: http://www.eclipse.org/equinox/

[13] Knopflerfish, Open Source OSGi SDK [Online]. Available from: http://www.knopflerfish.org/

[14] ProSyst: Bosch Group [Online]. Available from: http://www.prosyst.com/startseite/

[15] AllSeen Alliance - Working Groups [Online]. Available from: http://bit.ly/1SGYL5M

[16] GitHub: AllJoyn - Gateway-Agent-Connector. Available from: http://bit.ly/1nbCHD4

[17] Twitter - REST APIs [Online]. Available from: http://bit.ly/1IGg3tP