SERBIAN JOURNAL OF ELECTRICAL ENGINEERING
Vol. 17, No. 1, February 2020, 83-94

UDC: 910:004.65]:728.1 **DOI: https://doi.org/10.2298/SJEE2001083M**

Using Online Third Party Geolocation Services to Improve Smart Home User Experience

Milica Matić[1], Milan Tucić[2], Marija Antić[1], Roman Pavlović[2]

Abstract: This paper presents one approach to improve smart home user experience, by providing location-based services. Within the cloud of the existing smart home solution, the geolocation micro-service is first implemented. This micro-service communicates with third party online geolocation service to obtain latitude and longitude values from physical address and vice versa. The geolocation micro-service is used by different services to provide location-based experience. For example, weather micro-service uses the geolocation information, to provide weather content, while the traffic service monitors traffic conditions in the vicinity of the household. This information can be used to setup different automation rules in the system.

Keywords: Smart home, cloud, Online third party services, Geolocation, Weather.

1 Introduction

Home automation (HA) systems have become very popular and widespread over the past few years. Various sensors and actuators are already available on the market and can be connected to communicate among each other to create home automation systems. Systems like this represent one of the growing areas of IoT consumer applications [1−3].

Usually, home automation systems consist of four main parts – end devices, central controller, cloud, and user applications. The central controller controls the local network of end devices, cloud enables the remote control of the system and user applications allow users to control their system. Typically, commands in the HA system are issued by the end user via the user application. However, these commands can also be automated by the set of the predefined rules. Rule execution can be triggered by the activation of other device in the system [4]. For example, the user could create a rule which would turn on lights when the motion in the room is detected.

[1]Faculty of Technical Sciences, University of Novi Sad, 21000 Novi Sad, Serbia;
E-mails: milica.matic@rt-rk.uns.ac.rs; marija.antic@rt-rk.uns.ac.rs
[2]OBLO Living, Narodnog fronta 21a, 21000 Novi Sad, Serbia;
E-mails: milan.tucic@obloliving.com; roman.pavlovic@obloliving.com

In this paper we presented one way to improve smart home user experience, by integrating geolocation and weather services into existing smart home system [7].

Geolocation information in the IoT has been used as a method to the power consumption, by defining a boundary region surrounding power-consumption structure [8], as well as determining the location of the device in the network [9]. Also, it is possible to create security rules in the system depending on the location information [10].

Measurements of the current sunlight and temperature have been used to control power consumption in HA systems, and effort has been made to design algorithms to control this process automatically [11 – 13]

We created the system where HA central devices are abstracted by their physical location and receive different data depending on their location. The user can subscribe to receive messages which are important for the country his gateway is located in. For example, the system can send yellow warning messages for different countries. These messages will be broadcasted to every gateway in the specific country. Also, if the user is subscribed to receive messages sent to gateways in the specific locality in the city, it will be able to receive traffic information and warnings, such as crashes or traffic jams.

One use case of geolocation service is presented – integration of third party online weather service. HA cloud communicates with the third party online services in order to obtain location and weather data for the specific HA system. By combining this data, we created the system in which the user can monitor weather conditions via the HA user application. Furthermore, the user can create rules which use weather information as triggers. For example, all lights can be turned on at sunset. Also, the system can be set up to send notifications based on the location and weather information. Therefore, the user can be notified if it is raining outside or if the storm is close and similar.

This implementation represents the extension of the already existing HA system. Therefore, in Section 2 we will explain the architecture of the targeted system and the communication between main system components. Section 3 and Section 4 give details about the implementation of geolocation micro-services and its client micro-services within the HA cloud, respectively. Finally, functional and load testing of the system is performed and the results are presented in Section 5.

2 System Architecture

The existing smart home system [1, 5] allows users to connect end devices which are using different communication protocols (Zigbee, Z-Wave, ONVIF or IP). The central component of the system is the home automation gateway.

The gateway creates a level of abstraction, so that all end devices in the system are represent in the same way, by sets of their functionalities and characteristics [1]. This allows applying the same control logic to all of end devices, while being agnostics of the communication protocols end devices are using [6]. One of the main responsibilities of the home automation gateway is to forward commands from user application to end devices. Also, it provides the user with the real time information about the state changes of devices in the system.

User application allows users to monitor and control devices in the smart home system. It provides an intuitive UI to all of the gateway's functionalities, while hiding the complexities of the underlying technologies. It can be directly connected to the gateway in the local network, but in order to enable remote control, user application has to be connected to the home automation cloud.

The cloud component of the system acts as a connection between the gateway and the remote end user. It provides remote system control, as well as a number of other functionalities such as gateway firmware update, system usage tracking and integration with different third party services. The cloud has a micro-service architecture where every service is clearly decoupled and in charge of different functionalities. For communication between different micro-services HTTP protocol is used. Also, HTTP REST APIs are used to integrate with various third party services, such as video surveillance systems, voice control (Amazon Alexa, Google Home), etc.

2.1 Communication between system components

Main system components communicate using Message Queuing Telemetry Transport (MQTT) protocol, as well as HTTP protocol as depicted in Fig. 1.

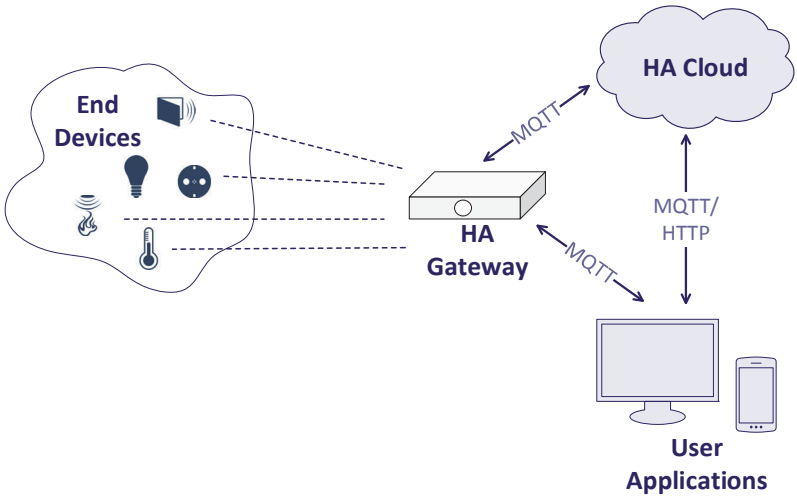

Fig. 1 – *Top level system architecture.*

The universal messaging model for communication based on MQTT protocol is applied to all system components. MQTT topic structure and message payloads are defined on system level. Every component in the system (gateway, mobile application, web application and cloud micro-services) represents one MQTT client, and MQTT broker is responsible for handling different messages. The broker defines access rights to the resources, as well as the topics that clients are allowed to publish messages to, based on the client type.

There are three different message types exchanged between MQTT clients:

1. Request messages
2. Response messages
3. Event messages

Request messages are used to send commands or queries. For example, the mobile application can send a request message to the gateway to change device state. Response messages are used to respond to request messages. Every request message requires a response, which carries either a confirmation message or the requested information. Event messages are used to asynchronously report different changes in the system (device property changes, alarm activates, etc.).

In this paper, the communication between the cloud micro-services and the gateway is of interest. Topics that MQTT messages are published on contain the following information:

– Gateway and user ID

– Service name

– Message type

Gateway ID and user ID are identifiers that uniquely represent the specific gateway and the user that gateway belongs to. Service name is the name of the cloud micro-service that communicates with the gateway. Message type can be one of the following: request, response or event.

Regardless of their type, MQTT messages contain the message ID parameter and a variable number of optional parameters as presented in **Table 1**.

Table 1
MQTT message payload.

Field name	Description	Required
ID	Unique message identifier	Yes
Parameters	Message parameters	No

Message parameters depend on the message type. Common parameters for every type of message are presented in **Tables 2**, **3** and **4**. Every request message contains the information about the message sender and the message type (defining if it is a command or a query request), as well as the message name. As a response to a command or query request, in the response message the client sends a status code, which may be accompanied by the additional information in the description. Finally, event messages carry the event name and, optionally, additional information to describe the change more closely.

Table 2
Request message parameters.

Field name	Description	Required
Sender	Message sender	Yes
Type	Message type (command or query)	Yes
Name	Message name	Yes

Table 3
Response message parameters.

Field name	Description	Required
Code	Request processed successfully or not	Yes
Description	Additional information	No

Table 4
Event message parameters.

Field name	Description	Required
Name	Event name	Yes
Description	Additional information	No

3 Geolocation Service

Various third party services, such as online weather service, rely on user's location in order to provide the useful information. To be able to communicate with similar third party online services, we implemented the geolocation micro-service within the home automation cloud. This micro-service is responsible for binding the home automation gateway with its physical address.

In order to obtain the relevant location information from the user, several different geolocation techniques can be used:

1. GPS location – satellite-based radio navigation system that provides geolocation information;
2. Mobile cell information – using triangulation algorithm based on cell tower information, where cell towers in exact locations are well known; this method is less precise than GPS, but it is available to devices that do not have GPS receivers and where GPS is not available;
3. WiFi network information – Internet services constantly update WiFi network information from routers all over the world; these services match GPS data and available WiFi networks information sent from mobile phones, thus calculating precise location of each WiFi router;
4. IP address – user's location can be fetched based on the device IP address.

The listed techniques are already used by mobile phones and browsers to estimate the user's location during the home automation system setup. In addition to those techniques, user can provide address manually if estimated address is not precise enough.

Geolocation micro-service is in charge of mapping the physical address to the latitude and longitude values, and vice versa. It communicates with the third party geolocation cloud, which provides geocoding and reverse geocoding functionalities [14, 15]. On the other hand, geolocation micro-service communicates with the gateway to provide it with location information. Top level system architecture is presented in Fig. 2.

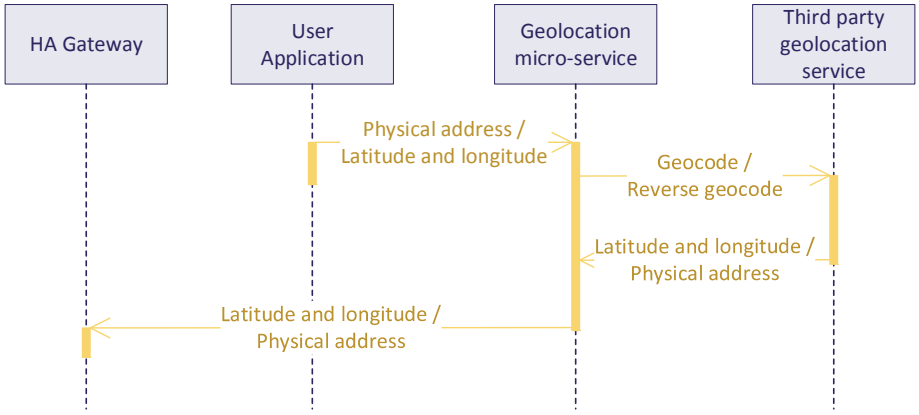

Fig. 2 – *System architecture with geolocation micro-service.*

Geolocation micro-service obtains gateway's physical address via one of the techniques listed above and sends it to the third party cloud for geocoding. Once the address is geocoded, and the latitude and the longitude values are obtained, the structure presented in Fig. 3 is created and stored into database.

Types of locations are obtained from the third party geolocation service. Relations are defined in a way that the wider location is the parent of the narrower location (a country will be a parent of a city; a city will be a parent of a street, etc.).

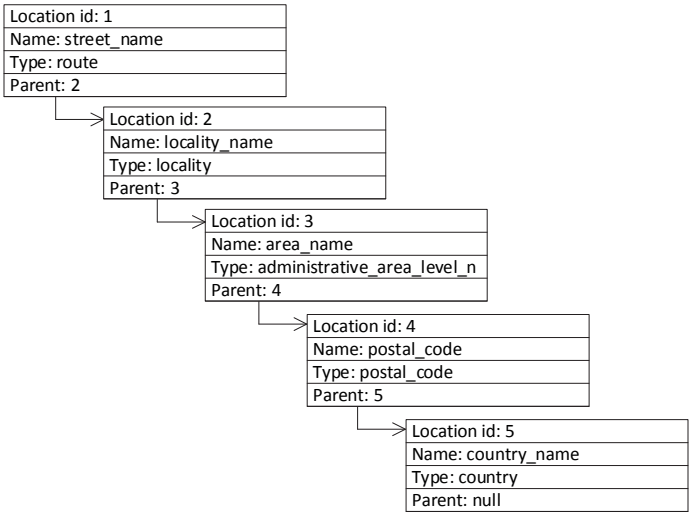

Fig. 3 – *Relations between location structures stored into database.*

Once gateway has received its location, it can request different data based on it. The MQTT broker generates topics for the communication by the following pattern:

– User id

– Service name

– Gateway id

– Location name.

For example, a message sent to the topic <u1/traffic/g1/Liman3> is intended for the micro-service named *traffic*. It is sent from gateway with id *g1* which belongs to user with id *u1* to ask for traffic data for the location – *Liman3*.

When the HA cloud micro-service receives a message on the specific topic, it is necessary to parse the topic for the relevant data. Once user and gateway ids are obtained, location has to be parsed. Geolocation micro-service stores location data into database with structure presented in Fig. 3. Every location is aware of its parent location. Therefore, if a gateway is subscribed to receive messages for the specific street name, it will be subscribed to messages intended for gateways in country that street belongs to.

Once we introduced gateway recognition by address, the next logical step is to integrate the HA system with other location-based services, and to enable sending notifications for specific locations, as well different types of alarms based on the data from third party services, such as weather and traffic services.

4 Location Based Services

In this chapter, we will briefly explain the implementation of location based micro-services. These services are implemented within the home automation cloud, as well as the geolocation micro-service, and rely on the information provided by the geolocation service. They use the location information as a parameter when requesting location-specific data from the third party online service.

On the gateway side, the information that location based services collect is incorporated into virtual devices. The virtual device is presented to the user in the same way as physical devices – as a set of services and properties. It is listed in user application among other physical devices, as well. For example, the weather device will display the information about current temperature, wind and humidity. This information can also be used to create automation rules on the gateway side, i.e. it is possible to send the notification if the windows are open and the rain is starting.

Gateway asks for new data from the location based micro-services in three cases:

1. Gateway has come online
2. Gateway's location has changed
3. Refresh period has expired
4. User has requested information via user application.

If the gateway was offline for some time, the first time it comes online, new data should be obtained. In the second case, when the user provides the new gateway address, the data has to be updated accordingly. The user can configure how often the gateway will update the data. The update interval is a configuration parameter of the virtual devices, and is referred to as refresh period. Upon its expiration, the gateway will request new data.

When the gateway comes online for the first time, it will ask for its location information. Depending on the actual micro-service and its application, the gateway can ask for the identifier of different location types, e.g. the weather information should be requested for the city, while the traffic information should be requested for the area where the household is. Once the gateway is aware of its location, it subscribes to the appropriate MQTT topic, and sends the MQTT request to the desired location based micro-service to get the data. The MQTT topic for location based requests has to contain the name of the location

data is requested for, as well as the information about gateway, user and cloud micro-service name the message is being sent to.

Within the HA system, gateways can also be grouped by their location. This allows administrators to send notifications to different gateway groups, based on their location. For example, power or water outages can be reported to the users, based on their address.

Usually, location-based services communicate with third party online services to gather information. With a goal to minimize the number of request being sent to third party services, once the data is obtained, it is saved to the cache database within the HA system, for a predefined period of time. Once this period expires, the data is removed from the cache. Therefore, when a micro-service receives the request for data from the gateway, it will first check if the cached information exists. If the cache entry exists, and it has not expired yet, the micro-service will return this data as a response to the gateway. However, if the data in the cache is not valid, the location-based micro-service contacts the proper third party service to fetch new data. Once the data is obtained, it is sent as response message to the gateway. Also, in order to minimize the number of request messages sent by gateways in the same location, this information is transmitted to the other gateways in the same location. This means that, after sending a response message to the gateway that requested new data, the micro-service will also send the MQTT event message containing new data to all of the gateways in the same location.

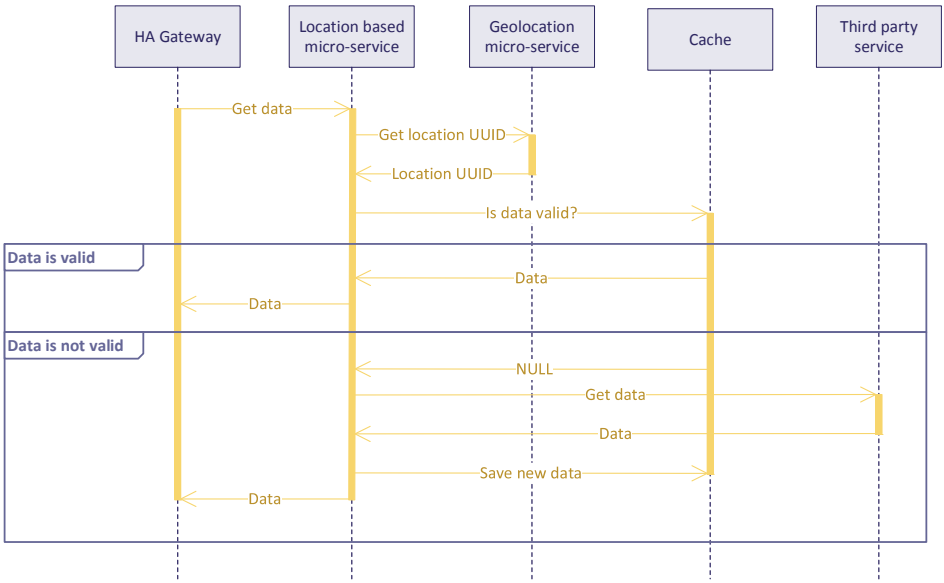

Fig. 4 – *Obtaining data from location based micro-services.*

The communication flow between the home automation gateway, location-based micro-service, cache and the third party online service is depicted in Fig. 4.

5 Experimental Evaluation

As experimental evaluation of proposed solution required using HA gateways located in different locations around the world, we used virtual gateway models for testing. Virtual gateway model is a software model of the real HA gateway created for system testing. This model is capable of communicating with user applications and HA cloud via the MQTT protocol. It is possible to configure virtual models to support various end devices and rules. The existing model is extended to support geolocation and weather services.

For testing purposes, 4000 virtual gateways, all located in different locations, were configured to ask for weather data in the same time.

First, we tested the performance of Redis database which was used as weather cache database. We observed memory consumption and CPU load of the Redis database instance. Average results are depicted in Fig. 6. Results show that CPU load does not exceed the value of 20% and memory consumption is beneath 1 GB even when 8000 different weather information entries are saved to database.

Next, we tested the system with the real gateway. We measured the time taken to get location information from third party online service and from database, as well as time taken to get weather information, both from third party online service and from weather cache. Results showed that on average it takes 910.89 ms to get geolocation information if data is obtained from third party service, and 39,43 ms when data is read from database. Obtaining weather data from third party service takes 222.11 ms on average, compared to 14.45 ms required to get the data from weather cache.

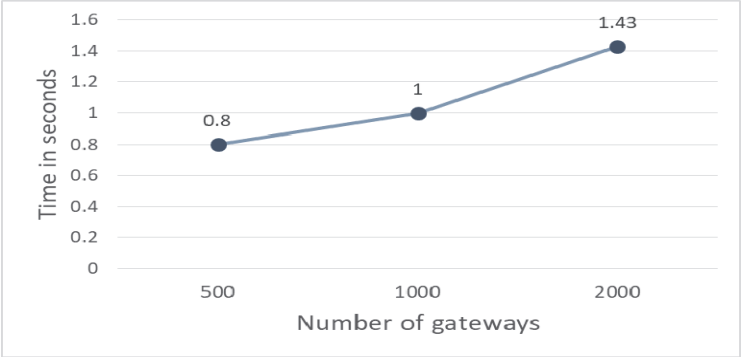

Fig. 5 – *Time taken for all of the gateways in the same location to receive message.*

Finally, we tested the HA cloud ability to process messages for fetching geolocation data for different locations, and to send responses and events to all of the gateways in the specific locations. We created a setup where 1000 gateways are located in the same country, but different cities. On average, it takes approximately 200 ms to send MQTT events to one gateway, a propos 1 s which is needed to find all gateways in provided country and send event messages to all of them. Same tests were conducted on sets of 500, 1000 and 2000 gateways, respectively. Results are depicted in Fig. 5 and Fig. 6.

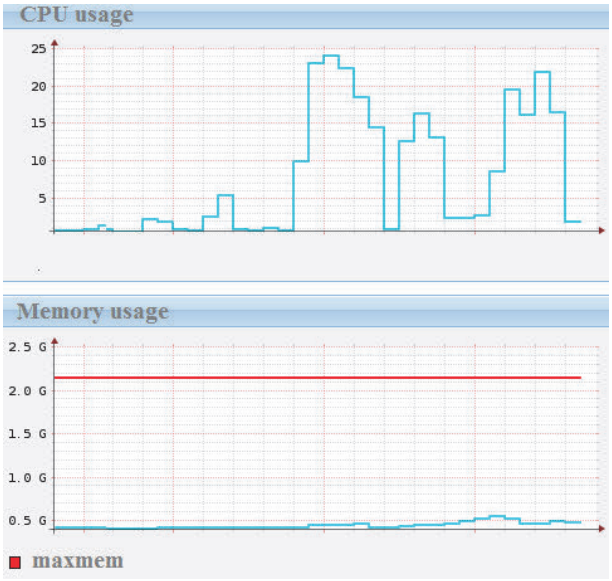

Fig. 6 – *Average memory consumption and CPU load on Redis instance.*

5 Conclusion

In this paper we presented one solution for integrating third party online services within the existing smart home system. This approach improves the smart home user experience by enabling users to monitor different data for their households and act upon it. User can monitor weather data for the city it is situated in, or set automated rules depending on the received data. Also, solution is extendable in a way that different location-based services can be added.

6 References

[1] A. J. Bernheim Brush, B. Lee, R. Mahajan, S. Agarwal, S. Saroiu, C. Dixon: Home Automation in the Wild: Challenges and Opportunities, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'11), Vancouver, Canada, May 2011, pp. 2115 – 2124.

[2] S. H. Park, S. H. Won, J. B. Lee, S. W. Kim: Smart Home – Digitally Engineered Domestic Life, Personal and Ubiquitous Computing Journal, Vol. 7, No. 3-4, July 2003, pp. 189 – 196.

[3] M. Soliman, T. Abiodun, T. Hamouda, J. Zhou, C.- H. Lung: Smart Home: Integrating Internet of Things with Web Services and Cloud Computing, Proceedings of the IEEE 5th International Conference on Cloud Computing Technology and Science, Bristol, UK, December 2013, pp. 317 – 320.

[4] I. Lazarević, M. Sekulić, M. S. Savić, V. Mihić: Modular Home Automation Software with Uniform Cross Component Interaction Based on Services, Proceedings of the IEEE 5th International Conference on Consumer Electronics - Berlin (ICCE-Berlin), Berlin, Germany, September 2015, pp. 363 – 365.

[5] M. Pandurov, I. Lazarević, R. Pavlović, N. Smiljković: Unified Device Access in Home Automation Environment, Proceedings of the 22nd Telecommunications Forum (TELFOR), Belgrade, Serbia, November 2014, pp. 971 – 974.

[6] M. Tucić, V. Moravčević, G. Velikić, Đ. Sarić, V. Mihić: Device Abstraction and Virtualization: Concept of Device in Device, Proceedings of the IEEE 5th International Conference of Consumer Electronics – Berlin, Berlin, Germany, September 2015, pp. 431 – 434.

[7] M. Pandurov, I. Lazarević, R. Pavlović, N. Smiljković: Unified Device Access in Home Automation Environment, Proceedings of the 22nd Telecommunications Forum (TELFOR), Belgrade, Serbia, November 2014, pp. 971 – 974.

[8] D. S. Benco, P. S. Beck: Method and System for Managing Power Consumption Using Geolocation Information, US Patent, No. US 2011/0153525 A1, June 2011.

[9] M. Smith, G. Maguire: Location Requests by a Network Device, US Patent, No. US 2004/0080412 A1, April 2004.

[10] S. Zellner: Location-Based Security Rules, US Patent, No. US 7428,411 B2, September 2008.

[11] H. Tischer, G. Verbic: Towards a Smart Home Energy Management System – A Dynamic Programming Approach, Proceedings of the IEEE PES Innovative Smart Grid Technologies, Perth, Australia, November 2011, pp. 1 – 7.

[12] T. Zhu, A. Mishra, D. Irwin, N. Sharma, P. Shenoy, D. Towsley: The Case for Efficient Renewable Energy Management in Smart Homes, Proceedings of the 3rd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings, Seattle, USA, November 2011, pp. 67 – 72.

[13] M. Khan, B. N. Silva, K. Han: Internet of Things Based Energy Aware Smart Home Control System, IEEE Access, Vol. 4, October 2016, pp. 7556 – 7566.

[14] X. Ge: Address Geocoding, US Patent, No. US 6,934,634 B1, August 2005.

[15] M. Zarem, E. Vuillermet, J. DeAguiar: Intelligent Reverse Geocoding, US Patent, US 8,731,585 B2, May 2014.

[16] P. Colombo, E. Ferrari: Access Control Enforcement within MQTT-Based Internet of Things Ecosystems, Proceedings of the 23rd ACM on Symposium on Access Control Models and Technologies, Indianapolis, USA, June 2018, pp. 223 – 234.