
**Open Archive Toulouse Archive Ouverte**

OATAO is an open access repository that collects the work of Toulouse researchers and makes it freely available over the web where possible

This is an author's version published in:
http://oatao.univ-toulouse.fr/22416

**Official URL**

DOI : https://doi.org/10.1007/978-3-319-94523-1_21

**To cite this version:** Van Den Bossche, Adrien and Gonzalez, Nicolas and Val, Thierry and Brulin, Damien and Vella, Frédéric and Vigouroux, Nadine and Campo, Eric *Specifying an MQTT Tree for a Connected Smart Home.* (2018) In: International Conference On Smart homes and health Telematics (ICOST 2018), 10 July 2018 - 12 July 2018 (Singapore).

Any correspondence concerning this service should be sent to the repository administrator: tech-oatao@listes-diff.inp-toulouse.fr

# Specifying an MQTT Tree for a Connected Smart Home

Adrien van den Bossche[1(✉)], Nicolas Gonzalez[1], Thierry Val[1], Damien Brulin[2], Frédéric Vella[1], Nadine Vigouroux[1], and Eric Campo[2]

[1] IRIT, CNRS, UPS, UT1, UT2J, Université de Toulouse, Toulouse, France
{Adrien.van-den-Bossche,Nicolas.Gonzalez,Thierry.Val,Frederic.Vella,
Nadine.Vigouroux}@irit.fr
[2] LAAS-CNRS, Université de Toulouse, CNRS, UT2J, Toulouse, France
{Damien.Brulin,Eric.Campo}@laas.fr

**Abstract.** Ambient Assisted Living (AAL) represents one of the most promising Internet of Things applications due to its influence on the quality of life and health of the elderly people. However, the interoperability is one of the major issues that needs to be addressed to promote the adoption of AAL solutions in real environments, and to find a way of common exchange between the available connected tools to share the data exchanged. This article will present software buses needs and specify an API based on a MQTT software bus treelike architecture. An example is given to illustrate the efficiency of the API developed in a smart home.

**Keywords:** Ambiant assisted living · Software bus · MQTT
Connected devices · Elderly people · Smart home

## 1 Introduction

Ambient Assisted Living (AAL) represents one of the most promising Internet of Things (IoT) applications due to its influence on the quality of life and health of the elderly people [1]. AAL can be used for preventing, curing, and improving wellness and health conditions of older adults [2]. IoT consists of connected smart objects capable of identifying, locating, sensing and assisting. [3] gives a survey on the emergence of AAL tools for older adults based on an ambient intelligence paradigm. The technologies mentioned include smart homes (ambient sensors), wearable and mobile sensors, mobile devices, assistive devices, automated systems of processing, and so on, which require a common exchange protocol to share their real time data [4]. However, the interoperability is one of the major issues to be addressed to promote the adoption of AAL solutions in real environments.

In the context of the connected smart homes, the software buses establish a simple and effective way to transfer these data. Due to their open nature, the software buses facilitate this information sharing and allow real time exchanges.

https://doi.org/10.1007/978-3-319-94523-1_21

However, to allow a reliable, effective and secured access to the sensors, the actuators, the connected objects and the assistive tools of the AAL, an Application Program Interface (API) is needed. For instance, this API allows the interaction devices and processing algorithms to create easily and safely the digital environment. This article specifies such an API based on a MQTT software bus treelike architecture.

After introducing the context of the study, the authors of the article propose a brief state of the art of the software buses and discuss the equivalent proposals found in the literature. The chosen bus, as well as the designed hierarchical model are then detailed. An application of the model in a living lab (smart home of Blagnac) to control ambient sensors is presented, before concluding with a conclusion and perspectives.

## 2 Context and Problematic

### 2.1 Messaging Protocols

Messaging Protocols enable simplification of real-time exchanges between connected agents. These information exchanges are done via message notifications instead of polling, which is more efficient for the network. Each message is received by each connected agent, so a high number of messages can be exchanged, which implies an important quantity of resources (Central Processing Unit, memory, bandwith, etc.) on embedded devices that are generally energy constrained. To limit messages, Messaging Protocols provide some filtering mechanisms. Filtering is a key element for the deployment of messaging protocols since it provides selection criteria to select useful messages.

### 2.2 Need of an API Dedicated to Smart Homes

Intelligent and connected smart home consists of heterogeneous sets of technologies. The high diversity of the current standardised communication technologies (LoRa, Zigbee, KNX, Z-Wave, Bluetooth, etc.) or proprietary substantially increases the complexity of the networking of sensors, actuators, assistive tools and interaction devices deployed in AAL. In addition, one of the propriety needed in the AAL is the high level of reconfigurability.

Instead of sensors and actuators cabling into an electrical panel, it should be preferable to create a logical separation by an erasable programmable element. Moreover, this problem is accentuated by the multiplication of connected devices in the smart home that are not directly interoperable. To solve this problem, the solution is to create a hardware and/or software component that can interact with a device and transcribe its data in a standardised form. This operation can be greatly facilitated by a software bus to unify all the objects deployed with various network technologies; to do this, intermediate gateway nodes must be deployed to help the convergence of data on a common bus.

To be complete and allow good reconfigurability, the software bus can also be used as an API dedicated to smart homes. This API makes it possible to create

a logical limit between access to all the physical devices of the house and the software defining the communication. This API must make it possible to activate an actuator of the ambient environment (such as light, shutter, door, etc.) and read the data coming from a sensor, etc. Thus, it will simplify the design and the evaluation of innovative technological components.

This API must be specified by making best use of the features of the selected software bus. The next section details the different software buses used in the field of AAL.

## 3 Related Works

As previously described, the efficiency of a solution based on a software bus depends strongly on the implemented sharpness of the filtering rules. In the literature, we meet generally three types of filtering [5]:

- Filtering based on the contents: the agents have to subscribe to a message content; to do this, they have to know the message or a part of it. A specification of contents by regular expression, for example, can be used. It is the case of the Ivy bus [6];
- Filtering based on the type: messages are typified by the emitting agent; the receiving agents have to subscribe to a type of particular message to receive it, it is the case of the KNX home automation bus which associates with every transmitted telegram a service identifier type destination;
- Filtering based on the subject: the agents have to subscribe to a shape of subject or topic. From then on, they receive all the subjects published with this subject. MQTT [7] is from this category.

Numerous protocols share the Internet of Things market, particularly CoAP (Constrained Application Protocol), MQTT (Message Queuing Telemetry Transport), AMQP (Advanced Message Queuing Protocol) and HTTP (HyperText Transfer Protocol). MQTT, by its publisher/subscriber paradigm and low protocol footprint, is a very interesting choice for a wide range of small objects [8]. The well-known HTTP protocol is not a binary protocol but a textual protocol, resulting in very large message sizes. It is mainly and historically dedicated to the IP world rather than IoT sensor networks and other connected objects world.

AMQP is a fairly complete protocol in terms of queue management algorithm, which allows it to be the asynchronous complement of HTTP while being able to interface easily with MQTT. AMQP stands out as an interesting choice for backends and gateways but not for constrained embedded systems [9]. CoAP and MQTT are two protocols that meet the same needs. They are more difficult to differentiate for our application [10]. CoAP is based on the User Datagram Protocol (UDP) and therefore it allows less restrictive embedded implementation than MQTT. However, MQTT has a topic filtering with a hierarchical tree organisation which makes it possible to consider very interesting filtering strategies thanks to the relationships between the nodes. In addition, if no queuing mechanism like AMQP is proposed, MQTT provides an option that allows the

last message to be retained by the server and sent to any new connection from a client. Finally, MQTT has an extension called MQTT-SN (MQTT For Sensor Networks) that allows the use of the protocol on very constrained networks.

Furthermore, in order to make application software relatively portable and independent from the hardware, it is necessary to create an object abstraction layer by modelling their capabilities and characteristics. This modelling is a major challenge of AAL and Internet of Things (IoT). The purpose of our work, presented in this article, is not to propose a new abstraction layer of IoT, but to demonstrate that MQTT, by means of the structure of topics, makes it possible to implement this representation of flows [11].

With its advanced filtering capabilities, the MQTT bus is thus an excellent candidate to manage data transfer in a connected smart home where many heterogeneous data coexist.

## 4 Specifying an MQTT Tree for a Connected Smart Home

### 4.1 Technical Background: MQTT

MQTT is currently popular in the field of IoT, itself very busy in the field of connected smart home. To take full advantage of its advanced filtering by topic, to facilitate the scalability and efficiency of connected objects, and in order to easily analyse the amount of data generated, it is necessary to devote an in depth reflection on the hierarchical organisation of topics.

MQTT is based on a client/server message transport protocol. The agents connected to the bus are the clients and the server is designated by the term "broker". The filtering by topic is provided by this broker. An agent wishing to deposit a message on the bus is a publisher whereas an agent which has subscribed to one or more topic is a subscriber. This principle is illustrated in Fig. 1. An object can be both publisher and subscriber.

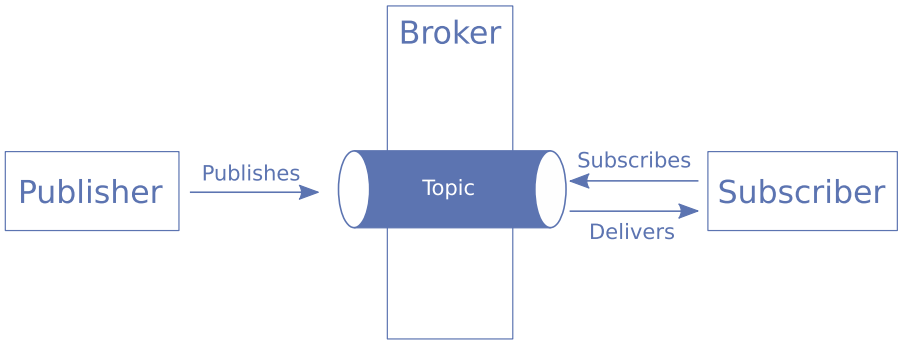


**Fig. 1.** Broker, suscriber, publisher with MQTT

An MQTT topic is the subject of the published message: it allows to filter messages. An MQTT topic is based on a tree structure where nodes are characters strings and where each level is separated by a slash.

Example: `house/bedroom/light/ceiling`.

MQTT topics are always absolute and noted from the root, which is implicit. To subscribe to a topic, it is possible to combine two wildcards: plus (`+`) and hash (`#`). Thus, an agent can subscribe to several subjects at the same time. Below two topics:

`house/bedroom/lamp/ceiling`
`house/bedroom/temperature/bed`

The wildcard `+` allows to consider the non-filtering and non-hierarchical case "Whatever the value of this topic level". For example, a subscription to `home/+/lamp/ceiling` will allow the agent to receive all ceiling lights in the house. The `#` wildcard allows you to consider the hierarchical non-filtering case "No matter what is below this topic level". For example, a subscription to `home/bedroom/#` will allow the agent to receive all messages related to the room: temperatures, lamps, etc. The topic tree-structure is free and chosen by the agent; nevertheless a specific configuration on the broker can introduce access rules on the tree (read-only, write-only, access denied) for each authenticated agent.

The MQTT broker does not provide for storing old messages; it is therefore generally necessary to associate the deployment of an MQTT bus to that of a database. However, MQTT proposes the retained mechanism which, if it is activated on a topic, forces the broker to retain the last message on this topic and also to communicate this last message from any new subscription to this topic. This storing is limited to a single previous message.

From this point of view, MQTT is a bus with very interesting features for AAL:

- It is based on TCP/IP: it is not dependent on a particular physical technology, and can be implemented on objects when an IP interface is available (smartphone, tactile tablet, embedded computing device such as Arduino/Raspberry, server...).
- It is based on a standard of IETF (Internet Engineering Task Forces) which is an open standard and recognised by the Internet standardisation agency.
- It is authenticated/encrypted: it makes it possible to initiate a thinking on the security of data and the respect for their retention to the users.
- It is deployable locally and is operational off-line: the broker can be deployed locally in a smart home and can be temporarily or permanently disconnected from the Internet without disruption. It is also possible to deploy a public broker, this makes it easier to share the data.
- MQTT works in real time and keeps in memory the last message: the messages are broadcast with a very low latency which allows the instantaneous processing of messages by automated systems. Moreover, by the retained option described above, the last message can be sent back to the connection, without soliciting the sending agent, which contributes to reducing the connection processing time of the intermittent or ephemeral agents (tablets in sleep mode for example).

However MQTT has some limitations. Because MQTT is based on the principle of flooding unicast messages to different subscribers it then can quickly overload the connected agents if the filtering is not done properly and, then, present a risk of no passage at scale. For example, some solutions minimise the description of the message in the topic and promote a rich content in the payload of the message by using a JavaScript Object Notation. This solution involves the reception and analysis of any message by the agent, whereas this analysis could be done by the broker. Indeed, as previously mentioned, the filtering by topic minimises the number of messages received by an agent. If the agent is an embedded device with few hardware resources (Central Processing Unit, memory, network channel) and energy (because battery power), an optimal filtering permit to maximise the performance of the agent and facilitates the transition to the scale of the entire system.

In a first trial, the housing environment is considered as known and relatively fixed. It can be thus described in a almost exhaustive way with a relatively weak modification frequency. The tree can thus take advantage of this quasi-exhaustive description to facilitate the filtering of messages. The proposed tree has to allow to maximise the possibilities of sorting by the topic specific for MQTT. We shall thus try to maximise the description of the environment to allow a very fine filtering. So, the broker can request the agent as little as possible, by transmitting only the necessary messages and to limit so the network overhead.

Some objects may also be natively MQTT compatible.

### 4.2 Contribution

As justified above, it is essential to take advantage intelligently of the flows filtering at the broker level to limit the number of messages. It is so necessary to create the most specific possible communication channels by using precise, explicit and hierarchical topics.

The second rule of design is being easily able to take advantage of the subscription in several topics via the use of wildcards. It is necessary to define for it an exhaustive treelike structure the root of which concerns general elements and sheets of which describe very precise elements. However, the design of this treelike structure need to have a complete representation of the ambient assistive tools available in AAL.

The tree proposed here allows to organise and to filter the transfers of messages on the MQTT bus. Its role is to interact with the connected house environment. This API which can evolve over time to propose a new organisation can have different versions. The next section presents the first version of this tree designed for research project on well being for elderly in a Living Lab.

The general format of the tree for the API #1 is represented by Fig. 2.

In this tree, the levels are to be considered by pair, with a first level (odd, $n$) allowing to interpret the second level (even, $n+1$). For example, at the first level, “api” allows to interpret the identifier “id” at the level 2 as being the version number of the API.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|
| api | <id api> | room | <piece> | device | <device type> | id | <id> | <request/response/indication> |

**Fig. 2.** Generic tree-representation

Levels 3, 5 and 7 are "free levels" where the fields are used to specify the place, the nature and to identify the equipment concerned by the message. Without being exhaustive, the Table 1 gives some examples of possible values by level.

**Table 1.** Examples of values of level 3, 5 and 7, with corresponding values 4, 6 and 8.

| Level $n$ | Content $n$ | Content $n+1$ | Example |
|---|---|---|---|
| 3 | `room` | Name of room at level 4 | `kitchen` |
| 3 | `device` | Name of object at level 4 | `smartphone` |
| 5 | `sensor` | Physical parameter at level 6 | `temperature` |
| 5 | `light` | Type of light at level 6 | `roof` |
| 7 | `id` | Identification at level 8 | `1` |
| 7 | `localisation` | Description of localisation at level 8 | `top door` |

The last level of the tree indicates the nature of the message, which can be:

- request: that is, a message intended to transport an order for an actuator or a request to interrogate a sensor;
- response: that is, a message in response to a request message, intended to carry the request processing confirmation for an actuator, or the data of a sensor;
- indication: that is, a message intended to carry the data from an actuator or a sensor on its initiative, either in case of change, or in case of regular sending.

Some topics, such as the on-off state of a lamp will be able to take advantage of the *retained* option of MQTT. At the subscription, the agent then automatically receives by the broker the last known value transported over the bus, without requesting the publisher again. This option makes it possible, for example, for a touch pad leaving a standby mode, to update the state of its interaction components without having to query the agents responsible for control of sensors and actuators linked to them.

## 5 Example of Deployment and Feedback

### 5.1 Test Platform

A first deployment of the contribution has been done on the *Smart home of Blagnac* living-lab (MIB) [12], dedicated to the elderly. The platform is a "real" house with several rooms: kitchen, bathroom, bedroom, living-room...

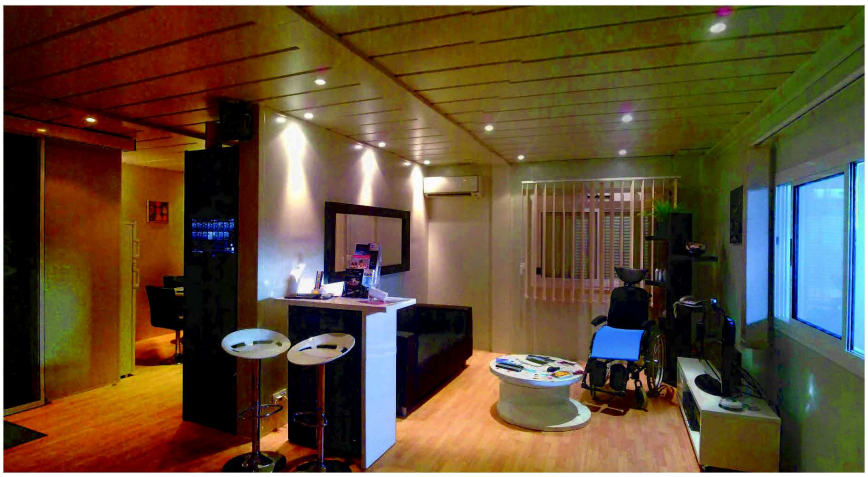

**Fig. 3.** The Smart Home of Blagnac

The MIB living-lab implements several heterogeneous networks and home-automation technologies. The main infrastructure is made of a KNX network for typical home-automation devices: lamps, rolling shutters, motion sensors, mobile furniture. Some others devices implement proprietary IP-based (WiFi, Ethernet) or Zigbee protocols (temperature sensors, other motion sensors, television remote control...). A Text-To-Speech system is also deployed (Fig. 3).

A simple middleware MiCOM [13] has been initially developed and deployed in the MIB living lab. This very simple middleware, based on the HTTP protocol has enabled several studies on usability and acceptability of Information Communication Technologies (tactile and speech interaction) to control an intelligent smart home. For more information, see [14,15]. However the HTTP protocol needs frequent polling for detecting sensors and actuators changes, which is not optimal.

### 5.2 Deployment on the Platform

The API has been deployed in the MIB Living Lab, in respect for the specification presented in Sect. 4. Several MQTT topics have been used for movement sensors, lamps, rolling shutters and Text-To-Speech system.

The movement sensors are available by room. If multiple sensors are available in a single room, the sensors are identified by the `id`. The deeper level of the tree is `indication`, since the data are published by the sensor; the *retained* option is activated: the broker memorises the last data and automatically sends it on each agent new connection, without need to explicitly request the data to the sensor. Topics are given on Fig. 4.

The lamps are also available by room. The lamp type (*ceiling*, *bedside*...) is given in the topic. If multiple lamps are available in a single room (such as in the kitchen) the lamps are identified by the `id`. The deeper level of the tree can be either `request` for a modification request or `indication` for a status request.

```
api/1/room/bathroom/sensor/movement/id/1/indication
api/1/room/bedroom/sensor/movement/id/1/indication
api/1/room/livingroom/sensor/movement/id/1/indication
api/1/room/livingroom/sensor/movement/id/2/indication
api/1/room/wc/sensor/movement/id/1/indication
```

**Fig. 4.** Topics used with movement sensors

The MQTT *retained* option is activated on the `indication` topic, as well as sensors. Topics are given on Fig. 5.

```
api/1/room/bathroom/lamp/ceiling/id/1/[request|indication]
api/1/room/bedroom/lamp/ceiling/id/1/[request|indication]
api/1/room/bedroom/lamp/bedside/id/1/[request|indication]
api/1/room/kitchen/lamp/ceiling/id/1/[request|indication]
api/1/room/kitchen/lamp/ceiling/id/2/[request|indication]
api/1/room/kitchen/lamp/ceiling/id/3/[request|indication]
api/1/room/livingroom/lamp/ceiling/id/1/[request|indication]
api/1/room/livingroom/lamp/ceiling/id/2/[request|indication]
```

**Fig. 5.** Topics used with lamps

As well as the lamps, the rolling shutters commands are available for each shutter with a `request`-type topic; the status (open/closed) is available with the `indication` topic. Topics are given on Fig. 6.

```
api/1/room/bedroom/window/shutter/id/1/[request|indication]
api/1/room/kitchen/window/shutter/id/1/[request|indication]
api/1/room/livingroom/window/shutter/id/1/[request|indication]
```

**Fig. 6.** Topics used with rolling shutters

At last, the Text-To-Speech system can be ordered by room or globally with the `request` topic. Topics are given on Fig. 7.

### 5.3 Using the API

This section proposes two examples (scenarios) that use the proposed API.

In a first scenario, the resident leaves the home and want to check if one or several lamps of the house are switched on. To proceed, the single following topic can be subscribed: `api/1/room/+/lamp/+/id/+/indication`. The state of each lamp (whatever the room, the lamp type and for all lamps in each room) is returned by the broker. Thanks to the retained option on these topics, the last state is returned, without need of requesting the current state.

```
api/1/room/bedroom/voice/say/id/1/request
api/1/room/kitchen/voice/say/id/1/request
api/1/room/livingroom/voice/say/id/1/request
```

**Fig. 7.** Topics used with the speech synthesis

In a second scenario, the resident wants to open all the rolling shutters of the house. As the jokers `+` and `#` cannot be used for publication, an exhaustive list of topics should be used to request the opening of the shutters:

```
api/1/room/bedroom/window/shutter/id/1/request
api/1/room/kitchen/window/shutter/id/1/request
api/1/room/livingroom/window/shutter/id/1/request
```

This example thus illustrates one of the limitations of the current (3.1.1) MQTT specification.

## 6 Conclusion and Perspectives

This paper has introduced the needs of a generic API to connect and share messages of connected objects via heterogeneous networking technologies in the AAL context. This paper has reported the importance and the principles of the MQTT bus. It also demonstrates the pertinence to accurately represent the tree of connected objects of AAL to take benefits of the features of the MQTT bus. In the prototyping of assistive tools for AAL applications, MQTT allows fast-prototyping and rapid deployment. The description syntax is simple and can be used by everyone, including researchers in human-computer interaction to test new control modes as well as engineers in home automation. The deployment on the MIB platform confirms the relevance of the accuracy of the tree-representation.

One of the next challenges concerns the self-adaptation of the API, taking into account the automatic discovery of new connected objects thanks to networking auto-configuration protocols. This automatic discovery mechanism should improve the scalabilty of the services provided by assistive tools of AAL.

## References


1. Hammi, B., Khatoun, R., Zeadally, S., Fayad, A., Khoukhi, L.: Internet of Things (IoT) technologies for smart cities. IET Netw. J. **7**(1), 1 (2018)
2. Vergaard Hansen, F.O.: Ambient assisted living healthcare frameworks, platforms, standards, and quality attributes. Sensors (Basel) **14**(3), 4312–4341 (2014)
3. Rashidi, P., Mihailidis, A.: A survey on ambient assisted living tools for older adults. Biomed. Heal. Inform. IEEE J. **17**(3), 579–590 (2013)
4. Yacchirema, D.C., Palau, C.E., Esteve, M.: Enable IoT interoperability in ambient assisted living: active and healthy aging scenarios. In: 2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, 2017, pp. 53–58 (2017)

5. Mühl, G., Fiege, L., Pietzuch, P.R.: Distributed Event-Based Systems. Chap. 2.3 Notification Filtering Mechanisms. Springer, Heidelberg (2006). https://doi.org/10.1007/3-540-32653-7
6. Buisson, M., Bustico, A., Chatty, S., Colin, F-R., Jestin, Y., Maury, S., Mertz, C., Truillet, P.: Ivy : Un bus logiciel au service du développement de prototypes de systémes interactifs, In: ACM IHM 2002, Poitiers, Novembre 2002, pp. 223–226 . ACM Press (2002). (in French)
7. MQTT Specification v3.1.1. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html
8. Naik, N.: Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP. In: 2017 IEEE International Systems Engineering Symposium (ISSE), Vienna (2017)
9. Cohn, R.: A Comparison of AMQP and MQTT. White Paper, StormMQ (2011)
10. Thangavel, D., Ma, X., Valera, A., Tan, H.X., Tan, C.K.Y.: Performance evaluation of MQTT and CoAP via a common middleware. In: 2014 IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore (2014)
11. Mainetti, L., Manco, L., Patrono, L., Sergi, I., Vergallo, R.: Web of topics: an IoT-aware model-driven designing approach. In: 2nd IEEE World Forum on Internet of Things (IEEE WF-IoT). IEEE (2015)
12. The "Maison Intelligente de Blagnac" (Blagnac City's Smart Home) living lab website. http://mib.iut-blagnac.fr. (in French)
13. Vella, F., Blanc Machado, M., Vigouroux, N., van den Bossche, A., Val, T.: Connexion du Middleware MiCom avec l'interface tactile InTacS pour le contrôle d'une smart home. Journées francophones Mobilité et Ubiquité, Lorient, France (2016). (in French)
14. Bougeois, E., Duchier, J., Vella, F., Machado, M.B., Van den Bossche, A., Val, T., Brulin, D., Vigouroux, N., Campo, E.: Post-test perceptions of digital tools by the elderly in an ambient environment. In: Chang, C.K., Chiari, L., Cao, Y., Jin, H., Mokhtari, M., Aloulou, H. (eds.) ICOST 2016. LNCS, vol. 9677, pp. 356–367. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39601-9_32
15. Van den Bossche, A., Campo, E., Duchier, J., Bougeois, E., Machado, M.B., Val, T., Vella, F., Vigouroux, N.: Multidimensional observation methodology for the elderly in an ambient digital environment. In: Miesenberger, K., Bühler, C., Penaz, P. (eds.) ICCHP 2016. LNCS, vol. 9758, pp. 285–292. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41264-1_39