# White Paper, First Draft: The Design of Software-based Smart hub supporting multiple wireless technlogies.[⋆]

No Author Given

No Institute Given

**Abstract.** IoT smart hub is a software-based module that acts as the central point connecting and controlling the access to different smart devices in the system. This proposed software-based smart hub supports different wireless technologies such as LoRa, Bluetooth and WiFi. The fine-grained access control system is also integrated into the design of smart hub system architecture. This paper covers extensive number of tasks aiming to define the general specification of software based smart hub; it presents also the design and implementation of the smart hub.

**Keywords:** LoRa; WiFi; Bluetooth BLE; smart home; IoT; security; smart home hub;

## 1 Introduction

Internet of Things (IoT) [25] can be described as connecting everyday objects like mobile phones, smart watches, electronic tablets, sensors and actuators to the Internet where the devices are intelligently linked together enabling various ways to communicate and control between things and peoples, and between things themselves. IoT devices use various network technologies (e.g. wire or wireless) for their communication. However, the most popular of which is the wireless network technologies such as Wifi, Bluetooth and the new emerging technology LoRa [23], which supports long range communication and requires less energy consumption.

In general, IoT devices are connected to smart hub, through which user can control and access them. The smart hubs, existed in the market, are built by different companies (e.g. samsung smartthing, logitech harmony, etc. ) [16] based on different specifications and design. And they are generally embedded into their devices, which are expensive. Thus, our goal is to provide the alternative for user in building their IoT systems, instead of relying on the company-made smart hub device, user can use the software-based smart hub supporting multiple network technologies. This software-based smart hub can be installed in any low-cost device such as Raspberry-pi [24].

In this paper, we address the issue of building the secure software-based smart hub for IoT system where the security aspect is integrated into the system

[⋆] This work is supported by FEDER-IDEE Project.

architecture. The work presented in this paper are divided into three main parts. The first of which focuses on the software-base smart hub protocol stack and global system architecture. The second part is about the physical and functional architecture and the last part dedicates to the implementation of this system.

This paper is structured as follows. Section 2 software based smart hub protocol stack. Section 3 talks about the functional requirements and global architecture of smart hub. Section 4 is about the detailed client interface system architecture. Section 5 focuses on the implementation of smart hub. Section 6 is about the communication message format between different modules. Section 7 presents sequence diagrams for different access scenarios for three different wireless network technologies: LoRa, Bluetooth and WiFi. Section 8 is the conclusion.

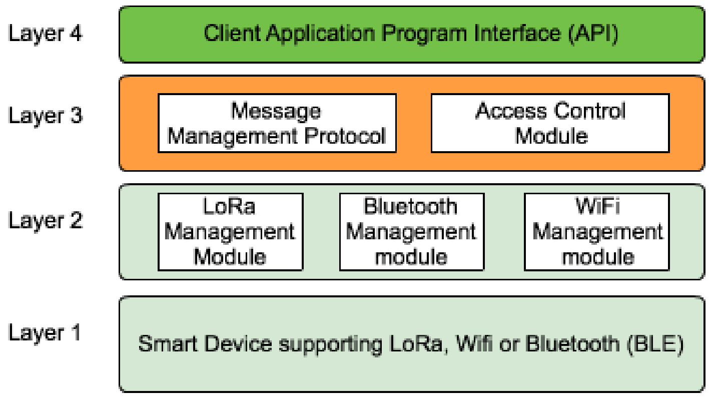
Layer 4
Client Application Program Interface (API)
Layer 3
Message Management Protocol
Access Control Module
Layer 2
LoRa Management Module
Bluetooth Management module
WiFi Management module
Layer 1
Smart Device supporting LoRa, Wifi or Bluetooth (BLE)

**Fig. 1.** High level smart hub protocol stack

## 2 Software-based smart hub protocol stack

The main role of software-based smart hub is to facilitate the communication between IoT client application and the smart devices through different network technologies. To do so, smart hub should be able to communicate with different devices supporting different network technologies, hence, multiple network APIs must be integrated into smart hub. Smart hub is actually the central point of controlling, it controls all the requests both from the IoT client application as well as the devices. Thus, the access control API should also be integrated into smart hub to ensure proper security protection. The Figure 1 provides the high level protocol stack of smart hub.

- Layer 1 is where the devices are handled. The devices are heterogenous, in the sense that they can support different network technology and configuration.

- Layer 2 is the network management layer where the information from different devices are processed according to the network technologies used. Three module are proposed to support three different technologies. They are LoRa, Bluetooth and WiFi network management module.
  1. LoRa network management module is responsible for managing all the incoming and outgoing request from lora nodes.
  2. Bluetooth network management module is responsible for managing the incoming and outgoing request from bluetooth nodes.
  3. WiFi network management module is responsible for managing the the incoming and outgoing request from WiFi nodes.
- layer 3 is where all the communication messages are controlled and exchanged. In this layer there are two important modules.
  1. Message management protocol is responsible for subscribing and publishing the data and requests from client application to devices and vice versa.
  2. Access control module is responsible for controlling all the incoming requests and outgoing data. All the instructing request to devices must be approved by access control model before any action can be executed.
- Client Application Program Interface is a module responsible for providing the structure data to the client application where the data is processed or displayed. The data can be structure with different language such as JSON, XML or other data expression/represention language.

We propose this architectures based the assumption that the existing smart home hub does not support access control mechanism and it uses only the traditional authentication method for securing devices and data pertaining to them.

## 3 Functional requirements and Global Architecture of smart hub

In this section, we focus on the functional requirements and smart hub's architecture.

### 3.1 Smart Hub Functional requirements

The main objective of building Smart Hub is to centralize and facilitate user in devices and access control policies management. Furthermore, the hub should be able to support different network technologies. Given that, smart hub should support different communication protocols and security mechanisms to be able to interoperate with different devices. The below functional requirements should exist in the design of our proposed smart hub.

1. Usability. User should be able to perform different settings to allow smart hub to connect to different devices supporting different network technologies.
2. Security. The communication between smart hub and devices should be secure.

3. Extensibility. Smart hub should be extensible. If new network technology is available in the market, the hub can operate with them by just extending its network management interface without going through major modification. This adjustment should be possibly done by ordinary user with a software package update.
4. Access Control. Smart hub should be able to provide the fine-grain access control to devices and data pertaining to them. User should also be able to enable or disable the access control option.

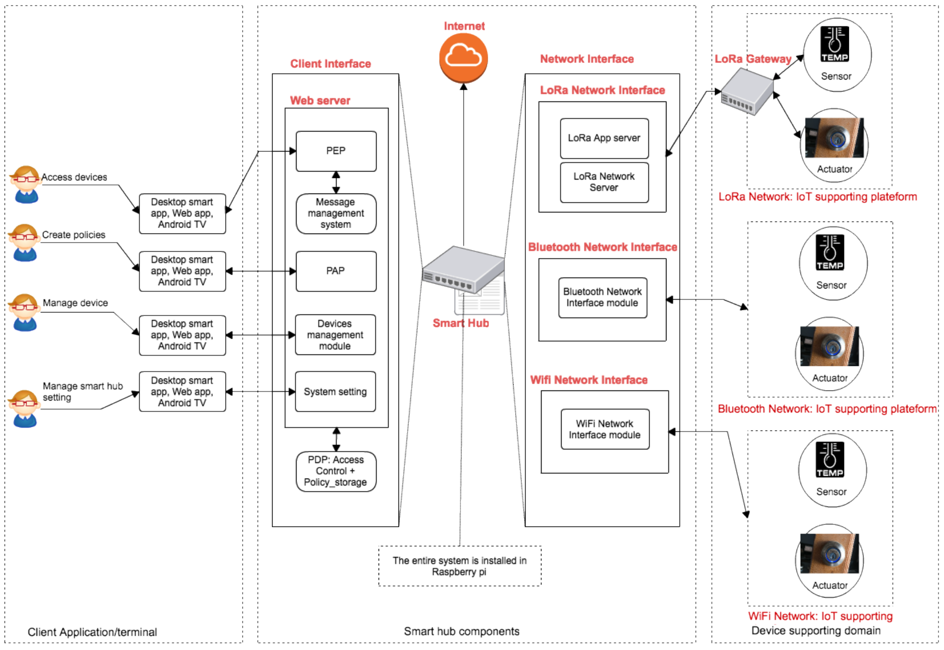
Internet
Client Interface
Web server
PEP
Message management system
PAP
Devices management module
System setting
PDP: Access Control + Policy_storage
Network Interface
LoRa Network Interface
LoRa App server
LoRa Network Server
Bluetooth Network Interface
Bluetooth Network Interface module
Wifi Network Interface
WiFi Network Interface module
Smart Hub
The entire system is installed in Raspberry pi
LoRa Gateway
Sensor
Actuator
LoRa Network: IoT supporting plateform
Bluetooth Network: IoT supporting plateform
WiFi Network: IoT supporting
Device supporting domain
Access devices
Create policies
Manage device
Manage smart hub setting
Desktop smart app, Web app, Android TV
Client Application/terminal
Smart hub components

**Fig. 2.** Smart hub physical architecture

## 3.2 Smart hub physical system architecture

As shown in Figure 2, smart hub architecture consists of different components. It can be divided into two main parts: network interface and client interface. First part of the architecture is for interfacing between smart hub and smart devices with different network technologies. The second part is the client interface where user can, through different type of devices (e.g. smart phone, android TV, etc.), connect to smart hub for system setting, device management and access control and instructing devices.

1. Client interface.
   - Policy Enforcement Point (PEP) is responsible for enforcing the duties that user or system needs to perform before or after access permission to device or data is granted.
   - Message management system is a module responsible for managing all the incoming and out going message between client application and devices in the system. It supports different message management protocols (e.g. MQTT, Websocket, etc.) that are used for exchanging structure data between client application and smart hub or smart hub and smart devices installed in the network. This module also contains a sub-module that is responsible for structuring data into a standard message structure (e.g. xml, json, etc.) used for each message management protocol.
   - Policy Administration Point (PAP) is a part of access control module. Through this module, user can manage the access control policies to devices as well as smart home. It allows user to create access control policies for a particular user and devices, to remove unwanted policies and to edit the existing policies. The main function of this module is to produce the access control polices and store them in the designated storage.
   - Device management module. This module is responsible for managing devices in the network. This module acts as the intermediately between client application and smart hub. The module allows user to add, remove and configure devices in the network. It also allows user to set different actions required for device operation.
   - System setting module is responsible for providing user with comprehensive interface for setting up the mega hub and other modules embedded in mega hub.
   - Policy Decision Point (PDP) and policy storage. This module has two parts. First of which is the PDP that is responsible for deciding whether to grant or deny access to devices based on user's request. The decision is done based on the defined policy in the storage. The second part is the policy storage, which is responsible or managing the policies created by PAP module. The management includes, how to store, to secure and to retrieve policies.
2. Network Interface is a module responsible for managing different network technologies in the system. Three wireless technologies are considered in our design of smart hubs: WiFi, LoRa and Bluetooth. It is worth noting that all the communication messages between user and devices need to get through this module. This module actually has the direct communication with message management system in Client Interface.
   - Lora Network Interface allows smart hub to communicate, push or pull data to and from LoRa supporting devices in the network. This interface consists of two main modules. The first module consists of a set of LoRa modules allowing LoRa network to function such as LoRa application server, LoRa network server and LoRa gateway. The second module is the interface between LoRa network module with message management

system. This second module is responsible for sending data or retrieving data from LoRa devices through LoRa network. Then, it subscribes or publishes data to message management system using different messaging protocol such at MQTT, Websocket or other protocols.

– Bluetooth Network Interface Module is responsible for managing all communications and messages exchanged between client interface and bluetooth supporting devices. Bluetooth network interface creates the star bluetooth network topology where devices are paired and exchanged data with a master or server, which is a part of bluetooth network interface. Beside its main role in constructing and managing bluetooth network, it also has an internal module, which is responsible for pushing and pulling data between client interface and bluetooth devices.
– WiFi Network Interface is responsible for managing all the communications between client interface and smart devices supporting WiFi network. Once the device joints WiFi network, smart device can publish or subscribe data to message management system through this interface.

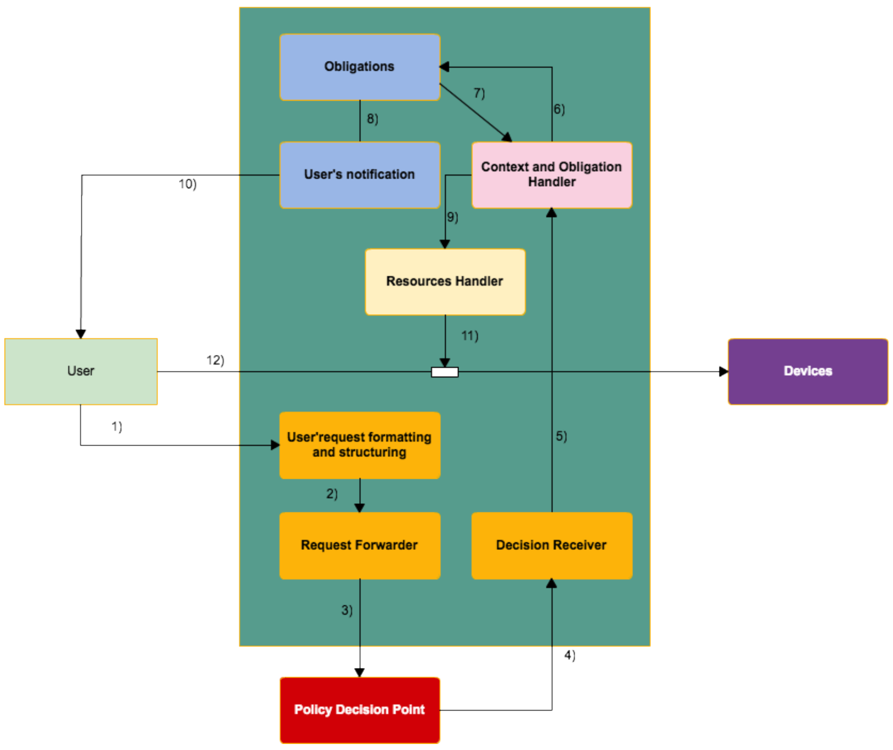

**Fig. 3.** Policy Enforcement Point Architecture

## 4 Detailed Client Interface System Architecture

In this section, we detail the architecture of each components of the Client Interface module, see Figure 2.

### 4.1 Policy Enforcement Point

The PEP's system architecture consists of the following components, see Figure 3.

1. User is the client's API used as an interface between physical person and smart hub. It can be web application or user desktop. All user's request are sent through this module.
2. User's request formatting and structuring is a module in PEP, which is responsible for formatting user's request and structuring data to the required format of policy decision point. For example, formatting request into XACML standard request or JSON data's structure.
3. Request forwarder is responsible for forwarding the formatted request to policy decision point.
4. Decision receiver retrieves the decision made by policy decision point and forwards the decision to context and obligation handler.
5. Context and obligation handler is responsible for examining all the requires obligations that need to be performed by user or system before allowing user to access device. After identifying the obligation it sends request to obligations module for executing.
6. Obligations module is responsible for executing all the required obligations by user or system.
7. User's notification is responsible for notifying user either about the required obligation that needs to be executed or about the access permission decision made by PDP.
8. Resources handler is responsible for handling the devices, this module is generally waiting the answer from context and obligation handler before releasing the access flow to user.

The step-by-step information flow is presented below, see Figure 3.

*Figure 1:* step-by-step information flow explanation

1) User sends access request to policy enforcement point. The request is then formatted and structured to the required standard.
2) Once access request is correctly formatted, the formatted request is sent to request forwarder.
3) Request forwarder is then forward the request to policy decision for validation.
4) Policy decision point performs policy validation against the available polices and the decision is then sent to decision receiver.
5) Decision receiver upon receiving the decision, it sends the decision with other

required actions that need to be executed to context and obligation handler.
6) If one or many obligations need to be fulfilled by user or system, obligation handler sends the request to obligations execution module.
7) Once the obligation is executed, it sends the acknowledgement to context and obligation handler to proceed.
8) Obligations module also send an acknowledgement to user's notification module.
9) Context and obligation handler signals the resource handler to release the flow if obligations are cleared.
10) User is notified.
11) Resources handler allows access.
12) User access devices.

---

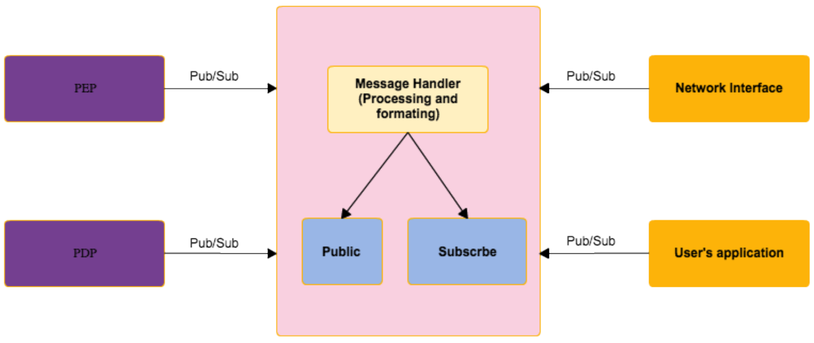

**Fig. 4.** Message management system

## 4.2 Message management system

As defined previously, the role of message management module is responsible for managing the the incoming and out going message between client Interface and network modules. It also manages the information exchange between different components in client interface module. This module has three sub components.

- Message handler is responsible for formatting message to the required structure before publishing it.
- Public module is used to publish data got from a particular module in the system.
- Subscribe module is used to get data from a particular module in the system.

In our proposed global system architecture, message management system communicates with four system components: the PEP, PDP, Network interface and user's application.

## 4.3 Policy Administration Point

The policy administration point is used to manage the access control policies defined for users. The detailed of PAP specification can be found in [20].

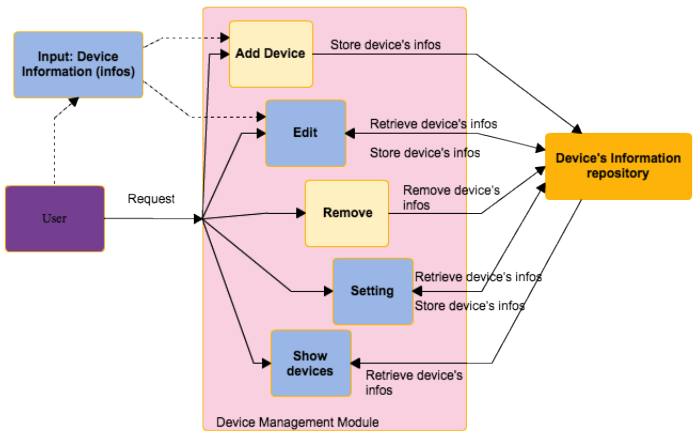

**Fig. 5.** Device management system

## 4.4 Device management module

In this section, we detail the device management module. This module allows user to add or remove device from the network. It also allows user to change device's setting. Figure 5 shows the detailed component of device management system.

- User is the client application allowing physical user to send request and input information required for device management operation.
- Input device information is a module responsible for temporarily storing device information provided by user. This information can be provided to other module in the system if required.
- Add device allows physical user to add IoT device into the network. Since our system supports different types of device for different network technologies, the information required for each device is also different.
- Edit allows user to modify the general information of a particular device.
- Remove allows user to remove a particular device from the device storage.

- Setting allows user to change the device's setting. For example, change a WiFi device to LoRa device or Bluetooth.
- Show allows user to show the existing devices in the network.
- Device's information storage is a module responsible for storing the device's information.

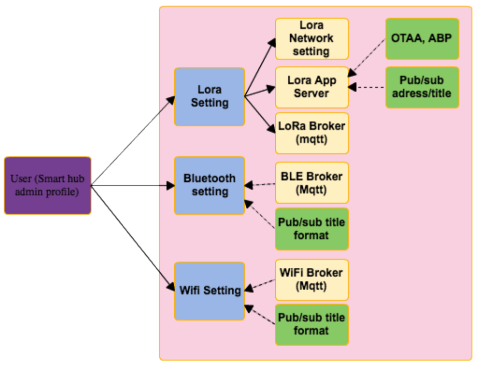

**Fig. 6.** Smart hub system setting

## 4.5 Smart Hub system setting

Smart hub system setting allows user to set different environment parameters for smart hub so that it can function correctly. The settings are divided into different part depending on types of network technology user wishes to user. There are three different settings for three types of network technologies: LoRa, WiFi and bluetooth system setting. As shown in Figure 6, the setting consists of the following components.

1. User is the super user or system admin who has rights to perform system setting.
2. Lora setting, this module is responsible for setting up all required parameters needed for hub to function correctly under LoRa. Three different settings need to be performed: Lora network, lora app server and lora broker settings.
3. Bluetooth setting is responsible for managing all the parameters required to be set in order for bluetooth to work correctly. Since communication between bluetooth module and client application uses MQTT, we need to set up the broker for bluetooth module. In addition to that, we need also to set up the pub/sub title structure for broker client.
4. WiFi setting is for setting up the parameters for WiFi communication. MQTT broker needs also to be set up in this module.

### 4.6 Policy Decision Point

See the implementation of PDP in [21]

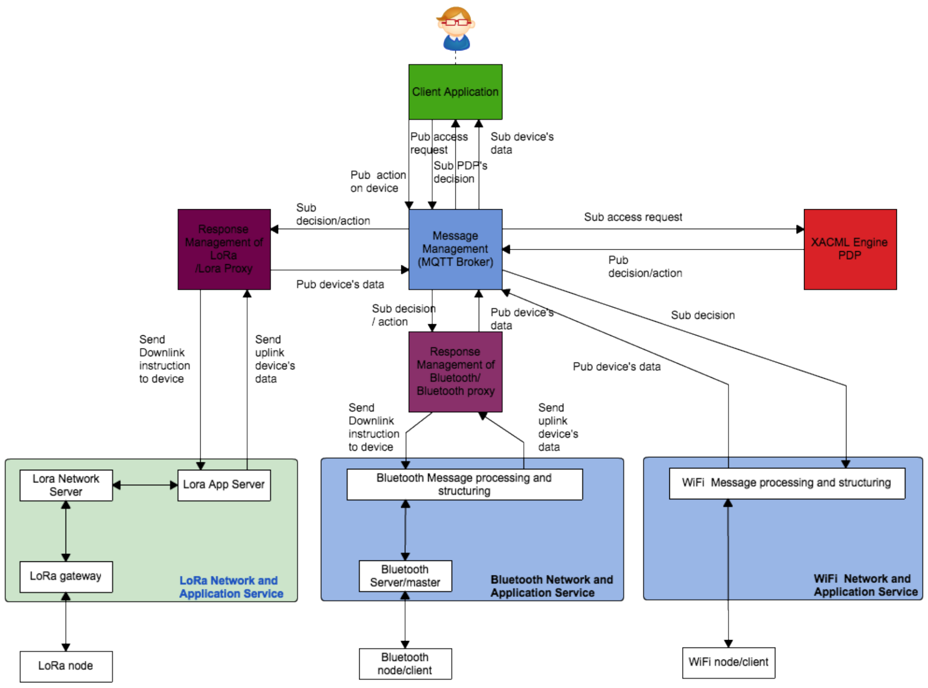
Client Application
Pub access request
Sub PDP's decision
Sub device's data
Pub action on device
Sub decision/action
Response Management of LoRa /Lora Proxy
Message Management (MQTT Broker)
Pub device's data
Sub access request
Pub decision/action
XACML Engine PDP
Sub decision / action
Pub device's data
Sub decision
Pub device's data
Response Management of Bluetooth/ Bluetooth proxy
Send Downlink instruction to device
Send uplink device's data
Send Downlink instruction to device
Send uplink device's data
Lora Network Server
Lora App Server
LoRa gateway
LoRa Network and Application Service
LoRa node
Bluetooth Message processing and structuring
Bluetooth Server/master
Bluetooth Network and Application Service
Bluetooth node/client
WiFi Message processing and structuring
WiFi Network and Application Service
WiFi node/client

**Fig. 7.** Smart hub functional architecture

## 5 Implementation of smart hub

In this section, we focus on the implementation of the software-based smart hub. We also detail each software components and the technologies used to build this hub. This smart hub is designed to work primarily with raspberry pi, however, it can also be installed in other linux system.

### 5.1 System architecture

As shown in Figure 7, the system consists of the following modules.

– Client application module is an interface between physical user and the system. This module handles the user's request and processes data provided by

devices in the network. Client application module can be developed separately from the smart hub system. In our system prototype, we implemented client application as the web-based client application.
- Message management is a central component that manages all the incoming and out-going messages from different modules in the system. in our implementation, we use MQTT as the messaging protocol, hence, MQTT broker is used as the message management module.
- XACML engine is the access control module, which is responsible for validating all the user requests. In our implementation, we use XACML 2.O Java-enterprise XACML.
- The final set of components are for LoRa, WiFi and Bluetooth network management components. The detailed of them are presented in the following section.

## 5.2 Lora module

Lora module consists of the lora network and response management module, which is responsible for handling all the data flow from lora network nodes to the client application. This module is developed in python language and the communication between this module and the central message management module is done based on MQTT protocol. The core components of lora module is divided into two main parts: the response management module (Lora proxy) and the core LoRa network system consisting of many more sub components.

1. Response management of LoRa. This module is responsible for retrieving and forwarding data to and from lora network to message management module. It acts like a proxy between lora network and client application module. We developed this module in Java as the MQTT (MQTT Paho [27]) client that can subscribe and publish data from and to Lora network. It can also publish and subscribe data to and from message management module.
2. Core Lora network module consists of Lora network server, Lora application server and lora gateway. The lora package sent by Lora node is securely processed in this network module. In our implementation the open source lora server is used and the detailed of lora server architecture and implementation can be found in [22].

## 5.3 Bluetooth module

The bluetooth module consists of two sub components: the response management module and the core bluetooth network.

1. Response management module acts like the bluetooth proxy managing the incoming and out-going data to and from bluetooth network.
2. Bluetooth core network is responsible for processing the data to and from bluetooth nodes. We implement the star topology network where a bluetooth server is stalled in Raspberry pi and acts as the central point of contact to

other bluetooth supporting device. This server can scan all the devices in range or initiate the add-hoc connection to bluetooth device upon the request from client application. The bluetooth network in constructed using Bluepy library [26].

### 5.4 WiFi module

The WiFi module is responsible for processing data from wifi supporting devices. This module is more straight forward if compared with bluetooth and Lora. The message exchange is done directly with central message management module through MQTT messaging protocol. WiFi node generally subscribes for the decision from XACML's decision and the instruction from message management module. Wifi node does two possible things after getting decision. Either to publish data or execute the instruction.

### 5.5 XACML engine

The XACML access control engine design can be found in [11].

## 6 Communication message format between different modules

In this section, we define the communication messages between different components of smart hub. The messages structure we defined in this section are mostly related to the text structure of subscribe and publish of MQTT protocol. It also covers the structure of user request in XACML standard request language [11]. For the access control policy expression, one can find the details in [11].

### 6.1 Message structure: client application: publish access request and action on device instruction

We design the smart hub in such a way so that the developer can build their own client application and connect to smart hub. The communication between client application and smart hub is defined. We use the standard XACML [11] request expressed in XML. When there is an access request from user, client application forms XACML request and publishes the request to message management module where access request will be retrieved by XACML PDP engine. It is worth noting that we use MQTT protocol as messaging protocol, hence, client application module must support this protocol. Client application publishes the request by following the below submission structure.

**publishing topic: PDP_request**
**publishing value: the standard XACML 2.0 access request.**

In access request, device EUID is used as the unique identification of each device

and EUID is used as the resource in XACML.
The second publish topic concern the instruction on device if the PDP's decision is positive. The structure of publish topic varies depending on the type of network technology used.

- Lora. In case of lora network the topic that client application needs to use depending on each lora network distribution. For example, in our implementation, we use Lora server [23], the topic should be formulated in the following structure.

  **publishing topic: APP/APPID/Node/NodeID/TX**
  **publishing value: The JSON file containing the device EUID and other required information.** For more details, see [25]

- Wifi. In case of WiFi network, client application needs to publish the following structure.

  **publishing topic: Device EUID**
  **publishing value: 0 for turn-on and 1 for turn-off instruction. 2 for get instruction.**

- Bluetooth. In case of bluetooth, client application needs to publish the following structure. **publishing topic: Device EUID**
  **publishing value: 0 for turn-on and 1 for turn-off instruction. 2 for get instruction.**

Concerning subscription for data, client application module needs to subscribe two topics. The decision by PDP engine and the data published by devices. Again for subscription of data, the subscribed topic varies depending on type of technology used.

- Lora. In case of lora, the subscription structure is **APP/APPID/Node/NodeID/RX** .
- Wifi. In case of WiFi network, client application needs to subscribe **Device EUID**.
- Bluetooth. In case of WiFi network, client application needs to subscribe **Device EUID**.

Concerning subscription for PDP's decision, client application module needs to use the following structure for subscription.

- Lora. In case of lora, the subscription structure is **Decision_DeviceEUID**.
- Wifi. In case of WiFi network, client application needs to subscribe **Decision_DeviceEUID**.
- Bluetooth. In case of WiFi network, client application needs to subscribe **Decision_DeviceEUID**.

## 6.2 Message structure: XACML engine: subscribe access request and publish decision

PDP module acts as policy evaluation module where the access request is validated against defined access control policies. Thus, to get access request, PDP needs to subscribe the access request with the following topic: **PDP-request**.

Once the PDD gets access request, it evaluates the access request. If the decision is positive, PDP publishes the action that device needs to perform. The publishing structure is as follows.

**publishing topic: Decision_deviceEUID** .

In case of bluetooth, the publishing topic is **BLE_decision** and the value has the following structure. "Device_EUID action ". The action can be 0, 1, 2. 0 is turn-off, 1 is turn-on and 2 is get.

**publishing value: 0 for turn-on and 1 for turn-off instruction. 2 for get instruction.**

## 6.3 Message structure: Lora Proxy: subscribe PDP's decision and action and publish device's data

Lora proxy needs to know the decision made by PDP engine before processing further action. To get decision and action allowed from PDP engine, Lora proxy needs to subscribe the defined topic: Decision_deviceEUID. The value of Decision_deviceEUID is the action allowed on device.

Lora proxy can also publish back the value provided by requested device. The publishing structure is as follows.

**publishing topic: deviceEUID** .

**publishing value: value provided by device.**

## 6.4 Message structure: Bluetooth Proxy: subscribe PDP's decision and action and publish device's data

Bluetooth proxy needs to know the decision made by PDP engine before processing any action requested by client application. To get the decision and action allowed from PDP engine, Bluetooth proxy needs to subscribe the defined topic: Decision_deviceEUID. The value of Decision_deviceEUID is the action allowed on device. The action can be 0, 1, 2. 0 is turn-off, 1 is turn-on and 2 is get.

In case of get, bluetooth node needs to send uplink the node's value. Thus, the publishing topic is.

**publishing topic: deviceEUID** .
**publishing value: value provided by device.**

It is worth noting that in case of bluetooth, the bluetooth proxy acts or MQTT client that can publish or subscribe. The bluetooth proxy has direct connection with bluetooth node.

### 6.5 Message structure: Wifi: subscribe PDP's decision and action and publish device's data

Unlike LoRa or Bluetooth, WiFi has a very simple architecture, it connects directly to message management module without passing any proxy. In case of WiFi, MQTT client is integrated into the node where it can publish or subscribe for a topic. WiFi node subscribes for decision from PDP engine and publish the data depending on the request from client application.

**subscribe topic: Decision_deviceEUID** .

**publishing topic: deviceEUID** .

**publishing value: value provided by device.**

## 7 Sequence diagrams

In this section we present the sequence diagram showing the interaction between different modules (see Figure 7). We separate the diagrams into three different parts. The sequence diagram for LoRa, Bluetooth and WiFi communication.

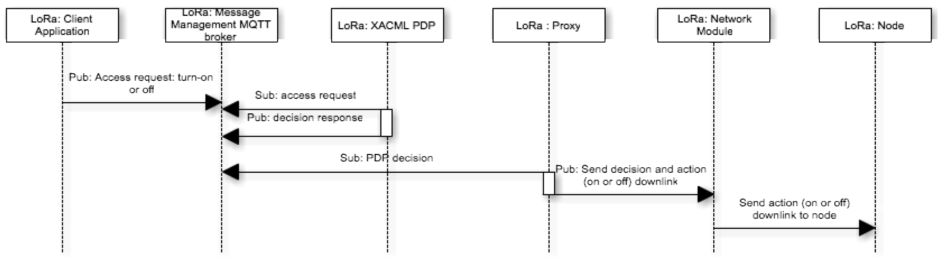
LoRa: Client Application
LoRa: Message Management MQTT broker
LoRa: XACML PDP
LoRa : Proxy
LoRa: Network Module
LoRa: Node
Pub: Access request: turn-on or off
Sub: access request
Pub: decision response
Sub: PDP decision
Pub: Send decision and action (on or off) downlink
Send action (on or off) downlink to node

**Fig. 8.** LoRa: sequence diagram, turn-on and off action request

### 7.1 LoRa: Sequence diagram

In case of LoRa there are two sequence diagrams for two different sets of scenarios: Turn-on or off and get (get data from node). These two scenarios have different number of processing and communication between modules.

1. Turn-on or off. In case of turn-on or off, client application needs to send a request and the access request to turn on or off needs to be validated against predefined access control policy at PDP XACML engine. The PDP engine, after policy validation and in case of possible decision, publishes the decision response (contain the action the device needs to perform) to message management module (central broker). Then, LoRa proxy subscribes the PDP decision and sends downlink message, through LoRa network, to lora node. The detailed sequence diagram is shown in Figure 8. The step-by-step information flow is presented below, see Figure 8.

*Figure 8:* step-by-step information flow explanation

1) When physical user executes an access request, client application forms the access request and publishes it to management management module (MQTT broker).
2) XACML engine subscribes the access request and is waiting for the callback from message management module.
3) After getting an access request, XACML engine validates the request against the predefined access control policies. If the decision is positive, XACML PDP publishes the decision response to message management module.
4) LoRa proxy subscribes for the decision from XACML PDP engine.
5) If decision response is positive, LoRa proxy extracts the necessary information from decision response, such as action on device and device EUID, and publishes those data to LoRa application server.
6) LoRa application server communicates with LoRa network server and sends the downlink message (required action that needs to be executed) to LoRa node (identified by device EUID).

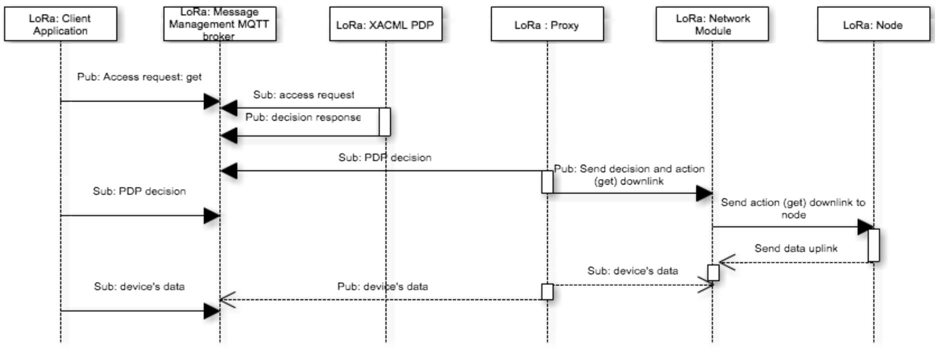

**Fig. 9.** LoRa: sequence diagram, get action request

2. Get. When user wants to get data from device (e.g. temperature, humidity, etc. sensors). “Get” action needs to be executed. Client application forms an access request and sends it to PDP XACML engine for validation. If the access request is granted, PDP engine sends the decision response to client application and LoRa proxy for further processing. LoRa proxy sends downlink message containing get action to LoRa application server where the message is further forwarded to LoRa node. Upon receiving the “get action” instruction, LoRa node starts sending uplink message (containing required information) to LoRa network server and application server. LoRa proxy subscribes for data sent by LoRa node and publishes it to message management module. Finally, client application subscribe data from LoRa node from massage management module.

The step-by-step information flow is presented below, see Figure 9.

*Figure 9:* step-by-step information flow explanation

1) When physical user executes an access request, client application forms the access request and publishes it to message management module (MQTT broker).
2) XACML engine subscribes the access request and is waiting for the callback from message management module.
3) After getting an access request, XACML engine validates the request against the predefined access control policies. If the decision is positive, XACML PDP publishes the decision response to message management module.
4) LoRa proxy subscribes for the decision from XACML PDP engine.
5) If decision response is positive, LoRa proxy extracts the necessary information from decision response, such as action on device and device EUID, and publishes those data to LoRa application server.
6) LoRa application server communicates with LoRa network server and sends the downlink message (required action that needs to be executed “get”) to LoRa node (identified by device EUID).
7) LoRa node sends uplink message to LoRa network module.
8) LoRa proxy module subscribes from LoRa network module the message sent by LoRa node.
9) LoRa proxy extracts the information from the message and publishes the required information to message management module.
10) Client application subscribes the data sent by LoRa node from message management module.

## 7.2 Bluetooth: Sequence diagram

Like LoRa, for Bluetooth technology, there are two sequence diagrams for two different sets of scenarios: Turn-on or off and get (get data from node). These two scenarios have different number of processing and communication between modules.

1. Turn-on or off. In the scenarios where turn-on or off is used, some processing steps in bluetooth are the same to that of LoRa. For example, the access request needs to be validated by XACMl PDP engine module. The different is that instead of LoRa proxy, Bluetooth proxy is responsible for communicating the access request message to bluetooth device or node. The step-by-step information flow is presented below, see Figure 10.

---

*Figure 10:* step-by-step information flow explanation

---

1) When physical user executes an access request, client application forms the access request and publishes it to message management module (MQTT broker).
2) XACML engine subscribes the access request and is waiting for the callback from message management module.
3) After getting an access request, XACML engine validates the request against the predefined access control policies. If the decision is positive, XACML PDP publishes the decision response to message management module.
4) Bluetooth proxy subscribes for the decision from XACML PDP engine.
5) If decision response is positive, Bluetooth proxy extracts the necessary information from decision response, such as action on device and device EUID, and publishes those data to Bluetooth network module.
6) Bluetooth network module communicate the message to bluetooth device/node.

---

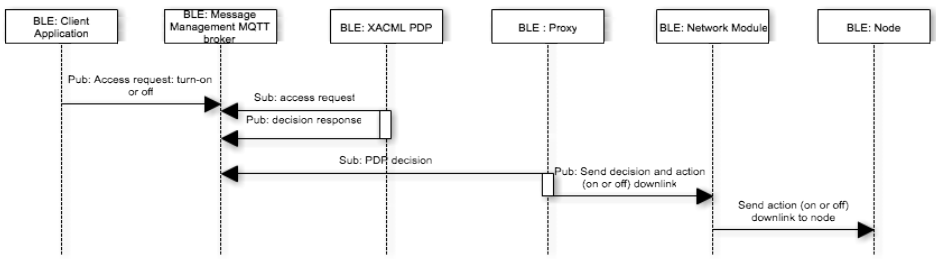
BLE: Client Application
BLE: Message Management MQTT broker
BLE: XACML PDP
BLE : Proxy
BLE: Network Module
BLE: Node
Pub: Access request: turn-on or off
Sub: access request
Pub: decision response
Sub: PDP decision
Pub: Send decision and action (on or off) downlink
Send action (on or off) downlink to node

**Fig. 10.** Bluetooth: sequence diagram, turn-on and off action request

2. Get. In case of "get" action, the process is similar to that of "get" in LoRa technology. The differences are at Bluetooth proxy and Bluetooth network. For Bluetooth network, when user requests an access to Bluetooth device, through client application, client application forms the request and communicates this request to XACML PDP engine where the request is validated against the predefined access control policies. If the decision is posi-

tive. XACML PDP publishes the decision response to message management module. Bluetooth proxy needs to subscribe for PDP decision response. After getting the decision response, Bluetooth proxy publishes the data extracted from decision response to Bluetooth network. Bluetooth network communication the action ("get") to Bluetooth node. Once node receives the "get action", it communicates data uplink with Bluetooth network module. The following step is the subscription of node's data from Bluetooth network by Bluetooth proxy. Once the Bluetooth proxy gets node's data, it publishes these data to message management module. Client application subscribes the node's data from message management module.

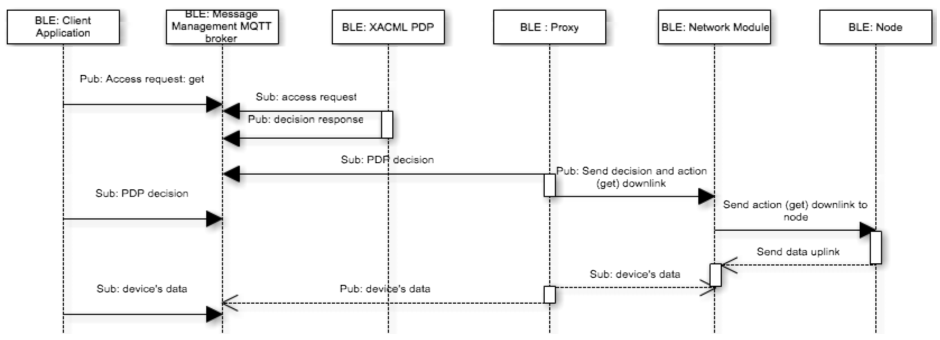

**Fig. 11.** Bluetooth: sequence diagram, get action request

The step-by-step information flow is presented below, see Figure 11.

*Figure 11:* step-by-step information flow explanation

1) When physical user executes an access request, client application forms the access request and publishes it to message management module (MQTT broker).
2) XACML engine subscribes the access request and is waiting for the callback from message management module.
3) After getting an access request, XACML engine validates the request against the predefined access control policies. If the decision is positive, XACML PDP publishes the decision response to message management module.
4) Bluetooth proxy subscribes for the decision from XACML PDP engine.
5) If decision response is positive, Bluetooth proxy extracts the necessary information from decision response, such as action on device and device EUID, and publishes those data to Bluetooth network.
6) Bluetooth application server communicates with LoRa network server and sends the downlink message (required action that needs to be executed "get") to Bluetooth node (identified by device EUID).

7) Bluetooth node sends uplink message to Bluetooth network module.
8) Bluetooth proxy module subscribes from Bluetooth network module the message sent by Bluetooth node.
9) Bluetooth proxy extracts the information from the message and publishes the required information to message management module.
10) Client application subscribes the data sent by Bluetooth node from message management module.

---

### 7.3 WiFi: Sequence diagram

In case of WiFi, there are also two sequence diagrams for two different sets of scenarios: Turn-on or off and get (get data from node). These two scenarios have different number of processing and communication between modules.

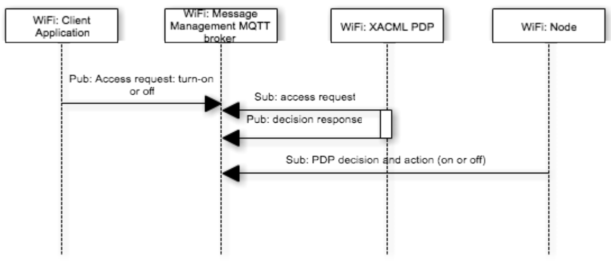
WiFi: Client Application
WiFi: Message Management MQTT broker
WiFi: XACML PDP
WiFi: Node
Pub: Access request: turn-on or off
Sub: access request
Pub: decision response
Sub: PDP decision and action (on or off)

**Fig. 12.** WiFi: sequence diagram, turn-on and off action request

1. turn-on or off. In case of WiFi technology, the process is easier compared with LoRa and Bluetooth. When user instructs the WiFi device to turn-on or off, the access request is formed at client application module and this request is forwarded to XACML PDP engine where the access request is validated. if XACML PDP provides the positive response, the decision response is published to message management module. WiFi node subscribes PDP decision in which device's EUID and the action needs to be executed are embedded. The step-by-step information flow is presented below, see Figure 12.

   ---

   *Figure 12:* step-by-step information flow explanation

   ---

   1) When physical user executes an access request, client application forms the access request and publishes it to message management module (MQTT broker).
   2) XACML engine subscribes the access request and is waiting for the call-back from message management module.

3) After getting an access request, XACML engine validates the request against the predefined access control policies. If the decision is positive, XACML PDP publishes the decision response to message management module.
4) WiFi node subscribes the decision response from message management module.

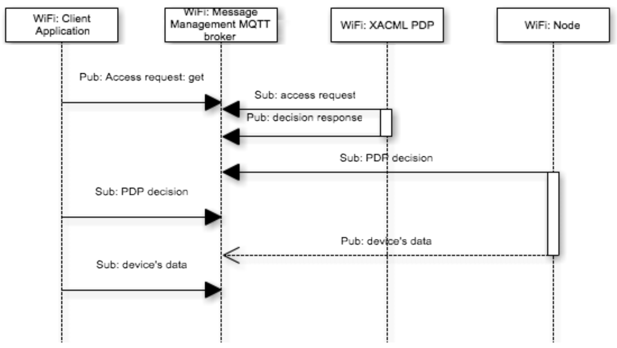

**Fig. 13.** WiFi: sequence diagram, get action request

2. Get. When user want to get data from WiFi device, the "get" request is instructed. Client application forms the request and send it to XACML PDP engine. Upon receiving the request, PDP engine validated the access request and publishes the decision response to message management module in case of positive decision. WiFi node subscribes the decision response. When node sees "get" action, it publishes the data back to message management module and finally client application subscribes the node's data. The step-by-step information flow is presented below, see Figure 13.

*Figure 13:* step-by-step information flow explanation

1) When physical user executes an access request, client application forms the access request and publishes it to message management module (MQTT broker).
2) XACML engine subscribes the access request and is waiting for the callback from message management module.
3) After getting an access request, XACML engine validates the request against the predefined access control policies. If the decision is positive, XACML PDP publishes the decision response to message management module.
4) WiFi node subscribes the decision response from message management module.

5) WiFi node publishes the node's data to message management module.
5) Client application subscribes node's data from message management module.

---

## 8 Conclusion

In this paper, we present the first draft of the design of smart hub. The presentation includes: smart hub protocol stack, the smart hub physical architecture and its functional architecture for prototype implementation. The messaging protocol and structure are also presented in this paper. The detailed sequence diagrams are presented for the access scenarios in three different wireless technologies: LoRa, WiFi and Bluetooth. We also developed a complete prototype software-based smart hub that can be used to connect three wireless network technologies, such as lora, wifi and bluetooth. Our future work is to focus on the design of policy administration point and refining the smart hub software.

## References

1. Bruce Ndibanje, Hoon-Jae Lee and Sang-Gon Lee. Security Analysis and Improvement of Authentication and Access Control in the Internet of Things. Open access Sensors, 14(8), 14786-14805; doi:10.3390/s140814786, 2014.
2. Rahul Godha, Sneh Prateek and Nikhita Kataria. Home Au- tomation: Access Control for IoT Devices. International Journal of Scientific and Research Publication, Volume 4, Issue 10, October 2014, ISSN 2250-3153.
3. Blase Ur, Jaeyeon Jung and Stuart Schechter. The current State of Access Control for Smart Devices in Homes. Workshop on Home Usable Privacy and Security (HUPS), July 24-26, 2013, Newcastle, UK.
4. Ricardo Neisse, Gary Steri, Igor Nai Fovino and Gianmarco Baldini. SecKit: A Model-based Security Toolkit for the Internet of Things. The journal of Computer and Security (2015), page 60-76. Published by ELSEVIER.
5. Sachin Babar, Parikshit Mahalle, Antonietta Stango, Neeli Prasad and Ramjee Prasad. Proposed Security Model and Threat Taxonomy for the Internet of Things. International Conference on Network Security and Applications (CNSA 2010). Recent Trends in Network Security and Applications pp 420-429. Published in Springer 2010.
6. J. Sathish Kular and Dhiren R. Patel. A survey on Internet of Things: Security and Privacy Issues. International Journal of Computer Applications. Volume 90. No 11, March 2014.
7. Assessement of Access Control Systems. National Institute of Standards and Technology. Technology Administration U.S. Department of Commerce. http://csrc.nist.gov/publications/nistir/7316/NISTIR-7316.pdf
8. Smart Home service providers. http://www.sensorsmag.com/components/top-10-smart- home-service-providers-us
9. Definition of cloud service. http://www.webopedia.com/TERM/C/cloud services.html

10. Agrawal, Rakesh and Imielinski, Tomasz and Swami, Arun. Mining Association Rules Between Sets of Items in Large Databases. SIGMOD Rec. June 1, 1993. Vol 22, No 2., New York, NY, USA.
11. Extensible Markup Language (XACML). http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html
12. M. Schiefer. Smart Home Definition and Security Threats. Ninth International Conference on IT Security Incident Man- agement IT Forensics. pages. 114-118, May 2015.
13. Jose L. Hernandez-Ramos, Antonio J. Jara, Leandro Marin and Antonio F. Skarmeta. Distributed Capability-based Access Control for the Internet of Things. Journal of Internet Services and Information Security (JISIS), volume: 3, number: 3/4, pp. 1-16.
14. N. Komninos and E. Philippou and A. Pitsillides. Survey in Smart Grid and Smart Home Security: Issues, Challenges and Countermeasures. IEEE Communications Surveys Tutorials. vol. 16, No. 4, P. 1933-1954, 2014.
15. Wireless Technologies. https://www.link-labs.com/blog/types- of-wireless-technology
16. Samsung smart thing . https://www.smartthings.com/
17. One M2M specification. http://www.onem2m.org/
18. Logitech harmony. http://www.logitech.com/en- us/product/harmony-hub
19. Google hub. https://on.google.com/hub/
20. PAP design and specification. https://doc.info.fundp.ac.be/mediawiki/index.php/FEDER-IDEE_PAP:_Policy_Administration_Point
21. PDP design and specification https://doc.info.fundp.ac.be/mediawiki/index.php/FEDER-IDEE_publication
22. Private LoRa server. https://github.com/brocaar/loraserver-setup
23. LoRa Alliance. https://www.lora-alliance.org/
24. Raspberry Pi. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
25. Internet of Things. https://www.thethingsnetwork.org/
26. Bluepy. https://github.com/IanHarvey/bluepy/tree/master/bluepy
27. MQTT Paho. https://pypi.python.org/pypi/paho-mqtt