# Domotic Module for the Internet-of-Things

Marco Soudo Cunha
marco.cunha@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

May 2017

**Abstract**

With the exponential growth of Internet-of-Things (IoT) usage and devices and with the increasing interest of users in acquiring these devices to improve comfort and optimize their household chores, it is necessary to develop a robust and secure home automation system that allows users to connect all these devices and manage their information in a flexible way, with low implementation costs. Thus, this paper presents a solution for a smart home system that only uses WiFi network for communication, avoiding intrusive installations (it does not need to add physical wiring), and allows to connect and control different devices, from any vendor, remotely from a mobile phone, a tablet or any other device with an internet connection. A complex and modular system has been developed, which uses standard technologies and protocols and consists in a central server (smart home server), which offers a set of services to control and manage the system; an auxiliary database to minimize information that devices need to store; a MQTT broker that allows the devices to communicate using the MQTT protocol; and home appliance devices whose requirements are minimum, they only must able to connect to an WiFi network.

**Keywords:** Internet Of Things, Domotic, Smart Home, Home Automation, WiFi Network, Interoperability, Security.

## 1. Introduction

As technology evolves it is desirable that the people's quality of life also improves. This means optimizing everyday tasks, reducing effort and time spent in boring or dissatisfying tasks. In order to achieve the desirable level of optimization, the devices we use on daily bases need to be *smarter*. In this context, smart means that they should be able, at least, to communicate with other devices. But this phase has already begun and the analysts predict that the IoT will comprise up to 26 billion interconnected devices by 2020[1].

A smart home, briefly, is a system containing a set of home appliance devices and services that work and communicate with each other to enhance the quality of life of their occupants. According to [3] in the near future, it is estimated that 90 million people around the world will live in smart homes, using technology to improve home security, comfort and energy usage.

Thus, if the quantity of available IoT devices is growing and the services are becoming more complex it is important to have robust protocols and a secure system that allows all of these devices to work properly together. This project main motivation is to give to the users the chance to easily deploy a system in their homes and control the smart home system from anywhere and from any device.

Instituto Nacional de Estatstica (INE)[2] published an article, where according to the 2015 Survey on Information and Communication Technologies Usage, 70% of the Portuguese households access internet at home. Most of them have WiFi routers, since they are usually provided by the Portuguese network providers. Thus, our second main motivation is to create a smart home system that can be deployed using the existing WiFi networks, allowing users to add or remove devices according to their needs.

Our smart home system should solve the major challenges that existing systems have and that are related with an intrusive installation, requiring to add physical wiring in their architectures, increasing the implementation costs; a lack of network interoperability and user interfaces inflexibility because they are normally limited to a single method of control and do not allow to extend or to create different interfaces.

Also, existing solutions are usually designed without security concerns, where these mechanisms are added *a posteori* leading to security flaws. Accord-

[1]http://www.gartner.com/newsroom/id/2636073 .Accessed: 2017-04-06

[2]https://www.ine.pt/ .Accessed: 2017-04-06

ing to all of these challenges, our main objective is to develop a secure and flexible smart home system, that does not require to know, *a priori*, the connected devices nor the user preferences, allowing users to add and remove devices *ad hoc* according to their needs.

Our system should be suitable to be applied in already existing buildings, using just WiFi networks to operate and for the devices communication. Thus, it is important, although not mandatory, that IoT home appliance devices can connect to an WiFi network. Our protocol should be open and hardware-independent working with any device, from any vendor, allowing the users to create or adapt their own devices.

## 2. Concepts and Related Work

The IoT, also called Internet of Everything, can be defined as the network of physical objects or "things" embedded with electronics, software, sensors and connectivity to enable objects to exchange data with each other and with the infrastructure.

In paper [5], authors presented three important requirements that an IoT system should have: comprehensive sense, the use of sensors to get information from the surrounding world; reliable transmission; and intelligent processing, a mechanism that analyze and process vast amounts of data and information with the purpose of add intelligence control to objects.

A smart environment, in the smart homes context, should provide an ubiquitous computing environment and operations in order to provide all the intelligence the users need, but in a continuously and imperceptibly fashion, for example, it might interconnect lighting and temperature controls with personal bio-metric monitors woven into clothing so that illumination and heating conditions in a room might be modulated according the users' preferences.

There are different network protocols that can be used in a smart home system architecture. The X10 industry standard, developed in 1975 for communication between electronic devices, is one of the oldest standards to control and manage home devices. This technology provides limited control over household devices through the home's power lines [2]. The power lines are used for signaling and control, where the signals involve brief radio frequency bursts representing digital information.

The great advantages of X10 are related with its inexpensive implementation cost and its simplicity. There is no need of qualified professionals to install a X10 system, because it does not require new or specialized wiring, which is suitable to install in existing buildings. X10 also allow users to control up to 256 devices and there is a big product range of devices available to purchase in specialist electrical shops.

The main disadvantages of X10 technology are related with its limited bandwidth and speed transmission. Since X10 signals can only transmit one command at a time, first by addressing the device to control, and then sending an operation for that device to perform, if two X10 signals are transmitted at the same time they may collide, which can lead to commands that cannot be decoded or trigger wrong operations. This protocol also lacks support for encryption and any form of error analysis.

Due to all discussed disadvantages we did not use X10 in our system, however since it is a popular technology, because it is one of the oldest and simplest standards for domotic solutions, our system allow users to include X10 devices, with gateways that allow the messages propagation between this technology and the WiFi network.

Another network protocol common used in smart home system is ZigBee[3] that is defined in [4] as a RF communication standard based on IEEE 802.15.4.A ZigBee-based network usually consists of a ZigBee coordinator and ZigBee nodes. The ZigBee coordinator is responsible for creating and maintaining the network and managing each ZigBee node in the network. All the communications between ZigBee nodes propagate through the coordinator to the destination node. The maximum ZigBee data rate is about 250kbps, and 40 kbps can meet the requirements of most control systems, and communication range can vary from 100m to 1km depending of the output power.

The main advantages of ZigBee, identified in [6] and [2] are related with the low implementation costs, low power and wider coverage and the wireless nature of ZigBee helps overcome the intrusive installation problem with the existing home automation systems. The low installation and running cost offered by ZigBee helps tackle the expensive and complex architecture problems with existing home automation systems.

There are already several ZigBee devices available on the market, and since it is a wireless technology with low implementation costs, it is important that our system offers an integration mechanism that allow users to deploy ZigBee devices.

### 2.1. MQTT

MQTT[4] is a publish-subscribe architecture on top of TCP/IP that allows bidirectional communication between a device and a MQTT broker. It is an extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks.

[3]http://www.zigbee.org/ .Accessed: 2017-04-06
[4]http://mqtt.org/ .Accessed: 2017-04-06

MQTT is more suitable for constrained devices with limited resources than HTTP because it has only a minimal packet overhead with a very light weight protocol and is extremely easy to implement on the client side.

The publish/subscribe (pub/sub) pattern in general, is an alternative to the client-server model. In pub/sub model, the publisher (who sends the message) is completely decouple from the subscriber (who receives the message), which means that the clients don't know about the existence of other clients. Thus, there is a third component, the Broker, which is known by both the publisher and subscriber. It handles and manages all the messages, filters and distributes them accordingly. Figure 1 illustrates the MQTT architecture.

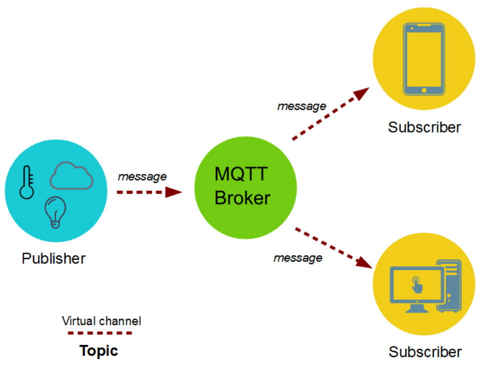

Figure 1: MQTT Architecture

In MQTT the messages are subject-based filtering, that means the filtering is based on a subject or topic which is part of each message. The client subscribes for the topics he is interested in and the broker distributes the messages accordingly. The topics are in general strings with an hierarchical structure, and allow different subscription levels, e.g. it is possible to use wildcards and subscribe home/kitchen/# to receive messages from all the topics related with the kitchen, such as home/kitchen/refrigerator or home/kitchen/temperature.

The MQTT protocol have Quality of Service(QoS) levels that allow to specify this delivery assurance level for each message or topic. This is very useful because we can have different messages with different QoS levels, according to its importance. There are three QoS levels available: QoS 0 (at most once), it is the minimal level and provides the same guarantee as the underlying TCP protocol, which means that a message won't be acknowledged by the receiver or stored and redelivered by the sender; QoS 1 (at least once), it is guaranteed that a message will be delivered at least once to the receiver, but it can also be delivered more than once; and QoS 2 (exactly once), this is the highest level, and it guarantees that each message is received only once by the subscriber. It is the safest but also the slowest quality of service level.

The different types of QoS levels should be assigned to the different messages, according to their importance. The higher the QoS level is, the slower and heavier the communication will become.

MQTT protocol also has a Retained Messages mechanism, where the broker store the last retained message for a specific topic. This feature allows a client that subscribes to a topic that has retained messages, to receive the last message immediately after subscribing. With this feature, clients do not have to wait until a new message is published to know the last known status of other devices.

Due to all the advantages that MQTT offers almost "out-of-the-box" this is a recommended protocol to use in the communication between constrained devices. Thus, we will apply this protocol in our project, for the devices communication, and also to provide flexibility to add new devices *ad hoc* with low effort.

## 2.2. Smart Home Control System based on Browse/Server Module

Paper [1] introduces a smart home control system based on Browser/Server architecture that provides flexibility, easy expansion and high reliability.

This system has a multilayer architecture, with a perception layer containing all the sensors/actuators; a network layer responsible for communication; and an application layer that supports the creation of smarter mechanisms and user interfaces.

In this project, authors choose ZigBee for the perception layer because it is a low-complexity, low-power and low-cost technology. The home gateway only provides the most basic interfaces and it consists of a ZigBee module, an USB communication module and an extensible interface module. The ZigBee module acts as a coordinator and it is responsible for the network establishing and maintaining. The wireless RF module is used for receiving the wireless signal and transmitting it to USB module through serial port.

The application layer and application support platform is composed by an website, a SQL database server and a command processing. The website provides an interface between the users and the system, allowing to check the home appliances information and to send commands to the smart home system.

The SQL database server is used for managing the information of household devices, users and control strategies.

Since this project was designed based in a multilayer architecture, it provides a great flexibility to add features and to work with a great variety of smart home technologies. It is not well defined how the authors solved some challenges related with real-time control, such as how they guarantee and maintain the correct state of the database or how they process and manage an higher number of commands, but we will take advantage of this multilayer architecture to create our system, adding more flexibility related with the user interface, allowing to control from any device or application and not only through an web-browser.

In this system, the home gateway only allows to the usage of ZigBee devices. In our system we have the goal to create an home gateway that can be used with any technology the user needs.

### 2.3. IoT with Webservices and Cloud Computing

The approach presented in paper [4] integrates IoT with cloud computing taking advantage of IoT to embed computer intelligence into home devices and cloud computing to provide scalable computing and storage power for developing, maintaining and running home services.

The architecture of this system consists in microcontroller-enabled sensors that measure home conditions; microcontroller-enabled actuators that receive commands and execute them; a database to store data from microcontroller-enabled sensors and cloud services, and acts as command queue being sent to actuators as well; server/API layer to process the data from the sensors and store in database and receive commands from the users to control the actuators and stores the commands in the database. The actuators make requests to consume the commands in the database through the server; an web application that enables to measure and visualize sensor data and control devices.

The authors choose Arduino for IoT devices, because it is a cheap and cross-platform with a simple programming environment. For the smart home network, authors choose ZigBee.

In order to communicate with the Cloud this projects uses JSON for data exchange, mainly because is is a lightweight data representation syntax, which is perfect to be handled by the IoT nodes that have limited computational resources.

The sensors readings transmit to the central server periodically, and the messages have the following format: {*source, 4, destination: 2,token: 1234, temp: 27, humidity: 120, proximity: 2207, ambient: 42*}, which it is self-explained by the JSON attribute names. The token field is an authentication mechanism used to verify that the message is sent from one of the system boards. This message is sent from the central transmitter board via ZigBee to the central receiver board, that is connected to the Internet and cloud services.

The web application is responsible for reading sensors, storing readings and monitoring home appliances. This web application is composed by two main parts: a front-end and a back-end. The front-end is the web client and the user interface, that was developed to be responsive and adapt to any screen size. The back-end offered support to computing services for logic processing and storage services for data storing.

The goals of this project and the technologies used are similar to what we will use in our system, because it uses JSON to exchange data, it has an home gateway that in this project is a central receiver board with ZigBee and Internet support, and it has a modular architecture. In our perspective this project only lacks in terms of security, because in the messages exchanged a single token is a limited solution and there is no information about the security mechanisms in web application access. Related with the flexibility to create external applications to control and monitor the smart home system, it only allows the web application developed by the authors, which limits the usage scenarios. Also, this architecture has a big performance issue, because the actuators make requests to server to check if there are new commands periodically. This is an additional overhead to the whole system due to the creation of more messages and the demand of actuators to be always in operation.

### 2.4. Smart Home System based on IPV6 and ZigBee

As introduced in article [6] it is possible to create a smart home network using IPV6 and ZigBee. This system is composed by three types of devices: coordinators, that manage and control home appliances; routers, responsible for message transmission; and end-devices, sensors/actuators connect with the home appliances.

In this project all the home appliances have their own IP address based on the IPV6, that allows to control and connect these devices directly from an external network. Thus, when the home gateway receives an external command, it unpacks it and gets the destination address, then selects the best routing path and transmits the message.

The internal network is based in ZigBee technology and all the home appliances must support this technology, because the appliances' status are periodically transmitted to the home gateway by a ZigBee module. The communication protocol used has three types of packets: request packets, response packets and event packets which are identified with the packet type field in the message header.

Each message has a name code (the appliance name, unique); a state code which is list of attributes and their values about device's status; a command code: 1-byte command code and input/output arguments; and an home code, unique, to make home network independent from other in packet level communication.

This project demonstrates that is possible to integrate ZigBee with IPV6, which are two protocols of network widely spread and cheap to implement. Although, the architecture presented does not have security mechanisms, that is a critical issue in a smart home architecture, and it has low flexibility and low support to integrate user interfaces, which is one our project goals.

### 2.5. The DomoBus System

The DomoBus System[5] is an academic project that provides a generic approach to home automation, independent of any technology. The specification is based in a standard language, XML, to describe the system, supporting inter-operation with different technologies.

This system offers an easy and extendable way to create and define scalable and flexible automation projects, with a distributed architecture. The authors have proposed a multi-level architecture, with *High-Level* components, Supervision Modules(SM), that are PCs or Raspberry-Pi like boards and perform supervision tasks, interface with users and offer remote access through an internet access point. The *Low-Level* components consist in Control Modules(CM), which are simple development boards plus interface electronics or power electronics to interconnect with physical devices - sensors and actuators and, optionally, a Router module(R) to allow to interconnect different DomoBus *low-level* network segments.

In the DomoBus system, a device is an abstract entity characterized by a set of *properties* where three standard operations over them are available: GET, read a property value; SET, modify a property value; and NOTIFY, each device can be configured to, autonomously, notify its DomoBus Supervisor (DS) when a property value changes, sending the new value information.

In our project we follow the same *Device Concept* to allow a generic approach and develop a system which can work with all types of sensors/actuators.

## 3. Implementation

The system architecture we propose is aligned and proper designed to be deployed in a multi-layer smart home system, as illustrated in Figure 2. Using a layered architecture we guarantee flexibility to add and remove components in our system and we provide interoperable services that can be specialized in the execution of specific tasks. This architecture allows to simplify the amount of data that *low-level* devices have to handle with, which is proportional to their hardware limitations.

[5]http://domobus.net .Accessed: 2017-04-06

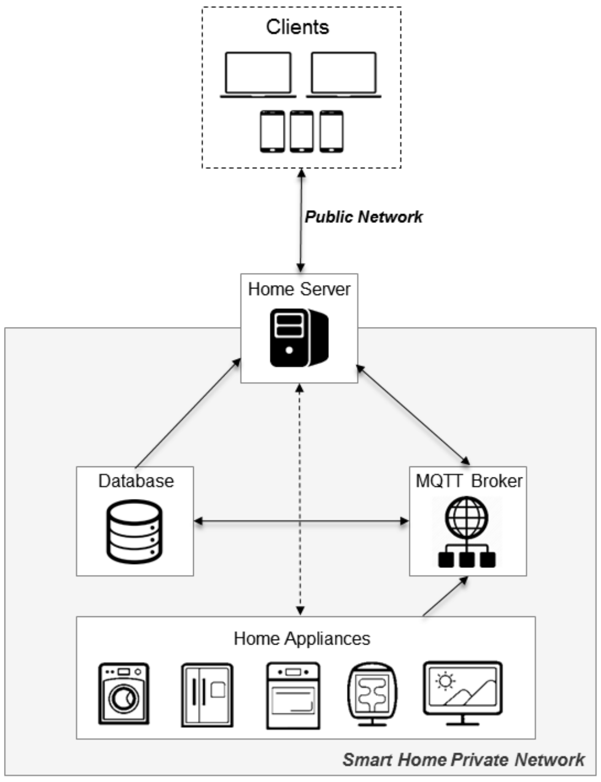

Figure 2: Smart Home System Architecture

### 3.1. Home Server

The Home Server is the core component in our system and it could be defined as a smart hub, since it allows the connection between different types of components from the internal network and also provides public access (from external network) to some of these components.

The Home Server, illustrated in Figure 2, is a complex system that includes smaller systems working together. The subsystems are described in the next subsections.

#### 3.1.1 Public Services

The home server provides public services in order to clients control and manage the smart home system. These services are mainly RESTful webservices that accepts HTTP requests with JSON data content and returns HTTP responses.

The public interfaces of our architecture were developed according to a Service-Oriented Architecture (SOA), where services are independent from each other and it is allow the creation of new ones without changing or stopping the system.

Another great advantage using RESTful webservices are related with the system scalability. The server end of REST is stateless, which means that the servers do not have to store state or data across requests. Thus, the communication between servers are minimal, making it highly scalable and possible to provide a good load balancer on the server-side that can easily redirect requests to different servers.

#### 3.1.2 Internal Devices Management

As discussed in the Subsection 3.2, we use the MQTT protocol to allow communication between devices, due to all of its advantages. But there is one great disadvantage in a Pub-Sub architecture: the subscribers must be aware, *a priori*, which topics are available to subscribe.

Thus, to take advantage of MQTT protocol and also to provide flexibility in our system we create an Internal Devices Management System that works together with the MQTT broker and manages all the devices connected, their components and their properties. With this system, the devices do not need to know, before being deployed in our network, what are the devices available and their properties.

The internal devices management system should always be synchronized with the real system information, thus it is mandatory that devices in their initialization connect with the home server (in the respective services) to register their information.

An example about how this internal device management system works is: an user wants to connect a chandelier (a device) in the smart home system. During the initialization the device should register itself in the home server using the private service *../api/device/devices*, and register their components and properties. So, the chandelier will also make a request to *../api/device/components* telling that it contains a Lamp (component) and this lamp has a Mode (property) that allows the values ON or OFF.

Using this mechanism, the Internal Devices Management system will always know the network state, the devices connected and their attributes. It allows to add new devices to the network *ad hoc* without resetting the entire network or updating the devices code.

#### 3.1.3 Complex Actions Management

In order to create an automated and smarter home system, the home server allow users to create complex actions. These actions are defined by a set of preconditions, or events (e.g. other devices' states) and a set of actions that are triggered when the events occur.

An example of how this system could work is: when the home temperature is 26C, turn on the air conditioner. It could more complex, such as, when the outdoor light sensor detects that exists a lack of sunlight and a person in a specific room, it will automatically close the electric window blinds and turn on ambient lighting.

This system is implemented using an approach that consists in a descriptive language that allow users to describe the complex actions and a polling system that will check periodically if the properties' values match with any conditions defined and, if they do, the system will execute the corresponding actions.

#### 3.1.4 User Permissions Management

This subsystem is responsible for managing the different user roles and different user permissions. In our system we propose an User Permission Management System that supports two types of user concepts: users and user roles. The first is profile that represents a specific user. The latter is a set of user groups that should be assigned to users' profiles, for instance, Guest, Regular User, Child, Gardener.

Our system also allows devices specific permissions and rooms permissions. In conceptual terms, a house has one or more rooms and each room has several devices. Thus, it is important to have an higher level of hierarchy, where users can control permissions regarding the rooms or more specifically, the devices.

To manage this feature, we apply a priority level mechanism where the permissions defined for a room have a lower priority than a device specific permission, as illustrate in Figure 3.

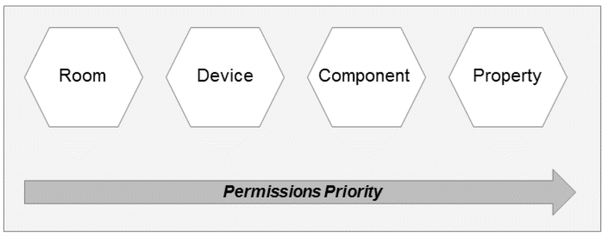

Figure 3: Permissions Priority, from lowest (left) to highest (right)

This mechanism allow the following scenario: The user Pedro, since he is a child, does not have permissions to control kitchen devices. So, their parents define a rule that deny accesses from user Pedro to room kitchen. But after some time, their parents want to give him access only to the kitchen television. Thus, they only need to add one rule that allows his user profile to manage the kitchen television, and the remaining kitchen devices will continue to be inaccessible from his user profile.

#### 3.1.5 Authentication / Security

The home server has a security layer requiring an authentication phase from clients to access the smart home system, using digital certificates. The communication should be done using HTTPS, which makes the conversation with the web server entirely encrypted.

### 3.2. MQTT Broker and MQTT messages

In the solution we propose, the devices communication is made using MQTT protocol due to all its advantages, that were already discussed in section 2.1. Thus, it is mandatory to have a MQTT broker (a MQTT server) that allows this communication.

As described before, the MQTT is a Publish-Subscribe protocol, and as any protocol of this type the subscribers need to know the topic names to subscribe. In our system we propose the following syntax regarding the topic names: {*Room*} / {*Device*} / {*Component*} / {*Property*}. For illustration purposes, an example of a topic could be: *kitchen / oven / temperature / value*

This structure provides a robust and a very flexible way to address devices, where the first level in the hierarchy is the room name. It allows to subscribe for an entire room and consequently all the devices "under" the room, to a specific device or even just to a specific component. It is very flexible because it allows to add more rooms or more devices without changing the old topics.

In our internal MQTT protocol we add an additional level to the topics to allow the home server or other devices to change a certain property value: */change*. The only subscribers of these topics are the devices that the property belongs to.

According to the example above, the topic where home server should publish new values whenever an user wants to change the oven temperature is: *kitchen / oven / temperature / value / change.*

Our MQTT broker is connected directly to the database and when a property value changes the MQTT broker updates the database with the corresponding value. A callback function on the MQTT broker side subscribes all *+/change* topics and updates the database with the new values whenever a new message is published.

Another MQTT feature that we use is its Quality-of-Service levels. In our system we allow devices to subscribe/publish messages from the three available QoS levels, but our recommendation is that critical messages should have the maximum QoS level: 2, that guarantees that messages are delivered and also it is a non-repeatable operation, which means messages are only sent once. One example of a message with a level 2 of QoS, could be a fire alarm. In this case, we want the system to keep trying to deliver the message until it finally reaches the right destination. The QoS levels of each topic are defined in the device registration phase and they could be easily changed afterwards. They are defined to a specific property, which provides great flexibility to manage the QoS levels according to the most fined grained level of devices. If the QoS level is not defined, the default value is 0 (the lowest and fastest one).

In our implementation we also take advantage of the MQTT retained messages. These messages are defined in the MQTT protocol and we use them with the purpose to let the other devices known when a device is disconnected and also to store the last topic message. The latter feature allows that a recently connected device can know immediately which is the last state of a specific property or component of other device, without waiting for a new message to get an updated state.

## 3.3. Database

An auxiliary database is needed to store all the system data, the user permissions, the devices connected and their state. The goal of using an external database is to reduce the amount of memory needed for server and devices, increasing the system's performance, and also to provide flexibility to scale the system. As discussed previously, the database are an auxiliary support of our system to improve the performance and allow horizontal scaling. In our implementation we use MySQL because it is free and it is more than suitable for our needs, but it is possible to use any other relational database system.

An explanation of some specific table fields is described in the next subsections. Notice that all the tables have an unique and auto-generated ID to simplify the message protocol.

### 3.3.1 Device

Each device is assigned to a specific room, referenced with *roomID*, and could have zero or more components. The *name* field is the one that will be used in the MQTT topic name, thus it can not have special characters nor whitespace characters. There is a field that is used for optimization purposes, the *lastSeen*, and its purpose is to store a time-stamp with the most recent date that the devices were seen. This allows to create smart mechanisms to avoid mistakes, such give information to an user that a specific device is available when it is not available, or even mechanisms to improve the performance, e.g. if a device was seen before a specific timeout, then the component's property value was the one stored in the database. The *description* field is used for user interaction purposes.

### 3.3.2 Component

In the component table we store information about the device that the components belongs to, in the *deviceID* field; the *name* of the component, for MQTT topic names, and a description that will be presented in an user interface. A component can have zero or more properties.

### 3.3.3 Property

Since each component can have zero or more properties, in the Property table we store information about the component that a property belongs to, *componentID* field. In this table we also have information about the property value, *value* field, and a description for user interface purposes. The field *qosLevel* is used to store information about the level of message delivery assurance. The acceptable values for this field are 0, 1 or 2 and the default is 0 (the lowest and fastest one), where there is no delivery assurance. The QoS level of 2 guarantees the delivery of a message and that each message is received only once by the counterpart. It should be used in important messages but it is also the slowest quality of service level.

#### 3.3.4 Room

A room can have multiple devices, that is why in Device table we store information about the *roomID*. In this table we have the *name* field which is used to create the MQTT topic name, thus it should not have special characters nor whitespace characters. The *description* field is used for user interface purposes.

#### 3.3.5 User

The User table store information about the users registered in our system. The *username* field should be unique and it represents the username that identify the user in the authentication phase. A user should have assigned an user role, stored in the *userRoleID* field.

#### 3.3.6 User Role

An user role may have assigned zero or more users. The User Role table have a *name* field with an user role profile name, e.g. GUEST, HOUSEKEEPER or ADMIN, the latter is created by default during the system initialization. The *description* field stores information related with the user role, for instance, with the scenarios where this user role should be used.

#### 3.3.7 Permission

The Permission table is the one responsible for storing information about all the permissions in our system. As described before, a permission could be configured at different levels, that is why this table have several foreign keys to specific IDs, that could have values or be empty (null) according to the desirable permission level. The validation of which permission should be used in each moment is done by the home server, as described in Subsection 3.1.4.

For each permission, there is a *mode* field that stores information regarding if it is read-only (RO), the default value, that only allows to read values, or if it is read-write (RW), that allows to read and also to change values.

### 3.4. Clients

The clients are the external users that connect remotely to the smart home system to control the devices or just to retrieve the devices' state. The access could be done using an application, with an user interface, or could be eventually M2M clients, e.g. an external IoT system. Since our system has an user permission mechanism in the home server, all the clients must have an unique identifier that match with the one defined in the user permissions, otherwise they will be assigned with the default user permissions. The permissions could include access rights to specific devices or to groups of devices, for instance it is possible to define that an user can access the values of all the kitchen devices, but can not change them.

Although not mandatory, the clients can have libraries to handle with JSON data. There are several ones, available for free and for different programming languages.

### 3.5. Devices

The devices are the IoT hardware components that are connected to the smart home system, usually constrained in terms of memory usage and power consumption, and are the sensors/actuators that clients will control remotely. There are three key concepts in our system architecture that are important to understand the following sections:

**Device**: An entity that represents a micro system, such an Arduino-board like, and have components.

**Component**: Something connected to a device, such a lamp, a toaster or an heater. Each component has specific properties.

**Property**: An attribute-value pair related to a specific component. For instance, a lamp can have two properties: Mode, with ON and OFF values and DIM, to control the brightness with a range from 1 to 10.

Our system was designed to enable the usage of any low-constrained device type in terms of memory and processing power. Thus our requirements are minimal and the complexity of our system is in the home server side, that is why a device should only be concerned about its own components and properties, substantially reducing their hardware requirements.

The minimum requirements for devices are an internet connection via WiFi; they should have implemented the TCP/IP protocol and be able to perform HTTP request/responses due to MQTT protocol and home server operations.

An additional feature that our system allows is the creation of *Gateways* to connect more basic devices that do not have the minimum requirements. The idea is that the user connects to one or more basic devices, a device slightly more powerful, e.g. an Arduino with an WiFi antenna, and this device acts as a gateway to connect the simple ones into our smart home system.

This is possible because in our system we can easily add or remove devices *ad hoc* and a gateway in the middle of the communication is transparent for the whole system. This gateway should be powerful enough to internally manage information related with the basic devices, e.g. the internal representation and the internal mapping used in the corresponding communication protocol.

## 4. Results

The microcontroller device used was the WeMos D1 R2 board that has an WiFi module embedded, the ESP8266EX chip, allowing the connection to an WiFi network. This board is compatible with the Arduino IDE and the programming language is the same as Arduino. Thus, we developed and uploaded in this board a program that has two main functions:

1. Setup: The program code has a *setup()* function, that only runs during the device initialization. Thus, in this function our device initializes and sets the initial values related with the components pins, the WiFi network properties, the home server address and the MQTT broker address. Then, it sends the registration messages to the home server, using WiFI, in order to register itself and all of its components and properties. Finally, the device connects to the MQTT broker and subscribes the topics related with its components.

2. Loop: the *loop()* function, as its name suggests, is a function that will be executed continuously until the board is switched off or reset, allowing our program to change and respond in each iteration. We have implemented in this function the MQTT protocol loop that checks, for each main loop iteration, if there are new MQTT messages from the subscribed topics. If it receives new messages, then the device will change its properties according to the content of these messages.

In our evaluation the smart home server, the MQTT broker and the MySQL database were implemented in the same device that was a Raspberry PI 2 Model B with an WiFi adapter. This device has a 900MHz quad-core ARM Cortex-A7 CPU, 1GB RAM, 16GB sdcard ROM, and because it has an ARMv7 processor, it can run the full range of ARM GNU/Linux distributions. We used the Raspbian OS, based on Debian and optimized for the Raspberry. To be able to evaluate the system performance, a simple test was performed, with one device, with one component (a Basic Red 5mm Led) and one property, *mode*: Off (0) or On (1).

In order to run this test, we used an WeMos D1 R2 board, similar to Arduino Uno, with a LED and a resistor connected to the D8 pin.

This board is compatible with the Arduino IDE and the programming language is the same as Arduino, the only difference is related with some specific libraries that need to be rewritten.

A LED works in a typical forward voltage and have a maximum rated forward current. These values are dependent of the LED type, diameter and color, and they always should be checked according to the LED manufacturer. The typical values for 5mm LED are illustrated in Table 1.

| **Color** | **Voltage (V)** | **Current (I)** |
|---|---|---|
| Green | 2.2 V | 20 - 25 mA |
| Red | 1.8 V | 15 mA |
| Yellow | 2.2 V | 20 - 25 mA |
| White | 2.8 - 4.2 V | 15 - 30 mA |

Table 1: LED typical values

In the current scenario, a red LED was used, which works with a voltage of 1.8V and a current of 15mA and the output voltage of the WeMos D1 R2 is 3.3V. Thus, according to the Ohm's Law:

$$V = R * I \Leftrightarrow R = \frac{V}{I} \tag{1}$$

Thus,

$$R = \frac{V_{in} - V_{led}}{I_{led}} \Leftrightarrow R = \frac{3.3 - 1.8}{0.015} = 100\ \Omega \tag{2}$$

Therefore a resistor of 100 Ohms was used to perform this test.

The most complex operation in our system is related with a client changing a property value, because it is an operation that evolves all the elements of our system: the client, the home server, the MQTT broker, the device and also the database. Thus, we ran a trial of 15 series where we recorded the request and the response times. In this trial all the devices were in the same network and the results are presented in Table 2.

The overall results were quite good, there was a short delay (in average, 1.57 seconds), with a duration range between 1.37s and 1.87s, due to the propagation time between client, home server, MQTT broker and device and also because the device is constrained in terms of power computational resources, thus it takes some time to update its component and publish a new MQTT message.

## 5. Conclusions

The main goal of this paper was to present a system that allows users to easily connect different IoT home appliances, regardless of their manufacturer, and control them remotely from any device with an internet connection. It was necessary to create a

| **Request** | **Response** | **Duration (ms)** |
|---|---|---|
| 20:10:12.320 | 20:10:13.922 | 1602 |
| 20:10:14.110 | 20:10:15.533 | 1423 |
| 20:10:17.320 | 20:10:19.001 | 1681 |
| 20:10:20.796 | 20:10:22.255 | 1459 |
| 20:10:23.453 | 20:10:24.826 | 1373 |
| 20:10:26.877 | 20:10:28.253 | 1376 |
| 20:10:29.057 | 20:10:30.786 | 1729 |
| 20:10:32.445 | 20:10:33.997 | 1552 |
| 20:10:35.003 | 20:10:36.438 | 1435 |
| 20:10:37.841 | 20:10:39.412 | 1571 |
| 20:10:40.067 | 20:10:41.798 | 1731 |
| 20:10:43.129 | 20:10:44.752 | 1623 |
| 20:10:46.062 | 20:10:47.932 | 1870 |
| 20:10:48.963 | 20:10:50.665 | 1702 |
| 20:10:52.352 | 20:10:53.755 | 1403 |

Table 2: Change Properties Trial

simple, more flexible protocol that would guarantee interoperability and security in the control and administration of this system.

Our system offers an inexpensive solution, with low implementation costs and easy installation, that allows any user to deploy it in his home. Our solution uses the WiFi network as communication technology, since most of the users already have in their homes and also allows to reuse existing devices by adding only a simple microcontroller, similar to an Arduino, that connects these devices to the internet.

Despite being versatile, our system is robust and offers security mechanisms so that users can install it with confidence. These security mechanisms result from the designed architecture that is transparent to the devices within the network, and allows differentiating two types of networks: the external network, where users connect remotely to control the devices after being properly authenticated, and the internal network restricted to home devices communication and administration purposes.

Because the system architecture was designed to be highly scalable, there is no limit regarding the quantity of devices that users can connect. It is designed according a service-oriented architecture with RESTful webservices, independent of each other, with a MQTT broker for internal communication, that allows lightweight and simple messages between devices and an auxiliary database that stores all the system information, improving the system performance.

Another feature that our system architecture offers is the chance to the users create complex actions in order to automate their home appliances. They may create a set of preconditions, usually related with the state of other devices, that when they are checked trigger certain actions on other devices. For example, we can define an action that consists if the outdoor temperature is over 26C and users are at home, then set the air conditioner temperature to 21C.

Our project can be the starter point to future projects that have the goal to create a flexible and truly smart, home system. Thus, some ideas that will improve the whole system might be: Add machine learning algorithms that are able to learn the users' preferences and make predictions about future actions or events; Create scenarios, that store information about a group of devices and their properties allowing users to set quickly predefined values for each device; Create gateways to allow devices that are not able to connect to WiFi networks, to connect and use our system; Allow to add more hierarchical levels inside a house, instead of just rooms; Allow to change the protocol syntax, having a descriptive language that map a different protocol to our system's protocol; Create an user interface, responsive and adaptive to multiple screen sizes.

## References

[1] G. Chong, L. Zhihao, and Y. Yifeng. The research and implement of smart home system based on internet of things. In *Electronics, Communications and Control (ICECC), 2011 International Conference on*, pages 2944–2947. IEEE, 2011.

[2] K. Gill, S.-H. Yang, F. Yao, and X. Lu. A zigbee-based home automation system. *IEEE Transactions on Consumer Electronics*, 55(2):422–430, 2009.

[3] A. Jacobsson, M. Boldt, and B. Carlsson. A risk analysis of a smart home automation system. *Future Generation Computer Systems*, 56:719–733, 2016.

[4] M. Soliman, T. Abiodun, T. Hamouda, J. Zhou, and C.-H. Lung. Smart home: Integrating internet of things with web services and cloud computing. In *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 2, pages 317–320. IEEE, 2013.

[5] M. Yun and B. Yuxin. Research on the architecture and key technology of internet of things (iot) applied on smart grid. In *Advances in Energy Engineering (ICAEE), 2010 International Conference on*, pages 69–72. IEEE, 2010.

[6] Z. Zou, K.-J. Li, R. Li, and S. Wu. Smart home system based on ipv6 and zigbee technology. *Procedia Engineering*, 15:1529–1533, 2011.