TÉCNICO LISBOA

Domotic Module for the Internet-of-Things

Marco Soudo Cunha

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Renato Jorge Caleira Nunes

Examination Committee

Chairperson: Prof. Ricardo Jorge Fernandes Chaves
Supervisor: Prof. Renato Jorge Caleira Nunes
Member of the Committe: Prof. Alberto Manuel Ramos da Cunha

May 2017

The world hates change, yet it is the only thing that has brought progress.

Charles Kettering

Acknowledgments

First of all, I would like to thank the Academy, that helped me growing as an engineer student, but most important, developing my scientific habits of reasoning and stimulate my critical thinking. Additionally, I would like to thank my advisor, Prof. Renato Nunes, for all the cooperation and clever advice.

And last but not least, I would like to thank my parents, Marco and Isabel, my brother, Bernardo, as well as my girlfriend, Darlene, for all the patience, attention and care over during my entire academic journey.

Thank you all.

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, the objective of this work is to present a solution for a smart home system that only uses the 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 this protocol, the MQTT, and finally the devices whose requirements are minimum, they only must able to connect to an WiFi network.

The system does not limit the quantity of connected devices since its architecture was designed and implemented to allow horizontal scalability and high availability to handle with a growing amount of devices.

Keywords

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

Resumo

Com o crescimento exponencial da quantidade de dispositivos eletrónicos preparados para estarem ligados à internet (IoT) e com o interesse cada vez maior dos utilizadores em adquiri-los para melhorar o conforto e otimizar as suas tarefas domésticas, torna-se necessário o desenvolvimento de um sistema de domótica que permita gerir e conectar todos estes dispositivos e a sua informação de uma forma robusta, segura e flexível, e com baixos custos de implementação.

Assim, o objetivo deste trabalho é apresentar uma solução para um sistema de domótica que apenas utiliza a rede WiFi, evitando assim instalações intrusivas, e que permite ligar diferentes dispositivos, independente do fabricante, e controlá-los remotamente a partir de um telemóvel, de um tablet ou qualquer outro aparelho com ligação à internet.

Para tal, foi desenvolvido um sistema complexo mas modular que recorre a tecnologias e protocolos standard e que é constituído por um servidor central (smart home server), que oferece um conjunto de serviços para controlar e gerir o sistema; uma base de dados auxiliar para minimizar a informação que os dispositivos precisam de guardar; um MQTT broker que é um servidor que permite que os dispositivos comuniquem entre si utilizando este protocolo, o MQTT, e por fim os dispositivos cujos requisitos são mínimos tendo apenas que se conseguir ligar a uma rede WiFi.

O sistema não limita a quantidade de dispositivos conectados, uma vez que a apresenta uma arquitetura projetada e implementada para permitir escalabilidade horizontal e alta disponibilidade para lidar com uma quantidade crescente de dispositivos.

Palavras Chave

Internet das Coisas, Domótica, Casas Inteligentes, Automatização Residencial, Internet sem-fios, Interoperabilidade, Segurança.

Contents

List of Figures

List of Tables

Abbreviations

IoT Internet-of-Things

RF Radio Frequency

BLE Bluetooth Low Energy

GSM Global System for Mobile communication

M2M Machine-to-Machine

WSDL Web Services Description Language

SOAP Simple Object Access Protocol

HTTP Hyper Text Transport Protocol

XML eXtensible Markup Language

JSON JavaScript Object Notation

REST REpresentational State Transfer

MQTT MQ Telemetry Transport

MAC Medium Access Control

QoS Quality of Service

SOA Service-Oriented Architecture

RFID Radio-Frequency IDentification

TCP Transmission Control Protocol

HTTPS Hyper Text Transport Protocol Secure

1

Introduction

Contents

1.1 Motivation

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 Internet-of-Things (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 [Jacobsson et al., 2016] 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, with low implementations costs, and control the smart home system from anywhere and from any device, such a mobile phone, a laptop or a smartwatch.

We want people to reuse, at maximum, what they already have in their houses, and Instituto Nacional de Estatística (INE) published an article[2] where according to the 2015 Survey on Information and Communication Technologies Usage in households and by individuals, 70% of the Portuguese households access internet at home, mainly via broadband access. The access to the internet by broadband prevails among households with children (90%) and among those living in the region of Lisbon (78%). Thus we can see that almost 70% of Portuguese people have internet access in their homes, and most of them have WiFi routers, since they are commonly 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, avoiding hard-installation processes and allowing users to add or remove devices from the system according to their needs.

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

[2] https://www.ine.pt/xportal/xmain?xpid=INE&xpgid=ine_destaques&DESTAQUESdest_boui=224732582&DESTAQUESmodo=2&xlang=en .Accessed: 2017-04-06

1.2 Objectives

This project aims to develop a smart home system to connect and control several IoT home appliance devices providing automation processes and optimizing the users' tasks. Our smart home system should solve the major challenges that existing systems have, and these challenges are related with:

1. Intrusive installation: some systems require add physical wiring in their architectures, increasing the implementation costs and requiring qualified professionals to install the system;
2. Lack of network interoperability: some home networks are implemented in an unplanned manner, leading to a complex maze of heterogeneous networks that working together do not guarantee network interoperability;
3. Interface inflexibility: user interfaces normally are limited to a single method of control and do not allow to extend and create different user interfaces;
4. Security and Safety: usually, existing solutions are designed without security concerns, these mechanisms are added *a posteriori* leading to security flaws;
5. Proprietary Protocols: these systems are designed with commercial purposes, to be profitable, thus they use proprietary protocols that only allow to connect licensed products.

According 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 must have low implementation costs and it 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.

Regarding the user interface, our system must allow several application types, such as smartphone applications, smartwatch applications, desktop or web-based applications, or even applications that involves Machine-to-Machine (M2M) communication, e.g. other smart home systems. The idea to achieve this flexibility is to create public services with a simple and light-weight protocol that can be consumed by any of the referred interaction types through the internet.

The system should be designed having security concerns from the initial design phases. There are four different types of security in a smart home system: the external security, related with the public services available online; the internal network security, which means the private network should only be accessible for the home appliances and administration purposes; the message delivering assurance, critical messages must be delivered or they can lead to safety flaws, such an alarm message alerting an intrusion for instance; and the last security type is related with the fault tolerance and

recovery from faults, where our system should cover a default behaviour for each device when the system is offline for some reason or not working properly.

Another feature that our system should provide is a mechanism that allows different device permissions for each user or user role. This feature is very useful for families where it is not desirable that all the family members have the same responsibilities nor the same permissions to manage devices.

In order to achieve all the previous goals, our system should be designed in a modular and flexible architecture, having different components assigned to specific tasks. Thus, this architecture should allow horizontal scalability as the system becomes more complex with more devices connected.

1.3 Document Structure

This document is organized as follows:

- Chapter 2 - Concepts and Related Work: presents some related concepts and technologies and gives an overview of the state of the art in the areas related to this project.
- Chapter 3 - Smart Home System: presents the architecture of the system, explaining the multiple system components and the relations between them, our implementation and evaluation.
- Chapter 4 - Conclusions: presents conclusions and a final overview about the work done and some proposals for future developments that could improve the system.

2

Concepts and Related Work

Contents

2.1 Concepts

In the following subsections we present important concepts related with our project subject.

2.1.1 IoT

The term IoT was first used by Kevin Ashton in 1999 in the context of supply chain management [Ashton, 2009]. However, in the past decade this concept has been more inclusive referring to a network of Internet-enabled objects connected to the Internet based on conventional protocols, to exchange information and communicate, in order to achieve intelligent identify, locate, track, monitor and manage a network.

Thus, it is possible to define IoT, also called Internet of Everything, 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.

According to [Yun and Yuxin, 2010], an IoT system should have three important requirements:

1. Comprehensive sense: The use of sensors to get information from the surrounding world;
2. Reliable transmission: Accurate real-time delivering information;
3. Intelligent processing: A mechanism that analyze and process vast amounts of data and information with the purpose of add intelligence control to objects.

These requirements, could be designed in an abstract architecture of IoT that split them into three different layers:

1. Perception layer: set of different types of sensors, responsible for the perception and identification of objects, and collecting and storing information.
2. Network layer: Converged network formed by different communication networks.
3. Application layer: Intelligent application to control and monitor all the data gathered.

M2M communication is the base of IoT, but is expected that IoT offers advanced connectivity of devices, systems and services that goes beyond M2M and covers a variety of protocols, domains and applications. Some categories of services that IoT should provide are:

- Network service: goods identification, communication and positioning;
- Information service: information collection, storage and query;
- Operation service: remote configuration, monitoring, operations and control;
- Security service: user management, access control, event alarm, intrusion detection, attack prevention;
- Management service: fault diagnosis, performance optimization, system upgrades, billing management services.

2.1.2 Smart environments

According to [Bélissent, 2010] a smart environment uses information and communication technologies to make the critical infrastructure components and services of a city administration, education, health-care, public safety, real estate, transportation and utilities more aware, interactive and efficient. Thus the concept `smart` here refers to something that has potential to interconnect different components and control each one, to optimize critical or regular processes.

A smart environment should also 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.

2.1.3 Home Automation / Smart home

[Gill et al., 2009] describes home automation as the introduction of technology within the home to enhance the quality of life of their occupants, through the provision of different services. It is desirable that smart home automation systems incorporate common devices to control home appliances, but they do not only turn devices ON and OFF [Ricquebourg et al., 2006].

A smart home system based on IoT, according to [Chong et al., 2011], should have at least the following features:

- Compatibility of different communication technologies: allow to add objects that use different technologies. This could be done creating communication interfaces in the home gateway;
- Ubiquitous service: no matter where the users are, the real-time smart home information should be always available to be obtained conveniently;
- Comprehensive perception: using a variety of physical and logical sensors, it should be able to monitor in real-time the home information;
- Conveniently control: the smart home system should be controlled by almost any device, such as a mobile phone, a smartwatch, a personal computer and the control results can be real-time displayed through all sorts of visual interfaces.

2.1.4 Cloud Computing

According to [Ye and Huang, 2011], cloud computing is a model for enabling convenient, on demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

The main advantages of this model are related with the service and infrastructures costs and the fast and efficient adaptation to changes in load balance throughput, dynamically managing computing capabilities and storage capacity.

Our solution should allow the increase of connected devices without requiring to change the system configuration. Thus, our system should be developed according to a cloud computing model that guarantees the rapidly provisioning provisioned of resources on-demand according to the system workload.

2.1.5 Web Service

Web service is a software system designed to support interoperable M2M interaction over a network[1]. Usually it has an interface described in a machine-processable format (Web Services Description Language (WSDL) and the other systems interact with the webservice using Simple Object Access Protocol (SOAP) messages, typically over a Hyper Text Transport Protocol (HTTP) or HTTPS, with an eXtensible Markup Language (XML) or JavaScript Object Notation (JSON) serialization.

The webservices can also have a REpresentational State Transfer (REST)ful architecture, which is more flexible because they do not require to have an interface with all the available operations and the data structure.

We introduced the webservices architecture design, because one of our project's goals is to create a flexible system architecture that allows the implementation of different user interfaces according to the user needs. Thus, this type of architecture offers a design pattern that only provides the services, which is the model layer, containing all the system data, logic and rules, decoupling this information from the user interface.

[1] `https://www.w3.org/TR/ws-gloss/` .Accessed: 2017-04-06

2.2 Technologies

In the following sections we present different technologies that can be used to create a smart home system architecture. For each one, we discuss the advantages and disadvantages and if they are suitable to reach our project goals.

2.2.1 Radio-Frequency IDentification (RFID)

RFID uses electromagnetic induction or electromagnetic propagation for the purpose of non-contact automatic identification of objects or humans. RFID has secure mechanisms against reproduction, combined encryption and secure data to avoid counterfeiting [Yun and Yuxin, 2010].

The main advantages of RFID are related with its simple usage and installation: the system only needs RFID tags and RFID readers, and this is a technology suitable to support tracking physical objects with well-defined confines (such as warehouses) [Kortuem et al., 2010].

RFID is limited regarding the sensing capabilities and deployment flexibility since it it as a proximity technology. Thus, the RFID technology is more oriented to identify objects or people instead of control devices and their properties. It can be used in a smart home system to identify users, for instance, but this feature is out of our project's scope.

2.2.2 Bluetooth Low Energy (BLE)

BLE, part of the Bluetooth v4.0 standard, is one of the more promising new technologies in the devices, included in almost all the latest smartphones. It allows small and low-cost tags to announce their presence by transmitting an advertising packet once per second, that enables them to operate for up to one year on a lithium coin cell battery. This new technology creates more opportunities for ubiquitous deployment, higher-accuracy tag reads and the ability to blend in invisibly with a product [Want et al., 2015].

We will not use this technology in our project, due to similar reasons as RFID. Although BLE provides greater coverage than RFID and this technology could be used for limited communication between devices.

2.2.3 Global System for Mobile communication (GSM)

GSM is an open, digital cellular technology used for transmitting mobile voice and data services[2]. It is the global standard for mobile communications – with over 90% market share, operating in over 219 countries and territories.

GSM is one of the most secure cellular telecommunications system available, and it maintains end-to-end security by retaining the confidentiality of calls and anonymity of the GSM subscriber. The GSM security protocols assigned a temporary identification to the subscriber's number to keep the user's privacy. The communication privacy is achieved applying encryption algorithms and frequency hopping that can be enabled using digital systems and signalling.

[2] `https://www.gsma.com/aboutus/gsm-technology/gsm` .Accessed: 2017-04-06

The advantages of GSM are related with its wide spread coverage, even where Internet may not be available, which makes the whole system online for almost all the time and its high security infrastructure, which provides maximum reliability so that the information sent or received can not be monitored by an eavesdropper [Yuksekkaya et al., 2006].

The disadvantages in the use of GSM as communication protocol in a smart home system are the additional implementation and usage costs, requiring an external GSM service provider to be able to interconnect all the devices. Also, GSM coverage is limited in underground places. For this reason we will not apply this technology in our project.

2.2.4 X10

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 [Gill et al., 2009]. 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 allow users to control up to 256 devices and there is a big product range of devices available to purchase in specialist electrical shops. Also, X10 it is a proven technology that has been used in real systems for over 20 years.

The main disadvantages of X10 technology are related with its limited bandwidth. 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.

Another disadvantage of X10 is related with its speed transmission. The X10 protocol is slow, up to one second in sending signals, and it does not allow more complex forms of communication, e.g. selecting a channel on a television. X10 protocol also lacks support for encryption and any form of error analysis.

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

2.2.5 ZigBee

According to [Soliman et al., 2013] ZigBee[3] is a Radio Frequency (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 man-

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

aging 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.

[Gomez and Paradells, 2010] explain that the ZigBee protocol stack is composed by four main layers: the physical (PHY) layer, the Medium Access Control (MAC) layer, the network (NWK) layer and the application (APL) layer. Zigbee also provides security functionality across layers.

In a Zigbee system there are three device roles:

- The ZigBee coordinator, which corresponds to an IEEE 802.15.4 Personal Area Network (PAN) coordinator;
- The ZigBee router;
- The ZigBee end device, which is normally a simple device with low capabilities.

The ZigBee NWK layer supports addressing and routing for the tree and mesh topologies. The tree topology is rooted at the ZigBee coordinator and is adequate for data collection. This scheme includes mechanisms for address assignment, that facilitate multi-hop data delivery.

In the mesh topology, routes are created on demand and are maintained using a set of mechanisms based on the *ad hoc* on-demand distance vector (AODV) routing protocol.

The main advantages of ZigBee, identified in [Zou et al., 2011] and [Gill et al., 2009] 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.

2.2.6 Z-Wave

Z-Wave is a wireless network protocol architecture developed by ZenSys and promoted by Z-Wave Alliance for automation in residential and light commercial environments. The main purpose is to allow reliable transmission of short messages from a control unit to one or more nodes in the network [Gomez and Paradells, 2010].

The Z-Wave protocol has an architecture of five main layers: the PHY, MAC, Transfer, Routing and Application Layers.

The Z-wave radio mainly operates in the 900MHz ISM bands and allows transmission at 9.6 and 40 kb/s data rates.

Z-wave has a mechanism to avoid collisions that checks if the channel is available before start a transmission. If the channel is busy or not available, the transmission is deferred for a random period of time.

The Transfer layer manages the communication between two consecutive modules and provides an optional re-transmission mechanism based on ACKs.

There are two types of devices in a Z-Wave system: controllers, that poll or send commands to slaves, which reply to the controllers and execute the commands.

The Z-Wave routing layers performs routing based on a source routing approach. That means when a controller transmits a packet, it includes the path to be followed by the packet. The controller maintains a table that represents the full topology of the network.

One of the Z-Wave problems is its limited coverage, that requires more z-wave devices to cover larger regions, increasing the implementations costs. Also, Z-Wave only supports 232 devices instead of 65000 that ZigBee support.

Thus, Z-Wave is out of our project's scope but it is another wireless network solution for who wants to create a wireless smart home system. Nevertheless, our system should support the integration of alternative wireless network protocols and the Z-Wave should be contemplated.

2.2.7 INSTEON

INSTEON is a solution developed for home automation by SmartLabs and promoted by the INSTEON Alliance, that defines a mesh topology composed of RF and power line links, which allows devices to operate through RF-only, power line-only or both [Gomez and Paradells, 2010].

INSTEON RF signals use frequency shift keying (FSK) modulation at the 904 MHz, with a raw data rate of 38.4 kb/s.

INSTEON devices are peers, which means that any of them can play the role of sender, receiver, or relayer.

The communication between devices that are not within the same range is achieved using a multi-hop approach (up to four hops) that consists in all devices re-transmit the messages they receive, unless they are the destination of the messages.

The multi-hop transmission is performed using a time slot synchronization scheme, defined by a number of power line zero crossings, where transmissions are allowed in certain time slots, and devices withing the same range do not transmit different messages at the same time. The RF devices that are not attached to the power line can transmit asynchronously.

The INSTEON main disadvantage is related with its expensive implementation costs and devices prices, and for this reason it is out of our project's scope.

2.2.8 Wavenis

Wavenis is a wireless protocol stack developed by Coronis Systems and promoted by Wavenis Open Standard Alliance (Wavenis OSA) for control and monitoring applications [Gomez and Paradells, 2010].

Wavenis operates mainly in the 433 MHz, 868 MHz and 915 MHz bands, which are ISM bands in Asia, Europe and United stands respectively. The data rates are between 4.8 kb/s and 100 kb/s, with 19.2 kb/s being the typical value.

The Wavenis MAC sublayer offers synchronized and non-synchronized schemes. In a synchronized network nodes allocate a time slot that is a pseudo-randomly calculated based on its address. Before transmission, the node checks if the channel is busy. If it is, it computes a new time slot and adds a delay in the message transmission.

Wavenis is an interesting wireless network protocol but it is not popular, thus the amount of available that support this technology is reduced. For this reason, we will not cover this technology in our project.

2.2.9 MQ Telemetry Transport (MQTT)

MQTT [MQT] 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. The design principles are to minimize network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery. 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 2.1 illustrates the MQTT architecture.

It is possible to identify three types of decoupling in a pub/sub model:

- Space decoupling: publisher and subscriber do not need to know each other (the IP address for example);
- Time decoupling: publisher and subscriber do not need to run at the same time;
- Synchronization decoupling: the operations on both components are not ceased during publish or receiving.

One of the major advantages on a pub/sub model is its scalability. The operations on the broker can be highly parallelized and processed event-driven and it is possible to apply caching and intelligent routing of messages. This can be achieved using clustered broker nodes and having good load balancers to distribute the load over different servers.

In a generic pub/sub model it is possible to filter messages according different types of filters. 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.

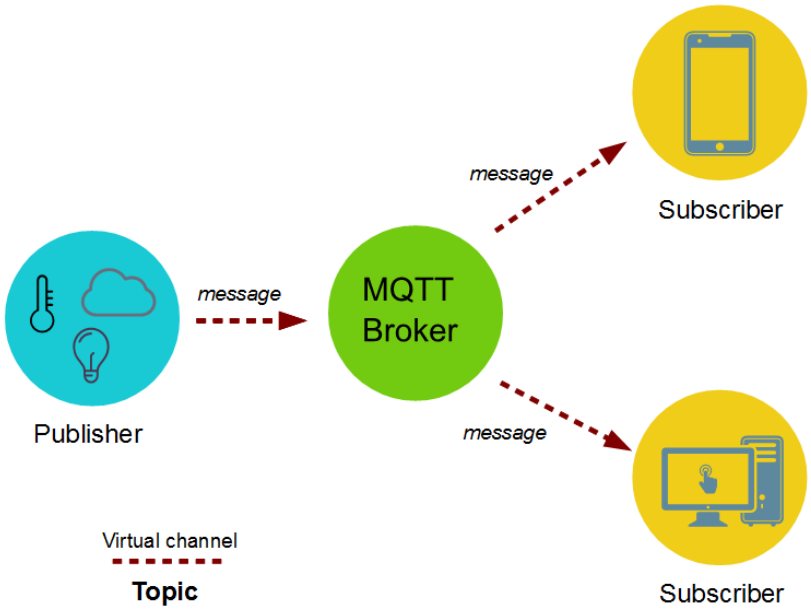

Figure 2.1: MQTT Architecture

An important feature of any communication protocol to be used in IoT devices is how the protocol implements the delivery assurance mechanism. The MQTT protocol have the Quality of Service (QoS) levels which allow to specify the level we want in each message or topic. This is very useful because we can have different messages with different QoS levels, according to its importance. In MQTT there are 3 QoS levels:

- QoS 0 - at most once: this 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.
- 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. It is worth mentioning that the higher the QoS level is, the slower and heavier the communication will become, therefore it is important to achieve a good balance.

Besides all the advantages of the MQTT that we have discussed, there is one more that can be very useful for an IoT system: Retained Messages. The MQTT protocol includes a 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.

2.2.10 HTTP and Hyper Text Transport Protocol Secure (HTTPS)

The HTTP is an application-level protocol for distributed, collaborative, hypermedia information systems and it is the foundation of data communication for the World Wide Web (WWW). The development of HTTP was initiated by Tim Berners-Lee at CERN in 1989. This protocol is a generic, stateless, protocol which can be used for hypertext, name servers and distributed object management systems, through extension of its request methods, error codes and headers [Fielding et al., 1999].

The HTTP as a important feature that allows the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred. Its definition requires an underlying and reliable transport layer protocol where Transmission Control Protocol (TCP) is commonly used.

Since HTTP is not encrypted, it is vulnerable to *man-in-the-middle* and *eavesdropping* attacks.

The HTTPS is an implementation of HTTP protocol over an additional security layer that uses the SSL/TLS protocol. This layer provides a secure channel, where all the data is encrypted, and it also allows to use digital certificates to validate the authenticity from the client and from the server. The main motivations for HTTPS usage are related with its authentication mechanisms and the data privacy and integrity protection.

The HTTP is the foundation of data communication for the WWW, thus we will use this protocol in our system, with the additional security layer (HTTPS), because it allows clients to easily connect to our system in a secure way.

2.2.11 REST

REST or RESTful webservices is a scalable architecture which provides interoperability between computer systems on the Internet. REST-compliant web services allow requesting systems to access and manipulate textual representations of web resources, using an uniform and predefined set of stateless operations.

Due to all of these advantages and according the flexibility that we want to accomplish, we will use RESTful webservices to provide services that will be public available for clients to connect and control the smart home system. REST uses HTTP for its application layer protocol, in our project we will use HTTPS

2.3 Projects

In the following subsections we present some examples of other smart home system projects where we discuss their advantages and disadvantages and what are the good ideas that we will reuse in our project.

2.3.1 Smart Home Mobile RFID-based

The paper from Darianian and Michael [2008] presents a RFID reader system architecture for a smart home system composed by several readers in a master-slave architecture. This project introduces a concept to overcome the RFID distance and energy limitations using a low cost ”RF Energy Generator” as transmission power source for mobile RFID readers. The Figure 2.2 illustrates the architecture of this project.

It consists mainly in three types of components:

- Master Reader (MR): a powerful fixed reader that is connected to the smart home server and manage the reader's services. It initiates a reading process in the slave and wakes up any passive tags for power-up or any other service initiation. It is also responsible to collect the information from the slave readers and forward it to the back-end.
- Slave Reader (SR): regular and simple readers acting as relays for capturing ID information tags which are not reachable by the master readers. This type of readers can be easily integrated in the home appliances and they can be used for localization of the tags, when the physical location of the slave reader is known by the system.

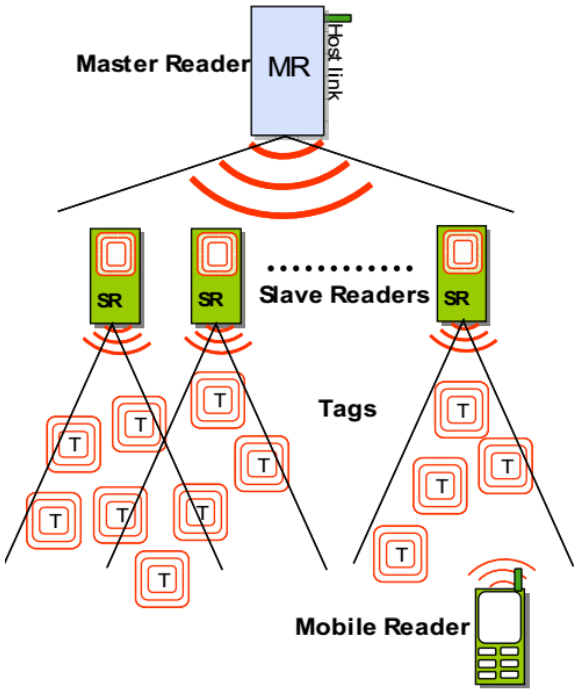

Figure 2.2: RFID Master-Slave Architecture

- Mobile RFID Reader (MRFID): since in the proposed system, the previous components act as ”RF Energy Generator” and energizes tags for wake up an operation, MRFID readers only deal with tags which are powered up and waked, thus this type of readers are only passive readers with low power consuming.

Thus, this project consists mainly in an architecture where master and slave readers act as ”RF Energy Generators”, reducing the MRFID readers power consumption, since they do not need to wake up the passive tags. It is an interesting solution for hierarchical object identification but it still have issues related with the power consumption, since the MRFID readers act as passive tags but requires energy to communicate with the master or slave readers. Also, since the technology used was created with the purpose of object identification, it lacks in flexibility to add complex features to a smart home system.

2.3.2 Smart Home Control System based on Browse/Server Module

Paper Chong et al. [2011] introduces a smart home control system based on B/S (Browser/Server) architecture that provides flexibility, easy expansion and high reliability.

The high level architecture of this system is shown in Figure 2.3. It is a multilevel architecture which promotes a great flexibility and allows the development of the different layers independently. The conceptual explanation of each layer is presented in Subsection 2.1.1.

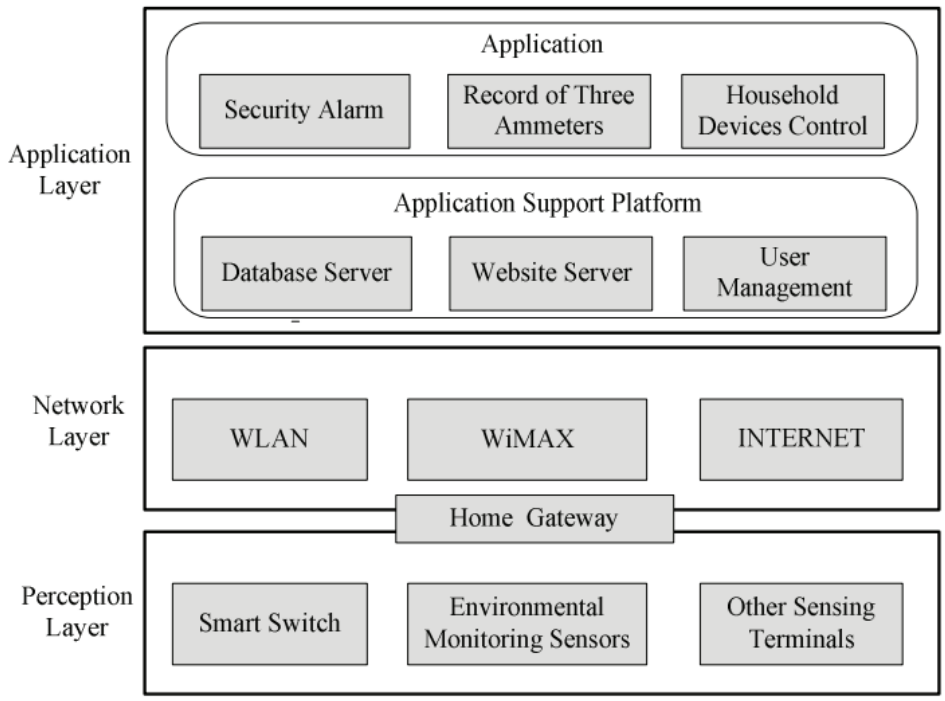

Figure 2.3: Layer architecture model of Smart Home Control System

In this project, authors choose ZigBee for the perception layer because it is a low-complexity, low-power and low-cost technology, as presented in Subsection 2.2.5. Regarding the home gateway, authors have decided to reduce the development cost and design difficulty, thus it 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 and the information is classified as follow:

- Information about household devices contains the device type, address, real-time status and history update records;
- User information contains the user's identity information, private keys and permissions;
- Control Strategies contains the device control instruction set, the device group information and user-defined control strategies.

When users send a command to change a device value, this command is processed according to the workflow presented in Figure 2.4. First, the server checks if the command is valid, according to the command list stored in the database. Then, if it is valid, the server sends this command to the home gateway that are connected with the home devices.

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

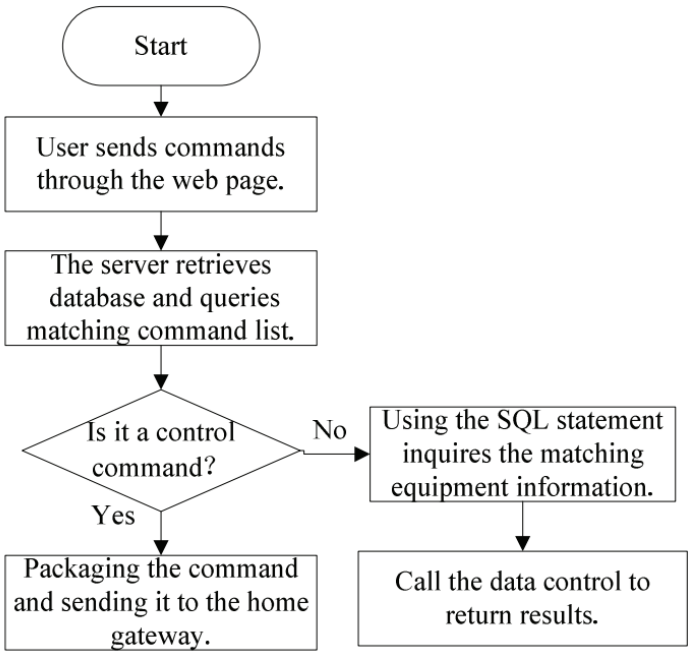

Figure 2.4: Command Transmission Workflow

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 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.

2.3.3 Smart Home: IoT with Webservices and Cloud Computing

The approach in paper Soliman et al. [2013] 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. Another advantage of cloud computing is that users can access home devices anytime and anywhere through the Internet.

The architecture of this system is presented in Figure 2.5 and it consists in the following components:

- Microcontroller-enabled sensors: sensors that measure home conditions;
- Microcontroller-enabled actuators: receive commands and execute them;
- Database: stores data from microcontroller-enabled sensors and cloud services and acts as command queue being sent to actuators as well;
- Server/API layer: 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;
- Web application: enable 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 due to all its advantages as described in Subsection 2.2.5.

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.

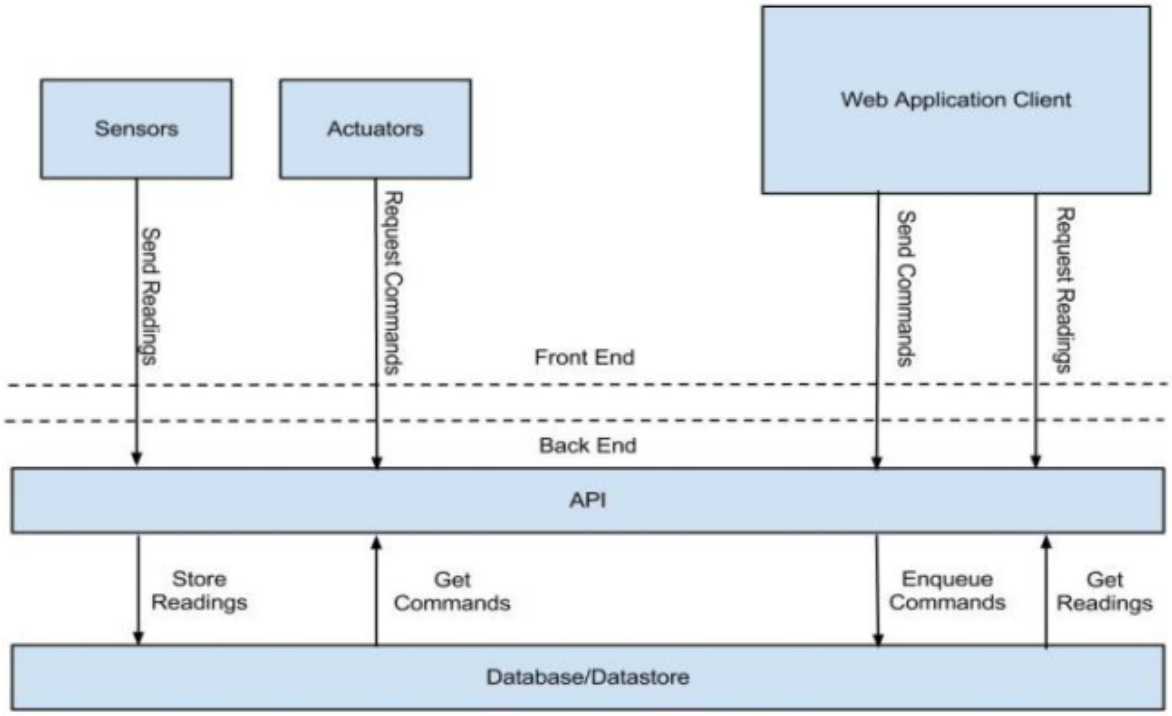

Figure 2.5: System Architecture

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. The authors used Google App Engine platform[4] for developing because it is simple to use, scalable to service requests and had built-in data store. 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 using HTML5 and JQuery mobile 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.

[4] `https://cloud.google.com/appengine/`. Accessed: 2017-04-06

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.

2.3.4 Smart Home System based on IPV6 and ZigBee

As introduced in article Zou et al. [2011] 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 the following fields:

- Name Code: The appliance name, unique;
- State Code: A list of attributes and their values about device's status;
- Command Code: 1-byte command code and input/output arguments;
- Home Code: An unique home code 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.

2.3.5 A ZigBee-Based Home Automation System

Authors in paper Gill et al. [2009] summarized the five main problems with existing smart home systems, which are exactly some challenges that we want to overcome in our project:

1. Complex and Expensive Architecture: existing systems generally incorporate a personal computer which adds more complexity to the system and increase the overall fiscal expense;
2. Intrusive Installation: systems require to add physical wiring in their architectures;

3. Lack of network interoperability: home networks and the home automation systems are developed and implemented in an unplanned and ad hoc manner, leading to a complex maze of heterogeneous networks;

4. Interface inflexibility: user interfaces normally are limited to a single method of control;

5. Security and Safety: existing solutions were developed without security mechanisms implemented.

Thus, the project presented in the paper Gill et al. [2009] proposes a standalone, low cost and flexible ZigBee based home automation system, with an architecture designed to reduce the system's complexity and lower the costs.

The architecture proposed, illustrated in Figure 2.6 has the following main components:

- Home Gateway: implemented to provide interoperability between ZigBee and WiFi networks and it is responsible for remote control and monitoring over the home's devices, offering an external interface;
- Virtual Home: integrated with the home gateway to provide real time security and safety, it pre-processes and checks all communications before they are allowed to continue to their respective destinations;
- Network Coordinator: connected with a device database and a ZigBee-enable module, it is the gateway between the Virtual Home and home appliances.

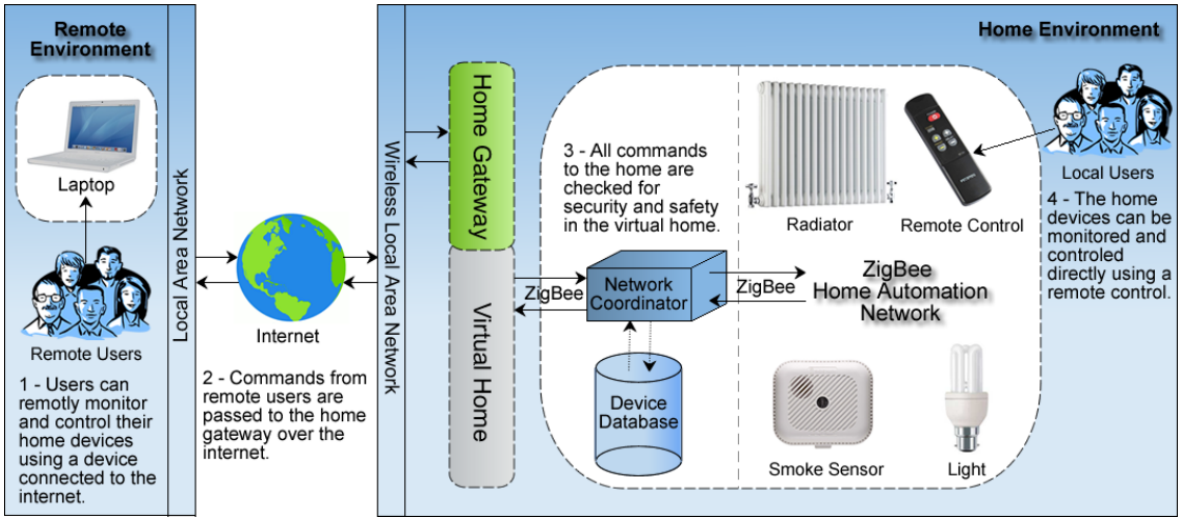

Figure 2.6: A ZigBee-Based System Architecture

This project is very important because first of all, it summarizes the main issues related with the existing smart home systems, that we want to overcome in our project. Secondly, it is implemented through a modular architecture, which is very important to create a robust and flexible system. The Virtual Home component is very useful because it is a component responsible mainly for the security between messages exchanged, providing more security to the smart home users and reducing the requirements of network coordinators.

2.3.6 CASAS: A Smart Home in a Box

The goal of the CASAS project is to design and create a smart home kit that is ready to work ”out of the box”. This means the users just have to purchase these kits and with a minimal effort can install and use all of the smart home components. The main advantage of these kits is that anyone, without technical knowledge, can create and install a smart home system. The system components are extendable, allowing the users to buy and add more components after getting this kit.

The architecture of this project is illustrated in Figure 2.7. The physical layer uses a ZigBee wireless mesh which communicates directly with the hardware components. It is a good choice, because ZigBee is a low cost wireless technology with low energy requirements, as referred in subsection 2.2.5.

The middleware layer works in a publisher/subscriber paradigm, where the manager adds named broadcast channels that allow other components to publish and receive messages. All the components of this system communicate with the Publish/Subscriber manager through a customized ZigBee bridge.

The main goal of this project was to create a product that allows any customer to create and deploy a smart home system with minimal effort, but it lacks in some important issues. It does not offer an user interface and it does not allow to remotely control the home appliances. Authors also do not explain how they provide messages delivery assurance between the devices communication.

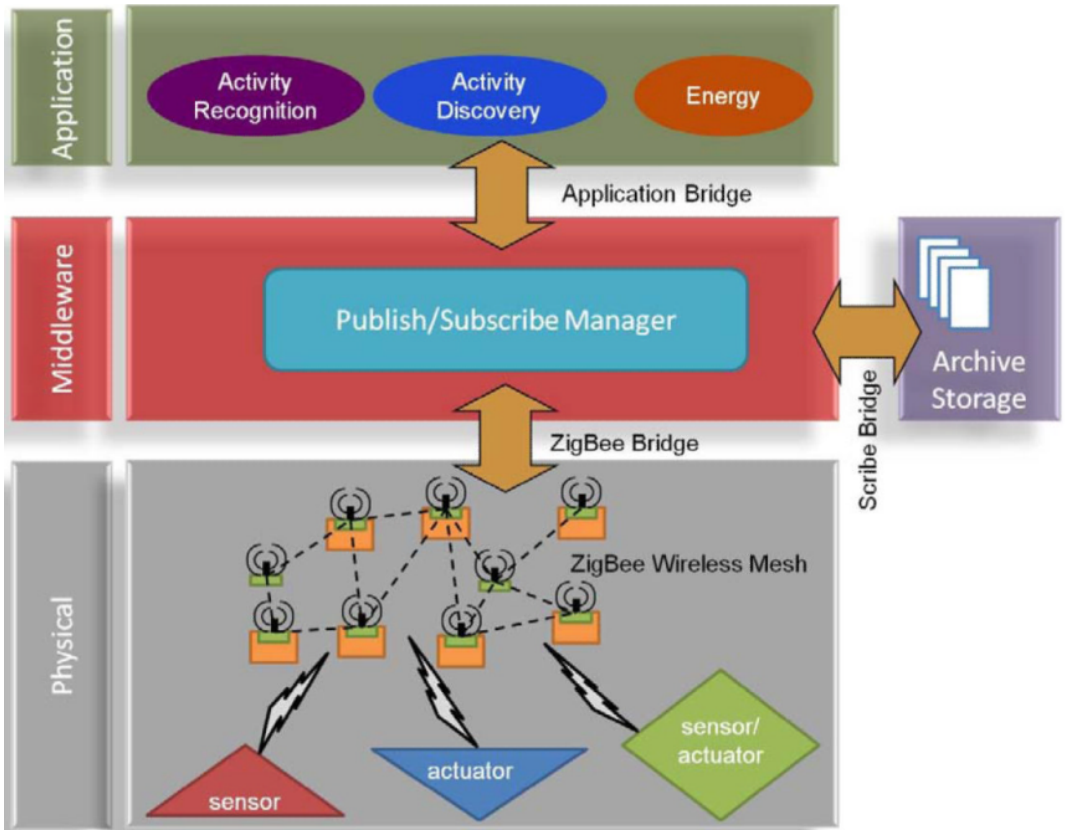

Figure 2.7: CASAS System Architecture

2.3.7 A GSM, Internet and Speech Controlled Wireless Interactive Home Automation System

In the project Yuksekkaya et al. [2006] authors have proposed an home automation system controlled via GSM, internet and speech.

The first two methods allow to control the home appliances remotely, while the last one is only available when the users are inside the house, thus we will not explore this method, since is out of our project scope.

Authors have chosen GSM for communication because it has a wide spread coverage, which makes the system online for almost all the time and because GSM provides an high reliability and security infrastructure that avoids eavesdropper to monitor information exchanged.

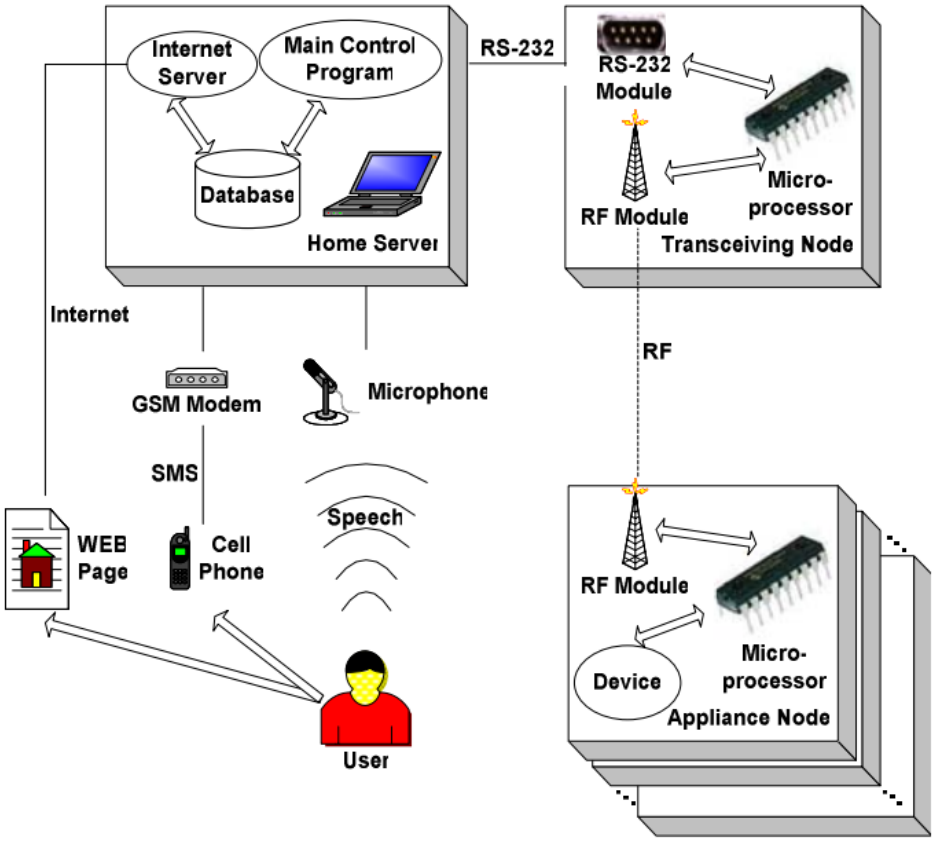

Figure 2.8: GSM,Internet and Speech System Architecture

The architecture of this system is illustrated in Figure 2.8 and has the main components:

- Internet server: to receive commands from the internet;
- GSM modem;
- Main control program;
- Database;

The communication between the home appliances is done via RF Communication protocol, due to its easy and low cost installation and maintenance.

The GSM communication is established according the SMS (Short Message Service) protocol. It is a good communication protocol when users are in remote locations, without internet access. A GSM modem is connected to the home automation server which translates the commands received via SMS to home automation commands. There is also an internet server to receive internet commands (from a webpage). All of these commands are managed by the Main Control Program, which is connected to a database, and is responsible for managing the home automation system and exchanging commands with the RS-232 module, that translates the serial commands to RF commands in order to communicate with the home appliances.

This projects illustrates a different way to control a smart home, using GSM, Internet and Speech, and we will take advantage of some concepts explored here, such as the Home Server component that supports different communication protocols and acts as a Gateway to the internal home automation protocol.

In our system we will not use GSM, but it is interesting that our project allows to create gateways in a flexible and abstract way to support different types of communication protocols that users might need.

2.3.8 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.

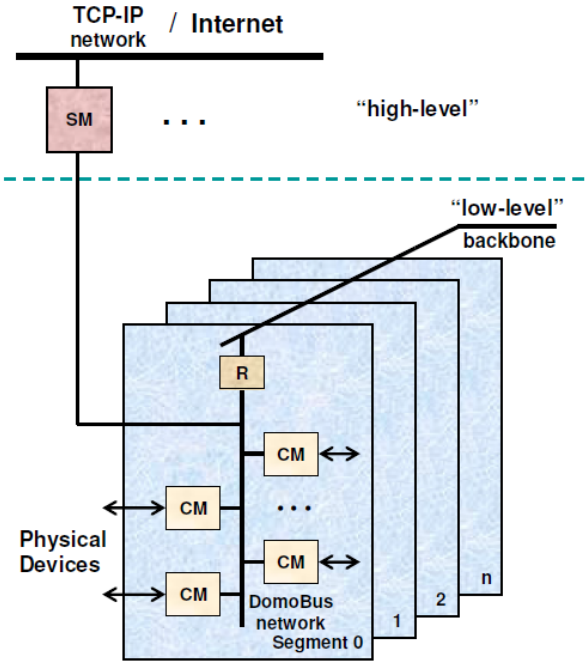

Figure 2.9: DomoBus System Architecture

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 architec-

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

ture, illustrated in Figure 2.9. The *High-Level* components, Supervision Modules(SM), 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;
- **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.

2.4 Products

The following subsections describe some products already available on the market. These products were developed with commercial purposes but we will analyze them and discuss their features.

2.4.1 Samsung ARTIK IoT Platform

Samsung ARTIK IoT platform[6] is an eco-system of tools and services, provided by Samsung, that offers hardware modules and cloud services to create an IoT system with secure, interoperable and intelligent products and services.

This platform has two types of components:

- ARTIK Modules: Pre-certified IoT modules, ready to integrate with ARTIK Cloud, with a wide range of modules, from the lowest levels (the things) to display-based kiosks;
- ARTIK Cloud: ARTIK Cloud is an open data exchange platform for the Internet of Things (IoT) that is designed to connect and communicate with any device regardless of how the data is structured. This is possible, because there is a ”Manifest” file which provides a syntax definition for each device. This file describes the device data and allows to convert the content format in order to store it properly, or send it to targeted devices.

The Samsung ARTIK IoT platform is a good start for anyone who wants to create a smart home ecosystem. The ARTIK hardware modules are ready to work almost out-of-the-box, and the ARTIK Cloud has some interesting features that we will take in advantage in our project development: It has

[6] `https://www.artik.io/`. Accessed: 2017-04-06

great flexibility to connect all the devices even if they communicate through different data formats. Also it allows to create rules, which sends actions to devices when triggered by incoming messages. Rules are one of the most powerful ARTIK Cloud features, because it allows to create an IoT ecosystem with smart device interactions quickly.

The main disadvantages of this platform are related with its costs: a free usage only allows 150 messages per device per day and retention period (includes fast access to stored data,statistics, and aggregations) for 3 months; another problem is related with personalized manifests, that are subject to approval of Samsung Team and Samsung does not provide any template or support to hardware from other vendors, such Arduino, the most popular board.

2.4.2 Samsung SmartThings

Samsung SmartThings[7] are smart products from Samsung that are ready to use and install quickly, out-of-the-box. The SmartThings are composed by SmartThings App (a mobile application to control devices), SmartThings Hub (the hub to manage and connect all the devices inside a smart home) and Smart Devices.

These smart products system has the advantage of quick and easy installation but they are restricted to Samsung's protocols which does not allow to add custom made products or products from other vendors.

2.4.3 Belkin WeMo Home Automation

Belkin WeMo Home Automation[8] is similar to Samsung SmartThings, but is developed by Belkin Company. It provides a full smart home system, with out-of-the-box smart devices and applications (mobile and web-based) to control and manage the smart home system. Its architecture is slightly different from Samsung SmartThings, because the devices are managed directly from the applications, without an intermediary hub, since they are WiFi enabled. The system uses the WiFi router to create a smart home network. To control from outside the home it is necessary to add an hub, such a Raspberry Pi.

Similar to Samsung SmartThings, Belkin WeMo Home Automation system does not provides flexibility to add custom made devices, without adding a WeMo product (WEMO Maker) to connect with all other devices, even if the users create devices that communicate through WiFi.

2.4.4 Amazon Echo (Alexa)

Amazon Echo[9] is a smart speaker developed by Amazon that allows voice interaction and works as a smart assistant. It can play music, making to-do lists, settings alarms, etc. but the more interesting feature regardless our project's subject, is that Amazon Echo can act as a smart hub automation to control and manage smart home devices.

[7] `https://www.smartthings.com`. Accessed: 2017-04-06

[8] `http://www.belkin.com/us/Products/home-automation/c/wemo-home-automation`. Accessed: 2017-04-06

[9] `https://www.amazon.com/Amazon-Echo-Bluetooth-Speaker-with-WiFi-Alexa/dp/B00X4WHP5E`. Accessed: 2017-04-06

It has a wide smart home devices compatibility, from different vendors, and it adds the feature of voice-interaction to control these devices.

Echo requires wireless internet connection to work and the interaction and communication is currently only available in English and German.

Echo allows to add custom device interactions through Smart Home Skill API[10], illustrated in Figure 2.10. As the figure shows, an Alexa smart home system contains the following components:

- Customer: The person who interacts with the Alexa-enabled device and the owner of cloud-enabled devices;
- The Smart Home Skill API: A service that understands the voice commands and converts them to directives (JSON messages) that are sent to smart home skills;
- AWS Lambda: A compute service offered by Amazon Web Services (AWS) that hosts the smart home skill code, which is called a skill adapter;
- Smart home skill: Code and configuration that interpret directives and send messages to a device cloud;
- Device cloud: The cloud environment provided by a device vendor that controls and manages the customer's cloud-enabled devices;

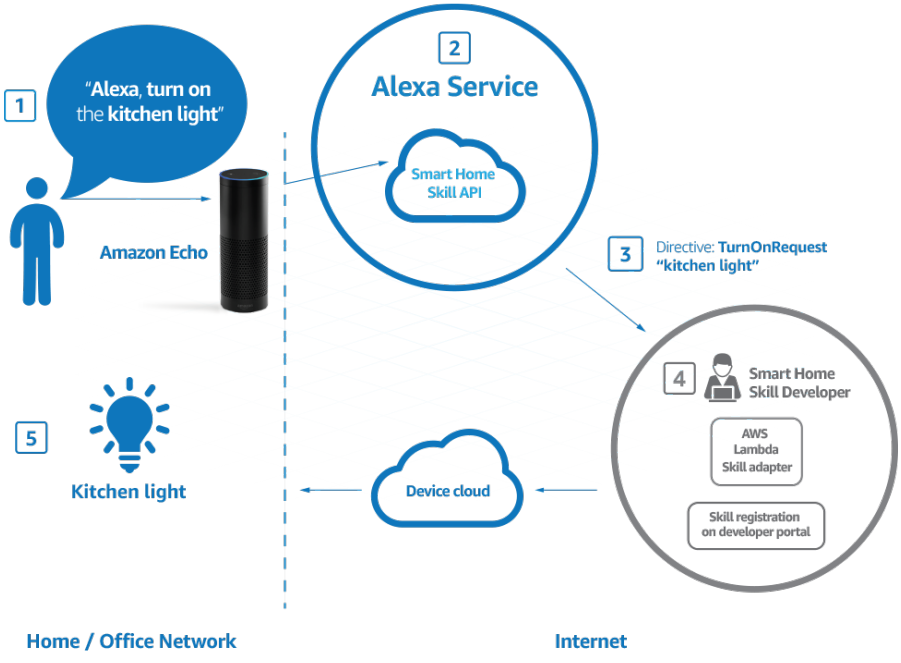

Figure 2.10: Amazon Smart Home Skill API

[10] https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/overviews/understanding-the-smart-home-skill-api. Accessed: 2017-04-06

2.4.5 Apple HomeKit

Apple HomeKit[11] is a kit to quickly create and manage a smart home system, developed by Apple. This kit is compatible with all Apple products but are also compatible with products from other vendors.

Apple HomeKit provides smart devices and an hub to manage all of those devices, but it also provides a framework to communicate and control the connected devices. This products offers interesting features like: it is possible to create actions to control devices; it allows to create scenes where each one has a set of actions for specific devices, e.g. a scene ”Good Night” will turn off all the lights, turn on the security system and adjust the temperature; another feature is that HomeKit integrates all the Apple ”ecosystems”, thus it is possible to control our smart home using Siri, the Apple voice assistant.

The disadvantage of Apple HomeKit is that HomeKit runs a proprietary protocol, so it is hard to create new devices that runs and communicates with Apple HomeKit devices. Also, the Apple HomeKit smart devices are usually more expensive that devices from other vendors.

For our project we will take advantage of the ”Scenes” feature, because it is a useful idea to help users reducing the time they spent configuring devices.

2.4.6 Wink

Wink[12] is a solution that allows to control different smart products from different vendors from a single mobile application. It is compatible with several smart home brands and it allows the users to customize the way that products communicate, creating some actions triggered by other actions, e.g. turn on lights every time user unlock the front door. Similar to Apple HomeKit Wink also allow to create scenes which are ”groups” of devices’ actions that the user defines.

Another important feature in Wink is the ”Schedule” feature, which allows to set timers for each device that are triggered at specific time.

Wink recently also launches a new hub, Wink Hub 2, which allows to consolidate different technologies (protocols) into one, in order to be able to control from the Wink app.

Wink Hub 2 is compatible with Bluetooth Low Energy (LE), Kidde, Lutron Clear Connect, WiFi, Z-Wave, and ZigBee, which are the protocols found in the majority of leading smart home products.

The main advantage of this product is its ability to interconnect and communicate with different smart products from different vendors and with different protocols.

The main disadvantage is its lack of flexibility to add new custom made devices, created by users. It should be more flexible in order to allow users adding their own devices.

2.4.7 Google Brillo / Google Weave

Google Brillo[13] was announced at Google’s I/O 2015 and it is a lightweight and basic backbone for IoT, that allows integration with Android OS devices and support WiFi and BLE. The communications

[11] `http://www.apple.com/ios/home`. Accessed: 2017-04-06
[12] `http://www.wink.com`. Accessed: 2017-04-06
[13] `https://developers.google.com/brillo`. Accessed: 2017-04-06

protocol is called Weave[14], it enables device setup, phone-to-device-to-cloud communication, and user interaction from mobile devices and the web over wireless networks.

Even thought these products are still in a beta phase at the time that we write this document, Google has announced that will provide a new and more robust Google's Internet of Things (IoT) platform, called Android Things.

2.4.8 Muzzley

Muzzley[15] is a Portuguese company that developed a solution very similar to Wink, an application to control and manage different smart devices from different vendors.

The main advantage of Muzzley is its easy integration with new devices. Figure 2.11 illustrates how interaction is achievable. It allows a cloud-to-cloud integration or cloud-to-IoT device integration.

The core of Muzzley consists in a real-time message system through which users interact with devices. It supports the MQTT Pub-Sub messaging protocol, allowing integration with any platform or language.

Muzzley use Hawk authentication with credentials generated to provide a safe communication channel.

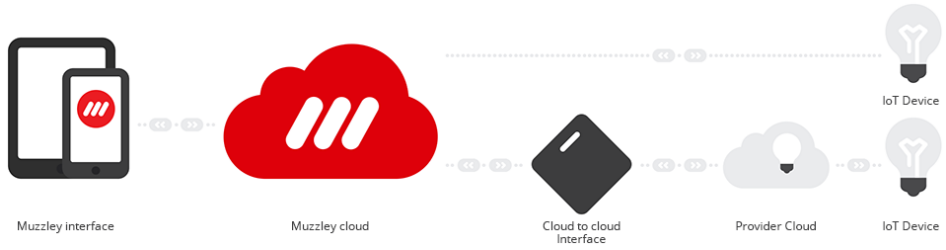

Figure 2.11: Muzzley Integration Archicture

2.4.9 IFTT

"If This Then That" (IFTT)[16] is a web-based service that allows users to create chains of simple conditional statements, called "applets", which are triggered based on changes to other web services. It is possible to create an applet that might consist of when the user enters a certain location (near home, for instance) it turns on A/C and sets the temperature to 25°C, using Nest Thermostat.

IFTT is very popular because it has already a large quantity of services available, which increases the possibilities to create Applets. It is also very popular because it is free and compatible with the most popular smart Hubs, such Amazon Alexa, Wink, Google Home and Apple HomeKit.

2.4.10 Arduino

Arduino[17] is an open-source platform used for building electronics projects that consists of both a physical programmable circuit board (often referred as a microcontroller) and a piece of software, or

[14] `https://developers.google.com/weave`. Accessed: 2017-04-06
[15] `https://www.muzzley.com`. Accessed: 2017-04-06
[16] `https://ifttt.com/`. Accessed: 2017-04-06
[17] `https://www.arduino.cc`. Accessed: 2017-04-06

IDE (Integrated Development Environment), used to write and upload computer code to the physical board.

It is one of the most popular platforms because, unlike most previous programmable circuit boards, the Arduino does not need a separate piece of hardware (called a programmer) in order to load new code onto the board, we can simply use an USB cable. Additionally, the Arduino IDE uses a simplified version of C++, making this language easier to learn and provides a standard form factor that breaks out the functions of the micro-controller into a more accessible and cheaper package.

There are different versions of Arduino boards for different needs, for instance, Uno is the best for beginners, but there is also more versions, such Nano or Mega with different hardware specifications.

To develop our project we need a device that can connect to the internet through WiFi. Thus, we have a wide variety of options available: we can use an Arduino Uno equipped with ESP-8266[18] WiFi module, or we can use a similar and cheaper board with ESP-8266 embedded, such as *WeMos D1 R2*[19].

2.4.11 WeMos D1 R2

WeMos D1 R2 is a mini WiFi board based on the ESP-8266EX and very similar to Arduino Uno, as it is possible to see in Figure 2.12. The technical specifications are presented in Table 2.1. It has 11 digital input/output pins, all have interrupt/pwm/I2C/one-wire supported (except for D0), 1 analog input(3.2V max input), a micro USB connection, a Power jack, 9-24V power input and is Arduino and nodemcu compatible.

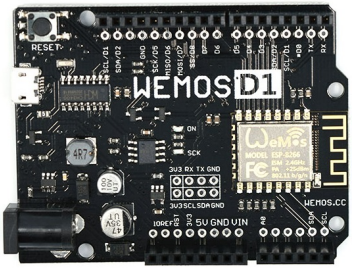

Figure 2.12: WeMos D1 R2

This board is compatible with the Arduino IDE and the programming language is the same as Arduino. Some arduino available libraries may need to be to be rewritten in order to use in the WeMos programas, because it has different pins configuration.

Microcontroller	ESP-8266EX
Operating Voltage	3.3V
Digital I/O Pins	11
Analog Input Pins	1(Max input: 3.2V)
Clock Speed	80MHz/160MHz
Flash	4M bytes

Table 2.1: WeMos D1 R2 Technical Specifications

[18] `http://www.esp8266.com`. Accessed: 2017-04-06

[19] `http://www.wemos.cc/Products/d1_r2.html`. Accessed: 2017-04-06

2.5 Summary

In this chapter we have discussed different technologies and protocols used in smart home systems and also some of the projects and products already available in the market. From all the products available the more robust are the Samsung SmartThings and Apple HomeKit, with several compatible devices that users can buy and easily deploy them in their systems. Although, since these systems were created for commercial purposes, they are restricted to the companies' protocols and do not allow, at least officially, that users create their own devices and connect them in the existing systems nor to create custom applications to control the system.

Nevertheless, we will take advantage of some of the concepts applied on these systems to create our own generic and open-source smart home system, such as the flexibility to add new devices and the home-server architecture.

Regarding the technologies, we want our system to be suitable for houses already finished, and we want to avoid hard installation processes (as changing house electrical system). Thus, we will develop our system using only the WiFi network, as we have seen. Our system will not only work with WiFi-ready devices, but it will allow users to create their own gateways that receive data from other network transport technologies, to connect in our system, for instance a ZigBee - WiFi Gateway or a Serial-WiFi Gateway.

Regarding the communication between devices, as we discussed above, the best protocol is the MQTT due to its flexibility and because it is a light weight protocol that includes some important communication features as the message delivery assurance mechanisms and the retained messages. Also, since the MQTT is a popular protocol, there are already available libraries that we can reuse to implement the clients and the MQTT broker.

3

Domotic Module for the Internet-of-Things

Contents

3.1 Architecture

The system architecture we propose is aligned and proper designed to be deployed in a multi-layer smart home system, as illustrated in Figure 3.1. 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.

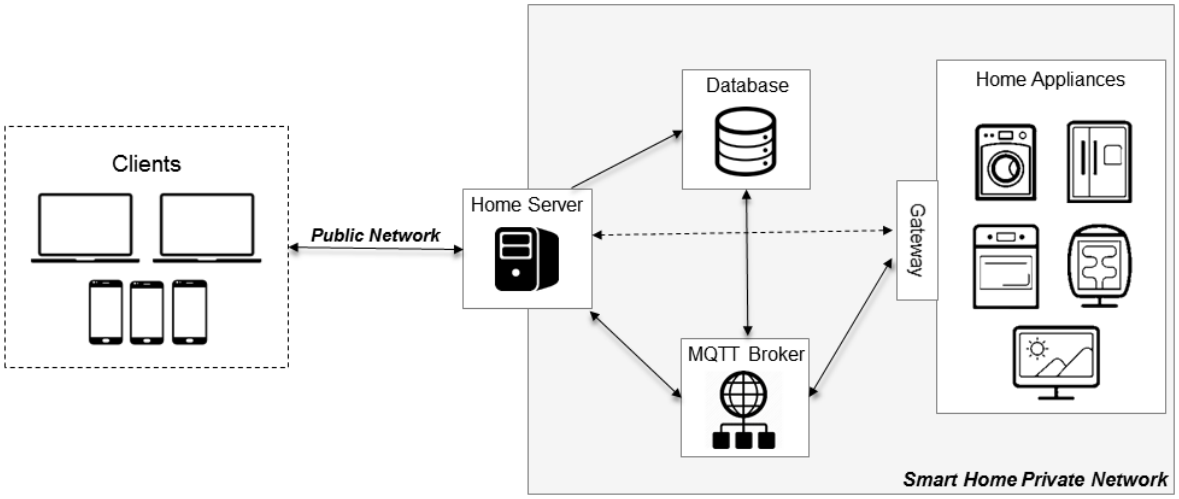

Figure 3.1: Smart Home System Architecture

3.1.1 Home Server

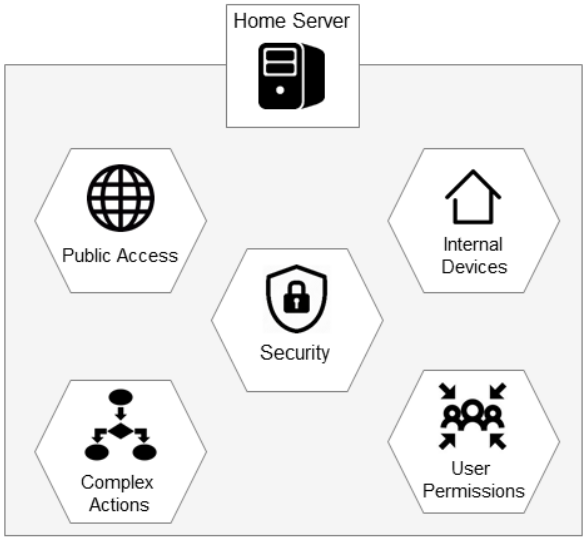

Figure 3.2: Home Server Modules

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 3.2, is a complex system with different modules where each one is responsible for a feature. The modules are:

- Public Access: 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.
- Security: the home server has a security layer requiring an authentication phase from clients to access the smart home system, using digital certificates. This security layer is customizable and it allows to identify users and their user roles.
- Internal Devices Manager: the home server is the core component to manage the devices connected and it is necessary to allow the discovery feature: a mechanism that allows new devices to know what are the available devices, their properties and the topics they should use to communicate with other devices. The Internal Devices Manager also provides administration services, such as, the creation of new rooms, user and user roles, as explained in the next section.
- User Permissions Manager: there is a component in the home server responsible to manage the user permissions. Users have specific roles and permissions to control the devices and this module evaluates, for each request, if the user has permissions to perform the required actions.
- Complex Actions Manager: 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 (e.g. other devices' states) and the corresponding actions that should be triggered when the preconditions are verified, for example, turning on A/C when the home temperature is 26^{o}C.

3.1.2 MQTT Broker

In the solution we propose, the devices communication is made using MQTT protocol due to all its advantages, that were already discussed in section 2.2.9 and are related with its simplicity, the easy implementation and because it is optimized for high-latency and unreliable networks. Thus, it is mandatory to have a MQTT broker (a MQTT server) that allows this communication.

The communication transport we use is a WiFi network and the MQTT broker also provides security mechanisms, such as using certificates to authenticate devices.

3.1.3 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.

3.1.4 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.

3.1.5 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.

3.2 Implementation

3.2.1 Home Server

The home server is a complex system that includes smaller systems working together. The subsystems are described in the next subsections.

3.2.1.A Public Services

The public services are the services that our smart home system make available for external users. They are available in the following URL: *https://{PublicServerIP}/MultiDevicesREST/api/client/{service}*. In Table 3.1 are described the webservices available.

Service	Type	Description
verify	GET	Check if the server is available.
devices	GET	Show connected devices.
device/{deviceID}/components	GET	Show the components of the {deviceID} device.
component/{componentID}/properties	GET	Show the properties of the {componentID} component.
property/{propertyID}	GET	Show the attributes of the {propertyID} property.
properties	PUT	Change a set of properties' values. Sends in JSON format.

Table 3.1: Client Server Methods

The public services are RESTful webservices, that consumes/produces JSON requests/responses. With this approach we guarantee flexibility and freedom to implement different applications, in any platform, without compatibility concerns. It could be a mobile application, a smartwatch, a web-browser application or even another IoT system. The only requirements are that clients should be able to connect to the internet and invoke these services using HTTP protocol with contents in JSON format data.

In our architecture the usage of webservices, makes the system easier to extend with other services that we might want to implement in the future, without changing the services that are already developed. The public services were developed according to a Service-Oriented Architecture (SOA) where they are ”black boxes” for customers and the services are independent from each other.

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. Also, the RESTful webservices are representational, which means that is possible to provide a good load balancer on the server-side that can easily route the requests to the right servers, according to the URLs and the HTTP methods, for instance, the GET requests could go to a group of servers while POSTs go to a different one.

3.2.1.B Private Services

Private services are also RESTful webservices, that handles JSON format data, but for internal usage. Internal means that the services are only available to the internal smart home system network, for devices and for administration purposes.

The private services available are described in Table 3.2. They are accessible using the following URL: *http://{PrivateServerIP}/MultiDevicesREST/api/device/{service}*.

Service	Type	Description
verify	GET	Check if the server is available.
devices	GET	Show connected devices.
devices	POST	Add new devices. Sends in JSON format the devices attributes.
device/{deviceID}	DELETE	Remove the {deviceID} device.
components	POST	Add new components. Sends in JSON format the components attributes.
device/{deviceID}/components	GET	Show the components of the {deviceID} device.
component/{componentID}	DELETE	Remove the {componentID} component.
component/{componentID}/properties	GET	Show the properties of the {componentID} component.
property/{propertyID}	GET	Show the attributes of the {propertyID} property.
properties	POST	Add new properties. Sends in JSON format the properties attributes.
property/{propertyID}	DELETE	Remove the {propertyID} property.
properties	PUT	Change a set of properties' values. Sends in JSON format.
MQTT-broker	GET	Get the MQTT broker URL.
rooms	POST	Create one or more rooms. Sends in JSON format.

Table 3.2: Device Server Methods

According to the best practices of RESTful webservices architecture design, we have used the HTTP methods to differentiate the available types of operations. *GET* operations are used to retrieve data in a read-only way; *DELETE* operations to delete resources, that can include properties, components or even devices; *POST* operations are used to create new resources; and *PUT* are used to update resources. The purpose of these service will be described in the following subsections.

3.2.1.C Internal Devices Management

As discussed in the Subsection 3.1.2, 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. This is a big disadvantage and it is completely in disagreement with our system's goal, that is mainly a very generic and very flexible system, that allows the addition and the removal of devices *ad hoc*.

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. This is an additional system but it is necessary to allow devices to know, after their configuration and with the system up and running, what are the network properties and the devices connected.

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.

A more complete example is illustrated in the Examples of Usage section (Section 3.3), but here is a brief explanation about how this internal device management system works:

We have a chandelier (a device) that we want to connect in our 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.

3.2.1.D Complex Actions Management

An home system to be smart should provide smarter mechanisms than just turning devices ON or OFF, or just reading sensor values. In order to achieve the desirable intelligence, our system provides a Complex Actions Management System that allows the creation of flows of execution according to specific conditions, such as device components' states.

The system consists in a set of preconditions (events), defined by users, 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 26^{o}C, turn on the air conditioner. It also 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.

Although we did not implement this feature in our system, it could be implemented using an approach that consists in a descriptive language that allow users to describe the complex actions. A polling system will check periodically if the properties' values match with any conditions defined and, if they do, the system will execute the corresponding actions.

3.2.1.E User Permissions Management

In a real smart home system, it is useful to have different users with different user roles and permissions according to their responsibilities. For instance, in a family with two parents and one young child, where everyone can control the smart home system, it is not desirable that the child can

access and control dangerous devices, such as an oven or a cooker or even critical house rooms, as a kitchen for instance.

Thus, it is a very important feature to have an User Permissions Management System 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: profile that represents a specific user;
- User Roles: set of user groups that should be assigned to users' profiles. Some examples of user roles includes: Guest, Regular User, Child, Gardener, Housekeeper, among others.

The permissions priority level is from the most generic to least generic, in this scenario, from user roles to users and it allows the following scenario: if there is a Child as user role, that allows the access to all house rooms except the kitchen, and a user had assigned this user role, he will not be able to control kitchen devices. But if his parents want him to control the microwaves, they can assign a specific permission to control this device and he will be able to control only this device, even though his user role does not allow.

The above explanation was about the difference between user profiles and user roles, but in our system we also propose to add another level of permissions: 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 applied the same priority level mechanism, from the most generic to least generic. So, the permissions defined for a room have a lower priority than a device specific permission, as illustrate in Figure 3.3

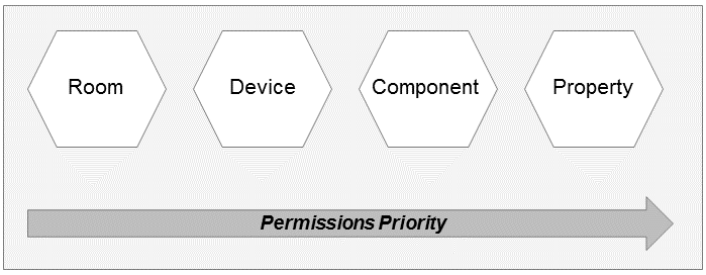

Figure 3.3: Permissions Priority, from lowest (left) to highest (right)

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.

The implementation of our User Permissions Management system is basically a set of queries to the Permission Table, according to the permission priority levels and, if in the end there is no permissions configured, the system will use the default one, i.e. read-only for all properties.

3.2.1.F Authentication / Security

The external users must be authenticated to access the available home server services. The authentication should be done using digital certificates. Whenever clients want to connect to a public service, they should authenticate using their own certificates. The communication should also be done using HTTPS, which makes the conversation with the web server entirely encrypted.

As we have discussed in the subsection User Permissions Management (3.2.1.E) there are two types of user concepts: Individual Users and User Roles. Thus, the digital certificates could be generated for an user role or even for specific users. The home server will use this information in the authentication phase to identify which user profile it will use.

3.2.1.G Minimum Requirements

The minimum requirements to configure the server node are dependent of the application server we use, in this case to use with WildFly 8 we must have:

- Java SE 7 or later;
- WiFi Network Card;
- WiFi Access Point;
- 512Mb of RAM;
- Windows 7 or later or Linux or Mac OS X.

3.2.2 MQTT Broker and MQTT messages

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. We have already addressed this problem in the section 3.2.1.C with the home server working closely with MQTT Broker. In this section we will describe the internal implementation of the MQTT protocol and how the topics are structured.

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, according an home conceptualization structure, where the first level in the hierarchy is the room name. It makes possible 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.

The MQTT Broker in our system has more responsibilities than just allow the MQTT protocol communication between devices. Our MQTT broker is connected directly to the database and when a property' value is changed the MQTT broker will update the database with the corresponding value. There is a callback function on the MQTT broker side that 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 the QoS internal system that is responsible for the delivery assurance of each message. 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 not sent twice or more times. 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).

Using a system with an external module (server) for the MQTT broker also allows the communication between devices without the additional ”work” to the home server. This is useful to reduce the payload in the home server and, if the users want to connect devices that have better computational resources, that can handle some ”complex” actions for themselves, they can have devices managing actions directly from other devices, without the home server acting as an intermediary in this process. The disadvantage of this approach is the additional complexity that it adds to manage all of these complex actions, with some actions defined at the home server level and others directly in the devices.

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 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.

3.2.2.A Minimum Requirements

There is no official information about the minimum requirements to configure a MQTT Broker, because it is dependent of the implementation, but since it runs smoothly on a Raspberry Pi, the minimum requirements are:

- WiFi Network Card;
- WiFi Access Point;
- 512Mb of RAM;
- Windows XP or later or Linux or Mac OS X.

3.2.3 Database

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.

The database relational schema is illustrated in Figure 3.4. 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, as illustrated in the Section 3.3.

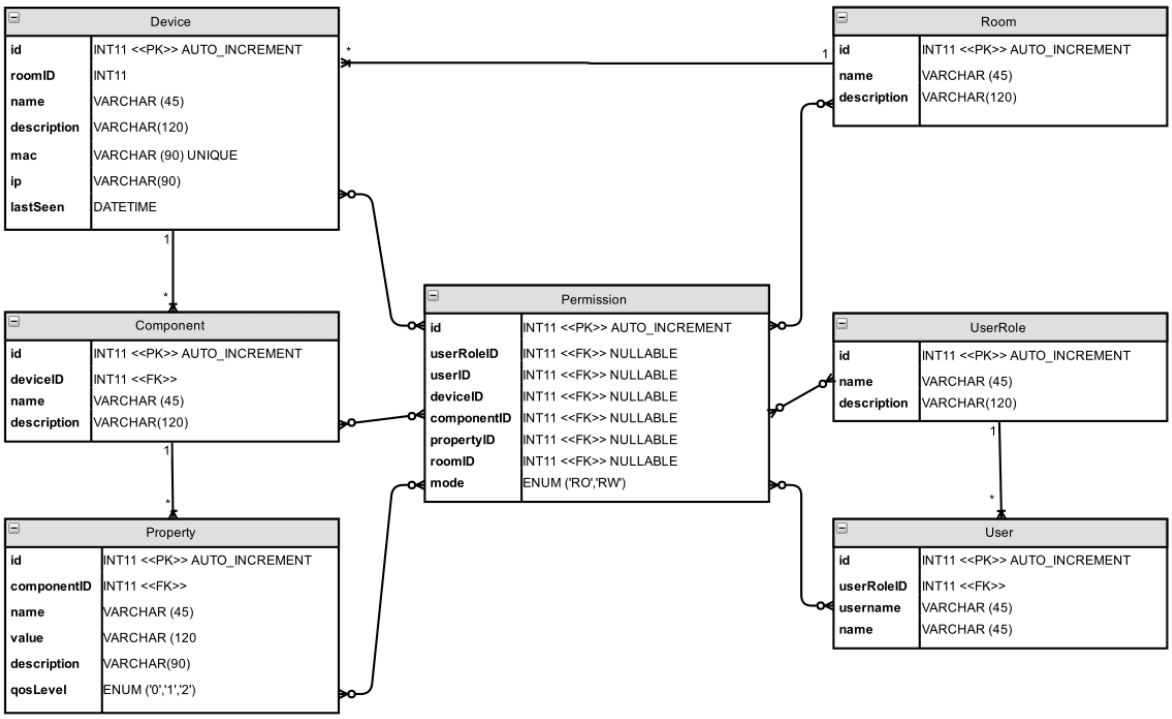

Figure 3.4: Database Schema

3.2.3.A 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.

3.2.3.B 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.

3.2.3.C 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.

3.2.3.D 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.

3.2.3.E 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.

3.2.3.F 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.

3.2.3.G 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 3.2.1.E.

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.

3.2.4 Devices

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 are presented in Subsection 3.2.4.A and it is basically a device that can connect to the internet via WiFi: it 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.

3.2.4.A Minimum Requirements

The minimum requirements are related with the quantity of components and properties. In our implementation, we used WeMos D1 R2 board, thus the devices should have a similar configuration which is:

- WiFi module, e.g. an ESP-8266EX chip;
- TCP/IP protocol stack;
- Clock Speed 80MHz/160MHz;
- Memory Flash: 16MB.

Although not mandatory, the devices can have libraries to manipulate JSON data and an implementation of a MQTT client.

3.2.5 Clients

This system allows to use any client in any platform: it can be a smartphone application, a smartwatch application, a web-page or anything else. The only requirement is that the application need to be able to implement HTTP protocol, because the clients should be able to perform RESTful API calls. Similarly to the devices, if the client has a JSON library the required code will be simpler, but this is not mandatory.

The clients in our system could be from two different types: other IoT systems, and in this scenario they don't provide any user interface (it is a typical scenario of M2M communication) or they are designed to offer user interaction, and in this scenario they should be designed according the best usability practices.

In our system implementation we have not created any user interface (is out-of-scope), we used only the command line to test the client REST API calls, which proves that the client minimal requirements are few, as summarized in Subsection 3.2.5.A.

3.2.5.A Minimum Requirements

The minimum requirements for the client application are:

- Internet connection;
- Able to handle with HTTP requests/responses;
- TCP/IP protocol stack;
- Valid digital certificate;

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.

3.3 Examples of Usage

To illustrate how the system works, we will explain an example of a system composed by one client, one home server and its database, one MQTT broker and one device with two components, a Green and a Red LED. Each of these components have one property, *mode*, whose values can be 0 (OFF) or 1 (ON).

This example illustrates how a device should setup their registration and how the clients can change a property's value. The remaining operations listed in Table 3.1 and Table 3.2 are quite similar.

3.3.1 Administration

In order to have a functional smart home system, it is important that the system's administrator configures the system according to his needs. To achieve that, the administrator should configure, at least, the available rooms. The following messages illustrates an example of a room registration.

Listing 3.1: Room Registration HTTP Request

```
POST /MultiDevicesREST/api/rooms HTTP/1.1
Host: 192.168.1.81:8080
Content-Type: application/json
Cache-Control: no-cache
[
    {"name": "BathroomSuite",
    "description": "The bathroom of 1st floor suite."}
]
```

Listing 3.2: Room Registration HTTP Response

```
HTTP/1.1 200 OK
Date: Thu, 22 April 2017 19:50:46 GMT
Server: Apache/2.2.14 (Win32)
Content-Type: text/html; charset=iso-8859-1
Connection: Closed
[
    {"id": "1",
    "name": "BathroomSuite",
    "description": "The bathroom of 1st floor suite."}
]
```

The system has, per default, one user role: Administrator, with one user assigned: admin. This user role has assigned the maximum permissions level: it allows to change all the devices, components and properties values; to create new user roles and assign them to specific users; to create new rooms and also to create new permissions.

3.3.2 Device Registration

In order to use a device, the system must know about their existence. Thus, a device registration is required, and all of its components and properties. This operation should be done once, during the device initialization phase. The following blocks of code illustrate these operations performed in the right order and their responses.

Listing 3.3: Device Registration HTTP Request

```
POST /MultiDevicesREST/api/devices HTTP/1.1
Host: 192.168.1.81:8080
Content-Type: application/json
Cache-Control: no-cache
[
    {"name": "ArduinoWC",
    "mac": "6A-D1-43-98-AF-6F",
    "ip": "192.168.56.1",
    "description": "To control bathroom LEDs.",
    "roomID": "1"}
]
```

Listing 3.4: Device Registration HTTP Response

```
HTTP/1.1 200 OK
Date: Thu, 22 April 2017 19:52:46 GMT
Server: Apache/2.2.14 (Win32)
Content-Type: text/html; charset=iso-8859-1
Connection: Closed
[
    {"id": "1",
    "name": "ArduinoWC",
    "mac": "6A-D1-43-98-AF-6F",
    "ip": "192.168.56.1",
    "description": "To control bathroom LEDs.",
    "roomID": "1",
    "lastSeen": "April 22, 2017 7:52:55 PM"}
]
```

Listing 3.5: Components Registration HTTP Request

```
POST /MultiDevicesREST/api/components HTTP/1.1
Host: 192.168.1.81:8080
Content-Type: application/json
Cache-Control: no-cache
[
    {"deviceID": "1",
    "name": "GreenLED",
    "description": "LED mirror."},

    {"deviceID": "1",
    "name": "RedLED",
    "description": "LED floor."}
]
```

Listing 3.6: Components Registration HTTP Response

```
HTTP/1.1 200 OK
Date: Thu, 22 April 2017 19:53:34 GMT
Server: Apache/2.2.14 (Win32)
Content-Type: text/html; charset=iso-8859-1
Connection: Closed
[
    {"id": "1",
    "deviceID": "1",
    "name": "GreenLED",
    "description": "LED mirror."},

    {"id": "2",
    "deviceID":1,
    "name": "RedLED",
    "description": "LED floor."}
]
```

Listing 3.7: Properties Registration HTTP Request

```
POST /MultiDevicesREST/api/properties HTTP/1.1
Host: 192.168.1.81:8080
Content-Type: application/json
Cache-Control: no-cache
[
    {"componentID": "1",
    "name": "mode",
    "value": "0",
    "QoSlevel": "0"},

    {"componentID": "2",
    "name": "mode",
    "value": "0",
    "QoSlevel": "0"}
]
```

Listing 3.8: Properties Registration HTTP Response

```
HTTP/1.1 200 OK
Date: Thu, 22 April 2017 19:54:02 GMT
Server: Apache/2.2.14 (Win32)
Content-Type: text/html; charset=iso-8859-1
Connection: Closed
[
    {"id": "1",
    "componentID": "1",
    "name": "mode",
    "value": "0",
    "QoSlevel": "0"},

    {"id": "2",
    "componentID": "2",
    "name": "mode",
    "value": "0",
    "QoSlevel": "0"}
]
```

3.3.3 Client setting a new property's value

Whenever a client wants to change a property value it should send a message similar to the one illustrated in Listing 3.9. The home server, after receiving this message, will check if that property exists and if the client has permissions to change it. If both conditions are true, the server sends the message illustrated in Listing 3.10 to the device, in order to change the desired property. If the operation has success, the device publish a success message, similar to the Listing 3.11, the MQTT Broker update the database and the home server send a message to the client similar to the Listing 3.12.

Listing 3.9: Change Properties HTTP Request

```
PUT /MultiDevicesREST/api/client/properties HTTP/1.1
Host: 192.168.1.81:8080
Content-Type: application/json
Cache-Control: no-cache
[
    {"id": "2", "value": "1"}
]
```

Listing 3.10: MQTT Publish message from Server to Device

```
MQTT-Packet: PUBLISH
packetID: 14
topicName: "BathroomSuite/ArduinoWC/RedLED/mode/change"
qos: 0
retainFlag: false
payload: "1"
dupFlag: false
```

Listing 3.11: MQTT Publish message from Device to Server

```
MQTT-Packet: PUBLISH
packetID: 14
topicName: "BathroomSuite/ArduinoWC/RedLED/mode"
qos: 0
retainFlag: false
payload: "1"
dupFlag: false
```

Listing 3.12: Change Properties HTTP Response from Serve to Client

```
HTTP/1.1 200 OK
Date: Thu, 22 April 2016 20:00:35 GMT
Server: Apache/2.2.14 (Win32)
Content-Type: text/html; charset=iso-8859-1
Connection: Closed
[
    {"id": 2,
    "componentID": 2,
    "name": "mode",
    "value": "1"}
]
```

3.4 Evaluation

During this project all the operations identified in the previous section were developed.

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).

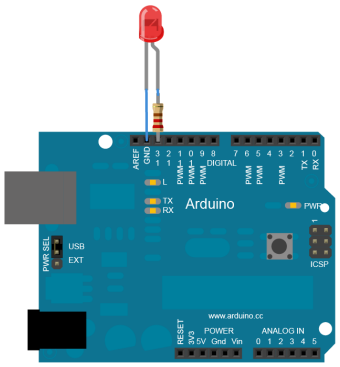

Figure 3.5: Electronic Schema

In order to run the test decribed, we used the circuit illustrated in Figure 3.5, but instead of using the Arduino Uno board, we used the WeMos D1 R2 board, with the LED connected to the D8 pin.

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 3.3: LED typical values

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 3.3.

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 [ohm]:

$$V = R * I \Leftrightarrow R = \frac{V}{I} \tag{3.1}$$

Thus,

$$R = \frac{V_{in} - V_{led}}{I_{led}} \Leftrightarrow R = \frac{3.3 - 1.8}{0.015} = 100\ \Omega \tag{3.2}$$

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 time, for the operation described in Subsection 3.3.3. In this trial all the devices were in the same network and the results are presented in Table 3.4.

#	Client Request Time	Server Response Time	Duration (ms)
1	20:10:12.320	20:10:13.922	1602
2	20:10:14.110	20:10:15.533	1423
3	20:10:17.320	20:10:19.001	1681
4	20:10:20.796	20:10:22.255	1459
5	20:10:23.453	20:10:24.826	1373
6	20:10:26.877	20:10:28.253	1376
7	20:10:29.057	20:10:30.786	1729
8	20:10:32.445	20:10:33.997	1552
9	20:10:35.003	20:10:36.438	1435
10	20:10:37.841	20:10:39.412	1571
11	20:10:40.067	20:10:41.798	1731
12	20:10:43.129	20:10:44.752	1623
13	20:10:46.062	20:10:47.932	1870
14	20:10:48.963	20:10:50.665	1702
15	20:10:52.352	20:10:53.755	1403

Table 3.4: Change Properties Trial

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.

3.5 Summary

The project architecture presented in this chapter was designed as a multilayer architecture with different components,developed to be easily deployed without requiring a complex infrastructure. The modular architecture guarantees a decoupling between the home core system and the devices connected, providing flexibility for the users to change them and the smart home system configurations according to their needs, with the system already in operation.

Even though our evaluation was performed using only one device with one property and one client connected, in a global balance evaluation, the system achieves all the proposed goals. It has a simple protocol that supports a network of several devices and clients connected and it has also important features to be used in real home scenarios, e.g. the users' permissions mechanism.

The WiFi network, as the core communication technology in our system, with the MQTT protocol for communication between devices was a good choice, because it reduces the minimal requirements for the devices and allows to connect devices with low computational power resources to operate in our system. Also, we took advantage of some features of the MQTT protocol to guarantee security and interoperability in our system, such as the QoS levels that we use to specify the messages importance,

where according to the defined levels, the messages could have delivery assurance. The retained message system, also reused from the MQTT protocol, allowed new devices to get information about the other devices' state immediately after they connect in our system.

The decision to provide webservices for clients to connect from the outside, through the Internet, was very important because it detaches the external network from the internal network, increasing the security and limiting the accesses to the devices.

This type of architecture also provides flexibility to create different user interfaces. Our webservices were implemented with standard protocols, RESTful webservices with JSON content data, therefore it allows several implementations: it could be a smartphone application, a website or just a command line, as we used in our evaluation.

Our system provides security related with the external accesses, requiring customers to use secure communications protocols, HTTPS, and authentication with digital certificates. These security mechanisms are simple to be used by customers, however, although this functionality is contemplated in the architecture of our solution, we have not been able to implement and test it properly, so we have not been able to prove that it is adequate to provide the desirable security.

During our evaluation, we noticed that there is a small delay in the messages' delivery, which is related to the message propagation time over the network and also to the amount of components involved. However, this duration is short, and the more complex action (when an user change a property value) takes 1.5 seconds on average to be processed by the system upon receipt, which is reasonable considering the constrained resources of IoT devices.

The architecture of our system includes a mechanism for managing users and their permissions. We proposed a solution, however, this was a mechanism that was not implemented and tested in our project, so we can not demonstrate a concrete evaluation regarding the effectiveness and good functioning of our solution.

The auxiliary database simplified the protocol and provided system scalability, since a relational database system provides support to deal efficiently with a large amount of data and accesses are usually quick. Using this database we also reduce the amount of information that devices need to store, thereby we reduced their storage requirements, and guaranteed the complete decoupling between system components.

4

Conclusions and Future Work

Contents

4.1 Conclusions

The main goal of this project was to create 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 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.

This is a complex system, although it is built in a modular fashion keeping decoupling between modules, allowing users to change the system settings whenever they want, adding or removing devices after the system is implemented and already operational, like a ”plug-and-play” system.

It is a system whose target audience is quite wide, including ordinary users without technical knowledge, that just want to buy the devices and incorporate them into their system, or Do-It-Yourself (DIY) Makers, who want to create their own IoT home appliances.

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.

The core component of our system is the smart home server that is a device with more processing power resources, such as a Raspberry PI or a personal computer. This component contains all the system security and management mechanisms, such as the user permission management, that allows to create users and user roles with permissions assigned at different levels. It is possible to define permissions at different levels of hierarchy (room, devices or properties) which provides great flexibility in managing accesses to the devices.

Regarding the devices allowed, the requirements are minimal since the system management is done on an higher level, requiring that devices only manage their own components and are able to connect to an WiFi network. Our system also offers a feature that allows the creation of gateways to connect with simpler devices, that can not connect to an WiFi network.

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. There is also an auxiliary database that stores all the system information, such as connected devices, their components and status, the structure of the house (rooms), registered users, allowed user roles and their permissions.

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.

Regarding usability, since public services have been deployed with standard technologies, it is possible for developers to create their own interfaces that can be used on different types of devices, such as smartphones, smart-watches, computers or other smart homes systems that they want to incorporate into our system.

With the solution presented in this document we hope we have made an important contribution to the development of smart home systems that are flexible and inexpensive and that allow users to reuse the resources they already have, adding automation and intelligence to increase their comfort and quality of life, but always guaranteeing the security and privacy of their personal data.

4.2 Future Work

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:

1. Add machine learning algorithms that are able to learn the users' preferences and make predictions about future actions or events;
2. Create scenarios, similar to what Apple Homekit system offers;
3. Create gateways to allow devices that are not able to connect to WiFi networks, to connect and use our system;
4. Allow to add more hierarchical levels inside a house, instead of just rooms
5. Allow to change the protocol syntax, similar to the Manifest idea in Samsung Artik IoT plataform.
6. Create an user interface, responsive and adaptive to multiple screen sizes.

Bibliography

Mqtt specification v3.1.1. http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. Accessed: 2017-04-06.

Ohm's law. https://www.physics.uoguelph.ca/tutorials/ohm/Q.ohm.intro.html. Accessed: 2017-04-06.

Kevin Ashton. That 'internet of things' thing. *RFiD Journal*, 22(7):97–114, 2009.

J Bélissent. Getting clever about smart cities: new opportunities require new business modelss. *Forrester Research*, 2010.

Gao Chong, Ling Zhihao, and Yuan 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.

Mohsen Darianian and Martin Peter Michael. Smart home mobile rfid-based internet-of-things systems and services. In *2008 International conference on advanced computer theory and engineering*, pages 116–120. IEEE, 2008.

Roy Fielding, Jim Gettys, Jeffrey Mogul, Henrik Frystyk, Larry Masinter, Paul Leach, and Tim Berners-Lee. Hypertext transfer protocol–http/1.1. Technical report, 1999.

Khusvinder Gill, Shuang-Hua Yang, Fang Yao, and Xin Lu. A zigbee-based home automation system. *IEEE Transactions on Consumer Electronics*, 55(2):422–430, 2009.

Carles Gomez and Josep Paradells. Wireless home automation networks: A survey of architectures and technologies. *IEEE Communications Magazine*, 48(6):92–101, 2010.

Andreas Jacobsson, Martin Boldt, and Bengt Carlsson. A risk analysis of a smart home automation system. *Future Generation Computer Systems*, 56:719–733, 2016.

Gerd Kortuem, Fahim Kawsar, Vasughi Sundramoorthy, and Daniel Fitton. Smart objects as building blocks for the internet of things. *IEEE Internet Computing*, 14(1):44–51, 2010.

Vincent Ricquebourg, David Menga, David Durand, Bruno Marhic, Laurent Delahoche, and Christophe Loge. The smart home concept: our immediate future. In *2006 1st IEEE international conference on e-learning in industrial electronics*, pages 23–28. IEEE, 2006.

Moataz Soliman, Tobi Abiodun, Tarek Hamouda, Jiehan Zhou, and Chung-Horng 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.

Roy Want, Bill N Schilit, and Scott Jenson. Enabling the internet of things. *IEEE Computer*, 48(1): 28–35, 2015.

Xiaojing Ye and Junwei Huang. A framework for cloud-based smart home. In *Computer Science and Network Technology (ICCSNT), 2011 International Conference on*, volume 2, pages 894–897. IEEE, 2011.

Baris Yuksekkaya, A Alper Kayalar, M Bilgehan Tosun, M Kaan Ozcan, and Ali Ziya Alkar. A gsm, internet and speech controlled wireless interactive home automation system. *IEEE Transactions on Consumer Electronics*, 52(3):837–843, 2006.

Miao Yun and Bu 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.

Zhenyu Zou, Ke-Jun Li, Ruzhen Li, and Shaofeng Wu. Smart home system based on ipv6 and zigbee technology. *Procedia Engineering*, 15:1529–1533, 2011.