TÉCNICO LISBOA

Domotic Framework for Internet-of-Things

João Pedro Salgueiro Peralta

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Renato Jorge Caleira Nunes

Examination Committee

Chairperson: Prof. José Luís Brinquete Borbinha
Supervisor: Prof. Renato Jorge Caleira Nunes
Member of the Committee: Prof. Alberto Manuel Ramos da Cunha

June 2019

Acknowledgments

I would like to thank my parents for their friendship, encouragement and caring over all these years, for always being there for me.

I would also like to acknowledge my dissertation supervisor Prof. Renato Nunes for his insight, support and sharing of knowledge that has made this Thesis possible.

Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life. Thank you.

To each and every one of you – Thank you.

Abstract

With the growth of Internet-of-Things (IoT) capable devices, there was also an increase of communication protocols available on the market. These new technologies have its own advantages and disadvantages between them, but this diversity leads to the fragmentation of the domotic area.

The DomoBus system is an academic project with the aim to solve this fragmentation, by presenting an abstract model for home automation devices, independent of the technology used.

The objective of this work is to present a solution to merge the DomoBus system and the modern gadgets that interact with their network. Following that idea, the DomoBus system is introduced in the modern IoT world where this system can grow even further.

As foundation of this work is one of the most used protocols for IoT, MQTT. It was developed a server that allows the connection with multiple devices, MQTT broker, and also a gateway so that the DomoBus system can connect as a whole system, maintaining its features, and expand its operations beyond his ecosystem.

This system supports others gateways of different technologies without them losing their original features.

Keywords

Internet Of Things, Domotic, DomoBus, Home Automation, Gateway, MQTT

Resumo

Com o aumento da quantidade de dispositivos com capacidade de estar ligados à Internet, Internet-of-Things, houve também um aumento na quantidade de protocolos de comunicação à disposição no mercado. Estas novas tecnologias apresentam vantagens e desvantagens entre si, mas com esta separação no mercado surge fragmentação na domótica.

O sistema DomoBus é um projeto académico com o objectivo de resolver esta fragmentação, apresentando um modelo abstrato para dispositivos da domótica, independentes da tecnologia usada.

Assim, o objetivo deste trabalho é apresentar uma solução que combine o sistema DomoBus com os dispositivos modernos que interagem na sua própria rede. Desta forma, introduzimos o sistema DomoBus no mundo moderno do IoT para que este ecossistema se estenda ainda mais.

Como base deste trabalho está um dos protocolos mais usados para IoT, o MQTT. Foi desenvolvido um servidor que permite a ligação dos vários dispositivos, o broker MQTT, e um gateway para que o sistema DomoBus se ligue como um sistema inteiro, o que mantém a sua função, e que expande as suas acções para fora do seu ecossistema.

Este sistema permite que outros gateway de outras tecnologias se liguem ao broker sem que percam a suas funções originais.

Palavras Chave

Internet Of Things, Domótica, DomoBus, Casas Inteligentes, Gateway, MQTT

Contents

List of Figures

Listings

Acronyms

API Application Program Interface

AWS Amazon Web Services

CM Control Module

CoAP Constrained Application Protocol

HTTP Hypertext Transfer Protocol

IoT Internet-of-Things

JSON JavaScript Object Notation

LAN Local Area Network

M2M Machine-to-Machine

MQTT MQ Telemetry Transport

OCF Open Connectivity Foundation

QoS Quality of Service

SM Supervision Module

XML Extensible Markup Language

1

Introduction

Contents

Home automation was, some years ago, an idea of science fiction, but now, with the current technology available, is a reality. Internet-connected household devices are increasingly being marketed to the general public.

Nowadays there is a number of devices present in our life who can connect to the Internet and there is already a market involved in home automation. Connecting these two worlds, the Internet and the automation, is the idea behind Internet-of-Things (IoT).

The IoT transforms these devices from being traditional to smart by making use of new technologies such as Internet protocols and applications, communication protocols, sensor networks, etc.

The evolution of home automation was difficult. This is due to fact that during the initial period it was dominated by proprietary products and some standardization movements were initiated. The existence of various standards divided and confused the market, and new products and technologies kept appearing.

In 2016, the Open Connectivity Foundation (OCF), a group of major industry leaders who decide to invest in the Internet-of-Things, was founded. The idea was that with OCF specifications, protocols and open source projects, ultimately, a wide-range of devices and sensors from a variety of manufacturers, could securely and seamlessly interact with one another. [1]

The number of connected devices that are in use worldwide now exceeds 17 billion, with the number of Internet-of-Things (IoT) devices at 7 billion (that number does not include smartphones, tablets, laptops or fixed line phones). The number of IoT devices that are active is expected to grow to 10 billion by 2020 and 22 billion by 2025. This number of IoT devices includes all active connections and does not take into consideration devices that were bought in the past but are not used anymore. [2]

The IoT, enabled by the already ubiquitous Internet technology, is the next major step in delivering Internet's promise of making the world a connected place.

The devices connected to the Internet are growing, as we said before, but even though they are connected and are considered "smart" devices, they locked in their ecosystem. This ecosystem is something every company creates when launches a set of IoT devices. Several ecosystems of IoT devices are available in the present day, but a great majority can't communicate with other devices, except inside the ecosystem developed by each manufacturer.

1.1 Objectives

This project aims to develop an architecture to overcome the gap between the current state of the IoT panorama and the DomoBus system.

The DomoBus system, to be detailed later in the document, embraces the idea of achieving a home automation system who is independent of any technology and capable of inter-operation among several

technologies. This is the idea behind this project, but as we stated the gap between IoT panorama and the DomoBus system, we set our goal to reduce this gap by giving the DomoBus system the possibility to interact with the IoT panorama.

To attain this task, it will be developed a broker that uses a technology present in the IoT panorama, the messaging protocol MQ Telemetry Transport (MQTT). Since the MQTT is only the messaging protocol, there is a need to maintain the data model approach of the DomoBus system. To establish a connection from the DomoBus to the broker, a gateway is going to be developed. Having the right data model approach, a technology present in the IoT and a gateway to establish the connection between them, we have a goal to map the data model from one system to the architecture of another system.

With this goals, the DomoBus system shall be a part of the IoT and with that capable of interaction to other technologies compatible with MQTT.

1.2 Organization of the Document

This thesis is organized as follows: Chapter 1 introduces the problem and what it is the goals to achieve.

In chapter 2 are presented some related technologies and products available to give an overall view of the state of art relative to this project.

In chapter 3 is described the architecture of the system and explanation for the decisions that are presented.

In Chapter 4, we describe the implementation of the system and the evaluation with some example cases.

In Chapter 5, we present conclusions and some ideas of the final work and some proposals for future work that would improve this project.

2

Related Work

Contents

2.1 Communication Protocols

2.1.1 KNX

KNX is an open standard for commercial and domestic building automation. KNX devices are connected by a bus that is routed in parallel to the electrical power supply to all devices and systems on the network. This way sensors, actuators, controllers and system devices are all linked.

Application software, together with system topology and commissioning software, is loaded onto the devices via a system interface component. With the help of a KNX IP router all system is accessible through Ethernet via that device. Therefore, we can communicate with it via Local Area Network (LAN).

The KNX system working with a KNX IP router uses KNXnet/IP standard. [3]

2.1.2 ZigBee

Zigbee [4] is a wireless technology developed as an open global standard to address the unique needs of low-cost, low-power wireless Machine-to-Machine (M2M) networks.

The Zigbee protocol is designed to communicate data through hostile RF environments that are common in commercial and industrial applications.

A key component of the Zigbee protocol is the ability to support mesh networking.

Mesh networks are decentralized in nature; each node is capable of self-discovery on the network. Also, as nodes leave the network, the mesh topology allows the nodes to reconfigure routing paths based on the new network structure. The characteristics of mesh topology and ad-hoc routing provide greater stability in changing conditions or failure at single nodes.

Amazon introduced a new house hub with Zigbee support in 2018. That support allowed the hub to directly connect to a new world of smart home gadgets, like light bulbs, power outlets, and tiny sensors that otherwise couldn't have reached. [5]

2.1.3 Wi-Fi

Wi-Fi, or 802.11, is a wireless protocol that was built with the intent of replacing Ethernet using wireless communication over unlicensed bands. Its goal was to provide off-the-shelf, easy to implement, easy to use short-range wireless connectivity with cross-vendor interoperability.

It allows to access the internet while on the move usually establishes a Local Area Network(LAN). Wi-Fi enabled computers send and receive data indoors and out.

2.2 Messaging Protocol

A message protocol defines the rules, formats and functions for transferring messages between the components of a messaging system. The next sections shows a few protocols used in the application layer (OSI Model) for IoT.

2.2.1 MQTT

IBM invented MQTT for satellite communications with oil-field equipment. The motivation for designing MQTT was to create a lightweight and bandwidth-efficient protocol that was data agnostic with support for multiple levels of Quality of Service (QoS). Interestingly, even today, those are the same reasons for which MQTT is chosen for implementing IoT solutions.

The MQTT standard has since been adopted by the OASIS open-standards society and released as version 3.1.1, it's also supported within the Eclipse community. [6] [7]

MQTT uses the publish/subscribe pattern instead of the traditional client-server architecture. In the client-server model, a client communicates directly with an endpoint. The publish/subscribe model separates the client that sends a message, the publisher, from the client or clients that receive the messages, the subscribers. The publishers and subscribers never contact each other directly, they are not even aware that the other exists. So, the connection between them is handled by another element, the broker. His job is to filter all incoming messages and distribute them to subscribers.

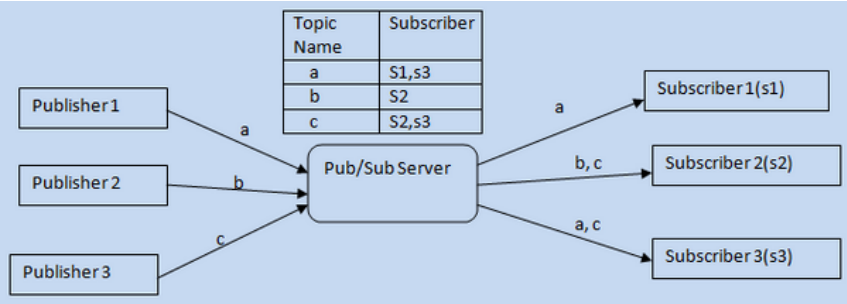

Figure 2.1: Publish/Subscribe Pattern

The publish/subscribe pattern, represented in Figure 2.1, scales better than the client-server model because the broker can parallelize his operations and messages can be processed in an event-driven way. The broker plays a pivotal role in this whole process.

In a generic publish/subscribe 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, denominated topic in MQTT, 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 for different subscription level.

In MQTT all devices connected to the broker are considered a client, as shown in Figure 2.2

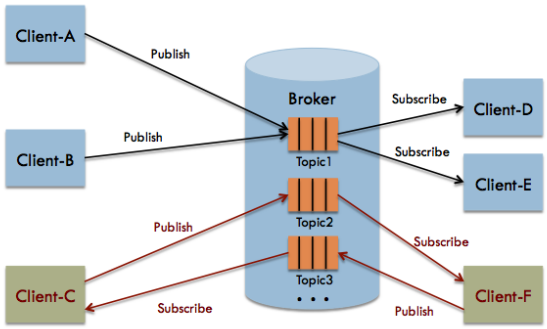

Figure 2.2: MQTT Example

An important feature of any protocol to be used in IoT devices is it implements the delivery assurance mechanism. The MQTT protocol has QoS levels which allow to specify the level we want in each message. In MQTT there are 3 QoS levels:

- QoS 0 - at most once: This service level guarantees a best-effort delivery. There is no guarantee of delivery. The recipient does not acknowledge receipt of the message and the message is not stored and re-transmitted by the sender.
- QoS 1 - at least once: guarantees that a message is delivered at least one time to the receiver. The sender stores the message until it gets a PUBACK packet from the receiver that acknowledges receipt of the message. It is possible for a message to be sent or delivered multiple times.
- QoS 2 - exactly once: guarantees that each message is received only once by the intended recipients. QoS 2 is the safest and slowest quality of service level. The guarantee is provided by at least two request/response flows between the sender and the receiver.

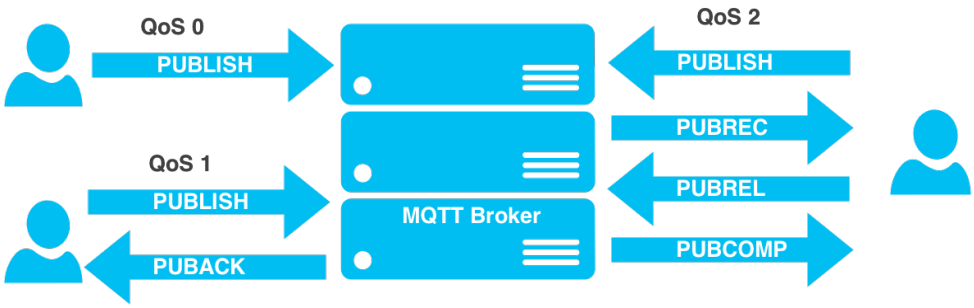

Figure 2.3: Connections for each QoS level

The MQTT protocol includes a mechanism where the broker stores the last retained message for a specific topic, this feature is called Retained Messages. This feature allows a client that subscribes to

a topic that has a retained message, to receive the last message immediately after subscribing. Thus, clients do not have to wait until a new message is published to know the last known status of those devices.

When a client subscribes to a topic, it can subscribe to the exact topic of a published message or it can use wildcards to subscribe to multiple topics simultaneously. wildcard can only be used to subscribe to topics, not to publish a message. There are two different kinds of wildcards: single-level and multi-level.

The single-level wildcard is represented by a plus symbol (+). Any topic matches a topic with single-level wildcard, for example, subscribing the topic *home/groundfloor/+/lights* is equivalent to subscribe to *home/groundfloor/kitchen/lights*, *home/groundfloor/bathroom/lights* or any other name in place of the wildcard.

The multi-level wildcard covers many topic levels. The hash symbol (#) represents the multi-level wildcard in the topic. For the broker to determine which topics match, the multi-level wildcard must be placed as the last character in the topic after the last topic level wanted. When a client subscribes to a topic with a multi-level wildcard, it receives all messages of a topic that begins with the pattern before the wildcard character, no matter how long or deep the topic is. For example, *home/groundfloor/* is equivalent to subscribe *home/groundfloor/kitchen/lights*, *home/groundfloor/bathroom/lights* or any other topic after *home/groundfloor/*.

This project will heavily rely on MQTT because this protocol is gaining such traction and has great compatibility with other technologies shown in this document.

2.2.2 HTTP

The Hypertext Transfer Protocol (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).

Probably is the best known protocol at this level as it is the protocol that powers the Web. HTTP is also important to the IoT as it is used for REST Application Program Interfaces (APIs) which are becoming the main mechanism for Web Applications and services to communicate.

2.2.3 CoAP

Constrained Application Protocol (CoAP) is a specialized Internet Application Protocol for constrained devices.

Like HTTP, Constrained Application Protocol (CoAP) is a document transfer protocol. Unlike HTTP, CoAP is designed for the needs of constrained devices. CoAP follows a client-server model. Clients

make requests to servers, servers send back responses. Clients may GET, PUT, POST and DELETE resources. CoAP is designed to inter-operate with HTTP and the RESTful web at large through simple proxies.

2.3 Projects for Home Automation

This section lists a few projects that have a reach a certain maturity and have a community working and helping maintain and grow.

2.3.1 NODE-RED

Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things.

Messages representing events flow between nodes, triggering processing that results in output. The flow-based programming model maps well to typical IoT applications.

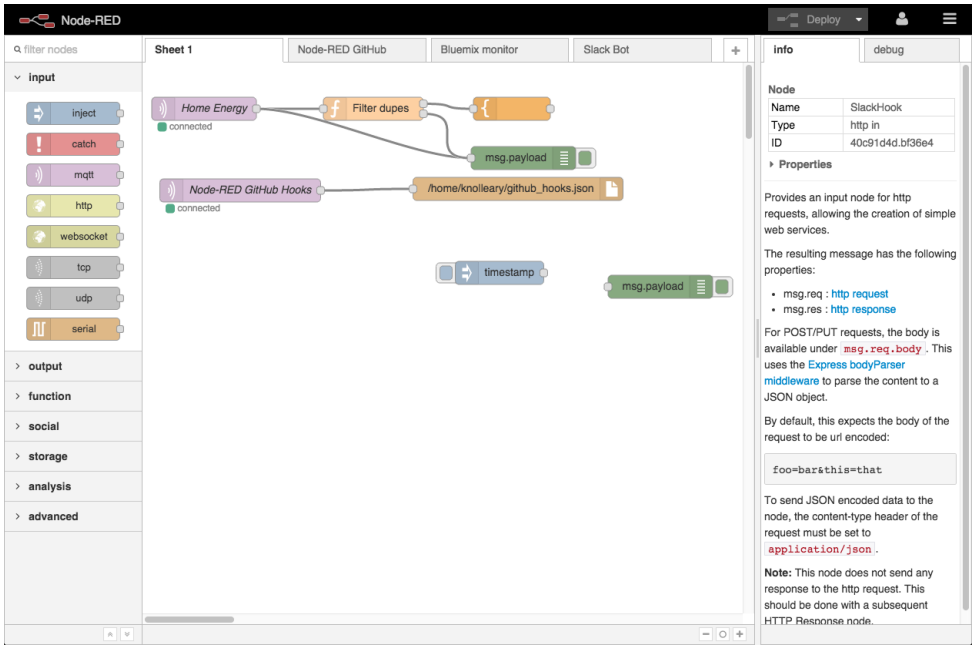

Figure 2.4: Example of NODE-RED events flow

2.3.2 Home Assistant

Home Assistant is an open source home automation platform designed to be easily deployed on almost any machine that can run Python3. It integrates with a large number of open source as well as commercial offerings.

Home Assistant has a huge list of components capable of running on the platform and from a variety of categories, from Automation to Hub's. In figure, we can see a sample of the compatible services and products available to connect to this system, with very known services being on that list.

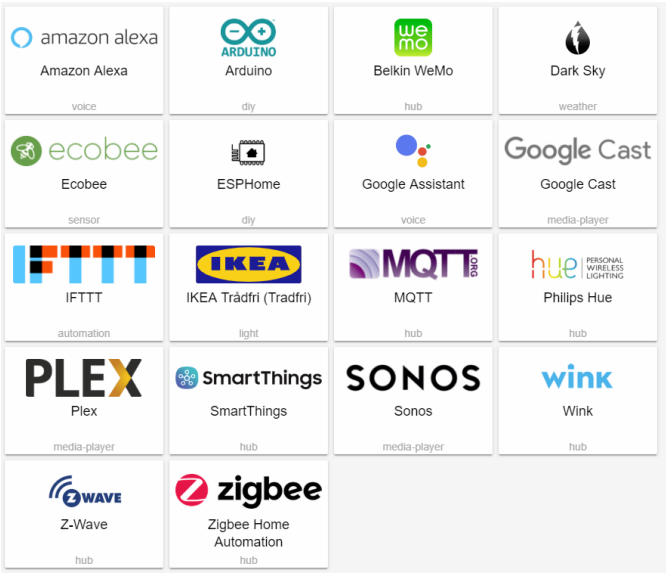

Figure 2.5: Sample of services compatible with Home Assistant

Important to notice that it supports MQTT. It can connect to a private MQTT broker or even run its own embedded MQTT broker.

2.4 Products

This section shows some products available on the market. These products were developed by companies with intent to have their own smart home ecosystem.

2.4.1 Amazon

Amazon is a big international company, they focus on e-commerce, cloud computing and artificial intelligence. As such, we are presenting some products and services that might fit in use cases of this project.

One product is Amazon Echo, a smart speaker that allows voice interaction and works as a smart assistant. It has a wide smart home devices compatibility, from different vendors, and it adds the feature of voice-interaction to control these devices. Echo also allows adding custom device interactions through Smart Home Skill API. [8]

As the Figure 2.6 shows, an Alexa smart home system contains the following components:

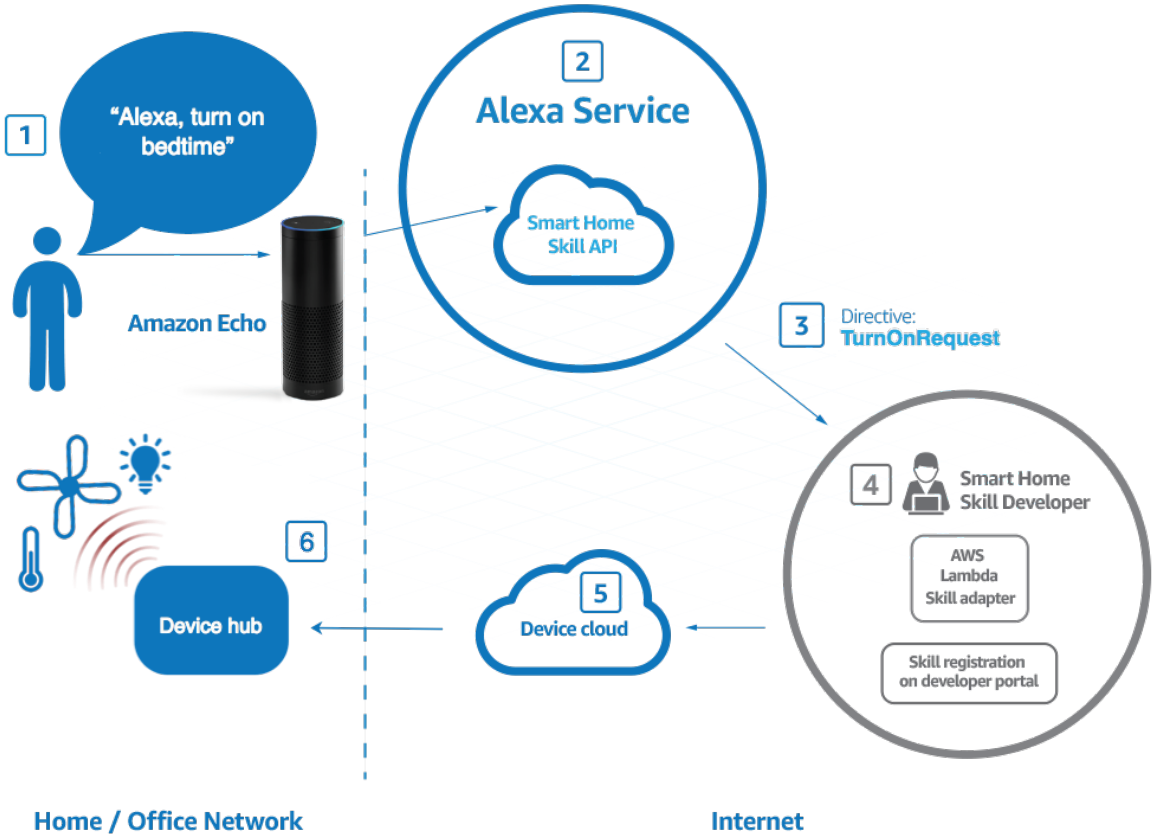

Figure 2.6: Amazon Smart Home Skill API

- 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 (JavaScript Object Notation (JSON) messages) that are sent to smart home skills;
- Amazon Web Services (AWS) Lambda: A compute service offered by 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;

Amazon provides a platform called AWS IoT offering a diversity of services, Figure 2.7, each in charge with a function of IoT, but all working to support great scale IoT deployments.

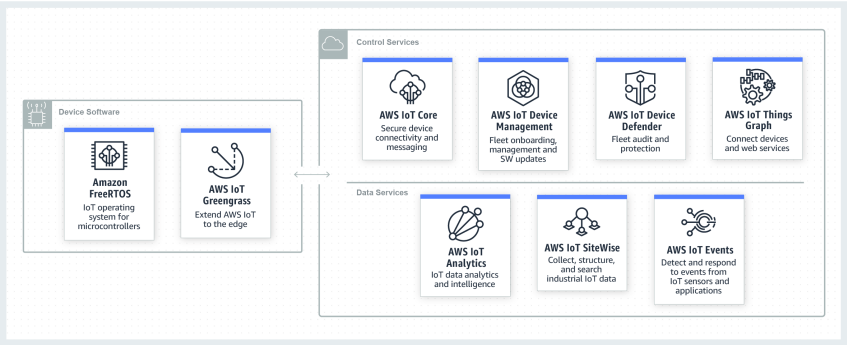

Figure 2.7: Amazon IoT Services

Amazon has joined the Zigbee Alliance, supporting the idea to standardize IoT using Zigbee technology. [9]

2.4.2 Google

Google, another big international company that specializes in Internet-related services and products. They also offer the same type of products and service that we referenced on Amazon topic. The product capable of voice-interaction to control devices is Google Home hub. They also allow us to add custom device interactions, through Actions on Google.

Unfortunately, Actions on Google doesn't support MQTT.

Google also provides a platform for IoT deployments, Google Cloud IoT. This platform supports both MQTT and HTTP.

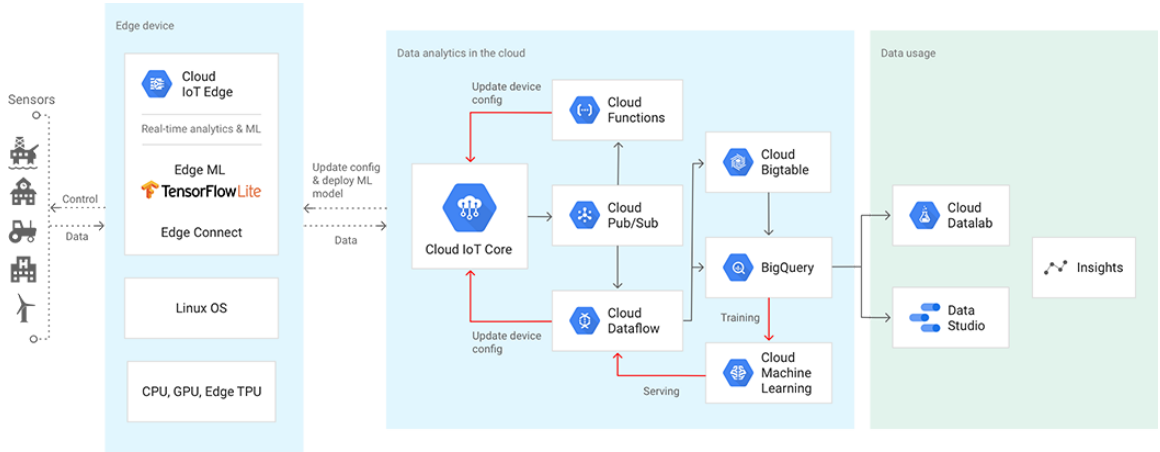

Figure 2.8: Google Cloud IoT

In the fig. 2.8 is shown an overview of the platform architecture. A lot of services present attract IoT deployments in large scale, but in small scale is not ideal.

2.4.3 Samsung

Samsung also wanted to enter the IoT world, so in 2014, they bought a company called SmartThings, company created through a Kickstarter campaign in 2012. [10]

A nice feature in the hubs of SmartThings is having both Z-Wave and Zigbee antennas incorporated, to accomplish more compatibility with the market. They also have Wi-Fi technology.

3

Architecture

Contents

3.1 DomoBus System

The DomoBus System [11] 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.

This system offers an easy and extendable way to create and define scalable and flexible automation projects, with a distributed architecture.

A DomoBus architecture is composed of two levels: one said to be low-level, called Control Level; and another one that functions at a higher level, in the perspective of the network, referred as Supervision Level.

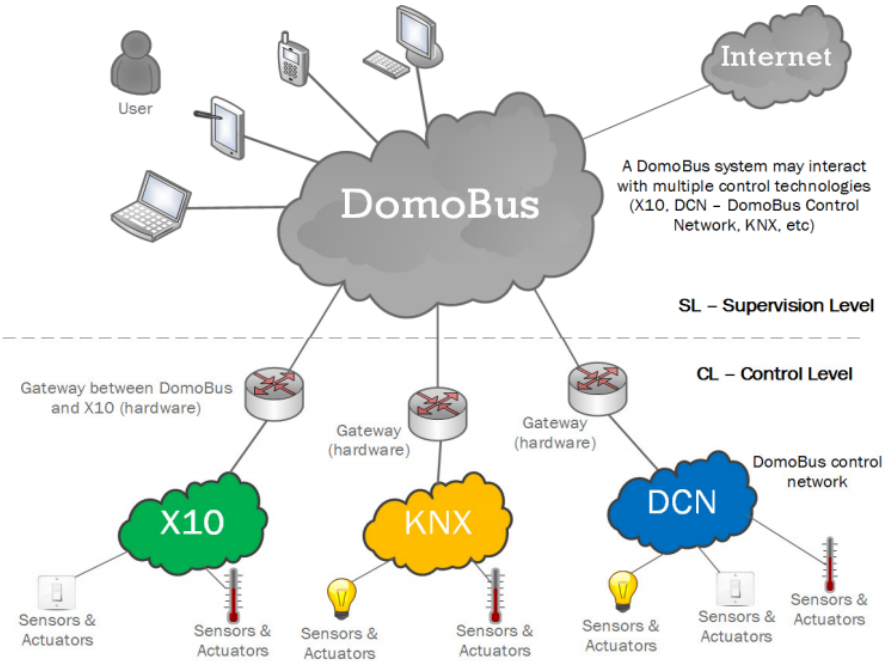

Figure 3.1: DomoBus Architecture (Control Level)

The Control Modules (CMs) are directly connected to physical devices and sensors, they are responsible for managing them and allowing external interactions through exchanged commands. Each CM is able to run different applications and perform different functions. In Figure 3.1 is illustrated the architecture previously detailed.

The Supervision Modules (SMs) are responsible for system management and supervision. They receive information from the CMs, process it accordingly to programmed rules and required behavior, and issue the appropriate commands to the CMs. A system may have as many SMs as needed. In a small system we can have just one SM, while in a big or complex system we may have, for example, one SM for each DomoBus network segment.

In Figure 3.2 it illustrates the architecture of the Supervision Level.

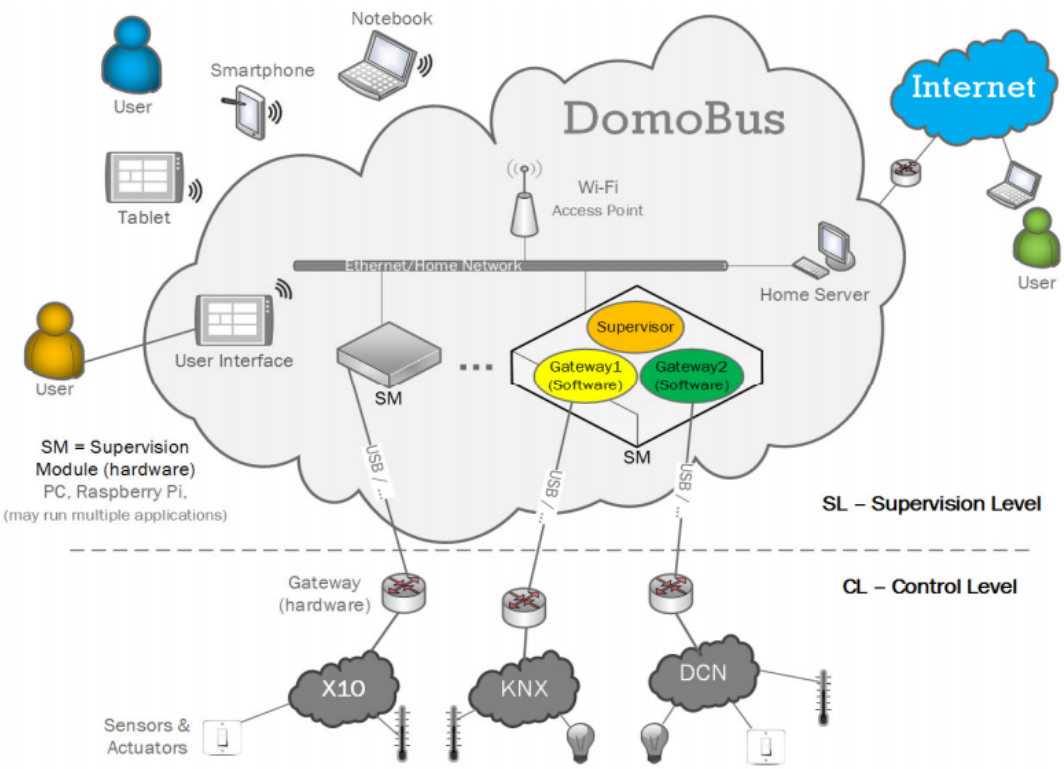

Figure 3.2: DomoBus Architecture (Supervision Level)

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 when a property value changes, sending the new value information.

This project focus primarily on expanding the DomoBus capabilities in the IoT, but with the this new introduced technology it allows for more gateways of other ecosystems to connect to our broker. The fig. 3.3 shows an example of the network with multiple gateways including MQTT communication.

In the fig. 3.3, a SM hosts the gateways related to MQTT. It is also shown the MQTT Broker running in the home server, but, as another option, the broker can also be hosted and running on the Internet.

3.2 MQTT Broker

In the solution, we propose adding a new communication technology to the DomoBus System using MQTT protocol considering the advantages it provides, as explained in the chapter 2 in more details on

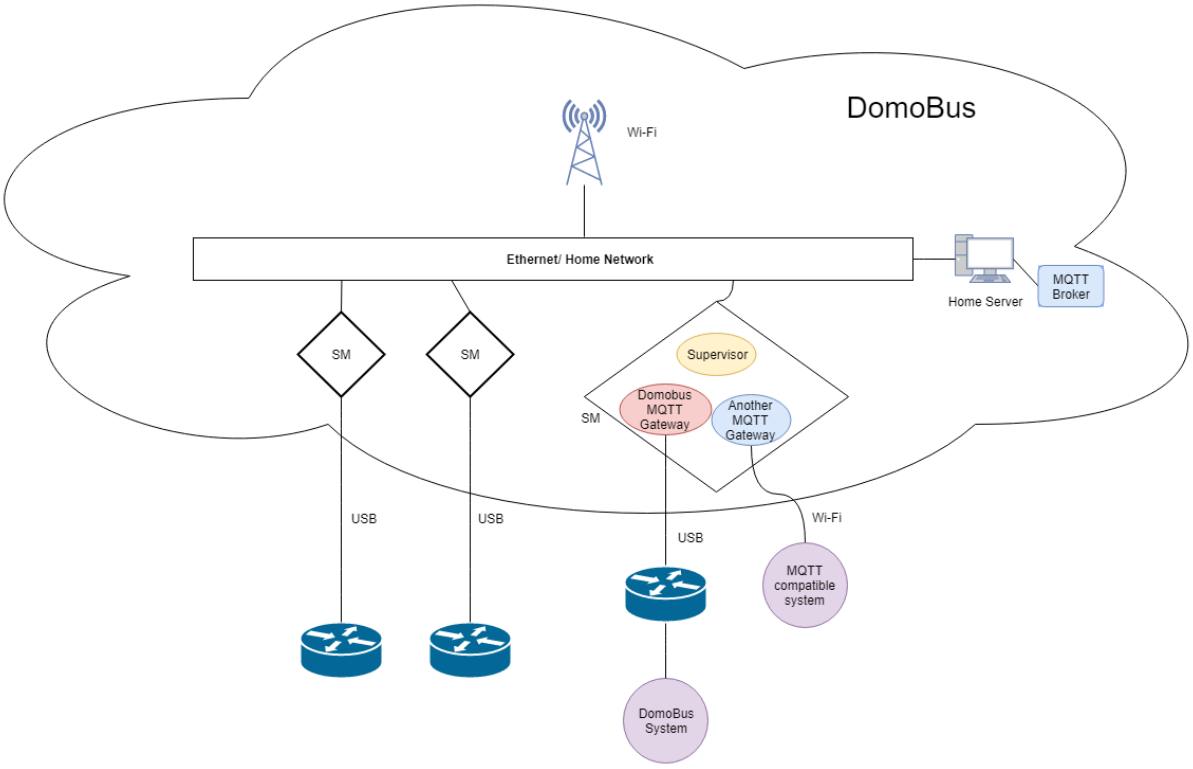

Figure 3.3: Example of the system topography with MQTT capabilities

section 2.2.1. Overall, the protocol is easy to implement, filters the messages and is one of the most used protocols in IoT projects.

As referenced before, to use MQTT, we need to use a MQTT Broker. All communications related to MQTT go through here.

3.3 MQTT Gateway

In the MQTT gateway, we have to make a decision about the organization of the information that will reside in the MQTT broker.

We propose to follow something along the specification already present in the DomoBus system because it is a simple specification and it has proved that it works. A device is an abstract entity with a set of properties. A DomoBus system is composed by a set of devices.

MQTT protocol, per se, does not follow any specification. In MQTT, the word topic refers to a string that the broker uses to filter messages for each connected client. The topic consists of one or more topic levels and each topic level is separated by a forward slash, considered a topic level separator.

By convention, the community has an architecture defined for the topic hierarchy. This convention states that the top level name will always to a specific instance of a hardware gateway. The second

Figure 3.4: Example of a multi-level topic

level defines a function such as status, set, get, command, etc. The subsequent level are defined by the specific gateway. Also, the gateway should maintain the topic connected on the second level, to specify if the gateway is currently connected to the broker.

Taking all these guidelines into thought, we propose the following architecture for the topics:

- First level topic stays defined as *domobus*, to refer the DomoBus System.
- Second level topic there are two functions: *status* and set, both self-explanatory. There is also the *connected* topic to ensure others device or gateways check the state of the connection before sending commands.
- Third level topic is where it will reside the identifier number of every device present in the DomoBus system.
- Fourth level topic is assigned the property or properties identifier of the device assigned in the third level topic.

The value of the property is published on the topic assigned that property.

3.4 XML parser

A DomoBus system specification is an XML specification. The DomoBus system has many aspects to take into account, the next topics are present in the specification and we analyse what is interesting and necessary to port to this gateway specification.

3.4.1 House structure

The house specification has a house identification and other data more personal to users, such as name, address and phone number.

A list of floors is described, having a height order to facilitate a visual organization. Following floors, the divisions are next, referencing the floor where each belong.

Even though this information is interesting to a more visual representation, it does not have a place in the MQTT specification, that way we don't over saturate the information needed to represent a DomoBus System to another systems.

3.4.2 Services

The name describes the functionality and that information is unnecessary to represent DomoBus in MQTT.

3.4.3 Devices

In this section all devices of the system are specified, containing information such as identification number in the house, every device has a unique identification number. Other information are related to what kind of device it is and where is located in the house.

Information related to what kind of device it is can be of value, but knowing that MQTT treats data agnostically, that information is irrelevant. The data focused on the location of the device can be ignored seen as we are not using the house structure information, it would be a complex way of referencing the device, so we choose not to use it to keep a simple architecture.

The devices is the main focus to keep in our specification. The devices are the backbone of our approach.

3.4.4 Scenarios

Scenarios is an optional specification on the DomoBus, its purpose is to facilitate some actions inside the DomoBus system. The premise of our project is to let the DomoBus System work on his own, just increase his capabilities. Since is an internal mechanism, there is no reason to it bring to our specification.

3.4.5 Favorites and Users

Favorites and users are connected, as the favorite list is referring the favorite devices and divisions of users present in the DomoBus system.

4

Implementation and Tests

Contents

4.1 Implementation

4.1.1 MQTT Broker

The broker is a vital piece in this system, but its only purpose is to store the data and to be the manager of the communication with other gateways that may connect to it. Knowing this, the implementation is fairly simple, the broker defines the port to configuration and Redis back-end, explained in 4.1.2.

4.1.2 Redis

Redis is an open source, in-memory data structure store, used as a database, cache and message broker. Redis is able to run a publish/subscribe model, and since the library adopted can utilize Redis to handle their publish/subscribe and store persistent data, we implemented it that way, as it gives more robustness to the project.

4.1.3 MQTT Gateway

The gateway holds the logic behind the communication to the MQTT broker. As explained in section 3.3, the topic hierarchy is defined by having the second level of topics defined the commands available in the system. When the gateway connects to the MQTT broker, the gateway is going to subscribe to the broker on the topic specified on the configuration file, by default the subscribed topic is domobus/. The gateway receives any message residing in the next levels of that topic.

4.1.4 Configuration

The MQTT gateway is equipped with a file of configuration. In this file is specified a few important settings.

- **mqtt**: the settings for the connection to the MQTT broker; the broker can be deployed locally or it can be used a broker platform online, for that scenarios we need to provide the **server address** and the **server port**. Related to the MQTT there is a possible scenario with multiple DomoBus networks available on the system, we need to specify the **topic prefix** on where our gateway is being deployed, to avoid collisions.

- **xml**: the path for the original xml file of specification for initial startup;

- **json**: the path for the parsed contents of the xml specification, ie, our json specification.

4.1.5 Topics

As discussed in the previous point, we specify the topic prefix, default being **domobus**. The sub topics following this topic have a purpose. These purposes are described in the next sub sections.

4.1.5.A Connected

The connected topic is used to show to another systems, that may be connected on the same network and working with MQTT, our DomoBus system and its corresponding gateway are running, showing the latest state of each device present in the DomoBus system and commands are able to be processed by the gateway.

4.1.5.B Status

The status topic discloses the current state of every property of every device present in the system. The specification is specified in chapter 3, in essence domobus/status/*device*/*property*/*value* is the architecture followed, being *device* the identification number of said device; *property* the property identification number of the same device, it can have multiple properties; *value* the corresponding value associated with the property and device referenced.

4.1.5.C Set

The set topic is a method of commanding the DomoBus system. It is useful when a Supervisor is deployed with a set of rules to accomplish automation mechanisms. The specification is specified in chapter 3, as status topic is. The architecture is very similar to the **status** domobus/set/*device*/*property* identifies the *device* and *property* and then publish the new *value*.

4.1.6 Procedures

At every message received it runs a function to filter the message to the corresponding function, the functions available by the gateway, introduced in Chapter 3.

When it relates to the *connected* status, the message is just ignored. The *status* of the devices is something to remain untouchable, because is information we wish to share. When the *set* function is triggered, the gateway should issue a set command to the DomoBus system.

The *status* is not considered a command, so whenever the devices connected to the gateway change any value of their properties, the DomoBus system should trigger a NOTIFY to the gateway to inform the value needs to be updated, and so, the gateway triggers a publish to the MQTT Broker when the new value is updated.

Having the specification it needs, the gateway publish the states of all devices according to the specification. All published messages are sent with the retain flag set to true, because every client that connects and wishes to know what is the state of the DomoBus system, it won't need to wait for an updated value to be published.

After every action in the gateway, the topic *connected* is updated, that way it indicates the gateway is still active. If the gateway is shutdown, it sends a last message to update the *connected* topic before it closes.

4.1.7 XML parser

When the gateway first launches, it should load an XML file with specifications of the devices present in the DomoBus system. The gateway uses the XML parse to filter the file, since there is a lot of important information present in a usual XML to use every feature of the DomoBus system. Some of that information is not pertinent to the gateway, it was needed a parser to filter the essential. The information that is relevant is described in chapter 3.The parser captures all devices and corresponding properties with default values to the minimum value of the property and writes in JSON format.

4.2 Tests

This system was tested on modern laptop running on Windows, which currently have a lot of resources, but the system was developed using NodeJS, an interpreter of JavaScript. NodeJS is able to run on Arduino and Raspberry Pi. NodeJS also runs on Windows, MacOS and Linux. Another software presented in this system is Redis. Redis can also run on Arduino and Raspberry Pi, supports both Linux and MacOS. Redis native does not run on Windows, but in Windows 10, there is a feature called Windows Subsystem for Linux (WSL), gives ability to run applications for Linux distros in Windows. In our setup, Redis was running in the WSL.

4.2.1 XML parser

The first feature tested was the load of a XML specification into the broker. The file in the test was loaded successfully and the resulting JSON file showed the proposed specification, both contetns in Appendix A.

4.2.2 MQTT gateway

After parsing, the next step was load the JSON contents in the MQTT gateway to publish the state of the DomoBus system. With the help of a tool, designated by MQTT Explorer, it was possible to have a

visual representation of the topic hierarchy imported (figure 4.1).

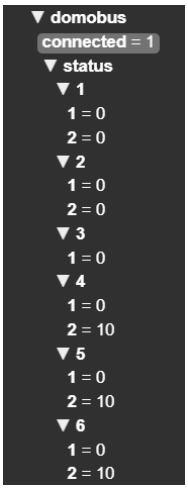

Figure 4.1: Visual Representation of topic hierarchy

The areas to check in the test are:

- **state of the gateway updated**: this point checks the use of the topic *connected*.
- **state of devices updated**: this point checks if the devices are updated.
- **persistence of messages**: this point checks if the messages that are publish, are sent with the retain flag, being sent to new subscribers of those topics.

Again, with the help of MQTT Explorer, most tests were checked.

Closing the gateway resulted in the topic *connected* changing to 0 and only when run again it would set to 1 again. First point is checked.

Unfortunately, since the communication with a real DomoBus system was not possible, the test that follows was done with a simulated DomoBus system, just to show how the communication between two systems with different technologies was successful.

We imagined a scenario: a user is enjoying a movie in the living room at night, only the lights are off and the movie is playing. Suddenly, the user needs to take a break to go to the bathroom, and since all the lights are off, and he has to find the remote to pause the movie. A simple solution would be to pause the movie as soon as he turns on the light and resume play as soon as he turns off the light. Based on this scenario, the test was setup using a media center, named Kodi, and a gateway for Kodi, that was available as a addon for Kodi.

The test setup and Kodi don't speak the same language, Kodi uses JSON-RPC for communicating. The test put both system in the same environment and along with it a supervisor with two rules: a specific device in the simulated DomoBus changed a property value to 1, meaning that, it would send

the command to pause video; The same device changed property to 0, meaning off, it would send the command to resume. The test was successful and there was no delay in pausing and resuming. This checks the second point, as the simulated DomoBus changed its value, it was immediately updated in the MQTT broker, and after that it reacted in the supervisor with a command for the media center.

To check the last point, a client of MQTT was developed with a simple task: subscribe to domobus/ print any message received. Subscribing to domobus/ means all subtopics are subscribed as well. First, the broker and gateway were launched, then the simple client. When the client connected received all the messages the gateway had sent: *connected* and the devices *states*.

5

Conclusion

Contents

5.1 Conclusions

The main goal was to create a bridge between what is currently the IoT scene and the DomoBus system.

The current scene is a fragmented market, with some companies having their own ecosystem on the market, but not all ecosystems talk to other ecosystems. Some companies and group are making an effort by embracing new technologies into their products. Such technologies were developed thinking of a united IoT system. MQTT is a protocol developed first because of a need outside IoT, but later adopted with the purpose to become the starting point of a IoT standard .

This project is all about MQTT, because in our point of view it is the next step to bring the DomoBus system closer to a more connected home automated system. Meaning, if users wish to build and configure their own network of devices using DomoBus and want to connect a retail IoT product to their network, it is closer to reach that. By using an architecture and a system capable of joining both technologies, like the one presented, the task is simpler.

Unfortunately, we were unable to connect a real DomoBus system to this project, the system works as it should, has the right architecture and methodology, but the communication to a DomoBus system is lacking. So, a full prototype working was not achieved.

The system allows a user to deploy its own local broker and if wanted the gateway can connect to an outside MQTT broker. Users can also connect others gateways to connect with another MQTT capable ecosystem and those ecosystems can perform has they normally would, with their own set of rules. The system grants the power to build a set of rules between two or more separate ecosystem and attach to the system. This project grants the opportunity to use community projects of home automation to define these set of rules between the gateways, projects like the ones discussed in section 2.3.

With this solution, we expect to have made progress in the development of a unified smart home system, like the DomoBus system, who is flexible and allows a lot of customization and with that bring it closer to the current scene of IoT development.

5.2 System Limitations and Future Work

The project has some work for a future project so it can be ready for deployment, such as integrate the DomoBus Communication Module with this gateway in order to interact with a real DomoBus system.

Some ideas that will improve this system are:

- Use the historic recorded by the gateway to feed an Artificial Intelligence Machine so that it may predict behaviours of the system.
- Add the capability of also work with rooms like the DomoBus system, also working with the system as a whole.

- Incorporate security access to the gateway.
- Advance with the idea of running an MQTT broker as the central piece in a home automation system and deploy several already compatible technologies with it.

Bibliography

[1] Open Connectivity Foundation. (2016) OPEN CONNECTIVITY FOUNDATION BRINGS MASSIVE SCALE TO IOT ECOSYSTEM. Accessed 20-February-2019. [Online]. Available: https://openconnectivity.org/news/open-connectivity-foundation-brings-massive-scale-to-iot-ecosystem

[2] Knud Lasse Lueth. (2018) State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating. Accessed 5-April-2019. [Online]. Available: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/

[3] H.-J. Langels, "KNX IP – using IP networks as KNX medium ," WHITEPAPER, 2008. [Online]. Available: https://www.auto.tuwien.ac.at/knxip/KNX%20IP%20-%20Using_IP_networks_as_KNX_medium.pdf

[4] Maggie Tillman, Chris Hall. What is ZigBee and why is it important for your smart home? Accessed 5-May-2019. [Online]. Available: https://www.pocket-lint.com/smart-home/news/129857-what-is-zigbee-and-why-is-it-important-for-your-smart-home

[5] Amazon. (2018) Alexa Connected Devices - Connect Your Devices Locally Using Zigbee. Accessed 22-September-2018. [Online]. Available: https://developer.amazon.com/alexa/connected-devices/zigbee

[6] R. Andrew Banks. (2014) OASIS Standard - MQTT Version 3.1.1. Accessed 2-November-2018. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

[7] I. Eclipse Foundation. Eclipse IoT Standards. Accessed 2-November-2018. [Online]. Available: https://iot.eclipse.org/standards/

[8] Amazon. Understand the Smart Home Skill API. Accessed 11-November-2018. [Online]. Available: https://developer.amazon.com/docs/smarthome/understand-the-smart-home-skill-api.html

[9] Zigbee. (2019) mazon to Join Zigbee Alliance Board of Directors. Accessed 6-February-2019. [Online]. Available: https://www.zigbee.org/amazon-to-join-zigbee-alliance-board-of-directors/

[10] Aaron Tilley. (2014) Samsung Acquires SmartThings, A Fast-Growing Home Automation Startup. Accessed 21-January-2019. [Online]. Available: https://www.forbes.com/sites/aarontilley/2014/08/14/samsung-smartthings-acquisition-2/#199bf8303367

[11] Renato Nunes, "DomoBus - A New Approach to Home Automation." 8CLEEE - 8th International Congress on Electrical Engineering, July 2003.

A

File for test

The following XML file is the file used in the test, having a complete description of a DomoBus System, we the purpose of parsing all important information.

Listing A.1: Complete XML file used in test

```
<DomoBusSystem ID="1" Name="Example System" Version="01" Date="01/03/2016">
<!-- Value types -->
  <!-- NumBits: 8 or 16 -->
  <ScalarValueTypeList>
    <ScalarValueType ID="1" Name="light intensity" NumBits="8" Units="%" MinValue="0" MaxValue="100"
        Step="5"/>
    <ScalarValueType ID="2" Name="heat duration" NumBits="8" Units="Sec" MinValue="10"
        MaxValue="120" Step="10"/>
  </ScalarValueTypeList>

  <EnumValueTypeList>
```

```
10      <EnumValueType ID="1" Name="on/off state">
11        <Enumerated Name="Off" Value="0" />
12        <Enumerated Name="On" Value="1" />
13      </EnumValueType>
14      <EnumValueType ID="2" Name="presence">
15        <Enumerated Name="Inactive" Value="0" />
16        <Enumerated Name="Active" Value="1" />
17      </EnumValueType>
18      <EnumValueType ID="3" Name="state">
19        <Enumerated Name="Off" Value="0" />
20        <Enumerated Name="Toast" Value="1" />
21        <Enumerated Name="Keep warm" Value="2" />
22      </EnumValueType>
23    </EnumValueTypeList>
24
25    <ArrayValueTypeList>
26      <ArrayValueType ID="1" Name="sound volume" MaxLen="3">
27        <ValueConversion Type="1" Ref="min" />
28        <ValueConversion Type="2" Ref="med" />
29        <ValueConversion Type="3" Ref="max" />
30      </ArrayValueType>
31    </ArrayValueTypeList>
32
33    <DeviceTypeList>
34      <DeviceType ID="1" Name="Dimmable light" Description="">
35        <PropertyList>
36            <!-- Value types: "SCALAR", "ENUM" or "ARRAY" -->
37          <Property ID="1" Name="on/off state" AccessMode="RW" ValueType="ENUM" RefValueType="1" />
38          <Property ID="2" Name="light intensity" AccessMode="RW" ValueType="SCALAR"
                RefValueType="1" />
39        </PropertyList>
40      </DeviceType>
41      <DeviceType ID="2" Name="Movement detector" Description="">
42        <PropertyList>
43            <!-- Value types: "SCALAR", "ENUM" or "ARRAY" -->
44          <Property ID="1" Name="presence" AccessMode="RO" ValueType="ENUM" RefValueType="2" />
45        </PropertyList>
46      </DeviceType>
```

```
47      <DeviceType ID="3" Name="Toaster" Description="">
48              <!-- Value types: "SCALAR", "ENUM" or "ARRAY" -->
49        <PropertyList>
50          <Property ID="1" Name="state" AccessMode="RW" ValueType="ENUM" RefValueType="3" />
51          <Property ID="2" Name="heat duration" AccessMode="RW" ValueType="SCALAR"
                RefValueType="2" />
52        </PropertyList>
53      </DeviceType>
54      </DeviceTypeList>
55
56    <AccessLevelList>
57      <AccessLevel Level="8" Name="Admin" />
58      <AccessLevel Level="5" Name="Owner" />
59      <AccessLevel Level="1" Name="Guest" />
60    </AccessLevelList>
61
62    <UserList>
63      <User ID="1" Name="Mr. Silva" Password="1234" AccessLevel="8" />
64      <User ID="2" Name="Mrs. Silva" Password="4321" AccessLevel="5" />
65    </UserList>
66
67    <House ID="123" Name="Vivenda Silva" Address="-">
68      <FloorList>
69        <Floor ID="1" Name="Entrance" HeightOrder="0" />
70        <Floor ID="2" Name="Floor 1" HeightOrder="1" />
71        <Floor ID="3" Name="Attic" HeightOrder="2" />
72        <Floor ID="4" Name="Basement" HeightOrder="-1" />
73      </FloorList>
74      <DivisionList>
75        <Division ID="1" Name="Kitchen" RefFloor="1" AccessLevel="5" />
76        <Division ID="2" Name="Living Room" RefFloor="1" AccessLevel="5" />
77        <Division ID="3" Name="Small WC" RefFloor="1" AccessLevel="5" />
78        <Division ID="4" Name="Bedroom" RefFloor="2" AccessLevel="5" />
79        <Division ID="5" Name="Suite" RefFloor="2" AccessLevel="5" />
80        <Division ID="6" Name="WC" RefFloor="2" AccessLevel="5" />
81        <Division ID="7" Name="Store-room" RefFloor="3" AccessLevel="5" />
82        <Division ID="8" Name="Gym" RefFloor="4" AccessLevel="5" />
83      </DivisionList>
```

```
84  </House>
85
86  <ServiceList>
87    <Service ID="1" Name="Lighting" />
88    <Service ID="2" Name="Entertainment" />
89    <Service ID="3" Name="Security" />
90  </ServiceList>
91
92  <DeviceList>
93    <Device ID="1" RefDeviceType="1" Name="Kitchen lamp" Address="100" RefDivision="1" />
94    <Device ID="2" RefDeviceType="1" Name="Bedroom lamp" Address="101" RefDivision="4" />
95    <Device ID="3" RefDeviceType="2" Name="People detector" Address="200" RefDivision="2" />
96    <Device ID="4" RefDeviceType="3" Name="Kitchen toaster" Address="300" RefDivision="1" />
97    <Device ID="5" RefDeviceType="3" Name="Dining toaster" Address="301" RefDivision="2" />
98    <Device ID="6" RefDeviceType="3" Name="Ceiling lamp" Address="302" RefDivision="2" />
99      <DeviceServiceList>
100       <DeviceService RefService="1" />
101       <DeviceService RefService="3" />
102     </DeviceServiceList>
103   </Device>
104 </DeviceList>
105
106 <ScenarioList>
107   <Scenario ID="1" Name="Arriving home">
108     <ActionList>
109       <Action ID="1" RefDevice="1" RefProperty="1" Value="1" />
110       <Action ID="2" RefDevice="1" RefProperty="2" Value="75" />
111       <Action ID="3" RefDevice="4" RefProperty="1" Value="1" />
112     </ActionList>
113   </Scenario>
114   <Scenario ID="2" Name="Going to sleep">
115     <ActionList>
116       <Action ID="1" RefDevice="1" RefProperty="1" Value="0" />
117       <Action ID="2" RefDevice="2" RefProperty="1" Value="0" />
118       <Action ID="3" RefDevice="4" RefProperty="1" Value="0" />
119       <Action ID="4" RefDevice="5" RefProperty="1" Value="0" />
120     </ActionList>
121   </Scenario>
```

```
  </ScenarioList>

  <Favorites>
    <FavoriteList RefUser="1">
      <FavoriteDevice RefDevice="2" />
      <FavoriteDivision RefDivision="1" />
    </FavoriteList>
    <FavoriteList RefUser="2">
      <FavoriteDevice RefDevice="1" />
    </FavoriteList>
  </Favorites>

</DomoBusSystem>
```

The file shown below is the final product of parsing the devices information from the previous XML file.

Listing A.2: JSON file after XML parser

```

[{
    "ID": 1,
    "Property": [{
        "ID": 1,
        "value": 0
    },
    {
        "ID": 2,
        "value": 0
    }
    ]
},
{
    "ID": 2,
        "Property": [{
        "ID": 1,
        "value": 0
    },
```

```
20      {
21          "ID": 2,
22          "value": 0
23      }
24      ]
25  },
26  {
27      "ID": 3,
28      "Property": [{
29          "ID": 1,
30          "value": 0
31      }]
32  },
33  {
34      "ID": 4,
35      "Property": [{
36          "ID": 1,
37          "value": 0
38      },
39      {
40          "ID": 2,
41          "value": 10
42      }
43      ]
44  },
45  {
46      "ID": 5,
47      "Property": [{
48          "ID": 1,
49          "value": 0
50      },
51      {
52          "ID": 2,
53          "value": 10
54      }
55      ]
56  },{
57      "ID": 6,
```

```
58      "Property": [{
59          "ID": 1,
60          "value": 0
61      },
62      {
63          "ID": 2,
64          "value": 10
65      }
66      ]
67  }]
```