Domotic Devices Emulator

Miguel Fonseca Gonçalves Martinho
miguel.martinho@tecnico.ulisboa.pt
Integrated Master's Degree in Electrical and Computer Engineering
Instituto Superior Técnico - University of Lisbon, Lisbon, Portugal

November 2016

Abstract

Home automation needs to be simple and pleasant to use, because that is the only way to reach the needs of the users in the development of smart houses. Currently, there are several domotic technologies that stand out, for example, KNX, LonWorks and X10. However, these technologies are incompatible with each other, usually they have a high cost and require the use of specific tools to design and implement a system, which is difficult to understand for the average user. Recently, new technologies have emerged in the context of the Internet of Things (IoT), such as, Samsung ARTIK, Google Brillo and Apple HomeKit. Nonetheless, they are also incompatible with each other and the existing tools for development and testing are underdeveloped. In this context, it was designed, in Java, a domotic devices emulator based on the DomoBus system, which allows to emulate any device and domotic system (simple or complex), without needing the actual devices or depending on their physical installation. The DomoBus system uses a generic model to characterize domotic devices and it has a specification language based on XML that allows the designing of any domotic system. These features give the DomoBus system great generality and flexibility, making it independent of any technology. The emulator allows the emulation of complex domotic scenarios, simplifying the design, development and validation of sophisticated behaviours. Thus, it becomes possible to check whether the user's preferences are satisfied, helping to perfect functionalities for his house, even before the system is physically installed.

Keywords: Home Automation, Smart House, Domotic Devices Emulator, DomoBus, Domotic Scenarios.

1. Introduction

1.1. Background and motivation

The term Domotics results from the Latin word *Domus* (house) with the word Telematics (telecommunications + computer). When this came up in the 80s, it was intended only to control the lighting, air conditioning and security of a system and the interconnection between the three, but now includes centralized control of lighting, Heating, Ventilation and Air Conditioning (HVAC), points of communication (television, Internet and phone) and security systems.

Among the various advantages offered by the home automation, it highlights the increased level of comfort and security (for example: motion sensors connected directly to alarms), automation of tasks (for example: scheduling to turn on certain electronic devices) and greater energy efficiency (for example: presence detectors to manage lighting, thermostats of heaters and air conditioners).

The popularity of home automation has increased significantly in recent years due to a greater accessibility and diversity, combined with a contribution of the developments made in the area of Internet of Things (IoT) [1]. However, there are still some problems due to competition between suppliers, various incompatible technologies in the area of domotics (EIB/KNX, X10, LonWorks), as well as in the area of IoT (Samsung ARTIK, Google Brillo, Apple HomeKit), high costs of installing a home automation system, and difficulty in using the system, which has limited the existence of more "Future Houses" or "Super-automated Houses" [2]. These are houses with a large number of home automation devices (sensors and actuators) with a variety of integrated functionalities. All these factors have meant that home automation has not had such a relevant role in the actual society as it would be expected.

It is in this context that the DomoBus system and the emulator play an important role in the area of home automation, giving a contribution to the integration of different technologies and the development and testing of complex behaviours, even before the physical implementation of the system.

The DomoBus system was developed by Professor Renato Nunes of Instituto Superior Técnico - University of Lisbon. It is a generic, open and flexible system, in that, the DomoBus can represent any type of home automation device, and any home automation system, through an abstraction model and a specification language based on XML. Furthermore, these

properties of DomoBus allow the existence of heterogeneous systems with different technologies, contributing to higher interoperation between the various devices and supporting the increase of smart houses.

Thus, it is intended that with the emulator created in this dissertation, it will be possible to emulate the behaviour of various domotic devices, which together with supervision applications (applications that receive information of the status' devices and, according to dynamically defined rules, determine what actions the system must perform) will make it possible to view and test domotic scenarios over time.

1.2. Objectives

The aim of this thesis is to design and develop an application based on the DomoBus system that allows the emulation of any device of a home automation system (sensor or actuator), regardless of its technology, without having physical devices and without resorting to physical installation, simplifying the development and validation of complex behaviours of the system that can involve multiple devices.

Moreover, it will be possible to easily adjust the behaviour of devices in the emulator for different situations, according to the needs of the users. The definition of these behaviours can be done iteratively, being subsequently adjusted according to users' preferences, as they often do not have a concrete idea of what they want and only after observing the operation of the domotic system, conclude what is best and most suitable for them.

To facilitate the interaction of the emulator with the user, the user can check and change the status of each domotic device in the emulator. Each device will have its own visual representation, which can change according to the current state of the device, for easy perception of the user. By selecting a device, it will be possible to view all its properties and easily change its values. For example, the user can see if a Lamp device is turned on or turned off by its graphic representation (image or animation).

All the features of a domotic system (house, devices, behaviours, etc.) are defined by a specification language, based on XML, that uses an abstraction model, which makes this language very flexible and expressive. Thus, it becomes possible to describe all the features of the system in a single XML file, which can be read by any application that takes in the necessary information for its operation. Moreover, it becomes easy at any time to modify configuration settings of certain domotic devices so the system fits the users' needs or preferences, or add new devices to a system, modifying the contents of the specification and introducing the images or animations representing the different possible states of the device.

The programming language chosen to implement the emulator is Java language, as it is a highly known multi-platform language and has a great online support, being adequate for graphics applications.

In terms of the market for home automation, it is important for technicians of domotic companies to show, to their customers, the best organization of home automation devices in the system to meet their preferences and ideas (through the use of the domotic scenarios).

In conclusion, the emulator to be developed within this thesis will be a very useful tool to test domotic systems, whatever its devices, allowing to view and verify if a given solution meets the desires of the users, even before physically implemented.

2. State-of-the-Art

2.1. Characterization of a home automation system

The main existing elements in a home automation system are as follows:

- **Sensors** - devices whose purpose is to detect events or changes in the environment in question, measure them and provide the related results (for example: temperature sensor);
- **Actuators** - components that a control system uses to act on an environment, based on a source of energy, typically electrical energy (for example: air conditioning);
- **Interfaces** - allow the human-system interaction. They may be simple (using buttons and leds) or complex (using, for example, a touch-sensitive screen or a computer), with an exchange of information between the two communication systems;
- **Connections** - permit the transmission and reception of data between the components of the home automation system, in various ways (cables, wireless, ...);
- **Controllers** - devices, such as microprocessors or computers, which monitor and change dynamically the mode of operation of a home automation system.

2.2. Home automation technologies

There are many technologies in the field of home automation, such as **KNX** that has evolved from the **EIB**, **LonWorks** and **X10**.

2.2.1. EIB/KNX.

The communication protocol EIB was developed by a group of leading companies in the European market for electrical equipment (Berker, Gira, Jung, Merten and Siemens AG). The EIB was developed with the primary aim to create a European standard and thus create an alternative system to the more developed systems available in the markets of Japan and the United States of America. The EIB system allows all electrical devices, such as switches, electric motors and sensors, to be interconnected and behave as a decentralized system in which each device can send

or receive commands from other components in the system, regardless of its location. This system has the possibility of communicating through the electrical housing network, twisted pair cable, RF signals and IR signals.

Convergence of this protocol and the communication protocols, Batibus and EHS, resulted in the KNX protocol.

The KNX was promoted by three European associations (EIBA, BCI and EHSA) and was established in May 1999. It is now run by the KNX Association. The KNX system is a standardized communication protocol based on the OSI model [3]. The operation of this system is based on a central cable to which all devices are connected via a communication medium, such as RF signals, IR signals, electric housing network, IP/Ethernet and twisted pair cable [4]. KNX is robust, has diverse features and high transmission rate, but its configuration and installation are complex and with high costs.

The software tool called ETS [5] allows the configuration of KNX facilities and the programming of the KNX devices by a specialist.

2.2.2. LonWorks.

The LonWorks protocol is a communication protocol developed by the company Echelon Corporation in 1992, being a direct rival of the protocol KNX due to being similar technologies. The LonWorks is based on a network of intelligent control devices (nodes), called LON, which use a common protocol, called LonTalk, to communicate with each other. These nodes contain special microcontrollers, called Neuron Chips, developed by the company Echelon, which allow the implementation of the protocol LonTalk and the execution of all the control functions. The nodes (sensors, actuators or interfaces to the operator) also have a physical interface (transducer) which allows its connection to the physical communication medium. The physical media are varied, including the electric housing network, twisted pair cable, RF signals, IR signals, coaxial cable, optical fibre and Internet. The LonWorks devices communicate with each other, mainly, through network variables in order to describe the data to be transmitted. A device can implement up to 62 network variables. The standard network variables are called SNVTs, and are used to transmit predefined data, such as time, temperature, speed or position. The advantages and disadvantages of LonWorks are similar to the ones of KNX.

The software tool LonMaker [6] is the application commercialized by the company Echelon for the design and configuration of home automation systems based on the technology LonWorks, as well as the documentation and maintenance of it.

2.2.3. X10.

The X10 protocol was developed in 1975 by the Scottish company Pico Electronics of Glenrothes to enable remote control of devices in a house. However, the patent has expired and now any company can produce X10 compatible modules. The X10 is a communication protocol that uses the electric housing network as a single physical medium. Each X10 device has an address consisting of a pair letter-number, where the letter is between A and P and the number between 1 and 16. Thus, it is possible to have up to 256 devices in a domotic system. For the transmission of commands, a 5-bit number is used, in which the last bit indicates whether it is a device (bit 0) or a function (bit 1). The X10 is easy to use and has low cost modules, but has few features and a low rate of transmission.

2.2.4. Control Applications.

There are several applications that allow monitoring and controlling of home automation systems composed by devices that use the technologies described above, such as, **HomeSeer HS3** [7], **ActiveHome Pro** [8] and **Control4 Composer** [9]. These applications, which have a graphical interface, allow only the control (not simulating or emulation) of the existing devices in the home automation system and interact in a simple and enjoyable way with the users. However, these applications have the disadvantage of being associated with specific technologies (HomeSeer HS3 - Insteon and X10, ActiveHome Pro - X10 and Control4 Composer - KNX) and being controlled by private companies.

2.3. Technologies in the context of IoT

Recently, there have been several technologies in the context of IoT, namely **Samsung ARTIK**, **Google Brillo** and **Apple HomeKit**, for use in the users' house. These technologies are at an early stage and represent an attempt of big companies like Apple, Google, Samsung, Amazon and Microsoft, to enter the market of home automation. These new solutions will allow the creation of an intelligent network for multiple electronic devices in the house and that can be extended also to vehicles, smart watches and other equipments including sensors and actuators and with connection to the Internet for the collection and exchange of data between them.

2.3.1. Samsung ARTIK.

Samsung ARTIK is a fully integrated IoT platform, this is, it has its own modules (Samsung ARTIK Modules [10]), development tools and its own network (Samsung ARTIK Cloud [11]), forming an ecosystem that allows a quick and effective communication between the various IoT products.

Samsung ARTIK Modules are integrated SoM, specially made for the area of the IoT. There are three types of Samsung ARTIK modules, in particular ARTIK 1, ARTIK 5 and ARTIK 10, with an increase in size and functions. These modules also have network packages for the connection to the Samsung ARTIK Cloud, so that it is possible to integrate an entire network of IoT devices, and an open software environment (Samsung ARTIK IDE), which allows

users to use the APIs of this platform to create and modify applications based on Samsung ARTIK. With this architecture of modules, made for optimization of power consumption, size, memory and performance, it is possible to perform various applications, from simple implementations in devices to powerful hubs with local processing and data analysis.

Samsung ARTIK Cloud is a platform for data exchange that allows to connect all Samsung ARTIK devices and applications and cloud services with each other, through various communication protocols. The security of the connection between the various devices is ensured through the use of TLS, the hardware SE and certificates issued by a TCA.

2.3.2. Google Brillo.

Google Brillo is an IoT platform based on three main elements: embedded operating system based on the Android system, essential services platform and a kit of development tools. Brillo is supported by hardware based on the architectures ARM, Intel x86 and MIPS, having a wide variety of hardware features and customization to the wishes of the users. It is also easy to prototype and produce. With the development tools, it is possible to manage network devices via OTA updates, statistics data and crash reports.

Google Weave is the Google Brillo communication platform. It is a protocol that allows the configuration of devices safely and consistently, the exchange of information between devices directly, or between devices and the cloud, and user interaction with the devices from mobile devices and computers.

2.3.3. Apple HomeKit.

Apple HomeKit is an IoT structure of communication and control of accessories/devices that support the Apple's HomeKit Accessory protocol. Certain applications for HomeKit will allow the configuration of the devices. The information of configuration is stored in a database in the user's iOS device and can be synchronized with other iOS devices via iCloud. The devices can be of various types, namely, thermostats, lights, cameras, gates or sensors. These accessories communicate across the iCloud to receive instructions and send information to the user's iOS device. Users can also control the devices using voice, creating actions and clusters of them in advance and then executing them using the Siri.

2.3.4. Simulators.

With the research carried out, two simulators that simulate the behaviour of home automation devices were found, namely **ARTIK Cloud Device Simulator** [12] and **HomeKit Accessory Simulator** [13].

ARTIK Cloud Device Simulator is a Samsung command-line simulation tool, developed in Java, to send predefined messages and data, for the Samsung ARTIK Cloud, simulating a real device in a Samsung ARTIK system. The ARTIK Cloud Device Simulator can operate in two possible modes:

- **Interactive Mode** - the user can execute commands, such as trying to get information about a certain device and seeing the results;
- **Automatic Mode** - the user can run predefined scenarios, retrieved from a JSON configuration file, and sets of data stored previously. The simulator can be used to generate data, such as for the behaviour of a sensor.

HomeKit Accessory Simulator is an Apple's tool to simulate accessories/devices, without having to physically have them, to test the communication with the control application Home of Apple HomeKit. In the simulator, it is possible to add accessories/devices to the simulated network, which have services with well-defined characteristics that the user can control and change their values. These devices, the relationship between them and their location in the simulated environment are stored in Apple HomeKit's database. In order to simulate devices that do not support the Apple HomeKit Accessory protocol, it is possible to create a bridge in the simulator and then add these accessories to it, which will allow the matching of the characteristics of the devices.

2.4. Balance

The domotic technologies (EIB/KNX, LonWorks, X10) have a good level of functionality and have been in the home automation market for a long time and there is a lot of information about them and many types of modules. On the other hand, the technologies in the context of IoT (Samsung ARTIK, Google Brillo, Apple HomeKit) are quite recent and are still in a development phase in the field of home automation, but have a promising way thanks to a good structure of modules and communication platforms.

Despite these positive points, these home automation technologies, and in the context of IoT, are totally dependent on private companies, they are not always open, there are not many manufacturers to produce compatible products and even issues with security and data confidentiality can be raised. So, there is less freedom in the choice of the consumer for the domotic system for his house.

According to the research work carried out, it was not possible to find simulators for home automation technologies, but only applications that control and monitor home automation systems (control applications). For the technologies in the context of IoT, there are two simulators, the ARTIK Cloud Device Simulator for the Samsung ARTIK and HomeKit Accessory Simulator for the Apple Homekit, that are a good approach to the emulator proposed in this dissertation. However, they are dependent on the respective companies and are not very developed in terms of features and types of devices.

From the analysis of the technologies described, it is concluded that the existing home automation systems and technologies in the context of IoT fall short of the functionalities of the DomoBus system and the emulator, in terms of its generality and flexibility to

represent devices of different protocols and adapt the behaviour of the system to the preferences of users.

3. The DomoBus System

3.1. Architecture

The architecture of the DomoBus system is, briefly, a distributed architecture, where a set of different modules connected together is presented as a single and coherent system. In Figure 1 the architecture of the DomoBus can be seen, which is essentially composed by Control Modules (CM) and Supervision Modules (SM), which are connected by a communication network that allows the exchange of information between them.

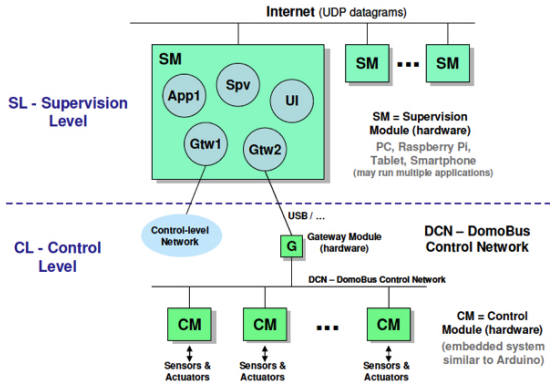

Figure 1: Architecture of the DomoBus System [14]

The control modules include microprocessors similar to those used in Arduino [15], which are connected to sensors and actuators. Thus, the control modules can read or act on physical devices such as light switches, motors, motion sensors, lights or even TVs or air conditioners by generating IR signals.

The control modules may be connected to one or more supervision modules through gateways.

The supervision modules are implemented using ordinary computers or modules like Raspberry Pi [16] or SBCs. These modules are responsible for overseeing certain CMs specified in the DomoBus architecture. The SMs receive information from the CMs, process this information, according to predefined rules and required behaviours, and send commands to be executed by several control modules. A DomoBus system can contain one or more supervision modules. These modules also allow the connection of the system to the Internet, enabling remote monitoring of it.

The interconnection with different domotic technologies is done via specific gateways. Each gateway has a table with the information of the addresses for the different target technologies and a conversion table of commands received from one technology to the DomoBus system. In brief, a gateway translates the operating logic of the DomoBus system for the various domotic technologies.

The DomoBus system only uses three types of messages to interact. These messages are:

- **GET** - allows reading the value of a property of a device, being able to view its status;
- **SET** - assigns a new value, chosen by the user, to a property of a device, changing its state;
- **NOTIFY** - this message is automatically sent during a change of a value of a property of a device, so that other applications, such as supervisors, can always be updated, without the need to always be sending GET messages to obtain the status of the devices. Additionally, the communication will be more effective and there is a decrease in the response time of the update of the system, by reducing the number of messages exchanged.

3.2. Abstraction models

The DomoBus system is based on abstraction models to represent a domotic system. These models allows the establishment of the physical structure of a house and identifies the existing domotic devices' location (floor and room). It also allows the representation of any device for its properties and services, as well as defining scenarios. Thus, with these models it is possible to apply the DomoBus system to any home automation system.

The house's abstraction model shows the description of the house and where the domotic devices are located. It defines the house as having one or more floors, which can have several rooms or spaces with specific interests to the users, containing a number of different domotic devices.

The device's abstraction model is divided into two sections. One is the definition of the generic type of the device based on its properties. The other section refers to the existing device in the system, with its specified type of device. This specification is flexible, because it is easy to modify or add new domotic devices to a given system; and it is generic, because it can represent any device, regardless of its structure.

It also presents an approach to a scenario's abstraction model, which uses IF ... THEN ... ELSE formulations that allow the user to configure and adapt the home automation system's behaviour according to their needs and desires, such as pre-defined actions for romantic dinners, parties and more. An example of a scenario is "On/Off all the house lights". In this scenario IF the value of the brightness property of the light sensor is below a certain value, THEN the actions executed will assign the enumerated value On to the property On-Off of all light devices of the house, ELSE is assigned the value Off to the property On-Off of all the light devices.

3.3. Specification language

The specification language used to describe the DomoBus system is based on a XML file, which defines the types of devices and their types of property, the structure of the house and its home automation system, existing devices in the system and also

the different types of services of the domotic devices. Thanks to the use of the XML language, it is possible to expand the specification of a system and include more details that are revealed of interest at any time. So it will be possible to describe any system with any combination of devices and installed on any house.

4. Solution description

4.1. Architecture

In Figure 2 is exposed a graphical representation of the global architecture of the emulator of domotic devices, developed in this dissertation.

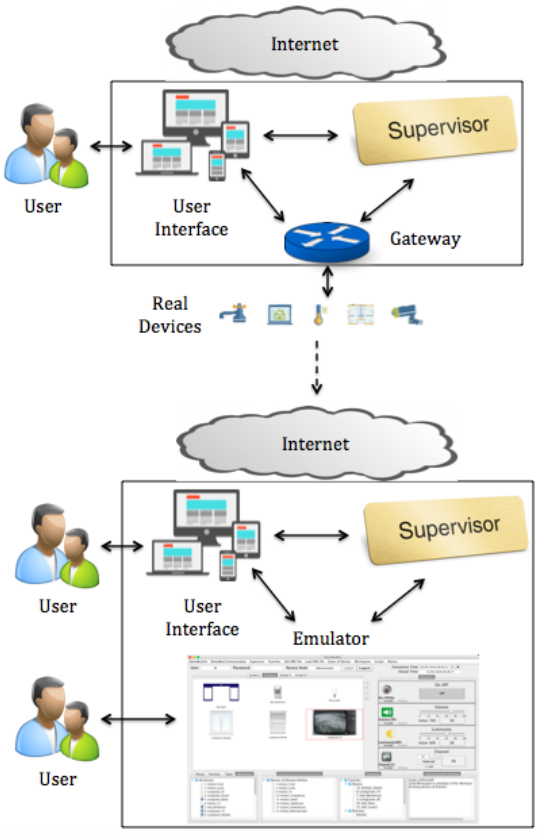

Figure 2: Global architecture of the emulator

The proposed solution to the emulator of domotic devices has the objective of a user instead of purchasing expensive devices, to control the house in an intelligent and efficient manner, it can in advance emulate the behaviour of these devices with the help of an application functioning as a supervisor of their behaviour. In other words, the user can test domotic scenarios with the emulated devices in the application, setting their dependency relationships, such as, the relationship between a switch and a lamp. With this emulator, the user can configure and shape the domotic system that he idealizes for his house, according to his needs and preferences.

4.2. Operating mode

The emulator created in this dissertation will give the users the ability to test a domotic system. In Figure 3, there is a brief explanation of the operating mode of the emulator.

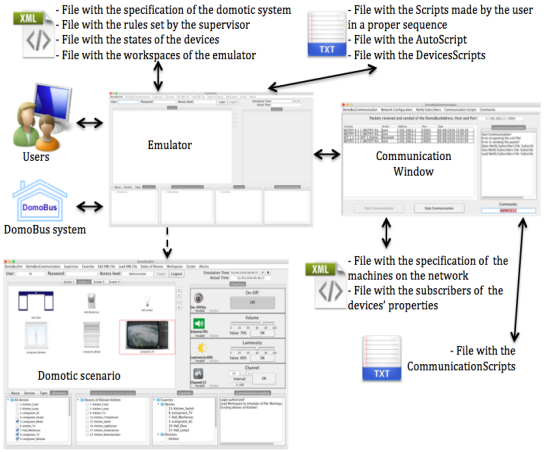

Figure 3: Operation mode of the domotic devices emulator

The emulator has the ability to read the XML file of a domotic system. In addition, the user can load the files with states of the devices and the workspaces of the emulator to resume previous domotic scenarios. Thus, the user can test various domotic scenarios using the functionalities Scripts, DevicesScript and AutoScript and communicating with an application as a supervisor, adjusting the behaviour of the domotic system according to his wishes and needs.

4.3. Main functionalities

The proposed emulator has several features that meet the user's objectives. These features are:

- **Selection of devices** - it is possible to view and select the domotic devices, organized according to the type of service, device's type, location in the house or simply view all the devices in the system;
- **Properties** - it is possible to read and write values in the devices' properties depending on the user's access level, and also set the properties to be of Normal or Action type (when changed, goes back to its initial state) and of Invalid type (the property's value is unknown);
- **Scripts** - allows the user to run multiple commands in sequence, whenever he wants or in a scheduled date and time. These commands are:
  - Set 'device' 'property' 'value' to change a value of a property of a device;
  - Wait 'number in seconds' to pause the execution of the script;
  - At 'date' 'time' or At 'time' to schedule the execution of the following commands of the script.

- **Communication Window** - allows the user to send UDP messages to other machines (SET, GET and NOTIFY) and receive them from other machines (SET, GET, ANSWER-GET, NOTIFY and ERROR);
- **NOTIFY Subscribers** - when sending the NOTIFY messages from the emulator, it will only send these messages to the machines that have subscribed certain property of a device;
- **CommunicationScripts** - offers the user to send multiple, and in sequence, UDP messages, which are:
  - SET 'DomoBusAddress' 'device' 'property' 'value' to change a value of a property of a device on the machine given by the specified DomoBusAddress, having associated the IP and port of the machine;
  - GET 'DomoBusAddress' 'device' 'property' to get the value of a property of a device in the machine given by the specified DomoBusAddress, having associated the IP and port of the machine;
  - NOTIFY 'device' 'property' 'value' to change a value of a property of a device on the machines that have subscribed that property of the device;
  - Wait 'number in seconds' to pause the execution of the communicationScript;
  - At 'date' 'time' or At 'time' to schedule the execution of the following commands of the communicationScript.
- **DevicesScripts** - allows to associate scripts to certain values of a property of the respective device. For example, if the device Switch has the value On in its On-Off property, then the emulator will run a script that will write the value On in the On-Off property of the device Lamp;
- **Supervisor** - offers the possibility of a dependency relationship between the various domotic devices, thanks to the functionality DevicesScripts (supervisor's rules), which can be done by an external application. Thus, it allows the implementation of a simple mechanism of automation for the system;
- **Simulation Time** - with this feature, it is possible to control the day and the time of the emulation chosen by the user. It is extremely important for the functionality of all kinds of scripts and verify the domotic scenarios throughout a minute, an hour or even a day;
- **AutoScript** - allows the user to assign values to properties of certain devices automatically through a single script file. For example, a case that fits this situation is the value read by a temperature sensor throughout a day, where when the value given by the At command is equal to the Simulation Time, the Set commands in the autoScript until the next At command will be executed;
- **Favourites** - there is a section in the emulator that contains the favourites rooms, services, types of devices and existing devices of each user to allow quick access to them;
- **Edit XML File** - the emulator allows to edit the XML file with the specification of the domotic system, and so it becomes flexible and easy to change the home automation system according to the needs and desires of the user;
- **Load XML File** - the emulator opens the XML file with the complete specification of the domotic system, and then it is possible to select the devices, rooms, floors, services or types of devices to emulate in order to allow an easier organization of the information in the emulator;
- **States** - it is possible for the user to save a state of the emulator, this is, store the values of the respective properties of each device in a file to work later or to save an important situation. It also allows to open files with states, this is, put the stored values from a file to the respective properties of each device in order to allow the continued use of previous emulations;
- **Workspaces** - it is possible to save the devices and the respective positions in the various workspaces, in order to save significant situations for the user. It also becomes useful when the user needs to repeat several times the same domotic scenario;
- **Alarm** - indicates the activation of certain undesirable situations, dynamically defined by the user in the emulator, due to certain values of the respective devices' properties. For example, excessive volume of a TV or lamp's brightness too low. It also shows information about these events and the relevant devices that originated this alarm;
- **Access List** - it is possible to see the number of accesses made by users in the emulator (login), and their level of access;
- **User Login** - the user must be described in the XML file of users and have a folder with his name and the XML with the specification of the domotic system to login in the emulator;
- **User's folder** - each user has a folder with his name that can be opened at any time to check the files in it;
- **Help** - in the emulator, there are plenty of tutorials and information on the various commands and features.

4.4. Code structure

In terms of the code of the emulator, it was written in Java in the program Eclipse IDE, and it is divided by 54 files and more than 18 000 lines of code.

It is composed of a main class with the graphical interface of the emulator, which appears in subsection 4.5, and each window of each menu of the bar menu of that interface has its own class. There is also a class for the Communication Window and a class that does the processing of the communication messages sent (SET, GET and NOTIFY) and received (SET,

GET, GET-ANSWER, NOTIFY and ERROR). For each feature, Scripts, AutoScript and DevicesScripts, there is a class to do the parsing of the user-written commands.

4.5. Graphical interface

The Figure 4 shows the graphical interface proposed for the domotic devices emulator.

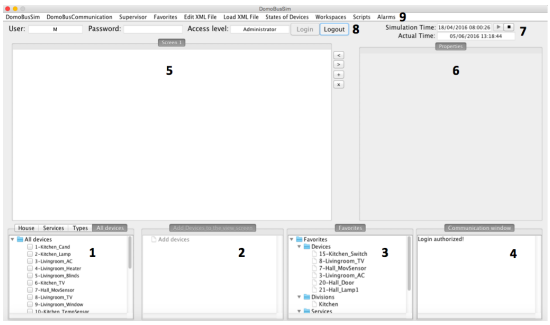

Figure 4: Graphical interface of the emulator

The numbers shown in Figure 4 have the following definitions:

1 Panels from which it is possible to add and remove existing devices to the workspaces, organized according to their characteristics in terms of their location in the house (floor and room), service, type, or just to see all of them;

2 Window where it is possible to see the devices when a room of the house, a service or a type of device is selected in the panel on the left;

3 In this panel, it is possible to view, select and remove the favourite devices, rooms, services and types of devices of the user;

4 Window that allows transmitting information about the actions performed in the emulator;

5 In these central windows of the emulator (workspaces), there are the various devices added by the user. By clicking the left mouse button on one of them, it is possible to see the properties and their values on the right panel. With the right mouse button, it is possible to add the device to favourites, define it as Inoperational (the device is defective) or view information about it;

6 In this window, it is possible to set the values that the user wants to put on the existing properties of the device selected in the workspace of the emulator. They can also set the properties of Normal or Action type (when changed, goes back to its initial state) and of Invalid type (the value of the property is unknown). For example, by clicking the device CozinhaTV in the workspace, the properties in this panel will be shown and in the mobile bar of the Volume property, the percentage value can be chosen and put in its property by clicking the OK button;

7 Field where it is possible to see the date and time defined by the user during the various domotic scenarios (Simulation Time) and the actual time, in order to get a sense of schedules and timings;

8 This part is used for the login and logout of users, specified in the XML file of the users, viewing his access level;

9 The menu bar allows viewing informations about the emulator, in particular the Website (Website Link: http://web.tecnico.ulisboa.pt/ist173703), and the wide range of functionalities described in the subsection 4.3, as well as tutorials.

5. Solution Evaluation

5.1. Characteristics of the application

The characteristics of the emulator are, briefly, exposed in Table 1.

Table 1: Main characteristics of the emulator

Characteristic	Description
Technology-based	DomoBus system
Key features offered	Scripts, Supervisor, AutoScript, Simulation Time and Communication Window
Compatibility with operative systems	Windows, Mac OS X and Linux
Execution requirements	Screen resolution higher than 1280x720
Price	Free
Size of the installation file (including XML file, images of the types of the devices and Java Runtime Environment)	Approximately 100 MB for each operative system
Target audience	Specialized technicians of domotic companies and people who have, at least, some knowledge of XML
Help support (manual, online help, ...)	Tutorials in the emulator and Website

5.2. Domotic scenarios

For a better view of the operation of the emulator, several domotic scenarios were taken as example, with an increasing level of complexity. One of them was a fairly common situation in our daily lives. In this case, the situation of a person entering the house is emulated, triggering a motion sensor and then the smart house will perform various actions, namely turning on the light in the hall, closing the window and blinds of the living room and turning on the television on the channel requested by the user.

In Figure 5, the domotic scenario described previously is showed in the domotic devices emulator.

By changing the value of the property State of the device HallDoor to 1 (Open), the supervisor will run several devicesScripts, which will change the value to 1 (On) of the property On-Off of the device HallLamp, change the value of the property State of the devices LivingroomWindow and LivingroomBlinds for 1 (Closed), put the value 1 (On) in the property On-Off of the device LivingroomTV and change the value of its property Channel to 35.

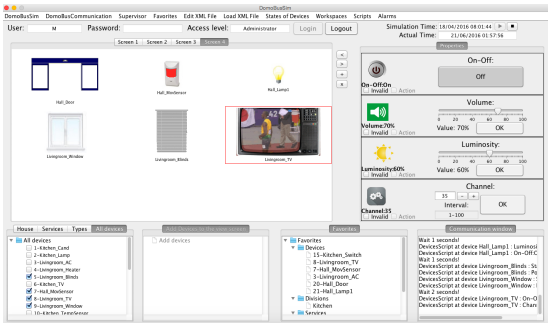

Figure 5: Domotic scenario in the emulator

5.3. Final evaluation

The results on the functionality of the emulator, in terms of the graphical part and the features of it, such as, the functionality Scripts, AutoScript, Simulation Time, Edit XML File, States, Alarms, Supervisor and UDP messages between multiple machines, were quite satisfactory.

For a more comprehensive evaluation, several experiments with multiple users were also done, in order to test the emulator created in this dissertation. There were several positive aspects to retain, including the graphic look of the emulator, especially the animations of certain types of devices and the variety of types of devices. The features Scripts, Simulation Time and Supervisor were quite important in the evaluation of the domotic scenarios and the emulator itself. A less positive point was the fact that the initial verification of the features of the emulator has proved somewhat confusing, mainly in the edition of the XML file with the domotic system specification. However, the tutorials of each menu, in the menu bar, were a very useful help.

6. Conclusions

6.1. Summary

This work has as objective present a solution for a domotic devices emulator, based on the DomoBus system, in order to be able to test behaviours for a domotic system, which can be simple or composed by numerous devices.

Although the home automation is growing and there are different kinds of domotic technologies (KNX, X10, LonWorks), there are still some problems. One of them is the fact that there are no simulators that allow the user to test any hdomotic system (simple or complex) compound by devices of any type and structure, without having them physically to analyse their functionality. It was only possible to identify control applications, such as, HomeSeer HS3, ActiveHome Pro and Control4 Composer, which allow users to control directly the devices of a domotic system. Another problem, except for the case of X10 due to its simplicity, is that they are complex and difficult to install, requiring the use of specific software tools to configure the domotic devices, such as, for example, application ETS for KNX and application LonMaker for LonWorks, having a high cost.

In the area of Internet of Things (IoT), there are some recent technologies, for an automation of the house and the daily life of the user. From these technologies, it is possible to highlight the Samsung ARTIK, Apple HomeKit and Google Brillo, wherein the first two have their own simulators (ARTIK Cloud Device Simulator and HomeKit Accessory Simulator, respectively). However, these simulators are not well developed in terms of features and types of devices.

The emulator developed in this dissertation allows the users to emulate any domotic device and so, will be possible to see the behaviour of the domotic system, through the emulation of various domotic scenarios, later adapting it to the needs and desires of the users.

It is also a generic and flexible solution, using an abstraction model of the DomoBus system and a specification language based on XML that allow describing all the relevant aspects of the domotic system in a single file. Additionally, it becomes easy to add or modify devices, or even change the entire domotic system, regardless of its size.

The graphical application of the emulator facilitates the visualization and management of the devices, by the user, suggestive images and animations that reflect directly their current state. Thus, the user is easily-aware of the states of the devices and, by clicking on its visual representation, can access its properties and modify their values.

The emulator has several essential features to help the user to emulate various domotic scenarios, such as, Scripts, Supervision, AutoScript, States, Workspaces, Time control and Communication.

Thus, this emulator is a very useful tool for viewing and testing complex domotic scenarios, composed by many devices of different types and functions. To confirm this statement, multiple tests were successfully carried out, using a wide variety of domotic scenarios with different combinations of devices. In addition, it was achieved all the objectives proposed in this dissertation.

In conclusion, with the emulator developed in this dissertation, it is expected that any user can emulate a domotic system with the desired devices, without purchasing them previously, adapting them more effectively to his needs and allowing to take the most out of the system for a better experience in his house.

6.2. Future work

The emulator proposed in this work may still have some improvements, confirmed in several experiments with different domotic scenarios, namely:

- Creating an adequate and functional version for mobile devices;
- Addition to the emulator of more types of devices, specifying their information, images and/or animations;

- Introduction of security mechanisms in the messages exchanged between the various applications.

Finally, it is hoped that this dissertation and the use of this emulator by different users can make a contribution, even if modest, to the growth of home automation and smart houses in the actual society.

Acknowledgements

First, I appreciate all the support that was given to me by my advisor, Professor Renato Jorge Caleira Nunes, during the implementation of this assignment and the precious help in the many meetings we had.

I am also grateful to my university, Instituto Superior Técnico - University of Lisbon, which contributed to my growth, level of knowledge and achieving academic and personal results.

Thank You to all colleagues and friends who have helped me along my university studies. In particular, I thank António Morgado, Francisco Santos, Gonçalo Glória, Imran Mamade, Jaime Antunes, João Brito, João Graça, João Silva, Jorge Matos, Luís Marques, Nuno Bajanca, Pedro Antunes, Ricardo Ferro and Tiago Viegas.

Finally, I thank my parents, Ana Paula Martinho and Fernando Martinho, and my sister, Sofia Martinho, who have encouraged me in my academic life and to be able to achieve my personal goals.

References

[1] "The Internet of Things (IoT)," October 2015. [Online]. Available: http://www.theinternetofthings.eu/what-is-the-internet-of-things

[2] R. J. C. Nunes, "Domobus – A New Approach to Home Automation," in *8CLEEE – 8th International Congress on Electrical Engineering*, Portugal, July 2003.

[3] "The OSI Model's Seven Layers Defined and Functions Explained," November 2015. [Online]. Available: https://support.microsoft.com/en-us/kb/103884

[4] "Communication Media of KNX," November 2015. [Online]. Available: http://www.knx.org/knx-en/knx/technology/communication-media/index.php

[5] "Software tool ETS," November 2015. [Online]. Available: http://www.knx.org/no/software/ets/about/index.php?navid=948232948232

[6] "Software tool LonMaker," December 2015. [Online]. Available: http://www.infranet-partners.co.uk/downloads/products/man/131025E%20LONMAKER.PDF

[7] "Control aplication HomeSeer HS3," November 2015. [Online]. Available: http://www.homeseer.com/home-control-software.html

[8] "Control aplicationActiveHome Pro," November 2015. [Online]. Available: http://kbase.x10.com/wiki/ActiveHome_Pro

[9] "Control aplication Control4 Composer," December 2015. [Online]. Available: http://www.control4.com/

[10] "Samsung ARTIK Modules," July 2016. [Online]. Available: https://www.artik.io/modules/

[11] "Samsung ARTIK Cloud," July 2016. [Online]. Available: https://artik.cloud/

[12] "ARTIK Cloud Device Simulator," July 2016. [Online]. Available: https://developer.artik.cloud/documentation/tools/device-simulator.html

[13] "HomeKit Accessory Simulator," July 2016. [Online]. Available: https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/HomeKitDeveloperGuide/TestingYourHomeKitApp/TestingYourHomeKitApp.html

[14] R. J. C. Nunes, "DomoBus Architecture," Instituto Superior Técnico/INESC-ID, Tech. Rep., February 2016.

[15] "What is Arduino?" November 2015. [Online]. Available: https://www.arduino.cc/en/Guide/Introduction

[16] "About Raspberry Pi," December 2015. [Online]. Available: https://www.raspberrypi.org/about/

[17] "Home automation," October 2015. [Online]. Available: https://en.wikipedia.org/wiki/Home_automation

[18] "EIB Protocol," November 2015. [Online]. Available: https://pt.wikipedia.org/wiki/EIB

[19] "Introduction to KNX," November 2015. [Online]. Available: https://www.knx.org/knx-en/knx/association/introduction/index.php

[20] "What is KNX?" November 2015. [Online]. Available: https://www.knx.org/knx-en/knx/association/what-is-knx/index.php

[21] "LonWorks Protocol or LON (Local Operating Network)," December 2015. [Online]. Available: www.prof2000.pt/users/lpa/lonworks.ppt

[22] "About X10," November 2015. [Online]. Available: https://www.x10.com/about-us/

[23] "X10 Basics," November 2015. [Online]. Available: http://www.x10.com/x10-basics.html

[24] R. J. C. Nunes, "Análise comparativa de tecnologias para domótica," in *JEACI 2002- III Jornadas de Engenharia de Automação, Controlo e Instrumentação*, Setúbal, Portugal, May 2002.

[25] J. MSV, "Google Brillo vs. Apple HomeKit: The Battleground Shifts To IoT," *Forbes*, October 2015.

[26] G. Marshall, "Brillo vs HomeKit vs the rest: Who will win the battle for your home?" *Wareable*, August 2015.

[27] "Samsung ARTIK," July 2016. [Online]. Available: https://www.artik.io

[28] "Google Brillo," July 2016. [Online]. Available: https://developers.google.com/brillo/

[29] "Google Weave," July 2016. [Online]. Available: https://developers.google.com/weave/

[30] "Apple HomeKit," July 2016. [Online]. Available: https://developer.apple.com/homekit/

[31] "DomoBus system," October 2015. [Online]. Available: http://www.domobus.net/

[32] R. J. C. Nunes, "Modelo de especificação e programação de um sistema domótico," in *IADIS Conferência Ibero-Americana WWW/Internet 2004*, Madrid, Spain, October 2004.

[33] ——, "The DomoBus System Specification Language," Instituto Superior Técnico/INESC-ID, Tech. Rep., February 2016.

[34] ——, "Decentralized Supervision for Home Automation," in *MELECON 2006 - The 13th IEEE Mediterranean Electrotechnical Conference*, Benalmádena, Spain, May 2006.

[35] ——, "A Communication Infrastructure for Home Automation," in *CCCT 2004 – International Conference on Computer, Communications and Control Technologies*, Austin, USA, August 2004.

[36] ——, "Implementing Tiny Embedded Systems with Networking Capabilities," in *IADIS International Conference on Applied Computing 2005*, Algarve, Portugal, February 2005.

[37] ——, "Domobus - the user in control," in *A Domótica e a Casa do Futuro, AveiroDomus - Workshop Domótica*, Aveiro, Portugal, March 2006.