Abstract — Home automation has brought many benefits to our interaction with homes, improving comfort, safety and allowing greater energy efficiency.

These and other advantages increase the demand for home automation systems.

But the fact that current systems are, or easy to install and configure, but not very powerful, or quite powerful but difficult to set up, creates problems in choosing the most suitable system.

So it becomes necessary to develop a system that can be powerful, accomplishing the most demanding tasks and, at the same time, be easy to configure and program by any user.

This dissertation aims to develop a home automation management software, which is based on the DomoBus system. This software will allow testing the functionality of the DomoBus system and its possible application in real environments.

Three usage scenarios were defined and the operation of our software was tested. We concluded that the system is very flexible and powerful, and it offers all the basic tools to manage most usage scenarios.

The software also showed it can be operated by an unskilled person, as it is easy to configure and operate.

However some limitations were identified regarding, in particular, the difficulty to configure complex scenarios. These scenarios are fully supported by the system but require more effort from the user to configure them. However, we believe future versions of the software may address these issues by offering, for example, predefined configuration patterns.

Index Terms — Domotics, Home Automation, Supervision, Configuration, DomoBus.

I. INTRODUCTION

The standard home automation systems currently on the market (KNX, X10, LonWorks, etc ...) are expensive and, often, they are difficult to install and configure. The X10 system is an exception because it can be installed by the average person. However, its easier installation is compromised by its functional limitations.

The aim of this work is to develop a system that can be configured by the end user, without the need of specific knowledge or expertise, beyond the simple operation of a computer. Nevertheless that system should offer a very dynamic, modular and robust solution to home automation.

The DomoBus system is being developed providing a generic model for home automation devices, which is independent of any technology. Because of this, the DomoBus system can function as an integration platform for different technologies. Thus, there is interest in developing supervision mechanisms for DomoBus, for it to become a powerful platform that allows the definition and management of any desired behavior for the home.

Given the interoperability characteristics of DomoBus, the supervision mechanisms can then be used to manage heterogeneous systems that use different technologies.

These mechanisms should be as generic and flexible as possible. We want to use an approach easy to understand by anyone, thus allowing a typical user to define or change how their house performs, using a proper application.

At the same time, we intend to follow a model in which the definition and modification of the rules of operation of the system may occur at any time without the need to stop the system for reconfiguration. Thus the user can change the system whenever desired, facilitating adaptation to their preferences or new needs.

II. DOMOTICS

DomoBus is an Home automation concept that seeks to find solutions that respond to the need of man to expend the least effort in the daily routine and activities. So, home automation, besides introducing comfort and improvement of life for its users, also introduces new concepts such as communication and security.

Home automation uses and combines the advantages of electronics and computers, to obtain a usable and integrated management of various equipment of a building, residential or commercial.

The variety of standard protocols targeted for home automation is great. In the market, the most common systems use the X10 protocol to be cheapest, but it has the great disadvantage of being rather robust and has many limitations.

Systems using this protocol are easily found either by using the electricity grid as a means of communication, (as shown in Fig.1) either by not requiring experienced installers.

These systems are only advised when requires a simple and undemanding control.

![Fig.1. Connection example using X10 modules.](image-url)
A standard protocol widely used in home systems is the KNX. This protocol provides a lot of flexibility and robustness. However, their products have a high cost.

The KNX protocol allows the use of multiple physical media, but the most used is twisted pair where all devices are connected to a bus, as shown in Fig. 2.

There is also the Lonworks protocol that despite being designed to cover all the requirements of most control applications has only been successful in its implementation in office buildings, hotels and industries. Due to its cost, the LonWorks devices have not had large deployment in homes, especially because there are other technologies with equal benefits but cheaper.

A binding model of the LonWorks protocol is represented in Fig. 3.

Table 1 presents a comparison between the three protocols (X10, KNX and Lonworks).

### III. DOMOBus

#### A. Super-Automated Homes

The concept of super-automated homes [8] comes up with the evolution of the technology presented in our homes today.

In modern homes, we have in a division, various switches for multiple lamps, TV, DVD player, electric shutters, air conditioning, motion sensors, temperature sensors and many more devices. When we want to control all these devices in an automated way, we realize that it is an enormous challenge. It then appears the concept of super-automated homes.

Is Fig. 4 there is an example of a division that is intended to automate and in Table 2 the devices contained in it.
TABLE II
DESCRIPTION OF THE DEVICES OF FIGURE 6

<table>
<thead>
<tr>
<th>#</th>
<th>DISPOSITIVO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Switch 1</td>
</tr>
<tr>
<td>2</td>
<td>Switch 2</td>
</tr>
<tr>
<td>3</td>
<td>Switch 3</td>
</tr>
<tr>
<td>4</td>
<td>Lamp 1</td>
</tr>
<tr>
<td>5</td>
<td>Lamp 2</td>
</tr>
<tr>
<td>6</td>
<td>Motion Detector</td>
</tr>
<tr>
<td>7</td>
<td>Blinds Button 1</td>
</tr>
<tr>
<td>8</td>
<td>Blinds Button 2</td>
</tr>
<tr>
<td>9</td>
<td>Blind 1</td>
</tr>
<tr>
<td>10</td>
<td>Blind 2</td>
</tr>
<tr>
<td>11</td>
<td>Thermostat</td>
</tr>
<tr>
<td>12</td>
<td>Air Conditioning</td>
</tr>
<tr>
<td>13</td>
<td>TV</td>
</tr>
</tbody>
</table>

Viewing this example we can imagine the complexity that will be to automate this division, so that it can perform complex tasks, or even remote control. We are talking about functions, which can be as simple as putting the switch 1 to turn on or off the lamp 1. We can also make the switch 2 and switch 3 work together to turn on and off the lamp 2 in a complementary manner, i.e. when one lights up then the other goes off, or when one goes off, then the other lights up.

We can imagine many different scenarios, which would be possible if this division has an automated system.

B. DomoBus Architecture

The DomoBus system addresses the automation at two distinct levels. A layer of low-level associated to the reading of sensors and control of actuators, and a high-level layer consisting in the management and supervision of the home automation system.

The DomoBus system is organized as shown in Figure 5, in the top we present the supervision module, it’s he who manage the control modules, to coordinate the commands sent to and received from physical devices (sensors and actuators) [8].

Although the figure illustrates only one supervision module, a system with a significant size may have multiple modules interconnected by a TCP/IP network.

The control modules are the bridge between the supervision module and the physical devices.

These modules are not yet completed. There is some prototypes under development.

Note that the supervision modules can be connected to devices from other technologies (X10, KNX, etc.) through appropriate adapters.

Fig.5. Representation of DomoBus system.

The level of management and supervision of the system, which defines the location and configuration of devices, follows a general approach independent of the technology used. A key element of this approach is the model that is followed for the home automation devices [6]. In this model, a home automation device is a generic entity, described by a set of properties and there is a basic set of operations to view or change these properties.

The DomoBus solution includes a specification language based on XML, as seen in [11], which allows describing the constitution of any system, explaining what devices exist and what are their characteristics. This language also allows describing the structure of any house and identifying the location of each device. It also allows the definition of scenarios that correspond to a list of actions to be performed on different devices.

In the future the language will be expanded to allow specifying the behavior and the desired settings for the system (which is directly related to this work).

The component of high level DomoBus system, intended to be as generic and flexible as possible and can be used to monitor parts of a system using another technology, thus supporting the integration of heterogeneous systems, allowing to explore what each specific technology may have best in each particular operating situation.

C. Supervision Module

It is the supervision module that controls all the home automation system. It’s him that runs the control software created in this work. This software consists of a library of functions in C language, which allows to manage in real time any given division, a house, an office or even an entire building. The choice of the C language aimed to facilitate
portability to different hardware platforms even with small memory capacity and processing. These functions allow the creation of programs that interact with the various devices in the system.

A system may have several supervision modules, thus creating a decentralized system in which different modules can be located and controlling different parts of the residence, but interacting with each other. In the context of the present work was tested a system with only one supervision module.

With respect to processing capability the supervision module does not need great processing power and may be comprised of a "single board computer" (SBC). We Took up the option to use the model Raspberry Pi [2, 23], which has a cost of about 30 euros and offers Linux operating system and ability to TCP/IP connection.

**D. Devices and Properties**

The DomoBus system allows generically representation of each device, this representation is simple but yet powerful enough to describe the essential capabilities of a device, regardless of their technology.

Thus DomoBus system proposes a solution (detailed in [6]) in which each device is characterized by a set of properties and every property has a value. For instance a thermostat can only have one property, temperature, which contains the current value of the temperature of the room measured by the device.

In the case of a lamp, it can have two properties: On/Off and brightness.

<table>
<thead>
<tr>
<th>Device Type</th>
<th>Property</th>
<th>Scalar</th>
<th>Enumerated</th>
<th>Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name</td>
<td>Value</td>
<td>Value</td>
<td>Dimension</td>
</tr>
<tr>
<td>Access Mode</td>
<td>Value</td>
<td></td>
<td></td>
<td>Value</td>
</tr>
</tbody>
</table>

**Fig.6. Model of devices of the DomoBus system.**

The Device Type abstract devices of a given type, such as lamps of adjustable intensity, temperature sensors, air conditioning, common switches, power outlets, shutter motors, solenoids, smoke detectors, motion detectors, Hi-fi, TV, DVD, etc.. It should be noted that it is possible at any time to add new types of devices, since it is possible to describe its characteristics with properties.

The characterization of a Device Type is performed using one or more property. The same property can be used by any number of device type.

In Fig. 6 are shown the key attributes of the Properties class. The name attribute contains the description of property. For each type of property you can define an access mode that can take on the values, read-only, write-only or read-write.

All interactions with the physical devices only take place reading or writing values on properties.

Another attribute of property is Value Type, which distinguishes three types of values: Scalar, Enumerated and Vector.

The scalar type identifies an integer that can range between a minimum value and a maximum value.

The Enumerated type represents a property that can have only a small number of values, for example, the On/Off property can only take the values 0 (off) and 1 (on).

The type Vector is used on properties that contain a list of characters (representing text or other information), for example, an alarm code.

Finally we have the devices, which are created using the most appropriate device type. Thus, being associated with the properties of this device type.

The model presented is therefore very flexible and effective.

**IV. SUPERVISION MECHANISM**

The supervision mechanism uses the concept of program, which basically consists in the use of clauses such as:

\[
\text{IF condition THEN actions-list ELSE actions-list}
\]

The condition can be an expression as complex as desired, which may involve the time or the value of any property of a device. The property value of a device is evaluated and if the condition is true then executes the activation actions list. The actions consist in assigning a value to a property of a device.

This way of specifying the behavior of a house is simple but effective.

Beyond what was defined, scenarios may also have a list of actions of deactivation. This list contains actions that are performed when an activation condition of a scenario no longer applies. I.e. there is a list of actions for activation, and a list of actions for deactivation, when the condition becomes true, the activation actions will be executed when the condition becomes false, deactivation actions are performed.

These programs are of various types, there are simple programs, complex programs, time schedules and also the use of sequences.

Whenever programming requires only one test of a property device, it’s called simple program.

The possible conditions are shown in Table 3.
When the program requires interaction of various conditions is designated complex program. Complex programs use various conditions that are evaluated using the operators presented in table 4.

The mode of operation is similar to that of simple programs, i.e. when the general condition becomes true, the actions are executed.

**TABLE III**

**POSSIBLE CONDITIONS FOR SIMPLE PROGRAMS**

<table>
<thead>
<tr>
<th>Sign</th>
<th>CONDITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equal to</td>
</tr>
<tr>
<td>≠</td>
<td>Different from</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>≤</td>
<td>Less or Equal to</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>≥</td>
<td>Greater or Equal to</td>
</tr>
<tr>
<td>DO</td>
<td>Always Take Action</td>
</tr>
</tbody>
</table>

The schedules, uses the information given by the clock to enable or disable certain actions.

The condition here is represented by a time: year, month, day, hours, minutes and seconds. When the time is reached, the actions are triggered.

The schedules are classified according to their level of repetition:

**Single Event** - Takes place only once.

**Diary** - Takes place daily

**Weekly** - Takes place on the same day (of the week) every week.

**Monthly, Day of the Month** - Takes place on the same day every month.

**Monthly, Weekday** - Takes place every month on the same day of the week (for example all first 5 Mondays of each month).

**Annual** - Takes place every year in the same month, day, hours, minutes and seconds.

Finally the sequences are a program format that uses pauses given in seconds, at the end of which activates the actions defined. The sequences are activated by other programs.

**TABLE IV**

**POSSIBLE CONDITIONS FOR SIMPLE PROGRAMS**

<table>
<thead>
<tr>
<th>Sign</th>
<th>OPERATOR</th>
<th>CONDITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>A == B</td>
<td>EQUAL</td>
<td>= B</td>
</tr>
<tr>
<td>! A</td>
<td>NOT</td>
<td>≠ A</td>
</tr>
<tr>
<td>A &amp;&amp; B</td>
<td>AND</td>
<td>&lt; B</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>A ⊕ B</td>
<td>XOR</td>
<td>&gt; B</td>
</tr>
<tr>
<td>A ⊕ B</td>
<td>XNOR</td>
<td>≥ B</td>
</tr>
</tbody>
</table>

**A. Supervision Application**

The supervision application, was the software created in the context of this work, in order to test the proper operation of the DomoBus library functions.

This software consisted of a C language program that provides the necessary tools to be able to interact with the DomoBus, both in terms of creating the necessary programs to its operation, and in terms of adding new devices and properties.

The supervision application also allows the passage of DomoBus commands to the devices.

The software is divided into two applications, Programador and Supervisor, which interact with each other, as seen in Fig.7. Each will be installed on a different computer communicating through TCP/IP network.

The Supervisor is the brain of the system, because it contains the DomoBus function library, and it is he who makes all the decisions and does all the processing.

**B. Programming Application**

It is with this application that the user interacts in order to parameterize the entire system.

It is through it that we can add new properties, device types or devices.

We can create new simple or complex programs, schedules or sequences. We can add actions, both activation and deactivation to each of the programs.

We can add new conditions or operators for complex programs. And also new periods to schedules, as well as add new items to sequences.

The programming application also allows you to list on screen all devices, device types and properties. And also all
programs, with various conditions, operators, time periods, sequences of items, as well as all the actions associated with these programs.

Finally, it also allows you to delete simple and complex programs (in future it is expected that you can also remove schedules and sequences).

V. IMPLEMENTATION

The software developed in this work consists of two applications. The supervision application, called Supervisor and the programming application called Programador.

The Programador will be used to configure all the functions required for automation, whether the creation of programs, whether the addition of new devices and properties to the system.

The Supervisor will be a kind of brain of the software, and it will run all orders relating to the management of the different devices in the system. It is he who receives status information from devices and triggers the sending of commands to perform actions. It is also he who makes time management.

In this first version of the supervision mechanism, communication with the devices is not fully implemented, the transfer of orders between the Supervisor and the devices does not exist. The orders are generated only in a generic format to test the proper operation of the Supervisor.

A. Aplication Programador

The programming application connects to the Supervisor via the TCP/IP protocol, being enough knowing the IP address of the Supervisor. This allows the user to connect to the system and configure it from any computer where the application Programador is installed.

The Programador sends a predetermined data package to the Supervisor and stays awaiting a response, confirming the successful reception of the data. This data packet, which corresponds to a data structure which can be seen in Fig.8, has 10 variables of type integer, a string and a vector of integers.

B. Application Supervisor

It is by the Supervisor who pass all commands, it’s he who makes all the processing and makes all the decisions and it’s he who will communicate with all the devices.

There are three actions to be executed in competition on the server, using threads. The structure of the server is represented in Fig. 9.
The first thread handles time. This thread informs the library of the current time, in seconds, once every second.

Another thread manages the keyboard, identifies the commands introduced and the commands that terminate the program. When the program becomes fully functional, this thread is no longer needed by simply detecting the command to terminate the program. This is because every user interaction with the system is planned to be done by the application Programador.

Finally, the last of the threads is the one that makes the management of users and devices. This thread is continuously listening to accept the connection from the application Programador, or any other device in the future. When a new message arrives the Supervisor read first the variable tipo and compares its contents with the list of commands.

If the command is of the type addition, adding new properties, devices or programs, the program will read the data from the remaining variables and then call the appropriate library function to execute the command.

Another action that the program does when it receives one of these commands is to add the already processed data to the text files.

Supervisor application uses two text files. These text files serve as backup for all operations. The need to provide a backup of this type lies in the fact that if there is the need to restart the system, or if he is terminated abruptly by a power outage, when it start again, the system has all the settings made previously which will be inserted automatically.

The two text files mentioned have the names listas.txt and programas.txt.

The listas.txt file holds all the information regarding the types of devices, devices and properties.

The programas.txt file stores information concerning programs. All programs made by the user and sent through the Programador are saved here.

For other types of commands, which are mostly requisitions for information or data validation, the Supervisor after reviewing what command to run and read the remaining data will process the information and send the response back to the Programador.

VI. APPLICATION EXAMPLES

To test the operation of our system and using the example of Fig. 4, three different scenarios were created, to illustrate three types of programming (one very simple, one moderate and a complex example).

The simplest example is a switch connected to a lamp to turn it on and off. The scenario is very simple to understand and was very simple to create.

The second example involves two switches working in a complementary manner to turn on and off a lamp. When one turns on, then the other turns off, and when one turns off the second turns on. This example was successful although it revealed to be more complex to implement.

The third example used a time schedule programming and sequences, so that during the night, the motion detector can turns on and off the lighting of the room, after a certain time without any people present in the division. This example was also successful.

With these three examples, we see that the system is quite intuitive and easy to configure for simple situations. But it can become difficult to configure, requiring some training, in more complex situations.

VII. CONCLUSION

This work aimed to create and test a software application that allows supervising of a home automation system. The supervision application developed uses the DomoBus system, taking advantage of its generic model for domotic devices, which allows the development of very flexible applications independent of any specific technology. This generic model allows us to describe a physical device through a set of properties, where each property will assume a value depending on its current state.

The developed application, called Supervisor, was programmed in C language for performance and portability reasons.

To interact with the supervision application, there is the application Programador. The Programador runs on another computer connected by a TCP/IP network to the Supervisor. The user uses the Programador to configure the entire system and define how he wants the system to behave. The user can modify the system, adding new devices or conditions, or amend existing, whenever he wants, without having to stop the system to update it.

It was verified that the developed mechanisms are powerful and robust, and at the same time, very flexible because is possible to change the system settings whenever desired and can be integrated with devices of various technologies.

However, although it can be configured by a typical user without expertise, the configuration process can become difficult when complex scenarios are involved.

Nevertheless, the developed solution offers benefits when compared with available commercial systems, as it is more flexible and powerful.

REFERENCES


