AutoDev- A system to simplify configuration processes of IoT devices for domestic users

Bernardo Rodrigues
bernardo.m.rodrigues@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

January 2018

Abstract

Many of the commercially available IoT devices for domestic users rely on complex configuration processes to configure them. Systems capable of reducing the complexity of their processes and simplifying them, are necessary for this devices are purchased and used by the users. In this scope, this paper proposes a system that addresses this problem. This system simplifies these processes for domestic users and in some cases automate them. The proposed system offers a software architecture installed on the device itself. Also offers a secure and simple alternative to configure the devices. The goal is to reduce the complexity of configuration processes without affecting user comfort. This paper analyzes the real state of the art in the area of IoT by analyzing its constituents. Additionally, it examines the existing IoT devices for domestic users. Our proposal is shown in detail and evaluated.

Keywords: IoT, Smart Objects, Configuration Processes, Configuration of devices, Domestic users

1. Introduction

The basic idea of Internet of Things (IoT) concept is the pervasive presence around us of a variety of things or objects such as Radio-Frequency Identification (RFID) tags, sensors, actuators, mobile phones, etc. These things, through unique addressing schemes, are able to interact with each other and cooperate with their neighbors to reach common goals [1]. The concept of IoT is a radical evolution of the current Internet, into a Network of interconnected objects that not only harvests information from the environment (sensing) but also interacts with the physical world (actuation/command/control). This concept uses existing Internet standards to provide services for information transfer, analytics, applications and communications to somehow build data patterns by analyzing transferred data. To know when it is necessary to interact in the physical world. These objects, normally are called by Smart Objects. Smart Objects collect the data in the place and send this data to others Smart Objects that will actuate in the environment where both are installed. IoT has many industrial applications, agriculture and healthcare [2, 3, 4]. Furthermore, there are several devices available on the market for domestic users to use in their homes. However, the configuration processes of these devices are extremely complex, which means that in most of cases these devices will not be purchased by these users. Thus, this paper will focus on IoT devices for domestic users. These type of users, are the ones with most difficult to configure IoT devices because they are not experts users in this matter. Besides that, in an industrial scenario, the configuration of the devices is made by highly qualified staff (expert users) and in general, the configuration processes turn out to be quite similar.

The configuration processes of IoT devices available to domestic users are too complex. It will eventually lead in a very long task in terms of time which ends up not appealing to many users. This happens because current devices are not flexible in their configuration. A different interface must be used for each device, which makes the situation even more confusing. Additionally, most of the existing configuration processes have unfriendly user interfaces and do not provide much user support. Furthermore, current devices that are already configured by the user are unable to help configure new devices. This causes the user to have to use several different interfaces to control and configure each device. These devices have limited customization since only the user who has configure the device can access, control and customize it.

This paper presents a solution for the lack of a system to configure IoT devices in a simple and less complex way. We designed a system capable of reducing these configuration processes and that in some cases is able to fully automate these pro-
cesses. In addition, our system ensures that devices can be controlled not only by users who have configured them. This system was developed for devices which communicate through protocol Internet Protocol (IP) (IEEE 802.11 (Wi-Fi) or Ethernet devices). When it is not possible to communicate through this protocol Near Field Communication (NFC) can be used. This solution contains two distinct modes. The local configuration mode, where the user configures the device through an interface. This interface will allow the user to enter all the parameters necessary for the configuration of the device in a simple way.

The automatic configuration mode, in which a device that has already been configured replicates its configuration to another device that is not configured, automatically. This mode will not require an interface since it is fully automatic. We will also take into account the protection requirements of the configuration process itself, that can be eavesdropped by an attacker in order to steal sensitive information about the device or the network. Unfortunately, security is often overlooked in order to simplify configuration to the users, being frequently relegated to a secondary role. Thus, our system will be able to protect all the information that is generated by it to configure a device.

2. Related Work
2.1. IoT & Smart Objects
Although there are many definitions, nowadays a commonly accepted one for this technology is "a dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols; physical and virtual ‘Things’ in an IoT have identities and attributes which are capable of using intelligent interfaces to being integrated as an information network" [5].

In other words, it is a network of uniquely addressable interconnected objects, based on standard communication protocols. These smart objects collect any useful information about the physical world to use it in various applications.

A smart object is an embedded system with small micro-electronic device that consists of a communication module, typically a low-power radio, a small microprocessor, a power source like a small battery, a sensor or an actuator.

The sensor gives the smart object the ability to sense the physical world, by for example measuring its air qualities. Actuators make it possible for the smart objects to change the physical world, by taking an action.

The communication device enables the smart object to communicate its sensor readings to the outside world. A smart object receives input from other smart objects, i.e., if there is more than one of this devices than it is possible to form a network of smart objects called sensor network [6]. The emerging wireless sensor technologies have significantly extended the sensory capabilities of devices. Therefore, the original concept of IoT is extending to ambient intelligence and autonomous control. To date, a number of technologies are involved in IoT, e.g., sensor networks, NFC, low energy wireless communications, RFID, cloud computing and ubiquitous computing [5].

2.2. Types of IoT devices
Nowadays, there are many IoT devices for domestic users to improve their lives either in their homes or outside of them.

We present these devices by grouping them into the following categories: Home Automation, Peripherals, Wearable/Sports Gear and Home Security. Home Automation category encompasses products for houses such as household appliances. Currently available devices, such as, Smart washing machines which can be controlled when the user is away from home. Or for example, Multi Room Speakers which the user gain the ability to listen music in several rooms inside the same house. Peripherals category, are indispensable in IoT for the user, i.e., mouse, keyboards, printers, etc. Wearable/Sports Gear devices are the most recent category. Wearable technology (also called wearable gadgets) is a category of technology devices that can be worn by a consumer and often include tracking information related to health and fitness. In the Home Security category are included devices such as Smart Intrusion Detection Systems, Smart Alarm System, both of which have the responsibility to monitor the environment where they are installed and notify the owner when there’s a notice.

2.3. What to configure
We have very IoT devices available in the market for domestic users. All of the devices have parameters to configure and we have split these parameters into three categories. The Connectivity parameters refer to the communication supported by the device which must be configured. The Security parameters refer to who is having access to the device or possible access accounts needed such as a cloud account or an account manufacturer. The Functionality parameters refers to the purpose of the device. For example, if it is a surveillance system then the image quality it will be a functionality parameter to configure.

2.4. Configuration Profile
In order to reduce the complexity of the configuration processes of the IoT devices, we need to know specifically what need to be configured in the device. A Configuration Profile (CP) is a file that
an administrator can use to customize settings on a domestic user’s device. CP allow administrators to enforce security measures, such as mobile device passcode requirements, and restrict user access to certain device features, such as the camera for example. CP also provide administrators with control over network connectivity settings. Administrators can create a configuration profile manually and will be delivered to a specific device as an email attachment, as a website download, as an over-the-air update or with a USB connection. We can write these profiles using two programming languages. The eXtensible Markup Language (XML) language is one of them. Such as the case with Apple¹ on their mobile devices. But in terms of IoT devices, an XML file seems to introduce some challenges, i.e, the fact of it will not be possible to require specific root element or the fact of that XML it is a language not self-described. This means that, the ’i’, ’p’, and ’”’ characters distinguish the information being described from the ”markup” that is part of its description. This syntax allows more flexibility in the encoding of the text stream. Another alternative is to use Yet Another Markup Language (YAML). YAML is a human-readable data serialization language. It is commonly used for configuration files, but could be used in many applications where data is being stored or transmitted. Custom data types are allowed, but YAML natively encodes scalars (such as strings, integers, and floats), lists, and associative arrays (also known as hashes or dictionaries). Compared with XML, YAML format simply matches the data structures of dynamic languages better. In YAML, there is no extra delimiter. So it is the lightweight then XML and not using delimiter also makes the reading light and simple. YAML makes the data understanding easy. So it is useful in case of configuration. The main disadvantages of using YAML relies on the fact that this language does not incorporate a syntax validation.

2.5. Security
Since there are many security and privacy issues with IoT devices we are describing, in this section, security terms and protocols that can be applied to an environment which contains these type of devices. Confidentiality of information, means that the information is protected from disclosure to unauthorized parties [7]. A very key component of protecting information confidentiality would be encryption. Encryption ensures that only the right people, i.e., people who has the key can read the information. Transport Layer Security (TLS) is a security protocol for communications over the Internet that has been used in conjunction for example with Hypertext Transfer Protocol (HTTP) as know as Hypertext Transfer Protocol Secure (HTTPS). Other ways to ensure information confidentiality include enforcing file permissions and access control list to restrict access to sensitive information.

Integrity involves maintaining the consistency, accuracy, and trustworthiness of data over its entire life cycle. Data must not be changed between the origin to the destination. We must ensure that data cannot be altered by unauthorized people [7]. Commonly used methods to protect data integrity includes hashing the data the user received and comparing it with the hash of the original message. However, this means that the hash of the original data must be provided to the user in a secure way. Over the internet, it is used HTTPS. HTTPS is a communications protocol for secure communication over a computer network which is widely used on the Internet. The main motivation for HTTPS is authentication of the visited website and protection of the privacy and integrity of the exchanged data. HTTPS provides authentication of the website and associated web server with which one is communicating. Thus, it protects against man-in-the-middle attacks. Additionally, it provides bidirectional encryption of communications between a client and server, which protects against eavesdropping and tampering with or forging the contents of the communication.

Availability of information refers to ensuring that authorized parties are able to access the information when needed [7]. Information only has value if the right people can access it at the right times. Denying access to information has become a very common attack nowadays. Almost every week we can find news about high profile websites being taken down by Distributed Denial of Service (DDoS) attacks. The primary aim of DDoS attacks is to deny users of the website access to the resources of the website.

The social acceptance of the new IoT technologies and services will strongly rely on the trustworthiness of information and protection of private data. Technically, as more and more objects become traceable through IoT, threats to personal privacy become more serious. In addition to securing data to make sure that it doesn’t fall into the wrong hands, issues of data ownership need to be addressed in order to ensure that users feel comfortable participating in the IoT.

3. Architecture
3.1. System Requirements
Requirements need to be defined in order to choose the appropriate architecture in terms of its performance, functionality, extensibility and comfort. Being this system appropriate to be in the home of the users we need to ensure that this system is universal so that it works in several houses in an inde-

¹https://www.apple.com/
dependent manner. Next, this system must support and adapt to the different IoT devices. More important, is not to forget that the user is our main target. To decrease the complexity of the configuration processes the system must be intuitive to use and user-friendly. Which will lead us to a system that ensures that the configuration processes are faster and simpler than the existing ones. Finally, this system must protect the information exchanged in this system. Also, protect messages eavesdropping and tampering.

3.2. AutoDev

Automatic Device (AutoDev) simplifies the configuration processes of IoT devices for domestic user which communicates through IP protocol and when it is not possible to communicate over this protocol may use NFC. There are two types of devices in this system. The configurator which is the device that is going to perform the configuration process of a device. The non-configured device which is the IoT device that needs to be configured is the IoT device that needs to be configured. These two devices will communicate in a secure channel to fulfill the security requirement as already described. The communication channels will be Wi-Fi, Ethernet or NFC. There are two possible modes in AutoDev system. The Local Configuration Mode in which the configurator device is a smartphone providing a Graphical User Interface (GUI). This GUI is an Android mobile application. And the Automatic Configuration Mode in which which the configurator device is a device already configured at user’s home. Thus, replicates its configuration to a the non-configured device. In order to achieve the flexibility requirement, our system must adapt to the different existing IoT devices. For that, we need a manufacturer’s description of what need to be configured on the device. Thus, we assume there is a configuration profile inside the IoT device with a normalized structure using YAML language.

3.3. Software Architecture

With this in mind, the software will follow a layered architecture which will accommodate the use of multi devices reducing its configuration. We propose the presented architecture in Figure 1, which was designed to run on common types of devices with the different protocol communication types, such as Wi-Fi, Ethernet or NFC.

The communication Layer includes a set of libraries that abstracts the communication protocols from the rest of the software. The Service layer is composed by a set of modules, each one responsible for a distinct task. The Service Discovery module is responsible for discovering devices within the same network. In this case we are choosing to use Apple Bonjour in order to the devices discover each other since is the most stable service discovery protocol to use. Let’s suppose that there is a Wi-Fi device already connected and configured at the user’s home. Also, there is an ethernet device that is not configured. Thus, automatic configuration use case is going to be applied. The configurator (Wi-Fi device) creates a service discoverable through the network that is not configured and announce it. When the ethernet device is connected to the cable, tries to discover a service in this network which contains an IP. Through this IP this device establishes a connection such as a socket, e.g, and the configuration is sent from the configurator to this device. The Replication Configuration module is responsible for replicating an existing configuration to another device, after the service discovery phase. The Local Configuration module is responsible for locally configuring the device through Mobile Application. With a mobile application as a GUI it will be possible to communicate with the device in order to get the configuration profile and to send the users parameters entered on the application. The Security module is responsible for protecting messages exchanged and to protect the protocol. Specifically, when a message contains the configuration of another device or the parameters which need to be configured or when the configuration profile is transmitted. This information needs to be protected in order to prevent message eavesdropping. The best current solution to ensure that the communications keep its integrity and confidentiality, is by using certificates that can be trusted by the user through a secure connection using TLS. The Configuration layer is responsible to create and manage existing configurations and store older ones. Configuration Manager module is
responsible to receive the configuration of the device. The configuration information is kept on the Storage Module. The Storage Module is used as a persistent database of the system to store information such as the existing configurations of the device. In order, to the user interface provided by the device can has access to this information. The devices layer abstracts the particularities of the interaction/configuration of each device.

4. Implementation

4.1. Prototype

There are generic and specific implementation components. In this case, the specific ones will be implemented for a prototype that simulates an IoT operational device. This prototype will be a Raspberry Pi (RPI) containing always a profile with normalized structure using YAML. In addition, the entire network configuration involved in this system is Linux based and specific to the operation system of our prototype. On the other hand, the generic components can easily be exported to any device and run without any kind of changes. The Android application that will be implemented can be used by other mobile devices, with the same operating system, as an example of a component of this type. To implement our system, it was necessary to have a prototype so that we can simulate the operation of a real device. We decided to use a Raspberry Pi 3 single board computer. It was chosen because it is a cheap and fast platform that runs Linux and has many I/O ports. We have created a Smart Switch which communicates through Ethernet, NFC and Wi-Fi as shown in Figure 2. To set up this device we have used a breadboard, a Red Green Blue (RGB) Light-Emitting Diode (LED), three Male/Female jumper wires, two 220 OHM resistors and one of 470 OHM. We had to wire this to the RPI’s General Purpose Input/Output (GPIO) headers. This device provides an Android mobile application to control the color or light intensity of the LED. In order to the device communicate with the mobile application we have used Python language to create a script. This script receives the values of each color chosen by the user in the application. Once these values are received, the script will change the color in the LED according to the values received. These values are sent from the mobile application. In order to connect to Wi-Fi, this prototype create an Access Point (AP) that assures a connection from the mobile application. We had to use specific hardware material, since the RPI does not have integrated NFC. We have chosen to use the PN532 chip from Adafruit.

4.2. Software Architecture Implementation

Starting with Service Layer, it was necessary to have an alternative for this device to be reachable by others within the created AP. So we choose to implement an Apache HTTPS Server on port 443 of RPI. Additionally, we have created on port 8080 an Apache Tomcat as we can see in Figure 3. With this infrastructure, in the security module we have created a self-signed certificate with OpenSSL. Therefore we are ensuring that the information exchanged is not eavesdropped. We have also created a JAVA Web Server program which has three endpoints. One to send the configuration profile after a HTTP GET request, another to receive the configuration entered by the user on the mobile application through a HTTP POST message. And another one to send the configuration to another device. In local configuration through Wi-Fi, after the mobile application connects to the device’s endpoint through a HTTP GET request then the certificate is exchanged. Since, our certificate is self-signed we had to manually accept this one in the two modes because it is not trustable by an Certificate Authorities (CA). If this system were applied to the real world a relationship of trust would be established in the future with the manufacturer’s CAs. After the connection is secure, the device sends the configuration profile to the mobile application. Next, after the user entered the configuration parameters through a HTTP POST request the mobile application sends the configurations through a JavaScript Object Notation (JSON). The device, automatically gets configured, then creates a text file with the configuration parameters and send it to configuration module. This module will allow the access to this configuration of the device’s in-
faces. After configured, the device creates a service through the Bonjour Protocol which is installed by default on the RPI. This is done automatically using JmDNS \(^8\). This service is advertised in the network for devices that are looking for existing configurations. In automatic configuration, the non-configured device, through Ethernet, it searches for services in the network. If it finds, then automatically gets this service’s IP and connects to the endpoint to obtain the replicated configuration. Again, we have done this, using JmDNS and JAVA. The configurator device, only expects for a HTTP GET request at the endpoint as in local configuration module. For use NFC, the configurator creates a NFC tag containing the configuration through the nfcpy \(^9\) library in Python. The non-configured device must touch in NFC configurator’s chip to get its configuration.

![Figure 3: Information exchanged on AutoDev’s system](image)

**4.2.1 Mobile Application**

The mobile application was developed for Android Software Development Kit (SDK) \(^{21}\) with Android Studio \(^{11}\) as a Integrated Development Environment (IDE). It was implemented for Android, since, our previous knowledge in mobile computing are dominant in this system. The first activity of this application, has a toggle button which turn ON/OFF the smartphone’s Wi-Fi. The WiFiManager class allow us to perform this without challenging problems. This class, provides the primary API for managing all aspects of Wi-Fi connectivity on Android smartphone. The result of the Wi-Fi networks scan is shown on a List View as we can see in Figure 4.

![Figure 4: Activity of choosing the existing wifi! networks](image)

The user is warned through an Alert Dialog that must choose one of the presented WiFi networks to connect to his device. After pressing one of the items in the List View, again through, WiFiManager class, the smartphone connects to this network. After connecting to device’s AP, the user is prompted to another activity. This is the moment of scanning the device and looking for profiles and parsing this file. Since smartphones do not have the computational power of a computer if we execute these two tasks directly connected to the Thread that deals with the User Interface (UI) then the application during these moments will no longer be responsive to the user and may even stop executing. In order to avoid this, we have used the Android Priority Job Queue library \(^{12}\). Job Queue provides us a framework to define our background tasks as Jobs and enqueue them to our JobManager.
instance. Job Manager will take care of prioritization, persistence, load balancing, delaying, network control and grouping. It also provides a lifecycle for our jobs to provide a better, consistent user experience. If we had used the Async Tasks that the Android system provides we would penalize application performance and user experience. In addition, we would greatly increase the complexity of the application logic. We also use the Android EventBus library 13. EventBus is an open-source library for Android using the publisher/subscriber pattern for loose coupling. EventBus enables central communication to decoupled classes with just a few lines of code simplifying the code, removing dependencies, and speeding up app development. After this happens, the TLS handshake will start and the mobile application shows the device serial number from the device’s signed certificate. The user must verify if the serial number sticker on the device matches with the presented one in order to trust that is really the device that seems to be. Through an Alert Dialog that the user must confirm or deny. By confirm it, the GET message will be sent and as a response the smartphone will receive the configuration profile of the device. Next, through the EventBus library, an event is published to let the subscribers jobs know that the file is already present on the smartphone. When the event occurs, the Parsing Profile Job begins to execute. This job basically parses this file by creating different structures that need to be configured. To parse this profiles we have used SnakeYAML library 14. SnakeYAML is a YAML processor for the Java Virtual Machine. The user is prompted to a Tabbed Activity 15 that contains 3 tabs as we can see in Figure 5. Each one of these tabs is a Fragment and it allows to configure the different configuration parameters defined already in Section 2. When the user finishes to introduce all the data, we are going to create a single object that contains the configuration for the device. Again, in each tab we provide an Alert Dialog Box describing what is need to do. We show this alert before the user gets to a new tab, i.e, we show an alert for device functionality which is the second tab after the user entered the configurations parameters on the first tab and so on. Finally, the application leaves this activity to proceed to JSON creation of the information that the user has entered in the application. We have created another background job and using GSON library 16 we have created the JSON object. Using the Android Volley library 17 we have created a HTTP POST message which contains this JSON object created in its body in JAVA code. By accessing to the second endpoint provided by the device which expects a POST message we send this configuration for the device gets configured. We have used this library, because it is the only solution to create HTTP POST messages from Android. After this point, the user will be forwarded directly to Google Play 18 in order to download the respective application to use their device. We have tried to create a very user-friendly interface with a lot of Alerts that explains everything to the user as we can see in Figure 6.

Figure 5: Tabbed Activity to configure the configuration parameters

\[\text{http://greenrobot.org/eventbus}\]
\[\text{https://github.com/asomov/snakeyaml}\]
\[\text{https://developer.android.com/training/implementing-navigation/lateral.html}\]
\[\text{https://github.com/google/gson}\]
\[\text{https://github.com/google/volley}\]
5. Evaluation
5.1. User Tests Scenario
In order to test this system we have performed tests with real users. They were executed using two fully functionals Smart Switches. To simulate a domestic Wi-Fi network we have used a router Cisco-Linksys WRT610N. For users achieve a configuration and control the devices that contain our system we used a Samsung Galaxy S4 with an Android Lollipop 5.0.1. In addition, two IoT devices were used that do not contain an implementation of our system. We have used the Dlink Home Smart Plug and the Belkin Wemo Insight Switch.

5.2. Tests Description
Users have undergone several tests to verify whether the system is intuitive or not. We have tested the local configuration with Wi-Fi and the automatic configuration with Ethernet and NFC without any instruction manual that the users could follow. The times of the configuration processes without instruction manual were timed. Next, we have provided a set of instructions for users to follow during the configuration processes for both system modes with all of the above technologies. Configuration processes times were also timed. In order to have a means of comparison between our system and real IoT devices, users have configured one of the real devices mentioned in the previous Section. To do this, we have provided the instruction guide of the devices for the users configure them. The configuration processes times for these devices were also timed. After all these tests, users have completed an inquiry into the intuition of the system and whether it is in fact user-friendly and simple to use. Users were asked to give their opinions about the system. With these tests it is expected that users will be able to configure devices with low complexity not wasting much time. In addition, an average of 5 to 10 minutes for the local configuration with and without instruction manual is expected. For the configuration automatic no more than 5 minutes to complete the processes.

5.3. Tests Results
Since, only the local configuration is comparable with real devices, we will only present results for the local configuration with Wi-Fi. There is no comparable model with our automatic configuration mode. In total, 25 users performed our tests. Of this number, 12 were female and 13 were male as we can see in Figure 7. All the results presented are going to be through histograms with intervals with 0.5 minutes.

5.3.1 Local Configuration
5.3.2 Without Instructions manual
Regarding the test of local configuration without user manual, the results of the times of the configuration processes can be found in Figure 8.

The average time obtained was 3.68 minutes. Overall, the results were quite good and showed that even for less experienced users. These users as
mentioned took a little longer than expected since there are older users in terms of age and have a greater difficulty in carrying out the processes. The most experienced users took up to about 5 minutes. This showed us that the mobile application has a good user support.

5.3.3 With Instructions Manual

The average time of this test was about 4.89 minutes. Most part of the users felt that the mobile application has a good support and that the instructions manual is not needed. During the test performed, the users lost time by reading the instructions manual and the mobile application alerts.

5.3.4 Real IoT devices

As we can verify, the configuration times obtained are considerably more time consuming than those of our system. Because they are time-consuming, it is also safe to assume that they are much more complex. In some cases the setup took more than 15 minutes.

As we can see, the shortest time obtained configuring these devices increased significantly compared to the local configuration with AutoDev (with or without instructions manual). As we showed in this document the existing processes are a time consuming task and it is not user’s fault. In this case, in general all the users had problems in the middle of the configuration because the instruction manual was not clear. Since this information was not clearly described the users did not know what to do so in most of cases they had to repeat this process again.

5.3.5 Inquiry to users

After all these tests, users answered to questionnaires which contained the some questions. The first one, if in the tests without an instruction manual, the system is intuitive for both modes. Next, to give their opinions about the system and things to be improved. Finally, after configuring the real devices, were asked if the process is much more difficult and complex than the performed with our system. Overall, in the answer to the first question the users thought that in the local configuration the system is very intuitive and simple to use. As far as the second question is concerned, user feedback was very positive. There was no user claiming that did not like the system. Many felt that the system is practical and should be implemented quickly in their devices. There was a minority of users, especially the less experienced, who said that in the local configuration the user will have to retain a lot of new information. Still, with these claims the configuration processes times remain much lower compared with the obtained in configuring the real devices as we showed. Regarding the third question, all users stated that the actual devices have configuration processes that are much more complex and time consuming than those presented by our system. Users assumed that the real devices that have configured were much more difficult to configure than our system.

6. Conclusions

In this document, we present a system that has the objective to reduce the complexity of the configuration processes of IoT devices. Furthermore, offer a better support and less complexity to the domestic users. In order to develop the solution for the problem, we started by analyzing the IoT concept technology. Next, we have analyzed its constituents such as the smart objects. After realizing how IoT technology works, an analysis was made to the devices available in the market for the domestic users.
The limited flexibility, customization and efficiency of the current solutions were identified and used as a base during the development of the AutoDev system. The architecture of the system was designed and implemented in a prototype. Finally, using this prototype, the solution was tested with users in a scenario. Some of the main achievements of AutoDev are the definition of flexible and less complex configuration processes, a software architecture and a implemented prototype. Another contribution of AutoDev is offering one intuitive user interface that ease the introduction of parameters to configure these type of devices. The proposed software architecture was instantiated in a fully functioning prototype. A deployment of the system in this prototype was used to validate its operation and to test system’s intuition. Even though most the developed prototype is fully operational and the goals set in the beginning were fully achieved, this is a very active research area and many ideas may be pursued. Many issues were found during the development of this thesis. The fact that we did not established a trust relationship with a CA made the generated certificates not trustworthy. So this should be one of the points to be explored in the future. In this area, security and the protection of user’s privacy should be a must. Finally, in the future this system can be able to take control of the IoT devices. Allowing, then the interoperability between several devices. With the flexibility introduced in this system through the description of the configuration parameters of the device in a configuration profile. It leads us to think that the commands to control the device can also be described. Our mobile application, after the configuration process, can show a list of the configured devices and when the user selects one of them can control it through this application.

References