Automatic Integration of IoT Devices

Abstract—**During the last years a new concept has gained ground in the technology world. With an increasingly dominant role in our days, Internet of Things (IoT) is a technological revolution that is changing our lives. The imagination is the limit regarding the new devices that may arise in the market. This phenomenon is derived from technology evolution and the growing acceptance of this type of products in our social life. Faced with such diversity, growing and with the existing uncertainty about the new devices emerging in the market, there is a need to integrate each new device in our lives. However, nowadays, there is no existing structure prepared for the expectable emerging diversity in the IoT world. This work pretends to provide some help in this specific point by automating (part of) the device integration process.**

Applications that integrate devices from many different vendors are now available, but these rely on manual configuration by the application developers for every new device integrated, which is tedious and requires application updates to be rolled out frequently. An application that can discover the new device properties, and decide to which class of devices it belongs to can automatically generate an interface and the necessary integration drivers for the new device with no/less human intervention. This is the main direction of this works' contribution.

To achieve this goal we need to identify the device that is communicating inside a network using the information shared by IoT devices. The final results show that IoT communication data can be used to identify the device, mainly if in possession of a considerable sized device information database.

Keywords—***Internet of Things, Discovery, Integration, Automation***

I. Introduction

The Internet of Things (IoT) concept is derived from the current set of existing devices on the market that have the property of being able to connect to the internet and become ”online”. There are more smart devices appearing each day, this phenomenon is derived from technology evolution and the growing acceptance of this type of products in our social life. The imagination is the limit regarding the new devices that may arise in the market. Faced with such diversity and with the existing uncertainty about the new devices emerging in the market, there is a need to integrate each new device in our lives.

Each device has its way of functioning, but there is a common feature in all: these devices must have the ability to communicate. This communication may be made through various communication protocols, however, the aim is always the sending / receiving of data between devices and the ability to communicate in the same protocol. To make this happen it is necessary to establish this connection, which involves some settings such as the authentication process.

What if there was a way to automate or facilitate this integration? Is it possible through the information generated by the device to discover what type of device that is, what steps are necessary to establish communication, and how to integrate it seamlessly in our own application's device catalog? The aim of this thesis work is to identify and study the communication of the different devices on the market. It is also an objective to search through device information in order to create ways to automate the integration of any new device.

This study aims to answer two fundamental questions: Can we ease or even automate the integration of new devices on a home network? Can this automation be applied on top of the current configuration system?

A. Definition

The definition of Internet of Things (IoT) is not fully consensual among different organizations and authors. Some examples of definitions show just that:

”The basic idea of the IoT is that virtually every physical thing in this world can also become a computer that is connected to the Internet (International Telecommunication Union (ITU), 2005). To be more accurate, things do not turn into computers, but they can feature tiny computers. When they do so, they are often called smart things, because they can act smarter than things that have not been tagged” Fleisch [1].

”A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies” I T U [2].

”The Internet of Things (IoT) is the network of physical objects accessed through the Internet, as defined by technology analysts and visionaries. These objects contain embedded technology to interact with internal states or the external environment. In other words, when objects can sense and communicate, it changes how and where decisions are made, and who makes them” Systems [3].

IoT is ”a pervasive and ubiquitous network which enables monitoring and control of the physical environment by collecting, processing, and analyzing the data generated by sensors or smart objects.” Systems [3].

However there are some important aspects that help define better this expression. Starting by analyzing the name, ”Internet of Things” is syntactically speaking, formed by the two terms ”Internet” and ”Things”. The first term shows the instrument under which the IoT works and one of the reasons for such a success, the emergence of this phenomenon was only possible due to the already existence of another major phenomenon, the Internet. It is also through the power of the Internet that all objects can communicate with each other and thus provide all its advantages. This statement leads us to the second term , the ”Things”, which according to ITU ”is an object of the physical world (physical things) or the information world (virtual things), Which is capable of being identified and integrated into communication networks” I T U [2] . The ”Things” are any object with a unique and identifiable ID that generates some kind of information and can then share that information over the internet. The definition of this term is more easily understood if we check the purposes of Internet of Things.

B. *IoT Purpose*

The purpose of the Internet of Things can be summarized to three important aspects. The first goal is to transform the objects in connectable objects or smart objects providing them with some intelligence and the means to allow them to share their own information. For this to be accomplished it is necessary to integrate the hardware that will make it possible for the object to communicate with any platform. In the next chapter we will detail the various ways of doing this. Due to miniaturization, it is possible nowadays to place this type of hardware even in the smallest of places, allowing its inclusion in all kinds of objects. This miniaturization allows the IoT to reach more and more applications where it was not possible in past.

Another important goal is to create the necessary conditions for these objects to communicate with each other (within the same network) and globally (Internet), allowing, through synergies, to add even more value to the information obtained. This communication is one of the main challenges of IoT nowadays, given the diversity of objects, platforms (Iotivity, HomeKit, AllJoyn ..) and methods of communication (WiFi, Bluetooth ..). The growth in the number of existing objects is increasing the amount of data generated and the main goal will be to explore well this information and make all efforts to get various benefits through the IoT world.

C. *IoT Growth*

The term Internet of Things makes its first appearance in a presentation made in 1999 about Radio Frequency Identification (RFID) Ashton [4].

Fifteen years go by and, the number of existing IoT devices never stopped growing and has even surpassed the number of existing people on the planet in 2008. In last years the number of new devices has grown exponentially and if today we have 15 billion devices, in 2020 it is anticipated that there will be 50 billion devices worldwide, about 6 per person.

This growth is mainly due to technological development that allowed the miniaturization of hardware and the reduce of material costs. Another reason is the ”hype” that we can verify with the increasing marketing around these products, using the word ”smart”. These are indeed incredible numbers that make us wonder if the structures used today are really prepared for this growth or at least if we are on the right track.

II. IoT Architecture

The architecture on Internet of Things (IoT) can be summarized by four functional layers: the interaction layer, the representation layer, the service layer, and the application layer.

The Interaction Layer is formed by the connected devices. On this layer are included all the hardware pieces physically attached to an object that enables its communication with other devices and the Internet. This layer allows the interaction of the objects with the real world and the information generated therefrom is sent to the upper layer.

The next layer (Representation Layer) is responsible for managing each of the objects, providing an identification method (e.g. the Uniform Resource Identifier (URI) / Uniform Resource Locator (URL)) and a way of establishing communication with each object using its own methods. This layer makes possible to the objects the Exchange of information with each other and with the outside world (Internet), allowing its use by the top layer.

The Service Layer works over the Representation Layer to provide the functionality and information from all the objects connected. This layer will allow users or programmers to use this information in a standardized way.

Finally, the Application Layer is formed by many types of applications where the features and the information provided by the Service Layer are consumed.

A. *Connection Modes*

There are several communication modes that allow objects to receive and send information. Each type of communication has its advantages and disadvantages and objects can choose one or more types of communication regarding its features and goal. The connection modes can be one of four types: Connection on demand, Connection when within range, Permanent wireless connection and Permanent wired connection.

Connection on demand type involves user interaction, and is typically used in cases where the exchange of information is needed only in the presence of the user. As examples, the Quick Response Code (QRCODE), Smart Card or RFID.

The type ”Connection when within range” requires proximity between the two parties. This type of communication is only possible when the two objects are close enough and it is used on objects that do not require a continuous flawless communication. The Bluetooth and the Near Field Communication (NFC) are examples of this type of communication.

Finally, within the Permanent connection mode we have the Wireless and Wired connection types. These two types are distinguished by the need of physical connection (via cable). Examples of wireless connections are the Third Generation (3G), the Wireless Fidelity (WIFI), Z-Wave and ZigBee. Finally as examples to cable connection we have the Digital Subscriber Line (DSL) or optical fiber (Ethernet).

B. *Communication Protocols*

In this section we will detail the most commonly used communication protocols inside the IoT world. The existence of many distinct protocols is directly related to the different requirements of the devices inside their own purpose. Besides the various connection modes that we define on subsection II-A other necessities like required bandwidth or energy consumption limitation opens space for this distinct ways to transport data. With no specific order, the following communication protocols are the most widely used nowadays on IoT:

Bluetooth: created by Ericsson company in 1994 Haartsen [5], Bluetooth is still one of the most used protocols. It is a short range communication technology that start to gain popularity by the exponential adoption on mobile devices on late 90's. It is a wireless protocol, initially based on Institute of Electrical and Electronics Engineers (IEEE) 802.15.1 that works on 2.4Ghz frequency and have a maximum range of 150 meters. Since the creation of the specification on 1998, there have been released several versions with many enhancements like faster connection and lower energy consumption to save battery power in mobile devices. The last version 5.0 was announced in June 2016 and the enhancements are mainly focused on Internet of Things emerging technology. This is a important sign that this protocol will continue to be adopted on IoT devices that needs this specific characteristics of Bluetooth communication.

Zigbee: this is a wireless protocol based on IEEE 802.15.4 specification and works on different radio bands depending on each country jurisdiction (868 MHz in Europe). It was developed by Zigbee Alliance Kinney et al. [6] in 1998 with the target of low energy consumption and applications monitoring with use of different kind of sensors. Zigbee protocol main use is low sized data transfers where it is more efficient than other protocols (e.g. Bluetooth). This protocol is completely settled on IoT world with the most recent version Zigbee 3.0 Alliance [7] promising even easier communication and interoperability among devices.

Z-Wave: developed with an explicit focus on home control applications Reinisch et al. [8] Z-Wave is a wireless protocol based on ITU-T G.9959 specification and it's purpose is to provide a faster and simpler development by using a simpler protocol (comparing with others like Zigbee). It is optimized for reliable and low-latency communication of small data packets with data rates that can reach 100kbit/s (vs 250kbit/s of Zigbee or 1Mb/s on Bluetooth) and is capable of communicate with a maximum distance of 30 meters. This protocol was created by Z-Wave Alliance Alliance [9] in 2005.

Ethernet: this is a wired connection type protocol. It is based on IEEE 802.3 specification and it was first standardized on 1983 CarlSSon [10]. Since it requires physical connection it is only used on very specific situations like high security demands. The last standard release was IEEE 802.3bz on September 2016 and the data throughput can reach up to 5 Gbit/s.

WiFi: created on 1997, this is probably the most well known protocol. WIFI comes from the words "wireless fidelity" Lee et al. [11] and is based on IEEE 802.11 specification. This protocol is by no means specific to IoT use, however many manufacturers choose this protocol because it's a very well settled protocol and due to the need of high data transfers. Besides 2.4 Ghz communication on first versions, the most recent IEEE 802.11 ac specification uses 5.0 Ghz that allows a maximum throughput of 1.33 Gb/s maintaining a maximum range of 70 meters. This protocol uses more energy power due to its range and high data transfer capacities.

Cellular: IoT applications that need to operate over longer distances can take advantage of Global System for Mobile Communications (GSM), 3G or Fourth Generation (4G) cellular communication protocol. GSM was released back in 1991 Liikanen et al. [12] and, most recently 4G LTE was released on 2009 Huang et al. [13]. The main use of this protocols is the wireless mobile telecommunications and, for that reason, the range is very large reaching from 35 kilometers on GSM to 200 kilometers on High Speed Packet Access (HSPA). In this case the energy consumption can vary between low to very high demands depending on the amount of data to send/receive. The data throughput can reach up to 100 Mb/s on 4G communication and the radio frequencies used depends on each country (900, 1800, 1900 and 2100 MHz are the most used worldwide).

NFC: this is a protocol based on ISO/IEC 18000-3 standard and it was has been developed jointly between NXP Semiconductors and Sony Corporation Curran et al. [14]. NFC has the objective of allow easy and safe two-way interactions between electronic devices like smartphones. One example of usage is the contact-less payment transactions. This protocol uses 13.56 MHz radio frequency and have a maximum data throughput of 420 kb/s. There is two of communication: active and passive. On active mode both devices uses electric fields where in case of passive mode only one needs to generate electric field. The range is cut to only 10 centimeters and the power consumption is minimal compared to Bluetooth or even Zigbee.

C. *MAC Address Discovery*

The IoT devices contain a specific piece of hardware (Network Interface) that is responsible for establishing a connection. Each one of them are labeled with a unique identifier that is called Media Access Control (MAC) address. These addresses are formed according to the rules of specific standards managed by IEEE. Part of this address is reserved for the Organizationally Unique Identifier (OUI) that allows the association of MAC addresses to a specific company. Any device can see the MAC addresses and respective Internet Protocol (IP) of other devices inside the same network. With this mac address we can discover which company the device belongs by reading the included OUI . However, this information is very poor because it only gives us the name of the company and sometimes no organization is associated with the MAC address received. In most of the cases, one company produces many different devices that will have the same OUI, thereby preventing the use of this technique for device discovery. One example of a software capable of doing this is Wireshark Wireshark [15].

III. Platforms

In this chapter we will present a brief summary of each of the actual major 4 IoT platforms in the market.

A. AllJoyn

AllJoyn is a open source Application Programming Interface (API) that provides a simple work environment for the IoT objects. Initially developed by Qualcomm Innovation Center, Inc the AllJoyn project is now on the hands of AllSeen Alliance. This environment allows the object to be easily discoverable and at the same time guarantees security. AllJoyn compromises to ”reduce the time, effort, and cost of adding advanced features to apps and help ensure interoperability across device types and operating systems.”Alliance [7]. There is now more than 200 companies using AllJoyn API and that can show the success of this Platform.

B. IoTivity

Iotivity is an Open Source Project sponsored by the Open Interconnect Consortium (OIC) Arseni et al. [16] and hosted by the Linux Foundation which was founded in July 2014. Counting now with more than 50 members, the aim of this project is to develop an open source software framework to seamlessly connect the billions of devices in the emerging IoT across multiple operating systems and network protocols. This open source implementation helps to ensure interoperability among products and services regardless of maker and across multiple industries, including smart home, automotive, industrial automation, and healthcare. The IoTivity architectural goal is to create a new standard by which billions of wired and wireless devices will connect to each other and to the internet. The goal is an extensible and robust architecture that works for smart and very small devices Foundation [17].

C. Apple HomeKit

HomeKit, is the Apple’s platform for home automation and it was introduced at the 2014 edition of the Apple Worldwide Developers Conference Nimmermark and Larsson [18]. In relation to the other 3 platforms, this platform covers less devices because it is designed only to home automation use. Unlike the other platforms discussed in this work, Apple’s Home Kit is not an open source framework, so you have to submit an enrollment form to join the Home Kit program. Apple has a big marketshare in the smartphone area so this platform has an initial advantage to be well succeeded. However this is not enough because the IoT world is scattered and this success is directly tied to the companies will to add Home Kit support in their devices. To Apple, the main advantage of this platform is to guarantee that home automation accessories compatible with Home Kit, can all be integrated into a single coherent whole without vendors having to coordinate directly with each other.

D. Google Brillo

Google is the latest entrant to the crowded IoT market with Google Brillo platform. Presented on Consumer Electronics Show (CES) 2016 Google Brillo is meant to make tiny IoT applications, highly efficient, very fast Hexmoor [19]. This open source platform is composed by a lightweight embedded OS based on Android, core services and a developer kit. Brillo uses Google’s own communication protocol (Weave) that is based on Bluetooth and WIFI together to communicate with other devices. Google is already working with hardware partners to certify the boards that are compatible with Brillo knowing that the OS can run on low-end devices with at least 128MB of storage and 32MB of RAM. As well as Home Kit, Brillo success depends on companies adoption and their hope relies on the widely used Android that can bring some advantage on this ride.

IV. Applications

This chapter demonstrates the variety of areas where the IoT is already implemented and future implementations that will be possible with next year’s technology advances. However, with such spread of devices existing on the market and new applications appearing everyday, it is almost impossible to cover all the possible uses of IoT, so we will only detail three main applications areas.

A. Smart Home

This is a special place for most people. We search for comfort, security, entertainment when we are home. So it’s normal that IoT applications goes in our needs direction. There are new IoT ideas made from scratch but also normal appliances that we use each day that are ”enriched” with the possibility to connect to the internet and become IoT devices. Previously we use air conditioners to acclimatize our homes and we control it with remote controls, but with IoT, these devices became connected and while we are working or arriving home we can turn it on/off. This provides the comfort of having our favorite temperature set when arriving at home. Washing machines, Lights, Cameras are all examples of devices that already entered on the IoT world providing advantages to our lives.

B. Healthcare

Medical objects can also benefit from IoT with the acquisition of new capabilities. For example heart rate monitors or other life supporting monitors are now connected to the Internet alerting for emergencies or anomalies. Scales and Blood Pressure monitors are now ready to save historic information that can be crucial on some medical decision. Automatic scan based on sensors reading specific biometric features could save time in triage and speed the processes up by categorizing certain symptoms. Providing such objects with intelligence and autonomy can represent a big step to an increased quality of life in healthcare area.

C. Environment

The use of IoT devices can bring many advantages to the environment too. The main idea is to alert the users of their footprint, something that cannot be done before without this new devices. An example of this are the new sensors that can be placed on energy counters or even gas/water counters, we can have more control on our usages leading to a greener life. All the IoT devices can have the intelligence to auto shutoff when not in use avoiding normal human forgetfulness. Devices connected to the car already alert conductor to an efficient driving spending less fuel and releasing less pollutant emissions. If more people contribute this way, with the help of IoT devices, we can believe in a less polluted environment.

V. IoT Challenges

As we already discussed, IoT can bring many benefits on different application areas and new devices are emerging everyday. Still, as the evolution of the IoT takes place, there are also barriers appearing that have to be surpassed to guarantee a sustained growth. Currently IoT is facing (or will face in the future) the following challenges:

Security: The simple fact that each device can send and receive information creates new security concerns. One of the main reasons for these concerns is that the devices themselves are often in vulnerable locations. Security will have to be integrated as part of IoT infrastructure because the networks where the devices communicate can't always be trusted and, in this case, each device becomes an ”open door” to malicious attacks. This possible vulnerabilities need to be checked and security measures needs to evolve at the same speed of IoT growth.

Privacy: Nowadays, internet already faces many privacy problems. Due to the nature of IoT devices, and its use on almost any situation, many private information is stored on servers and that leads to questions like: “is my privacy assured?”, ”could companies access my private data?”. Voice recognition is being used on devices like Amazon Echo which sends recordings of our voice to cloud servers. This kind of devices can continuously listen to conversations, dealing therefor with very sensitive data. The collection of this information exposes legal and regulatory challenges facing data protection and privacy law that needs to be surpassed.

Interoperability: This is one of the main challenges in the IoT world and also the barrier we want to surpass in this work with the integration of the different devices. As discussed in this chapter, there are many communication protocols used by devices and different platforms providing the necessary infrastructure to ensure communication between devices. The absence of a unique standard consequently allows each device to use their own implementation and, therefore, IoT interoperability continues to be a big challenge. There has to be an effort inside the main IoT companies to agree on a global standard that defines the rules for every IoT devices, however, in my opinion, most companies have other priorities.

Data Treatment: IoT devices generates more and more quantities of data that need to be processed and analyzed, sometimes in real time. Processing and analyzing these large quantities of data will increase at the same proportion the capacity of data centers. This brings to companies new capacity and analytic challenges that has to be surpassed to guarantee the right preparation to the expected IoT growth.

The Interoperability challenge described above is the main reason to the difficulties we face today on IoT devices integration. If we have a unique standard that defines the identification data that all IoT devices has to comply, the automatic integration of devices will be an easy task. In that case, we will know exactly where to get the essential information about each specific device and easily integrate it in any new environment or platform. However, due to this limitation, we have to find another ways of doing this integration automatically, and that is what we will try to accomplish with the prototype described on next chapter.

VI. Prototype on Device Discovery

With so many devices and their distinct goals, the use of different connection modes is a reality. This diversified way of establishing connections creates a handicap in one of the biggest purposes of the IoT: to make ”Things” connect with each other. In fact nowadays, the user needs to integrate each smart device one by one with their specific procedures. This work plans to bring some help to the user in that particular moment. Each IoT device in our homes connects to the internet through our home network (Permanent Connection).

With the development of a specific software that can listen in our home network, we can receive and see the communication of the devices. You can find an example in APPENDIX A. Our goal is to analyze this communication and, with some techniques, automatically discover and identify each device that is communicating inside our home network. By identifying each device we can automate or partially automate (depends on the device) the integration process. This prototype will provide an example of this automation technique demonstrating each step of the process discussed on this thesis.

A. *Collaboration with Muzzley*

To explaining all the steps of this prototype i need to start explaining the origin of this dissertation. This work was made in collaboration with Muzzley Muzzley [20]. Muzzley is a young start-up company working on the IoT world, and they have the brilliant idea of creating an app that can join together many IoT devices simplifying the way we interact with them. I was surprised with their great working ambient and I feel privileged to be able to do this dissertation with their collaboration. In Muzzley's ”home” i can find some IoT devices to work with and thereby to test my prototype. I also benefit of great know-how inside this theme to overcome all my doubts. This work can be very important to Muzzley team because this is a feature that they want to implement in their app in the future and, as a consequence, the main reason for the arise of this dissertation theme.

B. *Prototype Architecture*

To begin our prototype we need to design a structure to define how we will get the data from the IoT devices. As we already know, all the devices sends info into the network that is connected. The first step was to find a way to receive this information so that we can use it on our prototype. Taking advantage of the huge know-how from Muzzley, i was advised to use a tool called Charles Proxy. This tool is a HTTP proxy that enables a view of all the HTTP and SSL / HTTPS traffic between their machine and the Internet including requests, responses and the HTTP headers and it works as a man-in-the-middle to catch all the info that the devices are sending inside the network. So we will have three main components in this architecture:

- The IoT devices that are sending information through the network (right on Figure Figure 1)
- A computer that is running Charles Proxy working as man-in-the-middle
- One device (in our case a smartphone) that is receiving information from the devices (left on Figure Figure 1)

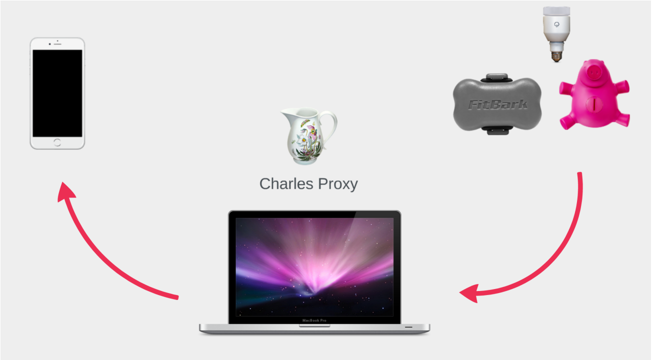


Figure 1. Prototype Architecture

When the information is sent from the IoT devices to the destination (in this case the smartphone) the computer acting as man-in-the-middle will be able to see the data. This information will be necessary to prepare our prototype dataset and will also serve as input to test our prototype and try to identify the device that is communicating.

C. *Device Data*

Using the method explained above it was possible to get the data to prepare our dataset. Each information we receive from a device is a communication file. One device can generate one or more communication files, for example one smart light can send a file informing the status of color and brightness and other file with the current schedule to turn On/Off. The communication files are sent by the devices in eXtensible Markup Language (XML) or JavaScript Object Notation (JSON) formats. We could not retrieve communication files from all the devices because some of them, even with the Secure Socket Layer (SSL) certificate installed on Charles Proxy, remains encrypted. In our data collection we were able to get info from 15 different devices. These devices originated a total of 37 communication files. Afterwards we made another data collection where able to get an additional 24 communication files from 5 new devices.This was the data we use to start out tests.

VII. App Development

After having in our possession all the communication files from the previous step it was necessary to start developing our prototype so that we can make use of the data already acquired. Taking in account the fact that Muzzley product is a smartphone app and that nowadays we also use our phones to consume IoT, it will certainly be a good decision to develop our prototype in a smartphone platform. So our decision was to implement our prototype in a Iphone Operating System (iOS) device. The iOS is an operating systems and it runs exclusively on devices made by Apple company. It is the second most used mobile operating system worldwide and Muzzley application also runs in iOS so it can be easier to integrate this functionality on the future.

The Integrated Development Environment (IDE) used to develop this app was Xamarin Studio. Xamarin Xamarin [21] is a platform that helps developers to share an average of 75% of app code across all the three main mobile systems. Thanks to this platform, although this app was made to work on iOS, it can easily be modified to be compatible with windows mobile and android.

The app is prepared to receive the communication files and perform two main tasks:

- Create a database that will store the received communication files, well organized and prepared to be used
- Try to identify a device based on the received communication file by performing the necessary algorithms

The application source code is available at Bitbucket [22].

A. *Import Data*

The data import process is a crucial stage inside the app because we need to be sure that we store the information received on the communication files the right way. The objective is to have the information prepared to be used by the algorithms on the device discovery stage. To accomplish this we separate each property from the respective value. One communication file have many properties. In Figure 2 we can see the property (”ActivityValue”) and the respective value (”30029”). We pick each of this values and distinguish them inside the database. We also add as a parameter the value type of the property, it can be a single value property or a multiple value property

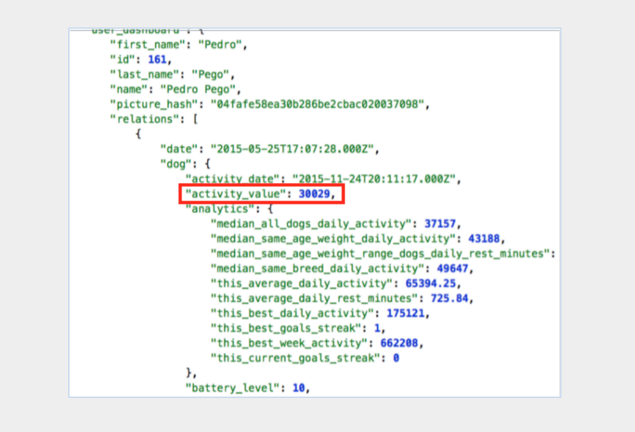
At the moment of importation of the communication file we specify the category and the name of the device. The categories are predefined on total of 9 categories: ”Security”, ”Other”, ”Health”, ”Activity”, ”Sensor”, ”Controller”, ”Light”, ”Display”, ”Utility”. This will enrich the information of each property. Finally we also store the name of the communication file as parameter.

So, to summarize, for each line of the database table we store the following parameters: ”Property”, ”Value”, ”Device”, ”Type (Category)”, ”Value Type” and ”Parent File”.

After importing all the 37 communication files we were able to retrieve a total of 17547 proprieties into the database table.

B. *Device Discovery*

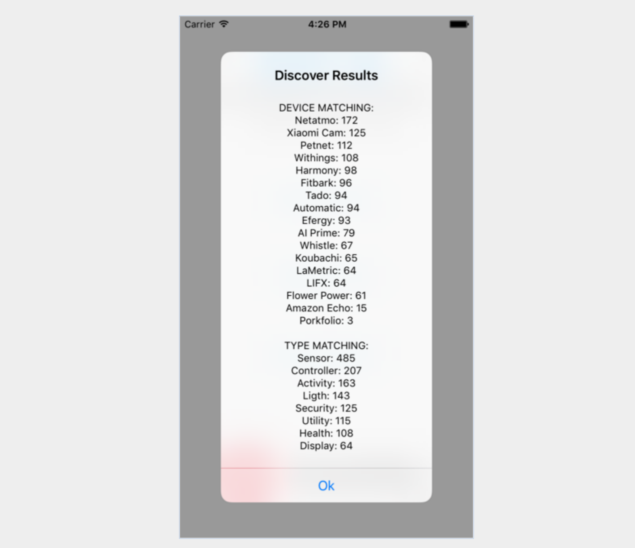
The other stage of the process is actually our main goal: identify the device. The iOS app is prepared to receive both



```
    "user_dashboard": {
        "first_name": "Pedro",
        "id": 161,
        "last_name": "Pego",
        "name": "Pedro Pego",
        "picture_hash": "04fafe58ea30b286be2cbac020037098",
        "relations": [
            {
                "date": "2015-05-25T17:07:28.000Z",
                "dog": {
                    "activity_date": "2015-11-24T20:11:17.000Z",
                    "activity_value": 30029,
                    "analytics": {
                        "median_all_dogs_daily_activity": 37157,
                        "median_same_age_weight_daily_activity": 43188,
                        "median_same_age_weight_range_dogs_daily_rest_minutes":
                        "median_same_breed_daily_activity": 49647,
                        "this_average_daily_activity": 65394.25,
                        "this_average_daily_rest_minutes": 725.84,
                        "this_best_daily_activity": 175121,
                        "this_best_goals_streak": 1,
                        "this_best_week_activity": 662208,
                        "this_current_goals_streak": 0
                    },
                    "battery_level": 10,
```

Figure 2. Property and Value Example

XML or JSON formats and when it receives a new communication file, besides the possibility to add it to database we can also select the option ”Discover Device”. This option has the purpose of use the database and the received properties on the communication file input to try to identify the device or the category of the device. This identification can be made with different algorithms that we will detail below. The final output on the app is illustrated in Figure 3.



Discover Results

DEVICE MATCHING:
Netatmo: 172
Xiaomi Cam: 125
Petnet: 112
Withings: 108
Harmony: 98
Fitbark: 96
Tado: 94
Automatic: 94
Efergy: 93
AI Prime: 79
Whistle: 67
Koubachi: 65
LaMetric: 64
LIFX: 64
Flower Power: 61
Amazon Echo: 15
Porkfolio: 3

TYPE MATCHING:
Sensor: 485
Controller: 207
Activity: 163
Ligth: 143
Security: 125
Utility: 115
Health: 108
Display: 64

Figure 3. App Discover Mode

VIII. Testing Device Discovery

Now that we have all the elements to test the device discovery we need to find strategies to relate the info present on the database with the info received from the device that we wish to identify. During our work we select three different approaches: The ”Levenshtein Algorithm”, making use of a Manual Keyword Table, and finally the Term Frequency – Inverse Document Frequency (TF-IDF) table approach. When we receive the new communication file in the app we are able to select one of those to start the discovery process.

After selecting the desired option, the user just need to click on ”Start Discover” and one of the detailed algorithms below will start running on the app.

A. TF-IDF Table

Besides the manual keyword table, we also use a special table that is based on a different statistic approach. Using the TF-IDF value we can generate a table where we can find the most significant words inside each category. TF-IDF algorithm value is calculated ”by determining the relative frequency of words in a specific document compared to the inverse proportion of that word over the entire document corpus. Intuitively, this calculation determines how relevant a given word is in a particular document. Words that are common in a single or a small group of documents tend to have higher TF- IDF numbers than common words such as articles and prepositions”Ramos [23]. The formula is shown in Equation 1.

$$W_{x,y} = tf_{x,y} \times log(\frac{N}{df_x}) \tag{1}$$

where:

$tf_{x,y}$ = frequency of $_x$ in $_y$
df_x = number of documents containing $_x$
N = total number of documents

To find the most relevant properties in each category, in this prototype, we calculate the frequency of a property in all communication files, comparing it with the frequency of a property in the communication files of a specific category. At the end of the algorithm it will appear a list of the devices and categories that contains more total ”value” based on the TF-IDF values calculated in this table.

B. Levenshtein Algorithm

The Levenshtein Distance Algorithm is a string metric for measuring the difference between two sequences and ”it calculates the minimal costs required to change a string of segments into another by means of insertions, deletions or substitutions”Beijering et al. [24]. So, for instance, let's imagine that we have two proprieties values (strings), one from the new device and one from the database, the Levenshtein Distance value will be the number of modifications needed to transform one string into another. The insertion and deletion of a character in the string have a cost of 1 unit and the substitution of a character has a cost of 2 units.

As an example, the strings ”rick” and ”rcik” have a Levenshtein Distance value of 2 units because we need to delete the ”c” and insert the ”c” in other position (other solutions with same cost are possible). Due to the relative comparison that we need to accomplish between the information on database and the new device proprieties, this distance value can be applied to our problem. We compare all the proprieties available with the each propriety received in the communication file (from the device that we want to identify).

Now that we have all the values compared, we need to define what is a positive value. After some trial error test we define as a positive match a Levenshtein Distance with value <4 (see Table V on section IX). To exemplify, the word ”color” is a positive match to ”colored”. We also remove the word with less than 4 characters from this comparison like ”ID” or ”key”. At the end of the algorithm it will appear a list of the devices with more positive matches as well as the categories that contains more proprieties that have positive matches.

C. Synonyms Match

With the aim of getting more positive true matching results we implemented a dictionary of synonyms in our prototype. The main objective is to use each propriety name of communicate file, and, on device discovery, ”transform” it in multiple proprieties with the same meaning. For instance, when the new device have a propriety ”hue” retrieve all the synonyms of this word like ”color” and search for matching proprieties on database for all the synonyms found. This way, if a communication file from a device contains 300 proprieties inside, we can use in our matching algorithm 600 properties or more (including the synonyms). To guarantee this additional information, we needed to find a way to get the synonyms into our prototype. Initially, we started to use a online API Altervista [25].

With this API we can make webservice calls containing a specific word and receive a list of synonyms. This API is very simple to use, however each call takes about 3 seconds to return values and sometimes the discovery process take 30 minutes or more to complete (depends on the number of properties inside the communication file of the device to discover). We had to choose another approach to retrieve the synonyms of each proprieties. After some research on the internet we were able to find a free English thesaurus file online in Gutenberg [26].

This way we can add this file to our project and parse the synonyms locally almost immediately. This dictionary is very complete with more than 33000 words with synonyms. For example the ”color” word returns the following synonyms: appearance, blackness, exaggeration, falsehood, painting, redness, resentment.

D. Multi-Property Matching

All the above approaches were property-based, i.e. the attributes used both for training and for testing were those of a single communication of one property per example. This research path was pursued because data, although rich in properties (over 17500), was poor on devices (roughly 20) due to the encrypted communication formats used by many devices (as explained in section subsection VI-C) and also because the identification timing is important and the possibility of identifying a device using a single property set action had to be investigated.

Of course, having more information on each device should entail better results, so a final experiment was necessary to assert how the use of accumulated information for one device over a certain time, where several of its properties appear, can affect the results.

We run the TF-IDF algorithm (as explained in section subsection VIII-A) over the proprieties generated with this method to obtain the most type specific ones (the proprieties that appear frequently on a specific type and not on the remaining). Due to our number of devices used in tests, we have to manually revise these tables by removing some type unspecific proprieties (i.e. ”status” or ”device”). This last step could be automated with the use of techniques like Principal Component Analysis (PCA) over a database containing a bigger number of devices.

Finally in possesion of these tables we used a technique named K-fold Cross-Validation. This technique consists in dividing data set into k folds (subsamples). Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k - 1 subsamples are used as training data Rodriguez et al. [27]. In this case we divided our data in 10 subsamples of 2 devices. With this 10 subsamples, we use 1 to simulate our new devices and the remaining 9 as our device database. We repeat this process 10 times with different subsamples of new devices.

On each iteration we verify which proprieties of new devices (1 subsample) match with these proprieties. The final output will be a table with the proprieties found and their matches with the devices of each subsample. The advantage of this technique applied in our work is that all the devices in the dataset are eventually used for both training and testing.

IX. Evaluation

In this chapter we will present the results achieved for each algorithm and evaluate them. We will differentiate two separate test cases:

· Communications from devices that already are in app device database.

· Communications from new devices that are not in app device database.

In first case we are testing new communication files from devices that we already gathered information before. So the communication file is new and may have a completely different set of properties than the last one found, however, he already have proprieties associated with the belonging device inside our database.

In the second case we test new communication files from devices that we never import to the database.

This differentiation is important because the results can differ between a device that we already imported data and a new device which we never gathered information before.

We will also present the results based on two different criteria: based on device and based on type.

On the first we will check which devices returns best results (most ”identical” devices or ideally the same device). On the second we show the matching results by device type.

To evaluate the performance of the algorithms used, the following values are used: the number of correct matches that are correctly classified (true positives (tp)), the number of incorrect matches classified as positive (false positives (fp)), the number of correct matches classified as negative (false negatives (fn)) and the number of incorrect matches classified as negative (true negatives (tn)). From this values it is possible to derive the following performance measures:

Precision (also called positive predictive value) measures the exactness of a classifier, it is the proportion of instances classified as positive that are really positive. If Precision = 1 then fp = 0 means that all positive matches detected are true positives.

$$Precision = \frac{tp}{tp+fp} \tag{2}$$

Recall (or sensitivity) measures the completeness of a classifier, it is the proportion of positive instances that are correctly classified as positive. If Recall = 1 then fn = 0 means that all positive matches were detected on the correct class.

$$Recall = \frac{tp}{tp+fn} \tag{3}$$

F-measure (or f1-score), is a measure that combines Precision and Recall. The closer to 1 the better.

$$F-measure = \frac{2 \times Precision \times Recall}{Precision+Recall} \tag{4}$$

A. Direct Match

We start the device discovery tests by running a direct comparison of the proprieties from the device to discover and the proprieties we have on our device database.

First we show the results obtained for devices that we already gathered information before (communication files imported to our test database). On **??** and Table III we can check the values obtained per device. There was a total of 1689 matching properties from all the devices discovered. The low total number of matching properties occur because we are using an exact match (both names must coincide exactly). The results per device are very good achieving 0.85 of precision and 0.80 on recall value.

Table I. Confusion Matrix for devices included on database by type (Direct Match)

Actual Classes	**Predicted Classes**							
	Lights	**Sensor**	**Activity**	**Controller**	**Display**	**Utility**	**Health**	**Security**
Lights	**114**	24	14	10	8	12	0	8
Sensor	15	**334**	6	1	2	16	9	1
Activity	0	0	**19**	0	0	7	0	0
Controller	0	0	0	**381**	0	0	0	0
Display	5	4	3	2	**10**	6	1	3
Utility	17	23	12	6	7	**227**	6	7
Health	0	13	11	11	0	12	**235**	0
Security	4	4	2	3	4	6	0	**64**

Table II. Classification Report for devices included on database by type (Direct Match)

Classes	**Precision**	**Recall**	**f-measure**	**Support**
Lights	0.74	0.60	0.66	190
Sensor	0.83	0.87	0.85	384
Activity	0.28	0.73	0.41	26
Controller	0.92	1.00	0.96	381
Display	0.32	0.29	0.31	34
Utility	0.79	0.74	0.77	305
Health	0.94	0.83	0.88	282
Security	0.77	0.74	0.75	87
Avg/Total	0.83	0.82	0.82	1689

Table III. Classification Report for devices included on database (Direct Match)

Classes	**Precision**	**Recall**	**f-measure**	**Support**
Ai prime 1	0.19	1.00	0.32	8
Automatic 2	0.23	0.58	0.33	31
Efergy 3	0.82	1.00	0.90	27
Fitbark 4	0.28	0.73	0.41	26
Flower 5	0.99	0.99	0.99	81
Harmony 6	0.97	1.00	0.98	357
Lametric 7	0.32	0.29	0.31	34
Lifx 8	0.83	0.52	0.64	182
Netatmo 9	0.94	0.80	0.86	245
Petnet 10	0.67	0.56	0.61	163
Porkfolio 11	0.88	0.92	0.90	142
Tado 12	0.53	1.00	0.70	24
Withings 13	0.94	0.83	0.88	282
Yi Cam 14	0.77	0.74	0.75	87
Avg/Total	0.85	0.80	0.81	1689

On Table I and Table II we present respectively, the Confusion Matrix and Classification Report for the direct match discovery by device type. In this case we consider as a positive match a propriety that has the same type of the device that is being identified. On these tests the precision slightly reduces to 0.82 (- 0.03) but the recall increase to 0.83 (+ 0.03). These high values are obtained mainly because in both situations, the correct device or type obtains the highest number of matching proprieties.

After testing devices from database we now tested the discovery on new devices also running a direct comparison. We can see the results of this test on Table IV. The values obtained are relatively poor with only 0.39 of precision and 0.24 on recall. There was a total of 903 matching between this new devices and the devices on database. We need make the matching process less restrict to try to obtain more similarities between the properties on new devices and our database. That is what we will try to accomplish with the next tests with the use of other techniques that allows some more flexibility in the matching process like the Levenshtein distance algorithm.

B. Levenshtein Algorithm

In this section we present the results of device discovery running the Levenshtein algorithm against each device. As explained on subsection VIII-B, after some trial and error tests we found the best results with a Levenshtein distance value of <4 (Table V).

The number of correct matches increases as the Levenshtein distance rises because it will discover more similar words on the correct properties. However, for distances higher then 3,

Table IV. CLASSIFICATION REPORT FOR DISCOVERY OF NEW DEVICES BY TYPE (DIRECT MATCH)

Classes	Precision	Recall	f-measure	Support
Light	0.37	0.22	0.28	309
Sensor	0.41	0.21	0.28	360
Activity	0.00	0.00	0.00	0
Controller	0.38	0.32	0.35	234
Display	0.00	0.00	0.00	0
Utility	0.00	0.00	0.00	0
Health	0.00	0.00	0.00	0
Security	0.00	0.00	0.00	0
Avg/Total	0.39	0.24	0.30	903

Table V. CLASSIFICATION REPORT FOR NEW DEVICES WITH DIFFERENT LEVENSHTEIN DISTANCE

Distance	Precision	Recall	f-measure	Support
<1	0.39	0.24	0.30	903
<2	0.41	0.25	0.31	1004
<3	0.44	0.28	0.34	1091
<4	0.46	0.31	0.37	1238
<5	0.41	0.29	0.33	1584

the incorrect matches starts to grow more quickly than the correct matches and therefore the precision and recall values start a decreasing curve.

Further results (omitted due to space constraints) show that the algorithm was able to match more proprieties with the correct device than not.

Table VI. CLASSIFICATION REPORT FOR DEVICES INCLUDED ON DATABASE (LEVENSHTEIN)

Classes	Precision	Recall	F-measure	Support
Ai prime	0.25	0.56	0.34	32
Automatic	0.12	0.53	0.19	34
Efergy	0.43	0.73	0.55	45
Fitbark	0.41	0.54	0.47	105
Flower	0.99	0.99	0.99	81
Harmony	0.84	0.72	0.78	495
Lametric	0.16	0.22	0.19	45
Lifx	0.69	0.47	0.56	200
Netatmo	0.84	0.65	0.71	314
Petnet	0.42	0.50	0.46	184
Porkfolio	0.74	0.78	0.76	167
Tado	0.47	0.57	0.52	82
Withings	0.84	0.60	0.70	422
Yi Cam	0.52	0.53	0.53	135
Avg/Total	0.70	0.63	0.65	2341

As we saw previously, the discovery obtains more matching proprieties on the correct device in all test cases. Table VI presents some statistical classification for the algorithm performance. The average precision value was 0.70 and the recall 0.63 giving a f-score of 0.65. Despite the relative good results, I think the most important information we acquire here is the fact of, in exception of two devices, we obtain more true positives than false positives (recall above 0.50 for each device). Another important note is that the positives in the correct device are always superior comparing to any other positives in each single device discovery. There was also a substantial increase on the total matching properties to 2341 comparing with the direct match test.

On Table VII and Table VIII we present the results of the discovery performance using the device type instead of the specific device. Once again we can verify more positives results on the correct type. The precision reduced slightly to 0.68 (-0.02) but the recall value rise to 0.66 (+0.03) leading to a better f-measure value of 0.67 (+0.02). In exception of the type ”Display” all the recall values are above 0.5. We notice that the worst recall results (”Display”, ”Activity” and ”Security”) appear on types that only contained on device of it’s kind in database, this can indicate that, with more devices on test database, the results on type discovery could be better.

Table VII. CONFUSION MATRIX FOR DEVICES INCLUDED ON DATABASE BY TYPE (LEVENSHTEIN)

Actual Classes	Predicted Classes							
	Lights	**Sensor**	**Activity**	**Controller**	**Display**	**Utility**	**Health**	**Security**
Lights	**128**	30	14	23	11	17	1	8
Sensor	17	**376**	6	18	3	22	11	21
Activity	6	10	**57**	8	4	12	5	3
Controller	27	25	12	**426**	11	33	20	23
Display	6	7	4	6	**10**	7	1	4
Utility	20	48	14	17	8	**227**	10	7
Health	0	45	31	11	10	72	**253**	0
Security	6	29	2	14	6	6	0	**72**

Table VIII. CLASSIFICATION REPORT FOR DEVICES INCLUDED ON DATABASE BY TYPE (LEVENSHTEIN)

Classes	Precision	Recall	f-measure	Support
Lights	0.61	0.55	0.58	232
Sensor	0.66	0.79	0.72	474
Activity	0.41	0.54	0.47	105
Controller	0.81	0.74	0.77	577
Display	0.16	0.22	0.19	45
Utility	0.57	0.65	0.61	351
Health	0.84	0.60	0.70	422
Security	0.52	0.53	0.53	135
Avg/Total	0.68	0.66	0.67	2341

Until now, we tested the discovery with devices that we know and already gathered communication files from them. However, when a new device is released to the market we don’t have any information about it and we never gathered communication files to our devices database before. To simulate this situation we tested the algorithm performance with 5 new devices. On **??** we can see the matching results for each one of the 5 new devices against the devices we already have on database. Ideally the matching properties should be the more ”equivalent” devices (of the same type). A positive match for the new device ”Philips Hue” (Light type) should be ”Ai Prime” or ”Lifx” that are also Light type devices. In our tests, all the new devices except one, have more matching properties on a device of the same type. The exception device was ”Nest” that is a controller and have more properties matching (59) with ”Netatmo” that is a sensor device. In this case, the second device with more matching properties (55) is the controller type ”Tado” that is the most ”equivalent” device to ”Nest”. Notice that ”Nest”, although it is definitely a controller type, has also a sensor to measure the ambient temperature that is main function of ”Netatmo”.

On Table IX we present the Confusion Matrix for this discovery by type. Here we can see the matching proprieties by type and verify that the most matching properties are contained in the correct type.

Table IX. CONFUSION MATRIX FOR NEW DEVICES BY TYPE (LEVENSHTEIN)

Actual Classes	Predicted Classes							
	Light	Sensor	Activity	Controller	Display	Utility	Health	Security
Light	**105**	62	53	75	30	48	16	31
Sensor	98	**129**	38	60	4	63	51	5
Activity	0	0	0	0	0	0	0	0
Controller	49	94	8	**155**	0	56	7	1
Display	0	0	0	0	0	0	0	0
Utility	0	0	0	0	0	0	0	0
Health	0	0	0	0	0	0	0	0
Security	0	0	0	0	0	0	0	0

Table X. CLASSIFICATION REPORT FOR NEW DEVICES BY TYPE (LEVENSHTEIN)

Classes	Precision	Recall	f-measure	Support
Light	0.42	0.25	0.31	420
Sensor	0.45	0.29	0.35	448
Activity	0.00	0.00	0.00	0
Controller	0.53	0.42	0.47	370
Display	0.00	0.00	0.00	0
Utility	0.00	0.00	0.00	0
Health	0.00	0.00	0.00	0
Security	0.00	0.00	0.00	0
Avg/Total	0.46	0.31	0.37	1238

Classification report for the same tests are presented on Table X. All the values dropped considerably comparing with the values the discovery of devices on database. However, in my opinion, it's a good sign to verify that the discovery successfully matched with the correct type. This is the main purpose here because we cant discover what is the specific device (only the most "equivalent"), but we can identify the specific type of the new device.

C. TF-IDF Table

We will now present the results of the application of Tf-Idf table in device discovery. As explained on subsection VIII-A, first we need to generate the Tf-idf tables for each type. As intermediate result we will have one table for each type. Each table have a list of keywords and their respective value calculated with Tf-Idf algorithm, for simplicity we also add a column with the value discretized to 0, 1. We can see on Table XI the top 5 output keywords for the algorithm applied to type "Light". The keywords "intensity" and "color" are very good results and can help improving the discovery precision, however "time", "luid" and "points" are generic words and not specific to "Light" type. The word "intensity" receive an higher value because the keyword "intensity" don't exist in any document of other device type. In this case, if the discovery algorithm find a match with "intensity", it will increment 0.28 points on type "Light".

After all the tables are generated with the respective discretized values calculated, we can now match the properties of the device to discover. Since our keyword tables are based on device type, the results are only based on type and not on specific device. On Table XII and Table XIII we summarize the results obtained from devices of our database and new devices (respectively).

Table XI. TF-IDF KEYWORK TABLE FOR TYPE "LIGHT"

Keyword	Rate	Discretized Value
intensity	60	0.281
time	30	0.141
luid	8	0.037
points	7	0.033
color	4.5	0.021

Table XII. CLASSIFICATION REPORT FOR DEVICES ON DATABASE (TF-IDF)

	Precision	Recall	f-measure
Average Result	0.70	0.67	0.66

Table XIII. CLASSIFICATION REPORT FOR NEW DEVICES (TF-IDF)

	Precision	Recall	f-measure
Average Result	0.42	0.32	0.32

The discovery of devices in our database reach 0.70 for precision and 0.67 for recall. Although lower than the obtained on Levenshtein algorithm, the results were satisfactory. The same can not be said for the new devices discovery results, with a precision of 0.42 and recall 0.32. The main reason of this lower values is the fact that some keywords more generic (not specific to a type like "name") should have lower values. With a bigger database, this kind of words will have more chances to appear on communication files from other device types, and thereby achieve a lower value.

D. Synonyms Match

In this section we present the results of device discovery using a dictionary of synonyms. As explained on subsection VIII-C, we try to improve our results introducing synonyms on the matching algorithm. This way, we will try to match not only each property but also all the synonyms related to each property. On Table XIV we present the results for the devices on database. As expected, the Support number increases and we were able to match more 170 proprieties (synonyms) in this test comparing with the results in subsection VIII-B. Unfortunately this new matching values not only found the same type devices but also other ones (not the correct device). This leads to a drop on precision and recall values to 0.66 and 0.64 respectively (difference of - 0.02 in both cases comparing with Levenshtein Algorithm).

Table XIV. CLASSIFICATION REPORT FOR DEVICES ON DATABASE BY TYPE (SYNONYMS)

Classes	Precision	Recall	F-measure	Support
Avg/Total	0.66	0.64	0.64	2511

Regarding the tests for new devices, we can say that the results were better. As we can see on Table XV, similarly to the devices on database, the support value also raised to 1287 (more 49 matched proprieties). However, the matched synonyms words were more assertive on the correct devices types providing an increase of 0.02 on precision and recall comparing to Levenshtein tests.

Table XV. Classification Report for new devices by type (Synonyms)

Classes	**Precision**	**Recall**	**F-measure**	**Support**
Light	0.46	0.29	0.35	441
Sensor	0.47	0.30	0.36	476
Activity	0.00	0.00	0.00	0
Controller	0.52	0.42	0.46	370
Display	0.00	0.00	0.00	0
Utility	0.00	0.00	0.00	0
Health	0.00	0.00	0.00	0
Security	0.00	0.00	0.00	0
Avg/Total	0.48	0.33	0.39	1287

E. Multi-Property Matching

Our last tests was made using the Multi-Property matching technique. As detailed on Table XVI, with this method we get the advantage of use the most specific proprieties of each device type and the possibility of use all devices for both training (device database) and testing (new device discovery).

Table XVI. Classification Report for discovery of new devices (Multi-Property)

Classes	**Precision**	**Recall**	**f1-score**	**Support**
Light	0.93	0.93	0.93	15
Sensor	0.91	0.91	0.91	22
Controller	0.90	0.86	0.88	21
Utility	0.88	1.00	0.93	7
Activity	0.92	0.80	0.86	15
Display	0.88	0.88	0.88	8
Health	0.78	1.00	0.88	7
Security	1.00	1.00	1.00	7
Avg/Total	0.91	0.90	0.90	102

Table XVII. Confusion Matrix for discovery of new devices (Multi-Property)

Classes	**Light**	**Sensor**	**Controller**	**Utility**	**Activity**	**Display**	**Health**	**Security**
Light	**14**	0	0	0	0	1	0	0
Sensor	0	**20**	2	0	0	0	0	0
Controller	1	2	**18**	0	0	0	0	0
Utility	0	0	0	**7**	0	0	0	0
Activity	0	0	0	1	**12**	0	2	0
Display	0	0	0	0	1	**7**	0	0
Health	0	0	0	0	0	0	**7**	0
Security	0	0	0	0	0	0	0	**7**

With the Multi-Property matching method we reach the best result of all the methods tested in this work. The test obtained an average precision of 0.91 and 0.90 of recall. Besides the 10 iterations made possible by this technique, the main factor for this result was the selection of the most specific proprieties for each device/type. Thanks to this detail, it was possible to reach a smaller number of False Positives and thus, achieve higher values.

X. Conclusion

A. Final Remarks

In 2020, 5 billion devices are expected to be in operation worldwide. Due to it's exponential growth, IoT is facing many challenges. One of these challenges is the decentralization of the existent implementations and platforms. Each company uses their own way to integrate the devices and there is no standard. So there are a mass of separated technologies and devices, but there are few integrated services. Therefore, a holistic design implementation is demanded to effectively integrate the scattered devices and technologies into more valuable services. This work attempted to provide an holistic view on the devices discovery and automatic integration phases reducing the existent decentralization effect. This automatic integration process relies on an effective way to identify the device that is exchanging communication data.

We try to prove the possibility to identify an IoT device type using the communication data exchanged between them. In this paper we detailed the method used to prove this possibility. We gathered the necessary communication data by listening to network traffic containing some of these devices and using a method named man-in-the-middle. This data was then used to generate a device information database that stores the proprieties of each device. Finally, using this database, we tried to correctly identify devices that are communicating inside a network, testing some techniques like the Levenshtein Distance algorithm, TF-IDF tables, Synonyms Match and finally Multi-Property matching. We differentiate the tests made for devices with some info already imported to our database from the completely new devices with no information gathered yet.

To make these tests we also developed a mobile application as a prototype. This prototype is capable to generate and manage a device information database by importing the communication files originated by IoT devices. This application is also capable to run the algorithms discussed in this work and to return the results for each one.

We have presented the result for each of the tests mentioned above, which are all property-based methods. This approach was chosen due to the nature of our data, rich in properties but poor on devices (roughly 20) due to the encrypted communication formats used by many devices. Of course, having more information on each device should entail better results, so we run a multi-property matching technique over a database generated with the TF-IDF algorithm containing specific proprieties by type. The final results were good (precision of 0.91) and we can verify that, with correctly valued TF-IDF tables (type specific proprieties), it is possible to identify the device type with good precision.

With the growing number of IoT devices on the market, an efficient integration is becoming more and more important. We think this results are promising and demonstrate that the use of communication data from these devices can be used to reach this objective.

B. Future Work

With the devices used in our work we were able to gather a total of 17547 properties. Although a considerable number of properties, it would be interesting to run these tests over bigger databases to get better results, namely on generation of the TF-IDF tables.

On the device data import phase we were unable to get information from some devices. The information was received encrypted even with the SSL certificate installed on the man-in-the-middle machine. Until the finish date of this thesis, we were unable to discover the reason and it will be important as future work to overcome this barrier.

Regarding our mobile application prototype, it can be improved in some areas. The design can be improved to be easier to receive the communication data files. The import data phase can be enhanced by developing a method to receive many files all at once (currently it can only receive one at a time). Finally a method should be developed to make a search over the properties database.

References

[1] E. Fleisch, "What is the internet of things? An economic perspective," *Economics, Management, and Financial Markets*, no. 2, pp. 125–157, 2010.

[2] I. I T U, "Recommendation ITU-T Y.2060," 2012. [Online]. Available: http://handle.itu.int/11.1002/1000/11559

[3] C. Systems, "Internet of Things (IoT) in Real World," 2012. [Online]. Available: https://learningnetwork.cisco.com/community/it{_}careers/internet-of-things-webinar-series

[4] K. Ashton, "That 'internet of things' thing," *RFiD Journal*, vol. 22, no. 7, pp. 97–114, 2009.

[5] J. Haartsen, "Bluetooth-the universal radio interface for ad hoc, wireless connectivity," *Ericsson review*, vol. 3, no. 1, pp. 110–117, 1998.

[6] P. Kinney *et al.*, "Zigbee technology: Wireless control that simply works," in *Communications design conference*, vol. 2, 2003, pp. 1–7.

[7] A. Alliance, "AllJoyn Proximal Connectivity Platform," 2015. [Online]. Available: https://developer.qualcomm.com/software/alljoyn

[8] C. Reinisch, W. Kastner, G. Neugschwandtner, and W. Granzer, "Wireless technologies in home and building automation," in *2007 5th IEEE International Conference on Industrial Informatics*, vol. 1. IEEE, 2007, pp. 93–98.

[9] Z.-W. Alliance, "Z-Wave Alliance History," 2015. [Online]. Available: http://z-wavealliance.org/z-wave_alliance_history/

[10] P. CarlSSon, "Multi-timescale modelling of ethernet traffic," 2003.

[11] J.-S. Lee, Y.-W. Su, and C.-C. Shen, "A comparative study of wireless protocols: Bluetooth, uwb, zigbee, and wi-fi," in *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE*. IEEE, 2007, pp. 46–51.

[12] J. Liikanen, P. Stoneman, and O. Toivanen, "Intergenerational effects in the diffusion of new technology: the case of mobile phones," *International Journal of Industrial Organization*, vol. 22, no. 8, pp. 1137–1154, 2004.

[13] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012, pp. 225–238.

[14] K. Curran, A. Millar, and C. Mc Garvey, "Near field communication," *International Journal of Electrical and Computer Engineering*, vol. 2, no. 3, p. 371, 2012.

[15] Wireshark, "Wireshark software," 2016. [Online]. Available: https://www.wireshark.org

[16] S.-C. Arseni, S. Halunga, O. Fratu, A. Vulpe, and G. Suciu, "Analysis of the security solutions implemented in current internet of things platforms," in *Grid, Cloud & High Performance Computing in Science (ROLCG), 2015 Conference*. IEEE, 2015, pp. 1–4.

[17] L. Foundation, "Iotivity architecture overview," 2016. [Online]. Available: https://www.iotivity.org/documentation/architecture-overview

[18] E. Nimmermark and A. Larsson, "Comparison of iot frameworks for the smart home," 2016.

[19] H. Hexmoor, "Compelling use cases for the internet of things," in *Proceedings on the International Conference on Internet Computing (ICOMP)*. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2016, p. 20.

[20] Muzzley, "Muzzley," 2016. [Online]. Available: https://www.muzzley.com/aboutus

[21] Xamarin, "Xamarin platform," 2016. [Online]. Available: https://www.xamarin.com/platform

[22] Bitbucket, "Bitbucket thesis app fork," 2016. [Online]. Available: https://bitbucket.org/pedroruben9/iot-finder/get/5e0f239dc562.zip

[23] J. Ramos, "Using tf-idf to determine word relevance in document queries," in *Proceedings of the first instructional conference on machine learning*, 2003.

[24] K. Beijering, C. Gooskens, and W. Heeringa, "Predicting intelligibility and perceived linguistic distances by means of the levenshtein algorithm," *Linguistics in the Netherlands*, vol. 15, pp. 13–24, 2008.

[25] Altervista, "Altervista thesaurus api," 2016. [Online]. Available: http://thesaurus.altervista.org

[26] Gutenberg, "Gutenberg thesaurus," 2016. [Online]. Available: http://www.gutenberg.org/ebooks/10681

[27] J. D. Rodriguez, A. Perez, and J. A. Lozano, "Sensitivity analysis of k-fold cross validation in prediction error estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 3, pp. 569–575, 2010.