Internet of Things for the Smart Home

António Gonçalves Vian Costa
Instituto Superior Técnico - Universidade de Lisboa
Email: antonio.v.costa@tecnico.ulisboa.pt

Abstract—As increasingly more things become connected to the internet, the Internet of Things is becoming a reality faster than expected. Smart homes, a setting in which the Internet of Things is predicted to have a high impact on the betterment of everyday lives, is a topic of high interest. Allied to other technological advances like cloud computing, machine learning, and miniaturization, the Internet of Things is turning devices away from performing taxing computing skills. Here, low power wide area networks are vital, providing energy efficient devices at low costs. Sigfox and LoRa are two significant players in this field. In this work a prototype using Sigfox technology was developed for the smart home and a study was made on possible consumer interaction solutions, using various Internet of Things web platforms. Another objective, to compare the Sigfox prototype with LoRa technology, is also achieved. It is concluded that Sigfox, because of its limitations, cannot be the only technology used in a smart home. LoRa technology, while offering more freedom of use, has a more difficult and complex implementation and is still behind on the simplicity offered by the Sigfox infrastructure.

Keywords—Internet of Things, Smart Home, LPWAN, Sigfox, LoRa, IoT Platforms.

I. INTRODUCTION

THE Internet changed the way people live their life’s. Now the next big revolution is coming in the form of the Internet of Things (IoT). However, its meaning and definition are still in a fog. It is related to many other technologies, such as Machine-to-Machine (M2M) communications, cloud computing and big data.

Smart home, a growing setting in IoT, is a living environment filled with advanced automatic computer systems [1], with a function to improve the quality of life of its inhabitants. It faces several challenges from a user point of view, such as [2]: security; ease of use; cost; energy efficiency; and privacy.

The main objective of this work is to develop a smart home system using LPWAN IoT technologies. To achieve this, specific objectives were defined.

To study IoT, its challenges and technologies with a focus on LPWAN.

To analyze the two most significant LPWAN technologies in the market, Sigfox and LoRa.

To study the various IoT web platforms that have a focus on the smart home setting.

To create a prototype using Sigfox and to synchronize it with the IoT platforms chosen.

To create a prototype using LoRa and also synchronize it with IoT platforms.

And finally, to compare the technologies and analyze the results from both experiences.

II. INTERNET OF THINGS

A. Defining IoT

There is not a clear consensus on what IoT is. According to the International Telecommunication Union it is “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 (ICT)” [3]. Figure 1 shows a technical overview of the IoT.

In IoT, the things can be physical or virtual. The physical things are objects which can be sensed, actuated and connected, such as people, buildings and plants. The virtual things are movies, books and application software, which can be stored, processed and accessed.

The physical things are connected to devices which are hardware equipment capable of communication and, if necessary, with sensing, actuation, data capture, data storage and data processing capabilities. The devices can communicate between themselves in three ways: through communication networks; without communication networks; or with gateways.

There is not a clear consensus on how to segment IoT applications. In [4] the IoT is segmented into nine settings. They are: human, home, retail, offices, factories, worksites, vehicles, cities, urban environments and outside urban environments settings. This segmentation was proposed as a better alternative to the categorization in vertical industry markets. For example, in a city setting sensors in cars can be used by an automaker to determine when a vehicle needs maintenance, but they can also be used to help manage traffic congestions, maximizing the use of smart devices.

Figure 1. Technical overview of the IoT [3].
B. IoT Reference Model

There is no standard architecture for IoT. The ITU proposed reference model for IoT is the one that better applies to a higher number of solutions, due to its level of abstraction. Figure 2 presents this architecture consisting of four layers: the device, the network, the service and application support and the applications layer. Present in all these layers are management and security capabilities [3].

Figure 2. IoT reference model [5].

The device layer is composed by devices and gateways. The devices identify, gather and upload information from the environment or from the thing they are connected to. Gateways are protocol converters that are needed when communications at the device layer differ from the ones used at the network layer. At the device layer wired or wireless technologies such as ZigBee, Bluetooth, Wi-Fi, Sigfox, LoRa, 2G, 3G, 4G and Ethernet can be used.

The network layer is responsible for the transport and networking capabilities. The networking capabilities connect things to networks and maintain the connectivity. The transport capabilities are in charge of assuring the transport of the IoT application data. The network layer is capable of supporting network layer protocols such as IPv4 and IPv6, cellular networks (2G, 3G, 4G) and various technologies such as Ethernet, Wi-Fi and Digital Subscriber Lines (DSL) [5].

The service and application support layer is responsible for generic support capabilities that can be used for different IoT applications like data processing or data storage as well as specific capabilities that are catered to specific applications. The application layer, is responsible for the interaction methods with users or applications. This layer contains the IoT applications, such as smart home, e-health, smart-grids, etc.

The security and management capabilities are divided into general and specific capabilities. In security, the generic capabilities include authorization, authentication, privacy protection and data confidentiality. The specific capabilities depend on the IoT application and can be, for example, tighter security requirements. In management, the generic capabilities include functions such as device management, for example, remote device activation and deactivation. Specific capabilities are normally application specific.

C. Cloud Computing

Cloud computing is a way to store, manage and process data in a network server, which is hosted on the Internet. There are three main drivers for cloud usage in IoT [6]:

Communication: the cloud offers low cost solutions to connect, track, and manage any thing from anywhere at any time by using web portals and specific applications.

Storage: cloud offers the most convenient and cost effective solution to deal with the data produced by IoT. It enables new opportunities for data aggregation, integration, and sharing with third parties.

Computation: with virtually unlimited processing capabilities, performing real-time data analysis to implement scalable, real-time sensor-centric applications, for managing complex events and for supporting task offloading for energy saving, becomes possible for IoT solutions.

IoT cloud computing solutions bring some challenges as well [7], such as: the cost of transferring and storing data; internet access, which may be unavailable, unreliable and slow; the possibility of Internet attacks; and others.

Fog computing, solves most of these problems. It acts as a middle cloud between the devices and the cloud. Some of the most important characteristics of fog computing solutions are [8]: fast transmission speeds; wide-spread geographical distribution; mobility; and a very large number of nodes. Examples of devices which may take advantage of fog computing are generally time sensitive.

D. Standards

There are several organizations currently working on IoT standards, such as: the International Organization for Standardization (ISO) [9], and the International Telecommunication Union (ITU) [10]. A table with all organizations working on IoT standards and their standardization activities can be found in [11].

ISO has fifteen published standards focusing on IoT related technologies with only one relating to IoT use cases. Nine others are currently under development with the two specific to IoT. One on IoT definition and vocabulary and the the other on the IoT reference architecture.

ITU was found to only have e-health IoT related standards, and a recommendation specific to IoT technology. The recommendation gives an overview on the technology, its scope and reference model [3].

E. Privacy and Legal Issues

Regarding privacy, reports made by government bodies, address some of the challenges that IoT brings. In [12], aspects
of the European Union protection laws, in regards to IoT, are discussed. Under EU law, users of IoT devices will qualify as data subjects. This means that if a device possesses personal information about that user, the individual in question will be classified as a data subject even if the device does not belong to them. Stakeholders, which are classified as data controllers, are responsible for processing and for the purpose of personal data. Examples of stakeholders include device manufacturers, social media platforms, and third-party application developers.

The fairness of data handling and processing is also addressed in [12]. Aspects like purpose limitations, which state that the data collected must have specific, explicit and legitimate proposes are addressed. Furthermore, data minimization aspects are mentioned as well, which discuss how the data should only be collected for the necessary use, and not stored by default.

F. Security

There are an extensive number of IoT security attacks, which can be categorized by type. They are [13]: physical attacks, network attacks, software attacks and encryption attacks.

In the physical attacks, the malicious code injection attack is when a malicious code is uploaded to devices or gateways that contains a virus. Or when an already infected device is added to the network. The affected device then sends the wrong information to other nodes in the network and corrupts them. The attacker may gain access to the network and even cause the service to become unavailable. Defenses to this type of attack are secure authentication and booting and intrusion detection technology.

In the network attacks, the sinkhole attack is when a node is compromised. The node then fakes the routing information so that the other nodes on the network think the compromised node is the closest to a base station or gateway, redirecting the traffic to it, which can result in altered or even dropped data. Defenses include encryption and secure authentication.

Viruses, worms, spyware and adware are attacks made with malicious code, capable of replicating itself, and are normally transmitted by emails or by downloading files from the Internet. Defenses against these types of attacks include anti-virus, anti-spyware and anti-adware software as well as advanced encryption and web application firewalls.

In encryption attacks, the side-channel attack uses cryptotokens, which are a series of cryptographic algorithms most commonly used in encryption, to gain information about the system such as power management, partial or full plaintexts or even the cryptographic key. There are several types of side-channel attacks, one of them is the timing attack, which measures how much time computations take to perform. Defenses against this type of attack focus on advanced encryption techniques.

III. COMMUNICATION TECHNOLOGIES FOR THE IOT

A. Wireless Technologies

There are various wireless technologies which can be used in IoT projects. They can be grouped from: short range, such as Wi-Fi and Zigbee; to long range cellular, such as, LTE-M and NB-IoT; and long range LPWAN, such as, Sigfox and LoRa.

Wi-Fi operates in the 2.4 GHz or 5 GHz band and can handle speeds from 300 Mbps to 1.3 Gbps, with ranges up to 100 meters. A new Wi-Fi protocol for IoT solutions, nicknamed HaLow, has a data rate of 150 kbps to 345 Mbps and can reach up to a kilometer.

Zigbee operates in the 2.4 GHz and sub-GHz bands. Data rates in the 2.4 GHz band are 250 kbps, and in the sub-GHz band are 20 kbps. It has a reach of 100 meters in line-of-sight conditions [14].

LTE-M can be used to refer to all LTE based technologies for IoT. LTE Cat-0 and Cat-1 solutions use licensed frequency bands, offer data rates of 1 Mbps and 10 Mbps, respectively, with battery life up to ten years [15].

NB-IoT uses 14 different frequency bands with 10 of them in the sub-GHz range [16]. Data rates range from 100 kbps to 1 Mbps, with battery life close to ten years.

B. Low Power Wide Area Networks

LPWANs are technologies which focus on three main aspects: low power; long range; and low data rates [17]. Most LPWANs operate in the sub-GHz band.

Currently there is no adopted standard available in the LPWAN market [18] and they are usually defined by their radio technology [19]: Ultra-Narrow Band; Narrowband; and Wideband.

1) Sigfox: Sigfox is a UNB technology. In Europe uses 192 kHz, in the 868-868.2 MHz band and in the rest of the world it uses the 902-928 MHz band [20].

Messages can be bi-directional. The uplink uses BPSK modulation and has a data rate of 100 bps in Europe and 600 bps in America. Each message is 0 to 12 bytes long. Only 140 messages are allowed per day, per device, because of Europe regulations for the band. It has ranges from 30 to 50 km in the countryside and 3 to 10 km in urban settings, with battery life usually in the 10 year range.

The downlink uses G-FSK modulation with a data rate of 600 bps. Downlink messages are static 8 bytes and only 4, per device, per day are permitted. Downlink messages can only be initiated by the devices following an uplink message.

Figure 3 presents the Sigfox architecture, which consists of devices, base stations, the Sigfox cloud and the interfaces used to access the information on it.

Figure 3. The Sigfox architecture.
responsible for detection, demodulation and reporting the messages to the cloud. Each base station has a direct point-to-point connection with the Sigfox cloud, through the public internet, with the use of a VPN. This connection uses DSL connectivity, 3G or 4G as a back-up, and when neither is available, satellite connectivity can be used as an alternative [20].

The Sigfox cloud is responsible for the processing of the messages sent from the base stations. It is in the cloud that the status of the network is monitored and the base stations managed globally. The cloud also has the tools to analyze the data collected or generated by the network. There are three ways to interact with the Sigfox cloud: with the Internet; with APIs; and with callbacks.

2) LoRa: LoRa (LOng RAnge) has two distinct aspects: the LoRa physical layer; and a MAC layer protocol called LoRaWAN. The latter is an open standard communication protocol, and the proposed MAC layer defined by the LoRa Alliance which drives the advancement of LoRa and the LoRaWAN protocol [21]. LoRa can also use other MAC layers protocols.

LoRa is a proprietary technology and a type of modulation based on CSS [21]. In the USA it uses the 915 MHz band, in Europe the 868 MHz band, and in Asia the 470 MHz band. A LoRa radio has four configuration parameters, which are [22]: carrier frequency, spreading factor, bandwidth, and coding rate.

LoRaWAN can use channels with a bandwidth of either 125, 250 or 500 kHz. The data rate ranges from 250 bps to 27 kbps. It supports data payloads between 19 to 250 bytes and uses a protocol overhead of 12 bytes [21]. It has ranges from 2 to 5 km in urban areas, and 15 to 45 km in rural areas, with battery life in the 10 year range.

LoRaWAN defines three types of classes for devices: class A, class B and class C.

Class A supports bi-directional communication. The devices send data at any time and wait for acknowledgment, some type of command, or a data packet from the network server. If nothing is received, the next opportunity will be after the next uplink transmission.

Class B allows for a bi-directional link with scheduled listening time-slots. Devices synchronize with the network server by means of beacon packets which are broadcasted on a regular basis by Class B gateways. The devices can then receive downlink data, or command packets in specific time windows, irrespective of the uplink traffic.

Class C allows the device to be always listening for data packets except during transmission.

Figure 4 presents the LoRa architecture. LoRa uses star topology, where the gateways relay messages between devices and a network server.

The communication between the devices and the gateways uses different spread out frequency channels and data rates. A standard LoRa device can decode only one spreading factor modulation over a single frequency [22]. The gateways forward the payload information to their associated network server and act as a bridge, totally transparent to the end devices. They are connected to the network server through Ethernet, 3G, 4G, Wi-Fi or satellite connection. At the base of all gateways sits a LoRa concentrator, which is a multi-channel demodulator, able to decode all versions of LoRa modulation, on several frequencies at the same time [23].

The network server is responsible for decoding the messages, performing security checks, filtering duplicate and unwanted packets and for replying to the end devices by choosing one of the gateways in range. It also enables remote management of gateway modules and device level management [22].

3) Differences between Sigfox and LoRa: Table I presents some the differences between the most important technical aspects of both technologies. As LoRa only represents the physical layer, the differences to Sigfox were made with LoRa using LoRaWAN.

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Sigfox</th>
<th>LoRaWAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>868 MHz</td>
<td>902-928 MHz</td>
<td>868 MHz</td>
</tr>
<tr>
<td>915 MHz</td>
<td>470 MHz</td>
<td></td>
</tr>
</tbody>
</table>

| Range | 3-10 km (urban) | 2-5 km (urban) |
| 30-50 km (rural) | 15-45 km (rural) |

| Data Rates | 100 bps | 250bps-27 kbps |
| 600 bps |

| Topology | STAR |
| CSS |

| Modulation | DBPSK (Up) | CSS |
| GFSK (Down) |

| Payload | 12 Bytes (Up) | 19-250 Bytes |
| 8 Bytes (Down) |

| Power levels | 0 - 14 dBm | 0 - 14 dBm |
| Battery Life | 10 years | 10 years |

| Messages/Day | 140 (Up) | Unlimited |
| 4 (Down) |

| Bandwidth | 100 Hz (Up) | 125, 250, 500 kHz |
| 1.5 kHz (Down) |

Sigfox has a message limit while LoRa does not. Therefore, Sigfox is better suited for applications that do not require constant exchange of messages, but only a handful of messages per day. LoRa has more leverage in the amount of messages it can send daily.

Another difference is in the base stations from Sigfox and the gateways from LoRa. Sigfox deploys and manages its own base stations, while LoRa gateways can be bought or made. They also differ in their business model. With Sigfox, the
company owns all of its technology from the back-end data and cloud servers to the device software. Sigfox partners with other companies in each country to build and operate the network infrastructure, with existing cell towers and placements being used to minimize cost. Additionally, only one Sigfox network can be deployed in an area, because the company has exclusive arrangements with its network operators [24].

LoRa uses a different strategy. Semtech leaves the development of network products and services to third party companies. Just like SigFox, the LoRa Alliance wants network operators to deploy the LoRa network, but they also want private companies and start-ups to do it.

Regarding devices, both technologies also differ. Sigfox does not design any hardware, opting instead to share their design and license its software to vendors. LoRa technology is free to use, but Semtech is the sole company that designs and fabricates the LoRa transceiver chips for the end devices, as well as the concentrator chips for the gateways.

C. State of the Art

Sigfox offers transceivers, system-on-chip, modules, devices and developer kits. Five transceivers are currently available from Texas Instruments and Silicon Labs all capable of working in the 433, 868 and 902 MHz frequency bands. The prices vary between the 0.5 to 5 €range. Also available are 10 system-on-chip, 54 modules, 374 devices and 66 developer kits [25].

For LoRa, Semtech currently has five transceivers available, and they all differ in the frequencies offered. These devices are currently all within the 4 to 10 €price range. Also available are two chips for LoRa gateways [26].

IV. SMART HOME PROJECT

A. Smart Home Architecture

There were two possible ways considered to implement a smart home IoT architecture solution. One made use of cloud services that could act as IoT platforms as well. The other redirected the messages from the cloud services to third party IoT platforms for better visualization and management.

B. Sigfox Solution

For the Sigfox smart home solution, the architecture designed for the work is presented in figure 5. To achieve it, several steps were taken.

First a prototype would need to be created which would send the information to the Sigfox back-end. Second the Sigfox cloud would send that information to an IoT platform where it could be accessed and managed.

For the Sigfox solution three developer kits were used: the TD1208 Evaluation Board Kit [27], the SmartEverything [28] evaluation board, and finally an Arduino UNO R3 equipped with a shield and Sigfox module from Cooking Hacks [29].

1) TD1208 Evaluation Board: This kit was used to test the Sigfox network and the first IoT platform. The board works by sending Sigfox messages with the use of AT commands. These are simple instructions used to communicate with a modem.

A Sigfox message was sent to confirm if the connection with the service worked. The first custom callback feature was created which redirected the Sigfox message to an email client. A second data callback was made to the Carriots platform [30].

2) SmartEverything Board: The SmartEverything evaluation board comes with a Sigfox module, has Bluetooth and NFC connectivity, a GPS receiver, a cryptography authentication chip, and a series of sensors. The board comes with a one year subscription to the Sigfox network.

All sensors of the board were tested. An Arduino sketch was used to test the connection with the Sigfox cloud. Other tests were conducted by using sketches available for the board. Further tests with this board were not possible because the token validity expired.

3) Sigfox Radio Shield for Arduino: The Arduino UNO R3 was equipped with a shield and Sigfox module from Cooking Hacks. It was the device used as the prototype for the smart home. Several sensor were tested with it, such as: two temperature and humidity sensors; two rain detection sensors; a moisture sensor; two PIR motion sensors; another temperature and humidity sensor; and an adapter board designed for the evaluation of an accelerometer. All sensors were tested with the Arduino.

The sensor chosen for the smart home system was the SHT31 temperature and humidity sensor. It was found that when the Sigfox module was connect to the shield sketches could not be uploaded. So for the upload the Sigfox module had to be disconnected.

A sketch was created which only used the loop function of the Arduino. This measure was taken to prevent errors from disconnecting and connecting the Sigfox module to upload sketches. The sketch first initializes the temperature sensor, followed by the Sigfox module; second, the temperature and humidity values are retrieved and sent in a 8 byte payload, 4 bytes for the temperature and 4 for the humidity; and third, a delay of twenty minutes is issued. Figure 6 shows the message received at the Sigfox back-end with a custom payload display for the temperature and humidity.

The prototype was also tested with a sleep library called “Narcoleptic” [31] to test the amount of time the device would
remain active only with batteries. A test was conducted with an 9V battery and an eight 1.5V AA battery holder.

<table>
<thead>
<tr>
<th>Time</th>
<th>Data / Decoding</th>
<th>Location</th>
<th>Link quality</th>
<th>Callbacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018-04-09 02:07:30</td>
<td>41BF277042949521c</td>
<td>TEMP: 17.894/137</td>
<td><img src="image" alt="Signal Strength" /></td>
<td><img src="image" alt="Callback" /></td>
</tr>
</tbody>
</table>

Figure 6. Example of a Sigfox message at the back-end.

For this module a new email callback was created, and a downlink test was conducted which had a payload containing the time and RSSI values. Another sketch which made use of the motion sensor was also created.

4) Sigfox IoT Platforms: To complete the smart home architecture defined in figure 5 several platforms were tested: the Cloudthing [32]; the Tago [34]; the Thinger [35]; the Databoom [36]; the Amazon AWS [37]; and the Microsoft Azure [38].

To connect to the Thinger platform two steps were taken at the platform: the creation of a data bucket, which stores the received information from Sigfox; and generation of an access token, necessary for the data to be accepted by the platform.

Then the Sigfox callback was made. It used a custom payload information which parsed temperature and humidity. The callback was made with the HTTP method in POST required the URL pattern, and access token obtained at the platform. The message, in JSON (JavaScript Object Notation) format, contained the information for the device ID, RSSI, SNR, latitude and longitude of the base station, temperature, humidity and the time at which the message was sent. At the platform a dashboard was created.

Another platform tested was the Amazon AWS. This platform uses a simplified bridge callback, dedicated to its service. The platform configuration had three steps. The first was to create the Sigfox callback choosing AWS IoT option. The second was to create a stack in the AWS Cloud Formation console, where several parameters to enable the Sigfox message to be received by the AWS cloud are generated. The third was to create an AWS IoT Rule, responsible for storing the data into an Amazon DynamoDB table. This platform did not allow for custom fields in the JSON message so only the raw hexadecimal data was sent.

Further tests to parse the information in the AWS platform were conducted. Amazon AWS also has a mobile application where it is possible to access some limited information about the devices connected to it.

C. LoRa Solution

For the Lora smart home solution, the architecture used was different from Sigfox, and is presented in figure 7.

For this architecture, a prototype device was needed to send the information to a network server. Next a gateway had to be acquired to enable the messages from the device to reach the LoRa network server. Lastly, the information could be accessed through the network server portal.

The first step to achieve this architecture was to buy a LoRa gateway device. The solution was found to be the Dragino LoRa gateway kit [39]. This kit came with a LoRa Gateway-LG01 P. The kit also had two Arduino UNOs, a LoRa shield, a LoRa/GPS shield, three 868 MHz antennas, and a flame sensor, an ultrasonic sensor, a photosensitive sensor, a buzzer, a relay, and a temperature and humidity sensor.

The LoRa technology was tested using two different protocols, the RadioHead and the LoRaWAN. The former was the recommended protocol, by the Dragino kit, to test the basic LoRa technology connection. The latter is the most used protocol with LoRa technology. For these LoRa experiments all modules used the Arduino IDE software program, including the LoRa gateway which also had a web browser interface. All the code used in the LoRa solution was found at [40].

1) RadioHead Protocol: For the experiments using this protocol the LG01 gateway was used with the Arduino LoRa shield, and one DHT11 temperature and humidity sensor.

The first test with LoRa technology was made to test the connection between the LoRa shield and the gateway, using the RadioHead library [41]. The sketch used sent a message to the gateway and then waited for a response. If successful it shows the RSSI and a response to its message. If no response is received the program would remain in loop asking if the LoRa server was running. The sketch for the gateway waited for the message from the LoRa shield, and if successful showed the message received, the RSSI, and sent a message back. For this experiment to work both sketches had to use the same frequency, bandwidth, coding rate and spreading factor.

The second test was to use IoT platform, in this case the ThingSpeak [42]. For this test the gateway had to be configured as an Wi-Fi client to be able to connect to the platform.

To configure the gateway it was necessary to disable the Wi-Fi access point and Wi-Fi Mesh Network, enable DHCP server in its LAN port and enable internet access via Wi-Fi client mode. Then the necessary SSID, password and encryption for the home network were filled.

The LoRa shield was used with and DHT11 temperature and humidity sensor. As with the previous experiment the frequency, bandwidth, coding rate and spreading factor had to be the same for the LoRa shield and gateway. The sketch was changed so that instead of the thirty seconds interval between messages, a twenty minute interval was used. In the gateway sketch, the
were set: frequency of 868.1 MHz; spreading factor seven (SF7); coding rate of 4/5; signal bandwidth of 125 kHz; and Things Network [43], used for the experiments did not have dashboard capabilities.

In the first experience, the gateway and the device had to be registered at the platform. The gateway also had to be configured to support the LoRaWAN protocol.

For the LoRa radio configurations the following parameters were set: frequency of 868.1 MHz; spreading factor seven (SF7); coding rate of 4/5; signal bandwidth of 125 kHz; and preamble length set at eight. These parameters needed to match the ones from the LoRa shield. The LoRaWAN server options had to be enabled, and its configurations such as the server address, server port and gateway ID, were filled according to the values obtained for the gateway at the platform. The sketch for the gateway was then uploaded.

For the device the ABP method had to be selected, the only one supported by the Dragino gateway. The device address "DevAddr", network session key "NwkSKey" and app session key "AppKey", necessary for authentication, were generated by the TTN service. These values were then inserted in the LoRa/GPS sketch, which sent a test message to the service. At the TTN platform the communication from the device could be confirmed.

In the second experience only the LoRa/GPS device had to be registered, as the gateway configurations would remain the same. The authentication values for the device had to be filled in the new sketch. This experiment used several GPS sketches to find one that worked. The problem was found to be related to the libraries used. At the TTN platform the message, which sent the latitude, longitude and altitude, was received empty. Several attempts were made to correct this experiment.

V. RESULTS AND ANALYSIS

A. Sigfox Boards

For Sigfox one module was considered which could be used with a shield on an Arduino, Raspberry Pi or Waspmote boards. Table II presents the three shield options and their associated prices. Two developer kits were also considered, the Akeru from Snootlab [44] and the SmartEverything board [28], which had prices around 90 € and 85 € respectively.

<table>
<thead>
<tr>
<th>Shield</th>
<th>Arduino</th>
<th>Raspberry Pi</th>
<th>Waspmote</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shield Price</td>
<td>65 €</td>
<td>90 €</td>
<td>160 €</td>
</tr>
<tr>
<td>Board Price</td>
<td>23 €</td>
<td>40 €</td>
<td>+160 €</td>
</tr>
</tbody>
</table>

The Arduino was chosen for providing the cheaper solution, and the SmartEverything board for coming equipped with a series of embedded sensors.

1) TD1208 Board: This board is no longer available, but Telecom Design offers other Sigfox devices. The tests using this board, both with Sigfox and the Carriots platform proved successful and the results were good. The board proved a good starting point for the project and through it, it was possible to see how the Sigfox back-end worked. It also provided a better insight at how the callback process was made.

2) SmartEverything Board: The tests using this board were made using code from examples in order to understand how the board worked and how to program it. All Sigfox tests using this board were successful. The board was not used to its full potential, which additional sketch ideas not implemented. Although the full use of the board was not possible, the tests conducted with it allowed to see that this board had a greater learning curve than the Arduino with the Sigfox shield.

3) Sigfox Radio Shield for Arduino: The sensors tested worked as expected, except for the rain sensor which with a single drop of water accused a flood warning. The accelerometer was tested several times but it was not possible to make it work, or conclude if it even worked. The SHT31 sensor was preferred to the DHT11 temperature sensor, because it had more accurate readings for both temperature and humidity. The decision to only use the SHT31 sensor for the prototype was made to ensure that the smart home architecture could be implemented first, before starting to add more sensors to the prototype.

For the Arduino Sigfox shield test a custom payload configuration had to be created to be able to view the temperature and humidity values at the Sigfox back-end. The first code used which converted the float values to hexadecimal did not work. A new way, by using a union data type was successful.

The problem of uploading sketches to the Sigfox module while it was still connected to the shield was not possible to resolve. So the only way to upload sketches was to remove the module. This method brought several other problems, such as the incorrect initialization of the Sigfox module. Figure 8 presents this case. It shows the module being switched on correctly and the packet not sent. This led to warnings at the Sigfox back-end, which alert to lost messages. To solve this problem the solution was found to only use the loop function at the Arduino sketch. The temperature and humidity values were 4 bytes each, making the payload 8 bytes total. This was not the most efficient way to send the temperature and humidity as both can be sent using only 2 bytes each.

The Narcoleptic sleep code used in the Arduino sketch did not prove very effective. The test conducted with the 9V battery made the device only last for 31 hours. With the eight 1.5V battery holder the device managed to stay on for four days before losing power.

The second sketch created, which used a motion sensor to detect movement, had the message received at the Sigfox back-end successfully, but its implementation was not further developed due to time constraints. The downlink test made was successful, but because there was not a need for downlink messages in the smart home system designed, further tests were not performed.
Only one email callback was created for the prototype where the custom fields for the temperature and humidity, already parsed by the Sigfox back-end were sent in the payload.

Although the Arduino Sigfox shield had its challenges, all tests originally designed for it were achieved and successful.

4) Sigfox IoT Platforms: The first platform used with the prototype was the CloudThing platform. It was chosen for the free use of 5 or less devices. Many attempts were made to synchronize with this platform but were all proved unsuccessful.

Wia was chosen for having a visually appealing dashboard. The tests conducted with it were initially unsuccessful. At the end of the work the platform was found to have radically changed. Although dashboards could not be created, it was possible to sync it with the Sigfox back-end. Even though the platform was used, it was not possible to achieve the parsing of the data or the use of the SMS and twitter solutions, which it offered.

The Tago platform did not offer the possibility to receive custom Sigfox fields. The hexadecimal data from Sigfox was not possible to be parsed. Widgets were created for all fields sent, such as the device ID, SNR, RSSI, base station ID and hexadecimal data. Figure 9 shows a view of the data widget created.

The Microsoft and Amazon IoT cloud services proved the most challenging, complex and time consuming. It was possible to synchronize the prototype with both of them, even though the creation of IoT dashboards was not achieved. Amazon did not have support for custom field data in the JSON structure, unlike Microsoft, but both had the possibility to parse the Sigfox data on their services. These platforms were concluded to not be good solutions for the smart home.

Another platform used was Databoom where it was possible to create a complete IoT dashboard with all data fields sent from the prototype. The widget created for the temperature and humidity is shown in figure 10. This platform was one of the two where a complete IoT dashboard for the smart home system planned was achieved.

The most successful IoT platform tested was the Thinger platform. In figure 11 the Thinger dashboard is shown, with widgets for the temperature and humidity, the SNR, the RSSI, the device ID, and a map showing the location of the base station, which received the message. This was the most successful dashboard implementation, completing the smart home architecture designed for the project, marking the success of the Sigfox solution implementation.

The Microsoft and Amazon IoT cloud services proved the most challenging, complex and time consuming. It was possible to synchronize the prototype with both of them, even though the creation of IoT dashboards was not achieved. Amazon did not have support for custom field data in the JSON structure, unlike Microsoft, but both had the possibility to parse the Sigfox data on their services. These platforms were concluded to not be good solutions for the smart home.

Another platform used was Databoom where it was possible to create a complete IoT dashboard with all data fields sent from the prototype. The widget created for the temperature and humidity is shown in figure 10. This platform was one of the two where a complete IoT dashboard for the smart home system planned was achieved.

The most successful IoT platform tested was the Thinger platform. In figure 11 the Thinger dashboard is shown, with widgets for the temperature and humidity, the SNR, the RSSI, the device ID, and a map showing the location of the base station, which received the message. This was the most successful dashboard implementation, completing the smart home architecture designed for the project, marking the success of the Sigfox solution implementation.

The Microsoft and Amazon IoT cloud services proved the most challenging, complex and time consuming. It was possible to synchronize the prototype with both of them, even though the creation of IoT dashboards was not achieved. Amazon did not have support for custom field data in the JSON structure, unlike Microsoft, but both had the possibility to parse the Sigfox data on their services. These platforms were concluded to not be good solutions for the smart home.

Another platform used was Databoom where it was possible to create a complete IoT dashboard with all data fields sent from the prototype. The widget created for the temperature and humidity is shown in figure 10. This platform was one of the two where a complete IoT dashboard for the smart home system planned was achieved.

The Microsoft and Amazon IoT cloud services proved the most challenging, complex and time consuming. It was possible to synchronize the prototype with both of them, even though the creation of IoT dashboards was not achieved. Amazon did not have support for custom field data in the JSON structure, unlike Microsoft, but both had the possibility to parse the Sigfox data on their services. These platforms were concluded to not be good solutions for the smart home.

Another platform used was Databoom where it was possible to create a complete IoT dashboard with all data fields sent from the prototype. The widget created for the temperature and humidity is shown in figure 10. This platform was one of the two where a complete IoT dashboard for the smart home system planned was achieved.

The Microsoft and Amazon IoT cloud services proved the most challenging, complex and time consuming. It was possible to synchronize the prototype with both of them, even though the creation of IoT dashboards was not achieved. Amazon did not have support for custom field data in the JSON structure, unlike Microsoft, but both had the possibility to parse the Sigfox data on their services. These platforms were concluded to not be good solutions for the smart home.

Another platform used was Databoom where it was possible to create a complete IoT dashboard with all data fields sent from the prototype. The widget created for the temperature and humidity is shown in figure 10. This platform was one of the two where a complete IoT dashboard for the smart home system planned was achieved.

The Microsoft and Amazon IoT cloud services proved the most challenging, complex and time consuming. It was possible to synchronize the prototype with both of them, even though the creation of IoT dashboards was not achieved. Amazon did not have support for custom field data in the JSON structure, unlike Microsoft, but both had the possibility to parse the Sigfox data on their services. These platforms were concluded to not be good solutions for the smart home.

Another platform used was Databoom where it was possible to create a complete IoT dashboard with all data fields sent from the prototype. The widget created for the temperature and humidity is shown in figure 10. This platform was one of the two where a complete IoT dashboard for the smart home system planned was achieved.
from the seven platforms used, the solution was only possible on two of them, the Thinger and Databoom. Although on the other platforms a dashboard was not able to be created, the synchronization with them was still achieved, except for the CloudThing platform.

B. LoRa Dragino Kit

For the LoRa solution, two modules were considered, both for LoRa and LoRaWAN. Table III shows the six Cooking Hacks shield options and their prices. The gateways solutions considered were: to build one, using a Raspberry Pi; or to find one which would not be very expensive.

Table III: Shield solutions for the LoRa module.

<table>
<thead>
<tr>
<th></th>
<th>Arduino</th>
<th>Raspberry Pi</th>
<th>Waspmote</th>
</tr>
</thead>
<tbody>
<tr>
<td>LoRa Shield Price</td>
<td>75 €</td>
<td>85 €</td>
<td>147 €</td>
</tr>
<tr>
<td>LoRaWAN Shield Price</td>
<td>90 €</td>
<td>95 €</td>
<td>163 €</td>
</tr>
<tr>
<td>Board Price</td>
<td>23 €</td>
<td>40 €</td>
<td>+160 €</td>
</tr>
</tbody>
</table>

The solution found was the Dragino kit, because it possessed a LoRa gateway, two LoRa nodes, and several sensors.

1) Dragino Gateway: The Dragino LG01 gateway proved a good device to gain knowledge on LoRa technology. It was not designed to support LoRaWAN, although it can with limited capabilities. Using LoRaWAN it can communicate with other LoRa devices, as long as they share the same frequency, spreading factor, bandwidth, preamble length and sync word. The gateway can only receive one packet at a time.

2) RadioHead Protocol Tests: In the RadioHead tests two other ways to connect the gateway to the home network were tried. The first by connecting the gateway directly to the home Wi-Fi router and the second by connecting the gateway to a Wi-Fi range extender. Both tests proved unsuccessful.

The first test conducted with LoRa between the gateway and the LoRa shield was successful. The test with the ThingSpeak platform also proved successful and a dashboard was created, with figure 12 showing the result of the temperature widget.

3) LoRaWAN Protocol Tests: The configuration of the gateway to support LoRaWAN proved very time consuming. The gateway was also not capable to support all the of the protocol functionalities. The first experience to send a simple tests message proved successful.

The second experience proved very challenging. The codes available for the tests were not working, so a solution had to be procured. The problems were found to be related to the libraries used for LoRaWAN. The solution was found by installing other libraries.

The GPS LoRaWAN message contained 32 bytes, which equated to an average time on air around 75 milliseconds. Following this test, and keeping the TTN fair access policy in mind, of allowing only thirty seconds uplink time, per device, per day, meant that four hundred LoRa messages could be sent in a day doubling what Sigfox offers.

The GPS test was made both indoors and outdoors and a signal was not found. Several attempts were made to solve the problem. This led to the possible conclusions that: there was a mistake in the code; the GPS module was not properly connected to the Arduino board; the library for the GPS was not the right one; or the GPS module was not functioning properly.

With LoRaWAN the proposed architecture was not achieved as TTN is a network server that does not provide capabilities for creating a dashboard. The challenges faced with the LoRaWAN tests proved very time consuming and understanding how the devices and the gateway worked was more difficult than originally though.

VI. Conclusions

The smart home system proposed in this work was achieved with Sigfox and partially with LoRa.

The prototype for Sigfox was created successfully and the results obtained at the Sigfox back-end are the ones which were initially desired. In regards to the platforms, the Thinger platform, proved to be the most suitable choice for a smart home project.

With LoRa, the prototype as not achieved because the technology proved to complex and time consuming to implement. Only one dashboard is used with this technology and successfully implemented. A network server is also used with two tests conducted, one which worked and the other did not.

Based on the experiments conducted it is concluded that both Sigfox and LoRa technologies can be used in a smart home setting. However, they do not have all the necessary requirements to be the sole technology in a smart home. Sigfox proved the best solution for a smart home project, because it offers more support for their customers and their coverage, today, is better than LoRa. LoRa is a more independent technology but it also proves more difficult to implement.
REFERENCES


