Building low-cost gateways and devices for open LoRa IoT test-beds

Congduc PHAM[1]

LIUPPA, University of Pau, France
Email: Congduc.Pham@univ-pau.fr

Abstract. While benefits of IoT are clearly stated the deployment of such devices in a large scale is still held back by technical challenges such as short communication distances. Recent long-range radio technologies such as Semtech's LoRa are promising to deploy Low Power WAN at a very low-cost for a large variety of applications. The paper describes our proposed low-cost and open IoT gateway & devices for both robust and simple deployment. Quick appropriation & customization by third parties for test-bed deployment is of utmost importance and the whole proposed architecture addresses this issue from the very beginning of the design process.

Key words: Internet of Thing test-bed, LoRa, Low-cost test-beds

1 Introduction

While benefits of IoT are clearly stated for increased process efficiency through automation & optimization, the deployment of such devices in a large scale is still held back by technical challenges such as short communication distances. Using the telco mobile communication infrastructure is still very expensive (e.g. GSM/GPRS, 3G/4G/LTE) and not energy efficient for autonomous devices that must run on battery for months. During the last decade, low-power but short-range radio such as IEEE 802.15.4 radio have been considered by the WSN community with multi-hop routing to overcome the limited transmission range. While such short-range communications can eventually be realized on smart cities infrastructures where high node density with powering facility can be achieved, it can hardly be generalized for the large majority of surveillance applications that need to be deployed in isolated or rural environments. Recent modulation techniques where the long transmission distance (several kilometers even in NLOS conditions) can be achieved without relay nodes greatly reduces the complexity of deployment and data collection. The long-range solution has the following advantages over traditional short-range technologies:

1. avoids relying on operator-based communications; no subscription fees;
2. removes the complexity and cost of deploying/maintaining a multi-hop infrastructure;
3. can offer out-of-the-box connectivity facilities.

Figure 1 shows a typical extreme long-range 1-hop connectivity scenario to a gateway which is the single interface to Internet servers. Most of long-range technologies can achieve 20km or higher range in LOS condition and about 2km in NLOS, urban area where the RF signal has to travel through several buildings.

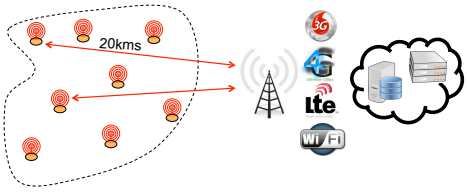

Fig. 1. Extreme long-range application

Some low-power long-range technologies such as Sigfox^{TM} are still operator-based and therefore cannot be deployed in an ad-hoc manner. However, other technologies such as LoRa^{TM} proposed by Semtech radio manufacturer can be privately used. Such technology can be deployed following the recently proposed LoRaWAN^{TM} specifications [1] for large-scale interoperability or using completely ad-hoc solutions where customization towards specific application's profile can be realized. The work presented in this paper mainly focuses on this last approach. We present here our low-cost LoRa platform & software for deploying ad-hoc LoRa IoT test-beds with a high degree of customization and flexibility.

The rest of the article is organized as follows. Section 2 details the long-range technology with focus on the LoRa technology by Semtech and explanations on low-power, long-range network architecture. In Section 3 we will present our low-cost architecture, hardware and software designed for test-bed deployment. Section 4 then describes how a test-bed can be rapidly set-up. We conclude in Section 5.

2 Review of long-range transmission and LPWAN

2.1 Semtech's LoRa technology

Semtech's long-range technology (called LoRa [2, 3]) belongs to the spread spectrum approaches where data can be "spreaded" in both frequencies and time to increase robustness and range by increasing the receiver's sensitivity, which can be as low as -137dBm in 868MHz band or -148dBm in the 433MHz band. Throughput and range depend on the 3 main LoRa parameters: BW, CR and SF. BW is the physical bandwidth for RF modulation (e.g. 125kHz). Larger signal bandwidth allows for higher effective data rate, thus reducing transmission time at the expense of reduced sensitivity. CR, the coding rate for forward error detection and correction. Such coding incurs a transmission overhead and

the lower the coding rate, the higher the coding rate overhead ratio, e.g. with $coding\_rate = 4/(4+CR)$ the overhead ratio is 1.25 for CR=1 which is the minimum value. Finally SF, the spreading factor, which can be set from 6 to 12. The lower the SF, the higher the data rate transmission but the lower the immunity to interference thus the smaller is the range. Figure 2 shows for various combinations of BW, CR and SF the time-on-air of a LoRa transmission depending on the number of transmitted bytes. The maximum throughput is shown in the last column with a 255B payload. Modes 4 to 6 provide quite interesting trade-offs for longer range, higher data rate and immunity to interferences.

				time on air in second for payload size of						
LoRa mode	BW	CR	SF	5 bytes	55 bytes	105 bytes	155 Bytes	205 Bytes	255 Bytes	max thr. for 255B in bps
1	125	4/5	12	0.95846	2.59686	4.23526	5.87366	7.51206	9.15046	223
2	250	4/5	12	0.47923	1.21651	1.87187	2.52723	3.26451	3.91987	520
3	125	4/5	10	0.28058	0.69018	1.09978	1.50938	1.91898	2.32858	876
4	500	4/5	12	0.23962	0.60826	0.93594	1.26362	1.63226	1.95994	1041
5	250	4/5	10	0.14029	0.34509	0.54989	0.75469	0.95949	1.16429	1752
6	500	4/5	11	0.11981	0.30413	0.50893	0.69325	0.87757	1.06189	1921
7	250	4/5	9	0.07014	0.18278	0.29542	0.40806	0.5207	0.63334	3221
8	500	4/5	9	0.03507	0.09139	0.14771	0.20403	0.26035	0.31667	6442
9	500	4/5	8	0.01754	0.05082	0.08154	0.11482	0.14554	0.17882	11408
10	500	4/5	7	0.00877	0.02797	0.04589	0.06381	0.08301	0.10093	20212

Fig. 2. Time on air for various LoRa modes as payload size is varied

Electromagnetic transmissions in the sub-GHz band of Semtech's LoRa technology falls into the Short Range Devices (SRD) category. For instance, in Europe, electromagnetic transmissions in the EU 863-870MHz ISM Band used by Semtech's LoRa technology falls into the Short Range Devices (SRD) category. The ETSI EN300-220-1 document [4] specifies various requirements for SRD devices, especially those on radio activity. Basically, transmitters are constrained to 1% duty-cycle (i.e. 36s/hour) in the general case. This duty cycle limit applies to the total transmission time, even if the transmitter can change to another channel. In most cases, however, the 36s duty-cycle is largely enough to satisfy communication needs of deployed applications.

2.2 LoRa LPWAN network deployment and architecture

As shown previously in figure 1, the deployment of a LoRa network is centered around a gateway that usually has Internet connectivity. Although direct communications between devices are possible, most of applications using sensors for surveillance follow the gateway-centric approach with mainly uplink traffic patterns. Data captured by end-devices are sent to the gateway which will push data to network servers. Then application servers managed by end-users could retrieve data from the network servers. If encryption is used for confidentiality, the application server can be the place where data could be decrypted and presented to end-users. Following this architecture, the LoRa Alliance proposes a LoRaWAN [1] specification for deploying large-scale, multi-gateways networks and full network/application servers.

The full network/application servers and LoRaWAN architecture can be greatly simplified for small, ad-hoc deployment scenarios where a privately owned

gateway can (*i*) locally store collected data and use short range wireless radio (WiFi or Bluetooth) for direct web connection or/and, (*ii*) push data to some end-user managed servers or IoT-specific cloud platforms if properly configured. This is the approach we take in this paper to promote application-specific LoRa test-beds deployments.

3 Low-cost LoRa gateway & devices

The implementation of the full LoRaWAN specification requires gateways to be able to listen on several channels and LoRa settings simultaneously. Commercial gateways therefore use advanced concentrators chips capable of scanning up to 8 different channels: the SX1301 concentrator is typically used instead of the SX127x chip serie which is designed for end-devices. They cost several hundredth euros with the cost of the SX1301-capable board alone to be more than a hundred euro. In many scenarios (e.g. small farms, developing countries, test-beds,...) it is more important to keep both the cost of the gateway and the system's complexity low, and to target small to medium size deployments for specific use cases instead of the large-scale, multi-purpose deployment scenarios defined by LoRaWAN. Note that our approach can deploy more than 1 gateway to serve several channel settings if needed. This solution presents the advantage of being more optimal in terms to cost as incremental deployment can be realized and also offer a higher level of redundancy. We believe this statement remains true even for recent LoRa community-based deployment initiatives such as the one conducted by `TheThingNetwork`™ [5] where the deployment mainly targets large-scale, public and multi-purpose networks.

3.1 Single-connection low-cost LoRa gateway

Our LoRa gateway [6] could be qualified as "single connection" as it is built around an SX1272/76, much like an end-device would be. The cost argument, along with the statement that too integrated components are difficult to repair and/or replace in the context of ad-hoc deployments or developing countries, also made the "off-the-shelves" design orientation an obvious choice. Our low-cost gateway is therefore based on a Raspberry PI (version 1 or 2) which is both a low-cost (less than 30 euro) and a highly reliable embedded Linux platform, see figure 3. There are many SX1272/76 radio modules available and we have fully tested 3: the Libelium SX1272 LoRa, the HopeRF RFM92W/95W and the Modtronix inAir9/9B. Normally, any SX1272/76-based radio module using native SPI interface should work. The total cost of the gateway is as low as 45 euro with the HopeRF or Modtronix LoRa modules.

Together with the "off-the-shelves" component approach, the software stack is completely open-source: (a) the Raspberry runs a regular Raspian distribution, (b) ArduPi and the original SX1272 library provided by Libelium are very simple to install/understand/modify and (c) the `lora_gateway` program (which receives and forwards radio packets) is kept as simple as possible.

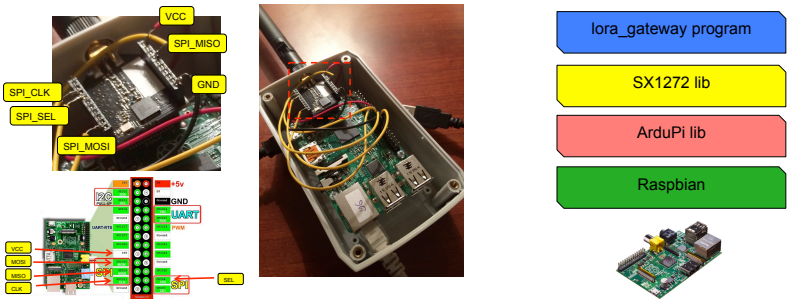

Fig. 3. Low cost gateway from off-the-shelves components

We improved the original SX1272 library in various ways to provide support for both SX1272 and SX1276 chip, enhanced radio channel access (CSMA-like with SIFS/DIFS) and radio activity time sharing. One of the main objectives of our architecture and software stack is to provide both robust and simple solution for either "out-of-the-box" utilization or quick appropriation & customization by third parties when deploying test-beds for specific applications.

After compiling the `lora_gateway` program, the most simple way to start the gateway is in standalone mode as shown is figure 4(left). By default, the LoRa mode is 4 (BW=500kHz, CR=4/5 and SF=12) and the frequency channel is 865.2MHz. All packets received by the gateway is sent to the standard Unix-stdout stream. The gateway can also be started on a given LoRa mode, see Figure 2, with the `--mode` option. For full customization, `--bw`, `--cr`, `--sf` and `--freq` options can indicate a bandwidth, coding rate, spreading factor and frequency channel combination. For instance, testing one of the LoRaWAN mandatory channel in the EU 863-868MHz ISM band can be done as follows `--bw 125 --cr 5 --sf 12 --freq 868.1`.

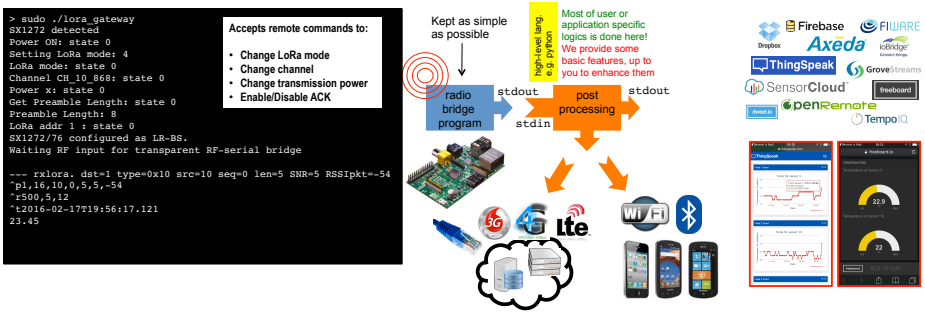

Fig. 4. Gateway architecture for radio data reception and post-processing stages

For each data packet received, `lora_gateway` provides various information that can be used by the post-processing stage. These informations are related to the packet (destination–normally the gateway–, packet type, source addr, sequence number, data length, SNR and packet's RSSI), the radio (bandwidth, coding rate and spreading factor) and the time of reception.

3.2 Post-processing and link with IoT cloud platforms

Advanced data post-processing tasks are performed after the radio stage by using Unix redirection of `lora_gateway`'s outputs as shown by the orange "post-processing" block in figure 4(middle). We promote the usage of high-level language such as `Python` to implement all the data post-processing tasks such as access to IoT cloud platforms and even AES decryption features. Our gateway is distributed with a `Python` template that explains and shows how to upload data on various IoT cloud platforms. Examples include `Dropbox`™, `Firebase`™, `ThingSpeak`™, `freeboard`™, `SensorCloud`™, `GrooveStream`™ & `FiWare`™ as illustrated in figure 4(right).

This architecture clearly decouples the low-level gateway functionalities from the high-level post-processing features. By using high-level languages for post-processing, running and customizing data management tasks can be done in a few minutes. "Out-of-the-box" data upload to IoT cloud can be realized as most of these platforms propose free accounts that can satisfy a large number of foreseen IoT applications. For instance, a small farm can deploy in minutes the sensors and the gateway using a free account with `ThingSpeak` platform to visualize captured data in real-time.

3.3 Gateway running without Internet access

One additional important issue that needs to be taken into account when deploying test-beds is the intermittent or no access to the Internet for the gateway. In all cases, received data are locally stored on the gateway in a NoSql database (e.g. `MongoDB`) and the gateway can interact with the end-users' smartphone through WiFi or Bluetooth as depicted in figure 4(middle). WiFi or Bluetooth dongles for Raspberry can be found at really low-cost and the smartphone can be used to display captured data (a web server is run by the gateway using `JQuery` for forms and graphs) or notify users of important events a without the need of Internet access as this situation can clearly happen in very remote areas.

3.4 Low-cost LoRa end-devices

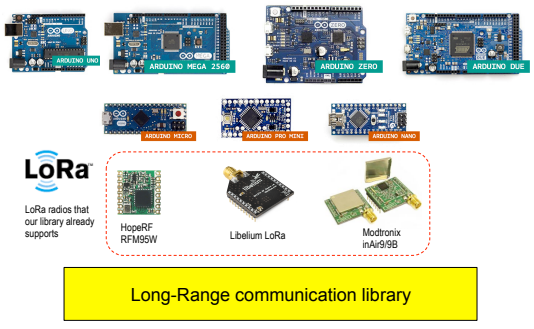

Fig. 5. Low-cost LoRa end-device for customization

Our communication library (the same library is used for the gateway) is mainly tested on Arduino boards. Figure 5 shows all the Arduino boards that have successfully been tested, from the Uno to the Nano platform. We use the MEGA as a prototyping platform. For better integration purposes, we use the Arduino Pro Mini or Nano which can be bulk purchased for about 2 euro per piece. They can be used to provide a generic platform for sensing and long-range transmission.

We provide templates for quick and easy new behaviour customization while integrating all the necessary code for advanced channel access and data encryption if needed. By default, they run "out-of-the-box" with the gateway.

4 Deploying a test-bed

We describe in this section how a test-bed can be deployed and set-up with the previously described components. We will use a `ThingSpeak` channel for storing received data to the cloud.

4.1 Post-processing data

The post-processing `Python` template `post_processing.py` contains a section to look for predefined data prefix for IoT clouds. The prefix `\!` is used to indicate the usage of a `ThingSpeak` channel. It is followed by an optional write key, an optional field index and the value to report. For instance `\!SGSH52UGPVAUYG3S#1#21.6` will be processed to upload `21.6` to the `ThingSpeak` channel which write key is `SGSH52UGPVAUYG3S` on field index `1`. Using default write key and field index can be done with `\!##21.6`.

The post-processing stage can also enforce the presence of a valid application key. This application key is coded on 4 bytes and is inserted before the prefixed data. A list of valid application key is defined in the `Python` post-processing script. For the test, we use an application key list consisting in `['\x05\x06\x07\x08']`.

4.2 Deploying gateways

The gateway can simply be started by launching `lora_gateway` with redirection of the output to the post processing stage as follows: `sudo ./lora_gateway | python ./post_processing.py`. Note that several gateways can be deployed to improve coverage or reliability. They can also work on the same LoRa parameters or on different settings for test purposes. All gateway's outputs can be further logged in a log file. We use the `Dropbox` file sharing service for log files therefore all the logs can be made available on various number of platforms.

4.3 Simple end-device for telemetry

The generic sensing template is used to drive a temperature sensor. The end-device will simple send `\!##21.6` if the sensed temperature is `21.6`. The application key `uint8_t my_appKey[4]={5, 6, 7, 8}` is inserted before. Figure

6(right) illustrates the Arduino Nano with a HopeRF RFM95W module running a temperature code based on the generic sensing template. By default, it takes a measure every 10 minutes and stay in sleep mode between 2 measures.

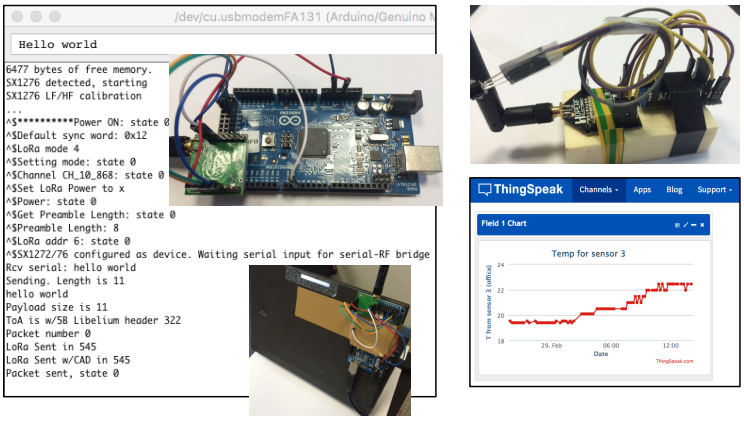

Fig. 6. Left: Interactive LoRa end-device; right: temperature LoRa device

At this point, the basic test-bed is set-up as the gateway will receive and upload on the `ThingSpeak` channel the measured temperature. The `ThingSpeak` channel can also be used to store and plot the SNR of received packets. If the log files use `Dropbox` sharing, then one can also check the log files for link quality (SNR and missing packets).

4.4 Interactive end-device for deployment tests

We also provide an interactive end-device template that can be used to interactively send ASCII strings to the gateway. Such device can be plugged and fixed to a laptop for field tests as illustrated in figure 6(left) which features an Arduino MEGA2560 with a Modtronix inAir9B radio. The serial monitor of the Arduino IDE is used for both input and output. Input strings beginning with `/@` are interpreted as command strings for the host program. Figure 7 shows the list of currently available commands.

Commands for configuring the LoRa parameters, for sending periodic messages and for requesting an acknowlegment (ACK) from the gateway are those that are useful in test situation. For instance, a range test can be easily realized by sending a message and requesting an ACK from the gateway: `/@ACK#hello`. As the gateway can also accept remote commands in ASCII format for configuring various LoRa parameters, the interactive end-device can be used to remotely configure the gateway for various test purposes. For instance the following command sequence `/@ACK#/@M2#`, `/@M2#`, `/@ACK#hello` switches both the gateway and the end-device to loRa mode 2 and check whether connectivity is still maintained. When an ACK is requested, the SNR value of the received message is

sent back to the device in the ACK by the gateway. At the end-device, the SNR of both the message and the ACK is displayed so that both uplink and downlink quality can be monitored.

Command	Action
/@M1#	set LoRa mode 1
/@C12#	use channel 12
/@PL/H/M/x/X#	set power to Low, High, Max, extreme (PA_BOOST), eXtreme (+20dBm)
/@A9#	set node addr to 9
/@ACK#hello w/ac	sends "hello w/ack" and request an ACK
/@ACKON#	enables ACK (for all messages)
/@ACKOFF#	disables ACK
/@CAD#	performs an SIFS CAD, i.e. 3 or 6 CAD depending on the LoRa mode
/@CADON3#	uses 3 CAD when sending data (normally SIFS is 3 or 6 CAD, DIFS=3SIFS)
/@CADOFF#	disables CAD (IFS) when sending data
/@RSSI#	toggles checking of RSSI before transmission and after CAD
/@EIFS#	toggles for extended IFS wait
/@T5000#	send a message at regular time interval of 5000ms. Use /@T0# to disable periodic sending
/@TR5000#	send a message at random time interval between [2000, 5000]ms.
/@Z200#	sets the packet payload size to 200 for periodic sending
/@S50#	sends a 50B user payload packet filled with '#'. The real size is 55B with the protocol header
/@D56#	set the destination node to be 56, this is permanent, until the next D command
/@D56#hello	send "hello" to node 56, destination addr is only for this message
/@D1#/@M1#	send the command string "/@M1#" to node 1 (i.e. gateway)

Fig. 7. LoRa end-device command list

4.5 Adding advanced features

Improved channel access. The framework we provide is interesting especially for adding and testing new features. For instance, a CSMA-like mechanism with SIFS/DIFS has been implemented using the Channel Activity Detection (CAD) functionality of the LoRa chip and can further be customized. It can be activated with command `/@CADON3#` where 3 is the number of CAD for an SIFS. A DIFS is defined as 3 SIFS. Prior to packet transmission a DIFS period free of activity should be observed, see figure 8(left). If "extended IFS" is activated with command `/@EIFS#` then an additional number of CAD followed by a DIFS is required. If RSSI checking is activated with command `/@RSSI90#` then the RSSI should be below -90dB for the packet to be transmitted. By running a background periodic source of LoRa packets, we observed that the improved channel access succeeds in reducing packet collisions. The current framework is used to study the impact of channel access methods in a medium-size LoRa deployment.

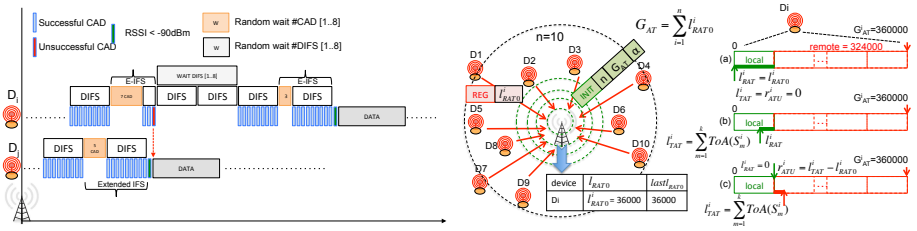

Fig. 8. Left: CSMA-like mechanism for increased robustness; Right: activity time sharing

Activity time sharing. We also implemented an exploratory activity time sharing mechanism for a pool of devices managed by a single organization. We propose to overcome the tight 36s/hour radio activity of a device by considering all the sensor's individual activity time in a shared/global manner. The approach we propose in this paper will allow a device that "exceptionally" needs to go beyond the activity time limitation to borrow some from other devices. A global view of the global activity time, G_{AT}, allowed per 1 hour cycle will be maintained at the gateway so that each device knows the potential activity time that it can use in a 1-hour cycle. Figure 8(left) shows how the deployed long-range devices D_i sharing their activity time initially register (REG packet) with the gateway by indicating their local Remaining Activity Time l^i_{RAT0}, i.e. 36s. The gateway stores all l^i_{RAT0} in a table, computes G_{AT} and broadcasts (INIT packet) both n (the number of devices) and G_{AT}. This feature is currently tested for providing better surveillance service guarantees.

5 Conclusions

Targeted for small to medium size deployment of test-beds, our low-cost, open long-range IoT framework allows for quick appropriation & customization by third parties. Developed within the EU H2020 WAZIUP project which addresses the challenges of low-cost IoT deployment in developing, low-income sub-saharan Africa countries, the platform is tested by WAZIUP's partners with a large-scale test-bed to be set-up in Senegal. The platform is also currently used by several organizations and companies for deploying LoRa test-beds and conducting field tests for various surveillance applications [6]: agriculture, oceanographic observation, pest traps monitoring....

Acknowledgments

This work is support by the WAZIUP project with funding from the EU's H2020 research and innovation program under grant agreement No 687607.

References

1. LoRaAlliance, "LoRaWAN specification, v1.0," 2015.
2. Semtech, "LoRa modulation basics. rev.2-05/2015," 2015.
3. S. Jeff McKeown, "LoRa™- a communications solution for emerging LPWAN, LPHAN and industrial sensing & IoT applications. http://cwbackoffice.co.uk/docs/jeff 20mckeown.pdf," accessed 13/01/2016.
4. ETSI, "Electromagnetic compatibility and radio spectrum matters (ERM); short range devices (SRD); radio equipment to be used in the 25 MHz to 1 000 MHz frequency range with power levels ranging up to 500 mw; part 1." 2012.
5. TheThingNetwork, "http://thethingsnetwork.org/," accessed 13/01/2016.
6. C. Pham, "A DIY low-cost LoRa gateway. http://cpham.perso.univ-pau.fr/lora/rpigateway.html," accessed 13/01/2016.