EU H2020 grant agreement number 687607

Low-cost LoRa IoT devices and gateway FAQ

1) What is Internet-of-Thing (IoT)?

From IERC (European Research Cluster on the Internet of Thing)

The IERC definition states that IoT is "A dynamic global network infrastructure with self-configuring capabilities based on standard and interoperable communication protocols where physical and virtual “things” have identities, physical attributes, and virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network."

From http://www.gartner.com/it-glossary/internet-of-things/

"The Internet of Things (IoT) is the network of physical objects that contain embedded technology to communicate and sense or interact with their internal states or the external environment."

From http://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT

"The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction."

2) What is WAZIUP?

The EU H2020 WAZIUP project, namely the Open Innovation Platform for IoT-Big Data in Sub-Saharan Africa is a collaborative research project using cutting edge technology applying IoT and Big Data to improve the working conditions in the rural ecosystem of Sub-Saharan Africa. First, WAZIUP operates by involving farmers and breeders in order to define the platform specifications in focused validation cases. Second, while tackling challenges which are specific to the rural ecosystem, it also engages the flourishing ICT ecosystem in those countries by fostering new tools and good practices, entrepreneurship and start-ups. Aimed at boosting the ICT sector, WAZIUP proposes solutions aiming at long term sustainability.

WAZIUP will deliver a communication and big data application platform and generate locally the know how by training by use case and examples. The use of standards will help to create an interoperable platform, fully open source, oriented to radically new paradigms for innovative application/services delivery. WAZIUP is driven by the following visions:

1. Empower the African Rural Economy. Develop new technological enablers to empower the African rural economy now threatened by the concurrent action of rapid urbanization and of climate change. WAZIUP technologies can support the necessary services and infrastructures to launch agriculture and breeding on a new scale;

2. Tailored IoT and Big-data Technology. Offer smart sensor and data-driven applications and services addressing the end-users needs and requirements (understanding users requirements and preference delivering towards more personalized and easy users interfaces and applications).

3. Value-added cost and energy efficiency. IoT application and services based on WAZIUP open IoT-Big data platform will focus on ease of maintenance and low cost of solutions;

4. Lower Entry Level. Provide to application developers a mature platform, as well as tools and standards that are inexpensive, easy and relevant.

In order to achieve the above aims, a strong dissemination and exploitation effort of the project will be dedicated to a) strengthening linkages of end-users with industries, b) engage innovation space and living labs to accelerate innovation coaching/training/start-up activities (e.g., community-driven development paradigms), c) promote value-addition to business outputs, d) challenge the value-chain of African agribusiness through technology for value increase.

The proposed solutions will be tested for a set of real-life use cases covering several countries. At higher level, WAZIUP will implement a regional innovation platform, where SMEs could continue to develop/plug-in solutions using the technical elements and the open data provided in the project. The ultimate target is to create large African industries, SMEs ecosystem, and induce a network-effect.

The consortium of WAZIUP involves 7 partners from 4 African countries and partners from 5 EU countries combining business developers, technology experts and local Africa companies operating in agriculture and ICT. The project involves also regional hubs with the aim to promote the results to the widest base in the region.

3) What is LoRa?

LoRa is a long-range radio technology developed by Semtech. Here is a definition from Semtech's LoRa FAQ:

"*LoRa™ (Long Range) is a modulation technique that provides significantly longer range than competing technologies. The modulation is based on spread-spectrum techniques and a variation of chirp spread spectrum (CSS) with integrated forward error correction (FEC). LoRa significantly improves the receiver sensitivity and as with other spread‐spectrum modulation techniques, uses the entire channel bandwidth to broadcast a signal, making it robust to channel noise and insensitive to frequency offsets caused from the use of low cost crystals. LoRa can demodulate signals 19.5dB below the noise floor while most frequency shift keying systems (FSK) need a signal power of 8-10dB above the noise floor to demodulate properly. The LoRa modulation is the physical layer (PHY), which can be utilized with different protocols and in different network architecture – Mesh, Star, point to point, etcetera.*" [http://www.semtech.com/wireless-rf/lora/LoRa-FAQs.pdf]

WAZIUP will use LoRa radio technology to deploy Low-Power Wide-Area Networks (LPWAN) for connecting the developed low-cost IoT devices.

4) What are the main parameters of LoRa radios?

Throughput and range depend on 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.

5) What is the typical range of LoRa radio technology?

Most of tests performed by Semtech and other companies show radio range of about 15-20kms in line-of-sight condition with BW=125kHz and SF=12. The range can be greatly increased if the antennas can be set up higher. Some High Altitude Ballooning experiences have reported range of several hundredths of kms. Under non line-of-sight condition such as in a dense urban area or indoor scenarios, the maximum range is about 2kms.

6) What is the usual LoRa topology?

With the longer range the usual LoRa topology is a star: end-devices (IoT device) will send data to a gateway which is usually assumed to have Internet connection to be able to push received data to Internet servers. Note that several gateways can be deployed to cover a larger area and it may happen that a message from an end-device is received by several gateways. Filtering for duplicated messages can be realized at the Internet server level, or more generally at a layer above the gateway.

7) What is LoRaWAN ?

From Semtech's LoRa FAQ:

“The LoRa modulation is the PHY, and LoRaWAN is a MAC protocol for a high capacity long range and low power star network that the LoRa Alliance is standardizing for Low Power Wide Area Networks (LPWAN). The LoRaWAN protocol is optimized for low cost, battery operated sensors and includes different classes of nodes to optimize the tradeoff between network latency and battery lifetime. It is fully bi-directional and was architected by security experts to ensure reliability and safety. The architecture of LoRaWAN was also designed to easily locate mobile objects for asset tracking, which is one of the fastest growing volume applications for Internet of Things (IoT). LoRaWAN is being deployed for nationwide networks by major telecom operators, and the LoRa Alliance is standardizing LoRaWAN to make sure the different nationwide networks are interoperable.”

LoRaWAN therefore defines common data and control channels (frequency and spreading factors), packet format, MAC commands,... for large-scale deployment with network servers and application servers. LoRaWAN also defines several classes for the end-device depending on the communication needs. Each class has its own requirements also defined by LoRaWAN specifications.

8) What does our LoRa framework provide?

The WAZIUP LoRa framework will provide an open-source, simple long-range communication library, generic building blocks and tools for building efficient low-cost IoT devices and gateways from “off-the-shelves” consumer-market components. The long-range communication library was initially developed by Libelium. We enhanced it with various mechanisms for WAZIUP.

For the low-cost end-devices, examples (and templates) show the usage of the long-range communication library and how low-power/duty-cycled applications can be programmed.

For the low-cost gateway, WAZIUP provides a low-level radio bridge program to receive LoRa packets from end-devices and higher-level programs/tools/scripts to further process the received data. A typical simple task would be for instance to push received data from deployed sensors to public cloud platforms.

Training materials such as step-by-step tutorial slides and short tutorial video sequences explain how to build the entire low-cost WAZIUP LoRa IoT ecosystem.

9) What is the deployment scenario of our LoRa framework?

Rather than providing large-scale deployment support which is the main target of the LoRaWAN specification, WAZIUP targets small size deployment scenarios where there will be most likely a single application owner, e.g. a fish farming manager willing to get in real-time various water quality indicators in the fish ponds.

Therefore our LoRa framework focuses on easy integration of low-cost "off-the-shelves" components with simple, open programming libraries and templates for easy appropriation and customization by third-parties.

In addition, WAZIUP also takes into account the fact that Internet connectivity can be quite unstable or simply impossible to get in some remote areas. Our framework can be deployed in a fully autonomous way without the need of Internet access nor Internet servers (especially avoiding network and application servers as defined by LoRaWAN) to get the sensed data. Therefore, our framework proposes local interaction methods with the end-user using smartphones/tablet/laptop with well-known technologies such as WiFi or Bluetooth.

10) Is it LoRaWAN compliant?

No.

11) Why is it not LoRaWAN compliant?

From the orientation and design choice perspective see 8) and 9) and because we want to provide a level of performance and customization that is not available with LoRaWAN. Technically, it is not LoRaWAN compliant for the following main reason:

> With the gateway-centric mode of LoRa LPWAN technology, commercial LoRaWAN gateways for large-scale deployment scenarios are able to listen on several channels and LoRa settings simultaneously. They typically use advanced radio concentrators chips capable of scanning up to 8 different channels: the

SX1301 concentrator is 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. Our gateway uses a different approach in the context of agriculture/micro and small farm/village environments: simpler "single connection/channel" (one combination of BW, CR, SF and one frequency at a time) gateways can be built around an SX1272/76 radio module, much like an end-device would be. This design choice greatly decreases the cost and complexity of the gateway. At the same time, advanced mechanisms to limit interferences can be implemented to compensate for the "single connection/channel" limitation.

Other reasons are: different (simpler) packet format and optional network & application servers for our framework.

Our framework also does not follow the LoRaWAN standard because it wants to propose the possibility to add advanced and ad-hoc mechanisms such as:

- P2P (device-to-device) communications to allow for direct device cooperation schemes;
- LoRa repeaters functionality in end-devices to handle practical deployment issues when some sensing devices are in very remote areas, placed very close to the ground or obstacles;
- Advanced (and ad-hoc) channel access methods to increase transmission reliability;
- Advanced Quality of Service mechanism to handle the duty-cycle limit of sub-GHz transmission, providing some means of guaranteeing transmission latency;

12) Practically, how do I build an IoT device with the framework?

Use a microcontroller board and connect a LoRa radio module. Then use our framework libraries, building blocks and templates to program the microcontroller according to the application requirements. Follow training materials that explain how to build the entire low-cost LoRa IoT ecosystem: how to connect the physical sensors, how to operate the IoT device with batteries, how to define duty-cycled behaviour for saving energy.

13) What microcontroller boards are supported?

The availability of low-cost, open-source hardware platforms such as Arduino-like boards is clearly an opportunity for building low-cost IoT devices from consumer market components. Our long-range communication library therefore targets such Arduino-like boards: original Arduino boards (Uno, MEGA, Due, Mini, Pro Mini, Nano, M0) but also many other Arduino-compatible boards from Sparkfun, Teensy, RFduino, Ideetron Nexus, Sodaq, Intel,... if they have compatibility with the Arduino IDE.

In the context of WAZIUP, we use the Arduino Pro Mini for simple, small-memory applications such as telemetry applications. Such Arduino Pro Mini can be purchased for less than 2€ a piece from Chinese manufacturers exhibiting a very reasonable quality. We then use the Teensy3.2 for more power/memory demanding applications.

14) What radio modules are supported?

There are many SX1272/76-based radio modules available and we currently tested with 6: the Libelium SX1272 LoRa, the HopeRF RFM92W(SX1272) & RFM95W(SX1276), the Modtronix inAir9(SX1276) & inAir9B(SX1276) and the NiceRF SX1276. Actually most native SPI-based LoRa modules are actually supported without modifications as reported by many users. In most cases, only a minimum soldering work is necessary to connect the required SPI pins of the radio (MISO,MOSI, CS, CLK) to the corresponding pins on the microcontroller board.

15) Practically, how do I build the gateway with the framework?

Our LoRa low-cost gateway can be qualified as "single connection" as it is built around an SX1272/76, much like an end-device would be. The low-cost gateway is based on a Raspberry PI (1B/1B+/2B/3B) which is both a low-cost and a reliable embedded Linux platform. To install the Raspberry, you can start from scratch or download our pre-installed SD card image. Complete instructions to install from scratch with the Raspbian OS is provided. When all the software components have been installed, connect a radio module and then start the gateway. Complete gateway configuration instructions is provided.

16) What are the main software components involved?

At the end-device, the main software components are the long-range communication library and the predefined building blocks for realizing duty-cycled behaviour and low-power management. Development will be based on C/C++ language using the Arduino IDE.

For the gateway, a low-level radio bridge program is provided. Higher level functionalities are implemented in Python language, organized in various scripts. Therefore customization to different cloud platforms or to specific application-oriented data processing tasks can be implemented in Python on the gateway. It is also possible to simply use the gateway to push data to Internet servers and implement specific application-oriented data processing tasks after the gateway.

WAZIUP provides a generic post-processing-gw.py Python template that already show how to upload received data to public cloud platforms such as ThingSpeak™, GroveStreams™, FIWARE, Freeboard™ and SensorCloud™.

17) What is the packet format?

The LoRa PHY layer packet format is unchanged and managed by the radio module. Then, a packet contains a 4-byte header before the real user data. The header is organized as follows:

[DST(1B), PTYPE(4bits), FLAGS(4bits), SRC(1B), SN(1B)] [DATA(nB)]

DST is the destination address. With the gateway-centric topology, the gateway usually have address 1 so DST will most likely be 1. SRC is the source address and SN is the packet sequence number. PTYPE has currently 2 values: 0001 for DATA packet and 0010 for an ACK packet. FLAGS is a 4-bit array that defines the following options:

- 1000: ack requested
- 0100: data is encrypted
- 0010: data has application key
- 0001: reserved

This 4-byte header is managed by our communication library and only the n bytes of [DATA] will be provided to the programmer. If the programmer wants to implement application key to perform further filtering, he can set the application key flag and insert in [DATA] a sequence of bytes that can be further checked at the Python post-processing stage. In general, one can implement its own packet format, variants or new functionalities (such as AES encryption) by defining a specific format in [DATA] and make the appropriate decoding at the Python post-processing stage.

18) What are the main functions involved for sending?

To send a packet, `sendPacketTimeout(dst,msg,pl)` is used. It accepts 3 parameters: `dst` is typically 1 indicating the address of the gateway, `msg` is the message buffer and `pl` is the message size. Prior to send a message, one should indicate its type using `setPacketType(PKT_TYPE_DATA)` for a DATA packet. Pre-defined option flags can be set as follows: `setPacketType(PKT_TYPE_DATA | PKT_FLAG_DATA_WAPPKEY)`. Example:

```
sx1272.setPacketType(PKT_TYPE_DATA);
sx1272.sendPacketTimeout(1,"temp is 18.56C",14);
```

Upon reception, the gateway will provide `temp is 18.56C` to the post-processing stage.

19) What is the most suitable data format for my application?

The most suitable data format is the one that can satisfy your application and allow simpler data processing tasks. Also, ASCII format is generally simpler than binary format as not always longer when floating-point numbers are involved. If you have several physical sensors connected to the same board, it is better to be able to differenciate the values from the various physical sensors.

Example: you have a temperature sensor, a humidity sensor and a dissolved oxygen sensor. You can use for instance TC prefix for the temperature sensor, HU prefix for the humidity sensor and DO prefix for the dissolved oxygen sensor. Therefore the following simple data format can suit your need:

```
TC/18.56/HU/68.6/DO/8.45
```

It is simple to understand and can easily be splitted into (prefix,value) fields for post-processing. If you want to save 3 bytes, you can use T, H and D instead of TC, HU and DO prefixes but you will maybe loose in clarity. So it's up to you.

20) How can I set the LoRa radio parameters?

As indicated previously, the 3 main LoRa parameters are BW, CR and SF. BW and SF being the 2 most important. However you do not need to act on these parameters directly. The communication library defines 10 so-called LoRa modes

(from 1 to 10) that are various combinations of BW and SF. For instance LoRa mode 1 defines BW=125kHz, CR=4/5 and SF=12. This combination provides the highest sensitivity at the receiver therefore it is suitable to achieve the longest range. However, the transmission time is the highest. Practically a real deployment can use this mode for all deployed devices to be sure to get the larger coverage. For the other modes, the range is generally decreased but transmission time is reduced.

To set the LoRa mode, simply use:

```
sx1272.setMode(myLoraMode);
```

The next parameter to set is the channel frequency. Again, the library has 8 pre-defined channels (from 10 to 17) in the 865-868MHz band and 13 pre-defined channels (from 0 to 12) in the 903-915MHz band. To set the LoRa channel, simply use:

```
sx1272.setChannel(CH_10_868);
```

The description of each LoRa mode and the list of predefined frequencies are on the github repository (see "Where can I get all the documentation").

21) How do I set the transmission power?

To set the LoRa transmission power, simply use:

```
sx1272.setPower("M");
```

to set power to Max for instance. Other available options are "L" for Low, "H" for High, "x" for extreme and "X" for eXtreme at 20dBm (100mW).

The Semtech SX1272/76 has actually 2 lines of RF power amplification (PA): a high efficiency PA up to 14dBm (RFO) and a high power PA up to 20dBm (PA_BOOST). "L", "H", and "M" only use the RFO and deliver 2dBm, 6dBm and 14dBm respectively. "x" and "X" use the PA_BOOST and deliver 14dBm and 20dBm respectively.

However even if the SX1272/76 chip has the PA_BOOST and the 20dBm features, not all radio modules (integrating these SX1272/76) do have the appropriate wiring and circuits to enable these features: it depends on the choice of the reference design that itself is guided by the main intended frequency band usage, and sometimes also by the target country's regulations (such as maximum transmitted power). So you have to check with the datasheet whether your radio module has PA_BOOST (usually check whether the PA_BOOST pin is wired) and 20dBm capability before using "x" or "X". Some other radio modules only wire the PA_BOOST and not the RFO resulting in very bad range when trying to use the RFO mode ("L", "H", and "M"). In this case, one has to use "x" to indicate PA_BOOST usage to get 14dBm.

Want to know more about Semtech SX127x Reference Design? Check for Semtech's AN1200.19 document.

22) Is there a maximum transmission power?

Yes! Because otherwise the transmitted signal will create interferences in a large area.

You have to check the regulation to know what is the maximum transmission power allowed in your country. For instance, in EU, many frequency bands are limited to 14dBm (25mW).

For the previously mentioned 8 predefined channels in the 865-868MHz band, the maximum transmission power is indeed 14dBm.

Want to know more about EU regulations? Check the ETSI EN300-220-1 and ERC/REC 70-03 documents.

23) Finally, what should I do to send a packet to the gateway?

First, both the end-device and the gateway must use the same LoRa mode and frequency channel.

Then, here is an example of a minimal setting for an end-device to actually send packets to a gateway.

```
sx1272.setMode(1);
sx1272.setChannel(CH_10_868);
sx1272.setPower('M');
sx1272.setNodeAddress(6);
sx1272.setPacketType(PKT_TYPE_DATA);
sx1272.sendPacketTimeout(1,"TC/18.56",8);
```

24) How can I request ACK from the gateway?

To requested an ACK from the gateway, use:

```
sx1272.sendPacketTimeoutACK(1,"TC/18.56",8);
```

Both the SNR and RSSI of the uplink packet received at the gateway will be put in the returned downlink ACK packet for range test purposes.

25) Why requesting ACK is costly?

Given the small throughput of LoRa radio, using ACK is not costly in terms of performance. With LoRa technology, having the gateway sending ACKs back to the end-device is costly because the total LoRa radio activity time for any transmitting device is limited. As a gateway is considered as a transmitting device, its radio activity time is also limited. However, a gateway is assumed to handle hundredths or thousands of end-device therefore if ACK are requested everytime, the gateway can very quickly utilize all its allowed radio activity time.

26) What is exactly radio activity time duty-cycled limitation?

In Europe, electromagnetic transmissions in the 868MHz ISM Band used by Semtech's LoRa technology falls into the Short Range Devices (SRD) category. The ETSI EN300-220-1 and ERC/REC 70-03 documents specify various requirements for SRD devices, especially those on radio activity. Basically,

transmitters are constrained to a maximum of 0.1%, 1% or 10% every hour depending on the transmission power and the frequency band. For instance a 1% duty-cycle means 36s of radio activity time per period of 1 hour. This duty cycle limit applies to the total transmission time, even if the transmitter can change to another channel. The rationale for such constraints is to avoid saturating the radio channel as this is an unlicensed band (free for everybody to use as opposed to most of frequency bands used in commercial mobile phone networks).

The ETSI recommendation is also used in other parts of the world. Check for your country.

27) In practice, what is the duty-cycled limitation value?

In the ETSI EN300-220-1 and ERC/REC 70-03 documents, it is mentioned that the 863-870MHz band has 0.1% duty-cycle which represents 3.6s. However, these documents also indicate that if the band is restricted to 865-868MHz, then this duty-cycle limit increases to 1%, i.e. 36s. This is the duty-cycled limitation that you can use under EU regulations as the previously mentioned 8 predefined channels are precisely in the 865-868MHz band.

In most cases, the 36s duty-cycle is largely enough to satisfy communication needs of deployed applications.

28) How can I know the transmission time of my message?

The transmission time, also called time-on-air (ToA), is an important value when there are radio activity time constraints.

With LoRa, the ToA depends on the BW, CR and SF. You can check the Semtech's LoRa FAQ or Semtech's SX127x datasheet documents for the detailed formula used to calculate the ToA when given notably BW, CR, SF and the message size.

With the communication library, you can easily get the ToA of a message under your current LoRa settings by using:

```
sx1272.getToA(my_msg_size+ OFFSET_PAYLOADLENGTH);
```

`OFFSET_PAYLOADLENGTH` is the header size which is normally 4 bytes. The result is expressed in milliseconds. For instance, for a total message size (including the header) of 35 bytes, the ToA with LoRa mode 1 is 1.942s.

29) Who should enforce the duty-cycled limitation?

When deploying a LoRa end-device in production mode, it has to comply with the country's regulation on maximum radio duty-cycled limit, e.g. 36s / hour.

The library is currently not automatically limiting this activity time so it is up to the application programmer to set a behavior that is compatible with the regulations.

For instance, if you need to send periodically a message which total size (including the header) is 35 bytes then you can do so up to 36/1.942=18.53 times per hour. If we round it to 18, then your device can report every 3.33 minutes. If you only need a report every 10 minutes then you are well below the limit. As you can see, in most

cases, the 36s duty-cycle is largely enough to satisfy communication needs of deployed applications.

30) What is LBT+AFA?

When reading ETSI EN300-220-1 and ERC/REC 70-03 documents, you may see a restriction such as " ≤ 1% duty cycle or LBT+AFA". LBT is "Listen Before Talk" which is similar to well-know carrier sense mechanism used in many wireless communications such as WiFi and means that the device listen to check whether there is an ongoing transmission before transmitting. AFA is "Adaptive Frequency Agility" and means that the device is able to change to another frequency if the current frequency has activity, i.e. there is an ongoing transmission.

" ≤ 1% duty cycle or LBT+AFA" means that if you implement LBT+AFA, you are not anymore restricted to the 1% duty cycle but to "an accumulated maximum Tx on-time of 100s within a period of 1 hour" [ETSI EN300-220-1]. However, then there are other hard constraints and one of them being that "The limit for a single transmission TX on-time is 1s". With LoRa mode 1 for maximum range, this is hardly tractable in practice as the transmission of the 4-byte header alone needs already 1s!

31) Why LBT+AFA is not used by the framework?

As the limit for a single transmission is 1s, it is not very interesting when long range (therefore typically using LoRa mode 1) is mainly required.

32) OK, but does it mean that a device does not check for ongoing tx?

No, we do check for ongoing transmission. Our communication library does not use LBT do avoid the 1% duty cycle but to actually check for ongoing transmission. A CSMA-like mechanism with backoff is implemented to prevent an end-device to transmit if some radio activity is detected. The Clear Channel Assessment (CCA) is performed by using both Channel Activity Detection (CAD) functionality of Semtech 127x radio chips and RSSI.

You can activate the CSMA-like mechanism by setting:

```
sx1272._enableCarrierSense=true;
```

and then call:

```
sx1272.CarrierSense();
```

before the call to `sendPacketTimeout()`.

Note that these mechanisms are still in developing and testing phase and the internal behavior may still change to be improved.

33) Is the 915MHz band supported?

Although the 915MHz band is supported (mostly for the US ISM band), the library does not implement the frequency hopping mechanism.

34) Where can I get all the documentation?

1. The DIY low-cost LoRa gateway is described at:
   http://cpham.perso.univ-pau.fr/LORA/RPIgateway.html

2. The github repository with all the source code, examples and documentation for devices and gateway can be accessed at:
   https://github.com/CongducPham/LowCostLoRaGw

3. Step-by-step tutorials on how to build low-cost IoT devices and gateways can be downloaded from:
   https://github.com/CongducPham/tutorials

4. Information on the WAZIUP project can be found at:
   http://www.waziup.eu

35) Finally, my gateway is still not receiving anything, why?

There can be several reasons.

First, check that both the end-device and the gateway use the same LoRa mode and frequency channel. For the gateway, look at the gateway documentation to see how to run it with a specific mode and frequency channel. If you use the provided examples without changing anything, by default both the end-devices and the gateway use LoRa mode 1 and CH_10_868.

At the end-device, use the serial monitor of the Arduino IDE to check that all the configuration steps of the radio module went OK: all returned state values are 0. Check the connection pins, especially the Chip Select (CS) pin. For the moment, for a board that IS NOT an Arduino Pro Mini, Mini, Nano or Teensy (actually for boards with Uno-compatible pin layout) the CS pin on the board is assigned to pin 2. For the Arduino Pro Mini, Mini (and Ideetron Nexus), Nano or Teensy (actually for small form factor boards), it is pin 10. The reason is because originally the library was written by Libelium for their multi-protocol radio shield that can be connected to an Uno-compatible pin layout board. The multi-protocol radio shield uses pin 2 for connecting the CS line. In order to keep compatibility with this board in case one might use it, our improved version of the library keeps pin 2 for CS. For small-form factor boards, it is not possible to use the multi-protocol radio shield so there is no point in using pin 2. Therefore, pin 10, which is the "standard" CS pin is used instead.

There might also be current issues. The radio module can draw more than 50mA when transmitting (depending on the programmed transmission power) so the current limit of most microcontroller's analog/digital pins is reached. See for instance http://playground.arduino.cc/Main/ArduinoPinCurrentLimitations. It is better to connect the radio module VCC to a dedicated 3.3v pin that usually has higher current limit. However, it may be still not enough on some boards if you use PA_BOOST at 20dBm for your radio module. So be careful. For the MEGA, Due, and Teensy, they have a 3.3v pin with higher current limit so there should not be current issues with these boards.

One frequent issue is that some radio modules need the PA_BOOST even for 14dBm transmission because the RFO pins are just not connected. This is for instance the case for the HopeRF RFM92W/95W and the Modtronix inAir9B. In these cases, use `setPower("x")` instead of `setPower("M")`.