Plug & Sense! Smart Parking

Technical Guide

libelium

WASP MOTE

INDEX

1. Introduction

The aim of this manual is to introduce the user to Smart Parking in a practical way.

This document applies to the following Smart Parking model, approved for FCC:

Model	FCC ID
Smart Parking US	XKM-PARKING-V1

The new version of **Waspmote Plug & Sense! Smart Parking**, the solution for Smart Cities that allows citizens **to detect available parking spots.**

The new device is **easier and cheaper to deploy** as it is mounted on the road surface. Unlike most market versions, it does not need to dig a hole in the ground for installation, reducing installation time from 30 to 5 minutes and allowing to be replaced by another unit in case of maintenance in just 10 minutes.

The smaller size -reduced over 50%-, its higher accuracy and reliability, and the faster time of detection, besides the independence from temperature are also important features of new Smart Parking device.

New sensor system is **fully compatible with LPWAN** radio technologies **-LoRaWAN** and **Sigfox-** to enable long range and low power consumption. It can be connected with both radios for the European 868 MHz band and for the 900-930 MHz band (US / Canada). One unique feature of the system is that it allows to use both radio technologies at the same time or changing from one to the other using the manager system from the Cloud.

With the new sensor system, one base station can give service to thousands of devices around a range of several kilometers in urban environment. This fact provides lower costs of installation since the number of **base stations can be dramatically reduced**. Besides, the new sensor model has been optimized for really low-power operation, so the battery lifetime is extended **up to 10 years** easily.

The new Smart Parking node has been granted with the CE / FCC marks and provides a **robust software which works out-of-the-box**. Developers do not have to cope with programming the nodes, they just have to specify the values of key parameters in the firmware such as working cycle or night mode to be ready to work. **Remote management and bidirectional communicatio**n allows to change several parameters of the nodes from the Cloud. This means we can reprogram thousands of nodes by just setting the right values from our web browser in the management platform.

2. General

2.1. General and safety information

- In this section, the term “Waspmote” encompasses both the Waspmote device itself and it enclosure.
- Read through the document “General Conditions of Libelium Sale and Use”.
- Do not allow contact of metallic objects with the electronic part to avoid injuries and burns.
- NEVER submerge the device in any liquid with the enclosure open.
- Keep the device in a dry place and away from any liquid which may spill.
- Waspmote consists of highly sensitive electronics which is accessible to the exterior, handle with great care and avoid bangs or hard brushing against surfaces.
- Check the product specifications section for the maximum allowed power voltage and amperage range and consequently always use a current transformer and a battery which works within that range. Libelium is only responsible for the correct operation of the device with the batteries, power supplies and chargers which it supplies.
- Keep the device within the specified range of temperatures in the specifications section.
- Do not connect or power the device with damaged cables or batteries.
- Place the device in a place only accessible to maintenance personnel (a restricted area).
- Keep children away from the device in all circumstances.
- If there is an electrical failure, disconnect the main switch immediately and disconnect that battery or any other power supply that is being used.
- If a hardware failure occurs, consult the Libelium Web Development section.
- Check that the frequency and power of the communication radio modules together with the integrated antennas are allowed in the area where you want to use the device.

2.2. Conditions of use

- Read the “General and Safety Information” section carefully and keep the manual for future consultation.
- Use Waspmote in accordance with the electrical specifications and the environment described in the “Hardware” section of this manual.
- Do not place Waspmote in contact with metallic surfaces; they could cause short-circuits which will permanently damage it.
- IMPORTANT It is the responsibility of the installer to find out about restrictions of use for frequency bands in each country and act in accordance with the given regulations. Libelium Comunicaciones Distribuidas S.L does not list the entire set of standards that must be met for each country.
- For further information go to:
  - CEPT ERC 70-03E - Technical Requirements, European restrictions and general requirements: http://www.erodocdb.dk/
  - R&TTE Directive - Equipment requirements, placement on market: http://www.erodocdb.dk/
- Further information you may need can be found at: http://www.libelium.com/development/waspmote
- The “General Conditions of Libelium Sale and Use” document can be found at:
  http://www.libelium.com/development/waspmote/technical_service

3. Hardware

3.1. Hardware description

Figure : Plug & Sense! Smart Parking

Power supply	Built-in Lithium batteries, expected lifetime of 4-6 years*
Antenna	Included
Detection	Magnetic
Mounting	Over the floor
Dimensions	230 mm diameter, 28 mm height
Protection	IP68 strictly under right closing (see section "Device installation")
Operating temperature	-20 to +65 °C

** Under normal circumstances and dependent on settings*

Figure : Plug & Sense! Smart Parking main characteristics

3.2. Power consumption

	Consumption
Measuring sensor	*TBD*
Transmission Sigfox	*TBD*
Transmission LoRaWAN	*TBD*
Sleep state	25 µA
Battery self discharge	< 1% month at +20 ºC
Battery type	Lithium non-rechargeable battery, 3.6 V, 10.4 A·h

Figure : Plug & Sense! Smart Parking power consumption

4. How the node works

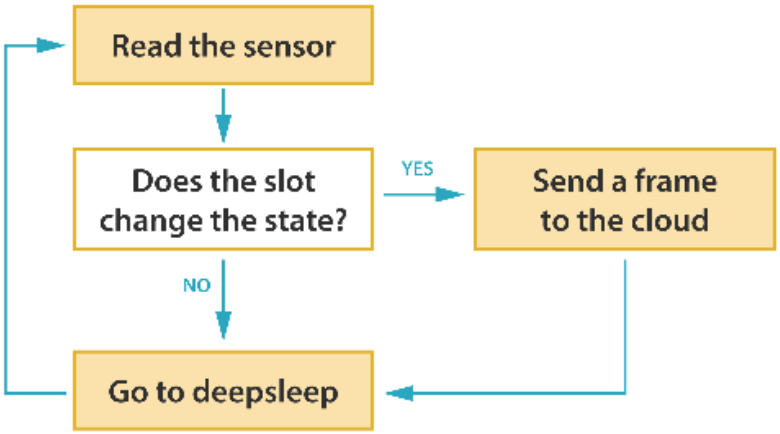


Figure : Basic working loop diagram

As the diagram indicate, the basic loop of the node consists in read the sensor and send a frame when the parking slot change it status. Then, it sleeps a desired time and starts the loop again.

Some events can forced the node to send a frame to the cloud. If a desired time elapsed since the last radio transmission, the node will send a Keep-Alive frame. This frame only contains basic data from the node (parking slot status and battery status). It is useful to know that there is no changes in the slot, but the node still working. The node also will send a frame each 24 hours with the working data of the day (number of measurements, number of transmissions,...).

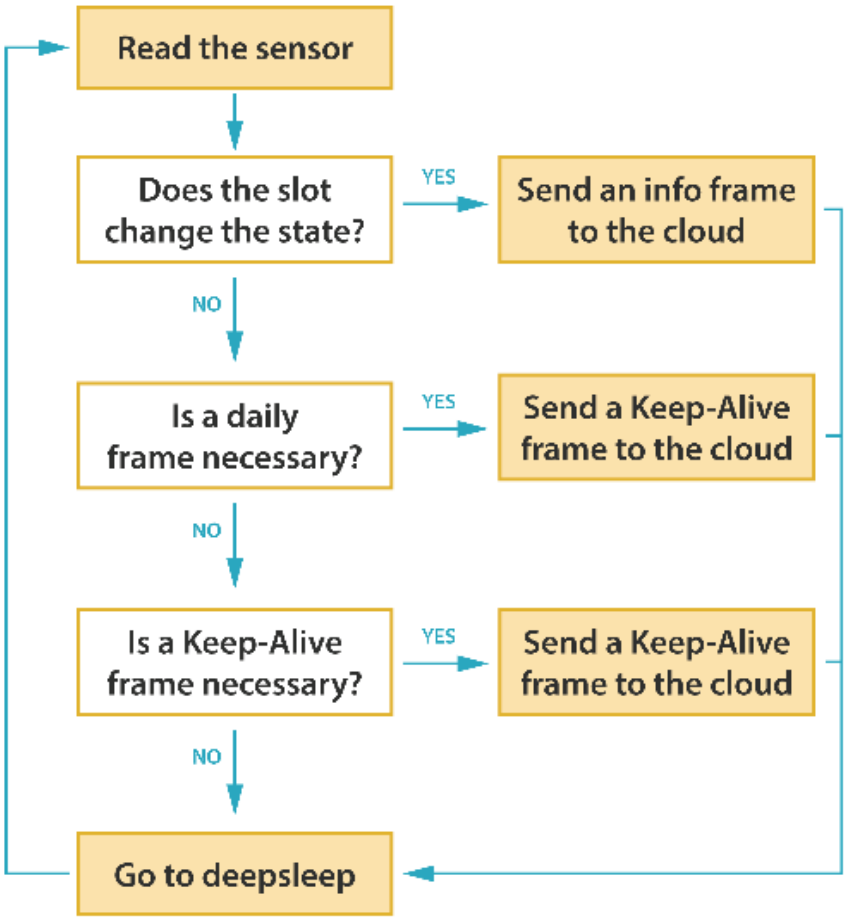


Figure : Extended loop diagram

5. Sleep modes

Plug & Sense! Smart Parking has 2 sleep modes: day mode and night mode. The second one has been developed to use when the parking slot is expected to have fewer changes (i.e. at night). Each mode has its own configuration parameters. The figure below shows an example for the node transmissions in a day. The time zone between 6 AM and 12 AM (in light gray) indicates that the node is working in day mode. In this mode, the sampling of the parking slot is made more regular (1 minute) and the Keep-Alive is only 2 hours. In the dark gray zone, from 12 AM to 6 AM, the node is working in night mode. As is shown in the example, the sampling time is greater (10 minutes) and the Keep-Alive increases too (3 hours).

Example configuration:

Parameter	Configuration
Sleep time	1 minute
Keep-Alive	2 hours
Night Mode start hour	00 hours
Night Mode duration	6 hours
Night Mode Sleep Time	10 minutes
Night Mode Keep-Alive	3 hours

Figure : Example configuration

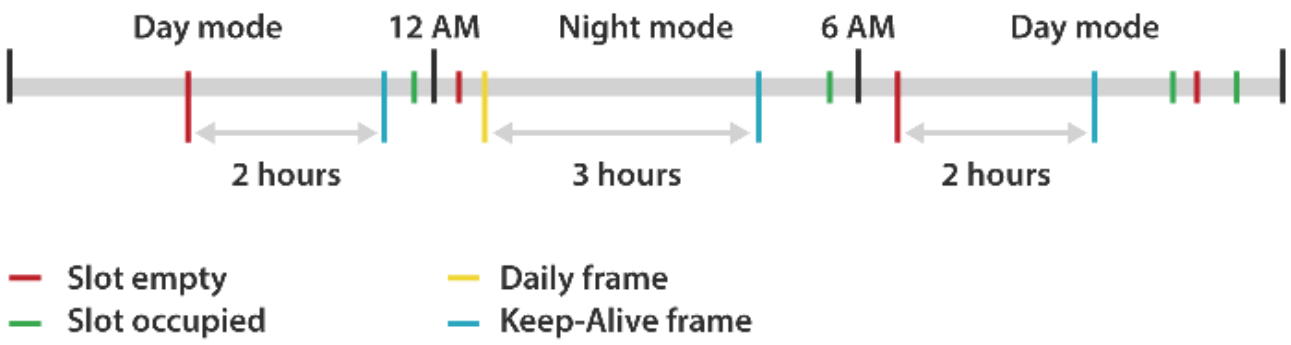


Figure : Day and night modes example

5.1. Day mode

It is the basic working mode and it has 2 configurable parameters:

- **Sleep time:** Sleep time between consecutive sensor measurements. 2 minutes option is configured by default.
- **Keep-Alive:** Elapsed time since last transmission to trigger a Keep-Alive frame. 10 hours option is configured by default. This frame only contains basic data from the node (parking slot status and battery status). It is useful to know that there are no changes in the slot, but the node still working. This mode can be disabled using both the USB Programmer or the Remote Manager.

5.2. Night mode

This mode has been developed to use when the parking slot is expected to have fewer changes (i.e. at night). It has 4 configurable parameters:

- **Night Mode start hour:** Beginning hour of the night mode. 22 hours option is configured by default.
- **Night Mode duration:** Night mode duration time. 8 hour option is configured by default.
- **Night Mode Sleep Time:** Sleep time between consecutive sensor measurements (during night mode). 10 minutes option is configured by default.
- **Night Mode Keep-Alive:** Elapsed time since last transmission to trigger a Keep-Alive frame (during night mode). 4 hours option is configured by default. This frame only contains basic data from the node (parking slot status and battery status). It is useful to know that there is no change in the slot, but the node is still working.

This mode can be disabled using both the USB Programmer or the Remote Manager.

6. Transmission modes

Plug & Sense! Smart Parking has 2 transmission modes allowing the user to choose between Sigfox, LoRaWAN:

- **Sigfox.** This mode only uses the Sigfox radio to send the data collected by the node. This mode is selected by default.
- **LoRaWAN.** This mode only uses the LoRaWAN radio to send the data collected by the node.

7. Frames

7.1. Firmware version v2.x.x

Plug & Sense! Smart Parking node can send 6 defined frames. All frames are 11 bytes length and they are the same for Sigfox and LoRaWAN. Bytes 0 and 1 are common for all frames. Byte 0 has the basic information of the node and frame.

Bit	Name	Description
7	Parking slot status	'0' indicates that the parking slot is empty
		'1' indicates that the slot is in occupied
6	Battery state	'0' indicates that the battery has a good level of charge
		'1' indicates that the battery has little charge and it will be necessary to change it. When the battery has little charge it is possible that the node does not work properly and the radios fail sending the frames.
5-4	Reserved	Reserved bits. Do not consider.
3	Frame type	0 – Info frame
2		1 – Keep-Alive frame
1		2 – Daily update frame
0		3 – Error frame 4 – Start frame 1 5 – Start frame 2 Values from 6 to 15 are reserved Values from 6 to 15 are reserved

Figure : Byte 0 description

Byte 1 is a frame counter, it goes from 0 to 255. This byte can be used to detect lost frames (sent by the node but not received).

7.1.1. Info frame

It is the most common frame sent by the node. The node will send this kind of frame each time it detects that the parking slot changed from empty to occupied or vice-versa. The other bytes are used to give additional data to the user.

Byte	Name	Description
0	Basic data	Detailed description in the section “Frame”
1	Frame counter	Detailed description in the section “Frame”
2	Temperature	Temperature (Celsius degrees) from the node's internal sensor. The value of temperature is a signed integer.
3	X axis measurement MSB	Raw value from the sensor associated to the X axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
4	X axis measurement LSB	
5	Y axis measurement MSB	Raw value from the sensor associated to the Y axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
6	Y axis measurement LSB	
7	Z axis measurement MSB	Raw value from the sensor associated to the Z axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
8	Z axis measurement LSB	
9-10	Reserved	Reserved bytes. Do not consider.

Figure : Info frame structure

7.1.2. Keep-Alive frame

This frame is used to indicate that the parking slot has not changed, but the node is still working.

Byte	Name	Description
0	Basic data	Detailed description in section “Frame”
1	Frame counter	Detailed description in section “Frame”
2	Timestamp (hh)	Current hours
3	Timestamp (mm)	Current minutes
4	Temperature	Temperature (Celsius degrees) from the node's internal sensor. The value of temperature is a signed integer.
5	X axis measurement MSB	Raw value from the sensor associated to the X axis. The value stored in these two bytes is a 16-bit value in 2’s complement form,.
6	X axis measurement LSB	
7	Y axis measurement MSB	Raw value from the sensor associated to the Y axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
8	Y axis measurement LSB	
9	Z axis measurement MSB	Raw value from the sensor associated to the Z axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
10	Z axis measurement LSB	

Figure : Keep-Alive frame structure

7.1.3. Daily update frame

This frame is sent daily at 1 AM. It contains a little summary.

Byte	Name	Description
0	Basic data	Detailed description in section “Frame”
1	Frame counter	Detailed description in section “Frame”
2	Sensor measurements MSB	Unsigned 16 bit counter. It stores the times that the sensor is used in the last 24 hours.
3	Sensor measurements LSB	
4	Sigfox transmissions MSB	Unsigned 16 bit counter. It stores the times that Sigfox radio is used in the last 24 hours.
5	Sigfox transmissions LSB	
6	LoRaWAN transmissions MSB	Unsigned 16 bit counter. It stores the times that LoRaWAN radio is used in the last 24 hours.
7	LoRaWAN transmissions LSB	
8	Resets	Number of resets generated in the last 24 hours
9	Config_id	Value of the configuration version loaded into the node
10	Reserved	Reserved bytes. Do not consider.

Figure : Daily update frame structure

This frame can be deactivated using the Plug & Sense! Smart Parking USB Programmer or via radio, with the Remote Manager, setting to 0 the enable/disable daily frame bit.

The daily update frame is very special because the node waits for a response after it is sent. This response is useful for reconfiguring the node “over the air”, without physical access. Also, a second use of this response frame is to synchronize the node’s internal clock, thanks to a timestamp. This response can be configured using the remote PHP.

7.1.4. Error frame

In some cases the node could send a frame if some internal components or processes fail.

Byte	Name	Description
0	Basic data	Detailed description in section “Frame”
1	Frame counter	Detailed description in section “Frame”
2	Error data	Detailed description below
3	Temperature	Temperature (Celsius degrees) from the node's internal sensor. The value of temperature is a signed integer.
4	X axis measurement MSB	Raw value from the sensor associated to the X axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
5	X axis measurement LSB	
6	Y axis measurement MSB	Raw value from the sensor associated to the Y axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
7	Y axis measurement LSB	
8	Z axis measurement MSB	Raw value from the sensor associated to the Z axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
9	Z axis measurement LSB	
10	Battery level	Battery voltage in millivolts. To convert to millivolts use the next formula: $Battery\ voltage(mV)=((Battery\ level)\cdot 4)+2800$

Figure : Error frame structure

Bit	Name	Description
7-6	Reserved	Reserved bits. Do not consider.
5	Error Sigfox	Set to '1' when an error related with the Sigfox radio is detected. Clear when no issues detected.
4	Error LoRaWAN	Set to '1' when an error related with the LoRaWAN radio is detected. Clear when no issues detected.
3	Error RTC	Set to '1' when an error related with the RTC (internal clock) is detected. Clear when no issues detected.
2	Error X axis	Set to '1' when an error appears in the X axis of the sensor. Clear when no issues detected.
1	Error Y axis	Set to '1' when an error appears in the Y axis of the sensor. Clear when no issues detected.
0	Error Z axis	Set to '1' when an error appears in the Z axis of the sensor. Clear when no issues detected.

Figure : “Error data” byte structure

7.1.5. Start frames

When the node starts to work in the parking slot, it will send 2 frames. The first one is dedicated to the sensor and the battery. The second one is used to send some parameters about the chosen configuration.

7.1.5.1. Start frame number 1

Byte	Name	Description
0	Basic data	Detailed description in section “Frame”
1	Frame counter	Detailed description in section “Frame”
2	Temperature	Temperature (Celsius degrees) from the node's internal sensor. The value of temperature is a signed integer.
3	X axis reference MSB	Reference value from the sensor associated to the X axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
4	X axis reference LSB	
5	Y axis reference MSB	Reference value from the sensor associated to the Y axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
6	Y axis reference LSB	
7	Z axis reference MSB	Reference value from the sensor associated to the Z axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
8	Z axis reference LSB	
9	Battery voltage MSB	Battery voltage in millivolts. The value stored in these two bytes is an unsigned 16-bit value.
10	Battery voltage LSB	

Figure : Start frame number 1 structure

7.1.5.2. Start frame number 2

Byte	Name	Description
0	Basic data	Detailed description in section “Frame”
1	Frame counter	Detailed description in section “Frame”
2	FIRMWARE_VERSION	Firmware version
3	NM_START	Beginning hour of the night mode
4	NM_PERIOD	Duration in hours of the night mode
5	NM_SLEEP_TIME	Sleep time between consecutive sensor measurements (during night mode)
6	NM_KEEP_ALIVE	Elapsed time since last transmission to trigger a Keep-Alive frame (during night mode)
7	RADIO_MODE	Selected transmission mode between Sigfox, LoRaWAN y their combinations
8	SLEEP_TIME	Sleep time between consecutive sensor measurements
9	KEEP_ALIVE	Elapsed time since last transmission to trigger a Keep-Alive frame
10	THRESHOLD	Threshold for detecting a vehicle over the parking slot

Figure : Start frame number 2 structure

7.2. Firmware version v1.x.x

Plug & Sense! Smart Parking node can send 6 defined frames. All frames are 12 bytes length and they are the same for Sigfox and LoRaWAN. Bytes 0 and 1 are common for all frames. Byte 0 has the basic information of the node, and frame and byte 1 is a frame counter. It can be used to detect lost frames.

Bit	Name	Description
7	Parking slot status	'0' indicates that the parking slot is empty '1' indicates that the slot is in occupied
6	Battery state	'0' indicates that the battery has a good level of charge '1' indicates that the battery has little charge and it will be necessary to change it. When the battery has little charge it is possible that the node does not work properly and the radios fail sending the frames.
5-4	Reserved	Reserved bits. Do not consider.
3 2 1 0	Frame type	0 – Info frame 1 – Keep-Alive frame 2 – Daily update frame 3 – Error frame 4 – Start frame 1 5 – Start frame 2 Values from 6 to 15 are reserved

Figure : Byte 0 description

7.2.1. Info frame

It is the most common frame sent by the node. The node will send this kind of frame each time it detects that the parking slot changed from empty to occupied or vice-versa. The other bytes are used to give additional data to the user.

Byte	Name	Description
0	Basic data	Detailed description in the section “Frame”
1	Frame counter	Detailed description in the section “Frame”
2	Temperature MSB	Raw temperature from the node's internal sensor. The value stored in these two bytes is a 16-bit value in 2’s complement form. To convert to Celsius degrees use the next formula: $Temperature(ºC) = \frac{(MSB \cdot 2^8 + LSB)}{8} + 25$
3	Temperature LSB	
4	X axis measurement MSB	Raw value from the sensor associated to the X axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
5	X axis measurement LSB	
6	Y axis measurement MSB	Raw value from the sensor associated to the Y axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
7	Y axis measurement LSB	
8	Z axis measurement MSB	Raw value from the sensor associated to the Z axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
9	Z axis measurement LSB	
10-11	Reserved	Reserved bytes. Do not consider.

Figure : Info frame structure

7.2.2. Keep-Alive frame

This frame is used to indicate that the parking slot has not changed, but the node is still working.

Byte	Name	Description
0	Basic data	Detailed description in section “Frame”
1	Frame counter	Detailed description in section “Frame”
2	Timestamp (hh)	Current hours
3	Timestamp (mm)	Current minutes
4	Temperature MSB	Raw temperature from the node’s internal sensor. The value stored in these two bytes is a 16-bit value in 2’s complement form. To convert to Celsius degrees use the next formula: $Temperature(^{\circ}C)=\frac{(MSB\cdot 2^8+LSB)}{8}+25$
5	Temperature LSB	
6	X axis measurement MSB	Raw value from the sensor associated to the X axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
7	X axis measurement LSB	
8	Y axis measurement MSB	Raw value from the sensor associated to the Y axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
9	Y axis measurement LSB	
10	Z axis measurement MSB	Raw value from the sensor associated to the Z axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
11	Z axis measurement LSB	

Figure : Keep-Alive frame structure

7.2.3. Daily update frame

This frame is sent daily at 1 AM. It contains a little summary.

Byte	Name	Description
0	Basic data	Detailed description in section “Frame”
1	Frame counter	Detailed description in section “Frame”
2	Sensor measurements MSB	Unsigned 16 bit counter. It stores the times that the sensor is used in the last 24 hours.
3	Sensor measurements LSB	
4	Sigfox transmissions MSB	Unsigned 16 bit counter. It stores the times that Sigfox radio is used in the last 24 hours.
5	Sigfox transmissions LSB	
6	LoRaWAN transmissions MSB	Unsigned 16 bit counter. It stores the times that LoRaWAN radio is used in the last 24 hours.
7	LoRaWAN transmissions LSB	
8	Resets	Number of resets generated in the last 24 hours
9	Config_id	Value of the configuration version loaded into the node
10-11	Reserved	Reserved bytes. Do not consider.

Figure : Daily update frame structure

This frame can be deactivated using the Plug & Sense! Smart Parking USB Programmer or via radio, with the Remote Manager, setting to 0 the enable/disable daily frame bit.

The daily update frame is very special because the node waits for a response after it is sent. This response is useful for reconfiguring the node “over the air”, without physical access. Also, a second use of this response frame is to synchronize the node’s internal clock, thanks to a timestamp. This response can be configured using the remote PHP.

7.2.4. Error frame

In some cases the node could send a frame if some internal components or processes fail.

Byte	Name	Description
0	Basic data	Detailed description in section “Frame”
1	Frame counter	Detailed description in section “Frame”
2	Error data	Detailed description below
3	Temperature MSB	Raw temperature from the parking internal sensor. The value stored in these two bytes is a 16-bit value in 2’s complement form. To convert to Celsius degrees use the next formula: $Temperature(ºC)=\frac{(MSB\cdot 2^8+LSB)}{8}+25$
4	Temperature LSB	
5	X axis measurement MSB	Raw value from the sensor associated to the X axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
6	X axis measurement LSB	
7	Y axis measurement MSB	Raw value from the sensor associated to the Y axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
8	Y axis measurement LSB	
9	Z axis measurement MSB	Raw value from the sensor associated to the Z axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
10	Z axis measurement LSB	
11	Battery level	Battery voltage in millivolts. To convert to millivolts use the next formula: $Battery\ voltage(mV)=((Battery\ level)\cdot 4)+2800$

Figure : Error frame structure

Bit	Name	Description
7-6	Reserved	Reserved bits. Do not consider.
5	Error Sigfox	Set to '1' when an error related with the Sigfox radio is detected. Clear when no issues detected.
4	Error LoRaWAN	Set to '1' when an error related with the LoRaWAN radio is detected. Clear when no issues detected.
3	Error RTC	Set to '1' when an error related with the RTC (internal clock) is detected. Clear when no issues detected.
2	Error X axis	Set to '1' when an error appears in the X axis of the sensor. Clear when no issues detected.
1	Error Y axis	Set to '1' when an error appears in the Y axis of the sensor. Clear when no issues detected.
0	Error Z axis	Set to '1' when an error appears in the Z axis of the sensor. Clear when no issues detected.

Figure : “Error data” byte structure

7.2.5. Start frames

When the node starts to work in the parking slot, it will send 2 frames. The first one is dedicated to the sensor and the battery. The second one is used to send some parameters about the chosen configuration.

7.2.5.1. Start frame number 1

Byte	Name	Description
0	Basic data	Detailed description in section “Frame”
1	Frame counter	Detailed description in section “Frame”
2	Temperature MSB	Raw temperature from the parking internal sensor. The value stored in these two bytes is a 16-bit value in 2’s complement form. To convert to Celsius degrees use the next formula: $Temperature(^{\circ}C) = \frac{(MSB \cdot 2^8 + LSB)}{8} + 25$
3	Temperature LSB	
4	X axis reference MSB	Reference value from the sensor associated to the X axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
5	X axis reference LSB	
6	Y axis reference MSB	Reference value from the sensor associated to the Y axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
7	Y axis reference LSB	
8	Z axis reference MSB	Reference value from the sensor associated to the Z axis. The value stored in these two bytes is a 16-bit value in 2’s complement form.
9	Z axis reference LSB	
10	Battery voltage MSB	Battery voltage in millivolts. The value stored in these two bytes is an unsigned 16-bit value.
11	Battery voltage LSB	

Figure : Start frame number 1 structure

7.2.5.2. Start frame number 2

Bit	Name	Description
0	Basic data	Detailed description in section “Frame”
1	Frame counter	Detailed description in section “Frame”
2	CODE_ID	Firmware version
3	NM_START	Beginning hour of the night mode
4	NM_PERIOD	Duration in hours of the night mode
5	NM_SLEEP_TIME	Sleep time between consecutive sensor measurements (during night mode)
6	NM_KEEP_ALIVE	Elapsed time since last transmission to trigger a Keep-Alive frame (during night mode)
7	RADIO_MODE	Selected transmission mode between Sigfox, LoRaWAN y their combinations
8	SLEEP_TIME	Sleep time between consecutive sensor measurements
9	KEEP_ALIVE	Elapsed time since last transmission to trigger a Keep-Alive frame
10	THRESHOLD	Threshold for detecting a vehicle over the parking slot
11	Reserved	Reserved byte. Do not consider.

Figure : Start frame number 2 structure

8. Smart Devices App

Libelium Smart Devices App is an important tool developed by Libelium that allows users install new firmware versions and program the configuration of the new Libelium devices in a few clicks. At the moment it is only available for Smart Parking and MySignals products, but the list will be incremented shortly.

8.1. Installation

First of all and before installing anything, users have to take into account the platform where the application is going to be installed. To install the Libelium Smart Devices App, it is compulsory to have installed the JDK 1.8. If it is not installed in the computer, you can follow the steps and download it from this website:
https://docs.oracle.com/javase/8/docs/technotes/guides/install/install_overview.html

Once installed JDK, users can download the application using the appropriate link depending on the operative system:

- **Ubuntu:** http://downloads.libelium.com/smart_device_app/SmartDeviceApp_linux64.zip
- **Windows:** http://downloads.libelium.com/smart_device_app/SmartDeviceApp_windows32.zip
- **Mac:** http://downloads.libelium.com/smart_device_app/SmartDeviceApp_macosx64.zip

Then customers only have to extract the content of the SmartDeviceApp zip file downloaded in a place with the right permissions, and finally execute the file called “SmartDeviceApp” that will initialize the application. Please, note that the extension of this file will depend on the operating system the user is using at the moment (.sh for Linux and OSX, and .bat for Windows).

8.2. Smart Parking

This section provides several options to Smart Parking users in order to take full advantage of all possibilities the devices offers.

8.2.1. Programmer

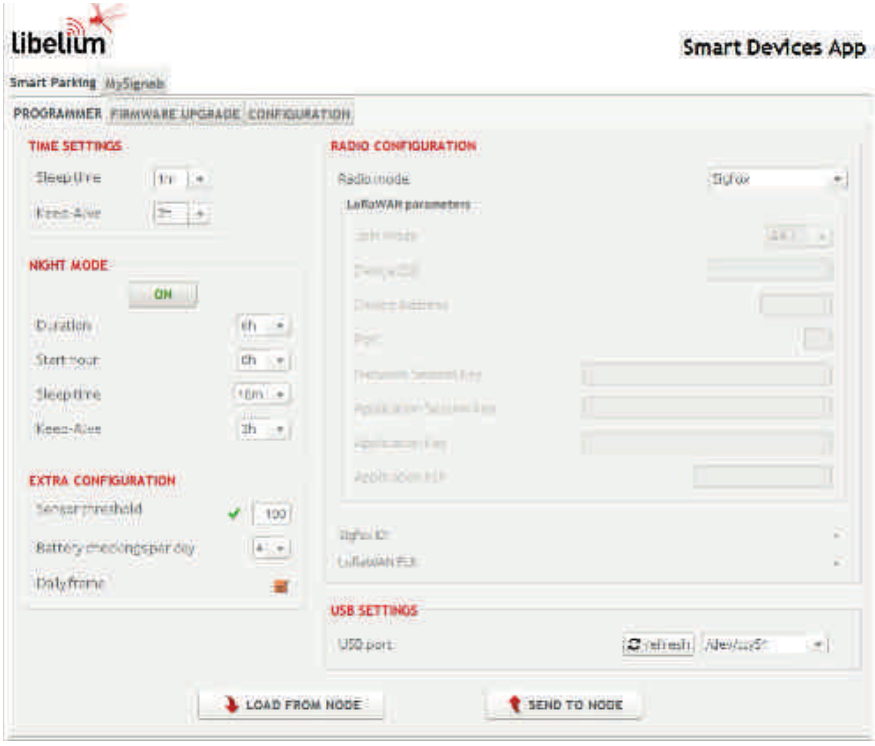


Figure : Smart Parking configuration form

Users can read and write all configuration parameters of their devices in this section. The process is quite simple. Just connect the device to the computer where the Smart Devices App is installed using the USB cable provided and switch the node on.

Then, refresh the “USB settings” block which is in the bottom-right corner, clicking in “refresh” button. Once done it, the port where the device has been connected must be selected.

Figure : USB settings

But before configuring the device, it is very important that users take in consideration the following list where all parameters are explained:

TIME SETTINGS

- **Sleep time:** Sleep time between consecutive sensor measurements.
- **Keep-Alive:** Elapsed time since last transmission to send a Keep-Alive frame.

NIGHT MODE

- **On/Off button:** Button to activate/deactivate this option. If it is not active, the following fields will not be effective.
- **Duration:** Night mode duration time.
- **Start hour:** Night Mode start hour.
- **Sleep time:** Sleep time between consecutive sensor measurements (during night mode).
- **Keep-Alive:** Elapsed time since last transmission to send a Keep-Alive frame (during night mode).

EXTRA CONFIGURATION

- **Sensor Threshold:** Threshold for detecting a vehicle over the parking slot.
- **Battery readings per day:** Battery readings per day.
- **Daily frame:** Enable/Disable daily frame sending.

RADIO MODE

- **Radio mode:** Radio transmission mode among Sigfox, LoRaWAN or their combinations.

LoRaWAN parameters

- **LoRaWAN join mode:** LoRaWAN join mode, ABP or OTAA.
- **Device EUI:** LoRaWAN device EUI.
- **Device Address:** LoRaWAN device address.
- **Port:** LoRaWAN port.
- **Network Session Key:** LoRaWAN network session key.
- **Application Session Key:** LoRaWAN application session key.
- **Application Key:** LoRaWAN application key.
- **Application EUI:** LoRaWAN application EUI.

Sigfox ID: Sigfox ID that will be loaded from the device.

- **LoRaWAN EUI:** LoRaWAN EUI that will be loaded from the device.

USB SETTINGS

- **USB Port:** In this list will be displayed all available USB ports to work out with the device. If you plug your device and the port is not listed, you have click on “Refresh” button in order to update the list.

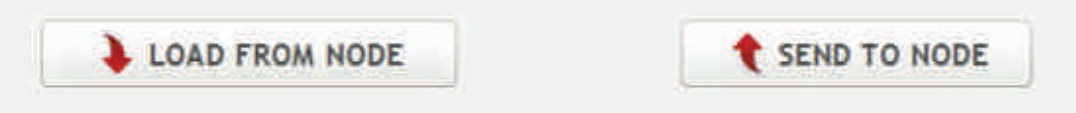


Figure : Load configuration from node & Send configuration to node buttons

The “Load from node” button will read all parameters from the node and will display the information in the form. On the other hand, the “Send to node” button will overwrite the configuration in the node. All available fields have to be filled, with the proper format. If any parameter does not have an acceptable format, a red cross like this ✘ will be displayed near it and you cannot write the information in the node. If the information introduced is valid, a green tick ✔ will be shown.

8.2.2. Firmware upgrade

In this tab, users can select the firmware version to install in their devices.

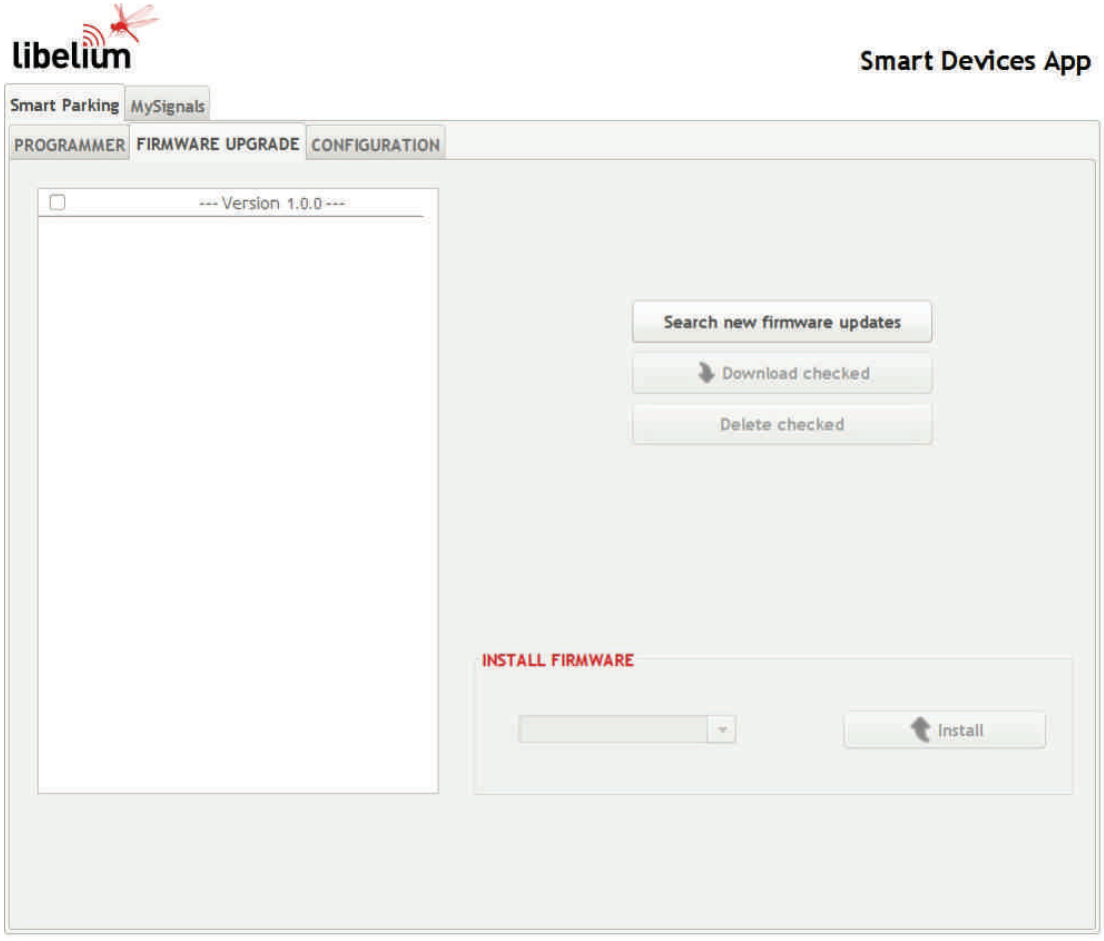


Figure : Smart Parking firmware upgrade form

The list with all available firmware is loaded when the program starts, but users can update it on demand, clicking on “Search new firmware updates” button. Before installing the firmware, it is necessary to download it. This process is very simple, just mark the check of the version you want to install from the list ✔ and click on “Download checked” button.

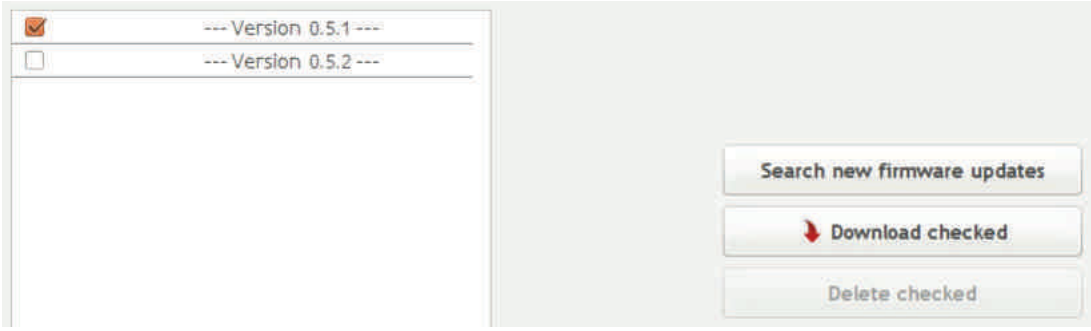


Figure : Download the firmware selected

When this item is downloaded, a disk will be displayed near it, indicating it is downloaded. Once the firmware is downloaded, it is ready to install using the “Install Firmware” section at the bottom. In the drop-down will appear all downloaded versions. Select one and then hit on “Install” button.

Figure : Install the firmware selected

You can also delete the downloaded firmware marking the check from the list ✔ and then clicking on “Delete checked” button.

Figure : Delete downloaded firmware

Remember that the USB port must be selected in the programmer tab.

8.2.3. Configuration

In the last tab, “Configuration”, all external parameters that the software uses to work will be displayed. Users can modify these values in order to get the wished application behavior.

In this case, there is only one parameter available, the API key to connect to Libelium Cloud. This value is provided by Libelium and it is very important to control the access and get some results needed in the programmer tab. If users do not fill this field, the software does not work.

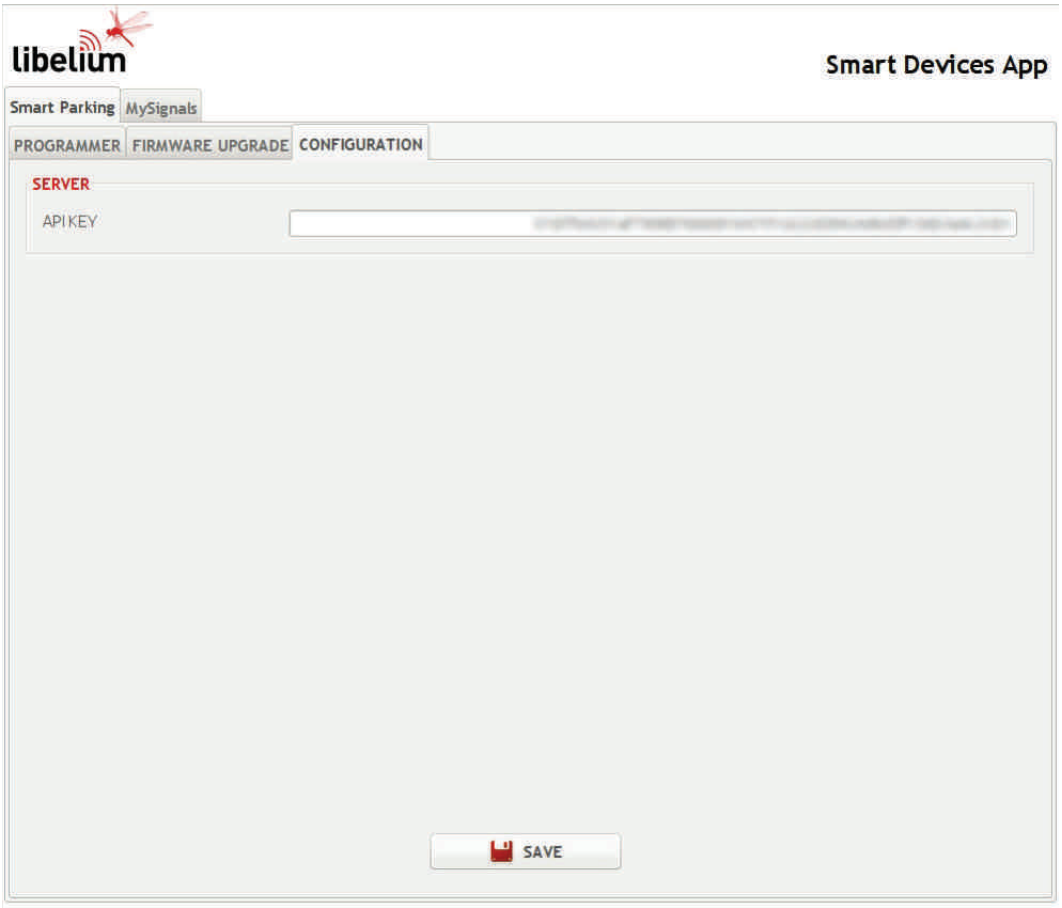


Figure : Smart Parking configuration form

9. Callback Server

Sigfox and LoRaWAN callback service requirements include a server with a web application up and running, this web application will receive Sigfox and LoRaWAN requests. Sigfox and LoRaWAN callback service will relay messages via POST/GET requests to your web application running in your server.

Libelium provides the source files of a simple web application to deploy in your server. This remote node configuration web application provided by Libelium will receive Sigfox and LoRaWAN requests, sending a response back with the Smart Parking node configuration.

The remote node configuration web application includes a simple web form to manage the configuration values of the Smart Parking nodes, and a background process to deal with the Sigfox and LoRaWAN callback services requests. The background process will deliver the proper replies with the configuration values for each radio ID, previously stored using the web form.

The remote node configuration web application should be deployed in your server and the callback services must be configured with the complete URL containing your server domain name or IP, and the context pointing to the web app deployment path.

Example URL: https://my_server.com/path/zip/extracted

NOTE: *Customers have to ask for this source code to Libelium Sales Department after buying the nodes.*

9.1. Installation

Assigning a public IP and a registered domain name to the server is recommended to ease the task of configuring Sigfox and LoRaWAN callback services to send HTTP/HTTPS requests to this server. It is also a good practice to implement existing security policies (user credentials, SSL, firewall, and tools to avoid DoS attacks) for servers with open ports to Intermet.

Apache web server with PHP support must be configured in your server, those are the minimun requirements to deploy the remote node configuration web application. Libraries for Databases (MySQL, PostgreSQL) are optional, the remote node configuration web application saves the Smart Parking configuration values in text files but could be extended to implement connections to any other DB storage.

Info and tutorials about installation of minimun requirements in your server:

- **Ubuntu**: https://help.ubuntu.com/community/ApacheMySQLPHP
- **Windows**: http://www.ampsoft.net/webdesign-l/how-to-install-apache-php-mysql.html
- **Mac**: http://jason.pureconcepts.net/2012/10/install-apache-php-mysql-mac-os-x/

Loriot LoRaWAN callback service has extra requirements, Node.js and NPM libraries are needed:

- **Ubuntu**: http://www.hostingadvice.com/how-to/install-nodejs-ubuntu-14-04/
- **Windows**: http://blog.teamtreehouse.com/install-node-js-npm-windows
- **Mac**: http://shapeshed.com/setting-up-nodejs-and-npm-on-mac-osx/

9.2. Deploying

Step 1, Extract in your server the ZIP file provided by Libelium containing the remote node configuration web application source files.

Step 2, Check the right owner/group and permissions of all the files extracted, usually using www-data group is default in Ubuntu environment.

Step 3, Check the permissions of all folders and files, usually using 0770 for directories and 0660 for files is default in Ubuntu environment.

Step 4, Configure the remote node configuration web application context in your server with the path where the source files were extracted.

Example configuration for a server running Ubuntu operating system and Apache web server:

Create a new configuration file my_server.conf in /etc/apache2/sites-available directory.

my_server.conf example file, replace paths to match your server deployment locations:

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName my_server.com
        ServerAlias my_server.com

        DirectoryIndex index.html index.php
        DocumentRoot /path/zip/extracted

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        <Directory /path/zip/extracted>
                Options Indexes FollowSymLinks
                AllowOverride All
                Require all granted
        </Directory>
</VirtualHost>
```

Step 5, Is mandatory to write the customer's API KEY in the server configuration file, a valid API KEY will be provided by your Libelium sales representative.

If the "apikey" parameter is not properly configured, the remote node configuration web application will not work.

The server configuration file is located in `data/app.ini`, inside the folder where the customer extracted the zip file in Step 1. The pattern of this file is:

```
[app]
server = "https://api.libelium.com/smartparking/config.php"
apikey = ""
```

server: URL to remote node configuration service hosted in Libelium's servers.

apikey: unique API KEY provided by Libelium to identify the customer, must be double quoted.

Step 6, Enable this new configuration site created and restart the Apache server, usually the command "a2ensite my_server" will do the task in Ubuntu environment.

A good date and time server configuration is recommended in order to guarantee the data integrity of the information sent to the Smart Parking nodes,using an NTP server to keep the clock system up to date is recommended.

9.3. Making the server accessible from anywhere

Compulsory for Sigfox and Actility platforms.

Optimal deployment includes a server name pointing to a public and static IP, using dynamic DNS could be done with services like no-ip which has a free package http://www.noip.com/. No-ip has also a client application responsible for updating any IP address changes in the background, more information in: http://www.noip.com/download

Finally, the URL to configure the Sigfox and LoRaWAN platforms to send the callback request to will be conformed with the server name and the paths:

https://my_server.com/path/zip/extracted/

9.4. Web form

The remote node configuration web application previously deployed in your server publishes a web form available in the URL (http://my_server.com/) established in the web server configuration file. The web form is used to manage the configuration values for every single Smart Parking node. Using the web form, the remote configuration web application will save a binary file storing the configuration values for the Smart Parking node identified with the serial ID:

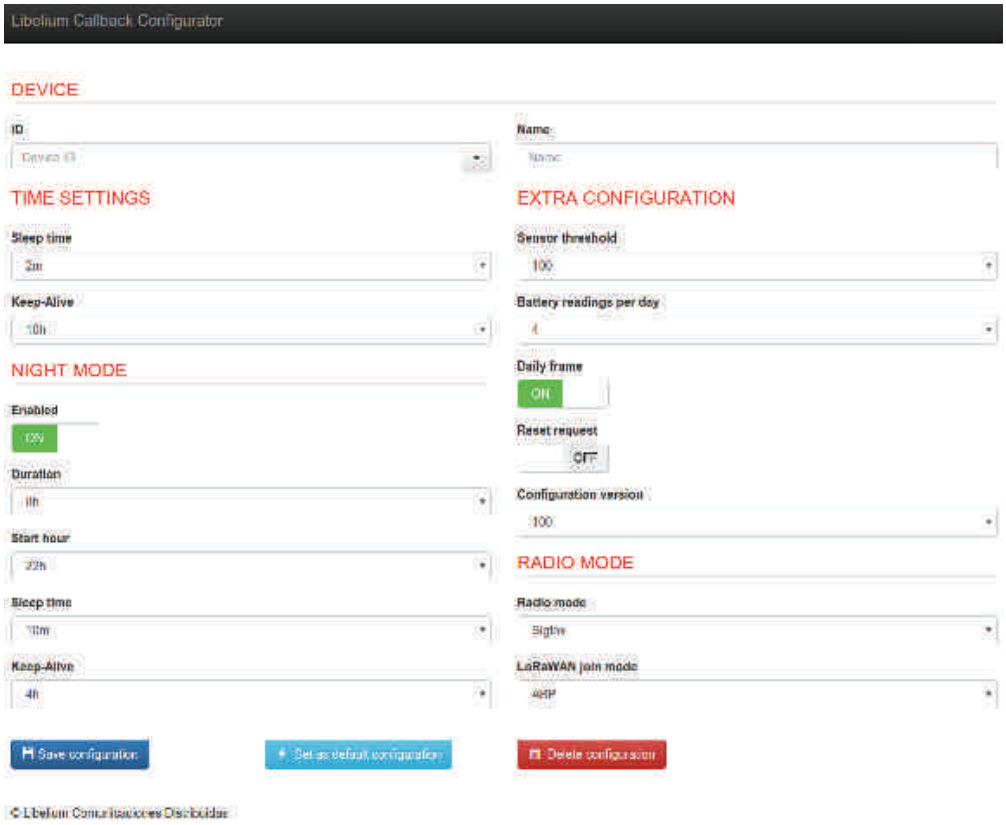


Figure : Libelium Callback Configurator screen-shot

Description of all the web form fields:

DEVICE

- **ID:** Device ID of the node. In each platform section we explain how to get this parameter. Note: Several ID may be written separated by semicolons ";" (Example: 00145F;001460;001461) to store identical configuration for many Smart Parking nodes at once.
- **Name**: A name associated to the device ID to make easier for you the identification of the node.

TIME SETTINGS

- **Sleep time**: Sleep time between consecutive sensor measurements.
- **Keep-Alive:** Elapsed time since last transmission to send a Keep-Alive frame.

NIGHT MODE

- **Enabled:** Button to activate/deactivate this option. If it is not active, the following fields will not be effective.
- **Duration:** Night mode duration time.
- **Start hour:** Night Mode start hour.
- **Sleep time:** Sleep time between consecutive sensor measurements (during night mode).
- **Keep-Alive:** Elapsed time since last transmission to send a Keep-Alive frame (during night mode).

EXTRA CONFIGURATION

- **Sensor Threshold:** Threshold for detecting a vehicle over the parking slot.
- **Battery readings per day:** Battery readings per day.
- **Daily frame:** Enable/Disable daily frame sending.
- **Reset request:** Enable/Disable the reset when a vehicle abandon the parking slot.
- **Configuration version:** Code version identifier.

RADIO MODE

- **Radio mode:** Radio transmission mode among Sigfox, LoRaWAN or their combinations.
- **LoRaWAN join mode:** LoRaWAN join mode, ABP or OTAA.

Each Smart Parking node has been assigned with two different radio ID, one unique ID for the Sigfox radio and a different ID for the LoRaWAN radio. Using the web form, a binary value with the configuration parameters will be stored internally for every radio ID. Users can create or update the information of every device.

Create new configuration: Type in the ID field of the DEVICE section the new ID value assigned to the Smart Parking node to configure. Configure all parameters properly.

Update existing configuration: Pick from the ID list the node ID to update (all parameters previously saved will be loaded in the form), or type it in the ID field of the DEVICE section.

Name fields with ones which do not exist, when the information is saved, it will be created. If the device ID exists, it will be updated like if you select the device from the list. Configure all parameters properly.

Click on “Save configuration”, a background procedure will start to save all information needed to generate a valid answer to the node.

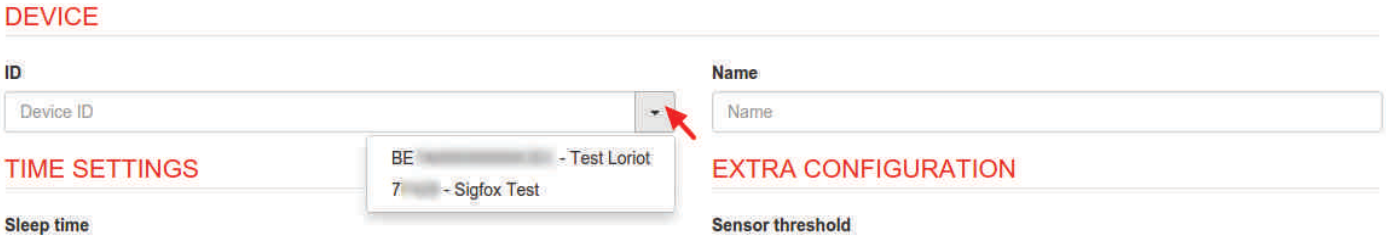


Figure : Device creation detail

Note: Sigfox and LoRaWAN radios unique ID must be known previously to use the web form. There are several ways those unique ID are provided. Additionally the Libelium Smart Devices App can be used to read those ID values directly from the Smart Parking node, the ID needed to create configuration values using the web formare pointed in the following picture:

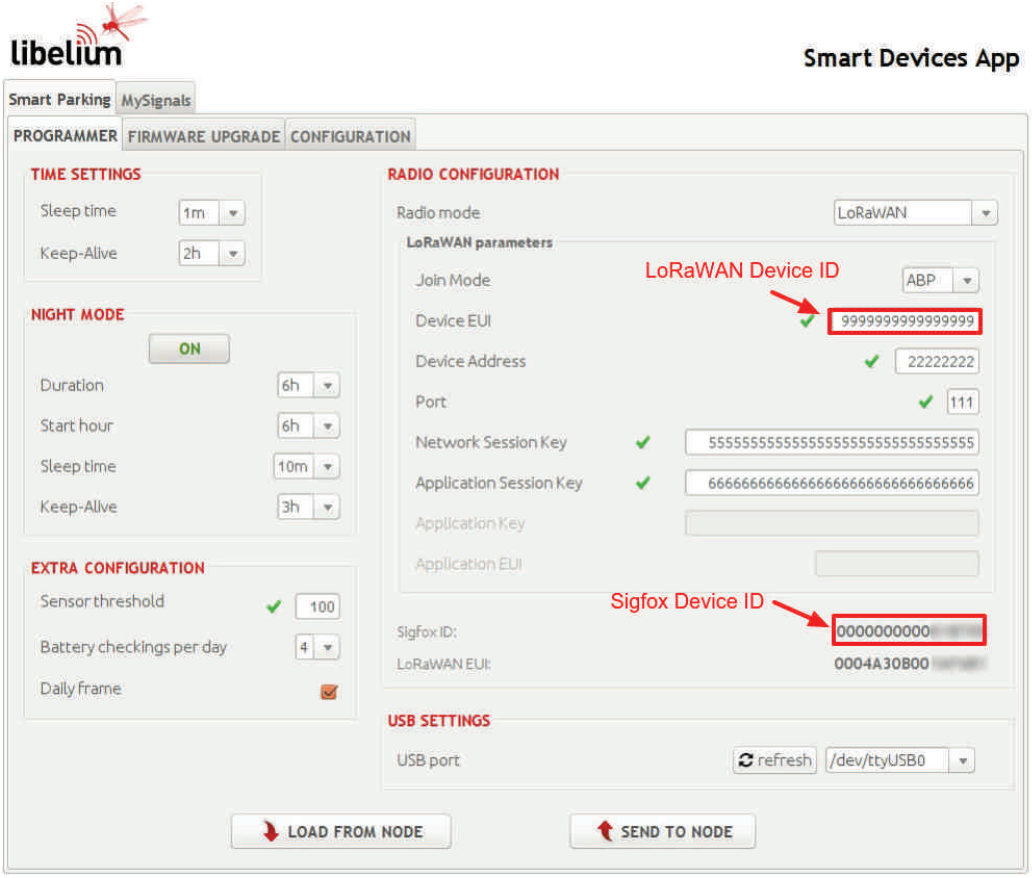


Figure : Device IDs read from the Smart Devices App

Creating configuration values for the radio ID of the Smart Parking nodes using the web form, enables the remote configuration web app to modify some parameters of the Smart Parking nodes as the Smart Java App does. Using the Smart Java App requires physical access to the Smart Parking node, the node must be attached to the computer using a USB cable. Using the remote node configuration web app will modify the Smart Parking node behavior remotely.

All Smart Parking nodes will synchronize date and time with the server (NTP recommended as commented previously in this chapter) in any case, no matter that a valid binary value has not been previously stored for the radio ID.

Remote node configuration web app manage three different request, to generate the proper answer to them when a message is received, a valid binary value is required to be configured for the radios of the Smart Parking node. Configuring all the Smart Parking nodes using the web form is recommended to get the most of the remote configuration web application.

Description of the 3 different request:

- Response to the Start Frame number 1, transmitted by the node in the activation process. This response frame allows the initial time synchronization of the internal clock. In normal conditions, the activation process happens only once.
- Response to the Daily Update Frame with a timestamp, in order to perform a time synchronization of the internal clock. This will happen once per day and helps to keep the node's time drift to minimum.
- Response to the Daily Update Frame with a new configuration for the node. This will happen very few times, a change in the Smart Parking node configuration will only happen when the administrator of the network decides to change the initial configuration parameters loaded via the USB Programmer.

9.5. How to extend the remote node configuration web application

Libelium provides to users a simple remote node configuration web application to modify remotely the behavior of the Smart Parking nodes. The remote node configuration web application includes a web form to manage the configuration values associated to the radio ID of the Smart Parking node, and a background process, to deal with the Sigfox and LoRaWAN Callback services requests delivering the proper replies with the configuration values previously stored using the web form.

The background process could be improved, or totally recoded using other programming languages, sharing with the web form the radio configurations.Some new features that could be implemented by users are: to save the node information in a database, grouping the nodes in zones, adding user credentials, etc.

For improvements on the web form, it is important to pay special attention to the endpoints described in this section to get the expected response containing the configuration message that will be sent to the nodes. Storage of the configuration values must be shared with the background process.

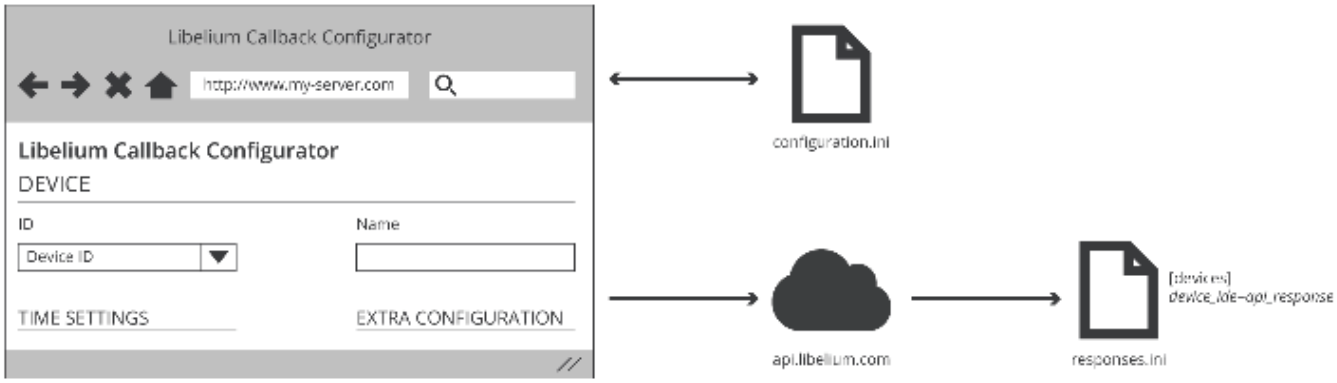


Figure : Backend working diagram

When a node configuration is saved using the web form, all the values of the fields are stored in the configuration. ini file. The web form will recover them if the device is selected again from the ID list.The web form also makes a call to the Libelium API with all the values of the parameters, the response is saved in the responses.ini file.

Endpoints description:

endpoint

https://api.libelium.com/smartparking/config.php

method

POST

header

key	`Authorization`
value	`"Bearer " + <APIKEY>`
The `<APIKEY>` provided in order to protect the system and control the access	

body

key	`NODE_TYPE`	
value	`"Parking"`	`Text`
Description	Kind of node to be used. Manual value `["Parking"]`	
App field	-	
App values	-	

key	BAT_READING	
value	[0-7]	Text
Description	Number of daily battery level checking	
App field	Extra configuration - Battery checking per day	
App values	[0 = 1 | 1 = 2 | 2 = 3 | 3 = 4 | 4 = 5 | 5 = 6 | 6 = 7 | 7 = 8]	

key	RADIO_MODE	
value	[0-4]	Text
Description	Selected transmission mode	
App field	Radio configuration - Radio mode	
App values	[0 = Sigfox | 1 = LoRaWAN | 2 = Sigfox + LoRaWAN | 3 = Sigfox → LoRaWAN | 4 = LoRaWAN → Sigfox]	

key	THRESHOLD	
value	[0-255]	Text
Description	Threshold for detecting a vehicle over the parking slot	
App field	Extra configuration - Sensor threshold	
App values	[0 = 1| 1 = 2 | 2 = 3 | 3 = 4 | 4 = 5 | … | 255 - 256]	

key	SLEEP_TIME	
value	[0-31]	Text
Description	Sleep time between consecutive sensor measurements	
App field	Time Settings - Sleep time	
App values	[0 = 1m | 1 = 2m | 2 = 3m | 3 = 4m | 4 = 5m | 5 = 10m | 6 = 15m | 7 = 20m | 8 = 30m | 9 = 1h | 10 = 2h | 11 = 3h | 12 = 4h | 13 = 5h | 14 = 6h | 15 = 7h | 16 = 8h | 17 = 9h | 18 = 10h | 19 = 11h | 20 = 12h | 21 = 13h | 22 = 14h | 23 = 15h | 24 = 16h | 25 = 17h | 26 = 18h | 27 = 19h | 28 = 20h | 29 = 21h | 30 = 22h | 31 = 24h] m = minutes h = hours	

key	KEEP_ALIVE	
value	[0-15]	Text
Description	Elapsed time since last transmission to send a Keep-Alive frame	
App field	Time Settings - Keep-Alive	
App values	[0 = 0h | 1 = 0.5h | 2 = 1h | 3 = 2h | 4 = 3h | 5 = 4h | 6 = 5h | 7 = 6h | 8 = 8h | 9 = 10h | 10 = 12h | 11 = 14h | 12 = 16h | 13 = 18h | 14 = 20h | 15 = 24h] h = hours	

key	NM_START	
value	[0-23]	Text
Description	Beginning hour of the night mode	
App field	Night mode – Start hour	
App values	[0 = 0h | 1 = 1h | 2 = 2h | 3 = 3h | 4 = 4h | 5 = 5h | 6 = 6h | 7 = 7h | 8 = 8h | 9 = 9h | 10 = 10h | 11 = 11h | 12 = 12h | 13 = 13h | 14 = 14h | 15 = 15h | 16 = 16h | 17 = 17h | 18 = 18h | 19 = 19h | 20 = 20h | 21 = 21h | 22 = 22h | 23 = 23h] h = hours	

key	NM_PERIOD	
value	[0-15]	Text
Description	Night mode duration time	
App field	Night mode - Duration	
App values	[0 = 0h | 1 = 1h | 2 = 2h | 3 = 3h | 4 = 4h | 5 = 5h | 6 = 6h | 7 = 7h | 8 = 8h | 9 = 9h | 10 = 10h | 11 = 11h | 12 = 12h | 13 = 13h | 14 = 14h | 15 = 15h] h = hours	

key	NM_SLEEP_TIME	
value	[0-15]	Text
Description	Sleep time between consecutive sensor measurements (during night mode)	
App field	Night mode – Sleep time	
App values	[0 = 2m | 1 = 5m | 2 = 10m | 3 = 15m | 4 = 20m | 5 = 30m | 6 = 1h | 7 = 2h | 8 = 3h | 9 = 4h | 10 = 5h | 11 = 6h | 12 = 8h | 13 = 10h | 14 = 12h | 15 = 14h] m = minutes h = hours	

key	NM_KEEP_ALIVE	
value	[0-15]	Text
Description	Elapsed time since last transmission to send a Keep-Alive frame (during night mode)	
App field	Night mode – Sleep time	
App values	[0 = 0h | 1 = 1h | 2 = 2h | 3 = 3h | 4 = 4h | 5 = 5h | 6 = 6h | 7 = 7h | 8 = 8h | 9 = 9h | 10 = 10h | 11 = 11h | 12 = 12h | 13 = 13h | 14 = 14h | 15 = 15h] h = hours	

key	CONFIG_ID	
value	[1-255]	Text
Description	Version number of the configuration	
App field	Extra configuration – Configuration version	
App values	[1-255]	

response

Users will receive a JSON string with these possibilities:

{ "error": Text }	Error received from the API system validation. text contains the error detail.
{ "status": [OK | NOK], "data": Text }	Information received after all values have been evaluated. NOK status includes in data the error detail. OK status includes in data the string to be saved in data/configuration.ini file, associated to the device ID provided.

10. Developing the network

10.1. Application considerations

10.1.1. Deployment of the motes

The optimum deployment point will be the one where the probability of detection is maximum, which means minimizing the probabilities of false detection (caused by other vehicles or objects near the lot under control) and false rejection (owed to a not high enough variation in the magnetic field above the mote with a vehicle parked in the spot).

This optimum deployment spot will depend on the kind of parking lot that we are going to monitor. In the case of parallel parking lots the mote should be deployed below one of the car sides, as shown in figure below, while for perpendicular parking spots the most adequate place will be the one nearest to the center of the motor or the backside of the vehicle.

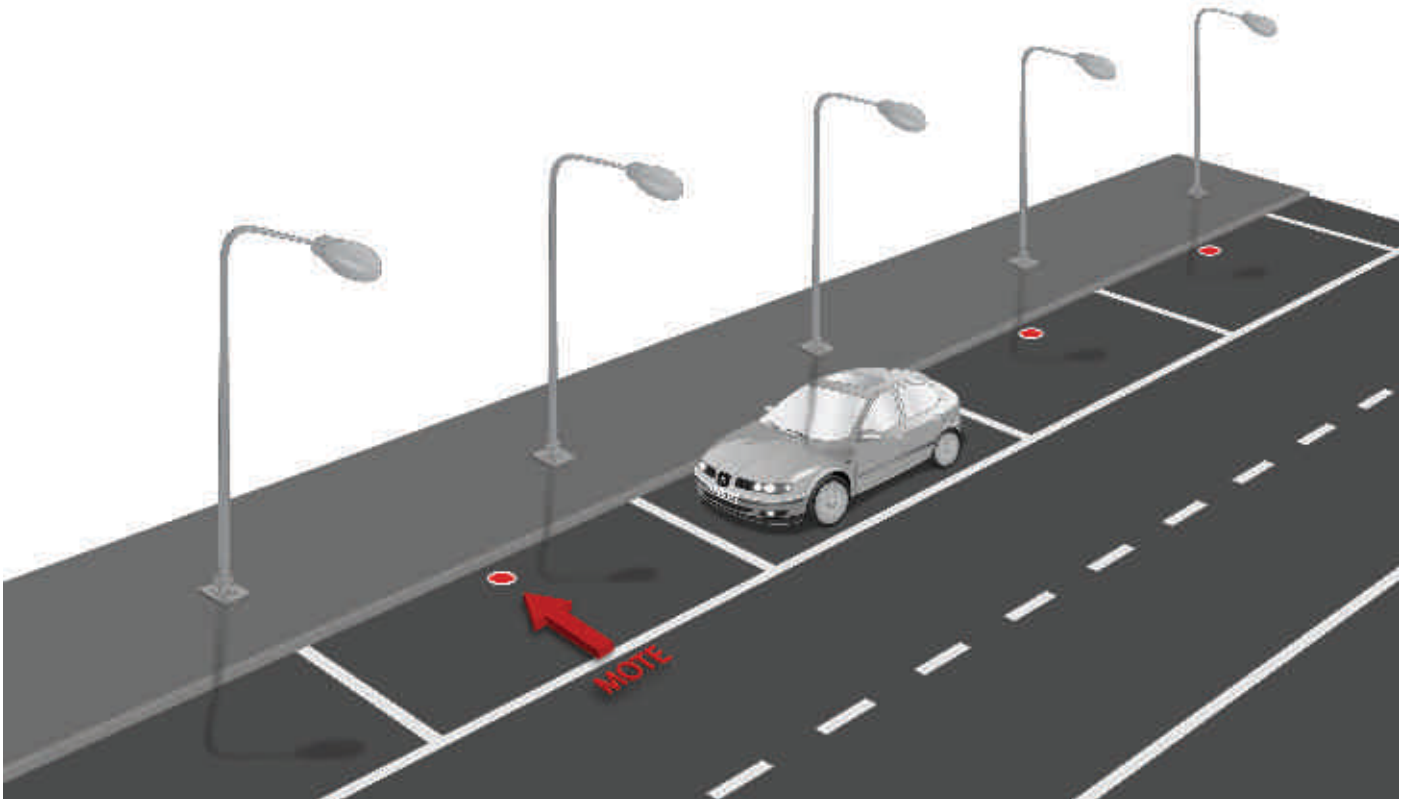


Figure : Diagram of the deployment points of the motes for parallel parking lots

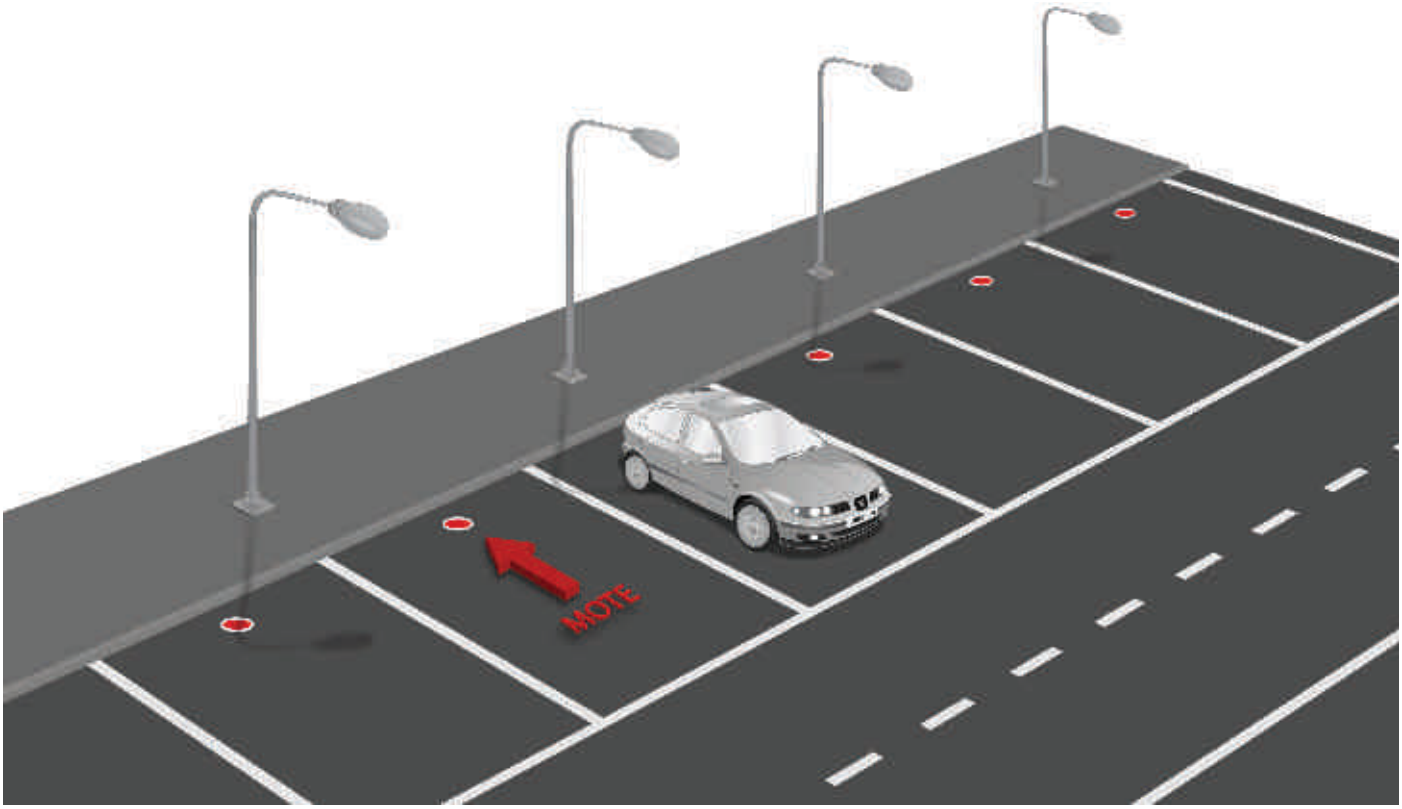


Figure : Diagram of the deployment points of the motes for perpendicular parking lots

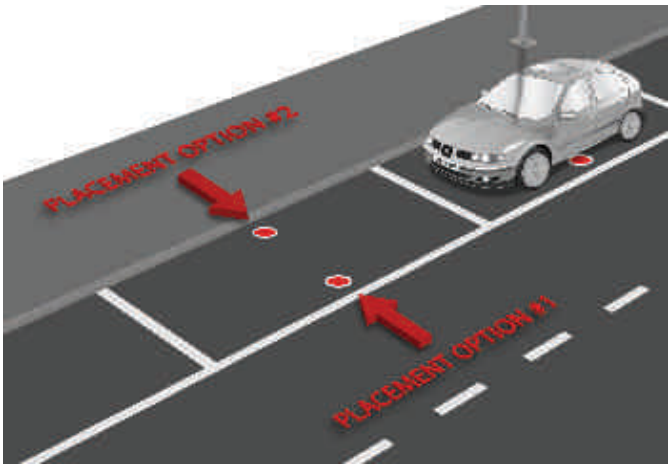


Figure : Placement options of the motes for parallel parking lots

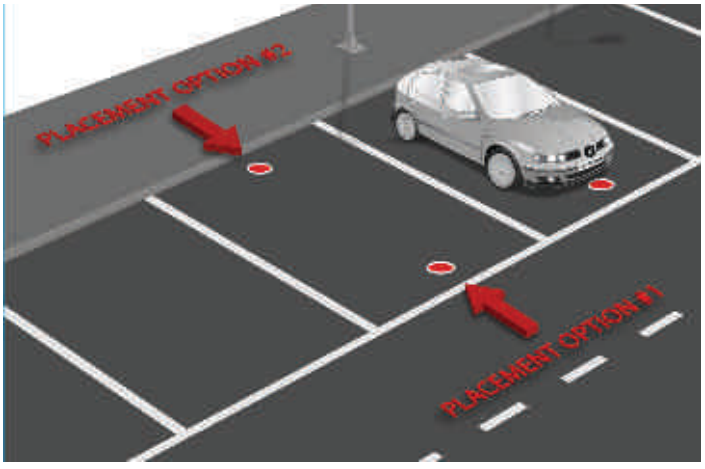


Figure : Placement options of the motes for perpendicular parking lots

Other consideration to be taken into account in mote deployment is the communication between this one and the gateway or router that will receive the data and process or redirect it. This is a very variable issue that will have to be analyzed independently for each scenario.

10.1.2. Interference of other vehicles

As pointed in the section “Deployment of the motes”, the presence of other vehicles in contiguous spots or near places may influence in the detection, modifying the detection threshold. This influence earns special importance in perpendicular lots, where the distance between the mote and the contiguous vehicles is shortest, and in the non-delimited parallel lots. The best way to avoid this disturbance is to take into account the state of the near spots in the detection decision when the variation in the magnetic field is very close to the detection threshold.

11. Device Installation

Important: Before deploying the nodes on the street, make sure that enough tests have been performed in order to achieve a 100% functional network and that all the necessary information related to the mote, such as identification numbers of the radios, has been compiled and stored, since once the mote is installed, the access will be very limited.

11.1. Assembly and set up

Step 1: Connect the USB Programmer to the node. Please note that an inadequate connection of the USB Programmer can damage the node.

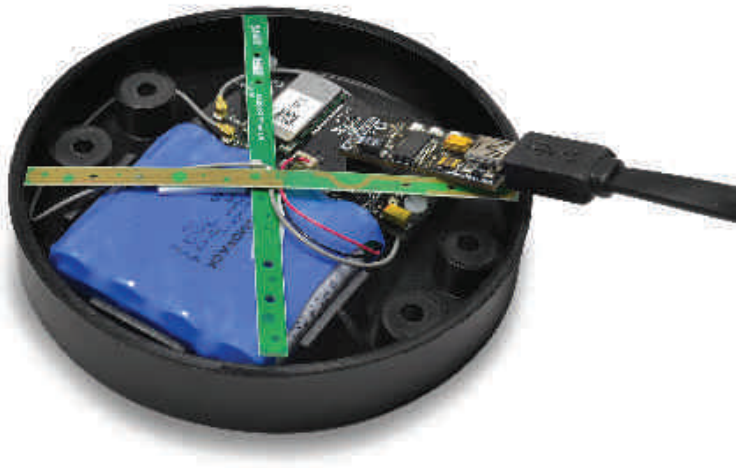


Figure : Plug & Sense! Smart Parking with the USB Programmer tool connected

Step 2: Once the parameters have been configured and the USB port selected in the Plug & Sense! Smart Parking Programmer, turn on the node and send the configuration with the button “Send to node”. If the node has been programmed successfully, the next message will be shown.

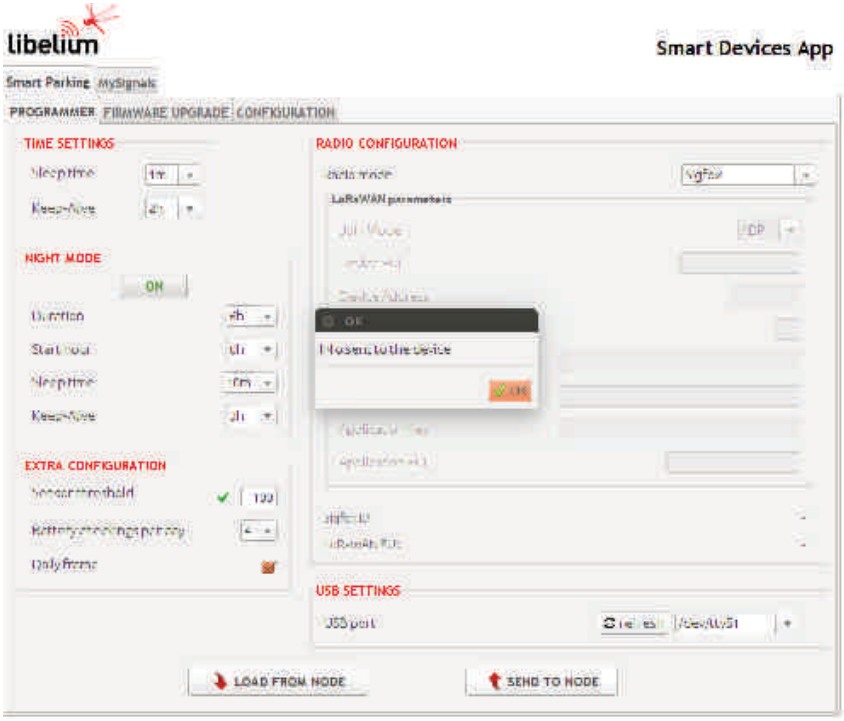


Figure : Configuration successfully programmed

Step 3: Now the node is configured. Turn off the node .

Figure : Battery connection

Step 4: Before closing the enclosure, the node must be powered on. It is mandatory that the node is powered off before this step. The node blinks the onboard red LED (1 second) and it will go to deepsleep state. In this state the node **must not be reset with the magnetic switch before the node is deployed**. If it is reseted, the node starts to work and will send frames, generating an unwanted consumption. Besides, the node would perform the calibration process before it is in the real location.

Figure : LEDs blinking

Step 5: Now, the node can be closed. It is ready to deploy.

11.2. How to close the Smart Parking enclosure to keep the waterproof IP68 features

In order to close the node correctly and ensure correct sealing, the following steps must be strictly followed.

Step 1: Make sure that the screws have the o-rings to prevent water ingress.

Figure : Screws with o-ring

Step 2: Ensure that the top surface of the gasket is clean and contains no foreign objects.

Step 3: Place the inner casing inside the outer casing and make sure that the two position marks match.

Figure : Enclosure position marks

Step 4: Insert the screws and tighten them halfway

Figure : Screws in their position

Step 5: Finally, tighten the four screws firmly. Do not use the maximum pressure (do not go all the way with the screws), because the o-rings could be ejected from the screws, and then the waterproof feature would NOT be valid. Besides, do not screw too hard and keep on screwing, because the screws could carve the female sockets, expanding their inner diameter; this would cancel the waterproof quality too.

11.3. Installation and boot

Step 1: Indicate the 4 holes in the asphalt. Select an area as flat and regular as possible, avoid irregularities in the terrain. You should place the enclosure in the final location and use the 4 holes as a reference to drill 4 little marks in the ground. You can also draw 4 dots with ink.

Figure : Indicating the holes for the node

Step 2: Drill the holes. Warning: The drill diameter must be 10 mm maximum to prevent the anchor from rotating freely in the hole. Try to drill as vertical as possible, to enable the best enclosure installation. The length of the hole must be about 42 mm. It is important to know that the screw will only penetrate to half of the length of the anchor, so there must be a distance of 12 mm or more between the ground surface and the top of the anchor. You can experience with slightly deeper holes because the screw does not need to be fully screwed inside the anchor: depths of 45-50 mm could work too.

Remove all the dust and little rocks inside the 4 holes, created during the drilling operation. This is important to allow the anchors go all the way down. You can clean each hole mechanically and then use a bottle of spray duster (high-pressure air bottle). Make sure that the real depth of the hole is enough.

Figure : Drilling the holes

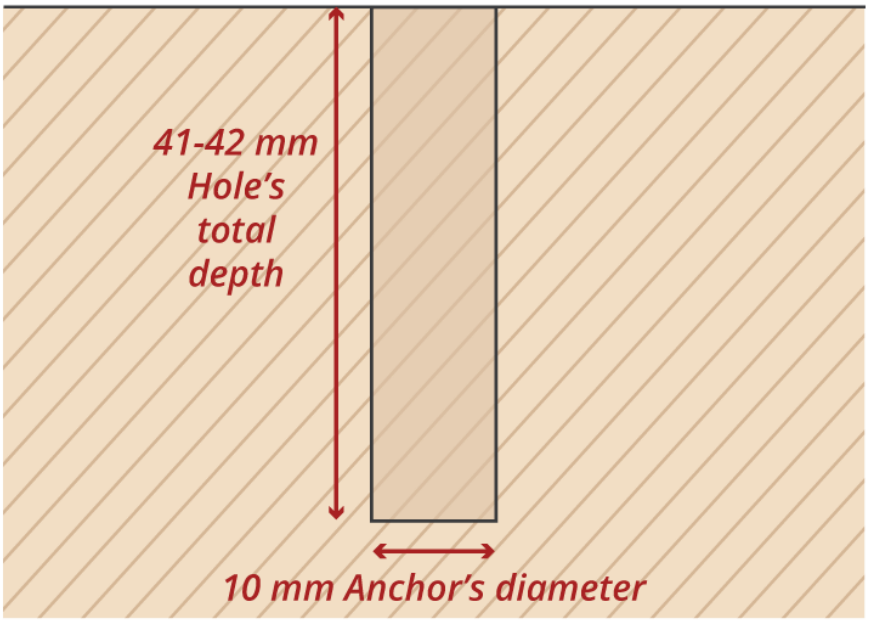


Figure : Section of the hole

Step 3: Setting anchors. The anchors used to fix the Plug & Sense! Smart Parking are metallic expansion anchors. The anchoring mechanism of these anchors is based on the expansion of the metal body against the base material. This expansion occurs when the expansion cylinder is propelled down, hitting on a punch with a hammer.

First, insert the anchors in the holes and take them all the way down. For that, you can use a hammer and punch of 9 mm in diameter. You can also use a sharper punch to hit on the top circle of the anchor, working on all the circle as you can see in the following diagram (take care of not hitting inside the anchor to avoid the cylinder goes down).

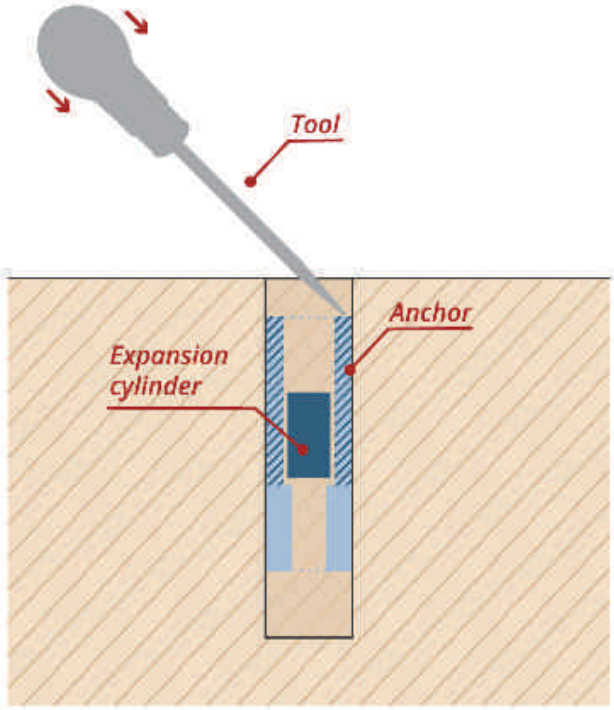


Figure : Hammering the anchor down

Figure : Good and bad anchor introduction

Once the anchor is totally inserted, make sure that it is at least 12 mm below the ground. Then take a sharp punch (few mm), put it inside the anchor and hit some very hard blows with a hammer.

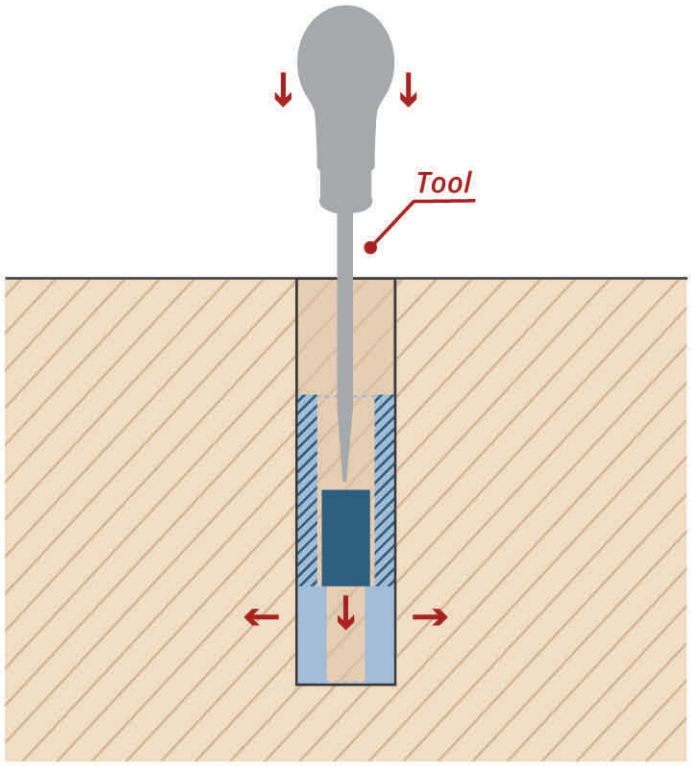


Figure : Hammering with a punch inside the anchor

You should notice that the cylinder went about 10 mm down. In that moment, the anchor installation is finished: it is fixed because its lower walls expanded.

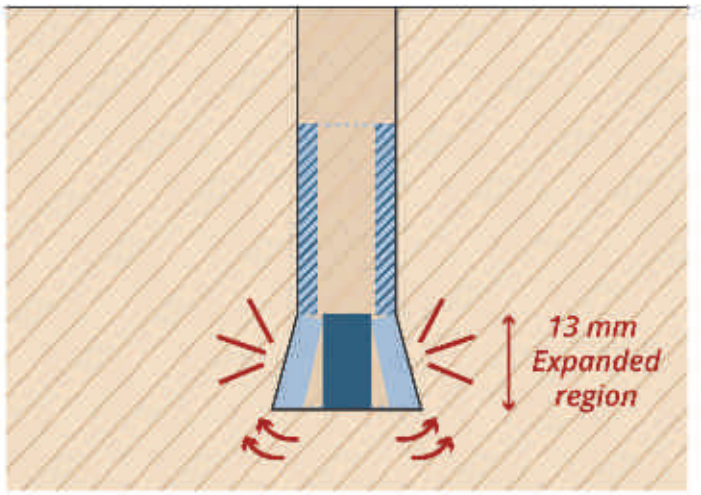


Figure : Anchor installed

Warning: *The anchors provided are indicated for concrete with at least 50 mm of depth, depending on the material of the terrain where the node is going to be installed, it could be that the anchors are not installed correctly. If the material on which the node is going to be anchored is different, it might be interesting to value an alternative type of fastening.*

Warning: *Make sure that the installation of the anchors is correct. If not, it is possible that the anchorage of the node is not correct and may even move from its position.*

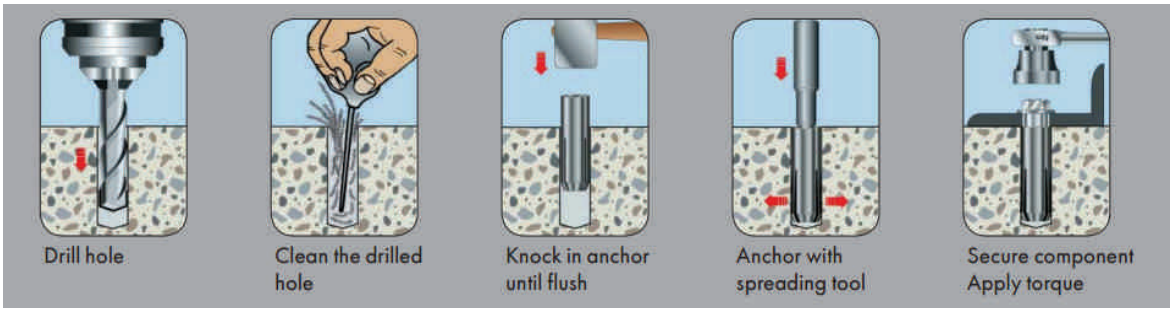


Figure : Setting anchors instructions

Step 4: Place the black circle rubber on the ground. To enable the correct screwing of the screws inside the anchors, make sure that there is no dust in the inner thread of the anchor (you can use the spray duster again). Install the node in its final position and screw the 4 special (anti-vandalic) screws provided. Anti-vandalic screws are recommended to avoid problems: anyone could unscrew a node if fixed with standard screws.

Figure : Screwing the node

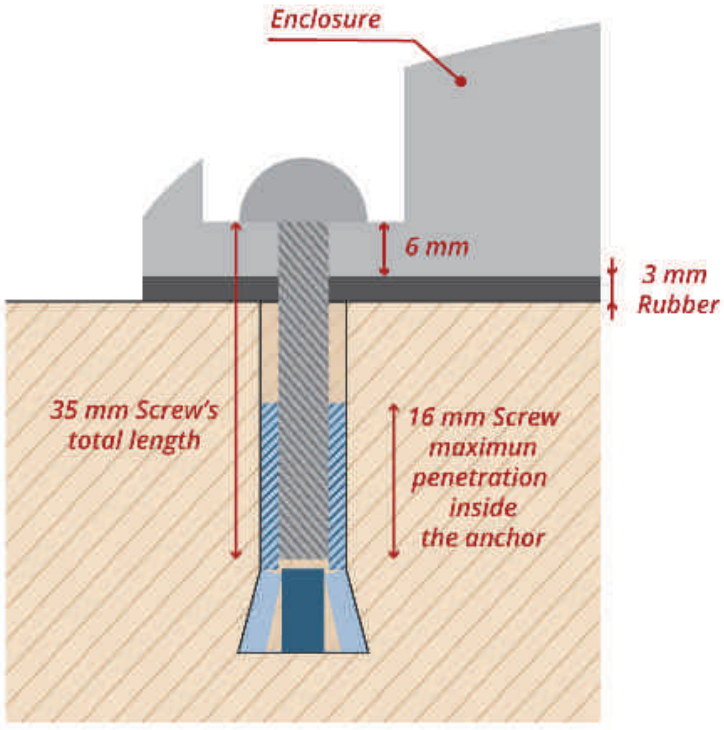


Figure : Final section of the anchor and screw

Figure : The node finally screwed

Step 5: Use the magnet to reset the node. Once the node has been reset, it will start to configure the radios and get a calibration of the parking slot. In this stage the parking slot **must be** empty, so the node learns when the slot is not occupied.

Figure : Using the magnet to reset the node

Step 6: When the node gets the reference calibration, it will send two frames to the cloud: Start frame 1 and Start frame 2.

Time	Delay (s)	Header	Data / Decoding	Location	Base station	RSSI (dBm)	SNR (dB)	Freq (MHz)	Rep	Callbacks
2016-03-21 12:54:42	1.4	0000	0501011706050203020 1ff00		07B8	-133.00	14.05	868.1116	3	
					07BC	-98.00	19.95	868.1115	3	
					0B3A	-130.00	15.36	868.1125	2	
2016-03-21 12:54:34	1.3	0010	090a0de30cf9000f Temp: 24.9 °C VDD idle: 3.338 V VDD tx: 3.299 V RSSI: -85.0		0B3A	-128.00	22.61	868.1242	1	
					07B8	-134.00	11.80	868.1232	1	
					07BC	-98.00	16.55	868.1232	1	
2016-03-21 12:53:57	< 1	0000 ack required	040000190003fe71fc930000		0B3A	-132.00	12.81	868.1409	3	
					08C9	-138.00	7.30	868.1348	1	
					07B8	-136.00	9.30	868.1280	3	

Figure : Start frames received in the Sigfox backend

11.4. Configuring the parking nodes in the callback server

Customers have installed the callback server, as explained before. Now, all nodes have to be configured using the on-line form available.

Considering that a real deployment will have groups of nodes working with different configurations (for example, street #1, street #2), it is recommended that the final Remote Manager is able to generate the configuration depending on the group that each node belong to. This can be achieved if the cloud keeps record of the groups, and which nodes are inside each group.

Please, see the Sigfox and LoRaWAN sections to get more information about specific callback configuration.

12. Services

Libelium has implemented 3 services in the remote node configuration web application: Sigfox, Loriot and Actility. In this section, is explained how to set them up with a simple configuration. If users want to use a service of other company not already implemented, we include in this section the instructions to develop that new service.

12.1. Sigfox

This section explains how to route the information received from the Sigfox platform to the callback server and generate the needed response.

12.1.1. Device configuration

Configure a new Group in the Sigfox back-end:

- 1. Display the parent group hitting on group top menu section.
- 2. It is not allowed to create two same groups in the same level. An alert is shown while the form is saved noticing the error. Skip to the Device Type creation step if the group already exists.
- 3. Create the new group clicking in “New” button.Select the parent group following the next pop-up windows. Finally, fill in the form with the information required, checking if the parent group written is the one chosen in the previous step.

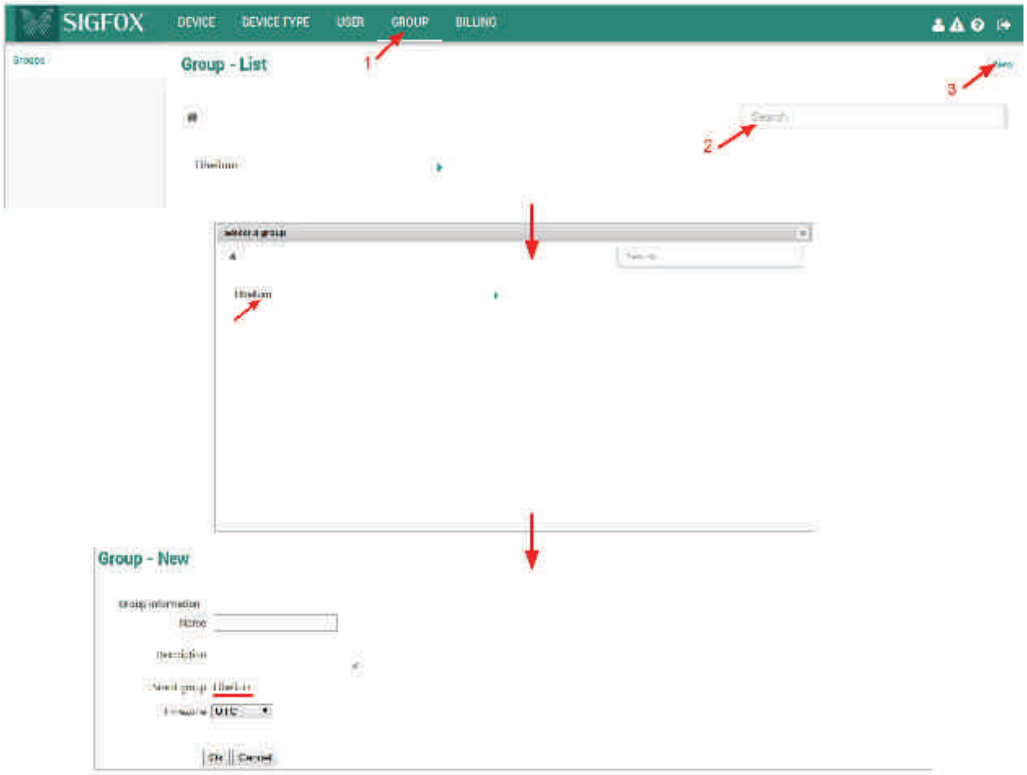


Figure : Group creation schema

The next step is creating the Device Type in the back-end, clicking on “Device Type” in the top menu and then on “New” button.

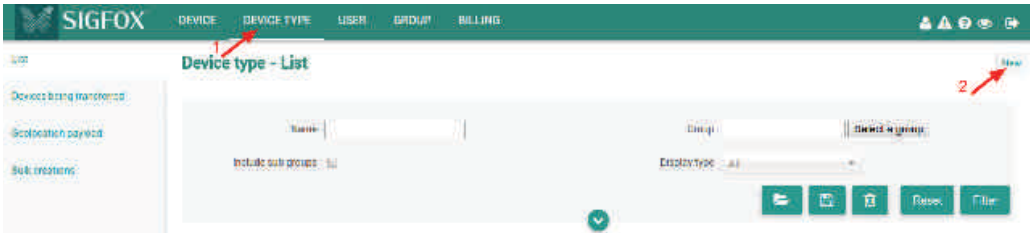


Figure : Device type creation schema

A new pop-up window will be shown to select the group for the new Device Type, usually the group created previously.

Select the group and a new form will appear, provide all information required to create the Device Type. Sigfox nodes assigned to this Device type will share the same functionality. It is mandatory to select CALLBACK as Downlink mode.

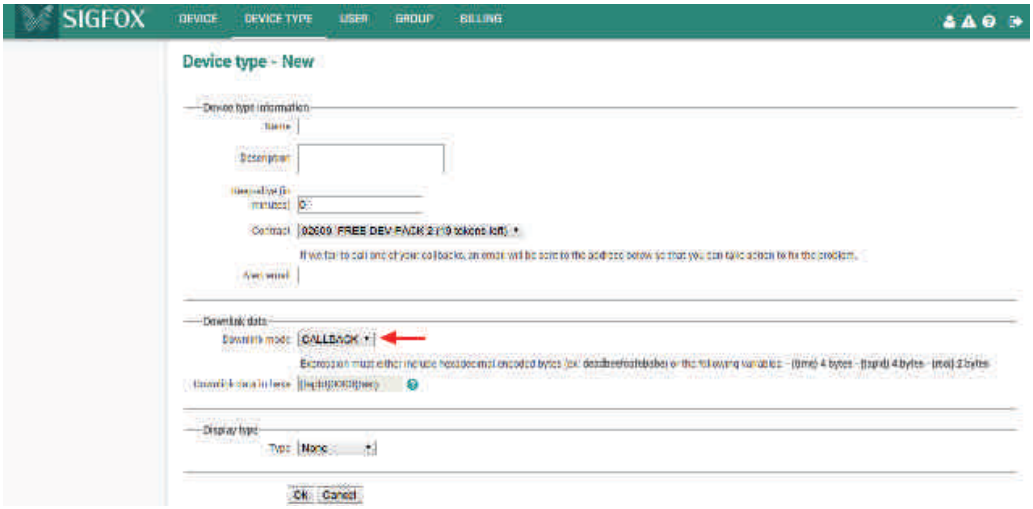


Figure : Device type form

The next step is configuring the callback. Click on “Callbacks” option in the left menu and then create a new callback: “New” button in the window loaded.

Figure : Callback creation

A new form will be displayed, please, fill in the form following the indications shown in the next image.

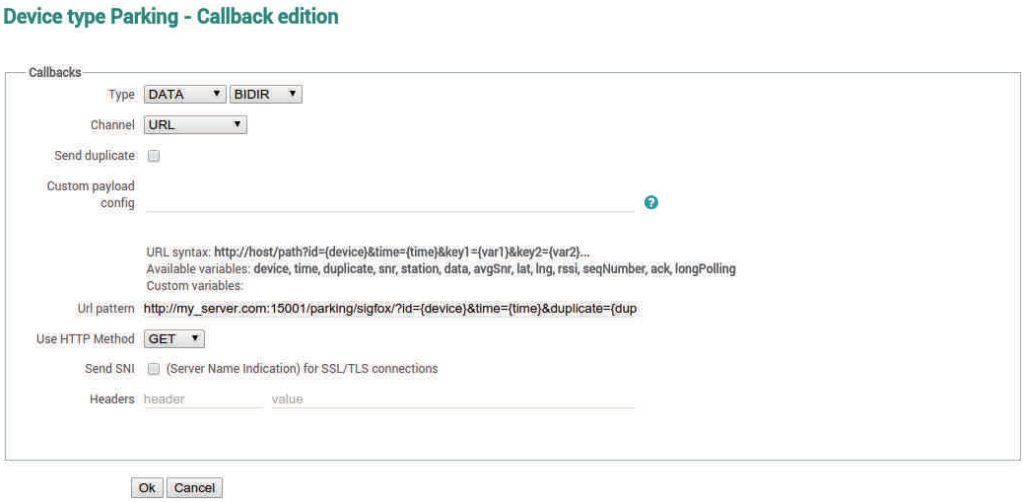


Figure : Callback form

Write the URL to access the Sigfox service in the field “Url pattern”. This URL has been previously configured in your server. Some extra variables have been added (to send to the service as much information as possible) following the information displayed in the form.

```
http://my_server.com/services/sigfox/?id={device}&time={time}&duplicate={duplicate}&snr={snr}&station={station}&data={data}&avgSnr={avgSnr}&lat={lat}&lng={lng}&rssi={rssi}&seqNumber={seqNumber}&ack=true
```

After the callback form is saved, the list of all available callbacks will be shown. Enable this entry with a downlink, the service running in your server will receive the information:

1. Activate “Downlink” column. There is a bullet in this column, and it must be activated. To activate, just click on it and leave the bullet coloured.

- ● Downlink active
- ○ Downlink inactive

2. “Enable” the callback. In the “Enable” column of the callback created, there is a check icon and you have to ensure that it is checked with a green colour.

- ☑ Callback enabled
- ☐ Callback disabled

Figure : Callback list

Finally it is time for creating the device node itself, click on “Device” option in the top menu, wait for the next window to be loaded, click on “New” button.

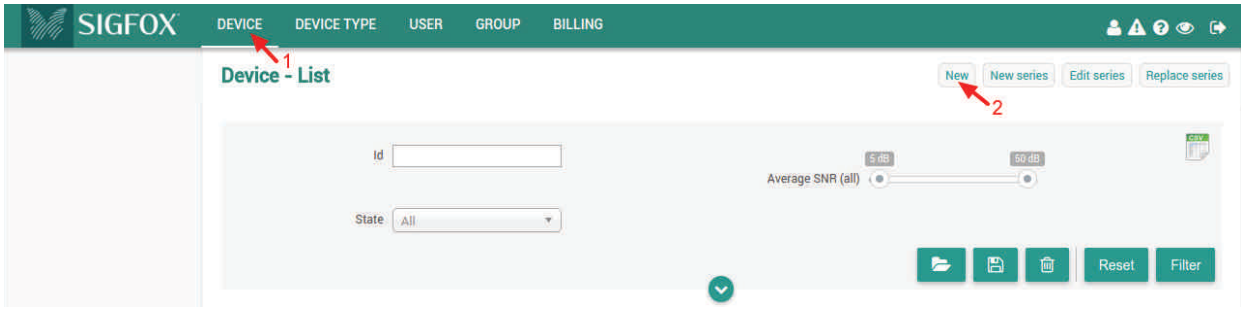


Figure : Device creation

As it happened with the Device Type, the parent group of the device must be selected. Fill the following information in the new device form:

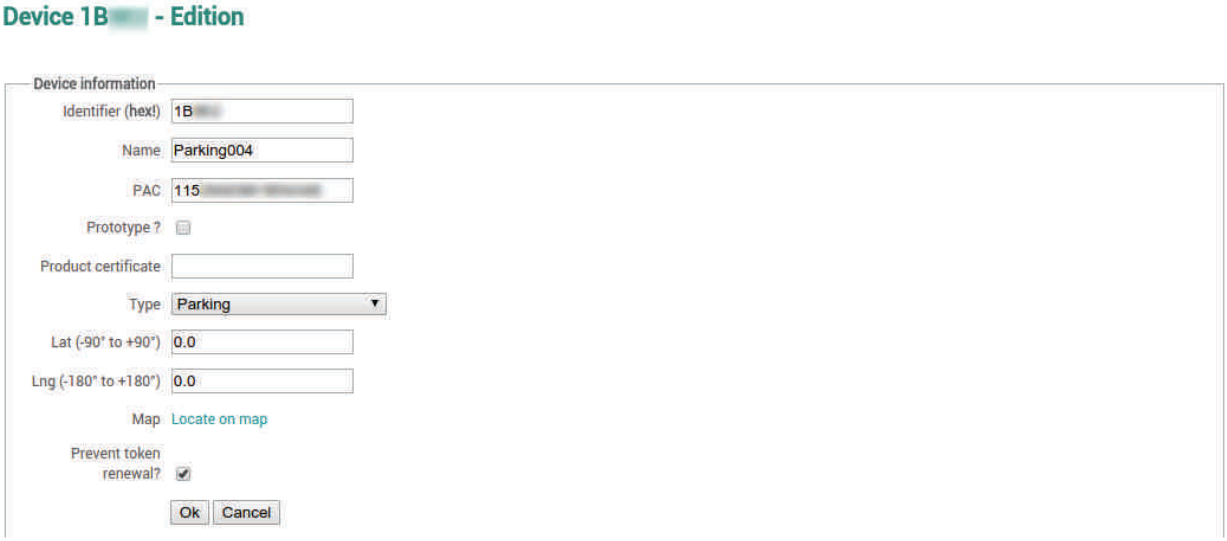


Figure : Device form

- **Identifier:** Device ID in hexadecimal format, given by the manufacturer.
- **Name:** Device name. A descriptive name useful to identify your device easily.
- **PAC:** Porting Authorization Code (PAC) is a unique hexadecimal number to identify the device regardless of the network. This code is given by the device manufacturer.
- **Prototype:** Mark this checkbox.
- **Product certificate:** Leave blank.
- **Type:** Select the device type created before.
- **Lat (-90º to +90º) / Lng (-180º to +180º) / Map:** Grab the coordinates of your node to use these three fields.
- **Prevent token renewal?:** Leave unchecked.

12.1.2. Server configuration

In the callback server previously installed, some parameters must be configured to enabe the communication with the node.

The services.ini file, located in data folder, has to be updated with the following information in the sigfox section.

```
[sigfox]
log_level = “ALL”
log_file = “../../logs/sigfox.log”
```

- **log_level:** This level is the minimum level to save logs in the system. Select among these levels:
  - OFF: This option deactivate the log.
  - ERROR : It only reports ERROR messages.
  - INFO: It reports ERROR + INFO messages.
  - DEBUG: it reports ERROR + INFO + DEBUG messages.
  - ALL: It reports everything happened in the process.
- **log_file:** The relative path where the log messages will be saved.

12.2. Loriot

This section explains how to route the information received from the Loriot platform to the callback server and generate a response if it is needed.

12.2.1. Device configuration

Log in and create a new application in the Loriot dashboard.

Figure : Creating an application

In this application all devices have to be added clicking on “Devices” left menu option and the device list will be displayed. Click on “Generate new device” button to add a new device. The new device will appear in the end of the list. Click on it to get more information.

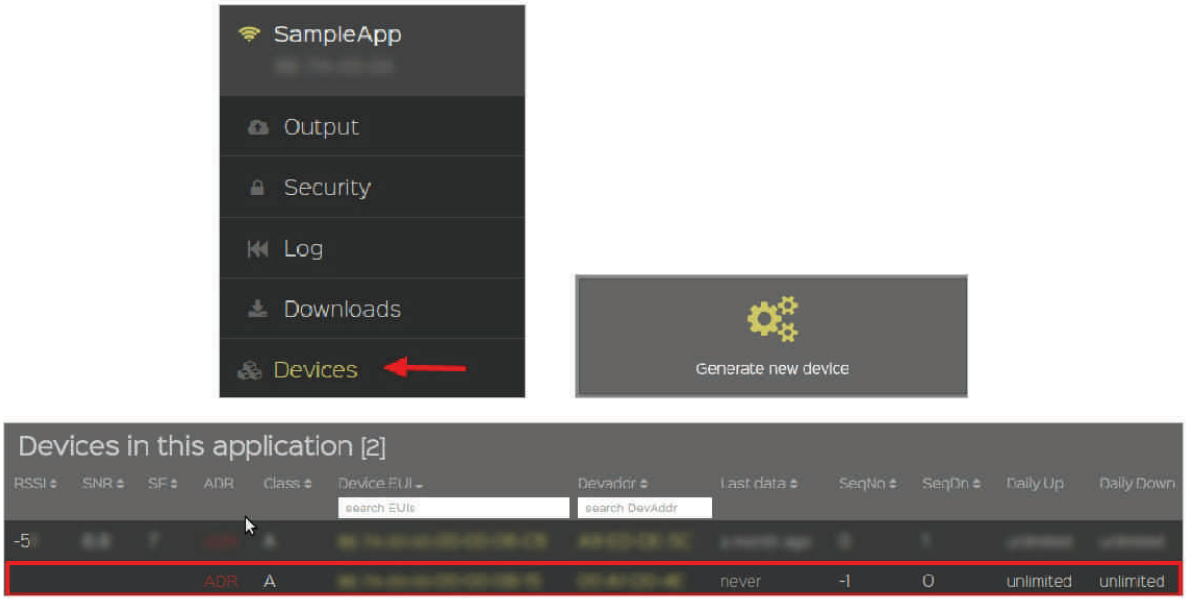


Figure : Device creation schema

Data output section is the responsible to stablish the communication between the platform and the callback server. Click on our application name in the left menu, the first right block “Network Application” contains the “Data output” option,the configuration is in the main window of the application created.

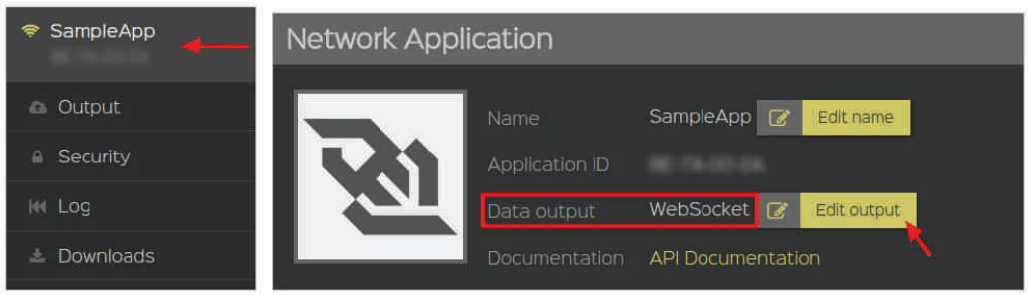


Figure : Output application selection

Click on “Edit output” to display all the information available about the selected output option.Click on “Change” button in the detailed window to change the data output, select one of the multiple choices from the list.

Figure : Selecting the output application

“Websocket” is the method recommended to deliver the end-device data.

- It’s cloud-friendly (existing HTTP traffic optimization can be used)
- It’s bi-directional, real-time interface
- It’s easy to implement
- It’s lower overhead compared to REST
- It’s already supported in all major web browsers
- It’s already supported in many programming languages

12.2.2. Server configuration

In the callback server previously installed, some parameters must be configured to enable the communication with the node.

The `services.ini` file, located in `data` folder, has to be updated with the following information in the `loriot` section.

```
[loriot]
log_level = “ALL”
log_file = “../../logs/loriot.log”
websocket_url = “wss://eu1.loriot.io/app?id=xxxxxxx&token=xxx-xxxx”
service_url = “http://my-server.com/services/loriot/”
```

- **log_level:** This level is the minimum level to save logs in the system. Select among these levels:
  - OFF: This option deactivate the log.
  - ERROR : It only reports ERROR messages.
  - INFO: It reports ERROR + INFO messages.
  - DEBUG: it reports ERROR + INFO + DEBUG messages.
  - ALL: It reports everything happened in the process.
- **log_file:** The relative path where the log messages will be saved.
- **websocket_url:** The URL of the websocket listed in the Loriot dashboard. In the last window shown “Application output”, in the second section “Current output setup” there are the target URL and the Authentication Token to access to the API interface.

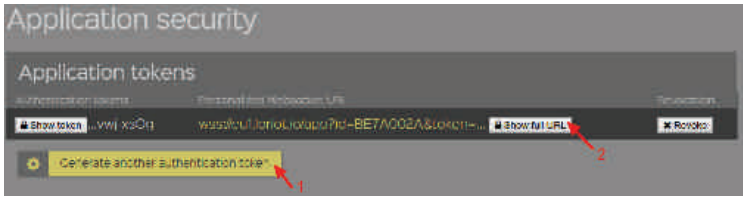


Figure : Output setup

Clicking in “Authentication token” link, a new window will be shown with the tokens generated in the Loriot application.

To create a new one, just click on “Generate another authentication token” button, and a new record will be created in the table. Finally, copy the full URL hitting on “Show full URL” and paste the URL in this parameter of the service configuration file.

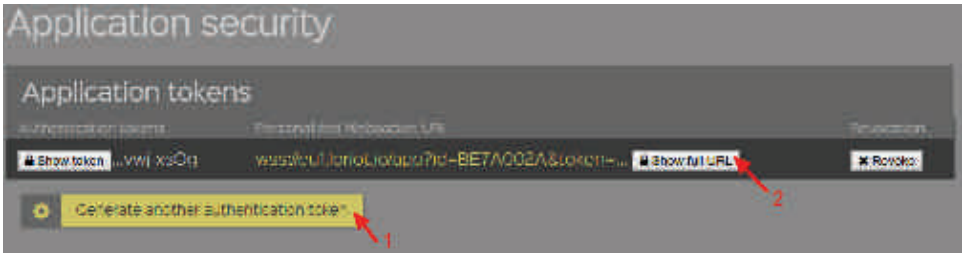


Figure : Generating an application token

- **service_url:** The URL to access to the Loriot service in your callback server previously configured. (http://my_server.com/services/loriot/)

12.3. Start the web-socket connection

This platform requires an additional step. Start the process which establish the connection to the Loriot server in order to receive and send information in all nodes.

Open a SSH connection to the web server in a terminal and navigate to the directory `your_web_server_path/services/loriot`. Execute following command to create the web-socket communication:

```
nohup nodejs websocket.js >>../../logs/loriot.log 2>&1 &
```

Kill this process to stop the web-socket communication.

12.4. Actility

This section explains how to route the information received from the Actility platform to the callback server and generate a response if it is needed.

12.4.1. Device configuration

Configure a new AS routing profile in the Device Manager clicking on “AS routing profiles” on the left sidebar menu.

Figure : AS Routing profiles menu

A list with all existing AS routing profiles will be displayed. Below, in the second section, New AS routing profile gives the capability to add new AS routing profiles. Clicking on Add button to create a new AS Routing profile.

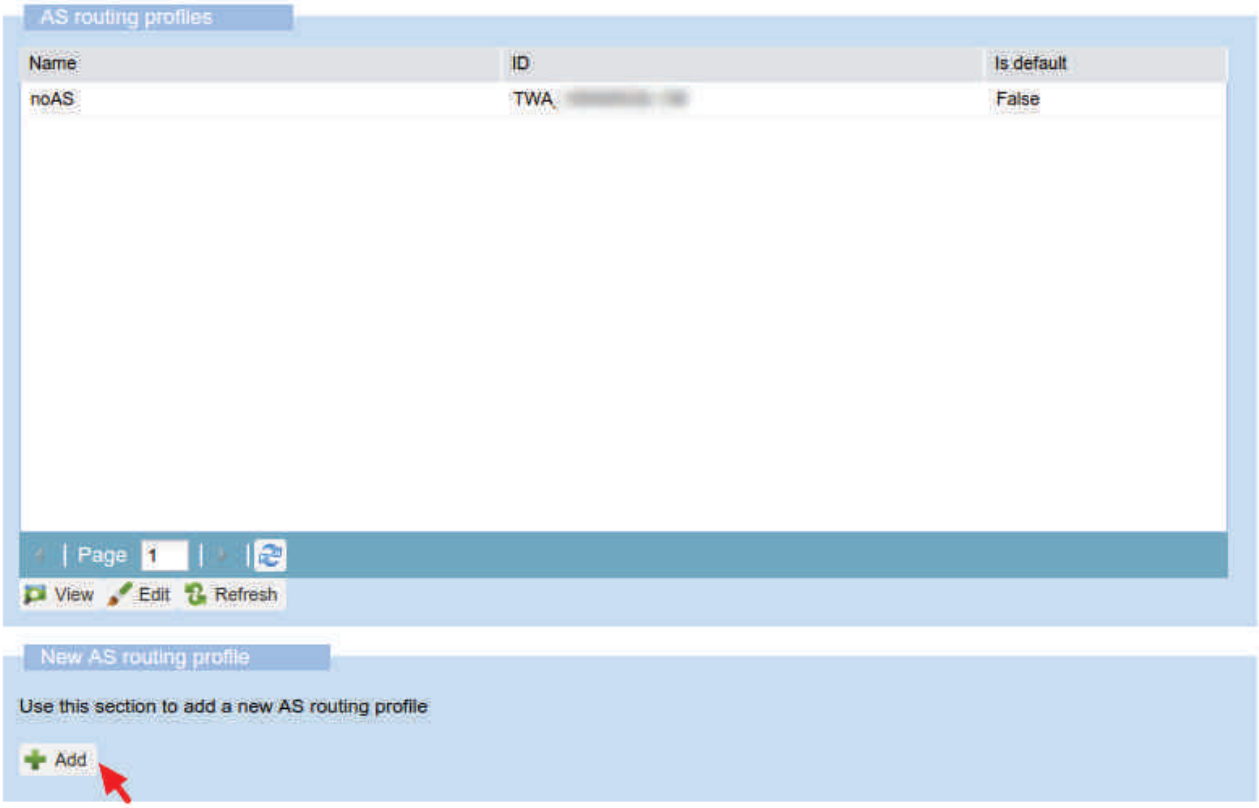


Figure : AS Routing profiles

A name must be typed in the new window displayed. Clicking on Create button to continue the process.

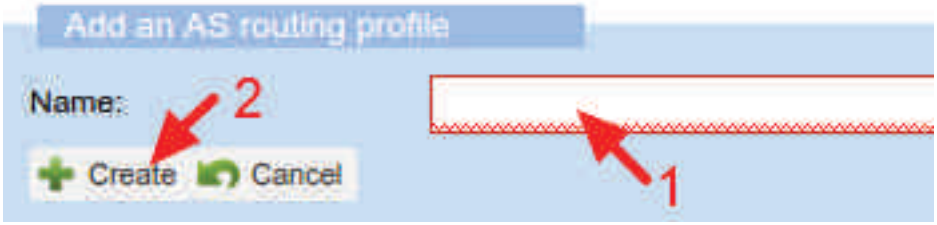


Figure : AS Routing profile name

In the new window, mark the check “Is default” and click the “Add button” in the Add a route section.

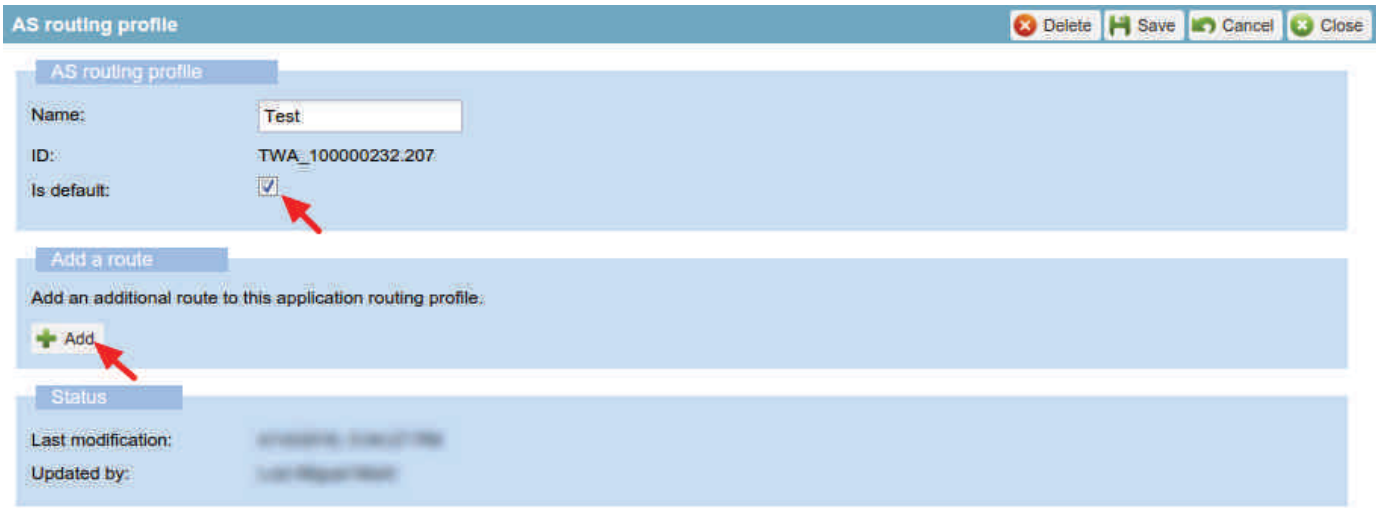


Figure : AS Routing profile route

A new section “Route” will appear with a default route created. The name will be always “bigONG1”. Only our destination URL must be in this table. The first step will be to delete the default route, clicking on the row and then on the “Delete button”

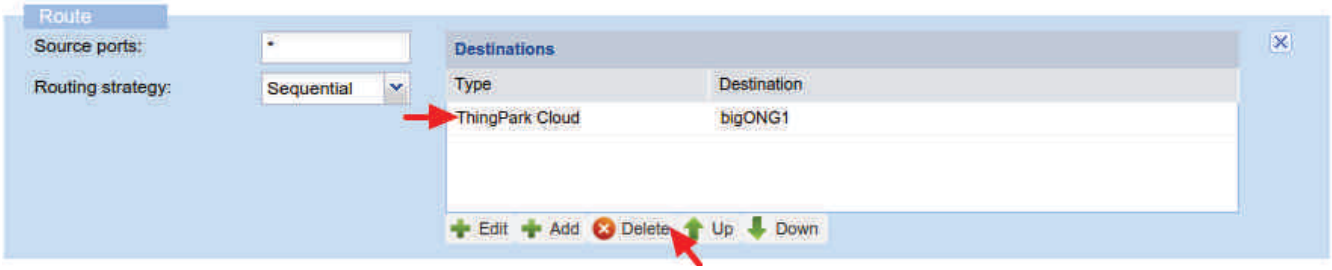


Figure : Add a route

After the default route has been deleted, create the destination URL.

- 1. Hitting on “Add” button a new pop up window will be displayed.
- 2. In “Type” field “Third party AS (HTTP)” must be selected.
- 3. In “Destination” field, the URL to access to the Actility service in your callback server previously configured has to be typed. (http://my_server.com/services/actility/)
- 4. Finally, hitting on “Add” button, the configuration will be saved.

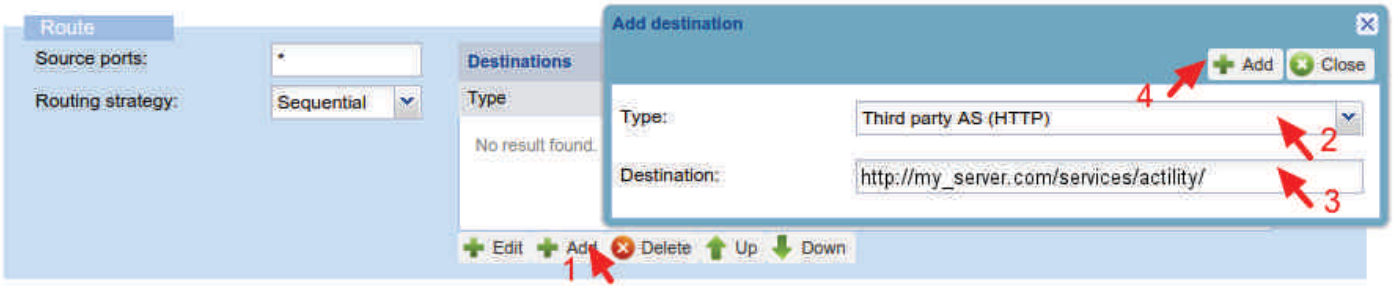


Figure : Route destination

The configuration will appear in the list. The last step is creating your device and associate it to the AS routing created. Right-click on “Devices” option in the left menu, and then hit on “Create device”.

Fill in all compulsory fields in the form displayed in the new window, according to the values in the Libelium Smart Devices App. Remember to select your AS routing in “AS routing profile”

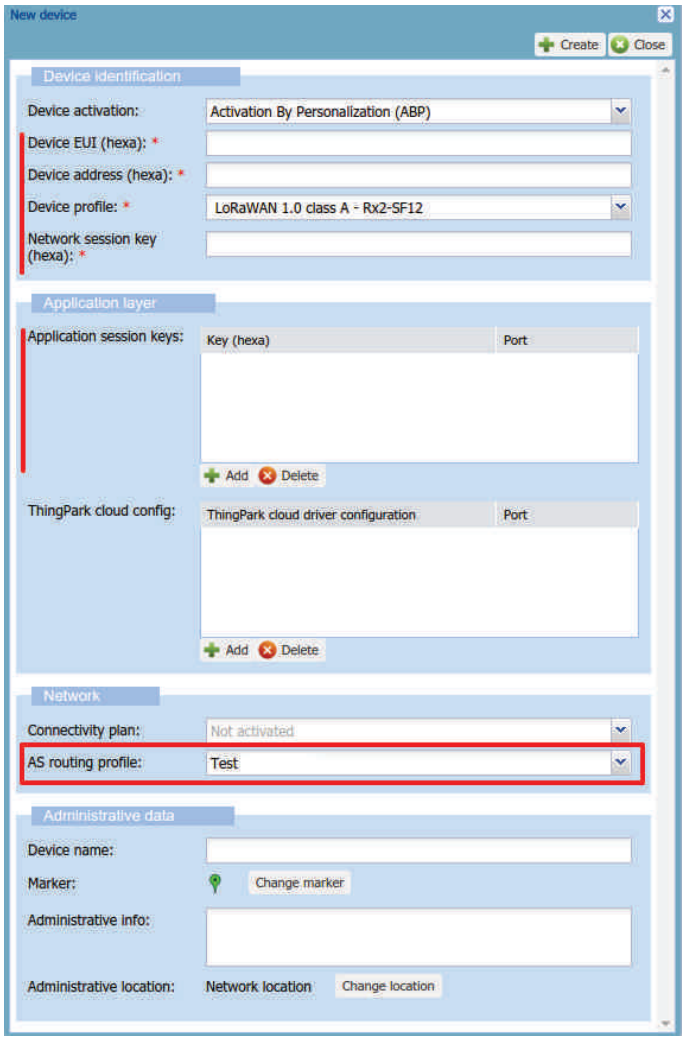


Figure : Create a new device

To change the network routing to the AS Routing profile select a device from the list and hit on “Edit” button. Then click on Network section in the left menu.

Figure : Steps to follow when the device is created

In the “Network/cloud routing” section, the new AS Routing profile created must be selected clicking on the “Change” button, select the profile created from the dropdown box displayed in the new pop-up window. Finally, click on the “Save” button.

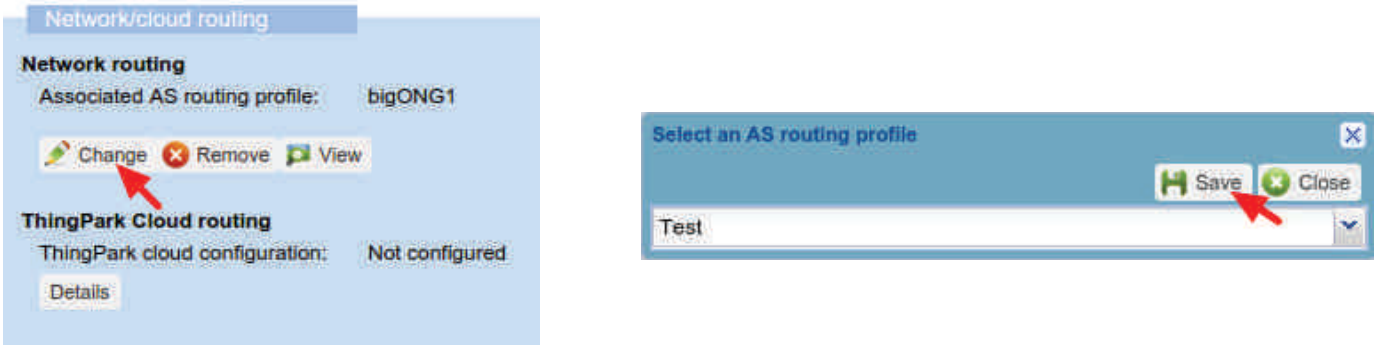


Figure : Select the AS Routing created

12.4.2. Server configuration

In the callback server previously installed, some parameters must be configured to enable the communication with the node.

The services.ini file, located in data folder, has to be updated with the following information in the actility section.

```
[actility]
log_level = “ALL”
log_file = “../../logs/actility.log”
server_url = “http://lrc99.thingpark.com:8807/sensor/”
```

- **log_level:** This level is the minimum level to save logs in the system. Users can select one among these levels:
  - OFF: This option deactivate the log.
  - ERROR : It only reports ERROR messages.
  - INFO: It reports ERROR + INFO messages.
  - DEBUG: it reports ERROR + INFO + DEBUG messages.
  - ALL: It reports everything happened in the process.
- **log_file:** The relative path where the log messages will be saved.
- **server_url:** The downlink URL that Actility gives to send information from the server to the node. This URL will be the address of the primary Actility LRC cluster.

12.5. Saving the information received

This section explains how to get the information sent by the Smart Parking nodes to the callback server.

The services implemented by Libelium log the information in a file by default. To save the informationin a database or a cloud service following files in charge of doing the operations described in the callback server need to be modified:

- **Sigfox:** *your_callback_server_path/services/sigfox/index.php*
- **Loriot:** *your_callback_server_path/services/loriot/websocket_response.php*
- **Actility:** *your_callback_server_path/services/actility/index.php*

your_callback_server_path has to be replaced with the path where the callback server was previously installed. Search for a comment line like this in the previous source code files:

```
//To-Do: get the information received: $frame->get_data();
```

Example to save the information in a MySQL database using the PHP PDO library:

```
$data = $frame->get_data();
$link = new PDO('mysql:host=server;port=port;dbname=dbname;charset=utf8', 'user', 'pas-
sword');
$statement = $link->prepare("INSERT INTO table(field1, field2) VALUES(:data1, :data2)");
$statement->execute(array(
   "data1" => $data['parking'],
   "data2" => $data['battery']
));
```

It is recommended to paste this code block below the commented line.

Take a close look at the next section in order to get more information about all the data obtained using the `get_data` method.

12.6. How to develop a new service

It is possible to develop a new service different to the services provided by Libelium. The aim of a service is to get the payload sent from the Smart Parking nodes and generate the correct answer when it is needed. Both, the payload and the answer, have to be decoded according to Libelium specifications.

Each service can have its own functionality: some will require a websocket, some a URL to call, ... The specifications of the new service should include all the information needed to implement the new service. We recommend to use PHP, so the new project could be developed faster reusing code created. The most useful PHP class is called frame, which has been created to simplify the payload-answer process and it is located in:
your_callback_server_path/includes/class.Frame.php

"your_callback_server_path" has to be replaced with the path where the user has the callback server installed.

This is a complete description of all attributes and methods of the class frame:

Attributes

Name	Description
`private String data`	Information received from the node.
`private String type`	Frame type decoded from the information sent from the node.
`private String return`	String to return to the node.
`private String device_id`	Device ID of the node.
`private Array bytes`	Information received from the node converted into bytes.
`private Array bits`	Array in which each previous byte is converted into bits.
`private Array info`	Array with the information decoded.

Methods

Name	Description
`void __construct (String device_ID)`	Class constructor. Device ID has to be sent as a parameter. It will be saved in the attribute `device_id`.
`void reset (void)`	Reset all attributes.
`void get_info (String data)`	Decode the information received from the node and store it in the attribute info. The information received from the node has to be sent as a parameter. The information decoded will be saved in the attribute `info`.
`Boolean waiting_response(void)`	It returns a boolean to know if it necessary to send a response for the frame received.
`String get_response(void)`	It returns a string to send to the node.
`Array get_data(void)`	It returns an array with all information decoded.

Using the `get_data` method of this class, users will get an array with the decoded information received from the node. Data will depend on the frame type received. These five parameters are common in all frame types:

Name	Description
`device_id`	ID of the device
`battery`	Battery level. `[0 = good charge level | 1 = little charge level]`
`parking`	Status of the parking slot `[0 = empty | 1 = occupied]`
`frame_counter`	Counter of the frames received.
`frame_type`	Counter of the frames received. `[0 = Info frame | 1 = Keep-Alive frame | 2 = Daily update frame | 3 = Error frame | 4 = Start frame 1 | 5 = Start frame 2]`

And then, these are the parameters for each frame type:

Frame type 0 = Info frame

Name	Description
`temperature_msb`	Raw temperature from the parking internal sensor. To convert to Celsius degrees use the next formula: $Temperature(^{\circ}C) = \frac{(MSB \cdot 2^8 + LSB)}{8} + 25$ Only available in firmware v1.x.x
`temperature_lsb`	
`temperature`	Temperature (Celsius degrees) from the node's internal sensor. The value of temperature is a signed integer. Only available from firmware v2.x.x
`x_msb`	Reference value from the sensor associated to the X axis.
`x_lsb`	
`y_msb`	Reference value from the sensor associated to the Y axis.
`y_lsb`	
`z_msb`	Reference value from the sensor associated to the Z axis.
`z_lsb`	

Frame type 1 = Keep-Alive frame

Name	Description
sensor_msb	Times that the sensor is used in the las 24 hours.
sensor_lsb	
sigfox_msb	Times that the Sigfox radio is used in the last 24 hours.
sigfox_lsb	
lorawan_msb	Times that the LoRaWAN radio is used in the last 24 hours.
lorawan_lsb	
resets_today	Number of resets generated in the last 24 hours.
CONFIG_ID	Value of the configuration version loaded into the node.

This class will generate automatically the response for this frame type which is formed by 8 bytes with the following structure:

Byte	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Bytes reserved for the information saved in the *configuration.ini* file of the specific device							
1								
2								
3								
4								
5								
6				Timestamp (hh) º hour with two numbers				
7	Enable daily frame	Reset	Timestamp (mm) º minutes with two numbers					

Frame type 3 = Error frame

Name	Description
error_z	Error detected in the X axis of the sensor. ["1" = error | "" = no error]
error_y	Error detected in the X axis of the sensor. ["1" = error | "" = no error]
error_x	Error detected in the X axis of the sensor. ["1" = error | "" = no error]
error_rtc	Error detected in the RTC (internal clock). ["1" = error | "" = no error]
error_lorawan	Error detected in the LoRaWAN radio. ["1" = error | "" = no error]
error_sigfox	Error detected in the Sigfox radio. ["1" = error | "" = no error]
temperature_msb	Raw temperature from the parking internal sensor. To convert to Celsius degrees use the next formula: $Temperature(ºC) = \frac{(MSB \cdot 2^8 + LSB)}{8} + 25$ Only available in firmware v1.x.x
temperature_lsb	
temperature	Temperature (Celsius degrees) from the node's internal sensor. The value of temperature is a signed integer. Only available from firmware v2.x.x

x_msb x_lsb	Reference value from the sensor associated to the X axis.
y_msb y_lsb	Reference value from the sensor associated to the Y axis.
z_msb z_lsb	Reference value from the sensor associated to the Z axis.
battery_level	Battery voltage in millivolts. To convert to millivolts, use the next formula: $Battery\ voltage(mV)=((Battery\ level)\cdot 4)+2800$

Frame type 4 = Start frame number 1

Name	Description
temperature_msb	Raw temperature from the parking internal sensor. To convert to Celsius degrees use the next formula: $Temperature(^{\circ}C)=\frac{(MSB\cdot 2^8+LSB)}{8}+25$ Only available in firmware v1.x.x
temperature_lsb	
temperature	Temperature (Celsius degrees) from the node's internal sensor. The value of temperature is a signed integer. Only available from firmware v2.x.x
x_calibration_msb x_calibration_lsb	Reference value from the sensor associated to the X axis.
y_calibration_msb y_calibration_lsb	Reference value from the sensor associated to the Y axis.
z_calibration_msb z_calibration_lsb	Reference value from the sensor associated to the Z axis.
battery_level_msb battery_level_lsb	Battery voltage in millivolts.

This class will generate automatically the response for this frame type. It consists of 8 bytes with date and time information which will be sent to the node according to this structure:

Byte	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
0	Timestamp (YY) º two last numbers of the year							
1	Timestamp (MM) º month with two numbers							
2	Timestamp (DD) º day with two numbers							
3	Timestamp (dow) º day of the week. 0 for Sunday, through 6 for Saturday.							
4	Timestamp (hh) º hour with two numbers							
5	Timestamp (mm) º minutes with two numbers							
6								
7								

Frame type 5 = Start frame number 2

Name	Firmware version.
CODE_ID	Firmware version.
NM_START	Beginning hour of the night mode.
NM_PERIOD	Duration in hours of the night mode.
NM_SLEEP_TIME	Sleep time between consecutive sensor measurements (during night mode).
NM_KEEP_ALIVE	Elapsed time since last transmission to trigger a Keep-Alive frame (during night mode).
RADIO_MODE	Selected transmission mode between Sigfox, LoRaWAN y their combinations.
SLEEP_TIME	Sleep time between consecutive sensor measurements.
KEEP_ALIVE	Elapsed time since last transmission to trigger a Keep-Alive frame.
THRESHOLD	Threshold for detecting a vehicle over the parking slot.

When using PHP as a programming language of the new service is not possible, using the raw information sent from the device will be mandatory.

S. Frames section of this document explains how to do some operations to get the real value of every parameter received and how to generate the answer to sent it back to the Smart Parking node.

Example

Payload: `0400003f34b909173f600fc2`

Node ID: `0102030405060708`.

Date: 13th February 2017 18:45:22.

responses.ini file content (response from the Libelium remote node service):

```
[devices]
0102030405060708 = “615d01991642”
```

configuration.ini file content (values saved in the web form):

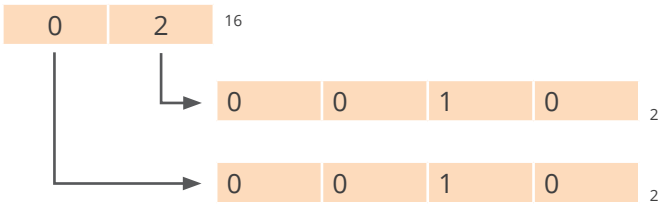
```
[0102030405060708]
DEVICE_NAME = “Set1”
SLEEP_TIME = 1
KEEP_ALIVE = 9
NM_STATUS = 1
NM_PERIOD = 9
NM_START = 22
NM_SLEEP_TIME = 2
NM_KEEP_ALIVE = 4
THRESHOLD = 93
BAT_READING = 0
ENABLE_DAILY_FRAME = 1
CONFIG_ID = 97
RADIO_MODE = 0
LORAWAN_MODE = 0
RESET_REQUEST = 0
```

Following the Frame section, we have 12 bytes (v1.x.x). Each byte is formed with two numbers, starting from the left. The result is:

byte	0	1	2	3	4	5	6	7	8	9	10	11
value	04	00	00	3f	34	b9	09	17	3f	60	0f	c2

Byte 0 has always the same format in all frames and it is compulsory to start with it in order to know which frame type is.

byte 0



byte	7	6	5	4	3	2	1	0
value	0	0	0	0	0	0	1	0

The information obtained in this step is:

- **Parking slot (bit 7)** 0 ® free
- **Battery state (bit 6)** 0 ® normal
- **Frame type (bits 0-3)** 0010 2 = 2 ® Frame type 2, daily update frame

Generate the answer according to the specifications explained in this section for this kind of frame. The first 6 bytes are for the value saved of this node from the responses.ini file (or from the request made to the Libelium remote node service explained in the section “How to create your own dashboard”). In this case, the value is 615d01991642. Then two bytes (6 and 7) have to be added following the instructions:

In byte 6, bits from 0 to 4 are booked for the hour of the current time, in this case, 18. The rest bits have no value. A conversion of the hour value to binary is needed to put all values according to the byte pattern: $18_{10} = 10010_2$

byte 6

byte	7	6	5	4	3	2	1	0
value	0	0	0	1	0	0	1	0

Convert the entire binary byte to hexadecimal format: $00010010_2 = 1216$. And this is the result for byte 6.

In byte 7, bits from 0 to 5 are reserved for the minutes of the current time, in this case, 45. Then, bit 6 is the reset value and bit 7 is the enable daily frame value. These last two values are taken from the configuration.ini file, where all the dashboard parameters are saved. In this case, the values are 0 and 1 respectively.

Change the minute value to binary to generate this byte: $45_{10} = 101101_2$. Now it is time for filling bit 6 and 7 with their values: 10101101_2.

byte 7

byte	7	6	5	4	3	2	1	0
value	1	0	1	0	1	1	0	1

A conversion from this binary value to hexadecimal format is needed to get byte 7: $10101101_2 = \mathbf{AD}_{16}$

Finally, the last step is to join the results of the bytes 6 and 7 to the value of the configuration.ini file to send it to the node. The result of the whole process is: **615D0199164212AD**.

13. Troubleshooting

13.1. Windows does not recognize USB ports

Sometimes it is possible that your computer does not recognize the USB port where the board is plugged. It is because you do not have installed the proper driver for the device, but do not worry, we will tell you how to fix this issue.

Firstly, it is necessary to open the Device Manager in order to see what device is not being detected. To open this window depends on the Windows version you have, but usually typing “Device Manager” in the Start > Search option it is enough.

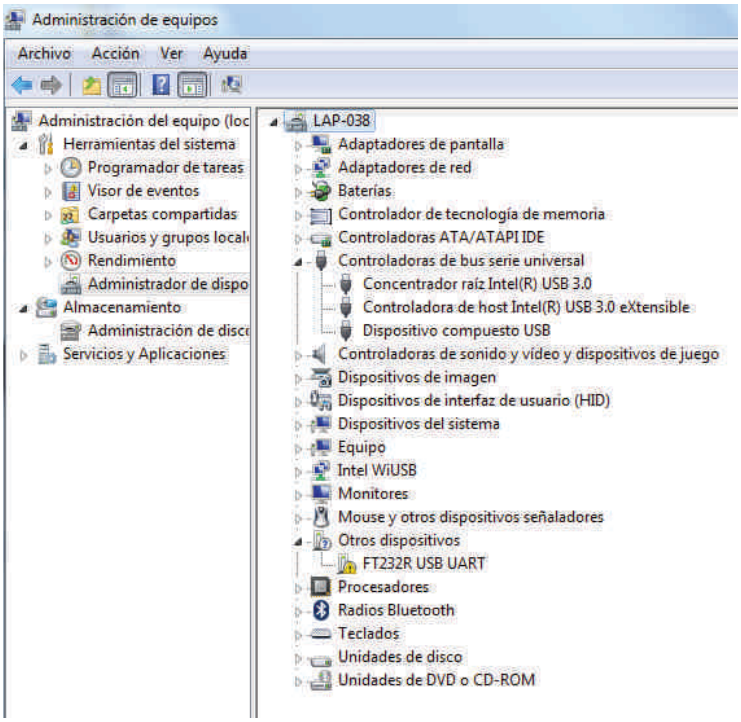


Figure : Windows Device Manager

The devices that are not detected are always marked with an alert icon as the next image shows. Right-click and select the first choice, “Update Driver Software”.

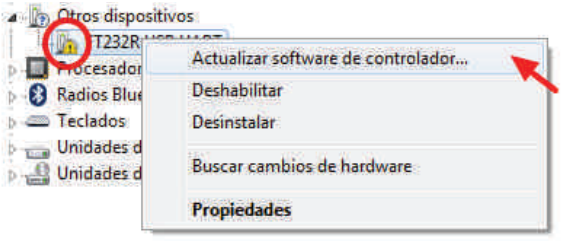


Figure : Update driver

Then, the driver must be searched in the computer, in particular the same path where the Smart Devices App has been installed.

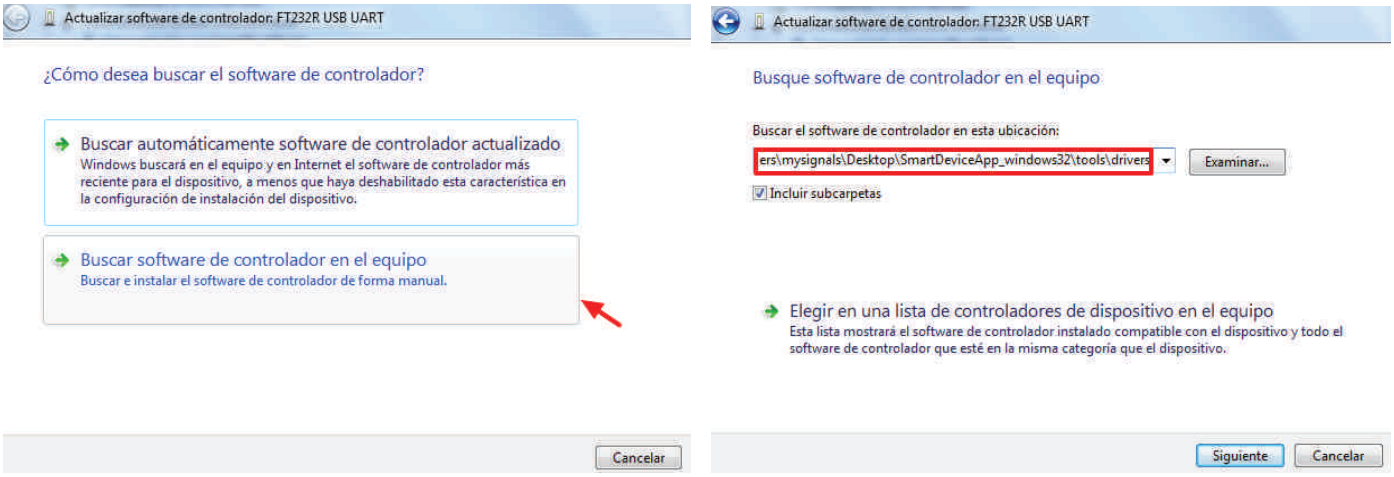


Figure : Search the driver in the computer

Once validated the path, the next pop-up will be displayed, noticing the driver is not verified. Users have to confirm clicking on “Install this driver software anyway”.

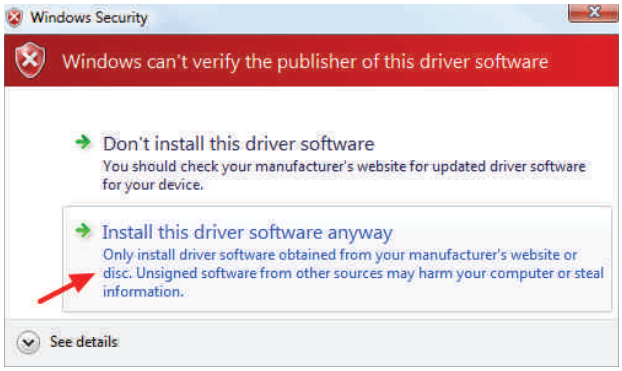


Figure : Security warning

After a while, the driver is installed.

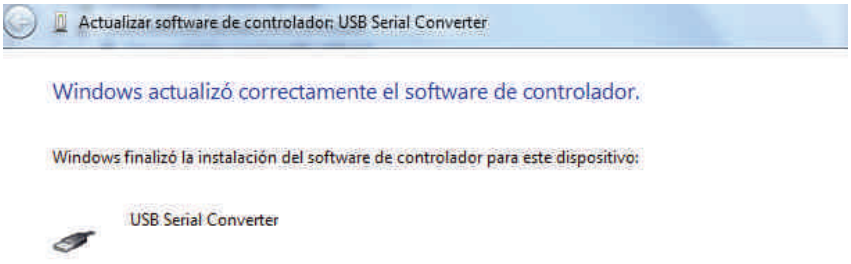


Figure : Driver installed

But the process is not finished. The first driver installed is “USB Serial Converter” but it is necessary to check the Device Manager list to verify the status. If the warning remains near the USB serial port, users have to repeat the “Update driver software” process again.

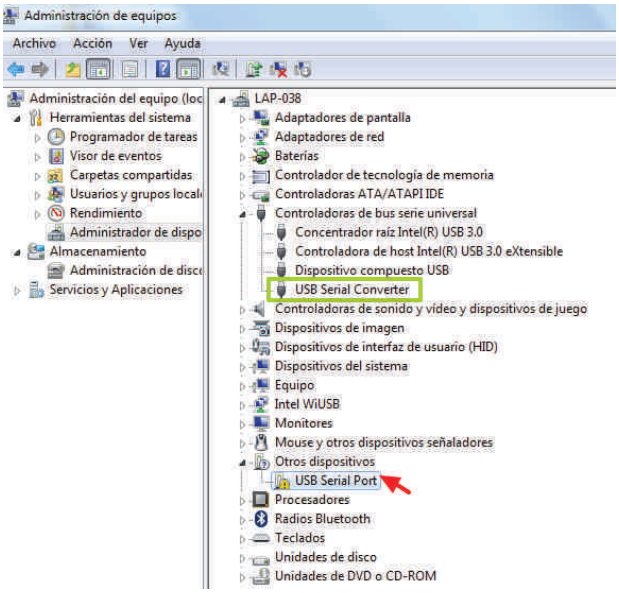


Figure : USB devices to verify

When the process is finished, users can check both drivers have been installed.

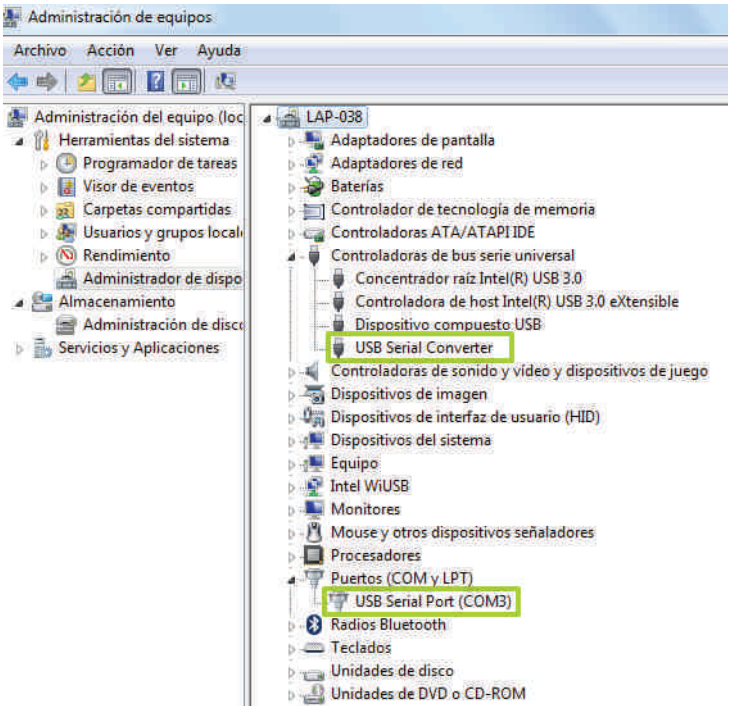


Figure : USB devices with all drivers installed

Now, the Smart Device App is ready to detect the ports and make the operation you want to do.

13.2. How to know the port where the device is plugged in

Users have to select in the Smart Devices App the port assigned to the Smart Parking node in the computers to make any operation with the Smart Parking node.

The assigned port depends on the USB devices that the users have plugged in their computers. The port displayed in the images of this guide may not correspond with the users port.

Follow these steps to discover the Smart Parking node port:

Step 1: Unplug the Smart Parking device from the computer and refresh the list. Take a look of all ports detected.

Figure : Port list without the node connected

Step 2: Plug the Smart Parking device and refresh the list again.

Figure : Port list with the node connected

Step 3: Compare the results with the previous ones and the new port detected is the Smart Parking node plugged in the last step.

Figure : Comparing the results

13.3. I cannot load nor save the configuration in the node

Sometimes users can experience that they cannot do any operation with the node in the Smart Devices App.

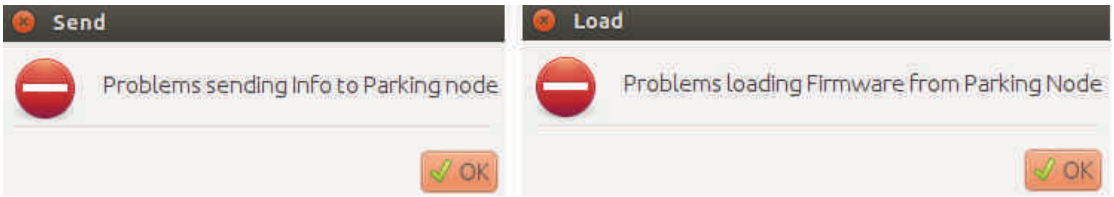


Figure : Error messages displayed in load / save operation

A message like one of the previous can be displayed. When the USB port is properly selected, the problem could be that the Smart Parking node has gone to deepsleep state. To solve it, power off the node and then power it on again. Then, click on “load” or “save” button as soon as possible, just before the node enter in deepsleep state (it takes 4 or 5 seconds).

14. Certifications

This document applies to the following Smart Parking model, approved for FCC:

Model	FCC ID
Smart Parking US	XKM-PARKING-V1

14.1. USA Certification:

Modification statement

Libelium has not approved any changes or modifications to this device by the user. Any changes or modifications could void the user's authority to operate the equipment.

Interference statement

This device complies with Part 15 of the FCC Rules license-exempt RSS standard(s). Operation is subject to the following two conditions: (1) this device may not cause interference, and (2) this device must accept any interference, including interference that may cause undesired operation of the device.

Wireless notice

This device complies with FCC radiation exposure limits set forth for an uncontrolled environment and meets the FCC radio frequency (RF) Exposure Guidelines rules. This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

This device needs to be installed and used on distance greater than 20 cm from human body.

For FCC Part 15 – Class B device: digital device or peripheral

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and receiver.
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

15. Disposal and recycling

In this section, the term “Waspmote” encompasses both the Waspmote device itself as well as it enclosure.

When Waspmote reaches the end of its useful life, it must be taken to an electronic equipment recycling point.

The equipment must be disposed of in a selective waste collection system, and not that for urban solid residue. Please manage its disposal properly.

Your distributor will inform you about the most appropriate and environmentally friendly disposal process for the used product and its packaging.