

# Events Board v30 / Plug & Sense!

## Security Guide



Document version: v7.2 - 06/2019  
© Libelium Comunicaciones Distribuidas S.L.

# INDEX

<b>1. General .....</b>	<b>5</b>
1.1. General and safety information .....	5
1.2. Conditions of use .....	5
<b>2. New version: Events Sensor Board v3.0.....</b>	<b>6</b>
<b>3. Wasp mote Plug &amp; Sense!.....</b>	<b>7</b>
3.2. General view .....	8
3.2.1. Specifications.....	8
3.2.2. Parts included.....	11
3.2.3. Identification.....	12
3.4. Solar powered .....	15
3.5. External Battery Module.....	16
3.6. Programming the Nodes.....	17
3.7. Program in minutes.....	18
3.8. Radio interfaces .....	19
3.9. Industrial Protocols.....	20
3.10. GPS.....	22
3.11. Models .....	23
3.11.1. Smart Security.....	24
<b>4. Hardware.....</b>	<b>26</b>
4.1. General Description .....	26
4.2. Specifications .....	26
4.3. Electrical Characteristics.....	26
<b>5. Sensors .....</b>	<b>27</b>
5.1. Temperature, humidity and pressure sensor .....	27
5.1.1. Specifications.....	27
5.1.2. Measurement process .....	28
5.1.3. Socket 5 .....	29
5.2. Liquid Level Sensors (PTFA3415, PTFA0100, PTFA1103) .....	30
5.2.1. Specifications.....	30
5.2.2. Measurement Process.....	31
5.3. Presence Sensor (PIR) .....	33
5.3.1. Specifications.....	33
5.3.2. Measurement Process.....	33
5.4. Hall Effect Sensor .....	35

5.4.1. Specifications.....	35
5.4.2. Measurement Process.....	35
5.5. Water Leakage / Liquid Detection sensor (Point) .....	36
5.5.1. Specifications.....	36
5.5.2. Measurement Process.....	36
5.6. Liquid Flow Sensors .....	37
5.6.1. Specifications.....	37
5.6.2. Measurement Process.....	38
5.7. Water Leakage / Liquid Detection sensor (Line) .....	39
5.7.1. Specifications.....	39
5.7.2. Measurement process .....	39
5.8. Relay Input-Output .....	42
5.8.1. Specifications.....	42
5.8.2. Precautions for safe use .....	42
5.8.3. Introduction .....	42
5.8.4. Relay input example.....	43
5.8.5. Relay output example .....	44
5.9. Relay Input-Output in Wasp mote Plug & Sense! .....	45
5.10. Ultrasound sensor (MaxSonar® from MaxBotix™).....	46
5.10.1. Specifications .....	46
5.10.2. Measurement process.....	48
5.10.3. Socket .....	48
5.11. Luminosity sensor (Luxes accuracy) .....	49
5.11.1. Specifications .....	49
5.11.2. Measurement process.....	49
5.12. Design and connections .....	50
5.12.1. Digital bus .....	50
5.12.2. Socket 1, Socket 2, Socket 3, Socket 4 and Socket 6 .....	50
5.12.3. Liquid flow.....	50
5.12.4. Relay Input-Output .....	50
5.12.5. Sockets for casing .....	50
5.13. Library.....	52
5.13.1. Events class .....	52
5.13.2. hallSensorClass .....	53
5.13.3. liquidLevelClass .....	53
5.13.4. liquidPresenceClass .....	53
5.13.5. pirSensorClass .....	54
5.13.6. flowClass .....	54
5.13.7. relayClass .....	54

<b>6. API changelog .....</b>	<b>56</b>
-------------------------------	-----------

<b>7. Documentation changelog .....</b>	<b>57</b>
---	-----------

<b>8. Consumption .....</b>	<b>58</b>
-----------------------------	-----------

8.1. Power control .....	58
8.2. Low Consumption Mode.....	58
<b>9. Maintenance .....</b>	<b>59</b>
<b>10. Disposal and recycling .....</b>	<b>60</b>
<b>11. Certifications.....</b>	<b>61</b>

# 1. General

## 1.1. General and safety information

- In this section, the term “Waspote” encompasses both the Waspote device itself and its modules and sensor boards.
- 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.
- Keep the device in a dry place and away from any liquid which may spill.
- Waspote 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 using a car lighter as a power supply, be sure to respect the voltage and current data specified in the “Power Supplies” section.
- If using a battery in combination or not with a solar panel as a power supply, be sure to use the voltage and current data specified in the “Power supplies” section.
- If a software or 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.
- Waspote is a device to be integrated in a casing so that it is protected from environmental conditions such as light, dust, humidity or sudden changes in temperature. The board supplied “as is” is not recommended for a final installation as the electronic components are open to the air and may be damaged.

## 1.2. Conditions of use

- Read the “General and Safety Information” section carefully and keep the manual for future consultation.
- Use Waspote in accordance with the electrical specifications and the environment described in the “Electrical Data” section of this manual.
- Waspote and its components and modules are supplied as electronic boards to be integrated within a final product. This product must contain an enclosure to protect it from dust, humidity and other environmental interactions. In the event of outside use, this enclosure must be rated at least IP65.
- Do not place Waspote in contact with metallic surfaces; they could cause short-circuits which will permanently damage it.

Further information you may need can be found at <http://www.libelium.com/development/waspote>

The “General Conditions of Libelium Sale and Use” document can be found at:

[http://www.libelium.com/development/waspote/technical\\_service](http://www.libelium.com/development/waspote/technical_service)

## 2. New version: Events Sensor Board v3.0

This guide explains the new Events Sensor Board v3.0. This board was specifically designed for our new product lines Waspote v15 and Plug & Sense! v15, released on October 2016.

This board is not compatible with Waspote v12 or Plug & Sense! v12, so it is NOT recommended to mix product generations. If you are using previous versions of our products, please use the corresponding guides, available on our [Development website](#).

You can get more information about the generation change on the document "[New generation of Libelium product lines](#)".

Differences of Events Sensor Board v3.0 with the previous versions:

- New Digital Bus sockets have been added to allow the connection of Digital Bus sensors, such the temperature, humidity and air pressure sensor.
- Added a Digital Bus isolator chip to avoid affecting to the Waspote Digital Bus.
- Now the board includes a new relay output to control small external loads or even other relays.
- It is possible to use potential-free relay inputs to detect the state of external switches like other relays, etc.
- The library has been improved to make easier the board handling.
- New connectors to improve the Plug & Sense! wiring, making it more robust.
- Temperature sensor MCP9700A replaced by BME280.
- Humidity sensor 808H5V5 replaced by BME280.
- Atmospheric pressure sensor MPX4115A replaced by BME280.
- Temperature and humidity sensor SHT75 Sensirion replaced by BME280.

### 3. Wasmote Plug & Sense!

The Wasmote Plug & Sense! line allows you to easily deploy Internet of Things networks in an easy and scalable way, ensuring minimum maintenance costs. The platform consists of a robust waterproof enclosure with specific external sockets to connect the sensors, the solar panel, the antenna and even the USB cable in order to reprogram the node. It has been specially designed to be scalable, easy to deploy and maintain.

**Note:** For a complete reference guide download the “Wasmote Plug & Sense! Technical Guide” in the [Development section](#) of the [Libelium website](#).

#### 3.1. Features

- Robust waterproof IP65 enclosure
- Add or change a sensor probe in seconds
- Solar powered external panel option
- Radios available: 802.15.4, 868 MHz, 900 MHz, WiFi, 4G, Sigfox and LoRaWAN
- Over the air programming (OTAP) of multiple nodes at once (via WiFi or 4G radios)
- Special holders and brackets ready for installation in street lights and building fronts
- Graphical and intuitive interface Programming Cloud Service
- Built-in, 3-axes accelerometer
- External, contactless reset with magnet
- Optional industrial protocols: RS-485, Modbus, CAN Bus
- Optional GPS receiver
- Optional External Battery Module
- External SIM connector for the 4G models
- Fully certified: CE (Europe), FCC (USA), IC (Canada), ANATEL (Brazil), RCM (Australia), PTCRB (USA, cellular connectivity), AT&T (USA, cellular connectivity)



Figure: Wasmote Plug & Sense!

## 3.2. General view

This section shows main parts of Waspote Plug & Sense! and a brief description of each one. In later sections all parts will be described deeply.

### 3.2.1. Specifications

- **Material:** polycarbonate
- **Sealing:** polyurethane
- **Cover screws:** stainless steel
- **Ingress protection:** IP65
- **Impact resistance:** IK08
- **Rated insulation voltage AC:** 690 V
- **Rated insulation voltage DC:** 1000 V
- **Heavy metals-free:** Yes
- **Weatherproof:** true - nach UL 746 C
- **Ambient temperature (min.):** -30 °C\*
- **Ambient temperature (max.):** 70 °C\*
- **Approximated weight:** 800 g

\* Temporary extreme temperatures are supported. Regular recommended usage: -20, +60 °C.

In the pictures included below it is shown a general view of Waspote Plug & Sense! main parts. Some elements are dedicated to node control, others are designated to sensor connection and other parts are just identification elements. All of them will be described along this guide.

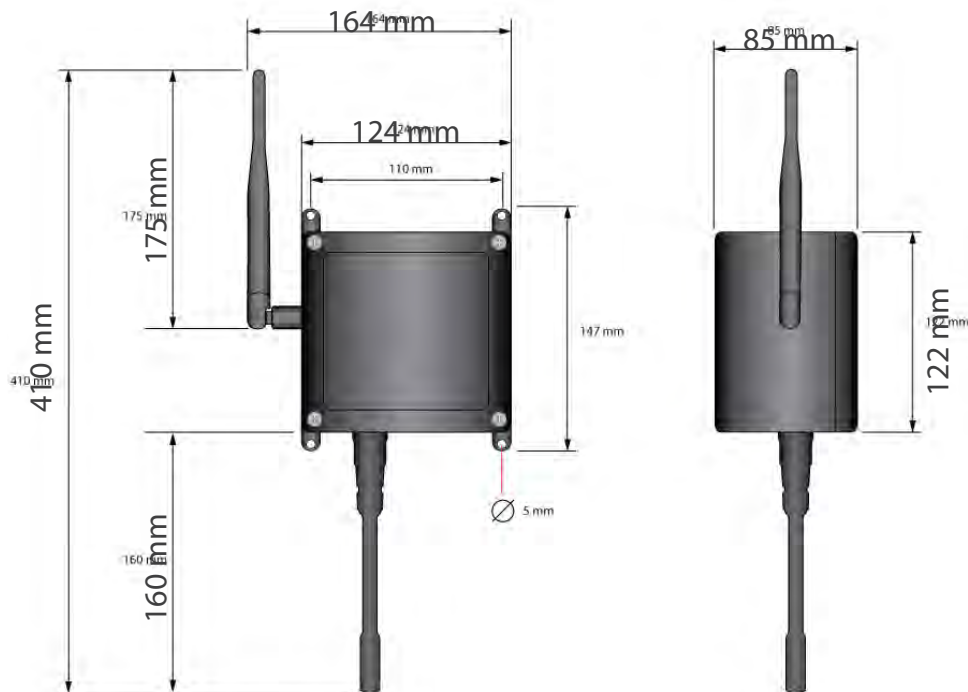


Figure : Main view of Waspote Plug & Sense!

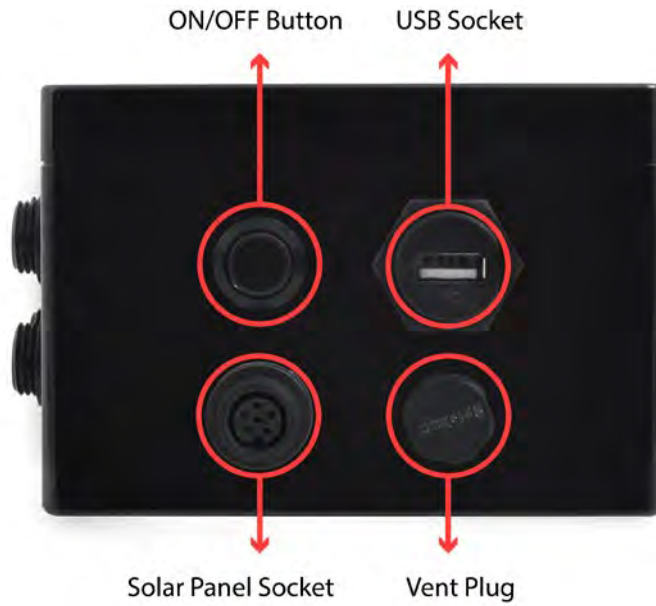
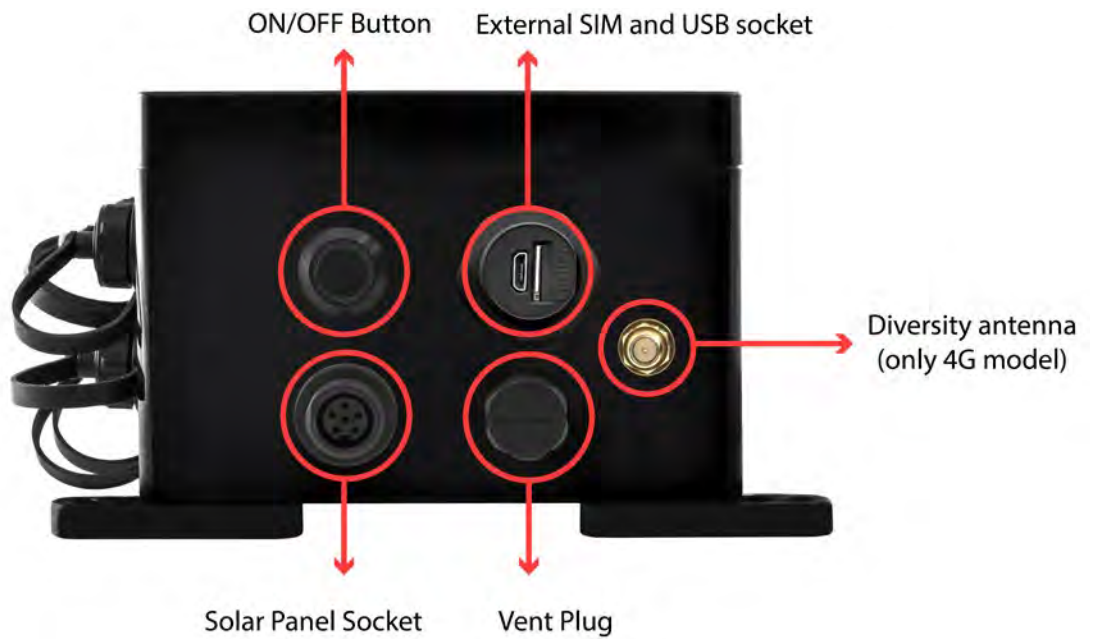


Figure : Control side of the enclosure



Control side of the enclosure for 4G model

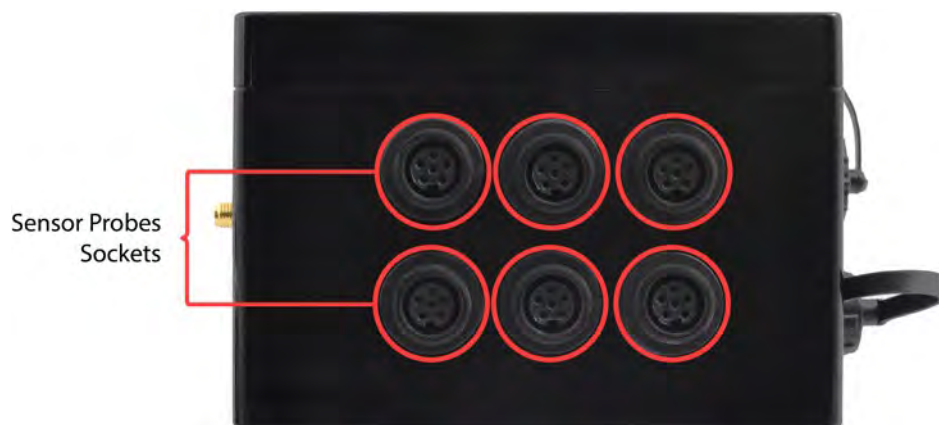


Figure : Sensor side of the enclosure



Figure : Antenna side of the enclosure



Figure : Front view of the enclosure



Figure : Back view of the enclosure



Figure : Warranty stickers of the enclosure

**Important note:** Do not handle black stickers seals of the enclosure (Warranty stickers). Their integrity is the proof that Wasmote Plug & Sense! has not been opened. If they have been handled, damaged or broken, the warranty is automatically void.

### 3.2.2. Parts included

Next picture shows Wasmote Plug & Sense! and all of its elements. Some of them are optional accessories that may not be included.



Figure : Wasmote Plug & Sense! accessories: 1 enclosure, 2 sensor probes, 3 external solar panel, 4 USB cable, 5 antenna, 6 cable ties, 7 mounting feet (screwed to the enclosure), 8 extension cord, 9 solar panel cable, 10 wall plugs & screws

### 3.2.3. Identification

Each Waspote model is identified by stickers. Next figure shows front sticker.

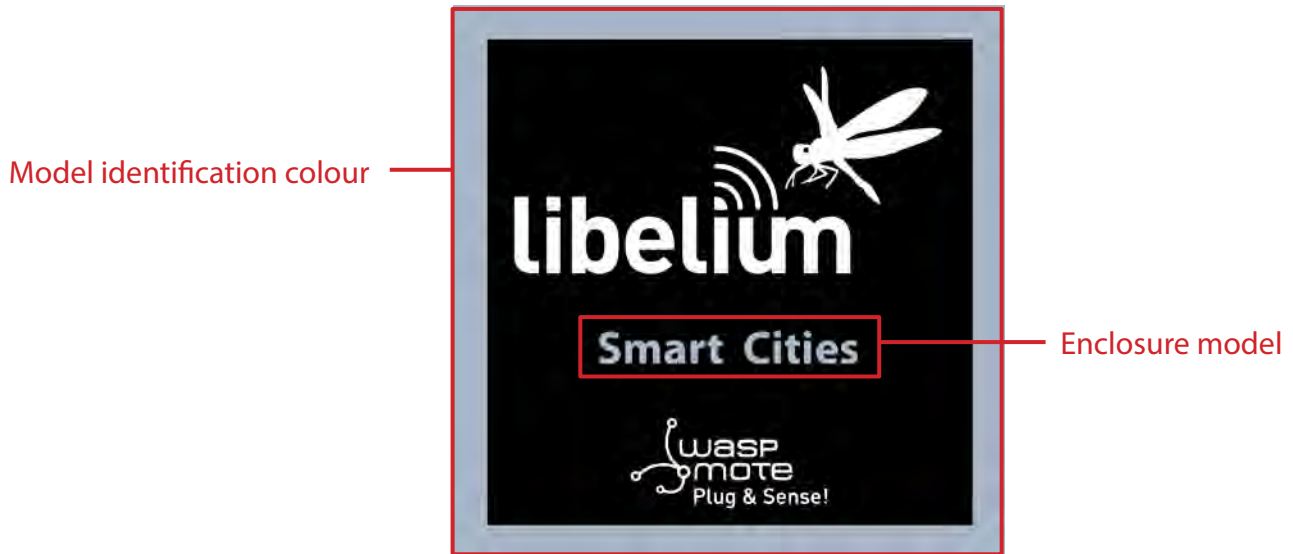


Figure : Front sticker of the enclosure

There are many configurations of Waspote Plug & Sense! line, all of them identified by one unique sticker. Next image shows all possibilities.



Figure : Different front stickers

Moreover, Waspote Plug & Sense! includes a back sticker where it is shown identification numbers, radio MAC addresses, etc. It is highly recommended to annotate this information and save it for future maintenance. Next figure shows it in detail.



Figure : Back sticker

Sensor probes are identified too by a sticker showing the measured parameter and the sensor manufacturer reference.



Figure : Sensor probe identification sticker

### 3.3. Sensor probes

Sensor probes can be easily attached by just screwing them into the bottom sockets. This allows you to add new sensing capabilities to existing networks just in minutes. In the same way, sensor probes may be easily replaced in order to ensure the lowest maintenance cost of the sensor network.



Figure : Connecting a sensor probe to Waspote Plug & Sense!

Go to the [Plug & Sense! Sensor Guide](#) to know more about our sensor probes.

### 3.4. Solar powered

The battery can be recharged using the waterproof USB cable but also the external solar panel option.

The external solar panel is mounted on a 45° holder which ensures the maximum performance of each outdoor installation.



*Figure : Waspote Plug & Sense! powered by an external solar panel*

### 3.5. External Battery Module

The External Battery Module (EBM) is an accessory to extend the battery life of Plug & Sense!. The extension period may be from months to years depending on the sleep cycle and radio activity. The daily charging period is selectable among 5, 15 and 30 minutes with a selector switch and it can be combined with a solar panel to extend even more the node's battery lifetime.

Note: Nodes using solar panel can keep using it through the External Battery Module. The EBM is connected to the solar panel connector of Plug & Sense! and the solar panel unit is connected to the solar panel connector of the EBM.

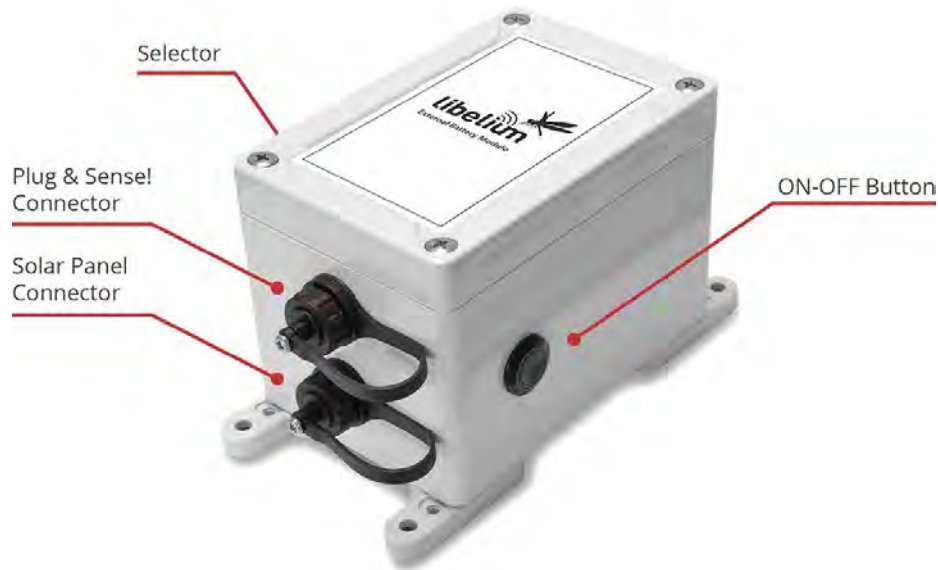


Figure : Plug & Sense! with External Battery Module



Figure : Plug & Sense! with External Battery Module and solar panel

### 3.6. Programming the Nodes

Waspote Plug & Sense! can be reprogrammed in two ways:

The basic programming is done from the USB port. Just connect the USB to the specific external socket and then to the computer to upload the new firmware.

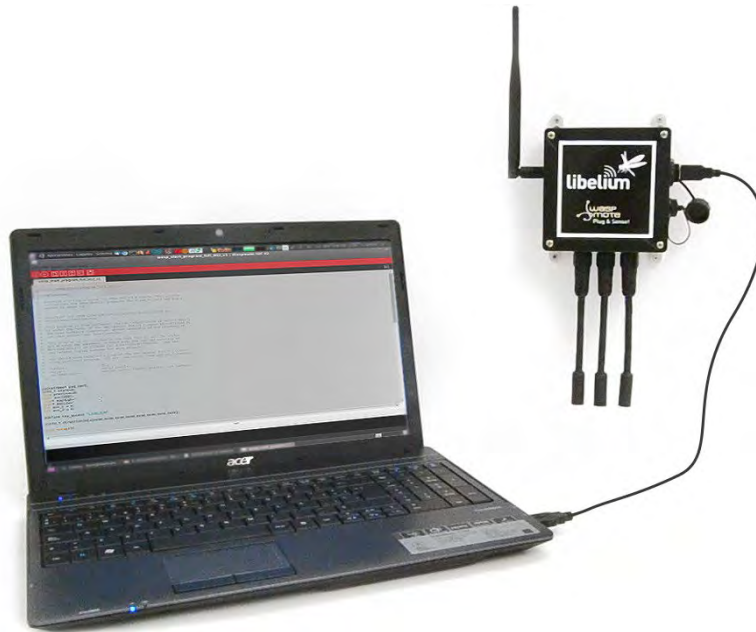


Figure : Programming a node

Over the Air Programming (OTAP) is also possible once the node has been installed (via WiFi or 4G radios). With this technique you can reprogram, wireless, one or more Waspote sensor nodes at the same time by using a laptop and Meshlium.



Figure : Typical OTAP process

## 3.7. Program in minutes

The Programming Cloud Service is an intuitive graphic interface which creates code automatically. The user just needs to fill a web form to obtain binaries for Plug & Sense!. Advanced programming options are available, depending on the license selected.

Check how easy it is to handle the Programming Cloud Service at:

<https://cloud.libelium.com/>

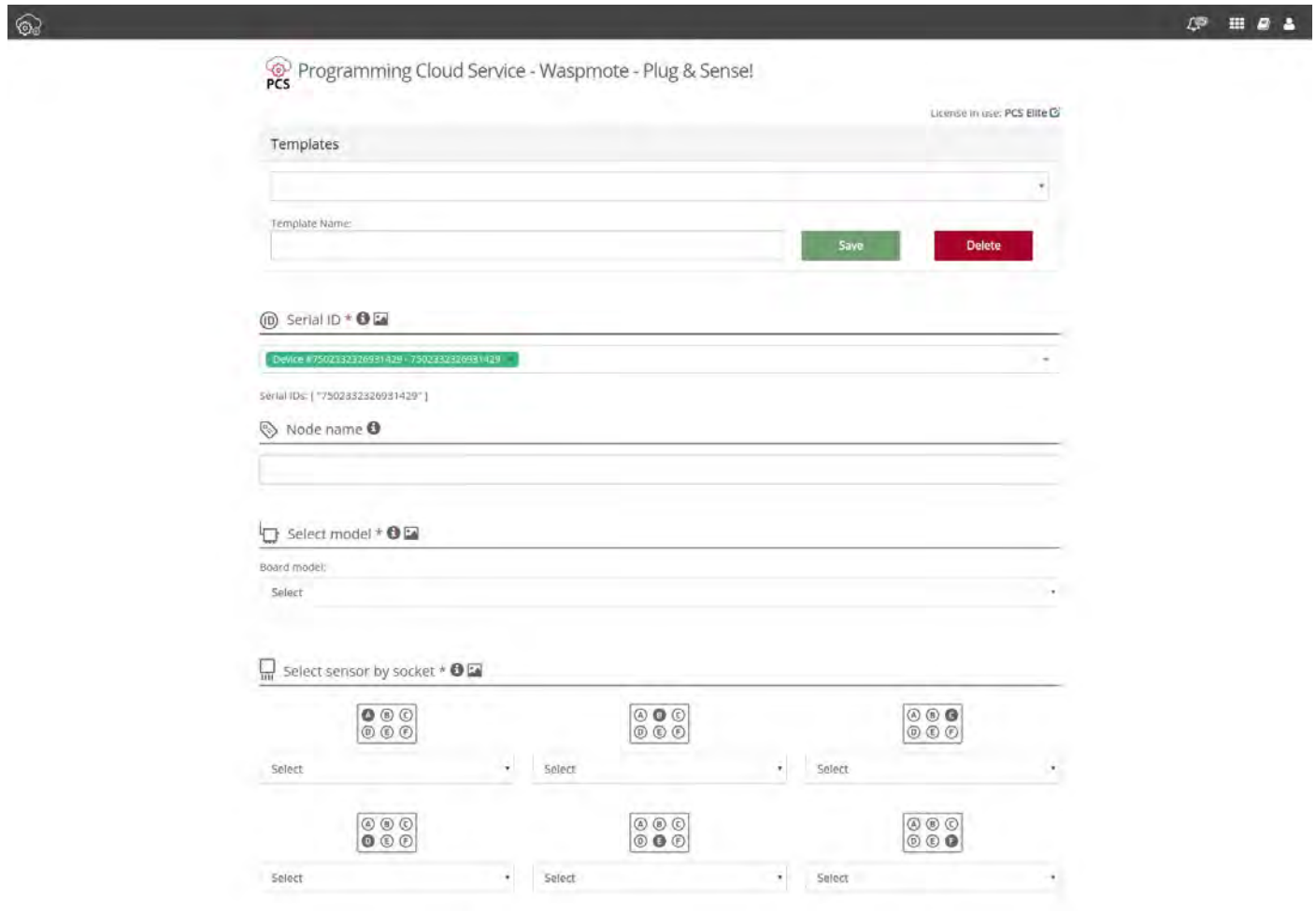


Figure: Programming Cloud Service

## 3.8. Radio interfaces

Radio	Protocol	Frequency bands	Transmission power	Sensitivity	Range*	Certification
XBee-PRO 802.15.4 EU	802.15.4	2.4 GHz	10 dBm	-100 dBm	750 m	CE
XBee-PRO 802.15.4	802.15.4	2.4 GHz	18 dBm	-100 dBm	1600 m	FCC, IC, ANATEL, RCM
XBee 868LP	RF	868 MHz	14 dBm	-106 dBm	8.4 km	CE
XBee 900HP US	RF	900 MHz	24 dBm	-110 dBm	15.5 km	FCC, IC
XBee 900HP BR	RF	900 MHz	24 dBm	-110 dBm	15.5 km	ANATEL
XBee 900HP AU	RF	900 MHz	24 dBm	-110 dBm	15.5 km	RCM
WiFi	WiFi (HTTP(S), FTP, TCP, UDP)	2.4 GHz	17 dBm	-94 dBm	500 m	CE, FCC, IC, ANATEL, RCM
4G EU/BR	4G/3G/2G  (HTTP, FTP, TCP, UDP)  GPS	800, 850, 900, 1800, 2100, 2600 MHz	4G: class 3 (0.2 W, 23 dBm)	4G: -102 dBm	- km - Typical base station range	CE, ANATEL
4G US v2	4G/3G  (HTTP, FTP, TCP, UDP)	700, 850, 1700, 1900 MHz	4G: class 3 (0.2 W, 23 dBm)	4G: -103 dBm	- km - Typical base station range	FCC, IC, PTCRB, AT&T
4G AU	4G (HTTP, FTP, TCP, UDP)	700, 1800, 2600 MHz	4G: class 3 (0.2 W, 23 dBm)	4G: -102 dBm	- km - Typical base station range	RCM
Sigfox EU	Sigfox	868 MHz	16 dBm	-126 dBm	- km - Typical base station range	CE
Sigfox US	Sigfox	900 MHz	24 dBm	-127 dBm	- km - Typical base station range	FCC, IC
Sigfox AU / APAC / LATAM	Sigfox	900 MHz	24 dBm	-127 dBm	- km - Typical base station range	-
LoRaWAN EU	LoRaWAN	868 MHz	14 dBm	-136 dBm	> 15 km	CE
LoRaWAN US	LoRaWAN	902-928 MHz	18.5 dBm	-136 dBm	> 15 km	FCC, IC
LoRaWAN AU	LoRaWAN	915-928 MHz	18.5 dBm	-136 dBm	> 15 km	-
LoRaWAN IN	LoRaWAN	865-867 MHz	18.5 dBm	-136 dBm	> 15 km	-
LoRaWAN ASIA-PAC / LATAM	LoRaWAN	923 MHz	18.5 dBm	-136 dBm	> 15 km	-

\* Line of sight and Fresnel zone clearance with 5 dBi dipole antenna.

### 3.9. Industrial Protocols

Besides the main radio of Waspote Plug & Sense!, it is possible to have an Industrial Protocol module as a secondary communication option. This is offered as an accessory feature.

The available Industrial Protocols are RS-485, Modbus (software layer over RS-485) and CAN Bus. This optional feature is accessible through an additional, dedicated socket on the antenna side of the enclosure.



Figure: Industrial Protocols available on Plug & Sense!

Finally, the user can choose between 2 probes to connect the desired Industrial Protocol: A standard DB9 connector and a waterproof terminal block junction box. These options make the connections on industrial environments or outdoor applications easier.



*Figure: DB9 probe*



*Figure: Terminal box probe*

## 3.10. GPS

Any Plug & Sense! node can incorporate a GPS receiver in order to implement real-time asset tracking applications. The user can also take advantage of this accessory to geolocate data on a map. An external, waterproof antenna is provided; its long cable enables better installation for maximum satellite visibility.



Figure : Plug & Sense! node with GPS receiver

Chipset: JN3 (Telit)

Sensitivity:

- Acquisition: -147 dBm
- Navigation: -160 dBm
- Tracking: -163 dBm

Hot start time: <1 s

Cold start time: <35 s

Positional accuracy error < 2.5 m

Speed accuracy < 0.01 m/s

EGNOS, WAAS, GAGAN and MSAS capability

Antenna:

- Cable length: 2 m
- Connector: SMA
- Gain: 26 dBi (active)

Available information: latitude, longitude, altitude, speed, direction, date&time and ephemeris management

## 3.11. Models

There are some defined configurations of Waspote Plug & Sense! depending on which sensors are going to be used. Waspote Plug & Sense! configurations allow to connect up to six sensor probes at the same time.

Each model takes a different conditioning circuit to enable the sensor integration. For this reason, each model allows connecting just its specific sensors.

This section describes each model configuration in detail, showing the sensors which can be used in each case and how to connect them to Waspote. In many cases, the sensor sockets accept the connection of more than one sensor probe. See the compatibility table for each model configuration to choose the best probe combination for the application.

It is very important to remark that each socket is designed only for one specific sensor, so **they are not interchangeable**. Always be sure you have connected the probes in the right socket. Otherwise, they can be damaged.

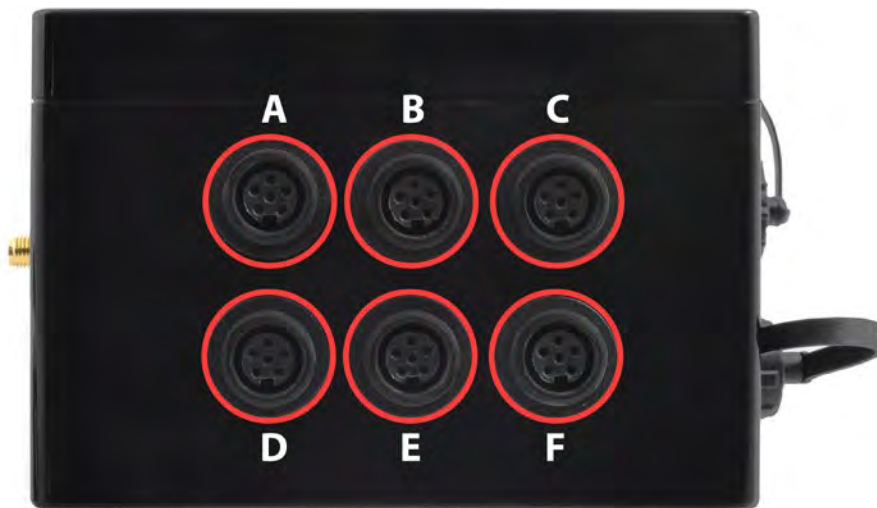


Figure : Identification of sensor sockets

### 3.11.1. Smart Security

The main applications for this Waspote Plug & Sense! configuration are perimeter access control, liquid presence detection and doors and windows openings. Besides, a relay system allows this model to interact with external electrical machines.



Figure: Smart Security Waspote Plug & Sense! model

**Note:** The probes attached in this photo could not match the final location. See next table for the correct configuration.

Sensor Socket	Sensor probes allowed for each sensor socket	
	Parameter	Reference
A, C, D or E	Temperature + Humidity + Pressure	9370-P
	Luminosity (Luxes accuracy)	9325-P
	Ultrasound (distance measurement)	9246-P
	Presence - PIR	9212-P
	Liquid Level (combustible, water)	9239-P, 9240-P
	Liquid Presence (Point, Line)	9243-P, 9295-P
	Hall Effect	9207-P
B	Liquid Flow (small, medium)	9296-P, 9297-P
F	Relay Input-Output	9270-P

Figure: Sensor sockets configuration for Smart Security model

As we see in the figure below, thanks to the directional probe, the presence sensor probe (PIR) may be placed in different positions. The sensor can be focused directly to the point we want.



Figure: Configurations of the Presence sensor probe (PIR)

**Note:** For more technical information about each sensor probe go to the [Development section](#) on the Libelium website.

## 4. Hardware

### 4.1. General Description

The sensors are **active** on the Events Sensor Board Sensor while Waspote is in **Sleep** or **Deep Sleep mode**. When a sensor changes its output signal to a “high” value, a signal is generated which wakes the mote from its low consumption status and tells it which sensor has generated the signal.

If Waspote were in **active** mode (on), it would receive the interruption and could respond to it in the same way as the previous case.

#### Operation:

The board allows simultaneous connection with up to **5** sensors whose outputs are combined in an **OR** logical gate which implements a change in the interruption bit which wakes the mote. You can also connect different I2C devices, in addition to a relay output and a relay input.

The sensor that has interrupted the mote is identified in a **shift register** which can be read by Waspote once it is in normal operation.

### 4.2. Specifications

Weight:	20 g
Dimensions:	73.5 x 51 x 1.3 mm
Temperature Range:	[-20 °C, 65 °C]

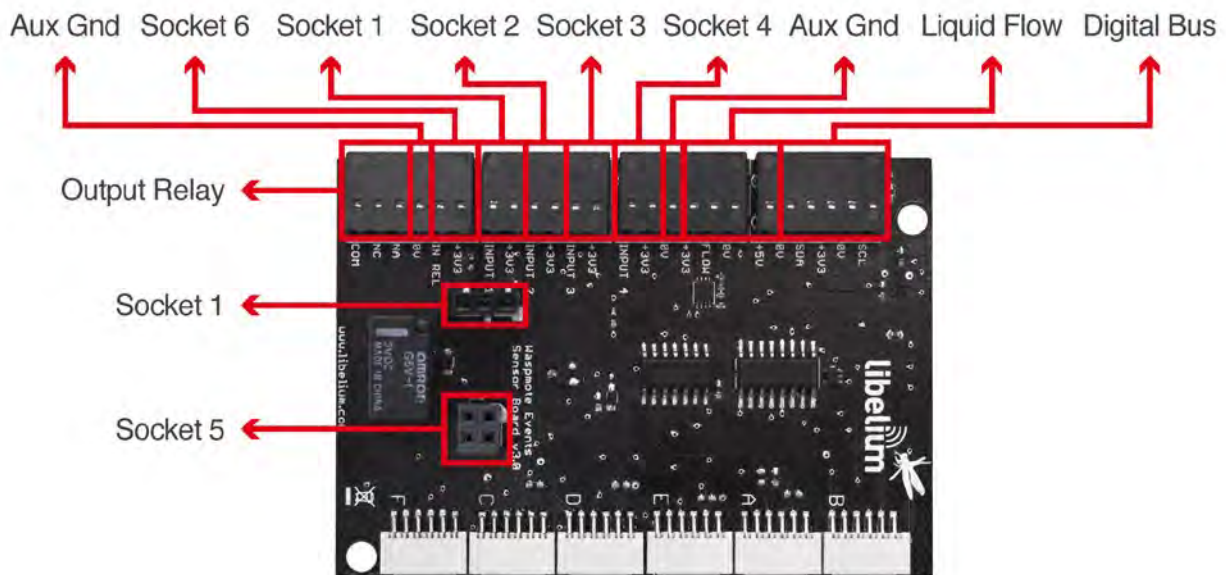


Figure : Upper side

### 4.3. Electrical Characteristics

• Board power voltages:	3.3 V
• Sensor power voltage:	3.3 V
• Maximum admitted current (continuous):	200 mA
• Maximum admitted current (peak):	400 mA

## 5. Sensors

### 5.1. Temperature, humidity and pressure sensor

#### 5.1.1. Specifications

##### Electrical characteristics

Supply voltage: 3.3 V  
 Sleep current typical: 0.1  $\mu$ A  
 Sleep current maximum: 0.3  $\mu$ A

##### Temperature sensor

Operational range: -40 ~ +85 °C  
 Full accuracy range: 0 ~ +65 °C  
 Accuracy:  $\pm 1$  °C (range 0 °C ~ +65 °C)  
 Response time: 1.65 seconds (63% response from +30 to +125 °C).  
 Typical consumption: 1  $\mu$ A measuring

##### Humidity sensor

Measurement range: 0 ~ 100% of relative humidity (for temperatures < 0 °C and > 60 °C see figure below)  
 Accuracy: <  $\pm 3\%$  RH (at 25 °C, range 20 ~ 80%)  
 Hysteresis:  $\pm 1\%$  RH  
 Operating temperature: -40 ~ +85 °C  
 Response time (63% of step 90% to 0% or 0% to 90%): 1 second  
 Typical consumption: 1.8  $\mu$ A measuring

Maximum consumption: 2.8  $\mu$ A measuring



Figure : Temperature, humidity and pressure sensor

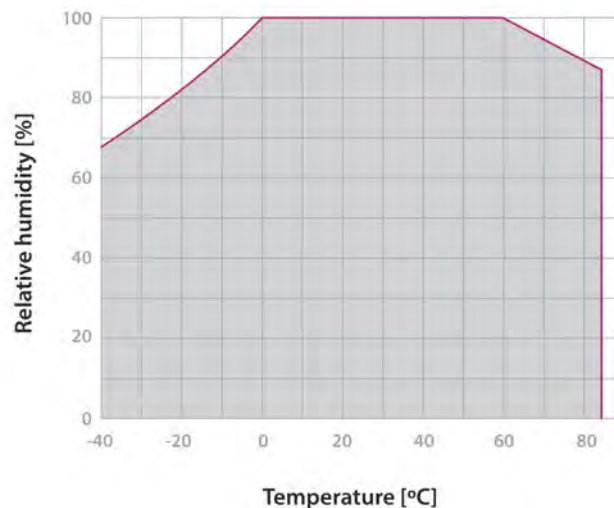


Figure : Humidity sensor operating range

##### Pressure sensor

Measurement range: 30 ~ 110 kPa  
 Operational temperature range: -40 ~ +85 °C  
 Full accuracy temperature range: 0 ~ +65 °C  
 Absolute accuracy:  $\pm 0.1$  kPa (0 ~ 65 °C)  
 Typical consumption: 2.8  $\mu$ A measuring  
 Maximum consumption: 4.2  $\mu$ A measuring

## 5.1.2. Measurement process

The BME280 is a combined digital humidity, pressure and temperature sensor based on proven sensing principles. The humidity sensor provides an extremely fast response time for fast context awareness applications and high overall accuracy over a wide temperature range.

The pressure sensor is an absolute barometric pressure sensor with extremely high accuracy and resolution and drastically lower noise.

The integrated temperature sensor has been optimized for lowest noise and highest resolution. Its output is used for temperature compensation of the pressure and humidity sensors and can also be used for estimation of the ambient temperature.

When the sensor is disabled, current consumption drops to 0.1  $\mu$ A.

To read this sensor, you should use the following functions:

### Temperature:

Reading code:

```
{
  float temperature;
  // Reads the BME280 sensor
  Events.ON();
  temperature = Events.getTemperature(); //Return Temperature from BME280
}
```

### Humidity:

Reading code:

```
float humidity;
// Reads the BME280 sensor
Events.ON();
humidity = EventsBoard.getHumidity(); //Return Humidity from BME280
}
```

### Pressure:

Reading code:

```
float pressure;
Events.ON();
// Reads the BME280 sensor
pressure = EventsBoard.getPressure(); //Return Atmospheric Pressure from BME280
}
```

You can find a complete example code for reading the BME280 sensor in the following link:

<http://www.libelium.com/development/waspmote/examples/ev-v30-01-temperature>

### 5.1.3. Socket 5

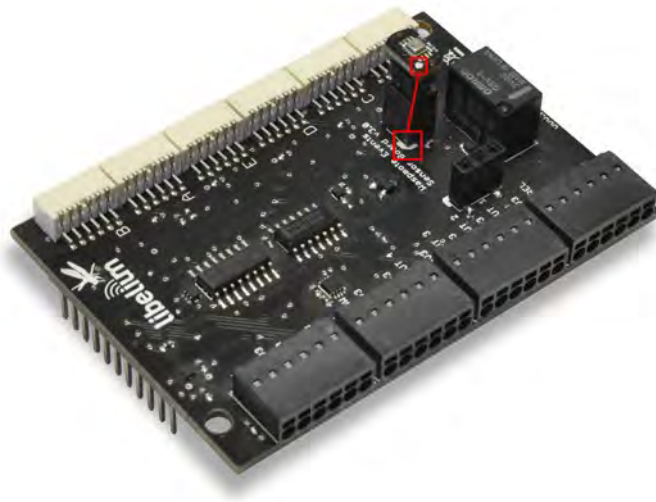


Figure : Image of the socket for the BME280 sensor

In the image above we can see highlighted the four pins of the terminal block where the sensor must be connected to the board. The white dot of the BME280 must match the mark on the Events Sensor Board.

## 5.2. Liquid Level Sensors (PTFA3415, PTFA0100, PTFA1103)

### 5.2.1. Specifications

#### PTFA3415

**Measurement Level:** Horizontal

**Liquids:** Water

**Material (box):** Propylene

**Material (float):** Propylene

**Operating Temperature:** -10 °C ~ +80 °C

**Minimum consumption:** 0 µA\*

*\*This sensor's consumption is included in the consumption ranges of the connectors on which they can be placed. (See section "Consumption table")*



Figure : PTFA3415 sensor

#### PTFA0100

**Measurement Level:** Horizontal

**Liquids:** Heavy oils and combustibles

**Material (box):** Polyamide

**Material (float):** Polyamide

**Operating temperature:** -10 °C ~ +80 °C

**Minimum consumption:** 0 µA\*

*\*This sensor's consumption is included in the consumption ranges of the connectors on which they can be placed. (See section "Consumption table")*



Figure : PTFA0100 sensor

#### PTFA1103

**Measurement Level:** Vertical

**Liquids:** Water

**Material (box):** Propylene

**Material (float):** Propylene

**Operating temperature:** -10 °C ~ +80 °C

**Minimum consumption:** 0 µA\*

*\*This sensor's consumption is included in the consumption ranges of the connectors on which they can be placed. (See section "Consumption table")*



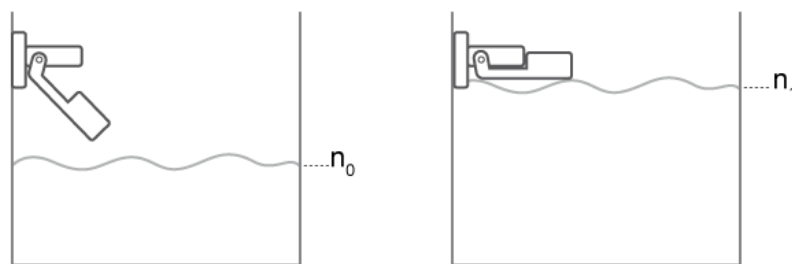
Figure : PTFA1103 sensor

### 5.2.2. Measurement Process

There are three liquid level sensors whose operation is based on the status of a switch which can be opened and closed (depending on its placing in the container) as the level of liquid moves the float at its end. The main differences between the three sensors, regarding its use in Waspote, are to be found in their process for placing them in the container (horizontal in the case of the PTFA3415 and PTFA0100 sensors, vertical for the PTFA1103 sensor) and in the material they are made of (the PTFA1103 and PTFA3415 sensors recommended for edible liquids and certain acids and the PTFA0100 for heavy oils and combustibles, more specific information can be found in the sensors' manual). Being switch sensors they can be placed in any of the socket 1, socket 2, socket 3, socket 4 and socket 6.

In the figure, two examples of applications of liquid level monitoring with these sensors can be seen.

#### Horizontal



#### Vertical

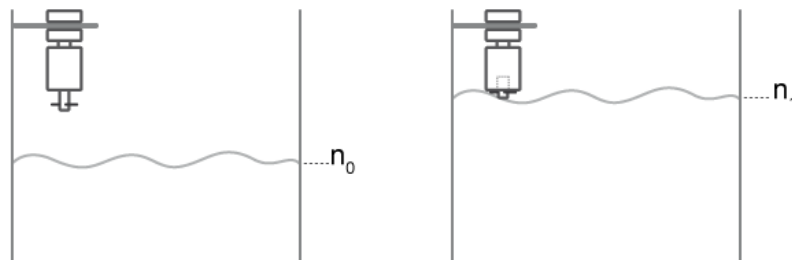


Figure : Monitoring the liquid level with PTFA sensors

This kind of sensors are yes/no sensors, in the sense that they inform:

- the liquid level is below the point where the sensor was installed
- the liquid level is above

The user may also be interested in the Ultrasound sensors (indoors/outdoors), which can be installed on top of a tank or silo and meter the distance from the top to the surface of liquid or grain, with great accuracy (cm). A code for reading the sensor is shown below:

```
uint8_t value;
//Instance Objects
liquidLevelClass liquidLevel(SOCKET_2);
{
  Events.ON();

  value = liquidLevel.readLiquidLevel();
}
```

`value` is an integer variable where the sensor state (a high value (3.3 V) or a low value (0 V), which will depend on the liquid level and sensor setup) will be stored.

Each sensor needs its own object. So, between the include of the library and the global variables declaration, the object must be created following the next structure:

- Name of the class: In this case `liquidLevelClass`
- Name of the object: In this case `liquidLevel`
- In brackets the socket selected for the sensor. In this case `SOCKET_2`

You can find a complete example code for reading the Liquid Level sensor in the following link:

<http://www.libelium.com/development/waspmote/examples/ev-v30-03-liquid-level-sensor/>

## 5.3. Presence Sensor (PIR)

### 5.3.1. Specifications

**Height:** 25.4 mm

**Width:** 24.3 mm

**Length:** 28.0 mm

**Consumption:** 100  $\mu$ A

**Range of detection:** 6 ~ 7 m

**Spectral range:** ~ 10  $\mu$ m



Figure : Image of the PIR presence sensor

### 5.3.2. Measurement Process

The PIR sensor (Passive Infra-Red) is a pyroelectric sensor mainly consisting of an infra-red receiver and a focusing lens that bases its operation on the monitoring of the variations in the levels of reception of detected infra-reds, reflecting this movement by setting its output signal high. Being a digital sensor it must be situated in socket 1, socket 2, socket 3, socket 4 and socket 6.

The 10  $\mu$ m spectrum corresponds to the radiation of heat from the majority of mammals as they emit temperatures around 36 °C.

The maximum detection direction goes perpendicular to the Events Sensor Board, this is why it is advised to place Wasp mote perpendicular to the ground when using the PIR sensor.



Figure : PIR maximun detection range

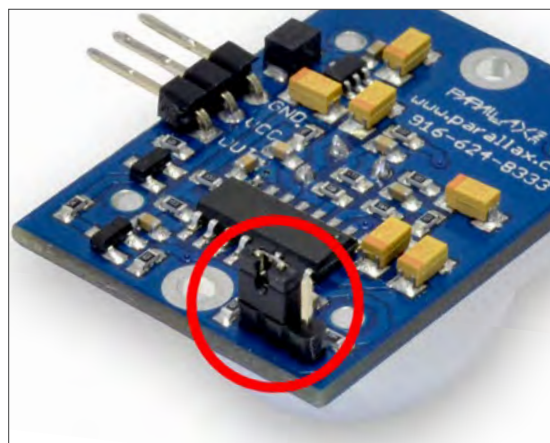


Figure : Maximun range jumper configuration

A code for reading the sensor is shown below:

```
uint8_t value;  
//Instance objects  
pirSensorClass pir(SOCKET_1);  
{  
  Events.ON();  
  
  value = pir.readPirSensor();  
}
```

`value` is an integer variable where the sensor state (a high value (3.3 V) or a low value (0 V), which will depend on the liquid level and sensor setup) will be stored.

Each sensor needs its own object. So, between the include of the library and the global variables declaration, the object must be created following the next structure:

- Name of the class: In this case `pirSensorClass`
- Name of the object: In this case `pir`
- In brackets the socket selected for the sensor. In this case `SOCKET_1`

You can find a complete example code for reading the PIR sensor in the following link:

<http://www.libelium.com/development/waspmote/examples/ev-v30-02-pir>

## 5.4. Hall Effect Sensor

### 5.4.1. Specifications

**Length:** 64 mm

**Width:** 19 mm

**Height:** 13 mm

**Maximum contact resistance (closed):** 200 mΩ

**Minimum contact resistance (open):** 100 GΩ

**Minimum consumption:** 0 μA\*

\*This sensor's consumption is included in the consumption ranges of the connectors on which they can be placed. (See section "Consumption table")



Figure : Image of the Hall effect sensor

### 5.4.2. Measurement Process

This is a magnetic sensor based on the Hall effect. The sensor's switch remains closed in the presence of a magnetic field, opening up in its absence. Together with its complementary magnet it can be used in applications of monitoring proximity or opening mechanisms. Being a switch sensor it must be placed on one of the socket 1, socket 2, socket 3, socket 4 and socket 6.

The next figure shows the way of using the sensor in an application of monitoring the opening of doors or windows.

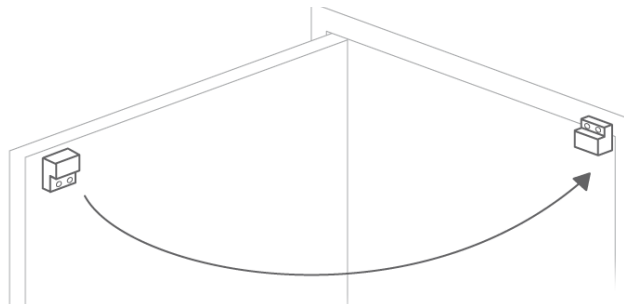


Figure : Application of the Hall effect sensor in monitoring opening mechanisms

A code for reading the sensor is shown below:

```
uint8_t value;
//Instance objects
hallSensorClass hall(SOCKET_3);
{
    Events.ON();

    value = hall.readHallSensor();
}
```

`value` is an integer variable where the sensor state (a high value (3.3 V) or a low value (0 V), which will depend on the liquid level and sensor setup) will be stored.

Each sensor needs its own object. So, between the include of the library and the global variables declaration, the object must be created following the next structure:

- Name of the class: In this case `hallSensorClass`.
- Name of the object: In this case `hall`.
- In brackets the socket selected for the sensor. In this case `SOCKET_3`

You can find a complete example code for reading the Hall Effect sensor in the following link:

<http://www.libelium.com/development/waspmote/examples/ev-v30-04-hall-efect>

## 5.5. Water Leakage / Liquid Detection sensor (Point)

### 5.5.1. Specifications

**Maximum Switching Voltage:** 100 V

**Operating temperature:** +5 °C ~ +80 °C

**Detectable liquids:** Water

**Minimum consumption:** 0 µA\*

*\*This sensor's consumption is included in the consumption ranges of the connectors on which they can be placed. (See section "Consumption table")*



Figure : Water Leakage / Liquid Detection sensor (Point)

### 5.5.2. Measurement Process

This sensor bases its operation on the variation in resistance between its two contacts in the presence of liquid to commute a switch reed from open to closed, commuting to open again when the liquid disappears (take care when it is used to detect liquids of high viscosity which may remain between the terminals blocking its drainage and preventing it from re-opening). Being a switch sensor it can be placed on any of the socket 1, socket 2, socket 3, socket 4 and socket 6.

The next figure shows an image of how the sensor must be placed to detect the presence of liquid.

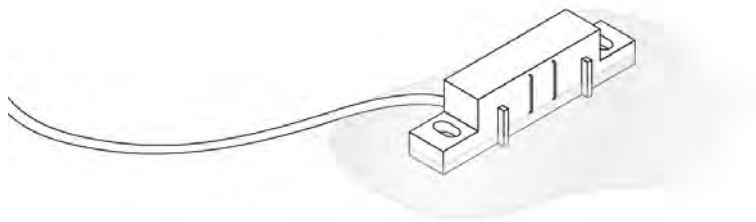


Figure : Application of the Water Leakage / Liquid Detection sensor (Point) for the detection of liquids

A code for reading the sensor is shown below:

```
uint8_t value;
//Instance object
liquidPresenceClass liquidPresence(SOCKET_4);
{
    Events.ON();

    value = lp.readliquidPresence();
}
```

`value` is an integer variable where the sensor state will be stored. A high value (3.3 V) or a low value (0 V), which will depend on the liquid level and sensor setup.

Each sensor needs its own object. So, between the include of the library and the global variables declaration, the object must be created following the next structure:

- Name of the class: In this case `liquidPresenceClass`
- Name of the object: In this case `liquidPresence`
- In brackets the socket selected for the sensor. In this case `SOCKET_4`

You can find a complete example code for reading the Liquid Detection sensor in the following link:

<http://www.libelium.com/development/waspmote/examples/ev-v30-05-water-point>

## 5.6. Liquid Flow Sensors

### 5.6.1. Specifications

#### Water Flow Small, YF-S402:

**Flow rate:** 0.3 ~ 6 L/Min

**Working voltage:** +5 V ~ +24 V

**Working temperature:** 0 °C ~ 80 °C

**Pipe connection:** 1/8"

**Max rated current:** 15 mA (DC 5 V)



Figure : Image of the YF-S402, Small Flow sensor

#### Water Flow Medium, FS300A:

**Flow rate:** 1 ~ 60 L/Min

**Working voltage:** +5 V ~ +24 V

**Working temperature:** 0 °C ~ 80 °C

**Pipe connection:** 3/4"

**Max rated current:** 15 mA (DC 5 V)



Figure : Image of the FS300A, Medium Flow sensor

## 5.6.2. Measurement Process

The liquid flow sensors output a signal that consists of a series of digital pulses whose frequency is proportional to the flow rate of the liquid through the sensor. That digital signal, whose frequency is in the range between 0 Hz and 100 Hz, is directly read through one of the digital input/output pins of the microcontroller. This sensor must be connected in liquid flow.

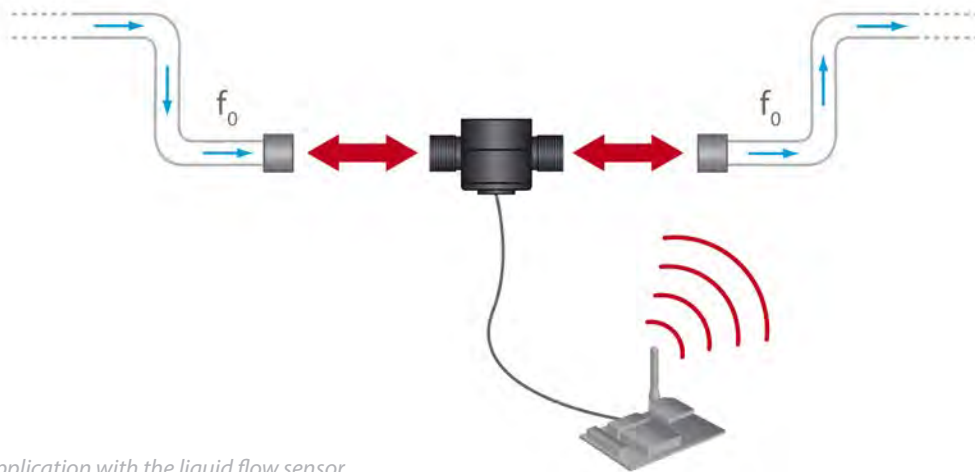


Figure : Example of application with the liquid flow sensor

A code for reading the sensor is shown below:

```
float value;
//Instance objects
flowClass yfs401(SENS_FLOW_YFS401);
{
  Events.ON();

  value = yfs401.flowReading();
}
```

`value` is an integer variable where the water flow value will be stored.

Each sensor needs its own object. So, between the include of the library and the global variables declaration, the object must be created following the next structure:

- Name of the class: In this case `flowClass`
- Name of the object: In this case `yfs401`
- In brackets the type of sensor connected (`SENS_FLOW_FS400`, `SENS_FLOW_FS200`, `SENS_FLOW_FS100`, `SENS_FLOW_YFS401`, `SENS_FLOW_FS300`, or `SENS_FLOW_YFG1`). In this case `SENS_FLOW_YFS401`

Interrupts for flow sensors are developed to wake up Waspote, not for measure directly. It means that Waspote must wake up when the liquid starts to flow or when the flow is low. If the flow rate is high, interruptions will not work properly.

You can find a complete example code for reading the Liquid Level sensor in the following links:

<http://www.libelium.com/development/waspote/examples/ev-v30-07-yf-s401>

<http://www.libelium.com/development/waspote/examples/ev-v30-08-fs300a>

<http://www.libelium.com/development/waspote/examples/ev-v30-09-yfg1>

## 5.7. Water Leakage / Liquid Detection sensor (Line)

### 5.7.1. Specifications

**Length:** 5 meters sensor + 2 meters jumper wire

**Material:** PE + alloy lend

**Weight:** 18 g/meter

**Pull force limit:** 60 kg

**Cable diameter:** 5.5 mm

**Core resistance:** 3 ohm/100 meters

**Maximum exposed temperature:** 75 °C

**Detectable liquids:** Water

**Minimum consumption:** 0  $\mu$ A\*



Figure : Image of the Water Leakage / Liquid Detection sensor (Line)

\*This sensor's consumption is included in the consumption ranges of the connectors on which they can be placed. (See section "Consumption table").

### 5.7.2. Measurement process

This sensor detects conductive liquids anywhere along its length. After it is installed, once the cable senses the leakage of liquids, it will trigger an alarm. The sensor cable can detect the leakage of water. It must be situated in socket 1, socket 2, socket 3, socket 4 and socket 6.

The installation of this sensor should be done in a safe place, far away from high magnetic fields and damp environment. During installation, keep the sensor cable away from sharp material to avoid scuffing the sensor.

Next figure shows an image of how the sensor must be placed to detect the presence of liquid.

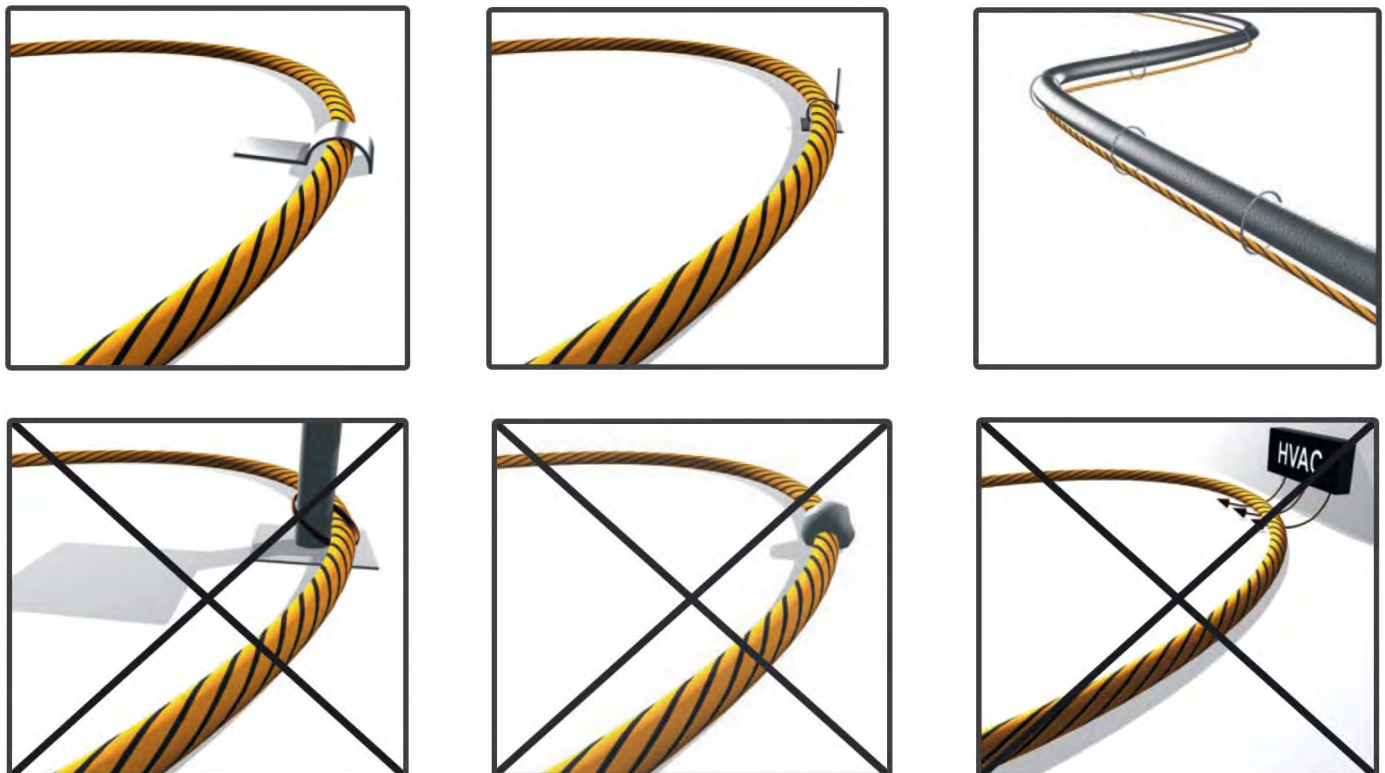


Figure : Application of the Water Leakage / Liquid Detection sensor (Line) for the detection of liquids

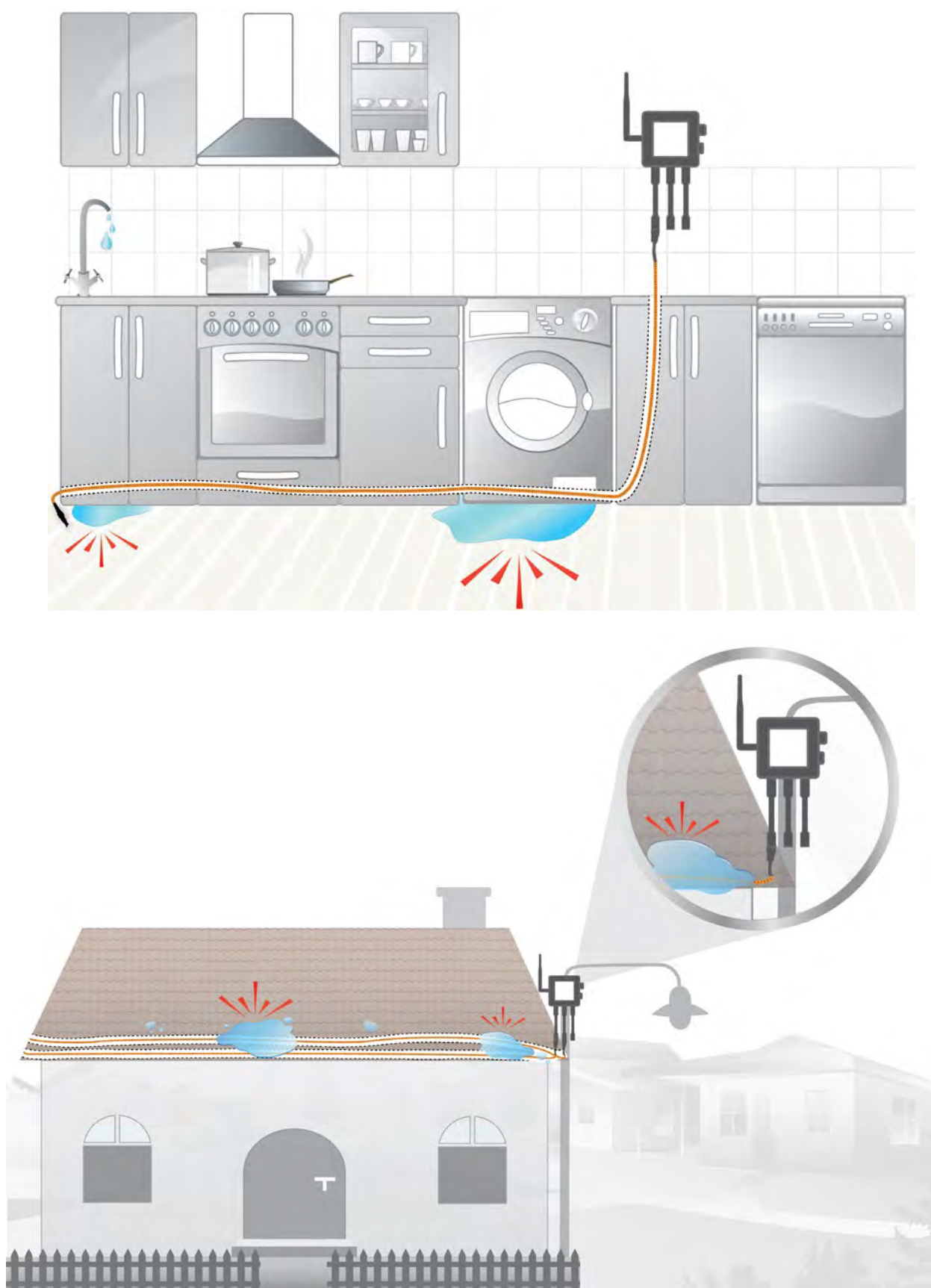


Figure : Some examples of use for the Water Leakage / Liquid Detection sensor (Line)

A code for reading the sensor is shown below:

```
uint8_t value;
//Instance objects
liquidPresenceClass liquidPresence(SOCKET_1);
{
    Events.ON();

    value = liquidPresence.readliquidPresence();
}
```

`value` is an integer variable where the sensor state will be stored.. A high value (3.3 V) or a low value (0 V), which will depend on the liquid level and sensor setup.

Each sensor needs its own object. So, between the include of the library and the global variables declaration, the object must be created following the next structure:

- Name of the class: In this case `liquidPresenceClass`
- Name of the object: In this case `liquidPresence`
- In brackets the socket selected for the sensor. In this case `SOCKET_1`

You can find a complete example code for reading the Liquid Leakage sensor in the following link:

<http://www.libelium.com/development/waspmote/examples/ev-v30-06-water-leakage-line>

## 5.8. Relay Input-Output

### 5.8.1. Specifications

**Contact Ratings VDC:** 1 A, 24 V DC

**Contact Form:** SPDT (1c)

**Coil Rated Current:** 50 mA



Figure : Image of the relay in Waspote Events Sensor Board

### 5.8.2. Precautions for safe use

- **Do not use this feature** if you do not have advanced **knowledge of electricity and electrical automation**.
- **The incorrect use** of this feature can cause **harm** to the **user** or other people and **damage** any connected equipment.
- **The incorrect use** of this feature can cause **death** to the **user** or **other people**!
- **The incorrect use** of this feature can **causes fires**!
- **Use only tools and equipment** with **non-conducting** handles when working on electrical devices.
- **Never handle** this feature when hands, feet, or body are wet or perspiring, or when standing on a wet floor.
- Do not store **highly flammable liquids** near this equipment.
- **Disconnect** the power source before operating on this equipment.
- **Do not touch** the charged relay terminal area while the power is turned on. Doing so may result in **electric shock**.
- Do not use a relay for a load that exceeds the relay's switching capacity or other contact ratings. Doing so will reduce the specified performance, causing insulation failure, contact welding, and contact failure, and the relay itself may be damaged or burnt.
- Make sure the number of switching operations is within the permissible range. If a Relay is used after performance has deteriorated, it may result in insulation failure between circuits and burning of the relay itself.
- **Do not use** s where flammable **gases or explosive gases** may be present. Doing so may cause combustion or explosion due to relay heating or arcing during switching.
- This **Limited Warranty** does **not cover**: (a) defects or damage resulting from accident, misuse, abnormal use, abnormal conditions, improper storage, exposure to liquid, moisture, dampness, sand or dirt, neglect, or unusual physical, electrical or electromechanical stress, defects or damage resulting from the use of Product in conjunction or connection with accessories, products, or ancillary/peripheral equipment.

### 5.8.3. Introduction

The relay inside the Events Sensor Board provides a **potential free contact**. This contact can be used to enable **low power loads** such as relays and contactors, or to enable inputs in a PLC. The Socket 6 (IN REL) is designed to be used by a **potential free contact** to detect on or off conditions, for example in power failure applications.

The relay output on Waspote Events Sensor Board allows controlling small external loads or other relays through the relay contacts.

Its important to remark that the relay Input-output is **not designed** for alternate current (**V AC**), therefore please use **only** continuous currents (**V DC**).

**NOTE:** the changeover contact is designed to be an auxiliary contact, **NEVER TO HANDLE LOADS**. Please never reach the current limitations defined in the relay specifications. The Events board can be damaged permanently. The input contact is designed to be used with a relay contact (the input must go from 0 V to 3.3 V, in Socket 6 / IN REL). If you have any question about the usage of the relay, please contact Libelium before any test.

### 5.8.4. Relay input example

In this example it is used the potential free contact “normally close – nc” and external relay, whose coil is permanently connected to an external power source. When the external power supply is cut off, the contact returns to its “normally closed” resting position, and the Events Sensor Board knows that the external power source has fallen.

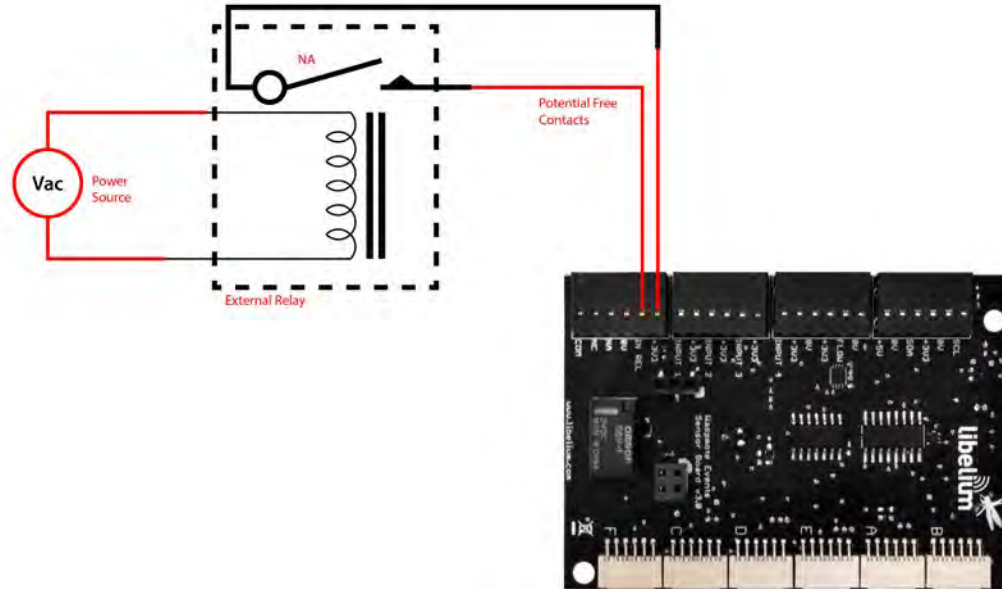


Figure : Typical application of the relay input in Wasp mote Events Sensor Board

A code for input relay is shown below:

```
int value;
//Instance object
relayClass relay;
{
  Events.ON();
  delay(10);
  // Read the sensor level
  value = relay.readInRel();
}
```

`value` is an integer variable where the sensor state (a high value (3.3 V) or a low value (0 V)) will be stored.

Each sensor needs its own object. So, between the include of the library and the global variables declaration, the object must be created following the next structure:

- Name of the class: In this case `relayClass`
- Name of the object: In this case `relay`

You can find a complete example code for use the relay input in the following link:

<http://www.libelium.com/development/waspmote/examples/ev-v30-10-ext-in>

### 5.8.5. Relay output example

In this example, the commuted contact of the relay is used to interact with the external contactor coil. Then a common small load, such as a light bulb, can be controlled through the “normally open – NO” or the the “normally closed – NC” contacts.

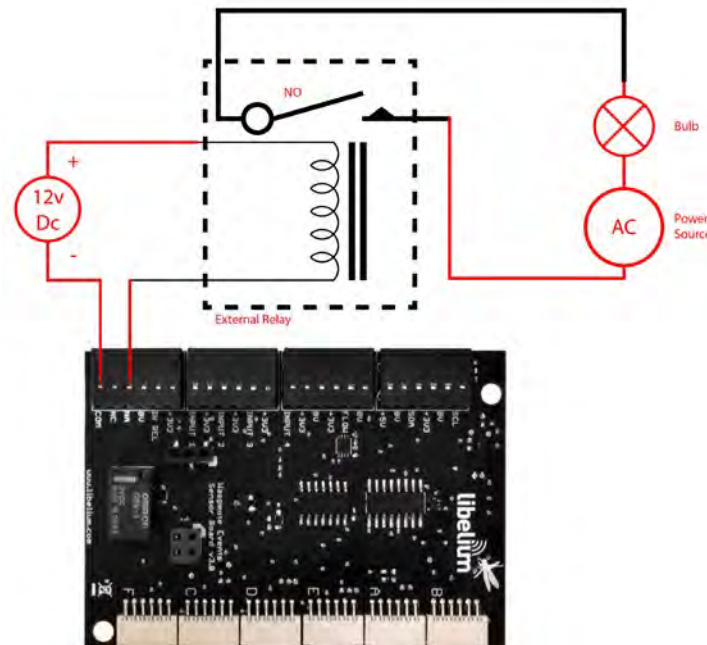


Figure : Typical application of the relay output in Wasp mote Events Sensor Board

A code for the Relay output is shown below:

```
int value;
//Instance objects
relayClass relay;
liquidLevelClass ll(SOCKET_2);
{
  Events.ON();
  delay(10);
  if(value == 1){
    //Switch ON the relay OUTPUT
    relay.relayON();
  }else{
    //Switch OFF the relay OUTPUT
    relay.relayOFF();
  }
  // Read the sensor level
  value = ll.readliquidLevel();
}
```

Each sensor needs its own object. So, between the include of the library and the global variables declaration, the object must be created following the next structure:

- Name of the class: In this case `relayClass`
- Name of the object: In this case `relay`

You can find a complete example code for use the relay output in the following link:

<http://www.libelium.com/development/waspote/examples/ev-v30-11-relay-output>

**Note:** The recommended pin for this application is Socket 6 (IN REL), but actually Socket 1, Socket 2, Socket 3 and Socket 4 also support this operation.

## 5.9. Relay Input-Output in Waspote Plug & Sense!

To provide access to the relay contacts in the Waspote Plug & Sense! encapsulated line, a waterproof terminal block junction box is provided as a Relay Input-Output probe, making the connections on industrial environments or outdoor applications easier. In addition, access to the socket 6 input is also provided.



Figure : Relay Input-Output - Terminal box probe

It consists of 2 cable glands and 6 terminal block connectors with screw. The junction box can be easily opened by removing the four external screws and the cover. Then, the user is able to make the necessary connections using the terminal block connectors. Finally, the cable glands should be adjusted and the junction box should be closed properly to avoid water ingress.

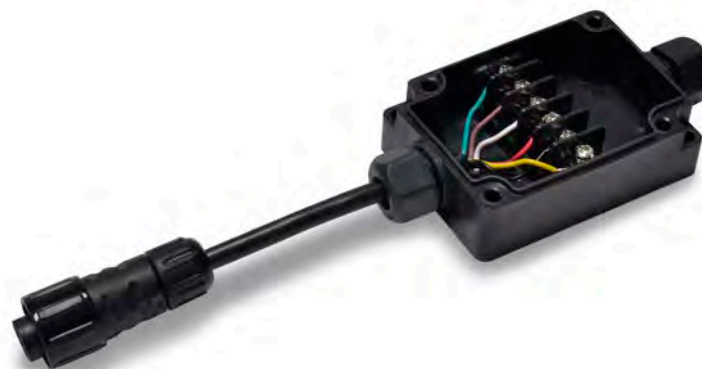


Figure : Pin-out of the relay Input-Output junction box.

Terminal	Signal
1	Common
2	NC
3	NA
4	3v3
5	Socket 6 (IN REL)
6	GND

*Note: Please double check the terminal block connections to avoid wrong wirings or short circuits between poles. The Waspote Plug & Sense! Unit can be seriously damaged. Besides, ensure that the junction box is properly closed to avoid damaged in outdoor applications. Libelium warranty will not cover damages caused by a wrong installation.*

## 5.10. Ultrasound sensor (MaxSonar® from MaxBotix™)

### 5.10.1. Specifications

#### I2CXL-MaxSonar®-MB7040™

**Operation frequency:** 42 kHz

**Maximum detection distance:** 765 cm

**Interface:** Digital Bus

**Power supply:** 3.3 V ~ 5 V

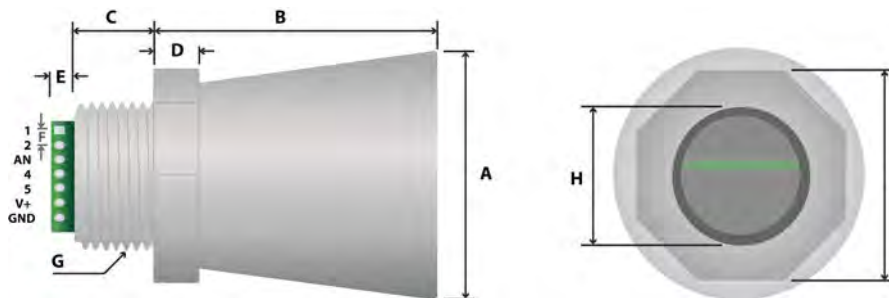
**Consumption (average):** 2.1 mA (powered at 3.3 V) – 3.2 mA (powered at 5 V)

**Consumption (peak):** 50 mA (powered at 3.3 V) – 100 mA (powered at 5 V)

**Usage:** Indoors and outdoors (IP-67)



Figure : Ultrasonic I2CXL-MaxSonar®-MB7040 from MaxBotix™ sensor



A	1.72" dia.	43.8 mm dia.
B	2.00"	50.7 mm
C	0.58"	14.4 mm
D	0.31"	7.9 mm
E	0.18"	4.6 mm
F	0.1"	2.54 mm
G	3/4" National Pipe Thread Straight	
H	1.032" dia.	26.2 dia.
I	1.37"	34.8 mm
weight: 1.76 oz. ; 50 grams		

Figure : Ultrasonic I2CXL-MaxSonar®-MB7040 sensor dimensions

In the figure below we can see a diagram of the detection range of the sensor developed using different detection patterns (a 0.63 cm diameter dowel for diagram A, a 2.54 cm diameter dowel for diagram B, an 8.25 cm diameter rod for diagram C and a 28 cm wide board for diagram D):

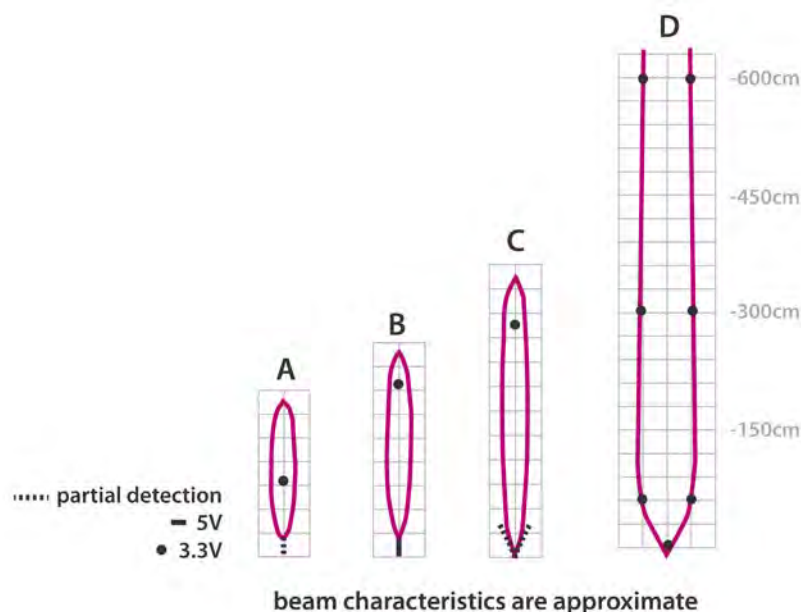


Figure : Diagram of the sensor beam extracted from the data sheet of the XL-MaxSonar®-WRA1™ sensor from MaxBotix

### I2CXL-MaxSonar®-MB1202™:

- **Operation frequency:** 42 kHz
- **Maximum detection distance:** 645 cm
- **Sensitivity (analog output):** 2.5 mV/cm (powered at 3.3 V) – 3.8 mV/cm (powered at 5 V)
- **Power supply:** 3.3 ~ 5 V
- **Consumption (average):** 2 mA (powered at 3.3 V) – 3 mA (powered at 5 V)
- **Usage:** Indoors



Figure : Ultrasonic I2CXL-MaxSonar®-MB1202 from MaxBotix™ Sensor

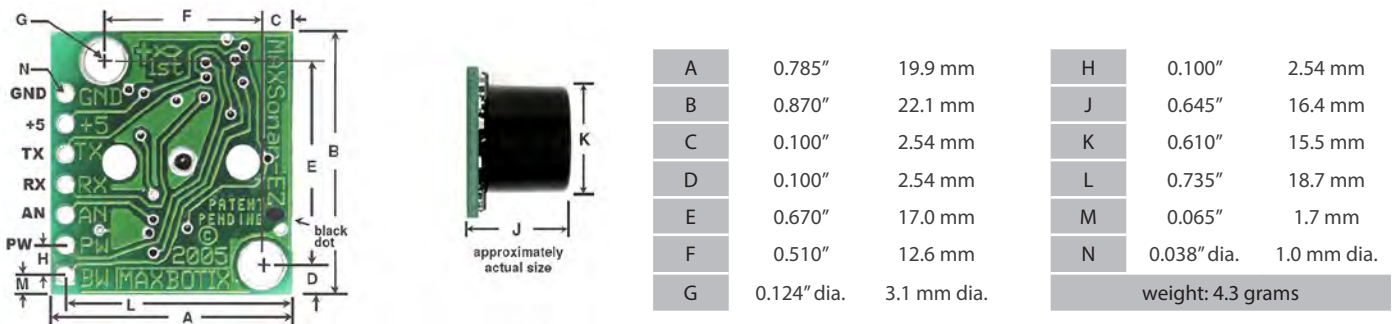


Figure : Ultrasonic I2CXL-MaxSonar®-MB1202 Sensor dimensions

In the figure below we can see a diagram of the detection range of the sensor developed using different detection patterns (a 0.63 cm diameter dowel for diagram A, a 2.54 cm diameter dowel for diagram B, an 8.25 cm diameter rod for diagram C and a 28 cm wide board for diagram D):

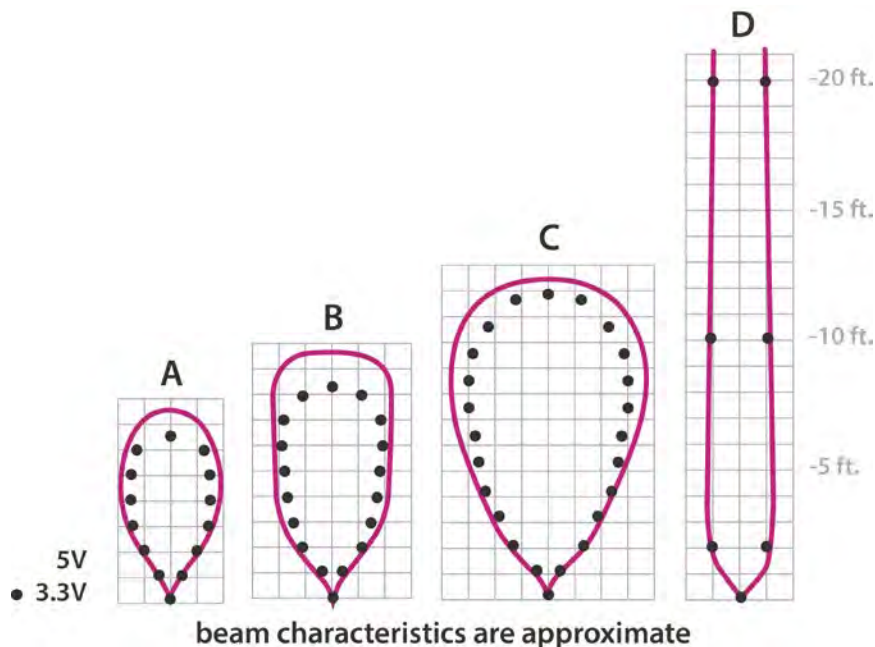


Figure : Diagram of the sensor beam extracted from the data sheet of the Ultrasonic I2CXL-MaxSonar®-MB1202 sensor from MaxBotix

## 5.10.2. Measurement process

The MaxSonar® sensors from MaxBotix can be connected through the digital bus interface.

In the next figure, we can see a drawing of two example applications for the ultrasonic sensors, such as liquid level monitoring or presence detection.

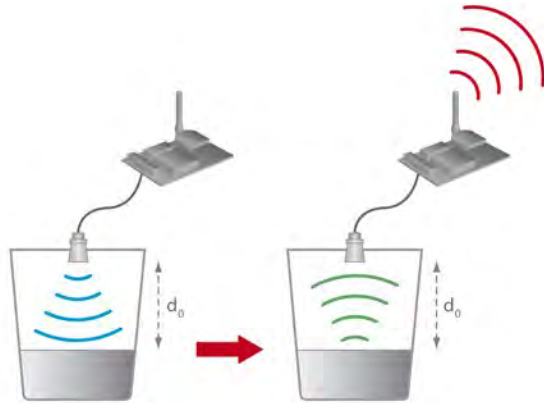


Figure : Example of application for the MaxSonar® sensors

The XL-MaxSonar®-WRA1TM sensor is endowed with an IP-67 casing, so it can be used in outdoors applications, such as liquid level monitoring in storage tanks.

Below a sample code to measure one of the ultrasound sensors (the XL-MaxSonar®-WRA1) is shown:

Reading code:

```
{
  uint16_t distance;
  Events.ON();
  distance = Events.getDistance();
}
```

You can find a complete example code for reading the distance in the following link:

<http://www.libelium.com/development/waspmote/examples/ev-v30-13-ultrasound-sensor/>

## 5.10.3. Socket

These sensors are connected in the socket highlighted in the figure below.

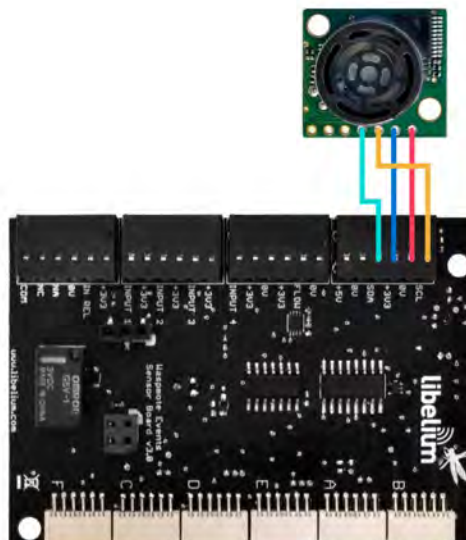


Figure : Image of the socket for connecting the MaxSonar® Sensors

## 5.11. Luminosity sensor (Luxes accuracy)

### 5.11.1. Specifications

#### Electrical characteristics

**Dynamic range:** 0.1 to 40000 lux

**Spectral range:** 300 ~ 1100 nm

**Voltage range:** 2.7 ~ 3.6 V

**Supply current typical:** 0.24 mA

**Sleep current maximum:** 0.3  $\mu$ A

**Operating temperature:** -30 ~ +70 °C



Figure : Image of the Luminosity sensor

### 5.11.2. Measurement process

This is a light-to-digital converter that transforms light intensity into a digital signal output. This device combines one broadband photo-diode (visible plus infrared) and one infrared-responding photo-diode on a single CMOS integrated circuit capable of providing a near-photopic response over an effective 20-bit dynamic range (16-bit resolution). Two integrating ADCs convert the photo-diode currents to a digital output that represents the irradiance measured on each channel. This digital output in lux is derived using an empirical formula to approximate the human eye response.

Reading code:

```
{
  uint16_t luxes = 0;
  // Reads the luxes sensor
  Events.ON();
  //Select the option of measure:
  // - OUTDOOR
  // - INDOOR
  luxes = Events.getLuxes(INDOOR);
  //Return luxes from luxes sensor
}
```

You can find a complete example code for reading the luminosity in the following link:

<http://www.libelium.com/development/waspmote/examples/ev-v30-14-luxes-sensor/>

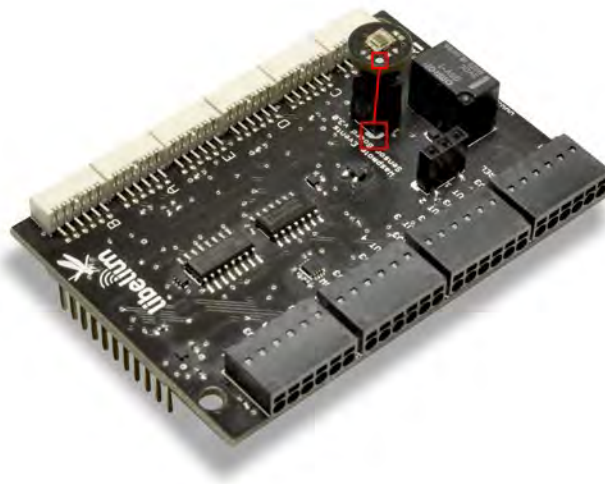


Figure : Image of the socket for the luxes sensor

In the image above we can see highlighted the four pins of the terminal block where the sensor must be connected to the board. The white dot on the luxes board, must match the mark of the Events Sensor Board.

## 5.12. Design and connections

The Waspote Events Sensor Board has been designed with the aim of facilitating the integration of the previously mentioned sensors and others of similar characteristics. Whenever a different sensor from those indicated in this manual is added, make sure it complies with the electrical specifications that appear in the Waspote manual. In the following sections the different board connectors are described so the connectivity and sensor integration can be fully taken advantage of.

### 5.12.1. Digital bus

In this socket you can connect any type of Digital Bus compatible devices, powered at 3.3 V. It also has a power pin to 5 V.

### 5.12.2. Socket 1, Socket 2, Socket 3, Socket 4 and Socket 6

In this socket any TTL digital sensor type (on – off, 0 V – 3.3 V range) can be connected.

### 5.12.3. Liquid flow

In this socket the Liquid Flow Sensor can be connected.

### 5.12.4. Relay Input-Output

In this socket, the contacts of the relay are available. The user can connect an external potential free contact to interact with the Waspote Events Sensor Board and also an external load can be controlled, always meeting the technical specifications of this component described in this guide.

### 5.12.5. Sockets for casing

In case the Events board is going to be used in an application that requires the use of a casing, such as an outdoors application, a series of sockets to facilitate the connection of the sensors through a probe has been disposed.

These sockets (PTSM from Phoenix Contact) allow to assemble the wires of the probe simply by pressing them into it. To remove the wire press the slot above the input pin and pull off the wire softly.

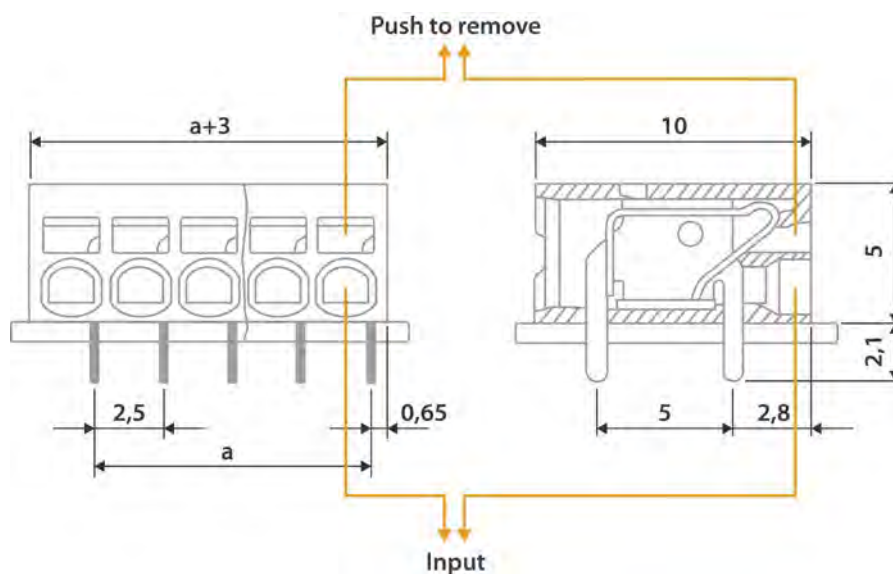


Figure : Diagram of a socket extracted from the Phoenix Contact data sheet

In the figure below an image of the board with the sockets in it and the correspondence between its inputs and the sensor's pins is shown.

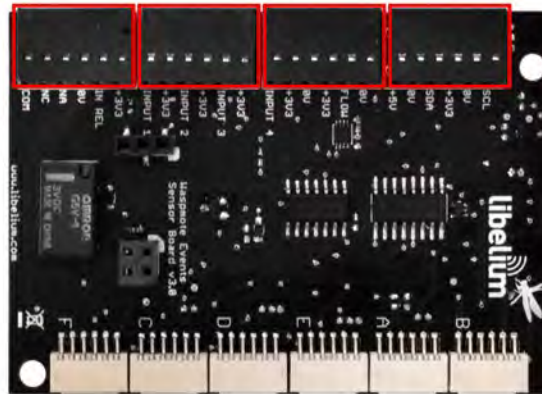


Figure : Sockets for casing applications

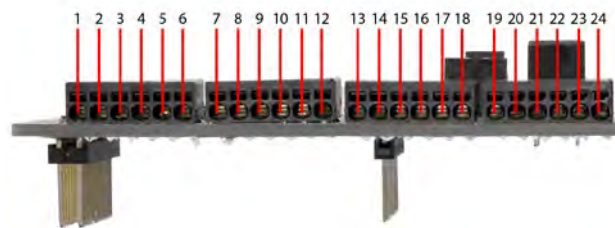


Figure : Pin correspondence between the sockets and the sensors

Socket	Pin	Function
Digital Bus	1	SCL
	2	GND
	3	VCC (3.3 V)
	4	SDA
	5	GND
	6	VCC (5 V)
Liquid Flow	7	GND
	8	Liquid Flow Sensor
	9	VCC (3.3 V)
Auxiliary ground pin	10	GND
4	11	VCC (3.3 V)
	12	Socket 4
3	13	VCC (3.3 V)
	14	Socket 3
2	15	VCC (3.3 V)
	16	Socket 2
1	17	VCC (3.3 V)
	18	Socket 1
6	19	VCC (3.3 V)
	20	Socket 6 (Relay Input)
Auxiliary ground pin	21	GND
Output Relay	22	Normally Open Relay Contact
	23	Normally Closed Relay Contact
	24	Relay contact common

## 5.13. Library

The Events Sensor Board for Waspote has its own library which contains the set of necessary instructions to easily configure and read each one of the sensors which can be connected to the board. Next, each one of the functions is described and the process of configuration detailed for each sensor. The specific configuration which must be applied to each one of the sensors is explained in the specific sensor's section.

When using the Events Sensor Board on Waspote, remember it is mandatory to include the WaspSensorEvent\_v30 library by introducing the next line at the beginning of the code:

```
#include <WaspSensorEvent_v30.h>
```

Each Events sensor needs its own object. So, between the include of the library and the global variables declaration, the object must be created following the next structure:

- Name of the class: `hallSensorClass`.
- Name of the object: We recommend to use the name of the sensor. In this case `hall`.
- In brackets the socket selected for the sensor. In this case `SOCKET_1`.

### 5.13.1. Events class

Methods:

Turn on the sensor board by enabling the 3.3 V and 5 V.

```
Events.ON();
```

Turn off the sensor board by disabling the 3.3 V and 5 V.

```
Events.OFF();
```

Read the distance in cm when the ultrasound sensor is connected.

```
Events.getDistance();
```

Read the temperature in °C when the BME280 is connected.

```
Events.getTemperature();
```

Read the pressure in Pa when the BME280 is connected.

```
Events.getPressure();
```

Read the humidity in % when the BME280 is connected:

```
Events.getHumidity()
```

Read luxes in lux when the TSL2561 is connected. In this case, its necessary to include the place where the sensor is being used (INDOOR or OUTDOOR) into the brackets.

```
Events.getLuxes(INDOOR);
```

The `attachInt` function, implemented as such in the code, enables interruptions generated by the board sensors, allowing the microprocessor to recognise and process them as such.

```
Events.attachInt()
```

Complementing the previous function, the aim of `detachInt` is to disable the interruptions if the microprocessor is not required to react in the event of a change in one of the sensors. After its implementation the mote will ignore any interruption which arrives from the sensors until the `attachInt` instruction is activated again.

```
Events.detachInt()
```

The instruction `loadInt` is used to read the content of the shift register and store its output in an method of each sensor class call `getInt()`, in which the sensor which has caused the interruption and the other sensors that were activated when it happened appear. Once all the registers have been read, they restart from zero, not loading again until a new interruption triggers. To recognize if a sensor has produced an interruption, it is sufficient to carry out a logic comparison between the `getInt()` method of each sensor and true condition.

### 5.13.2. hallSensorClass

Method:

It is necessary to specify the socket where the hall sensor is connected and instance the object.

```
hallSensorClass hall(SOCKET_A);
```

Read the hall sensor value in Events board.

```
hall.readHallSensor();
```

Return interruption value of hall sensor.

```
hall.getInt();
```

### 5.13.3. liquidLevelClass

Method:

It is necessary to specify the socket where the liquid level is connected and instance the object.

```
LiquidLevelClass ll(SOCKET_A);
```

Read the liquid level sensor value in Events board.

```
ll.readliquidLevel();
```

Return interruption value of liquid level sensor.

```
ll.getInt();
```

### 5.13.4. liquidPresenceClass

Method:

It is necessary to specify the socket where the liquid presence is connected and instance the object.

```
liquidPresenceClass lp(SOCKET_A);
```

Read the liquid presence sensor value in Events board.

```
lp.readliquidPresence();
```

Return interruption value of liquid presence sensor.

```
lp.getInt();
```

### 5.13.5. pirSensorClass

Method:

It is necessary to specify the socket where the PIR is connected and instance the object.

```
pirSensorClass pir(SOCKET_A);
```

Read the PIR sensor value in Events board.

```
pir.readPirSensor();
```

Return interruption value of PIR sensor.

```
pir.getInt();
```

### 5.13.6. flowClass

Method:

It is necessary to specify the flow sensor model and instance the object.

```
flowSensorClass flow(SENS_FLOW_YFS401);
```

Read the flow sensor value in Events board.

```
flow.flowReading();
```

Return interruption value of flow sensor.

```
flow.getInt();
```

### 5.13.7. relayClass

Method:

In this case, it is necessary to instance the object.

```
relayClass relay();
```

Read the Socket 6 value in Events board.

```
relay.readInRel();
```

Return interruption value of InRel value.

```
relay.getInt();
```

Switch on the relay.

```
relay.relayON();
```

Switch off the relay.

```
relay.relayOFF();
```

A basic program to detect events from the board will present the following structure:

1. The board is switched on using the function `Events.ON`
2. Enable interruptions from the board using the function `Events.attachInt`
3. Put the mote to sleep with the functions `PWR.sleep` or `PWR.deepSleep`
4. When the mote wakes up, disable interruptions from the board using function `Events.detachInt`
5. Load the value stored in the shift register with function `Events.loadInt`
6. Process the interruption
7. Return to step 3 to enable interruptions and put the mote to sleep

In case the data acquired from the board is to be sent through any of the wireless communication modules it is highly recommended to use the Wasp mote Frame format. You can find more information about how to handle the Wasp mote Frame in the Programming Guide in the Development Section of the Libelium website.

The files of the Events Sensor Board itself are: **WaspSensorEvent\_v30.cpp**, **WaspSensorEvent\_v30.h**

They can be downloaded from: [http://www.libelium.com/development/waspmote/sdk\\_and\\_applications](http://www.libelium.com/development/waspmote/sdk_and_applications)

## 6. API changelog

Keep track of the software changes on this link:

[www.libelium.com/development/waspmote/documentation/changelog/#Events](http://www.libelium.com/development/waspmote/documentation/changelog/#Events)

## 7. Documentation changelog

### From v7.1 to v7.2

- Deleted reference to the discontinued YF-G1 sensor
- Updated info about the liquid flow sensors
- 4G US module LE910 NAG was discontinued, now Libelium offers the LE910 NA V2 (different bands, no 2G and no GPS)
- Added information about the Bridge
- Added references to new LoRaWAN ASIA-PAC / LATAM module
- Added references to new LoRaWAN AU module
- Added references to new LoRaWAN IN module
- Added references to new Sigfox AU / APAC / LATAM module
- Deleted references to the discontinued Internal Solar Panel
- Deleted references to the discontinued RS-232 module
- Added references for the new GPS accessory for Plug & Sense!

### From v7.0 to v7.1:

- Added references to the integration of Industrial Protocols for Plug & Sense!

## 8. Consumption

### 8.1. Power control

The power of the Events board is drawn from the Waspote 3.3 V line and is controlled by the solid state switch which is accessed through the 2x11 connector, defined as SENS\_3V3\_PW in the application. From this switch it is therefore possible to completely activate or deactivate the Events board power (see Waspote manual for more details about the switches to manage the mote's power and the appropriate section in the WaspotePWR API manual's library for more information on its handling). This means that, independently of the board's internal mechanisms, Waspote will decide at any moment if the board is on or off.

Inside the board itself, the OR logic gate, the shift register remain powered at all times without possible disconnection by the user provided that the board is active. However, the consumption of these components is only up to 50  $\mu$ A.

**Note:** Each sensor has its own power consumption and it also depends on the sensor state, so it would have to be taken into account to estimate the overall power consumption.

### 8.2. Low Consumption Mode

The Waspote Events board has been designed to have the least consumption possible. For this, the only recommendations which the user must try to follow are the following:

- **Use the Waspote low consumption modes**

One of the main advantages of Waspote, its efficient energy management, is especially used in this board, which allows the mote to be placed in one of the low consumption modes at the expense of the appearance of an event triggering an interruption which wakes it in order to respond to the event in the way it has been programmed and returning to the low consumption mode (consult the Waspote manual and the WaspotePWR library for more information on the mote's energy management).

- **Care of new sensors**

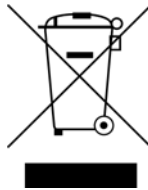
Libelium has chosen a series of sensors which are outstanding thanks to their high functionality and low consumption in active mode. Make sure that the sensors that are installed in this board maintain minimum consumption in this status, since they are thought to be always operating and wake Waspote when any event occurs.

## 9. Maintenance

- In this section, the term “Waspote” encompasses both the Waspote device itself as well as its modules and sensor boards.
- Take care with the handling of Waspote, do not drop it, bang it or move it sharply.
- Avoid putting the devices in areas of high temperatures since the electronic components may be damaged.
- The antennas are lightly threaded to the connector; do not force them as this could damage the connectors.
- Do not use any type of paint for the device, which may damage the functioning of the connections and closure mechanisms.

## 10. Disposal and recycling

- In this section, the term “Wasmote” encompasses both the Wasmote device itself as well as its modules and sensor boards.
- When Wasmote reaches the end of its useful life, it must be taken to a recycling point for electronic equipment.
- The equipment has to be disposed on a selective waste collection system, different to that of urban solid waste. Please, dispose it properly.
- Your distributor will inform you about the most appropriate and environmentally friendly waste process for the used product and its packaging.



## 11. Certifications

Libelium offers 2 types of IoT sensor platforms, Waspote OEM and Plug & Sense!:

- **Waspote OEM** is intended to be used for research purposes or as part of a major product so it needs final certification on the client side. More info at: [www.libelium.com/products/waspote](http://www.libelium.com/products/waspote)
- **Plug & Sense!** is the line ready to be used out-of-the-box. It includes market certifications. See below the specific list of regulations passed. More info at: [www.libelium.com/products/plug-sense](http://www.libelium.com/products/plug-sense)

Besides, Meshlium, our multiprotocol router for the IoT, is also certified with the certifications below. Get more info at:

[www.libelium.com/products/meshlium](http://www.libelium.com/products/meshlium)

List of certifications for Plug & Sense! and Meshlium:

- CE (Europe)
- FCC (US)
- IC (Canada)
- ANATEL (Brazil)
- RCM (Australia)
- PTCRB (cellular certification for the US)
- AT&T (cellular certification for the US)



Figure : Certifications of the Plug & Sense! product line

You can find all the certification documents at:

[www.libelium.com/certifications](http://www.libelium.com/certifications)