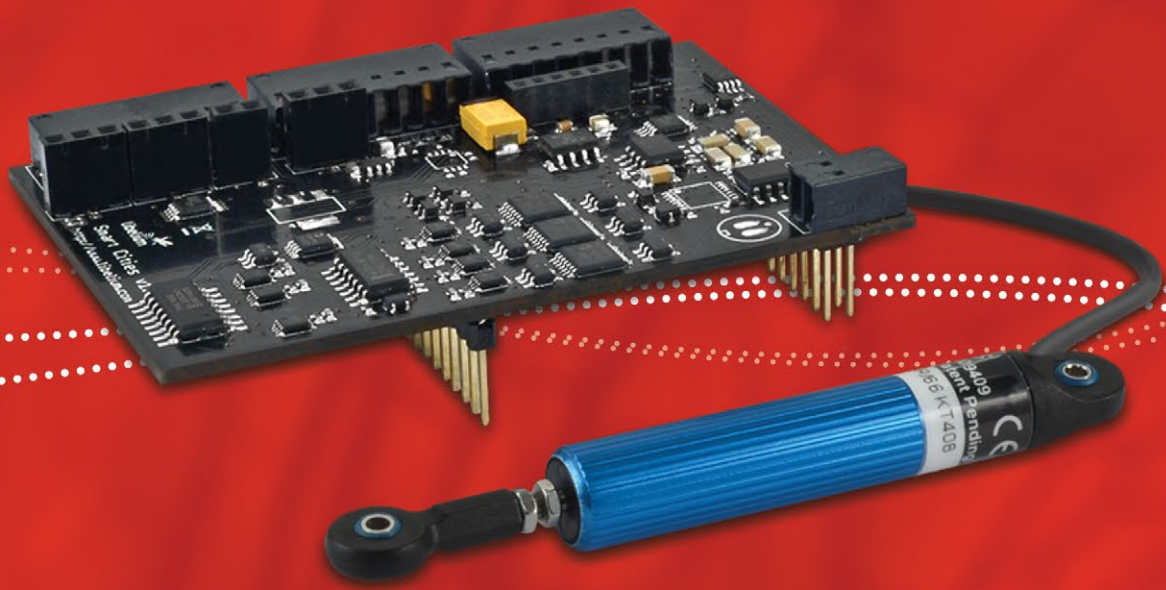


# Smart Cities Technical Guide



Document version: v5.4 - 01/2016  
 © Libelium Comunicaciones Distribuidas S.L.

# INDEX

<b>1. General .....</b>	<b>4</b>
1.1. General and safety information .....	4
1.2. Conditions of use .....	4
<b>2. Waspnote Plug &amp; Sense!.....</b>	<b>5</b>
2.1. Features .....	5
2.2. Sensor Probes.....	5
2.3. Solar Powered .....	6
2.4. Programming the Nodes.....	7
2.5. Radio Interfaces .....	8
2.6. Program in minutes.....	9
2.7. Data to the Cloud.....	9
2.8. Meshlium Storage Options.....	10
2.9. Meshlium Connection Options.....	10
2.10. Models .....	11
2.10.1. Smart Cities .....	12
<b>3. Hardware.....</b>	<b>14</b>
3.1. General Description .....	14
3.2. Specifications .....	14
3.3. Electrical Characteristics.....	14
<b>4. Sensors .....</b>	<b>15</b>
4.1. Particle Sensor – Dust Sensor (GP2Y1010AU0F).....	15
4.1.1. Specifications.....	15
4.1.2. Measurement Process.....	15
4.1.3. Socket .....	17
4.2. Crack detection sensors (Vishay).....	18
4.2.1. Specifications.....	18
4.2.2. Measurement Process.....	19
4.2.3. Socket .....	20
4.3. Linear Displacement Sensor - Crack measurement (SLS095) .....	21
4.3.1. Specifications.....	21
4.3.2. Measurement Process.....	21
4.3.3. Socket .....	23
4.4. Noise Sensor (Microphone POM-2735P-R) .....	24
4.4.1. Specifications.....	24
4.4.2. Measurement Process.....	24
4.4.3. Socket .....	26

4.5. Ultrasonic Sensor (MaxSonar® from MaxBotix™) .....	26
4.5.1. Specifications.....	26
4.5.2. Measurement Process.....	29
4.5.3. Socket.....	30
4.6. Humidity Sensor (808H5V5).....	31
4.6.1. Specifications.....	31
4.6.2. Measurement Process.....	31
4.6.3. Socket.....	32
4.7. Temperature Sensor (MCP9700A) .....	32
4.7.1. Specifications.....	32
4.7.2. Measurement Process.....	32
4.7.3. Socket.....	33
4.8. Luminosity Sensor (LDR) .....	34
4.8.1. Specifications.....	34
4.8.2. Measurement Process.....	34
4.8.3. Socket.....	35
4.9. Sensor interruptions .....	35
4.10. Sockets for casing.....	36
<b>5. Board configuration and programming .....</b>	<b>38</b>
5.1. Hardware configuration .....	38
5.2. API.....	38
<b>6. Consumption .....</b>	<b>43</b>
6.1. Power control .....	43
6.2. Tables of consumption.....	43
6.3. Low consumption mode .....	44
<b>7. API Changelog .....</b>	<b>45</b>
<b>8. Documentation changelog .....</b>	<b>46</b>
<b>9. Maintenance .....</b>	<b>48</b>
<b>10. Disposal and recycling .....</b>	<b>49</b>

# 1. General

## 1.1. General and safety information

- In this section, the term “Wasp mote” encompasses both the Wasp mote 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.
- Wasp mote 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.
- Wasp mote 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 Wasp mote in accordance with the electrical specifications and the environment described in the “Electrical Data” section of this manual.
- Wasp mote 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 IP-65.
- Do not place Wasp mote 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. Wasmote Plug & Sense!

The new Wasmote Plug & Sense! line allows you to easily deploy wireless sensor networks in an easy and scalable way ensuring minimum maintenance costs. The new 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](#).

### 2.1. Features

- Robust waterproof IP65 enclosure
- Add or change a sensor probe in seconds
- Solar powered with internal and external panel options
- Radios available: ZigBee, 802.15.4, WiFi, 868MHz, 900MHz, LoRaWAN, LoRa, Sigfox, 3G/GPRS and Bluetooth Low Energy
- Over the air programming (OTAP) of multiple nodes at once
- Special holders and brackets ready for installation in street lights and building fronts
- Graphical and intuitive programming interface
- External, contactless reset with magnet
- External SIM connector for GPRS or 3G models

### 2.2. 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 Wasmote Plug & Sense!



## 2.3. Solar Powered

Battery can be recharged using the internal or external solar panel options.

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*

For the internal option, the solar panel is embedded on the front of the enclosure, perfect for use where space is a major challenge.



*Figure : Internal solar panel*



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

## 2.4. 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.

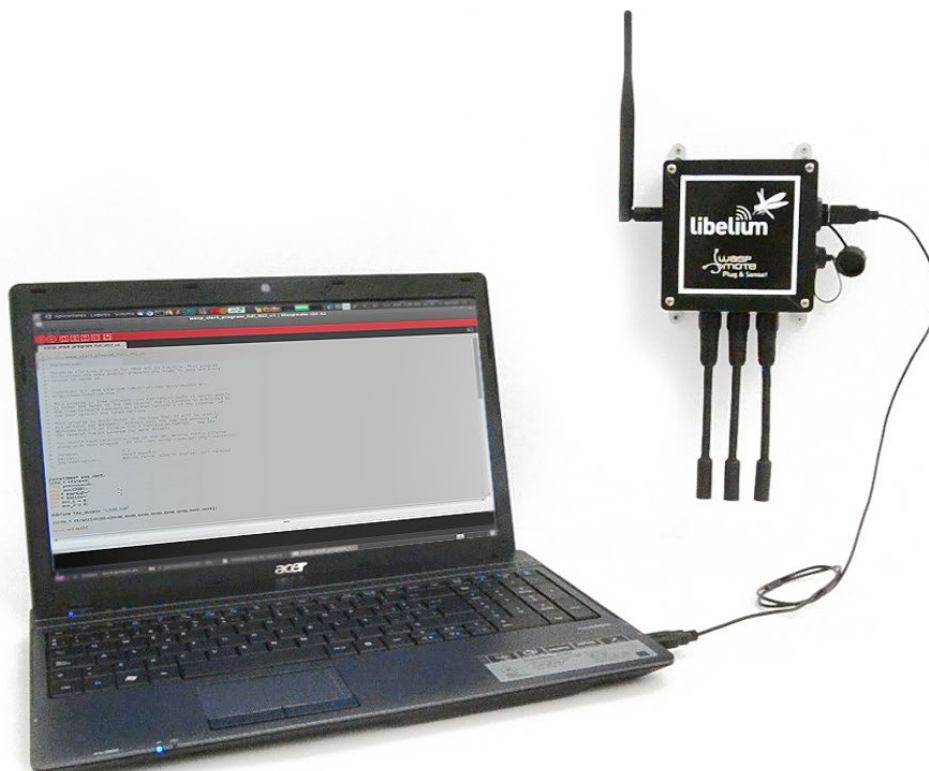


Figure : Programming a node

Over the Air Programming is also possible once the node has been installed. With this technique you can reprogram wirelessly one or more Wasp mote sensor nodes at the same time by using a laptop and the Wasp mote Gateway.

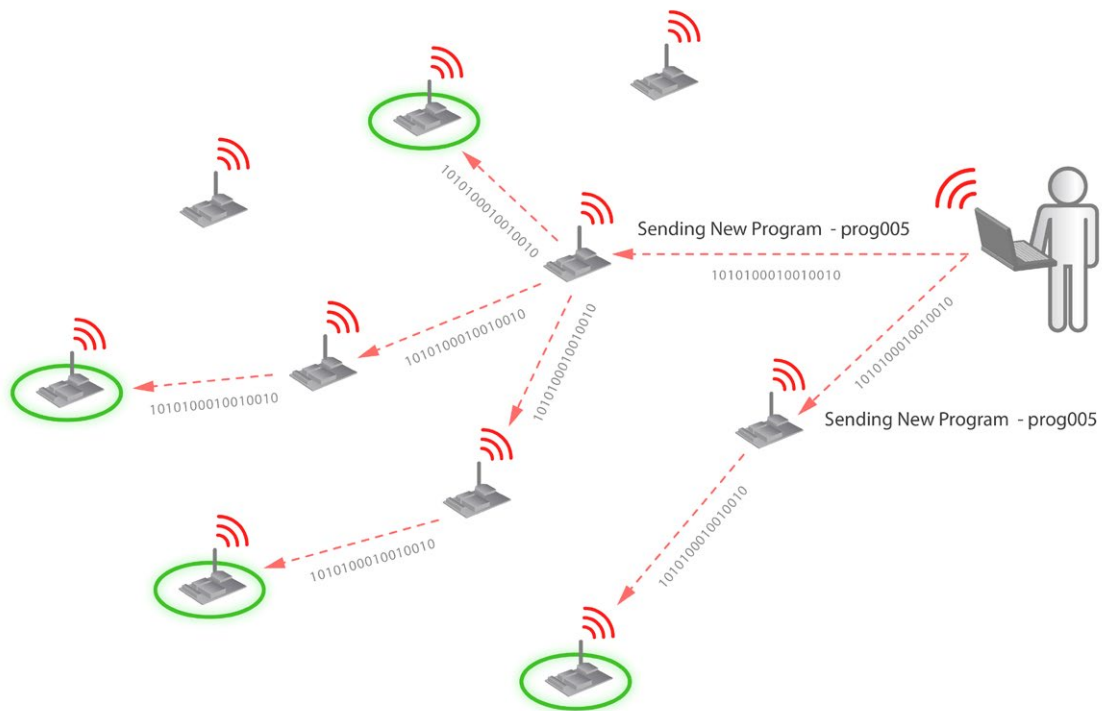


Figure : Typical OTAP process

## 2.5. Radio Interfaces

Model	Protocol	Frequency	txPower	Sensitivity	Range *
XBee-802.15.4-Pro	802.15.4	2.4GHz	100mW	-100dBm	7000m
XBee-ZB-Pro	ZigBee-Pro	2.4GHz	50mW	-102dBm	7000m
XBee-868	RF	868MHz	315mW	-112dBm	12km
XBee-900	RF	900MHz	50mW	-100dBm	10Km
LoRaWAN	LoRaWAN	868 and 433MHz. 900-915MHz version coming in 2016.	14dBm	-136dBm	- km - Typical base station range
LoRa	RF	868 and 900 MHz	14dBm	-137dBm	22km
Sigfox	Sigfox	868MHz	14dBm	-126dBm	- km - Typical base station range
WiFi	802.11b/g	2.4GHz	0dBm - 12dBm	-83dBm	50m-500m
GPRS Pro and GPRS+GPS	-	850MHz/900MHz/1800MHz/1900MHz	2W(Class4) 850MHz/900MHz, 1W(Class1) 1800MHz/1900MHz	-109dBm	- Km - Typical carrier range
3G/GPRS	-	Tri-Band UMTS 2100/1900/900MHz Quad-Band GSM/EDGE, 850/900/1800/1900 MHz	UMTS 900/1900/2100 0,25W GSM 850MHz/900MHz 2W DCS1800MHz/PCS1900MHz 1W	-106dBm	- Km - Typical carrier range
Bluetooth Low Energy	Bluetooth v.4.0 / Bluetooth Smart	2.4GHz	3dBm	-103dBm	100m

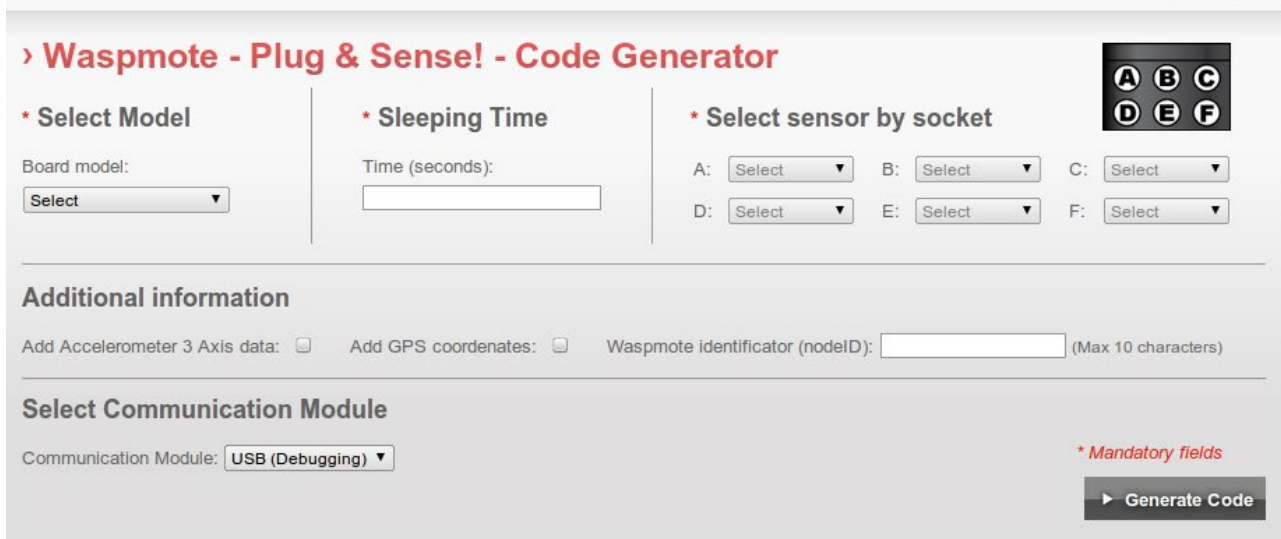
\* Line of sight, Fresnel zone clearance and 5dBi dipole antenna.



## 2.6. Program in minutes

In order to program the nodes an intuitive graphic interface has been developed. Developers just need to fill a web form in order to obtain the complete source code for the sensor nodes. This means the complete program for a specific application can be generated just in minutes. Check the Code Generator to see how easy it is at:

[http://www.libelium.com/development/plug\\_&\\_sense/sdk\\_and\\_applications/code\\_generator](http://www.libelium.com/development/plug_&_sense/sdk_and_applications/code_generator)



The screenshot shows a web form titled "Waspote - Plug & Sense! - Code Generator". It is divided into several sections:

- Select Model:** A dropdown menu labeled "Board model:" with "Select" as the current option.
- Sleeping Time:** A text input field labeled "Time (seconds):".
- Select sensor by socket:** Six dropdown menus labeled A, B, C, D, E, and F, each with "Select" as the current option. Above these is a small keypad icon with letters A-F.
- Additional information:** Three checkboxes: "Add Accelerometer 3 Axis data:", "Add GPS coordinates:", and "Waspote identifier (nodeID):" followed by a text input field (Max 10 characters).
- Select Communication Module:** A dropdown menu labeled "Communication Module:" with "USB (Debugging)" as the current option.
- Buttons:** A red asterisk icon and the text "\* Mandatory fields" are on the right. A dark grey button labeled "Generate Code" is at the bottom right.

Figure: Code Generator

## 2.7. Data to the Cloud

The Sensor data gathered by the Waspote Plug & Sense! nodes is sent to the Cloud by [Meshlium](#), the Gateway router specially designed to connect Waspote sensor networks to the Internet via Ethernet, WifF and 3G interfaces.

Thanks to Meshlium's new feature, the Sensor Parser, now it is easier to receive any frame, parse it and store the data into a local or external Data Base.



Figure: Meshlium

## 2.8. Meshlium Storage Options

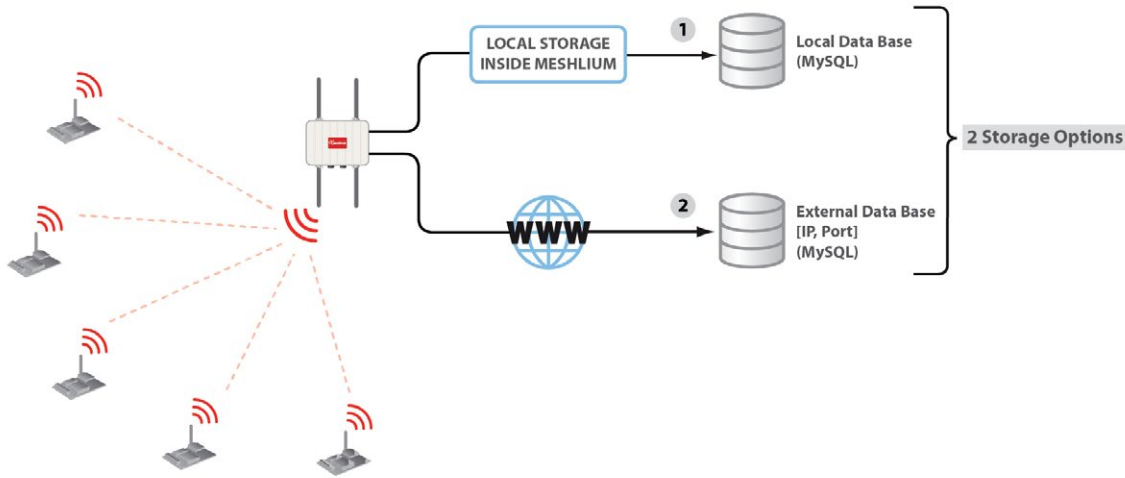


Figure : Meshlium Storage Options

- Local Data Base
- External Data Base

## 2.9. Meshlium Connection Options

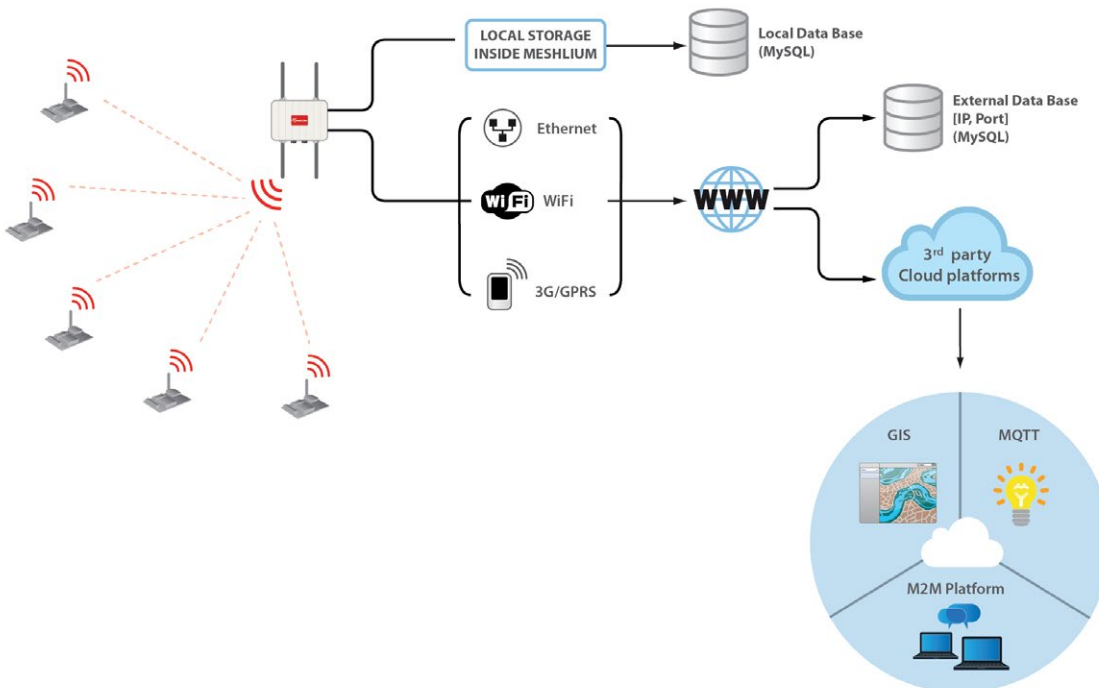


Figure : Meshlium Connection Options

- XBee / LoRa / GPRS / 3G / WiFi → Ethernet
- XBee / LoRa / GPRS / 3G / WiFi → WiFi
- XBee / LoRa / GPRS / 3G / WiFi → 3G/GPRS

## 2.10. Models

There are some defined configurations of Wasmote Plug & Sense! depending on which sensors are going to be used. Wasmote 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 to connect 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 Wasmote. 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 connected probes in the right socket, otherwise they can be damaged.

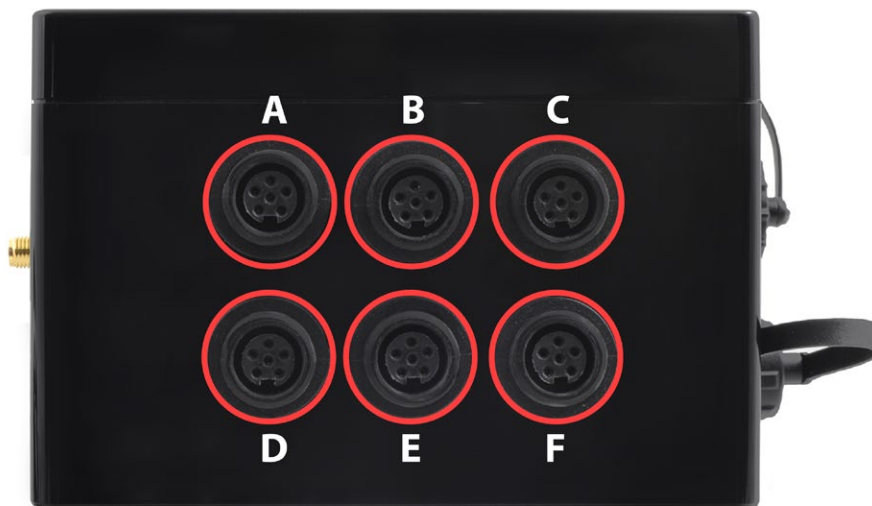


Figure : Identification of sensor sockets

### 2.10.1. Smart Cities

The main applications for this Waspote Plug & Sense! model are noise maps (monitor in real time the acoustic levels in the streets of a city), air quality, waste management, structural health, smart lighting, etc. Refer to [Libelium website](#) for more information.



Figure : Smart Cities Waspote Plug & Sense! model



Sensor sockets are configured as shown in the figure below.

Sensor Socket	Sensor probes allowed for each sensor socket	
	Parameter	Reference
A	Temperature	9203
	Soil temperature	86949*
	Ultrasound (distance measurement)	9246
B	Humidity	9204
	Ultrasound (distance measurement)	9246
C	Luminosity (LDR)	9205
D	Noise sensor	9259
F	Linear displacement	9319

Figure : Sensor sockets configuration for Smart Cities model

\* Ask Libelium [Sales Department](#) for more information.

As we see in the figure below, thanks to the directionable probe, the ultrasound sensor probe may be placed in different positions. The sensor can be focused directly to the point we want to measure.



Figure : Configurations of the ultrasound sensor probe

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

### 3. Hardware

#### 3.1. General Description

The purpose of the Waspote Smart Cities Board is to extend the monitoring functionalities from indoor environments to outdoor locations. Apart from the humidity, luminosity and temperature sensors, present in all the Libelium boards, other two sensors for specific applications have been included: three sensors destined to monitor cracks in buildings and structures, a linear displacement sensor (SLS095) for crack width and a single strand strain gage for crack detection. Also a dust and particles sensor (GP2Y1010AU0F) has been introduced, used to measure the concentration of particles in suspension in the environment in air quality control applications, and finally the POM-2735P-R microphone, adapted to measure the environmental noise in the A decibels scale.

#### 3.2. Specifications

**Weight:** 20gr

**Dimensions:** 73.5 x 51 x 1.3 mm

**Temperature Range:** [-20°C, 65°C]

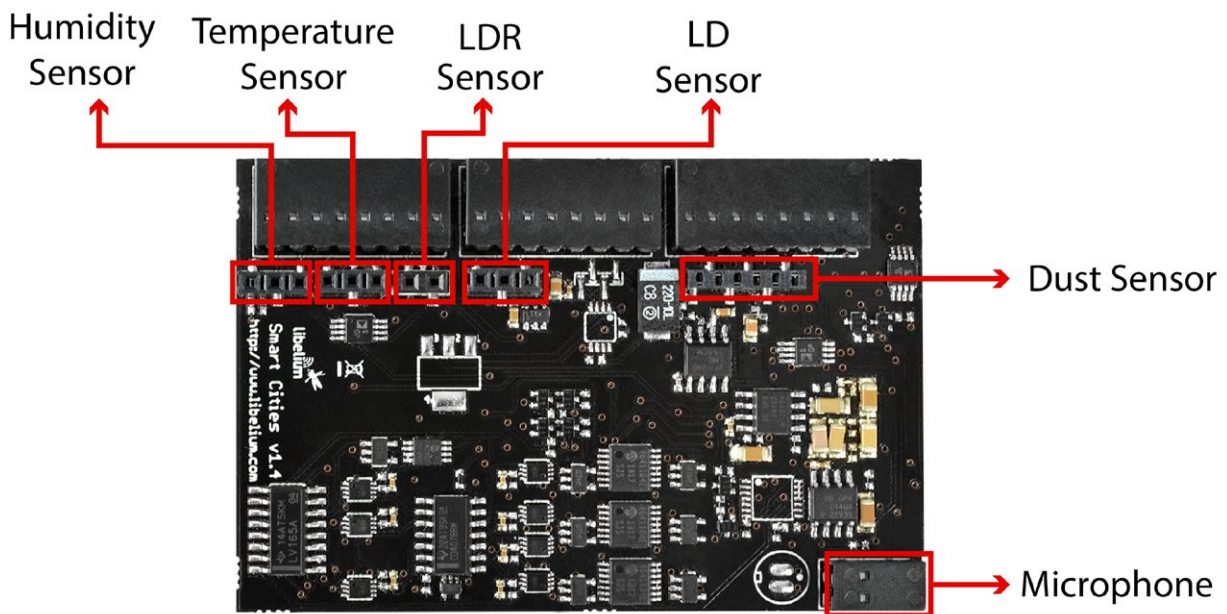


Figure: Upper side

#### 3.3. Electrical Characteristics

**Board Power Voltages:** 3.3V and 5V

**Sensor Power Voltages:** 3.3V and 5V

**Maximum admitted current (continuous):** 200mA

**Maximum admitted current (peak):** 400mA

## 4. Sensors

### 4.1. Particle Sensor – Dust Sensor (GP2Y1010AU0F)

#### 4.1.1. Specifications

**Supply voltage:** -0.3V ~ 7V

**Sensitivity:** Typical: 0.5V/(0.1 mg/m<sup>3</sup>), Minimum: 0.35V/(0.1 mg/m<sup>3</sup>), Maximum: 0.65V/(0.1 mg/m<sup>3</sup>)

**Output voltage at no dust:** Typical: 0.9V, Minimum: 0V, Maximum: 1.5V

**Output voltage range:** 3.4V

**Operation temperature:** -10°C ~ +65°C

**Current consumption:** Typical: 11mA, Maximum: 20m

**LED Pulse Cycle:** 10±1ms

**LED Pulse width:** 0.32±0.02ms

**LED Operating supply voltage:** 5±0.5V



Figure : GP2Y1010AU0F Dust Sensor

#### 4.1.2. Measurement Process

The GPY21010AU0F is an optical sensor whose principle of operation is based on the detection of the infrared light emitted by an ILED diode, reflected by the dust particles and captured by means of a phototransistor. The ILED diode needs to be supplied with a signal of pulses of 0.32ms width and a period of 10ms, generated automatically by the hardware of the board when the sensor is turned on, being the output a signal of pulses of the same time characteristics whose amplitude is proportional to the environmental dust density (see the graph in the figure below). To read this signal has been added a demodulation circuit that extracts the envelope of the train of pulses at whose output results an analog voltage in a range between 0V and 3V approximately that can be read at one of the analog inputs of the mote ([ANALOG1](#)). The supply voltage is controlled through a solid state switch activated with the signal [DIGITAL2](#).

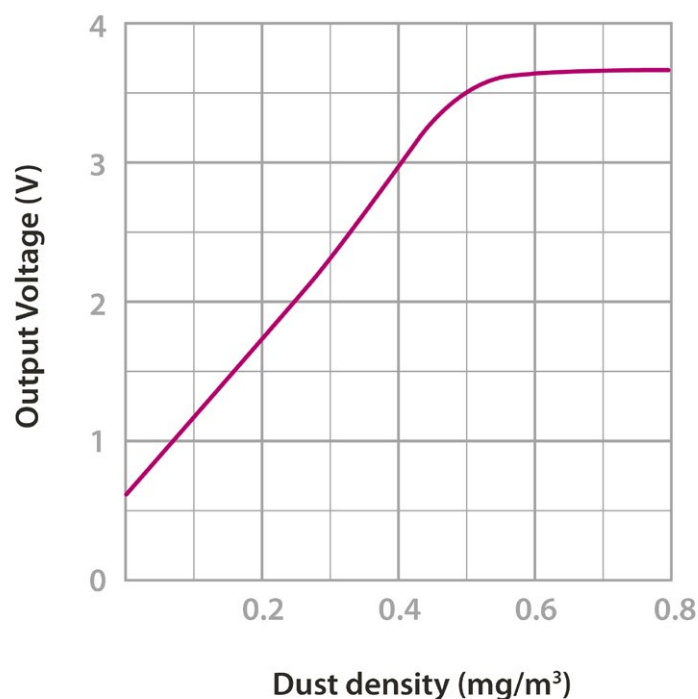


Figure : Graph of the output voltage vs dust density extracted from the Sharp's sensor data sheet



Figure : Example of application for the particle sensor

Below a sample code for reading the output of the sensor and converting the voltage measured into a dust density value using the libraries of the board is shown:

```
{
  SensorCities.ON();
  SensorCities.setSensorMode(SENS_ON, SENS_CITIES_DUST);
  delay(2000);
  float dust_value;
  dust_value = SensorCities.readValue(SENS_CITIES_DUST);
}
```

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

<http://www.libelium.com/development/waspmote/examples/sc-4-dust-sensor-reading>

Dust, dirt or pollen may be accumulated inside the dust sensor structure, especially when the sensor is close to possible solid particle sources: parks, construction works, deserts. That is why it is highly recommended to perform maintenance tasks with this sensors. The little hole where the optics are allocated should be cleaned from time to time; the frequency of this operation depends on the environment conditions or amount of obstructing dust.

The sensor construction makes it necessary to install the sensor protected from the weather conditions. The big Solar Shield is needed for this sensor, in order to avoid oxidation.

**Note:** This sensor is only available for the OEM line, but not for the Plug & Sense! line.

**Note:** The user should also evaluate the Particle Matter (PM1, PM2.5, PM10) – Dust Sensor, available for Plug & Sense! Environment PRO. This is a high-end sensor, calibrated in factory. It can classify the particles in 16 types according to their diameter and performs extreme accuracy, delivering the exact particle concentration in [ $\mu\text{g}/\text{m}^3$ ]. Besides, Libelium offers this sensor in a waterproof enclosure, ready to installed outdoors. All the electronics are protected from rain or dirt. Kindly read the Plug & Sense! Sensors Guide for more information.



### 4.1.3. Socket

In the following figure we can see an image of the socket for the sensor (6 ways, 2mm pitch) with the pin correspondence between them highlighted. In section "Sockets for casing" more information about the corresponding pinout at the sockets for casing applications can be found.

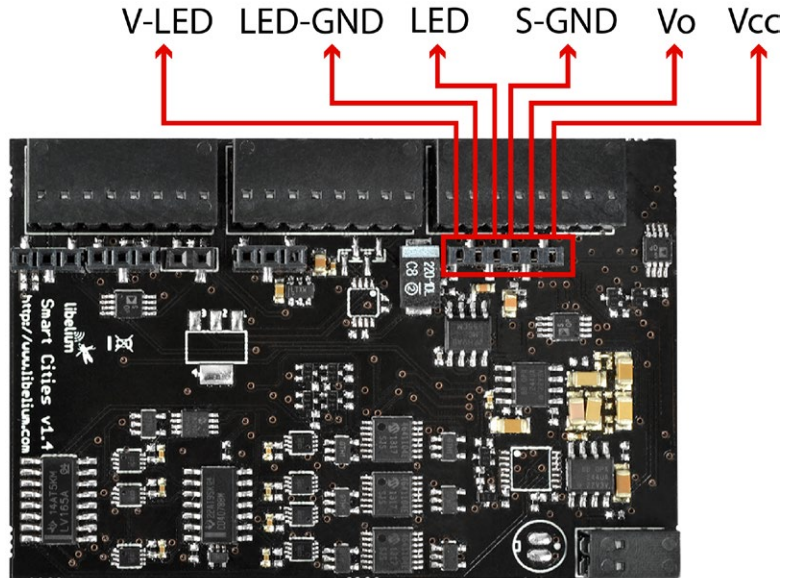


Figure : Image of the socket for the GP2Y1010AU0F sensor

In the figure below we can see the sensor pinout corresponding with the connector shown above. The recommended connector for this sensor is the ZHR-6(P) along with six crimp terminals SZH-002T-P0.5, both of them by JST.

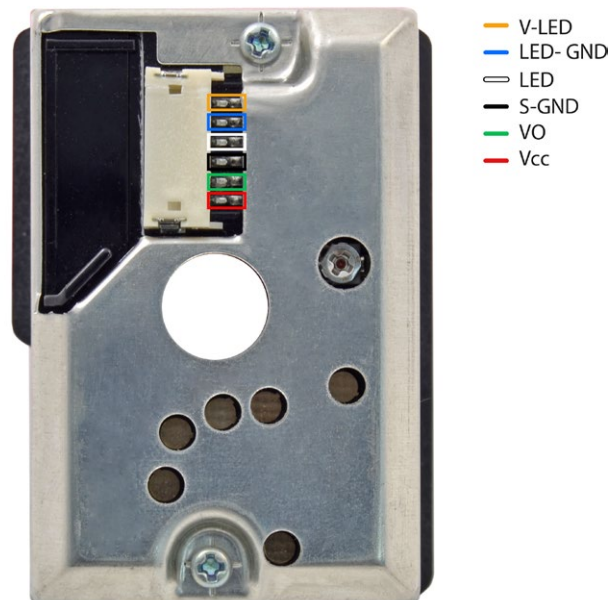


Figure : Image of the GP2Y1010AU0F sensor with the pinout indicated

**Note:** this sensor is not sold ready to use for the OEM line, the user who wills to connect it directly to the Smart Cities Board will have to add the connector necessary to wire it and plug it into its socket.

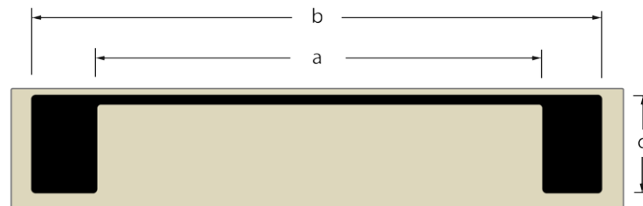
## 4.2. Crack detection sensors (Vishay)

### 4.2.1. Specifications

**Operating temperature:** -195°C~120°C



Figure : Image of the crack detection sensor



GAGE DESIGNATION		DIMENSIONS				
		a	b	c	MATRIX	
					Length	Width
CD-02-10A		0.40	0.56	0.10	0.60	0.13
CD-23-10A		10.2	14.2	2.5	15.2	3.2
CD-02-15A		0.60	0.76	0.10	0.80	0.13
CD-23-15A		15.2	19.3	2.5	20.3	3.2
CD-02-20A		0.80	0.96	0.10	1.00	0.13
CD-23-20A		20.3	24.4	2.5	25.4	3.2
CD-02-25A		1.00	1.16	0.10	1.20	0.13
CD-23-25A		25.4	29.5	2.5	30.5	3.2
CD-02-50A		2.00	2.16	0.10	2.22	0.13
CD-23-50A		50.8	54.9	2.5	56.4	3.2

□ inch

■ millimeter

Figure : Dimensions of the crack detection sensor extracted from the datasheet of the Vishay sensor

## 4.2.2. Measurement Process

The crack detection sensor consists of a small conductive strand with a very low resistance value embedded in a fiber-glass film, when the sensor remains intact it sets a logic 'one' in a digital input of the Wasp mote. In presence of a crack, the sensor shall break, turning to a logic 'zero' in the input pin of the microcontroller (ANALOG5). The sensor must be fixed to the surface using a special adhesive. being recommended the use of a protective coating in long term installations.

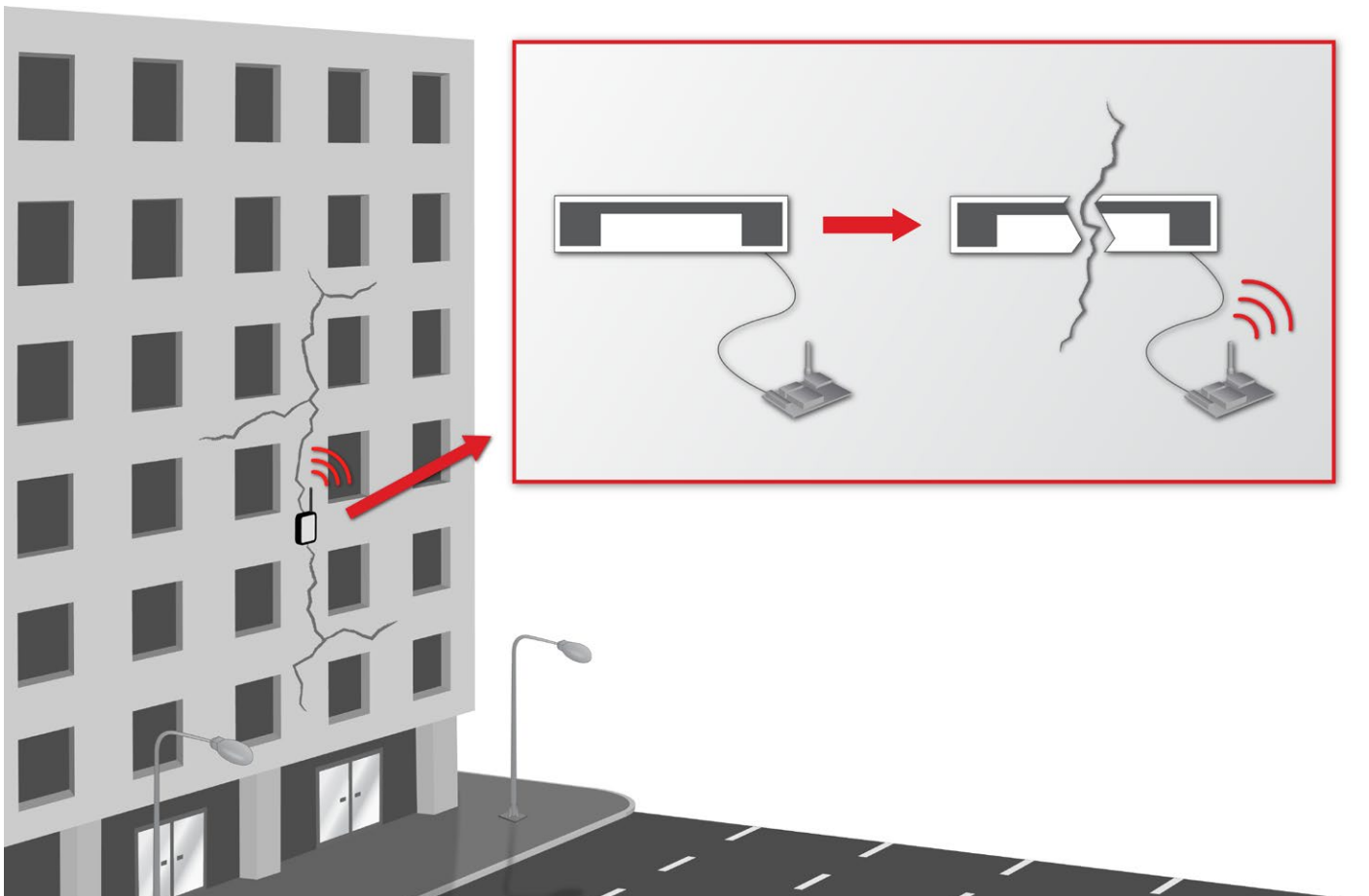


Figure : Example of application for the crack detection sensor

Below a code to measure the sensor is shown:

```
{
  SensorCities.ON();
  SensorCities.setSensorMode(SENS_ON, SENS_CITIES_CD);
  delay(100);
  float crack_value;
  crack_value = SensorCities.readValue(SENS_CITIES_CD);
}
```

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

<http://www.libelium.com/development/waspmote/examples/sc-7-crack-detection-sensor-reading>

### 4.2.3. Socket

This sensor shares the socket with the luminosity sensor (LDR), upon which may be connected independently of the pin position, since no polarity is required. In section "Sockets for casing" more information about the corresponding pinout at the sockets for casing applications can be found.

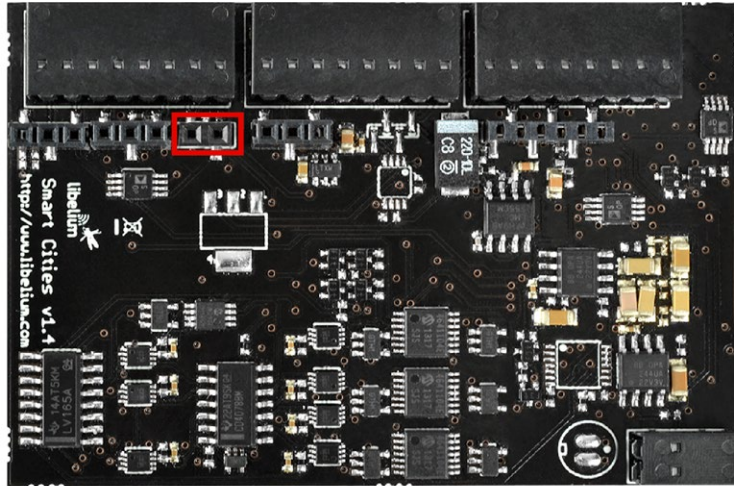


Figure : Image of the socket for the crack detection sensor



## 4.3. Linear Displacement Sensor - Crack measurement (SLS095)

### 4.3.1. Specifications

**Electrical stroke:** 10mm

**Sensor resistance:** 400Ω

**Linearity:** ±0.5%

**Resolution:** 10μm (imposed by the analog-to-digital conversion)

**Supply Voltage:** +8.9V

**Power dissipation (20°C):** 0.2W

**Temperature Operation:** -30°C ~ 100°C



Figure : SLS095 displacement sensor

### 4.3.2. Measurement Process

The SLS095 linear displacement sensor by Penny+Giles is a potentiometer whose wiper moves along with an axis guided by the sensor's body. Fixing both ends of the potentiometer at the sides of the crack we can measure its width by reading the voltage at the wiper. For this, the sensor has been configured as a voltage divider, with one of the ends sourced from a 3V supply, the other end grounded and the wiper connected to the input **ANALOG7** of the analog-to-digital converter of the Waspote, which leads to a resolution of 11μm approximately. The supply voltage comes from a solid state switch controlled by the pin **DIGITAL1**.

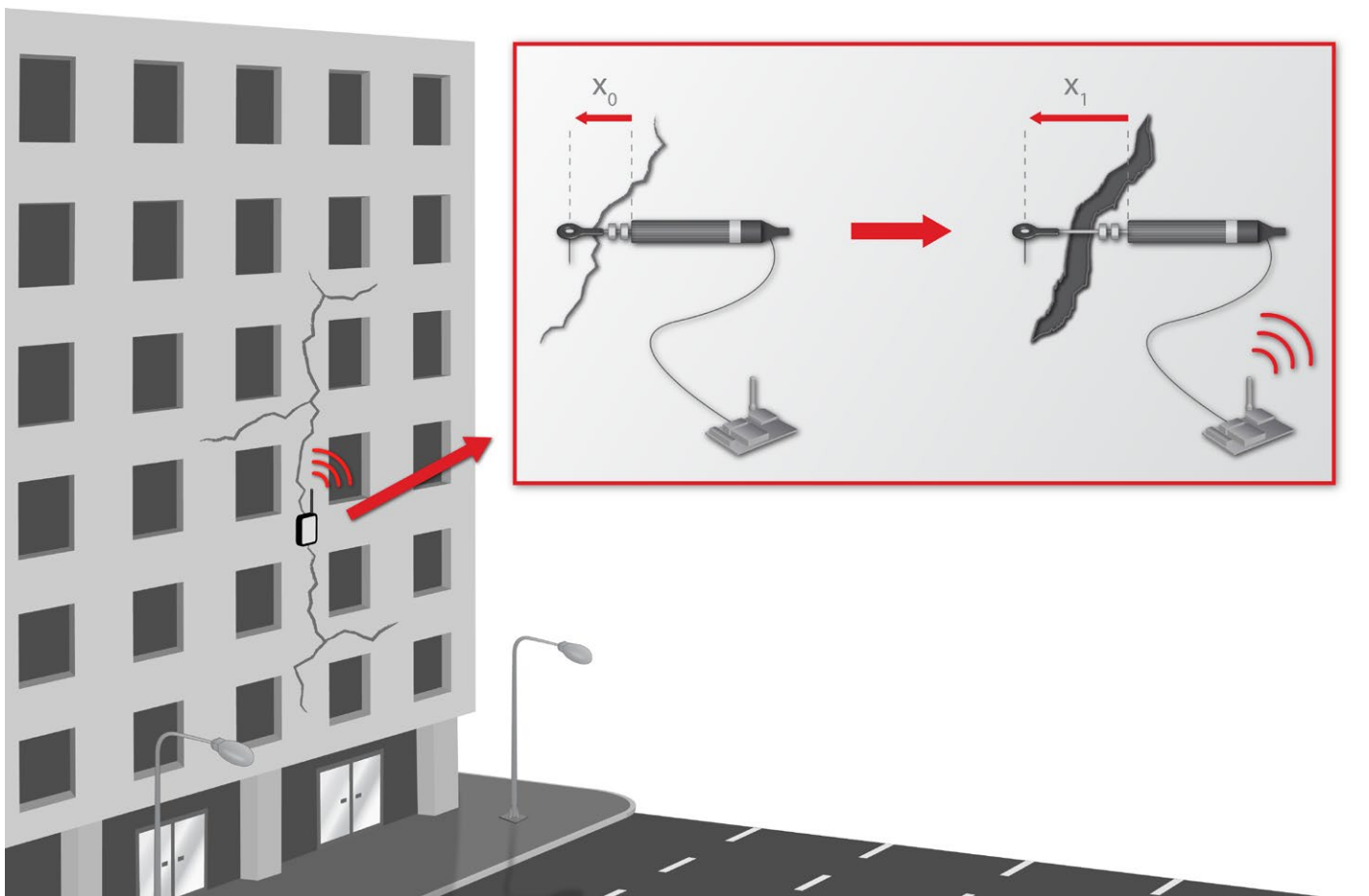


Figure : Example of application for the linear displacement sensor

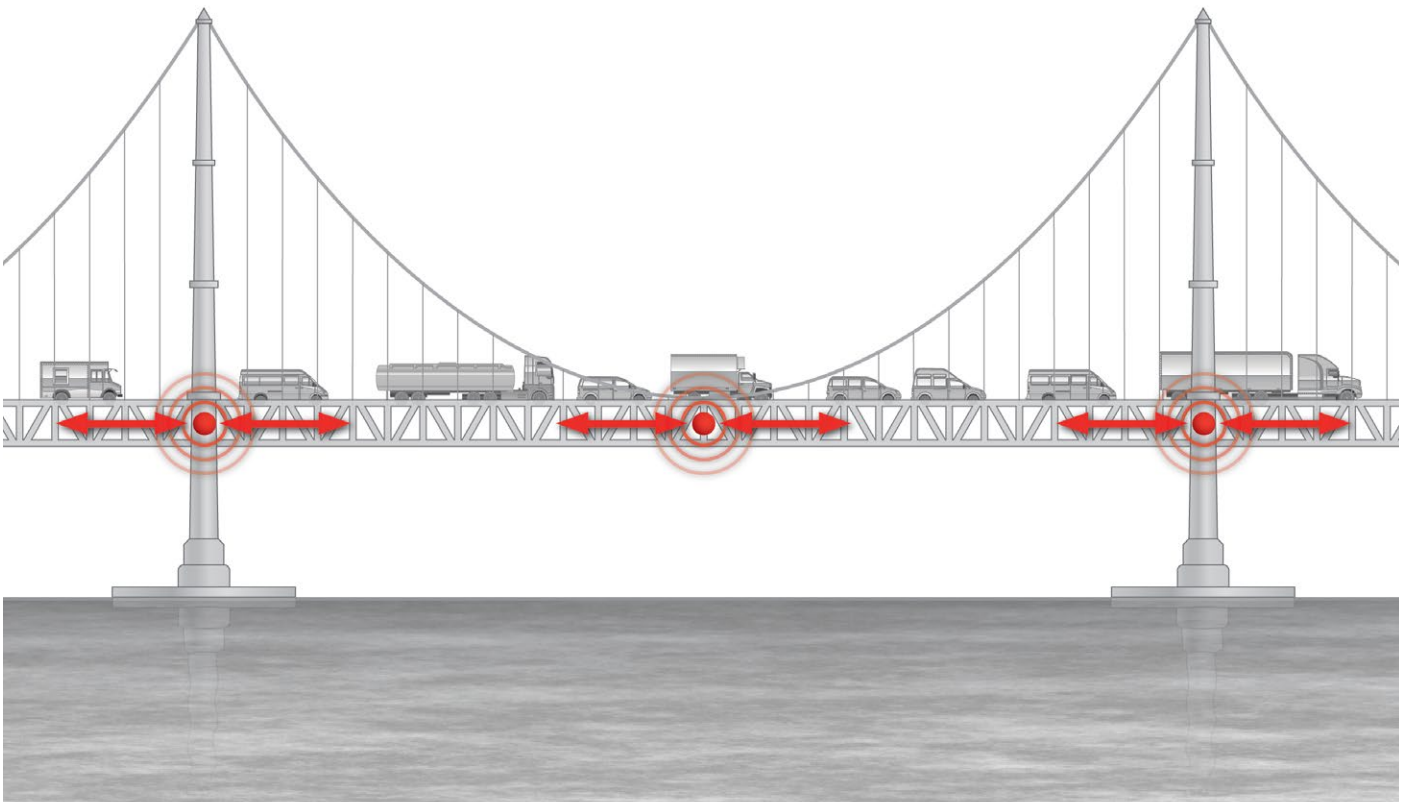


Figure : Example of measurement of expansion and contraction of a bridge for the linear displacement sensor

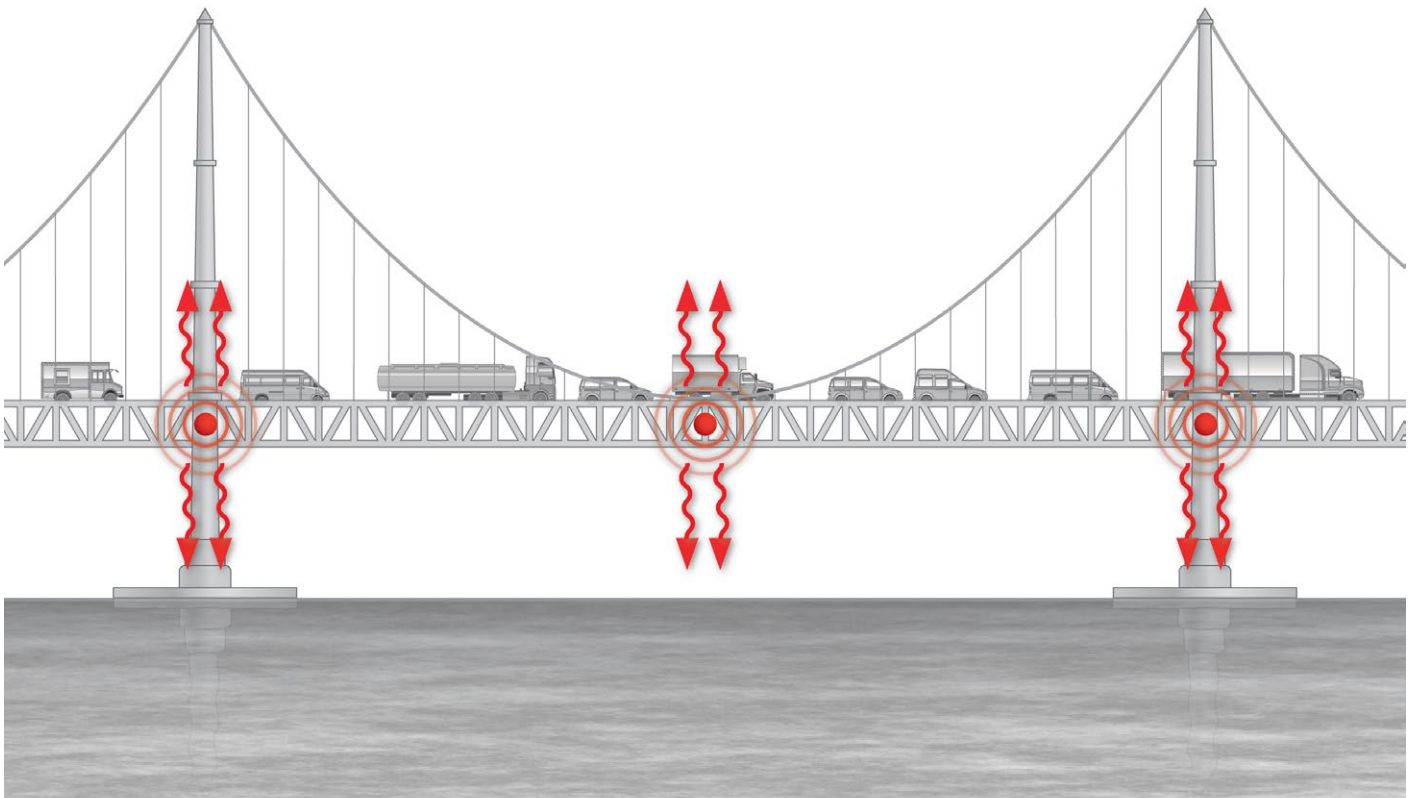


Figure : Example of measurement of vibration in a bridge for the linear displacement sensor. The vibration measurement is complemented by the accelerometer integrated in Wasp mote

Below a sample code for reading the output of the sensor and converting the voltage measured into micrometers using the libraries of the board is shown:

```

{
  SensorCities.ON();
  SensorCities.setSensorMode(SENS_ON, SENS_CITIES_LD);
  delay(100);
  float ld_value;
  ld_value = SensorCities.readValue(SENS_CITIES_LD);
}

```

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

<http://www.libelium.com/development/waspmote/examples/sc-5-crack-sensor-reading>

### 4.3.3. Socket

We can see an image of the socket for the sensor in the next figure (3 ways, 2.54mm pitch) where the correspondence with the pins of the sensor is indicated. The red and the black wires of the SLS095 corresponds with the two ends of the potentiometer (interchangeable with ground or power supply connections), while the yellow wire corresponds with its wiper. In section “Sockets for casing” more information about the corresponding pinout at the sockets for casing applications can be found.

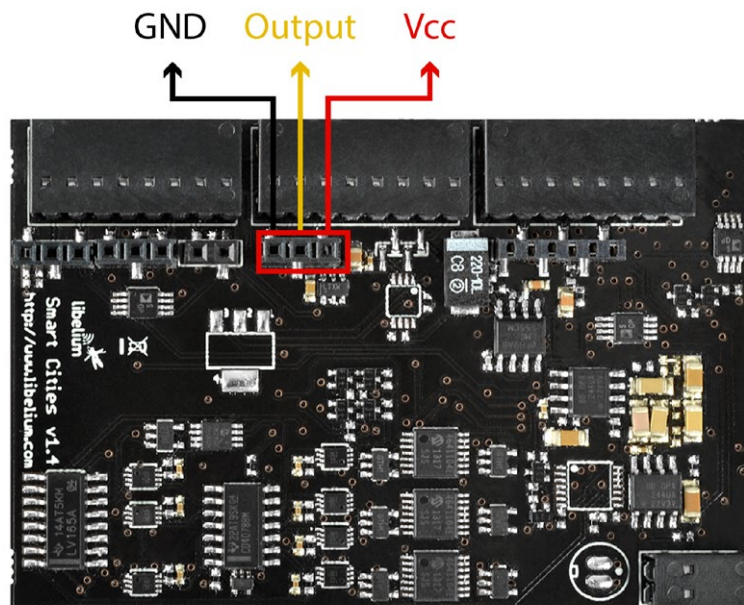


Figure : Image of the socket for the SLS095 sensor

## 4.4. Noise Sensor (Microphone POM-2735P-R)

### 4.4.1. Specifications

**Microphone specifications:**

**Sensitivity:**  $-35 \pm 4$  dB

**Impedance:**  $< 2.2$  k $\Omega$

**Directivity:** Omnidirectional

**Frequency:** 20 Hz ~ 20 kHz

**Supply voltage:** +3V (Standard), +10V (Maximum)

**Maximum current consumption:** 0.5 mA

**Sensitivity reduction:** -3 dB @ 1.5V

**Maximum sound pressure level:**  $114.5 \pm 10$  dB SPL approximately

**S/N ratio:** 60 dB

**Noise Level:** 26 +/- 1 dBA

**Stage Measurement range:** 50 dBA ~ 100 dBA



Figure : Image of the POM-2735P-R microphone by Panasonic

### 4.4.2. Measurement Process

The POM-2735P-R, introduced in the Smart Cities board to monitor the environmental noise, is an omnidirectional microphone with an almost flat response in the whole frequency range of human hearing, between 20 Hz and 20 kHz. A circuit to filter the signal to adapt it to the A decibel scale and output a continuous voltage readable from the mote's processor (at the analog input pin [ANALOG6](#)) has been introduced. When sold along with a microphone, the Smart Cities board is supplied calibrated by Libelium to return an output in the range between 50 dBA and 100 dBA with an accuracy of  $\pm 2.5$  dBA. The calibration data associated to the microphone reading is stored in the microcontroller's EEPROM, between addresses 164 and 185. **Be very careful not to overwrite this memory positions** or it could lead to an irreparable error when reading this sensor.

**Note:** Because of this needed calibration process, the user always must purchase any noise sensor with its corresponding Wasp mote and Smart Cities board.

The A weighting for the audio measurements is a compensation curve that is used to fit the sound pressure measurement to the ear response in function of the frequency, and is the most common standard for noise measurement. Below we can see a table of noise pressure generated by different sources in dBA.

The supply voltage of the microphone and its electronics may be turned on or off with a solid state switch controlled from the mote's processor by the [DIGITAL6](#) output pin.

Sound	dBA
Audition threshold	0
Quiet Room	30
Normal conversation	60~70
Heavy traffic (hearing loss under continued exposure)	90
Pain threshold	130
Jet engine (permanent damage)	140

Figure : Noise in dBA produced by different sources

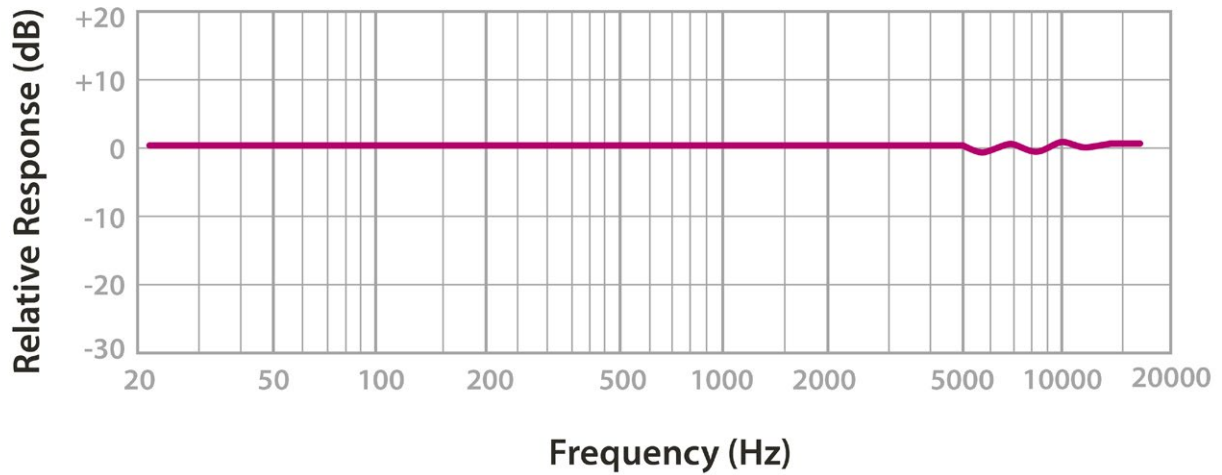


Figure : Graph of the frequency response of the POM-2735P-R extracted from the sensor's data sheet

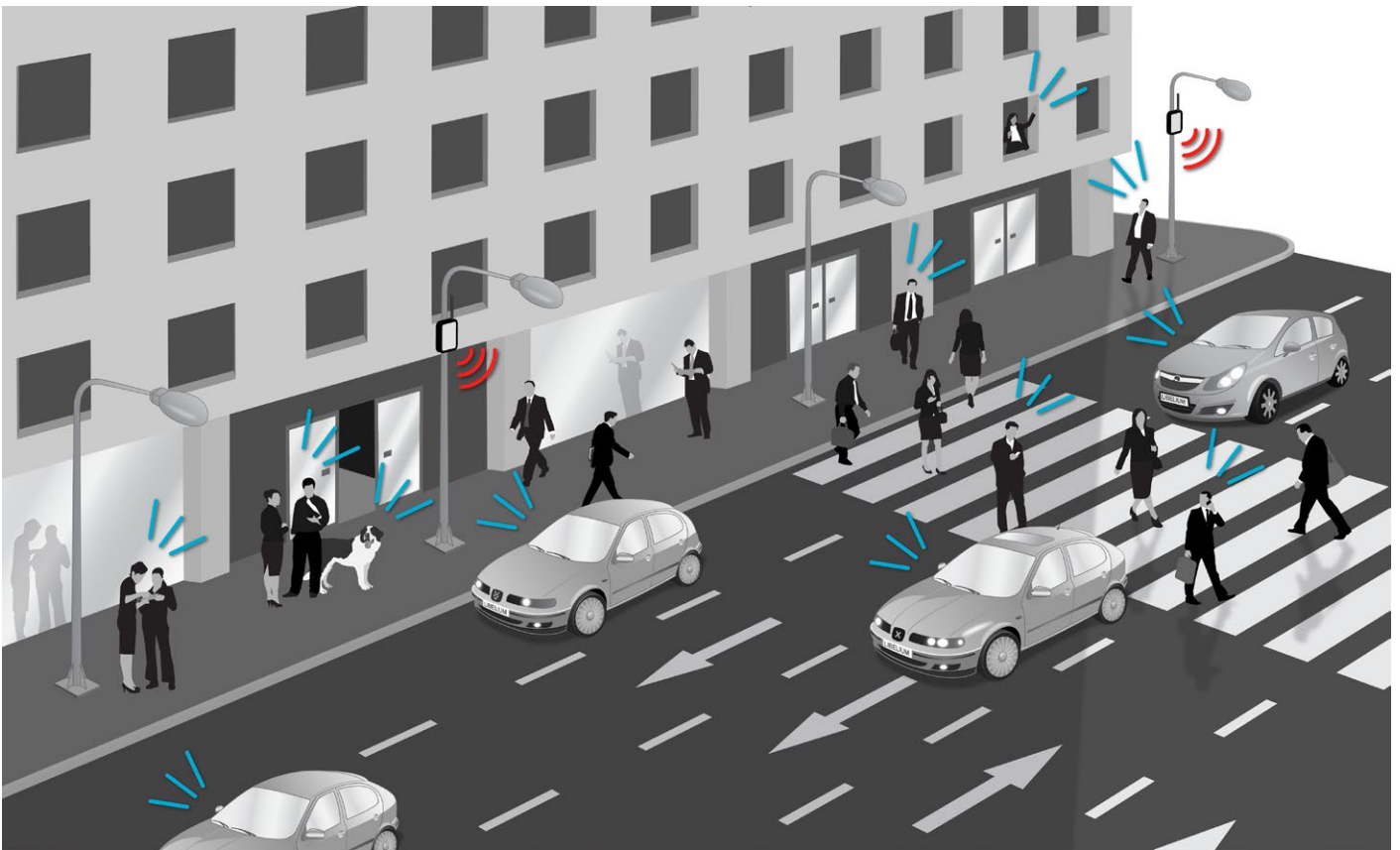


Figure : Example of application for the noise sensor

Below a code to measure the sensor is shown:

```

{
  SensorCities.ON();
  SensorCities.setSensorMode(SENS_ON, SENS_CITIES_AUDIO);
  delay(100);
  float audio_value;
  audio_value = SensorCities.readValue(SENS_CITIES_AUDIO);
}
    
```

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

<http://www.libelium.com/development/waspmote/examples/sc-9-audio-sensor-reading>



### 4.4.3. Socket

The POM-2735P-R microphone may be connected to the board in three different ways. Directly welded to the board, in the position highlighted in blue in the following figure, through the terminal block highlighted in the image in red, or through the connectors for casing, for which more information is given in section “Sockets for casing”.

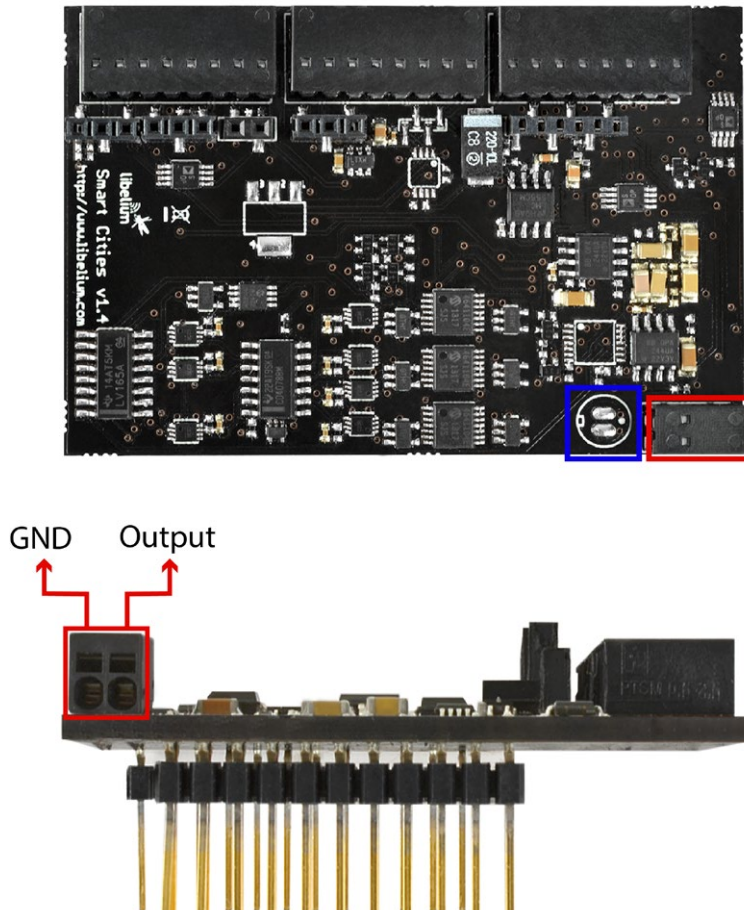


Figure : Image of the sockets for the POM-2735P-R microphone

## 4.5. Ultrasonic Sensor (MaxSonar® from MaxBotix™)

### 4.5.1. Specifications

#### XL-MaxSonar®-WRA1™:

**Operation frequency:** 42kHz

**Maximum detection distance:** 765cm

**Maximum detection distance (analog output):** 600cm (powered at 3.3V) - 700cm (powered at 5V)

**Sensitivity (analog output):** 3.2mV/cm (powered at 3.3V) – 4.9mV/cm (powered at 5V)

**Power supply:** 3.3 ~ 5V

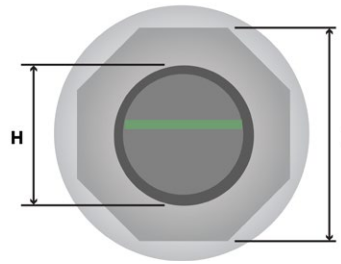
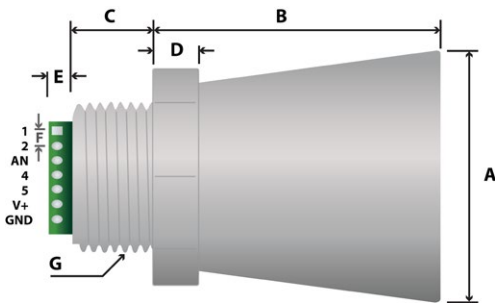
**Consumption (average):** 2.1mA (powered at 3.3V) – 3.2mA (powered at 5V)

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

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



Figure : Ultrasonic XL-MaxSonar®-WRA1 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 XL-MaxSonar®-WRA1 sensor dimensions

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

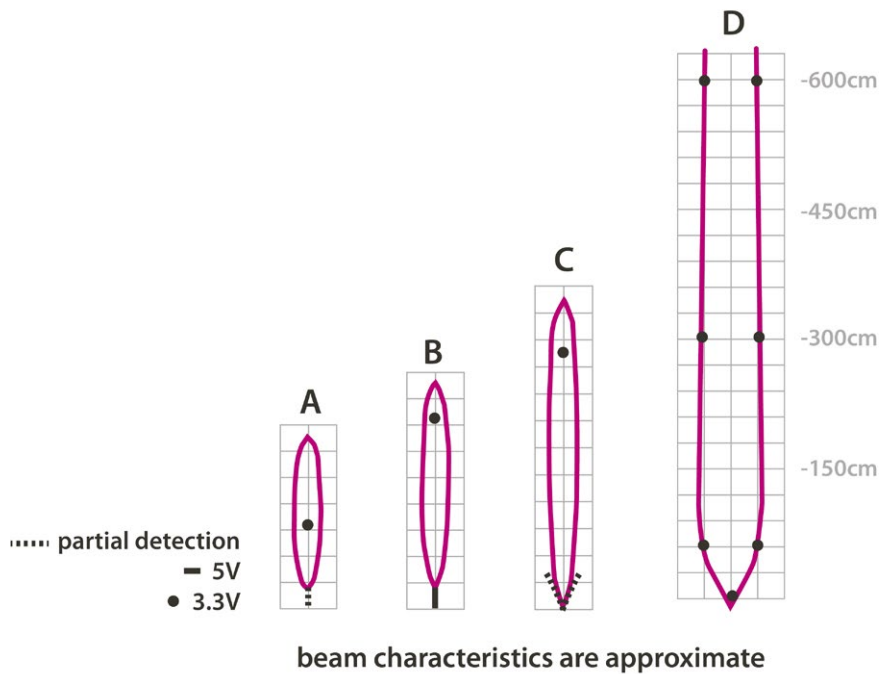


Figure : Diagram of the sensor beam extracted from the data sheet of the XL-MaxSonar®-WRA1™ sensor from MaxBotix

#### LV-MaxSonar®-EZ0™:

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



Figure : Ultrasonic LV-MaxSonar®-EZ0 from MaxBotix™ sensor

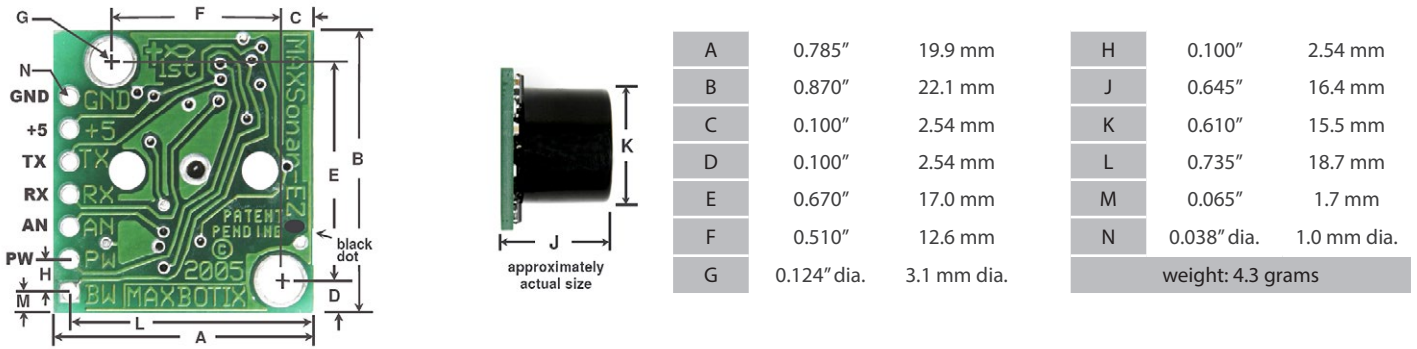


Figure : Ultrasonic LV-MaxSonar®-EZ0 sensor dimension

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

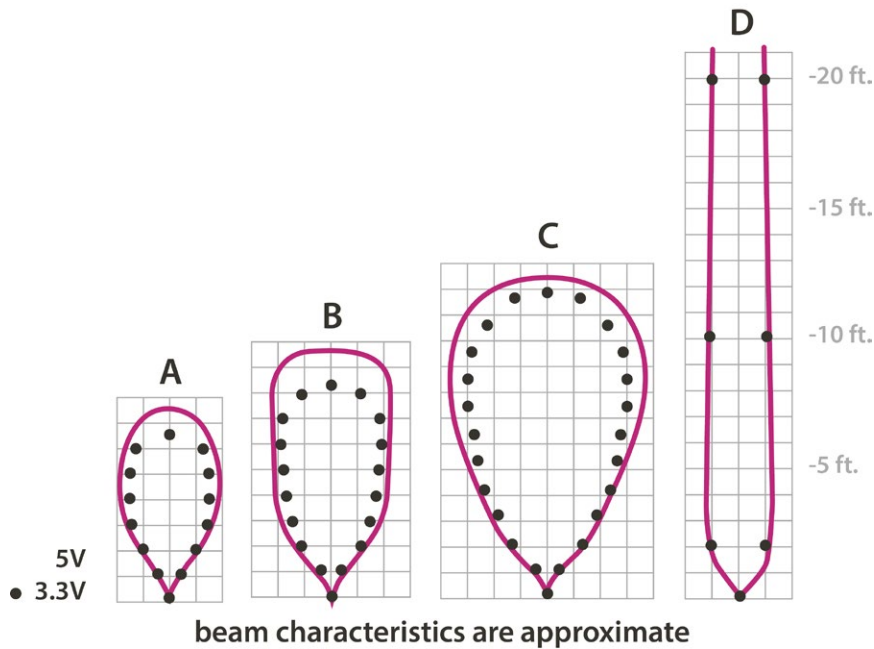


Figure : Diagram of the sensor beam extracted from the data sheet of the LV-MaxSonar®-EZ0™ sensor from MaxBotix

## 4.5.2. Measurement Process

The MaxSonar® sensors from MaxBotix outputs an analog voltage proportional to the distance to the object detected. That voltage may be read through the analog input `ANALOG4`, while pin `DIGITAL2` can be used to activate or deactivate the power supply of the sensor.

In the next figure we can see a drawing of an example application for the ultrasonic sensors.

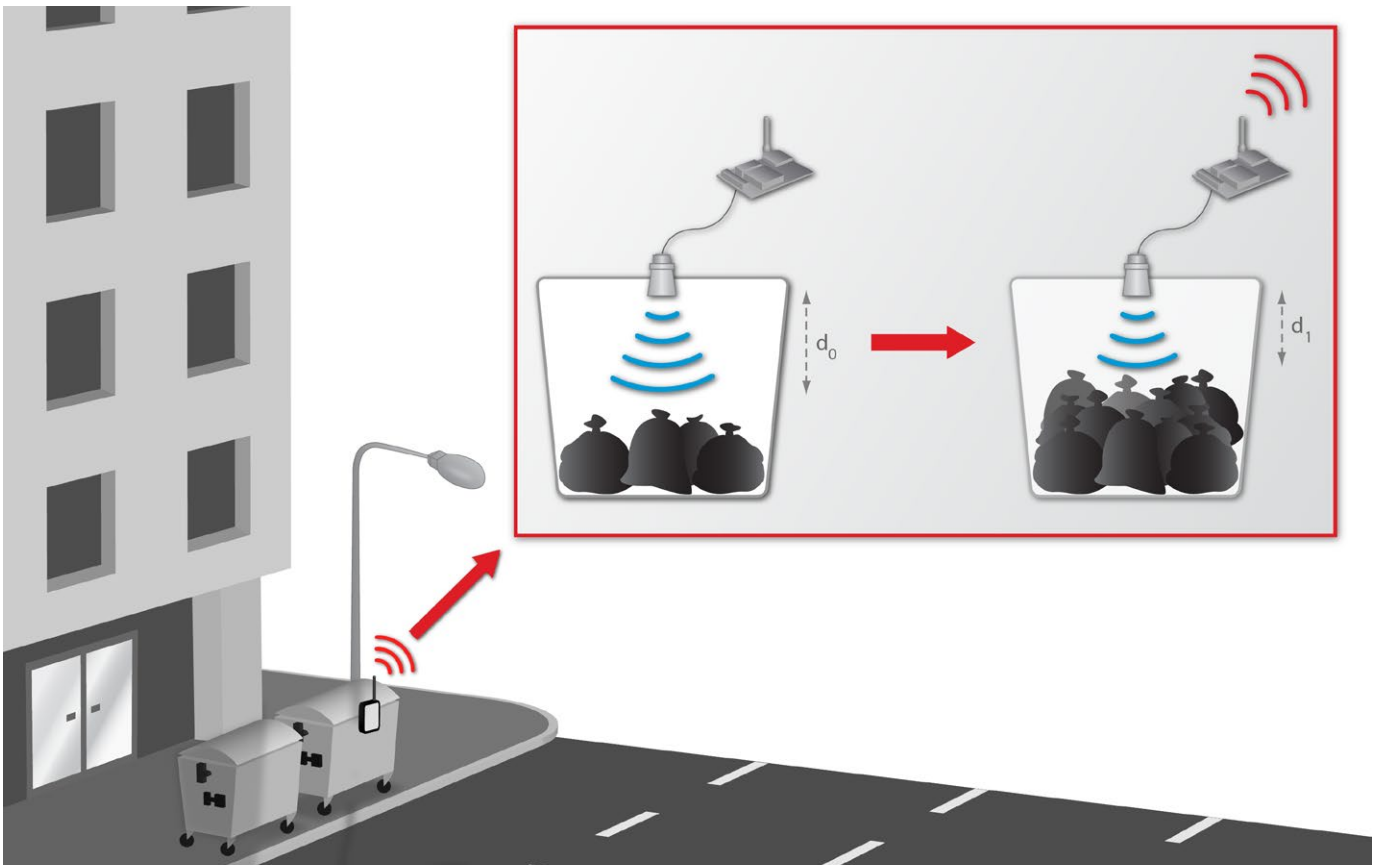


Figure : Example of application for the MaxSonar® sensors

The garbage levels in bins can be controlled in order to ensure an efficient waste collection system.

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

```
{
  SensorCities.ON();
  SensorCities.setSensorMode(SENS_ON, SENS_CITIES_ULTRASOUND_3V3);
  delay(2000);
  float distance_value;
  distance_value = SensorCities.readValue(SENS_CITIES_ULTRASOUND_3V3, SENS_US_WRA1);
}
```

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

<http://www.libelium.com/development/waspmote/examples/sc-8-ultrasound-sensor-reading>

### 4.5.3. Socket

This sensors share the sockets with the MCP9700A temperature sensor or the humidity 808H5V5 sensor. The pin correspondence, highlighted in the figure below, is the same for both. In section “Sockets for casing” more information about the corresponding pinout at the sockets for casing applications can be found.

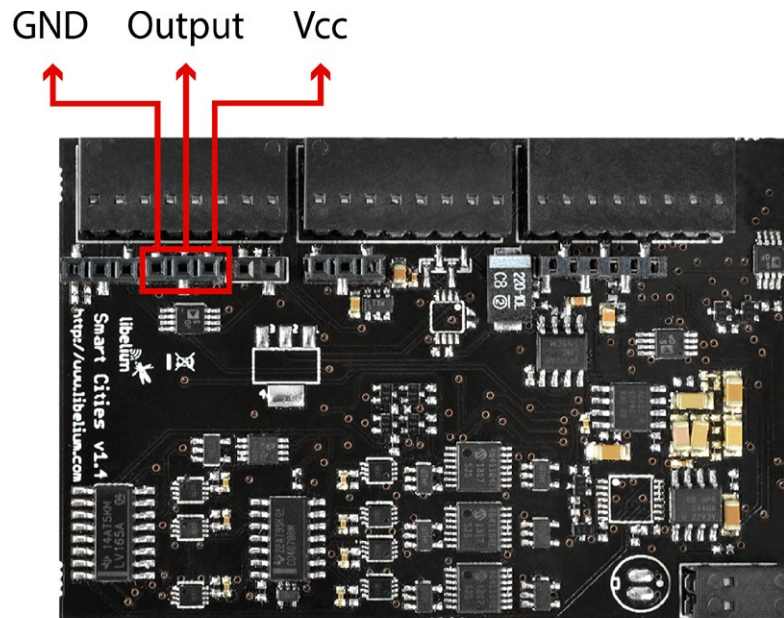


Figure : Image of the socket for connecting the MaxSonar® Sensors



## 4.6. Humidity Sensor (808H5V5)

### 4.6.1. Specifications

**Measurement range:** 0 ~ 100%RH

**Output signal:** 0,8 ~ 3.9V (25°C)

**Accuracy:**  $<\pm 4\%$  RH (a 25°C, range 30 ~ 80%),  $<\pm 6\%$  RH (range 0 ~ 100)

**Typical consumption:** 0.38mA

**Maximum consumption:** 0.5mA

**Power supply:** 5VDC  $\pm 5\%$

**Operation temperature:** -40 ~ +85°C

**Storage temperature:** -55 ~ +125°C

**Response time:** <15 seconds



Figure : Image of the 808H5V5 sensor

### 4.6.2. Measurement Process

This is an analog sensor which provides a voltage output proportional to the relative humidity in the atmosphere. As the sensor's signal is outside of that permitted at the input of the analog-to-digital converter of the Waspote's processor, its output voltage has been adapted to a range of values between 0,48 and 2,34V. In the next figure we can see a graph of the output voltage vs the relative humidity prior to this conversion. The supply voltage of the sensor is controlled through a solid state switch, shared with the temperature, luminosity and dust sensors and activated with signal **DIGITAL2**.

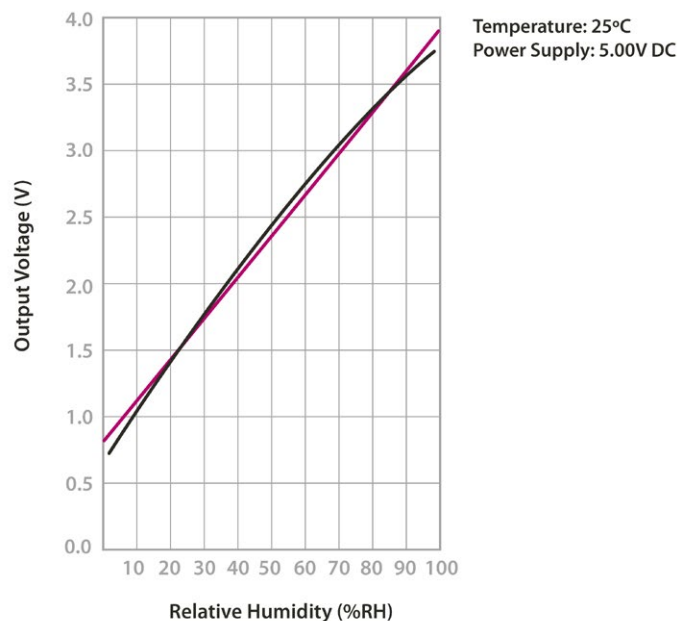


Figure : 808H5V5 humidity sensor output taken from the Sencera Co. Ltd sensor data sheet

Below a sample code for reading the output of the sensor and converting the voltage measured into relative humidity using the libraries of the board is shown:

```
{
  SensorCities.ON();
  SensorCities.setSensorMode(SENS_ON, SENS_CITIES_HUMIDITY);
  delay(15000);
  float humidity_value;
  humidity_value = SensorCities.readValue(SENS_CITIES_HUMIDITY);
}
```

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

<http://www.libelium.com/development/waspote/examples/sc-2-humidity-sensor-reading>

### 4.6.3. Socket

The socket of this sensor (2 ways, 2.54mm pitch) is shown in the following figure, in which the pin correspondence between them has been highlighted (pin 1 corresponds to Vcc, pin 2 corresponds to the output and pin 3 corresponds to GND). In section “Sockets for casing” more information about the corresponding pinout at the sockets for casing applications can be found.

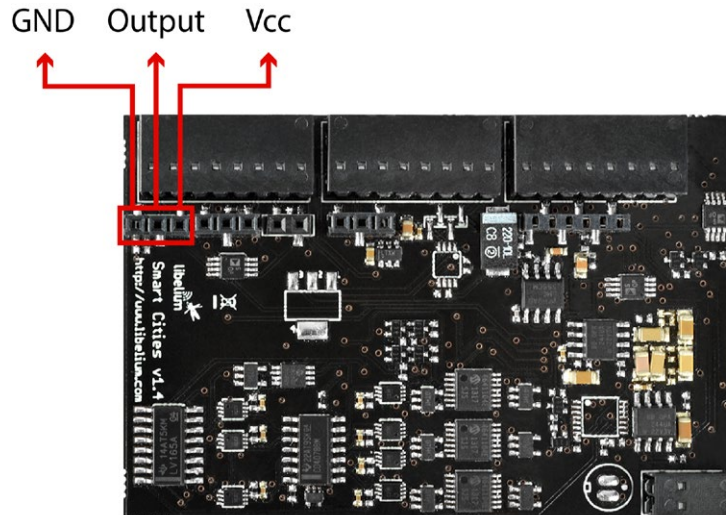


Figure : Image of the socket for the 808H5V5 sensor

## 4.7. Temperature Sensor (MCP9700A)

### 4.7.1. Specifications

**Measurement range:** -40°C ~ +125°C

**Output voltage (0°C):** 500mV

**Sensitivity:** 10mV/°C

**Accuracy:** ±2°C (range 0°C ~ +70°C), ±4°C (range -40 ~ +125°C)

**Typical consumption:** 6µA

**Maximum consumption:** 12µA

**Power supply:** 2.3 ~ 5.5V

**Operation temperature:** -40 ~ +125°C

**Storage temperature:** -65 ~ 150°C

**Response time:** 1.65 seconds (63% of the response for a range from +30 to +125°C)



Figure: Image of the MCP9700A temperature sensor

### 4.7.2. Measurement Process

The MCP9700A is an analog sensor which converts a temperature value into a proportional analog voltage. The range of output voltages is between 100mV (-40°C) and 1.75V (125°C), resulting in a variation of 10mV/°C, with 500mV of output for 0°C. The output voltage may be directly captured by the analog-to-digital converter of the mote’s processor in the input analog pin [ANALOG4](#). The supply voltage of this sensor is controlled through the same switch that the sensors of humidity, luminosity and dust, activated by the digital pin [DIGITAL2](#).

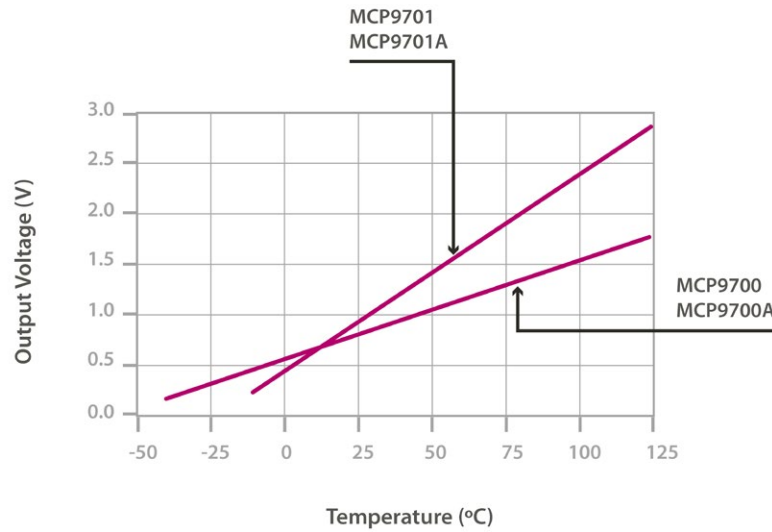


Figure: Graph of the MCP9700A sensor output voltage with respect to temperature, taken from the Microchip sensor's data sheet

A sample code for reading the sensor is provided below:

```
{
  SensorCities.ON();
  SensorCities.setSensorMode(SENS_ON, SENS_CITIES_TEMPERATURE);
  delay(100);
  float temperature_value;
  temperature_value = SensorCities.readValue(SENS_CITIES_TEMPERATURE);
}
```

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

<http://www.libelium.com/development/waspmote/examples/sc-1-temperature-sensor-reading>

### 4.7.3. Socket

The socket for the MCP9700A is shown in the image below, with the pin correspondence highlighted (Pin 1 corresponds to Vcc, pin 2 corresponds to the output and pin 3 corresponds to GND). More information about the socket for casing applications is shown in section "Sockets for casing".

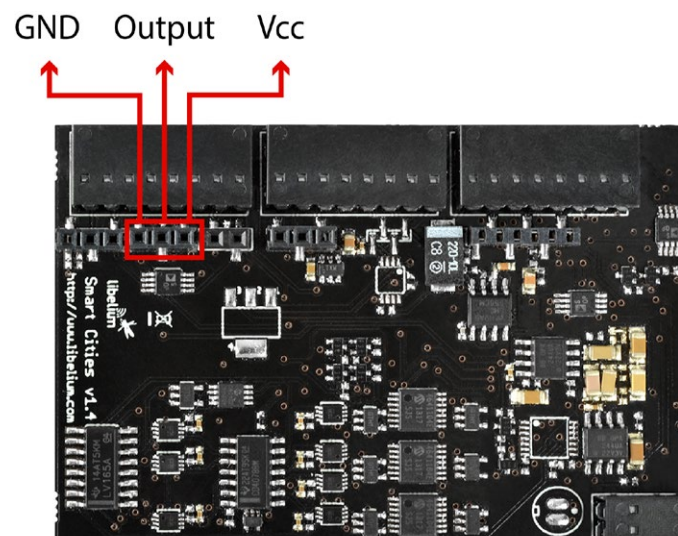


Figure: Image of the socket for the MCP9700A sensor

## 4.8. Luminosity Sensor (LDR)

### 4.8.1. Specifications

**Resistance in darkness:** 20M $\Omega$

**Resistance in light (10lux):** 5 ~ 20k $\Omega$

**Spectral range:** 400 ~ 700nm

**Operating Temperature:** -30°C ~ +75°C

**Minimum consumption:** 0uA approximately



Figure : Image of the LDR luminosity sensor

**Note:** This sensor saturates soon in the presence of “normal intensity” of light, so it is not possible to distinguish “medium light” from “much light”. Actually, the sensor is more suitable to detect the presence (or not) of light. For more advanced applications, requiring the exact measurement of light, Libelium recommends the Luminosity (luxes accuracy) Sensor Probe. It is available in the Ambient Control models of the Plug & Sense! line. For more information, please read the “Ambient Control” chapter of the Plug & Sense! Sensors Guide.

### 4.8.2. Measurement Process

This is a resistive sensor whose conductivity varies depending on the intensity of light received on its photosensitive part. The measurement of the sensor is carried out through the analog-to-digital converter of the mote’s microcontroller, reading the resulting voltage out of a voltage divider formed by the sensor itself and the load resistor of the socket upon which it has been connected. This sensor shares the power supply with the dust, humidity and temperature sensors, which can be controlled through the output digital pin **DIGITAL2**, used to handle a solid state switch.

The measurable spectral range (400nm – 700nm) coincides with the human visible spectrum so it can be used to detect light/darkness in the same way that a human eye would detect it.

Below, a small sample of code for reading the output value of the sensor is shown:

```
{
  SensorCities.ON();
  SensorCities.setSensorMode(SENS_ON, SENS_CITIES_LDR);
  delay(100);
  float ldr_value;
  ldr_value = SensorCities.readValue(SENS_CITIES_LDR);
}
```

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

<http://www.libelium.com/development/waspmote/examples/sc-3-ldr-sensor-reading>

### 4.8.3. Socket

In the following figure we can see highlighted the socket upon which the LDR sensor must be placed. Since this sensor behaves as a simple resistor, polarity should not be taken into account when connecting it. More information about the sockets for casing applications can be found in section “Sockets for casing”.

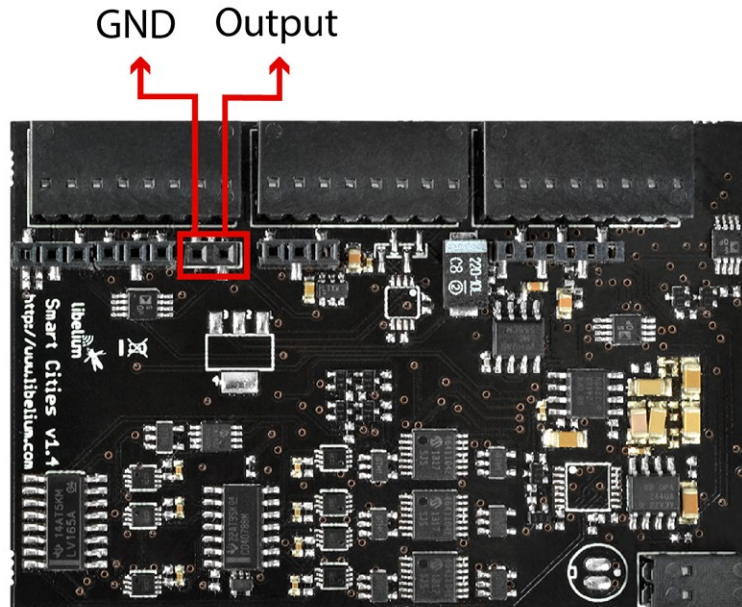


Figure: Image of the socket for the LDR sensor

## 4.9. Sensor interruptions

The Smart Cities board includes all the electronics necessary to generate interruptions from the output signals of all the sensors in it (with the exception of the Noise Sensor), so they can be used to manage associated processes in applications where a continuous monitoring of some of the parameters is required. It must be taken into account, regarding the calculation of the life of the battery, that not all the sensors in this board are low consumption devices, so it is possible that this monitoring will not be feasible unless a power source to recharge the battery continuously is available.

The interruptions from the sensors are generated in the board when one of those surpasses a threshold, defined with a digital potentiometer, setting the output of a voltage comparator high. Since there is only one interruption pin accessible from the board, the outputs of the comparators merge into a logic OR gate that generates a single output to trigger the interruption in the processor. A shift register captures the state of the comparators at the moment of the interruption, so it is not necessary to read all the sensors immediately afterwards to know which of them has triggered it.

In section “API” all the information necessary to manage the interruptions with the functions of the board’s library is provided.



## 4.10. Sockets for casing

In case the Smart Cities 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 above the slot input pin and pull off the wire softly.

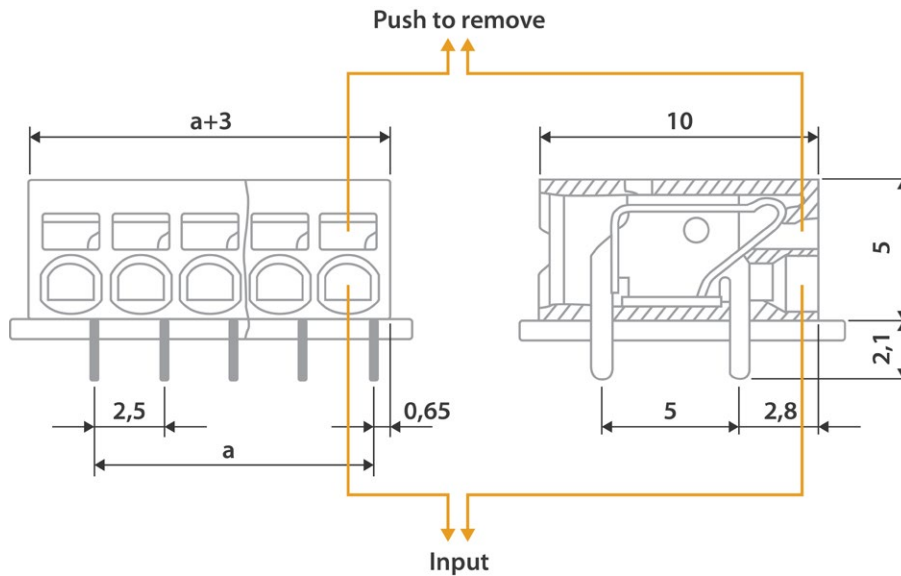


Figure : Diagram of a socket extracted from the Phoenix Contact data sheet

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.

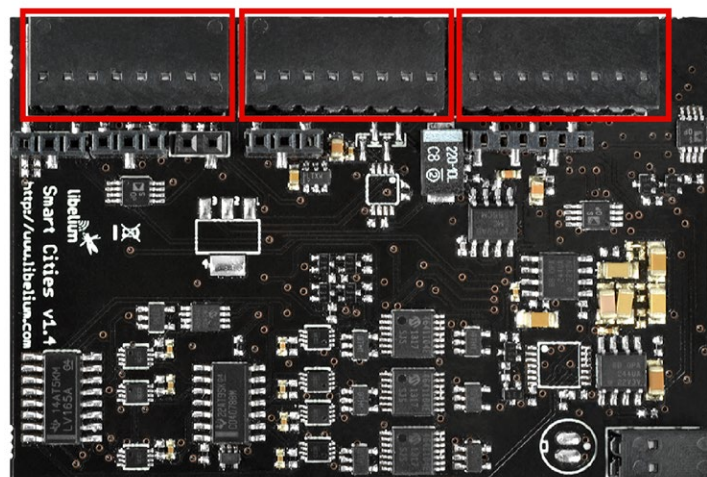


Figure : Image of the sockets for casing applications

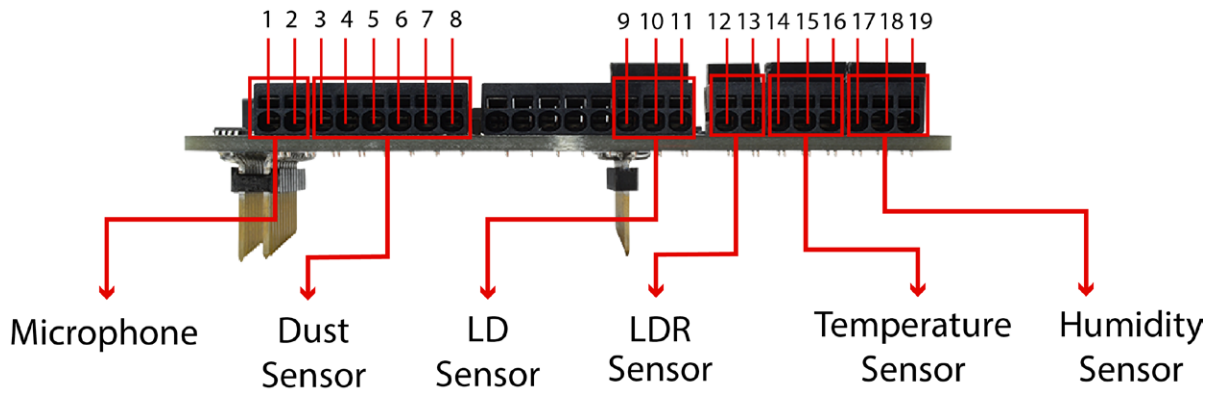


Figure: Image of the pin correspondence between the sockets and the sensors

Sensor	Pin	Function
<b>Microphone</b>	1	Output
	2	GND
<b>Particles sensor (PM-10) - Dust sensor</b>	3	Vcc
	4	Vo
	5	S-GND
	6	LED
	7	LED-GND
	8	V-LED
<b>Linear displacement Sensor</b>	9	Vcc
	10	Output
	11	GND
<b>LDR Sensor / Crack detection and propagation sensors</b>	12	Vcc
	13	Output
<b>Temperature sensor / Ultrasonic sensors</b>	14	Vcc
	15	Output
	16	GND
<b>Humidity sensor</b>	17	Vcc
	18	Output
	19	GND

## 5. Board configuration and programming

### 5.1. Hardware configuration

The Smart Cities board does not require of any handling of the hardware by the user except for placing the sensors in their corresponding position. In the section dedicated to each connector we can see an image of the connections between the socket and its corresponding sensor.

### 5.2. API

The Smart Cities Board library compiles a series of functions that allow the user to handle in an easy way all the resources of the board, the power supply, the start-up and reading of the sensors and the interruptions.

**Important:** for those users of Smart Cities v1.3 or lower and Smart Cities v1.4 without a red-coloured logo (sent before March 2014) and additional change must be done to configure the system. You must open the `WaspSensorCities.cpp` file and change the following line:

```
WaspRegister |= REG_CITIES_V15;
```

By this one:

```
WaspRegister |= REG_CITIES_V14;
```

When using this board remember it is mandatory to include the `SensorCities` library by introducing the next line at the beginning of the code:

```
#include <WaspSensorCities.h>
```

The functions to handle all the features of the board, included in the `WaspSensorCities` API library, are detailed below:

#### `SensorCities.ON()`

Turns on the sensor board by activating the 3.3V and 5V supply lines.

#### `SensorCities.OFF()`

Turns off the sensor board by cutting the 3.3V and 5V supply lines.

#### `SensorCities.setBoardMode(MODE)`

This function is used to manage the power supply applied to the Smart Cities board. Assigning the value `SENS_ON` to the variable `MODE` activates the Waspote's switches which allow the passage of the 3.3V and 5V voltages, while assigning `SENS_OFF` disconnects both switches cutting the power.

#### `SensorCities.setSensorMode(MODE, SENSOR)`

This function, similar to `setBoardMode`, allows to activate or deactivate the power of each sensor independently.

The state on which the sensor should be set is defined through the variable `MODE`, which can take the values `SENS_ON`, to connect the power of the sensor, or `SENS_OFF`, to disconnect it.

The sensor, circuit or group of sensors that we are going to manage is stored in the variable `SENSOR`, that can take the following values:

- `SENS_CITIES_DUST`, to activate the dust sensor.
- `SENS_CITIES_CD`, to activate the crack detection sensor.
- `SENS_CITIES_LD`, to activate the linear displacement sensor.
- `SENS_CITIES_AUDIO`, to activate the microphone and its electronics.

- `SENS_CITIES_ULTRASOUND_3V3`, to activate the ultrasonic sensor on the temperature socket.
- `SENS_CITIES_AUDIO`, to activate the microphone and its electronics.
- `SENS_CITIES_ULTRASOUND_5V`, to activate the ultrasonic sensor on the humidity socket.
- `SENS_CITIES_HUMIDITY`, to activate the humidity sensor.
- `SENS_CITIES_TEMPERATURE`, to activate the temperature sensor.
- `SENS_CITIES_LDR`, to activate the LDR sensor.

As said in their respective sections, the DUST, LDR, TEMPERATURE and HUMIDITY sensors are controlled through the same switch, so when turning on or off one of them you will be acting on the whole group.

#### `SensorCities.readValue(SENSOR, TYPE)`

The function `readValue` may be used to execute the configuration, reading and conversion process of any of the sensors on the board through the analog-to-digital converter of the mote's processor. In the variable `SENSOR` the sensor to be read is introduced, and the output value is given in floating point format (type `float`). The values that can be assigned to that variable are:

- `SENS_CITIES_DUST`, to read the output of the dust sensor.
- `SENS_CITIES_CD`, to read the value of the crack detection sensor.
- `SENS_CITIES_LD`, to read the value of the linear displacement sensor.
- `SENS_CITIES_AUDIO`, to read the output of the microphone.
- `SENS_CITIES_ULTRASOUND_3V3`, to read the output of the ultrasonic sensor on the temperature socket.
- `SENS_CITIES_HUMIDITY`, to read the output of the humidity sensor.
- `SENS_CITIES_ULTRASOUND_5V`, to read the output of the ultrasonic sensor on the humidity socket.
- `SENS_CITIES_TEMPERATURE`, to read the output of the humidity sensor.
- `SENS_CITIES_LDR`, to read the value of the LDR.

The parameter `TYPE` indicates the kind of ultrasound sensor when reading the any of the MaxBotix sensors. It is not necessary to introduce it when reading any other sensor of the board. The values that the `TYPE` parameter may take are:

- `SENS_US_WRA1`, to read the XL-MaxSonar®-WRA1 sensor.
- `SENS_US_EZ0`, to read the LV-MaxSonar®-EZ0 sensor.

#### `SensorCities.setThreshold(SENSOR, THRESHOLD)`

This function is used to configure the comparison threshold that regulates the interruption trigger from the Smart Cities Board. In the variable `SENSOR` the sensor whose comparison threshold is to be changed is introduced, the identifiers of the sensors mentioned before may be assigned to it. In the `THRESHOLD` variable the value to be given to this threshold is introduced in floating point format (`float`), which must be within a range between 0 and 3.3V.

#### `SensorCities.attachInt()`

The `attachInt` function, implemented as is in the code, including no parameters, enables interruptions generated by the board's sensors, allowing the microprocessor to recognize and process them as such.

#### `SensorCities.detachInt()`

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

#### `SensorCities.loadInt()`

The `loadInt` instruction is used to read the content of the shift register and store its output in an integer variable called `SensorCities.intFlag`, in which the sensor that has caused the interruption and other sensors activated at that moment appear. Once all the registers have been read, they restart from zero, not loading again until a new interruption is triggered. To recognize if a sensor has produced an interruption, it is sufficient to carry out a logic comparison between the identifier of the sensor and the `intFlag` variable.

A basic program to detect events from the board will present a similar structure to the following, subject to changes in dependence of the application:

1. The board is switched on using the function `SensorCities.ON`.
2. Initialization of the RTC using `RTC.ON` to avoid conflicts in the I2C bus.
3. Configuration of the thresholds of those sensors which may generate an interruption with function `SensorCities.setThreshold`.
4. Activation of the sensors to generate given interruptions using function `SensorCities.setSensorMode`.
5. Enable interruptions from the board using the function `SensorCities.attachInt`.
6. Put the mote to sleep with the functions `PWR.sleep` or `PWR.deepSleep`.
7. When the mote wakes up, disable interruptions from the board using function `SensorCities.detachInt`.
8. Load the value stored in the shift register with function `SensorCities.loadInt`.
9. Process the interruption:
  - Turn on those inactive sensors to be read using function `SensorCities.setSensorMode`.
  - Take the measurements needed using function `SensorCities.readValue`.
  - Turn off the sensors that shall not generate an interrupt with function `SensorCities.setSensorMode`.
  - Store or send via a radio module the gathered information.
10. Return to step 5 to enable interruptions and put the mote to sleep.

Below is shown a sample code where the temperature, dust and linear displacement sensors are read every five minutes and their values sent through XBee 802.15.4. If an excessive audio value is detected, an alarm message is sent including the measurement from the microphone. In any case, if a crack has been detected, a message indicating it is included in every frame.

```
/* -----Smart Cities board example-----

www.Libelium.com
*/

// Inclusion of the Smart Cities Sensor Board library
#include <WaspSensorCities.h>

// Inclusion of the Frame library
#include <WaspFrame.h>

// Inclusion of the XBee 802.15.4 library
#include <WaspXBee802.h>

// Set interruption threshold
#define THRESHOLD 1.5

float audio_value = 0;
float crack_width_value = 0;
float temperature_value = 0;
float dust_value = 0;

// Pointer to an XBee packet structure
packetXBee* packet;

void setup()
{
  //Switch on the board
  SensorCities.ON();
  delay(100);

  // Init RTC
  RTC.ON();
  delay(100);
```



```
// Configure not used interruption thresholds
SensorCities.setThreshold(SENS_CITIES_LDR, 3.3);
SensorCities.setThreshold(SENS_CITIES_DUST, 3.3);
SensorCities.setThreshold(SENS_CITIES_LD, 3.3);
SensorCities.setThreshold(SENS_CITIES_HUMIDITY, 3.3);
SensorCities.setThreshold(SENS_CITIES_TEMPERATURE, 3.3);

//Configure the threshold for the crack detection sensor
SensorCities.setThreshold(SENS_CITIES_AUDIO, THRESHOLD);
//Turn on the LDR
SensorCities.setSensorMode(SENS_ON, SENS_CITIES_AUDIO);
delay(3000);
}
void loop()
{
//Enable interruptions from the Cities Metering Board
SensorCities.attachInt();

//Put the mote to sleep
PWR.deepSleep("00:00:00:10", RTC_OFFSET, RTC_ALM1_MODE1, UART0_OFF | UART1_OFF | BAT_
OFF);

//Disable interruptions from the sensor board
SensorCities.detachInt();
//Load the interruption register
SensorCities.loadInt();

// Create new frame (ASCII)
frame.createFrame(ASCII, "Waspote_Pro");

SensorCities.setSensorMode(SENS_ON, SENS_CITIES_CD);
delay(10);
if( SensorCities.readValue( SENS_CITIES_CD ) == 0 )
{
// Add the alarm message to the composition
frame.addSensor(SENSOR_STR, "New crack appeared!");
}

if ( SensorCities.intFlag & SENS_CITIES_AUDIO)
{
//Reading the temperature sensor
audio_value = SensorCities.readValue(SENS_CITIES_AUDIO);
frame.addSensor(SENSOR_MCP, audio_value);
}
else if ( intFlag & RTC_INT )
{
//Turn on the temperature sensor
SensorCities.setSensorMode(SENS_ON, SENS_CITIES_TEMPERATURE);
//Reading the temperature sensor
temperature_value = SensorCities.readValue(SENS_CITIES_TEMPERATURE);
// Turn off the temperature sensor
SensorCities.setSensorMode(SENS_OFF, SENS_CITIES_TEMPERATURE);

//Turn on the crack width sensor
SensorCities.setSensorMode(SENS_ON, SENS_CITIES_LD);
delay(100);
//Reading the crack width sensor
crack_width_value = SensorCities.readValue(SENS_CITIES_LD);
//Turn off the crack width sensor
SensorCities.setSensorMode(SENS_OFF, SENS_CITIES_LD);
```

```
//Turn on the dust sensor
SensorCities.setSensorMode(SENS_ON, SENS_CITIES_DUST);
delay(2000);
//Reading the crack width sensor
dust_value = SensorCities.readValue(SENS_CITIES_DUST);
//Turn off the crack width sensor
SensorCities.setSensorMode(SENS_OFF, SENS_CITIES_DUST);

// Add the values read to the frame composition
frame.addSensor(SENSOR_TCA, temperature_value);
frame.addSensor(SENSOR_LD, crack_width_value);
frame.addSensor(SENSOR_DUST, dust_value);
}
// Init XBee
xbee802.ON();
// Set parameters to packet:
packet=(packetXBee*) calloc(1,sizeof(packetXBee));
packet->mode=BROADCAST;

// Set destination XBee parameters to packet
xbee802.setDestinationParams( packet, "000000000000FFFF", frame.buffer, frame.length);

// Send XBee packet
xbee802.sendXBee(packet);

// Turn off the XBee Module
xbee802.OFF();
delay(100);

// Clear the interruption flag
clearIntFlag();

}
```

The files related to this sensor board are: `WaspSensorCities.cpp`, `WaspSensorCities.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. Consumption

### 6.1. Power control

The Smart Cities Board for Waspote requires both the 3.3V and 5V power supplies output from the mote.

The sensors are powered through three solid state switches that allow to cut or activate separately the supply voltages for each sensor or group of sensors.

The microphone and the linear displacement sensor can be turned on separately, while the power supply of the humidity, luminosity, temperature and dust sensors is controlled through a switch shared by all of them.

Thus, **DIGITAL6** pin controls the switch that activates the 3.3V supply for the microphone, the **DIGITAL7** pin controls the linear displacement sensor's switch (also powered at 5V) and the **DIGITAL2** pin controls the supply voltages of the remaining group of sensors (dust, humidity, luminosity and temperature), which needs both 3.3V and 5V supplies.

All these switches may be controlled through the `SensorCities.setBoardMode` and `SensorCities.setSensorMode` functions implemented in the API. You can find more information about it in section "API".

### 6.2. Tables of consumption

In the following table the consumption of the board is shown, the constant minimum consumption (fixed by the permanently active components) and the consumption of each of the independent blocks that may be powered independently. The board's power can be completely disconnected, reducing the consumption to zero, using the 3.3V and the 5V main switches disconnection `SensorCities.setBoardMode` command included in the library.

	Consumption
<b>Minimum (Constant)</b>	400µA
<b>Various Sensors Group (no sensors connected)</b>	12.5mA
<b>Various Sensors Group (only temperature sensor connected)</b>	12.5mA
<b>Various Sensors Group (only humidity sensor connected)</b>	13.2mA
<b>Various Sensors Group (only luminosity sensor connected)</b>	12.8mA
<b>Various Sensors Group (only ultrasonic sensor connected)</b>	15mA
<b>Crack detection sensor</b>	500µA
<b>Various Sensors Group (only dust sensor connected)</b>	37.5mA
<b>Linear displacement sensor</b>	8.3mA
<b>Microphone</b>	1.1mA

## 6.3. Low consumption mode

The Smart Cities Board has been designed to minimize the consumption of the mote in operation conditions as long as in low consumption modes.

- **Avoid activating all the sensors at the same time**

Although there are no high consumption sensors in the Smart Cities board, increasing the load in the 5V line implies an increase in the DC-DC voltage converter in the mote, so it is recommended, if possible, to avoid activating the sensors powered at 5V at the same time.

- **Use the Waspote low consumption mode**

As the other sensor boards for Waspote, the library of the Smart Cities Board includes all the functions needed to deactivate the sensors which are not being used and put the mote in low consumption mode.

- **Do not connect sensors that are not going to be used**

Since several sensors share the same power line, a sensor that is not going to be used connected to the board will entail an additional consumption, and so a shorter life of the battery.

## 7. API Changelog

Keep track of the software changes on this link:

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

## 8. Documentation changelog

### From v5.3 to v5.4

- Note about the differences between the LDR sensor and the Luminosity (luxes accuracy) sensor probe

### From v5.2 to v5.3

- References to the new LoRaWAN module

### From v5.1 to v5.2

- References to the new Sigfox module
- Note indicating interruption operations are no available for the Noise Sensor

### From v5.0 to v5.1:

- The Dust sensor is discontinued in the Plug & Sense! ecosystem
- Note about the new Particle Matter (PM1, PM2.5, PM10) – Dust Sensor for Plug & Sense! Smart Environment PRO; this is the recommended sensor for dust sensing

### From v4.9 to v5.0:

- References to the new LoRa module
- Link to the new online API changelog

### From v4.8 to v4.9:

- Radios table for Plug&Sense! updated

### From v4.7 to v4.8:

- The correct noise sensor's output is indicated: dBA
- Introduced the new equivalent noise sensor's name
- Crack propagation sensor was discontinued
- Noted that the dust sensor must be maintained periodically, and purchased with the big Solar Shield
- Noted that the noise sensor, Smart Cities board and Waspote must be purchased together

### From v4.6 to v4.7:

- Radios table for Plug&Sense! updated

### From v4.5 to v4.6:

- API changelog updated to API v008
- Note to old users of Smart Cities added

### From v4.4 to v4.5:

- API changelog updated to API v006

### From v4.3 to v4.4:

- Added reference to the External SIM socket



**From v4.2 to v4.3:**

- Fixed some errata in code examples
- Magnet reset reference in Plug & Sense!

**From v4.1 to v4.2:**

- Added references to 3G/GPRS Board in section: Radio Interfaces

## 9. Maintenance

- In this section, the term “WaspMote” encompasses both the WaspMote device itself as well as its modules and sensor boards.
- Take care with the handling of WaspMote, 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.

