Events 2.0

Technical Guide

libelium

waspmote

Document version: v4.2 - 04/2013

INDEX

1. General 4
1.1. General and safety information 4
1.2. Conditions of use 4

2. Waspmote Plug & Sense! 5
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 Security 12

3. Hardware 14
3.1. General Description 14
3.2. Specifications 14
3.3. Electrical Characteristics 14

4. Sensors 15
4.1. Pressure Sensor (Flexiforce PS-02) 15
4.1.1. Specifications 15
4.1.2. Measurement Process 15
4.2. Bend Sensor (FLX-01-H) 16
4.2.1. Specifications 16
4.2.2. Measurement Process 16
4.3. Vibration Sensor (PZ-01 and PZ-08) 17
4.3.1. Specifications 17
4.3.2. Measurement Process 18
4.4. Temperature Sensor (MCP9700A) 19
4.4.1. Specifications 19
4.4.2. Measurement Process 19
4.5. Liquid Level Sensors (PTFA3415, PTFA0100, PTFA1103) 20
4.5.1. Specifications 20
4.5.2. Measurement Process 21

4.6. Luminosity Sensor (LDR) 22
4.6.1. Specifications 22
4.6.2. Measurement Process 22
4.7. Presence Sensor (PIR) 22
4.7.1. Specifications 22
4.7.2. Measurement Process 23
4.8. Hall Effect Sensor (PLA41201) 24
4.8.1. Specifications 24
4.8.2. Measurement Process 24
4.9. Liquid Presence Sensor 25
4.9.1. Specifications 25
4.9.2. Measurement Process 25
4.10. Stretch Sensor (STRX-04) 26
4.10.1. Specifications 26
4.10.2. Measurement Process 26
4.11. Humidity Sensor (808H5V5) 27
4.11.1. Specifications 27
4.11.2. Measurement Process 27
4.12. Liquid Flow Sensor (FS100A, FS200 and FS400 from Broiltech) 28
4.12.1. Specifications 28
4.12.2. Measurement Process 29
4.13. Design and connections 30
4.13.1. Connectors 1, 2 and 3 30
4.13.2. Connector 4 31
4.13.3. Connectors 5 and 6 32
4.13.4. Connector 7 32
4.13.5. Connector 8 33
4.13.6. Sockets for casing 33

5. Board configuration and programming 36
5.1. Hardware configuration 36
5.2. API 37

6. API Changelog 41

7. Consumption 42
7.1. Power control 42
7.2. Consumption table 42
7.3. Low Consumption Mode 43

8. Documentation changelog 44

9. Maintenance 45

10. Disposal and recycling 46

1. General

1.1. General and safety information

- In this section, the term "Waspmote" encompasses both the Waspmote 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.
- Waspmote consists of highly sensitive electronics which is accessible to the exterior, handle with great care and avoid bangs or hard brushing against surfaces.
- Check the product specifications section for the maximum allowed power voltage and amperage range and consequently always use a current transformer and a battery which works within that range. Libelium is only responsible for the correct operation of the device with the batteries, power supplies and chargers which it supplies.
- Keep the device within the specified range of temperatures in the specifications section.
- Do not connect or power the device with damaged cables or batteries.
- Place the device in a place only accessible to maintenance personnel (a restricted area).
- Keep children away from the device in all circumstances.
- If there is an electrical failure, disconnect the main switch immediately and disconnect that battery or any other power supply that is being used.
- If 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.
- Waspmote 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 Waspmote in accordance with the electrical specifications and the environment described in the "Electrical Data" section of this manual.
- Waspmote 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 Waspmote 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/waspmote**

The "General Conditions of Libelium Sale and Use" document can be found at:
http://www.libelium.com/development/waspmote/technical_service

2. Waspmote Plug & Sense!

The new Waspmote 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 "Waspmote 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 and 3G/GPRS
- 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

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 1: Connecting a sensor probe to Waspmote 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 45º holder which ensures the maximum performance of each outdoor installation.

Figure 2: Waspmote 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 3: Internal solar panel

Figure 4: Waspmote Plug & Sense! powered by an internal solar panel

2.4. Programming the Nodes

Waspmote 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 5: 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 Waspmote sensor nodes at the same time by using a laptop and the Waspmote Gateway.

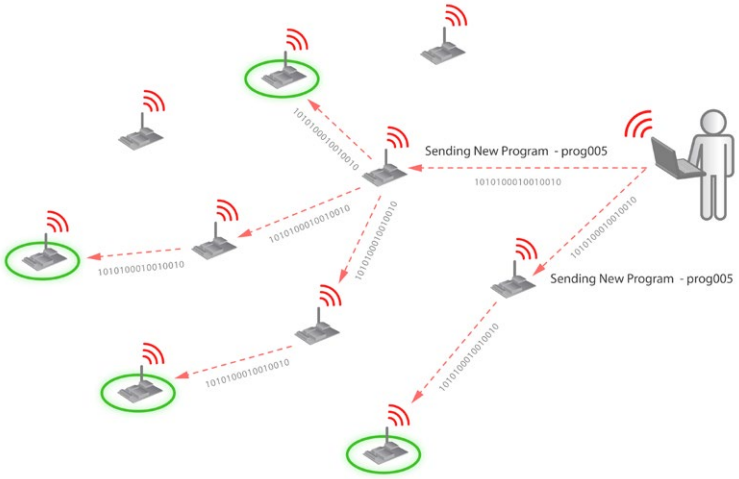

Figure 6: Typical OTAP process

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
Wifi	802.11b/g	2.4GHz	0dBm - 12dBm	-83dBm	50m-500m
GPRS	-	850MHz/900MHz/ 1800MHz/1900MHz	2W(Class4) 850MHz/900MHz, 1W(Class1) 1800MHz/1900MHz	-109dBm	
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	

** 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 an 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

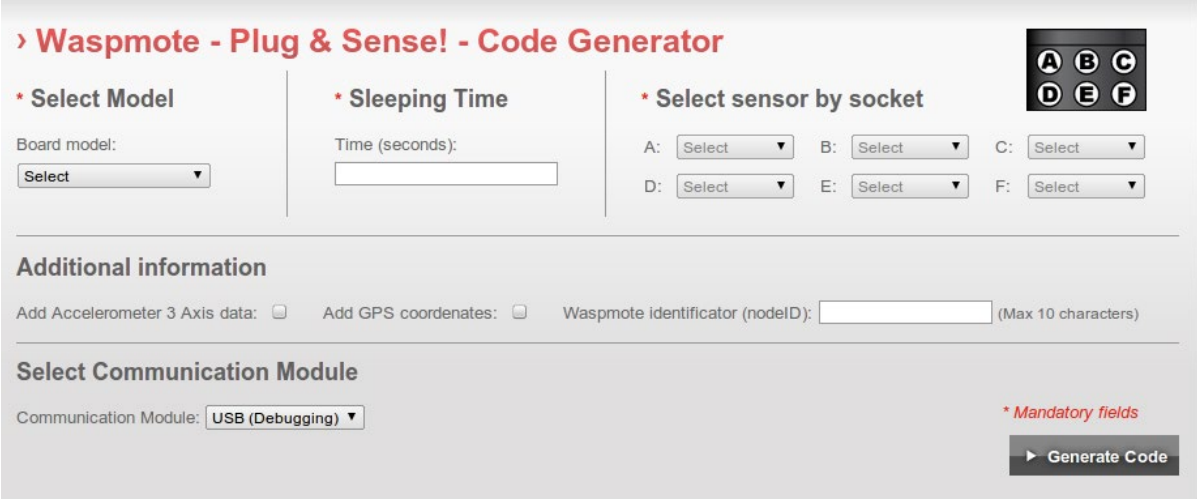

Figure 7: Code Generator

2.7. Data to the Cloud

The Sensor data gathered by the Waspmote Plug & Sense! nodes is sent to the Cloud by **Meshlium**, the Gateway router specially designed to connect Waspmote sensor networks to the Internet via Ethernet, Wifi 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 8: Meshlium

2.8. Meshlium Storage Options

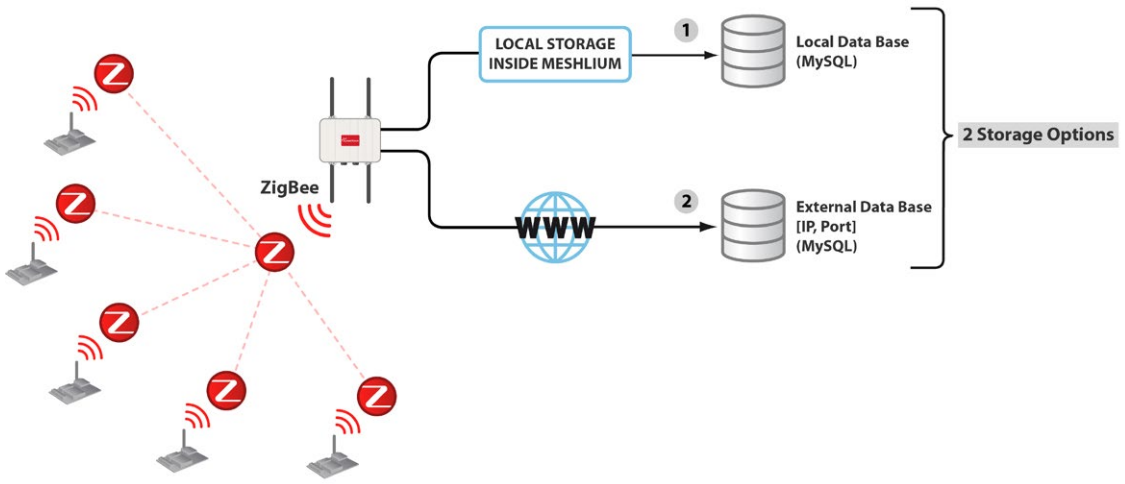

Figure 9: Meshlium Storage Options

- Local Data Base
- External Data Base

2.9. Meshlium Connection Options

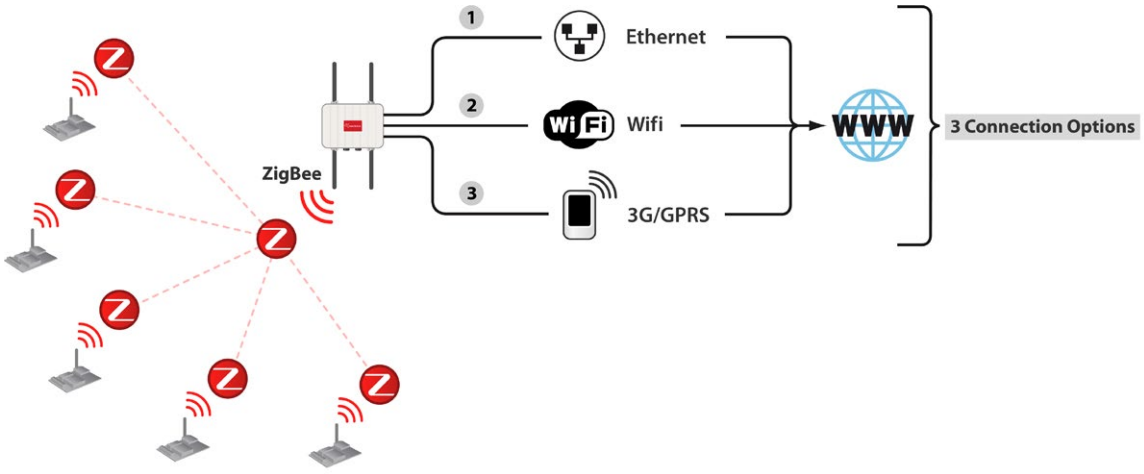

Figure 10: Meshlium Connection Options

- ZigBee → Ethernet
- ZigBee → Wifi
- ZigBee → 3G/GPRS

2.10. Models

There are some defined configurations of Waspmote Plug & Sense! depending on which sensors are going to be used. Waspmote 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 Waspmote. 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.

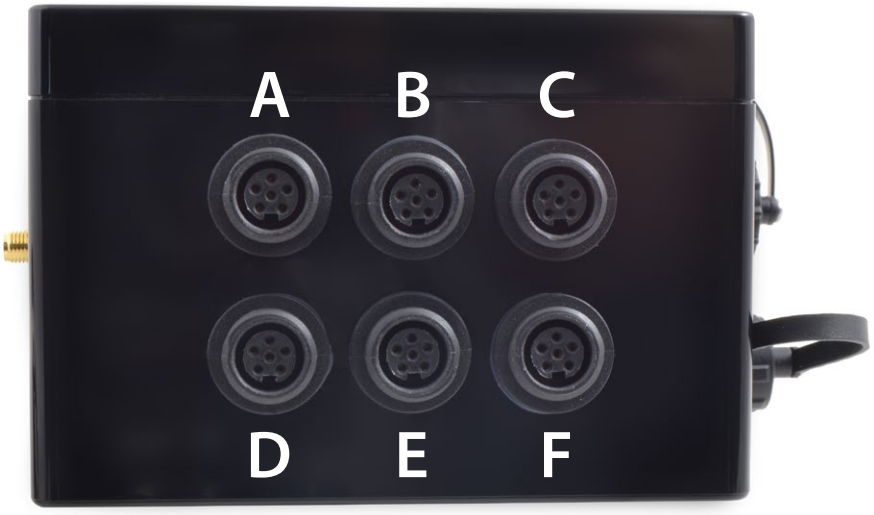

Figure 11: Identification of sensor sockets

2.10.1. Smart Security

The main applications for this Waspmote Plug & Sense! configuration are perimeter access control, liquid presence detection and doors and windows openings.

Figure 12: Smart Security Waspmote 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	Temperature + Humidity (Sensirion)	9247
B	Liquid flow	9296, 9297, 9298
C	Presence - PIR	9212
D	Luminosity	9205
	Liquid level	9239, 9240, 9242
	Liquid presence	9243
	Hall effect	9207
E	Luminosity	9205
	Liquid level	9239, 9240, 9242
	Liquid presence	9243
	Hall effect	9207
F	Luminosity	9205
	Liquid level	9239, 9240, 9242
	Liquid presence	9243
	Hall effect	9207

Figure 13: Sensor sockets configuration for Smart Security model

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

Configurations of the Presence sensor probe (PIR)

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

3. Hardware

3.1. General Description

The sensors are **active** on the Events Sensor Board 2.0 while Waspmote is in **Sleep** or **Deep Sleep mode**. When a sensor picks up a value higher than a previously programmed **threshold**, a signal is generated which wakes the mote from its low consumption status and tells it which sensor has generated the signal.

If Waspmote 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 **8** sensors whose outputs are compared with a threshold value and are combined in a **OR** logical gate which implements a change in the interruption bit which wakes the mote. The value of these thresholds is programmed by the microcontroller through the bus I2C as the system is controlled through **digital potentiometers (digipots)**.

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

3.2. Specifications

Weight:	20gr
Dimensions:	73.5 x 51 x 1.3 mm
Temperature Range:	[-20ºC, 65ºC]

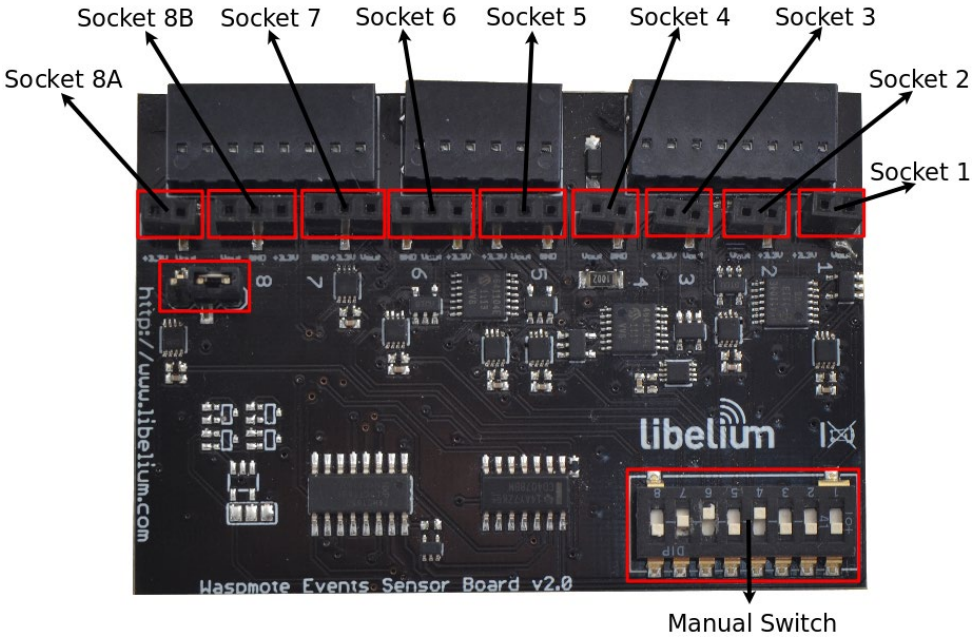

Figure 14: Upper side

3.3. Electrical Characteristics

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

4. Sensors

4.1. Pressure Sensor (Flexiforce PS-02)

4.1.1. Specifications

Measurement range: 0 ~ 25lb (11.34kg)
Resistance range: 5000 ~ 20kΩ
Length: 203mm
Width: 14mm
Sensitive Area: 10mm (diameter)
Linearity error: <±5%
Repeatability: <±2.5% from the bottom of the scale
Operating temperature: -9ºC ~ 60ºC
Response time: < 5 microseconds
Minimum consumption: 0µA*

Figure 15: FlexiforcePS-02Sensor

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

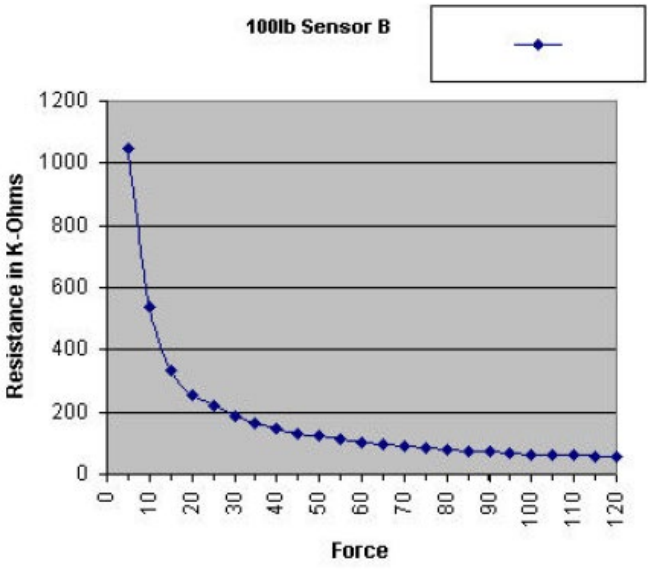

Figure 16: Graph of resistance of the PS-03 Flexiforce sensor of up to 120lbs taken from the sensor's manual

4.1.2. Measurement Process

The Flexiforce PS-02 is a pressure-sensitive resistive sensor. Resistance between the two ends of the terminals therefore varies depending on the force exerted on the circular end of the sensor. The Flexiforce PS-02 can be connected to the Waspmote board with a probe, using connectors 1, 2, and 3, although connector 1 is recommended because of its high resistance.

In figure 17, we can see an example of the sensor's use to monitor the status of a water tank depending on its weight, generating an alarm when the liquid goes over the limit marked as the board's threshold.

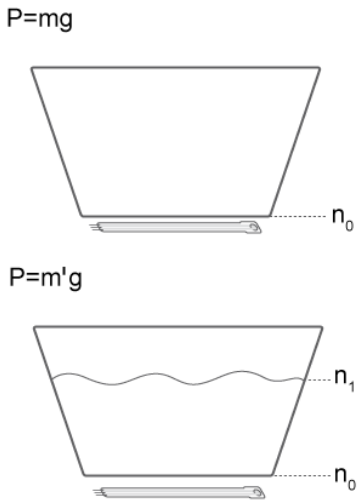

Figure 17: Monitoring content of a tank using the Flexiforce PS-02

A code for reading the sensor is shown below:

```
float value;

{
  SensorEventv20.ON();
  delay(10);
  value = SensorEventv20.readValue(SOCKET, SENS_RESISTIVE);
}
```

`value` is a float variable where the resistance of the in kiloohms will be stored.

`SOCKET` indicates on which connector the sensor is placed (for this sensor it may be `SENS_SOCKET1`, `SENS_SOCKET2` and `SENS_SOCKET3`).

4.2. Bend Sensor (FLX-01-H)

4.2.1. Specifications

Resistance range: 50 ~ 200kΩ
Length: 11.43cm
Width: 0.95cm
Thickness: 0.096cm
Minimum consumption: 0µA*

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

Figure 18: FLX-01-H Sensor

4.2.2. Measurement Process

This is a resistive sensor whose conductivity increases when folded. Its initial resistance is 200kΩ and can be reduced to about 50kΩ depending on the angle and degree of fold, allowing it to be used in applications such as movement detection relating to pieces or impact. The sensor can be monitored through any of connectors 1, 2, and 3, but because of the range of resistances in which it works, using connector 2 is recommended. Figure 19 shows an example of two situations where a change in the sensor's resistance would be found.

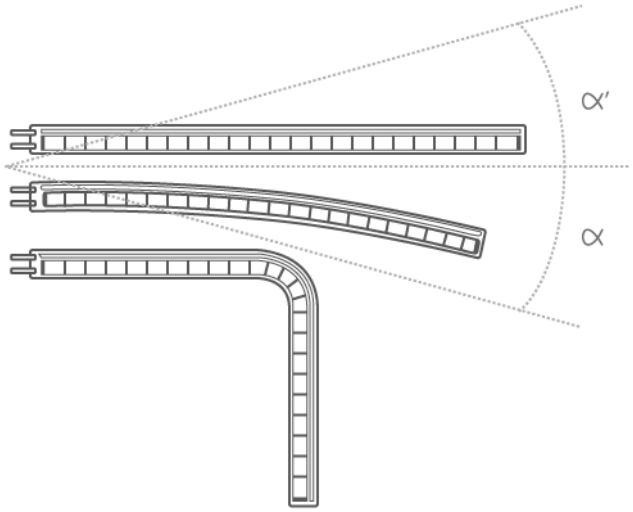

Figure 19: Diagram of folds that would generate a change in resistance of the FLX-01-H sensor

A code for reading the sensor is shown below:

```
float value;

{
  SensorEventv20.ON();
  delay(10);
  value = SensorEventv20.readValue(SOCKET, SENS_RESISTIVE);
}
```

`value` is a float variable where the resistance of the in kiloohms will be stored.

`SOCKET` indicates on which connector the sensor is placed (for this sensor it may be `SENS_SOCKET1`, `SENS_SOCKET2` and `SENS_SOCKET3`).

4.3. Vibration Sensor (PZ-01 and PZ-08)

4.3.1. Specifications

PZ-01

Length: 2.41cm
Width: 1.30cm
Operating temperature: -15ºC ~ +60ºC
Storage temperature: -40ºC ~ +60ºC
Sensitivity: 10mV/microstrain
Minimum consumption: 0µA

Figure 20: PZ-01 sensor

PZ-08

Length: 1.28cm
Width: 0.60cm
Operating temperature: -20ºC ~ +60ºC
Storage temperature: -40ºC ~ +60ºC
Sensitivity: 1V/g
Minimum consumption: 0µA

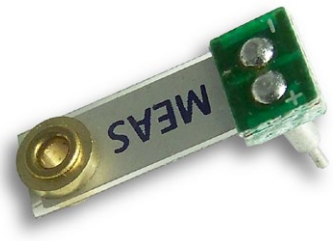

Figure 21: PZ-08 sensor

4.3.2. Measurement Process

The PZ-01 and PZ-08 sensors are two piezoelectric laminates which generate voltage at its output when subjected to a distortion. This therefore makes it useful as a system to measure vibrations caused by acceleration, knocks or contact, being especially useful in low consumption applications as they do not need power voltage for their operation. A connector (connector 4) is included on the Waspmote Events Sensor Board 2.0 specifically for this type of sensor.

Given that the sensor generates current pulses that may be extremely short in function of the pressure detected, it is highly recommended to use the detection of the interruption as reading method instead of the polling of the analog value.

In figure 22 an application example is shown which uses the PZ-08 sensor to detect vibrations generated by contact with a tied thread at the free end of the sensor.

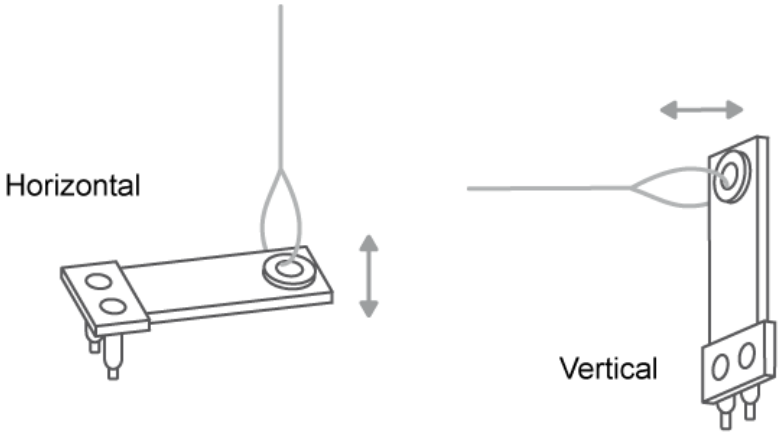

Figure 22: Example of application with the PZ-08 sensor

A code for reading these sensors is shown below:

```
int value;
{
  SensorEventv20.ON();
  delay(10);
  SensorEventv20.attachInt();
  PWR.sleep(UART0_OFF | UART1_OFF | BAT_OFF | RTC_OFF);
  SensorEventv20.detachInt();
  SensorEventv20.loadInt();
  if (SensorEventv20.intFlag & SENS_SOCKET4)
  {
        value = 1;
  } else
  {
        value = 0;
  }
}
```

This code puts the mote to sleep awaiting for an interruption from the sensor board and processes it in function of whether it arrived from the piezoelectric sensor or not.

`value` is an integer variable that takes 1 if an interruption from the sensor has arrived and 0 if arrived from a different sensor.

4.4. Temperature Sensor (MCP9700A)

4.4.1. Specifications

Figure 23: MCP9700A temperature sensor

Measurement range: [-40ºC,+125ºC]
Output voltage (0°): 500mV
Sensitivity: 10mV/ºC
Accuracy: ±2ºC (range 0ºC ~ +70ºC), ±4ºC (range -40 ~ +125ºC)
Typical consumption: 6µA
Maximum consumption: 12µA

4.4.2. 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°) and 1.75V (125°C), resulting in a variation of 10mV/°C, with 500mV of output for 0°C. Being an analog sensor it must be placed on connectors 5 or 6.

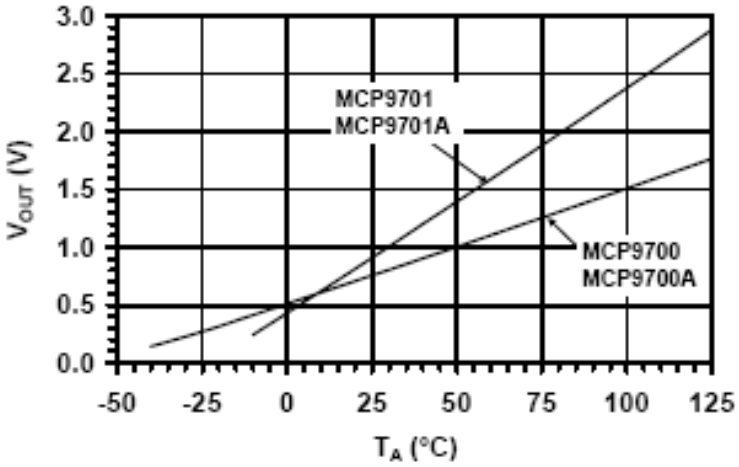

Figure 24: Graphic of the MCP9700A sensor output voltage with respect to temperature, taken from the Microchip sensor's data sheet

A code for reading the sensor is shown below:

```
float value;
{
  SensorEventv20.ON();
  delay(10);
  value = SensorEventv20.readValue(SOCKET, SENS_TEMPERATURE);
}
```

`value` is a float variable where the temperature in Celsius degree (ºC) will be stored.

`SOCKET` indicates on which connector the sensor is placed (for this sensor it may be `SENS_SOCKET5` and `SENS_SOCKET6`).

4.5. Liquid Level Sensors (PTFA3415, PTFA0100, PTFA1103)

4.5.1. Specifications

PTFA3415

Measurement Level: Horizontal
Liquids: Water
Material (box): Propylene
Material (float): Propylene
Operating Temperature: -10ºC ~ +80ºC
Minimum consumption: 0µA*

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

Figure 25: PTFA3415 sensor

PTFA0100

Measurement Level: Horizontal
Liquids: Heavy oils and combustibles
Material (box): Polyamide
Material (float): Polyamide
Operating temperature: -10ºC ~ +80ºC
Minimum consumption: 0µA*

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

Figure 26: PTFA0100 sensor

PTFA1103

Measurement Level: Vertical
Liquids: Water
Material (box): Propylene
Material (float): Propylene
Operating temperature: -10ºC ~ +80ºC
Minimum consumption: 0µA*

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

Figure 27: PTFA1103 sensor

4.5.2. 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 Waspmote, 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 connectors 1, 2, 3 and 8.

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

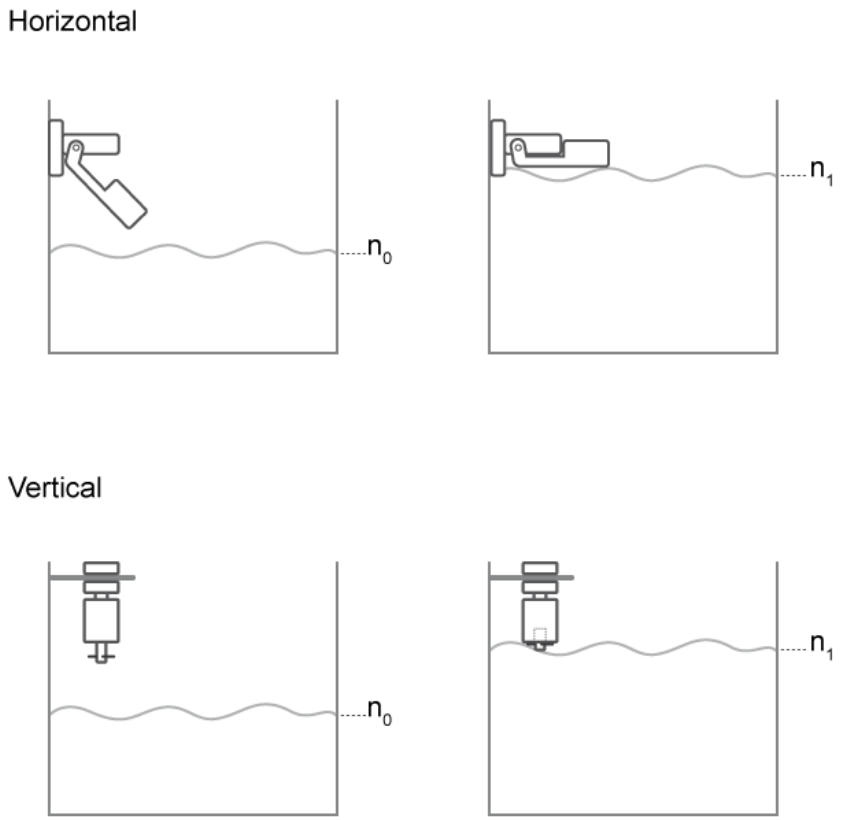

Figure 28: Monitoring the liquid level with PTFA sensors

A code for reading the sensor is shown below:

```
int value;
{
  SensorEventv20.ON();
  delay(10);
  value = SensorEventv20.readValue(SOCKET);
}
```

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

`SOCKET` indicates on which connector the sensor is placed (for this sensor it may be `SENS_SOCKET1`, `SENS_SOCKET2`, `SENS_SOCKET3` and `SENS_SOCKET8`).

4.6. Luminosity Sensor (LDR)

4.6.1. Specifications

Resistance in darkness: 20MΩ
Resistance in light(10lux): 5 ~ 20kΩ
Spectral range: 400 ~ 700nm
Operating temperature: -30ºC ~ +75ºC
Minimum consumption: 0µA*

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

Figure 29: Light sensor LDR

4.6.2. Measurement Process

This is a resistive sensor whose conductivity varies depending on the intensity of light received on its photosensitive part, which must be situated in one of the connectors 1, 2, and 3 depending on the range of luminosity intensity to be measure (1 for measurements in very low light, 3 and 6 for measurements in high luminosity).

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.

A code for reading the sensor is shown below:

```
float value;
{
  SensorEventv20.ON();
  delay(10);
  value = SensorEventv20.readValue(SOCKET, SENS_RESISTIVE);
}
```

`value` is a float variable where the resistance of the in kiloohms will be stored.

`SOCKET` indicates on which connector the sensor is placed (for this sensor it may be `SENS_SOCKET1`, `SENS_SOCKET2` and `SENS_SOCKET3`).

4.7. Presence Sensor (PIR)

4.7.1. Specifications

Height: 25.4mm
Width: 24.3mm
Length: 28.0mm
Consumption: 100µA
Range of detection: 6 ~ 7m
Spectral range: ~ 10µm

Figure 30: PIR presence sensor

4.7.2. 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 be must situated in connector 7.

The 10µm 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 2.0, this is why it is advised to place Waspmote perpendicular to the ground when using the PIR sensor.

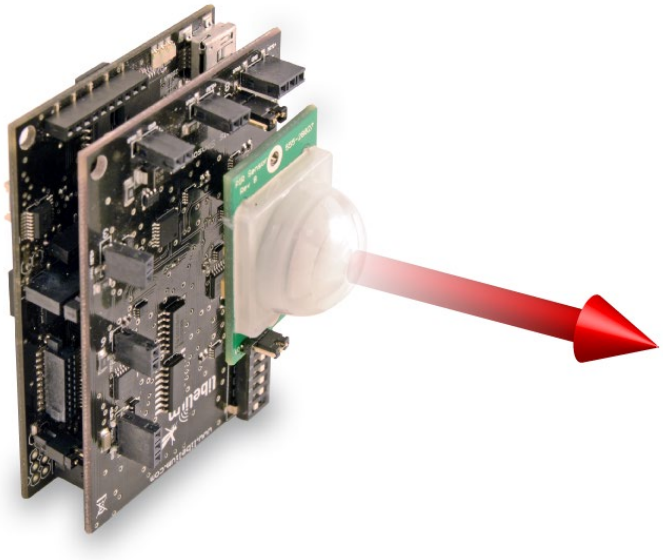

Figure 31: PIR maximun detection range

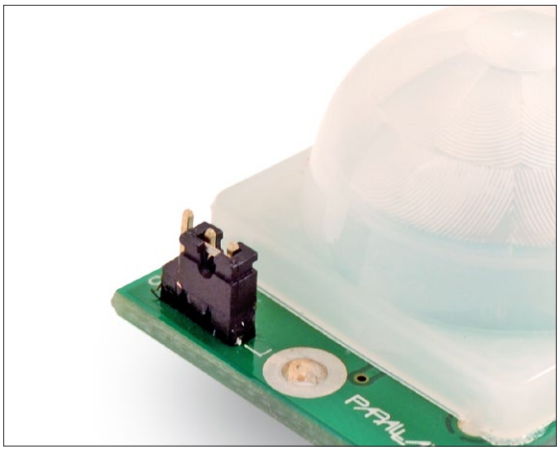

Figure 32: Maximun range jumper configuration

A code for reading the sensor is shown below:

```
int value;
{
  SensorEventv20.ON();
  delay(10);
  value = SensorEventv20.readValue(SENS_SOCKET7);
}
```

`value` is an integer variable where the sensor state (a high value ('1'), that indicates presence, or a low value ('0')) will be stored.

4.8. Hall Effect Sensor (PLA41201)

4.8.1. Specifications

Length: 32mm
Width: 15mm
Thickness: 7mm
Maximum contact resistance (closed): 150mΩ
Minimum contact resistance (open): 100GΩ
Minimum consumption: 0μA*

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

Figure 33: Hall effect sensor

4.8.2. 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 (P6250000) it can be used in applications of monitoring proximity or opening mechanisms. Being a switch sensor it must be placed on one of the connectors 1, 2, 3 or 8.

Figure 34 shows the way of using the sensor in an application of monitoring the opening of doors or windows.

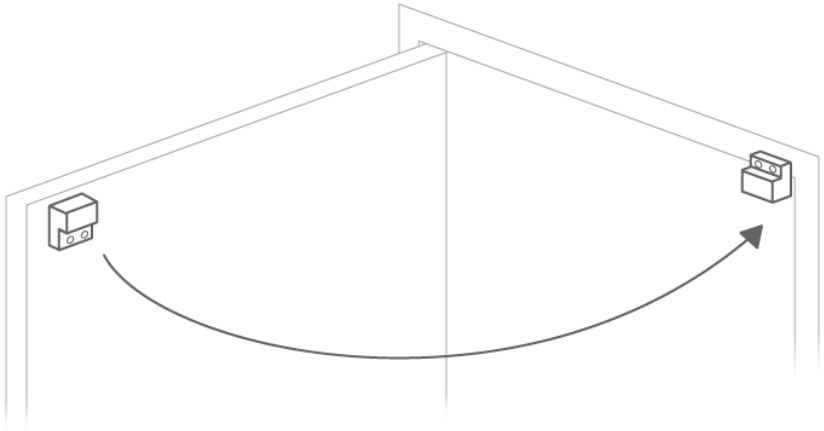

Figure 34: Application of the PLA41202 sensor in monitoring opening mechanisms

A code for reading the sensor is shown below:

```
int value;
{
  SensorEventv20.ON();
  delay(10);
  value = SensorEventv20.readValue(SOCKET);
}
```

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

`SOCKET` indicates on which connector the sensor is placed (for this sensor it may be `SENS_SOCKET1`, `SENS_SOCKET2`, `SENS_SOCKET3` and `SENS_SOCKET8`).

4.9. Liquid Presence Sensor

4.9.1. Specifications

Maximum Switching Voltage: 100V
Operating temperature: +5ºC ~ +80ºC
Detectable liquids: Water
Minimum consumption: 0µA*

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

Figure 35: Liquid Presence sensor

4.9.2. 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 connectors 1, 2, 3 and 8.

Figure 36 shows an image of how the sensor must be placed to detect the presence of liquid.

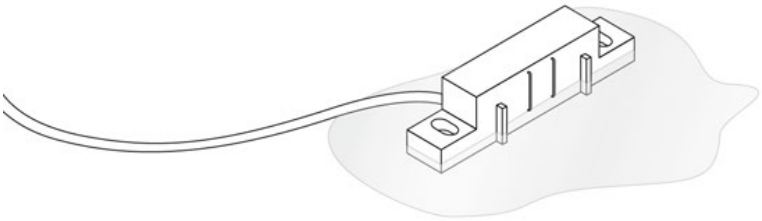

Figure 36: Application of the Liquid Presence sensor for the detection of liquids

A code for reading the sensor is shown below:

```
int value;
{
  SensorEventv20.ON();
  delay(10);
  value = SensorEventv20.readValue(SOCKET);
}
```

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

`SOCKET` indicates on which connector the sensor is placed (for this sensor it may be `SENS_SOCKET1`, `SENS_SOCKET2`, `SENS_SOCKET3` and `SENS_SOCKET8`).

4.10. Stretch Sensor (STRX-04)

4.10.1. Specifications

Resistance of the sensor at rest: 1kΩ per inch
Length: 4 inches (10.16cm)
Resistance of the sensor in tension (50% stretch): Approximately 2kΩ per inch
Minimum consumption: 0μA*

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

Figure 37: STRX-04 sensor

4.10.2. Measurement Process

The STRX-04 is a resistive sensor whose conductivity drops when it is stretched lengthwise, regaining its original resistance once returned to rest, so it is useful in applications such as monitoring of force or variations of distances. It must be handled with care so the contacts on its ends do not come away from the dielectric filament. Being a resistive sensor it can be placed on connectors 1, 2 and 3, preferably on the last one given its range of resistance.

Figure 38 shows an application of the sensor for monitoring the presence of weight.

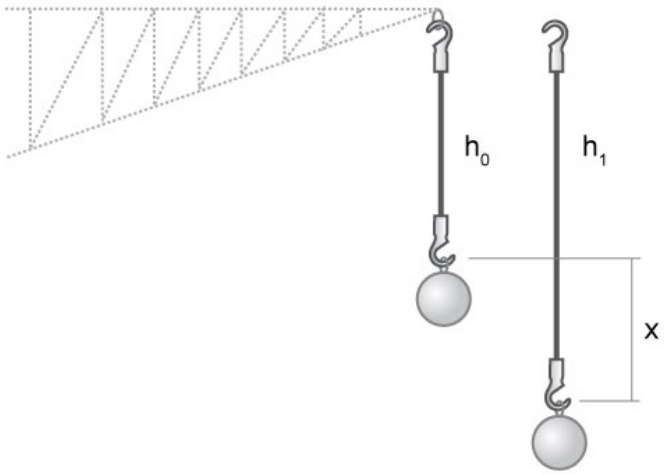

Figure 38: Application of the STRX sensor for monitoring the presence of weight

A code for reading the sensor is shown below:

```
float value;
{
  SensorEventv20.ON();
  delay(10);
  value = SensorEventv20.readValue(SOCKET, SENS_RESISTIVE);
}
```

`value` is a float variable where the resistance of the in kiloohms will be stored.

`SOCKET` indicates on which connector the sensor is placed (for this sensor it may be `SENS_SOCKET1`, `SENS_SOCKET2` and `SENS_SOCKET3`).

4.11. Humidity Sensor (808H5V5)

4.11.1. Specifications

Measurement range: 0 ~ 100%RH
Output signal: 0 ~ 3.0V (25ºC)
Accuracy: <±4%RH (a 25ºC, range 30 ~ 80%), <±6%RH (range 0 ~ 100%)
Supply voltage: 3.3V DC ±3%
Operating temperature: -40 ~ +85ºC
Response time: <15 seconds
Typical consumption: 0.18mA
Maximum consumption: 0.2mA

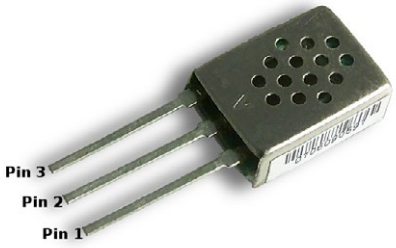

Figure 39: 808H5V6 sensor

4.11.2. Measurement Process

This is an analog sensor which provides a voltage output proportional to the relative humidity in the atmosphere, according to the graph seen in figure 40, with an accuracy of ±6%RH along the whole range and of ±4%RH between 30% and 80%. Being an analog sensor it must be placed on connectors 5 or 6.

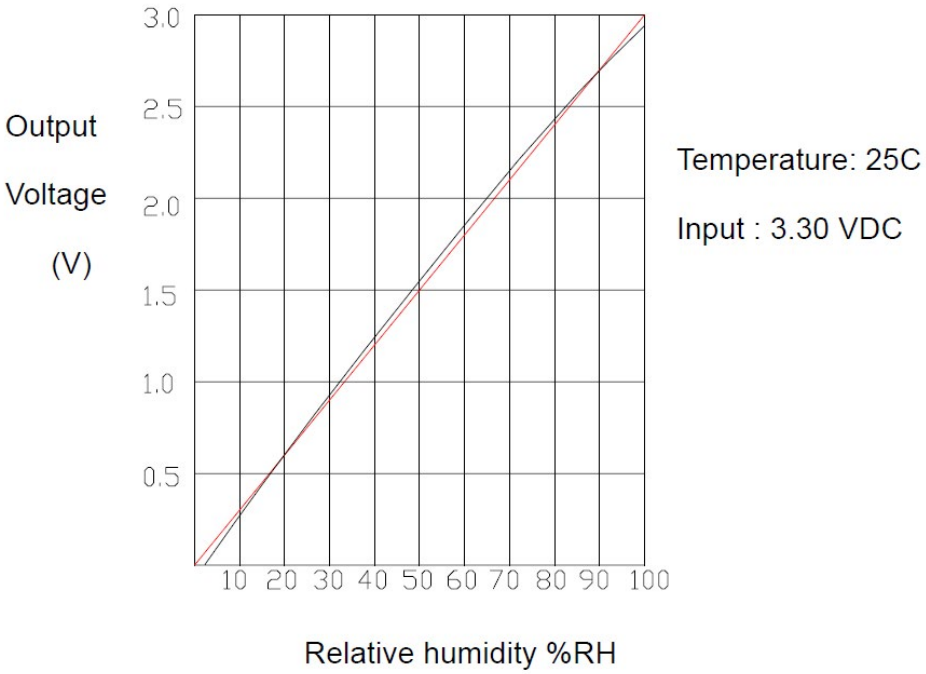

Figure 40: 808H5V6 Humidity sensor output taken from the Sencera Co. Ltd sensor data sheet

A code for reading the sensor is shown below:

```
float value;
{
  SensorEventv20.ON();
  delay(10);
  value = SensorEventv20.readValue(SOCKET, SENS_TEMPERATURE);
}
```

`value` is a float value where the humidity in relative humidity percentage (%RH) will be stored.

`SOCKET` indicates on which connector the sensor is placed (for this sensor it may be `SENS_SOCKET5` and `SENS_SOCKET6`).

4.12. Liquid Flow Sensor (FS100A, FS200 and FS400 from Broiltech)

4.12.1. Specifications

FS100A:

Flow rate: 0.15 ~ 2.5L/Min
Working voltage: +3.3V ~ +24V
Working temperature: -10ºC ~ 120ºC
Pulse number: 3900 pulses/liter
Inlet pipe size: 2mm
Outlet pipe size: 4mm
Accuracy: ±0.5%
Max rated current: 8mA

FS200A:

Flow rate: 0.5 ~ 25L/Min
Working voltage: +3.3V ~ +24V
Working temperature: -10ºC ~ 120ºC
Pulse number: 450 pulses/liter
Pipe connection: ½"
Accuracy: ±1%
Max rated current: 8mA

Figure 41: Liquid FS200A Flow sensor

FS400:

Flow rate: 1 ~ 60L/Min
Working voltage: +3.3V ~ +24V
Working temperature: -10ºC ~ 120ºC
Pulse number: 390 pulses/liter
Pipe connection: 1"
Accuracy: ±2%
Max rated current: 8mA

Figure 42: Liquid FS400 Flow sensor

4.12.2. 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 0Hz and 100Hz, is directly read through one of the digital input/output pins of the microcontroller. This sensor must be connected in socket 8, with the selection jumper placed in the pull up position.

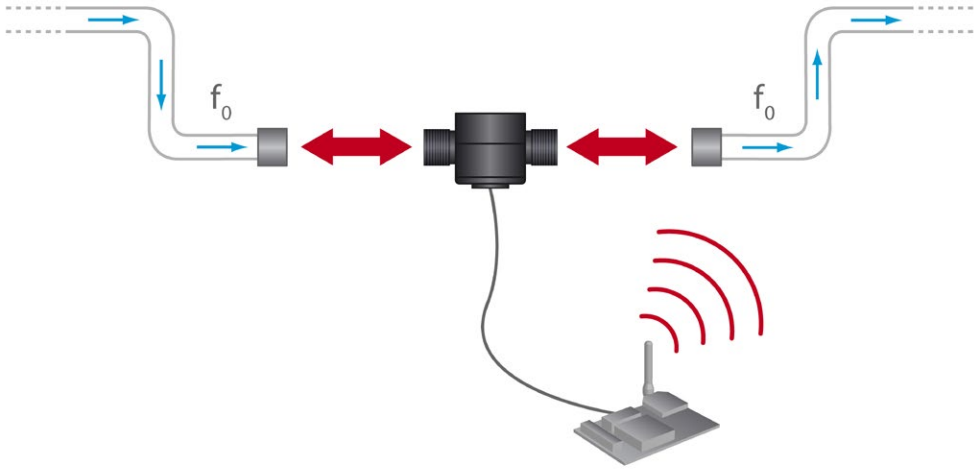

Figure 43: Example of application with the liquid flow sensor

A code for reading the sensor is shown below:

```
float value;
{
  SensorEventv20.ON();
  delay(10);
  value = SensorEventv20.readValue(SENS_SOCKET8, TYPE);
}
```

`value` is a float value where the flow rate in liters per minute (l/min) will be stored.

TYPE indicates the type of sensor connected (`SENS_FLOW_FS400`, `SENS_FLOW_FS200` or `SENS_FLOW_FS100`).

4.13. Design and connections

The Waspmote Events Sensor Board v2.0 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 Waspmote manual. In the following sections the different board connectors are described so the connectivity and sensor integration can be fully taken advantage of.

4.13.1. Connectors 1, 2 and 3

Connectors 1, 2 and 3 (see the diagram of components in section "Hardware. Specifications") consist of two 2.54mm pitch female pins for the sensor's connection, the first of which is connected to the 3.3V power through the manual switch, while the second is connected to ground through a pull-down resistance. The value of this resistance as well as the comparator input to which the sensor output leads are the only points which the three connectors functionally differentiate. The output of connectors 1 and 3, the first connected to a pull-down resistance of 560kΩ and the second 10kΩ, access the comparator through its positive input, which involves activation of the interruption when the voltage value in one of them surpasses the value defined in the threshold; while connector 2, whose output is connected to a pull-down resistance of 100kΩ, accesses the comparison stage through the negative input of the comparator, which implies that the interruption will trigger when the voltage value provided by the sensor falls below a fixed threshold on the potentiometer. An image of the diagram of these connectors' adaptation can be seen in figure 44.

The output of each connector, taken after the pull-down resistance, joins to an analog microprocessor pin at the input of a voltage comparator, the comparison threshold being fixed through a 100kΩ digital potentiometer (digipot) (it can take values from 0kΩ to 100kΩ), whose variable terminal is connected to the comparator's negative input and whose fixed terminals are connected to the 3.3V supply voltage and ground respectively. The comparator's output is connected to the OR gate. The microprocessor's input which the connectors access are analog input 1 (ANALOG1) for connector 1, analog input 2 (ANALOG2) for connector 2 and analog input 4 (ANALOG4) for connector 3.

These connectors have been installed to integrate both resistive sensors (such as the Flexiforce PS-02, the FLX-01-H, the STRX-04 and the LDR) as well as switch sensors (such as the ASLS-2, the AG2401-1, the AG3011-1, the PTFA3415, the PTFA0100, the PTFA1103 and the PLA41201), on condition that they are normally open if connected to connectors 1 or 3 and closed if connected to connector 2.

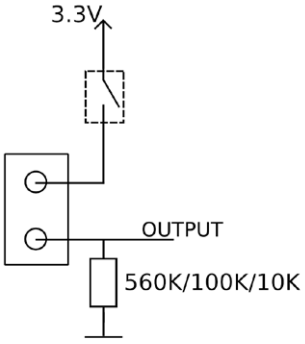

Figure 44: Diagram of connectors 1, 2 and 3

You can find complete example codes for reading the different sockets with resistive sensors in the following links:

http://www.libelium.com/development/waspmote/examples/ev-1-socket-1-reading-with-resistive-sensor

http://www.libelium.com/development/waspmote/examples/ev-2-socket-2-reading-with-resistive-sensor

http://www.libelium.com/development/waspmote/examples/ev-3-socket-3-reading-with-resistive-sensor

4.13.2. Connector 4

Connector 4 (see the diagram of components in section "Hardware. Specifications") consists of a strip of two 2.54mm pitch female pins. The first pin is connected to the board's ground, while the second is connected to the sensor's output by a 10kΩ series resistance, both connected through a diode. The diagram of the pins' connection can be seen in figure 45. The sensor's output is compared with a fixed threshold by the user through a 100kΩ digipot using a comparator, which is connected to the sensor through its positive input and whose output acts on the OR gate. In the same way, the connector's output is attached to the microprocessor through its analog 3 input (ANALOG3).

This connector has been designed to accommodate the piezoelectric sensors (such as the PZ-01 and the PZ-08) which do not need power voltage, only a connection to ground and an analog reading of its output.

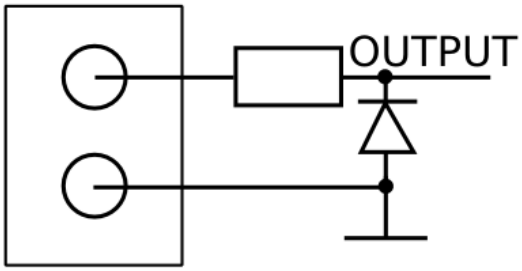

Figure 45: Diagram of connector 4

You can find a complete example code for reading the piezoelectric sensor on socket 4 in the following link:

http://www.libelium.com/development/waspmote/examples/ev-4-socket-4-for-piezoelectric-sensors-reading

4.13.3. Connectors 5 and 6

Connectors 5 and 6 (see the diagram of components in section "Hardware. Specifications") are formed by a strip of three 2.54mm pitch female pins which are directly connected to the 3.3V supply (through the manual switch), to ground and to the connector's output. These outputs are carried directly to the analog 6 input (ANALOG6) and analog 7 input (ANALOG7) from the threshold comparator whose output activates the OR gate (positive comparison for connector 5 and negative for connector 6). The connector's power and ground pins are decoupled by a 100nF capacitor. A diagram of the connectors' structure can be seen in figure 46.

This connector allows the connection of **analog sensors** to the Events Sensor Board 2.0 (such as the MCP9700A or the 808H5V6) which provide a voltage output and only requires power and ground.

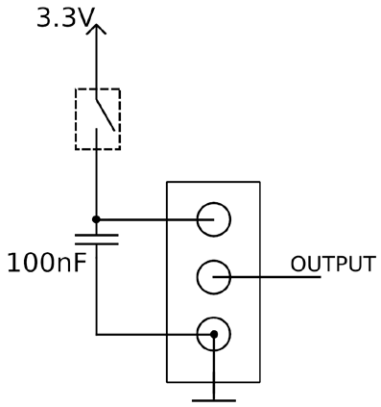

Figure 46: Diagram of connectors 5 and 6

You can find complete example codes for reading the temperature and humidity sensors on sockets 5 and 6 in the following links:

http://www.libelium.com/development/waspmote/examples/ev-5-socket-5-reading-with-temperature-sensor

http://www.libelium.com/development/waspmote/examples/ev-6-socket-6-reading-with-humidity-sensor

4.13.4. Connector 7

Connector 7 (see the diagram of components in section "Hardware. Specifications") consists of a strip of three 2.54mm pitch female pins connected to ground, power supply (through manual switch 7) and the OR gate and microprocessor inputs. The diagram can be seen in figure 47.

The connector's output is directly connected to the microprocessor's digital input DIGITAL5.

This connector has been installed for the integration of **digital sensors** (such as the PIR sensor) which only require connection to ground and the 3.3V supply and whose output can be read by a digital pin in the microprocessor.

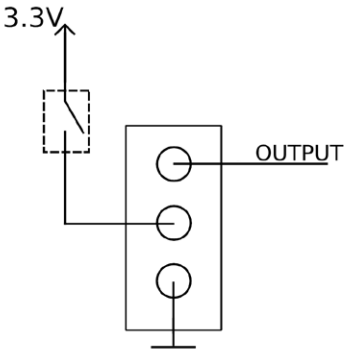

Figure 47: Diagram of connectors 7

You can find a complete example code for reading the PIR sensor on socket 7 in the following link:

http://www.libelium.com/development/waspmote/examples/ev-7-socket-7-reading-with-pir-sensor

4.13.5. Connector 8

Connector 8 (see the diagram of components in section "Hardware. Specifications") is formed by two strips of two and three 2.54mm pitch female pins, the first of which allows connections to ground, the 3.3V supply from the manual switch and the sensor's output, connected to power through a 100kΩ pull-up resistance. The pins on the second strip are connected to ground and the sensor's output, connected to ground through a pull-down resistance. The connector's output which will access the OR gate and the analog pin 5 (ANALOG 5) is selected from the previous two through a jumper. A diagram of the adaptation circuit can be seen in figure 48.

This connector allows the connection of switch sensors (such as the ASLS-2, the AG2401-1, the AG3011-1, the PTFA3415, the PTFA0100, the PTFA1103 and the PLA41201). The two outputs available, connected to ground and power through pull-down and pull-up resistances respectively, allow the use of sensors which are normally both closed and open. They also can be used for the integration of digital sensors (such as the PIR), provided the possible influence that the pull-up or pull-down resistances can have on the output has been considered.

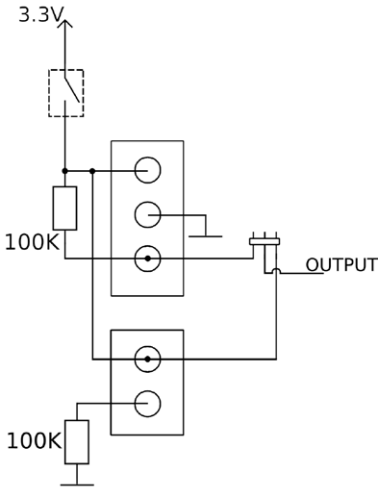

Figure 48: Diagram of connector 8

You can find a complete example code for reading the Flow sensor and the liquid level sensor on socket 8 in the following links:

http://www.libelium.com/development/waspmote/examples/ev-8-socket-8-reading-with-liquid-level-sensor

http://www.libelium.com/development/waspmote/examples/ev-9-socket-8-reading-with-flow-sensor

4.13.6. Sockets for casing

In case the Events Sensor Board 2.0 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.

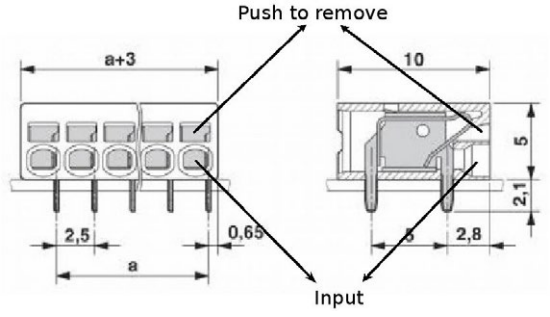

Figure 49: Diagram of a socket extracted from the Phoenix Contact data sheet

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.

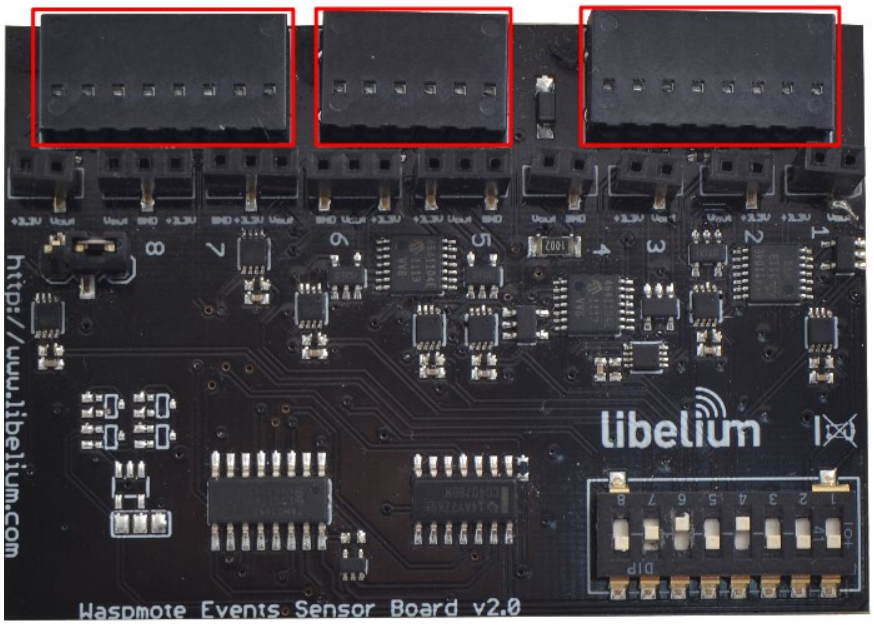

Figure 50: Sockets for casing applications

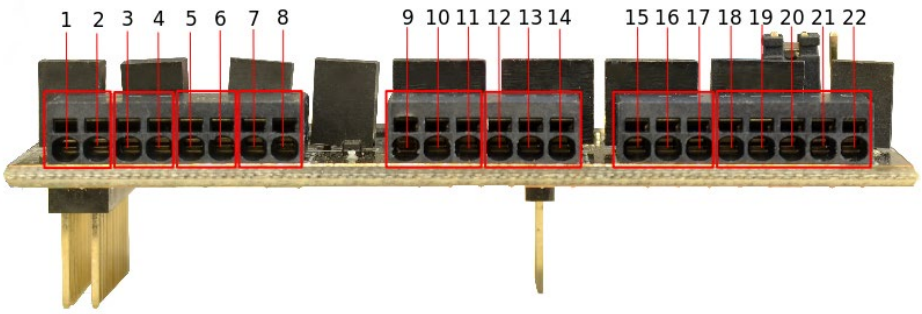

Figure 51: Pin correspondence between the sockets and the sensors

Sensor	Pin	Function
Socket 1	1	Output
	2	VCC (3.3V)
Socket 2	3	VCC (3.3V)
	4	Output
Socket 3	5	Output
	6	VCC (3.3V)
Socket 4	7	Output
	8	GND
Socket 5	9	GND
	10	Output
	11	VCC (3.3V)
Socket 6	12	VCC (3.3V)
	13	Output
	14	GND
Socket 7	15	Output
	16	VCC (3.3V)
	17	GND
Socket 8	18	VCC (3.3V)
	19	GND
	20	Output (pull up)
	21	Output (pull down)
	22	VCC (3.3V)

5. Board configuration and programming

5.1. Hardware configuration

The different sensors must be connected on the connector's pin strips. The pin function is printed on the board: power supply (+3.3V), circuit's ground (GND) or connector's output signal (Vout). Although the output of many of the sensors can be connected indiscriminately to the input pins, for both resistive and switch sensors, the user must maintain precaution to avoid short-circuits and erroneous connections which may damage both the sensor and board, following the instructions in this manual and the sensor's data sheet, and consulting the distributor with any queries.

In addition to the sensor's connections, the Waspmote Events Sensor Board 2.0 also includes two components which must be manually configured by the user: a manual switch for controlling the power of each sensor and a jumper for the choice of output from one of the connectors. A manual switch has been chosen because the insertion of more digital elements would greatly limit the maximum number of sensors which can be connected to the board, as long as increase the consumption of the board.

Firstly, the power of each sensor must be enabled or disabled manually. A three-position 8 switches socket has been installed for this purpose, from which each of the connectors is controlled. As seen in image 52, placing the switch opposite the identification number enables power to all the connector components and any element attached to it, while in any of the other two positions the power input of the different devices are unconnected. Each one of these switches, for its respective connector, allows controlling the power of the sensor, of the electronics enclosed and, if included, the comparator and voltage on the end of the digital potentiometer from which the comparison threshold is adjusted.

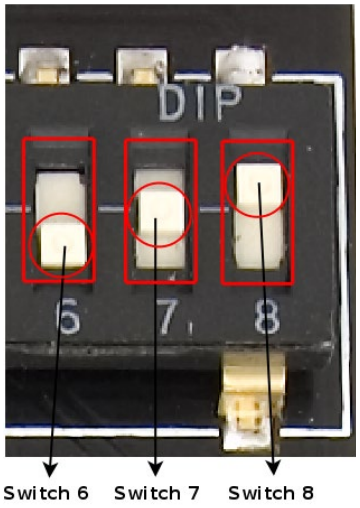

Figure 52: Three switches in the on position (switch 8) and off (switches 7 and 6)

Secondly, in the case of connector 8 a jumper is used to choose the output which accesses the OR gate, to allow the use of switch sensors which are normally closed as well as those which are normally open. If the jumper connects the central pin with the left one, see Figure 53, the sensor's output is connected to the 3.3V supply through a pull-up resistance, which means that connecting the sensor switch between this pin and the ground pin will produce an activation signal in the OR gate when the sensor changes from closed to open, meaning that the sensor's normal status must be closed. Conversely, if the pin connected by the jumper to the central pin is the right one, the output signal selected is connected to the circuit ground through a pull-down resistance. Therefore, the activation signal of the interruption is generated when the sensor, connected between this output and the 3.3V power supply, passes from open to closed, so the sensor normally must be open.

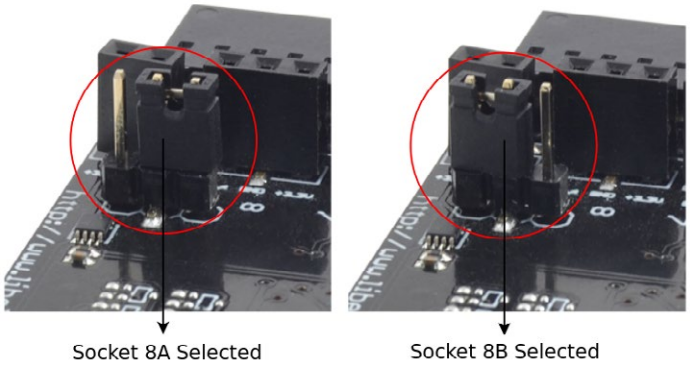

Figure 53: Selection jumper positioned to enabled the pulled down output (socket 8A) or the pulled up output (socket 8B)

5.2. API

The Waspmote's Event Sensor board v2.0 has a library in which the necessary instructions are included for handling and management of interruptions and the sensors built in to it.

When using the Event Sensor Board v20 on Waspmote PRO, remember it is mandatory to include the SensorEventv20 library by introducing the next line at the beginning of the code:

```
#include <WaspSensorEvent_v20.h>
```

Next, the different functions that make up the library are described:

SensorEventv20.ON()

Turns on the sensor board by activating the 3.3V supply line.

SensorEventv20.OFF()

Turns off the sensor board by cutting the 3.3V supply line.

SensorEventv20.setBoardMode(MODE)

This function is used to manage the power applied to the board. Assigning the value SENS_ON to the MODE variable, the board's switches are activated, which allows the passage of the 3.3V voltage, while allocating the value SENS_OFF both switches are disconnected, switching off the power.

SensorEventv20.setThreshold(SENSOR, THRESHOLD)

The setThreshold function is used to configure the comparison threshold value which regulates the interruption trigger. In the SENSOR variable, the connector corresponding to the sensor whose comparison threshold is to be changed is introduced, taking values from SENS_SOCKET1 to SENS_SOCKET6, while 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.

SensorEventv20.readValue(SENSOR)

The readValue instruction captures the value of the sensor's output and stores it as a floating point in the variable to which the function has been assigned. The connectors whose output is to be measured is introduced through the SENSOR variable, which allows values from SENS_SOCKET1 to SENS_SOCKET8. The captured value is analog, except in the case of connector 7 (SENS_SOCKET7) whose output is connected to a digital input of the Waspmote microprocessor.

SensorEventv20.readValue(SENSOR, TYPE)

The readValue may be called introducing a second parameter in order to also perform the conversion from the voltage read at the output of the sensor into the units available for the given sensors. This allows the user to assign the following values to the parameter TYPE:

- SENS_RESISTIVE: Used when a resistive sensor has been placed on connectors 1, 2 or 3, with this parameter the function will return the resistance of the sensor in kiloohms.
- SENS_FLOW_FS100: The readValue function performs a reading of the frequency of the FS100 flow sensor connected on socket 8 and returns the water flow measured.
- SENS_FLOW_FS200: The readValue function performs a reading of the frequency of the FS200 flow sensor connected on socket 8 and returns the water flow measured.
- SENS_FLOW_FS400: The readValue function performs a reading of the frequency of the FS400 flow sensor connected on socket 8 and returns the water flow measured.
- SENS_TEMPERATURE: Reads the value of the MCP9700A temperature sensor on connectors 5 or 6 and returns its value in Celsius degree (ºC).
- SENS_HUMIDITY: Reads the value of the 808H5V6 humidty sensor on connectors 5 or 6 and returns its value in relative humidity percentage (%RH).

SensorEventv20.attachInt()

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.

SensorEventv20.detachInt()

Complementing the previous function, the aim of dettachInt 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 implementation the mote will ignore any interruption which arrives from the sensors until the attachInt instruction is activated again.

SensorEventv20.loadInt()

The instruction loadInt is used to read the content of the shift register and store its output in an integer variable called SensorEventv20.intFlag, 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 variable which represents the connector in which the sensor is placed (SENS_SOCKET'X', x=1…8) and the intFlag variable.

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

1. The board is switched on using the function **SensorEventv20.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 **SensorEventv20.setThreshold**.
4. Enable interruptions from the board using the function **SensorEventv20.attachInt**.
5. Put the mote to sleep with the functions **PWR.sleep or PWR.deepSleep**.
6. When the mote wakes up, disable interruptions from the board using function **SensorEventv20.detachInt**.
7. Load the value stored in the shift register with function **SensorEventv20.loadInt**.
8. Process the interruption.
9. Return to step 4 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 Waspmote Frame format. You can find more information about how to handle the Waspmote Frame in the Programming Guide in the Development Section of the Libelium website.

Next an example code is shown in which interruptions are enabled, Waspmote is placed in a low consumption mode and the arrival of an interruption is expected. The shift register is read and if it has been generated by the sensor placed in connector 1, it reads its value and sends a warning message through XBee:

```
/* ------------Events 2.0 Sensor Board example---------------

 www.Libelium.com
 */

// Inclusion of the Events Sensor Board v20 library
#include <WaspSensorEvent_v20.h>

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

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

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

// Variable to store the sensor read value
int value = 0;

void setup()
{
  // Turn on the Events Sensor Board
  SensorEventv20.ON();
  // Init the RTC
  RTC.ON();
  // Configure the sensor threshold
  SensorEventv20.setThreshold(SENS_SOCKET1, 1.5);
}

void loop()
{
  // Enable interruptions from the board
  SensorEventv20.attachInt();
  // Put the mote to sleep
  PWR.sleep(UART0_OFF | UART1_OFF | BAT_OFF | RTC_OFF);
  // Disable interruptions from the board
  SensorEventv20.detachInt();
  // Load the interruption register
  SensorEventv20.loadInt();
```

```
  // Init XBee
  xbee802.ON();
  // Set parameters to packet:
  packet=(packetXBee*) calloc(1,sizeof(packetXBee));
  packet->mode=BROADCAST;

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

  // Compare the interruption received with the sensor identifier
  if (SensorEventv20.intFlag & SENS_SOCKET1)
  {
    // Read the sensor
    value = SensorEventv20.readValue(SENS_SOCKET1, SENS_RESISTIVE);

    // Add the read value to the frame
    frame.addSensor(SENSOR_VBR, value);

    // Transmit the response
    if (value > 300)
    {
      frame.addSensor(SENSOR_STR, "Interrupt level 1");
    }
    else
    {
      frame.addSensor(SENSOR_STR, "Interrupt level 2");
    }
  } else
  {
    frame.addSensor(SENSOR_STR, "Not interrupted from the sensor");
  }

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

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

The files of the sensor board itself are: **WaspSensorEvent_v20.cpp, WaspSensorEvent_v20.h**

They can be downloaded from: **http://www.libelium.com/development/waspmote/sdk_and_applications**

6. API Changelog

Function / File	Changelog	Version
`#include`	Remember to include the WaspSensorEvent_v20 library in the top of your pde	v.31 → v1.0
`SensorEventv20.ON()`	New function to turn on the board	v.31 → v1.0
`SensorEventv20.OFF()`	New function to turn off the board	v.31 → v1.0
`SensorEventv20.readValue()`	Added the possibility of including the type of sensor integrated in order to calculate the conversion from volts into the parameter units.	v.31 → v1.0

7. Consumption

7.1. Power control

The power of the Events Sensor Board 2.0 is drawn from the Waspmote 3.3V 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 Sensor Board 2.0 power (see Waspmote manual for more details about the switches to manage the mote's power and the appropriate section in the WaspmotePWR API manual's library for more information on its handling). This means that, independently of the board's internal mechanisms, Waspmote will decide at any moment if the board is on or off.

Inside the board itself, the OR logic gate, the shift register and the three digital potentiometers 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 of 3.6µA. On the other hand, the power of each of the connectors can be switched off or connected manually by its corresponding switch, inside the socket situated in the upper right part of the board which can be seen in the image of components in section "Hardware. Specifications". This allows that only the electronics associated with the connectors in which a sensor has been placed is powered.

As explained in section Hardware Configuration, each one of the switches remains closed and allows powering the connector when it is in the left position (the furthest from the connector's indicating number), any of the other positions causes a cut in the current to the circuit. Each one of the switches allows or denies power supplying to the adaptation electronics of the sensor, the voltage comparator and the positive end of the potentiometer through which the comparison threshold is fixed.

7.2. Consumption table

In the following table the consumption of the board is shown, from the constant minimum consumption (fixed by the permanently active components) together with one of each of the connectors which may be selected by the 8-way switch and an estimate of consumption through the shift register which the sensors use to store their values when they reach the threshold. Remember that the board's power can be completely disconnected, reducing consumption to zero, using the 3.3V main switch disconnection command included in the library.

	Consumption
Minimum (constant)	3.6µA
Connector 1	32µA - 37µA
Connector 2	32µA - 64µA
Connector 3	32µA - 360µA
Connector 4 (without sensor)	32µA
Connector 5 (without sensor)	32µA
Connector 6 (without sensor)	32µA
Connector 7 (without sensor)	0µA
Connector 8	0µA - 33µA
Reading register (<20ms)	150µA

To know the total consumption of the board with the selected sensors the consumption of each sensor must be added to that of the connector upon which it will be connected, except in the case of connectors 1, 2, 3 and 8 for switch and resistive sensors, whose consumption would already be within the range provided in the table. This consumption may be consulted and calculated from the information in the sensor section where all the characteristics are described.

7.3. Low Consumption Mode

The Waspmote Events Sensor Board 2.0 has been designed to have the least consumption possible. For this, the only recommendations which the user must try to follow are the following:

- **Switch off the manual switch of those sensors which are not to be used**

  This is not only relevant for energy saving, but it is also necessary to prevent the possibility of a false response in the mote caused by a random voltage in the connector which is not being used but remains active.

- **Use the Waspmote low consumption modes**

  One of the main advantages of Waspmote, 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 Waspmote manual and the WaspmotePWR 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 Waspmote when any event occurs.

8. Documentation changelog

- 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 "Waspmote" encompasses both the Waspmote device itself as well as its modules and sensor boards.
- When Waspmote 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.