Movee
LoRa motion sensor
User Guide

LoRa Motion sensor for public or private networks

Revision 1.05
Product version: 3.6
SW version: 1.8.5
LoRaWan version: 4.3.1

éolane

Table of contents

I. Introduction

I.1. Overview

Movee is a low power LoRa™ Motion sensor for indoor and outdoor industrial applications (asset management, predictive maintenance…), coupled with a temperature sensor for environmental conditions, and a push button for activation or debug frame emission.

Movee implements several algorithms, which can synthesize the MPU (Motion Processing Unit) data into usefull information, lowering the data transmitted on the LoRa™ Network:

- ALIVE = sends an alive frame (battery level + temperature information) periodically
- TEMPERATURE = sends the ambient temperature either periodically or if a threshold min or max has been crossed
- SHOCK = sends the acceleration information if the configured threshold has been crossed
- TILT = sends the orientation information if the orientation threshold has been crossed
- ORIENT = sends the orientation information either periodically, or if the orientation threshold has been crossed
- MOTION = sends the information (still/in motion) either periodically, or when the activity threshold has been crossed. It can also send the activity (motion/stillness) duration.
- ROTATION = sends the number of turns periodically
- VIBRATION = sends the detected vibration frequency for measures above a defined amplitude threshold – Available on SW version 1.6 and above

The expected battery life is 7 years (for 4 frames per day), and 6 years (for Shock + Motion detection algorithms)

A user interface is available in order to set the parameters for the algorithms (algorithm activation, detection threshold, min, max value…)

Movee also comes in an industrial form factor, with a ruggedized 75x50x22mm casing, IP68 compliant, with multiple fastening options.

1.2. Key Features

Product Application

Industry Industrial equipment	M2M / Command & control Preventive and predictive maintenance Asset management Geotracking Smart Factory / Traceability
Defence Civil Security Naval	Security and tracking Monitoring of hazardous materials Preventive maintenance
Telecom Multimedia	LoRaWAN Public Network LoRa Network coverage test

Main Features

- LoRa motion sensor (accelerometer & gyroscope) with ambient temperature sensor and push button
- LoRaWAN 1.0 compatible with public LoRa networks
- 7 years battery life (4 frames per day)
- 6 years battery life (for Shock + Motion detection algorithms)
- IP68 case (75x22x50mm)
- Operating temperature: -30°C to +70°C
- Fastening: Screws, Rivet, hose clip, Glue, Adhesive...
- PC UI for programming and calibration

TECHNICAL SPECIFICATIONS

Performances

- Drop detection (+/-1g)
- 3-axis shock detection (+/-1mg up to +/-16g)
- 3-axis tilt and orientation (+/-1°)
- Vibrations measurement and detection (min 7Hz)
- Temperature sensor (+/-1°C)

Hardware

- 6-axis accelerometer and gyroscope
- Temperature sensor
- Silicon Labs Cortex-M3 microcontroller EFM32
- Flash: up to 256KB
- RAM: up to 32KB
- Semtech Sx1272 LoRa™ transceiver
- 2600mAh battery (non rechargeable)
- 1x RGB LED
- 1x Push button

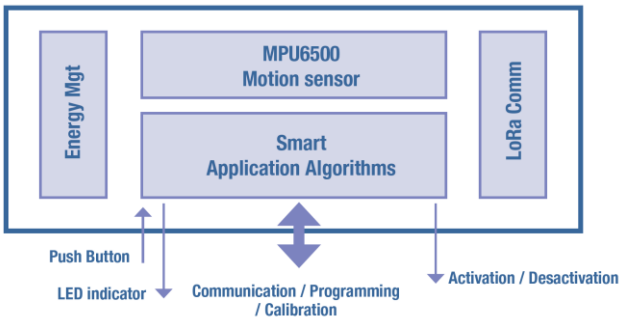
Protocol - Network

- LoRa™ SF6-SF12
- LoRa™ 1.0 Class A, Class C (compatible)
- LoRa™ sensitivity: -137 dBm

Casing

- CE marking
- Polyamide 6.6 (PA 66) material
- Indoor & outdoor use (IP68)
- 2m drop resistant
- Dimensions: 75mmx50mmx22mm
- Fastening: Screws, Rivet, hose clip, Glue, Adhesive...
- Lexan front panel (customizable)
- Storage temperature: -40°C to +85°C

Contact : wireless@eolane.com



ADDITIONAL SERVICES

Software - Firmware

- Custom algorithm

Antenna design

- Ceramic antenna

Casing

- Custom Lexan
- Magnetic fastening

ORDERING INFORMATION

Packaging	HW configuration	Part Number
Unitary	Accelerometer + Gyroscope + LoRa 1.0 + non rechargeable battery +IP68 + éolane Lexan	CS-10000A0U
Box (50)	Accelerometer + Gyroscope + LoRa 1.0 + non rechargeable battery +IP68 + éolane Lexan	CS-10000A0B

II. Functional Overview

2.1. Manufacturing configuration

2.1.1. Device default configuration

When delivered, the device will be OFF. To switch the device ON, please see *§2.2.1 Turn the device ON*.

The default LoRa configuration is OTAA (Over The Air Activation, with DevEUI, AppEUI, AppKey contained in the QR code datamatrix of the product sticker, see *§2.1.2 QR code Sticker*)

And the default algorithm configuration is:

- ALIVE: 1 alive frame sent every hour
- SHOCK: threshold set at 5000mG on each axis, frame sent if threshold is crossed.

2.1.2. QR code Sticker

The sticker under the device contains the coordinate system used by the MPU, regulatory information and device configuration details coded in the datamatrix of the QR code:

Datamatrix content (77 digits for P/N < 3.6 ; 45 for P/N >= 3.6):

- 16 digits: AppEUI (e.g. 70B3D531C0001190)
- 16 digits: DevEUI (e.g. 70B3D531C0001242)
- 32digits: AppKey (B9937B9BC6E4F3EFE9B3437107F84EA3) => Only for for P/N < 3.6
- 4 digits: manufacturing Year on 2 digits (e.g. 17 for 2017) and Week on 2 digits (e.g. 03 for W03)
- 9 digits: Product serial Number (e.g. 123456789)

Datamatrix content example:
For P/N < 3.6

70B3D531C000119070B3D531C0001242B9937B9BC6E4F3EFE9B3437107F84EA31703123456789

For P/N >= 3.6

70B3D531C000119070B3D531C00012421703123456789

2.2. Device interaction

2.2.1. Turn the device ON

OTAA configuration (default)

To turn the device ON, press the push button once, the LED will blink as follows:

MPU OK

<200ms

If MPU is KO => Red LED:

Flash OK

<200ms

If flash is KO => Red LED:

Join Request Transmitted

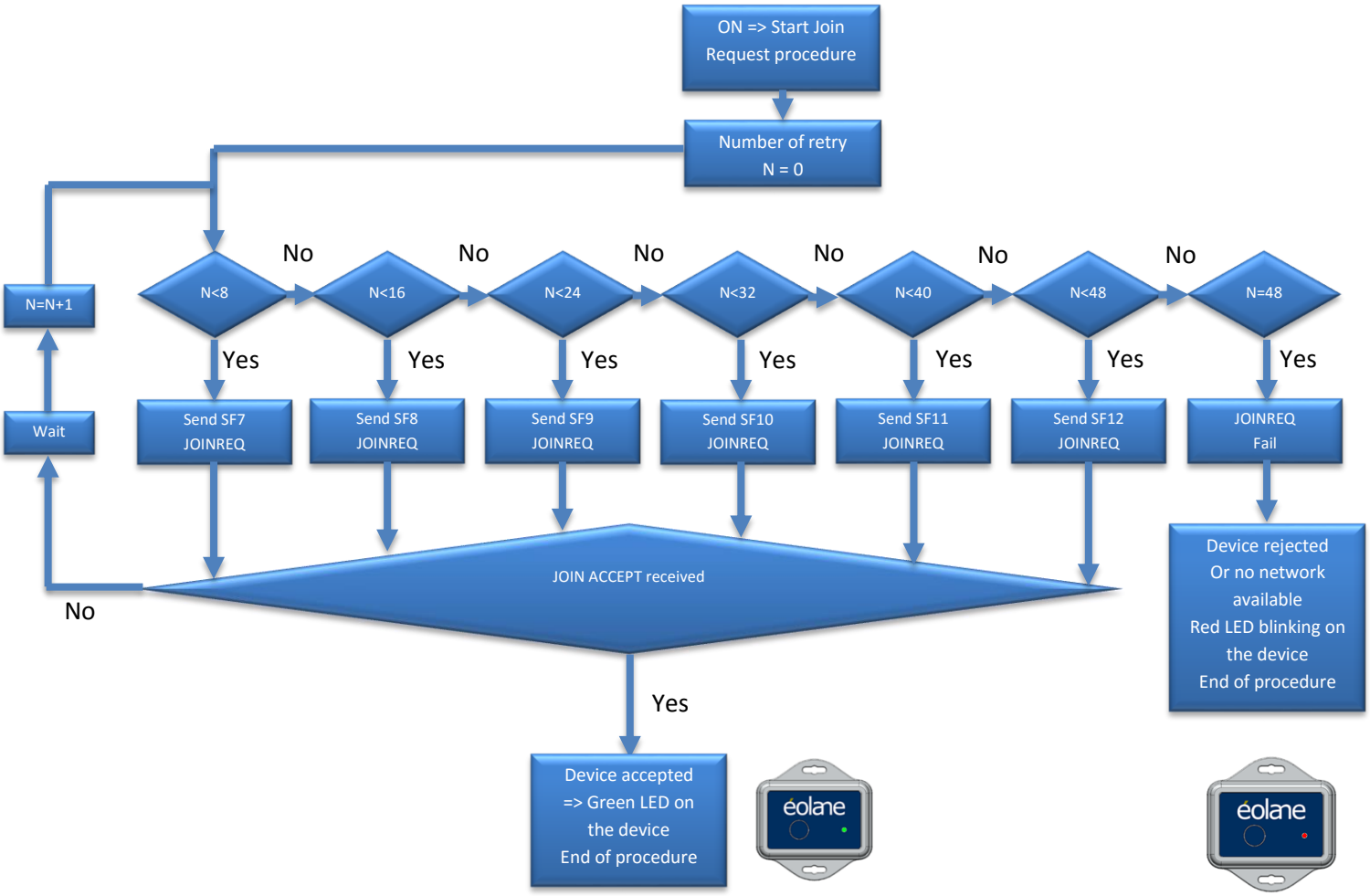
~5s

Join Accept received

<1s

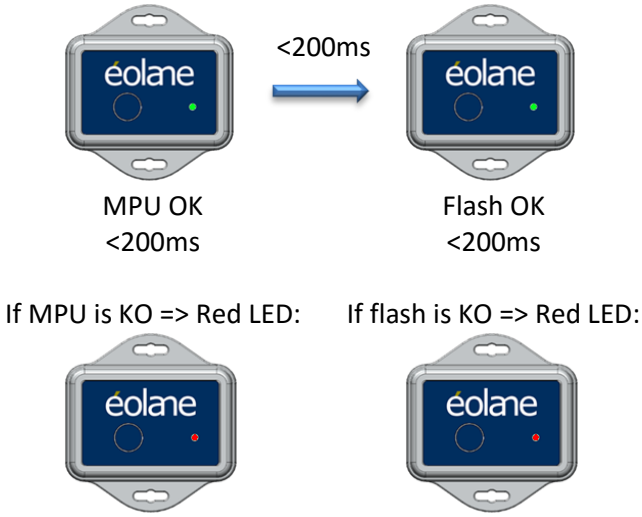
If Join Accept is not received after multiple retries=> Blinking Red LED

The Join Accept reception might take some time, depending on the Network signal strength. The following figure describes the back off procedure when no Join Accept has been received:



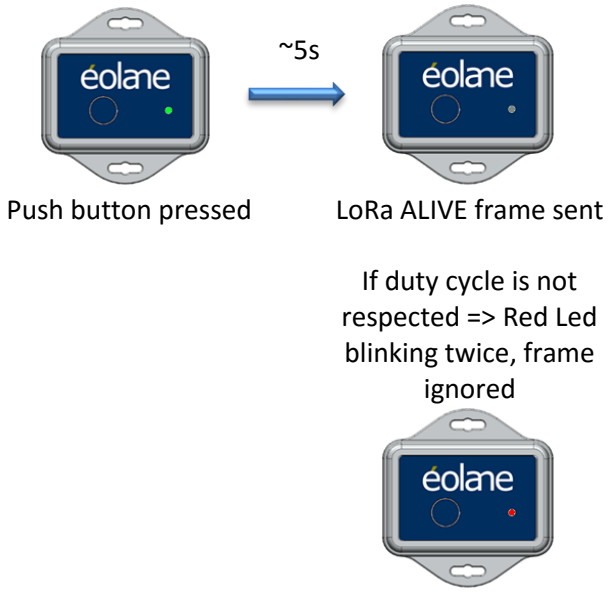
ABP configuration – NOT AVAILABLE in v1.7 SW

When in ABP mode there is no join procedure, the device starts sending messages directly to the gateway.



2.2.2. Send an ALIVE frame

To send an ALIVE frame (when the device is ON), press the push button for more than 1s (and less than 7 seconds), the LED will stay green for 2s, and the LED will blink green when the frame is sent.



2.2.3. Duty cycle limitation

The LoRaWAN™ protocol is based on the 868MHz frequency band, which requires that the devices using this band can't use more than 1% of the bandwidth.

The embedded LoRaWAN stack will calculate the duty cycle conformity.

Providing that the allowed bandwidth has already been used (e.g. by the OTAA Join Request/Join Accept procedure just after startup), the device will not be able to send LoRaWAN frames as long as the duty cycle calculation at the LoRaWAN™ stack level will not release the transmit permission.

Once the transmit permission is granted by the LoRaWAN™ stack, transmissions will resume.

Information can then be lost due to the duty cycle limitation. For example, a shock is detected, a shock frame shall be sent, but the duty cycle has not been respected. As long as the transmission permission is not granted, other events (alive frame requested by the push button, tilt detection…) will not be recorded.

When a frame is dismissed because of the duty cycle limitation, a red LED blinks on the device:

2.2.4. Turn the device OFF

To turn the device OFF, press the push button for more than 7 seconds. The RGB LED will go from Green to Yellow to Red. Once the RGB LED is Red, release the push button, the device is OFF.

2.3. Tools

The device can be configured with a Windows compliant application called “MoveeConfigurator”.

You can download the tool on our support cloud platform:
https://transfer.eolane.com/public.php?service=files&t=7bbed3de4307af52e06302090c565373

Password: eolane

See *§3.6Movee Configurator user interface* for detailed user interface manual

III. User Guide

3.1. Operating modes

3.1.1. NORMAL mode

Normal mode is the default mode entered by the device upon power-on / reset

3.1.2. SERVICE mode

Service mode is entered upon a downlink command reception to enter service mode.

In this mode, all the running algorithms are stopped, and a service frame is transmitted every minute (with eventual duty cycle limitation), in order to speed up the parameters reception through DownLink (DL) frames. The LED alerts that the product is in service mode by blinking yellow (1Hz).

It is advised to enter this mode do update the parameters of the algorithms, and to leave the service mode once the parameters have been updated.

3.2. Movee algorithms and parameters

3.2.1. MPU - Motion Processing Unit

MPU parameters

	Parameter name	Description	Value			Precision	Unit
			min	typ	max		
MPU	maxRange (= MR)	Dynamic range – **Default value is ±8000**		±16000		N/A	mG
				±8000			
				±4000			
				±2000			

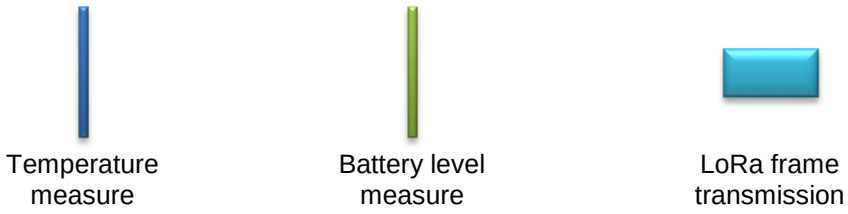
3.2.2. ALIVE algorithm

ALIVE parameters

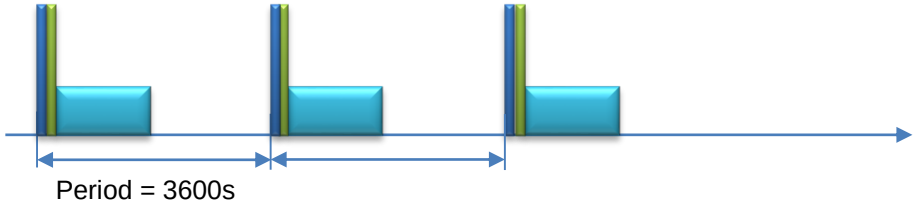
	Parameter name	Description	Value			Preci-sion	Unit
			min	typ	max		
ALIVE	modeRefresh	Activates Periodical frame transmission	0	1	1	N/A	boolean
	period	Period between 2 frames	60	3600	4.3E+06	s	s
	nbSavedValue	Number of period between 2 LoRa frames, with temperature measure for each period	1	1	48	N/A	N/A

See *§3.4 LoRa frames payload description* for payload decoding

ALIVE implementation



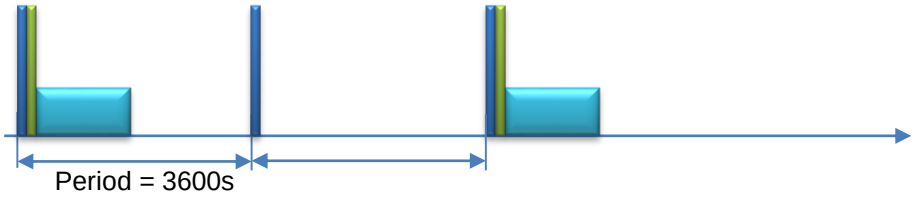
Example 1: modeRefresh = 1 / period = 3600 / nbSavedValue = 1



LoRa frame sent: **e61801aa**

- Payload header: **e618**
  - **0xE6**: battery level : 0xE6 = 230: $\frac{(3,6-2,8)}{255} * 230 + 2,8 = 3,52\ Volts$
  - **0x18**: Temperature = 24°C
- Payload data : **01**
  - **0x01**: Data type = ALIVE => no more payload data
- Payload end of frame: **aa**
  - **0xAA**: End of frame

Example 2: modeRefresh = 1 / period = 3600 / nbSavedValue = 2



LoRa frame sent: **e6180218aa**

- Payload header: **e618**
  - **0xE6**: battery level : 0xE6 = 230: $\frac{(3,6-2,8)}{255} * 230 + 2,8 = 3,52\ Volts$
  - **0x18**: last temperature = 24°C
- Payload data : **0218**
  - **0x02**: Data type = TEMPERATURE => expected payload data = N bytes (signed)
    - **0x18**: Recorded temperature = 24°C
- Payload end of frame: **aa**
  - **0xAA**: End of frame

3.2.3. SHOCK algorithm

SHOCK parameters

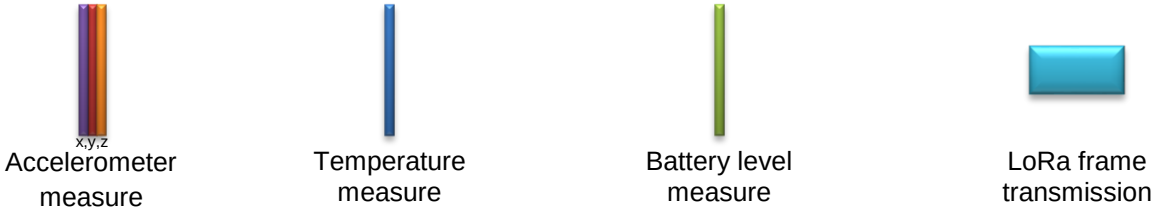
	Parameter name	Description	Value			Preci-sion	Unit
			min	typ	max		
SHOCK	gxSup	Positive threshold on X axis	25	2000	MR[1]	mG	G
	gxInf	Negative threshold on X axis	25	2000	MR		
	gySup	Positive threshold on Y axis	25	2000	MR		
	gyInf	Negative threshold on Y axis	25	2000	MR		
	gzSup	Positive threshold on Z axis	25	2000	MR		
	gzInf	Negative threshold on Z axis	25	2000	MR		
	freq	Sensor sampling rate Possible values = 0.24, 0.49, 0.98, 1.95, 3.91, 7.81, 15.63, 31.25, 62.50, 125, 250 and 500Hz	0.24	15.63	500	N/A	Hz
	inhibition	Inhibition time before a shock can be detected ***If set to 0, then inhibition is deactivated***.	50	250	65535	ms	s
	removeGravity	Activates relative acceleration measure mode (Gravity removed)	0	0	1	N/A	boolean

See *§3.4 LoRa frames payload description* for payload decoding

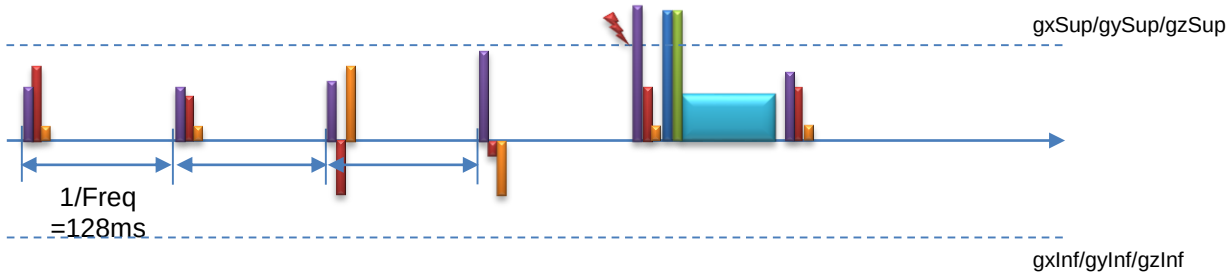
removeGravity parameter activates an algorithm, which suppress the gravity ‘s vector projection from the gx, gy, gz values recorded by the MPU when an acceleration is detected. This algorithm returns less reliable values than normal operation mode (removeGravity not activated).

[1] MR = Max Range, see *§3.2.1 MPU - Motion Processing Unit*

SHOCK implementation



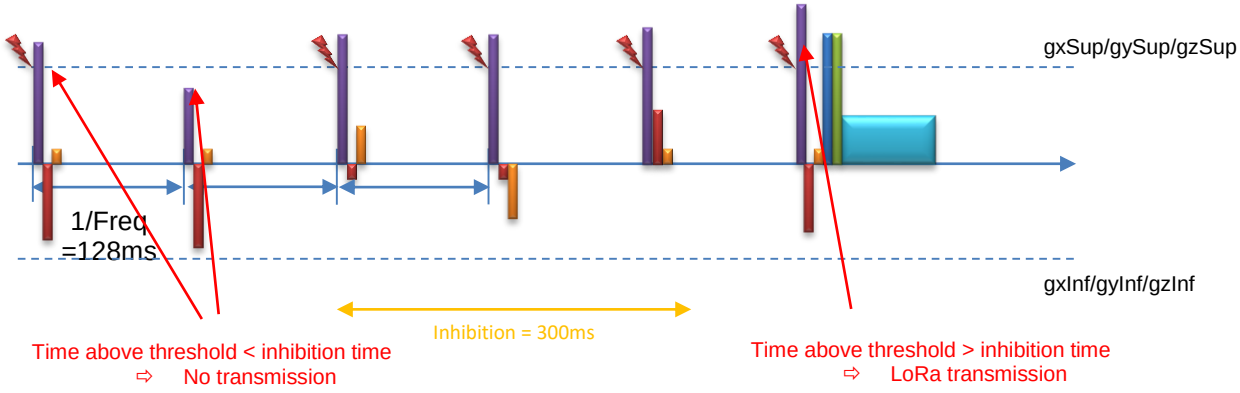
Example 1: gxSup / gxInf / gySup / gyInf / gzSup / gzInf = 2000mG; freq = 7.81; inhibition = 0, removeGravity = 0



LoRa frame sent: **c11a040ad104540134aa**

- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **040ad104540134**
  - **0x04**: Data type = SHOCK => expected payload data = Gx (2 bytes signed), Gy (2 bytes signed), Gz (2 bytes signed)
    - **0x0AD104540134**
      - **0x0AD1**: Gx = 2769mG
      - **0x0454**: Gy = 1108mG
      - **0x0134**: Gz = 308mG
- Payload end of frame: **aa**
  - **0xAA**: End of frame

Example 2: gxSup / gxInf / gySup / gyInf / gzSup / gzInf = 2000mG; freq = 7.81; inhibition = 300, removeGravity = 0



LoRa frame sent: **c11a040c9fff970134aa**

- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **040c9fff970134**
  - **0x04**: Data type = SHOCK => expected payload data = Gx (2 bytes signed), Gy (2 bytes signed), Gz (2 bytes signed)
    - **0x0C9FFF970134**
      - **0x0C9F**: Gx = 3231mG
      - **0xFF97**: Gy = - 1385mG
      - **0x0134**: Gz = 308mG
- Payload end of frame: **aa**
  - **0xAA**: End of frame

3.2.4. MOTION algorithm

MOTION parameters

	Parameter name	Description	Value			Precision	Unit
			min	typ	max		
MOTION	gxSup	Positive threshold on X axis	25	500	MR[2]	mG	G
	gxInf	Negative threshold on X axis	25	500	MR		
	gySup	Positive threshold on Y axis	25	500	MR		
	gyInf	Negative threshold on Y axis	25	500	MR		
	gzSup	Positive threshold on Z axis	25	500	MR		
	gzInf	Negative threshold on Z axis	25	500	MR		
	Freq	Sensor sampling rate Possible values = 0.24, 0.49, 0.98, 1.95, 3.91, 7.81, 15.63, 31.25, 62.50, 125, 250 and 500Hz	0.24	15.63	500	N/A	Hz
	timerA	Lapse of time with motion detection before a motion is detected/reported	500	2000	65535	ms	s
	timerB	Lapse of time without motion detection before stillness is detected/reported	500	3000	65535		
	sensitivity	Number of samples greater than the defined threshold, during timerA value, involving activity detection	0	10	$\frac{freq.timerA}{1000}$	N/A	integer
	activity	Activates motion duration calculation	0	1	1	N/A	boolean
	additionateActivity	If activated, the motion duration adds up, and is not reset at each LoRa transmission	0	1	1		
	periodicActivity	Activates a periodic summary of the activity measured on the period defined by "activityResumePeriod"	0	1	1		
	activityResumePeriod	Period of the activity summary transmission. Defined in hours.	1	1	360	h	s
	loraAtStartMvt	The LoRa frame is sent when motion is detected/reported	0	0	1	N/A	boolean
	loraAtStopMvt	The LoRa frame is sent when stillness is detected/reported	0	1	1		

See *§3.4 LoRa frames payload description* for payload decoding

If *activity* parameter is set to 1, *loraAtStartMvt* parameter will be ignored.

If activity parameter is set to 0, *additionateActivity*, *periodicActivity* and *activityResumePeriod* will be ignored.

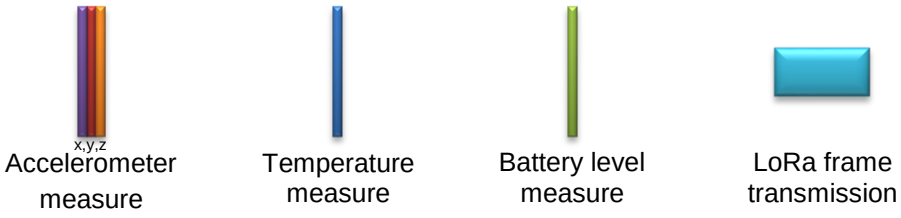
If *loraAtStartMvt* is set to 1, *timerB* parameter will be ignored.

If *periodicActivity* is set to 1, *loraAtStartMvt* and *loraAtStopMvt* parameters will be ignored, and *activityResumePeriod* parameter will be used.

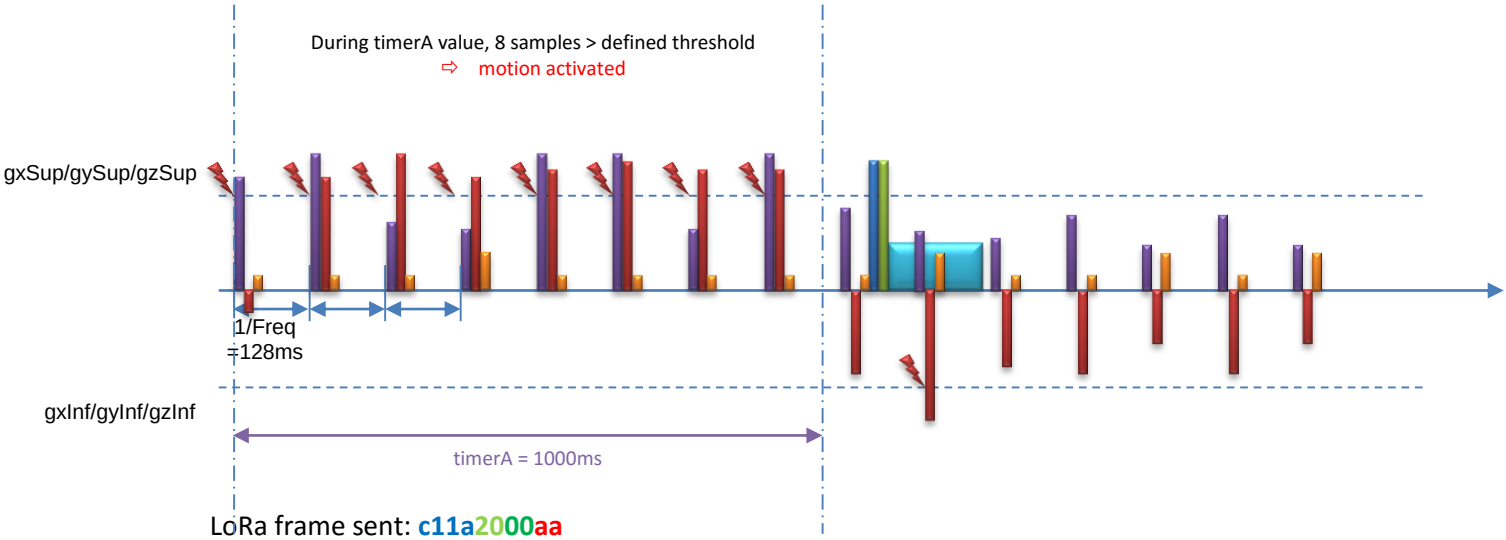
activityResumePeriod parameter will only be used when *activity* and *periodicActivity* are set to 1.

[2] MR = Max Range, see *§3.2.1 MPU - Motion Processing Unit*

MOTION implementation



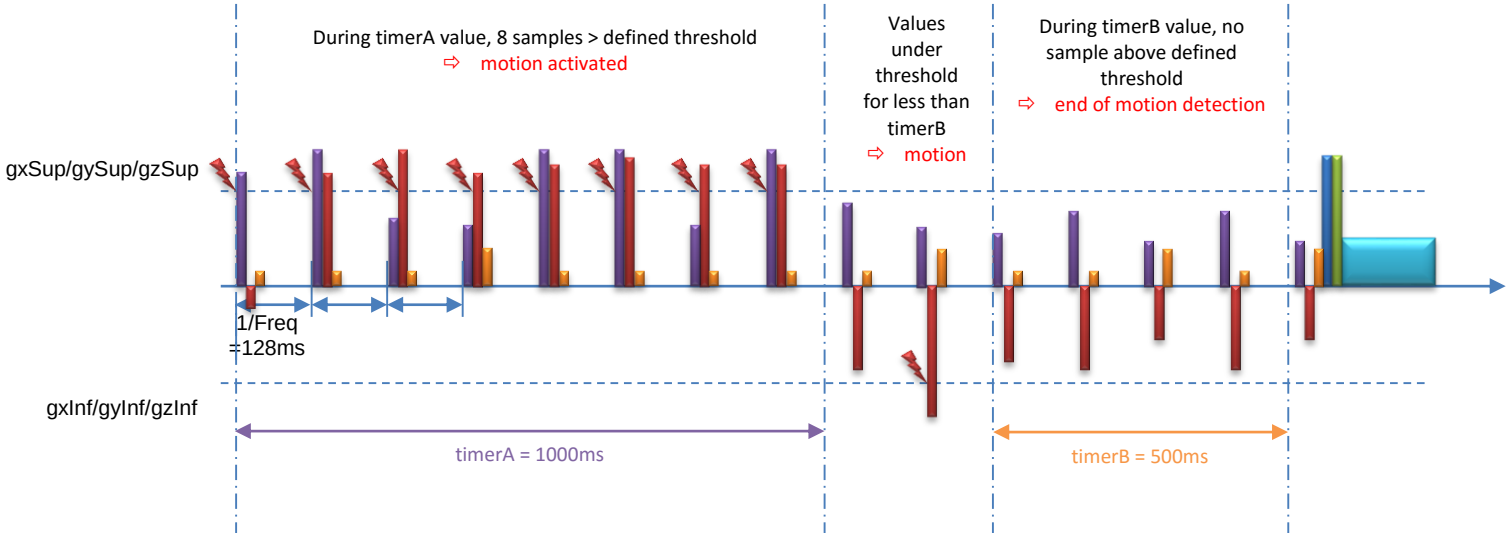
Example 1: gxSup / gxInf / gySup / gyInf / gzSup / gzInf = 2000mG; freq = 7.81; timerA = 1000; *timerB = 500*; sensitivity = 6; **activity = 0**; *additionateActivity = 0; periodicActivity = 0; activityResumePeriod = 1*; loraAtStartMvt = 1; loraAtStopMvt = 0 (*parameter ignored*)



LoRa frame sent: **c11a2000aa**

- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **2000**
  - **0x20**: Data type = MOTION => expected payload data = Motion/Stillness (1 byte)
    - **0x00**: Motion
- Payload end of frame: **aa**
  - **0xAA**: End of frame

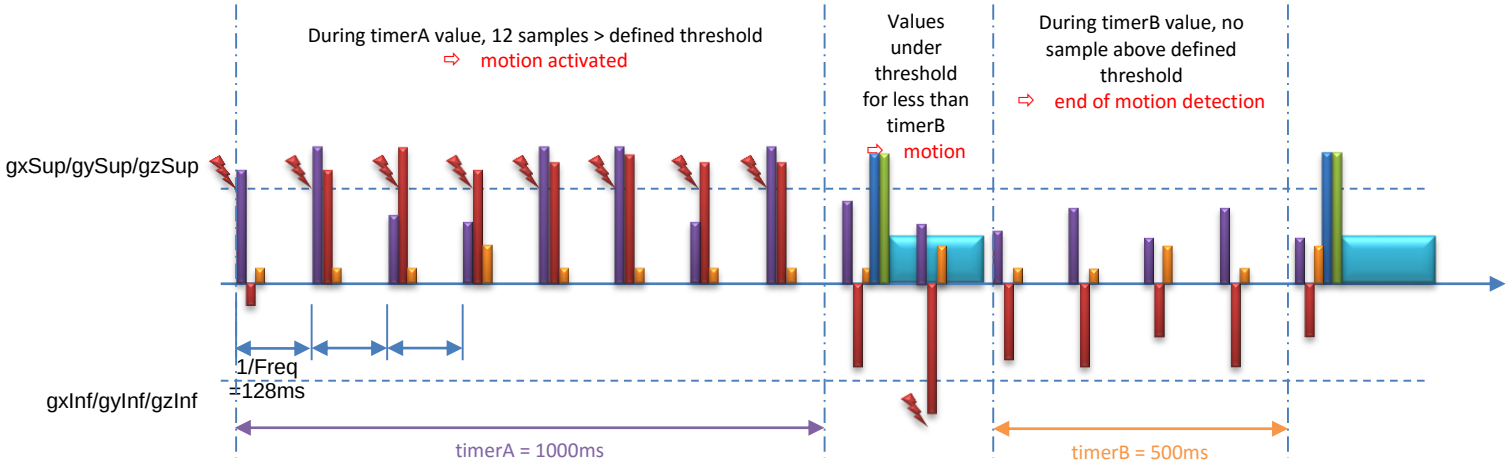
Example 2: gxSup / gxInf / gySup / gyInf / gzSup / gzInf = 2000mG; freq = 7.81; timerA = 1000; timerB = 500; sensitivity = 6; **activity = 0**; *additionateActivity = 0; periodicActivity = 0; activityResumePeriod = 1*; loraAtStartMvt = 0; loraAtStopMvt = 1 (*parameter ignored*)



LoRa frame sent: **c11a2001aa**

- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **2001**
  - **0x20**: Data type = MOTION => expected payload data = Motion/Stillness (1 byte)
    - **0x01**: Stillness
- Payload end of frame: **aa**
  - **0xAA**: End of frame

Example 3: gxSup / gxInf / gySup / gyInf / gzSup / gzInf = 2000mG; freq = 7.81; timerA = 1000; timerB = 500; sensitivity = 6; **activity = 0**; *additionateActivity = 0; periodicActivity = 0; activityResumePeriod = 1*; loraAtStartMvt = 1; loraAtStopMvt = 1 (*parameter ignored*)

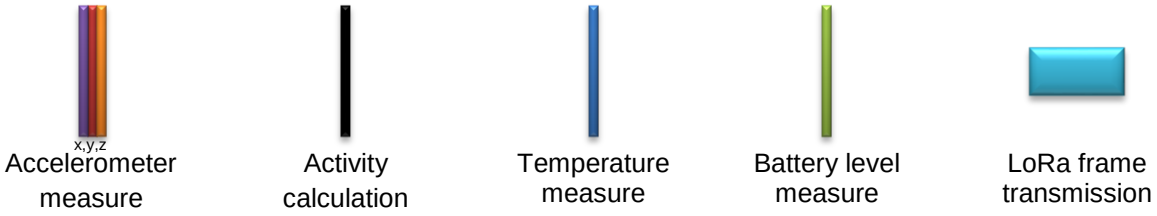


LoRa frame sent at start: **c11a2000aa**

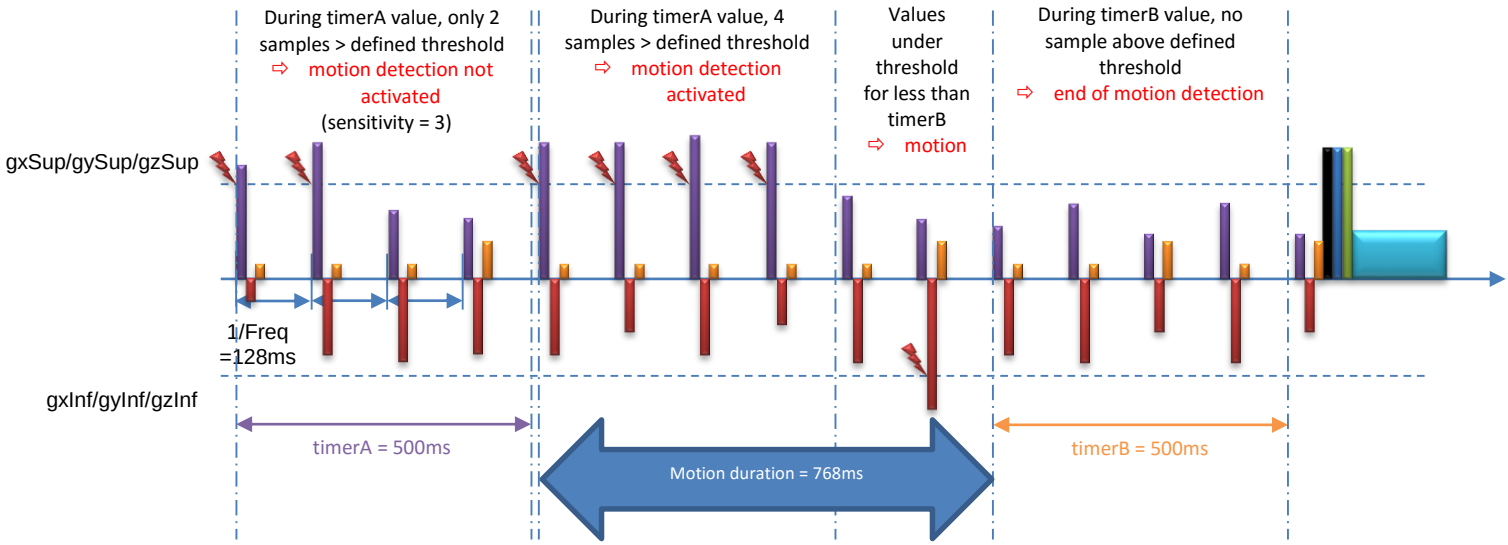
- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **2000**
  - **0x20**: Data type = MOTION => expected payload data = Motion/Stillness (1 byte)
    - **0x00**: Motion
- Payload end of frame: **aa**
  - **0xAA**: End of frame

LoRa frame sent at stop (example given, real life duty cycle restrictions will not allow this transmission): **c11a2001aa**

- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **2001**
  - **0x20**: Data type = MOTION => expected payload data = Motion/Stillness (1 byte)
    - **0x01**: Stillness
- Payload end of frame: **aa**
  - **0xAA**: End of frame



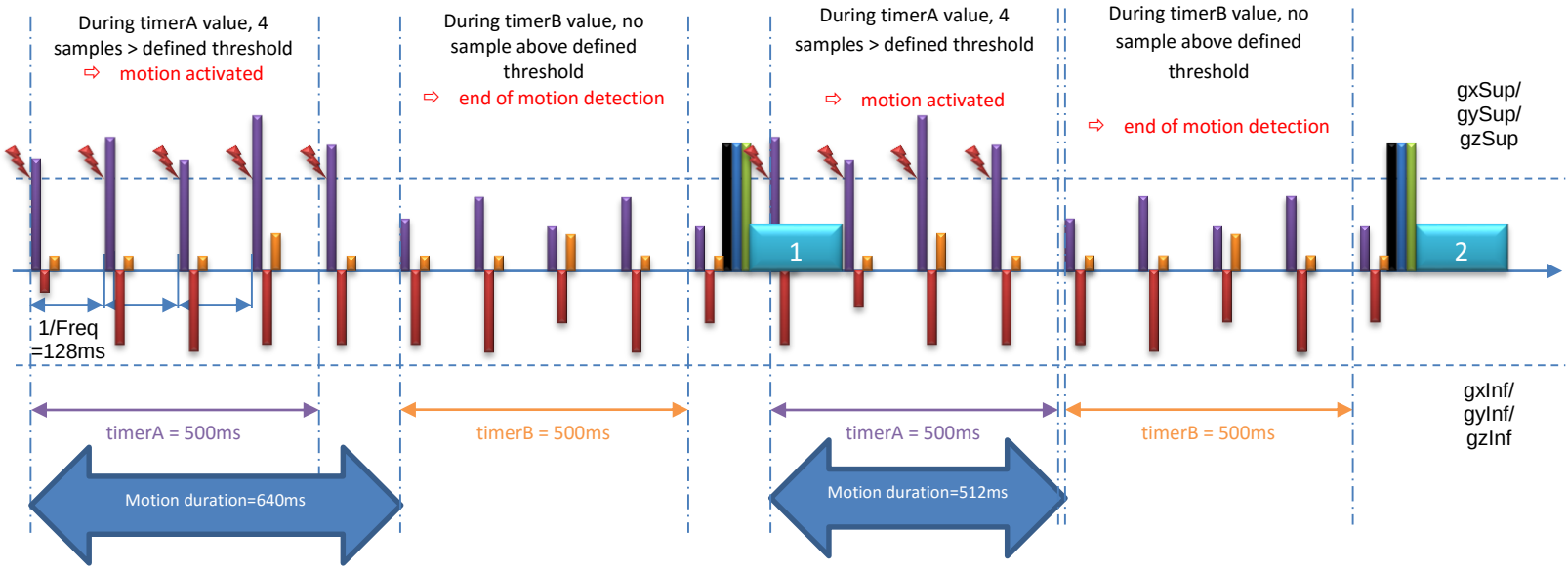
Example 4: gxSup / gxInf / gySup / gyInf / gzSup / gzInf = 2000mG; freq = 7.81; timerA = 500; timerB = 500; sensitivity = 3; **activity = 1**; **additionateActivity = 0**; *periodicActivity = 0; activityResumePeriod = 1; loraAtStartMvt = 1; loraAtStopMvt = 1* (*parameter ignored*)



LoRa frame sent: **c11a400000000300aa**

- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **400000000300**
  - **0x40**: Data type = ACTIVITY => expected payload data = Motion/Stillness (1 byte), duration (4 bytes unsigned)
    - **0x00**: Motion
      - **0x00000300**: duration = 768ms
- Payload end of frame: **aa**
  - **0xAA**: End of frame

Example 5: gxSup / gxInf / gySup / gyInf / gzSup / gzInf = 2000mG; freq = 7.81; timerA = 500; timerB = 500; sensitivity = 3; **activity = 1**; **additionateActivity = 1**; periodicActivity = 0; *activityResumePeriod = 1; loraAtStartMvt = 1; loraAtStopMvt = 1* (*parameter ignored*)



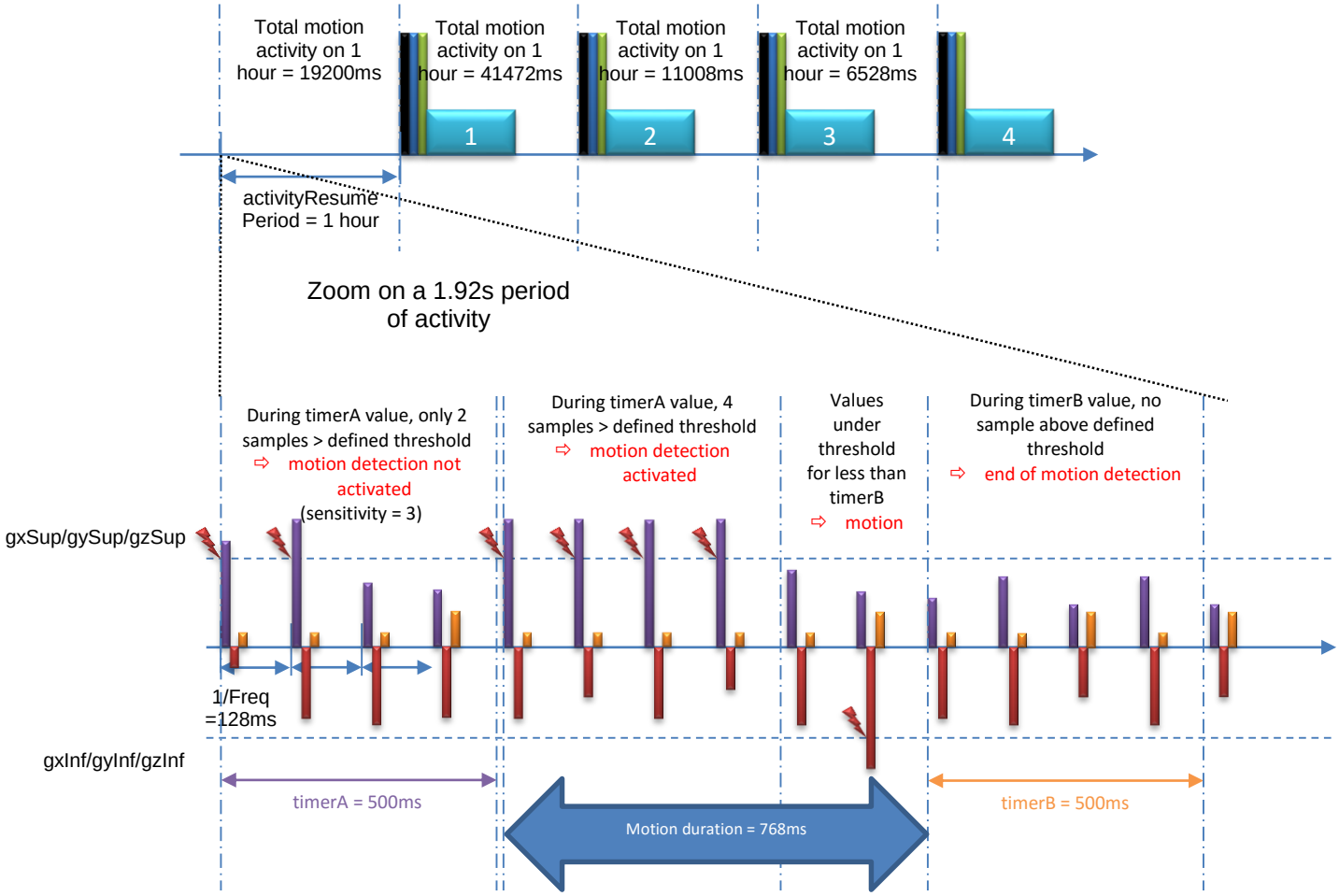
LoRa frame "1" sent: **c11a400000000280aa**

- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **400000000280**
  - **0x40**: Data type = ACTIVITY => expected payload data = Motion/Stillness (1 byte), duration (4 bytes unsigned)
    - **0x00**: Motion
      - **0x00000280**: duration = 640ms
- Payload end of frame: **aa**
  - **0xAA**: End of frame

LoRa frame “2” sent: **c11a400000000480aa**

- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **400000000480**
  - **0x40**: Data type = ACTIVITY => expected payload data = Motion/Stillness (1 byte), duration (4 bytes unsigned)
    - **0x00**: Motion
      - **0x00000480**: duration = 1152ms (=640+512ms)
- Payload end of frame: **aa**
  - **0xAA**: End of frame

Example 6: gxSup / gxInf / gySup / gyInf / gzSup / gzInf = 2000mG; freq = 7.81; timerA = 500; timerB = 500; sensitivity = 3; **activity = 1**; **additionateActivity = 1**; **periodicActivity = 1**; **activityResumePeriod = 1***; loraAtStartMvt = 1; loraAtStopMvt = 1* (*parameter ignored*)



LoRa frame "1" sent: **c11a400000004b00aa**

- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **400000004b00**
  - **0x40**: Data type = ACTIVITY => expected payload data = Motion/Stillness (1 byte), duration (4 bytes unsigned)
    - **0x00**: Motion
      - **0x00004B00**: duration = 19200ms
- Payload end of frame: **aa**
  - **0xAA**: End of frame

LoRa frame "2" sent: **c11a40000000ed00aa**

- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **40000000ed00**
  - **0x40**: Data type = ACTIVITY => expected payload data = Motion/Stillness (1 byte), duration (4 bytes unsigned)
    - **0x00**: Motion
      - **0x0000ED00**: duration = 60672ms (=19200+41472ms)
- Payload end of frame: **aa**
  - **0xAA**: End of frame

LoRa frame "3" sent: **c11a400000011800aa**

- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **400000011800**
  - **0x40**: Data type = ACTIVITY => expected payload data = Motion/Stillness (1 byte), duration (4 bytes unsigned)
    - **0x00**: Motion
      - **0x00011800**: duration = 71680ms (=60672+11008ms)
- Payload end of frame: **aa**
  - **0xAA**: End of frame

LoRa frame "4" sent: **c11a400000013180aa**

- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **400000013180**
  - **0x40**: Data type = ACTIVITY => expected payload data = Motion/Stillness (1 byte), duration (4 bytes unsigned)
    - **0x00**: Motion
      - **0x00013180**: duration = 78208ms (=71680+6528ms)
- Payload end of frame: **aa**
  - **0xAA**: End of frame

If **periodicActivity = 0**, LoRa payloads would be the activity on the corresponding activityResumePeriod time slot (e.g. LoRa frame 3 payload would have been **0x0000ED00**: duration = 60672ms)

3.2.5. TEMPERATURE algorithm

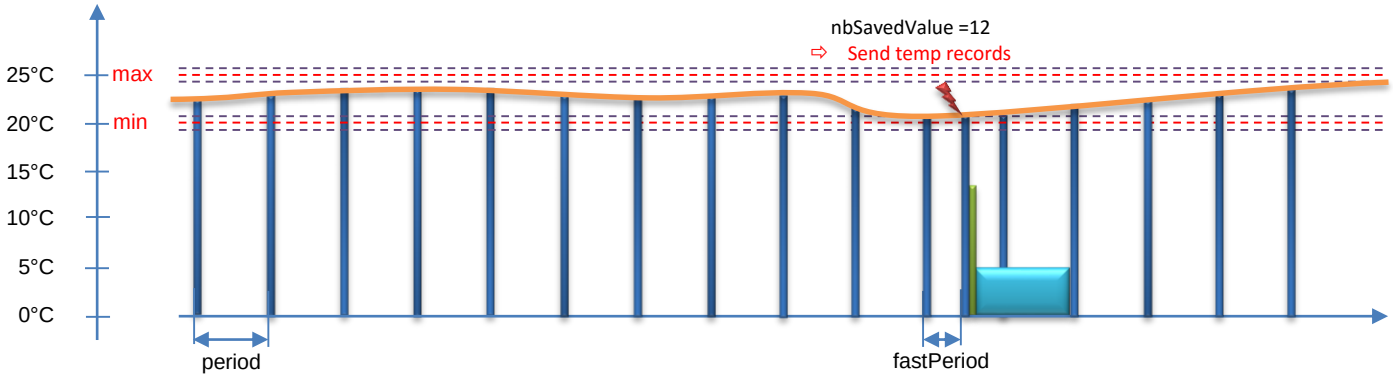
TEMPERATURE parameters

	Parameter name	Description	Value			Precision	Unit
			min	typ	max		
TEMPERATURE	nbSavedValue	Number of values to measure and store between 2 frames	1	12	48	N/A	Integer
	modeThreshold	Activates Temperature threshold	0	1	1		Boolean
	max	Max temperature threshold	-32768	2500	32767	0.01 °C	°C
	min	Min temperature threshold	-32768	2000	32767		
	delta	Gap before a threshold (min or max) is reached. If the gap is reached, the period between 2 measures is set to fastPeriod	0	100	255		
	period	Period between 2 measures when the temperature is normal (gap not reached)	500	30000	4.3E+09	ms	S
	fastPeriod	Period between 2 measures when the temperature has reached the defined gap (min or max)	250	5000	65535		
	ultraFastPeriod	Period between 2 measures when the temperature has reached the threshold (min or max)	150	2000	65535		
	inhibition	Number of values above max threshold or under min threshold before an alert is sent	0	3	255	N/A	Integer

See *§3.4 LoRa frames payload description* for payload decoding

TEMPERATURE implementation

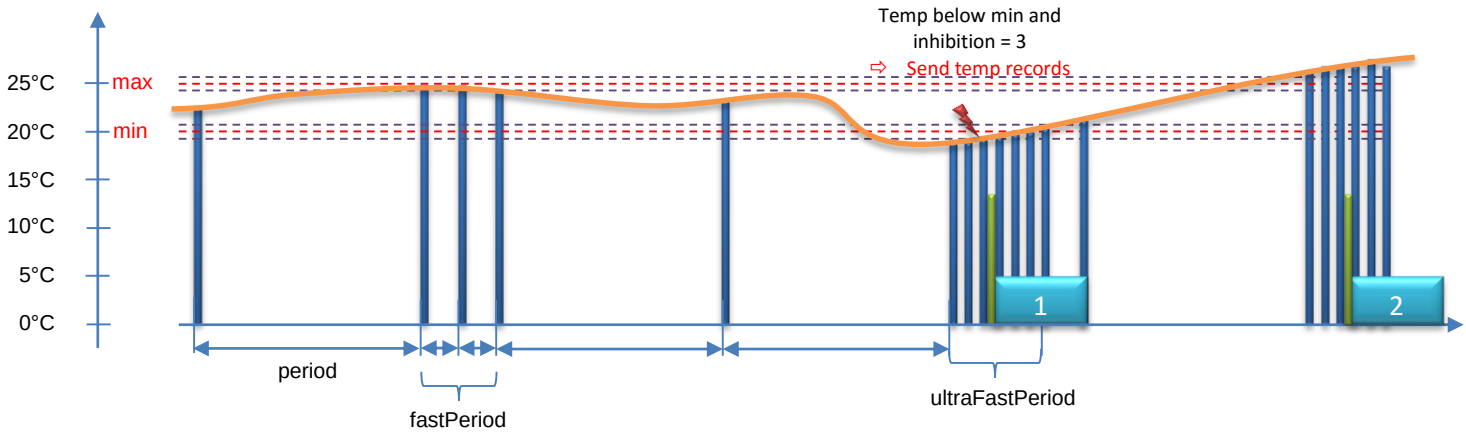
Example 1: nbSavedValue = 12; modeThreshold = 1; max = 2500; min = 2000; delta = 100; period = 30000; fastPeriod = 15000; ultraFastPeriod = 2000; inhibition = 3



LoRa frame sent: **c115021515161817171718181817 16aa**

- Payload header: **c115**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x15**: Temperature = 21°C
- Payload data : **021515161817171718181817 16**
  - **0x02**: Data type = TEMPERATURE => expected payload data = N bytes (signed)
    - **0x15**: Recorded temperature 1 = 21°C
    - **0x15**: Recorded temperature 2 = 21°C
    - **0x16**: Recorded temperature 3 = 22°C
    - **0x18**: Recorded temperature 4 = 24°C
    - **0x17**: Recorded temperature 5 = 23°C
    - **0x17**: Recorded temperature 6 = 23°C
    - **0x17**: Recorded temperature 7 = 23°C
    - **0x18**: Recorded temperature 8 = 24°C
    - **0x18**: Recorded temperature 9 = 24°C
    - **0x18**: Recorded temperature 1 = 24°C
    - **0x17**: Recorded temperature 1 = 23°C
    - **0x16**: Recorded temperature 1 = 22°C
- Payload end of frame: **aa**
  - **0xAA**: End of frame

Example 2: nbSavedValue = 12; modeThreshold = 1; max = 2500; min = 2000; delta = 100; period = 30000; fastPeriod = 5000; ultraFastPeriod = 2000; inhibition = 3



LoRa frame "1" sent: **c113021312121718181816aa**

- Payload header: **c113**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x13**: Temperature = 19°C
- Payload data : **021312121718181816**
  - **0x02**: Data type = TEMPERATURE => expected payload data = N bytes (signed)
    - **0x13**: Recorded temperature 1 = 19°C
    - **0x12**: Recorded temperature 2 = 18°C
    - **0x12**: Recorded temperature 3 = 18°C
    - **0x17**: Recorded temperature 4 = 23°C
    - **0x18**: Recorded temperature 5 = 24°C
    - **0x18**: Recorded temperature 6 = 24°C
    - **0x18**: Recorded temperature 7 = 24°C
    - **0x16**: Recorded temperature 8 = 22°C
- Payload end of frame: **aa**
  - **0xAA**: End of frame

LoRa frame “2” sent: **c11b021b1b1a1615141413aa**

- Payload header: **c11b**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1B**: Temperature = 27°C
- Payload data : **021b1b1a1615141413**
  - **0x02**: Data type = TEMPERATURE => expected payload data = N bytes (signed)
    - **0x1B**: Recorded temperature 1 = 27°C
    - **0x1B**: Recorded temperature 2 = 27°C
    - **0x1A**: Recorded temperature 3 = 26°C
    - **0x16**: Recorded temperature 4 = 22°C
    - **0x15**: Recorded temperature 5 = 21°C
    - **0x14**: Recorded temperature 6 = 20°C
    - **0x14**: Recorded temperature 7 = 20°C
    - **0x13**: Recorded temperature 8 = 19°C
- Payload end of frame: **aa**
  - **0xAA**: End of frame

3.2.6. TILT algorithm

TILT parameters

	Parameter name	Description	Value			Precision	Unit
			min	typ	max		
TILT	modeRefresh	Activates periodical automatic tilt measurement	0	1	1	N/A	boolean
	modeOnMove	Activates tilt measurement on motion detection	0	1	1		
	pitch	Sends a LoRa frame when this threshold is reached (threshold defined as absolute value)	1	10	90	0.1°	degree
	roll	Sends a LoRa frame when this threshold is reached (threshold defined as absolute value)	1	10	180		
	threshold	Motion detection threshold	50	200	1020	mG	G
	inhibition	Number of values above threshold before tilt calculation	1	5	255	N/A	Integer
	period	Period between 2 measures	500	10000	4.3E+09	ms	s

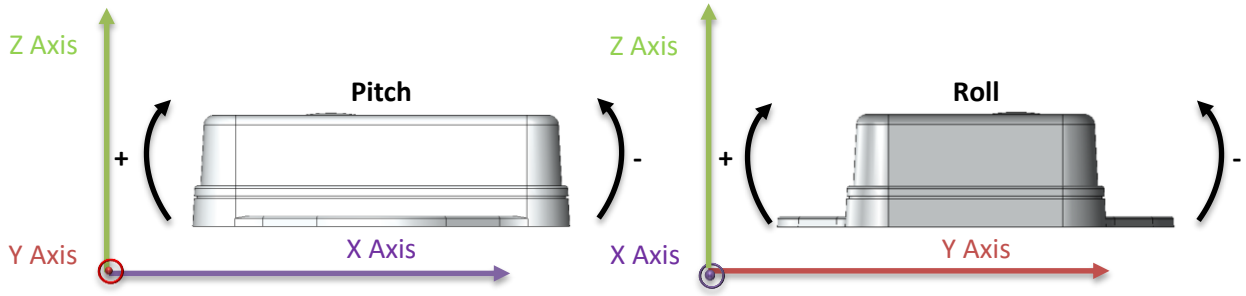
See *§3.4 LoRa frames payload description* for payload decoding

If *modeOnMove* is set to 0, *threshold* parameter is ignored.

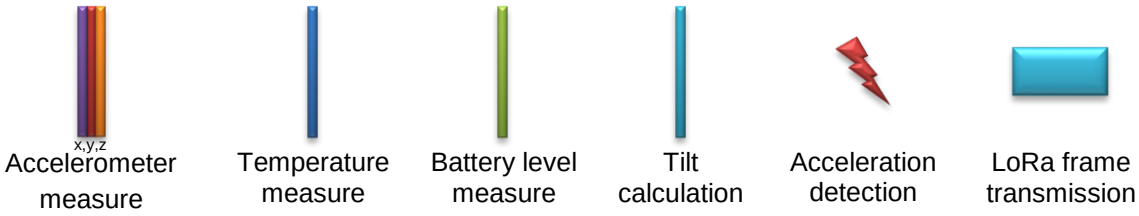
If *modeRefresh* is set to 0, *period* parameter is ignored.

modeOnMove and *modeRefresh* can be activated at the same time.

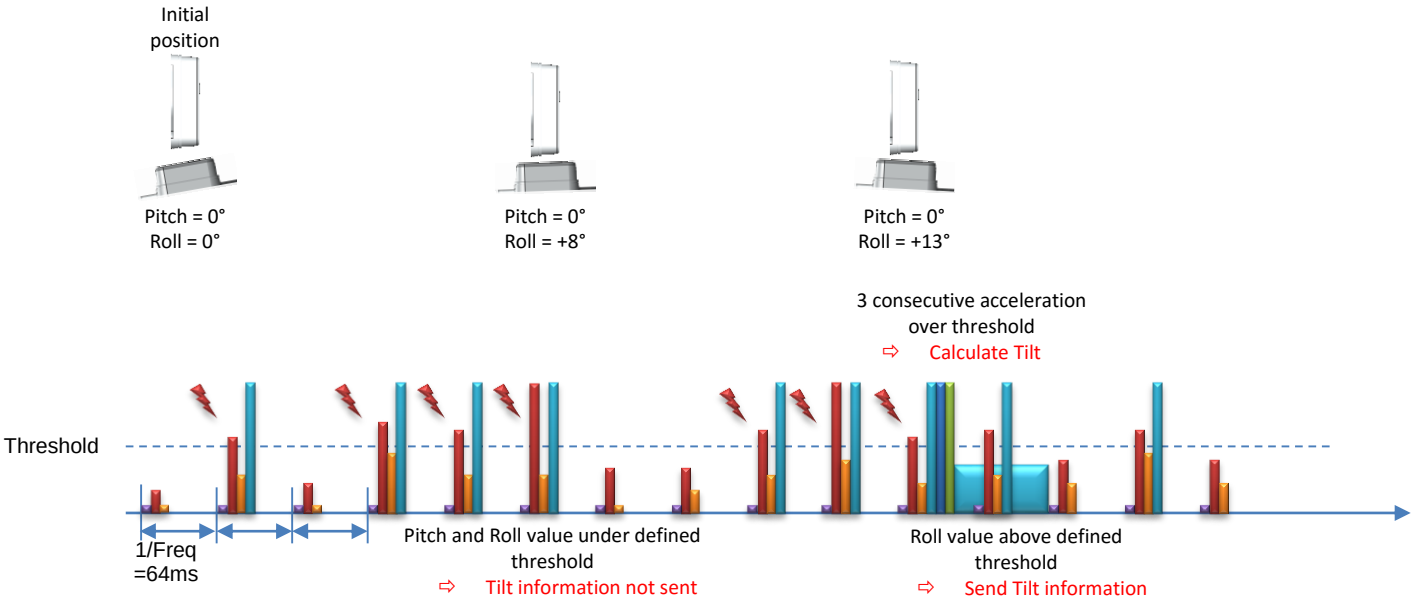
Motion sensor sampling rate is set to 15.63Hz.



TILT implementation



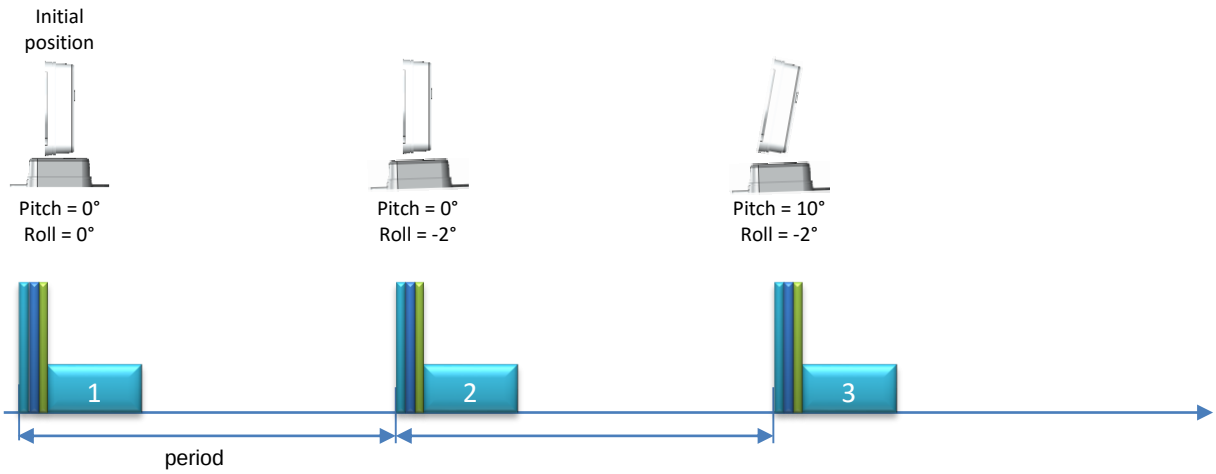
Example 1: modeRefresh = **0**; modeOnMove = **1**; pitch = +10; roll = +10; threshold = 200; inhibition = 3; *period = 10000* (*parameter ignored*)



LoRa frame sent: **c115**0800000082**aa**

- Payload header: **c115**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x15**: Temperature = 21°C
- Payload data : **0800000082**
  - **0x08**: Data type = TILT => expected payload data = 2 bytes Pitch (signed) + 2 bytes Roll (signed)
    - **0x0000**: Pitch angle value = 0 x 0.1° = 0°
    - **0x0082**: Roll angle value = 130 x 0.1° = +13°
- Payload end of frame: **aa**
  - **0xAA**: End of frame

Example 2: modeRefresh = **1**; modeOnMove = **0**; pitch = 10; roll = 10; *threshold = 200; inhibition = 3*; period = 10000 (*parameter ignored*)



LoRa frame "1" sent: **c115080000000000aa**

- Payload header: **c115**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x15**: Temperature = 21°C
- Payload data : **0800000000**
  - **0x08**: Data type = TILT => expected payload data = 2 bytes Pitch (signed) + 2 bytes Roll (signed)
    - **0x0000**: Pitch angle value = 0 x 0.1° = 0°
    - **0x0000**: Roll angle value = 0 x 0.1° = 0°
- Payload end of frame: **aa**
  - **0xAA**: End of frame

LoRa frame “2” sent: **c115080000ffecaa**

- Payload header: **c115**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x15**: Temperature = 21°C
- Payload data : **080000ffec**
  - **0x08**: Data type = TILT => expected payload data = 2 bytes Pitch (signed) + 2 bytes Roll (signed)
    - **0x0000**: Pitch angle value = 0 x 0.1° = 0°
    - **0xFFEC**: Roll angle value = -20 x 0.1° = -2°
- Payload end of frame: **aa**
  - **0xAA**: End of frame

LoRa frame “3” sent: **c115080064ffecaa**

- Payload header: **c115**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x15**: Temperature = 21°C
- Payload data : **080064ffec**
  - **0x08**: Data type = TILT => expected payload data = 2 bytes Pitch (signed) + 2 bytes Roll (signed)
    - **0x0064**: Pitch angle value = 100 x 0.1° = +10°
    - **0xFFEC**: Roll angle value = -20 x 0.1° = -2°
- Payload end of frame: **aa**
  - **0xAA**: End of frame

3.2.7. ROTATION algorithm

ROTATION parameters

	Parameter name	Description	Value			Precision	Unit
			min	typ	max		
ROTATION	modeRefresh	Activates Rotation measurement (with defined period between 2 measures)	0	1	1	N/A	boolean
	modeOnMove	Activates rotation measurement if motion is detected	0	1	1		boolean
	lap	Number of turns before a LoRa frame is sent	1	5	65535		Integer
	resetLap	If activated, a press on the push button (more than 3s and less than 5s = yellow LED) will reset the turn counter	0	1	1		boolean
	threshold	Motion detection threshold	50	200	1020	mG	G
	period	Period between 2 measures (if modeRefresh = 1)	500	5000	65535	ms	S

See *§3.4 LoRa frames payload description* for payload decoding

If *modeOnMove* is set to 0, *threshold* parameter is ignored.

If *modeRefresh* is set to 0, *period* parameter is ignored.

In *modeRefresh* mode, maximum detectable speed is 30 turns/min (period = 500ms), else the number of data acquisition won't be sufficient to get consistent measures.

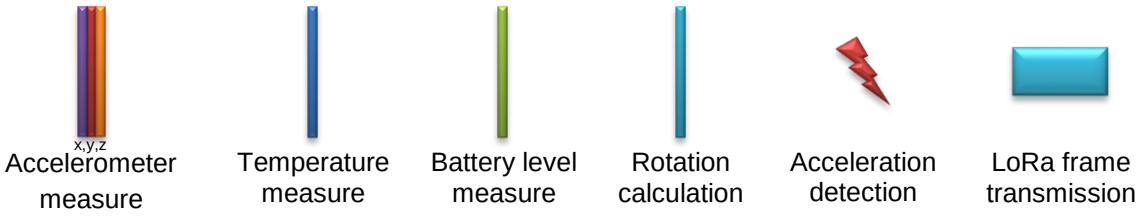
modeOnMove and *modeRefresh* can be activated at the same time.

In order to calculate the lap number on a motor axis, a system rotation is operated on the accelerometer value in order to set the gravity ($\vec{g}$) on -X axis, instead of -Z axis in Tilt and Orientation modes. Yaw is then placed on X axis instead of Z axis.

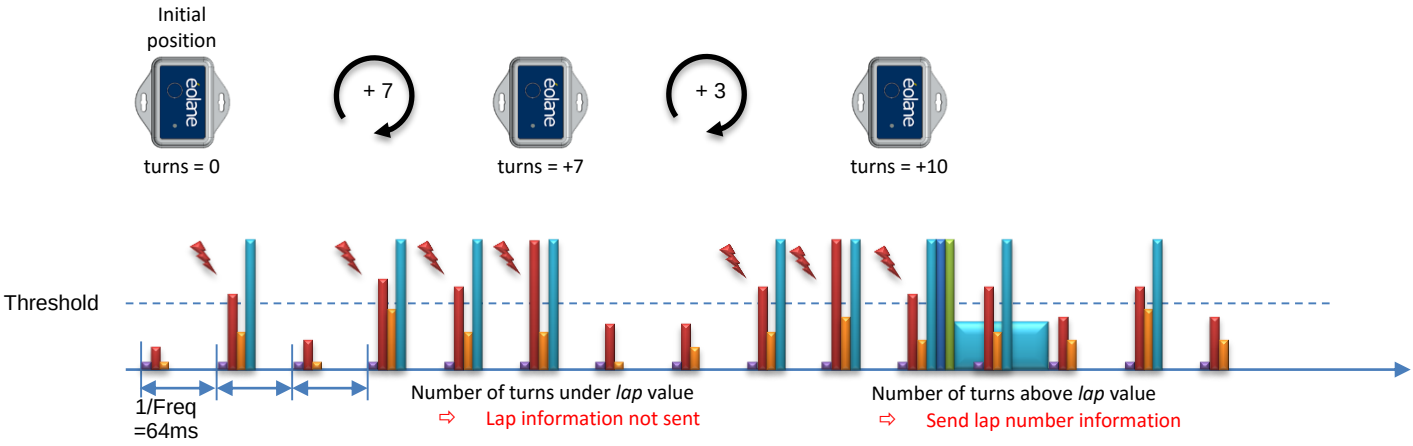
For slow turn count, *threshold* shall be set with a low value. Maximum detectable speed is 230 turns/min.

Motion sensor sampling rate is set to 15.63Hz.

ROTATION implementation



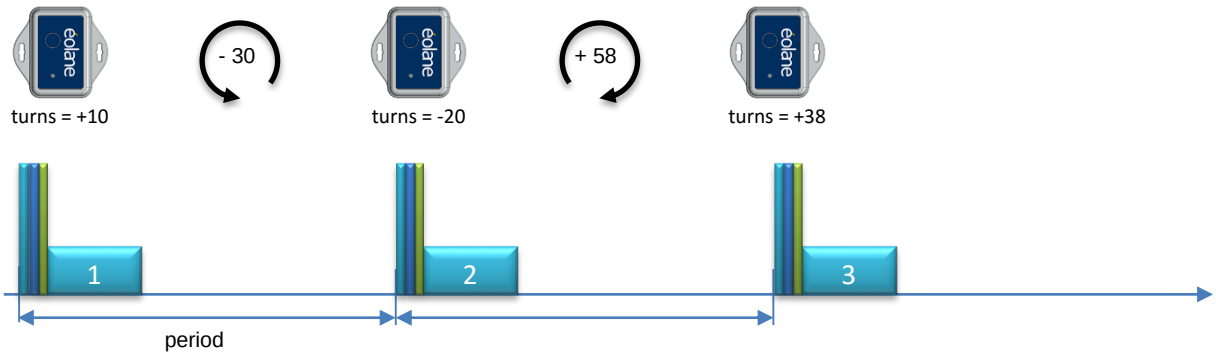
Example 1: modeRefresh = **0**; modeOnMove = **1**; lap = 10; resetLap = 1; threshold = 200; *period = 5000* (*parameter ignored*)



LoRa frame sent: **c11580000aaa**

- Payload header: **c115**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x15**: Temperature = 21°C
- Payload data : **80000a**
  - **0x80**: Data type = ROTATION => expected payload data = 2 bytes (signed) number of turns
    - **0x000A**: number of turns = +10
- Payload end of frame: **aa**
  - **0xAA**: End of frame

Example 2: modeRefresh = **1**; modeOnMove = **0**; *lap = 10*; resetLap = 1; *threshold = 200*; period = 5000 (*parameter ignored*)



LoRa frame "1" sent: **c11580000aaa**

- Payload header: **c115**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x15**: Temperature = 21°C
- Payload data : **80000a**
  - **0x80**: Data type = ROTATION => expected payload data = 2 bytes (signed) number of turns
    - **0x000A**: number of turns = +10
- Payload end of frame: **aa**
  - **0xAA**: End of frame

LoRa frame "2" sent: **c11580ffe2aa**

- Payload header: **c115**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x15**: Temperature = 21°C
- Payload data : **80ffe2**
  - **0x80**: Data type = ROTATION => expected payload data = 2 bytes (signed) number of turns
    - **0xFFE2**: number of turns = -20
- Payload end of frame: **aa**
  - **0xAA**: End of frame

LoRa frame “3” sent: **c115800026aa**

- Payload header: **c115**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x15**: Temperature = 21°C
- Payload data : **800026**
  - **0x80**: Data type = ROTATION => expected payload data = 2 bytes (signed) number of turns
    - **0x0026**: number of turns = +38
- Payload end of frame: **aa**
  - **0xAA**: End of frame

3.2.8. ORIENT algorithm

ORIENT parameters

	Parameter name	Description	Value			Preci-sion	Unit
			min	typ	max		
ORIENT	period	Minimum period between 2 measures	1	30	65535	s	s
	modeRefresh	Activates orientation measurement (with defined period between 2 measures)	0	1	1	N/A	boolean
	modeOnMove	Activates orientation measurement if motion is detected	0	1	1		
	threshold	Acceleration threshold for orientation measurement	0	800	2000	mG	G
	mesureLenght	Orientation duration measurement (gyroscope activation period)	3	5	255	s	S
	pitch[3]	Pitch threshold for LoRa frame transmission	-180	90	180	1°	degree signed
	roll[4]	Roll threshold for LoRa frame transmission	-90	-45	90		
	yaw[5]	Yaw threshold for LoRa frame transmission	-180	90	180		

See *§3.4 LoRa frames payload description* for payload decoding

If *modeOnMove* is set to 0, *threshold* parameter is ignored.

If *modeRefresh* is set to 0, *period* parameter is ignored.

In *modeOnMove* mode, the Yaw information is consistent with the real position as long as the threshold is set with a low value. Pitch and Roll values stay consistent with the real position of the product at any time.

In *modeRefresh* mode, the Yaw information is reset at each orientation measure, and any Yaw rotation of the product is lost between 2 measures. Pitch and Roll values stay consistent with the real position of the product at any time.

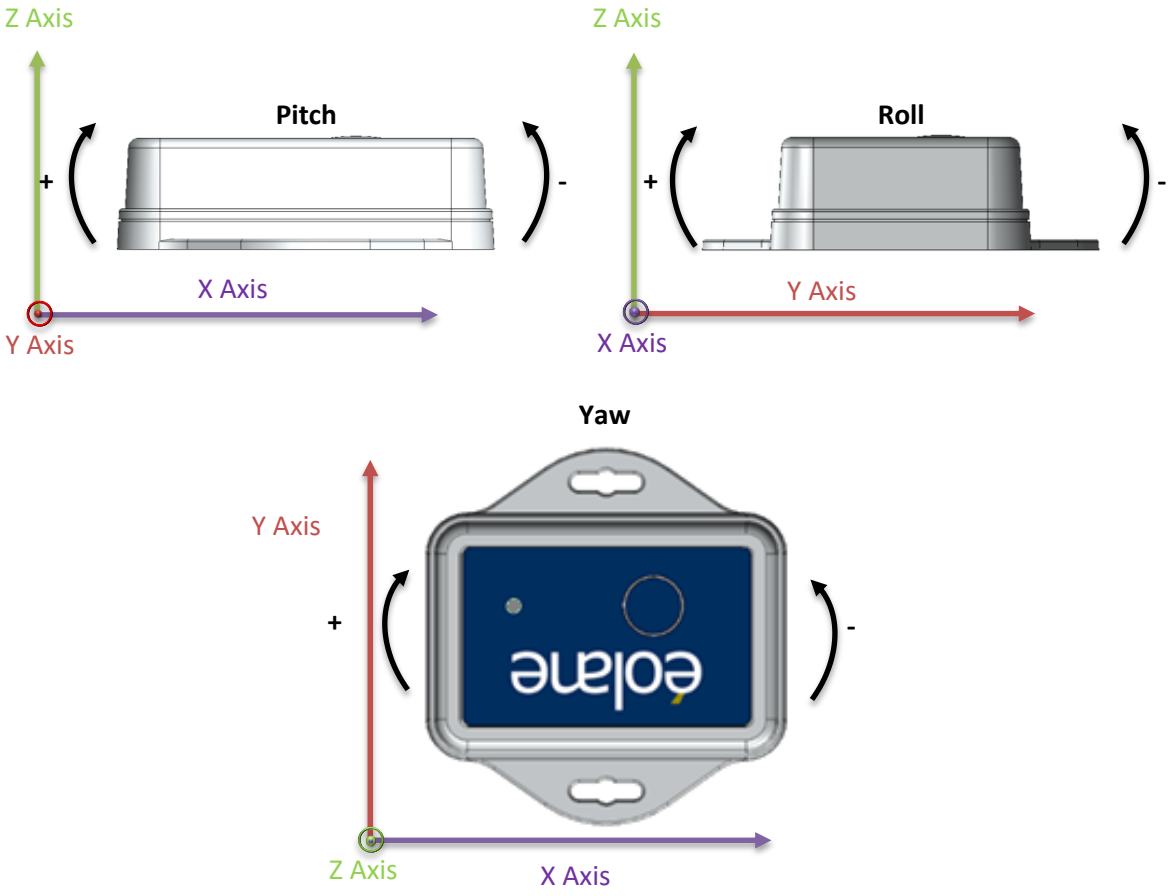
modeOnMove and *modeRefresh* can be activated at the same time.

Motion sensor sampling rate is set to 7.81Hz

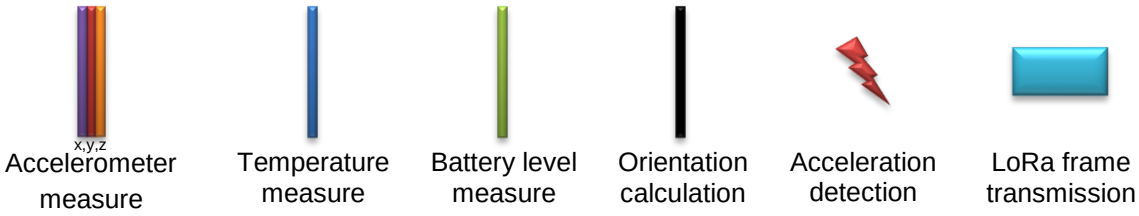
In the *modeOnMove* mode, the acceleration threshold is used to start the gyroscope activation and orientation measurement. The orientation measurement is then activated for a *mesureLenght* period.

Orientation algorithm is more demanding on resources than Tilt algorithm, as a consequence the measure precision is of 1° (instead of 0.1° for Tilt algorithm)

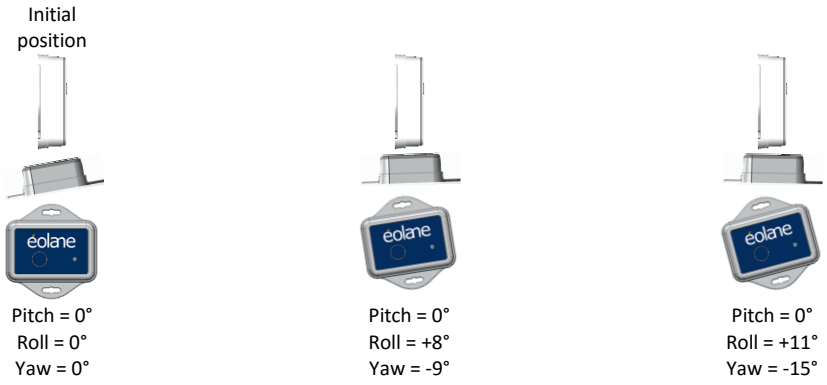
[3] Threshold measured from initial position at device init.
[4] Threshold measured from initial position at device init.
[5] Yaw is measured upon measure trigger, it does not correspond to the device Yaw measured from initial position at device init as the device doesn't have a magnetic compass.

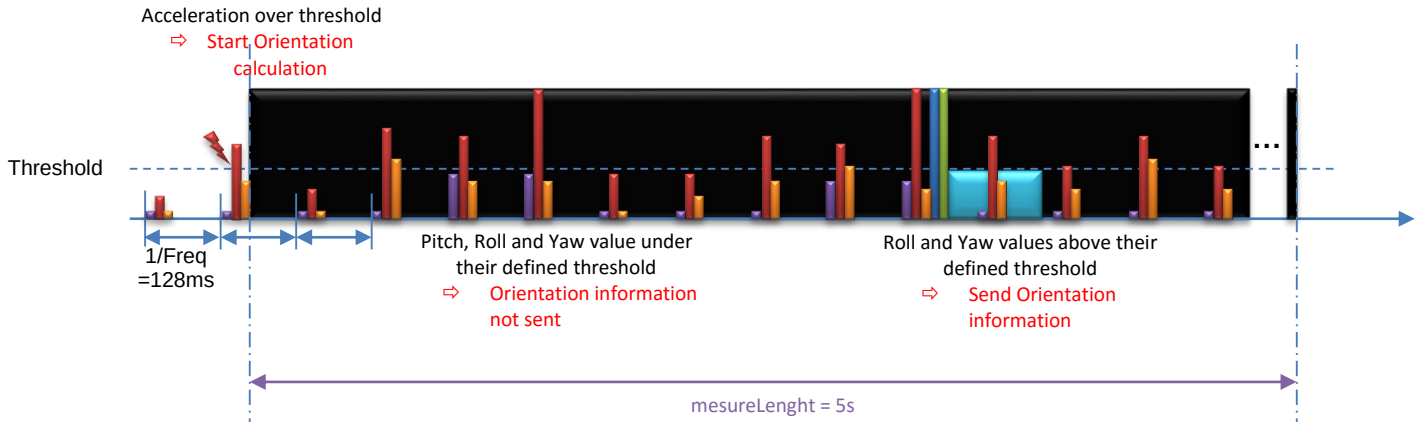

Z Axis
Pitch
+
-
X Axis
Y Axis
Z Axis
Roll
+
-
Y Axis
X Axis
Yaw
Y Axis
+
-
Z Axis
X Axis

ORIENT implementation



Example 1: *period = 30*; modeRefresh = **0**; modeOnMove = **1**; threshold = 200; mesureLenght = 5; pitch = +10; roll = +10; yaw = -14 (*parameter ignored*)

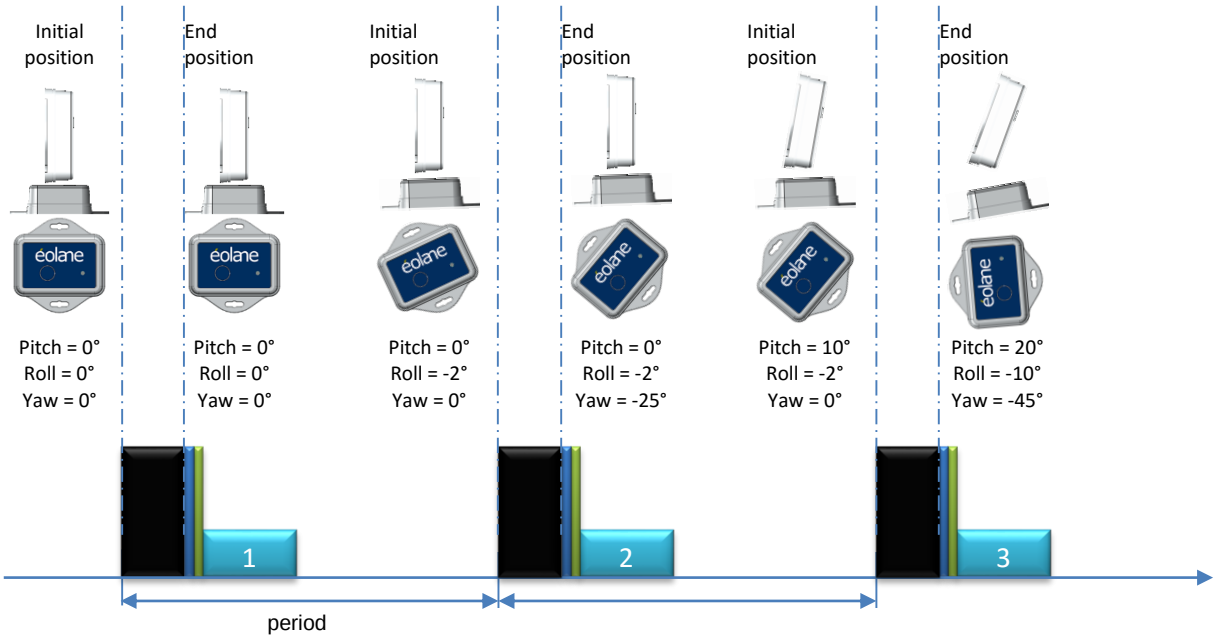




LoRa frame sent: **c11510000000bfff1aa**

- Payload header: **c115**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x15**: Temperature = 21°C
- Payload data : **100000000bfff1**
  - **0x10**: Data type = ORIENTATION => expected payload data = 2 bytes Pitch (signed) + 2 bytes Roll (signed) + 2 bytes Yaw (signed)
    - **0x0000**: Pitch angle value = 0 x 1° = 0°
    - **0x000B**: Roll angle value = 11 x 1° = +11°
    - **0xFFF1**: Yaw angle value = -15 x 1° = -15°
- Payload end of frame: **aa**
  - **0xAA**: End of frame

Example 2: period = 30; modeRefresh = **0**; modeOnMove = **1**; *threshold = 200*; mesureLenght = 5; *pitch = +10; roll = +10; yaw = -14* (*parameter ignored*)



LoRa frame "1" sent: **c115100000000000000aa**

- Payload header: **c115**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x15**: Temperature = 21°C
- Payload data : **100000000000000**
  - **0x10**: Data type = ORIENTATION => expected payload data = 2 bytes Pitch (signed) + 2 bytes Roll (signed) + 2 bytes Yaw (signed)
    - **0x0000**: Pitch angle value = 0 x 1° = 0°
    - **0x0000**: Roll angle value = 0 x 1° = 0°
    - **0x0000**: Yaw angle value = 0 x 1° = 0°
- Payload end of frame: **aa**
  - **0xAA**: End of frame

LoRa frame “2” sent: **c115100000fffeffe7aa**

- Payload header: **c115**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x15**: Temperature = 21°C
- Payload data : **100000fffeffe7**
  - **0x10**: Data type = ORIENTATION => expected payload data = 2 bytes Pitch (signed) + 2 bytes Roll (signed) + 2 bytes Yaw (signed)
    - **0x0000**: Pitch angle value = 0 x 1° = 0°
    - **0xFFFE**: Roll angle value = -2 x 1° = -2°
    - **0xFFE7**: Yaw angle value = -25 x 1° = -25°
- Payload end of frame: **aa**
  - **0xAA**: End of frame

LoRa frame “3” sent: **c115100014fff6ffd3aa**

- Payload header: **c115**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x15**: Temperature = 21°C
- Payload data : **100014fff6ffd3**
  - **0x10**: Data type = ORIENTATION => expected payload data = 2 bytes Pitch (signed) + 2 bytes Roll (signed) + 2 bytes Yaw (signed)
    - **0x0014**: Pitch angle value = 20 x 1° = +20°
    - **0xFFF6**: Roll angle value = -10 x 1° = -10°
    - **0xFFD3**: Yaw angle value = -45 x 1° = -45°
- Payload end of frame: **aa**
  - **0xAA**: End of frame

3.2.9. VIBE algorithm

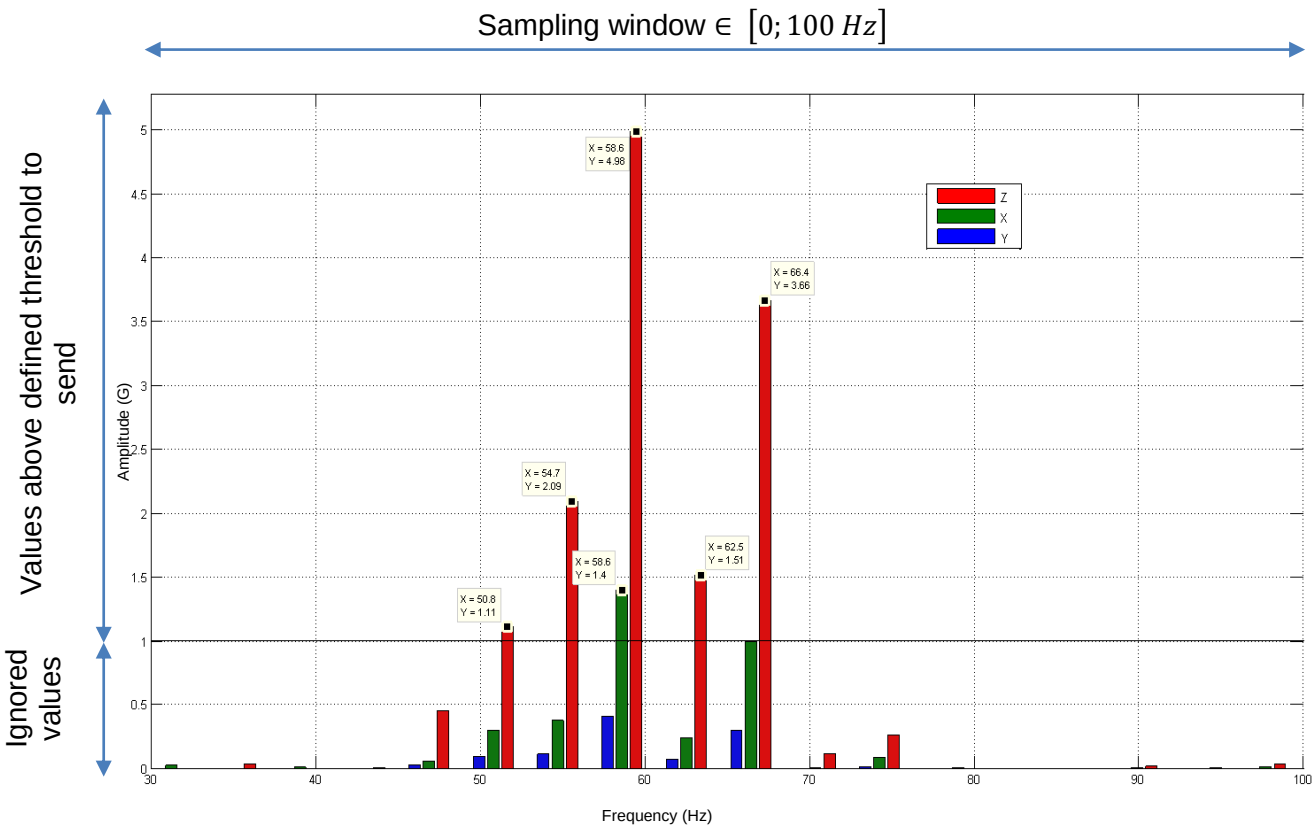
VIBE parameters

	Parameter	Description	Valeur			Préci-sion	Unité
			min	typ	max		
VIBE	onX	Activates vibration measurement on X-Axis	0	1	1	N/A	boolean
	onY	Activates vibration measurement on Y-Axis	0	1	1		
	onZ	Activates vibration measurement on Z-Axis	0	1	1		
	period	Minimum period between 2 measures	1	60	65535	s	s
	amplitudeX	X axis vibration amplitude threshold	100	1000	65535	mG	G
	amplitudeY	Y axis vibration amplitude threshold	100	1000	65535	mG	G
	amplitudeZ	Z axis vibration amplitude threshold	100	1000	65535	mG	G
	freqMin	Minimum frequency (lower limit for the sampling window)	0	0	400	Hz	Hz
	freqMax	Maximum frequency (upper limit for the sampling window)	100	250	500	Hz	Hz

See *§3.4 LoRa frames payload description* for payload decoding

VIBE implementation

Example: onX = 1; onY = 1; onZ = 1; period = 60; amplitude = 1000; amplitude = 1000; amplitudeZ = 1000; freqMin = 0; freqMax = 100



LoRa frame sent: **c11a86020456 0d02082a0e0005780f0213740f0205e610020e4c11aa**

- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **86020456 0d02082a0e0005780f0213740f0205e610020e4c11**
  - **0x86**: Data type = VIBE => expected payload data = N x **Axis (1 byte)**, **Amplitude (2 bytes unsigned)**, **frequency index (1 byte unsigned)**
    - **0x02**: Z Axis
      - **0x0456**: amplitude = 1110mG
        - **0D**: index = 13 => frequency = 3,90625 x 13 = 50,8HZ
    - **02**: Z Axis
      - **082A**: amplitude = 2090mG
        - **0E**: index = 14 => frequency = 3,90625 x 14 = 54,7HZ
    - **00**: X Axis
      - **0578**: amplitude = 1400mG
        - **0F**: index = 15 => frequency = 3,90625 x 15 = 58,6HZ
    - **02**: Z Axis
      - **1374**: amplitude = 4980mG
        - **0F**: index = 15 => frequency = 3,90625 x 15 = 58,6HZ
    - **02**: Z Axis
      - **05E6**: amplitude = 1510mG
        - **10**: index = 16 => frequency = 3,90625 x 16 = 62,5HZ
    - **02**: Z Axis
      - **0E4C**: amplitude = 3660mG
        - **11**: index = 17 => frequency = 3,90625 x 17 = 66,4HZ
- Payload end of frame: **aa**
  - **0xAA**: End of frame

3.3. Algorithms compatibility

In order to guarantee that the device is fully functional, a limitation on the number of algorithm activated at the same time has been set.

The following table shows which algorithm with which other algorithm. The table must be read from left to right, and when a green arrow is encountered from top to bottom.

		VIBE	ALIVE		SHOCK	MOTION	TEMP		TILT		ROTATION		ORIENT	
	Mode	-	Auto	Manual	-	-	Continuous	Threshold	Refresh	Motion	Refresh	Motion	Refresh	Motion
VIBE	-	Yes	X	X	X	X	X	X	X	X	X	X	X	X
ALIVE	Auto	No	X	X	X	X	X	X	X	X	X	X	X	X
	Manual	Yes	Yes	X	X	X	X	X	X	X	X	X	X	X
SHOCK	-	No	Yes	Yes	X	X	X	X	X	X	X	X	X	X
MOTION	-	No	Yes	Yes	Yes	X	X	X	X	X	X	X	X	X
TEMPERATURE	Continuous	No	No	Yes	Yes	Yes	X	X	X	X	X	X	X	X
	Threshold	No	No	Yes	Yes	Yes	No	X	X	X	X	X	X	X
TILT	Refresh	No	No	Yes	Yes	Yes	No	No	X	X	X	X	X	X
	Motion	No	Yes	Yes	No	No	Yes	Yes	Yes	X	X	X	X	X
ROTA-TION	Refresh	No	No	Yes	Yes	Yes	No	No	No	No	X	X	X	X
	Motion	No	Yes	Yes	No	No	Yes	Yes	No	No	Yes	X	X	X
ORIENT	Refresh	No	No	Yes	Yes	No	No	No	No	Yes	No	Yes	X	X
	Motion	No	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes	No	Yes	X

Examples:

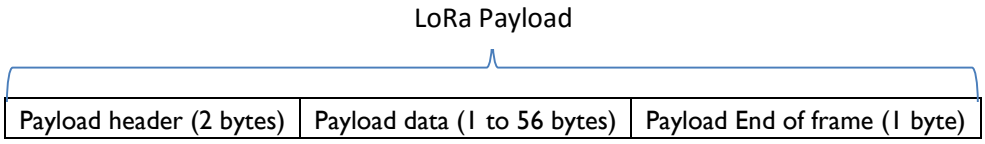
- ALIVE algorithm can be activated at the same time as any other algorithm
- TILT algorithm in Motion detection mode can be activated at the same time as
  - ALIVE (auto or manual mode),
  - Or TEMP (continuous or Threshold mode),
  - Or TILT in Refresh mode,
  - Or ORIENTATION in Refresh mode,

    but NOT at the same time as
  - SHOCK,
  - MOTION
  - ROTATION (Refresh or Motion detection mode)
  - ORIENTATION in Motion Detection mode

3.4. LoRa frames payload description

The data are sent in the payload of LoRa frames. A specific payload layout has been defined in order to detect which algorithm is running, and allows to retrieve the corresponding data.

The payload will then change depending on which algorithm is running on the product. The payload structure is defined as:



3.4.1. Payload header

The payload header is always sent, and contains the battery level and the temperature measures:

Name	**Battery level**	**Temperature**
Size	1 byte	1 byte
Coding	8 bits (unsigned)	8 bits (signed)
Coded value:	2,8V…3,6V	-127°C…+128°C

The battery level is coded on the first byte (MSB) as an unsigned byte, the value is then between 0 and 255.
0 is for Vbat=2.8V, and 255 is for Vbat=3.6V.
Each step equals $\frac{(3,6-2,8)}{255} \cong 3,14\ mV$

The temperature is coded on the second byte (LSB) as a signed byte, the value is between -128 and +127. The value is in Celsius degrees.

3.4.2. Payload data

The payload data is made of an MSB "data type" byte (unsigned) describing the algorithm sending the data, followed by the corresponding data.

Data type		Data 1			Data 2			Data 3		
Name	**Value**	**Name**	**Coding**	**Unit**	**Name**	**Coding**	**Unit**	**Name**	**Coding**	**Unit**
ALIVE	0x01	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
TEMPERA-TURE[6]	0x02	Temp1	8 bits (signed)	°C	Temp2	8 bits (signed)	°C	Temp3	8 bits (signed)	N/A
SHOCK	0x04	Gx	16 bits (signed)	mG	Gy	16 bits (signed)	mG	Gz	16 bits (signed)	N/A
TILT	0x08	Pitch	16 bits (signed)	0.1°	Roll	16 bits (signed)	0.1°	N/A	N/A	N/A
ORIENT	0x10	Pitch	16 bits (signed)	1°	Roll	16 bits (signed)	1°	Yaw	16 bits (signed)	1°
MOTION	0x20	Motion n	0x00	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		Stillness	0x01	N/A	N/A	N/A	N/A	N/A	N/A	N/A
ACTIVITY	0x40	Motion	0x00	N/A	N/A	N/A	N/A	N/A	N/A	N/A
		Stillness	0x01	N/A	Activity duration	32 bits (unsigned)	ms	N/A	N/A	N/A
ROTATION	0x80	Number of turns	16 bits (signed)	turns	N/A	N/A	N/A	N/A	N/A	N/A
VIBRATION[7]	0x86	X axis	0x00	N/A	Vibration amplitude	16 bits (unsigned)	mG	Index[8] (*i*) of the vibration frequency	8 bits (unsigned)	N/A
		Y axis	0x01	N/A						
		Z axis	0x02	N/A						
INFORMATION	0xFE	Product version	ASCII	N/A	SW version	ASCII	N/A	LoRa stack version	ASCII	N/A
SERVICE	0xFF	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

[6] TEMPERATURE algorithm can send up to 50 successive values.
Temp1 temperature value is the latest temperature measurement before the transmission of the data.
TempN temperature values are the recorded values since the last transmission.
[7] For vibration frames, the number of data depends on the result of the measure. Maximum size for a vibration frame is 48 bytes (12 x data packets)
[8] Frequency is retrieved with the following formula: $f = 3{,}90625 * i$ Hz

3.4.3. Payload End of frame

The end of the payload is defined as a byte with a specific value: 0xAA

Size	1 byte
Coding	-
Value:	0xAA

3.4.4. Payload examples

ALIVE payload

Payload example: **e61801aa**

- Payload header: **e618**
  - **0xE6**: battery level : 0xE6 = 230: $\frac{(3,6-2,8)}{255} * 230 + 2,8 = 3,52\ Volts$
  - **0x18**: Temperature = 24°C
- Payload data : **01**
  - **0x01**: Data type = ALIVE => no more payload data
- Payload end of frame: **aa**
  - **0xAA**: End of frame

SHOCK algorithm payload

Payload example: **c11a040290fe700db0aa**

- Payload header: **c11a**
  - **0xC1**: battery level : 0xC1 = 193: $\frac{(3,6-2,8)}{255} * 193 + 2,8 = 3,4\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **040290fe700db0**
  - **0x04**: Data type = SHOCK => expected payload data = Gx (2 bytes unsigned), Gy (2 bytes unsigned), Gz (2 bytes unsigned)
    - **0x0290FE700DB0**
      - **0x0290**: Gx = 656mG
      - **0xFE70**: Gy = -400mG
      - **0x0DB0**: Gz = 3504mG
- Payload end of frame: **aa**
  - **0xAA**: End of frame

TEMPERATURE algorithm payload

Payload example: **be1a021a1919191919191919aa**

- Payload header: **be1a**
  - **0xBE**: battery level : 0xBE = 230: $\frac{(3,6-2,8)}{255} * 190 + 2,8 = 3,40\ Volts$
  - **0x1A**: Temperature = 26°C
- Payload data : **021a1919191919191919**
  - **0x02**: Data type = TEMPERATURE => expected payload data = N bytes (signed)
    - **0x1A1919191919191919**
      - **0x1A**: temp = 26°C (latest measure)
      - **0x19**: temp = 25°C
      - **0x19**: temp = 25°C
      - **0x19**: temp = 25°C
      - **0x19**: temp = 25°C
      - **0x19**: temp = 25°C
      - **0x19**: temp = 25°C
      - **0x19**: temp = 25°C
      - **0x19**: temp = 25°C
- Payload end of frame: **aa**
  - **0xAA**: End of frame

VERSION payload (after a VERSION Downlink request)

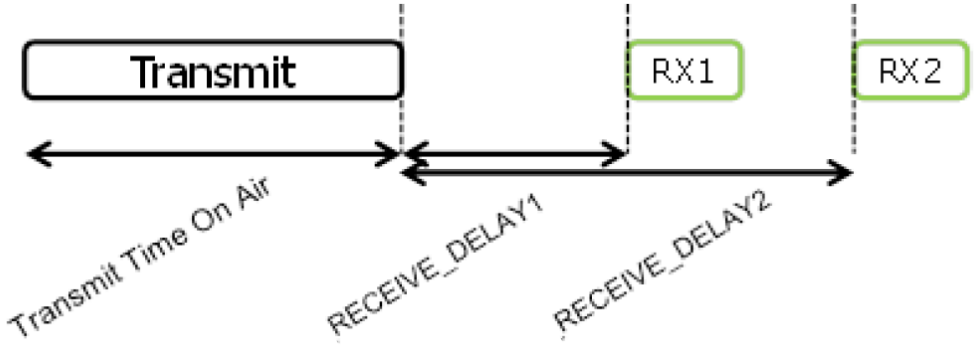
Payload example: **8a19fe76312e323b76312e372062657461343b76342e332e3020726331aa**

- Payload header: **8a19**
  - **0x8A**: battery level : 0x8A = 138: $\frac{(3,6-2,8)}{255} * 138 + 2,8 = 3,125\ Volts$
  - **0x19**: Temperature = 25°C
- Payload data : **fe76312e323b76312e372062657461343b76342e332e3020726331**
  - **0xFE**: Data type = VERSION => expected payload data = Product version (N bytes); SW version (N bytes); LoRa Stack version (N bytes)
    - **0x76312E323B76312E372062657461343B76342E332E3020726331**
      - **0x76312E323B**: Product version = “v1.2;”
      - **0x76312E373B**: SW version = “v1.7;”
      - **0x76342E332E3020726331**: “v4.3.0 rc1”
- Payload end of frame: **aa**
  - **0xAA**: End of frame

3.5. Downlink

Only available on Product version 1.2 and above (SW version 1.7 and above)

LoRa™ protocol offers a bidirectional link between the sensor and the LoRa™ Core Network Server. The data transfer is always initiated by the sensor, sending an uplink to the LoRa™ Core Network Server. Once the sensor has sent the uplink, it will open 2 receive windows, which can be used by the LoRa™ Core Network Server to initiate a Downlink transmission.



Movee is able to get parameters updates Over The Air (OTA), through LoRa™ downlink frame transmission. The downlink frame transmission can not only change the parameters, but also the algorithm running on the product.
The downlink cannot be used to update the embedded firmware of the product.

Downlink frame transmission allows 2 actions:

- Change the parameters and running algorithm
- Execute a remote action on the product (e.g.: save parameters, restart the product…)

3.5.1. Ports

Commands are received/sent on port 1.

Parameters are received/sent on port 2.

3.5.2. Commands

It is possible to send up to 3 commands in a single downlink frame. Upon reception, the product will execute these commands one after another (starting by Command ID#1). The following table describes the downlink frame format to send commands to the product:

Name	Command ID #1	Command ID #2	Command ID #3	0xFF
Syze	1 byte	1 byte	1 byte	1 byte
	Mandatory	*Option*	*Option*	Mandatory

The following table gives the command list:

Command ID	**Name**	**Description**
0	Reserved	Reserved
1	Save	Save the updated parameters on the internal flash
2	Clean parameters	Erase all the parameters loaded in RAM. Each parameter is then set to 0
3	Reset parameters	Restaure default parameters
4	Reset board	Reset the product
5	Service mode	Set the product in service mode
6	Normal mode	Restaure normal mode of operation (= exit service mode)
7	Version	Send a frame with product version details
8	RFU	
9	RFU	
A	Dump RAM param	Dump the parameters loaded in RAM on the serial output
B	Dump FLASH param	Dump the parameters saved in flash on the serial output
C	Enable debug	Enable detailed debug traces on debug output
D	Disable debug	Disable detailed debug traces on debug output
E	Enable LED	Enable LED activity with motion detection
F	Disable LED	Disable LED activity with motion detection
10	ADR On	Enable ADR for LoRa communication
11	ADR Off	Disable ADR for LoRa communication
12	ACK On	Enable confirmed frames for LoRa communication
13	ACK off	Disable confirmed frames for LoRa communication

3.5.3. Parameters

It is possible to send up to 10 parameters in a single frame. Upon reception, the product will modify the parameters one by one. The following table describes the downlink frame format to change parameters on the product

Name	Parameter #1		Parameter #2		…	Parameter #10		EOF[9]
Content	ID	Valeur	ID	Valeur		ID	Valeur	0xFF
Size (bytes)	1	4	1	4	…	1	4	1
Status	Mandatory		*Optional*		…	*Optional*		Mandatory

Note: the size of the received payload is between 6 and 51 bytes

The time relative parameters (period, timer) are to be defined in ms.

[9] EOF = End Of Frame

The following table gives the parameters list:

ID (hexa)	Algorithm	Name
1	MPU	maxRange
2	Algo	choseAlgo[10]
B	Alive	modeRefresh
C	Alive	period
D	Alive	nbSavedValue
13	Shock	gxSup
14	Shock	gxInf
15	Shock	gySup
16	Shock	gyInf
17	Shock	gzSup
18	Shock	gzInf
19	Shock	freq
1A	Shock	inhibition
1B	Shock	removeGravity
21	Motion	gxSup
22	Motion	gxInf
23	Motion	gySup
24	Motion	gyInf
25	Motion	gzSup
26	Motion	gzInf
27	Motion	Freq
28	Motion	timerA
29	Motion	timerB
2A	Motion	sensitivity
2B	Motion	activity
2C	Motion	additionateActivity
2D	Motion	periodicActivity
2E	Motion	activityResumePeriod
2F	Motion	loraAtStartMvt
30	Motion	loraAtStopMvt
36	Temp	nbSavedValue
37	Temp	modeThreshold
38	Temp	max
39	Temp	min
3A	Temp	delta
3B	Temp	period
3C	Temp	fastPeriod
3D	Temp	ultraFastPeriod
3E	Temp	inhibition
44	Tilt	modeRefresh
45	Tilt	modeOnMove
46	Tilt	pitch
47	Tilt	roll
48	Tilt	threshold
49	Tilt	inhibition
4A	Tilt	period
50	Rotation	modeRefresh
51	Rotation	modeOnMove
52	Rotation	lap
53	Rotation	resetLap
54	Rotation	threshold
55	Rotation	period
5B	Orient	period
5C	Orient	modeRefresh
5D	Orient	modeOnMove
5E	Orient	threshold
5F	Orient	mesureLenght
60	Orient	pitch
61	Orient	roll
62	Orient	yaw
68	Vibe	onX
69	Vibe	onY
6A	Vibe	onZ
6B	Vibe	period
6C	Vibe	amplitudeX
6D	Vibe	amplitudeY
6E	Vibe	amplitudeZ
6F	Vibe	freqMin
70	Vibe	freqMax

[10] See: *§3.5.4 Algorithm selection*.

3.5.4. Algorithm selection

The parameters allow the selection of the algorithm(s) running on the product, **please refer to § *3.3 Algorithms compatibility* for algorithm mutual compatibility**.

In order to modify the activated algorithm(s), it is necessary to code the *chosealgo* parameter value, which is coded on 9 bits:

Bit	8	7	6	5	4	3	2	1	0
Name	VIBE	ROTATION	Reserved	MOTION	ORIENT	TILT	SCHOCK	TEMP	ALIVE

3 examples on how to code the *chosealgo* parameter:

- ALIVE and SCHOCK activation: *chosealgo* = $1 * 2^0 + 1 * 2^2$ = 0x05
- MOTION and SHOCK activation: *chosealgo* = $1 * 2^2 + 1 * 2^5$ = 0x24
- VIBE activation : *chosealgo* = $1 * 2^8$ = 0x100

Please refer to *§3.3 Algorithms compatibility* for algorithm mutual compatibility.

3.5.5. Downlink examples

Enter Service mode

If you want to change some parameters, you can either send the parameters directly on port N°2, or use the service mode to send a long list of parameters.

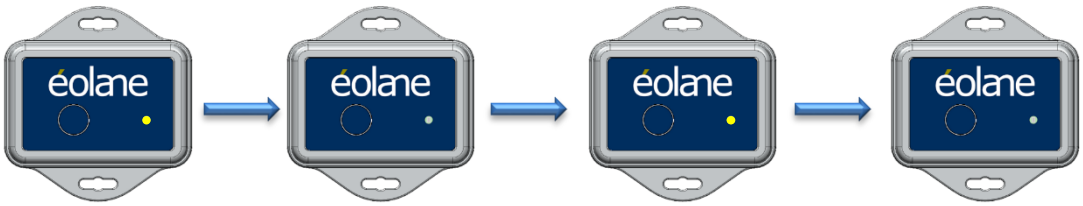
Entering service mode will force the Movee to send uplink frames every ~2 minutes, in order to get the ability to send the needed Downlink messages as fast as possible.

Activate Service mode
=> Service mode command ID = **0x05**
=> End of Downlink message = **0xFF**

In order to enter Service mode, send:

05FF on port #1 (commands are sent on port #1, parameters on port #2)

Once the downlink is received by the device, it will start blinking yellow.



The device will keep blinking as long as it has not exited the Service mode.

Note: Before asking the network to send another downlink, make sure that the “enter service mode” downlink has been sent by the network. If you do not wait for the first downlink to be sent, the network will cancel the first downlink, and replace it with the new one.

Send parameters

Shock and Alive example

In this example, we will activate Shock and Alive modes, with the following configuration:

Alive:

- modeRefresh parameter: Refresh mode activated
- period parameter: period set to 10 minutes
- nbSavedValue parameter: One period between 2 LoRa frames, with temperature measure for each period.

MPU:

- maxRange parameter: set maximum range to 8000mG

Shock:

- gxSup, gxInf, gySup, gyInf, gzSup, gzInf parameters: set threshold to 1250mG on each axis
- freq parameter: not changed (use previously stored value)
- inhibition parameter: de-activate inhibition (set to 0)
- removeGravity

Note: If parameters are not updated, default or previously stored values will be used.

Alive parameter frame

Activate Shock and Alive algorithms
=> choseAlgo ID = **0x02** / Parameter = **0x00000005** (see *§3.5.4 Algorithm selection*)

Set modeRefresh parameter to 1
=> modeRefresh ID = **0x0B** / Parameter = **0x00000001**

Set the period value to 10 minutes = 600 000 ms (**Reminder, period value has to be set in ms**)
=> period ID = **0x0C** / Parameter = **0x000927C0**

Set the number of period between 2 frames to 1
=> nbSavedValue ID = **0x0D** / Parameter = **0x00000001**
=> End of Downlink message = **0xFF**

In order to set this configuration, send (**Reminder: maximum 10 parameters in a single downlink frame**):

02000000050B00000001 0C000927C0 0D00000001FF on port #2 (commands are sent on port #1, parameters on port #2)

MPU & Shock parameter frame

Set MPU range to 8000mG: Shock + Alive
=> maxRange ID = **0x01** / Parameter = **0x00001F40**

Set gxSup, gxInf, gySup, gyInf, gzSup, gzInf parameters to 1250mG
=> gxSup ID = **0x13** / Parameter = **0x00004E2**
=> gxInf ID = **0x14** / Parameter = **0x00004E2**
=> gySup ID = **0x15** / Parameter = **0x00004E2**
=> gyInf ID = **0x16** / Parameter = **0x00004E2**
=> gzSup ID = **0x17** / Parameter = **0x00004E2**
=> gzInf ID = **0x18** / Parameter = **0x00004E2**

Set the inhibition time to 0
=> inhibition ID = **0x1A** / Parameter = **0x00000000**

In order to set this configuration, send (**Reminder: maximum 10 parameters in a single downlink frame**):

0100001F4013000004E214000004E215000004E216000004E217000004E218000004E21A00000000FF on port #2 (commands are sent on port #1, parameters on port #2)

Exit service mode

Once the parameters are received by the device, they are stored in RAM. In order to make these new parameters persistent when the device is switched off or reset, it is necessary to send the command to write the new configuration in flash (Command ID = 0x01).

Save parameters and exit Service mode
=> Save command ID = **0x01**
=> Normal mode (= exit Service mode) command ID = **0x06**
=> End of Downlink message = **0xFF**

In order to save the new parameters in flash and exit Service mode, send:

0106FF on port #1 (commands are sent on port #1, parameters on port #2)

3.6. Movee Configurator user interface

3.6.1. PC User interface install

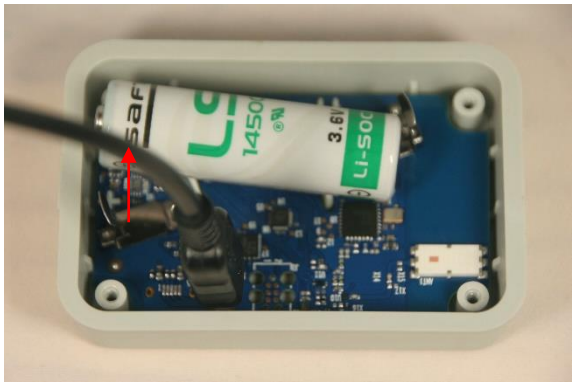
To install the user interface on your computer, please download and execute the "MoveeConfigurator_setup_vx.x.exe" available on the shared cloud directory:
https://transfer.eolane.com/public.php?service=files&t=7bbed3de4307af52e06302090c565373

Pwd = eolane

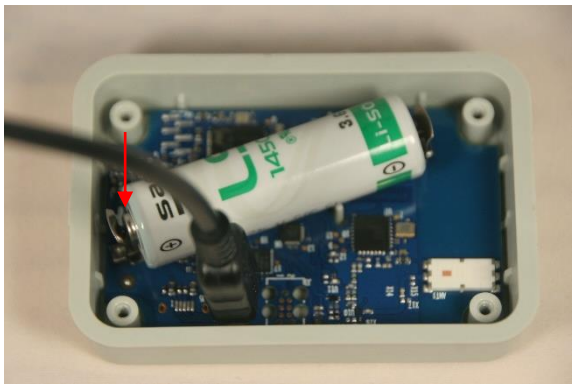
3.6.2. Setup

To access the internal debug interface, remove the 4 screws underneath the product:

Once opened, remove the battery and plug a micro-USB cable on the micro-USB connector:

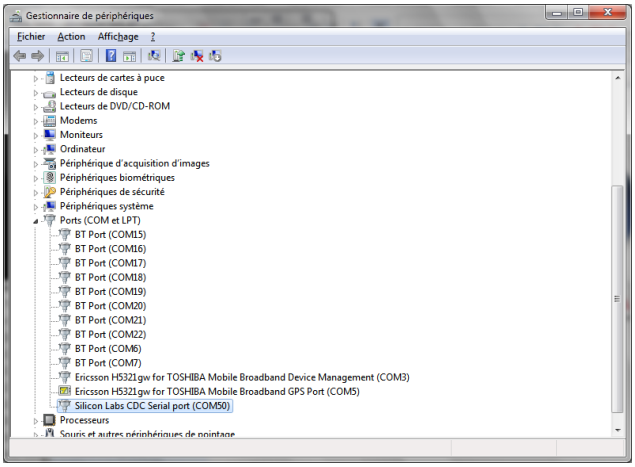


Plug the other side of the Micro-USB cable to a host computer, and put the battery back in place.

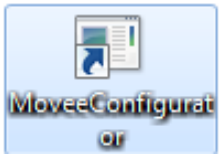


Once plugged on the host computer (where you should have installed the configuration “Movee configurator” PC user interface, with the Silicon Labs USB CDC/ACM driver) switch the device on (1s push on the button)

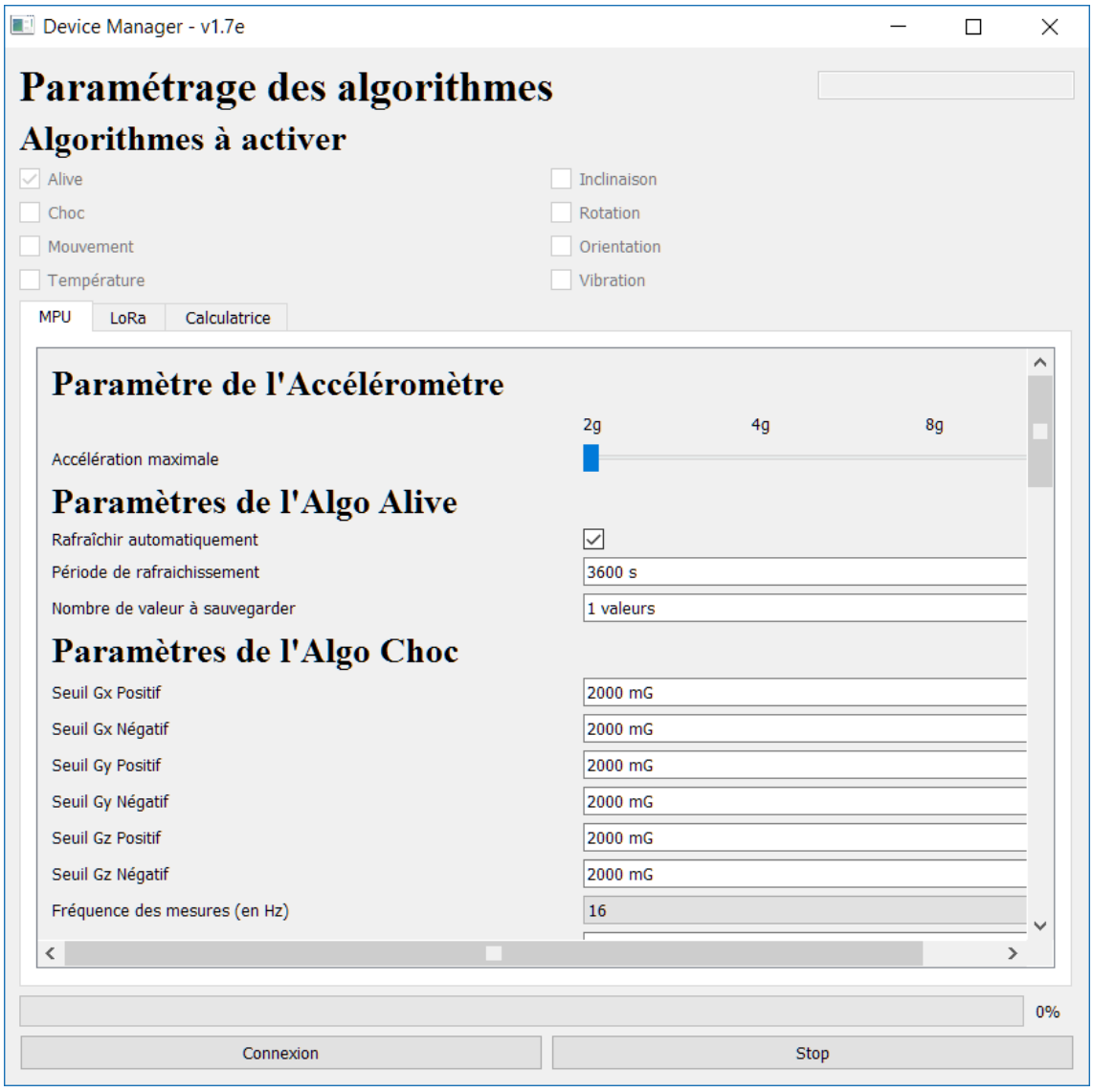
You should then see the device in the COM ports of your computer:



Open the “Movee Configurator” tool



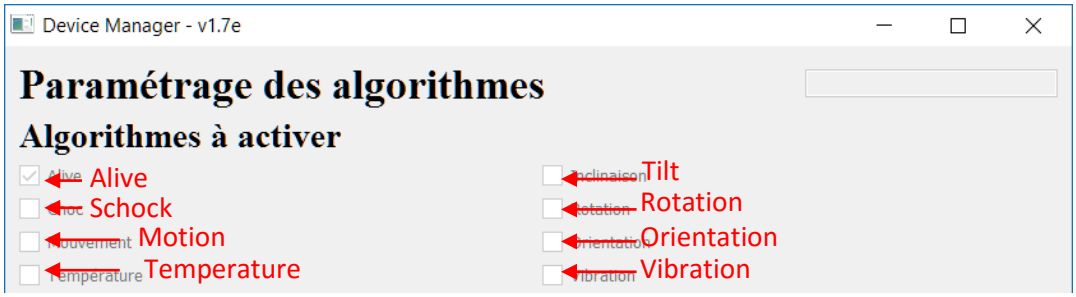
The UI is displayed on your screen:



3.6.3. User interface

Algorithm selection

The upper section of the UI allows the selection of the algorithms. **Please refer to § *3.3 Algorithms compatibility* for algorithm mutual compatibility**.

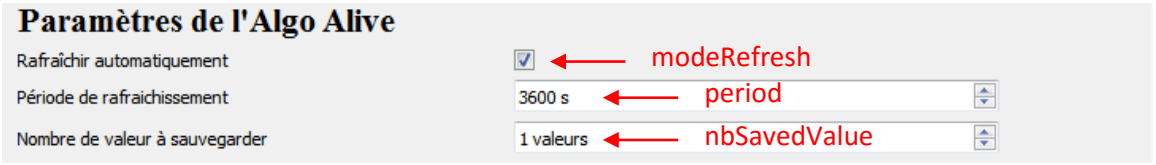


MPU tab

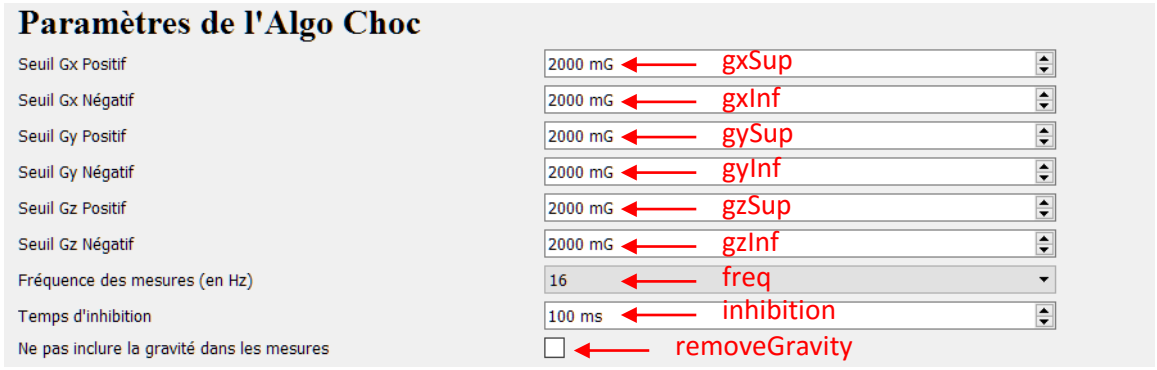
The MPU tab allows the setting of the algorithms parameters

MPU parameters

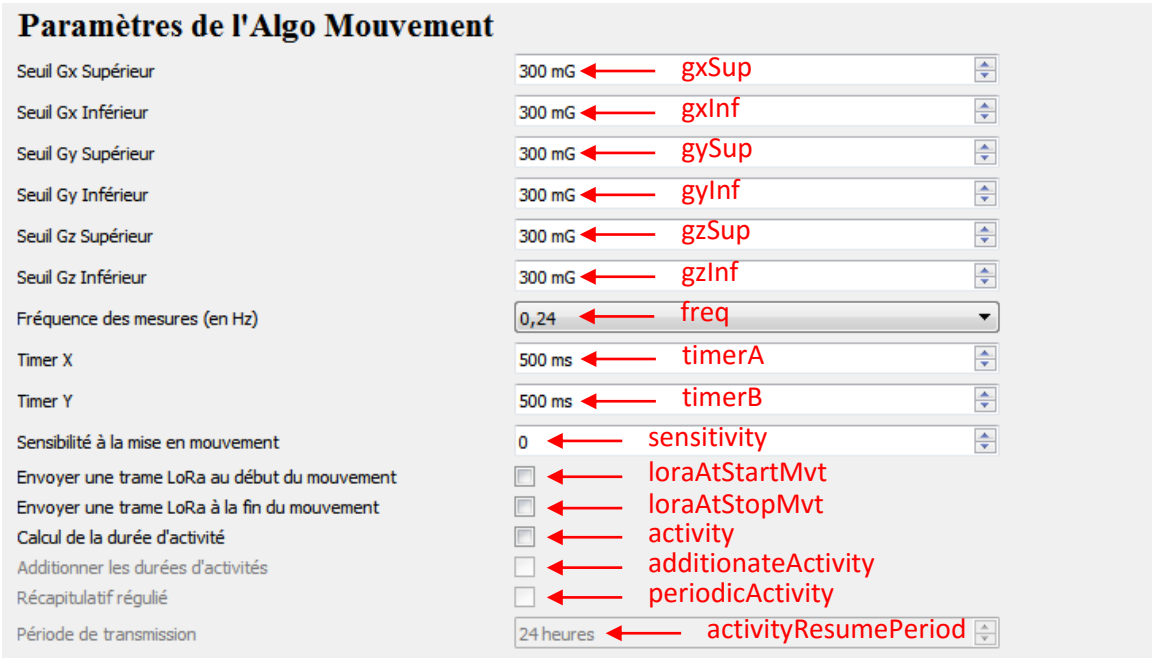
ALIVE parameters



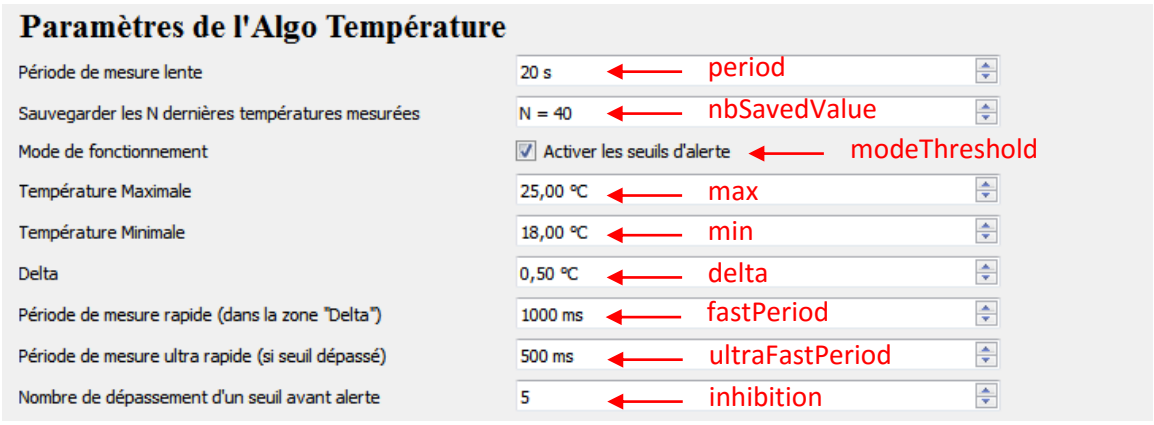
SHOCK parameters



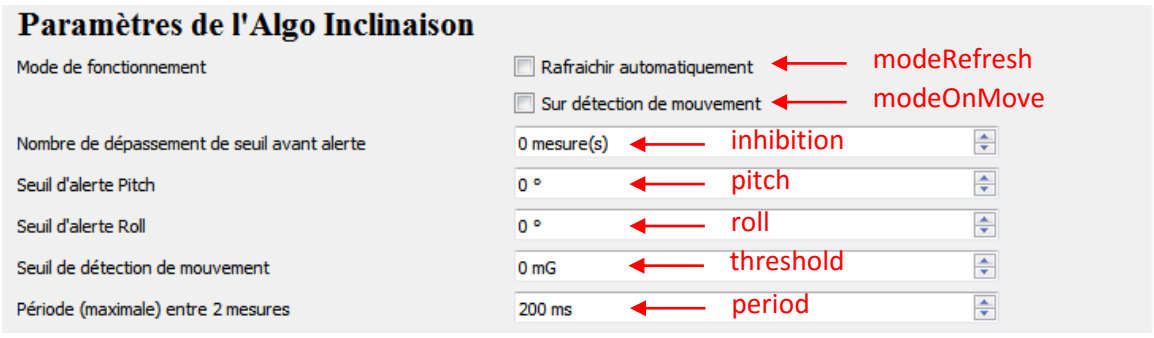
MOTION parameters



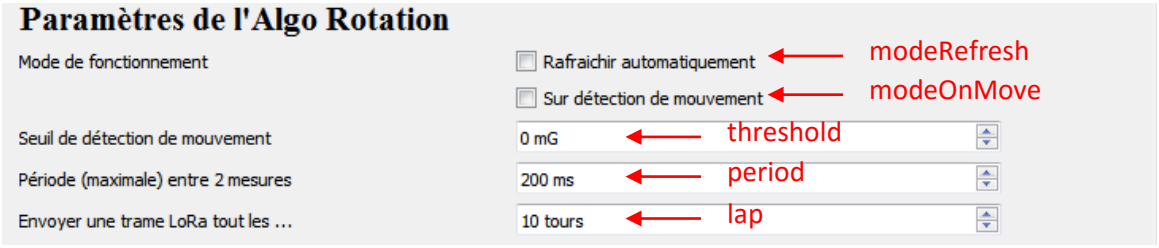
TEMPERATURE parameters



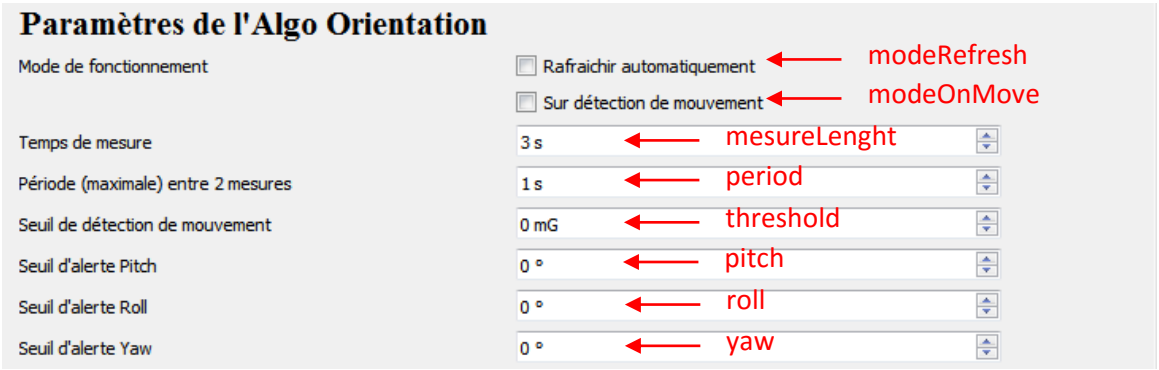
TILT parameters



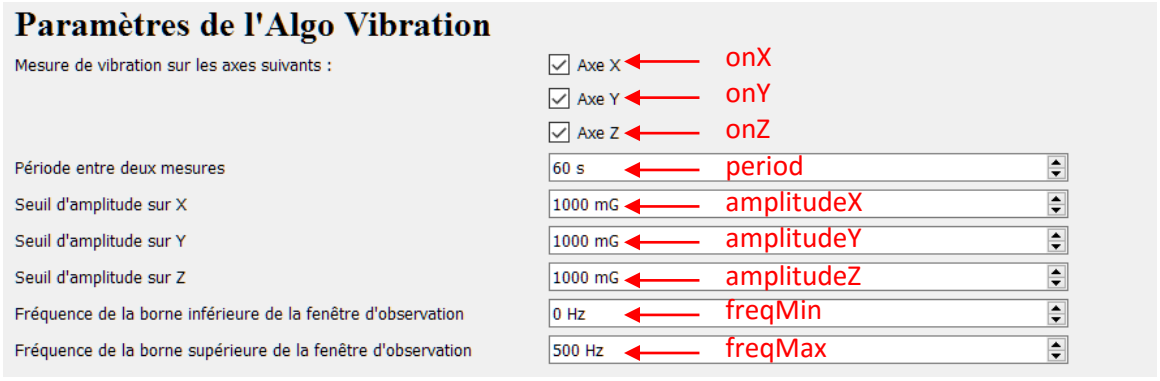
ROTATION parameters



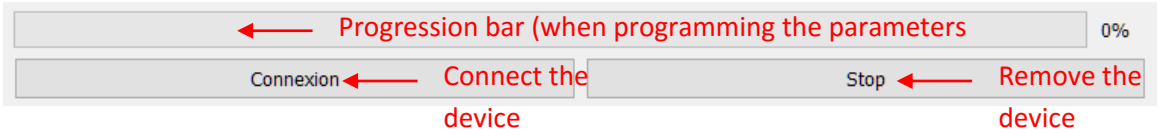
ORIENTATION parameters



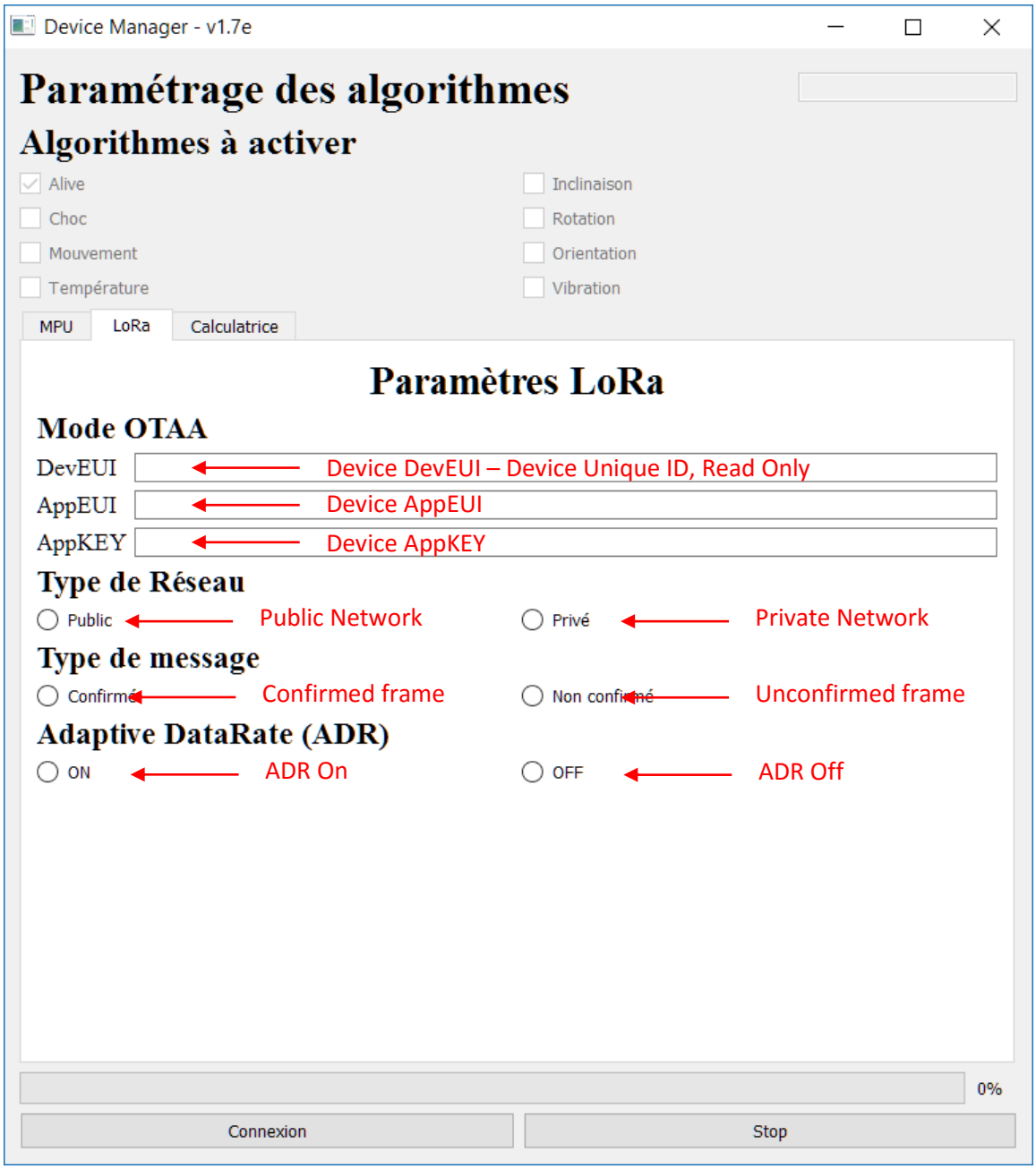
VIBRATION parameters



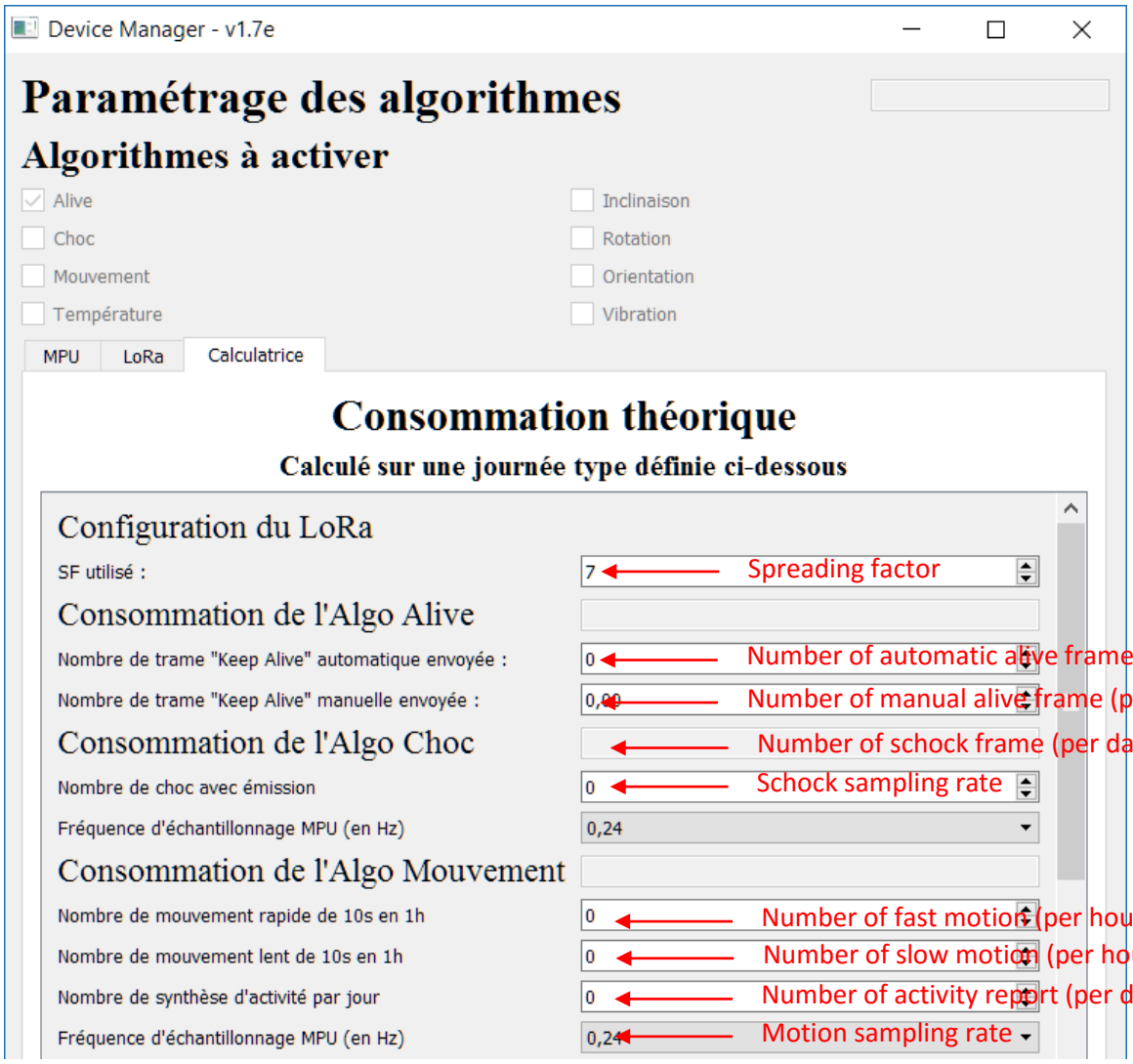
Status bar

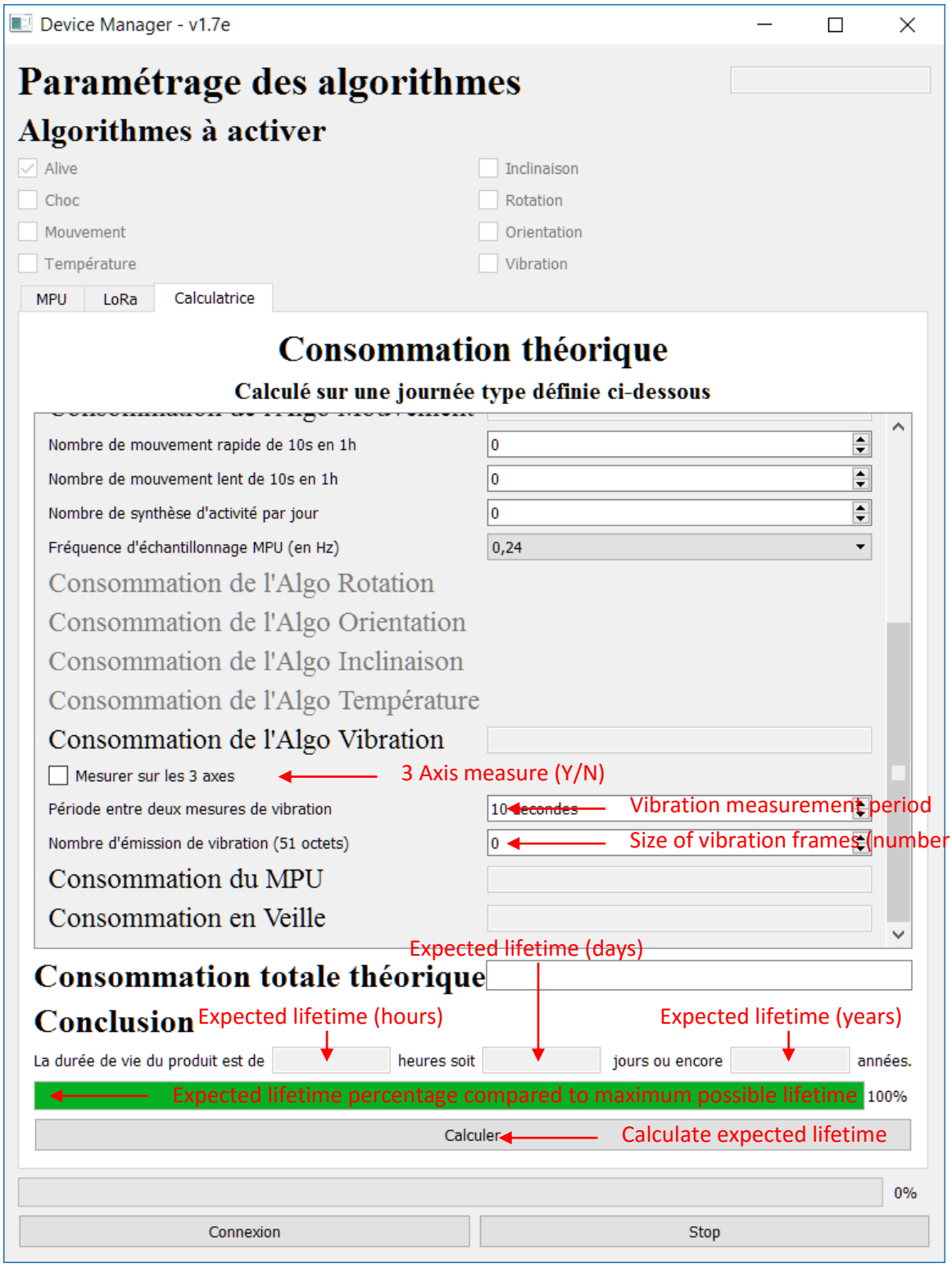


LoRa settings tab



LoRa calculator




Device Manager - v1.7e
Paramétrage des algorithmes
Algorithmes à activer
Alive
Choc
Mouvement
Température
Inclinaison
Rotation
Orientation
Vibration
MPU
LoRa
Calculatrice
Consommation théorique
Calculé sur une journée type définie ci-dessous
Nombre de mouvement rapide de 10s en 1h
Nombre de mouvement lent de 10s en 1h
Nombre de synthèse d'activité par jour
Fréquence d'échantillonnage MPU (en Hz)
0,24
Consommation de l'Algo Rotation
Consommation de l'Algo Orientation
Consommation de l'Algo Inclinaison
Consommation de l'Algo Température
Consommation de l'Algo Vibration
Mesurer sur les 3 axes
3 Axis measure (Y/N)
Période entre deux mesures de vibration
10 secondes
Vibration measurement period
Nombre d'émission de vibration (51 octets)
Size of vibration frames (number of bytes)
Consommation du MPU
Consommation en Veille
Expected lifetime (days)
Consommation totale théorique
Conclusion
Expected lifetime (hours)
Expected lifetime (years)
La durée de vie du produit est de
heures soit
jours ou encore
années.
Expected lifetime percentage compared to maximum possible lifetime
100%
Calculer
Calculate expected lifetime
0%
Connexion
Stop

IV. Appendix

4.1. Revision history

Revision	Modifications
0.50	First edition
0.51	Typo corrections
0.9	Add Vibration algorithm details, LoRa frame examples, SW upgrade to v1.7 (downlink integration)
1.00	Add Tilt, Rotation and Orientation algorithm details
1.01	Correction on TILT, ROTATION and ORIENTATION frame examples (incorrect number of bytes in frame examples) Update operating temperature, and add storage temperature information
1.02	Correction on TILT frame examples (wrong scale for pitch and roll values)
1.03	Add Downlink commands
1.04	Correction on Shock data type (unsigned to signed) in §3.2.3
1.05	Add Downlink examples chapter Update Movee configurator user interface chapter