NEON Contact Sensor

Communication Protocol v1

Version A
23-11-2021
By: NY

NEON-Contact-Sensor_Communication-Protocol-v1_VS-xxx-01-CSxx_4003_1_A

Contents

1 General

- All variables are little endian.
- All application messages are sent through port 15.

2 Message Schemes

2.1 Activation

Activation of the contact sensor is initiated by the operator. After activation an **Activation Message** is sent. The full sequence is depicted below.

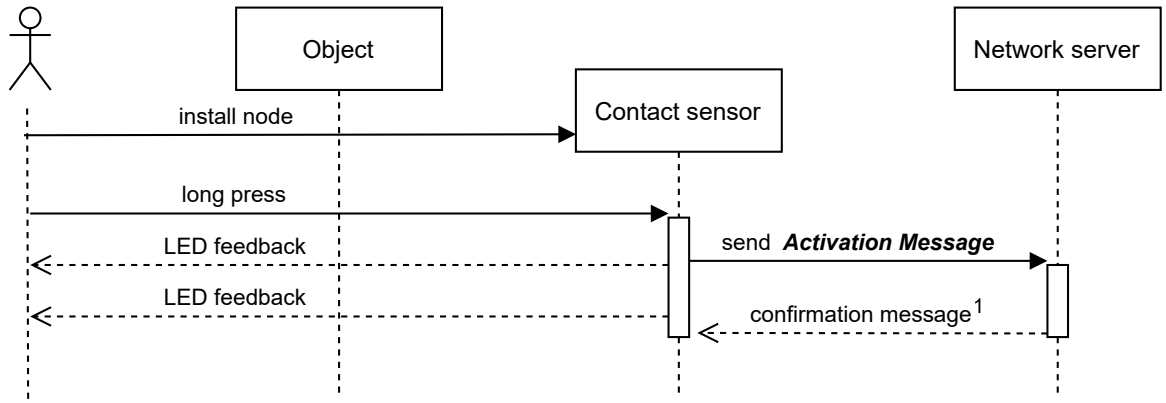

Figure 1: Activation Sequence Scheme

In case of an error during activation, the procedure is aborted. The error situations can be as followed (see numbers in diagram):

1. No LoRa confirmation message from network server is received: no LoRa coverage.

Note that press and release of the magnet key can trigger transition to close and open states, respectively, and application event messages (see 2.2), subsequently.

2.2 Application Event (Magnet contact state change)

The contact sensor state is determined by periodically measuring its magnetic field. The contact sensor triggers an ***Application Event Message*** on state transition.

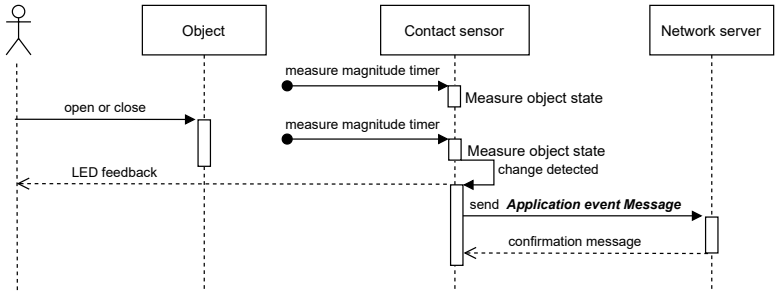

Figure 2: Event Message Scheme

An ***Application Event Message*** is also sent periodically. Timing is independent from the measure timer for the magnet contact state.

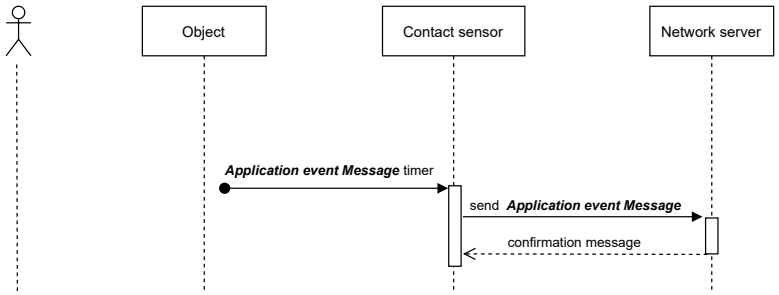

Figure 3: Periodic Event Message Scheme

2.3 Device Status

Periodically a ***Device Status Message*** is sent. Timing is independent from the temperature measure timer and the magnet contact measure timer. The message contains average, min and max values which are reset on send.

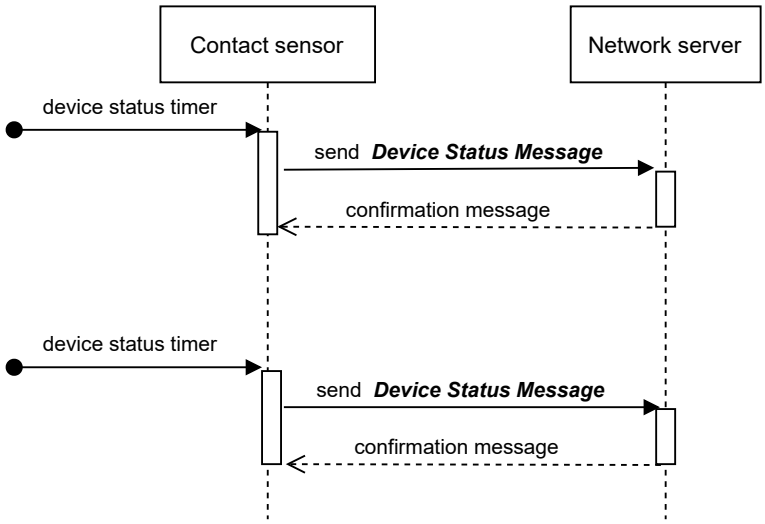

Figure 4: Device Status Scheme (periodic)

A ***Device Status Message*** can also be triggered by the operator by the short press of the magnet key.

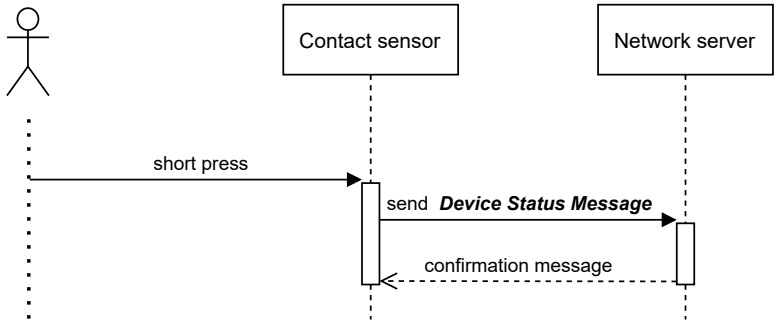

Figure 5: Device Status Scheme (short button press)

Note that press and release of the magnet key can trigger transition to close and open states, respectively, and application event messages (see 2.2), subsequently.

2.4 Deactivation Event

The operator can deactivate the device by long press of the magnet key. This will result in ***Deactivation Message***.

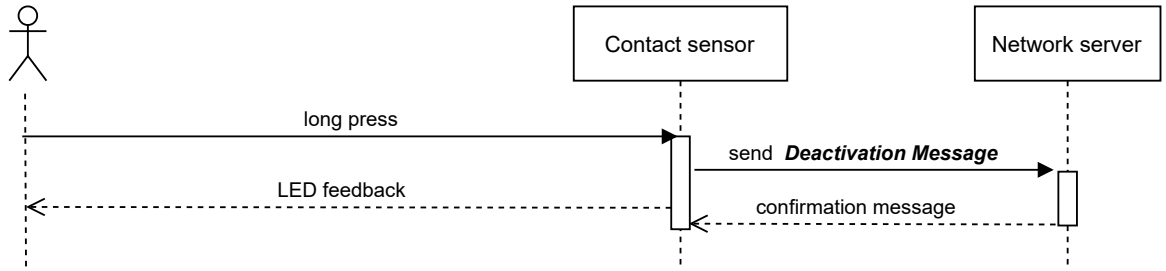

Figure 6: Deactivation Sequence Scheme

Note that press of the magnet key can trigger transition to close state, and an application event message, subsequently.

3 Message Overview

3.1 Message Overview

ID	Name	Up/Down	Purpose
0	Boot	up	Informs on device reboot
1	Activated	up	Indicates that a device is activated by the magnet key and operational.
2	Deactivated	up	Indicates that a device is deactivated by the magnet key.
3	Application event	up	Inform on the state transition. It is sent on state transitions (open-close, close-open).
4	Device status	up	Informs on device health, battery info, counter for statistics etc. It is sent periodically.
5	Device configuration	down	Configures the device with radio setting.
6	Application configuration	down	Configure the application specific settings.

Table 1: Message Overview

4 Decoding/Encoding

The messages are sent over LoRa in a binary bytestring. A LoRa network server can decode raw bytestrings coming from the LoRa devices into JSON objects. It can also encode JSON objects back into bytestrings that form the payload of downlink messages.

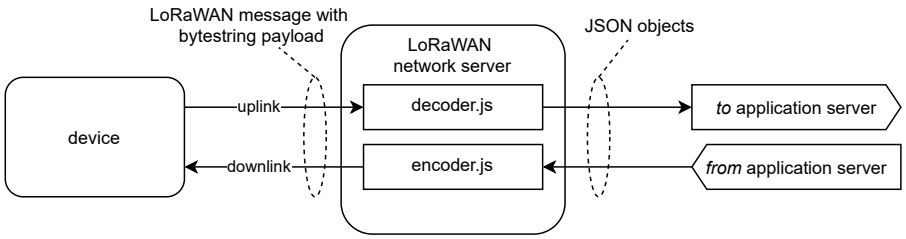

Figure 7: Decoding / Encoding data flow

Below each message is described and how it is encoded in the bytestring. The decoder and encoder according to this document are available on the TWTG Github. If using this Decoder / Encoder the bytestring information below can be ignored.

5 Message Header

The header is at the head of all messages. It indicates the protocol version and message type.

5.1 JSON Structure

```
"header": {
    "message_type": <json string>,
    "protocol_version": <json number>
}
```

Listing 1: JSON Header Structure

5.2 Binary encoding and description

Description	Binary encoding	Byte index
`message_type` • boot(0) • activated(1) • deactivated(2) • application_event(3) • device_status(4) • device_configuration(5) • application_configuration(6)	uint8 bits[3..0] message_id	0
`protocol_version` Version of the protocol, range: For this protocol version the value is 1	bits[7..4] protocol version nr.	

Table 2: Header Binary Encoding

6 Uplink Messages

6.1 Boot

The boot message is sent when the device is used for the first time or when a reboot occurs. Reboots might occur intended in case of activating a newly received configuration. Or the device might reboot due to a system error as a matter to recovery (e.g. continues communication failures, unforeseen situations etc). The boot message contains information why it has (re)booted. Typically this information can be ignored and is only used for solving problems in the field. During normal operation and with sufficient network quality reboots seldomly occur, other than activating a newly received configuration.

6.1.1 JSON Structure

```
"boot": {
    "device_type": <json string>,
    "version_hash": <json string>,
    "device_config_crc": <json string>,
    "application_config_crc": <json string>,
    "reset_flags": <json number>,
    "reboot_counter": <json number>,
    "reboot_info": <json string>,
    "last_device_state": <json number>,
    "bist": <json number>
}
```

Listing 2: Boot JSON Structure

6.1.2 Binary encoding and description

Description	Binary encoding	Byte index
`message_header` As described in Message Header.	uint8	0
`device_type` The device type as known in the NEON family. For this device the value is: • "vs-cs" (7)	uint8	1
`version_hash` Version hash of the currently running firmware, represented as a hexadecimal string. Range: `0x00000000 .. 0xFFFFFFFF`	uint32	2..5
`device_config_crc` CRC of the currently loaded protocol configuration, represented as a hexadecimal string. This can be used to verify if the corrected configuration is loaded. Range: `0x0000 .. 0xFFFF`	uint16	6..7

Description	Binary encoding	Byte index
`application_config_crc` CRC of the currently loaded application configuration, represented as a hexadecimal string. This can be used to verify if the corrected configuration is loaded. Range: `0x0000 .. 0xFFFF`	uint16	8..9
`reset_flags` A bitmask field from the microcontroller that indicated the physical reason for the reset that caused the reboot. This can be used for analysis when a device is not working properly. bit[0] - Option Byte Loader bit[1] - Pin bit[2] - Power On bit[3] - Software bit[4] - IWDG bit[5] - WWDG bit[6] - Low Power	uint8	10

`reboot_counter`
Counter of the number of reboots. Each time a reboot occurs the counter is increased. The counter is 8bit and will wrap after 255 to 0. This can be used for detecting abnormal rebooting behavior.
 	
Range:
`0 .. 255`
	uint8	11

`reboot_info` Informational string with: information on the reboot. Example values: • "swdog (ABCD)" • "assert (test:1234)" • "application (0xaabbccdd)" • "system (0xaabbccdd)"	uint8[9] byte[0] reboot type byte[1..8] specific payload	12..20
`last_device_state` Last state the device was in before reboot. This can be used for analysis when a device is not working properly.	uint8	21

Description	Binary encoding	Byte index
`bist` A bitmask with the result of the build in self test. At boot the device performs a self test in order to verify the working of essential components. This can be used for analysis when a device is not working properly. Bit value: 0: Test Failed 1: Test Succeeded bit[0] - Reserved (Always 1) bit[1] - Reserved (Always 0) bit[2] - Battery Measurement bit[3] - Reserved (Always 0) bit[4] - Sensor bit[5] - LoRa Module bit[6] - Provisioning bit[7] - Reserved (Always 0)	uint8	22

Table 3: Boot Message Binary Encoding

6.2 Activated

On activation an activated message is sent. It is an indication that the device is operational.

6.2.1 JSON Structure

The activation message does not have any payload

6.2.2 Binary encoding and description.

Description	Binary encoding	Byte index
`message_header` As described in Message Header.	uint8	0

Table 4: Activated Message Binary Encoding

6.3 Deactivated

When the device is deactivated by the magnet key a deactivation message is sent. It is an indication that the device is not operational.

6.3.1 JSON Structure

The deactivation message does not have any payload.

6.3.2 Binary encoding and description

Description	Binary encoding	Byte index
`message_header` As described in Message Header.	uint8	0

Table 5: Deactivated Message Binary Encoding

6.4 Application Event

The application event message contains information on the magnet contact state. It is triggered by state transitions (open-close, close-open) or periodic.

6.4.1 JSON Structure

```
"application_event": {
    "state": <json string>,
    "state_transition_sequence": <json number>,
    "debug": {
        "temperature": <json number>,
        "magnet_magnitude": <json number>
    }
}
```

Listing 3: Application JSON Structure

6.4.2 Binary encoding and description

Description	Binary encoding	Byte index
`message_header` As described in Message Header.	uint8	0
`state` String with the possible values: "open" (0) "closed" (1)	uint8 bit[0]	1
`state_transition_sequence` Sequence number of magnet contact state transitions. Every time a magnet contact state transition is detected the number will be increased. This can be used to detect missed valve state transitions. Range: 0 .. 255 After 255 it will wrap to 1. Zero value is reserved for after boot only.	uint8	2

Table 6: Application Event Binary Encoding

6.4.3 Binary encoding and description debug

This information is added to the message when bit[1] of "Switch bitmask" in the device configuration is set.

Description	Binary encoding	Byte index
`debug_temperature` Temperature of PCB at moment of the event units of 1 ℃. Range: -40 ℃ .. 80 ℃	int8 1 ℃ per LSB	3
`magnet_magnitude` Magnet magnitude value at the moment of event. Range: 0 milligauss .. 56756 milligauss	uint32	4..7

Table 7: Application Debug Binary Encoding

6.5 Device Status

This message contains information for maintenance, device health and other analysis. It is sent periodically and on short press of the magnet key.

6.5.1 JSON Structure

```
"device_status": {
     "device_config_crc": <json string>,
     "application_config_crc": <json string>,
     "event_counter": <json number>,
     "battery_voltage": {
        "low": <json number>,
        "high": <json number>,
        "settle": <json number>
    },
    "temperature": {
        "min": <json number>,
        "max": <json number>,
        "avg": <json number>
    },
    "tx_counter": <json number>,
    "avg_rssi": <json number>,
    "avg_snr": <json number>,
}
```

Listing 4: Device Status JSON Structure

6.5.2 Binary encoding and description

Description	Binary encoding	Byte index
`message_header` As described in Message Header.	uint8	0

`device_config_crc`
CRC of the currently loaded protocol configuration, represented as a hexadecimal string. This can be used to verify if the corrected configuration is loaded.
 	
Range:
`0x0000 .. 0xFFFF`
	uint16	1..2

`application_config_crc`
CRC of the currently loaded application configuration, represented as a hexadecimal string. This can be used to verify if the corrected configuration is loaded.
 	
Range:
`0x0000 .. 0xFFFF`
	uint16	3..4

Description	Binary encoding	Byte index
`event_counter` Counter for number of events. Every time an event message is sent this counter is increased. This can be used to detect missed event messages. After 255 it will wrap to 1. Zero value is reserved for after boot only. Range: 0 .. 255	uint8	5
`battery_voltage` Voltage measurement in Volt, meant as input for battery charge estimation in the backend. Note that a single voltage level on itself is not suitable for battery charge determination because of the type of battery. To feed the model on the backend with accurate data the voltage is measured on different moments. During low load, high load and after a settle time after high load. Range: 0.000V .. 4.000V	uint16[3] 0.001V per LSB	6..11
`temperature` PCB temperature in units of 1 °C. It is reported in the **min**, **max** and **avg** temperature since the last device status message. Range: -40 °C .. 80 °C	uint8[3] 1 °C per LSB	12..14
`tx_counter` Number of LoRa transmissions since the last device status message. It is reset after the device status message is sent. If the number of LoRa transmissions becomes bigger than 255 this field is set to 255.	uint8	15
`avr_rssi` The average RSSI of received messages since the last device status message. The value is in units of 1 dBm. Range: -255 dBm .. 0 dBm	uint8 -1dBm per LSB	16
`avr_snr` The average SNR of received messages since the last device status message. Range: –20 dB .. 127 dB	int8	17

Table 8: Device Status Binary Encoding

7 Downlink Messages

7.1 Device Configuration

In the device configuration the non application related behavior can be configured. Changing these parameters will have an effect on battery life and quality of service.

7.1.1 JSON Structure

```
"device_config" = {
    "switch_mask": {
       "enable_confirmed_changed_message": <json bool>,
       "enable_debug_data": <json bool>
    },
    "communication_max_retries": <json number>,
    "number_of_unconfirmed_messages": <json number>,
    "periodic_message_random_delay_seconds": <json number>,
    "status_message_interval_seconds": <json number>,
    "status_message_confirmed_interval": <json number>,
    "lora_failure_holdoff_count": <json number>,
    "lora_system_recover_count": <json number>,
    "lorawan_fsb_mask": [
        <json string>,
        <json string>,
        <json string>,
        <json string>,
        <json string>
    ]
}
```

Listing 5: Device Config JSON Structure

7.1.2 Binary encoding and description

Description	Default value	Binary encoding	Byte index
`message_header` As described in Message Header.		uint8	0

`switch_mask`
booleans to turn features on or off.
 	
`enable_confirmed_changed_message (0)`
Enable confirmed *application event* message.
 	
`enable_debug_data (1)`
Enale extra debug data on the event message
	
	true
	
	false
	
	Binary (hex):
	0x01
		uint8
		bitmask
		
		bit[0]
		
		bit[1]
		(other bits are unused)
			1

`communication_max_retries` The maximum number of retries on failing confirmed messages. Range: 1 .. 10	3 Binary (hex): 0x03	uint8	2

Description	Default value	Binary encoding	Byte index
`number_of_unconfirmed_messages` The number of repeating unconfirmed messages. Range: 1 .. 5	2 Binary (hex): 0x02	uint8	3
`periodic_message_random_delay_seconds` To avoid clustering and collisions of uplink transmissions of multiple devices a random delay is added to periodic messages (device status message and timer triggered event message). Range: 0 .. 255 seconds	60 Binary (hex): 0x3C	uint8	4
`status_message_interval_seconds` Interval in seconds at which periodic device status messages are sent. Range: 60 .. 604800 seconds (= 7 days)	86400 (once per day) Binary (hex): 0xA0 0x05	uint16 60 seconds per LSB	5..6
`status_message_confirmed_interval` Confirm every n messages, the messages in between are sent unconfirmed. Default is 1, such that all periodic messages are confirmed. The number can be increased to require less downlinks (for reasons of gateway RF duty cycle or network server costs), but will degrade the quality of service. Range: 0 .. 255	1 Binary (hex): 0x01	uint8	7
`lora_failure_holdoff_count` In case of persistent network problems (not receiving acknowledgements on confirmed messages) the device tries to recover by a device reboot. This parameter configures the number of consecutive failed confirmed messages needed before it reboots. Range: 0 .. 5	1 Binary (hex): 0x02	uint8	8
`lora_system_recover_count` The number of attempts the LoRa handler is trying to recover from a system failure (not responsive radio). Range: 0 .. 5 DISCLAIMER: This value should not be changed for normal use.	1 Binary (hex): 0x01	uint8	9

Description	Default value	Binary encoding	Byte index
`lorawan_fsb_mask` Frequency sub-band (FSB) mask for upstream. [Only for US915] DISCLAIMER: The device is only tested and certified for the FCC 125KHz Hybrid transmission mode (1st 8 channels), therefore this channel configuration should not be changed for operational conditions	US915 Hybrid: {"0x00FF", "0x0000", "0x0000", "0x0000", "0x0000"} Binary (hex): 0xFF 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 EU868: {"0x0000", "0x0000", "0x0000", "0x0000", "0x0000"} Binary (hex): 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00	uint16[5]	10..19
`device_config_crc` CRC of the configuration values above. It is calculated by the Encoder.	See TWTG Github for examples.	uint16	20..21

Table 9: Device Configuration Binary Encoding

7.2 Application Configuration

7.2.1 JSON Structure

```
"application_config" = {
    "device_type": <json string>,
    "magnet_measurement_interval_seconds": <json number>,
    "magnitude_threshold": <json number>,
    "magnitude_hysteresis": <json number>,
    "periodic_event_message_interval_seconds": <json number>
}
```

Listing 6: Application Config JSON Structure

7.2.2 Binary encoding and description

Description	Default value	Binary encoding	Byte index
`message_header` As described in Message Header.		uint8	0
`device_type` The device type as known in the NEON family. For this device, Contact Sensor, the value is: • "vs-cs" (7)	"vs-cs" Binary (hex): 0x07	uint8 "vs-cs" = 7	1
`magnet_measurement_interval_seconds` Interval in seconds, at which the magnetometer is read. Changing this value has an effect on responsiveness and battery life. Range: 1 .. 255 seconds	2 Binary (hex): 0x02	uint8	2
`magnitude_threshold` magnitude threshold for closed to open detection. Range: 0 .. 31875 milligauss	5000 Binary (hex): 0x28	uint8 125 milligauss per LSB	3
`magnitude_hysteresis` Hysteresis window on the threshold for open to closed detection. Range: 0 .. `magnitude_threshold` milligauss	1000 Binary (hex): 0x08	uint8 125 milligauss per LSB	4
`periodic_event_message_interval_seconds` Interval in seconds at which the application event messages are sent additionally to the other triggers. Range: 60 .. 604800 seconds(=7 days)	86400 (once per day) Binary (hex): 0xA0 0x05	uint16 60 seconds per LSB	5..6
`application_config_crc` CRC of the configuration values above. It is calculated by the Encoder.	See TWTG Github for examples.	uint16	7..8

Table 10: Application Configuration Binary Encoding

Revision History

Revision	Date	Author(s)	Description
A	23-11-2021	NY	Initial version of the Communication document for the NEON Contact Sensor.