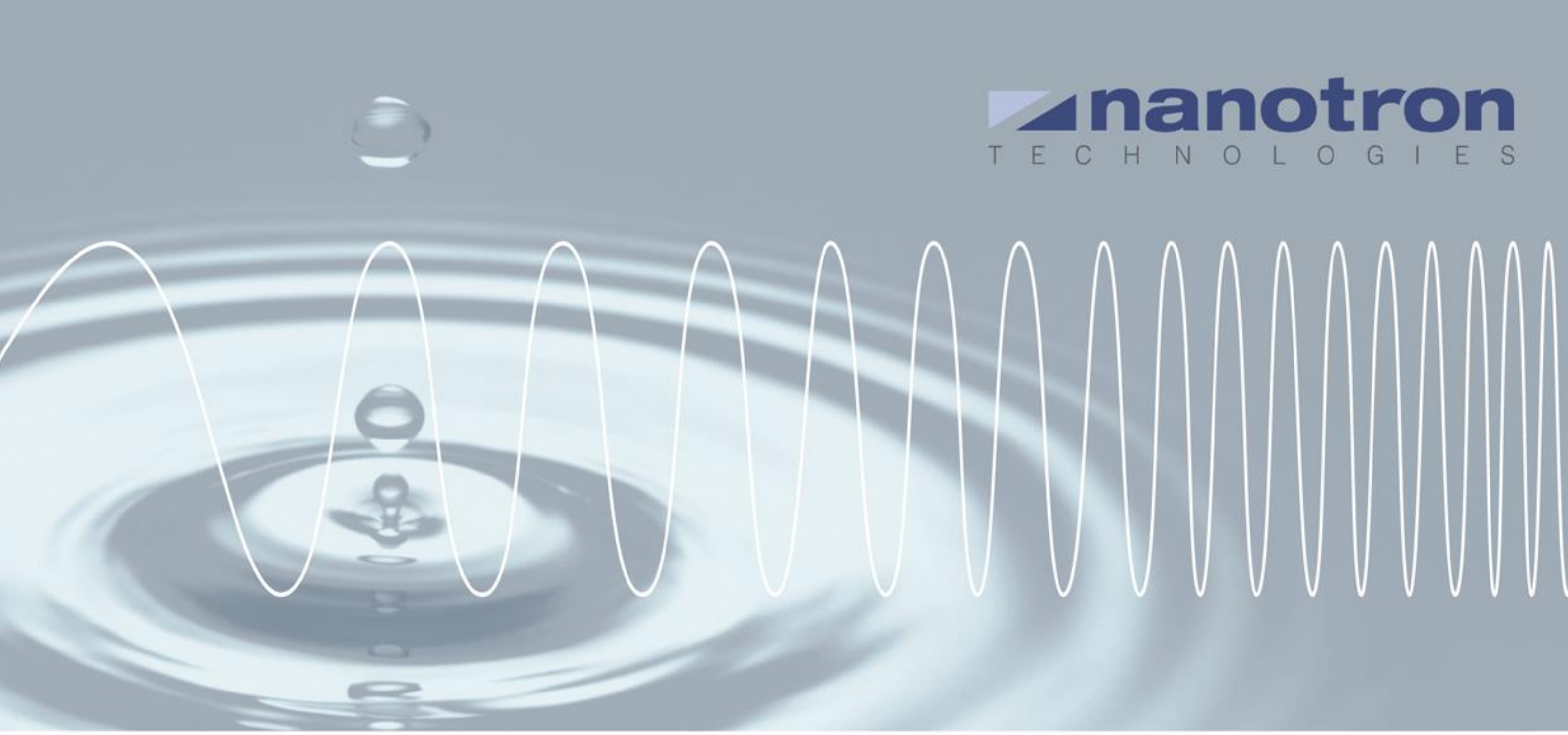

AN0506 - Understanding the swarm Bee LE Payload

1.1

NA-15-0356-0022-1.1

Document Information

Document Title:	AN0506 - Understanding the swarm Bee LE Payload
Document Version:	1.1
Current Date:	2015-09-18
Print Date:	2015-09-18
Document ID:	NA-15-0356-0022-1.1
Document Author:	MLA

Contents

1. Introduction

The primary function of a swarm bee is to announce its presence to other devices in the area and to estimate its distance to other swarm bee modules. In addition to this, they can also transmit sensors data and other information relevant for the user, such as battery level, status of the GPIOs, sensors data and even short messages customized by the user. All this data is added in the packet's payload, and it is decoded when using the swarm PC tool. But, how can this be read with other tools or integrated in the specific users' application?
This document lists the different tools that can be used to read the payload of a packet coming from a swarm bee device.

2. Payload information

2.1. Content of the payload

Before explaining how to read the payload, it may be convenient to see what the content of the payload itself is:

- Device Class: indicates the device class the node
  Format ASCII
  Format BINARY
  Example ASCII interface: 7 Binary interface: 0x0007 → device class 7

- MEMS values consists of x, y, z acceleration with a range from -32768 mg to +32767 mg. The three of them follow the same format.
  Format ASCII: 6 bytes (dec), first byte "+" or "-"
  Format BINARY: 2 bytes (int16_t)
  Example ASCII interface: +296, +335, +968 Binary interface: 0x03070027000604
  → Acceleration of 296 mg in x direction, 335 mg in y direction and 968 in z direction.
  This is only transmitted when the sensors are enabled.

- RSSI: strength with which the signal of the remote node was received, in dBm. The range is from -35dBm to -128 dBm. The value -128 indicates that there was an error in the strength estimation. The value -35 indicates the signal strength is -35 dBm or higher.
  Format ASCII: 4 byte (dec)
  Format BINARY: 1 byte (int8_t)
  Example ASCII interface: -56 Binary interface: 0xc8 → RSSI = -56 dBm

- Temperature in Celsius degrees. Its range is from -99°C to 99°C.
  Format ASCII: 3 byte (dec), first byte "+" or "-"
  Format BINARY: 1 byte (int8_t)
  Example ASCII interface: 23 Binary interface: 0xc8 → Temperature 23°C
  This is only transmitted when the sensors are enabled.

- Power mode: 0 = receiver of remote node is always active
  1 = remote node is in low power mode
  2 = remote node is in autonomous mode
  Format ASCII: 1 byte (dec)
  Format BINARY: 1 byte (uint8_t)
  Example ASCII interface: 1 Binary interface: 0x01 → Node in low power mode

- Battery level: indicates the battery voltage level. In order to have a correct value a voltage divider should be place between Vin and the pin ADC_IN. The value has been calibrated for a voltage divider similar to the one stated in the data sheet (with resistors values of 2.7 MΩ and 2.1 MΩ). Its range is form 0 to 255.
  Format ASCII: 3 bytes (dec)
  Format BINARY: 1 byte (uint8_t)
  Example ASCII interface: 25 Binary interface: 0x20 → Voltage level 2.5 V

- GPIO Status: The range is from 00 to FF and the correspondence bit-pin is:
  Bit 0 = PA0 = DIO_0
  Bit 1 = PA2 = DIO_1
  Bit 2 = PA3 = DIO_2

Bit 3 = PA8 = DIO_3
Format ASCII: 2 bytes (hex)
Format BINARY: 1 byte (uint8_t)
Example ASCII interface: 05 Binary interface: 0x05 → Binary: 0101, pins 2 and 0 are set

- Wakeup reason: it indicates whether the message was sent due to an interrupt of the GPIOs or the MEMS. The range is from 00 to FF and the correspondence bit-reason:
  Bit 0 = DIO_0
  Bit 1 = DIO_1
  Bit 2 = DIO_2
  Bit 3 = DIO_3
  Bit 4 = MEMS
  A value 0 indicates that neither the GPIO nor the MEMS caused the interrupt, the blink was triggered by the timer.
  Format ASCII 2 bytes (hex)
  Format BINARY: 1 byte (uint8_t)
  Example ASCII interface: 8 Binary interface: 0x08 → Mask 00001000: interrupt triggered by DIO_3

- BlinkID: sequence number generated by the device before every transmission. The range is from 0 to 255; after 255 goes to 0 again.
  Format ASCII: 3 bytes (dec)
  Format BINARY: 1 byte (uint8_t)
  Example ASCII interface: 187 Binary interface: 0xbb → Blink number 187

- RX slot counter: when the device is in low power mode or in autonomous mode, it announces how many transmissions will still happen before the reception window is open. The value 0 indicates the RX window is open just after that transmission.
  Format ASCII: 3 bytes (dec)
  Format BINARY: 1 byte (uint8_t)
  Example ASCII interface: 2 Binary interface: 0x02 → Still another 2 blink before the rx window is open

- Timestamp, in ms, indicates in what instant the message was sent. It starts to count when the swarm is powered on and it is restarted if the swarm bee is restarted. The range is from 0 to 4294967295 ms. When 4294967295 is reached it goes again to 0.
  Format ASCII: 8 bytes (dec)
  Format BINARY: 4 bytes (uint32_t)
  Example ASCII interface: 4234877 Binary interface: 0x00409e7d → 4234877 ms since the device went on.

- User data: this is information included by the user and can be such that the total payload length is 128 bytes.

All the field of the payload are included in the packet in the same order in which they have been shown.
[2] includes more information about the sensor data.

2.2. What packets contain payload?

Not all packets sent by a swarm device include payload. Some of them do, others include only part of it and some have no payload at all. The following list includes those packets that may have some payload:

- Node ID broadcast (blink) can contain all the data mentioned in previous subsection. Note that the RSSI value is not included.
- Ranging results broadcast message can have all data except for the RX slot counter and user data. They include also the RSSI field to indicate the signal strength of the blink that triggered the ranging operation.
- Data messages, both broadcast and unicast, include only user data.
- Ranging request is the packet sent to a device to request that they carry on a ranging operation. This packet can include user data

3. How to see the swarm packets over the air

3.1. Using another swarm bee device

When a packet containing payload is received, the swarm device passes to its host in the shape of a notification.

DataNotification (DNO):

This notification only informs of the arrival of user data, independently of what kind of packet includes thet user data, it could be a data packet, a ranging packet, a node ID notification… When it receives the DNO indicating that there is information from a certain source, the host can requesting it to the swarm bee using the API command GDAT.

NodeIdNotification (NIN) or RangeResultNotification (RRN):

The payload (excluding user data) contained in the node ID broadcast and the range result broadcast is passed to the host together with their respective notifications. The user can configure the notification message so that the swarm only includes the relevant information in the notification. For this the API command NCFG followed by its mask. The mask indicate the fields present or not in the notification and if present in what order.
Note that when a packet also includes user data the swarm bee will generate 2 notifications: a NIN and a DNO.
More information about the API commands can be found in [1].
The following example show a RRN received at the host in ASCII mode. Before its reception the notifications were configured to show all the contained information.

```
*RRN:000000000003,000011293CD1,0,000097,07FF,1,+7,-31,+1015,-71,+27,0,35,0F,00,107,1,58381
```

As explained in [1] the first fields of the RRN are source, destination, error and range. The value immediately after is the mask of the NCFG; the user can read it to know what values of eth payload will come after. In this case it is `07FF`, indicating that all the field are present. They will appear in the same order as in section 2.1. The payload starts: `1,+7,-31,+1015,-71,+27,1,35,0F,00,107,1,58381`

- Device Class :1
- MEMS: x = -7 mg, y = -31 mg, z = +1015 mg
- RSSI: -71 dBm
- Temperature: 27°C
- Power mode: 1 → the device is in low power mode
- Battery: 35 dV
- GPIO Status: 0x0F → binary: 0000 1111
- Wakeup reason: 00 → indicates that the timer caused the interrupt
- BlinkID: 107
- RX slot counter: 1 the rx window will be open after next blink
- Timestamp: 58,381ms

When the RRN format is binary, the order in which the data is received is the same but we need to read them in hexadecimal:

```
7f286162000000000003000011293cd1000000005907ff01000f0017040eb91a01220f006f010000ebe4185f
```

As explained in [1] the first byte, `0x7f`, is a synchronization byte always present at the beginning of any packet. It is important when reading the messages at the serial port.
The next byte indicates the length of the message excluding the 2-byte CRC at the end of the message: `0x28` → 40 bytes. We can then read the next 40 bytes:

```
6162000000000003000011293cd1000000005907ff01000f0017040eb91a01220f006f010000ebe4
```

The first byte, `0x61`, indicates the message type and the second one, `0x62`, that it is a RRN.

```
000000000003000011293cd1000000005907ff01000f0017040eb91a011c0f006f010000ebe4
```

As in the ASCII form we receive first the source, `0x000000000003`, then the destination, `0x000011293cd1`, the error code, `0x00`, and the range `0x00000059` → 89 cm.
This is followed by the mask indicating what fields are present. This is again: 0x07ff and the rest of the payload: `01 000f 0017 040e b9 1a 01 22 0f 00 6f 01 0000ebe4`

- Device Class :`0x01` → 1
- MEMS: x = `0x000f` → +15 mg, y = `0x0017` → +23 mg, z = `0x040e` → +10385 mg
- RSSI: `0xb9` → -71 dBm
- Temperature: `0x1a` → 26°C
- Power mode: `0x01` → the device is in low power mode

- Battery: `0x22` → 33 dV
- GPIO Status: `0x0F` → binary: 0000 1111
- Wakeup reason: `0x00` → indicates the it was not caused by any interrupt,; it was the timer
- BlinkID: `0x6f` → 111
- RX slot counter: `0x00` → 0 (as the device is always on, this is not relevant)
- Timestamp: `0000ebe4` → 60,388 ms

3.2. Using the RTLS: nanoLES

The swarm bee devices can also be used as tags in the RTL System. NanoLES will received the NodeIDNotification messages from the swarm devices and will use it to estimate position of each of them. The payload can be read at two different server ports: the output port, were it will come together with the position data and at the TCP port.

3.2.1. nanoLES 2

The result is offered by the application interface at the TCP port 3456.This interface delivers the location data to all client applications connected to the port.
The information is organized in one line per blink; and in every line the different fields are separated by ',' and follows the structure:
The last field of the line corresponds to the payload.

```
nanoPAL,TP,<source_address>,<error_code>,<x>,<y>,<z>,<battery>,<time_stamp>,<blink_ID>,
<quality_indicator>,<payload>,<position_valid>
```

Note that in this case the blink_ID and the timestamp are given outside the payload. Moreover, the time stamp follows a different format, and indicates when the blink arrived at the server.
For the swarm device, <battery> is equal to 'inf' (infinity) as its value appears inside the payload. The field <time_stamp> refers to the time when the blink was detected by nanoLES, to timestamp generated by the swarm device itself, is included in the payload.
The first byte of the payload indicates what kind of message the packet is:

0x60 node ID broadcast
0x61 range result broadcast
0x62 data broadcast

The rest of the payload can be explained with some examples:

<u>Example 1 – node ID broadcast</u>

We are going to analyze the output of TCP port 3456 when a node ID broadcast packet is received.

```
nanoPAL,TP,00000004,00, 25.08,-18.74,0.00,inf,2015-09-
04T17:11:01.566,184,0,602001000023150124000407021f000307002700060405b86f01000048656c6c6f
20776f726c642121,1
```

source_address: `00000004`
error_code: `00`
x,y,z: `25.08,-18.74,0.00` (in meters)
`inf` (formerly this was indicating the value of the battery level)
time_stamp `2015-09-04T17:11:01.348`
blink_ID `184`
quality_indicator: `0`
payload:

```
602001000023|150124000407021f000307002700060405b86f01000048656c6c6f20776f726c642121
   header   |         payload
```

position_valid: `1`
The payload consist of a fix part at the beginning followed by a variable part. The variable part is the sensors data and the user data that may be or not present. Let's start with the fix part:
`0x60` → indicates the packet was a node ID broadcast from a swarm bee
`0x20` → protocol version
`0x01` → device class: 1
`0x00` → swarm bee power mode: always active
`0x00` → wake-up reason: because of the timer, no interrupt
`0x23` → length of the variable part of the payload: 35 bytes

To continue reading the payload we need to take the 35 bytes of variable payload:

```
150124000407021f000307002700060405b86f010000|48656c6c6f20776f726c642121
                sensors data                |   user data
```

`0x15` → length of the sensors data: 21 bytes. This means that the data after these 21 bytes is user data.

Sensors data: `0124000407021f0003070027000604005b86f010000`
The rest is user data: `48656c6c6f20776f726c642121`
As already mentioned, not all swarm devices may have all sensors enabled. For this reason, the data is presented always with a sensor_type indicator followed by the corresponding sensor value. The indicator `0x00` means that no more sensors data is available. Table 3-1 indicates the different sensors with its corresponding sensor_type, and the length of the sensors value that follows:

Table 3-1 Sensor indicators and their data length

Sensor	Sensor_type	Value_length
Battery indicator	0x01	2 bytes
Temperature	0x02	2 bytes
MEMS	0x03	6 bytes (2 per axis)
GPIO	0x04	1 byte
Timestamp	0x05	4 byte

The way to read the sensor data would be: `012400 0407 021f00 03070027000604 05b86f010000`
`01` → sensor indicator: battery value, 2 bytes of data
`1c`
`00` → 0x0024 = 36 dV
`04` → sensor indicator: GPIO, 1 bytes of data
`07` → 0x07 → GPIO mask 0000 0111
`02` → sensor indicator: temperature, 2 bytes of data
`1f`
`00` → 0x001f = 31°C
`03` → sensor indicator: MEMS, 2 bytes of data per axis
`07`
`00` → acceleration axis x (2's complement): 0x0007 = 7 mg
`27`
`00` → acceleration axis y (2's complement): 0x0027 = 39 mg
`06`
`04` → acceleration axis y (2's complement): 0x0406 = 1030 mg
`05` → sensor indicator: timestamp, 4 bytes of data
`b8`
`6f`
`01`
`00` → 0x00016fb8 = 94,136 ms
`00` → sensor indicator: end of sensor data

<u>Example 2 – range result broadcast</u>

In this case we analyze the output of TCP port 3456 when the received packet is a range result broadcast.

```
nanoPAL,TP,00000004,00, 25.08,-18.74,0.00,inf,2015-09-04T17:11:01.566,184,0,
61200100002715011f00040702200003f1ff0000f70305f46e0100000000000000004000000000000300000000cf,1
```

The same analysis is repeated for the payload:

```
612001000027|15011f00040702200003f1ff0000f70305f46e0100|0000000000004000000000000300000000cf
   Header    |         payload (sensors data)           |      payload (range result)
```

`0x61` → indicates the packet was a range result broadcast from a swarm bee
`0x20` → protocol version
`0x01` → device class: 1
`0x00` → swarm bee power mode: always active
`0x00` → wake-up reason: because of the timer, no interrupt
`0x27` → length of the variable part of the payload: 39 bytes
`0x15` → length of the sensor data

From this we can split the sensor data and the rest. The sensor data should be analyzed as in the previous example. And, what about the 'other data'? From section 2.1 we know that no user data can be available in the range result broadcast packet. What kind of information may be the 'other data'? Just the ranging information that is always present in this kind of packets. Let's analyze it.

Sensors data: `011f00040702200003f1ff0000f70305f46e0100`

Other data: `000000000000004|000000000003|00|000000cf`

[1] states the information available in a range result broadcast:
source_address: 6 bytes
destination_address: 6 bytes
error_code: 1 byte
range or distance: 4 bytes, it is given in cm

What we have then is:

`0x000000000004` → source address
`0x000000000003` → destination address
`0x00` → error code
`0x000000cf` → distance: 206 cm

3.2.2. nanoLES 3

NanoLES3 delivers the result at the result interface on TCP port 3458.For every blink the interface delivers the location data plus payload to all client applications connected to the port.
The information is passed using Google Protocol Buffers. The application note AN0601 [5] explains how to access it.

3.3. Using the Sniffer

The packets exchanged among devices and their content can be monitored with the swarm Sniffer. For each packet detected, the Sniffer is able of identifying what kind of packet it is and decodes it. The payload is, thus, showed in its binary format (similar to how it was shown in previous section) and also in ASCII format, already decoded so that the user can easily find the data that he needs.
Figure 3-1 shows a data packet detected by the Sniffer. This kind of packets only have user data as payload, thus the way all the information is presented is much simpler. However, it is still given in binary and ASCII formats.

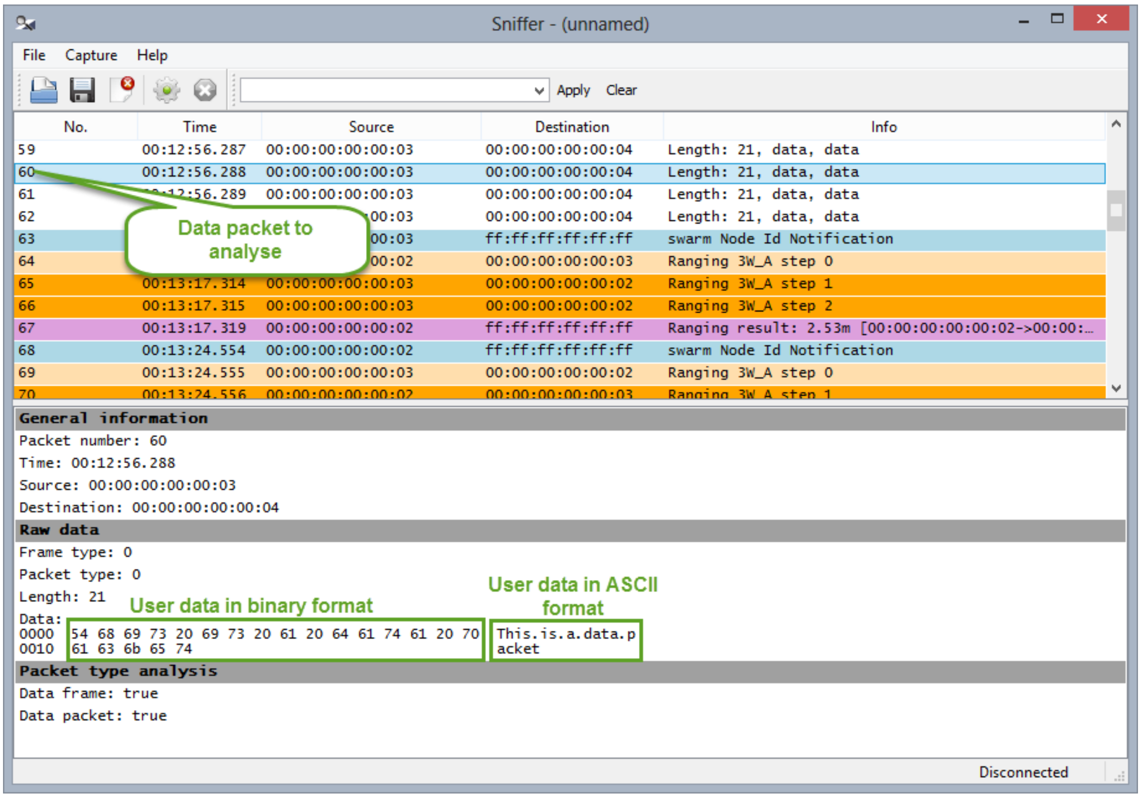

Figure 3-1 Data packet detected by the Sniffer

Figure 3-2 shows an example of how the sniffer interprets the payload included in a node ID broadcast packet. In the upper left part the user can read the packet identification, ‘swarm Node Id Notification’ in this case. In the middle part the payload is given in binary format. The same data is transmitted and interpreted by the sniffer, which organizes it in a more user friendly format (lower part).

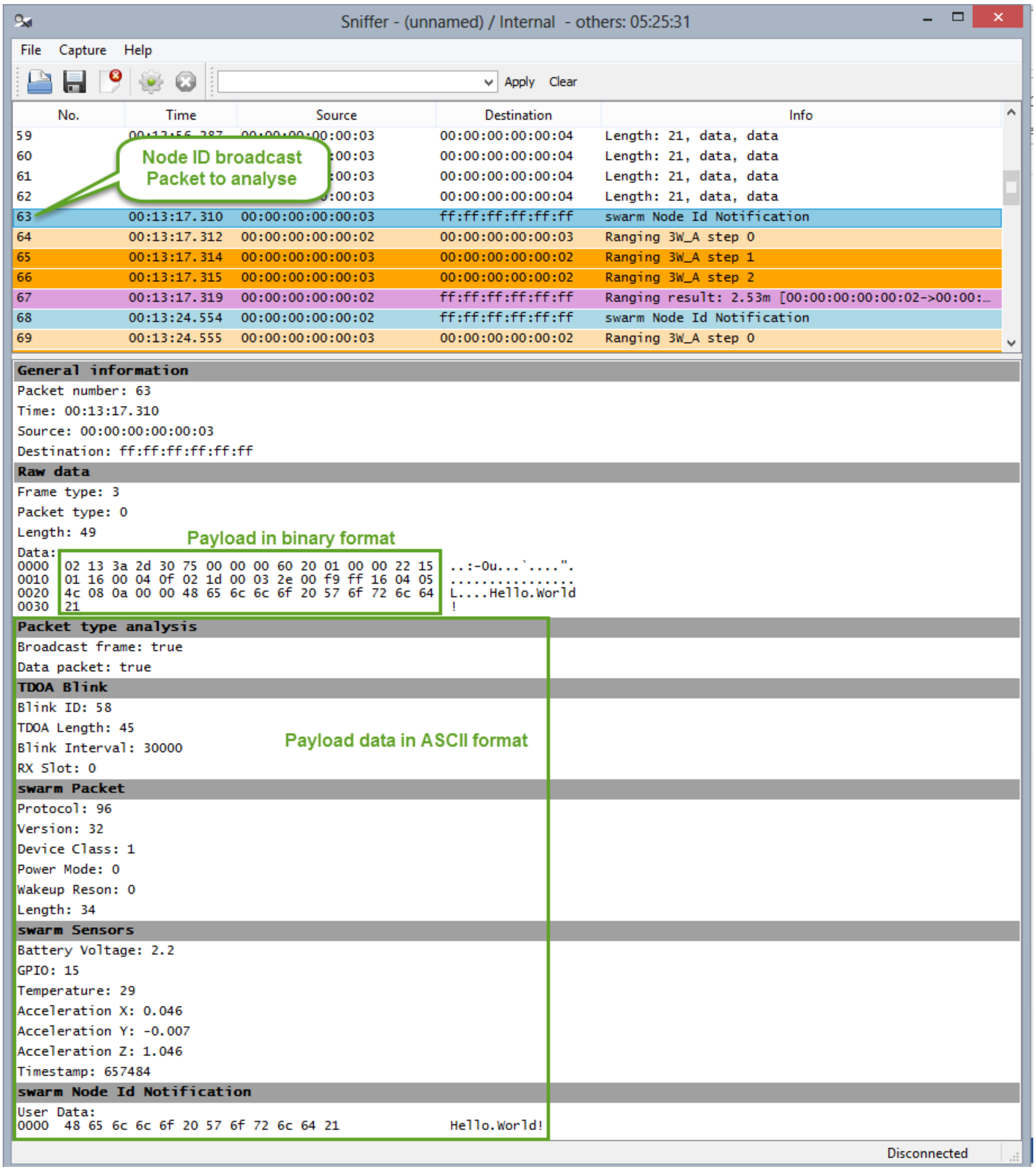

Figure 3-2 Node ID broadacst detected by the Sniffer

For those who are not familiar yet with the swarm bee LE Sniffer tool, we recommend to check the swarm Sniffer GUI User Guide [6].

4. Summary

The messages transmitted by a swarm bees can contain not only ranging information but also other payload, such as sensors data and user data. This payload can be access in multiple ways; each of them with some differences with respect to the others.
This application note shows what kind of information can be containing in the different packets transmitted by a swarm bee. It also lists the different ways to access the payload and, in each case, how this payload can be interpreted and understood.

5. References

[1] swarm API Description_V2.1, nanotron Technologies, 2015
[2] AN0505_Using 3D Acceleration and Temperature Sensor Data from swarm Bee LE_V1.1, nanotron Technologies, 2015
[3] nanoPAL RTLS Toolbox UG, nanotron Technologies, 2014
[4] nanoLES3_UG_1.4, nanotron Technologies, 2014
[5] AN0601 Using the binary interfaces of nanoLES v1.0, nanotron Technologies, 2015
[6] swarm Sniffer GUI UG, nanotron Technologies, 2015

Document History

Date	Author	Version	Description
9/11/2015	MLA	1.0	Initial version
9/18/2015	MLA	1.1	Minor correction

Life Support Policy

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Nanotron Technologies GmbH customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify nanotron Technologies GmbH for any damages resulting from such improper use or sale.

About Nanotron Technologies GmbH

Today nanotron's *embedded location platform* delivers location-awareness for safety and productivity solutions across industrial and consumer markets. The platform consists of chips, modules and software that enable precise real-time positioning and concurrent wireless communication. The ubiquitous proliferation of interoperable location platforms is creating the location-aware Internet of Things.

Further Information

For more information about products from nanotron Technologies GmbH, contact a sales representative at the following address:

nanotron Technologies GmbH
Alt-Moabit 60
10555 Berlin, Germany
Phone: +49 30 399 954 – 0
Fax: +49 30 399 954 – 188
Email: sales@nanotron.com
Internet: www.nanotron.com