

---

# Getting started with X-CUBE-SUBG2, Sub-1 GHz RF software expansion for STM32Cube

## Introduction

X-CUBE-SUBG2 is an expansion software package for STM32Cube. The software runs on the STM32 and includes drivers that recognize the Sub-1 GHz RF communication for S2-LP.

The expansion is built on STM32Cube software technology to ease portability across different STM32 microcontrollers.

The software comes with sample applications of P2P, CLI, FIFO TX/RX, and 6LoWPAN communication protocols, running on an [X-NUCLEO-S2868A2](#) or [X-NUCLEO-S2915A1](#) expansion board when connected to a compatible [STM32 Nucleo](#) development board.

The software is also available on [GitHub](#), where the users can signal bugs and propose new ideas through **[Issues]** and **[Pull requests]** tabs.

---

### Related links

*Visit the [STM32Cube ecosystem web page](#) on [www.st.com](#) for further information*

---

# 1 Acronyms and abbreviations

Table 1. List of acronyms

Acronym	Description
GHz	Giga Hertz
AMR	Automatic meter reading
CLI	Command line interface
IRQ	Interrupt request
RSSI	Received signal strength indication
FIFO	First-in, first-out
EEPROM	Electrically erasable programmable read-only memory
GUI	Graphical user interface
P2P	Point-to-point communication
RF	Radio frequency
MCU	Microcontroller unit
HAL	Hardware abstraction layer
SPI	Serial peripheral interface
USB	Universal serial bus
NVIC	Nested vectored interrupt controller
WSN	Wireless sensors network
BSP	Board support package
LED	Light emitting diode
IPv6	Internet protocol vers. 6
UDP	User datagram protocol
TCP	Transmission control protocol
6LoWPAN	IPv6 over low-power wireless personal area networks
RPL	Routing protocol for low-power and lossy networks
MAC	Medium access control
CSMA	Carrier-sense multiple access
DMA	Direct memory access
USART	Universal synchronous asynchronous receiver transmitter
wM-Bus	Wireless metering bus

## 2 X-CUBE-SUBG2 software expansion for STM32Cube

### 2.1 Overview

X-CUBE-SUBG2 software package expands STM32Cube functionality .

The key features of the package are:

- Firmware package to start developing using S2-LP expansion boards
- Point-to-point communication sample application for simple buffer transmission and acknowledgment implementation
- CLI example to be used with S2-LP DK GUI to configure the S2-LP radio
- Multi-GPIOs usage demonstration in FIFO TX/RX examples
- Contiki-NG based applications for 6LoWPAN connectivity
- Low-power optimizations for the STM32 MCU family
- Easy portability across different MCU families thanks to STM32Cube
- Package compatible with STM32CubeMX; it can be downloaded from and installed directly into STM32CubeMX
- Free user-friendly license terms
- Sample implementation available on the X-NUCLEO-S2868A1, X-NUCLEO-S2868A2 or X-NUCLEO-S2915A1 expansion boards when connected to a NUCLEO-F401RE, NUCLEO-L053R8, or NUCLEO-L152RE development board

Starting from this software, it is possible to develop other applications, such as:

- automatic meter reading
- home and building automation
- industrial monitoring and control
- wireless fire and security alarm systems

The firmware partitioning among the STM32 microcontroller on the STM32 Nucleo development boards and the S2-LP is:

- STM32 MCU
  - P2P and 6LowPAN applications implementation
  - low power mode handling
  - interrupt services
- S2-LP role
  - basic/stack modes
  - header, sync and trailer fields
  - encoding/decoding
  - sync detection
  - RX and TX 128 bytes FIFO buffers
  - IEEE 802.15.4g hardware packet support with whitening, FEC, CRC and dual sync word detection.

### 2.2 Architecture

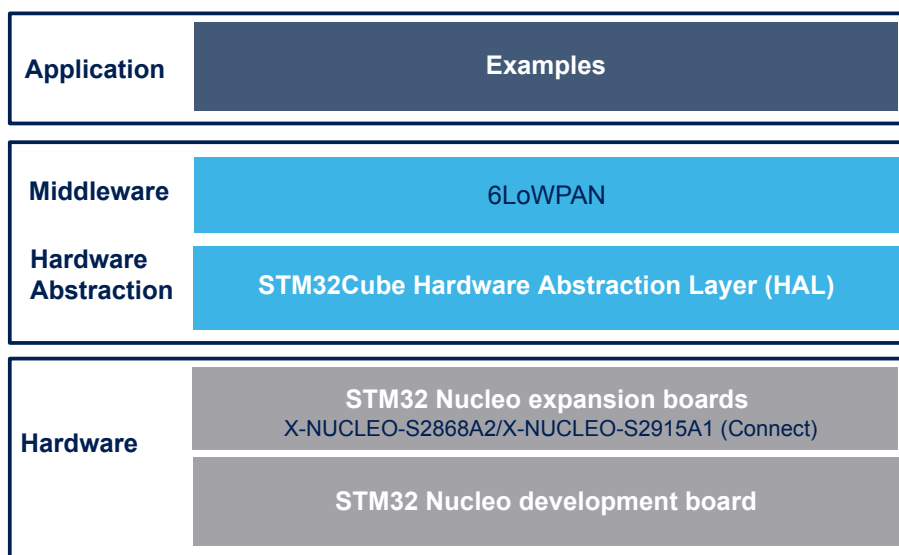
This software is fully compliant with and expands STM32Cube to enable development of applications using X-NUCLEO-S2868A1, X-NUCLEO-S2868A2 or X-NUCLEO-S2915A1 expansion board hosting the S2-LP devices.

The software is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller. The package extends STM32Cube by providing a board support package (BSP) for the S2-LP expansion board and firmware examples for P2P communication.

The software layers used by the application software to access and use the [S2-LP](#) expansion board are:

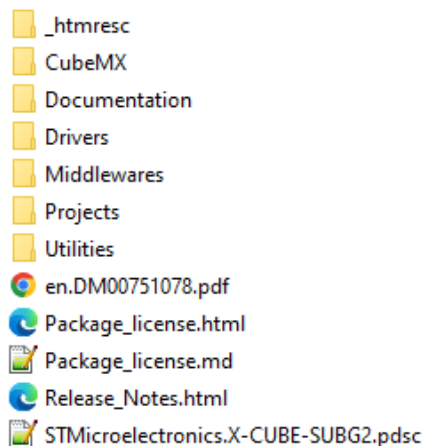
- STM32Cube HAL layer: provides a generic, multi-instance set of APIs to interact with the upper layers (the application, libraries and stacks). It consists of generic and extension APIs based on a common architecture which allows other layers like the middleware layer to function without specific Microcontroller Unit (MCU) hardware configurations. This structure improves library code reusability and guarantees easy device portability.
- Board support package (BSP) Layer: includes the software to support the peripherals on the [STM32 Nucleo](#) board (apart from the MCU). It is a set of APIs which provides a programming interface for certain board-specific peripherals (LED, user button etc.). The [X-NUCLEO-S2868A1](#), [X-NUCLEO-S2868A2](#) or [X-NUCLEO-S2915A1](#) expansion board BSP firmware layer contains APIs for the hardware components and consists of:
  - Component driver: related to the on-board device (not related to the STM32). The [S2-LP](#) BSP driver is known as the firmware component. The [S2-LP](#) component driver provides specific APIs and can be ported to and used on any board.
  - BSP driver: enables the component driver to be linked to a specific board and provides a set of user-friendly APIs.
- Application layer: provides the CLI example to be used with the [S2-LP](#) DK GUI to configure the [S2-LP](#) radio.
- Application layer: provides FIFO TX/RX examples to demonstrate multi-GPIOs usage and how to send packets longer than the physical radio payload.
- Application layer: provides a Point-to-Point communication example for sending a buffer from one node to another and acknowledgments using the [S2-LP](#) link layer features.
- Application layer for Contiki-NG based applications: provides 6LoWPAN communication for mesh-network as well as featured examples for UDP Client/Server, Serial Sniffer and Border Router.

**Figure 1. X-CUBE-SUBG2 software architecture**



## 2.3 Folder structure

Figure 2. X-CUBE-SUBG2 package folder structure



The software package includes the following folders:

- 'CubeMX': contains the meta data files for the package support in the [STM32CubeMX](#) tool.
- 'Documentation': contains a compiled HTML file generated from the source code and detailed documentation of the software components and APIs
- 'Drivers': contains the HAL drivers and the board-specific drivers for supported board and hardware platforms, including those for the on-board components and the CMSIS vendor-independent hardware abstraction layer for the Cortex-M processor series
- 'Middlewares': contains Contiki-NG source code for 6LoWPAN communication over [S2-LP](#) radio
- 'Projects': contains sample applications used for P2P, CLI, FIFO TX/RX firmware examples, and 6LoWPAN communication for the [NUCLEO-L053R8](#) (P2P, CLI and FIFO TX/RX only), [NUCLEO-F401RE](#), or [NUCLEO-L152RE](#) platforms with three development environments (IAR Embedded Workbench for ARM (IAR-EWARM), RealView Microcontroller Development Kit (MDK-ARM-STR) and [STM32CubeIDE](#))
- 'Utilities': contains tools to be used on a host PC to interact with Serial Sniffer and Border Router firmware applications (6LoWPAN)

## 2.4 APIs

Detailed descriptions of all the functions and parameters of the user APIs user can be found in a compiled HTML file located inside the 'Documentation' folder.

## 3 Point-to-Point (P2P) demo firmware description

### 3.1 P2P application details

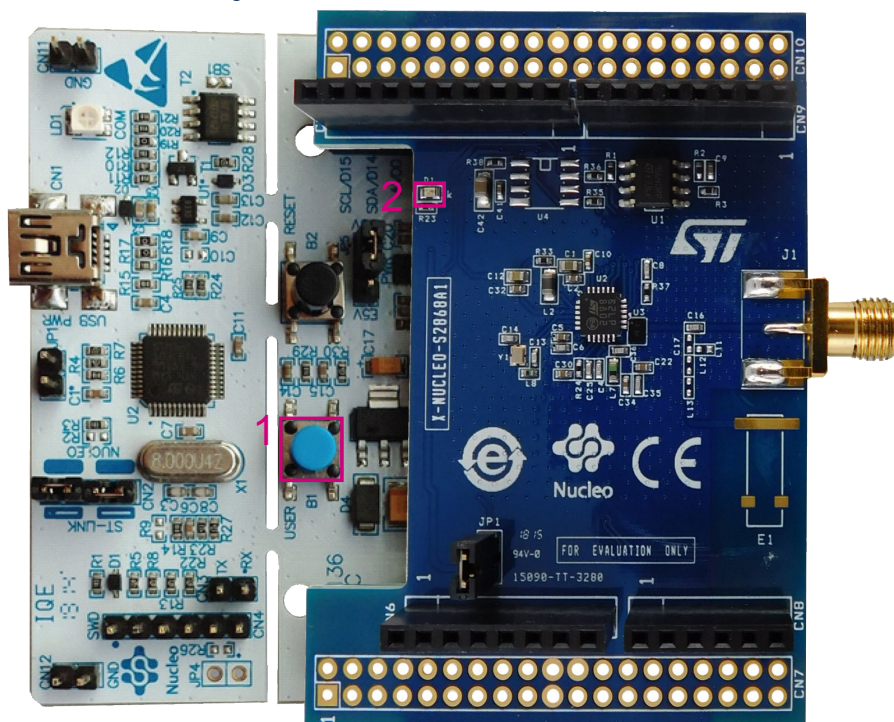
The P2P application operates using two nodes (STM32 Nucleo development board plus S2-LP expansion board) as follows:

1. by pressing the STM32 Nucleo board user button (shown in the picture below), each node can transmit a buffer to the other node
2. on receiving the signal, the receiver node LED lights up and an acknowledgment (ACK) signal is returned to the transmitter node
3. on reception of the ACK signal, the transmitter node LED flashes four times and switches off after a delay period

**Figure 3. X-NUCLEO-S2868A1 plus STM32 Nucleo used as a node (transmitter/receiver) in P2P communication**

1. STM32 Nucleo user button
2. X-NUCLEO-S2868A1 expansion board LED

**Note:** The LED on the expansion board is mounted but not connected: only the STM32 Nucleo LED blinks during P2P communication.



### 3.2 Application state diagram

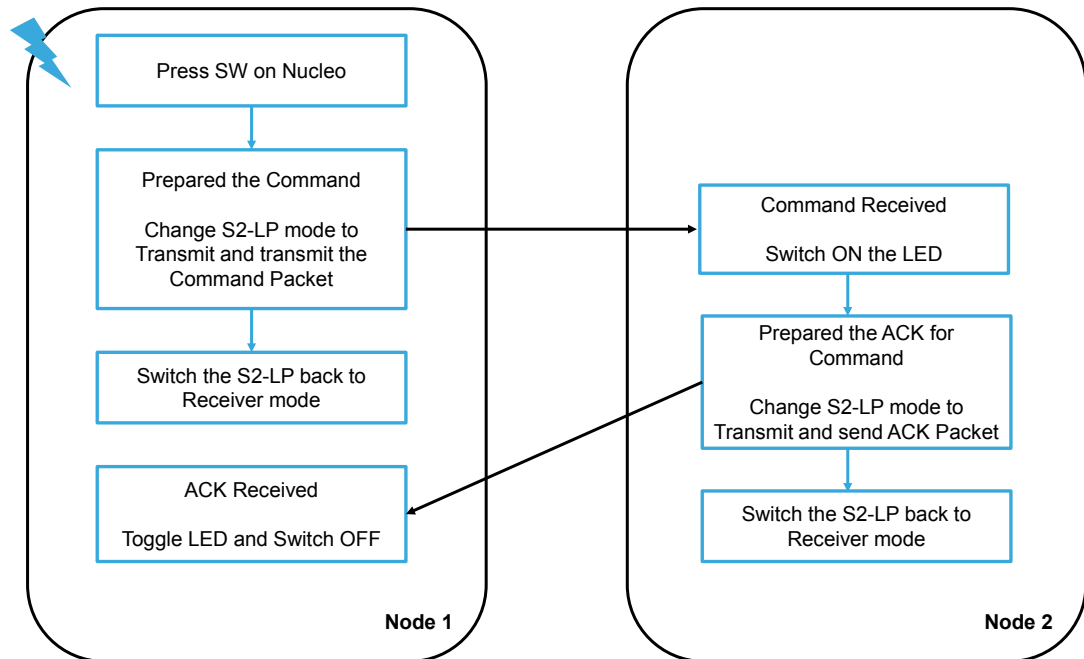
When running the demo sample with the STM32 Nucleo boards, S2-LP remains by default in receive mode but changes to transmit mode when the user button is pressed.

Once transmission stops, the transceiver returns to its default receive mode. On successful completion of the two-way communication (Command/ Ack), the MCU enters low-power mode.

To limit low-power mode current consumption, the LED is switched off by default.

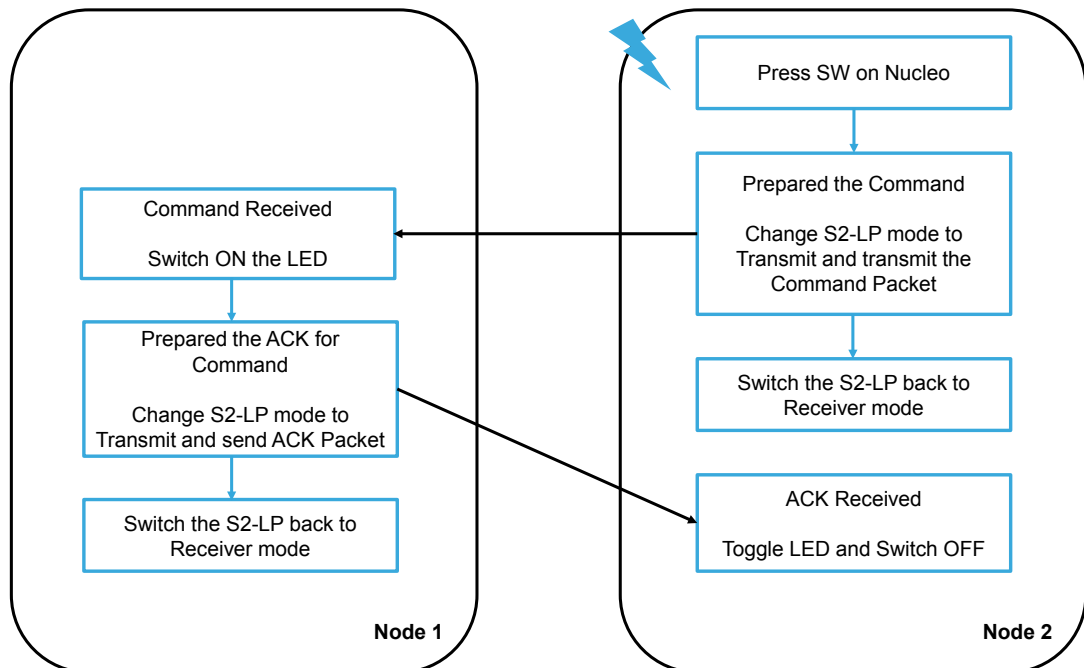
P2P nodes have the same functionality; the address of each node is set in the firmware by the user.

**Figure 4. Application state diagram when Node 1 user button is pressed**



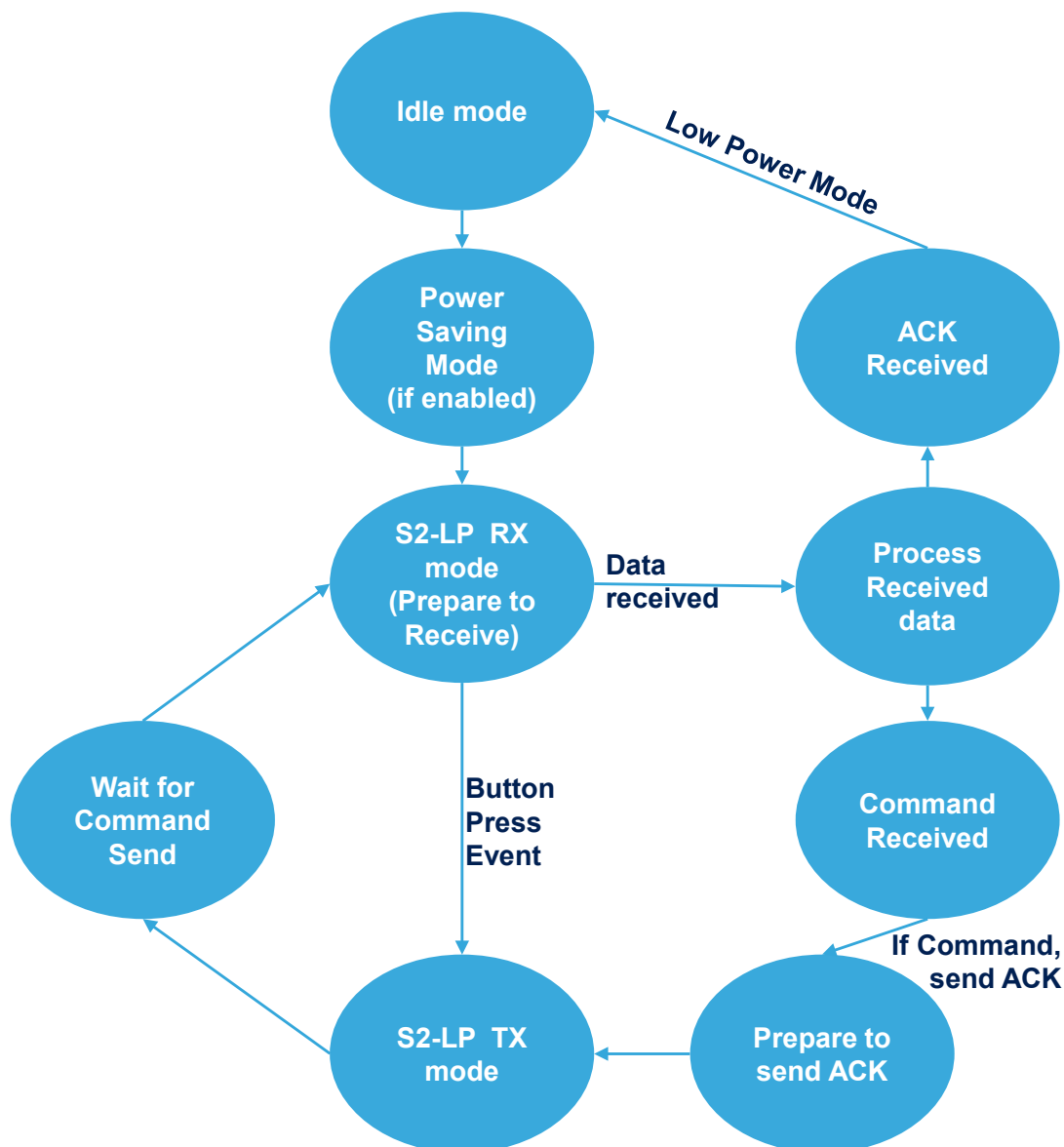
If the user presses the other node user button, the functionality is the same: Node 2 wakes up from low-power mode, prepares the command for transmission, sends the data packet and waits for acknowledgment.

**Figure 5. Application state diagram when Node 2 user button is pressed**



The following diagram shows data communication in low-power mode.

**Figure 6.** Application state diagram (low-power mode): data communication transmit and receive states



### 3.3 S2-LP packet handler overview

Before on-air transmission, raw data is arranged in a packet structure. S2-LP offers a highly flexible and fully programmable packet which lets you configure the structure of the packet, the number, the type, and the dimension of the fields inside the packet.

Through a register, the user can choose from one of the formats shown in the tables below.

**Table 2. Stack**

Preamble	Sync	Length	Destination address	Source address	Control	Seq. no.	No ACK	Payload	CRC
----------	------	--------	---------------------	----------------	---------	----------	--------	---------	-----

**Table 3. wM-Bus**

Preamble	Sync	Payload	Postamble
----------	------	---------	-----------



Table 4. Basic

Preamble	Sync	Length	Destination address	Control	Payload	CRC
----------	------	--------	---------------------	---------	---------	-----

See S2-LP datasheet for further details on the embedded packet handler.

Since P2P communication requires the receiving node destination address, the P2P demo is based on stack and basic packet handlers.

*Note:* The wM-Bus packet format is not used in this sample demonstration.

Table 5. Packet handler feature comparison

Features	Stack	wM-Bus	Basic
Destination address filtering	Yes	No	Yes
Broadcast and multicast addressing	Yes	No	Yes
Source address filtering	Yes	No	No
Custom filtering	Yes	No	Yes
CRC filtering	Yes	No	Yes
LLP: automatic acknowledgment <sup>(1)</sup>	Yes	No	No
LLP: automatic acknowledgment with piggybacking <sup>(1)</sup>	Yes	No	No
LLP: automatic retransmission <sup>(1)</sup>	Yes	No	No

1. Link layer protocol

### 3.4 Transmit and receive (command and response) packet structure

Command packet features:

- command with data sent at the same time
- S2-LP can handle 65535 bytes of data
- customizable command structure
- customizable data packet maximum size

Figure 7. Command data packet structure

Preamble	Sync	Length	Destination Address	Source Address	Control	Seq. No.	No ACK	Payload	CRC
----------	------	--------	---------------------	----------------	---------	----------	--------	---------	-----

STack packet

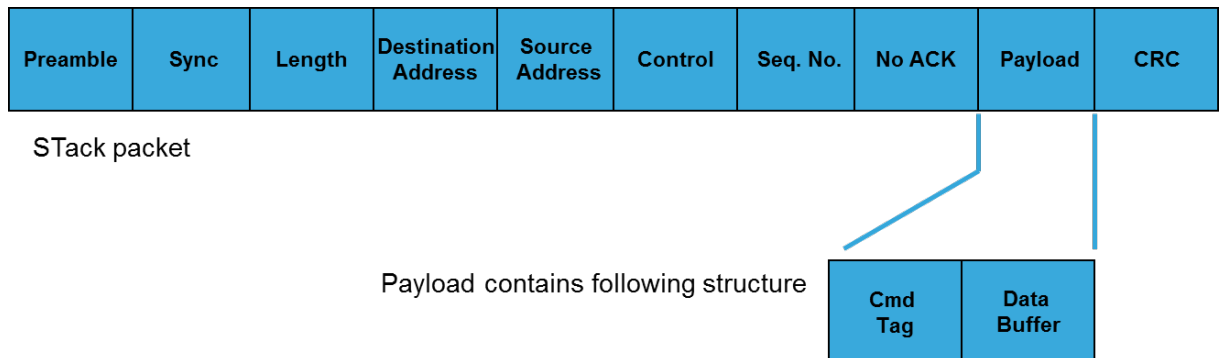
Payload contains following structure

Cmd Length	Cmd Tag	Cmd Type	Commands	Data Length	Data Buffer
------------	---------	----------	----------	-------------	-------------

Response packet features:

- data buffer is replied from the node
- tag contains the number associated with the command so the receiver can associate the response with the specific command

Figure 8. Response packet structure



### 3.4.1 Packet field description

- **Cmd Length:** the basic command is 1 byte long, but you can set multiple command bytes
- **Cmd Tag:** a unique tag number is linked to each command issued from the node and the response must replicate the same number
- **Cmd Type:** flag to identify application level or network command
- **Commands:** the actual command set sent from the source to destination (it may include parameters)
- **Data Length:** the data packet length
- **Data buffer:** the actual data associated with the command

## 3.5 User configuration

You can modify the configuration file s2868A1\_conf.h (or s2868a2\_conf.h, s2915a1\_conf.h depending on the expansion board) according to the application used.

### 3.5.1 Selecting packet handler

The user can select the desired features by setting the relevant macros:

```
#define USE_BASIC_PROTOCOL

#define USE_BASIC_PROTOCOL_ADDRESS /* to activate Basic protocol with Address field, to be
activated with USE_BASIC_PROTOCOL */
#define USE_RADIO_868MHz

/*...*/

#define EN_AUTOACK S_DISABLE
#define EN_PIGGYBACKING S_DISABLE
#define MAX_RETRANSMISSIONS PKT_DISABLE_RETX

#if defined(USE_STack_PROTOCOL)
//#define USE_STack_LLIP /*Uncomment if LLIP featured need to be used*/
#endif

#ifndef USE_BASIC_PROTOCOL
#define CSMA_ENABLE /* Comment this line to disable the CSMA */
#endif
```

By default, S2-LP works with the basic packet handler.

S2-LP uses the STack packet handler only if the link layer features (such as auto-ack, piggybacking and auto-retransmission) are defined.

### 3.5.2 Setting low-power mode

The P2P application supports low-power mode (disabled by default). It allows the MCU to either enter stop or sleep mode (check app\_x-cube-subg2.c file for these settings).

Optional Low Power modes for the S2-LP can also be activated (RF\_\* macros).

```
#if defined (USE_LOW_POWER_MODE)
//#define MCU_SLEEP_MODE
#define MCU_STOP_MODE
#endif
/*Possible Low Power modes for the S2-LP: */
//#define RF_STANDBY
//#define RF_SHUTDOWN
//#define RF_SLEEP
```

### 3.5.3 Setting radio configuration parameters

You can set the radio parameters in the configuration file, even though it is not recommended to change them.

```
/* Radio configuration parameters */
#define XTAL_OFFSET_PPM 0
#define INFINITE_TIMEOUT 0.0

#ifdef USE_RADIO_868MHz
#define BASE_FREQUENCY 868.0e6
#endif

#define CHANNEL_SPACE 100e3
#define CHANNEL_NUMBER 0
#define DATARATE 38400
#define FREQ_DEVIATION 20e3
#define BANDWIDTH 100E3
#define POWER_INDEX 7
#define RECEIVE_TIMEOUT 2000.0 /*change the value for required timeout
period*/

#define RSSI_THRESHOLD -120 /* Default RSSI at reception, more
than noise floor */
#define CSMA_RSSI_THRESHOLD -90 /* Higher RSSI to Transmit.
If it's lower, the Channel will be seen as busy */

/* Packet configuration parameters */

#define MODULATION_SELECT MOD_2FSK
#define POWER_DBM 12.0
```

### 3.5.4 Setting packet configuration parameters

You can set the packet configuration, even though it is not recommended to change default settings.

```
/* Packet configuration parameters */
#define SYNC_WORD 0x88888888
#define LENGTH_WIDTH 7
#define CRC_MODE PKT_CRC_MODE_8BITS
#define EN_FEC S_DISABLE
#define EN_WHITENING S_ENABLE

#define PREAMBLE_LENGTH PREAMBLE_BYTE(4)
#define SYNC_LENGTH SYNC_BYTE(4)
#define CONTROL_LENGTH 0x00
#define VARIABLE_LENGTH S_ENABLE
#define EXTENDED_LENGTH_FIELD S_DISABLE

#define PREAMBLE_BYTE(v) (4*v)
#define SYNC_BYTE(v) (8*v)
```

### 3.5.5 Setting node address

Node addresses parameters can be set in following section of the system setup guide.

```

/* Addresses configuration parameters */
#define EN_ADDRESS          S_DISABLE
#define EN_FILT_MY_ADDRESS  S_DISABLE
#define EN_FILT_MULTICAST_ADDRESS S_DISABLE
#define EN_FILT_BROADCAST_ADDRESS S_DISABLE
#define EN_FILT_SOURCE_ADDRESS S_DISABLE
#define SOURCE_ADDR_MASK    0xf0
#define SOURCE_ADDR_REF     0x37
#define MULTICAST_ADDRESS   0xEE
#define BROADCAST_ADDRESS   0xFF

```

### 3.5.6 User defined commands and macros

```

/* User Command */
#define APPLI_CMD          0x11
#define NWK_CMD            0x22
#define LED_TOGGLE        0xff
#define ACK_OK             0x01
#define MAX_BUFFER_LEN     96
#define TIME_TO_EXIT_RX    3000
#define DELAY_RX_LED_TOGGLE 100
#define DELAY_TX_LED_GLOW   100
#define LPM_WAKEUP_TIME    100

```

## 4 6LoWPAN applications

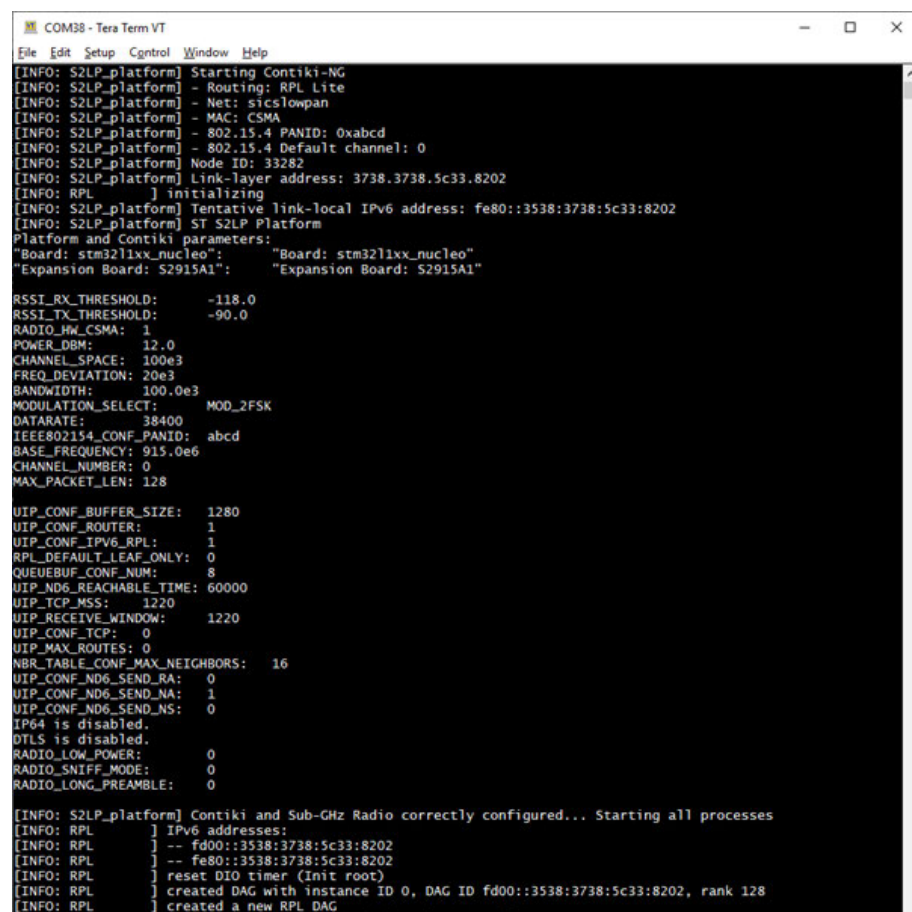
### 4.1 Contiki-NG

Contiki-NG is an OS for embedded systems providing 6LoWPAN stack on top of 802.15.4-like radio transceivers. Contiki-NG is included as a third-party middleware library in [X-CUBE-SUBG2](#) (starting from version 3.0.0). The software includes samples to send messages via UDP over 6LoWPAN, using the [S2-LP](#) sub-1GHz radio transceiver. The key features are:

- Middleware library with Contiki-NG OS protocol stack 4.5
- Support for mesh networking technology via the standard RPL protocol (lite, with default support for non-storing mode, and classic versions are available)
- Sample applications (such as UDP Client and UDP Server, Serial Sniffer and RPL Border Router)
- Nodes can be monitored and controlled using the embedded Serial Shell and the flexible logging system
- Link-Layer encryption using AES128
- 802.15.4 MAC ACK packets
- Optional end-to-end security using TinyDTLS
- Samples available for [NUCLEO-F401RE](#) and [NUCLEO-L152RE](#)
- Easy portability across different MCU families, thanks to [STM32Cube](#)
- Free and user-friendly license terms

At boot time, the firmware based on Contiki-NG prints a set of information regarding the system identification and chosen parameters as shown below.

**Figure 9. System configuration and parameters print at boot time (example)**



```

COM38 - Tera Term VT
File Edit Setup Control Window Help
[INFO: S2LP_platform] Starting Contiki-NG
[INFO: S2LP_platform] - Routing: RPL Lite
[INFO: S2LP_platform] - Net: 6LoWPAN
[INFO: S2LP_platform] - MAC: CSMA
[INFO: S2LP_platform] - 802.15.4 PANID: 0xabcd
[INFO: S2LP_platform] - 802.15.4 Default channel: 0
[INFO: S2LP_platform] Node ID: 33282
[INFO: S2LP_platform] Link-layer address: 3738.3738.5c33.8202
[INFO: RPL ] initializing
[INFO: S2LP_platform] Tentative link-local IPv6 address: fe80::3538:3738:5c33:8202
[INFO: S2LP_platform] ST S2LP Platform
Platform and Contiki parameters:
"Board: stm32l1xx_nucleo": "Board: stm32l1xx_nucleo"
"Expansion Board: S2915A1": "Expansion Board: S2915A1"

RSSI_RX_THRESHOLD: -118.0
RSSI_TX_THRESHOLD: -90.0
RADIO_HW_CSMA: 1
POWER_DBM: 12.0
CHANNEL_SPACE: 100e3
FREQ_DEVIATION: 20e3
BANDWIDTH: 100.0e3
MODULATION_SELECT: MOD_2FSK
DATABRATE: 38400
IEEE802154_CONF_PANID: abcd
BASE_FREQUENCY: 915.0e6
CHANNEL_NUMBER: 0
MAX_PACKET_LEN: 128

UIP_CONF_BUFFER_SIZE: 1280
UIP_CONF_ROUTER: 1
UIP_CONF_IPV6_RPL: 1
RPL_DEFAULT_LEAF_ONLY: 0
QUEUEBUF_CONF_NUM: 8
UIP_ND6_REACHABLE_TIME: 60000
UIP_TCP_MSS: 1220
UIP_RECEIVE_WINDOW: 1220
UIP_CONF_TCP: 0
UIP_MAX_ROUTES: 0
NBR_TABLE_CONF_MAX_NEIGHBORS: 16
UIP_CONF_ND6_SEND_RA: 0
UIP_CONF_ND6_SEND_NA: 1
UIP_CONF_ND6_SEND_NS: 0
IP64 is disabled.
DTLS is disabled.
RADIO_LOW_POWER: 0
RADIO_SNIFF_MODE: 0
RADIO_LONG_PREAMBLE: 0

[INFO: S2LP_platform] Contiki and Sub-GHz Radio correctly configured... Starting all processes
[INFO: RPL ] IPv6 addresses:
[INFO: RPL ] -- fd00::3538:3738:5c33:8202
[INFO: RPL ] -- fe80::3538:3738:5c33:8202
[INFO: RPL ] reset DIO timer (Init root)
[INFO: RPL ] created DAG with instance ID 0, DAG ID fd00::3538:3738:5c33:8202, rank 128
[INFO: RPL ] created a new RPL DAG

```

For the Contiki-NG firmware, interaction with a Serial Terminal utility is available. The utility has to be configured with the following parameters: 115200 bps, 8 bit, No Parity, 1 stop bit.

#### 4.1.1 UDP Client and UDP Server sample application overview

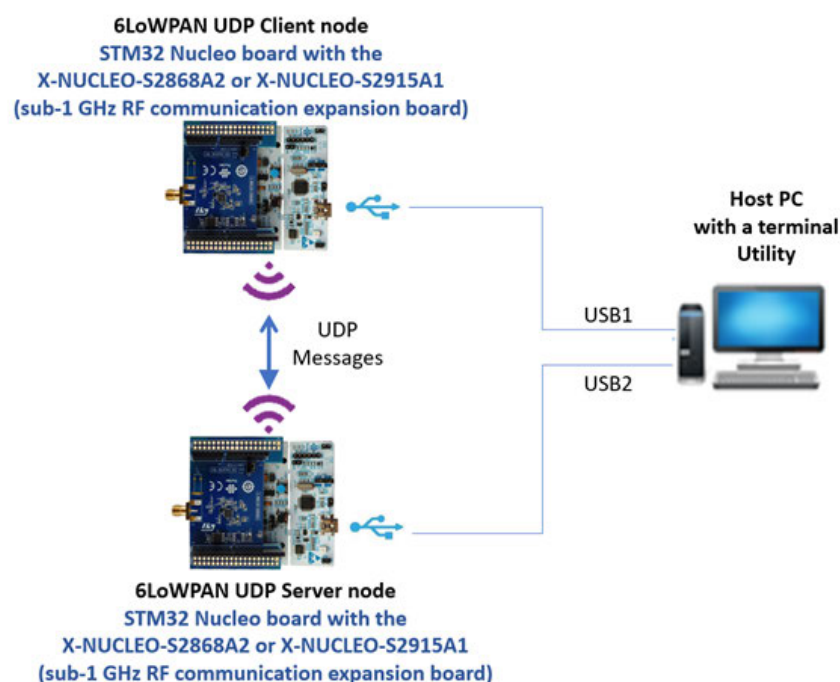
UDP Client and UDP Server sample application workflow can be summarized as follows:

1. the UDP Client node periodically creates a message, prints it on the terminal window along with the IPv6 address of the UDP Server it is going to transmit to (usually the RPL root) and sends it via UDP packets over the air
2. the UDP Server node is indefinitely listening for UDP packets, until it receives the data packets from the UDP Client node
3. the UDP Server node sends back a UDP packet with a reply and outputs the received message packet content in the terminal window
4. the UDP Client node receives the answer from the UDP Server and prints the received message in the terminal window

If more than one UDP Client node is used, the UDP Server receives messages from and send back replies to all of them.

By default, the UDP Server firmware is setup as root of the RPL tree and by default the UDP Client firmware sends messages to the IPv6 address of the RPL root (i.e. the root of the tree is also the UDP Server, no additional configuration is required).

**Figure 10. UDP Client and UDP Server node communication with a PC**



Contiki-NG provides a flexible log system that can be controlled using the Serial Shell (enabled by default).

The UDP Client and UDP Server firmware starts with a verbose log level for RPL, MAC and FRAMER layers, as per the following configuration (that can be found in the project-conf.h file at application level):

```
#define LOG_CONF_LEVEL_MAC LOG_LEVEL_DBG
#define LOG_CONF_LEVEL_RPL LOG_LEVEL_DBG
#define LOG_CONF_LEVEL_FRAMER LOG_LEVEL_ERR
```

The following figures show the terminal outputs.



Figure 11. UDP Server console output

```

Tera Term - [disconnected] VT
File Edit Setup Control Window Help
[INFO: S2LP_platform] Contiki and Sub-GHz Radio correctly configured... Starting all processes
[INFO: CSMA] received packet from 3738.3738.5c33:8202, seqno 65535, len 80
[INFO: RPL] received a multicast-DIO from fe80::3538:3738:5c33:8202, instance_id 0, DAG ID fd00::3538:3738:5c33:8202, version 240, dst
n 240, rank 128
[INFO: RPL] adding global IP address fd00::3532:3230:6b36:700d
[INFO: RPL] reset MHOP
[INFO: RPL] initialized DAG with instance ID 0, DAG ID fd00::3538:3738:5c33:8202, prefix fd00::/64, rank 65535
[WARN: RPL] just joined, no parent yet, setting timer for leaving
[INFO: RPL] refreshing lifetime
[INFO: RPL] best parent is not fresh, schedule urgent probing to fe80::3538:3738:5c33:8202
[INFO: RPL] probing fe80::3538:3738:5c33:8202 (urgent) last tx 0 min ago
[INFO: RPL] sending a unicast-DIO with rank 65535 to fe80::3538:3738:5c33:8202
[INFO: CSMA] sending to 3738.3738.5c33:8202, len 79, seqno 221, queue length 1, free packets 7
[DBG: CSMA] scheduling transmission in 27 ticks, NB=0, BE=3
[INFO: CSMA] preparing packet for 3738.3738.5c33:8202, seqno 221, tx 0, queue 1
[INFO: CSMA] LLSEC-OUT:3732.3230.6b36.700d 3738.3738.5c33:8202 83 (109) LV:5, KEY:0x00
[INFO: CSMA] tx to 3738.3738.5c33:8202, seqno 221, status 0, tx 0, coll 0
[INFO: CSMA] packet sent to 3738.3738.5c33:8202, seqno 221, status 0, tx 1, coll 0
[DBG: CSMA] free_queued_packet, queue length 0, free packets 8
[INFO: RPL] packet sent to 3738.3738.5c33:8202, status 0, tx 1, new link metric 224
[INFO: RPL] best parent is not fresh, schedule urgent probing to fe80::3538:3738:5c33:8202
[INFO: App] Not reachable yet
[INFO: RPL] probing fe80::3538:3738:5c33:8202 (urgent) last tx 0 min ago
[INFO: RPL] sending a unicast-DIO with rank 65535 to fe80::3538:3738:5c33:8202
[INFO: CSMA] sending to 3738.3738.5c33:8202, len 79, seqno 222, queue length 1, free packets 7
[DBG: CSMA] scheduling transmission in 18 ticks, NB=0, BE=3
[INFO: CSMA] preparing packet for 3738.3738.5c33:8202, seqno 222, tx 0, queue 1
[INFO: CSMA] LLSEC-OUT:3732.3230.6b36.700d 3738.3738.5c33:8202 83 (109) LV:5, KEY:0x00
[INFO: CSMA] tx to 3738.3738.5c33:8202, seqno 222, status 0, tx 0, coll 0
[INFO: CSMA] packet sent to 3738.3738.5c33:8202, seqno 222, status 0, tx 1, coll 0
[DBG: CSMA] free_queued_packet, queue length 0, free packets 8
[INFO: RPL] packet sent to 3738.3738.5c33:8202, status 0, tx 1, new link metric 200
[INFO: RPL] best parent is not fresh, schedule urgent probing to fe80::3538:3738:5c33:8202
[INFO: RPL] probing fe80::3538:3738:5c33:8202 (urgent) last tx 0 min ago
[INFO: RPL] sending a unicast-DIO with rank 65535 to fe80::3538:3738:5c33:8202
[INFO: CSMA] sending to 3738.3738.5c33:8202, len 79, seqno 223, queue length 1, free packets 7
[DBG: CSMA] scheduling transmission in 22 ticks, NB=0, BE=3
[INFO: CSMA] preparing packet for 3738.3738.5c33:8202, seqno 223, tx 0, queue 1
[INFO: CSMA] LLSEC-OUT:3732.3230.6b36.700d 3738.3738.5c33:8202 83 (109) LV:5, KEY:0x00
[INFO: CSMA] tx to 3738.3738.5c33:8202, seqno 223, status 0, tx 0, coll 0
[INFO: CSMA] packet sent to 3738.3738.5c33:8202, seqno 223, status 0, tx 1, coll 0
[DBG: CSMA] free_queued_packet, queue length 0, free packets 8

```

Figure 12. UDP Client console output

```

Tera Term - [disconnected] VT
File Edit Setup Control Window Help
[INFO: S2LP_platform] Contiki and Sub-GHz Radio correctly configured... Starting all processes
[INFO: CSMA] received packet from 3738.3738.5c33:8202, seqno 65535, len 80
[INFO: RPL] received a multicast-DIO from fe80::3538:3738:5c33:8202, instance_id 0, DAG ID fd00::3538:3738:5c33:8202, version 240, dst
n 240, rank 128
[INFO: RPL] adding global IP address fd00::3532:3230:6b36:700d
[INFO: RPL] reset MHOP
[INFO: RPL] initialized DAG with instance ID 0, DAG ID fd00::3538:3738:5c33:8202, prefix fd00::/64, rank 65535
[WARN: RPL] just joined, no parent yet, setting timer for leaving
[INFO: RPL] refreshing lifetime
[INFO: RPL] best parent is not fresh, schedule urgent probing to fe80::3538:3738:5c33:8202
[INFO: RPL] probing fe80::3538:3738:5c33:8202 (urgent) last tx 0 min ago
[INFO: RPL] sending a unicast-DIO with rank 65535 to fe80::3538:3738:5c33:8202
[INFO: CSMA] sending to 3738.3738.5c33:8202, len 79, seqno 221, queue length 1, free packets 7
[DBG: CSMA] scheduling transmission in 27 ticks, NB=0, BE=3
[INFO: CSMA] preparing packet for 3738.3738.5c33:8202, seqno 221, tx 0, queue 1
[INFO: CSMA] LLSEC-OUT:3732.3230.6b36.700d 3738.3738.5c33:8202 83 (109) LV:5, KEY:0x00
[INFO: CSMA] tx to 3738.3738.5c33:8202, seqno 221, status 0, tx 0, coll 0
[INFO: CSMA] packet sent to 3738.3738.5c33:8202, seqno 221, status 0, tx 1, coll 0
[DBG: CSMA] free_queued_packet, queue length 0, free packets 8
[INFO: RPL] packet sent to 3738.3738.5c33:8202, status 0, tx 1, new link metric 224
[INFO: RPL] best parent is not fresh, schedule urgent probing to fe80::3538:3738:5c33:8202
[INFO: App] Not reachable yet
[INFO: RPL] probing fe80::3538:3738:5c33:8202 (urgent) last tx 0 min ago
[INFO: RPL] sending a unicast-DIO with rank 65535 to fe80::3538:3738:5c33:8202
[INFO: CSMA] sending to 3738.3738.5c33:8202, len 79, seqno 222, queue length 1, free packets 7
[DBG: CSMA] scheduling transmission in 18 ticks, NB=0, BE=3
[INFO: CSMA] preparing packet for 3738.3738.5c33:8202, seqno 222, tx 0, queue 1
[INFO: CSMA] LLSEC-OUT:3732.3230.6b36.700d 3738.3738.5c33:8202 83 (109) LV:5, KEY:0x00
[INFO: CSMA] tx to 3738.3738.5c33:8202, seqno 222, status 0, tx 0, coll 0
[INFO: CSMA] packet sent to 3738.3738.5c33:8202, seqno 222, status 0, tx 1, coll 0
[DBG: CSMA] free_queued_packet, queue length 0, free packets 8
[INFO: RPL] packet sent to 3738.3738.5c33:8202, status 0, tx 1, new link metric 200
[INFO: RPL] best parent is not fresh, schedule urgent probing to fe80::3538:3738:5c33:8202
[INFO: RPL] probing fe80::3538:3738:5c33:8202 (urgent) last tx 0 min ago
[INFO: RPL] sending a unicast-DIO with rank 65535 to fe80::3538:3738:5c33:8202
[INFO: CSMA] sending to 3738.3738.5c33:8202, len 79, seqno 223, queue length 1, free packets 7
[DBG: CSMA] scheduling transmission in 22 ticks, NB=0, BE=3
[INFO: CSMA] preparing packet for 3738.3738.5c33:8202, seqno 223, tx 0, queue 1
[INFO: CSMA] LLSEC-OUT:3732.3230.6b36.700d 3738.3738.5c33:8202 83 (109) LV:5, KEY:0x00
[INFO: CSMA] tx to 3738.3738.5c33:8202, seqno 223, status 0, tx 0, coll 0
[INFO: CSMA] packet sent to 3738.3738.5c33:8202, seqno 223, status 0, tx 1, coll 0
[DBG: CSMA] free_queued_packet, queue length 0, free packets 8

```

This logging level can be useful to debug a new developed application. The built-in Shell provides a list of useful commands that can be listed by selecting the **[help]** command as shown below.

**Figure 13. List of commands available in the Serial Shell**

```

Available commands:
> help': Shows this help
> reboot': Reboot the board by watchdog_reboot()
> log module level': Sets log level (0--4) for a given module (or "all"). For module "mac", level 4 also enables per-slot logging.
> mac-addr': Shows the node's MAC address
> ip-addr': Shows all IPv6 addresses
> ip-nbr': Shows all IPv6 neighbors
> ping addr': Pings the IPv6 address 'addr'
> routes': Shows the route entries
> rpl-set-root 0/1 [prefix]': Sets node as root (1) or not (0). A /64 prefix can be optionally specified.
> rpl-local-repair': Triggers a RPL local repair
> rpl-refresh-routes': Refreshes all routes through a DTSN increment
> rpl-status': Shows a summary of the current RPL state
> rpl-nbr': Shows the RPL neighbor table
> rpl-global-repair': Triggers a RPL global repair
> llsec-set-level <lv>': Set the level of link layer security (show if no lv argument)
> llsec-set-key <id> <key>': Set the key of link layer security
  
```

Different commands are available to show routes and neighbors, and to ping other nodes or to control and manipulate the status of the RPL tree.

The log level can be controlled as well: for any layer/module, a specific log level can be set from 0 up to the level set at compilation time for each layer/module. For example, the `log all 0` command disables the messages from lower layers (FRAMER, MAC and RPL in this case), leaving only the Applicative messages as shown in the following figures.

**Figure 14. Shell before and after log all 0 command**

```

Tera Term - (disconnected) VT
File Edit Setup Control Window Help

[INFO: App] Received response 'hello 2' from fd00::3538:3738:5c33:8202 LLSEC LV:5
[INFO: App] Sending request 3 to fd00::3538:3738:5c33:8202
[INFO: RPL] creating hop-by-hop option
[DBG : RPL] no SNH found
[INFO: CSMA] sending to 3738.3738.5c33.8202, len 27, seqno 231, queue length 1, free packets 7
[DBG : CSMA] scheduling transmission in 24 ticks, NB=0, BE=3
[INFO: CSMA] preparing packet for 3738.3738.5c33.8202, seqno 231, tx 0, queue 1
[INFO: CSMA] LLSEC-OUT:3732.3230.6b36.700d 3738.3738.5c33.8202 31 (57) LV:5, KEY:0x00
[INFO: CSMA] tx to 3738.3738.5c33.8202, seqno 231, status 0, tx 0, coll 0
[INFO: CSMA] packet sent to 3738.3738.5c33.8202, seqno 231, status 0, tx 1, coll 0
[DBG : CSMA] free_queued_packet, queue length 0, free packets 8
[INFO: RPL] packet sent to 3738.3738.5c33.8202, status 0, tx 1, new link metric 155
[INFO: CSMA] LLSEC-IN: 3738.3738.5c33.8202 3732.3230.6b36.700d 26 31 (57) LV:5 AM:0 KEY:0x00
[INFO: CSMA] received packet from 3738.3738.5c33.8202, seqno 99, len 27
[INFO: App] Received response 'hello 3' from fd00::3538:3738:5c33:8202 LLSEC LV:5

Log levels:
-- rpl : 0 (None)
-- tcpip : 0 (None)
-- ipv6 : 0 (None)
-- 6lowpan : 0 (None)
-- nullnet : 0 (None)
-- mac : 0 (None)
-- framer : 0 (None)
-- stop : 0 (None)
-- coap : 0 (None)
-- snmp : 0 (None)
-- lwmm : 0 (None)
-- main : 0 (None)

#3732.3230.6b36.700d> #3732.3230.6b36.700d> [INFO: App] Sending request 4 to fd00::3538:3738:5c33:8202
[INFO: App] Received response 'hello 4' from fd00::3538:3738:5c33:8202 LLSEC LV:5
[INFO: App] Sending request 5 to fd00::3538:3738:5c33:8202
[INFO: App] Received response 'hello 5' from fd00::3538:3738:5c33:8202 LLSEC LV:5
[INFO: App] Sending request 6 to fd00::3538:3738:5c33:8202
[INFO: App] Received response 'hello 6' from fd00::3538:3738:5c33:8202 LLSEC LV:5
[INFO: App] Sending request 7 to fd00::3538:3738:5c33:8202
[INFO: App] Received response 'hello 7' from fd00::3538:3738:5c33:8202 LLSEC LV:5
[INFO: App] Sending request 8 to fd00::3538:3738:5c33:8202
[INFO: App] Received response 'hello 8' from fd00::3538:3738:5c33:8202 LLSEC LV:5
[INFO: App] Sending request 9 to fd00::3538:3738:5c33:8202
[INFO: App] Received response 'hello 9' from fd00::3538:3738:5c33:8202 LLSEC LV:5
[INFO: App] Sending request 10 to fd00::3538:3738:5c33:8202
[INFO: App] Received response 'hello 10' from fd00::3538:3738:5c33:8202 LLSEC LV:5
  
```



Figure 15. UDP Client (left) and UDP Server (right) console output after log all 0 command

## 4.2 Serial Sniffer sample application overview

The Serial Sniffer application allows the user to capture RF packets and send them to an analysis tool (like Wireshark) to see and analyze the messages exchanged among the nodes in a 6LoWPAN network.

To be connected to Wireshark, the Serial Sniffer firmware needs some tools, to be launched on a host PC, located in the X-CUBE-SUBG2 Utilities/PC\_Software/SerialSniffer folder. A pre-requisite for using the tools is the use of Perl and, of course, Wireshark.

If running under Microsoft Windows, another pre-requisite is the Cygwin emulator of the Linux environment (in this case, Perl can be installed using the Cygwin setup).

### 4.2.1 How to use the Serial Sniffer application under Linux

**Step 1.** Compile serialdump-linux utility from X-CUBE-SUBG2 package.

```
cd Utilities/PC_Software/SerialSniffer/serialdump-src
make
(or gcc -o serialdump-linux serialdump.c)
mv serialdump-linux ..
cd ..
```

**Step 2.** From the Serial Sniffer folder, run the following command chain (one line command made up of three commands in line, separated by "|"):

```
> sudo serialdump-linux -b115200 /dev/ttyACMx | ./convert-to-binary |wireshark -k -i -
```

### 4.2.2 How to use the Serial Sniffer application under Windows

**Step 1.** Run Cygwin as administrator

**Step 2.** The serialdup-windows.exe utility is provided pre-compiled, but if you need to recompile it, from the X-CUBE-SUBG2 package root, run:

```
cd Utilities/PC_Software/SerialSniffer/serialdump-src

make
(or gcc -o serialdump-windows.exe serialdump.c)
mv serialdump-windows.exe ..
cd ..
```

**Step 3.** Run the following command chain (one line command made up of three commands in line, separated by "|"):

```
>serialdump-windows.exe -b115200 /dev/ttySx | ./convert-to-binary |wireshark.exe -k -i -
```

### 4.2.3 Important additional information

- In the commands mentioned in the previous sections, mind the trailing dash (-): it is mandatory
- The `ttySx / ttyACMx` numbers depend on the device enumeration of the [STM32 Nucleo](#) development board running the Serial Sniffer firmware: to this aim, you can use the auto-completion tab under Linux and Cygwin
- It is mandatory to invoke the above commands from the Serial Sniffer folder (actually, the `header.pcap` file should be in the same folder of the `convert-to-binary` script)
- Wireshark application is required: it is recommended to use the application latest version to have the state-of-the-art protocol dissectors
- In the above commands, Wireshark is supposed to be in the system shell path, otherwise you must provide the full command path or create a proper link
- Under Windows, if you need to use the full path for `wireshark.exe` and this contains spaces, use `"` as escape character before spaces and parenthesis
- Serial Sniffer firmware must be compiled with the same radio (channel, modulation, etc.) settings as the network under investigation
- On some Windows PCs, you may need to hardcode the baudrate (115200) in the `serialdump.c` code and recompile
- Also, on some Windows PCs, if you get the `"could not get options: Invalid argument"` error message, it is required to open a terminal utility (like Tera Term) on the console of the Serial Sniffer node, reset it (see the figure below), close the terminal and then run the command chain again in the Cygwin shell.

Figure 16. Serial Sniffer output in Tera Term

```

COM10 - Tera Term V1
File Edit Setup Control Window Help
[INFO: S2LP_platform] Link-layer address: 3559.3678.3034.5119
[INFO: S2LP_platform] ST S2LP Platform
Platform and Contiki parameters:
"Board: stm32f4xx_nucleo": "Board: stm32f4xx_nucleo"
"Expansion Board: S2868A2": "Expansion Board: S2868A2"

RSSI_RX_THRESHOLD: -118.0
RSSI_TX_THRESHOLD: -90.0
RADIO_HW_CSMA: 1
POWER_DBM: 12.0
CHANNEL_SPACE: 100e3
FREQ_DEVIATION: 20e3
BANDWIDTH: 100.0e3
MODULATION_SELECT: MOD_2FSK
DATARATE: 38400
IEEE802154_CONF_PANTO: abcd
BASE_FREQUENCY: 868.0e6
CHANNEL_NUMBER: 0
MAX_PACKET_LEN: 128

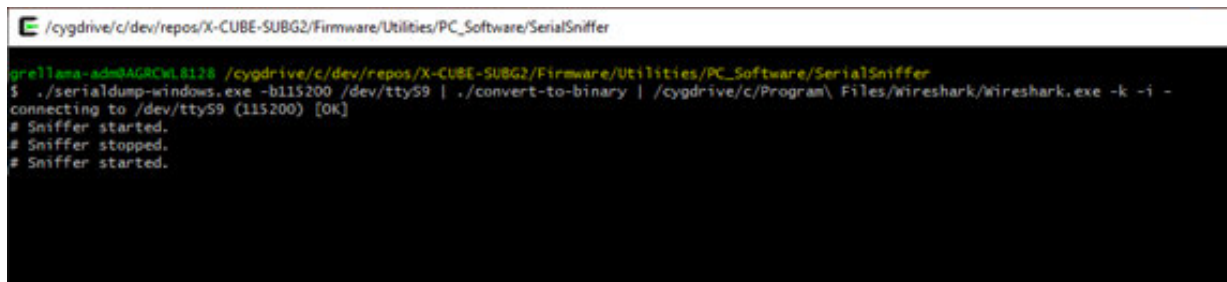
UIP_CONF_BUFFER_SIZE: 1280
UIP_CONF_ROUTER: 1
UIP_CONF_IPV6_RPL: 0
QUEUEBUF_CONF_NUM: 8
UIP_ND6_REACHABLE_TIME: 60000
UIP_TCP_MSS: 1220
UIP_RECEIVE_WINDOW: 1220
UIP_CONF_TCP: 0
UIP_MAX_ROUTES: 16
NBR_TABLE_CONF_MAX_NEIGHBORS: 16
UIP_CONF_ND6_SEND_RA: 0
UIP_CONF_ND6_SEND_RA: 0
UIP_CONF_ND6_SEND_NS: 0
IP64 is disabled.
DTLS is disabled.
RADIO_LOW_POWER: 0
RADIO_SNIFF_MODE: 0
RADIO_LONG_PREAMBLE: 0

[INFO: S2LP_platform] Contiki and Sub-GHz Radio correctly configured... Starting all processes
# Sniffer started but STOPPED. Press User Button to Start/Pause.

```

The Serial Sniffer firmware uses the User Button on the node to start/stop acquisition. It boots up in stop mode. This can be seen on the console (as shown in the figures above and below).

Figure 17. Cygwin shell: command line to interact with Serial Sniffer



```

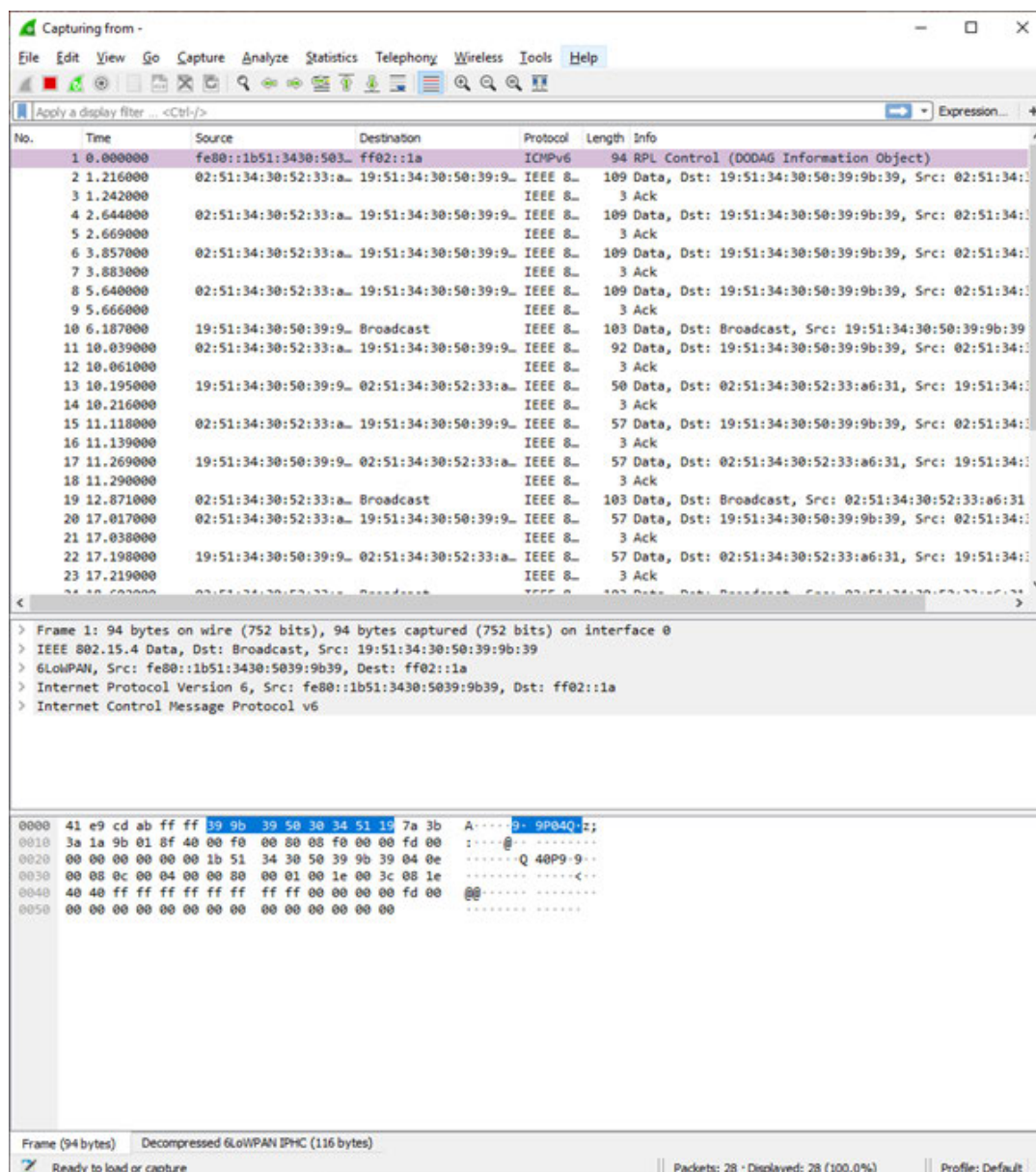
/cygdrive/c/dev/repos/X-CUBE-SUBG2/Firmware/Utilities/PC_Software/SerialSniffer

grellama-adm@AG8Cw1.8128 /cygdrive/c/dev/repos/X-CUBE-SUBG2/Firmware/Utilities/PC_Software/SerialSniffer
$ ./serialdump-windows.exe -b115200 /dev/ttyS9 | ./convert-to-binary | /cygdrive/c/Program\ Files/wireshark/wireshark.exe -k -i -
connecting to /dev/ttyS9 (115200) [OK]
# Sniffer started.
# Sniffer stopped.
# Sniffer started.
  
```

The sequence of commands will launch the Wireshark application through which the RF packets can be seen, analyzed and saved.

**Important:** *In X-CUBE-SUBG2, the default configuration of the Contiki-NG based application comes with Link Layer security enabled. In this case, the Serial Sniffer captures RF packets and shows them encrypted in Wireshark.*

**Figure 18. Message exchange flow between UDP Server and UDP Client (capture of Link Layer encrypted packets)**



Link Layer security can be disabled by editing the `contiki-conf.h` file (located in the `Inc/` folder of each application) for all the network node (UDP Client and UDP Server in this case) firmware and changing the value of the following macro from:

```
/* Enable Link Layer security */
#define LLSEC802154_CONF_ENABLED 1
```

to:

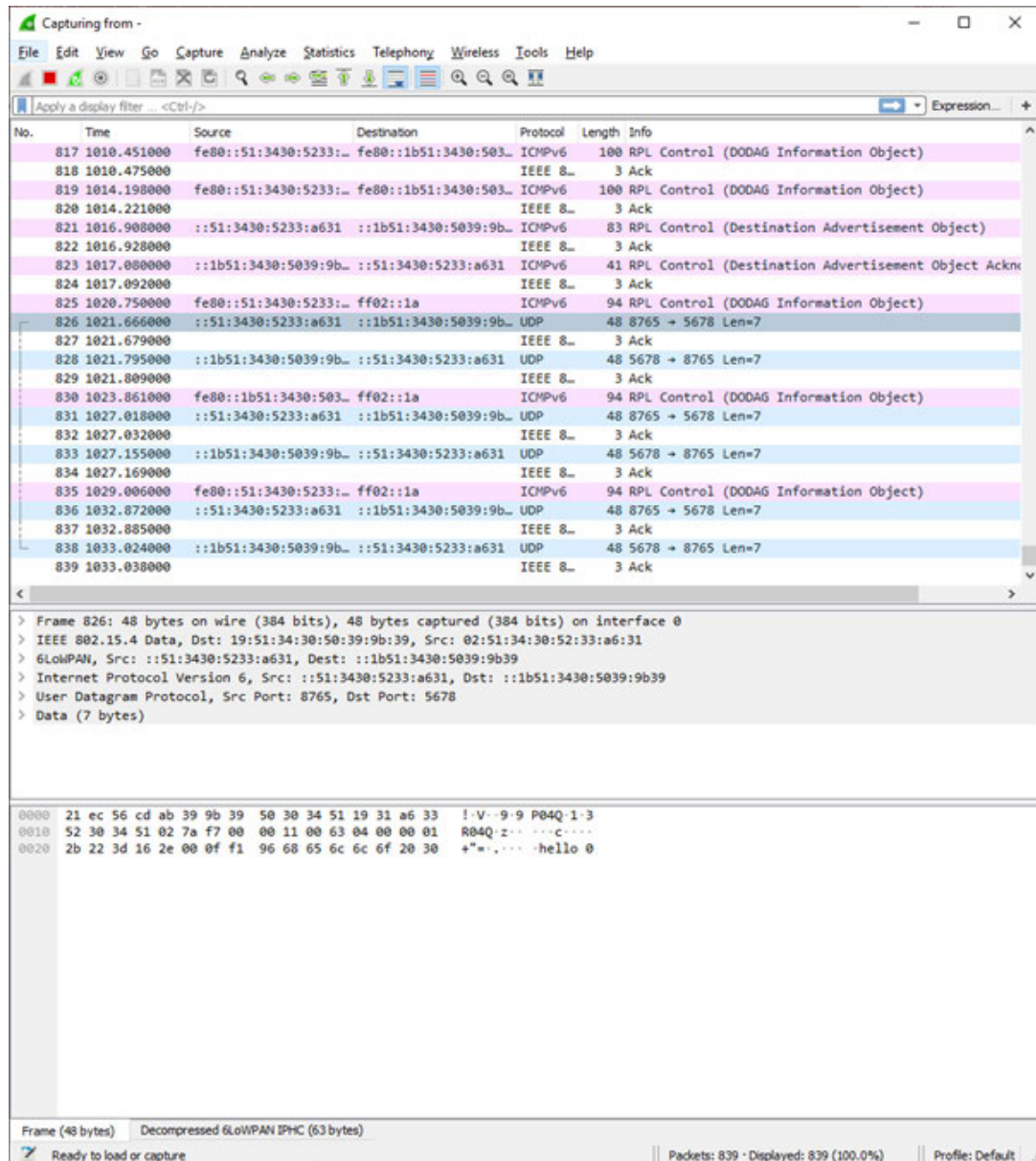
```
#define LLSEC802154_CONF_ENABLED 0
```

The firmware applications must now be recompiled and re-flashed on the nodes.

By disabling the Link Layer security, the UDP messages exchanged by the two applications are visible in clear in Wireshark.



**Figure 19. Message exchange flow between UDP Server and UDP Client (capture of Link Layer plain text packets)**



## 4.3 Border Router sample application overview

This example lets the user configure an “RPL Border Router” node between a 6LoWPAN network and a host system, such as a PC, typically connected to the Wide Area Network. The following sections describe how to integrate this border router into a system.

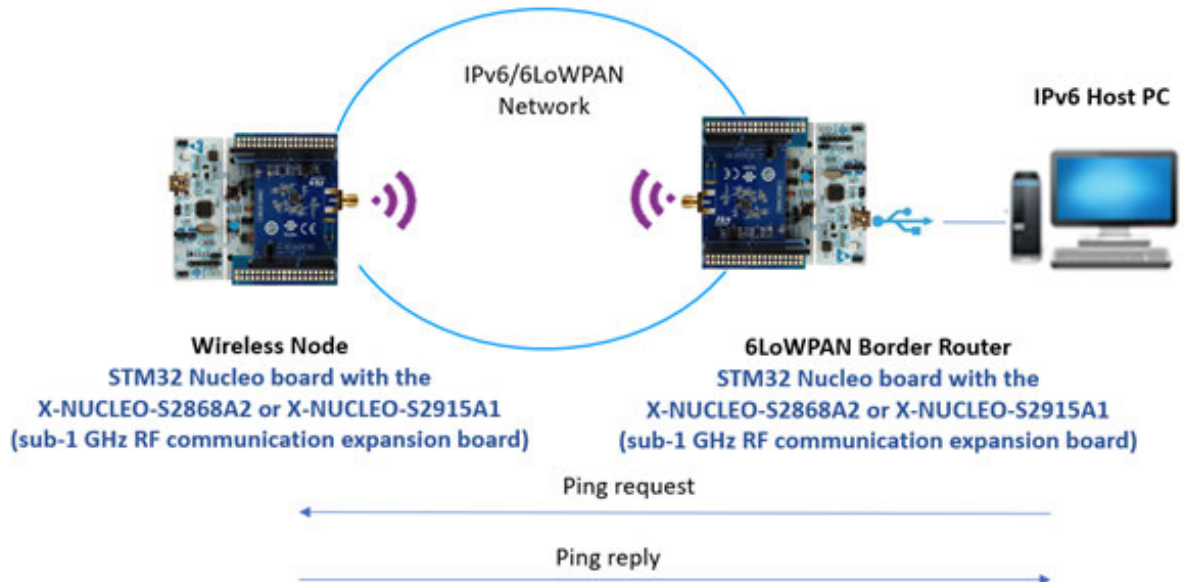
### 4.3.1 System overview

The Border Router application system architecture consists of:

- an IPv6 host device that runs the client application (e.g., a web browser) over an IPv6-based protocol stack. This device can also provide Wide Area Network connectivity
- an edge (or border) router device that creates a 6LoWPAN network and is connected to the wireless nodes on one side and to the IPv6 host on the other. In our case, this device is an [STM32 Nucleo](#) development board with a sub-1 GHz RF communication expansion board. To generate the firmware for this device, you have to select the “Border Router” project folder and compile the project with one of the supported IDEs or use the sample binary firmware provided in the same project folder

- a low-power wireless node connected to the 6LoWPAN network. In our case, this device is an [STM32 Nucleo](#) development board with a sub-1 GHz RF communication expansion board. To generate the firmware for this device, you have to select the “UDP Client” project folder and compile the project with one of the supported IDEs or use the sample binary firmware provided in the same project folder.

**Figure 20. System architecture for Border Router application**



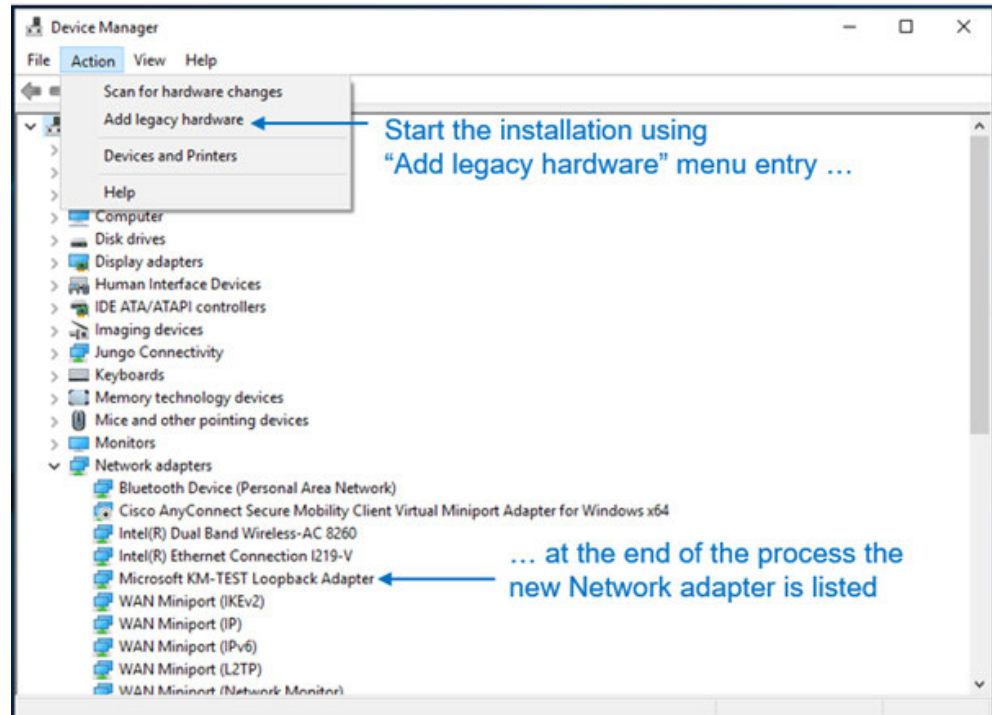
## 4.3.2 IPv6 host setup

### 4.3.2.1 IPv6 packet bridging - Windows setup

This setup information is for a Windows (version 7 and later) host PC; the host side implements a standard IPv6-based networking stack. The “wpcapslip6” application exchanges IPv6 packets over the serial line between the host PC stack and the border router IPv6 stack.

- Step 1.** Install the Microsoft loopback network adapter by selecting [**Add legacy hardware**] in the Device Manager.

**Figure 21. Adding the network adapter in Windows**



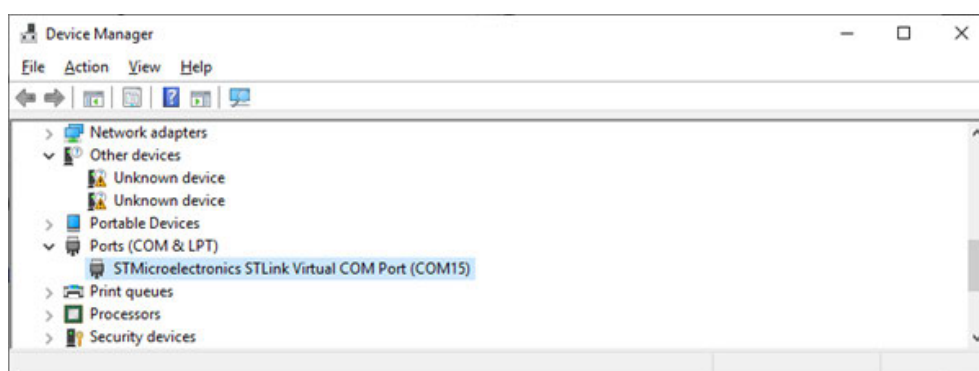
- Step 2.** Follow the Installer instructions by selecting the following entries: [**Add legacy hardware**]>[**Next**]>[**Install the hardware manually selected from a list**]>[**Next**]>[**Select Networks Adapters**]>[**Next**]>[**Select Microsoft from the Manufacturer list then Microsoft KM-TEST Loopback Adapter from the Network Adapter list**]>[**Next**] → then finish the installation by clicking [**Next**] again.
- Step 3.** Reboot your PC.
- Step 4.** Install the WinPcap Windows packet capture library. The installer and setup information can be found at <https://www.winpcap.org/install/>

- Step 5.** Launch the Command prompt shell and run the wpcaslip6 utility with a node running the Border Router firmware connected to the host PC.

```
cd [X-CUBE-SUBG2]\Utilities\PC_Software\wpcaslip6
wpcaslip6.exe -s /dev/ttyS14 -b aaaa:: -a aaaa::1/128 [addr]
```

You can obtain the serial device number to be used (in this example, “ttyS14”) by looking for the Virtual COM Port number associated with the [STM32 Nucleo](#) board. The value to use when running the “wpcaslip6.exe” command is the COM Port number minus 1. In our example, the COM Port number is 15 (hence the “ttyS14” device parameter).

**Figure 22. STM32 Nucleo board virtual COM Port value**



The [addr] parameter is the MAC address of the local network adapter (Microsoft KM-TEST Loopback Adapter). This information can be found by launching the `ipconfig /all` command shown below.

**Figure 23. Reading the MAC address of the Microsoft KM-TEST loopback adapter**

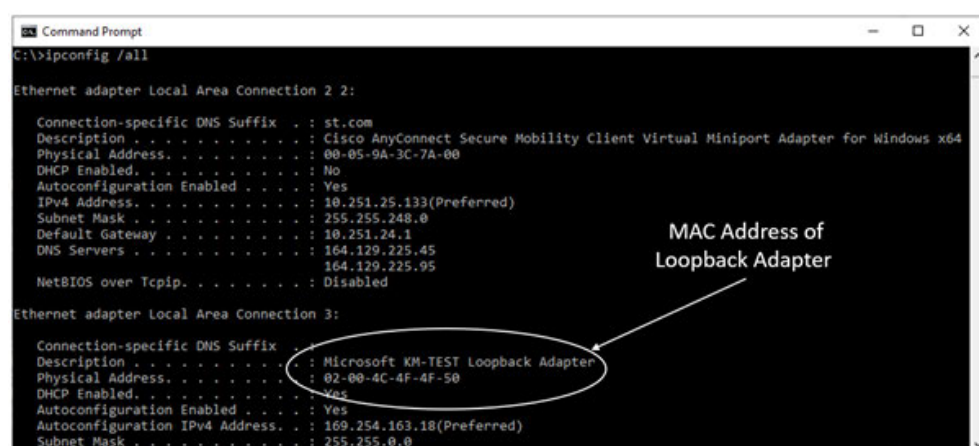




Figure 24. wpcapslip6 utility terminal window (snapshot)

```
$ ./wpcapslip6.exe -s /dev/ttyS14 -b aaaa:: -a aaaa::1/128 02-00-4C-4F-4F-50
Using local network interface: Local Area Connection 5
10:10:56 netsh interface ipv6 add address "Local Area Connection 5" aaaa::1/128
10:10:58 wpcapslip6 started on "/dev/ttyS14"
10:10:58 Got request message of type M
10:10:58 *** Gateway's MAC address: 08-00-f7-ff-b7-bd-48-42
10:10:58 Fictitious MAC-48: 0A-00-F7-8D-48-42
10:10:58 netsh interface ipv6 add route aaaa::/64 "Local Area Connection 5" aaaa::a00:f7ff:b7bd:4842
Ok.
10:10:58 netsh interface ipv6 add neighbor "Local Area Connection 5" aaaa::a00:f7ff:b7bd:4842 "0A-00-F7-8D-48-42"
10:10:58 Got configuration message of type 0
10:10:58 *** Address:aaaa:: => aaaa:0000:0000:0000
10:10:58 Got configuration message of type P
10:10:58 Setting prefix aaaa::
10:10:59 Server IPv6 addresses:
10:10:59 aaaa::a00:f7ff:b7bd:4842
10:10:59 fc00::a00:f7ff:b7bd:4842
10:10:59 fe80::a00:f7ff:b7bd:4842
```

#### 4.3.2.2 IPv6 packet bridging - Linux setup

This setup information is for a Linux host PC; the host side implements a standard IPv6-based networking stack.

**Note:** *If you use the same Linux PC for firmware development, note that only the projects provided with the STM32CubeIDE are available for this platform.*

**Step 1.** Install the tunslip6 software module to exchange IPv6 packets over the serial line between the host PC stack and the border router IPv6 stack. To compile it, from the root directory of the X-CUBE-SUBG2 package, run:

```
cd Utilities/PC_Software/Contiki-NG/serial-io
make tunslip6
```

**Step 2.** Assuming that the border router device set up in the previous section is connected to the Linux host, launch the tunslip6 application by providing the parameter for the virtual communication port to which the border router device is connected (located in "/dev/ttyACMx" where "x" is the corresponding port number, here we assume x=0):

```
sudo ./tunslip6 -s /dev/ttyACM0 aaaa::1/64
```

This command creates a new virtual interface in the Linux host, called "tun0".

- Step 3.** Reset the STM32 Nucleo board on which the border router is implemented by pressing the black [RESET] button which triggers the system initialization and the exchange of configuration information with the tunslip6 application. A snapshot of the terminal running the tunslip6 utility is shown below.

**Figure 25. tunslip6 terminal window (snapshot)**

```
marco@stm-devel: ~
File Edit View Search Terminal Help
marco@stm-devel:~$ sudo tunslip6 -B 115200 -s /dev/ttyACM0 aaaa::1/64
*****SLIP started on `/dev/ttyACM0'
opened tun device `/dev/tun0'
ifconfig tun0 inet `hostname` up
ifconfig tun0 add aaaa::1/64
ifconfig tun0 inet 172.16.0.1 pointopoint 172.16.0.2
ifconfig tun0 add fe80::0:0:0:1/64
ifconfig tun0

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
-00

      inet addr:172.16.0.1  P-t-P:172.16.0.2  Mask:255.255.255.255
      inet6 addr: fe80::1/64 Scope:Link
      inet6 addr: aaaa::1/64 Scope:Global
      UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
      RX packets:0 errors:0 dropped:0 overruns:0 frame:0
      TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:500
      RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

[INFO: BR      ] Waiting for prefix
*** Address:aaaa::1 => aaaa:0000:0000:0000
[INFO: BR      ] Waiting for prefix
[INFO: BR      ] Server IPv6 addresses:
[INFO: BR      ] aaaa::351:3433:8835:6636
[INFO: BR      ] fe80::351:3433:8835:6636
Write to tun

```

Public Address of the Border Router to be used for the Web Server for routes and neighbors info

#### 4.3.2.3 IPv6 Host setup troubleshooting

- The tunslip6 application does not work properly with Ubuntu kernel 3.13.0-65-generic (part of the Ubuntu 14.04 LTS)
- The Cygwin library used by the wpcapslip6 tool limits the "/dev/ttyS\*" number to 100. If you have a device with a higher number, either:
  - a. Recompile Cygwin DLL library as suggested at
  - Remove allocated COM ports (you can find some help at <http://superuser.com/questions/408976/howdo-i-clean-up-com-ports-in-use>)
- Disable your firewall
- Ensure IPv6 is enabled:
  - in the properties of the Loopback Adapter
  - by setting (if present) registry key HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\services\TCPIP6\Parameters\DisabledComponents to the value 0x8E
  - You may need to change the default name of the network connection associated with the Microsoft loopback adapter to a name that does not contain spaces
- Under Windows the functionality of wpcapslip6 utility depends on the level of administrative rights of the user

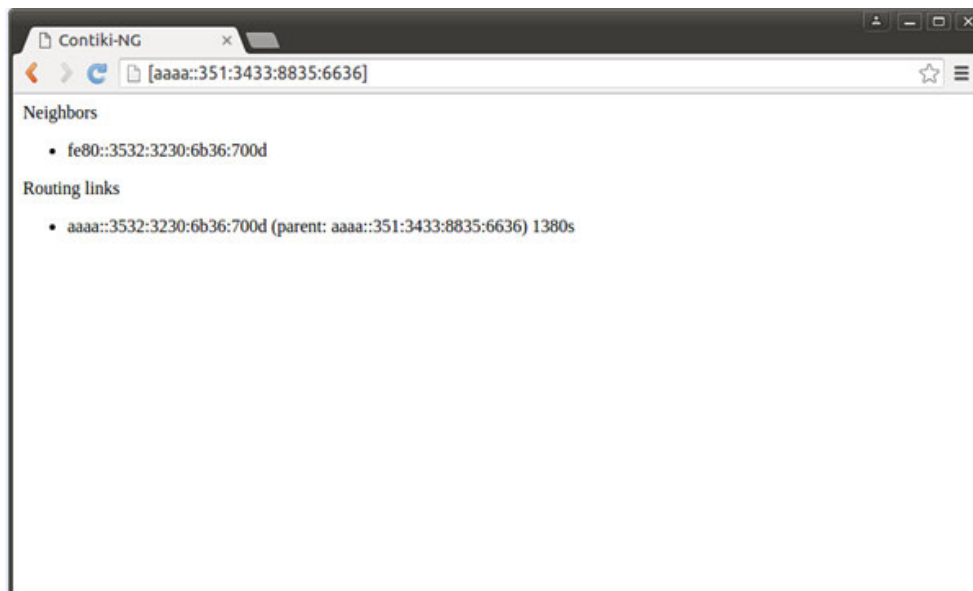
#### 4.3.3 Contiki server access and connectivity test

- Step 1.** Open a web browser to access the Contiki-NG border router web server, which returns the RPL neighbors and routes.

**Step 2.** Access it via the first address listed in the “wpcapslip6” (Windows) or “tunslip6” (Linux) terminal window, after the “Server IPv6 addresses:” line.

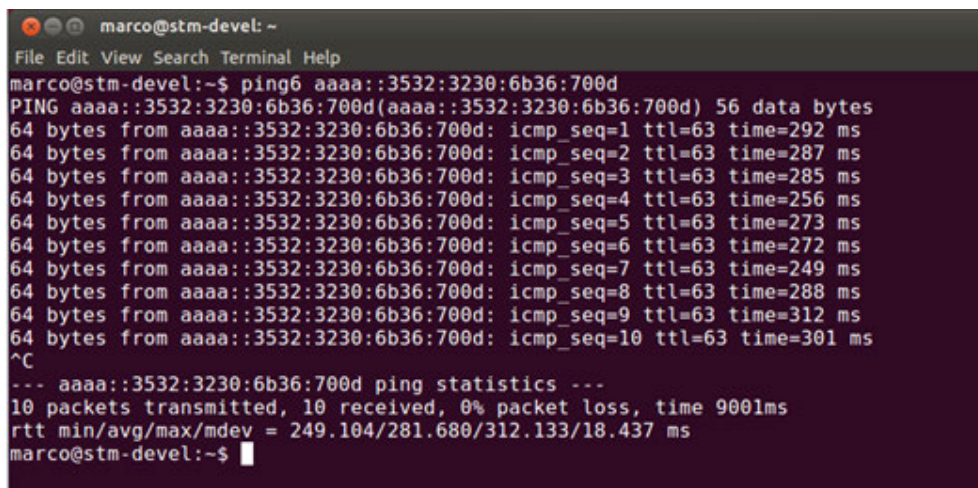
The following figure shows the web browser page after inserting the IPv6 address of the server in the URL (the IPv6 address must be enclosed in square brackets). Assuming that a wireless node is running, the IPv6 address of the wireless node should be in the “Neighbors” list.

**Figure 26. Contiki-NG border router web server**



**Step 3.** Run a simple ping6 command to check the end-to-end IPv6 connectivity between the Linux IPv6 host and the wireless sensor node using the full 128-bit IPv6 address in the “Routes” list. In this specific example, the ping6 command (or “ping -6” in Windows) is `ping6 aaaa::3532:3230:6b36:700d`. The following figure is a snapshot of the ping6 command execution showing the ping from the Linux host replied by the remote wireless sensor node.

**Figure 27. ping6 command window snapshot**



**Step 4.** To perform another test, recompile the UDP Client application by changing the macro

```
#define BORDER_ROUTER_SCENARIO 0
```

to

```
#define BORDER_ROUTER_SCENARIO 1
```

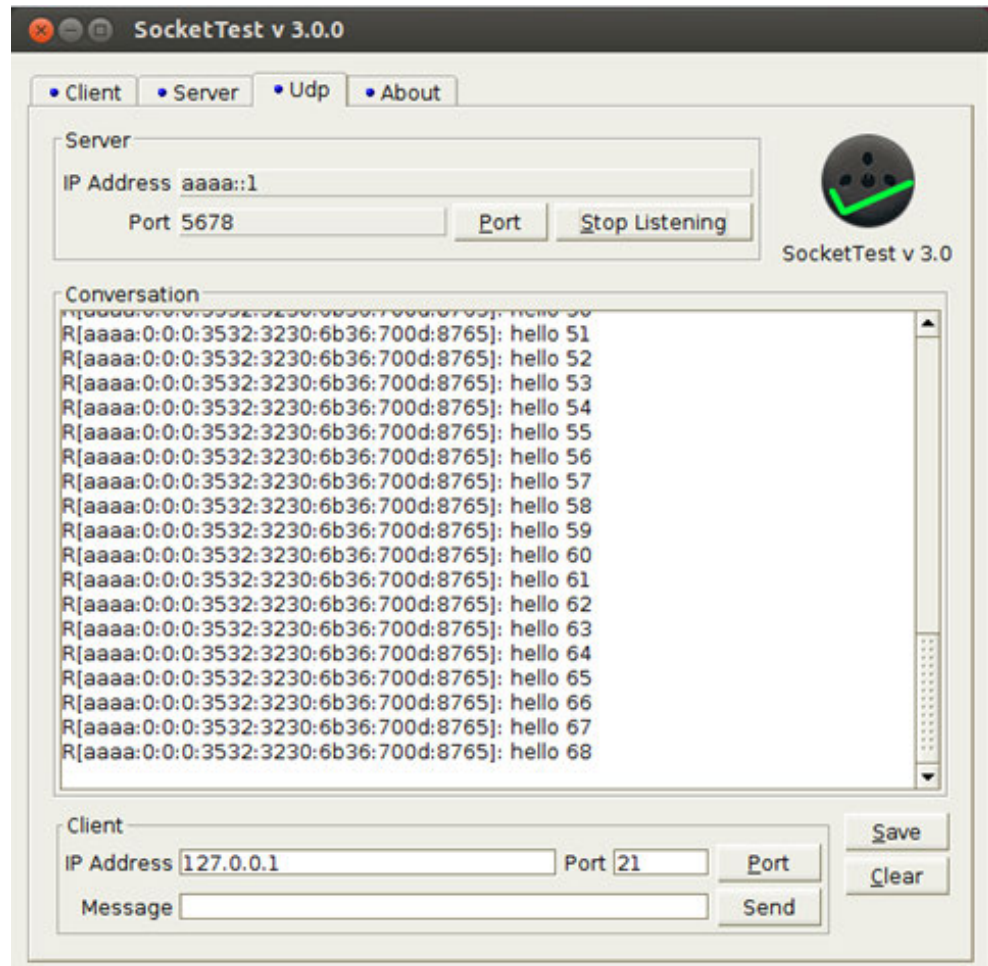
in the udp-client.c file.

By doing this, the UDP Client will not send the packets to the address of the root of the RPL tree, like in the default configuration (this setup is thought to work with the UDP Server firmware), but to another hardcoded address specified in the code, i.e.

```
#define BORDER_ROUTER_SCENARIO_ADDRESS "aaaa::1"
```

that must match the address passed to tunslip6 or wpcapslip6 tools. This address is the one taken by the virtual network interface of the host PC, so, by running any UDP Server application on it and setting it to listen to proper address and port, you will be able to receive the UDP messages sent by the UDP Client node on the host PC (as they are routed from the wireless 6LoWPAN network to the IPv6 protocol stack of the PC by the Border Router firmware) as shown below.

**Figure 28. UDP Server running behind the Border Router**





## 5 Command line interface (CLI) application for S2-LP DK GUI

The CLI application allows connecting an [STM32 Nucleo](#) development board, equipped with an [S2-LP](#)-based expansion board ([X-NUCLEO-S2868A1](#), [X-NUCLEO-S2868A2](#), or [X-NUCLEO-S2915A1](#)), to the GUI available in the [S2-LP DK](#) software. The GUI is available in the GUI subfolder. You just have to run the S2-LP\_DK.exe application.

**Important:** Use the GUI version 1.3.4 or later. Refer to the [S2-LP DK](#) documentation for detailed information on the GUI and on the CLI firmware usage.

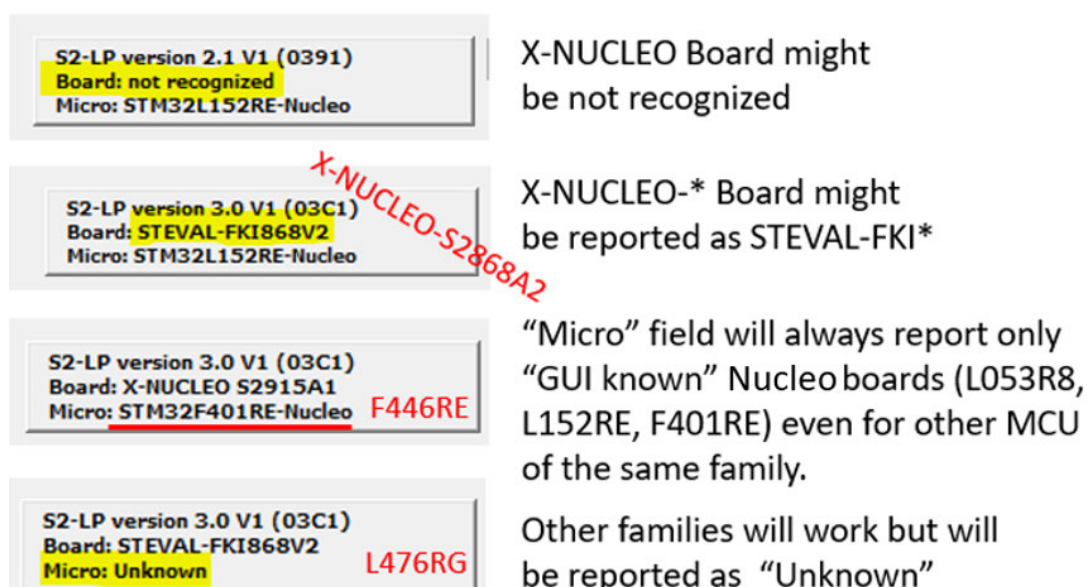
The CLI firmware porting for the [X-CUBE-SUBG2](#) solution grants the compatibility with the [STM32Cube](#) ecosystem and the [STM32CubeMX](#) tool.

**Note:** Only the [X-NUCLEO-S2868A1](#), [X-NUCLEO-S2868A2](#), or [X-NUCLEO-S2915A1](#) are supported, regardless of what is detailed in the GUI. This information depends on the content of the expansion board EEPROM. Originally, the CLI application has been developed for [NUCLEO-L152RE](#) and [NUCLEO-L053R8](#) only. Thanks to the [STM32CubeMX](#) integration, you can port the application to other MCUs. A sample porting is available in the examples for the [NUCLEO-F401RE](#). Due to the original development, also the MCU-related information might be not accurate on the GUI:

- “STM32L152RE-Nucleo” and “STM32L053R8-Nucleo” are reported for any L1 or L0 board
- “STM32F401RE-Nucleo” is reported for any F4 board (F401 in “unofficially” known to the GUI)
- “Unknown” is reported as the microcontroller information for other families

The following figure summarizes the above-mentioned limitations/warnings.

**Figure 29. S2-LP GUI limitations**



The [S2-LP](#) GUI allows the user to configure the radio with different settings. It also allows exporting the register configuration to a C file.

Figure 30. S2-LP DK GUI: main view

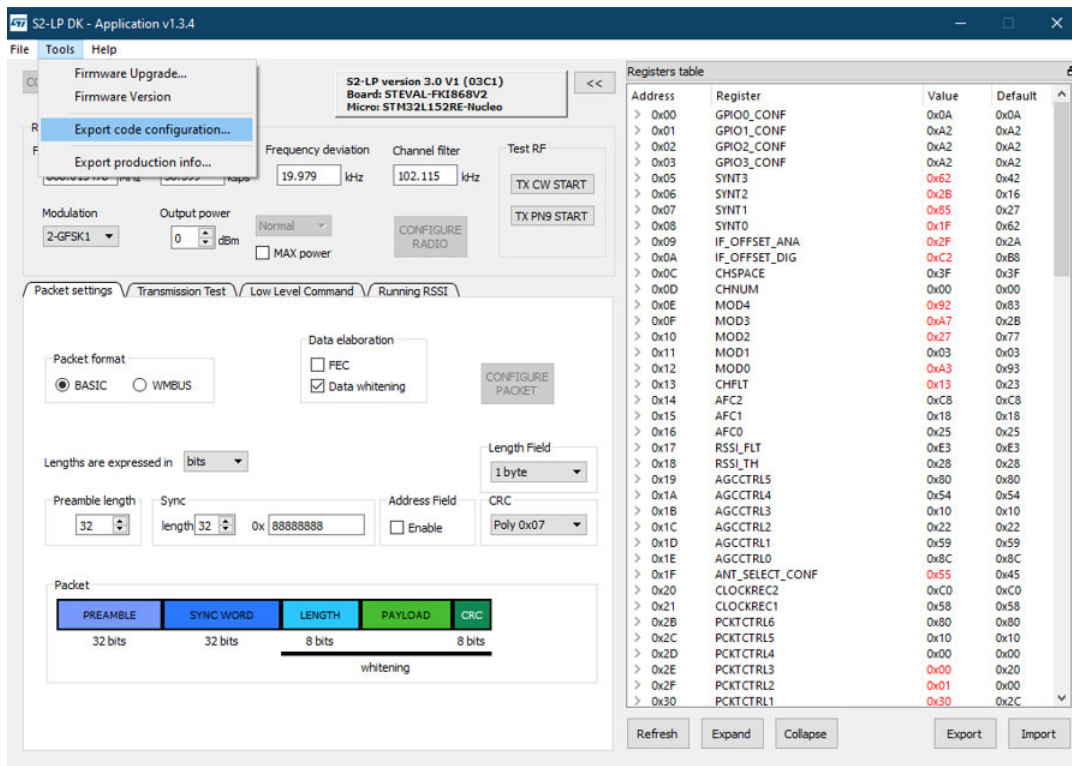
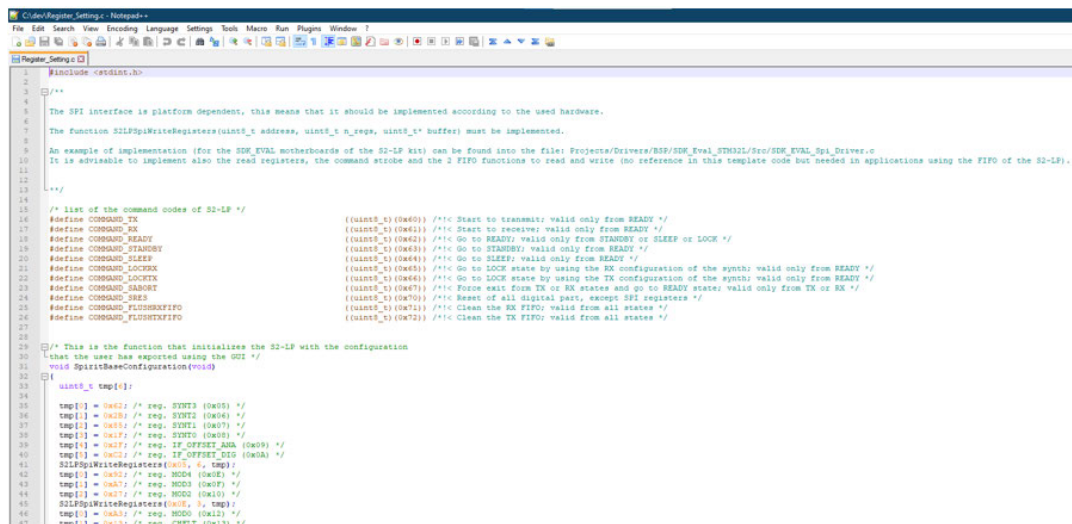


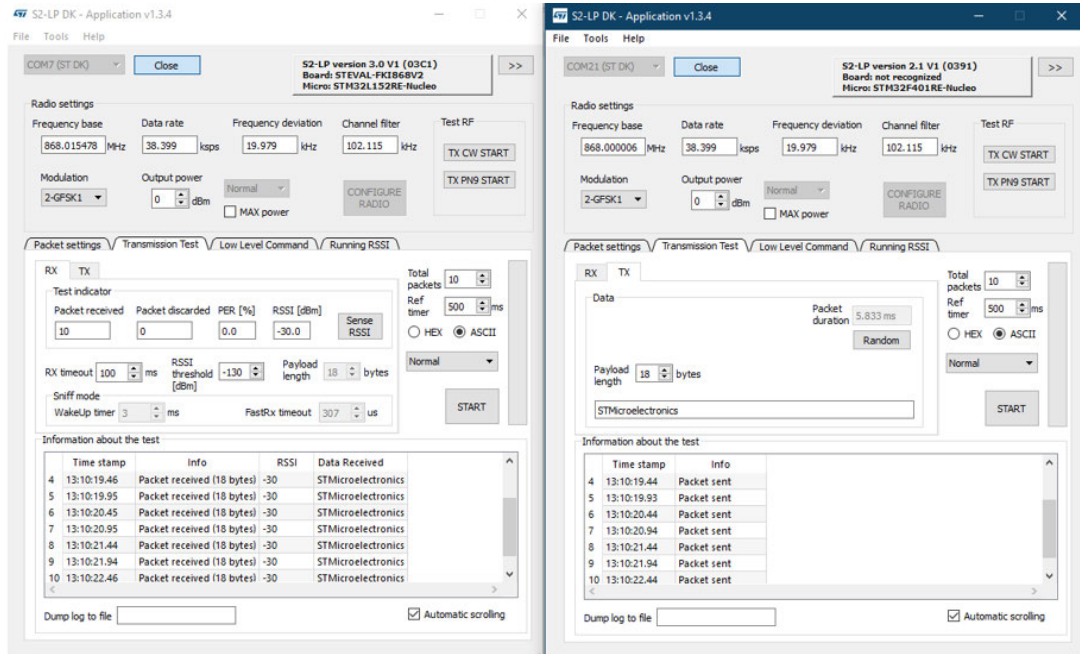
Figure 31. Source file with the register settings exported by the S2-LP DK GUI



As an example, using two different boards connected to two instances of the GUI, it is possible to perform:

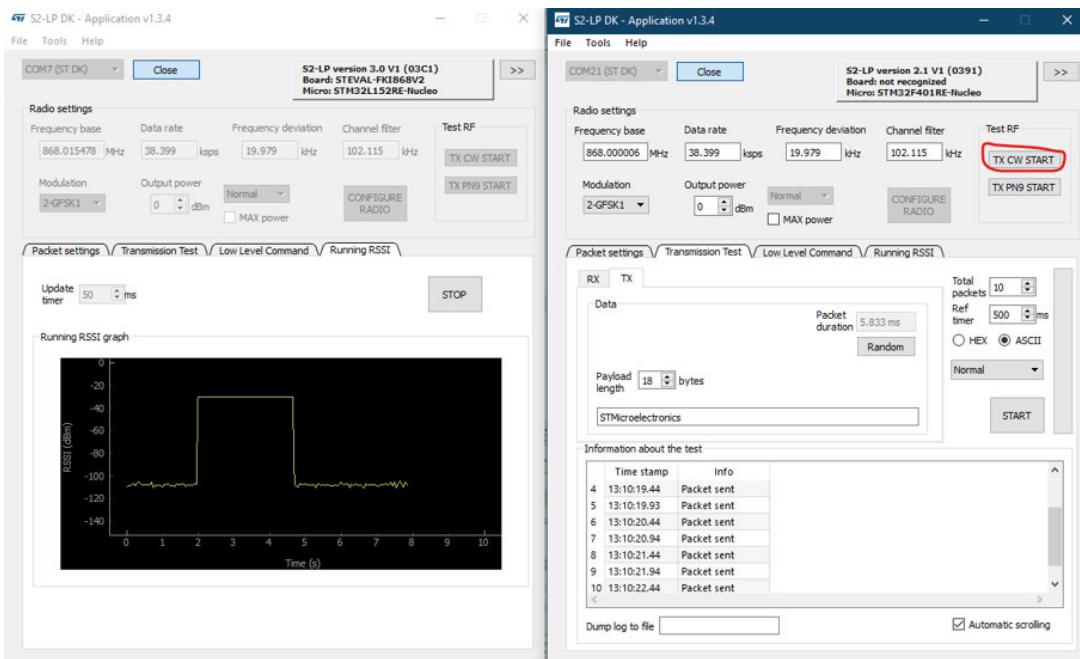
- transmit or receive tests, with a customizable timing and report of the packet loss ratio

Figure 32. S2-LP DK GUI transmit/receive tests



- measure the RSSI on a board, making the other one send data or generate an unmodulated signal (for example, the [TX CW START/STOP] button)

Figure 33. S2-LP DK GUI RSSI test



## 6 FIFO TX/RX applications

The FIFO TX and FIFO RX applications have been ported from the set of examples delivered in [S2-LP DK](#) (SDK\_FifoHandler\_A and SDK\_FifoHandler\_B, respectively) to demonstrate the support for multiple [S2-LP](#) GPIOs, which is a new feature of the [X-CUBE-SUBG2](#) (from version 5.0.0).

The preconfigured examples are provided for a custom board (that is, when configuring the project through [STM32CubeMX](#), you are not obliged to use a prebuilt expansion board), instead of a standard BSP ([X-NUCLEO-S2868A1](#), [X-NUCLEO-S2868A2](#), or [X-NUCLEO-S2915A1](#)). However, these examples are actually compatible with the [X-NUCLEO-S2868A2](#) (and [X-NUCLEO-S2868A1](#)) GPIO settings, without supporting the LED and EEPROM.

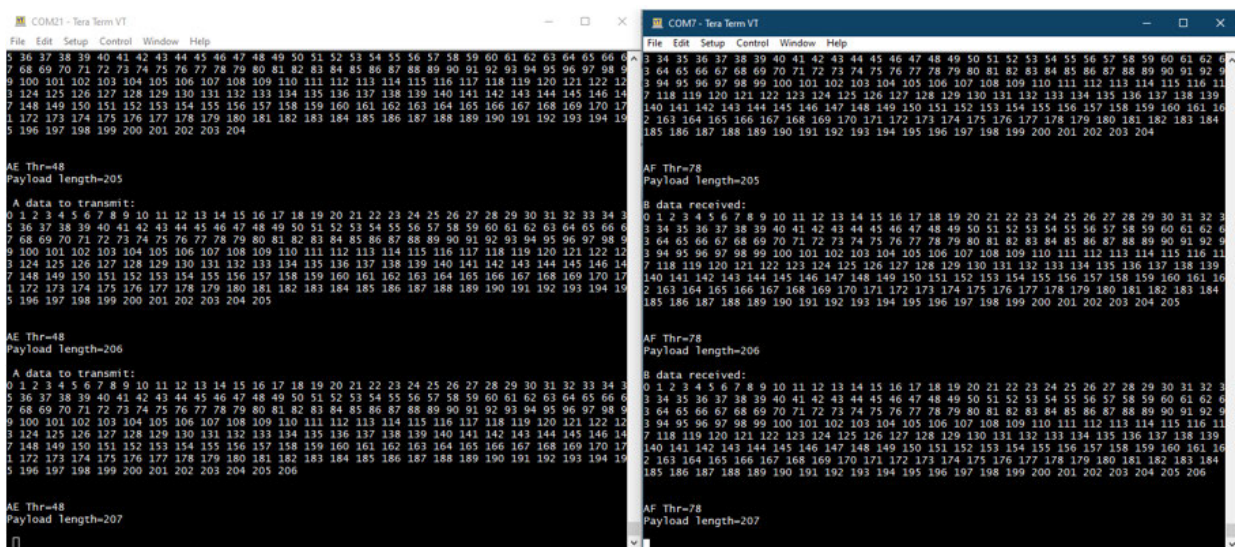
The two applications also demonstrate how the [S2-LP](#) radio can send packets longer than the 128 basic payload and FIFO sizes, using the `EXTENDED_LENGTH_FIELD` field of the basic packet format and the:

- almost empty IRQ and threshold for the transmitter FIFO
- almost full IRQ and threshold for the receiver FIFO

You can program one board as the transmitter (FIFO TX) and another board as the receiver (FIFO RX), power them up, and connect them to a terminal application like Teraterm.

The following figure shows how the transmitted packet assumes the progressively increasing sizes (up to 300 bytes), going over the 128 bytes that are the raw radio payload (and also the dimension of the two RX and TX FIFOs).

Figure 34. FIFO Tx (left) and FIFO Rx (right) application terminals



On the receiver side, the two interrupts (“main S2-LP” and “Rx FIFO almost full”) are mapped on two different GPIOs configured in the EXTI mode, as shown in the following figure. `S2LP_GPIO_3` is for the main [S2-LP](#) interrupt, while `S2LP_GPIO_1` is for the almost full interrupt, which is configured with a higher preemption priority (that is, the lowest number) compared to the former.



**Figure 35. FIFO RX application using two different GPIOs in the EXTI mode**

```

app_custom_s2lp_fifo_rx.c
80 /* max payload size */
81 #define PAYLOAD_LENGTH_FIX 300
82
83 #undef EXTENDED_LENGTH_FIELD
84 #define EXTENDED_LENGTH_FIELD_S_ENABLE
85
115 SGPIOInit xSpiIRQ=
116 S2LP_GPIO_3,
117 S2LP_GPIO_MODE_DIGITAL_OUTPUT_LP,
118 S2LP_GPIO_DIG_OUT_IRQ
119 };
120
121 /**
122  * @brief GPIO RX FIFO AF structure fitting
123  */
124 SGPIOInit xSpiAF=
125 S2LP_GPIO_1,
126 S2LP_GPIO_MODE_DIGITAL_OUTPUT_LP,
127 S2LP_GPIO_DIG_OUT_TXRX_FIFO_ALMOST_FULL
128 };
129
224 void FifoRx_S2LP_Callback_GPIO_1(void)
225 {
226     uint8_t cRxDataLen;
227
228     cRxDataLen = S2LP_FIFO_ReadNumberBytesRxFifo();
229
230     if(cRxDataLen>2)
231     {
232         /* Read the RX FIFO */
233         S2LP_ReadFIFO(cRxDataLen-2, &vectcRxBuff[nRxIndex]);
234         nRxIndex+=cRxDataLen-2;
235     }
236 }
237
238 void FifoRx_S2LP_Callback_GPIO_3(void)
239 {
240     uint8_t cRxDataLen;
241
242     /* Get the IRQ status */
243     S2LP_GPIO_IrqGetStatus(&xIrqStatus);
244
245     /* Check the RX_DATA_READY IRQ flag */
246     if(xIrqStatus.IRQ_RX_DATA_READY)
247     {
248         cRxDataLen = S2LP_FIFO_ReadNumberBytesRxFifo();
249
250         if(cRxDataLen!=0)
251         {
252             /* Read the RX FIFO */
253             S2LP_ReadFIFO(cRxDataLen, &vectcRxBuff[nRxIndex]);
254         }
255
256         S2LP_CMD_StrobeFlushRxFifo();
257
258         /* update the number of received bytes */
259         nRxIndex += cRxDataLen;
260
261         /* set the Rx done flag */
262         xRxDoneFlag = SET;
263     }
264
265     /* Check the SPIRIT RX_DATA_DISC IRQ flag */
266     if(xIrqStatus.IRQ_RX_DATA_DISC)
267     {
268         /* RX Command - to ensure the device will be ready for the next reception */
269         S2LP_CMD_StrobeRx();
270     }
271 }
  
```

## 7 System setup

### 7.1 Hardware description

#### 7.1.1 STM32 Nucleo

STM32 Nucleo development boards provide an affordable and flexible way for users to test solutions and build prototypes with any STM32 microcontroller line.

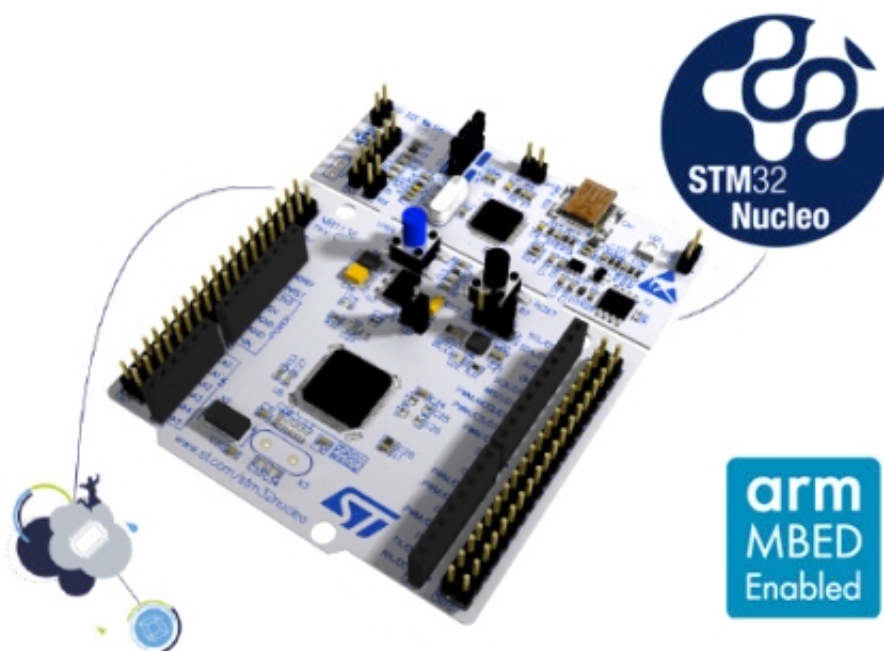
The Arduino connectivity support and ST morpho connectors make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide range of specialized expansion boards to choose from.

The STM32 Nucleo board does not require separate probes as it integrates the ST-LINK/V2-1 debugger/programmer.

The STM32 Nucleo board comes with the comprehensive STM32 software HAL library together with various packaged software examples for different IDEs (IAR EWARM, Keil MDK-ARM, STM32CubeIDE, mbed and GCC/LLVM).

All STM32 Nucleo users have free access to the mbed online resources (compiler, C/C++ SDK and developer community) at [www.mbed.org](http://www.mbed.org) to easily build complete applications.

Figure 36. STM32 Nucleo board



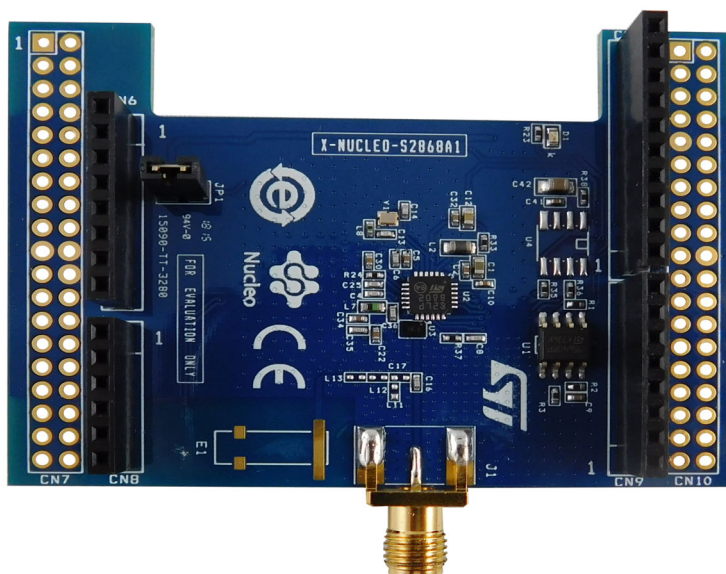
#### 7.1.2 X-NUCLEO-S2868A1 expansion board

The X-NUCLEO-S2868A1 expansion board is based on the S2-LP radio and operates in the 868 MHz ISM frequency band.

The expansion board is compatible with ST morpho and Arduino UNO R3 connectors.

The X-NUCLEO-S2868A1 interfaces with the STM32 Nucleo microcontroller via SPI connections and GPIO pins. You can change some of the GPIOs by mounting or removing the resistors.

Figure 37. X-NUCLEO-S2868A1 expansion board



### 7.1.3 X-NUCLEO-S2868A2 and X-NUCLEO-S2915A1 expansion boards

The [X-NUCLEO-S2868A2](#) expansion board is based on the [S2-LP](#) radio and operates in the 868 MHz ISM frequency band.

The [X-NUCLEO-S2915A1](#) expansion board is based on the [S2-LP](#) radio and operates in the 915 MHz ISM frequency band.

The expansion boards are compatible with ST morpho and Arduino UNO R3 connectors, and interface with the [STM32 Nucleo](#) microcontroller via SPI connections and GPIO pins.

You can change some of the GPIOs by mounting or removing the resistors.

Figure 38. X-NUCLEO-S2868A2 expansion board

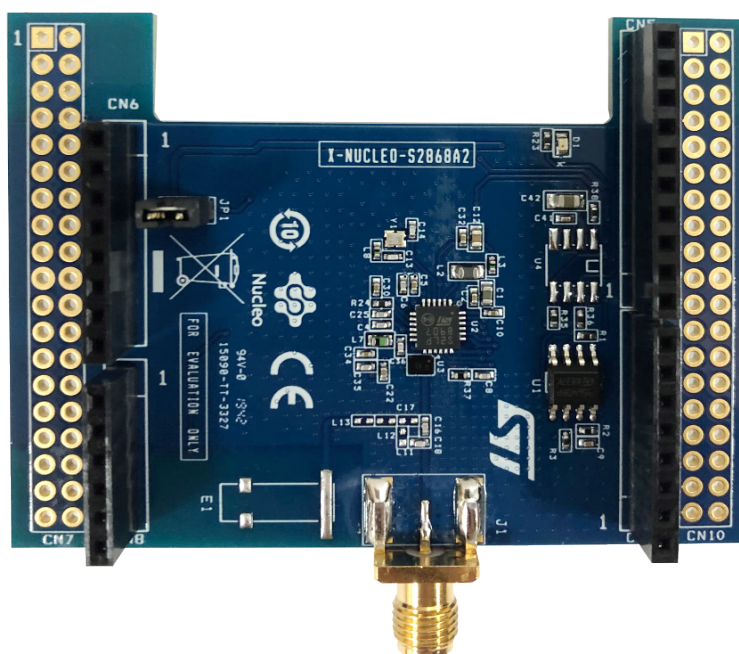
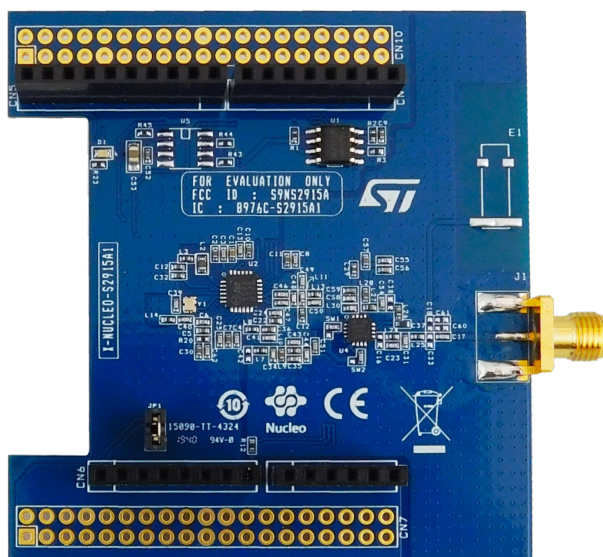


Figure 39. X-NUCLEO-S2915A1 expansion board



## 7.2 Software description

To use STM32 Nucleo development boards with the X-NUCLEO-S2868A1, X-NUCLEO-S2868A2, X-NUCLEO-S2915A1 expansion boards, the following software specification are required:

- X-CUBE-SUBG2 expansion for STM32Cube. The X-CUBE-SUBG2 firmware and related documentation is available on [www.st.com](http://www.st.com).
- Development tool-chain and Compiler supported by the STM32Cube expansion software:
  - IAR Embedded Workbench for ARM® toolchain + ST-LINK
  - RealView Microcontroller Development Kit (MDK-ARM-STR) toolchain + ST-LINK
  - STM32CubeIDE + ST-LINK

## 7.3 Hardware setup

The following hardware components are required:

- an STM32 Nucleo development board (order code: NUCLEO-F401RE, NUCLEO-L152RE or NUCLEO-L053R8)
- an S2-LP expansion board (order code: X-NUCLEO-S2868A1, X-NUCLEO-S2868A2 or X-NUCLEO-S2915A1)
- a USB type A to Mini-B USB cable to connect the STM32 Nucleo to the PC

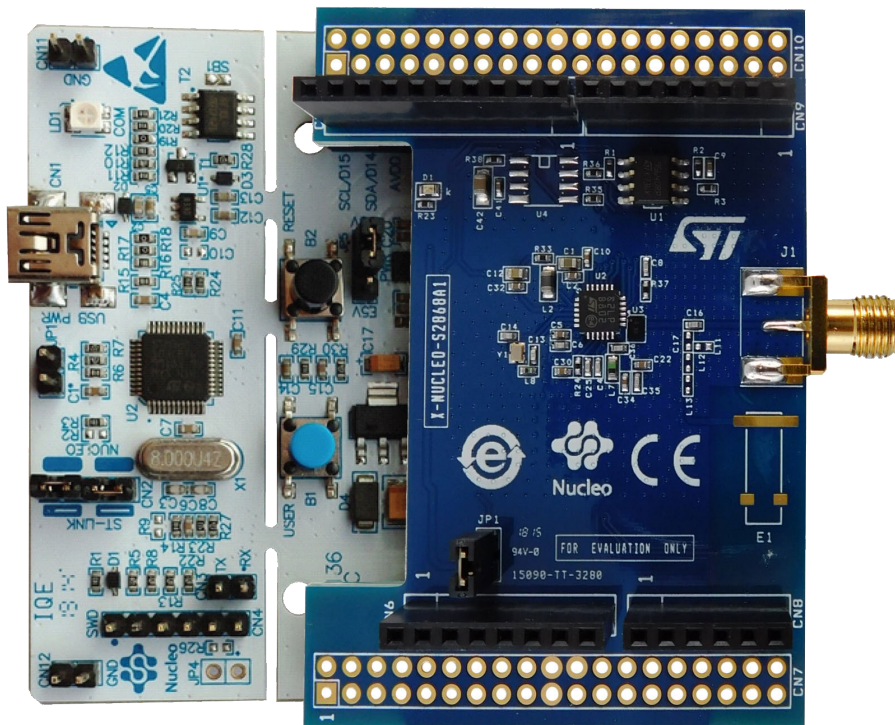
## 7.4 Board setup

**Step 1.** Check that the jumper on J1 is connected to provide the required voltage to the board devices



- Step 2.** Connect the X-NUCLEO-S2868A1, X-NUCLEO-S2868A2 or X-NUCLEO-S2915A1 to the STM32 Nucleo

**Figure 40.** X-NUCLEO-S2868A1 expansion board plugged onto an STM32 Nucleo board



- Step 3.** Power the STM32 Nucleo board using the Mini-B USB cable
- Step 4.** Program the firmware in the STM32 on the Nucleo development board using the firmware sample provided
- Step 5.** Press the reset button on the STM32 Nucleo board  
The demonstration kit is ready-to-use

---

## 8 References

---

Freely available on [www.st.com](http://www.st.com):

1. [STM32 Nucleo](#) development board datasheet
2. UM2405: Getting started with the X-NUCLEO-S2868A1 Sub-1 GHz 868 MHz RF expansion board based on S2-LP radio for STM32 Nucleo
3. UM2638: Getting started with the X-NUCLEO-S2868A2 Sub-1 GHz 868 MHz RF expansion board based on S2-LP radio for STM32 Nucleo
4. UM2641: Getting started with the X-NUCLEO-S2915A1 Sub-1 GHz 915 MHz RF expansion board based on S2-LP radio for STM32 Nucleo
5. [S2-LP](#) datasheet
6. [STSW-S2LP-DK](#): Evaluation SW package based on S2-LP

## Revision history

**Table 6. Document revision history**

Date	Revision	Changes
12-Dec-2019	1	Initial release.
14-Feb-2020	2	Updated Figure 1. X-CUBE-SUBG2 software architecture and Section 4 References. Added Section 3.6.3 X-NUCLEO-S2868A2 and X-NUCLEO-S2915A1 expansion boards. Added X-NUCLEO-S2868A2 and X-NUCLEO-S2915A1 expansion board compatibility information.
02-Dec-2020	3	Updated Section 1 Acronyms and abbreviations, Section 2.1 Overview, Section 2.2 Architecture and Section 2.3 Folder structure. Added Section 4 6LoWPAN applications, Section 4.1 Contiki-NG, Section 4.1.1 UDP Client and UDP Server sample application overview, Section 4.2 Serial Sniffer sample application overview, Section 4.2.1 How to use the Serial Sniffer application under Linux, Section 4.2.2 How to use the Serial Sniffer application under Windows, Section 4.2.3 Important additional information, Section 4.3 Border Router sample application overview, Section 4.3.1 System overview, Section 4.3.2 IPv6 host setup, Section 4.3.2.1 IPv6 packet bridging - Windows setup, Section 4.3.2.2 IPv6 packet bridging - Linux setup, Section 4.3.2.3 IPv6 Host setup troubleshooting and Section 4.3.3 Contiki server access and connectivity test.
10-May-2022	4	Updated <a href="#">Section 1 Acronyms and abbreviations</a> , <a href="#">Section 2.1 Overview</a> , <a href="#">Section 2.2 Architecture</a> , <a href="#">Section 2.3 Folder structure</a> , and <a href="#">Section 8 References</a> . Added <a href="#">Section 5 Command line interface (CLI) application for S2-LP DK GUI</a> and <a href="#">Section 6 FIFO TX/RX applications</a> .

## Contents

<b>1</b>	<b>Acronyms and abbreviations</b>	<b>2</b>
<b>2</b>	<b>X-CUBE-SUBG2 software expansion for STM32Cube</b>	<b>3</b>
2.1	Overview	3
2.2	Architecture	3
2.3	Folder structure	5
2.4	APIs	5
<b>3</b>	<b>Point-to-Point (P2P) demo firmware description</b>	<b>6</b>
3.1	P2P application details	6
3.2	Application state diagram	6
3.3	S2-LP packet handler overview	8
3.4	Transmit and receive (command and response) packet structure	9
3.4.1	Packet field description	10
3.5	User configuration	10
3.5.1	Selecting packet handler	10
3.5.2	Setting low-power mode	10
3.5.3	Setting radio configuration parameters	11
3.5.4	Setting packet configuration parameters	11
3.5.5	Setting node address	11
3.5.6	User defined commands and macros	12
<b>4</b>	<b>6LoWPAN applications</b>	<b>13</b>
4.1	Contiki-NG	13
4.1.1	UDP Client and UDP Server sample application overview	14
4.2	Serial Sniffer sample application overview	17
4.2.1	How to use the Serial Sniffer application under Linux	17
4.2.2	How to use the Serial Sniffer application under Windows	17
4.2.3	Important additional information	18
4.3	Border Router sample application overview	21
4.3.1	System overview	21
4.3.2	IPv6 host setup	22
4.3.3	Contiki server access and connectivity test	26
<b>5</b>	<b>Command line interface (CLI) application for S2-LP DK GUI</b>	<b>29</b>
<b>6</b>	<b>FIFO TX/RX applications</b>	<b>32</b>
<b>7</b>	<b>System setup</b>	<b>34</b>
7.1	Hardware description	34
7.1.1	STM32 Nucleo	34



---

7.1.2	X-NUCLEO-S2868A1 expansion board. ....	34
7.1.3	X-NUCLEO-S2868A2 and X-NUCLEO-S2915A1 expansion boards. ....	35
7.2	Software description. ....	36
7.3	Hardware setup. ....	36
7.4	Board setup. ....	36
<b>8</b>	<b>References. ....</b>	<b>38</b>
	<b>Revision history. ....</b>	<b>39</b>
	<b>List of figures. ....</b>	<b>42</b>
	<b>List of tables. ....</b>	<b>43</b>

## List of figures

<b>Figure 1.</b>	X-CUBE-SUBG2 software architecture . . . . .	4
<b>Figure 2.</b>	X-CUBE-SUBG2 package folder structure . . . . .	5
<b>Figure 3.</b>	X-NUCLEO-S2868A1 plus STM32 Nucleo used as a node (transmitter/receiver) in P2P communication . . . . .	6
<b>Figure 4.</b>	Application state diagram when Node 1 user button is pressed . . . . .	7
<b>Figure 5.</b>	Application state diagram when Node 2 user button is pressed . . . . .	7
<b>Figure 6.</b>	Application state diagram (low-power mode): data communication transmit and receive states . . . . .	8
<b>Figure 7.</b>	Command data packet structure . . . . .	9
<b>Figure 8.</b>	Response packet structure . . . . .	10
<b>Figure 9.</b>	System configuration and parameters print at boot time (example) . . . . .	13
<b>Figure 10.</b>	UDP Client and UDP Server node communication with a PC . . . . .	14
<b>Figure 11.</b>	UDP Server console output . . . . .	15
<b>Figure 12.</b>	UDP Client console output . . . . .	15
<b>Figure 13.</b>	List of commands available in the Serial Shell . . . . .	16
<b>Figure 14.</b>	Shell before and after log all 0 command . . . . .	16
<b>Figure 15.</b>	UDP Client (left) and UDP Server (right) console output after log all 0 command . . . . .	17
<b>Figure 16.</b>	Serial Sniffer output in Tera Term . . . . .	18
<b>Figure 17.</b>	Cygwin shell: command line to interact with Serial Sniffer . . . . .	19
<b>Figure 18.</b>	Message exchange flow between UDP Server and UDP Client (capture of Link Layer encrypted packets) . . . . .	20
<b>Figure 19.</b>	Message exchange flow between UDP Server and UDP Client (capture of Link Layer plain text packets) . . . . .	21
<b>Figure 20.</b>	System architecture for Border Router application . . . . .	22
<b>Figure 21.</b>	Adding the network adapter in Windows . . . . .	23
<b>Figure 22.</b>	STM32 Nucleo board virtual COM Port value . . . . .	24
<b>Figure 23.</b>	Reading the MAC address of the Microsoft KM-TEST loopback adapter . . . . .	24
<b>Figure 24.</b>	wpcapslip6 utility terminal window (snapshot) . . . . .	25
<b>Figure 25.</b>	tunslip6 terminal window (snapshot) . . . . .	26
<b>Figure 26.</b>	Contiki-NG border router web server . . . . .	27
<b>Figure 27.</b>	ping6 command window snapshot . . . . .	27
<b>Figure 28.</b>	UDP Server running behind the Border Router . . . . .	28
<b>Figure 29.</b>	S2-LP GUI limitations . . . . .	29
<b>Figure 30.</b>	S2-LP DK GUI: main view . . . . .	30
<b>Figure 31.</b>	Source file with the register settings exported by the S2-LP DK GUI . . . . .	30
<b>Figure 32.</b>	S2-LP DK GUI transmit/receive tests . . . . .	31
<b>Figure 33.</b>	S2-LP DK GUI RSSI test . . . . .	31
<b>Figure 34.</b>	FIFO Tx (left) and FIFO Rx (right) application terminals . . . . .	32
<b>Figure 35.</b>	FIFO RX application using two different GPIOs in the EXTI mode . . . . .	33
<b>Figure 36.</b>	STM32 Nucleo board . . . . .	34
<b>Figure 37.</b>	X-NUCLEO-S2868A1 expansion board . . . . .	35
<b>Figure 38.</b>	X-NUCLEO-S2868A2 expansion board . . . . .	35
<b>Figure 39.</b>	X-NUCLEO-S2915A1 expansion board . . . . .	36
<b>Figure 40.</b>	X-NUCLEO-S2868A1 expansion board plugged onto an STM32 Nucleo board . . . . .	37

## List of tables

Table 1.	List of acronyms . . . . .	2
Table 2.	Stack . . . . .	8
Table 3.	wM-Bus . . . . .	8
Table 4.	Basic . . . . .	9
Table 5.	Packet handler feature comparison . . . . .	9
Table 6.	Document revision history . . . . .	39

**IMPORTANT NOTICE – READ CAREFULLY**

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgment.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to [www.st.com/trademarks](http://www.st.com/trademarks). All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2022 STMicroelectronics – All rights reserved