# Getting started with Contiki6LP, Contiki OS and 6LoWPAN sub-1 GHz RF in X-CUBE-SUBG1 expansion for STM32Cube

## Introduction

Contiki6LP is a middleware library part of X-CUBE-SUBG1. This library provides an implementation of the Contiki Operating System and Contiki 6LoWPAN stack 3.x running on the STM32 with a SPIRIT1 or an S2-LP-based sub-1 GHz RF communication board.

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

The software comes with sample implementations for the X-NUCLEO-IDS01A4, X-NUCLEO-IDS01A5 or X-NUCLEO-S2868A1 expansion board connected to a NUCLEO-L152RE or NUCLEO-F401RE board.

# 1 Acronyms and abbreviations

**Table 1. List of acronyms**

| Acronym | Description |
|---|---|
| UDP | User datagram protocol |
| 6LoWPAN | IPv6 over low power wireless personal area networks |
| RPL | Routing protocol for low power and lossy networks |
| MCU | Microcontroller unit |
| RF | Radio frequency |
| OS | Operating system |

# 2 Contiki6LP middleware library for STM32Cube

## 2.1 Overview

Contiki6LP is a middleware library ready to be integrated in projects based on STM32Cube and the X-CUBE-SUBG1 expansion software.

The key features of the Contiki6LP library are:

- Support for Contiki OS and Contiki 6LoWPAN protocol stack 3.x
- Support for mesh networking technology via the standard RPL protocol
- Sample applications including UPD sender and receiver, and border router

This library provides a Contiki operating system port for the STM32 platform in the STM32Cube software environment, enabling SPIRIT1 and S2-LP sub-1 GHz radio transceiver support.

It also includes three sample applications that you can use to start experimenting with the code. Two of these applications should be used together since they implement communication between two wireless nodes. A sender node periodically sends UDP packets to a receiver node looking out for a specific service number on the same 6LoWPAN network. A third sample application implements the functionality of a 6LoWPAN border (or edge) router.

Technical details regarding the Contiki OS are available at https://github.com/contiki-os/contiki/wiki.

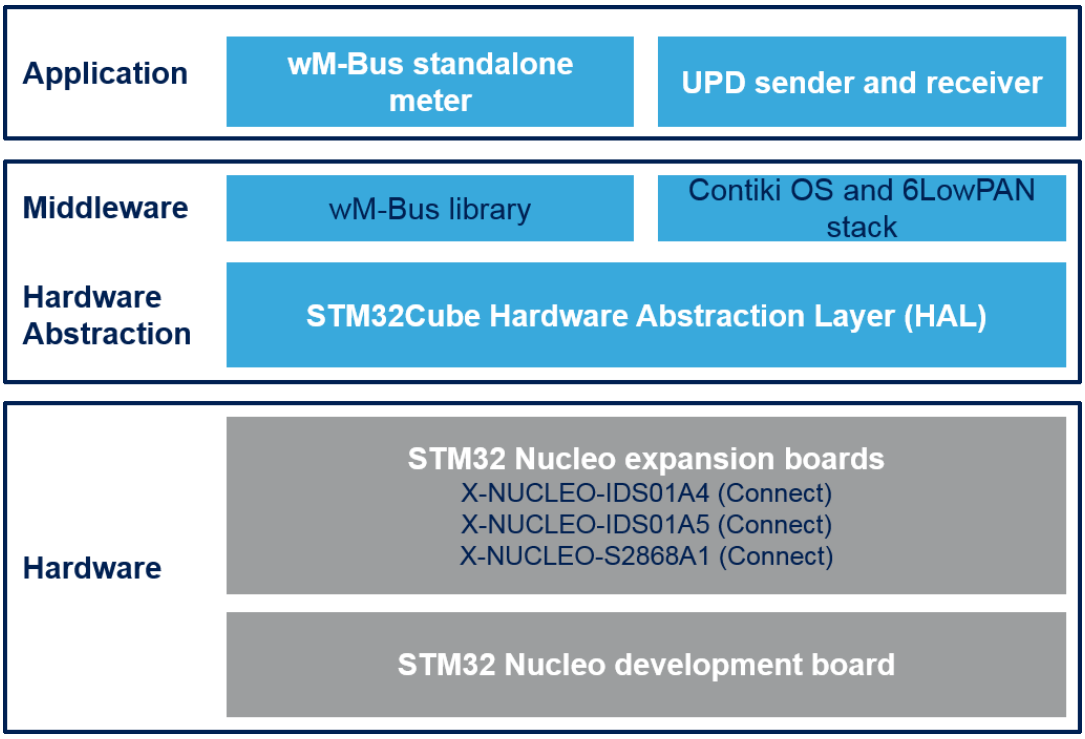
## 2.2 Architecture

This software is based on the STM32Cube and X-CUBE-SUBG1. STM32CubeHAL is the hardware abstraction layer for the STM32 microcontroller, while X-CUBE-SUBG1 extends STM32Cube by providing a board support package (BSP) for the SPIRIT1 and S2-LP sub-1 GHz RF communication expansion boards.

The X-CUBE-SUBG1 package, from version 2.0.0 onward, includes the Contiki6LP middleware library that provides an implementation of the Contiki OS and 6LoWPAN protocol stack version 3.x and uses the SPIRIT1 or S2-LP transceiver for communications.

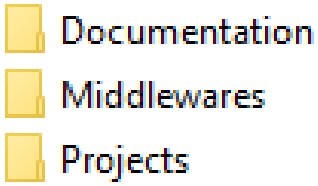
The package also includes some sample applications that the developer can use to start experimenting with the code. These applications allow sending and receiving UPD packets to/from another node in the 6LoWPAN network. A sample application that can be used to create a 6LoWPAN border (edge) router is also provided.

Figure 1. **X-CUBE-SUBG1 software architecture**




## 2.3 Folder structure

Figure 2. **Contiki6LP library related folders**

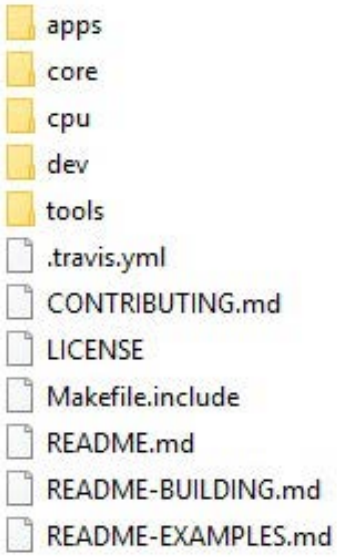



The following folders related to Contiki6LP are included in the X-CUBE-SUBG1 software package:

- **Documentation**: contains a compiled HTML file detailing the software components and APIs.
- **Middlewares**: this folder contains the full porting of the Contiki operating system for the STM32 platform. There are two sub-folders:
  - "ST" contains STM32-specific files to configure both the Contiki OS and the SPIRIT1 and S2-LP radio
  - "Third_Party" includes the Contiki source files
- **Projects**: contains sample applications for the NUCLEO-L152RE and NUCLEO-F401RE platforms in IAR Embedded Workbench for ARM, RealView Microcontroller Development Kit (MDK-ARM-STM32), and System Workbench for STM32 (SW4STM32) development environments.

*Note: Several folders from the official Contiki repository are omitted as they are not required for the STM32 platform.*

Figure 3. **/Middlewares/Third_Party/Contiki folder structure**




The files for the porting layer on STM32 can be found in the /Middlewares/ST/STM32_Contiki_Library directory.

## 2.4 APIs

Full API function and parameter descriptions can be found in a compiled HTML file in the “Documentation” folder.

## 2.5 Sample application description

Several sample applications involving the Contiki OS and the SPIRIT1 or S2-LP transceiver with either NUCLEO-F401RE or NUCLEO-L152RE boards are provided in the “Projects” directory.

### 2.5.1 UDP-sender and UDP-receiver

These applications set up a simple communication scenario between two nodes, using the UDP protocol. One node acts as sender of UDP datagrams and one node acts as a receiver of UDP datagrams.

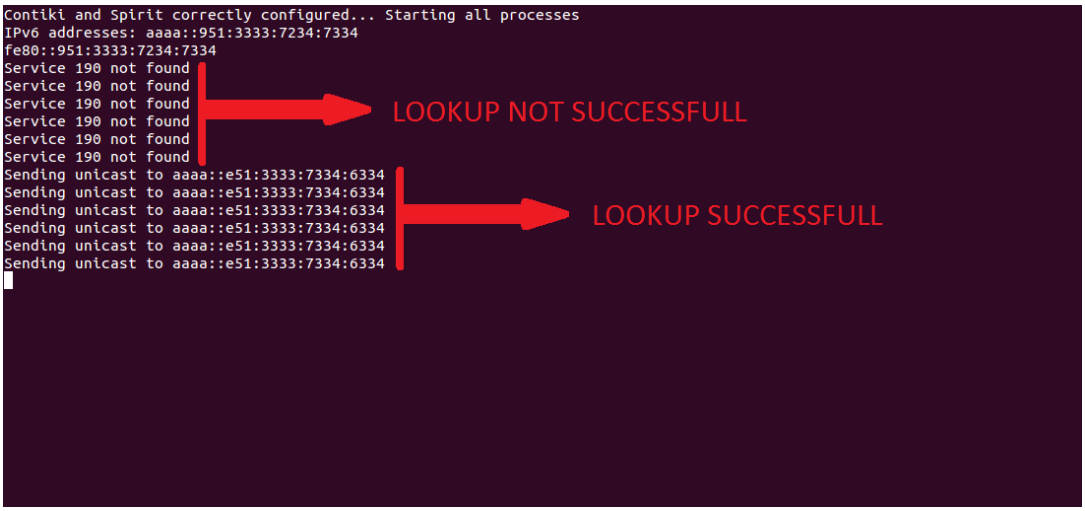
Communication between the nodes is set up through the definition of a `SERVICE_NUMBER`. The sender periodically looks for a node with the same `SERVICE_NUMBER` and starts sending messages if it finds one; the receiver node prints the message content to the serial line monitor. Actual communication is carried out by the SPIRIT1 or S2-LP radio, which can either be used at 868MHz (X-NUCLEO-IDS01A4 expansion board) or at 915MHz (X-NUCLEO-IDS01A5 expansion board).

A host PC running Linux can use the “minicom” application to show the results on a serial line monitor thus (assuming ttyACM0 is the device used by the board):

1. `minicom -b 115200 -D /dev/ttyACM0`
2. Press **CTRL-A-Z** and then **U** to allow a proper carriage return to be printed.
3. Press **CTRL-A-Z** and then **O** to open “minicom” configuration
   a. Select “Serial port setup” and press **Enter**
   b. Press **E**, then **Q** and then **Enter** (this configures the terminal as 8N1)
   c. Select “Exit” and press **Enter**

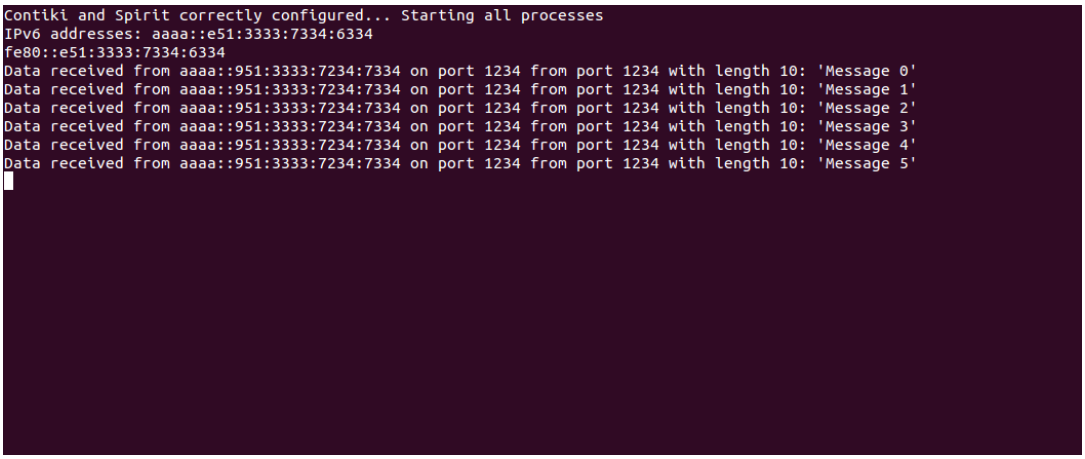
The following figure shows an example of data printed by the “UDP-sender” node.

Figure 4. **Snapshot of terminal utility running UDP-sender example**




The next figure shows an example of the unicast packets from the sender node printed by the “UDP-receiver” node.

Figure 5. **Snapshot of terminal utility running UDP-receiver example**




If the host PC is running Windows, a terminal utility such as Tera Term can be used. The parameters to configure the utility are: 115200 bps, 8 bit, No Parity, 1 stop bit. The application output on the terminal utility are similar to the ones of the figures above.

### 2.5.2 RPL border router

This example lets you 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 an overall system.

#### 2.5.2.1 *System overview*

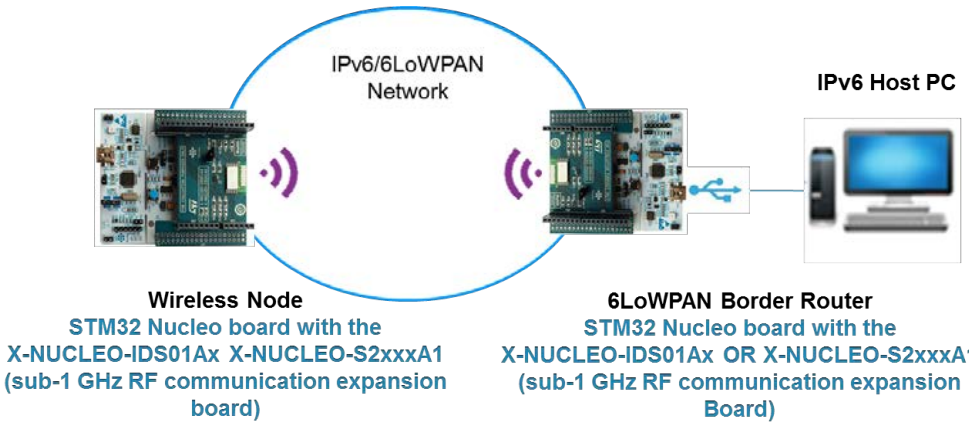
Take the example of a system consisting of:

- 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 is connected to the IPv6 host on the other. In our case, this device is an STM32 Nucleo board with a sub-1 GHz RF communication expansion board. To generate the firmware for this device, 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 board with a sub-1 GHz RF communication expansion board. To generate the firmware for this device, select the “UDP-sender” 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 6. Overall system architecture**




### 2.5.2.2 *IPv6 host setup*

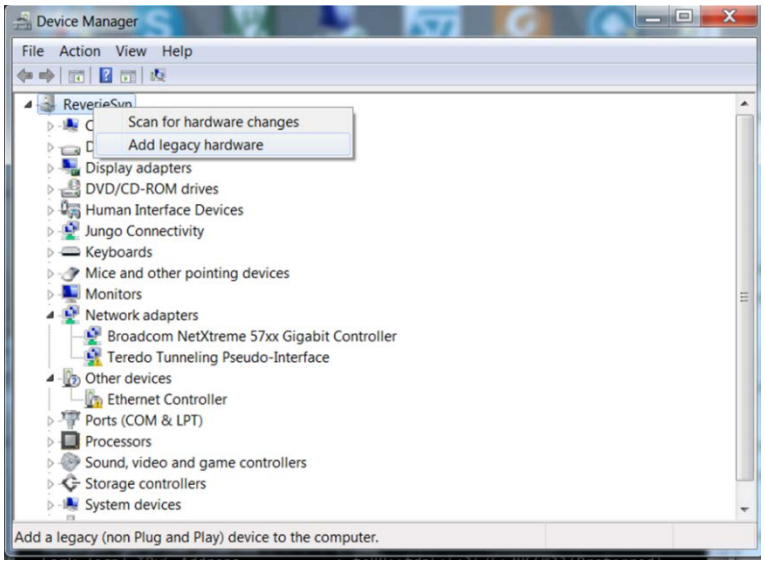
#### 2.5.2.2.1 IPv6 packets bridging (Windows setup)

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 using “Add legacy hardware” in the Device Manager, as shown in the following picture.

**Figure 7. Adding the network adapter in Windows**




**Step 2.** Follow the instructions on the Installer, by selecting the following entries: Add legacy hardware → Next → Install the hardware manually selected from a list → Next → Select Networks Adapters → Next →

Select Microsoftfrom the Manufacturer list then icrosoft Loopback Adapterfrom the Network Adapter list → "Next" → then finish the installation by clicking ext again

**Step 3.** Reboot your PC.

**Step 4.** Copy “cygwin1.dll” from “contiki/tools/cygwin” to the “wpcapslip6” folder.

**Step 5.** Install the WinPcaP Windows packet capture library. The installer and setup information can be found at https://www.winpcap.org/install/

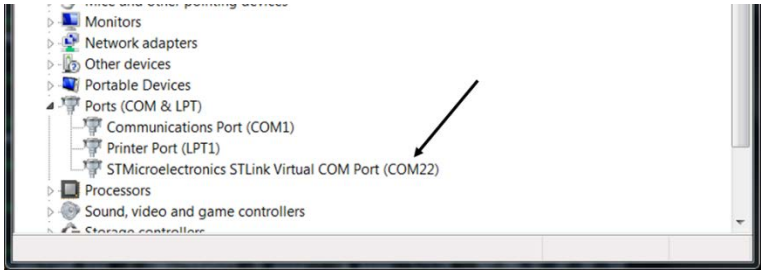
**Step 6.** Launch the Command prompt shell, run the wpcaslip6 utility with the rpl-border-router example

```
cd tools\stm32w\wpcapslip6
```

```
wpcapslip6.exe -s /dev/ttyS21 -b aaaa:: -a aaaa::1/128 [addr]
```

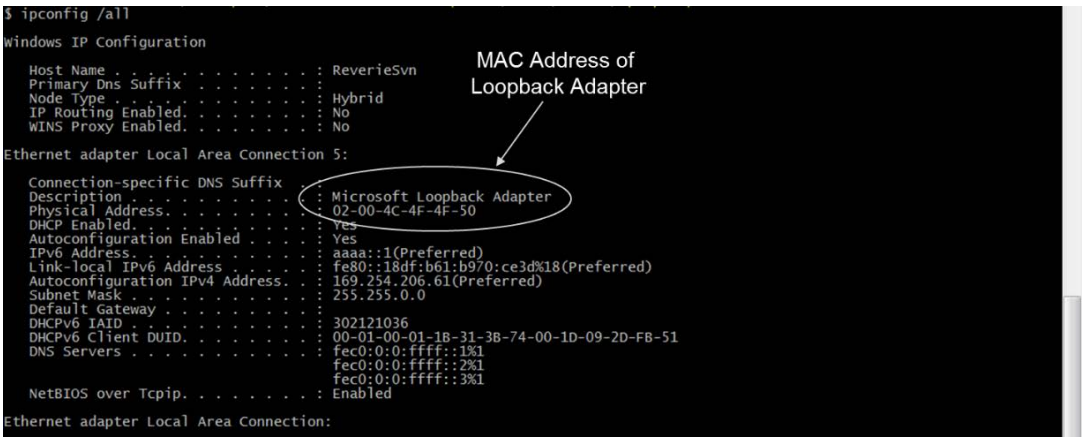
You can obtain the serial device number to be used (in this example, it is “ttyS21”) by looking at the Virtual COM Port number associated with the STM32 Nucleo board (see the following screenshot). The value to use when running the “wpcapslip6.exe” command is the COM port number minus 1. In our example, the COM Port number is 22, hence the “ttyS21” device parameter.

**Figure 8. STM32 Nucleo board virtual COM Port value**



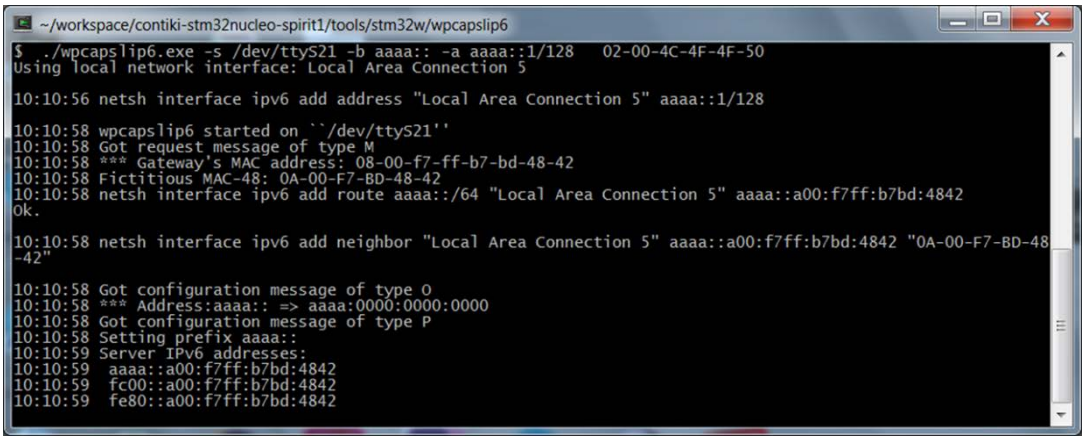

The `[addr]` parameter is the MAC address of the local network adapter. This information can be found via the `ipconfig /all` command shown below

**Figure 9. Reading the MAC address of the Microsoft loopback adapter**




A snapshot of the terminal running the wpcapslip6 utility is shown below.

Figure 10. **wpcapslip6 terminal window**




#### 2.5.2.2.2 IPv6 packets 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, bear in mind that only the projects provided with the SystemWorkbench for STM32 (SW4STM32) IDE are available for this platform.*

**Step 1.** Install the “tunslip6” software module in the Contiki folder order to exchange IPv6 packets over the serial line between the host PC stack and the border router IPv6 stack. To compile it, from the contiki directory, run:

```
cd ./tools
```

```
make tunslip6
```

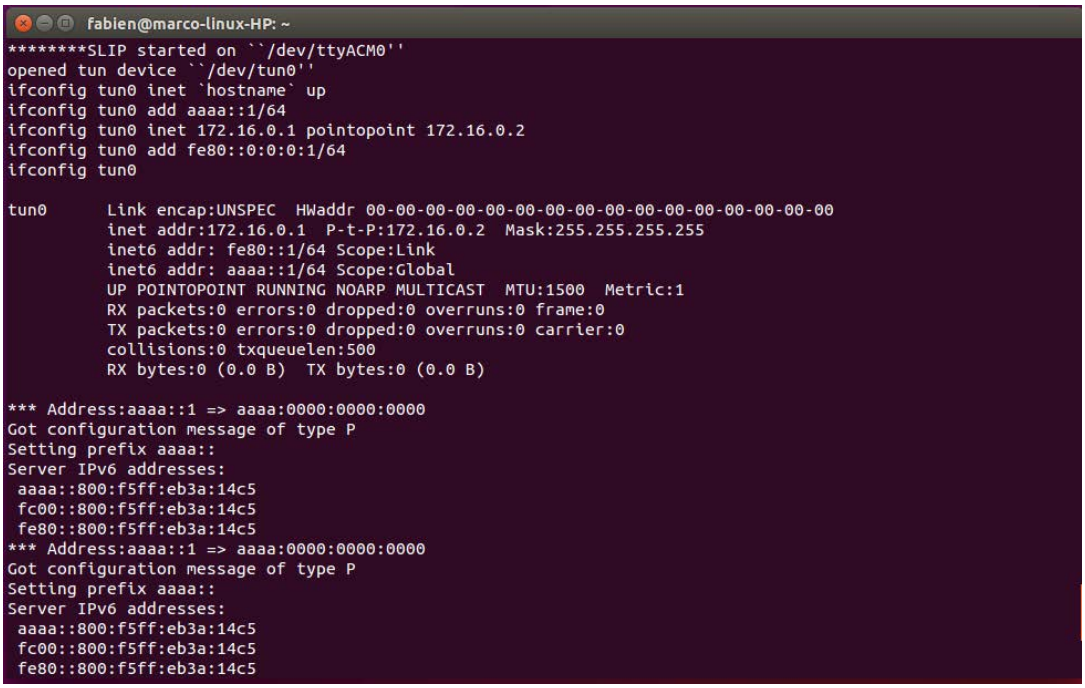
**Step 2.** We will assume that the border router device set up in the previous section is connected to the Linux host. Launch the “tunslip6” application by providing the parameter for the virtual communication port to which the border router device is connected (located in “/dev/ttyACMx” where “x” is the corresponding port number):

```
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. This triggers 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 11. **tunslip6 terminal window**



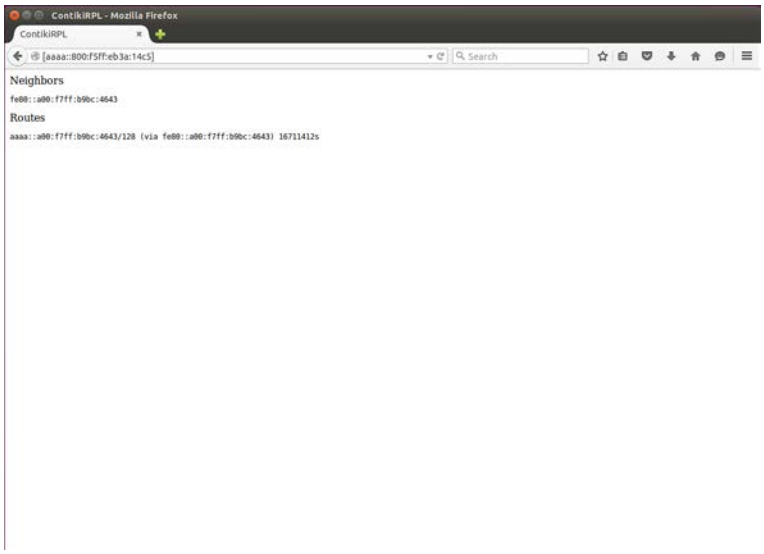

#### 2.5.2.2.3 IPv6 Host setup troubleshooting

1. The “tunslip6” application does not work properly with Ubuntu kernel 3.13.0-65-generic (part of the Ubuntu 14.04 LTS)
2. 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 https://www.cygwin.com/ml/cygwin/2008-08/msg00151.html
   b. Remove allocated COM ports, you can find some help at http://superuser.com/questions/408976/how-do-i-clean-up-com-ports-in-use
3. Disable your firewall
4. Ensure IPv6 is enabled:
   a. In the properties of the Loopback Adapter
   b. By setting registry key HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\TCPIP6\Parameters\DisabledComponents to the value 0x8E
   c. 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.

#### 2.5.2.3 *Contiki server access (border router) and connectivity test*

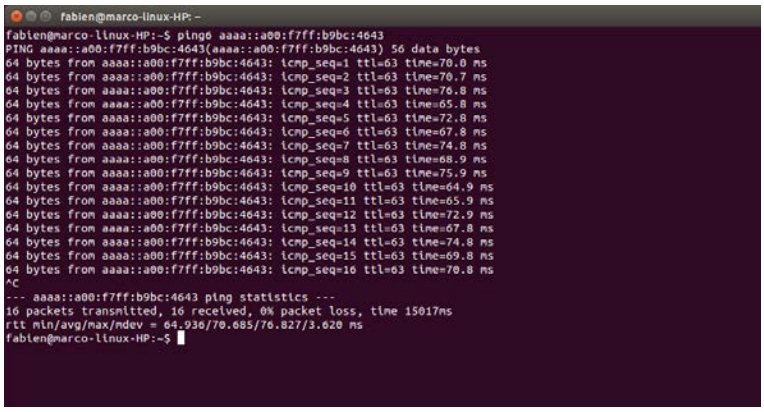
1. Open a web browser to access the Contiki server, which returns the RPL neighbors and routes. Access it via the first address listed in the “wpcapslip6” (Windows) or “tunslip6” (Linux) terminal window, after the “Server IPv6 addresses:” line. The following figure shows the web browser page after inserting in the URL the IPv6 addess of the server (since it is an IPv6 address, it must be enclosed in square brackets). Assuming that a wireless node, the setup of which is described in the previous section, is running, the IPv6 address of the wireless node should be in the “Neighbors” list.

Figure 12. **ContikiRPL server access**




1. 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::a00:f7ff:b9bc:4643`. 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 13. **Ping6 command window snapshot**




## 2.6 System setup guide

### 2.6.1 Hardware description

This section describes the hardware components needed for developing a Contiki-based application using the SPIRIT1 or S2-LP sub-1 GHz RF communication transceiver.

#### 2.6.1.1 *STM32 Nucleo platform*

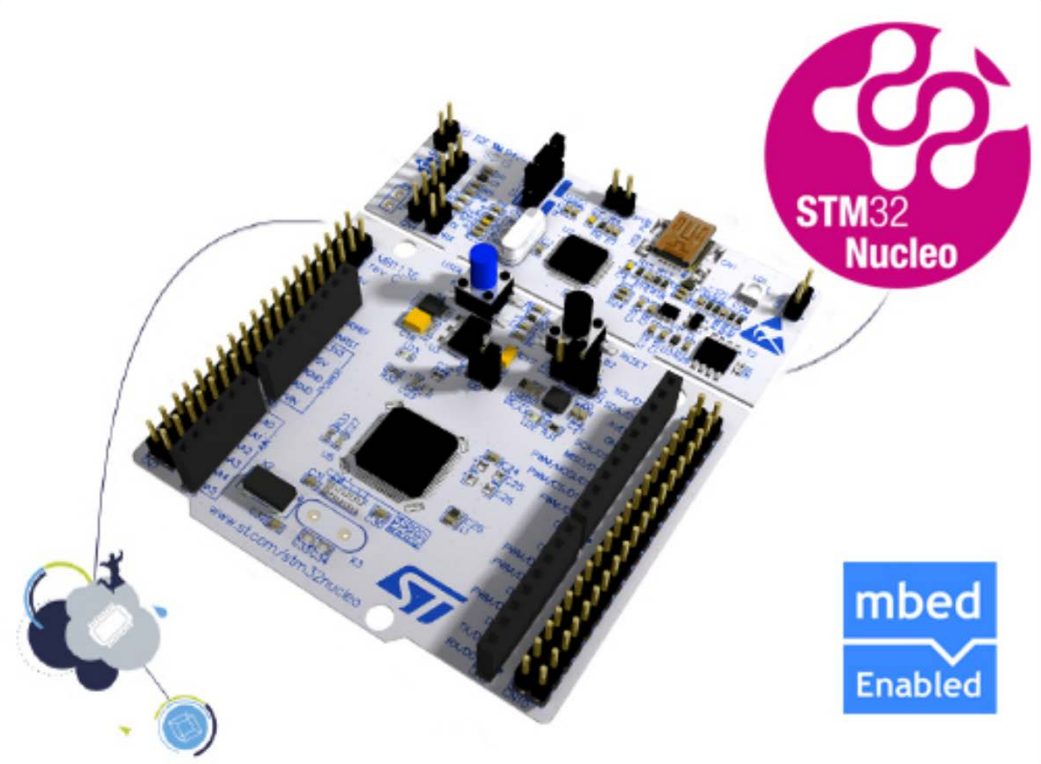
STM32 Nucleo development boards provide an affordable and flexible way for users to test solutions and build prototypes with any STM32 microcontroller line.

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.

**Figure 14. STM32 Nucleo board**



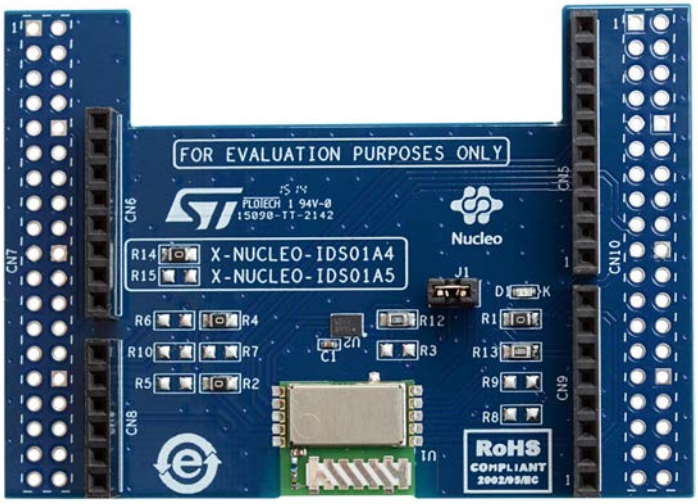

Information regarding the STM32 Nucleo board is available at www.st.com/stm32nucleo

#### 2.6.1.2 *X-NUCLEO-IDS01Ax expansion board*

The X-NUCLEO-IDS01A4 and X-NUCLEO-IDS01A5 evaluation boards allow you to evaluate the features and capabilities of the SPIRIT1 or low data rate, low power sub-1 GHz transceiver device.

These expansion boards include on-board SPI EEPROM to store parameters and user interface signal LED. The X-NUCLEO-IDS01A4 board operates the SPIRIT1 transceiver at 868MHz, while the X-NUCLEO-IDS01A5 board operates the SPIRIT1 transceiver at 915MHz.

Figure 15. **X-NUCLEO-IDS01Ax expansion board**




Information regarding the X-NUCLEO-IDS01A4 and X-NUCLEO-IDS01A5 expansion boards is available on www.st.com at www.st.com/x-nucleo.

### 2.6.1.3 *X-NUCLEO-S2868A1 expansion board*

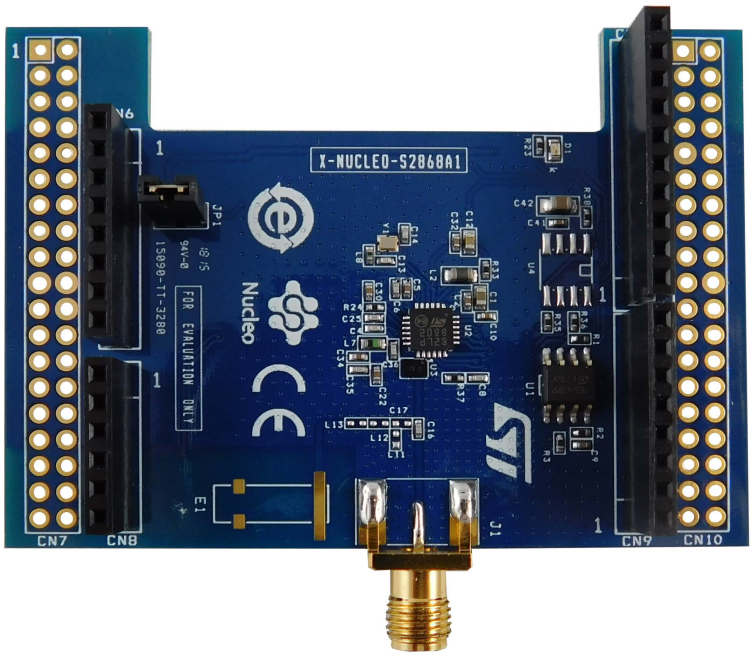
The X-NUCLEO-S2868A1 expansion board is based on the S2-LP radio and operates in the 868 MHz ISM frequency band.

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 16. **X-NUCLEO-S2868A1 expansion board**



### 2.6.2 Software description

The following software components are required to set up a suitable development environment for creating applications with the STM32 Nucleo board plus SPIRIT1 or S2-LP sub-1 GHz RF communication expansion board, running Contiki OS:

- **X-CUBE-SUBG1** expansion for STM32Cube. The X-CUBE-SUBG1 firmware and related documentation is available on st.com. The Contiki6LP library and the related documentation and projects files are included in version 2.0.0 (or higher) of X-CUBE-SUBG1.
- **Development tool-chain and Compiler** supported by the STM32Cube expansion software:
  - IAR Embedded Workbench for ARM® (IAR-EWARM) toolchain + ST-LINK
  - RealView Microcontroller Development Kit (MDK-ARM-STM32) toolchain + ST-LINK
  - System Workbench for STM32 (SW4STM32) + ST-LINK

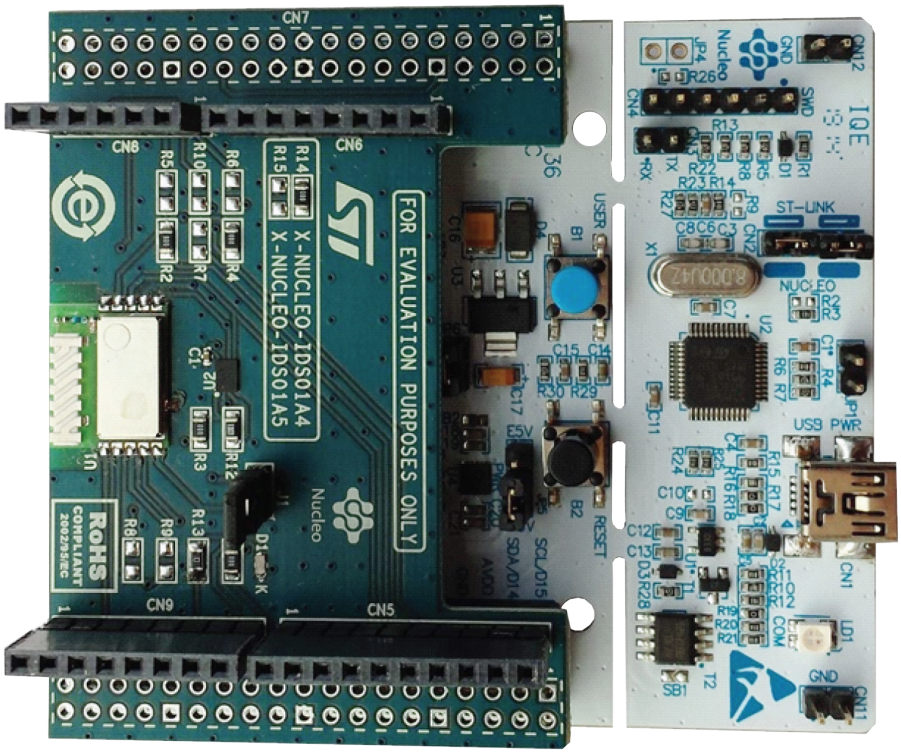
### 2.6.3 Board setup guide

This section describes how to set up and program the STM32 Nucleo boards to test the applications provided or developed by the user.

**Step 1.** Check that the jumper on the J1 connector on the X-NUCLEO-IDS01Ax expansion board is connected. This jumper provides the required voltage to the devices on the board.

**Step 2.** Connect the X-NUCLEO-IDS01Ax board to the top of the STM32 Nucleo board (NUCLEO-L152RE or NUCLEO-F401RE), as shown below.

**Figure 17. X-NUCLEO-IDS01Ax expansion board connected to STM32**




**Step 3.** Power the STM32 Nucleo board through the Mini-B USB cable connected to your PC.

**Step 4.** Program the firmware on the STM32 Nucleo board. This can be done by copying the binary file on the USB mass storage that is automatically created when plugging the STM32 Nucleo board to the PC (e.g., drag and drop operation). If the host is a Linux PC, the STM32 Nucleo L1 device can be found in the /media folder with the name of the board (e.g., NUCLEO_L152RE – if more than one STM32 Nucleo board is connected to the PC USB port, the name is “NUCLEO_L152REx”, x=1, 2,….).

For example, if the path to the mass storage created when connecting the STM32 Nucleo L1 platform is “/media/NUCLEO_L152RE”, then programming the board with a binary file named “my_firmware.bin” can simply be done by running the following command: `cp my_firmware.bin /media/`

`NUCLEO_L152RE`. Alternatively, the user can program the STM32 Nucleo board directly using one of the supported development toolchains (refer to the relevant toolchain user manual for details).

**Step 5.** Reset the MCU with the black RESET button on the STM32 Nucleo board.

**Step 6.** Open a serial line monitor utility, select the serial port name to which the board is connected and configure it with the following parameters:

- Baud Rate = 115200
- Parity = None
- Bits = 8
- Stopbits = 1

The serial line monitor utility will show the data printed by the selected example.

*Note: (Linux users can refer to Section 2.5.1 UDP-sender and UDP-receiver and follow the steps to configure minicom).*

# Revision history

Table 2. **Document revision history**

| Date | Version | Changes |
|---|---|---|
| 18-Mar-2016 | 1 | Initial release. |
| 04-Apr-2016 | 2 | Updated Section 2.5.2.1: "System overview". |
| 13-Mar-2017 | 3 | Updated title, Section "Introduction", Section 2.1: "Overview", Section 2.2: "Architecture", Section 2.3: "Folder structure", Section 2.5.1: "UDP-sender and UDP-receiver", Section 2.5.2.2.1: "IPv6 packets bridging (Windows setup)", Section 2.5.2.2.3: "IPv6 Host setup troubleshooting" and Section 2.6.2: "Software description". |
| 14-May-2018 | 4 | Added Section 2.6.1.3 X-NUCLEO-S2868A1 expansion board.<br>Added S2-LP and X-NUCLEO-S2868A1 compatibility information.<br>Updated Figure 1. X-CUBE-SUBG1 software architecture and Figure 6. Overall system architecture. |

# Contents

# List of tables

# List of figures