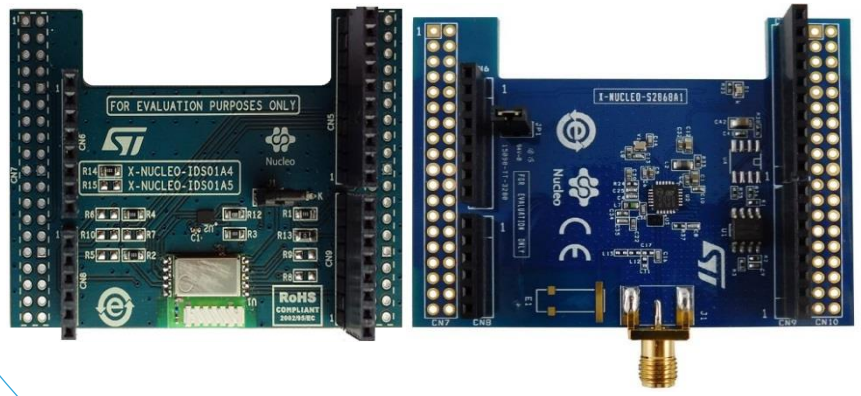
Quick Start Guide

Contiki OS and 6LoWPAN sub-1GHz RF communication software expansion for STM32 Cube (Contiki6LP)



life.augmented

Version 1.3 (December 10, 2018)

Quick Start Guide Contents

Contiki6LP: Contiki OS/6LoWPAN and sub-1GHz RF communication Hardware and Software overview

Setup & Demo Examples

Documents & Related Resources

STM32 Open Development Environment: Overview

Sub-1 GHz RF expansion boards based on SPIRIT1

Hardware overview

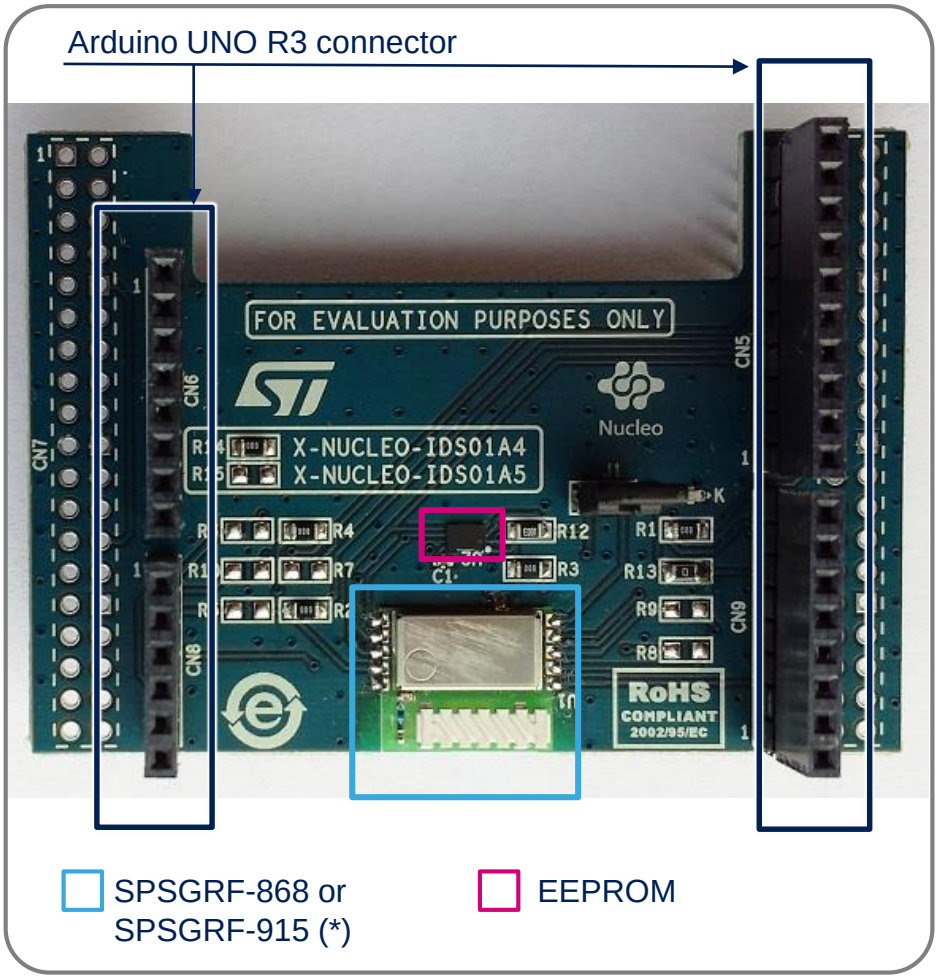
Hardware description

- The X-NUCLEO-IDS01A4, X-NUCLEO-IDS01A5 are evaluation boards based on the SPIRIT1 RF modules SPSGRF-868 and SPSGRF-915
- The SPIRIT1 module communicates with the STM32 Nucleo developer board host microcontroller though an SPI link available on the Arduino UNO R3 connector.

Key products on board

SPSGRF
SPIRIT1 (Low data-rate, low-power sub-1GHz transceiver) module

M95640-RMC6TG
64-Kbit serial SPI bus EEPROM



(*) Identification of the operating frequency of the X-NUCLEO-IDS01Ax (x=4 or 5) is performed through two resistors (R14 and R15).

Latest info available at www.st.com
X-NUCLEO-IDS01A4
X-NUCLEO-IDS01A5

life.augmented

Sub-1 GHz 868 MHz RF expansion board based on S2-LP

Hardware overview

Hardware description

- The X-NUCLEO-S2868A1 evaluation board is based on the S2-LP sub-1 GHz ultra-low power low data-rate transceiver.
- The S2-LP IC communicates with the STM32 Nucleo developer board host microcontroller though an SPI link available on the Arduino UNO R3 connector.

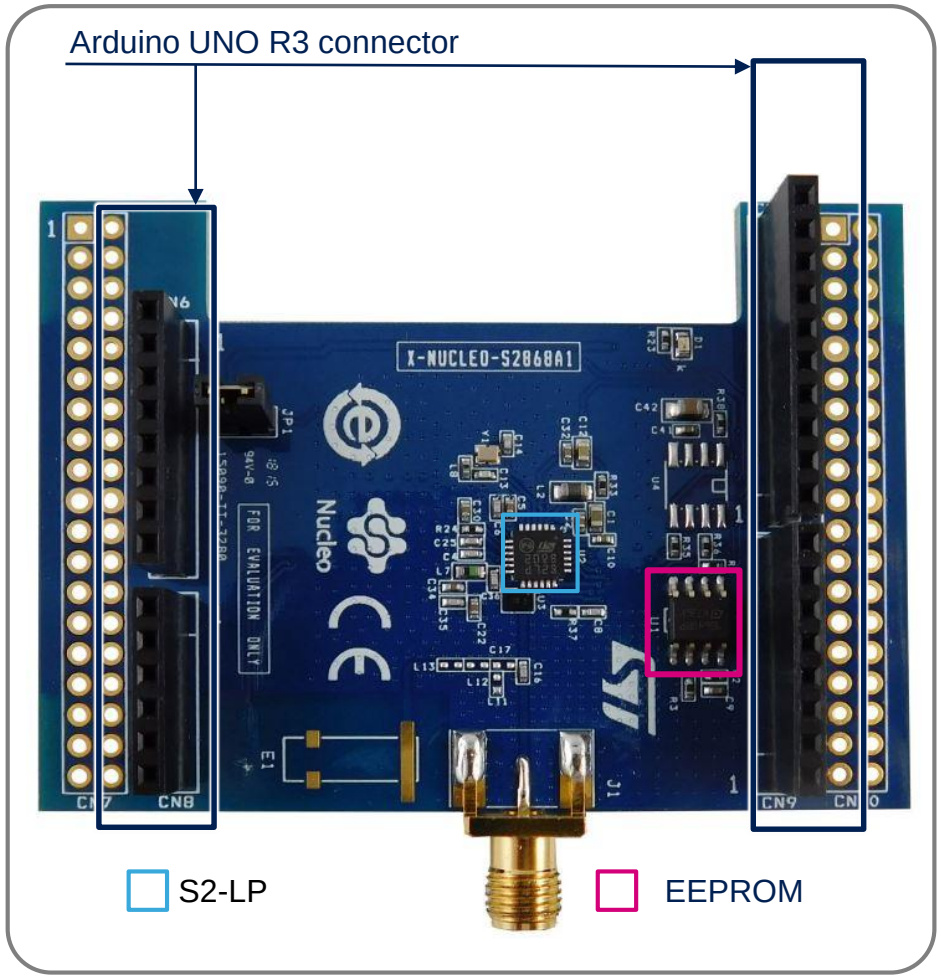
Key features

- Programmable RF output power up to +16 dBm
- Modulation schemes: 2-FSK, 2-GFSK, 4-FSK, 4-GFSK, OOK and ASK
- Air data rate from 0.1 to 500 kbps
- Ultra-low power consumption: 7 mA RX and 10 mA TX at +10 dBm
- IEEE 802.15.4g hardware packet support with whitening, FEC, CRC and dual SYNC word detection
- RX and TX 128 byte FIFO buffers

Key products on board

S2LP
Ultra-low power, high performance, sub-1GHz transceiver

M95640-RMN6TP
64-Kbit serial SPI bus EEPROM



Latest info available at www.st.com
X-NUCLEO-S2868A1

life.augmented

Software Overview

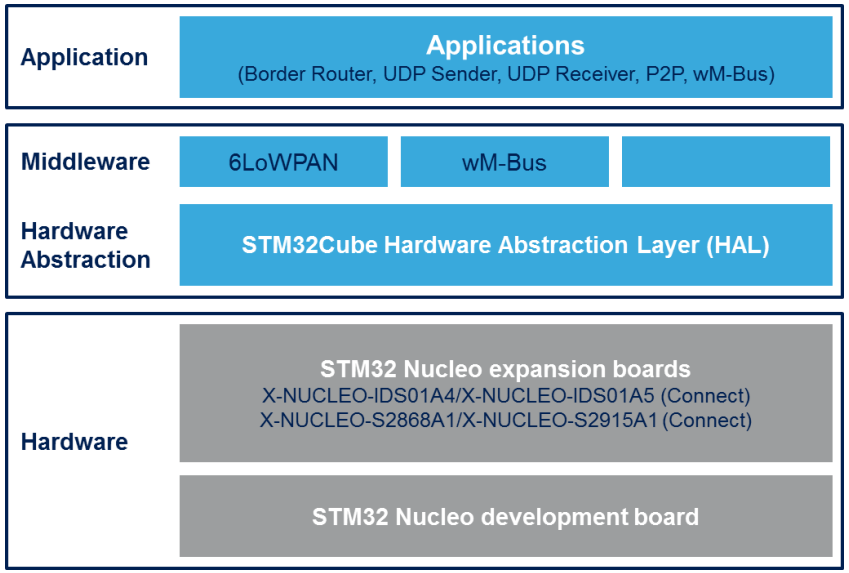
Contiki6LP Software Description

Contiki6LP is a library implemented as a STM32Cube middleware ready to be integrated in projects based on STM32Cube and X-CUBE-SUBG1 expansion software. The expansion software is built on STM32Cube software technology for portability across different STM32 microcontrollers. The software includes examples for sending messages via UDP over 6LoWPAN, using the SPIRIT1/S2-LP sub-1GHz radio transceiver.

Key features

- Middleware library with Contiki OS and Contiki 6LoWPAN protocol stack 3.x
- Support for mesh networking technology by the means of the standard RPL protocol
- Built-in support for STM32 L1 and F4 platforms
- Example applications including UPD sender and receiver, and border router
- Examples available for NUCLEO-F401RE and NUCLEO-L152RE
- Easy portability across different MCU families, thanks to STM32Cube
- Free, user-friendly license terms

Overall Software Architecture



Latest info available at www.st.com
X-CUBE-SUBG1

Quick Start Guide Contents

Contiki6LP: Contiki OS/6LoWPAN and sub-1GHz RF communication Hardware and Software overview

Setup & Demo Examples

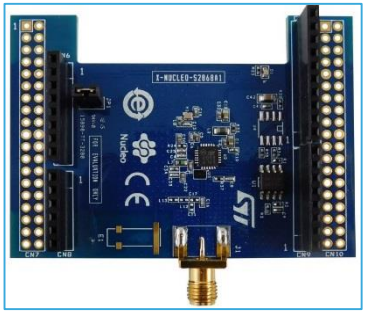
Documents & Related Resources

STM32 Open Development Environment: Overview

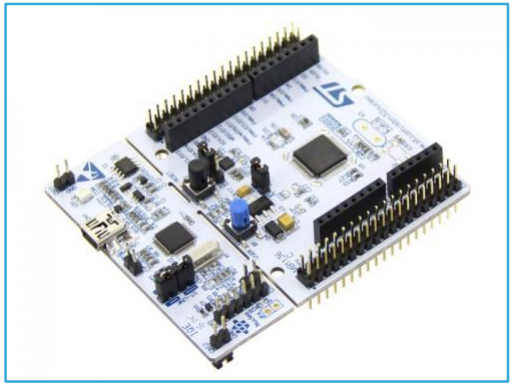
Setup & Demo Examples
HW prerequisites

- STM32 Nucleo development board
  NUCLEO-L152RE or NUCLEO-F401RE
- Sub-1GHz RF expansion board for STM32 Nucleo based on the Spirit 1 (**X-NUCLEO-IDS01A4**) or S2LP-868 module (**X-NUCLEO-S2868A1**)
- Windows/Linux PC
- mini USB cable

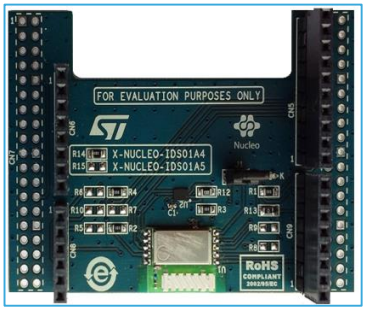
mini USB cable



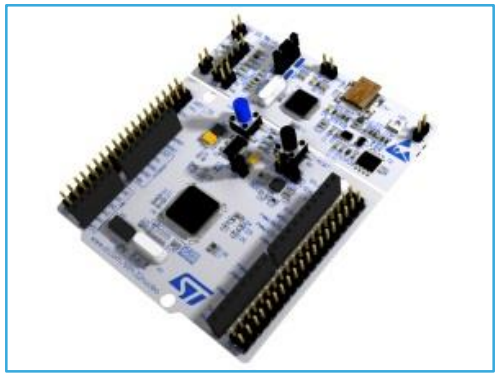
X-NUCLEO-S2868A1



NUCLEO-F401RE



X-NUCLEO-IDS01A4



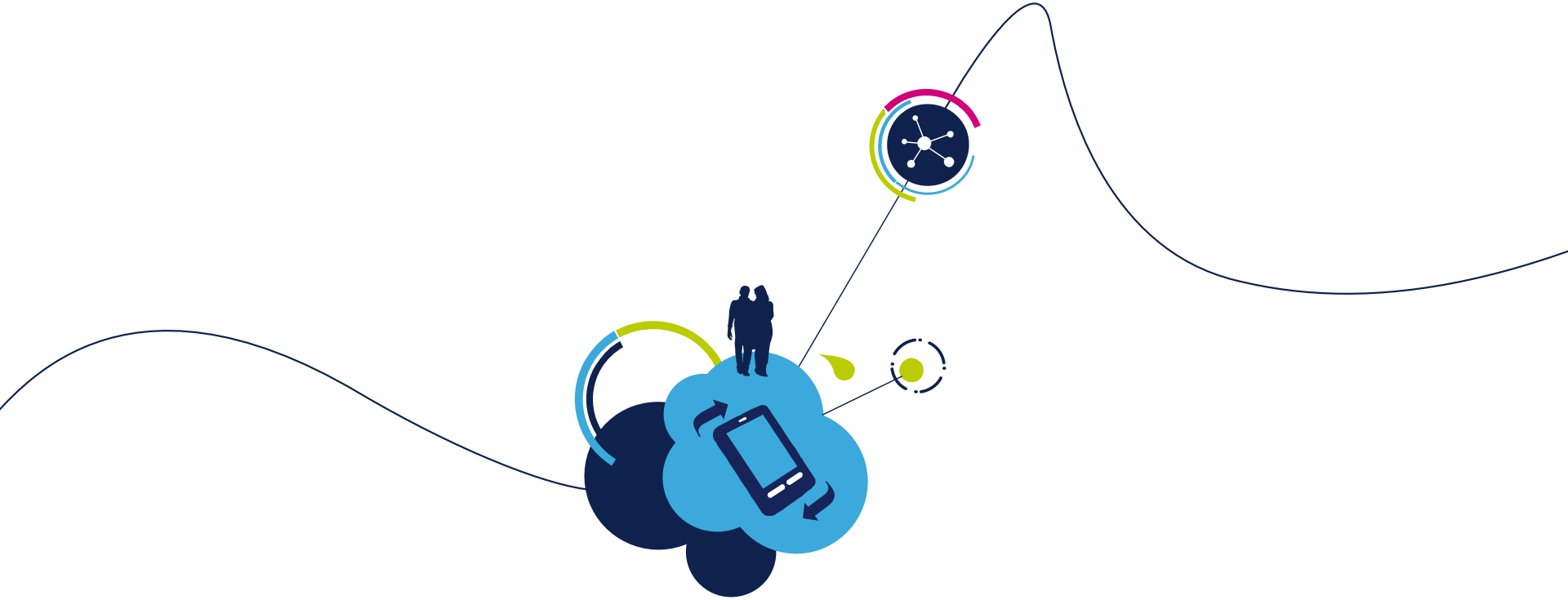
NUCLEO-L152RE

life.augmented

SW prerequisites

- X-CUBE-SUBG1 package
  - Download and extract the **X-CUBE-SUBG1** package, version 3.0.0 or higher
- A toolchain to build the firmware
  - The Contiki6LP library has been developed and tested with
    - IAR Embedded Workbench for ARM® (EWARM) toolchain + ST-Link
    - RealView Microcontroller Development Kit (MDK-ARM) toolchain + ST-LINK
    - System Workbench for STM32 (SW4STM32) + ST-LINK (*)
- Serial line monitor e.g. Termite (Windows), or Minicom (Linux)

(*) For Linux users: System Workbench for STM32 (SW4STM32) is the only supported IDE



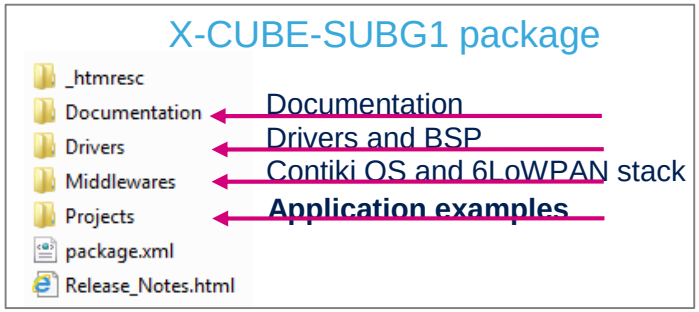
Demo Execution Using SPSGRF (SPIRIT1)

Start coding in just a few minutes with Contiki6LP

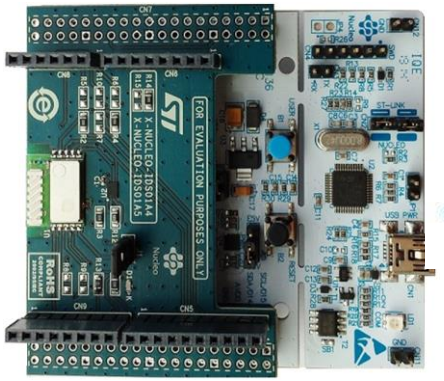
1 Go to www.st.com/x-nucleo

2 Select X-NUCLEO-IDS01Ax

3 Download and unpack X-CUBE-SUBG1

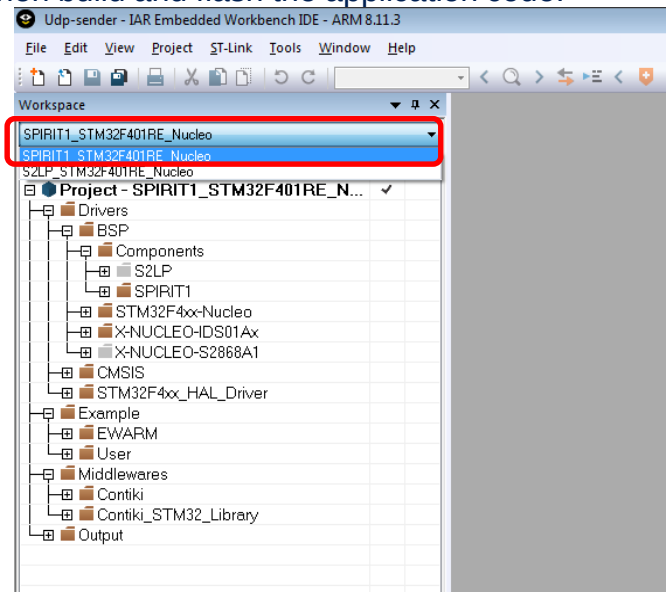


4 Download & install STM32 Nucleo ST-LINK/V2-1 USB driver



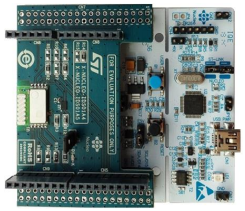
5 Open project example e.g. Udp-sender

6 Select the SPIRIT1 radio configuration And then build and flash the application code.

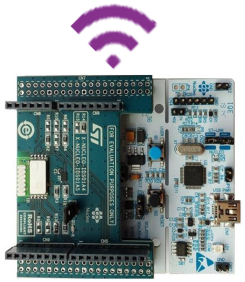


Demo Overview – UDP Sender and Receiver

6LoWPAN Udp-sender node
NUCLEO-L152RE or NUCLEO-F401RE
X-NUCLEO-IDS01A4 or X-NUCLEO-IDS01A5



UDP
Messages



6LoWPAN Udp-receiver node
NUCLEO-L152RE or NUCLEO-F401RE
X-NUCLEO-IDS01A4 or X-NUCLEO-IDS01A5

Host PC
with a terminal
Utility

USB1

USB2

UDP Sender and Receiver examples in a few steps (1/2)

Download and extract **X-CUBE-SUBG1**

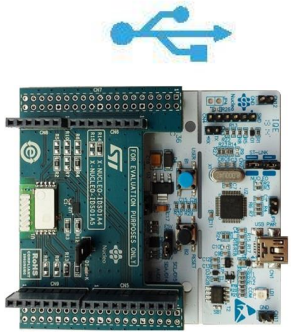
1

2

Compile the firmware for the UDP Receiver node:
Select the “**Udp-receiver**” application and build the Project using a supported IDE. Alternatively you can use a pre-built binary that is provided for running this application with the selected STM32 Nucleo board

3

Connect the STM32 Nucleo based kit acting as a “UDP Receiver” to a PC USB slot and program the device



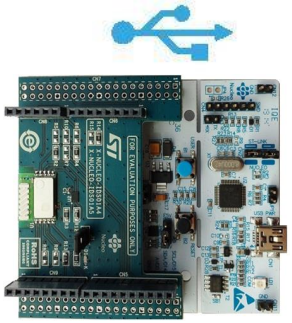
Copy the binary file
(e.g. drag & drop) to the USB mass storage corresponding to the STM32 Nucleo board

4

Compile the firmware for the UDP sender node:
Select the “**Udp-sender**” application and build the Project using a supported IDE. Alternatively you can use a pre-built binary that is provided for running this application with the selected STM32 Nucleo board

5

Connect the STM32 Nucleo based kit acting as a “UDP Sender” to a PC USB slot and program the device



copy the file
(e.g. drag & drop) to the USB mass storage corresponding to the STM32 Nucleo board

UDP Sender and Receiver examples in a few steps (2/2)

Launch the terminal application and set the UART port to 115200 bps, 8 bit, No Parity, 1 stop bit

Select the device corresponding to the UDP sender node (e.g. on a Linux host, it will be a *ttyACMx* device type)

6

7

The terminal should be printing something like

```
Contiki and Spirit correctly configured... Starting all processes
IPv6 addresses: aaaa::951:3333:7234:7334
fe80::951:3333:7234:7334
Service 190 not found
Service 190 not found
Service 190 not found
Service 190 not found
Service 190 not found
```

Udp-sender window

8

If everything has been done correctly, the output in the terminal should now be something similar to this:

```
Contiki and Spirit correctly configured... Starting all processes
IPv6 addresses: aaaa::951:3333:7234:7334
fe80::951:3333:7234:7334
Service 190 not found
Service 190 not found
Service 190 not found
Service 190 not found
Service 190 not found
Service 190 not found
Sending unicast to aaaa::e51:3333:7334:6334
Sending unicast to aaaa::e51:3333:7334:6334
Sending unicast to aaaa::e51:3333:7334:6334
Sending unicast to aaaa::e51:3333:7334:6334
Sending unicast to aaaa::e51:3333:7334:6334
Sending unicast to aaaa::e51:3333:7334:6334
```

Udp-sender window

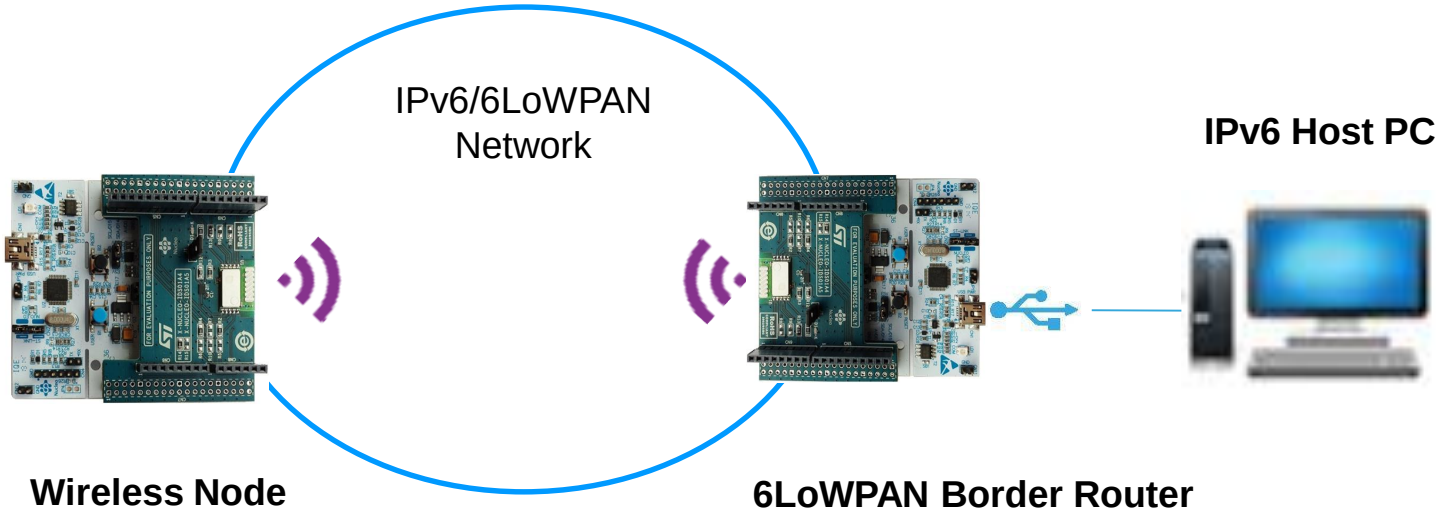
9

Repeat step 6-8 for the project **Udp-receiver** (remember to open a new *terminal* window): The received UDP messages are shown

```
Contiki and Spirit correctly configured... Starting all processes
IPv6 addresses: aaaa::e51:3333:7334:6334
fe80::e51:3333:7334:6334
Data received from aaaa::951:3333:7234:7334 on port 1234 from port 1234 with length 10: 'Message 0'
Data received from aaaa::951:3333:7234:7334 on port 1234 from port 1234 with length 10: 'Message 1'
Data received from aaaa::951:3333:7234:7334 on port 1234 from port 1234 with length 10: 'Message 2'
Data received from aaaa::951:3333:7234:7334 on port 1234 from port 1234 with length 10: 'Message 3'
Data received from aaaa::951:3333:7234:7334 on port 1234 from port 1234 with length 10: 'Message 4'
Data received from aaaa::951:3333:7234:7334 on port 1234 from port 1234 with length 10: 'Message 5'
```

Udp-receiver window

Demo Overview – Border Router Example



Border Router Example in a few steps (1/3)

Download and extract **X-CUBE-SUBG1**

1

2

Compile the firmware for a wireless node:
Select the “**Udp-sender**” application and build the Project using a supported IDE. Alternatively you can use a pre-built binary that is provided for running this application with the selected STM32 Nucleo board

3

Connect the board to a
PC USB slot and program the device

Copy the binary file
(e.g. drag & drop) to the USB mass storage corresponding to the STM32 Nucleo board

4

Compile the firmware for the border router node:
Select the “**Border-router**” application and build the Project using a supported IDE. Alternatively you can use a pre-built binary that is provided for running this application with the selected STM32 Nucleo board

5

Connect the board to USB and program the device

copy the file
(e.g. drag & drop) to the USB mass storage corresponding to the STM32 Nucleo board

Border Router Example in a few steps (2/3)

6 Setup the IPv6 Host PC for IP traffic bridging between host and 6LowPAN border Router

From: *[ROOT]/Middlewares/Third_Party/Contiki*

Windows PC setup (Win 7/8)
using "**wpcapslip6**" utility

1. wpcapslip6 needs a working network adapter:
The Microsoft loopback adapter can be installed via "Add legacy hardware" in the Windows Device Manager (reboot is needed after installation of the loopback adapter)
2. Copy "cygwin1.dll" from "tools/cygwin" to wcapslip6 folder
3. Install WinPcaP
4. run Cygwin as administrator

OR

Linux PC setup (Ubuntu)
using "**tunslip6**" utility

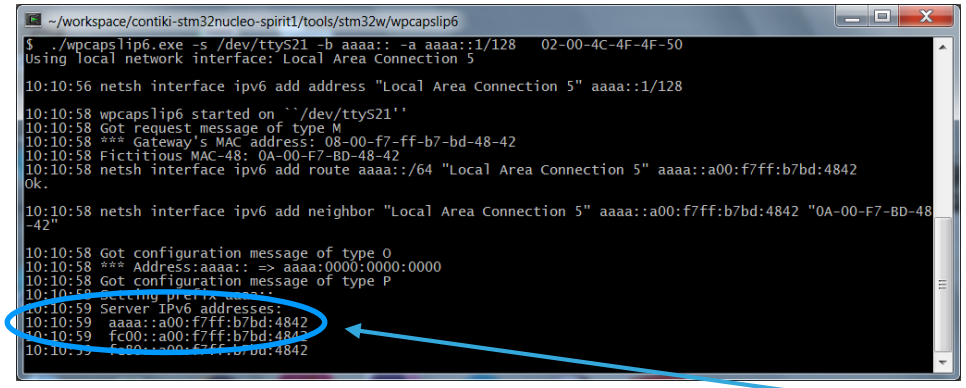
```
cd ./tools
make tunslip6
sudo ./tunslip6 –s /dev/ttyACMx aaaa::1/64
```

ttySz / ttyACMx depends on the device enumeration, you can use tab auto completion under both Linux and Cygwin

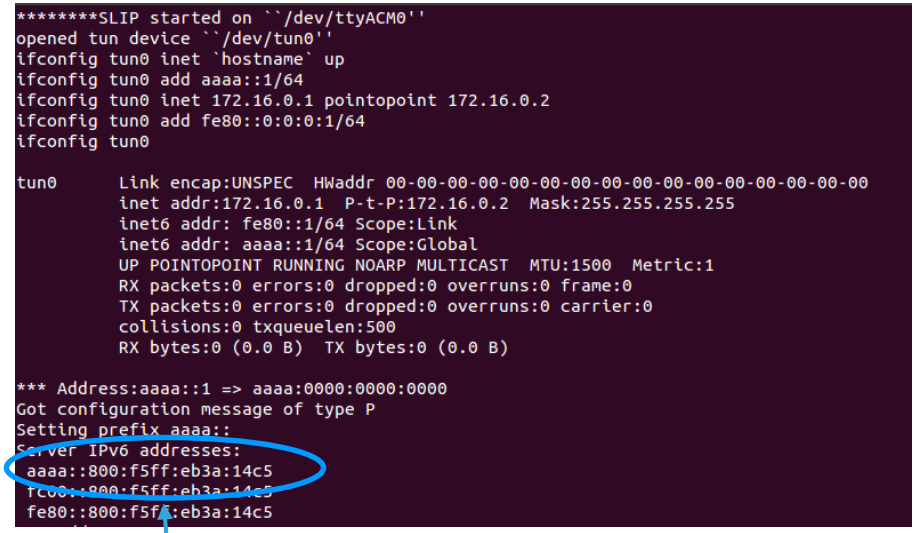
wpcapslip6 utility can then be used with the rpl-border-router example

```
cd ./tools/stm32w/wpcapslip6
./wpcapslip6 –s /dev/ttySz –b aaaa:: -a aaaa::1/128 [addr]
```

Where *[addr]* is the MAC address of the local net adapter



wpcapslip6 terminal window output



Tunslip6 terminal window output

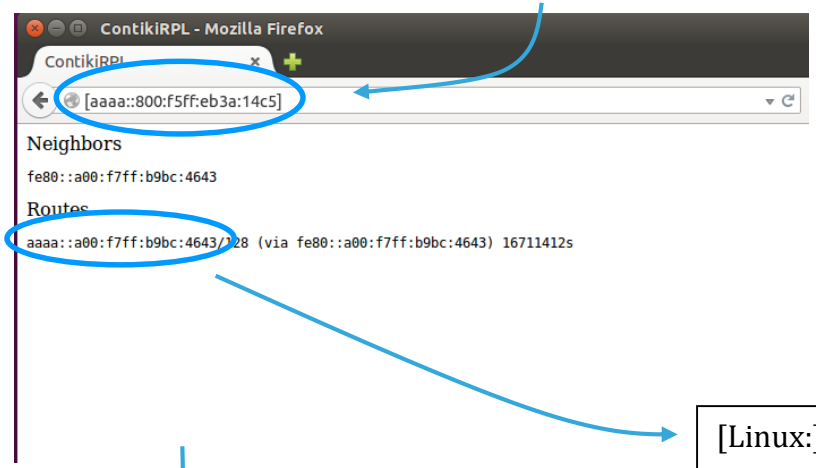
Contiki server address (used in the next step)

life.augmented

Border Router Example in a few steps (3/3)

7 Open a Web browser (Firefox) to access the Contiki server providing the RPL neighbors and routes information.

Contiki server address (see previous step) between brackets, e.g. [aaaa::800:f5ff:eb3a:14c5]



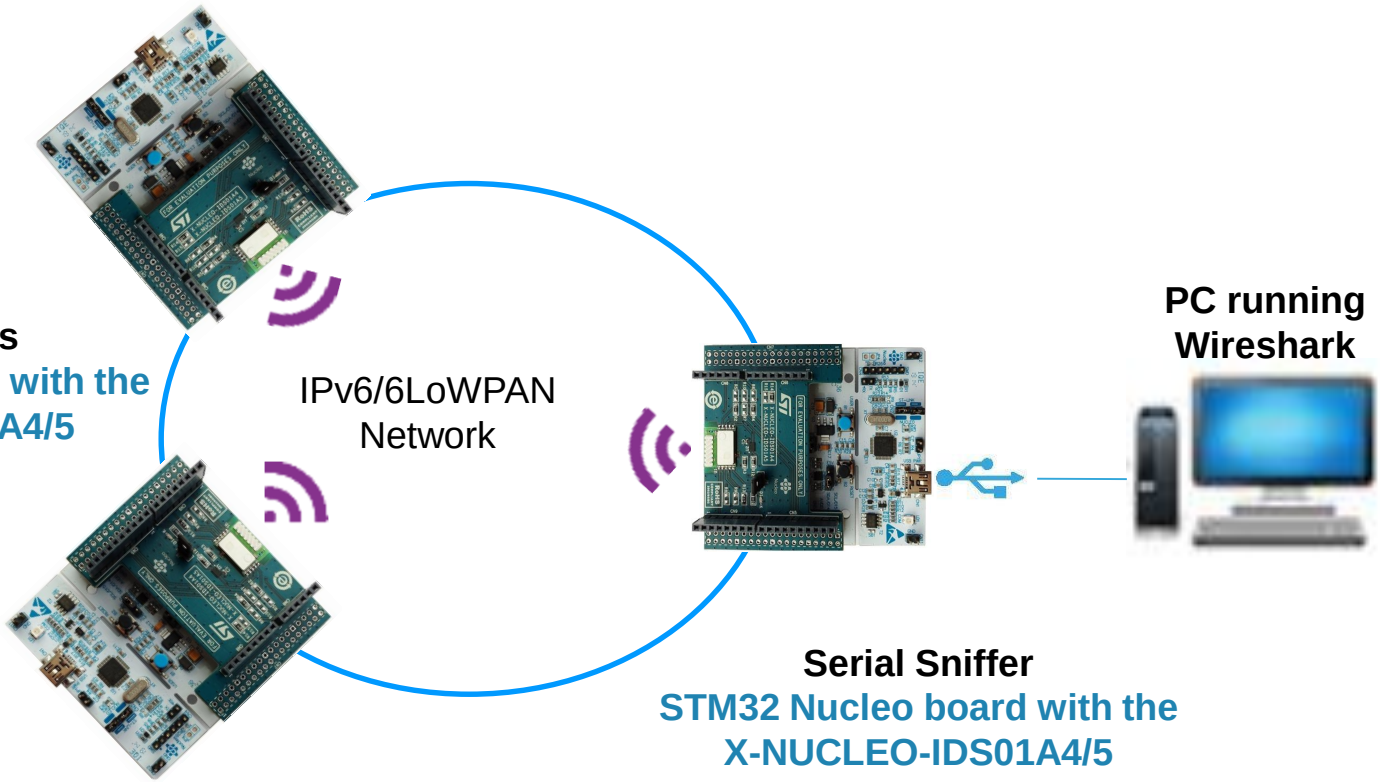
[Linux:] *ping6 aaaa::a00:f7ff:b9bc:4643*

[Windows:] *ping -6 aaaa::a00:f7ff:b9bc:4643*

8 Ping the wireless Node to test the 6LoWPAN connectivity

```
PING aaaa::a00:f7ff:b9bc:4643(aaaa::a00:f7ff:b9bc:4643) 56 data bytes
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=1 ttl=63 time=70.0 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=2 ttl=63 time=70.7 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=3 ttl=63 time=76.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=4 ttl=63 time=65.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=5 ttl=63 time=72.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=6 ttl=63 time=67.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=7 ttl=63 time=74.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=8 ttl=63 time=68.9 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=9 ttl=63 time=75.9 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=10 ttl=63 time=64.9 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=11 ttl=63 time=65.9 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=12 ttl=63 time=72.9 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=13 ttl=63 time=67.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=14 ttl=63 time=74.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=15 ttl=63 time=69.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=16 ttl=63 time=70.8 ms
^C
--- aaaa::a00:f7ff:b9bc:4643 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 15017ms
rtt min/avg/max/mdev = 64.936/70.685/76.827/3.620 ms
fabien@marco-linux-HP:~$
```

Demo Overview – Serial Sniffer Example



Serial Sniffer Example in a few steps (1/3)

Download and extract **X-CUBE-SUBG1**

1

2

Compile the firmware for the Serial Sniffer:
Select the "**Serial-sniffer**" application and build the Project using a supported IDE. Alternatively you can use a pre-built binary that is provided for running this application with the selected STM32 Nucleo board

3

Connect the board to a
PC USB slot and program the device

Copy the binary file
(e.g. drag & drop) to the USB mass storage corresponding to the STM32 Nucleo board

4

A pre-requisite to use the Serial Sniffer is a running 6LoWPAN network, you can refer to the example UDP Sender and Receiver described in previous slides, i.e. select "**Udp-sender**" and "**Udp-receiver**" applications and build the Projects using a supported IDE. Alternatively you can use the pre-built binaries that are provided for running these applications with the selected STM32 Nucleo board

5

Connect the board to USB and
program the device (repeat this for both firmwares)

copy the file
(e.g. drag & drop) to the USB mass storage corresponding to the STM32 Nucleo board

Serial Sniffer Example in a few steps (2/3)

6 Setup the PC running Wireshark application

From: *[ROOT]/Utilities/serial-sniffer*

Windows PC setup (Win 7/8)
using "**serialdump-windows.exe**" utility

1. run Cygwin as administrator
2. serialdup-windows.exe utility is provided pre-compiled, but in case you need to recompile it:
   - *cd serialdump-src*
   - *make*
     - (or *gcc –o serialdump-windows.exe serialdump.c*)
   - *mv serialdump-windows.exe ..*
3. Run the following command chain (it is ONE line of three commands in pipe "|" one with the next one)

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

OR

Linux PC setup (Ubuntu)
using "**serialdump-linux**" utility

1. serialdump-linux utility has to be compiled:
   - *cd serialdump-src*
   - *make*
     - (or *gcc –o serialdump-linux serialdump.c*)
   - *mv serialdump-linux ..*
2. Run the following command chain (it is ONE line of three commands in pipe "|" one with the next one)

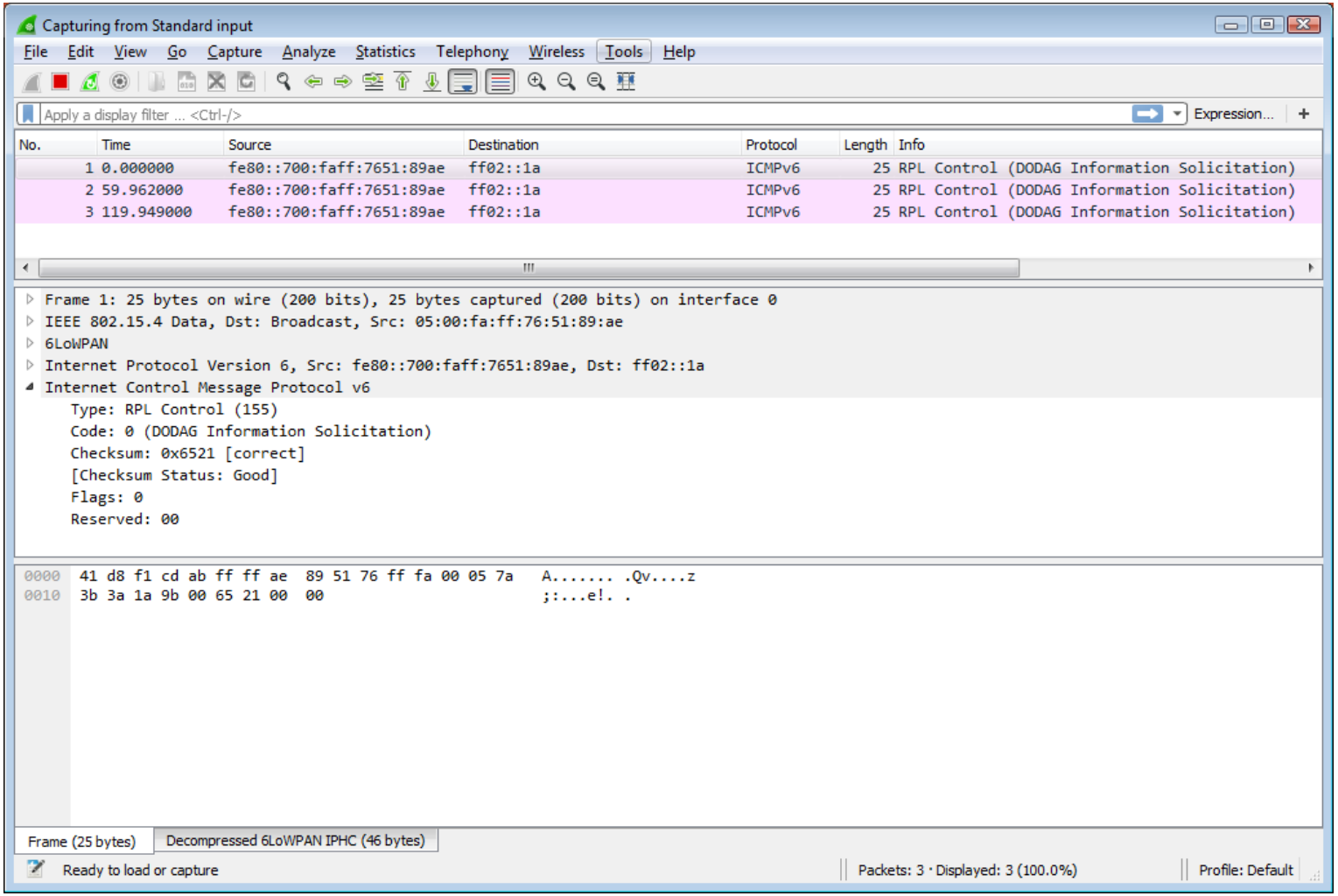
> sudo serialdump-linux -b115200 /dev/ttyACMx | ./convert-to-binary | wireshark -k -i -

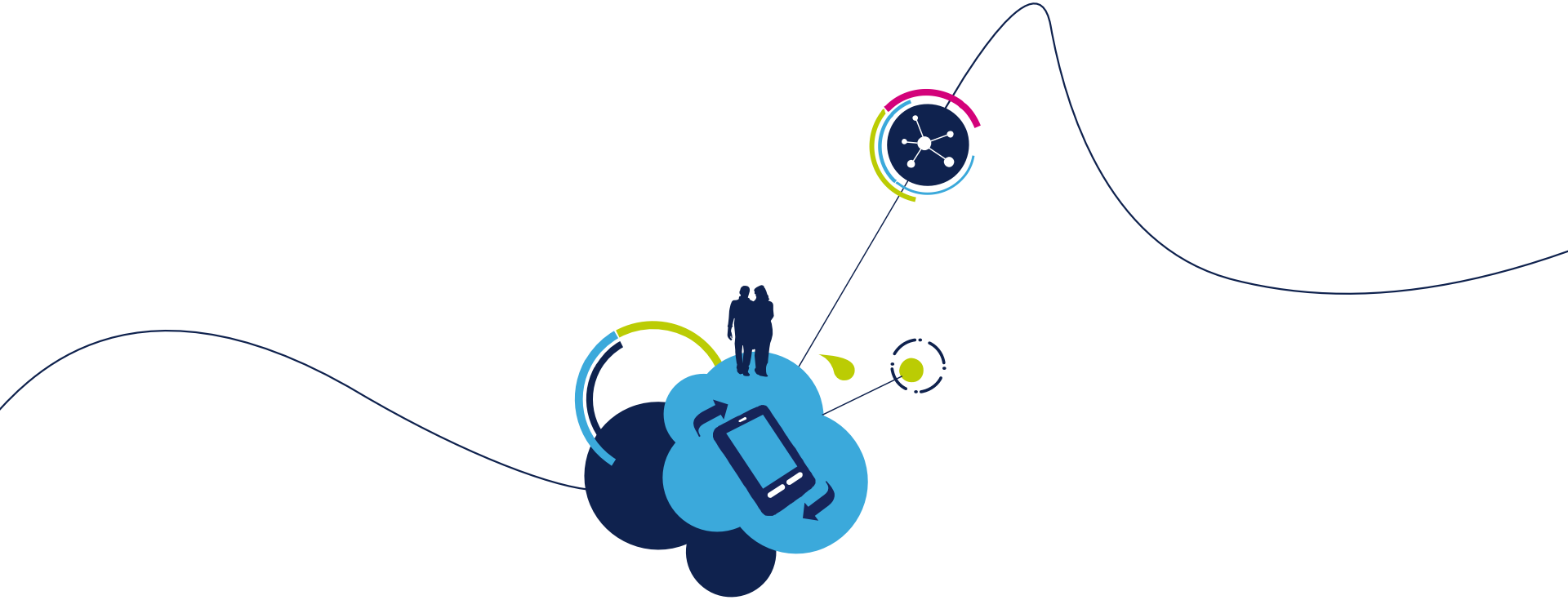
NOTES:

- Mind the trailing dash (-), it is mandatory, not a typo!
- The *ttySz* / *ttyACMx* numbers depends on the device enumeration of the Nucleo board running the Serial-sniffer firmware, you can use tab auto completion under both Linux and Cygwin
- It is mandatory to invoke the above commands from the "*serial-sniffer*" folder (actually, the "*header.pcap*" file is supposed to be in the same folder of the "*convert-to-binary*" script)
- *Perl* is needed in order for the "*convert-to-binary*" script to work
  - You can install it either via Cygwin setup or your Linux Package Manager
- *Wireshark* application is required, a recent version it is recommended in order to have state of the art protocols dissectors
  - In the above commands, *Wireshark* is supposed to be in System's Shell PATH, if it is not the case 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 char before spaces and parenthesis
- Serial-sniffer firmware must be compiled with the same radio (channel, modulation, …) settings as the network under investigation
- Under Windows you may need to hardcode the baudrate (115200) in the serialdump.c code and recompile

Serial Sniffer Example in a few steps (3/3)

7 Wireshark application should be running, packets seen on the air can be analyzed and saved





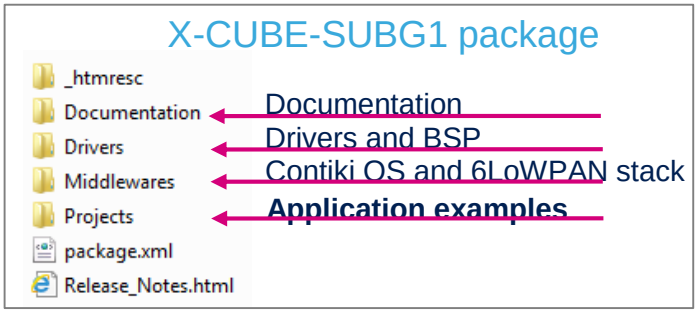
Demo Execution Using S2-LP

Start coding in just a few minutes with Contiki6LP

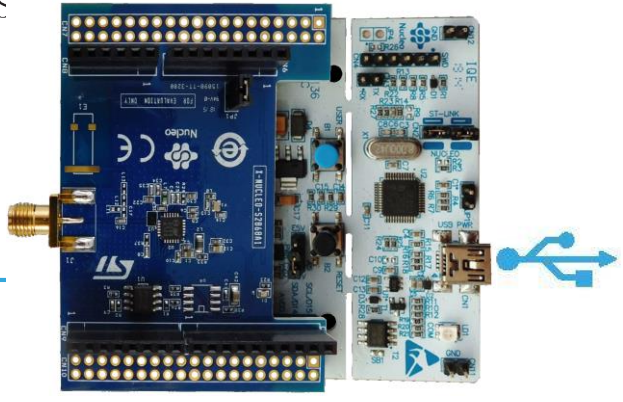
1 Go to www.st.com/x-nucleo

2 Select X-NUCLEO-S2868A1

3 Download and unpack X-CUBE-SUBG1

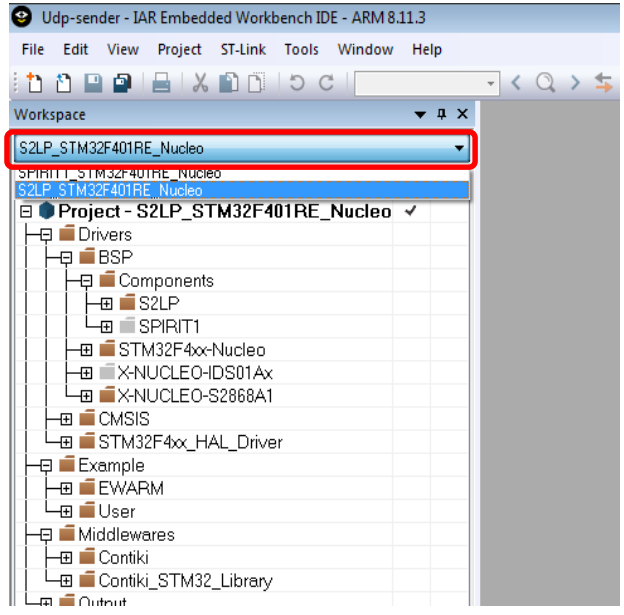


4 Download & install STM32 Nucleo ST-LINK/V2-1 USB driver



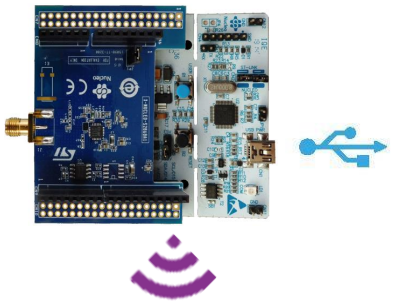
5 Open project example e.g. Udp-sender

6 Select the S2-LP radio configuration And then build and flash the application code.

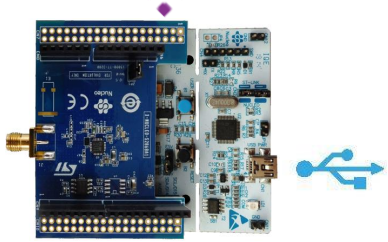


Demo Overview – UDP Sender and Receiver

6LoWPAN Udp-sender node
NUCLEO-L152RE or NUCLEO-F401RE
X-NUCLEO-S2868A1 or X-NUCLEO-S2915A1



UDP
Messages



6LoWPAN Udp-receiver node
NUCLEO-L152RE or NUCLEO-F401RE
X-NUCLEO-S2868A1 or X-NUCLEO-S2915A1

Host PC
with a terminal
Utility

USB1

USB2

UDP Sender and Receiver – S2LP examples in a few steps (1/2)

Download and extract **X-CUBE-SUBG1**

1

2

Compile the firmware for the UDP Receiver node:
Select the “**Udp-receiver**” application and build the Project using a supported IDE. Alternatively you can use a pre-built binary that is provided for running this application with the selected STM32 Nucleo board

3

Connect the STM32 Nucleo based kit acting as a “UDP Receiver” to a PC USB slot and program the device

Copy the binary file
(e.g. drag & drop) to the USB mass storage corresponding to the STM32 Nucleo board

4

Compile the firmware for the UDP sender node:
Select the “**Udp-sender**” application and build the Project using a supported IDE. Alternatively you can use a pre-built binary that is provided for running this application with the selected STM32 Nucleo board

5

Connect the STM32 Nucleo based kit acting as a “UDP Sender” to a PC USB slot and program the device

copy the file
(e.g. drag & drop) to the USB mass storage corresponding to the STM32 Nucleo board

UDP Sender and Receiver examples in a few steps (2/2)

Launch the terminal application and set the UART port to 115200 bps, 8 bit, No Parity, 1 stop bit

Select the device corresponding to the UDP sender node (e.g. on a Linux host, it will be a *ttyACMx* device type)

6

7

The terminal should be printing something like

```
COMPENSATE_TICKS:       48
RADIO_LOW_POWER:        1
RADIO_USES_CONTIKIMAC:  0
RADIO_USES_SNIFF_MODE:  1
RADIO_USES_LONG_PREAMBLE:       1

IPv6 addresses: aaaa::700:faff:7651:89ae
fe80::700:faff:7651:89ae
Service 190 not found
Service 190 not found
Service 190 not found
Service 190 not found
Service 190 not found
```

Udp-sender window

8

If everything has been done correctly, the output in the terminal should now be something similar to this:

9

Repeat step 6-8 for the project **Udp-receiver** (remember to open a new *terminal* window): The received UDP messages are shown

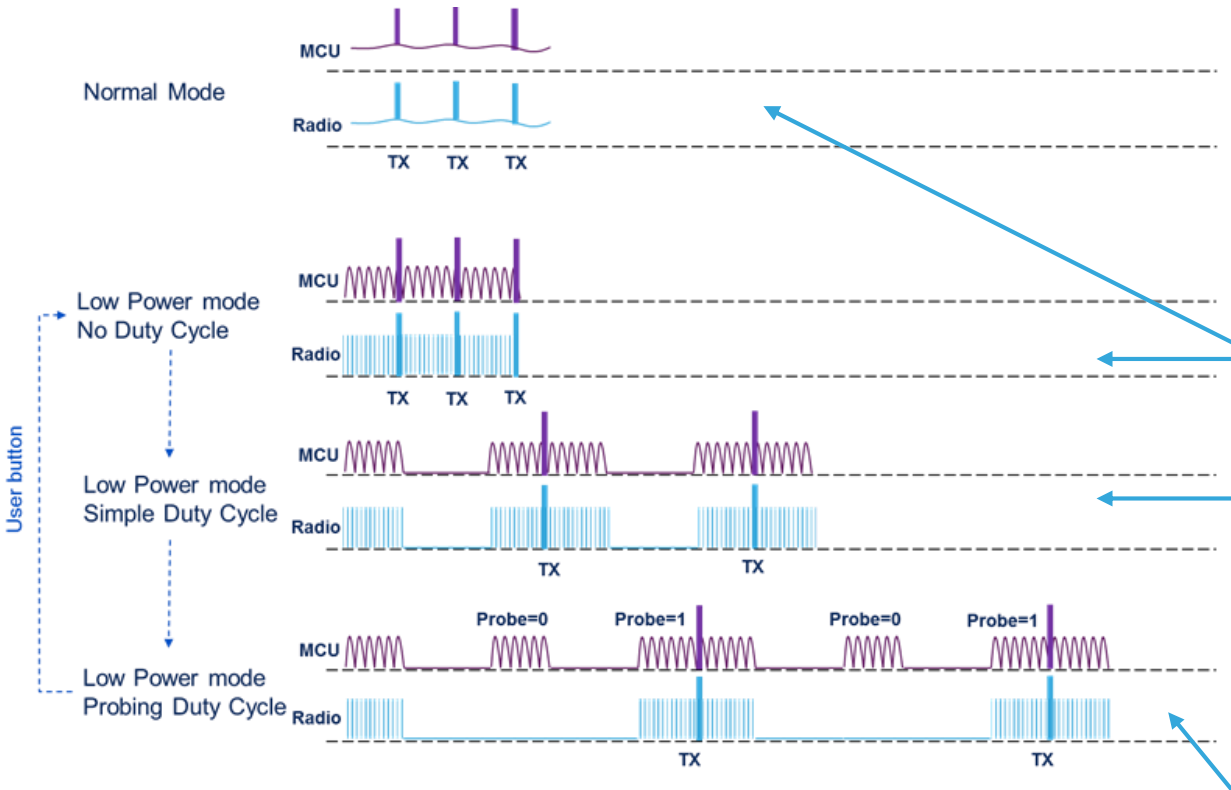
```
IPv6 addresses: aaaa::b00:f6ff:dca:f235
fe80::b00:f6ff:dca:f235
Data received from aaaa::700:faff:7651:89ae on port 1234 from port 1234 with length 14:
'[N] Message 0'
Data received from aaaa::700:faff:7651:89ae on port 1234 from port 1234 with length 14:
'[N] Message 1'
Data received from aaaa::700:faff:7651:89ae on port 1234 from port 1234 with length 14:
'[N] Message 2'
Data received from aaaa::700:faff:7651:89ae on port 1234 from port 1234 with length 14:
'[N] Message 3'
Data received from aaaa::700:faff:7651:89ae on port 1234 from port 1234 with length 14:
'[N] Message 4'
Data received from aaaa::700:faff:7651:89ae on port 1234 from port 1234 with length 14:
'[N] Message 5'
Data received from aaaa::700:faff:7651:89ae on port 1234 from port 1234 with length 14:
'[N] Message 6'
```

Udp-receiver window

```
IPv6 addresses: aaaa::700:faff:7651:89ae
fe80::700:faff:7651:89ae
Service 190 not found
Service 190 not found
Sending message 0 to aaaa::b00:f6ff:dca:f235
Sending message 1 to aaaa::b00:f6ff:dca:f235
Sending message 2 to aaaa::b00:f6ff:dca:f235
Sending message 3 to aaaa::b00:f6ff:dca:f235
Sending message 4 to aaaa::b00:f6ff:dca:f235
Sending message 5 to aaaa::b00:f6ff:dca:f235
Sending message 6 to aaaa::b00:f6ff:dca:f235
```

Udp-sender window

Demo Overview – Low Power features



When using NUCLEO-L152RE and X-NUCLEO-S2868A1 Low Power features are available for both MCU and Radio.

The following settings are enabled by default on Udp-sender and Udp-receiver applications:

- MCU System Clock is set to 4 MHz
- MCU is sent to SLEEP mode (with reduced clock to 65 KHz) when in idle loop
- Radio uses SNIFF mode

Compared to Normal Mode, these settings ensure the same functionalities but with almost 1/20 of the current consumption in idle.

By pressing the User Button, it is possible to cycle through two Application Level Duty Cycle that are implemented as a demo in the Udp-sender firmware. The idea is to demonstrate that it is also possible to switch “off” (actually, set to a very low consumption state) both MCU and Radio for a given period. The node is then waken up by a timer, sends the next message and goes back to STOP mode.

In the “Probing” Duty Cycle, the Radio is turned on only if there is something to send (in the demo, this happens every two periods).

The Duty Cycle phase is reported with a character in the message sent by the Udp-sender, and can be checked on the Udp-receiver console.

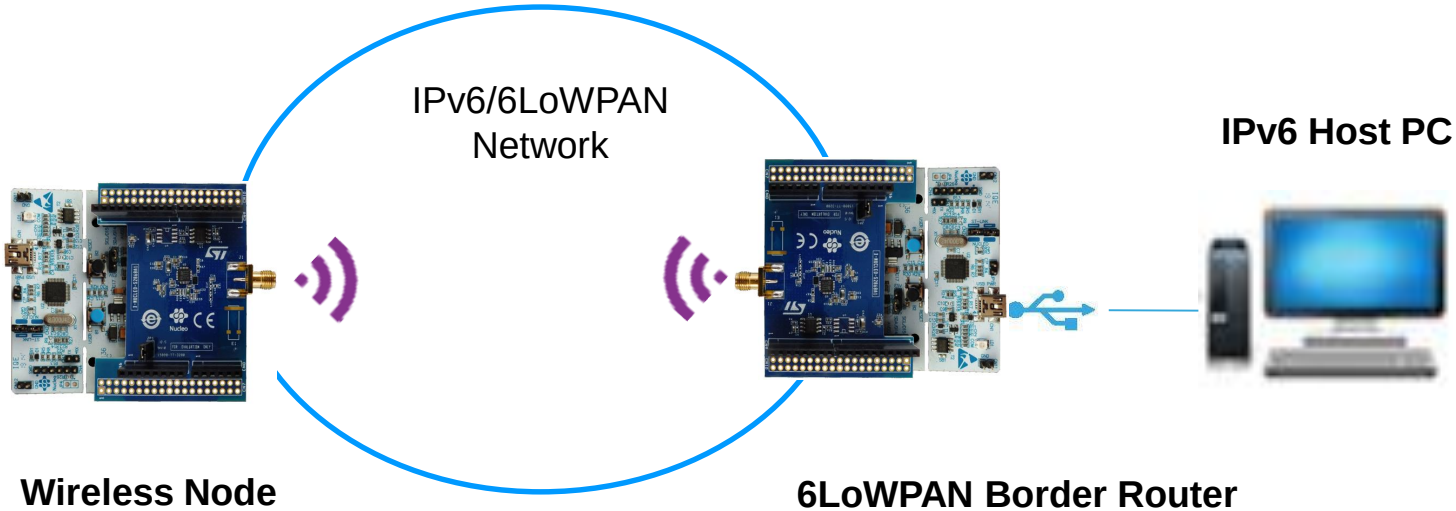
```
Duty cycle is now PROBING.
Sending message 637 to aaaa::b00:f6ff:dca:f235
PROBING Duty Cycle, going in stop mode.
Entering STOP mode for about 10 seconds.
No data to send, will keep probing with radio off.
PROBING Duty Cycle, going in stop mode.
Entering STOP mode for about 10 seconds.
Found some data to send.
Sending message 638 to aaaa::b00:f6ff:dca:f235
PROBING Duty Cycle, going in stop mode.
Entering STOP mode for about 10 seconds.
```

Udp-sender window

```
'[N] Message 617'
Data received from aaaa::700:faff:7
'[S] Message 618'
Data received from aaaa::700:faff:7
'[S] Message 619'
Receiver listening for messages.
Data received from aaaa::700:faff:7
'[S] Message 620'
Receiver listening for messages.
Data received from aaaa::700:faff:7
'[P] Message 621'
```

Udp-receiver window

Demo Overview – Border Router Example



Border Router For S2-LP Example in a few steps (1/3)

Download and extract **X-CUBE-SUBG1**

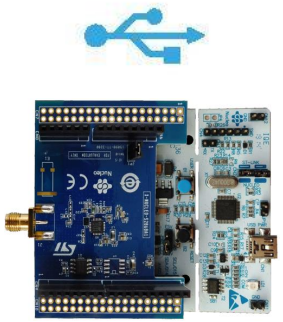
1

2

Compile the firmware for a wireless node:
Select the “**Udp-sender**” application and build the Project using a supported IDE. Alternatively you can use a pre-built binary that is provided for running this application with the selected STM32 Nucleo board

3

Connect the board to a
PC USB slot and program the device



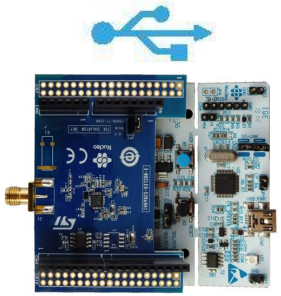
Copy the binary file
(e.g. drag & drop) to the USB mass storage corresponding to the STM32 Nucleo board

4

Compile the firmware for the border router node:
Select the “**Border-router**” application and build the Project using a supported IDE. Alternatively you can use a pre-built binary that is provided for running this application with the selected STM32 Nucleo board

5

Connect the board to USB and program the device



copy the file
(e.g. drag & drop) to the USB mass storage corresponding to the STM32 Nucleo board

Border Router Example in a few steps (2/3)

6 Setup the IPv6 Host PC for IP traffic bridging between host and 6LowPAN border Router

From: *[ROOT]/Middlewares/Third_Party/Contiki*

Windows PC setup (Win 7/8)
using "**wpcapslip6**" utility

1. wpcapslip6 needs a working network adapter:
The Microsoft loopback adapter can be installed via "Add legacy hardware" in the Windows Device Manager (reboot is needed after installation of the loopback adapter)
2. Copy "cygwin1.dll" from "tools/cygwin" to wcapslip6 folder
3. Install WinPcaP
4. run Cygwin as administrator

OR

Linux PC setup (Ubuntu)
using "**tunslip6**" utility

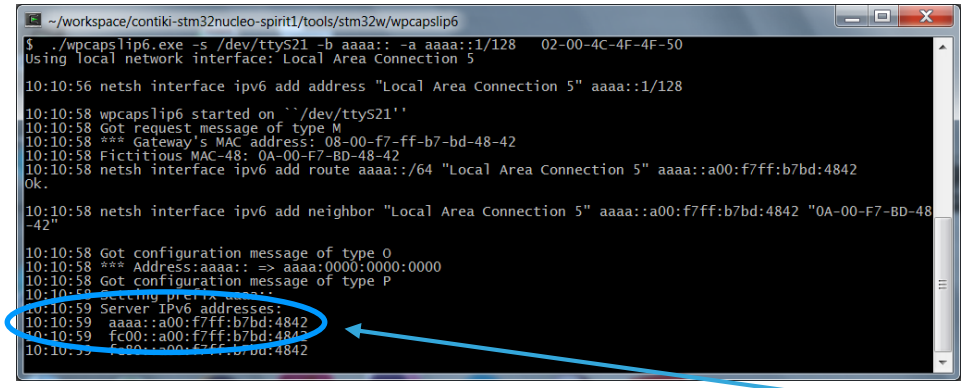
```
cd ./tools
make tunslip6
sudo ./tunslip6 –s /dev/ttyACMx aaaa::1/64
```

ttySz / ttyACMx depends on the device enumeration, you can use tab auto completion under both Linux and Cygwin

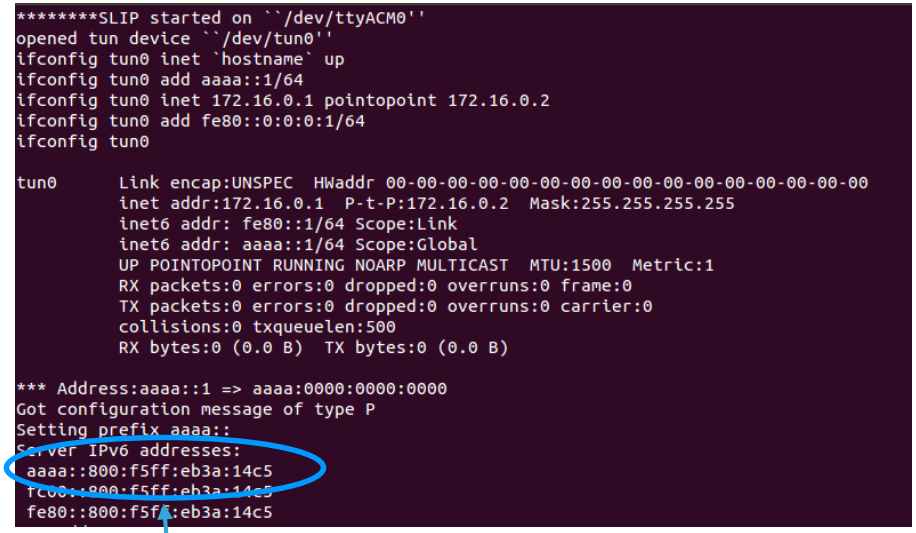
wpcapslip6 utility can then be used with the rpl-border-router example

```
cd ./tools/stm32w/wpcapslip6
./wpcapslip6 –s /dev/ttySz –b aaaa:: -a aaaa::1/128 [addr]
```

Where *[addr]* is the MAC address of the local net adapter



wpcapslip6 terminal window output



Tunslip6 terminal window output

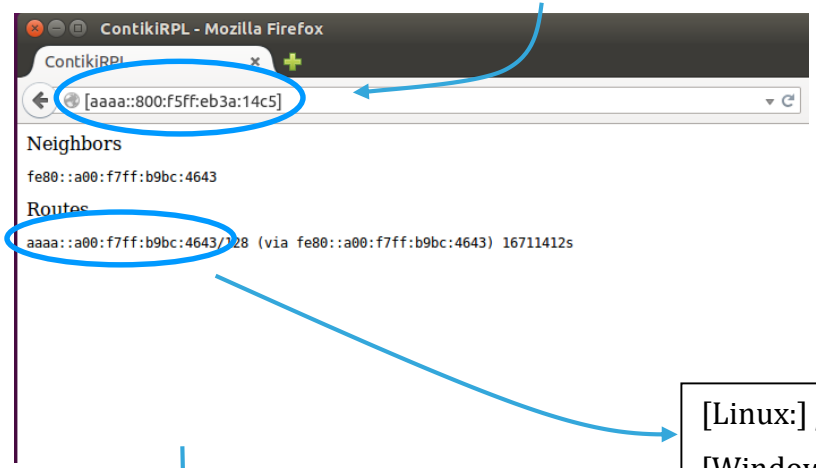
Contiki server address (used in the next step)

life.augmented

Border Router Example in a few steps (3/3)

7 Open a Web browser (Firefox) to access the Contiki server providing the RPL neighbors and routes information.

Contiki server address (see previous step) between brackets, e.g. [aaaa::800:f5ff:eb3a:14c5]



[Linux:] *ping6 aaaa::a00:f7ff:b9bc:4643*

[Windows:] *ping -6 aaaa::a00:f7ff:b9bc:4643*

8 Ping the wireless Node to test the 6LoWPAN connectivity

```
PING aaaa::a00:f7ff:b9bc:4643(aaaa::a00:f7ff:b9bc:4643) 56 data bytes
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=1 ttl=63 time=70.0 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=2 ttl=63 time=70.7 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=3 ttl=63 time=76.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=4 ttl=63 time=65.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=5 ttl=63 time=72.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=6 ttl=63 time=67.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=7 ttl=63 time=74.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=8 ttl=63 time=68.9 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=9 ttl=63 time=75.9 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=10 ttl=63 time=64.9 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=11 ttl=63 time=65.9 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=12 ttl=63 time=72.9 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=13 ttl=63 time=67.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=14 ttl=63 time=74.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=15 ttl=63 time=69.8 ms
64 bytes from aaaa::a00:f7ff:b9bc:4643: icmp_seq=16 ttl=63 time=70.8 ms
^C
--- aaaa::a00:f7ff:b9bc:4643 ping statistics ---
16 packets transmitted, 16 received, 0% packet loss, time 15017ms
rtt min/avg/max/mdev = 64.936/70.685/76.827/3.620 ms
fabien@marco-linux-HP:~$
```

Demo Overview – Serial Sniffer Example

Wireless Nodes
STM32 Nucleo boards with the X-NUCLEO-S2868A1

IPv6/6LoWPAN Network

Serial Sniffer
STM32 Nucleo board with the X-NUCLEO-S2868A1

PC running Wireshark

Serial Sniffer Example in a few steps (1/3)

Download and extract **X-CUBE-SUBG1**

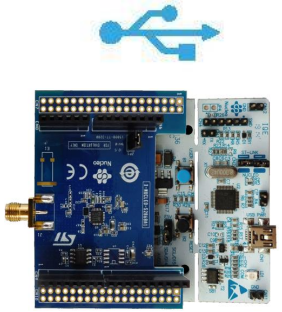
1

2

Compile the firmware for the Serial Sniffer:
Select the "**Serial-sniffer**" application and build the Project using a supported IDE. Alternatively you can use a pre-built binary that is provided for running this application with the selected STM32 Nucleo board

3

Connect the board to a
PC USB slot and program the device



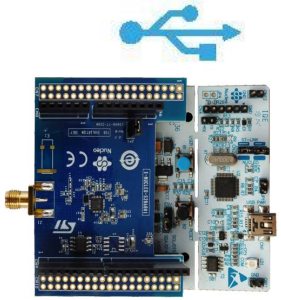
Copy the binary file
(e.g. drag & drop) to the USB mass storage corresponding to the STM32 Nucleo board

4

A pre-requisite to use the Serial Sniffer is a running 6LoWPAN network, you can refer to the example UDP Sender and Receiver described in previous slides, i.e. select "**Udp-sender**" and "**Udp-receiver**" applications and build the Projects using a supported IDE. Alternatively you can use the pre-built binaries that are provided for running these applications with the selected STM32 Nucleo board

5

Connect the board to USB and
program the device (repeat this for both firmwares)



copy the file
(e.g. drag & drop) to the USB mass storage corresponding to the STM32 Nucleo board

Serial Sniffer Example in a few steps (2/3)

6 Setup the PC running Wireshark application

From: *[ROOT]/Utilities/serial-sniffer*

Windows PC setup (Win 7/8)
using "**serialdump-windows.exe**" utility

1. run Cygwin as administrator
2. serialdup-windows.exe utility is provided pre-compiled, but in case you need to recompile it:
   - *cd serialdump-src*
   - *make*
     - (or *gcc –o serialdump-windows.exe serialdump.c*)
   - *mv serialdump-windows.exe ..*
3. Run the following command chain (it is ONE line of three commands in pipe "|" one with the next one)

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

OR

Linux PC setup (Ubuntu)
using "**serialdump-linux**" utility

1. serialdump-linux utility has to be compiled:
   - *cd serialdump-src*
   - *make*
     - (or *gcc –o serialdump-linux serialdump.c*)
   - *mv serialdump-linux ..*
2. Run the following command chain (it is ONE line of three commands in pipe "|" one with the next one)

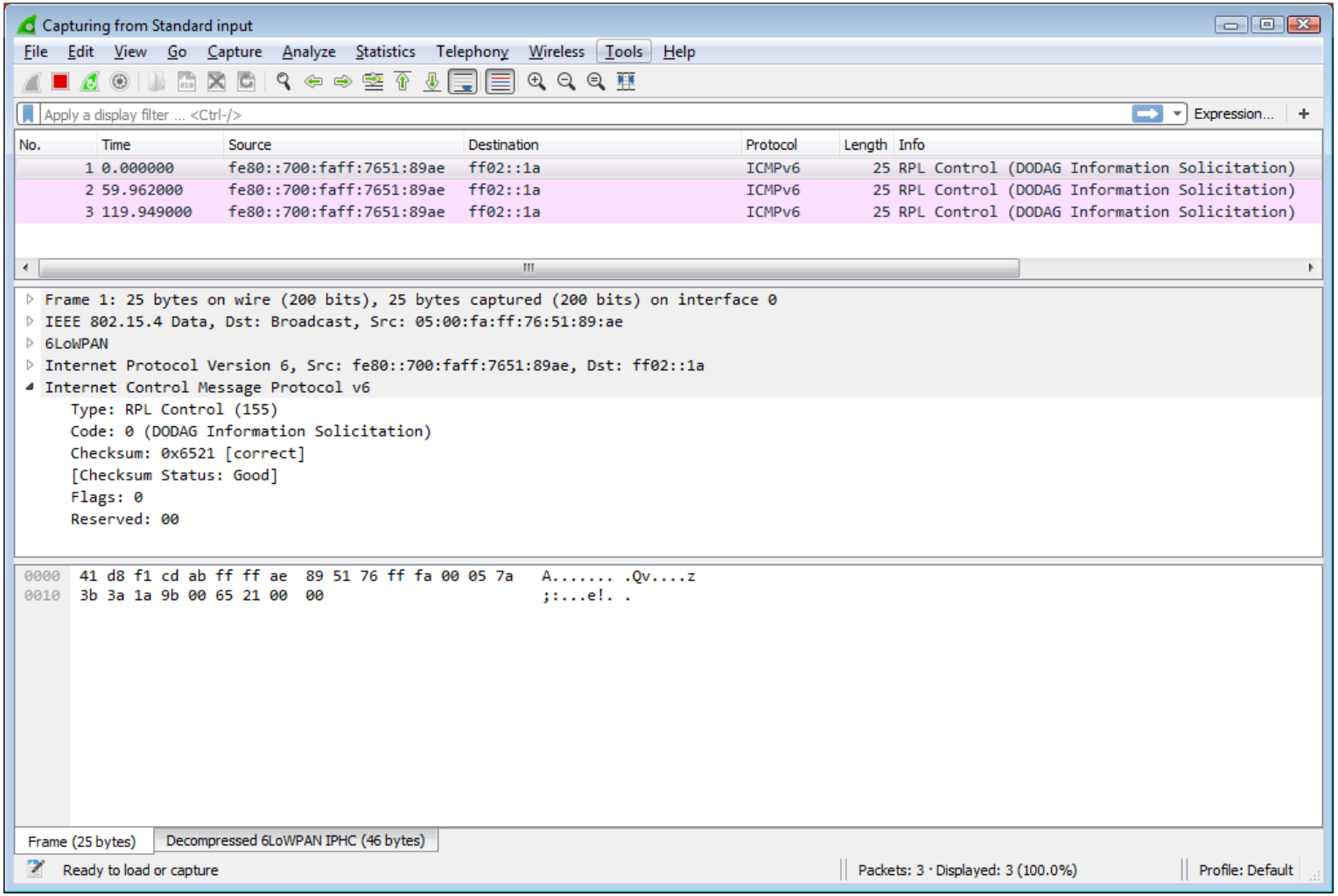
> sudo serialdump-linux -b115200 /dev/ttyACMx | ./convert-to-binary | wireshark -k -i -

NOTES:

- Mind the trailing dash (-), it is mandatory, not a typo!
- The *ttySz* / *ttyACMx* numbers depends on the device enumeration of the Nucleo board running the Serial-sniffer firmware, you can use tab auto completion under both Linux and Cygwin
- It is mandatory to invoke the above commands from the "*serial-sniffer*" folder (actually, the "*header.pcap*" file is supposed to be in the same folder of the "*convert-to-binary*" script)
- *Perl* is needed in order for the "*convert-to-binary*" script to work
  - You can install it either via Cygwin setup or your Linux Package Manager
- *Wireshark* application is required, a recent version it is recommended in order to have state of the art protocols dissectors
  - In the above commands, *Wireshark* is supposed to be in System's Shell PATH, if it is not the case 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 char before spaces and parenthesis
- Serial-sniffer firmware must be compiled with the same radio (channel, modulation, …) settings as the network under investigation
- Under Windows you may need to hardcode the baudrate (115200) in the serialdump.c code and recompile

Serial Sniffer Example in a few steps (3/3)

7 Wireshark application should be running, packets seen on the air can be analyzed and saved



Documents & Related Resources

All documents are available in the DESIGN tab of the related products webpage

X-CUBE-SUBG1:

- **DB2556**: Sub-1 GHz RF communication software expansion for STM32Cube **– data brief**
- **UM1904**: Getting started with the software package for Point-to-Point communications using SPIRIT1 sub-1GHz modules in X-CUBE-SUBG1, Expansion for STM32Cube **– user manual**
- **UM2040:** Getting started with Contiki6LP, Contiki OS and 6LoWPAN sub-1GHz RF communications software expansion for STM32Cube – **user manual**

X-NUCLEO-IDS01A4:

- Gerber files, BOM, Schematic
- **DB2552:** Sub-1 GHz RF expansion board based on the SPSGRF-868 module for STM32 Nucleo – **data brief**
- **UM1872:** Getting started with the Sub-1 GHz expansion board based on SPSGRF-868 and SPSGRF-915 modules for STM32 Nucleo – **user manual**

X-NUCLEO-IDS01A5:

- Gerber files, BOM, Schematic
- **DB2553:** Sub-1 GHz RF expansion board based on SPSGRF-915 module for STM32 Nucleo – **data brief**
- **UM1872:** Getting started with the Sub-1 GHz expansion board based on SPSGRF-868 and SPSGRF-915 modules for STM32 Nucleo – **user manual**

X-NUCLEO-S2868A1 :

- Gerber files, BOM, Schematic
- **DB3602:** Sub-1 GHz RF expansion board based on S2-LP radio for STM32 Nucleo – **data brief**
- **UM2405**: Getting started with the X-NUCLEO-S2868A1 Sub-1 GHz 868 MHz RF expansion board based on S2-LP radio for STM32 Nucleo

Consult www.st.com for the complete list

Quick Start Guide Contents

Contiki6LP: Contiki OS/6LoWPAN and sub-1GHz RF communication Hardware and Software overview

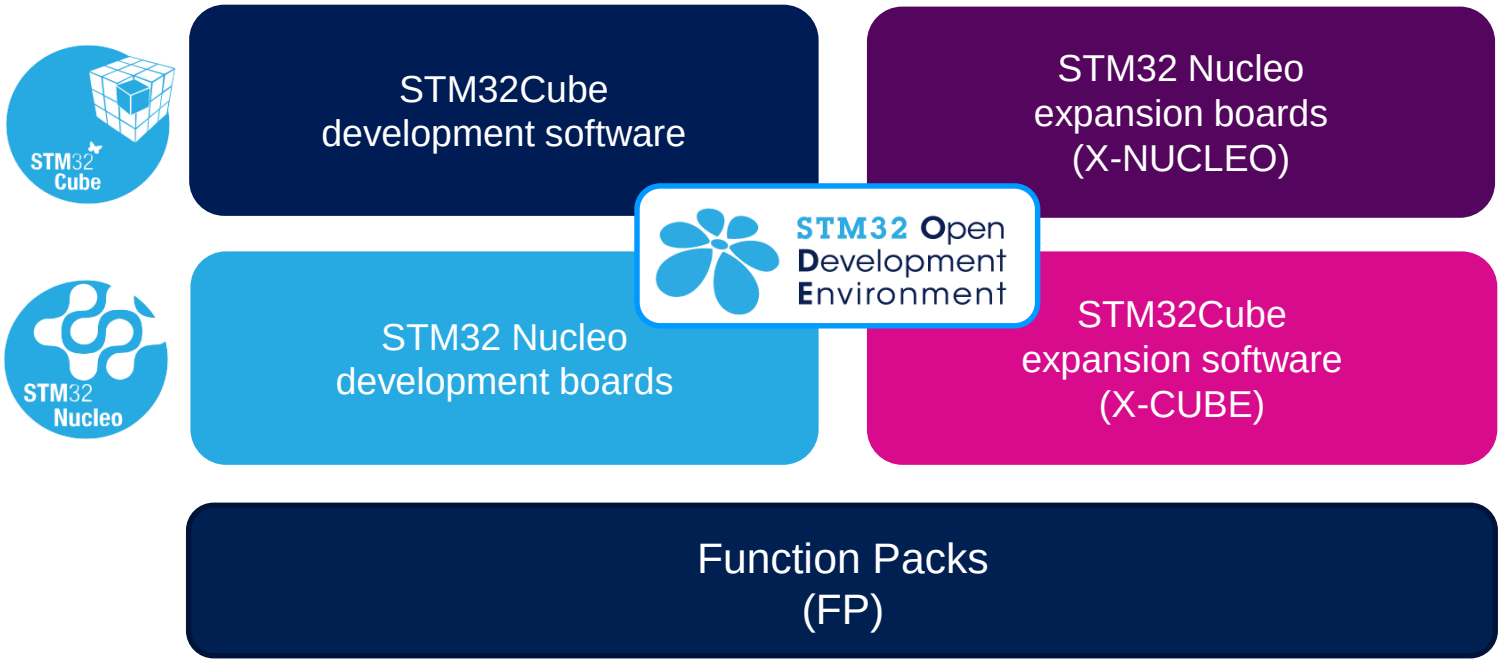
Setup & Demo Examples

Documents & Related Resources

STM32 Open Development Environment: Overview

STM32 Open Development Environment
Fast, affordable Prototyping and Development

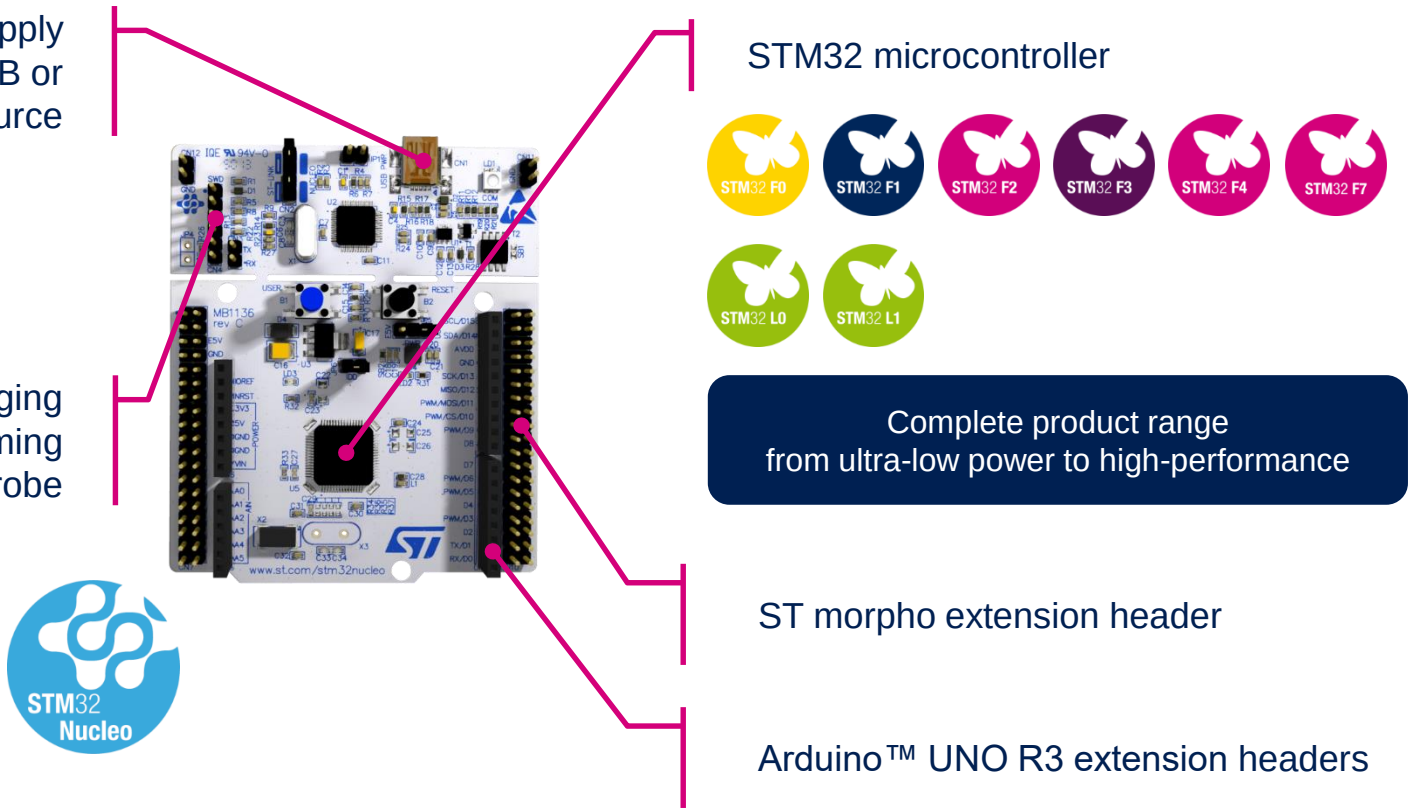
- The STM32 Open Development Environment (ODE) consists of a set of stackable boards and a modular open SW environment designed around the STM32 microcontroller family.



www.st.com/stm32ode

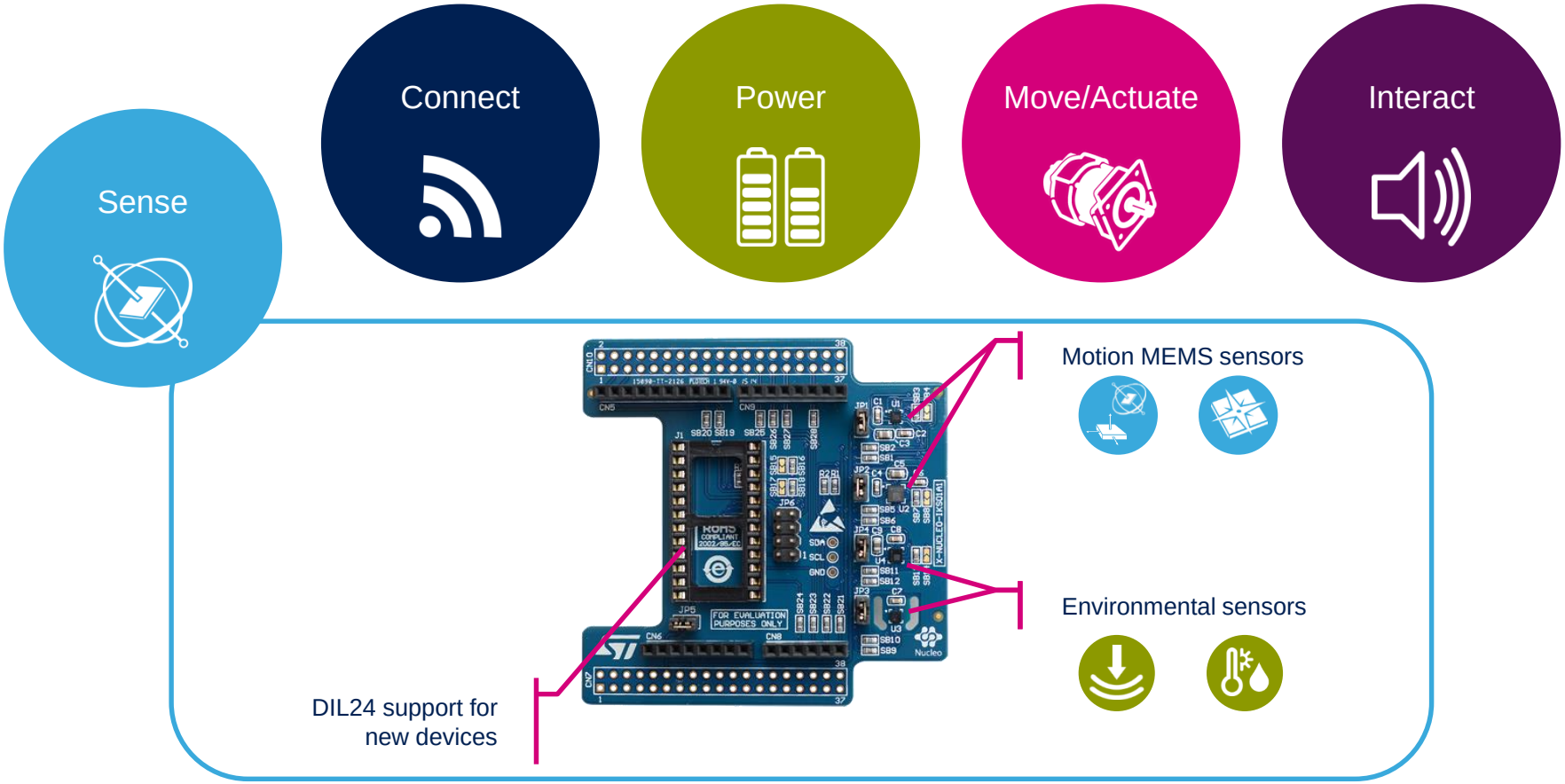
Development Boards (NUCLEO)

- A comprehensive range of affordable development boards for all the STM32 microcontroller series, with unlimited unified expansion capabilities and integrated debugger/programmer functionality.



Expansion Boards (X-NUCLEO)

- Boards with additional functionality that can be plugged directly on top of the STM32 Nucleo development board directly or stacked on another expansion board.



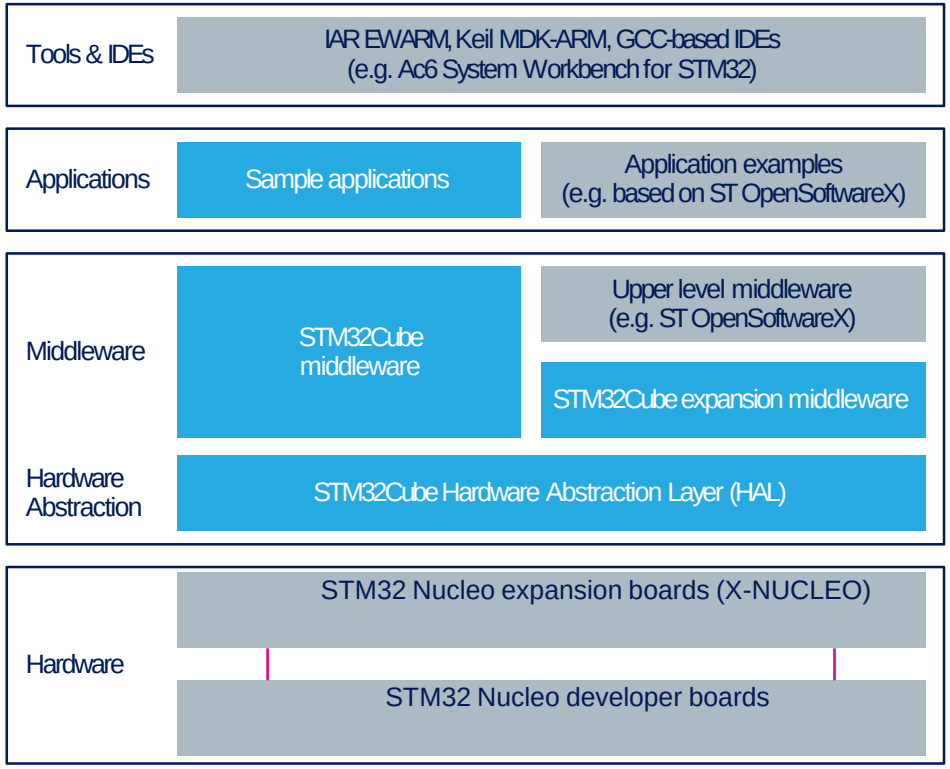
Example of STM32 expansion board (X-NUCLEO-IKS01A1)

STM32 Open Development Environment

Software components

- **STM32Cube software (CUBE)** - A set of free tools and embedded software bricks to enable fast and easy development on the STM32, including a Hardware Abstraction Layer and middleware bricks.

- **STM32Cube expansion software (X-CUBE)** - Expansion software provided free for use with the STM32 Nucleo expansion board and fully compatible with the STM32Cube software framework. It provides abstracted access to expansion board functionality through high-level APIs and sample applications.



- **Compatibility with multiple Development Environments -** The STM32 Open Development Environment is compatible with a number of IDEs including IAR EWARM, Keil MDK, and GCC-based environments. Users can choose from three IDEs from leading vendors, which are free of charge and deployed in close cooperation with ST. These include Eclipse-based IDEs such as Ac6 System Workbench for STM32 and the MDK-ARM environment.

www.st.com/stm32cube

www.st.com/x-cube

STM32 Open Development Environment

Building block approach

The building blocks	Your need	Our answer

Accelerometer, gyroscope
Inertial modules, magnetometer
Pressure, temperature, humidity
Proximity, microphone

Sense — COLLECT

Bluetooth LE, Sub-GHz radio
NFC, Wi-Fi, GNSS

Connect — TRANSMIT

Audio amplifier
Touch controller
Operation Amplifier

Translate — ACCESS

Stepper motor driver
DC & BLDC motor driver
Industrial input / output

Move / Actuate — CREATE

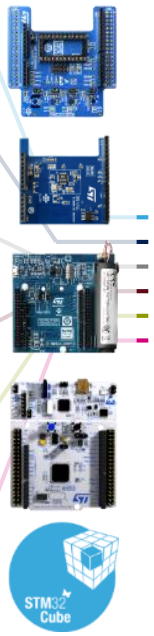
Energy management & battery

Power — POWER

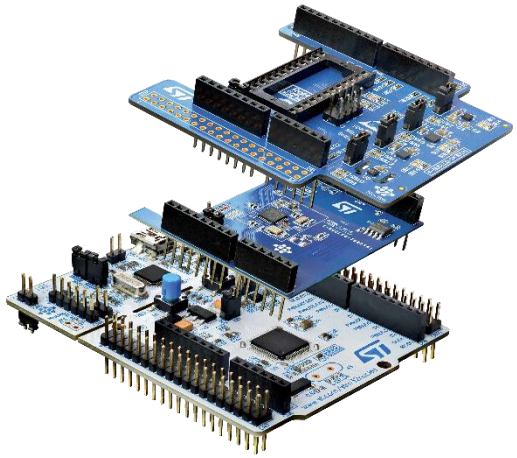
General-purpose microcontrollers
Secure microcontrollers

Process — PROCESS

Software



STM32 Open Development Environment



www.st.com/stm32ode