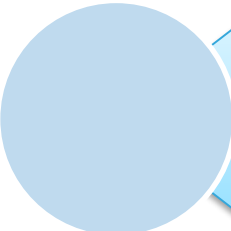




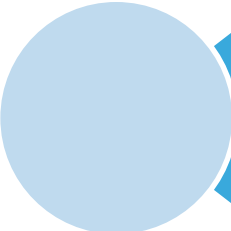
Quick Start Guide

Contiki OS and 6LoWPAN sub-1GHz RF communication software expansion for STM32 Cube (osxContiki6LP)





osxContiki6LP: Contiki OS/6LoWPAN and sub-1GHz RF communication
Hardware and Software overview



Setup & Demo Examples
Documents & Related Resources



STM32 Open Development Environment: Overview

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

Contiki OS/6LoWPAN and sub-1GHz RF communication (osxContiki6LP)

Software Overview

4

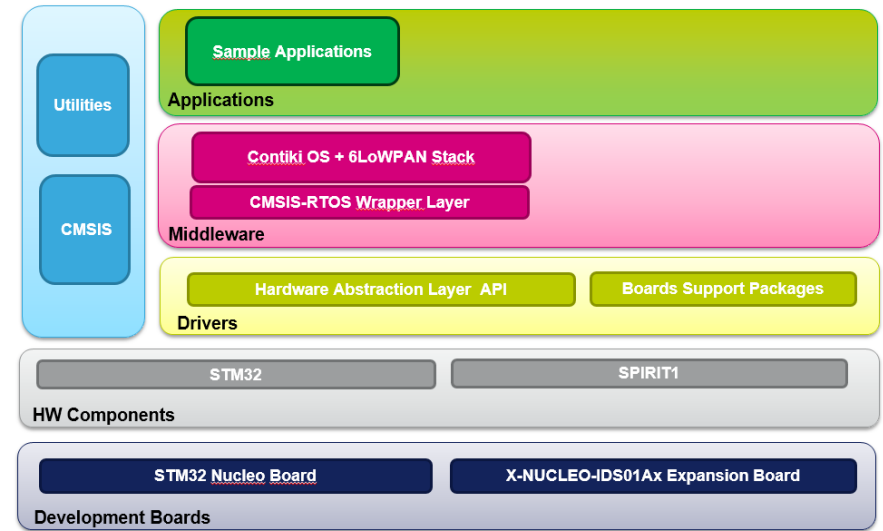
osxContiki6LP Software Description

OsxContiki6LP is a software package that expands the functionality provided by STM32Cube. The library is implemented as a middleware ready to be integrated in projects based on STM32Cube and X-CUBE-SUBG1 expansion software. OsxContiki6LP requires, in fact, the X-CUBE-SUBG1 (v1.1.0 or higher) package to work. 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 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

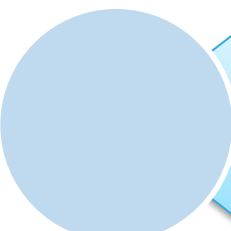
Overall Software Architecture



Latest info available at www.st.com
osxContiki6LP



osxContiki6LP: 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

15

- STM32 Nucleo development board
NUCLEO-L152RE or **NUCLEO-F401RE**
- Sub-1GHz RF expansion board for STM32 Nucleo based on the SPSGRF-868 module (**X-NUCLEO-IDS01A4**) or the on the SPSGRF-915 module (**X-NUCLEO-IDS01A5**)
- Windows/Linux PC
- mini USB cable



Setup & Demo Examples

SW prerequisites

16

- X-CUBE-SUBG1 package
 - Download and extract the **X-CUBE-SUG1** package, version 1.1.0 or higher
- osxContiki6LP package
 - Download and extract the **osxContiki6LP** package
 - Open the extracted folder and copy all sub-folders to the X-CUBE-SUBG1 path (merging of the two packages)
- A toolchain to build the firmware
 - The port 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

Start coding in just a few minutes with osxContiki6LP

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



2 Select
X-NUCLEO-IDS01A4,
X-NUCLEO-IDS01A5



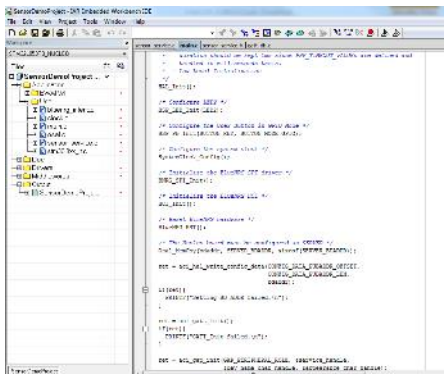
3
Download, unpack
and merge
osxContiki6LP
with
X-CUBE-SUBG1

osxContiki6LP + X-CUBE-SUBG1 packages

_htmresc	
Documentation	Documentation
Drivers	Drivers and BSP
Middlewares	Contiki OS and 6LoWPAN stack
Projects	Application examples
package.xml	
Release_Notes.html	

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

6
Modify and build application



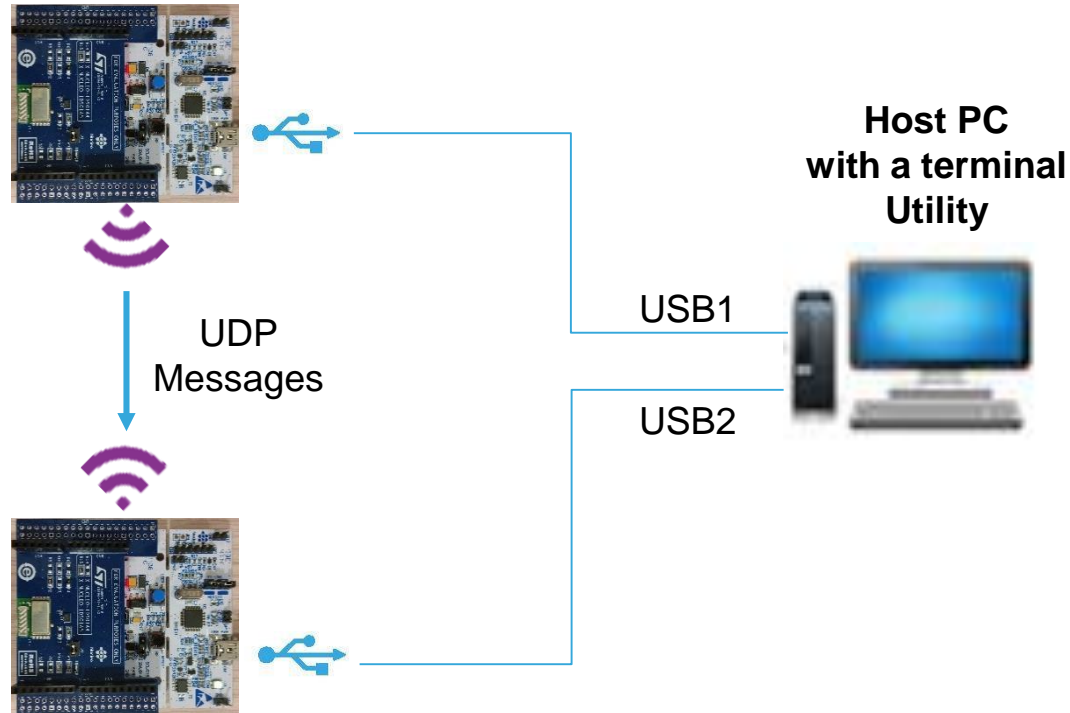
5
Open project example
e.g. Udp-sender



Demo Overview – UDP Sender and Receiver

18

6LoWPAN Udp-sender node
NUCLEO-L152RE or NUCLEO-F401RE
X-NUCLEO-IDS01A4/5



6LoWPAN Udp-receiver node
NUCLEO-L152RE or NUCLEO-F401RE
X-NUCLEO-IDS01A4/5

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

Download and extract **osxContiki6LP**
 Download and extract **X-CUBE-SUBG1**
Merge osxContiki6LP with X-CUBE-SUBG1 folders

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

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

4

3

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

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



5



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

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)

20

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)

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

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

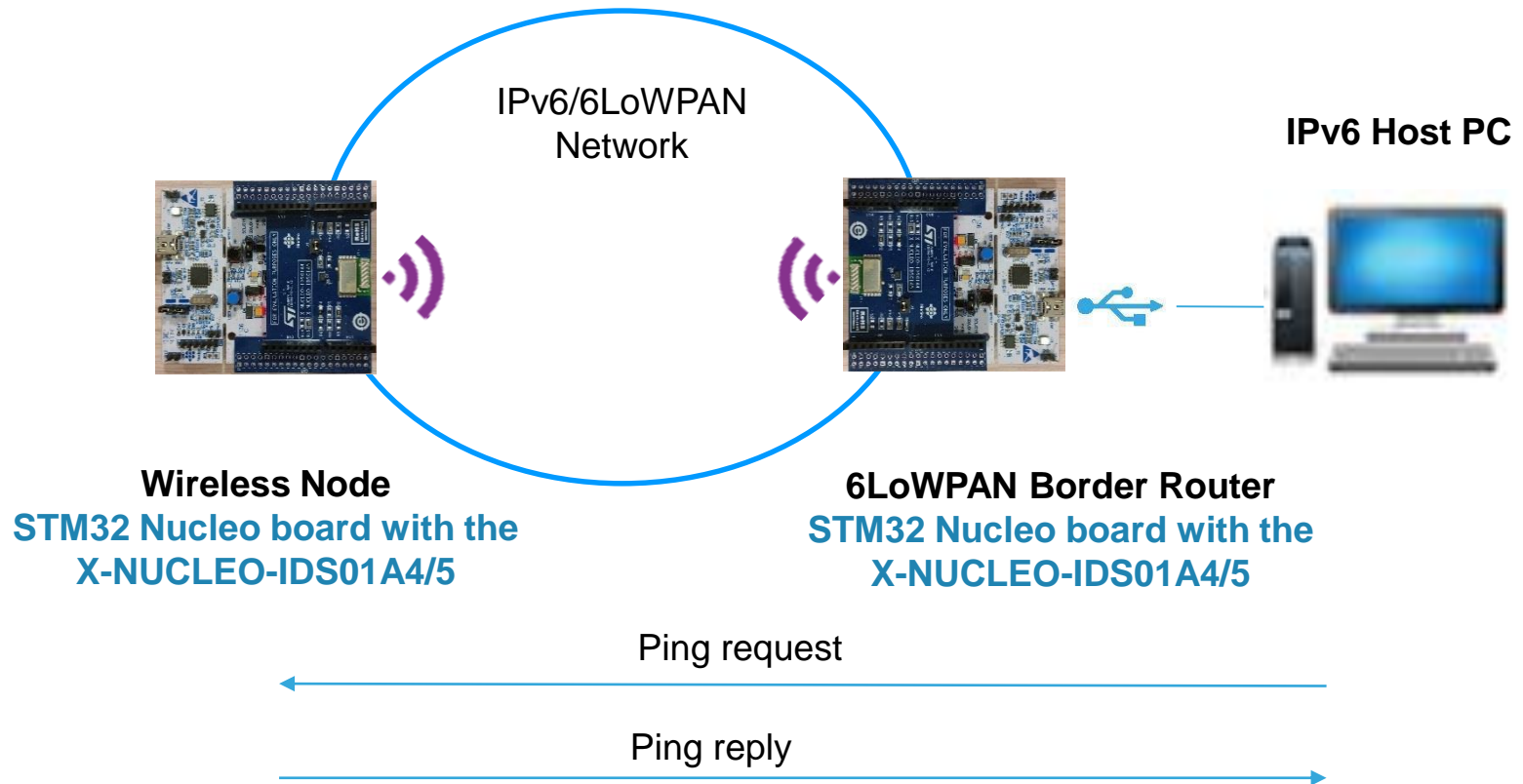
If everything has been done correctly, the output in the terminal should now be something similar to this:

```
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

Demo Overview – Border Router Example

21



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

11

Download and extract **osxContiki6LP**
Download and extract **X-CUBE-SUBG1**
Merge osxContiki6LP with X-CUBE-SUBG1 folders

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

Connect the board to USB and
program the device



5
copy the file
(e.g. drag & drop) to the USB mass storage
corresponding to the STM32 Nucleo board

6

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

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

22

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



Linux PC setup (Ubuntu)
using "tunslip6" utility

OR

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 "contiki/tools/cygwin" to wpcapslip6 folder
3. Install WinPcap
4. run Cygwin as administrator

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

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

```
cd ./tools/stm32w/wpcapslip6
./wpcapslip6 -s /dev/ttyS21 -b aaaa:: -a aaaa::1/128 [addr]
```

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

```
~/workspace/contiki-stm32nucleo-spirit1/tools/stm32w/wpcapslip6
$ ./wpcapslip6.exe -s /dev/ttyS21 -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/ttyS21"
10:10:58 Got request message of type M
10:10:58 *** Gateway's MAC address: 08-00-f7-ff-bd-bd-48-42
10:10:58 Fictitious MAC-48: 0A-00-F7-BD-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-BD-48-42"
10:10:58 Got configuration message of type O
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
```

wpcapslip6 terminal window output

```
*****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
          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)

*** Address:aaaa::1 => aaaa:0000:0000:0000
Got configuration message of type P
Setting prefix aaaa::
Server IPv6 addresses:
aaaa::800:f5ff:eb3a:14c5
fc00::800:f5ff:eb3a:14c5
fe80::800:f5ff:eb3a:14c5
```

Tunslip6 terminal window output

Contiki server address (used in the next step)

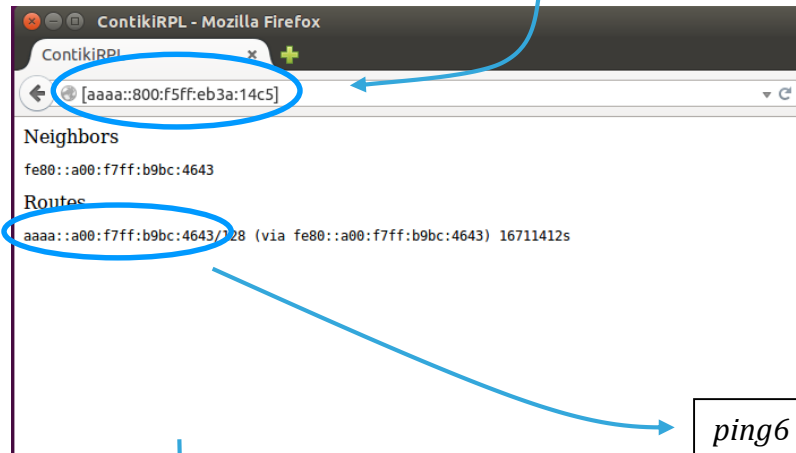
Border Router Example in a few steps (3/3)

23

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]



`ping6 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:~$
```

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

osxCONTIKI6LP:

- **DB2884:** Contiki OS/6LoWPAN middleware add-on for X-CUBE-SUBG1 expansion for STM32Cube – **data brief**
- **UM2040:** Getting started with osxContiki6LP, Contiki OS and 6LoWPAN sub-1GHz RF communications software expansion for STM32Cube – **user manual**

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**

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**

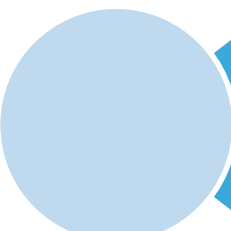
X-NUCLEO-IDS01A5:

- 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**

Consult www.st.com for the complete list



osxContiki6LP: Contiki OS/6LoWPAN and sub-1GHz RF communication
Hardware and Software overview



Setup & Demo Examples
Documents & Related Resources



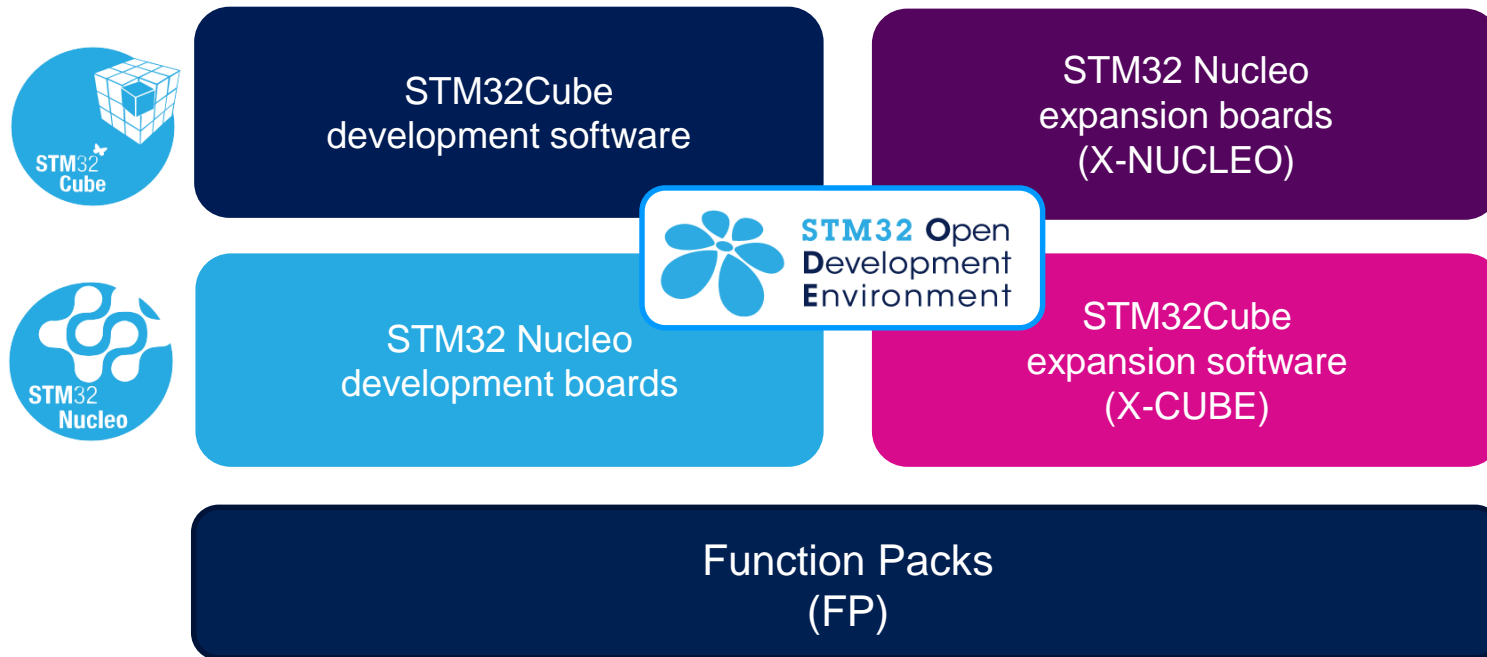
STM32 Open Development Environment: Overview

STM32 Open Development Environment

Fast, affordable Prototyping and Development

18

- 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.

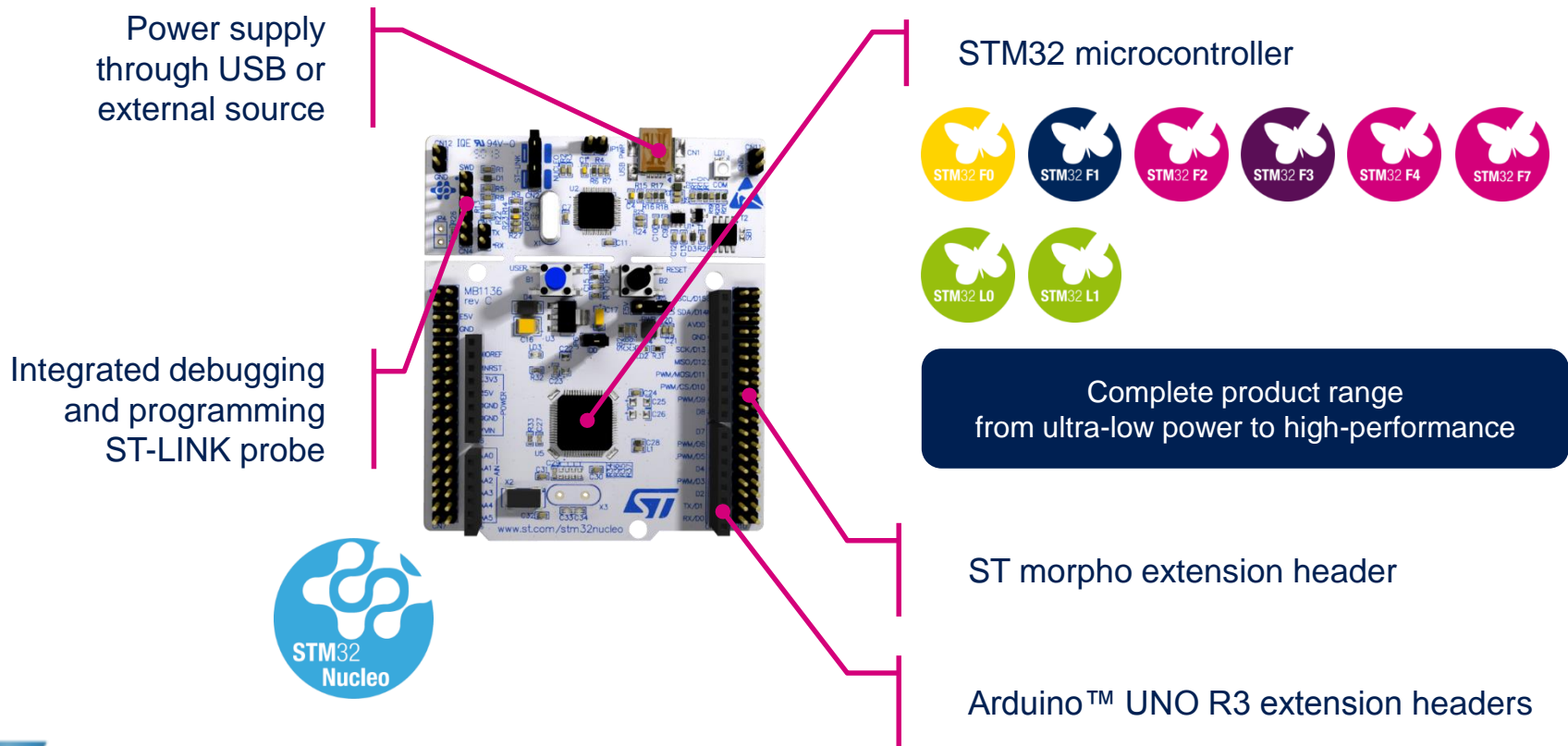


www.st.com/stm32ode

STM32 Nucleo Development Boards (NUCLEO)

19

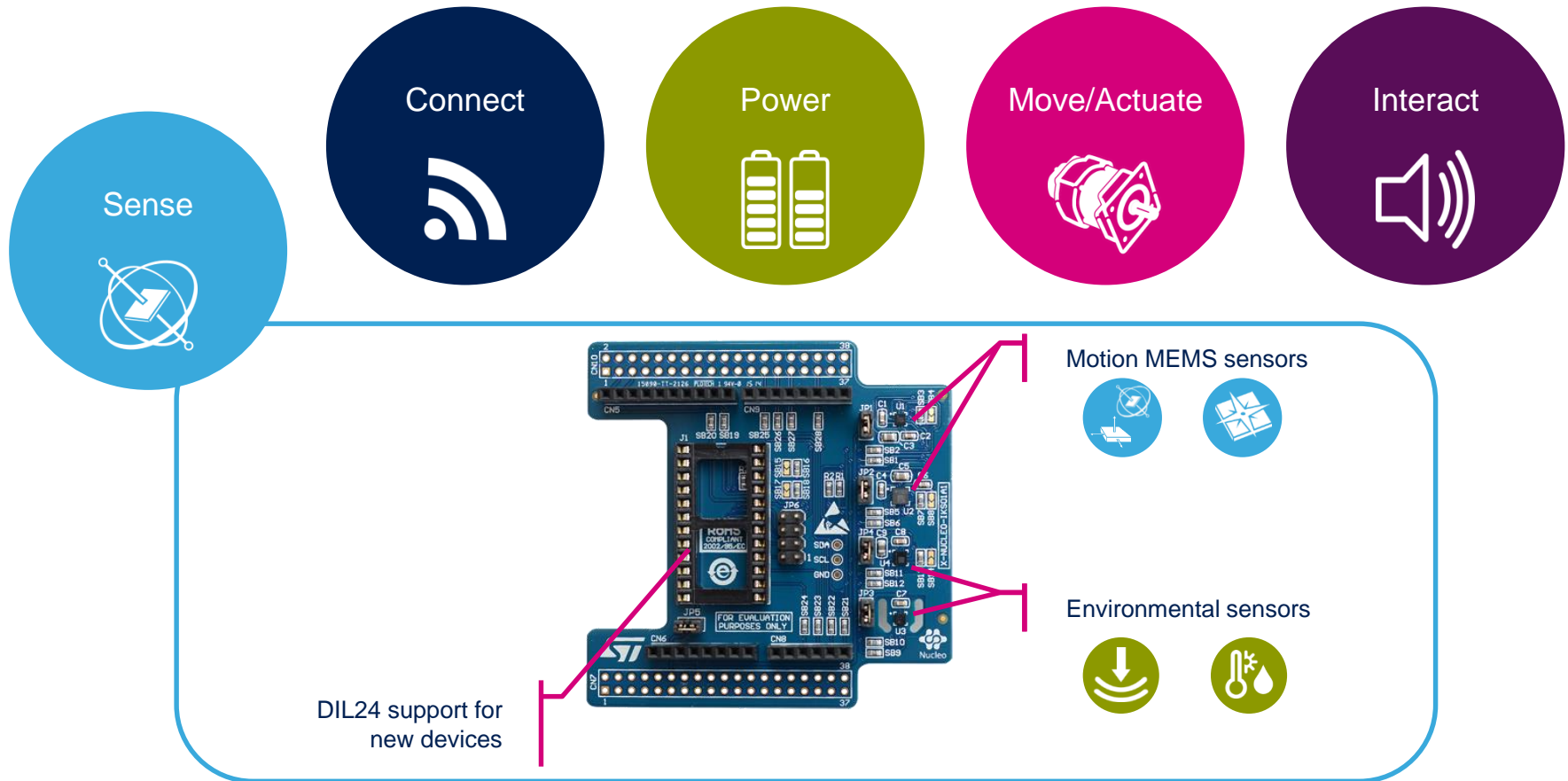
- A comprehensive range of affordable development boards for all the STM32 microcontroller series, with unlimited unified expansion capabilities and integrated debugger/programmer functionality.



STM32 Nucleo Expansion Boards (X-NUCLEO)

20

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



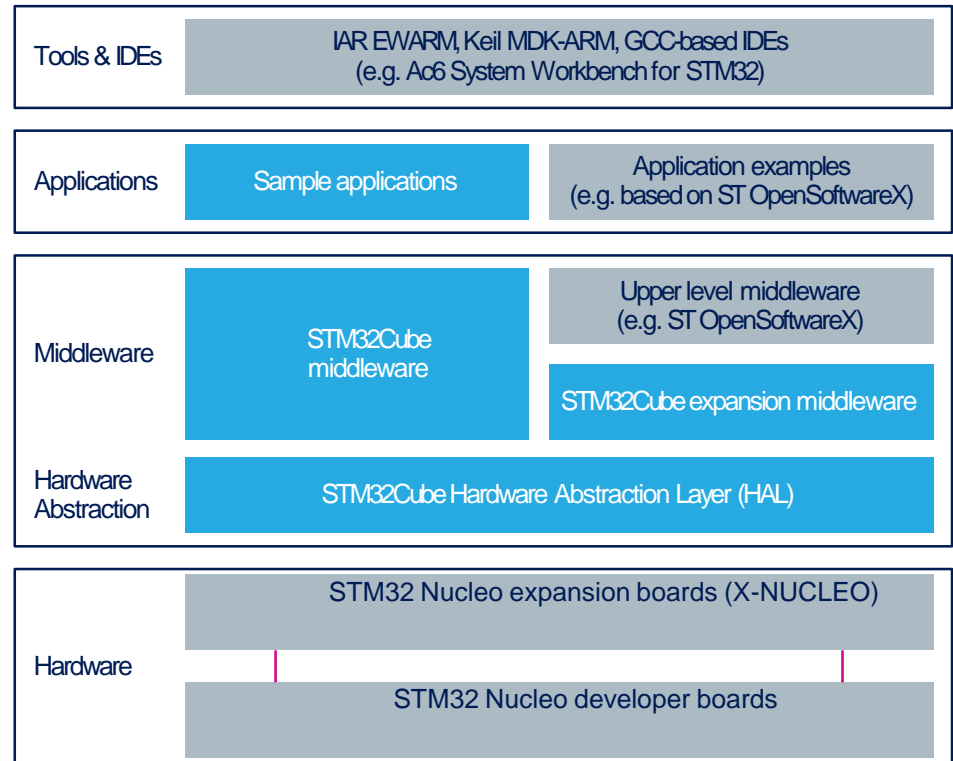
Example of STM32 expansion board (X-NUCLEO-TPS01A1)

STM32 Open Development Environment

Software components

21

- **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.

STM32 Open Development Environment

Building block approach

22

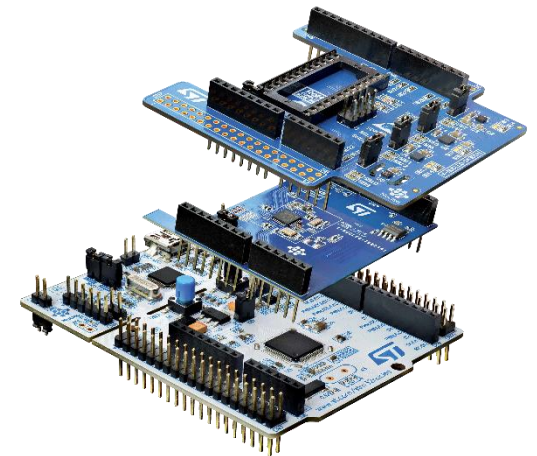
The building blocks

Your need

Our answer



 **STM32 Open Development Environment**



www.st.com/stm32code