Getting started with the STMicroelectronics X-CUBE-SUBG2 software package for STM32CubeMX

Introduction

This document provides the guidelines to configure and use the X-CUBE-SUBG2 software package V4.1.0 for STM32CubeMX (minimum required version V6.2.0). The document contains a description of the provided sample application, a description of the steps required to configure a generic project using the X-NUCLEO-S2868A1 or X-NUCLEO-S2868A2 or X-NUCLEO-S2915A1 expansion board with a Nucleo board, as well as a description of the steps to configure and use the sample application provided in the package.

Information and documentation related to the S2LP components, the X-NUCLEO-S2868A1 or X-NUCLEO-S2868A2 or X-NUCLEO-S2915A1 expansion board and the ST expansion software for S2LP are available on www.st.com.

Contents

List of figures

1 Acronyms and abbreviations

Table 1 List of acronyms

Acronym	Description
GHz	Giga Hertz
P2P	Point-to-Point communication
RF	Radio frequency
HAL	Hardware Abstraction Layer
SPI	Serial peripheral interface
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
USART	Universal Synchronous Asynchronous Receiver Transmitter

2 What is STM32Cube?

STM32Cube™ represents an original initiative by STMicroelectronics to ease developers' life by reducing development effort, time and cost. STM32Cube covers the STM32 portfolio. Version 1.x of STM32Cube includes:

- STM32CubeMX, a graphical software configuration tool that allows the generation of C initialization code using graphical wizards.
- A comprehensive embedded software platform, delivered per series (such as the STM32CubeF4 for STM32F4 series).
    - STM32Cube HAL, an STM32 abstraction layer embedded software, ensuring maximized portability across the STM32 portfolio;
    - a consistent set of middleware components, such as RTOS, USB, TCP/IP, graphics;
    - all embedded software utilities, including a full set of examples.

3 License

The software provided in this package is licensed under Software License Agreement SLA0077.

4 Sample Application Description

In this section, a short overview of the sample applications and examples included in the X-CUBE-SUBG2 pack is provided. For more info kindly refer User Manual of X-CUBE-SUBG2 available on www.st.com

The sample applications/examples:

- are ready-to-use projects that can be generated through the STM32CubeMX for any Nucleo board and using the X-NUCLEO-S2868A1 or X-NUCLEO-S2868A2 or X-NUCLEO-S2915A1 expansion board.
- show the users how to use the APIs to correctly initialize and use the S2-LP

4.1 P2P (Point to Point) Application

This application provides a Point-to-Point communication example which involves sending a buffer from one node to another and acknowledgments using the features in S2LP.

4.2 Contiki-NG based Applications

Contiki-NG *[2]* is a middleware that provides 6LoWPAN communication on top of the S2-LP radio. This involves MAC layers, RPL routing protocols, 6LoWPAN adaptation sub-layer, IPv6 network layer, UDP/TCP and above protocols.

- UDP Client: sends periodic UDP messages to a UDP Server and receives a UDP response (if any).
- UDP Server: receives UDP messages and replies with another UDP message.
- Border Router: connects the wireless 6LoWPAN with the IPv6 protocol stack of a host PC.
- Serial Sniffer: captures RF packets and sends them via serial line to Wireshark application running on a host PC.

5 Installing the X-CUBE-SUBG2 pack in STM32CubeMX

After downloading (from www.st.com), installing and launching the STM32CubeMX (V≥6.2.0), the X-CUBE-SUBG2 pack can be installed in few steps.

1. From the menu, select Help > Manage embedded software packages

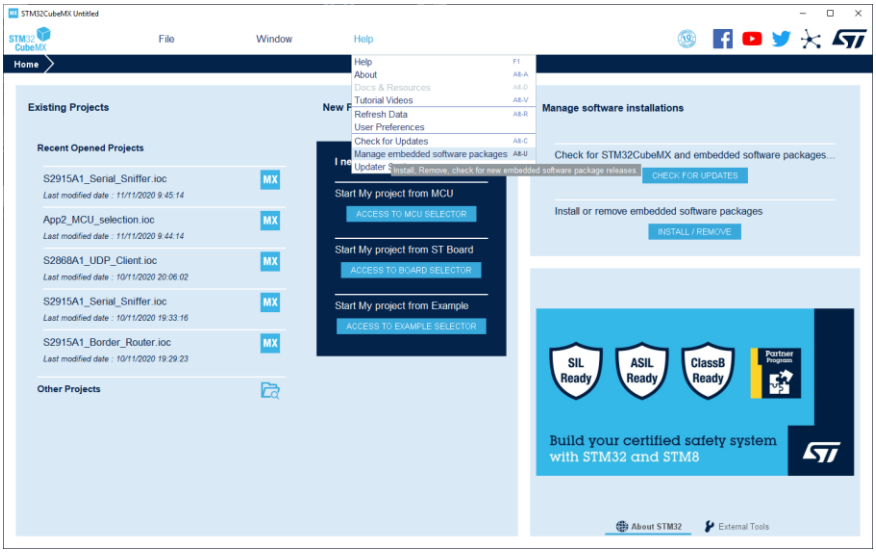


Figure 1 *Managing embedded software packs in STM32CubeMX*

2. From the Embedded Software Packages Manager window, press the ‘Refresh’ button to get an updated list of the add-on packs. Go to the ‘STMicroelectronics’ tab to find the X-CUBE-SUBG2 pack.

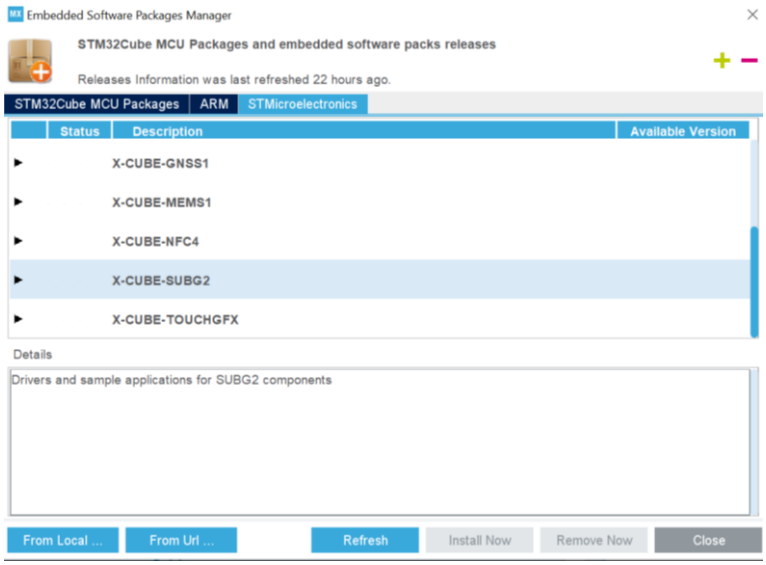


Figure 2 *Installing the X-CUBE-SUBG2 pack in STM32CubeMX*

3. Select it checking the corresponding box and install it pressing the ‘Install Now’ button. Once the installation is completed, the corresponding box will become green, the ‘Close’ button can be pressed and the configuration of a new project can start.

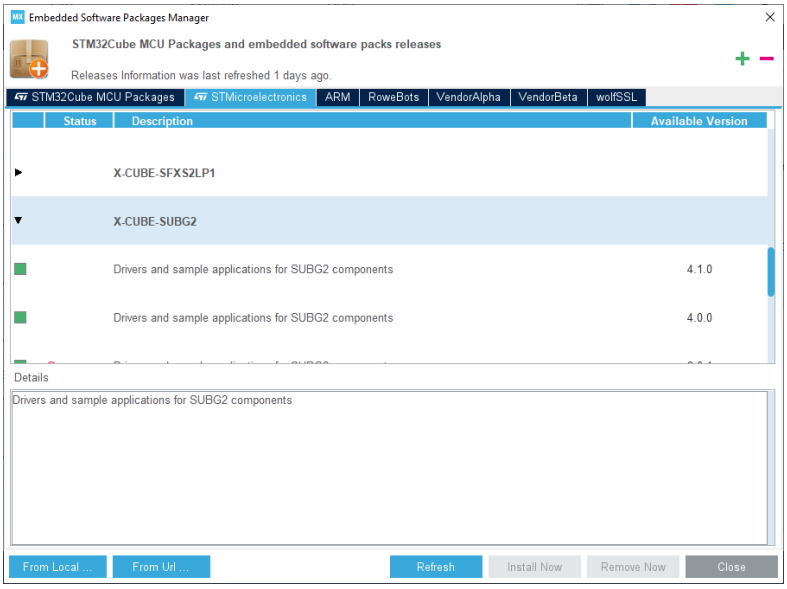


Figure 3 *The X-CUBE-SUBG2 pack in STM32CubeMX*

6 Starting a new project

After launching the STM32CubeMX, you can choose if starting a New Project from the MCU Selector or from the Board Selector.

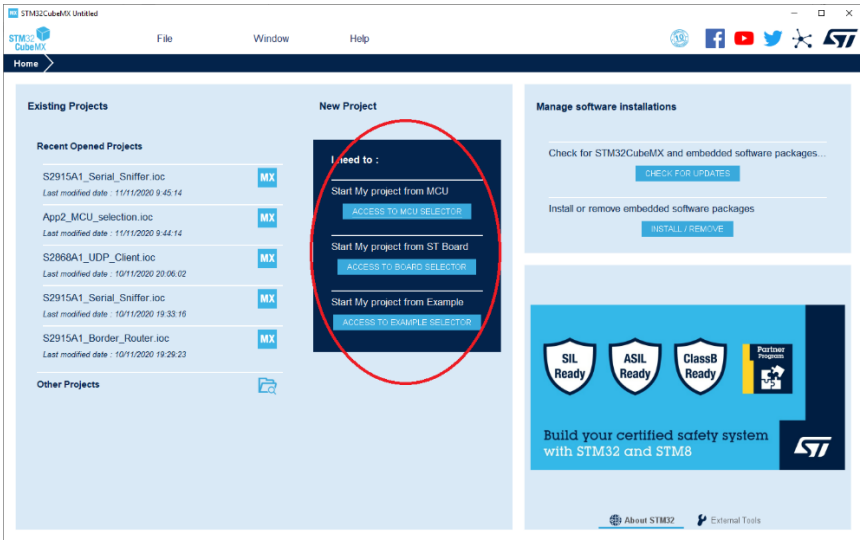


Figure 4 *STM32CubeMX main page*

The **MCU/Board selector** window will pop up. From this window, the STM32 MCU or platform can be selected.

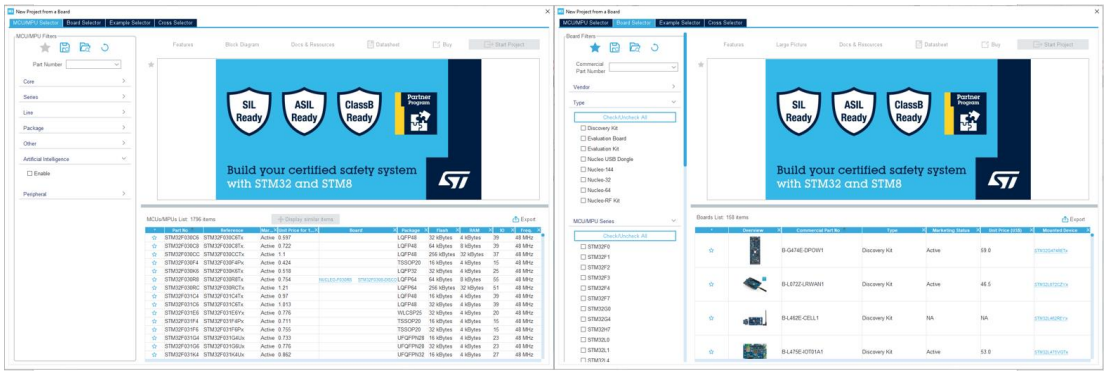


***Figure 5** STM32CubeMX **MCU/Board Selector** windows*

After selecting the MCU or the Board, the selected STM32 pinout will appear. From this window the user can set up the project, by adding one or more Additional Software and peripherals and configuring the clock.

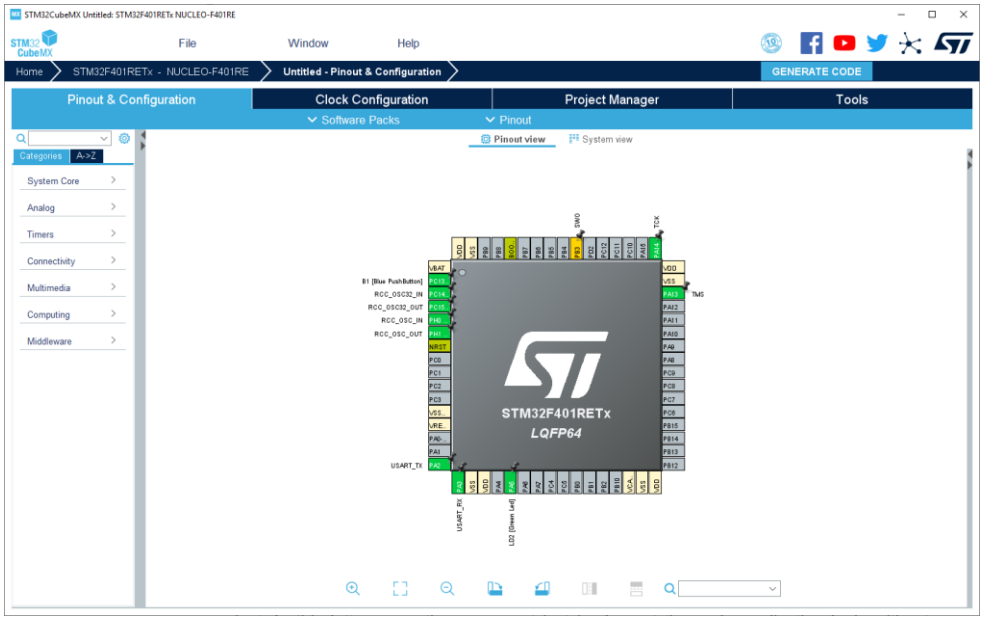


***Figure 6** STM32CubeMX **Pinout & Configuration** window*

To add the X-CUBE-SUBG2 additional software to the project, the “Software Packs” and then “Select Components” button must be clicked.
From the Software Pack Component Selector window, the user can either choose to generate, for the selected MCU/Board, one of the enclosed sample applications or a new project. In this latter case, the user must just implement the main application logic without bothering with the pinout and peripherals configuration code that will be automatically generated by STM32CubeMX.

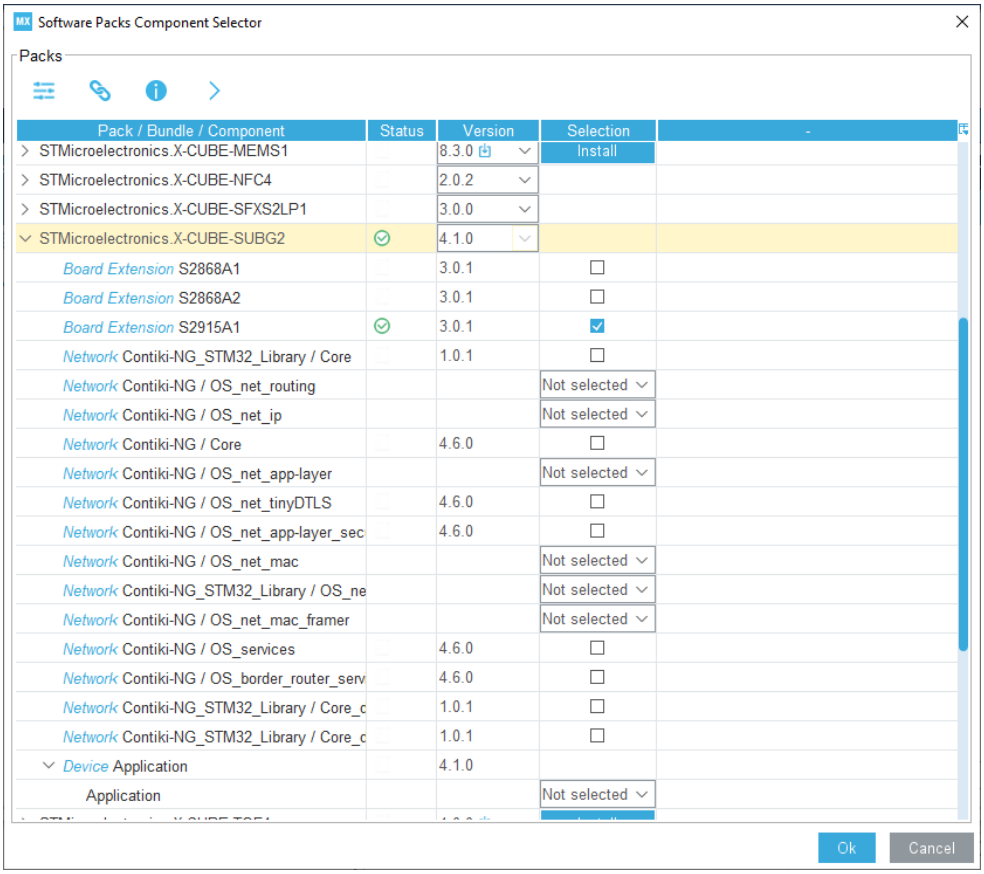


***Figure 7** STM32CubeMX **Software Pack Component Selector** window*

7 STM32 Configuration Steps

The X-NUCLEO-S2868A1 or X-NUCLEO-S2868A2 or X-NUCLEO-S2915A1 interfaces with the STM32 microcontroller via the SPI bus.

Hence, assuming a user wants to interface the ST X-NUCLEO-S2868A1 or X-NUCLEO-S2868A2 or X-NUCLEO-S2915A1 expansion board with a STM32 Nucleo 64 pins board (e.g. a Nucleo-F401RETx) no particular hardware modification must be done.

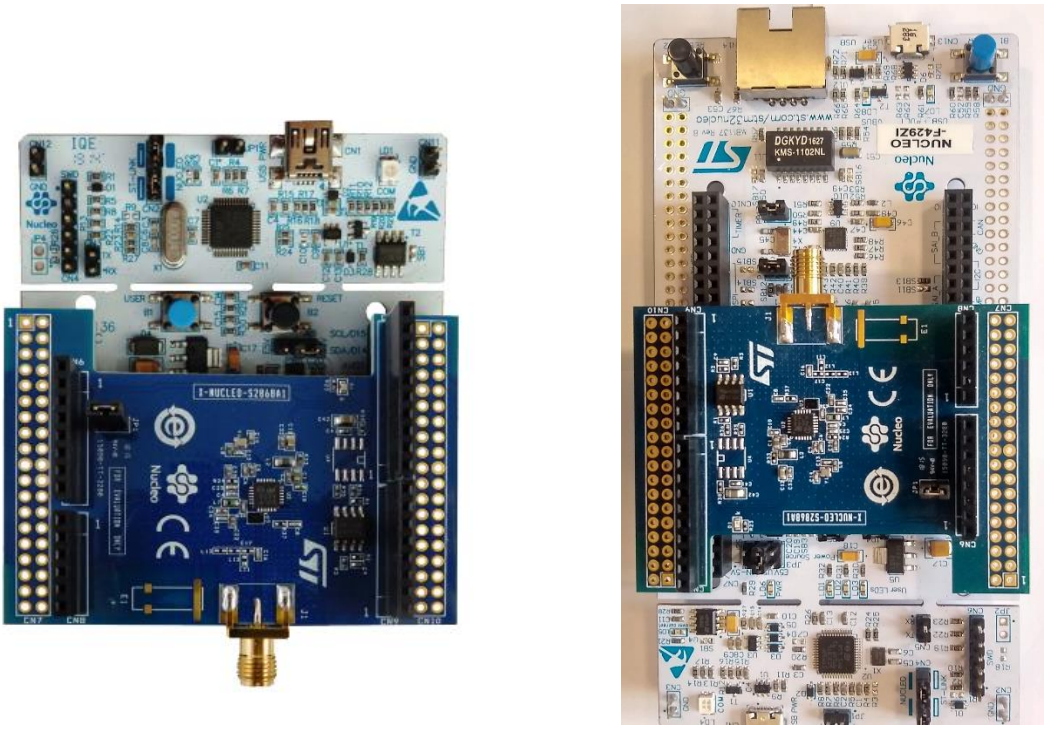


Figure 8 *STM32 Nucleo 64 & Nucleo 144 with X-NUCLEO-S2868A1*

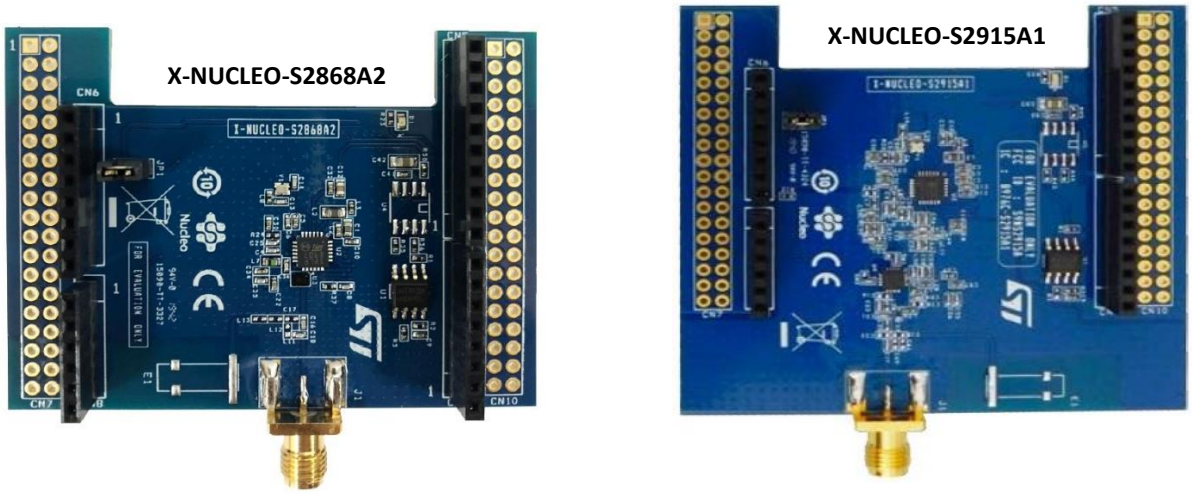


Figure 9 *X-NUCLEO-S2868A1 & X-NUCLEO-S2915A1*

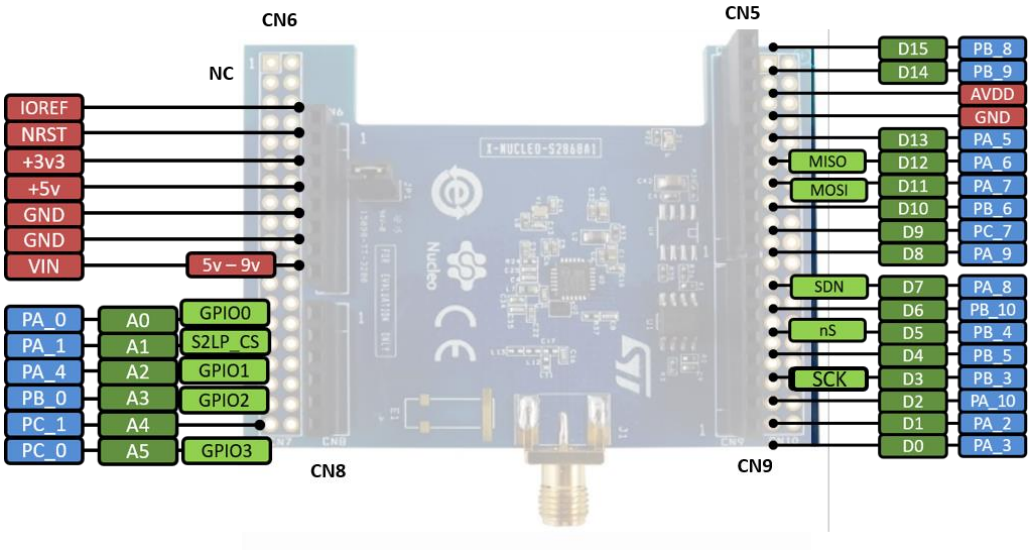


***Figure 10** X-NUCLEO-S2868A1 pinout*

Note that the X-NUCLEO-S2868A1 and X-NUCLEO-S2868A2 have the same pinout. However, the pinout of the X-NUCLEO-S2915A1 are different and depicted in Figure 11.

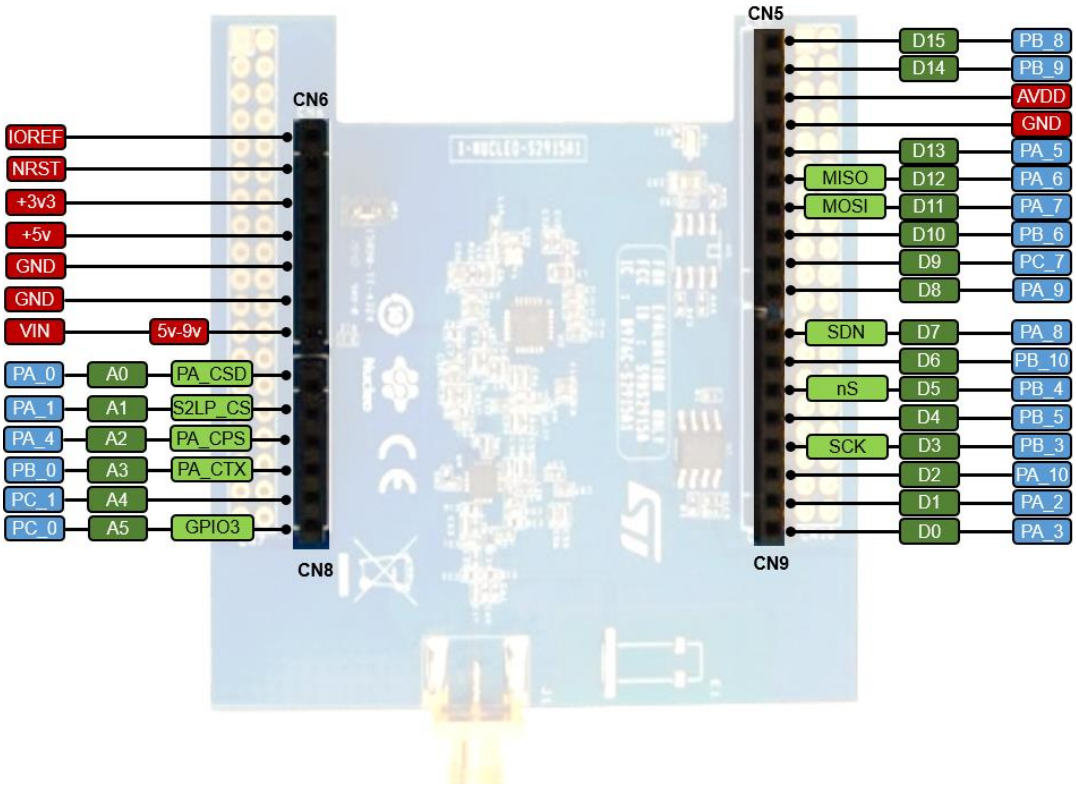


***Figure 11** X-NUCLEO-S2915A1 pinout*

7.1 P2P Application based on X-NUCLEO-S2868A1 or X-NUCLEO-S2868A2 or X-NUCLEO-S2915A1

This section outlines how to configure STM32CubeMX with X-NUCLEO-S2868A1 or X-NUCLEO-S2868A2 or X-NUCLEO-S2915A1 when the use of the P2P sample example is required. With such setup, only driver layers will be configured.

To add the X-CUBE-SUBG2 additional software to the project, the “Software Packs” and then “Select Components” button must be clicked. From the “Software Pack Component Selector” window, the user has to select the example from the “Device” class and “Board Extension” class as shown in the figure below.

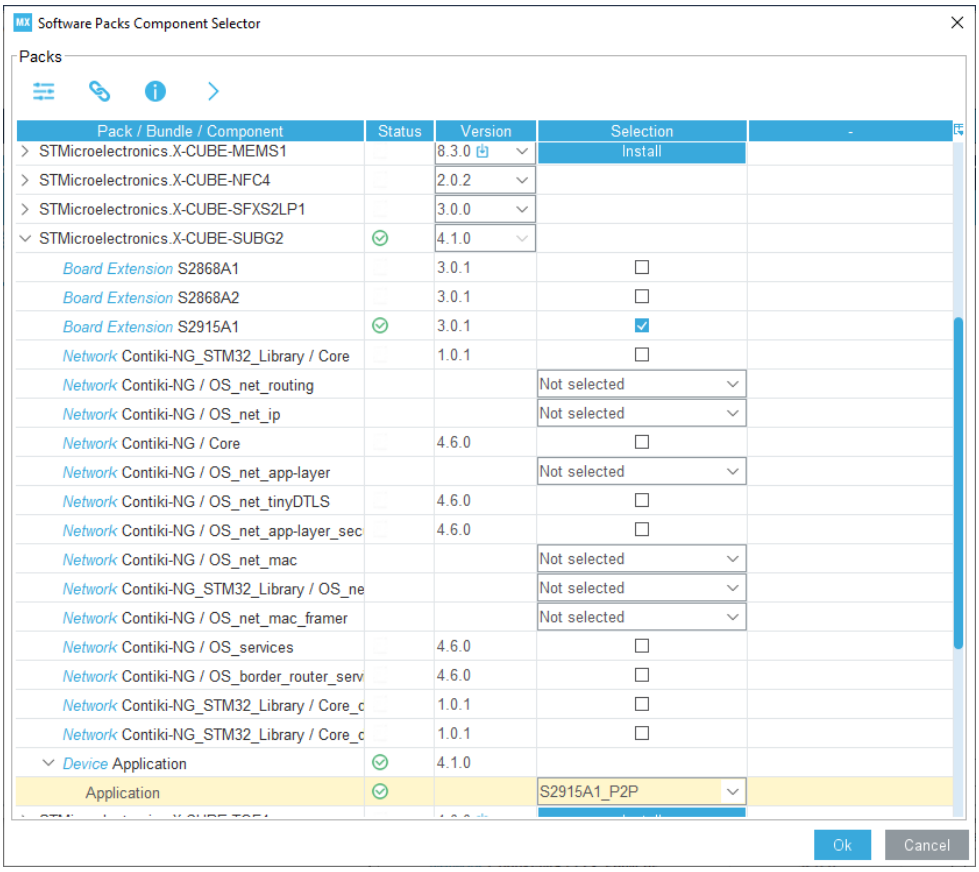


Figure 12 *STM32CubeMX Software Pack Component Selector window example for S2915A1*

From the **Pinout & Configuration** tab, set the SPI according to Table 2.

Group	Nucleo boards	SPI_SCK	SPI_MISO	SPI_MOSI	SPI_Instance
BG1	F030R8, F070RB, F072RB, F091RC, F334R8, F303RE, F401RE, F411RE, F446RE, F410RB, L010RB, L053R8, L073RZ, L152RE, L452RE, L476RG, G070RB, G071RB, G431RB, G474RE	“PB3”	“PA6”	“PA7”	SPI1
BG2	F103RB	“PA5”	“PA6”	“PA7”	SPI1
BG3	F302R8 L412RB-P, L452RE-P, L433RC-P	“PB13”	“PB14”	“PB15”	SPI2

BG4	F429ZI, F446ZE, F207ZG, F746ZG, F767ZI, H743ZI, F722ZE, F303ZE, F412ZG, F413ZH, L552ZE-Q, WB-55	“PA5”	“PA6”	“PA7”	SPI1
BG5	L496ZG-P, L496ZG, L4R5ZI, L4R5ZI-P	“PE13”	“PA6”	“PA7”	SPI1
BG6	H753ZI, H743ZI2 H745ZI-Q, H755ZI-Q	“PA5”	“PA6”	“PB5”	SPI1

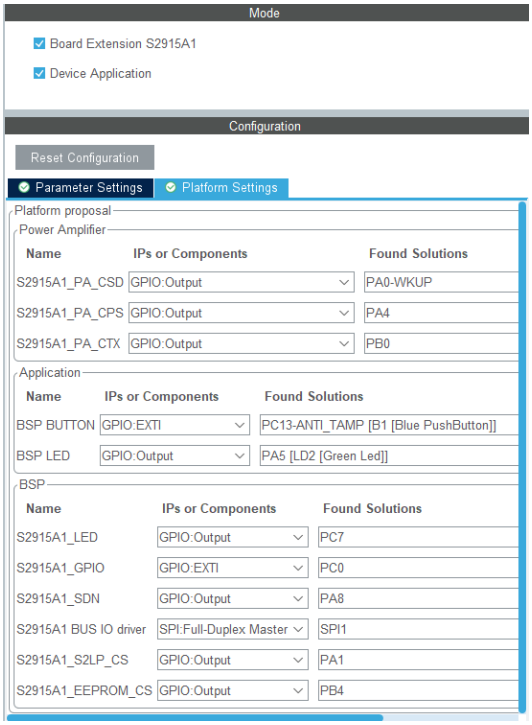
Table 2 *SPI Configurations for different Nucleo Boards*

For example, using the Nucleo F401RE,

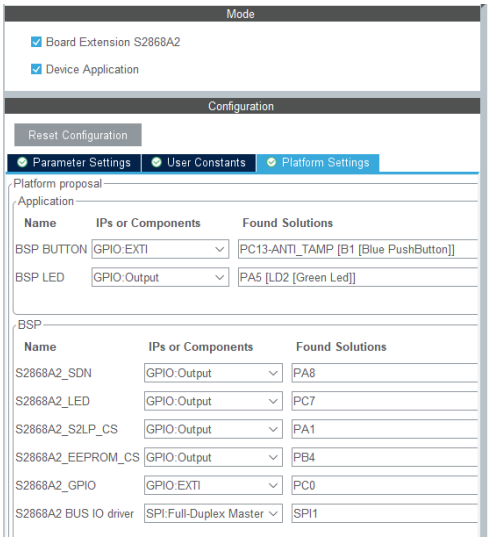
- from the **Pinout** scheme, click on PA6 and set it as SPI1_MISO;
- from the **Pinout** scheme, click on PA7 and set it as SPI1_MOSI;
- from the **Pinout** scheme, click on PB3 and set it as SPI1_SCK;
- enable the SPI1 as SPI from the “Connectivity” category;

While using X-NUCLEO-S2868A1 or X-NUCLEO-S2915A1 or X-NUCLEO-S2868A2 with boards under **“BG2”**, **“BG3”**, **“BG4”** and **“BG6”** groups presented in Table 2 (i.e., F103RB, F302R8, L412RB-P, L452RE-P, L433RC-P, F429ZI, F446ZE, F207ZG, F746ZG, F767ZI, H743ZI, F722ZE, F303ZE, F412ZG, F413ZH, L552ZE-Q, WB-55, H753ZI, H743ZI2, H745ZI-Q and H755ZI-Q), the SPI clock (i.e., SPI_SCK) is available on pin D13 instead of D3. In these cases, the following modification are required to correctly set the SPI clock.

- Remove R11 and Connect R6.



(a)



(b)

Figure 13 *STM32CubeMX Additional Software settings for P2P application: Mode and Configuration view a) for X-NUCLEO-S2915A1 b) for X-NUCLEO-S2868A2 (or X-NUCLEO-S2868A1)*

As shown from Figure 12, the power amplifier configurations are only needed for the X-NUCLEO-S2915A1.
From the **Software Packs** category, press the 'Stmicroelectronics.X-CUBE-SUBG2.4.1.0' item, enable the "Board Extension" checkbox from the "Mode" view and set the following Platform Settings from the "Configuration" view (take into account that according the example chosen some settings can appear or not) based on Table 3.

Table 3 *Platform Settings configurations*

Name			BSP_API	Supported Ips	Nucleo 64	Nucleo 144
S2868A1	**S2868A2**	**S2915A1**				
S2868A1 BUS IO driver	S2868A2 BUS IO driver	S2915A1 BUS IO driver	BSP_BUS_DRIVER	SPI: SPI	SPI1(2)	SPI1
S2868A1_GPIO3	S2868A2_GPIO3	S2915A1_GPIO3	HAL_EXTI_DRIVER	GPIO:EXTI	A5	
S2868A1_S2LP_CS	S2868A2_S2LP_CS	S2915A1_S2LP_CS		GPIO:Output	A1	
S2868A1_SDN	S2868A2_SDN	S2915A1_SDN		GPIO:Output	D7	
BSP BUTTON	BSP BUTTON	BSP BUTTON	BSP_COMMON_DRIVER	GPIO: EXTI	PC13	
BSP LED	BSP LED	BSP LED		GPIO:Output	PA5(PB13)	PB7
S2868A1_EEPROM_CS	S2868A2_EEPROM_CS	S2915A1_EEPROM_CS	BSP_COMMON_DRIVER	GPIO:Output	D5	
S2868A1_LED	S2868A2_LED	S2915A1_LED		GPIO:Output	D9	
		S2915A1_PA_CSD		GPIO:Output	A0	
		S2915A1_PA_CPS		GPIO:Output	A2	
		S2915A1_PA_CTX		GPIO:Output	A3	

Table 4 presents the "BSP LED" pin names for different Nucleo Boards.

Table 4 *BSP LED pin names for different Nucleo Boards*

Nucleo Board	BSP LED
F030R8, F070RB, F072RB, F091RC, F103RB, F303RE, F334R8, F401RE, F410RB, F411RE, F446RE, L010RB, L053R8, L073RZ, L152RE, L452RE, L476RG	“PA5”
F302R8, L452RE-P, L433RC-P, L412RB-P	“PB13”
WB-55	“PB0”
Nucleo-144 boards	“PB7”

For the P-NUCLEO-WB55 board, the “BSP BUTTON” should be set to “PC4” and not to “PC13”.
The A0, A1, A2, A3, A5, D9, D7 and D5 pins depend on the Nucleo Board and their appropriate names are recapitulated in Table 5.

Table 5 *A0, A1, A2, A3, A5, D9, D7 and D5 pin names for different Nucleo Boards*

<table>
<tr><th>Boards</th><th>A0</th><th>A1</th><th>A2</th><th>A3</th><th>A5</th><th>D9</th><th>D7</th><th>D5</th></tr>
<tr><td>1) F030R8
2) F070RB
3) F072RB
4) F091RC
5) F303RE
6) F334R8</td><td>PA0</td><td rowspan="5">PA1</td><td rowspan="5">PA4</td><td rowspan="5">PB0</td><td rowspan="5">PC0</td><td rowspan="5">PC7</td><td rowspan="5">PA8</td><td rowspan="4">PB4</td></tr>
<tr><td>7) F401RE
8) F410RB
9) F411RE
10) F446RE</td><td>PA0-WKUP</td></tr>
<tr><td>11) L053R8
12) L073RZ</td><td>PA0</td></tr>
<tr><td>13) L152RE</td><td>PA0-WKUP1</td></tr>
<tr><td>14) L452RE
15) L476RG</td><td>PA0</td><td>PB4
(NJTRST)</td></tr>
<tr><td>16) F302R8
17) L010RB</td><td>PA0</td><td>PA1</td><td>PA4</td><td>PB0</td><td>PC0</td><td>PC7</td><td>PA8</td><td>PB4</td></tr>
<tr><td>18) G070RB
19) G071RB</td><td rowspan="2">PA0</td><td rowspan="2">PA1</td><td rowspan="2">PA4</td><td rowspan="2">PB1</td><td rowspan="2">PB12</td><td rowspan="2">PC7</td><td rowspan="2">PA8</td><td rowspan="2">PB4</td></tr>
<tr><td>20) G431RB
21) G474RE</td></tr>
<tr><td>22) L412RB-P
23) L452RE-P</td><td rowspan="2">PA0</td><td rowspan="2">PA1</td><td rowspan="2">PC3</td><td rowspan="2">PC2</td><td rowspan="2">PC0</td><td rowspan="2">PA8</td><td rowspan="2">PC7</td><td rowspan="2">PA15
(JTDI)</td></tr>
<tr><td>24) L433RC-P</td></tr>
<tr><td>25) F103RB</td><td>PA0-WKUP</td><td>PA1</td><td>PA4</td><td>PB0</td><td>PC0</td><td>PC7</td><td>PA8</td><td>PB4</td></tr>
</table>

26) F207ZG **27) F429ZI** **28) F446ZE** **29) F722ZE** **30) F746ZG** **31) F767ZI**	PA3	PC0	PC3	PF3	PF10	PD15	PF13	PE11
32) H743ZI			PC3_C					
33) L496ZG **34) L496ZG-P** **35) L4R5ZI** **36) L4R5ZI-P** **37) F412ZG** **38) F413ZH**	PA3	PC0	PC3	PC1	PC5	PD15	PF13	PE11
39) F303ZE	PA3	PC0	PC3	PD11	PD13	PD15	PF13	PE11
40) H753ZI	PA3	PC0	PC3_C	PB1	PF10	PD15	PG12	PE11
41) L552ZE-Q	PA3	PA2	PC3	PB0	PC0	PD15	PF13	PE11
42) WB55	PC0	PC1	PA1	PA0	PC2	PA9	PC13	PA15

From the **Pinout** & **Configuration** tab, click on “System Core” category and then on NVIC item to enable the EXTI line interrupts.

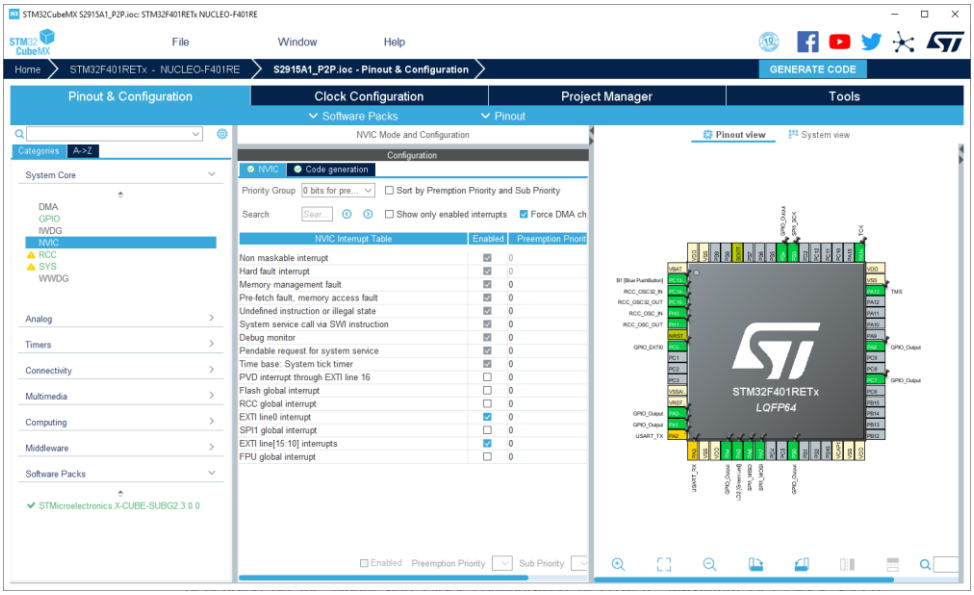


Figure 14 *STM32CubeMX NVIC Configuration*

From the **Pinout** & **Configuration** tab, click on “Connectivity” category and then select SPI instance and set the SPI Prescaler (for Baud Rate), SPI instance and prescaler will vary depending on the pinout and Clock configuration of STM32. Maximum SPI Clock speed supported by S2LP is 10 MHz. Hence, the prescaler (for Baud rate) should be such that Clock Speed on STM32 NUCLEO is less than 10Mhz. In Figure 13, NUCLEO-F401RE is taken into account, since HCLK is 84 MHZ and so the prescaler set for SPI is 16.

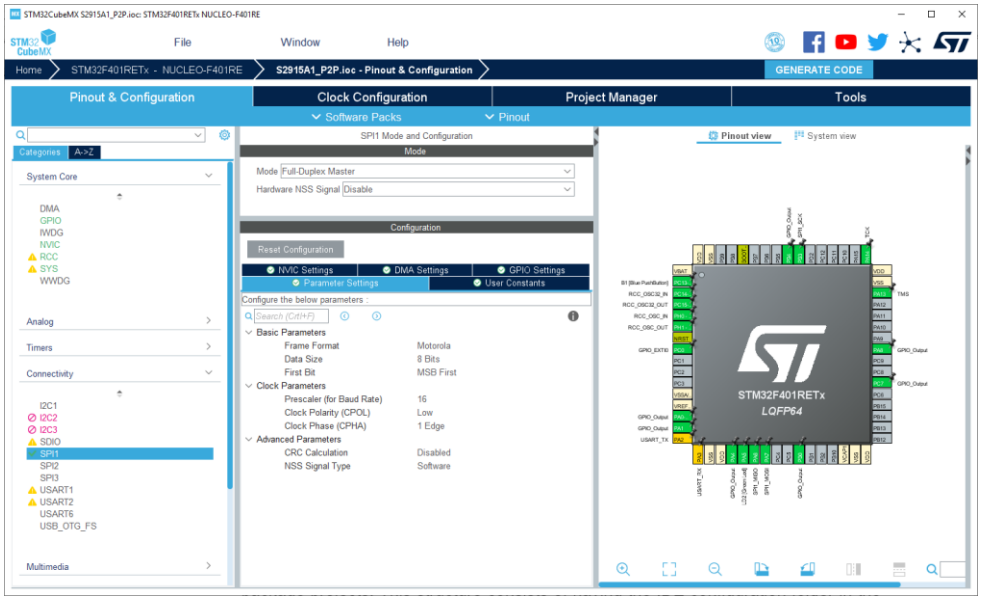


Figure 15 *STM32CubeMX SPI Configuration*

Once all the above described steps have been performed, the source code of the project using the **STMicroelectronics.X-CUBE-SUBG2** software can be generated clicking the “GENERATE CODE” button.

7.2 Contiki-NG applications based on X-NUCLEO-S2868A1 or X-NUCLEO-S2868A2 or X-NUCLEO-S2915A1

All the configurations steps described in the previous section apply as the base also for the Contiki-NG based application, for what concerns the Power Amplifier and the BSP.
In this case, more components can be selected in order to have the applications working.
For any Contiki-NG based application to work, three components **must** always be selected:

- Contiki-NG_STM32_Library / Core
- Contiki-NG / Core
- Contiki-NG / OS_services

There are other components that are optional or can be configured. We can highlight a common suggested configuration for UDP Client and UDP Server. In **Figure 16** this configuration is reported:

- Contiki-NG / OS_net_routing configured as **rpl_lite**
- Contiki-NG / OS_net_ip configured as **Ipv6**
- Contiki-NG / OS_net_mac configured as **csma**
- Contiki-NG / OS_net_mac_framer configured as **framer_802_15_4**

This is the common configuration meaning that Ipv6 protocol is enabled, “lite” version of the RPL routing protocol is used, Contiki-NG CSMA protocol is chosen for mac layer and standard 802.15.4 framer layer is selected.
This configuration can be used also to build other applications based on Contiki-NG.
It is important to note that the configuration of these 4 components must match in all the nodes to make them able to communicate.
The user can optionally choose to change the RPL routing protocol version by using:

- Contiki-NG / OS_net_routing configured as **rpl_classic**

but the same choice must be done for all the nodes to communicate.

The configuration of the following components (that enable usage of LED and Button):

- Contiki-NG_STM32_Library / Core_dev_leds

- Contiki-NG_STM32_Library / Core_dev_buttons

is optional for UDP applications.

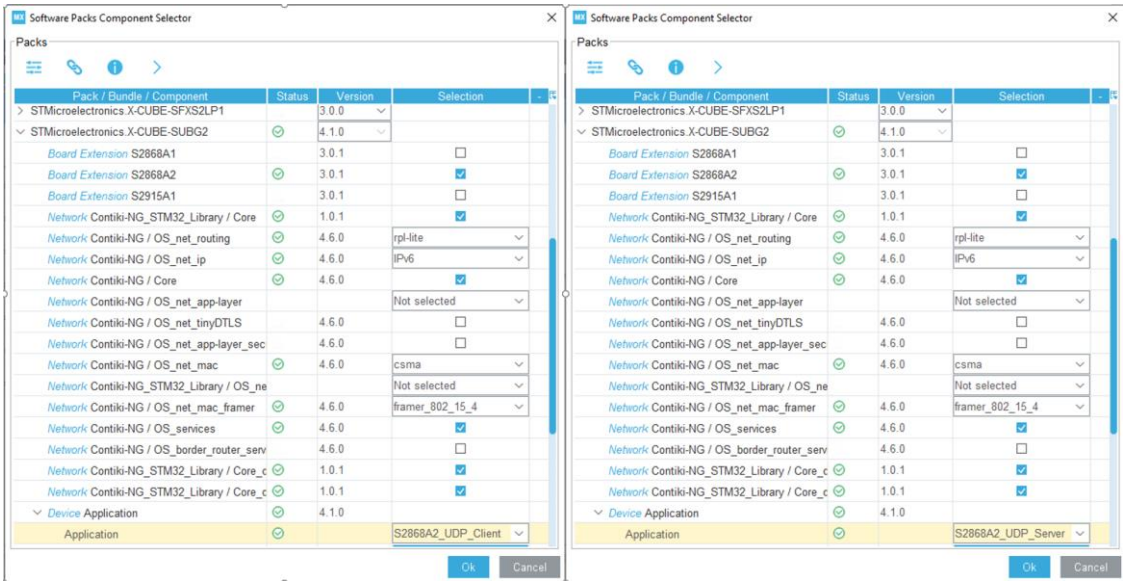


Figure 16 *Contiki-NG configuration for UDP applications*

From the **Software Packs** category, press the 'Stmicroelectronics.X-CUBE-SUBG2.4.1.0' item, enable the "Board Extension", "Contiki-NG" related and "Application" checkboxes from the "Mode" view. For the BSP parameters to be set Platform Settings from the "Configuration" view, you can check previous section (related to P2P Example).
In the Contiki-NG case, User Button and LED (if enabled) can be found in the Contiki-NG section.

As shown in **Figure 17** the two main differences are the TIMER and the USART, that must be selected for Contiki-NG based applications.

From the **Pinout** & **Configuration** tab, click on "Timers" category and then select TIM instance and enable the Internal Clock checkbox. No other configuration is needed since the actual initialization will be done from the application (taking into account the RTIMER_ARCH_SECOND parameter to compute the prescaler: default value should be left). **Figure 18** shows the TIM final configuration.

From the **Pinout** & **Configuration** tab, click on "Connectivity" category and then select USART instance and enable it by choosing *Asynchronous* mode. Baud Rate=115200 bps, Word Length=8 bits, Parity=None and Stop Bits=1 parameters must be chosen. It is important to go in "System Core" category and then on NVIC, "Code Generation" tab and ensure no code generation is selected for the chosen USART, since the application will take care of it.
Figure 19 shows the USART final configuration (left) and the NVIC settings (right).

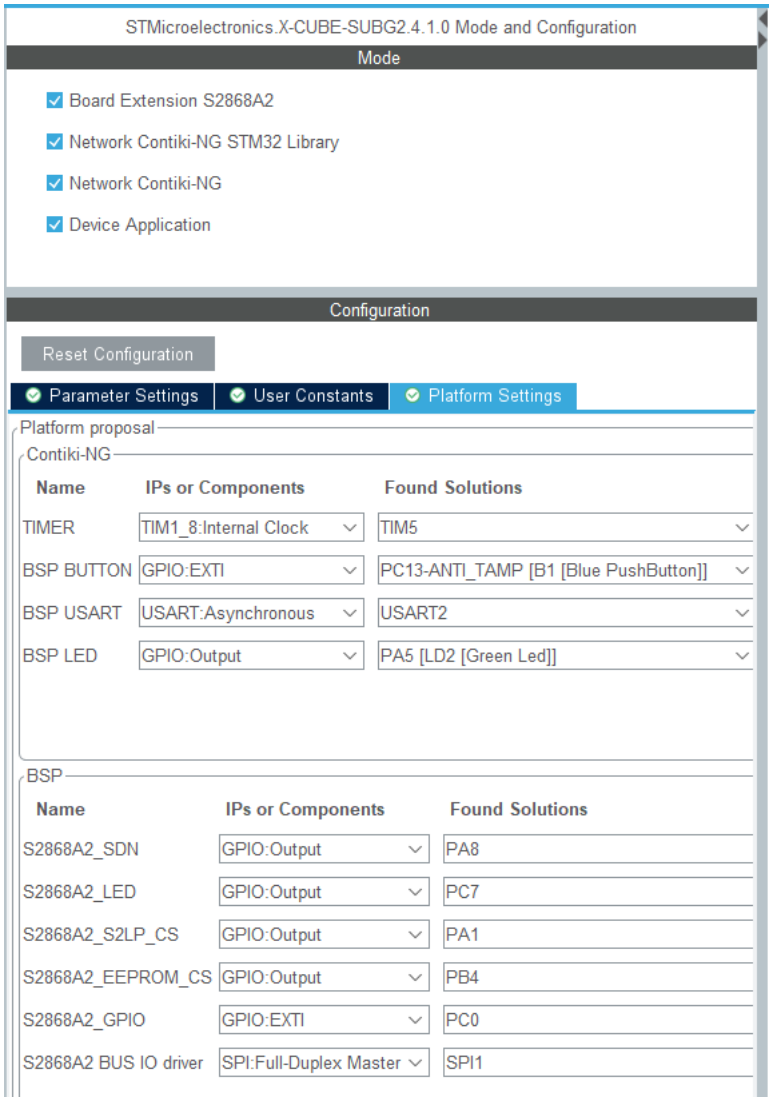


Figure 17 *STM32CubeMX Additional Software settings for Contiki-NG application: Mode and Configuration view*

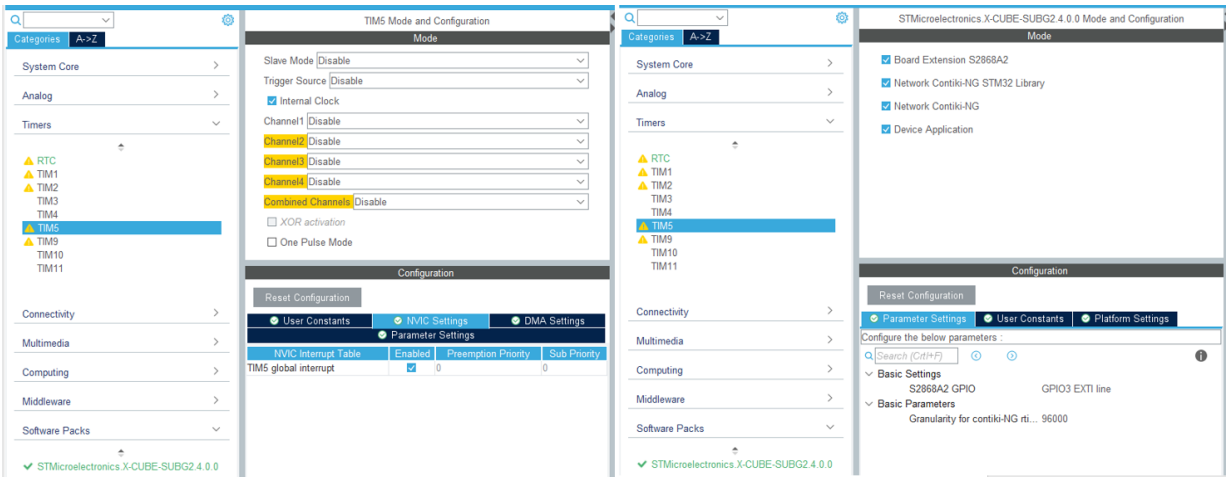


Figure 18 *STM32CubeMX TIM Configuration (left) and RTIMER parameter (right)*

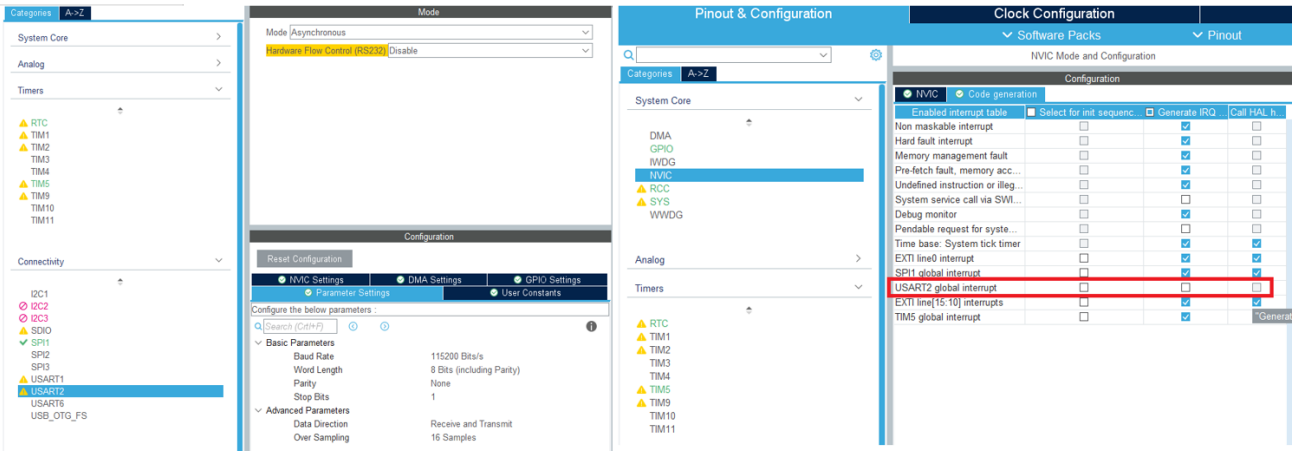


Figure 19 *STM32CubeMX USART Configuration*

For the Border Router application the same base configuration of the UDP applications is used, adding the following component:

- Contiki-NG / OS_border_router_services

For this application the Button component:

- Contiki-NG_STM32_Library / Core_dev_buttons

is also required.
The Border Router configuration is shown in **Figure 20**.

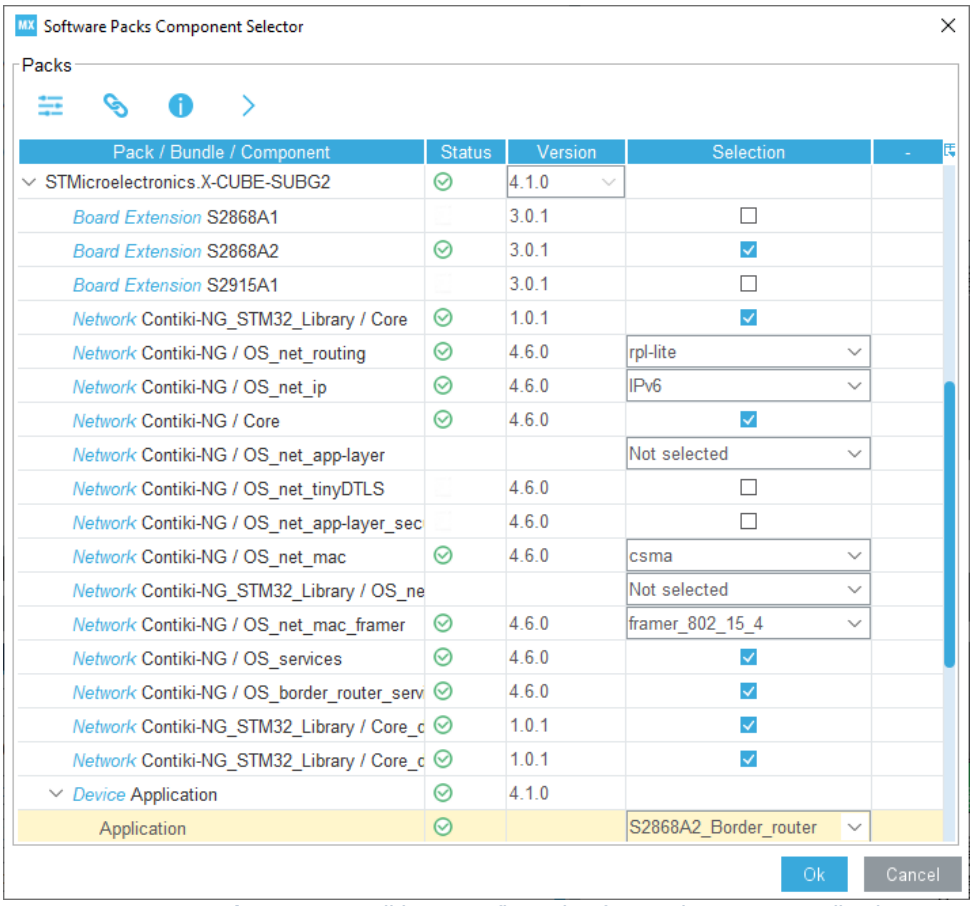


Figure 20 *Contiki-NG configuration for Border Router application*

The Serial Sniffer application differs in term of basic configuration since there is no need of a

full protocol stack to process packets that just need to be captured and sent to Wireshark application.
So, the configuration for the Serial Sniffer application must be as follows (and as summarized in **Figure 21**):

- Contiki-NG / OS_net_routing configured as **nullrouting**
- Contiki-NG / OS_net_ip configured as **nullner**
- Contiki-NG / OS_net_mac **Not selected**
- Contiki-NG_STM32_Library / OS_net_mac configured as **custom**
- Contiki-NG / OS_net_mac_framer configured as **nullframer**

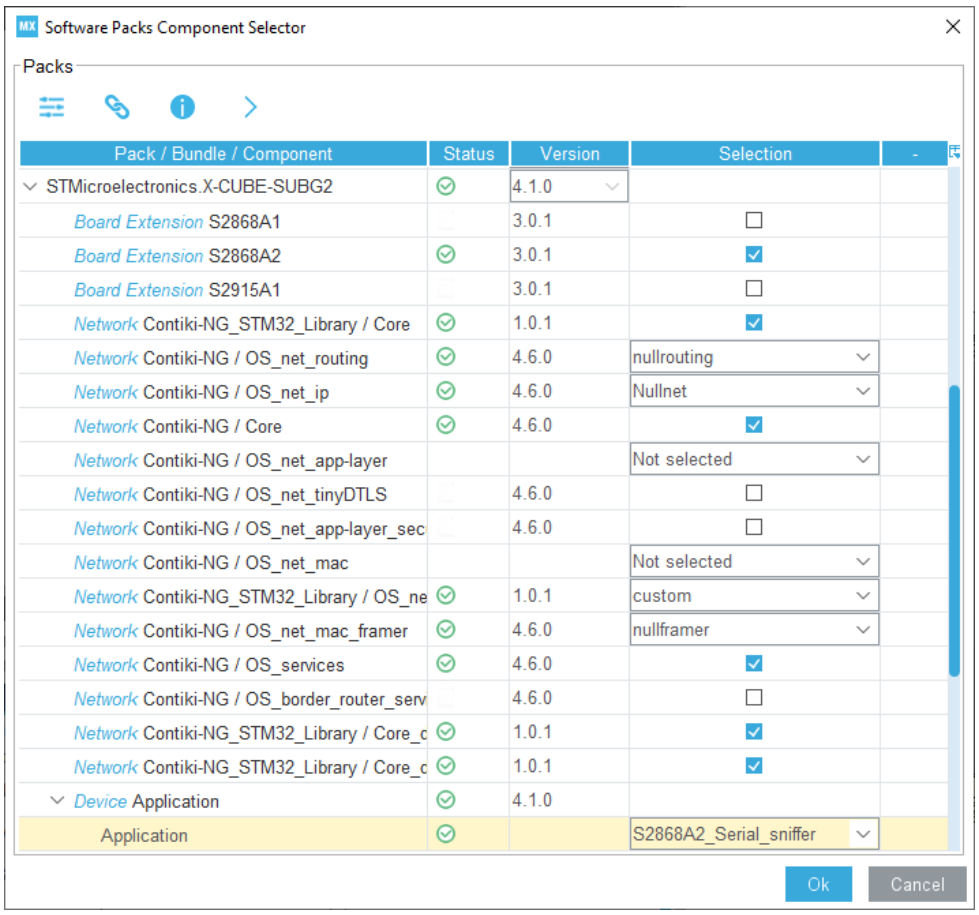


***Figure 21** Contiki-NG configuration for Serial Sniffer application*

The Contiki-NG_STM32_Library / OS_net_mac component implements an almost empty MAC layer that simply forward the captured packets.
For this application the Button component:

- Contiki-NG_STM32_Library / Core_dev_buttons

is also required.

To experiment with different settings, not allowed in the provided applications, the user can choose not to select any application, like shown in **Figure 22**. It is important to notify that these configuration options are not fully tested so it might be difficult to get a fully working scenario.

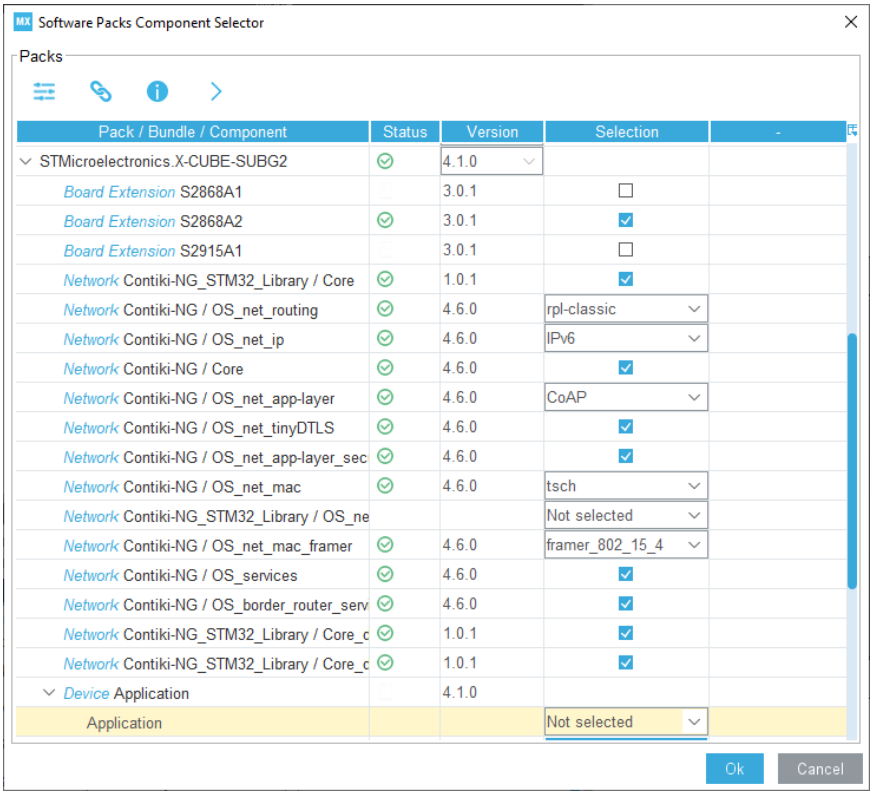


***Figure 22** Contiki-NG configuration not linked to any application*

For all the Contiki-NG based applications the source code can be generated using the "GENERATE CODE" button like in the P2P Example case.

8 Generated Folders Structure

When generating a project, two models of folders structure can be adopted when using a high level firmware component (i.e. a middleware in the STM32Cube MCU package):

- **Basic Structure**: the basic structure is often used with HAL examples and single package projects. This structure consists of having the IDE configuration folder in the same level as the sources (organized in *Inc* and *Src* subfolders).
- **Advanced Structure:** the advanced structure provides a more efficient and organized folders model that allows ease middleware applications integration when several packages are used.
  In the Advanced mode *Src* and *Inc* are generated under folder *Core*.

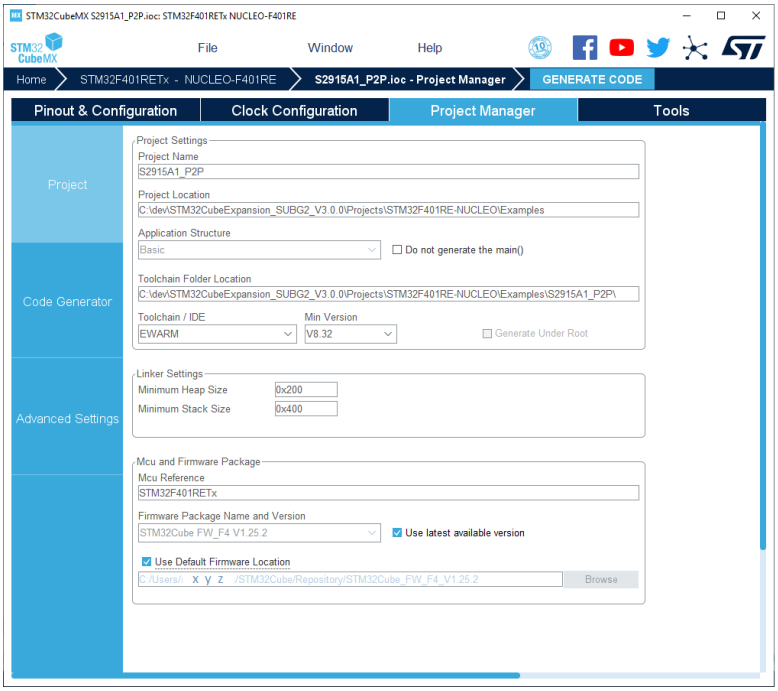


***Figure 23** STM32CubeMX **Application Structure** Configuration*

If using STM32CubeIDE please deselect the (by-default enabled) “Generate Under Root” option, like reported in **Figure 24**.

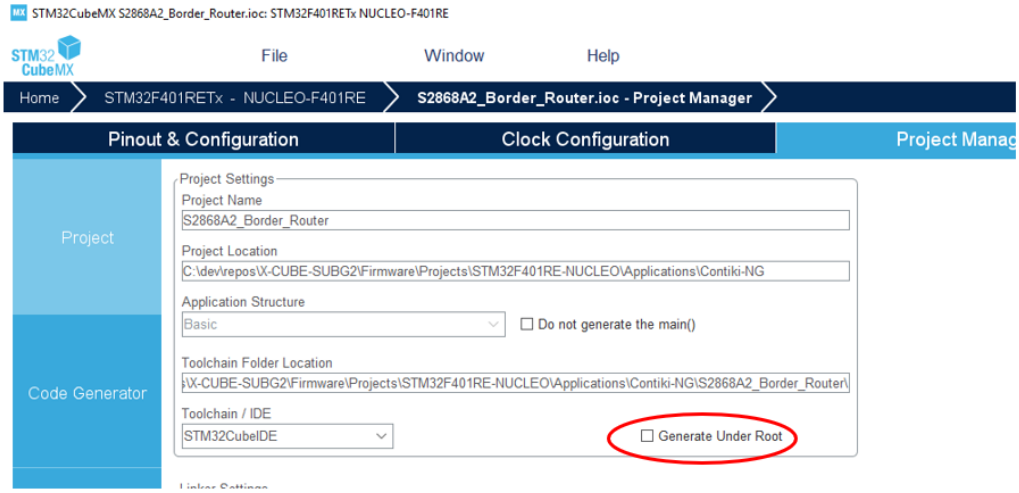


***Figure 24** Generate Under Root option to be disabled*

9 Known Limitations and workarounds

- The Contiki-NG based projects for STM32CubeIDE must use “Standard C” library instead of the default “Reduced C”: Properties, C/C++ Build, Settings, MCU Settings, Runtime library => Standard C, like shown in next figure.

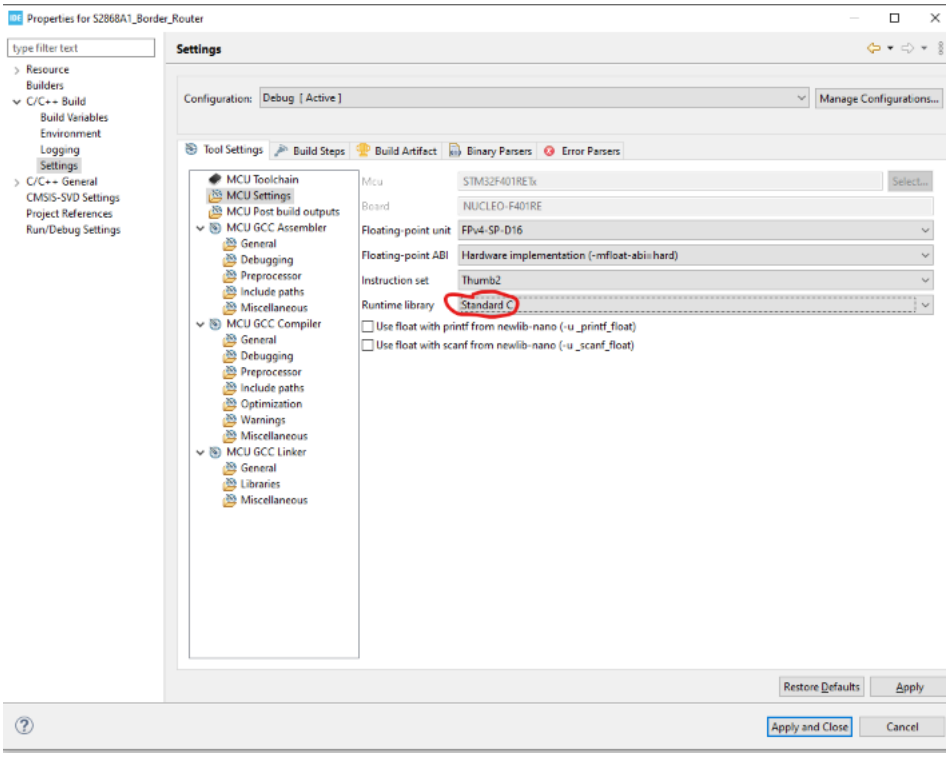


***Figure 25** Library to be enabled for Contiki-NG based projects in STM32CubeIDE 1.6.x*

- If a project has been generated for P2P, to modify it for a Contiki-NG based application that uses LED or Button, it is required to deselect any application and save the .ioc project file before choosing the new Application.
- If you build a project with the BSP only (no application, no Contiki-NG), to successfully use the radio you must add, at your application level, the following declaration: volatile int irq_disable_cnt = 0;
- For the **NUCLEO-F303ZE** Board, the Arduino Pin “A5”, corresponding to the “PD13” MCU pin, should be configured as “GPIO_EXTI13” for the “S2868A1_GPIO3”, “S2868A2_GPIO3” and “S2915A1_GPIO3” for S2868A1, S2868A2 and S2915A1, respectively. At the same time, the user button, corresponding to “PC13”, should be also configured as “GPIO_EXTI13”. Given the fact that only one pin can be configured as “GPIO_EXTI13”, we exclude this board from the list of boards, or you can use different S2868A1 or S2868A2 or S2915A1 GPIO pin for the Interrupt event.
- For the **NUCLEO-F302R8, NUCLEO-L412RB-P**, **NUCLEO-L452RE-P** and **NUCLEO-L433RC-P** Board, the user should configure “SPI2” with the following details:

    a. SPI2_MOSI ← D11 (PB15)

    b. SPI2_MISO ← D12 (PB14)

    c. SPI2_SCK ← D13 (PB13)

  With this setting, we have the following problem. The “D13”, corresponding to “PB13” MCU pin, is also used by the P2P example as “GPIO_Output” for the “BSP LED”. Due to the fact that we can configure “PB13” either as “GPIO_Output” or as “SPI2_SCK”. So, for LED indication, one can use LED of the X-NUCLEO (S2868A1 or S2868A2 or S2915A1) connected to PC7 by mounting R17 with 0-ohm resistor.

  Following will be change in software, to reflect (S2868A1 or S2868A2 or S2915A1) LED instead of BSP LED on NUCLEO-F302R8 or NUCLEO-L412RB-P or NUCLEO-L452RE-P or NUCLEO-L433RC-P.

1. Replace BSP_LED_Init (LED2) with S2868A1_LED_Init, S2868A2_LED_Init and S2915A1_LED_Init for S2868A1_LED, S2868A2_LED and S2915A1_LED, respectively, in MX_SUBG2_P2P_Init.
2. Replace BSP_LED_Toggle (LED2) with S2868A1_LED_Toggle, S2868A2_LED_Toggle and S2915A1_LED_Toggle, for S2868A1_LED, S2868A2_LED_Toggle and S2915A1_LED_Toggle, respectively.
3. Replace BSP_LED_On (LED2) with S2868A1_LED_On, S2868A2_LED_On and S2915A1_LED_On for S2868A1_LED, S2868A2_LED and S2915A1_LED, respectively.
4. Replace BSP_LED_Off (LED2) with S2868A1_LED_Off, S2868A2_LED_Off and S2915A1_LED_Off for S2868A1_LED, S2868A2_LED and S2915A1_LED, respectively.

- For the **NUCLEO-F103RB** Board, the user should configure “SPI1” with the following details:
  a. SPI1_MOSI ← D11 (PA7)
  b. SPI1_MISO ← D12 (PA6)
  c. SPI1_SCK ← D13 (PA5)

  Following will be change in software, to reflect (S2868A1 or S2868A2 or S2915A1) LED instead of BSP LED on NUCLEO-F103RB.

  1. Replace BSP_LED_Init (LED2) with S2868A1_LED_Init, S2868A2_LED_Init and S2915A1_LED_Init for S2868A1_LED, S2868A2_LED and S2915A1_LED, respectively, in MX_SUBG2_P2P_Init.
  2. Replace BSP_LED_Toggle (LED2) with S2868A1_LED_Toggle, S2868A2_LED_Toggle and S2915A1_LED_Toggle, for S2868A1_LED, S2868A2_LED_Toggle and S2915A1_LED_Toggle, respectively.
  3. Replace BSP_LED_On (LED2) with S2868A1_LED_On, S2868A2_LED_On and S2915A1_LED_On for S2868A1_LED, S2868A2_LED and S2915A1_LED, respectively.
  4. Replace BSP_LED_Off (LED2) with S2868A1_LED_Off, S2868A2_LED_Off and S2915A1_LED_Off for S2868A1_LED, S2868A2_LED and S2915A1_LED, respectively.

- For platforms with very little FLASH memory it may happen that the generated binary does not fit into it. If using STM32CubeIDE try the Release configuration instead of the Debug one. Other possibilities are to disable not required components (via STM32CubeMX) or to change some parameters at compile time (project-conf.h file and Contiki-NG configuration files).
- In MDK-ARM projects for STM32H7 series, if no log message is printed on the serial terminal, enable the Use MicroLib option in the project settings.
- On dual-core STM32 series this expansion software can be used on both cores but exclusively.

10 References

[1] UM2669 – User Manual – *Getting started with X-CUBE-SUBG2, Sub-1 GHz RF software expansion for STM32Cube*

[2] Contiki-NG GitHub repository

11 Revision history

Table 6 Document revision history

Date	Version	Changes
21-Nov-2019	1	Initial release
04-Feb-2020	2	Add S2868A2 and S2915A1 support Improvement in the table of configuration of pins
11-Nov-2020	3	Added Contiki-NG middleware and related applications
11-Feb-2021	4	Updated info for v4.0.0
21-Apr-2021	5	Updated info for v4.1.0

IMPORTANT NOTICE – PLEASE 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 acknowledgement.
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, please refer to 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.