# Getting started with the STMicroelectronics X-CUBE-WB05N software package for STM32CubeMX

## Introduction

This document provides the guidelines to configure and use the X-CUBE-WB05N software package V1.1.0 for STM32CubeMX (minimum required version V6.12.0). The document contains the description of the provided sample applications, the description of the steps required to configure a generic project using the Bluetooth LE middleware as well as of the steps to configure and use the sample application provided in the package.

Information and documentation related to the ST STM32WB05xN network processor, the X-NUCLEO-WB05KN1 expansion board and the X-CUBE-WB05N expansion software for Bluetooth Low Energy are available on www.st.com.

# Contents

# List of figures

# 1 Acronyms and abbreviations

**Table 1: list of acronyms**

| Acronym | Description |
|---|---|
| BLE | Bluetooth Low Energy |
| FOTA | Firmware Over The Air |
| HAL | Hardware Abstraction Layer |
| HID | Human Interface Device |
| IOT | Internet Of Things |
| IP | Internet Protocol |
| LAN | Local Area Network |
| NVIC | Nested Vectored Interrupt Controller |
| PCB | Printed Circuit Board |
| RTC | Real Time Clock |
| RTOS | Real Time Operating System |
| SPI | Serial Peripheral Interface |
| UID | Unique Identifier |
| URL | Uniform Resource Locator |
| U(S)ART | Universal (Synchronous) Asynchronous Receiver Transmitter |
| USB | Universal Serial BUS |
| TCP | Transmission Control Protocol |
| HCI | Host Controller Interface |
| DTM | Direct Test Mode |

# 2 What is STM32Cube?

STM32Cube is a combination of a full set of PC software tools and embedded software blocks running on STM32 microcontrollers and microprocessors:

- STM32CubeMX configuration tool for any STM32 device; it generates initialization C code for Cortex-M cores and the Linux device tree source for Cortex-A cores
- STM32CubeIDE integrated development environment based on open-source solutions like Eclipse or the GNU C/C++ toolchain, including compilation reporting features and advanced debug features
- STM32CubeProgrammer programming tool that provides an easy-to-use and efficient environment for reading, writing and verifying devices and external memories via a wide variety of available communication media (JTAG, SWD, UART, USB DFU, I2C, SPI, CAN, etc.)
- STM32CubeMonitor family of tools (STM32CubeMonRF, STM32CubeMonUCPD, STM32CubeMonPwr) to help developers customize their applications in real-time
- STM32Cube MCU and MPU packages specific to each STM32 series with drivers (HAL, low-layer, etc.), middleware, and lots of example code used in a wide variety of real-world use cases
- STM32Cube expansion packages for application-oriented solutions

# 3 License

The software provided in this package is licensed under Software License Agreement SLA0095.

# 4 Sample Applications Description

In this section a short overview of the sample applications included in the X-CUBE-WB05N pack is provided.
The sample applications:

- are ready-to-use projects that can be generated through the STM32CubeMX for any board equipped with an STM32 MCU and using the STM32WB05xN chip.
- show the users how to use the Bluetooth LE APIs, provided by the STM32WB05N middleware, for correctly initialize and use a Bluetooth LE device.

## 4.1 SensorDemo_BLESensor-App

This application is an example showing how to implement an application tailored for interacting with the "ST BLE Sensor" app for Android/iOS devices.
The "ST BLE Sensor" app is freely available on both Play Store and iTunes.
The source code of the "ST BLE Sensor" app is also available on GitHub for both iOS and Android devices.
After establishing the connection between the STM32 board and the smartphone:

- the temperature and the pressure emulated values are sent by the STM32 board to the mobile device and are shown in the ENVIRONMENTAL tab;
- the plot of the emulated data (temperature, pressure, accelerometer, gyroscope and magnetometer) sent by the board are shown in the PLOT DATA tab;
- in the RSSI & Battery tab the RSSI value is shown.

## 4.2 SampleApp

This sample application shows how to simply use the Bluetooth LE Stack.
It provides the user with a complete example on how to perform an ATT MTU exchange procedure, in order that the server and the client can agree on the supported max MTU.
To test this application two STM32 Nucleo boards with their respective X-NUCLEO-WB05KN1 STM32 expansion boards should be used. After flashing both the STM32 boards, one board configures itself as Bluetooth LE Server-Peripheral device, while the other as Bluetooth LE Client-Central device.
After the connection between the two boards is established (signaled by the LD2 LED blinking on the Client-Central device), pressing the USER button on one board, the LD2 LED on the other one gets toggled and vice versa.
If you have only one STM32 Nucleo board, you can the “BLE Scanner” app (available for both the Android and the iOS devices) as Bluetooth LE Client-Central device.

## 4.3 Beacon

This example application shows how to use the STM32WB05xN STM32 expansion board to implement a Beacon device.
A Beacon device is a smart Bluetooth Low Energy device that transmits a small data payload at regular intervals using Bluetooth advertising packets.
Beacons are used to mark important places and objects. Typically, a beacon is visible to a user's device from a range of a few meters, allowing for highly context-sensitive use cases.
To locate the beacon, it is necessary to have a scanner application running on a Bluetooth LE -capable smartphone, such as “BLE Scanner” app (available for both the Android and the iOS devices).

## 4.4 BLE_FOTA

The BLE_FOTA application shows how to use the dual bank flash features to allow Firmware-Over-the-Air update of a running program without using the BootLoader.
Example Description:
Before starting the boot procedure (which can start from bank1 or bank2), it is necessary to change STM32 MCU user bytes. Once settings have been changed, the BLE_FOTA receives the new firmware from the STBLESensor application, saves it on one flash bank (either bank1 or bank2) and performs a reboot executing the new code saved on the other flash bank. Even if the BootLoader allows more flexibility as you can split the flash memory into any number of regions, each program related to a specific region can run in that region only. The BLE_FOTA application, however, can swap among different flash banks and each program can run in any flash memory bank.

- Running BLE_FOTA application:

  1. Download ST BLE Sensor Mobile App -IOS/Android
  2. Save the X-CUBE-WB05N_V1.1.0\Utilities\FOTA_Custom_DB_Entry.json file in the Smartphone memory.
  3. Open ST BLE Sensor App and go to Settings, Click on option "Add Custom Fw DB Entry" and select "FOTA_Custom_DB_Entry.json" file.
  4. To perform the FOTA connect to the device visible as "BLEFOTA" in the mobile app.
  5. Go to Settings and Select "Firmware Upgrade" option.
  6. Select the Binary file to be flashed from Smartphone's local storage.

This application requires the following linker settings:

- CSTACK minimum size 0xC00
- HEAP minimum size 0x200

## 4.5 Virtual_COM_Port

Virtual_COM_Port is the application to be used with the STM32CubeMonRF latest version supporting STM32WB05N.
STM32CubeMonitor-RF (STM32CubeMonRF) is a software tool allowing the test of the radio performances of STM32WB based hardware devices on Bluetooth® Low Energy and 802.15.4 technologies.

## 4.6 IFRStack_Updater

The IFRStack_Updater sample application can be used to update the STM32WB05xN firmware stack to the latest version or to change the IFR configuration.
To change the IFR configuration, the `#define STM32WB05N_DEV_CONFIG_UPDATER` must be defined (see file `app_stm32wb05n.h`).
The IFR parameters can be changed editing the file `Middlewares\ST\STM32WB05N\hci\stm32wb05n_devConfig.c`.

WARNING *Be sure you know what you are doing when modifying the IFR settings. This operation should be performed only by expert users.*

To update the STM32WB05N firmware stack, the `#define STM32WB05N_STACK_UPDATER` must be defined (see file `app_stm32wb05n.h`).
The firmware image used by the application is contained in the `FW_IMAGE` array defined in file

`update_fw_image.c`. In the same file, the information on the DTM SPI and the FW versions is reported.

This application requires the following linker settings:

- CSTACK minimum size 0xC00
- HEAP minimum size 0x200

# 5 Installing the X-CUBE-WB05N pack in STM32CubeMX

After downloading (from www.st.com), installing and launching the STM32CubeMX (V≥6.12.0), the X-CUBE-WB05N pack can be installed in few steps.

1. From the top menu bar, select Help → Manage embedded software packages.

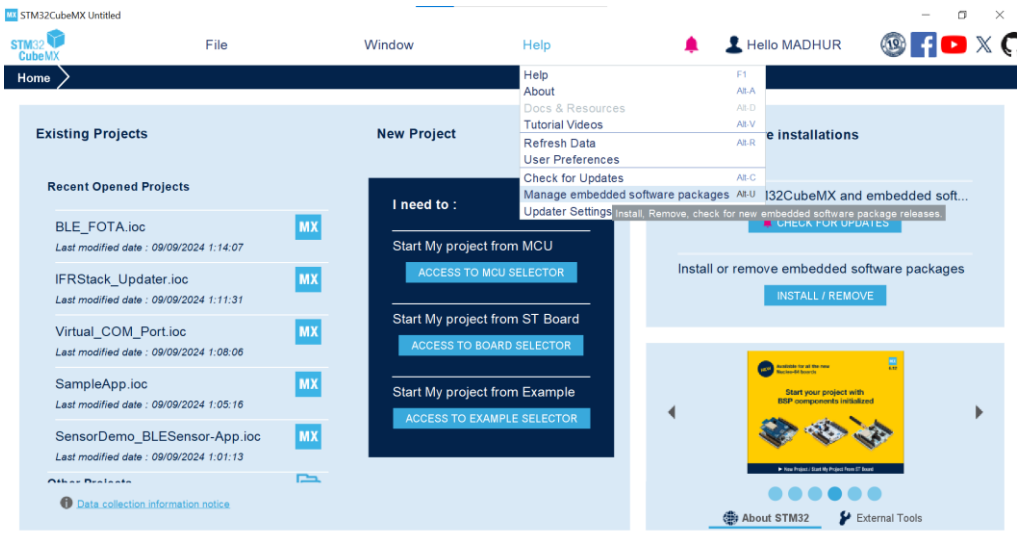


***Figure 1*** *Managing embedded software packs in STM32CubeMX*

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-WB05N pack.


***Figure 2*** *Installing the X-CUBE-WB05N pack in STM32CubeMX*

3. Select the X-CUBE-WB05N pack checking the corresponding box and install it pressing the Install Now button. After accepting the license terms and 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-WB05N pack in STM32CubeMX*

# 6 Starting a new project

After launching the STM32CubeMX, click either the ACCESS TO MCU SELECTOR or the ACCESS TO BOARD SELECTOR button in the GUI to start a project for your MCU or board.

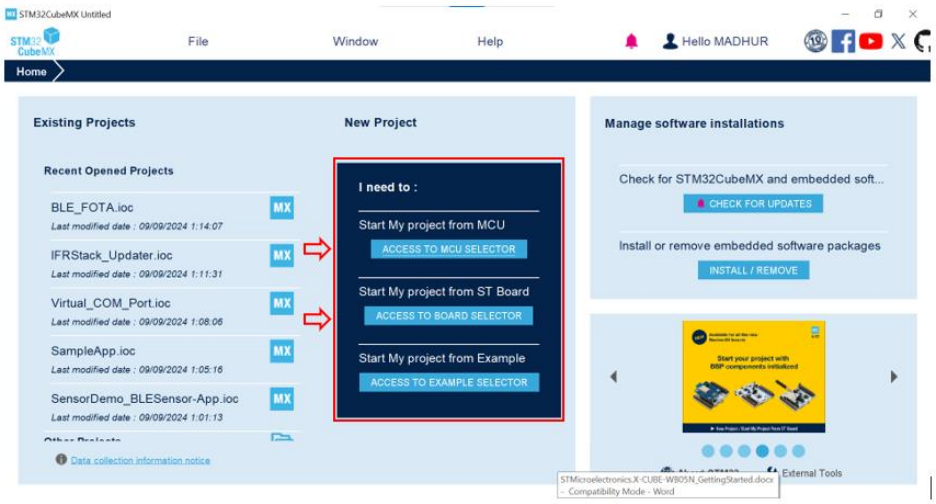


***Figure 4*** *STM32CubeMX main page*

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

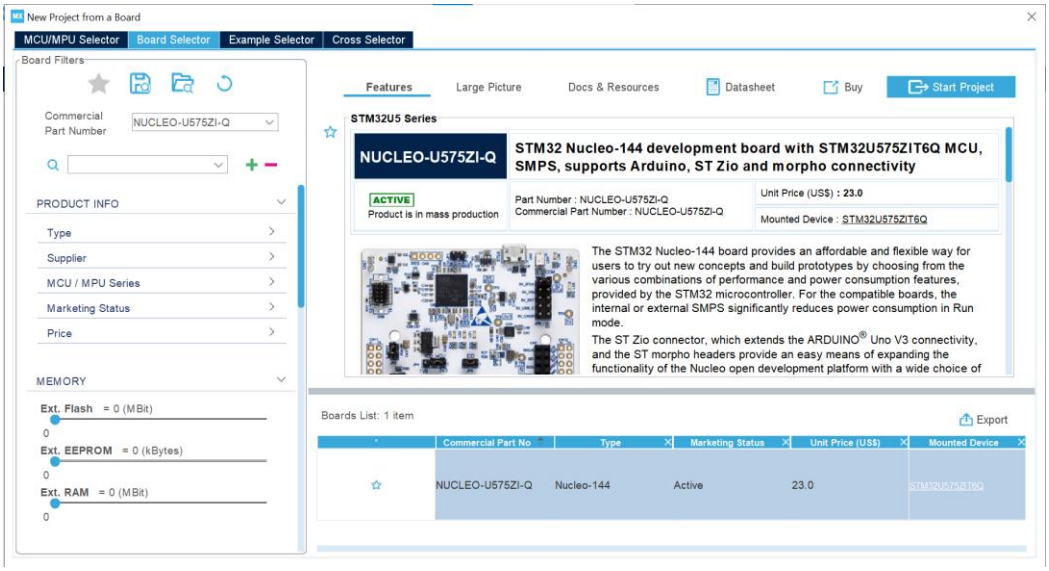


***Figure 5*** *STM32CubeMX* ***MCU/Board Selector*** *windows*

After selecting the MCU or the Board, the selected STM32 pinout will appear (the user can either choose to Initialize all peripherals with their default Mode or not). From this window the user can set up the project, by adding one or more Software Packs and peripherals and configuring the clock.

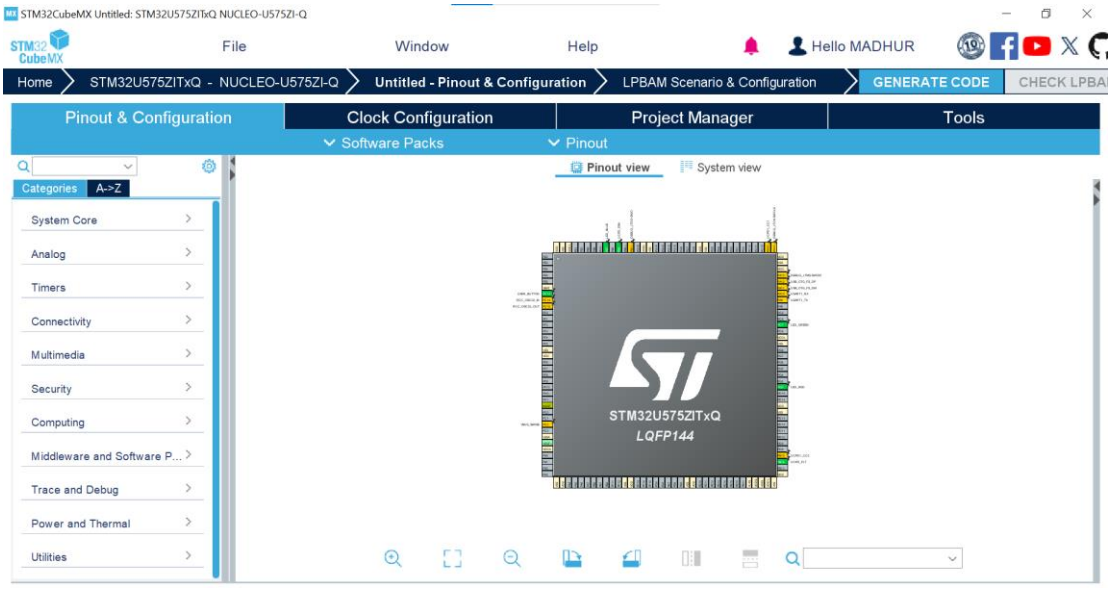


***Figure 6** STM32CubeMX **Configuration** window*

To add the X-CUBE-WB05N additional software to the project, the Software Packs → Select Components menu item must be selected.
From the Software Packs 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.

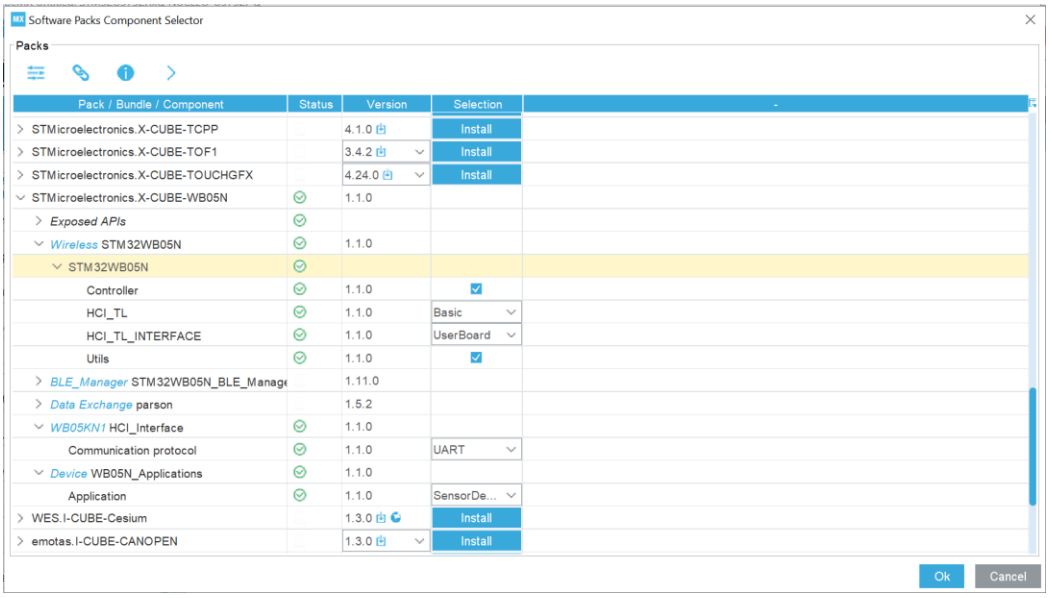


***Figure 7** STM32CubeMX **Software Packs Component Selector** window*

# 7 HCI_TL and HCI_TL_INTERFACE Configuration

The HCI_TL (Host Controller Interface Transport Layer) and the HCI_TL_INTERFACE (Host

Controller Interface Transport Layer Interface) are the interfaces between the HAL/BSP layer and both the Middleware Core Layer and the Application Layer.

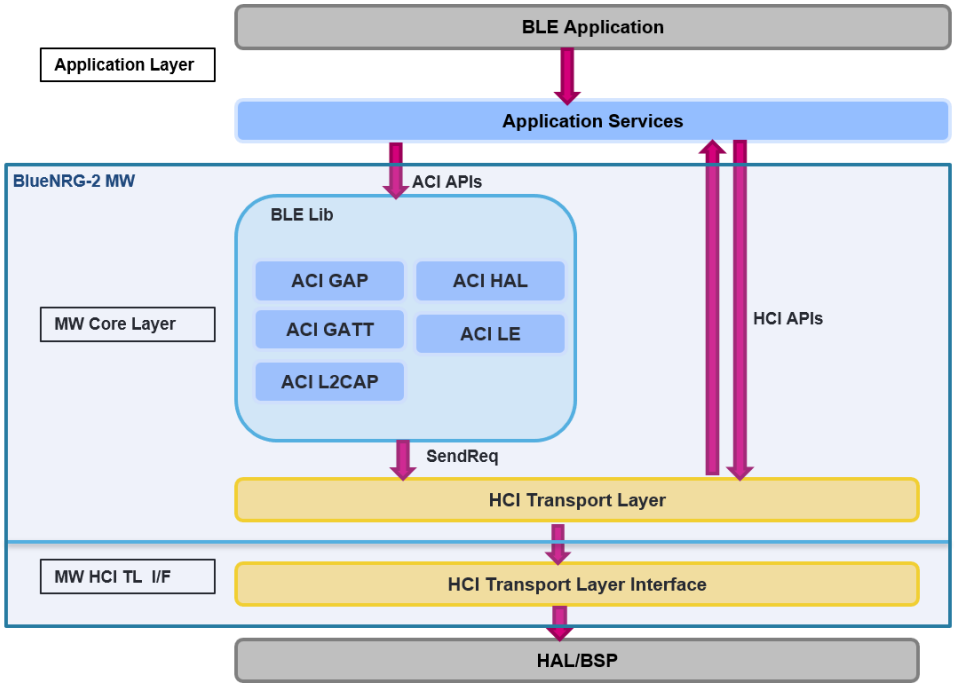


**Figure 8** STM32WB05xN software block scheme

Two different configurations may be used for both the components.

- HCI_TL
  - *Basic* the user can use a basic set of already implemented APIs
  - *Template* the user can implement his own APIs for building his own customized HCI TL
- HCI_TL_INTERFACE
  - *UserBoard* the user can use a basic set of already implemented APIs
  - *Template* the user can implement his own APIs for building his own customized HCI TL Interface.

For generating a ready to work sample application, the *Basic* and *UserBoard* configurations must be selected.

# 8 HCI Interface Communication protocol

There are two options available for HCI_TL_INTERFACE communication protocol -:

- SPI
- UART

This is the communication protocol that will be used by the Host STM32 to communicate with STM32WB05xN network processor on X-NUCLEO-WB05KN1. Depending on the requirements, the user can select either one of them.

**Note**: Appropriate DTM binary image should be flashed on X-NUCLEO-WB05KN1 depending on the communication protocol selected. Details about the DTM binary images present in the X-CUBE-WB05N\Utilities folder are present in X-CUBE-WB05N user manual. The X-CUBE-WB05N user manual section “How to emulate the STM32WB05xN device with X-NUCLEO-WB05KN1” also contains the procedure to program the X-NUCLEO-WB05KN1 using STM32CubeProgrammer and STlink-V3SET debugger.

By default, the X-NUCLEO-WB05KN1 is factory programmed with UART based DTM binary

present at Utilities\DTM_Location_Essentials_Configurations\ DTM_UART_WITH_UPDATER.bin. To use SPI protocol user must re-flash the X-NUCLEO-WB05KN1 with SPI based DTM binary present at Utilities\DTM_Location_Essentials_Configurations\ DTM_SPI_WITH_UPDATER.bin.

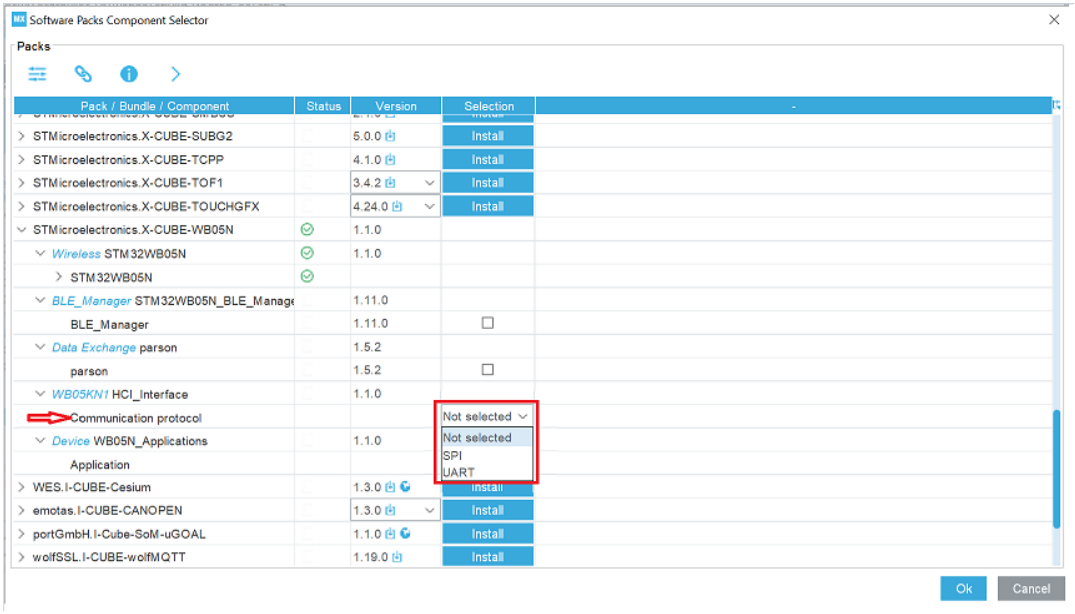


***Figure 9** HCI_Interface communication protocol option selection*

# 9 STM32 Configuration Steps

The X-NUCLEO-WB05KN1 interfaces with the STM32 microcontroller via the SPI or UART pins.

Hence, assuming a user wants to interface the ST X-NUCLEO-WB05KN1 expansion board with a STM32 NUCLEO board (e.g. a NUCLEO-U575ZI-Q, the following steps must be executed before generating a project:

1. Stack the X-NUCLEO-WB05KN1 board on top of the NUCLEO board (Eg. NUCLEO-U575ZI-Q)
2. Make sure the flash correct DTM binary is Flashed on X-NUCLEO-WB05KN1 board – SPI/UART based. Refer Section 8 of this document for details.
3. On some STM32 Nucleo 64/144 pins, it could be required to bridge certain pins of the Arduino connector to correctly set the required signals. Please refer to the respective NUCLEO board schematic and verify the required connections with X-NUCLEO-WB05KN1 (MB2160) schematic available on st.com.

***Figure 10*** *X-NUCLEO-WB05KN1*

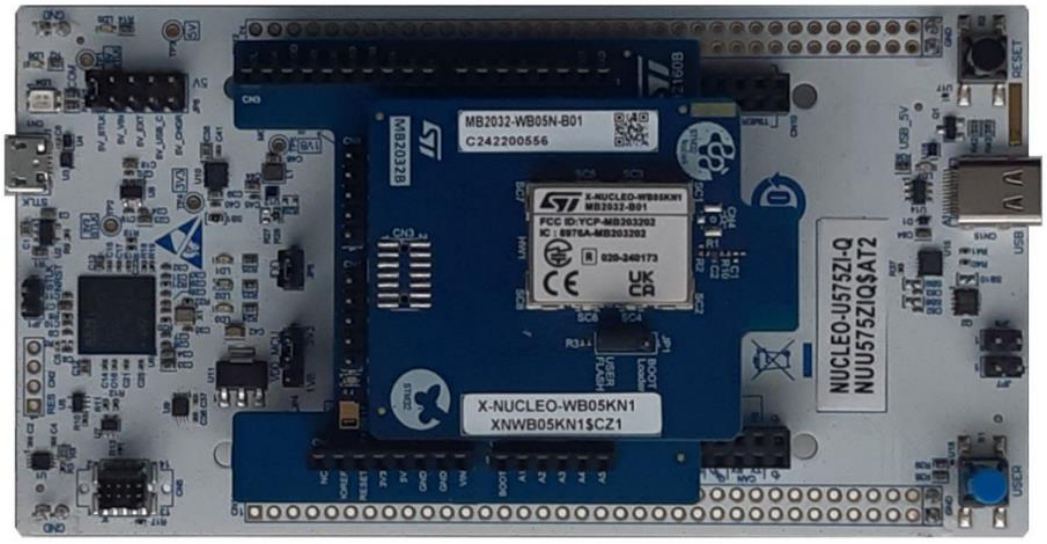


***Figure 11*** *STM32 NUCLEO-U575ZI-Q board and X-NUCLEO-WB05KN1*

| Left Connectors | | | | |
|---|---|---|---|---|
| X-NUCLEO connector | Pin number | Signal name | X-NUCLEO-WB05KN1 signal(1) | NUCLEO-U575ZI-Q MCU port(1) |
| CN5 Power | 1 | - | - | 5V_IN test |
| | 2 | IOREF | **IOREF** | **IOREF** |
| | 3 | NRST | RSTN | NRST |
| | 4 | 3V3 | **3V3** | **3V3** |
| | 5 | 5V | - | 5V |
| | 6 | GND | **GND** | **GND** |
| | 7 | GND | **GND** | **GND** |
| | 8 | VIN | - | VIN |
| CN6 Analog | 1 | A0 | **PA10/BOOT/SPI_IRQ** | **PA3** |
| | 2 | A1 | - | PA2 |
| | 3 | A2 | - | PC3 |
| | 4 | A3 | - | PB0 |
| | 5 | A4 | - | PC1 |
| | 6 | A5 | - | PC0 |

| Right Connectors | | | | |
|---|---|---|---|---|
| X-NUCLEO-WB05KN1 signal(1) | NUCLEO-U575ZI-Q MCU port(1) | Signal name | Pin number | X-NUCLEO connector |
| - | PB8 | D15 | 10 | CN3 Digital |
| - | PB9 | D14 | 9 | |
| - | VREFP_ARD | AVDD | 8 | |
| **GND** | **GND** | GND | 7 | |
| **PB3/SPI3_SCK** | **PA5/SPI_SCK** | D13 | 6 | |
| **PA8/SPI3_MISO** | **PA6/SPI_MISO** | D12 | 5 | |
| **PA11/SPI3_MOSI** | **PA7/SPI_MOSI** | D11 | 4 | |
| **PA9/SPI3_NSS** | **PD14/SPI_CS** | D10 | 3 | |
| PB0/UART_RX | PD15 | D9 | 2 | |
| PA1/UART_TX | PF12 | D8 | 1 | |
| **RSTN** | **PF13** | D7 | 8 | CN4 Digital |
| - | PE9 | D6 | 7 | |
| - | PE11 | D5 | 6 | |
| LED | PF14 | D4 | 5 | |
| - | PE13 | D3 | 4 | |
| PA1/UART_TX | PF15 | D2 | 3 | |
| **PB0/UART_RX** | **PG7/UART_TX** | D1 | 2 | |
| **PA1/UART_TX** | **PG8/UART_RX** | D0 | 1 | |

1. Default connected signals are in bold.

***Figure 12** X-NUCLEO-WB05KN1 and NUCLEO-U575ZI-Q connection details*

## 9.1 Use of Expansion Software without sample applications

This section outlines how to configure STM32CubeMX when the use of the sample applications is not required. With such setup, only middleware and driver layers will be configured. This setup is useful when the user does not intend to leverage the sample applications provided in the package. Depending on the HCI_Interface communication protocol selected in Select Components tab there are two cases:

Case 1: **SPI**
From the Pinout & Configuration tab:

- from the ∨ Pinout → Clear Pinouts menu option reset the state of all pins (only for Nucleo 144);
- from the Connectivity > menu:
  - check that the ETH is disabled (only for Nucleo 144)
  - enable the SPI1 in Full-Duplex Master Mode

Check that the enabled SPI uses the following pins:

| | Arduino PINs |
|---|---|
| SPI_MISO | D12 |
| SPI_MOSI | D11 |
| SPI_SCK | D13 |

In case different pins have been enabled by default, change them using the Alternative pins (if available) or bridge the enabled ones and the ones reported in the table above.
From the Pinout view set:

| Signal Name | Nucleo 64 | Nucleo 144 | Arduino PINs |
|---|---|---|---|
| SPI_IRQ | GPIO_EXTI0 | GPIO_EXTI3 | A0 |
| SPI_CS | GPIO_Output | GPIO_Output | D10 |
| Reset | GPIO_Output | GPIO_Output | D7 |

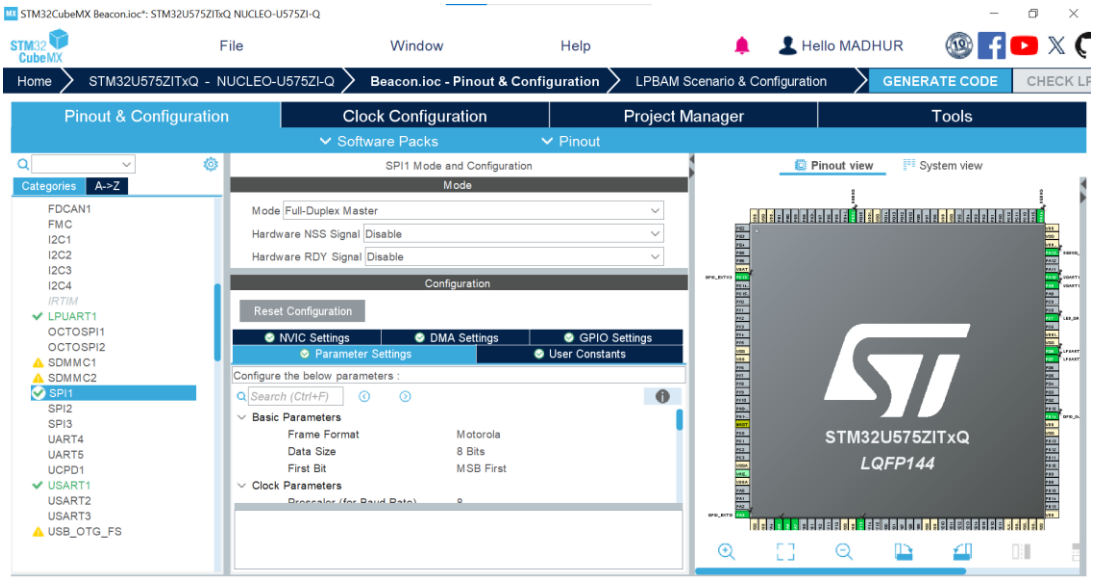


***Figure 13*** *Pinout view*

From the Software Packs category item, click on the STMicroelectronics.X-CUBE-WB05N.x.y.z pack. Check the Wireless STM32WB05xN box and set the following Platform Settings:

| Name | IPs or Components | Found solutions | | BSP Api |
|---|---|---|---|---|
| | | Nucleo 64 | Nucleo 144 | |
| Exti Line | GPIO:EXTI | PA0 | PA3 | HAL_EXTI_DRIVER |
| BUS IO driver SPI | SPI:Full-Duplex Master | SPI1 | SPI1 | BSP_BUS_DRIVER |
| CS Line | GPIO:Output | PB6 | PD14 | Unknown |
| Reset Line | GPIO:Output | PA8 | PF13, PG12 | Unknown |

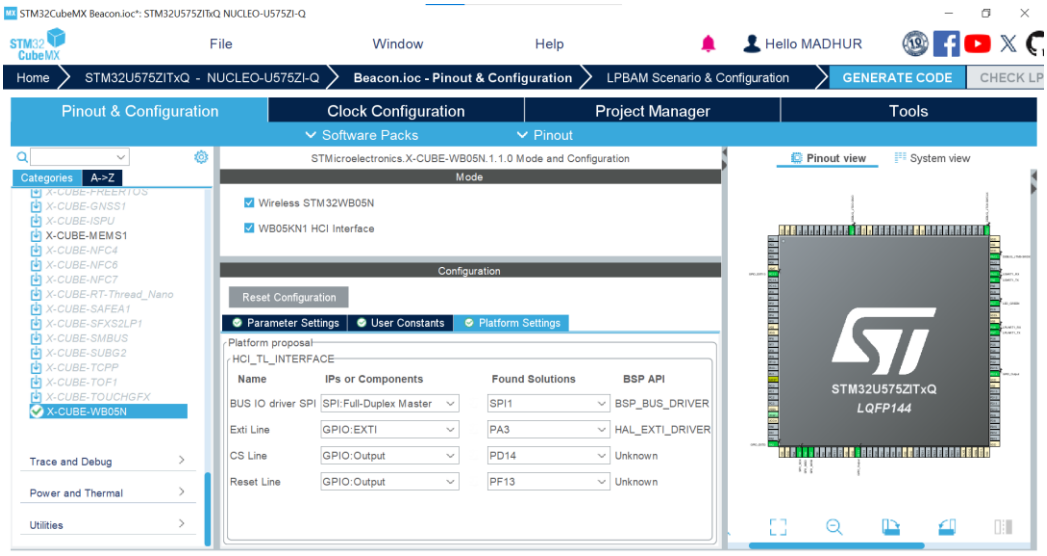


***Figure 14*** *STMicroelectronics.X-CUBE-WB05N Mode and Configuration view*

From the System view, click on NVIC button under System Core category to enable the EXTI line interrupt:

| Nucleo 64 | Nucleo 144 |
|---|---|
| EXTI line 0 interrupt | EXTI line 3 interrupt |

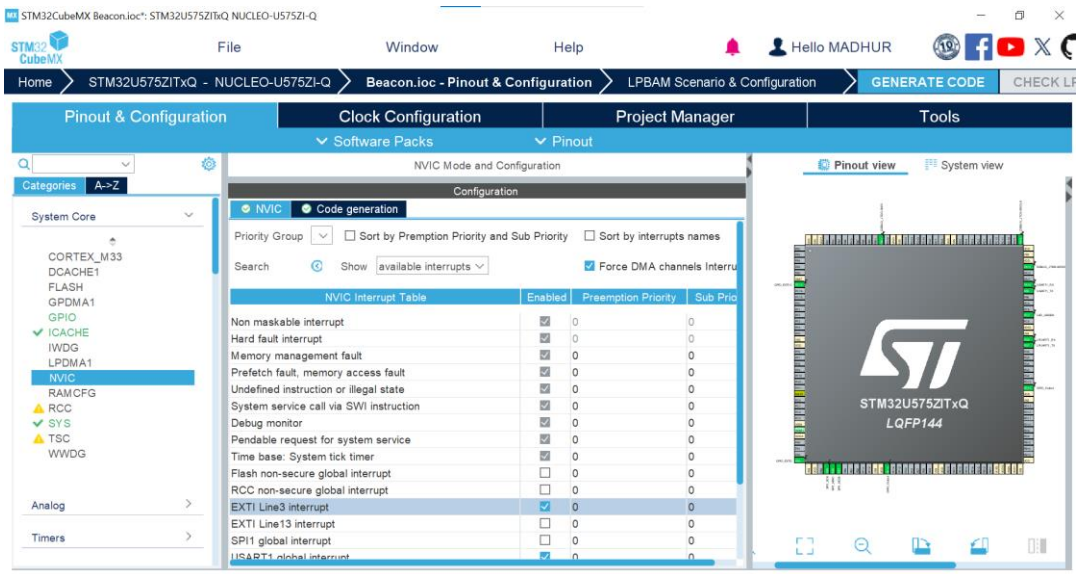


***Figure 15** NVIC Mode and Configuration view*

From the System view, click on SPIx button under Connectivity category and:

- check that the Data size is 8 Bits;
- set the Prescaler (for Baud Rate) to a value so that the HClock/Prescaler is less or equal to 10 Mbits/s (the maximum supported SPI speed);
- set the Clock Phase (CPHA) to 2 Edge
- set the Clock Polarity (CPOL) to High

Once all the above-described steps have been performed, from the Project Manager tab the Project Name, the Project Location, the Toolchain/IDE, the Firmware Package Name and Version and so on can be set.

In the Advanced Settings tab, the following options are set by default. You can use or change them according to your needs.

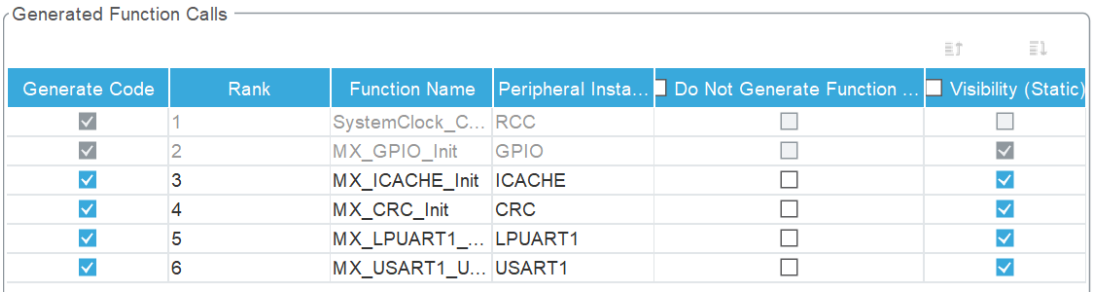


***Figure 16** Advanced Settings*

the source code of the project using the **STMicroelectronics X-CUBE-WB05N** software can be generated clicking the GENERATE CODE button.

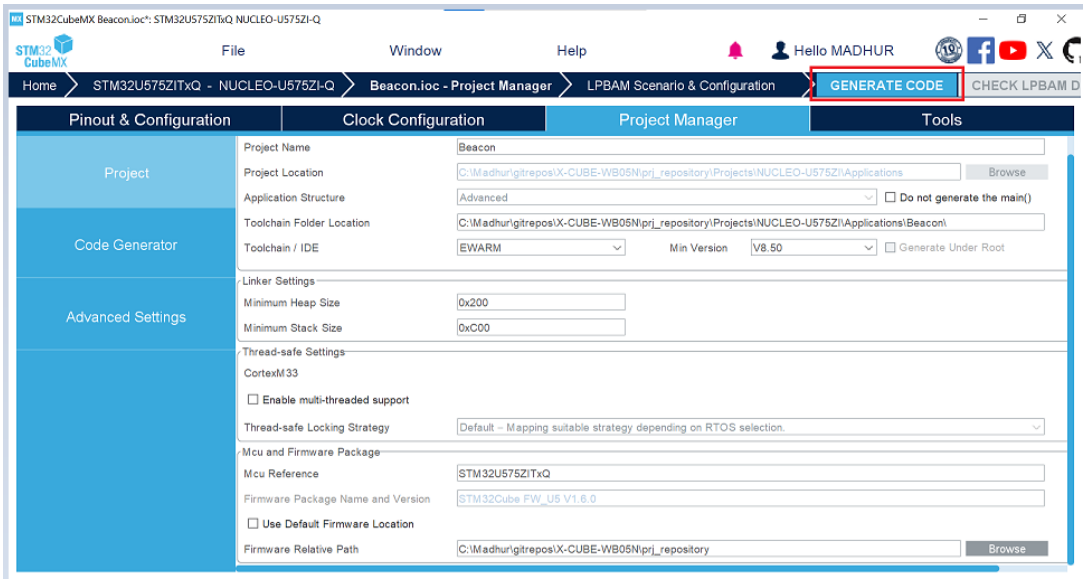


***Figure 17*** *Project Manager view*

Case 2: **UART**
From the Pinout & Configuration tab:

- from the ᵛ Pinout → Clear Pinouts menu option reset the state of all pins (only for Nucleo 144);
- from the Connectivity > menu:
  - Check that the ETH is disabled (only for Nucleo 144)
  - Enable the LPUART1 in Asynchronous Mode (only for Nucleo 144)
    Check that the enabled LPUART1 uses the following pins:

| | **Arduino PINs** |
|---|---|
| LPUART1_TX | D1 |
| LPUART1_RX | D0 |

  - Enable the USART1 in Asynchronous mode (only for Nucleo 64).

| | **Arduino PINs** |
|---|---|
| USART1_TX | D8 |
| USART1_RX | D2 |

**NOTE**: In some Nucleo - 64 boards, user needs to use Jumper wires to Connect USART1 Rx/Tx with X-NUCLEO-WB05KN1(MB2160). Refer respective boards schematic to confirm the connections.
In case different pins have been enabled by default, change them using the Alternative pins (if available) or bridge the enabled ones and the ones reported in the table above.

From the Pinout view set:

| **Nucleo 64** | | **Nucleo 144** | |
|---|---|---|---|
| PA8 | GPIO_Output | PF13, PG12 | GPIO_Output |

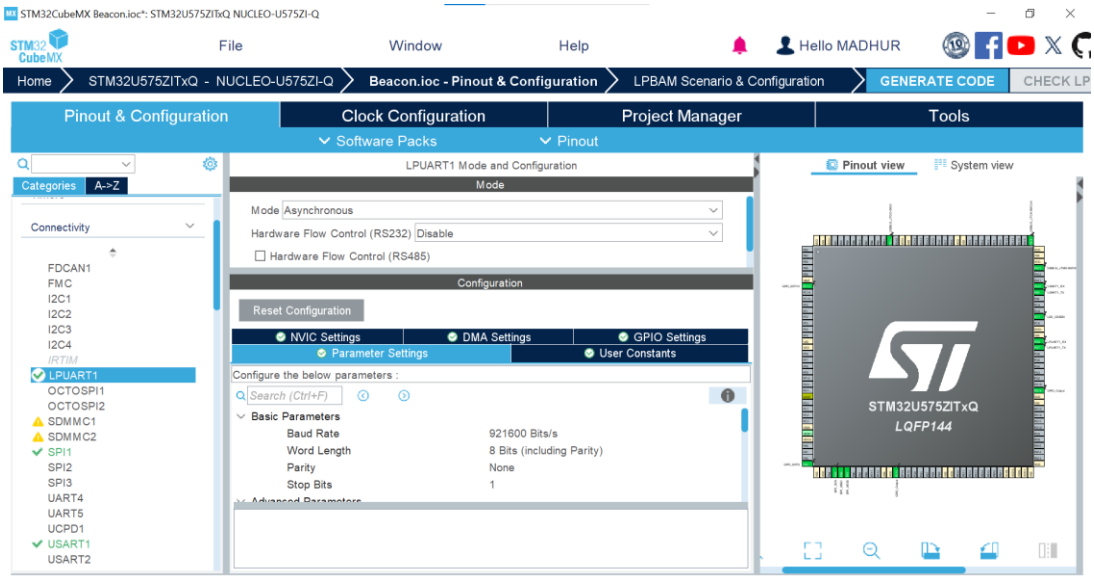


***Figure 18** Pinout view*

From the Software Packs category item, click on the STMicroelectronics.X-CUBE-WB05N.x.y.z pack. Check the Wireless STM32WB05xN box and set the following Platform Settings:

| Name | IPs or Components | Found solutions | | BSP Api |
|---|---|---|---|---|
| | | Nucleo 64 | Nucleo 144 | |
| BUS IO driver | LPUART:Asynchronous, USART:Asynchronous | USART1 | LPUART1 | BSP_BUS_DRIVER |
| Reset Line | GPIO:Output | PA8 | PF13, PG12 | Unknown |

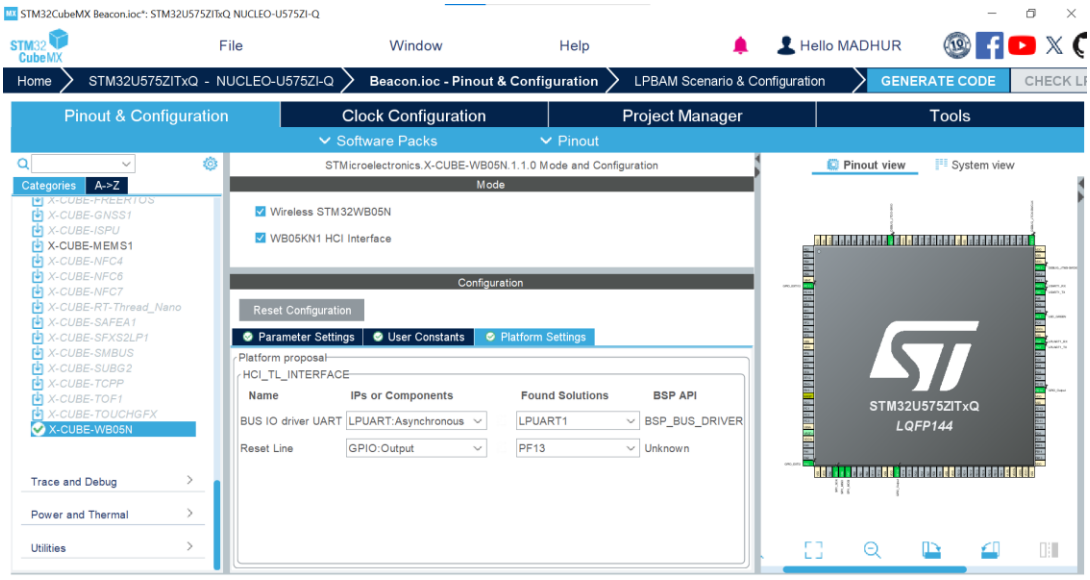


***Figure 19** STMicroelectronics.X-CUBE-WB05N Mode and Configuration view*

From the System view, click on NVIC button under System Core category to **disable** the EXTI

line interrupt:

| Nucleo 64 | Nucleo 144 |
|---|---|
| EXTI line 0 interrupt | EXTI line 3 interrupt |

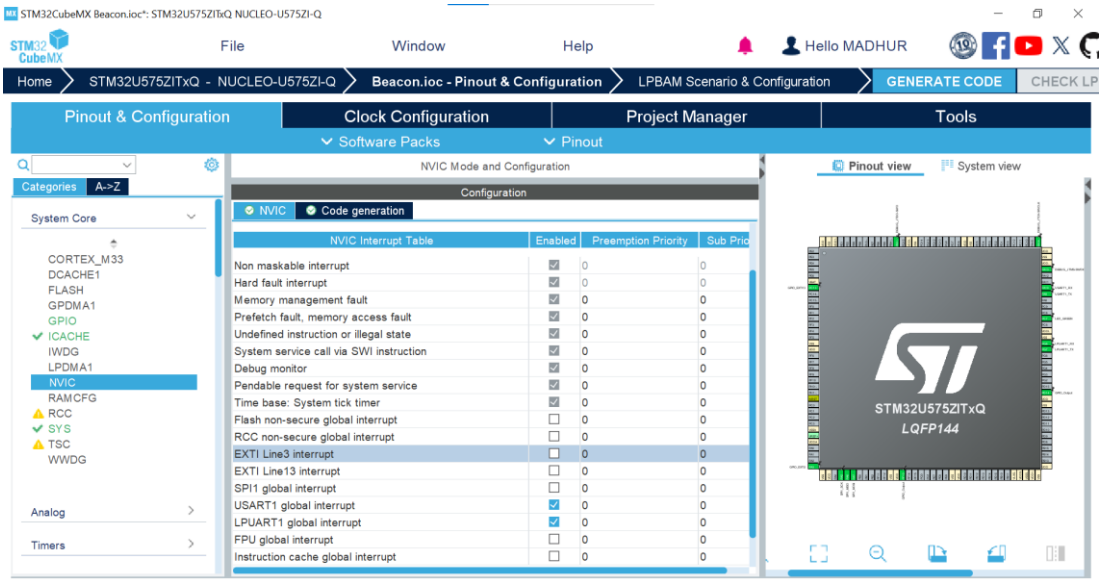


***Figure 20** NVIC Mode and Configuration view*

From the System view, click on LPUART1 button under Connectivity category and:

- check that the Word Length is 8 Bits (including parity);
- set the Baudrate to 921600 Bits/sec
- set the Stop Bit to 1
- set the parity to None

Once all the above described steps have been performed, from the Project Manager tab the Project Name, the Project Location, the Toolchain/IDE, the Firmware Package Name and Version and so on can be set.

In the Advanced Settings tab, the following options are set by default. You can use or change them according to your needs.

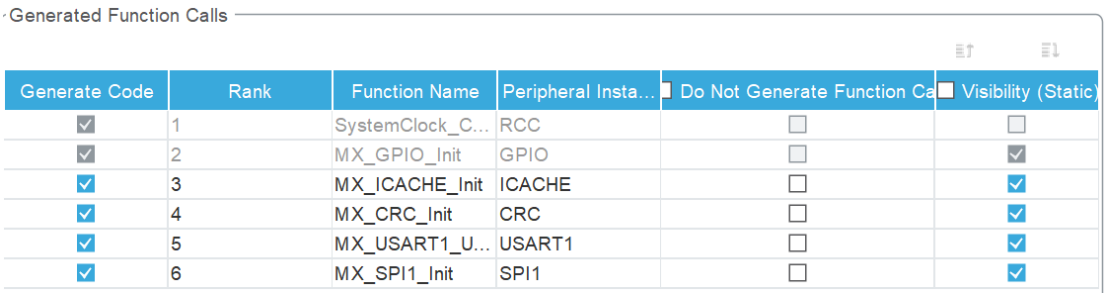


***Figure 21** Advanced Settings*

the source code of the project using the **STMicroelectronics X-CUBE-WB05N** software can be generated clicking the GENERATE CODE button.

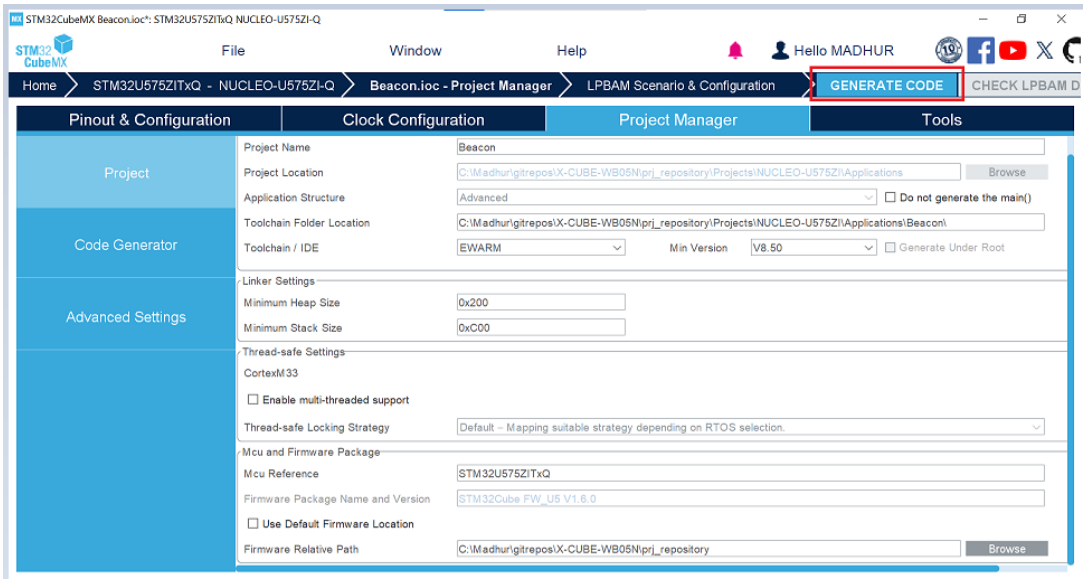


***Figure 22*** *Project Manager view*

## 9.2 Use of Expansion Software with sample applications

This section outlines how to configure STM32CubeMX when the use of the sample applications is required. With such setup, all the components of the expansion software package, including applications, will be properly configured. Depending on the HCI_Interface communication protocol selected in Select Components tab there are two cases:

Case 1: **SPI**
From the Pinout & Configuration tab:

- from the ᵛ Pinout → Clear Pinouts menu option reset the state of all pins (only for Nucleo 144);
- from the Connectivity > menu:
  - check that the ETH is disabled (only for Nucleo 144);
  - enable the SPI1 in Full-Duplex Master Mode;
  - if not enabled yet, enable the USART2 in Asynchronous mode for COM port (for Nucleo 64)
  - if not enabled yet, enable the USART1 in Asynchronous mode for COM port(for Nucleo 144)

  Check that the enabled SPI uses the following pins:

| | Arduino PINs |
|---|---|
| SPI_MISO | D12 |
| SPI_MOSI | D11 |
| SPI_SCK | D13 |

In case different pins have been enabled by default, change them using the Alternative pins (if available) or bridge the enabled ones and the ones reported in the table above.
Also, the USART enabled above is used for the logs on a terminal via serial link. The correct USART to be enabled may change for some Nucleo board. Information on the correct USART to use are reported in the board User Manual available on st.com.

NOTE: Only for BLE_FOTA application for STM32U575ZI MCU: from the Computing > menu enable the CRC checking the Activated checkbox (this module is required by the FOTA

feature - available only for the STM32U575ZI MCU -).

From the Pinout view set:

| Nucleo 64 | | | Nucleo 144 | | |
|---|---|---|---|---|---|
| PIN | Mode | Label | PIN | Mode | Label |
| PA0 | GPIO_EXTI0 | | PA3 | GPIO_EXTI3 | |
| PB6 | GPIO_Output | | PD14 | GPIO_Output | |
| PA8 | GPIO_Output | | PF13, PG12 | GPIO_Output | |
| PA2 | USART2_TX | USART_TX | PA9 | USART1_TX | USART_TX |
| PA3 | USART2_RX | USART_RX | PA10 | USART1_RX | USART_RX |
| PA5* | GPIO_Output | LD2 [Green Led] | PB7 | GPIO_Output | LD2[Blue] |
| PC13 | GPIO_EXTI13 | B1 [Blue PushButton] | PC13 | GPIO_EXTI13 | USER_Btn[B1] |

Pins in the green rows are used only by the SensorDemo_BLESensor-App and SampleApp applications

* NOTE: In the User Manual UM1724 “STM32 Nucleo-64 boards (MB1136)” it is reported that *the green LED is a user LED connected to Arduino signal D13 corresponding to STM32 I/O PA5 (pin 21) or PB13 (pin 34) depending on the STM32 target*. That means the for some Nucleo-64 board (for instance the Nucleo-F302R8) the LD2[Green Led] may be connected to the PB13 pin instead of to the PA5 pin.

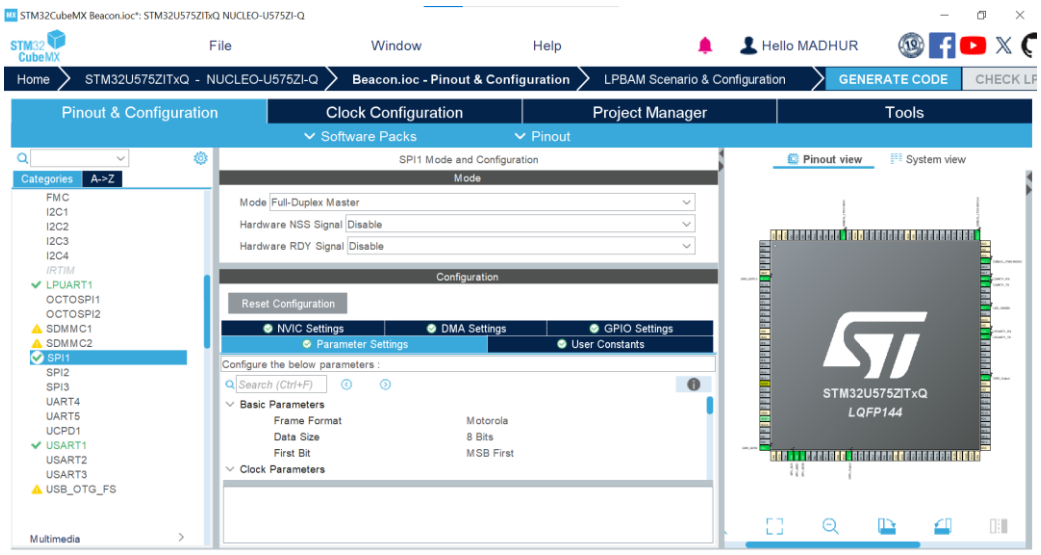


***Figure 23*** *Pinout view*

From the Additional Software > menu, click on the STMicroelectronics.X-CUBE-WB05N.x.y.z pack. Check the Wireless STM32WB05xN box and set the following Platform Settings:

| Name | Supported IPs | Found solutions | | BSP Api |
|---|---|---|---|---|
| | | Nucleo 64 | Nucleo 144 | |
| BUS IO driver | SPI:Full-Duplex Master | SPI1 | SPI1 | BSP_BUS_DRIVER |
| Exti Line | GPIO:EXTI | PA0 | PA3 | HAL_EXTI_DRIVER |

| Name | Supported IPs | Found solutions | | BSP Api |
|---|---|---|---|---|
| | | Nucleo 64 | Nucleo 144 | |
| CS Line | GPIO:Output | PB6 | PD14 | Unknown |
| Reset Line | GPIO:Output | PA8 | PF13, PG12 | Unknown |
| BSP LED | GPIO:Output | PA5 | PB7 | BSP_COMMON_DRIVER |
| BSP USART | USART:Asynchronous | USART2 | USART1 | BSP_COMMON_DRIVER |
| BSP BUTTON | GPIO:EXTI | PC13 | PC13 | BSP_COMMON_DRIVER |

Pins in the green rows are used only by the SensorDemo_BLESensor-App and SampleApp sample applications.

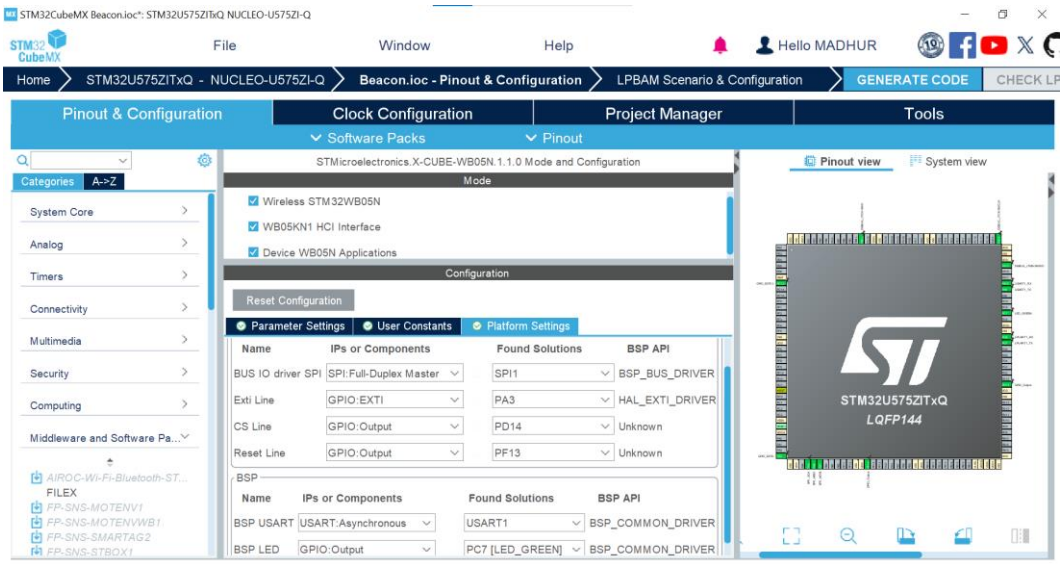


***Figure 24*** *STMicroelectronics.X-CUBE-WB05N Mode and Configuration view*

Case 2: **UART**
From the Pinout & Configuration tab:

- from the ᵛ Pinout → Clear Pinouts menu option reset the state of all pins (only for Nucleo 144);
- from the Connectivity > menu:
  - check that the ETH is disabled (only for Nucleo 144);
  - enable the LPUART1 in Asynchronous Mode (only for Nucleo 144);

  Check that the enabled LPUART uses the following pins:

| | Arduino PINs |
|---|---|
| LPUART1_TX | D1 |
| LPUART1_RX | D0 |

  - Enable the USART1 in Asynchronous mode (only for Nucleo 64).

| | Arduino PINs |
|---|---|
| USART1_TX | D8 |
| USART1_RX | D2 |

**NOTE**: In some Nucleo - 64 boards, user needs to use Jumper wires to Connect USART1 Rx/Tx with X-NUCLEO-WB05KN1(MB2160). Refer respective boards schematic to confirm the connections.

- If not enabled yet, enable the USART2 in Asynchronous mode for COM Port (for Nucleo 64)
- If not enabled yet, enable the USART1 in Asynchronous mode for COM Port (for Nucleo 144)

In case different pins have been enabled by default, change them using the Alternative pins (if available) or bridge the enabled ones and the ones reported in the table above.
Also, the USART enabled above is used for the logs on a terminal via serial link. The correct USART to be enabled may change for some Nucleo board. Information on the correct USART to use are reported in the board User Manual available on st.com.

NOTE: Only for BLE_FOTA for STM32U575ZI MCU: from the Computing > menu enable the CRC checking the Activated checkbox (this module is required by the FOTA feature - available only for the STM32U575ZI MCU ).

From the Pinout view set:

| **Nucleo 64** | | | **Nucleo 144** | | |
|---|---|---|---|---|---|
| PIN | Mode | Label | PIN | Mode | Label |
| PA9 | USART1_TX | | PG7 | LPUART1_TX | |
| PA10 | USART1_RX | | PG8 | LPUART1_RX | |
| PA8 | GPIO_Output | | PF13, PG12 | GPIO_Output | |
| PA2 | USART2_TX | USART_TX | PA9 | USART1_TX | USART_TX |
| PA3 | USART2_RX | USART_RX | PA10 | USART1_RX | USART_RX |
| PA5* | GPIO_Output | LD2 [Green Led] | PB7 | GPIO_Output | LD2[Blue] |
| PC13 | GPIO_EXTI13 | B1 [Blue PushButton] | PC13 | GPIO_EXTI13 | USER_Btn[B1] |

Pins in the green rows are used only by the SensorDemo_BLESensor-App and SampleApp applications

* NOTE: In the User Manual UM1724 “STM32 Nucleo-64 boards (MB1136)” it is reported that *the green LED is a user LED connected to Arduino signal D13 corresponding to STM32 I/O PA5 (pin 21) or PB13 (pin 34) depending on the STM32 target*. That means the for some Nucleo-64 board (for instance the Nucleo-F302R8) the LD2[Green Led] may be connected to the PB13 pin instead of to the PA5 pin.

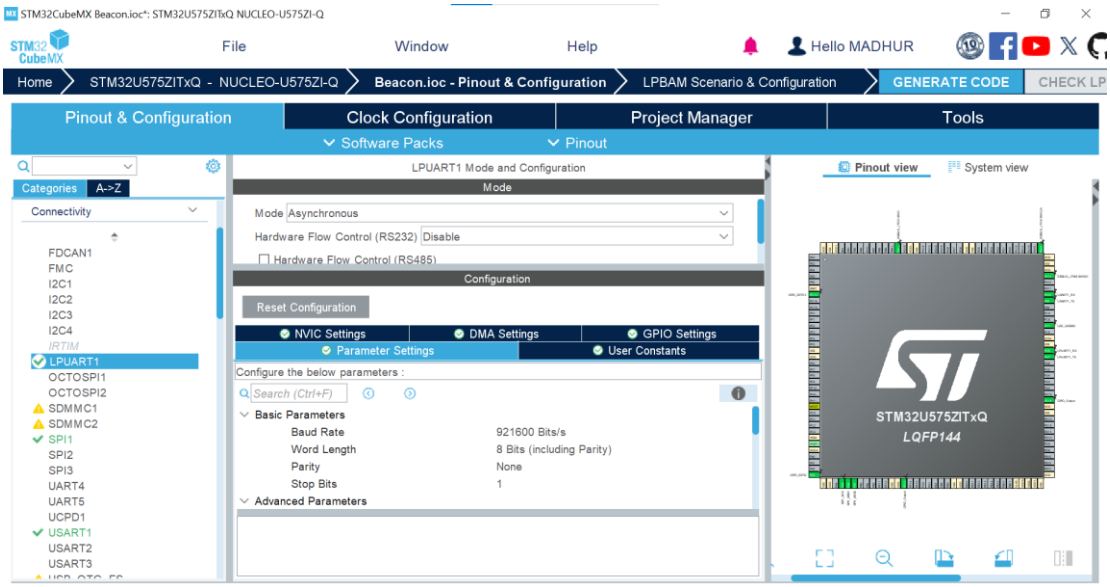


***Figure 25** Pinout view*

From the Additional Software > menu, click on the STMicroelectronics.X-CUBE-WB05N.x.y.z pack. Check the Wireless STM32WB05xN box and set the following Platform Settings:

| Name | Supported IPs | Found solutions | | BSP Api |
|---|---|---|---|---|
| | | Nucleo 64 | Nucleo 144 | |
| BUS IO driver | LPUART:Asynchronous, USART:Asynchronous | USART1 | LPUART1 | BSP_BUS_DRIVER |
| Reset Line | GPIO:Output | PA8 | PF13, PG12 | Unknown |
| BSP LED | GPIO:Output | PA5 | PB7 | BSP_COMMON_DRIVER |
| BSP BUTTON | GPIO:EXTI | PC13 | PC13 | BSP_COMMON_DRIVER |
| BSP USART | USART:Asynchronous | USART2 | USART1 | BSP_COMMON_DRIVER |

Pins in the green rows are used only by the SensorDemo_BLESensor-App and SampleApp applications.

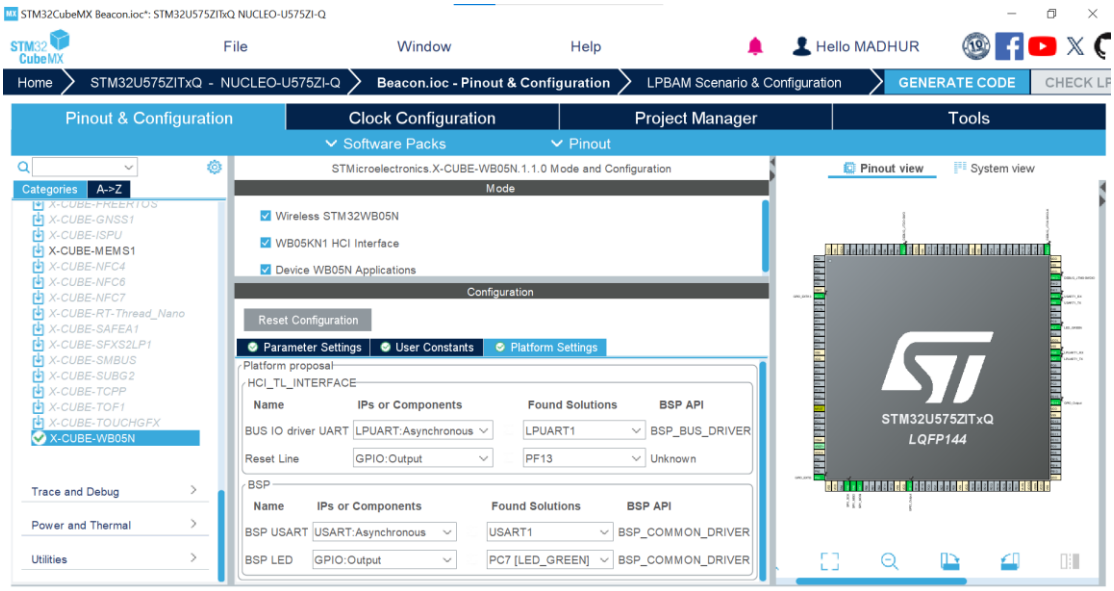


***Figure 26*** *STMicroelectronics.X-CUBE-WB05N Mode and Configuration view*

From the Parameter Settings tab, some parameters for the data logging, the debugging and for the Bluetooth LE scanning, advertising and connection can be set.
For all the sample applications, the default parameters can be used.
For the Beacon sample application, the Advertising Type and the Minimum and Maximum Advertising Intervals should be set as in the following table (but the application works also with the default parameters):

| **Beacon Sample Application** | |
|---|---|
| Advertising Type (ADV_DATA_TYPE) | Non Connectable Undirected Advertising (ADV_NONCONN_IND) |
| Minimum Advertising Interval (ADV_INTERV_MIN) | 1600 |
| Maximum Advertising Interval (ADV_INTERV_MAX) | 1600 |

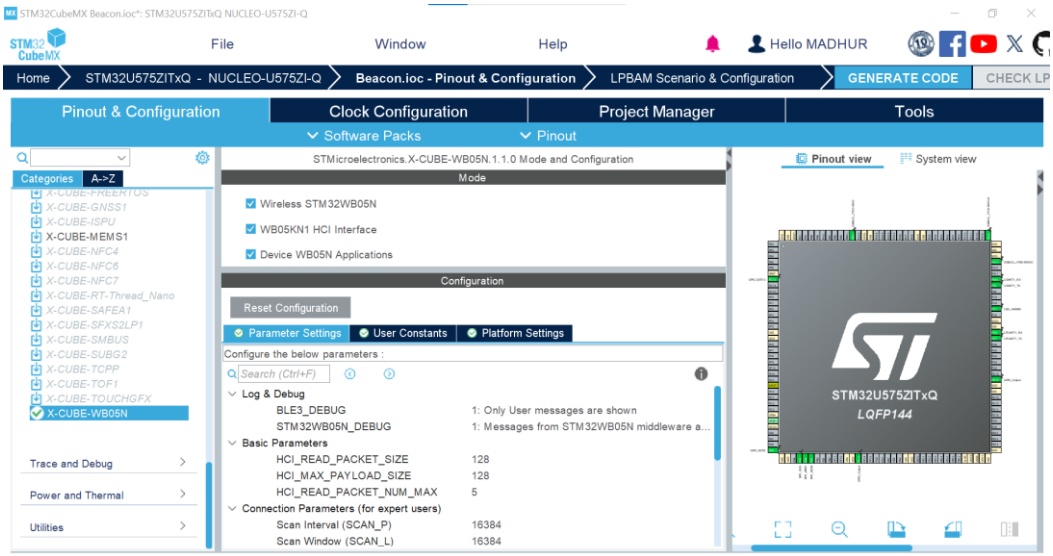


***Figure 27*** *Bluetooth LE Parameter Settings*

From the **Configuration** tab, click on NVIC button under System to enable the EXTI line interrupts for both the SPI IRQ and the User Button (when used):

| Name | Nucleo 64 | Nucleo 144 |
|---|---|---|
| Exti Line | EXTI line 0 interrupt | EXTI line 3 interrupt |
| BSP BUTTON | EXTI line 13 interrupt | EXTI line 13 interrupt |

EXTI line interrupt in the green row must be enabled only for the SensorDemo_BLESensor-App and SampleApp sample applications.

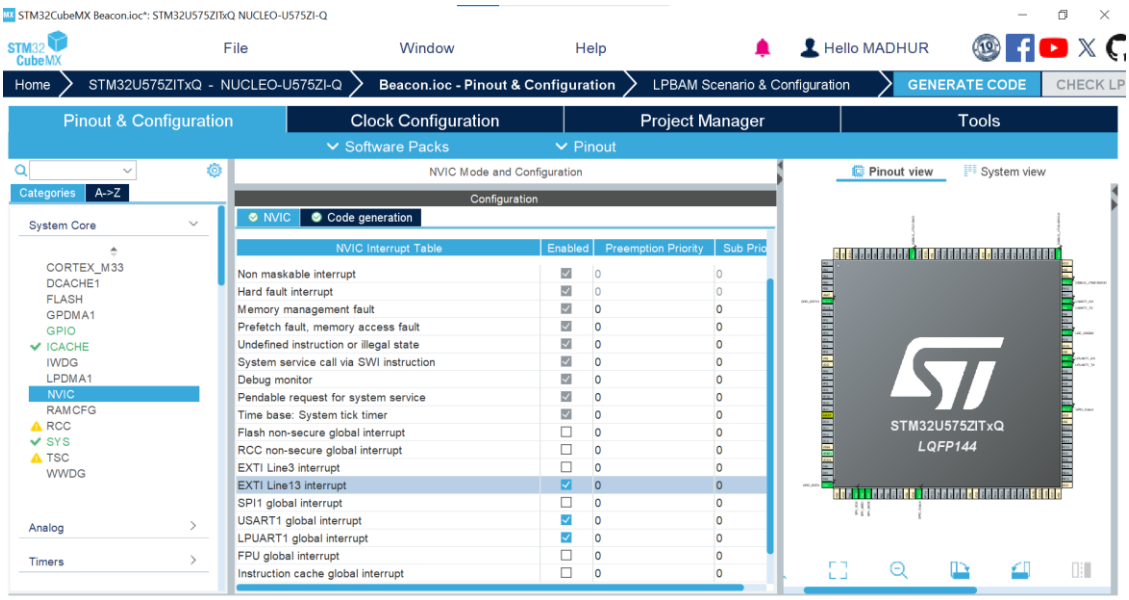


***Figure 28** NVIC Mode and Configuration view*

From the System view, click on SPIx button under Connectivity category and:

- check that the Data size is 8 Bits;
- set the Prescaler (for Baud Rate) to a value so that the HClock/Prescaler is less or equal to 10.0 MBits/s (the maximum supported SPI speed);
- set the Clock Phase (CPHA) to 2 Edge
- set the Clock Polarity (CPOL) to High

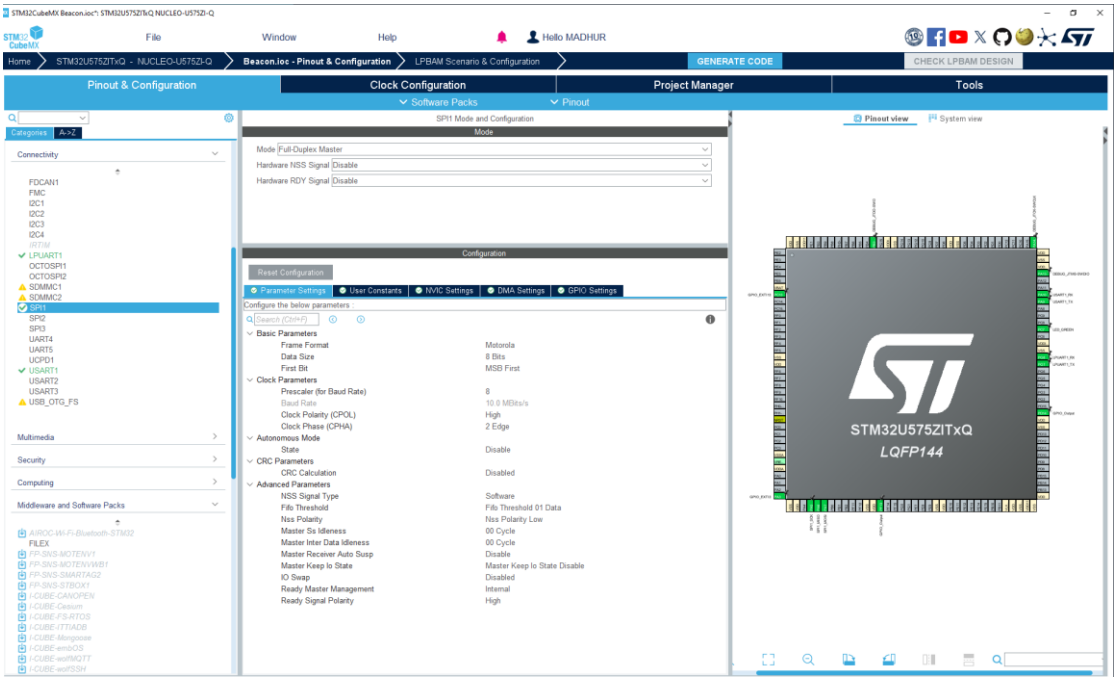


***Figure 29** STM32CubeMX SPI Configuration*

From the System view, click on LPUART1 button under Connectivity category and:

- check that the Word Length is 8 Bits (including Parity);
- set the Baud rate to 921600 Bits/s
- set the Parity to None
- Set the Stop Bits 1

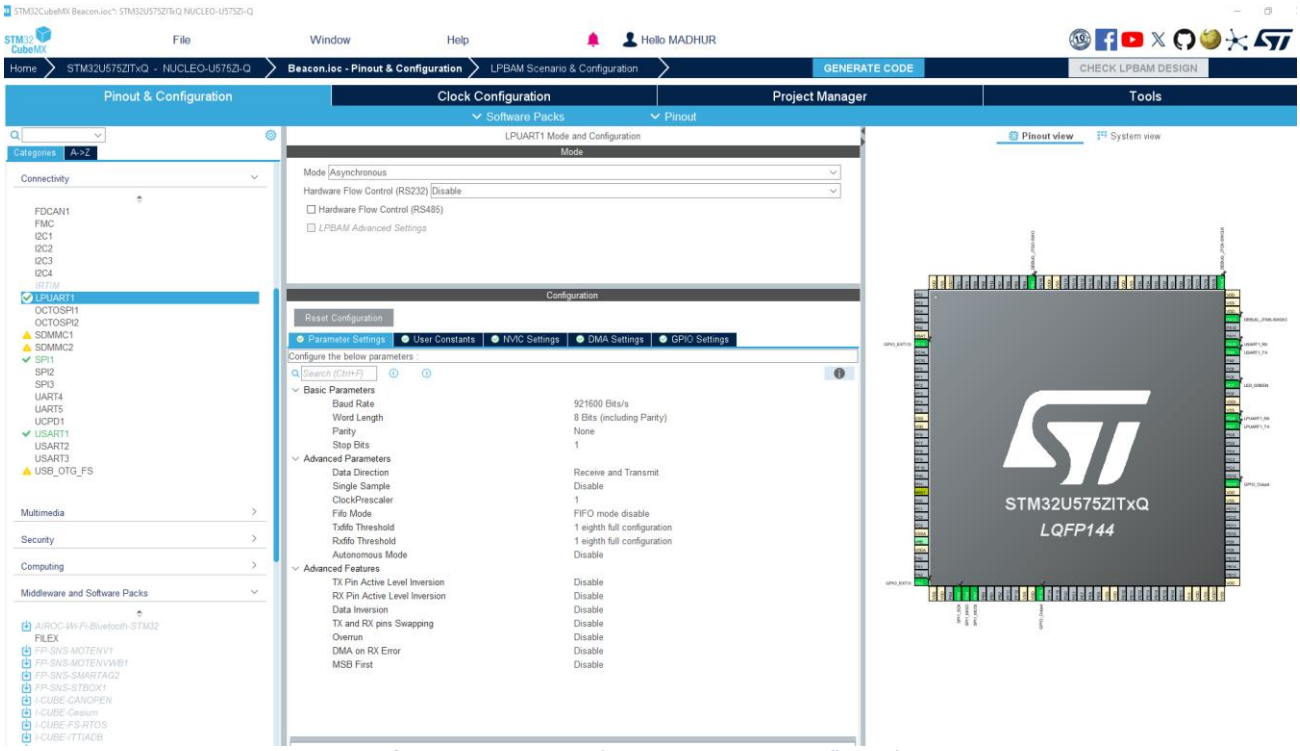


***Figure 30** STM32CubeMX LPUART1 Configuration*

From the System view, click on USARTx button under Connectivity category and check that the following configuration is set:

| | |
|---|---|
| Baud Rate | 921600 Bits/s |
| Word Length | 8 Bits (including Parity) |
| Parity | None |
| Stop Bits | 1 |

Also, from the GPIO Settings tab, be sure the USART_TX and USART_RX labels are set.

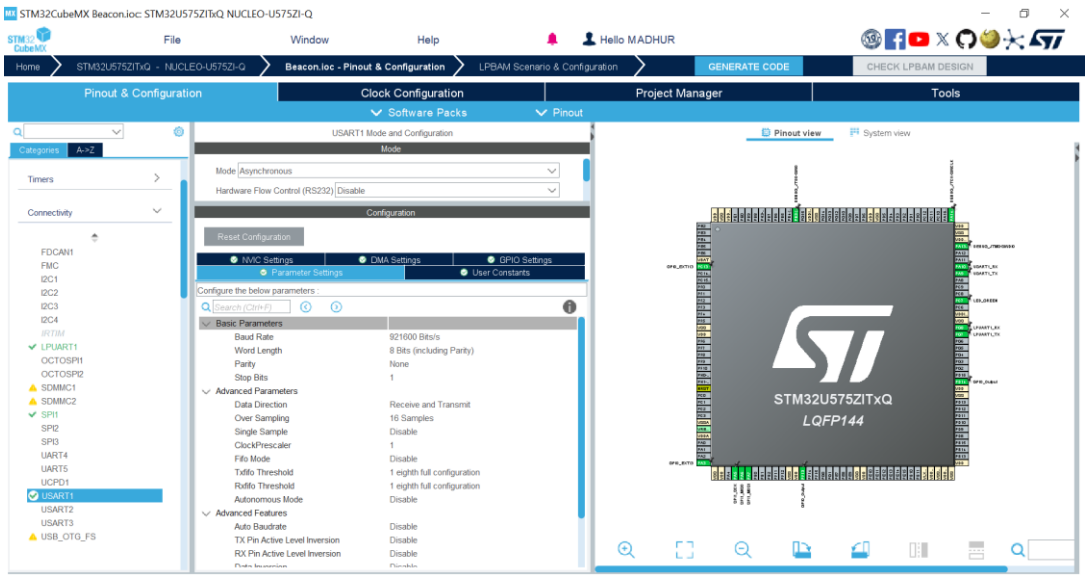


***Figure 31*** *USART Configuration*

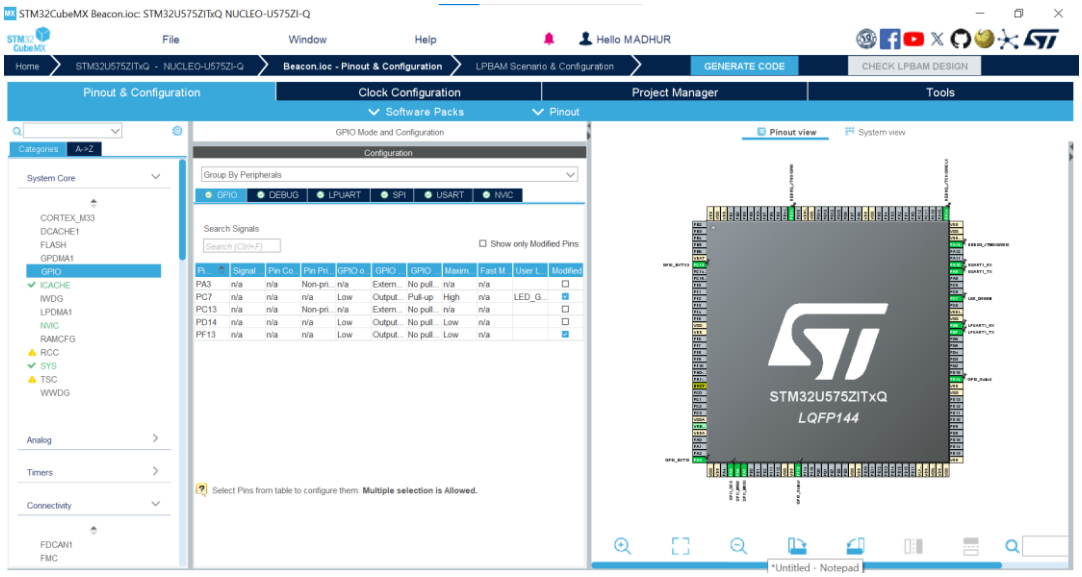


***Figure 32*** *GPIO Configuration*

Once all the above-described steps have been performed, from the Project Manager tab the Project Name, the Project Location, the Toolchain/IDE, the Firmware Package Name and Version and so on can be set.

Hence, after checking that in the Advanced Settings tab the following options are set

| Generate Code | Rank | Function Name | Peripheral Instance Name | Do Not Generate Function Call | Visibility (Static) |
|---|---|---|---|---|---|
| ☑ | 1 | SystemClock_Config | RCC | ☐ | ☐ |
| ☑ | 2 | MX_GPIO_Init | GPIO | ☐ | ☑ |
| ☑ | 3 | MX_ICACHE_Init | ICACHE | ☐ | ☑ |
| ☑ | 4 | MX_CRC_Init | CRC | ☐ | ☑ |
| ☑ | 5 | MX_SPI1_Init | SPI1 | ☐ | ☑ |
| ☑ | 6 | MX_STM32WB05N_Init | STMicroelectronics.X-CUBE-WB05… | ☐ | ☐ |
| ☑ | 7 | MX_STM32WB05N_Process | STMicroelectronics.X-CUBE-WB05… | ☐ | ☐ |

***Figure 33*** *Advanced Settings*

and that for the IFRStack_Updater and BLE_FOTA sample application the following linker settings are used:

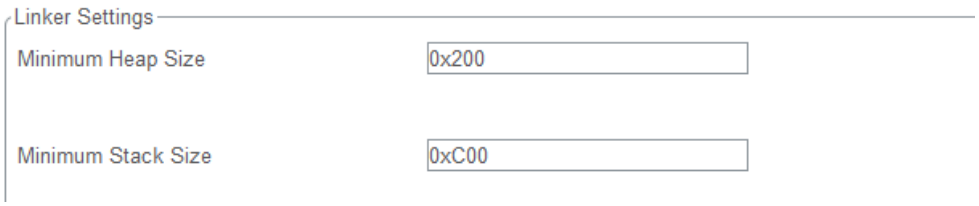


***Figure 34*** *IFRStack_Updater and BLE_FOTA Linker Settings*

the source code of the project using the **STMicroelectronics X-CUBE-WB05N** software can be generated clicking the GENERATE CODE button.

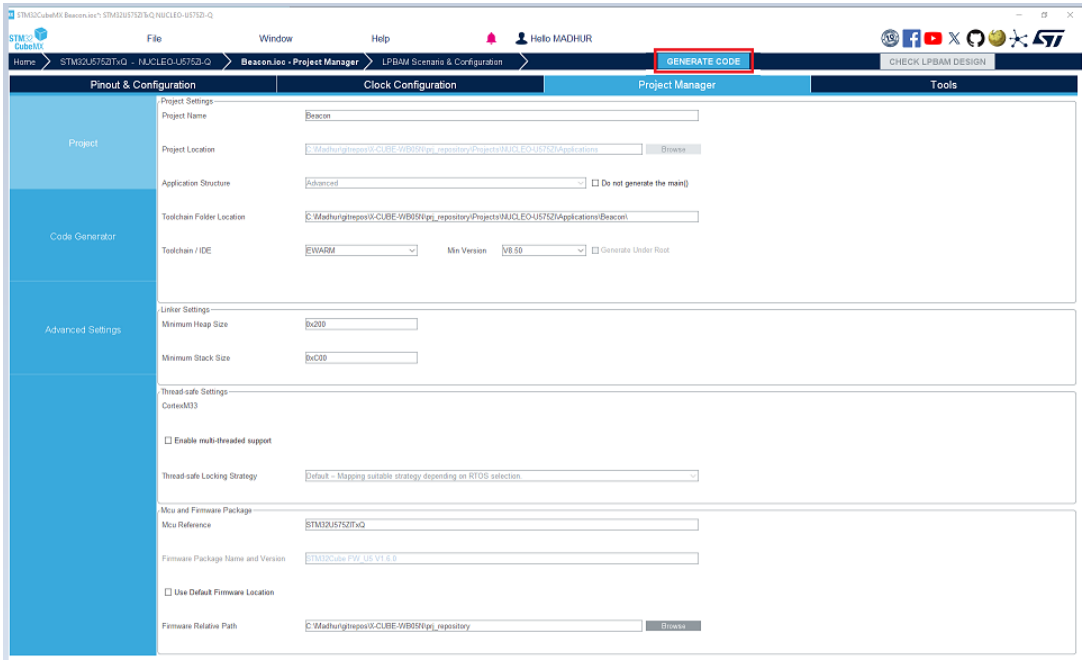


***Figure 35*** *Project Manager view*

# 10 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 middleware 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 middlewares are used.

In the Advanced mode `Src` and `Inc` are generated under folder *Core*.
For each middleware, the list of the generated files is under `<MW_Name>` (`STM32WB05xN` for the X-CUBE-WB05N pack), at the same level as `Core` and either in `App` or in `Target` subfolder.

# 11 Known Limitations and workarounds

- For sample applications using any low power feature, such as **Beacon**, the ST-Link reset must be set in *Connect during reset* mode into the generated project configuration options.
- In Beacon sample application, Beacon is not visible in “BLE Scanner” app in IOS. Check with any other Beacon scanning app available or on Android platform.
- The Virtual_COM_Port and BLE_FOTA sample application must be used with the following configuration for the HCI Transport Layer (HCI_TL) and the HCI Transport Layer Interface (HCI_TL_INTERFACE):
  - HCI_TL → Basic
  - HCI_TL_INTERFACE → UserBoard

  Other configurations using the template files are not supported yet.
- BLE_FOTA application is functional only with Dual Bank Flash device such as STM32U5x.
- IFRStack_Updater application is only supported for NUCLEO-U575ZI-Q board.
- No support to **Low Level (LL) Driver** is provided yet for the SPI interface used by the STM32WB05xN chip.
- In MDK-ARM projects (e.g., for STM32H7 and STM32L4 series), if no log message is printed on the serial terminal, enable the Use MicroLib option in the project settings.
- While using UART as HCI communication between NUCLEO 64 boards and X-NUCLEO-WB05KN1 user needs to use Jumper wires to Connect USART1 Rx/Tx with X-NUCLEO-WB05KN1(MB2160). Refer respective boards schematic to confirm the connections.

# 12 References

[1] AN4979 – Application Notes – *Bluetooth Low Energy beacons with Eddystone*
[2] UM1724 – User Manual – *STM32 Nucleo-64 boards (MB1136)*

# 13 Revision history

**Table 2: Document revision history**

| Date | Version | Changes |
|---|---|---|
| 12-Sep-2024 | 1 | Initial release |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

## 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. 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.