P2P server application
BLE profile concept & build

life.augmented

Let’s start enabling Bluetooth Low Energy applications with STM32WBx

Create BLE Profile 2

Starting from an .ioc file create a Bluetooth low energy profile and connect to your smartphone.

1 Basic beacon

2 Create BLE Profile

3 PAwR Advanced feature

4 BLE Add On

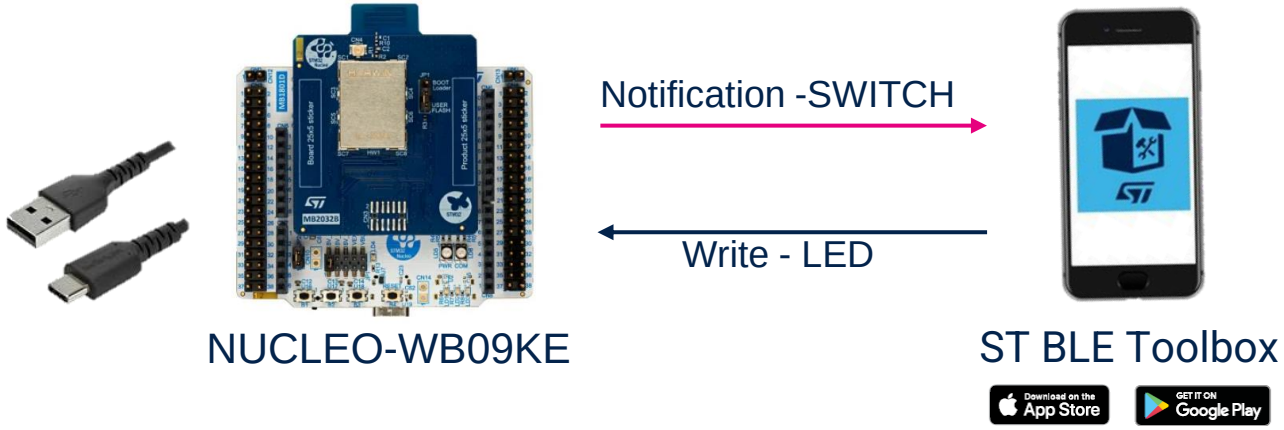
5 Test your RF design

life.augmented

Purpose

- In this part we will show following:
  - How to build a Bluetooth® Low Energy application using STM32CubeIDE.
  - Create a custom Bluetooth Low Energy Service : “P2PServer”
  - Establish connection with smartphone
    - Control Nucleo-WB09 LEDs from Smartphone
    - Received Notification on Smartphone when pushing button on Nucleo-WB09

1.16.0



STM32CubeIDE capabilities

1	**Example application** complete application running over NUCLEO
2	**Board level** all the hardware is already configured
3	**Chipset level** require to configure your HW (PCB) & your application

Create and configure your BLE application STM32CubeIDE “step by step”

PCB HW initialisation using STM32cubeIDE : Pinout, xTals,..

STM32WB0 IPs & peripherals configuration

Clock Tree configuration

BLE configuration : Advertising, Service, Characteristic

Code generation & application code management over CubeIDE

Hands-on Focus

STM32CubeMx design from chipset level Hands-on focus (1/2)

3 **Chipset level**
require to configure your HW (PCB) & your application

To ease Hands-on session use BLE_WS_WB0_HandsOn.ioc
Needed HW IPs & required peripheral to use RF are already initialized : NVIC, RNG, RCC,...
Thanks to BLE_WS_WB0_HandsOn.ioc let's focus on BLE application design

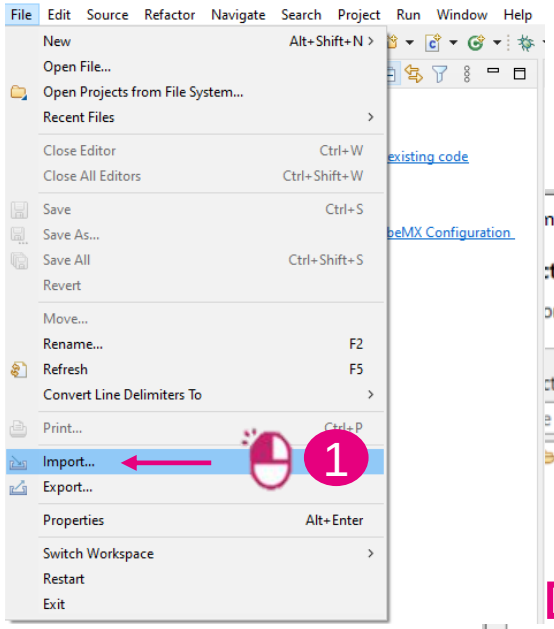
Copy Hands-on_WS_WB0.ioc on your favorite local repository :
example : C:\users\...\STM32WB0_WS\project

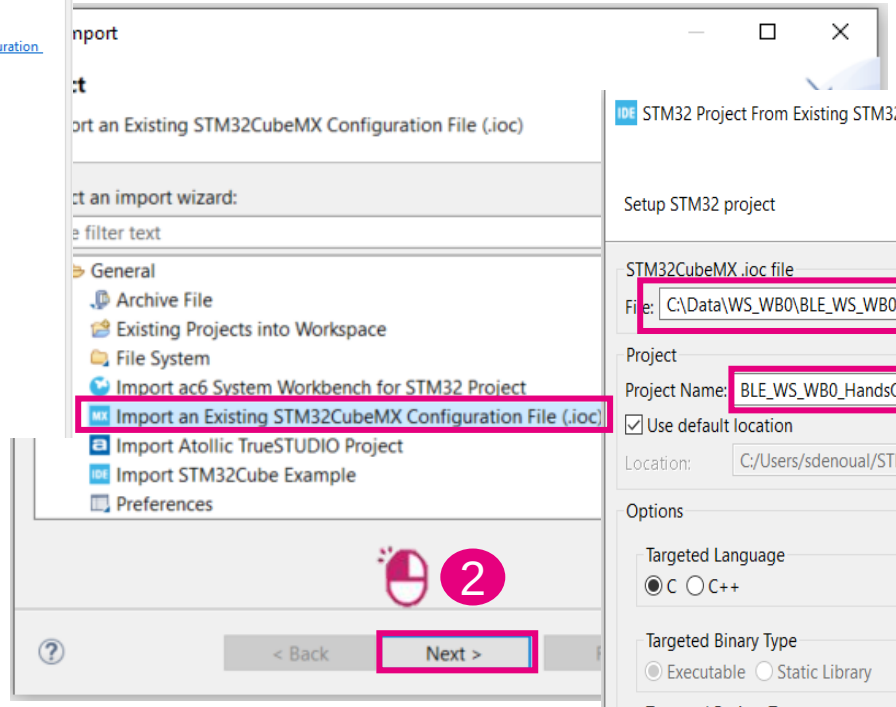
BLE_WS_WB0_HandsOn.ioc can be downloaded here from this github link

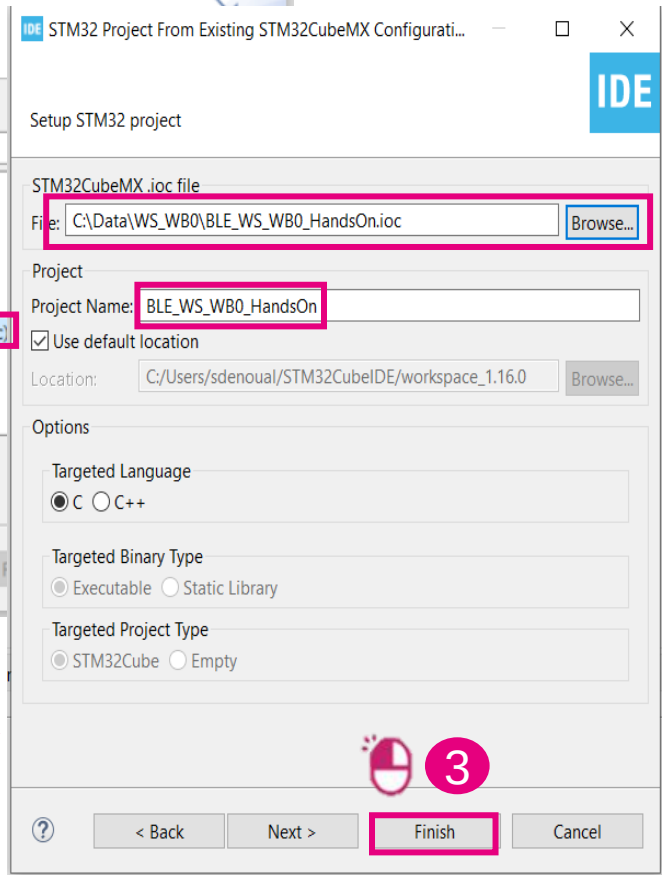
This .ioc was built based on **AN5977 :** "How to build a Bluetooth® Low Energy application with STM32WB0 MCUs"

life.augmented

Open and Start STM32CubeIDE





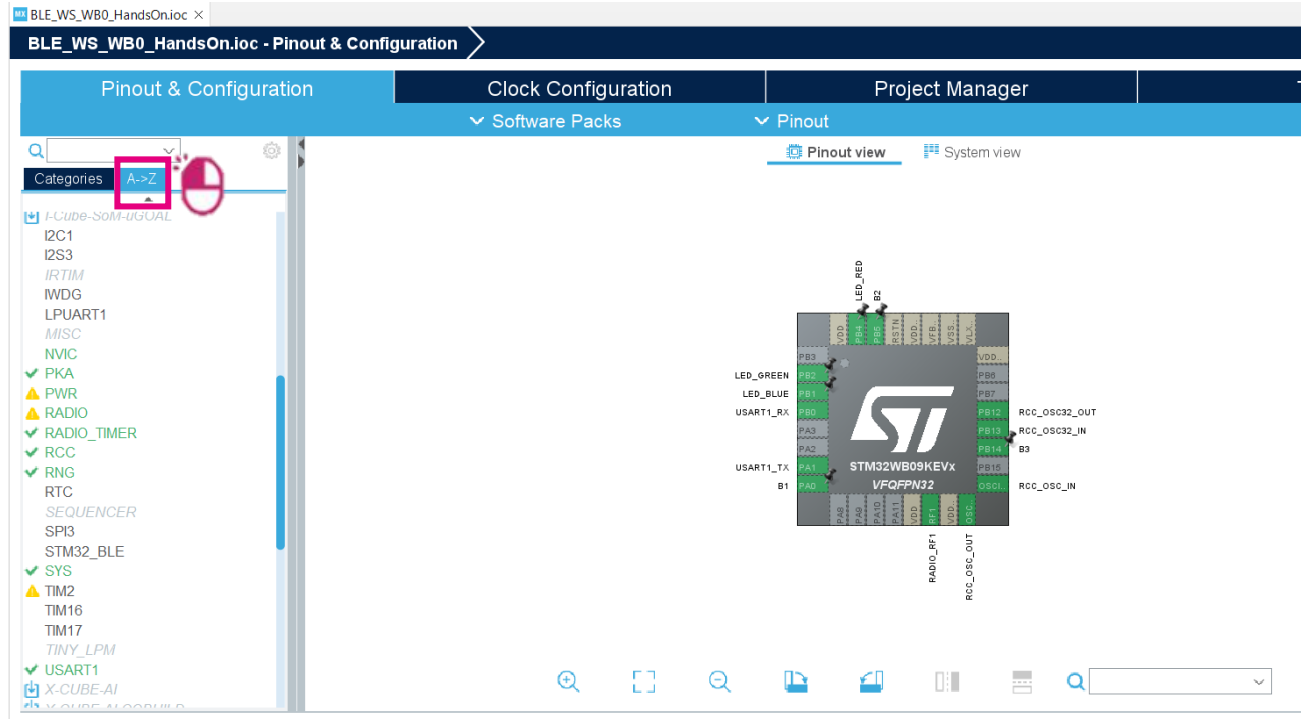


life.augmented

Peripherals in place to start BLE configuration !

BLE_WS_WB0_HandsOn.ioc

All peripherals in place to start BLE configuration
More details in AN5977 : “How to build a Bluetooth® Low Energy application with STM32WB0 MCUs”

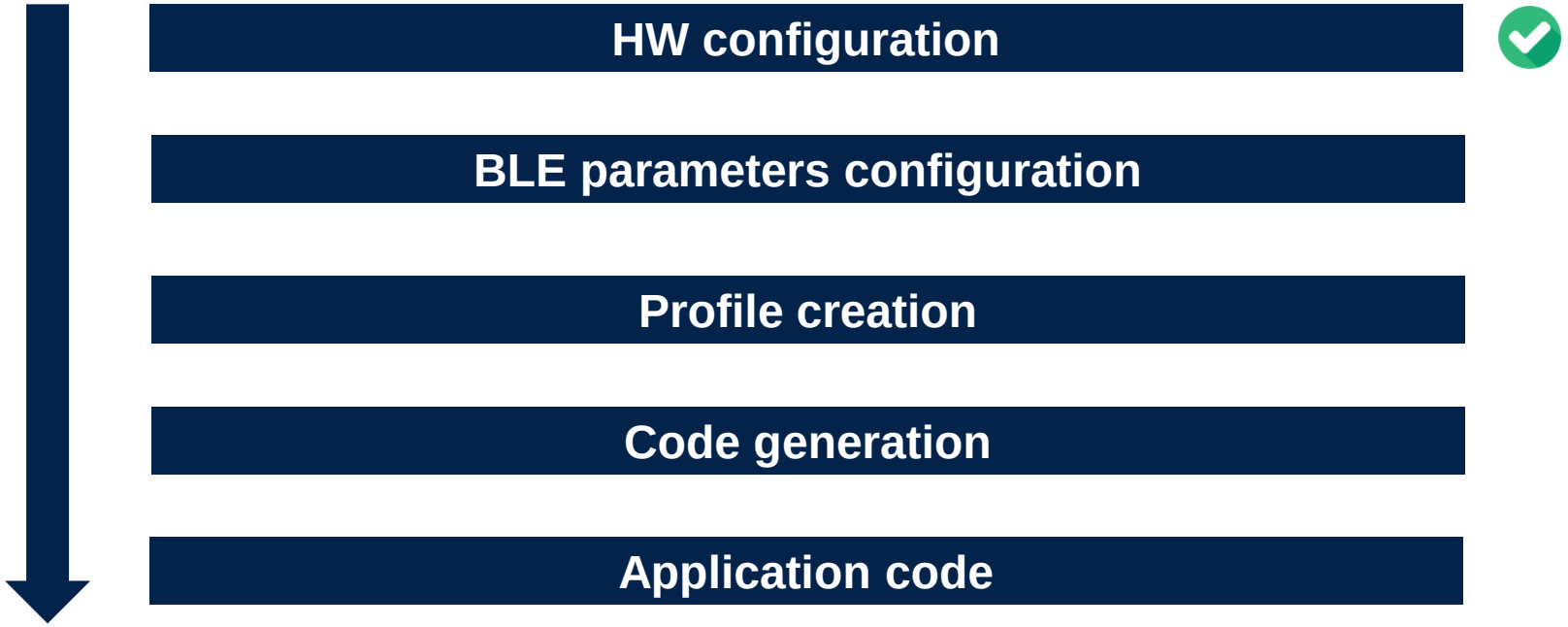


BLE_WS_WB0_HandsOn.ioc

- HW configuration (LED/Button) and needed peripherals for BLE activities
  - enable STM32_BLE (**BLE middleware activation**)
  - We can focus on Bluetooth Low Energy application!

life.augmented

Configuration completed
What's next ? BLE config and Profile creation



Implementing proprietary profile P2P_Server

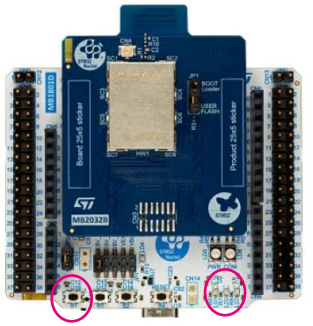
Notification -SWITCH

Write - LED

ST BLE Toolbox

Custom SERVICE

Service Long Name	P2P_Server	
Service Short Name	P2P_Server	
UUID Type	128 bits	
UUID	8F E5 B3 D5 2E 7F 4A 98 2A 48 7A CC 40 FE 00 00	
Characteristic Long Name	My_LED_Char	My_Switch_Char
Characteristic Short Name	LED_C	SWITCH_C
UUID Type	128 bits	128 bits
UUID	19 ED 82 AE ED 21 4C 9D 41 45 22 8E 41 FE 00 00	19 ED 82 AE ED 21 4C 9D 41 45 22 8E 42 FE 00 00
Char Properties	Read + Write w/o response	Notify
Char Permissions	None	None
Char GATT Events	GATT_NOTIFY_ATTRIBUTE_WRITE	GATT_NOTIFY_ATTRIBUTE_WRITE



Button 1 LEDs

NUCLEO-WB09KE

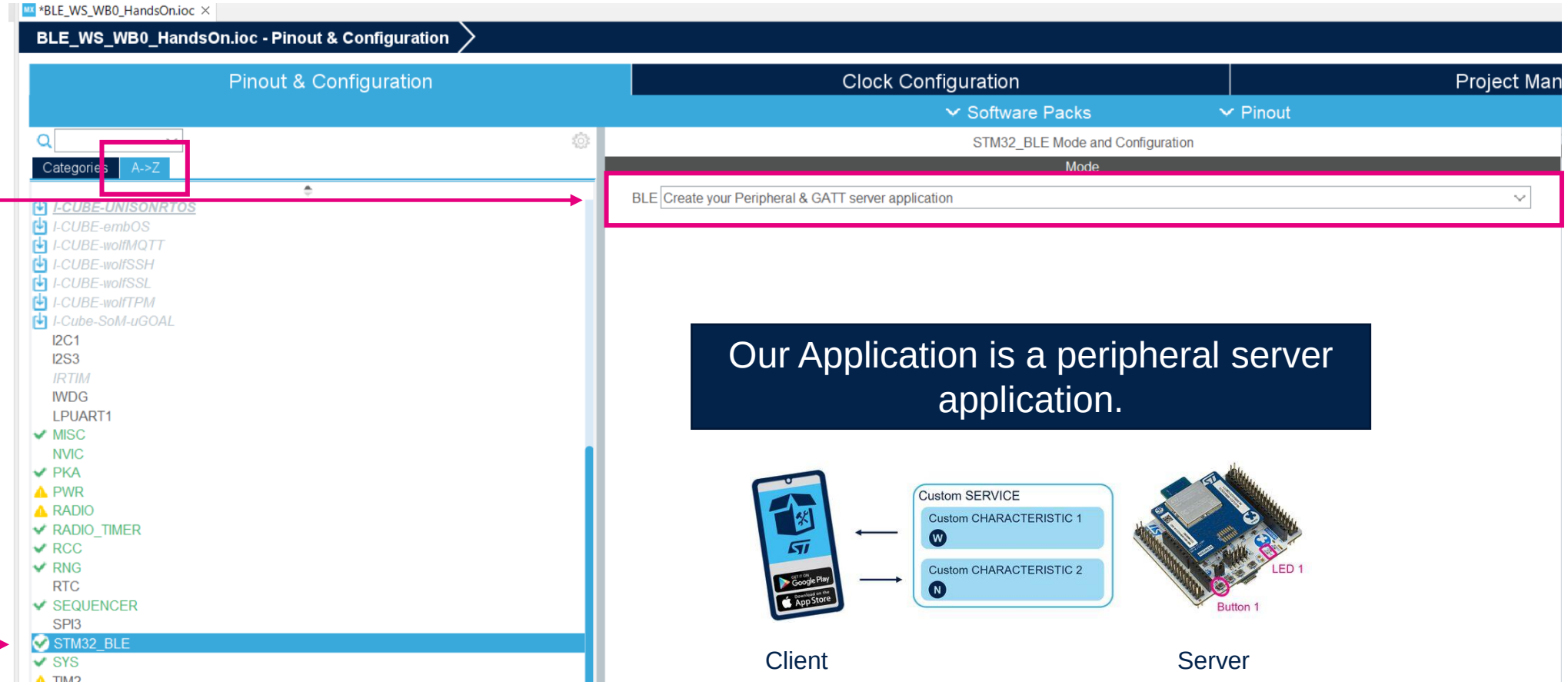
All services/characteristics are identified by a UUID (**U**niversally **U**nique **Id**entifier)

Characteristic can take 3 types of properties: **R**EAD, **W**RITE, **N**OTIFY

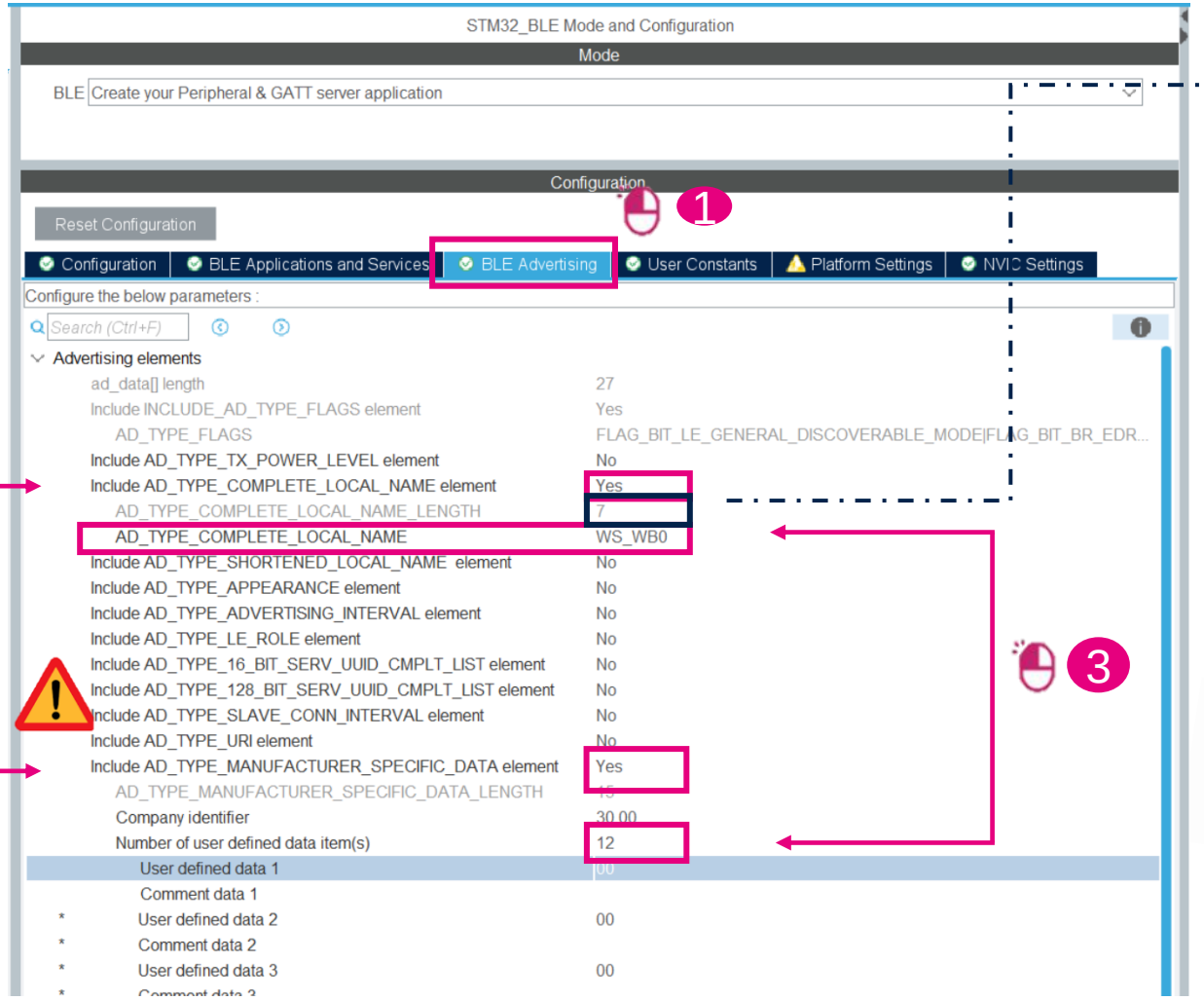
Profile can be defined by **Bluetooth® SIG** → **UUID : 16 bits**

Profile can be a **custom** (proprietary) profile → **UUID : 128 bits**

Enabling Bluetooth® Low Energy



Advertising Elements configuration



Local Name length = Local name + 1

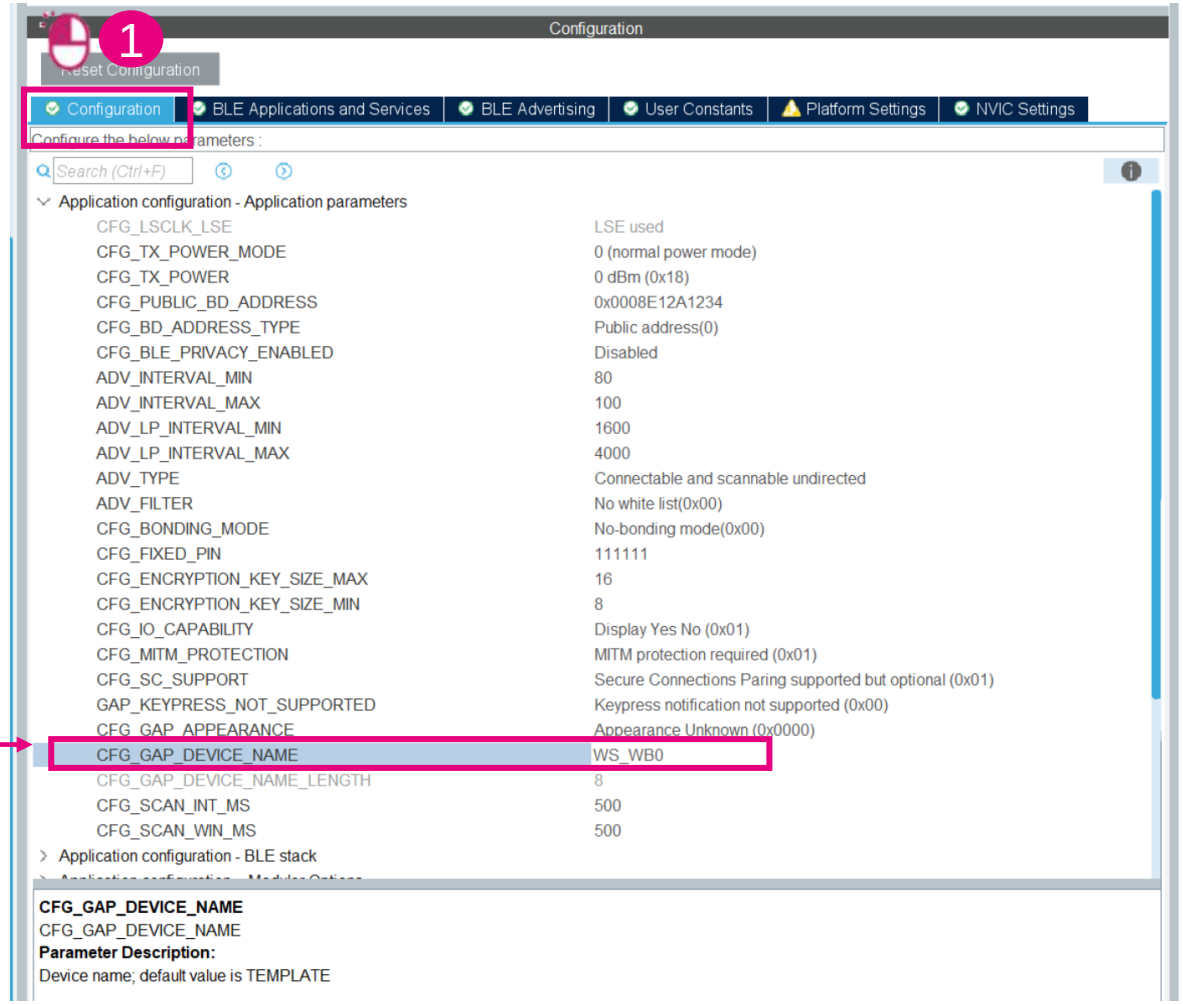
As a server, our application will have to advertise waiting for connection request from a client.

Define here your “custom” local name part of advertising frame.

Manufacturer Ad Type , with company ID 0x30 (STMicroelectronics)

Allow to detect device as an ST device and to connect as P2P profile

Customize Device Name



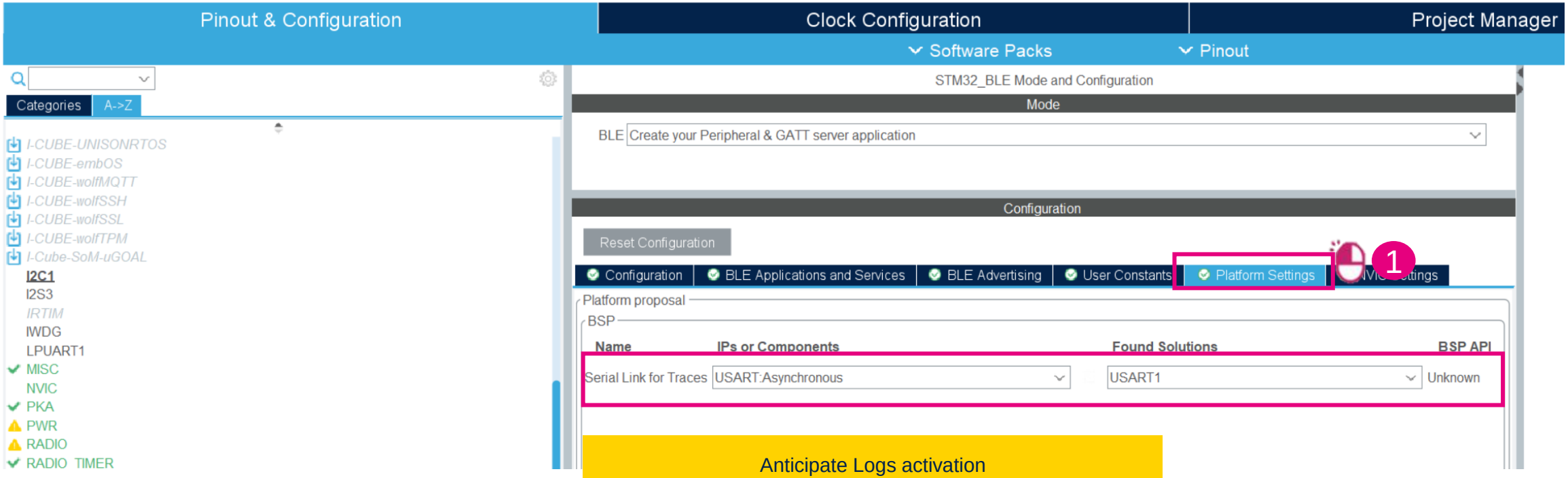
2

set same Device name = Local Name

iOS displays Local Name (advertising data) prior to a 1st connexion.
After a 1st connexion iOS displays Device name (thanks to look up table : associates BLE MAC @ & Device Name)

life.augmented

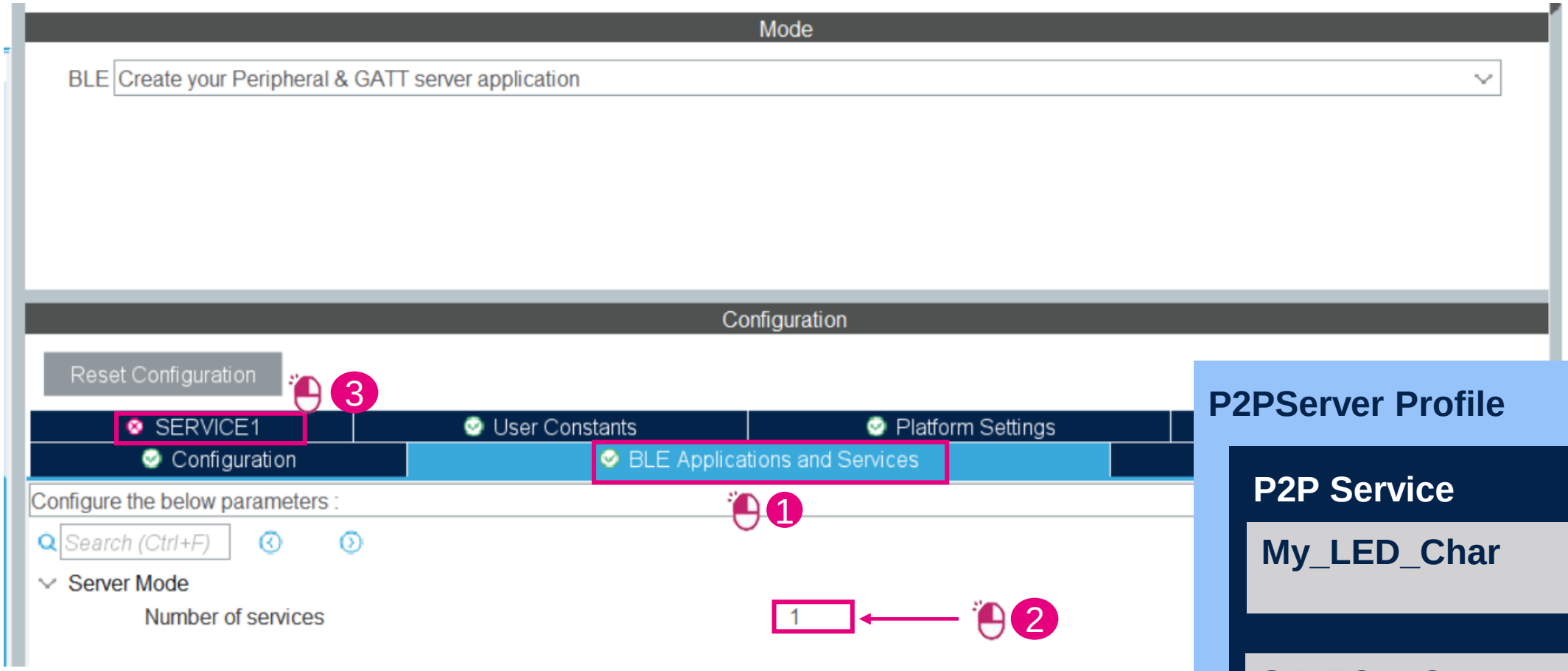
Platform Settings
Trace & Logs: BSP settings



Anticipate Logs activation

Logs activation may require application configuration changes

Profile Creation Service



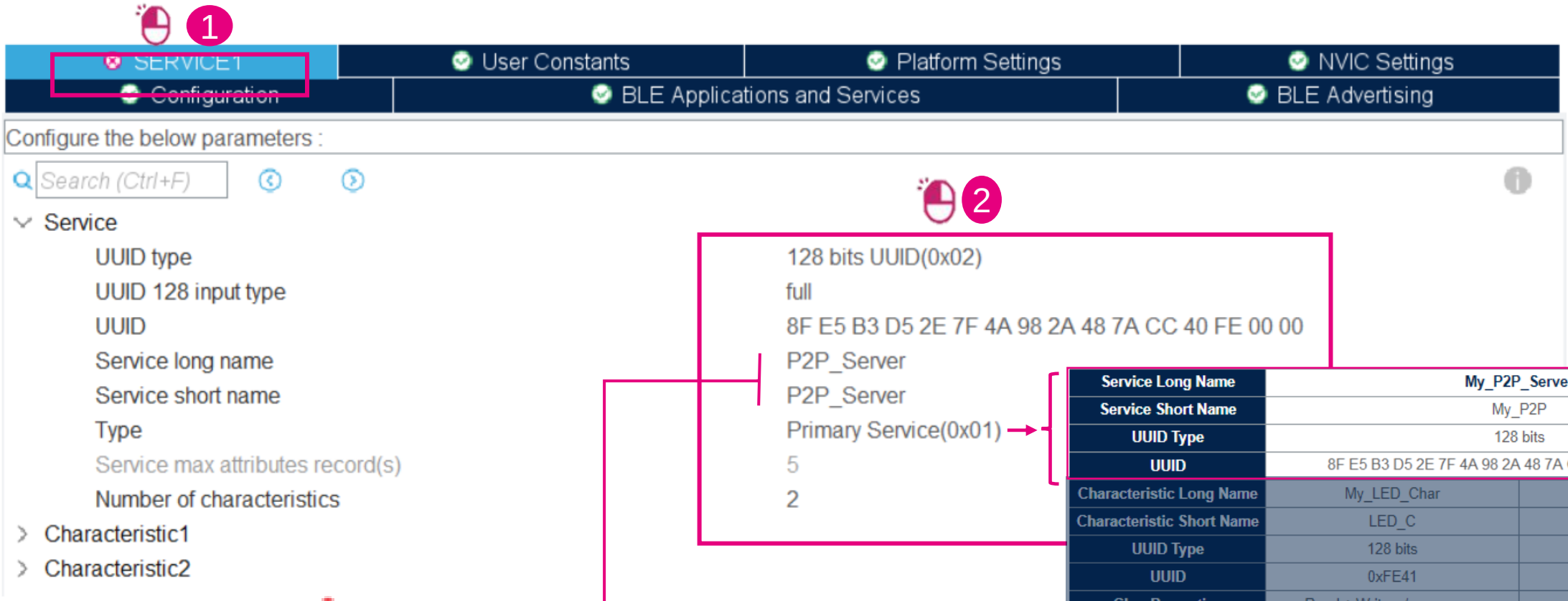
P2PServer Profile

P2P Service

My_LED_Char

SWITCH_C

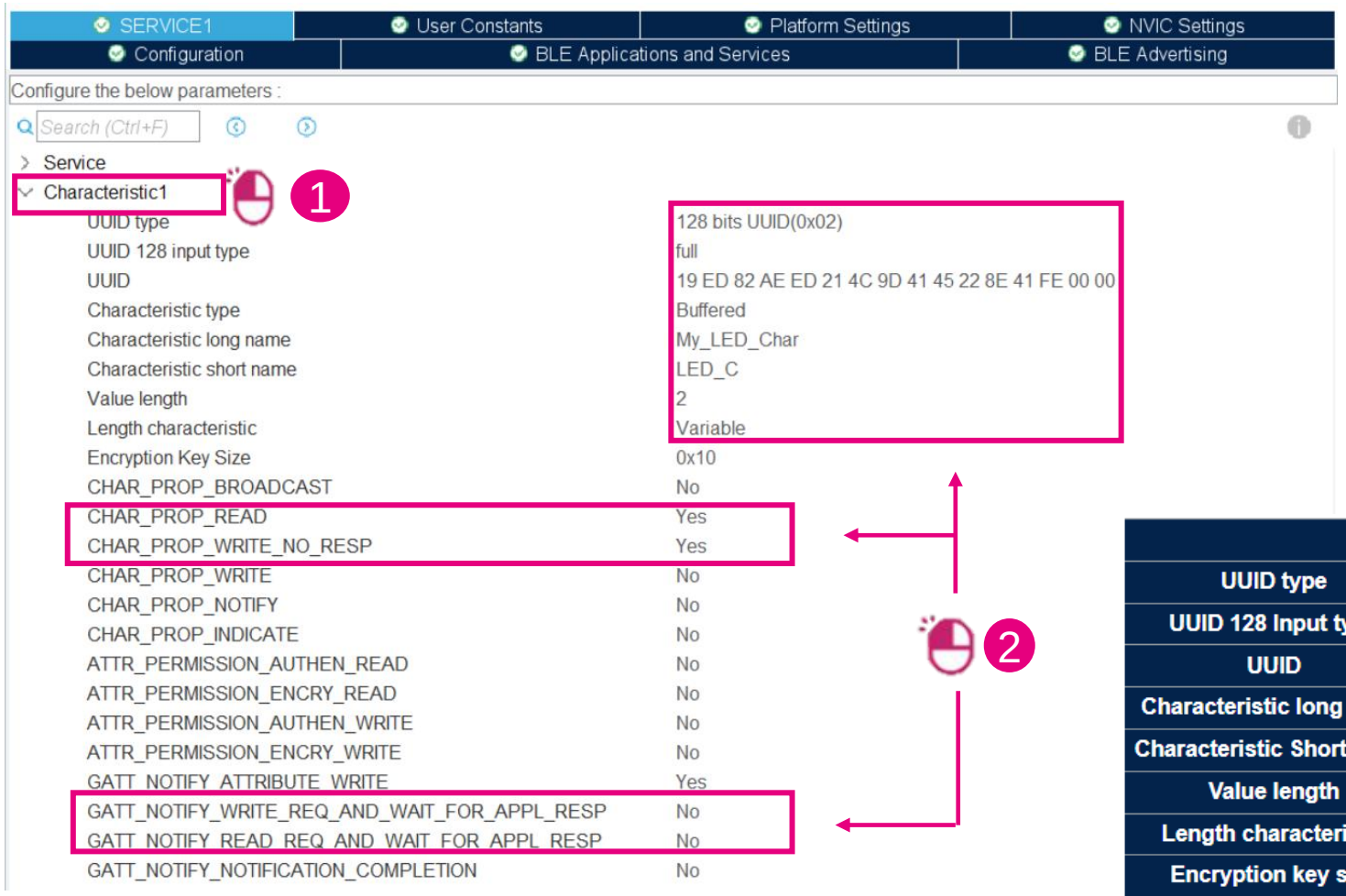
Profile Creation
Configure my P2P Service



Service Long Name	My_P2P_Server	
Service Short Name	My_P2P	
UUID Type	128 bits	
UUID	8F E5 B3 D5 2E 7F 4A 98 2A 48 7A CC 40 FE 00 00	
Characteristic Long Name	My_LED_Char	My_Switch_Char
Characteristic Short Name	LED_C	SWITCH_C
UUID Type	128 bits	128 bits
UUID	0xFE41	0xFE42
Char Properties	Read + Write w/o response	Notify
Char Permissions	None	None
Char GATT Events	GATT_NOTIFY_ATTRIBUTE_WRITE	GATT_NOTIFY_ATTRIBUTE_WRITE

service & characteristic naming used to name function at code generation
Use : "P2P_Server"

Profile Creation
Configure 1st Characteristic



UUID :19 ED 82 AE ED 21 4C 9D 41 45 22 8E 41 FE 00 00

Properties

Data (**2 bytes**) can be read and write.
The purpose of characteristic 1 is to write data in order to control LED

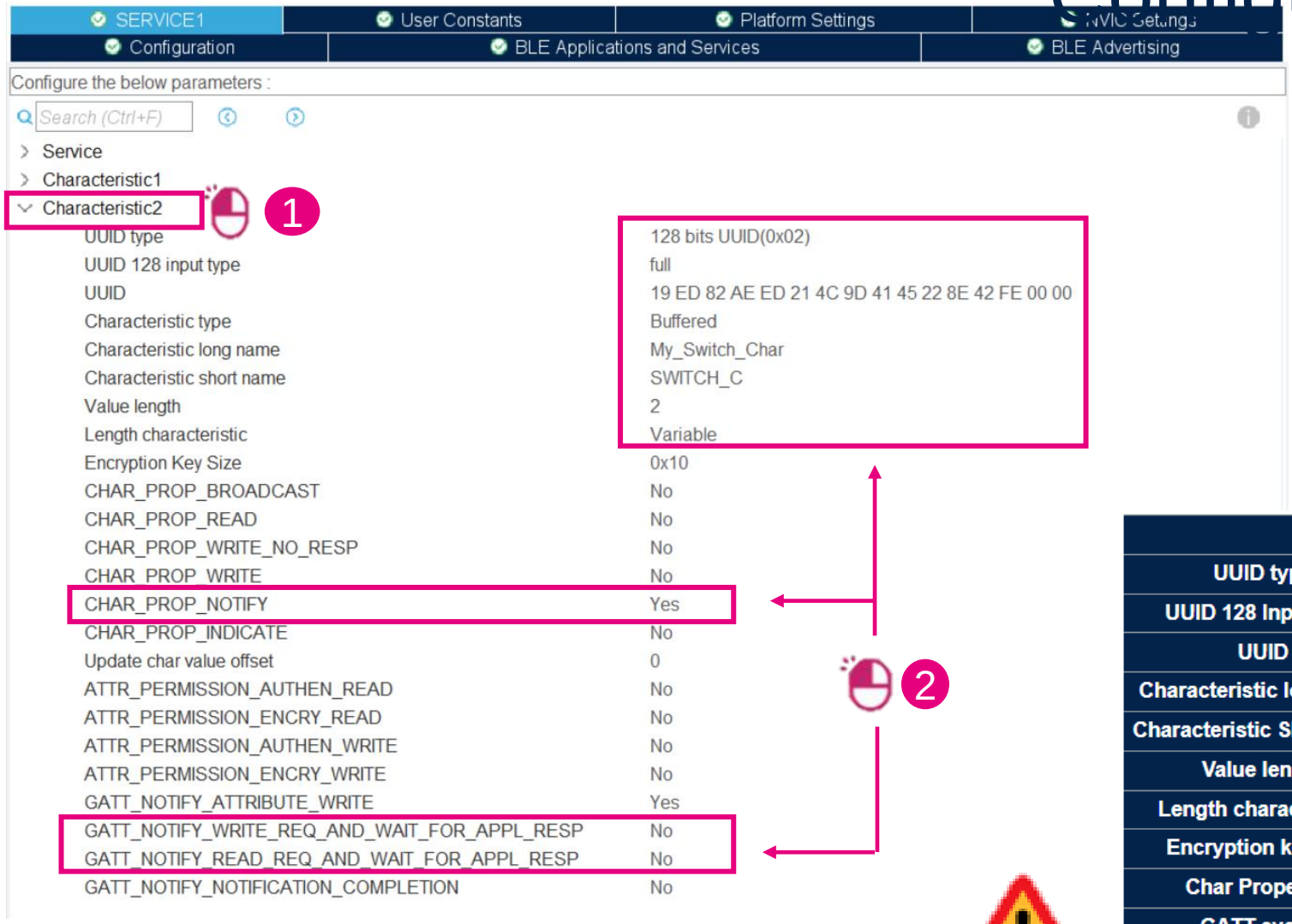
Permission

Thanks to **notify write**, application is informed that attribute has been modified and can accordingly process expected use case

	Characteristic 1	Characteristic 2
UUID type	128 bits UUID (0x02)	128 bits UUID (0x02)
UUID 128 Input type	full	full
UUID	19 ED 82 AE ED 21 4C 9D 41 45 22 8E 41 FE 00 00	19 ED 82 AE ED 21 4C 9D 41 45 22 8E 42 FE 00 00
Characteristic long name	My_LED_Char	My_Switch_Char
Characteristic Short Name	LED_C	SWITCH_C
Value length	2	2
Length characteristic	Variable	Variable
Encryption key size	0x10	0x10
Char Properties	READ | WRITE_WITHOUT_RESP	NOTIFY
GATT events	GATT_NOTIFY_ATTRIBUTE_WRITE	GATT_NOTIFY_ATTRIBUTE_WRITE

Characteristic short name used at code generation
Use : “LED_C”

Profile Creation
Configure 2nd Characteristic



UUID : 19 ED 82 AE ED 21 4C 9D 41 45 22 8E 42 FE 00 00

Properties

Data (**2 bytes**) as a **notify** characteristic
Each time user press button over NUCLEO, information sent to client

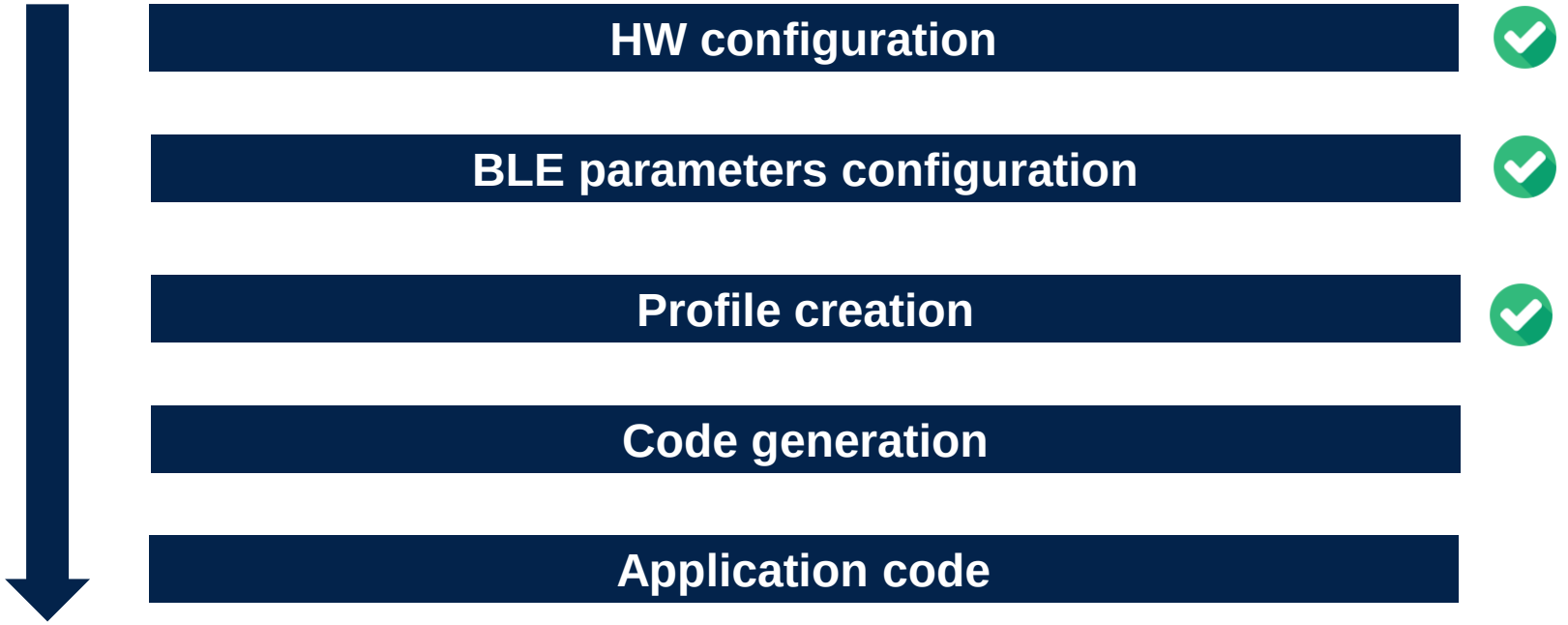
Permission

Here permission has not impact. The server is here sending data to client

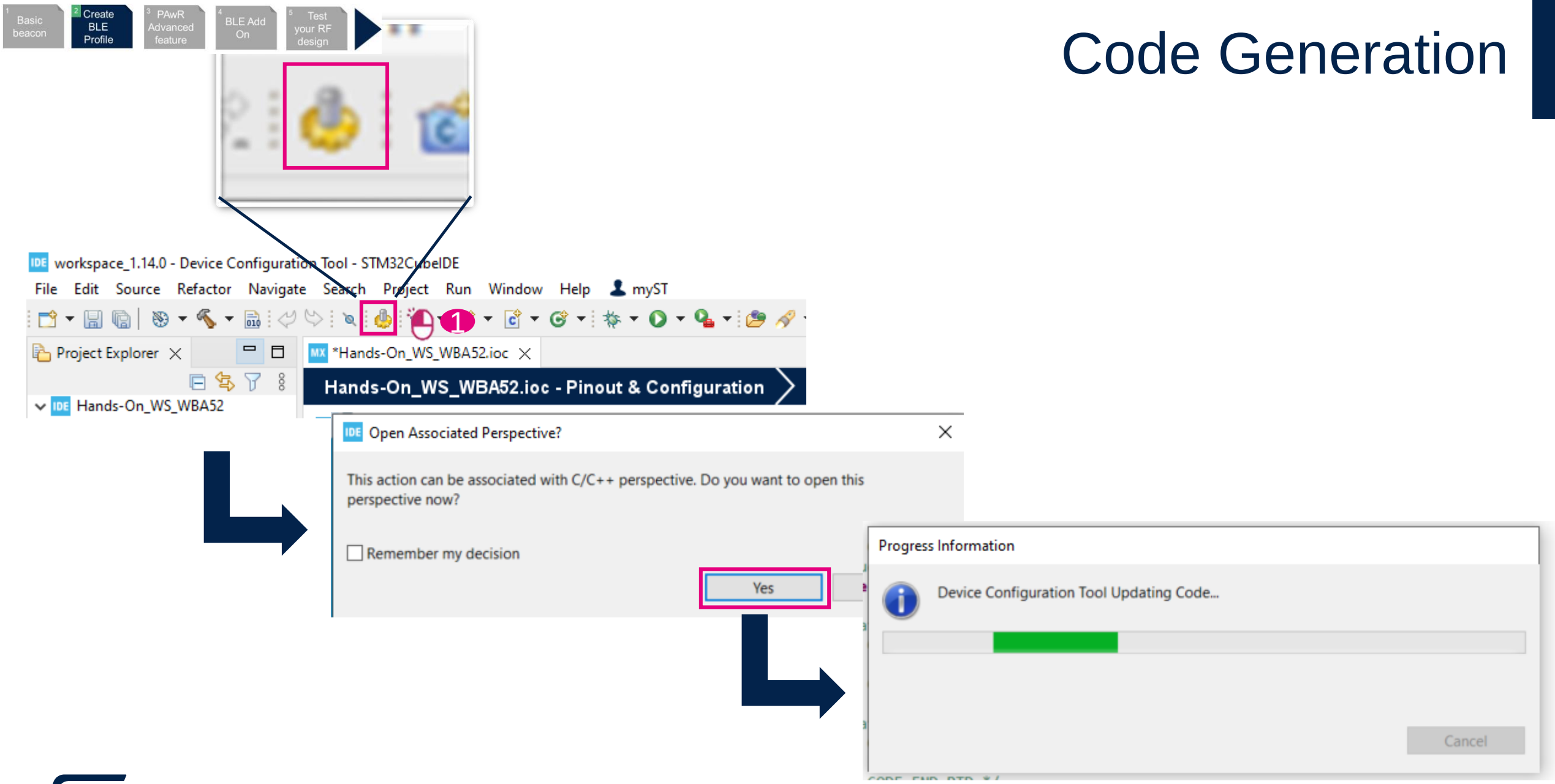
	Characteristic 1	Characteristic 2
UUID type	128 bits UUID (0x02)	128 bits UUID (0x02)
UUID 128 Input type	full	full
UUID	19 ED 82 AE ED 21 4C 9D 41 45 22 8E 41 FE 00 00	19 ED 82 AE ED 21 4C 9D 41 45 22 8E 42 FE 00 00
Characteristic long name	My_LED_Char	My_Switch_Char
Characteristic Short Name	LED_C	SWITCH_C
Value length	2	2
Length characteristic	Variable	Variable
Encryption key size	0x10	0x10
Char Properties	READ | WRITE_WITHOUT_RESP	NOTIFY
GATT events	GATT_NOTIFY_ATTRIBUTE_WRITE	GATT_NOTIFY_ATTRIBUTE_WRITE

Characteristic short name used at code generation
Use : "SWITCH_C"

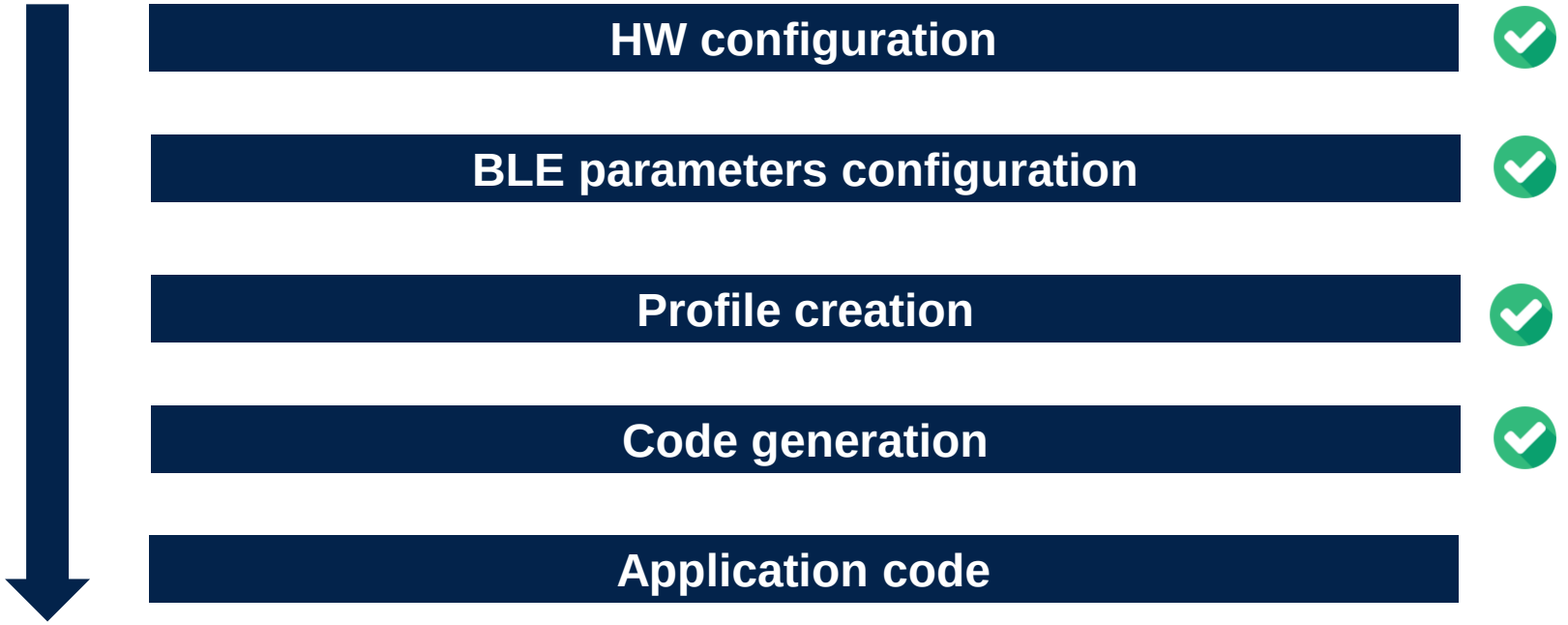
Configuration completed
What's next ? Code generation



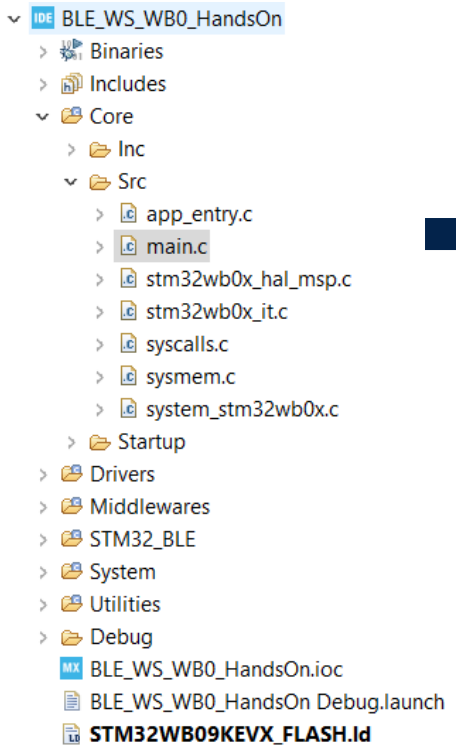
Code Generation



Configuration completed
What's next – Application Code



SYSCFG clock activation in Main.c



Missing SysCfg clock:

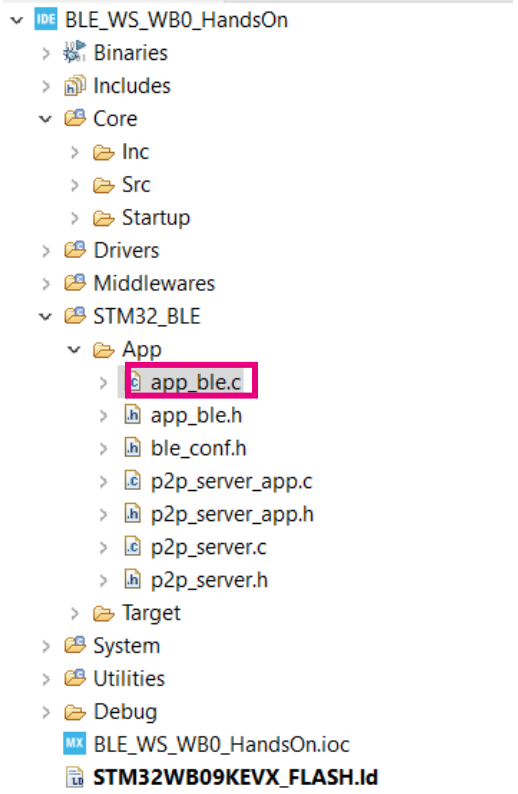
In main.c > function MX_GPIO_Init ()

```
/* USER CODE BEGIN MX_GPIO_Init_1 */
  __HAL_RCC_SYSCFG_CLK_ENABLE();
/* USER CODE END MX_GPIO_Init_1 */
```

This is workaround linked to an issue found in CubeIDE1.16 code generation where SYSCFG clock is not enabled. It will be fixed in next release and SYSCFG clk will be automatically enabled when soem GPIO_IRQs are enabled.

life.augmented

Move to discoverable
Start advertising



Set device discoverable at init :
In app_ble.c > function APP_BLE_Init()

(ADV_MIN+ADV_MAX)/2

```
/* USER CODE BEGIN APP_BLE_Init_2 */
APP_BLE_Procedure_Gap_Peripheral(PROC_GAP_PERIPH_ADVERTISE_START_FAST);
/* USER CODE END APP_BLE_Init_2 */
```

Search for "APP_BLE_Init_2"

Set device discoverable at disconnection :
In app_ble.c > SVCCTL_App_Notification - HCI_DISCONNECTION_COMPLETE_EVT_CODE

```
/* USER CODE BEGIN EVT_DISCONN_COMPLETE */
APP_BLE_Procedure_Gap_Peripheral(PROC_GAP_PERIPH_ADVERTISE_START_FAST);
/* USER CODE END EVT_DISCONN_COMPLETE */
```

Search for "EVT_DISCONN_COMPLETE "

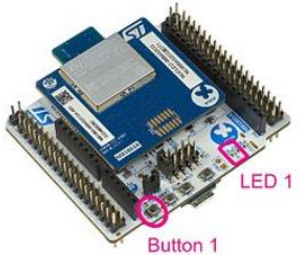
At disconnection, stack is not moving back to advertising, this is an application decision

Open Project | Add application code to move to discoverable | Build& Flash

Please refer to cheatsheet for copy/paste

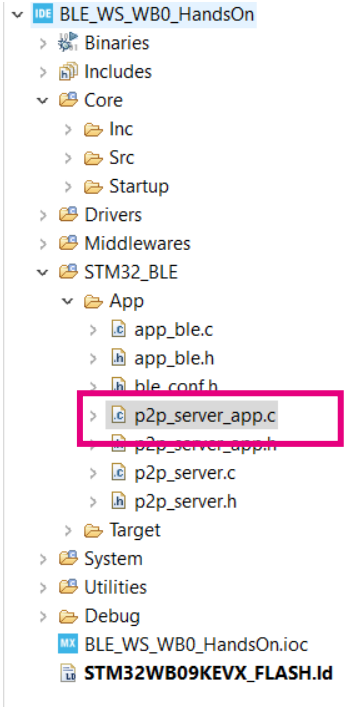
Add application code
Toggle LED from client

write to My_LED_Char



	Characteristic 1	Characteristic 2
UUID type	128 bits UUID (0x02)	128 bits UUID (0x02)
UUID 128 Input type	Reduced	Reduced
UUID	FE 41	FE 42
Characteristic long name	My_LED_Char	My_Switch_Char
Characteristic Short Name	LED_C	SWITCH_C
Value length	2	2
Length characteristic	Variable	Variable
Encryption key size	0x10	0x10
Char Properties	READ | WRITE_WITHOUT_RESP	NOTIFY
GATT events	GATT_NOTIFY_ATTRIBUTE_WRITE	GATT_NOTIFY_ATTRIBUTE_WRITE

write client procedure triggers an ACI_GATT_ATTRIBUTE_MODIFIED_VSEVT_CODE at server application level



in p2p_server_app.c / function P2P_SERVER_Notification()

```
/* USER CODE BEGIN Service1Char1_WRITE_NO_RESP_EVT*/
HAL_GPIO_TogglePin(GPIOB, LED_GREEN_Pin|LED_BLUE_Pin|LED_RED_Pin); /*
USER CODE END Service1Char1_WRITE_NO_RESP_EVT */<
```

Search for “Service1Char1_WRITE_NO_RESP_EVT”

life.augmented

How to add a task in sequencer ?

#1 Define a **TaskID** for your « new task » :

In app_conf.h
define a new ID in enum CFG_Task_Id_t
(USER code section)

```
/**
 * These are the lists of task id registered to the sequencer
 * Each task id shall be in the range [0:31]
 */
typedef enum
{
  CFG_TASK_BLE_STACK,
  CFG_TASK_VTIMER,
  CFG_TASK_NVM,
  /* USER CODE BEGIN CFG_Task_Id_t */
  TASK_BUTTON_1,
  /* USER CODE END CFG_Task_Id_t */
  CFG_TASK_NBR,  /**< Shall be LAST in the list */
} CFG_Task_Id_t;
```

#2 **UTIL_SEQ_RegTask()** to register your task in the sequencer

```
UTIL_SEQ_RegTask(1U << TASK_BUTTON_1, UTIL_SEQ_RFU, P2P_SERVER_Switch_c_SendNotification);
```

It associates a callback to your Task.
To be done only Once

#3 **UTIL_SEQ_SetTask()** to notify the sequencer shall execute the registered task

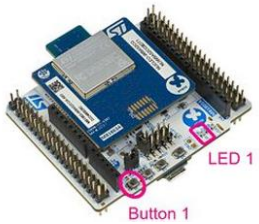
```
UTIL_SEQ_SetTask(1U << TASK_BUTTON_1, CFG_SEQ_PRIO_0);
```

It notify the sequencer that the task must be triggered.
It will generate a call to registered function
(here : APPE_Button1Action())

life.augmented

Add application code

Raise an alarm from device to Smartphone(1/3)

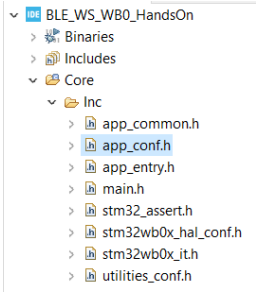


press button

notify peer device trough SWITCH_C (FE 42)

	Characteristic 1	Characteristic 2
UUID type	128 bits UUID (0x02)	128 bits UUID (0x02)
UUID 128 Input type	Reduced	Reduced
UUID	FE 41	FE 42
Characteristic long name	My_LED_Char	My_Switch_Char
Characteristic Short Name	LED_C	SWITCH_C
Value length	2	2
Length characteristic	Variable	Variable
Encryption key size	0x10	0x10
Char Properties	READ | WRITE_WITHOUT_RESP	NOTIFY
GATT events	GATT_NOTIFY_ATTRIBUTE_WRITE	GATT_NOTIFY_ATTRIBUTE_WRITE

On press button use notify procedure use to push data to client

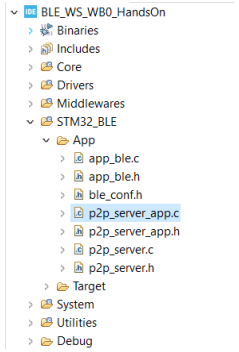


#1 need to define specific task for button press

In app_conf.h

```
/* USER CODE BEGIN CFG_Task_Id_t */
TASK_BUTTON_1,
/* USER CODE END CFG_Task_Id_t*/
```

Search for "*CFG_Task_Id_t*"



#2 register a « button task »

in p2p_server_app.c / function P2P_SERVER_APP_Init

```
/* USER CODE BEGIN Service1_APP_Init */
 UTIL_SEQ_RegTask( 1U << TASK_BUTTON_1, UTIL_SEQ_RFU, P2P_SERVER_Switch_c_SendNotification);
 /* USER CODE END Service1_APP_Init */
```

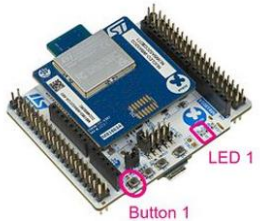
Search for "*Service1_APP_Init*"

Function generated by CubeMx as per as Characteristic Short Name

life.augmented

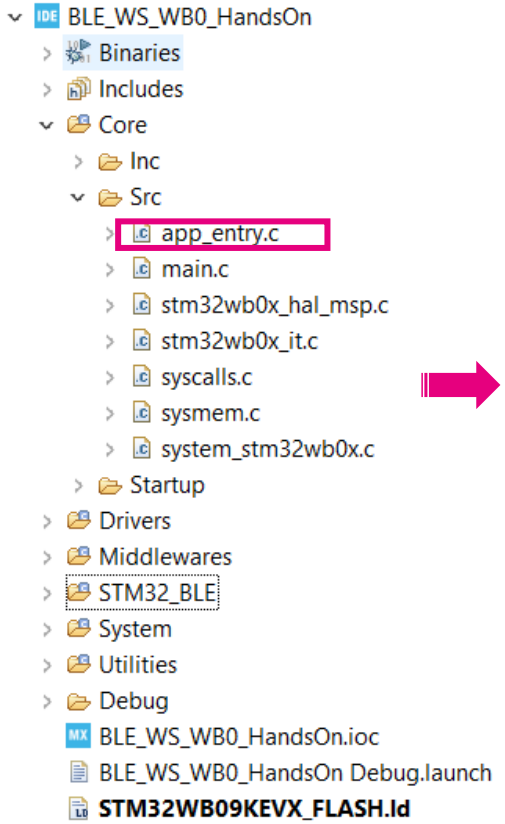
Add application code

Raise an alarm from device to Smartphone(2/3)



press button

notify peer device trough SWITCH_C



#3 Manage Button1 interrupt : implement IRQ callback

In app_entry.c / function HAL_GPIO_EXTI_Callback()

```
  /* USER CODE BEGIN FD_WRAP_FUNCTIONS */
void HAL_GPIO_EXTI_Callback(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
                if (GPIO_Pin == B1_Pin)
                {
                  UTIL_SEQ_SetTask(1U << TASK_BUTTON_1,
CFG_SEQ_PRIO_0);
                }
                return;
}
/* USER CODE END FD_WRAP_FUNCTIONS */
```

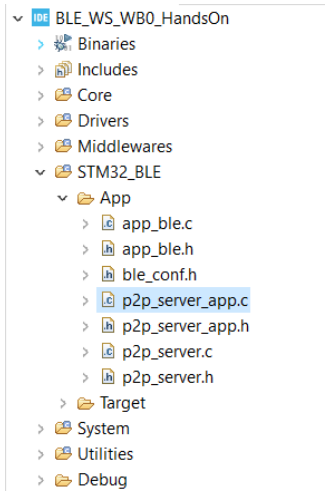
Copy function (weak) at end of file – under FD_WRAP_FUNCTIONS tags

Add application code

Raise an alarm from device to Smartphone(3/3)

notify peer device trough SWITCH_C

	Characteristic 1	Characteristic 2
UUID type	128 bits UUID (0x02)	128 bits UUID (0x02)
UUID 128 Input type	Reduced	Reduced
UUID	FE 41	FE 42
Characteristic long name	My_LED_Char	My_Switch_Char
Characteristic Short Name	LED_C	SWITCH_C
Value length	2	2
Length characteristic	Variable	Variable
Encryption key size	0x10	0x10
Char Properties	READ | WRITE_WITHOUT_RESP	NOTIFY
GATT events	GATT_NOTIFY_ATTRIBUTE_WRITE	GATT_NOTIFY_ATTRIBUTE_WRITE



#4 Manage BLE notification procedure

In p2p_server_app.c/ function P2P_SERVER_Switch_c_SendNotification

```
/* USER CODE BEGIN Service1Char2_NS_1*/

a_P2P_SERVER_UpdateCharData[0] = 0x01; /* Device Led selection */
a_P2P_SERVER_UpdateCharData[1] = 0x00;
/* Update notification data length */
p2p_server_notification_data.Length = (p2p_server_notification_data.Length) + 2;

notification_on_off = Switch_c_NOTIFICATION_ON;

/* USER CODE END Service1Char2_NS_1*/
```

Peer to Peer Service - SWITCH Characteristic

Byte Index	0	1
Name	Button Selection	Status
Value	0x01: button 1	0x00 or 0x01

STM32WBA Bluetooth® LE – Peer 2 Peer Applications - stm32mcu

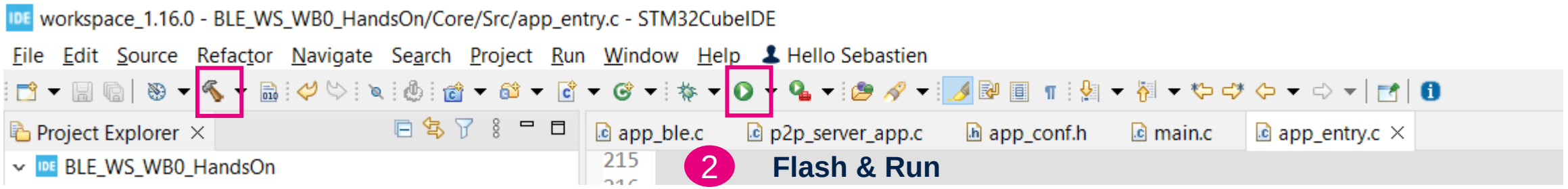
Search for "*Service1Char2_NS_1* "

P2P_SERVER_UpdateValue

aci_gatt_update_char_value BLE stack API

Time to build, flash and execute !

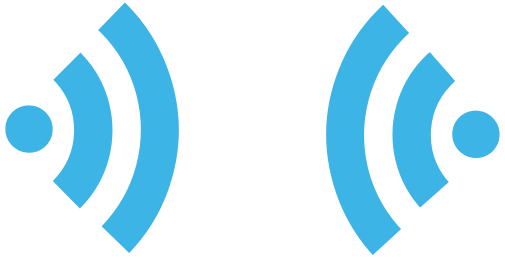
1 **Build**



2 **Flash & Run**

life.augmented

Open your App and Connect

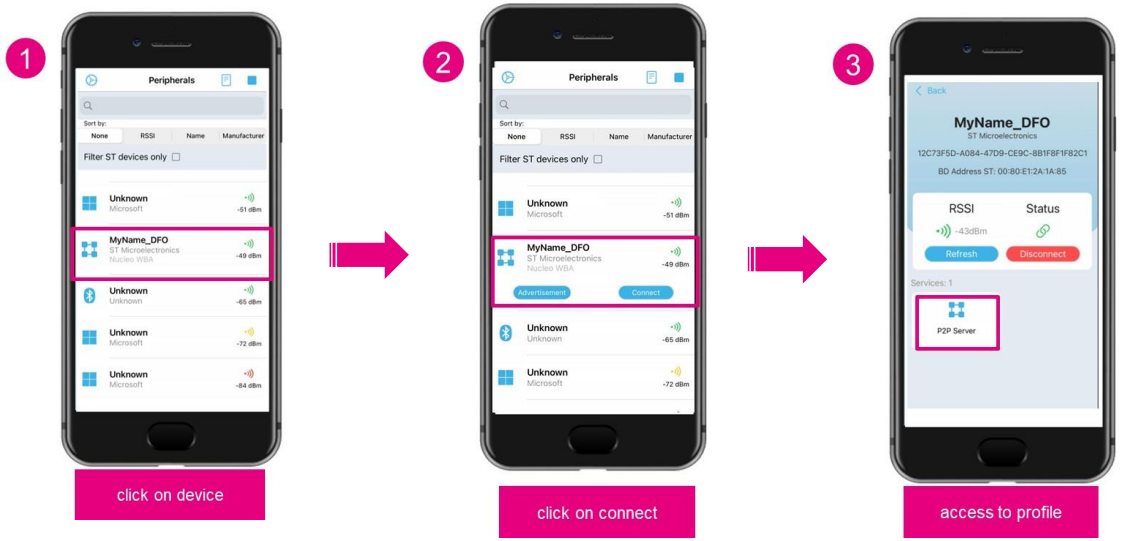


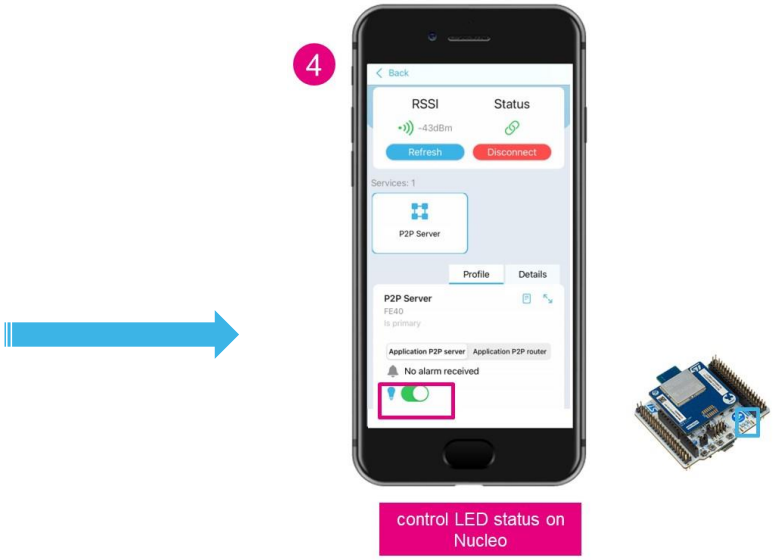
ST BLE Toolbox

Download on the App Store

GET IT ON Google Play

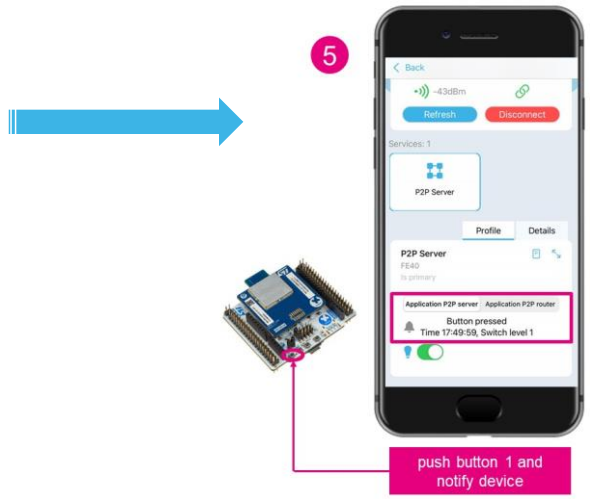
Open your App and Connect (1/2)





	Characteristic 1
UUID type	128 bits UUID (0x02)
UUID 128 Input type	Reduced
UUID	FE 41
Characteristic long name	My_LED_Char
Characteristic Short Name	LED_C
Value length	2
Length characteristic	Variable
Encryption key size	0x10
Char Properties	READ | WRITE_WITHOUT_RESP
GATT events	GATT_NOTIFY_ATTRIBUTE_WRITE

write client procedure control LED



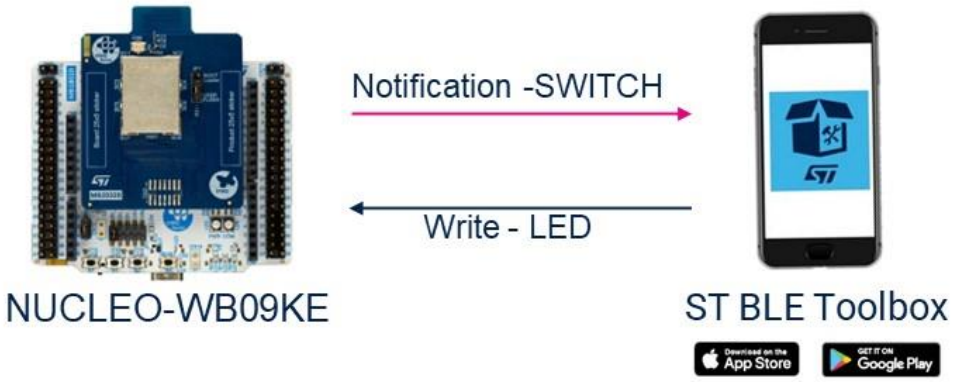
	Characteristic 2
UUID type	128 bits UUID (0x02)
UUID 128 Input type	Reduced
UUID	FE 42
Characteristic long name	My_Switch_Char
Characteristic Short Name	SWITCH_C
Value length	2
Length characteristic	Variable
Encryption key size	0x10
Char Properties	NOTIFY
GATT events	GATT_NOTIFY_ATTRIBUTE_WRITE

notify procedure use to push data to client

Takeaways

Easily create your own BLE application on STM32WB0

Thanks to user friendly STM32ecosystem, create in few clicks your own application like BLE_P2PServer application just demonstrated.

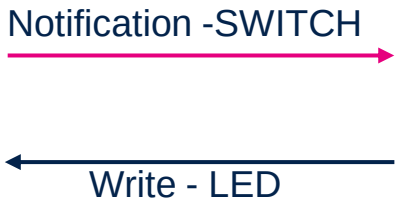


STM32 CubeIDE

Thank you

life.augmented

Implementing proprietary profile P2P_Server



NUCLEO-WB09KE

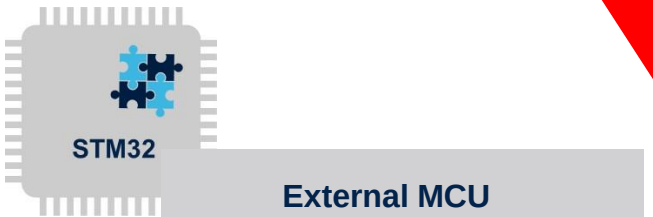
ST BLE Toolbox

Download on the App Store
GET IT ON Google Play

life.augmented

STM32WB05N
Network Processor architecture

STM32WB05xN is suitable f ations where B ® Low Energy needs to be added to existing s

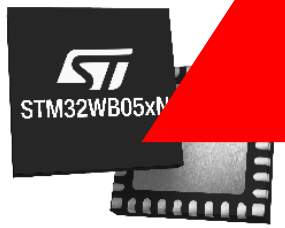


External MCU

Application

ACI & Middleware (driver)

SPI/UART



BLE Ful stack

X-CUBE-WB05N

Easy integration over STM32CubeMx

External MCU/MPU

Application

3rd party Bluetooth Stack

HCI

SPI/UART

BLE stack HCI only

STM32WB05N fw image @st. com

X-CUBE-WB05N - Bluetooth low energy software expansion for STM32Cube

life.augmented