

# **P2P server application**

## **BLE profile concept & build**

# Let's start enabling Bluetooth Low Energy applications with STM32WBx

## Create BLE Profile

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

2

1

Basic  
beacon

2

Create  
BLE  
Profile

3

PAwR  
Advanced  
feature

4

BLE Add  
On

5

Test  
your RF  
design



# 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



NUCLEO-WB09KE

Notification -SWITCH

Write - LED



ST BLE Toolbox



# 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”](#)



# Open and Start STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help

New Alt+Shift+N

Open File...

Open Projects from File System...

Recent Files

Close Editor Ctrl+W

Close All Editors Ctrl+Shift+W

Save Ctrl+S

Save As...

Save All Ctrl+Shift+S

Revert

Move...

Rename... F2

Refresh F5

Convert Line Delimiters To

Print... Ctrl+P

**Import...**

Export...

Properties Alt+Enter

Switch Workspace

Restart

Exit

Import

Import an Existing STM32CubeMX Configuration File (.ioc)

Select an import wizard:

Filter text

- General
  - Archive File
  - Existing Projects into Workspace
  - File System
  - Import ac6 System Workbench for STM32 Project
  - Import an Existing STM32CubeMX Configuration File (.ioc)**
  - Import Atollic TrueSTUDIO Project
  - Import STM32Cube Example
  - Preferences

< Back Next >

STM32 Project From Existing STM32CubeMX Configurati...

Setup STM32 project

STM32CubeMX .ioc file

File: C:\Data\WS\_WB0\BLE\_WS\_WB0\_HandsOn.ioc Browse...

Project

Project Name: BLE\_WS\_WB0\_HandsOn

☒ Use default location

Location: C:/Users/sdenoual/STM32CubeIDE/workspace\_1.16.0 Browse...

Options

Targeted Language

☒ C ☐ C++

Targeted Binary Type

☐ Executable ☐ Static Library

Targeted Project Type

☒ STM32Cube ☐ Empty

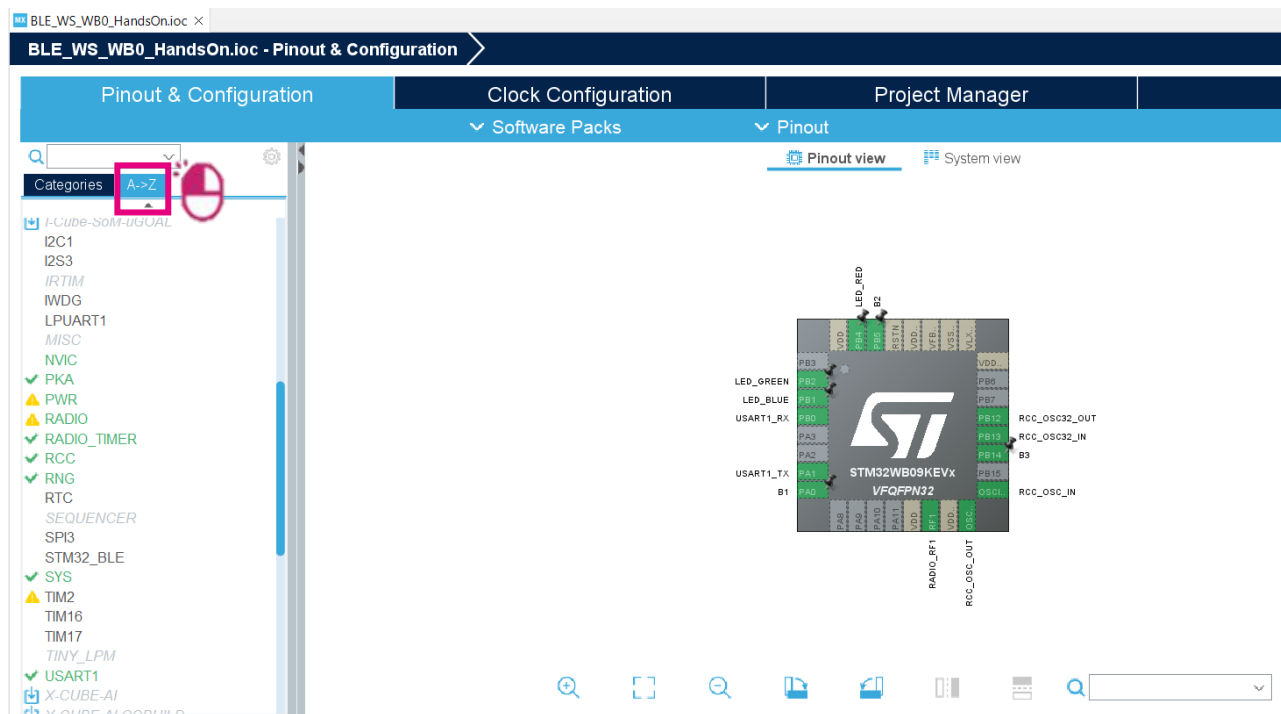
< Back Next > Finish Cancel



# 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”

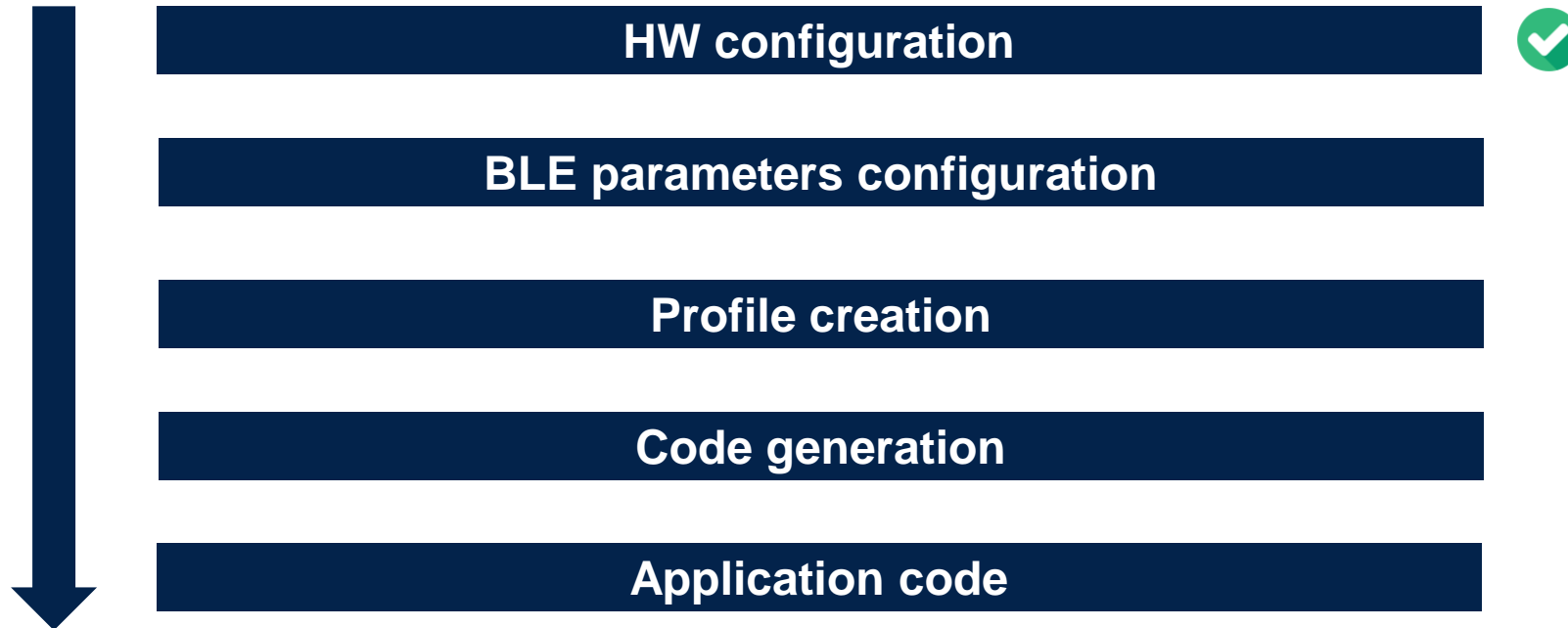


## BLE\_WS\_WB0\_HandsOn.ioc

- HW configuration (LED/Button) and needed peripherals for BLE activities
  - enable STM32\_BLE (**BLE middleware activation**)
  - We can focus on Bluetooth Low Energy application!



## What's next ? BLE config and Profile creation

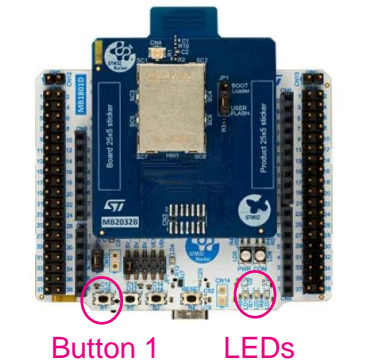


# Implementing proprietary profile

## P2P\_Server



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



NUCLEO-WB09KE



All services/characteristics are identified by a UUID (**U**niversally **U**nique **I**dentifier)

Characteristic can take 3 types of properties: **READ**, **WRITE**, **NOTIFY**

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

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

Server

# Advertising Elements configuration

STM32\_BLE Mode and Configuration

Mode

BLE Create your Peripheral & GATT server application

Configuration

Reset Configuration

Configuration

BLE Applications and Services

BLE Advertising

User Constants

Platform Settings

NVIC Settings

Configure the below parameters :

Search (Ctrl+F)

Advertising elements

ad\_data[] length

27

Include INCLUDE\_AD\_TYPE\_FLAGS element

Yes

AD\_TYPE\_FLAGS

FLAG\_BIT\_LE\_GENERAL\_DISCOVERABLE\_MODE|FLAG\_BIT\_BR\_EDR...

Include AD\_TYPE\_TX\_POWER\_LEVEL element

No

Include AD\_TYPE\_COMPLETE\_LOCAL\_NAME element

Yes

AD\_TYPE\_COMPLETE\_LOCAL\_NAME\_LENGTH

7

AD\_TYPE\_COMPLETE\_LOCAL\_NAME

WS\_WB0

Include AD\_TYPE\_SHORTENED\_LOCAL\_NAME element

No

Include AD\_TYPE\_APPEARANCE element

No

Include AD\_TYPE\_ADVERTISING\_INTERVAL element

No

Include AD\_TYPE\_LE\_ROLE element

No

Include AD\_TYPE\_16\_BIT\_SERV\_UUID\_CMPLT\_LIST element

No

Include AD\_TYPE\_128\_BIT\_SERV\_UUID\_CMPLT\_LIST element

No

Include AD\_TYPE\_SLAVE\_CONN\_INTERVAL element

No

Include AD\_TYPE\_URI element

No

Include AD\_TYPE\_MANUFACTURER\_SPECIFIC\_DATA element

Yes

AD\_TYPE\_MANUFACTURER\_SPECIFIC\_DATA\_LENGTH

15

Company identifier

30.00

Number of user defined data item(s)

12

User defined data 1

00

Comment data 1

\* User defined data 2

00

\* Comment data 2

\* User defined data 3

00

\* Comment data 3

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

1

Reset Configuration

Configuration

BLE Applications and Services

BLE Advertising

User Constants

Platform Settings

NVIC Settings

Configure the below parameters :

Search (Ctrl+F)

Application configuration - Application parameters

CFG\_LSLCK\_LSE

LSE used

CFG\_TX\_POWER\_MODE

0 (normal power mode)

CFG\_TX\_POWER

0 dBm (0x18)

CFG\_PUBLIC\_BD\_ADDRESS

0x0008E12A1234

CFG\_BD\_ADDRESS\_TYPE

Public address(0)

CFG\_BLE\_PRIVACY\_ENABLED

Disabled

ADV\_INTERVAL\_MIN

80

ADV\_INTERVAL\_MAX

100

ADV\_LP\_INTERVAL\_MIN

1600

ADV\_LP\_INTERVAL\_MAX

4000

ADV\_TYPE

Connectable and scannable undirected

ADV\_FILTER

No white list(0x00)

CFG\_BONDING\_MODE

No-bonding mode(0x00)

CFG\_FIXED\_PIN

111111

CFG\_ENCRYPTION\_KEY\_SIZE\_MAX

16

CFG\_ENCRYPTION\_KEY\_SIZE\_MIN

8

CFG\_IO\_CAPABILITY

Display Yes No (0x01)

CFG\_MITM\_PROTECTION

MITM protection required (0x01)

CFG\_SC\_SUPPORT

Secure Connections Paring supported but optional (0x01)

GAP\_KEYPRESS\_NOT\_SUPPORTED

Keypress notification not supported (0x00)

CFG\_GAP\_APPEARANCE

Appearance Unknown (0x0000)

CFG\_GAP\_DEVICE\_NAME

WS\_WB0

CFG\_GAP\_DEVICE\_NAME\_LENGTH

8

CFG\_SCAN\_INT\_MS

500

CFG\_SCAN\_WIN\_MS

500

Application configuration - BLE stack

CFG\_GAP\_DEVICE\_NAME

CFG\_GAP\_DEVICE\_NAME

Parameter Description:

Device name; default value is TEMPLATE

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)

# Platform Settings

## Trace & Logs: BSP settings

Pinout & Configuration

Clock Configuration

Project Manager

Software Packs

Pinout

STM32\_BLE Mode and Configuration

Mode

BLE Create your Peripheral & GATT server application

Configuration

Reset Configuration

ConfigurationBLE Applications and ServicesBLE AdvertisingUser ConstantsPlatform SettingsNVIC Settings

Platform proposal

BSP

Name	IPs or Components	Found Solutions	BSP API
Serial Link for Traces	USART:Asynchronous	USART1	Unknown

Anticipate Logs activation

Logs activation may require application configuration changes

# Profile Creation Service

- 1 Basic beacon
- 2 Create BLE Profile
- 3 PAwR Advanced feature
- 4 BLE Add On
- 5 Test your RF design

Mode

BLE Create your Peripheral & GATT server application

Configuration

Reset Configuration

SERVICE1

User Constants

Platform Settings

Configuration

BLE Applications and Services

Configure the below parameters :

Search (Ctrl+F)

Server Mode

Number of services

1

## P2P Server Profile

### P2P Service

My\_LED\_Char

SWITCH\_C

# Profile Creation

## Configure my P2P Service

1

SERVICE1

Configuration

User Constants

Platform Settings

NVIC Settings

BLE Applications and Services

BLE Advertising

Configure the below parameters :

Search (Ctrl+F)

Service

UUID type

UUID 128 input type

UUID

Service long name

Service short name

Type

Service max attributes record(s)

Number of characteristics

Characteristic1

Characteristic2

2

128 bits UUID(0x02)

full

8F E5 B3 D5 2E 7F 4A 98 2A 48 7A CC 40 FE 00 00

P2P\_Server

P2P\_Server

Primary Service(0x01)

5

2

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

SERVICE1User ConstantsPlatform SettingsNVIC SettingsConfigurationBLE Applications and ServicesBLE Advertising

Configure the below parameters :

Search (Ctrl+F)

> Service

Characteristic1

1

UUID type128 bits UUID(0x02)  
UUID 128 input typefull  
UUID19 ED 82 AE ED 21 4C 9D 41 45 22 8E 41 FE 00 00  
Characteristic typeBuffered  
Characteristic long nameMy\_LED\_Char  
Characteristic short nameLED\_C  
Value length2  
Length characteristicVariable  
Encryption Key Size0x10  
CHAR\_PROP\_BROADCASTNo  
CHAR\_PROP\_READYes  
CHAR\_PROP\_WRITE\_NO\_RESPYes  
CHAR\_PROP\_WRITENo  
CHAR\_PROP\_NOTIFYNo  
CHAR\_PROP\_INDICATENo  
ATTR\_PERMISSION\_AUTHEN\_READNo  
ATTR\_PERMISSION\_ENCRY\_READNo  
ATTR\_PERMISSION\_AUTHEN\_WRITENo  
ATTR\_PERMISSION\_ENCRY\_WRITENo  
GATT\_NOTIFY\_ATTRIBUTE\_WRITEYes  
GATT\_NOTIFY\_WRITE\_REQ\_AND\_WAIT\_FOR\_APPL\_RESPNo  
GATT\_NOTIFY\_READ\_REQ\_AND\_WAIT\_FOR\_APPL\_RESPNo  
GATT\_NOTIFY\_NOTIFICATION\_COMPLETIONNo

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	READWRITE_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 2<sup>nd</sup> Characteristic

SERVICE1

User Constants

Platform Settings

Service Settings

Configuration

BLE Applications and Services

BLE Advertising

Configure the below parameters :

Search (Ctrl+F)

> Service

> Characteristic1

> Characteristic2

UUID type

UUID 128 input type

UUID

Characteristic type

Characteristic long name

Characteristic short name

Value length

Length characteristic

Encryption Key Size

CHAR\_PROP\_BROADCAST

CHAR\_PROP\_READ

CHAR\_PROP\_WRITE\_NO\_RESP

CHAR\_PROP\_WRITE

CHAR\_PROP\_NOTIFY

CHAR\_PROP\_INDICATE

Update char value offset

ATTR\_PERMISSION\_AUTHEN\_READ

ATTR\_PERMISSION\_ENCRY\_READ

ATTR\_PERMISSION\_AUTHEN\_WRITE

ATTR\_PERMISSION\_ENCRY\_WRITE

GATT\_NOTIFY\_ATTRIBUTE\_WRITE

GATT\_NOTIFY\_WRITE\_REQ\_AND\_WAIT\_FOR\_APPL\_RESP

GATT\_NOTIFY\_READ\_REQ\_AND\_WAIT\_FOR\_APPL\_RESP

GATT\_NOTIFY\_NOTIFICATION\_COMPLETION

128 bits UUID(0x02)

full

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

Buffered

My\_Switch\_Char

SWITCH\_C

2

Variable

0x10

No

No

No

No

Yes

No

0

No

No

No

No

Yes

No

No

No

1

2

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

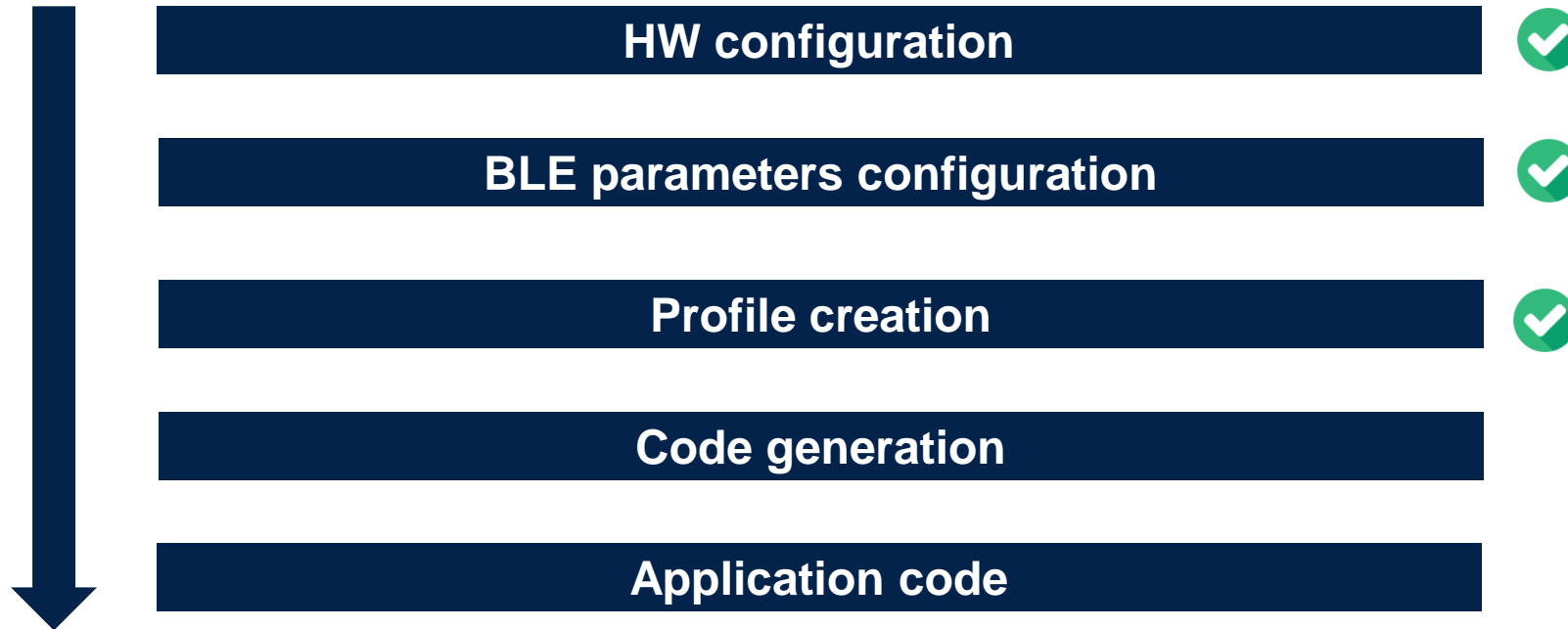
	Characteristic 1	Characteristic 2
UUID type	128 bits UUID (0x02)	128 bits UUID (0x02)
UUID 128 Input type	full	full
UUID	19 ED 82 AE ED 21 4C 9D 41 45 22 8E 41 FE 00 00	19 ED 82 AE ED 21 4C 9D 41 45 22 8E 42 FE 00 00
Characteristic long name	My_LED_Char	My_Switch_Char
Characteristic Short Name	LED_C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



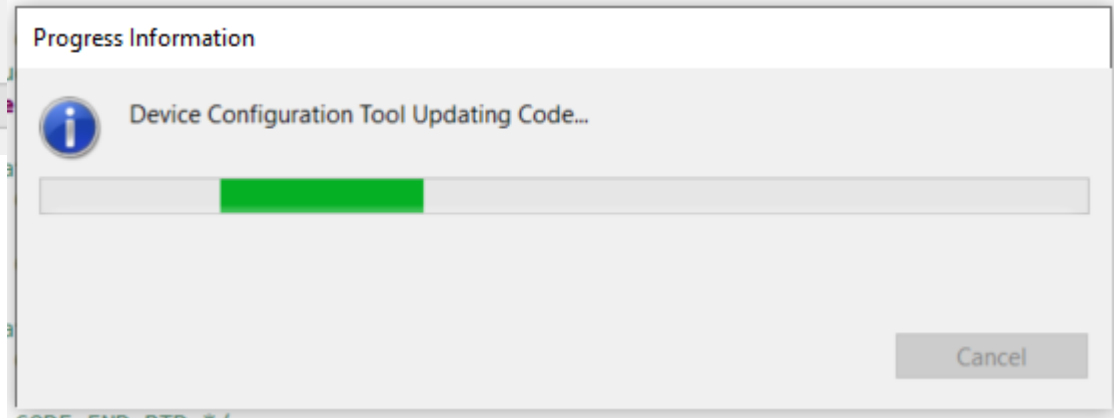
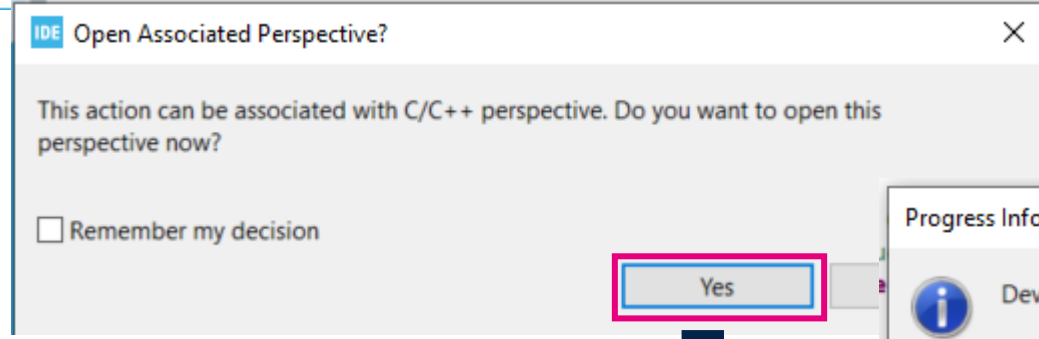
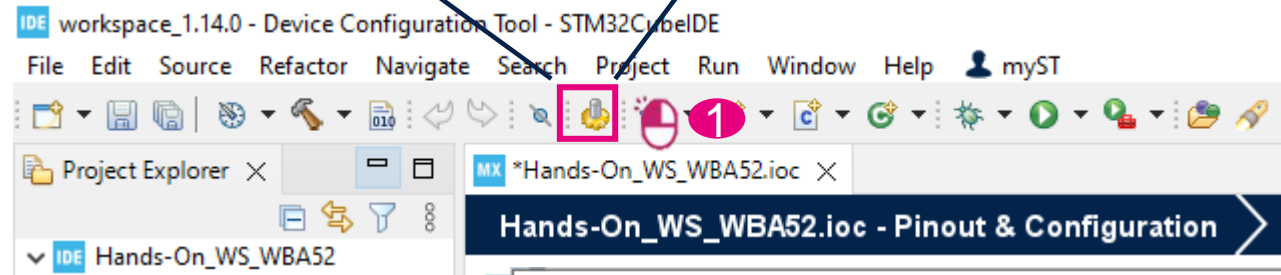
Characteristic short name used at code generation  
Use : "SWITCH\_C"

# Configuration completed

## What's next ? Code generation

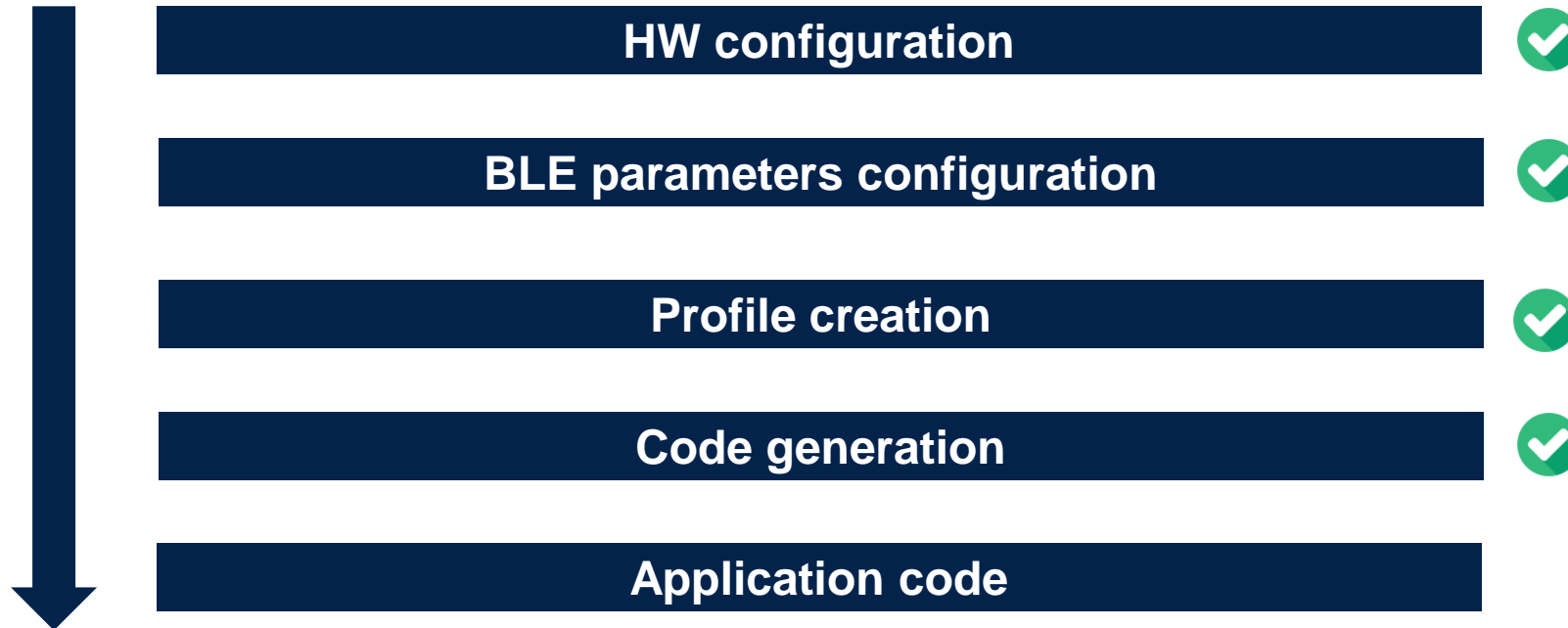


# Code Generation

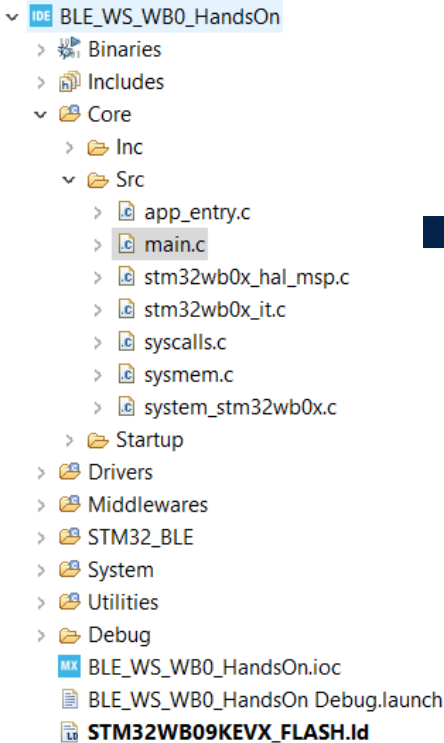


# Configuration completed

## What's next – Application Code



# SYSCFG clock activation in Main.c



**Missing SysCfg clock:**  
In main.c > function MX\_GPIO\_Init ()

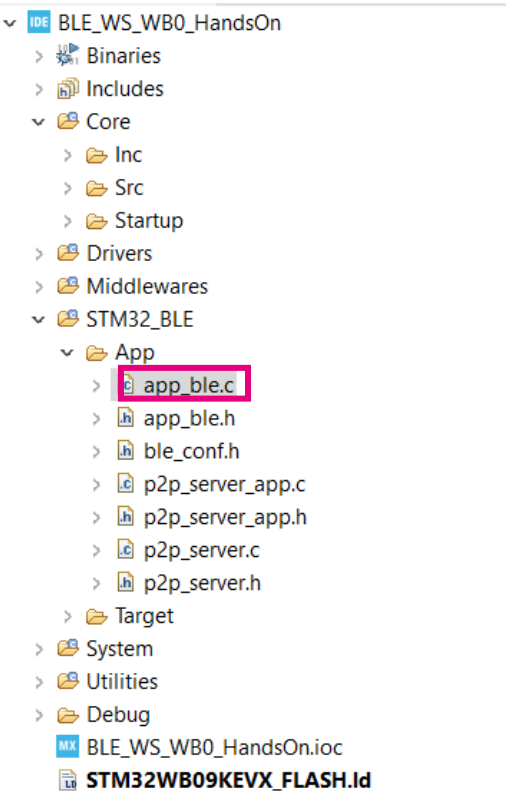
```

/* 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.

# Move to discoverable

## Start advertising



### Set device discoverable at init :

In app\_ble.c > function APP\_BLE\_Init()

```

/* 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 */

```

(ADV\_MIN+ADV\_MAX)/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 ”

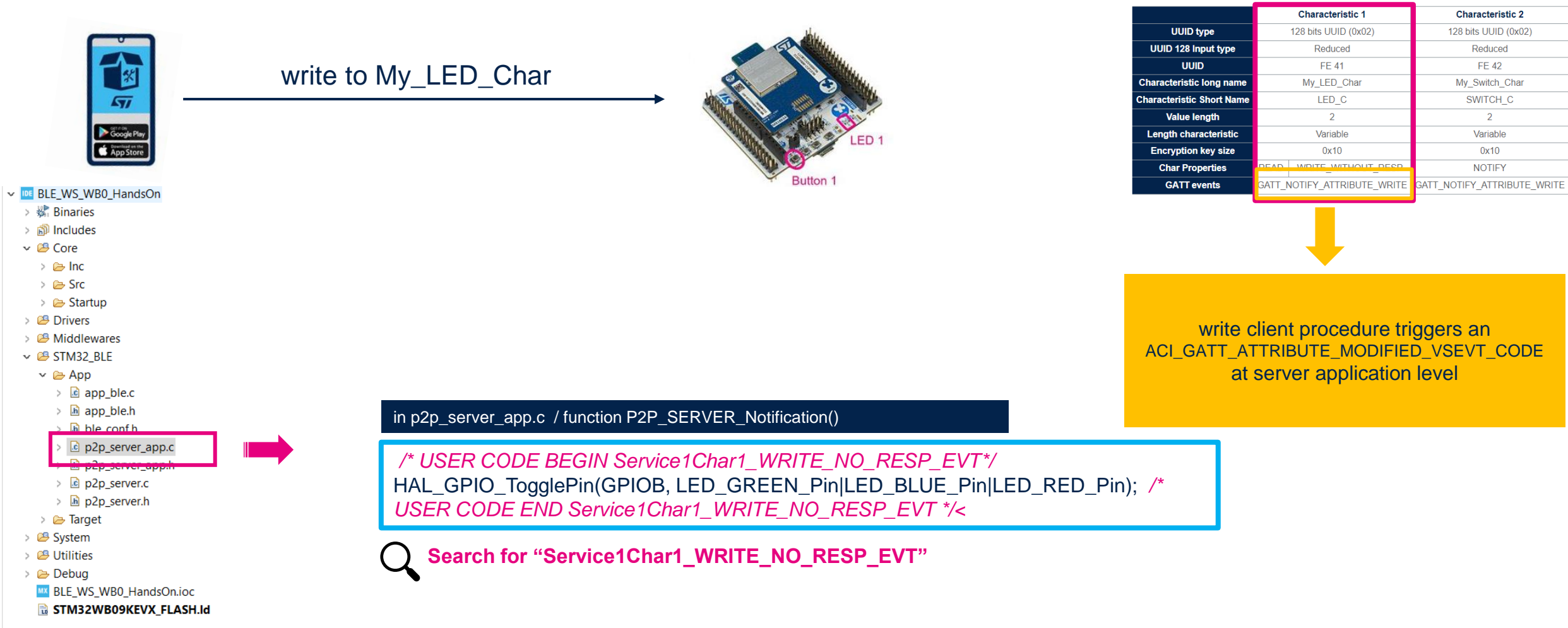


At disconnection, stack is not moving back to advertising, this is an application decision



# Add application code

## Toggle LED from client





# 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() )

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

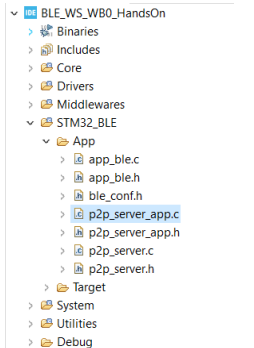
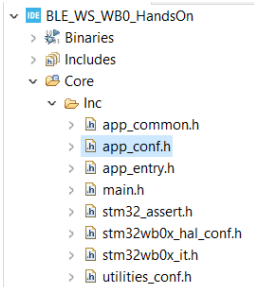


press button

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



#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 */

```

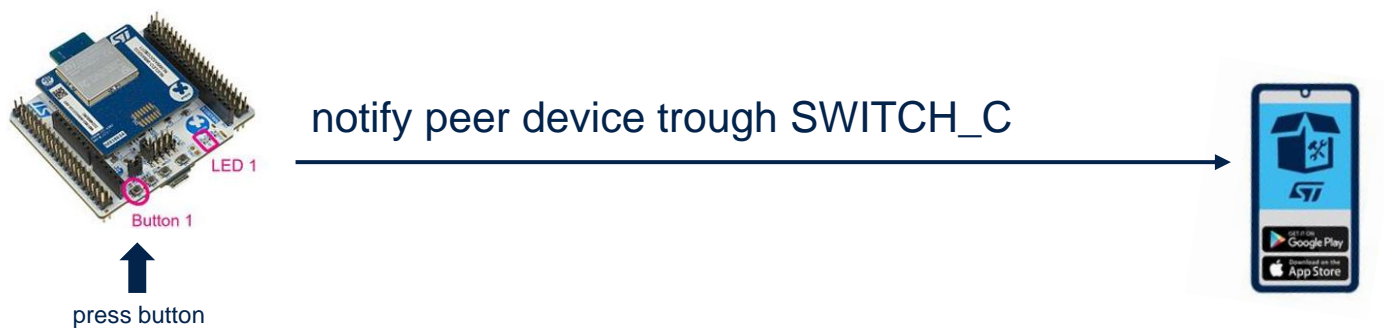
Search for “Service1\_APP\_Init”

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

Function generated by CubeMx as per as Characteristic Short Name

# Add application code

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



IDE

BLE\_WS\_WB0\_HandsOn

Binaries

Includes

Core

Inc

Src

app\_entry.c

main.c

stm32wb0x\_hal\_msp.c

stm32wb0x\_it.c

syscalls.c

systemem.c

system\_stm32wb0x.c

Startup

Drivers

Middlewares

STM32\_BLE

System

Utilities

Debug

BLE\_WS\_WB0\_HandsOn.ioc

BLE\_WS\_WB0\_HandsOn Debug.launch

STM32WB09KEVX\_FLASH.ld

#3 Manage Button1 interrupt : implement IRQ callback

In app\_entry.c / function HAL\_GPIO\_EXTI\_Callback()

```

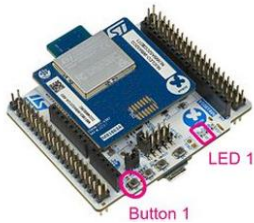
/* USER CODE BEGIN FD_WRAP_FUNCTIONS */
void HAL_GPIO_EXTI_Callback(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
{
    if (GPIO_Pin == B1_Pin)
    {
        UTIL_SEQ_SetTask(1U << TASK_BUTTON_1,
CFG_SEQ_PRIO_0);
    }
    return;
}
/* USER CODE END FD_WRAP_FUNCTIONS */

```

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;

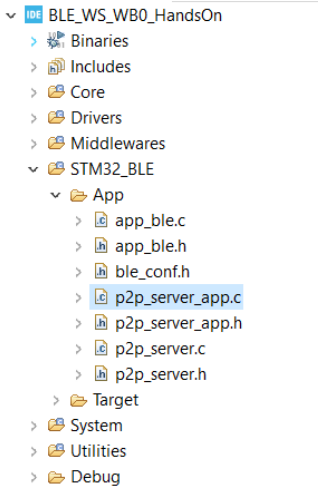
notification_on_off = Switch_c_NOTIFICATION_ON;

/* 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 ”

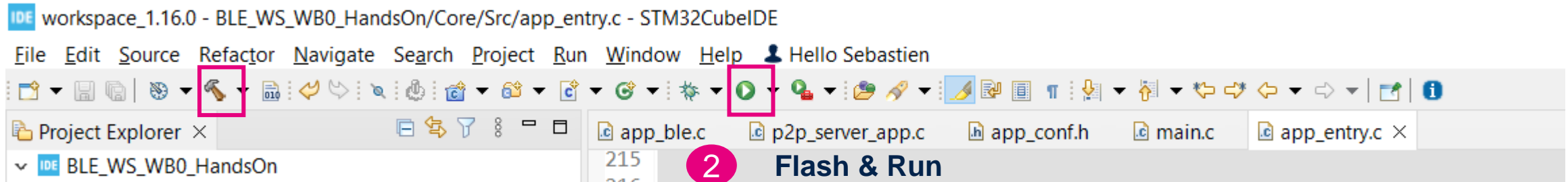
P2P\_SERVER\_UpdateValue

aci\_gatt\_update\_char\_value

BLE stack API

# Time to build, flash and execute !

## 1 Build



## 2 Flash & Run

Open Project

Add application code to move to discoverable

Build& Flash

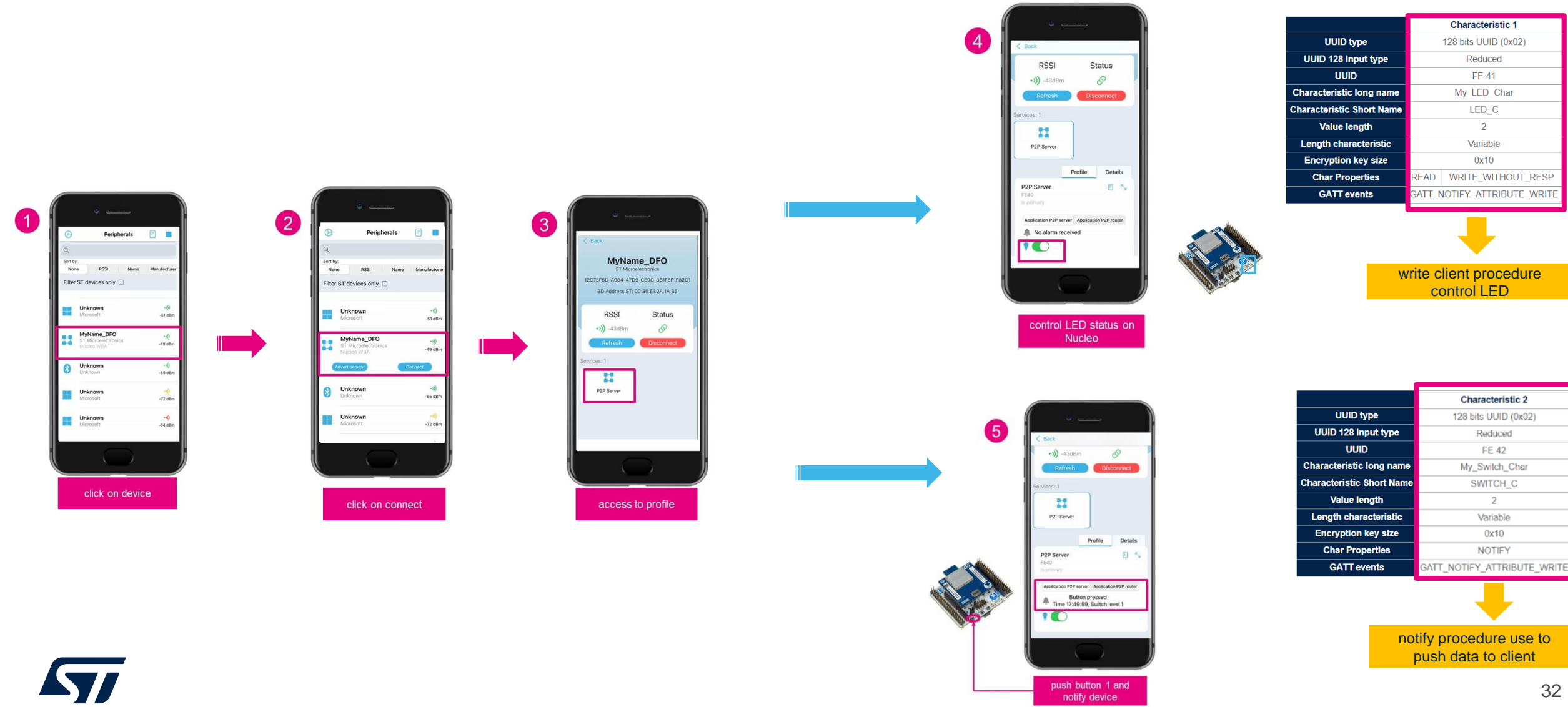
# Open your App and Connect



ST BLE Toolbox



# Open your App and Connect (1/2)

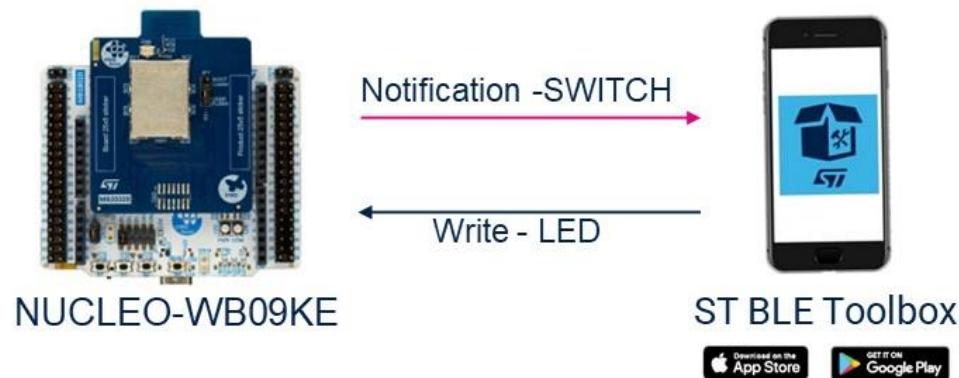




# Takeaways

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

© STMicroelectronics - All rights reserved.

The STMicroelectronics corporate logo is a registered trademark of the STMicroelectronics group of companies. All other names are the property of their respective owners.



life.augmented

# Implementing proprietary profile P2P\_Server

Notification -SWITCH



Write - LED



NUCLEO-WB09KE

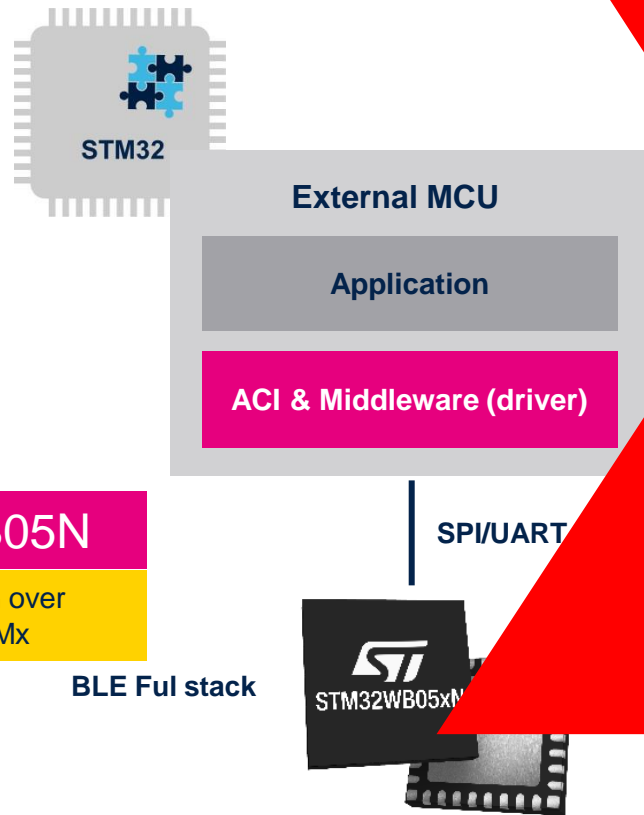


ST BLE Toolbox



## Network Processor architecture

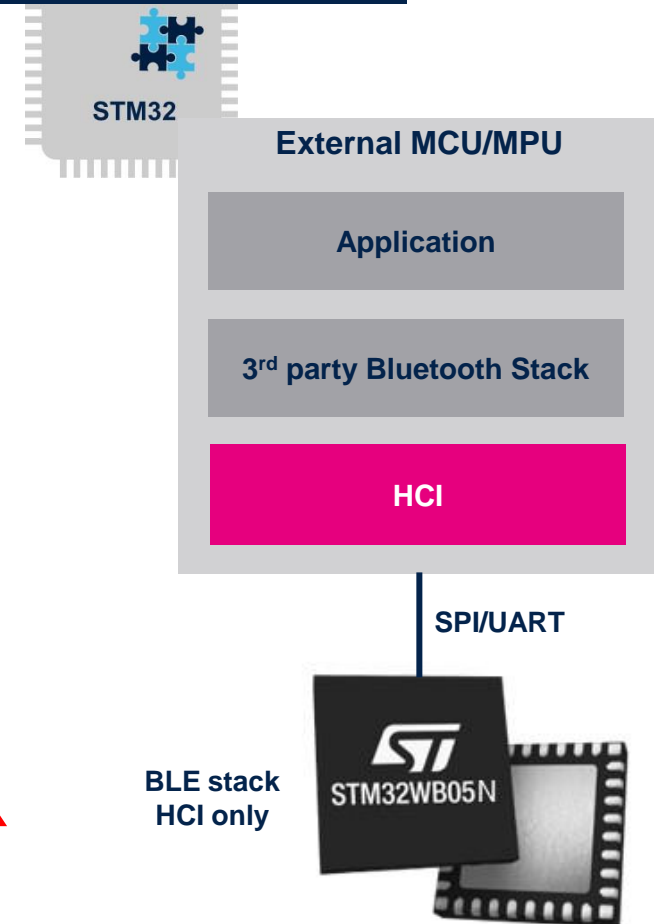
STM32WB05xN is suitable for applications where Bluetooth® Low Energy needs to be added to existing systems



X-CUBE-WB05N

Easy integration over  
STM32CubeMx

BLE Full stack



STM32WB05N fw image @st.com