



life.augmented



## *STM32WBA Hands-on #2* Build basic **p2pServer** application and connect



# Prerequisites Refresh

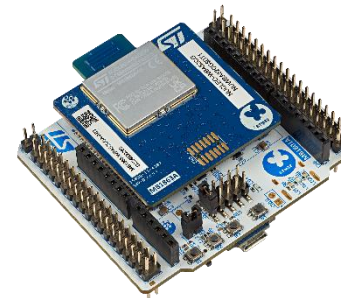
## SW prerequisites

- STM32CubeMX software (v6.9.0 or up)
- STM32CubeWBA MCU package (v1.1.0 or up)
- IDE: STM32CubeIDE
- A serial terminal (e.g. TeraTerm)
- **ST BLE ToolBox Smartphone application**

## HW prerequisites

- NUCLEO-WBA52
- USB A to Micro-B Cable

**Hands-on #1 to be first completed**



ST BLE Toolbox





# Agenda

1 Hands-on Presentation

3 Step 2 : Application code

2 Step 1 : Profile creation  
demystification and details

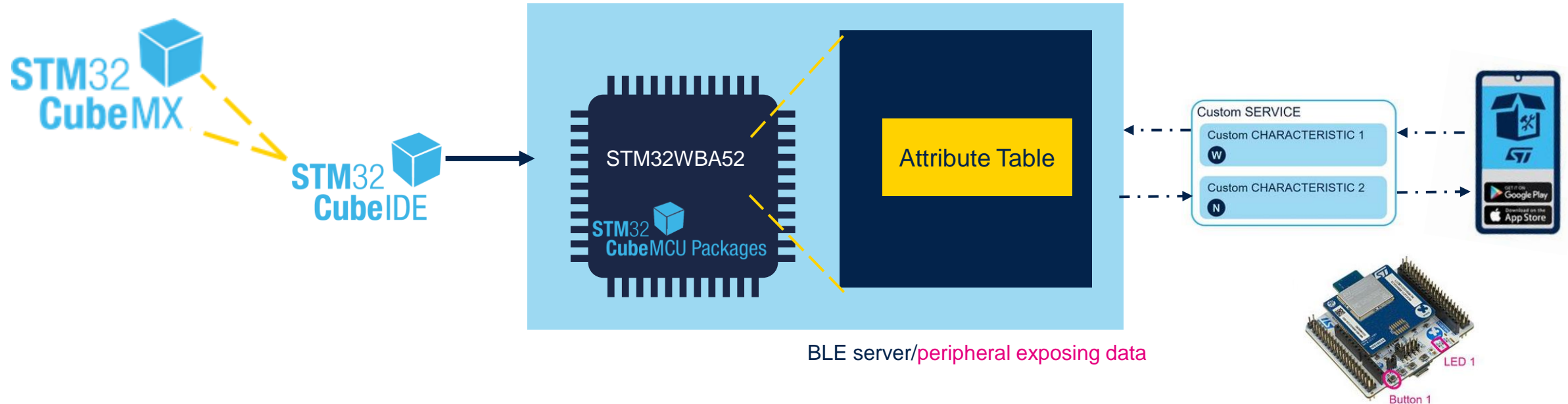
4 Step 3: Low Power



life.augmented

# Hands-on presentation

- The purpose is to start from WB5A52 chipset level and build a basic server (**p2pServer**) application using STM32CubeMX and associated STM32CubeIDE
- In this second part, focus is to enhance existing application code (Hands-on #1) to **control device and share data**



Enhance application code to enable a BLE Application Profile (**p2pServer**)

- Slides including following symbol are purely theoretical ones



- Optional steps during development in are marked with a grey bar



- Source code for development is included inside pink boxes

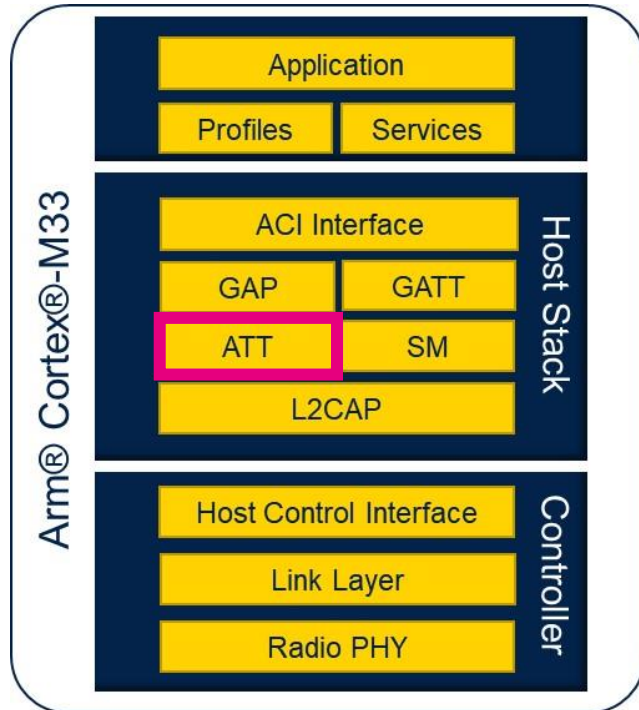
```
HAL_Delay(500);
```

# Step1 : GAP/GATT custom application configuration : **Profile creation**





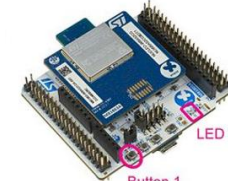
# What is a Bluetooth Low Energy Profile **Attribute Protocol (ATT)**



## Define **Client** - Server architecture



« I am looking for data »  
**Client**



« I have data to share » (sensors, raw data...**what you want !**)  
**Server**

## Logical data structure – « **How to access data** » : **Attribute table**

user data is accessible trough attribute

service

characteristic

data

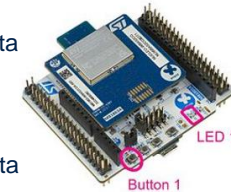
Custom SERVICE **UUID**

Custom CHARACTERISTIC 1  
**W** **UUID | value**

value = user data

Custom CHARACTERISTIC 2  
**N** **UUID | value**

value = user data





# What is a Bluetooth Low Energy Profile

A profile is a collection of data (**attributes**) exposes by device trough associated Service and Characteristic

## Profile

**Service** **UUID**

**Characteristic**

**R** **UUID**

**Service** **UUID**

**Characteristic 1**

**W** **UUID**

**Characteristic 2**

**R** **N** **UUID**

- All attributes have a type which is 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**  
Service Heart Rate **0x180D**  
Characteristic Heart Rate Measurement **0x2A37**
- Profile can be a **custom** (**proprietary**) profile
  - ↳ **UUID : 128 bits**  
Service P2P **0000FE40-cc7a-482a-984a-7f2ed5b3e58f**  
Characteristic LED **0000FE41-cc7a-482a-984a-7f2ed5b3e58f**

# BLE standard profile vs. proprietary profile

## Standard Heart Rate Profile

**Heart Rate Service** 0x180D

**Heart Rate Measurement  
Characteristic** 0x2A37

**Body sensor Location  
Characteristic** 0x2A38

## Proprietary ST P2P Server Profile

**P2P Service** 0x0000FE40CC7A482A984A7F2ED5B3E58F

**My\_LED\_Char** 0x0000FE41CC7A482A984A7F2ED5B3E58F

**SWITCH\_C** 0x0000FE42CC7A482A984A7F2ED5B3E58F

Define by the **SIG**, define the role, requirements, behavior and the structure of Attribute Table of each entity (central & peripheral)

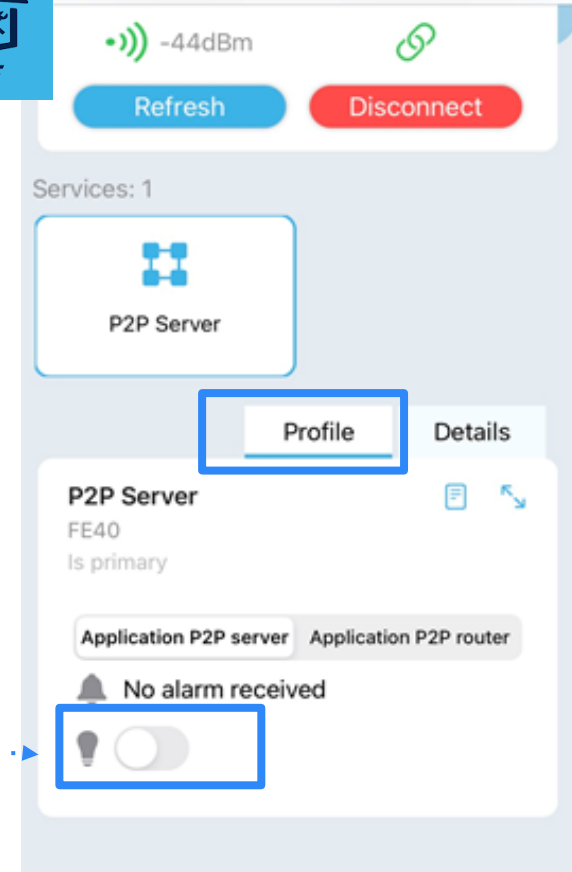
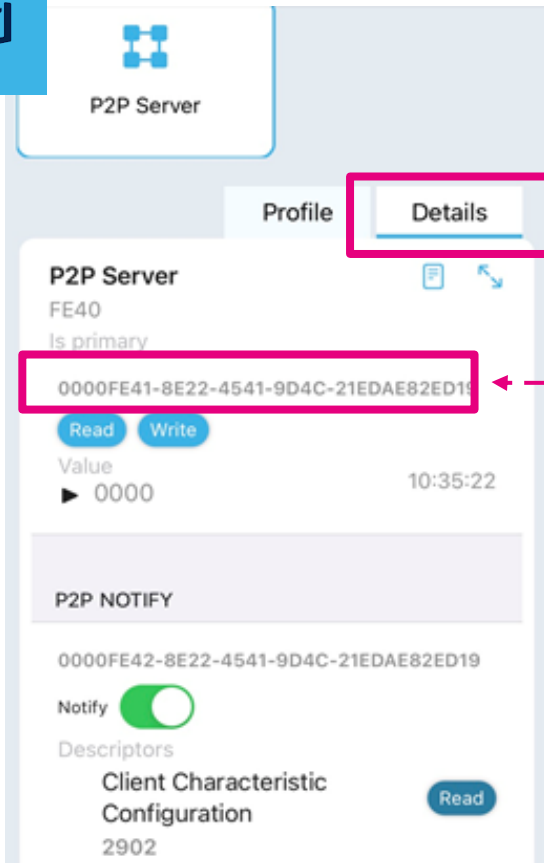
→ Any standard smartphone App will be able to communicate

Define your own behavior using your own Attribute Table based in 128 bits UUID

→  Only your own App will be able to communicate



# Proprietary profile ST Toolbox App

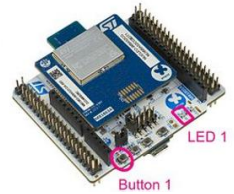


ST ToolBox App knows that **My\_LED\_Char proprietary UUID** is defined to toggle led .  
As consequence App displays nice **toggle button**



# Data exchanges

## what is the magic behind

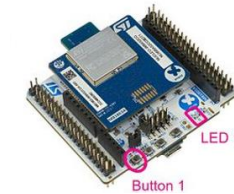


#1 At profile initialization and entry point (**handle @**) will be created in RAM to expose data to client



WBA52 - Attribute Table (RAM)		
@ Serv service UUID	@ Char <b>W R N</b> char UUID	@ Char +1 user data

#2 As soon as connected client will discover server attribute table (**handle**) and will be able to access (**write/read**) data



#3 Application will update data (ie : push button), client will receive **notification** with data updates



@ Serv	@ Char <b>W R N</b>	@ Char +1 update data

BLE write, read, notify procedures using the right attribute handle make data exchange possible

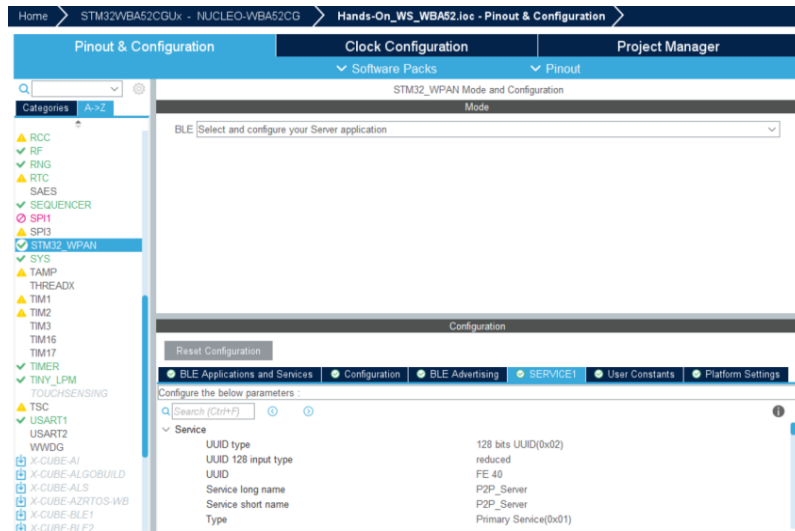
Hands-on #1



You  
succeed



start back from running .ioc



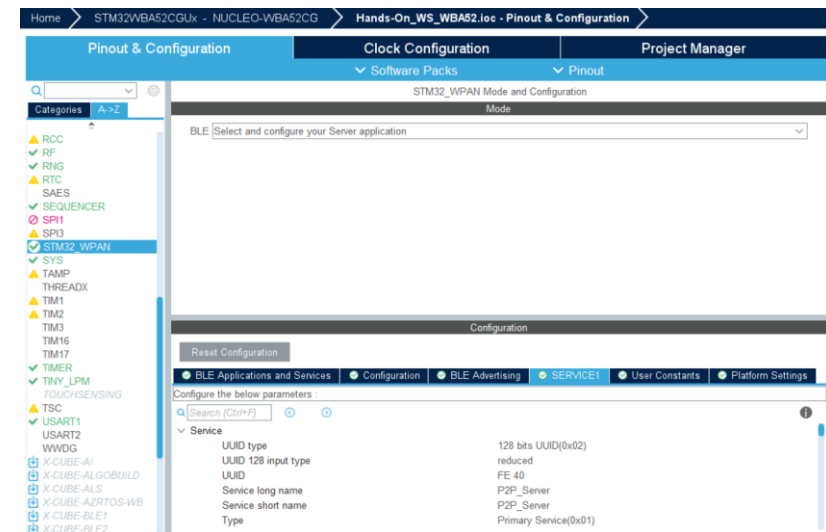
Hands-on #1



More or  
less...



load Hands-On\_WS\_WBA52\_complete.ioc



# Profile Creation Service

Pinout & Configuration

Clock Configuration

Software Packs

Pinout

STM32\_WPAN Mode and Configuration

Mode

BLE Select and configure your Server application

Configuration

Reset Configuration

BLE Advertising

SERVICE1

BLE Applications and Services

User Constants

Platform Settings

Configuration

Configure the below parameters :

Search (Ctrl+F)

1

2

3

Server Mode

Number of services

1

BLE Services Configuration

Peripheral Role

1

Central Role

0

BLE\_CFG\_SVC\_MAX\_NBR\_CB

7

BLE\_CFG\_CLT\_MAX\_NBR\_CB

0

# Profile Creation

## Configure my P2P Service

Configuration

Reset Configuration

BLE Applications and Services Configuration BLE Advertising **SERVICE1** User Constants Platform Settings

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

128 bits UUID(0x02)  
reduced  
FE 40  
P2P\_Server  
P2P\_Server  
Primary Service(0x01)  
5  
2

2

Service Long Name	My_P2P_Server	
Service Short Name	My_P2P	
UUID Type	128 bits	
UUID	0xFE40	
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

UUID : FE 40

The application code will append 112 bits ( based on UUID generator) to have a complete **128 bits UUID**





# Profile Creation

## Configure 1st Characteristic

Configuration

Reset Configuration

BLE Applications and Services Configuration BLE Advertising SERVICE1 User Constants Platform Settings

Configure the below parameters :

Search (Ctrl+F)

> Service

Characteristic1

1

128 bits UUID(0x02)  
reduced  
FE 41  
My\_LED\_Char  
LED\_C  
2  
Variable  
0x10

CHAR\_PROP\_READ Yes  
CHAR\_PROP\_WRITE\_WITHOUT\_RESP Yes  
CHAR\_PROP\_WRITE No  
CHAR\_PROP\_NOTIFY No  
CHAR\_PROP\_INDICATE No  
ATTR\_PERMISSION\_AUTHEN\_READ No  
ATTR\_PERMISSION\_AUTHOR\_READ No  
ATTR\_PERMISSION\_ENCRY\_READ No  
ATTR\_PERMISSION\_AUTHEN\_WRITE No  
ATTR\_PERMISSION\_AUTHOR\_WRITE No  
ATTR\_PERMISSION\_ENCRY\_WRITE No  
GATT\_NOTIFY\_ATTRIBUTE\_WRITE Yes  
GATT\_NOTIFY\_WRITE\_REQ\_AND\_WAIT\_FOR\_APPL\_RESP No  
GATT\_NOTIFY\_READ\_REQ\_AND\_WAIT\_FOR\_APPL\_RESP No

2

> Characteristic2

UUID : FE 41

Application code will complete to have a complete **128 bits UUID**

### 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	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



# Profile Creation

## Configure 2<sup>nd</sup> Characteristic

Configuration

Reset Configuration

BLE Applications and Services Configuration BLE Advertising SERVICE1 User Constants Platform Settings

Configure the below parameters :

Search (Ctrl+F)

< >

i

> Service

> Characteristic1

▼ Characteristic2

UUID type

UUID 128 input type

UUID

Characteristic long name

Characteristic short name

Value length

Length characteristic

Encryption Key Size

CHAR\_PROP\_BROADCAST

CHAR\_PROP\_READ

CHAR\_PROP\_WRITE\_WITHOUT\_RESP

CHAR\_PROP\_WRITE

CHAR\_PROP\_NOTIFY

CHAR\_PROP\_INDICATE

Update char value offset

ATTR\_PERMISSION\_AUTHEN\_READ

ATTR\_PERMISSION\_AUTHOR\_READ

ATTR\_PERMISSION\_ENCRY\_READ

ATTR\_PERMISSION\_AUTHEN\_WRITE

ATTR\_PERMISSION\_AUTHOR\_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

128 bits UUID(0x02)

reduced

FE 42

My\_Switch\_Char

SWITCH\_C

2

Variable

0x10

No

No

No

No

Yes

No

0

No

No

No

No

No

No

Yes

No

No

1

2

UUID : FE 42

Application code will complete to have a complete 128 bits UUID

Properties

Data (2 bytes) as a notify characteristic

Each time user press button over NUCLEO, information sent to client

Permission

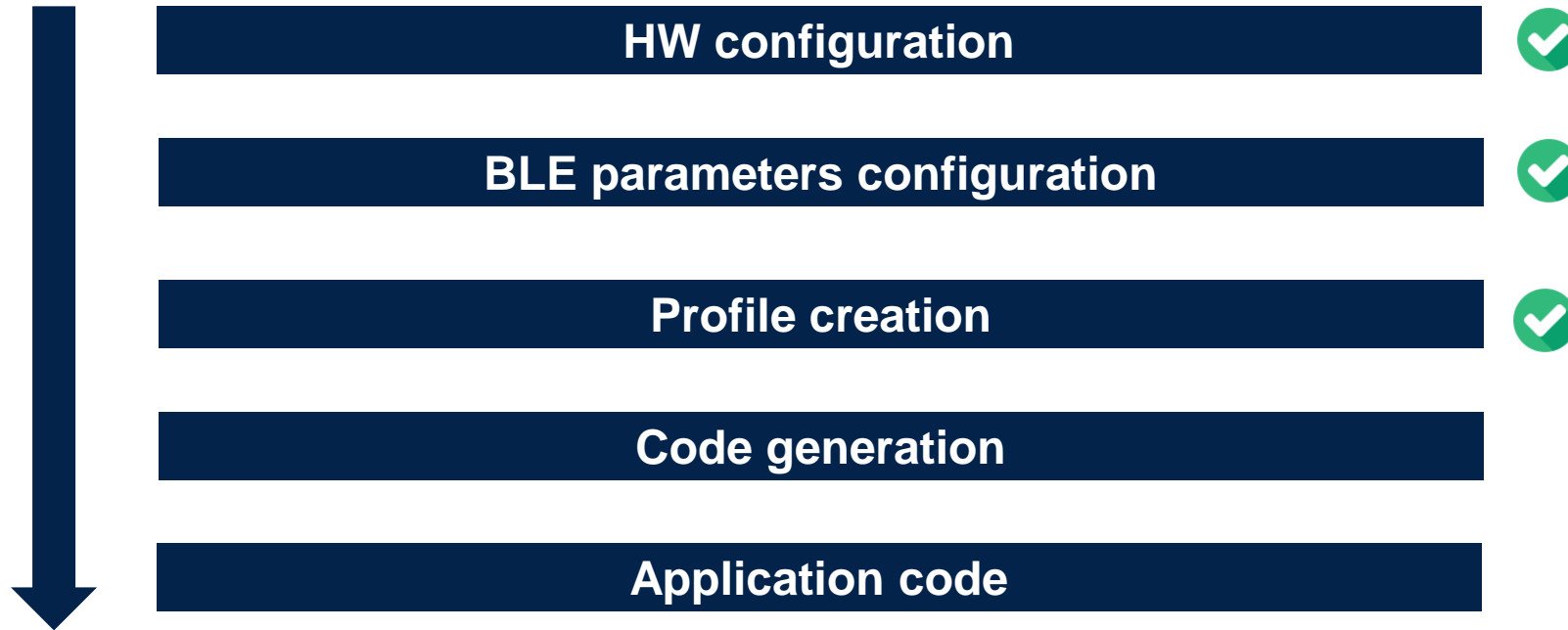
Here permission has not impact. The server is here sending data to client

	Characteristic 1	Characteristic 2
UUID type	128 bits UUID (0x02)	128 bits UUID (0x02)
UUID 128 Input type	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



# Configuration completed

## What's next - Yes code generation



## **Step 2 : Code generation and user application code**

Home
STM32WBA52CGUx - NUCLEO-WBA52CG
Hands-On\_WS\_WBA52.ioc - Project Manager
1
GENERATE CODE

Pinout & Configuration
Clock Configuration
Project Manager
Tools

2
Project
Code Generator
Advanced Settings

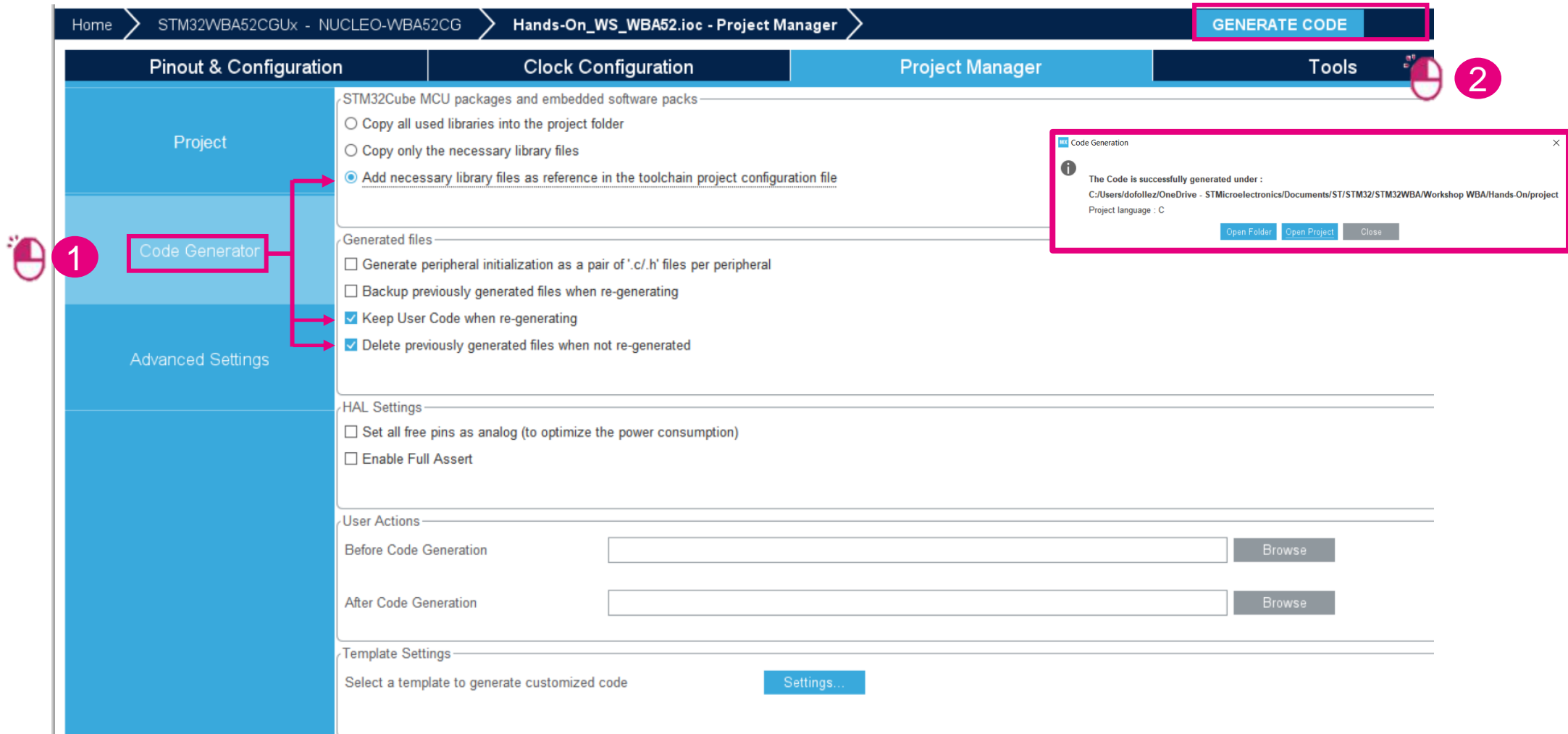
Project Settings
Project Name
Hands-On\_WS\_WBA52
Project Location
OneDrive - STMicroelectronics\Documents\ST\STM32\STM32WBA\Workshop WBA\Hands-On\project\
Browse
Application Structure
Advanced
Do not generate the main()
Toolchain Folder Location
ve - STMicroelectronics\Documents\ST\STM32\STM32WBA\Workshop WBA\Hands-On\project\Hands-On\_WS\_WBA52\
Toolchain / IDE
STM32CubeIDE
Generate Under Root

Linker Settings
Minimum Heap Size
0x200
Linker settings – Heap and CStack
Minimum Stack Size
0x400

Thread-safe Settings
CortexM33
Enable multi-threaded support
Thread-safe Locking Strategy
Default – Mapping suitable strategy depending on RTOS selection.

Mcu and Firmware Package
Mcu Reference
STM32WBA52CGUx
Firmware Package Name and Version
STM32Cube FW\_WBA V1.1.0
Use Default Firmware Location
Use default fw location
Firmware Relative Path
C:/Users/dofollez/STM32Cube/Repository/STM32Cube\_FW\_WBA\_V1.1.0
Browse

# Project configuration



The screenshot displays the STM32CubeMX Project Manager interface for the project "Hands-On\_WS\_WBA52.ioc". The interface is divided into four main tabs: Pinout & Configuration, Clock Configuration, Project Manager, and Tools. The Project Manager tab is currently active.

On the left sidebar, the "Code Generator" option is highlighted with a red box and a red circle labeled "1". The "Advanced Settings" option is also visible below it.

The main content area shows the "Code Generator" settings. Under "STM32Cube MCU packages and embedded software packs", the option "Add necessary library files as reference in the toolchain project configuration file" is selected with a radio button. Under "Generated files", the options "Keep User Code when re-generating" and "Delete previously generated files when not re-generated" are checked. The "HAL Settings" section includes options for "Set all free pins as analog" and "Enable Full Assert". The "User Actions" section has input fields for "Before Code Generation" and "After Code Generation", each with a "Browse" button. The "Template Settings" section has a "Select a template to generate customized code" dropdown and a "Settings..." button.

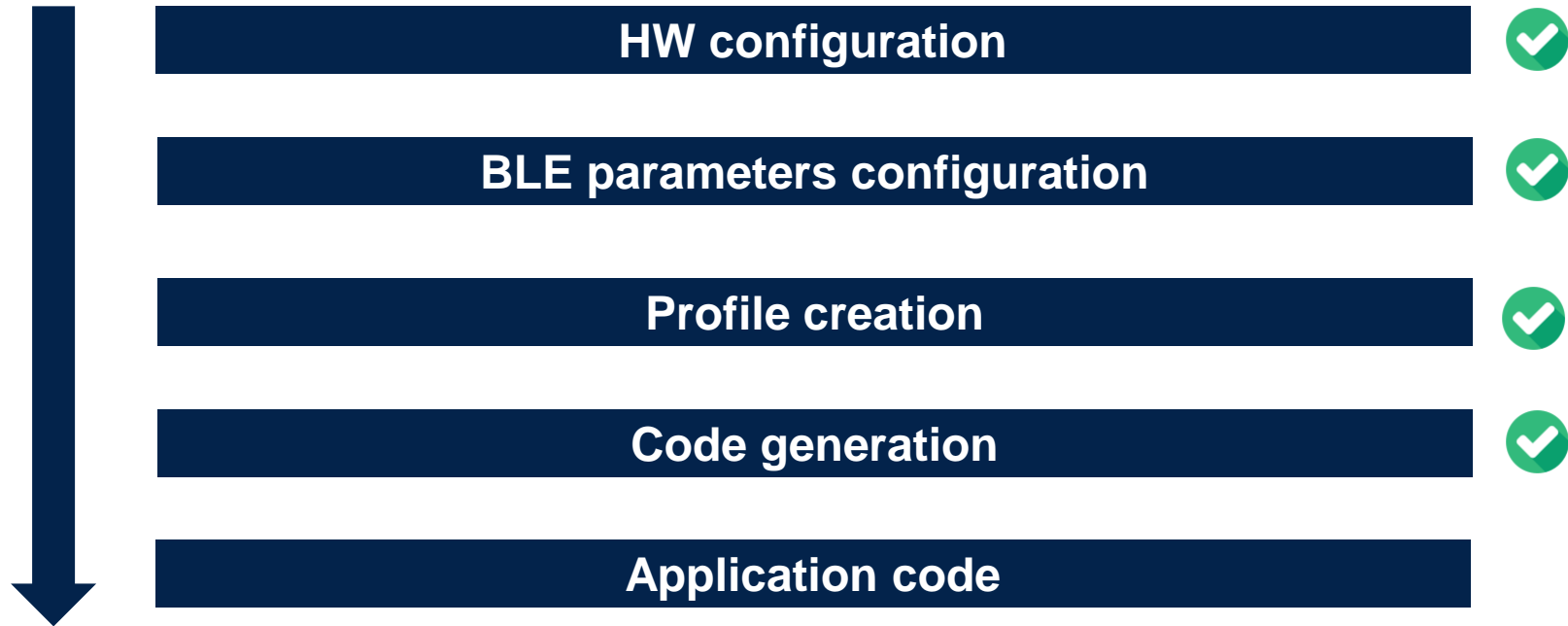
A red box and a red circle labeled "2" highlight the "GENERATE CODE" button in the top right corner of the Project Manager tab.

A "Code Generation" dialog box is open on the right side of the screen, displaying a success message: "The Code is successfully generated under : C:/Users/dofollez/OneDrive - STMicroelectronics/Documents/ST/STM32/STM32WBA/Workshop WBA/Hands-On/project". The dialog also shows the project language as "C" and includes buttons for "Open Folder", "Open Project", and "Close".



# Configuration completed

## What's next - Yes code generation



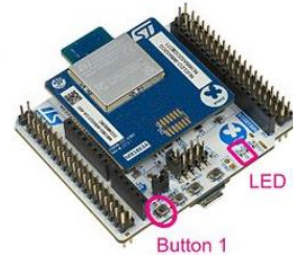




# Add application code Toggle LED from client



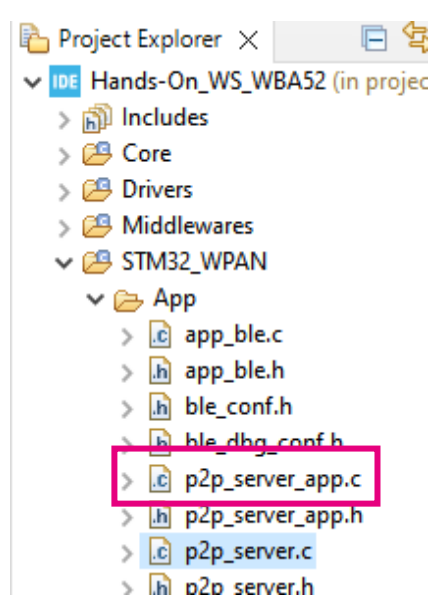
write to My\_LED\_Char (FE 41)



	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   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_EV*/  
HAL_GPIO_TogglePin(GPIOB, LD2_Pin|LD3_Pin|LD1_Pin);  
/* USER CODE END Service1Char1_WRITE_NO_RESP_EV */<
```

# 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_HCI_ASYNC_EVT_ID,
    CFG_TASK_LINK_LAYER,
    CFG_TASK_LINK_LAYER_TEMP_MEAS,
    CFG_TASK_BLE_HOST,
    CFG_TASK_BPKA,
    CFG_TASK_HW_RNG,
    CFG_TASK_AMM_BCKGND,
    CFG_TASK_FLASH_MANAGER_BCKGND,
    CFG_TASK_BLE_TIMER_BCKGND,
    /* 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, APPE_Button1Action);
```

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

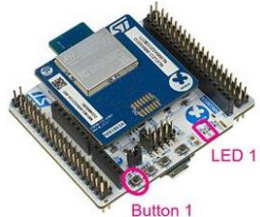
```
UTIL_SEQ_SetTask(1U << TASK_BUTTON_1, CFG_SEQ_PRIO_0);
```

It notify the sequencer that the task must be triggered.  
It will generate a call to registered function  
(here : APPE\_Button1Action() )



# 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

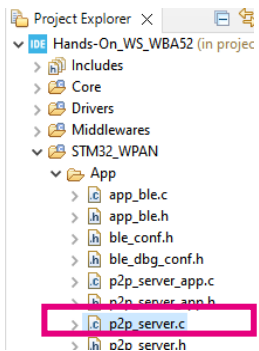
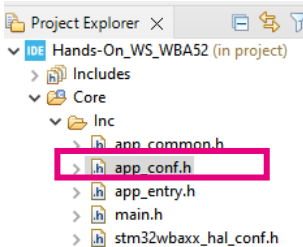
```
/* USER CODE BEGIN CFG_Task_Id_t */  
TASK_BUTTON_1,  
/* USER CODE END CFG_Task_Id_t*/
```

#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 */
```

Function generated by CubeMx as per as Characteristic Short Name

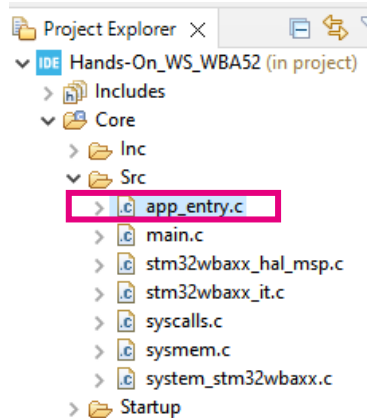


life.augmented

## Raise a notification from device to Smartphone(2/3)



press button



### #3 Manage Button1 interrupt : implement IRQ callback

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

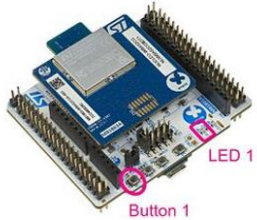
```
/* USER CODE BEGIN FD_WRAP_FUNCTIONS */
void HAL_GPIO_EXTI_Rising_Callback(uint16_t GPIO_Pin)
{
    if (GPIO_Pin == B1_Pin)
    {
        UTIL_SEQ_SetTask(1U << TASK_BUTTON_1, CFG_SEQ_PRIO_0);
    }

    return;
/* USER CODE END FD_WRAP_FUNCTIONS */
```



# Add application code

## Raise a notification from device to Smartphone(3/3)



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

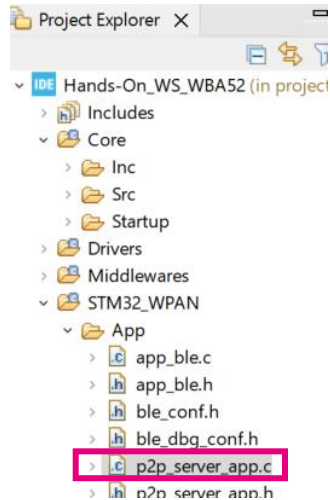
#### #4 Manage BLE notification procedure

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



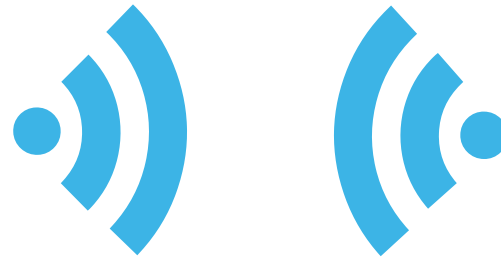
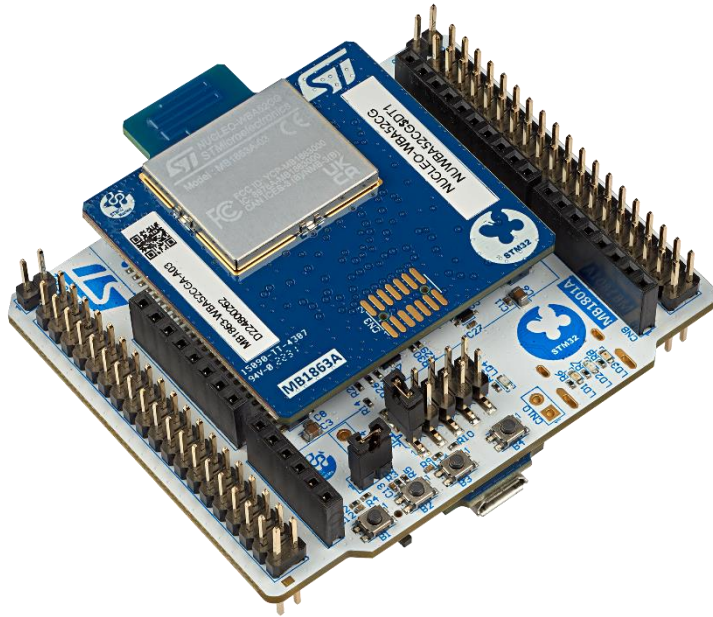
P2P\_SERVER\_UpdateValue

aci\_gatt\_update\_char\_value

BLE stack API



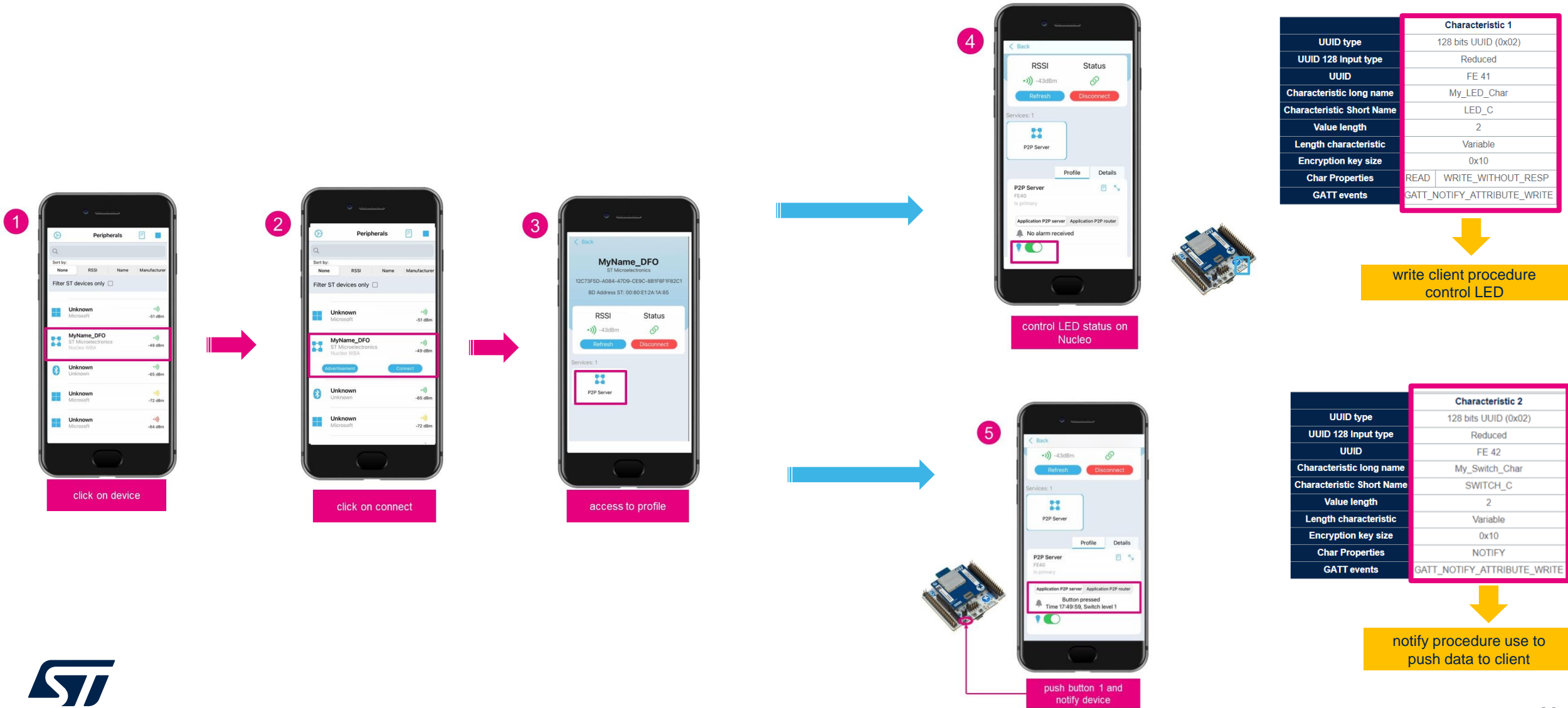
# Open your App and Connect



ST BLE Toolbox



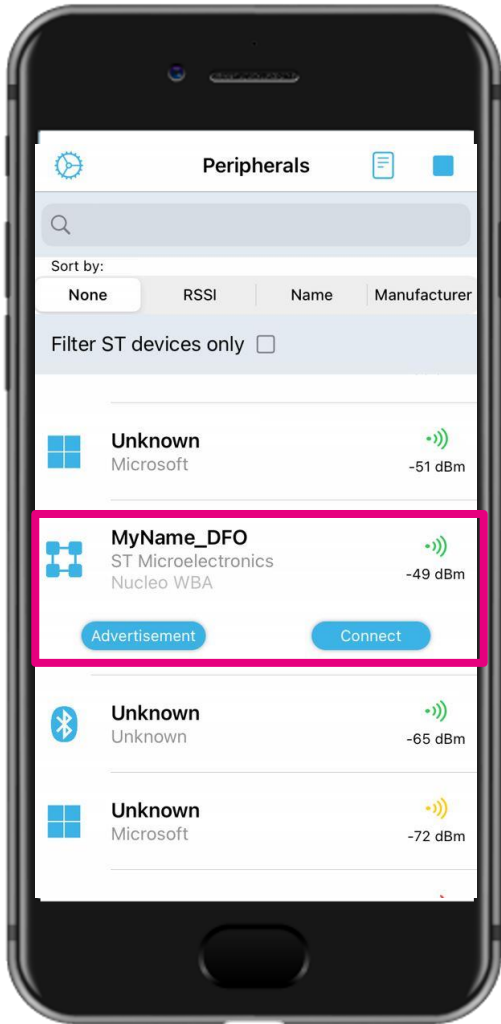
# Open your App and Connect (1/2)



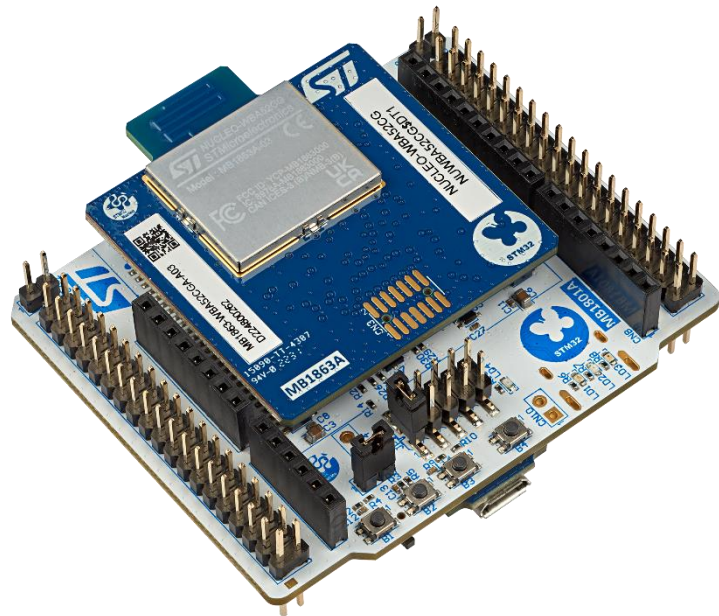
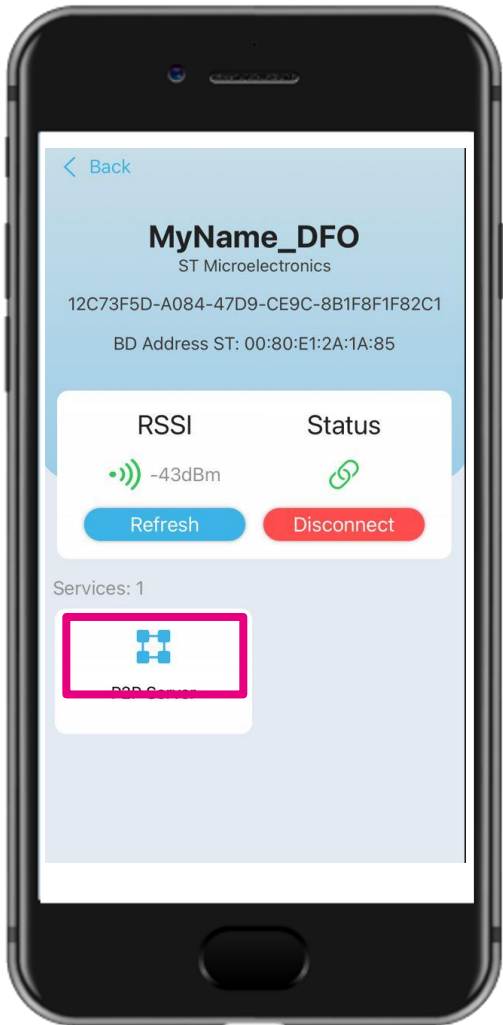


# STBLE Toolbox (Connection)

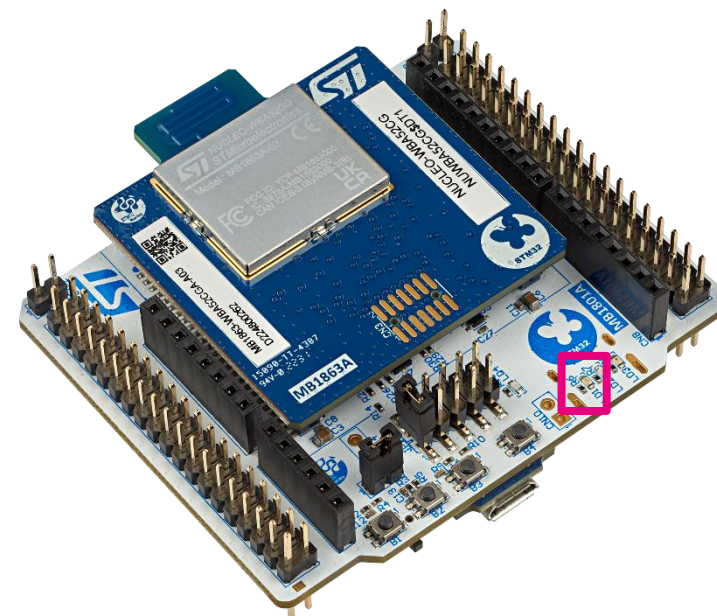
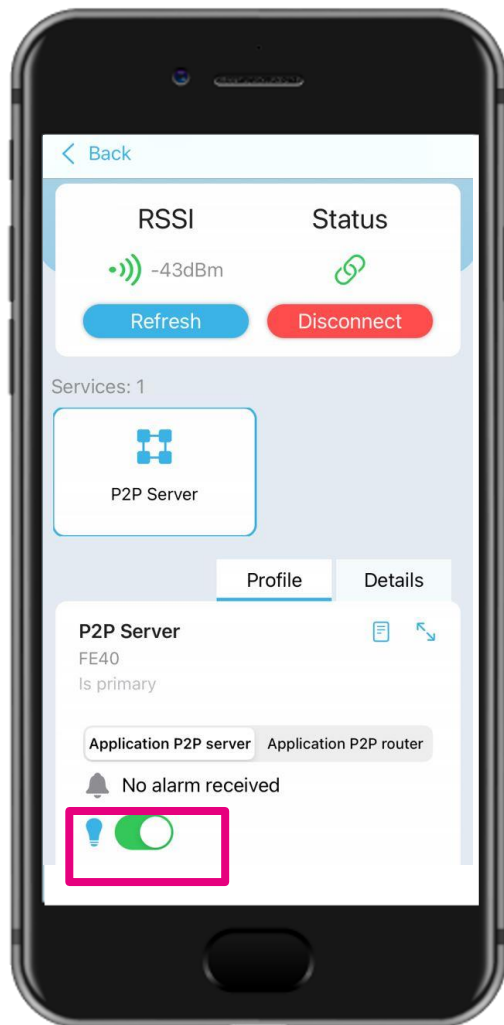
1



2



# STBLE Toolbox (LED)

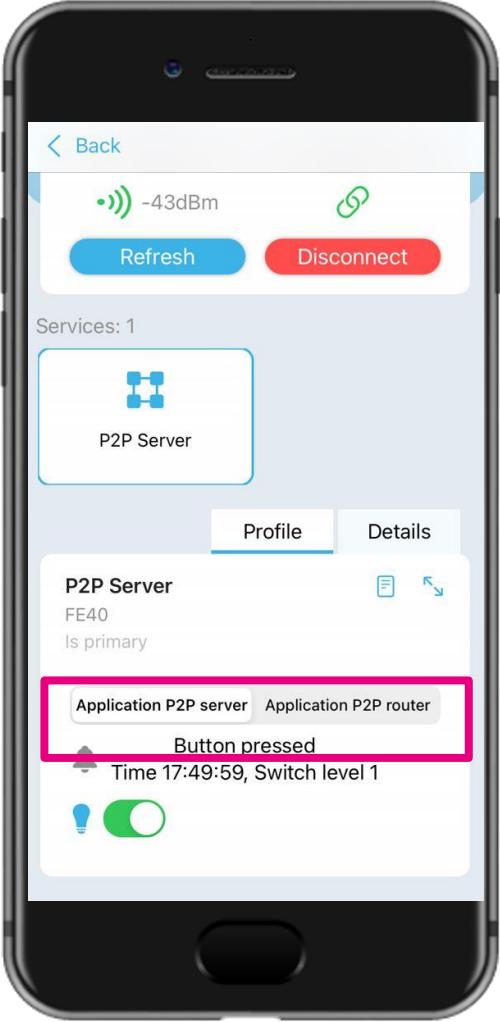


control LED status on  
Nucleo



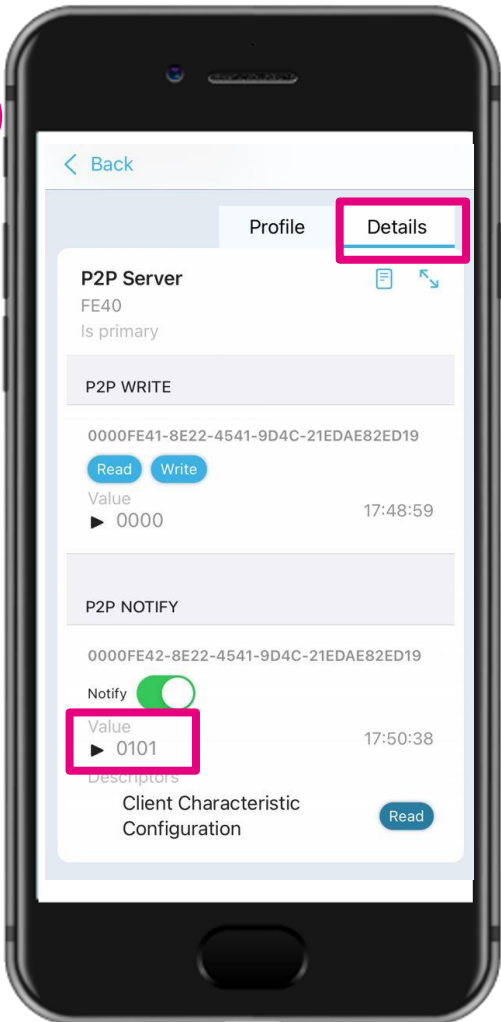
# STBLE Toolbox (Push Button)

1



push button 1 and  
notify device

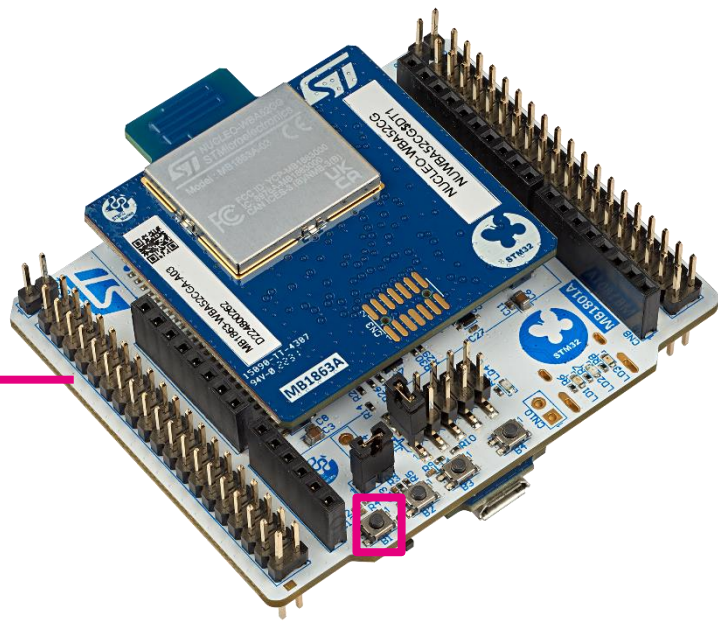
2



click on details to see  
bytes sent/received



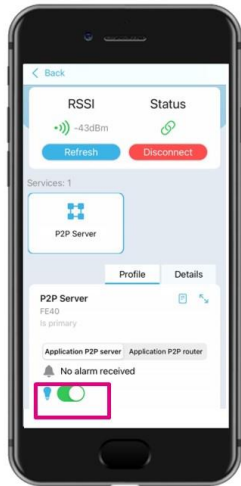
Notification - SWITCH





# Open your App and Connect call stack

4



control LED status on Nucleo

3

2

1

Add break point line 111



BLE write procedure initiated by client

ACI\_GATT\_ATTRIBUTE\_MODIFIED event received at application level

# HW project development & certification

# A complete set of documentation

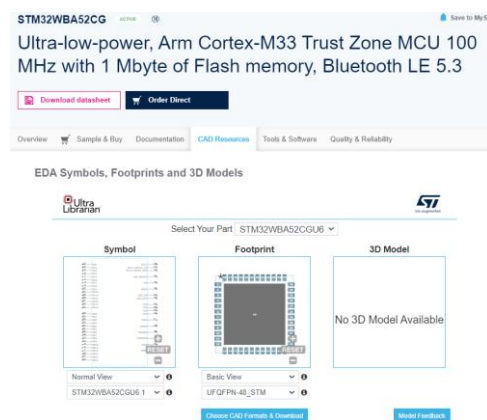
## AN5948 : Development of RF Hardware using STM32WBA



**COMING SOON on st.com!**

- ➡ RF basis generalities
- ➡ Schematics & components selection guidelines
- ➡ STM32WBA5x layout checklist & guidelines

## STM32WBA5x CAD resources on st.com



- ➡ Download STM32WBA5x symbol
- ➡ Download STM32WBA5x footprint









# HW design with STM32WBA5x : key points

- RF matching & filtering:
  - Integrated balun so single ended RF matching.
  - Very limited number of discrete components for STM32WBAx matching and filtering.
  
- Power management. SMPS implementation for STM32WBA55 use case:
  - STM32WBA55 embeds a SMPS that can be used to improve power efficiency.
  
- Main layout recommendation:
  - Refer our various reference kits layout (Gerbers and Altium files available).
  - 4 layers stack-up recommended but 2 layers is possible.

# STM32WBA5x : a certified solution

- STM32WBA5x is compliant in regards of regional (CE, FCC etc.) and Bluetooth requirements.
- We are providing complete set of documentation, FW and tools to certify your product.



## Full set of tools and documentation

- Certification guideline on wiki ([Certification guideline with STM32WB and STM32WBA - stm32mcu](#))
- Transparent mode FW available in [STM32CubeWBA](#) MCU Package
- [STM32CubeMonRF](#) PC tool

# Bluetooth certification

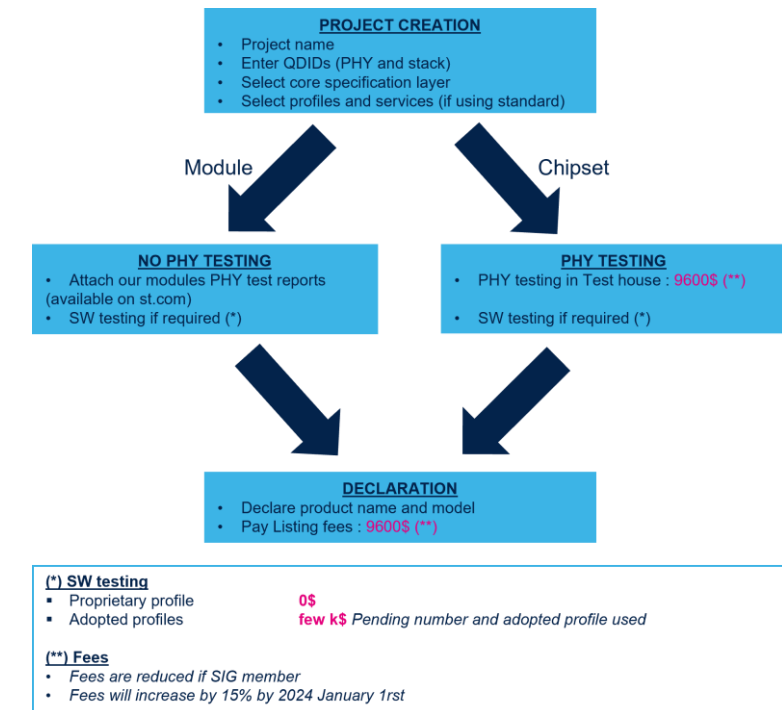
- STM32WBA5x is having reference QDIDs (components and stack) at Bluetooth SIG.
- Customer musty performed PHY testing and declare its product at Bluetooth SIG.

## PHY QDID

Package	Part number	Cut version	RF PHY QDID
QFN48	STM32WBA52 (BLE5.4)	1.x	<b>197135</b> (TCRL 2022-2)

## Stack QDID

Features	Host Stack version	QDID
4.0 HCI Low Energy LL with extended advertising – ATT – GAP – GATT – L2CAP with Enhanced Connected Oriented Channel -SMP BLE 5.3	STM32Cube_WBA_BLE_HCI_STACK STM32Cube_WBA_BLE_FULL_STACK	<b>198195</b> (TCRL 2022-1)



Refer wiki [Certification guideline](https://www.st.com/en/development-tools/certification-guideline.html) on st.com

# Thank you