Welcome to
STM32WBA55 workshop

Hands-on #2
Build basic p2pServer application and connect

Workshop team

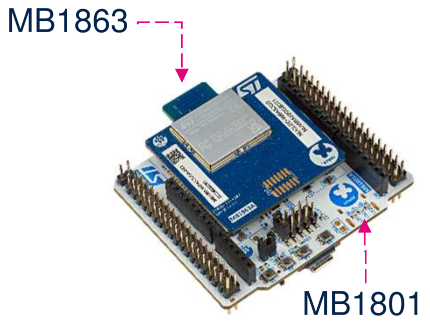
Prerequisites Refresh

SW prerequisites

- STM32CubeWBA MCU package v1.2.0
- IDE: STM32CubeIDE 1.14.0
- A serial terminal (e.g. TeraTerm)
- ST BLE ToolBox Smartphone application
- Dedicated “cheat sheet”

HW prerequisites

- NUCLEO-WBA55
- USB A to Micro-B Cable



ST BLE Toolbox

Agenda

1 Hands-on Presentation

2 Step 1 : Profile creation demystification and details

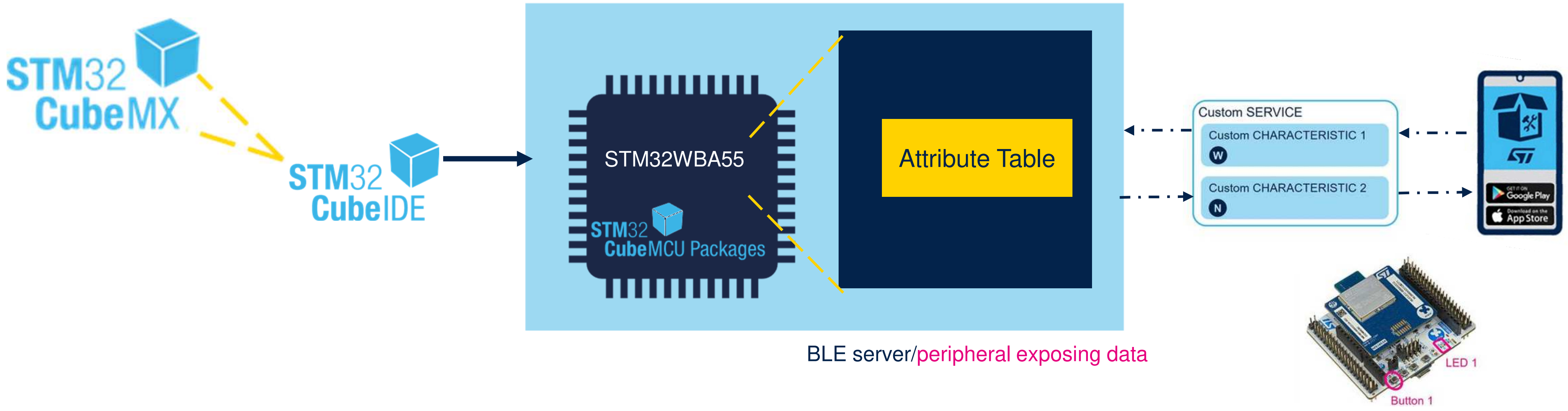
3 Step 2 : Application code

life.augmented

Hands-on presentation

Purpose

- The purpose is to start from WBA55 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



BLE server/peripheral exposing data

Enhance application code to enable a Bluetooth® Low Energy Application Profile (p2pServer)

Legenda

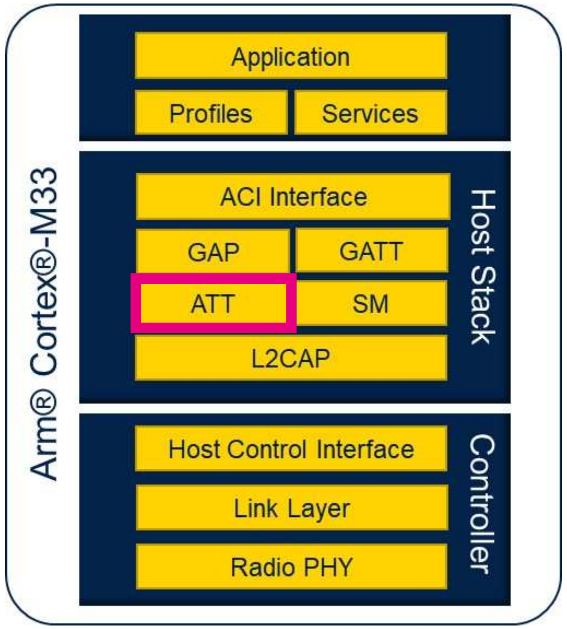
- Slides including following symbol are purely theoretical ones
- Source code for development is included inside blue boxes

```
HAL_Delay(500);
```

Step1 : GAP/GATT custom application configuration : Profile creation

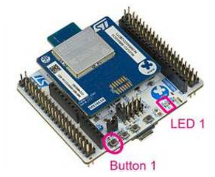
life.augmented

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



Define Client - Server architecture

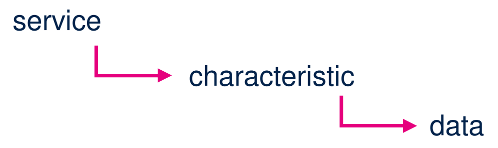
« I am looking for data »
Client

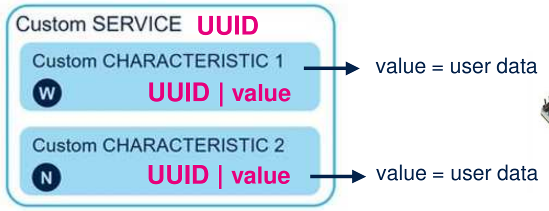


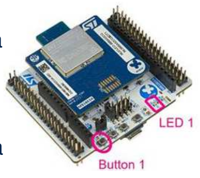
« I have data to share» (sensors, raw data...**what you want** !)
Server

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

user data is accessible trough attribute

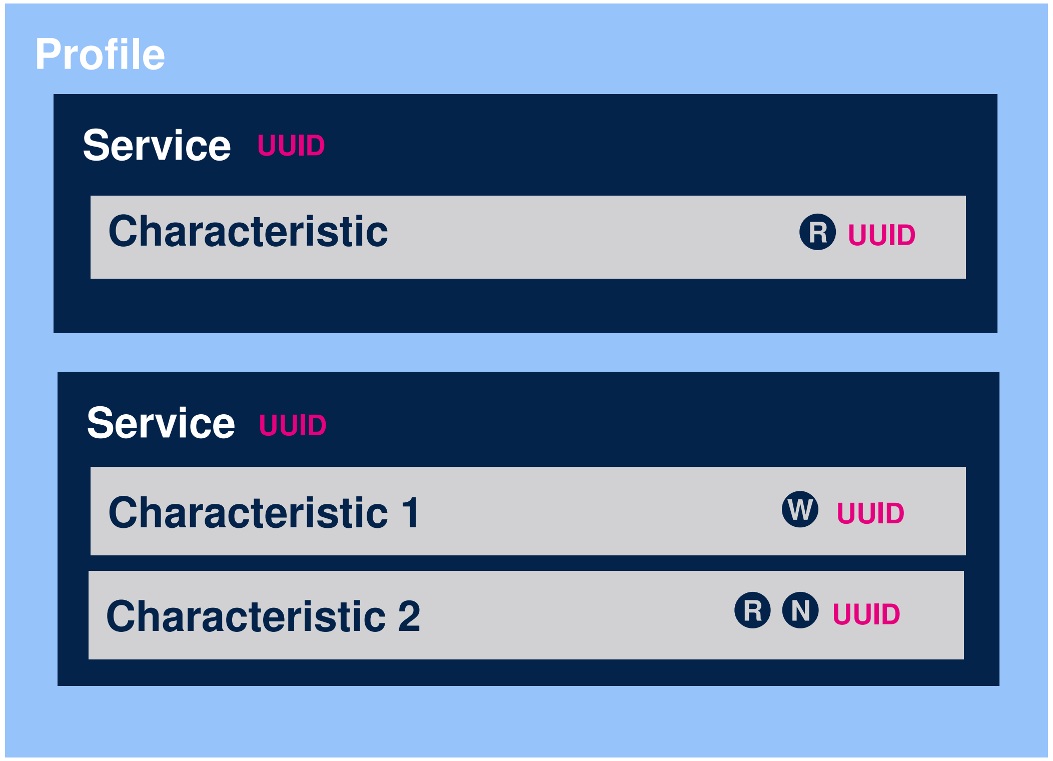






What is a Bluetooth Low Energy Profile ?

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



- All attributes have a type which is identified by a UUID (**U**niversally **U**nique **Id**entifier)
- Characteristic can take 3 types of propreties: **R**EAD, **W**RITE, **N**OTIFY
- 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

Bluetooth® Low Energy standard profile vs. proprietary profile

Standard Heart Rate Profile

Heart Rate Service 0x180D

- **Heart Rate Measurement Characteristic** 0x2A37
- **Body sensor Location Characteristic** 0x2A38

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

Proprietary ST P2PServer Profile

P2P Service 0x0000FE40CC7A482A984A7F2ED5B3E58F

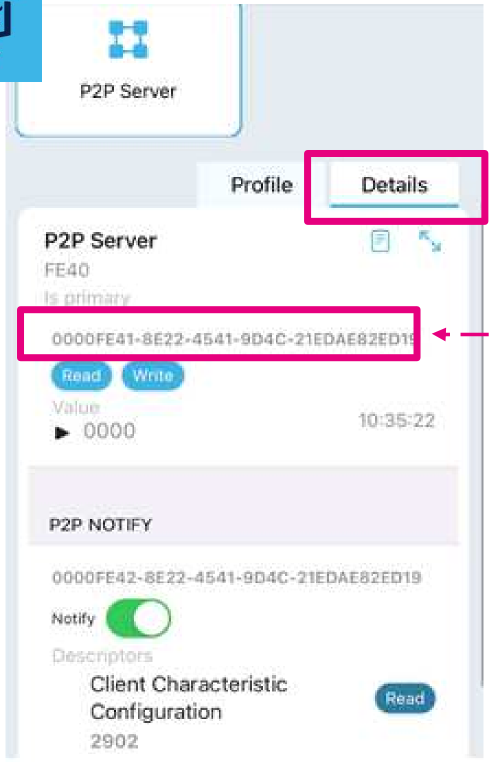
- **My_LED_Char** 0x0000FE41CC7A482A984A7F2ED5B3E58F
- **SWITCH_C** 0x0000FE42CC7A482A984A7F2ED5B3E58F

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

→ Only your own App will be able to communicate

life.augmented

Proprietary profile ST Toolbox App



P2P Server

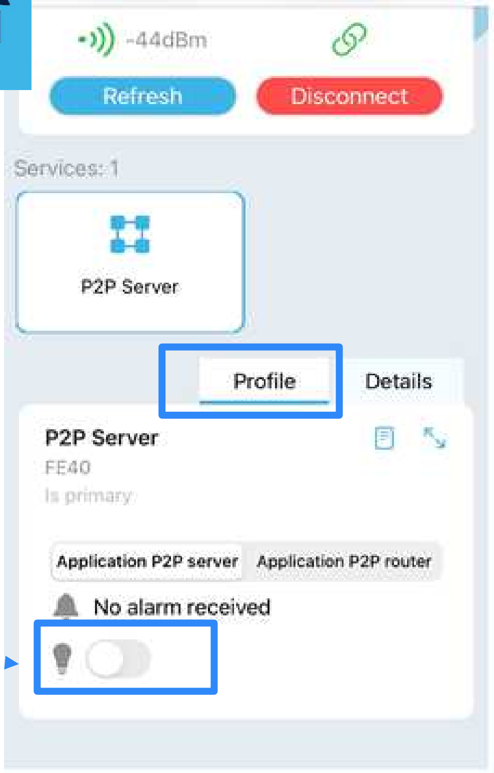
Profile | Details

P2P Server
FE40
Is primary

0000FE41-8E22-4541-9D4C-21EDAE82ED19
Read Write
Value
▶ 0000 10:35:22

P2P NOTIFY

0000FE42-8E22-4541-9D4C-21EDAE82ED19
Notify
Descriptors
Client Characteristic Configuration
2902
Read



-44dBm
Refresh Disconnect

Services: 1

P2P Server

Profile | Details

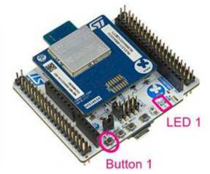
P2P Server
FE40
Is primary

Application P2P server | Application P2P router

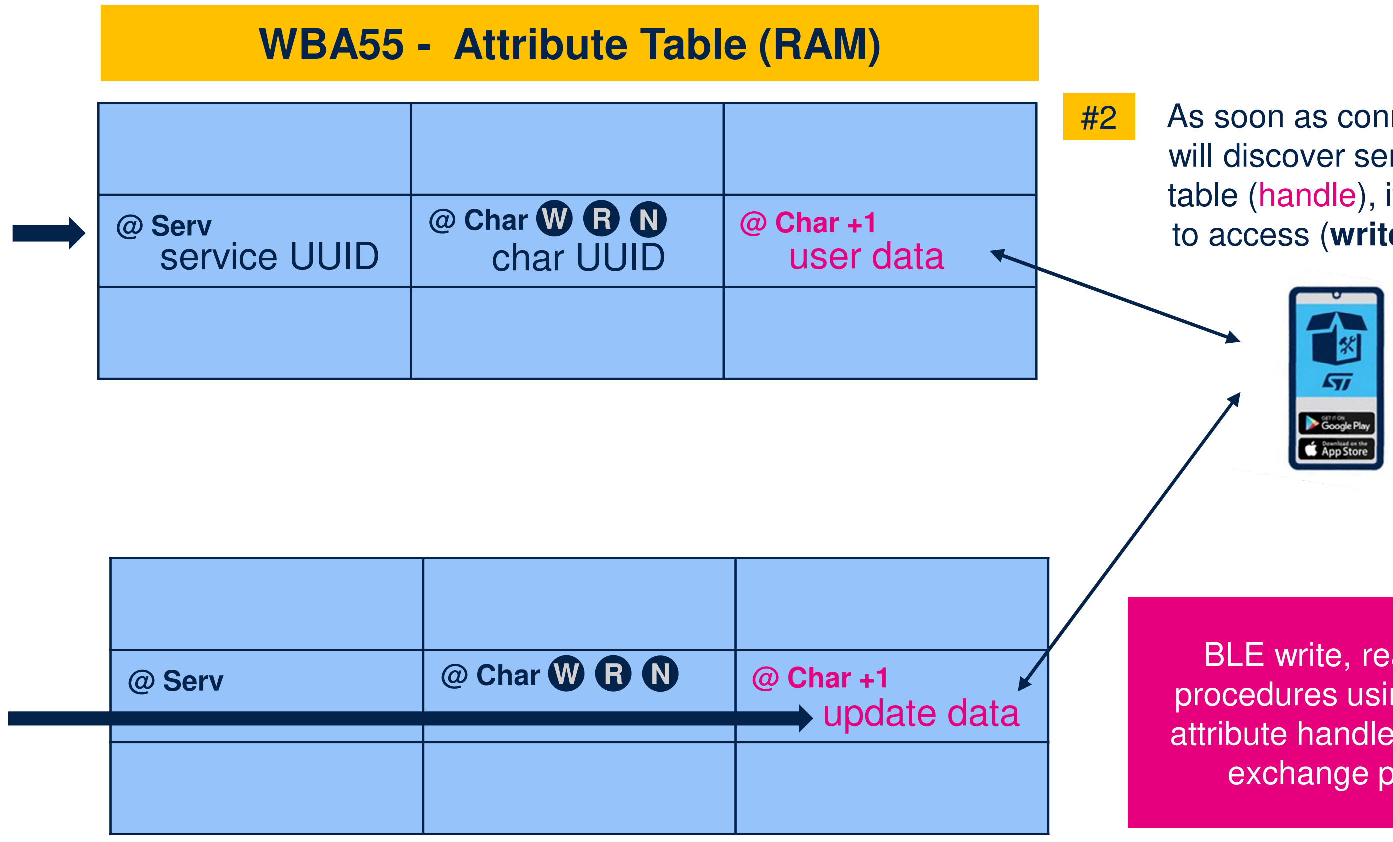
No alarm received

ST ToolBox App knows that My_LED_Char proprietary UUID is defined to toggle led .
As consequence App displays nice **toggle button**

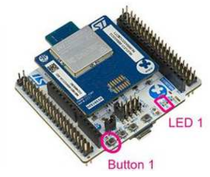
Data exchanges
what is the magic behind ?



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



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



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

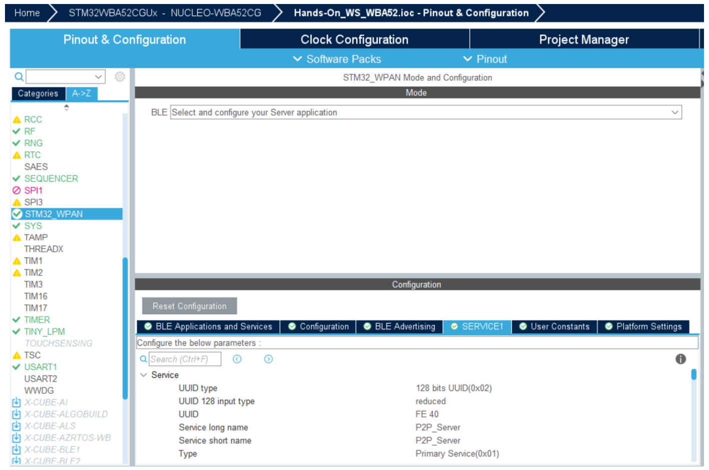
BLE write, read, notify procedures using the right attribute handle make data exchange possible

Profile Creation

Hands-on #1

You succeed

start back from running .ioc



Home > STM32WBA52CGUx - NUCLEO-WBA52CG > **Hands-On_WS_WBA52.ioc - Pinout & Configuration**

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

STM32_WPAN Mode and Configuration

Mode

BLE: Select and configure your Server application

Configuration

Reset Configuration

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

Configure the below parameters :

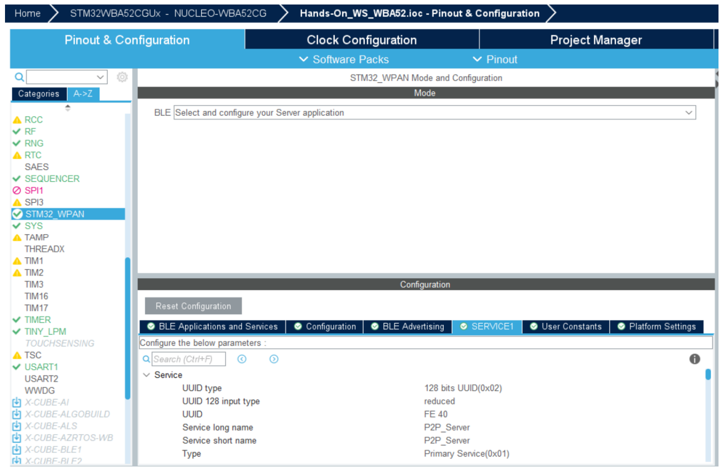
Service

UUID type	128 bits UUID(0x02)
UUID 128 input type	reduced
UUID	FE 40
Service long name	P2P_Server
Service short name	P2P_Server
Type	Primary Service(0x01)

Hands-on #1

More or less…

load Hands_On_WBA55_backup.ioc



Home > STM32WBA52CGUx - NUCLEO-WBA52CG > **Hands-On_WS_WBA52.ioc - Pinout & Configuration**

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

STM32_WPAN Mode and Configuration

Mode

BLE: Select and configure your Server application

Configuration

Reset Configuration

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

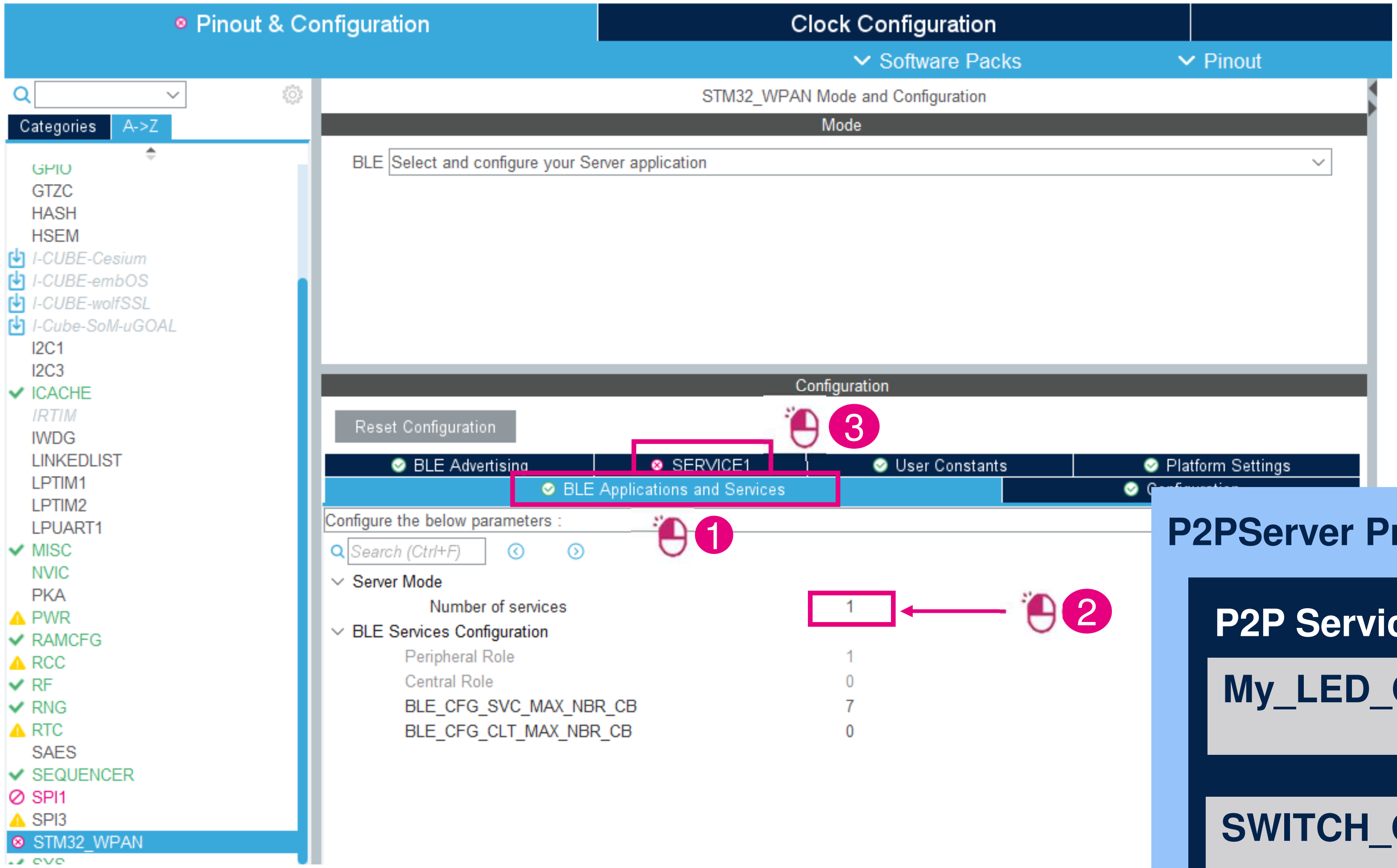
Configure the below parameters :

Service

UUID type: 128 bits UUID(0x02)
UUID 128 input type: reduced
UUID: FE 40
Service long name: P2P_Server
Service short name: P2P_Server
Type: Primary Service(0x01)

life.augmented

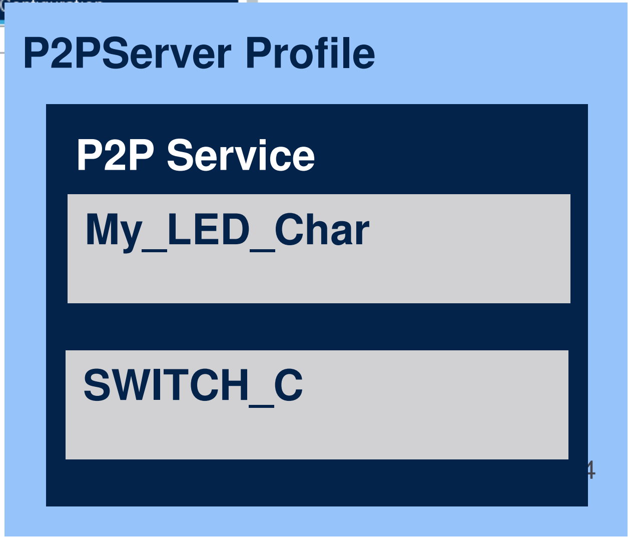
Profile Creation Service



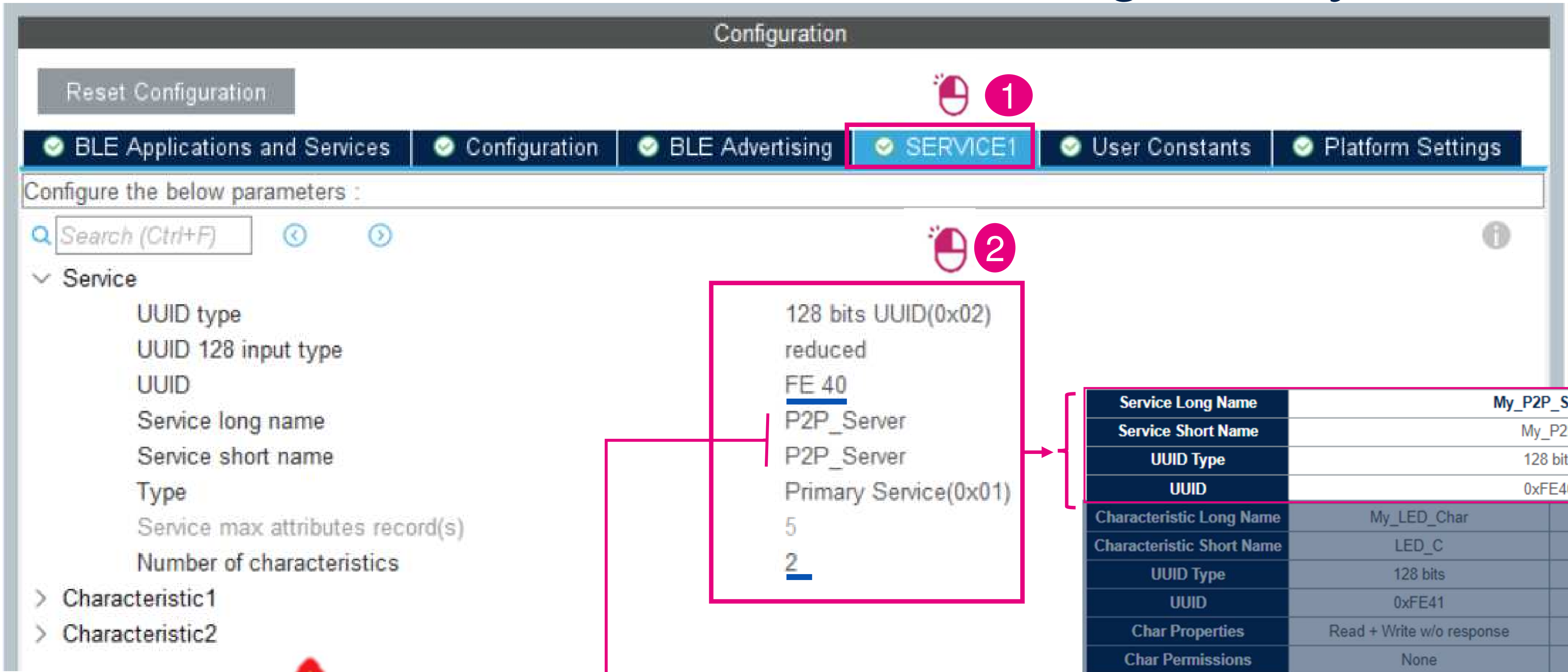
BLE Select and configure your Server application

Configure the below parameters :

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



Service

UUID type	128 bits UUID(0x02)
UUID 128 input type	reduced
UUID	FE 40
Service long name	P2P_Server
Service short name	P2P_Server
Type	Primary Service(0x01)
Service max attributes record(s)	5
Number of characteristics	2

Characteristic1

Characteristic2

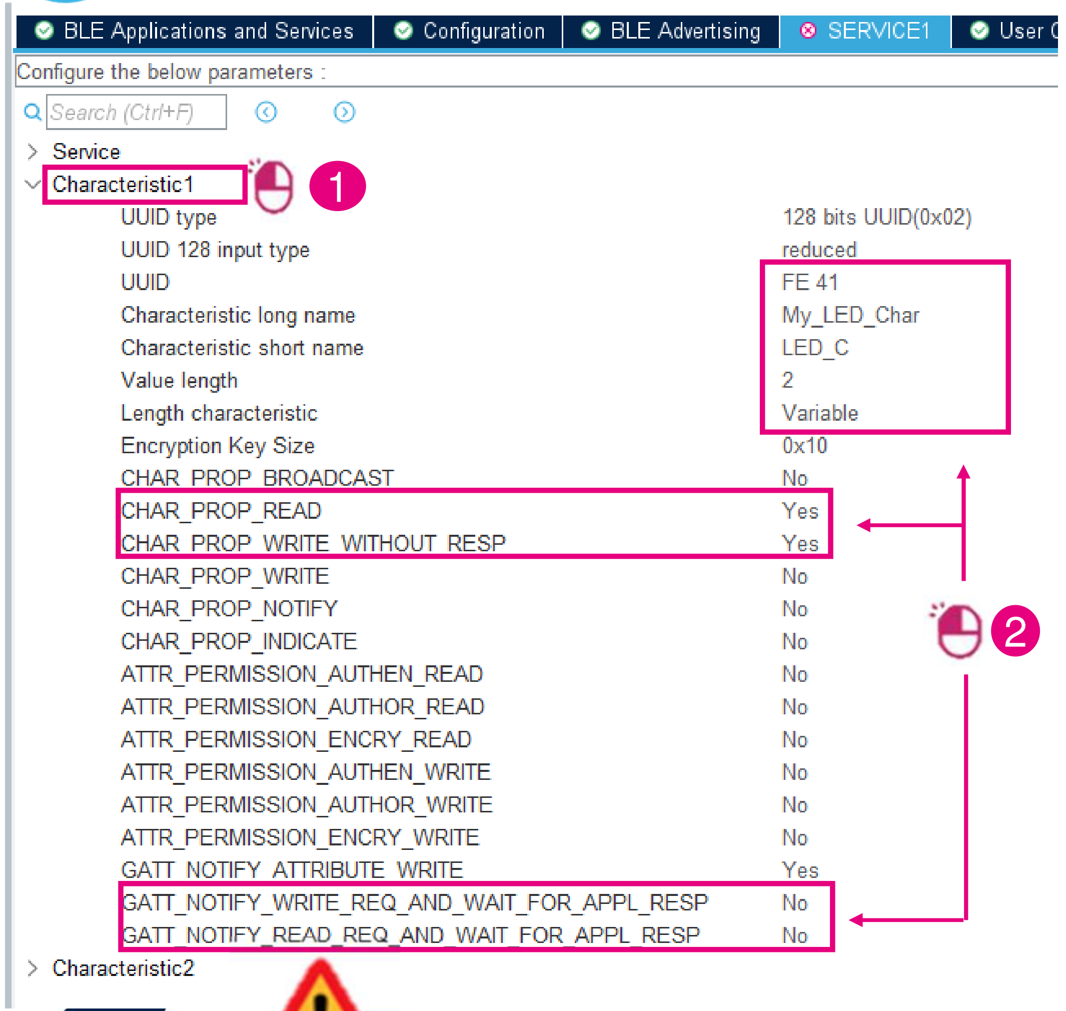
Service Long Name	My_P2P_Server	
Service Short Name	My_P2P	
UUID Type	128 bits	
UUID	0xFE40	
Characteristic Long Name	My_LED_Char	My_Switch_Char
Characteristic Short Name	LED_C	SWITCH_C
UUID Type	128 bits	128 bits
UUID	0xFE41	0xFE42
Char Properties	Read + Write w/o response	Notify
Char Permissions	None	None
Char GATT Events	GATT_NOTIFY_ATTRIBUTE_WRITE	GATT_NOTIFY_ATTRIBUTE_WRITE

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

UUID : **FE 40**
The application code will append 112 bits (based on UUID generator) to have a complete **128 bits UUID**

life.augmented

Profile Creation
Configure 1st Characteristic



Configure the below parameters :

Service

Characteristic1

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_PROP_BROADCAST	No
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

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

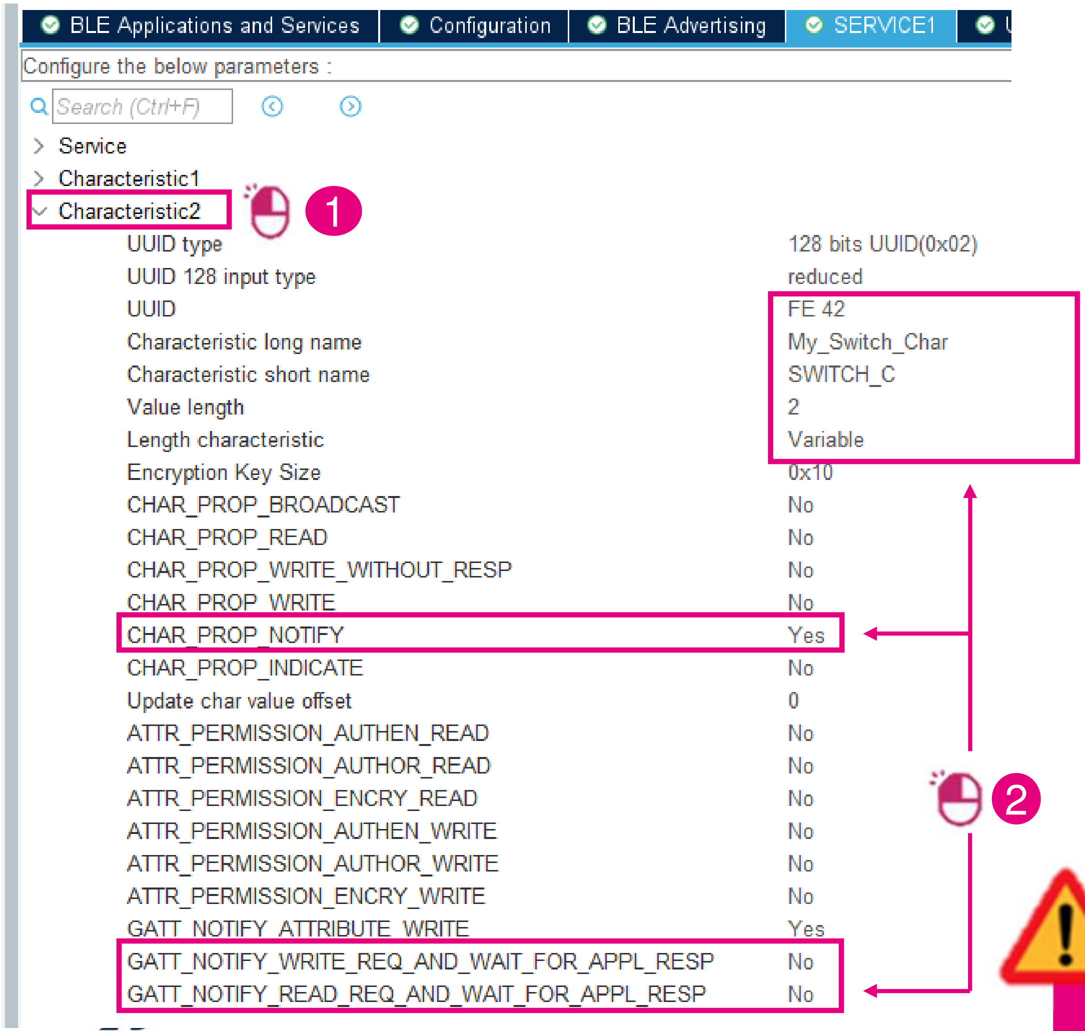
Permission

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

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

Profile Creation
Configure 2nd Characteristic



BLE Applications and Services | Configuration | BLE Advertising | SERVICE1

Configure the below parameters :

Search (Ctrl+F)

- Service
- Characteristic1
- Characteristic2 1

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_PROP_BROADCAST	No
CHAR_PROP_READ	No
CHAR_PROP_WRITE_WITHOUT_RESP	No
CHAR_PROP_WRITE	No
CHAR_PROP_NOTIFY	Yes
CHAR_PROP_INDICATE	No
Update char value offset	0
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

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

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

Configuration completed
What’s next - Yes code generation

- HW configuration ✓
- BLE parameters configuration ✓
- Profile creation ✓
- Code generation
- Application code

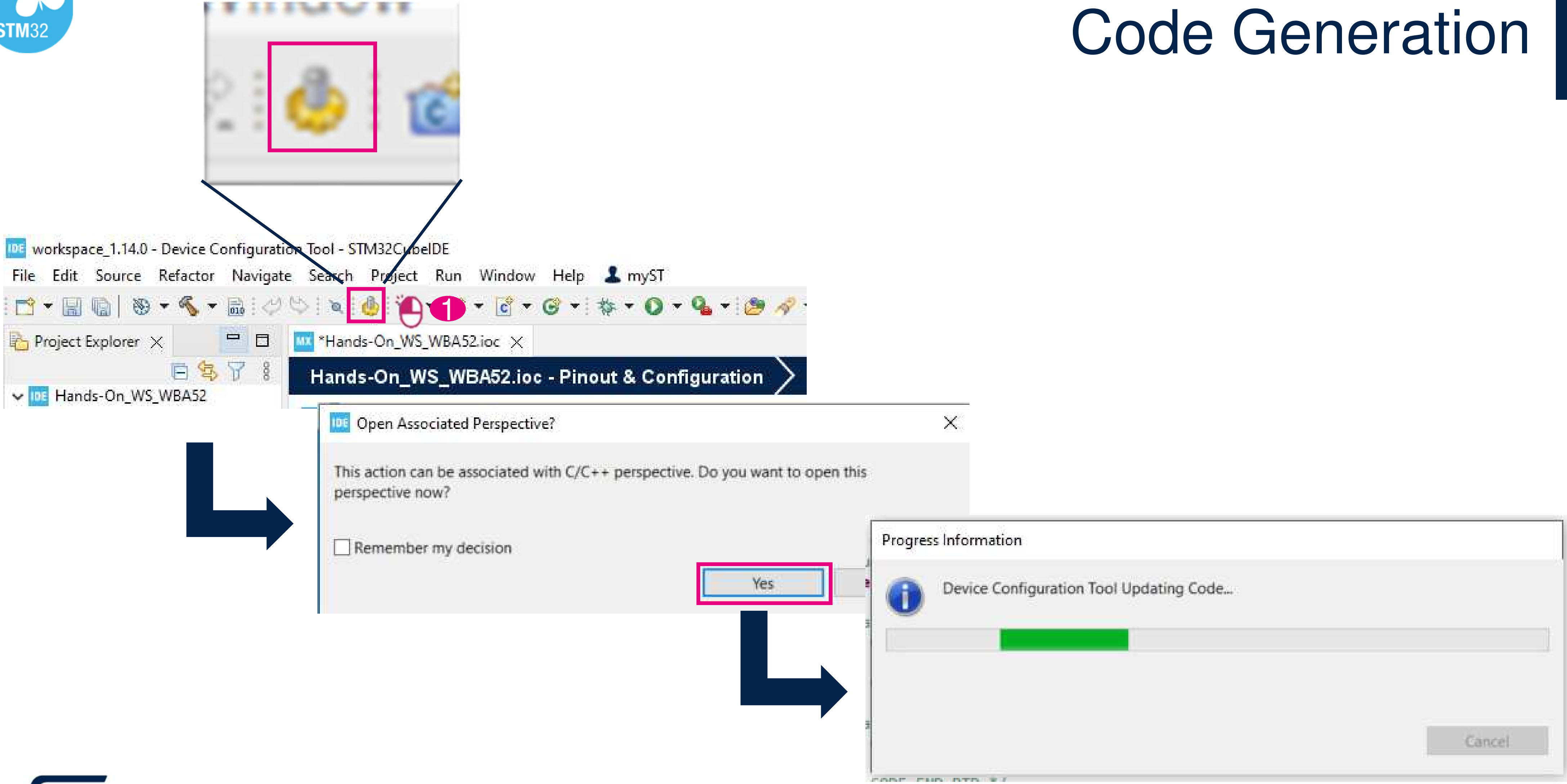
life.augmented

Step 2 : Code generation and user application code

life.augmented

A0

Code Generation


workspace_1.14.0 - Device Configuration Tool - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help myST
1
Project Explorer
Hands-On_WS_WBA52
*Hands-On_WS_WBA52.ioc
Hands-On_WS_WBA52.ioc - Pinout & Configuration

Open Associated Perspective?

This action can be associated with C/C++ perspective. Do you want to open this perspective now?

Remember my decision

Yes

Progress Information

Device Configuration Tool Updating Code...

Cancel

Slide 20

A0 Put a small comment also on the first two tabs

Author; 2023-08-03T11:59:22.255

Configuration completed
What’s next - Yes code generation

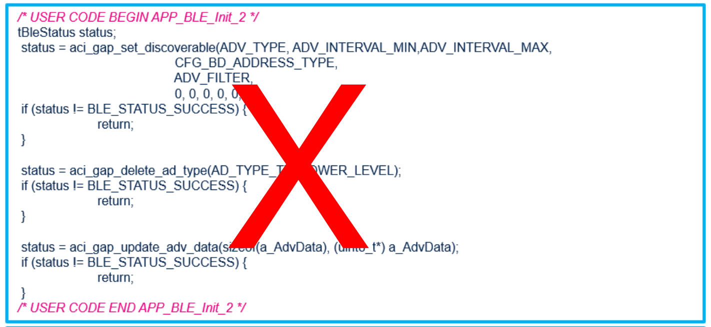
- HW configuration ✓
- BLE parameters configuration ✓
- Profile creation ✓
- Code generation ✓
- Application code

Remove previous code

Why should I remove previous <u>functional</u> code ?

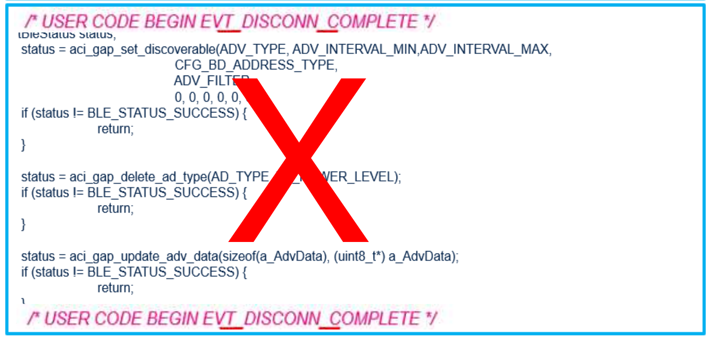
As we have created profile, STM32CubeMX generated new skeleton code with more friendly APIs

Let’s use this API to move to discoverable !



```
/* USER CODE BEGIN APP_BLE_Init_2 */
tBleStatus status;
 status = aci_gap_set_discoverable(ADV_TYPE, ADV_INTERVAL_MIN,ADV_INTERVAL_MAX,
                                   CFG_BD_ADDRESS_TYPE,
                                   ADV_FILTER,
                                   0, 0, 0, 0, 0[illegible]
 if (status != BLE_STATUS_SUCCESS) {
                 return;
 }

 status = aci_gap_delete_ad_type(AD_TYPE_T[illegible]WER_LEVEL);
 if (status != BLE_STATUS_SUCCESS) {
                 return;
 }

 status = aci_gap_update_adv_data(si[illegible](a_AdvData), ([illegible]_t*) a_AdvData);
 if (status != BLE_STATUS_SUCCESS) {
                 return;
 }
/* USER CODE END APP_BLE_Init_2 */
```

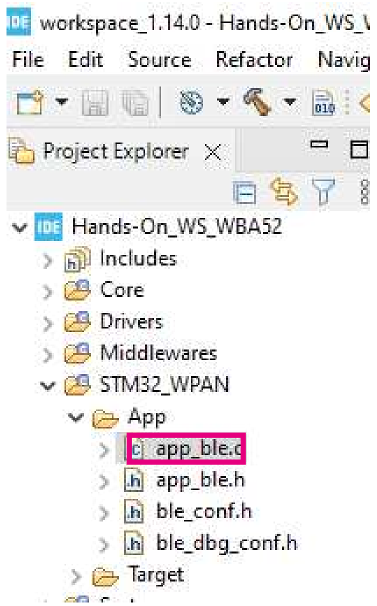
Search for “APP_BLE_Init_2”



```
 /* USER CODE BEGIN EVT_DISCONN_COMPLETE */
tBleStatus status;
 status = aci_gap_set_discoverable(ADV_TYPE, ADV_INTERVAL_MIN,ADV_INTERVAL_MAX,
                                   CFG_BD_ADDRESS_TYPE,
                                   ADV_FILT[illegible]
                                   0, 0, 0, 0, [illegible]
 if (status != BLE_STATUS_SUCCESS) {
                 return;
 }

 status = aci_gap_delete_ad_type(AD_TYPE[illegible]WER_LEVEL);
 if (status != BLE_STATUS_SUCCESS) {
                 return;
 }

 status = aci_gap_update_adv_data(sizeof(a_AdvData), (uint8_t*) a_AdvData);
 if (status != BLE_STATUS_SUCCESS) {
                 return;
 }
 /* USER CODE BEGIN EVT_DISCONN_COMPLETE */
```

Search for “EVT_DISCONN_COMPLETE ”

Add application code to move to discoverable



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 */
```

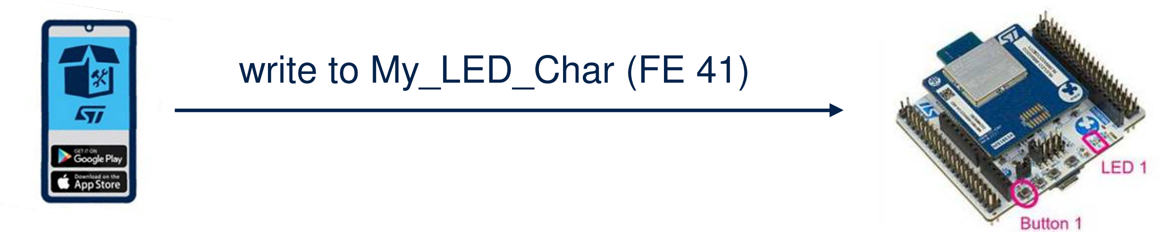
Search for "EVT_DISCONN_COMPLETE "

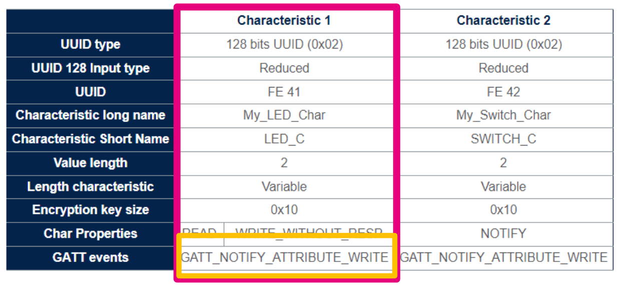
At disconnection, stack is not moving back to advertising, this is an application decision

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

Please refer to cheatsheet for copy/paste

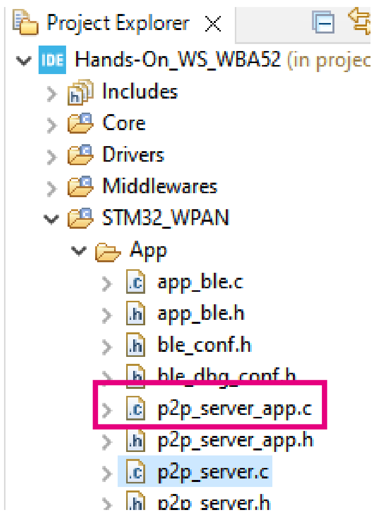
Add application code
Toggle LED from client





	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



```
Project Explorer
Hands-On_WS_WBA52 (in projec
  Includes
  Core
  Drivers
  Middlewares
  STM32_WPAN
    App
      app_ble.c
      app_ble.h
      ble_conf.h
      ble_dbg_conf.h
      p2p_server_app.c
      p2p_server_app.h
      p2p_server.c
      p2p_server.h
```

in p2p_server_app.c / function P2P_SERVER_Notification()

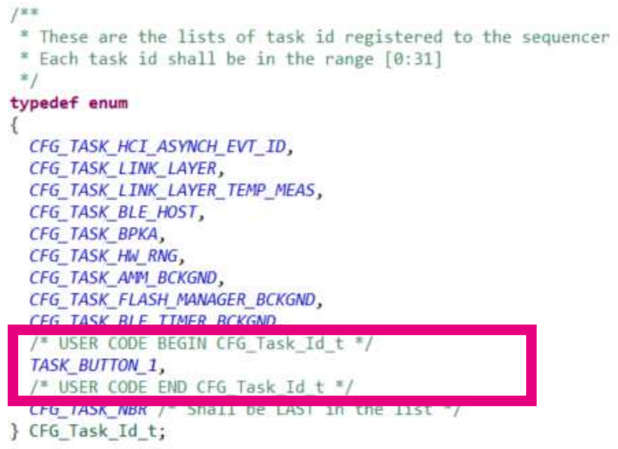
```
/* USER CODE BEGIN Service1Char1_WRITE_NO_RESP_EVT*/
HAL_GPIO_TogglePin(GPIOB, LD2_Pin|LD3_Pin|LD1_Pin);
/* USER CODE END Service1Char1_WRITE_NO_RESP_EVT */<
```

Search for "Service1Char1_WRITE_NO_RESP_EVT"

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_ASYNCH_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

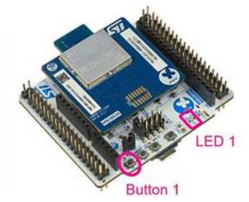
#3 **UTIL_SEQ_SetTask()** to notify the sequencer shall execute the registered task

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())

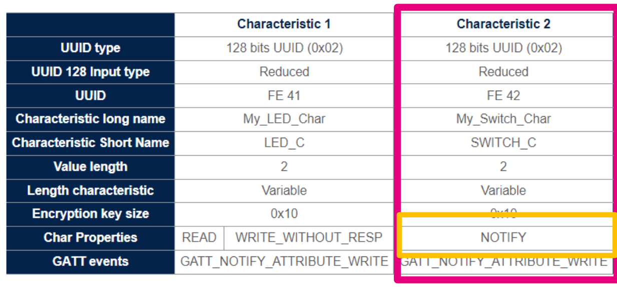
life.augmented

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



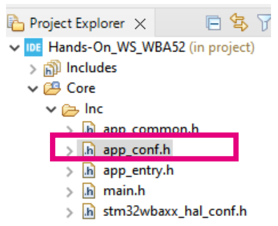
press button

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



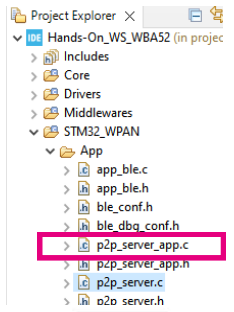
```
Project Explorer
v Hands-On_WS_WBA52 (in project)
  > Includes
  v Core
    v Inc
      > app_common.h
      > app_conf.h
      > app_entry.h
      > main.h
      > stm32wbaxx_hal_conf.h
```

#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*”



```
Project Explorer
v Hands-On_WS_WBA52 (in projec
  > Includes
  > Core
  > Drivers
  > Middlewares
  v STM32_WPAN
    v App
      > app_ble.c
      > app_ble.h
      > ble_conf.h
      > ble_dbg_conf.h
      > p2p_server_app.c
      > p2p_server_app.h
      > p2p_server.c
      > p2p_server.h
```

#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 */
```

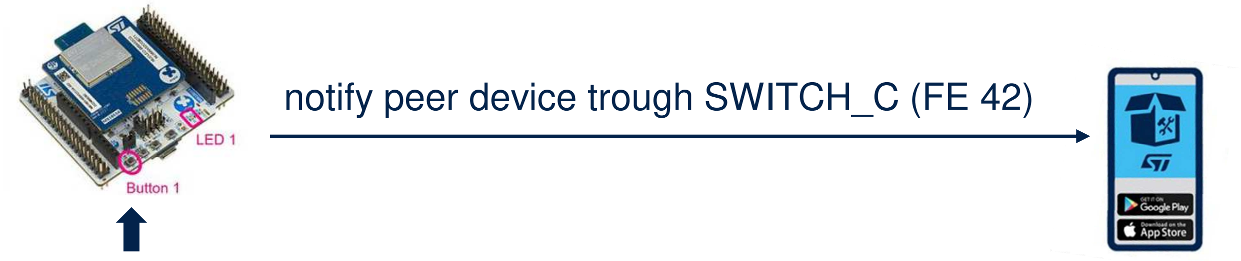
Search for “*Service1_APP_Init*”

Function generated by CubeMx as per as Characteristic Short Name

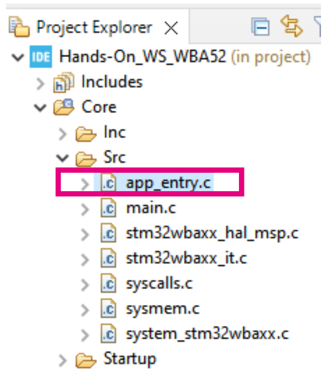
life.augmented

Add application code

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



press button



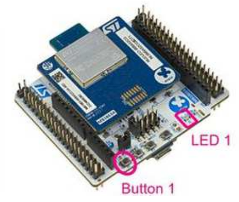
#3 Manage Button1 interrupt : implement IRQ callback

In app_entry.c / function HAL_GPIO_EXTI_Rising_Callback

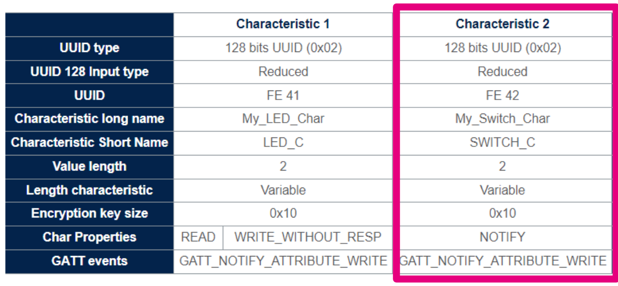
```
  /* USER CODE BEGIN FD_WRAP_FUNCTIONS */
void HAL_GPIO_EXTI_Rising_Callback(uint16_t GPIO_Pin)
{
  if (GPIO_Pin == B1_Pin)
  {
    UTIL_SEQ_SetTask(1U << TASK_BUTTON_1, CFG_SEQ_PRIO_0);
  }

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

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

Add application code
Raise an alarm from device to Smartphone(3/3)



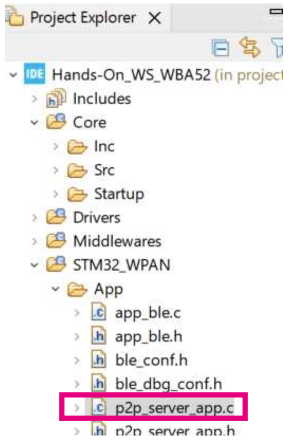
notify peer device trough SWITCH_C (FE 42)



	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



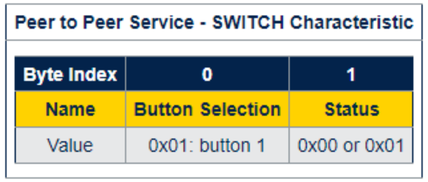
Project Explorer
- Hands-On_WS_WBA52 (in project
  - Includes
  - Core
    - Inc
    - Src
    - Startup
  - Drivers
  - Middlewares
  - STM32_WPAN
    - App
      - app_ble.c
      - app_ble.h
      - ble_conf.h
      - ble_dbg_conf.h
      - p2p_server_app.c
      - p2p_server_app.h

```
/* 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*/
```

Search for "*Service1Char2_NS_1* "

P2P_SERVER_UpdateValue

aci_gatt_update_char_value BLE stack API

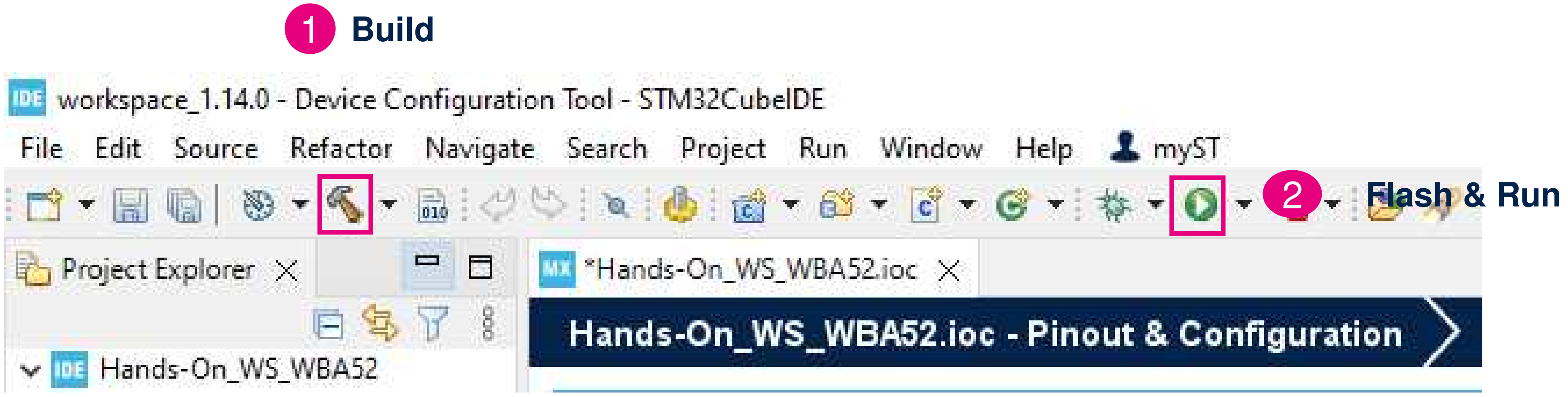


Peer to Peer Service - SWITCH Characteristic

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

STM32WBA Bluetooth® LE – Peer 2 Peer Applications - stm32mcu

Time to build, flash and execute !



workspace_1.14.0 - Device Configuration Tool - STM32CubeIDE

File Edit Source Refactor Navigate Search Project Run Window Help myST
Project Explorer
*Hands-On_WS_WBA52.ioc
Hands-On_WS_WBA52.ioc - Pinout & Configuration
Hands-On_WS_WBA52

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

Open your App and Connect

ST BLE Toolbox

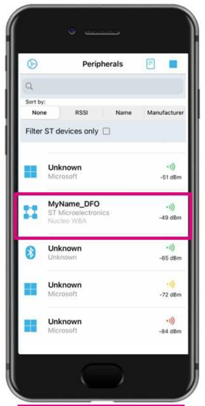
Download on the App Store

GET IT ON Google Play

life.augmented

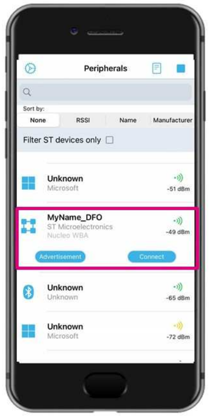
Open your App and Connect (1/2)

1



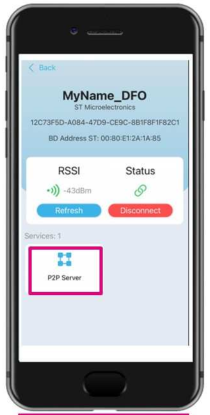
click on device

2



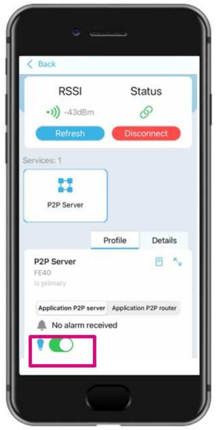
click on connect

3



access to profile

4

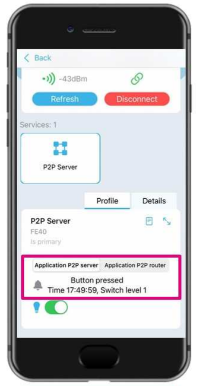


control LED status on Nucleo

	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

5



push button 1 and notify device

	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

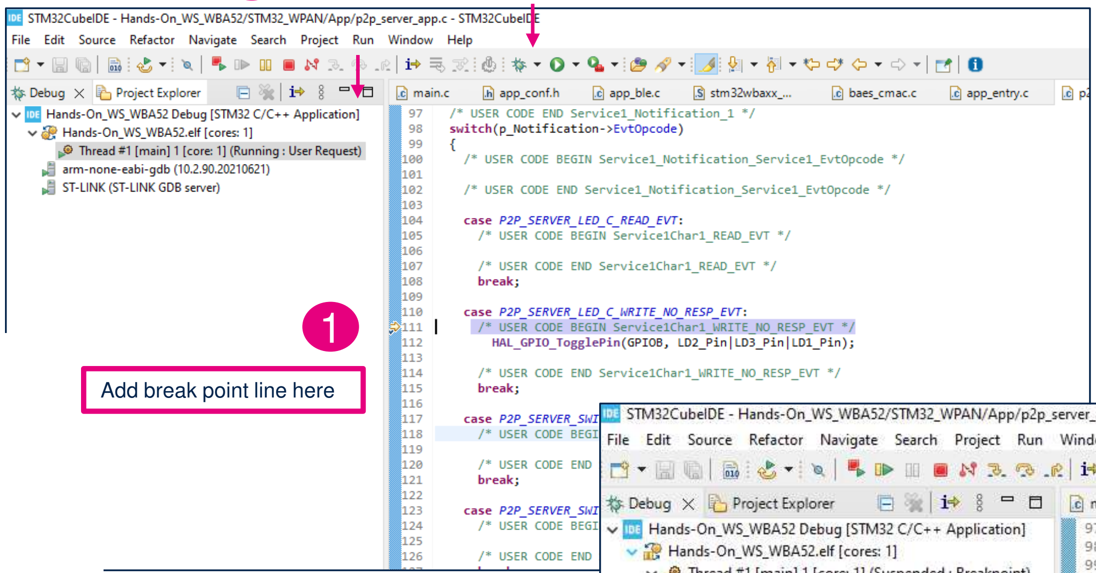
notify procedure use to push data to client

life.augmented

Bonus : Open your App and Connect call stack

3

2


STM32CubeIDE - Hands-On_WS_WBA52/STM32_WPAN/App/p2p_server_app.c - STM32CubeIDE

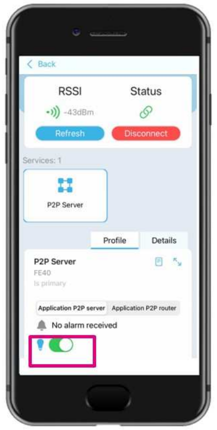
```
Hands-On_WS_WBA52 Debug [STM32 C/C++ Application]
  Hands-On_WS_WBA52.elf [cores: 1]
    Thread #1 [main] 1 [core: 1] (Running : User Request)
  arm-none-eabi-gdb (10.2.90.20210621)
  ST-LINK (ST-LINK GDB server)
```

```c
/* USER CODE END Service1_Notification_1 */
switch(p_Notification->EvtOpcode)
{
  /* USER CODE BEGIN Service1_Notification_Service1_EvtOpcode */

  /* USER CODE END Service1_Notification_Service1_EvtOpcode */

  case P2P_SERVER_LED_C_READ_EVT:
    /* USER CODE BEGIN Service1Char1_READ_EVT */

    /* USER CODE END Service1Char1_READ_EVT */
    break;

  case P2P_SERVER_LED_C_WRITE_NO_RESP_EVT:
    /* USER CODE BEGIN Service1Char1_WRITE_NO_RESP_EVT */
      HAL_GPIO_TogglePin(GPIOB, LD2_Pin|LD3_Pin|LD1_Pin);

    /* USER CODE END Service1Char1_WRITE_NO_RESP_EVT */
    break;

  case P2P_SERVER_SWI
    /* USER CODE BEGI

    /* USER CODE END
    break;

  case P2P_SERVER_SWI
    /* USER CODE BEGI

    /* USER CODE END
```

1

Add break point line here

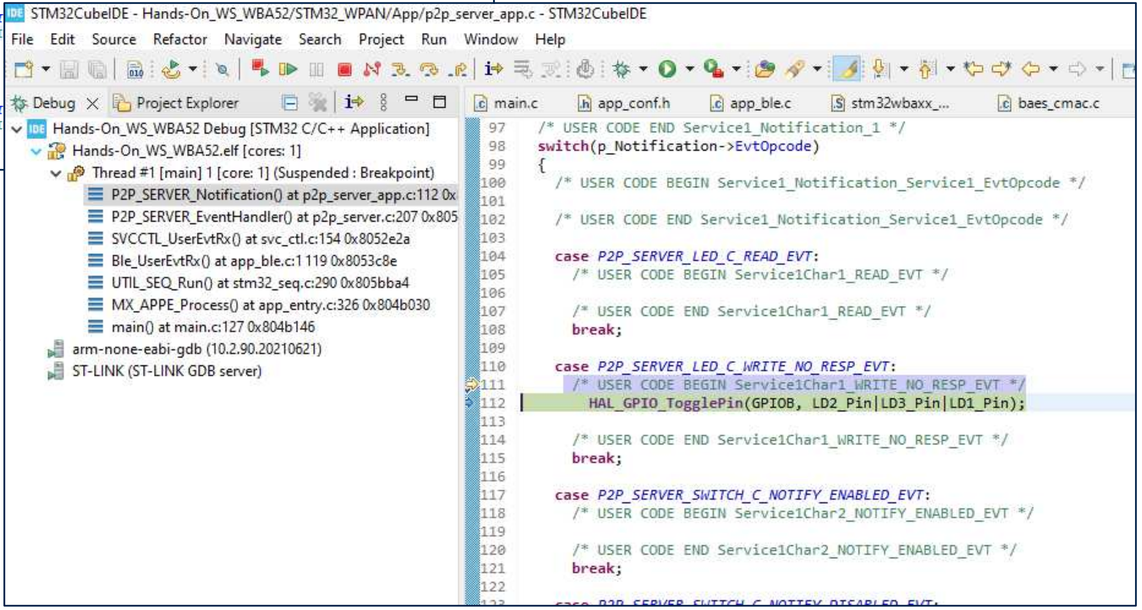
4



control LED status on Nucleo

BLE write procedure initiated by client

ACI_GATT_ATTRIBUTE_MODIFIED event received at application level


STM32CubeIDE - Hands-On_WS_WBA52/STM32_WPAN/App/p2p_server_app.c - STM32CubeIDE

```
Hands-On_WS_WBA52 Debug [STM32 C/C++ Application]
  Hands-On_WS_WBA52.elf [cores: 1]
    Thread #1 [main] 1 [core: 1] (Suspended : Breakpoint)
      P2P_SERVER_Notification() at p2p_server_app.c:112 0x
      P2P_SERVER_EventHandler() at p2p_server.c:207 0x805
      SVCCTL_UserEvtRx() at svc_ctl.c:154 0x8052e2a
      Ble_UserEvtRx() at app_ble.c:1 119 0x8053c8e
      UTIL_SEQ_Run() at stm32_seq.c:290 0x805bba4
      MX_APPE_Process() at app_entry.c:326 0x804b030
      main() at main.c:127 0x804b146
  arm-none-eabi-gdb (10.2.90.20210621)
  ST-LINK (ST-LINK GDB server)
```

```c
/* USER CODE END Service1_Notification_1 */
switch(p_Notification->EvtOpcode)
{
  /* USER CODE BEGIN Service1_Notification_Service1_EvtOpcode */

  /* USER CODE END Service1_Notification_Service1_EvtOpcode */

  case P2P_SERVER_LED_C_READ_EVT:
    /* USER CODE BEGIN Service1Char1_READ_EVT */

    /* USER CODE END Service1Char1_READ_EVT */
    break;

  case P2P_SERVER_LED_C_WRITE_NO_RESP_EVT:
    /* USER CODE BEGIN Service1Char1_WRITE_NO_RESP_EVT */
      HAL_GPIO_TogglePin(GPIOB, LD2_Pin|LD3_Pin|LD1_Pin);

    /* USER CODE END Service1Char1_WRITE_NO_RESP_EVT */
    break;

  case P2P_SERVER_SWITCH_C_NOTIFY_ENABLED_EVT:
    /* USER CODE BEGIN Service1Char2_NOTIFY_ENABLED_EVT */

    /* USER CODE END Service1Char2_NOTIFY_ENABLED_EVT */
    break;
```

Takeaways
What’s next

✓

Hands-on#2 – Build a BLE advertising device

Evaluate, prototype & customize your own project with your own BLE proprietary profile requirements.

↓

Build and optimize you PCB and move to certification

HW guideline, what are the available resources what I should focus on.

Thank you

life.augmented