



life.augmented

STM32 Quest : 2024 University Developer Contest (GFX & Wireless)

Mission 2 : Wireless basics with simple P2P BLE server

STMicroelectronics

Mission overview

- Goal :
 - The goal of the second mission is to become familiar with STM32WB55 and the most common BLE profile, the P2P server.
- Steps :
 - In the first part “STM32WB – Preparations”, you will setup your PC and install the right tools to start developing on STM32WB55.
 - In the “STM32WB – RF Stack upgrade” part, you will update the wireless stack. This is a mandatory step prior to starting the user application development.
 - In the last part, you will implement a P2P Server that will connect to a phone and send debug data to your PC.
- Completion Conditions :
 - After the project is complete, record a video of the STM32WB55 and smartphone interaction.

Useful resources

- STM32WB Wiki page :
https://wiki.st.com/stm32mcu/wiki/Category:STM32WB_Series
- STM32WB Online training:
https://www.st.com/content/st_com/en/support/learning/stm32-education/stm32-online-training/stm32wb-online-training.html
- STM32WB Online training video session (Youtube):
https://www.youtube.com/playlist?list=PLnMKNibPkDnGkMxFkRArr9uOq_Es_a7vG

STM32WB - Preparations

Hardware Preparations

- **Laptop**

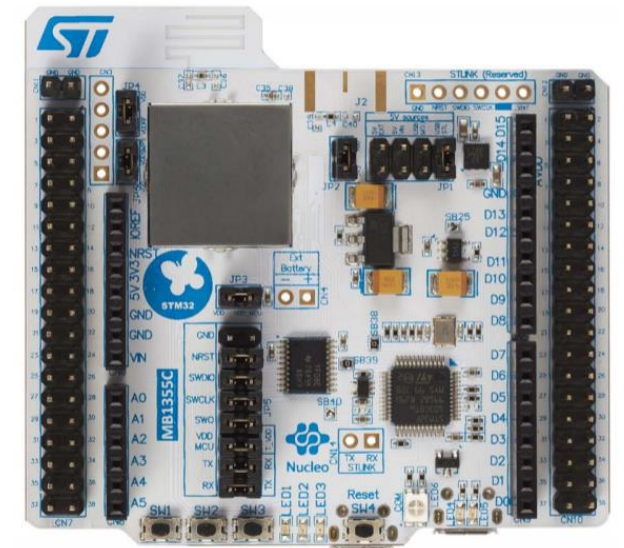
- Administrative privileges are needed for the driver/software installation and later during the workshop for compiling the code.
- Window 10-64bit is preferred.

- **Mobile Phone (Android or iOS)**

- Need a cable to connect to the laptop.

- **NUCLEO-WB55RG**

- Need a USB A to Micro-B cable
- <https://www.st.com/en/evaluation-tools/nucleo-wb55rg.html>



Hardware NUCLEO-WB55RG

RF area

- 2.4GHz PCB Antenna
- Impedance matching
- Ext. LP filter (IPD)
- (space for SMA connector and RF switch)

MCU area

- STM32WB55RG
- HSE & LSE crystals
- Decoupling
- SMPS ext. parts

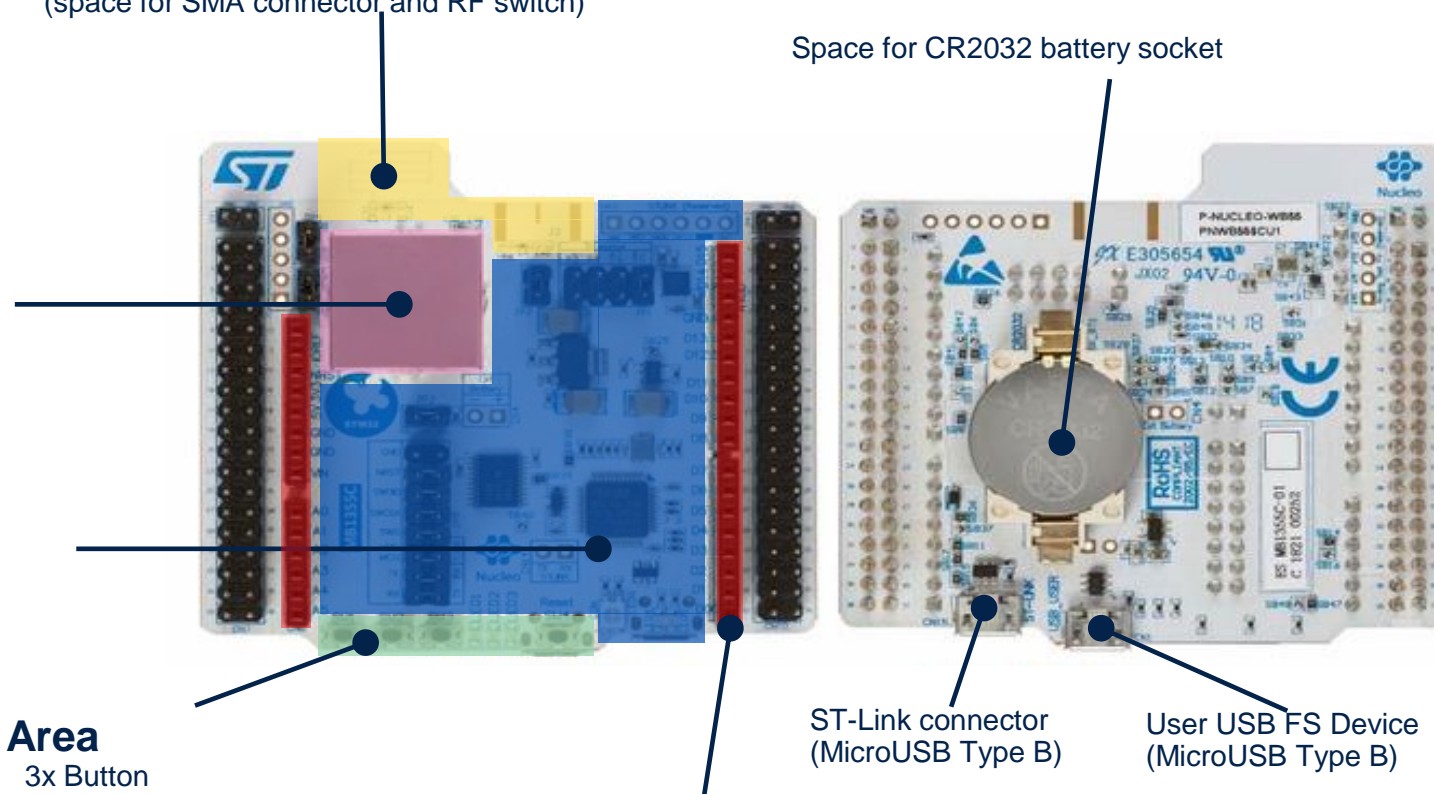
ST-Link area

- ST-Link/V2-1
- SWD debugger
- Virtual COM Port
- USB MSC (.bin flashing)

UI Area

- 3x Button
- 3x LED
- Reset Button

Space for CR2032 battery socket



Arduino Uno V3 Connector

ST-Link connector
(MicroUSB Type B)

User USB FS Device
(MicroUSB Type B)

Software Tools Preparations

- Register and account at www.st.com & download the following software form the links provided.
- **STM32CubeIDE (latest version) – Download and Install**
 - <https://www.st.com/en/development-tools/stm32cubeide.html>
- **STM32CubeMX (latest version) – Download and Install**
 - <https://www.st.com/en/development-tools/stm32cubemx.html>
- **STM32CubeProgrammer (latest version) – Download and Install**
 - <https://www.st.com/en/development-tools/stm32cubeprog.html>
- **Mobile Applications**
 - **STBLESensor** - <https://www.st.com/en/embedded-software/stblesensor.html>
 - **STBLEToolbox** - <https://www.st.com/en/embedded-software/stbletoolbox.html>

CubeMX Installation STM32WB Package

The screenshot shows the STM32CubeMX Embedded Software Packages Manager window. The main window displays a list of packages with columns for Description, Installed Version, and Available Version. The STM32WB package is highlighted, and a yellow box around it contains the text "Download latest version". A yellow arrow points from this box to the "Install" button in the bottom right corner of the window. The "Install" button is also highlighted with a yellow box. The window title is "STM32CubeMX Embedded Software Packages Manager".

SEGG	WES	emotas	portGmbH	wolfSSL
STM32Cube MCU Packages		STMicroelectronics		
Description		Installed Version	Available Version	
STM32WB				
STM32Cube MCU Package for STM32WB Series			Download latest version	
STM32Cube MCU Package for STM32WB Series (Size : 163 MB)			1.15.0	

Details

[STM32CubeWB Firmware Package V1.16.0 / 08-February-2023](#)

Main Changes

Maintenance Release for BLE, 802.15.4, Thread and Zigbee updates

Porting IAR toolChain 9.20.1.

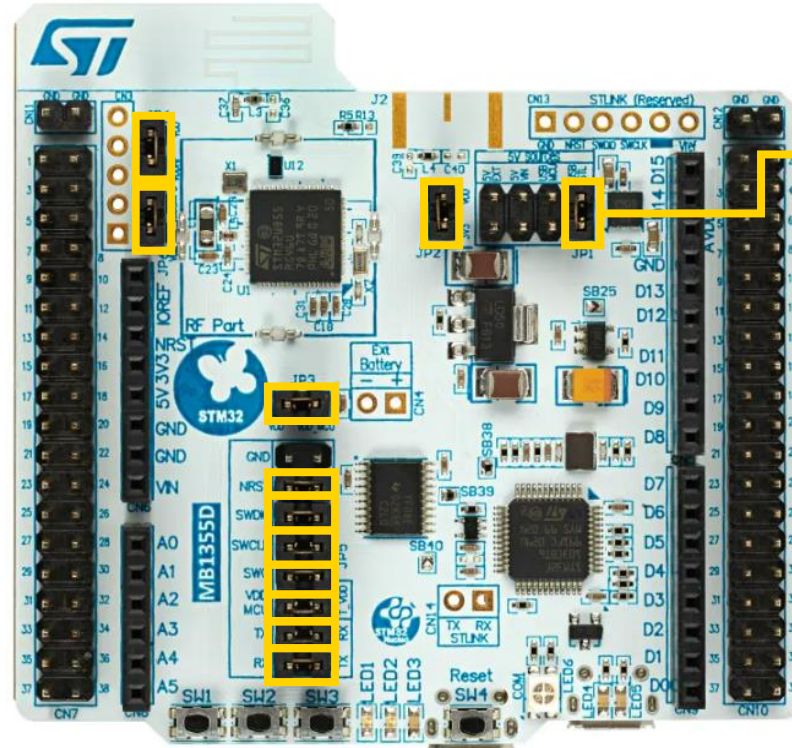
Thread updates

From Local ... From Url ... Refresh **Install** Remove Close

STM32WB – RF Stack upgrade



Wireless Stack Update Jumper Setting



JP1 setting should be "USB STL".

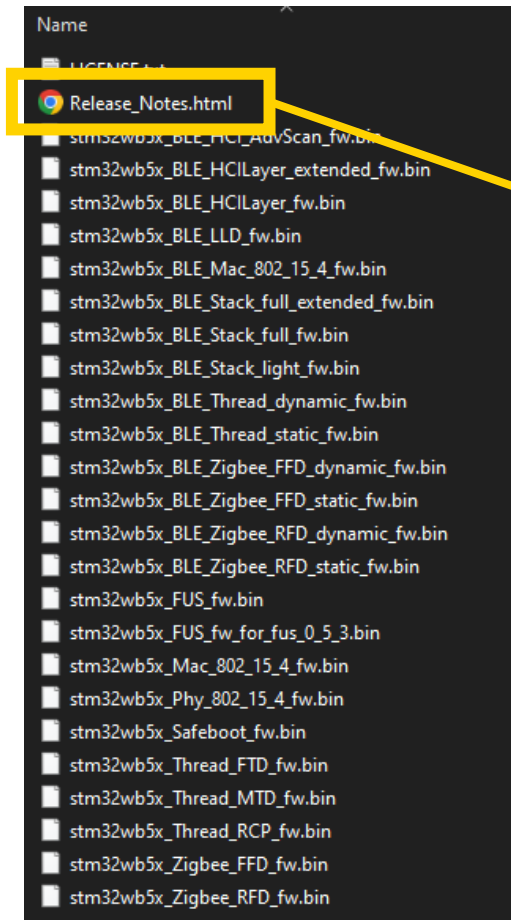
Please make sure all the
jumpers are correctly placed.



Wireless Stack Update

Confirm the install address of co-processor binary

- **C:\Users\{user_name}\STM32Cube\Repository\STM32Cube_FW_WB_V1.16.0\Projects\STM32WB_Copro_Wireless_Binaries\STM32WB5x**
- Refer to **Release_Notes.html** in the above path to update **FUS** (Firmware Update Services) and **wireless stack** firmware.
- Confirm the install address of co-processor binary according to your part number.
(NUCLEO_WB55 : STM32WB55RG)
- We will use “stm32wb5x_BLE_Stack_full_fw.bin” for today’s hand-on training.



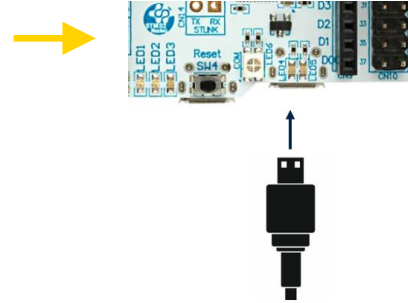
Wireless Coprocessor Binary Table: Provides Install address for the

Wireless Coprocessor Binary	STM32WB5xxG(1M)
stm32wb5x_BLE_HCI_Layer_extended_fw.bin	0x080DC000
stm32wb5x_BLE_HCI_Layer_fw.bin	0x080E0000
stm32wb5x_BLE_HCI_AdvScan_fw.bin	0x080EB000
stm32wb5x_BLE_LLD_fw.bin	0x080ED000
stm32wb5x_BLE_Mac_802_15_4_fw.bin	0x080B9000
stm32wb5x_BLE_Stack_full_extended_fw.bin	0x080C7000
stm32wb5x_BLE_Stack_full_fw.bin	0x080CE000
stm32wb5x_BLE_Stack_light_fw.bin	0x080C8000
stm32wb5x_BLE_Thread_dynamic_fw.bin	0x080C9000
stm32wb5x_BLE_Thread_static_fw.bin	0x080CA000
stm32wb5x_BLE_Zigbee_FFD_dynamic_fw.bin	0x080CB000
stm32wb5x_BLE_Zigbee_FFD_static_fw.bin	0x080CC000
stm32wb5x_BLE_Zigbee_RFD_dynamic_fw.bin	0x080CD000
stm32wb5x_BLE_Zigbee_RFD_static_fw.bin	0x080CE000
stm32wb5x_FUS_fw.bin	0x080CF000
stm32wb5x_FUS_fw_for_fus_0_5_3.bin	0x080D0000
stm32wb5x_Mac_802_15_4_fw.bin	0x080D1000
stm32wb5x_Phy_802_15_4_fw.bin	0x080D2000
stm32wb5x_Safeboot_fw.bin	0x080D3000
stm32wb5x_Thread_FTD_fw.bin	0x080D4000
stm32wb5x_Thread_MTD_fw.bin	0x080D5000
stm32wb5x_Thread_RCP_fw.bin	0x080D6000
stm32wb5x_Zigbee_FFD_fw.bin	0x080D7000
stm32wb5x_Zigbee_RFD_fw.bin	0x080D8000

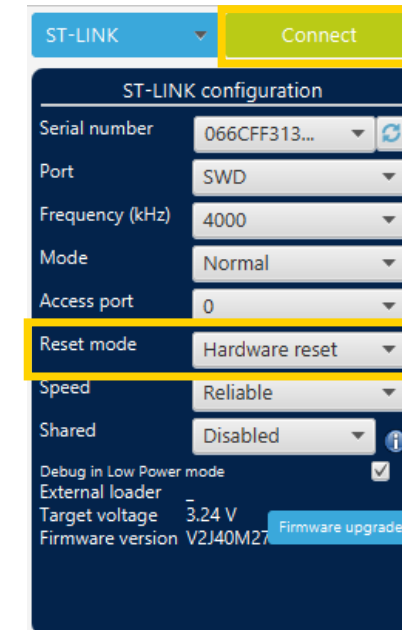
stm32wb5x_BLE_Stack_full_fw.bin
→ 0x080CE000

Wireless Stack Update Changing boot mode

1. Connect Nucleo board to laptop



2. Click connect ST-LINK with Hardware reset mode.



3. Click “Option Bytes” icon and expand “User Configuration”

- ✓ nBOOT0 – tick
- ✓ nBOOT1 – tick
- ✓ nSWBOOT0 – untick






User Configuration	
Name	Value
nBOOT0	<input checked="" type="checkbox"/>
nBOOT1	<input checked="" type="checkbox"/>
nSWBOOT0	<input type="checkbox"/>

4. Then, “Apply”



Wireless Stack Update Confirm FUS Version

1. Click “Firmware Upgrade Service” icon 
2. Start FUS (Need to try it twice at least if fail) 
3. Click “Read FUS infos” after StartFus activated.
4. FUS Version : 1.2.0.0

 →

FUS State	FUS_IDLE
FUS Status	FUS_NO_ERROR
FUS Version	v1.2.0.0
STACK Version	v1.14.1.1
FUS Operator	v3.1.0

If FUS version is not v1.2.0.0, Should do the following steps in release note. (Refer. the release note.)

- STEP 4: Read and upgrade FUS Version
 - it can be obtained selecting “Read FUS infos”
 - 00050300: FUSv0.5.3 => **Must be updated using STEP 5.**
 - 010X0Y00: FUSv1.x.y => **Must be updated using STEP 6 (when x < 2).**
 - 01020000: FUSv1.2.0 => **Up to date, you can download the new wireless stack using STEP 7.**
- STEP 5: Download latest FUS for only FUSv0.5.3 upgrade
 - in Firmware Upgrade Services: (File Path: [stm32wb5x_FUS_fw_for_fus_0_5_3.bin], Start Address: [Install@])
 - then select “Firmware Upgrade” Please check **Firmware Upgrade Services Binary Table** for Install@ parameter depending of the binary.
- STEP 6: Download latest FUS or Safeboot
 - in Firmware Upgrade Service: (File Path: [FUS_Binary], Start Address: [Install@])
 - then select “Firmware Upgrade” Please check **Firmware Upgrade Services Binary Table** for Install@ parameter depending of the binary.

Wireless Stack Update Co-processor Binary Update

- **C:\Users\{user_name}\STM32Cube\Repository\STM32Cube_FW_WB_V1.16.0\Projects\STM32WB_Copro_Wireless_Binaries\STM32WB5x**

1. Click “Browse” to choose the binary of co-processor.

- stm32wb5x_BLE_Stack_full_fw.bin

2. Input the install address to Start address

- 0x080CE000

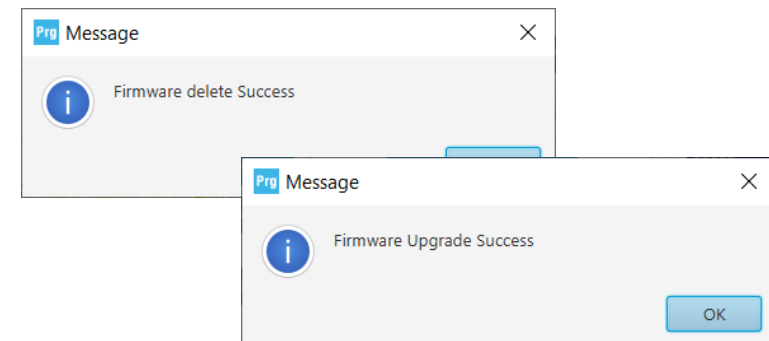
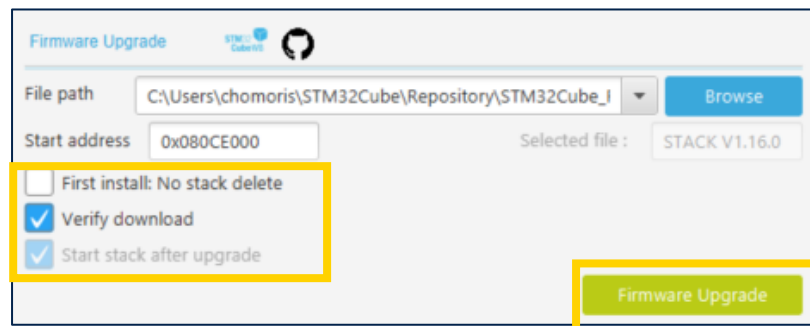
3. Check options

- First install – Untick
- Verify download – Tick
- Start stack after upgrade – Tick (Default)

4. Click “Firmware Upgrade”

Wireless Coprocessor Binary Table: Provides Install address for the

Wireless Coprocessor Binary	STM32WB5xxG(1M)
stm32wb5x_BLE_HCI_Layer_extended_fw.bin	0x080DC000
stm32wb5x_BLE_HCI_Layer_fw.bin	0x080E0000
stm32wb5x_BLE_HCI_AdvScan_fw.bin	0x080EB000
stm32wb5x_BLE_LLD_fw.bin	0x080ED000
stm32wb5x_BLE_Mac_802_15_4_fw.bin	0x080B9000
stm32wb5x_BLE_Stack_full_extended_fw.bin	0x080C7000
stm32wb5x_BLE_Stack_full_fw.bin	0x080CE000
stm32wb5x_BLE_Stack_Light_fw.bin	0x080C8000

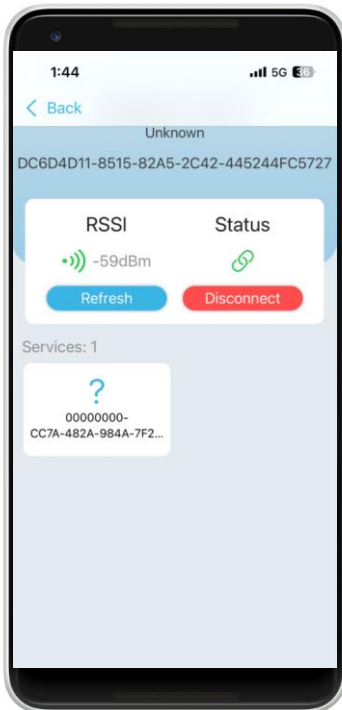


Making simple BLE project

Making simple BLE project Receiving data from phone

Central

GATT Client



- Write a single arbitrary byte to the characteristic.
- Green LED will be turned on or off.
- Parsing the input value is shown as optional hands-on at the end of this presentation.

Write

Custom SERVICE

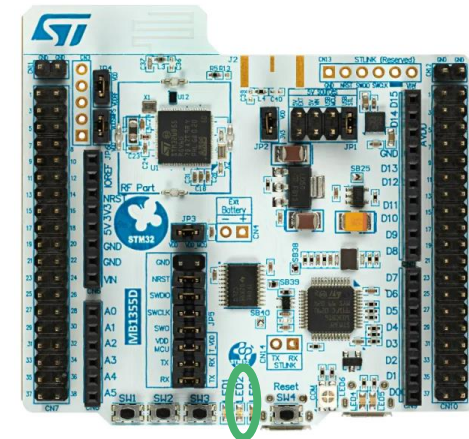
Custom CHARACTERISTIC 1

W

LED Control

Peripheral

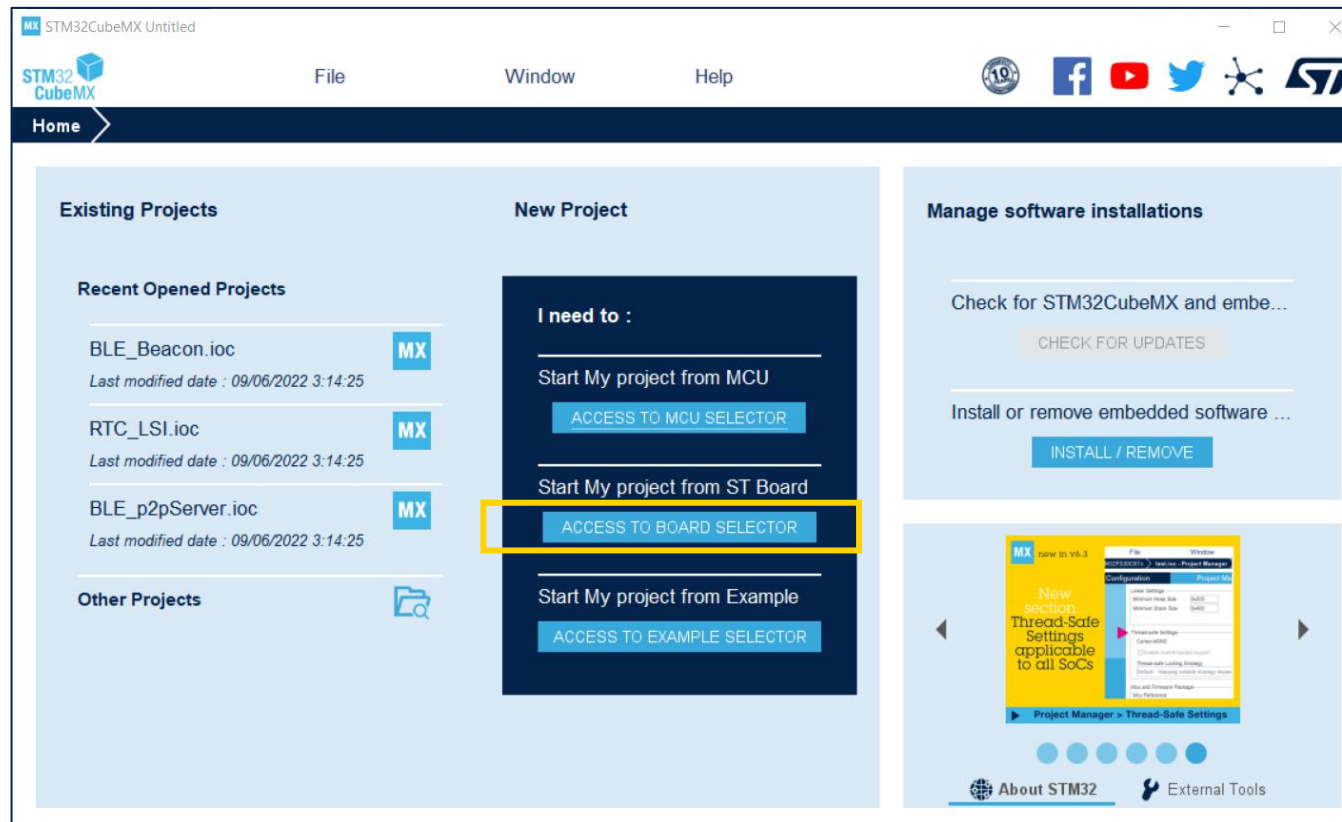
GATT Server



Green LED

Making simple BLE project Step #1

1. Execute “STM32CubeMX”.
2. Select “ACCESS TO BOARD SELECTOR”



Making simple BLE project

Step #2

- Typing “NUCLEO-WB55RG” on Commercial Part Number.

The screenshot shows the 'New Project from a Board' interface. The 'Board Filters' section on the left has a 'Commercial Part Number' dropdown menu with 'NUCLEO-WB55RG' selected. Below this, there are sections for 'PRODUCT INFO', 'MEMORY', and 'FEATURES'. The main content area displays a banner for the 'STM32U5 ultra-low-power MCU series with comprehensive STM32Cube ecosystem'. Below the banner, a 'Boards List' table shows one item:

	Overview	Commercial Part No	Type	Marketing Status	Unit Price (US\$)	Mounted Device
☆		NUCLEO-WB55RG	Nucleo-64	Active	35.0	STM32WB55RGV8

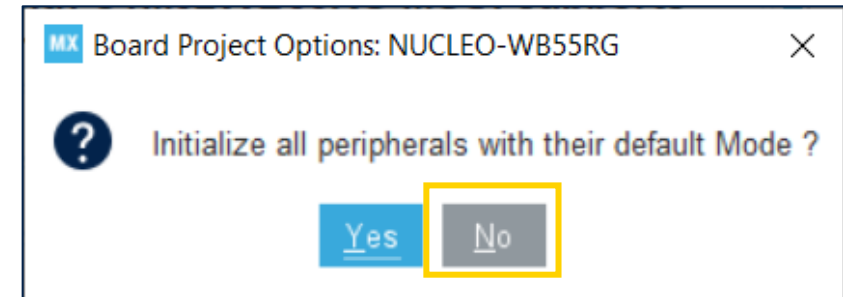
Making simple BLE project

Step #3

1. Click NUCLEO-WB55RG
2. Making an example project for Hid after select IDE and project's folder.

Boards List: 1 item Export

	Overview	Commercial Part No	Type	Marketing Status	Unit Price (US\$)	Mounted Device
		NUCLEO-WB55RG	Nucleo-64	Active	35.0	STM32WB55RGV6



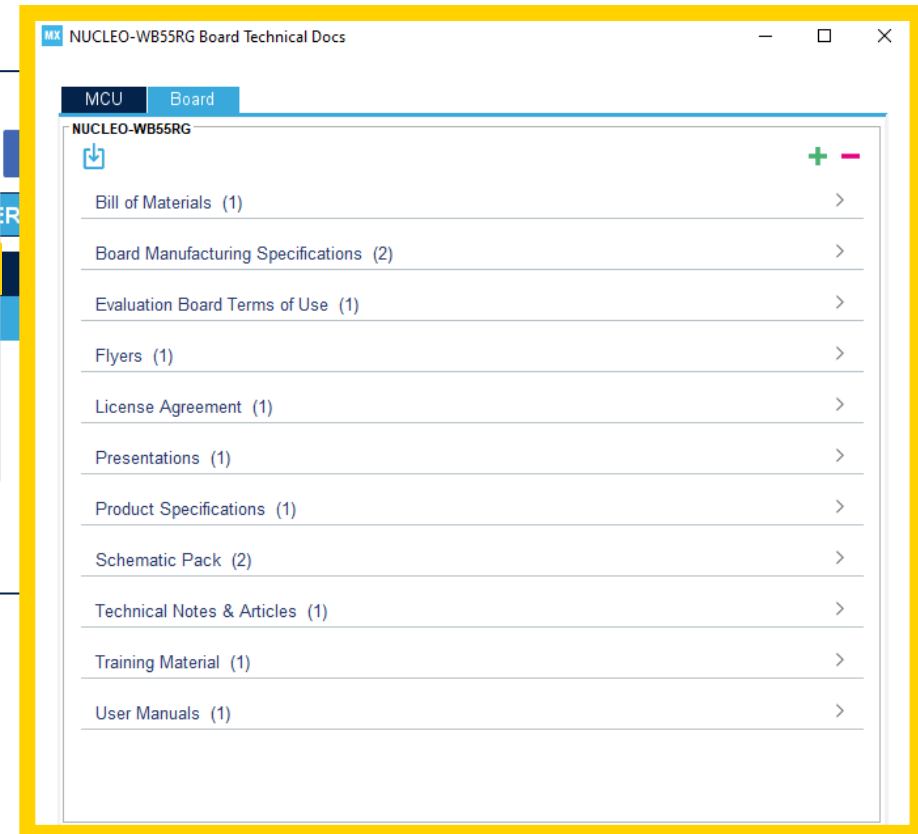
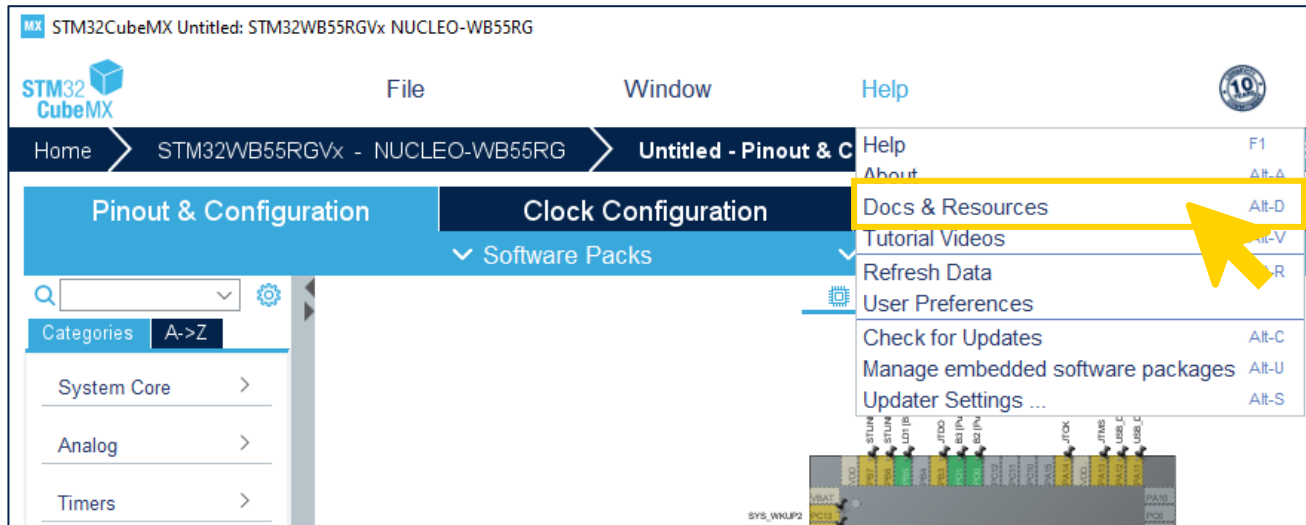
Initialize all peripherals with their default Mode? **No**

This will only initialize GPIO for LEDs and Buttons

Making simple BLE project

Step #4

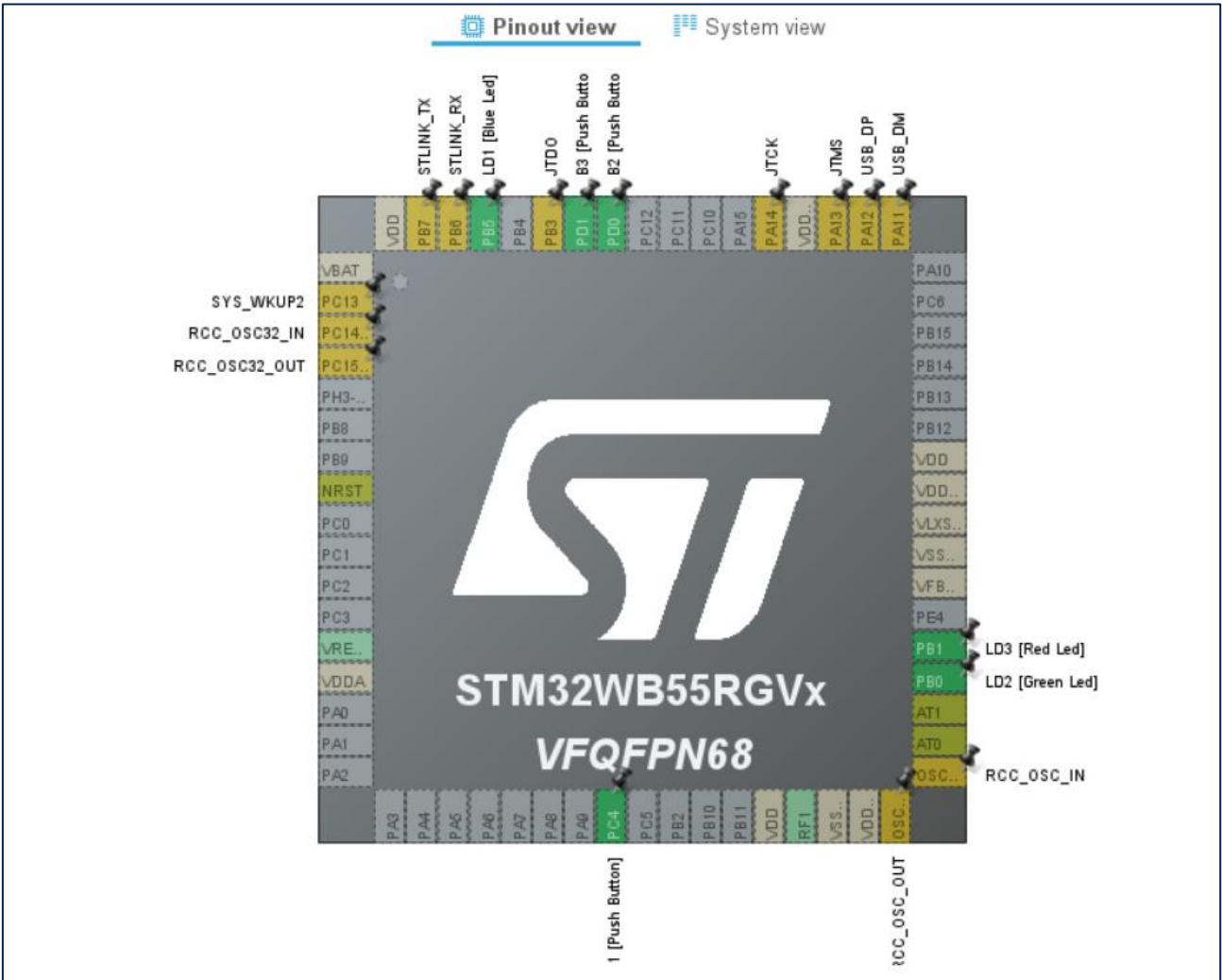
- If you need more information about NUCLEO board, you can check “Docs & Resources”



Making simple BLE project

Step #5

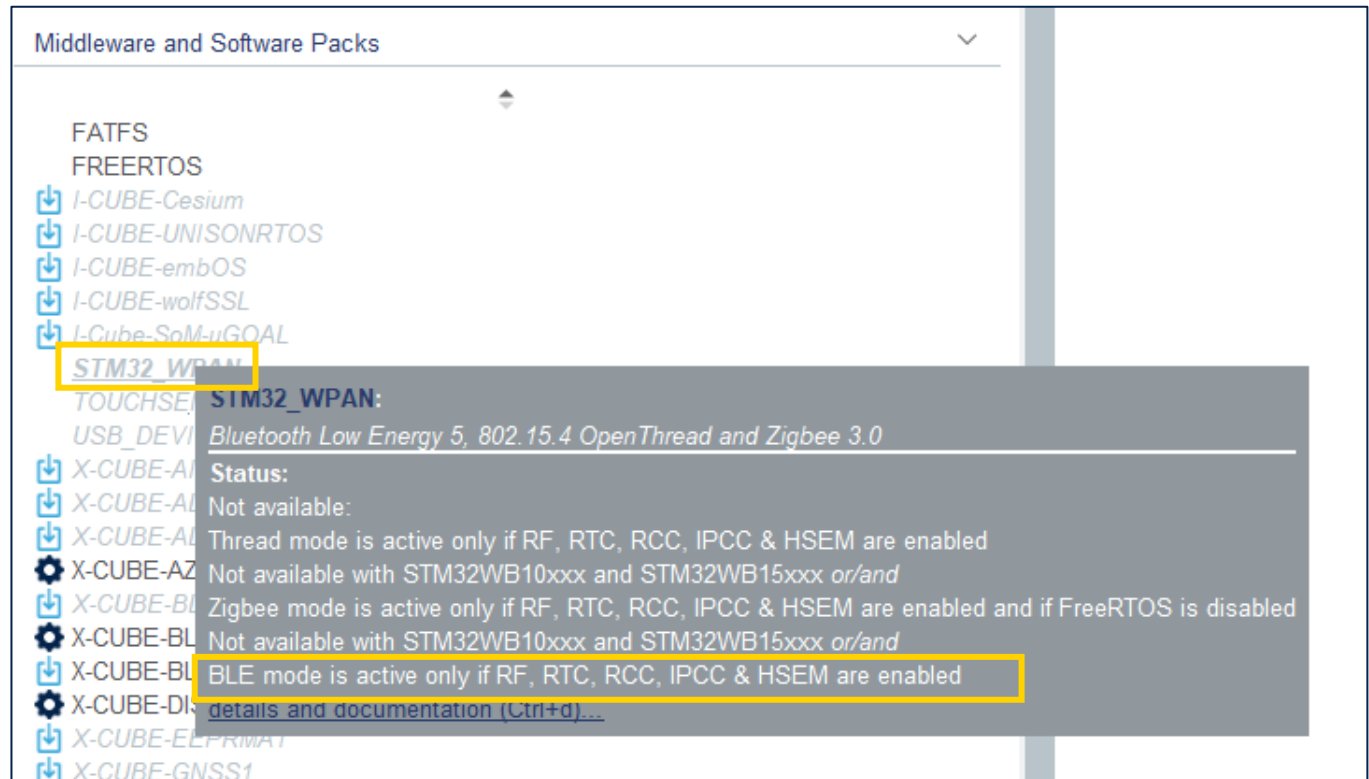
- GPIOs are assigned by default to match the corresponding board.



Making simple BLE project

Step #6

- Hover the cursor on STM32_WPAN Middleware.
- It will give you contextual help on what to do. We will do each step together.

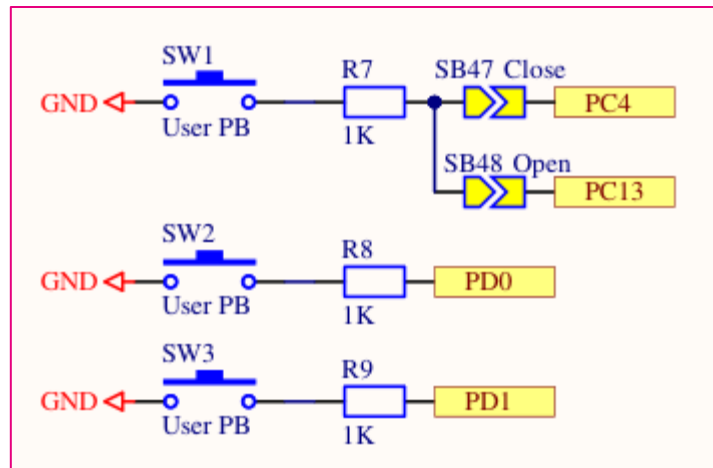


BLE mode is active only if RF, RCC, IPCC & HSEM are enabled.

Making simple BLE project

Step #7

- Select GPIO under System Core Tab.
- Configuring Pull-up for B1, B2 and B3.



The screenshot shows the STM32CubeMX software interface. The 'Pinout & Configuration' tab is active, and the 'GPIO Mode and Configuration' table is displayed. The table shows the following configuration for pins PC4, PD0, and PD1:

Pin N...	Signal on ...	GPIO out...	GPIO mode	GPIO Pul...	Maximum...	Fast...	User Label	Modified
PB0	n/a	Low	Output P...	No pull-u...	Low	n/a	LD2 [Green ...	<input checked="" type="checkbox"/>
PB1	n/a	Low	Output P...	No pull-u...	Low	n/a	LD3 [Red Led]	<input checked="" type="checkbox"/>
PB5	n/a	Low	Output P...	No pull-u...	Low	n/a	LD1 [Blue L...	<input checked="" type="checkbox"/>
PC4	n/a	n/a	Input mode	Pull-up	n/a	n/a	B1 [Push Bu...	<input checked="" type="checkbox"/>
PD0	n/a	n/a	Input mode	Pull-up	n/a	n/a	B2 [Push B...	<input checked="" type="checkbox"/>
PD1	n/a	n/a	Input mode	Pull-up	n/a	n/a	B3 [Push B...	<input checked="" type="checkbox"/>

Below the table, the 'PC4 Configuration' section shows the following settings:

- GPIO mode: Input mode
- GPIO Pull-up/Pull-down: Pull-up
- User Label: B1 [Push Button]

Making simple BLE project

Step #8

- Select SYS under System Core Tab.
- Debug
 - Serial Wire

The screenshot displays the STM32CubeMX interface for configuring the Serial Wire Debug (SWD) peripheral. The left sidebar shows the 'System Core' tab selected, with 'SYS' highlighted. The main configuration area shows the 'Debug [Serial Wire]' mode selected. Below this, several options are listed, including 'System Wake-Up 1' through 'System Wake-Up 5', 'Power Voltage Detector In' (set to 'Disable'), 'VREFBUF Mode' (set to 'Disable'), and 'Timebase Source' (set to 'SysTick'). A warning message at the bottom of the configuration area reads: 'Warning: This peripheral has no parameters to be configured.' An inset image shows the physical pins PA14 (JTCK) and PA13 (JTMS) on a microcontroller package.

Making simple BLE project

Step #9

- Select RCC under System Core Tab.
- Enable HSE.
→ Crystal/Ceramic Resonator.
- Enable LSE.
→ Crystal/Ceramic Resonator.
- HSE 32Mhz is directly used by the radio PHY.
- LSE is used as a low-speed clock by radio PHY.
It is used to time events such as connection or advertising interval.

The screenshot shows the STM32CubeMX interface for configuring the RCC (Reset and Clock Controller) for an STM32WB55RGVx NUCLEO-WB55RG. The 'RCC Mode and Configuration' window is open, showing the 'Mode' section where both High Speed Clock (HSE) and Low Speed Clock (LSE) are set to 'Crystal/Ceramic Resonator'. The 'Configuration' section shows a table of pin configurations for the oscillator signals.

Pin No.	Signal on	GPIO outp.	GPIO mode	GPIO Pull...	Maximum ...	Fast Mode	User Label
OSC_IN	RCC_OSC...	n/a	n/a	n/a	n/a	n/a	
OSC_OUT	RCC_OSC...	n/a	n/a	n/a	n/a	n/a	
PC14-OS...	RCC_OSC...	n/a	n/a	n/a	n/a	n/a	
PC15-OS...	RCC_OSC...	n/a	n/a	n/a	n/a	n/a	

Additional callouts in the image show the physical pin connections: RCC_OSC32_IN is connected to PC14 and RCC_OSC32_OUT is connected to PC15. Another callout shows the oscillator pins (OSC_IN and OSC_OUT) connected to the board's oscillator pins.

Making simple BLE project

Step #10

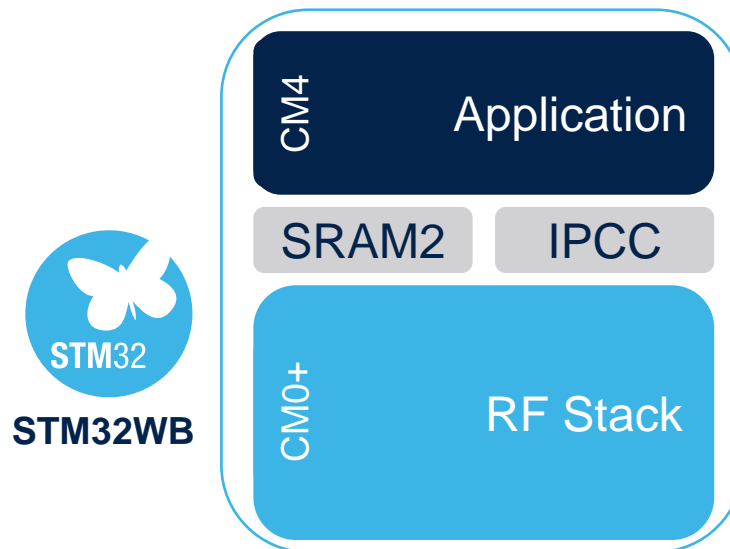
- Select HSEM (Hardware Semaphores) and tick activate.
- HSEM provides synchronization between CM0+ and CM4 when using shared resources. (ex. Clock tree registers, RNG, etc.)

The screenshot shows the STM32CubeMX interface for configuring the HSEM (Hardware Semaphore) on a NUCLEO-WB55RG. The 'Pinout & Configuration' tab is active, and the 'HSEM Mode and Configuration' window is open. The 'Mode' section shows 'Activated' checked. The 'Configuration' section shows the 'NVIC Interrupt Table' with the 'HSEM global interrupt' enabled.

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
HSEM global interrupt	<input checked="" type="checkbox"/>	0	0

Making simple BLE project Step #11

- Select IPCC (Inter-processor communication controller)
- Tick “activated”.
- Enable both interrupts in NVIC.
- IPCC provides asynchronous messaging mechanism between CM4 and CM0+.
- Part of SRAM2 is shared.



The screenshot shows the STM32CubeMX software interface. The 'Pinout & Configuration' tab is active, and the 'IPCC Mode and Configuration' section is expanded. The 'Mode' is set to 'Activated', which is highlighted with a yellow box. In the left-hand 'Categories' list, 'IPCC' is selected and highlighted with a yellow box. Below, the 'Configuration' section shows the 'NVIC Settings' tab. A table titled 'NVIC Interrupt Table' is visible, with the following data:

Parameter Settings	Enabled	Preemption Priority	Sub Priority
IPCC RX occupied interrupt	<input checked="" type="checkbox"/>	0	0
IPCC TX free interrupt	<input checked="" type="checkbox"/>	0	0

Making simple BLE project

Step #12

- Select RTC under Timer Tab.
- Tick Activate Clock Source
- Enable Internal Wakeup
- Enable interrupt in NVIC
- Required due to Virtual timer server FW component. (We will not use it in this hands-on)

The screenshot displays the STM32CubeMX software interface for configuring the RTC (Real Time Clock) on a STM32WB55RGVx NUCLEO-WB55RG board. The interface is divided into several sections:

- Pinout & Configuration**: The left sidebar shows the configuration tree with **RTC** selected under the **Timers** category.
- Clock Configuration**: The main panel shows the **RTC Mode and Configuration** settings. Key options are highlighted with yellow boxes:
 - Activate Clock Source**
 - Activate Calendar**
 - Alarm A**: Disable
 - Alarm B**: Disable
 - Timestamp**
 - WakeUp**: Internal WakeUp
 - Tamper 1**: Enabled (highlighted in pink)
 - Tamper 2**
 - Tamper 3**
 - Calibration**: Disable
 - Reference clock detection**
- Configuration**: The bottom section shows the **NVIC Settings** table, where the **RTC wake-up interrupt through EXTI line 19** is configured as **Enabled** (highlighted in yellow).

Making simple BLE project

Step #13

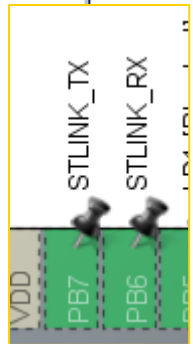
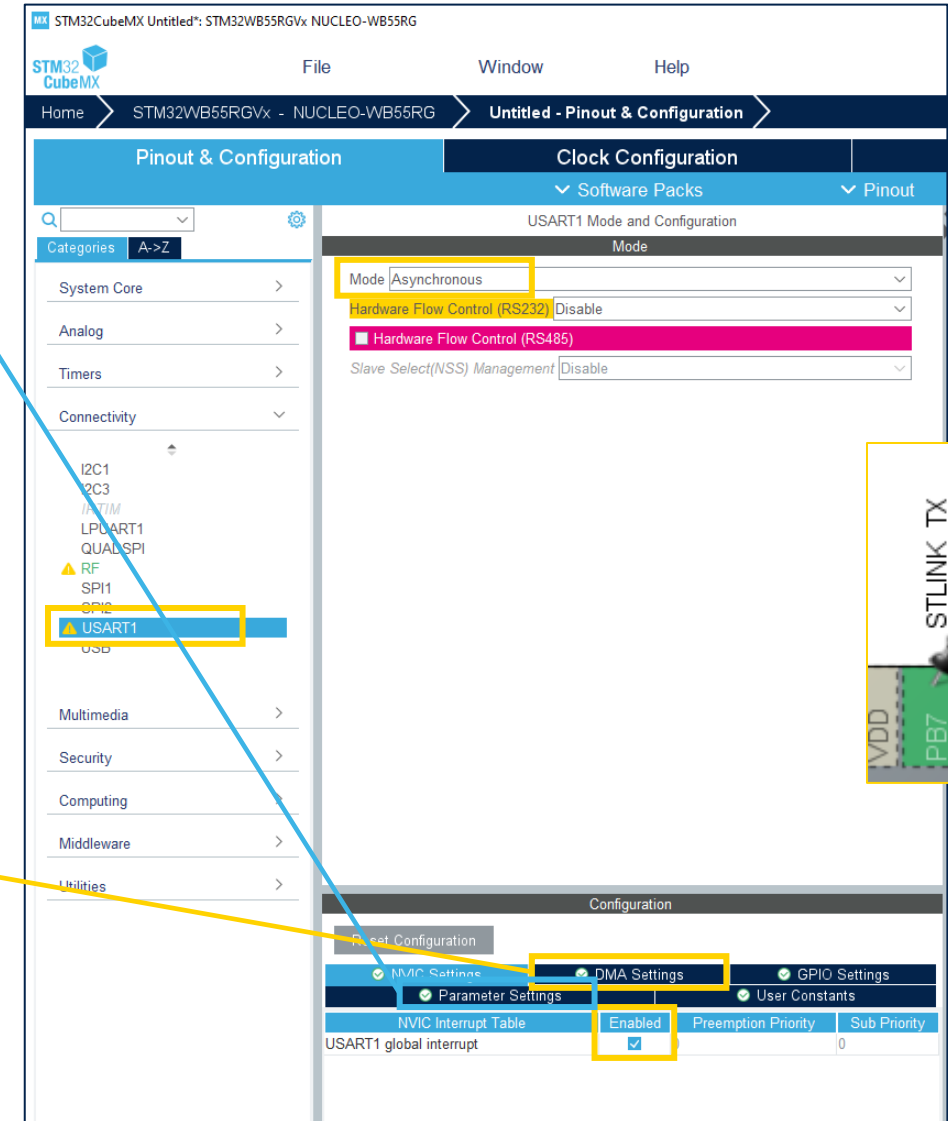
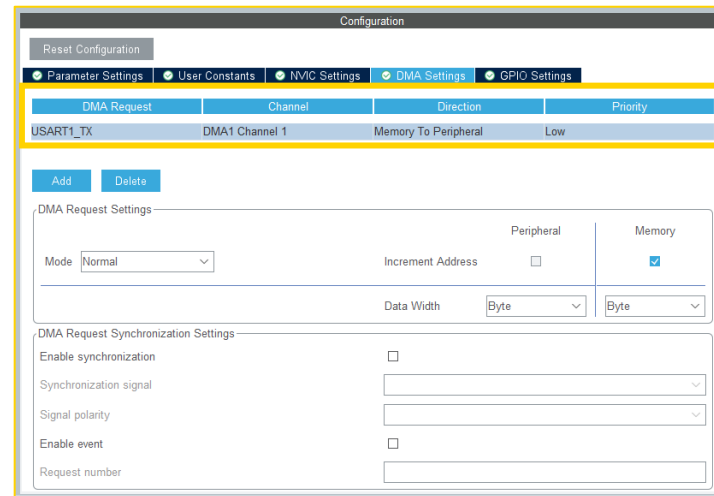
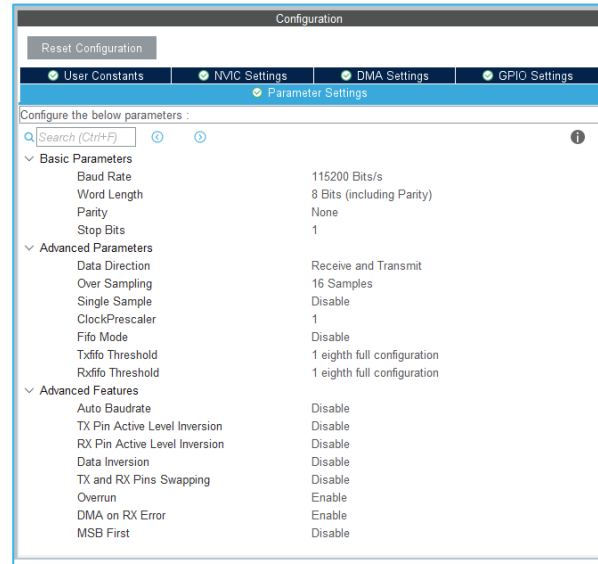
- Select RF under Connectivity Tab.
- Tick Active RF1

The screenshot shows the STM32CubeMX interface for configuring a BLE project. The 'Connectivity' tab is selected in the left sidebar, and the 'RF' option is highlighted. In the main configuration area, the 'RF Mode and Configuration' section is visible, where the 'Activate RF1' checkbox is checked and highlighted with a yellow box. Below it, the 'External PA transmit control' option is also visible. A callout box on the right shows a physical component labeled 'RF_RF1' with pins for VDD, RF1, and VSSRF.

Pin Na...	Signal on ...	GPIO outp...	GPIO mode	GPIO Pull...	Maximum ...	Fast Mode	User Label	Modifie
RF1	RF_RF1	n/a	n/a	n/a	n/a	n/a		<input type="checkbox"/>

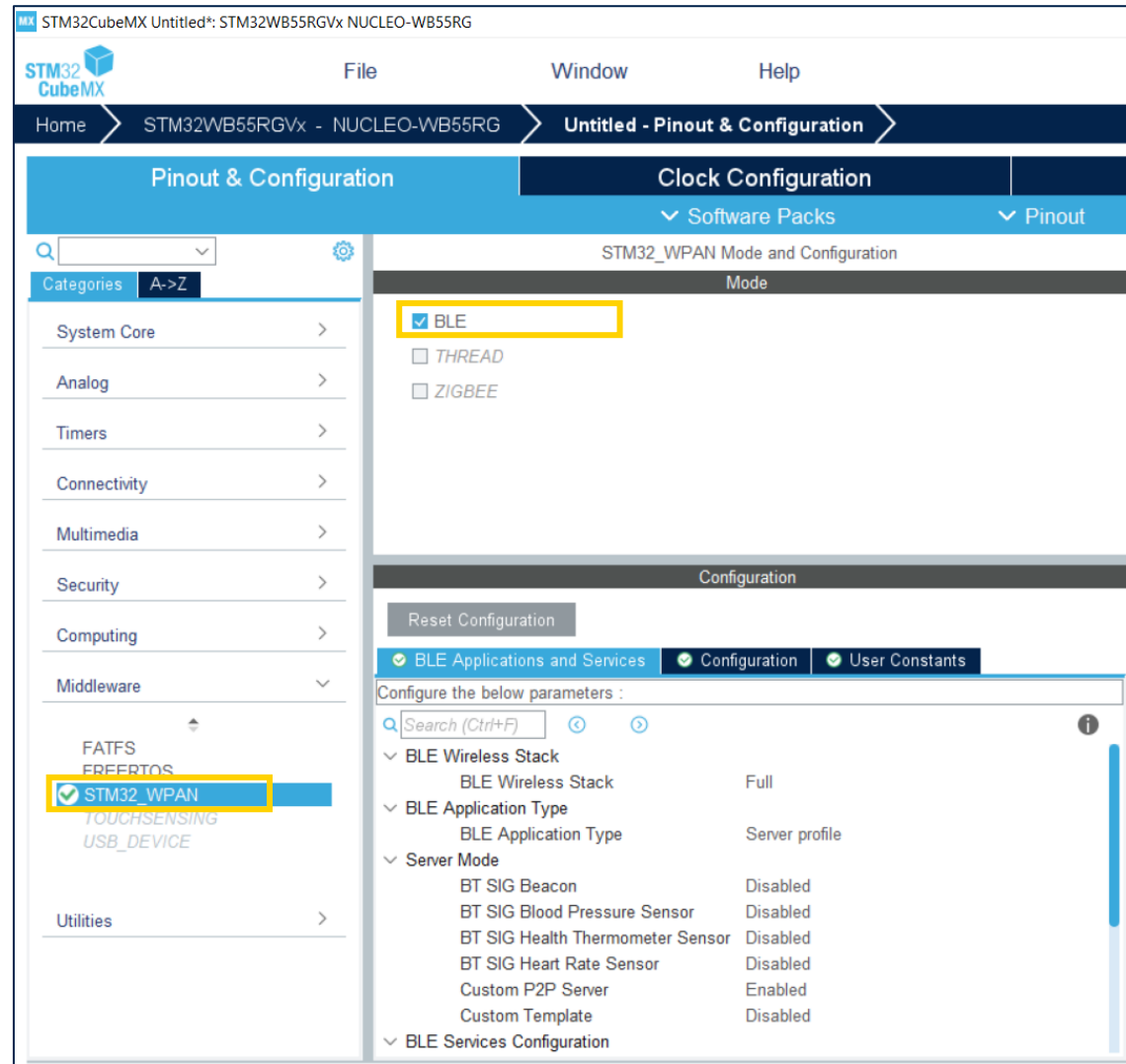
Making simple BLE project Step #14

- Select USART1 under Connectivity Tab.
- Mode : Asynchronous
- Disable Hardware Flow Control
- Enable USART1 global interrupt in NVIC Setting tab.
- DMA : USART1_TX, Normal, Memory to Peripheral, Width(Byte)



Making simple BLE project Step #15

- Select STM32_WPAN under Middleware Tab.
- Tick BLE.



Making simple BLE project

Step #16

- Select BLE Applications and Services Tab
- BLE Wireless Stack : Full
- BLE Application Type : Server profile
- Custom P2P Server : Disabled
- Custom Template : Enabled

Configure the below parameters :

Parameter	Value
BLE Wireless Stack	Full
BLE Application Type	Server profile
Server Mode	
BT SIG Beacon	Disabled
BT SIG Blood Pressure Sensor	Disabled
BT SIG Health Thermometer Sensor	Disabled
BT SIG Heart Rate Sensor	Disabled
Custom P2P Server	Disabled
Custom Template	Enabled
BLE Services Configuration	
The device needs to support the Peripheral Role	1
The device needs to support the Central Role	0
BLE_CFG_SVC_MAX_NBR_CB	7
BLE_CFG_CLT_MAX_NBR_CB	0

Making simple BLE project

Step #17

- Select Configuration Tab
- CFG_DEBUG_TRACE_UART : hw_uart1
- CFG_HW_USART1_ENABLE : Enabled
- CFG_HW_USART1_DMA_TX_SUPPORTED : Enabled
- CFG_DEBUG_BLE_TRACE : Enabled
- CFG_DEBUG_APP_TRACE : Enabled
- CFG_DEBUG_TRACE_LIGHT : Enabled
- DBG_TRACE_USE_CIRCULAR_QUEUE : Disabled
- CFG_IO_CAPABILITY : No input, no output (0x03)

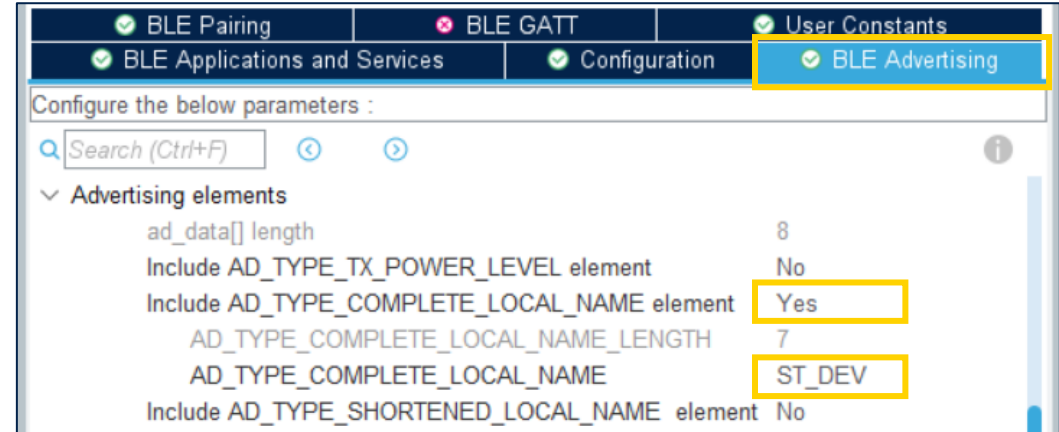
The screenshot shows the Configuration tab in the STM32CubeIDE software. The 'Configuration' tab is selected and highlighted in yellow. The parameters are organized into sections: HW UART, Generic parameters, and Application parameters. Several parameters are highlighted with yellow boxes and numbered 1 through 6, corresponding to the list in the previous block.

Parameter	Value	Number
CFG_HW_LPUART1_ENABLED	Disabled	
CFG_HW_LPUART1_DMA_TX_SUPPORTED	Disabled	1
CFG_HW_USART1_ENABLE	Enabled	6
CFG_HW_USART1_DMA_TX_SUPPORTED	Enabled	
CFG_HW_RESET_BY_FW	Enabled	
CFG_USE_SMPS	Disabled	
CFG_LPM_SUPPORTED	Disabled	
CFG_DEBUGGER_SUPPORTED	Enabled	
CFG_DEBUG_BLE_TRACE	Enabled	3
CFG_DEBUG_APP_TRACE	Enabled	
CFG_DEBUG_TRACE_LIGHT	Enabled	
CFG_DEBUG_TRACE_FULL	Disabled	
DBG_TRACE_USE_CIRCULAR_QUEUE	Disabled	4
DBG_TRACE_MSG_QUEUE_SIZE	4096	
MAX_DBG_TRACE_MSG_SIZE	1024	
CFG_TX_POWER	-0.15dBm (0x18)	
CFG_DEBUG_TRACE_UART	hw_uart1	2
CFG_CONSOLE_MENU	No UART selected. You need to activate LP...	
CFG_ADV_BD_ADDRESS	0x000000000000	
CFG_FAST_CONN_ADV_INTERVAL	80	
CFG_FAST_CONN_ADV_INTERVAL	100	
CFG_LP_CONN_ADV_INTERVAL	1000	
CFG_LP_CONN_ADV_INTERVAL	2500	
CFG_IO_CAPABILITY	No input, no output (0x03)	5
CFG_MITM_PROTECTION	MITM protection required (0x01)	

Making simple BLE project

Step #18

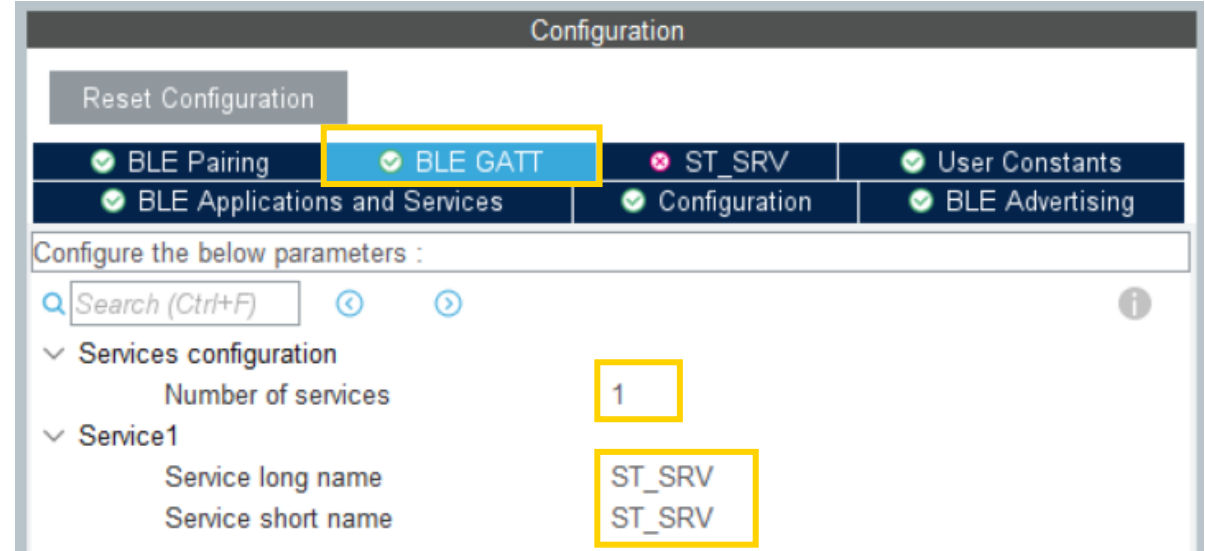
- Select BLE Advertising Tab
- Include AD_TYPE_COMPLETE_LOCAL_NAME : Yes
- Name your device with a unique ID
- Example : XX_DEV



Making simple BLE project

Step #19

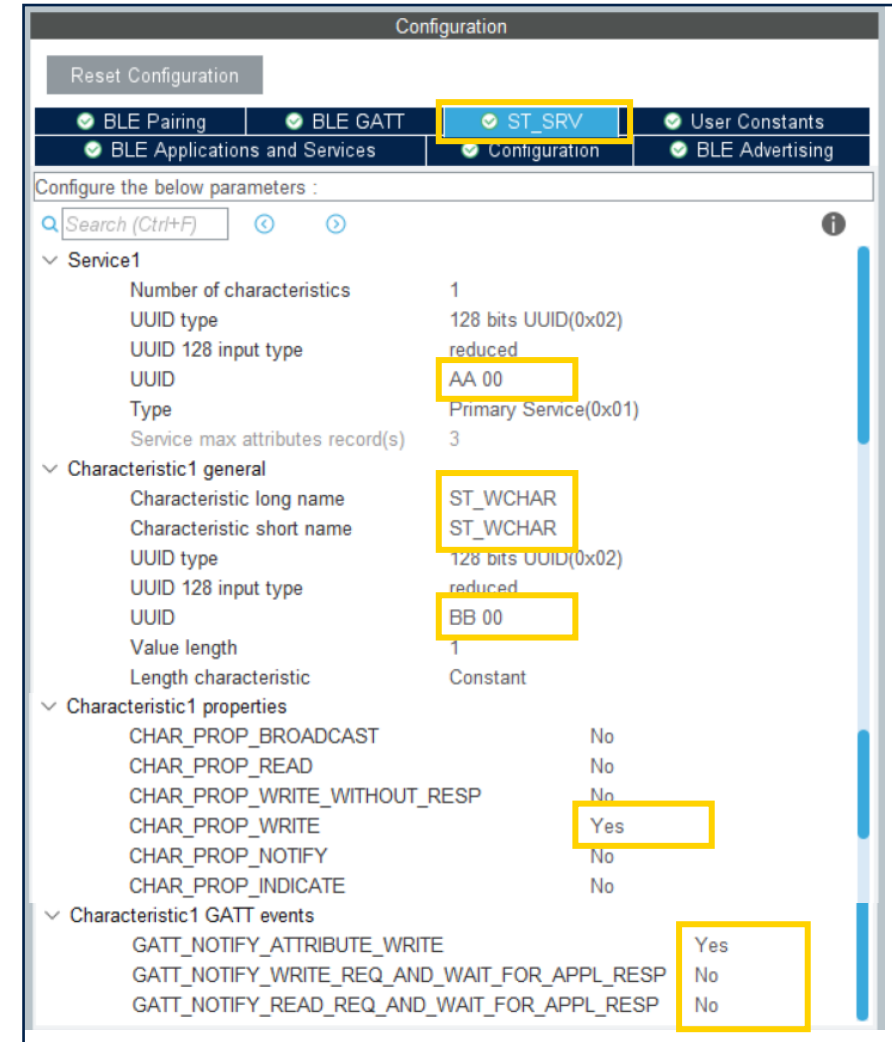
- Select BLE GATT tab.
- Set number of services to 1
- Name your BLE service, both a long and short name.
- For example : XX_SRV



Making simple BLE project

Step #20

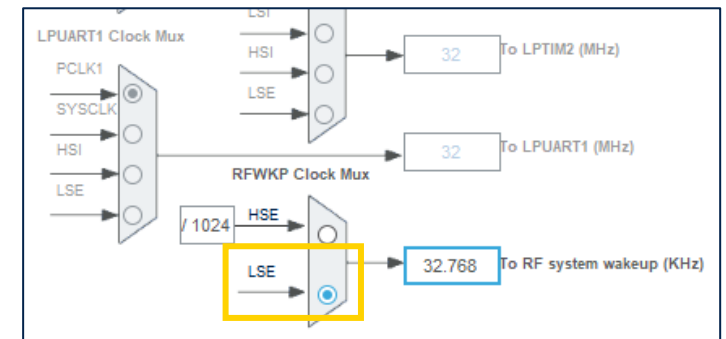
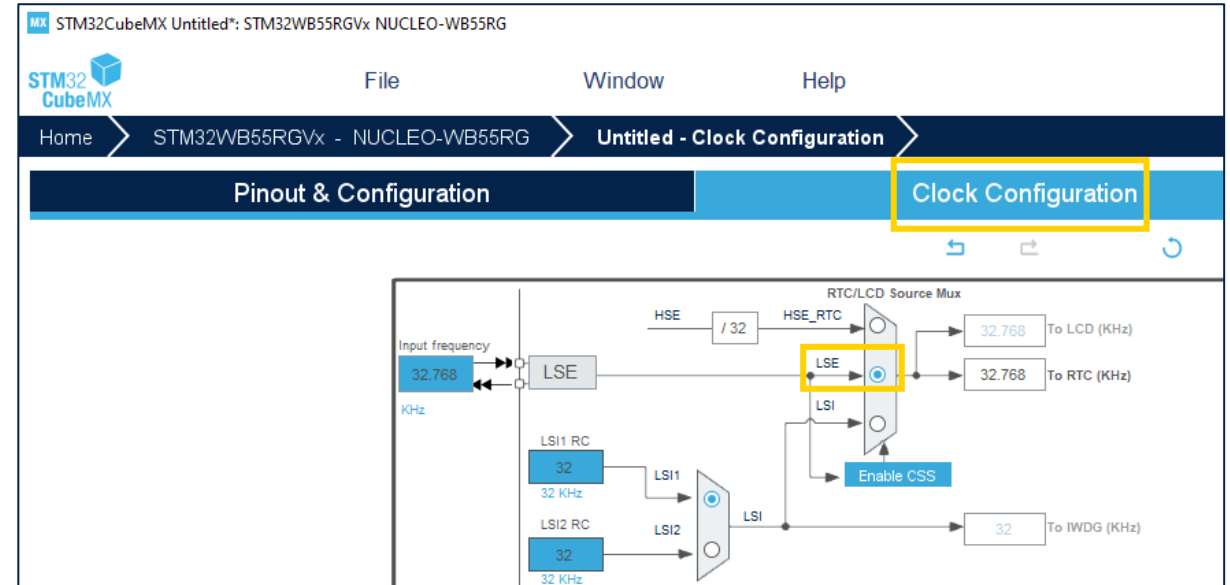
- Select “ST_SRV” tab. (The exact name depends on your previous selection.)
- UUID 0xAA00 is for Primary Service
- Name your characteristic “XX_WCHAR” as both long and short name.
- UUID 0xBB00 is for Characteristic1.
- CHAR_PROP_WRITE : Yes
- GATT_NOTIFY_ATTRIBUTE_WRITE : Yes
- GATT_NOTIFY_WRITE_REQ_AND_WAIT_FOR_APPL_RESP : No
- GATT_NOTIFY_READ_REQ_AND_WAIT_FOR_APPL_RESP : No
- GATT_NOTIFY_NOTIFICATION_COMPLETION : No



Making simple BLE project

Step #21

- Go to Clock Configuration Tab.
- Set RTC clock to LSE.
- Set RFWKP clock to LSE.



Making simple BLE project

Step #22

- Go to Project Manager
- In Project Tab.
 - Name for project.
 - Set path for saving new project.
 - IDE : STM32CubeIDE
- In Code Generator Tab.
 - Tick “Copy only the necessary library files”

The screenshot shows the STM32CubeMX Project Manager configuration window. The 'Project Manager' tab is active, showing the following settings:

- Project Name: Simple_BLE_Project
- Project Location: C:\Hands-on
- Application Structure: Advanced
- Toolchain Folder Location: C:\Hands-on\Simple_BLE_Project\
- Toolchain / IDE: STM32CubeIDE
- Generate Under Root:

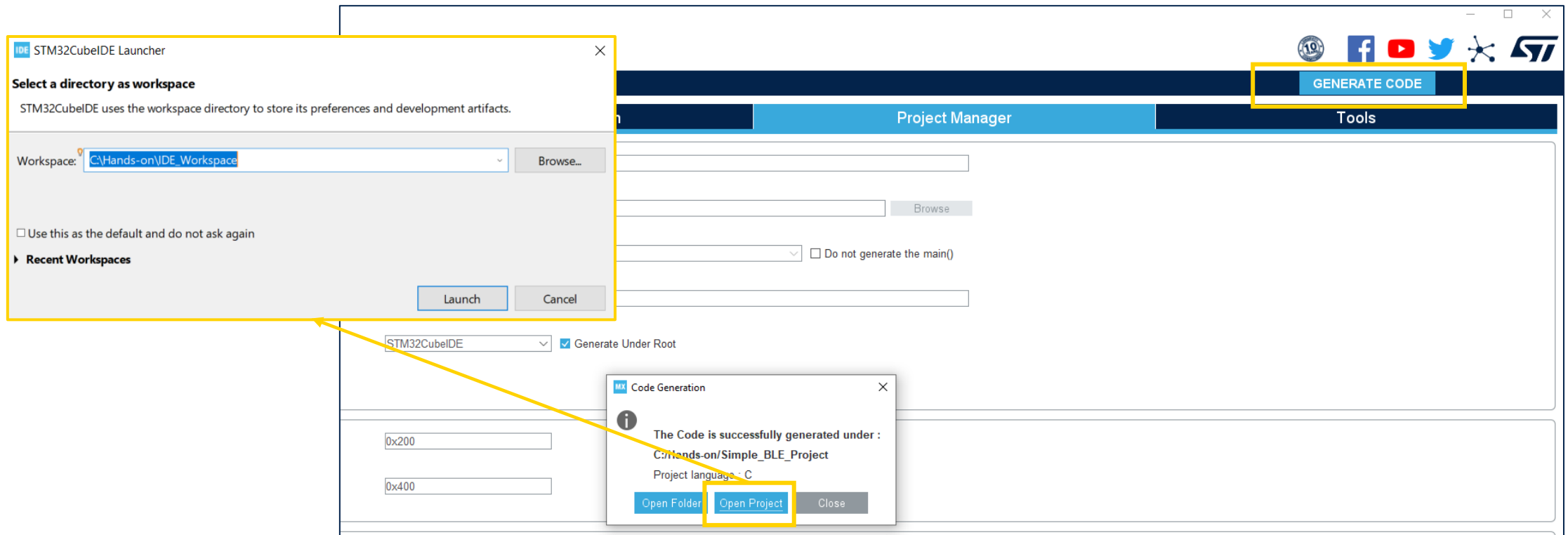
The 'Code Generator' tab is also visible, showing the following settings:

- Copy all used libraries into the project folder:
- Copy only the necessary library files:
- Add necessary library files as reference in the toolchain project configuration file:

A yellow box highlights the 'Copy only the necessary library files' option in the Code Generator tab.

Making simple BLE project Step #23

- Click GENERATE CODE.
- Click Open Project.
- Launch STM32CubeIDE after set the path of workspace

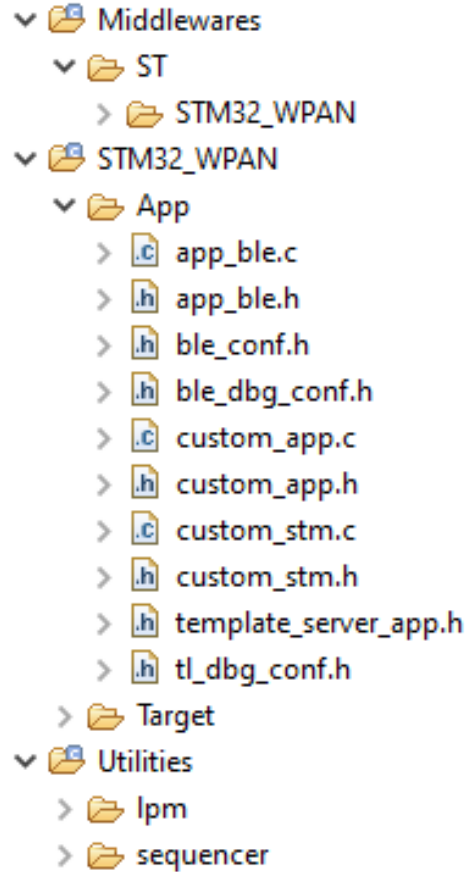
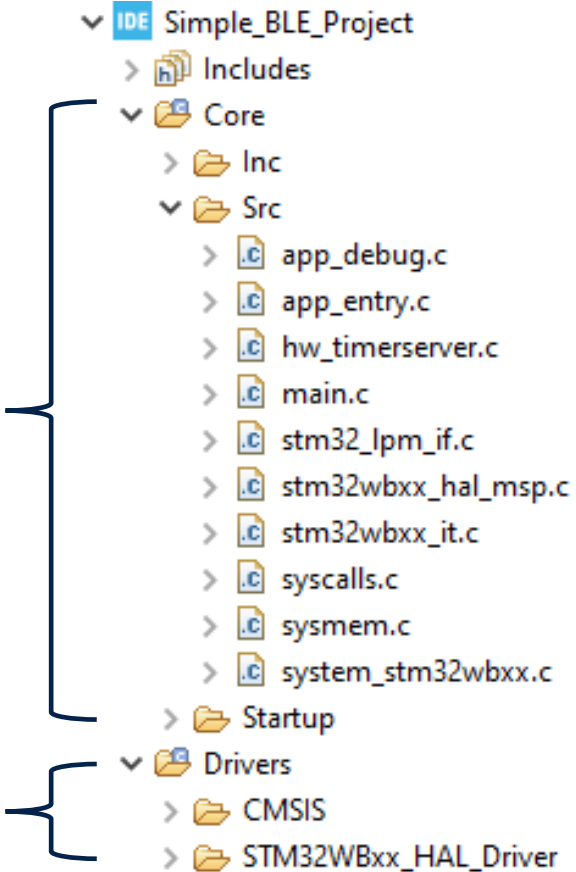


Making simple BLE project

Step #24

Application and system initialization
+ Interrupt service routine

CMSIS and Hal Drivers



BLE Middleware

BLE application files.
Configuration, BLE service
definition. Event handlers etc.

Utilities

- Low power manager
- Sequencer

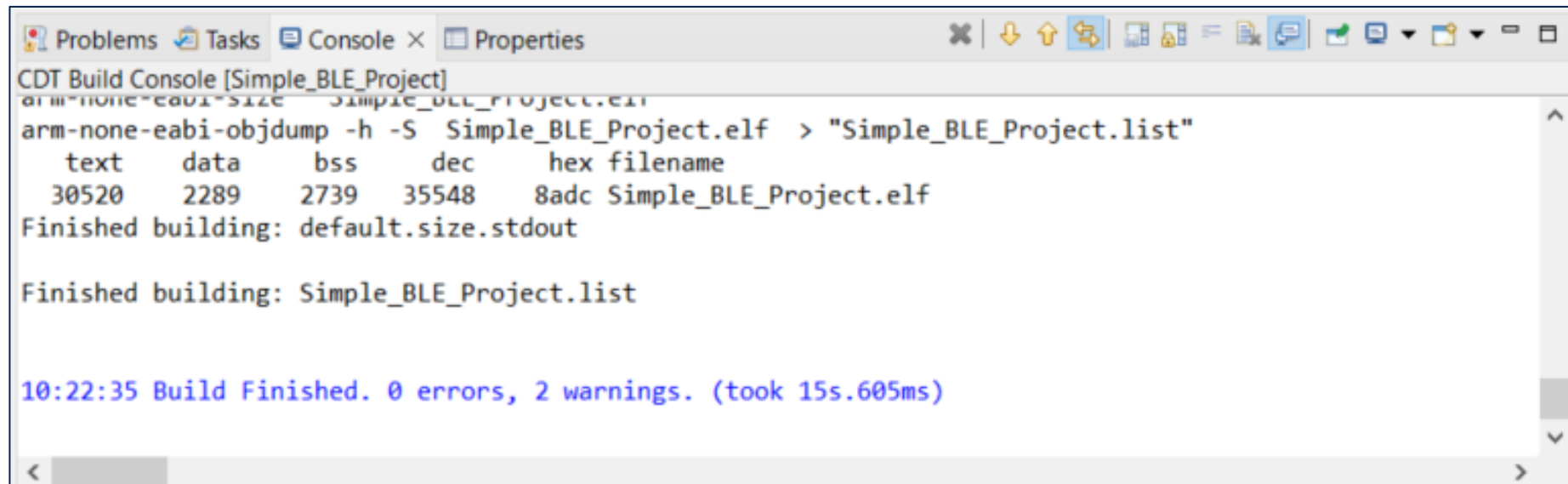


Making simple BLE project

Step #25

- Build Project.

- Ctrl + B


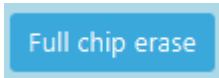


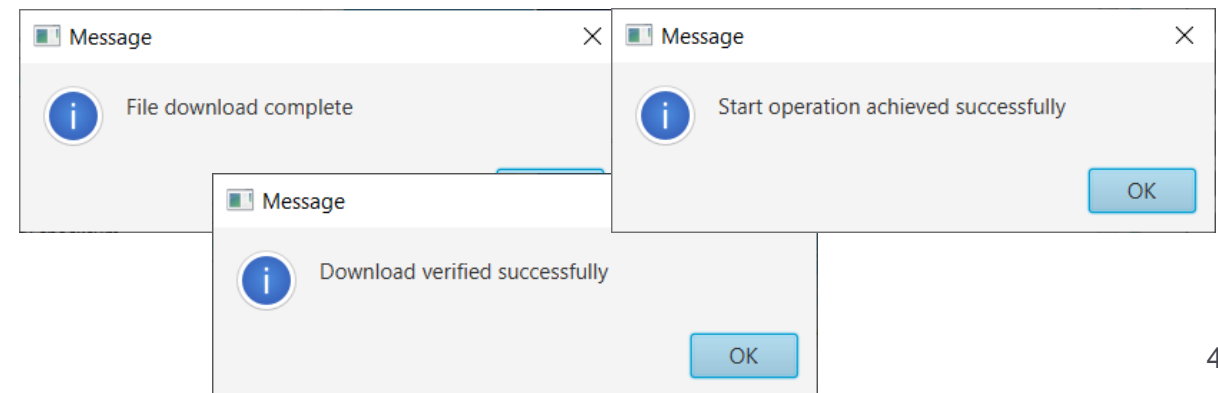
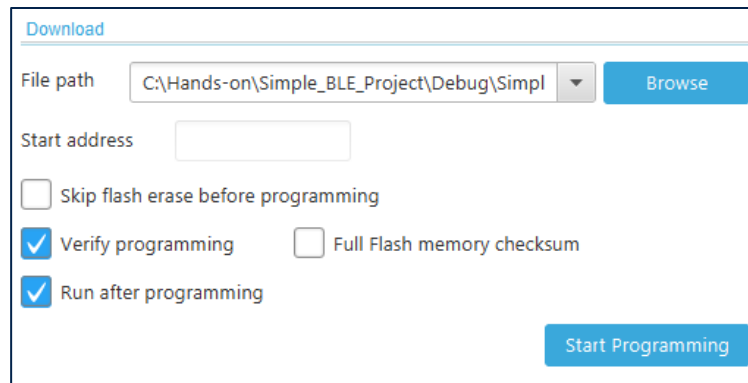
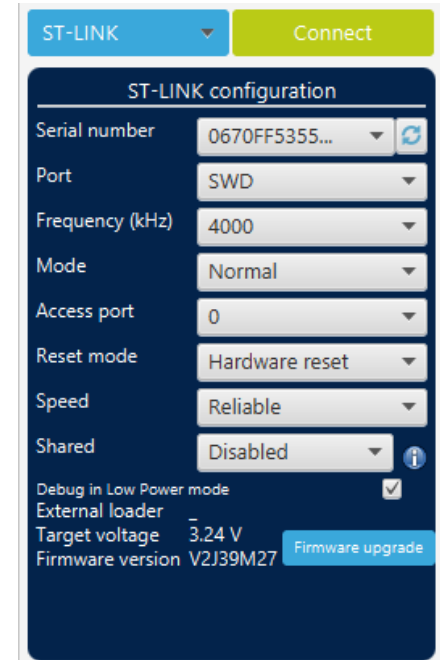
```
CDT Build Console [Simple_BLE_Project]
arm-none-eabi-size Simple_BLE_Project.elf
arm-none-eabi-objdump -h -S Simple_BLE_Project.elf > "Simple_BLE_Project.list"
  text  data  bss  dec  hex filename
 30520  2289  2739  35548  8adc Simple_BLE_Project.elf
Finished building: default.size.stdout

Finished building: Simple_BLE_Project.list

10:22:35 Build Finished. 0 errors, 2 warnings. (took 15s.605ms)
```

Making simple BLE project Step #26

- Download FW using CubeProgrammer.
- Connect via SWD. (with hardware reset mode)
- Click “Erasing & Programming” menu. → 
- Click “Full chip erase” first. → 
- Load firmware file from below path.
 - `C:\Hands-on\Simple_BLE_Project\Debug\Simple_BLE_Project.elf`
- Click “Start Programming” with following flags.

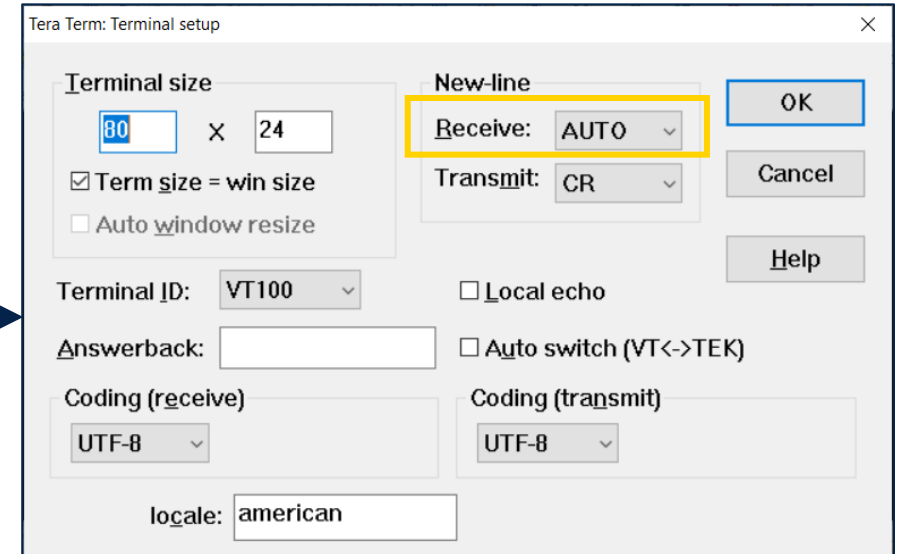
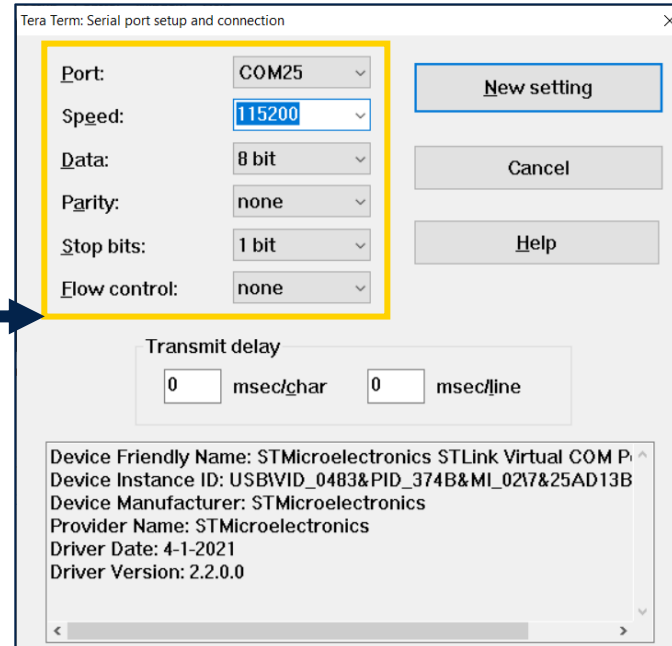
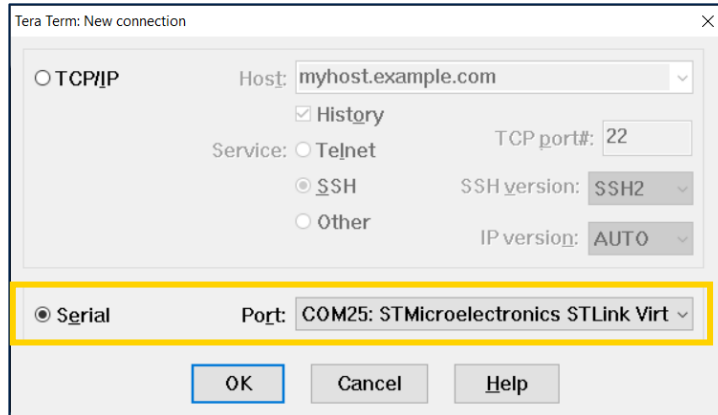


Making simple BLE project

Step #27

- **Launch TeraTerm to see the UART message**

- ✓ Baudrate:115200, 8bit, No parity, 1bit, No flow control
- ✓ New-line: Receive (AUTO)



Making simple BLE project

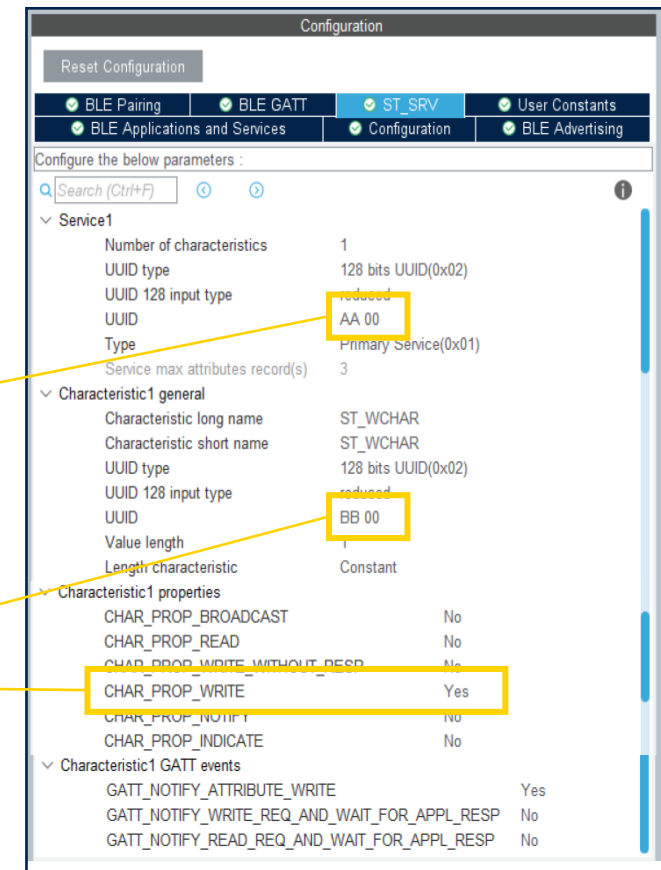
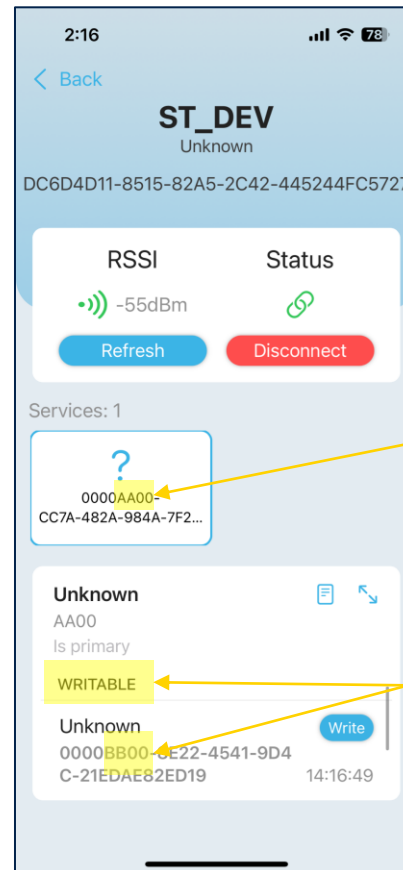
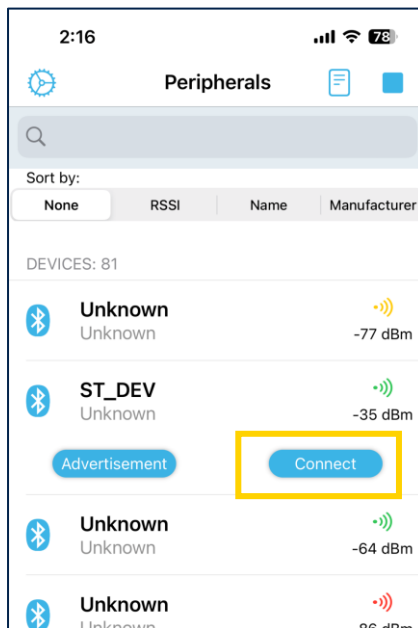
Step #28

- Press “Reset SW4”.
- You can see the characteristics that you added on boot log.

```
COM49 - Tera Term VT
File Edit Setup Control Window Help
Wireless Firmware version 1.16.0
Wireless Firmware build 4
FUS version 1.2.0
>>= SHCI_SUB_EUT_CODE_READY
>>= WIRELESS_FW_RUNNING
>>= DBGMCU_GetRevisionID= 2003
>>= DBGMCU_GetDeviceID= 495
Success: SHCI_C2_BLE_Init command
==>> Start Ble_Hci_Gatt_Init function
Success: hci_reset command
Success: aci_hal_write_config_data command - CONFIG_DATA_PUBADDR_OFFSET
Public Bluetooth Address: 00:80:e1:27:ae:17
Success: aci_hal_write_config_data command - CONFIG_DATA_IR_OFFSET
Success: aci_hal_write_config_data command - CONFIG_DATA_ER_OFFSET
Success: aci_hal_set_tx_power_level command
Success: aci_gatt_init command
Success: aci_gap_init command
Success: hci_le_set_default_phy command
Success: aci_gap_set_io_capability command
Success: aci_gap_set_authentication_requirement command
==>> End Ble_Hci_Gatt_Init function
Success: aci_gatt_add_service command: ST_SRU
Success: aci_gatt_add_char command: ST_UCUR
Success: aci_hal_set_radio_activity_mask command
==>> aci_gap_set_discoverable - Success
==>> Success: Start Fast Advertising
```

Making simple BLE project Step #29

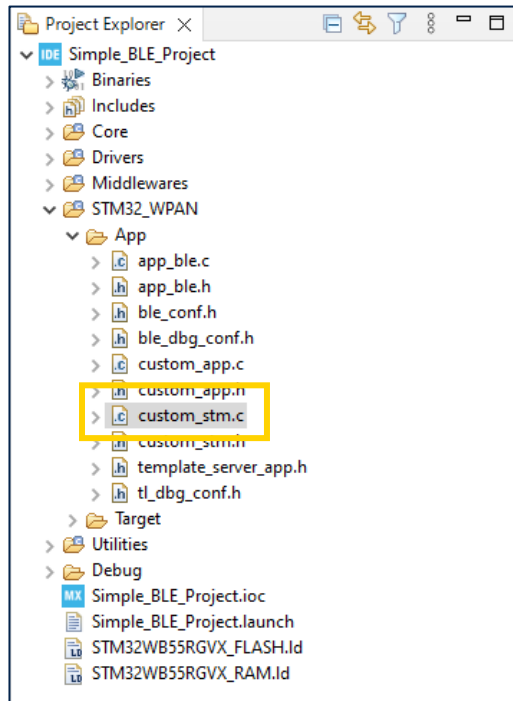
- Check the custom service & characteristic using BLEToolBox



Making simple BLE project

Step #30

- Add the green LED On/Off code.
(Simple_BLE_Project/STM32_WPAN/app/custom_stm.c)



```
/* USER CODE BEGIN Includes */  
#include "main.h"  
/* USER CODE END Includes */
```

```
else if (attribute_modified->Attr_Handle == (CustomContext.CustomSt_WcharHdle + CHARACTERISTIC_VALUE_ATTRIBUTE_OFFSET))  
{  
    return_value = SVCCTL_EvtAckFlowEnable;  
    /* USER CODE BEGIN CUSTOM STM Service 1 Char 1 ACT GATT ATTRIBUTE MODIFIED VSEVT_CODE */  
    APP_DBG_MSG(" Write_Cmd : 0x%02x \n\r", attribute_modified->Attr_Data[0]);  
    if(attribute_modified->Attr_Data[0] == 0x00)  
    {  
        HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);  
    }  
    else if(attribute_modified->Attr_Data[0] == 0x01)  
    {  
        HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);  
    }  
    /* USER CODE END CUSTOM STM Service 1 Char 1 ACT GATT ATTRIBUTE MODIFIED VSEVT_CODE */
```

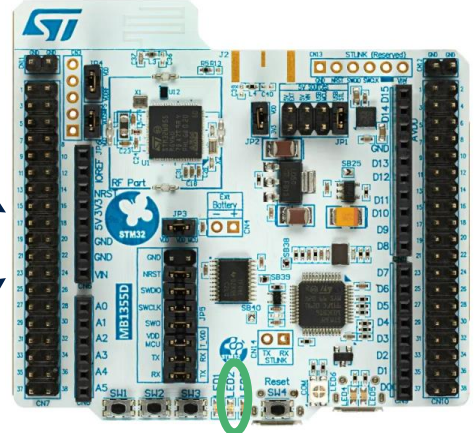
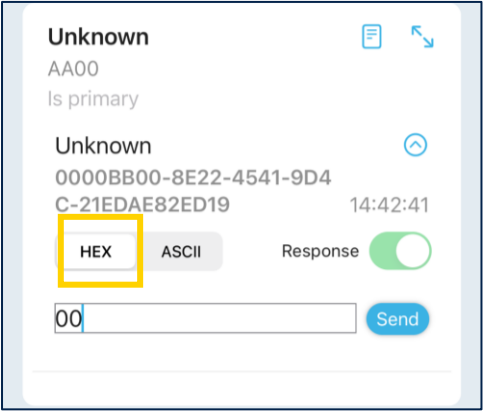
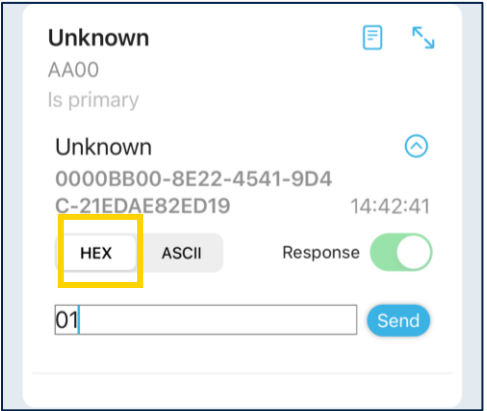
Making simple BLE project Step #31

- Build Project.



- Download new firmware again.
(Refer, Hands-on Step #26)

- Write data to the characteristic.
0x00(hex) : Green LED off
0x01(hex) : Green LED on



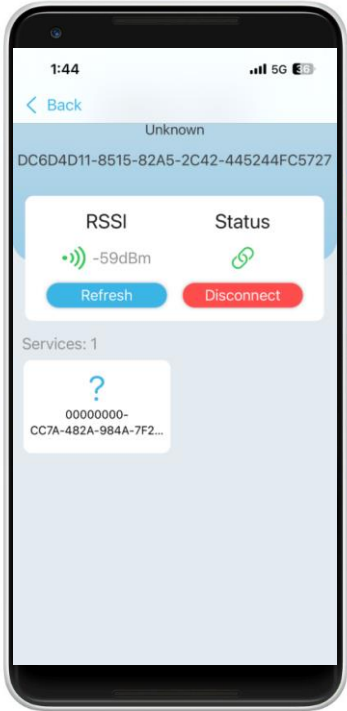
Green LED

```
COM14 - Tera Term VT
File Edit Setup Control Window Help
Public Bluetooth Address: 00:80:e1:26:40:e8
Success: aci_hal_write_config_data command - CONFIG_DATA_IR_OFFSET
Success: aci_hal_write_config_data command - CONFIG_DATA_ER_OFFSET
Success: aci_hal_set_tx_power_level command
Success: aci_gatt_init command
Success: aci_gap_init command
Success: aci_le_set_default_phy command
Success: aci_gap_set_io_capability command
Success: aci_gap_set_authentication_requirement command
==>> End Ble_Hci_Gatt_Init function
Success: aci_gatt_add_service command: ST_SRU
Success: aci_gatt_add_char command : ST_WCHAR
Success: aci_hal_set_radio_activity_mask command
==>> aci_gap_set_discoverable - Success
==>> Success: Start Fast Advertising
==>> HCI_LE_CONNECTION_COMPLETE_SUBEVENT_CODE - Connection handle: 0x801
==>> - Connection established with Central: e:75:68:00:47:28:fb
- Connection Interval: ms
- Connection latency: 0
- Supervision Timeout: 720 ms
Write_Cnd : 0x01
Write_Cnd : 0x00
Write_Cnd : 0x01
```

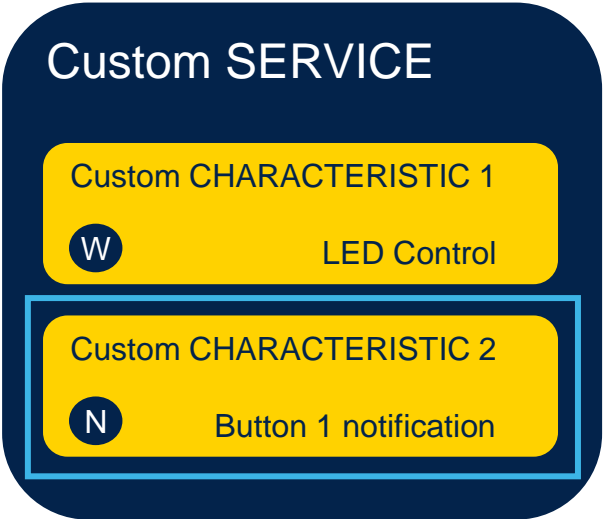
Making simple BLE project

Sending data to phone

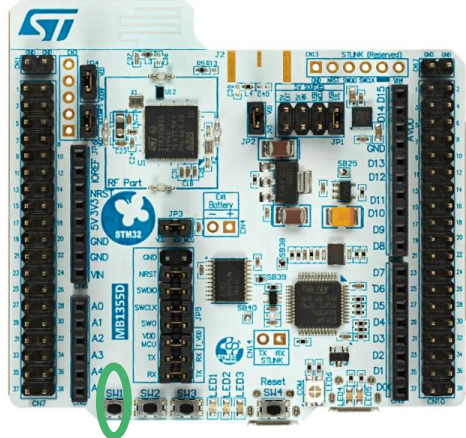
Central **GATT Client**



- Slave will send notification data to phone after increasing number if button_1 is pressed.



Peripheral **GATT Server**

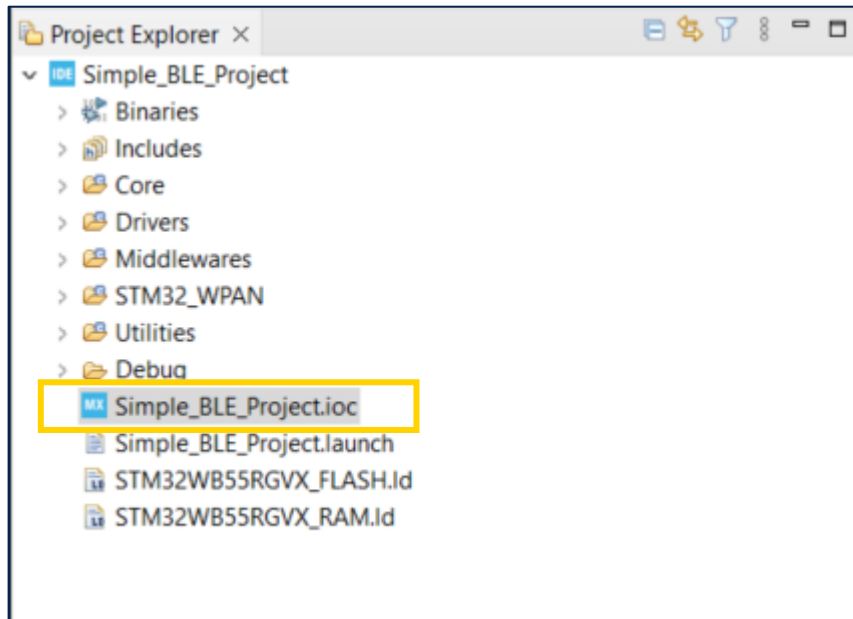


Button 1

Making simple BLE project

Step #32

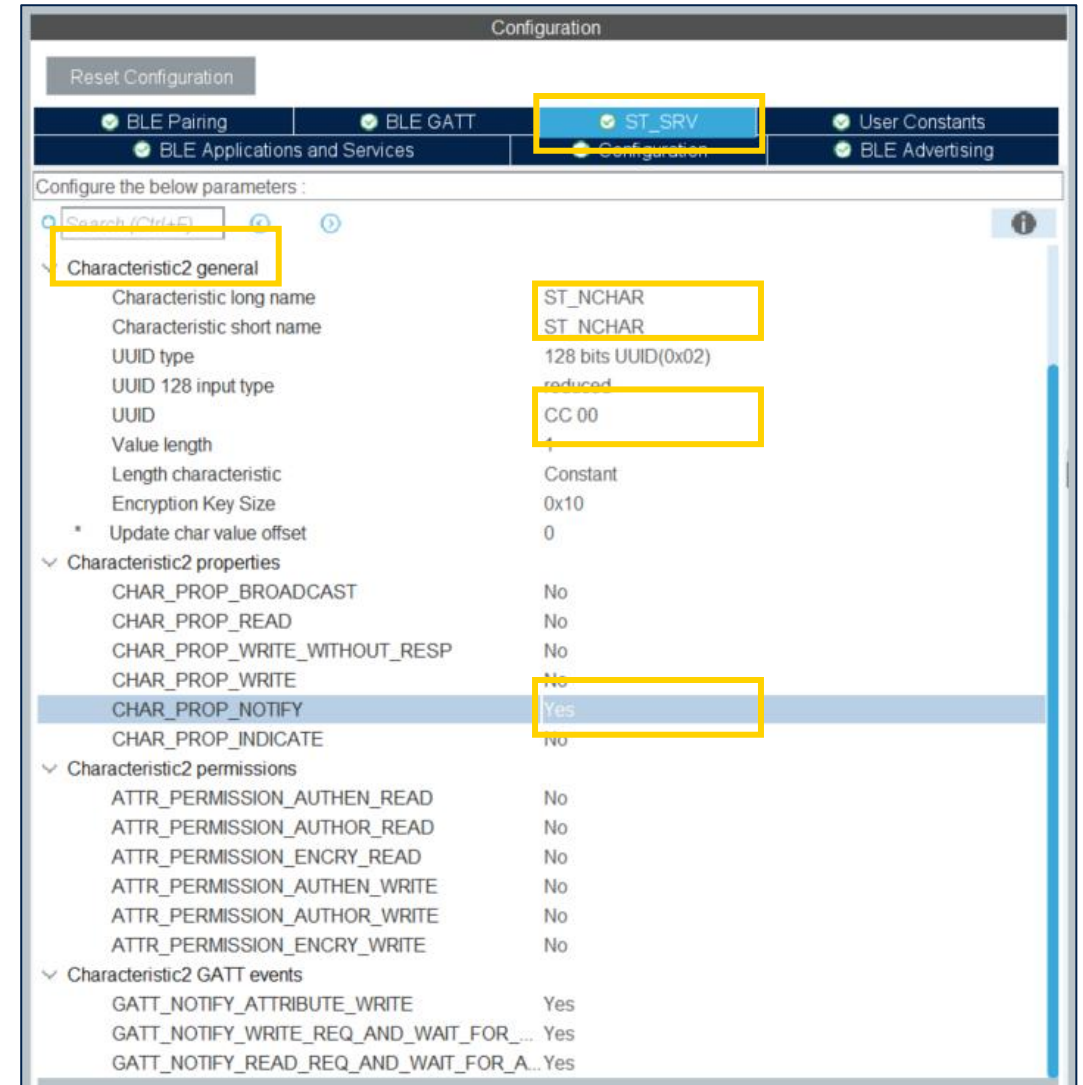
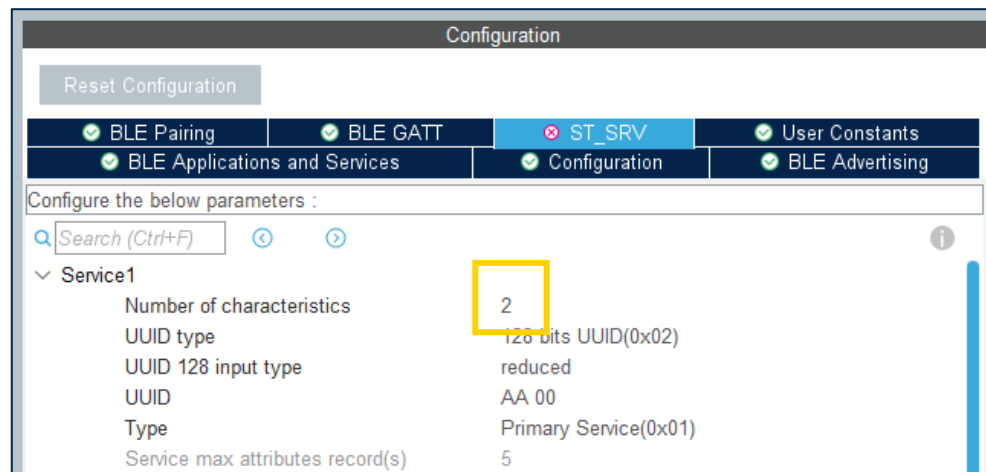
1. Double click “Simple_BLE_Project.ioc” on CubeIDE



Making simple BLE project

Step #33

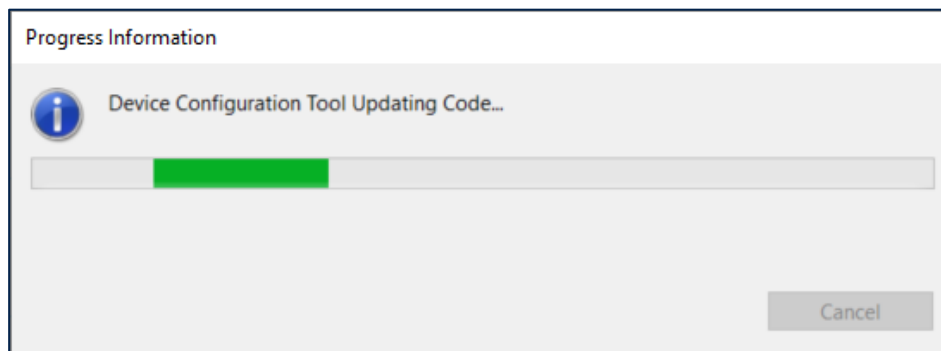
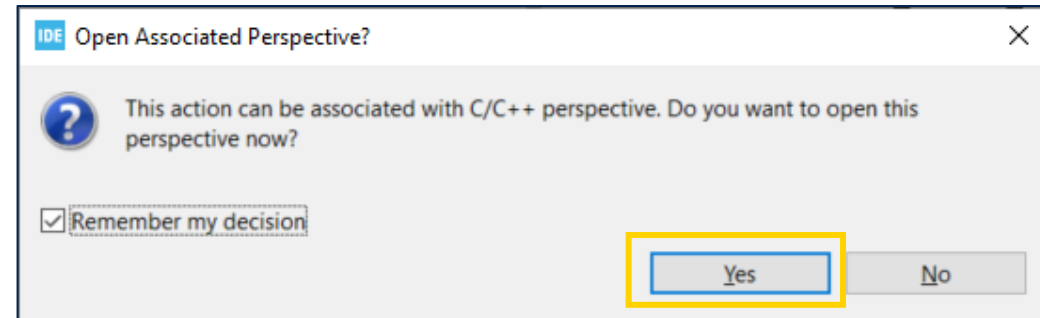
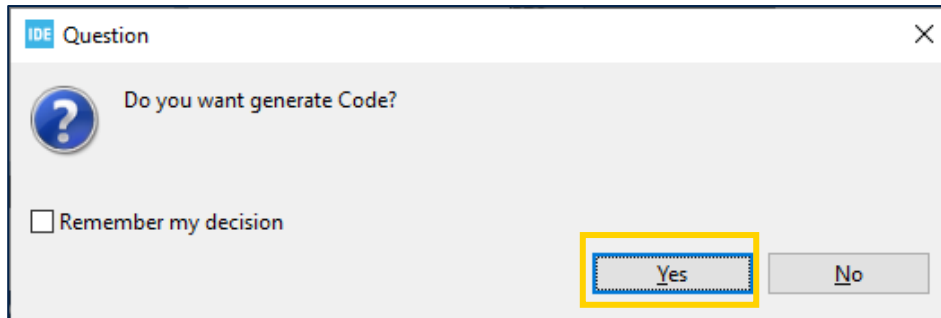
- Select “ST_SRV” tab in STM32WPAN Tab. (The exact name depends on your previous naming.)
- Add a second characteristic with notification properties.
- Set Number of characteristic to 2.
- Name your second characteristic “XX_NCHAR” as both log and short name.
- UUID 0xCC00 is for Characteristic2.
- CHAR_PROP_NOTIFY → Yes



Making simple BLE project

Step #34

- Save all (Ctrl+Shift+S)
- Code will be updated

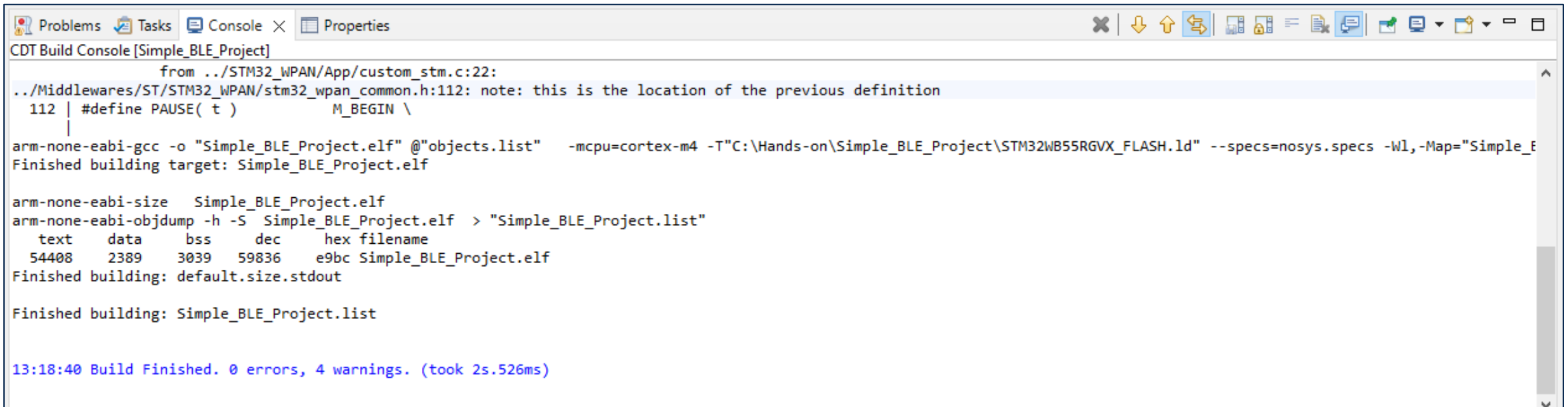


Making simple BLE project

Step #35

- Build Project again.

- Ctrl + B




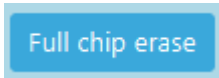
```
CDT Build Console [Simple_BLE_Project]
from ../STM32_WPAN/App/custom_stm.c:22:
../Middlewares/ST/STM32_WPAN/stm32_wpan_common.h:112: note: this is the location of the previous definition
112 | #define PAUSE( t )          M_BEGIN \
    |
arm-none-eabi-gcc -o "Simple_BLE_Project.elf" @"objects.list" -mcpu=cortex-m4 -T"C:\Hands-on\Simple_BLE_Project\STM32WB55RGVX_FLASH.ld" --specs=nosys.specs -Wl,-Map="Simple_E
Finished building target: Simple_BLE_Project.elf

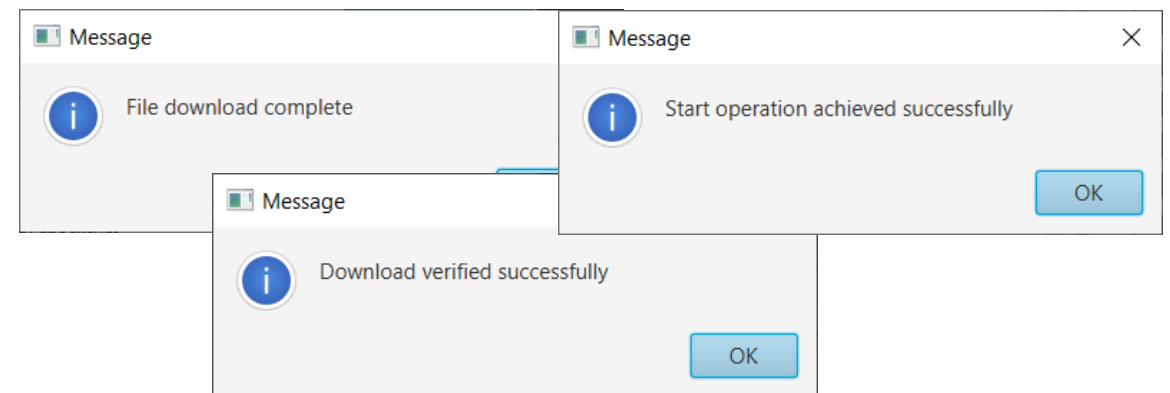
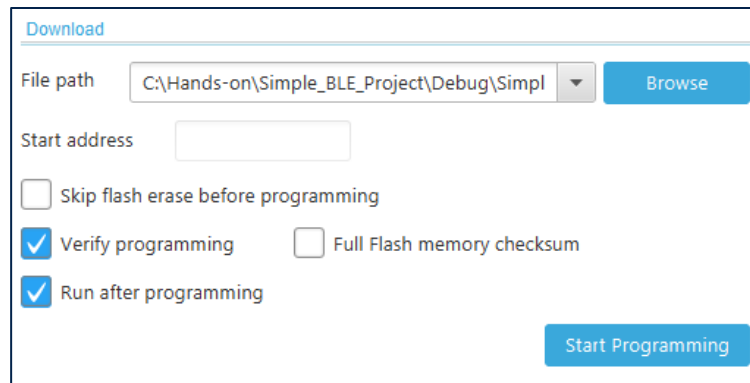
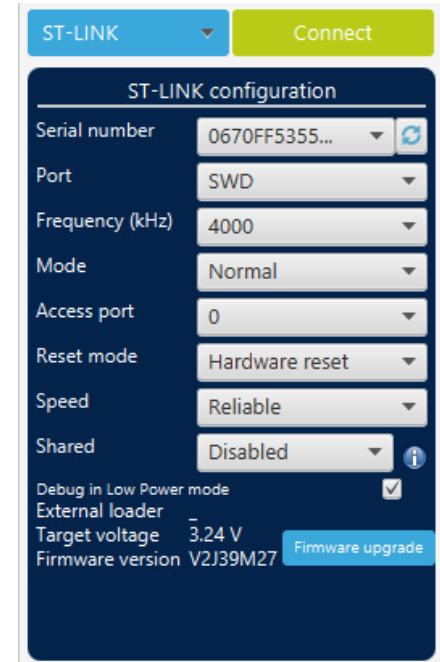
arm-none-eabi-size Simple_BLE_Project.elf
arm-none-eabi-objdump -h -S Simple_BLE_Project.elf > "Simple_BLE_Project.list"
text data bss dec hex filename
54408 2389 3039 59836 e9bc Simple_BLE_Project.elf
Finished building: default.size.stdout

Finished building: Simple_BLE_Project.list

13:18:40 Build Finished. 0 errors, 4 warnings. (took 2s.526ms)
```

Making simple BLE project Step #36

- Download FW using CubeProgrammer.
- Connect via SWD. (with hardware reset mode)
- Click “Erasing & Programming” menu. → 
- Click “Full chip erase” first. → 
- Load firmware file from below path.
 - `C:\Hands-on\Simple_BLE_Project\Debug\Simple_BLE_Project.elf`
- Click “Start Programming” with following flags.

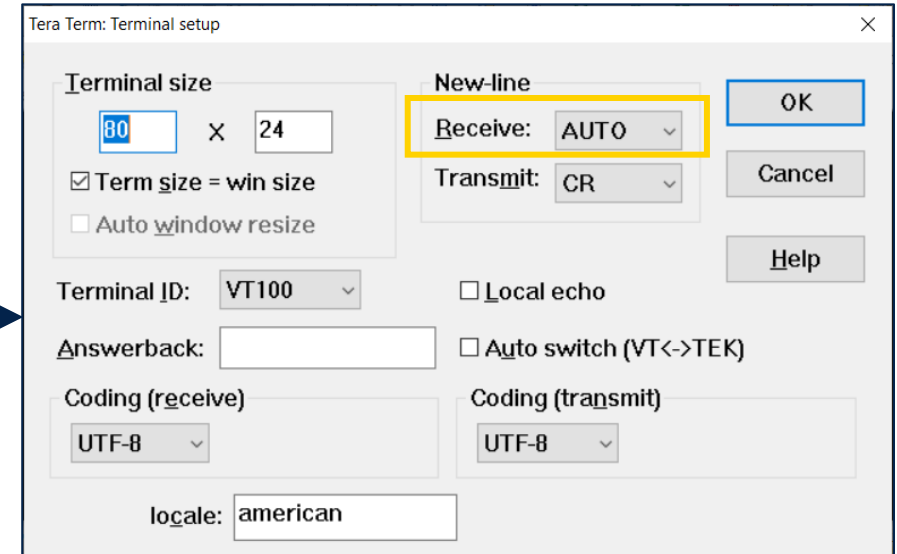
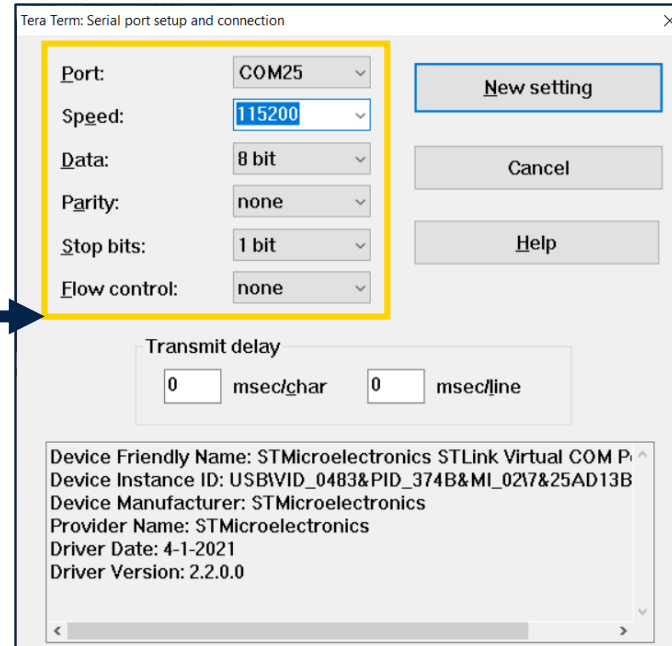
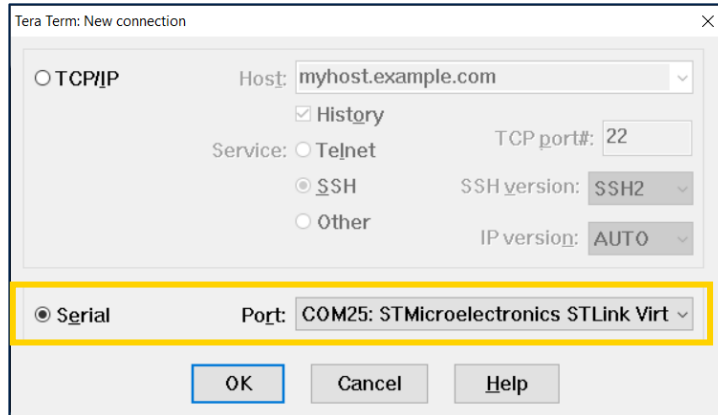


Making simple BLE project

Step #37

- **Executing TeraTerm to see the uart message**

- ✓ Baudrate:115200, 8bit, No parity, 1bit, No flow control
- ✓ New-line: Receive (AUTO)



Making simple BLE project

Step #38

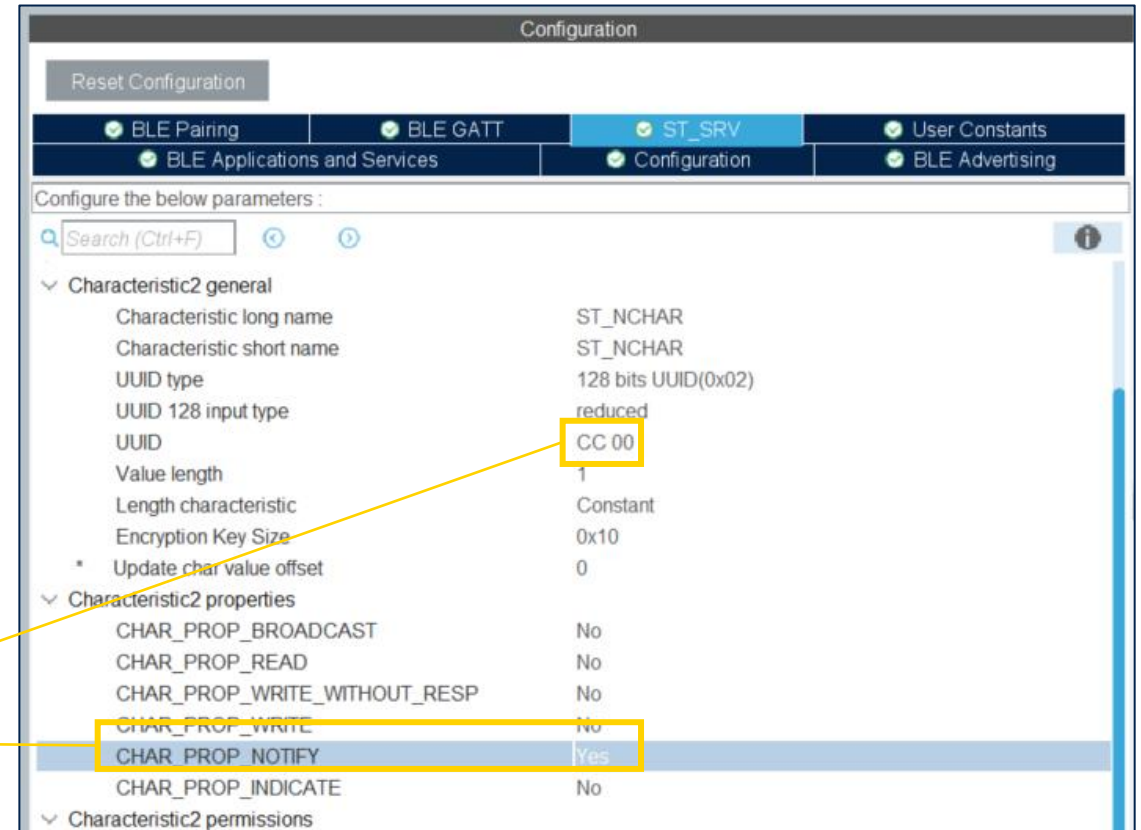
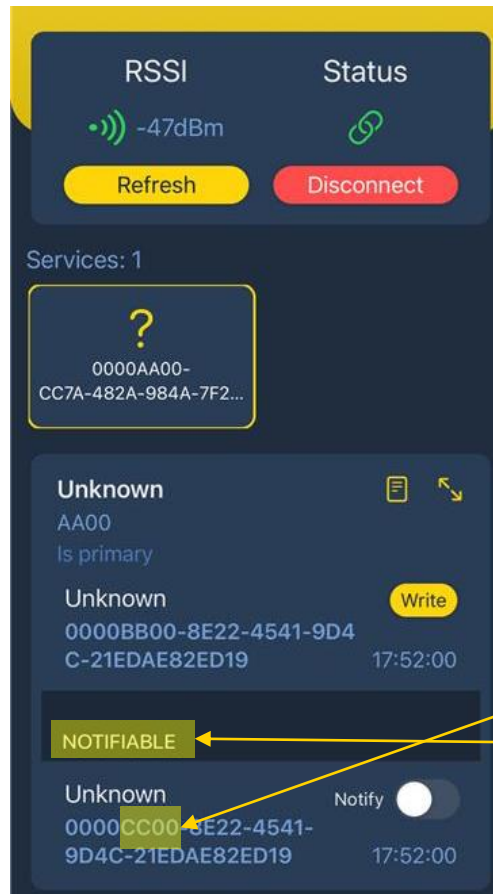
- Press “Reset SW4”.
- You can see the new characteristic that you added.

```
COM49 - Tera Term VT
File Edit Setup Control Window Help
Wireless Firmware version 1.16.0
Wireless Firmware build 4
FUS version 1.2.0
>>== SHCI_SUB_EUT_CODE_READY
>>== WIRELESS_FW_RUNNING
>>== DBGMCU_GetRevisionID= 2003
>>== DBGMCU_GetDeviceID= 495
  Success: SHCI_C2_BLE_Init command
=>> Start Ble_Hci_Gap_Gatt_Init function
  Success: hci_reset command
  Success: aci_hal_write_config_data command - CONFIG_DATA_PUBADDR_OFFSET
  Public Bluetooth Address: 00:80:e1:27:ae:17
  Success: aci_hal_write_config_data command - CONFIG_DATA_IR_OFFSET
  Success: aci_hal_write_config_data command - CONFIG_DATA_ER_OFFSET
  Success: aci_hal_set_tx_power_level command
  Success: aci_gatt_init command
  Success: aci_gap_init command
  Success: hci_le_set_default_phy command
  Success: aci_gap_set_io_capability command
  Success: aci_gap_set_authentication_requirement command
=>> End Ble_Hci_Gap_Gatt_Init function
  Success: aci_gatt_add_service command: ST_SRU
  Success: aci_gatt_add_char command : ST_WCHAR
  Success: aci_gatt_add_char command : ST_NCHAR
  Success: aci_hal_set_radio_activity_mask command
=>> aci_gap_set_discoverable - Success
=>> Success: Start Fast Advertising
```

Making simple BLE project

Step #39

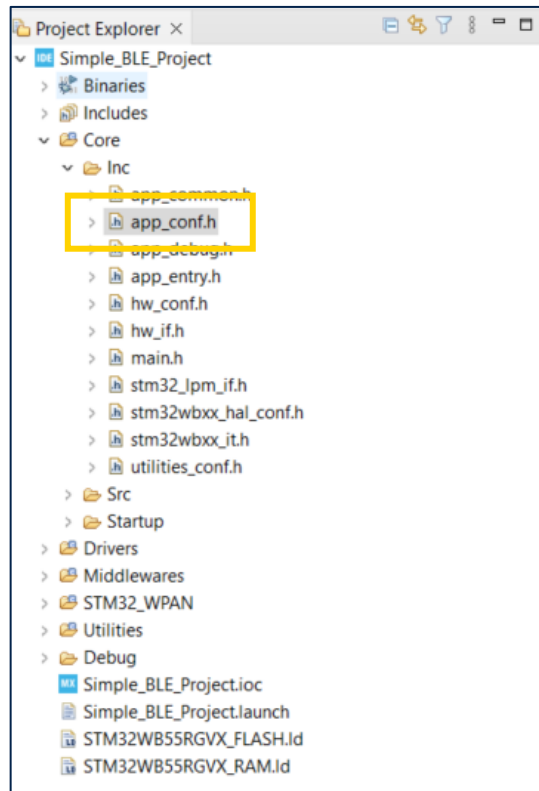
- Checking the second characteristic using BLEToolBox



Making simple BLE project

Step #40

- Update app_conf.h as follows
(location : Simple_BLE_Project/Core/Inc/)



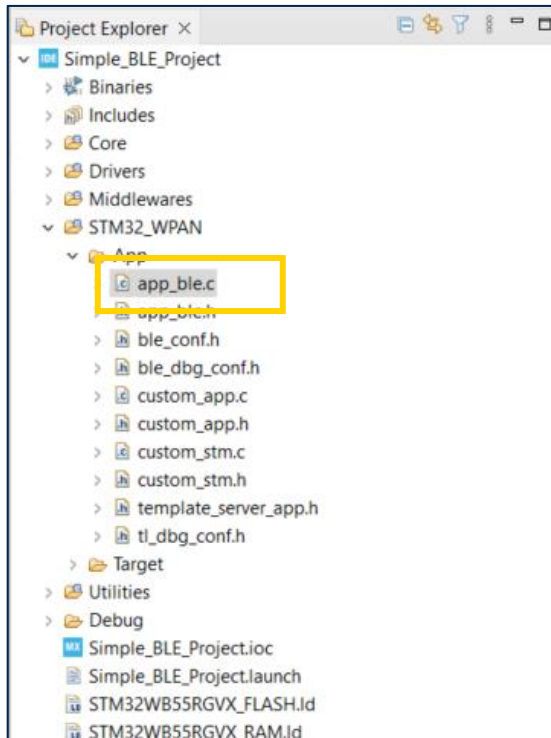
```
/* USER CODE BEGIN Defines */  
void task_button1(void);  
/* USER CODE END Defines */
```

```
typedef enum  
{  
    CFG_TASK_ADV_CANCEL_ID,  
    #if (L2CAP_REQUEST_NEW_CONN_PARAM != 0 )  
    CFG_TASK_CONN_UPDATE_REG_ID,  
    #endif  
    CFG_TASK_HCI_ASYNC_EVT_ID,  
    /* USER CODE BEGIN CFG_Task_Id_With_HCI_Cmd_t */  
    CFG_TASK_BUTTON1_ID,  
    /* USER CODE END CFG_Task_Id_With_HCI_Cmd_t */  
    CFG_LAST_TASK_ID_WITH_HCICMD,  
} CFG_Task_Id_With_HCI_Cmd_t;
```

Making simple BLE project

Step #41

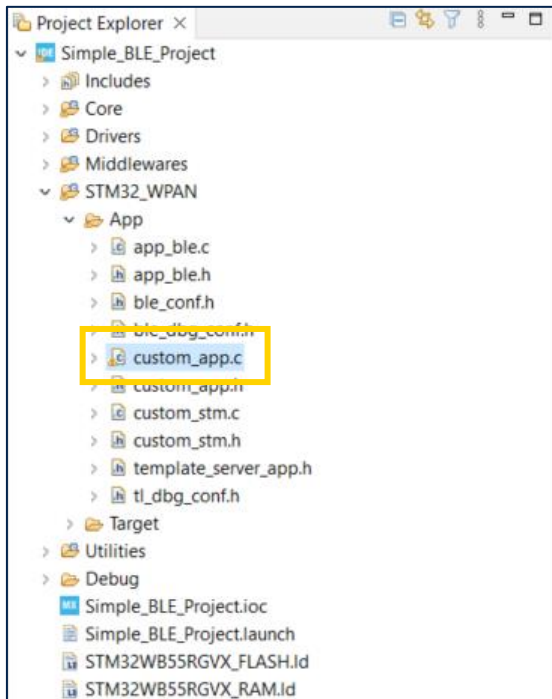
- Update app_ble.c as follows
(location : Simple_BLE_Project/STM32_WPAN/App/)



```
void APP_BLE_Init(void)
{
    SHCI_CmdStatus_t status;
    #if (RADIO_ACTIVITY_EVENT != 0)
        tBleStatus ret = BLE_STATUS_INVALID_PARAMS;
    #endif /* RADIO_ACTIVITY_EVENT != 0 */
    /* USER CODE BEGIN APP_BLE_Init 1 */
    UTIL_SEQ_RegTask(1<<CFG_TASK_BUTTON1_ID, UTIL_SEQ_RFU, task_button1);
    UTIL_SEQ_SetTask(1<<CFG_TASK_BUTTON1_ID, CFG_SCH_Prio_0);
    /* USER CODE END APP_BLE_Init 1 */
}
```

Making simple BLE project Step #42

- Update custom_app.c as follows
(location : Simple_BLE_Project/STM32_WPAN/App/)



```
/* USER CODE BEGIN PV */
uint8_t notidata = 0;
uint8_t notiflag = 0;
uint8_t keystate = 0;
/* USER CODE END PV */
```

```
/* USER CODE BEGIN PFP */
void task_button1(void)
{
    if(!HAL_GPIO_ReadPin(B1_GPIO_Port, B1_Pin))//Press
    {
        if(keystate == 0)
        {
            APP_DBG_MSG("button1 press!\n\r");
            keystate = 1;

            NotifyCharData[0] = notidata;
            Custom_St_nchar_Send_Notification();

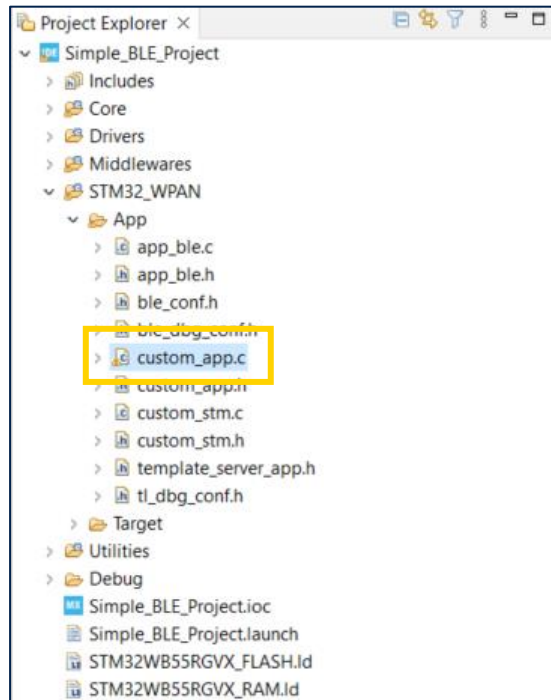
            notidata++;
            if(notidata == 0xFF)
                notidata = 0;
        }
    }
    else//Release
    {
        keystate = 0;
    }

    UTIL_SEQ_SetTask(1<<CFG_TASK_BUTTON1_ID, CFG_SCH_PRIO_0);
}
/* USER CODE END PFP */
```

Making simple BLE project

Step #43

- Update custom_app.c as follows
(location : Simple_BLE_Project/STM32_WPAN/App/)

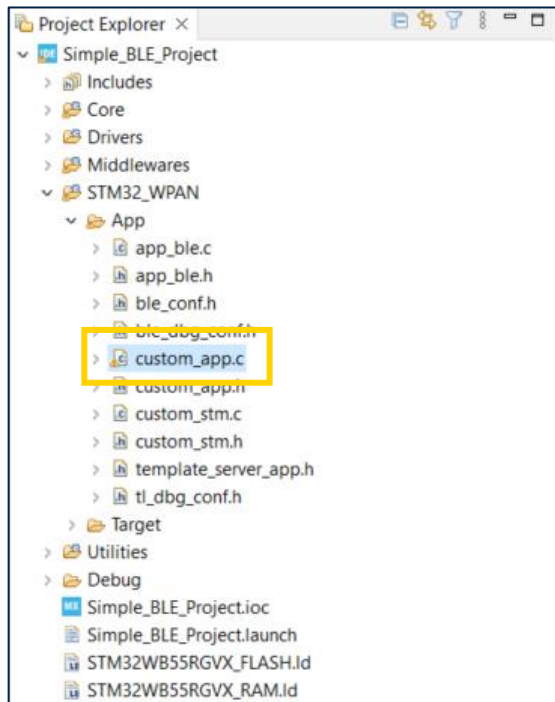


```
case CUSTOM_STM_ST_NCHAR_NOTIFY_ENABLED_EVT:  
    /* USER CODE BEGIN CUSTOM_STM_ST_NCHAR_NOTIFY_ENABLED_EVT */  
    APP_DBG_MSG("notification enabled!\n\r");  
    notiflag = 1;  
    /* USER CODE END CUSTOM_STM_ST_NCHAR_NOTIFY_ENABLED_EVT */  
    break;  
  
case CUSTOM_STM_ST_NCHAR_NOTIFY_DISABLED_EVT:  
    /* USER CODE BEGIN CUSTOM_STM_ST_NCHAR_NOTIFY_DISABLED_EVT */  
    APP_DBG_MSG("notification disabled!\n\r");  
    notiflag = 0;  
    /* USER CODE END CUSTOM_STM_ST_NCHAR_NOTIFY_DISABLED_EVT */  
    break;
```

Making simple BLE project

Step #44

- Update custom_app.c as follows
(location : Simple_BLE_Project/STM32_WPAN/App/)



```
void Custom_St_nchar_Send_Notification(void) /* Property Notification */
{
    uint8_t updateflag = 0;

    /* USER CODE BEGIN St_nchar_NS_1*/

    /* USER CODE END St_nchar_NS_1*/

    if (updateflag != 0)
    {
        Custom_STM_App_Update_Char(CUSTOM_STM_ST_NCHAR, (uint8_t *)NotifyCharData);
    }

    /* USER CODE BEGIN St_nchar_NS_Last*/
    if (notiflag != 0)
    {
        Custom_STM_App_Update_Char(CUSTOM_STM_ST_NCHAR, (uint8_t *)NotifyCharData);
    }
    else
        APP_DBG_MSG("Error. notification is not enabled!\n\r");
    /* USER CODE END St_nchar_NS_Last*/

    return;
}
```

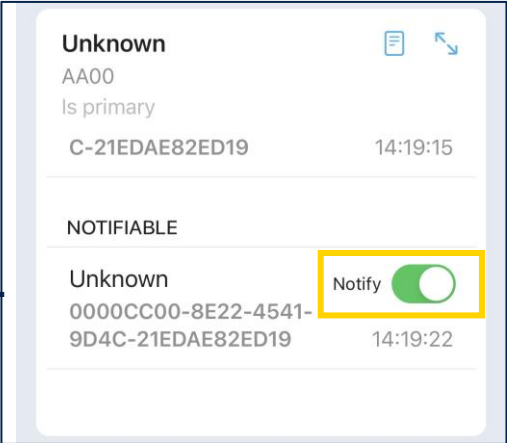
Making simple BLE project

Step #45

- Build Project.



- Programming new firmware again.
(Refer, Hands-on Step #36)

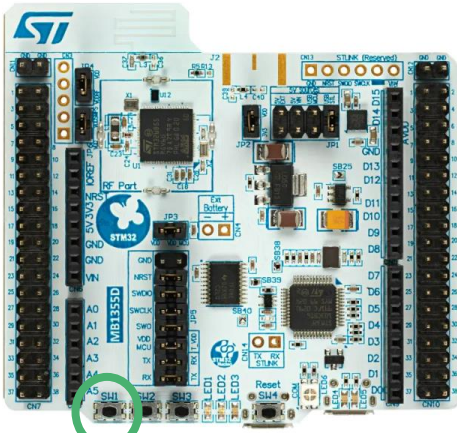


Enable Notify

```

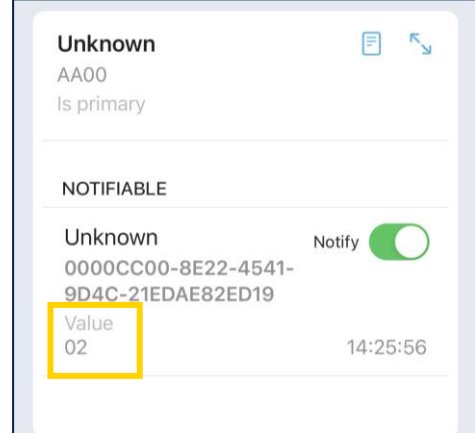
COM26 - Tera Term VT
File Edit Setup Control Window Help
>>== DBGMCU_GetRevisionID= 2001
>>== DBGMCU_GetDeviceID= 495
Success: SHCI_G2_BLE_Init command
==>> Start Ble_Hci_Gatt_Init function
Success: hci_reset command
Success: aci_hal_write_config_data command - CONFIG_DATA_PUBADDR_OFFSET
Public Bluetooth Address: 00:80:e1:26:d8:32
Success: aci_hal_write_config_data command - CONFIG_DATA_IR_OFFSET
Success: aci_hal_write_config_data command - CONFIG_DATA_ER_OFFSET
Success: aci_hal_set_tx_power_level command
Success: aci_gatt_init command
Success: aci_gap_init command
Success: hci_le_set_default_phy command
Success: aci_gap_set_io_capability command
Success: aci_gap_set_authentication_requirement command
==>> End Ble_Hci_Gatt_Init function
Success: aci_gatt_add_service command: ST_SRV
Success: aci_gatt_add_char command : ST_MCHAR
Success: aci_gatt_add_char command : ST_MCHAR
Success: aci_hal_set_radio_activity_mask command
==>> aci_gap_set_discoverable - Success
==>> Success: Start Fast Advertising
>>== HCI_LE_CONNECTION_COMPLETE_SUBEVENT_CODE - Connection handle: 0x801
- Connection established with Central: e:53:ah:56:09:4f:87
- Connection Interval: ns
- Connection latency: 0
- Supervision Timeout: 720 ms
notification enabled!
    
```

- Enable notify by phone
- Press Button 1



Button 1

Send notification data



The number will increase with every press.

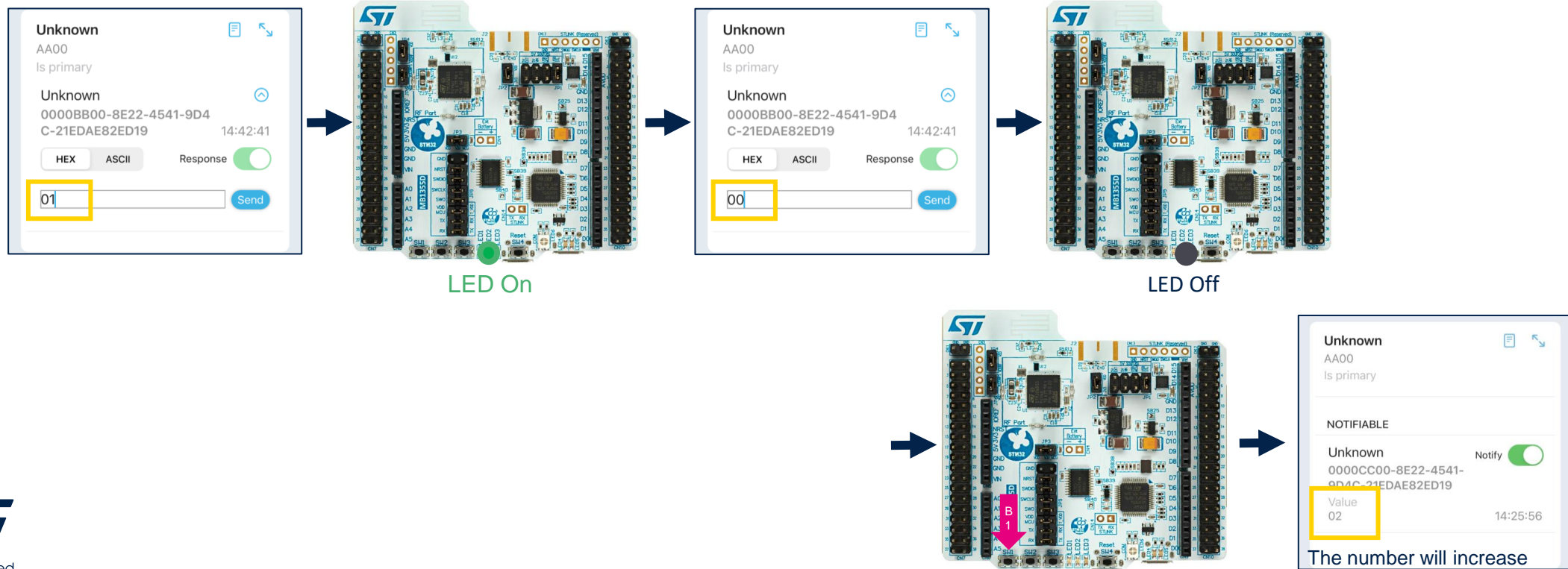
Recording a video for completion

- Please record a video for the following three operations.

1. Turn on LED by phone.

2. Turn off LED by phone.

3. Increasing number by pressing button 1.



Our technology starts with You



Find out more at www.st.com

© STMicroelectronics - All rights reserved.

ST logo is a trademark or a registered trademark of STMicroelectronics International NV or its affiliates in the EU and/or other countries.

For additional information about ST trademarks, please refer to www.st.com/trademarks.

All other product or service names are the property of their respective owners.



life.augmented