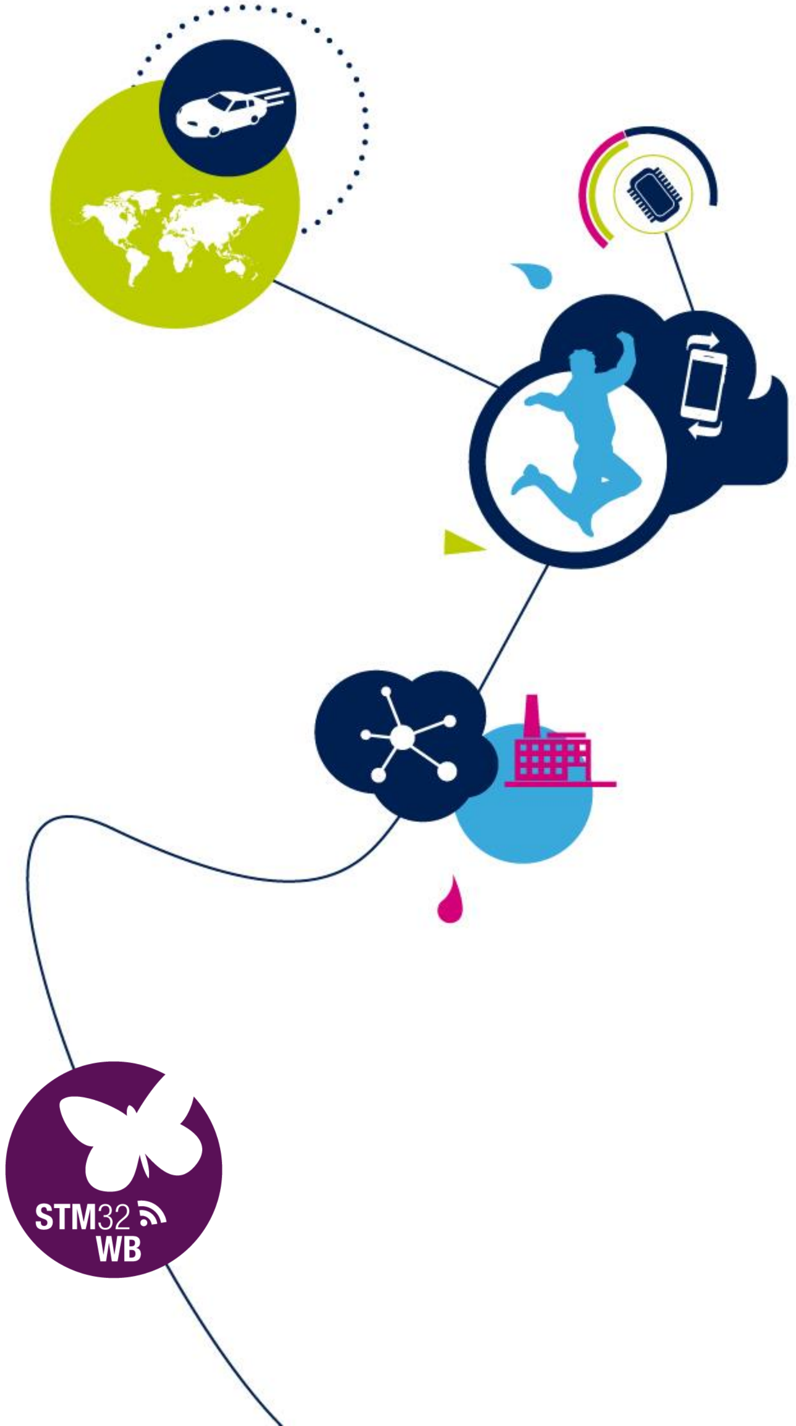
STM32WB Workshop

Americas Marketing and Applications Team

v20



life.augmented

Harald Blatand Gormsson, Danish King

Reign: 958-986

a.k.a. Harald “Bluetooth” Gormsson

Bluetooth: A wireless technology named after Harald Bluetooth, with the analogy that Bluetooth would unite devices like Harald Bluetooth united the tribes of Denmark into a single kingdom.

Objectives

STM32WB Introduction

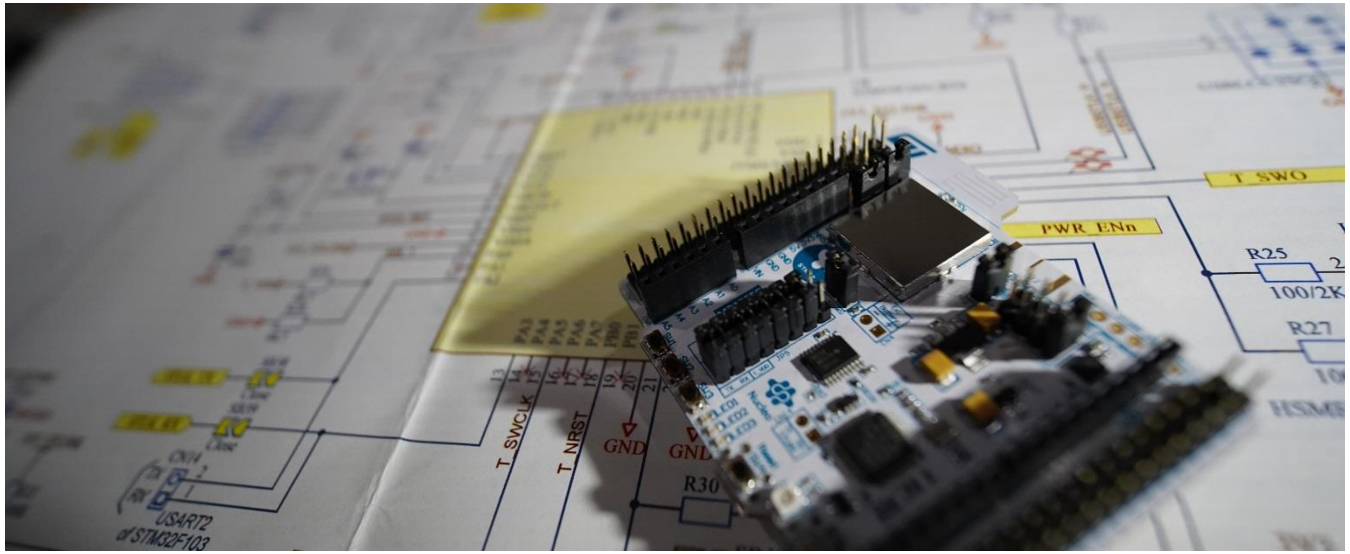
BLE Basics

Tools & Firmware

Lot's of Hands-On coding!

Agenda

8:00	Welcome and Tools install (if needed)
9:00	A few words..
	~~Hands-On: Out-of-the-Box~~
	Hands-On: CubeMX
	BLE Basics
	Hands-On: HRM
10:30	Break
	Architecture
	Hands-On: CubeMonitorRF
12:00	Lunch (1h)
1:00	More WB Detail
	Hands-on: Cable replacement
	Hardware considerations
2:15	Break
	Hands-on: OTA Firmware Updates
2:45	Wrap-Up, Q&A, Survey

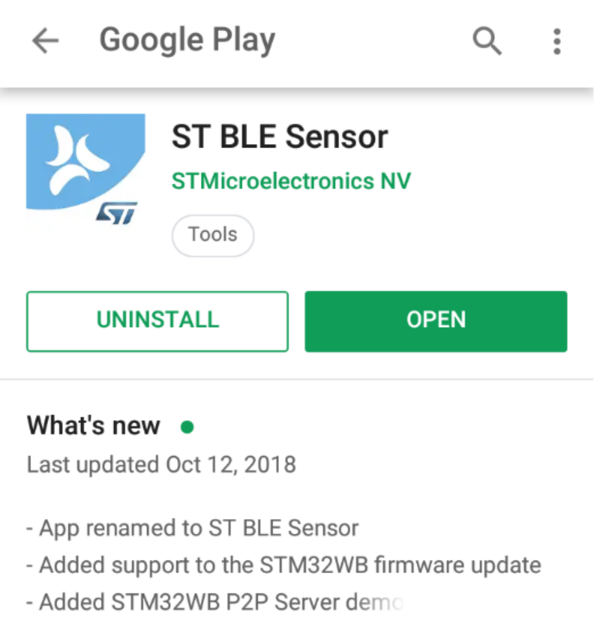


life.augmented

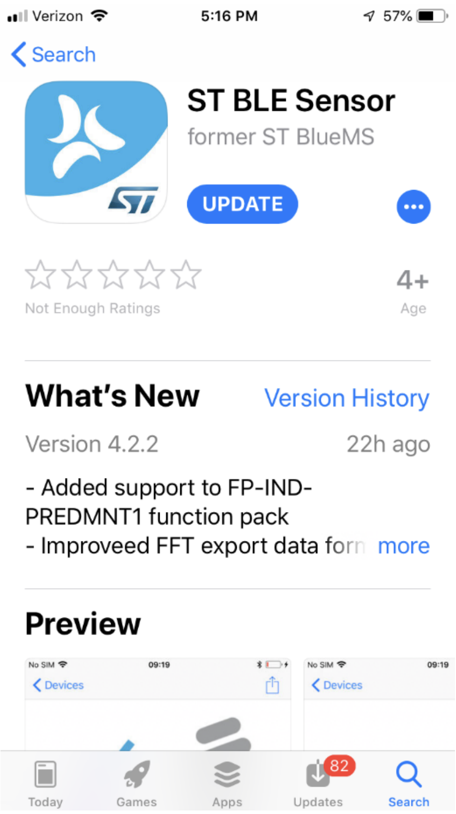
Prerequisites

- Windows 7/10
  - Java JRE v8 (v1.80.0_191 or newer)
- CubeMX, CubeWB, CubeMonitorRF, CubeProgrammer
- ST BLE Sensor App
- LightBlue Explorer App
- IAR EWARM, v8.40.1 + License
- TeraTerm, or equiv.

Play Store



App Store



LightBlue Explorer

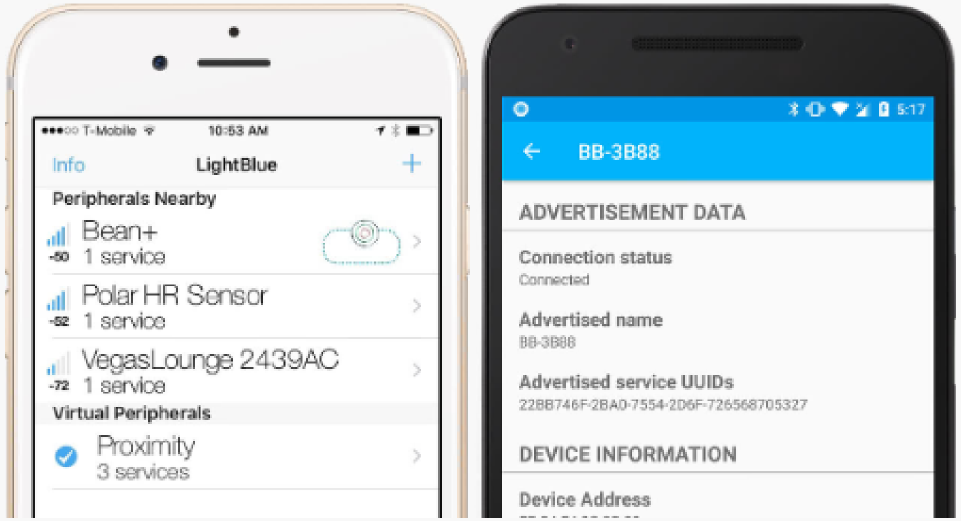
PunchThrough

LightBlue | Explorer

The industry-leading BLE test app for iOS and Android. Used by over a half million people, LightBlue Explorer lets you scan, connect to and browse any nearby Bluetooth Smart device. Includes full support for logging data and simulating peripherals.

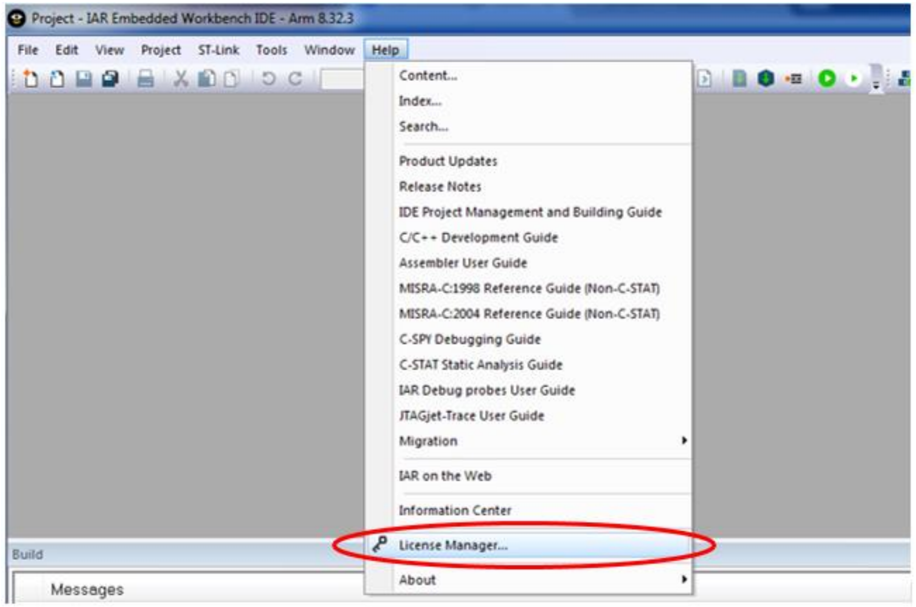
Download on the App Store

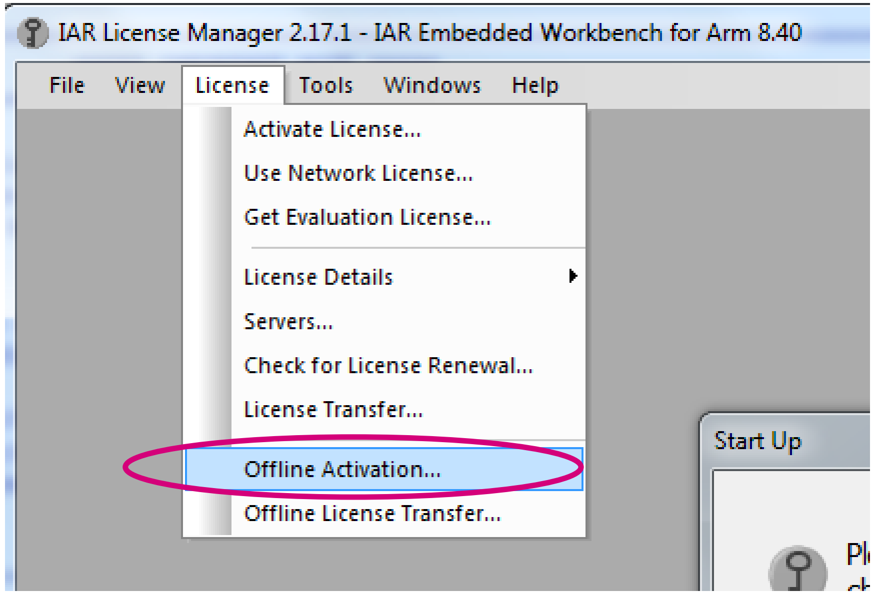
GET IT ON Google Play



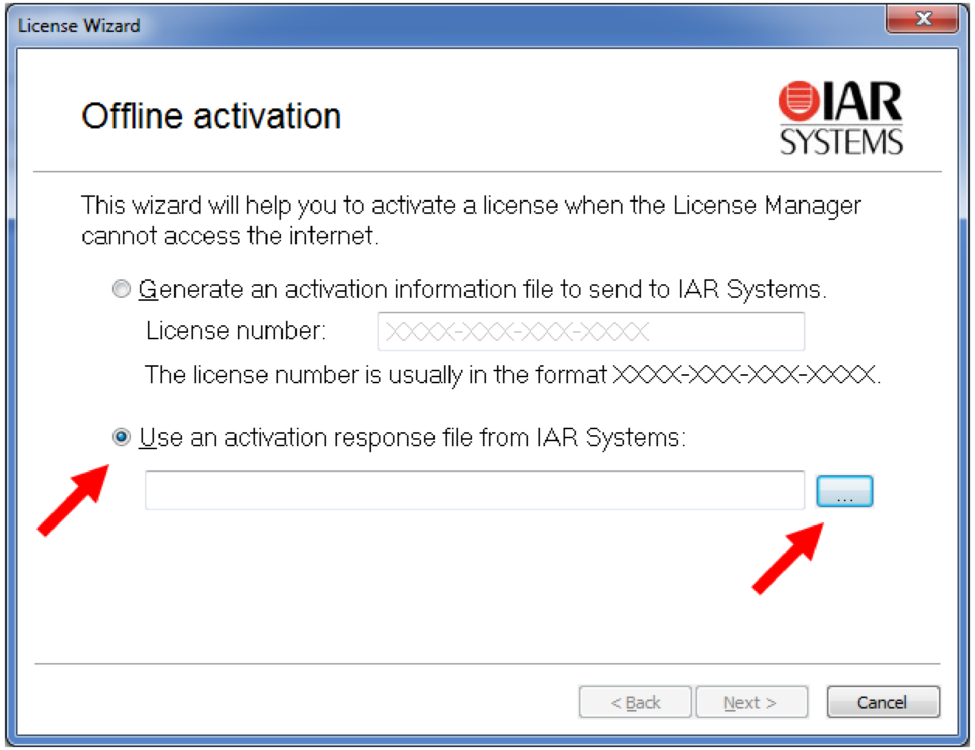
IAR EWARM 8.40.1 - License Installation

- Activate the time-limited evaluation license with Activation Response file included in the provided zip file
- From IAR Embedded Workbench, go to Help->License Manager
- From the License manager, go to License->Offline Activation

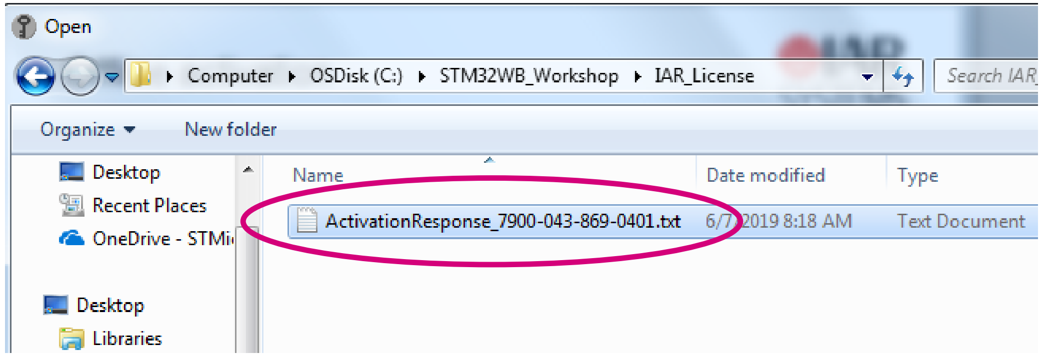




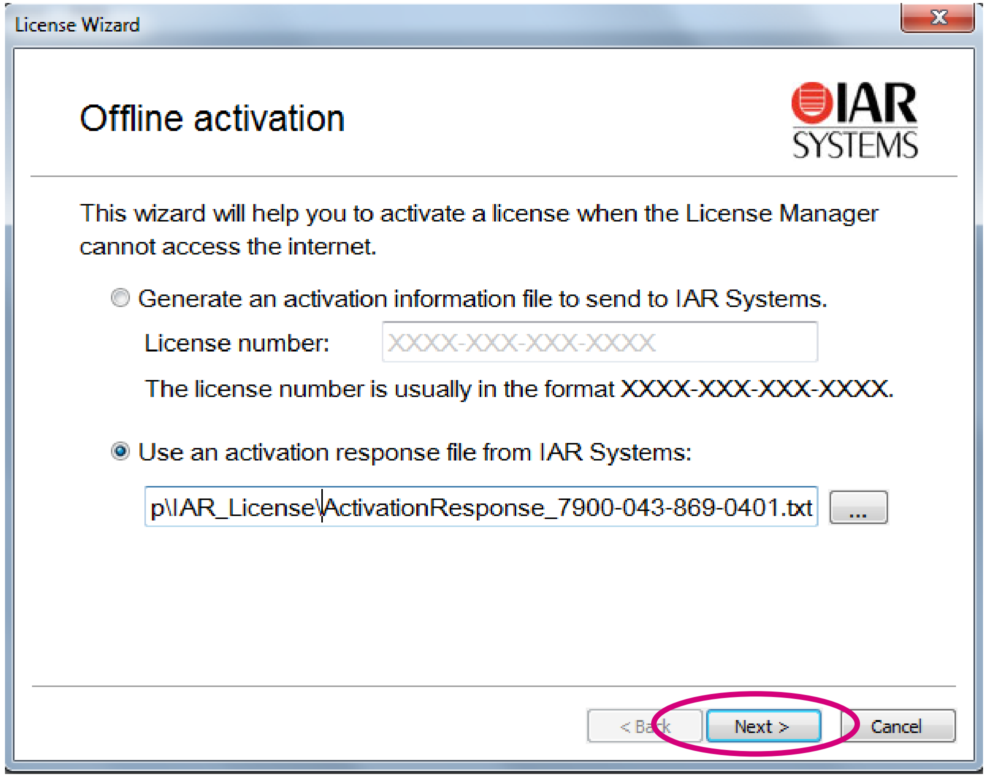
Click “Use an activation response file from IAR Systems” option



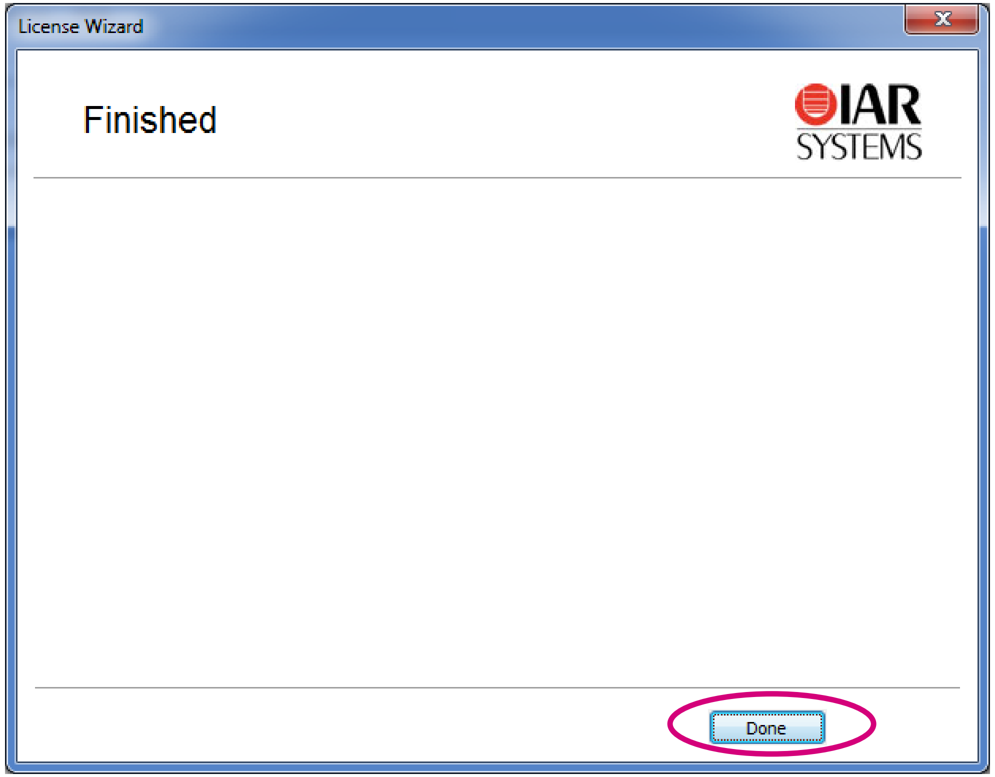
Open the *ActivationResponse_7900-043-869-0401.txt* file

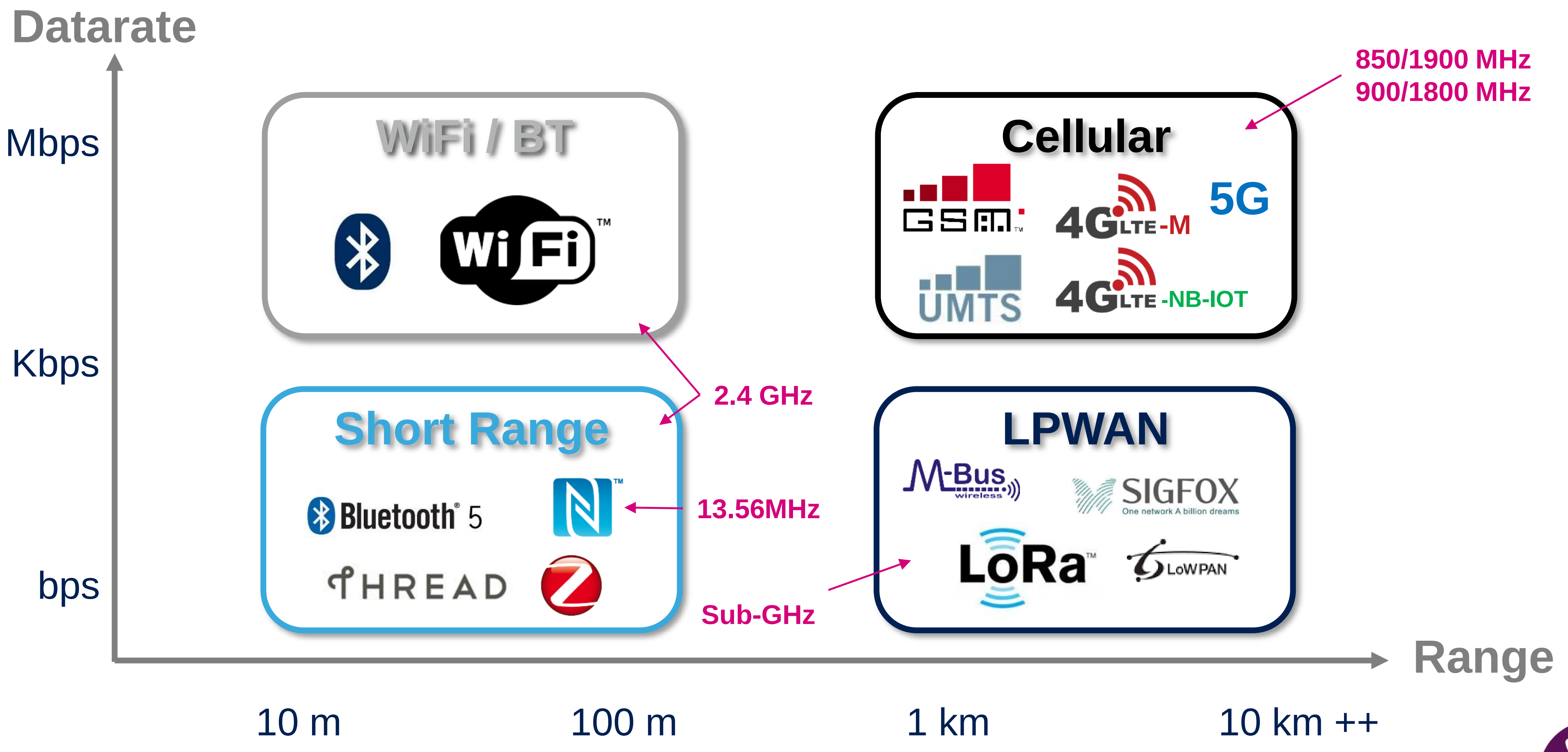


Click “Next”



Click “Done”




Datarate
Mbps
Kbps
bps
WiFi / BT
Wi Fi
Cellular
GSM
4G LTE-M
5G
UMTS
4G LTE -NB-IOT
850/1900 MHz
900/1800 MHz
2.4 GHz
Short Range
Bluetooth 5
THREAD
13.56MHz
Sub-GHz
LPWAN
M-Bus wireless
SIGFOX
One network A billion dreams
LoRa
6LoWPAN
Range
10 m
100 m
1 km
10 km ++

Insulin Pump

Hearing Aid

Watches

Glasses

Locator Tag

Fitness

Bluetooth Smart
Point-to-point communication with smartphones and other wireless devices

Alarm

Heating/Cooling

Door lock

White goods

Smoke detectors

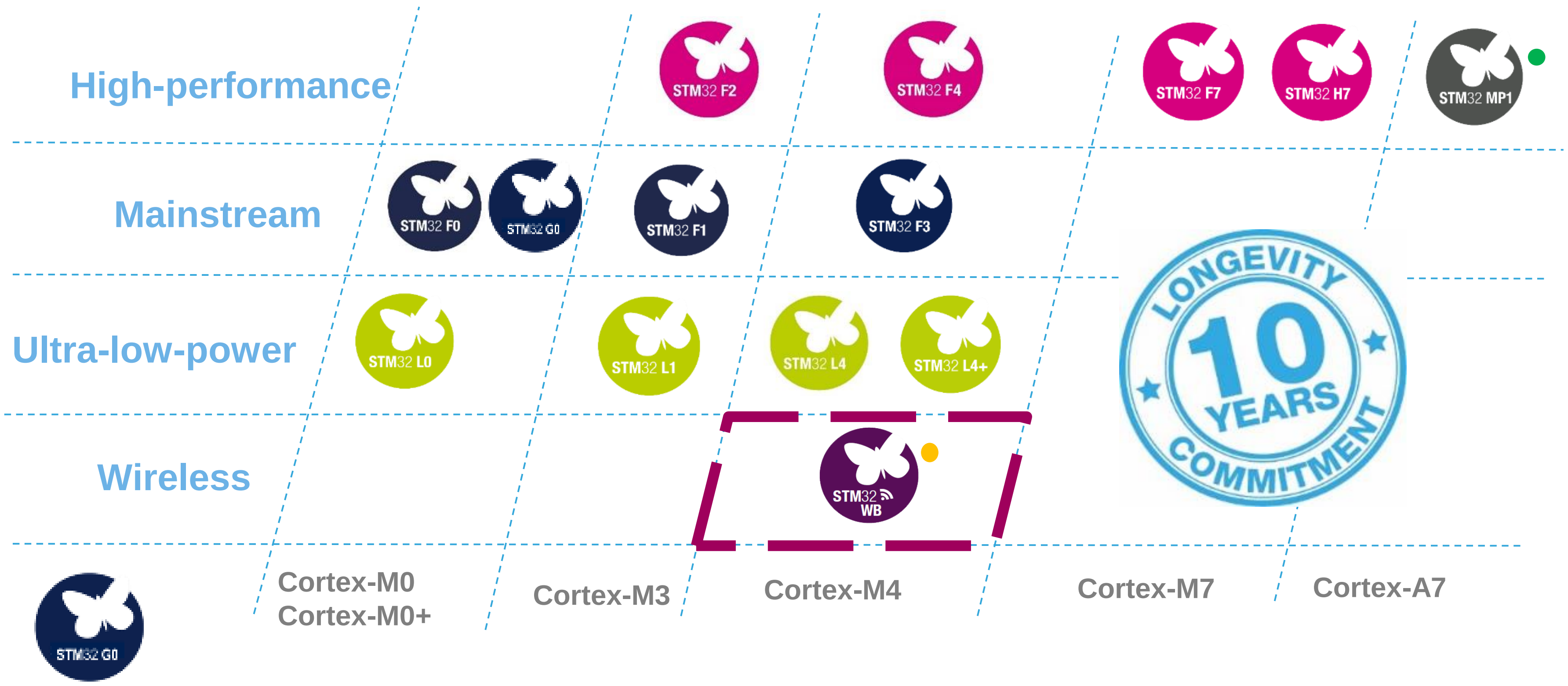
Lighting

BLE Mesh / 802.15.4
Home automation with Mesh network

Classic? Smart Ready? Smart?

"SMART READY" and ***"SMART"*** are abandoned markings

STM32 Portfolio

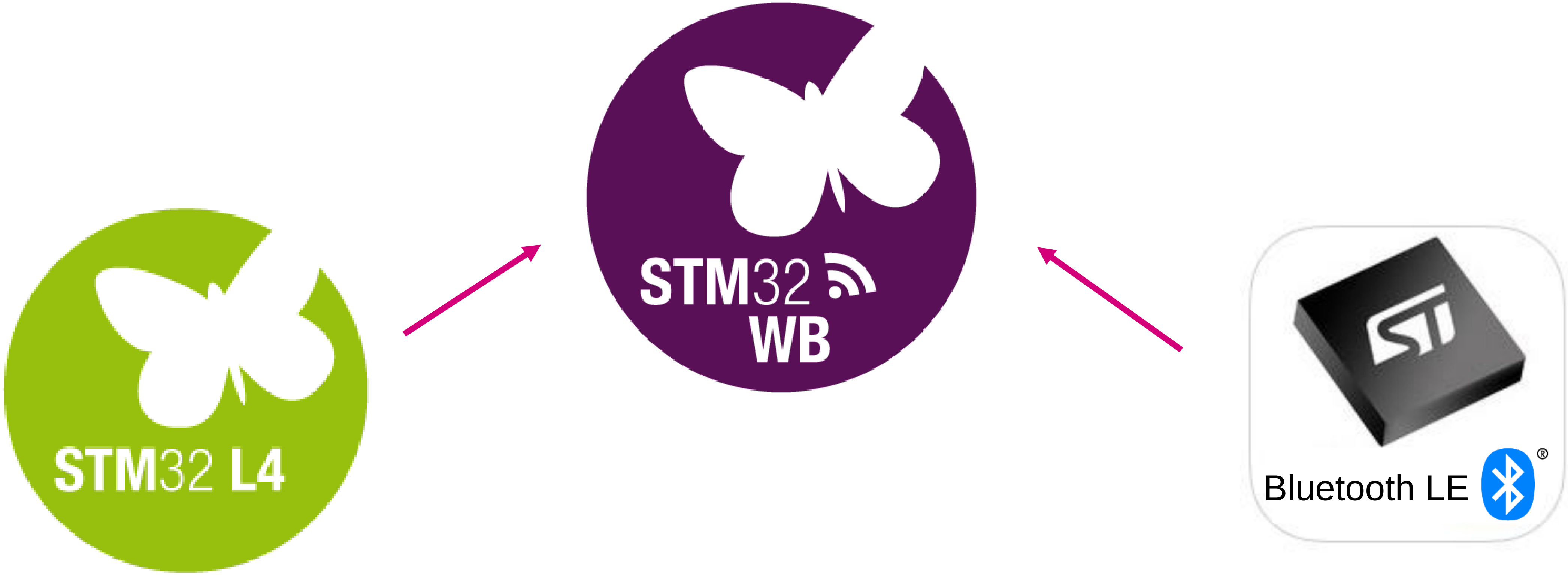


More than
40,000 customers

● Legend: Cortex-M0+ Radio Co-processor

● Legend: Single or Dual-Core A7 with Cortex M4

life.augmented


STM32 L4
STM32 WB
Bluetooth LE

Bluetooth® 5
THREAD
ZigBee®

Multi-protocol

Dual-core
Ultra-Low Power

Secure

STM32 Cube
STM32 CubeMonitor-RF

Comprehensive Ecosystem

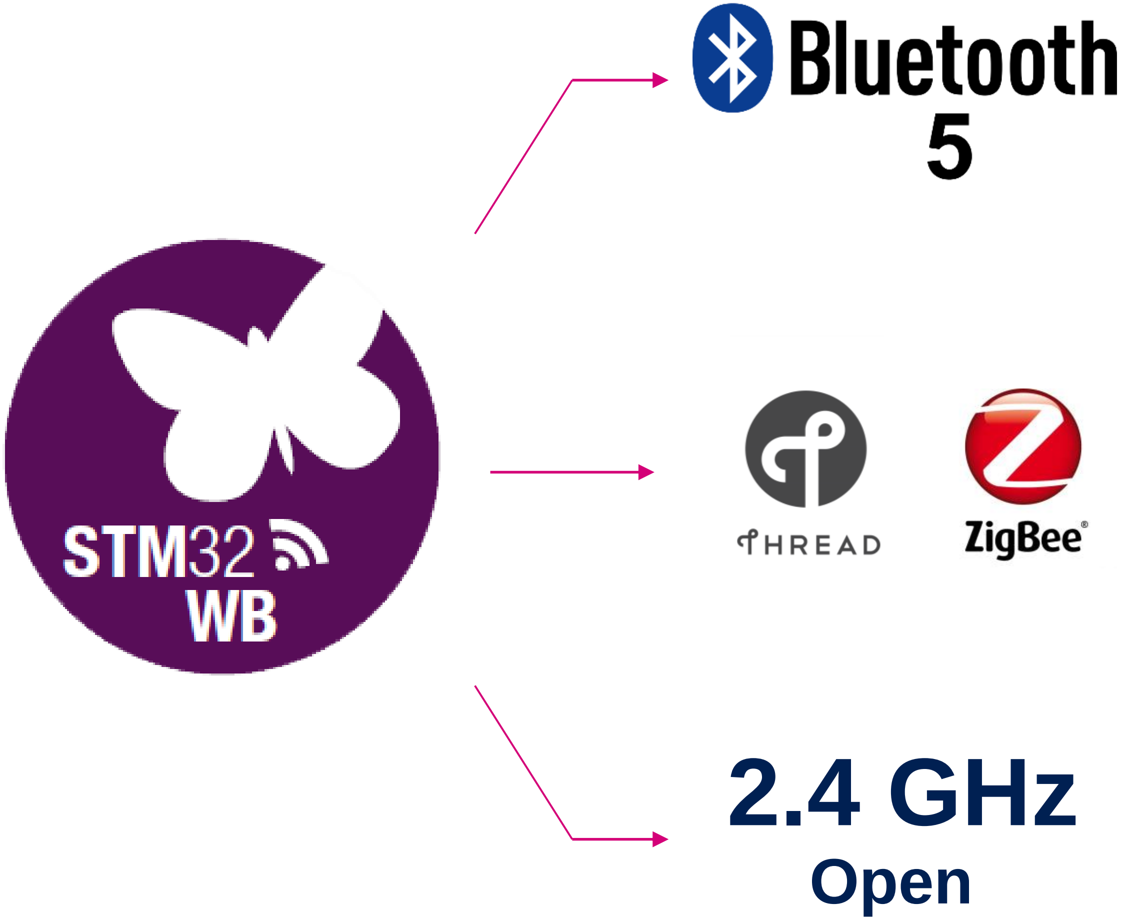
STM32 WB

Bluetooth 5

THREAD

ZigBee

2.4 GHz
Open



- Fully certified
- 2Mbps
- BLE Mesh

STM32 WB

Bluetooth 5

THREAD ZigBee

- Zigbee 3.0
- OpenThread
- Concurrent BLE + OpenThread

2.4 GHz
Open

Multiprotocol

STM32 WB

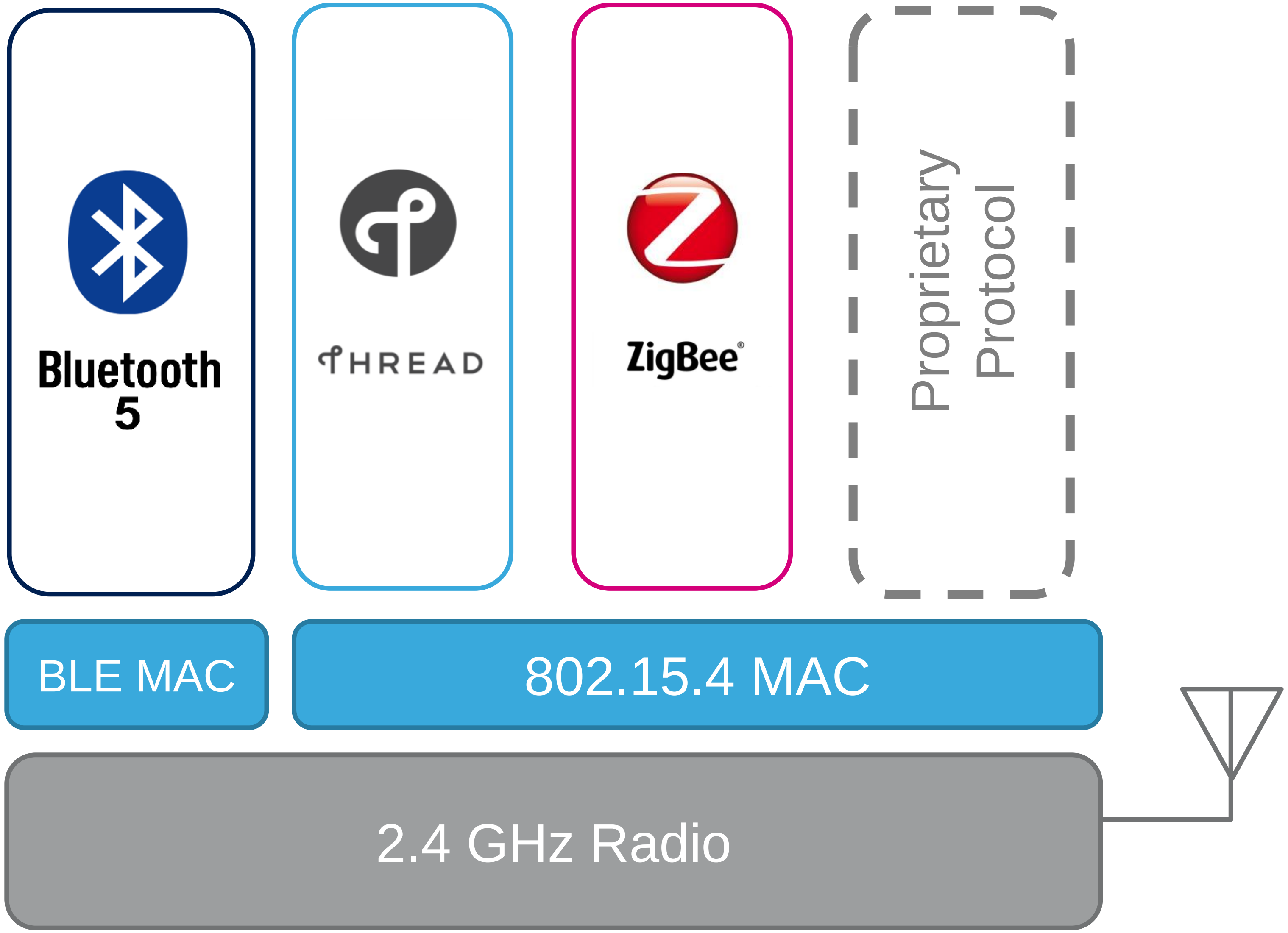
Bluetooth 5

THREAD

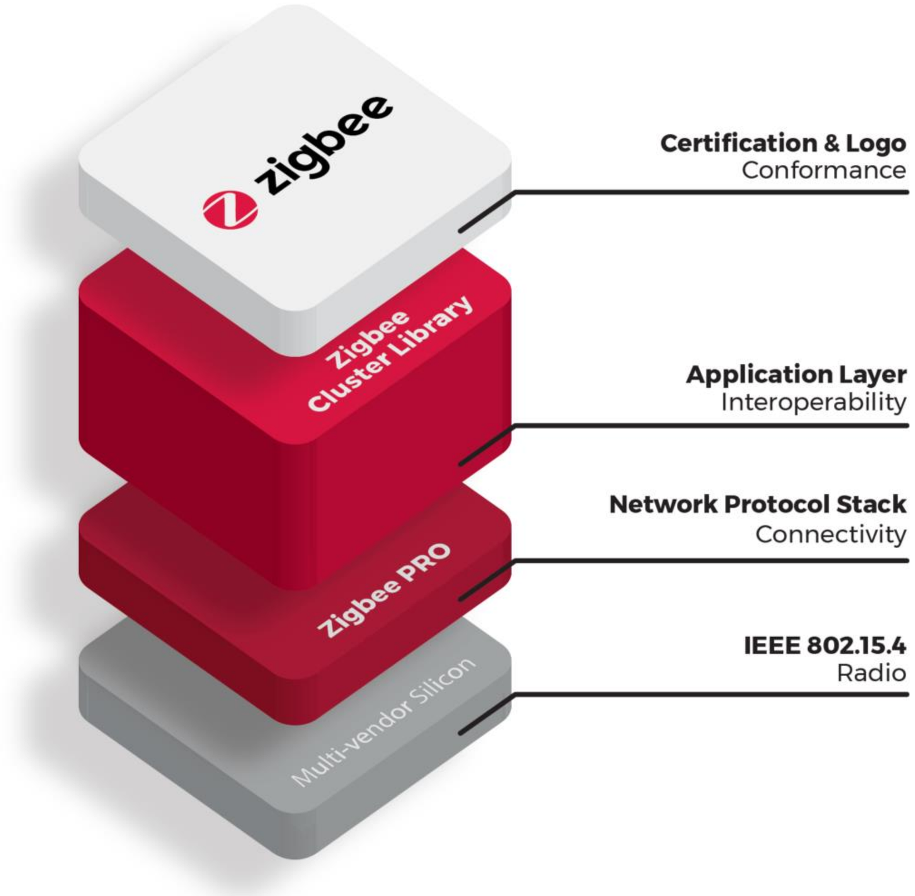
ZigBee

2.4 GHz Open

- OpenMAC


Bluetooth 5
THREAD
ZigBee
Proprietary Protocol
BLE MAC
802.15.4 MAC
2.4 GHz Radio

- Fully certified
- Legacy cluster support
- Revision R21 to R23
- Coming in late 2019



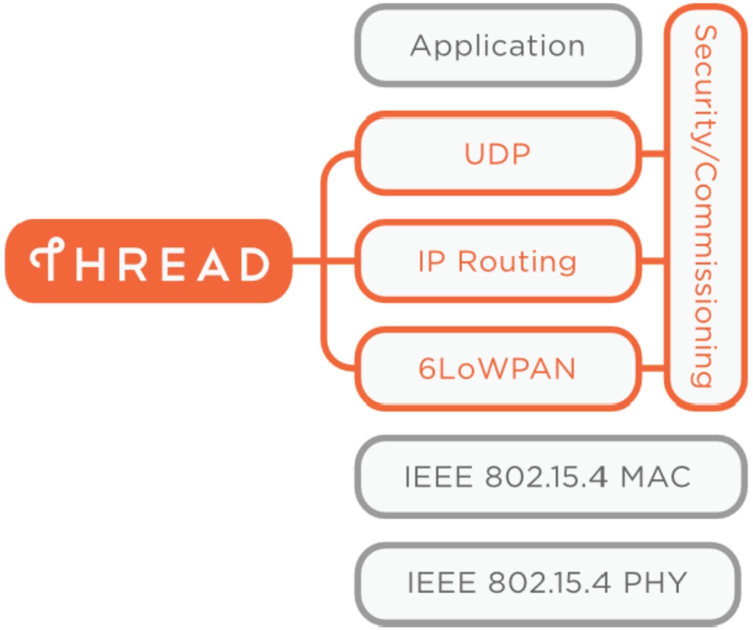
THREAD What it delivers

ST is member of Thread Group
(Contributor level)

A secure wireless mesh network for your home and its connected products

- Built on well-proven, existing technologies
- Uses 6LoWPAN and carries IPv6 natively
- Runs on existing 802.15.4 silicon
- New security architecture to make it simple and secure to add / remove products
- 250+ products per network
- Designed for very low power operation
- Reliable for critical infrastructure

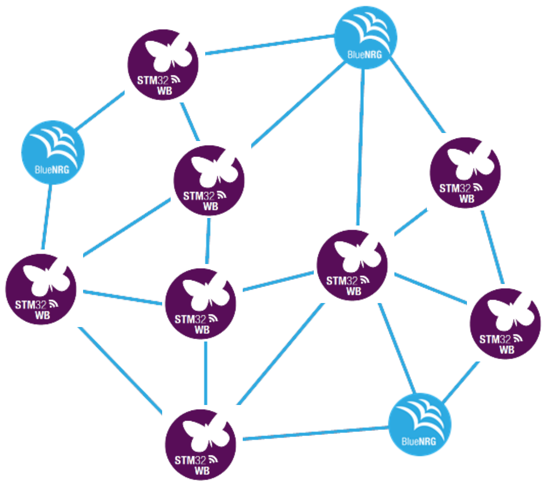
Can support many popular application layer protocols and platforms



A software upgrade can add Thread to currently shipping 802.15.4 products

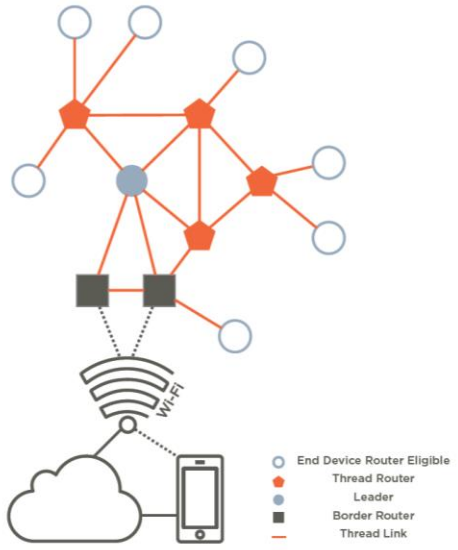
Bluetooth Mesh

- Based on Bluetooth 4.0 and later
- Broadcast type, flood the network with messages, no routing
- Shorter range, 3kbps application data rate, 1Mbps on-air bit-rate
- High power consumption



Thread

- IPv6-based using 802.15.4 MAC
- Routing table approach with network self healing
- Medium range, 40Kbps application data rate, 250Kbps on-air bit-rate
- Low power consumption

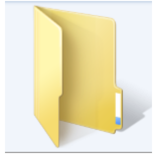


STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

STM32CubeWB

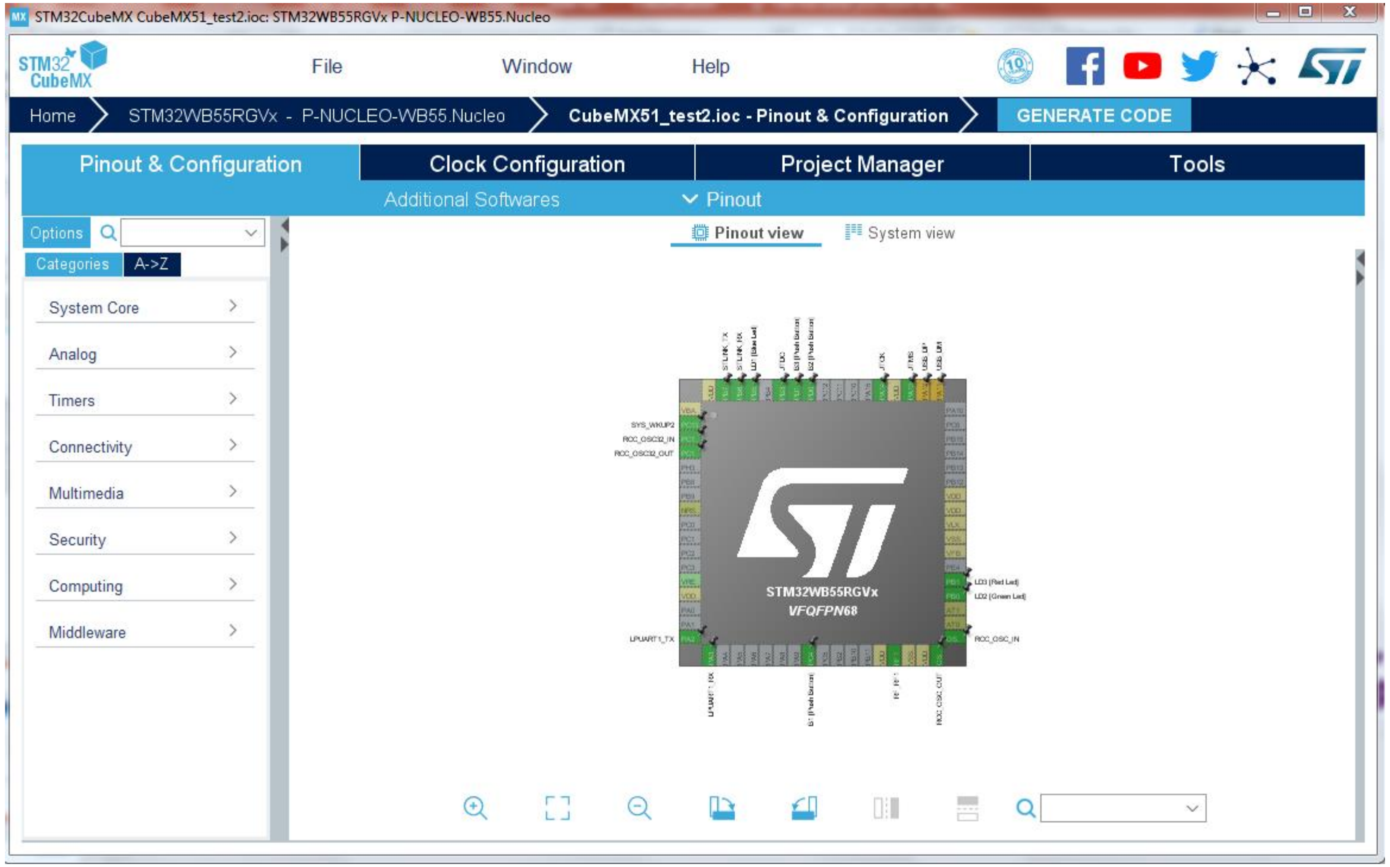


STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware

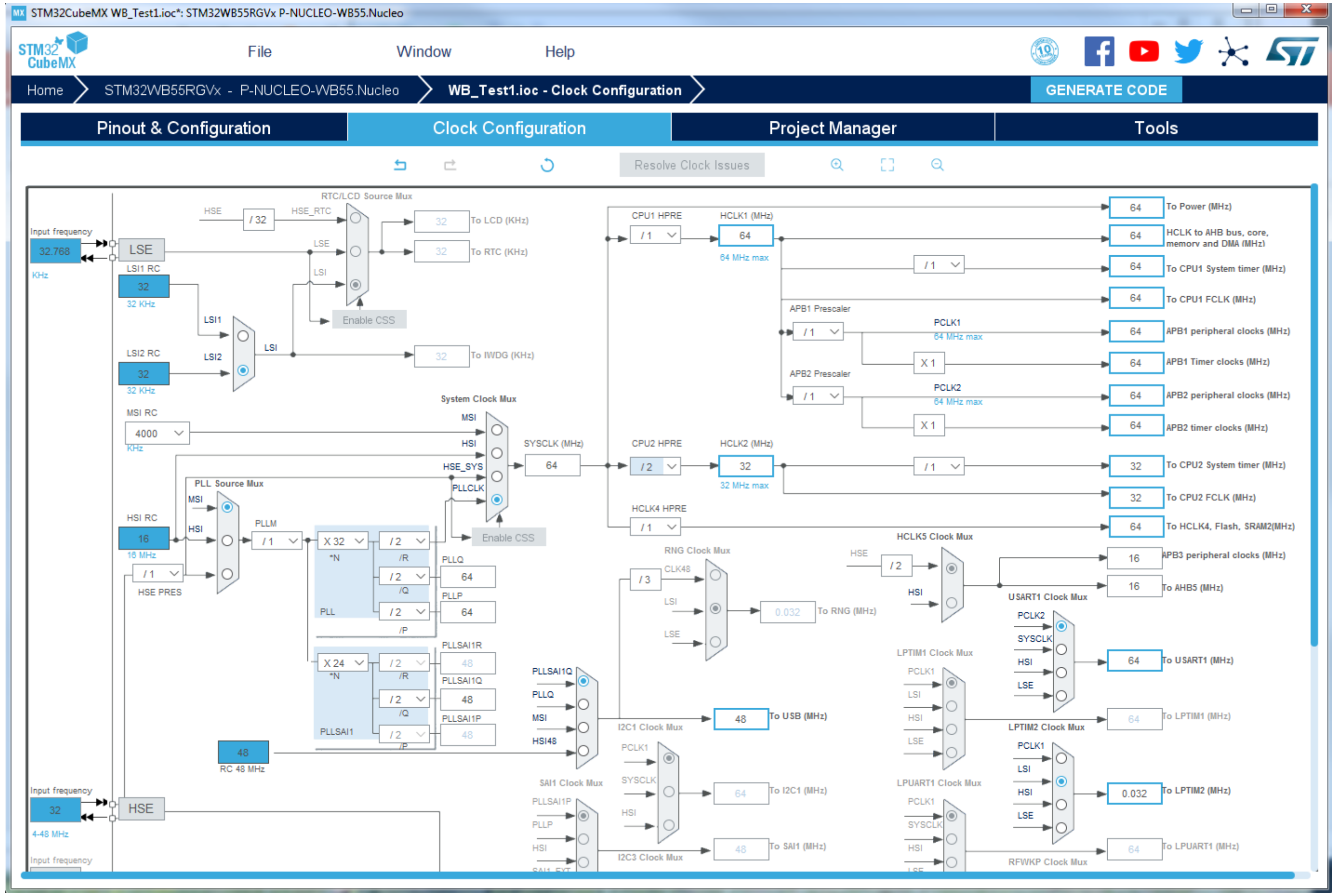


STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware

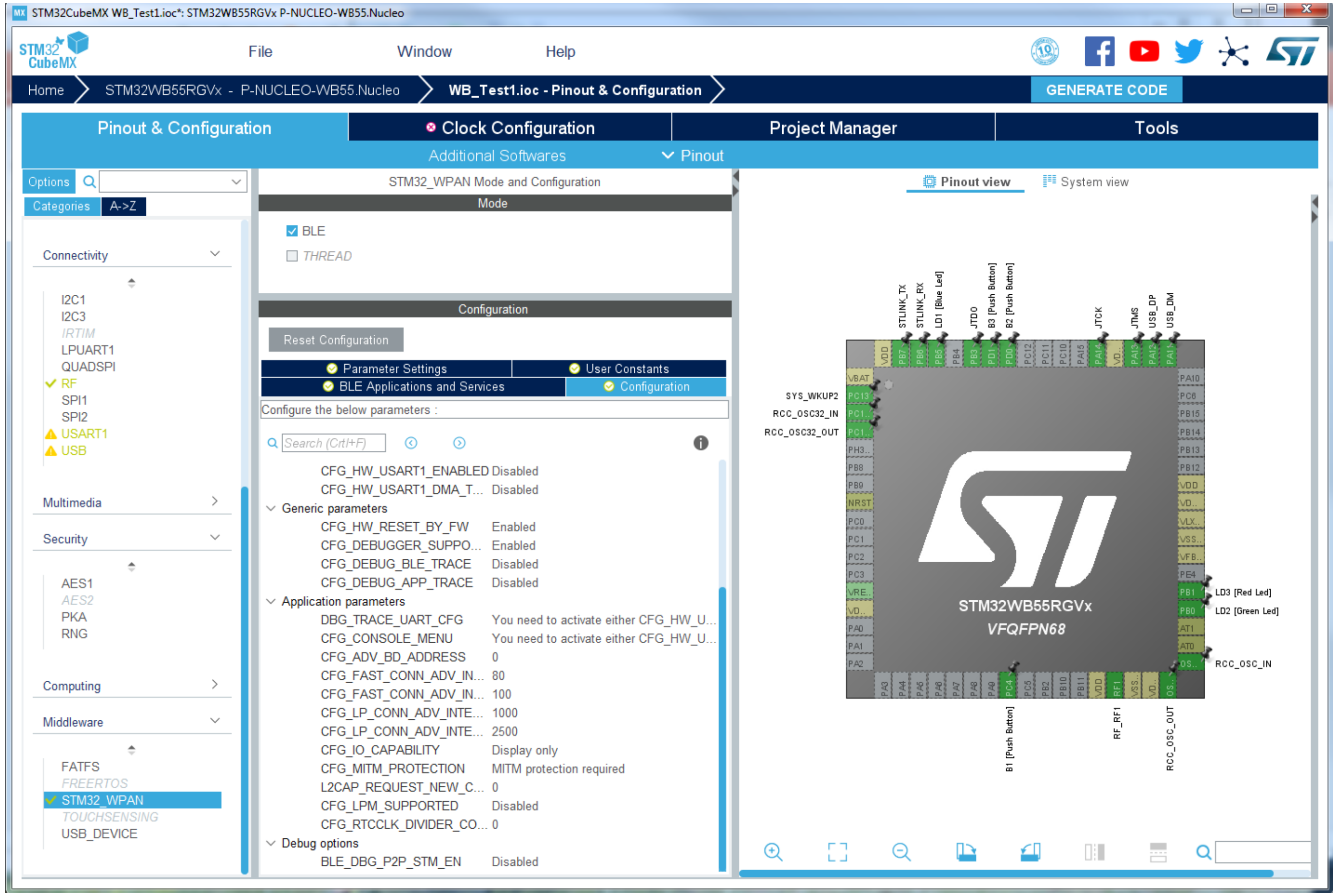


STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware

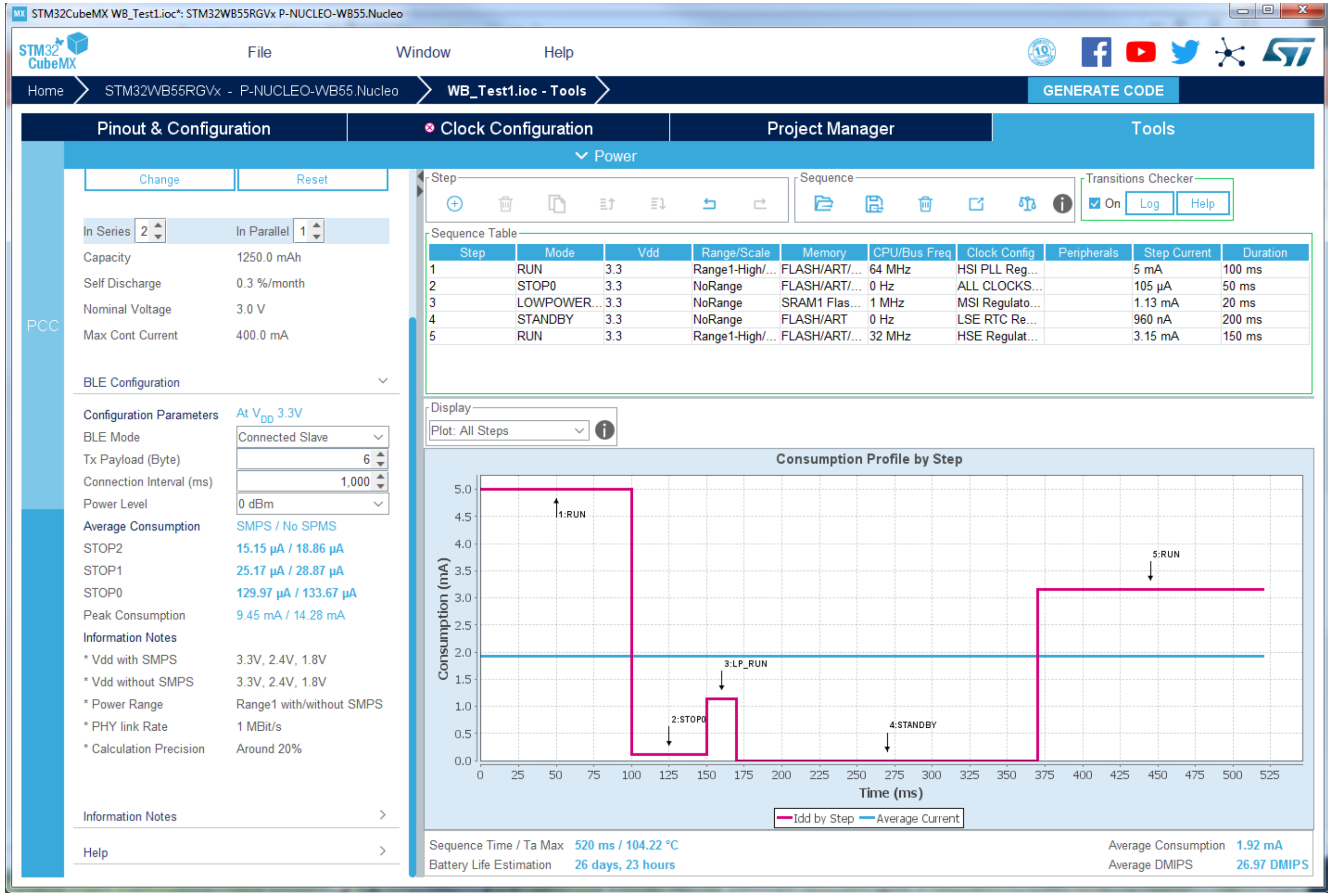


STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware



life.augmented

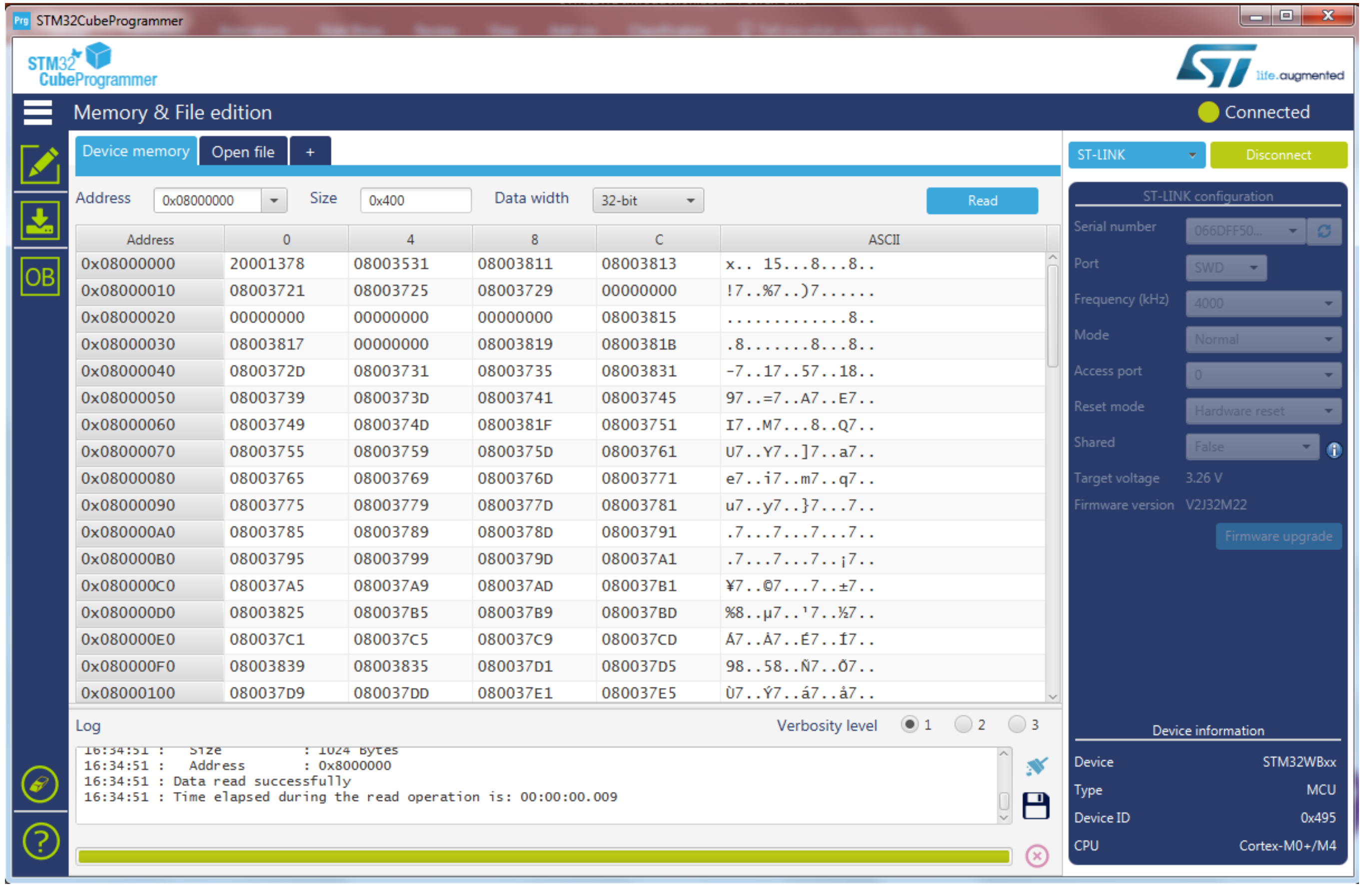
STM32 WB

STM32CubeMX

STM32CubeProgrammer

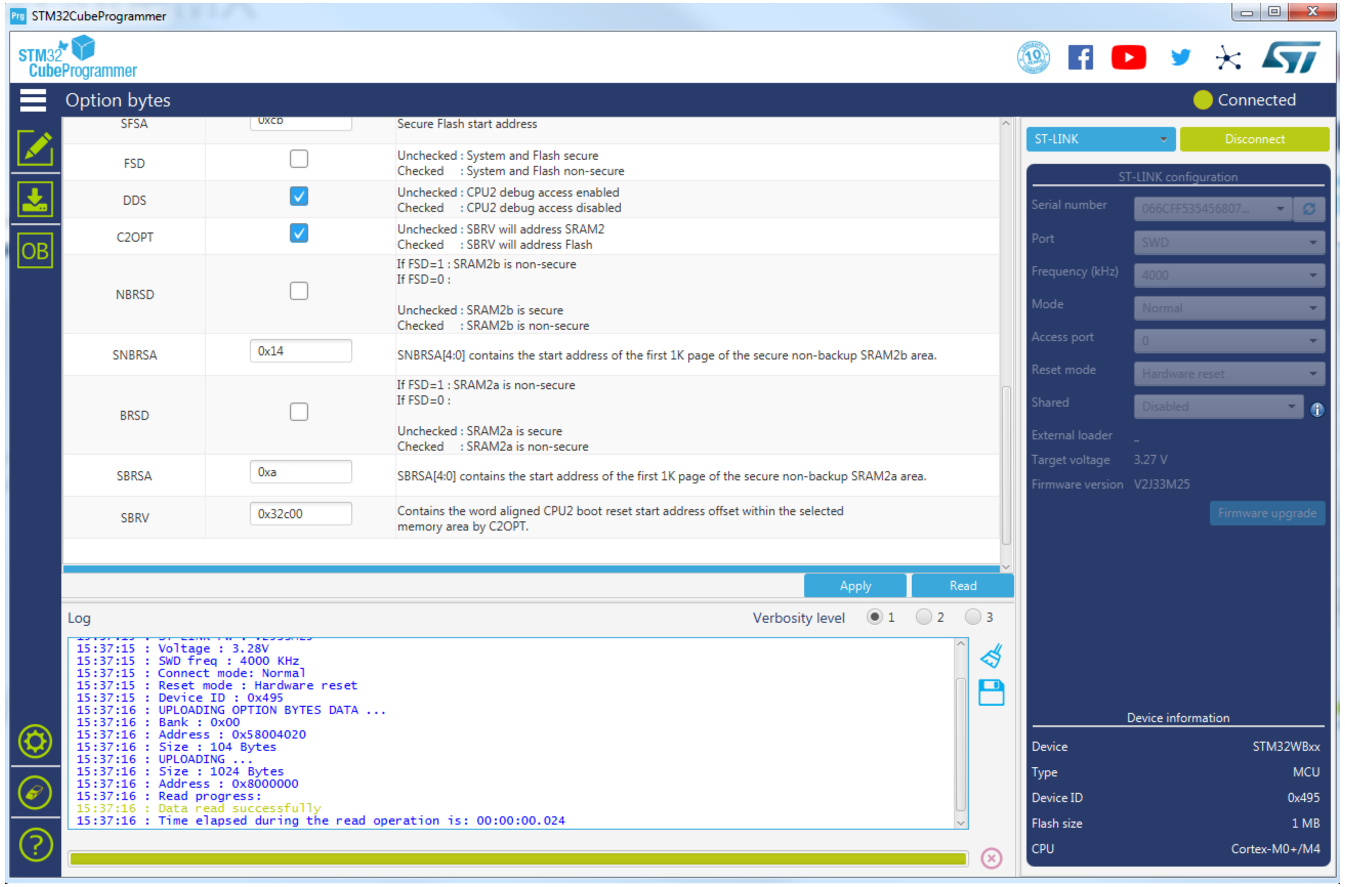
STM32CubeMonitorRF

CubeWB HAL Firmware



STM32CubeMX

STM32CubeProgrammer



STM32CubeMonitorRF

CubeWB HAL Firmware

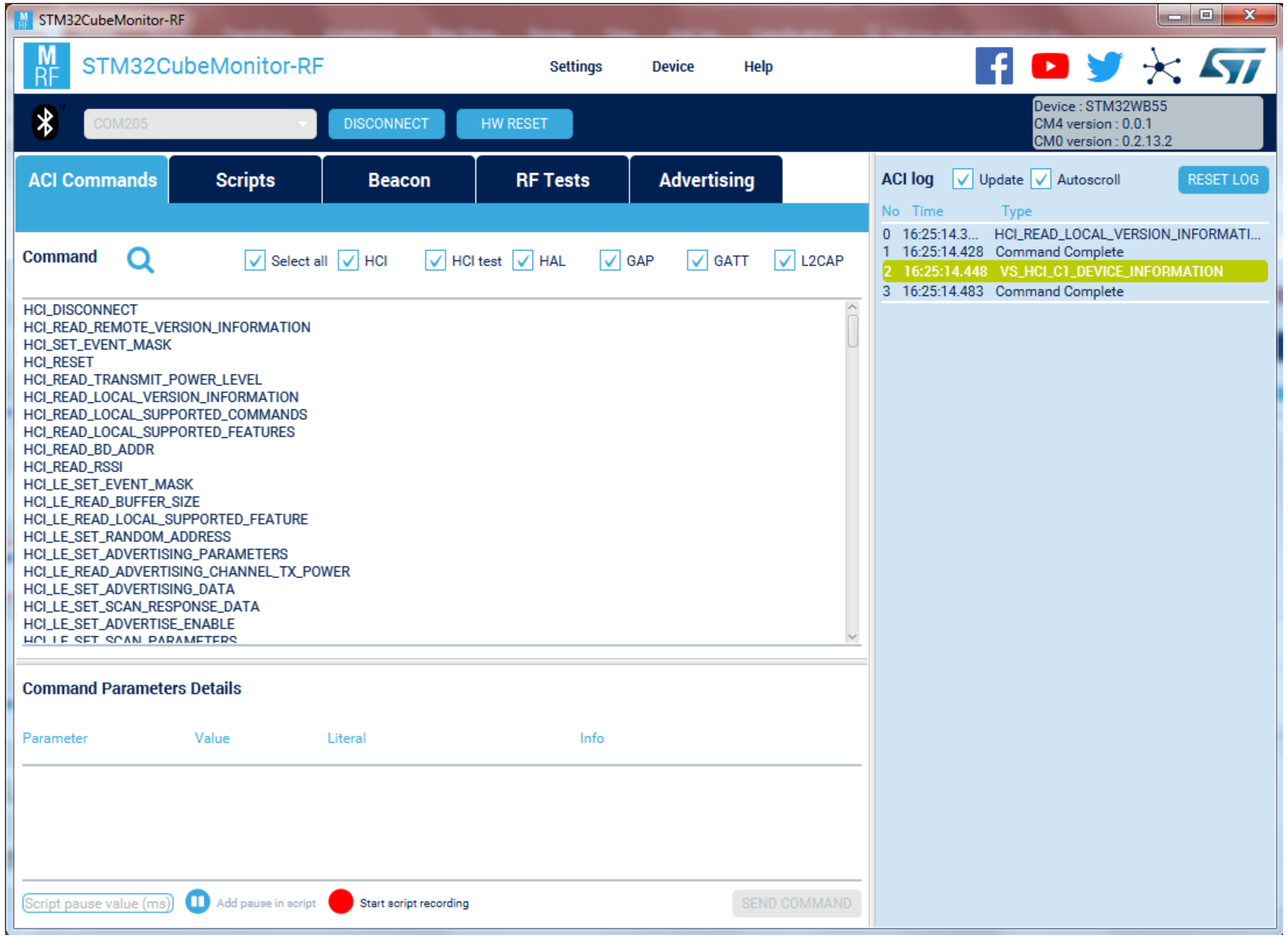
life.augmented

STM32 WB

STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF



CubeWB HAL Firmware

STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware

- ADC
- BSP
- COMP
- Cortex
- CRC
- CRYP
- DMA
- FLASH
- GPIO
- HAL
- HSEM
- I2C
- IWDG
- LPTIM
- PKA
- PWR
- RCC
- RNG
- SPI
- TIM
- UART
- WWDG

- Ble_Thread_Static

- Thread_Cli_Cmd
- Thread_Coap_DataTransfer
- Thread_Coap_Generic
- Thread_Coap_MultiBoard
- Thread_Commissioning
- Thread_FTD_Coap_Multicast
- Thread_SED_Coap_Multicast

- BLE_Beacon
- BLE_BloodPressure
- BLE_CableReplacement
- BLE_DataThroughput
- BLE_HealthThermometer
- BLE_HeartRate
- BLE_HeartRate_ota
- BLE_HeartRateFreeRTOS
- BLE_Hid
- BLE_MeshLightingDemo
- BLE_Ota
- BLE_p2pClient
- BLE_p2pRouteur
- BLE_p2pServer
- BLE_p2pServer_ota
- BLE_Proximity
- BLE_TransparentMode

- FreeRTOS_Mail
- FreeRTOS_MPU
- FreeRTOS_Mutexes
- FreeRTOS_Queues
- FreeRTOS_Semaphore
- FreeRTOS_SemaphoreFromISR
- FreeRTOS_Signal
- FreeRTOS_SignalFromISR
- FreeRTOS_ThreadCreation
- FreeRTOS_Timers

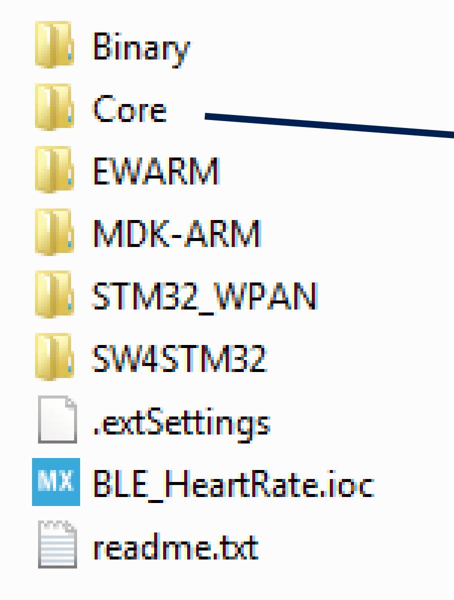
- CDC_Standalone
- DFU_Standalone
- HID_Standalone
- MSC_Standalone

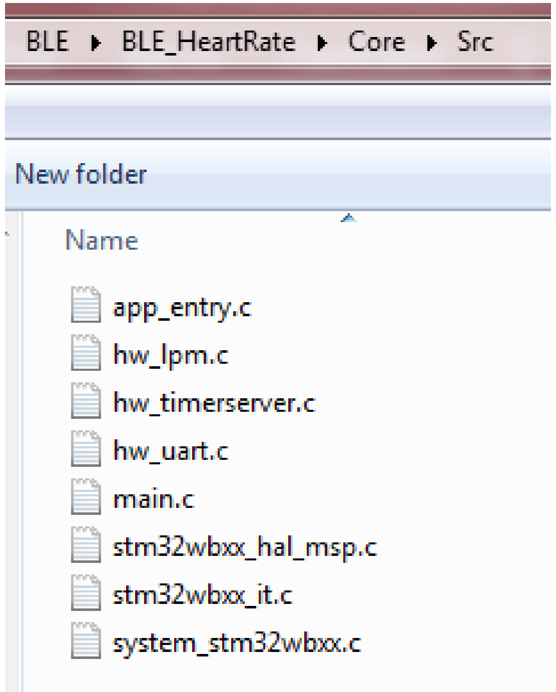
- Mac_802_15_4_FFD
- Mac_802_15_4_RFD

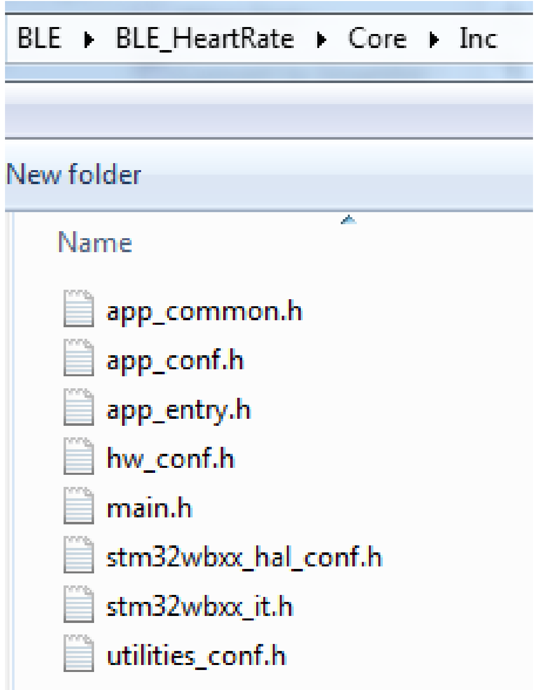
life.augmented

STM32 WB

Core **folder contains application-related source code**







CubeWB firmware

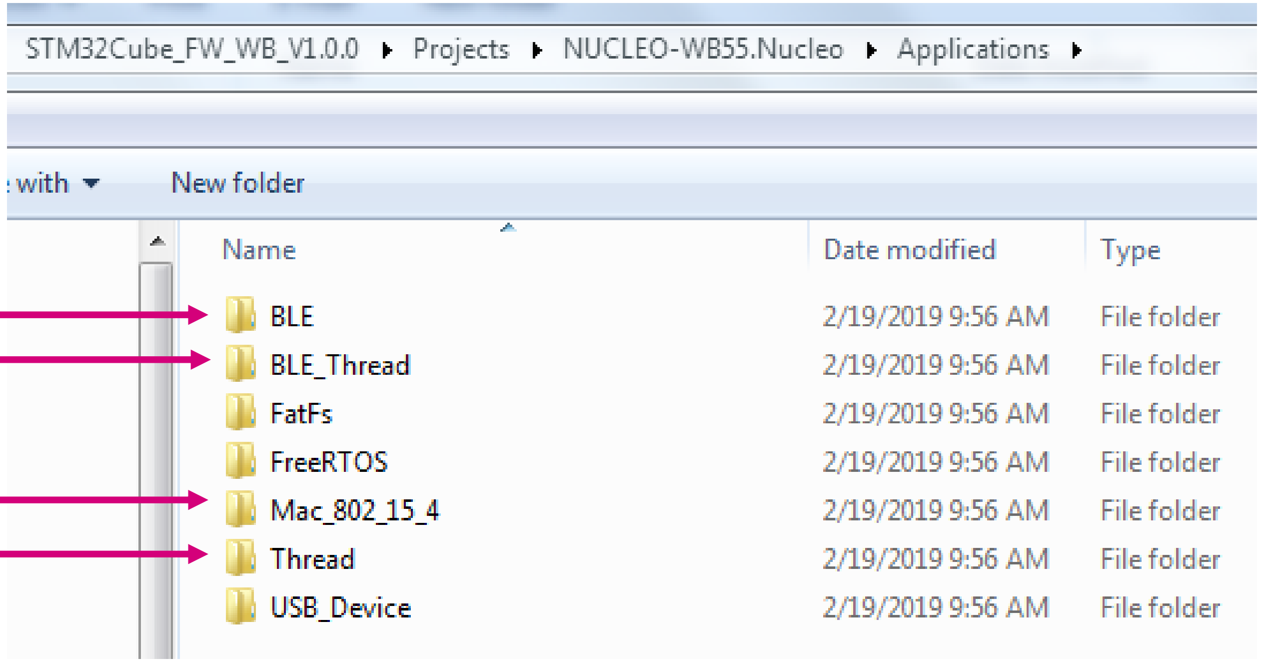
Different stacks required for different application types

BLE projects

BLE + Thread Static Concurrent mode project

Open MAC project

Thread projects

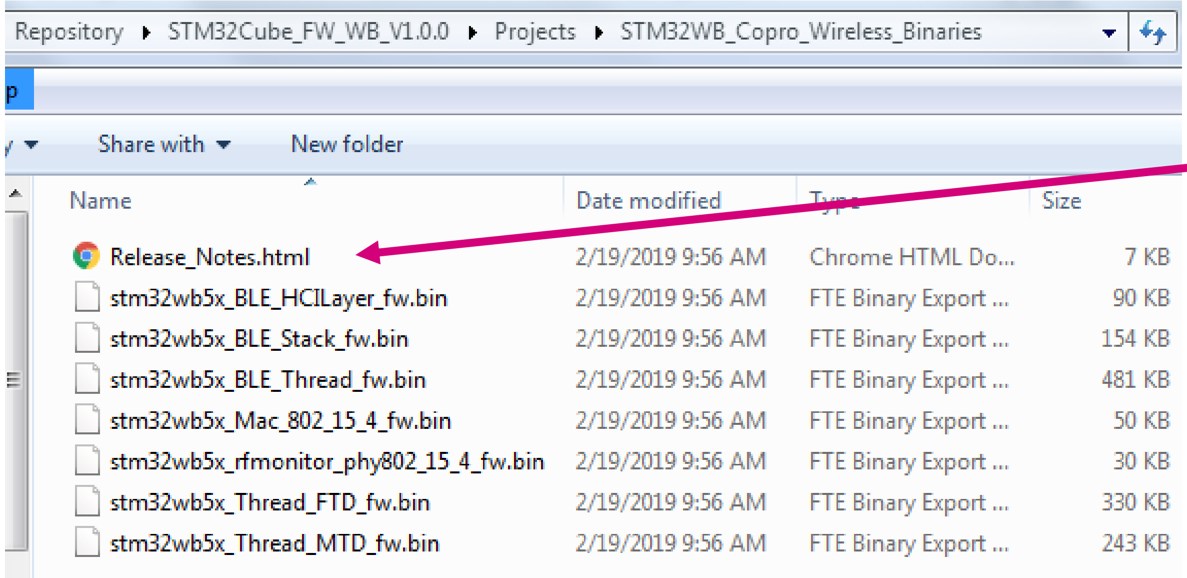


Zigbee 3.0 coming soon!

life.augmented

STM32 WB

Encrypted radio stack binaries here

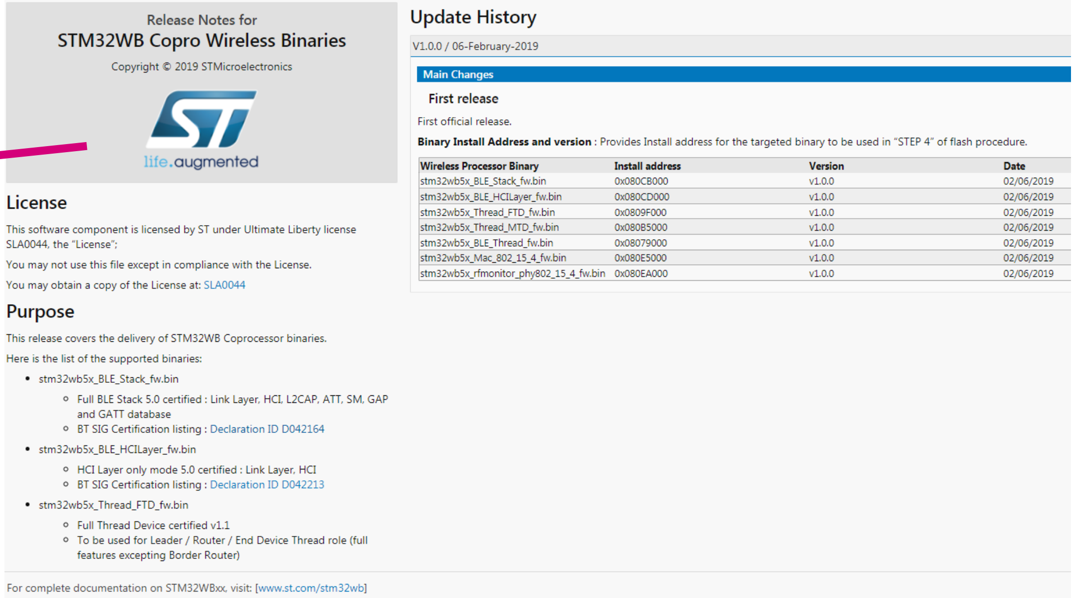


Repository ▸ STM32Cube_FW_WB_V1.0.0 ▸ Projects ▸ STM32WB_Copro_Wireless_Binaries

Share with ▾ New folder

Name	Date modified	Type	Size
Release_Notes.html	2/19/2019 9:56 AM	Chrome HTML Do...	7 KB
stm32wb5x_BLE_HCILayer_fw.bin	2/19/2019 9:56 AM	FTE Binary Export ...	90 KB
stm32wb5x_BLE_Stack_fw.bin	2/19/2019 9:56 AM	FTE Binary Export ...	154 KB
stm32wb5x_BLE_Thread_fw.bin	2/19/2019 9:56 AM	FTE Binary Export ...	481 KB
stm32wb5x_Mac_802_15_4_fw.bin	2/19/2019 9:56 AM	FTE Binary Export ...	50 KB
stm32wb5x_rfmonitor_phy802_15_4_fw.bin	2/19/2019 9:56 AM	FTE Binary Export ...	30 KB
stm32wb5x_Thread_FTD_fw.bin	2/19/2019 9:56 AM	FTE Binary Export ...	330 KB
stm32wb5x_Thread_MTD_fw.bin	2/19/2019 9:56 AM	FTE Binary Export ...	243 KB

HTML file details update procedure



Release Notes for
STM32WB Copro Wireless Binaries

Copyright © 2019 STMicroelectronics

life.augmented

License

This software component is licensed by ST under Ultimate Liberty license SLA0044, the "License";

You may not use this file except in compliance with the License.

You may obtain a copy of the License at: SLA0044

Purpose

This release covers the delivery of STM32WB Coprocessor binaries.

Here is the list of the supported binaries:

- stm32wb5x_BLE_Stack_fw.bin
  - Full BLE Stack 5.0 certified : Link Layer, HCI, L2CAP, ATT, SM, GAP and GATT database
  - BT SIG Certification listing : Declaration ID D042164
- stm32wb5x_BLE_HCILayer_fw.bin
  - HCI Layer only mode 5.0 certified : Link Layer, HCI
  - BT SIG Certification listing : Declaration ID D042213
- stm32wb5x_Thread_FTD_fw.bin
  - Full Thread Device certified v1.1
  - To be used for Leader / Router / End Device Thread role (full features excepting Border Router)

Update History

V1.0.0 / 06-February-2019

Main Changes

First release

First official release.

Binary Install Address and version : Provides Install address for the targeted binary to be used in "STEP 4" of flash procedure.

Wireless Processor Binary	Install address	Version	Date
stm32wb5x_BLE_Stack_fw.bin	0x080CB000	v1.0.0	02/06/2019
stm32wb5x_BLE_HCILayer_fw.bin	0x080CD000	v1.0.0	02/06/2019
stm32wb5x_Thread_FTD_fw.bin	0x0809F000	v1.0.0	02/06/2019
stm32wb5x_Thread_MTD_fw.bin	0x080B5000	v1.0.0	02/06/2019
stm32wb5x_BLE_Thread_fw.bin	0x08079000	v1.0.0	02/06/2019
stm32wb5x_Mac_802_15_4_fw.bin	0x080E5000	v1.0.0	02/06/2019
stm32wb5x_rfmonitor_phy802_15_4_fw.bin	0x080EA000	v1.0.0	02/06/2019

For complete documentation on STM32WBxx, visit: [www.st.com/stm32wb]

Nucleo & Dongle boards come preloaded with the BLE stack

STM32CubeMonitor-Power

$70

X-NUCLEO-LPM01A

STM32CubeIDE ACTIVE

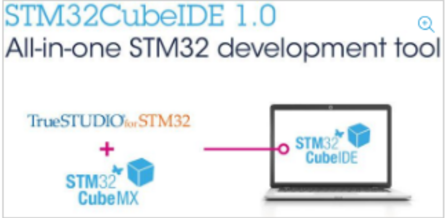
Print Save to MyST Share via Email

Integrated Development Environment for STM32

Get Software Download databrief

Overview Tools & Software Resources

STM32CubeIDE is an all-in-one multi-OS development tool, which is part of the STM32Cube software ecosystem.



STM32CubeIDE is an advanced C/C++ development platform with IP configuration, code generation, code compilation, and debug features for STM32 microcontrollers. It is based on the ECLIPSE™/CDT framework and GCC toolchain for the development, and GDB for the debugging. It allows the integration of the hundreds of existing plugins that complete the features of the ECLIPSE™ IDE.

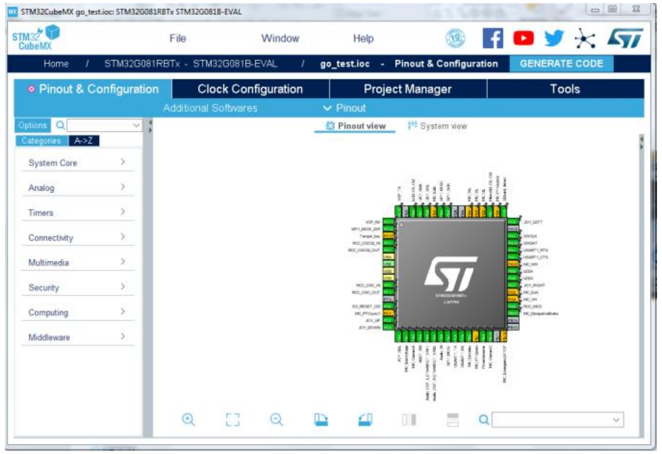
STM32CubeIDE integrates all STM32CubeMX functionalities to offer all-in-one tool experience and save installation and development time. After the selection of an empty STM32 MCU or preconfigured microcontroller from the selection of a board, the project is created and initialization code generated. At any time during the development, the user can return to the initialization and configuration of the IPs or middleware and regenerate the initialization code with no impact on the user code.

STM32CubeIDE includes build and stack analyzers that provide the user with useful information about project status and memory requirements.

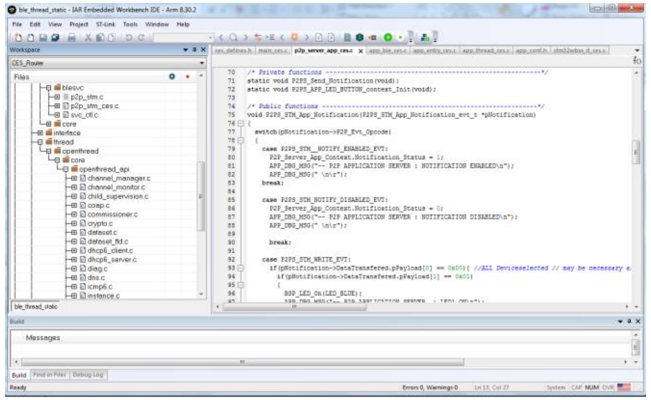
STM32CubeIDE also includes standard and advanced debugging features including views of CPU core registers, memories, and peripheral registers, as well as live variable watch, Serial Wire Viewer interface, or fault analyzer.

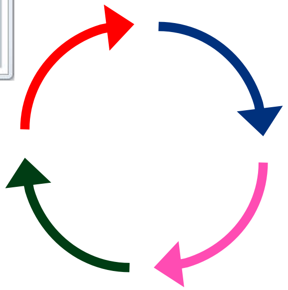
Iterative Design Process

Configure

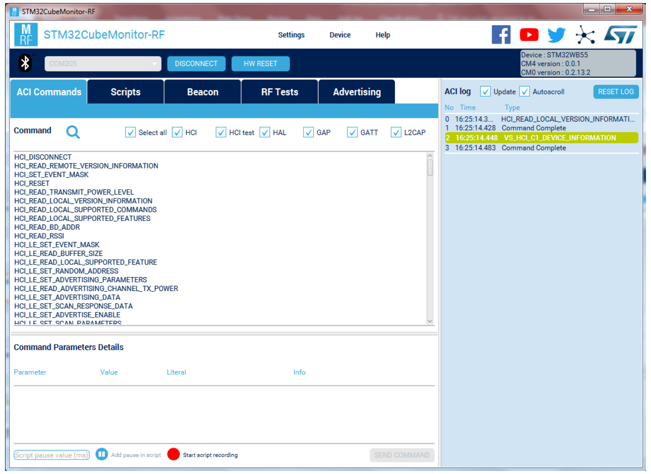


Code & Debug

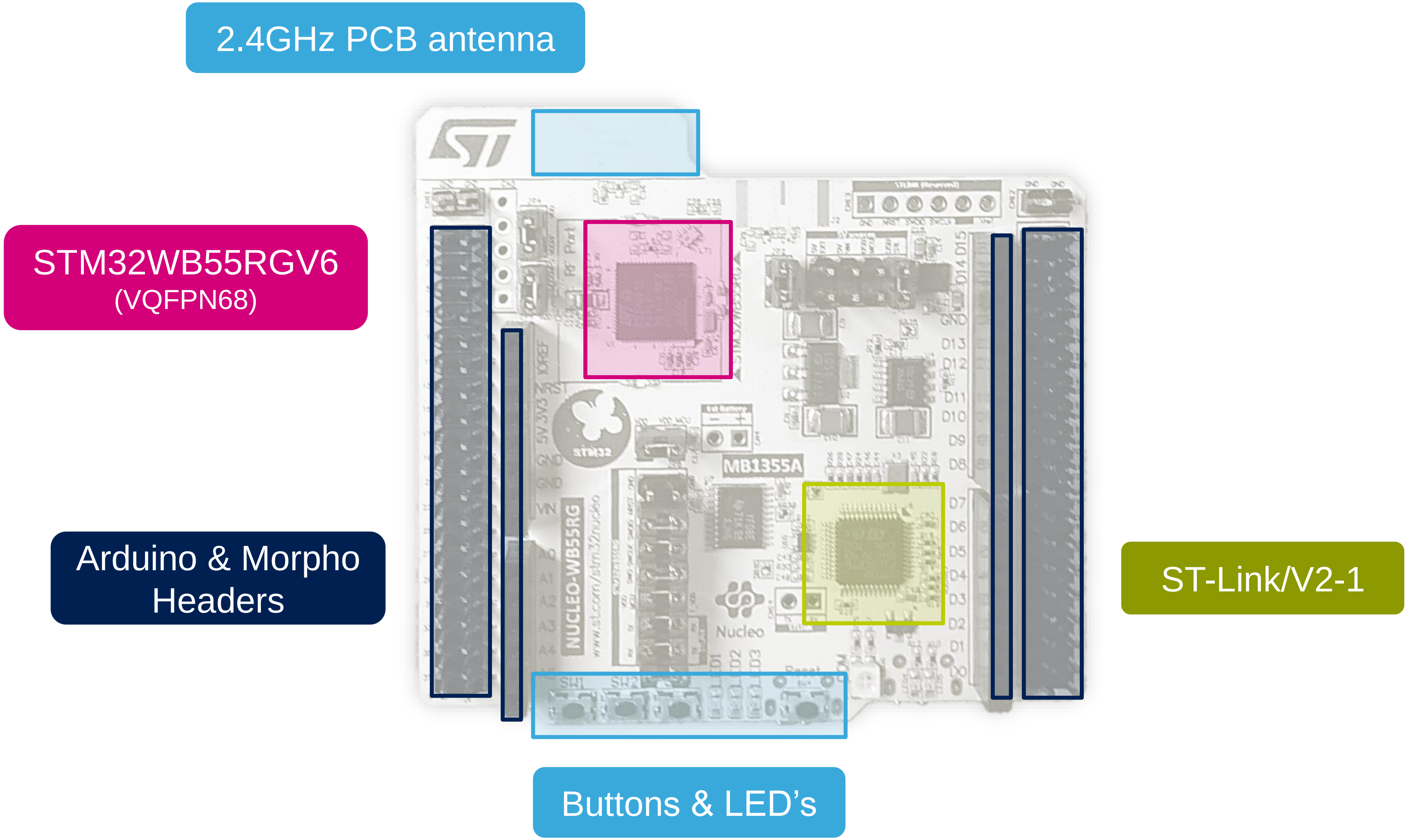


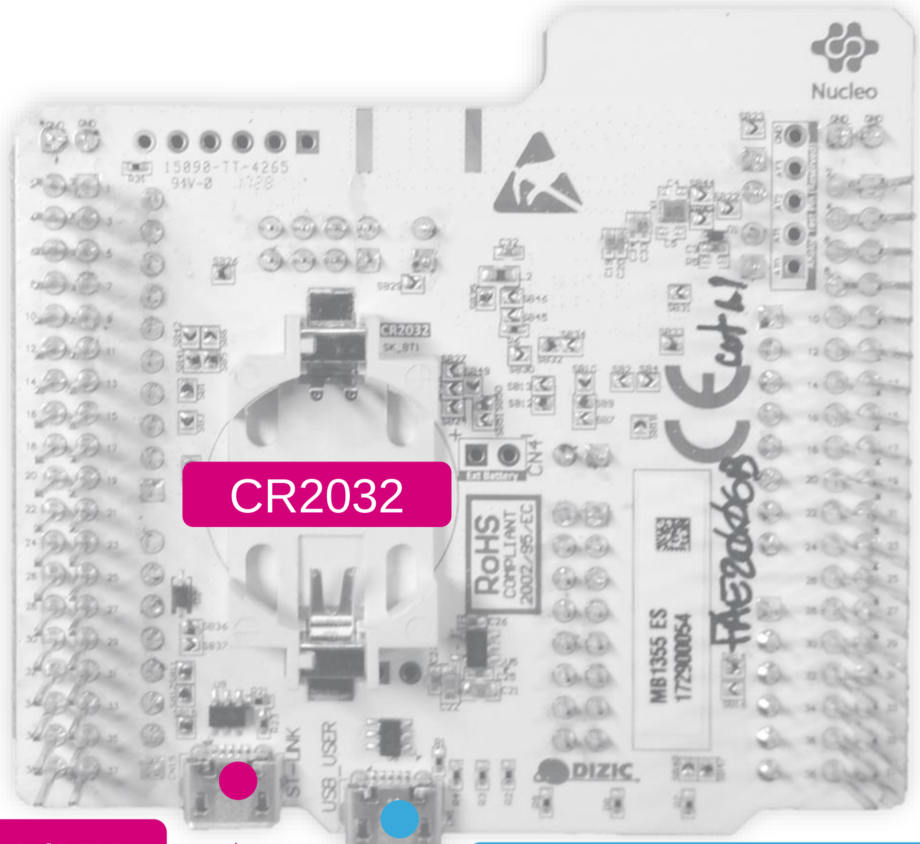


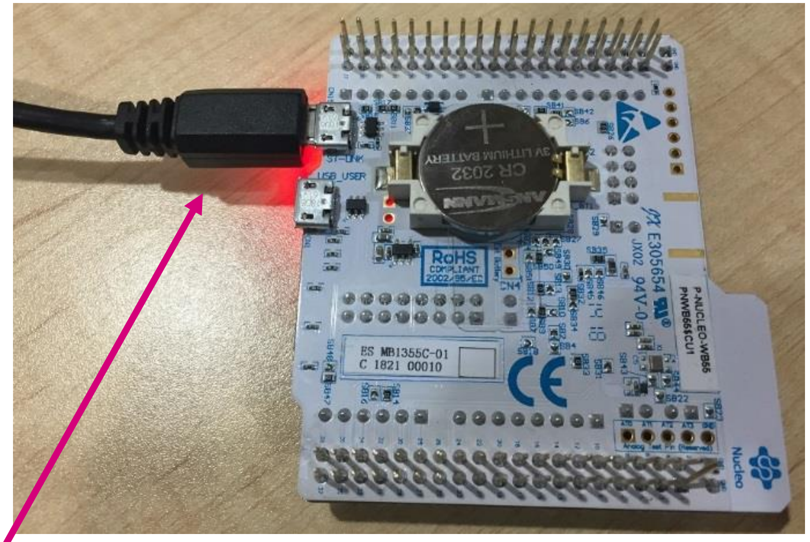
Test



Measure


2.4GHz PCB antenna
STM32WB55RGV6
(VQFPN68)
Arduino & Morpho
Headers
ST-Link/V2-1
Buttons & LED's

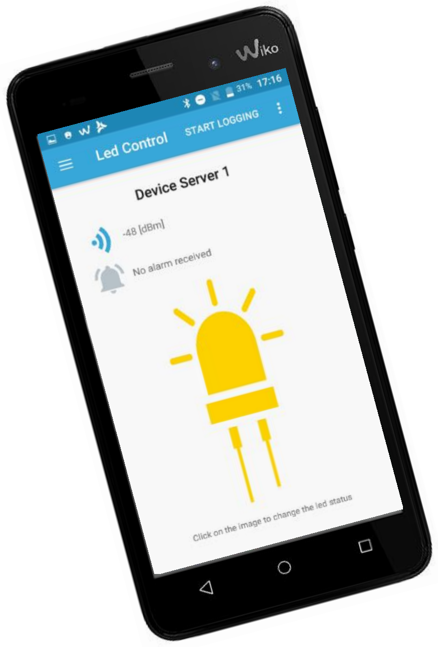




We will use this one!

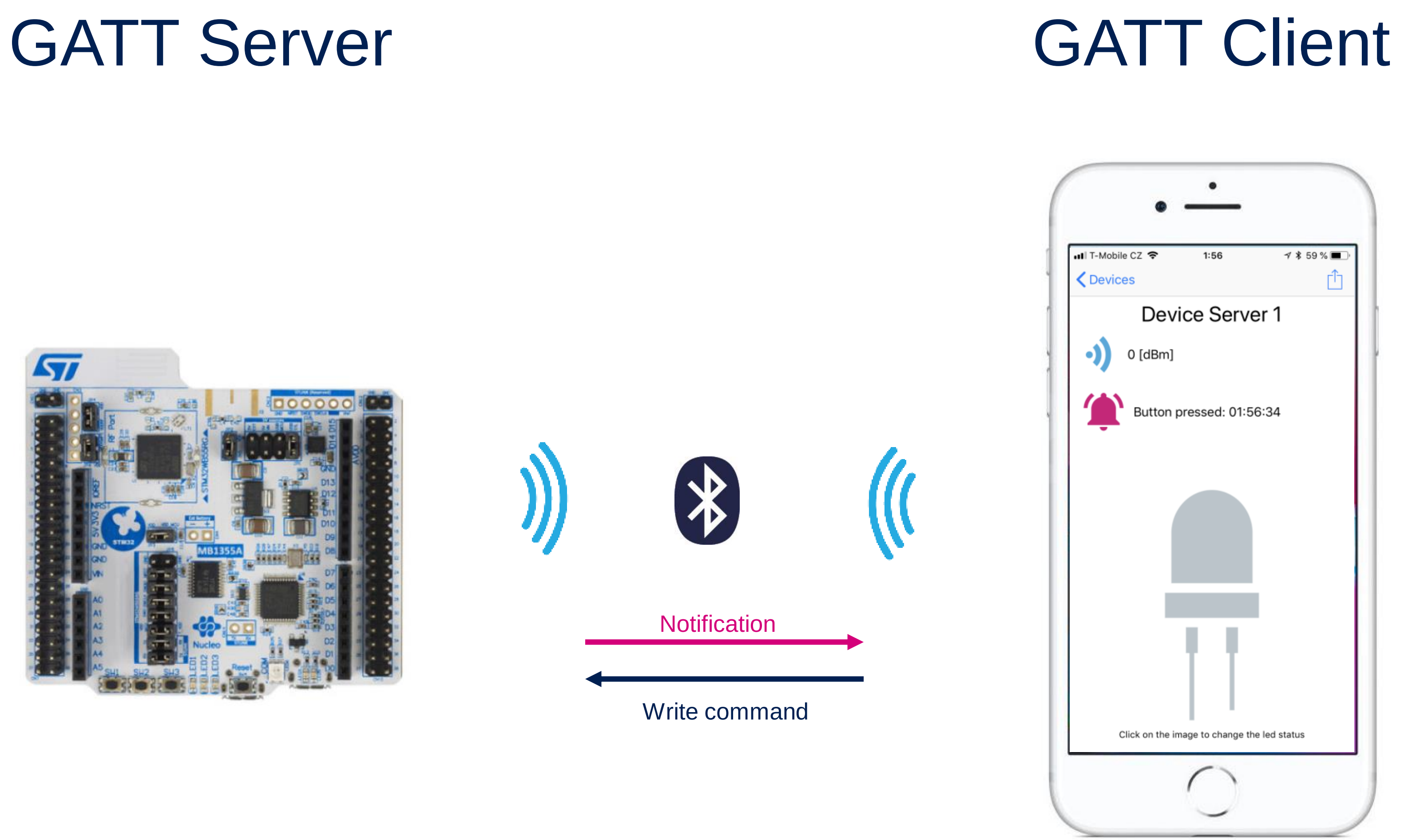
Hands-On

Out-of-the-Box

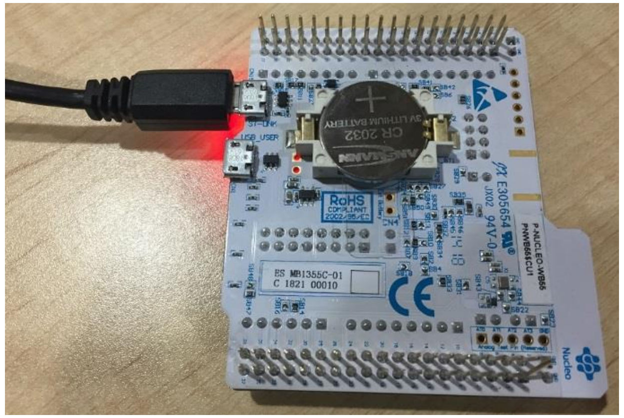


life.augmented

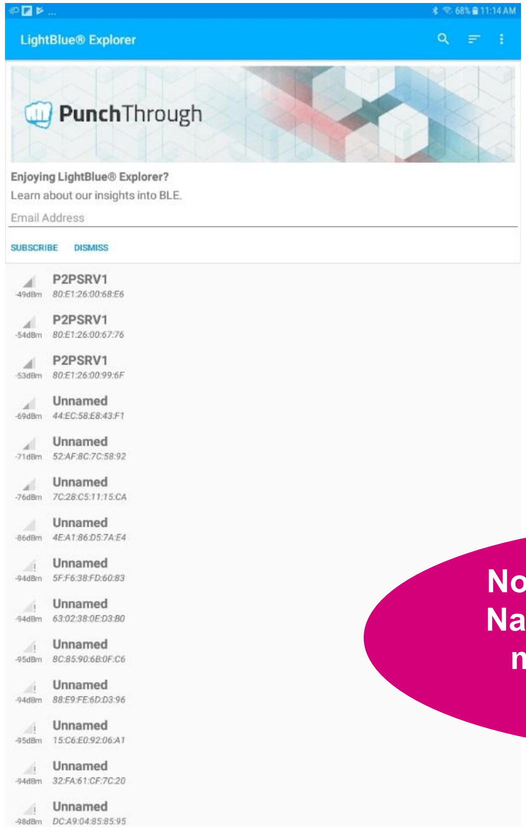
STM32 WB

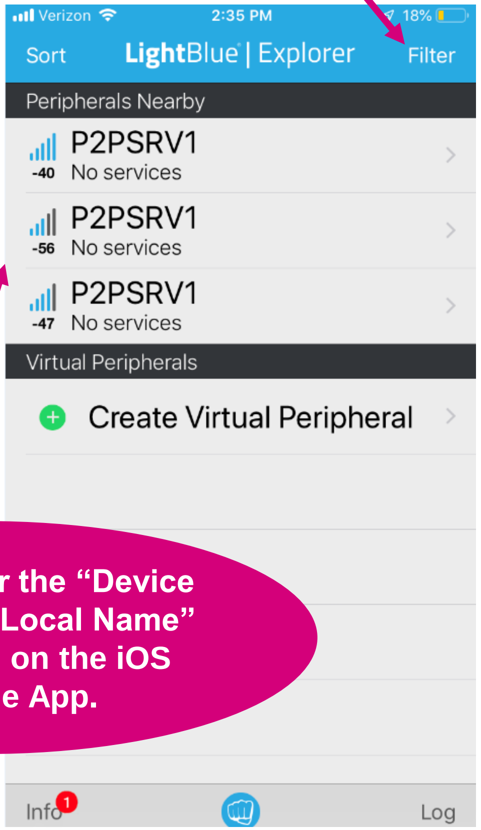

GATT Server
GATT Client
Notification
Write command
T-Mobile CZ
1:56
59 %
Devices
Device Server 1
0 [dBm]
Button pressed: 01:56:34
Click on the image to change the led status

- Power Nucleo board
- Launch ST BLE Sensor app
- What happens?

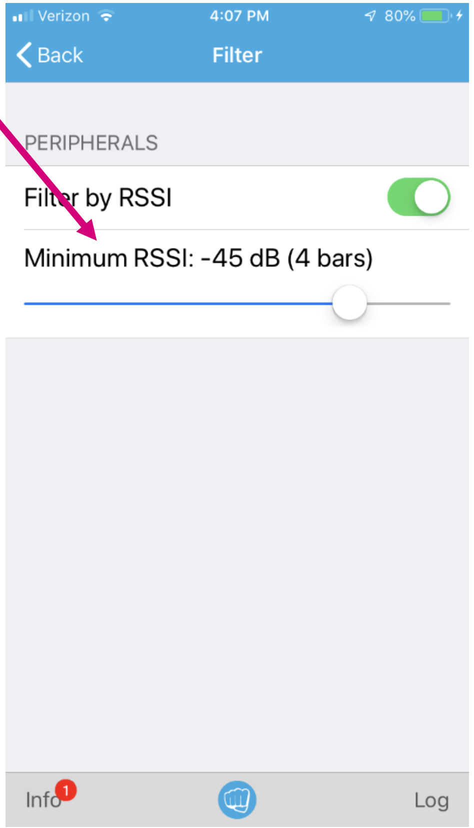


- Launch LightBlue Explorer app
- We can filter by RSSI to find our device (on iOS)



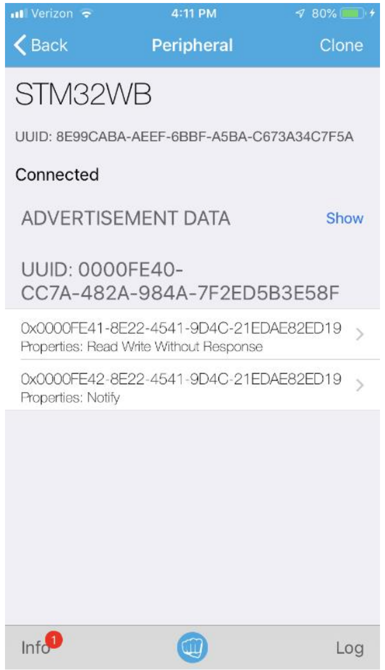


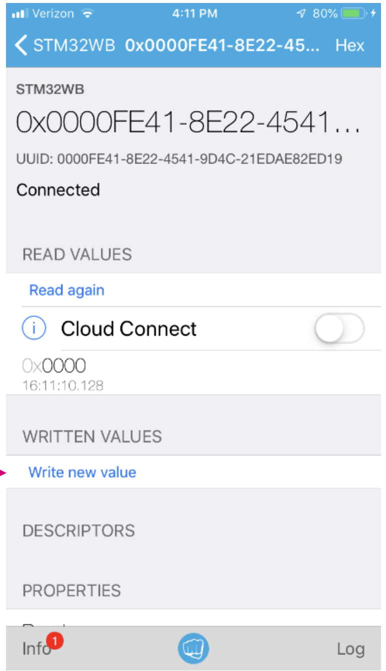
Note that either the "Device Name" or the "Local Name" may show up on the iOS LightBlue App.

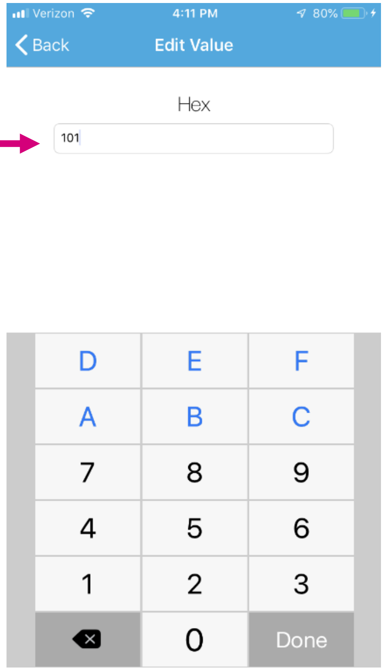


life.augmented

- Click on the “0x0000FE41-” characteristic
- Write new value – 101 hex
- Did the LED come on?
- 100 hex to turn it off





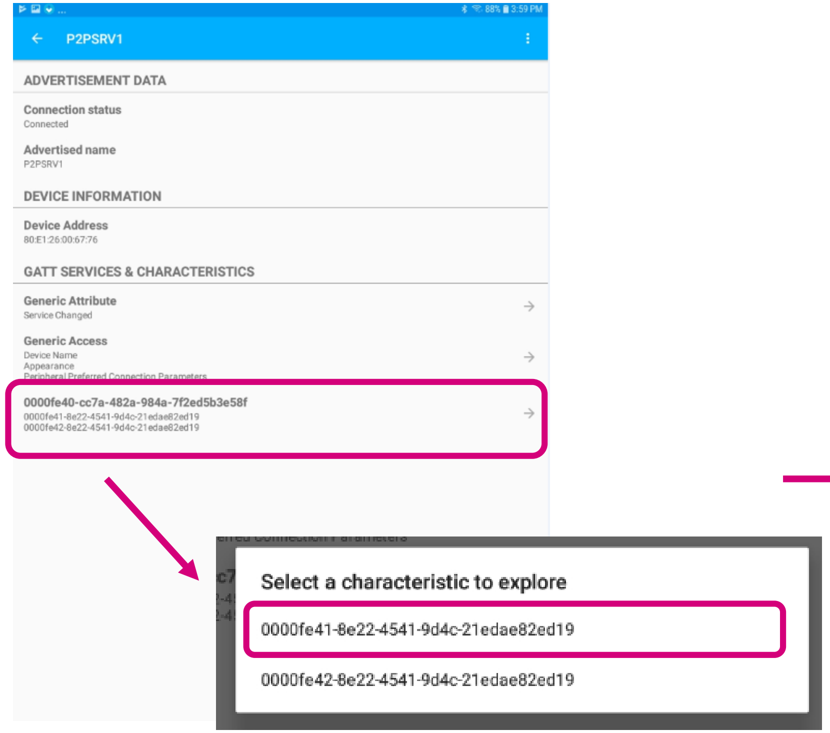


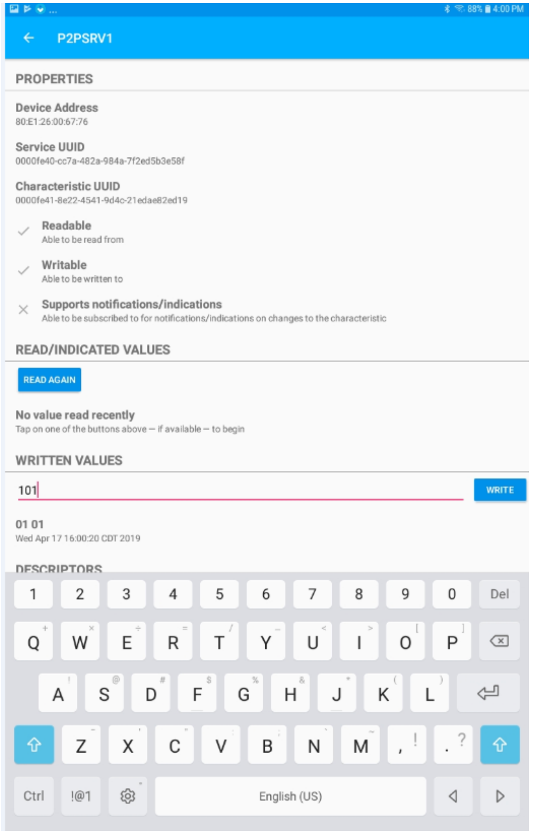
life.augmented

STM32 WB

- Click on the “0x0000FE41-” characteristic
- Write new value – 101 hex
- Did the LED come on?
- 100 hex to turn it off

Android

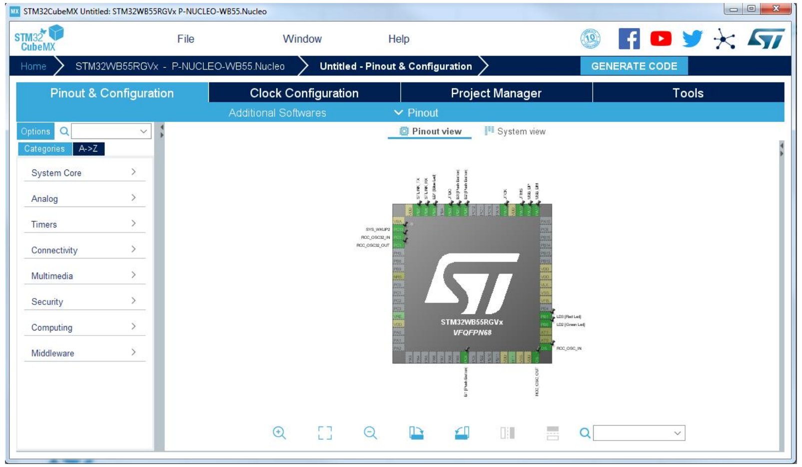




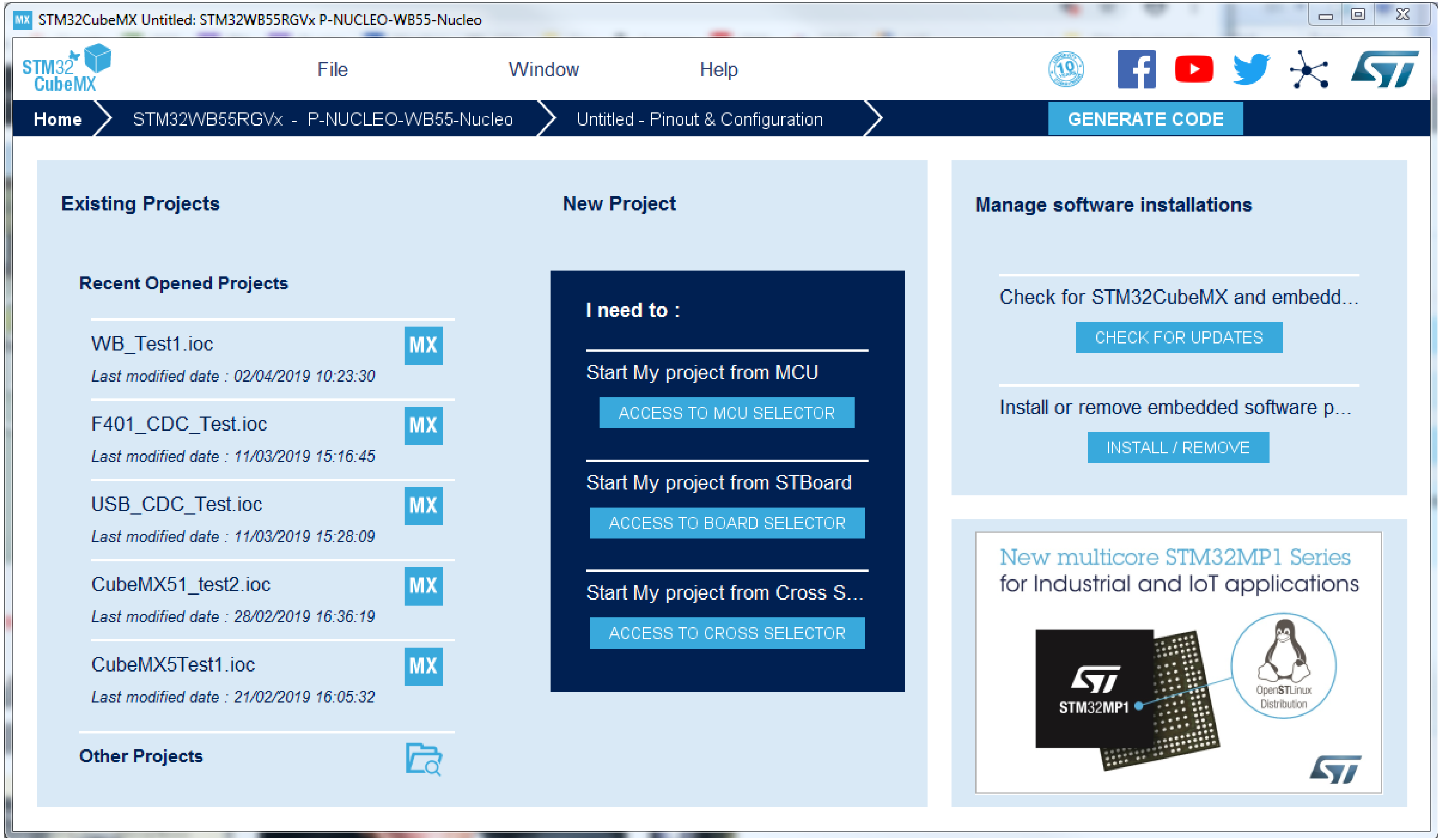
Your Magic number!

Hands-On

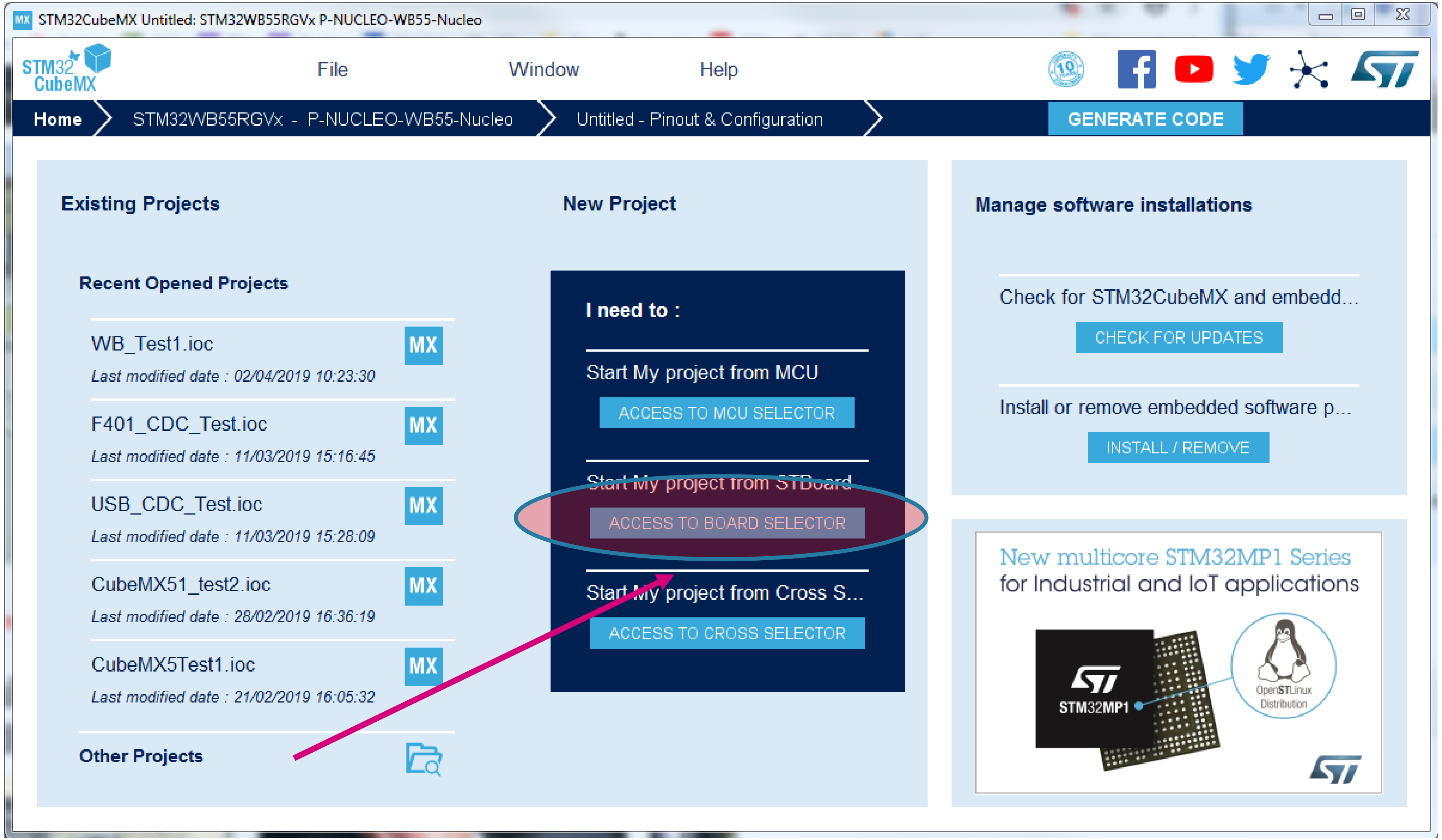
CubeMX



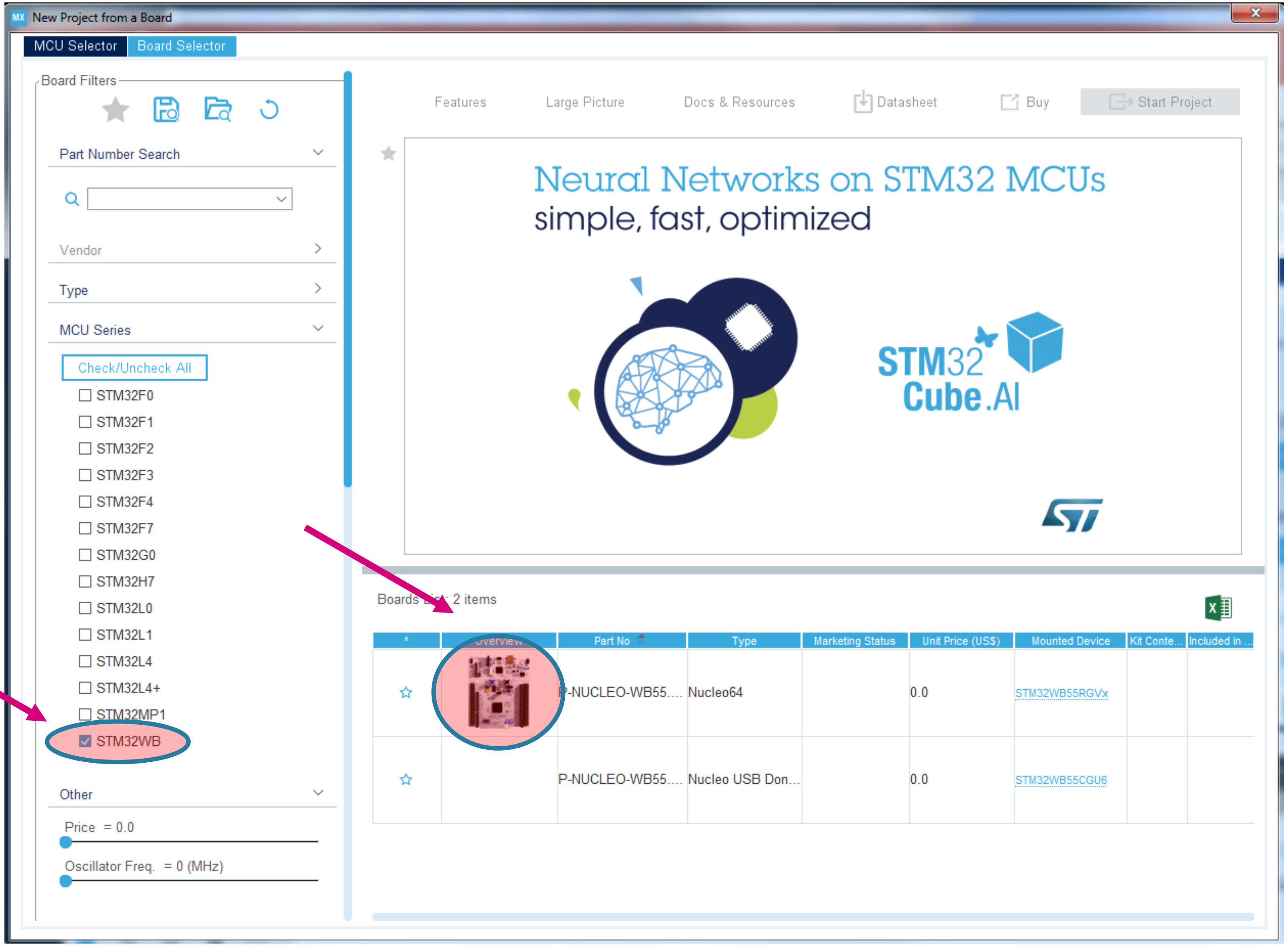
Launch CubeMX



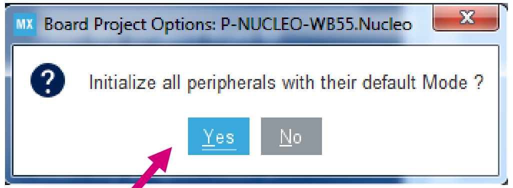
Launch CubeMX & Start project from Board Selector



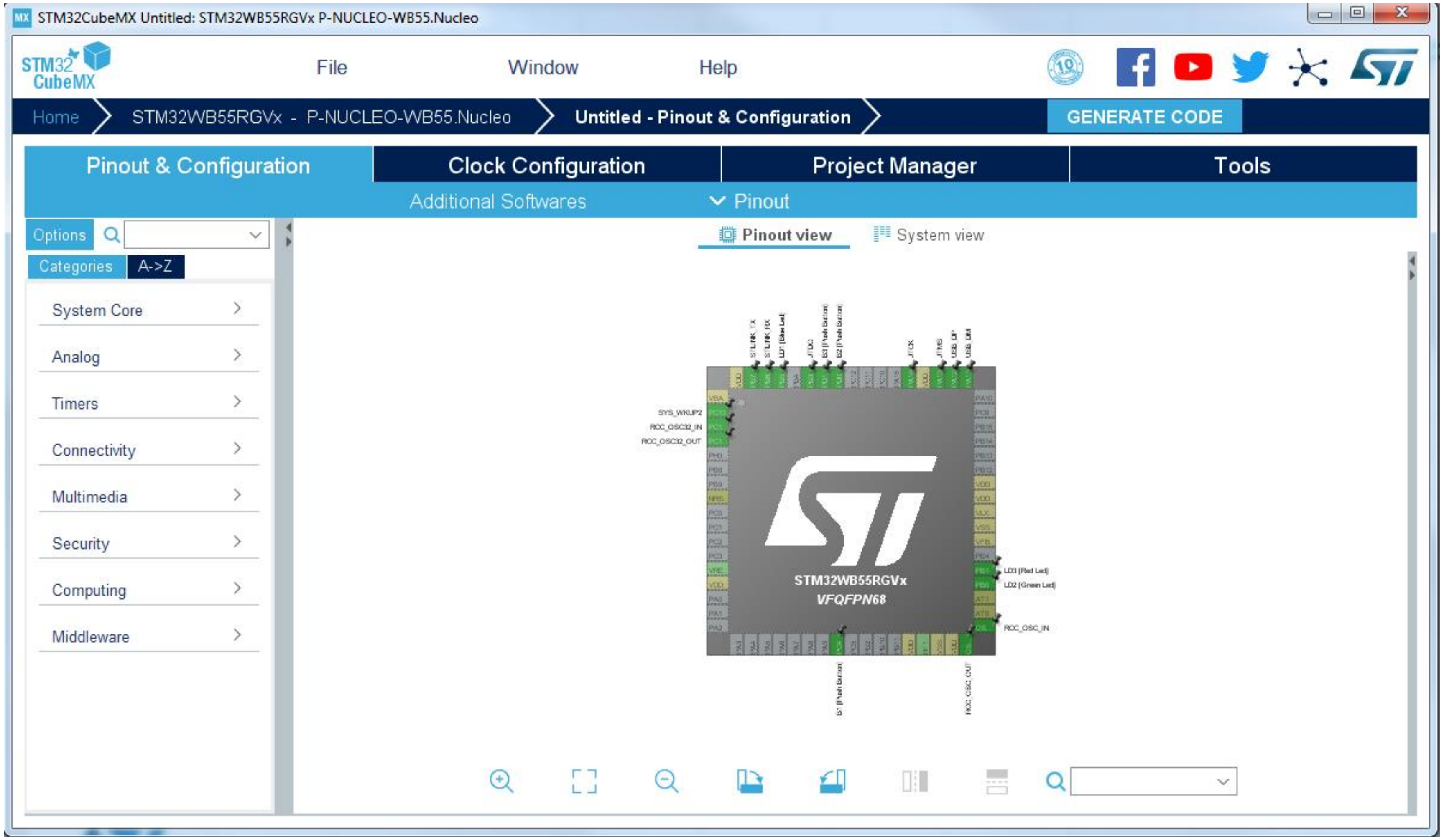
Filter by STM32WB and double-click on the Nucleo board!



P-NUCLEO-WB55 Board Project

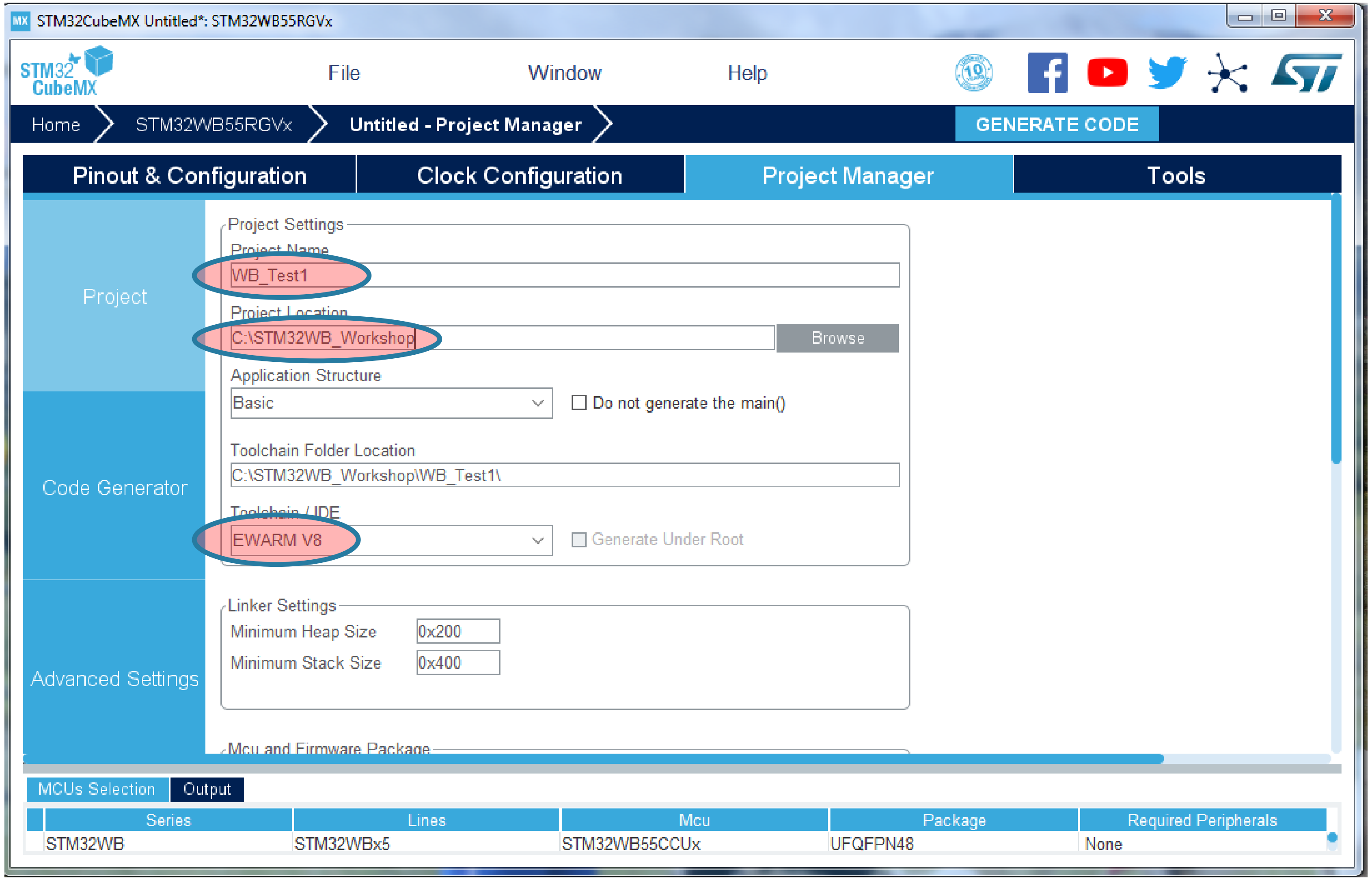


Yes

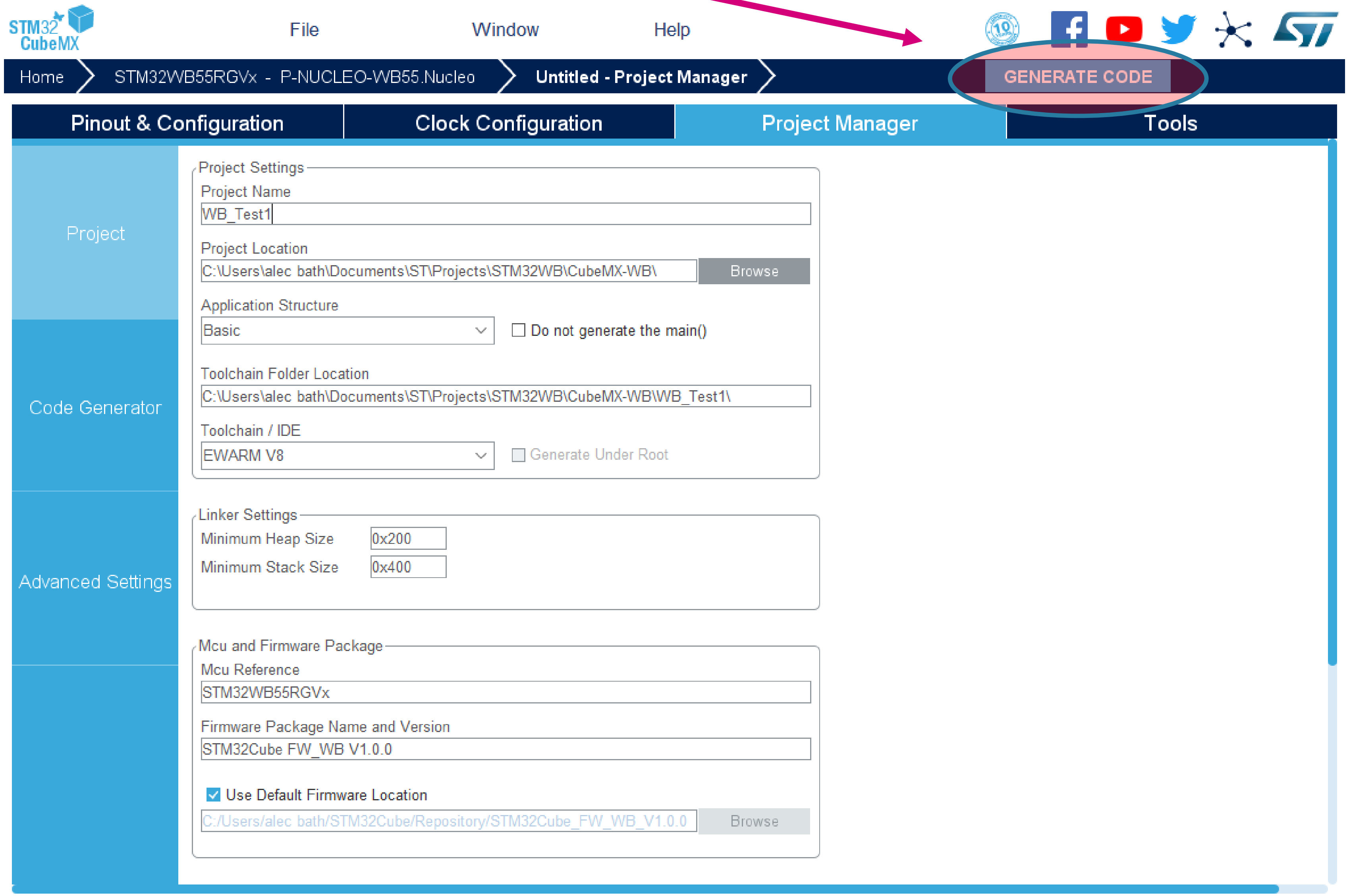


life.augmented

- Name your project
- Recommended Project location: C:\STM32WB_Workshop\
- Use EWARM V8 toolchain



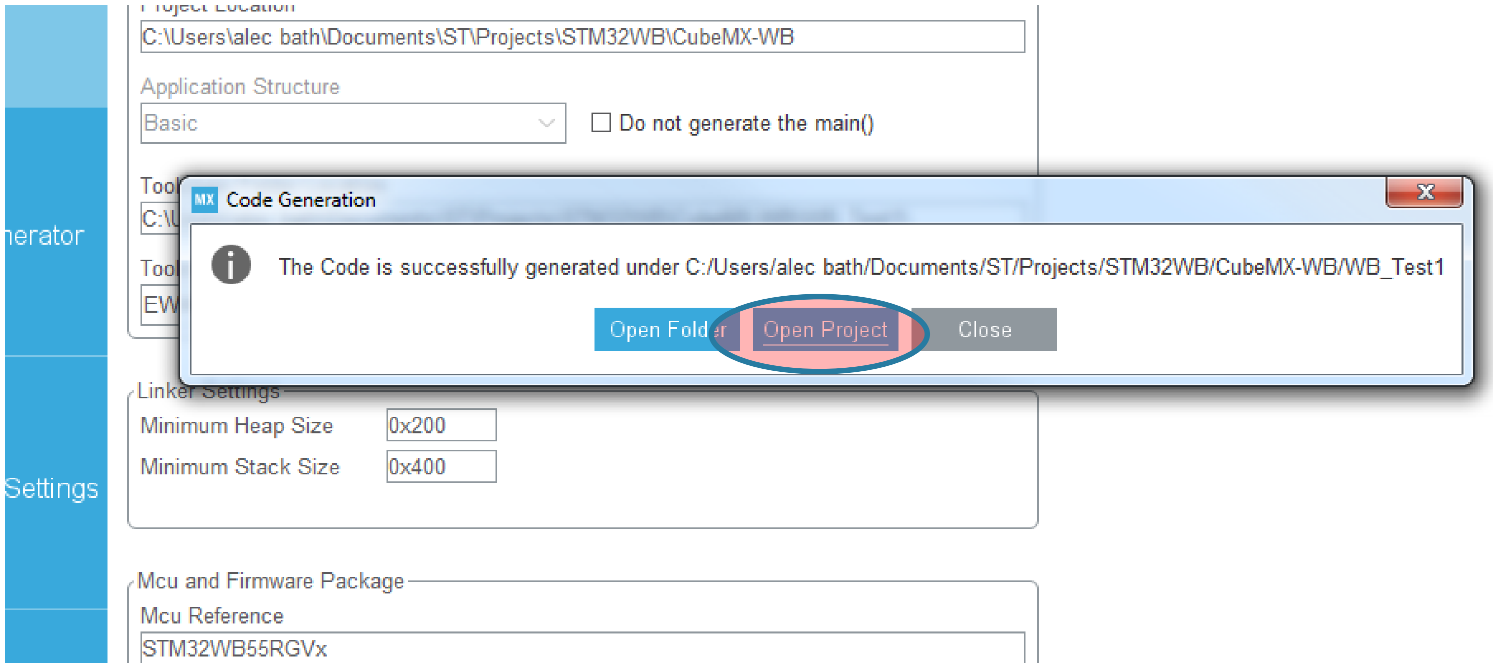
GENERATE CODE



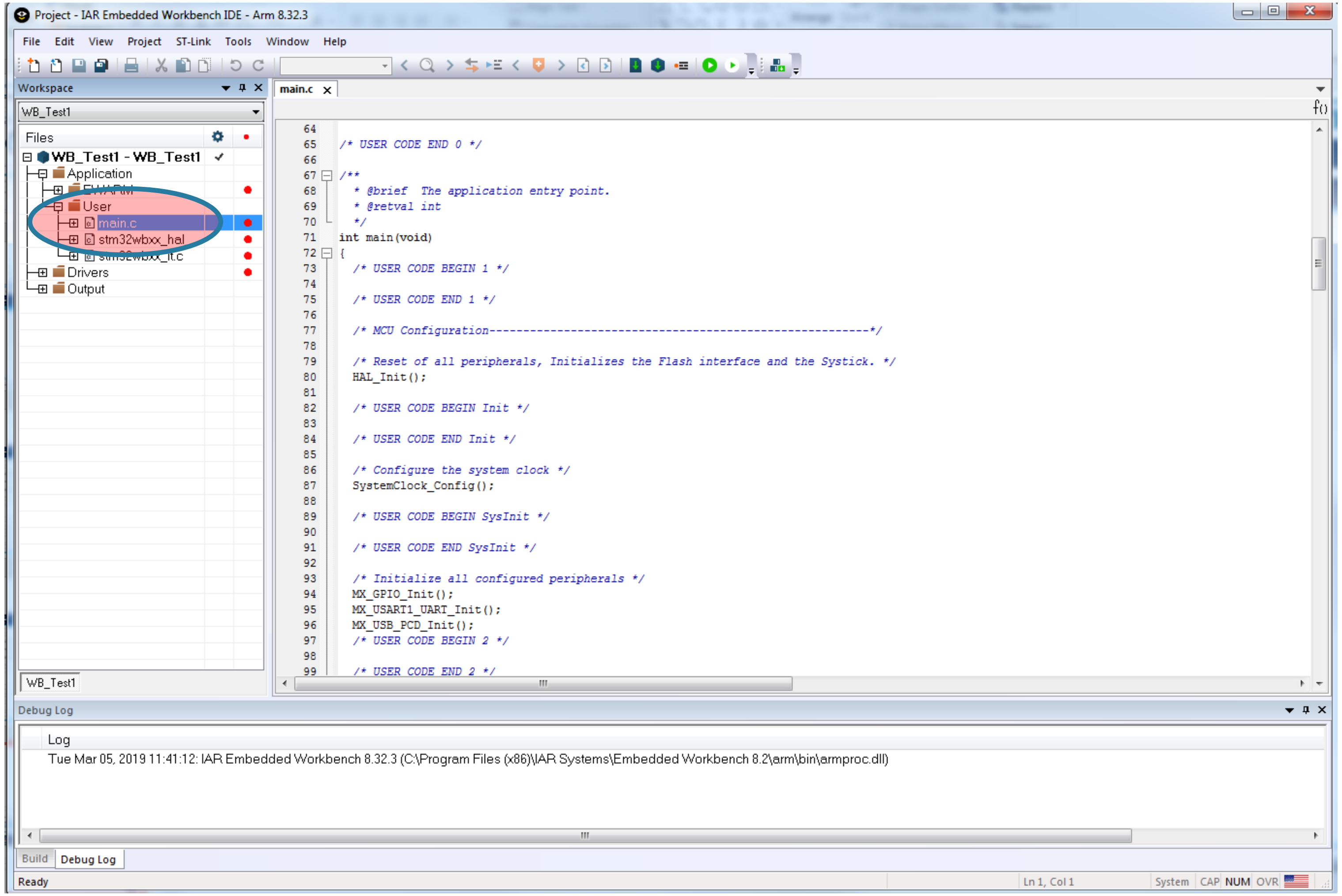
life.augmented

STM32 WB

- Open Project



- Expand the **User** file tree and Open **main.c**
- Enable line numbers in IAR (Tools > Options > Editor > Show line numbers)

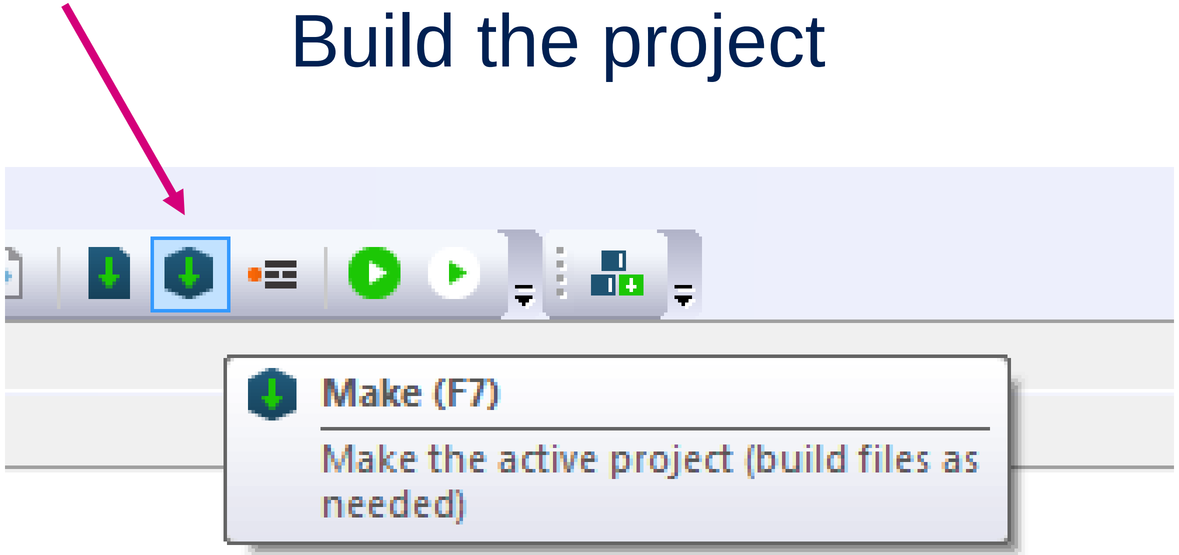


Add some code to while(1) loop:

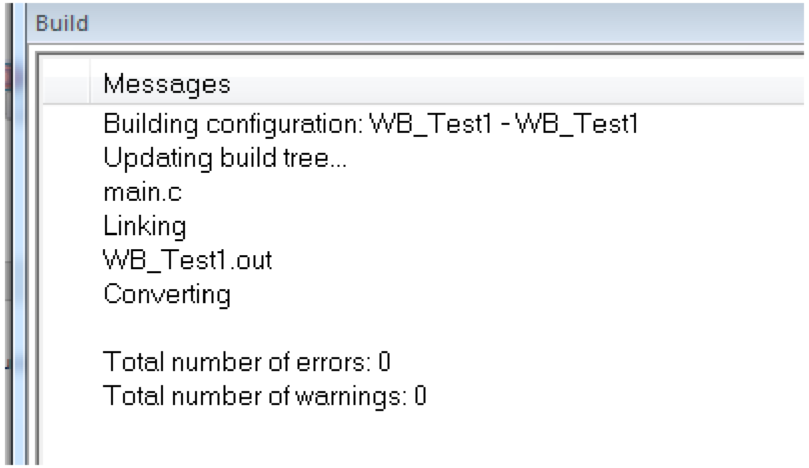
You can copy/paste the code bits from ***LED_Blinky_Lab.txt*** *file from your* ***Labs*** *folder*

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{

  HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_5);
  HAL_Delay(100);
  HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0);
  HAL_Delay(100);
  HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_1);
  HAL_Delay(100);

  /* USER CODE END WHILE */

```

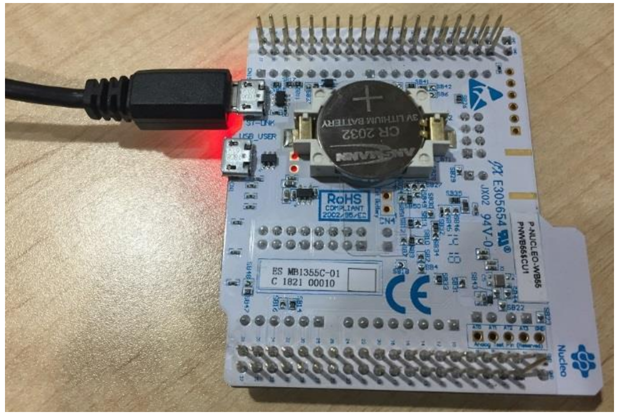
Build the project

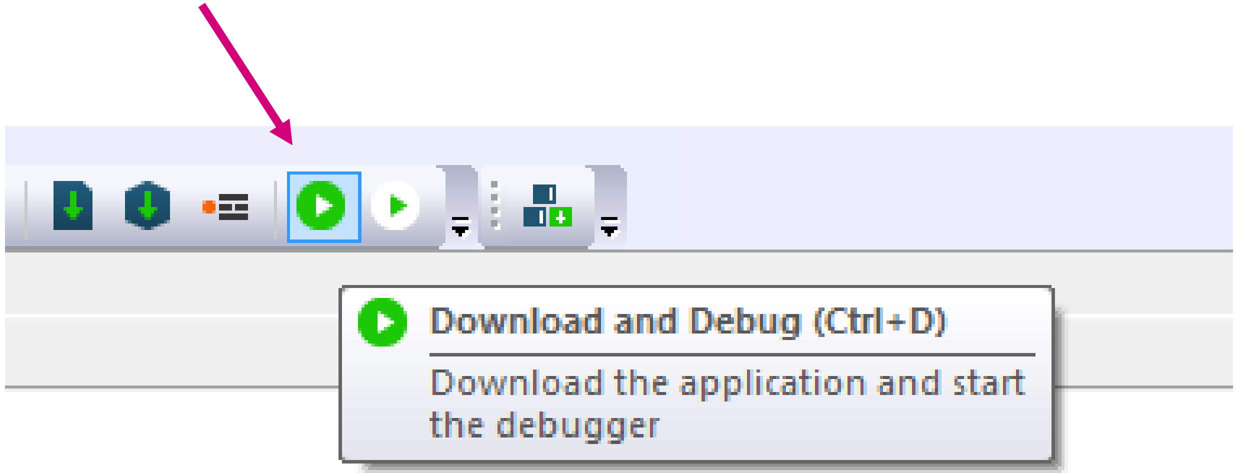


Check for errors

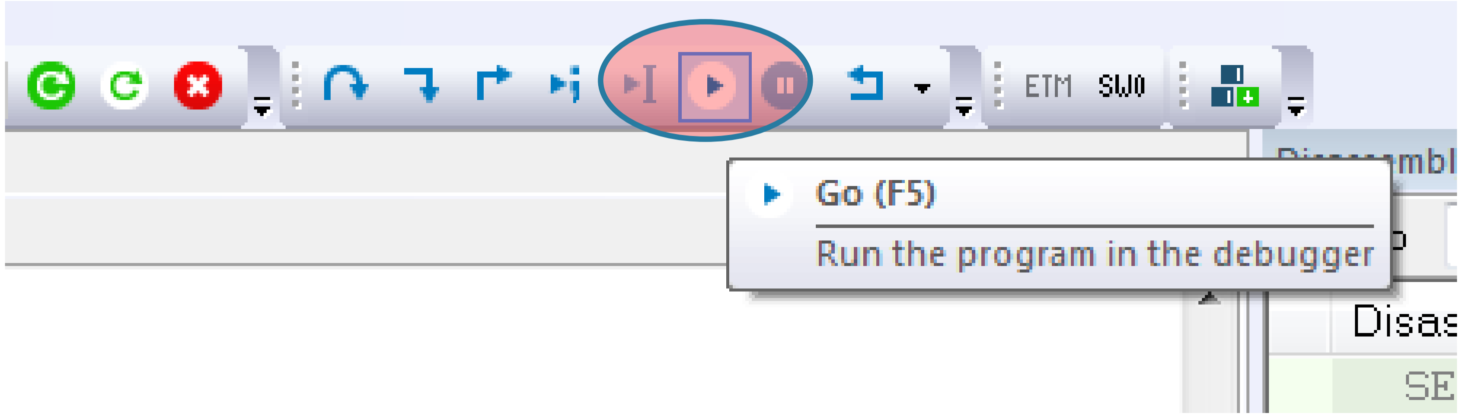


Download & Debug (attach your board) ☺

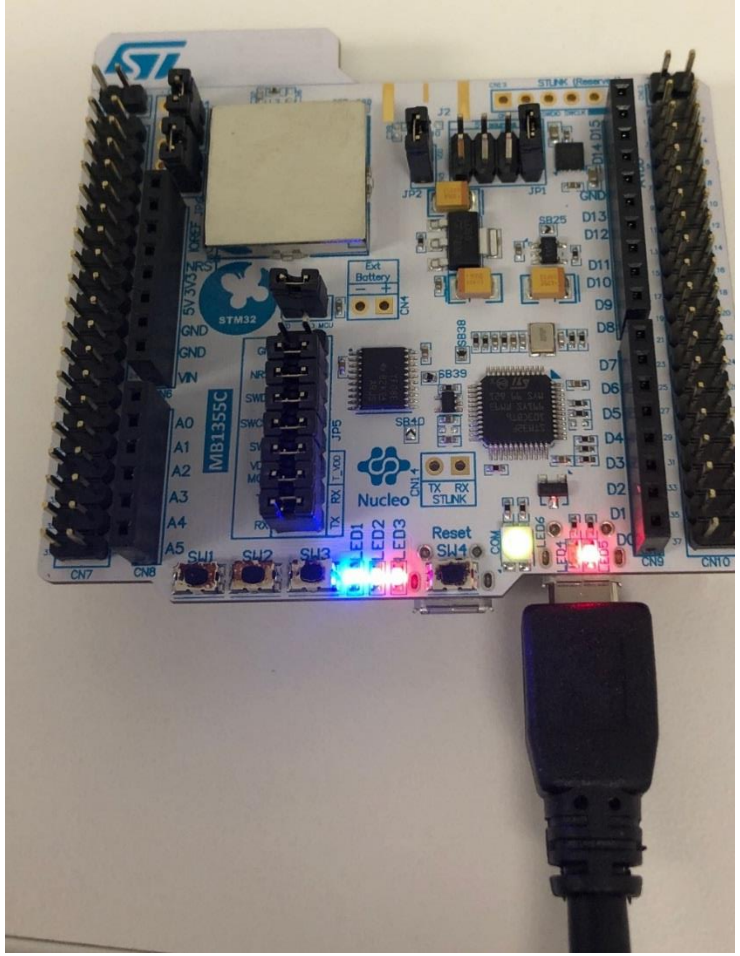




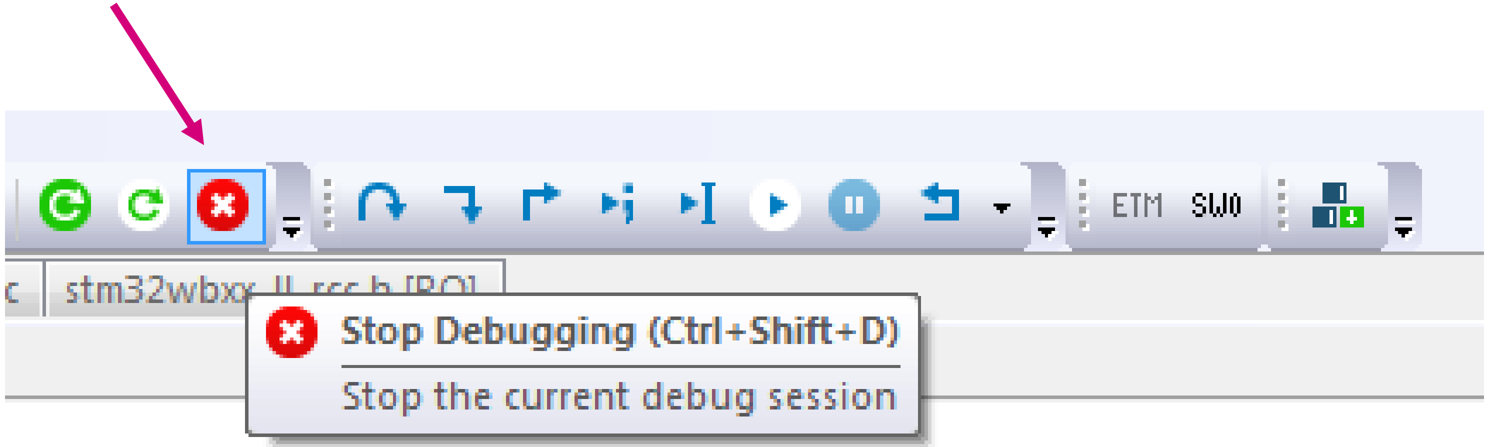
GO!



Enjoy the dancing LED’s! ☺

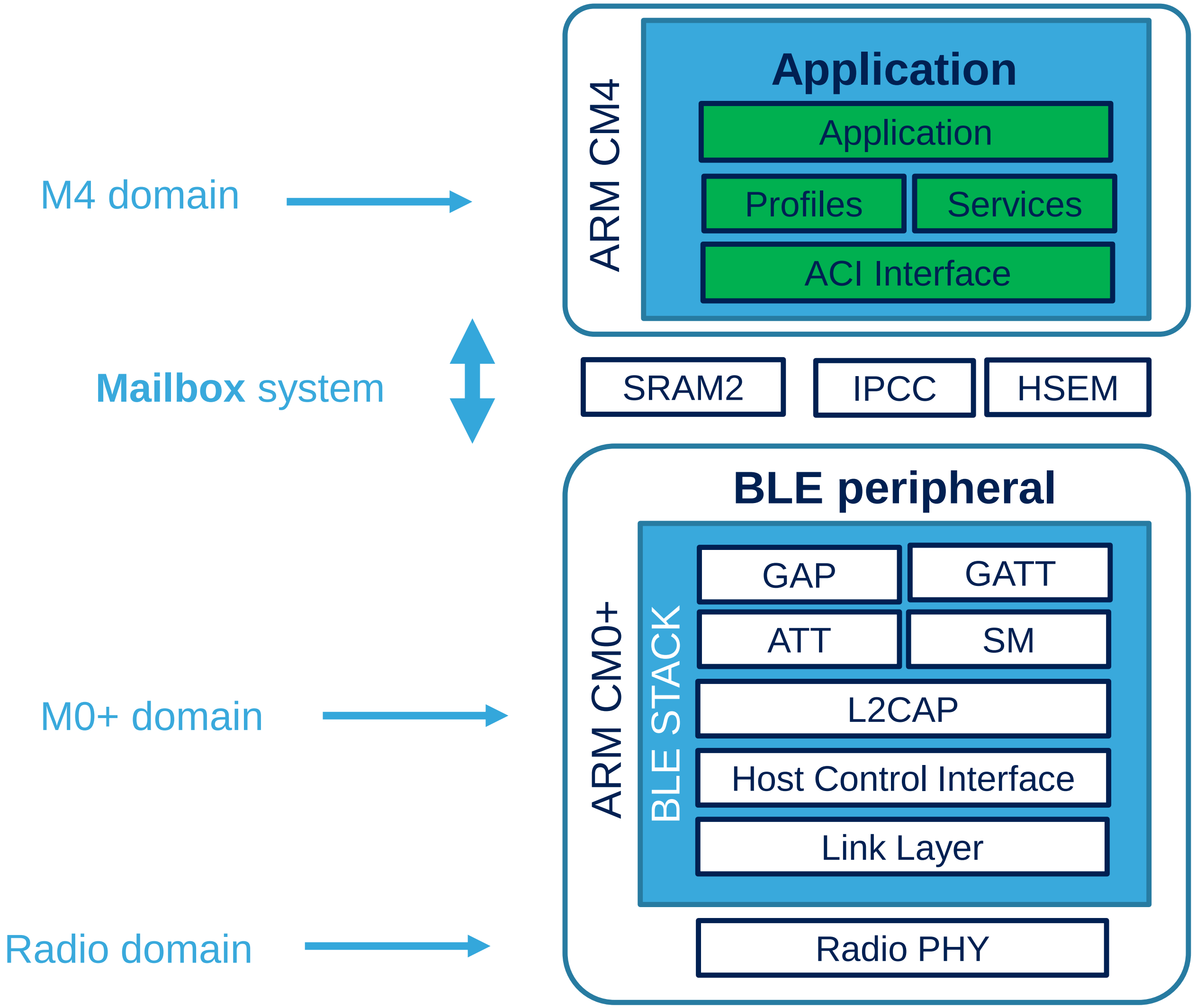


Stop Debugging at the end of each lab and close IAR Embedded Workbench



BLE Fundamentals

BLE Protocol Stack layers



Bluetooth Classic (BR/EDR) vs Low Energy (LE)

100X lower

Longer range

Fast connection (only 3 advertising channels to scan)

Relaxed RF requirements (lower cost silicon / passives)

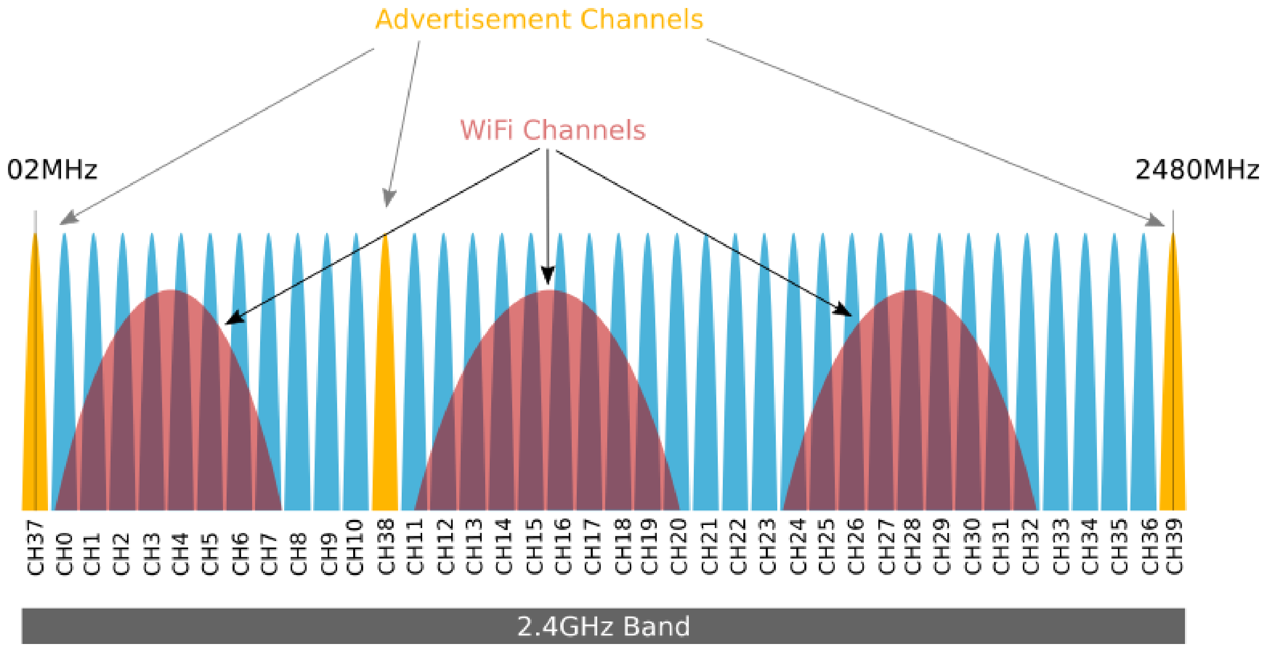
Comparison of Classic and Low Energy		
	Classic (BR/EDR)	**Low Energy (LE)**
Application	Cell phones, headsets, stereo/audio streaming, automotive (handsfree), PCs, etc.	Smartwatches, sport & fitness, home electronics, automation, industry, healthcare, smartphones, etc.
Voice	Yes	No
RF band ISM	2.4 GHz	2.4 GHz
Energy consumption	Reference	0.5…0.01 times Classic as reference
Coverage	10 m	≥ 10 m
Power	3 classes (max.): • +20 dBm • +4 dBm • 0 dBm	max. + 20 dBm four informative classes
Connection	Inquiry Yes, always hopping	Advertising Connection only if necessary, then hopping
Connection setup	100 ms	6 ms
RF channels	79 with 1 MHz spacing	40 with 2 MHz spacing • 3 advertising • 37 data (+ secondary advertis-ing)
Modulation	GFSK • BT = 0.5 • Deviation = 160 kHz • Mod index = 0.28….0.35 π/4-DQPSK 8DPSK	GFSK • BT = 0.5 • Deviation = 250 kHz or 500 kHz • Mod index = 0.45…0.55 • Stab Mod index = 0.495…0.505
Gross data rate	1…3 Mbit/s	1…2 Mbit/s
Application data rate	0.7…2.1 Mbit/s	0.2…0.6 Mbit/s

life.augmented

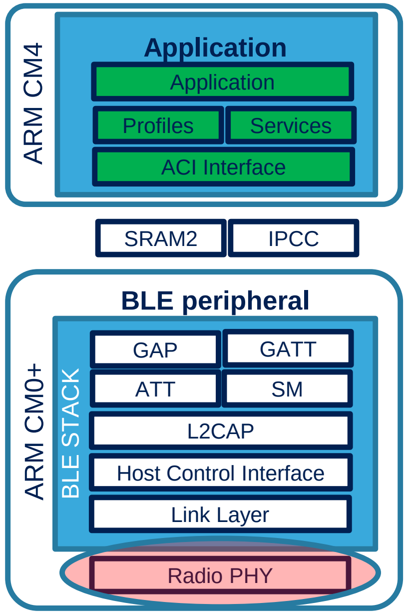
STM32 WB

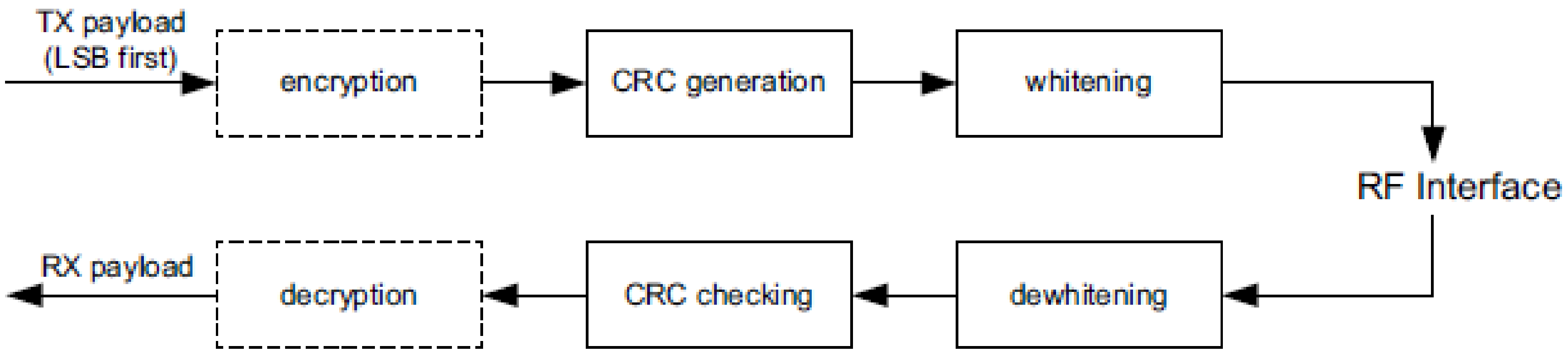
Strategically placed advertising channels

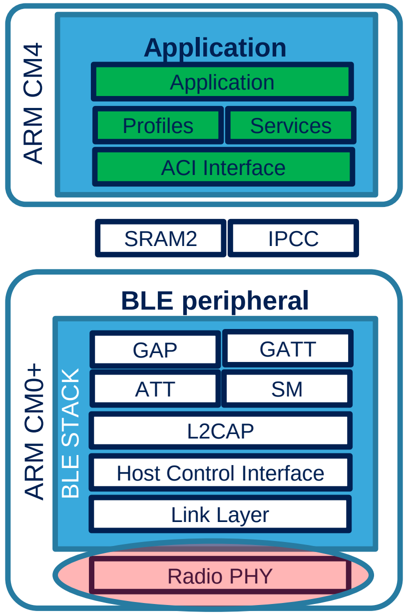
Remaining 37 channels are data channels



	BLE	Classic	
	BLE	BR	EDR
Modulation	GFSK 0.45 to 0.55	GFSK 0.28 to 0.35	DQPSK / 8DSPK
Data Rate	1Mbit/s	1 Mbit/s	2 and 3 Mbit/s
Channels	40	79	79
Spacing	2MHz	1MHz	-



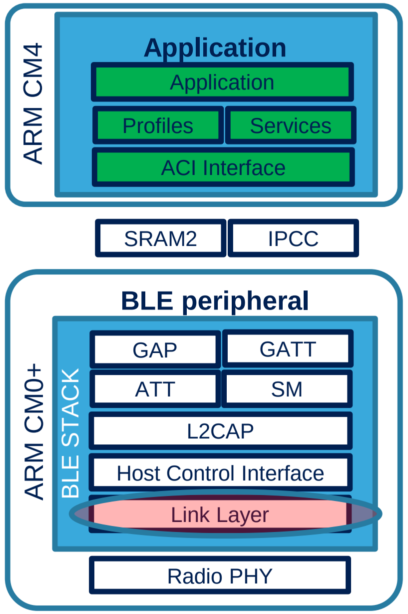

TX payload
(LSB first)
encryption
CRC generation
whitening
RF Interface
RX payload
decryption
CRC checking
dewhitening


ARM CM4
Application
Application
Profiles
Services
ACI Interface
SRAM2
IPCC
BLE peripheral
ARM CM0+
BLE STACK
GAP
GATT
ATT
SM
L2CAP
Host Control Interface
Link Layer
Radio PHY

- Link Layer (LL)
  - Radio control
  - Defines packet structure
  - One or more state machines
  - Link-layer-level encryption (via Security Manager)

- Host Control Interface (HCI)
  - Bridge between Radio Domain and M0+ Domain

- L2CAP (Logical Link Control and Adaptation Protocol)
  - Multiplex packets from higher-level protocols (ATT / SMP)
  - Handles segmentation and reassembly of packets
  - Quality of Service (QoS)

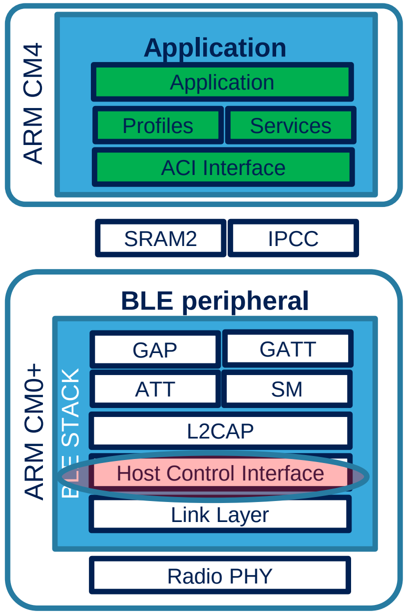


These Protocol Stack features are "Hands-Off", from an App Developer Point-of-View

- Link Layer (LL)
  - Radio control
  - Defines packet structure
  - One or more state machines
  - Link-layer-level encryption (via Security Manager)

- Host Control Interface (HCI)
  - Bridge between Radio Domain and M0+ Domain

- L2CAP (Logical Link Control and Adaptation Protocol)
  - Multiplex packets from higher-level protocols (ATT / SMP)
  - Handles segmentation and reassembly of packets
  - Quality of Service (QoS)

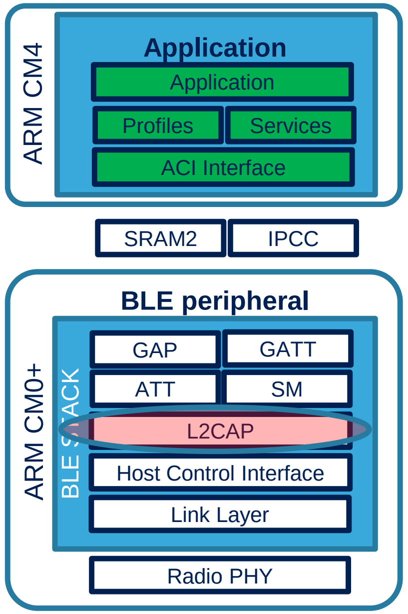


These Protocol Stack features are “Hands-Off”, from an App Developer Point-of-View

- Link Layer (LL)
    - Radio control
    - Defines packet structure
    - One or more state machines
    - Link-layer-level encryption (via Security Manager)

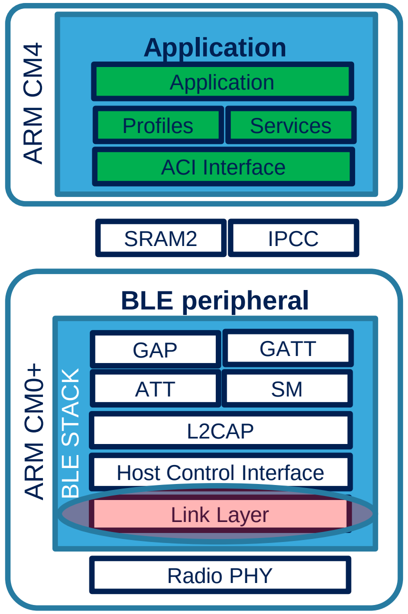
- Host Control Interface (HCI)
    - Bridge between Radio Domain and M0+ Domain

- L2CAP (Logical Link Control and Adaptation Protocol)
    - Multiplex packets from higher-level protocols (ATT / SMP)
    - Handles segmentation and reassembly of packets
    - Quality of Service (QoS)



These Protocol Stack features are "Hands-Off", from an App Developer Point-of-View

- **Standby** state: Sleep, Stop, Standby
- **Advertising** is the key to initiating all BLE communications!
- An **Initiator** and **Advertiser** negotiate a **Connection**
- In a Connection
  - The Link-Layer ***Master*** is also the GAP **Central**
  - The Link-Layer ***Slave*** is also the GAP **Peripheral**



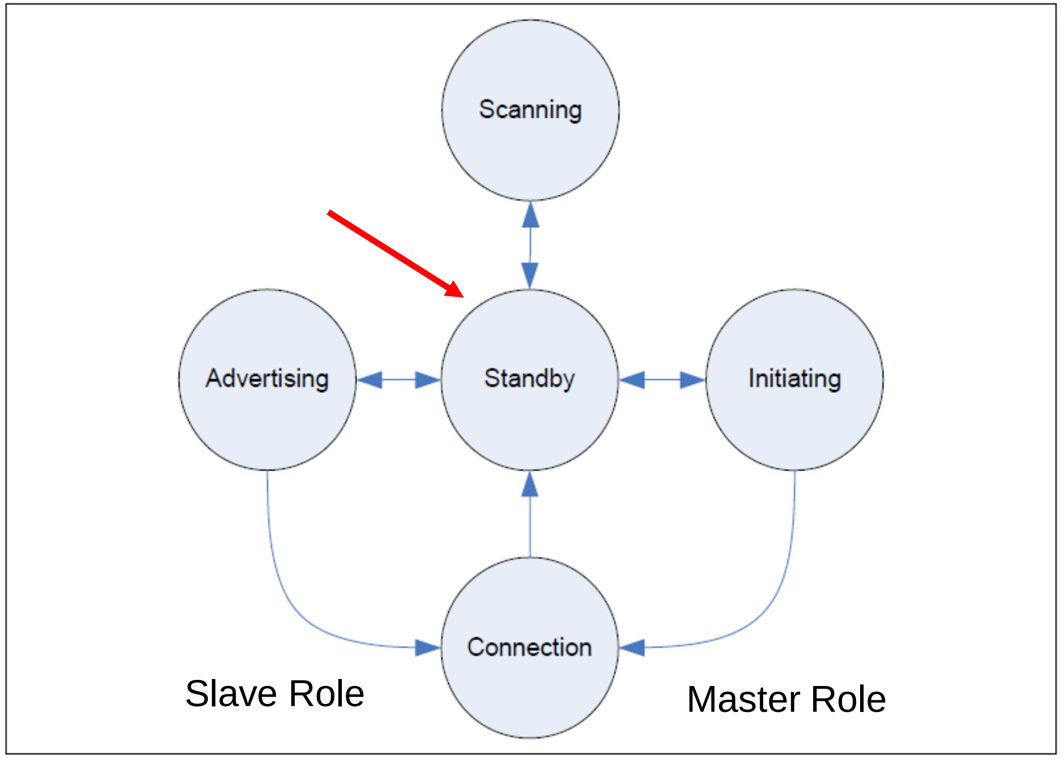
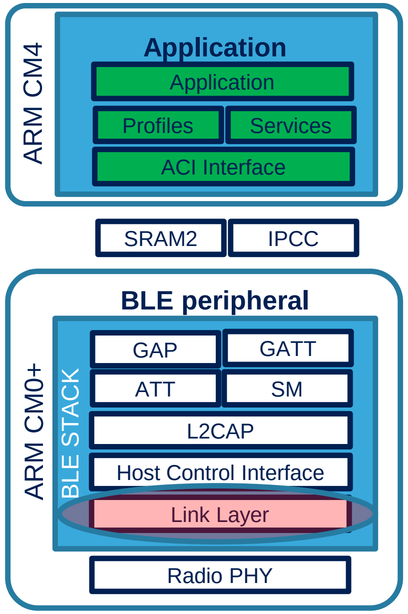


Figure 1.1: State diagram of the Link Layer state machine

Link Layer State Machine

- **Standby** state: Sleep, Stop, Standby
- **Advertising** is the key to initiating all BLE communications!
- An **Initiator** and **Advertiser** negotiate a **Connection**
- In a Connection
  - The Link-Layer ***Master*** is also the GAP **Central**
  - The Link-Layer ***Slave*** is also the GAP **Peripheral**



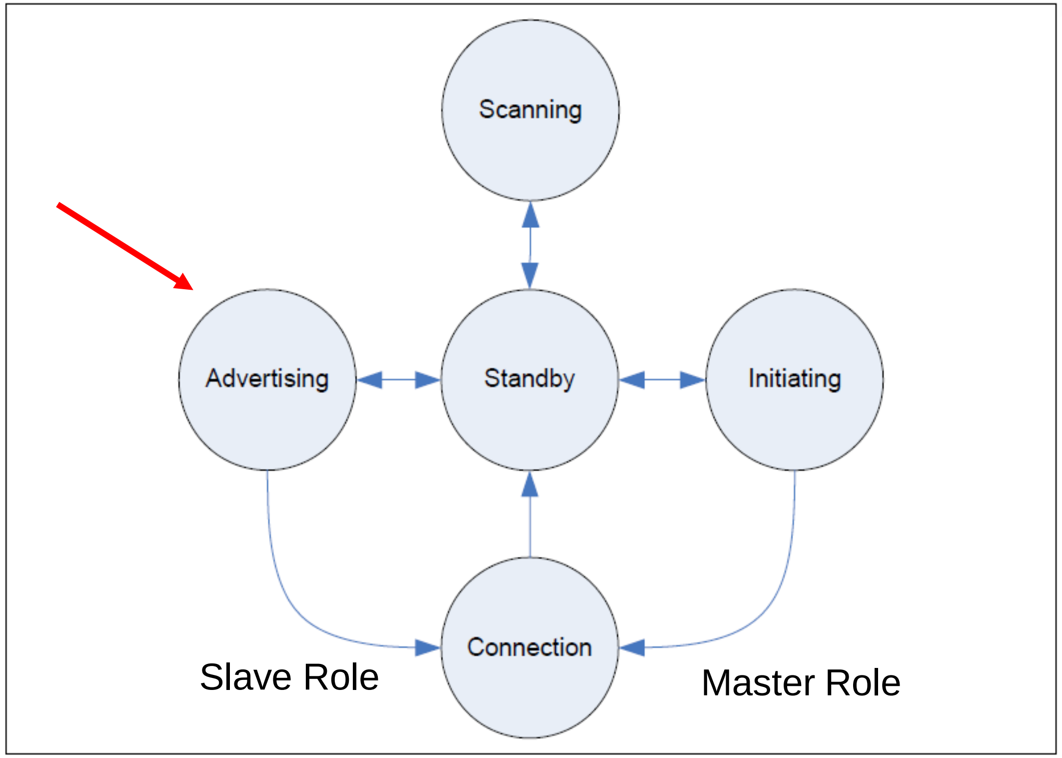
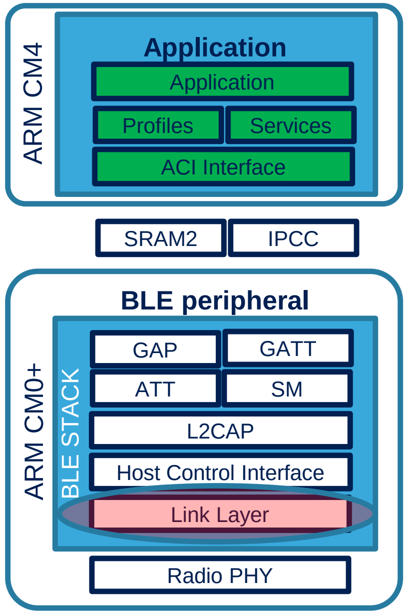


Figure 1.1: State diagram of the Link Layer state machine

life.augmented

Link Layer State Machine

- **Standby** state: Sleep, Stop, Standby
- **Advertising** is the key to initiating all BLE communications!
- An **Initiator** and **Advertiser** negotiate a **Connection**
- In a Connection
  - The Link-Layer ***Master*** is also the GAP **Central**
  - The Link-Layer ***Slave*** is also the GAP **Peripheral**



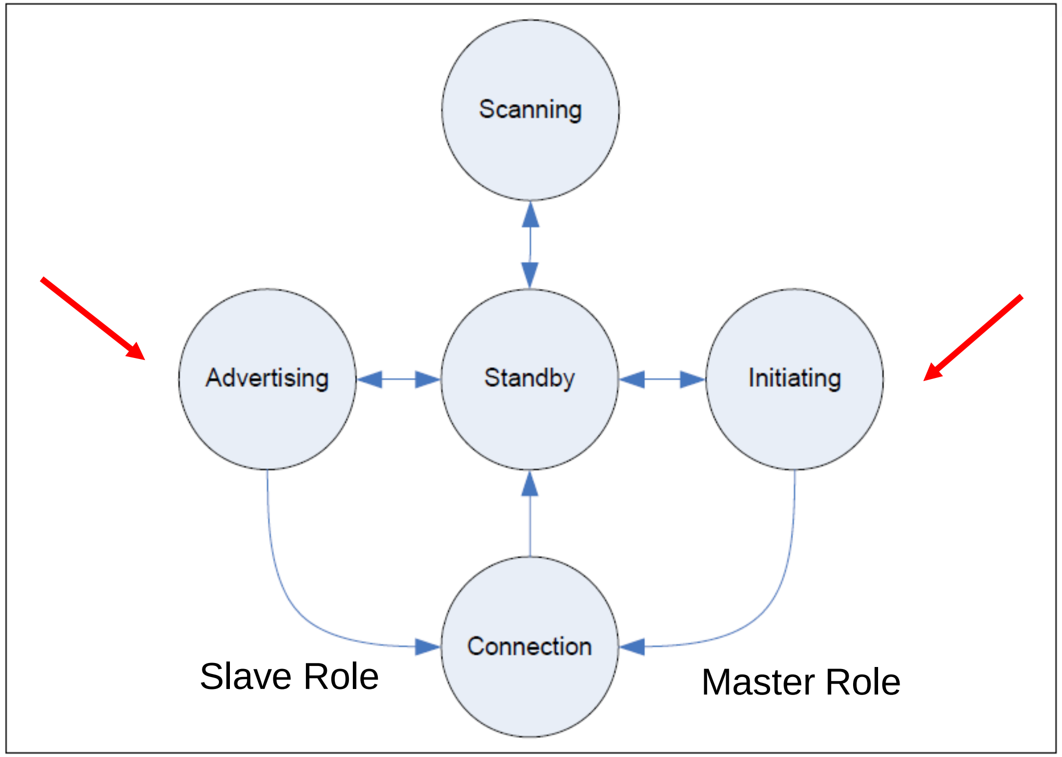


Figure 1.1: State diagram of the Link Layer state machine

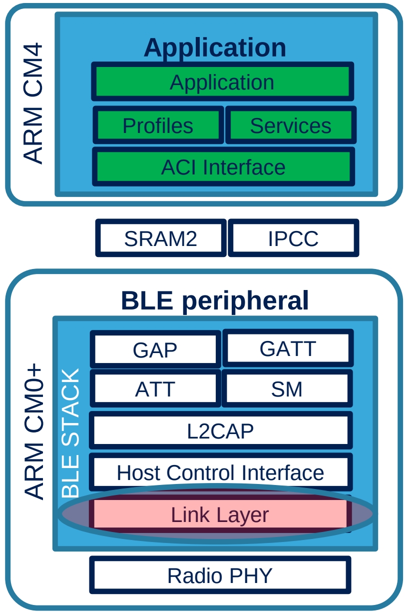
life.augmented

STM32 WB

Link Layer State Machine

Up to 8 simultaneous State Machines!

- **Standby** state: Sleep, Stop, Standby
- **Advertising** is the key to initiating all BLE communications!
- As an **Initiator** and **Advertiser** negotiate a **Connection**
- In a Connection
  - The Link-Layer ***Master*** is also the GAP **Central**
  - The Link-Layer ***Slave*** is also the GAP **Peripheral**



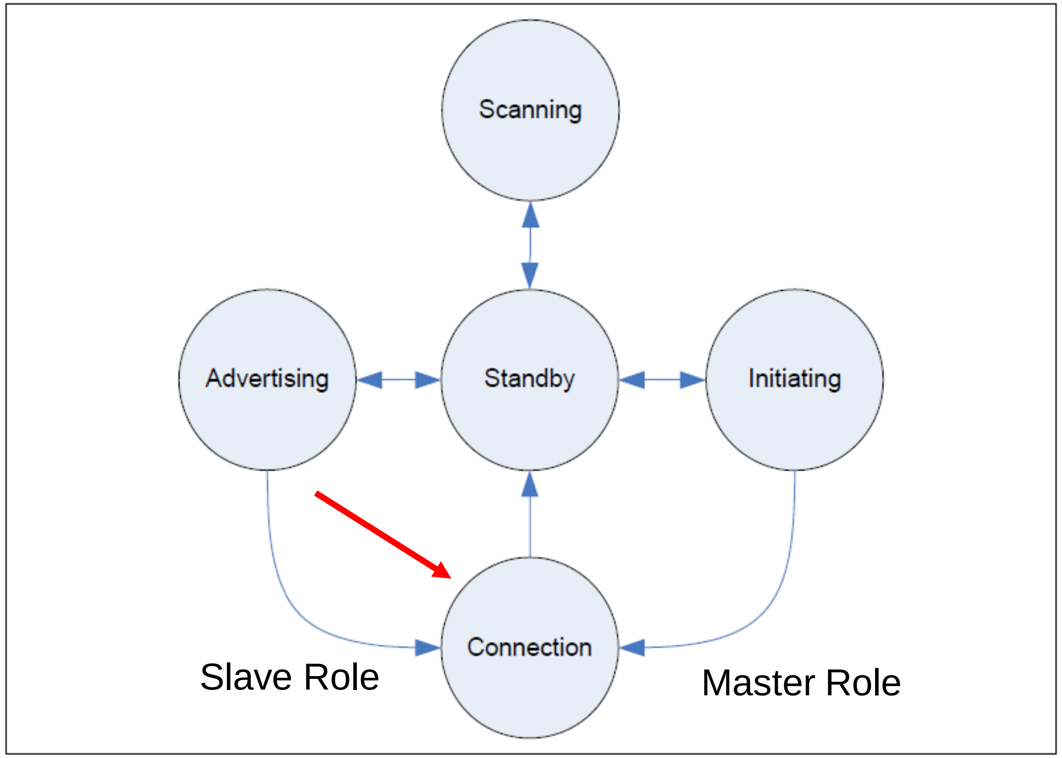
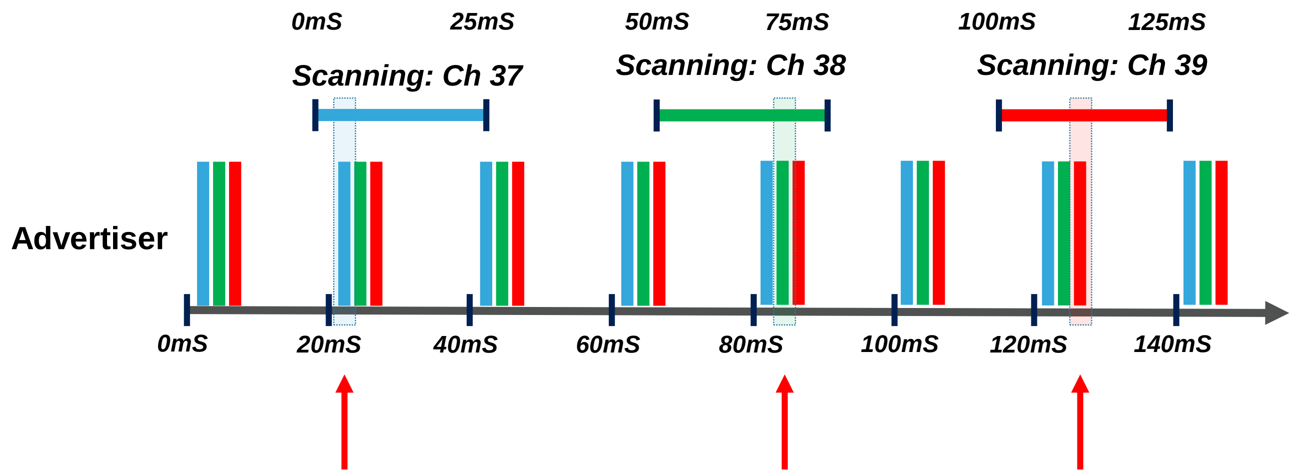


Figure 1.1: State diagram of the Link Layer state machine

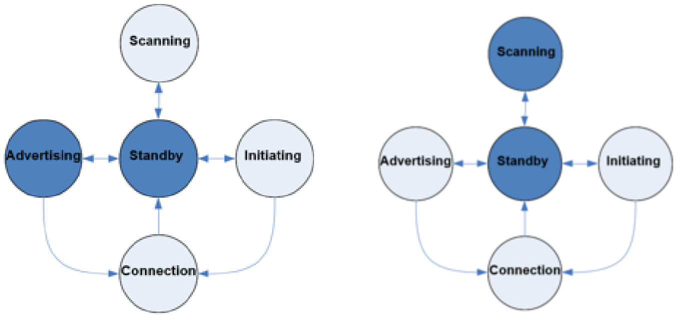
life.augmented

STM32 WB

Discovery: Advertising & Scanning



Discovery Events



Advertising on Ch 37:
Advertising on Ch 38:
Advertising on Ch 39:

Advertiser Settings:
- **Advertising Interval: 20mS**

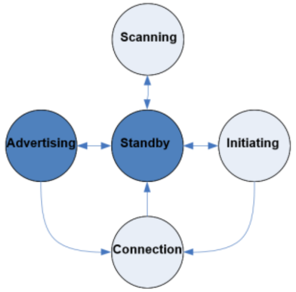
Scanner Settings:
- **Scan Interval: 50mS**
- **Scan Window: 25mS**

Roles and Modes

- Advertising Mode
- Connected Mode

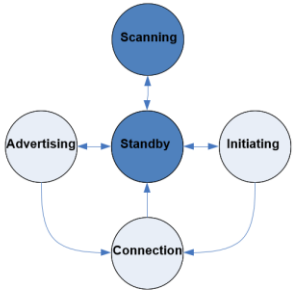
Broadcaster

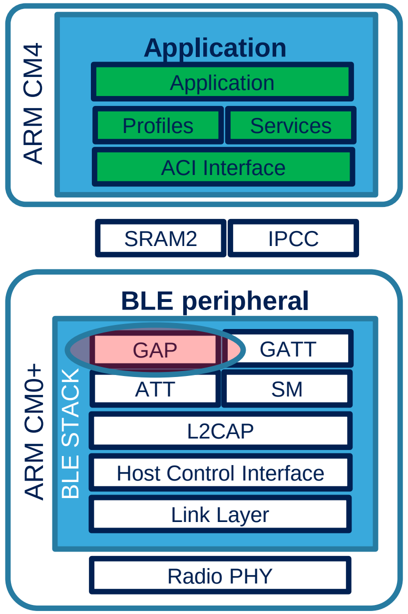
Sends advertising events
Can include characteristics and service data
Doesn't need receiver
Can be discoverable if it does have receiver



Observer

Receives advertising events
Listens for characteristics and service data
Doesn't need transmitter
Can discover devices if it does have transmitter



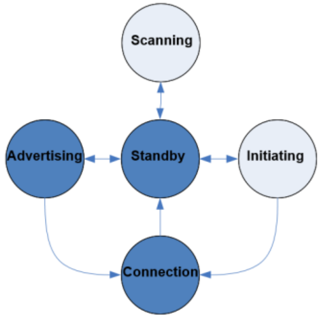


Roles and Modes

- Advertising Mode
- Connected Mode

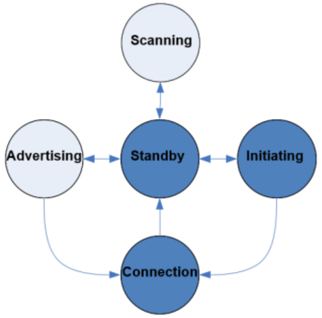
Peripheral

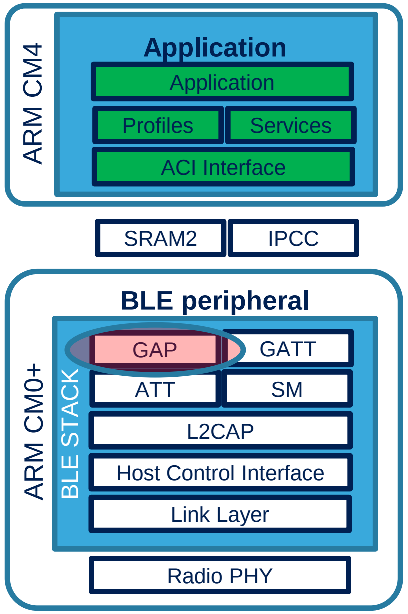
Has transmitter and receiver
Always slave
Connectable advertising



Central

Has transmitter and receiver
Always master
Never advertises



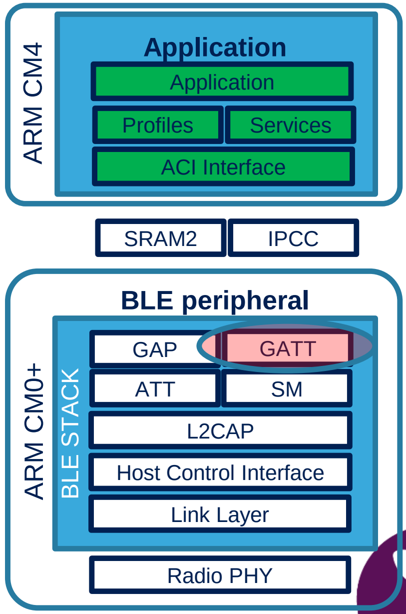


life.augmented

STM32 WB

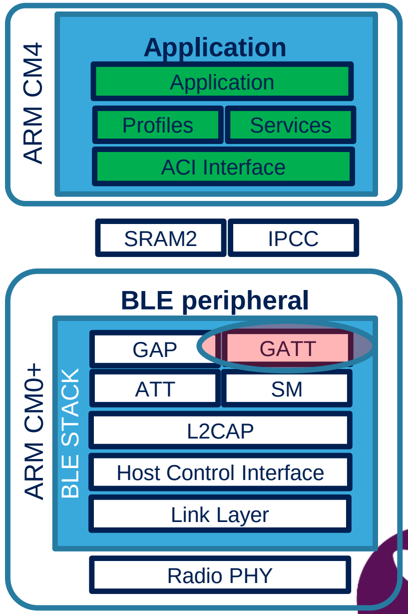
GAP Central is also a “GATT Client”

GAP Peripheral” is also a “GATT Server”



GAP Central is also a “GATT Client”

GAP Peripheral” is also a “GATT Server”



GAP Central is also a “GATT Client”
GAP Peripheral” is also a “GATT Server”

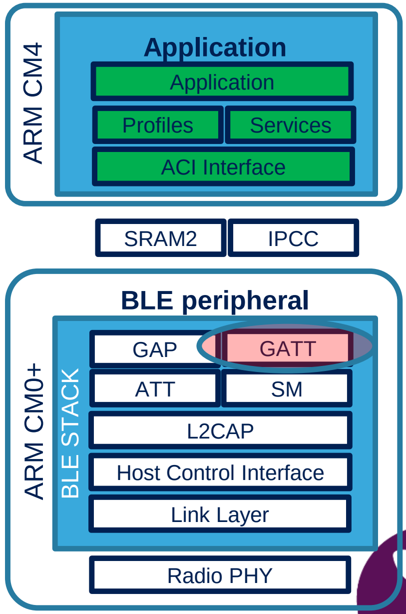
- Central queries the Services available
  - Peripheral **Services** and **Characteristics** are exposed via its’ **GATT** database

What is my heartrate?

147 bpm

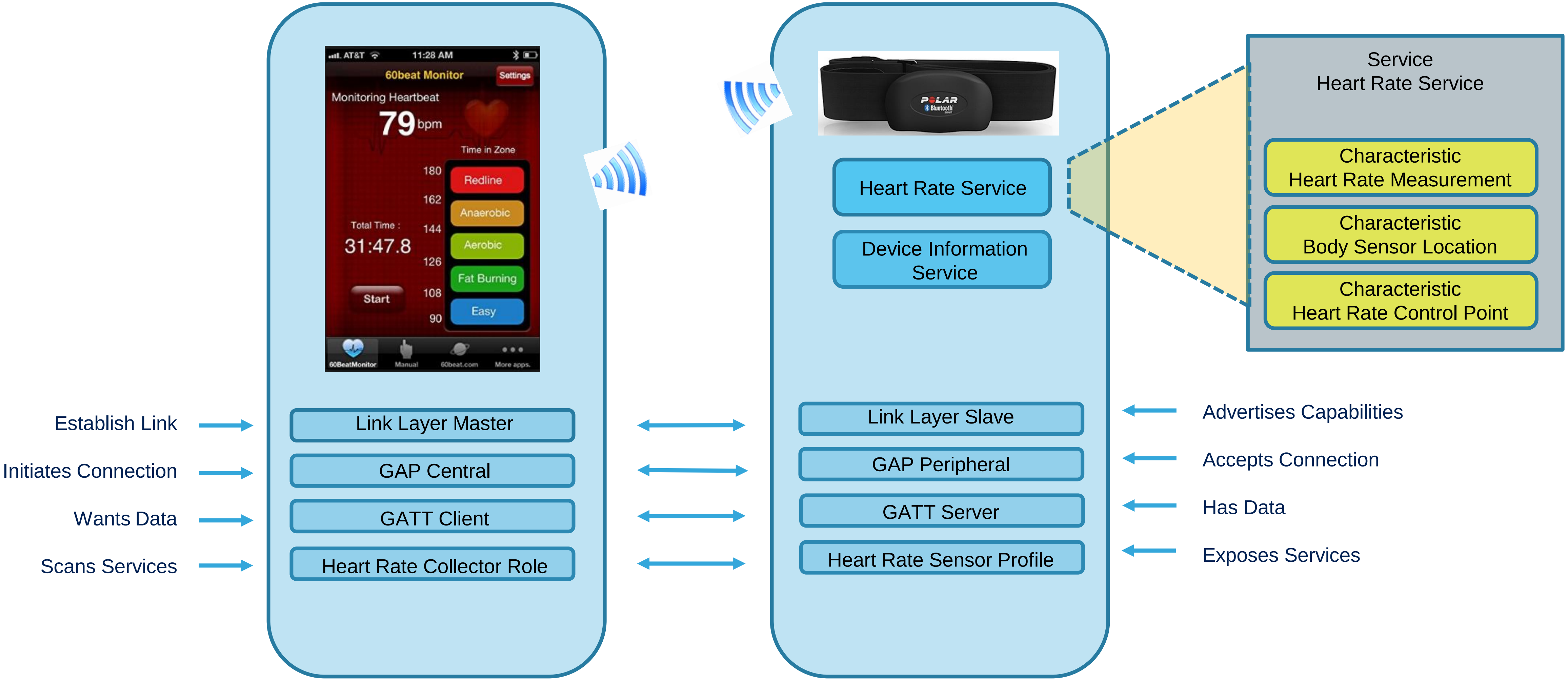
What is your Mfr ID?

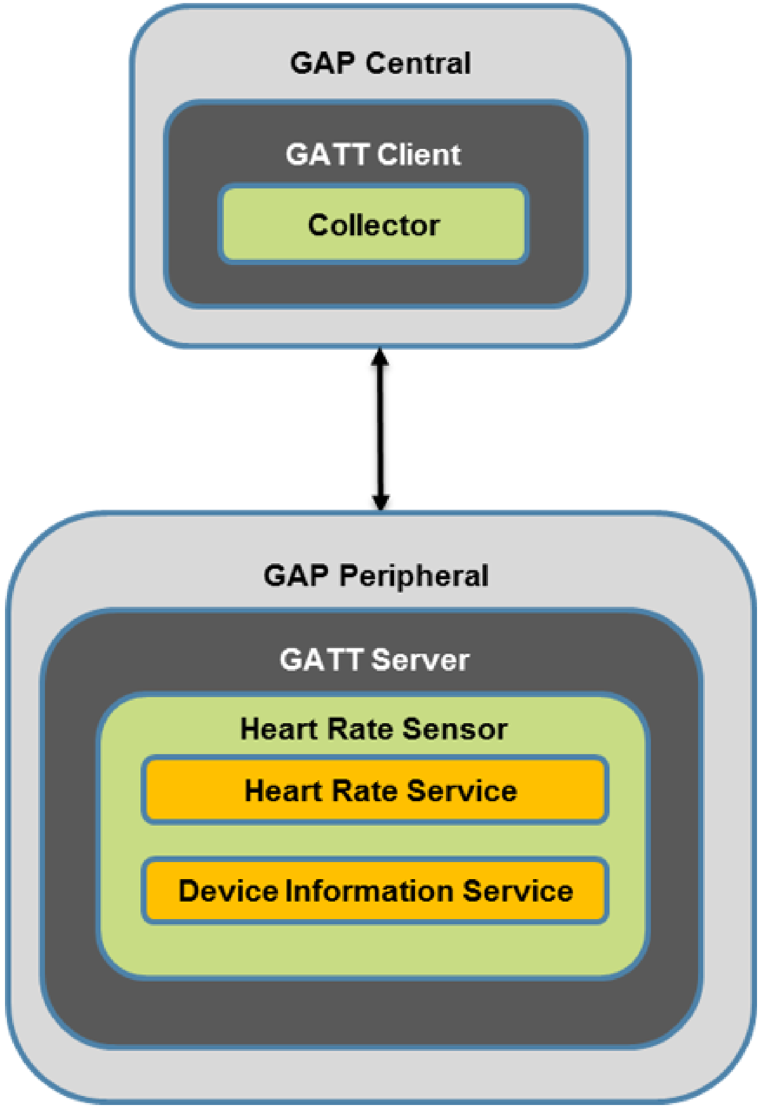
Polar

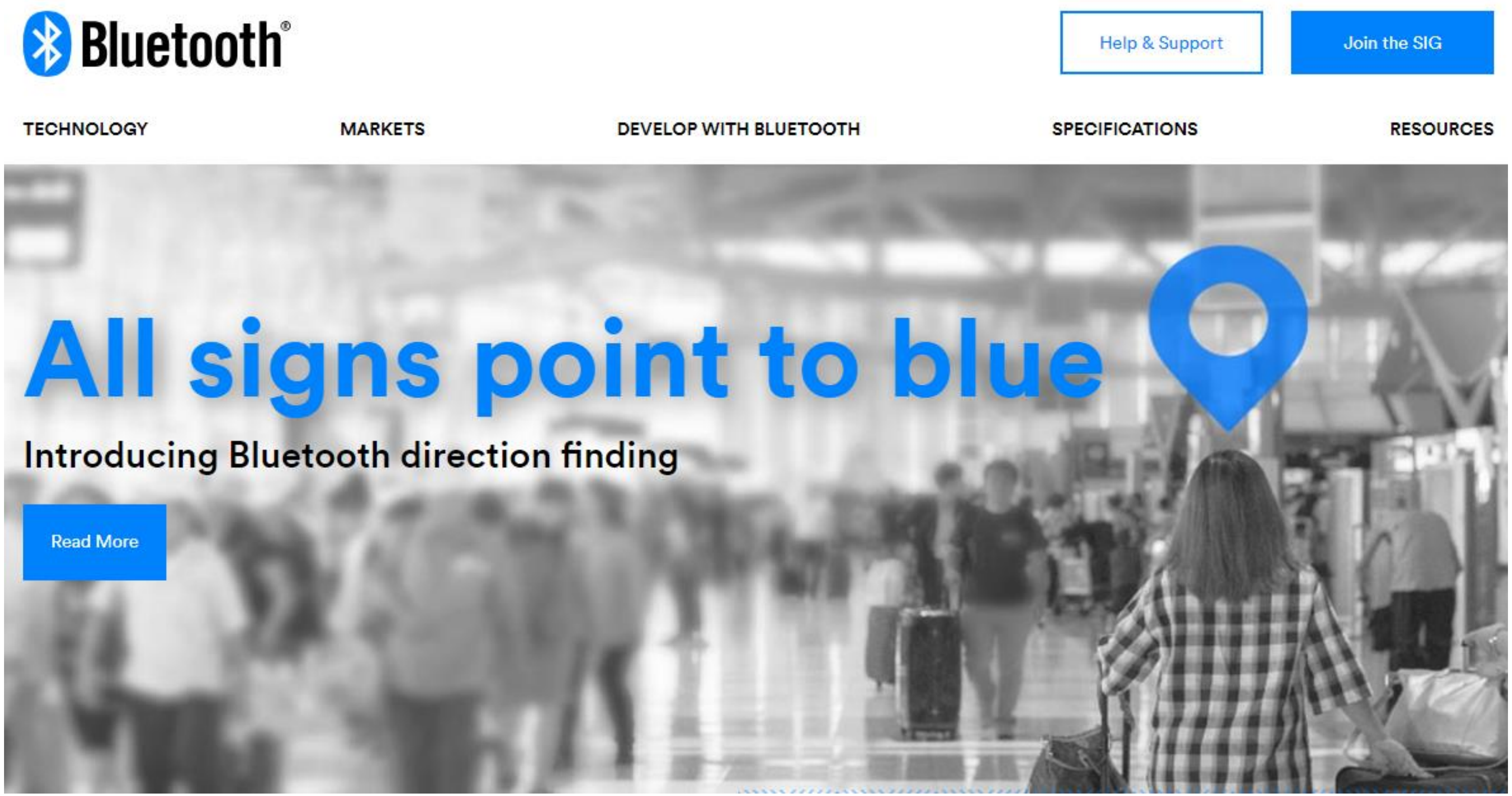


life.augmented

GATT: Profiles, Services, Characteristics & Descriptors

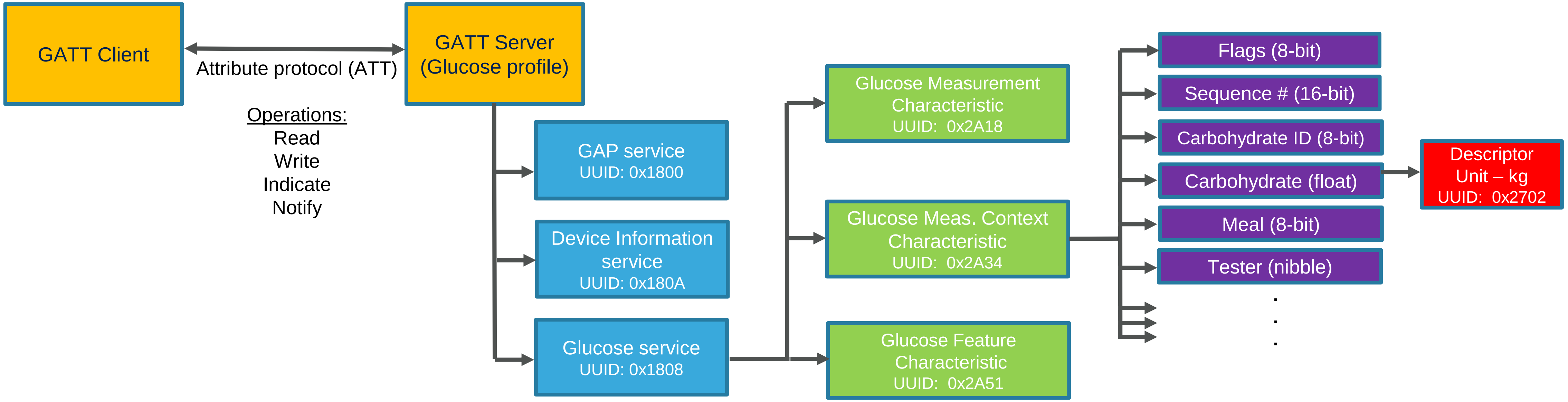


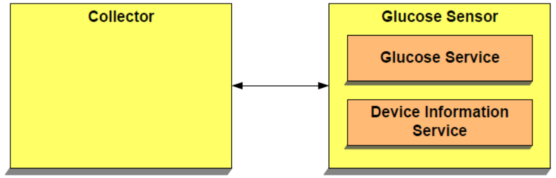

GAP Central
GATT Client
Collector
GAP Peripheral
GATT Server
Heart Rate Sensor
Heart Rate Service
Device Information Service



www.bluetooth.com

GLP Profile defines two roles: Collector & Glucose Sensor



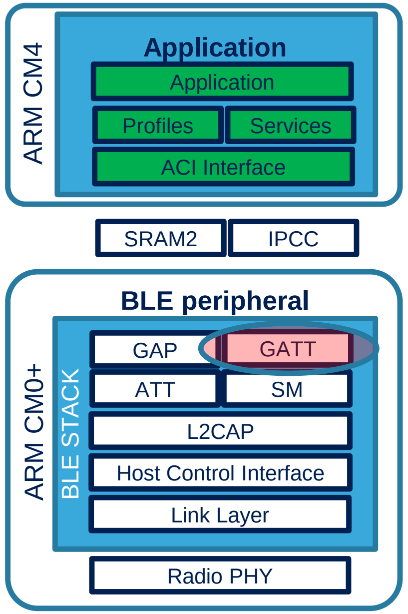


GATT Database details – Handles, UUID’s & Values

Once the GATT Server’s database information is known to the GATT Client, it can reference data via **Handles**

- ***“What is the temperature reported by the Thermometer Service?”*** ATT read command of Handle 0x0102
- ***“What are the units of temperature used?”*** ATT read command of Handle 0x0104

Handle	UUID	Description	Value
0x0100	0x2800	Thermometer service definition	UUID 0x1816
0x0101	0x2803	Characteristic: temperature	UUID 0x2A2B Value handle: 0x0102
0x0102	0x2A2B	Temperature value	20 degrees
0x0104	0x2A1F	Descriptor: unit	Celsius
0x0105	0x2902	Client characteristic configuration descriptor	0x0000
0x0110	0x2803	Characteristic: date/time	UUID 0x2A08 Value handle: 0x0111
0x0111	0x2A08	Date/Time	1/1/1980 12:00

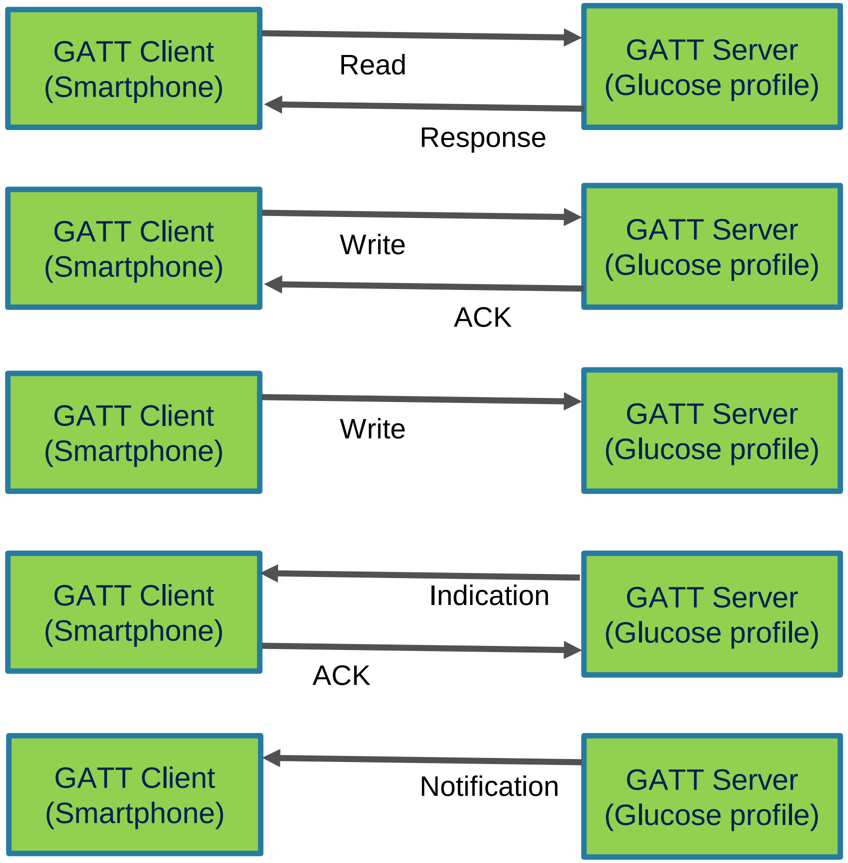


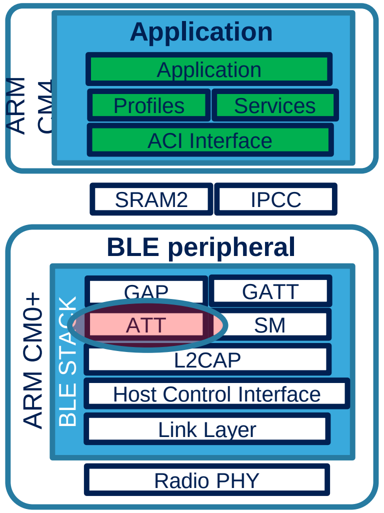
life.augmented

Attribute protocol details (ATT)

- Access GATT database information on the *Server*
- Operations
  - Read
  - Write / Write without response
  - Indicate / Notify
- Four elements
  - 16-bit **Handle**
  - **Type** of attribute (UUID)
  - **Value**
  - Attribute **Permissions** (Read-only, etc)

Handle	UUID	Description	Value
0x0100	0x2800	Thermometer service definition	UUID 0x1816
0x0101	0x2803	Characteristic: temperature	UUID 0x2A2B Value handle: 0x0102
0x0102	0x2A2B	Temperature value	20 degrees
0x0104	0x2A1F	Descriptor: unit	Celsius
0x0105	0x2902	Client characteristic configuration descriptor	0x0000
0x0110	0x2803	Characteristic: date/time	UUID 0x2A08 Value handle: 0x0111
0x0111	0x2A08	Date/Time	1/1/1980 12:00





life.augmented

- Battery Monitoring Service (BAS)
- Alert Notification Service (ANS)
- Elapsed motor use in minutes
- Unlock the drill via smartphone password
  - Add standard Services & Characteristics (16-bit UUID's from Bluetooth SIG)
  - Create custom services (128-bit UUID's)

UUID's: 16-bit or 128-bit

- **Universally Unique Identifiers (UUID's) are simply 128-bit (16-byte) numbers:**
  - **10c17863-9471-4427-8d66-82579bf9161a**
  - **Format is typically arranged as 4-2-2-2-6 and hexadecimal is assumed**
  - **To send packets more efficiently, the Bluetooth SIG has adopted a standard 112-bit UUID base:**
  - **0000XXXX-0000-1000-8000-00805F9B34FB**
  - **With a 16-bit SIG-identified service, characteristic, etc, you can use this short-form**
    - **For example, the Glucose Service in our CGM profile is:**
  - **00001808-0000-1000-8000-00805F9B34FB**
  - ***Custom* services / characteristics / descriptors need a fully defined 128-bit UUID**
    - ***Our Custom Drill needs an Unlock service.***
    - **We can generate a random UUID for it at https://www.uuidgenerator.net/**
  - ***There is also a 32-bit UUID specifier option***

UUID’s: 16-bit or 128-bit

- Universally Unique Identifiers (UUID’s) are simply 128-bit (16-byte) numbers:
  - 10c17863-9471-4427-8d66-82579bf9161a
  - Format is typically arranged as 4-2-2-2-6 and hexadecimal is assumed
  - **To send packets more efficiently, the Bluetooth SIG has adopted a standard 112-bit UUID base:**
    - **0000XXXX-0000-1000-8000-00805F9B34FB**
  - With a 16-bit SIG-identified service, characteristic, etc, you can use this short-form
    - For example, the Glucose Service in our CGM profile is:
    - 00001808-0000-1000-8000-00805F9B34FB
  - *Custom* services / characteristics / descriptors need a fully defined 128-bit UUID
    - *Our Custom Drill needs an Unlock service.*
    - We can generate a random UUID for it at https://www.uuidgenerator.net/
  - *There is also a 32-bit UUID specifier option*

life.augmented

STM32 WB

UUID's: 16-bit or 128-bit

- Universally Unique Identifiers (UUID's) are simply 128-bit (16-byte) numbers:
  - 10c17863-9471-4427-8d66-82579bf9161a
  - Format is typically arranged as 4-2-2-2-6 and hexadecimal is assumed
  - To send packets more efficiently, the Bluetooth SIG has adopted a standard 112-bit UUID base:
    - 0000XXXX-0000-1000-8000-00805F9B34FB
  - **With a 16-bit SIG-identified service, characteristic, etc, you can use this short-form**
    - **For example, the Glucose Service in our CGM profile is:**
    - **00001808-0000-1000-8000-00805F9B34FB**

Glucose service
UUID: 0x1808

  - *Custom* services / characteristics / descriptors need a fully defined 128-bit UUID
    - *Our Custom Drill needs an Unlock service.*
    - We can generate a random UUID for it at https://www.uuidgenerator.net/
  - *There is also a 32-bit UUID specifier option*

life.augmented

STM32 WB

UUID's: 16-bit or 128-bit

- Universally Unique Identifiers (UUID's) are simply 128-bit (16-byte) numbers:
  - 10c17863-9471-4427-8d66-82579bf9161a
  - Format is typically arranged as 4-2-2-2-6 and hexadecimal is assumed
  - To send packets more efficiently, the Bluetooth SIG has adopted a standard 112-bit UUID base:
  - 0000XXXX-0000-1000-8000-00805F9B34FB
  - With a 16-bit SIG-identified service, characteristic, etc, you can use this short-form
    - For example, the Glucose Service in our CGM profile is:
  - 00001808-0000-1000-8000-00805F9B34FB
  - *Custom* services / characteristics / descriptors need a fully defined 128-bit UUID
    - *Our Custom Drill needs an Unlock service.*
    - We can generate a random UUID for it at https://www.uuidgenerator.net/
  - *There is also a 32-bit UUID specifier option*

life.augmented

STM32 WB

Custom Drill Profile

- Standard Services & Characteristics (16-bit UUID's)
  - Battery service (BAS) UUID: 0x180F
    - *Battery Level Characteristic*: 0x2A19
  - Alert Notification Service UUID: 0x1811
    - *Alert Notification Control Point Characteristic:* 0x2A44
    - *Unread Alert Status Characteristic :* 0x2A45
    - *New Alert Characteristic :* 0x2A46
    - *Supported New Alert Category Characteristic:* 0x2A47
    - *Supported Unread Alert Category Characteristic :* 0x2A48
- Create custom services (128-bit UUID's)
  - **10c17863-9471-4427-8d66-82579bf9161a** (Motor run time service)
    - **5567fa77-721f-4e1a-9875-7ae95ead642d** xxx Characteristic
    - **3d78d6f3-7d34-4f89-a14d-ed3cac297438** xxx Characteristic
  - **0226b0db-d9a6-49c8-bce1-fccd3a40e6e2** (Unlock service)
    - **997e28a5-f05e-4027-89c7-e84ce4ce67ec** xxx Characteristic
    - **b3b7d2a1-4eeb-4a39-85ef-7ddd7b1e4abf** xxx Characteristic

Name: Battery Service

Type: org.bluetooth.service.battery_service

Assigned Number: 0x180F

Name: Battery Level

Type: org.bluetooth.characteristic.battery_level

Assigned Number: 0x2A19

Name: Alert Notification Service

Type: org.bluetooth.service.alert_notification Download / View

Assigned Number: 0x1811

Name: Alert Notification Control Point

Type: org.bluetooth.characteristic.alert_notification_control_point

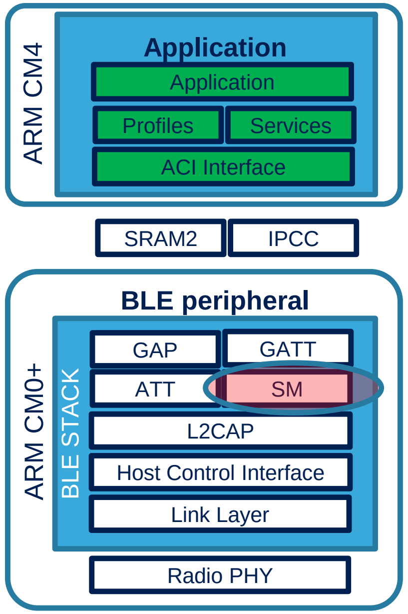
Assigned Number: 0x2A44

Name: Supported New Alert Category

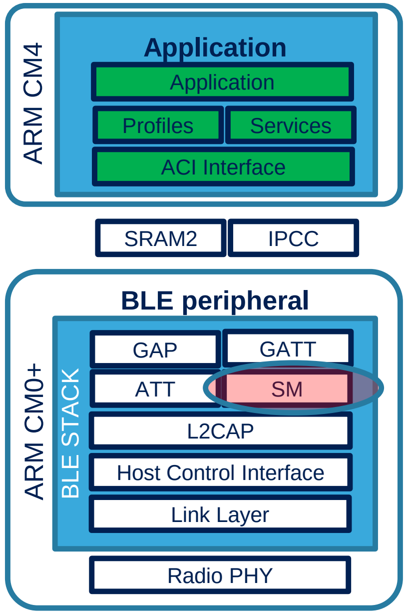
Type: org.bluetooth.characteristic.supported_new_alert_category

Assigned Number: 0x2A47

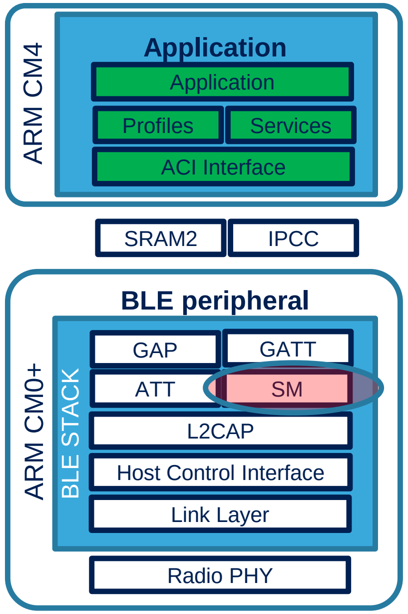
- **Connection**: GAP Central connected to a GAP Peripheral (Connection interval = 7.5ms to 4 secs)
- **Pairing**: Connected devices exchange encryption keys to **encrypt** the link. There are now **paired**.
- **Bonding**: Paired devices can be bonded – Keys are stored for the next connection.
- **Whitelisting**: Restrict connections from any other than known devices.



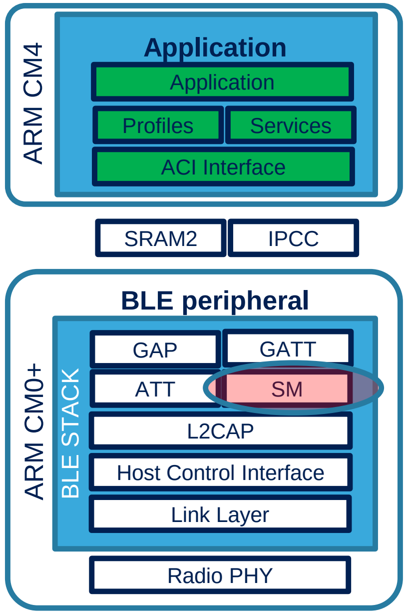
- **Connection**: GAP Central connected to a GAP Peripheral (Connection interval = 7.5ms to 4 secs)
- **Pairing**: Connected devices exchange encryption keys to **encrypt** the link. There are now **paired**.
- **Bonding**: Paired devices can be bonded – Keys are stored for the next connection.
- **Whitelisting**: Restrict connections from any other than known devices.



- **Connection**: GAP Central connected to a GAP Peripheral (Connection interval = 7.5ms to 4 secs)
- **Pairing**: Connected devices exchange encryption keys to **encrypt** the link. There are now **paired**.
- **Bonding**: Paired devices can be bonded – Keys are stored for the next connection.
- **Whitelisting**: Restrict connections from any other than known devices.



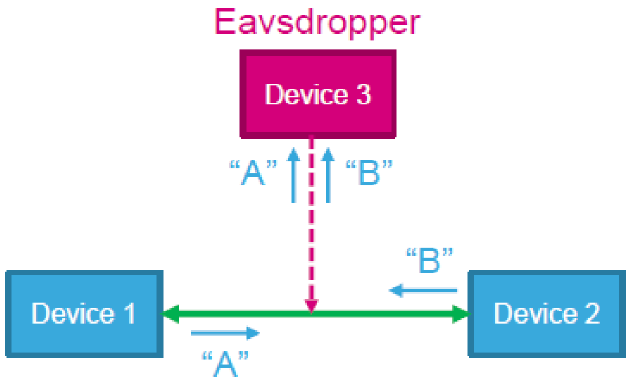
- **Connection**: GAP Central connected to a GAP Peripheral (Connection interval = 7.5ms to 4 secs)
- **Pairing**: Connected devices exchange encryption keys to **encrypt** the link. There are now **paired**.
- **Bonding**: Paired devices can be bonded – Keys are stored for the next connection.
- **Whitelisting**: Restrict connections from any other than known devices.

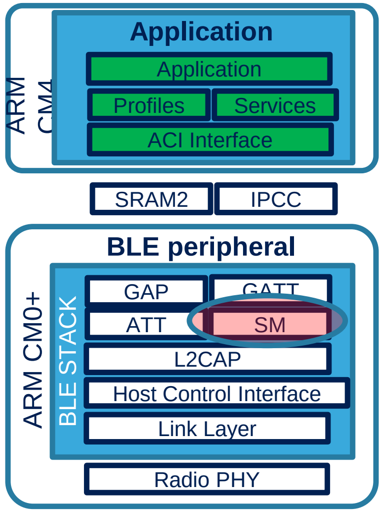


- Security modes are deployed after a BLE connection is established
- BLE Link Layer uses AES-128 CCM mode for authenticated encryption

Typical attacks

- **Passive eavesdropping:**
  - A third device listens in to the data being exchanged between the two paired devices
  - Overcome by AES-CCM encryption
- MITM
  - A malicious device impersonates the other two legitimate devices
- Identity tracking
  - Malicious entity associates BLE device address to physically track the user
  - BLE overcomes this is by periodically changing the device address.



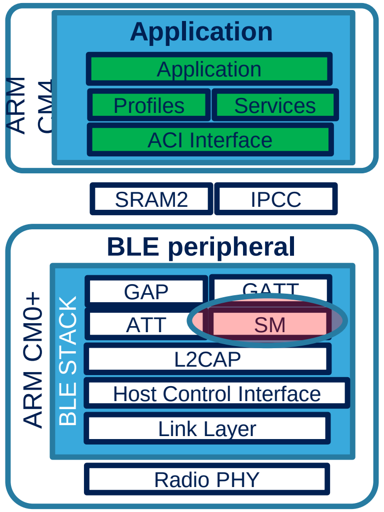


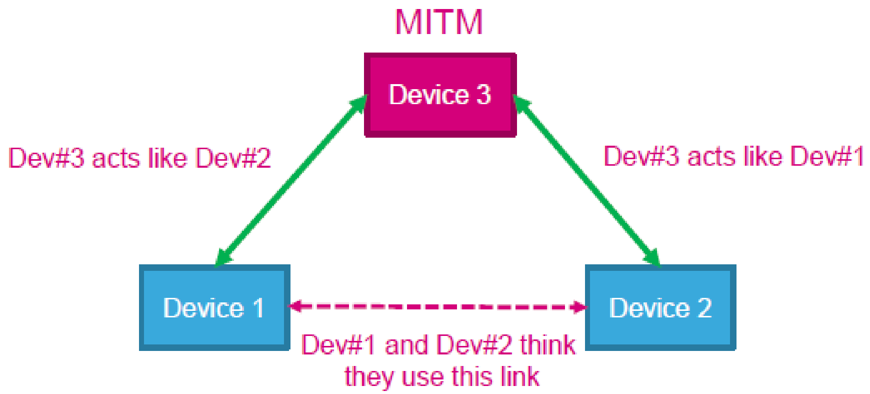
life.augmented

STM32 WB

Typical attacks

- **Passive eavesdropping:**
  - A third device listens in to the data being exchanged between the two paired devices
  - Overcome by AES-CCM encryption
- **MITM**
  - A malicious device impersonates the other two legitimate devices
- **Identity tracking**
  - Malicious entity associates BLE device address to physically track the user
  - BLE overcomes this is by periodically changing the device address.



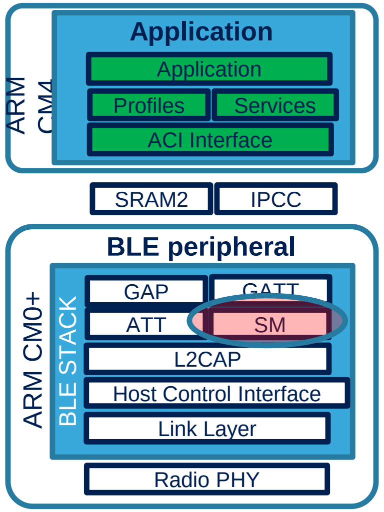


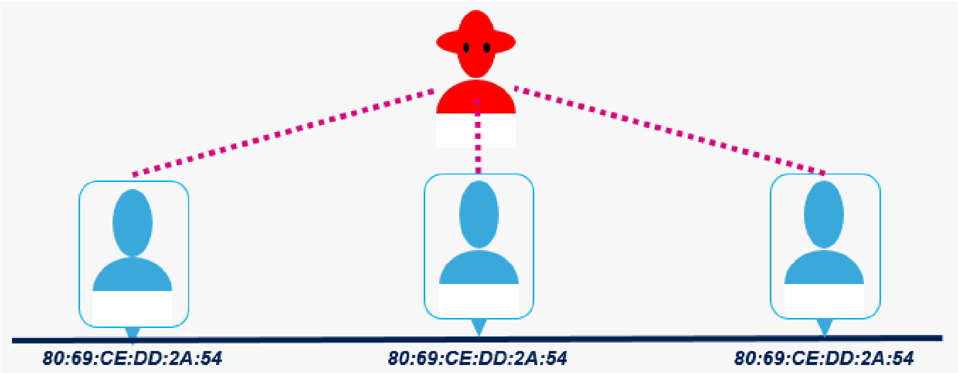
Typical attacks

- **Passive eavesdropping:**
  - A third device listens in to the data being exchanged between the two paired devices
  - Overcome by AES-CCM encryption

- **MITM**
  - A malicious device impersonates the other two legitimate devices

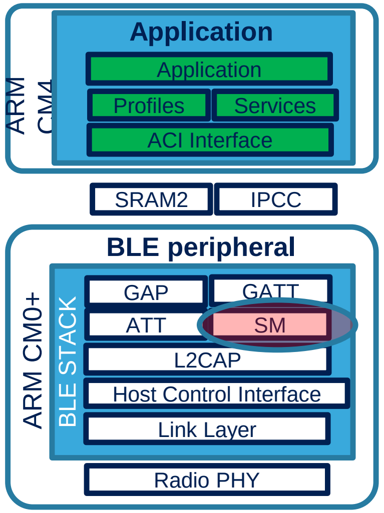
- **Identity tracking**
  - Malicious entity associates BLE device address to physically track the user
  - BLE overcomes this is by periodically changing the device address.





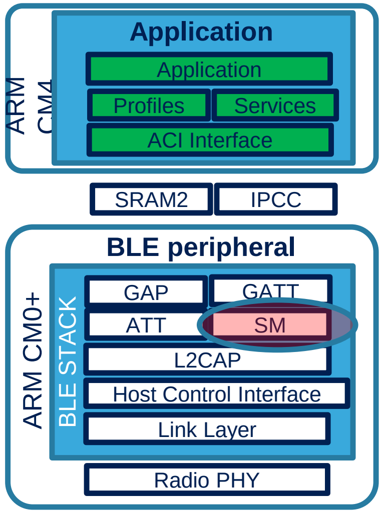
Pairing Methods

- **Just Works^{TM}:**
  - Still vulnerable to MITM attack
- **Out of Band (OOB) Pairing:**
  - Keys exchanged over a different wireless technology such as NFC
- **Passkey:**
  - 6-digit number entered on each device
  - Assumes keypad capability
- **Numeric Comparison:**
  - Similar to Just Works^{TM}, but adds a 6-digit confirmation value
  - Additional protection from MITM attacks



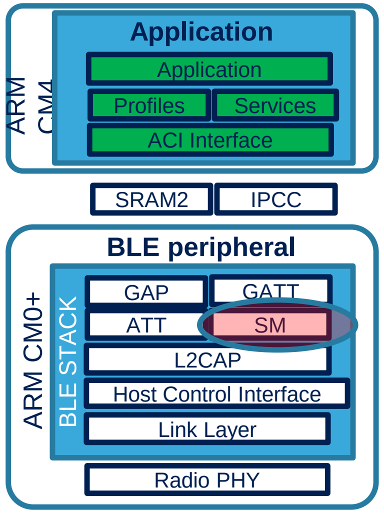
Pairing Methods

- **Just Works™:**
  - Still vulnerable to MITM attack
- **Out of Band (OOB) Pairing:**
  - Keys exchanged over a different wireless technology such as NFC
- **Passkey:**
  - 6-digit number entered on each device
  - Assumes keypad capability
- **Numeric Comparison:**
  - Similar to Just Works™, but adds a 6-digit confirmation value
  - Additional protection from MITM attacks



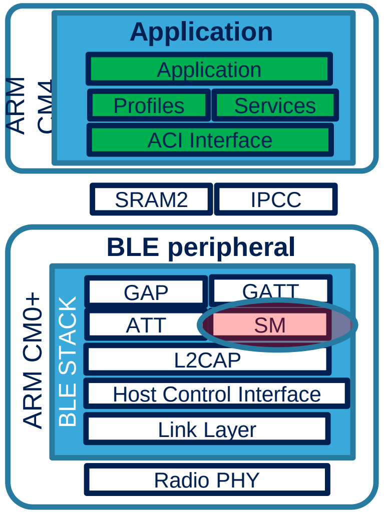
Pairing Methods

- **Just Works^{TM}:**
  - Still vulnerable to MITM attack
- **Out of Band (OOB) Pairing:**
  - Keys exchanged over a different wireless technology such as NFC
- **Passkey:**
  - 6-digit number entered on each device
  - Assumes keypad capability
- **Numeric Comparison:**
  - Similar to Just Works^{TM}, but adds a 6-digit confirmation value
  - Additional protection from MITM attacks



Pairing Methods

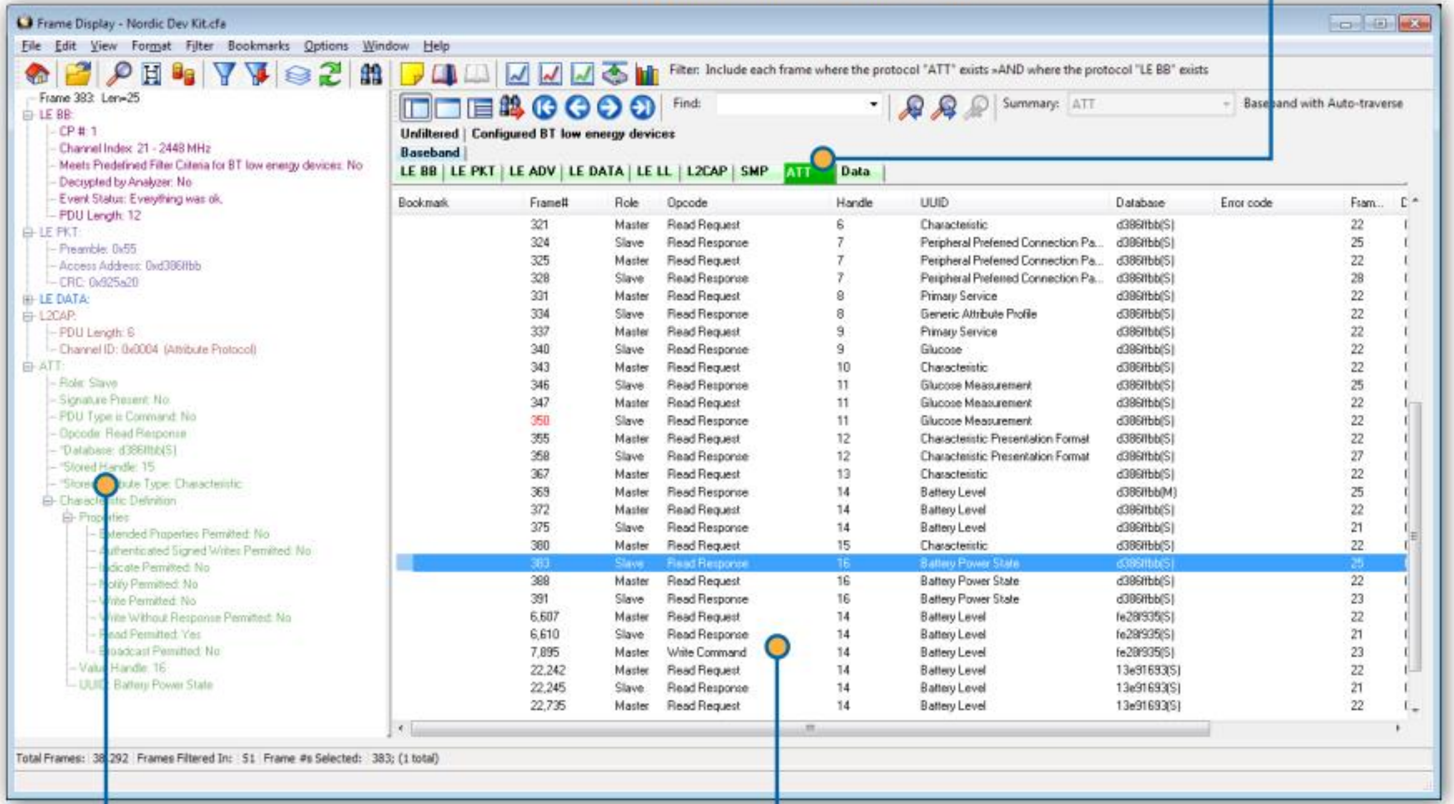
- **Just Works™:**
  - Still vulnerable to MITM attack
- **Out of Band (OOB) Pairing:**
  - Keys exchanged over a different wireless technology such as NFC
- **Passkey:**
  - 6-digit number entered on each device
  - Assumes keypad capability
- **Numeric Comparison:**
  - Similar to Just Works™, but adds a 6-digit confirmation value
  - Additional protection from MITM attacks



A “White List” can **optionally** be used to filter device addresses

- **Advertising State** - An advertiser *shall process connection requests only from devices in the White List*
- **Scanning State** – *A scanner shall process advertising packets only from White-Listed devices*
- **Initiating State** - *An initiator shall process connectable advertising packets only from White-Listed devices*

Filter Tabs isolate frames by profile or protocol for quick and convenient viewing of specific kinds of data.



Decode Pane shows comprehensive layered decoders of each frame/message with clear, concise descriptions.

Summary Pane displays a one line overview of each data frame/message. Click on any line to reveal detail in mutiple panes below.

Connection request

Unfiltered | Configured BT low energy devices | Errors |
LE BB | LE PKT | LE ADV |

Advertiser Address	Bo...	Frame#	Chan	Type	AddrTypel	InitA/ScanA	Add...	AdvA	Len	Frame
0x59d77766be00		323	37	ADV_IND			(rand)	0x59d77766be00	17	35
0x59d77766be00		324	38	ADV_IND			(rand)	0x59d77766be00	17	35
0x59d77766be00		325	39	ADV_IND			(rand)	0x59d77766be00	17	35
0x59d77766be00		326	37	ADV_IND			(rand)	0x59d77766be00	17	35
0x59d77766be00		327	39	ADV_IND			(rand)	0x59d77766be00	17	35
0x0080fae1008a		328	39	ADV_IND			(pub)	0x0080fae1008a	33	51
0x0080fae1008a		329	37	ADV_IND			(pub)	0x0080fae1008a	33	51
0x0080fae1008a		330	37	CONNECT_IND	(rand)	0x59d77766be00	(pub)	0x0080fae1008a	34	52
0x59d77766be00		331	37	ADV_IND			(rand)	0x59d77766be00	17	35

- LE BB:
  - Header Length: 13
  - Header Version: 3
  - CP #: 1
  - RF Channel: 0 - 2402 MHz
  - Channel Index: 37
  - Meets Predefined Filter Criteria for BT low energy devices: No
  - Receive Status: Received without errors
  - Decryption Initiated: No
  - Signal Strength: 8 (medium)
  - PDU Length: 36
  - Preamble: 0xaa
  - Access Address: 0x8e89bed6
- LE PKT:
  - Preamble: 0xaa
  - Access Address: 0x8e89bed6
  - CRC: 0x8f2231
- LE ADV:
  - Channel Selection #2: Not Supported
  - PDU Type: CONNECT_IND
  - Advertiser Address Type: public
  - Initiator Address Type: random
  - Payload Length: 34
  - Initiator Address: 0x59d77766be00
    - Init Type: Resolvable
  - Advertiser Address: 0x0080fae1008a
  - Access Address: 0xaf9a9b2e
  - CRC initialization value: 0xee7a11
  - transmitWindowSize: 3.75 ms
  - transmitWindowOffset: 1.25 ms
  - connInterval: 30.00 ms
  - connSlaveLatency: 0
  - connSupervisionTimeout: 720.00 ms
  - Channel Map: 0001111111111111111111111111111111111111
  - masterSCA: 31 ppm to 50 ppm
  - hopIncrement: 7

CONNECT_IND PDU

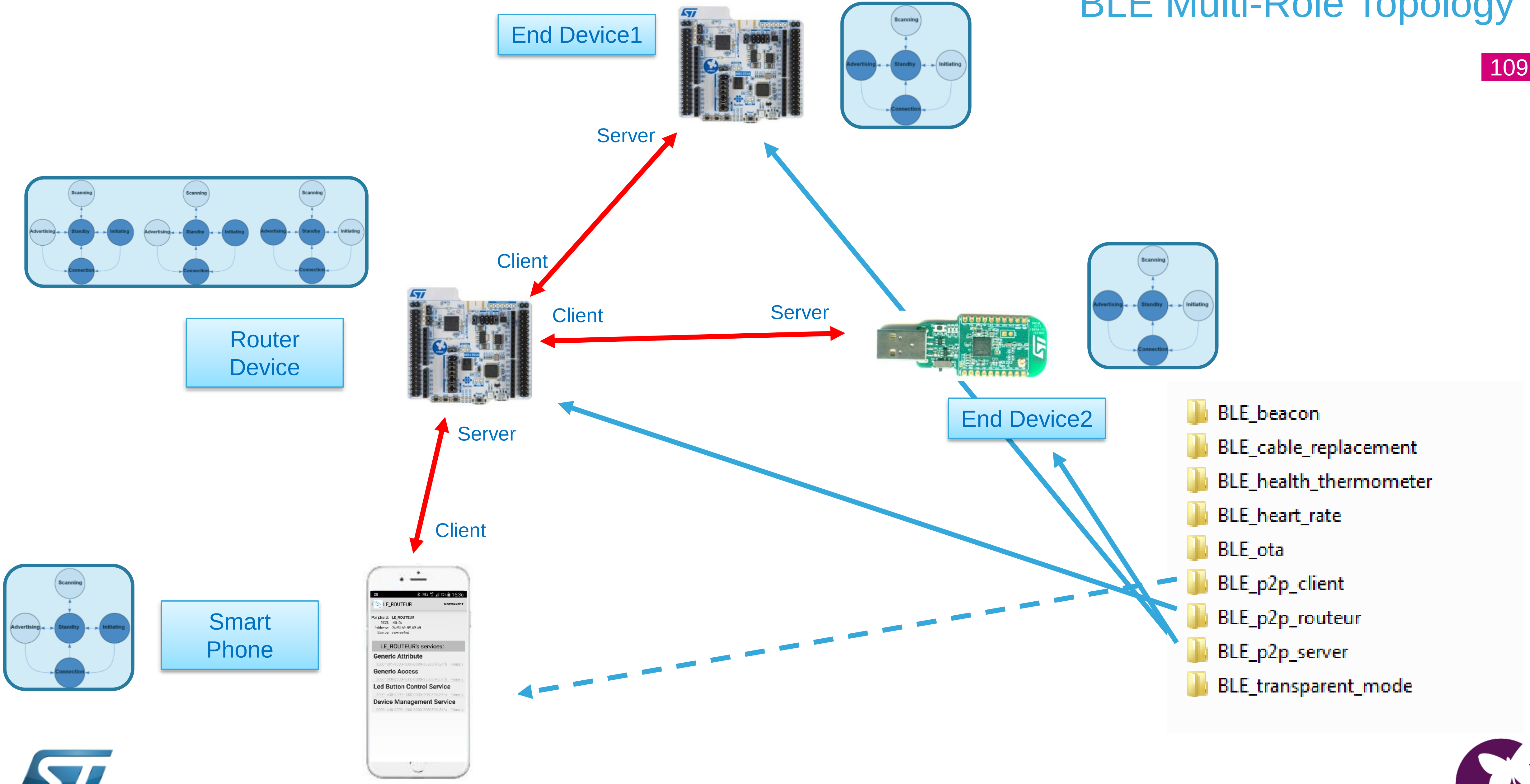
Preamble	Access Addr		Length 0x22=36 bytes	Initiator Addr	Advertiser Addr	Access Addr to use
aa	d6 be 89 8e	45	22	00 be 66 77 d7 59	8a 00 e1 fa 80 00	2e 9b 9a af

CRC Init	Tx WindowSize 0x03*1.25 = 3.75mS	Tx Offset 0x0001*1.25 = 1.25mS	Conn Interval 0x0018*1.25 = 30mS	Slave Latency 0x0000 = No ignored events	Conn Timeout 0x0048*10 = 900mS	Channel Map All 37 data channels enabled	Hop / Sleep Clk Accuracy	CRC
11 7a ee	03	01 00	18 00	00 00	48 00	ff ff ff ff 1f	a7	8f 22 31

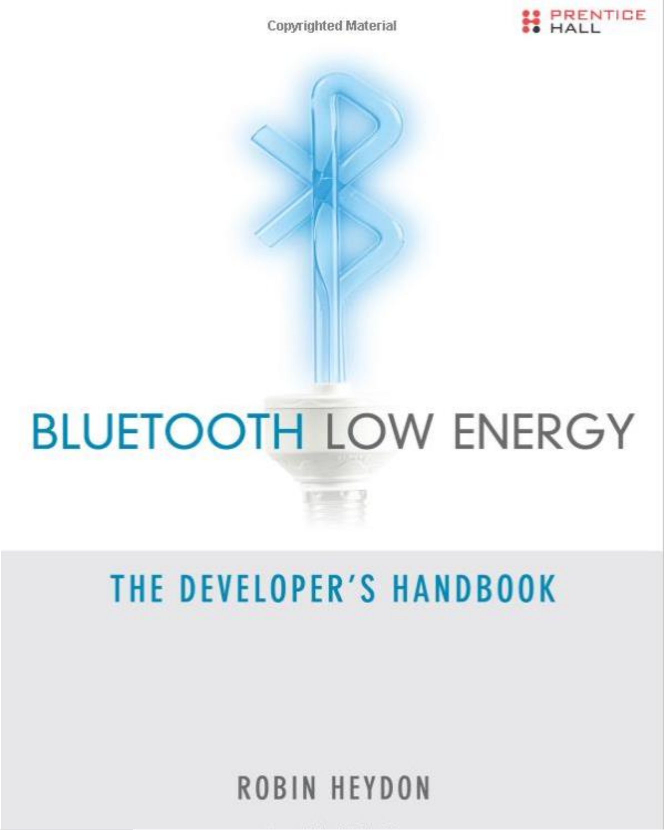
life.augmented

STM32 WB

BLE Multi-Role Topology

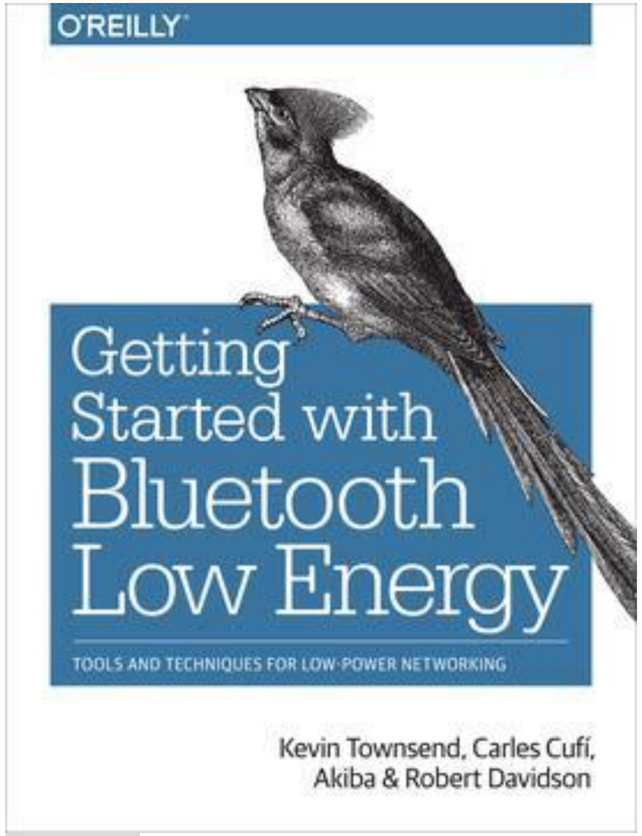


BLE4.0 only but highly detailed



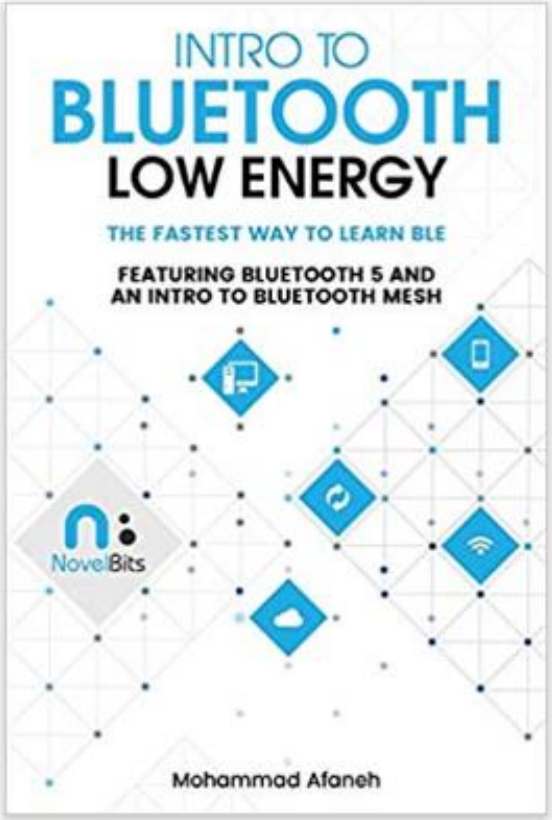
2012

Up to 4.1, 5.0 not covered



2014

Includes 5.0 & Mesh



2018

Bluetooth®

https://www.bluetooth.com/

Core Specifications

The *Bluetooth®* Core Specification defines the technology building blocks that developers use to create the interoperable devices that make up the thriving Bluetooth ecosystem. The Bluetooth specification is overseen by the Bluetooth Special Interest Group (SIG) and is regularly updated and enhanced by Bluetooth SIG Working Groups to meet evolving technology and market needs.

Specification		Version	Status	Adoption Date
CS	Core Specification	5.0	Active	06 Dec 2016
CSS	Core Specification Supplement	7	Active	06 Dec 2016
CSA	Core Specification Addendum	6	Active	12 Jul 2017

Bluetooth®

https://www.bluetooth.com/specifications/gatt

- Working Groups
- Core Specifications
- Mesh Networking Specifications
- Traditional Profile Specifications
- Protocol Specifications
- **GATT Specifications**
  - GATT Overview
  - GATT Characteristics
  - GATT Declarations
  - GATT Descriptors
  - GATT Services
  - Mesh GATT Services XML
  - Available Schemas
- Errata Service Releases
- Qualification Test Requirements
- Assigned Numbers

GATT Specifications

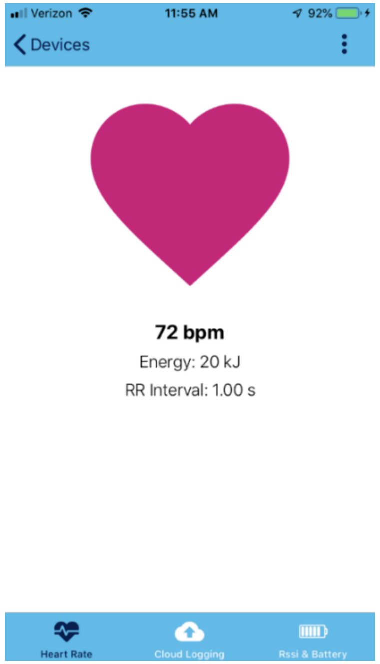
Generic Attributes (GATT) services are collections of characteristics and relationships to other services that encapsulate the behavior of part of a device.

A *GATT profile* describes a use case, roles, and general behaviors based on the GATT functionality, enabling extensive innovation while maintaining full interoperability with other *Bluetooth*® devices.

The documents in the “Informative document showing changes” column are provided as a courtesy to help readers identify changes between two versions of a Bluetooth specification. When implementing specifications, use the adopted versions in the “Adopted Version” column.

More about GATT

Profile Specification		Version	Status	Adoption Date	Informative document showing changes
ANP	Alert Notification Profile	1.0	Active	13 Sep 2011	N/A
ANS	Alert Notification Service	1.0	Active	13 Sep 2011	N/A
AIOP	Automation IO Profile	1.0	Active	14 Jul 2015	N/A
AIOS	Automation IO Service	1.0	Active	14 Jul 2015	N/A
BAS	Battery Service	1.0	Active	27 Dec 2011	N/A
BCS	Body Composition Service	1.0	Active	21 Oct 2014	N/A

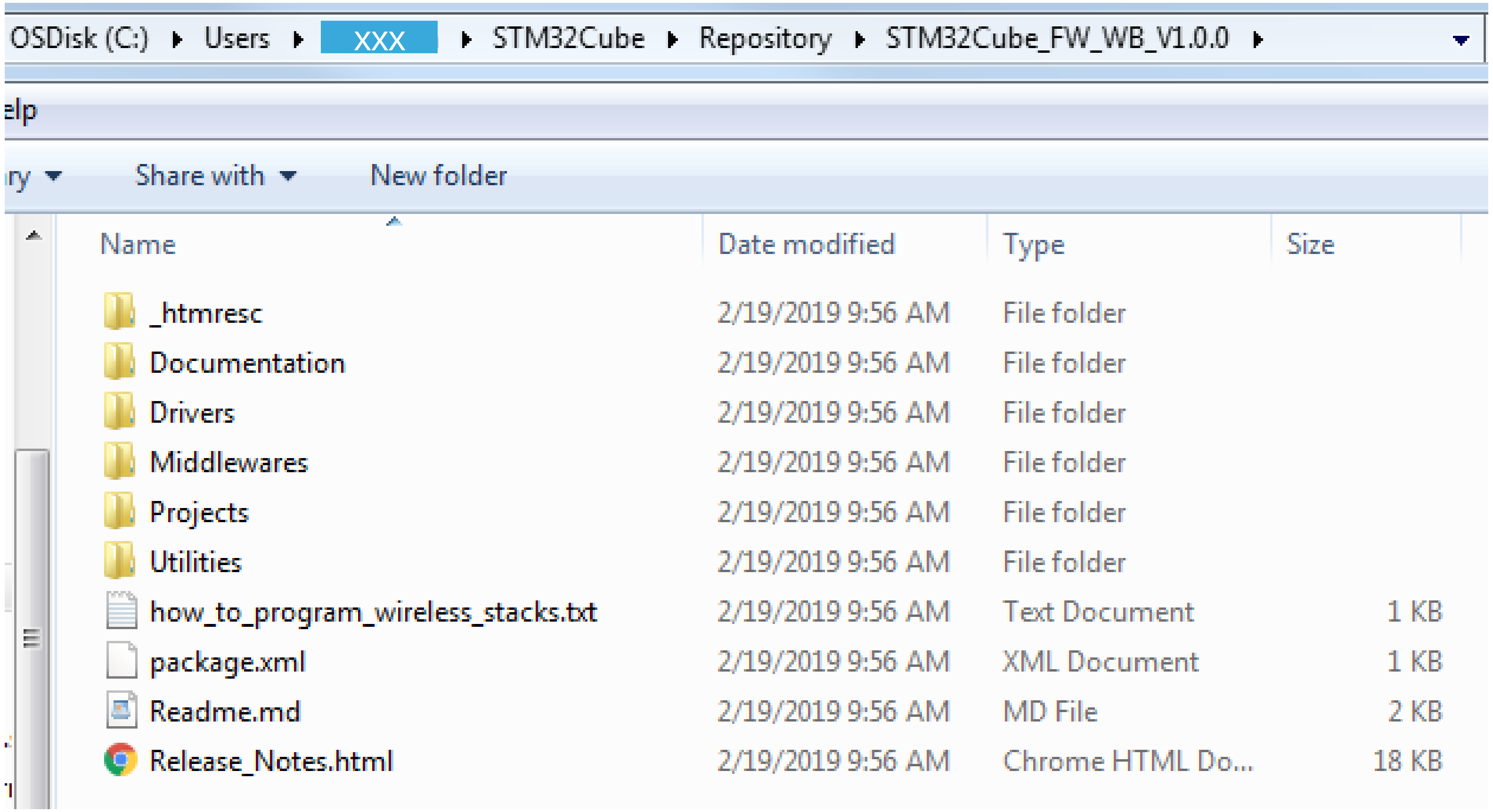


Hands-On

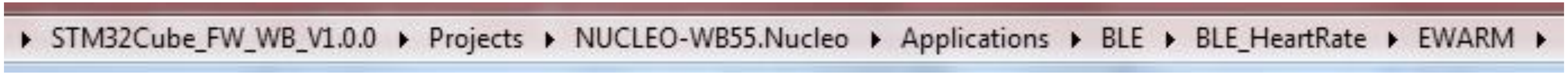
Heart-Rate Monitor

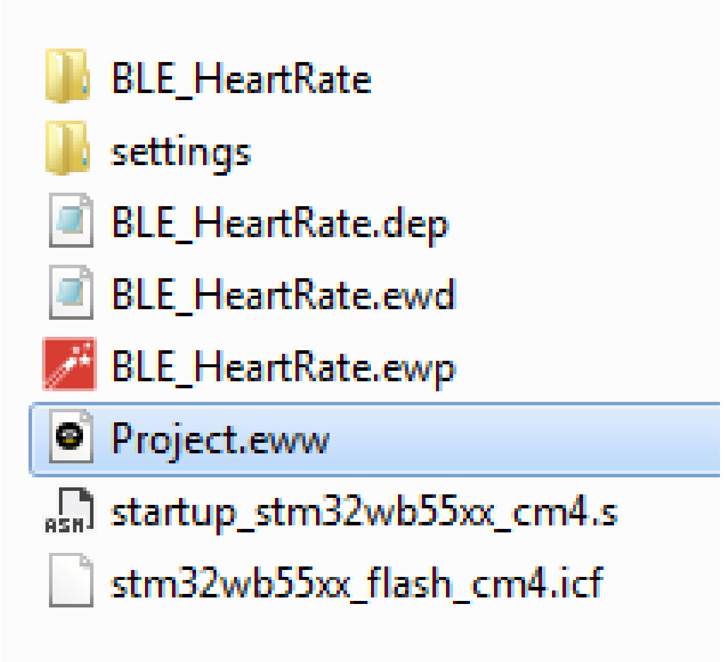
STM32 WB

All CubeWB Projects referenced today can be found in the CubeMX Repository folder:

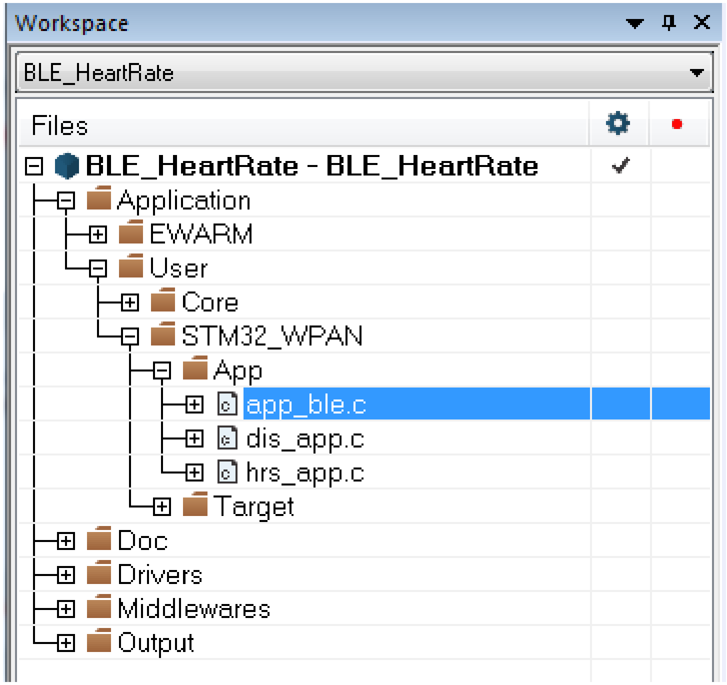


Open the BLE_HeartRate workspace





Open **app_ble.c**

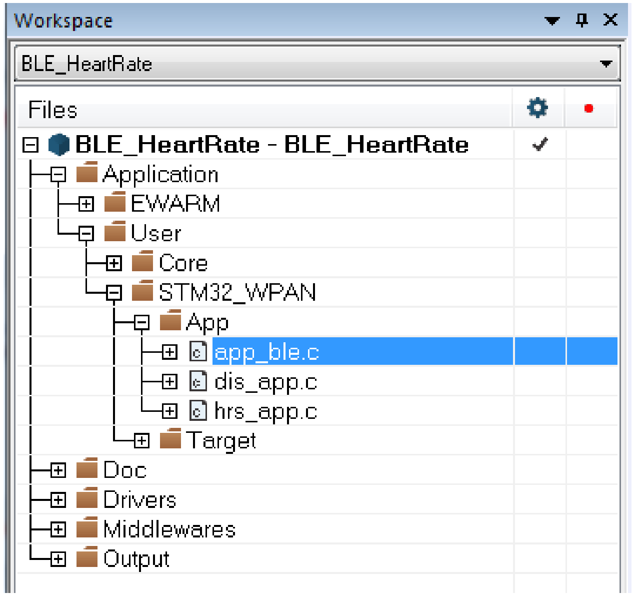


Change the advertised **local name**, using your *Magic Number*!

*(You can change it as you wish, however keep the # of ASCII chars to **5**)*

```
static const char local_name[] = { AD_TYPE_COMPLETE_LOCAL_NAME ,'H','R','S','T','M'};
uint8_t  manuf_data[14] = {
```

```
static const char local_name[] = { AD_TYPE_COMPLETE_LOCAL_NAME ,'S','T','M','1','2'};
```

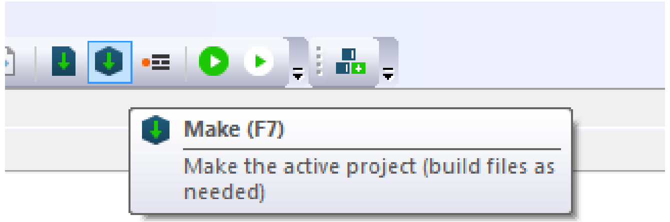


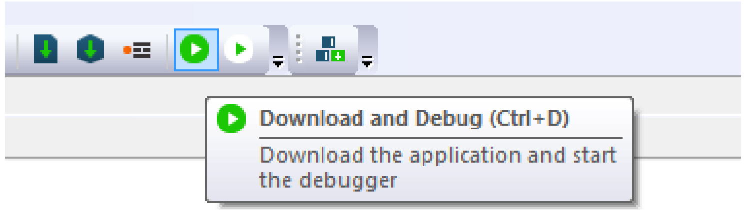
Also change the BLE Device **name** and the **NAME_LENGTH**, using your *Magic Number*!

```
/* Private defines -----------------------
#define APPBLE_GAP_DEVICE_NAME_LENGTH 9
```

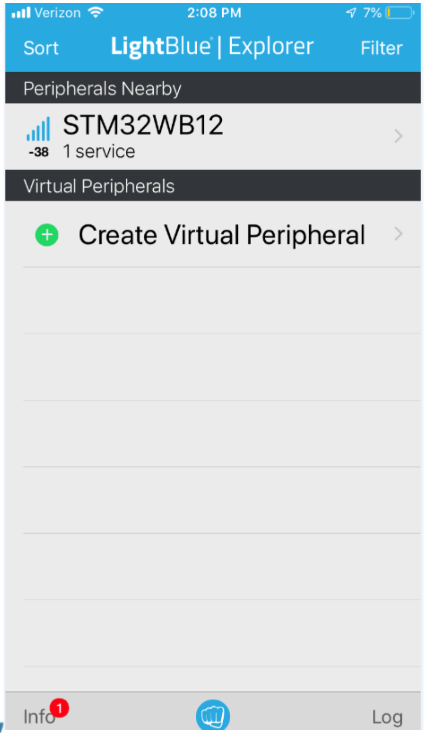
```
  if (role > 0)
  {
    const char *name = "STM32WB12";
    aci_gap_init(role, 0,
```

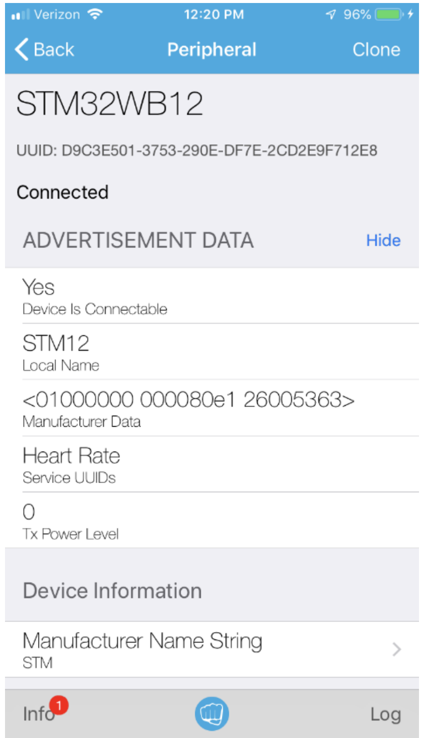
Build, Program and Launch debugger

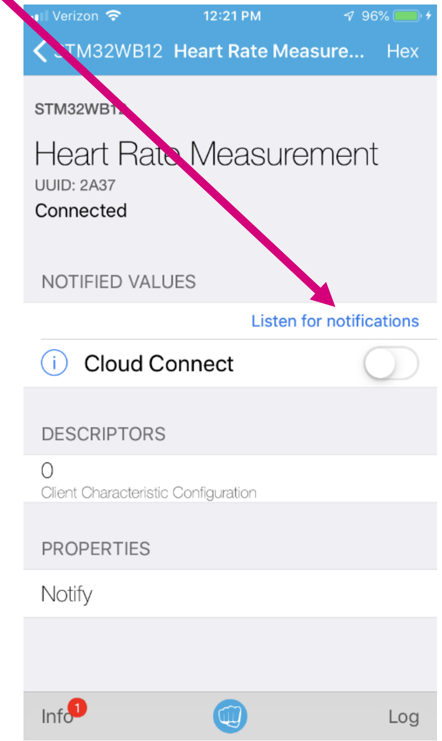


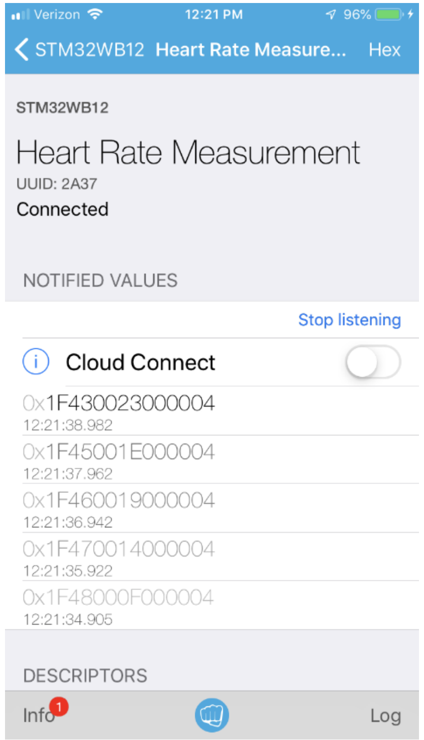


- Open your LightBlue Explorer App on **iOS**
- Find your device and tap on it
- **Show** Advertisement Data
- Click on the Heart Rate Measurement and Enable Notifications



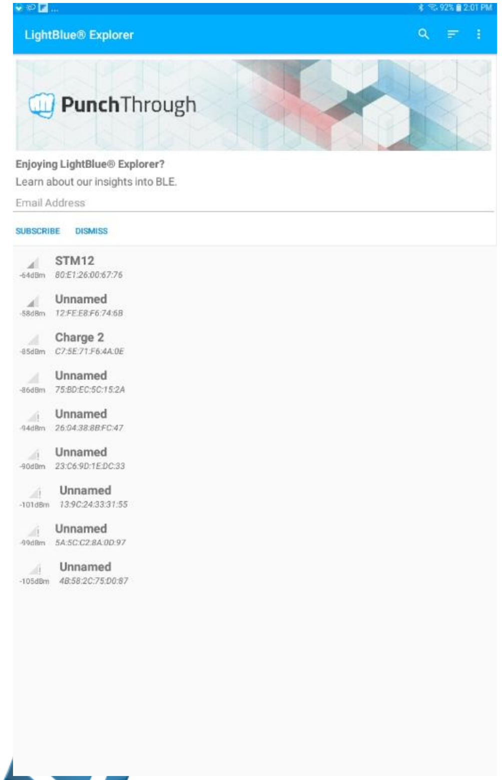


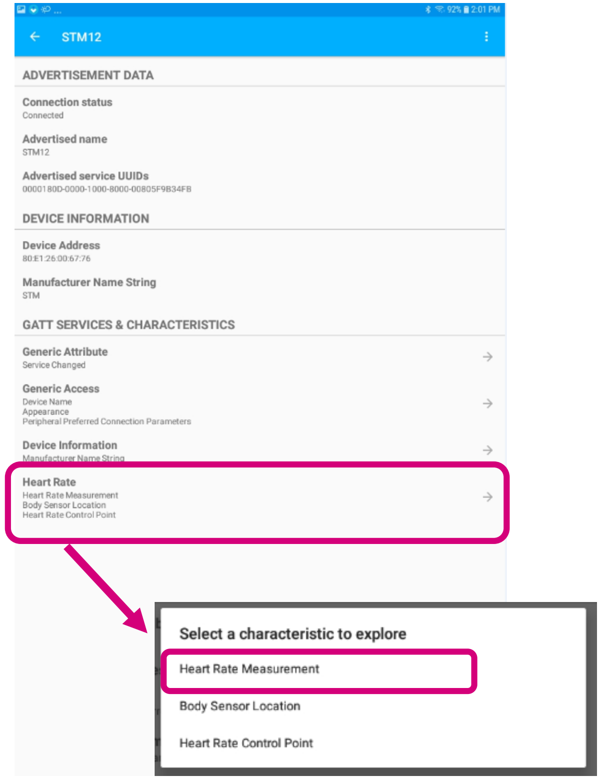


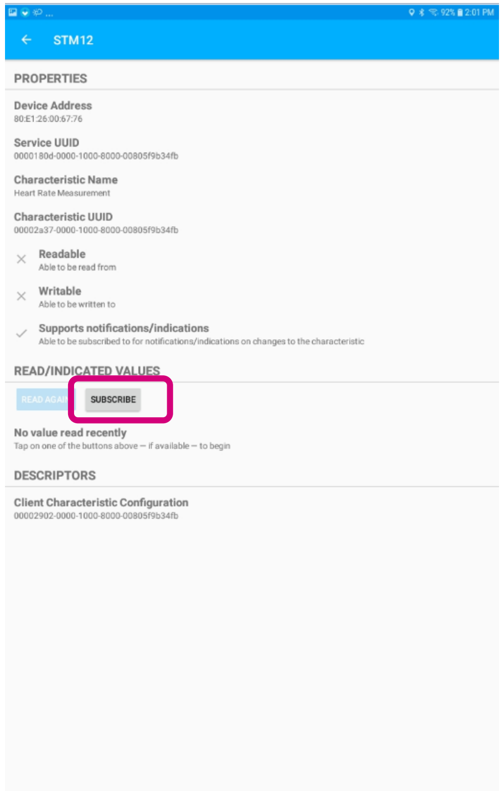


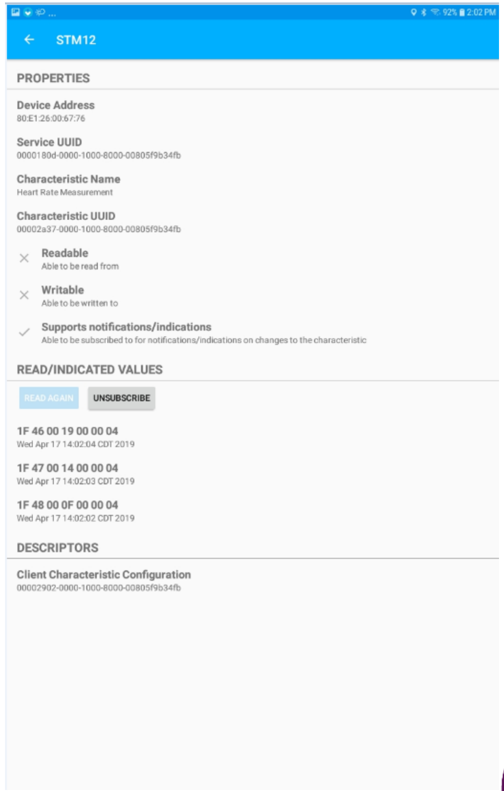
life.augmented

- Open your LightBlue Explorer App on **Android**
- Find your device and tap on it
- Tap on the Heart Rate section and select Heart Rate Measurement
- Tap on “SUBSCRIBE” to Enable Notifications

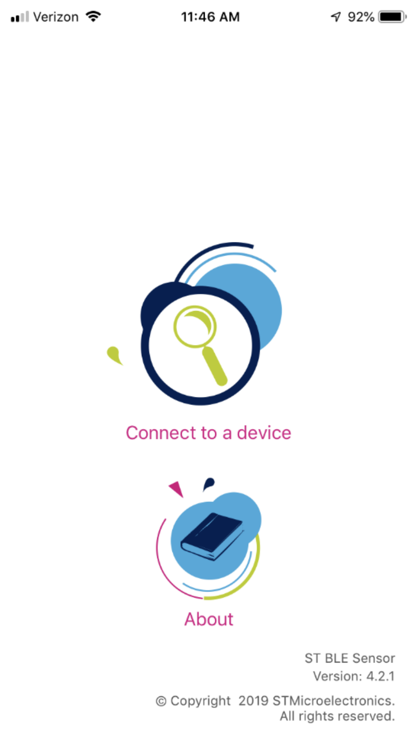


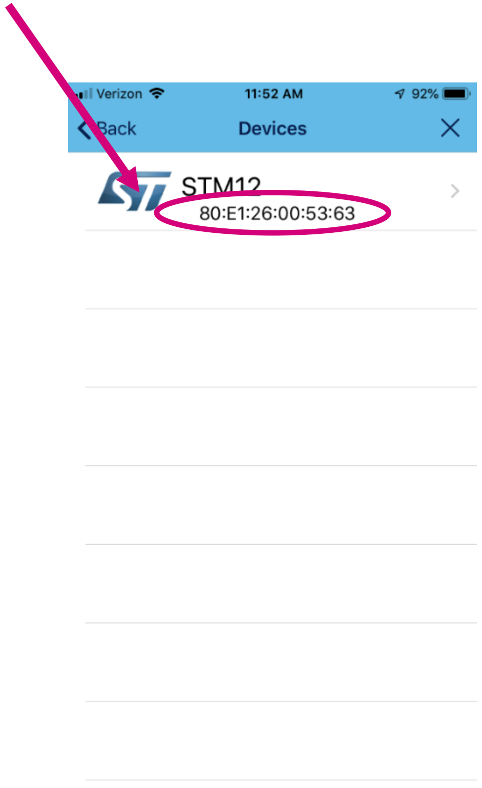


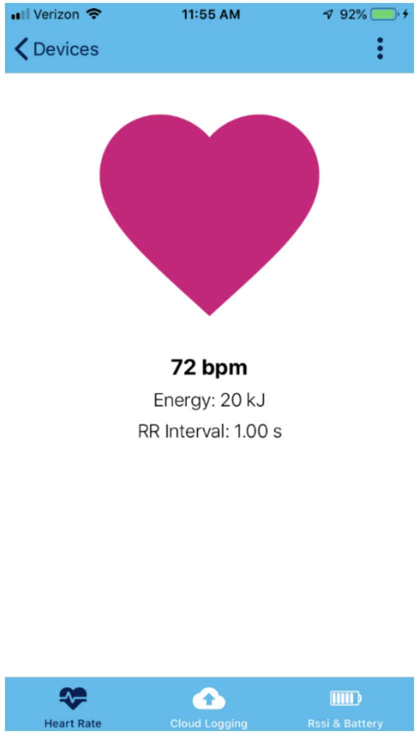




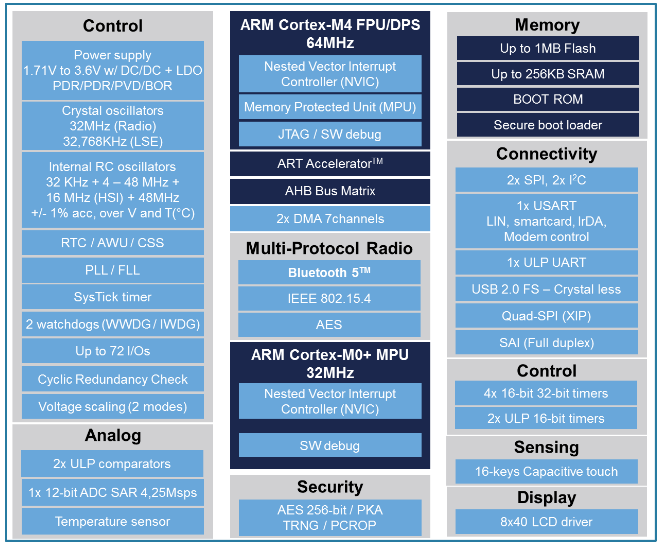
- Disconnect from the LightBlue Explorer App
- Launch the ST BLE Sensor App
- Tap on your device name
- Write down your Nucleo Bluetooth Device Address.
    - ➢ *Can you find it in the Mfr-Specific advertised data via LightBlue app?*







WB Architecture



life.augmented

STM32 WB

Balun – Combine TX and RX signals

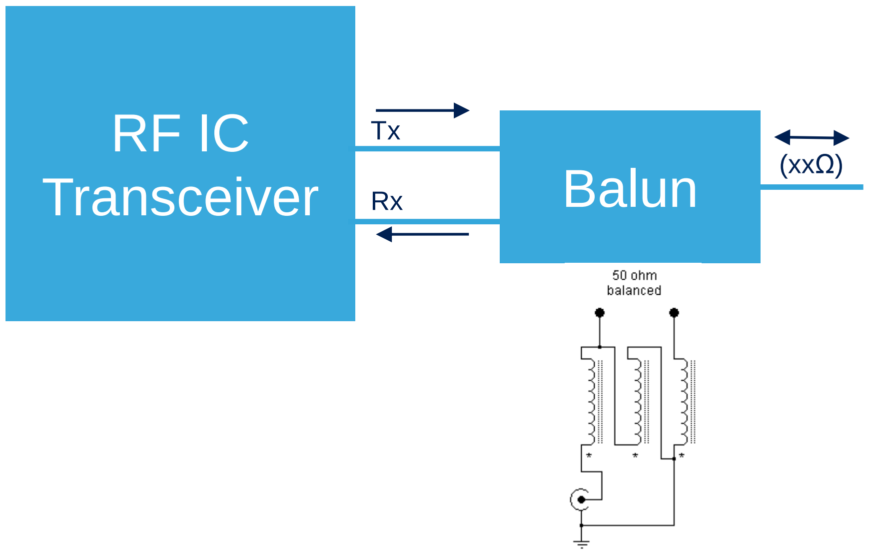
Matching Network – 50 Ω impedance transformation

Harmonic Filter – Reduce out-of-band harmonics

Balun – Combine TX and RX signals

Matching Network – 50 Ω impedance transformation

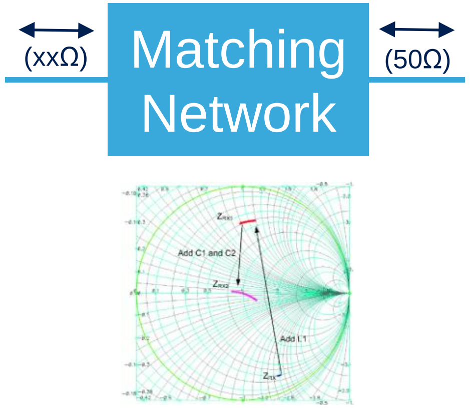
Harmonic Filter – Reduce out-of-band harmonics



Balun – Combine TX and RX signals

Matching Network – 50 Ω impedance transformation

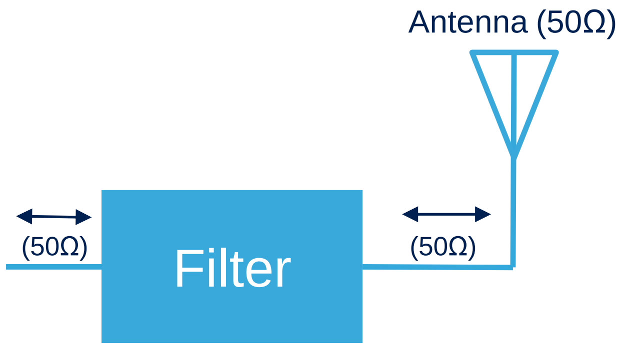
Harmonic Filter – Reduce out-of-band harmonics

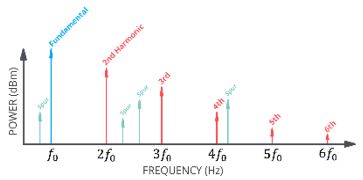


Balun – Combine TX and RX signals

Matching Network – 50 Ω impedance transformation

Harmonic Filter – Reduce out-of-band harmonics

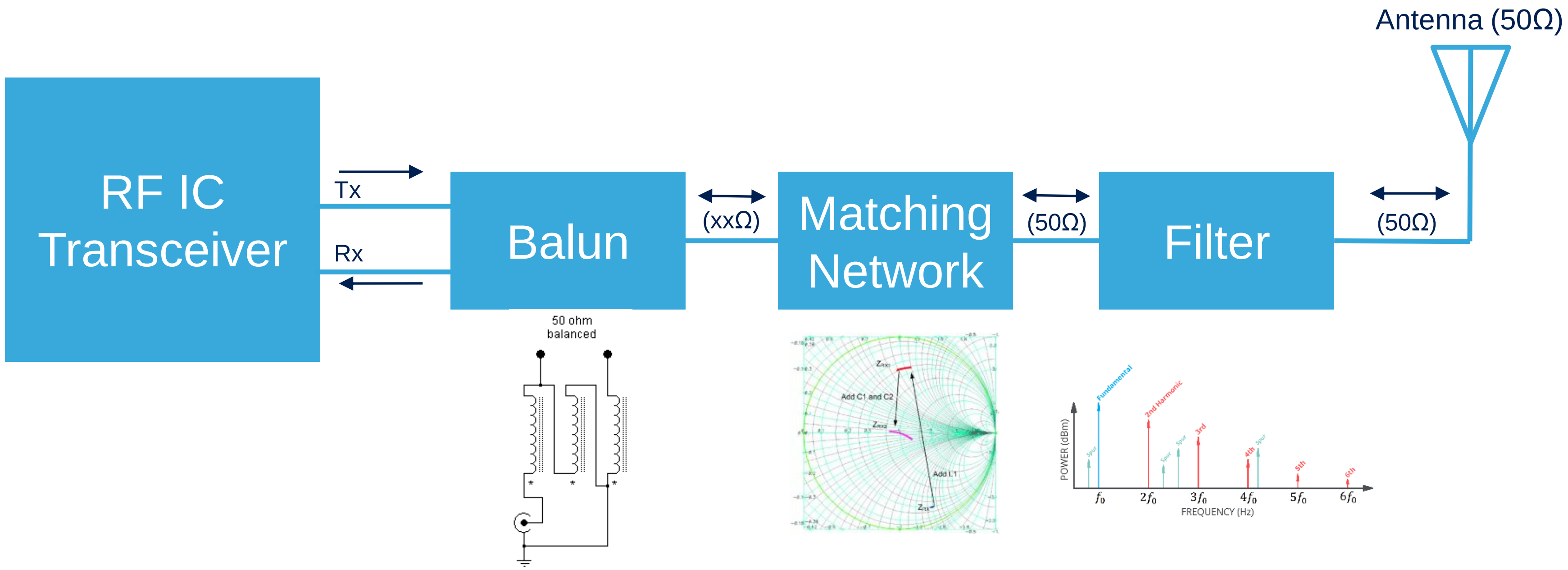




Balun – Combine TX and RX signals

Matching Network – 50 Ω impedance transformation

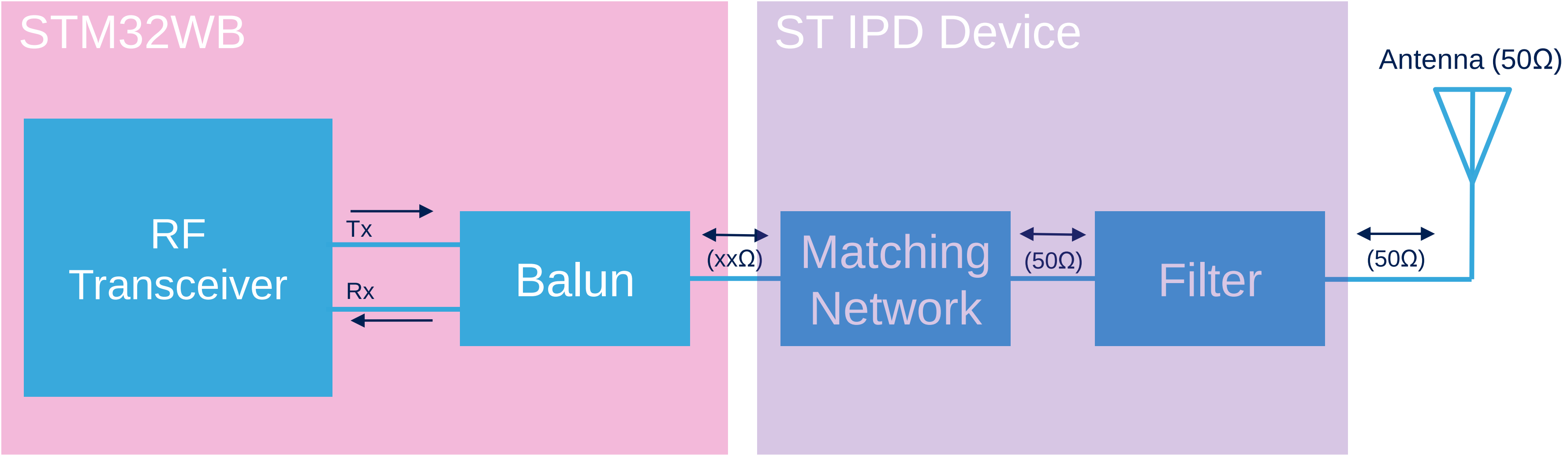
Harmonic Filter – Reduce out-of-band harmonics



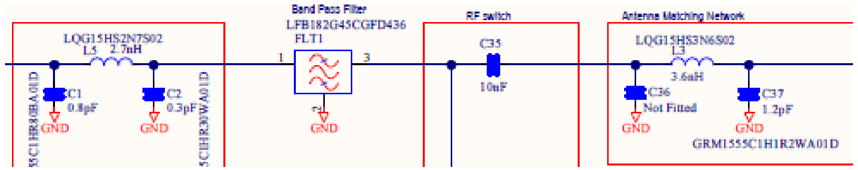
Balun – Combine TX and RX signals

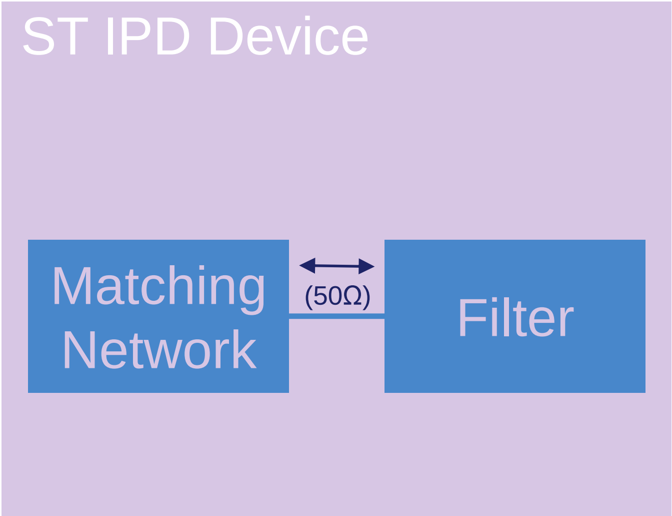
Matching Network – 50 Ω impedance transformation

Harmonic Filter – Reduce out-of-band harmonics



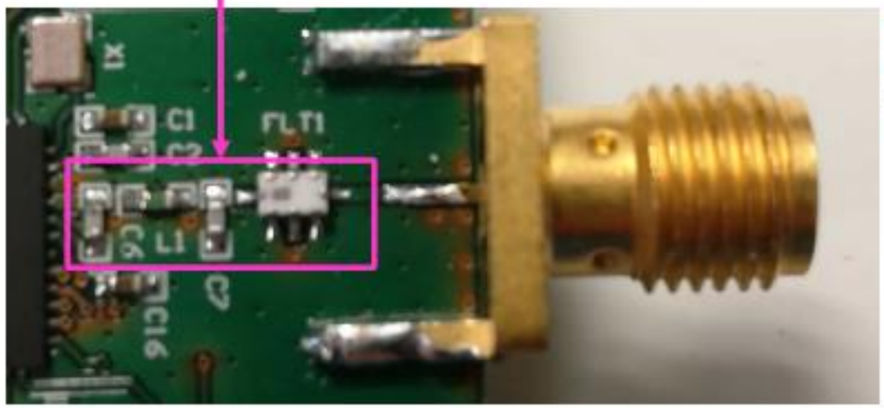
Matching Network + Harmonic Filter

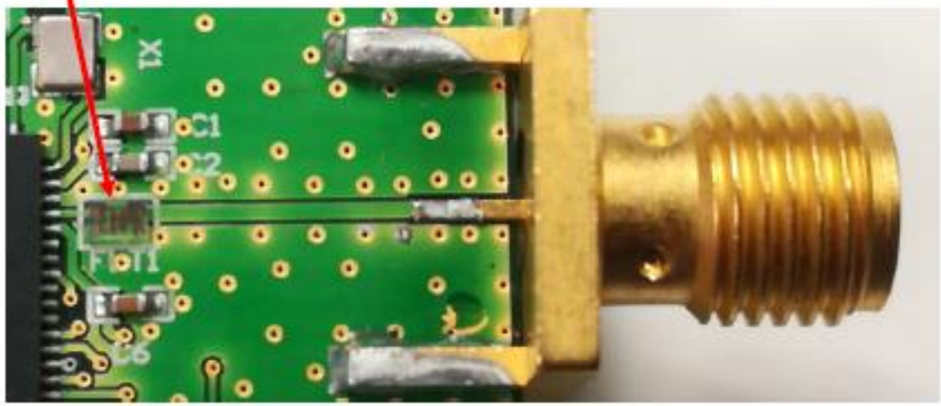




Discrete solution

IPD device from ST





Mass Production NOW

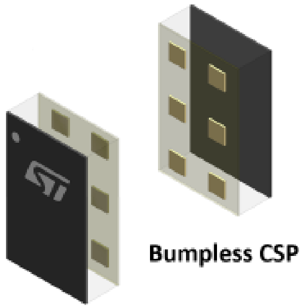
Versions coming for upcoming WLCSP / BGA packages

life.augmented

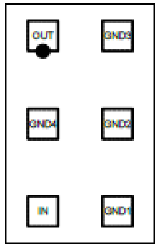
MLPF-WB55-01E3

Datasheet

2.4 GHz low pass filter matched to STM32WB55Cx/Rx



Top view (pads down)



Features

- Integrated impedance matching to STM32WB55Cx and STM32WB55Rx
- LGA footprint compatible
- 50 Ω nominal impedance on antenna side
- Deep rejection harmonics filter
- Low insertion loss
- Small footprint
- Low thickness ≤ 450 µm
- High RF performance
- RF BOM and area reduction
- ECOPACK®2 compliant

Applications

- Bluetooth 5
- OpenThread
- Zigbee®
- IEEE 802.15.4
- Optimized for STM32WB55Cx and STM32WB55Rx

1mm x 1.6mm CSP

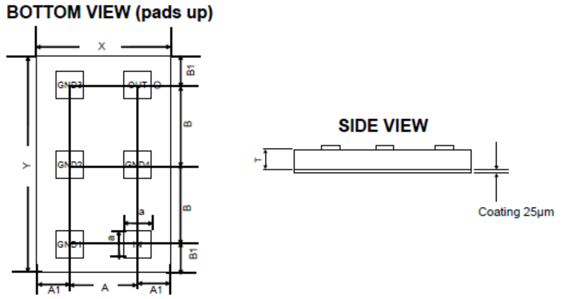
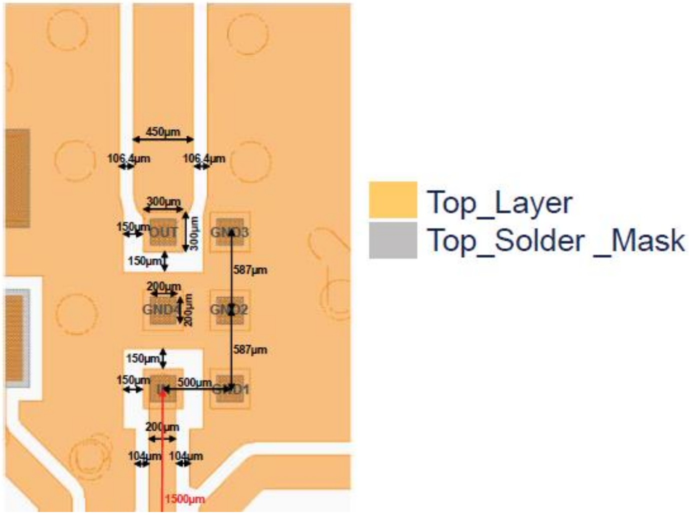


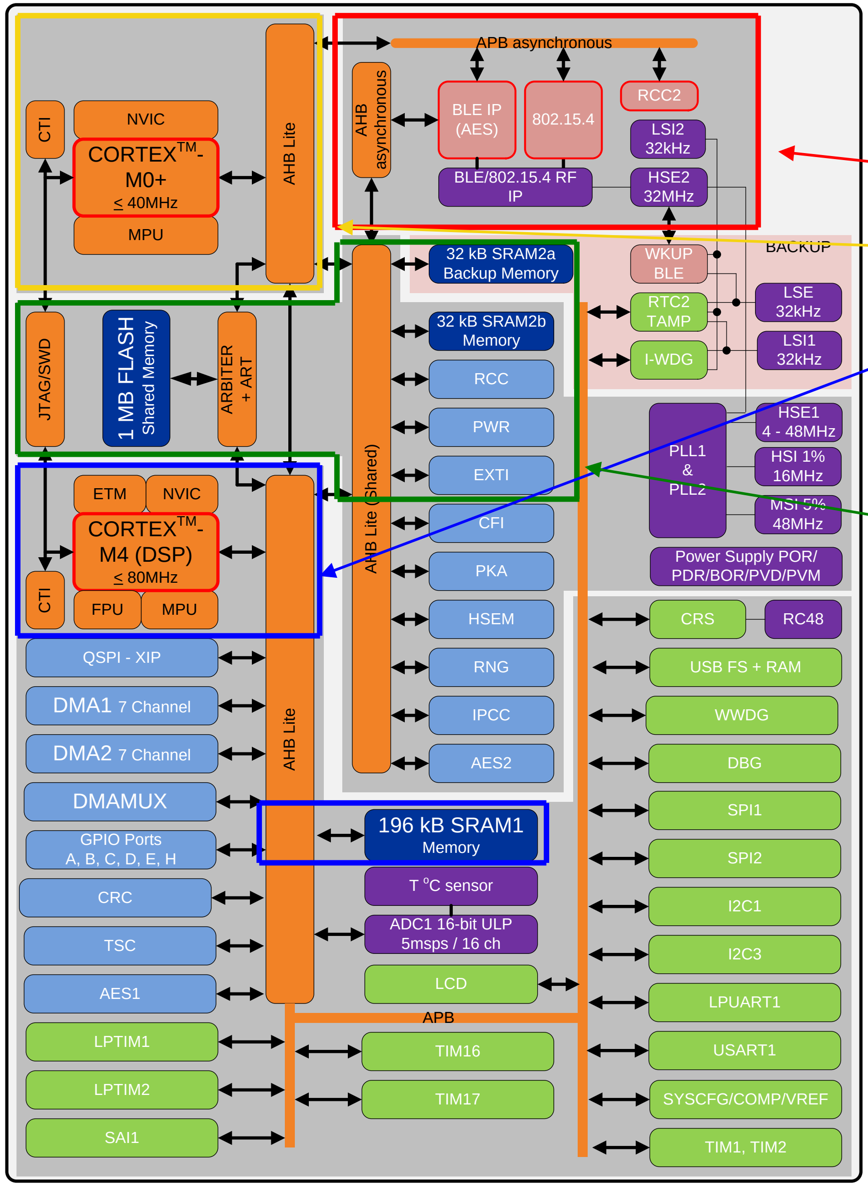
Table 4. Bumpless CSP package mechanical data

Parameter	Description	Min.	Typ.	Max.	Unit
X	X dimension of the die	975	1000	1025	µm
Y	Y dimension of the die	1575	1600	1625	µm
A	X pitch		500		µm
B	Y pitch		587		µm

PCB recommendations included in datasheet

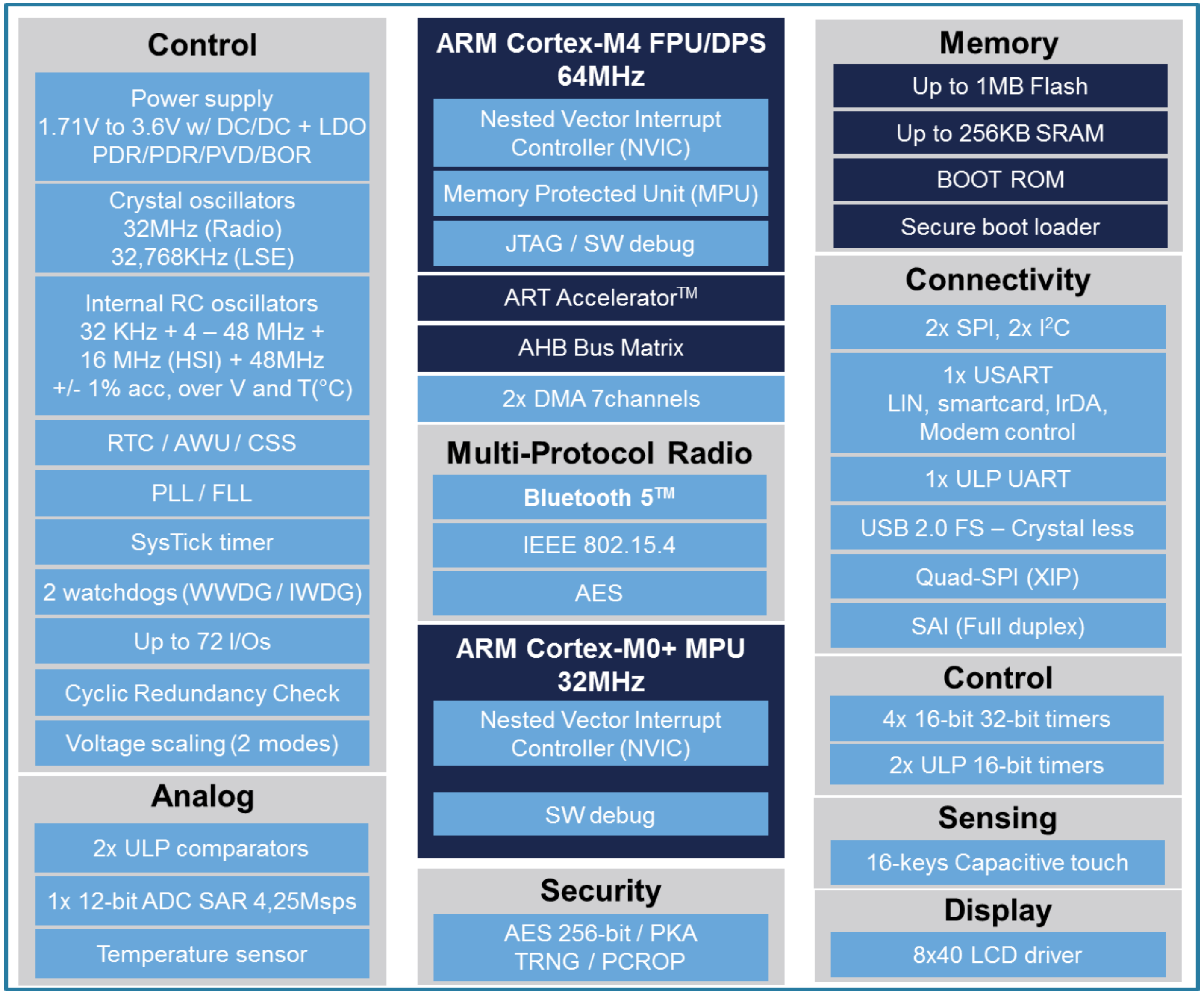
Figure 13. PCB land pattern recommendations





- 3 autonomous sub-systems
  - Radio sub-system
  - Cortex-M0+ (CPU2)
  - Cortex-M4 (CPU1)
- Common run domain
  - Flash, SRAM2, RCC, PWR, EXTI

- Radio with integrated balun
  - Output power: +6.0 dBm
  - BLE RX sensitivity: -96 dBm
  - 802.15.4 RX sensitivity: -100 dBm
  - RX: 4.5mA
  - TX: 5.2mA (0dBm)
- -40°C to +105°C
- Packages
  - QFN48 / 68
  - WLCSP100
  - BGA129



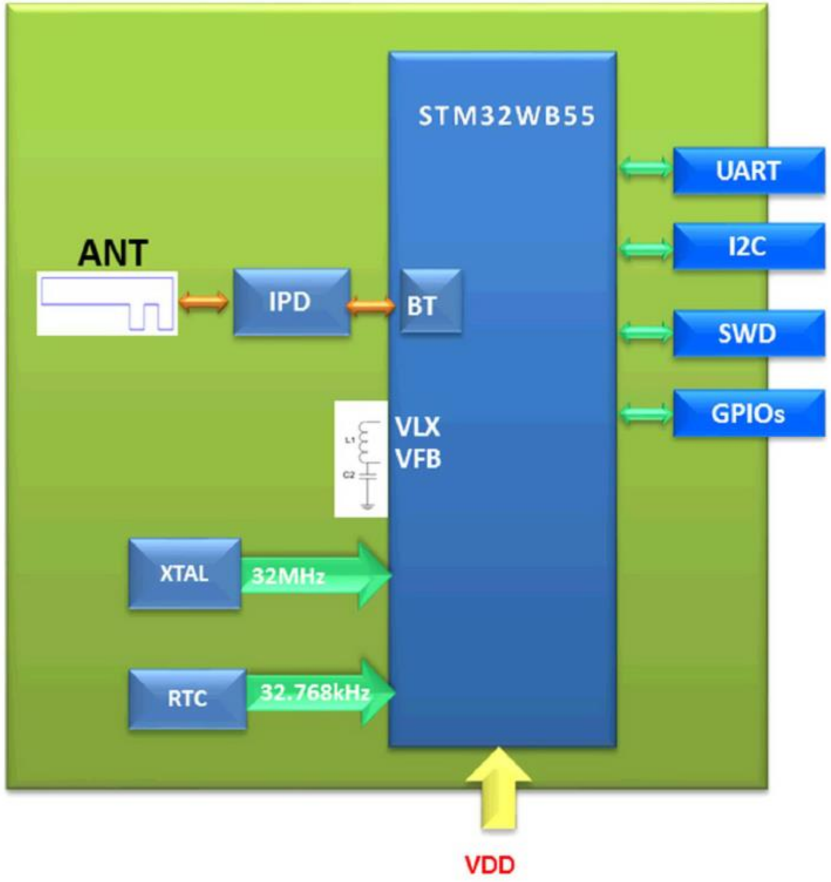
life.augmented

STM32 WB

STM32WB55 Series Portfolio

Flash / RAM	48 pins 7x7mm UQFN (p=0.5)	68 pins 8x8mm VQFN (p=0.4)	100 pins 4.39x4.37mm WLCSP (p=0.4)	129 pins 7x7mm BGA (p=0.5)
1M / 256K	STM32WB55CG	STM32WB55RG	STM32WB55VG	
512K / 256K	STM32WB55CE	STM32WB55RE	STM32WB55VE	
256K / 128K	STM32WB55CC	STM32WB55RC	STM32WB55VC	

- ST Branded
- Pre-Certified
- Chip Antenna
- 10x10mm
- Large GPIO count
- Pin pitch = 2 layer PCB-ready
- Production in early 2020.



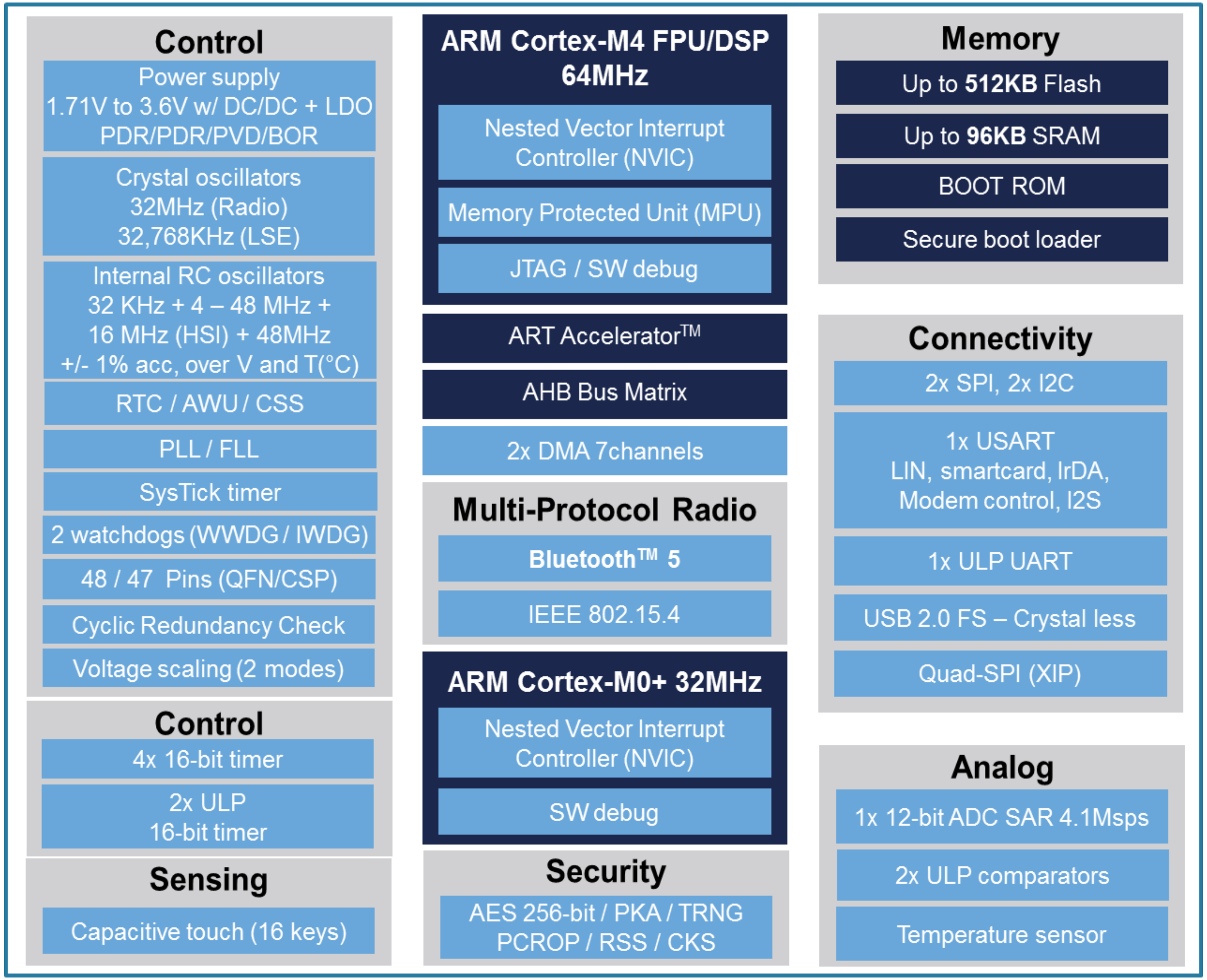
STM32WB35 – Block Diagram

256KB or 512KB Flash

96KB SRAM

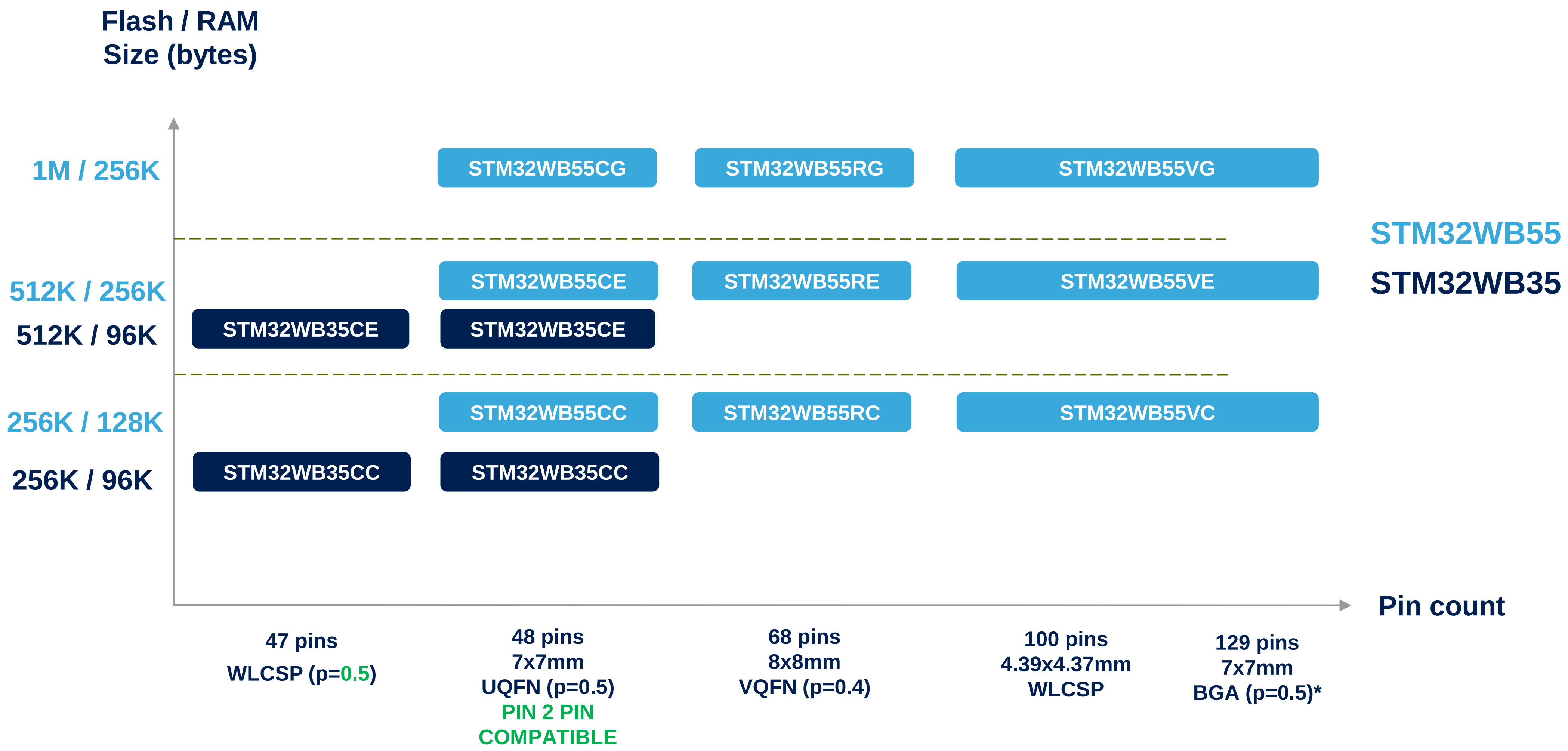
- QFN48
- WLCSP47

Late 2019



life.augmented

Positioning

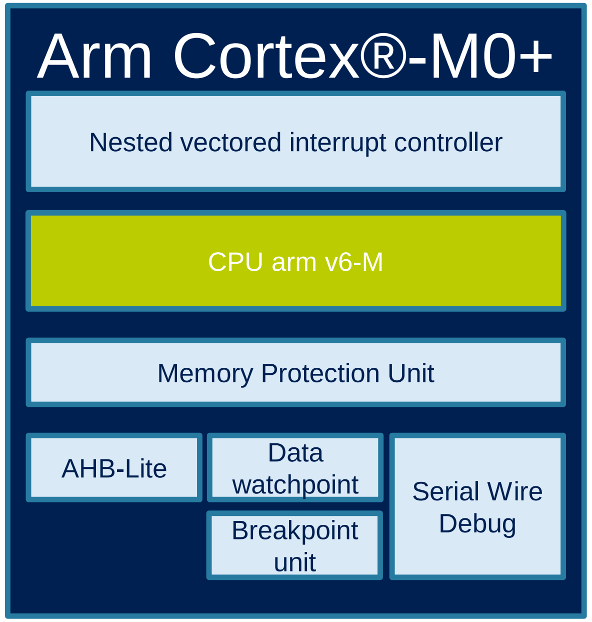


* Exact part number on
BGA 129 To be confirmed

BLE4.1	BLE4.2	BLE5.0	BLE5.0 IEEE 802.15.4
BlueNRG-MS	**BlueNRG-1 / 2**	**BlueNRG-LP**	**STM32WB**
ARM Cortex-M0 Core RX: 7.3mA TX: 8.2mA (0dBm) Sensitivity: -88dBm	ARM Cortex-M0 Core 160KB / 256KB Flash 24KB RAM I2C, SPI, UART, ADC RX: 7.3mA TX: 8.2mA (0dBm) Sensitivity: -88dBm	ARM Cortex-M0+ Core 256KB Flash 48KB RAM I2C, SPI, UART, ADC RX: 4.5mA TX: 5.2mA (0dBm) Sensitivity: -96dBm	ARM Cortex-M4F Core 1MB Flash 256KB RAM I2C, SPI, UART, QSPI, USB, ADC, LCD ARM Cortex-M0+ Core RX: 4.5mA TX: 5.2mA (0dBm) Sensitivity: -96dBm
NETWORK PROCESSOR	SINGLE-CORE		DUAL-CORE

APPLICATIONS PROCESSOR

- ARMv6-M architecture
- Von Neumann architecture
- 2-stage pipeline
- Single-issue architecture
- Single-cycle MULTIPLY



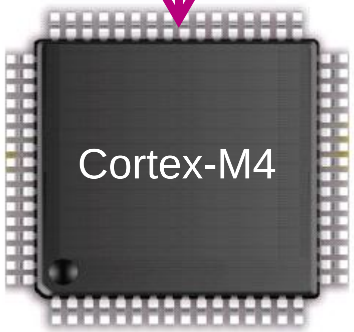
ARM® Cortex® M4 Core

FPU

Single precision
Better code efficiency
Eliminate scaling and saturation
Support meta-language tools (MATLAB, etc)

MCU

Easy C programming
Interrupt handling
Ultra-low power



DSP

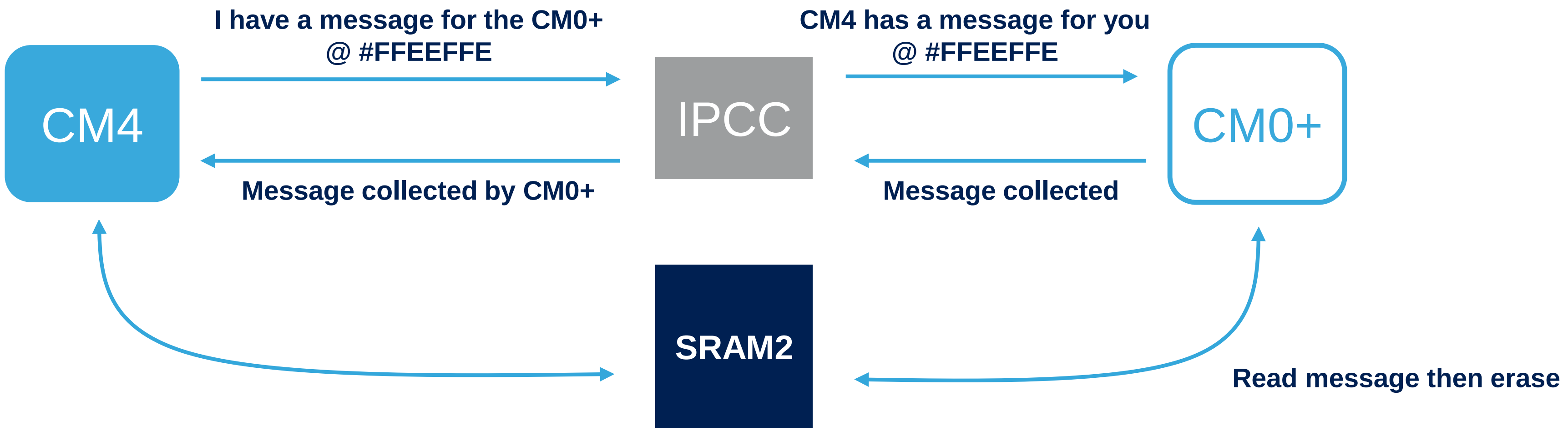
Harvard architecture
Single-cycle MAC
Barrel shifter

http://www.arm.com/products/processors/cortex-m/cortex-m4-processor.php

Dual core – How does that work?

IPCC: Inter Processor Communication Controller

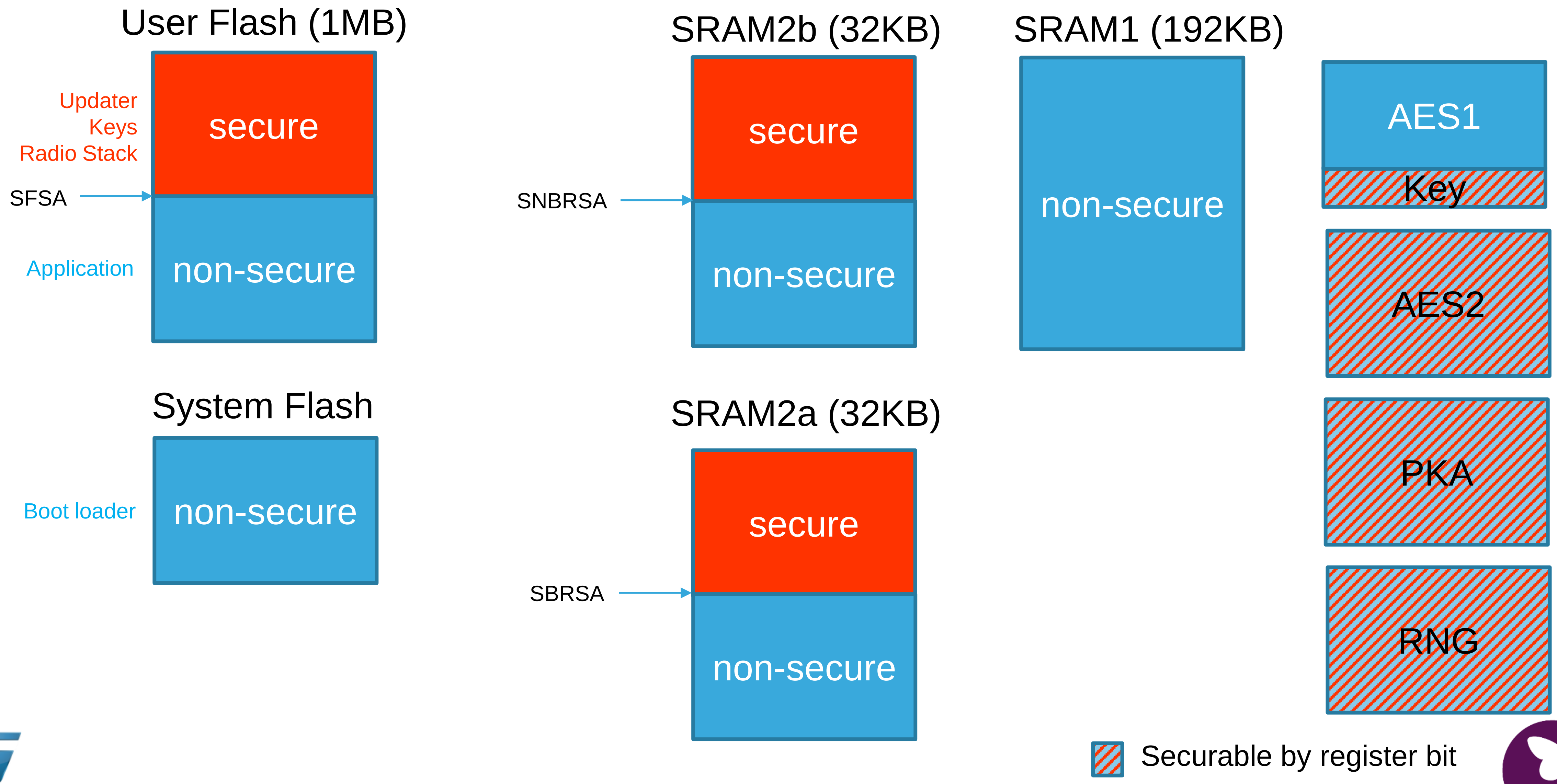
HSEM: Hardware Semaphore – prevent shared resource access conflicts



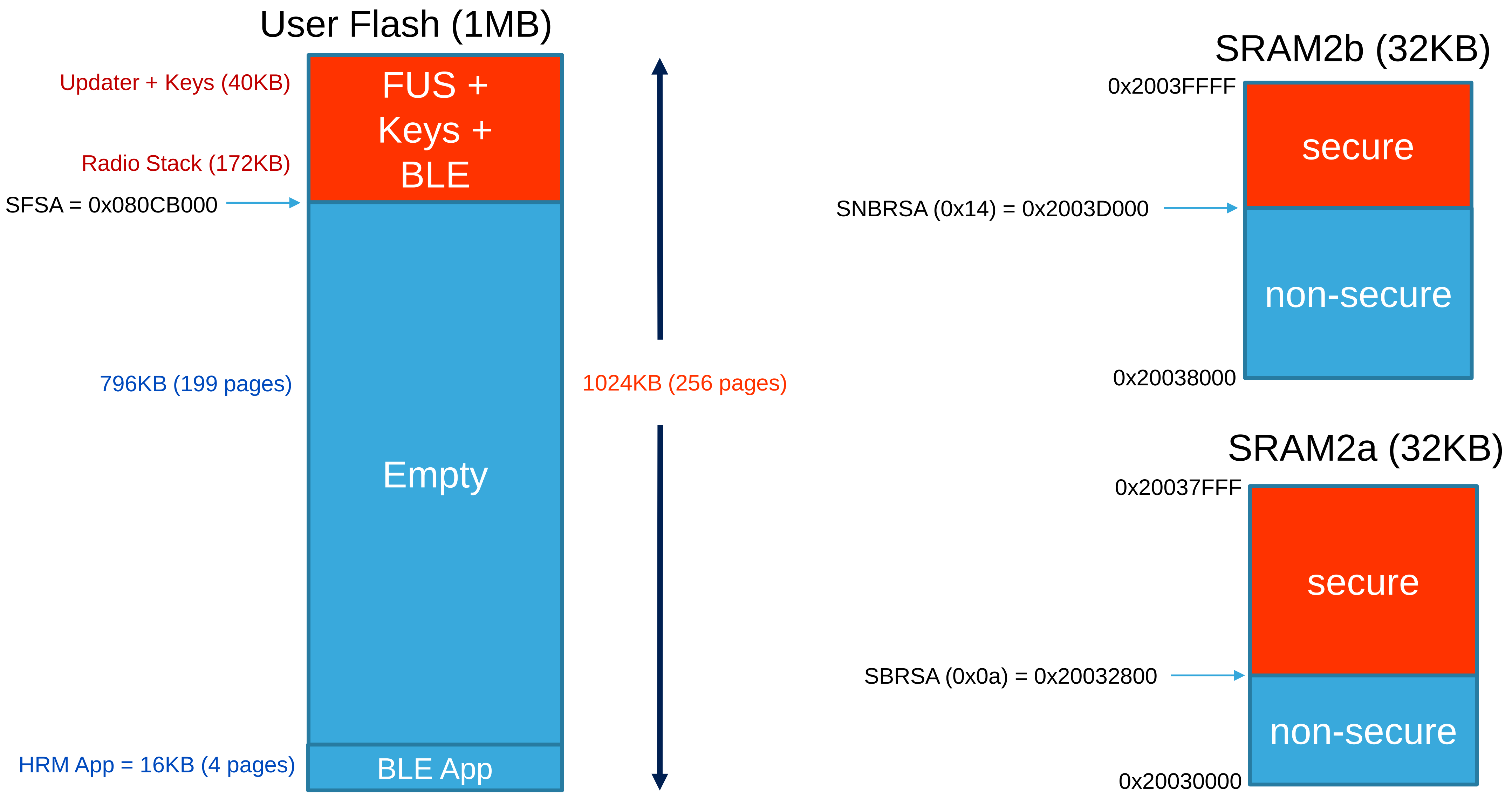
IPCC works in both directions

life.augmented

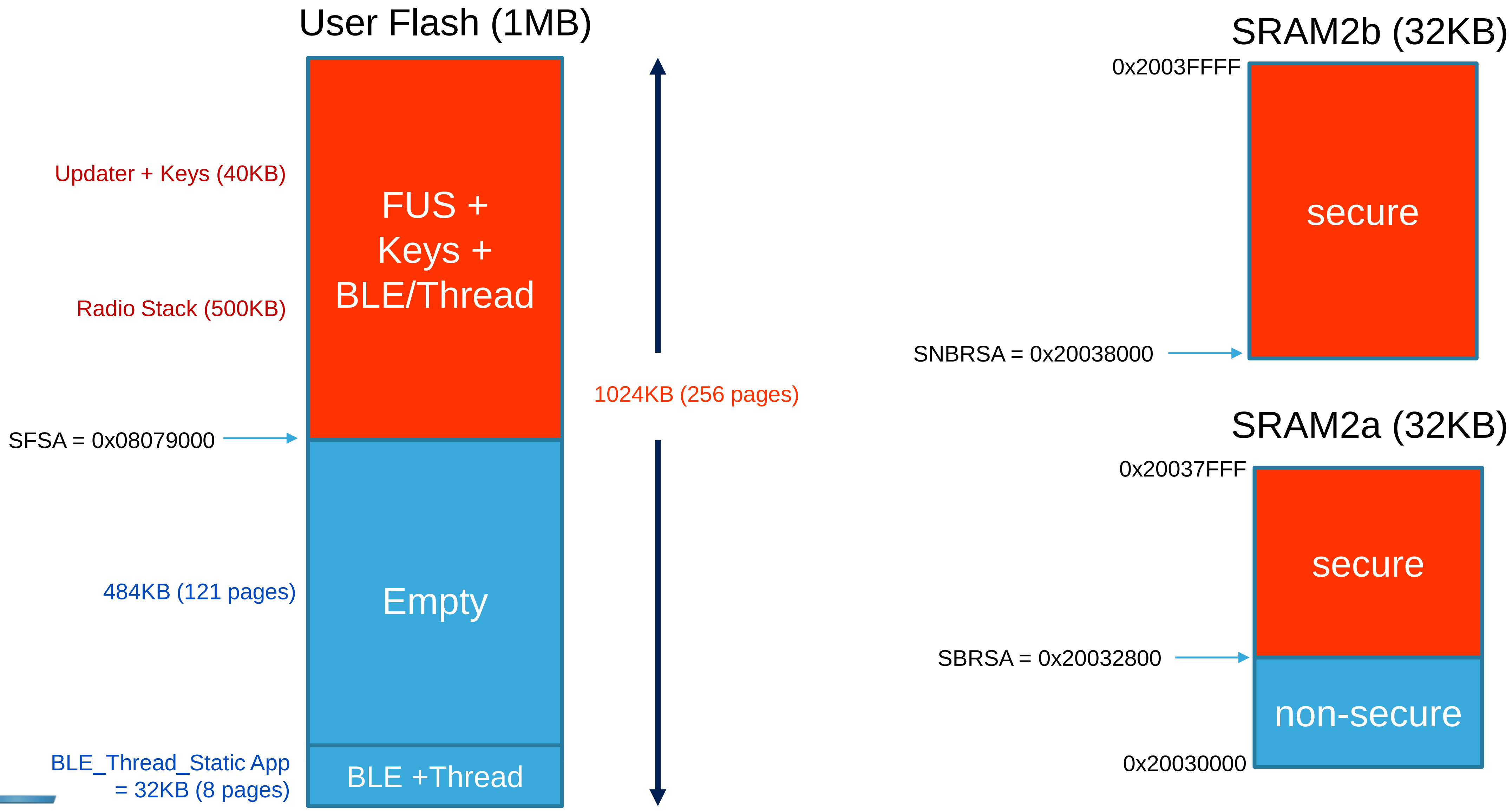
Memory Partitioning



Memory Partitioning: BLE Stack



Memory Partitioning: BLE+Thread (Concurrent) Stack



life.augmented

STM32 WB

Release Notes for

STM32WB Copro Wireless Binaries

License

This software component is licensed by ST under Ultimate Liberty license SLA0044, the “License”;

You may not use this file except in compliance with the License.

You may obtain a copy of the License at: SLA0044

Purpose

This release covers the delivery of STM32WB Coprocessor binaries.

Here is the list of the supported binaries:

- stm32wb5x_BLE_Stack_fw.bin
  - Full BLE Stack 5.0 certified : Link Layer, HCI, L2CAP, ATT, SM, GAP and GATT database
  - BT SIG Certification listing : Declaration ID D042164
- stm32wb5x_BLE_HCILayer_fw.bin
  - HCI Layer only mode 5.0 certified : Link Layer, HCI
  - BT SIG Certification listing : Declaration ID D042213
- stm32wb5x_Thread_FTD_fw.bin
  - Full Thread Device certified v1.1
  - To be used for Leader / Router / End Device Thread role (full features excepting Border Router)

Update History

V1.0.0 / 06-February-2019

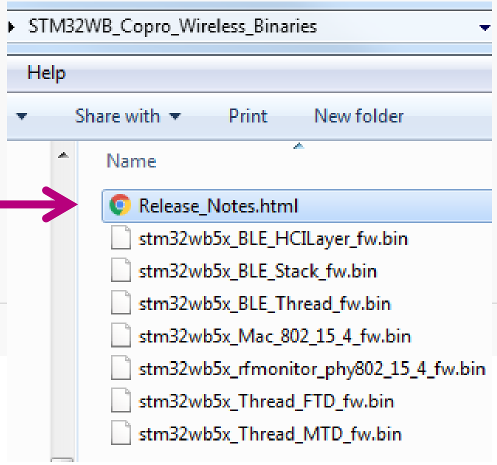
Main Changes

First release

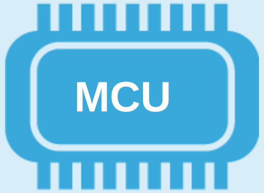
First official release.

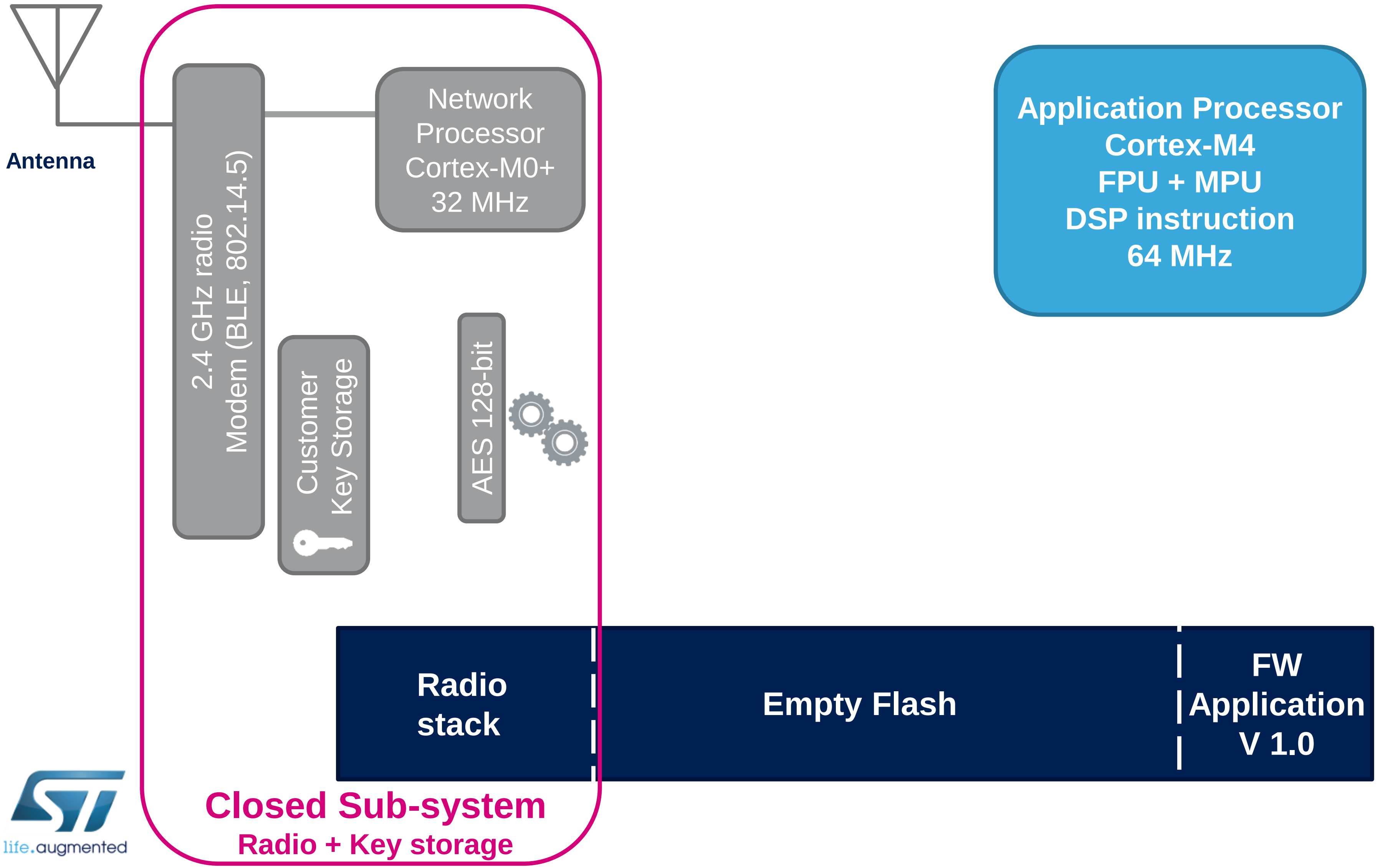
Binary Install Address and version : Provides Install address for the targeted binary to be used in “STEP 4” of flash procedure.

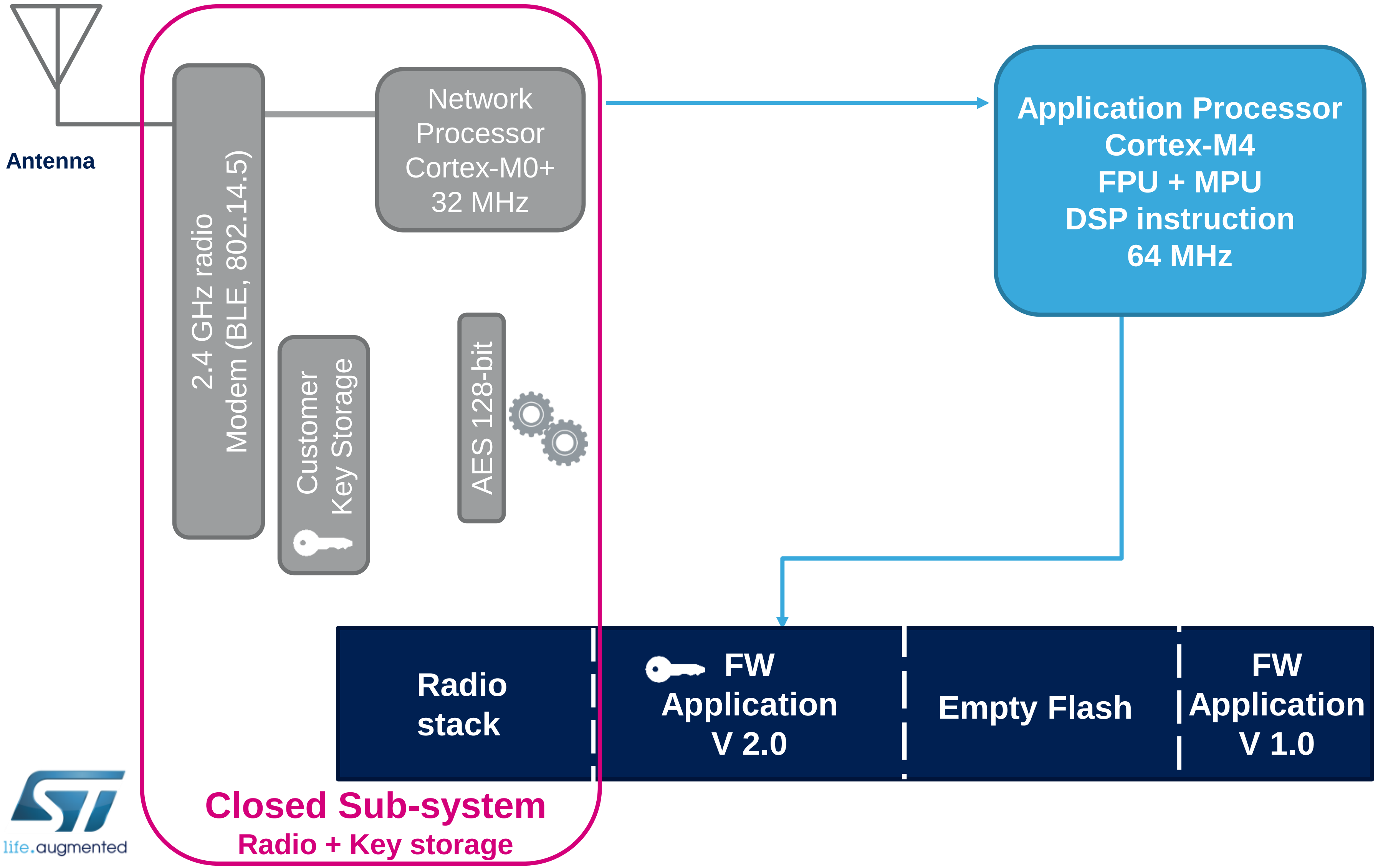
Wireless Processor Binary	Install address	Version	Date
stm32wb5x_BLE_Stack_fw.bin	0x080CB000	v1.0.0	02/06/2019
stm32wb5x_BLE_HCILayer_fw.bin	0x080CD000	v1.0.0	02/06/2019
stm32wb5x_Thread_FTD_fw.bin	0x0809F000	v1.0.0	02/06/2019
stm32wb5x_Thread_MTD_fw.bin	0x080B5000	v1.0.0	02/06/2019
stm32wb5x_BLE_Thread_fw.bin	0x08079000	v1.0.0	02/06/2019
stm32wb5x_Mac_802_15_4_fw.bin	0x080E5000	v1.0.0	02/06/2019
stm32wb5x_rfmonitor_phy802_15_4_fw.bin	0x080EA000	v1.0.0	02/06/2019



For complete documentation on STM32WBxx, visit: [www.st.com/stm32wb]

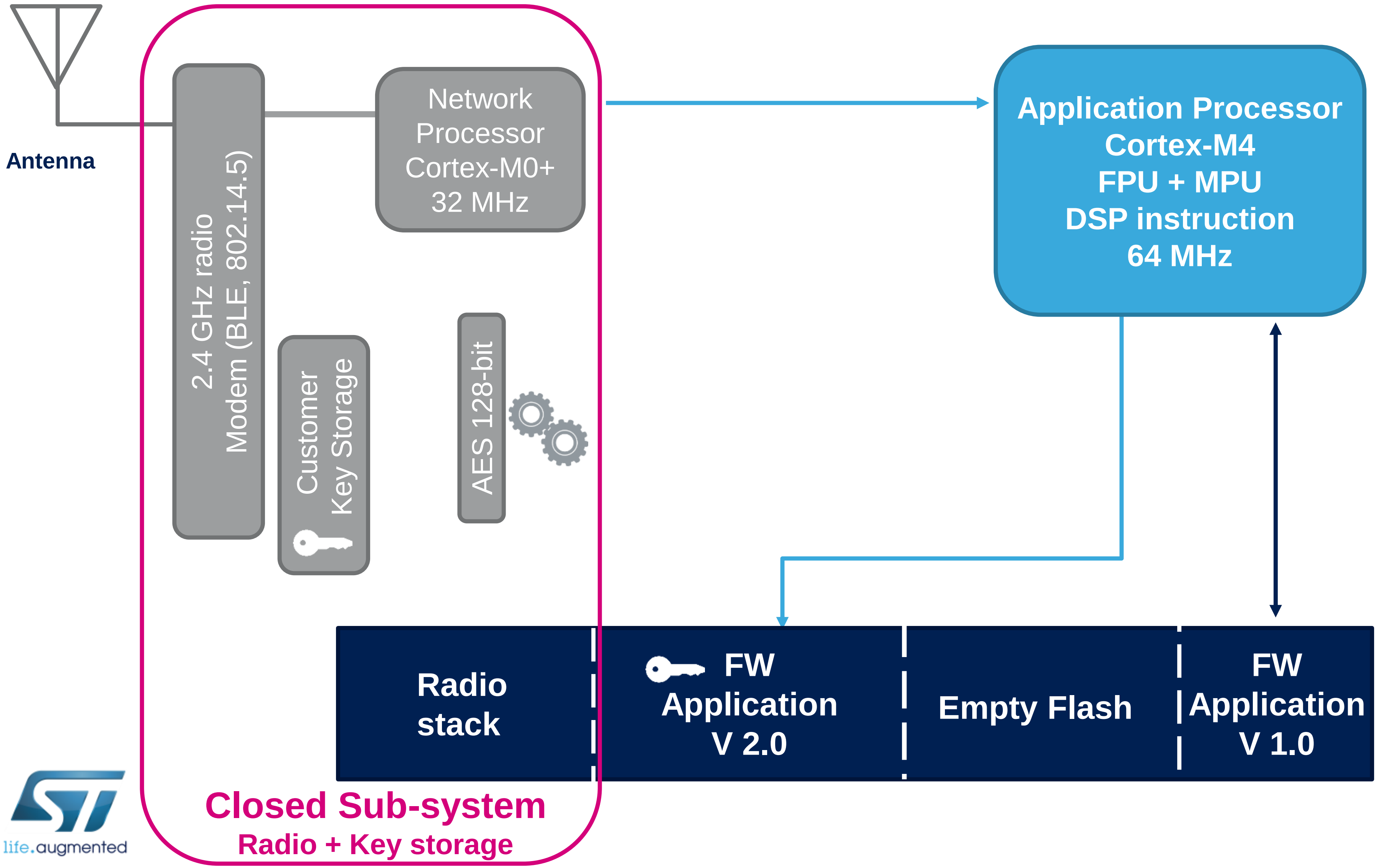
Attacks	Attacks description	STM32WB Countermeasures
Non Invasive  	• Environment • Temp / Voltage / Clocks • Fault injection • Exploit debugger • Side channel • Power Analysis	• Temp sensor • Power supply monitor • Clock security system • Tamper pads • ECC, Parity check • SRAM mass erase • Read out protection • Flash-only boot
Software 	• Break the encryption • Extract keys • Exploit debugger / test modes • Malware • Replay	• Customer Key Storage • RNG, Crypto accelerator, CRC • Readout / Write memory protections • Memory Protection Unit • Root Security Service • Secure Firmware Update (SFU) • 96-bit Unique ID


Antenna
2.4 GHz radio
Modem (BLE, 802.14.5)
Network
Processor
Cortex-M0+
32 MHz
Customer
Key Storage
AES 128-bit
Application Processor
Cortex-M4
FPU + MPU
DSP instruction
64 MHz
Radio
stack
Empty Flash
FW
Application
V 1.0
Closed Sub-system
Radio + Key storage



1 New FW package received

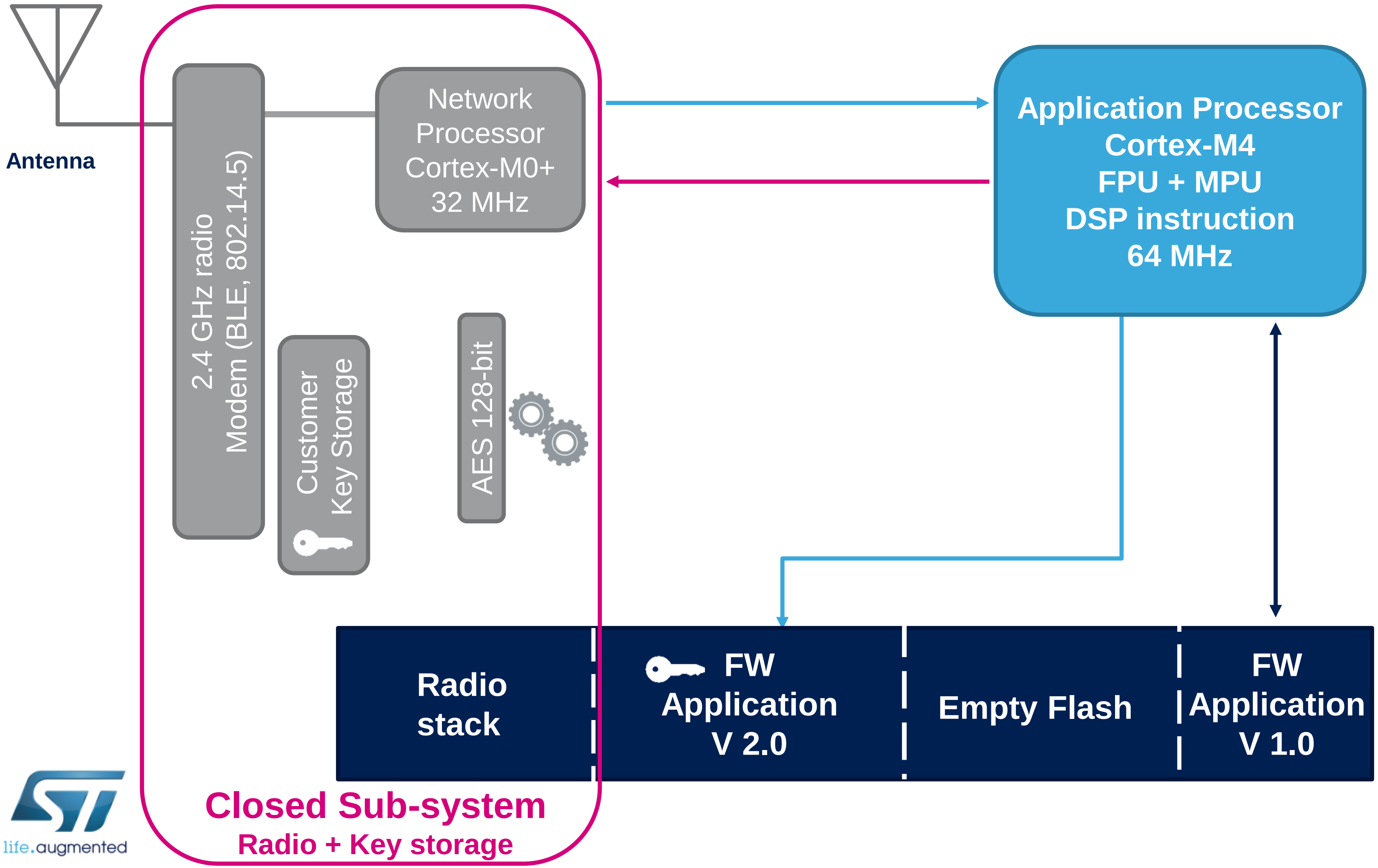
OTA Update Scenario



1 New FW package received

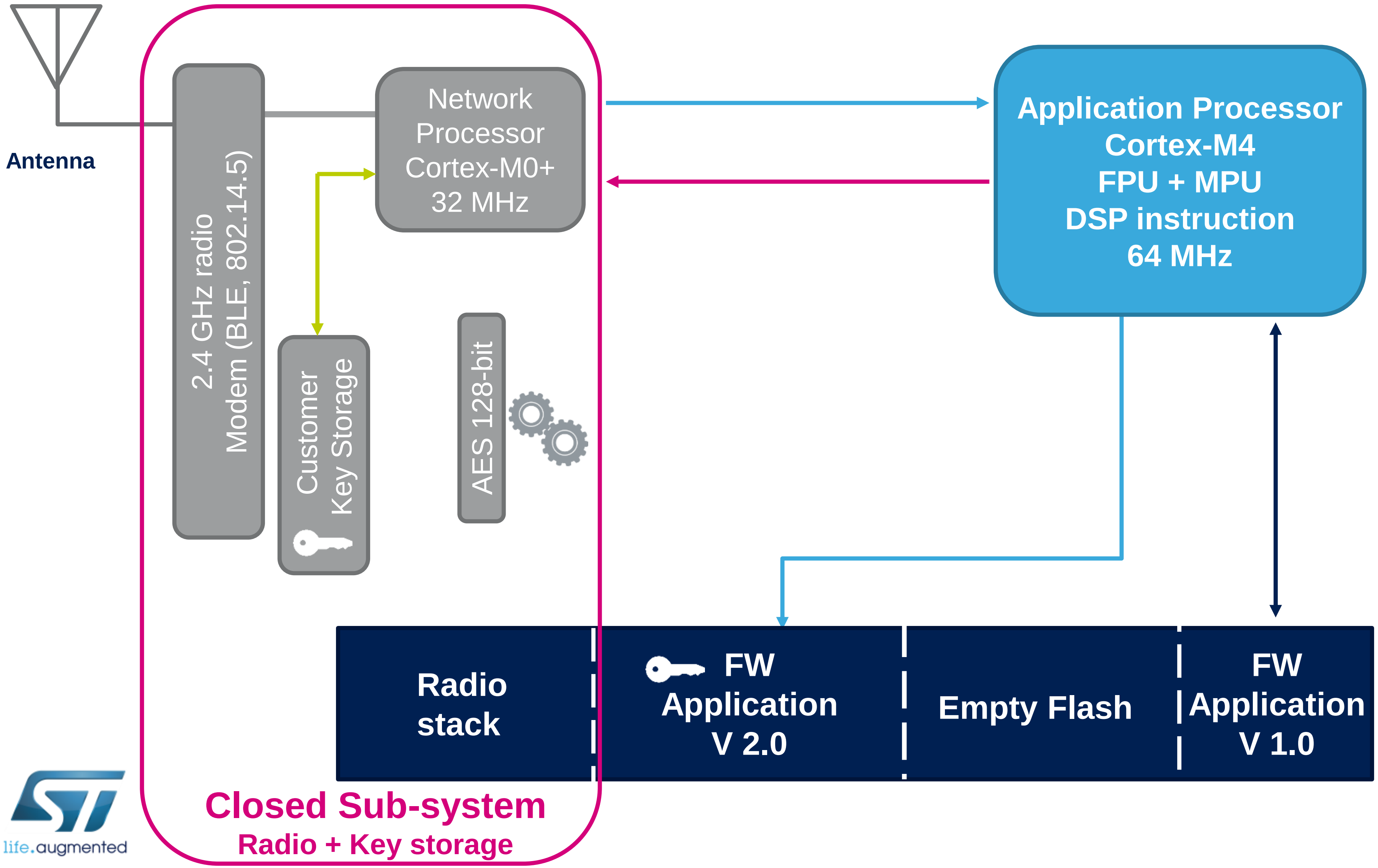
2 New FW detected
Update is launched

OTA Update Scenario

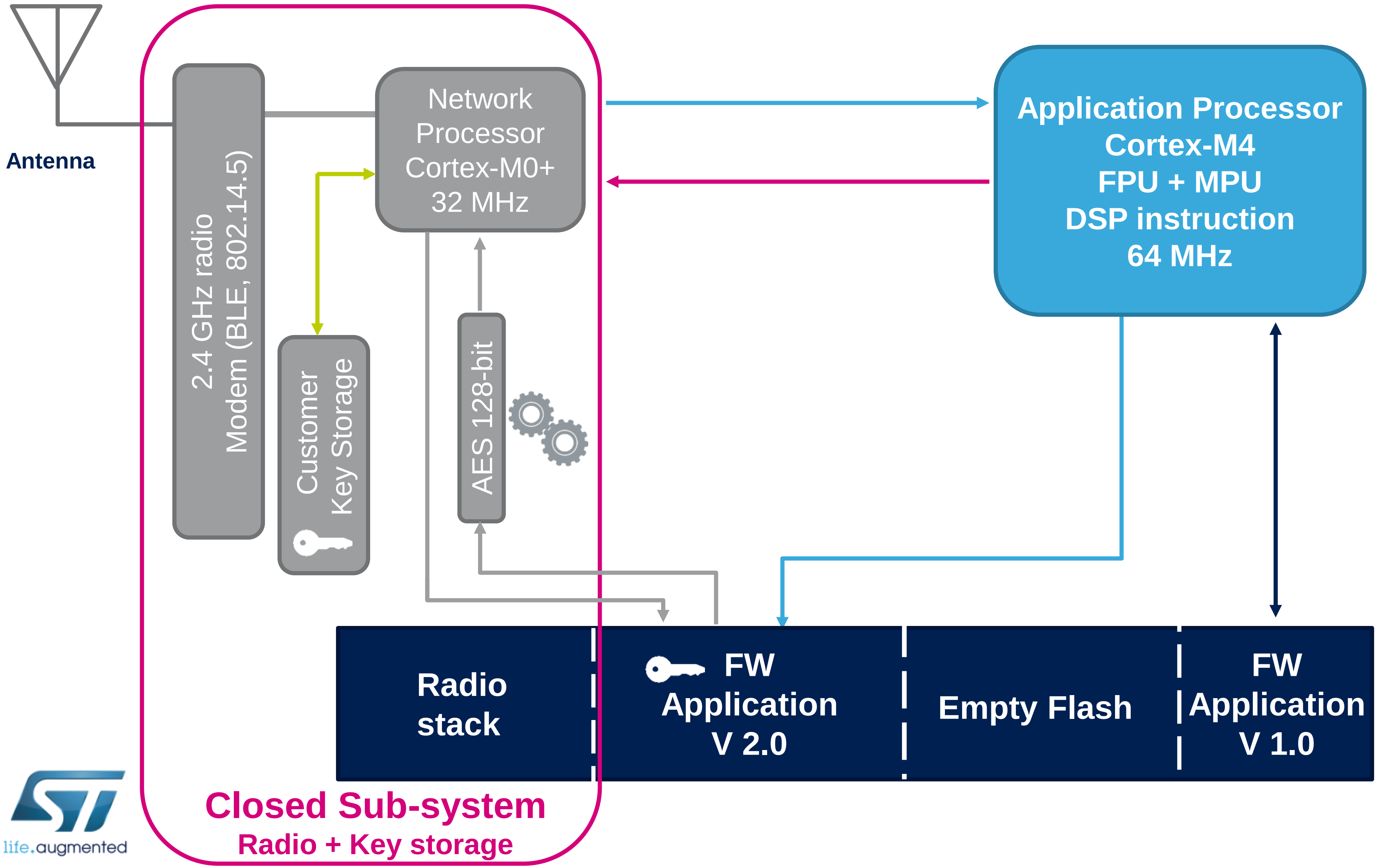


1. New FW package received
2. New FW detected
   Update is launched
3. App Processor send New FW package signature and encryption key for authentication

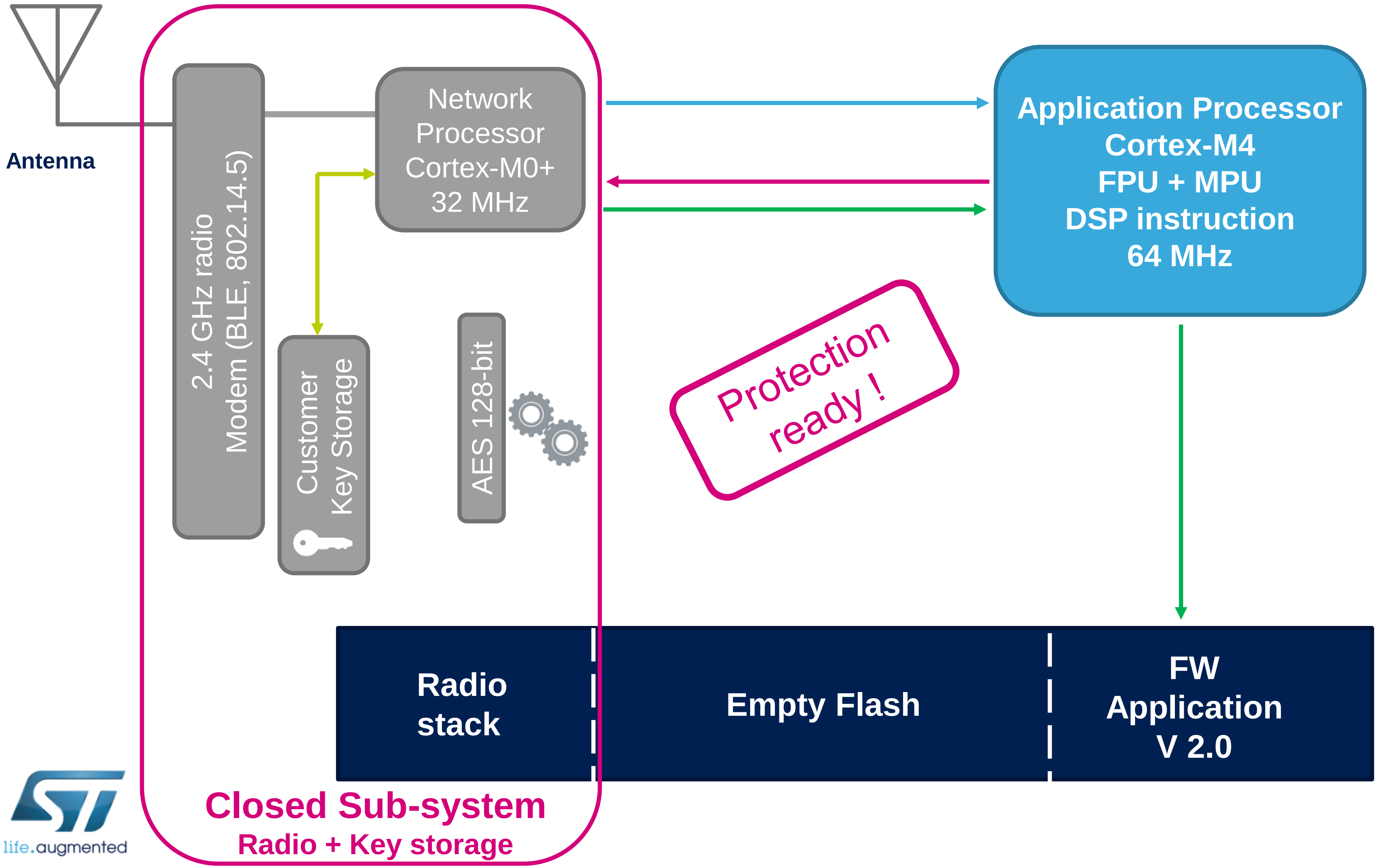
OTA Update Scenario



1. New FW package received
2. New FW detected
Update is launched
3. App Processor send New FW package signature and encryption key for authentication
4. Authentication signature matches preprogrammed key if not, the process is aborted and device resets



1. New FW package received
2. New FW detected
   Update is launched
3. App Processor send New FW package signature and encryption key for authentication
4. Authentication signature matches preprogrammed key if not, the process is aborted and device resets
5. New FW package is decrypted with proprietary Key.



1. New FW package received
2. New FW detected
   Update is launched
3. App Processor send New FW package signature and encryption key for authentication
4. Authentication signature matches preprogrammed key if not, the process is aborted and device resets
5. New FW package is decrypted with proprietary Key.
6. New Firmware replaces older firmware device resets.

AN5156 is a deep-dive into many security topics, some common and some WB-specific

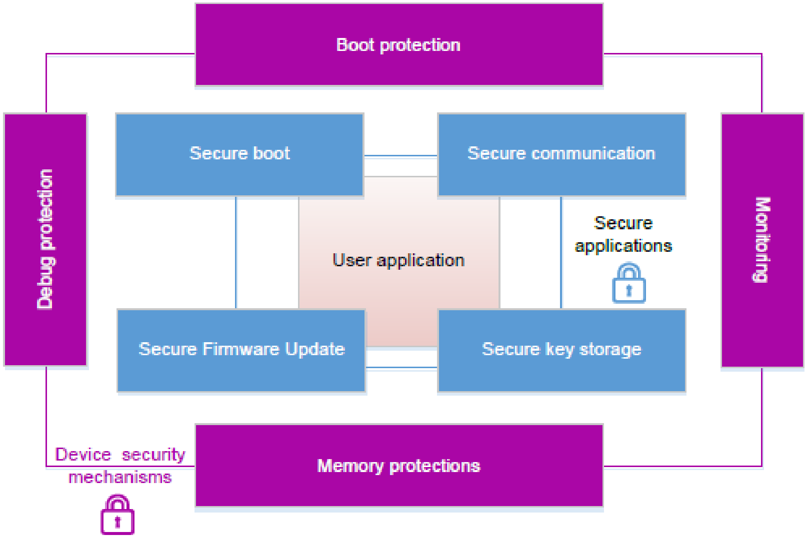


Table 4. Attacks types and costs

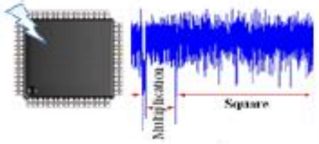
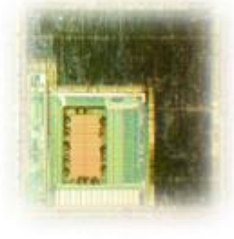
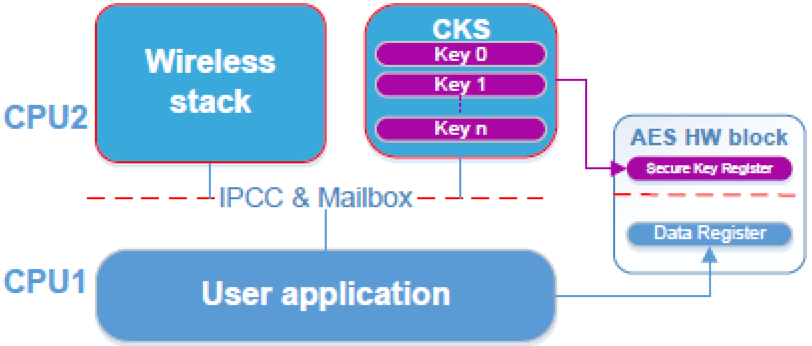
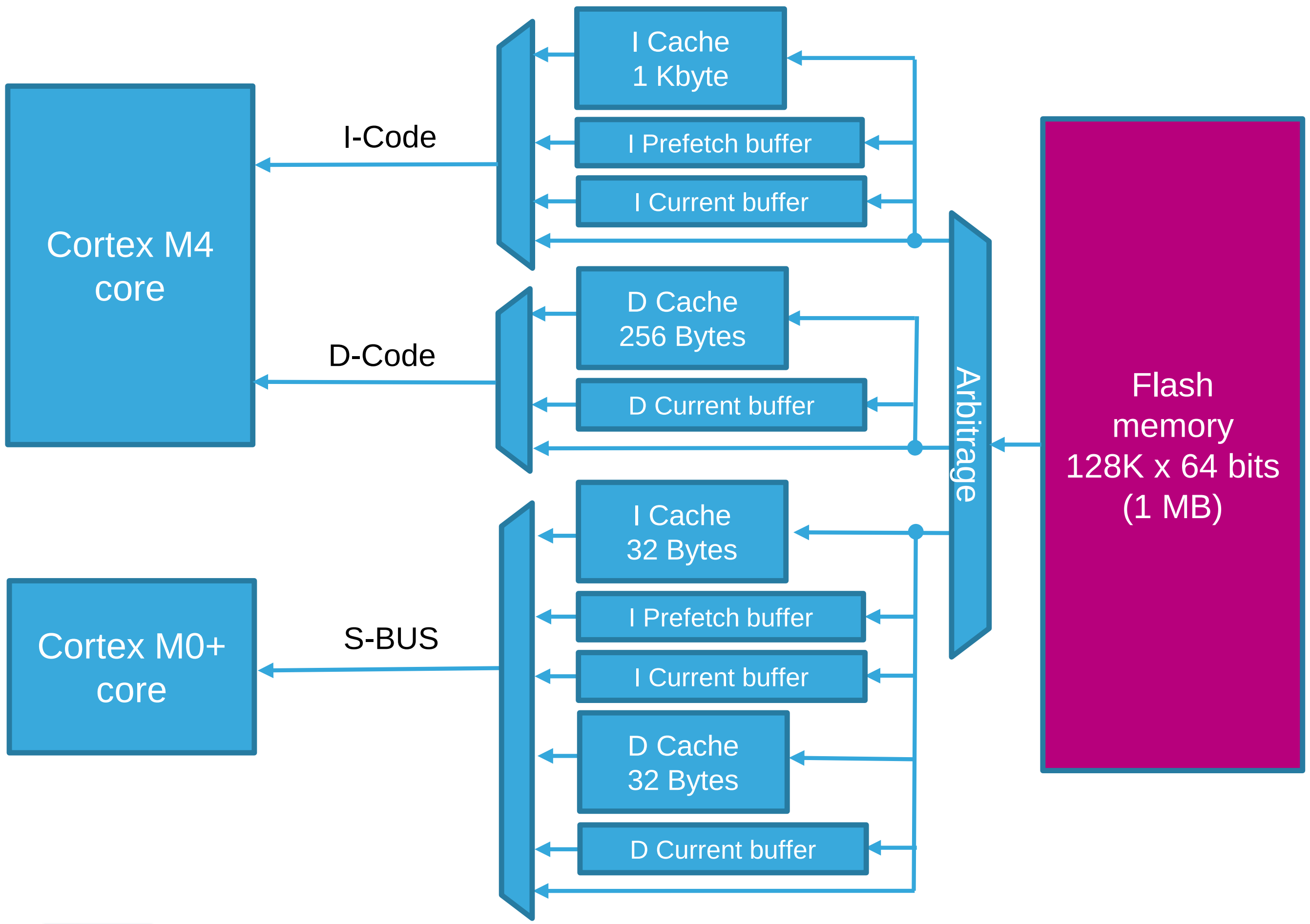
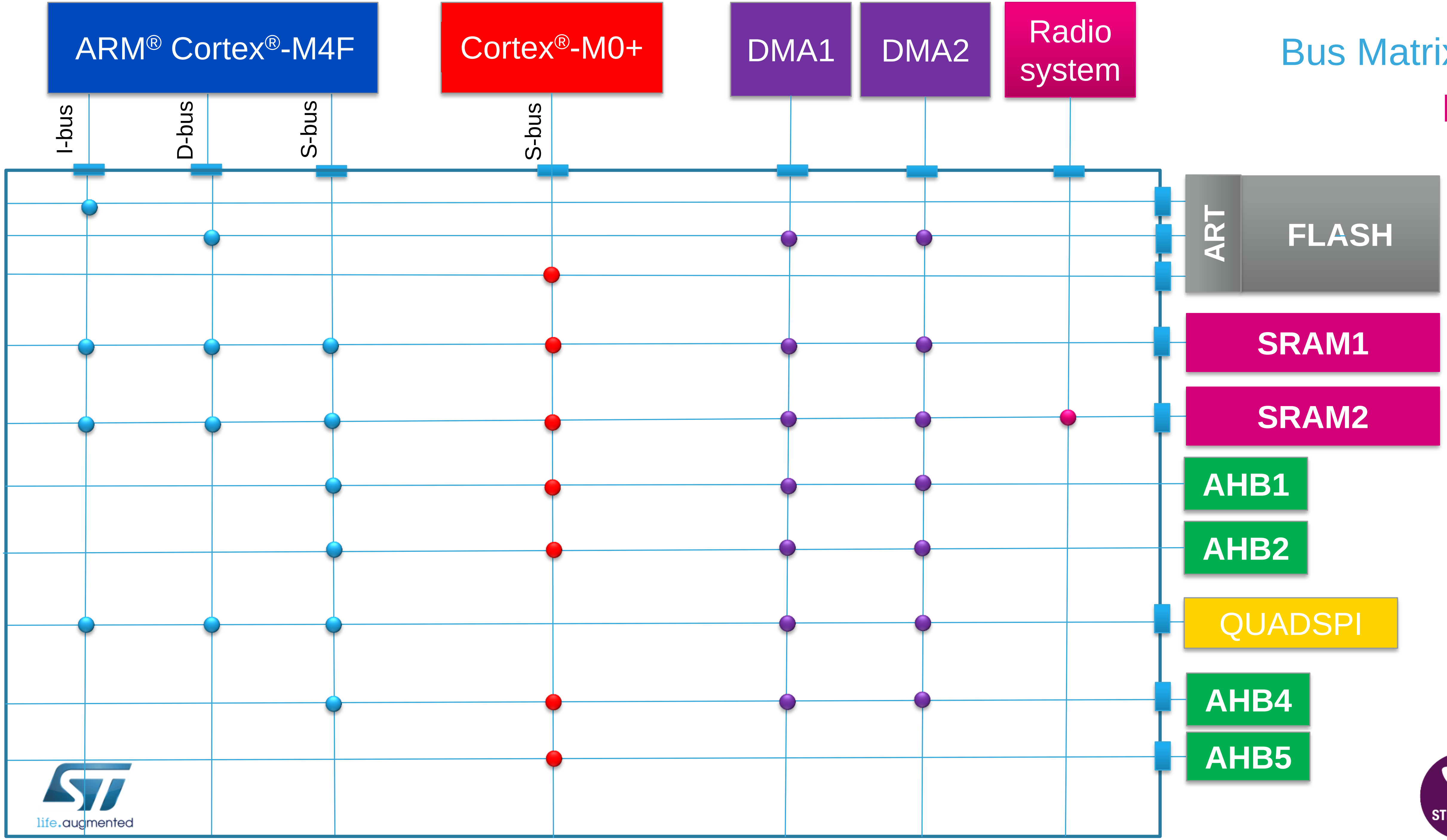
Attacks types	Software	Hardware non–invasive	Hardware invasive
-			
Scope	Remote or local	Local board and device level	Local device level
Technics	Software bugs Protocol weaknesses Trojan horse Eavesdropping...	Debug port Power Glitches Fault injection Side-channels analysis...	Probing Laser FIB Reverse engineering...
Cost/ expertise	From very low to high depending on the security failure targeted	Quite low cost. Need only moderately sophisticated equipment and knowledge to implement	Very expensive. Need dedicated/ heavy equipment and very specific skills
Objectives	Access to confidential assets (code and data). Usurpation Denial of service	Access to secret data or device internal behavior (algorithm)	Reverse engineering of the device (silicon intellectual property) Access to hidden hardware and software secrets (Flash access)

Figure 13. Dual-core architecture with CKS service

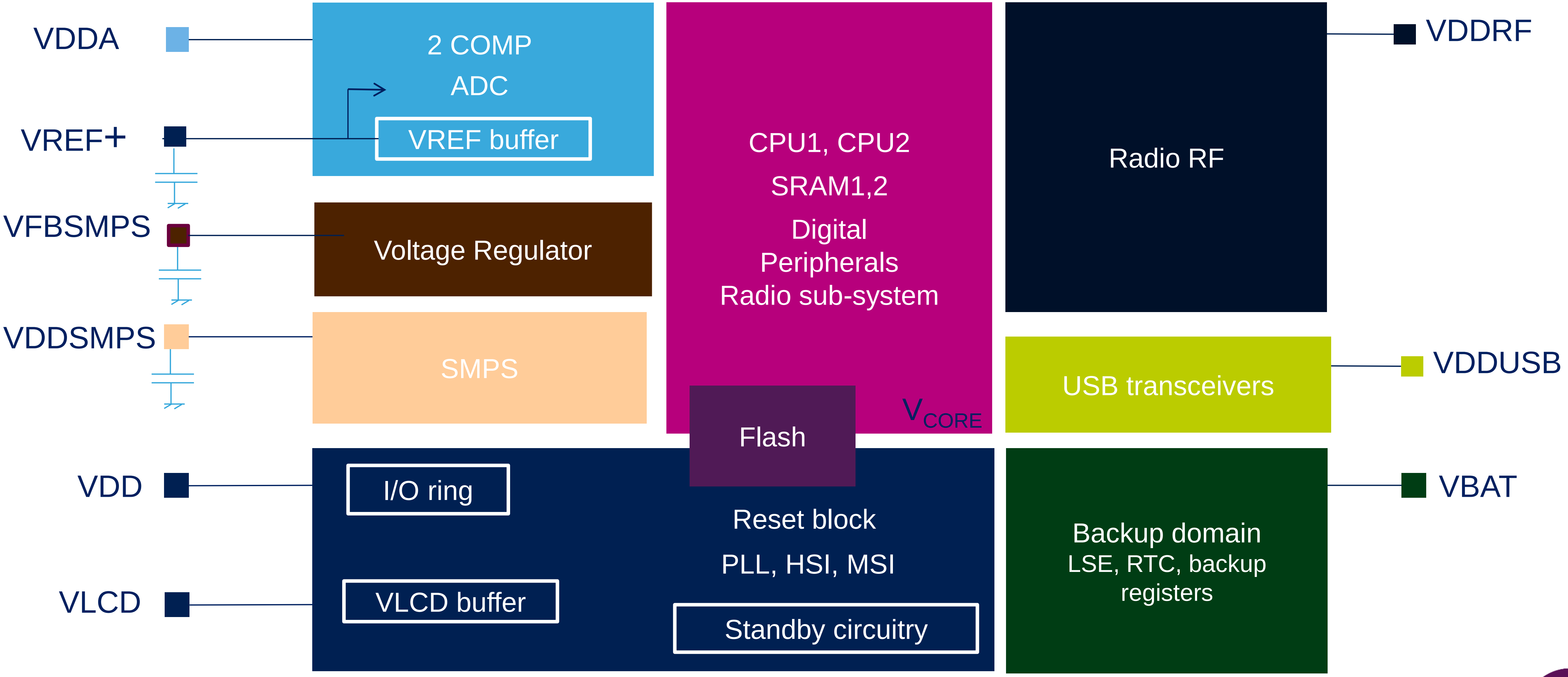


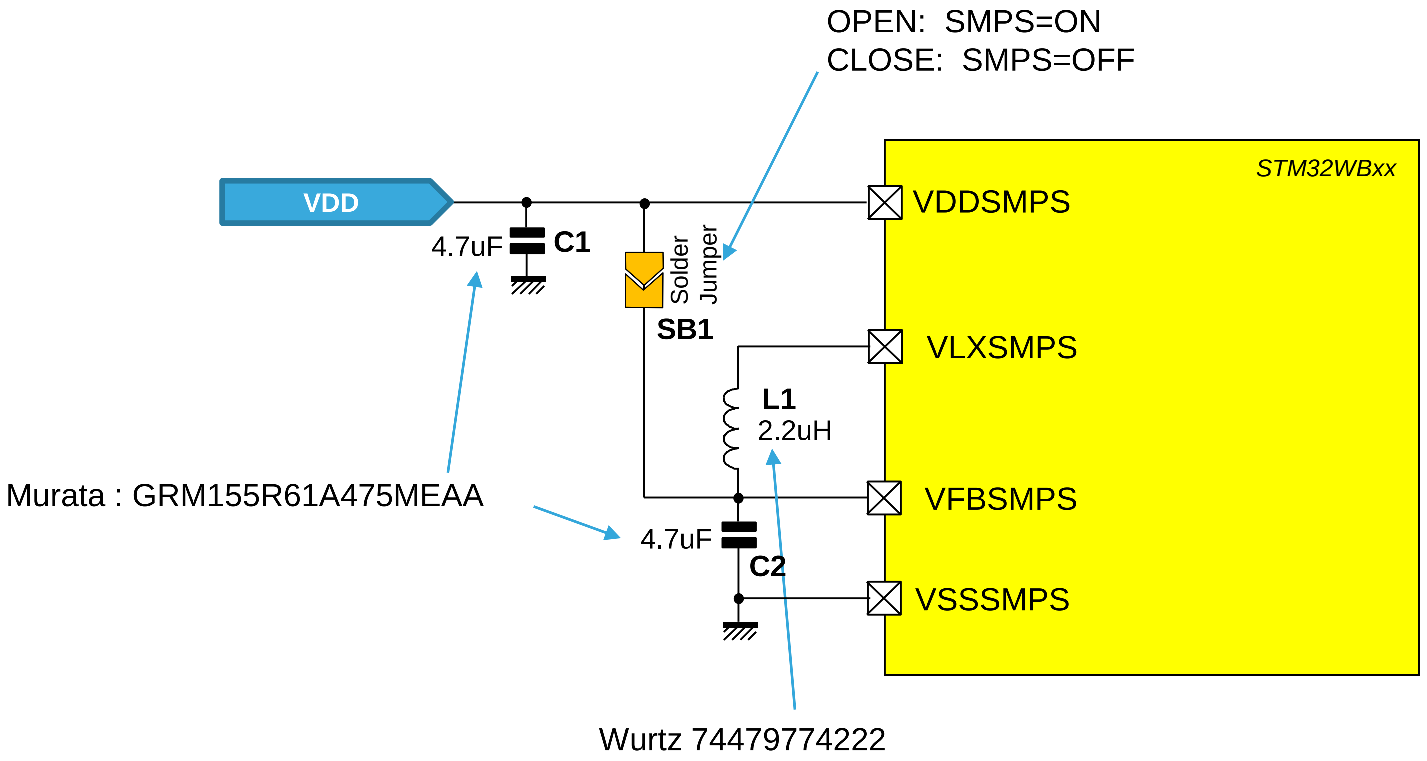


- **Cortex-M4**
  - **Instruction cache** = 32 lines of 4x64 bits
  - **Data cache** = 8 lines of 4x64 bits
  - **Pre-fetch buffer**
- **Cortex-M0+**
  - **Instruction cache** = 4 lines of 1x64 bits
  - **Data cache** = 4 lines of 1x64 bits
  - **Pre-fetch buffer**


ARM® Cortex®-M4F
Cortex®-M0+
DMA1
DMA2
Radio system
I-bus
D-bus
S-bus
S-bus
ART
FLASH
SRAM1
SRAM2
AHB1
AHB2
QUADSPI
AHB4
AHB5
life.augmented
STM32 WB

Bus Matrix


VDDA
VREF+
VFBSMPS
VDDSMPS
VDD
VLCD
2 COMP
ADC
VREF buffer
Voltage Regulator
SMPS
CPU1, CPU2
SRAM1,2
Digital
Peripherals
Radio sub-system
VCORE
Flash
I/O ring
Reset block
PLL, HSI, MSI
VLCD buffer
Standby circuitry
Radio RF
VDDRF
USB transceivers
VDDUSB
Backup domain
LSE, RTC, backup
registers
VBAT



8MHz SMPS configuration

For 4MHZ SMPS configuration change L1 = 4.7µH

FlexPowerControl

Wake-up time	Mode	Consumption
	RUN (Range1) at 64 MHz	117 (73) µA / MHz**
9 cycles	SLEEP at 64 MHz	41 µA / MHz
5 µs (20 µs)	STOP 2 (full retention)	1.8 µA / 2.2 µA*
14 µs (25 µs)	STANDBY + 32 KB RAM	320 nA / 600 nA*
14 µs (25 µs)	STANDBY	110 nA / 440 nA*
50 µs	SHUTDOWN	30 nA / 315 nA*
	VBAT	2 nA / 300 nA*

(..) SMPS mode

RF Operation

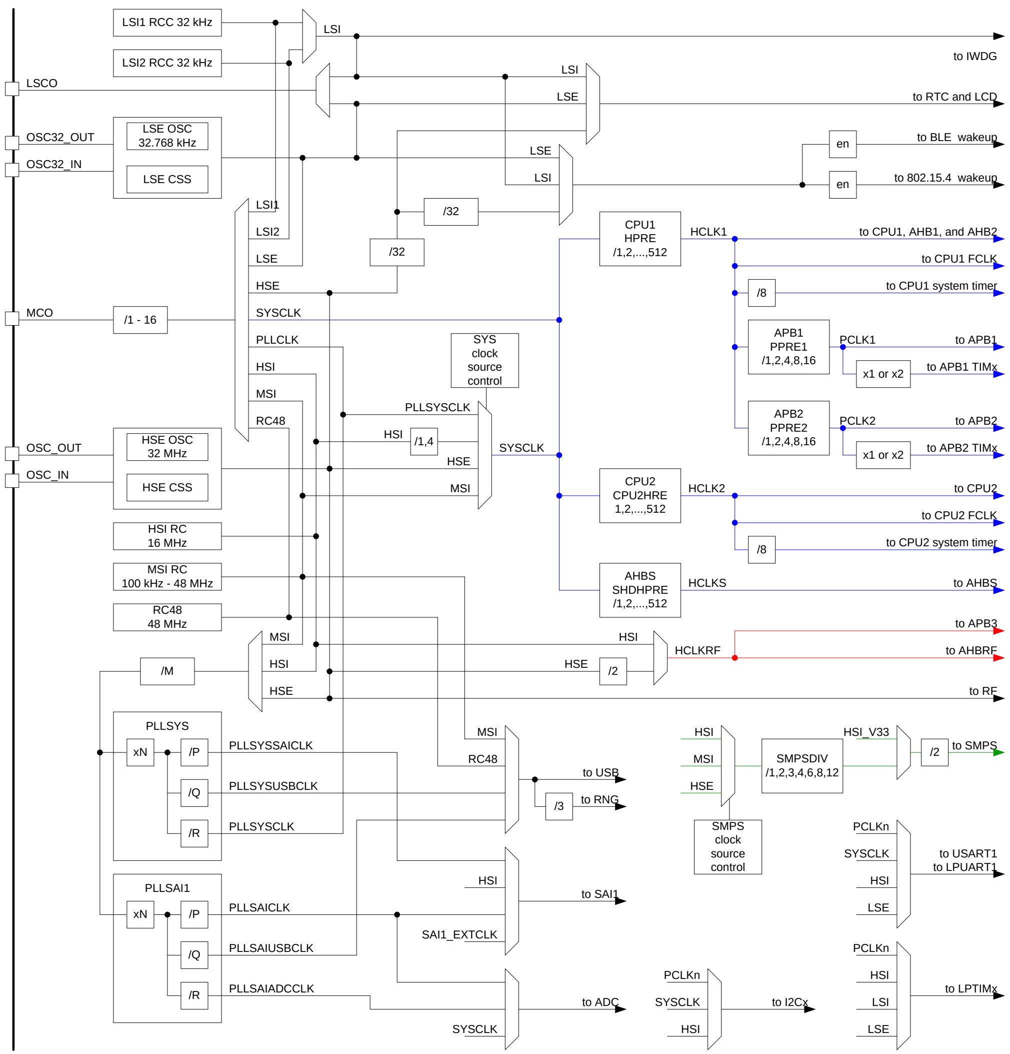
Typ @ VDD =1.8 V @ 25 °C

** with RTC*
*** from SRAM1*

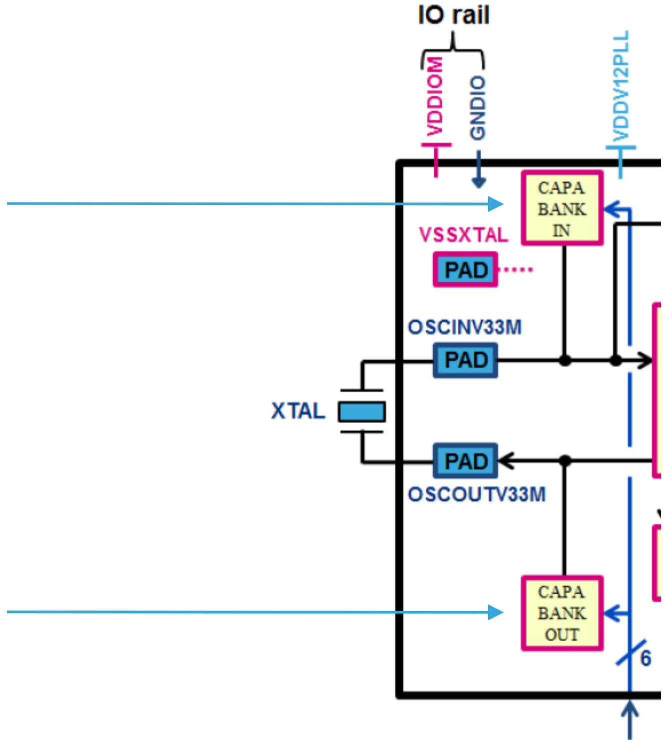
- High performance
  - ➔ CoreMark score = 215
- Outstanding power efficiency
  - ➔ ULPBbench score = 175

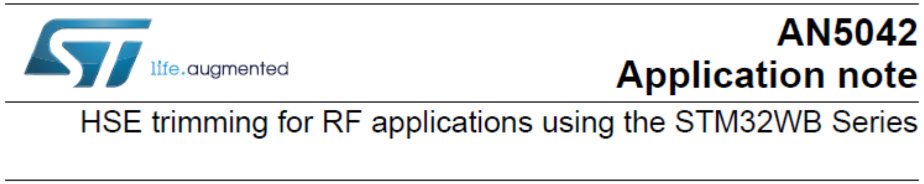
HSE (32MHz) required for radio operation

LSE (32.768KHz) required for most BLE applications

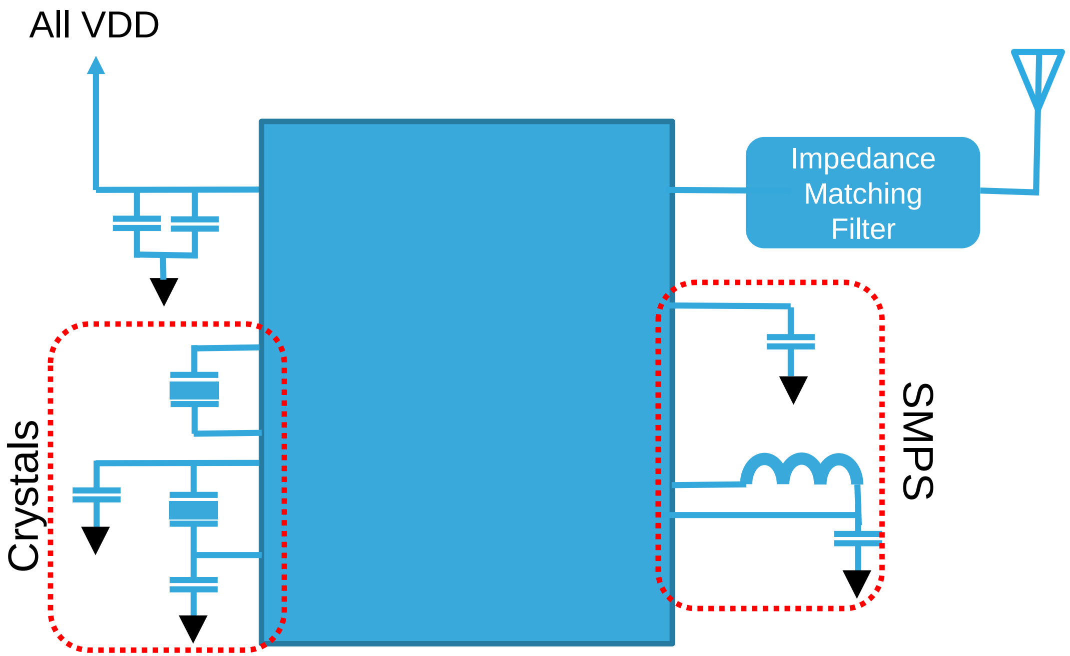


- BLE requires very accurate 32 MHz clock
- Frequency can vary
  - Manufacturing process variations
  - Crystal used
  - PCB design
- Integrated load capacitor bank
  - 64 values for fine tuning
  - MCO clock output pin used for measurement at factory test
  - Stored in OTP
- **No need for external capacitance**
- AN5042 provides details




AN5042
Application note
HSE trimming for RF applications using the STM32WB Series

- Embedded RF balun
- Single IPD from ST
- Simple SMPS circuit
- Integrated HSE crystal tuning caps
- Minimal passives needed
- Simple 2 layer PCB design



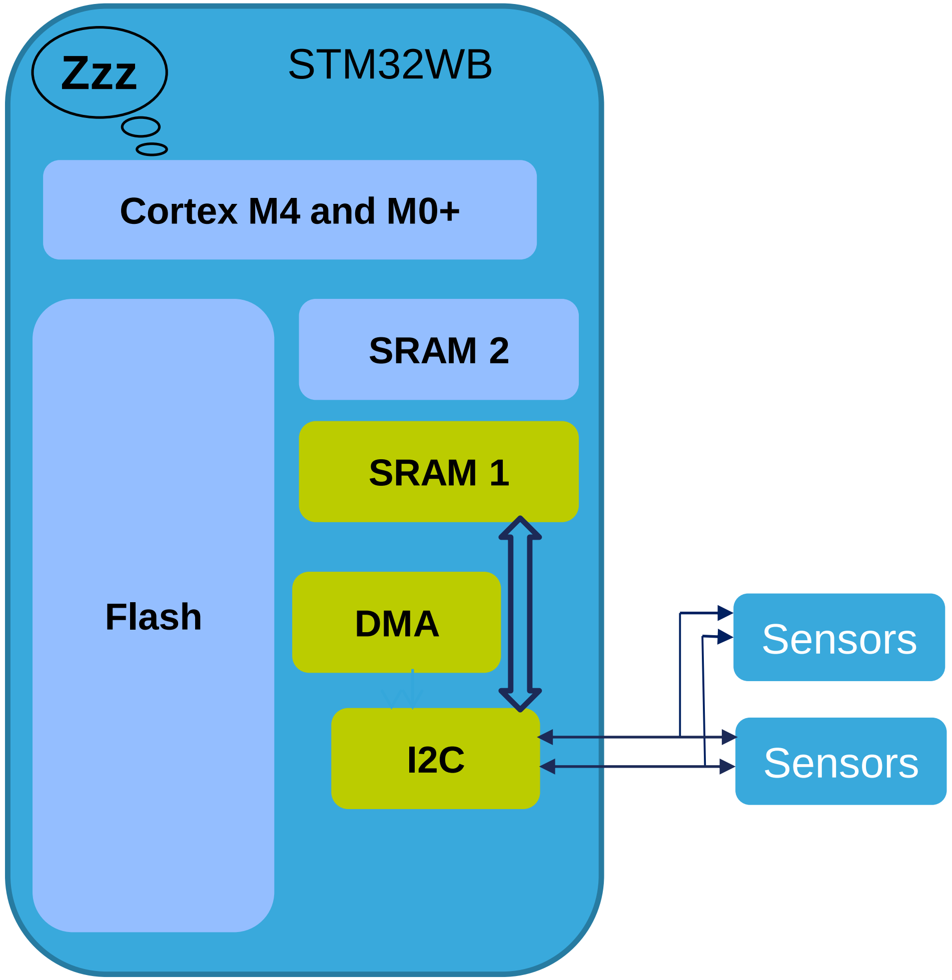
STM32WB55
Simplified Schematic Diagram

Peripheral + DMA + SRAM1

Flash in Power-down mode

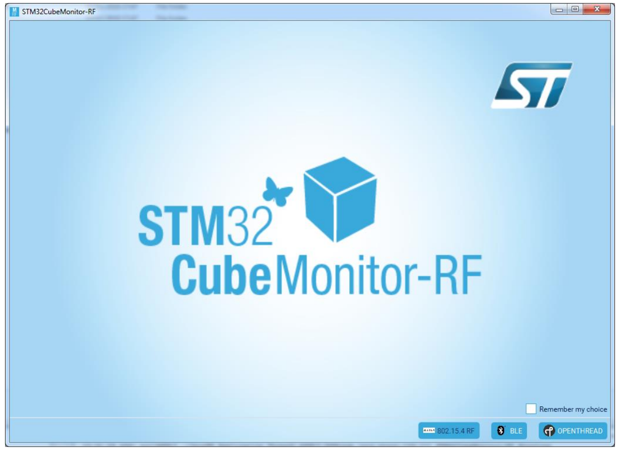
CortexM4 in Sleep mode

CortexM0+ in Sleep mode



Hands-On

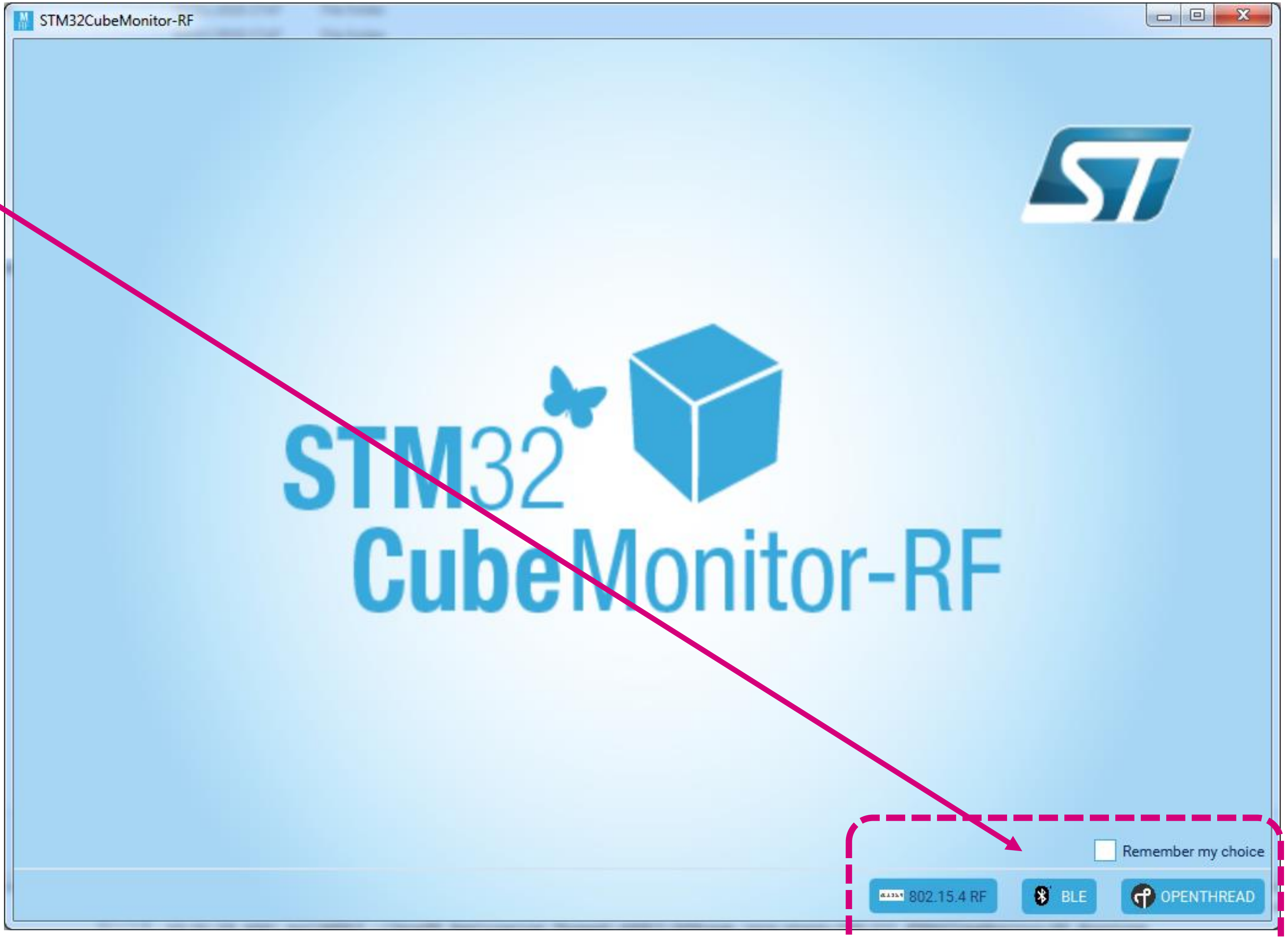
CubeMonitorRF



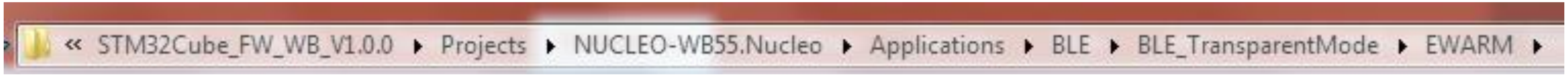
- BLE commands
- OpenThread commands
- BLE & 802.15.4 RF tests
- COM-port based

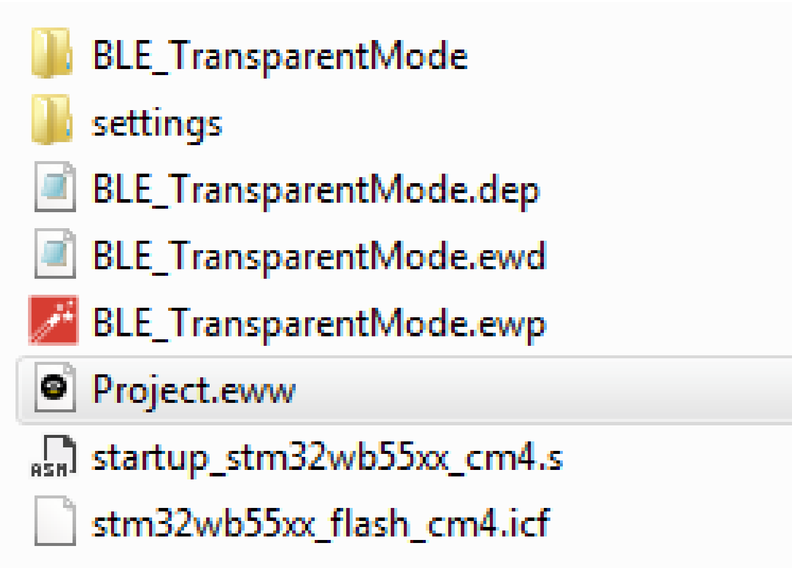
Mode selection

We will run in BLE mode

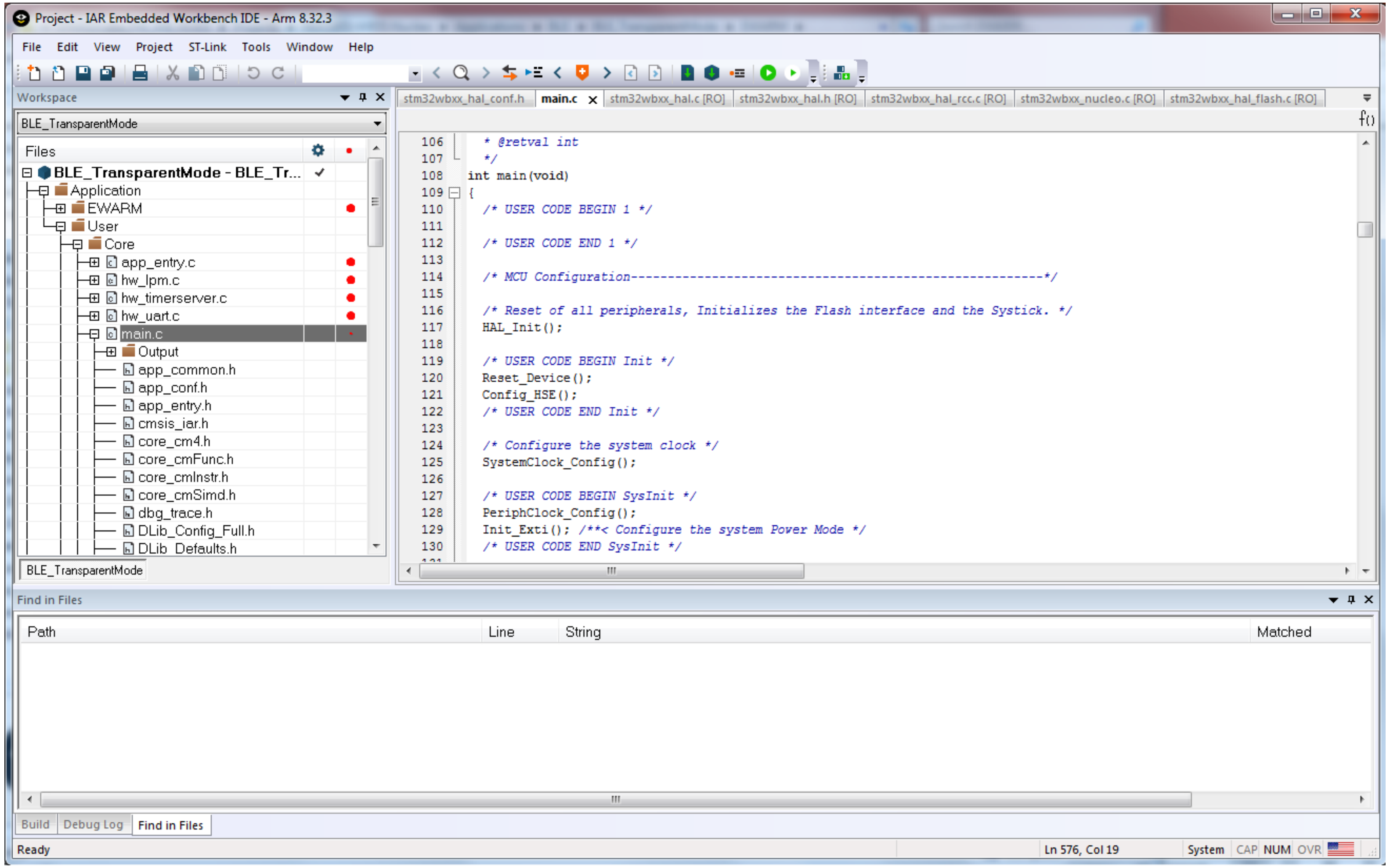


Open the **Transparent Mode** workspace





Build, Debug & Run on your Nucleo board

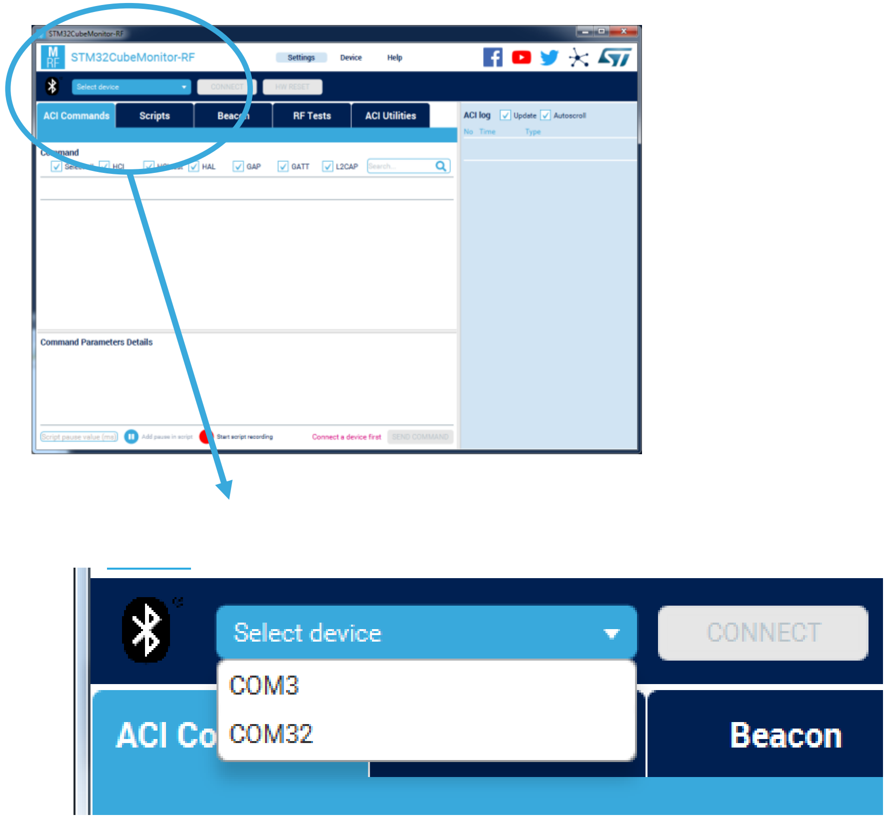


Connect to CubeMonitorRF

Program via USB debug port

Select device on relevant COM port

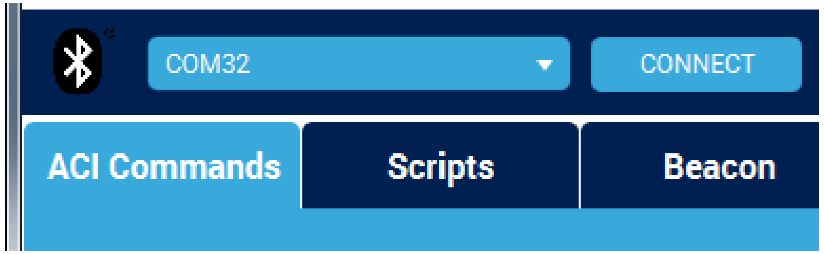
Connect to start communication



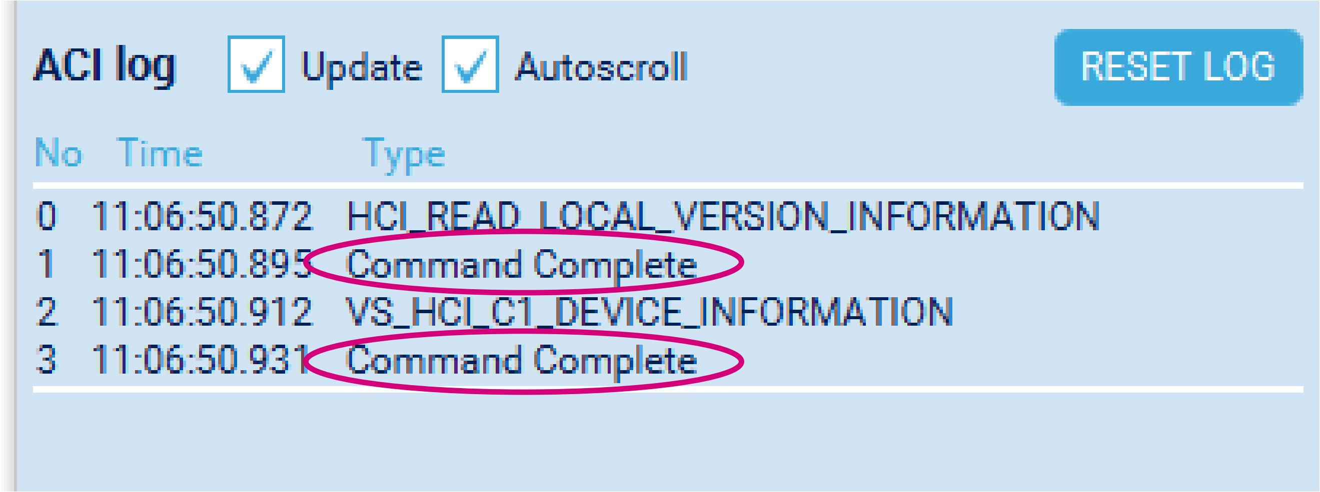
Program via USB debug port

Select device on relevant COM port

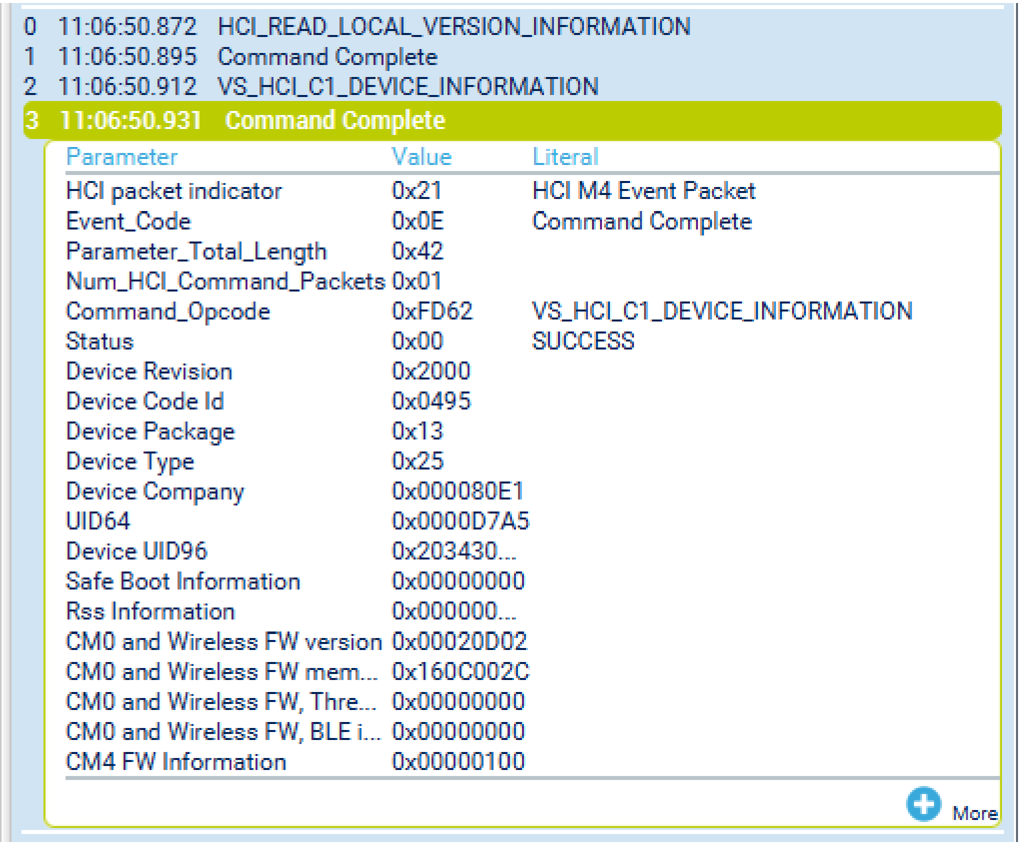
Connect to start communication



Command Complete signals successful communications



Click on the **Command Complete** line to get more information on the command sent



0	11:06:50.872	HCI_READ_LOCAL_VERSION_INFORMATION
1	11:06:50.895	Command Complete
2	11:06:50.912	VS_HCI_C1_DEVICE_INFORMATION
3	11:06:50.931	Command Complete

Parameter	Value	Literal
HCI packet indicator	0x21	HCI M4 Event Packet
Event_Code	0x0E	Command Complete
Parameter_Total_Length	0x42	
Num_HCI_Command_Packets	0x01	
Command_Opcode	0xFD62	VS_HCI_C1_DEVICE_INFORMATION
Status	0x00	SUCCESS
Device Revision	0x2000	
Device Code Id	0x0495	
Device Package	0x13	
Device Type	0x25	
Device Company	0x000080E1	
UID64	0x0000D7A5	
Device UID96	0x203430...	
Safe Boot Information	0x00000000	
Rss Information	0x000000...	
CM0 and Wireless FW version	0x00020D02	
CM0 and Wireless FW mem...	0x160C002C	
CM0 and Wireless FW, Thre...	0x00000000	
CM0 and Wireless FW, BLE i...	0x00000000	
CM4 FW Information	0x00000100	

More

Click on + More for additional detail

0 11:06:50.872 HCI_READ_LOCAL_VERSION_INFORMATION
1 11:06:50.895 Command Complete
2 11:06:50.912 VS_HCI_C1_DEVICE_INFORMATION
3 11:06:50.931 Command Complete

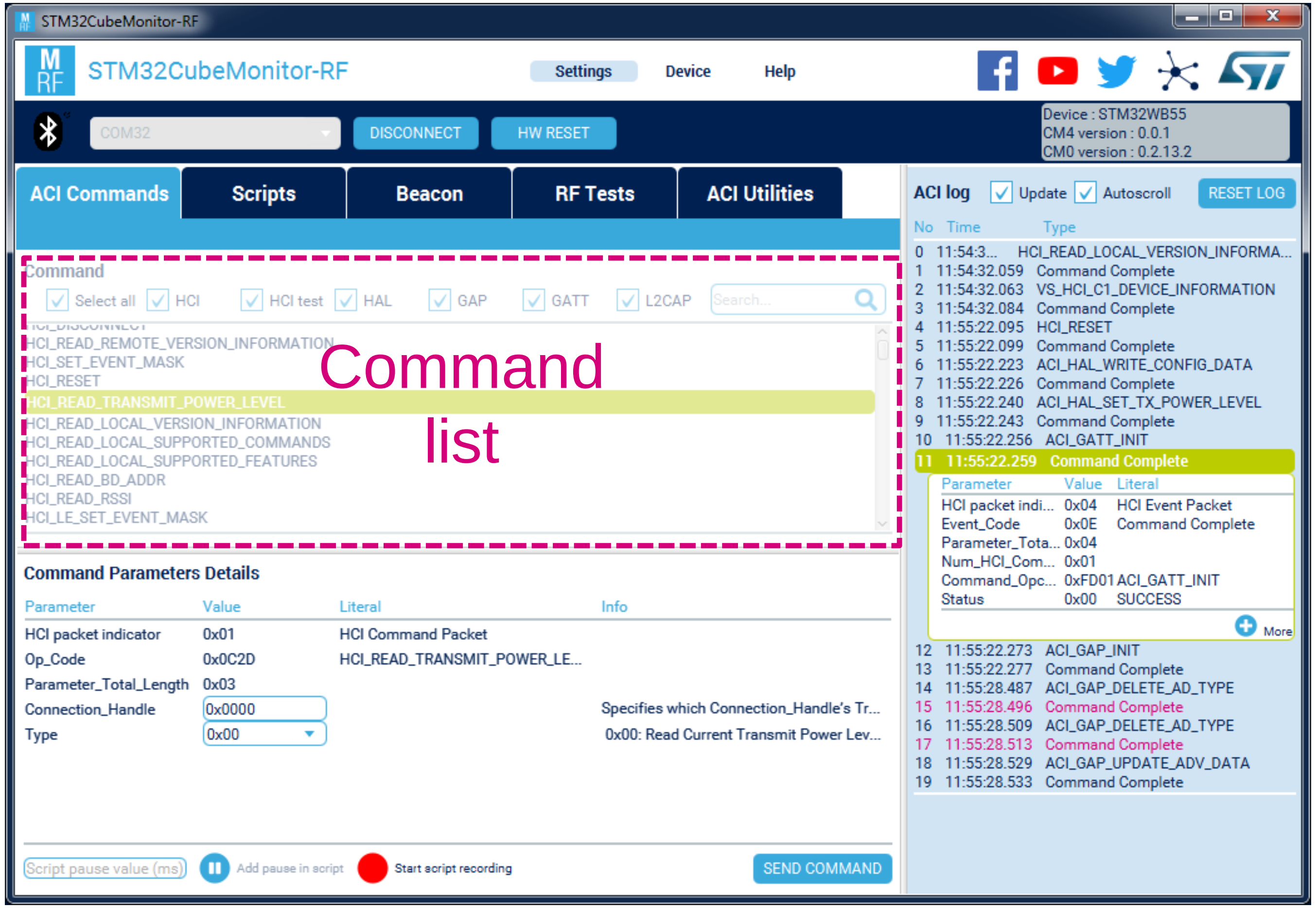
Parameter	Value	Literal
HCI packet indicator	0x21	HCI M4 Event Packet
Event_Code	0x0E	Command Complete
Parameter_Total_Length	0x42	
Num_HCI_Command_Packets	0x01	
Command_Opcode	0xFD62	VS_HCI_C1_DEVICE_INFORMATION
Status	0x00	SUCCESS
Device Revision	0x2000	
Device Code Id	0x0495	
Device Package	0x13	
Device Type	0x25	
Device Company	0x000080E1	
UID64	0x0000D7A5	
Device UID96	0x203430...	
Safe Boot Information	0x00000000	
Rss Information	0x000000...	
CM0 and Wireless FW version	0x00020D02	
CM0 and Wireless FW mem...	0x160C002C	
CM0 and Wireless FW, Thre...	0x00000000	
CM0 and Wireless FW, BLE i...	0x00000000	
CM4 FW Information	0x00000100	

More

Command Details

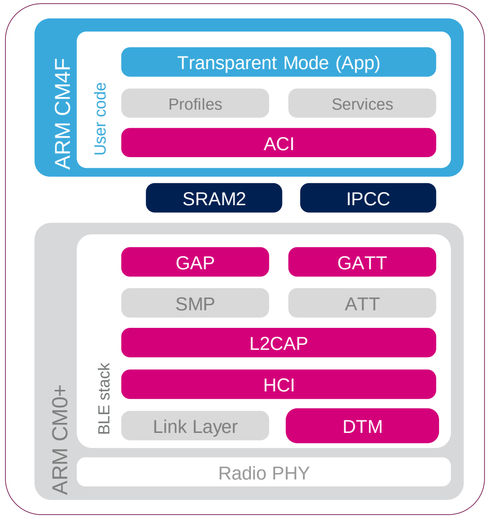
Parameter	Value	Literal	Info
HCI packet indicator	0x21	HCI M4 Event Packet	
Event_Code	0x0E	Command Complete	
Parameter_Total_Length	0x42		
Num_HCI_Command_Packets	0x01		The number of HCI command packets which are allowed tc
Command_Opcode	0xFD62	VS_HCI_C1_DEVICE_INFORMATION	Opcode of this command which caused this event.
Status	0x00	SUCCESS	Error code. See Core v4.1, Vol. 2, part D.
Device Revision	0x2000		Device revision information (From MCU)
Device Code Id	0x0495		Device Code identifier (From MCU)
Device Package	0x13		Device Package (from package data register)
Device Type	0x25		Device Type Id (from FLASH UID64)
Device Company	0x000080E1		Device Type Id (from FLASH UID64)
UID64	0x0000D7A5		UID64 (From flash)
Device UID96	0x203430523036500600390048		UID96 from Unique Device ID register
Safe Boot Information	0x00000000		Safe Boot Information (from SRAM2)
Rss Information	0x000000000000000000000000		Rss Information (from SRAM2)
CM0 and Wireless FW version	0x00020D02		CM0+ Wireless FW Information (from SRAM2)
CM0 and Wireless FW memory size	0x160C002C		CM0+ Wireless FW Information (from SRAM2)
CM0 and Wireless FW, Thread information	0x00000000		CM0+ Wireless FW Information (from SRAM2)
CM0 and Wireless FW, BLE information	0x00000000		CM0+ Wireless FW Information (from SRAM2)
CM4 FW Information	0x00000100		CM4 FW Information (Coded in user flash)

Lots of categories to choose and filter from



Command categories:

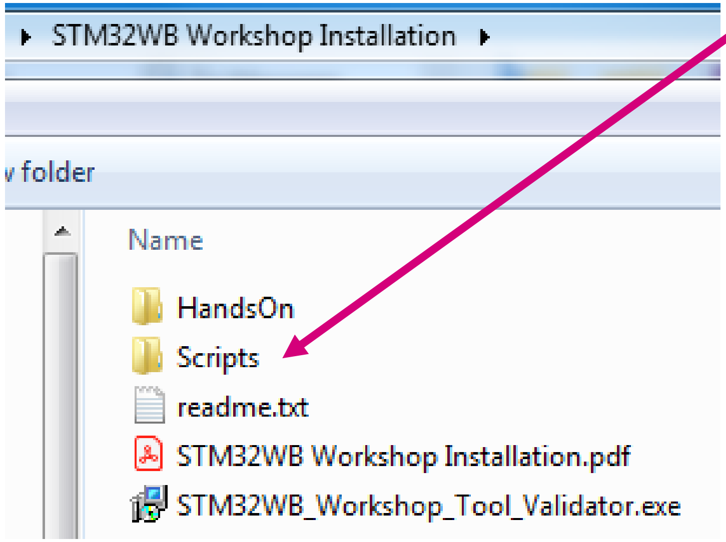
- HCI
- HCI test
- HAL
- GAP
- GATT
- L2CAP

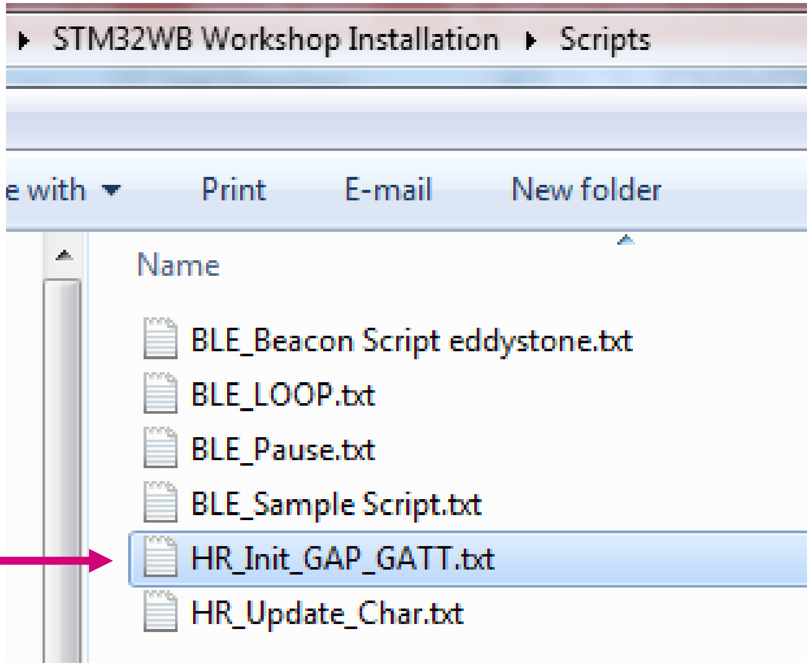


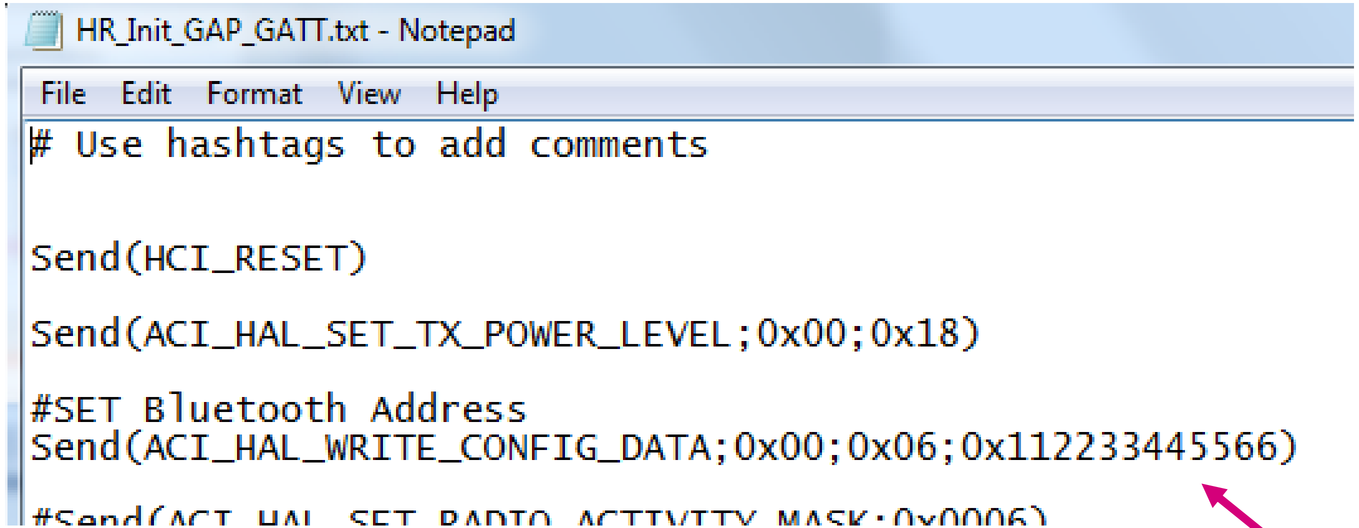
life.augmented

Open and Edit the HR_Init_GAP_GATT.txt Script file

(In your installation zip file, ***Scripts*** folder)






HR_Init_GAP_GATT.txt - Notepad

File Edit Format View Help

```
# Use hashtags to add comments


Send(HCI_RESET)

Send(ACI_HAL_SET_TX_POWER_LEVEL;0x00;0x18)

#SET Bluetooth Address
Send(ACI_HAL_WRITE_CONFIG_DATA;0x00;0x06;0x112233445566)

#Send(ACI_HAL_SET_RADIO_ACTIVITY_MASK;0x0006)
```

Set the Bluetooth Address

Modify this value as you wish

Send(ACI_HAL_WRITE_CONFIG_DATA;0x00;0x06;0x112233445566)

- Change the two characters of the Local Name with your Magic number (e.g. change 0x4257 to 0x3130 for magic number “01”.

```
Send(ACI_GAP_SET_DISCOVERABLE;0x00;0x0080;0x00A0;0x00;0x00;0x08;0x425732334D545309;0x03;0x180D02;0x0000;0x0000)

#0x42 57 32 33 4D 54 53 09
# 0x09 - Local name
# 0x53 - "S"
# 0x54 - "T"
# 0x4D - "M"
# 0x33 - "3"
# 0x32 - "2"
# 0x57 - "W"
# 0x42 - "B"
```

** Note Little-Endian byte ordering*

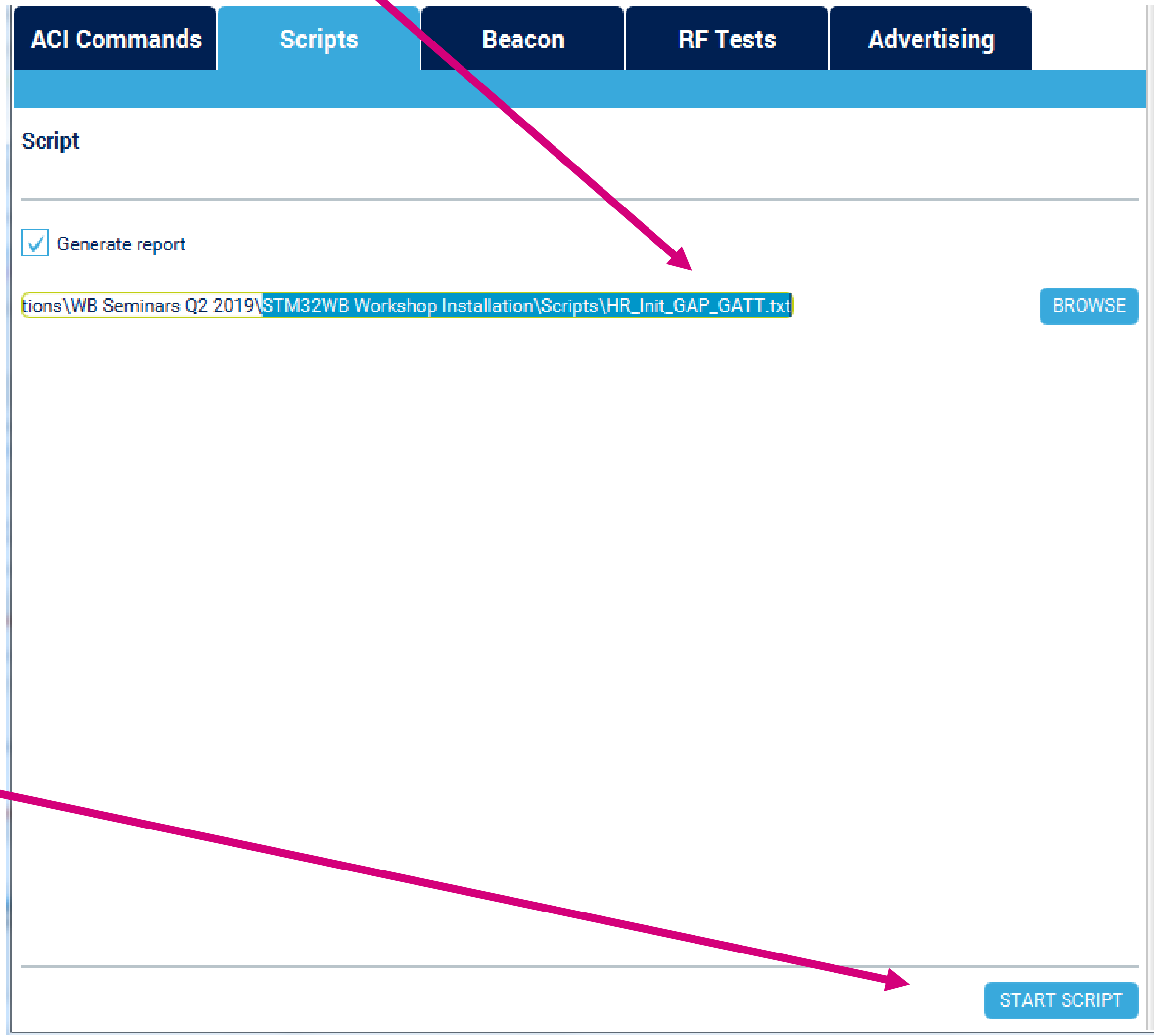
7 ASCII chars + 0x09 = 0x08. To add characters, also change the LENGTH parameter (x+1)

Hex	Char
30	0
31	1
32	2
33	3
34	4
35	5
36	6
37	7
38	8
39	9

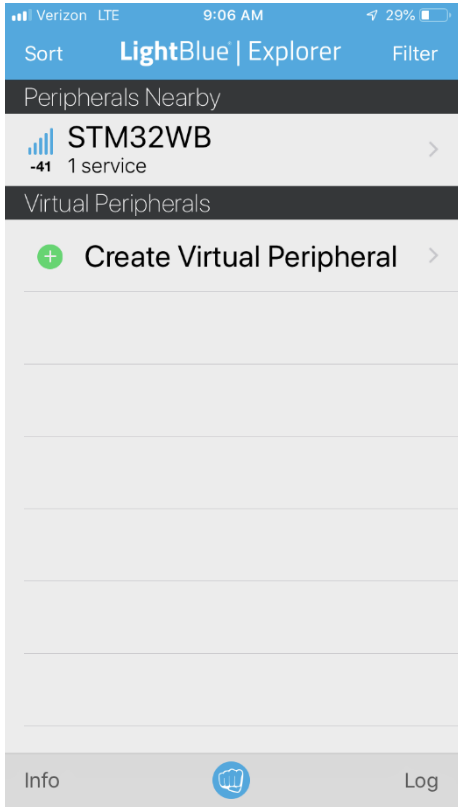
** ASCII Character Set for Magic numbers*

Save and Load your script

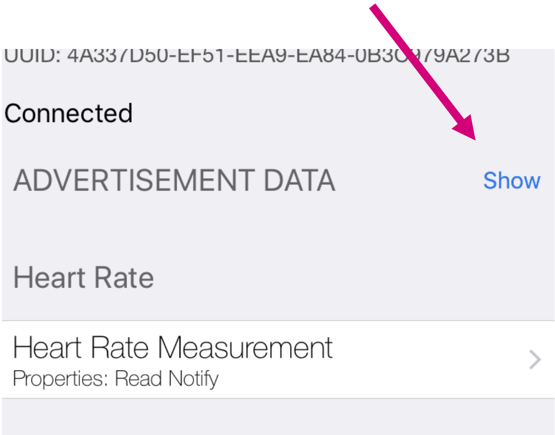
Start Script



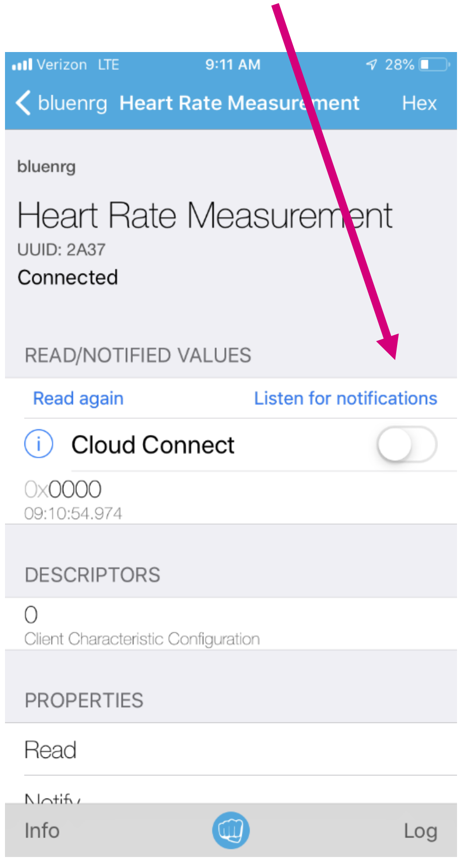
Find your device



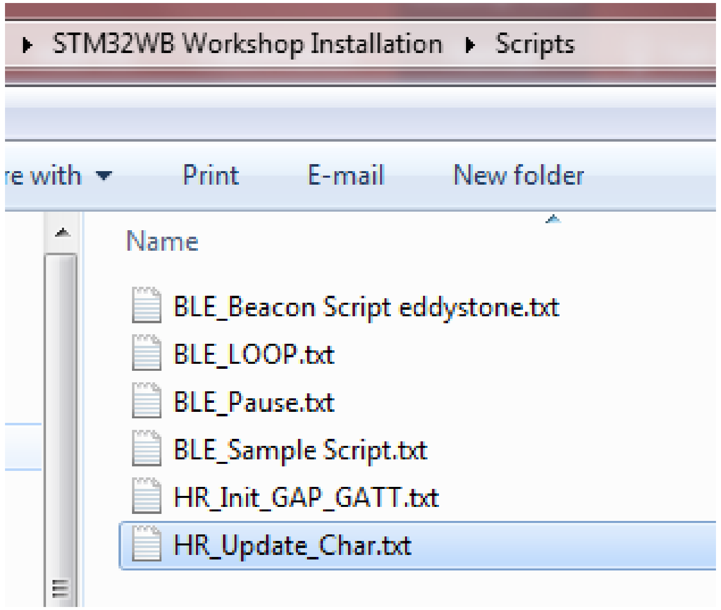
Show ADV data



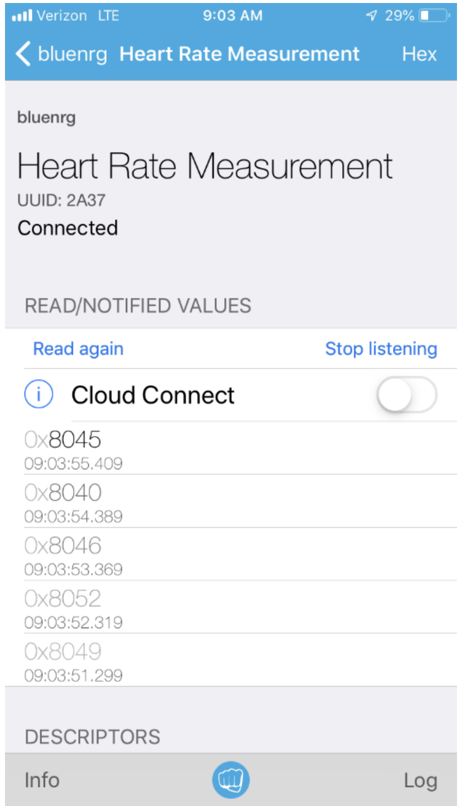
Enable Notifications



Now load the ***HR_Update_Char.txt*** script to send Notification Updates



Dummy Heart Rate Values are sent



Application Note AN5270 describes the ACI/HCI commands available

- via CubeMonitorRF
- via Application API's

2.3.2 ACI_HAL_WRITE_CONFIG_DATA

Description

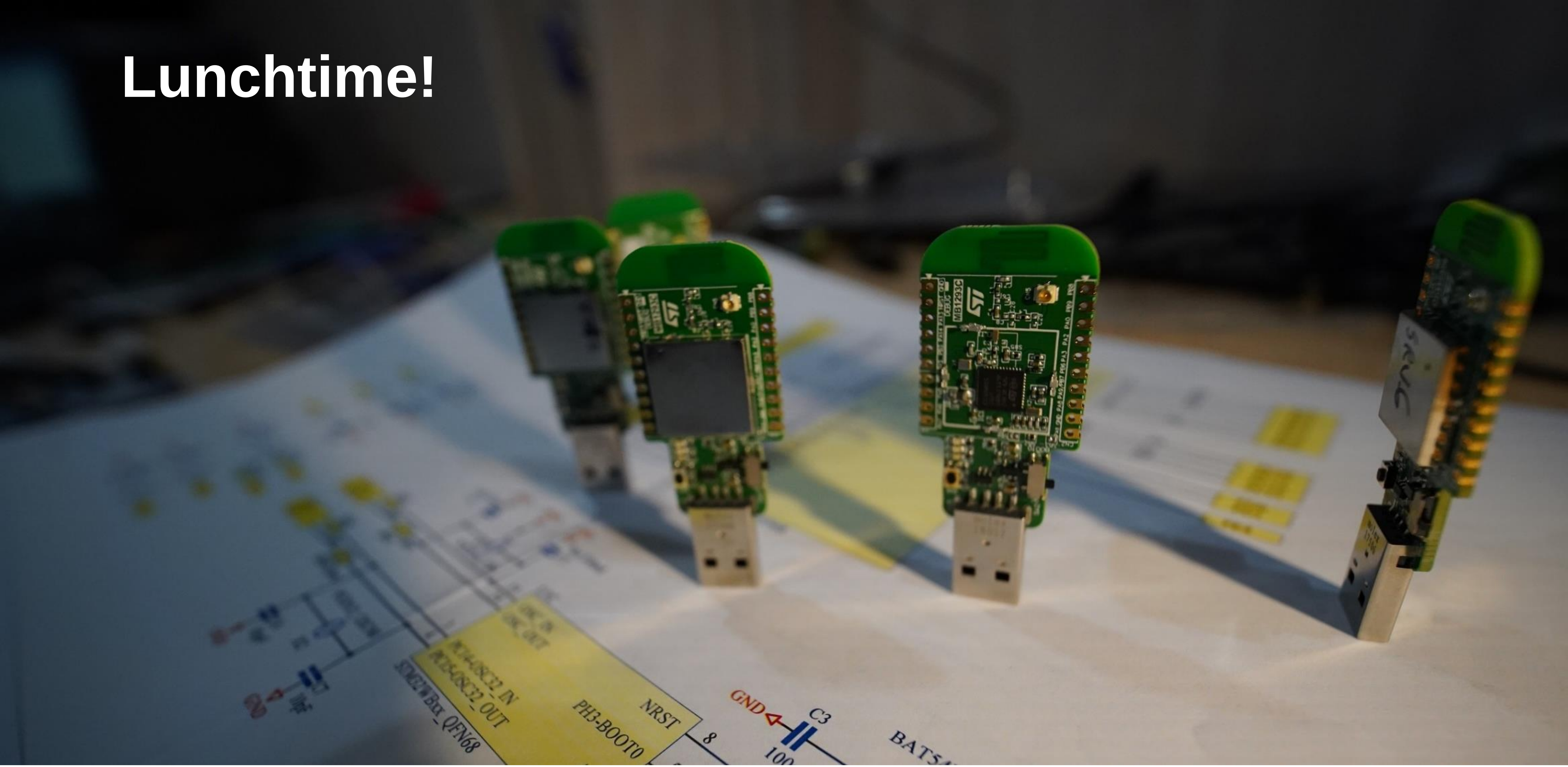
This command writes a value to a low level configure data structure. It is useful to setup directly some low level parameters for the system in the runtime.

Input parameters

Table 108. **Input parameters**

Parameter	Size	Description	Possible values
Offset	1	Offset of the element in the configuration data structure which has to be written. The valid offsets are: • 0x00: Bluetooth public address, value length to be written: 6 bytes • 0x06: DIV used to derive CSRK, value length to be written: 2 bytes • 0x08: Encryption root key used to derive LTK and CSRK, value length to be written: 16 bytes • 0x18: Identity root key used to derive LTK and CSRK, value length to be written: 16 bytes • 0x2C: Link layer without host (for certification purposes), Value length to be written: 1 byte • 0x2E: Static random address: 6 bytes • 0x2F: Disable watchdog (1=disable, 0=enable), value length to be written: 1 byte	• 0x00: CONFIG_DATA_PUBADDR_OFFSET • 0x06: CONFIG_DATA_DIV_OFFSET • 0x08: CONFIG_DATA_ER_OFFSET • 0x18: CONFIG_DATA_IR_OFFSET • 0x2C: LL_WITHOUT_HOST • 0x2E: CONFIG_DATA_RANDOM_ADDRESS_WR • 0x2F: CONFIG_DATA_WATCHDOG_DISABLE
Length	1	Length of data to be written	-
Value	Length	Data to be written	-

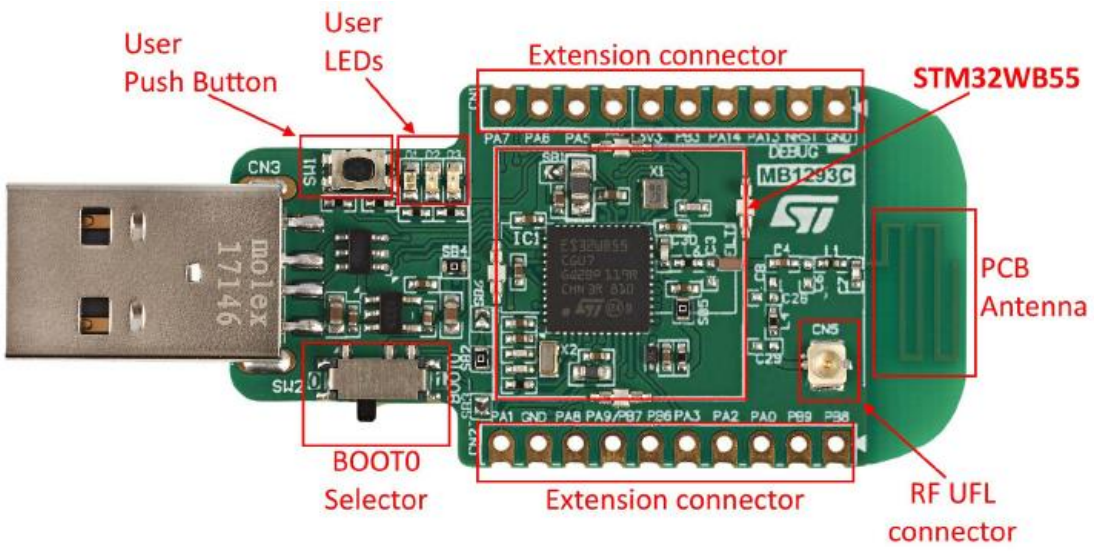
Lunchtime!


NRST
PH3-BOOT0
PC14-OSC32_IN
PC15-OSC32_OUT
STM32WBxx_QFN68
GND
C3

The USB Dongle is quite useful as the CubeMonitorRF sniffer

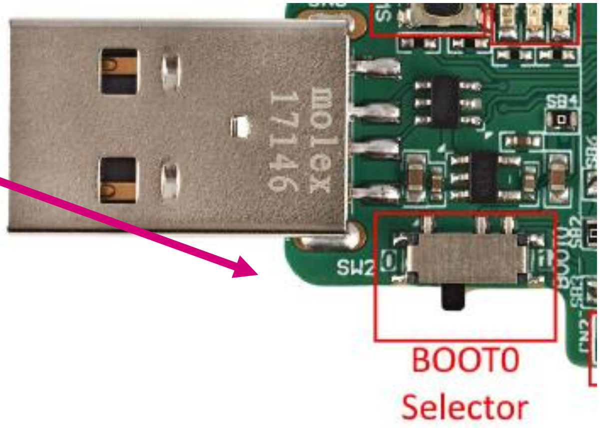
This project uses the USB CDC class directly (not the STLINK VCOM port) to parse commands

Although there is no STLINK on board, the USB bootloader can be invoked via **BOOT0 switch** & CubeProgrammer, and the binary can be programmed



Dongle config

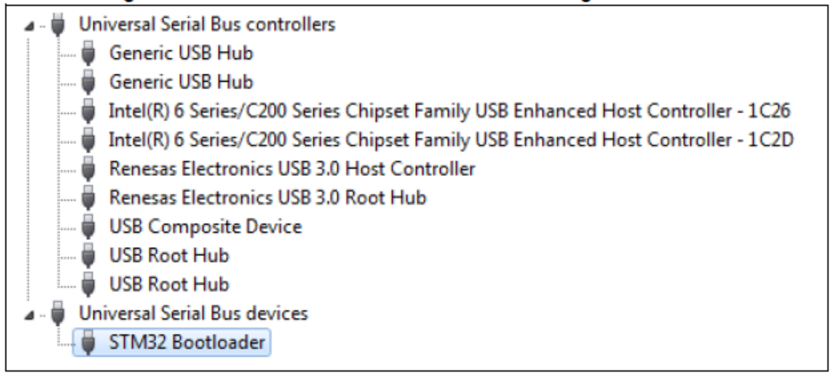
- Move Dongle Switch to Bootloader mode
- Plug in Dongle



Bootloader active to the right

- Ensure the driver has enumerated **“STM32 Bootloader”**

Figure 4. STM32 DFU device with STM32CubeProgrammer driver



life.augmented

Chapter 1.2.4 details the DFU driver install / update procedure

Old or Native MS drivers must be replaced to properly access the bootloader

Figure 2. Deleting the old driver software

Old Driver

Figure 3. STM32 DFU device with DfuSe driver

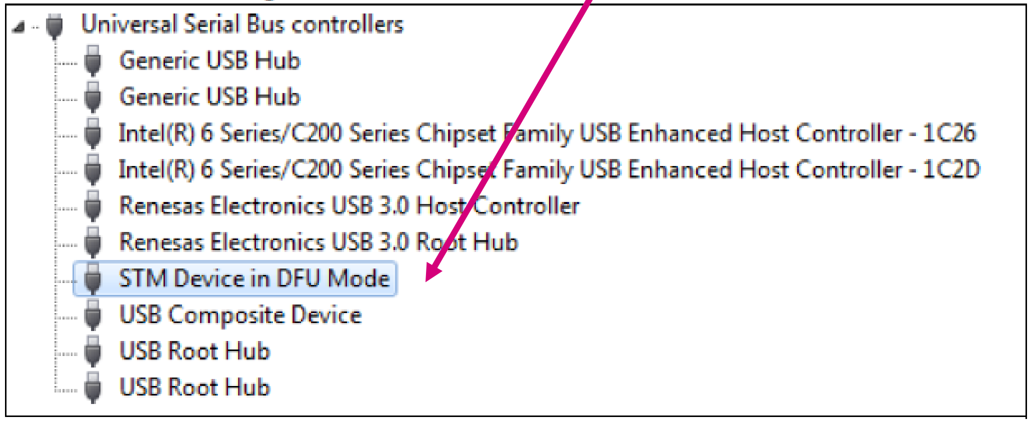
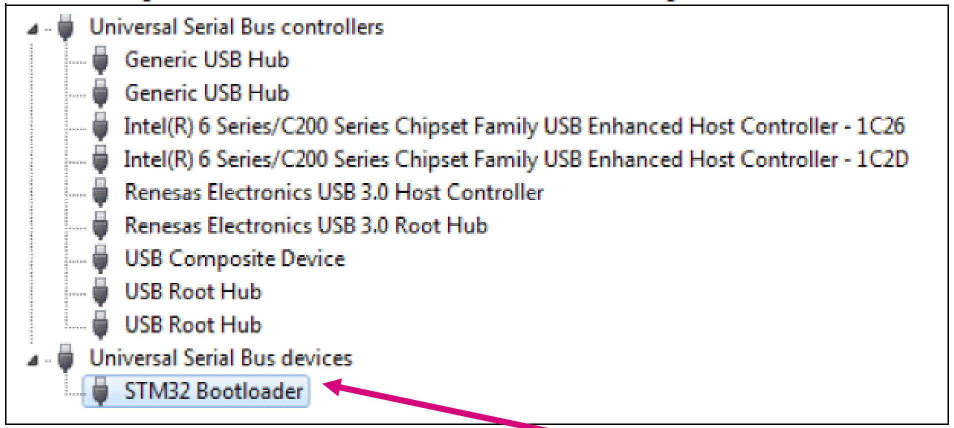
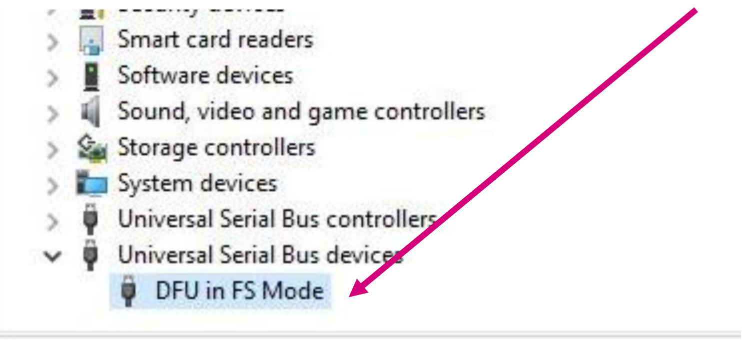


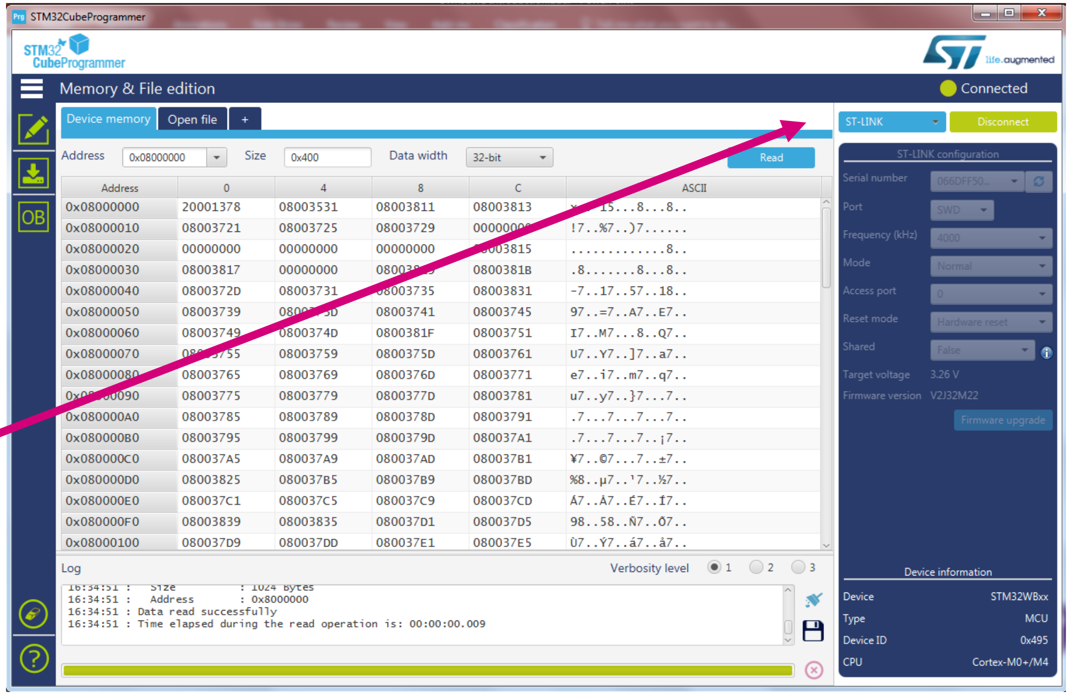
Figure 4. STM32 DFU device with STM32CubeProgrammer driver



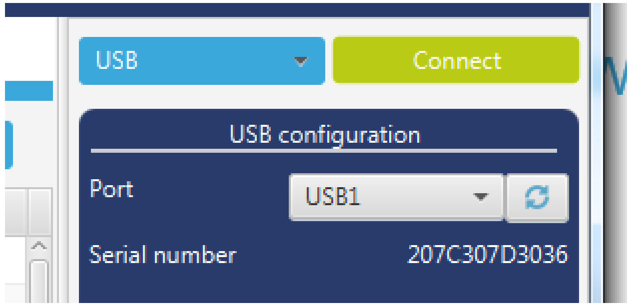
New Driver!



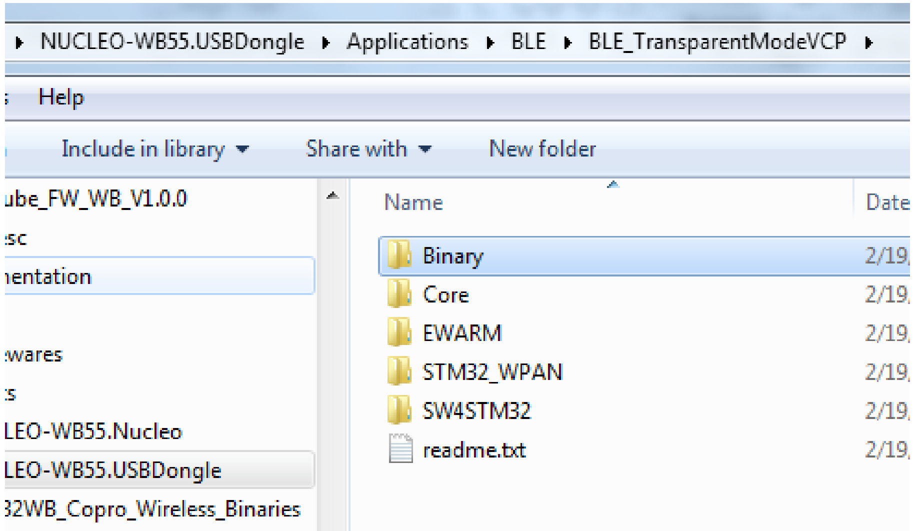
- Open STM32 CubeProgrammer

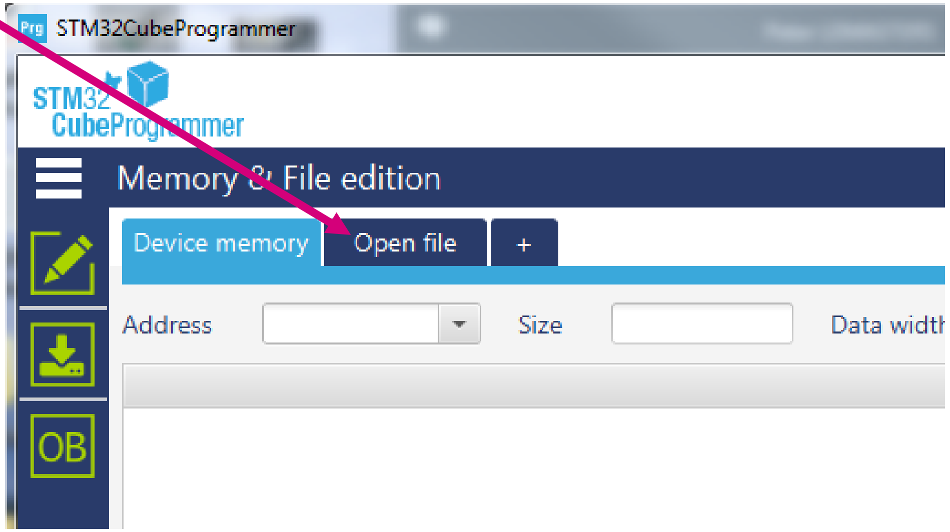


- Select **USB** mode and **Connect**

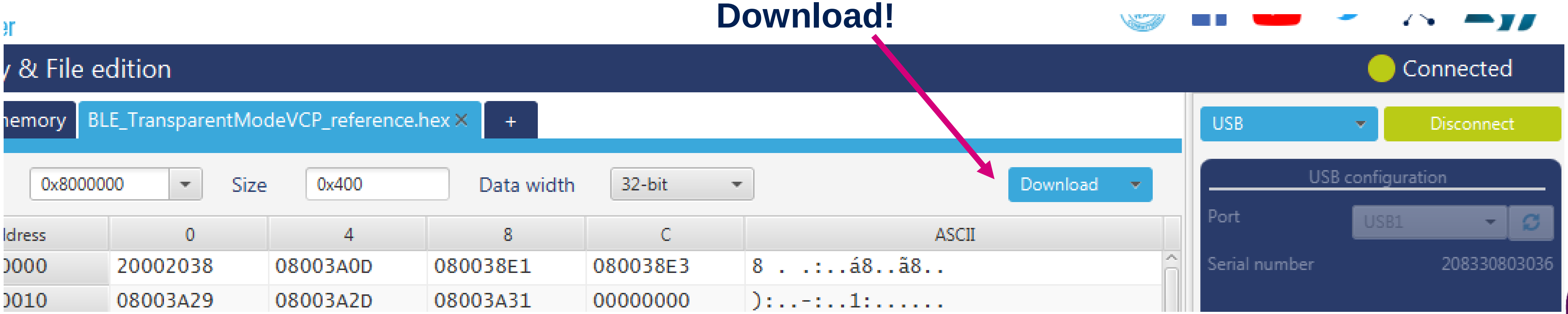


Open the BLE_TransparentModeVCP_reference.hex file for Dongle

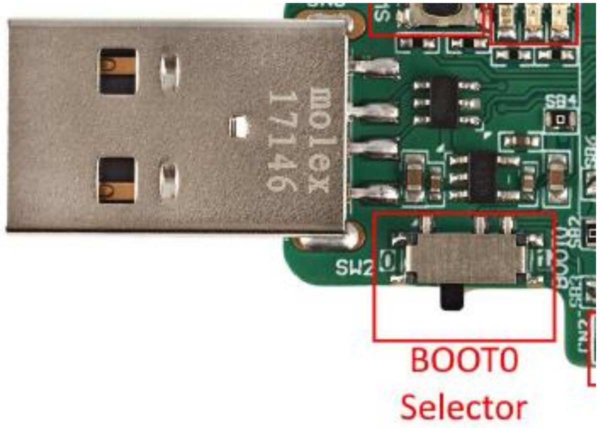




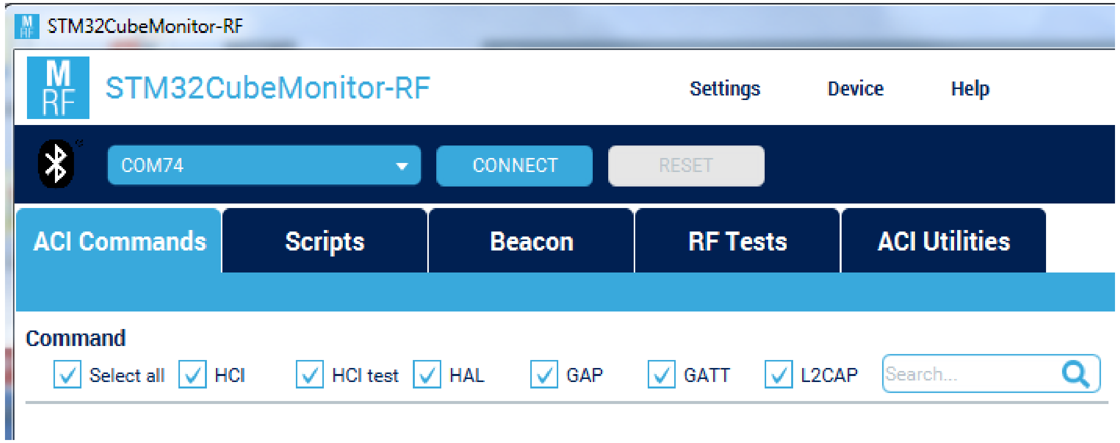
Download!



- Disconnect from CubeProgrammer
- Unplug Dongle
- Move Dongle Switch back to normal boot mode
- Plug Dongle back in for normal boot startup
- Now you should be able to use **COMxx** in CubeMonitorRF
  - *(may differ from COM74)*
- **CONNECT**

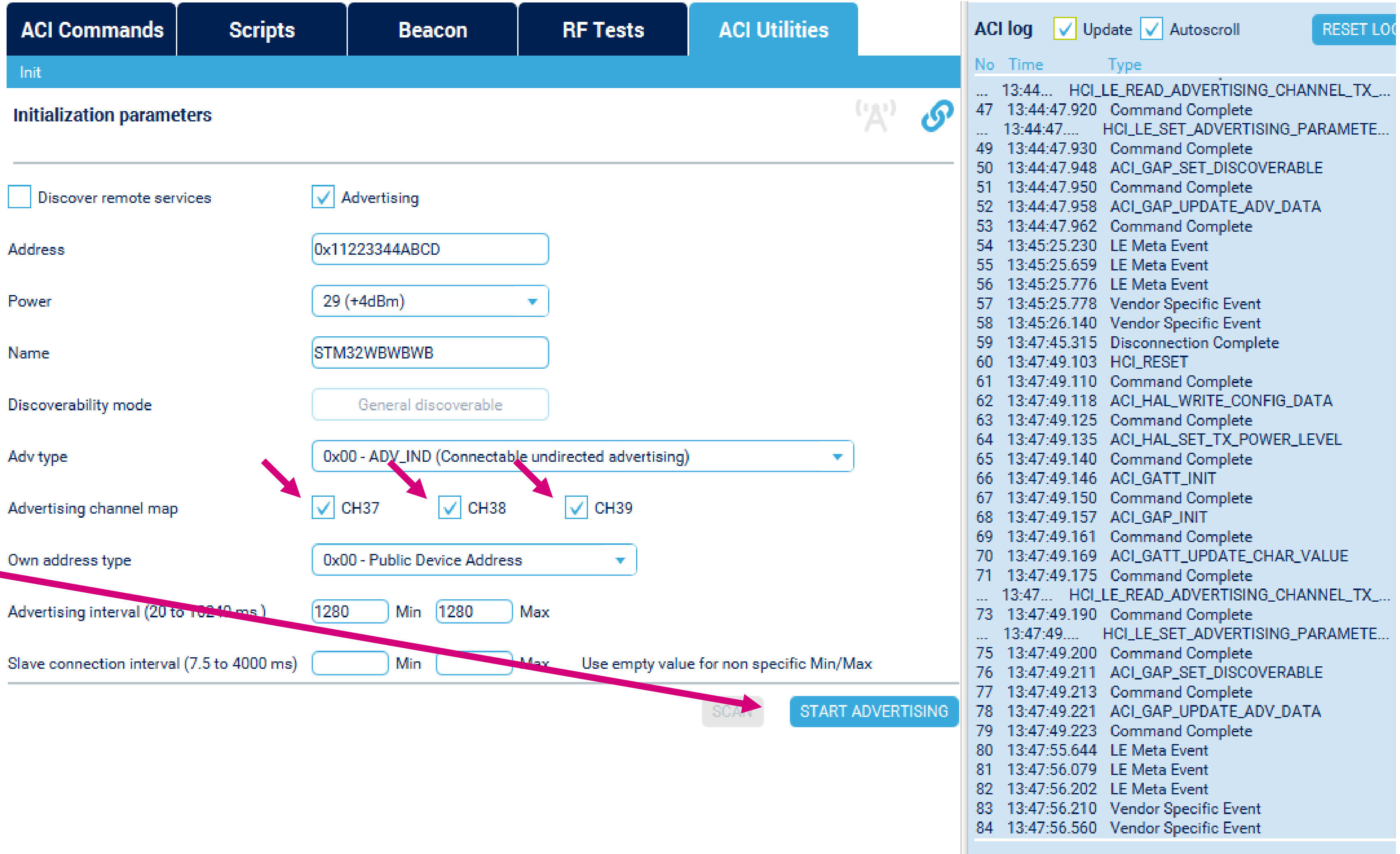


Normal Boot to the left

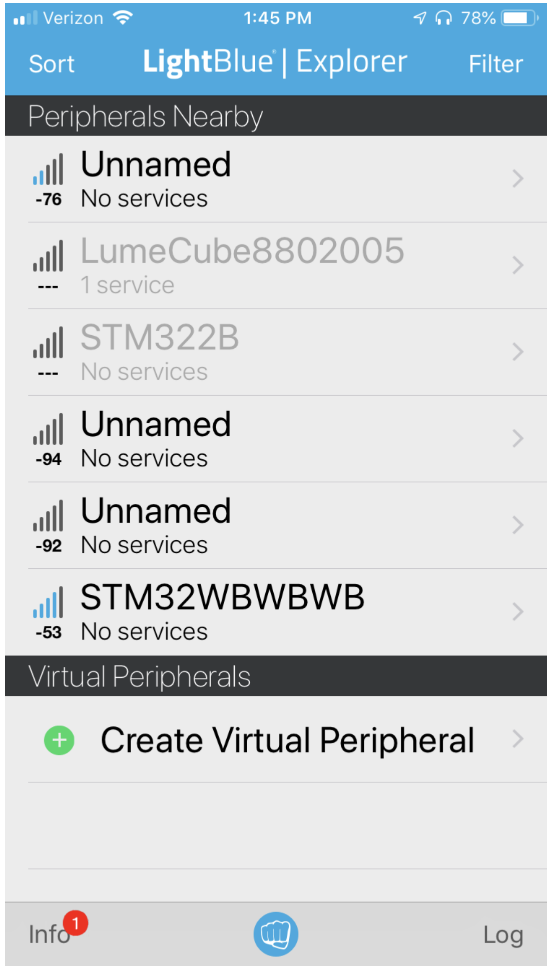


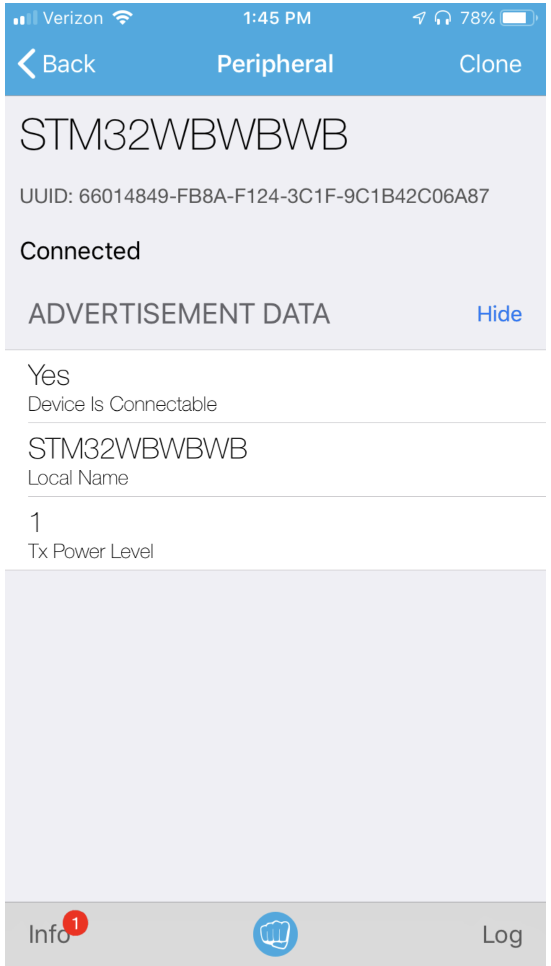
- Change the Bluetooth Address and Name. (Use your Magic Number!)
- Use Connectable advertising on all channels (37/38/39)

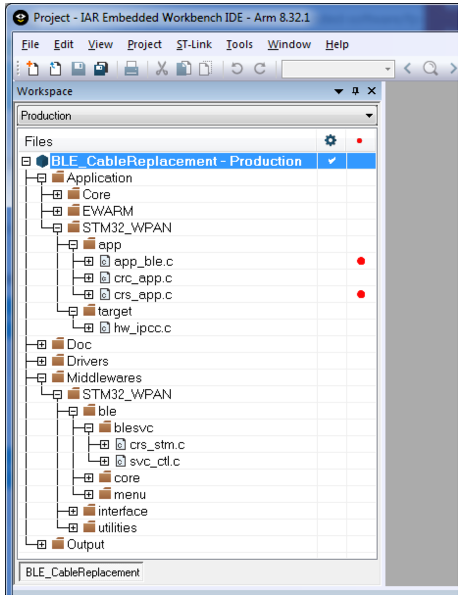
Start



- Use LightBlue Explorer to connect to and interrogate your GAP peripheral







Hands-On

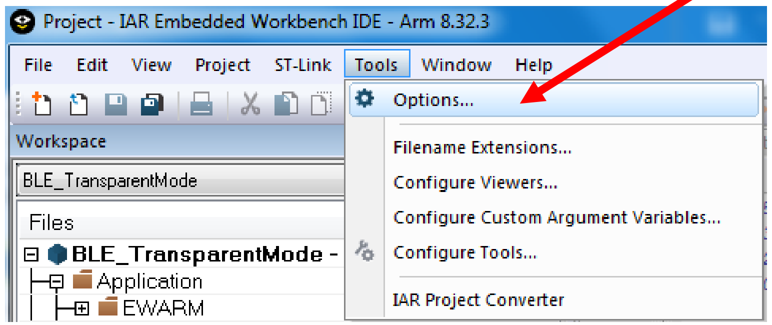
Custom GATT & Cable Replacement

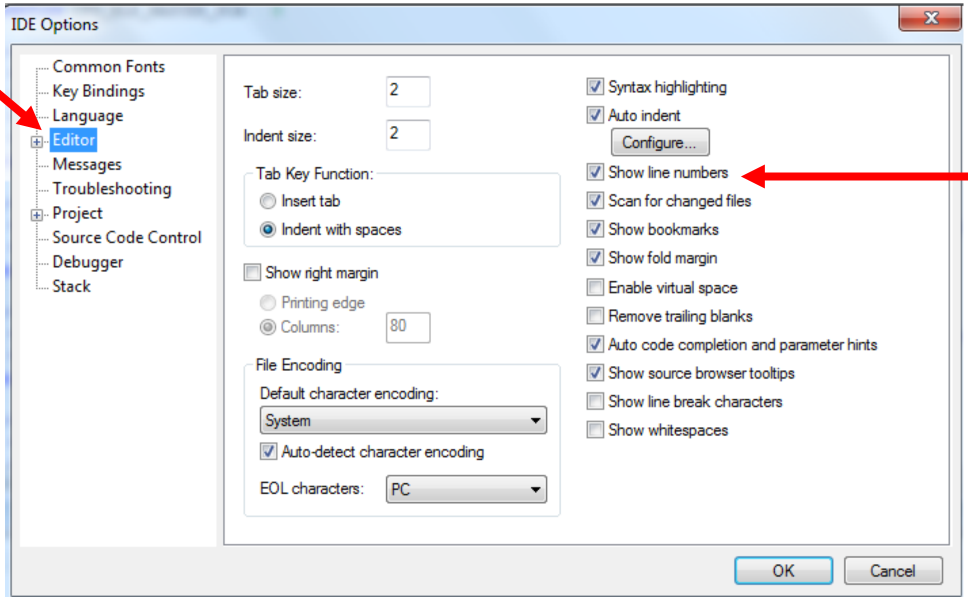
Open, Compile, Program & Run the Nucleo Board ***CableReplacement*** example

Add a custom GATT Characteristic for LED control

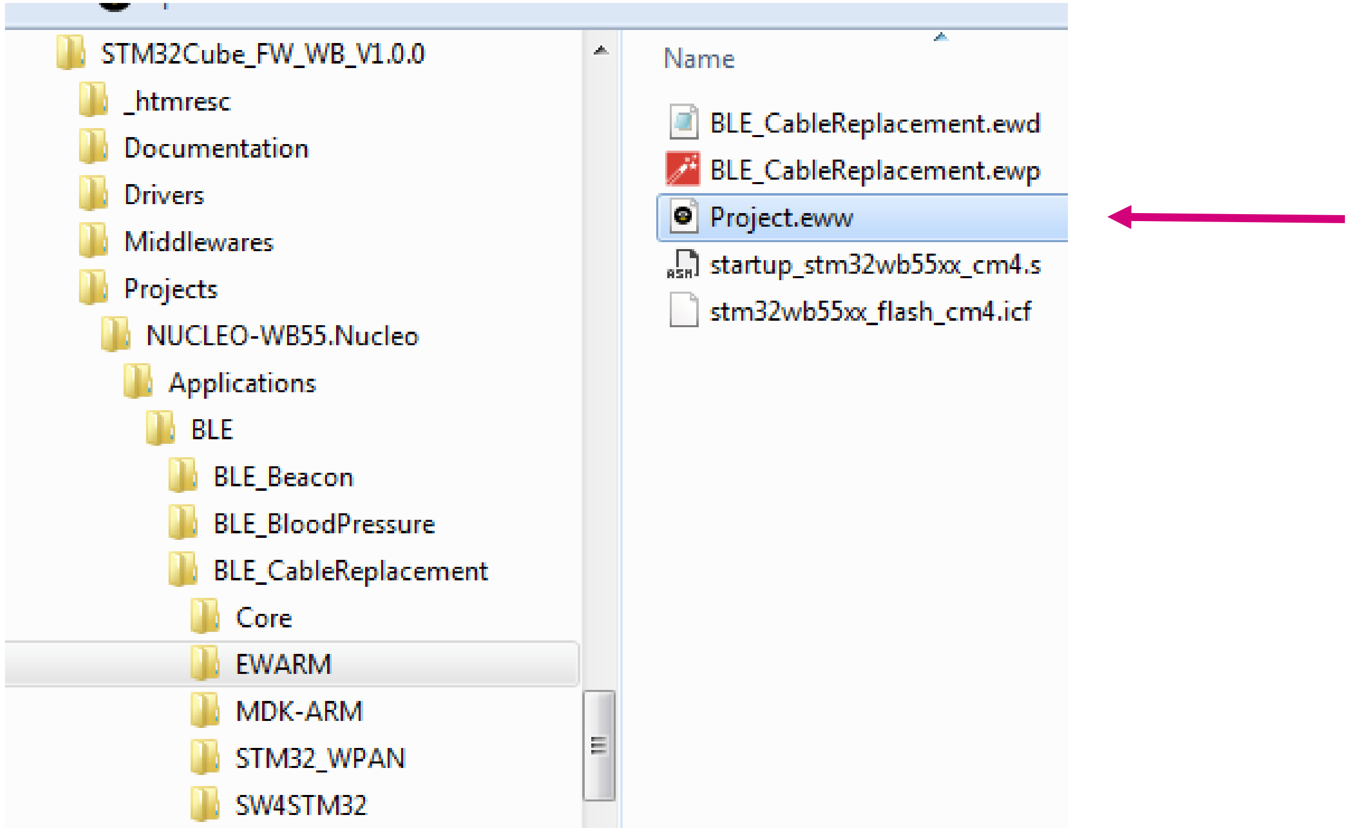
You can copy/paste the code bits from **CableReplacement_Lab.txt** *file from your install files* **Labs** *folder*

To add line numbers in IAR:





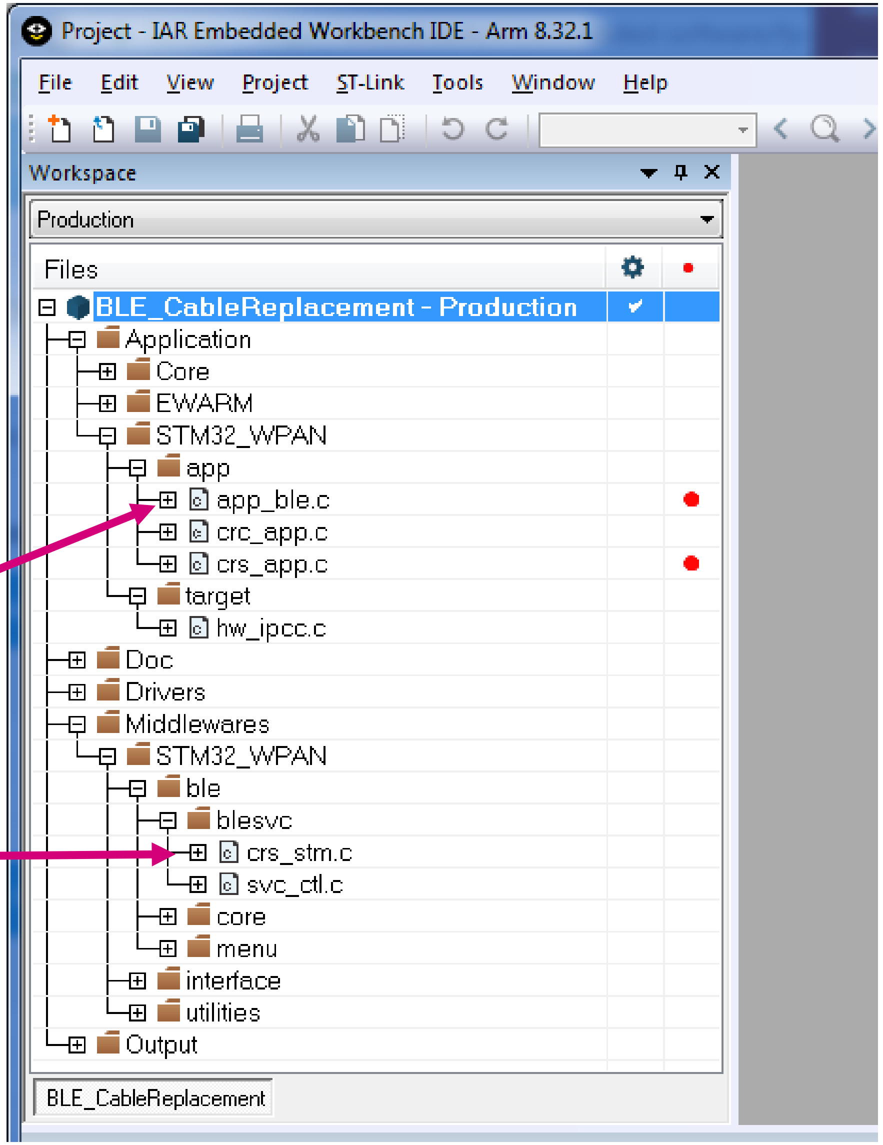
- Open the workspace



- Build the Project

- Open the following files
  - app_conf.h
  - app_ble.c
  - ble_conf.h
  - crs_stm.h
  - crs_stm.c
  - crs_app.c

To see the header files, expand the C source files



- STM32WB is the GAP Peripheral / GATT server
- Smartphone is the GAP Central / GATT client.

Compile for GATT Server

- Modify the #define (line# 100 of **app_conf.h)**

```
#define GATT_CLIENT          0    /* 1 = Device is GATT Client, 0 = Device is GATT Server */
```

Identify your unique device with your magic number

- Modify your local name (line# 204 of app_ble.c)

```
static const char local_name[] = { AD_TYPE_COMPLETE_LOCAL_NAME, 'C', 'R', 'S', '0', '1' };
```

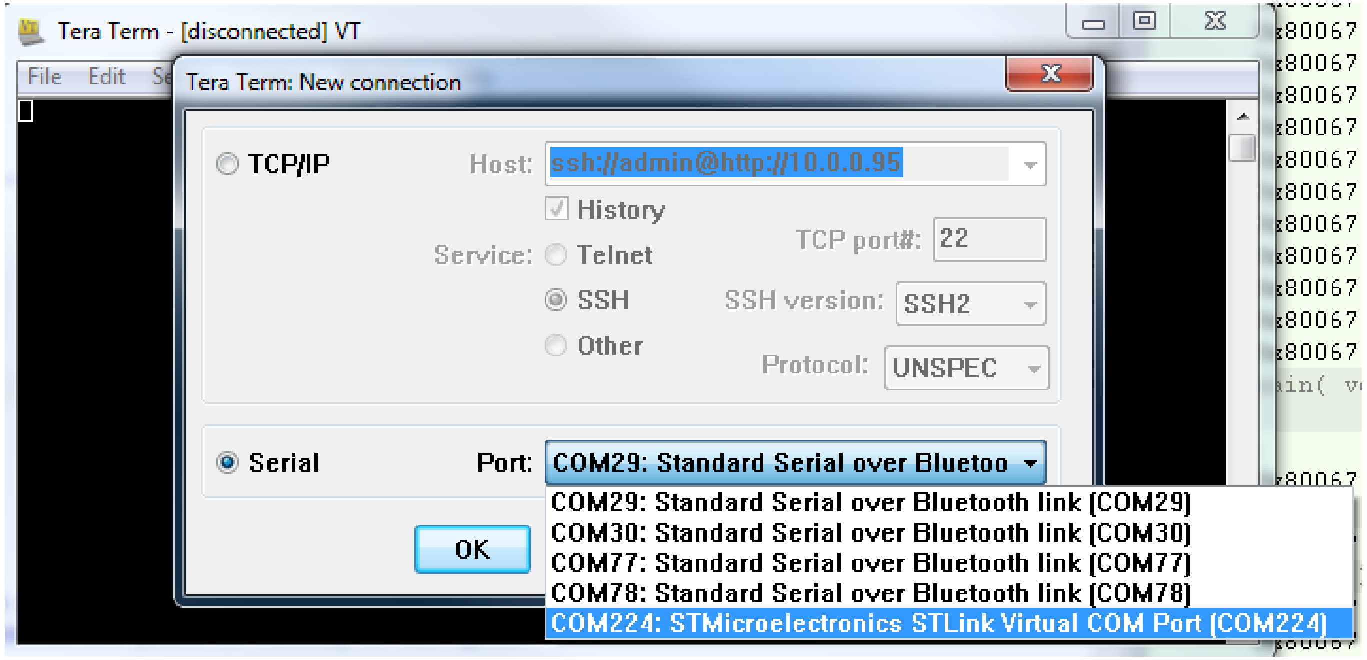
- Modify your BLE device name (line# 822 of app_ble.c)

```
const char *name = “BLEcore";
```

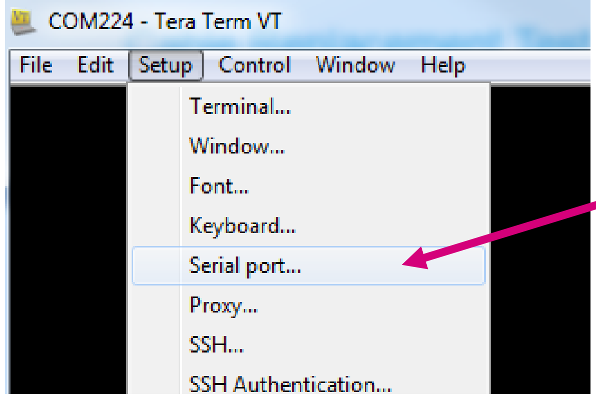
- Ensure that the BLE device name length in ASCII chars matches (line# 165 of app_ble.c)

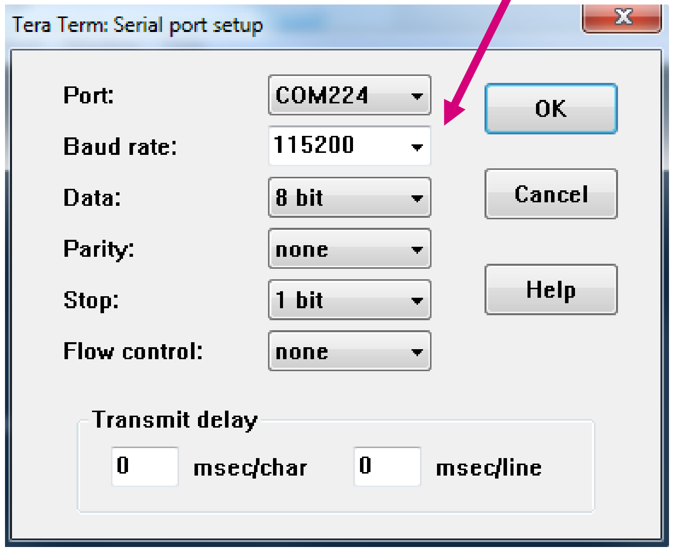
```
#define APPBLE_GAP_DEVICE_NAME_LENGTH 7
```

- Build and Run the project
- Connect your TeraTerm to the Nucleo's STLink Virtual COM port

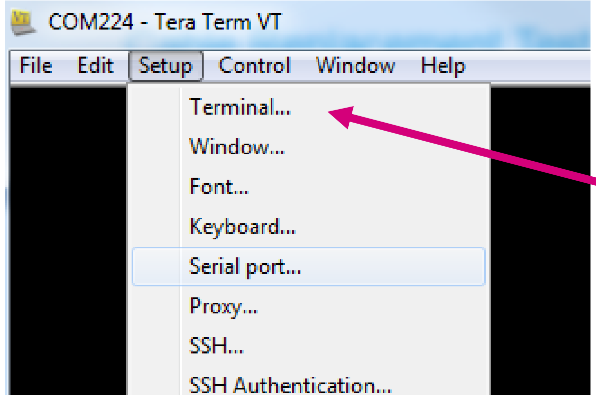


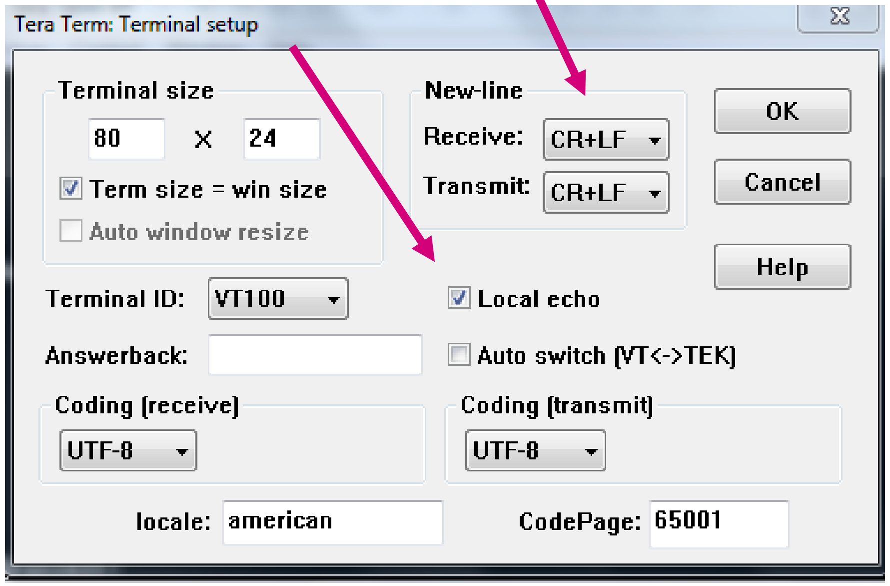
Configure your Serial port for 115,200bps / N / 8 / 1



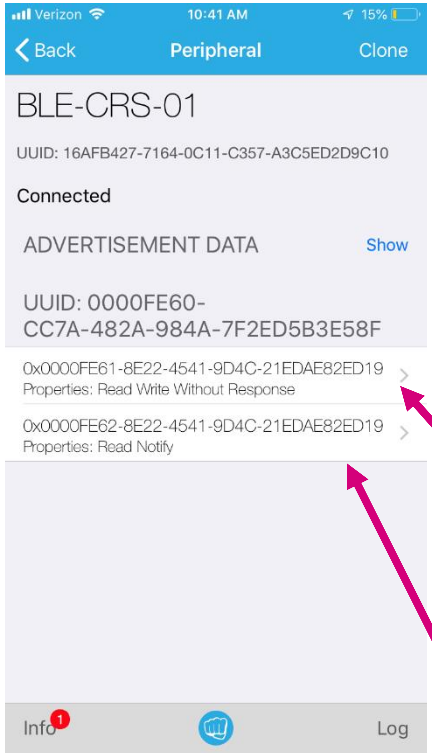


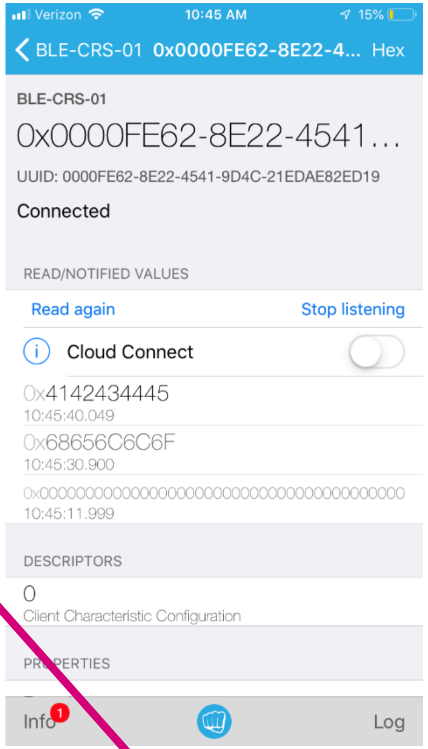
Configure Terminal Setup

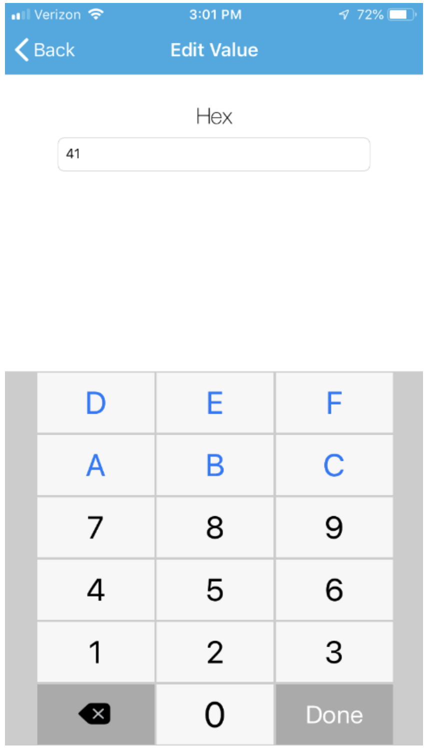


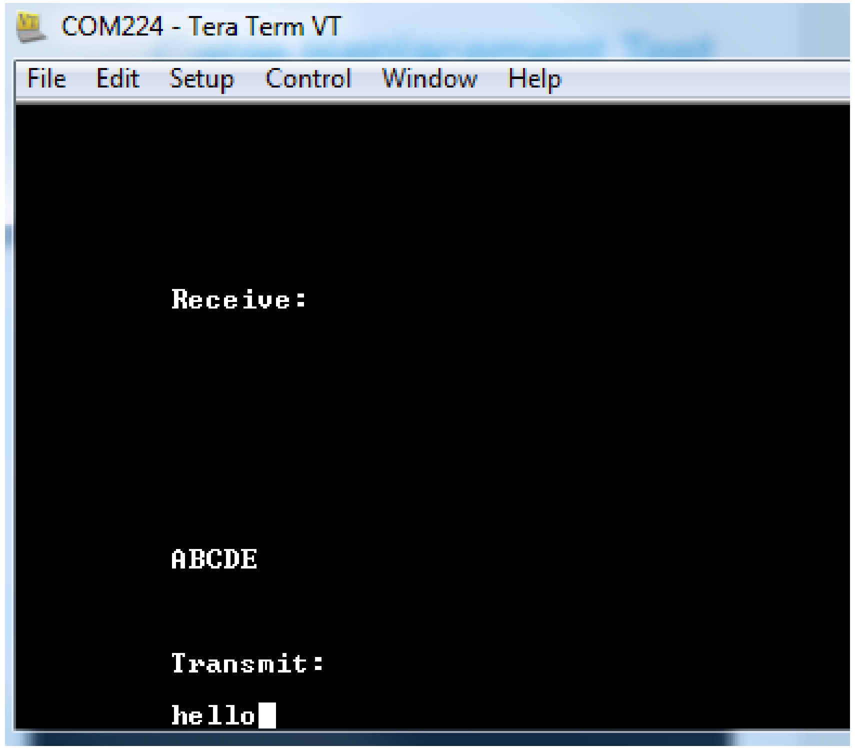


- Connect to your device with LightBlue Explorer
- Send and receive ASCII-based messages using the different characteristics









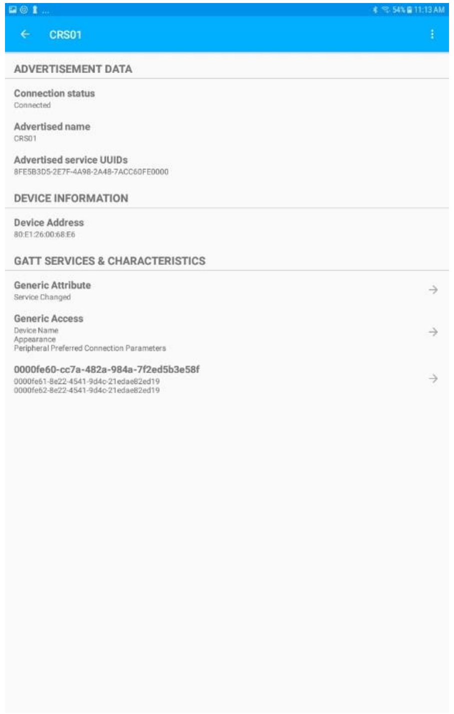
Write values to Nucleo

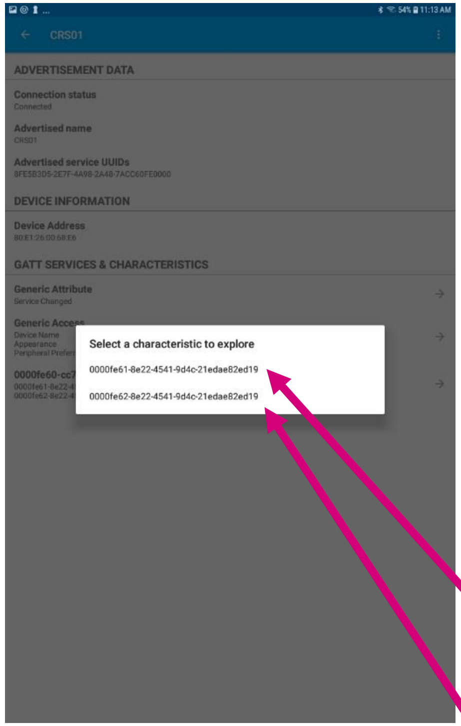
Enable Notifications to receive messages from Nucleo

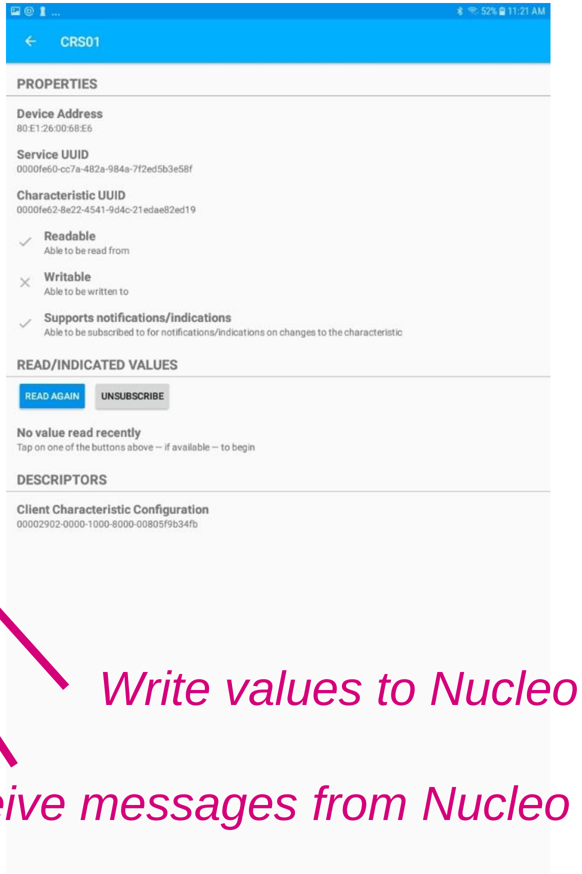
life.augmented

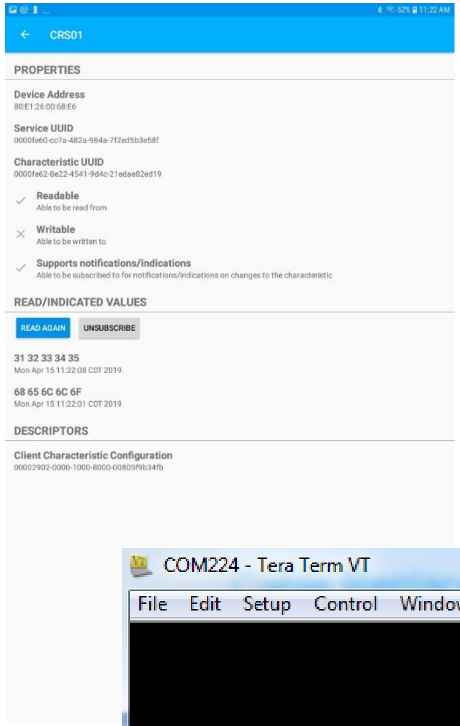
STM32 WB

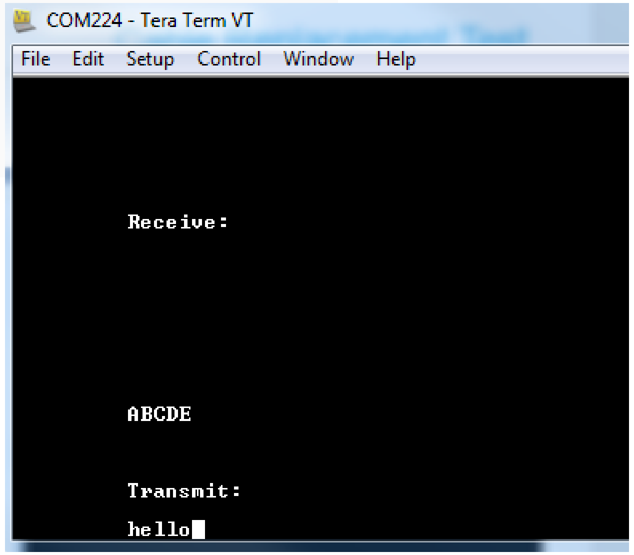
- Here is the LightBlue Explorer on Android











Write values to Nucleo

Enable Notifications to receive messages from Nucleo

life.augmented

STM32 WB

Add a custom characteristic to an existing Service

- Add the UUID definition (line# 74 of ble_conf.h)

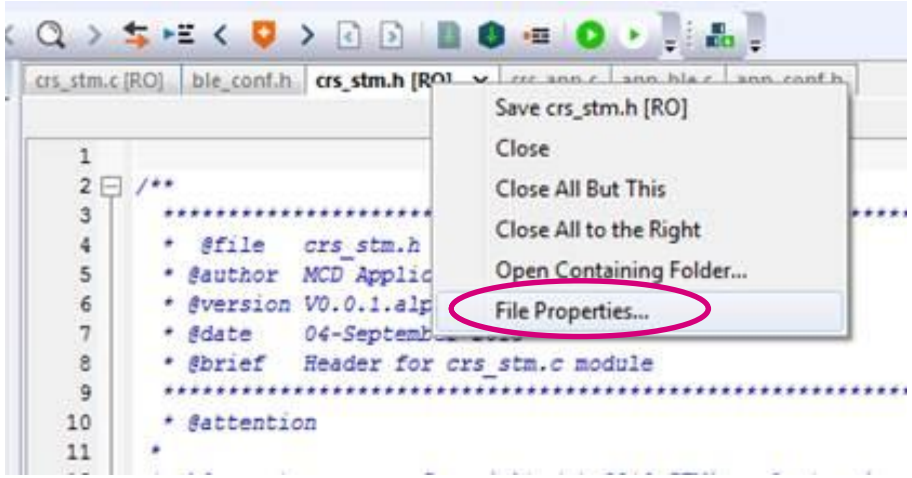
```
#define STM_LED_UUID128    0x00, 0x00, 0xfe, 0x64, 0x8e, 0x22, 0x45, 0x41, 0x9d, 0x4c, 0x21, 0xed, 0xae, 0x82, 0xed, 0x19
```

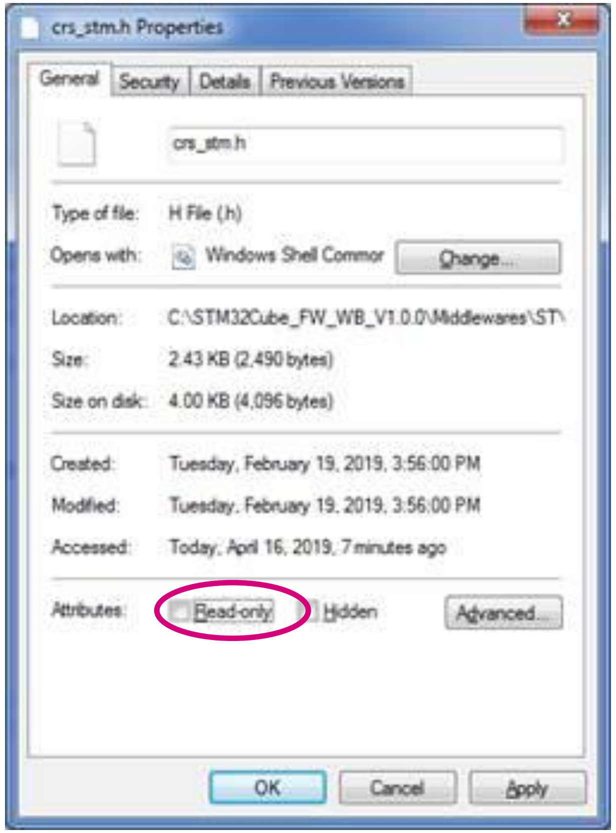
- Add event element (line# 37 of crs_stm.h)

```
typedef enum {
   STM_LED_WRITE_EVT,
   CRS_NOTIFY_ENABLED_EVT,
…
} CRS_Opcode_evt_t;
```

Note that crs_stm.h is read-only, you will need to change permissions

- From IAR, right click on the file tab and select “File Properties”
- Uncheck the “Read-only” box
- Click OK





- Add characteristic handle (line# 32 of crs_stm.c)

```
typdef struct {
  …
  uint16_t CRSRXCharHdle;
  uint16_t LedWriteClientToServerCharHdle;
} CRSContext_t;
```

Note that crs_stm.c is read-only, you will need to change permissions

- Check for the handle (line# 122 of crs_stm.c)

```
case EVT_BLUE_GATT_ATTRIBUTE_MODIFIED:
{
  attribute_modified = (aci_gatt_attribute_modified_event_rp0*)blue_evt->data;
  if(attribute_modified->Attr_Handle == (CRSContext.LedWriteClientToServerCharHdle + 1))
  {
    Notification.CRS_Evt_Opcode = STM_LED_WRITE_EVT;
    Notification.DataTransfered.Length = attribute_modified->Attr_Data_Length;
    Notification.DataTransfered.pPayload = attribute_modified->Attr_Data;
    CRSAPP_Notification(&Notification);
  }
```

life.augmented

STM32 WB

- Add uuid array (line# 193 of crs_stm.c)

```
uint8_t led_uuid[]      = { STM_LED_UUID128 };
```

- Change the Max_Attribute_Records parameter (line# 215 of crs_stm.c)

```
hciCmdResult = aci_gatt_add_service(
                              UUID_TYPE_128,
                              (Service_UUID_t *) &uuid,
                              PRIMARY_SERVICE,
                              8,
                              &(CRSContext.SvcHdle));
```

- Add LED characteristic (line# 281 of crs_stm.c)

```
COPY_CRS_UUID(uuid.Char_UUID_128, led_uuid);
hciCmdResult = aci_gatt_add_char(CRSContext.SvcHdle,
                                 UUID_TYPE_128,
                                 &uuid,
                                 2,    /* Char_Value_Length */
                                 CHAR_PROP_WRITE_WITHOUT_RESP,
                                 ATTR_PERMISSION_NONE,
                                 GATT_NOTIFY_ATTRIBUTE_WRITE,/* gattEvtMask*/
                                 10, /* encryKeySize */
                                 1,  /* isVariable */
                                 &(CRSContext.LedWriteClientToServerCharHdle));
```

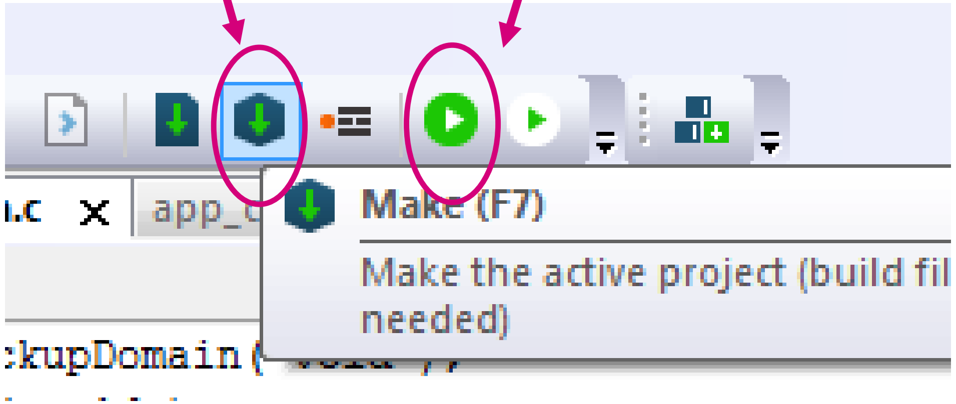
- Add event action (line# 194 of crs_app.c)

```
case STM_LED_WRITE_EVT:
      if(pNotification->DataTransfered.pPayload[0] == 0x01)
      {
        BSP_LED_On(LED_BLUE);
      }
      if(pNotification->DataTransfered.pPayload[0] == 0x00)
      {
        BSP_LED_Off(LED_BLUE);
      }
      break;
```

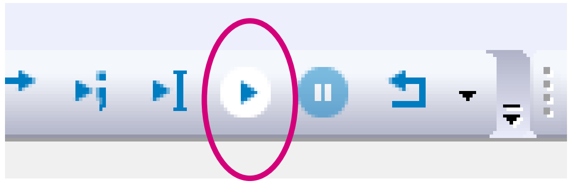
Compile and Program

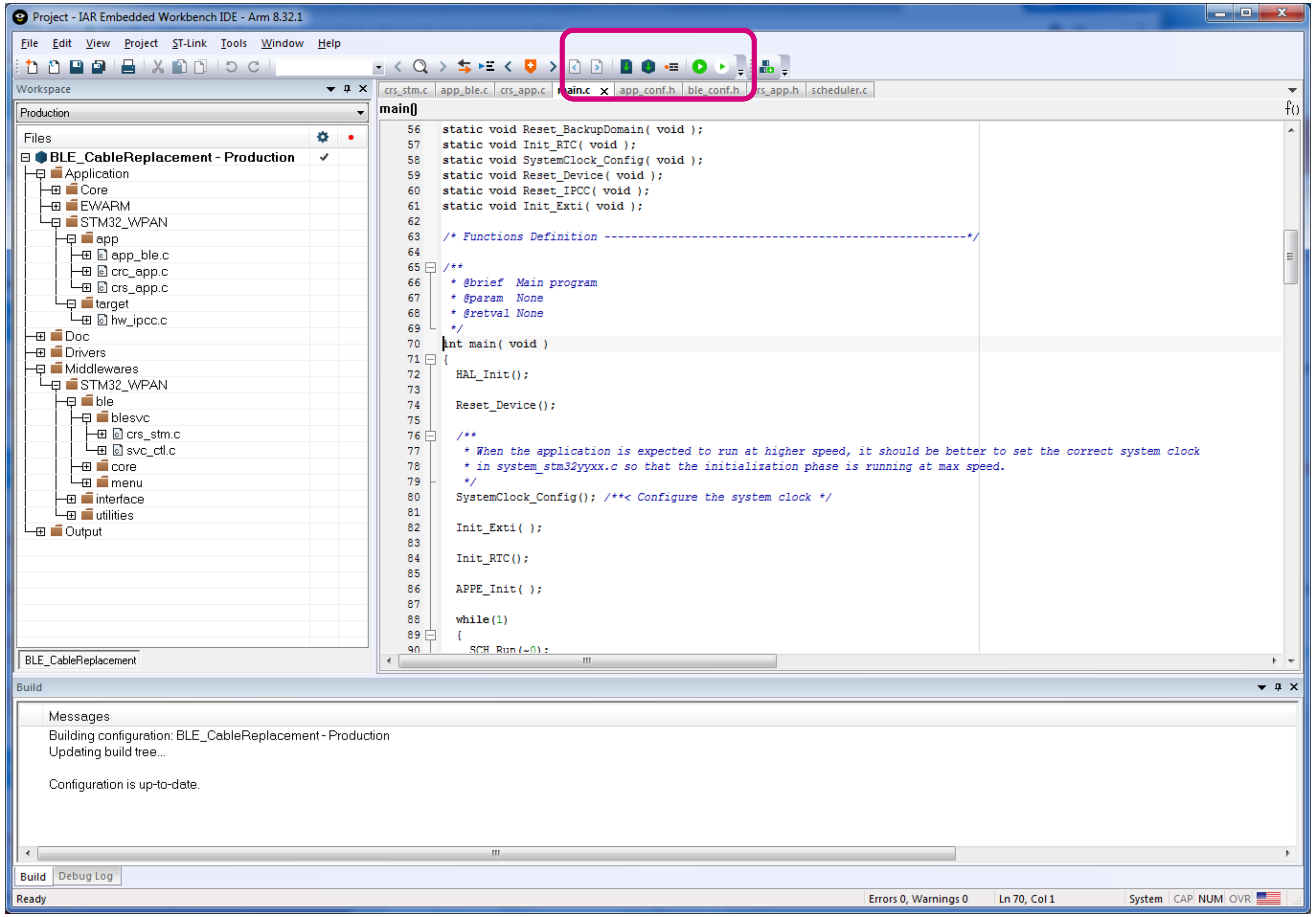
Build Project

Download and Debug



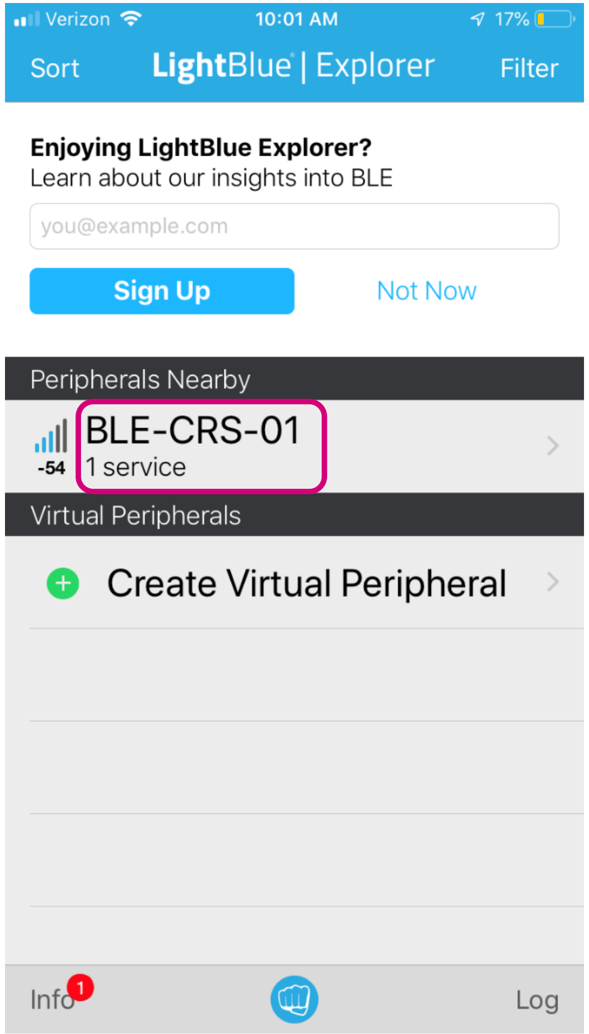
Run

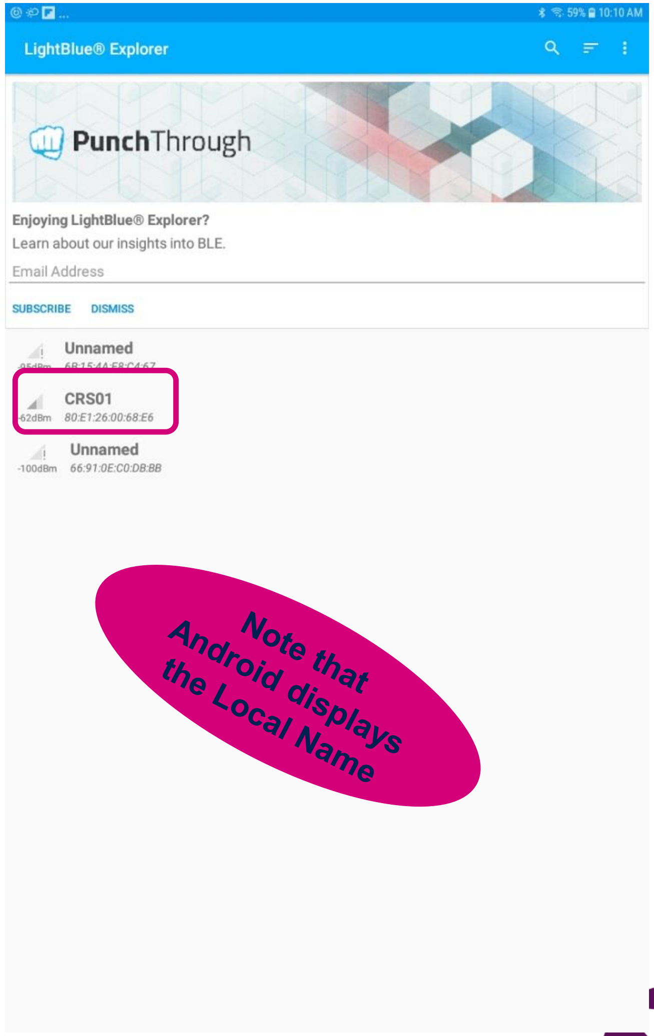




Start LightBlue App

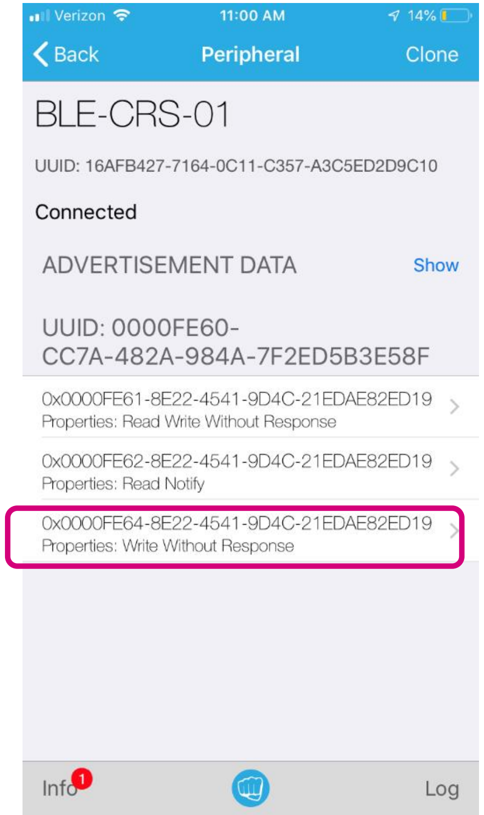
- Launch the LightBlue app
- Find your device
- Select your device

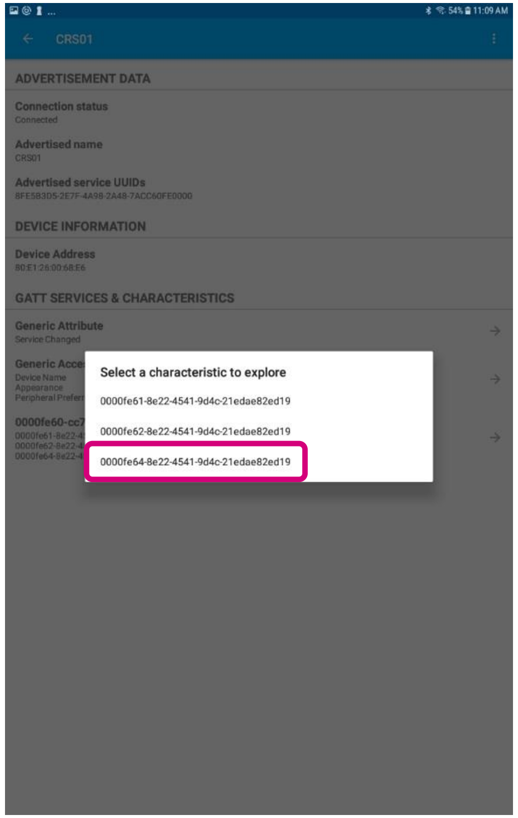




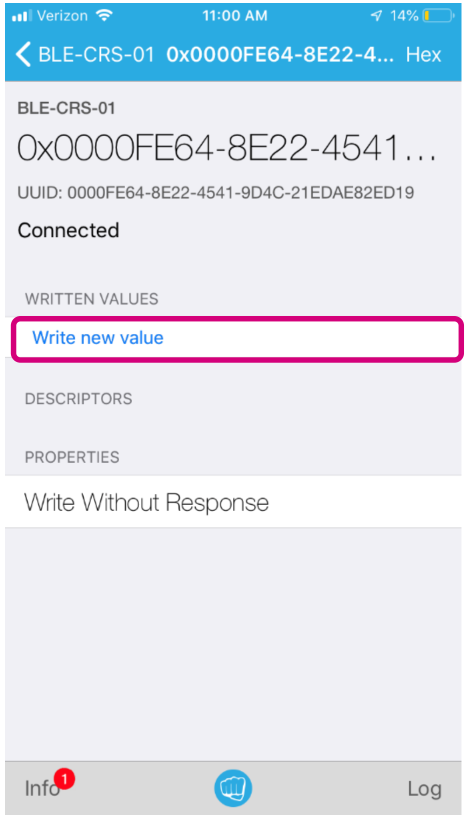
Note that Android displays the Local Name

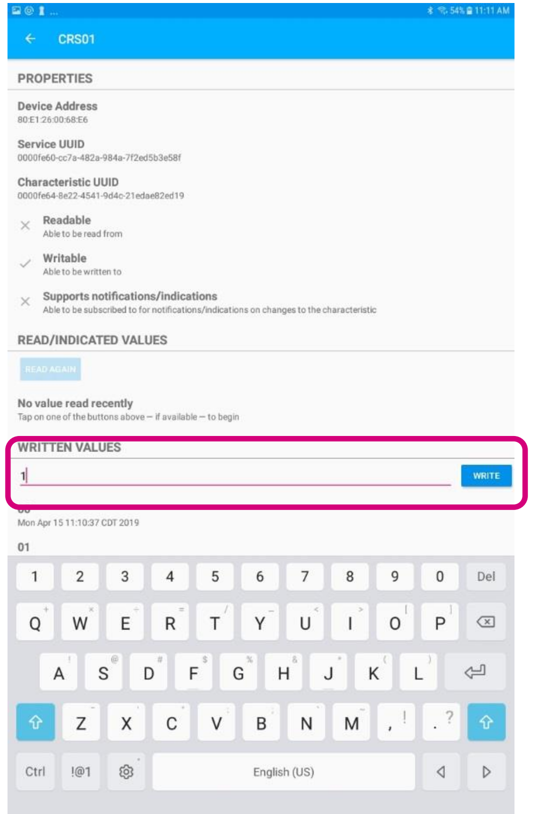
Find your LED characteristic UUID





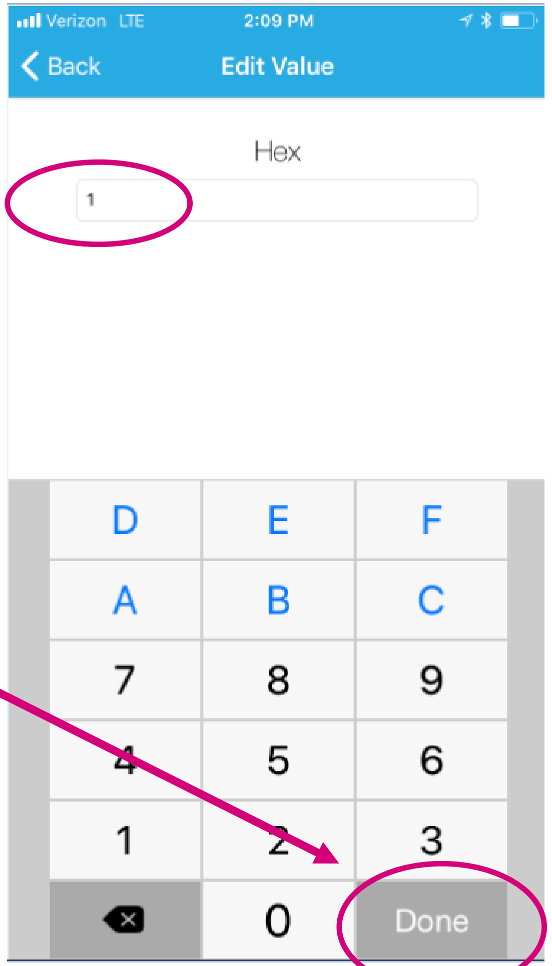
Write new value





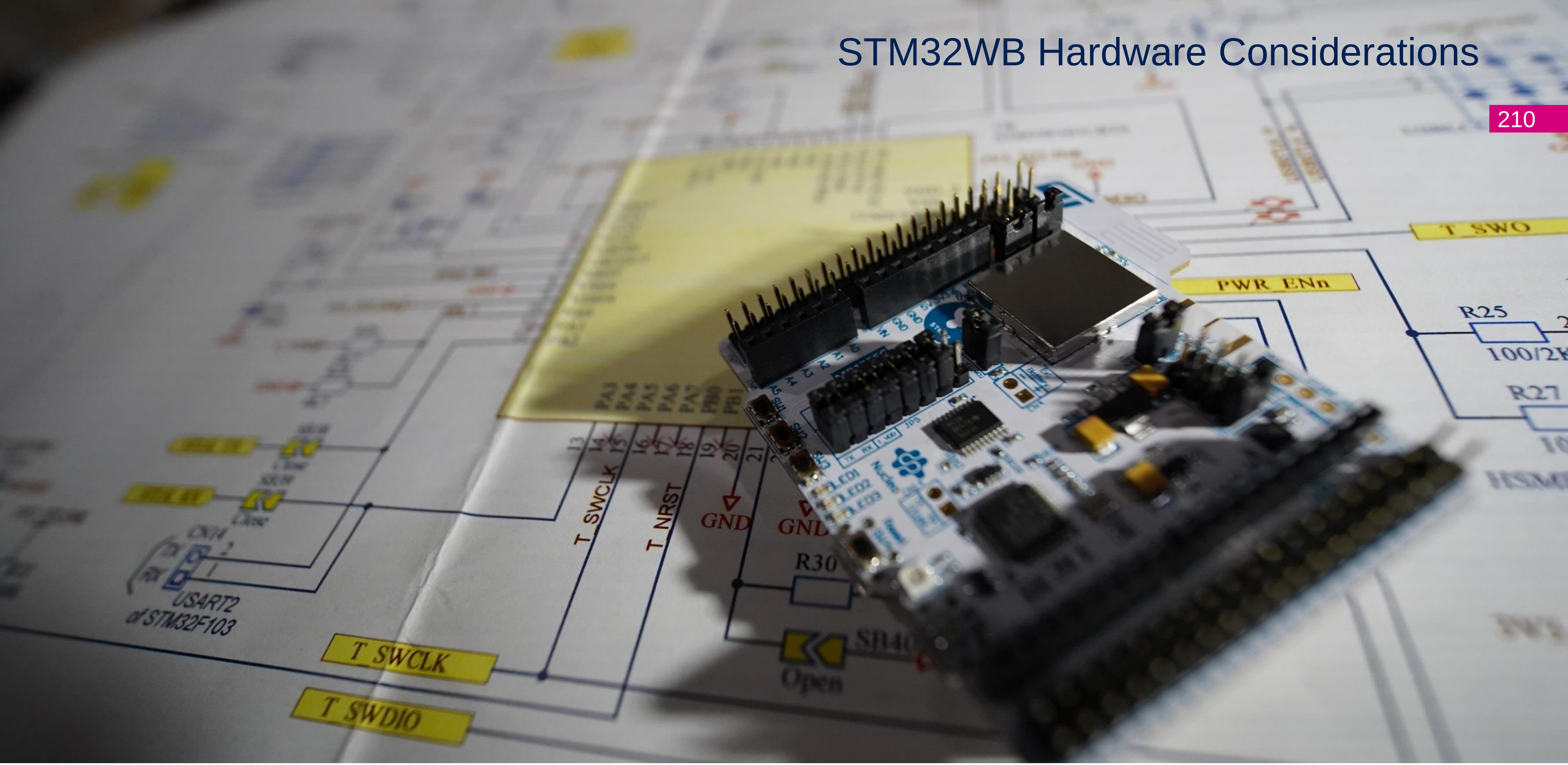
Write a value

- LED ON = 1
- LED OFF = 0



Concurrently, the CableReplacement characteristics can also be used

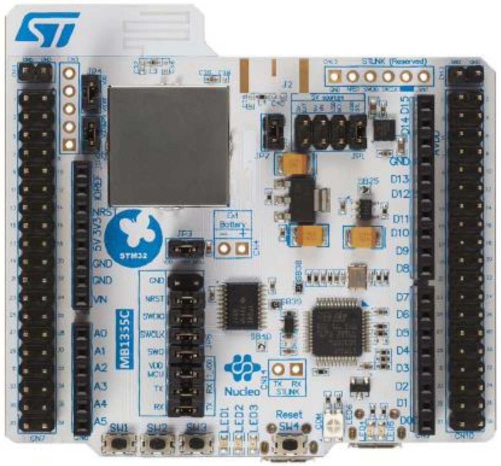
STM32WB Hardware Considerations



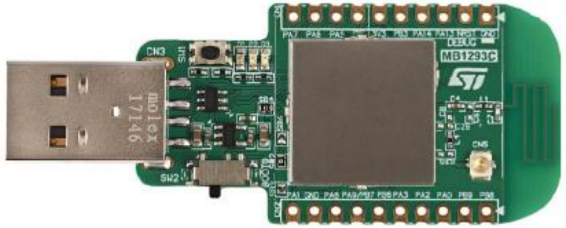
life.augmented

STM32 WB

FUS + BLE Stack



FUS + BLE Stack



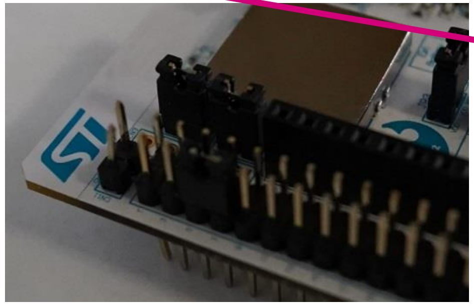
FUS only

Stack must be loaded

Nucleo Hardware Config for Bootloader access

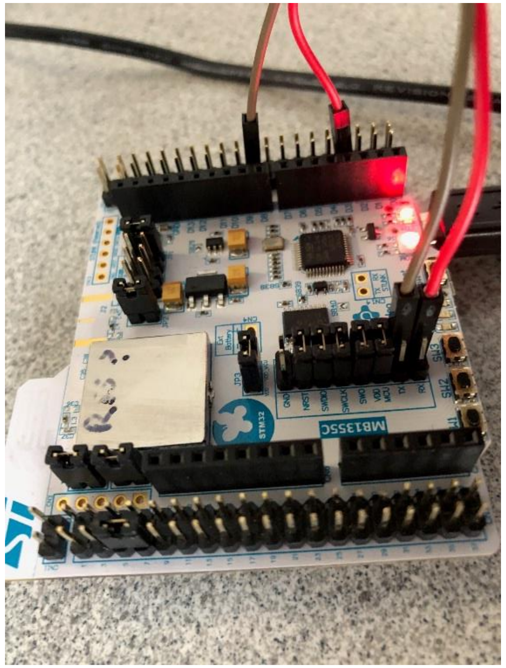
Only **USB-DFU or USART1** (on PA9/PA10 only) bootloader modes available for secure stack loading!

BOOT0 pin HIGH: CN7-5 to CN7-7

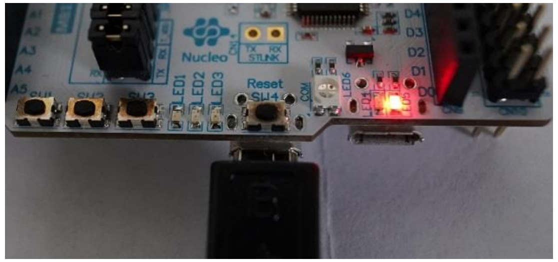


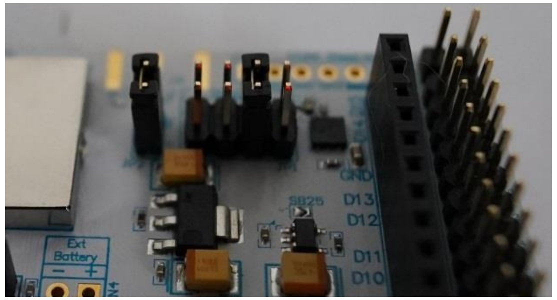
USART1 on PA9/10 to ST-LINK

- This is not the native USART connection to the STLINK!
- STL-RX to CN10-19 (PA9)
- STL-TX to CN10-31 (PA10)



USB User connector + Power Jumper JP2 to 5-6

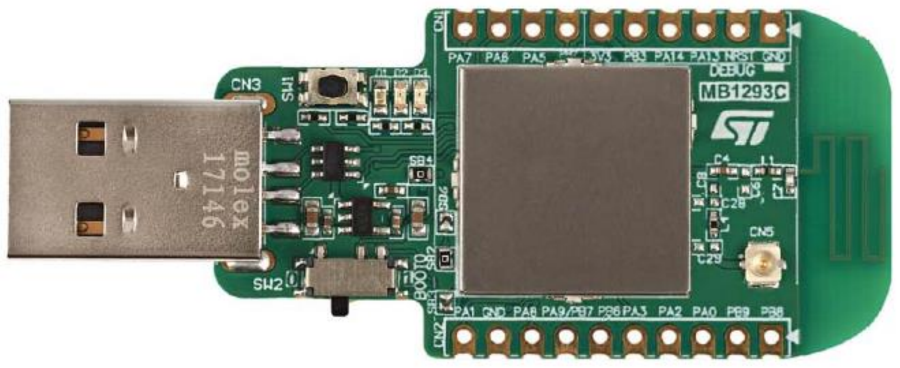


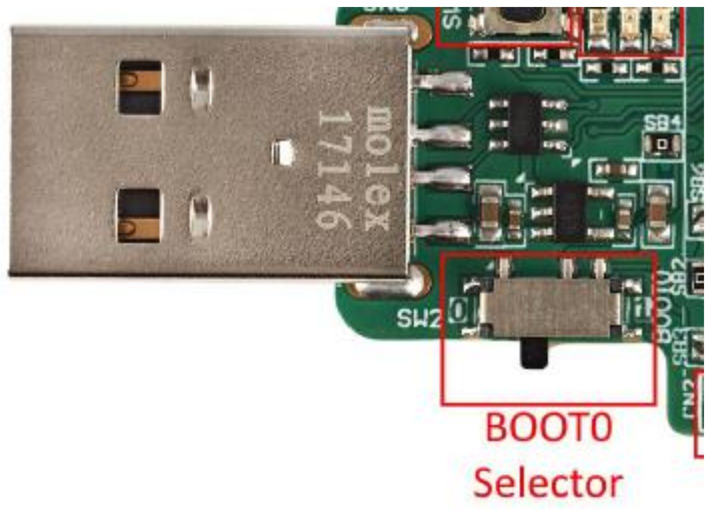


Dongle Hardware Config for Bootloader access

As we have seen, configuring the Dongle board for USB-DFU is quite easy.

Move the switch and repower the board





Bootloader active to the right

AN5185 details the sequence to create your own secure stack loader project, running on the M4

- Command / Response HCI event transactions to the M0+ similar to BLE

FUS commands

FUS uses same commands/response structure as wireless stacks and based on HCI model. FUS uses a subset of the HCI commands, namely:

- Vendor specific HCI command packet: used to send command from Cortex®-M4 to Cortex®-M0+.
- HCI command complete event packet: used to send response from Cortex®-M0+ to Cortex®-M4
- Vendor specific HCI event packet: used to send asynchronous events from Cortex®-M0+ to Cortex®-M4.

Figure 6. FUS HCI subset

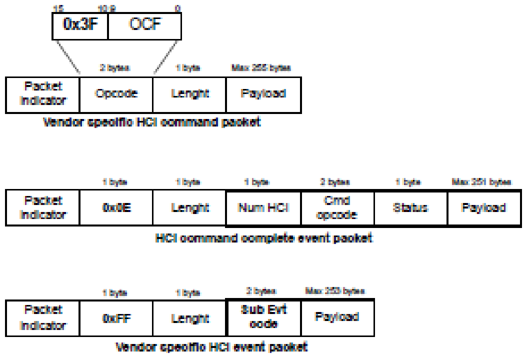
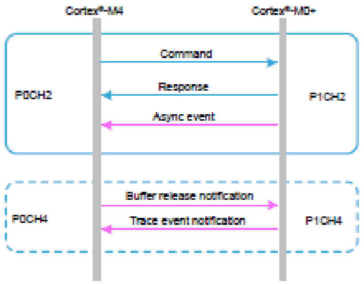
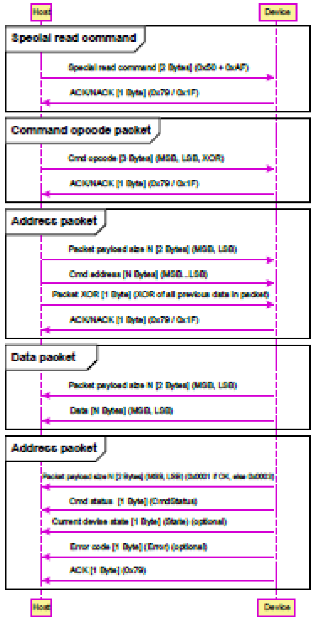


Figure 5. IPCC channels used by FUS



Also details on the bootloader sequences used

Figure 10. USART special read command



STM32 system bootloader extension for FUS

A command set extension has been added to STM32WB system bootloader in order to support FUS operation. These commands are implemented on USART and USB-DFU interfaces and follow the same rules as existing standard bootloader commands.

In order to help to understand this section, a prior reading of *STM32 microcontroller system memory boot mode* (AN2606) and *USART protocol used in the STM32 bootloader* (AN3155) and *USB DFU protocol used in the STM32 bootloader* (AN3156) documentation is required.

life.augmented

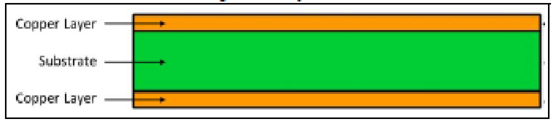
AN5165 details RF hardware considerations

- PCB stackup recommendations
- RF Front-end (discrete or IPD-based)
- SMPS passives selection
- Clocks

2-layer PCB

With the 2-layer PCB (see *Figure 21*), the RF signals and routing are on the top layer while the bottom layer is used for grounding under the RF zones, and for routing in others parts. The ground plane must be continuous under the RF zones, otherwise the return path current can increase and degrade the RF performance.

Figure 21. 2-layer PCB

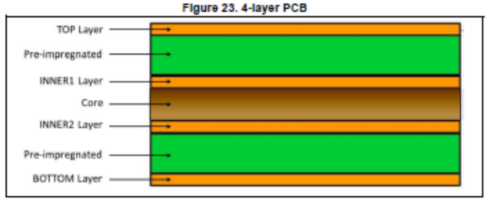


4-layer PCB

With the 4-layer PCB shown in *Figure 23*, it is recommended to have the following distribution:

- TOP layer: RF signal and routing on the top layer.
- INNER1 layer: grounding under the RF zones, routing in the others parts.
- INNER2 layer: power and low frequency routing.
- BOTTOM layer: low frequency routing.

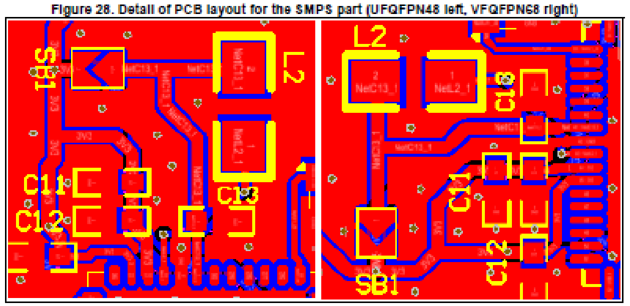
Figure 23. 4-layer PCB



6.1.2 SMPS

In addition to the recommendations given in *Section 4.3: SMPS*, to avoid important current loop when the STM32WB is in SMPS mode, it is recommended to place C11, C12 and C13 as close as possible to their respective pins on STM32WB. Do not forget to connect the solder pad to ground to have a strong current return path.

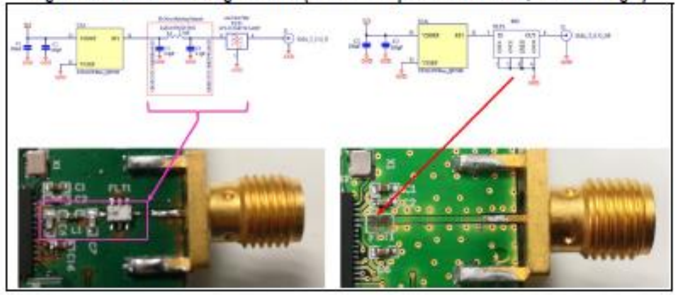
Figure 28. Detail of PCB layout for the SMPS part (UFQFPN48 left, VFQFPN68 right)



8 UFQFN48/VFQFN68 reference boards with IPD

The goal of the IPD (integrated passive device) is to replace the discrete matching network plus the integrated low-pass filter keeping equivalent TX/RX performance. *Figure 41* shows the differences between the two approaches.

Figure 41. Different matching networks (discrete components on the left, IPD on the right)



Layout recommendations for the 2-layer PCB

Figure 25. PCB layout for UFQFPN48 (left to right: all, top and bottom layers)

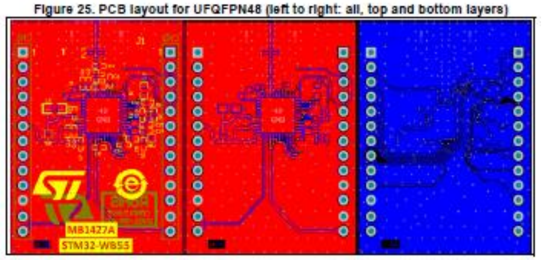


Figure 26. PCB layout for VFQFPN68 (left to right: all, top and bottom layers)

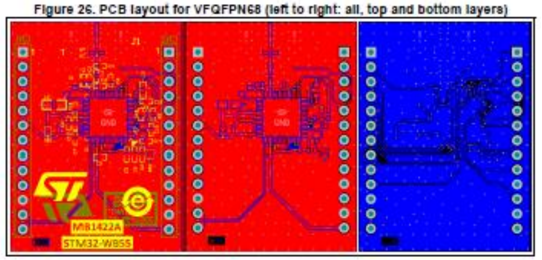
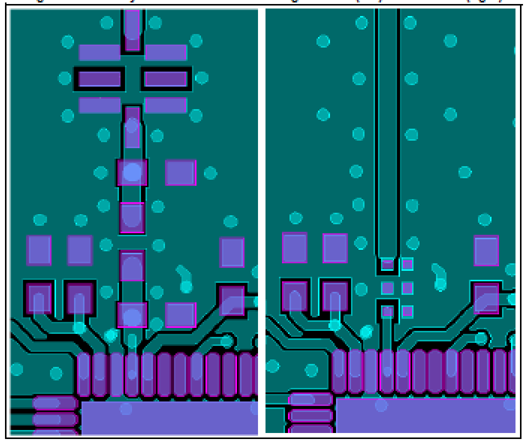


Figure 44. PCB layout with discrete matching network (left) and with IPD (right)



AN5290: Minimal BOM

AN5290 details the minimal Bill-of-Materials needed for various scenarios

Figure 5. Optimized solution with IPD

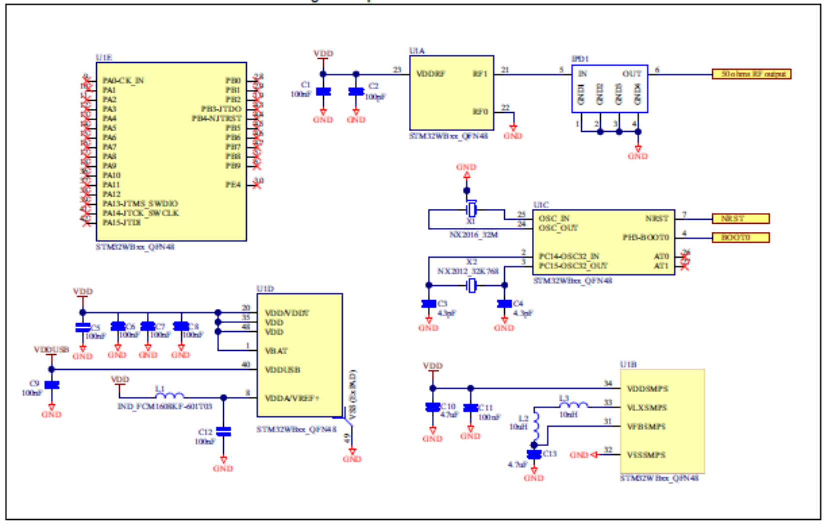


Table 2. Bill of materials- Optimized solution with IPD

Designator	Description	Comment	Footprint	Manufacturer	Part number
C1, C5, C6, C7, C8, C9, C11, C12	Capacitor, not polarized (X5R)	100 nF decoupling capacitors	0402	Murata	GRM155R61H104KE19D
C2	Capacitor, not polarized	100 pF decoupling capacitors		Yageo	CC0402KRX7R9BB101
C3, C4		4.3 pF LSE crystal capacitor		Murata	GRM1555C1H4R3CA01D
C10, C13		4.7 µF decoupling capacitor			GRM155R61A475MEAAD
L1	Coil	Filtering coil	0603	TAI-TECH	FCM1608KF-601T03
L2	Inductor	10 µH SMPS inductor	0805	Murata	LQM21FN100M70L
L3		10 nH SMPS inductor	0402		LQG15WZ10NJ02D
X1	Crystal	32 MHz - HSE	NX2016	NDK	NX2016SA_32MHz
X2		32.768 kHz - LSE	NX2012		NX2012SA_32-768kHz
IPD1	Integrated passive device	Matching network and low-pass filter	Bumpless CSP	STMicroelectronics	MLPF-WB55-01E3

Figure 4. Optimized solution with discrete components

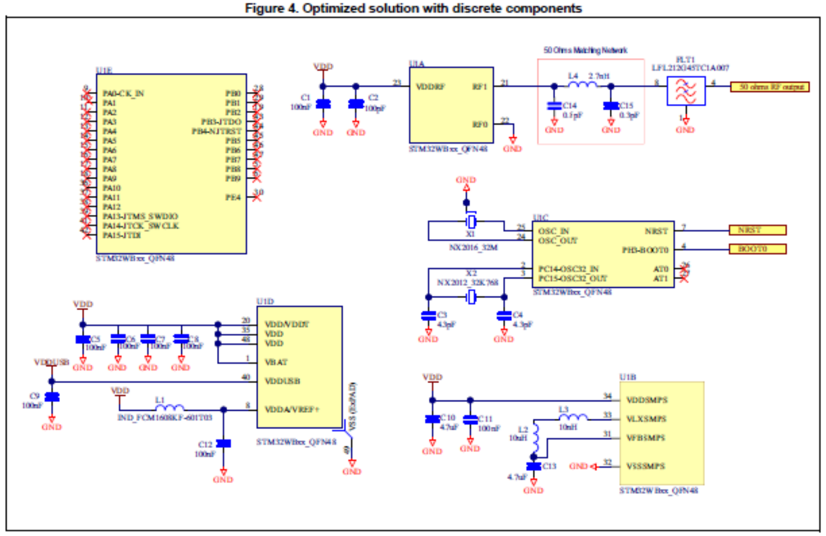


Table 1. Bill of materials - Optimized solution with discrete components

Designator	Description	Comment	Footprint	Manufacturer	Part number
C1, C5, C6, C7, C8, C9, C11, C12	Capacitor, not polarized (X5R)	100 nF decoupling capacitors	0402	Murata	GRM155R61H104KE19D
C2	Capacitor, not polarized	100 pF decoupling capacitors		Yageo	CC0402KRX7R9BB101
C3, C4		4.3 pF LSE crystal capacitor		Murata	GRM1555C1H4R3CA01D
C10, C13		4.7 µF decoupling capacitor			GRM155R61A475MEAAD
C14		0.8 pF matching network			GRM1555C1HR80BA01D
C15		0.3 pF matching network			GRM1555C1HR30WA01D
L1	Coil	Filtering coil	0603	TAI-TECH	FCM1608KF-601T03
L2	Inductor	10 µH SMPS inductor	0805	Murata	LQM21FN100M70L
L3		10 nH SMPS inductor	0402		LQG15WZ10NJ02D
L4		2.7 nH matching network			LQG15HS2N7S02D
X1	Crystal	32 MHz - HSE	NX2016	NDK	NX2016SA_32MHz
X2		32.768 kHz - LSE	NX2012		NX2012SA_32-768kHz
FLT1	Low-pass filter	Harmonics rejection	-	Murata	LFL212G45TC1A007

life.augmented

STM32 WB

AN5129 details a “meander-style” PCB antenna design

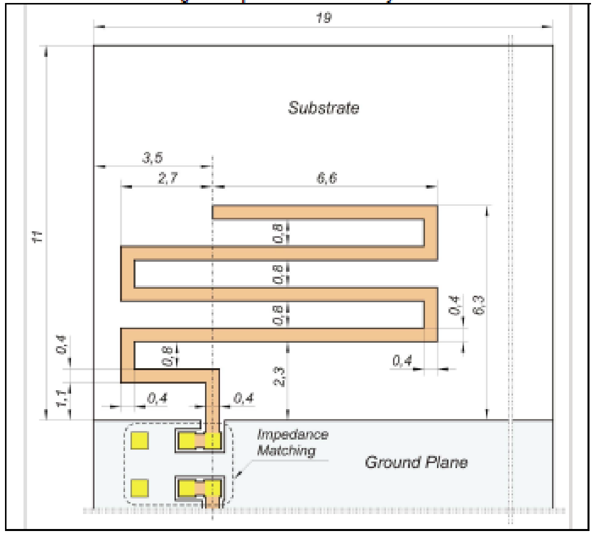
AN5129
Application note

Low cost PCB antenna for 2.4 GHz radio: meander design for STM32WB Series

3 Layout specification

The PCB antennas, including the electrical parameters of PCB materials used, are layout sensitive. It is recommended to use a layout as close as possible to the one shown in *Figure 2*.

Figure 2. Spherical coordinate system



The electrical parameters and performance of the PCB antenna are also determined by the substrate used, in particular the thickness of the core and dielectric constants.

Figure 3. PCB cross section at antennae area

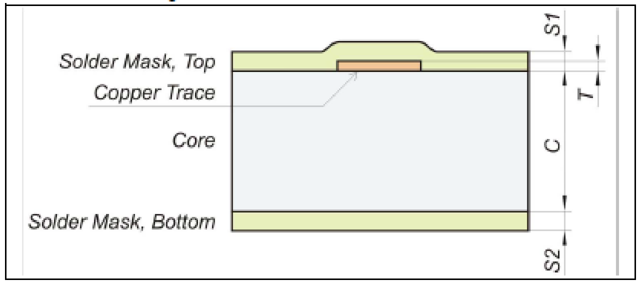
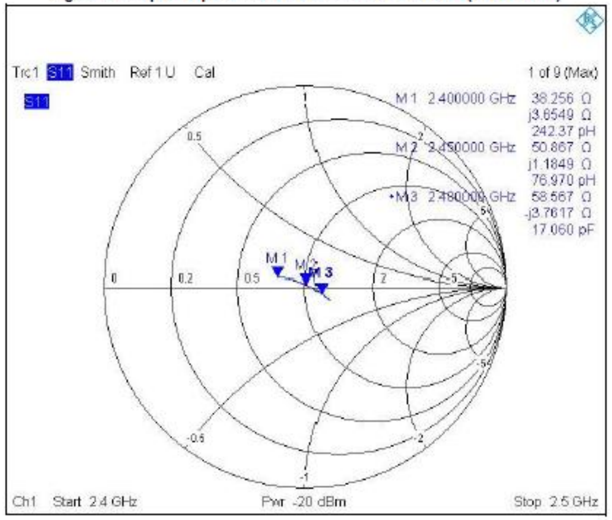


Figure 6. Complex impedance of the meander-like antenna (Smith Chart)



AN5246 details SMPS use cases, component selection, and various typical operating parametrics

AN5246
Application note

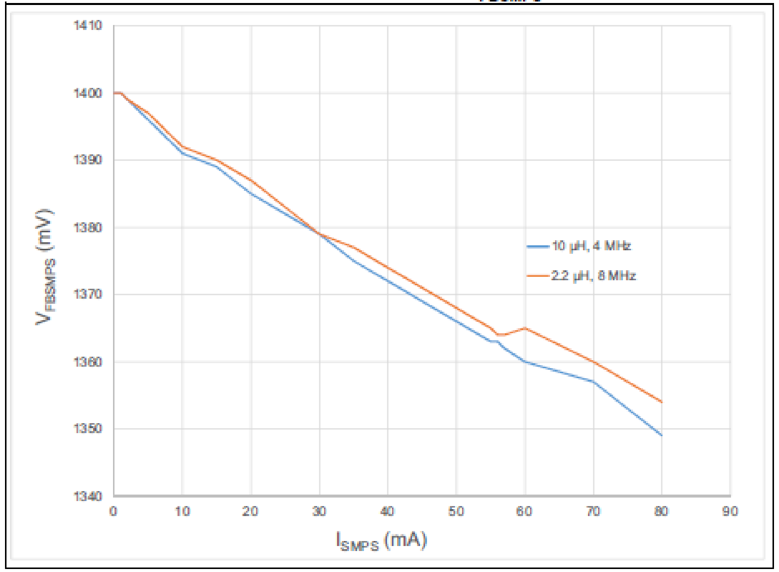
Usage of SMPS on STM32WB Series microcontrollers

Introduction

This document describes how the use the SMPS (switched mode power supply) integrated in microcontrollers of the STM32WB Series. It is intended to be used by system architects and by HW and board-level SW developers.

The patented implementation detailed in this document differs from the standard ones because it is able to maintain the RF transceiver full performance while, at the same time, providing the best power figure in burst application like those generally used by Bluetooth® Low Energy and IEEE 802.15.4 protocols.

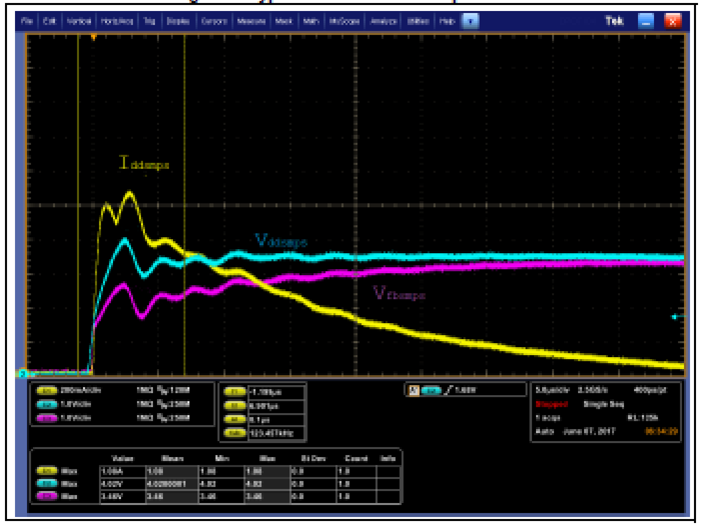
Figure 2. Load impact on V_{FBSMPS}



Inrush current at power ON

As the SMPS starts in BYPASS mode when powering up, the bulk capacitance needs to be powered when V_{DD} rises. At start up, when the V_{DD} voltage enters the 0.7 to 1 V range, the SMPS PMOS starts to conduce and $V_{FBDSMPS}$ follows V_{DDSMPS}. This leads to a temporary inrush current that can be as high as 1.1 A if the power supply is strong enough.

Figure 4. Typical inrush current at power-on



AN5071 details the multitude of low-power options available on the WB

AN5071
Application note
STM32WB ultra-low-power features overview

Figure 5. Low-power modes possible transitions

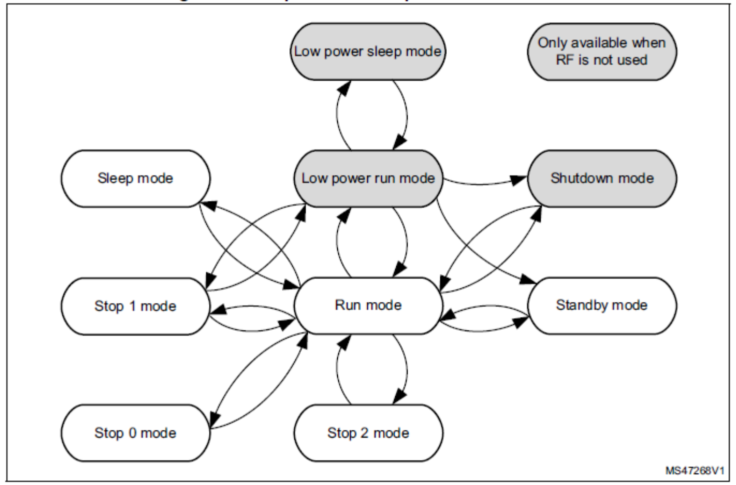


Figure 3. STM32WB55 - Current consumption for different memory configurations

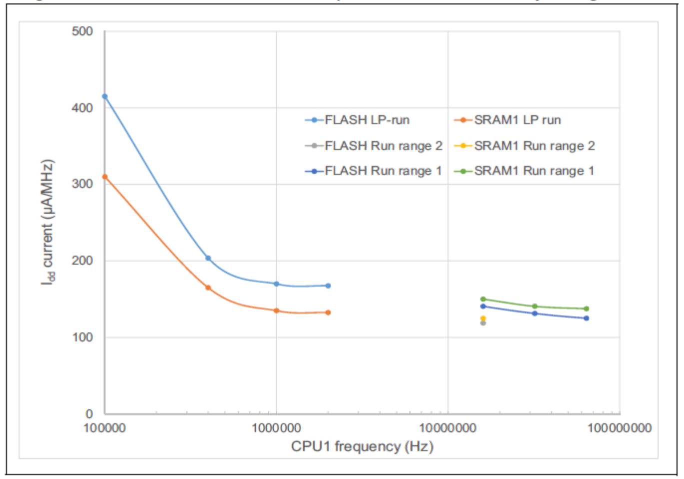


Figure 2. Power distribution architecture

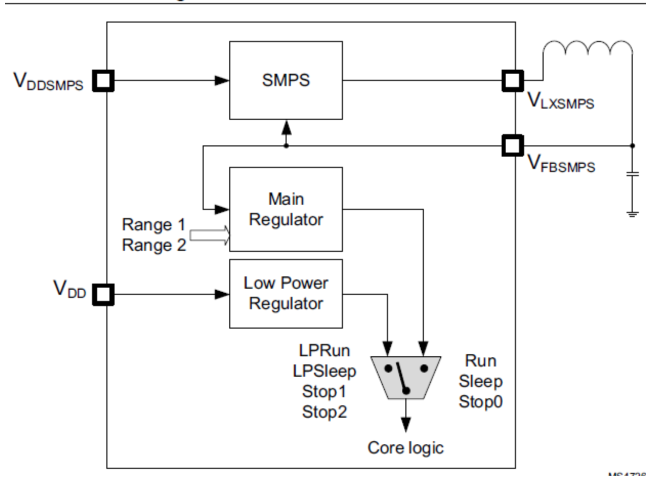


Table 2. STM32WB55 performance with SMPS

Configuration	mA/MHz	CoreMark® per MHz	CoreMark® per mA
FLASH ART On	0.077	3.25	42
SRAM1	0.073	2.40	33

life.augmented

STM32 WB

AN5155 is an exhaustive list of all firmware examples and descriptions

Thread®	Thread_Cli_Cmd	How to control the Thread® stack via Cli commands.	X	CubeMx
	Thread_Coap_DataTransfer	How to transfer large blocks of data through the CoAP messaging protocol.	X	X
	Thread_Coap_Generic	How to build Thread® application based on Coap messages.	X	CubeMx
	Thread_Coap_MultiBoard	How to use Coap for sending message to multiple boards.	-	X
	Thread_Commissioning	How to use Thread® commissioning process.	-	X
	Thread_FTD_Coap_Multicast	How to exchange multicast Coap messages.	X	X
	Thread_SED_Coap_Multicast	How to exchange a Coap message using the Thread® protocol.	X	X

Module Name	Project Name	Description	P-NUCLEO-WB55.USBDongle	P-NUCLEO-WB55.Nucleo
ADC	ADC_AnalogWatchdog_Init	How to use an ADC peripheral with an ADC analog watchdog to monitor a channel and detect when the corresponding conversion data is outside the window thresholds.	-	CubeMx
	ADC_ContinuousConversion_TriggerSW	How to use an ADC peripheral to perform continuous ADC conversions on a channel, from a software start.	-	X
	ADC_ContinuousConversion_TriggerSW_Init	How to use an ADC peripheral to perform continuous ADC conversions on a channel, from a software start.	-	CubeMx
	ADC_ContinuousConversion_TriggerSW_LowPower_Init	How to use an ADC peripheral with ADC low-power features.	-	CubeMx
	ADC_GroupsRegularInjected_Init	How to use an ADC peripheral with both ADC groups (regular and injected) in their intended use cases.	-	CubeMx
	ADC_Oversampling_Init	How to use an ADC peripheral with ADC oversampling.	-	CubeMx
	ADC_SingleConversion_TriggerSW_DMA_Init	How to use an ADC peripheral to perform a single ADC conversion on a channel, at each software start. This example uses the DMA programming model (for polling or interrupt programming models, refer to other examples).	-	CubeMx
	ADC_SingleConversion_TriggerSW_IT_Init	How to use an ADC peripheral to perform a single ADC conversion on a channel, at each software start. This example uses the interrupt programming model (for polling or DMA programming models, please refer to other examples).	-	CubeMx
	ADC_SingleConversion_TriggerSW_Init	How to use an ADC peripheral to perform a single ADC conversion on a channel at each software start. This example uses the polling programming model (for interrupt or DMA programming models, please refer to other examples).	-	CubeMx
	ADC_SingleConversion_TriggerTimer_DMA_Init	How to use an ADC peripheral to perform a single ADC conversion on a channel at each trigger event from a timer. Converted data are indefinitely transferred by DMA into a table (circular mode).	-	CubeMx
	ADC_TemperatureSensor	How to use an ADC peripheral to perform a single ADC conversion on the internal temperature sensor and calculate the temperature in Celsius degrees.	-	X

CubeMx denotes that there is an “ioc” CubeMX project file also

AN5292 shows how to get started using BLE Mesh

Figure 7. Internal project folder

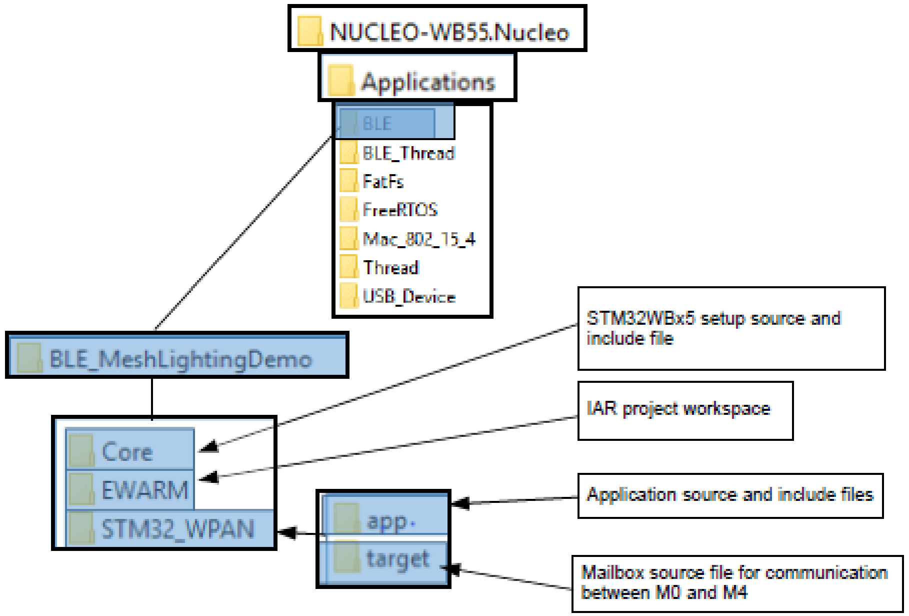
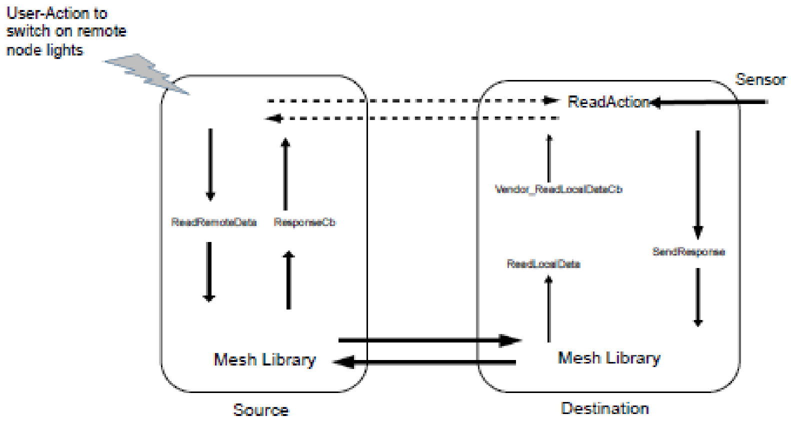
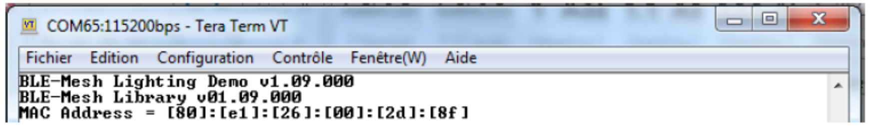


Figure 14. Read command from a remote node



The response data from the node is sent via the `BLEMesh_SendResponse` function.

Figure 10. VCOM window



MAC address management

Each node in the mesh network requise a unic MAC address. The following table describes the available options to configure the MAC addresses for a node

Table 2. MAC address management

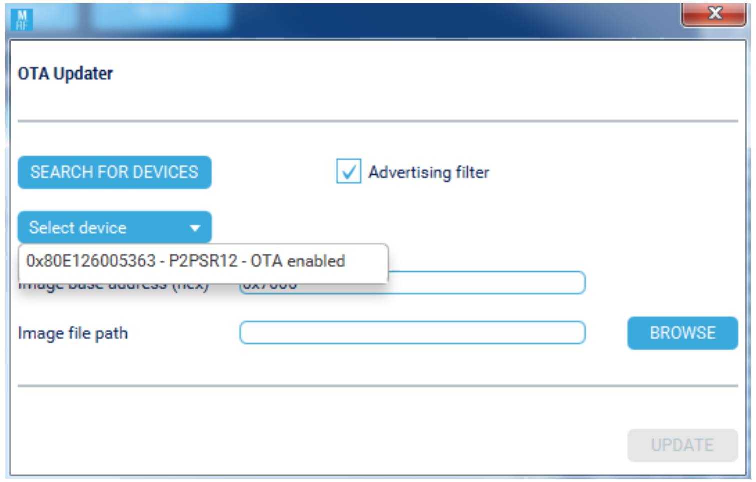
Number	MAC address Management	Comments
1	Using external MAC address	User can program the nodes with desired unique MAC address. This is stored at specific location in the flash. It is the user's responsibility to make sure that the programmed MAC address in the device is compliant with the Bluetooth communication requirements.
2	Using the unique device serial number	It is possible to configure the MAC address of the device using the unique serial number available in each device. This is the default setting.
3	Using static random MAC address	It is possible to configure the MAC address of devices using the static random MAC address

life.augmented

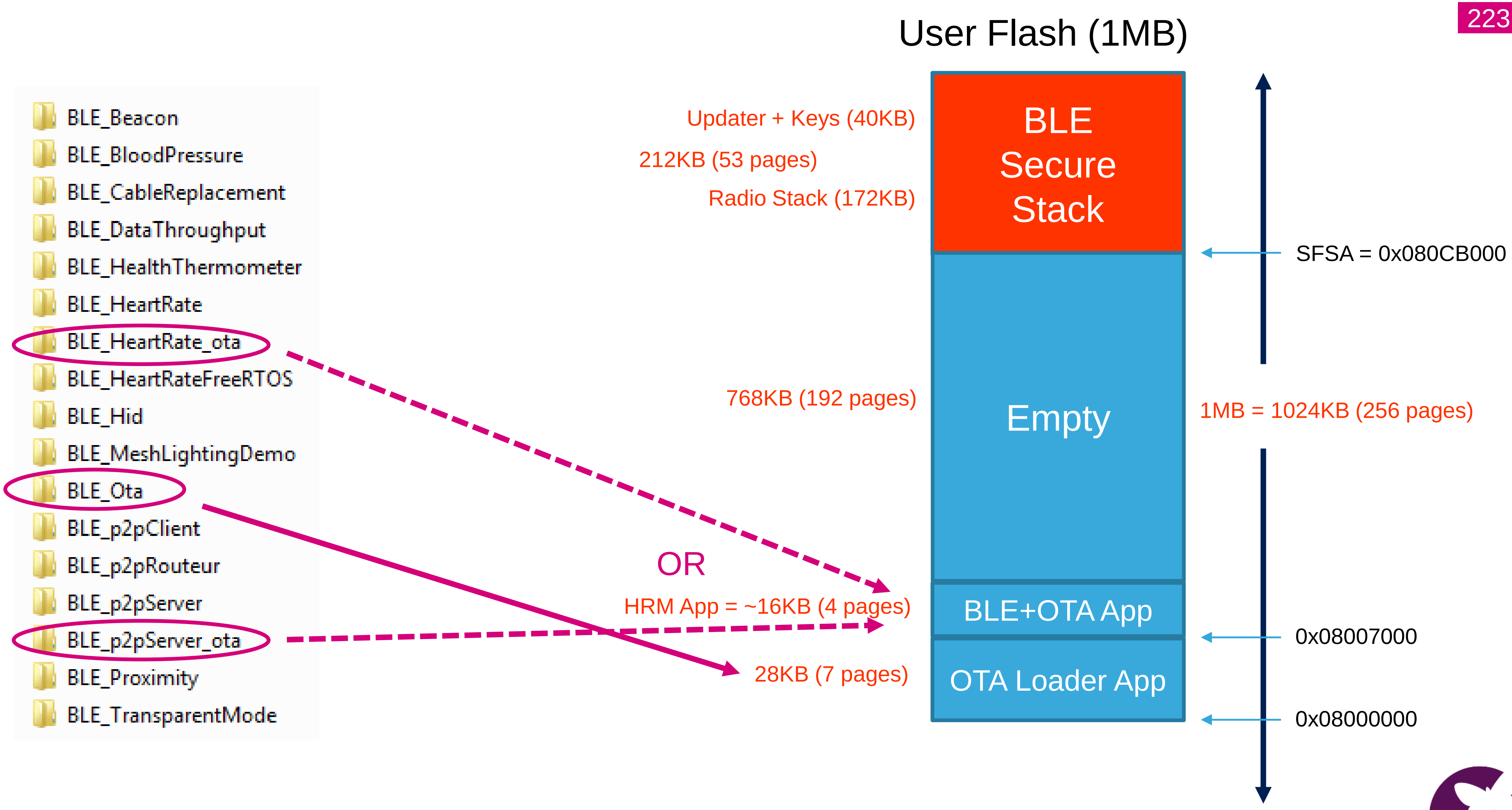
STM32 WB

Hands-On

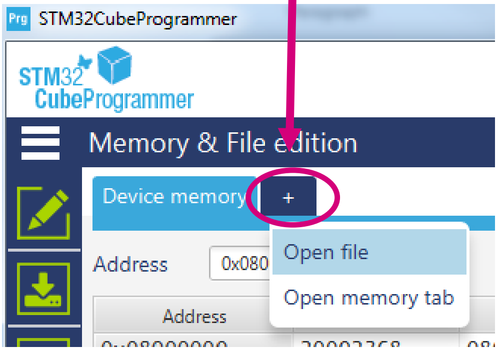
Over-the-Air Firmware Update

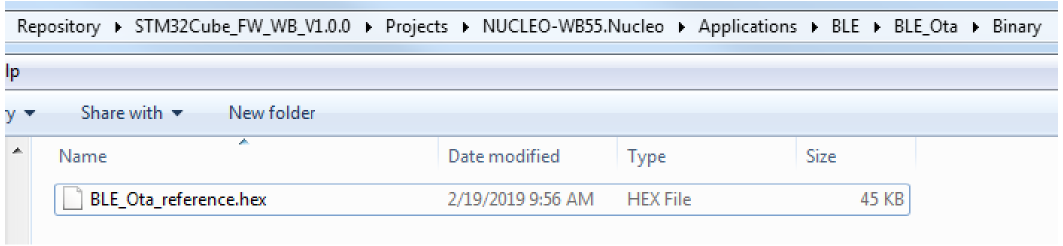


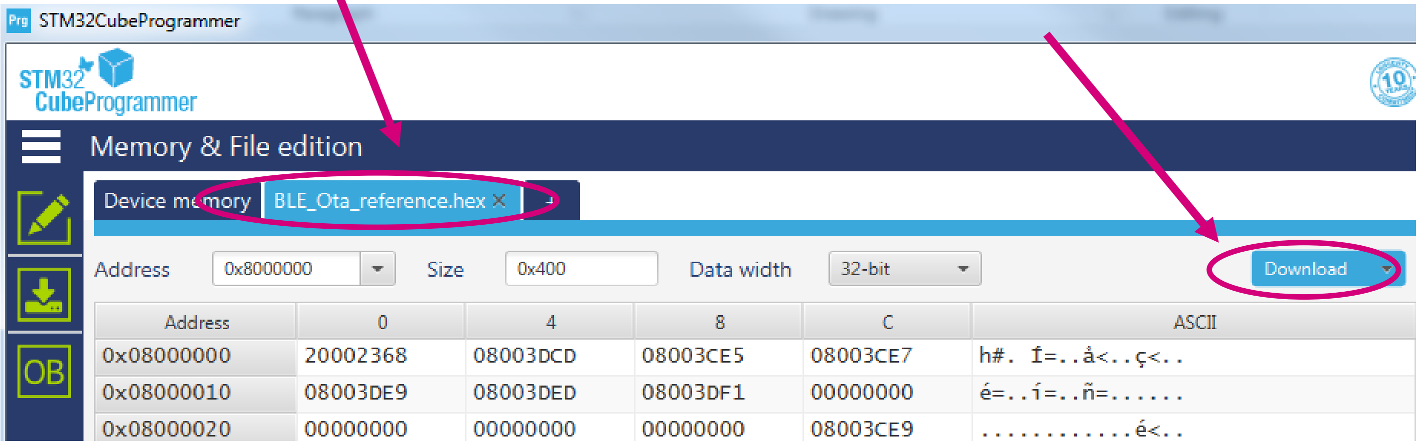
Over-The-Air Firmware Updates



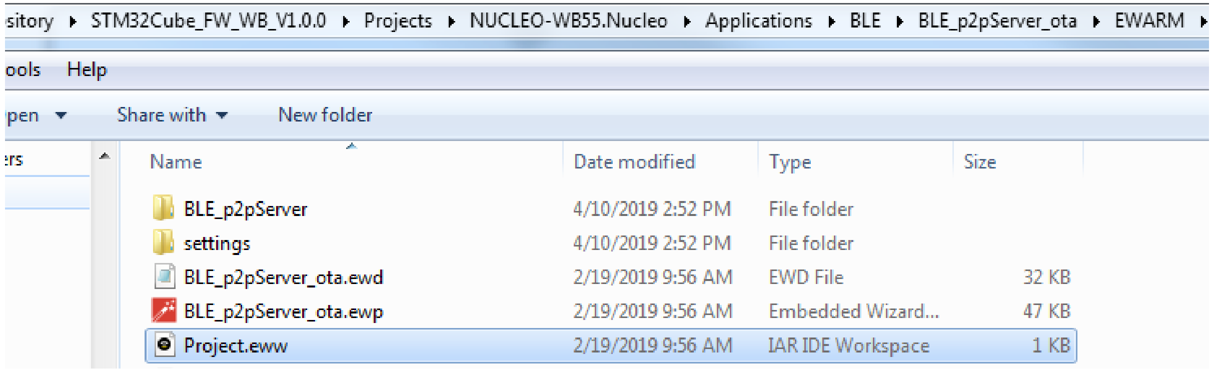
Flash Nucleo board with **BLE_Ota_reference.hex** using CubeProgrammer







Load and personalize your **BLE_p2pServer_ota.eww** project



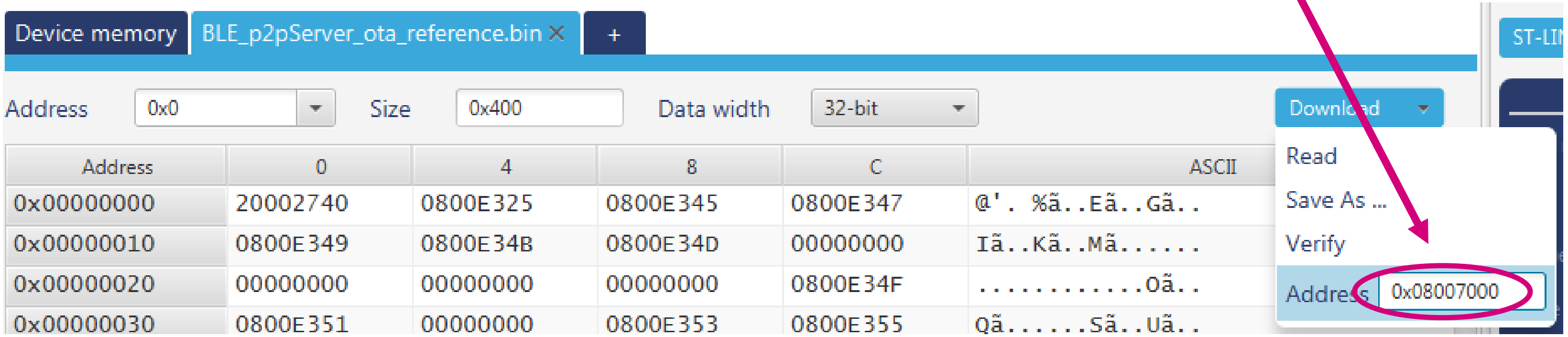
In **app_ble.c**

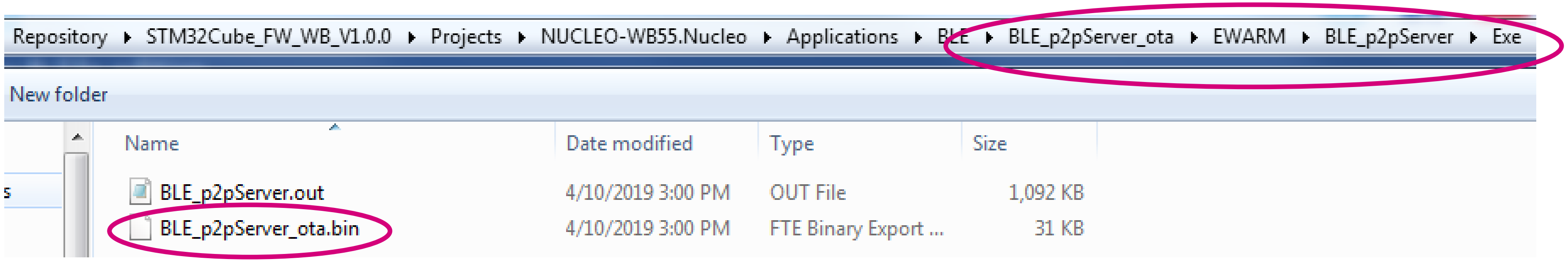
```
#if (P2P_SERVER1 != 0)
static const char local_name[] = { AD_TYPE_COMPLETE_LOCAL_NAME ,'P','2','P','S','R','1','2'};
uint8_t manuf_data[14] = {
```

```
#define APPBLE_GAP_DEVICE_NAME_LENGTH 9
```

```
const char *name = "STM32WB12";
```

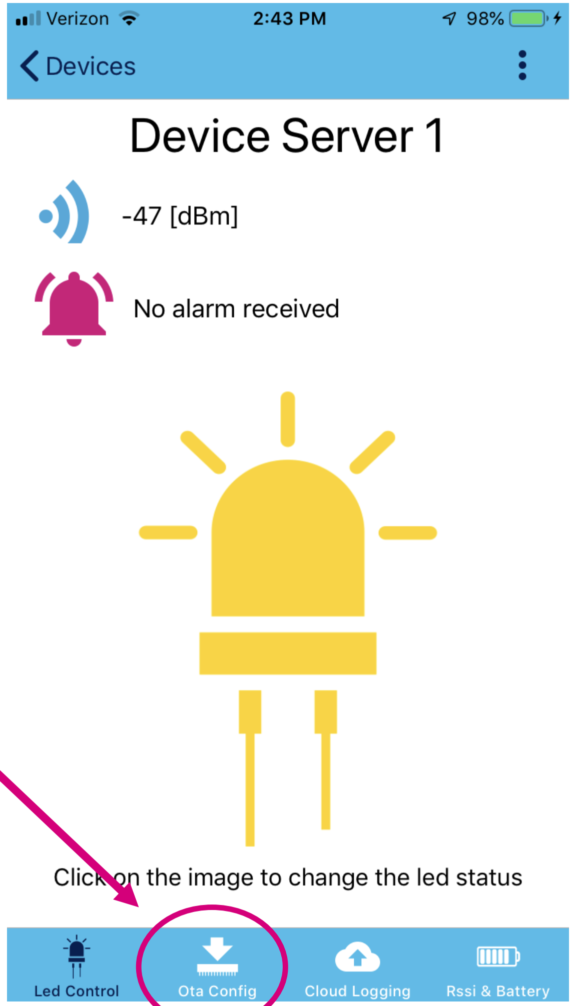
Flash your newly created **BLE_p2pServer_ota.bin** to 0x08007000





Verify functionality on the ST BLE Sensor app

You should also see OTA capability



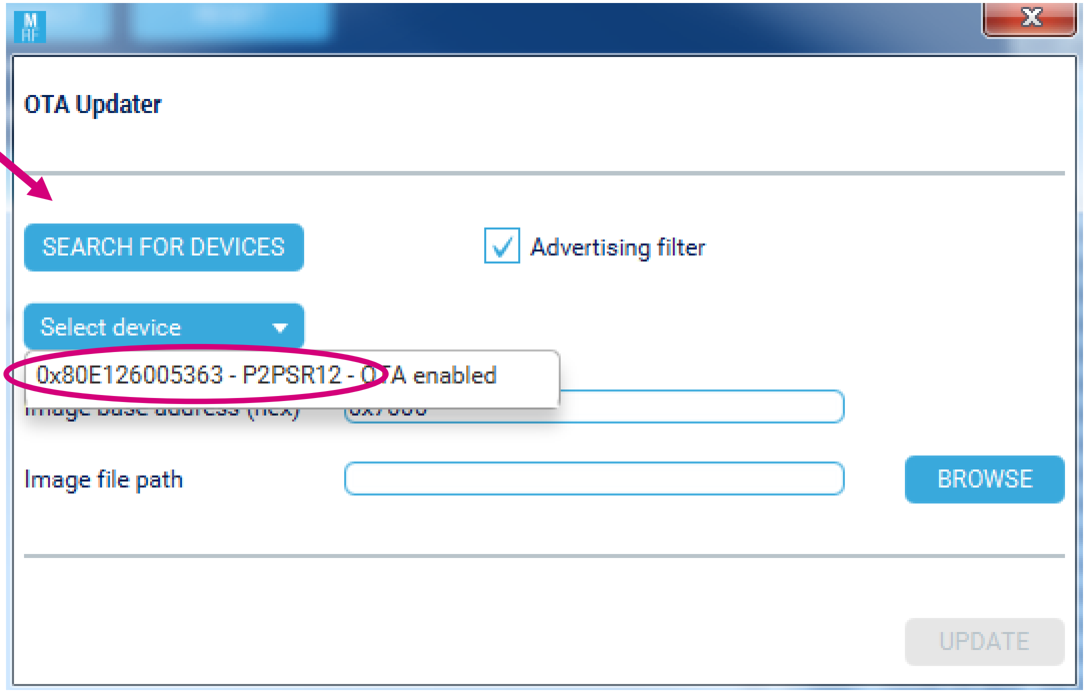
Once seen, disconnect from your device

Connect to OTA-enabled device

Connect the Dongle and select **OTA Updater**

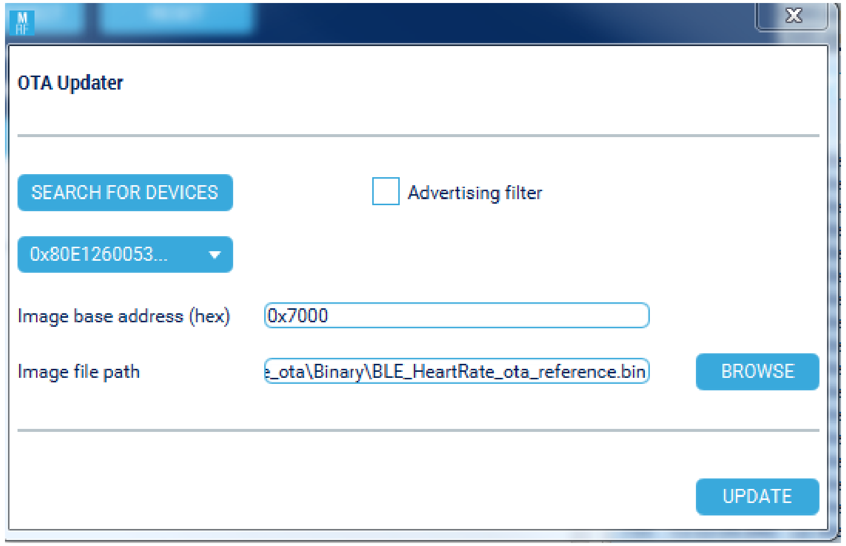
Search and Select your Device

(you can see your local name & BLE Address)

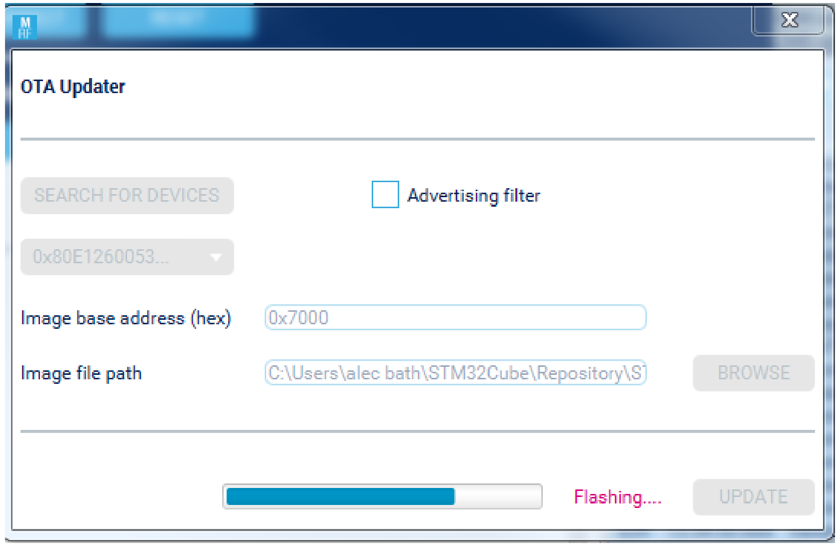


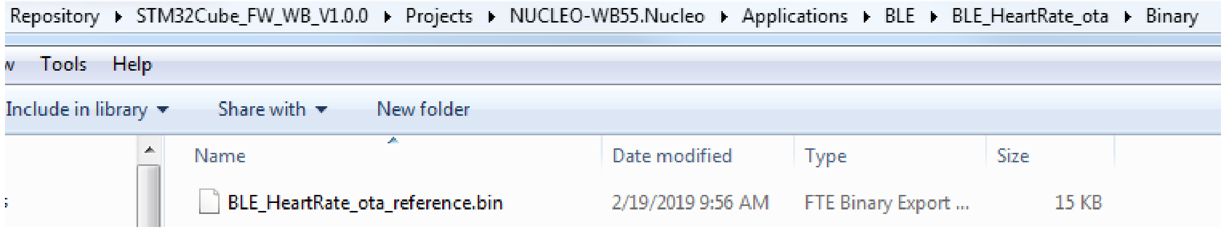
Browse for the other OTA binary

- BLE_HeartRate_ota_reference.bin



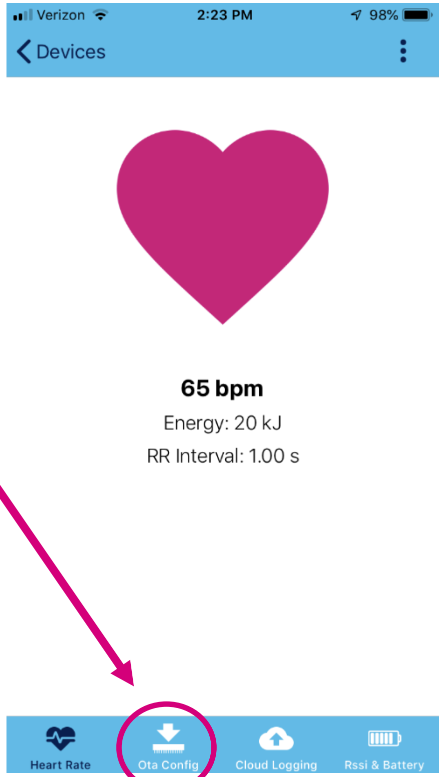
Update image





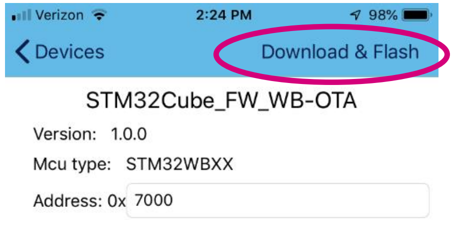
Update via ST BLE Sensor app

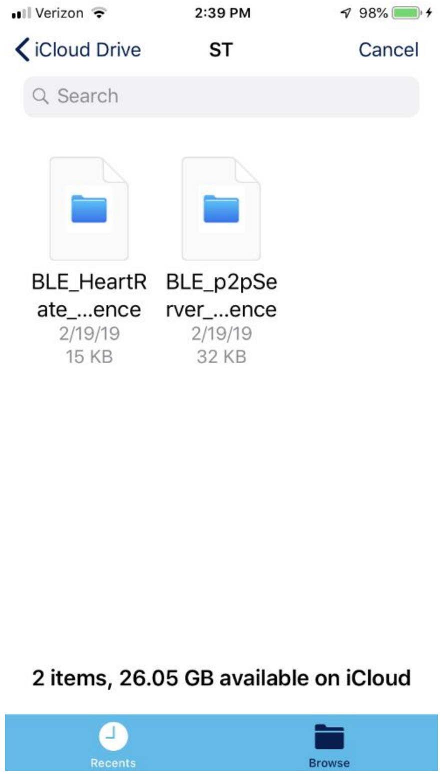
OTA capability detected
Click to start



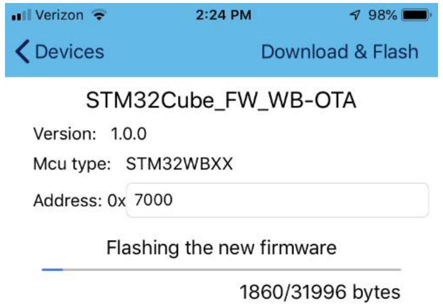
Erase & Reboot

Select Smartphone file (iCloud, etc)





Flashing begins



life.augmented

STM32 WB

AN5247 details the OTA application in further detail.

AN5247
Application note

Over-the-air application and wireless firmware update for STM32WB Series microcontrollers

Introduction

This document describes the procedure for over-the-air (OTA) firmware update on ST32WB devices with BLE connection. It explains how to use the OTA application provided within the STM32Cube firmware package.

This application can update both the user application and the wireless firmware.

Figure 1. STM32WB dual core FW architecture

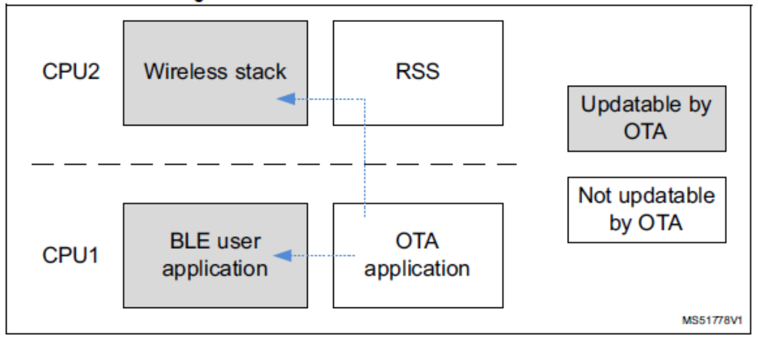
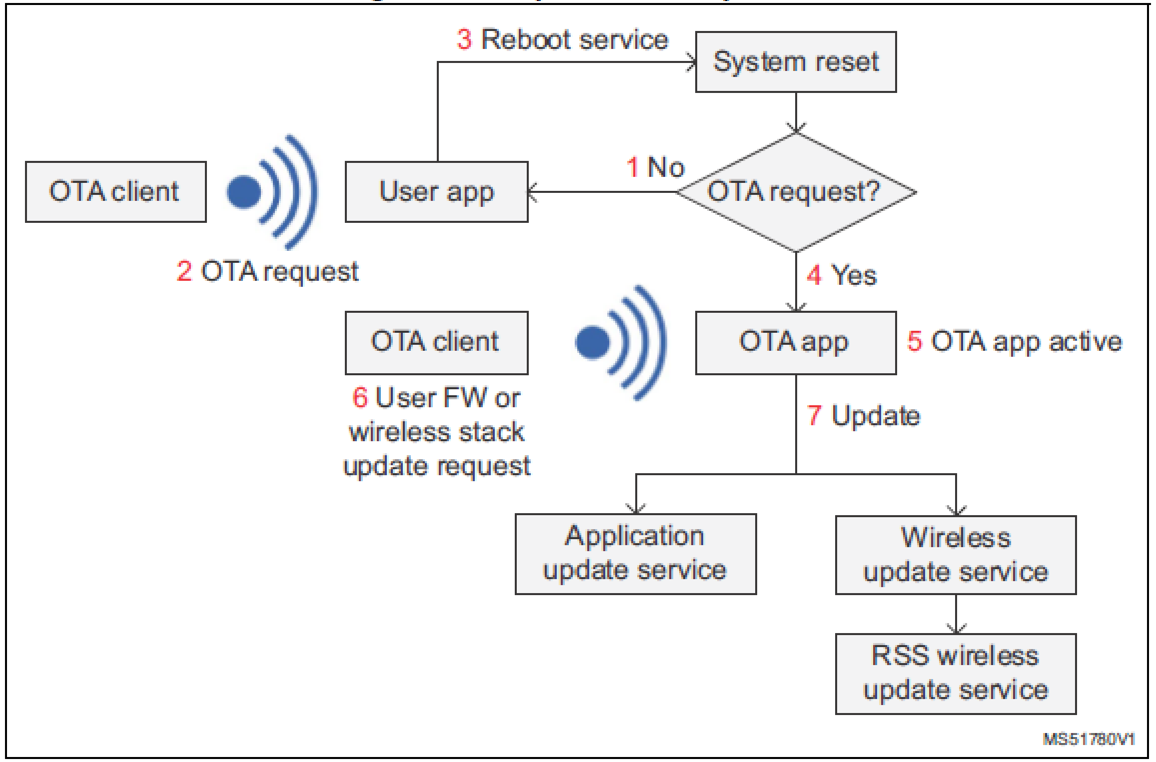
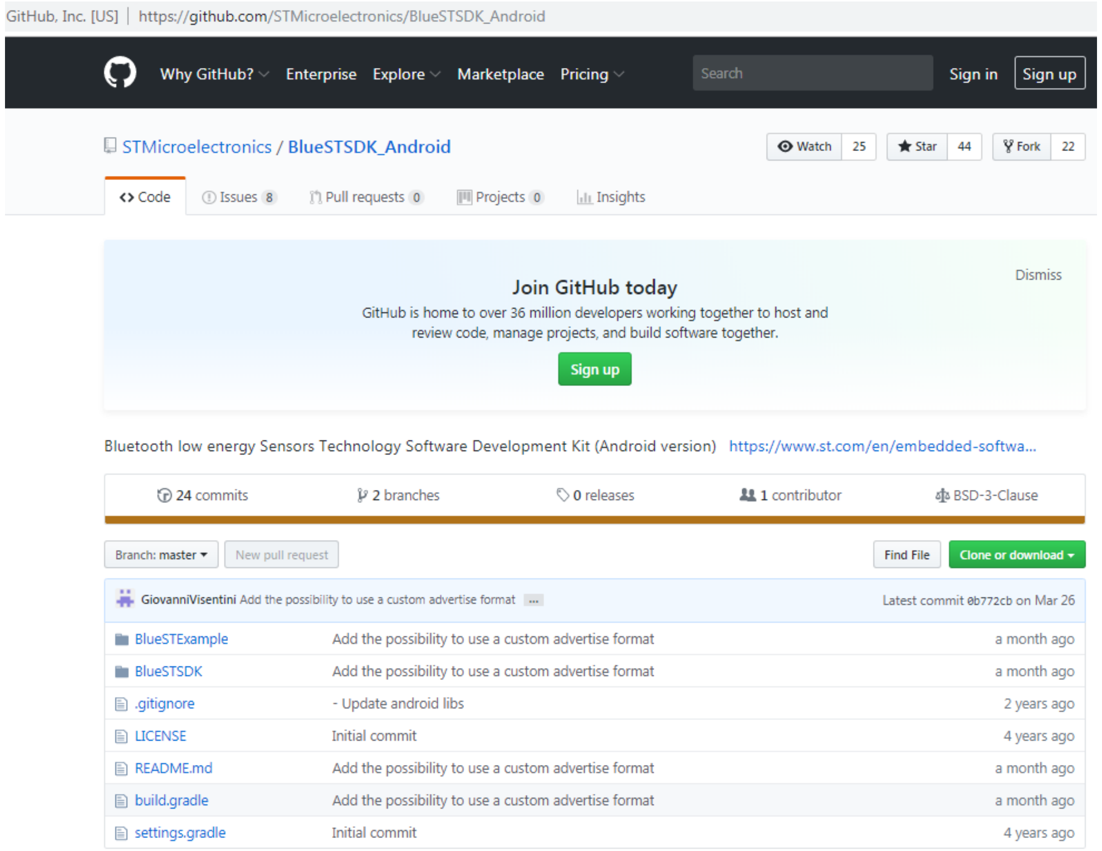
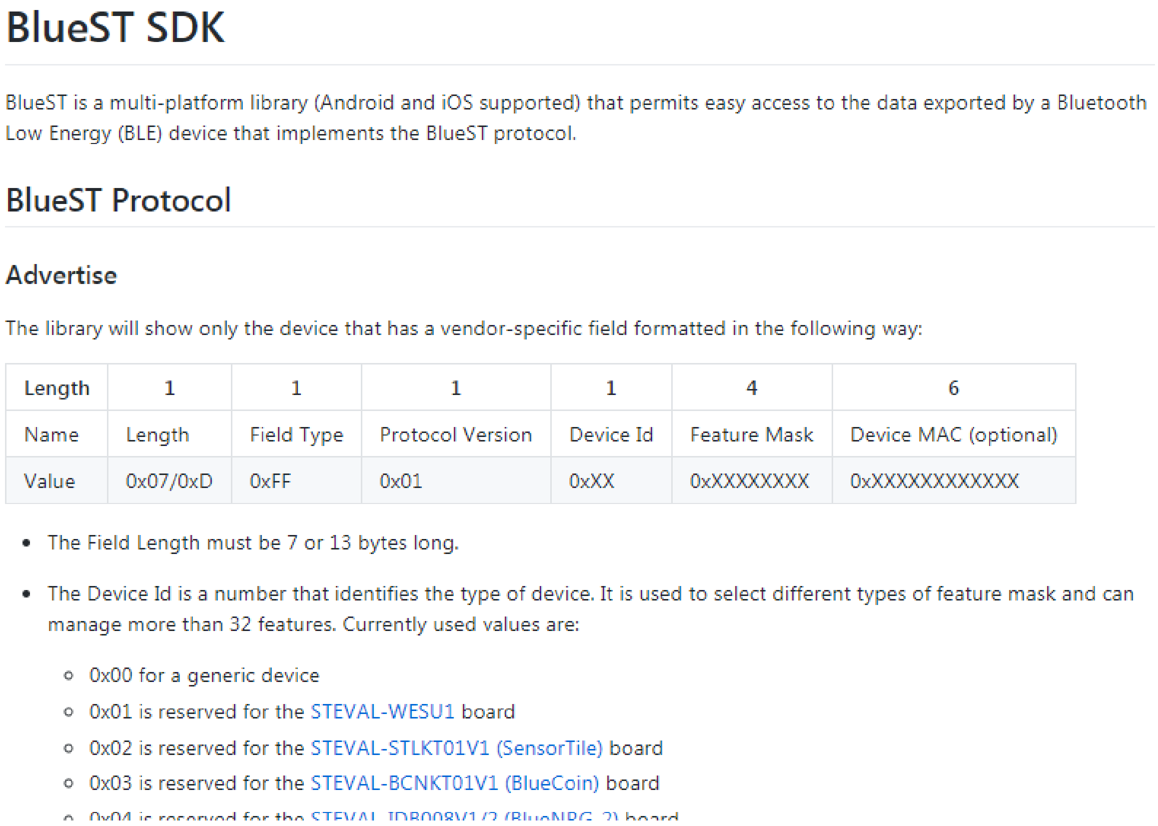


Figure 3. OTA procedure sequence



BlueST SDK on Github: https://github.com/STMicroelectronics/BlueSTSDK_Android





BlueST SDK

BlueST is a multi-platform library (Android and iOS supported) that permits easy access to the data exported by a Bluetooth Low Energy (BLE) device that implements the BlueST protocol.

BlueST Protocol

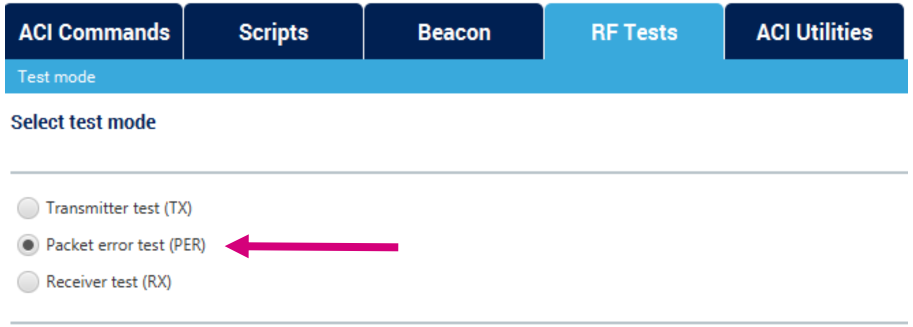
Advertise

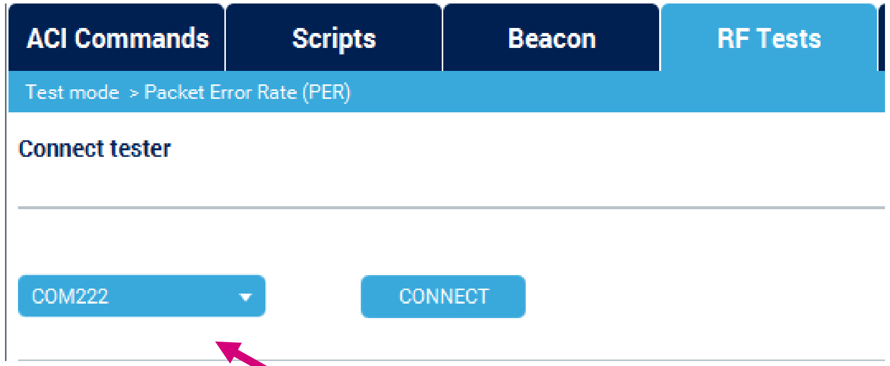
The library will show only the device that has a vendor-specific field formatted in the following way:

Length	1	1	1	1	4	6
Name	Length	Field Type	Protocol Version	Device Id	Feature Mask	Device MAC (optional)
Value	0x07/0xD	0xFF	0x01	0xXX	0xXXXXXXXX	0xXXXXXXXXXXXX

- The Field Length must be 7 or 13 bytes long.
- The Device Id is a number that identifies the type of device. It is used to select different types of feature mask and can manage more than 32 features. Currently used values are:
  - 0x00 for a generic device
  - 0x01 is reserved for the STEVAL-WESU1 board
  - 0x02 is reserved for the STEVAL-STLKT01V1 (SensorTile) board
  - 0x03 is reserved for the STEVAL-BCNKT01V1 (BlueCoin) board
  - 0x04 is reserved for the STEVAL-IDB008V1/2 (BlueNRG-2) board

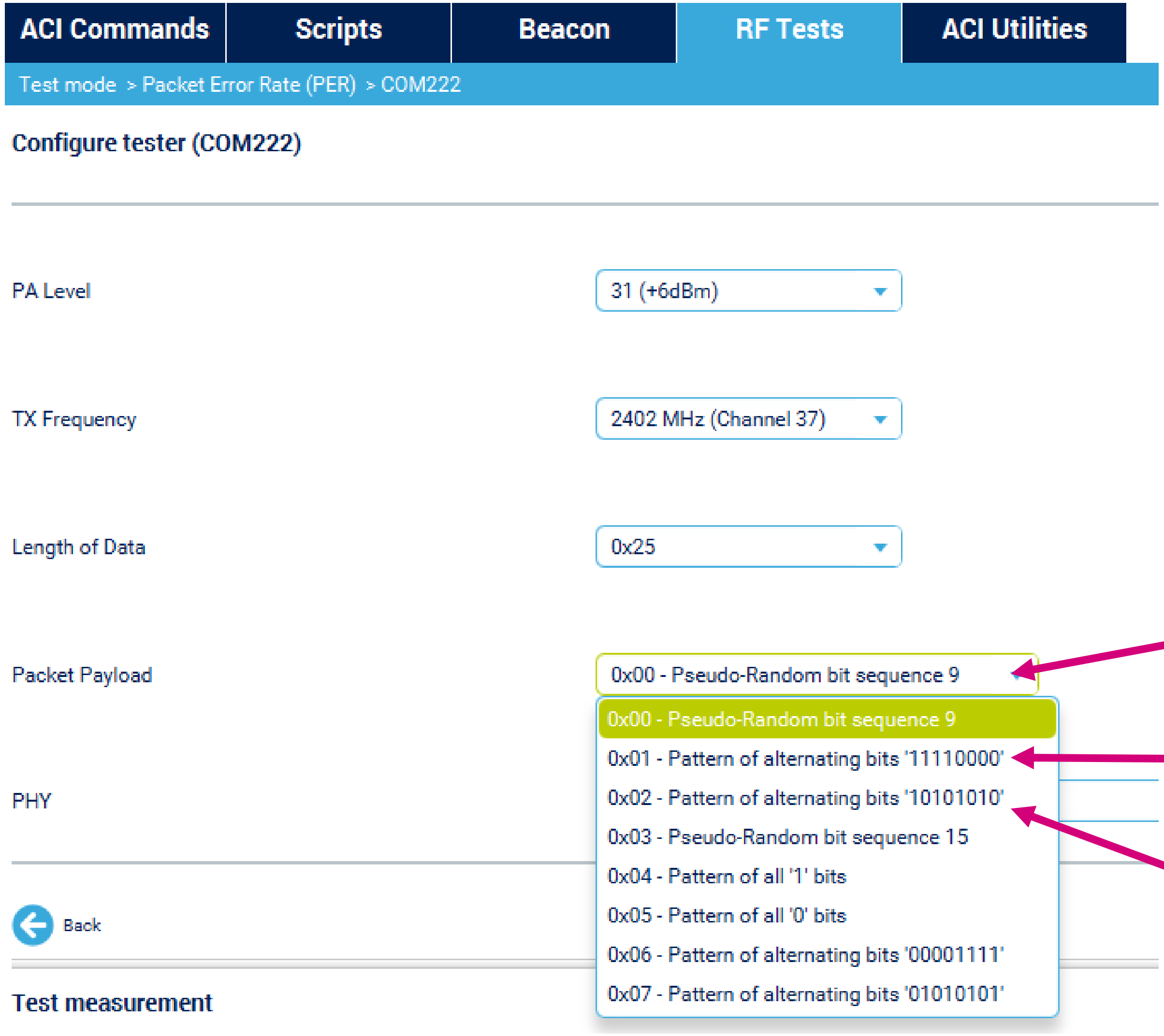
Packet Error Rate (PER) tests with two boards connected





Connect a second board
(Dongle for instance)

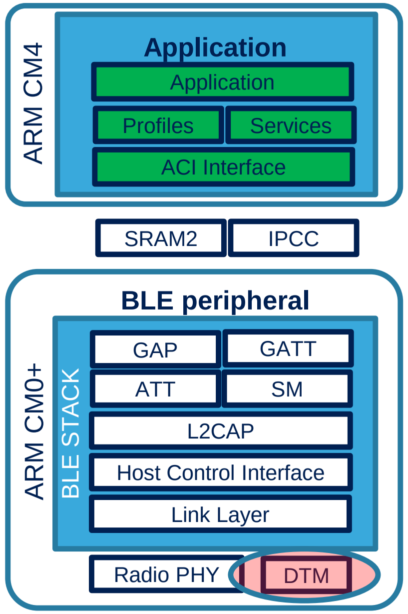
Packet whitening is disabled in Direct Test Mode (DTM)



Power Transmission testing

Frequency Deviation testing

Carrier Frequency Accuracy testing



life.augmented

STM32 WB

We Greatly Value Your Feedback

Use your phone to scan the QR code or type the link into your browser.

https://www.surveymonkey.com/r/PLYWMDC

Thank you!

Releasing Your Creativity

/STM32 @ST_World ST community

life.augmented

www.st.com/stm32wb