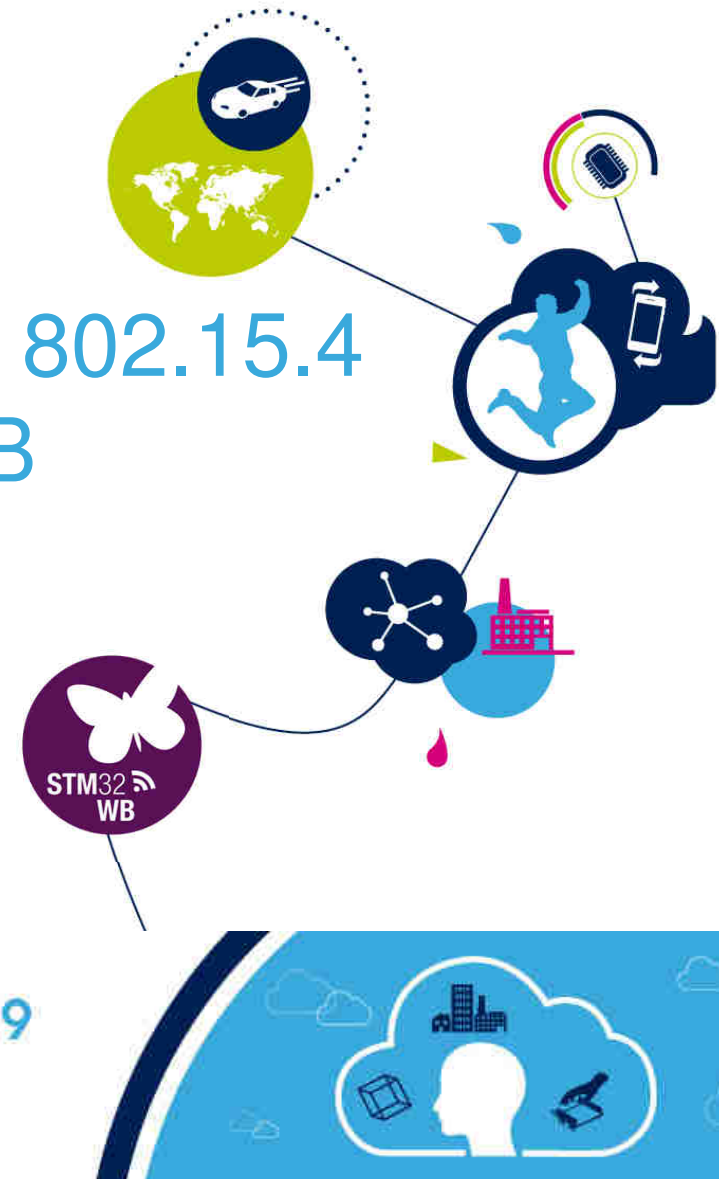


Bluetooth Low Energy 5.0 and 802.15.4 Made Easy with the STM32WB

Americas Marketing and Applications Team

Colin Ramrattan



Technology Tour 2019

Toronto, Canada | May 29

STM32WB Introduction

BLE Basics

Tools & Firmware

Lot's of Hands-On coding!



Agenda

3

Tools install

A few words..

Hands-On: CubeMX

BLE Basics

Hands-On: HRM

Break

Architecture

Hands-On: CubeMonitorRF

More WB Detail

Hardware & Software resources

Hands-On: Cable Replacement (time-permitting)

Wrap-up, Q&A

Prerequisites & Tools Install

4

- ❑ 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, v8.32.3 + License
- ❑ TeraTerm, or equiv.





[STM32WB Workshop - Installation Procedure from USB drive \(v1.0\)](#)

Welcome to the STM32WB Workshop!
Please follow all installation steps below on the day of the Workshop:

[STM32WB Workshop – Requirements](#)

Important:

You need to have administration rights on your Laptop Computer to be able to install drivers and software and also to do the workshop.

You need to bring an iOS or Android Smartphone/Tablet with Bluetooth 4.0 (BLE) support.

Note: Most modern iOS and Android devices, since 2012, support Bluetooth 4.0 (BLE) or later.

System requirements for Laptop Computer:

Windows® 7 and later or MacBook running Windows using Parallels, VM Fusion

Software requirements for Laptop Computer:

Install Java™ Run Time Environment for 1.8.0 or later. If Java™ is not installed on your computer, we are providing the installer for the latest Java on the USB drive.

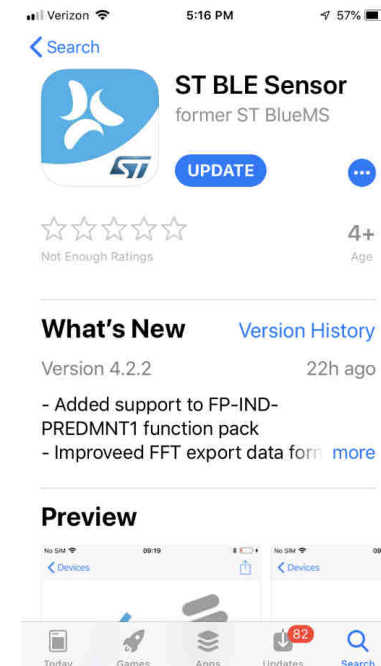
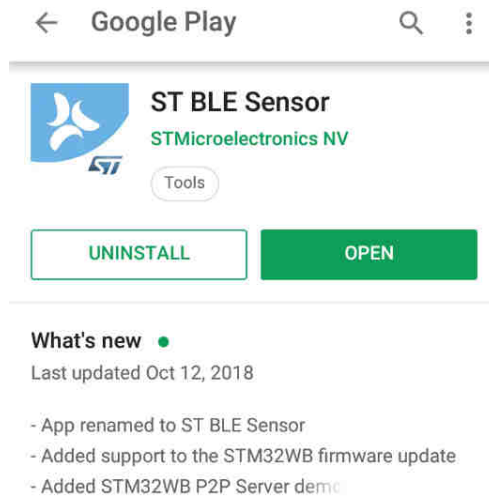
Minimum Recommended Hardware Requirements for Laptop Computer:

- Type A USB port
- 2+ GHz processor
- 4 GB of system memory (RAM)
- 10 GB of available disk space

Note: For machines with USB Type C, please bring a Type A to Type C adapter.

ST BLE Sensor app, v4.3

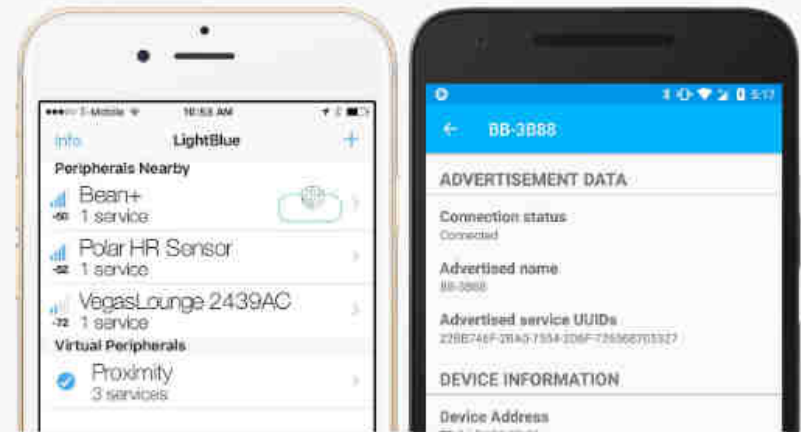
6

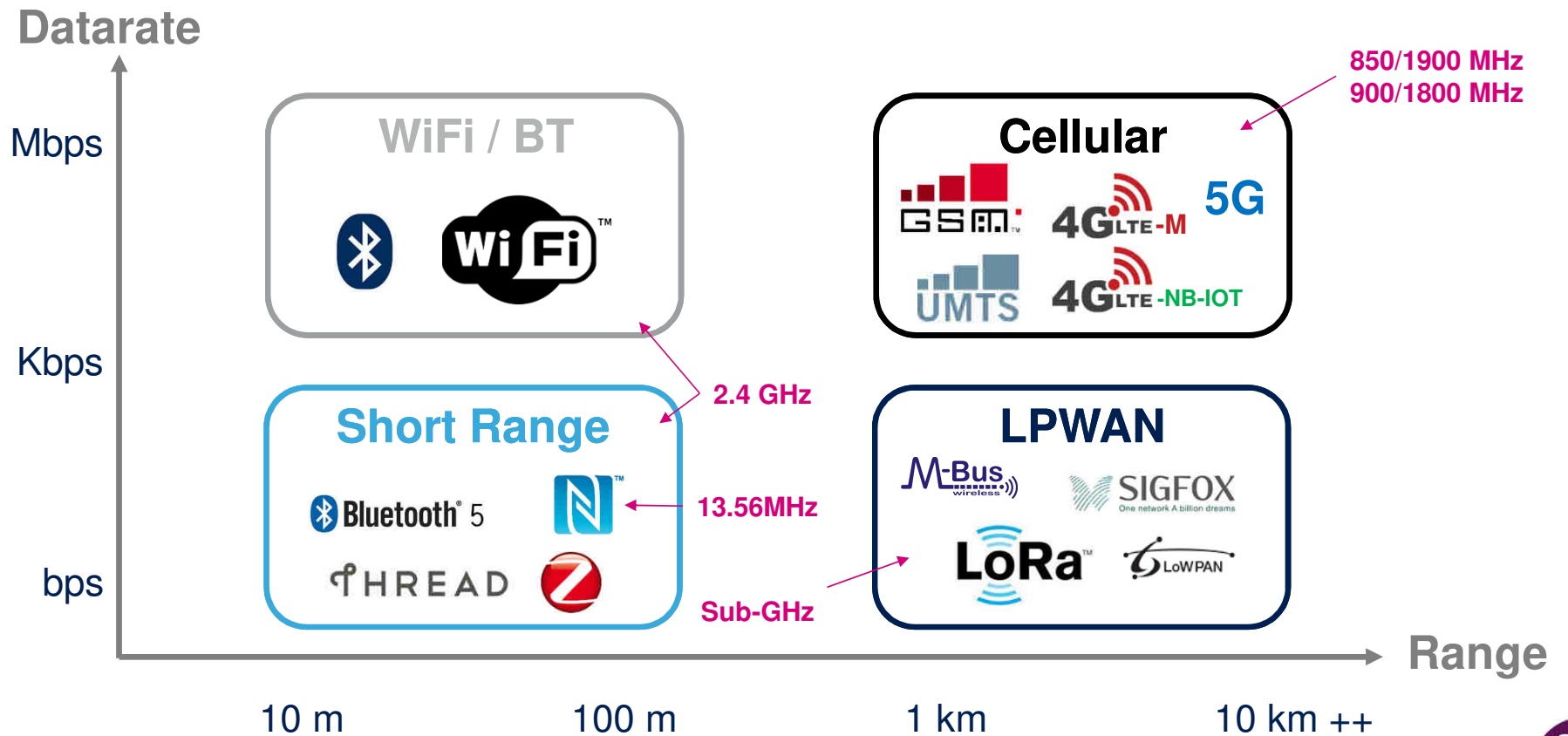




LightBlue | Explorer

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.





Low-data-rate 2.4GHz connectivity

9



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?

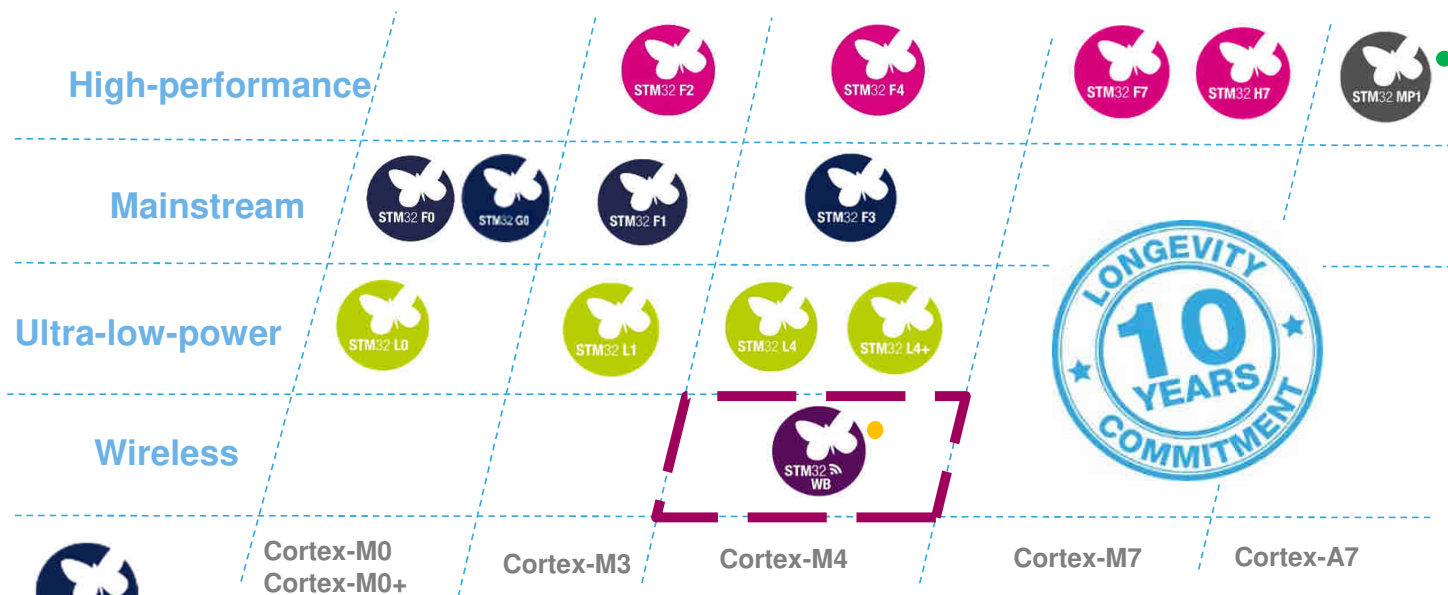
10



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

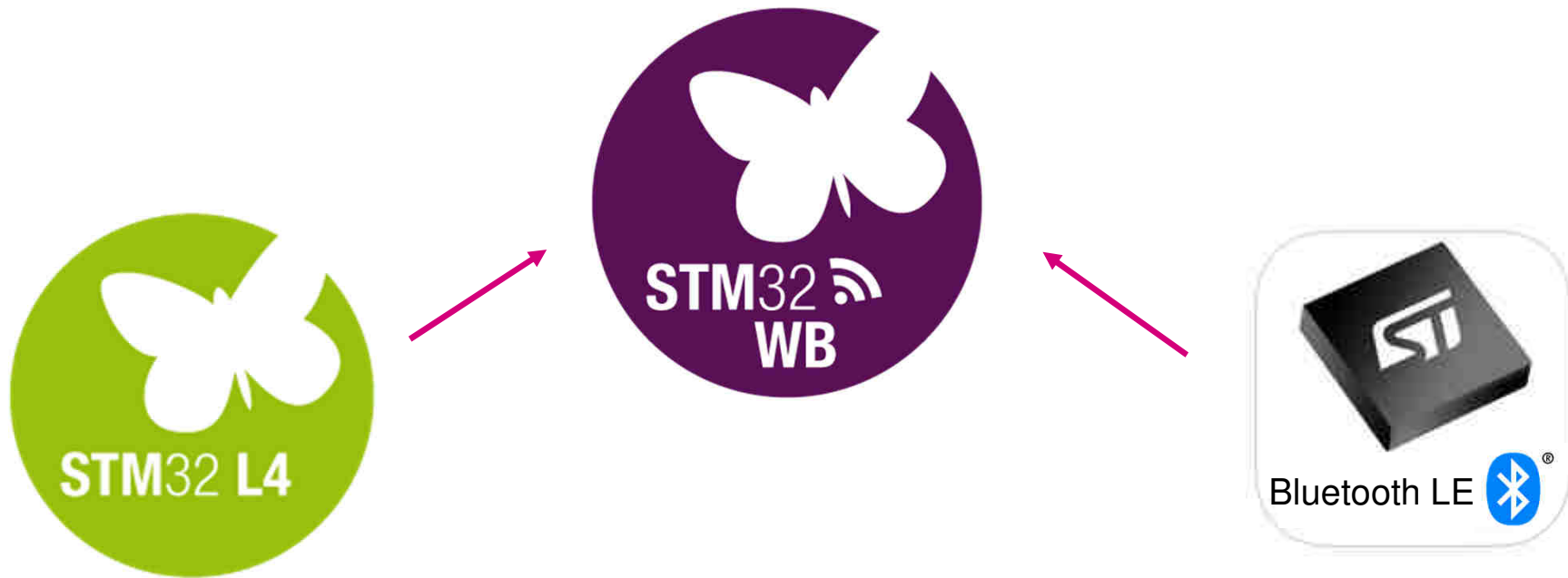
STM32 Portfolio

11



More than
40,000 customers





STM32WB Key Takeaways

13



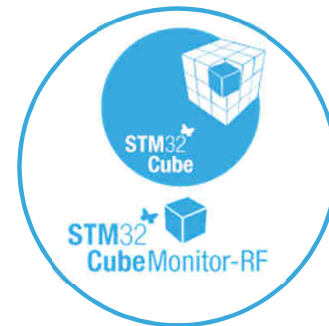
Multi-protocol



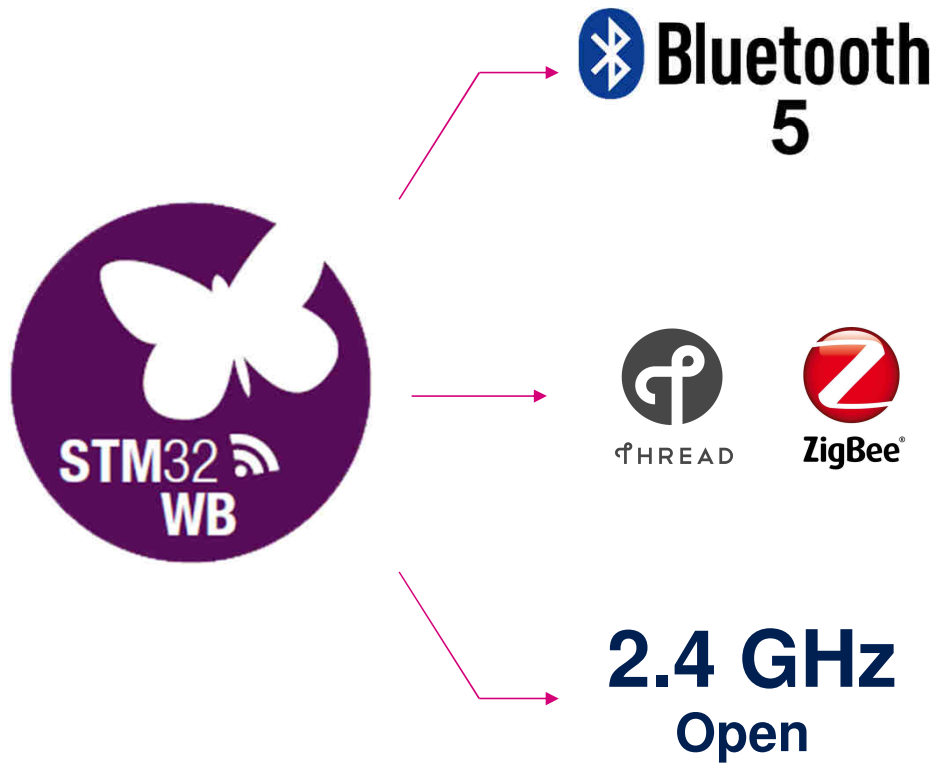
**Dual-core
Ultra-Low Power**

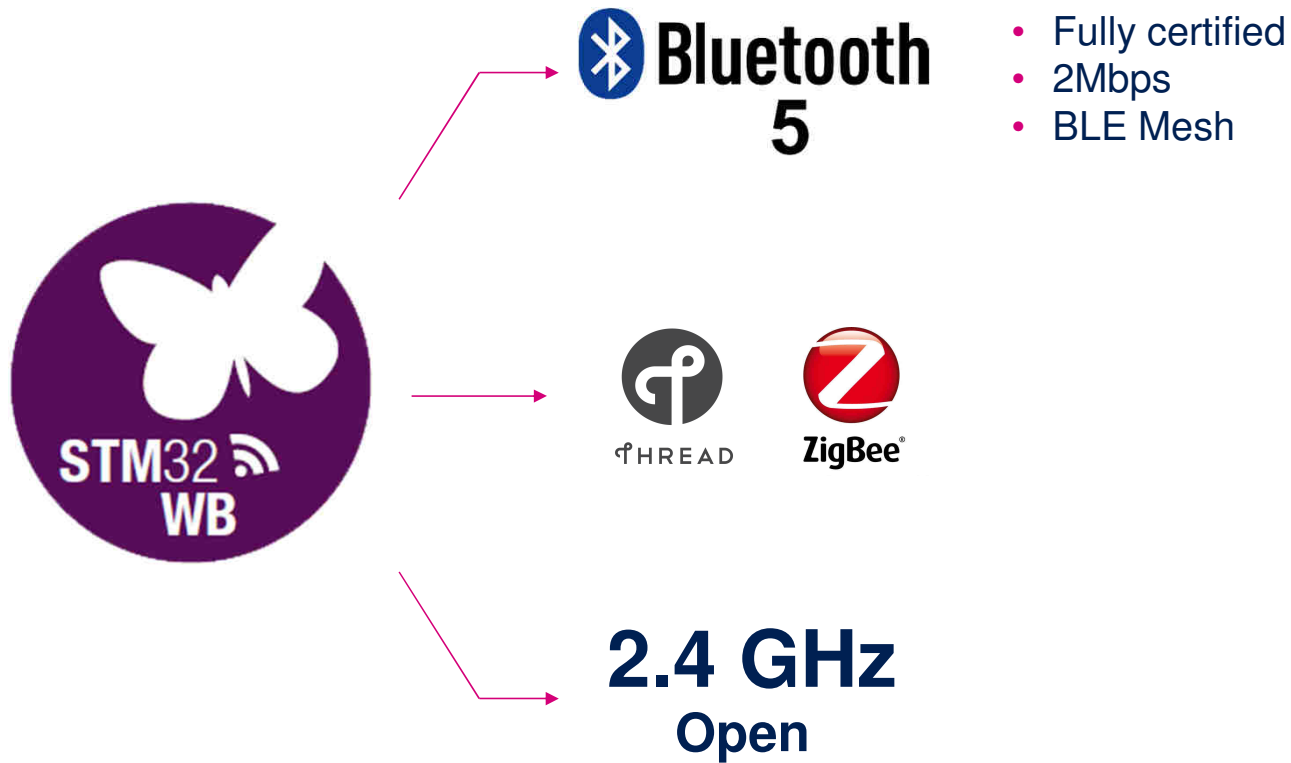


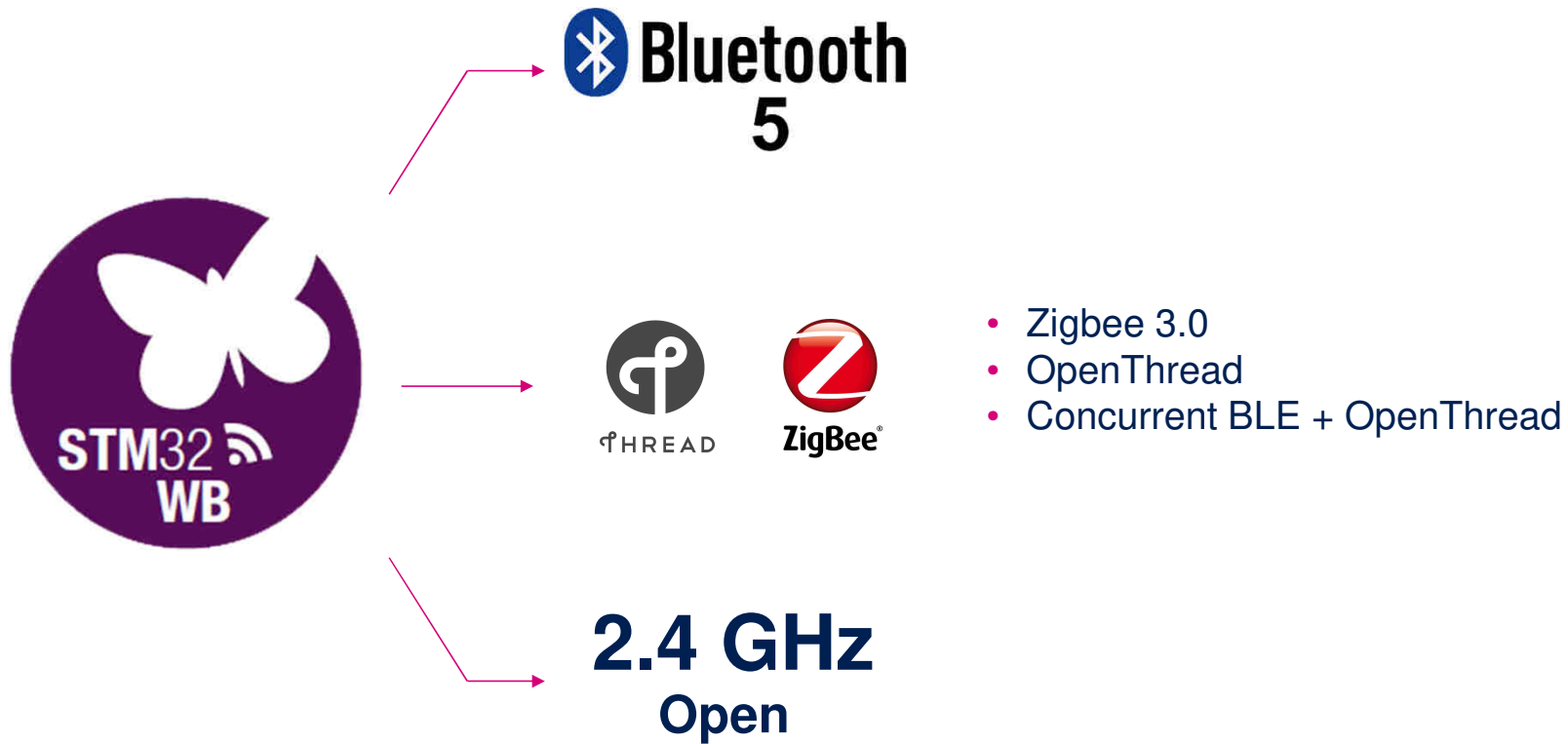
Secure

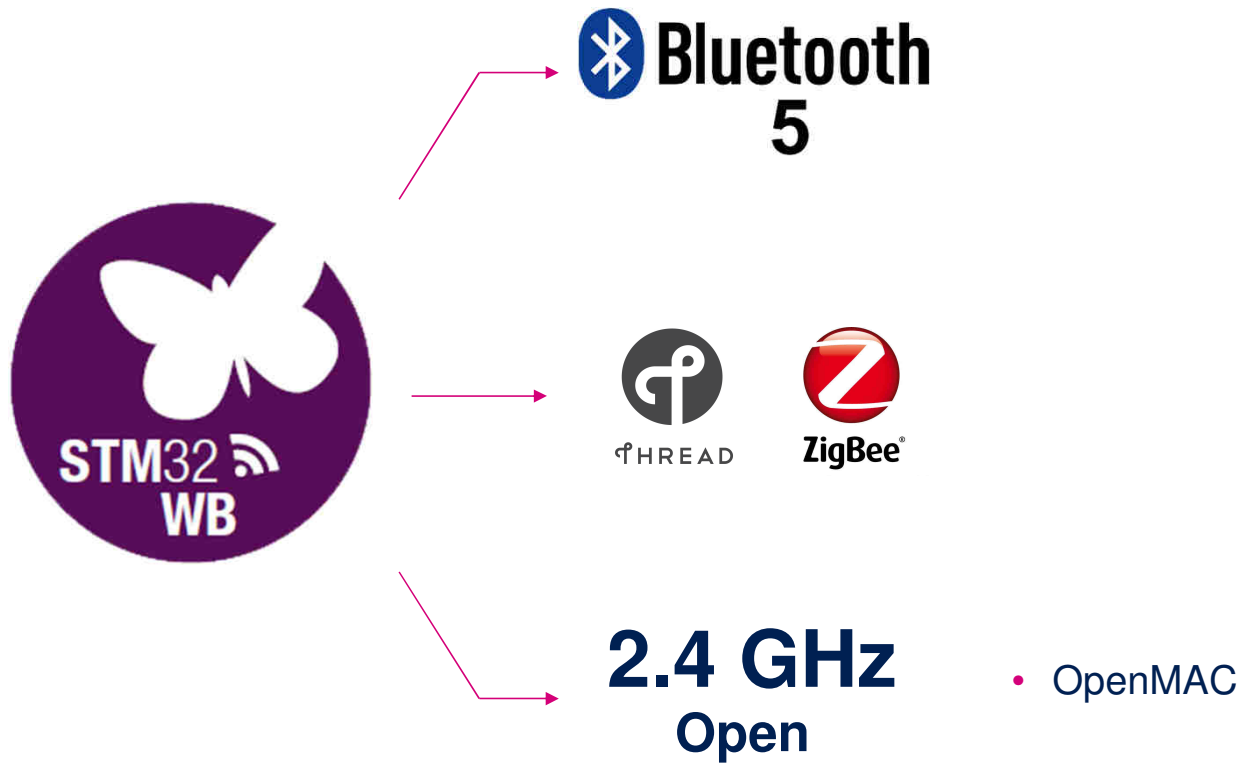


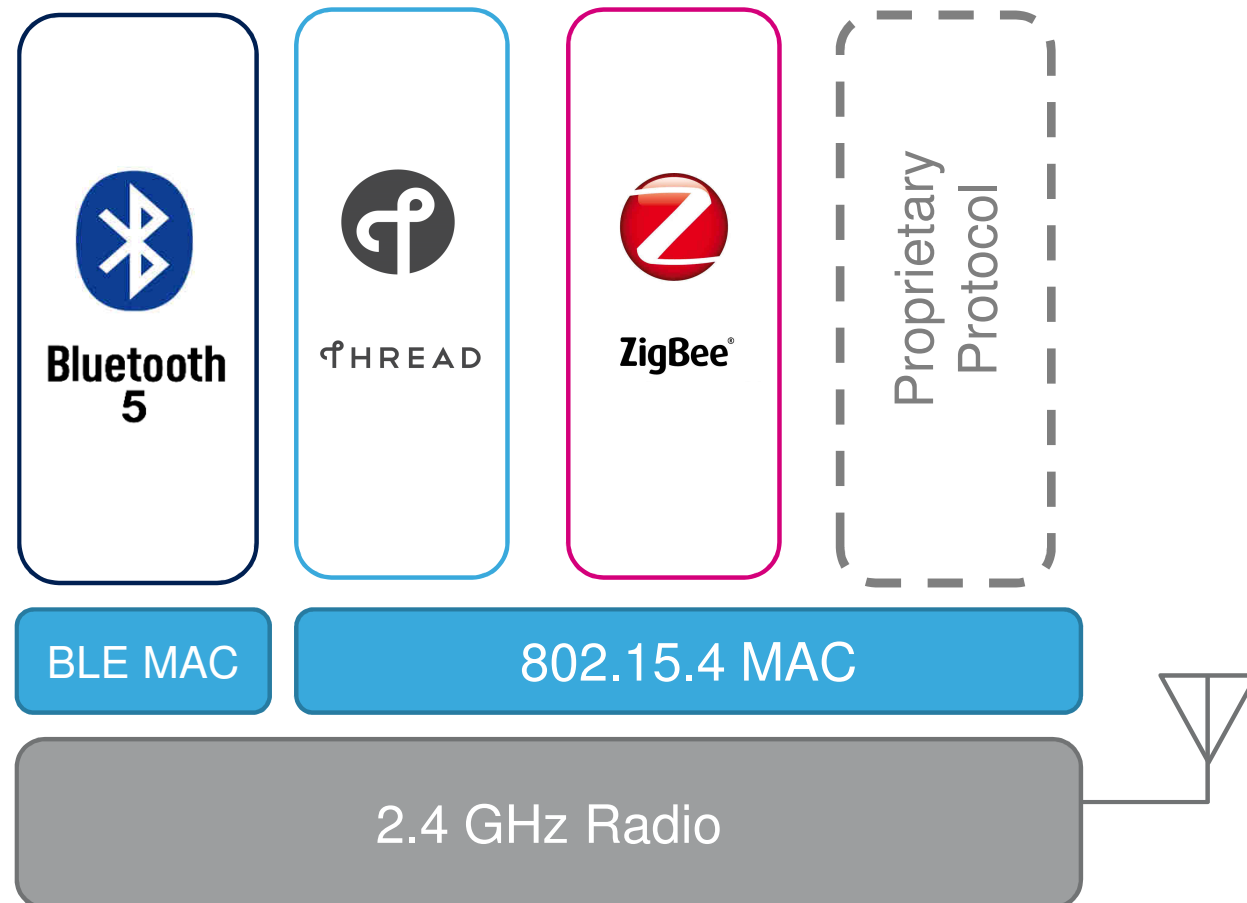
Comprehensive Ecosystem







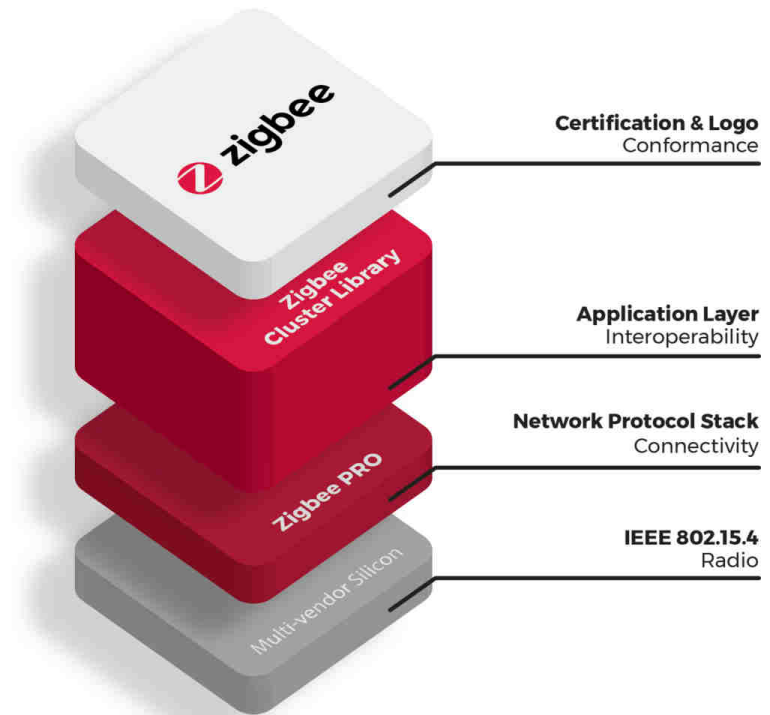




Zigbee 3.0 Stack

19

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



Coming soon
in the
ecosystem !

THREAD What it delivers

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

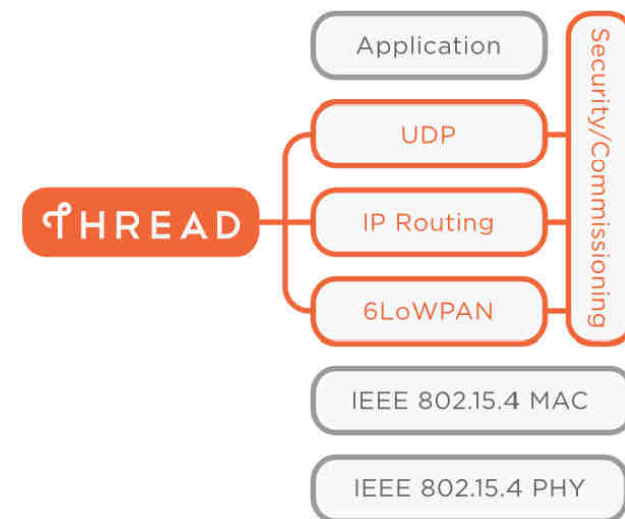
250+ products per network

Designed for very low power operation

Reliable for critical infrastructure

ST is member of
Thread Group
(Contributor level)

Can support many popular application layer protocols and platforms



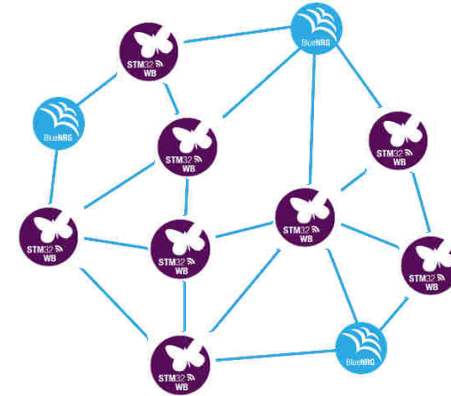
A software upgrade can add Thread to currently shipping 802.15.4 products

Bluetooth Mesh vs Thread

21

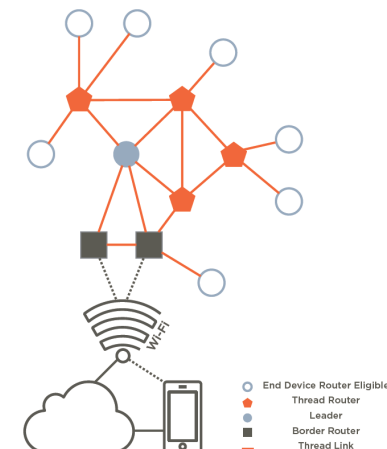
Bluetooth Mesh

- 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 data rate
- High power consumption



Thread

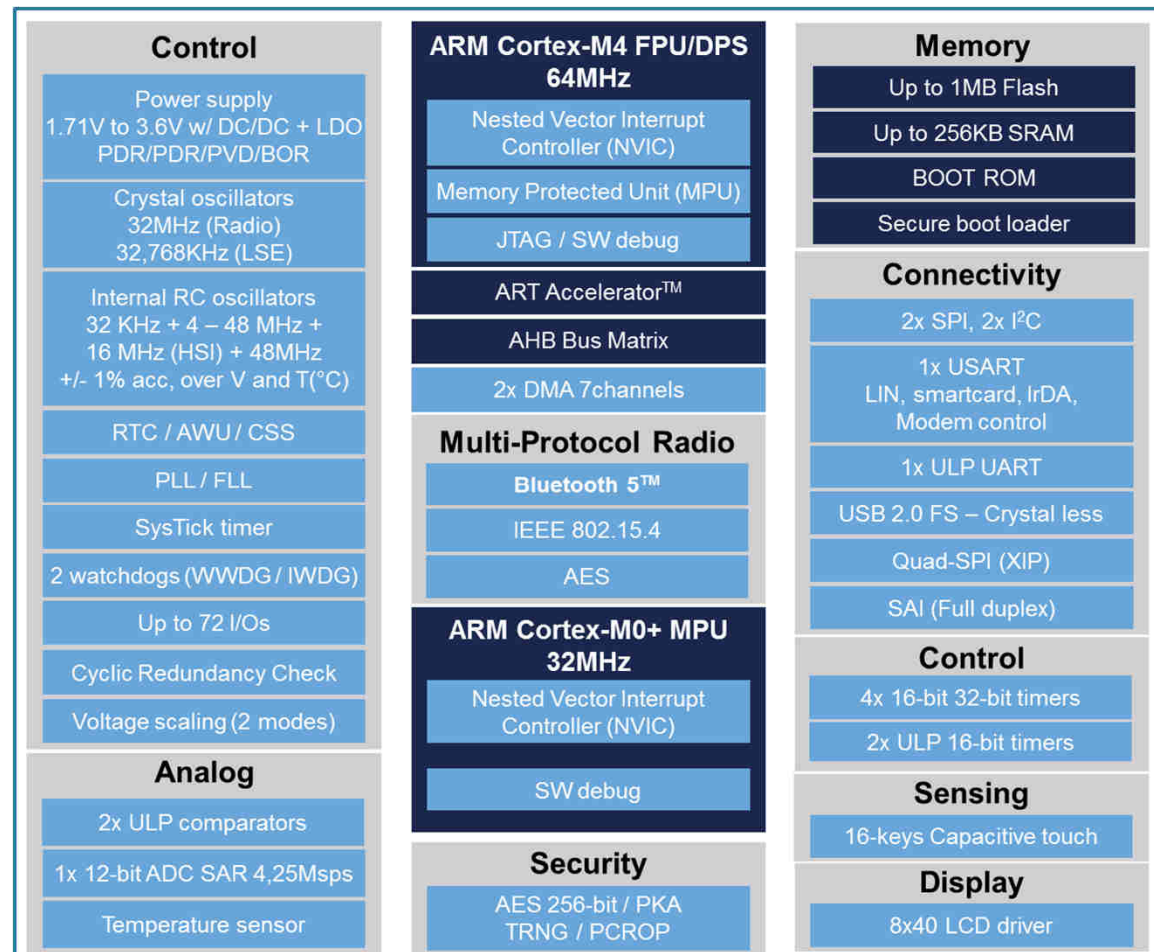
- IPv6-based using 802.15.4 MAC
- Routing table approach with network self healing
- Medium range, 40Kbps application data rate, 250Kbps on air data rate
- Low power consumption



Block Diagram

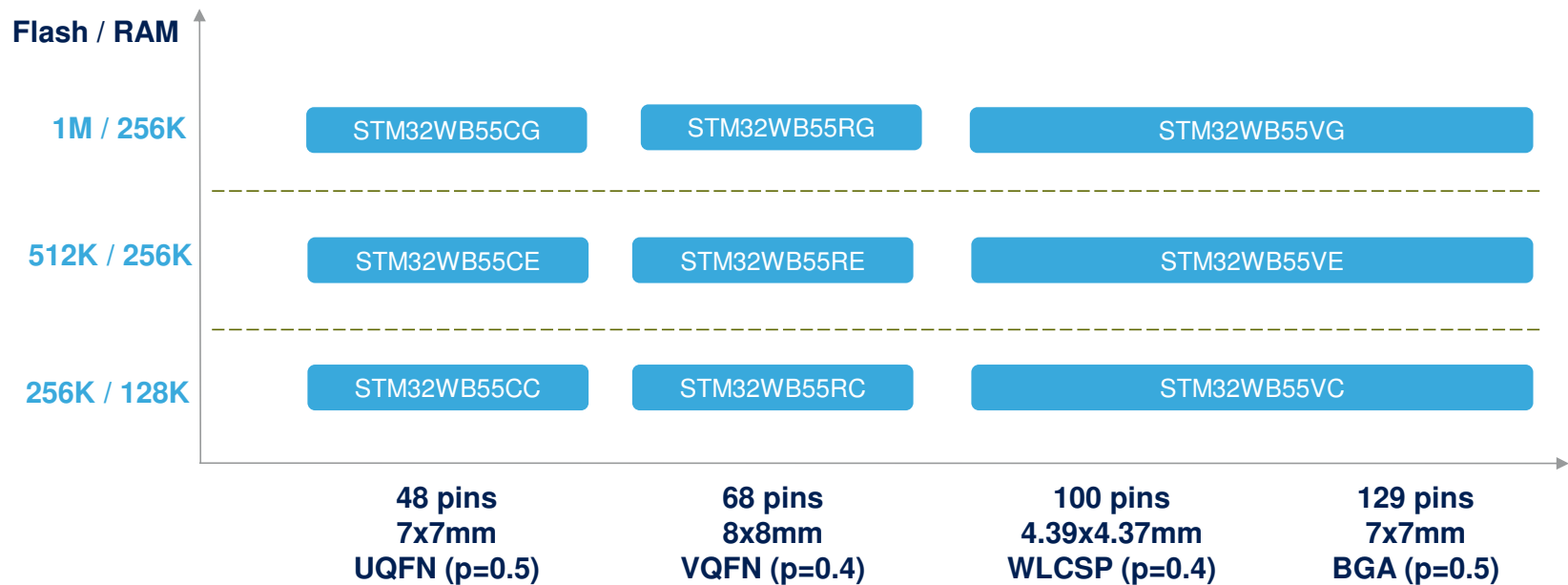
22

- 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



STM32WB55 Series Portfolio

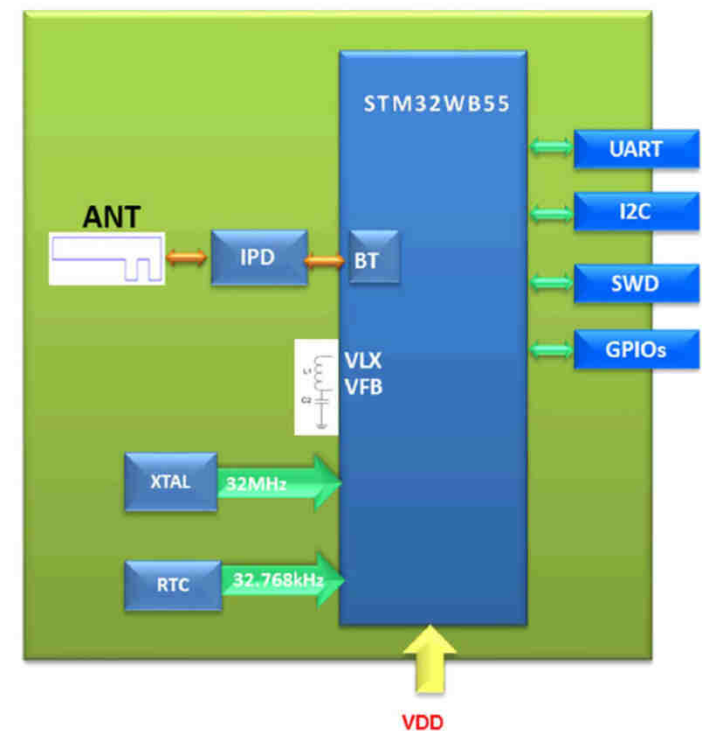
23



STM32WB55 Module

24

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



STM32WB35 – Block Diagram

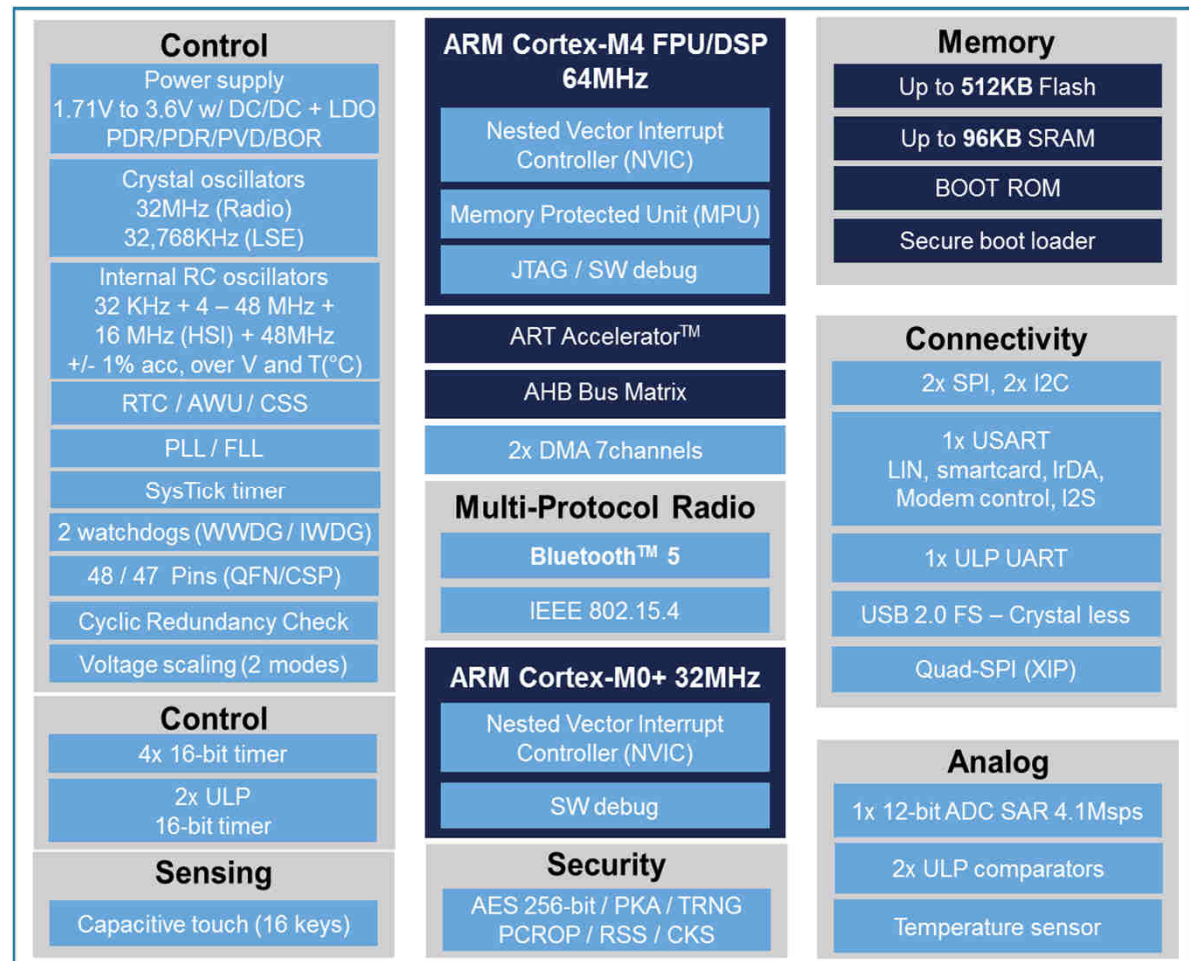
25

256KB or 512KB Flash

96KB SRAM

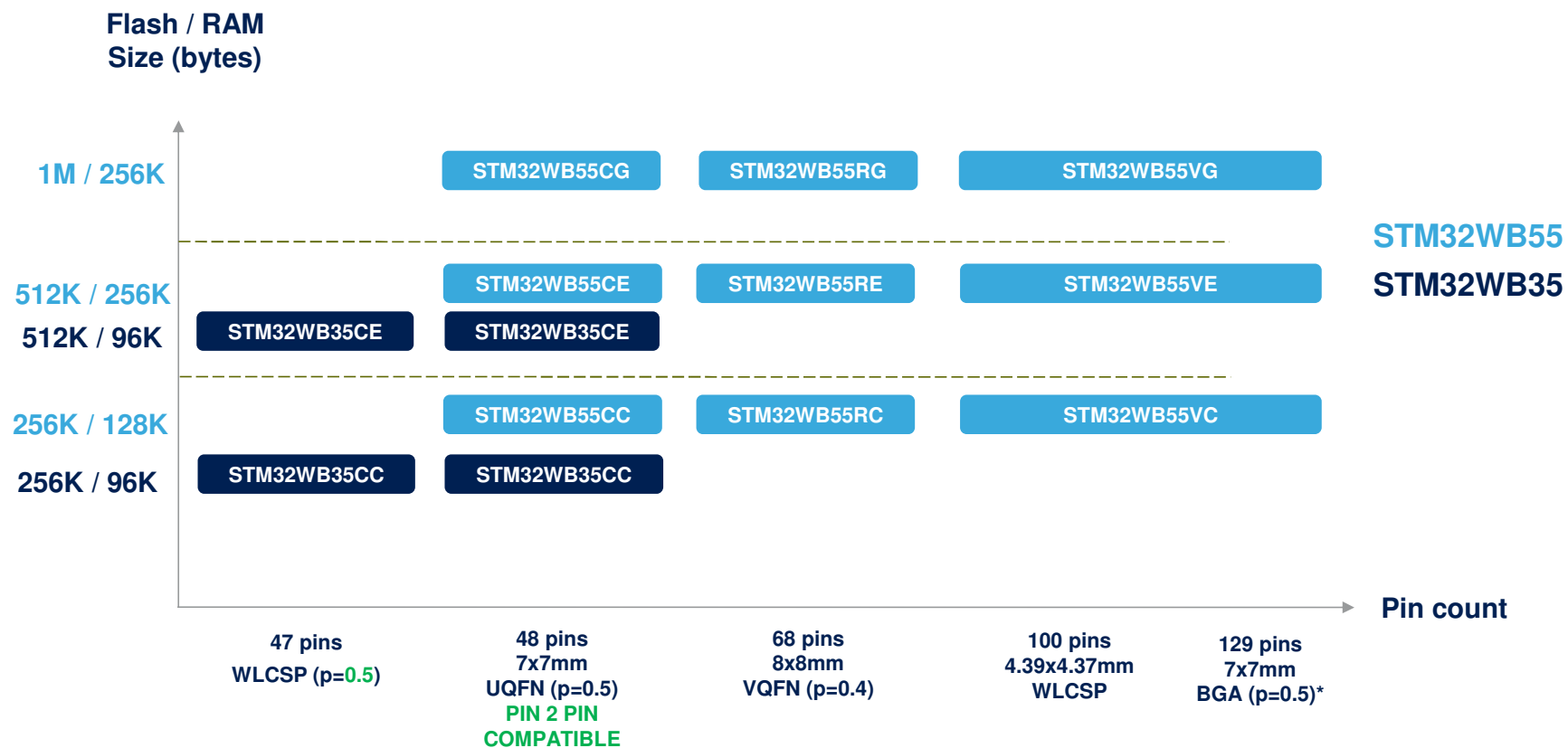
- QFN48
- WLCSP47

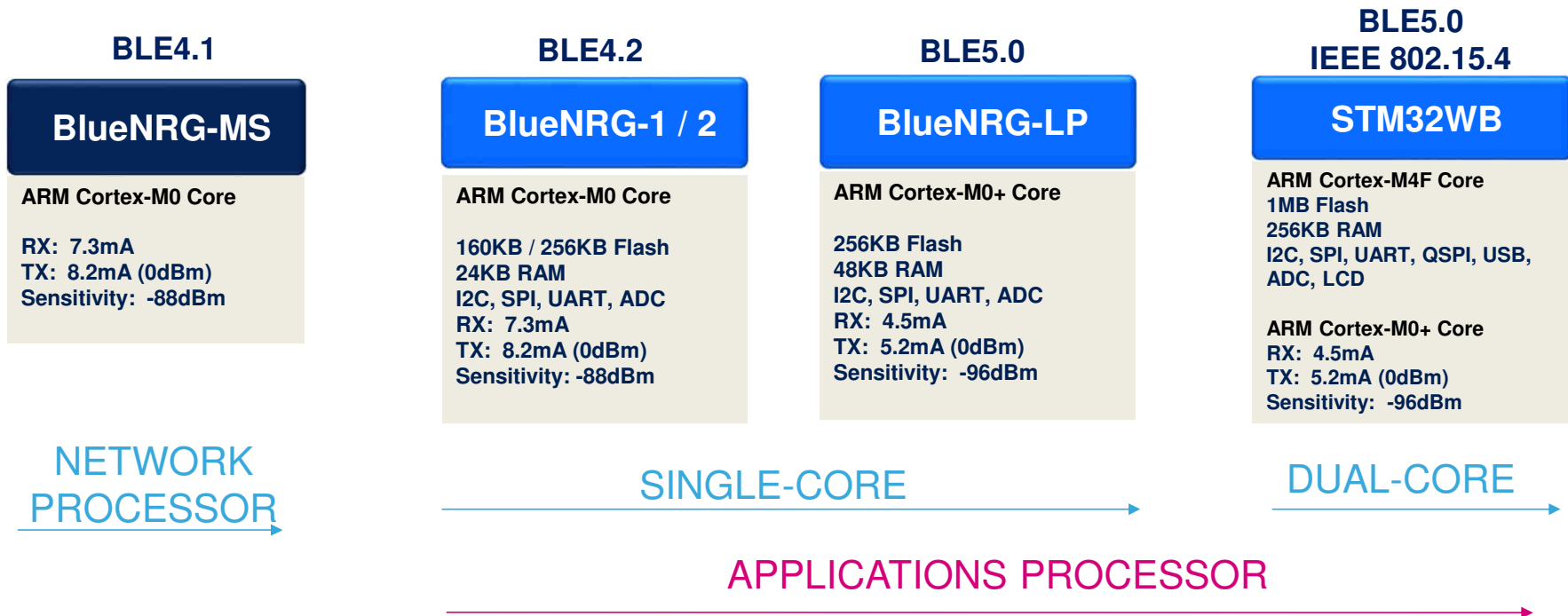
Late 2019



Positioning

26





STM32CubeMX



STM32CubeProgrammer



STM32CubeMonitorRF



STM32CubeWB



CubeWB HAL Firmware

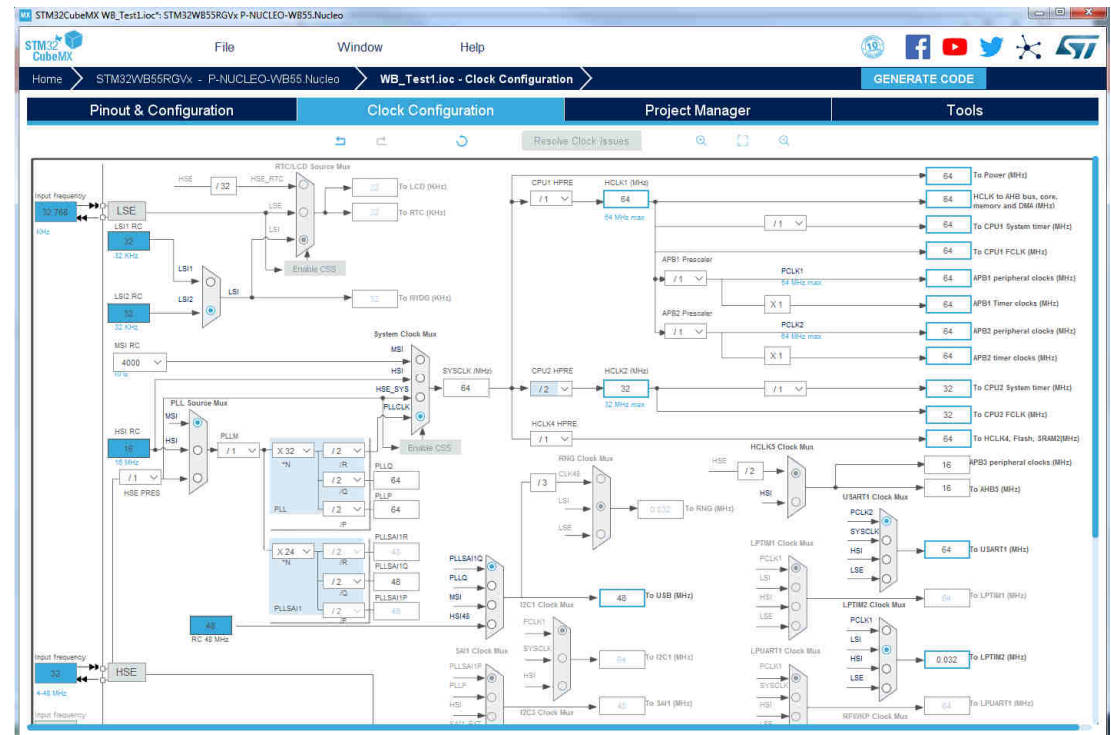


STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware

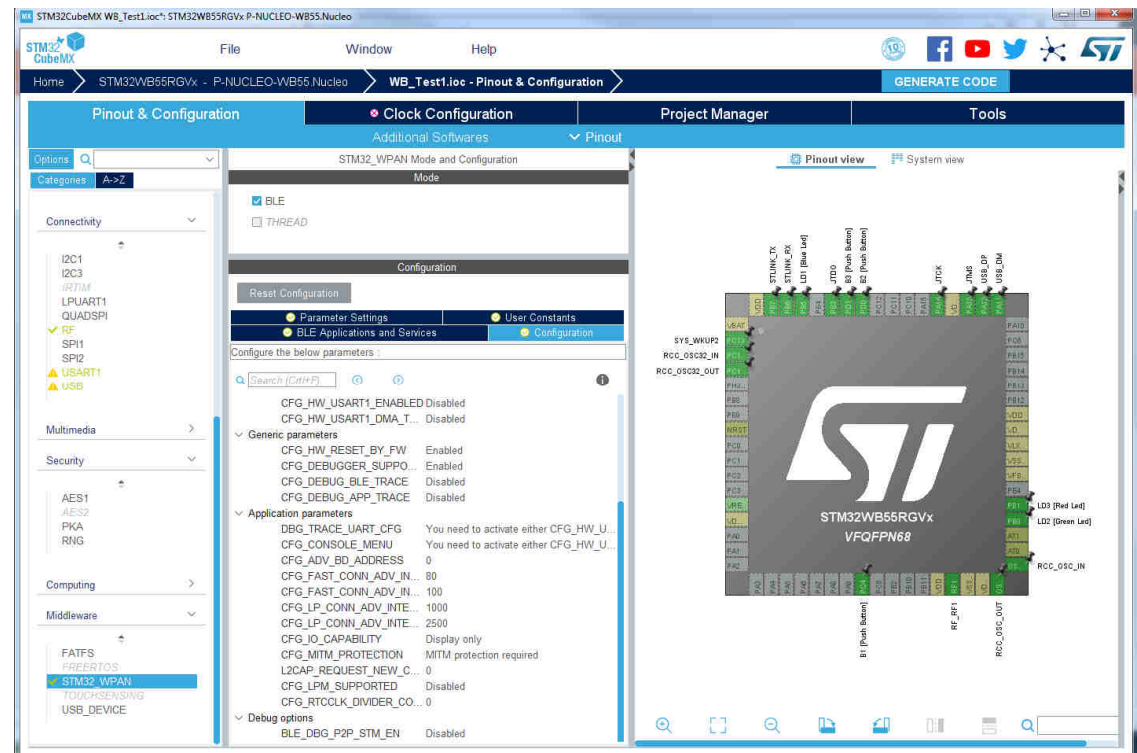


STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware

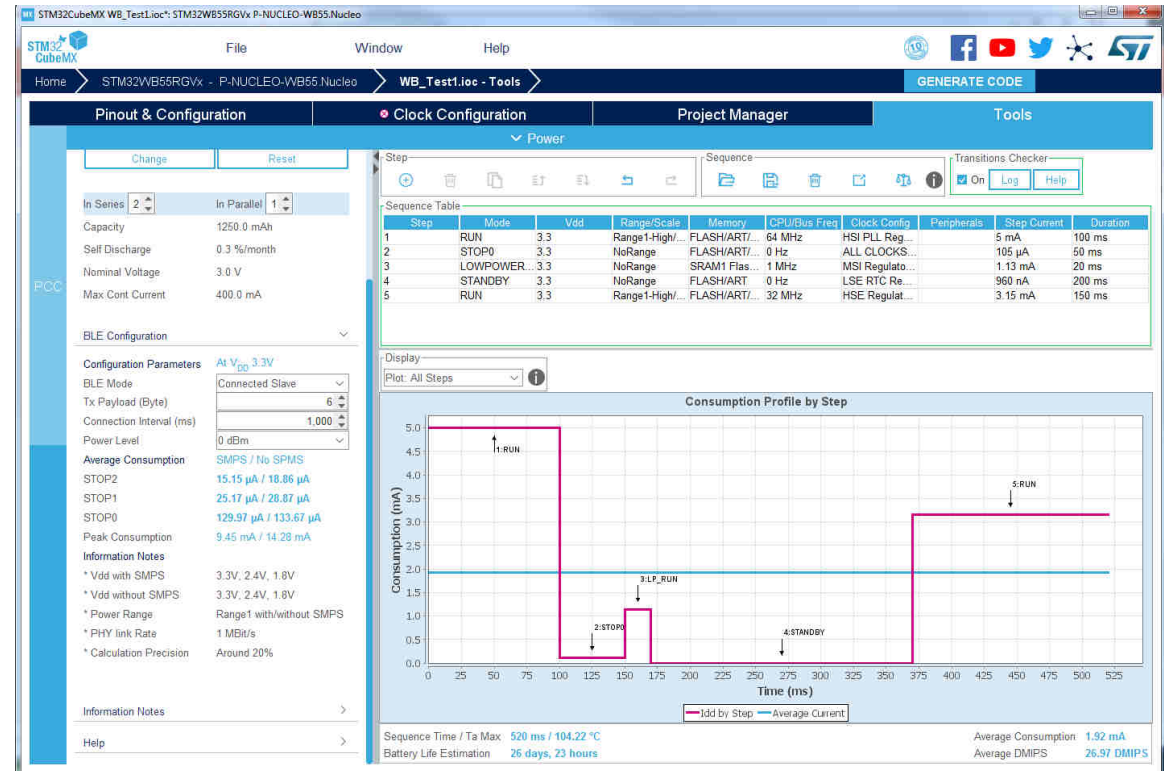


STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware

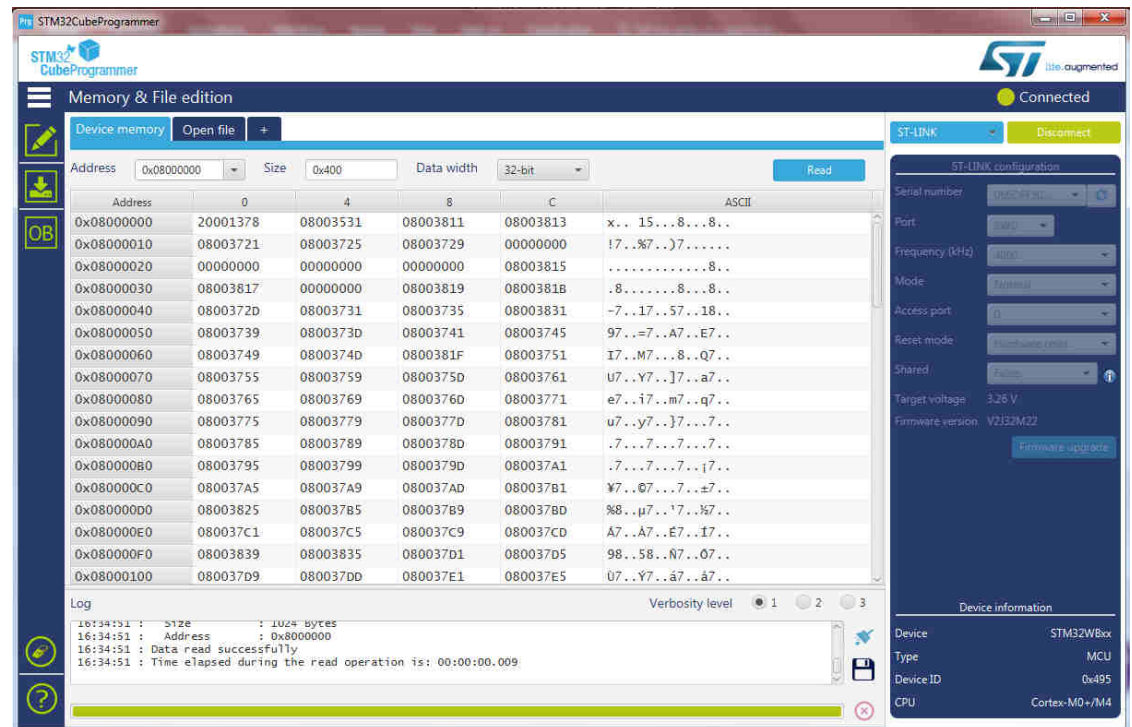


STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware

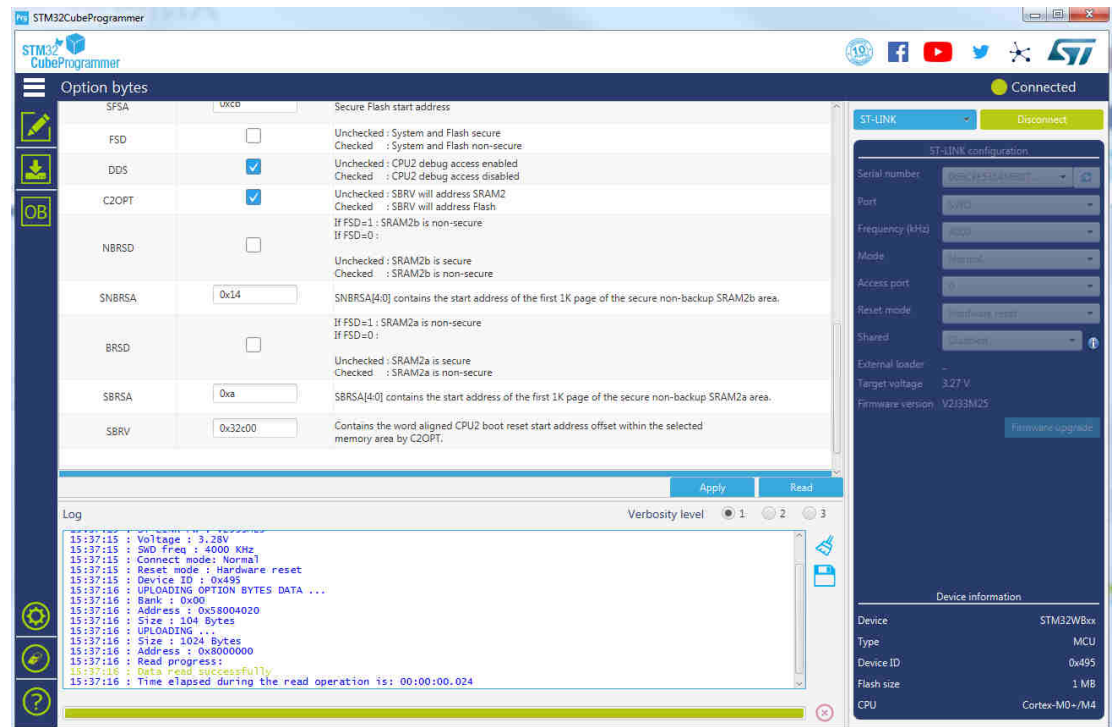


STM32CubeMX

STM32CubeProgrammer

STM32CubeMonitorRF

CubeWB HAL Firmware

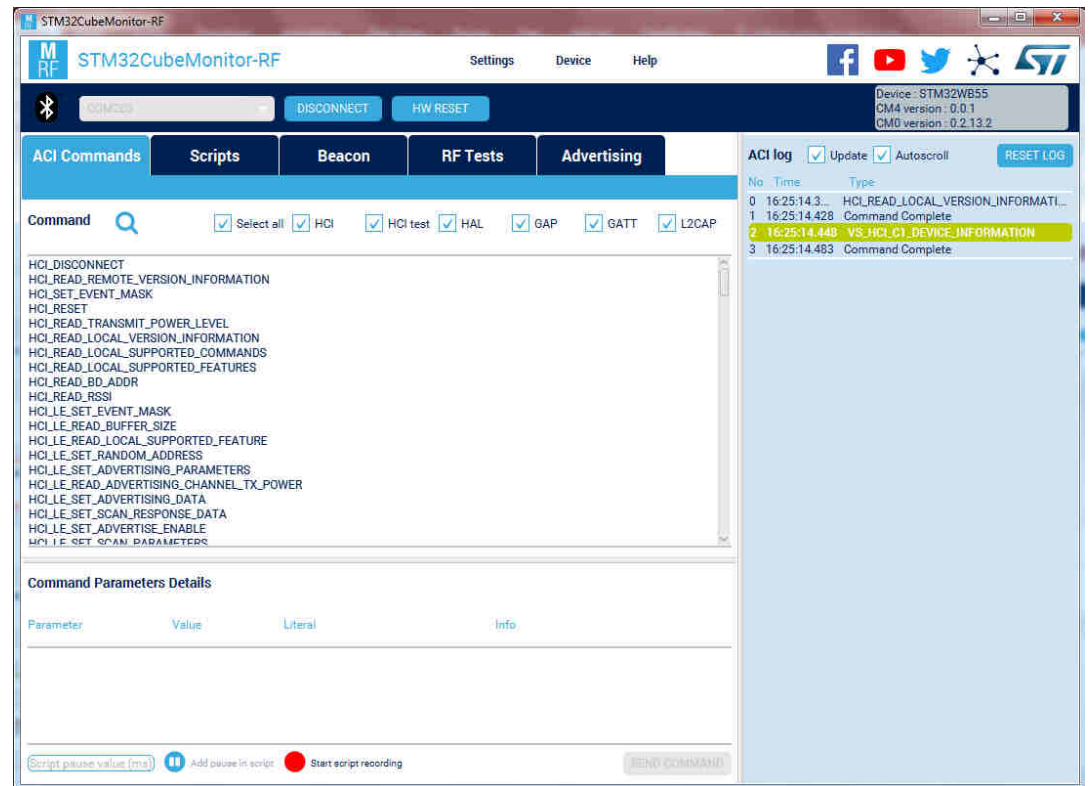


STM32CubeMX

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_Queue
- 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

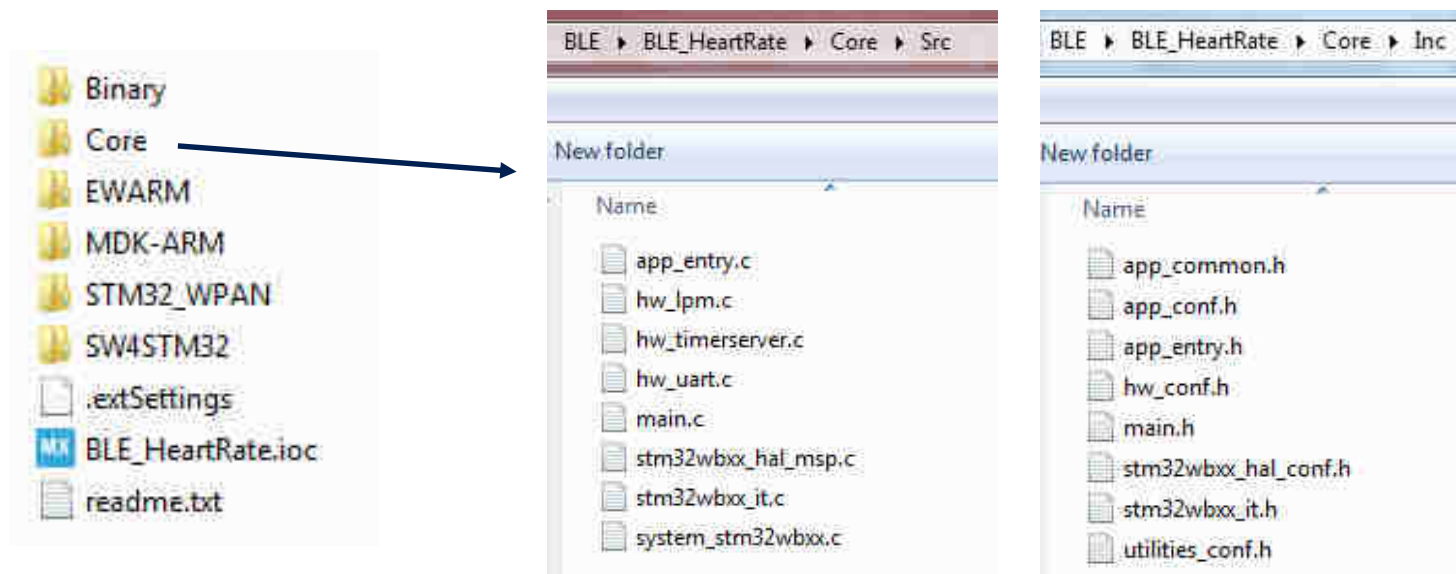
Cube Tools

36



CubeWB firmware

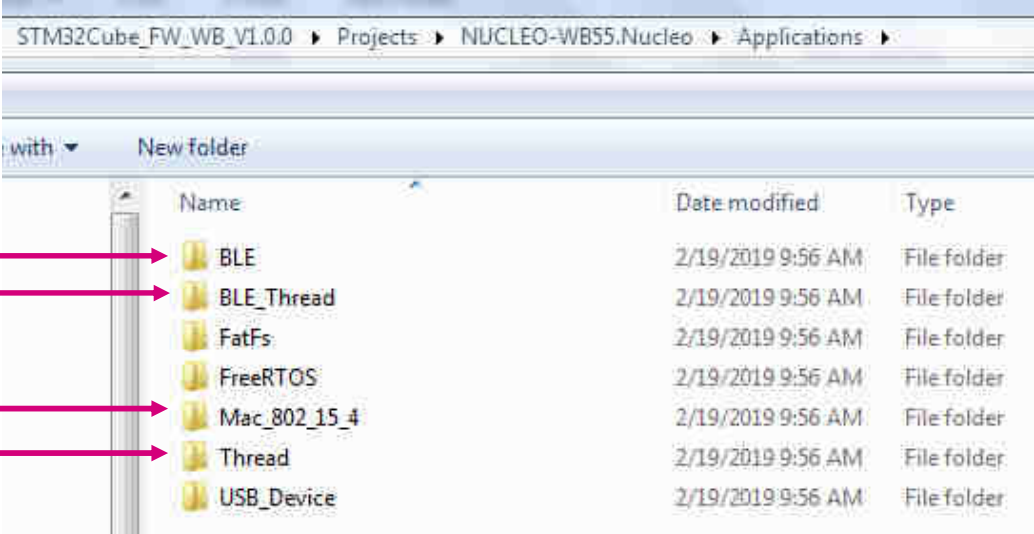
Core folder contains application-related source code



Different stacks required for different application types

BLE projects
BLE + Thread Static Concurrent mode project

Open MAC project
Thread projects

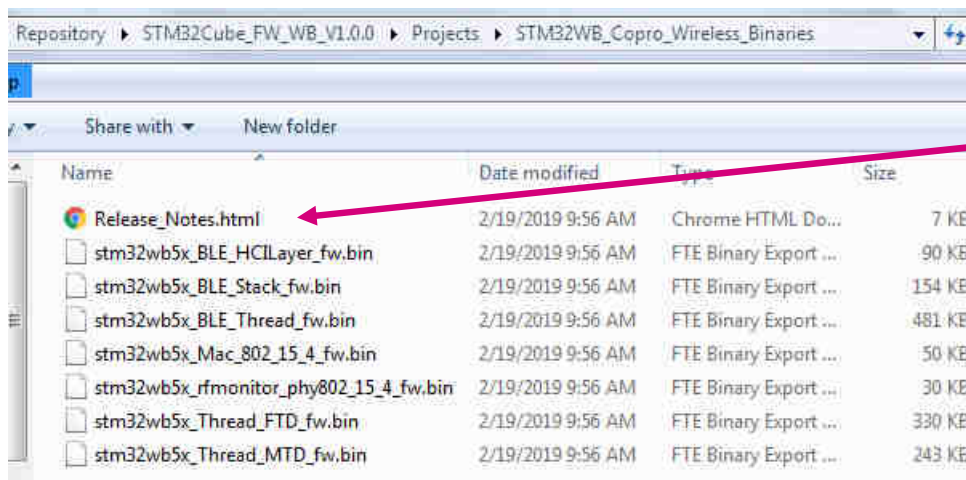


Name	Date modified	Type
BLE	2/19/2019 9:56 AM	File folder
BLE_Thread	2/19/2019 9:56 AM	File folder
FatFs	2/19/2019 9:56 AM	File folder
FreeRTOS	2/19/2019 9:56 AM	File folder
Mac_802_15_4	2/19/2019 9:56 AM	File folder
Thread	2/19/2019 9:56 AM	File folder
USB_Device	2/19/2019 9:56 AM	File folder

Zigbee 3.0 coming soon!

CubeWB firmware

Encrypted radio stack binaries here



HTML file details update procedure

Release Notes for
STM32WB Copro Wireless Binaries

Copyright © 2019 STMicroelectronics



life.augmented

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	0x08C8000	v1.0.0	02/06/2019
stm32wb5x_ble_hclayer_fw.bin	0x08C8000	v1.0.0	02/06/2019
stm32wb5x_thread_ftd_fw.bin	0x0809000	v1.0.0	02/06/2019
stm32wb5x_thread_mtd_fw.bin	0x080E000	v1.0.0	02/06/2019
stm32wb5x_ble_thread_fw.bin	0x0807000	v1.0.0	02/06/2019
stm32wb5x_mac_802_15_4_fw.bin	0x080E000	v1.0.0	02/06/2019
stm32wb5x_rfmonitor_phy802_15_4_fw.bin	0x080E000	v1.0.0	02/06/2019

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:

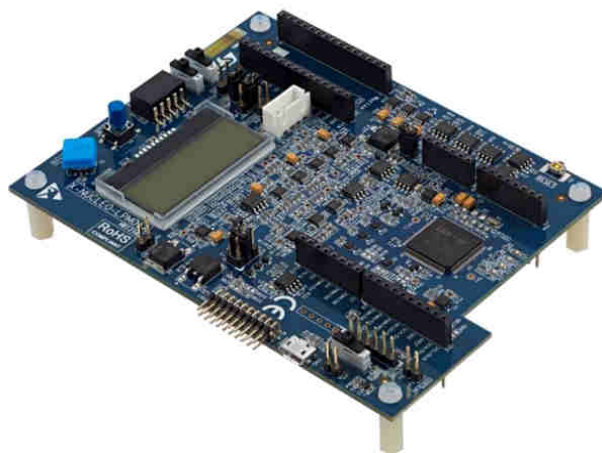
- stm32wb5x_BLE_Stack_fw.bin
 - Full BLE Stack 5.0 certified : Link Layer, HCI, L2CAP, ATT, SM, GAP and GATT database
 - BT SIG Certification listing : Declaration ID D042164
- stm32wb5x_BLE_HCLayer_fw.bin
 - HCI Layer only mode 5.0 certified : Link Layer, HCI
 - BT SIG Certification listing : Declaration 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)

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

Nucleo & Dongle boards come preloaded with the BLE stack

STM32CubeMonitor-Power

\$70

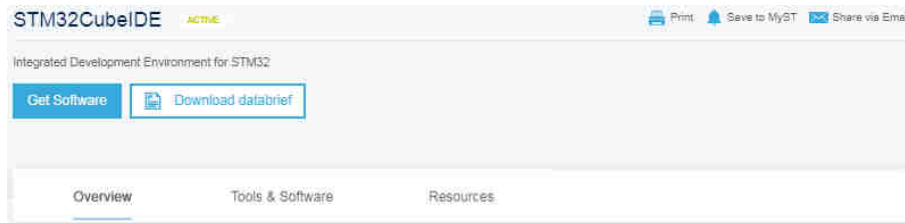


X-NUCLEO-LPM01A



CubeIDE & Atollic TrueSTUDIO

41



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



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.

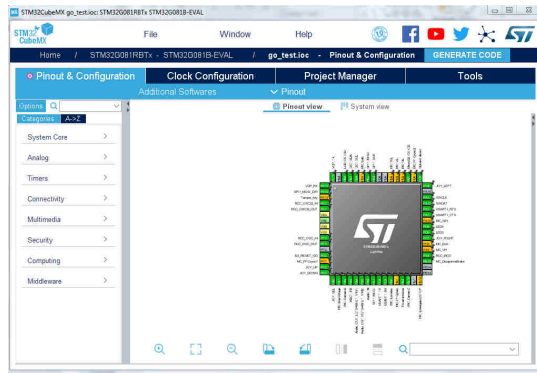
STM32CubeIDE also includes standard and advanced debugging features including views of CPU core registers, memories, and peripheral registers, as well as live variable watch, Serial Wire Viewer interface, or fault analyzer.

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.

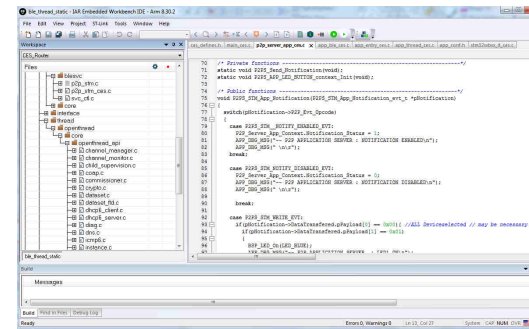
STM32CubeIDE integrates all STM32CubeMX functionalities to offer all-in-one tool experience and



Configure



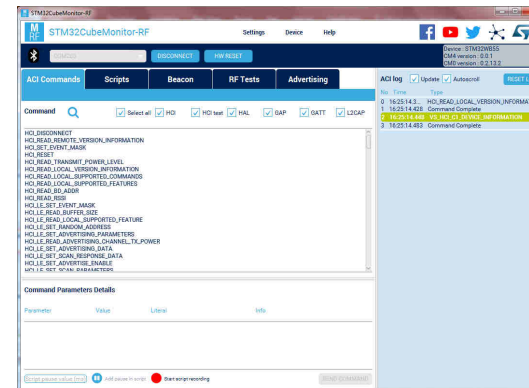
Code & Debug



Measure



Test



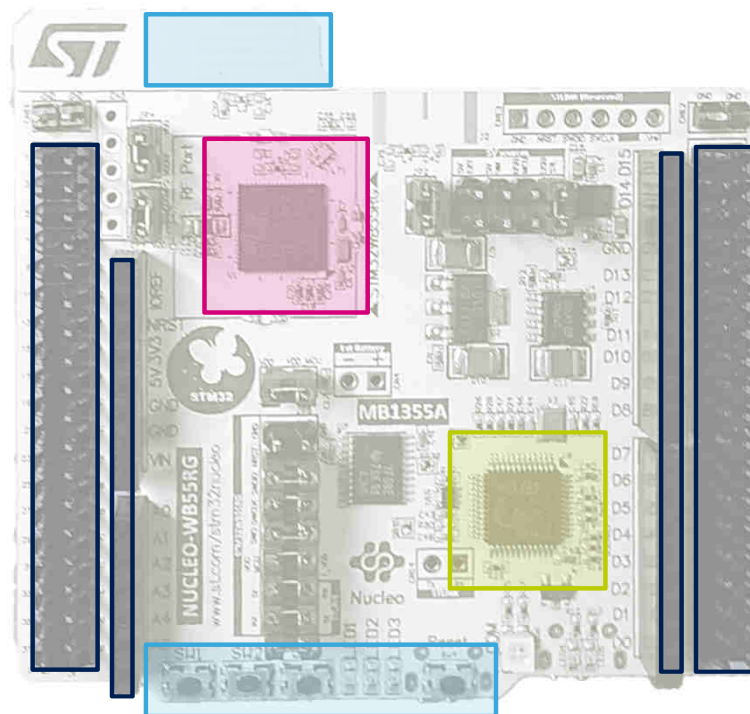
2.4GHz PCB antenna

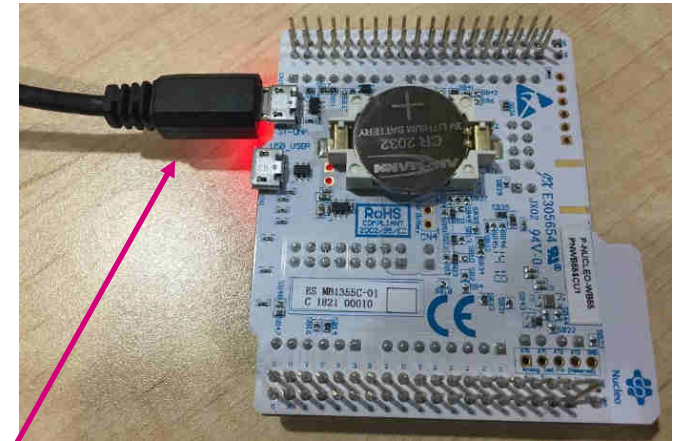
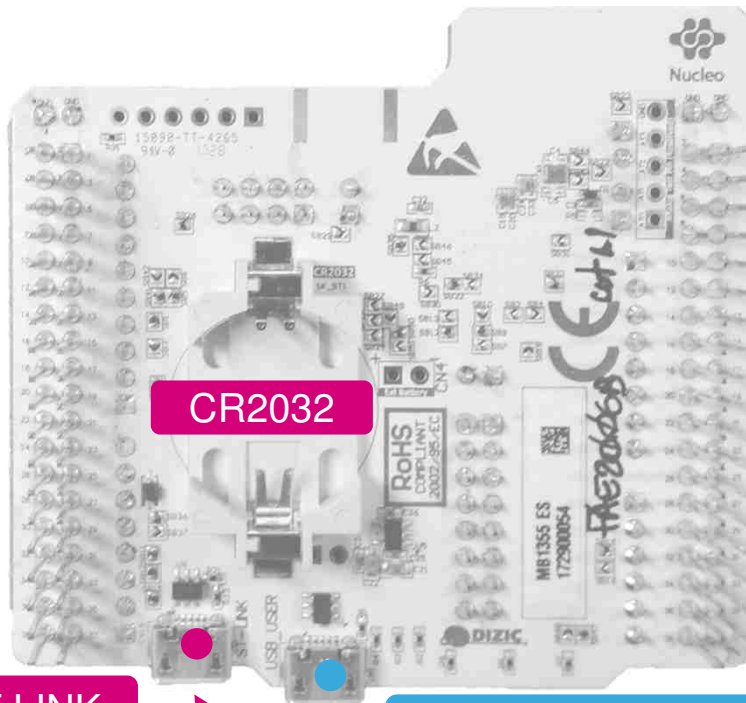
STM32WB55RGV6
(VQFPN68)

Arduino & Morpho
Headers

ST-Link/V2-1

Buttons & LED's



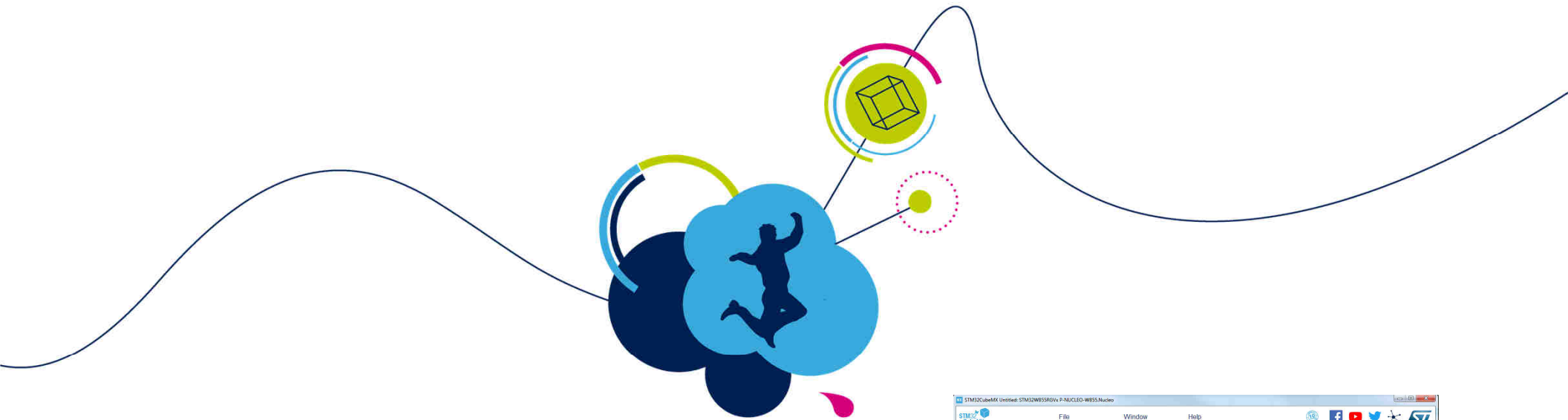


We will use this one!

Your Magic number!

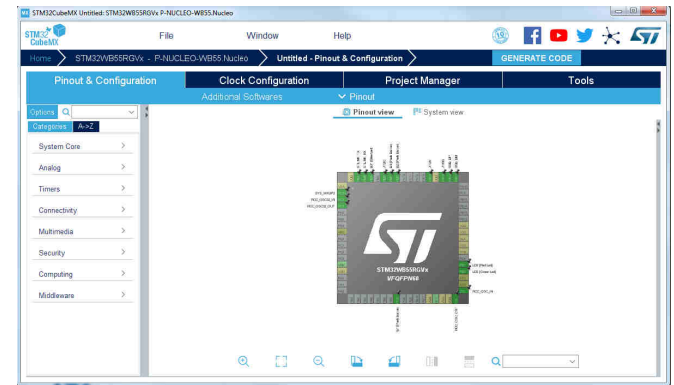
45





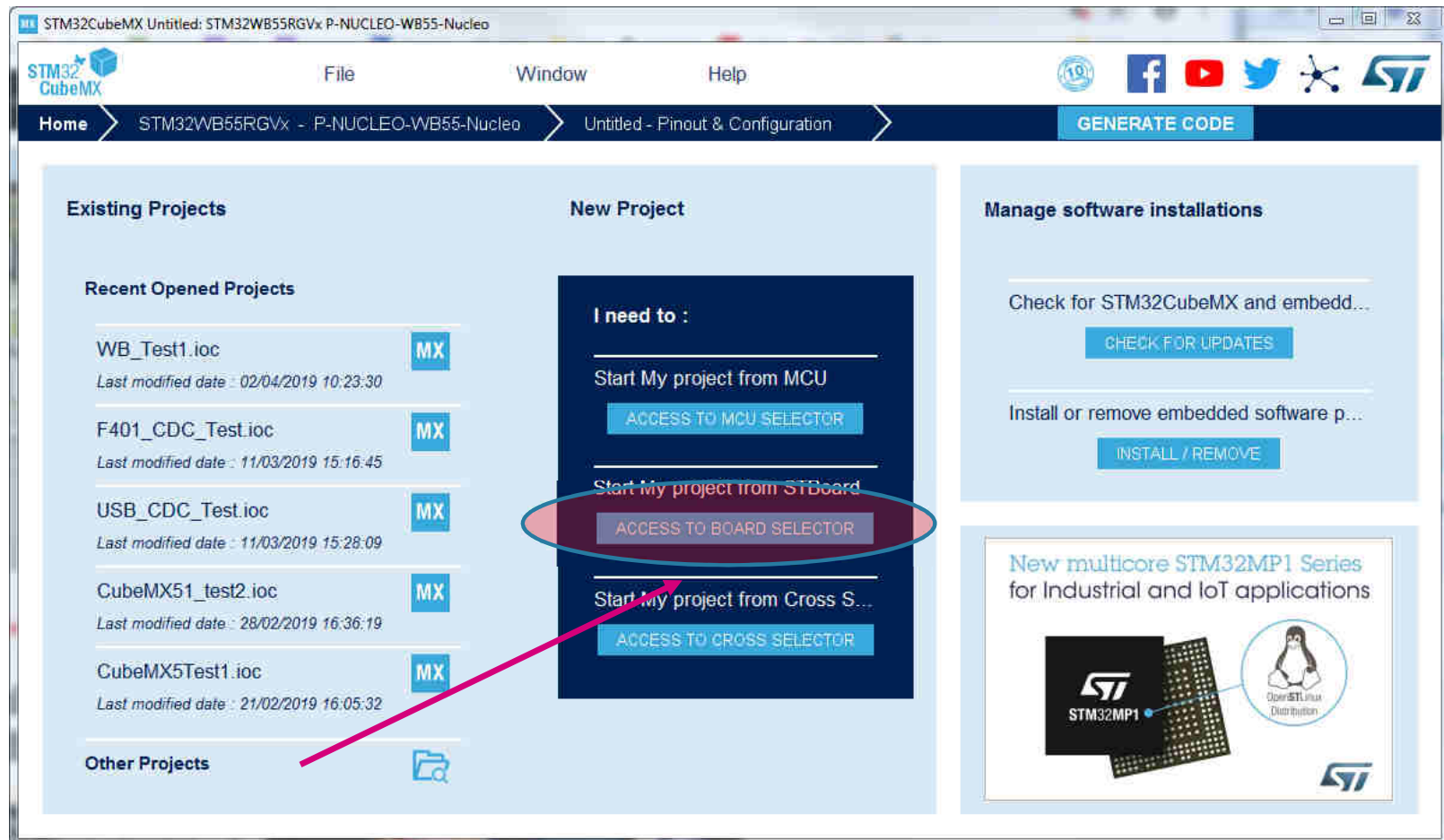
Hands-On

CubeMX



Launch CubeMX & Start project from Board Selector

47



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

48

Board Filters

Part Number Search

Vendor

Type

MCU Series

Check/Uncheck All

- ☐ STM32F0
- ☐ STM32F1
- ☐ STM32F2
- ☐ STM32F3
- ☐ STM32F4
- ☐ STM32F7
- ☐ STM32G0
- ☐ STM32H7
- ☐ STM32L0
- ☐ STM32L1
- ☐ STM32L4
- ☐ STM32L4+
- ☐ STM32MP1
- ☒ STM32WB

Other

Price = 0.0

Oscillator Freq. = 0 (MHz)

Neural Networks on STM32 MCUs
simple, fast, optimized

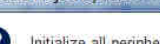
STM32Cube.AI

ST

Boards 2 items

	Part No.	Type	Marketing Status	Unit Price (US\$)	Mounted Device	Kit Conte.	Included in
	P-NUCLEO-WB55...	Nucleo64		0.0	STM32WB55RGVx		
	P-NUCLEO-WB55...	Nucleo USB Don...		0.0	STM32WB55CGU0		

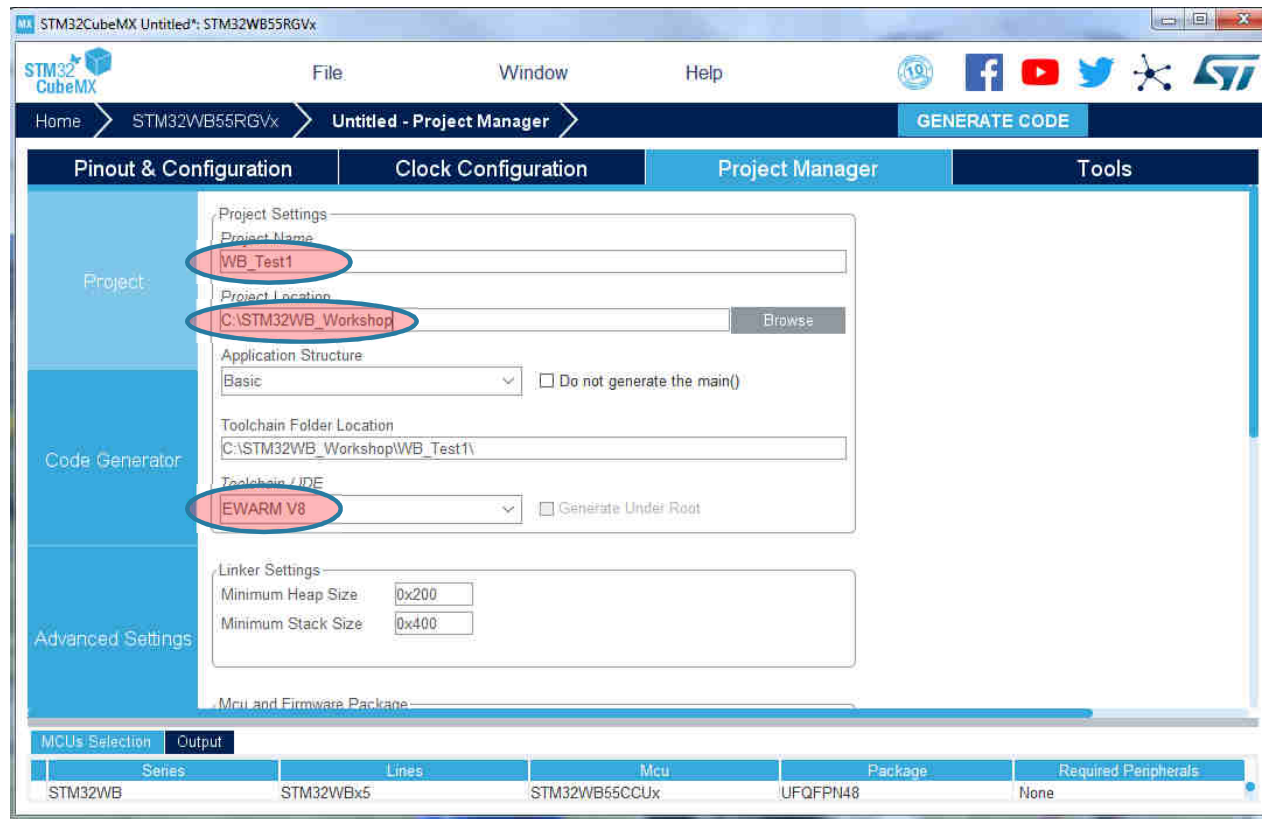
49



P-NUCLEO-WB55 Board Project

50

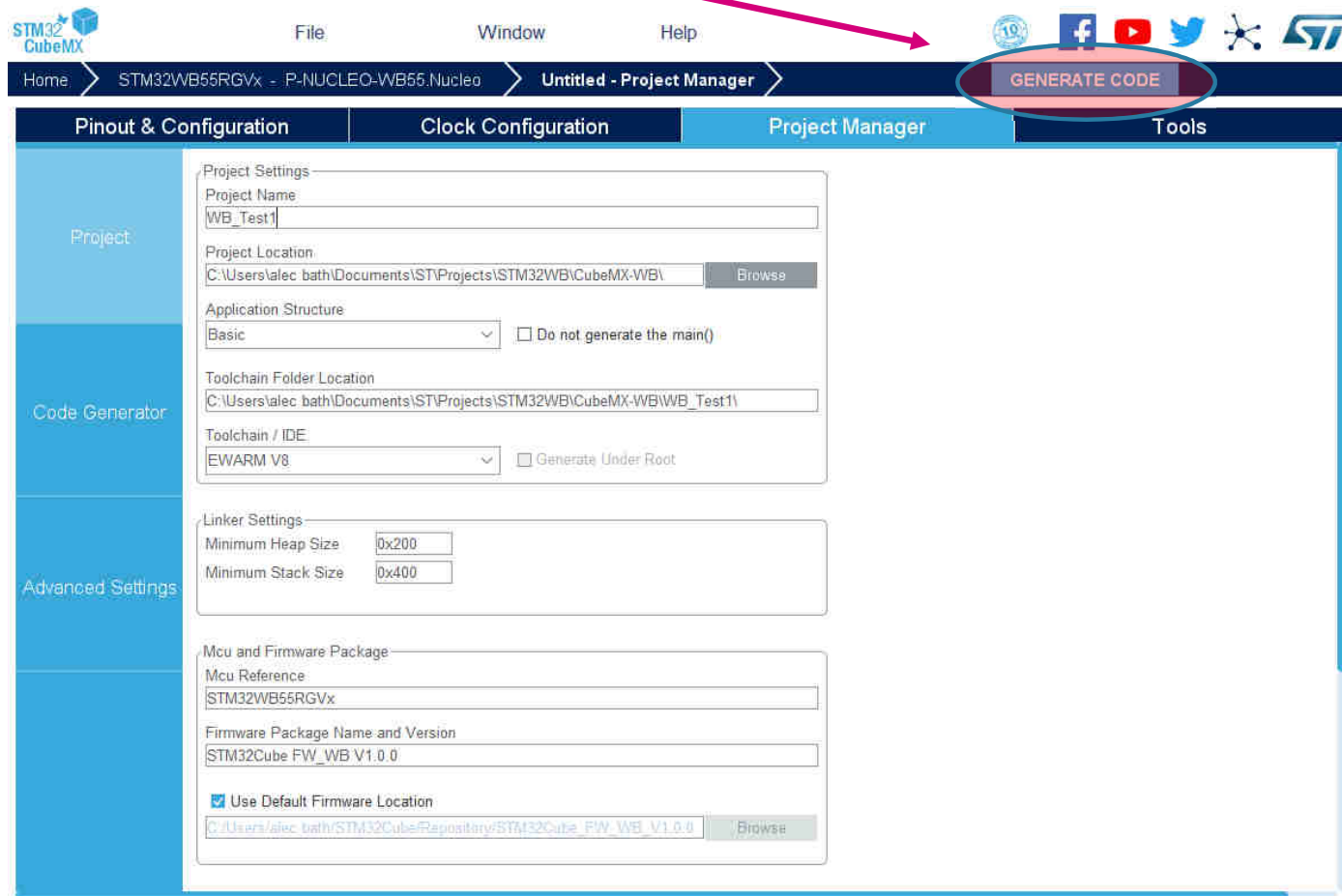
- Name your project
- Recommended Project location: C:\STM32WB_Workshop\
- Use EWARM V8 toolchain



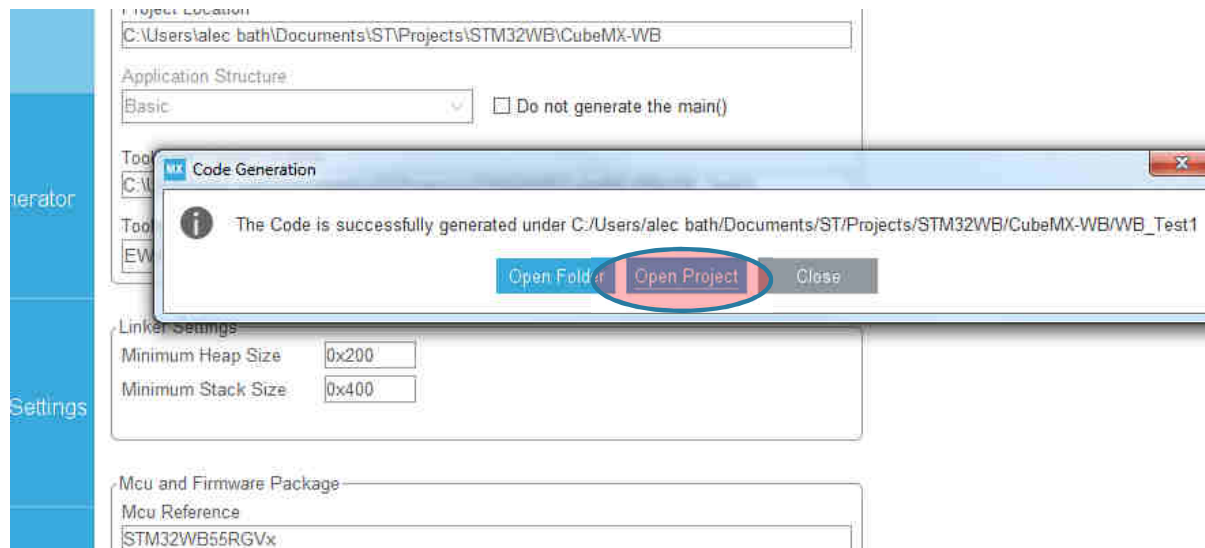
P-NUCLEO-WB55 Board Project

GENERATE CODE

51



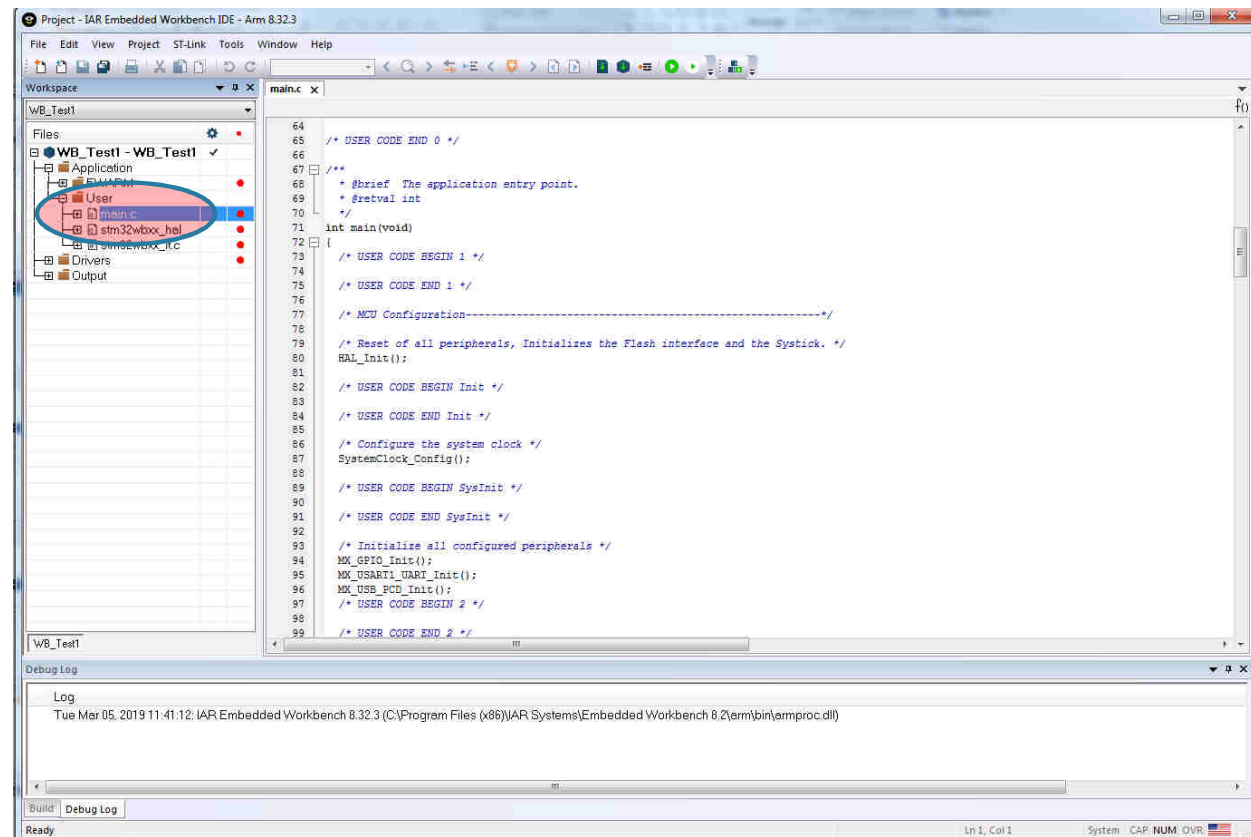
- Open Project



P-NUCLEO-WB55 Board Project

53

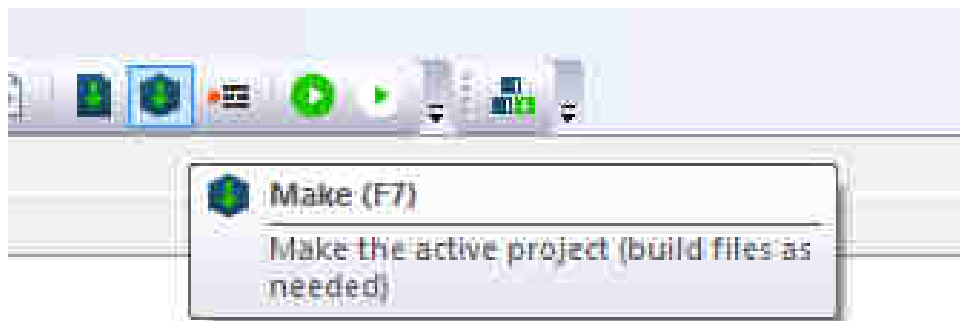
- Expand the **User** file tree and Open **main.c**
- Enable line numbers in IAR (Tools > Options > Editor > Show line numbers)



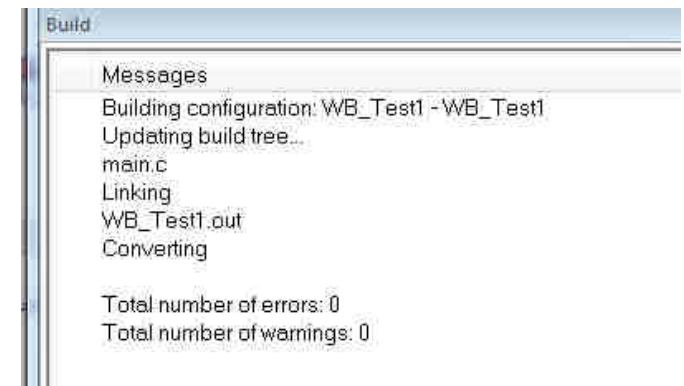
Add some code to while(1) loop:

```
101  /* Infinite loop */
102  /* USER CODE BEGIN WHILE */
103  while (1)
104  {
105
106      HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_5);
107      HAL_Delay(100);
108      HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_0);
109      HAL_Delay(100);
110      HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_1);
111      HAL_Delay(100);
112
113      /* USER CODE END WHILE */
114
```

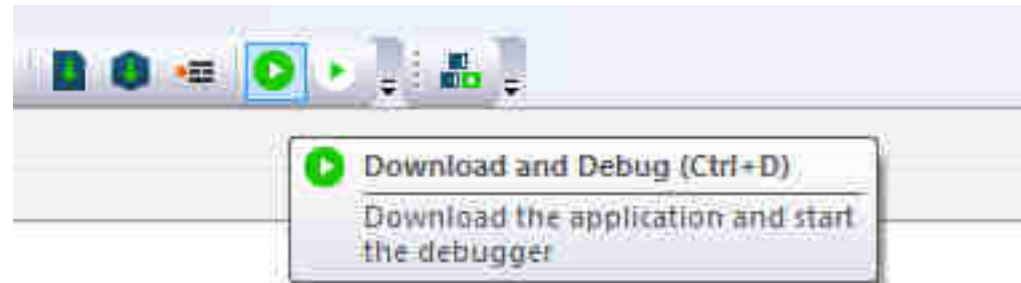
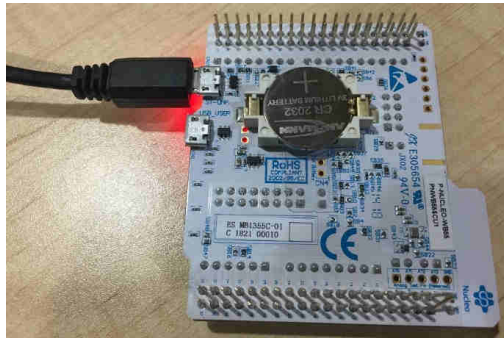
Build the project



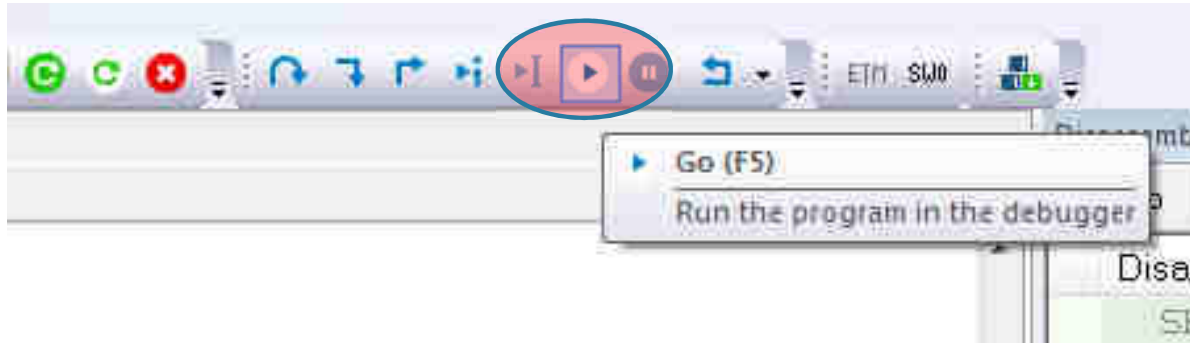
Check for errors



Download & Debug (attach your board) 😊



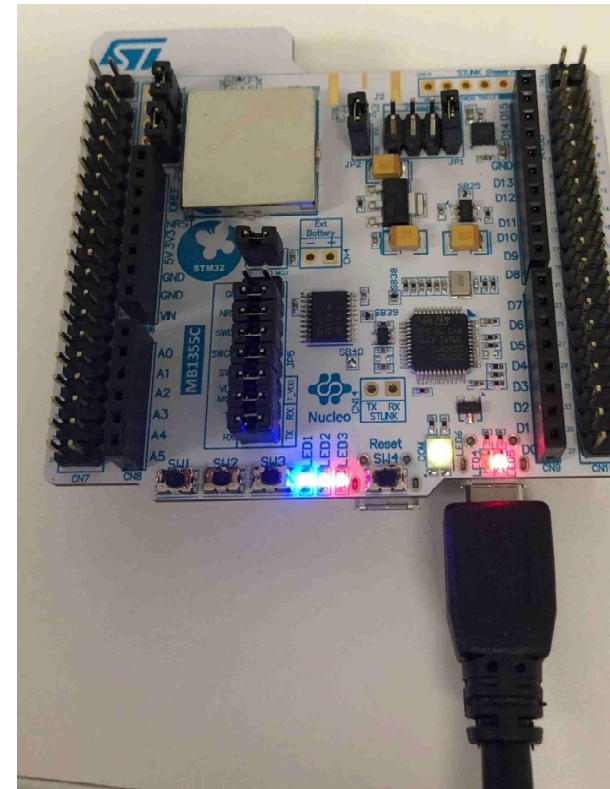
GO!



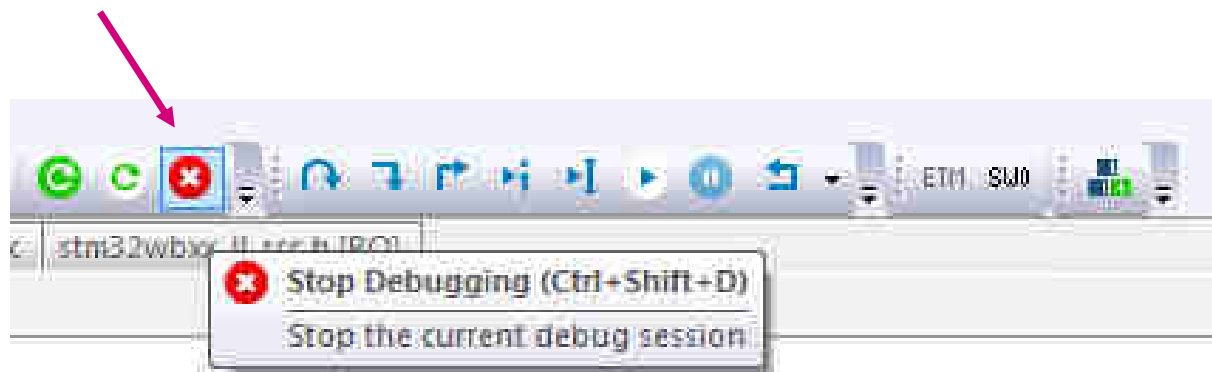
P-NUCLEO-WB55 Board Project

58

Enjoy the dancing LED's! 😊



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

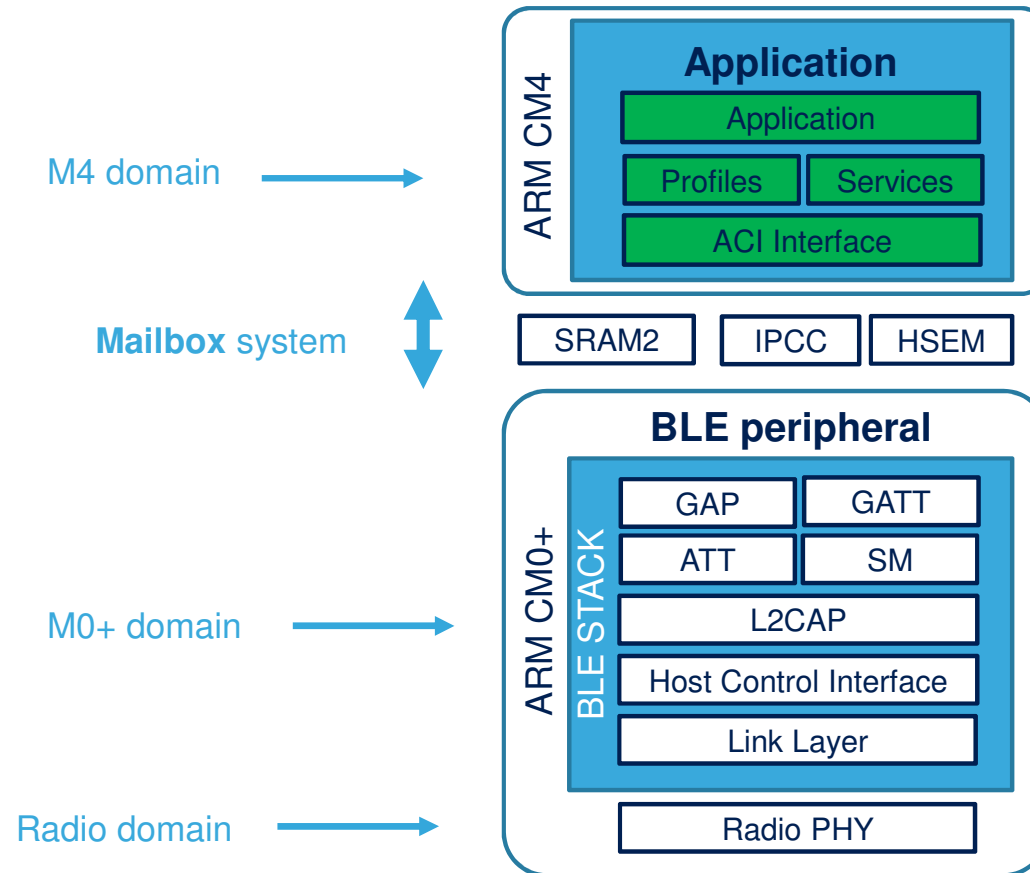




BLE Fundamentals

BLE Protocol Stack layers

61



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

62

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 advertising)
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

100X lower

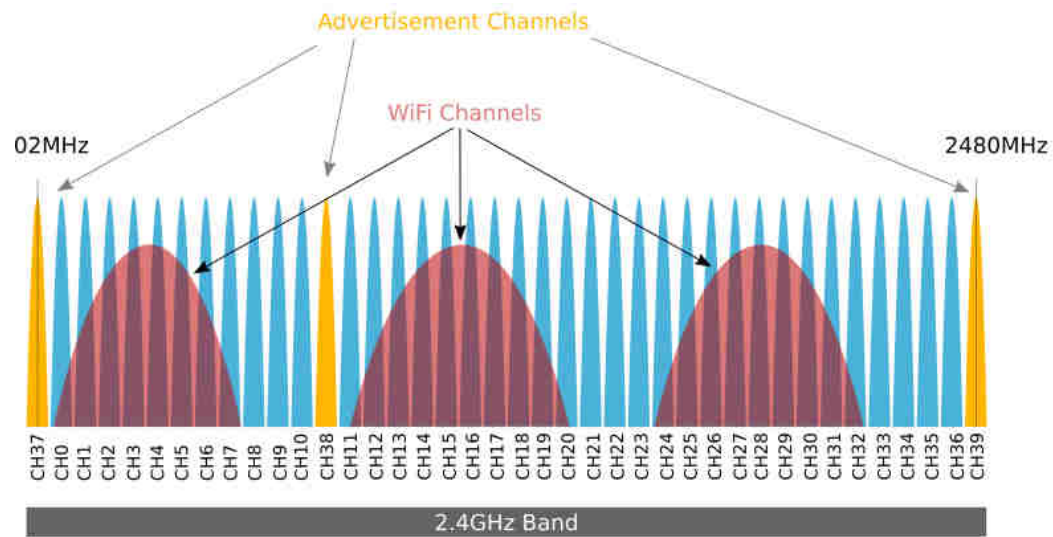
Longer range

Fast connection (only 3 advertising channels to scan)

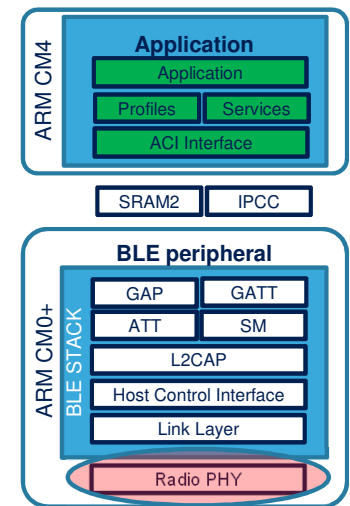
Relaxed RF requirements (lower cost silicon / passives)

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 / 8DPSK
Data Rate	1 Mbit/s	1 Mbit/s	2 and 3 Mbit/s
Channels	40	79	79
Spacing	2MHz	1MHz	-



Link Layer State Machine

64

- **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**

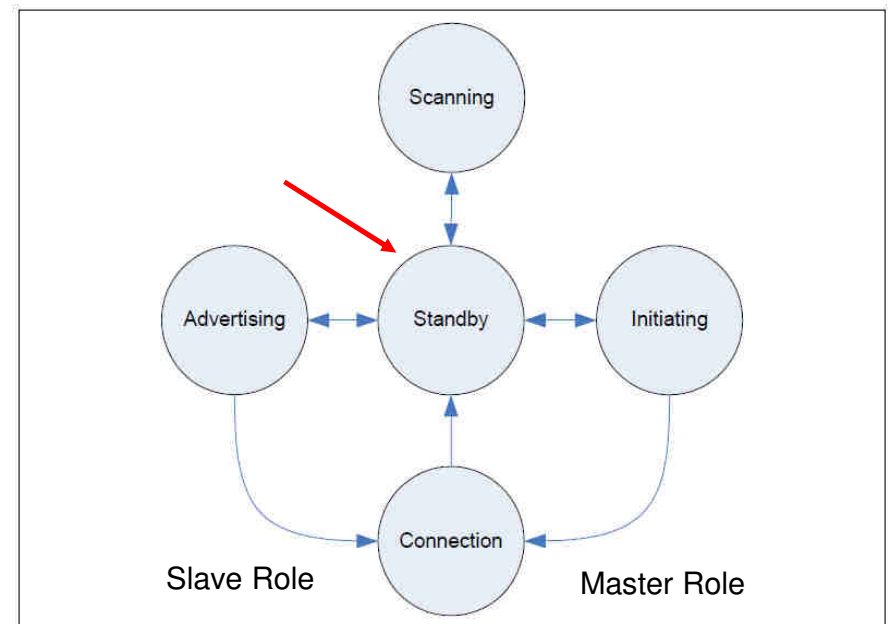
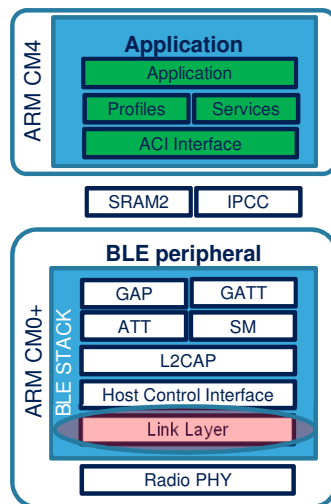


Figure 1.1: State diagram of the Link Layer state machine

Link Layer State Machine

65

- **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**

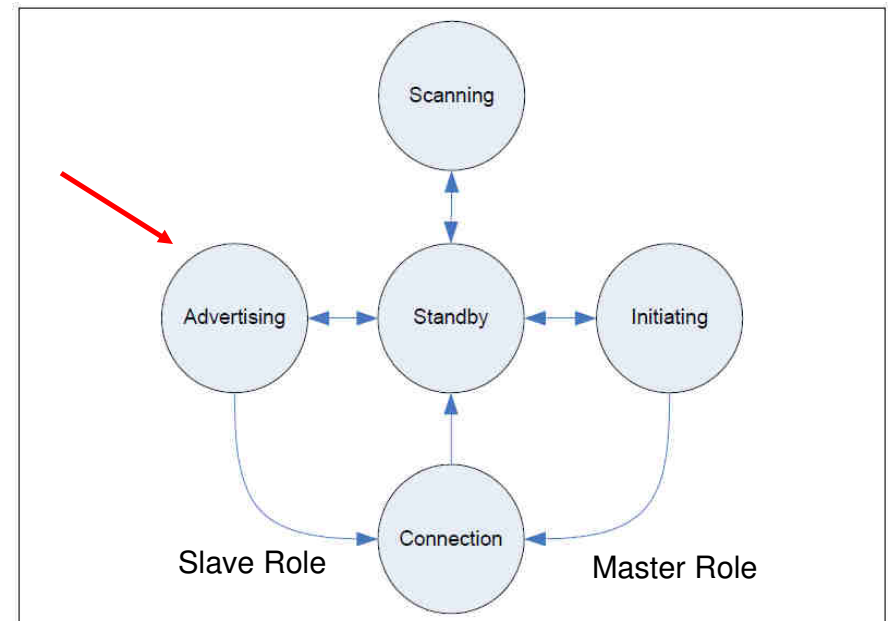
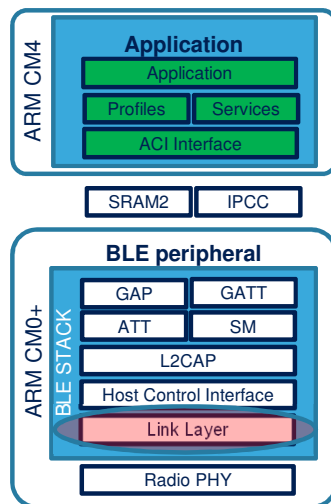


Figure 1.1: State diagram of the Link Layer state machine

Link Layer State Machine

66

- **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**

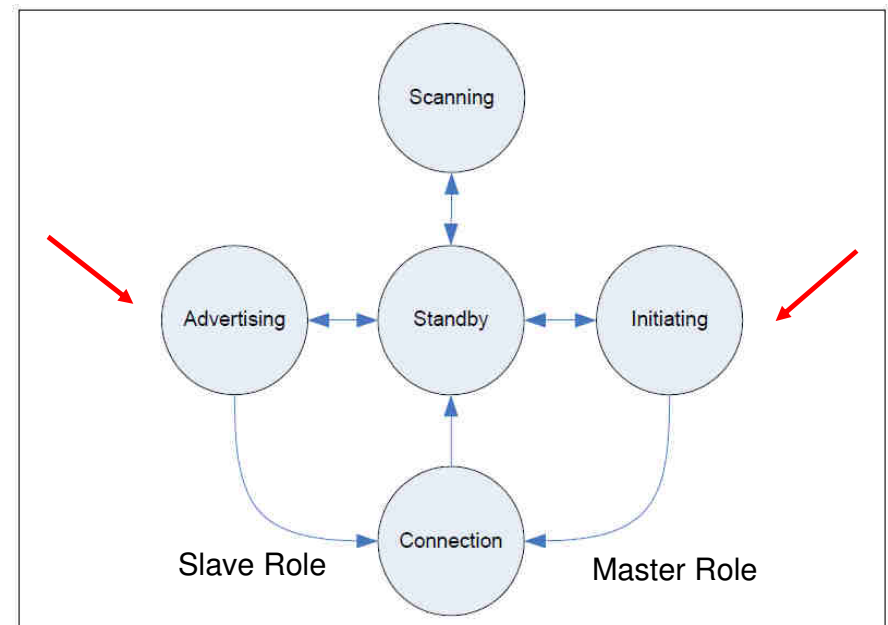
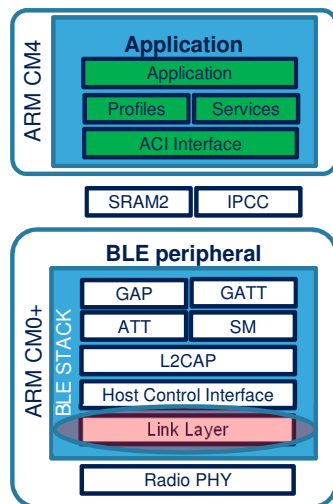


Figure 1.1: State diagram of the Link Layer state machine

Link Layer State Machine

67

- **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**

Up to 8 simultaneous State Machines!

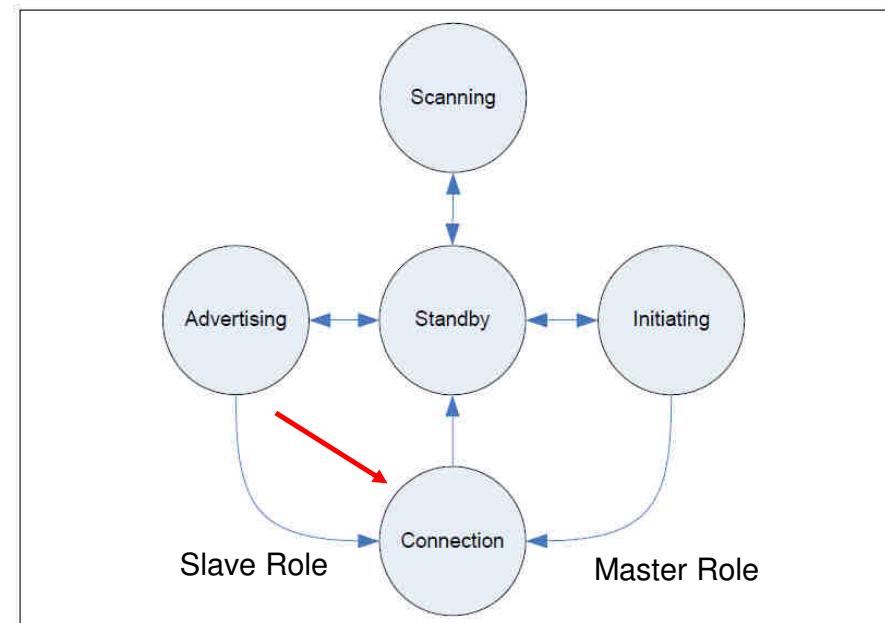
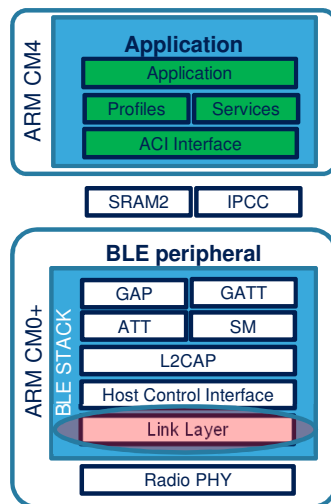
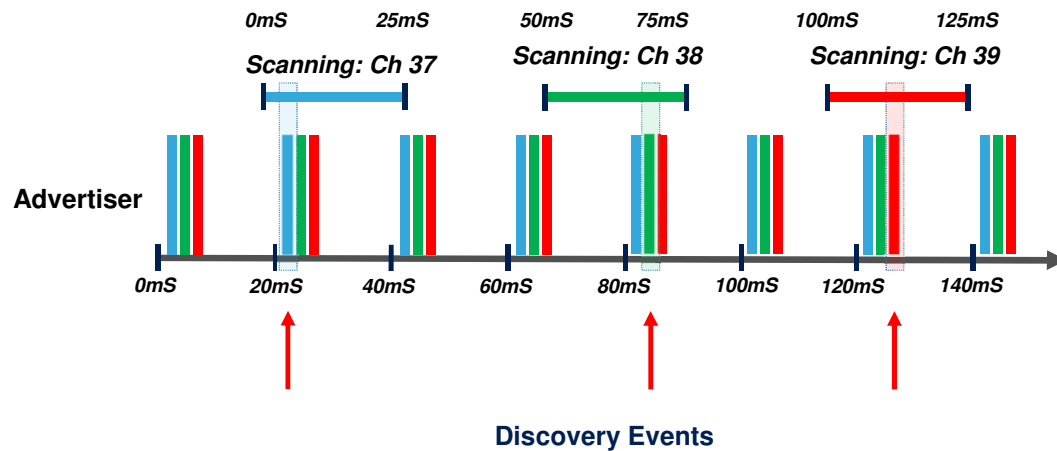


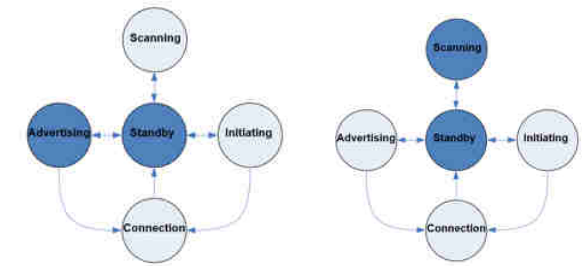
Figure 1.1: State diagram of the Link Layer state machine

Discovery: Advertising & Scanning

68



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



Advertiser Settings:

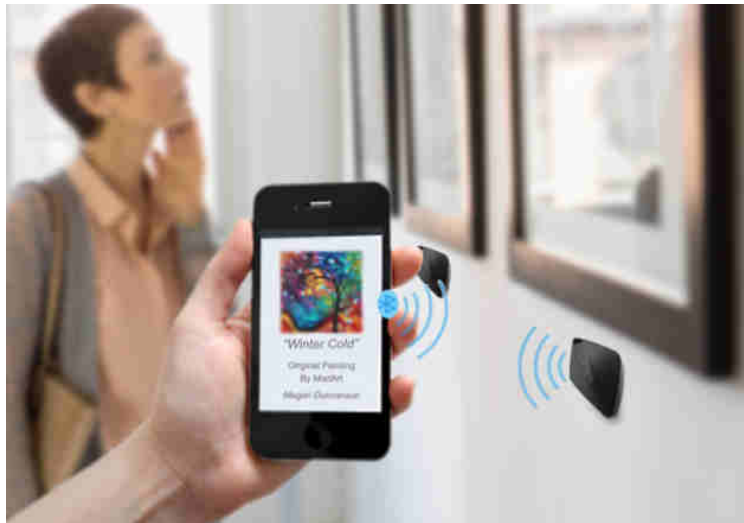
- Advertising Interval: 20mS

Scanner Settings:

- Scan Interval: 50mS
- Scan Window: 25mS

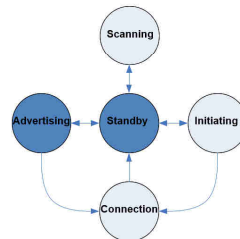
Roles and Modes

- Advertising Mode
- Connected Mode



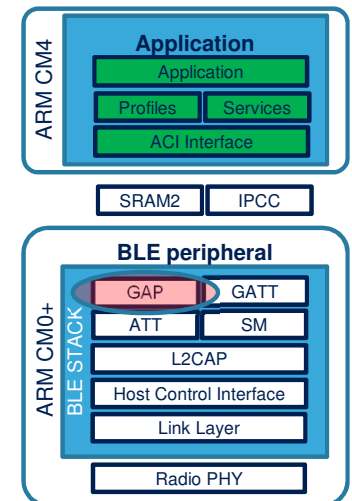
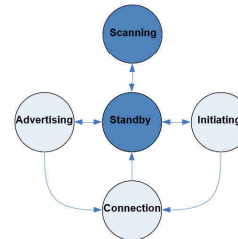
Broadcaster

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



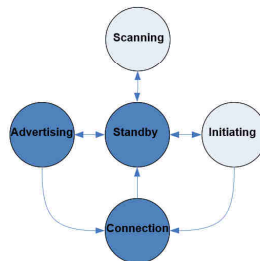
Roles and Modes

- Advertising Mode
- Connected Mode



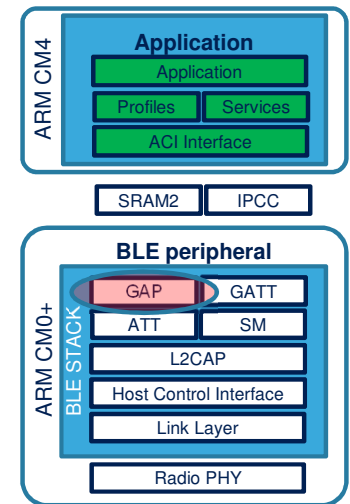
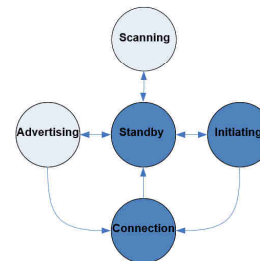
Peripheral

Has transmitter and receiver
Always slave
Connectable advertising



Central

Has transmitter and receiver
Always master
Never advertises

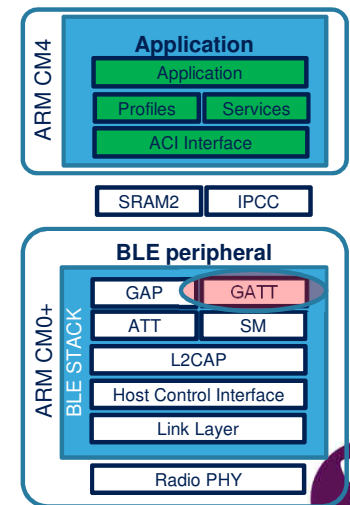


GATT (Generic Attribute Profile)

71

GAP Central is also a “GATT Client”

GAP Peripheral is also a “GATT Server”

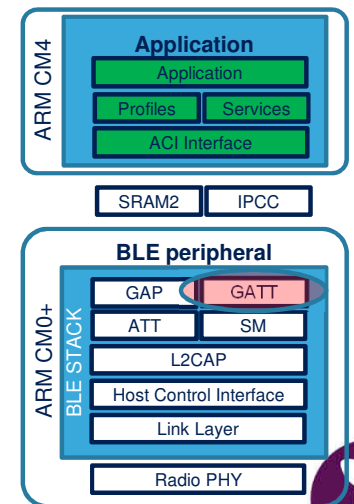


GATT (Generic Attribute Profile)

72

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

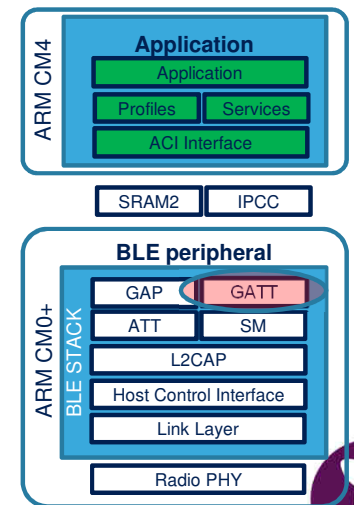


What is my heartrate?

147 bpm

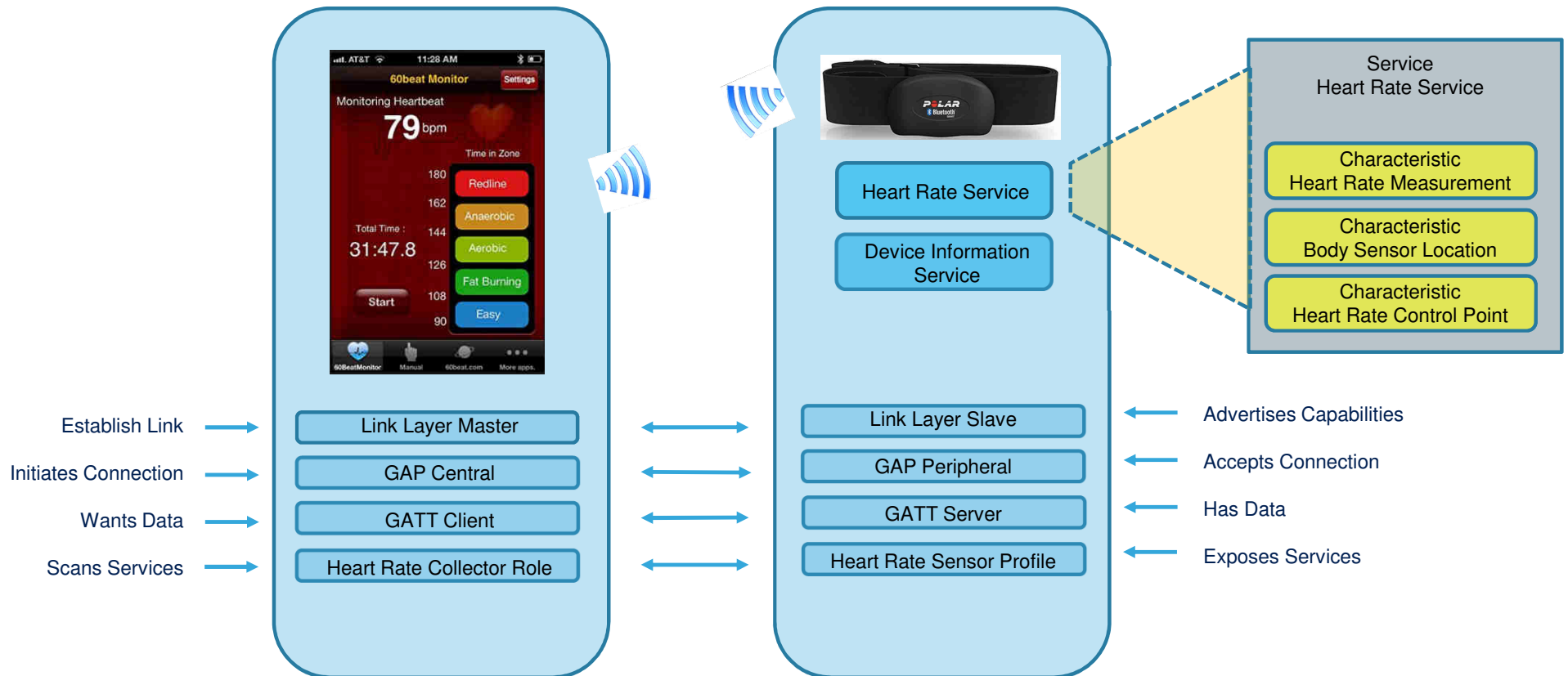
What is your Mfr ID?

Polar

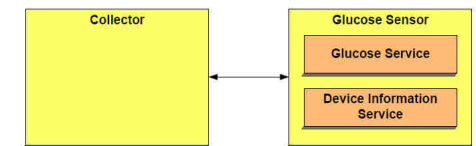
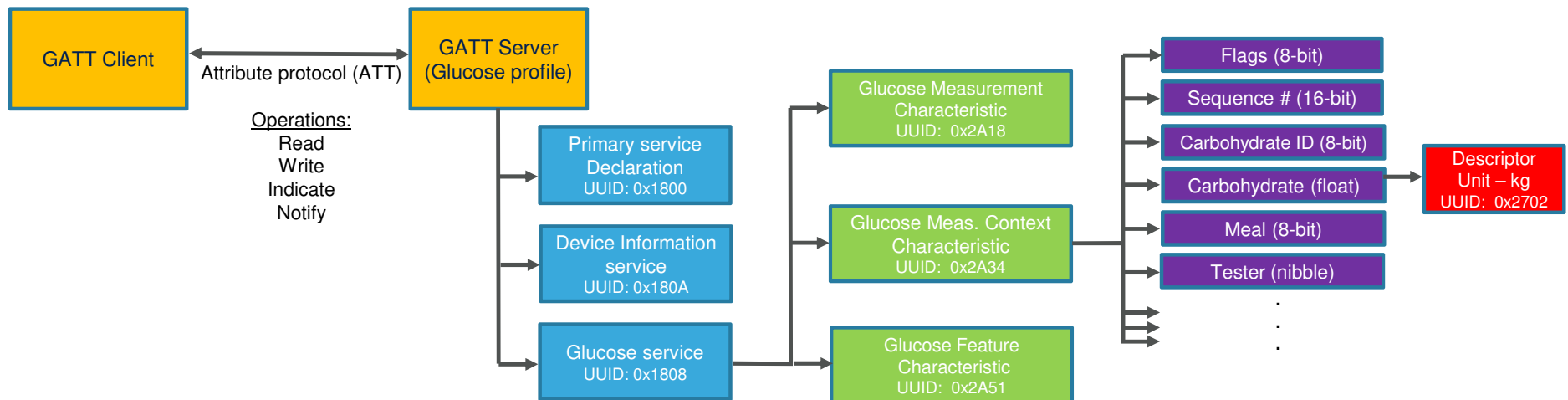


GATT: Profiles, Services, Characteristics & Descriptors

74



GLP Profile defines two roles: Collector & Glucose Sensor



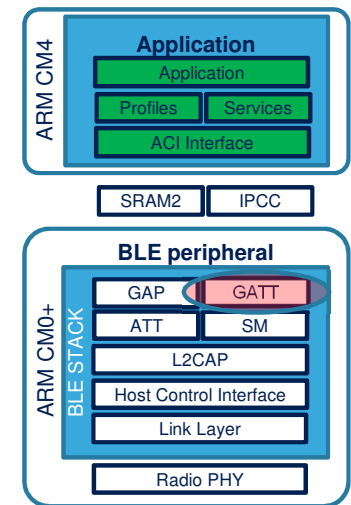
GATT Database details – Handles, UUID's & Values

76

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



Attribute protocol details (ATT)

77

- Access GATT database information on the *Server*

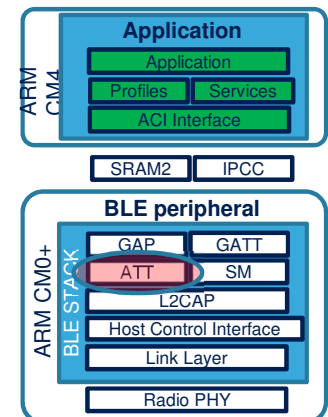
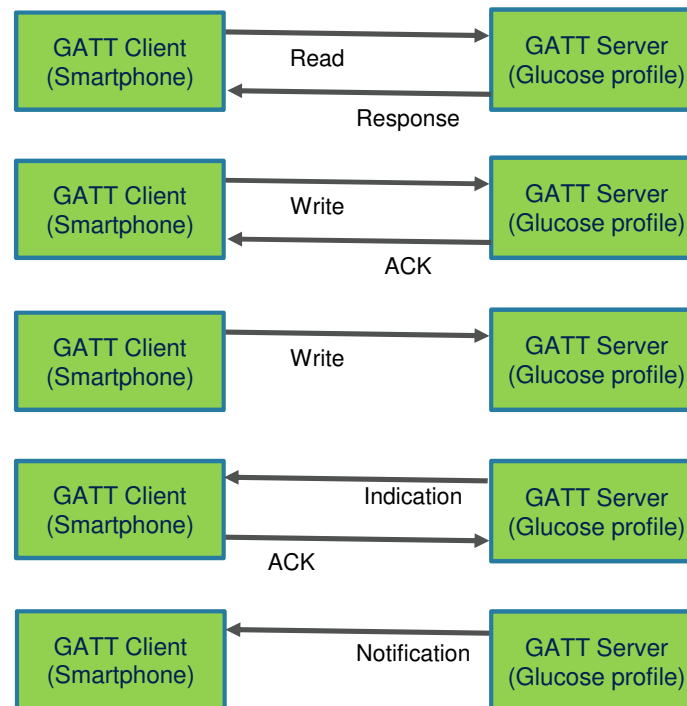
- Operations

- Read
- Write / Write without response
- Indicate / Notify

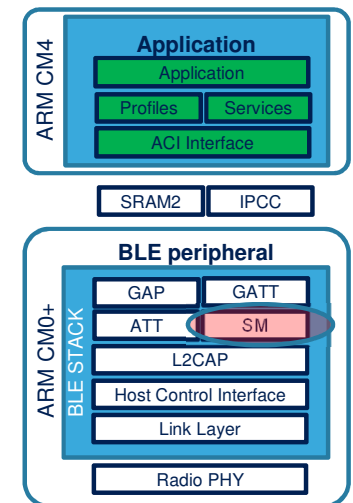
- Four elements

- 16-bit **Handle**
- **Type** of attribute (UUID)
- **Value**
- Attribute **Permissions** (Read-only, etc)

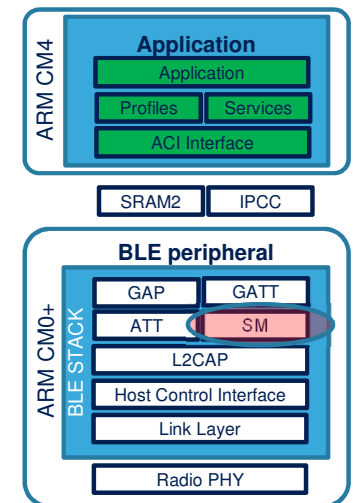
Handle	UUID	Description	Value
0x0100	0x2800	Thermometer service definition	UUID 0x1816
0x0101	0x2803	Characteristic: temperature	UUID 0x2A2B Value handle: 0x0102
0x0102	0x2A2B	Temperature value	20 degrees
0x0104	0x2A1F	Descriptor: unit	Celsius
0x0105	0x2902	Client characteristic configuration descriptor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



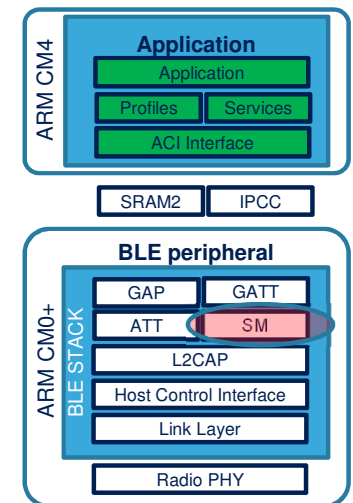
- **Connection:** GAP Central connected to a GAP Peripheral (Connection interval = 7.5ms to 4 secs)
- **Pairing:** Connected devices exchange encryption keys to **encrypt** the link. There are now **paired**.
- **Bonding:** Paired devices can be bonded – Keys are stored for the next connection.
- **Whitelisting:**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



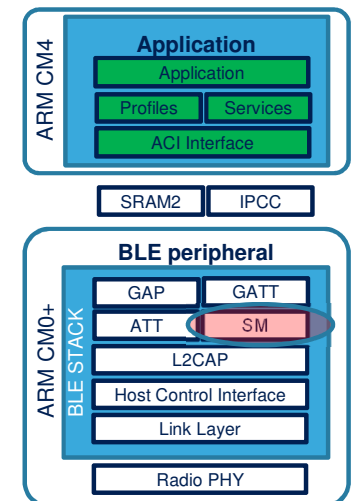
- **Connection:** GAP Central connected to a GAP Peripheral (Connection interval = 7.5ms to 4 secs)
- **Pairing:** Connected devices exchange encryption keys to **encrypt** the link. There are now **paired**.
- **Bonding:** Paired devices can be bonded – Keys are stored for the next connection.
- **Whitelisting:**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

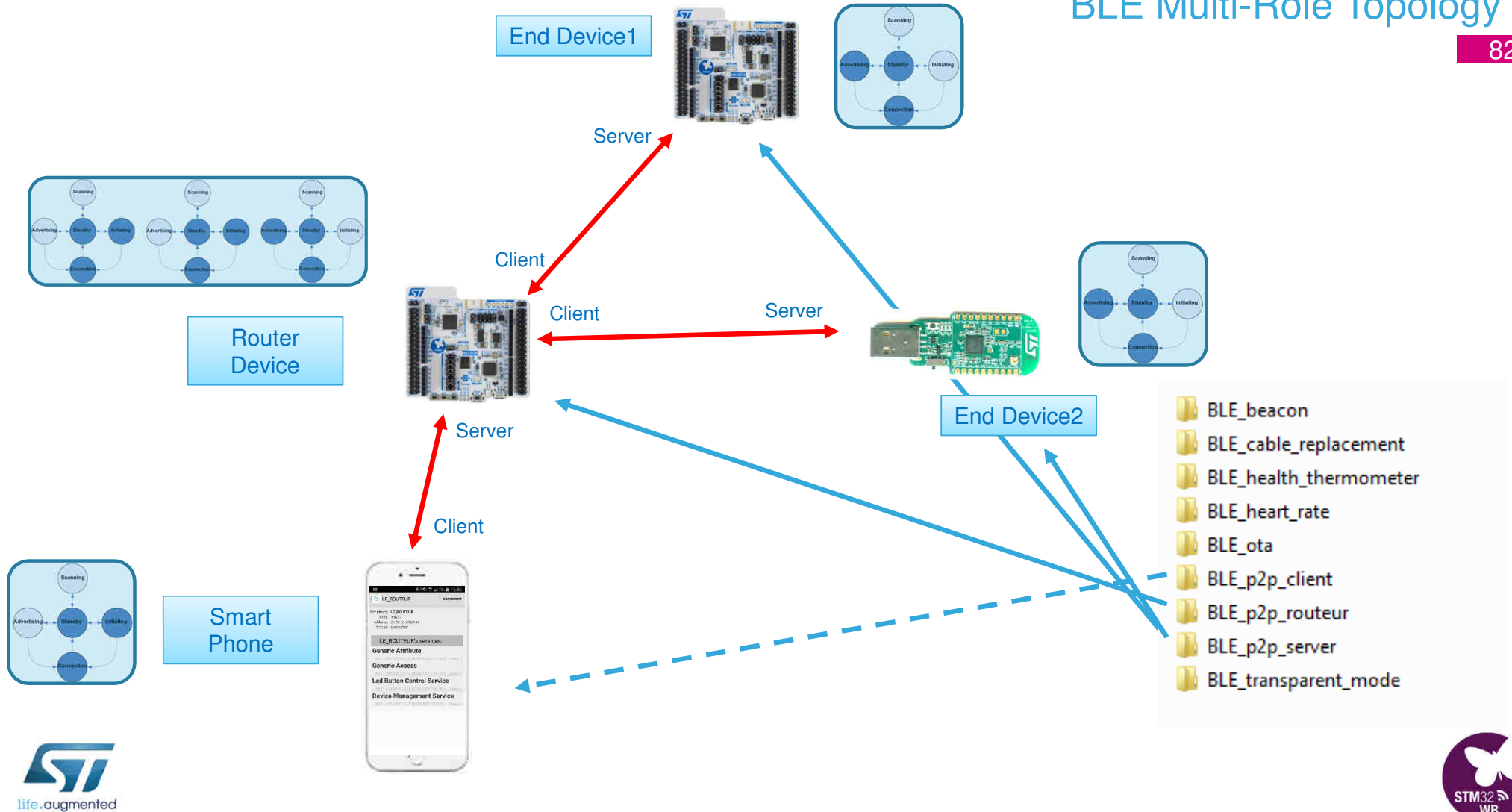


- **Connection:** GAP Central connected to a GAP Peripheral (Connection interval = 7.5ms to 4 secs)
 - **Pairing:** Connected devices exchange encryption keys to **encrypt** the link. There are now **paired**.
 - **Bonding:** Paired devices can be bonded – Keys are stored for the next connection.
 -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

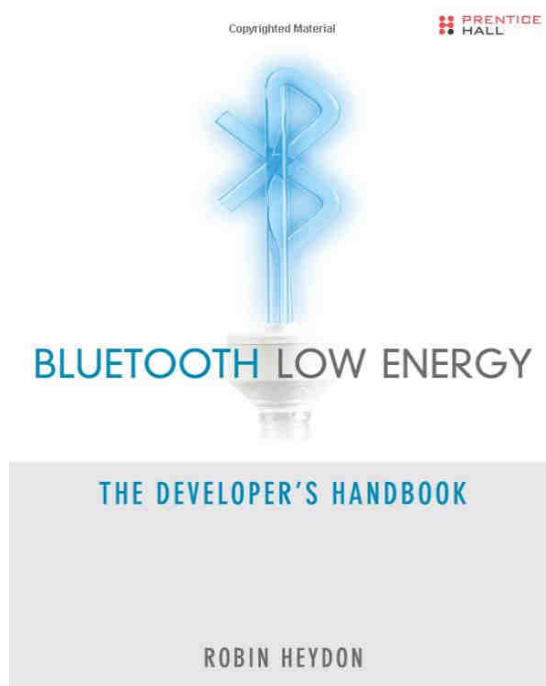


BLE Multi-Role Topology

82

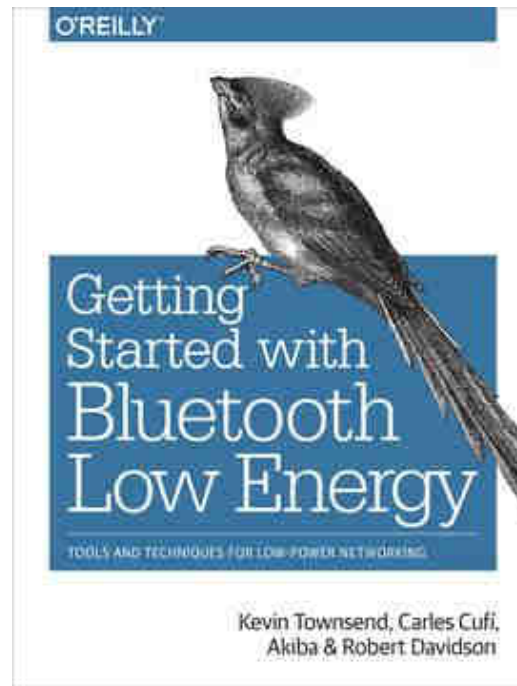


BLE4.0 only but highly detailed



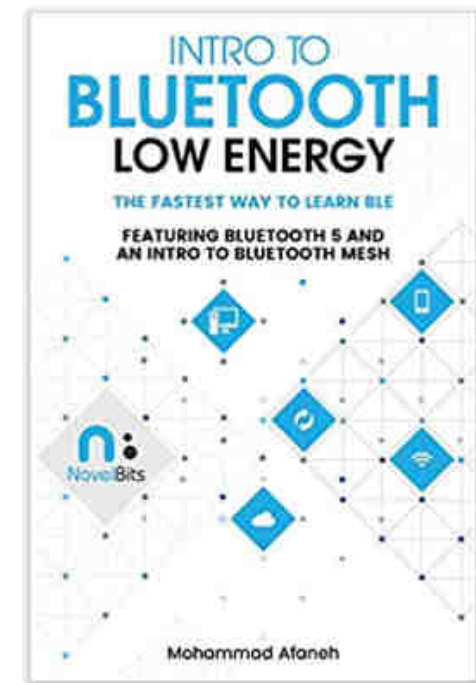
2012

Up to 4.1, 5.0 not covered



2014

Includes 5.0 & Mesh



2018



<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



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

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
BPS	Battery Power Service	1.0	Active	21 Oct 2014	N/A

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

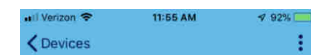
Qualification Test Requirements

Assigned Numbers



Hands-On

Heart-Rate Monitor



< Devices



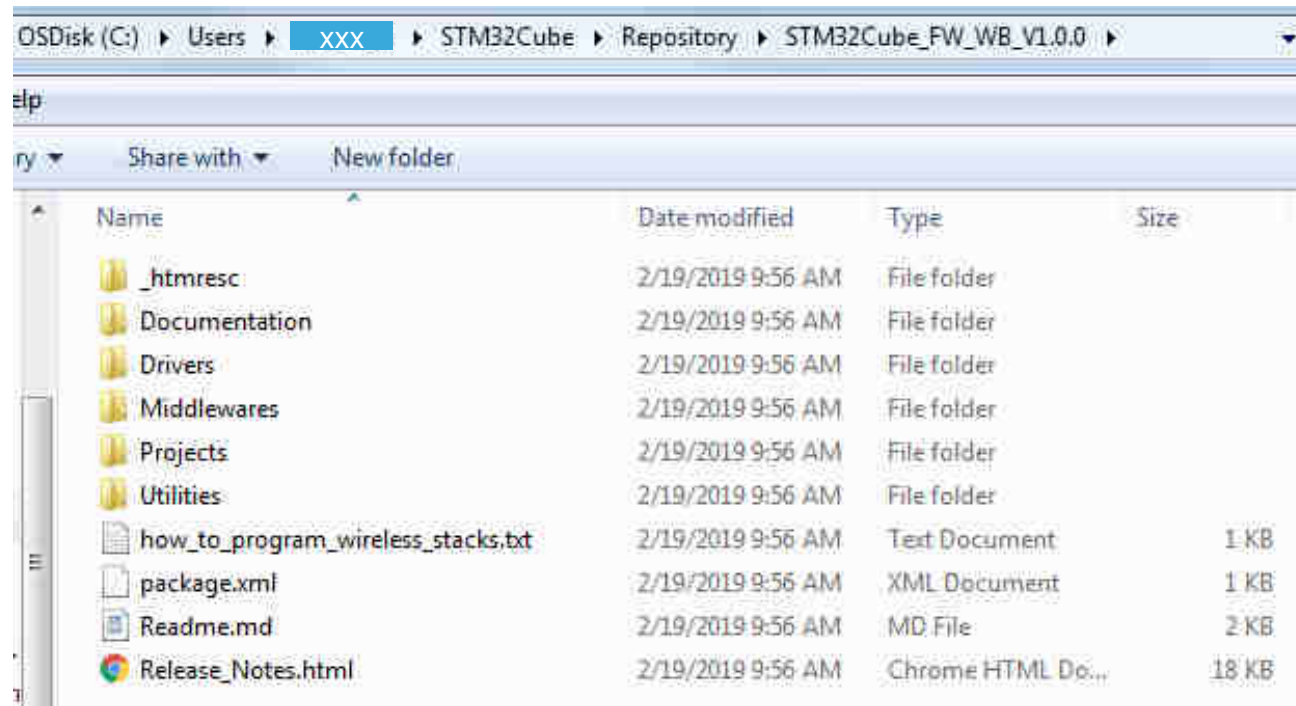
72 bpm

Energy: 20 kJ

RR Interval: 1.00 s

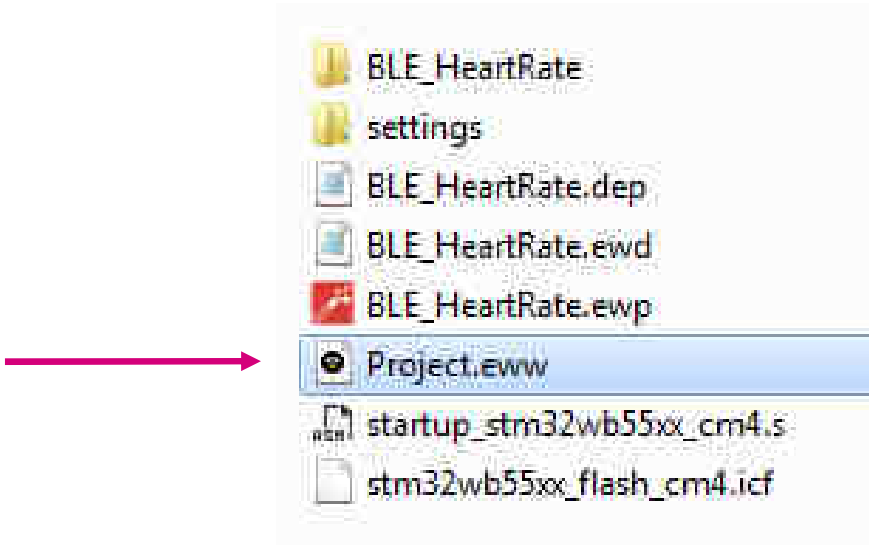


All CubeWB Projects referenced today can be found in the CubeMX Repository folder:



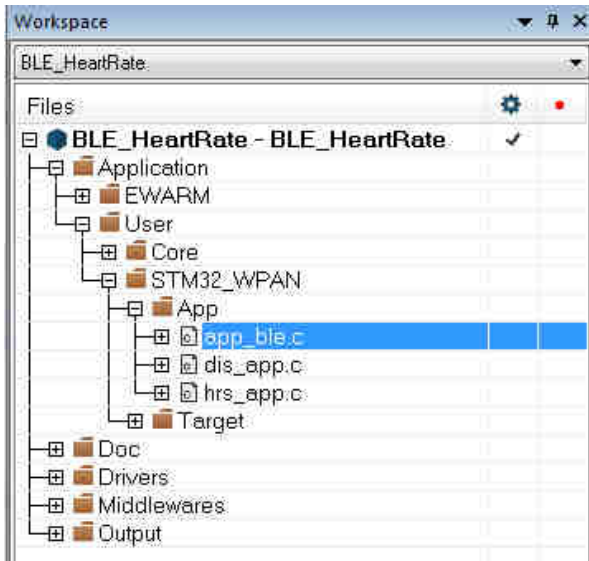
Open the BLE_HeartRate workspace

STMB2Cube_FW_WB_V1.0.0 > Projects > NUCLEO-WB55.Nucleo > Applications > BLE > BLE_HeartRate > EWARM >



- BLE_HeartRate
- settings
- BLE_HeartRate.dep
- BLE_HeartRate.ewd
- BLE_HeartRate.ewp
- Project.eww**
- startup_stm32wb55xx_cm4.s
- stm32wb55xx_flash_cm4.icf

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)

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

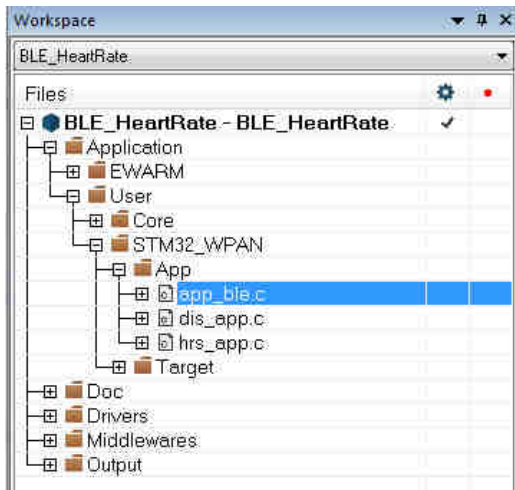


```
229 static const char local_name[] = { AD_TYPE_COMPLETE_LOCAL_NAME, 'S', 'T', 'M', '1', '2'};
```

Hands-On: HRM example

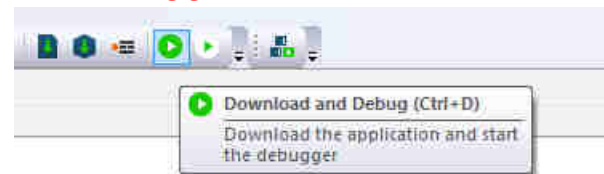
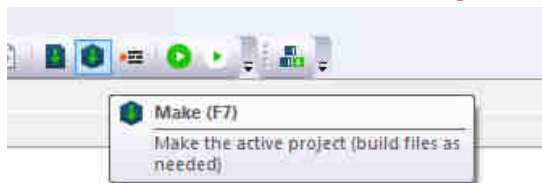
90

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



```
177  /* Private defines -----  
178  #define APPBLE_GAP_DEVICE_NAME_LENGTH 9  
  
630  if (role > 0)  
631  {  
632      const char *name = "STM32WB12";  
633      aci_gap_init(role, 0,
```

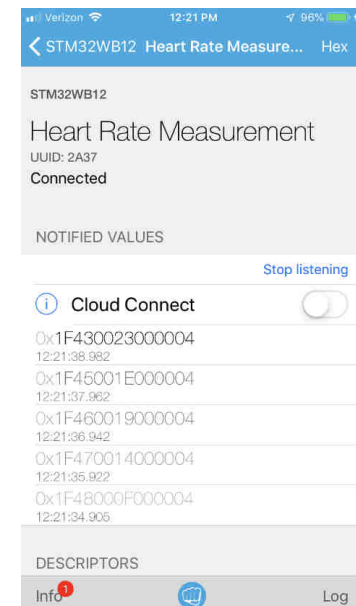
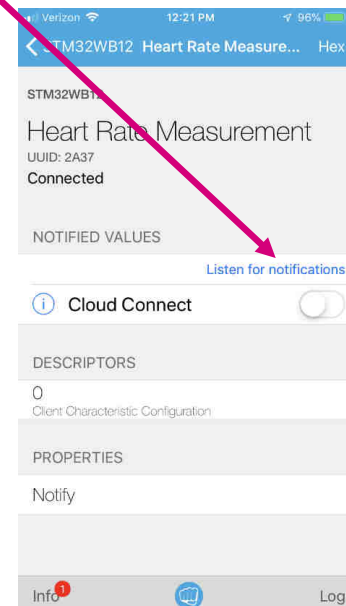
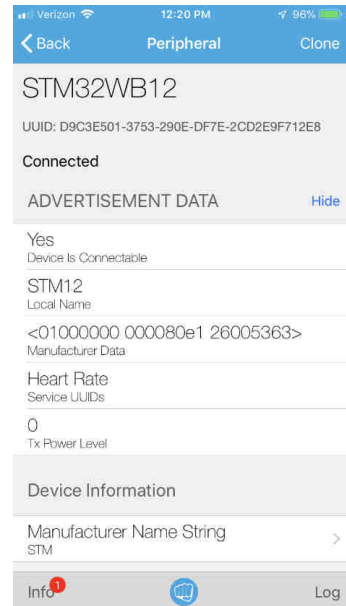
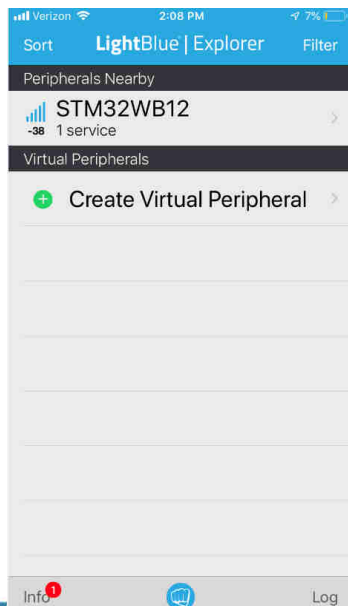
Build, Program and Launch debugger



Hands-On: HRM example

91

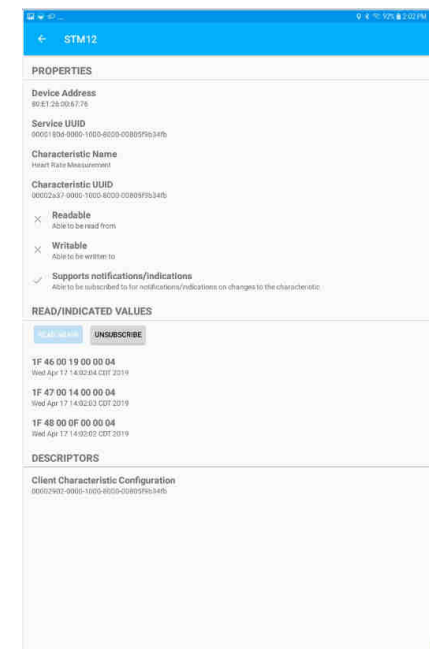
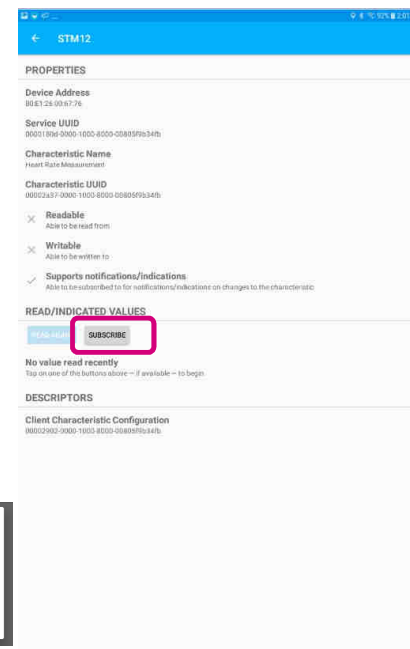
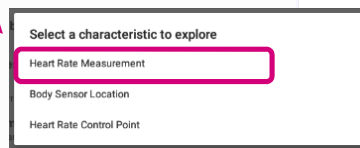
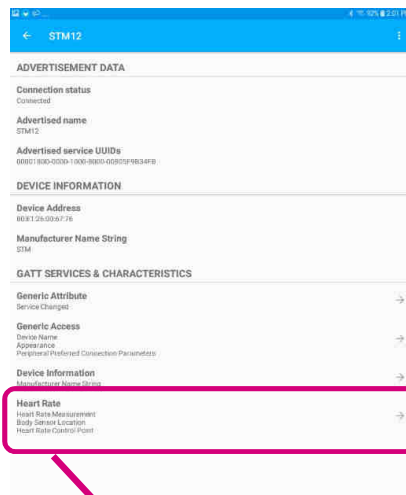
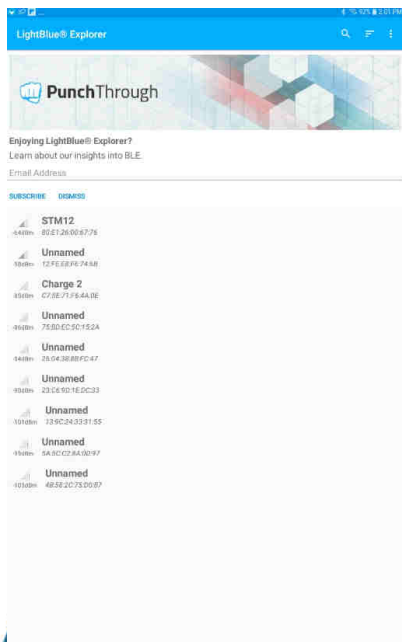
- Open your LightBlue Explorer App on **iOS**
- Find your device and tap on it
- **Show** Advertisement Data
- Click on the Heart Rate Measurement and Enable Notifications



Hands-On: HRM example

92

- 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
- Note your Nucleo Bluetooth Device Address.
 - *Can you find it in the Mfr-Specific advertised data via LightBlue app?*

Verizon 11:46 AM 92%



Connect to a device



About

ST BLE Sensor
Version: 4.2.1
© Copyright 2019 STMicroelectronics.
All rights reserved.



Verizon 11:52 AM 92%

< Back Devices X

STM12
80:E1:26:00:53:63

Hands-On: HRM example

Verizon 11:56 AM 92%

< Devices ⋮



72 bpm

Energy: 20 kJ
RR Interval: 1.00 s

Heart Rate Cloud Logging Rssi & Battery



WB Architecture

Control	ARM Cortex-M4 FPU/DPS 64MHz	Memory
Power supply: 1.71V to 3.6V w/ DC/DC + LDO PDR/PDR/PVD/BOR	Nested Vector Interrupt Controller (NVIC)	Up to 1MB Flash
Crystal oscillators: 32MHz (Radio) 32.768KHz (LSE)	Memory Protected Unit (MPU)	Up to 256KB SRAM
Internal RC oscillators: 32 KHz + 4 - 48 MHz + 16 MHz (HSI) + 48MHz +/- 1% acc. over V and T(°C)	JTAG / SW debug	BOOT ROM
RTC / AWU / CSS	ART Accelerator™	Secure boot loader
PLL / FLL	AHB Bus Matrix	Connectivity
SysTick timer	2x DMA 7channels	2x SPI, 2x I2C
2 watchdogs (WWDG / IWDG)	Multi-Protocol Radio	1x USART LIN, smartcard, IrDA, Modem control
Up to 72 I/Os	Bluetooth 5™	1x ULP UART
Cyclic Redundancy Check	IEEE 802.15.4	USB 2.0 FS - Crystal less
Voltage scaling (2 modes)	AES	Quad-SPI (XIP)
Analog	ARM Cortex-M0+ MPU 32MHz	SAI (Full duplex)
2x ULP comparators	Nested Vector Interrupt Controller (NVIC)	Control
1x 12-bit ADC SAR 4.25MSPS	SW debug	4x 16-bit 32-bit timers
Temperature sensor	Security	2x ULP 16-bit timers
	AES 256-bit / PKA TRNG / PCROP	Sensing
		16-keys Capacitive touch
		Display
		8x40 LCD driver

Balun – Combine TX and RX signals

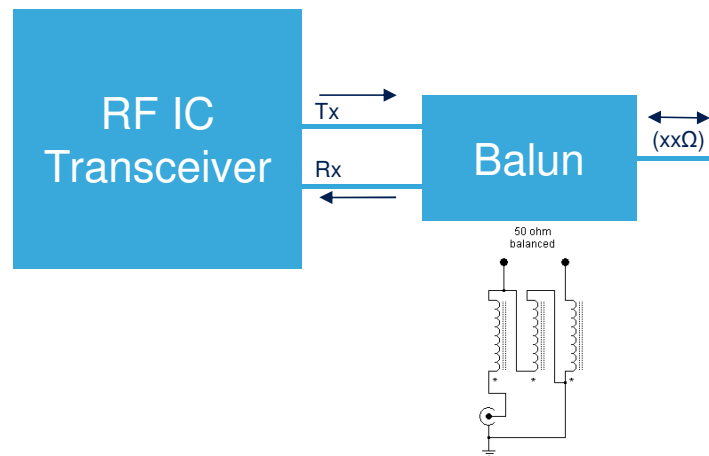
Matching Network – 50 Ω impedance transformation

Harmonic Filter – Reduce out-of-band harmonics

Balun – Combine TX and RX signals

Matching Network – 50 Ω impedance transformation

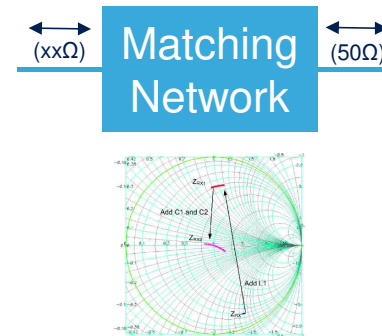
Harmonic Filter – Reduce out-of-band harmonics



Balun – Combine TX and RX signals

Matching Network – 50 Ω impedance transformation

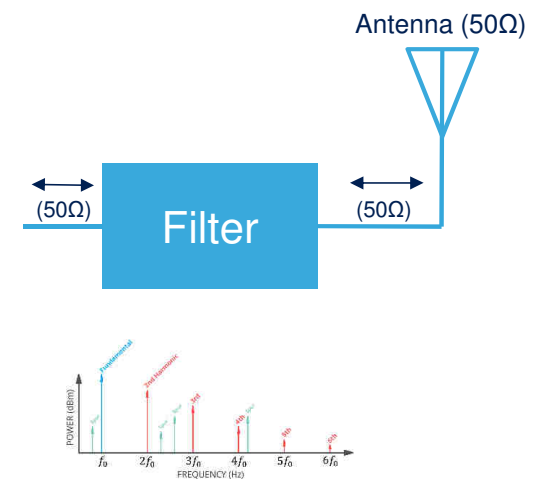
Harmonic Filter – Reduce out-of-band harmonics



Balun – Combine TX and RX signals

Matching Network – 50 Ω impedance transformation

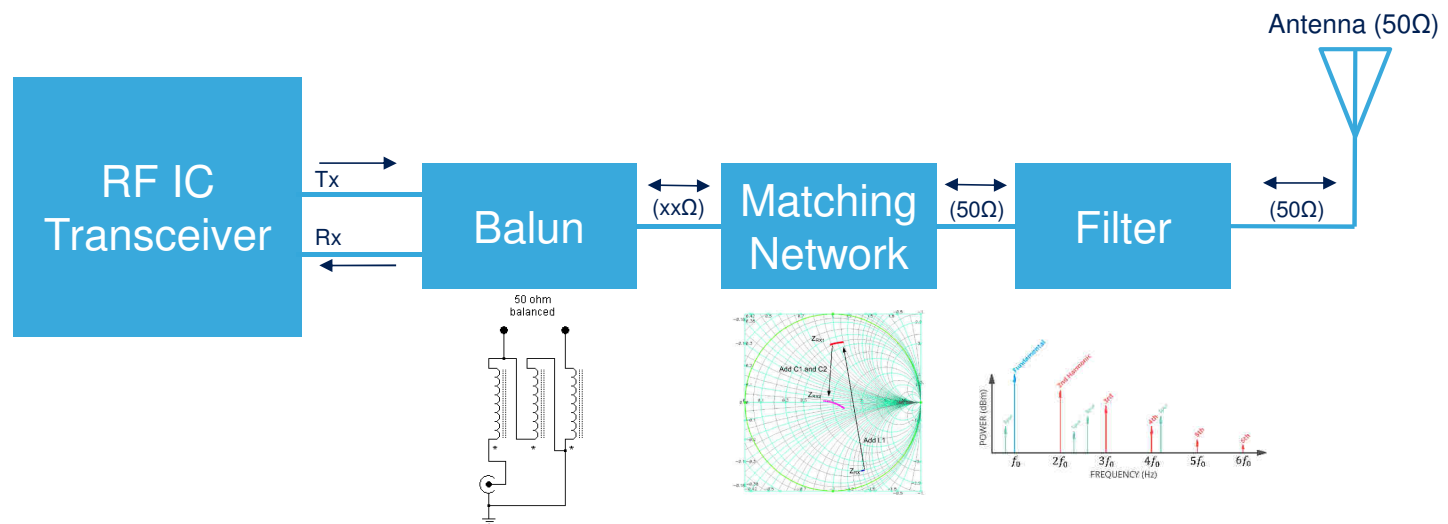
Harmonic Filter – Reduce out-of-band harmonics



Balun – Combine TX and RX signals

Matching Network – 50 Ω impedance transformation

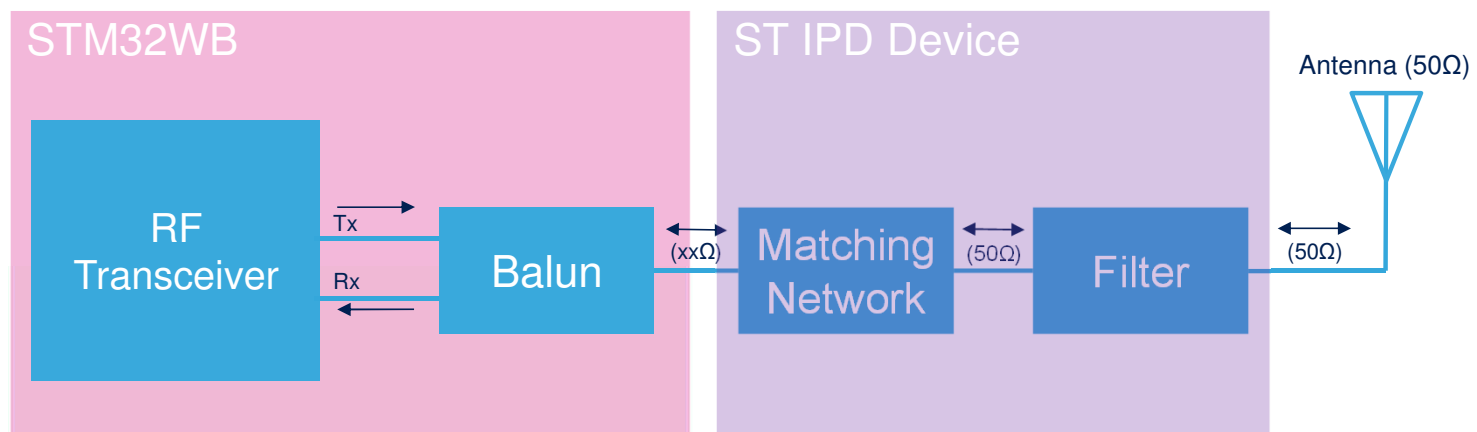
Harmonic Filter – Reduce out-of-band harmonics



Balun – Combine TX and RX signals

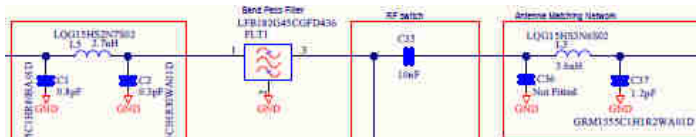
Matching Network – 50 Ω impedance transformation

Harmonic Filter – Reduce out-of-band harmonics



Matching Network + Harmonic Filter

101



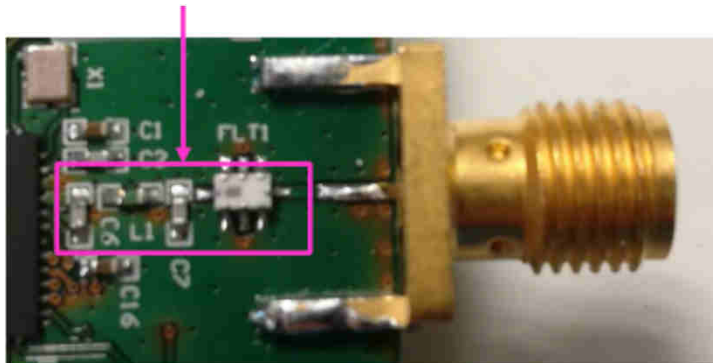
ST IPD Device

Matching Network

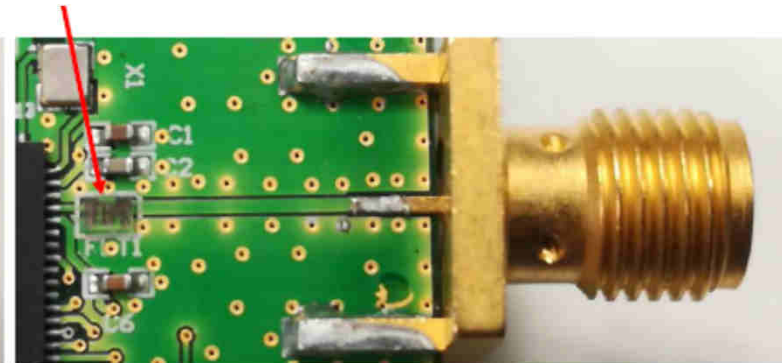
(50Ω)

Filter

Discrete solution



IPD device from ST



MLPF-WB55-01E3

Datasheet

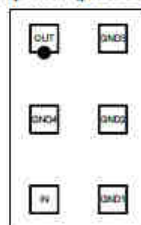
Mass Production NOW

2.4 GHz low pass filter matched to STM32WB55Cx/Rx



Bumpless CSP

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 $\leq 450 \mu\text{m}$
- 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

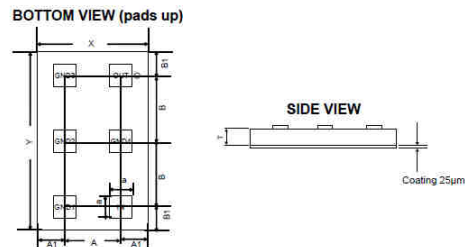
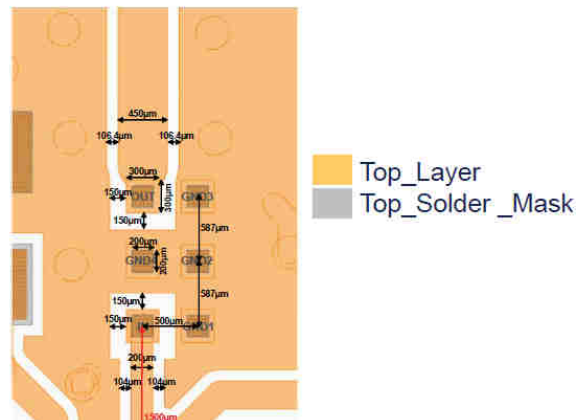


Table 4. Bumpless CSP package mechanical data

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

Figure 13. PCB land pattern recommendations



PCB recommendations included in datasheet



MLPF-WB55-01E3

Datasheet

2.4 GHz low pass filter matched to STM32WB55Cx/Rx

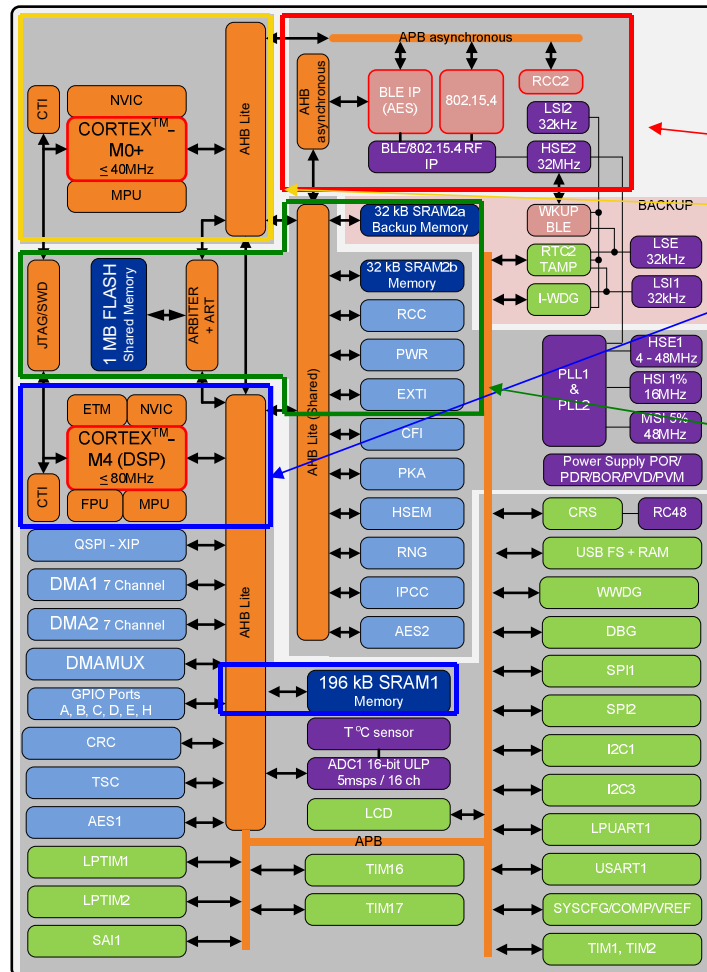


Bumpless CSP

- 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



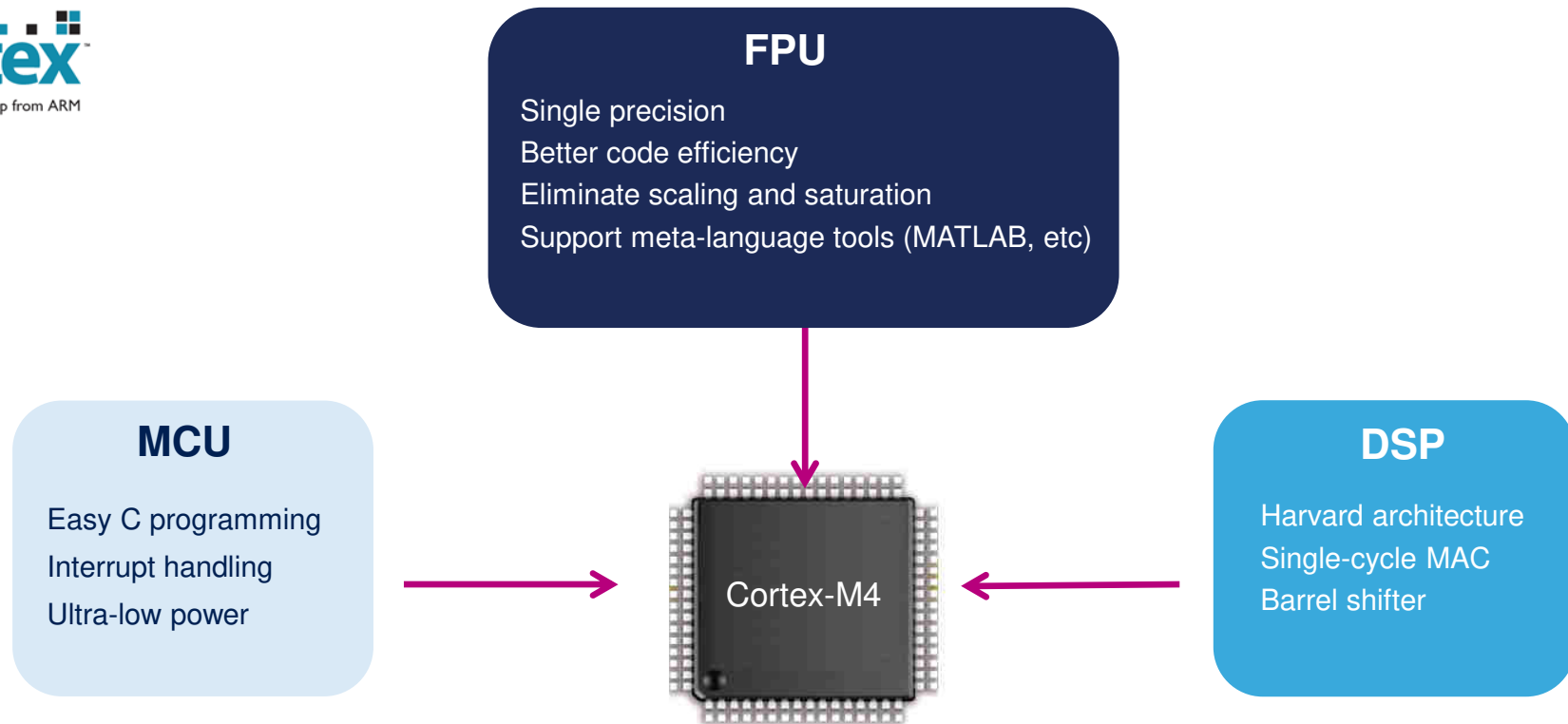
- 3 autonomous sub-systems
 - Radio sub-system
 - Cortex-M0+ (CPU2)
 - Cortex-M4 (CPU1)
- Common run domain
 - Flash, SRAM2, RCC, PWR, EXTI

ARM

Cortex
Low-Power Leadership from ARM

ARM® Cortex® M4 Core

105

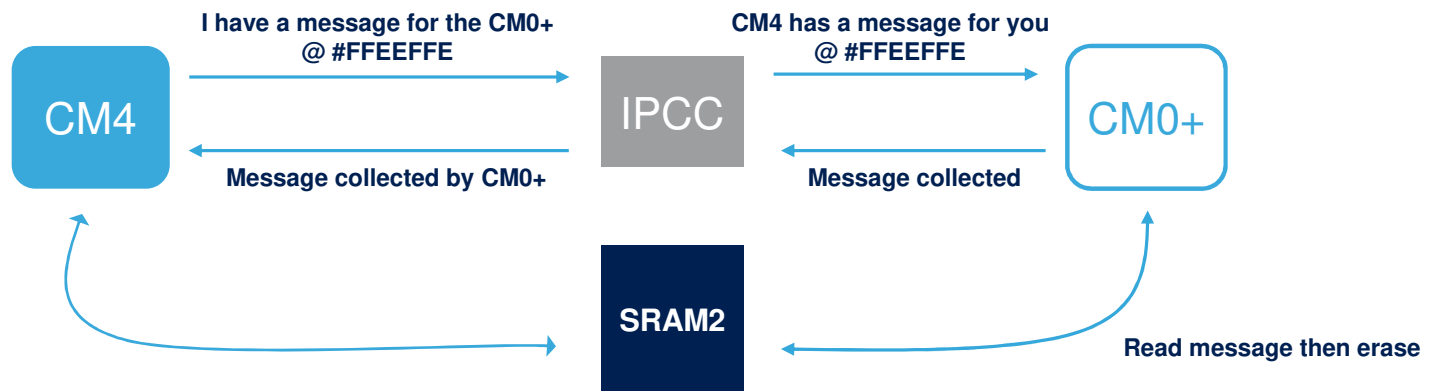


Dual core – How does that work?

106

IPCC: Inter Processor Communication Controller

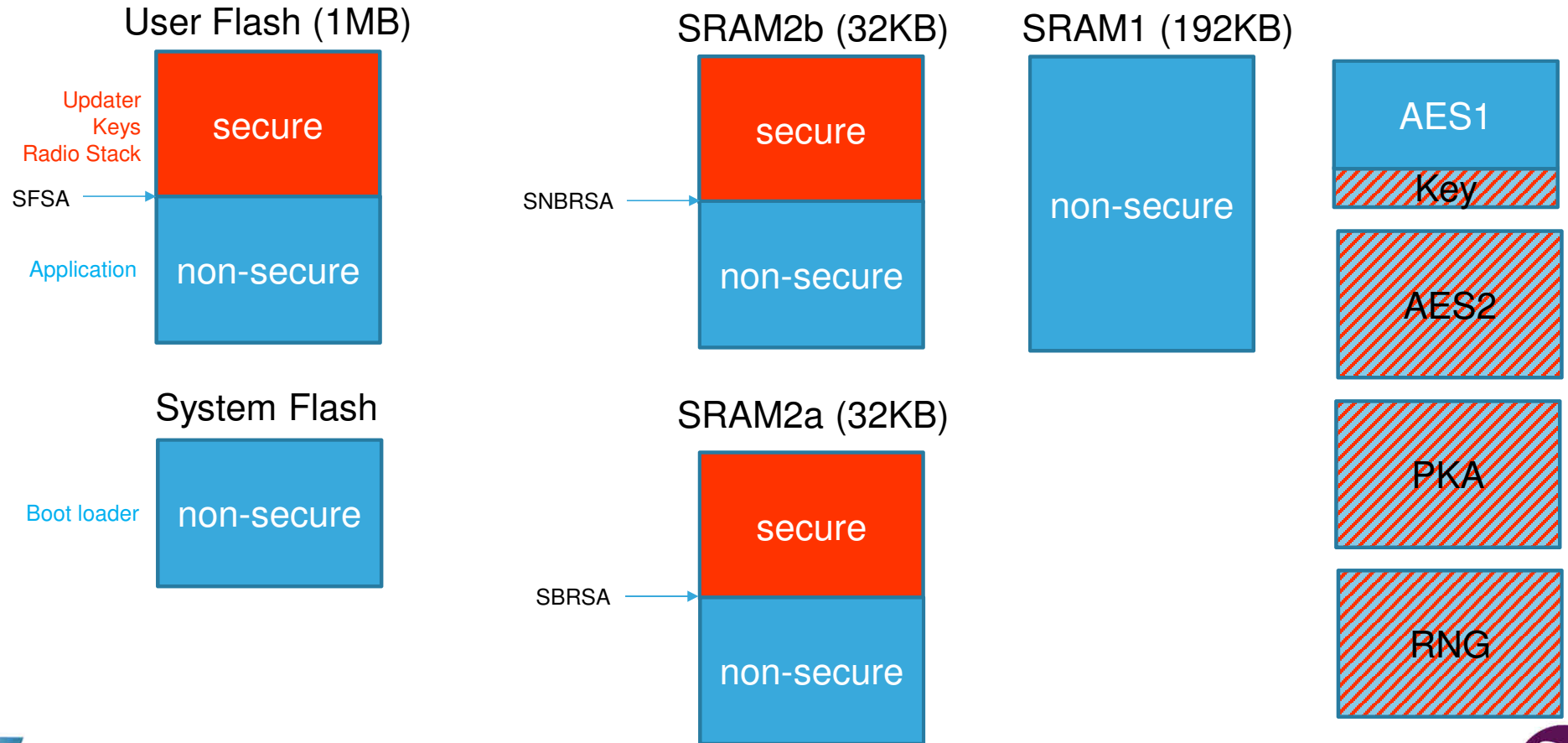
HSEM: Hardware Semaphore – prevent shared resource access conflicts



IPCC works in both directions

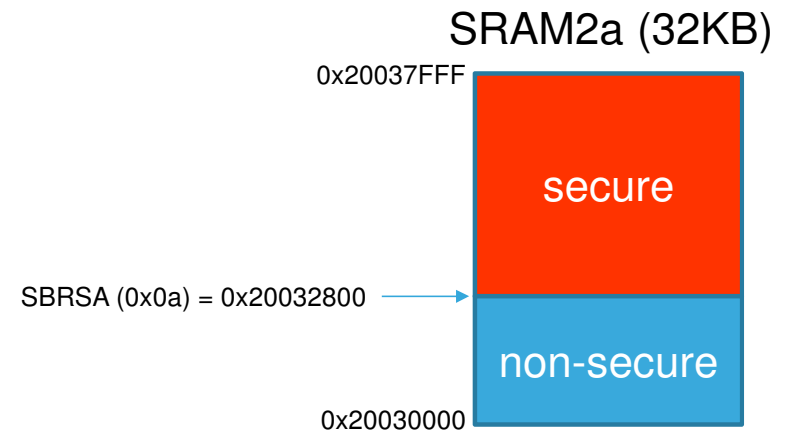
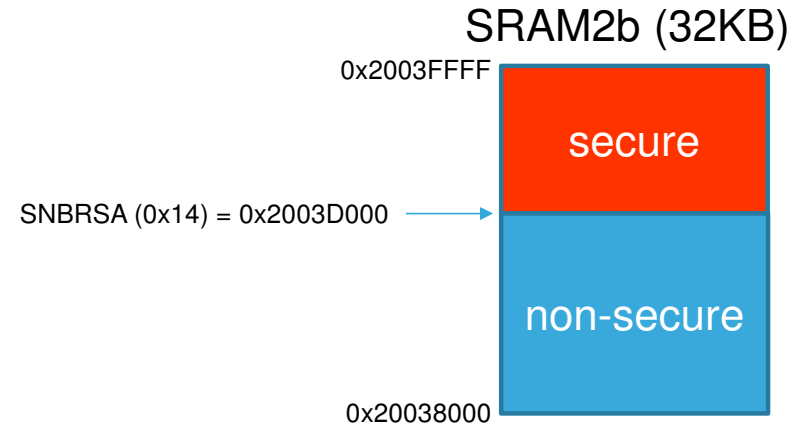
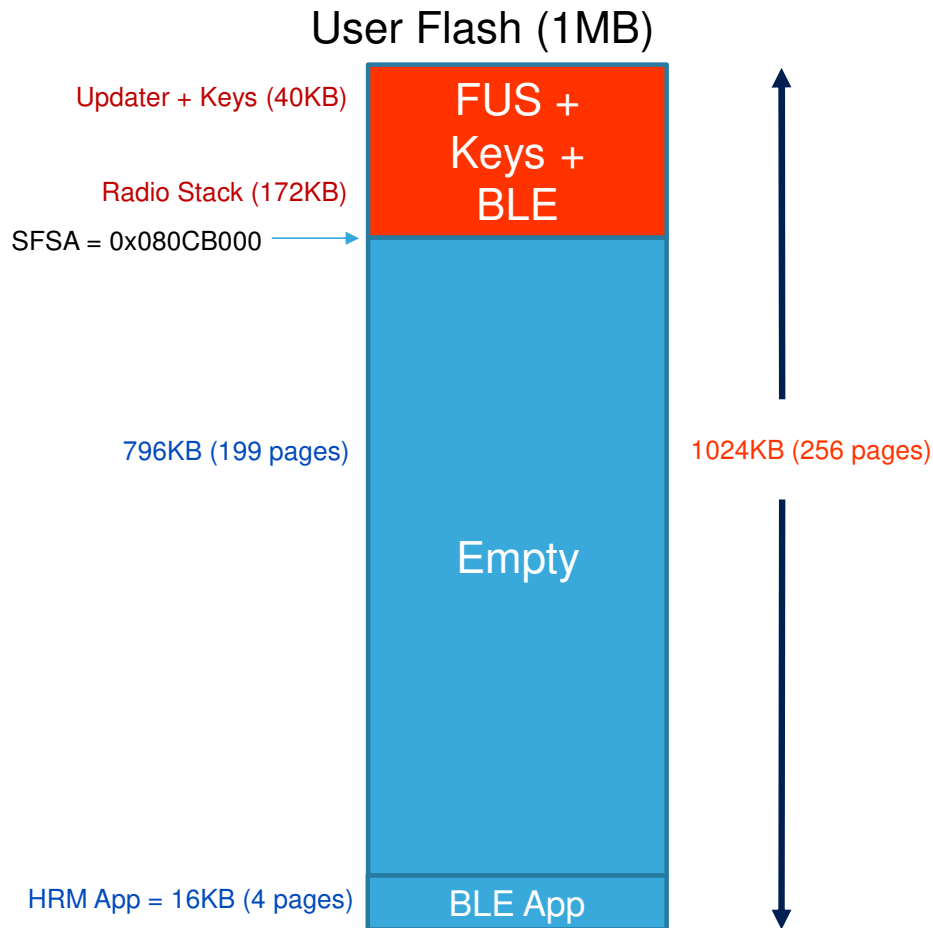
Memory Partitioning

107

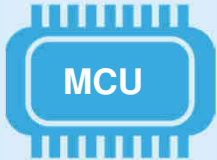


 Securable by register bit

Memory Partitioning: BLE Stack



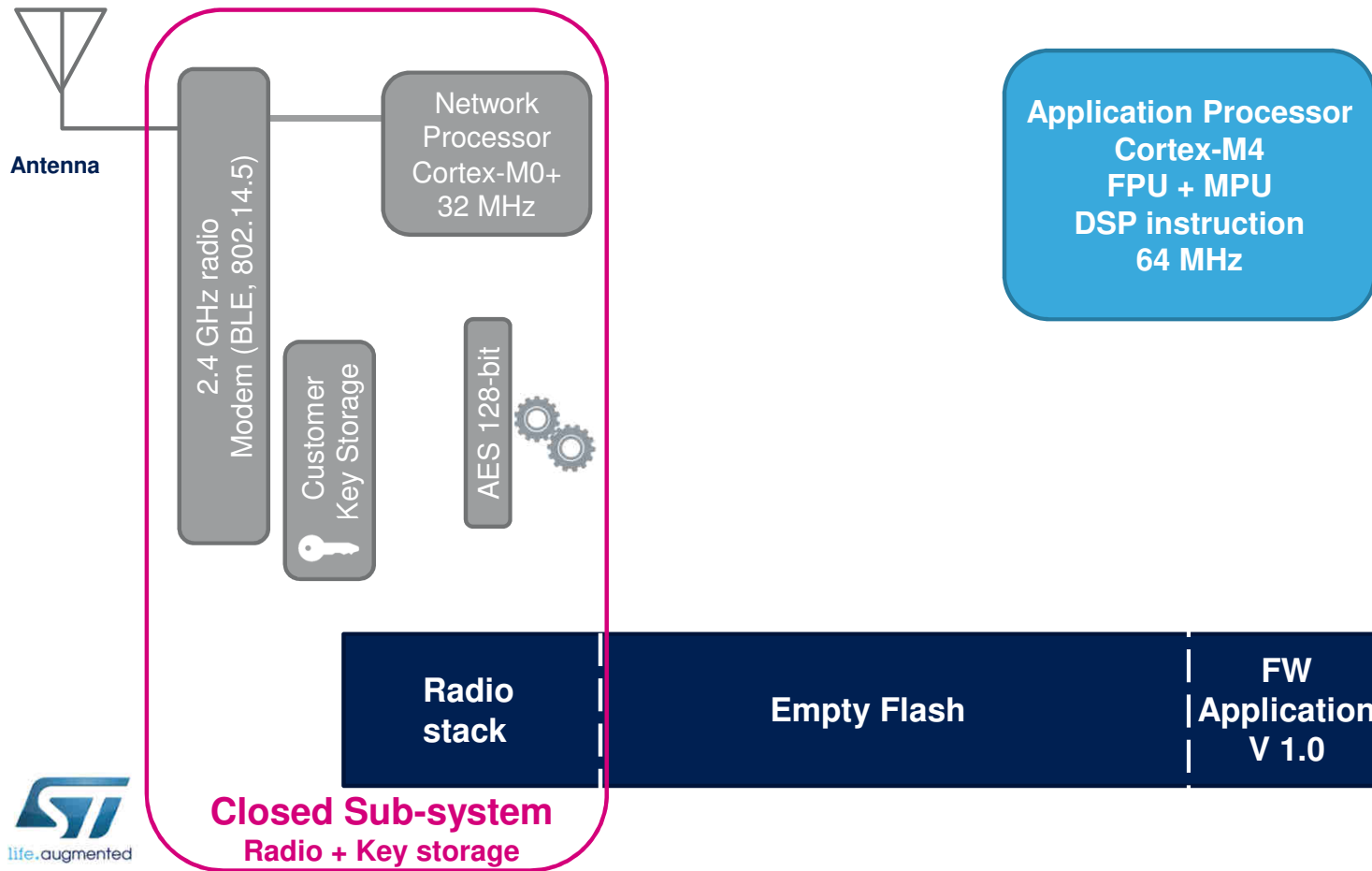


Attacks	Attacks description	STM32WB Countermeasures
Non Invasive 	• Environment • Temp / Voltage / Clocks • Fault injection • Exploit debugger • Side channel • Power Analysis	• Temp sensor • Power supply monitor • Clock security system • Tamper pads • ECC, Parity check • SRAM mass erase • Read out protection • 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



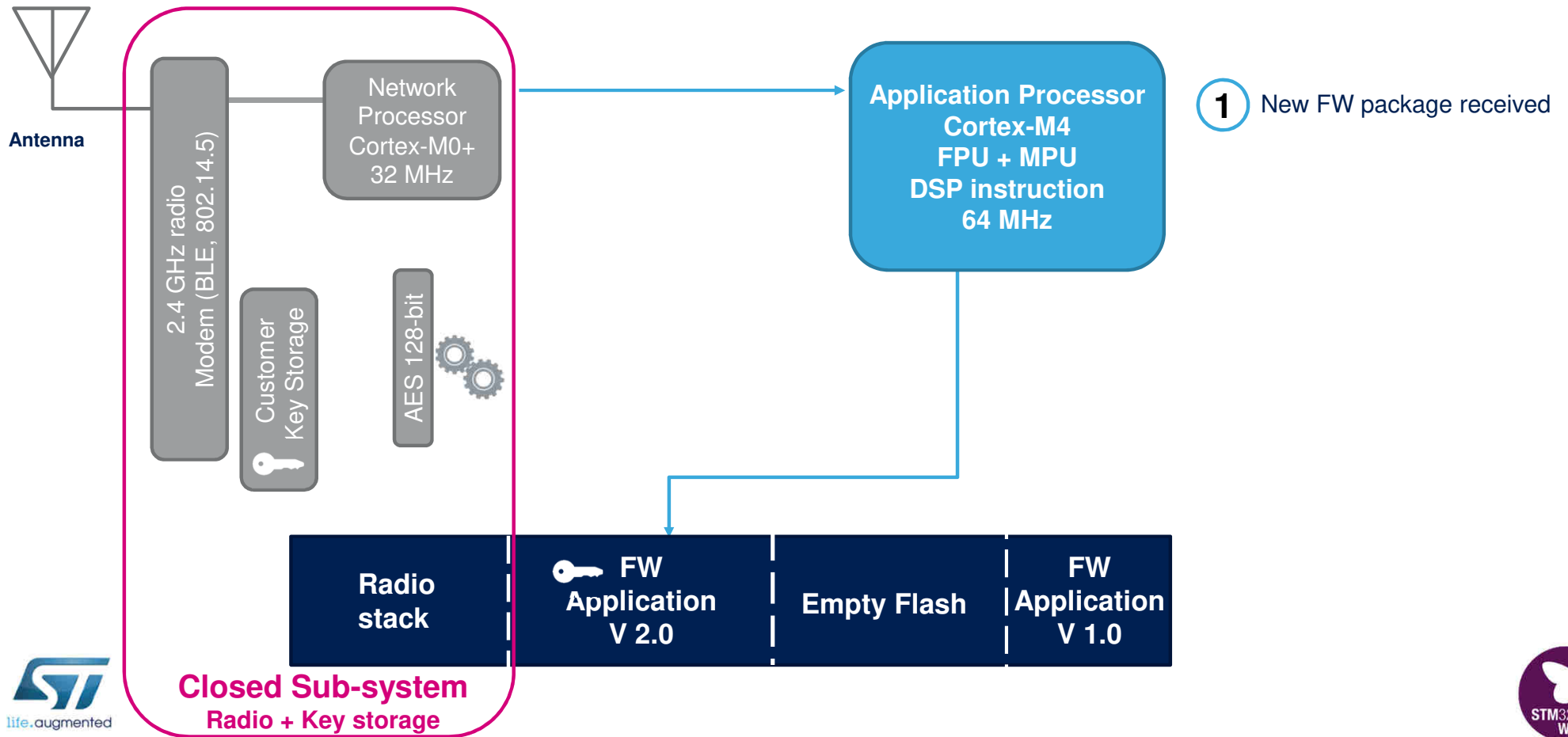
OTA Update Scenario

110



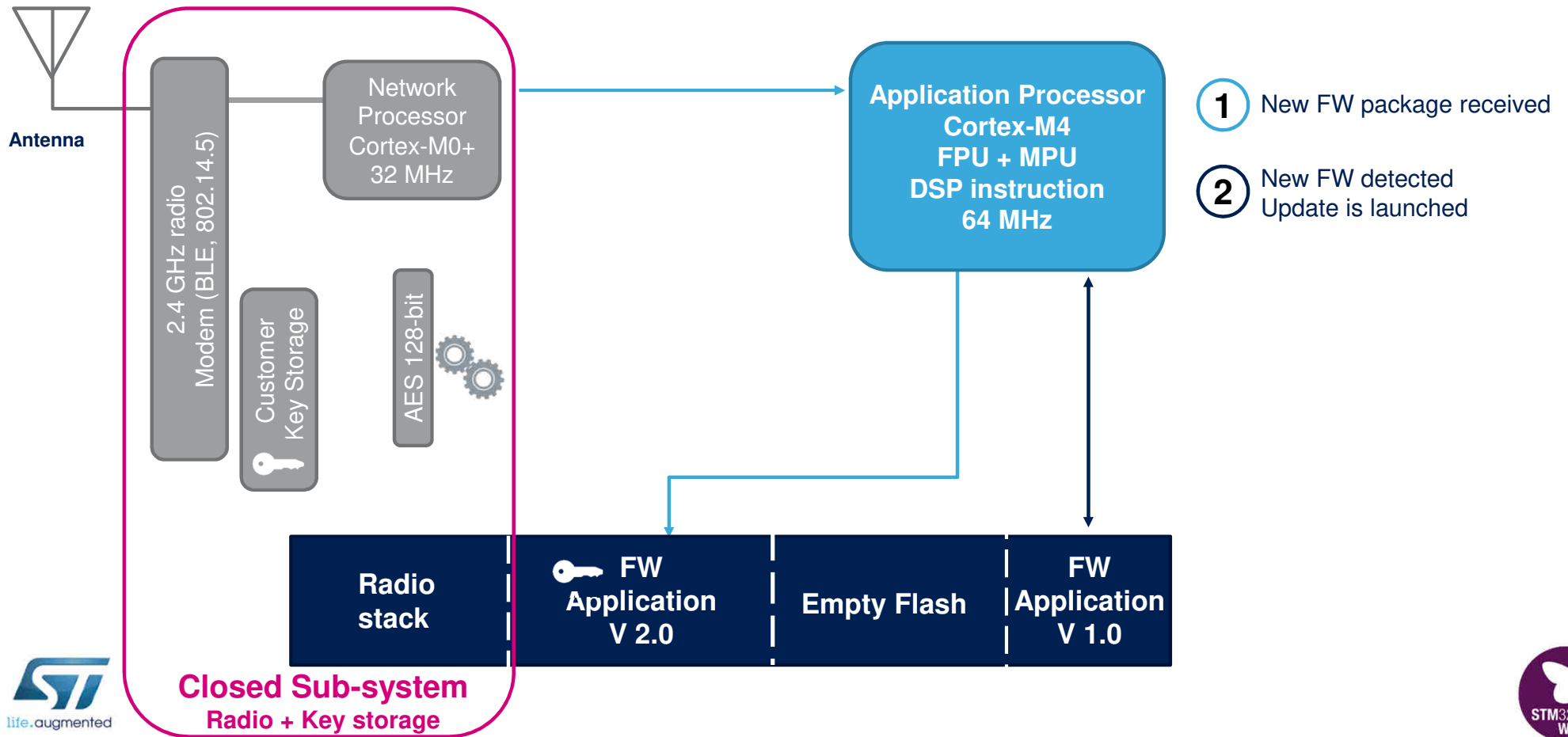
OTA Update Scenario

111



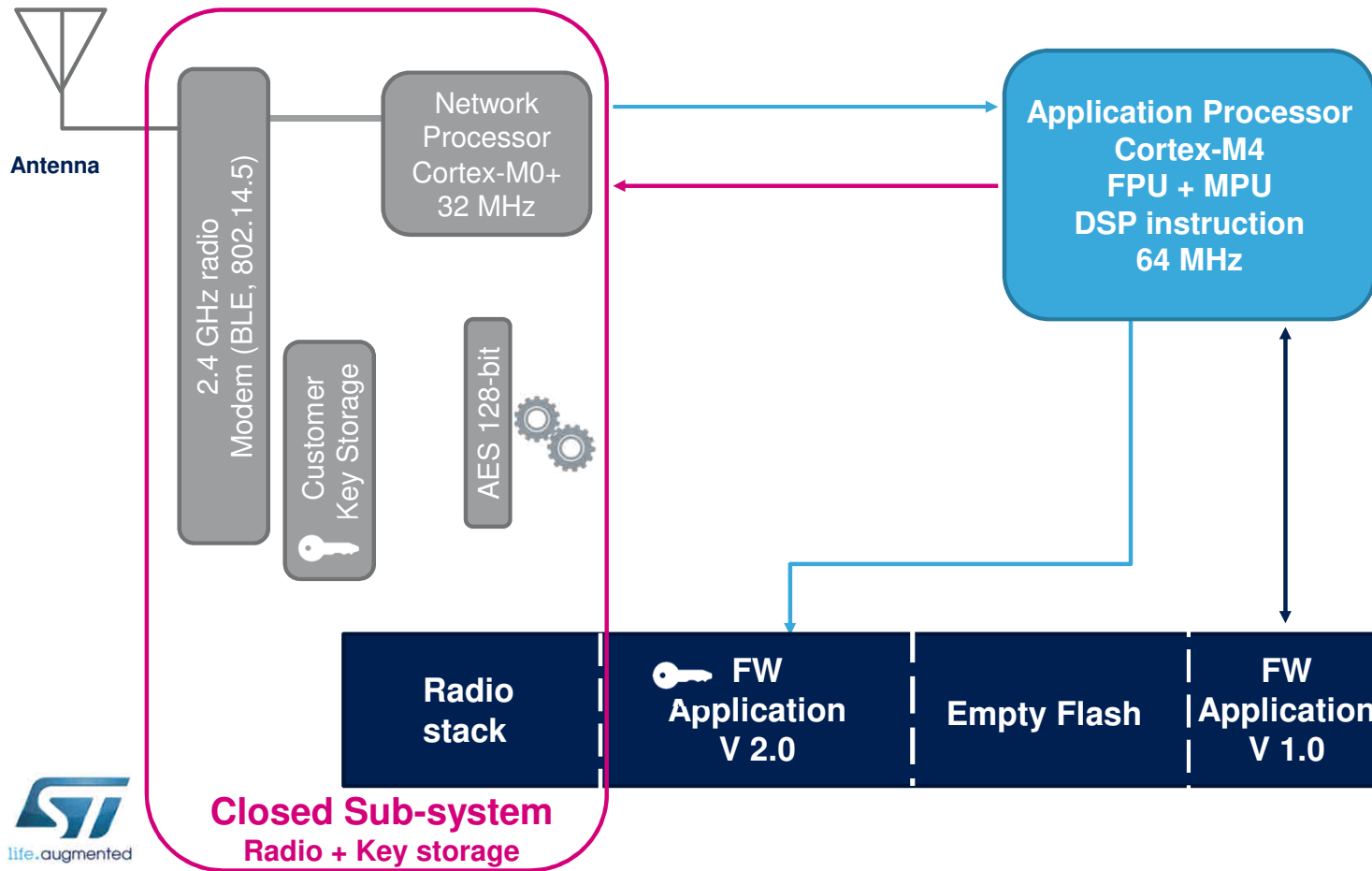
OTA Update Scenario

112



OTA Update Scenario

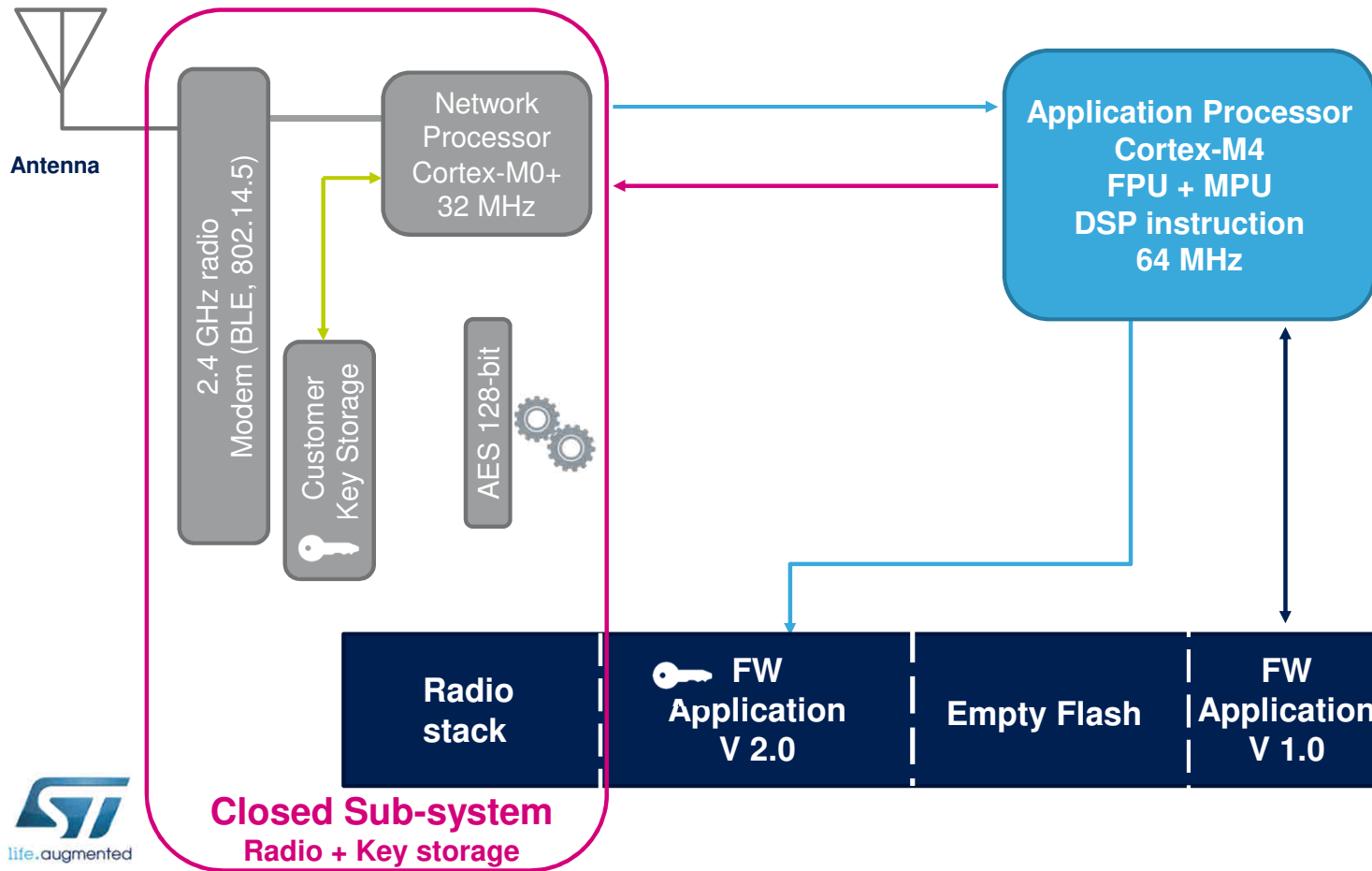
113



- 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

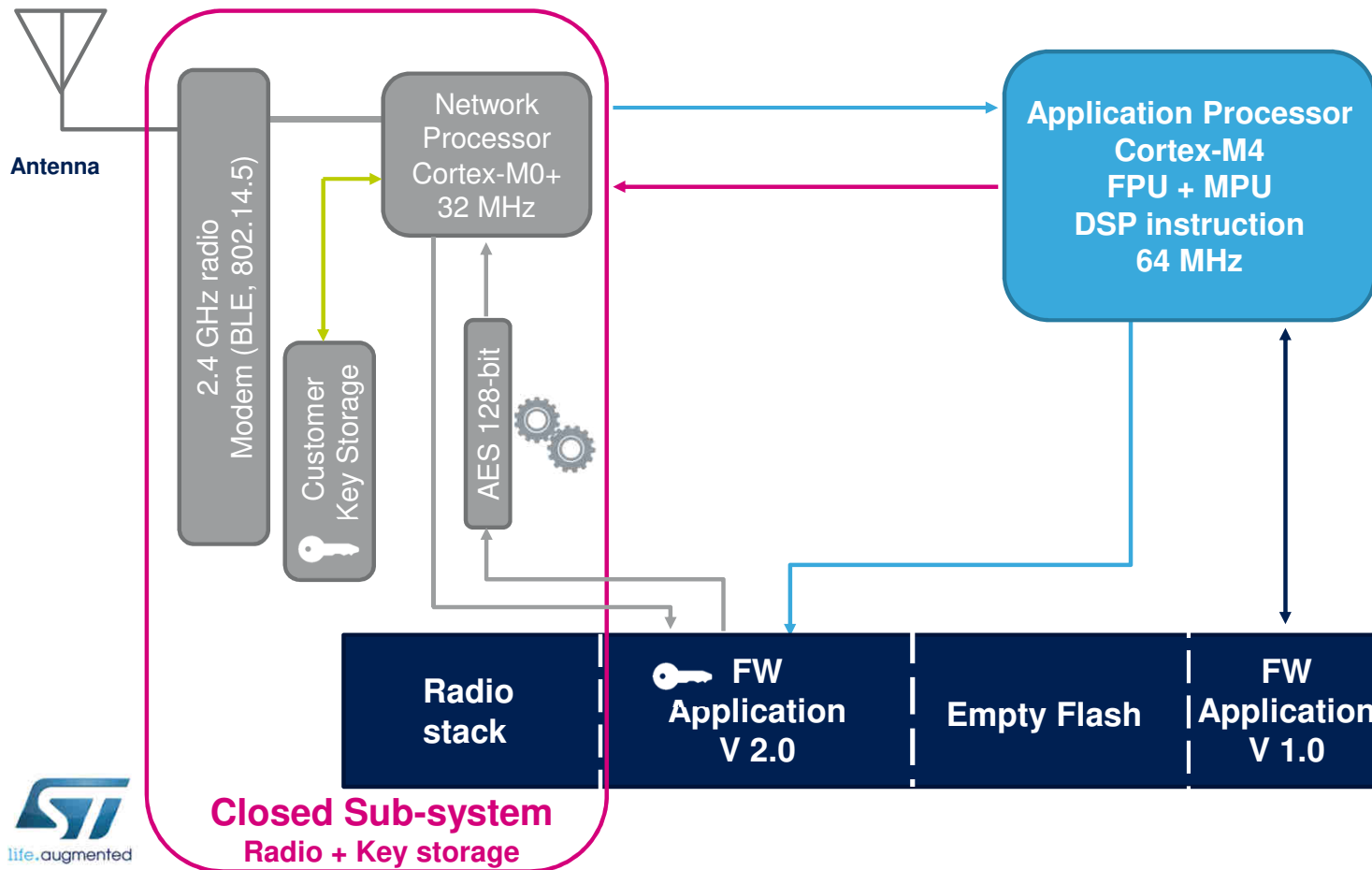
114



- 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

OTA Update Scenario

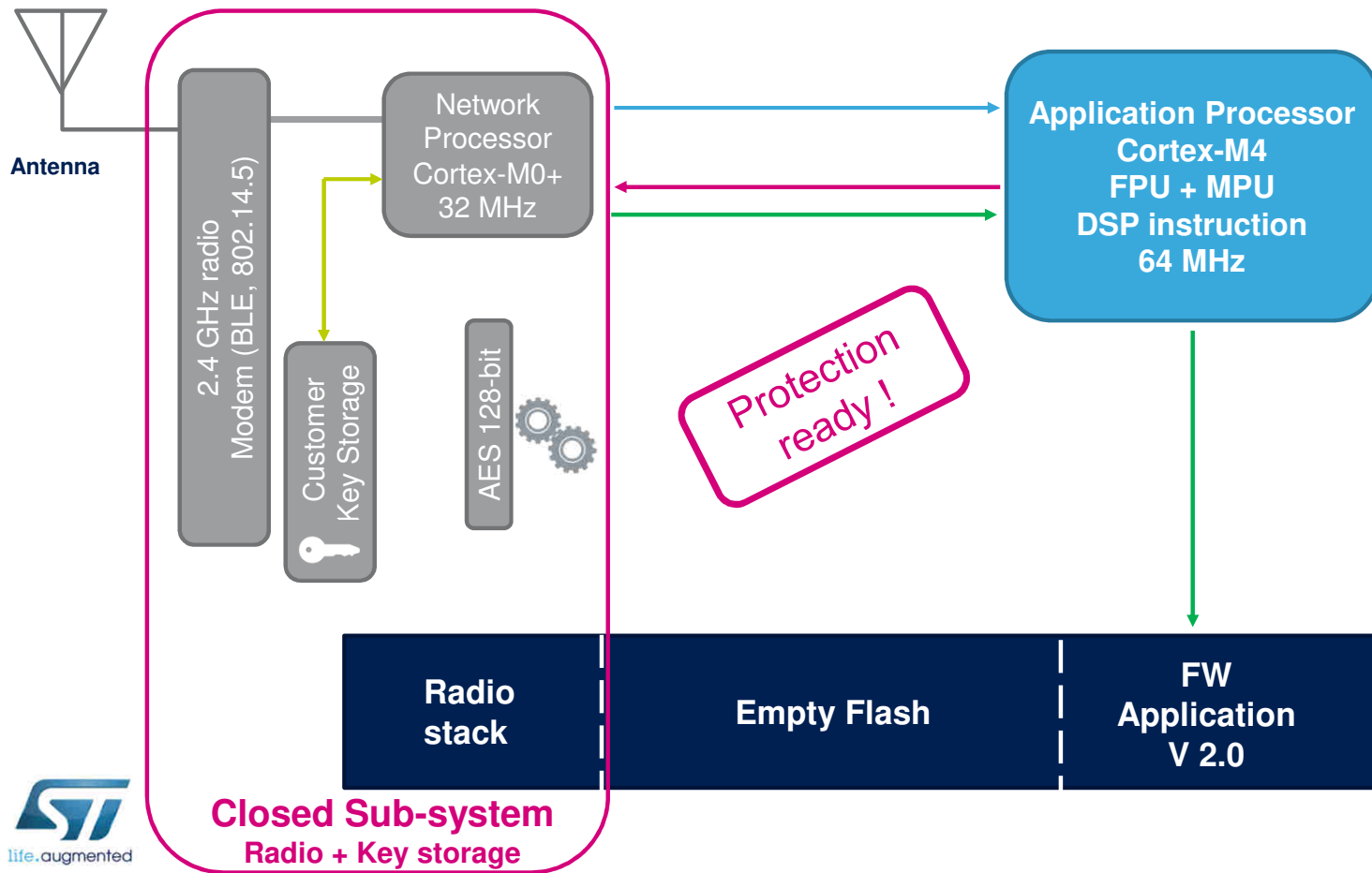
115



- 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.

OTA Update Scenario

116



- 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

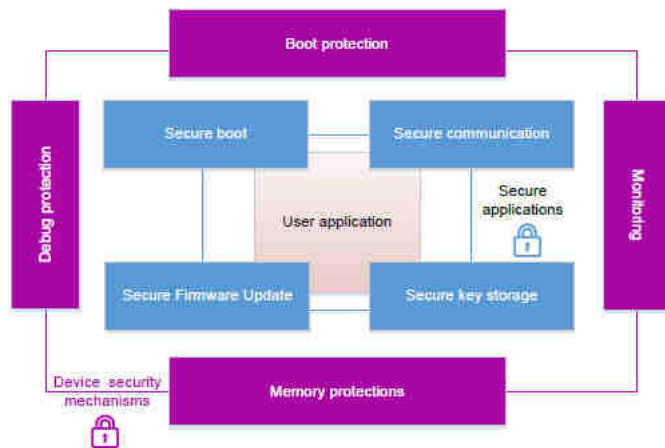


Figure 12. Dual-core architecture with CKS service

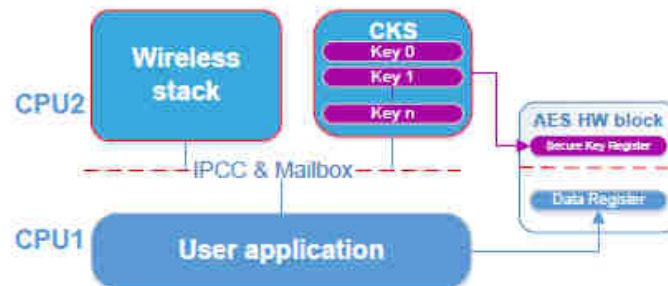
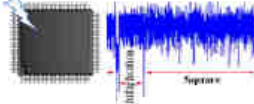
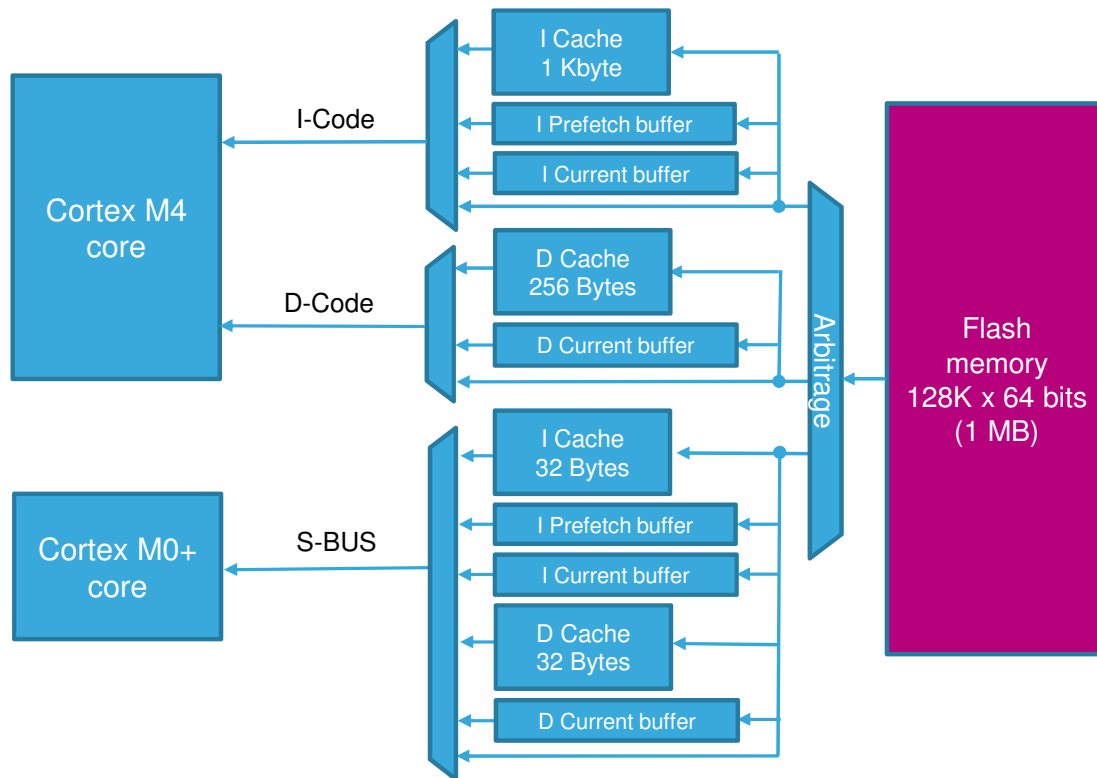


Table 4. Attacks types and costs

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)



- **Cortex-M4**

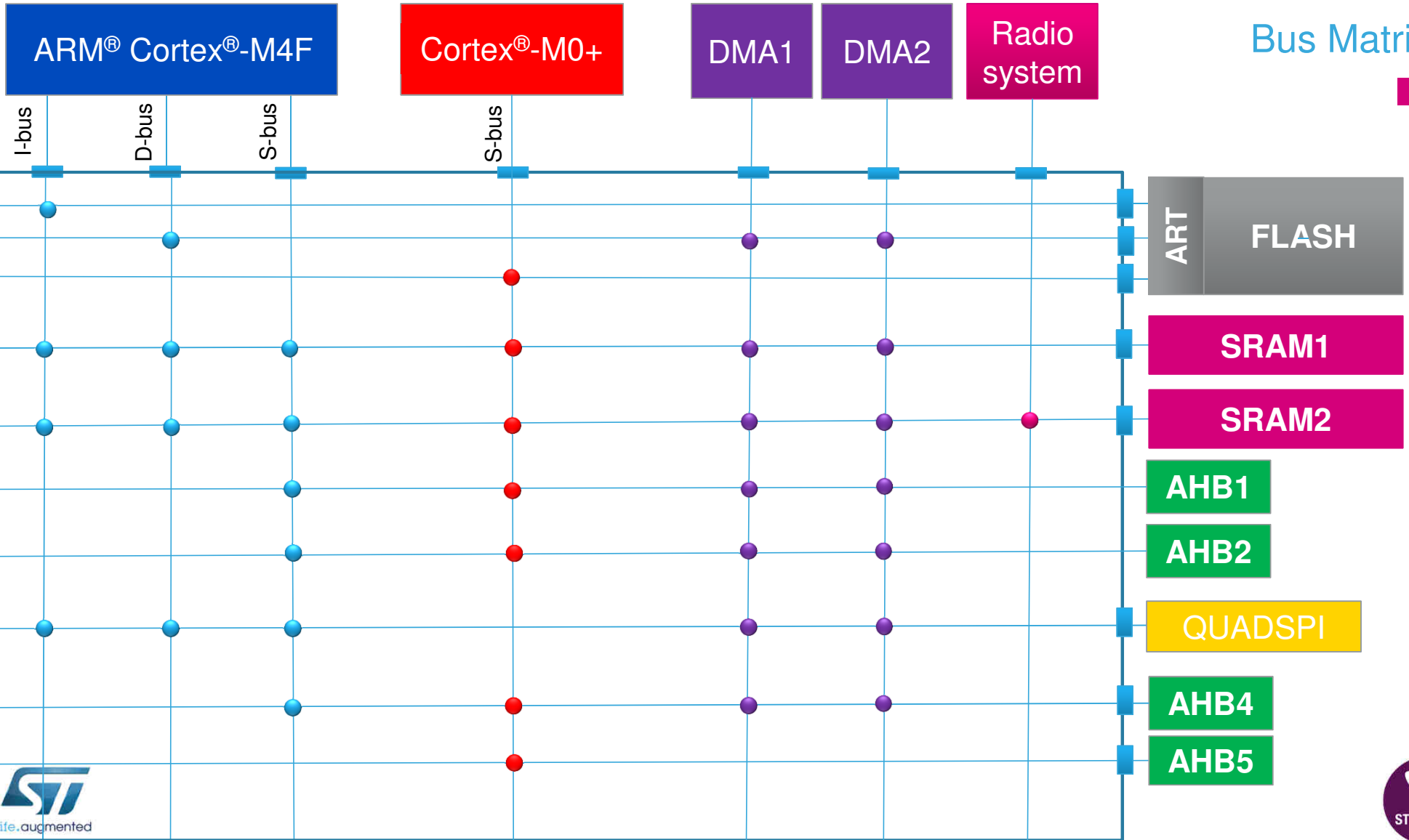
- **Instruction cache** = 32 lines of 4x64 bits
- **Data cache** = 8 lines of 4x64 bits
- **Pre-fetch buffer**

- **Cortex-M0+**

- **Instruction cache** = 4 lines of 1x64 bits
- **Data cache** = 4 lines of 1x64 bits
- **Pre-fetch buffer**

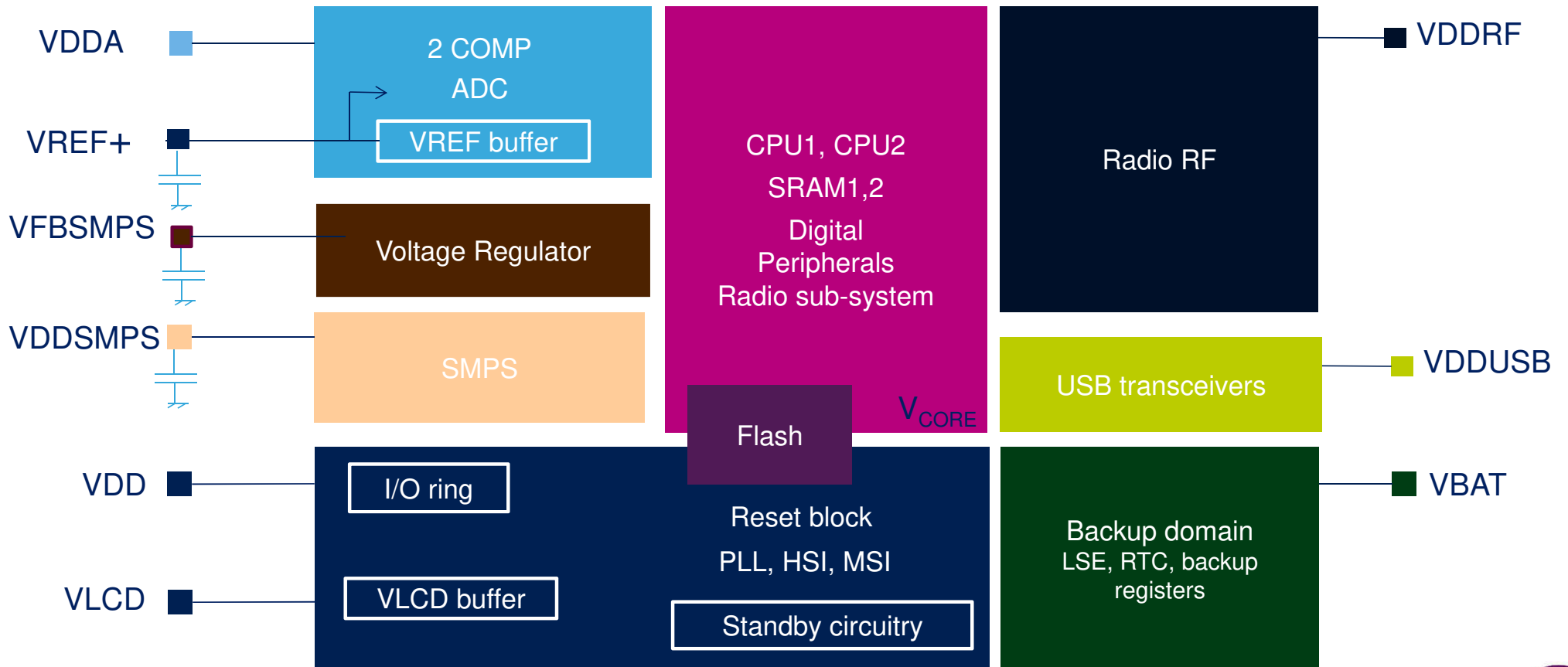
Bus Matrix

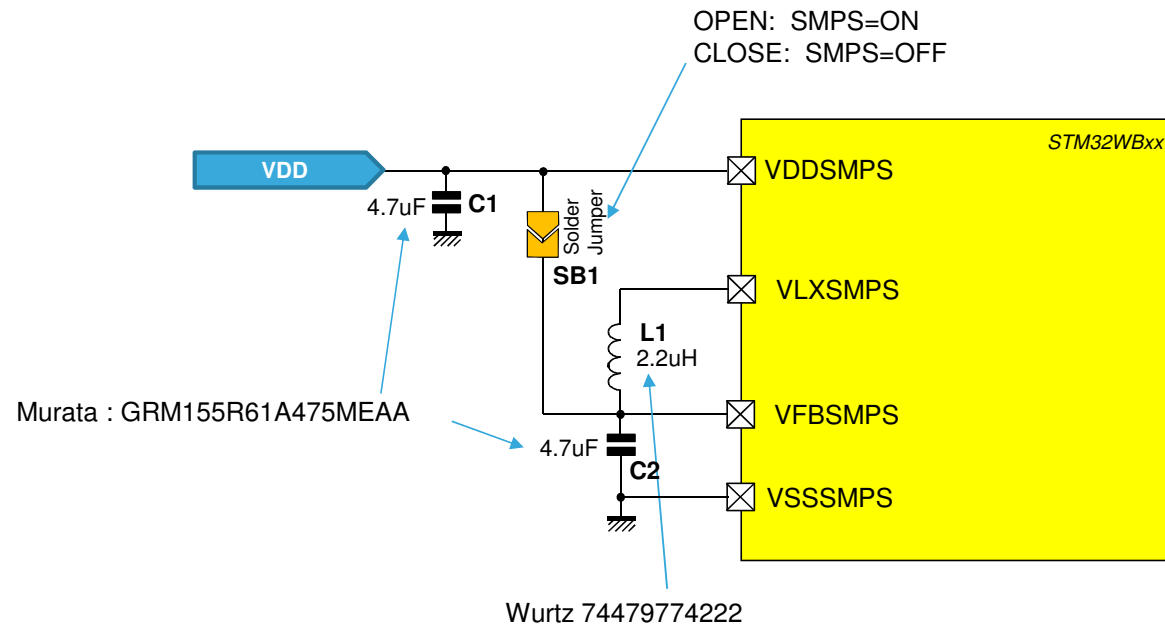
119



Power schemes

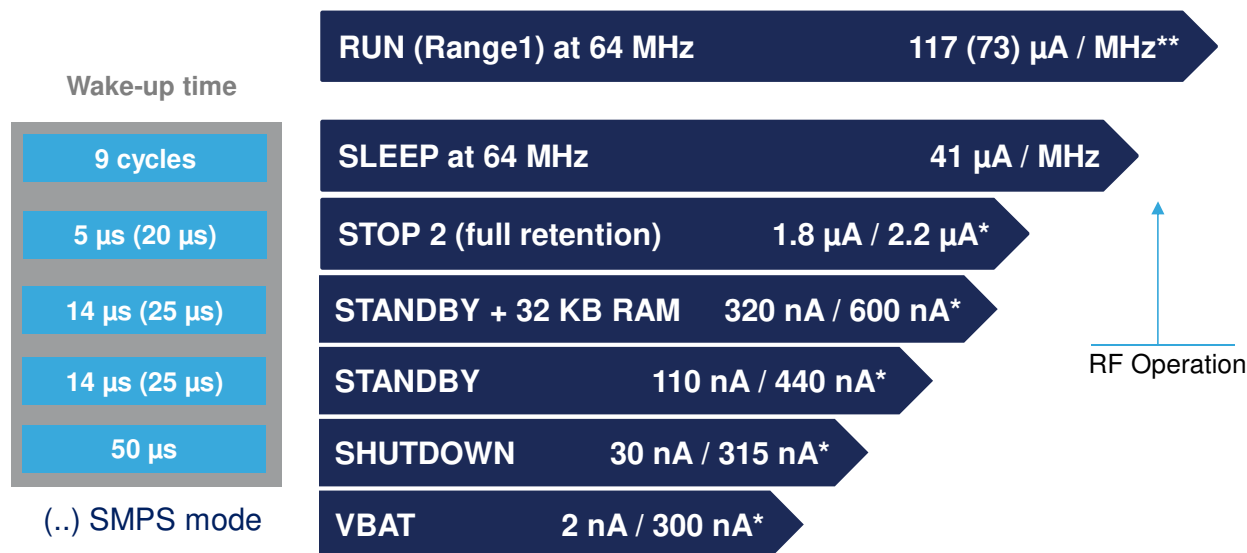
120





8MHz SMPS configuration

For 4MHZ SMPS configuration change $L1 = 4.7\mu H$



Typ @ VDD =1.8 V @ 25 °C

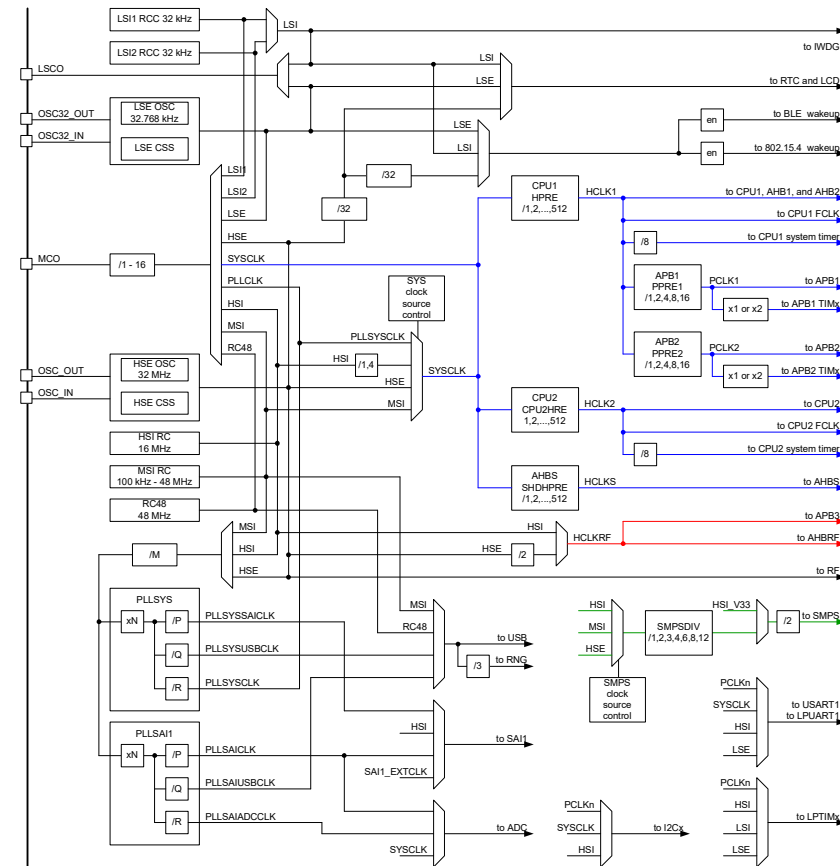
* with RTC

** from SRAM1

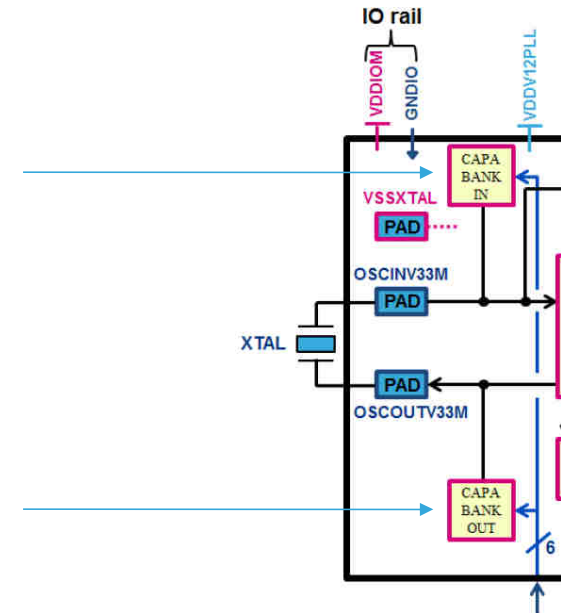
- High performance
 - CoreMark score = 215
- Outstanding power efficiency
 - ULPBbench score = 175

123

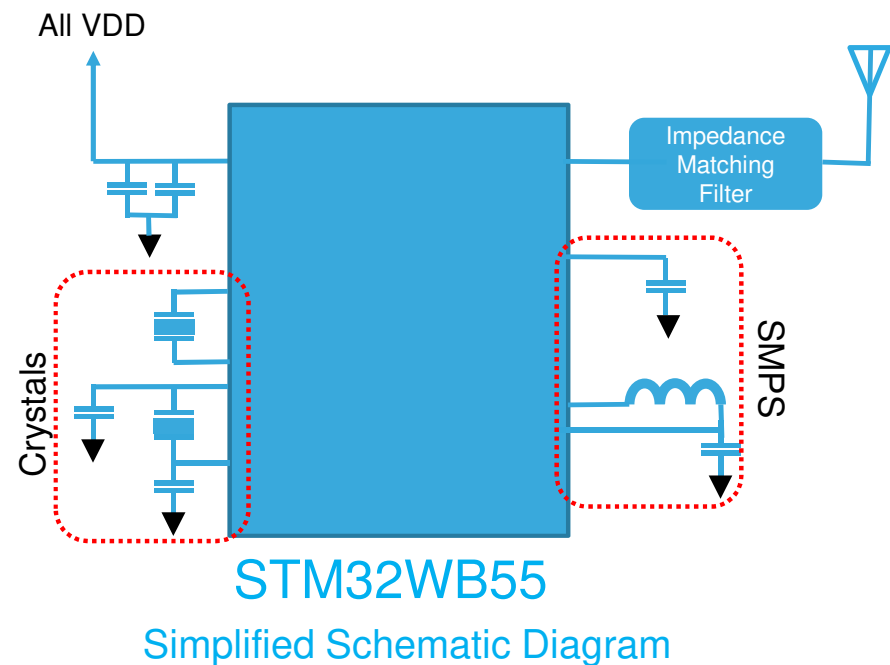
LSE (32.768KHz) required for most BLE applications



- 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



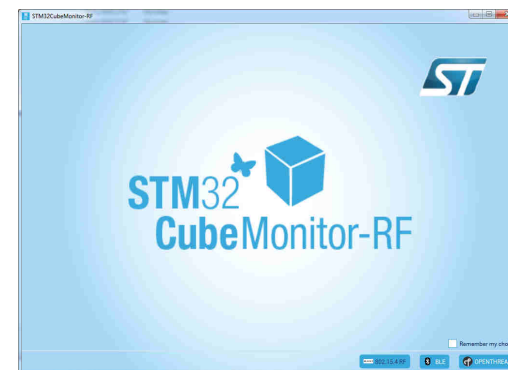
- Embedded RF balun
- Single IPD from ST
- Simple SMPS circuit
- Integrated HSE crystal tuning caps
- Minimal passives needed
- Simple 2 layer PCB design



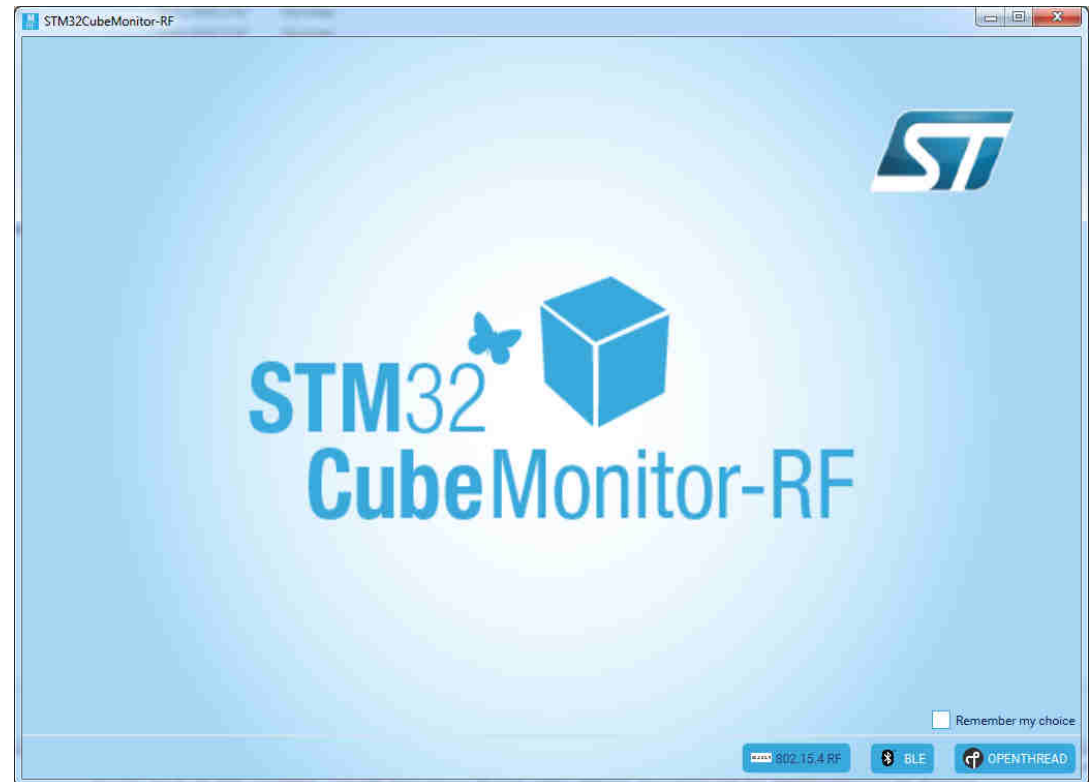


Hands-On

CubeMonitorRF



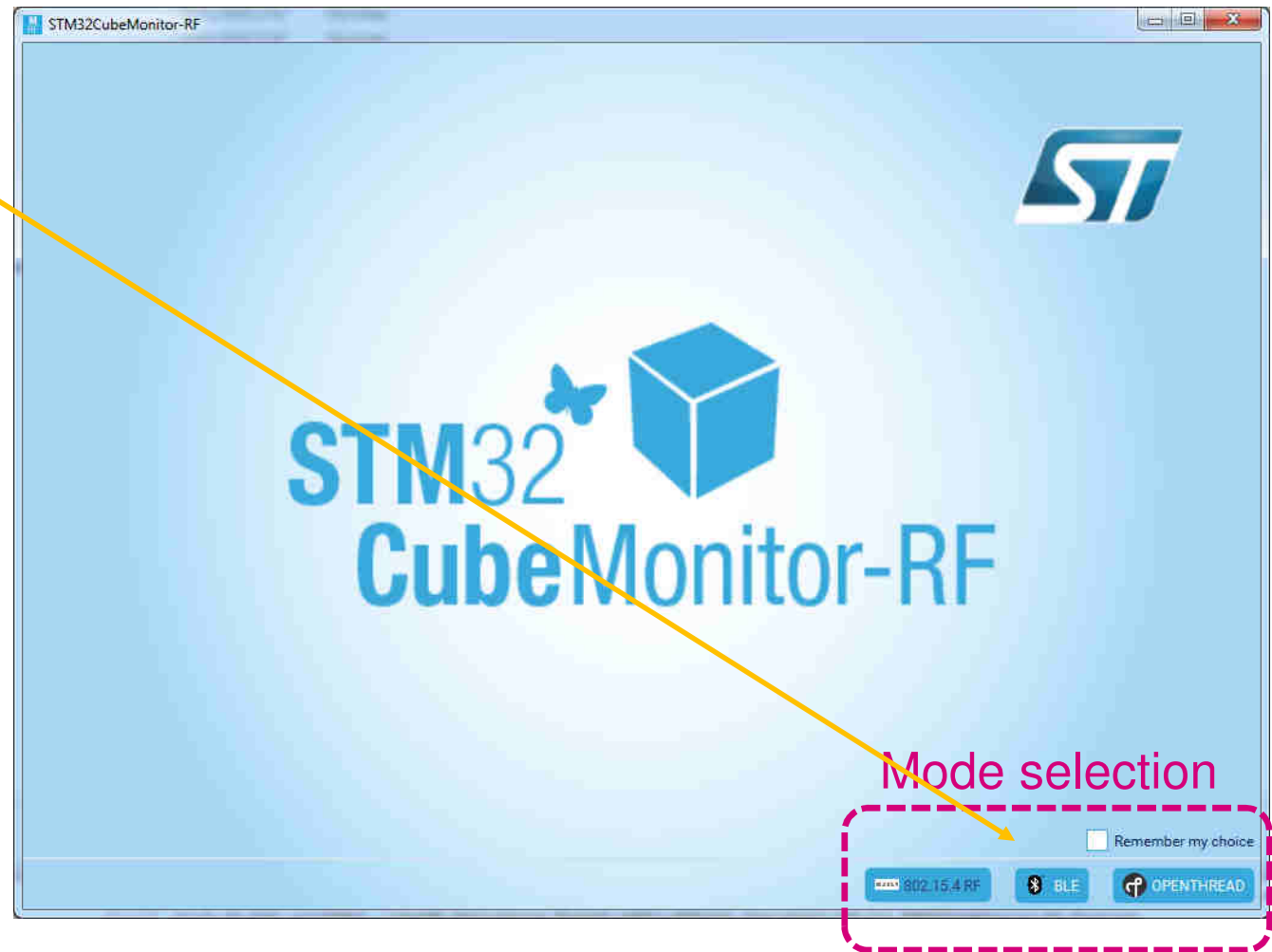
- BLE commands
- OpenThread commands
- BLE & 802.15.4 RF tests
- COM-port based



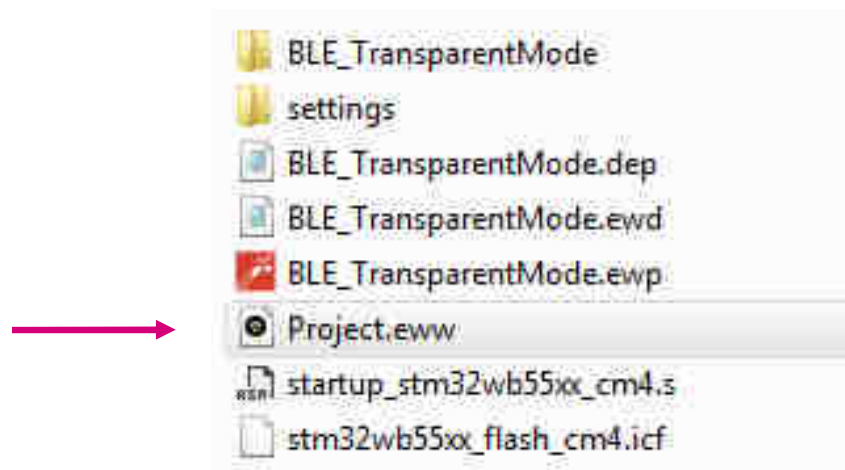
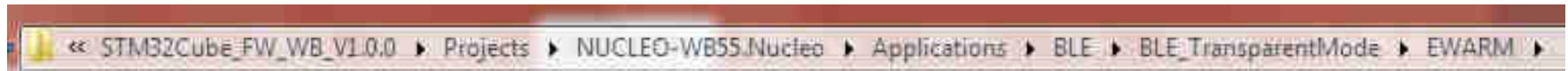
Mode selection

128

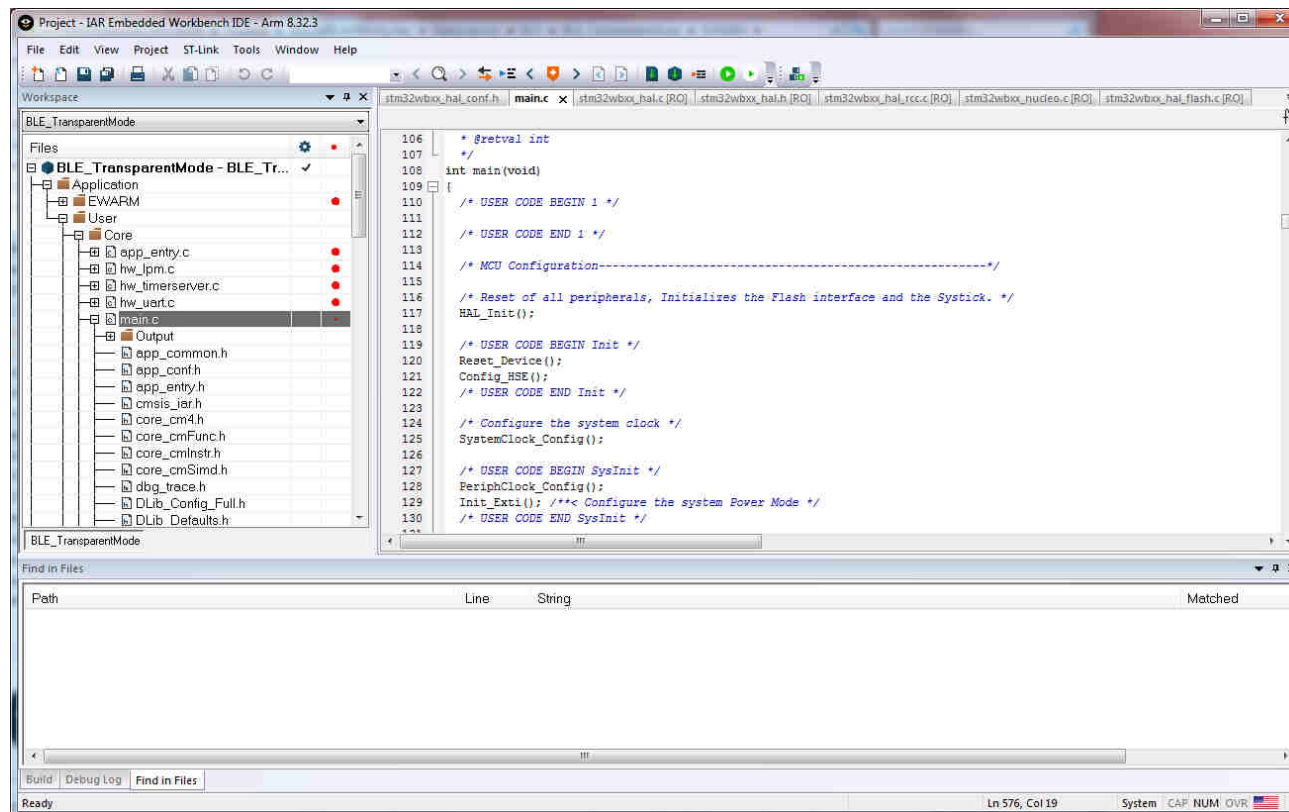
We will run in BLE mode



Open the **Transparent Mode** workspace



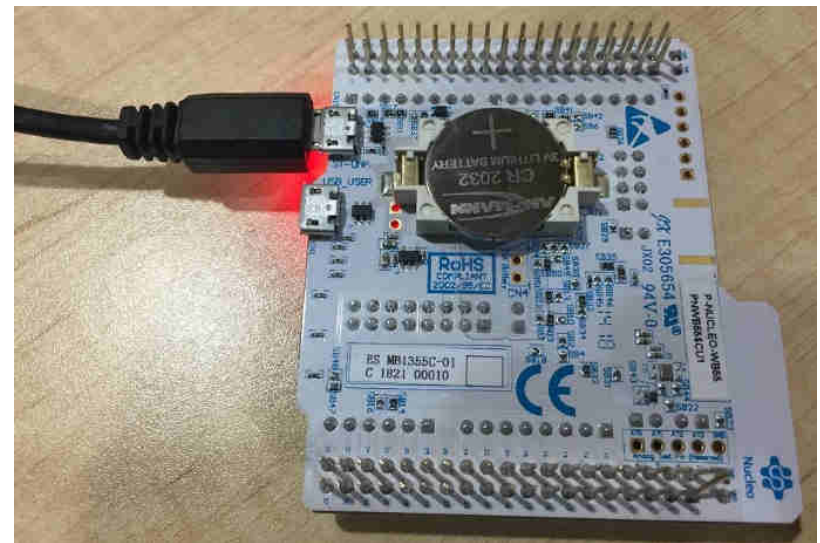
Build, Debug & Run on your Nucleo board



Connect via USB (virtual COM) port

Select device on relevant COM port

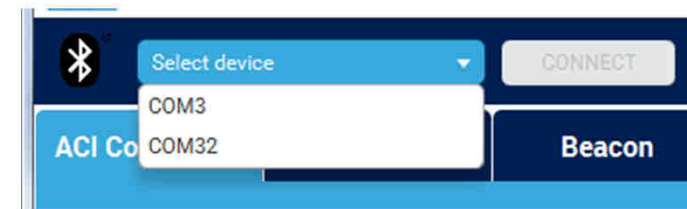
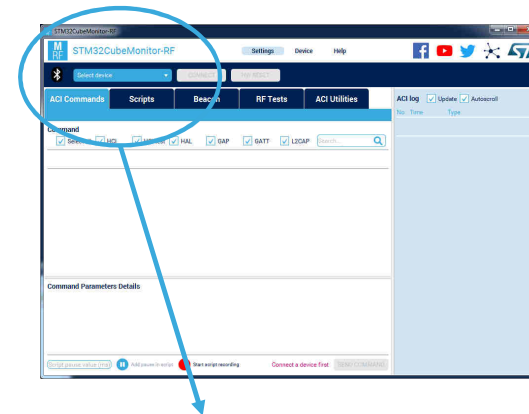
Connect to start communication



Connect via USB (virtual COM) port

Select device on relevant COM port

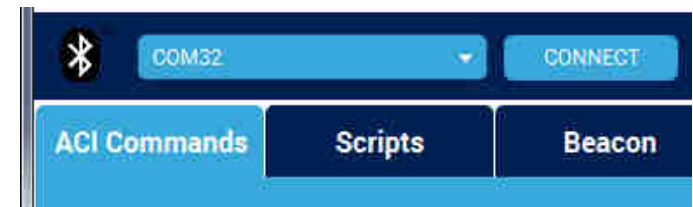
Connect to start communication



Connect via USB (virtual COM) port

Select device on relevant COM port

Connect to start communication



Command Complete signals successful communications

ACI log			☒ Update	☒ Autoscroll	RESET LOG
No	Time	Type			
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			

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

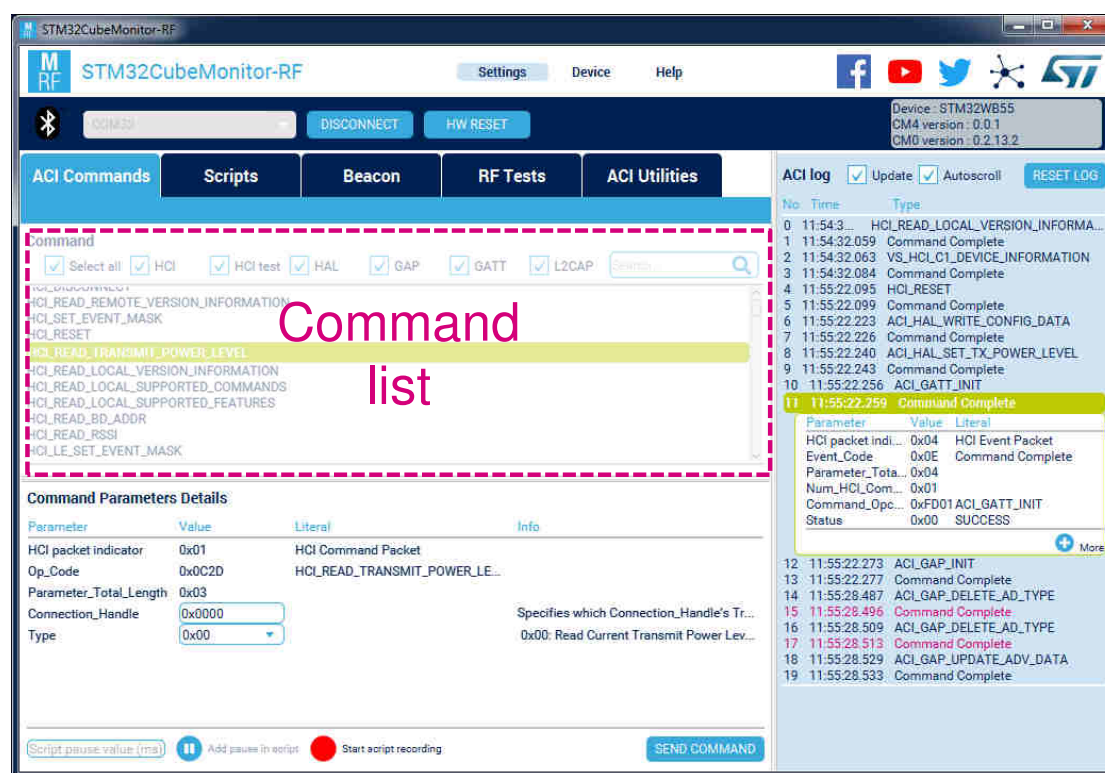
Parameter	Value	Literal
HCI packet indicator	0x21	HCI M4 Event Packet
Event_Code	0x0E	Command Complete
Parameter_Total_Length	0x42	
Num_HCI_Command_Packets	0x01	
Command_Opcode	0xFD62	VS_HCI_C1_DEVICE_INFORMATION
Status	0x00	SUCCESS
Device Revision	0x2000	
Device Code Id	0x0495	
Device Package	0x13	
Device Type	0x25	
Device Company	0x000080E1	
UID64	0x000007A5	
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

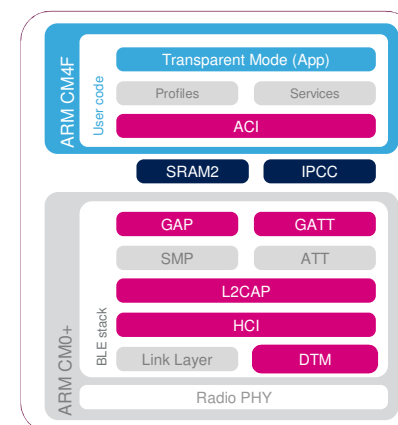
Parameter	Value	Literal	Info
HCI packet indicator	0x21	HCI M4 Event Packet	
Event_Code	0x0E	Command Complete	
Parameter_Total_Length	0x42		
Num_HCI_Command_Packets	0x01		The number of HCI command packets which are allowed to
Command_Opcode	0xFD62	VS_HCI_C1_DEVICE_INFORMATION	Opcode of this command which 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	0x000007A5		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



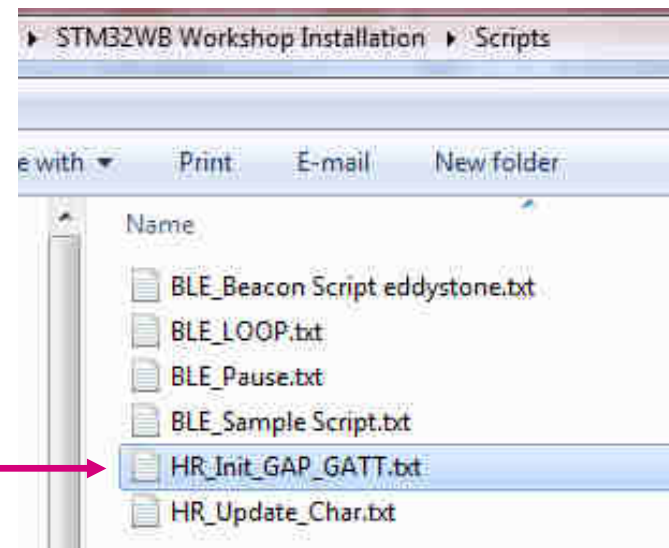
Command categories:

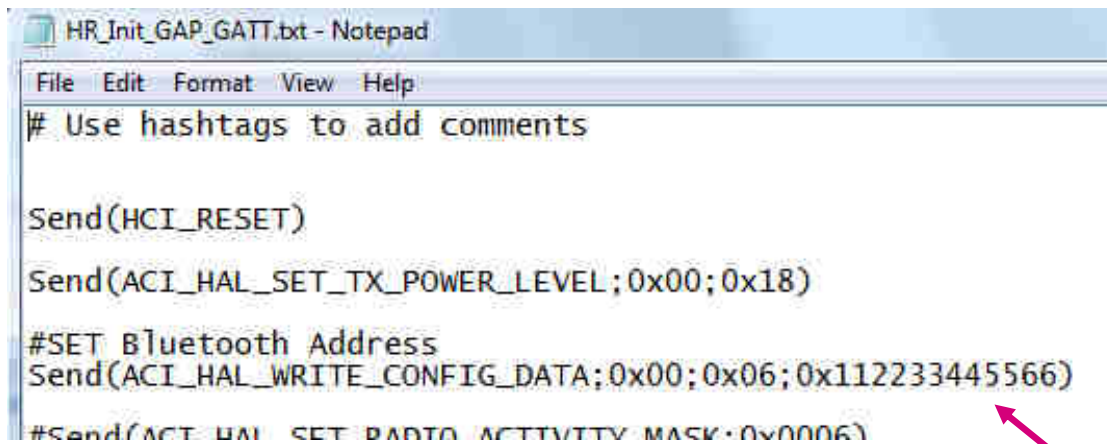
- HCI
- HCI test
- HAL
- GAP
- GATT
- L2CAP



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*

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

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

** ASCII Character Set for Magic numbers*

Save and Load your script

ACI Commands
Scripts
Beacon
RF Tests
Advertising

Script

☒ Generate report

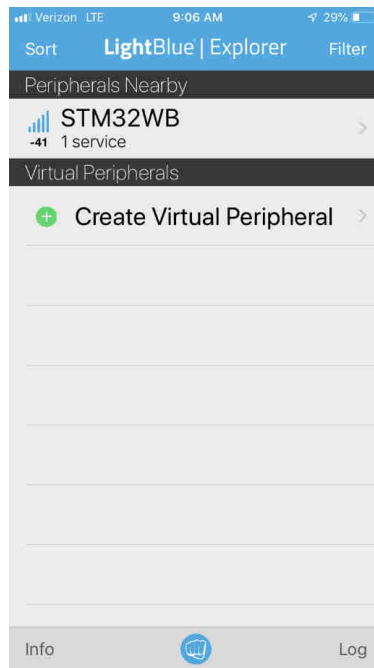
tions\WB Seminars Q2 2019\STM32WB Workshop Installation\Scripts\HR_init_GAP_GATT.txt

BROWSE

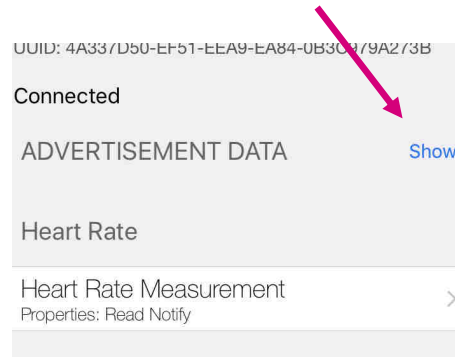
START SCRIPT

Start Script

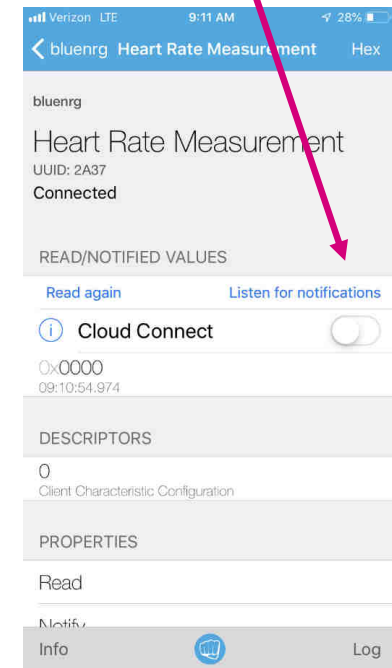
Find your device



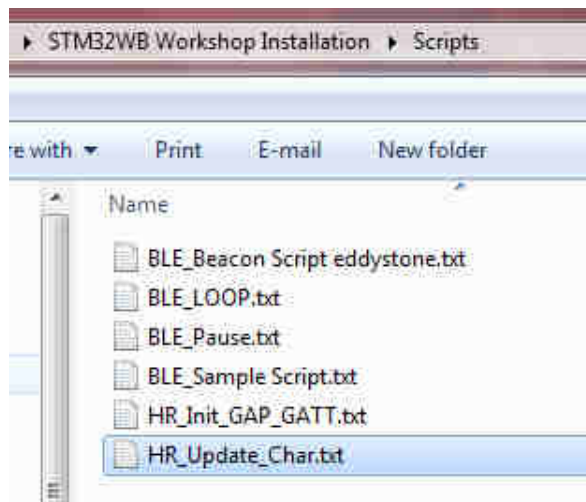
Show ADV data



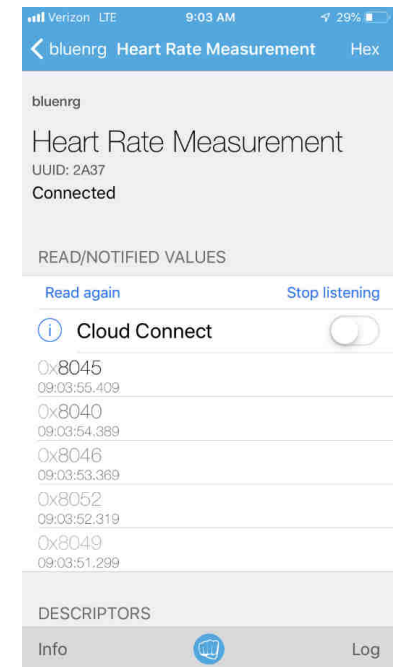
Enable Notifications



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

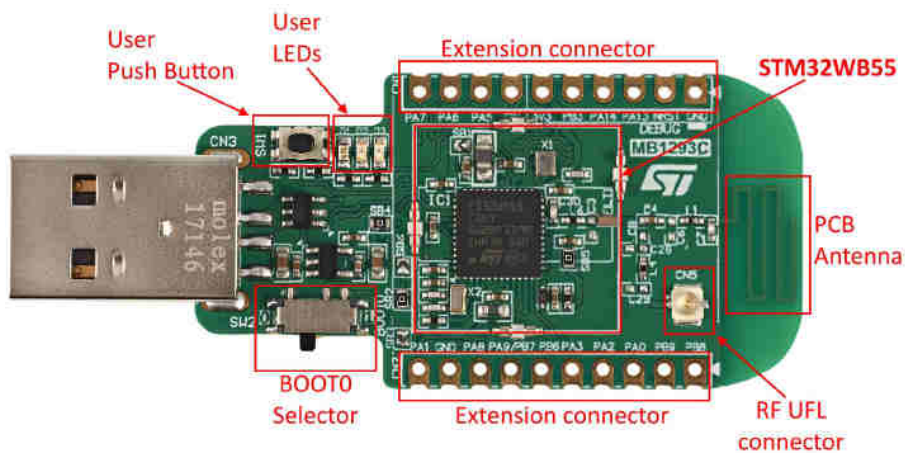
Table 108. Input parameters

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	-

WB55 Dongle Board

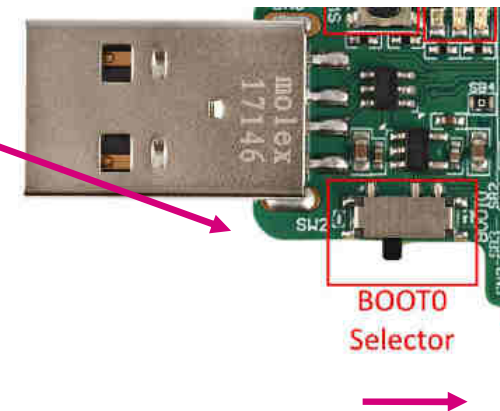
145

- The USB Dongle is quite useful as the CubeMonitorRF sniffer
- Uses the USB CDC class directly to parse commands
- USB bootloader invoked via **BOOT0 switch** & CubeProgrammer, and the binary can be programmed



❑ 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



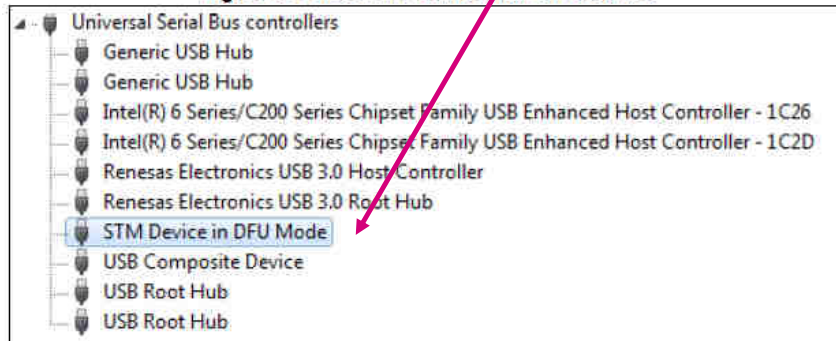
Chapter 1.2.4 details the DFU driver install / update procedure

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

Figure 2. Deleting the old driver software

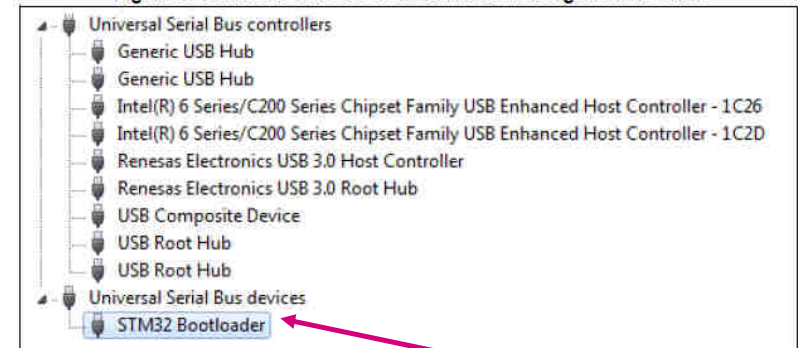


Figure 3. STM32 DFU device with DfuSe driver

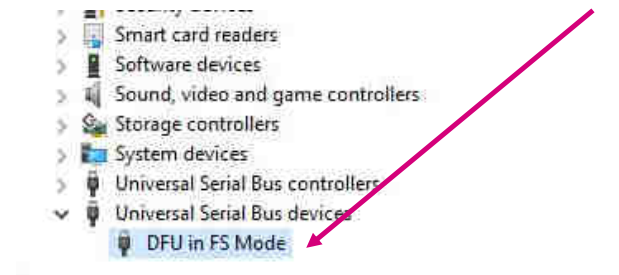


Old Driver

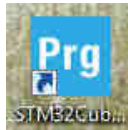
Figure 4. STM32 DFU device with STM32CubeProgrammer driver



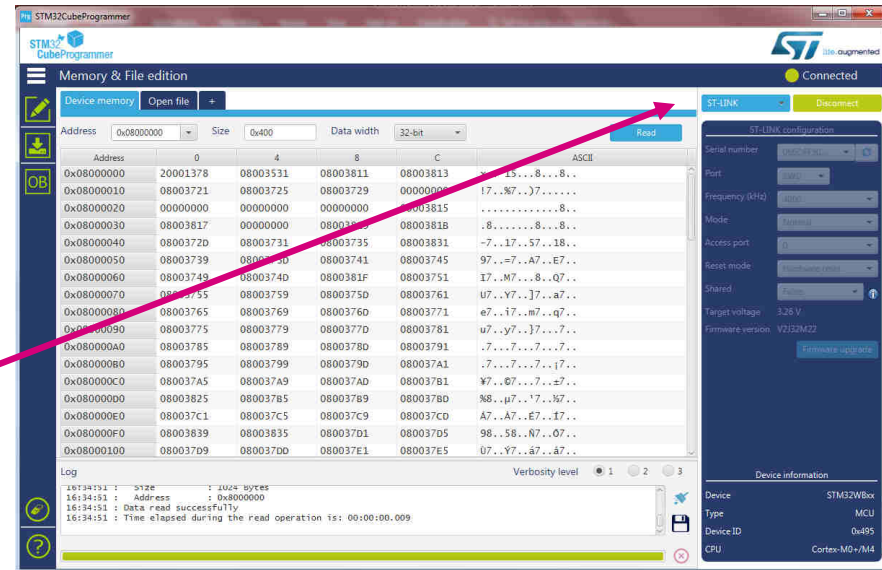
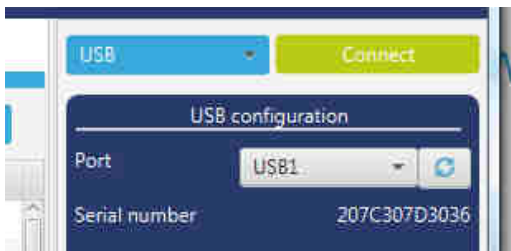
New Driver!



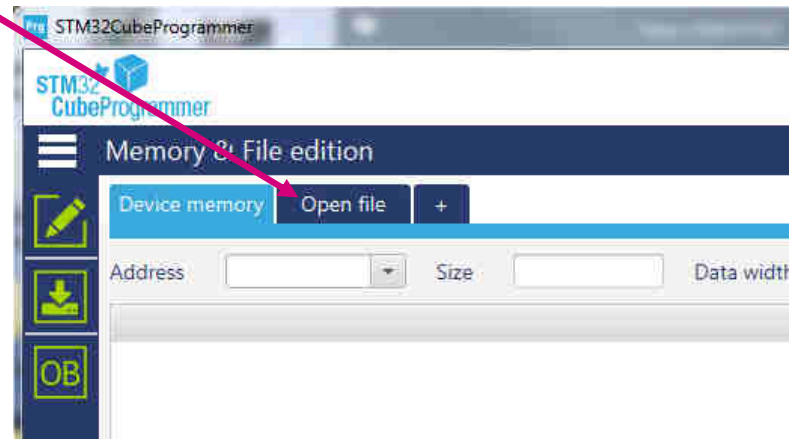
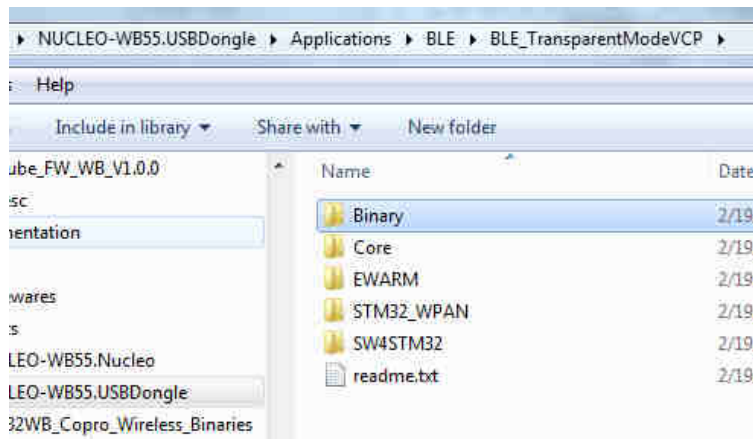
- Open STM32 CubeProgrammer



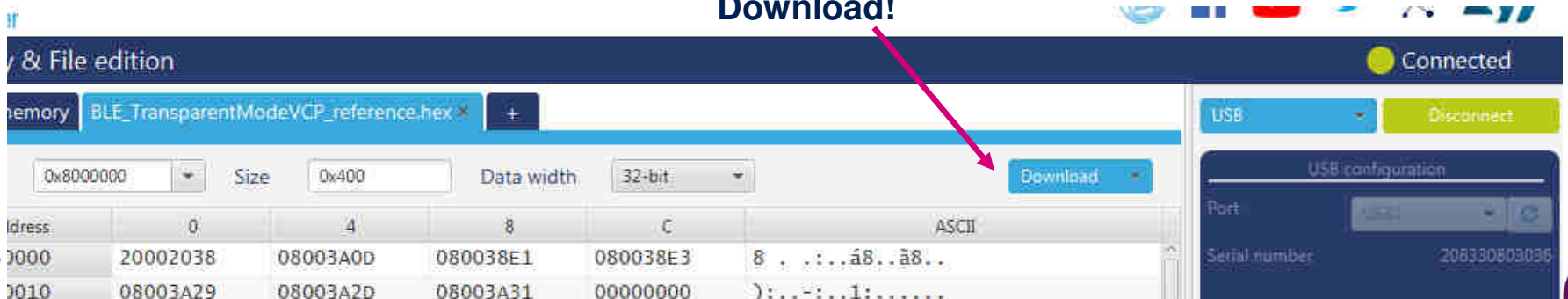
- Select **USB** mode and **Connect**



Open the **BLE_TransparentModeVCP_reference.hex** file for Dongle



Download!



Dongle config

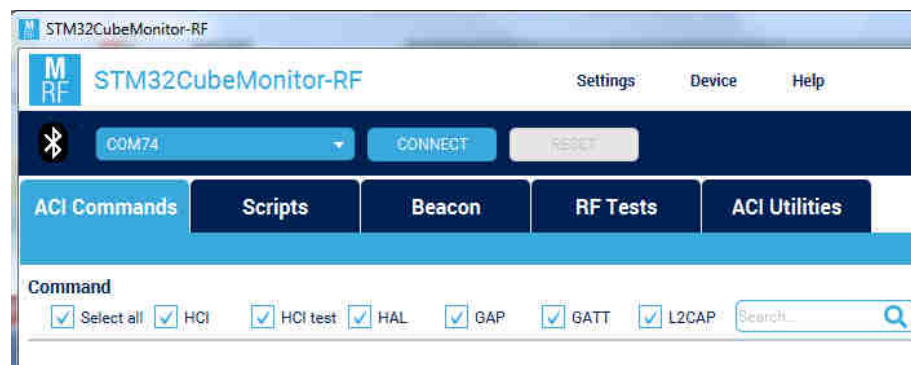
150

- ❑ 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)



Normal Boot to the left

❑ CONNECT



Change the Bluetooth Address and Name

Use Connectable advertising on all channels (37/38/39)

ACI Commands

Scripts

Beacon

RF Tests

ACI Utilities

Init.

Initialization parameters

☐ Discover remote services

☒ Advertising

Address

0x11223344ABCD

Power

29 (+4dBm)

Name

STM32WBWBWB

Discoverability mode

General discoverable

Adv type

0x00 - ADV_IND (Connectable undirected advertising)

Advertising channel map

☒ CH37
 ☒ CH38
 ☒ CH39

Own address type

0x00 - Public Device Address

Advertising interval (20 to 10240 ms)

1280 Min 1280 Max

Slave connection interval (7.5 to 4000 ms)

Min Max Use empty value for non specific Min/Max

START ADVERTISING

ACI log

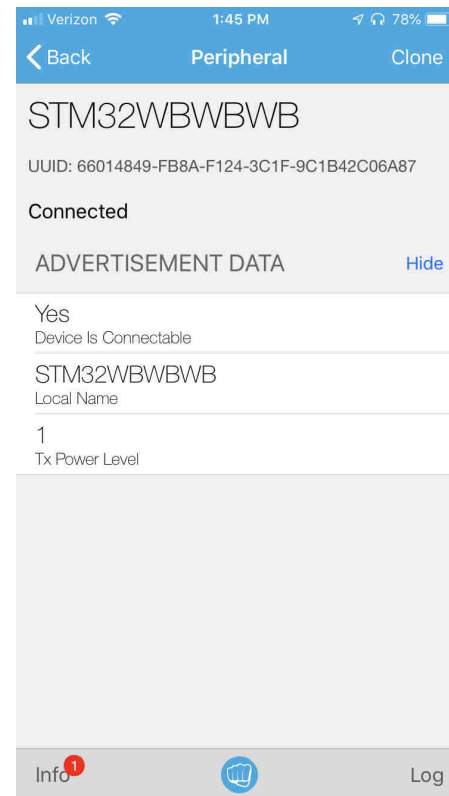
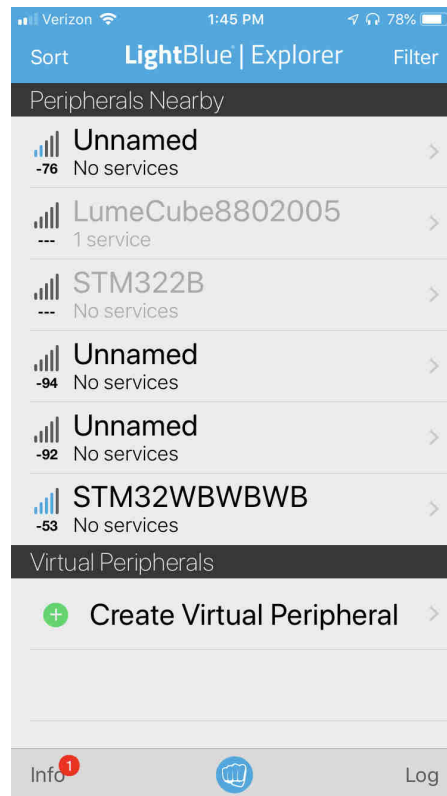
☒ Update ☒ Autoscroll

RESET LOG

No	Time	Type
...	13:44...	HCI_LE_READ_ADVERTISING_CHANNEL_TX...
47	13:44:47.920	Command Complete
...	13:44:47...	HCI_LE_SET_ADVERTISING_PARAMETE...
49	13:44:47.930	Command Complete
50	13:44:47.948	ACL_GAP_SET_DISCOVERABLE
51	13:44:47.950	Command Complete
52	13:44:47.958	ACL_GAP_UPDATE_ADV_DATA
53	13:44:47.962	Command Complete
54	13:45:25.230	LE Meta Event
55	13:45:25.659	LE Meta Event
56	13:45:25.776	LE Meta Event
57	13:45:25.778	Vendor Specific Event
58	13:45:26.140	Vendor Specific Event
59	13:47:45.315	Disconnection Complete
60	13:47:49.103	HCI_RESET
61	13:47:49.110	Command Complete
62	13:47:49.118	ACL_HAL_WRITE_CONFIG_DATA
63	13:47:49.125	Command Complete
64	13:47:49.135	ACL_HAL_SET_TX_POWER_LEVEL
65	13:47:49.140	Command Complete
66	13:47:49.146	ACL_GATT_INIT
67	13:47:49.150	Command Complete
68	13:47:49.157	ACL_GATT_INIT
69	13:47:49.161	Command Complete
70	13:47:49.169	ACL_GATT_UPDATE_CHAR_VALUE
71	13:47:49.175	Command Complete
...	13:47...	HCI_LE_READ_ADVERTISING_CHANNEL_TX...
73	13:47:49.190	Command Complete
...	13:47:49...	HCI_LE_SET_ADVERTISING_PARAMETE...
75	13:47:49.200	Command Complete
76	13:47:49.211	ACL_GAP_SET_DISCOVERABLE
77	13:47:49.213	Command Complete
78	13:47:49.221	ACL_GAP_UPDATE_ADV_DATA
79	13:47:49.223	Command Complete
80	13:47:55.644	LE Meta Event
81	13:47:56.079	LE Meta Event
82	13:47:56.202	LE Meta Event
83	13:47:56.210	Vendor Specific Event
84	13:47:56.560	Vendor Specific Event

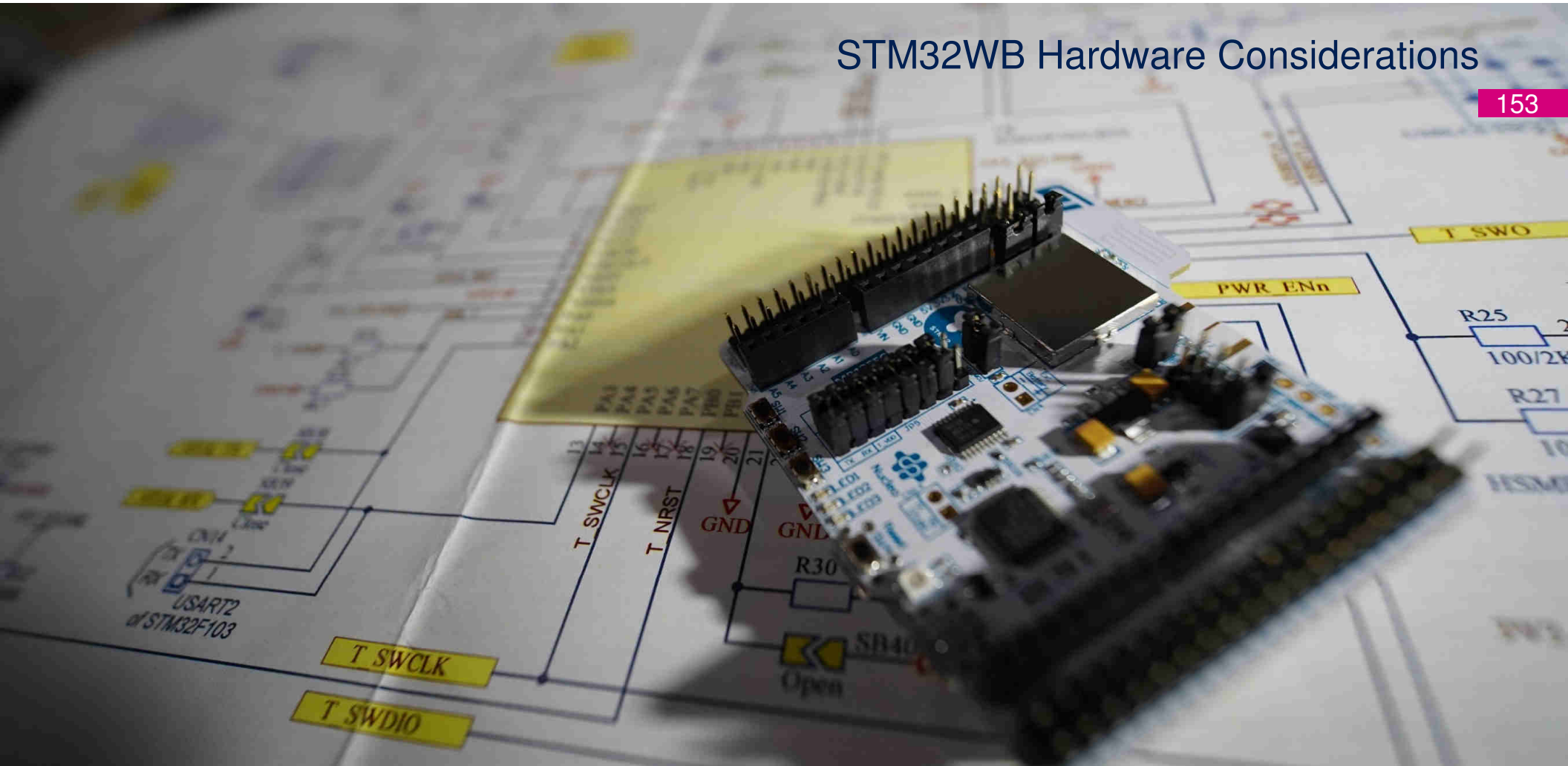
Start

Use LightBlue Explorer to connect to and interrogate your GAP peripheral

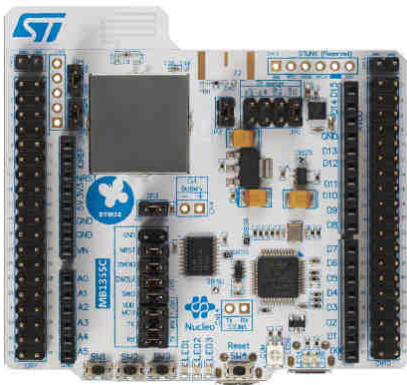


STM32WB Hardware Considerations

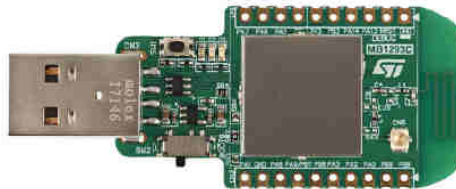
153



FUS + BLE Stack



FUS + BLE Stack



FUS only

Stack must be loaded

48-pin UQFN
(0.5 mm pitch)

68-pin VQFN
(0.4 mm pitch)



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 8. FUS HCI subset

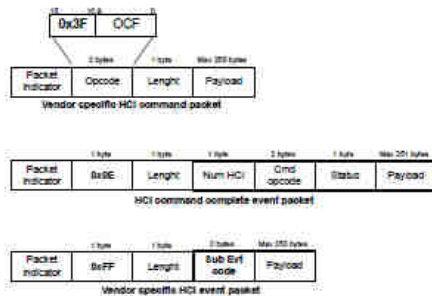
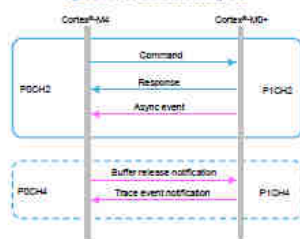
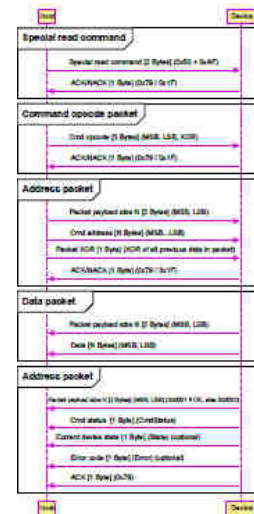


Figure 9. IPCC channels used by FUS



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 (AN2406)* and *USART protocol used in the STM32 bootloader (AN3155)* and *USB DFU protocol used in the STM32 bootloader (AN3156)* documentation is required.

AN51659 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

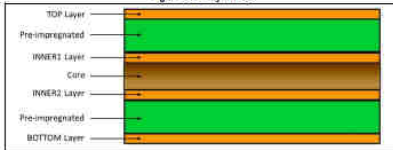


4-layer PCB

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, and for routing in others parts.
- INNER2 layer: power and low frequency routing.
- BOTTOM layer: low frequency routing.

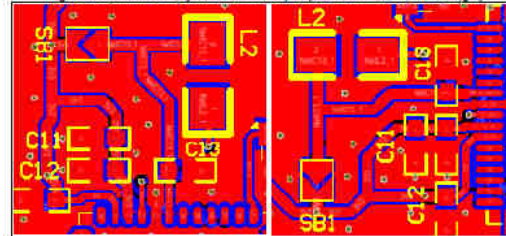
Figure 23. 4-layer PCB



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.

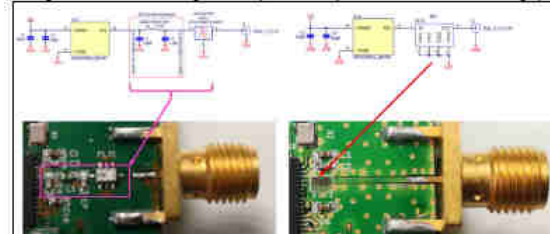
Figure 28. Detail of PCB layout for the SMPS part (UFQFPN48 left, VFQFPN68 right)



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)

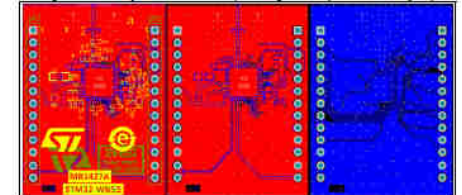


Figure 26. PCB layout for VFQFPN68 (left to right: all, top and bottom layers)

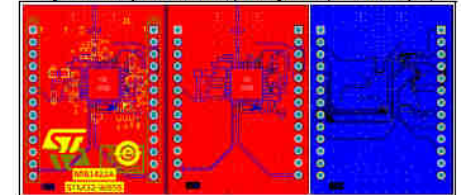
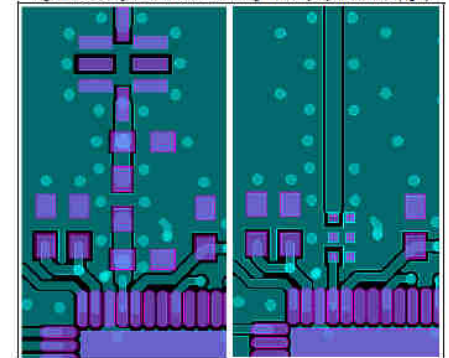


Figure 44. PCB layout with discrete matching network (left) and with IPD (right)



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

Figure 5. Optimized solution with IPD

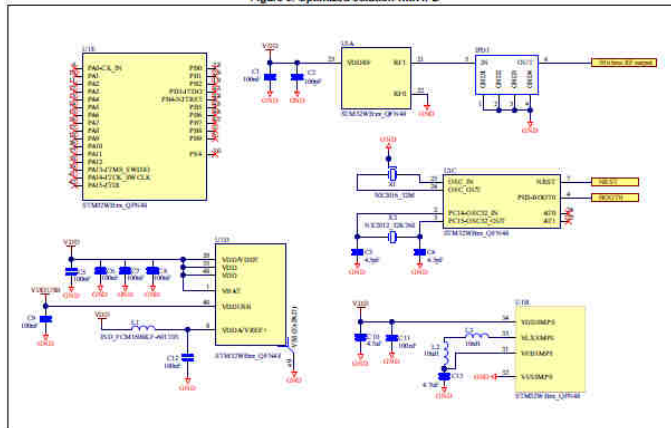


Figure 4. Optimized solution with discrete components

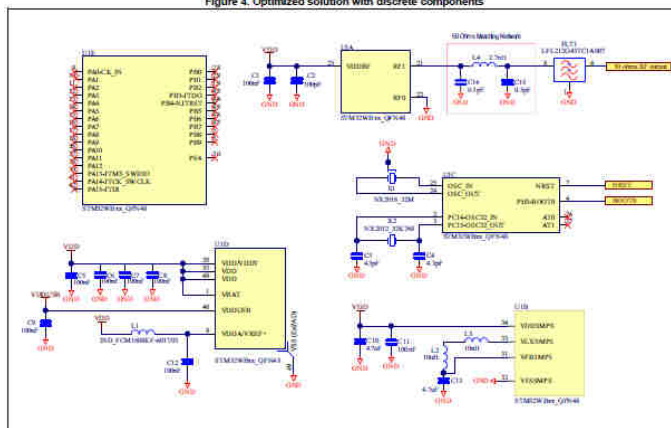


Table 2. Bill of materials - Optimized solution with IPD

Designator	Description	Comment	Footprint	Manufacturer	Part number
C1, C5, C6, C7, C8, C9, C11, C12	Capacitor, not polarized (XSR)	100 nF decoupling capacitors	0402	Murata	GRM155R61H104KE19D
C2	Capacitor, not polarized	100 pF decoupling capacitors		Yageo	CC0402KRX7R56B101
C3, C4	Capacitor, not polarized	4.3 pF LSE crystal capacitor		Murata	GRM1555C1H4R3CA01D
C10, C13	Capacitor, not polarized	4.7 μF decoupling capacitor			GRM155R61A475MEAD
L1	Coil	Filtering coil	0603	TAI-TECH	FCM1508/F-601T03
L2	Inductor	10 μH 50MP inductor	0805	Murata	LQM21FN100M70L
L3		10 nH 50MP inductor	0402		LOG15WZ10N02D
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	Burpress CGP	STMicroelectronics	MLPF-WB55-01E3

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 (XSR)	100 nF decoupling capacitors	0402	Murata	GRM155R61H104KE19
C2	Capacitor, not polarized	100 pF decoupling capacitors		Yageo	CC0402KRX7R56B10
C3, C4		4.3 pF LSE crystal capacitor		Murata	GRM1555C1H4R3CA01
C10, C13		4.7 μF decoupling capacitor			GRM155R61A475MEAA
C14		0.8 pF matching network			GRM1555C1HR080B0A1
C15		0.3 pF matching network			GRM1555C1HR030VAD1
L1	Coil	Filtering coil	0603	TAI-TECH	FCM1608/F-601T03
L2	Inductor	10 μH SMDP inductor	0805	Murata	LQM21FN100M70L
L3		10 nH SMDP inductor	0402		LOG15WZ10N02D
L4		2.7 nH matching network			LOG15HG2V7S02D
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	LFL212Q45TC1A007

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



AN5246 Application note

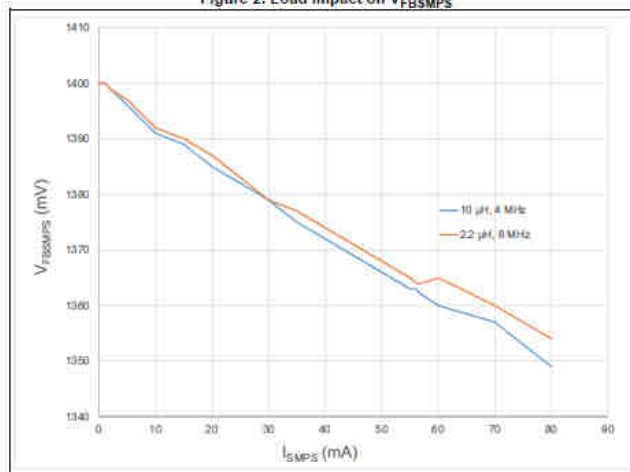
Usage of SMPS on STM32WB Series microcontrollers

Introduction

This document describes how to 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 $V_{FB\text{SMPS}}$



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 conduct and $V_{\text{FB SMPS}}$ follows $V_{\text{DD SMPS}}$. 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





STM32WB ultra-low-power features overview

```

graph TD
    Run((Run mode)) <--> Sleep((Sleep mode))
    Run <--> LowPowerRun((Low power run mode))
    Run <--> Shutdown((Shutdown mode))
    Run <--> Standby((Standby mode))
    Run <--> Stop0((Stop 0 mode))
    Run <--> Stop2((Stop 2 mode))
    LowPowerRun <--> LowPowerSleep((Low power sleep mode))
    LowPowerSleep --- Note([Only available when RF is not used])
  
```

Figure 10 is a line graph showing the relationship between Icc current (µA/MHz) and CPU1 frequency (Hz). The y-axis represents Icc current (µA/MHz) ranging from 0 to 500. The x-axis represents CPU1 frequency (Hz) on a logarithmic scale, ranging from 10,000 to 10,000,000. The graph displays six data series, each representing a different operational mode or range:

- FLASH LP-run (Blue line with circles): Shows a sharp decrease in Icc current as frequency increases, starting around 420 µA/MHz at 10,000 Hz and leveling off around 170 µA/MHz at 10,000,000 Hz.
- FLASH Run range 1 (Dark blue line with circles): Shows a sharp decrease in Icc current as frequency increases, starting around 310 µA/MHz at 10,000 Hz and leveling off around 130 µA/MHz at 10,000,000 Hz.
- FLASH Run range 2 (Grey line with circles): Shows a sharp decrease in Icc current as frequency increases, starting around 420 µA/MHz at 10,000 Hz and leveling off around 170 µA/MHz at 10,000,000 Hz.
- SRAM1 LP run (Orange line with circles): Shows a sharp decrease in Icc current as frequency increases, starting around 310 µA/MHz at 10,000 Hz and leveling off around 130 µA/MHz at 10,000,000 Hz.
- SRAM1 Run range 1 (Green line with circles): Shows a relatively flat Icc current around 150 µA/MHz across the frequency range from 10,000,000 Hz to 10,000,000,000 Hz.
- SRAM1 Run range 2 (Yellow line with circles): Shows a relatively flat Icc current around 120 µA/MHz across the frequency range from 10,000,000 Hz to 10,000,000,000 Hz.

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

AN5155: CubeWB Examples

AN5155 is an exhaustive list of all firmware examples and descriptions

161

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-WB56LUS BDongle	P-NUCLEO-WB56LUS 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

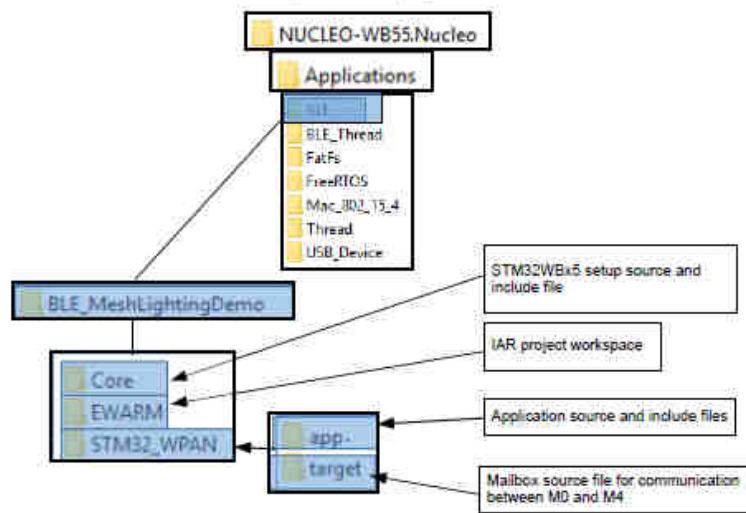
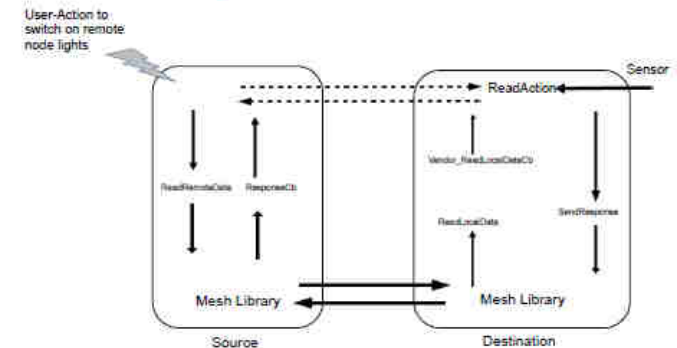
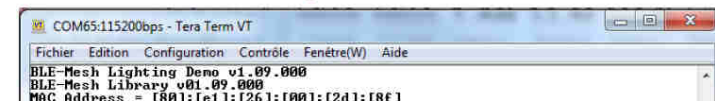


Figure 14. Read command from a remote node



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 requires a unique 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.

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

GitHub, Inc. [US] | https://github.com/STMicroelectronics/BlueSTSDK_Android

Why GitHub? Enterprise Explore Marketplace Pricing Search Sign in Sign up

STMicroelectronics / BlueSTSDK_Android Watch 25 Star 44 Fork 22

Code Issues Pull requests Projects Insights

Join GitHub today
 GitHub is home to over 36 million developers working together to host and review code, manage projects, and build software together.
 Sign up

Bluetooth low energy Sensors Technology Software Development Kit (Android version) <https://www.st.com/en/embedded-software...>

24 commits 2 branches 0 releases 1 contributor BSD-3-Clause

Branch: master New pull request Find File Clone or download

GiovanniVisentini Add the possibility to use a custom advertise format Latest commit 6b772cb on Mar 26

File	Commit Message	Time
BlueSTExample	Add the possibility to use a custom advertise format	a month ago
BlueSTSDK	Add the possibility to use a custom advertise format	a month ago
.gitignore	Update android libs	2 years ago
LICENSE	Initial commit	4 years ago
README.md	Add the possibility to use a custom advertise format	a month ago
build.gradle	Add the possibility to use a custom advertise format	a month ago
settings.gradle	Initial commit	4 years ago

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:

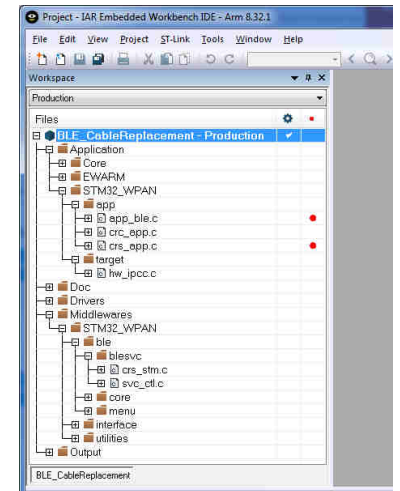
Length	1	1	1	1	4	6
Name	Length	Field Type	Protocol Version	Device Id	Feature Mask	Device MAC (optional)
Value	0x07/0xD	0xFF	0x01	0XX	0XXXXXXXX	0XXXXXXXXXXXX

- 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-TOB000V1 (BlueNB-IoT) board



Hands-On

*Custom GATT & Cable Replacement
(time permitting)*



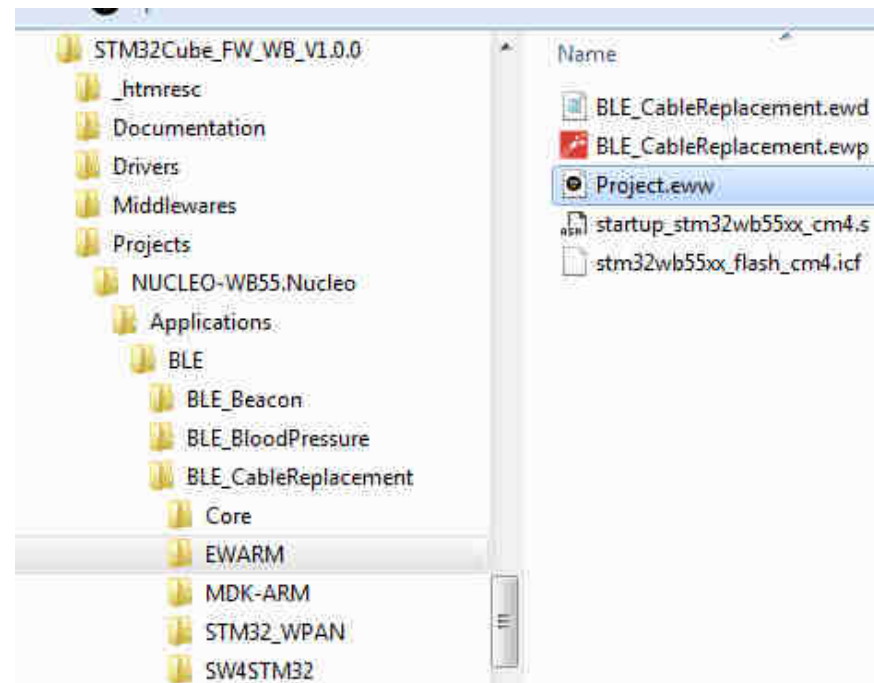
Open, Compile, Program & Run the Nucleo Board ***CableReplacement*** example

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*

- Open the workspace

STM32Cube_FW_WB_V1.0.0 > Projects > NUCLEO-WB55.Nucleo > Applications > BLE > BLE_CableReplacement > EWARM >

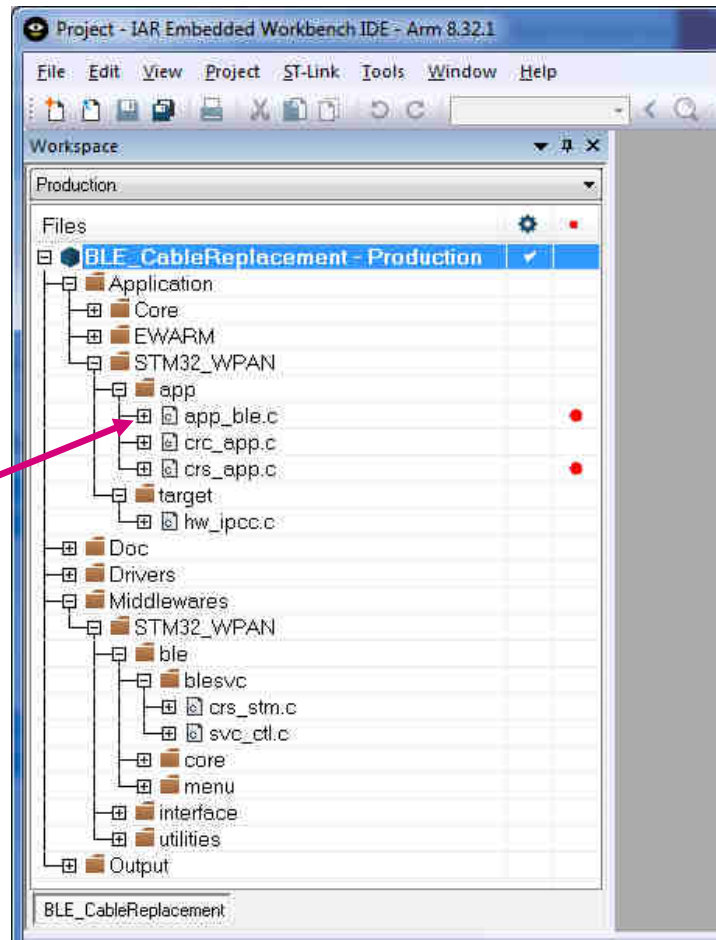


Build the Project

Open the following files

- app_conf.h
- app_ble.c – Under Application/STM32_WPAN/app
- 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
 - #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# 819 of app_ble.c)

```
const char *name = "BLE-CRS-01";
```

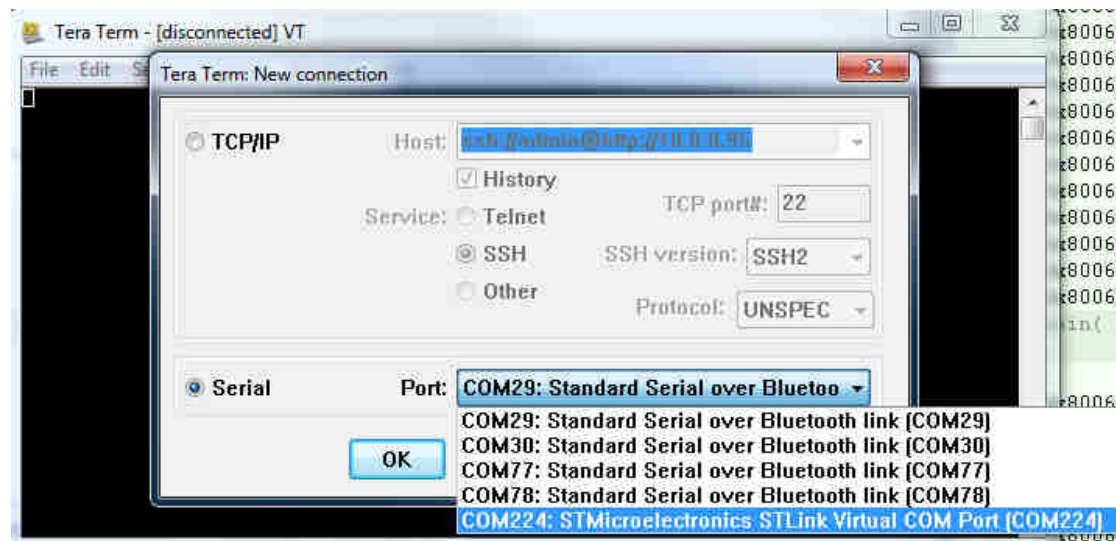
- Ensure that the BLE device name length in ASCII chars matches (line# 165 of app_ble.c)

```
#define APPBLE_GAP_DEVICE_NAME_LENGTH 10
```

Cable Replacement Test

170

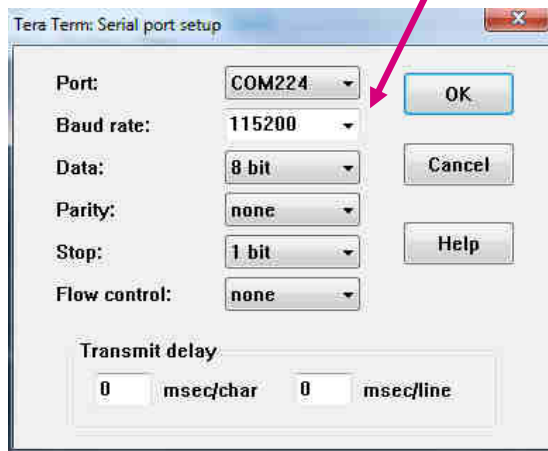
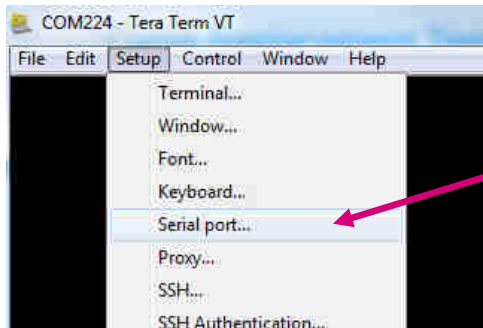
- Build and Run the project
- Connect your TeraTerm to the Nucleo's STLink Virtual COM port



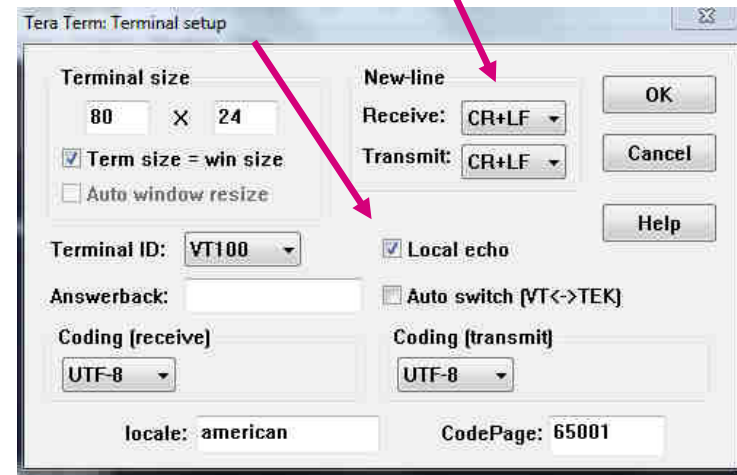
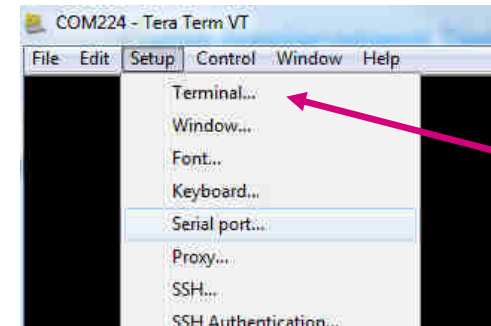
Cable Replacement Test

171

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



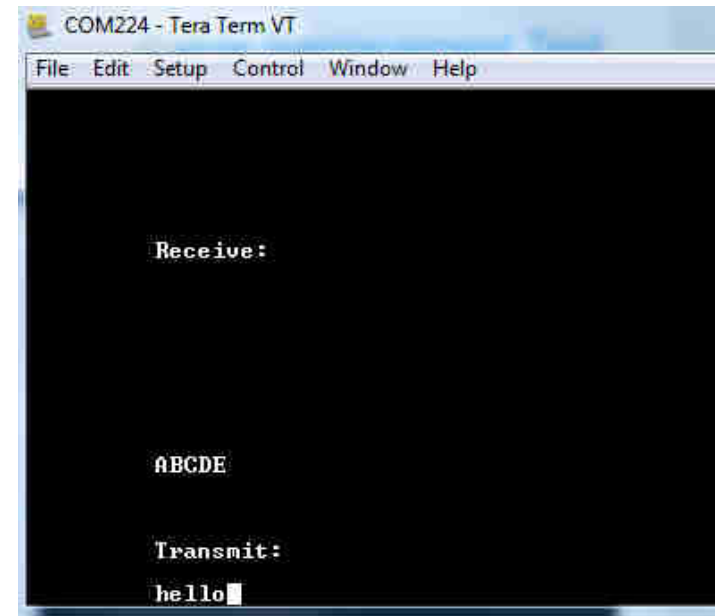
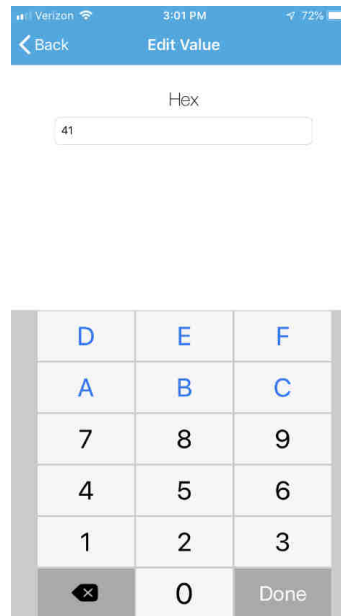
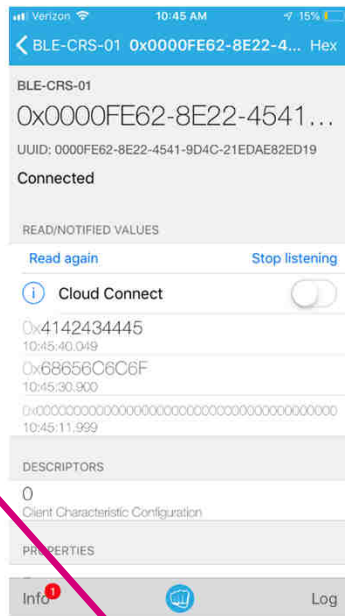
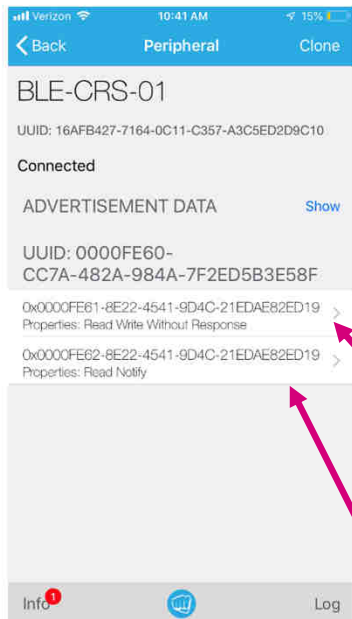
Configure Terminal Setup



Cable Replacement Test

172

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



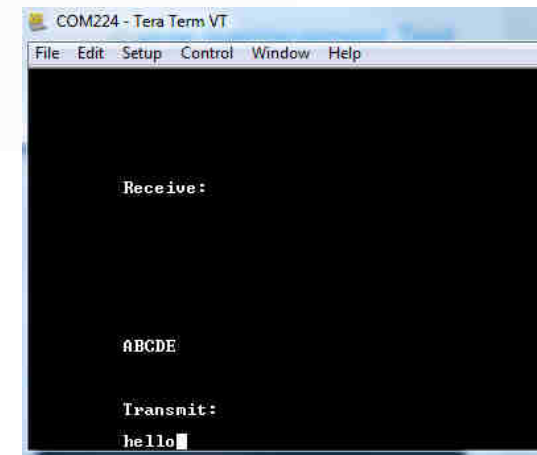
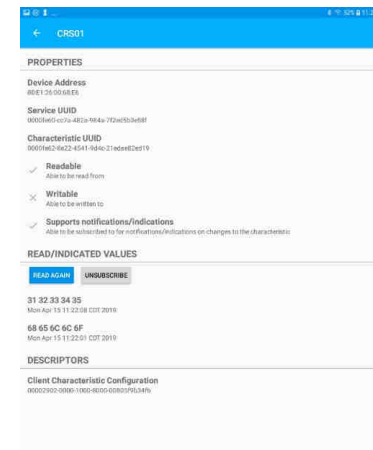
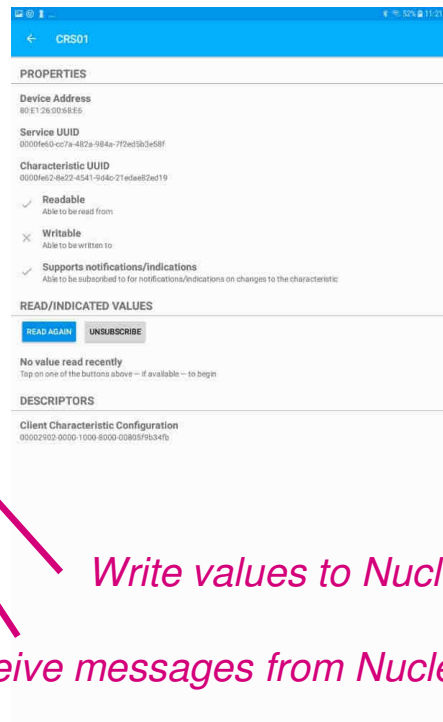
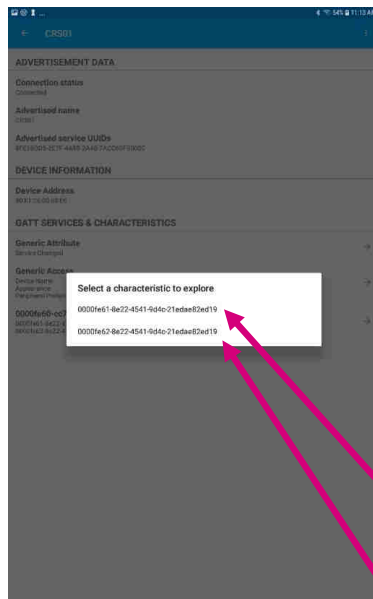
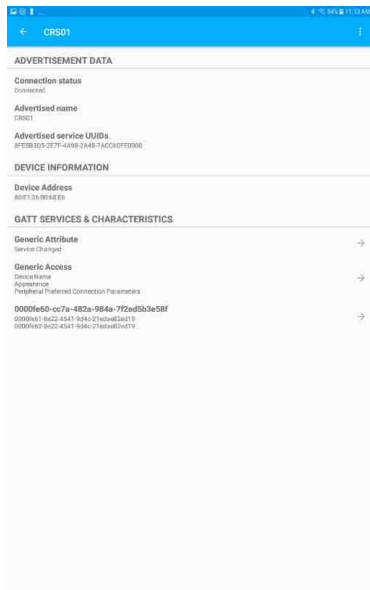
Write values to Nucleo

Enable Notifications to receive messages from Nucleo

Cable Replacement Test

173

- Here is the LightBlue Explorer on Android



Add a custom characteristic to an existing Service

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

```
#define STM_LED_UUIDI128    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 these files may be
read-only and may need
permissions changed

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

```
typedef struct {  
    ...  
    uint16_t CRSRXCharHdle;  
    uint16_t LedWriteClientToServerCharHdle;  
} CRSContext_t;
```

- 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.DataTransferred.Length = attribute_modified->Attr_Data_Length;  
        Notification.DataTransferred.pPayload = attribute_modified->Attr_Data;  
        CRSAPP_Notification(&Notification);  
    }  
}
```

Note that these files may be read-only and may need permissions changed

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

```
uint8_t led_uuid[] = { STM_LED_UUIDI128 };
```

- 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)

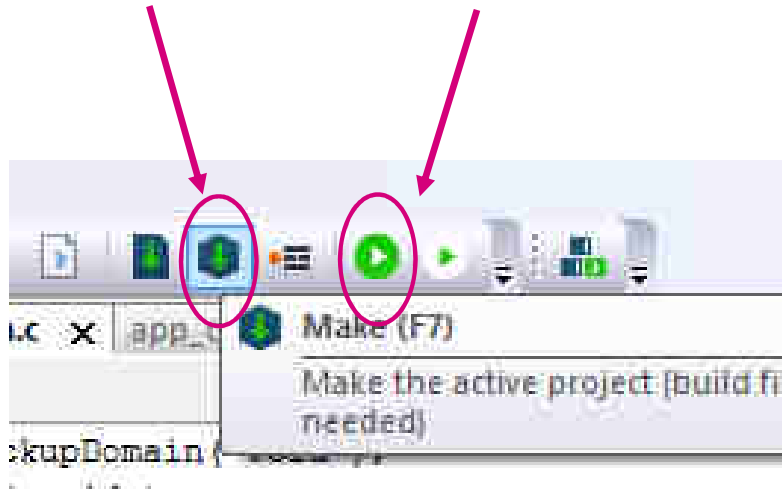
```
COPY_CRS_UUID(uuid.Char_UUID_128, led_uuid);  
hciCmdResult = aci_gatt_add_char(CRSContext.SvcHdle,  
                                UUID_TYPE_128,  
                                &uuid,  
                                2, /* Char_Value_Length */  
                                CHAR_PROP_WRITE_WITHOUT_RESP,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

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

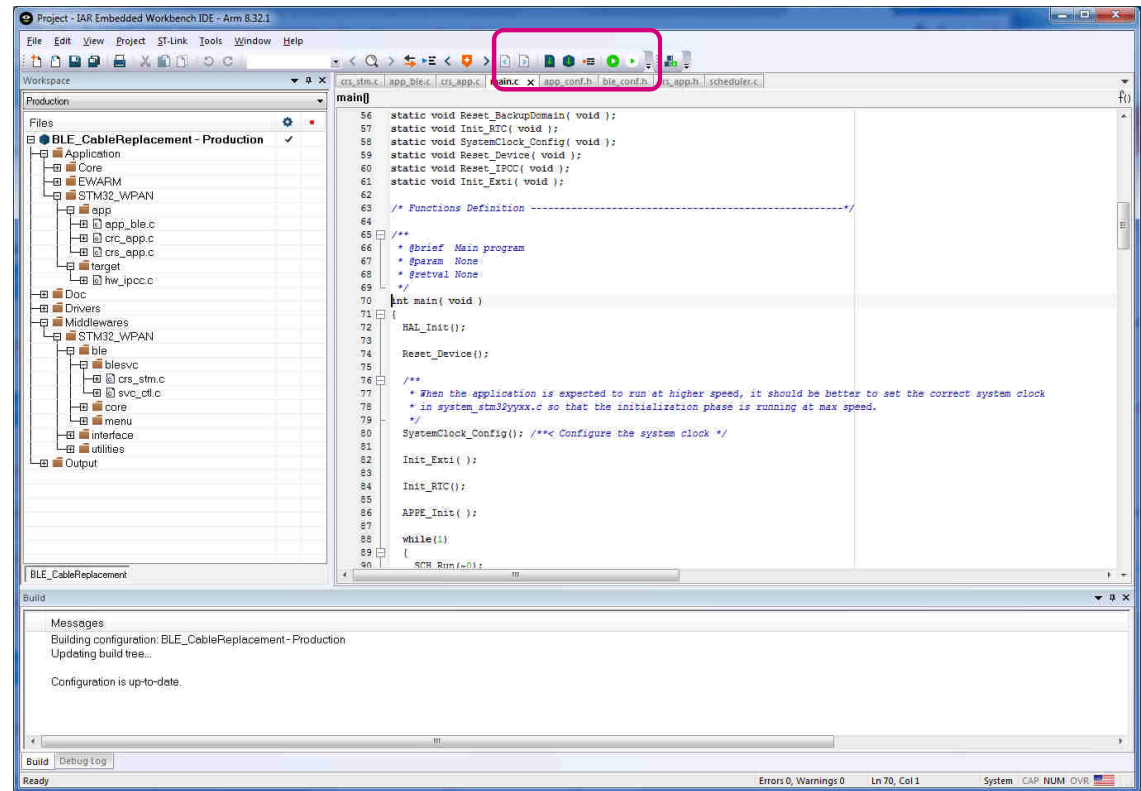
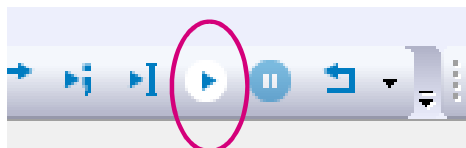
```
case STM_LED_WRITE_EVT:
    if(pNotification->DataTransferred.pPayload[0] == 0x01)
    {
        BSP_LED_On(LED_BLUE);
    }
    if(pNotification->DataTransferred.pPayload[0] == 0x00)
    {
        BSP_LED_Off(LED_BLUE);
    }
    break;
```

Build Project

Download and Debug



Run



Start LightBlue App

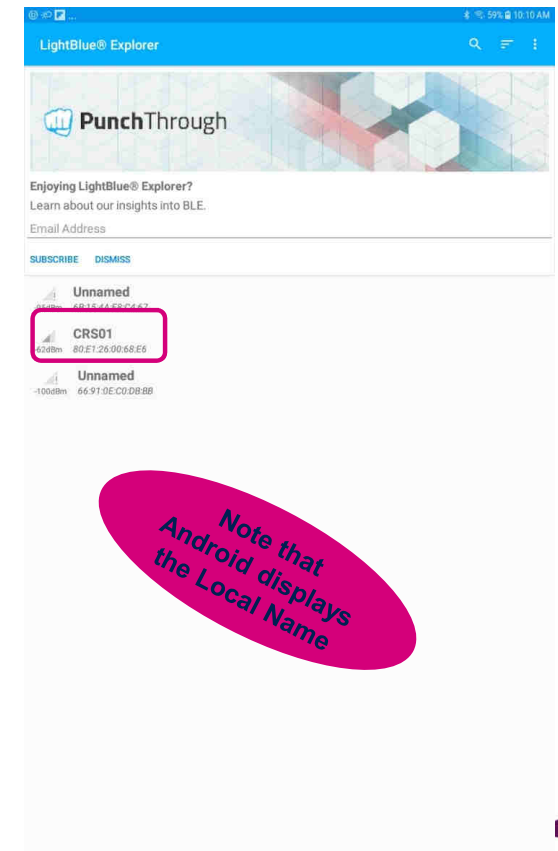
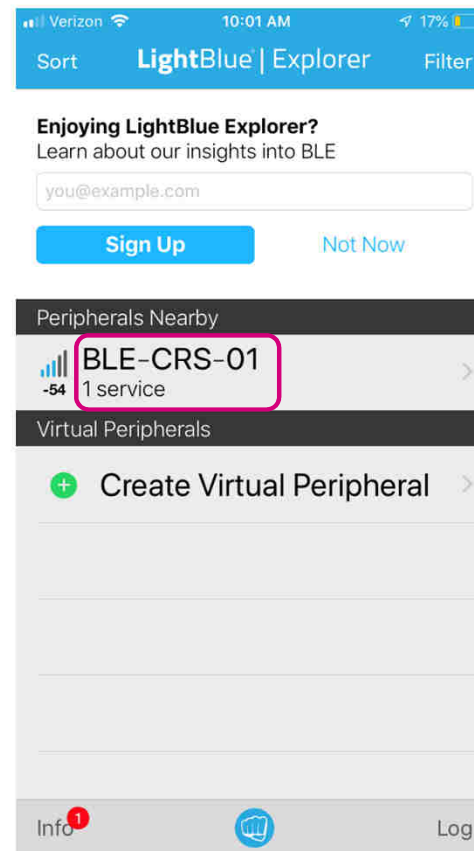
180

- Launch the LightBlue app



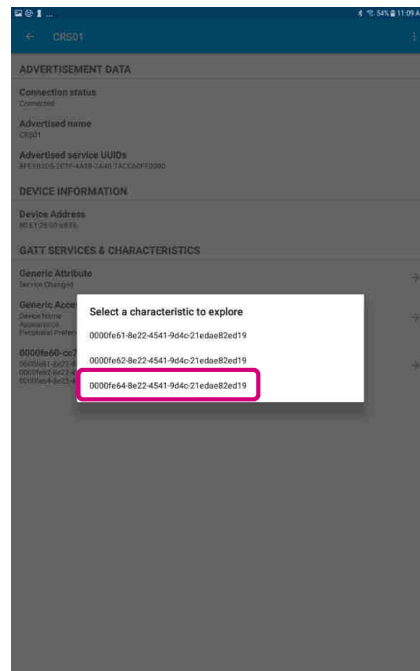
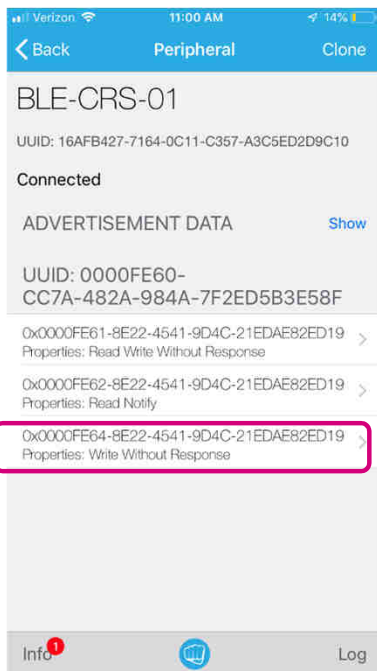
- Find your device

- Select your device

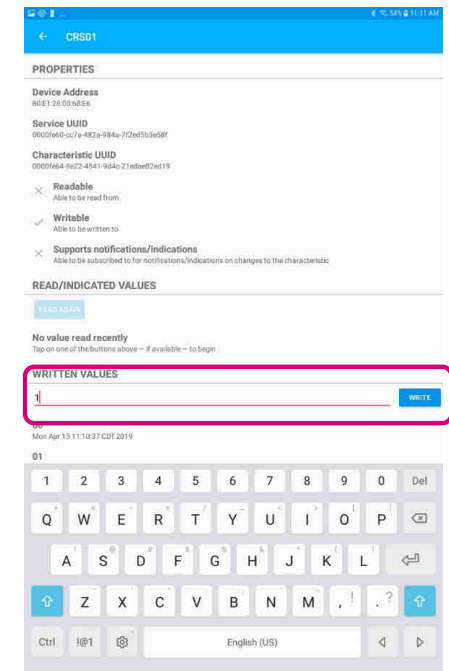
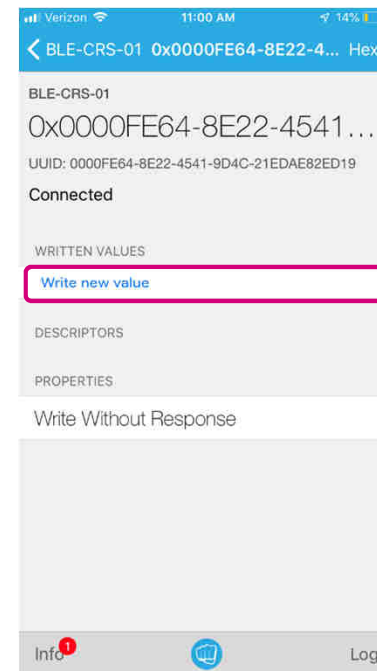


Note that
Android displays
the Local Name

Find your LED characteristic UUID



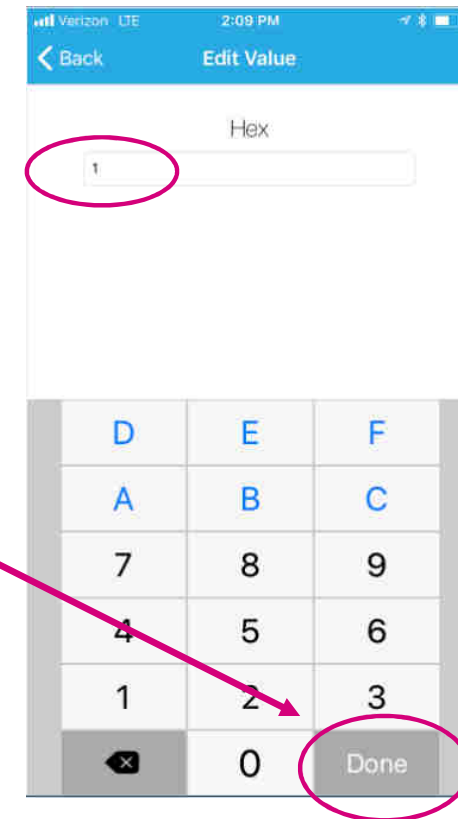
Write new value



Write a value

- LED ON = 1
- LED OFF = 0

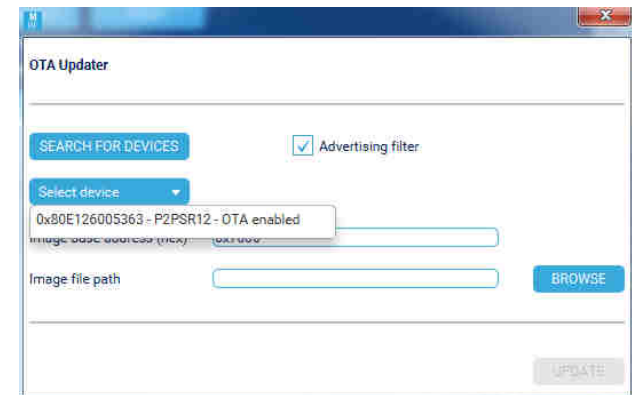
Concurrently, the CableReplacement characteristics can also be used





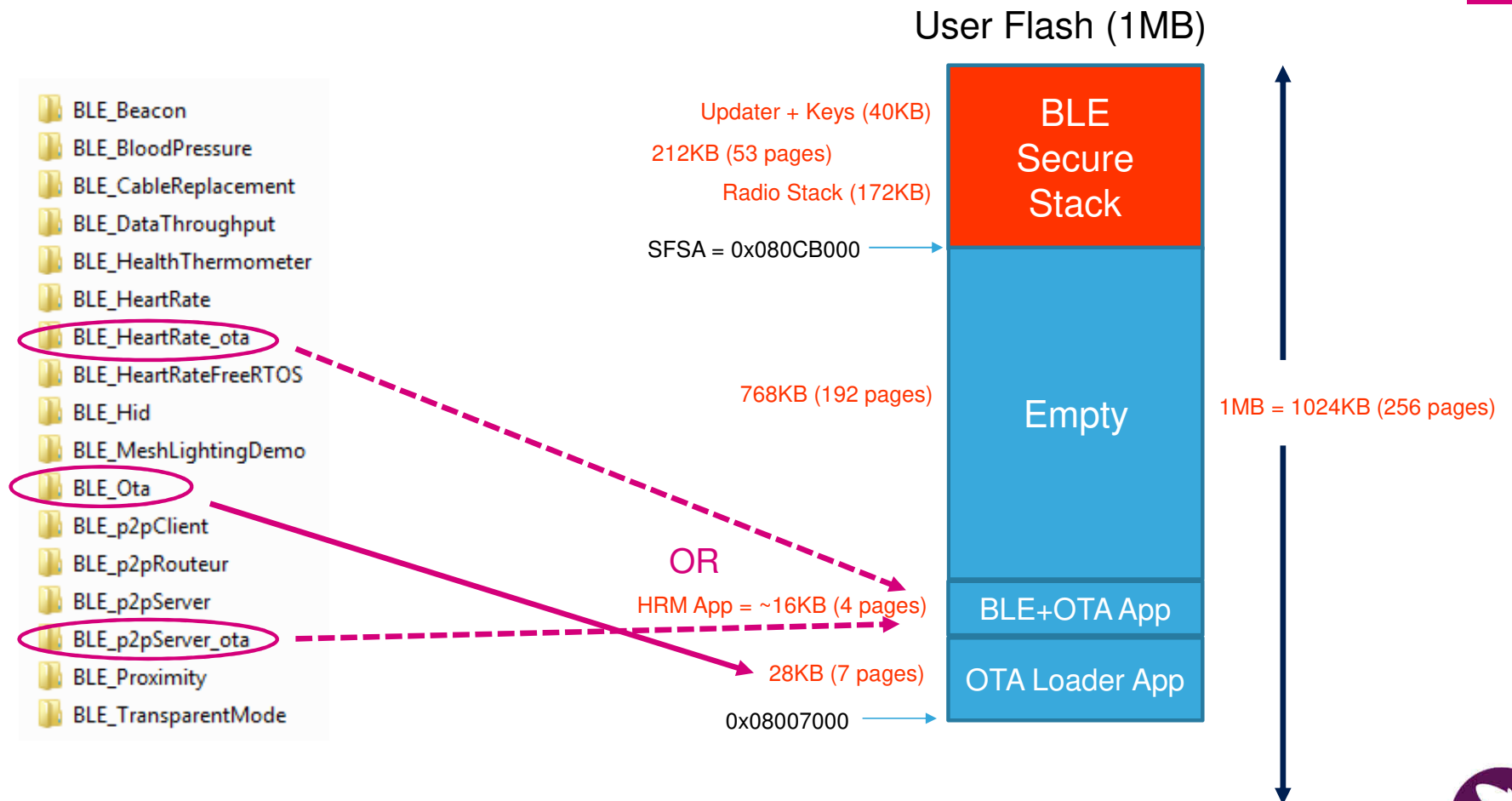
Bonus Hands-On!

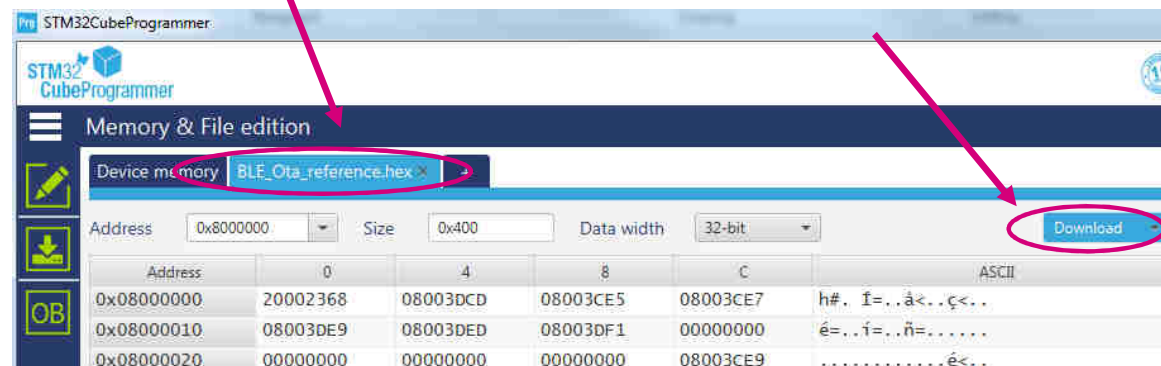
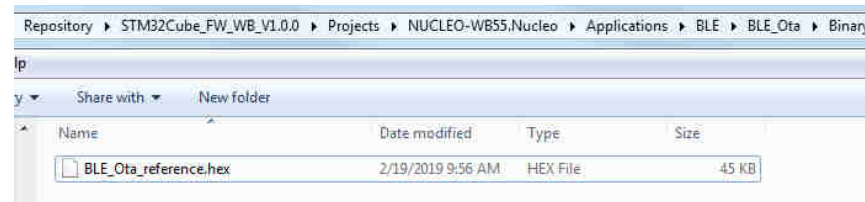
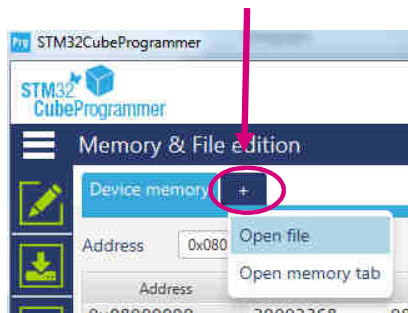
Over-the-Air Firmware Update



Over-The-Air Firmware Updates

184

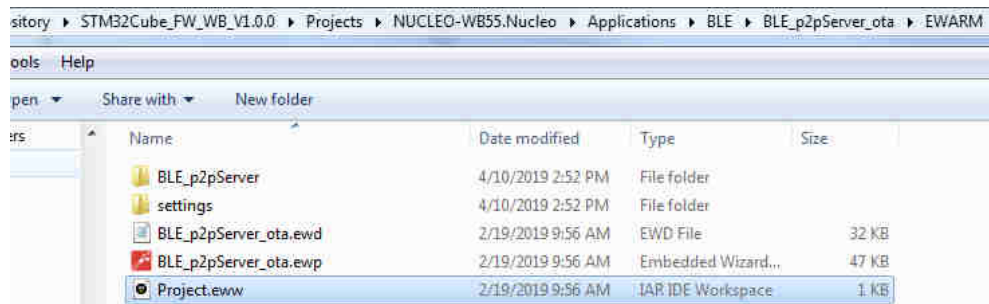


Flash **BLE_Ota_reference.hex** using CubeProgrammer

P2P Server + OTA project

186

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



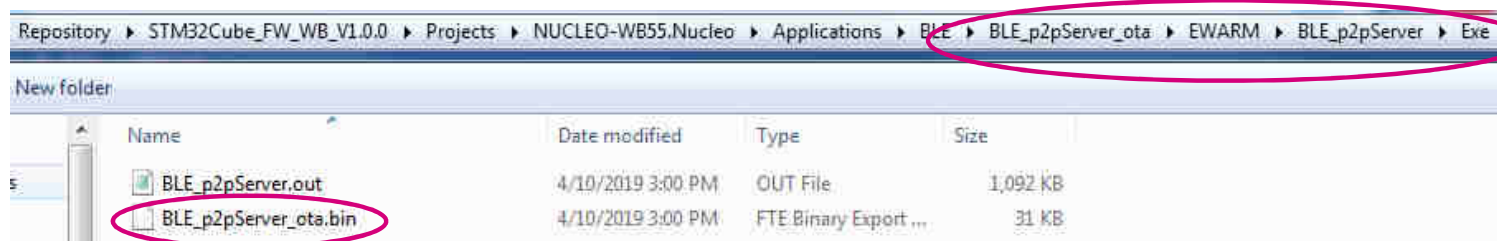
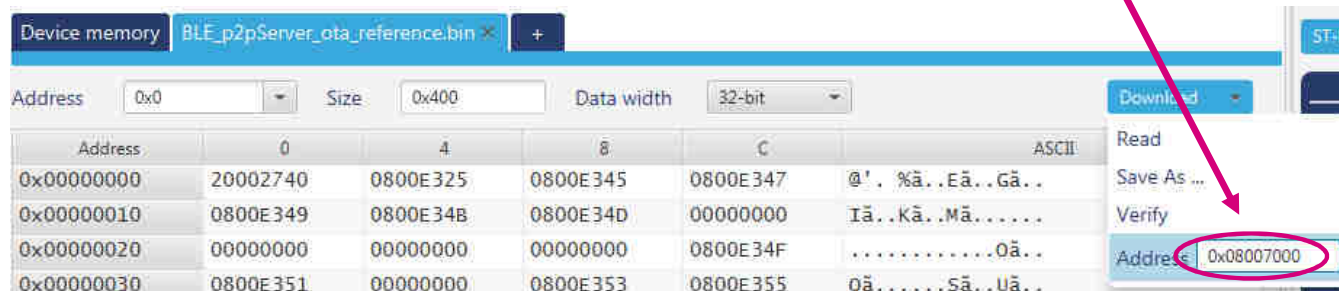
In **app_ble.c**

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

```
178 #define APPBLE_GAP_DEVICE_NAME_LENGTH 9
```

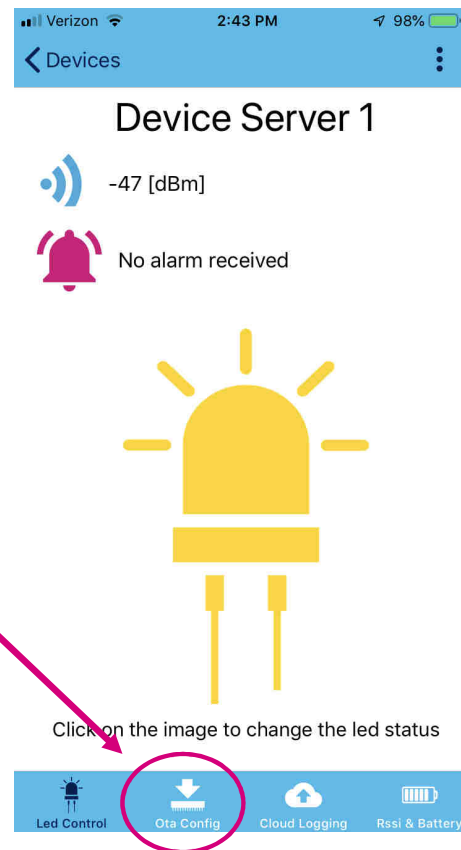
```
772 const char *name = "STM32WB12";
```

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

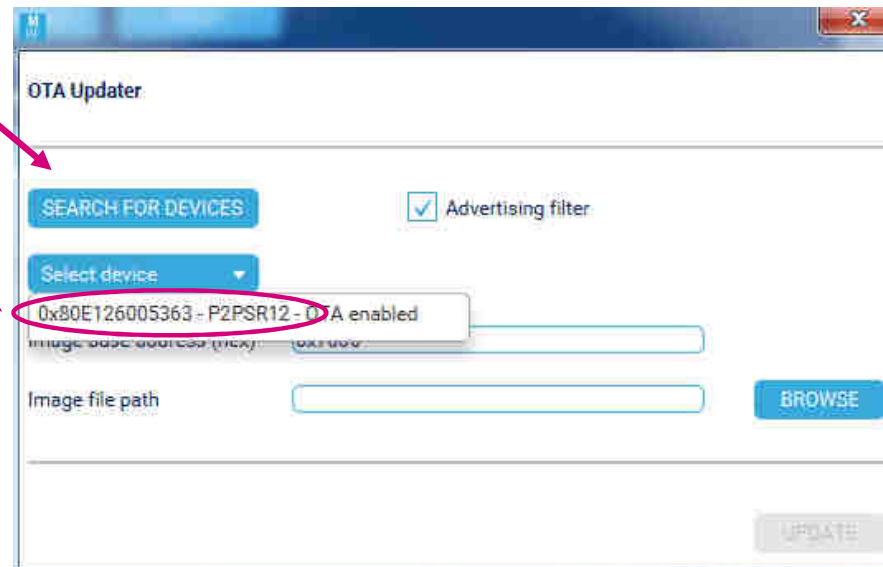
Connect to OTA-enabled device

Connect the Dongle and select **OTA Updater**

189



Search and Select your Device
(you can see your local name & BLE Address)



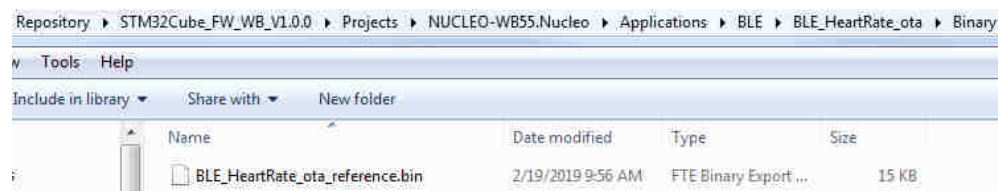
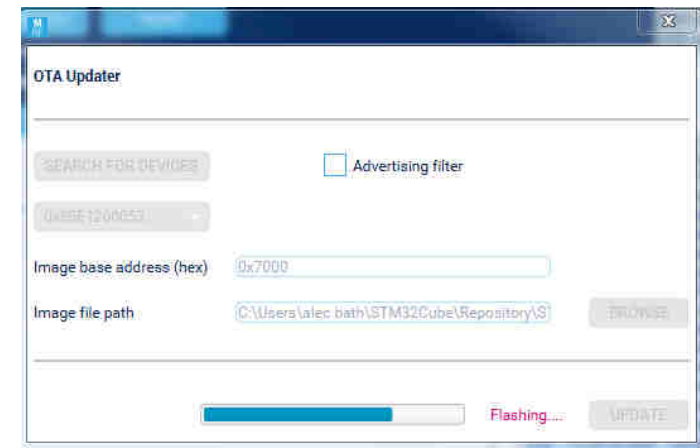
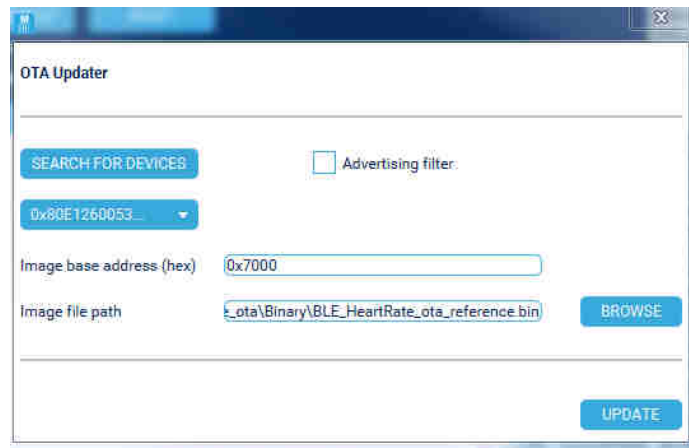
Update BLE Application

190

Browse for the other OTA binary

- BLE_HeartRate_ota_reference.bin

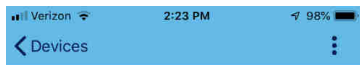
Update image



Update via ST BLE Sensor app

191

OTA capability detected
Click to start

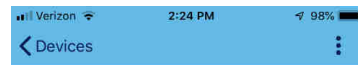


65 bpm

Energy: 20 kJ
RR Interval: 1.00 s



Erase & Reboot



First Sector to delete:

7

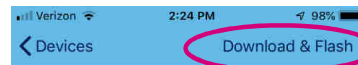
Number of sector to delete:

119

Reboot



Select Smartphone file (iCloud, etc)



STM32Cube_FW_WB-OTA

Version: 1.0.0

Mcu type: STM32WBXX

Address: 0x 7000



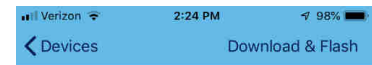
BLE_HeartRate_...ence
2/19/19
15 KB

BLE_p2pServer_...ence
2/19/19
32 KB

2 items, 26.05 GB available on iCloud



Flashing begins



STM32Cube_FW_WB-OTA

Version: 1.0.0

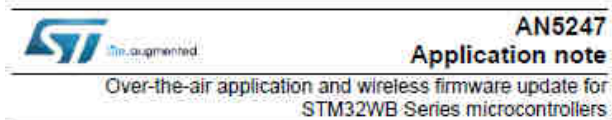
Mcu type: STM32WBXX

Address: 0x 7000

Flashing the new firmware

1860/31996 bytes

AN5247 details the OTA application in further detail.



Introduction

This document describes the procedure for over-the-air (OTA) firmware update on STM32WB 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

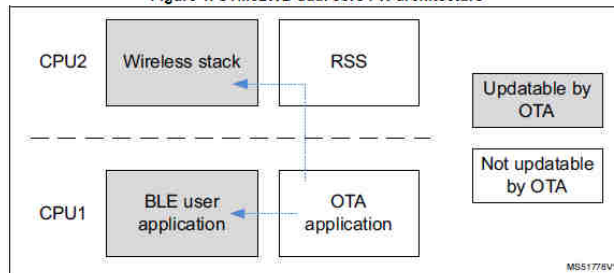
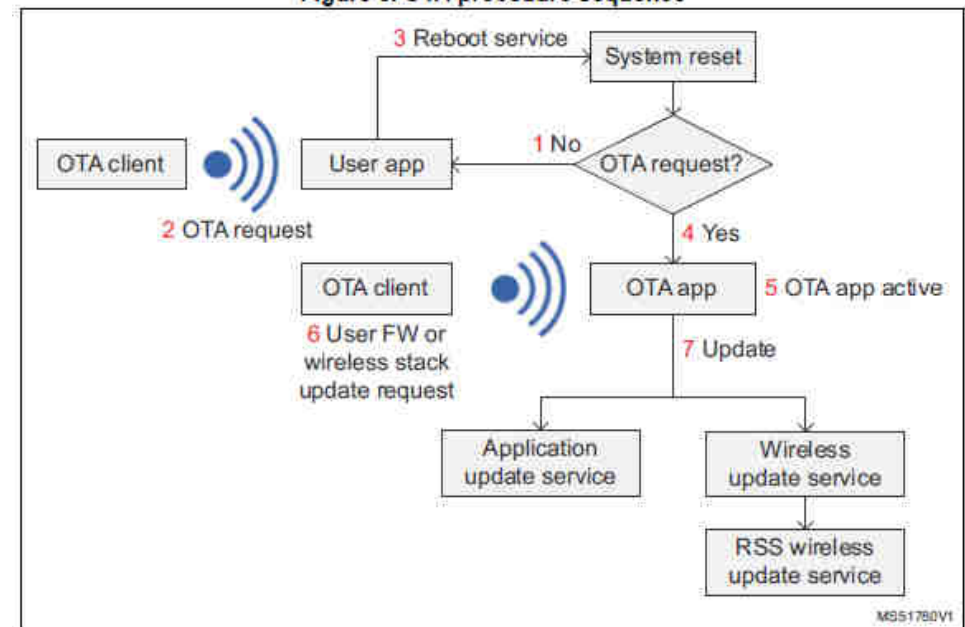


Figure 3. OTA procedure sequence



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



www.st.com/stm32wb

