



BlueNRG-1

BLE SOC

AMS

Application team EMEA



BlueNRG-1 BLE SOC

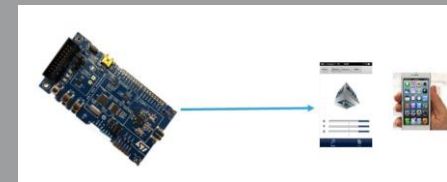
BlueNRG-1 and associated DK
(& promotion) package presentation



BLE concept demystification



Enable BLE link over BlueNRG-1



BlueNRG-1

BLE SOC presentation



BlueNRG-1 BLE SOC presentation

BlueNRG-1 Power consumption figures

BlueNRG-1 Development Tools

BlueNRG-1 SW API to ease your design

How to promote - Navigator tool

Bluetooth® SMART offering Roadmap

Discrete

Application Processor = SoC

Network Processor

BlueNRG-MS

Cortex-M0 **BLE 4.1**
Master & Slave
Output power: +8dBm
Rx: 7.3mA
Tx: 8.2mA@0dBm
QFN32, WCSP34

BlueNRG

Cortex-M0 **BLE 4.0**
Output power: +8dBm
Rx: 7.3mA
Tx: 8.2mA@0dBm
QFN32, WCSP34

no more use for new design

BlueNRG-1

BLE 4.2
Cortex-M0
160KB Flash, 24kB RAM
I²C, SPI, UART, ADC
Output power: +8dBm
Rx: 7.3mA
Tx: 8.2mA@ 0dBm
QFN32 (AEC), WCSP34

BLE 4.2 : secure connection & privacy 1.2

BlueNRG-2

BLE 4.2
Cortex-M0
256KB Flash, 24kB RAM
I²C, SPI, UART, ADC
Output power: +8dBm
Rx: 7.3mA
Tx: 8.2mA@ 0dBm
QFN32 (AEC), WCSP34

BLE 4.2 : data packet extension length

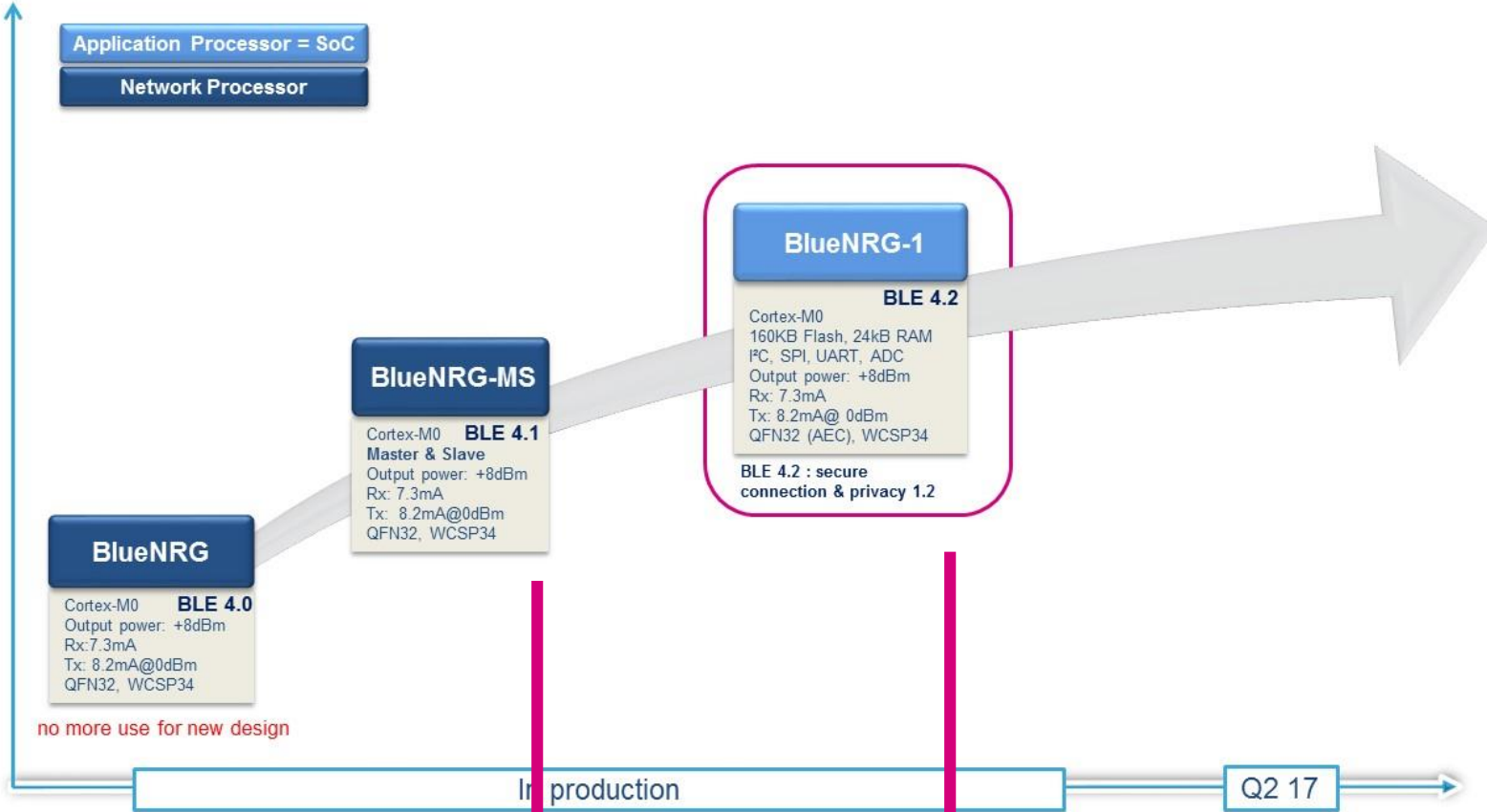
In production

Q1 17

2018

Bluetooth® SMART offering Roadmap

Module



Application Processor = SoC
Network Processor

BlueNRG
Cortex-M0 **BLE 4.0**
Output power: +8dBm
Rx: 7.3mA
Tx: 8.2mA@0dBm
QFN32, WCSP34

no more use for new design

BlueNRG-MS
Cortex-M0 **BLE 4.1**
Master & Slave
Output power: +8dBm
Rx: 7.3mA
Tx: 8.2mA@0dBm
QFN32, WCSP34

BlueNRG-1
BLE 4.2
Cortex-M0
160KB Flash, 24kB RAM
I²C, SPI, UART, ADC
Output power: +8dBm
Rx: 7.3mA
Tx: 8.2mA@ 0dBm
QFN32 (AEC), WCSP34
BLE 4.2 : secure connection & privacy 1.2

In production

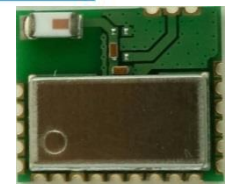
Q2 17

11.5 mm x 13.5 mm



SPBTLE-RF

11.5 mm x 13.5 mm



SPBTLE-1S

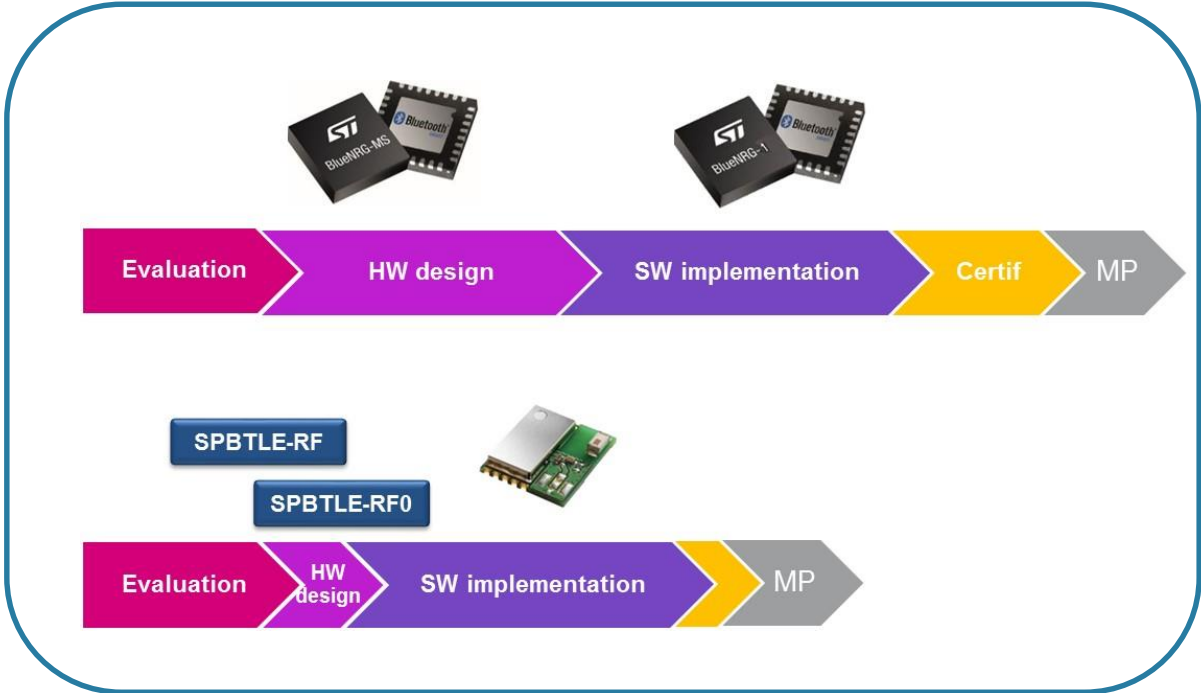
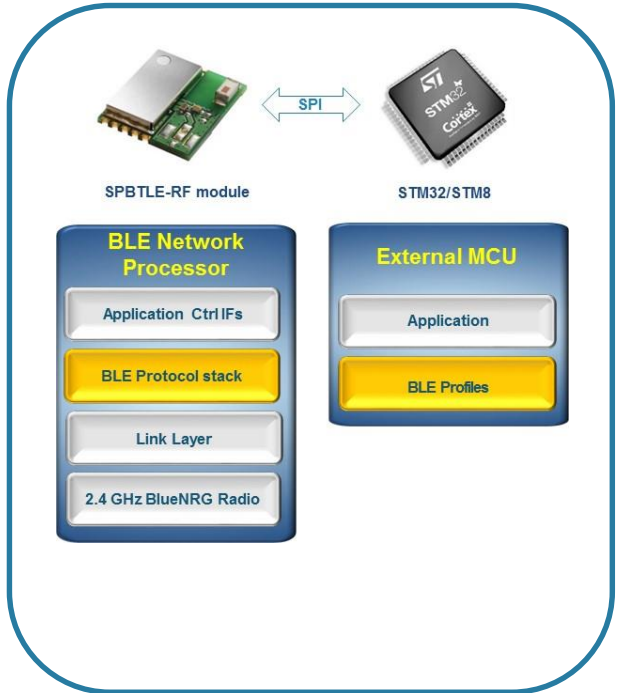


SPBTLE modules

BLE 4.1 RF Module 6

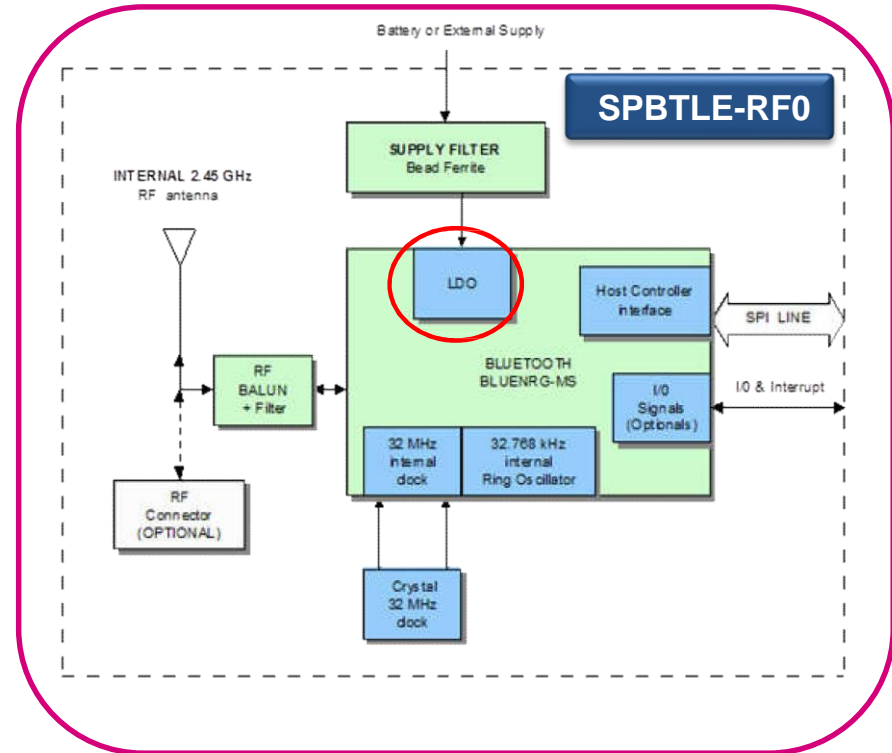
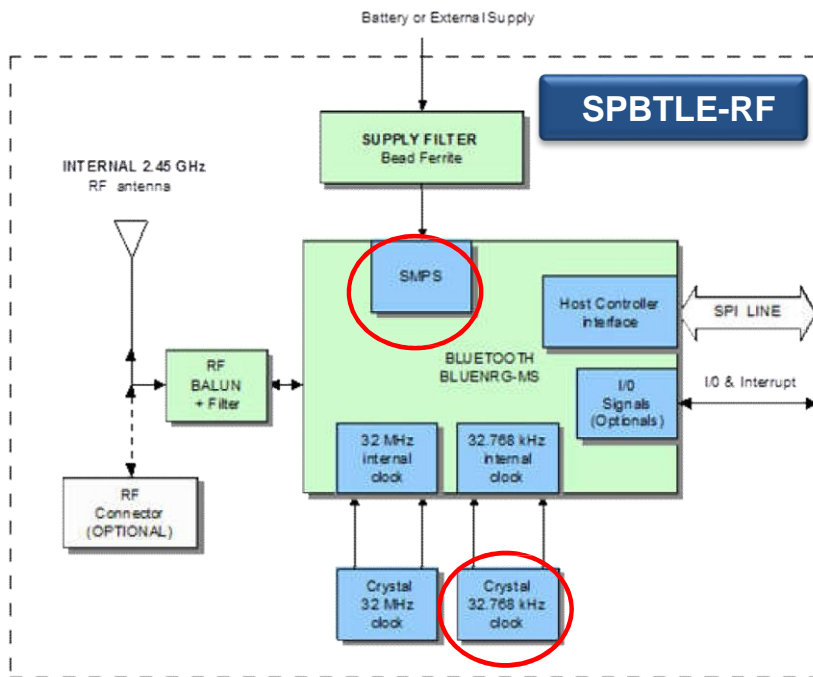
FCC, CE, IC
BLE certified

Modules designed for time to market





SPBTLE-RF0 – Sub 3\$ ST Bluetooth Smart module



SPBTLE-RF0 Pin to Pin compatible SPBTLE-RF

Bluetooth® SMART offering Roadmap

Discrete

Application Processor = SoC

Network Processor

BlueNRG-MS

Cortex-M0 **BLE 4.1**
Master & Slave
Output power: +8dBm
Rx: 7.3mA
Tx: 8.2mA@0dBm
QFN32, WCSP34

BlueNRG

Cortex-M0 **BLE 4.0**
Output power: +8dBm
Rx: 7.3mA
Tx: 8.2mA@0dBm
QFN32, WCSP34

no more use for new design

BlueNRG-1

BLE 4.2
Cortex-M0
160KB Flash, 24kB RAM
I²C, SPI, UART, ADC
Output power: +8dBm
Rx: 7.3mA
Tx: 8.2mA@ 0dBm
QFN32 (AEC), WCSP34

BLE 4.2 : secure connection & privacy 1.2

BlueNRG-2

BLE 4.2
Cortex-M0
256KB Flash, 24kB RAM
I²C, SPI, UART, ADC
Output power: +8dBm
Rx: 7.3mA
Tx: 8.2mA@ 0dBm
QFN32 (AEC), WCSP34

BLE 4.2 : data packet extension lenght

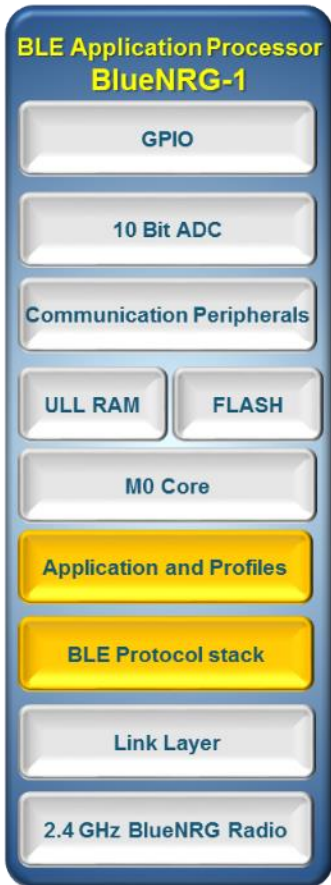
In production

Q1 17

2018

BlueNRG-1 Application Processor

Highlights



Application and Profiles

- Application and BlueNRG profiles run in BlueNRG-1

BLE stack

- 4.2 Bluetooth Low Energy compliancy
 - Privacy 1.2
 - Secure connection

Q2
2017

Peripherals Drivers

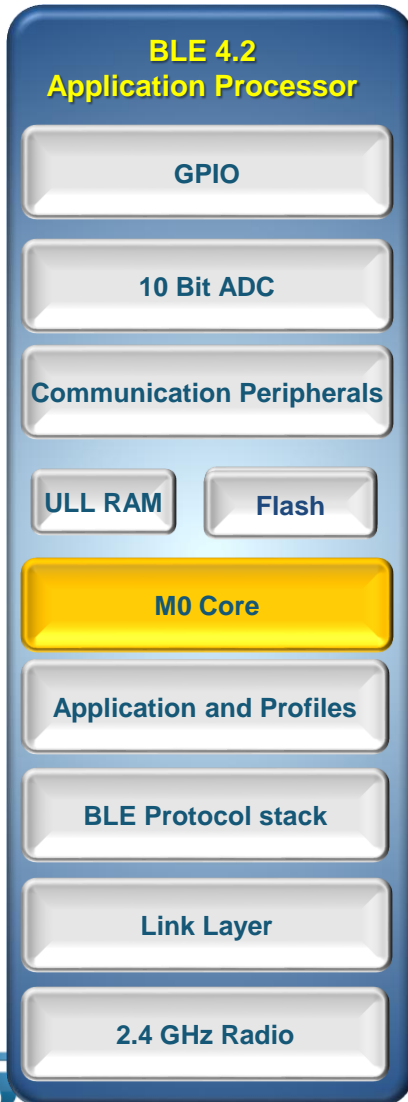
- STM32 “like” drivers





BlueNRG-1 Application Processor

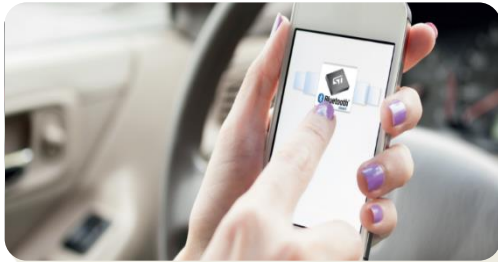
Applications



BlueNRG-1 capabilities to enable low to mid end smart connected applications



Beacon
Sensor tags
Remote Control



Automotive grade



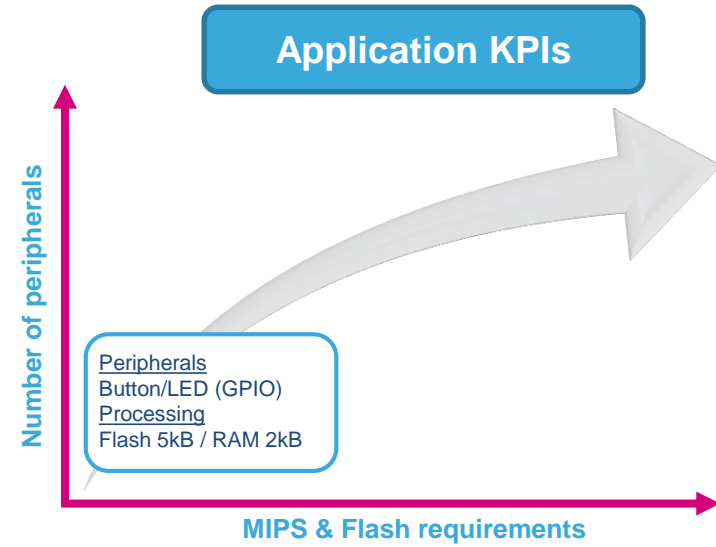
BlueNRG-1 Application Processor Applications



Beacon



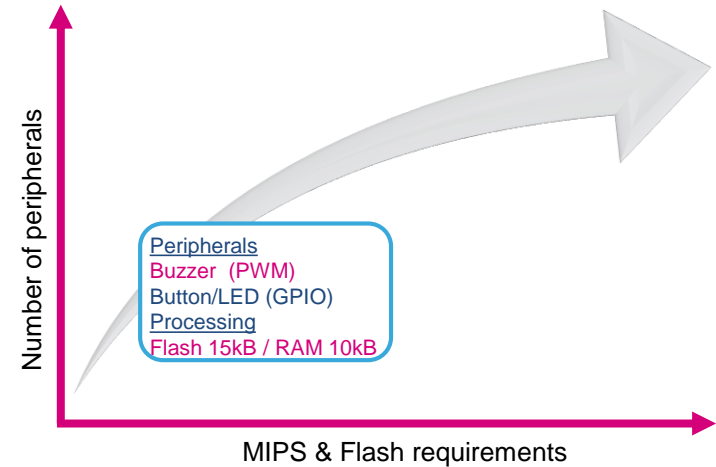
- broadcasting data



Key Fob



- connected basic application
- localization



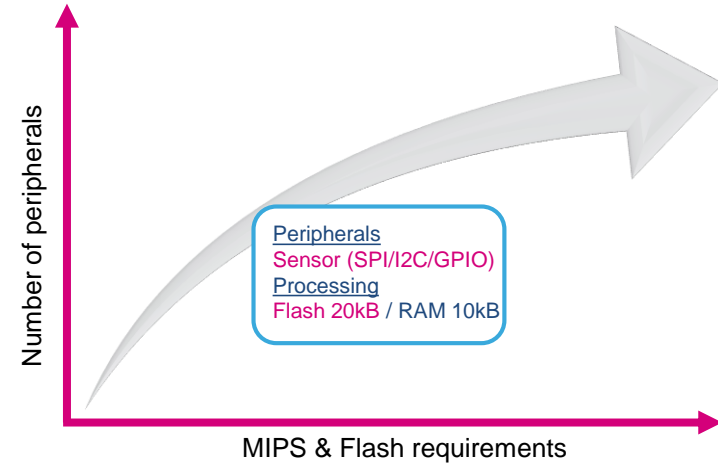


BlueNRG-1 Application Processor Applications

Sensor Tag



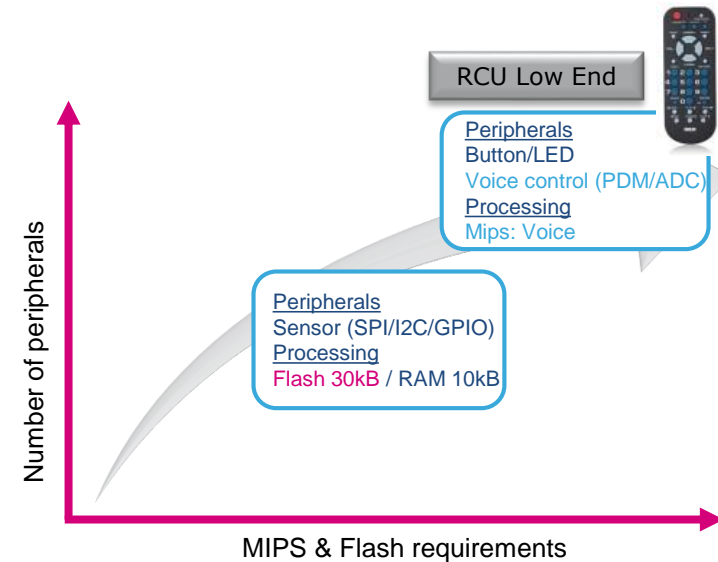
➤ Collect Sensor data



Appliance Remote Control



- Control remote device
- Device Configuration
- Device Application update



BlueNRG-1

Main Figures



BLE Application Processor BlueNRG-1

GPIO

10 Bit ADC

Communication Peripherals

ULL RAM

Flash

M0 Core

Application and Profiles

BLE Protocol stack

Link Layer

2.4 GHz BlueNRG Radio



Superior Battery life (DCDC)

- RX 7.3mA
- TX 8.2mA @0dBm
- **Sleep 1 μ A**

Excellent RF perfs

- Best in Class Output Power **Level: +8dBm**
- Receiver sensitivity -88dBm

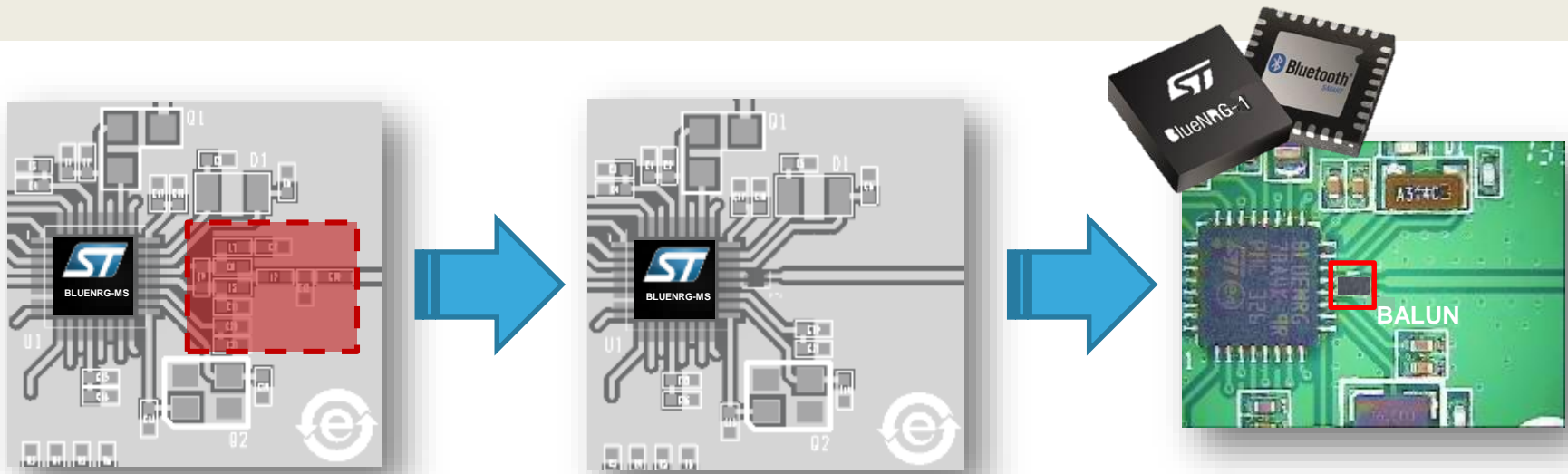
Two Package flavors

- Extended Temperature Range: up to 105⁰C
- **WCSP34 2.65x2.65mm**
- **QFN32 5x5 mm (Automotive)**

BlueNRG family optimized footprint

RF balun and filtering

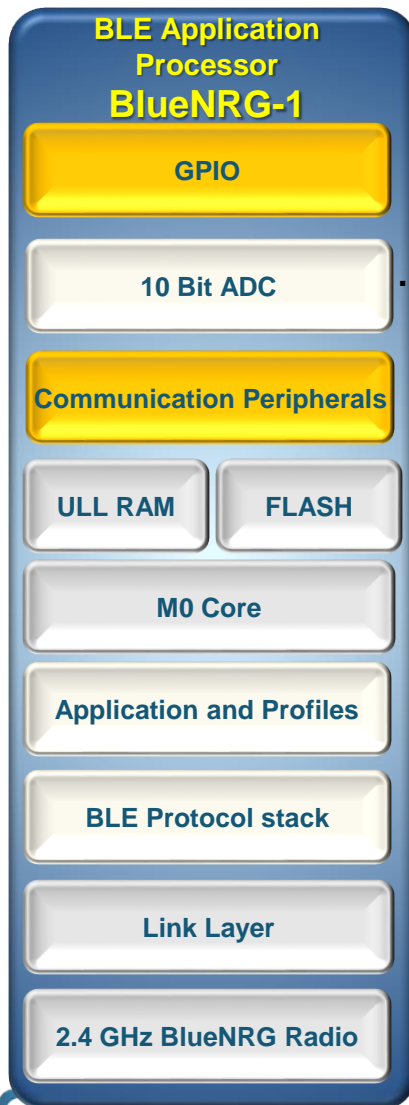
- The **BALF-NRG-01D3** is an ultra-miniature balun integrating a matching network and harmonics filter. The matching impedance is customized for ST's **BlueNRG** , **BlueNRG-MS** , **BlueNRG-1** transceiver (both QFN and WLCSP versions). It uses ST's IPD technology on a non-conductive glass substrate which optimizes RF performance.
- PACKAGE: flip-chip package 4 bumps, 1.2 mm² footprint.



Footprint and Cost optimization

- From **9** to **1** SMD
- PCB real-estate savings: from **32mm²** to **1.2mm²**
- Optimized RF tuning antenna matching
- Simplified PCB layout and lower manufacturing costs

BlueNRG-1 Peripherals



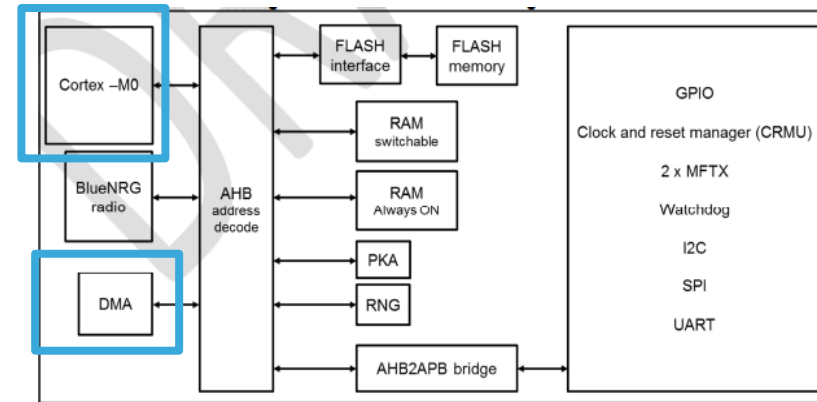
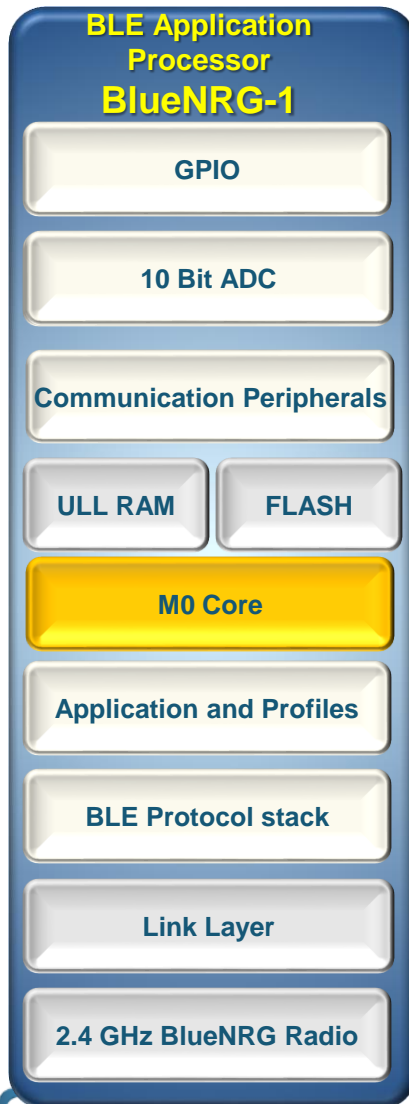
GPIO

- 15 pin (QFN package)
- 14 pin (CSP package)
- Wake up function

Communication Peripherals

- SPI: Master and Slave support
- I2C: Baud rate supported up to 400 kb/s
- UART: Programmable Baud Rate, support of HW flow control
- PDM streaming (audio MEMS interface)

BlueNRG-1 Core



M0 Core

- 32 Bit architecture, 32 MHz speed
- ultra-low leakage retention state
- SWD debug port

TIMER

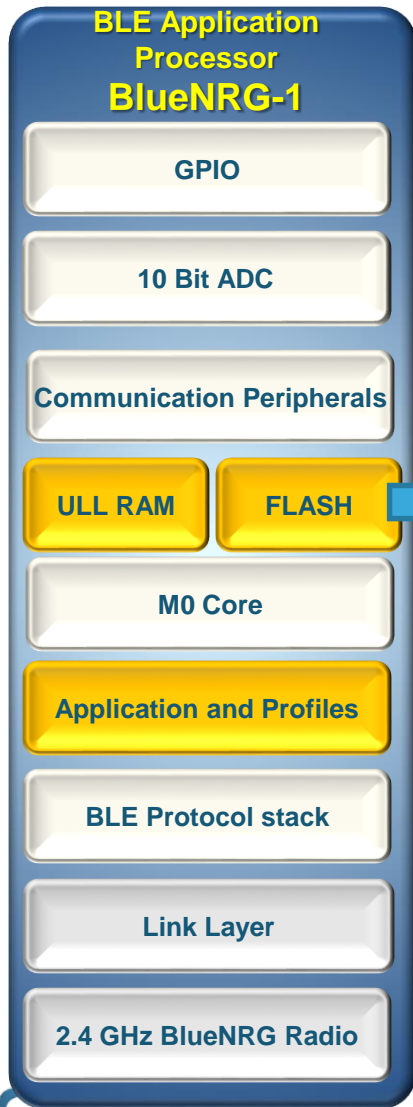
- MFTX: Two multi function timers

DMA

- Data transfer without CPU intervention



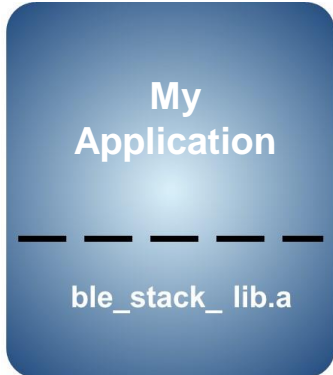
BlueNRG-1 Application Flash



ULL RAM

- One 12kB block always in retention
- One 12kB block switchable

160KB



FLASH

Application and Profiles

- 70kB available for application code (full feature BLE stack)
- 110kB available for application code (minimum feature BLE stack)

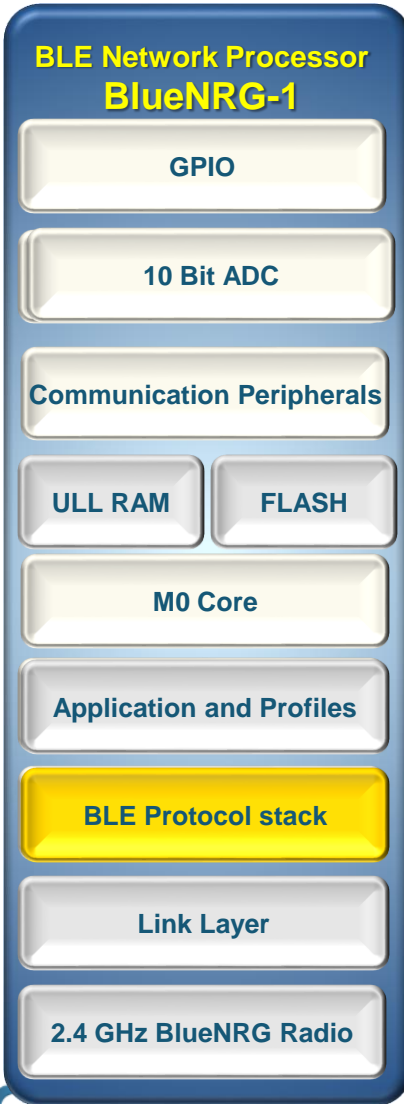
~100KB for application

BlueNRG-1

BLE stack

Q1 2017

BLE 4.2 - Improved privacy and security



Advantages	Features	End User Examples
<p>Industry-leading Privacy Keeps Bluetooth Smart devices from being tracked</p>	<p>LE Privacy 1.2</p>	<p>A Bluetooth Smart location tracker can only be followed by the owner or trusted group all while consuming less power</p>
<p>More Power Efficient Introduces refinements that help Bluetooth Smart devices save even more energy</p>		
<p>Highly Secure Features FIPS-compliant encryption ensuring confidential data stays that way</p>	<p>LE Secure Connections</p>	<p>A Bluetooth Smart lock or other smart home device provides industry standard security for added user confidence during device pairing</p>



ECC Encryption

Elliptic-Curve-Cryptography



Elliptic Curve Cryptography

BlueNRG-1

BLE SOC presentation



BlueNRG-1 BLE SOC presentation



BlueNRG-1 Power consumption figures



BlueNRG-1 Development Tools



BlueNRG-1 SW API to ease your design



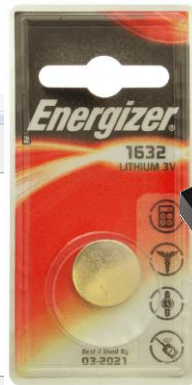
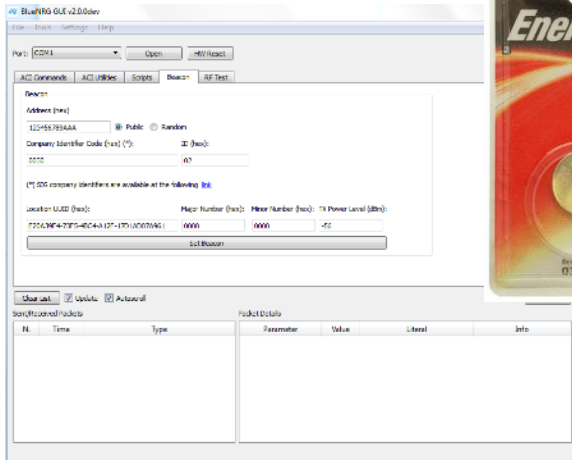
How to promote - Navigator tool

Optimized for ultra-low-power

“Engineered to Advertise”

Ultra-low-power consumption in advertisement mode

16uA @ 1.28s



Prolonged battery life

Low Power average power consumption

Beyond DS specs:
how to benchmark
for real-case
scenarios

From **PEAK** to **AVERAGE**
power consumption

- RX **7.7mA**
- TX **8.3mA** @0dBm
- Sleep **0.9µA**
- Shut Down **2.5nA**

+

Ultra-fast **SLEEP** to
ACTIVE transition time

=

Ultra-low average
power
consumption

Advertisement Scenario

(with 15 byte payload)

Advertising Interval

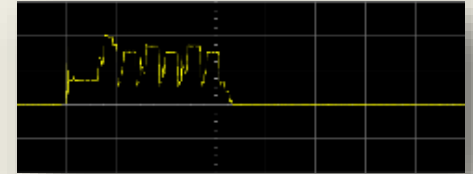
Average Consumption
SMPS ON **SMPS OFF**

1.28s

16µA **26µA**

500ms

37µA **64µA**



Connection Scenario

(without payload)

Connection Interval

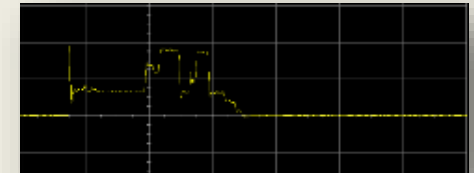
Average Consumption
SMPS ON **SMPS OFF**

1.28s

5.5µA **9.6µA**

30ms

167µA **316µA**



Connection Scenario

(with payload)

Connection Interval

Application Data

Average Consumption

SMPS ON **SMPS OFF**

1.28s

2 byte

6µA **10µA**

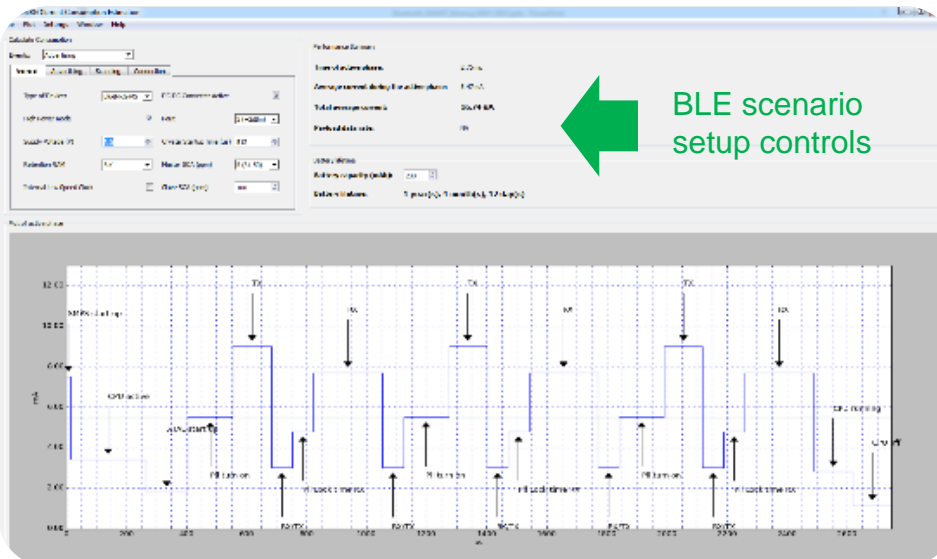
30ms

20 byte

233µA **437µA**

Measurement conditions: Vin=3.0V, Slave Mode, 32KHz XOSC - High Power Mode - Pout 2dBm

Power consumption Tool



BLE scenario
setup controls

- ST provides a **Current Consumption Estimation Tool**
- It enables the user to estimate the average current consumption and the battery lifetime in the applicative cases
- The user can select:
 - **General:**
 - Supply voltage
 - TX output power
 - Master/Slave sleep clock accuracy
 - Retention RAM
 - **Connection Advertising or Scanning Interval**
 - **Data length**
 - **DC-DC converter active or not**



Quickly estimate
the battery life

STSW-BNRG001

BlueNRG current consumption estimation tool

<http://www.st.com/web/en/catalog/tools/PF260405>

BlueNRG-1

BLE SOC presentation



BlueNRG-1 BLE SOC presentation

BlueNRG-1 Power consumption figures



BlueNRG-1 Development Tools

BlueNRG-1 SW API to ease your design

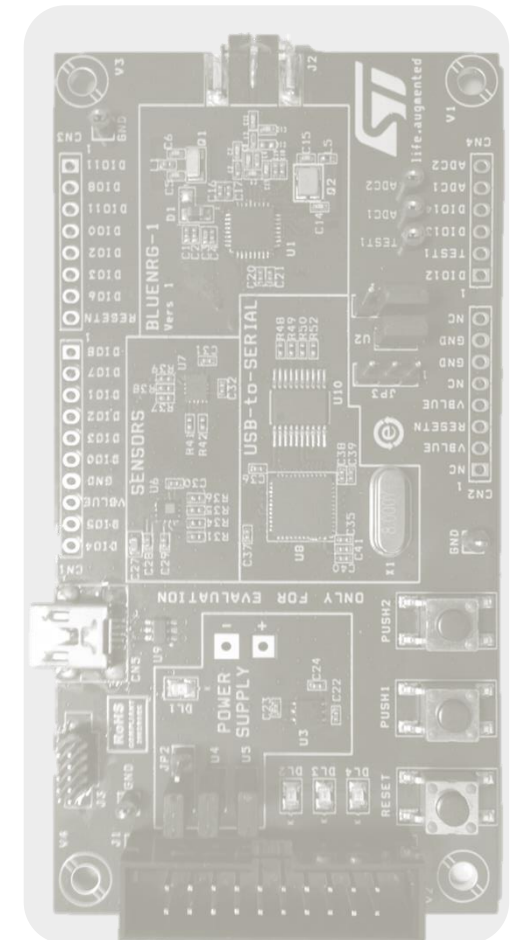
How to promote - Navigator tool

BlueNRG-1 Development Tools

Full-featured DK

DK Resources

- High-level abstraction layer APIs
- Firmware documentation
- Pre-compiled HEX files (for rapid evaluation)
- Examples and templates, in source code
- Drivers for sensors (motion and environmental)
- Beacon Application for iOS / Android (source code)
- Multiple tool-chains supported (IAR, Keil, Atollic, GCC)
- Real-time debugging capabilities



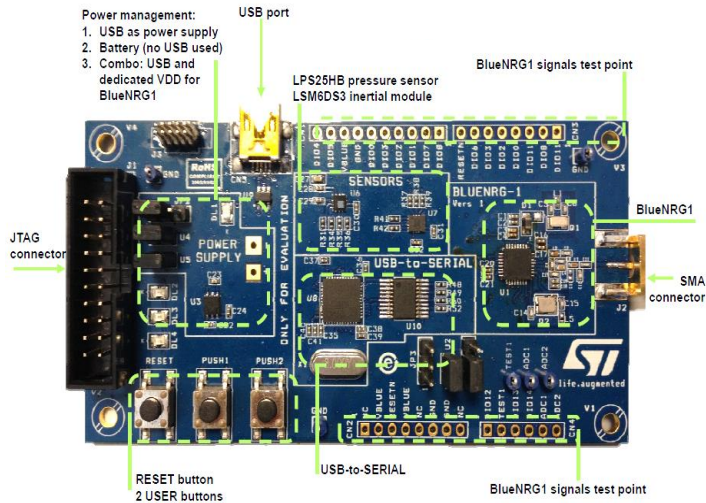


BlueNRG-1 Development Tools

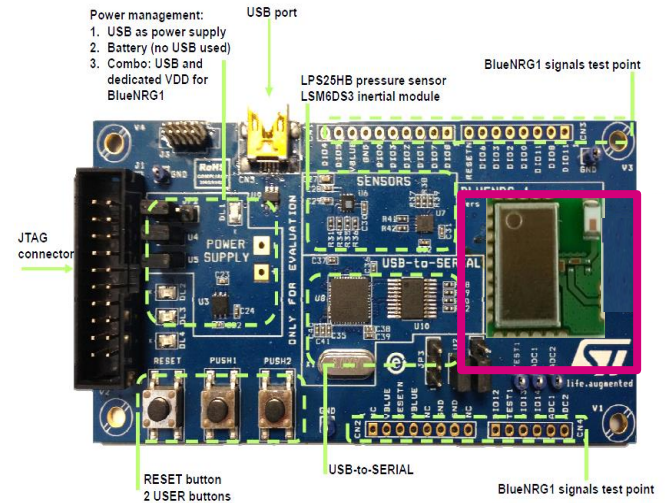
HW material

DK HW Resources

BlueNRG-1



SPBTLE-1S



STEVAL IDB007V1

STEVAL IDB007V1M

1 SW development kit
@ STSW-BLUENRG1-DK

BlueNRG-1 Development Tools

SW material

SW DK - BLE examples

```
void main(void)
{
    uint8_t ret;

    /* System Init */
    SystemInit();

    /* Identify BlueNRG-1 platform */
    SdkEvalIdentification();

    /* Init the UART peripheral */
    SdkEvalComUartInit(UART_BAUDRATE);

    /* BlueNRG-1 stack init */
    ret = BlueNRG_Stack_Initialization(&BlueNRG_Stack_Init_params);
    if (ret != BLE_STATUS_SUCCESS) {
        printf("Error in BlueNRG_Stack_Initialization() 0x%02x\r\n", ret);
        while(1);
    }

    /* Init the BlueNRG-1 device */
    Device_Init();

    /* Start Beacon Non Connectable Mode*/
    Start_Beaconing();

    printf("BlueNRG-1 BLE Beacon Application (version: %s)\r\n", BLE_BEACON_VERSION_STRING);

    while(1)
    {
        /* BlueNRG-1 stack tick */
        BTLE_StackTick();

        /* Enable Power Save according the Advertising Interval */
        BlueNRG_Sleep(SLEEPMODE_WAKETIMER, 0, 0, 0);
    }
}
```



« SPP like »
Chat Demo



BlueNRG-1 Development Tools

SW material

SW DK - Peripherals examples

```
void main(void)
{
    uint8_t ret;

    /* System Init */
    SystemInit();

    /* Identify BlueNRG-1 platform */
    SdkEvalIdentification();

    /* Init the UART peripheral */
    SdkEvalComUartInit(UART_BAUDRATE);

    /* BlueNRG-1 stack init */
    ret = BlueNRG_Stack_Initialization(&BlueNRG_Stack_Init_params);
    if (ret != BLE_STATUS_SUCCESS) {
        printf("Error in BlueNRG_Stack_Initialization() 0x%02x\r\n", ret);
        while(1);
    }

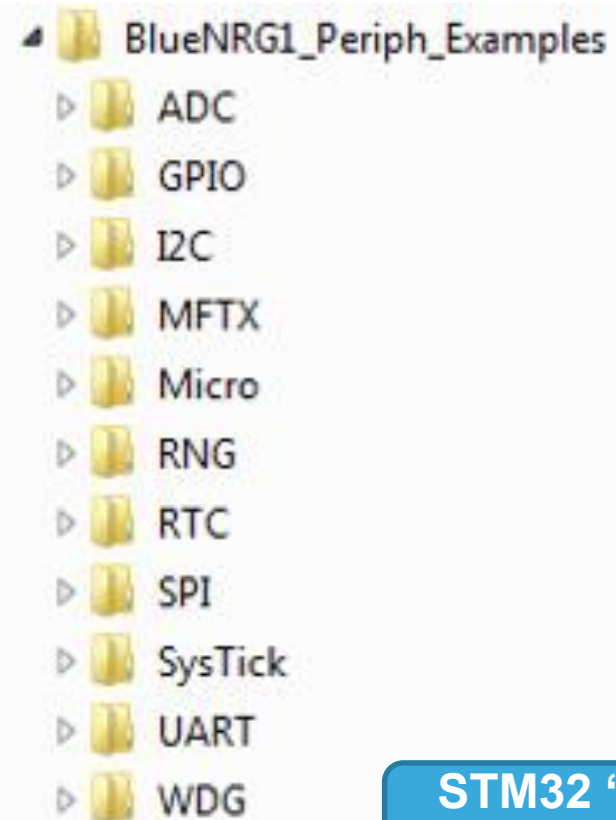
    /* Init the BlueNRG-1 device */
    Device_Init();

    /* Start Beacon Non Connectable Mode*/
    Start_Beaconing();

    printf("BlueNRG-1 BLE Beacon Application (version: %s)\r\n", BLE_BEACON_VERSION_STRING);

    while(1)
    {
        /* BlueNRG-1 stack tick */
        BTLE_StackTick();

        /* Enable Power Save according the Advertising Interval */
        BlueNRG_Sleep(SLEEPMODE_WAKETIMER, 0, 0, 0);
    }
}
```



STM32 “like”
drivers





BlueNRG-1 Development Tools

SW material- Documentation

SW DK - Documentation

Following is an example of advertising packet that lists the Service UUID that the device implements (General Discoverable flag, Tx power = 48dbm, Service data = temperature service and 16 bits service UUIDs).

Figure 8: Advertising packet with AD type flags

Protocol	Advertising Address	Advertising Interval	Physical Length	Advertising Address	Flags/LE General Discoverable flag	Tx Power Level (dBm)	Service Data (UUIDs)	Service Interval (Temperature sensor)	CCC
0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

The Flags AD type byte contains the following flag bits:

- Limited Discoverable Mode (bit 0);
- General Discoverable Mode (bit 1);
- BR/EDR Not Supported (bit 2, 1 is 1 on BLE);
- Simultaneous LE and BR/EDR to Same Device Capable (Controller) (bit 3);
- Simultaneous LE and BR/EDR to Same Device Capable (Host) (bit 4).

The Flags AD type shall be included in the advertising data if any of the bits are non-zero (it is not included in scan response).

PM0257 programming guide : BLE concepts & associated ST API

Doxygen documentation : BlueNRG-1 BLE and peripherals API

BlueNRG-1 Peripherals drivers APIs

life.augmented

Main Page | Modules | Classes | Files

File List

Here is a list of all files with brief descriptions:

- BlueNRG1_adc.h This file contains all the functions prototypes for the ADC firmware library
- BlueNRG1_dma.h This file contains all the functions prototypes for the DMA firmware library
- BlueNRG1_dma2.h This file contains all the functions prototypes for the DMA2 firmware library
- BlueNRG1_gpio.h This file contains all the functions prototypes for the GPIO firmware library
- BlueNRG1_i2c.h This file contains all the functions prototypes for the I2C firmware library
- BlueNRG1_i2c3.h This file contains all the functions prototypes for the I2C3 firmware library
- BlueNRG1_i2s.h This file contains all the functions prototypes for the I2S firmware library
- BlueNRG1_irq.h This file contains all the functions prototypes for the IRQ firmware library
- BlueNRG1_irq2.h This file contains all the functions prototypes for the IRQ2 firmware library
- BlueNRG1_irq3.h This file contains all the functions prototypes for the IRQ3 firmware library
- BlueNRG1_irq4.h This file contains all the functions prototypes for the IRQ4 firmware library
- BlueNRG1_irq5.h This file contains all the functions prototypes for the IRQ5 firmware library
- BlueNRG1_irq6.h This file contains all the functions prototypes for the IRQ6 firmware library
- BlueNRG1_irq7.h This file contains all the functions prototypes for the IRQ7 firmware library
- BlueNRG1_irq8.h This file contains all the functions prototypes for the IRQ8 firmware library
- BlueNRG1_irq9.h This file contains all the functions prototypes for the IRQ9 firmware library
- BlueNRG1_irq10.h This file contains all the functions prototypes for the IRQ10 firmware library
- BlueNRG1_irq11.h This file contains all the functions prototypes for the IRQ11 firmware library
- BlueNRG1_irq12.h This file contains all the functions prototypes for the IRQ12 firmware library
- BlueNRG1_irq13.h This file contains all the functions prototypes for the IRQ13 firmware library
- BlueNRG1_irq14.h This file contains all the functions prototypes for the IRQ14 firmware library
- BlueNRG1_irq15.h This file contains all the functions prototypes for the IRQ15 firmware library
- BlueNRG1_irq16.h This file contains all the functions prototypes for the IRQ16 firmware library
- BlueNRG1_irq17.h This file contains all the functions prototypes for the IRQ17 firmware library
- BlueNRG1_irq18.h This file contains all the functions prototypes for the IRQ18 firmware library
- BlueNRG1_irq19.h This file contains all the functions prototypes for the IRQ19 firmware library
- BlueNRG1_irq20.h This file contains all the functions prototypes for the IRQ20 firmware library
- BlueNRG1_irq21.h This file contains all the functions prototypes for the IRQ21 firmware library
- BlueNRG1_irq22.h This file contains all the functions prototypes for the IRQ22 firmware library
- BlueNRG1_irq23.h This file contains all the functions prototypes for the IRQ23 firmware library
- BlueNRG1_irq24.h This file contains all the functions prototypes for the IRQ24 firmware library
- BlueNRG1_irq25.h This file contains all the functions prototypes for the IRQ25 firmware library
- BlueNRG1_irq26.h This file contains all the functions prototypes for the IRQ26 firmware library
- BlueNRG1_irq27.h This file contains all the functions prototypes for the IRQ27 firmware library
- BlueNRG1_irq28.h This file contains all the functions prototypes for the IRQ28 firmware library
- BlueNRG1_irq29.h This file contains all the functions prototypes for the IRQ29 firmware library
- BlueNRG1_irq30.h This file contains all the functions prototypes for the IRQ30 firmware library
- BlueNRG1_irq31.h This file contains all the functions prototypes for the IRQ31 firmware library
- BlueNRG1_irq32.h This file contains all the functions prototypes for the IRQ32 firmware library
- BlueNRG1_irq33.h This file contains all the functions prototypes for the IRQ33 firmware library
- BlueNRG1_irq34.h This file contains all the functions prototypes for the IRQ34 firmware library
- BlueNRG1_irq35.h This file contains all the functions prototypes for the IRQ35 firmware library
- BlueNRG1_irq36.h This file contains all the functions prototypes for the IRQ36 firmware library
- BlueNRG1_irq37.h This file contains all the functions prototypes for the IRQ37 firmware library
- BlueNRG1_irq38.h This file contains all the functions prototypes for the IRQ38 firmware library
- BlueNRG1_irq39.h This file contains all the functions prototypes for the IRQ39 firmware library
- BlueNRG1_irq40.h This file contains all the functions prototypes for the IRQ40 firmware library
- BlueNRG1_irq41.h This file contains all the functions prototypes for the IRQ41 firmware library
- BlueNRG1_irq42.h This file contains all the functions prototypes for the IRQ42 firmware library
- BlueNRG1_irq43.h This file contains all the functions prototypes for the IRQ43 firmware library
- BlueNRG1_irq44.h This file contains all the functions prototypes for the IRQ44 firmware library
- BlueNRG1_irq45.h This file contains all the functions prototypes for the IRQ45 firmware library
- BlueNRG1_irq46.h This file contains all the functions prototypes for the IRQ46 firmware library
- BlueNRG1_irq47.h This file contains all the functions prototypes for the IRQ47 firmware library
- BlueNRG1_irq48.h This file contains all the functions prototypes for the IRQ48 firmware library
- BlueNRG1_irq49.h This file contains all the functions prototypes for the IRQ49 firmware library
- BlueNRG1_irq50.h This file contains all the functions prototypes for the IRQ50 firmware library
- BlueNRG1_irq51.h This file contains all the functions prototypes for the IRQ51 firmware library
- BlueNRG1_irq52.h This file contains all the functions prototypes for the IRQ52 firmware library
- BlueNRG1_irq53.h This file contains all the functions prototypes for the IRQ53 firmware library
- BlueNRG1_irq54.h This file contains all the functions prototypes for the IRQ54 firmware library
- BlueNRG1_irq55.h This file contains all the functions prototypes for the IRQ55 firmware library
- BlueNRG1_irq56.h This file contains all the functions prototypes for the IRQ56 firmware library
- BlueNRG1_irq57.h This file contains all the functions prototypes for the IRQ57 firmware library
- BlueNRG1_irq58.h This file contains all the functions prototypes for the IRQ58 firmware library
- BlueNRG1_irq59.h This file contains all the functions prototypes for the IRQ59 firmware library
- BlueNRG1_irq60.h This file contains all the functions prototypes for the IRQ60 firmware library
- BlueNRG1_irq61.h This file contains all the functions prototypes for the IRQ61 firmware library
- BlueNRG1_irq62.h This file contains all the functions prototypes for the IRQ62 firmware library
- BlueNRG1_irq63.h This file contains all the functions prototypes for the IRQ63 firmware library
- BlueNRG1_irq64.h This file contains all the functions prototypes for the IRQ64 firmware library
- BlueNRG1_irq65.h This file contains all the functions prototypes for the IRQ65 firmware library
- BlueNRG1_irq66.h This file contains all the functions prototypes for the IRQ66 firmware library
- BlueNRG1_irq67.h This file contains all the functions prototypes for the IRQ67 firmware library
- BlueNRG1_irq68.h This file contains all the functions prototypes for the IRQ68 firmware library
- BlueNRG1_irq69.h This file contains all the functions prototypes for the IRQ69 firmware library
- BlueNRG1_irq70.h This file contains all the functions prototypes for the IRQ70 firmware library
- BlueNRG1_irq71.h This file contains all the functions prototypes for the IRQ71 firmware library
- BlueNRG1_irq72.h This file contains all the functions prototypes for the IRQ72 firmware library
- BlueNRG1_irq73.h This file contains all the functions prototypes for the IRQ73 firmware library
- BlueNRG1_irq74.h This file contains all the functions prototypes for the IRQ74 firmware library
- BlueNRG1_irq75.h This file contains all the functions prototypes for the IRQ75 firmware library
- BlueNRG1_irq76.h This file contains all the functions prototypes for the IRQ76 firmware library
- BlueNRG1_irq77.h This file contains all the functions prototypes for the IRQ77 firmware library
- BlueNRG1_irq78.h This file contains all the functions prototypes for the IRQ78 firmware library
- BlueNRG1_irq79.h This file contains all the functions prototypes for the IRQ79 firmware library
- BlueNRG1_irq80.h This file contains all the functions prototypes for the IRQ80 firmware library
- BlueNRG1_irq81.h This file contains all the functions prototypes for the IRQ81 firmware library
- BlueNRG1_irq82.h This file contains all the functions prototypes for the IRQ82 firmware library
- BlueNRG1_irq83.h This file contains all the functions prototypes for the IRQ83 firmware library
- BlueNRG1_irq84.h This file contains all the functions prototypes for the IRQ84 firmware library
- BlueNRG1_irq85.h This file contains all the functions prototypes for the IRQ85 firmware library
- BlueNRG1_irq86.h This file contains all the functions prototypes for the IRQ86 firmware library
- BlueNRG1_irq87.h This file contains all the functions prototypes for the IRQ87 firmware library
- BlueNRG1_irq88.h This file contains all the functions prototypes for the IRQ88 firmware library
- BlueNRG1_irq89.h This file contains all the functions prototypes for the IRQ89 firmware library
- BlueNRG1_irq90.h This file contains all the functions prototypes for the IRQ90 firmware library
- BlueNRG1_irq91.h This file contains all the functions prototypes for the IRQ91 firmware library
- BlueNRG1_irq92.h This file contains all the functions prototypes for the IRQ92 firmware library
- BlueNRG1_irq93.h This file contains all the functions prototypes for the IRQ93 firmware library
- BlueNRG1_irq94.h This file contains all the functions prototypes for the IRQ94 firmware library
- BlueNRG1_irq95.h This file contains all the functions prototypes for the IRQ95 firmware library
- BlueNRG1_irq96.h This file contains all the functions prototypes for the IRQ96 firmware library
- BlueNRG1_irq97.h This file contains all the functions prototypes for the IRQ97 firmware library
- BlueNRG1_irq98.h This file contains all the functions prototypes for the IRQ98 firmware library
- BlueNRG1_irq99.h This file contains all the functions prototypes for the IRQ99 firmware library
- BlueNRG1_irq100.h This file contains all the functions prototypes for the IRQ100 firmware library

BlueNRG-1 ACI APIs and events callba

life.augmented

Main Page | Modules | Classes | Files

File List

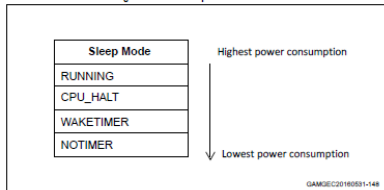
Here is a list of all files with brief descriptions:

- ble_status.h Header file for BLE stack status codes
- bluenrg1_api.h Header file for BlueNRG-1 Bluetooth Low Energy stack APIs
- bluenrg1_events.h Header file for BlueNRG-1 Bluetooth Low Energy stack events callbacks
- bluenrg1_gap.h Header file for BlueNRG-1 GAP layer constants
- bluenrg1_gap_server.h Header file for BlueNRG-1 GATT server layer constants
- bluenrg1_hal.h Header file with HAL define for BlueNRG-1
- bluenrg1_stack.h Header file for BlueNRG-1 BLE stack initialization and sleep timer
- hal_constants.h This file defines constants and functions for HCI layer
- link_layer.h Header file for BlueNRG-1's link layer constants
- sm.h Header file for BlueNRG-1's security manager constants

BlueNRG-1 Development Tools

SW material- Documentation

SW DK - Documentation

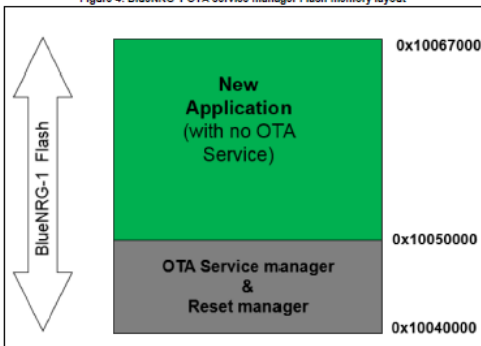


AN4820 : BlueNRG-1 Low Power modes

2.3 OTA Service manager framework

A simpler approach coming from the BLE OTA service architecture described in [Figure 3 BlueNRG-1 Lower and Higher applications with OTA service](#), consists of using a basic OTA service manager application which only includes the BLE OTA service & characteristics and the OTA Reset Manager capabilities.

Figure 4: BlueNRG-1 OTA service manager Flash memory layout



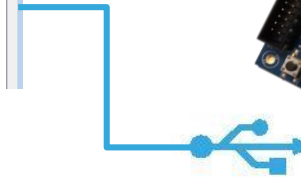
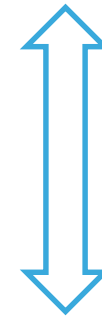
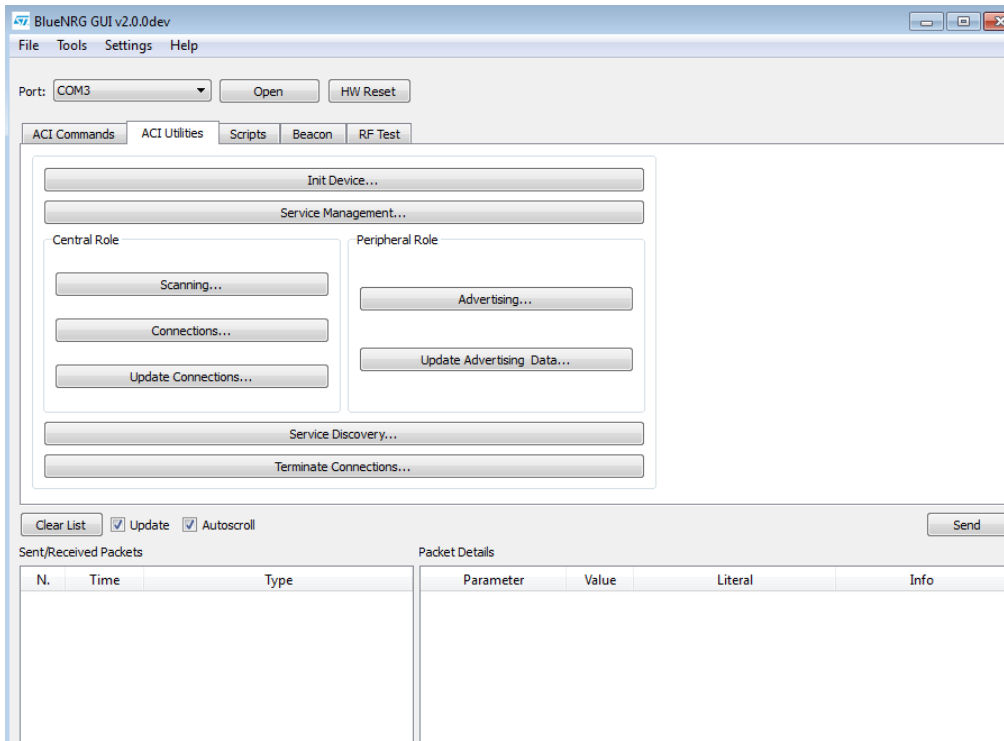
AN4869 : BlueNRG-1 Firmware Upgrade Over the Air

BlueNRG-1 Development Tools

Tool material

ST BLE GUI

@ STSW-BNRGUI



Comprehensive GUI to understand BLE concept and associated ST APIs

BlueNRG-1

BLE SOC presentation

BlueNRG-1 BLE SOC presentation

BlueNRG-1 Power consumption figures

BlueNRG-1 Development Tools

BlueNRG-1 SW API to ease your design

How to promote - Navigator tool

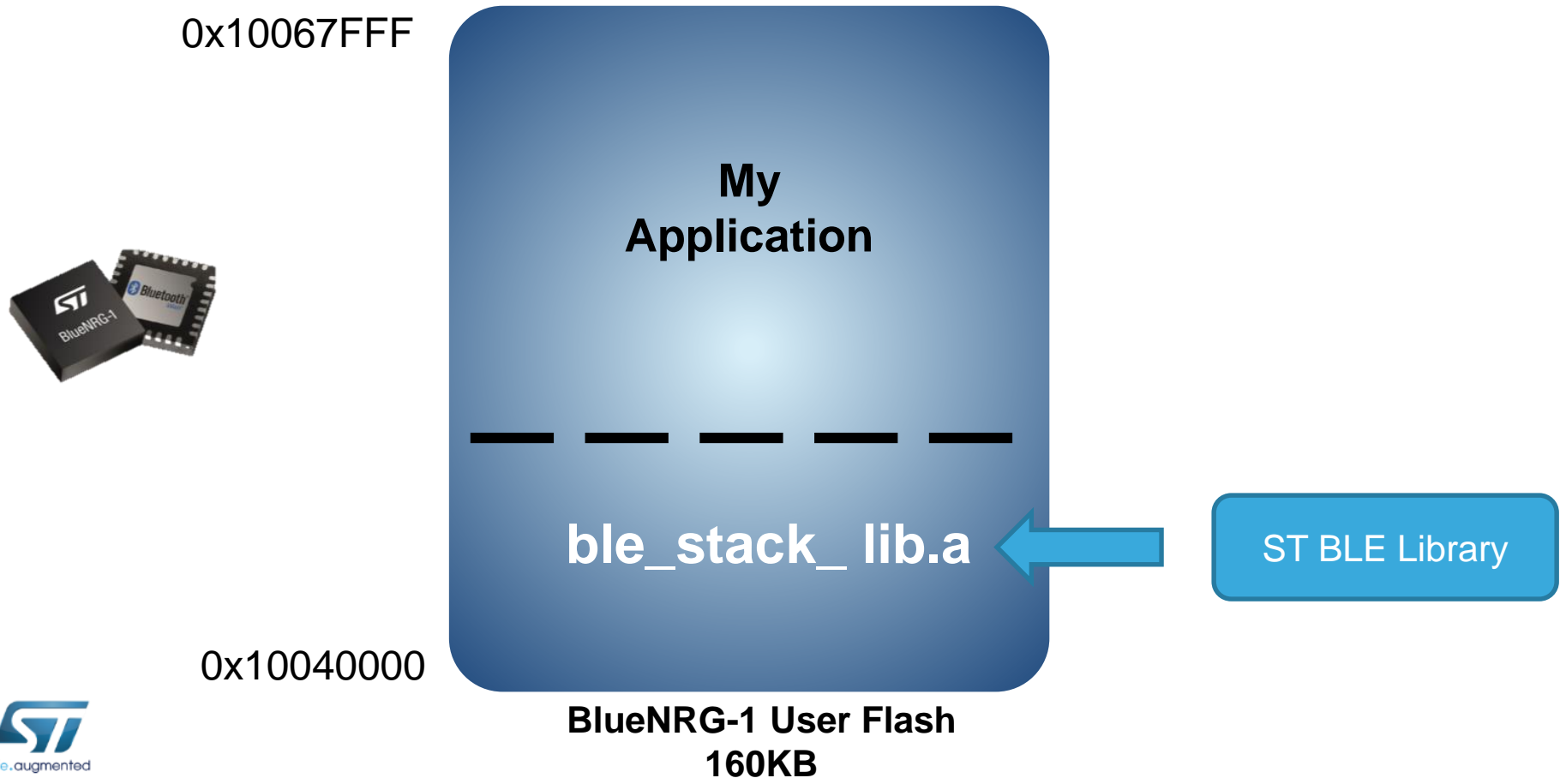




BlueNRG-1 Development Tools

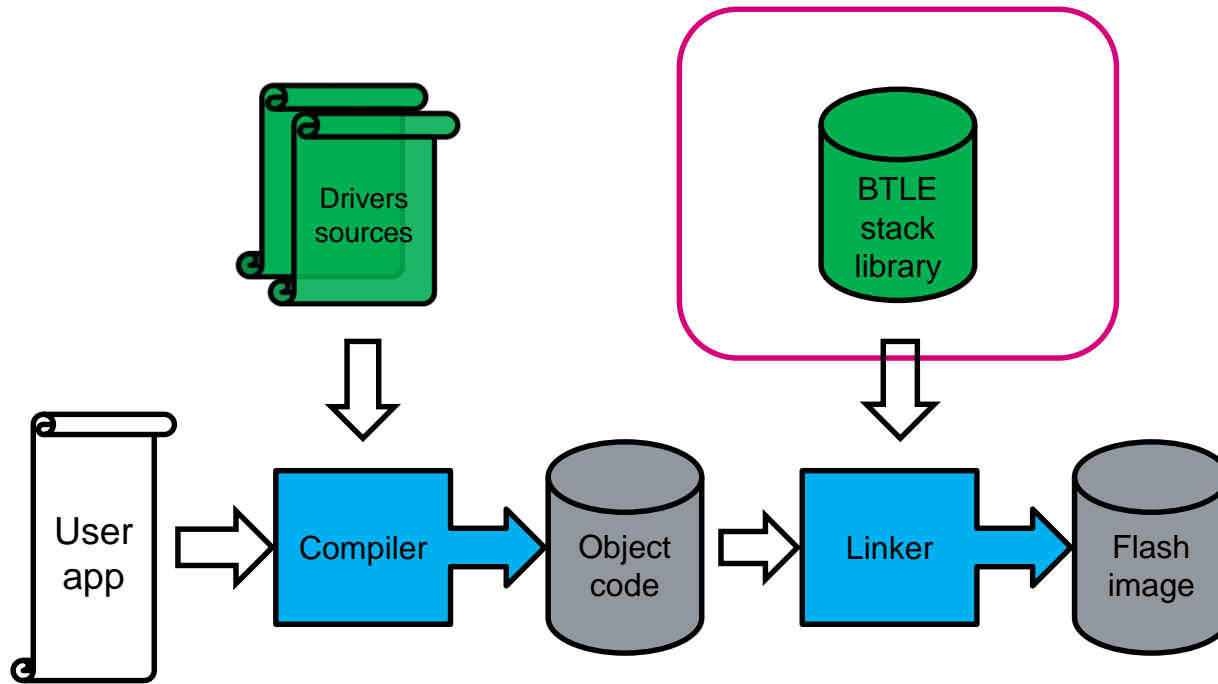
SOC solution = SW leverage (1/3)

BlueNRG-1 User Flash – 160KB



BlueNRG-1 Development Tools



SOC solution = SW leverage (2/3)



 atollic

 IAR
SYSTEMS

 KEIL™
Tools by ARM

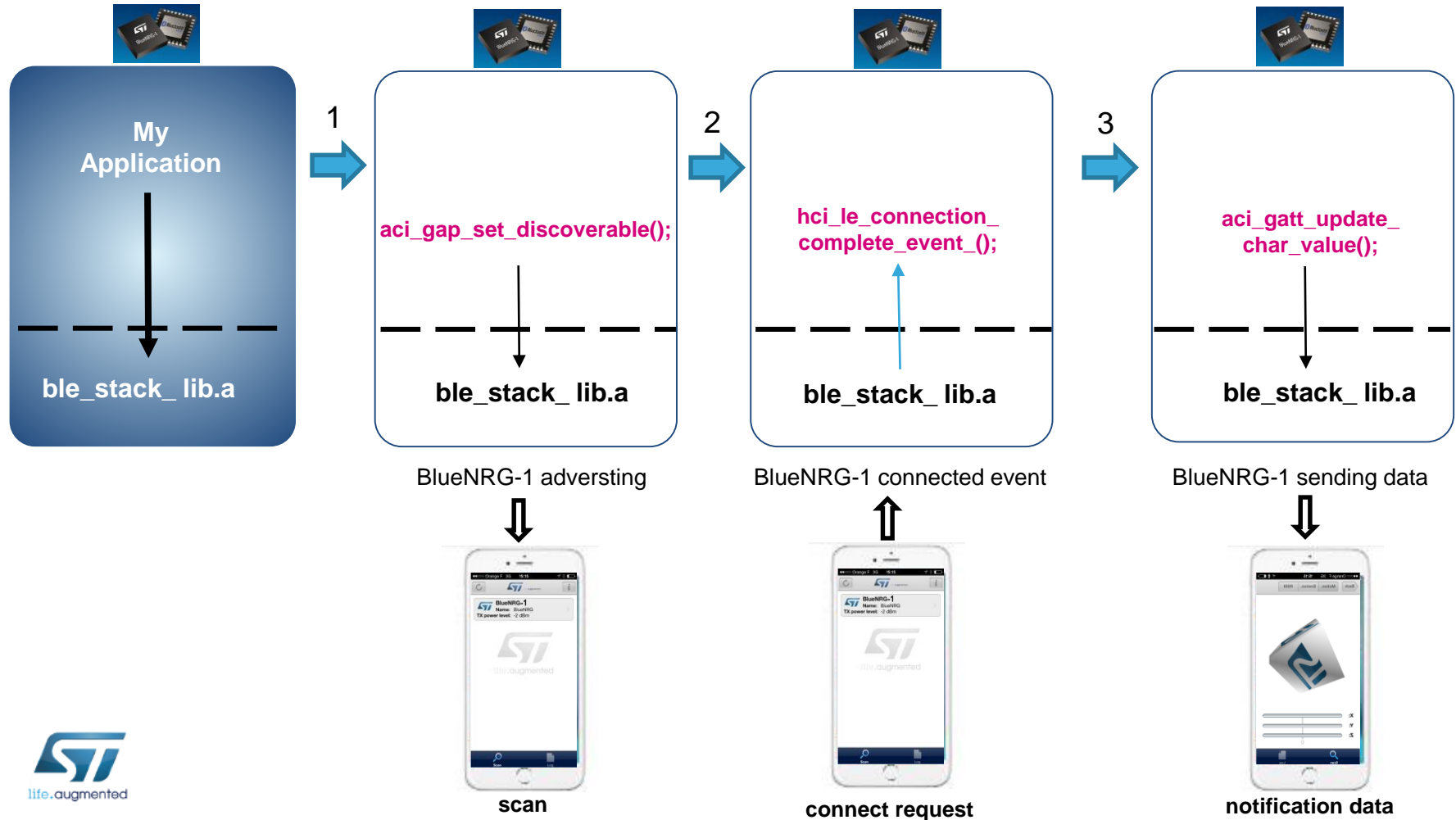
-  Provided by ST
-  Provided by third party



BlueNRG-1 Development Tools

SOC solution = SW leverage (3/3)

High-level abstraction layer APIs



BlueNRG-1 DK to ease your SW design

SW architecture takeaways

```
void main(void)
{
    uint8_t ret;

    /* System Init */
    SystemInit();

    /* Identify BlueNRG-1 platform */
    SdkEvalIdentification();

    /* Init the UART peripheral */
    SdkEvalComUartInit(UART_BAUDRATE);

    /* BlueNRG-1 stack init */
    ret = BlueNRG_Stack_Initialization(&BlueNRG_Stack_Init_params);
    if (ret != BLE_STATUS_SUCCESS) {
        printf("Error in BlueNRG_Stack_Initialization() 0x%02x\r\n", ret);
        while(1);
    }

    /* Init the BlueNRG-1 device */
    Device_Init();

    /* Start Beacon Non Connectable Mode*/
    Start_Beaconing();

    printf("BlueNRG-1 BLE Beacon Application (version: %s)\r\n", BLE_BEACON_VERSION_STRING);

    while(1)
    {
        /* BlueNRG-1 stack tick */
        BTLE_StackTick();

        /* Enable Power Save according the Advertising Interval */
        BlueNRG_Sleep(SLEEPMODE_WAKETIMER, 0, 0, 0);
    }
}
```



1 peripheral and stack on same core
=
simply application design

2 peripherals interface = "STM32 like"

stack and application over same flash

=
4 simplify FW upgrade procedure
&
Forget IFR process (if you know...)

BlueNRG-1

BLE SOC presentation



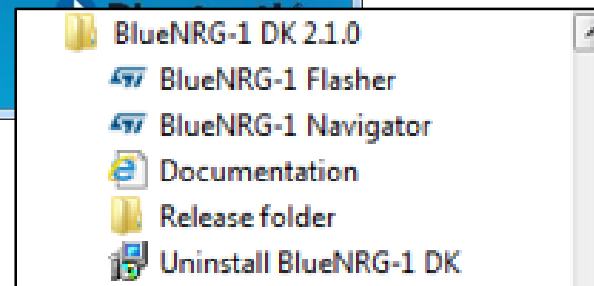
BlueNRG-1 BLE SOC presentation

BlueNRG-1 Power consumption figures

BlueNRG-1 Development Tools

BlueNRG-1 SW API to ease your design

How to promote - Navigator tool



download and run the selected application prebuilt binary image into the BlueNRG-1 platform

access to the demo description, board configuration and to the source code

BlueNRG-1 Navigator v.x.x.x

BLE Beacon

Hover the mouse pointer to highlight the item on the board

RESET - Reset BlueNRG1

This is a BLE beacon demo that shows how to configure a BlueNRG-1 device in order to advertise specific manufacturing data and allow another BLE device to know if it is in the range of the BlueNRG-1 beacon device. It also provides a reference example about how using the BLE Over-The-Air (OTA) Service Manager firmware upgrade capability.

Usage

The Beacon demo configures a BlueNRG-1 device in advertising mode (non-connectable mode) with specific manufacturing data. It transmits advertisement packets at regular intervals which contain the following manufacturing data:

flashing 38%

Board plugged: COM5

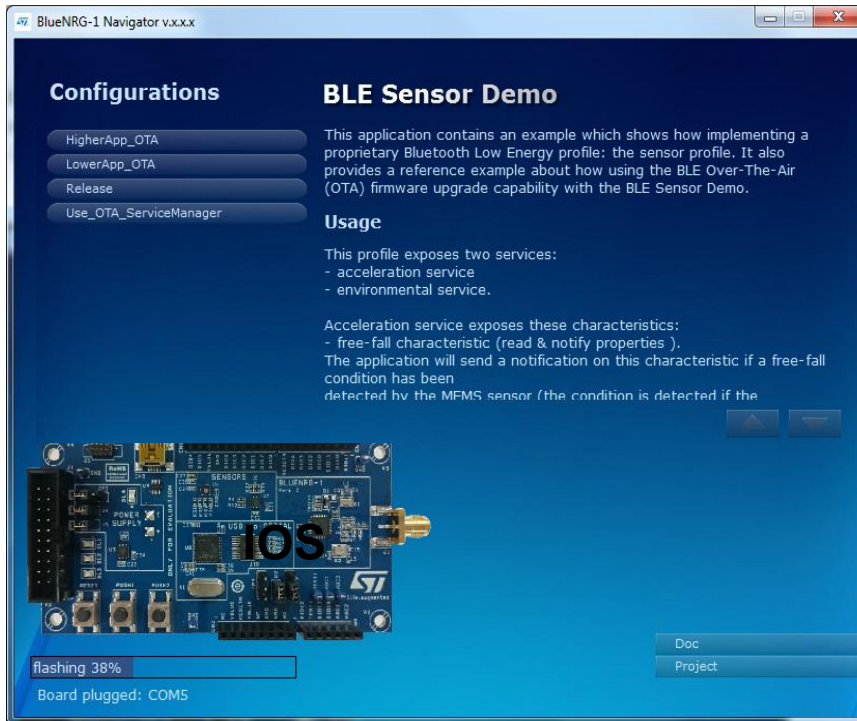
Doc
Project

BlueNRG-1

BlueNRG-1 Navigator (3/3)

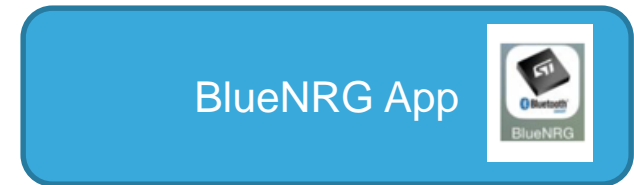
Sensor Demo

1



**Discoverable as
BlueNRG**

2



BlueNRG App
≠
Blue MS (BlueMicrosystem
over Nucleo)

BlueNRG-1

BlueNRG-1 Navigator (3/3) Remote Control Demo



1

BlueNRG-1 Navigator v.2.2.0

BLE Remote Control

Hover the mouse pointer to highlight the item on the board

Temperature is given in tenths of Celsius degree.

DL1 - Led under remote control
DL2 - Led under remote control
DL3 - Led under remote control
RESET - Reset BlueNRG1

To use this example, build and download the firmware in one of the supported boards.
Use a BLE device (another BlueNRG-1 or any SMART READY device), to scan and see broadcast data.
To control one of the LEDs, just connect to BlueNRG and write into the exposed characteristic.
The characteristic is identified with a 128bit UUID. Service UUID is ed0ef62e-9b0d-11e4-89d3-123b93f75cba.
The characteristic UUID is ed0efb1a-9b0d-11e4-89d3-123b93f75cba.

Flash & Run
Doc
Project

Board selected: COM16



**Discoverable as
Node**

2



LightBlue
Apple Store



BLE Scanner
Google store

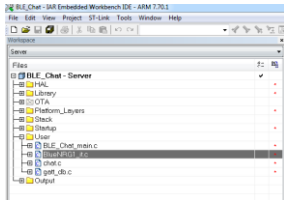
Free debug App on google
and apple store

From smartphone write
0x04 , 0x05 ..to change
LED status

BlueNRG-1 Takeways



Optimized for ultra-low-power
Beacon mode : 16uA @ 1.28s



BlueNRG-1 Powerful Development Kit
Navigator : Promotion Tool
ST GUI : Comprehensive BLE concepts
BlueNRG-1 various code examples



BlueNRG-1 on the web
Dedicated and specific documentation on demand
@ rf-support-emea@st.com

BlueNRG-1 BLE SOC

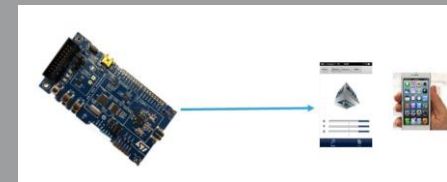
BlueNRG-1 and associated development (& promotion) package presentation



BLE concept demystification

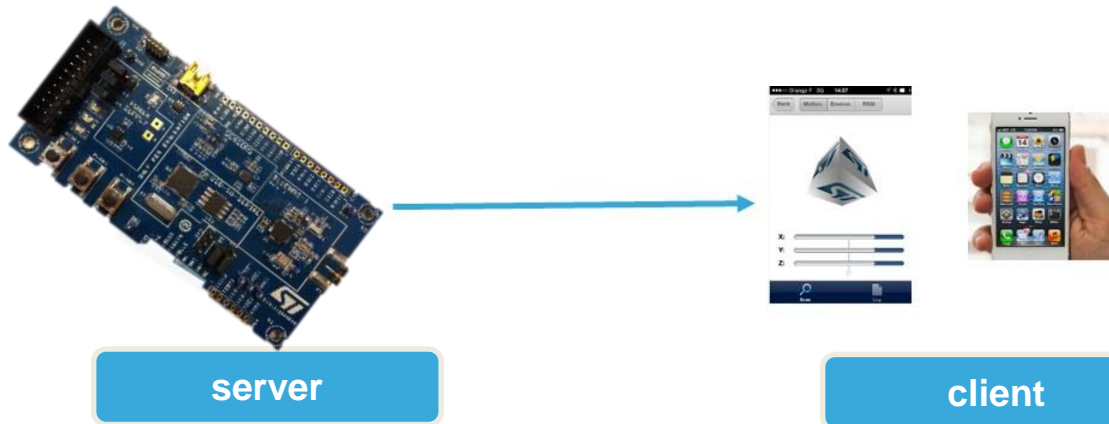


Enable BLE link over BlueNRG-1



BLE Concepts

Definitions



- **2 entities in a BLE communication**

1. **The server** : exposing data (temperature, position, raw data, **what you want !**)
2. **The client** : connecting to server and looking for data

- **A BLE application is based on an application profile**

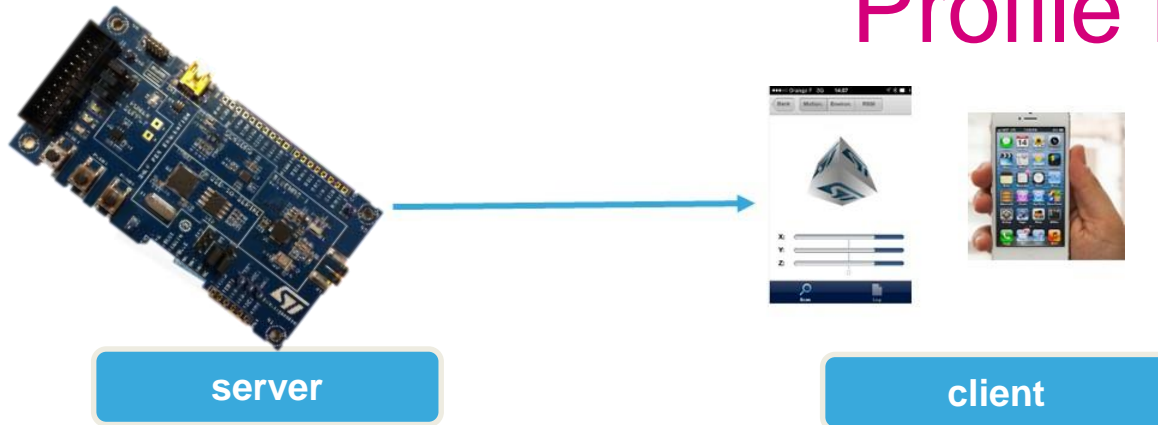
- standard : glucose meter, Heart Rate Monitor, Find me
- Proprietary : sensor profile, chat profile , **my custom profile !**

- **A profile is a basic collection of data exposed by the device**

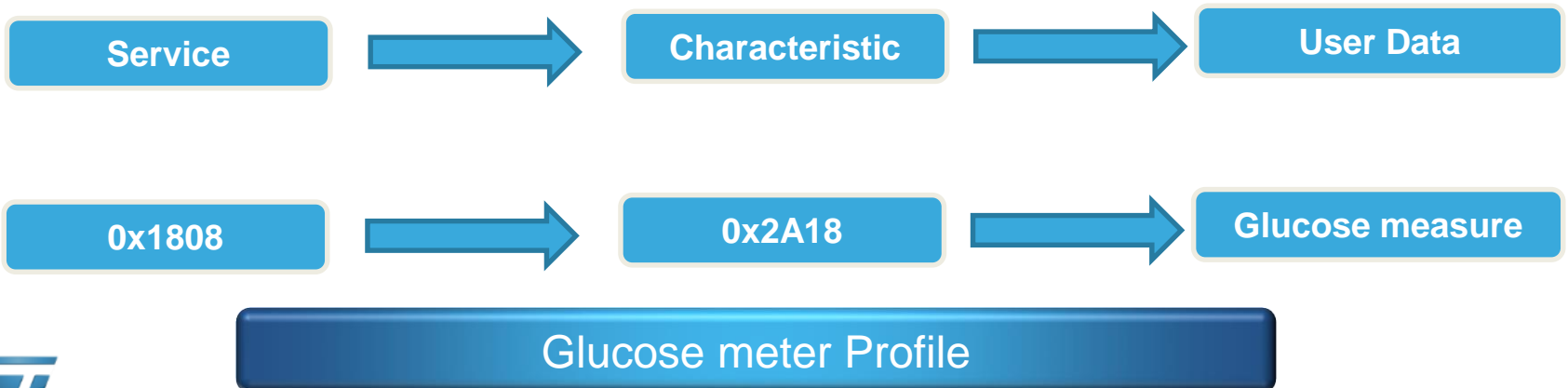
- service : **a basic UUID** (0x1808 = Glucose meter Service)
- characteristic : **basic UUID** and associated **data exposed** (MEMS,ect...)

BLE Concepts

Profile Definition



- **A profile is a basic collection of data exposed by the device**
 - service : **a basic UUID** (0x1808 = Glucose meter Service)
 - characteristic : **basic UUID** and associated data you are willing to expose (MEMS)

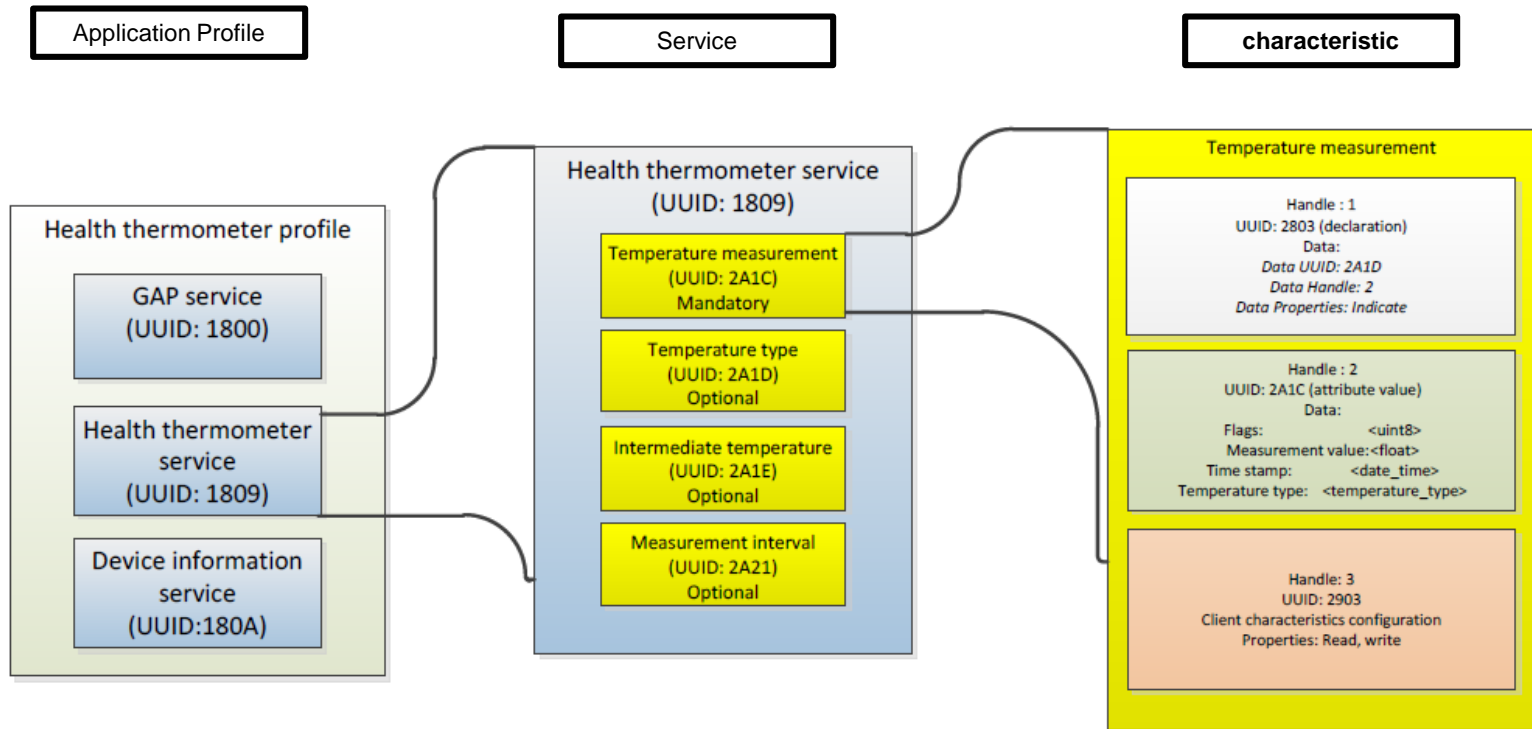


BLE Concepts

Profile Definition



server



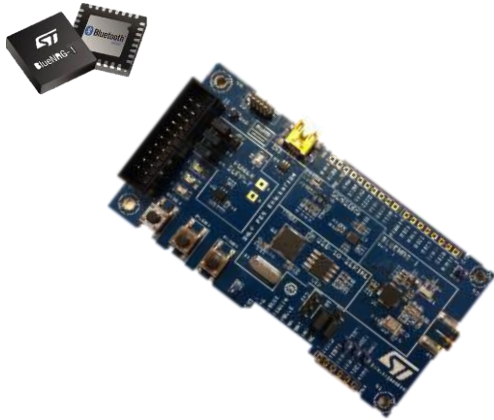
- Standard services & characteristics specification & UUID assignment available:
 - <https://developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx>
 - <https://developer.bluetooth.org/gatt/characteristics/Pages/CharacteristicsHome.aspx>



BLE Concepts

Attribut Table

Sensor demo Application Profile
service = Sensor Service
characteristic = Sensor characterisic



server



client

BlueNRG-1
Attribut Table

BLE Concepts

Attribut Table



server

BlueNRG-1
Attribut table

1 The sensor proprietary profile at initialisation will add in BlueNRG-1RAM an entry (attribut table) to expose application data (MEMS)

@1	@2	@3
Sensor Service UUID	Sensor Char UUID	MEMS Data



client

3 Application MEMS update will update the attribut table with the new characteristic value

		MEMS Data

2 As soon as connected client will be able to access (write/read) to attribute table thanks to BLE specification

BlueNRG-1 BLE SOC

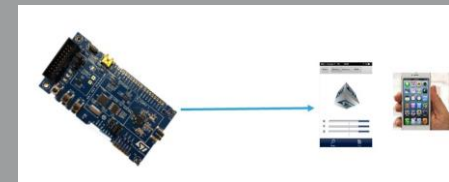
BlueNRG-1 and associated development (& promotion) package presentation

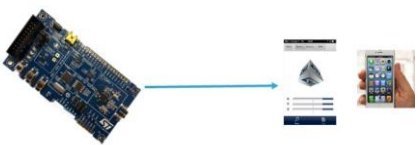


BLE concept demystification



Enable BLE link over BlueNRG-1

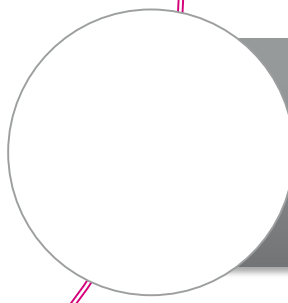




Enable BLE link over BlueNRG-1



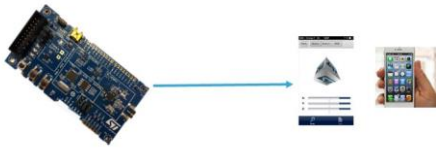
BlueNRG-1 GUI Hands On



SW code implementation

BlueNRG-1 – GUI Hands On

Prerequisites (2/2)



Free BLE debug application

- Required to perform GUI basic hands on to enable connection



LightBlue
Apple Store

IOS



BLE Scanner
Google store

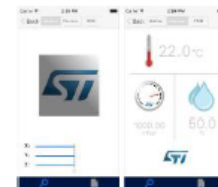
Android

Sensor debug application

- Required to execute and perform the sensor demo part of BlueNRG-1 DK binaries



Keyword: BlueNRG
Apple Store & Google store

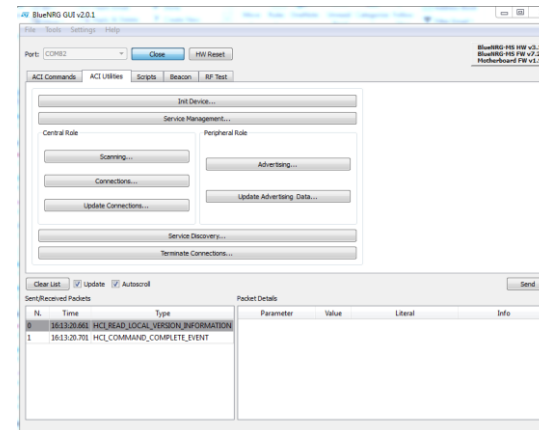


ST BLE GUI - Hands On

- 1 from Navigator load DTM 16Mhz UART application on BlueNRG-1 @ [STSW-BLUENRG1-DK](#)

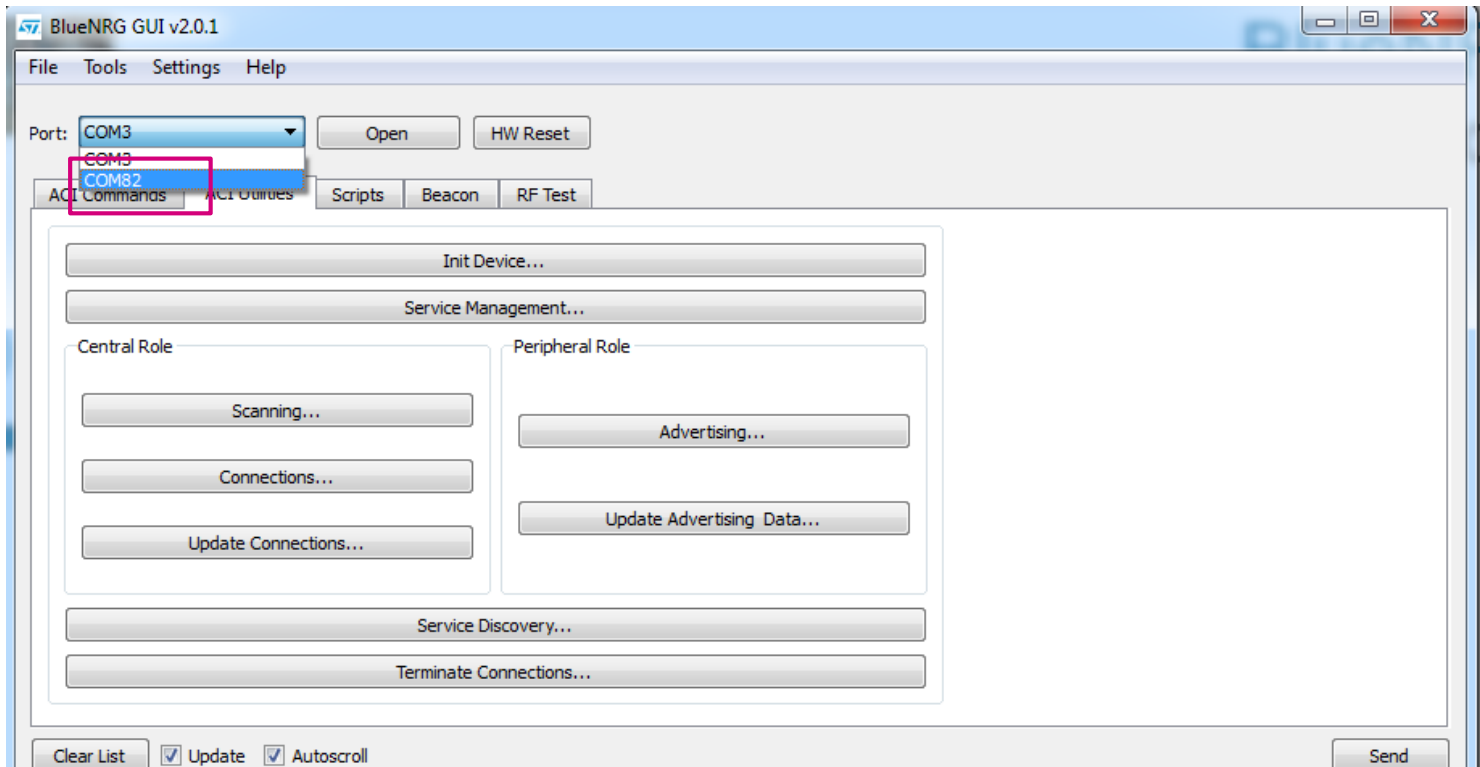


- 2 open ST BlueNRG-1 GUI @ [STSW-BNRGUI](#)



ST BLE GUI - Hands On

3

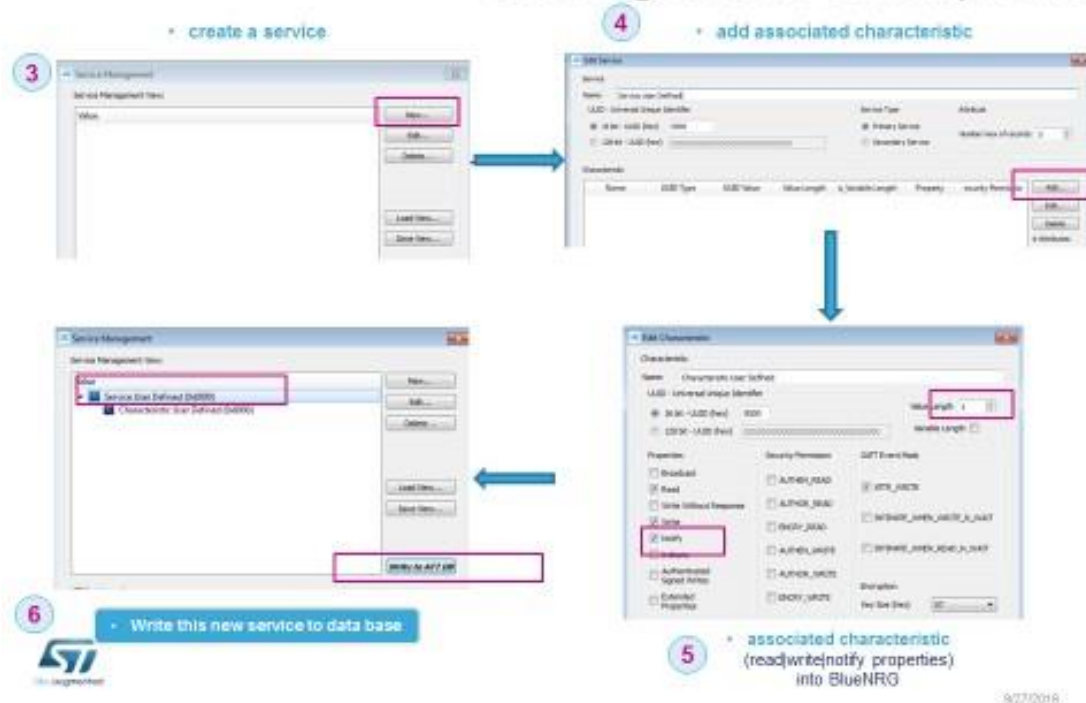


**GUI now controlling
BlueNRG-1 SOC**

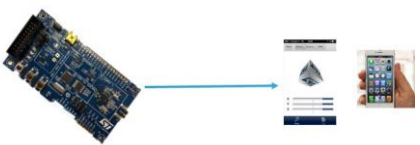
ST BLE GUI - Hands On



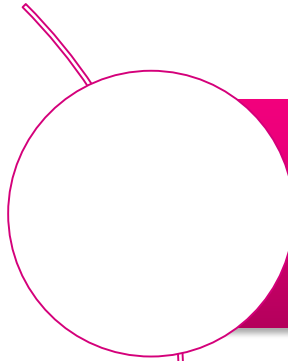
BlueNRG-1 Sharing data to smartphone



@ BlueNRG-1 Hands On IDB007V1 V3.0.pdf



Enable BLE link over BlueNRG-1



BlueNRG-1 GUI Hands On



SW code implementation

Application structure

```
/* System configuration */
SystemInit();

/* BlueNRG-1 stack init*/
BlueNRG_Stack_Initialization(&BlueNRG_Stack_Init_params
);

while(1) {

/* BLE state machine */
BTLE_StackTick();

/* Application state machine */
APP_Tick();

/* Power Save management */
BlueNRG_Sleep(SLEEPMODE_WAKETIMER, 0, 0, 0);

}/* while (1) */
```

Classical embedded system structure

- Two state machines for BLE stack and Application

- Advanced power management controlled by BLE stack and Applications

Application structure

```
/* System configuration */
SystemInit();

/* BlueNRG-1 stack init*/
BlueNRG_Stack_Initialization(&BlueNRG_Stack_Init_params
);

while(1) {

/* BLE state machine */
BTLE_StackTick();

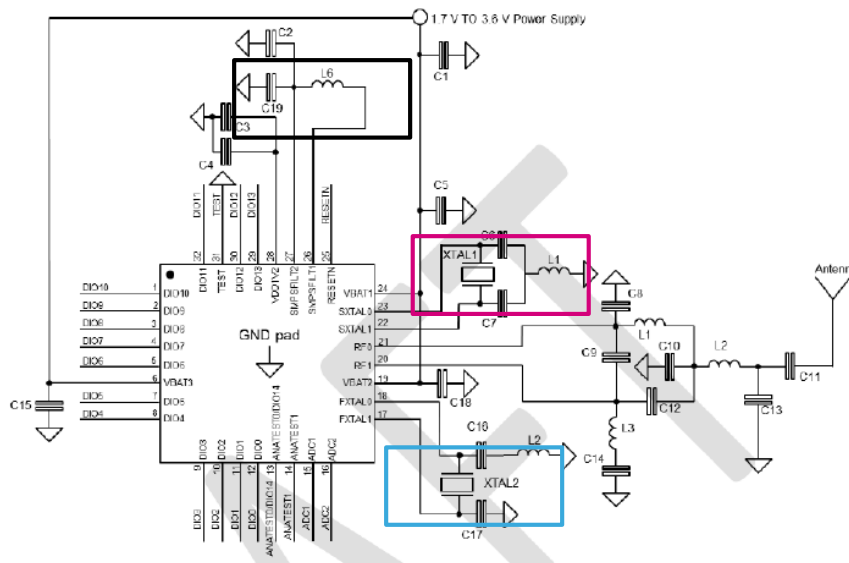
/* Application state machine */
APP_Tick();

/* Power Save management */
BlueNRG_Sleep(SLEEPMODE_WAKETIMER, 0, 0, 0);

}/* while (1) */
```

```
/* System configuration */
SystemInit();
```

BlueNRG-1 device configuration parameters: Project Preprocessor options



HS_SPEED_XTAL

- HS_SPEED_XTAL_16MHZ /* High Speed crystal 16 MHz (Default) */
- HS_SPEED_XTAL_32MHZ /* High Speed crystal 32 MHz */
- HS_SPEED_XTAL_INTERNAL_RO /* High Speed Internal RO. Not useful when radio operations are needed or in any case when accurate ref clock is needed */

LS_SOURCE

- LS_SOURCE_INTERNAL_RO /* Low Speed Internal RO */
- LS_SOURCE_EXTERNAL_32KHZ /* Low Speed External 32 KHz (Default) */

SMPS_INDUCTOR

- SMPS_INDUCTOR_10uH /* SMPS Inductor 10 uH(Default) */
- SMPS_INDUCTOR_4_7uH /* SMPS Inductor 4.7 uH*/
- SMPS_INDUCTOR_NONE /* SMPS Inductor None */

Application structure

```
/* System configuration */
SystemInit();

/* BlueNRG-1 stack init*/
BlueNRG_Stack_Initialization(&BlueNRG_Stack_Init_params
);

while(1) {

/* BLE state machine */
BTLE_StackTick();

/* Application state machine */
APP_Tick();

/* Power Save management */
BlueNRG_Sleep(SLEEPMODE_WAKETIMER, 0, 0, 0);

}/* while (1) */
```

```
/* BlueNRG-1 stack init*/  
BlueNRG_Stack_Initialization(&BlueNRG_Stack_Init_params);
```

On file *SensorDemo_config.h*:

```
•const BlueNRG_Stack_Initialization_tBlueNRG_Stack_Init_params= {  
(uint8_t*)stacklib_flash_data,  
FLASH_SEC_DB_SIZE  
  
FLASH_SERVER_DB_SIZE  
  
(uint8_t*)stacklib_stored_device_id_data,  
  
(uint8_t*)dyn_alloc_a,  
  
NUM_GATT_ATTRIBUTES,  
  
NUM_GATT_SERVICES,  
  
ATT_VALUE_ARRAY_SIZE  
  
NUM_LINKS,  
  
CONFIG_TABLE};
```



```
/* BlueNRG-1 stack init*/
```

```
BlueNRG_Stack_Initialization(&BlueNRG_Stack_Init_params);
```

```
FLASH_SEC_DB_SIZE, // Flash Security DB Size: 1024bytes (FLASH_SEC_DB_SIZE)
```

```
FLASH_SERVER_DB_SIZE, // Flash Server DB Size: 1024bytes (FLASH_SERVER_DB_SIZE)
```

```
(uint8_t*)stacklib_stored_device_id_data,
```

```
(uint8_t*)dyn_alloc_a,
```

```
NUM_GATT_ATTRIBUTES,
```

```
NUM_GATT_SERVICES,
```

```
ATT_VALUE_ARRAY_SIZE, // ATT_VALUE_ARRAY_SIZE (1344)
```

```
NUM_LINKS,
```

```
CONFIG_TABLE};
```

```
/* BlueNRG-1 stack init*/
```

```
BlueNRG_Stack_Initialization(&BlueNRG_Stack_Init_params);
```

NUM_GATT_ATTRIBUTES

Number of attributes = **9 + 15** (NUM_GATT_ATTRIBUTES)

- The BlueNRG-1 stack uses **9 default attributes**
- The sensor demo application needs of attributes:
 - Free fall characteristic = **3 attributes (declaration, value, client characteristic configuration descriptor)**
 - Accelerometer characteristic = **3 attributes (declaration, value, client characteristic configuration descriptor)**
 - Temperature characteristic = **3 attributes (declaration, value, characteristic format descriptor)**
 - Pressure characteristic = **3 attributes (declaration, value, characteristic format descriptor)**
 - Humidity characteristic = **3 attributes (declaration, value, characteristic format descriptor)**

```
/* BlueNRG-1 stack init*/  
BlueNRG_Stack_Initialization(&BlueNRG_Stack_Init_params);
```

NUM_GATT_SERVICES

Number of services = 2 + 2 (NUM_GATT_SERVICES)

- The BlueNRG-1 stack uses two default services GATT and GAP
- The sensor demo application needs of two services: accelerometer and environmental sensor

Application structure

```
/* System configuration */
SystemInit();

/* BlueNRG-1 stack init*/
BlueNRG_Stack_Initialization(&BlueNRG_Stack_Init_params
);

while(1) {

/* BLE state machine */
BTLE_StackTick();

/* Application state machine */
APP_Tick();

/* Power Save management */
BlueNRG_Sleep(SLEEPMODE_WAKETIMER, 0, 0, 0);

}/* while (1) */
```



```
/* BLE state machine */  
BTLE_StackTick();
```

If there are BLE stack activities ongoing, user application is requested to call it:

- Timers state machine
- Link layer TX/RX state machine
- GAP procedures state machine
- GATT procedures state machine
- Security manager state machine
- Perform crystal calibration of low speed clock (internal or external).

BlueNRG-1 BLE Stack events callbacks

Application structure

```
/* System configuration */
SystemInit();

/* BlueNRG-1 stack init*/
BlueNRG_Stack_Initialization(&BlueNRG_Stack_Init_params
);

while(1) {

/* BLE state machine */
BTLE_StackTick();

/* Application state machine */
APP_Tick();

/* Power Save management */
BlueNRG_Sleep(SLEEPMODE_WAKETIMER, 0, 0, 0);

}/* while (1) */
```

Application structure

```
/* System configuration */
SystemInit();

/* BlueNRG-1 stack init*/
BlueNRG_Stack_Initialization(&BlueNRG_Stack_Init_params
);

while(1) {

/* BLE state machine */
BTLE_StackTick();

/* Application state machine */
APP_Tick();

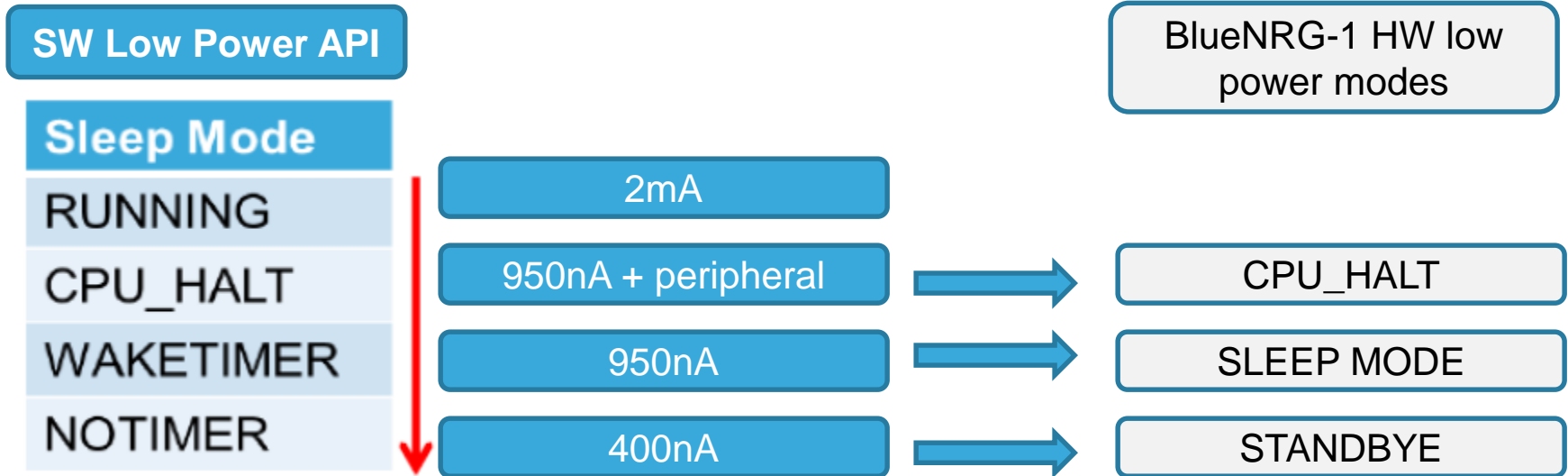
/* Power Save management */
BlueNRG_Sleep(SLEEPMODE_WAKETIMER, 0, 0, 0);

}/* while (1) */
```

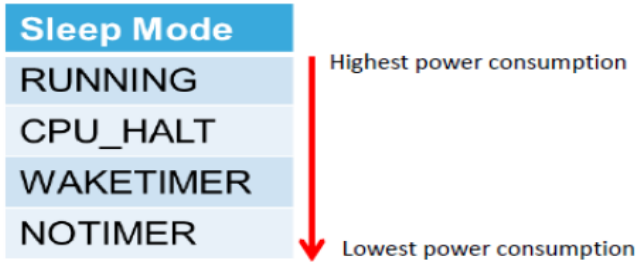


BlueNRG-1

Sleep Mode management



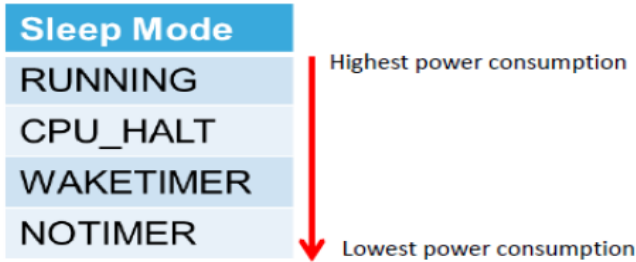
@ Please refer to AN4820




CPU_HALT

- In this mode only the CPU is stopped.
- All peripherals continue to operate and can wake up the CPU.
- The wakeup criteria are external interrupts or events (timers, RF timers).
- This is the lowest power save mode

950nA + peripheral

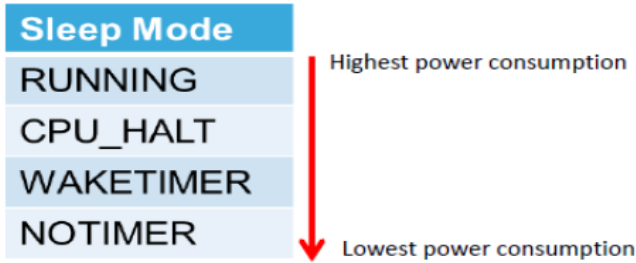


Sleep Mode

- In this mode the CPU is stopped and all the peripherals are disabled.
- Only the low speed oscillator block and the external wakeup sources block are running.
- The wakeup source will be:
 - Wakeup timer 
 - IO9, IO10, IO11, IO12 and IO13
- When a wakeup is triggered by a previous listed source, the system reverts to the run mode with all the peripherals on.

RF timer (Adv or Con events)
+ 4 Virtual Timers (application)
managed by stack

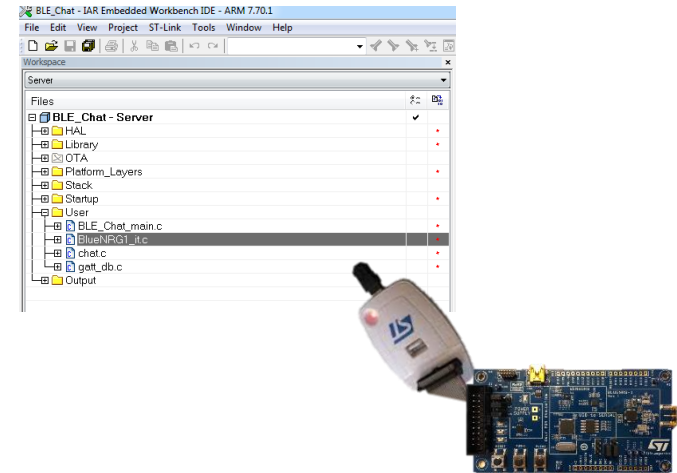
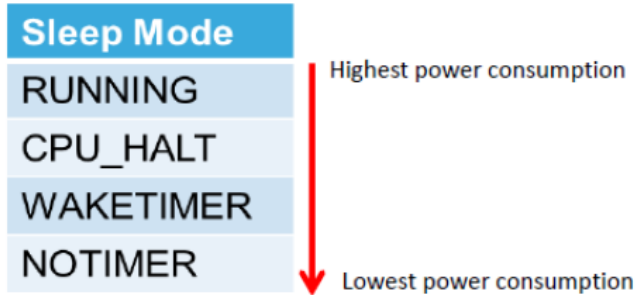
950nA



Standby Mode

- In this mode the CPU is stopped and all the peripherals are disabled.
- The only wakeup sources are:
 - IO9, IO10, IO11, IO12 and IO13
- This mode is the highest power save mode.

Sleep Mode management vs. debug constraints



SOC enabling RF and application brings real time constraints and so debug capabilities are limited

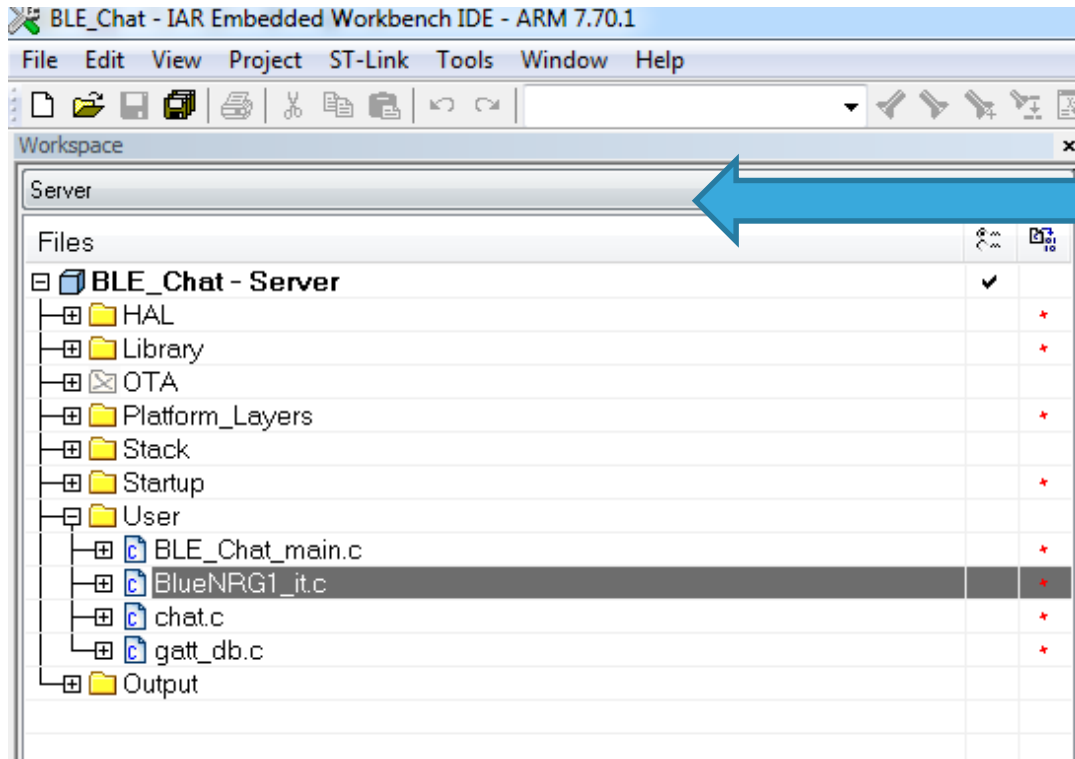


To ease debug we recommend to disable sleep
`// BlueNRG_Sleep(SLEEPMODE_WAKETIMER, 0, 0, 0);`

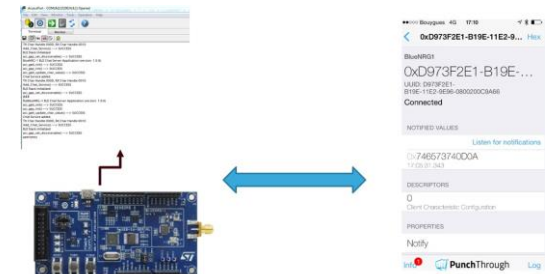
Enable SPP like connection- Chat Demo

The Chat demo described in BlueNRG-1 User Manual section 7 [UM2071](#)

1 Open Chat application from code example



Select Server workspace.
The server is the device
exposing service and
characteristic (TX and RX)



Enable SPP like connection- Chat Demo



② modify device name and BLE MAC @

In file chat.c

```
uint8_t CHAT_DeviceInit(void)
{
    uint8_t ret;
    uint16_t service_handle;
    uint16_t dev_name_char_handle;
    uint16_t appearance_char_handle;
    uint8_t name[] = {'B', 'l', 'u', 'e', 'N', 'R', 'G', '0 '}; // from 0 to 10 //

#ifdef SERVER
    uint8_t role = GAP_PERIPHERAL_ROLE;
    uint8_t bdaddr[] = {0xaa, 0x00, 0x00, 0xE1, 0x80, 0x00}; // from 0 to 10 //
#else
    ...
    ...

ret = aci_gap_init(role, 0x00, 0x08, &service_handle, &dev_name_char_handle, &appearance_char_handle);
// change device name length from 0x07 to 0x08 //
```

Enable SPP like connection- Chat Demo

3 modify local name

In file chat.c

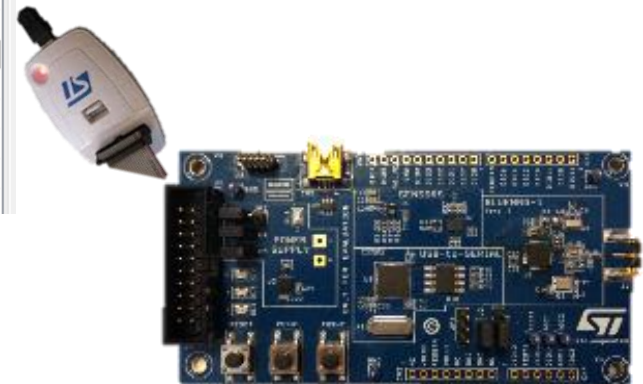
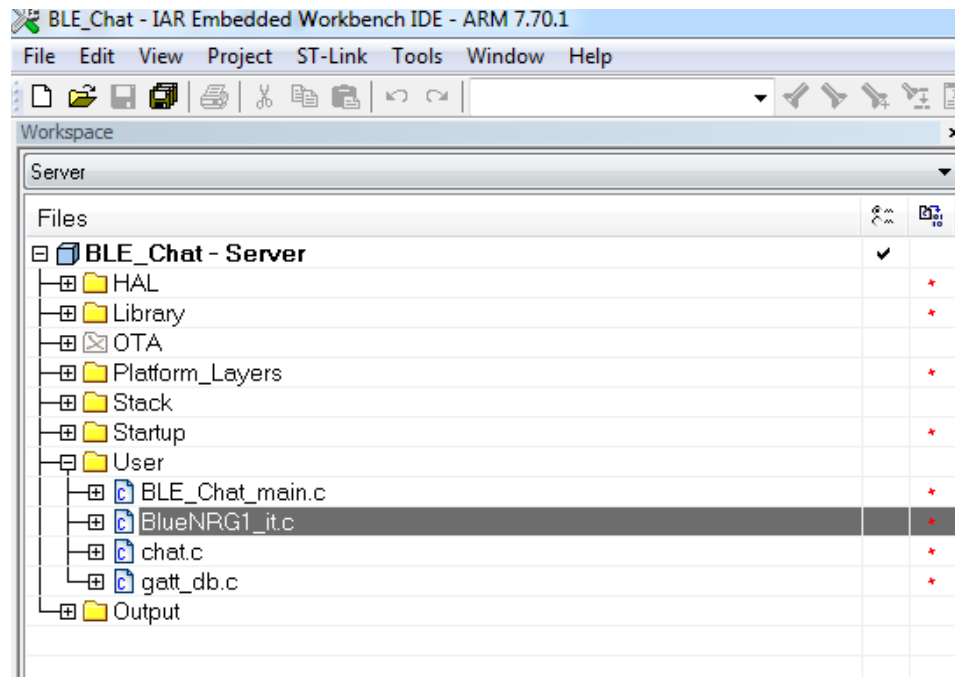
```
void Make_Connection(void)
```

```
uint8_t local_name[] = {AD_TYPE_COMPLETE_LOCAL_NAME,'B','l','u','e','N','R','G','1','_','C','h','a','t','0'};
```

```
ret = aci_gap_set_discoverable(ADV_IND, 0, 0, PUBLIC_ADDR, NO_WHITE_LIST_USE,  
sizeof(local_name), local_name, 0, NULL, 0, 0);
```

Enable SPP like connection- Chat Demo

4 Build and download



Enable SPP like connection- Chat Demo

5

```
AccessPort - COM16(115200,N,8,1) Opened
File Edit View Monitor Tools Operation Help
Terminal Monitor
TX Char Handle 000D, RX Char Handle 0010
Add_Char_Service() --> SUCCESS
BLE Stack Initialized
aci_gap_set_discoverable() --> SUCCESS
BlueNRG-1 BLE Chat Server Application (version: 1.0.0)
aci_gatt_init() --> SUCCESS
aci_gap_init() --> SUCCESS
aci_gatt_update_char_value() --> SUCCESS
Chat Service added.
TX Char Handle 000D, RX Char Handle 0010
Add_Char_Service() --> SUCCESS
BLE Stack Initialized
aci_gap_set_discoverable() --> SUCCESS
###
$atBlueNRG-1 BLE Chat Server Application (version: 1.0.0)
aci_gatt_init() --> SUCCESS
aci_gap_init() --> SUCCESS
aci_gatt_update_char_value() --> SUCCESS
Chat Service added.
TX Char Handle 000D, RX Char Handle 0010
Add_Char_Service() --> SUCCESS
BLE Stack Initialized
aci_gap_set_discoverable() --> SUCCESS
azertytest
```

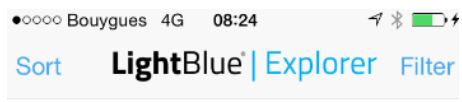
AccessPort
115 000

+

Reset the
board



Server image



Enjoying LightBlue Explorer?
Learn about our insights into BLE

Sign Up

Not Now

Peripherals Nearby

BlueNRG1
-66 No services

Virtual Peripherals

Blood Pressure
2 services

+ Create Virtual Peripheral

Client

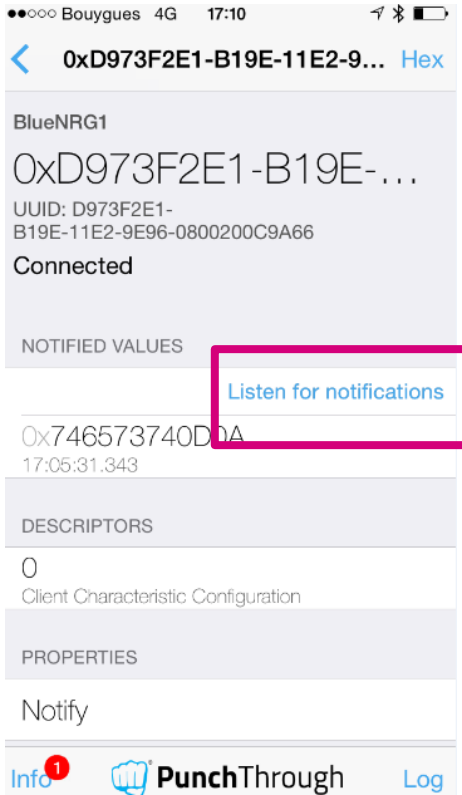
Enable SPP like connection- Chat Demo



6

```
AccessPort - COM16(115200,N,8,1) Opened
File Edit View Monitor Tools Operation Help
Terminal Monitor
TX Char Handle 000D, RX Char Handle 0010
Add_Chat_Service() --> SUCCESS
BLE Stack initialized
acl_gap_set_discoverable() --> SUCCESS
BlueNRG-1 BLE Chat Server Application (version: 1.0.0)
acl_gatt_intr() --> SUCCESS
acl_gatt_update_char_value() --> SUCCESS
Chat Service added.
TX Char Handle 000D, RX Char Handle 0010
Add_Chat_Service() --> SUCCESS
BLE Stack initialized
acl_gap_set_discoverable() --> SUCCESS
###
BlueNRG-1 BLE Chat Server Application (version: 1.0.0)
acl_gatt_intr() --> SUCCESS
acl_gatt_update_char_value() --> SUCCESS
Chat Service added.
TX Char Handle 000D, RX Char Handle 0010
Add_Chat_Service() --> SUCCESS
BLE Stack initialized
acl_gap_set_discoverable() --> SUCCESS
azertytest
```

Connect and enable Listen for notification



Server image

Client

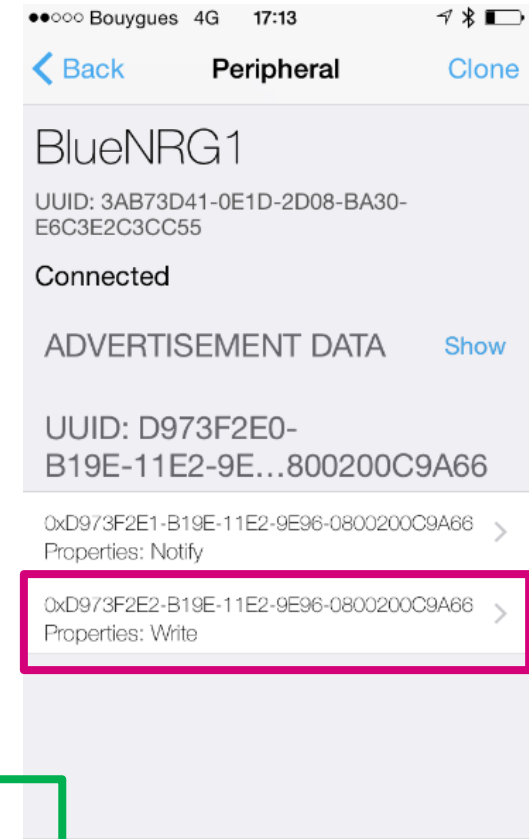
Enable SPP like connection

The Chat demo described in BlueNRG-1 User Manual section 7 [UM2071](#)

7

```
AccessPort - COM16(115200,N,8,1) Opened
File Edit View Monitor Tools Operation Help
Terminal Monitor
TX Char Handle 000D, RX Char Handle 0010
Add_Chart_Service() --> SUCCESS
BLE Stack initialized
acl_gap_set_discoverable() --> SUCCESS
BlueNRG-1 BLE Chat Server Application (version: 1.0.0)
acl_gatt_intr() --> SUCCESS
acl_gatt_intr() --> SUCCESS
acl_gatt_update_char_value() --> SUCCESS
Chat Service added.
TX Char Handle 000D, RX Char Handle 0010
Add_Chart_Service() --> SUCCESS
BLE Stack initialized
acl_gap_set_discoverable() --> SUCCESS
###
$atBlueNRG-1 BLE Chat Server Application (version: 1.0.0)
acl_gatt_intr() --> SUCCESS
acl_gatt_intr() --> SUCCESS
acl_gatt_update_char_value() --> SUCCESS
Chat Service added.
TX Char Handle 000D, RX Char Handle 0010
Add_Chart_Service() --> SUCCESS
BLE Stack initialized
acl_gap_set_discoverable() --> SUCCESS
azertytest
```

Write data from smartphone to BlueNRG-1

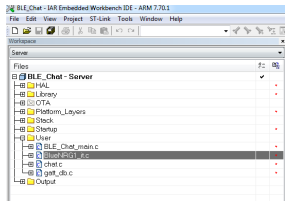


Attribute_Modified_CB Event pushed to Application

BlueNRG-1 Takeways



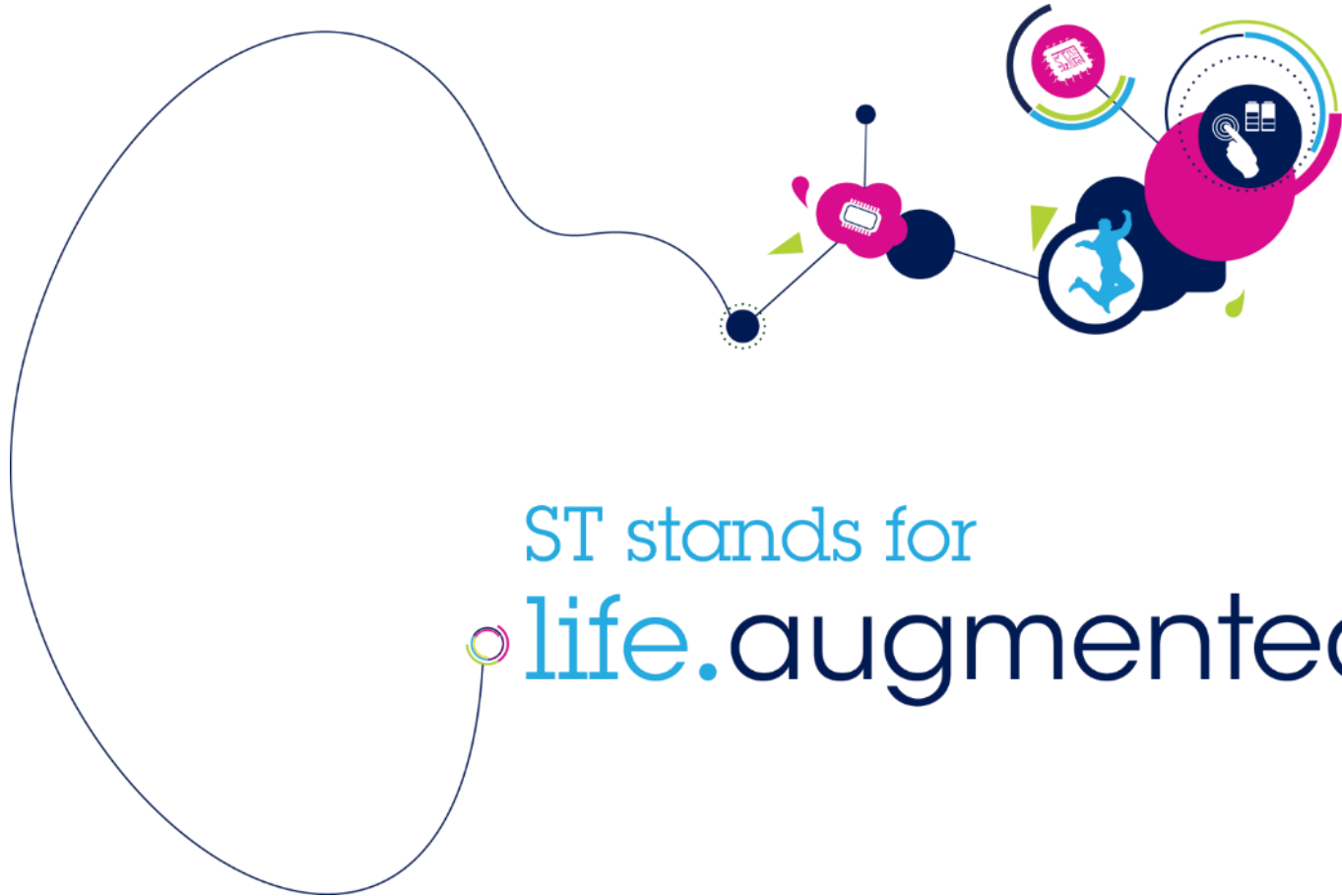
Optimized for ultra-low-power
Beacon mode : 16uA @ 1.28s



BlueNRG-1 Powerful Development Kit
Navigator : Promotion Tool
ST GUI : Comprehensive BLE concepts
BlueNRG-1 various code examples



BlueNRG-1 on the web
Dedicated and specific documentation on demand
@ rf-support-emea@st.com



ST stands for
life.augmented