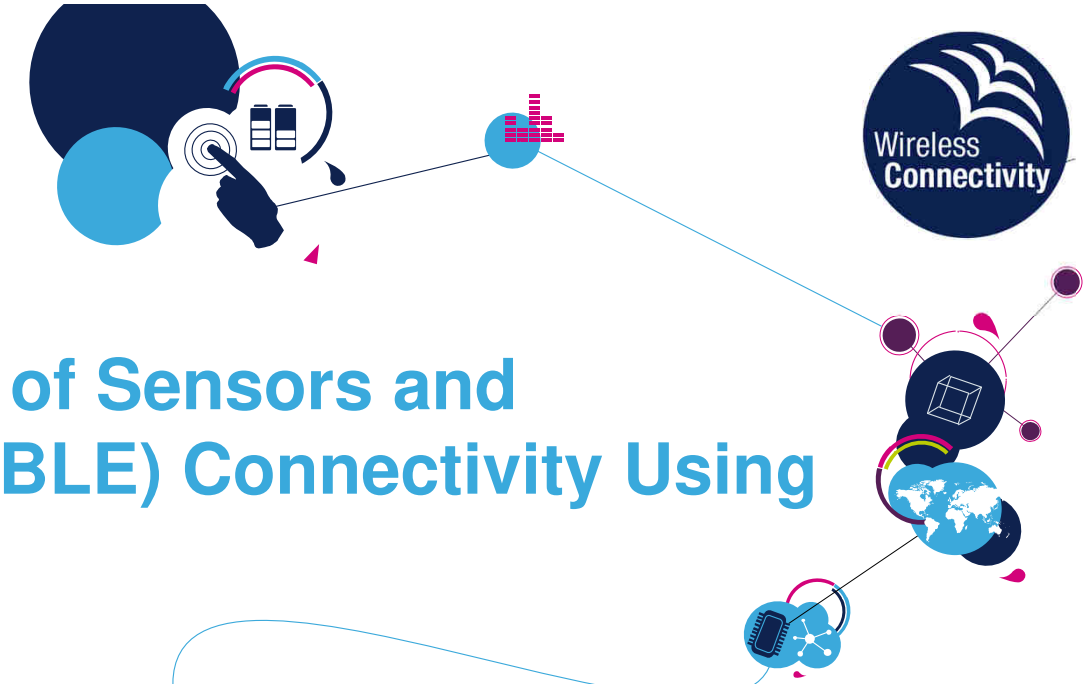


Simplify the Integration of Sensors and Bluetooth Low Energy (BLE) Connectivity Using the BlueTile Eval Kit

IoT Systems Development - Connectivity

WiFi Network: **STMicro**
Password: **STMTTT19**

life.augmented

Technology Tour 2019
Toronto, Canada | May 29

Agenda

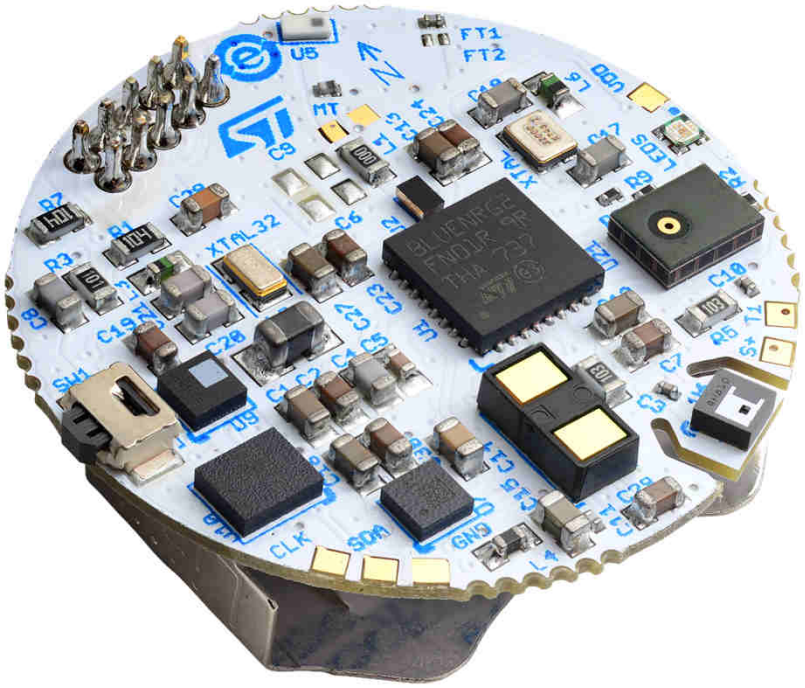
BlueTile
(STEVAL-BCN002V1)

- Training Material Check/Installation Help
- ARM® Keil MDK Installation
- Introduction to Bluetooth® Low Energy
- BlueTile Development Kit
- ARM® Keil MDK License Installation
- Lab 1: Getting Started with BlueTile “Serial Terminal Test”
- Lab 2: Connecting to the ST BLE Sensor app
- Lab 3: LED characteristic
- Lab 4: Accelerometer embedded events detection
- Lab 5: 9-axis Acc+Gyro+Mag Sensor Data Fusion
- Lab 6: Cloud data logging on IBM Watson
- Lab 7: Bonus – Voice over BLE

life.augmented

WiFi Network: **STMicro**
Password: **STMTTT19**

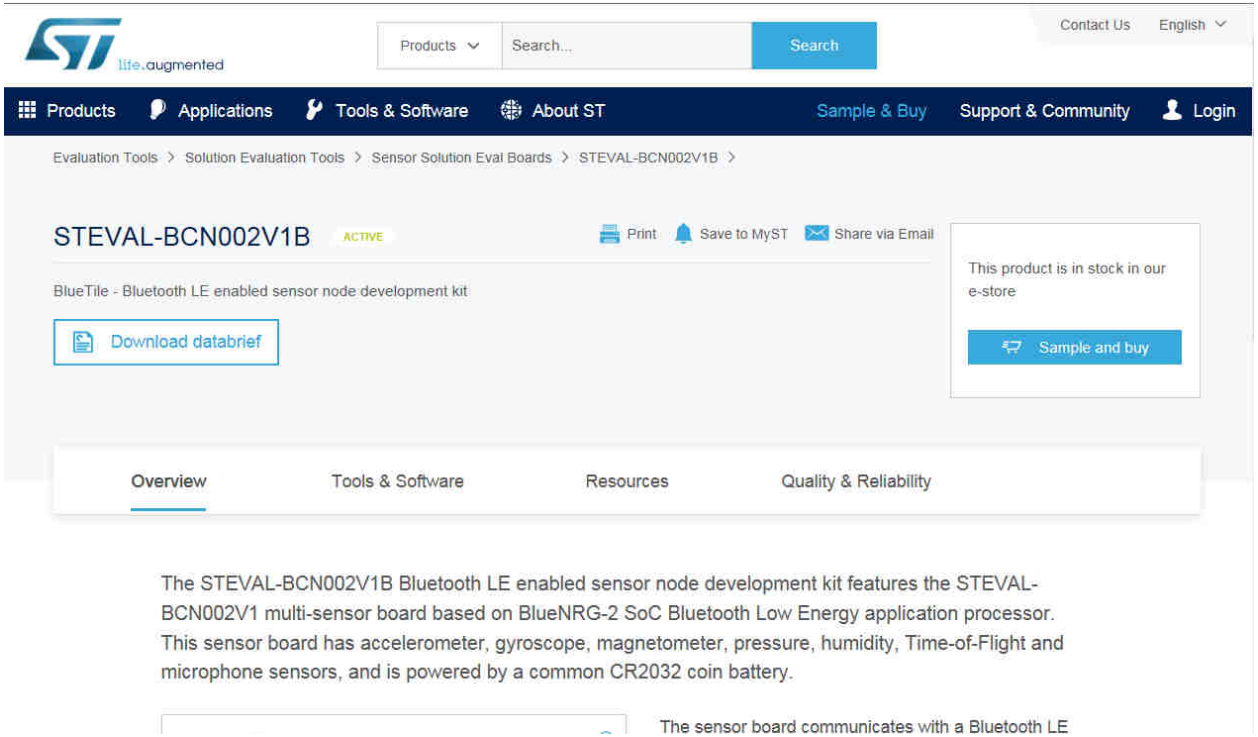
BlueTile (STEVAL-BCN002V1B)



life.augmented

On st.com

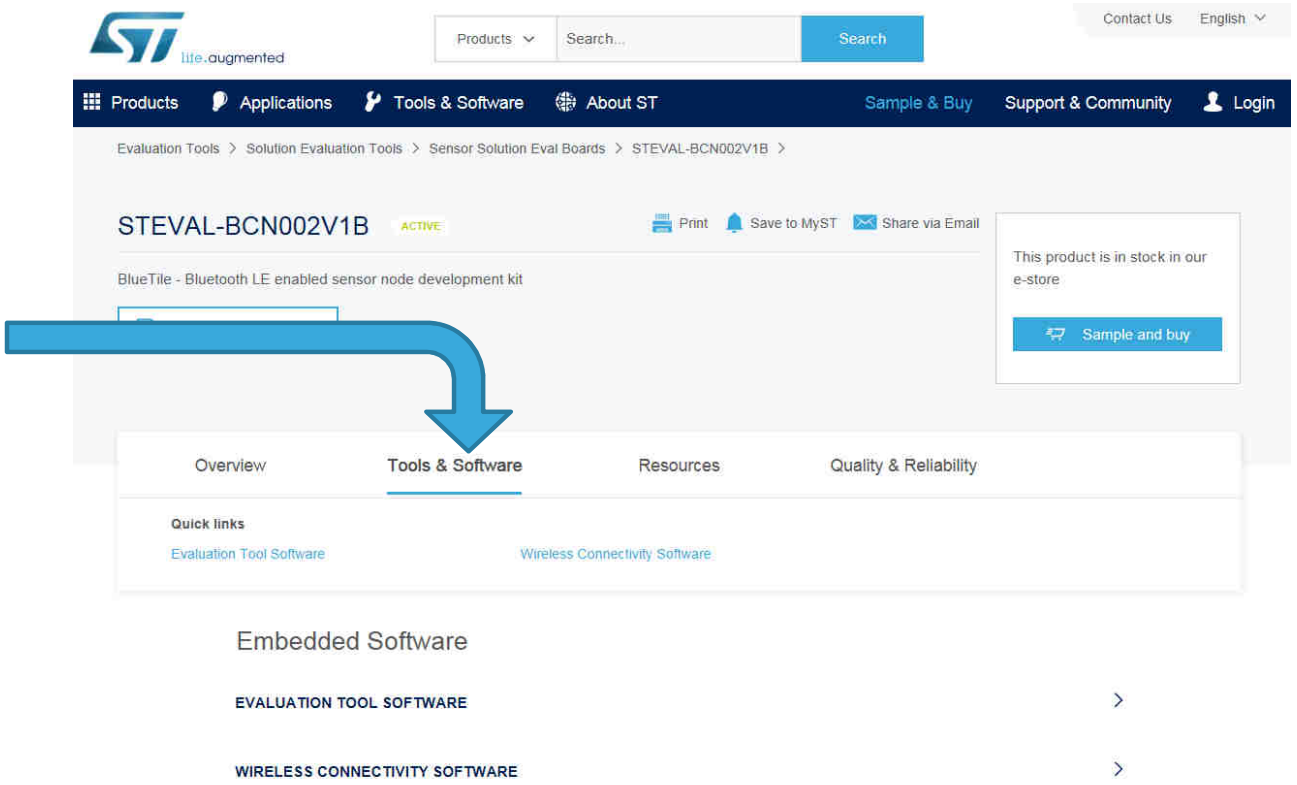
http://www.st.com/bluetile



BlueTile SDK

http://www.st.com/bluetile

Software Development Kit



life.augmented

BlueTile SDK

https://www.st.com/content/st_com/en/products/embedded-software/evaluation-tool-software/stsw-bluetile-dk.html

STSW-BLUETILE-DK ACTIVE

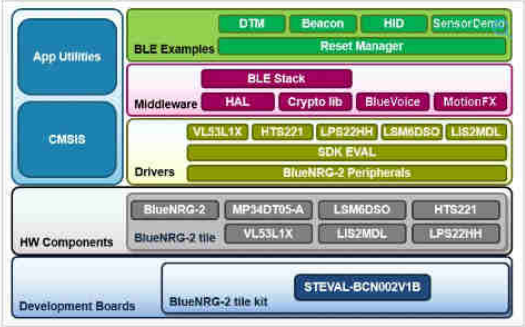
Print Save to MyST Share via Email

Software development kit for the BlueTile, including sensor data streaming over BLE, HID peripheral, Beacon, etc.

Get Software | Download databrief

Overview | Tools & Software | Resources

The STSW-BLUETILE-DK is based on STSW-BLUENRG1-DK evaluation SW package.



The STSW-BLUENRG1-DK package supports the BlueNRG-1 and BlueNRG-2 Bluetooth Low-Energy system-on-chip. This package includes a Wizard PC application to automatically generate the configuration header file needed for the BLE radio initialization. This package also includes BlueNRG-1 and BlueNRG-2 peripheral drivers and related examples, the BLE firmware stack together with the HAL (Hardware Abstraction Layer) and CryptoLib (Cryptographic Library).

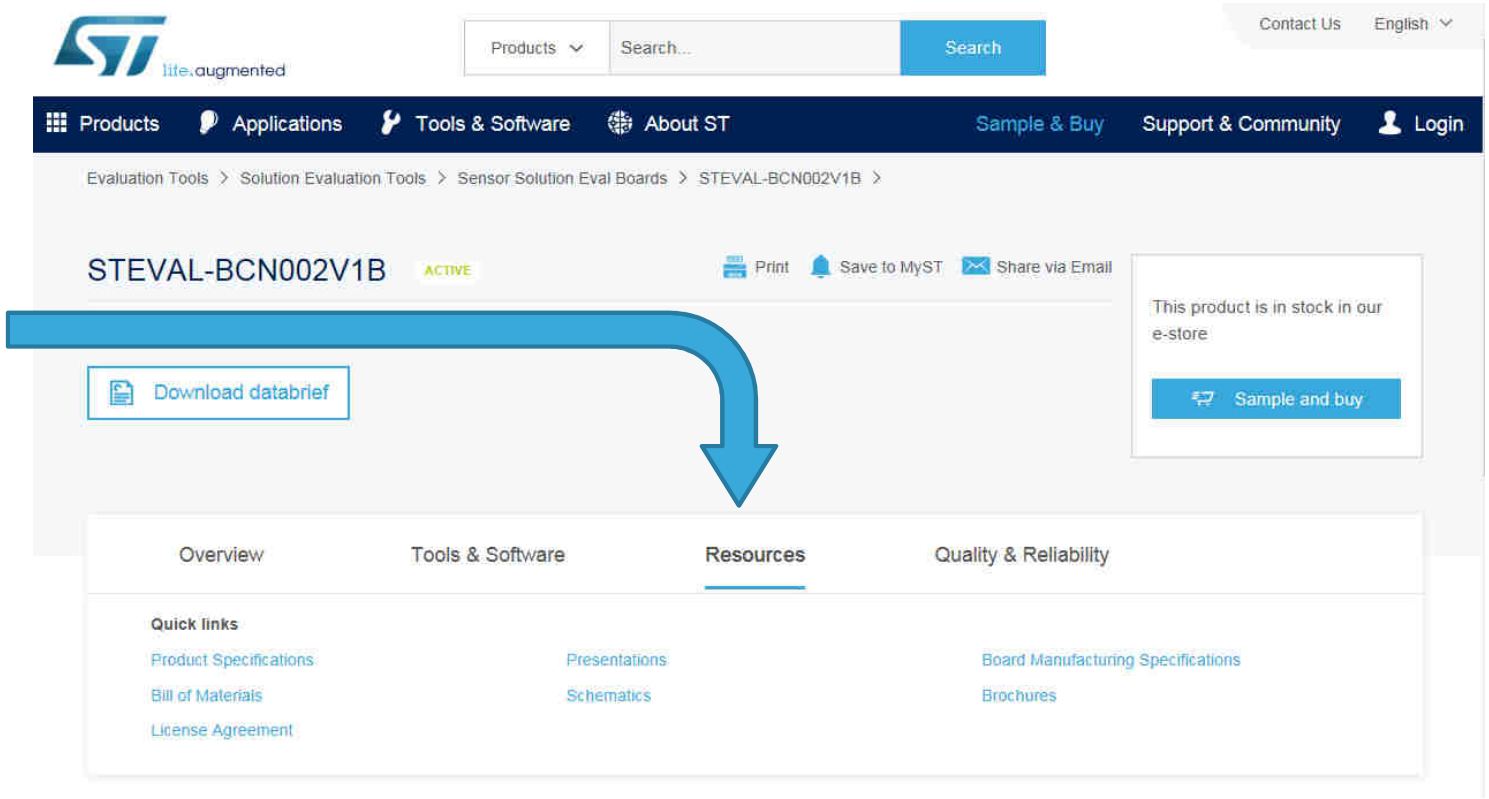
The STSW-BLUETILE-DK supports the BlueTile platform. This package extends STSW-BLUENRG1-DK

life.augmented

On st.com

http://www.st.com/bluetile

Schematic
Gerbers
BOM



life.augmented

Companion ST BLE Sensor app

ON YOUR PHONE

- Look for “**ST BLE Sensor**” on the App Store or Google Play

Download on the App Store

GET IT ON Google Play

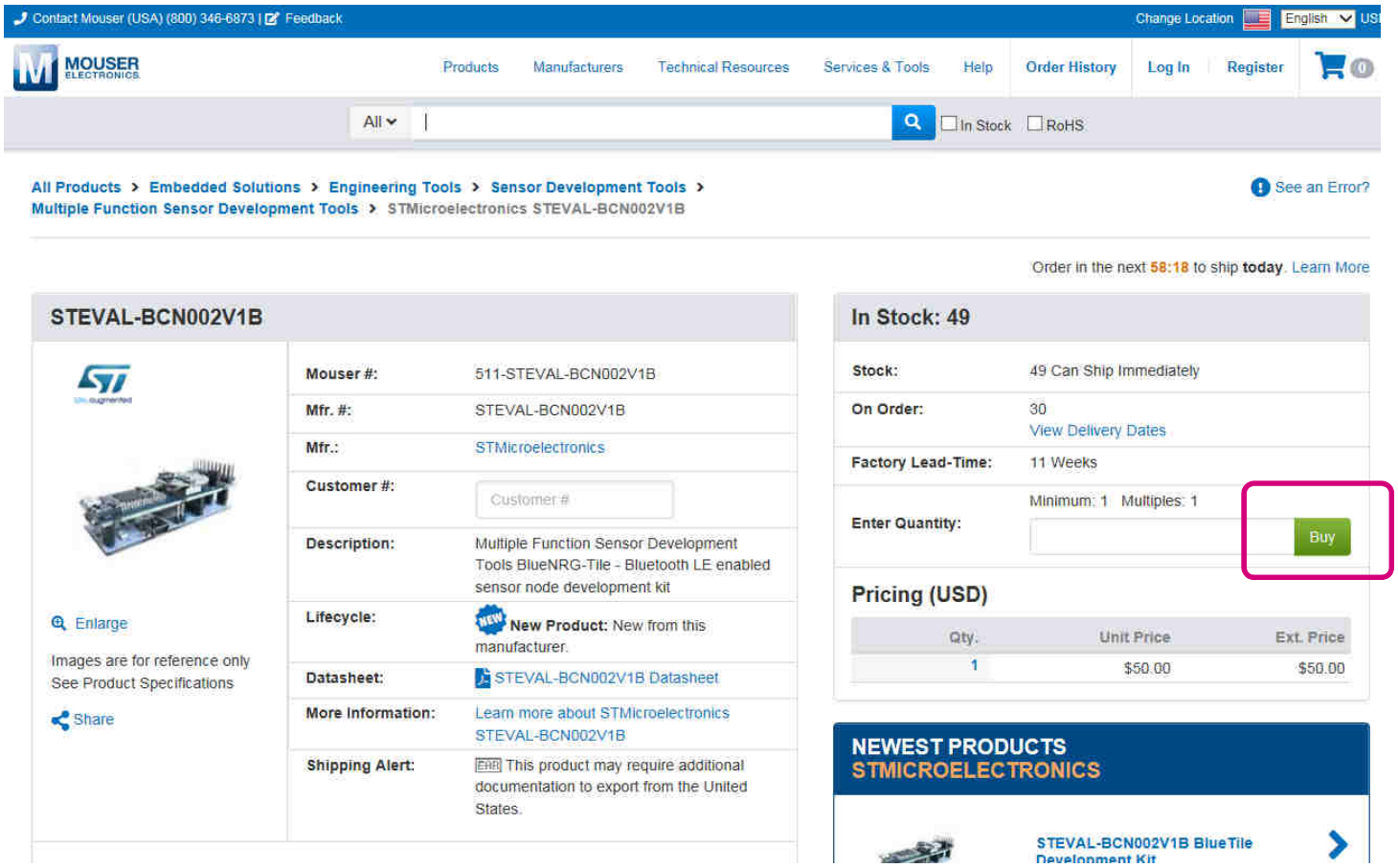
life.augmented

Where to get it

Sample & Buy

Part Number	Marketing Status	Budgetary Price (US$) *	Quantity	Core Product	ECCN (US)	Country of Origin	Order from Distributors	Order from ST
STEVAL-BCN002V1B	ACTIVE	50.0	1	BlueNRG-2, BALF-NRG-02D3, LSM6DSO, LIS2MDL, VL53L1X, MP34DT05TR-A, LPS22HH, HTS221	5A992C	ITALY	Check Availability	Buy direct

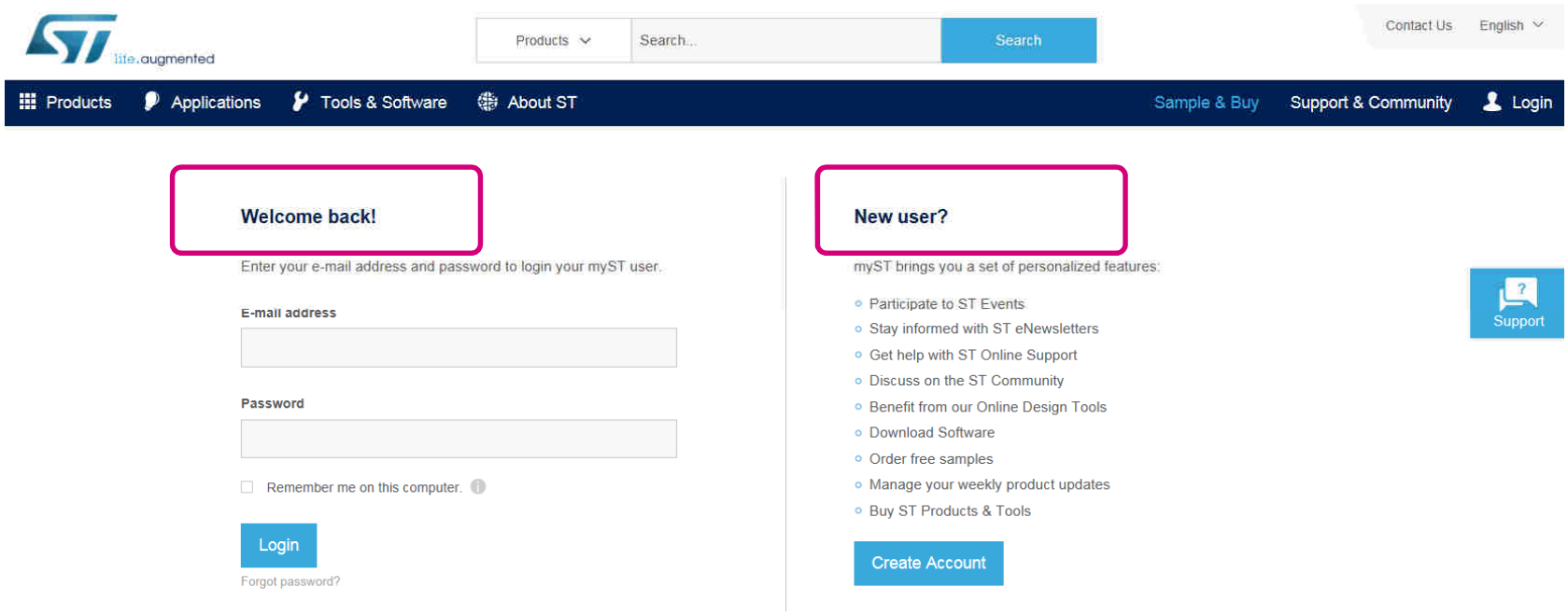
Where to get it



life.augmented

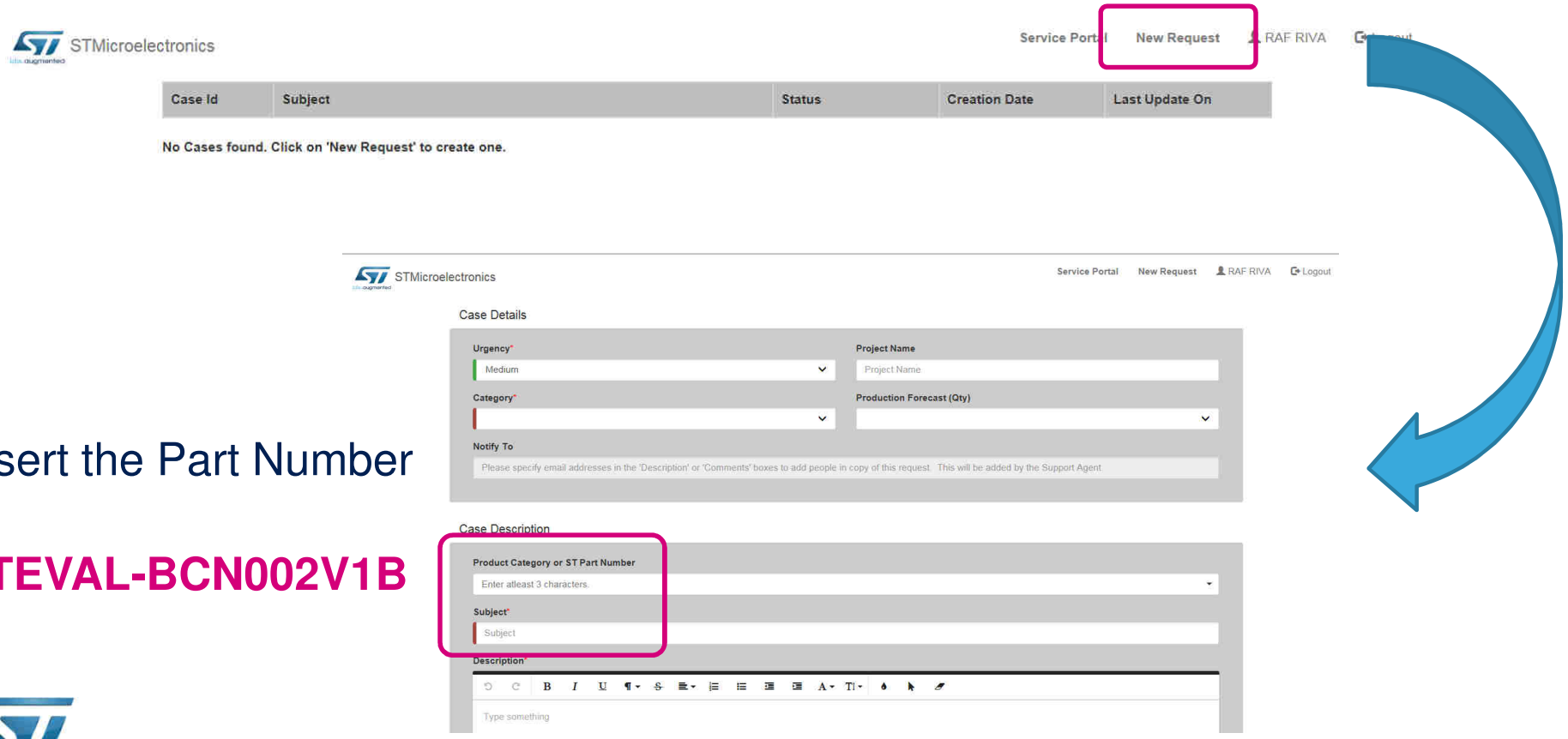
Online Support

https://my.st.com/ols



life.augmented

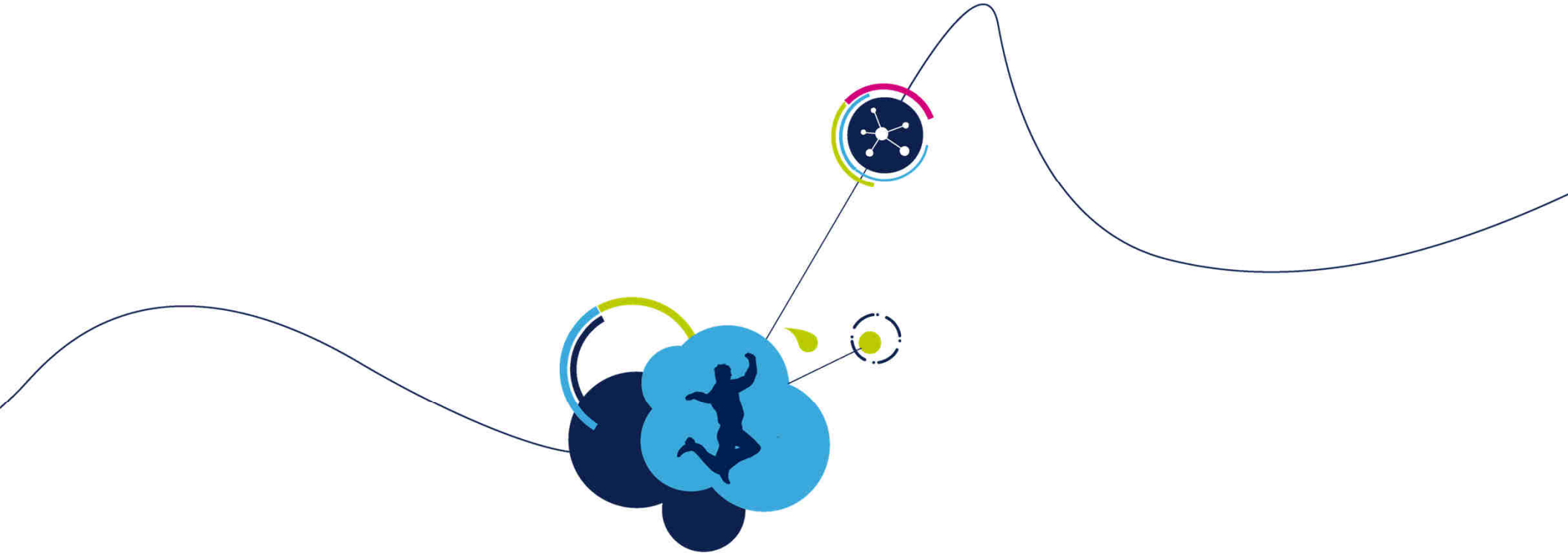
Online Support



Insert the Part Number

STEVAL-BCN002V1B

life.augmented

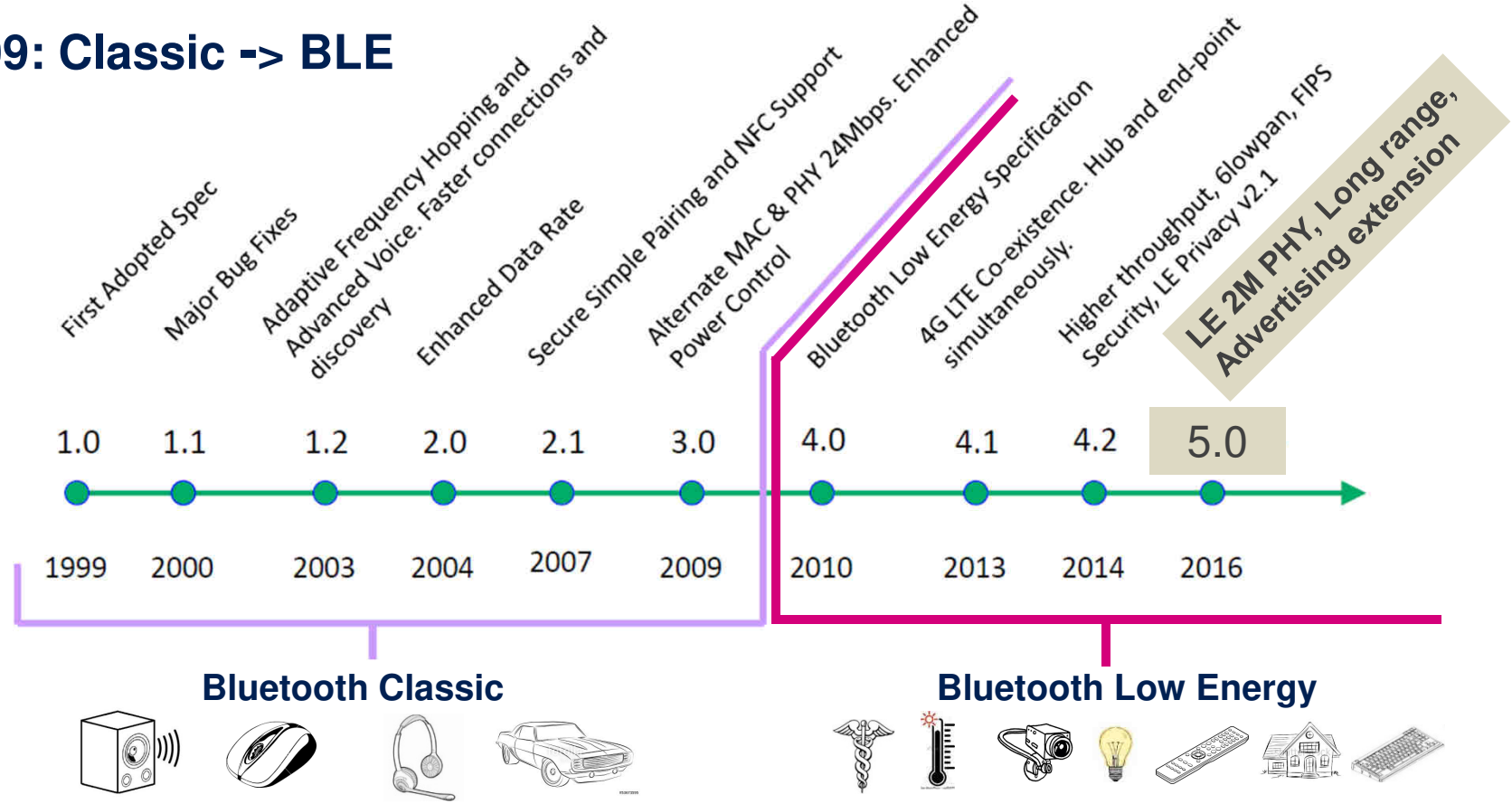


Introduction to Bluetooth Low Energy

life.augmented

Bluetooth® Evolution

- **20 Years in the market!**
- **2009: Classic -> BLE**

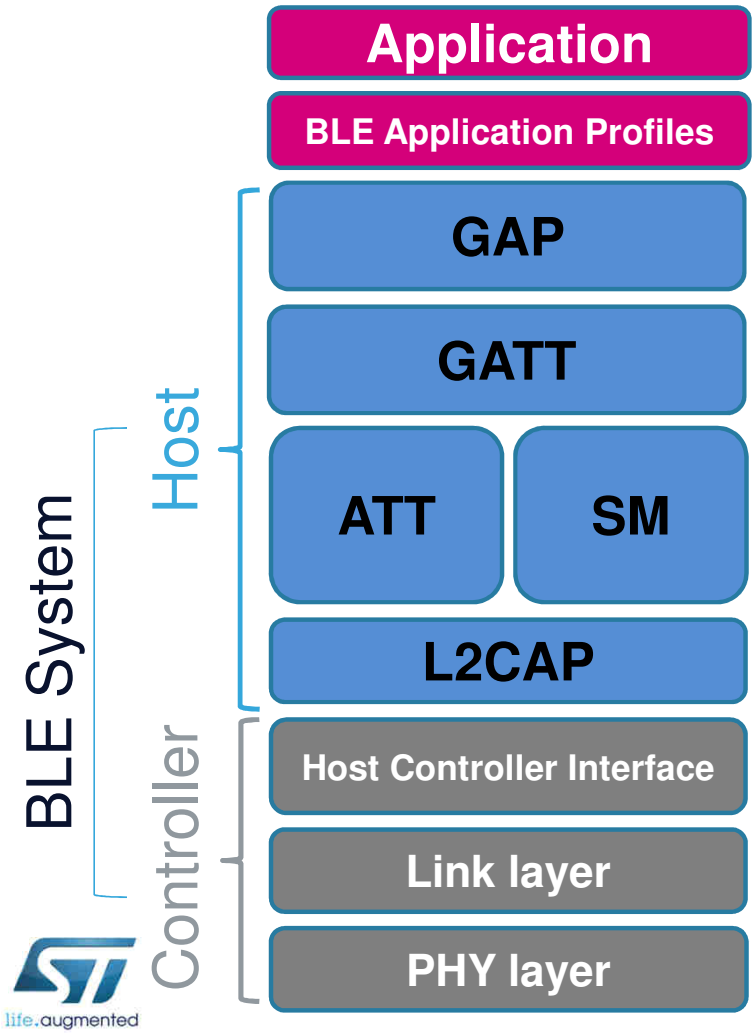


life.augmented

Bluetooth low energy (LE): Designed for Success

- **Lower power** than Bluetooth “Classic” (or Bluetooth 3.0)
  - **Lower duty cycle**
  - **Faster connection**
  - **Lower peak power** (relaxed RF parameters)
- Compatible with all major platforms (iOS, Android, Windows, Linux)
- **Multiple network topologies**:
  - **Point to point**: single master connects to single slave (and each slave can connect to 2 master)
  - **Star**: Multiple slaves connected to a single master
  - **Mesh**: introduced in the 2017
- State of the Art **encryption**, **security** including privacy/authentication

Protocol Stack



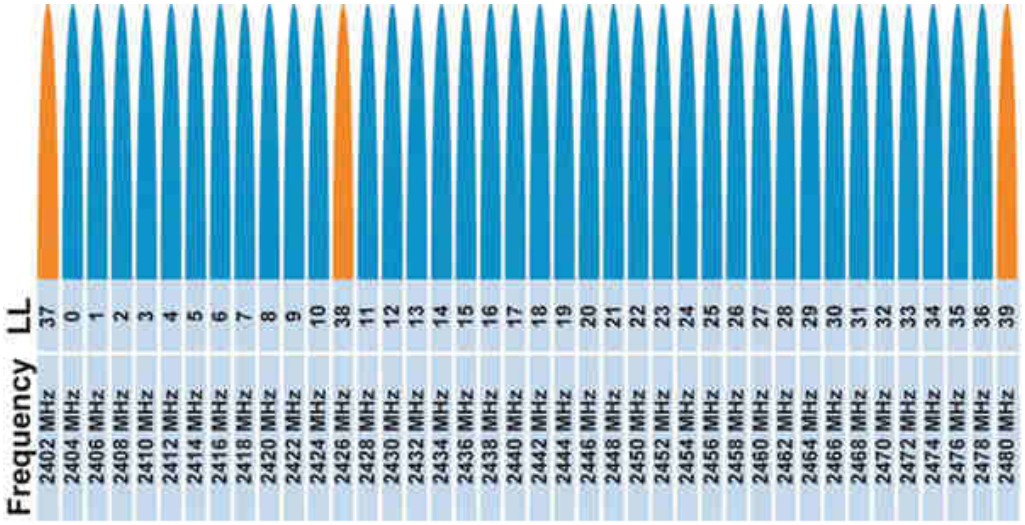
- **Generic Acces Profile**
  - How a device can discover and connect with one another
- **Generic Attribute Profile**
  - How services, characteristics can be discovered and then used
- **Attribute Protocol**
  - Protocol for discovering, reading, and writing attributes on a peer
- **Security Manager**
  - Handles the secure communication
- **L2CAP**
  - Protocol multiplexer. Segmentation and reassembly of packets
- **HCI**
  - Interface between Host and Controller
- **Link Layer**
  - Handles packets, channels, advertising, scanning & connections
- **Physical Layer**
  - Transmits/receives bits

life.augmented

Protocol Stack: PHY

- A **BLE Radio** is a **2.4 GHz ISM Band Transceiver**

- **40 RF Channels**, **2 MHz Channel Spacing**. Two types of channels:
  - **Advertising channels** (**3 - orange**) for Advertising Packets. Used for Discoverability and for Broadcasting/Observing
  - **Data Channels** (**37 - blue**) for Data Packets. Used to send application data in Connection

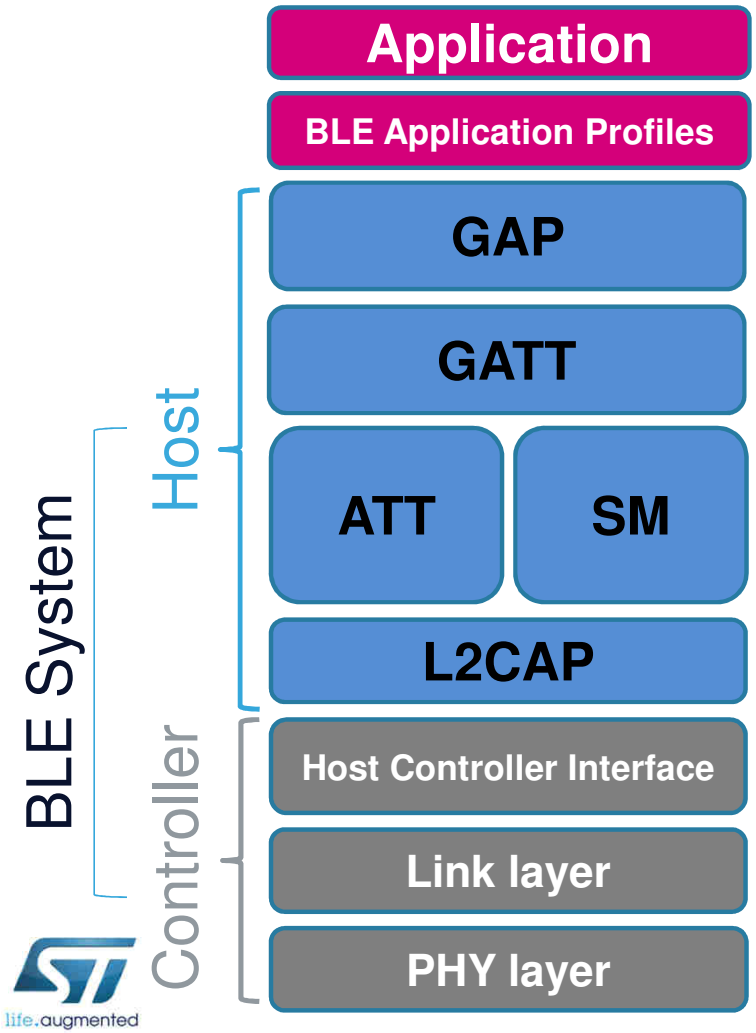


Source : Bluetooth® SIG

- **GFSK Modulation**
  - BT = 0.5
  - Modulation Index = 0.5
  - “pulse shaping” Gaussian filter “smooths” transitions from zero to one reduces spectral width

life.augmented

Protocol Stack - GAP



- Who controls the radio network?
  - **GAP (Generic Access Profile)**
  - Defines 4 roles
    - **Broadcaster**
    - **Observer**
    - **Peripheral**
    - **Central**

Peripheral

Connection

I'm looking for John

Central

Peripheral is in Advertising mode.
Sends Advertising packets.

Central is **Scanning**.
Look for known peripherals by *listening* to advertising packets.

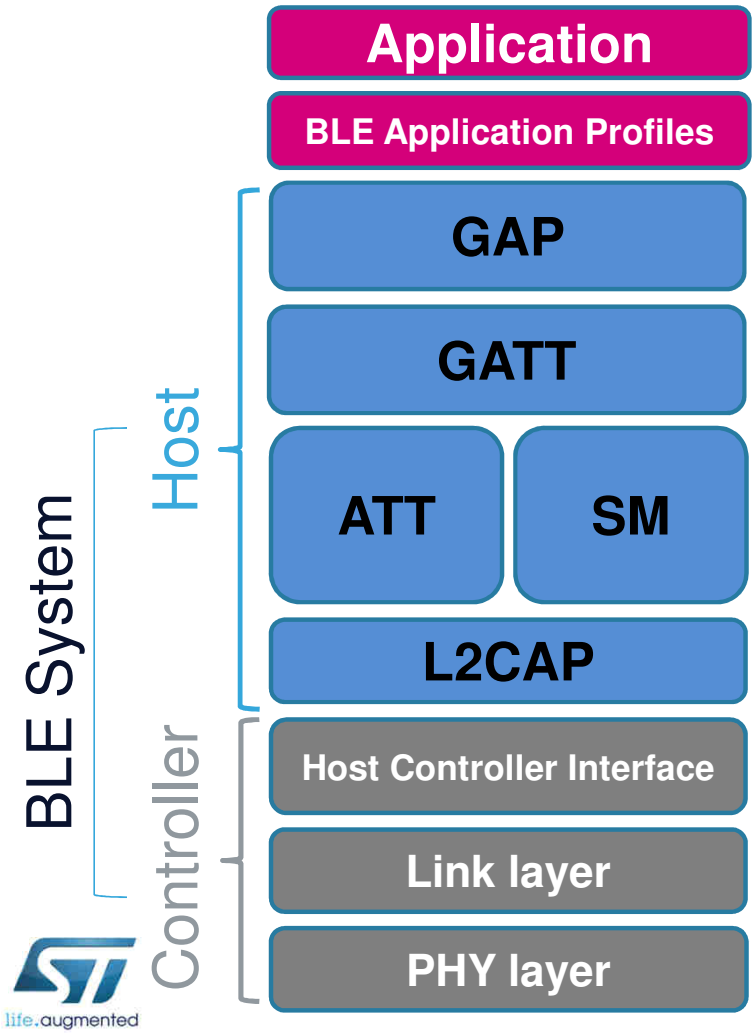
My name is John
My capabilities are X, Y, Z, etc…

Advertising **packets consists of multiple information units known as AD types**.
E.g.: Local Name, TX Output Power, Manufacturer Specific info, etc…

As the John peripheral is found, it initiates a BLE connection.

life.augmented

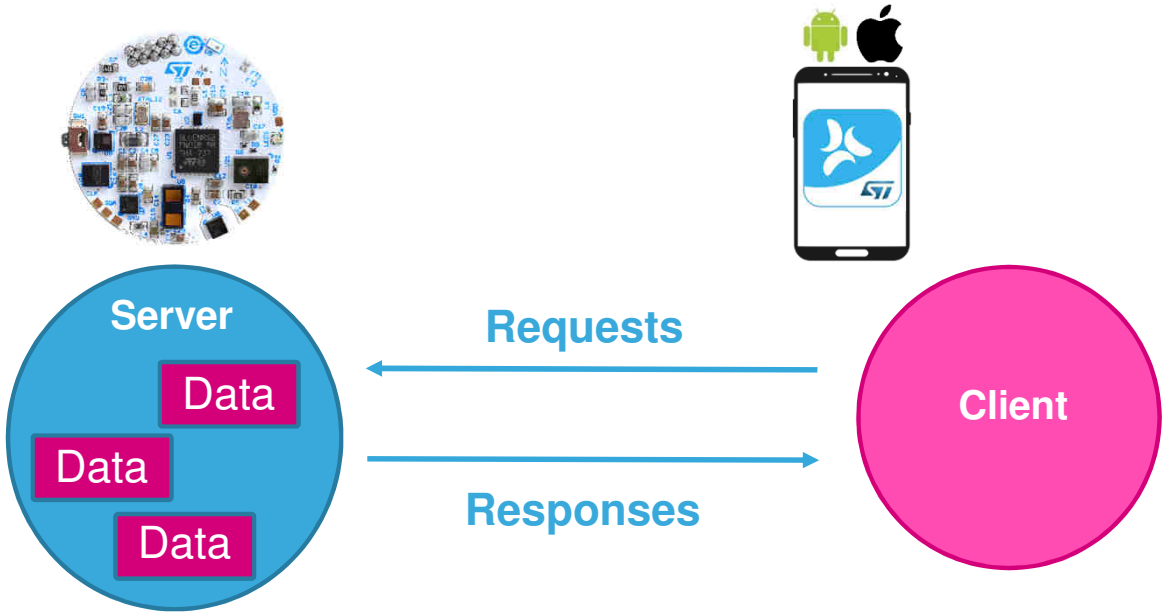
Protocol Stack - GATT



- Who controls the data flow?
  - **GATT (Generic Attribute Profile)**
  - Defines 2 roles
    - **Server**
    - **Client**

GATT

Server contains the data.
Receives requests, executes, responds.
Can indicate value

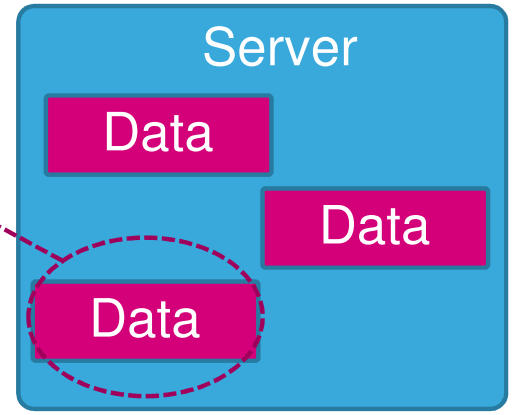


Client talks to the server.
Sends requests, wait for response.
Can confirm indications

Protocol PDU Type	Sent by	Description
Request	Client	Client requests something from server (it always causes a response)
Response	Server	Server sends response to a request from a client
Command	Client	Client commands something to server (no response)
Notification	Server	Server notifies client of new value (no confirmation)
Indication	Server	Server indicates to client new value (it always causes a confirmation)
Confirmation	Client	Confirmation to an indication

Attributes

- **Data are organized** and exposed as **attributes**



Each **Data** element in the Server is called **Attribute**

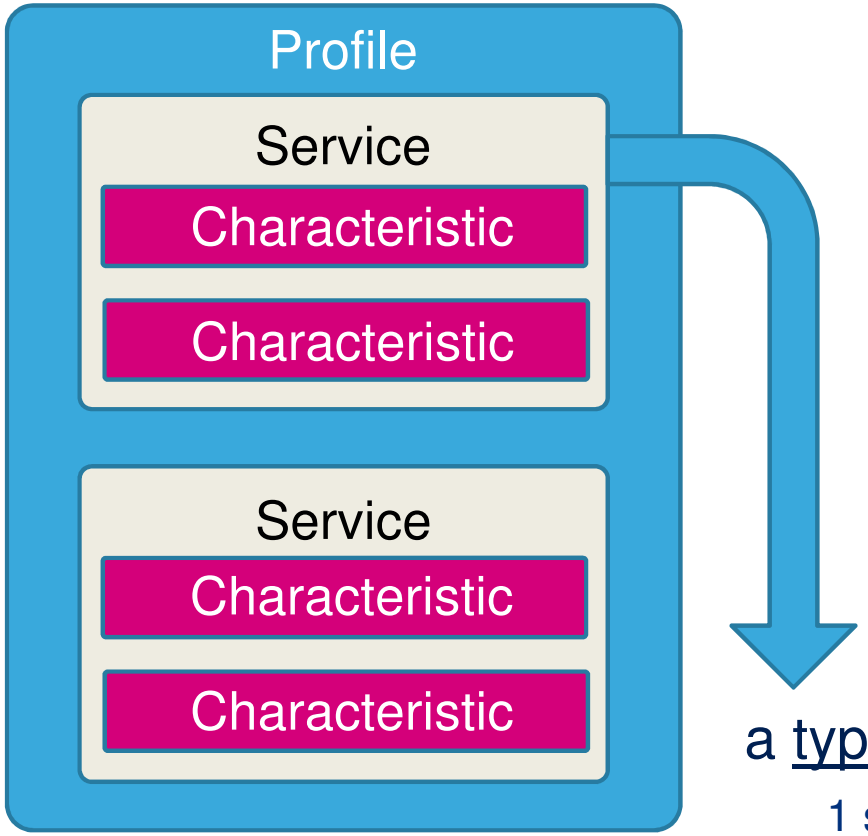
- Each attribute has:
  - A **handle**: it identifies the attribute on the server
  - A **type** (defined by Universal Unique ID - **UUID**): what the value means
    - 16-bit UUID: pre-defined by Bluetooth SIG
    - 128-bit UUID: Vendor Specific identifiers
  - A **value** (0 to 512 octets)
- Example

Handle	Type	Value
0x0009	«Device Name»	“Temperature Sensor”
0x0022	«Battery State»	0x04
0x0098	«Temperature»	0x0802

Source : Bluetooth® SIG

life.augmented

GATT Profile



- A GATT Profile defines how attributes are organized and how the application can access them.
- Attributes are organized in Services and Characteristics

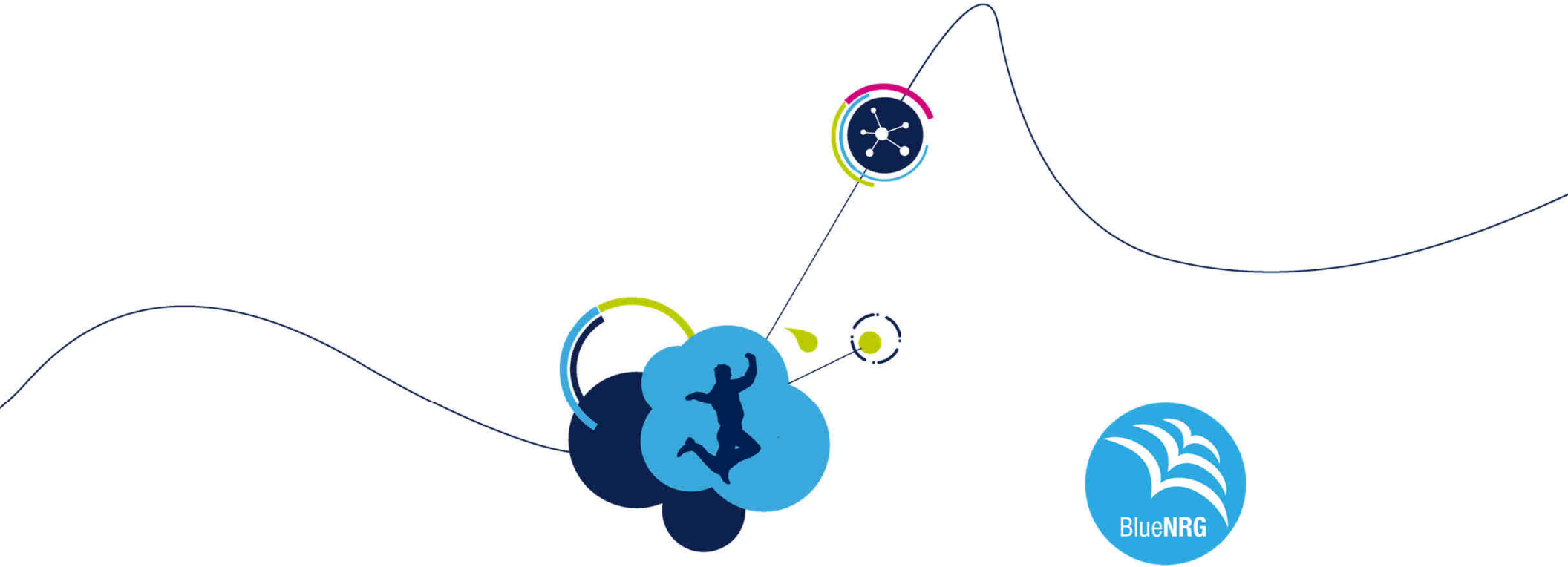
a typical example:

1 service: “ARG” *(Angular Rate and Gravity)*

2 characteristics: “Gyro”, “Acc”

Values: [0,-1,+2], [-10,+15,+950]

life.augmented

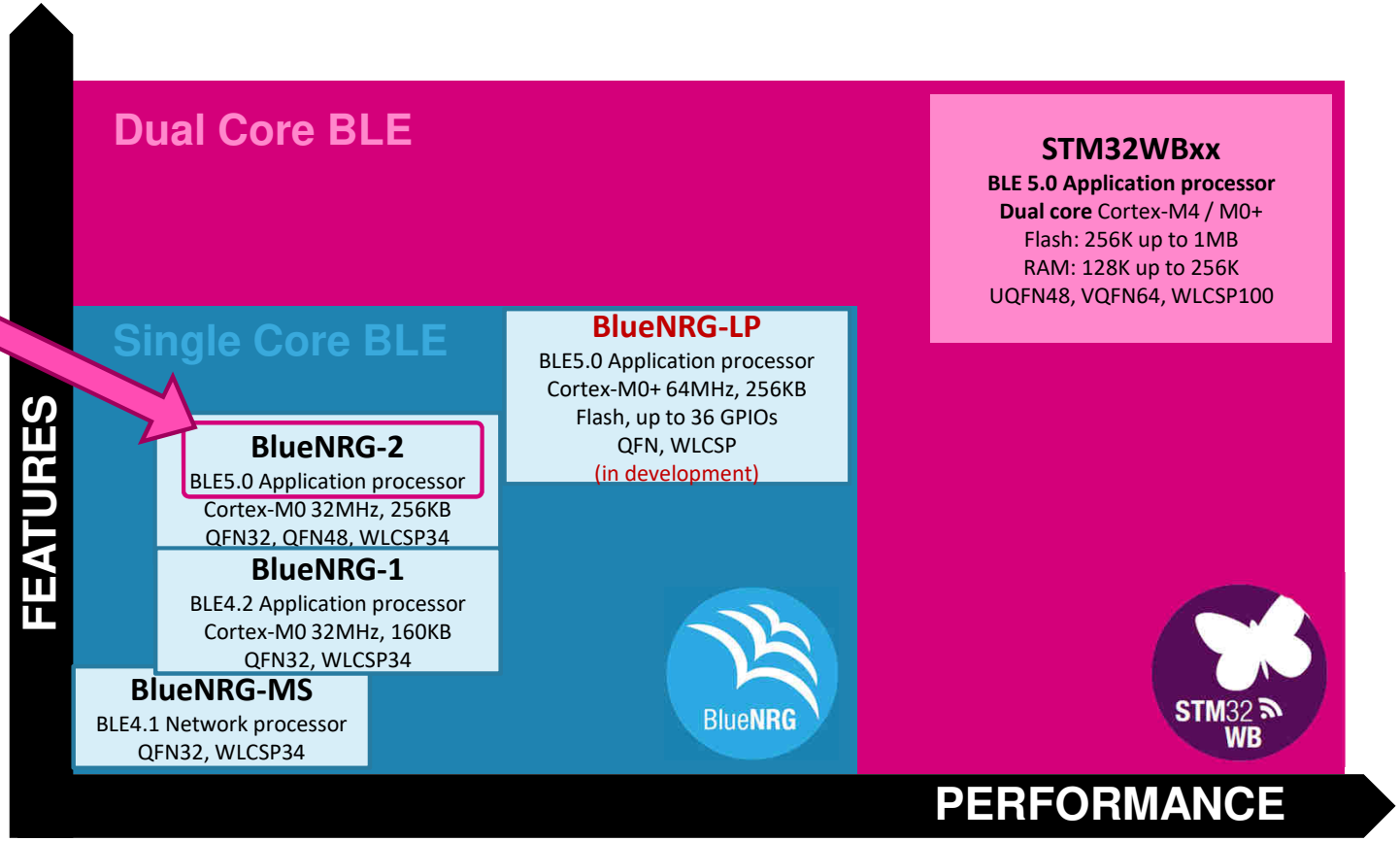


ST BLE devices

life.augmented

ST BLE Roadmap

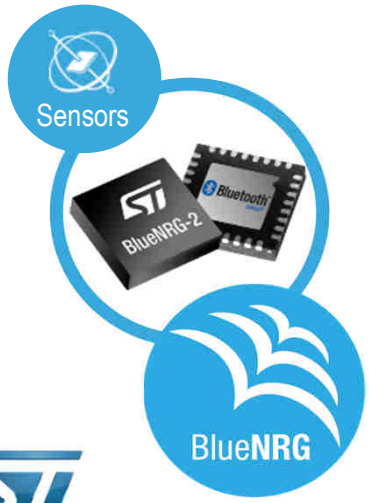
BlueNRG-Tile



life.augmented

BlueNRG-2 SoC at a glance

The lowest power consumption	Processing power on demand	Flexible memory architecture	Optimized BLE Radio stack	Maximum security
>3 years lifetime on CR2032(*) 59 µA/MHz 0.9 µA sleep	Low-power architecture, Cortex-M0 @ 32 MHz	256 KB eFLASH 24 KB ULL SRAM (with full SRAM data retention)	70 kB FLASH 8 kB RAM 0.9uA with full RAM retention	ECC-256 AES-128 Factory UID Secure KEY



Seamless connection with SENSORS

Robust and Reliable BLE Link

Small form factor

5

Bluetooth 5.0 certified

Privacy 1.2 and secure connection 4.2

data

Faster and more reliable data transfer

(*) Based on the average current consumption in connection mode (7.059 µA, connection interval 1000 ms)

life.augmented

Bluetooth LE programmable processor

Bluetooth LE
5.0
ULP radio

RF
PHY
+8 dBm

BLE 5.0

ARM
Cortex M0
32 MHz

Flash
256 KB

SRAM
24 KB

PROGRAMMABLE
MCU
with eFLASH
to host application code

Application and Profiles

BLE Protocol stack

Advanced High-Performance Bus (AHB)

DMA

PROGRAMMABLE
PERIPHERALS

PDM Interface

Decimation filter

I2C

RTC (timers)

ADC 10 bit
4 channels

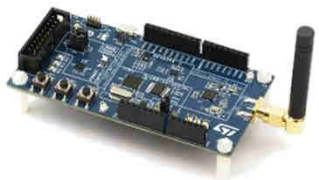
UART

SPI

DC/DC
LDO

GPIO x 26

PROGRAMMABLE
Digital I/O



STEVAL-IDB008V2

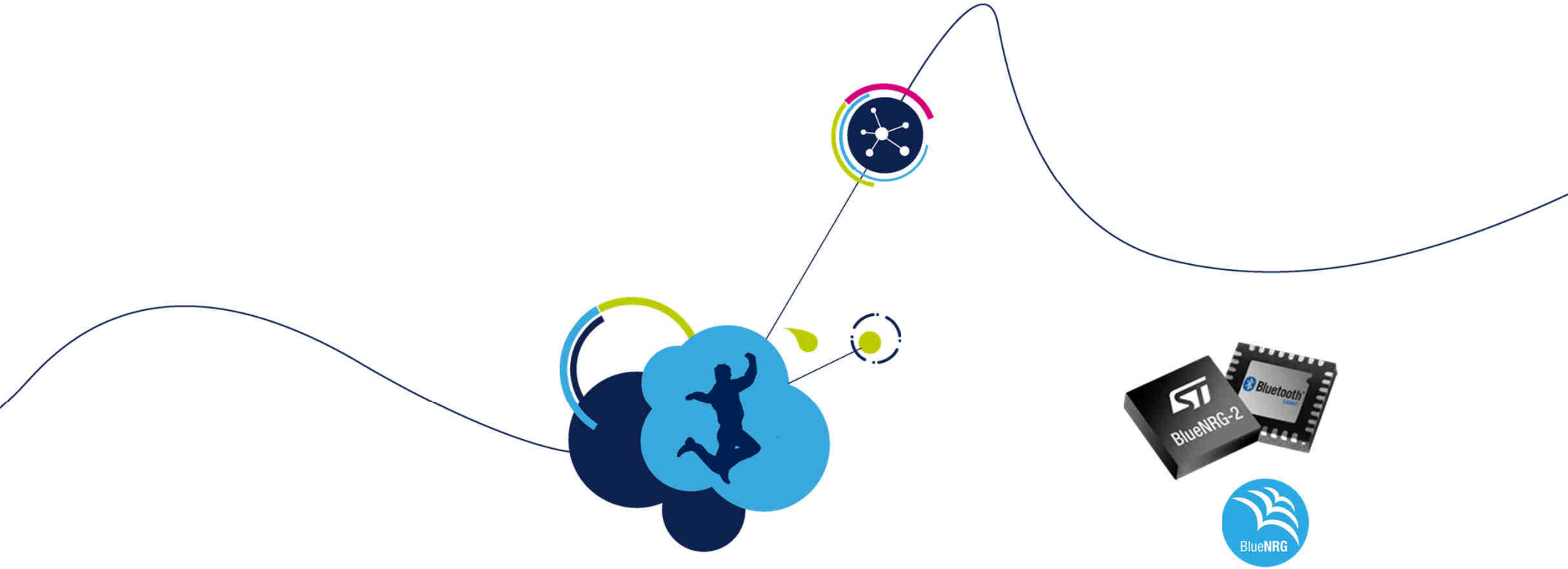
STEVAL-BCN002V1B

life.augmented

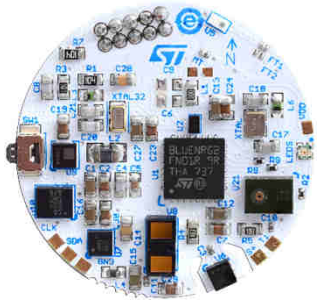
IAR SYSTEMS

KEIL™ Tools by ARM

atollic

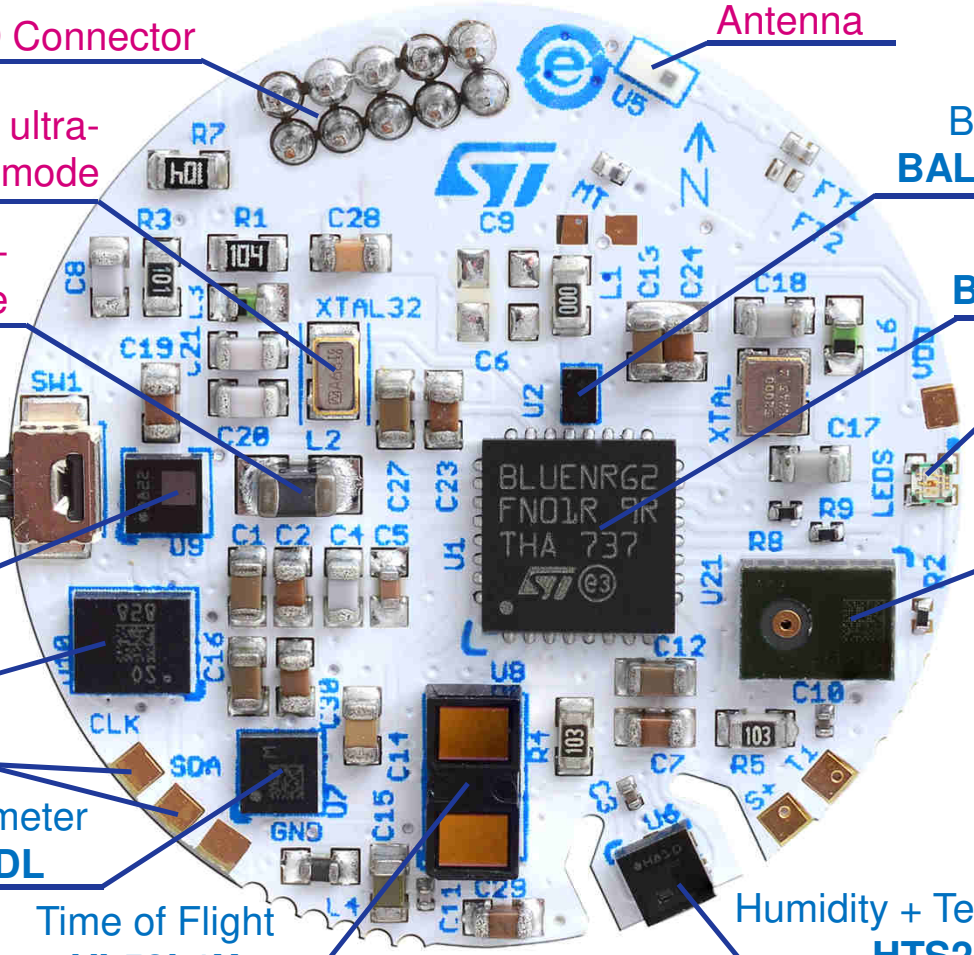


BlueTile Development Kit



life.augmented

STEVAL-BCN002V1



SWD Connector

32KHz XTAL to enable ultra-low-power BLE sleep mode

L2 bead to enable low-power BLE active mode

User button

Pressure Sensor
LPS22HH

Accelerometer + Gyro
LSM6DSO

I2C for expansion

Magnetometer
LIS2MDL

Time of Flight
VL53L1X

Antenna

Balun
BALF-NRG

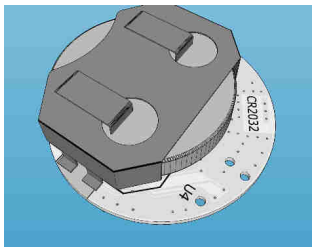
BLE SoC
BlueNRG-232

RGB LED

Microphone
MP34DT05-A

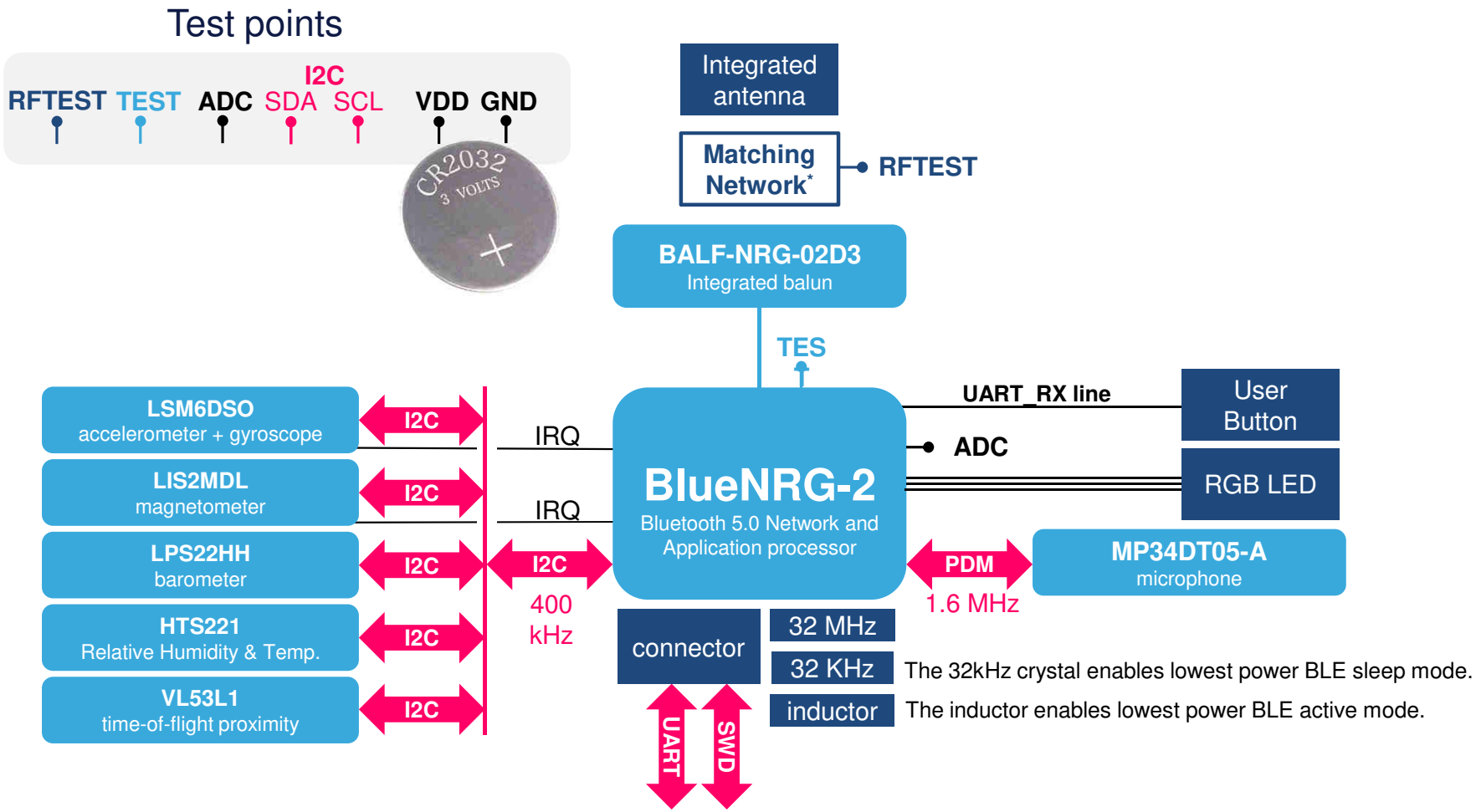
Humidity + Temperature
HTS221

Bottom view



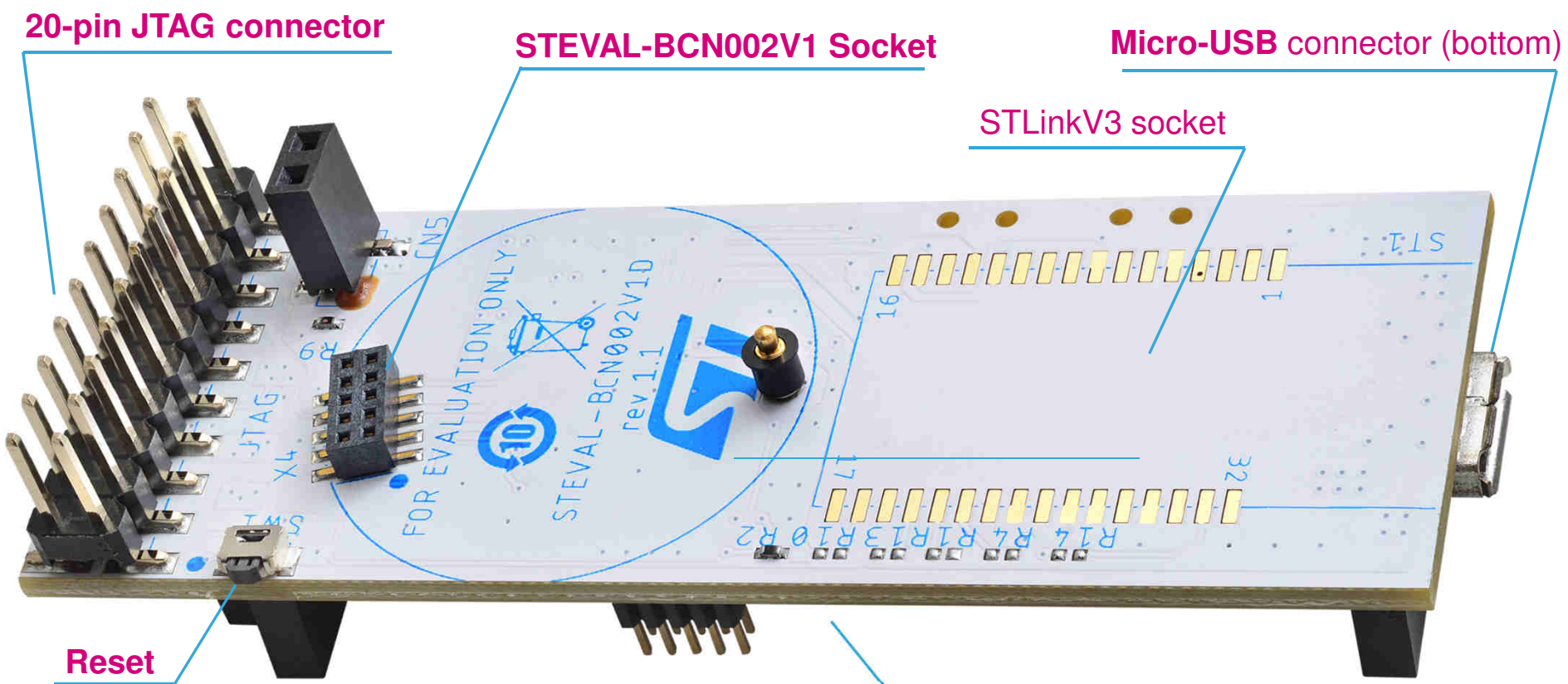
life.augmented

STEVAL-BCN002V1 Block Diagram



life.augmented

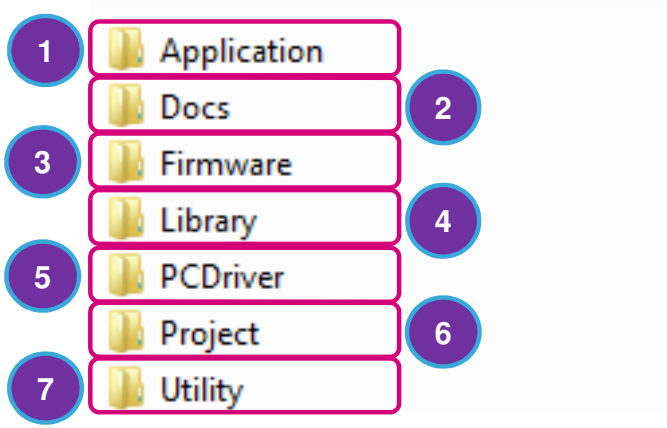
STEVAL-BCN002V1D



life.augmented

BlueTile SDK overview

https://www.st.com/content/st_com/en/products/embedded-software/evaluation-tool-software/stsw-bluetile-dk.html

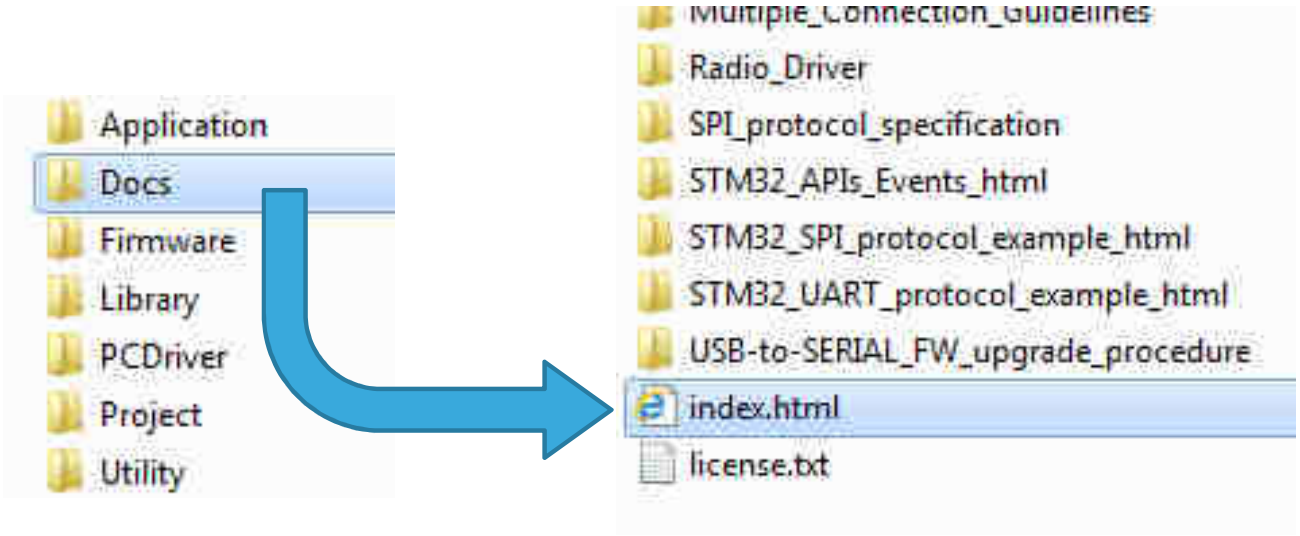


1. PC Applications
2. Documentation
3. Pre-built FW images
4. Low level drivers and BLE stack library
5. Virtual COM port drivers
6. Reference examples in source code
7. Utility section: IAR BlueNRG-2 Flasher

life.augmented

Documents

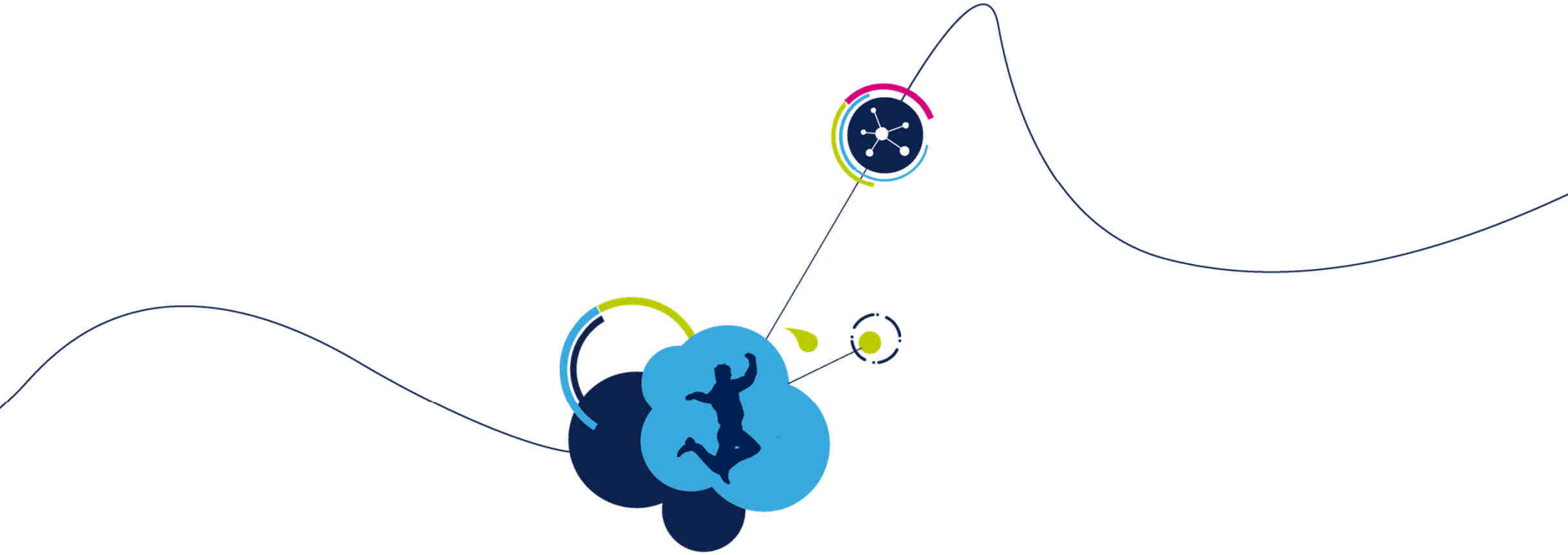
- Open the **Docs** folder



- **Double click** on **index.html**

- Device Datasheet
- BLE stack documentation
  - Release notes
  - APIs and events
  - Programming manual
- Device (and kits) documentation
  - PCB design guidelines
  - Bring up guide
  - Getting started

life.augmented

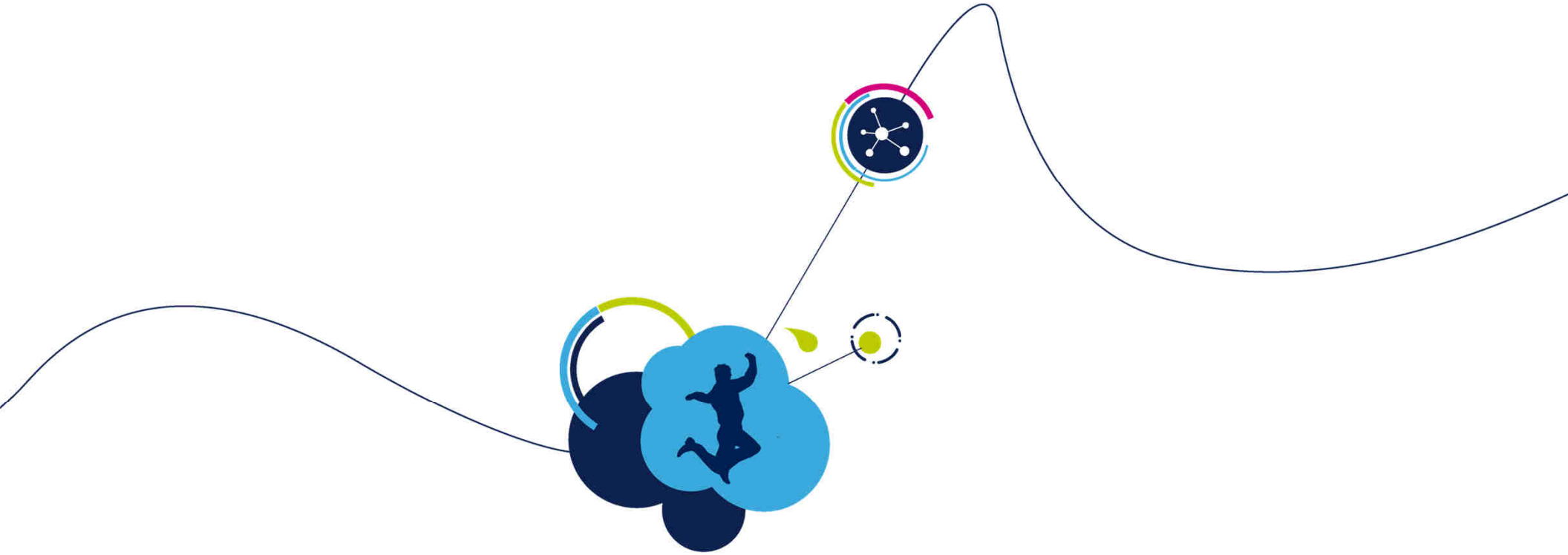


Hands on overview

life.augmented

Labs overview

- **Lab 1**: Getting started with STEVAL-BCN002V1 “Serial Terminal Test”
- **Lab 2**: Connecting to the ST BLE Sensor app
- **Lab 3**: LED characteristic
- **Lab 4**: Accelerometer embedded events detection
- **Lab 5**: 9-axis Acc+Gyro+Mag Sensor Data Fusion
- **Lab 6**: Cloud data logging on IBM Watson
- **Lab 7**: Voice over BLE



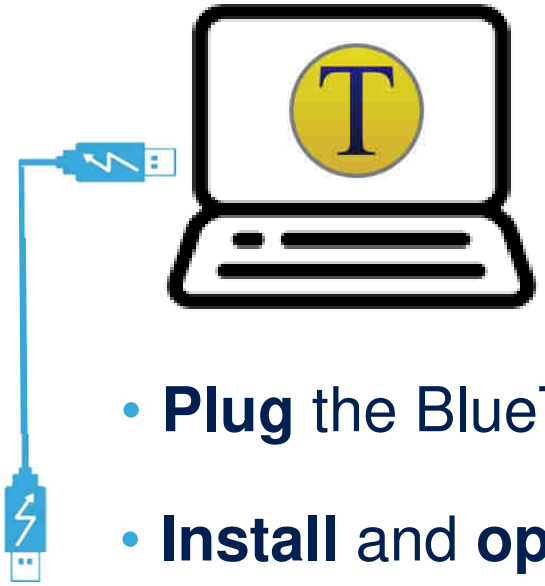
Lab 1

Getting Started with STEVAL-BCN002V1B

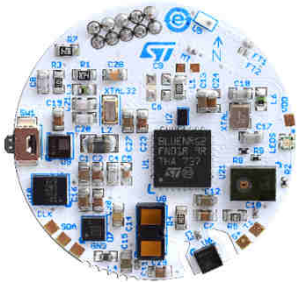
“Serial Terminal Test”

life.augmented

“Serial Terminal Test”



- **Plug** the BlueTile to the PC using the USB cable
- **Install** and **open** **Tera Term** and **configure** serial terminal

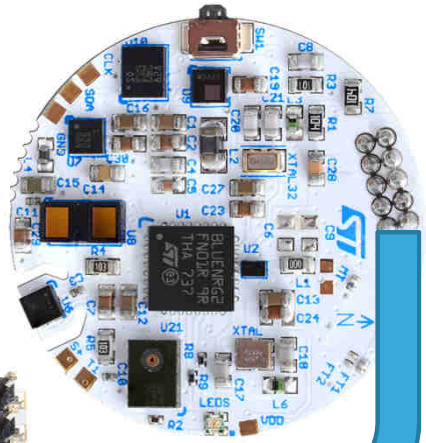


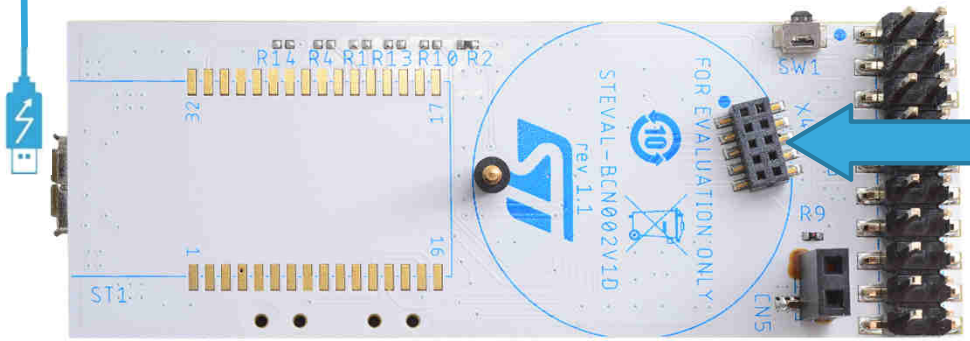
life.augmented

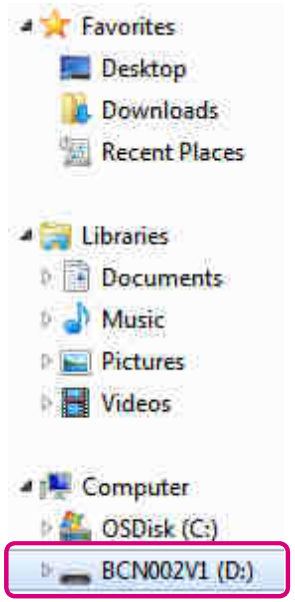
Connect your STEVAL-BCN002V1B to the PC using USB

Connect the USB cable

The BlueTile comes already connected to the daughterboard





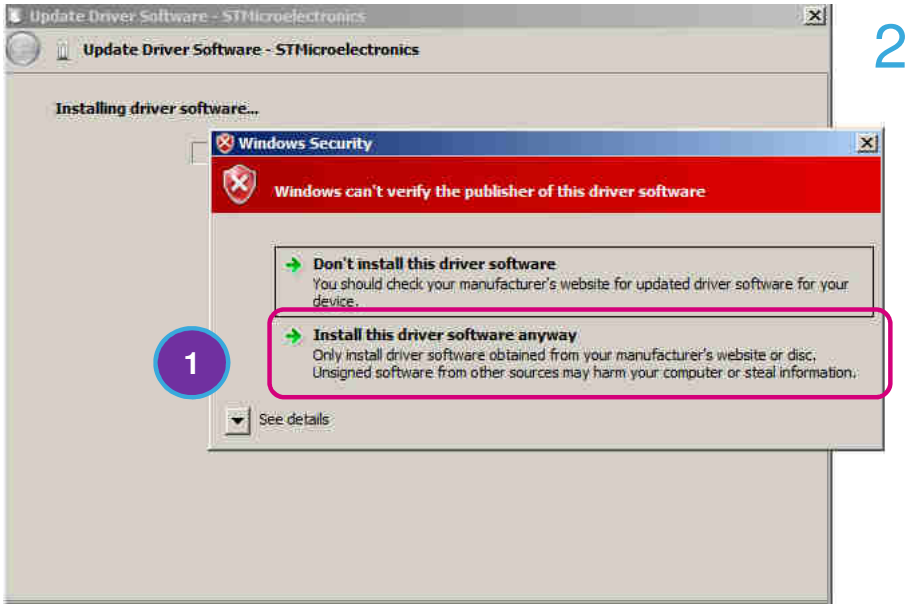


On the PC file system a BCN002V1 drive will appear

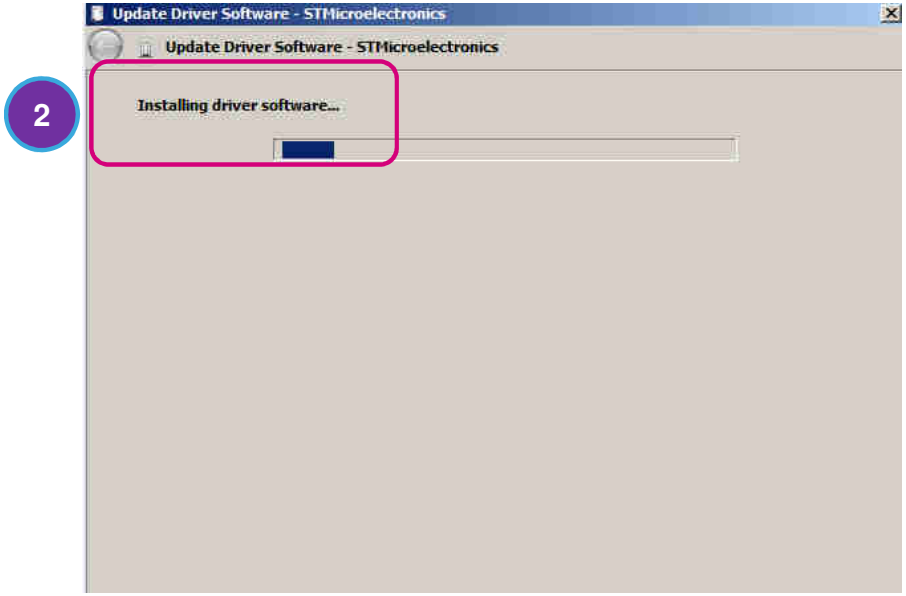
life.augmented

Windows7: Allow the driver installation

1. Click on **Install driver software anyway**



2. Installation starts



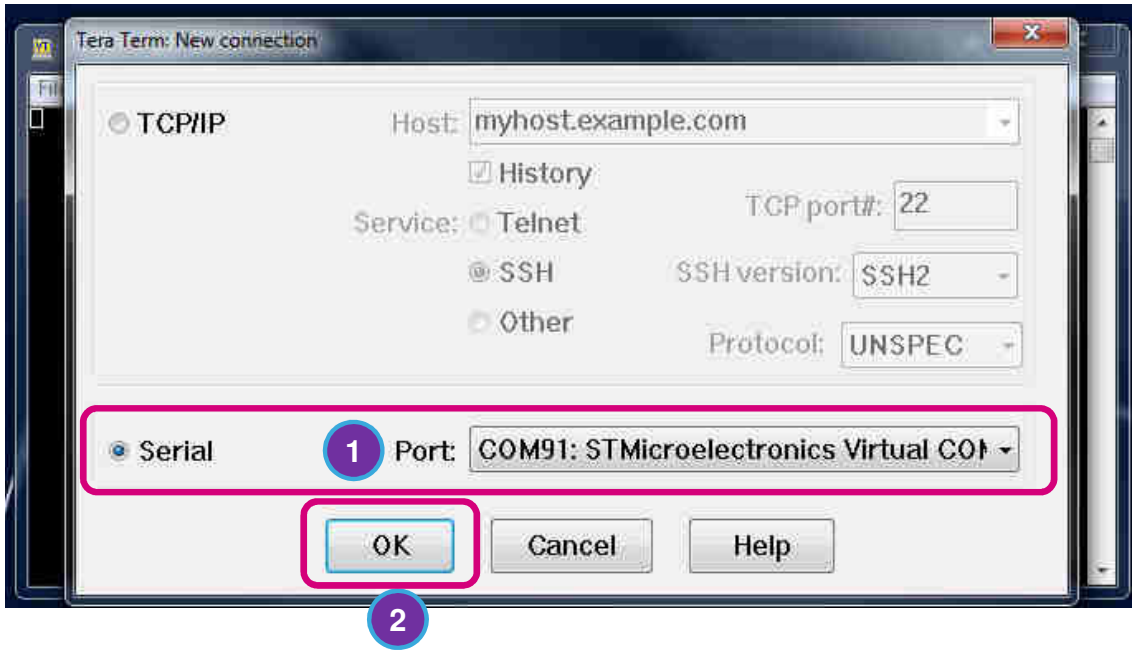
life.augmented

Virtual COM port driver

If you have issue with the **STMicroelectronics Virtual COM Port** device driver installation here the **instructions** for installing the Virtual COM port driver:

Win7 **Win10**

Tera Term Configuration 1/4



1. **Select** the **STMicroelectronics Virtual COM Port**
   - **NOTE:** on **Win10 PC** the serial port is labeled just as **"COMxx"**
2. Click **OK**

life.augmented

Tera Term Configuration 2/4

1. In **Setup** -> **Serial port…**
   Set the following:

   Baud rate : 115200

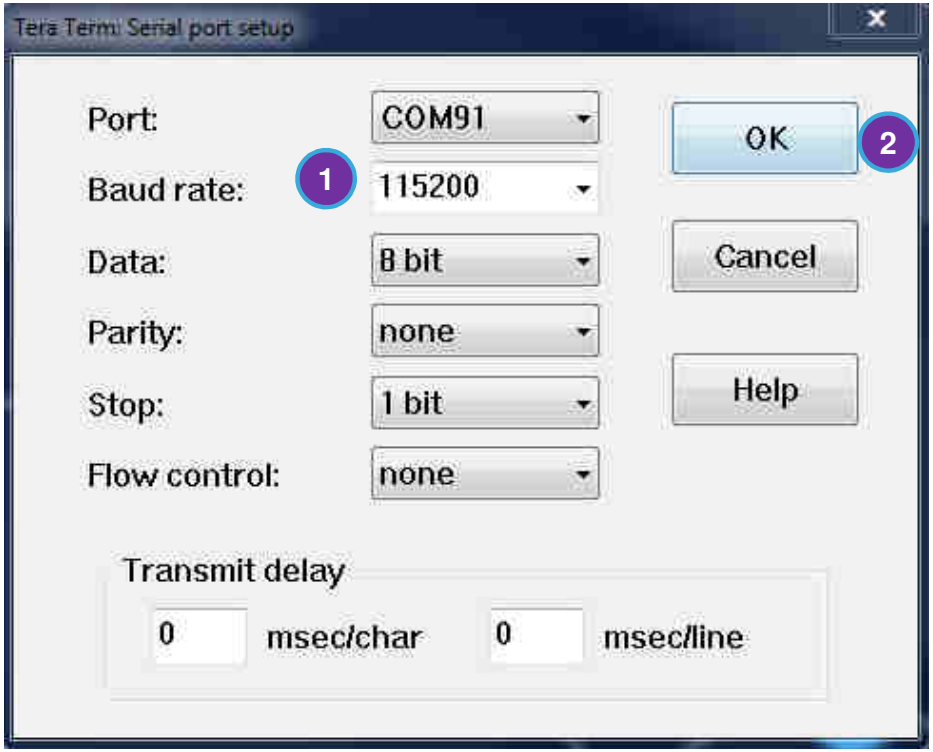
   Data : 8 bit

   Parity : none

   Stop : 1 bit

   Flow control : none

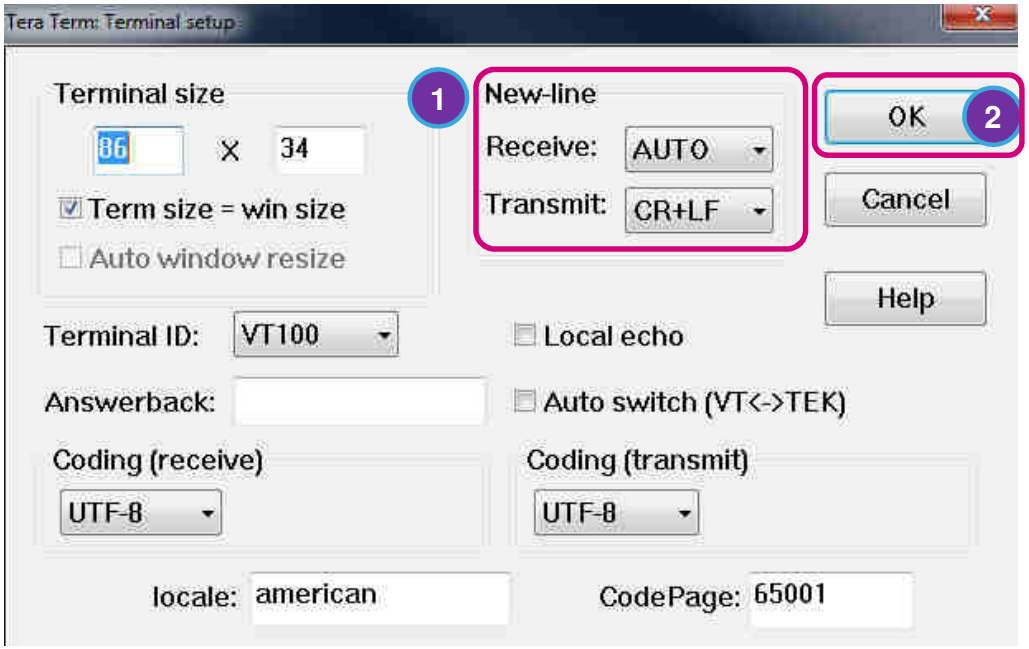
2. **Click OK**



life.augmented

Tera Term Configuration 3/4

1. In **Setup** -> **Terminal…** set **New-line Receive : AUTO**
2. **Click OK**

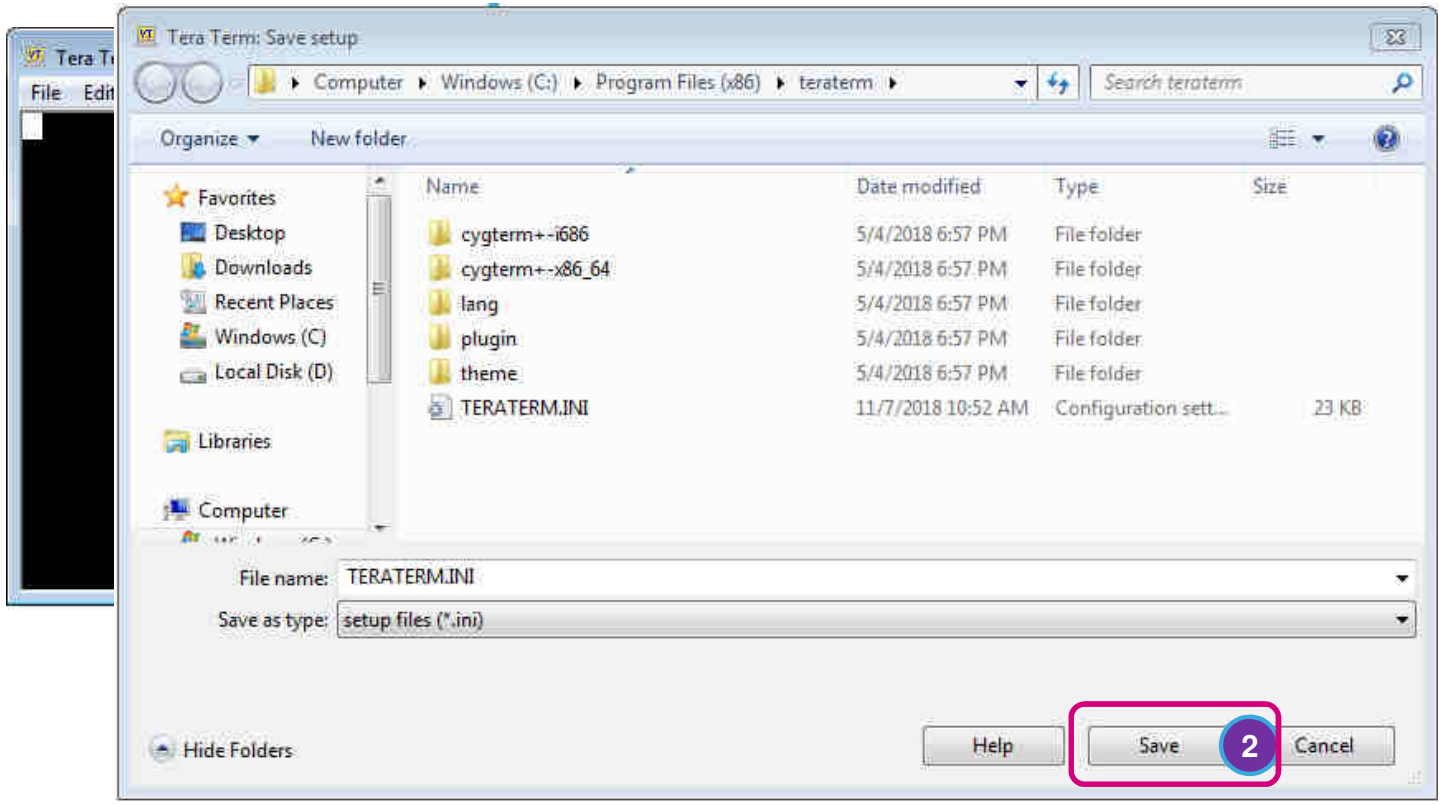


life.augmented

Tera Term Configuration 4/4

1. Click **Setup->Save setup…**

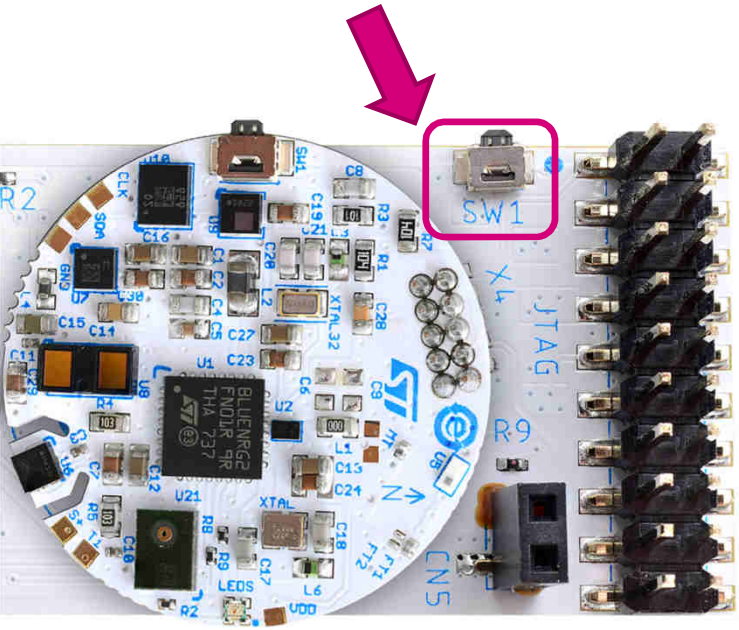
2. Click **Save**

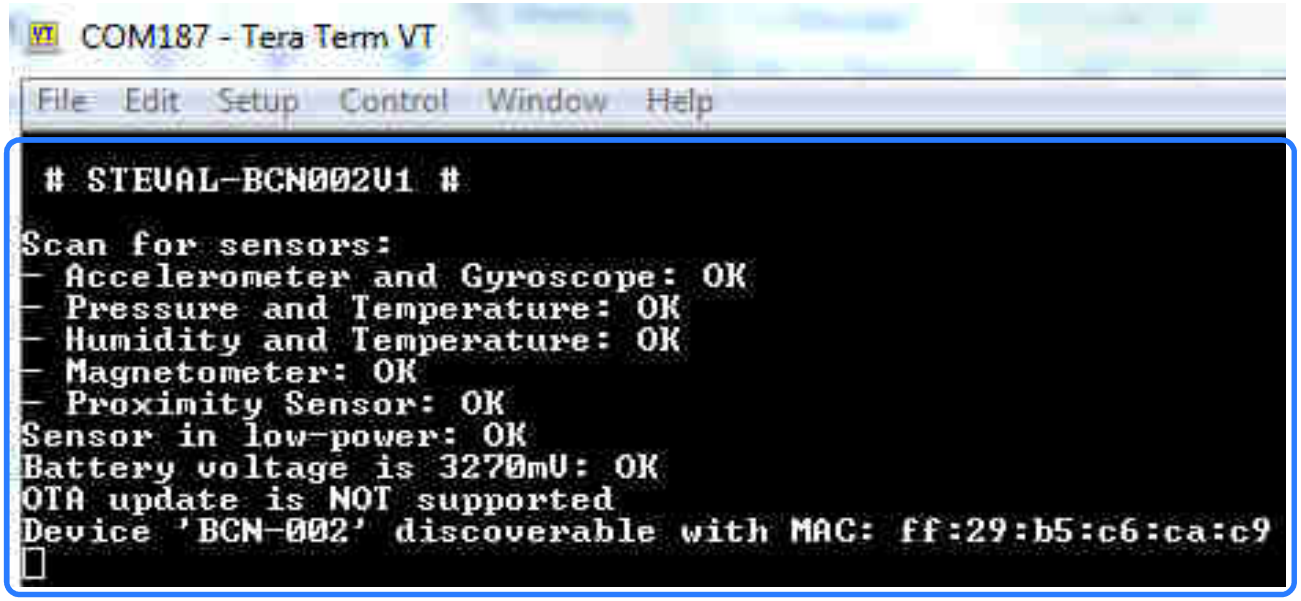


life.augmented

…done!

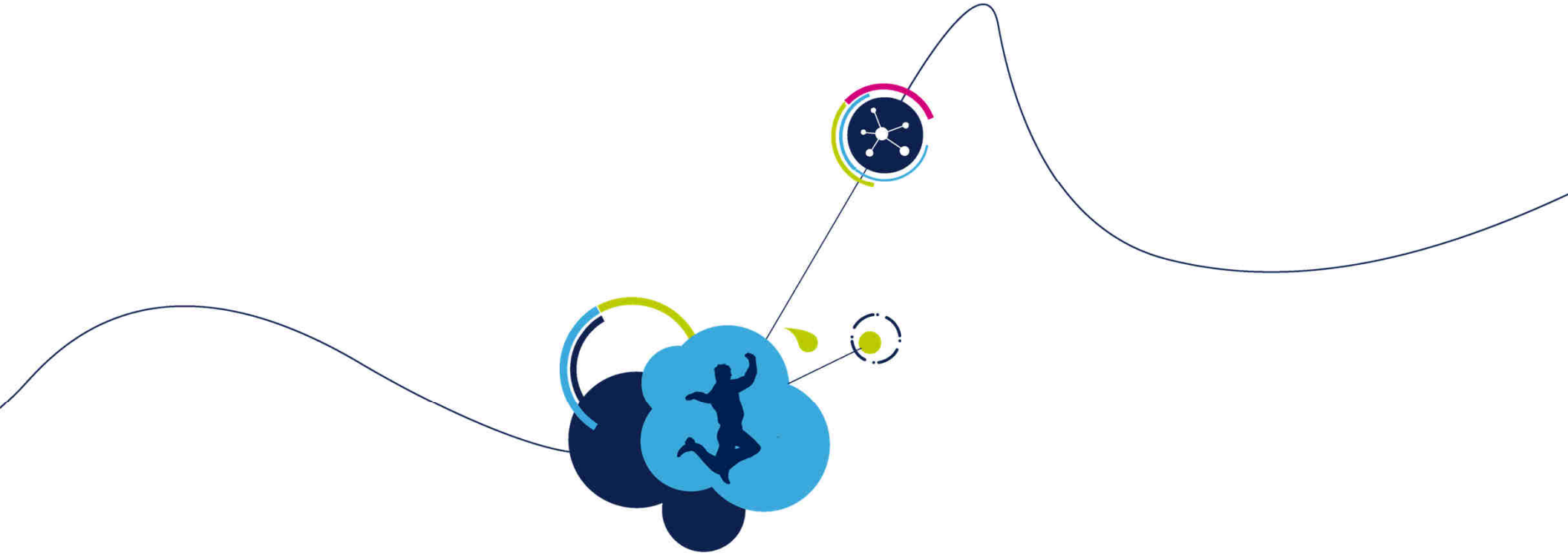
Push SW1 button on the daughterboard





- If you see the output above, **Tera Term** is now properly configured

life.augmented



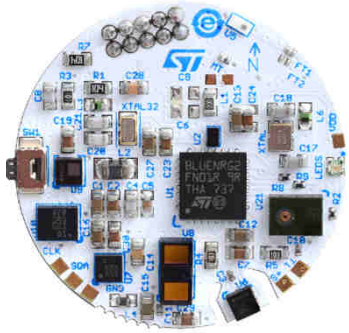
Lab 2

Connect to the ST BLE Sensor app

life.augmented

Customize *YOUR* STEVAL-BCN002V1

+ STEVAL-BCN002V1 DK
Sensor Demo ref FW

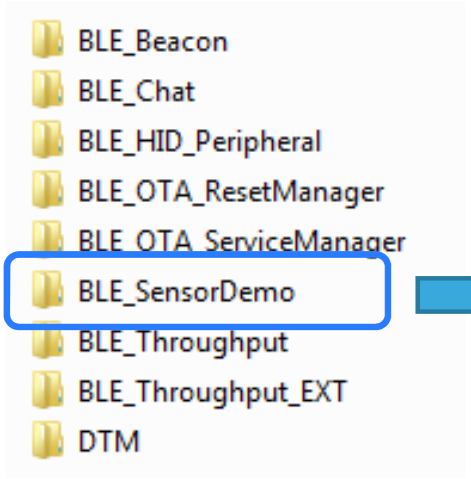


Client

life.augmented

BLE_SensorDemo application

1. In the BlueTile SDK browse the following path: *STEVAL-BCN002V1 DK 1.1.0\Project\BLE_Examples*



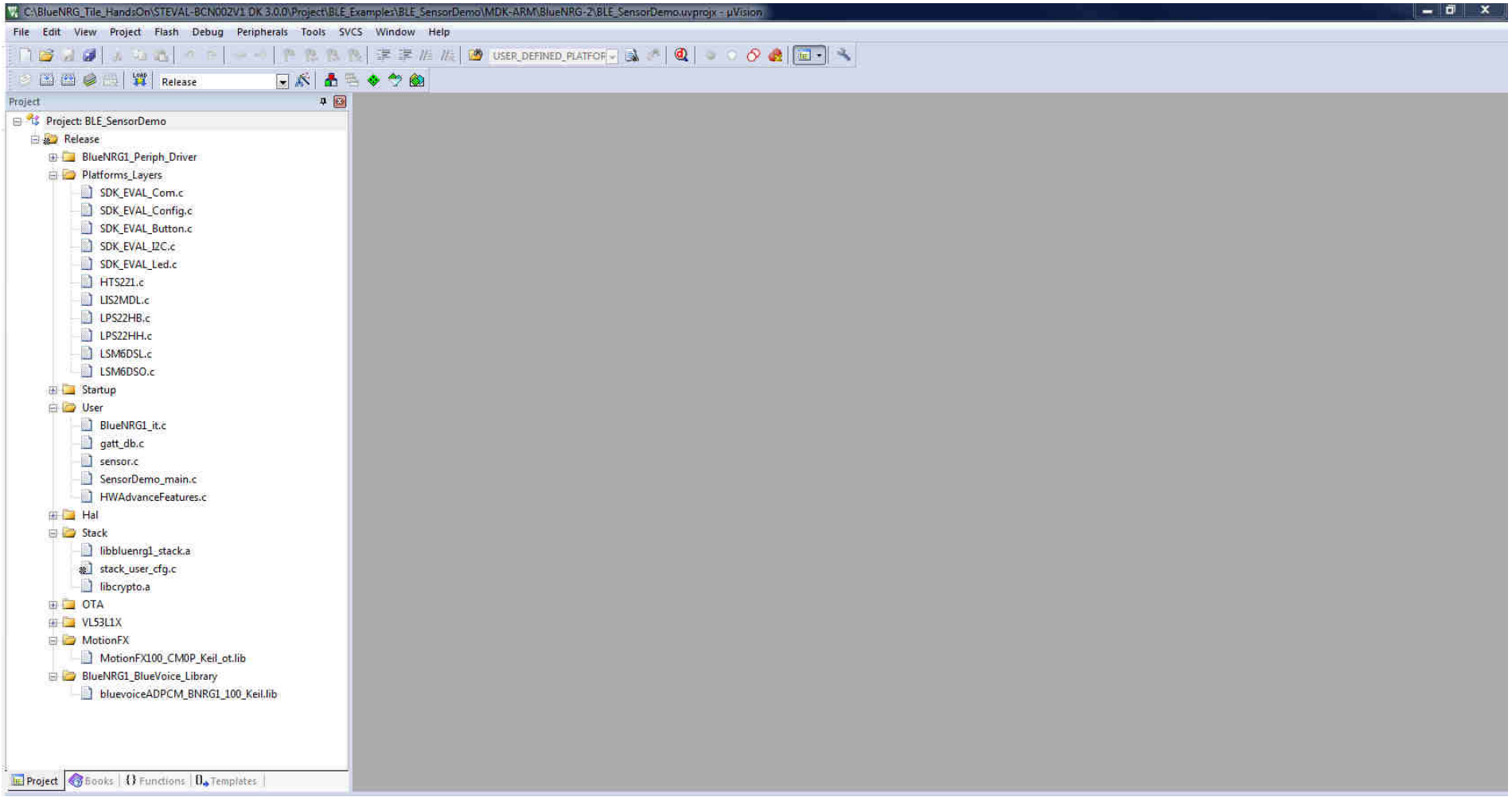
The BLE_SensorDemo embedded application is the main reference

Integrated Development Environment: Arm Keil MDK

- **MDK**: **M**icrocontroller **D**evelopment **K**it
  - Complete software development environment from Arm for a wide range of Arm Cortex-M based microcontroller devices.
  - MDK includes the **µVision IDE**, **debugger**, and **Arm C/C++ compiler**

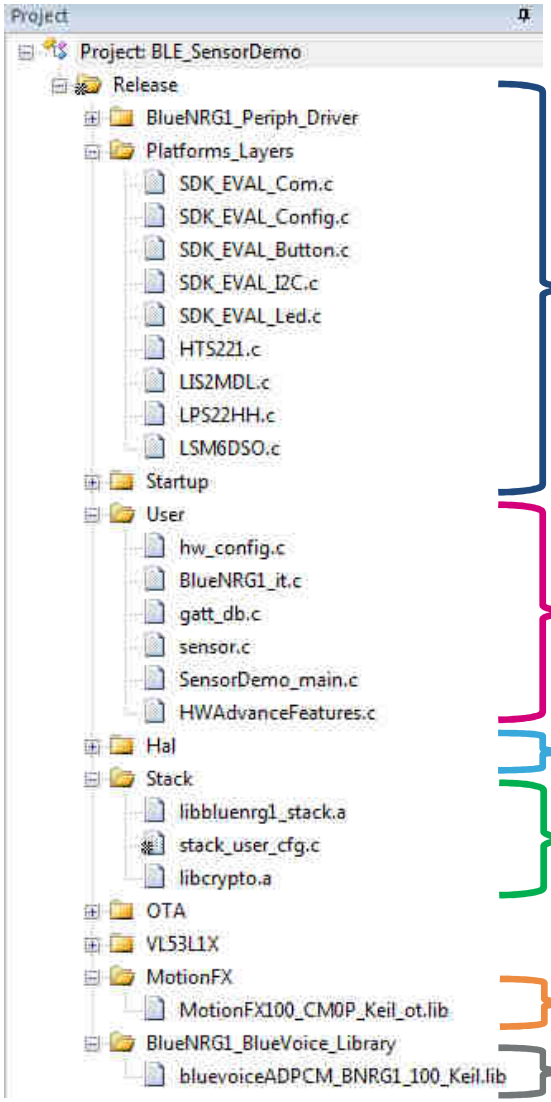
- A **FREE** license is available for ST BlueNRG-2
  - Go **HERE** to get the free license!

BLE_SensorDemo application



life.augmented

Application structure



HW peripherals drivers and platform layer files

Application source code
- **Main**
- **ATTRIBUTE Database**
- **Application**

Hardware Abstraction Layer and sleep management

BLE Stack library – provided in binary format

Sensor Data Fusion library

Voice over BLE library

life.augmented

A look at the main application


```
int main(void) {
```

`SystemInit();` Remap the vector table and configure all the interrupts priority

`SdkEvalIdentification();` Identifies STEVAL or custom PCB

`PlatformInit();` HW peripherals initialization

`BlueNRG_Stack_Initialization(&BlueNRG_Stack_Init_params);` BLE stack initialization

`Sensor_DeviceInit();` Sensors initialization

`Set_DeviceConnectable();` Set device in advertising

`while(1){` Start of while loop

`BTLE_StackTick();` Advances the stack FSM

`User_AppTick();` Advances the application FSM. **THIS IS DEVELOPERS USER SPACE!**

```
  } // end while(1)
}
```

life.augmented

BLE flow on the LAB

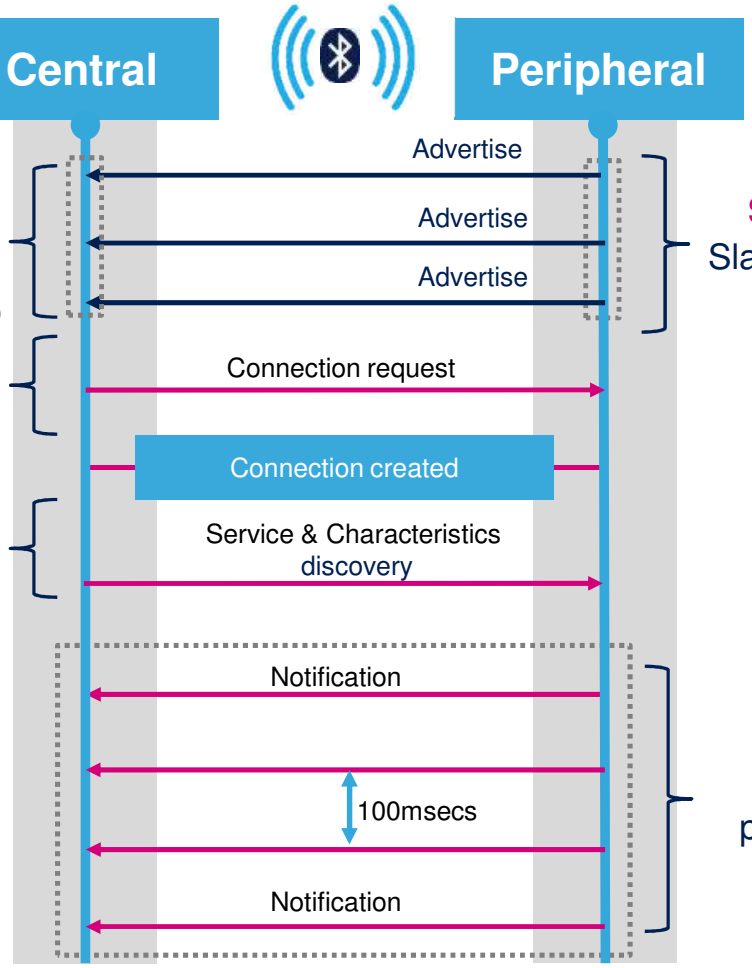
Step 2: ***Scan***

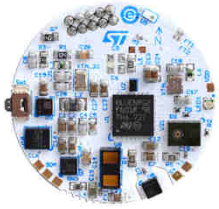
Master is in discovery mode looking for a ***specific*** slave to connect to

Step 3: ***Connection request***

Step 4: ***Services & Chars discovery***

Master (Client) starts the BLE Service and Characteristics discovery procedure to understand the Server ATT DB





Step 1: ***Advertising***

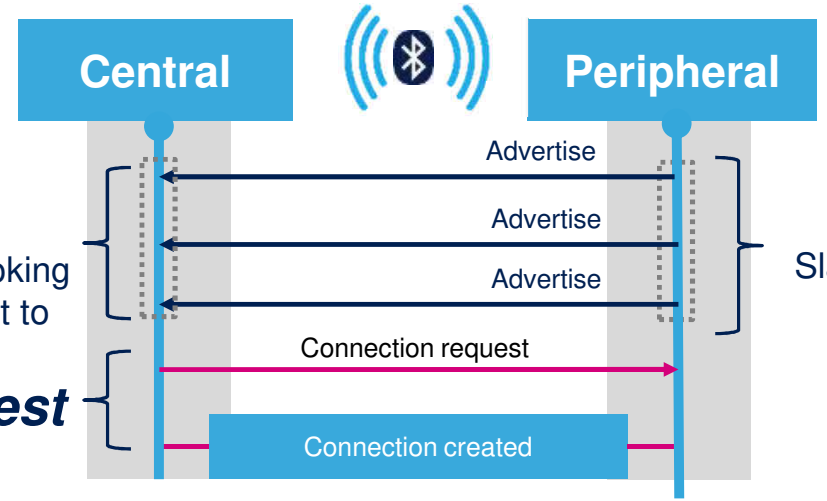
Slave is in Peripheral mode and sends ADV_IND PDU

Step 5: ***Data flow***

Slave (Server) starts sending periodically (100 ms) to the master, **notifications packets** of sensors values (acc&gyro and pressure)

life.augmented

Advertising and scanning



Step 2: ***Scan***
Master is in discovery mode looking for a ***specific*** slave to connect to

Step 3: ***Connection request***

Step 1: ***Advertising***
Slave is in Peripheral mode and sends ADV_IND PDU

Master:
needs an app for discovering the slave device in advertising

Off-the-shelf app: e.g.

LightBlue® Explorer

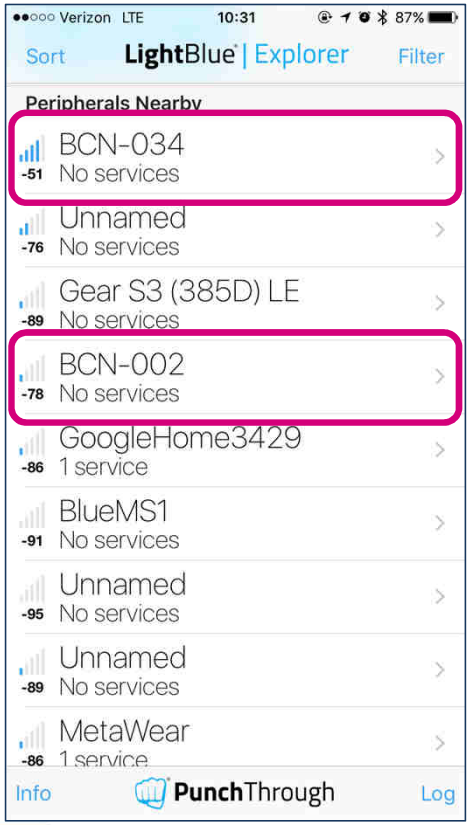
https://itunes.apple.com/us/app/lightblue-explorer/id557428110?mt=8

Custom app: e.g.

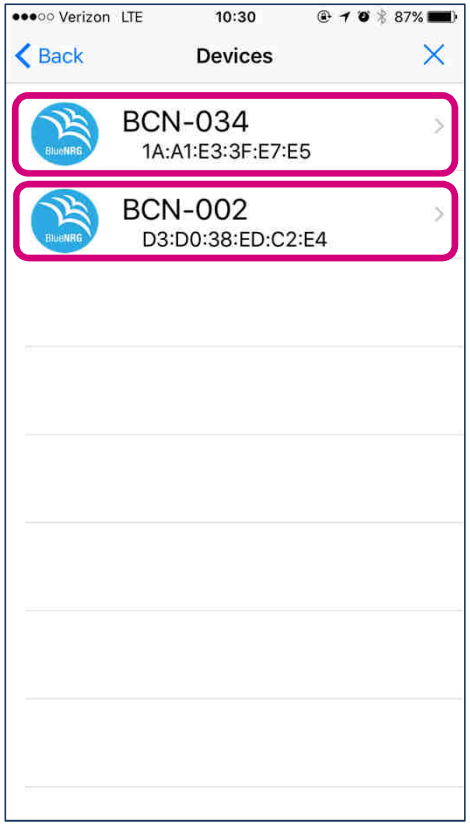
life.augmented

Scan results

LightBlue scan results



ST BLE Sensor scan results



Why some devices are not present in the ST BLE Sensor app?

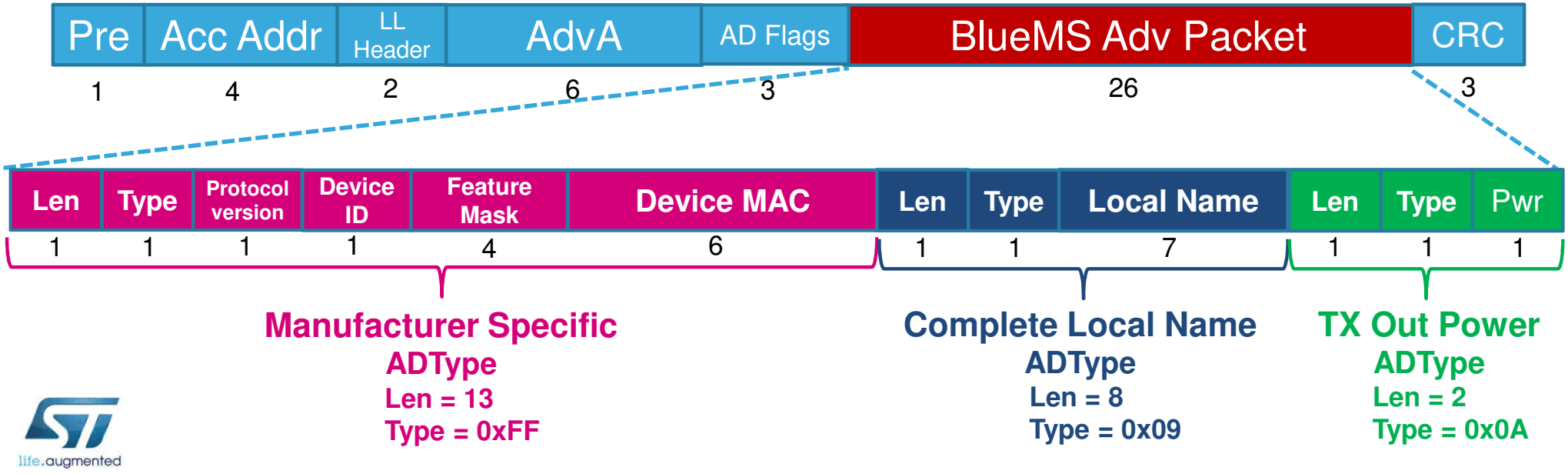
How devices will pop up in the ST BLE Sensor scan list?

life.augmented

ST BlueMS Protocol

- In order to be connected to the **ST BLE Sensor** app, a BLE peripheral SHALL **comply** with a specific advertising packet format

- **ST BlueMS protocol** specifies a **26-byte packet format** composed of Advertisement Types – **ADTypes** - compliant with BT SIG definitions



life.augmented

BlueST SDK for Android and iOS

Android

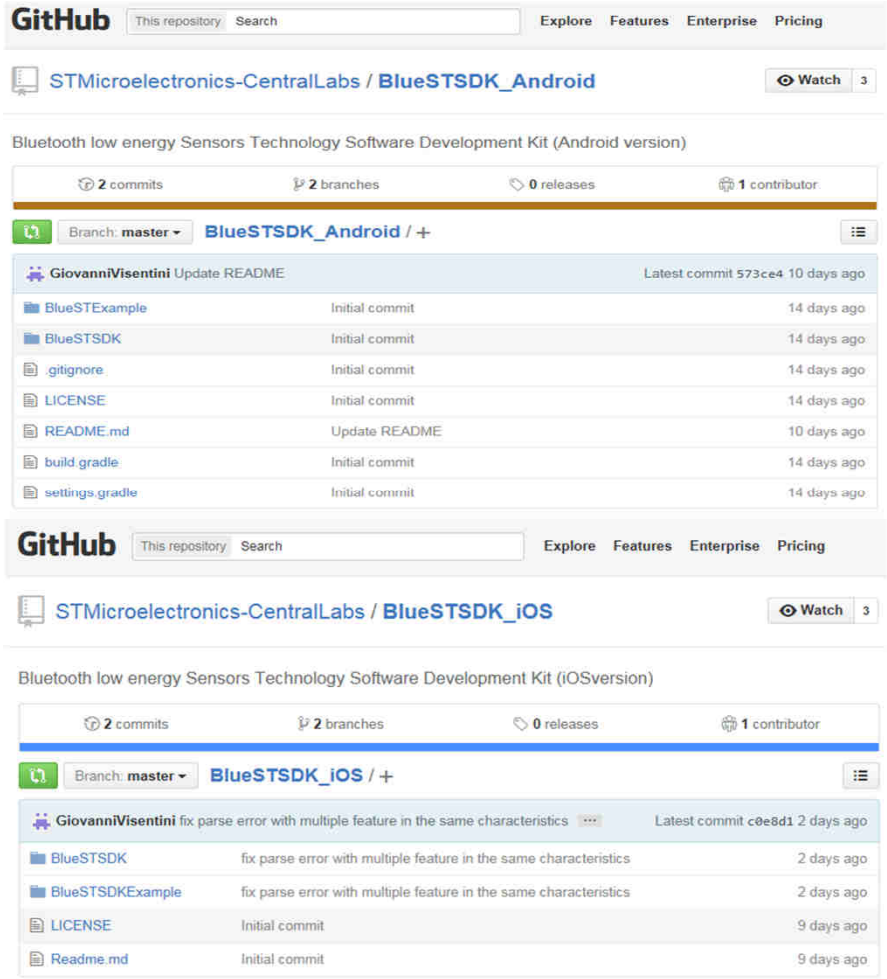
https://github.com/stmicroelectronics-centrallabs/bluestsdk_android

Source code for iOS and Android

available online and maintained on GitHub repos

iOS

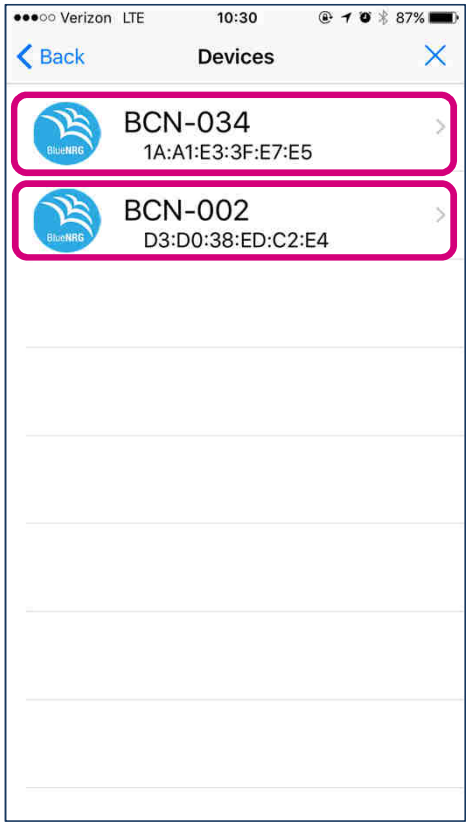
https://github.com/stmicroelectronics-centrallabs/bluestsdk_ios



life.augmented

ST BLE Sensor Scan results

Len	Type	Protocol version	Device ID	Feature Mask	Device MAC	Len	Type	Local Name



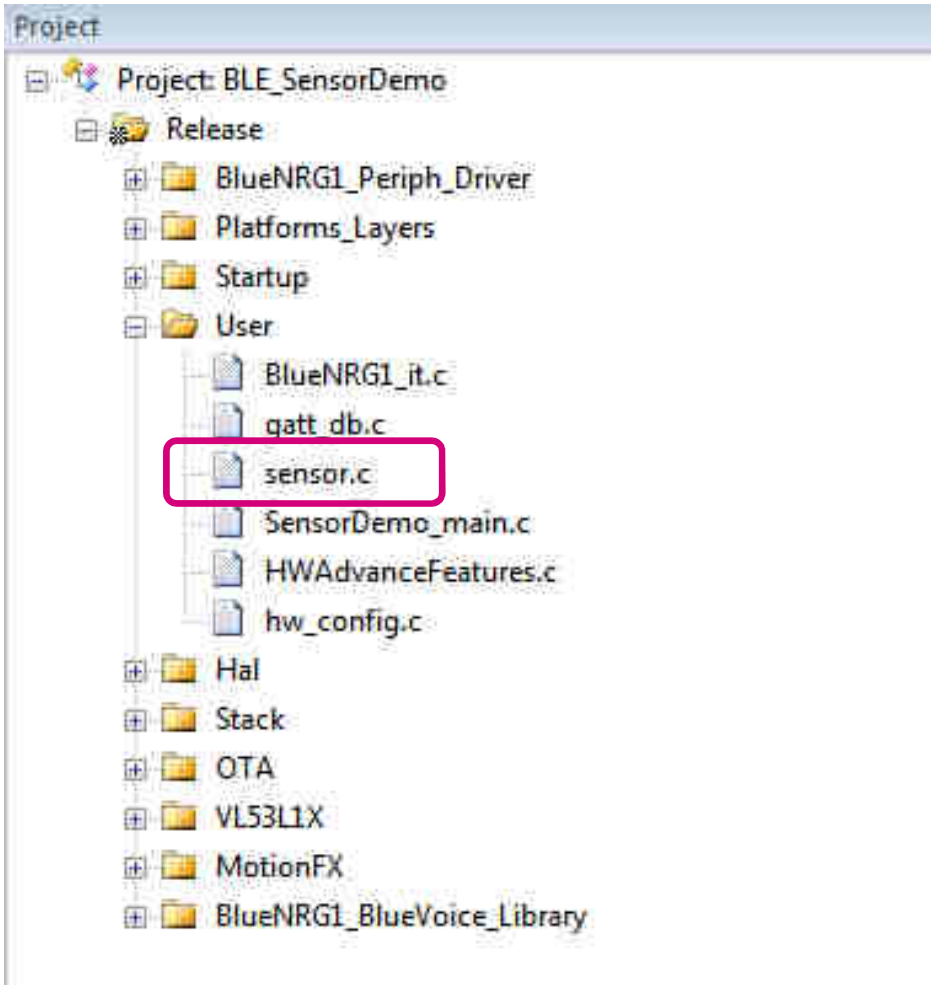
Each ST Platform is recognized by means of a Device ID

(for STEVAL-BCN002V1 it is equal to 0x05)

Users can distinguish their node from the Local Name

life.augmented

Customize your BlueTile



Customize your BlueTile

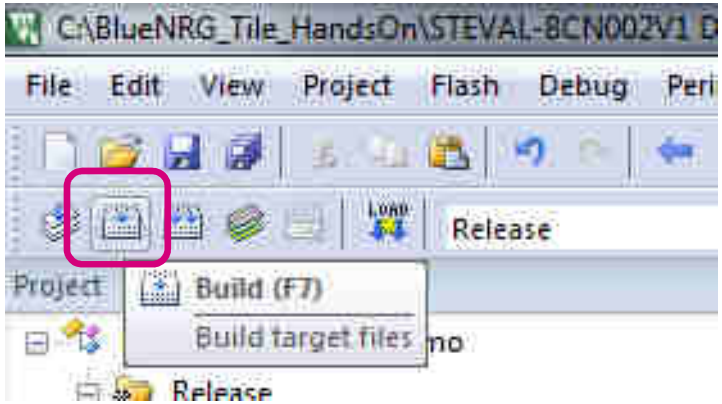
Modify the **local name** in the advertisement payload

1. In the file **sensor.c** **go to** **line 52**
2. **“BCN-002” is the default local name value**
3. **You can modify it as you prefer with a 7-characters string**

```
52  #define DEFAULT_NAME_ALLMEMS        'B','C','N','-','0','0','2'
```

Build the new code

1. Click on the **Build button** (top left corner) or hit **F7** on your keyboard
2. In the **Build Output** window (bottom) wait for the build to be completed.
   - **BLE_SensorDemo.bin** created
   - "**0 Error(s), 0 Warning(s)**" message appear




```
Build Output
compiling vl53l1_wait.c...
compiling vl53l1_register_funcs.c...
compiling vl53l1_platform.c...
linking...
Program Size: Code=121908 RO-data=1428 RW-data=1136 ZI-data=21252
FromELF: creating hex file...
After Build - User command #1: fromelf.exe --bin ..\..\..\..\..\..\HandsOn\BLE_SensorDemo_PrjOutput\BLE_SensorDemo.axf --output ..\..\..\..\..\..\HandsOn\BLE_SensorDemo_PrjOutput\BLE_SensorDemo.bin
"..\..\..\..\..\..\HandsOn\BLE_SensorDemo_PrjOutput\BLE_SensorDemo.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed:  00:00:12
```

life.augmented

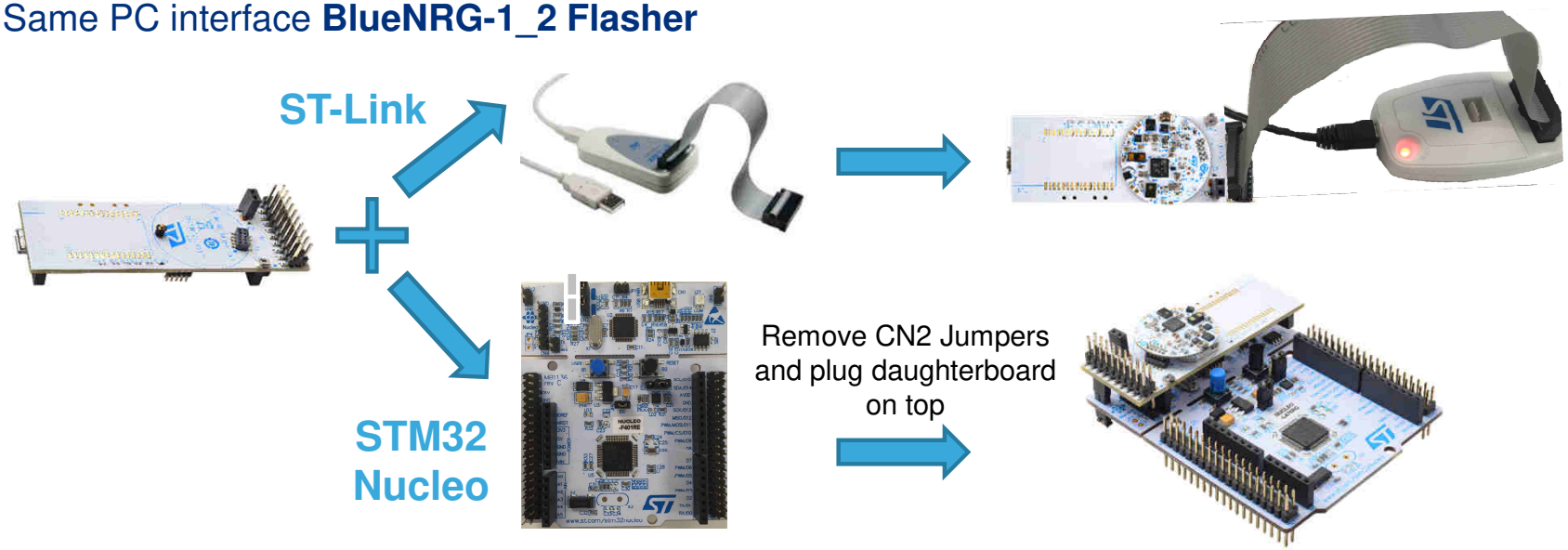
Programming embedded Flash

1. UART Bootloader

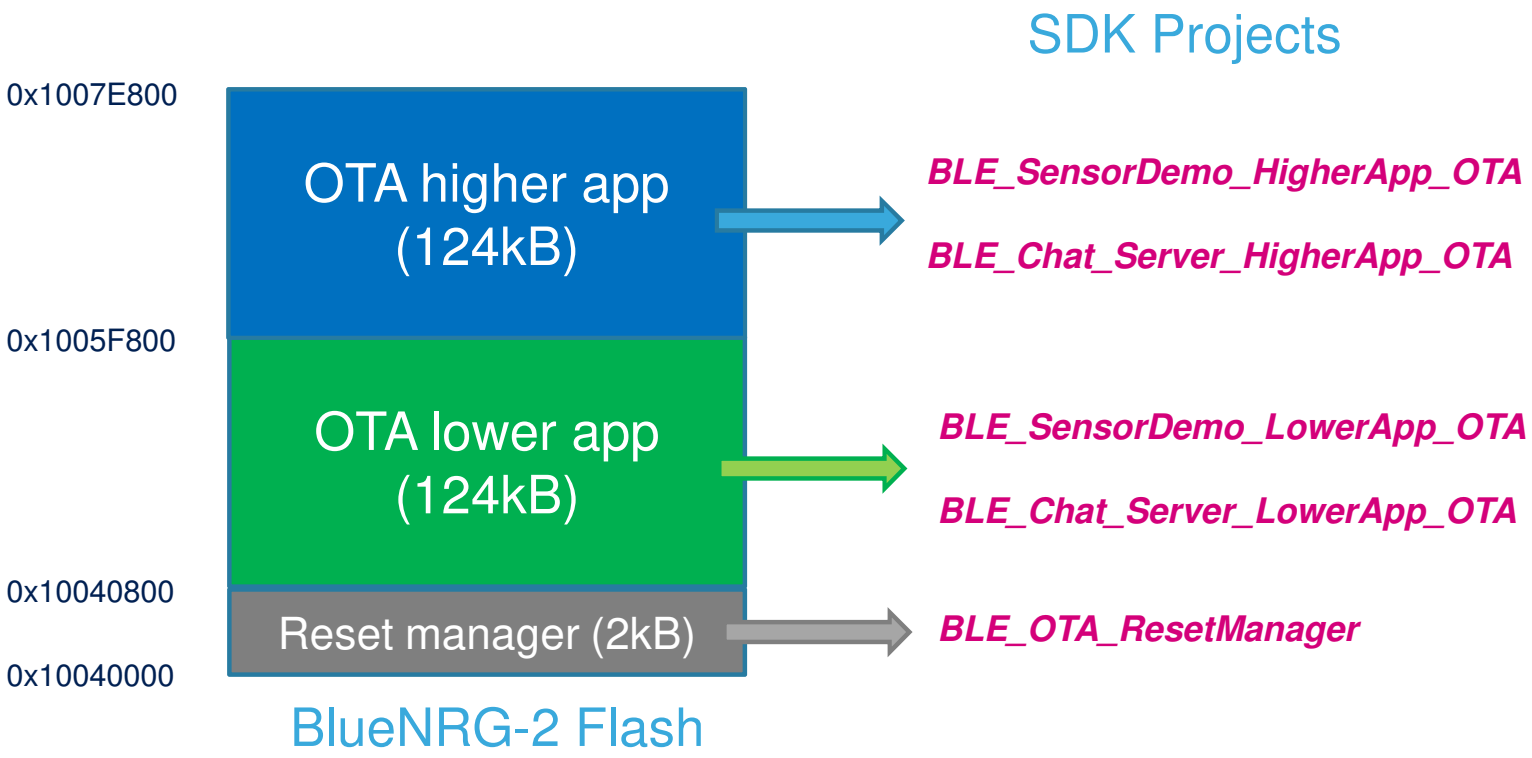
- **ROM bootloader**. HW **activation through dedicate pin** (**DIO7**) configured for Boot
- **PC interface** named **"BlueNRG-1_2 Flasher Utility"** available in the SW package

2. SWD interface + ST-LINK

- Interface with the STEVAL-BCN002V1 through the **20-pin JTAG connector**
- Same PC interface **BlueNRG-1_2 Flasher**

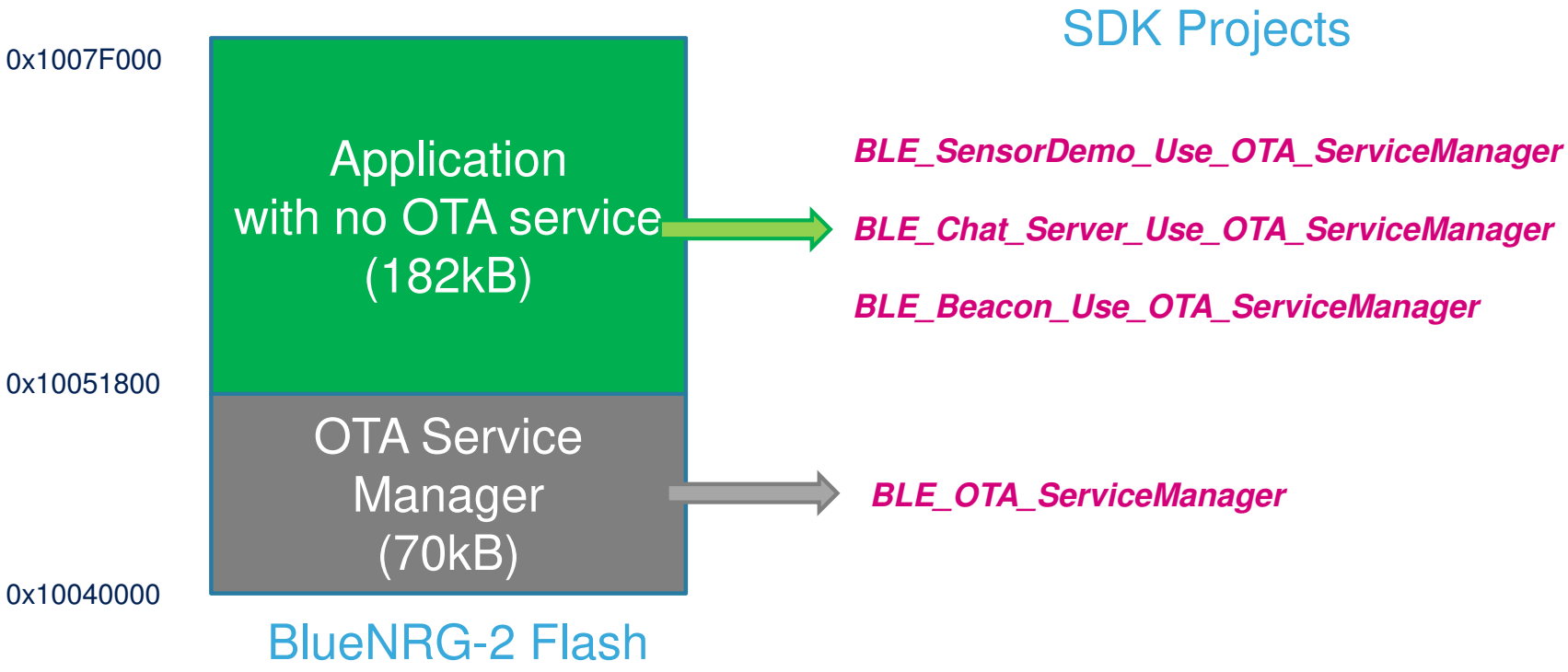


Flash memory layout 1/2



life.augmented

Flash memory layout 2/2

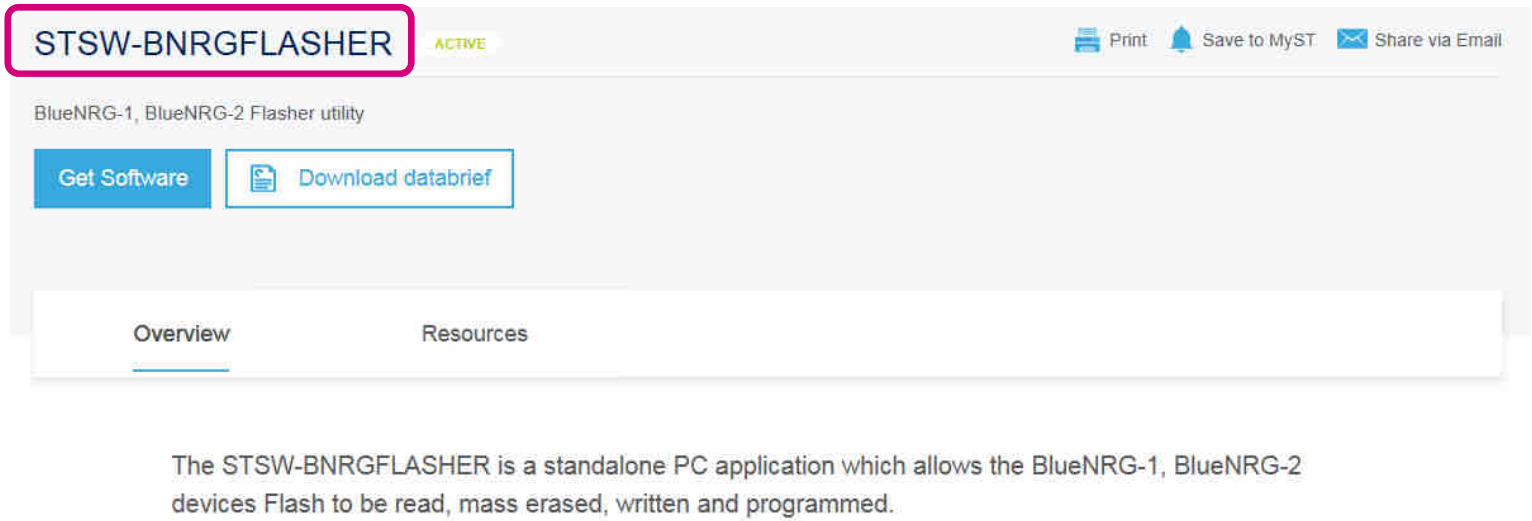


life.augmented

UART Bootloader: use BlueNRG-2 Flasher Utility

- On st.com at this link:

  https://www.st.com/content/st_com/en/products/embedded-software/wireless-connectivity-software/stsw-bnrgflasher.html



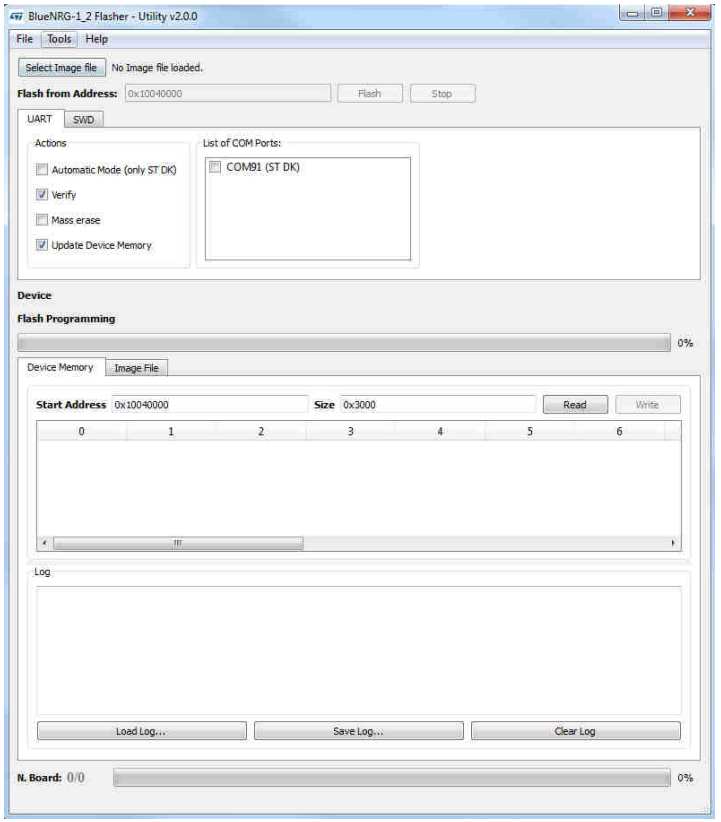
life.augmented

Open the Flasher Utility

1. **Go to**

   C:\Program Files (x86)\STMicroelectronics\BlueNRG-1_2 Flasher Utility 3.0.0\Application

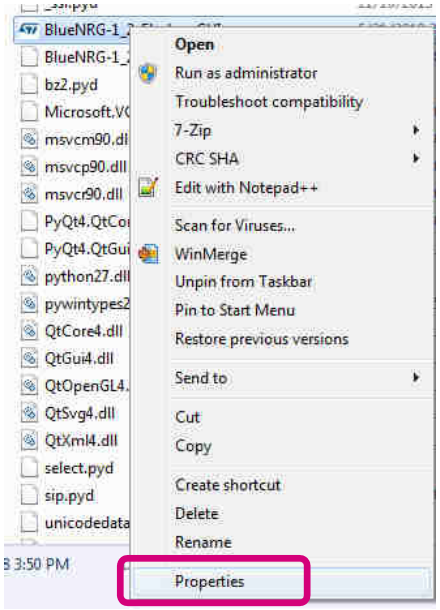
1. **Double click** on **BlueNRG-1_Flasher_GUI.exe**

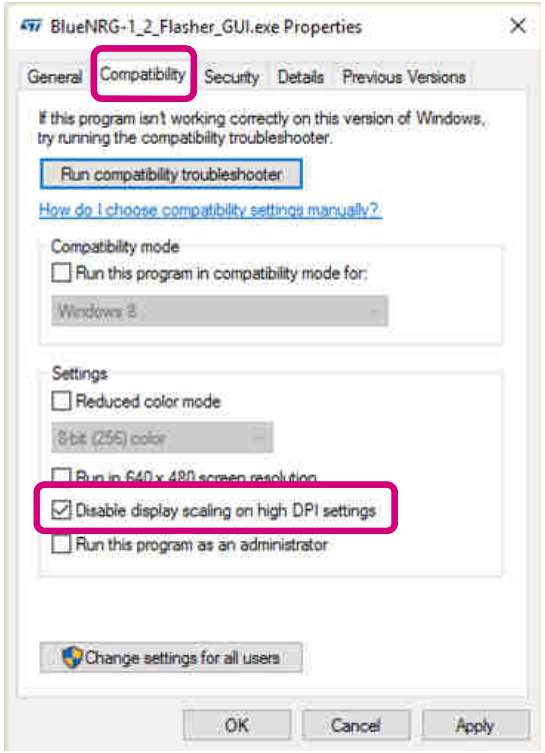


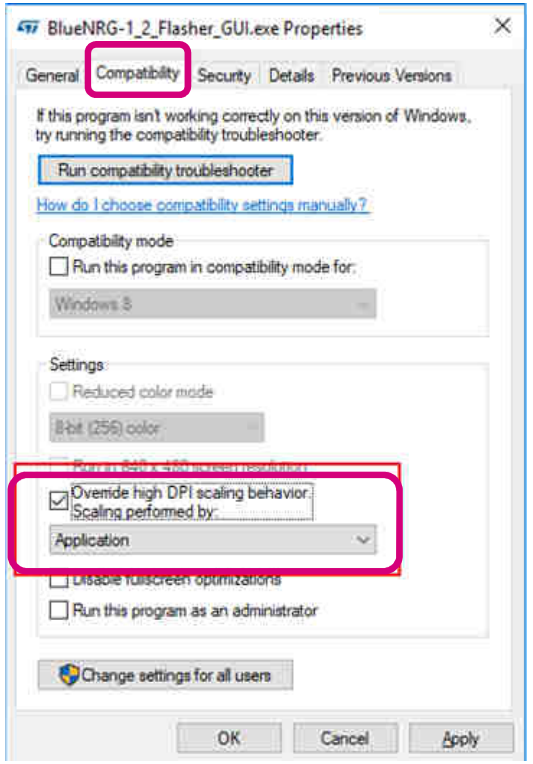
life.augmented

For HD screens…If you've problem with the resolution

- **Right Click** on the .exe file and **select** **Properties**
- In the **Compatibility** tab **select** either **Disable display scaling…** or **Override high DPI scaling… –> Application**

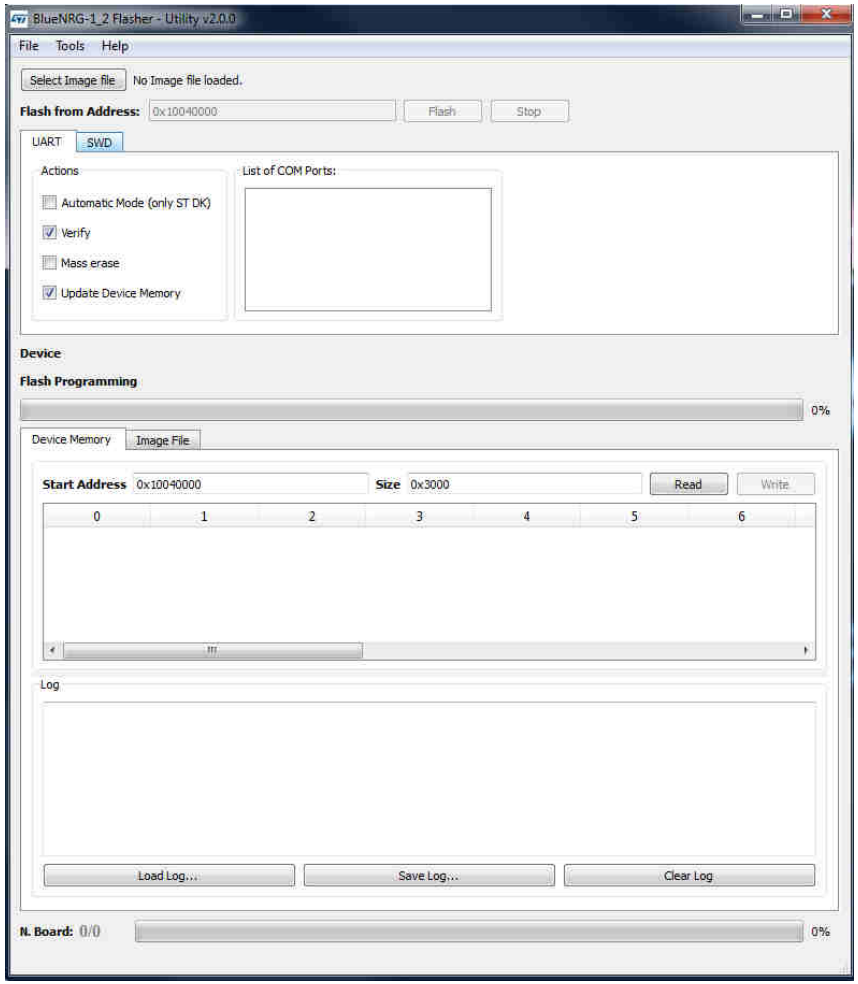




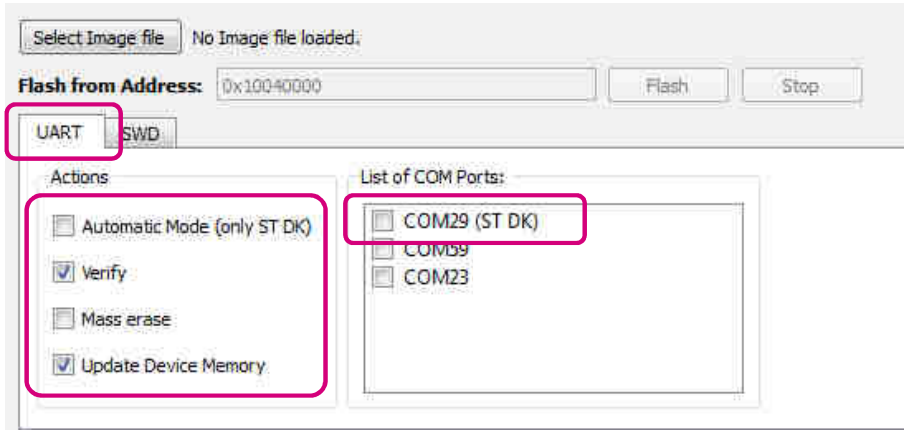


life.augmented

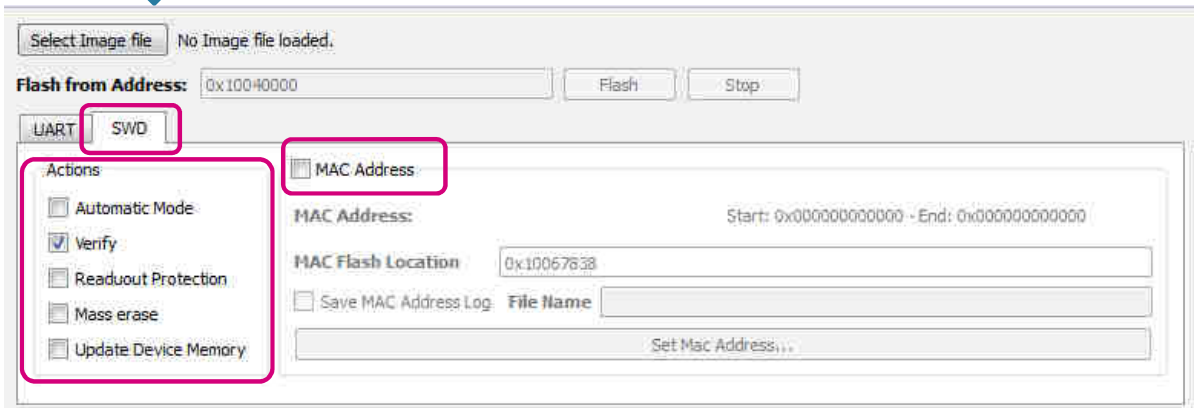
BlueNRG Flasher Utility



UART

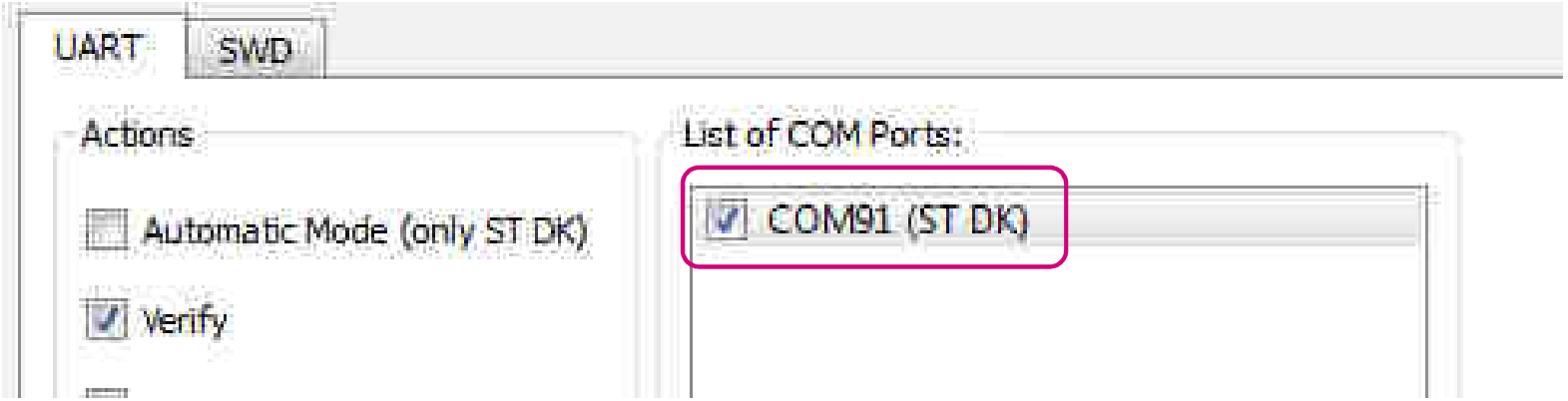


SWD



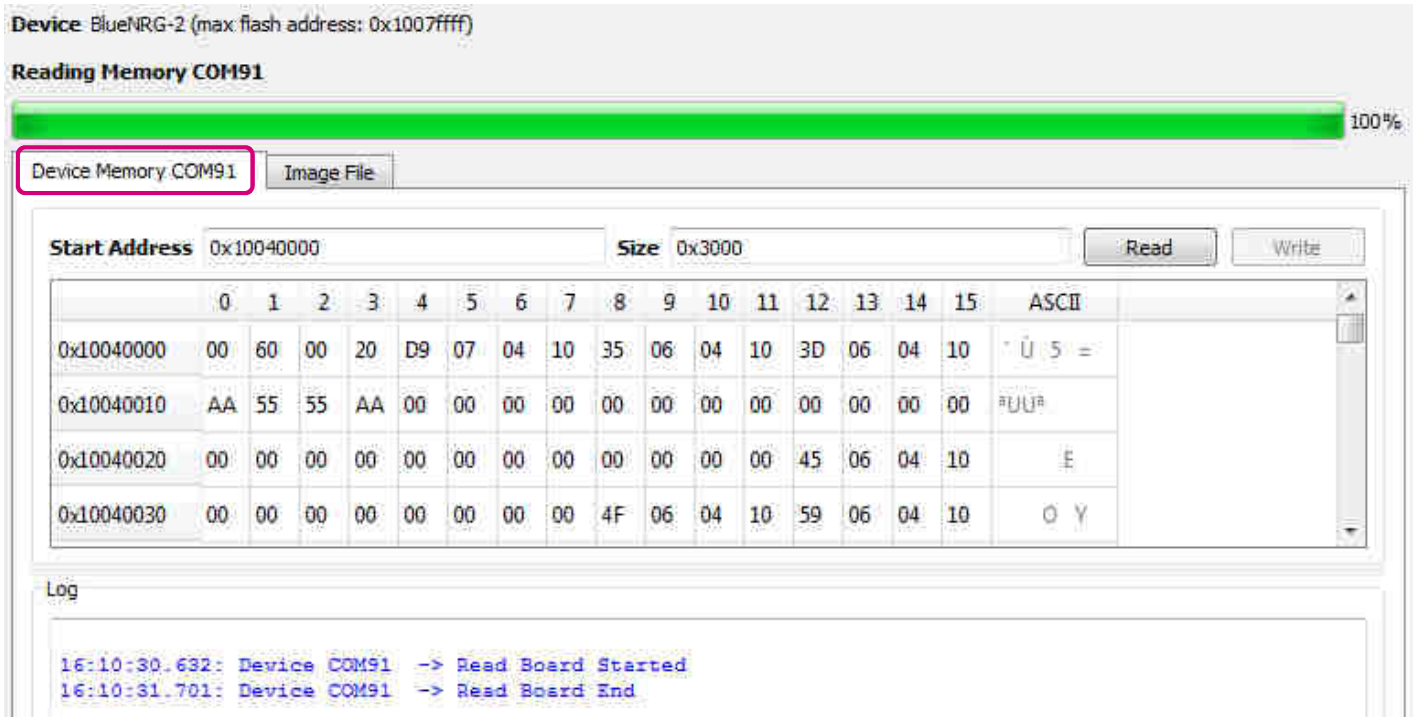
Flash the BlueNRG-2 1/6

1. **Select the COM port** labeled **(ST DK)**



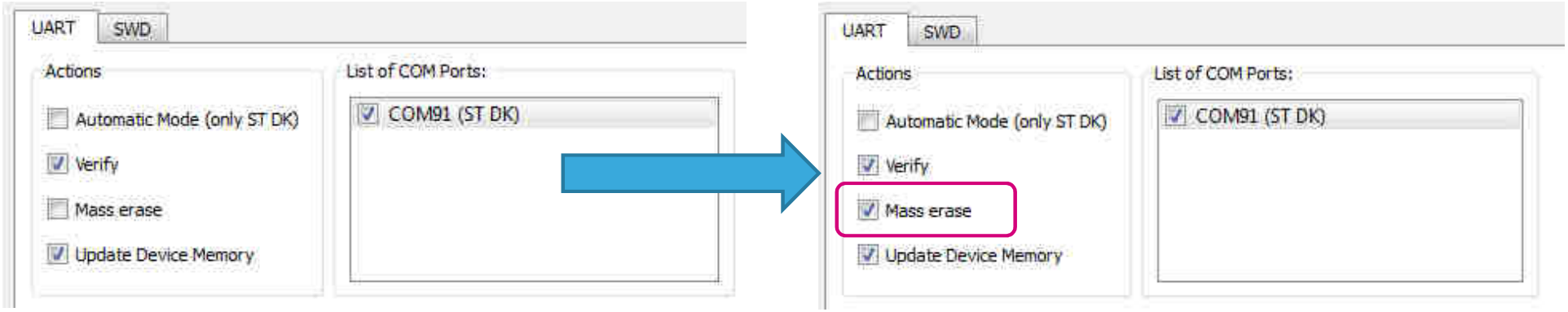
Flash the BlueNRG-2 2/6

1. **Device Memory** will populate with data



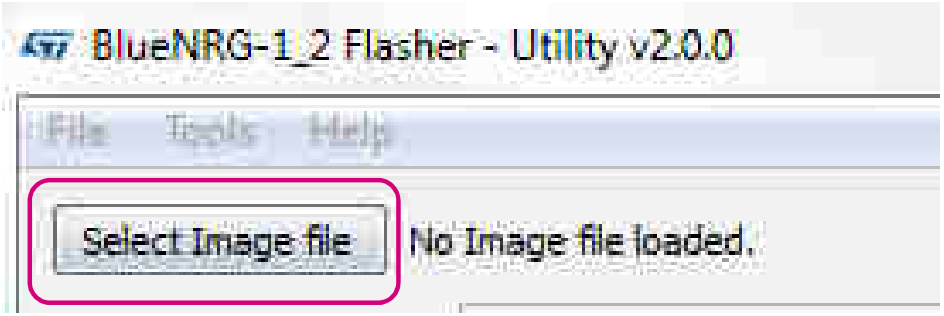
Flash the BlueNRG-2 3/6

1. Select **Mass Erase**



Flash the BlueNRG-2 4/6

1. **Click** on the **Select Image file** button



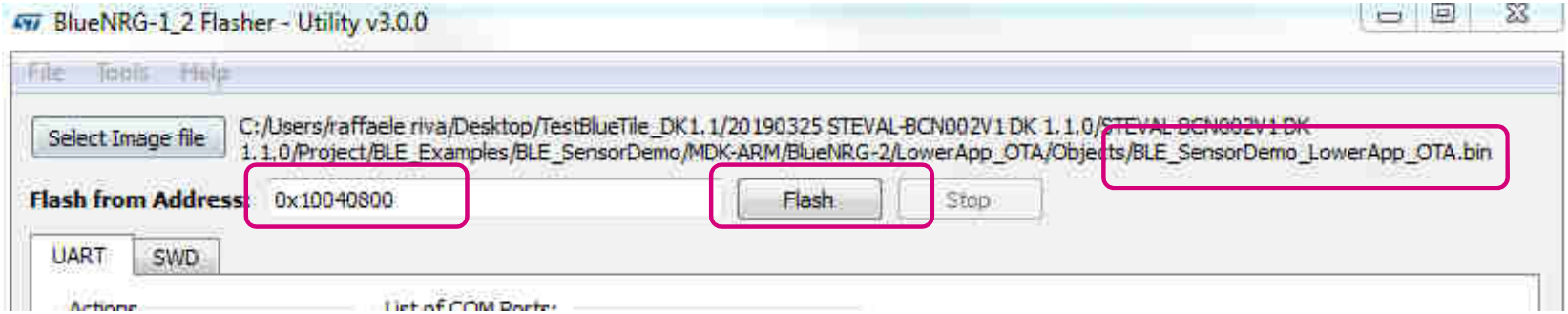
2. And **browse** the following **path**

life.augmented

Flash the BlueNRG-2 5/6

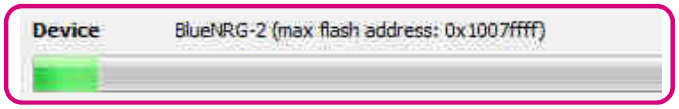
1. **Select** the binary file (e.g. BLE_SensorDemo_LowerApp_OTA.bin) and **click** **Open**

2. Insert the correct **start address** (e.g. 0x10040800 for Lower App)
3. **Click** on the **Flash** button

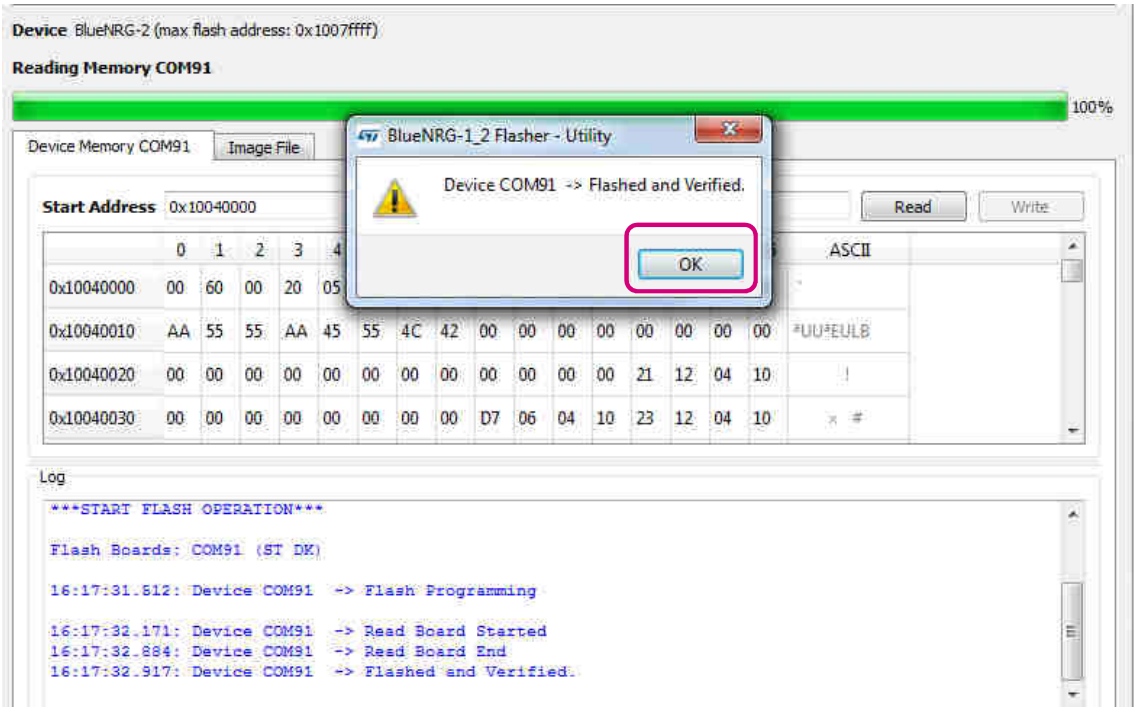


Flash the BlueNRG-2 6/6

1. Flashing starts: **green bar** proceeding



2. **Wait** for the **pop-up** window and **click** on **OK**



SWD: use BlueNRG ST-LINK utility

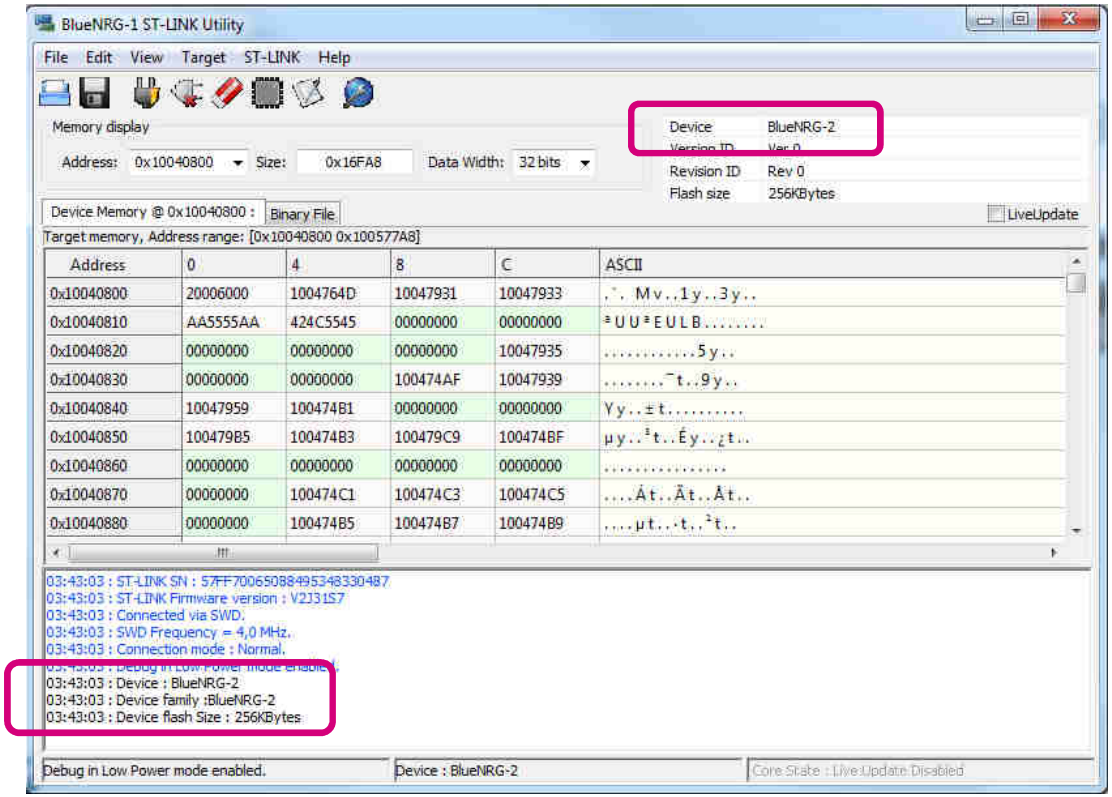
- On st.com at this link:

  https://www.st.com/content/st_com/en/products/embedded-software/wireless-connectivity-software/stsw-bnrg1stlink.html

life.augmented

Programming embedded Flash

1. SWD interface + ST-LINK



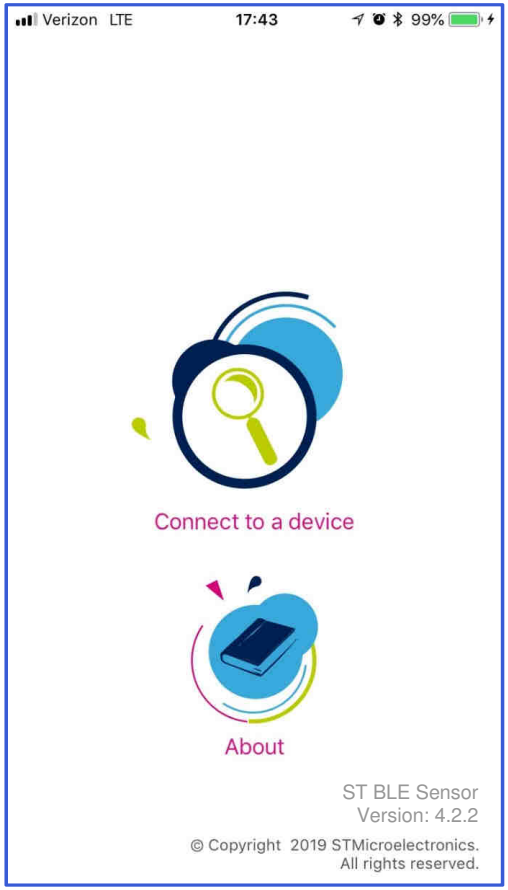
life.augmented

Open the ST BLE Sensor App

Download on the App Store

GET IT ON Google Play

Launch the **ST BLE Sensor** app previously installed



Note: in the following slides all the pictures are referred to the iOS version of the ST BLE Sensor app. The Android version is slightly different

life.augmented

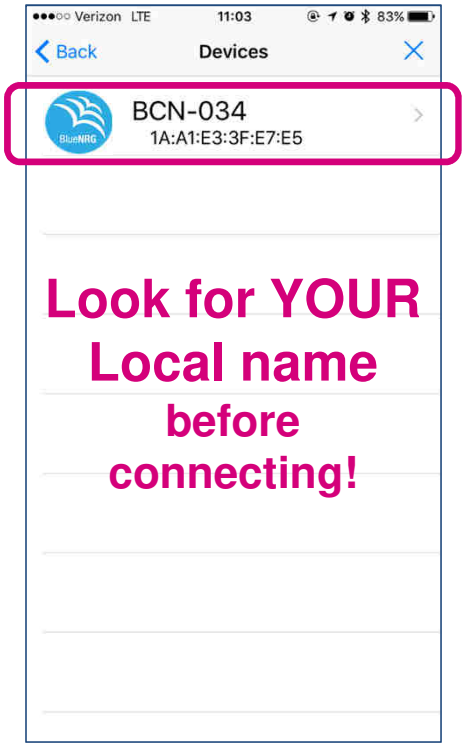
Connect using the ST BLE Sensor App

1

Tap “Connect to a device”

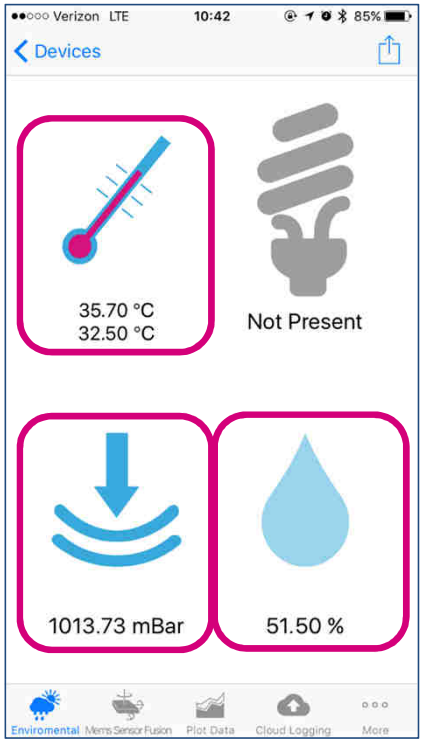
2

Select your BlueTile



3

You are connected

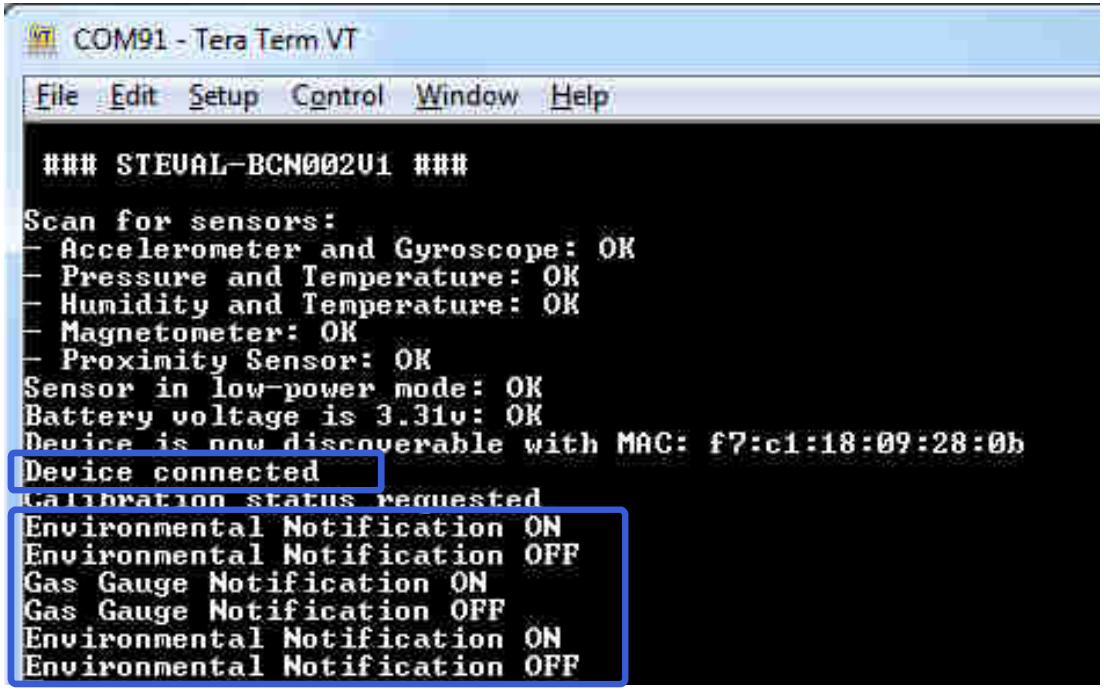


Blow air on the Temperature sensor to change the temperature and humidity values

life.augmented

Tera Term output

- ***"Device connected"*** will appear as connection is created
- Each time user moves on different tabs/screens in the app:
  - **BLE notifications** on different BLE characteristics **are enabled**.
  - **Environmental and Gas Gauge/Battery characteristics** are involved



life.augmented

ST BlueMS Protocol

In file **sensor.c**

Len	Type	Protocol version	Device ID	Feature Mask	Device MAC	Len	Type	Local Name	Len	Type	Pwr

```
389     uint8_t manuf_data[26] = {  2,  /* length*/0x0A,  0x00 /* 0 dBm */,          // Transmission Power
390               8, /* Length*/0x09, DEFAULT_NAME_ABLEMS,                         // Complete Name
391               13, /* Length*/0xFF, 0x01, /* SKD version */
392               0x05,/* 0x05 BlueNRG-Tile Board */
393   #if ENABLE_BLUEVOICE
394               0x6A, /* ADPCM Sync + ADPCM Audio + Led + Prox */
395    #else
396               0x22, /* Led + Prox */
397   #endif
398               0xFE, /* Acc + Gyro + Mag + Press + Hum + Temp + Batt*/
399   #if ENABLE_MOTIONFX
400               0x05, /* AccEvents + iNemo Compact */
401               0x40, /* eCompass */
402    #else
403               0x04, /* AccEvents */
404               0x00,
405   #endif
406               0x00, /* BLE MAC start */
407               0x00, 0x00, 0x00, 0x00, 0x00, /* BLE MAC stop */
408       };
```

3 bytes TX output power

9 bytes Local Name

13 bytes Manufacturer Specific

life.augmented

BlueMS Protocol – complete Feature Mask

4 Bytes

Len	Type	Protocol version	Device ID	Feature Mask	Device MAC	Len	Type	Local Name	Len	Type	Pwr

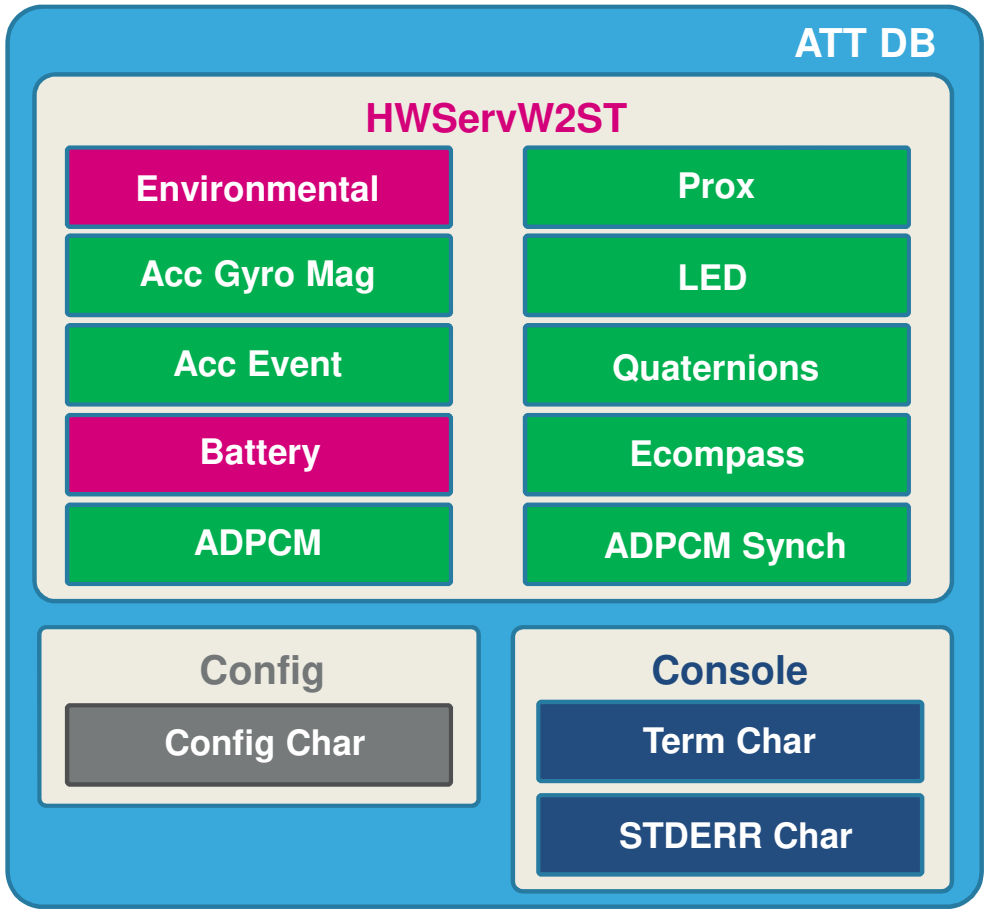
31	30	29	28	27	26	25	24	
RFU	ADPCM	Switch	PoA	ADPC	MicLevel	Proximity	Lux	0x6A
23	22	21	20	19	18	17	16	
Acc	Gyro	Mag	Pressure	Humidity	Temperat	Battery	2nd Temp	0xFE
15	14	13	12	11	10	9	8	
RFU	RFU	RFU	RFU	Beamforming	AccEvent	FreeFall	SensFusC	0x05
7	6	5	4	3	2	1	0	
SensFus	Compass	MotionInt	Activity	Carry Pos	MemsGes	ProxGes	Pedo	0x40

After all labs

life.augmented

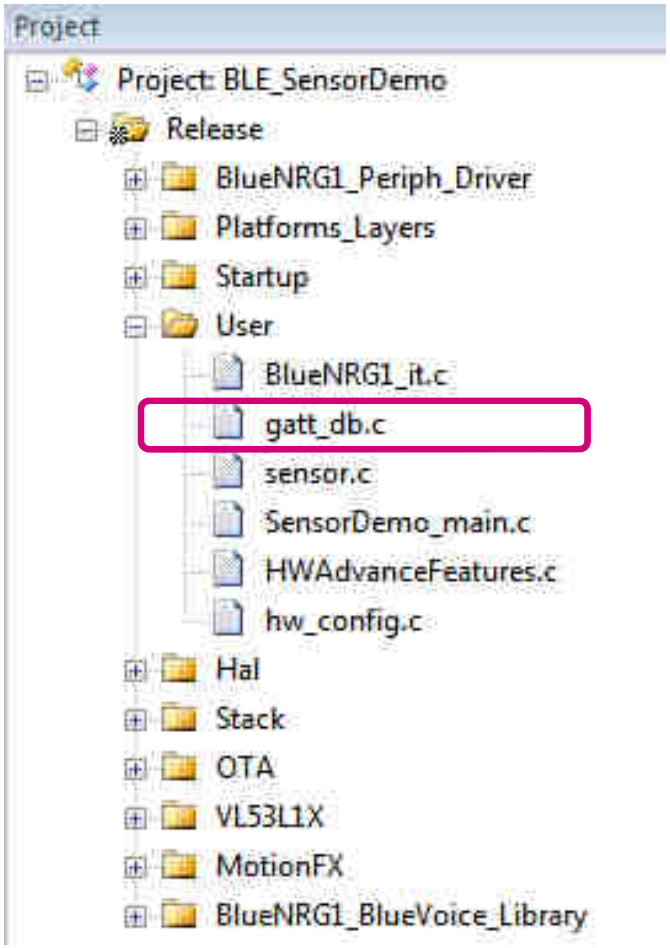
How Feature Mask is mapped onto ATT DB?

- **3 services:** HWServW2ST, Config, and Console
- **Each bit** of the 4-byte Feature Mask in the Advertising packet **corresponds to a HW/SW feature**
- In the Server ATT DB **a BLE Characteristic has to be added for each bit of the Advertising Feature Mask**



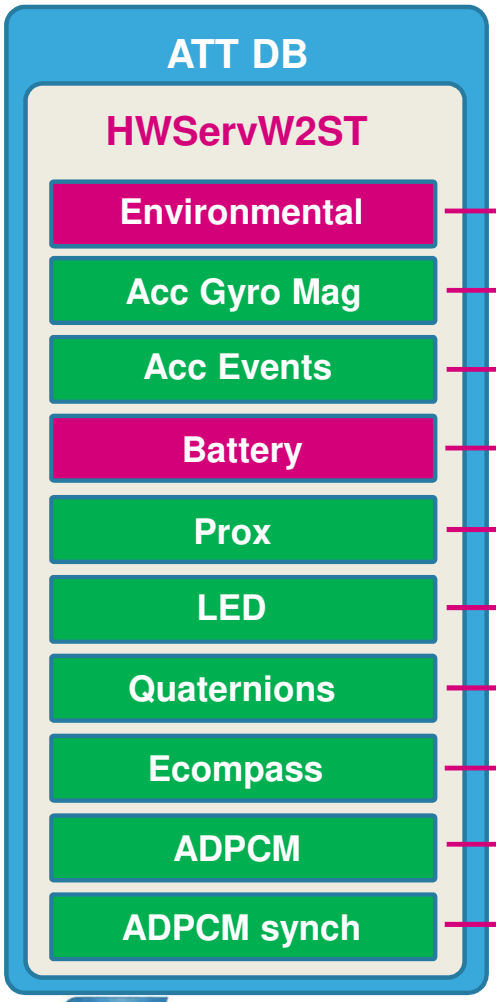
life.augmented

BLE_SensorDemo ATT DB



The **ATT DB** is defined and created in the file **gatt_db.c**

BLE_SensorDemo ATT DB



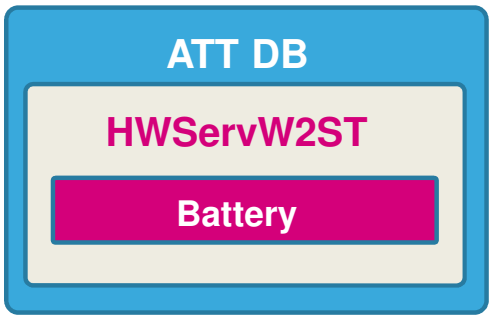
Len	PROPERTIES	UUID TYPE	UUID VALUE
12	N,R	128-bit	001D0000000111e1ac360002a5d5c51b
20	N	128-bit	00E0000000111e1ac360002a5d5c51b
5	N,R	128-bit	0000040000111e1ac360002a5d5c51b
9	N,R	128-bit	0002000000111e1ac360002a5d5c51b
4	N,R	128-bit	0200000000111e1ac360002a5d5c51b
3	N,R	128-bit	2000000000111e1ac360002a5d5c51b
8	N	128-bit	0000010000111e1ac360002a5d5c51b
4	N	128-bit	0000004000111e1ac360002a5d5c51b
20	N	128-bit	0800000000111e1ac360002a5d5c51b
6	N	128-bit	4000000000111e1ac360002a5d5c51b

NOTE: two additional bytes for a timestamp for each char

life.augmented

Example: Battery Characteristic

31	30	29	28	27	26	25	24	0x00
RFU	ADPCM	Switch	DoA	ADPC	MicLevel	Proximity	Lux	
23	22	21	20	19	18	17	16	0x02
Acc	Gyro	Mag	Pressure	Humidity	Temperat	**Battery**	2nd Temp	
15	14	13	12	11	10	9	8	0x00
RFU	RFU	RFU	RFU	Beamform	AccEvent	FreeFall	SensFusC	
7	6	5	4	3	2	1	0	0x00
SensFus	Compass	MotionInt	Activity	Carry Pos	MemsGes	ProxGes	Pedo	

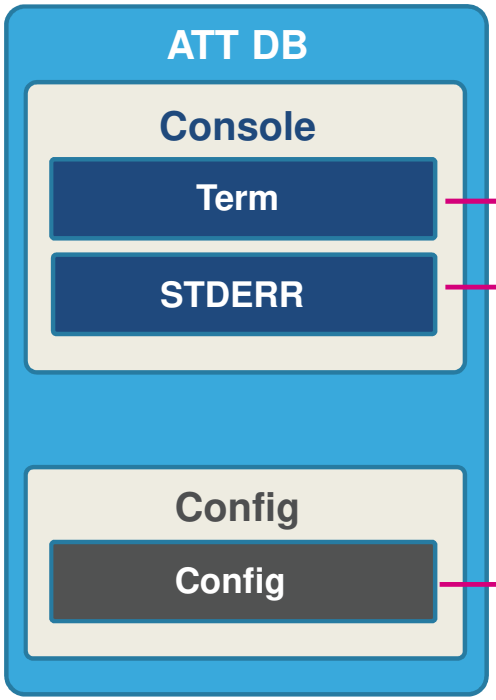


UUID VALUE
00020000000011e1ac360002a5d5c51b

The **UUID values** are linked to the Feature Mask in advertising

life.augmented

BLE_SensorDemo ATT DB



Each characteristic is defined by the following key parameters

Len	PROPERTIES	UUID TYPE	UUID VALUE
20	N,R	128-bit	00000001000E11e1ac360002a5d5c51b
20	N,R	128-bit	0000000200**0E**11e1ac360002a5d5c51b

This byte differentiates the **Service**

Len	PROPERTIES	UUID TYPE	UUID VALUE
20	N, Ww/oR, R,W	128-bit	0000000200**0F**11e1ac360002a5d5c51b

life.augmented

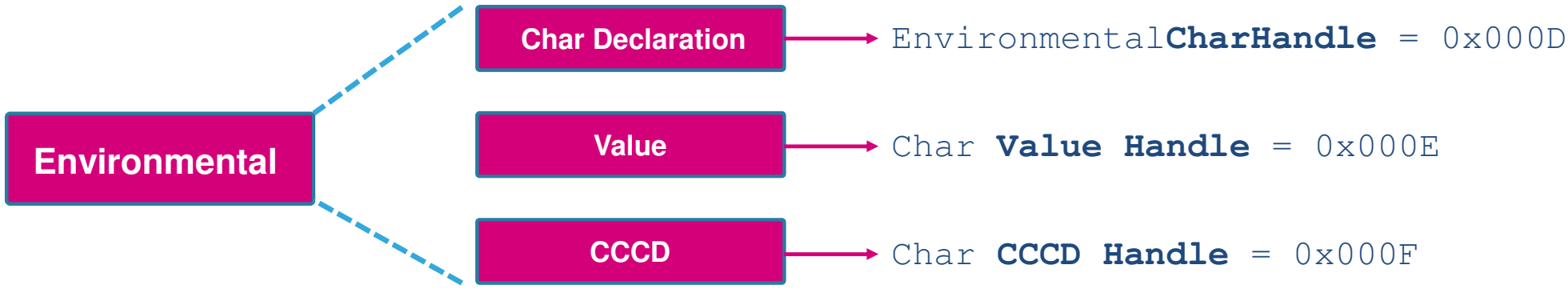
BLE_SensorDemo ATT DB


```
tBleStatus Add_HWServW2ST_Service(void)
aci_gatt_add_service(UUID_TYPE,UUID, …,MAX_NB_ATTRIBUTES,&ServHandle)
aci_gatt_add_char(ServHandle,UUID_TYPE,UUID,Length,PROPERTIES,PERMISSIONS,…,&CharHandle)
```


```
HWServW2STHandle = 0x000C
EnvironmentalCharHandle = 0x000D
AccGyroMagCharHandle = 0x0010
ConfigW2STHandle = 0x002B
ConfigCharHandle = 0x002C
```

life.augmented

BLE_SensorDemo ATT DB



- **Declaration Handle**: used by the application to access the Characteristic
- **Characteristic Value Handle**: used by the Client for Read/Write operations
- **Client Characteristic Configurator Descriptor** (**CCCD**): a GATT descriptor is added by default by the stack, if **char** has **Notify/Indicate property**. Used by Client to enable notifications/indications on char value.

life.augmented

Central: Services and Characteristics discovery procedure

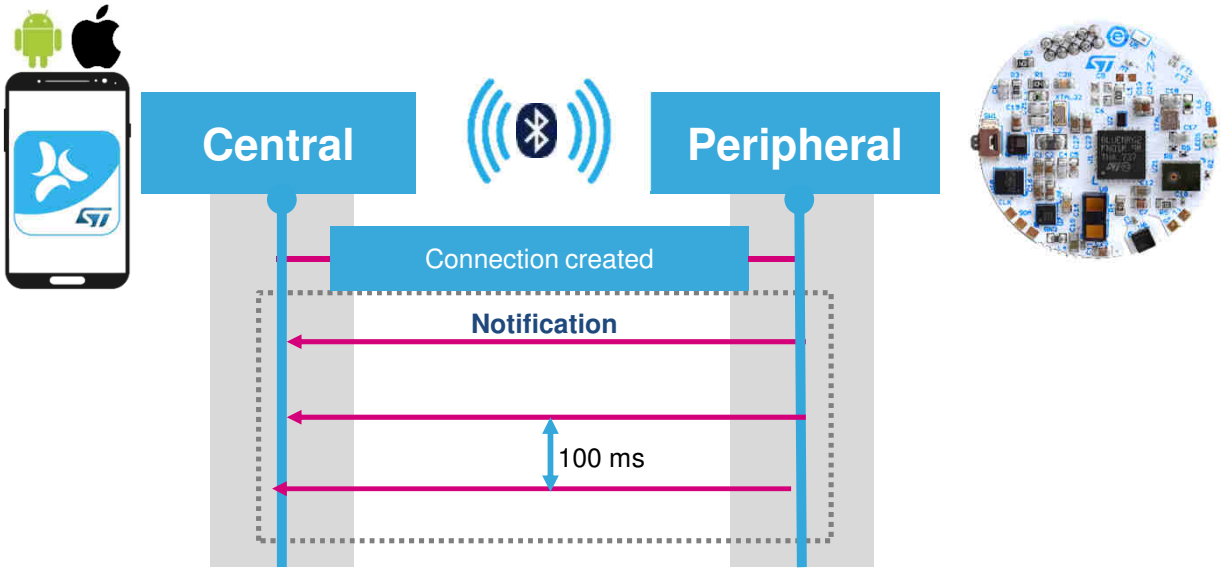
UUIDs **Handles** **Values**

This is how ATT DB looks like from the Central

Name	Handle	Property	Value	Value Length
Service User Defined (0x0000000000111E19AB40002A5D5C51B)	0x000C			
Characteristic User Defined (0x001D0000000111E1AC360002A5D5C51B)	0x000D	Read, Notify	0x011F0111023C00018B9CA605	0x0C
Client Characteristic Configuration (0x2902)	0x000F		0x0000	0x02
Characteristic User Defined (0x00E00000000111E1AC360002A5D5C51B)	0x0010	Notify		
Client Characteristic Configuration (0x2902)	0x0012		0x0000	0x02
Characteristic User Defined (0x00000400000111E1AC360002A5D5C51B)	0x0013	Read, Notify	0x0000002405	0x05
Client Characteristic Configuration (0x2902)	0x0015		0x0000	0x02
Characteristic User Defined (0x02000000000111E1AC360002A5D5C51B)	0x0016	Read, Notify	0x88A1232E	0x04
Client Characteristic Configuration (0x2902)	0x0018		0x0000	0x02
Characteristic User Defined (0x20000000000111E1AC360002A5D5C51B)	0x0019	Read, Notify	0x0024E6	0x03
Client Characteristic Configuration (0x2902)	0x001B		0x0000	0x02
Characteristic User Defined (0x00020000000111E1AC360002A5D5C51B)	0x001C	Read, Notify	0x0180000CBA03E8AB63	0x09
Client Characteristic Configuration (0x2902)	0x001E		0x0000	0x02
Characteristic User Defined (0x00000100000111E1AC360002A5D5C51B)	0x001F	Notify		
Client Characteristic Configuration (0x2902)	0x0021		0x0000	0x02
Characteristic User Defined (0x00000040000111E1AC360002A5D5C51B)	0x0022	Notify		
Client Characteristic Configuration (0x2902)	0x0024		0x0000	0x02
Characteristic User Defined (0x08000000000111E1AC360002A5D5C51B)	0x0025	Notify		
Client Characteristic Configuration (0x2902)	0x0027		0x0000	0x02
Characteristic User Defined (0x40000000000111E1AC360002A5D5C51B)	0x0028	Notify		
Client Characteristic Configuration (0x2902)	0x002A		0x0000	0x02
Service User Defined (0x00000000000F11E19AB40002A5D5C51B)	0x002B			
Characteristic User Defined (0x00000002000F11E1AC360002A5D5C51B)	0x002C	Write w/o resp, Notify		
Client Characteristic Configuration (0x2902)	0x002E		0x0000	0x02
Service User Defined (0x00000000000E11E19AB40002A5D5C51B)	0x002F			
Characteristic User Defined (0x00000001000E11E1AC360002A5D5C51B)	0x0030	Read, Write w/o resp, Write, Notify	0x0	0x00
Client Characteristic Configuration (0x2902)	0x0032		0x0000	0x02
Characteristic User Defined (0x00000002000E11E1AC360002A5D5C51B)	0x0033	Read, Notify	0x0	0x00
Client Characteristic Configuration (0x2902)	0x0035		0x0000	0x02

life.augmented

BLE Sensor – Characteristic update

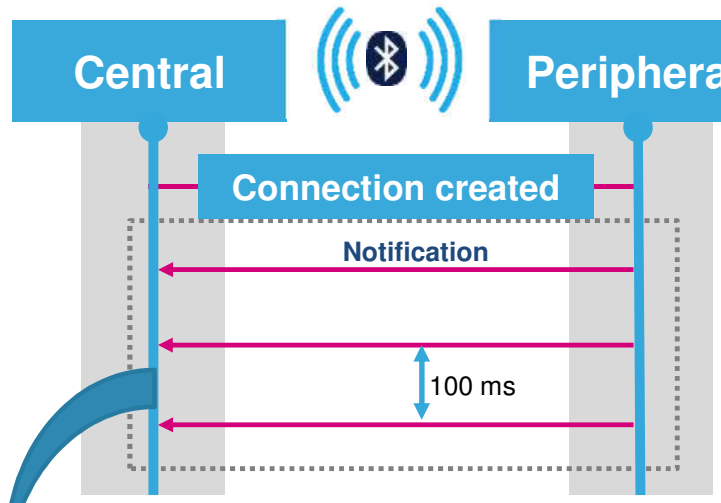



```
aci_gatt_update_char_value(ServiceHandle, 0x000C,  //HWserv handle
                           CharHandle,    0x000D,  //EnvChar handle
                           Offset,        0x00,
                           Length,        0x08,
                           Value)         Data)    //EnvChar Value
```

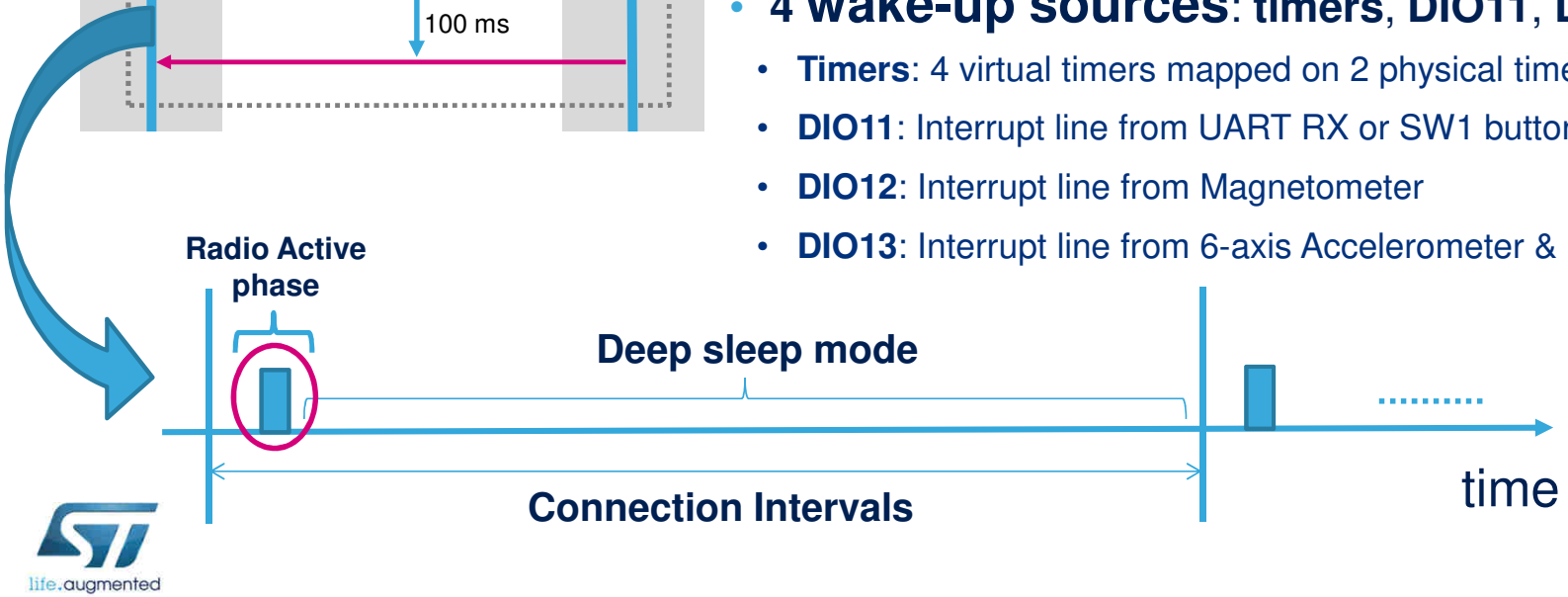
Description	Where
If notifications (or indications) are enabled on the characteristic, this API sends a notification (or indication) to the client.	GATT server

life.augmented

Cortex-M0 Sleep management

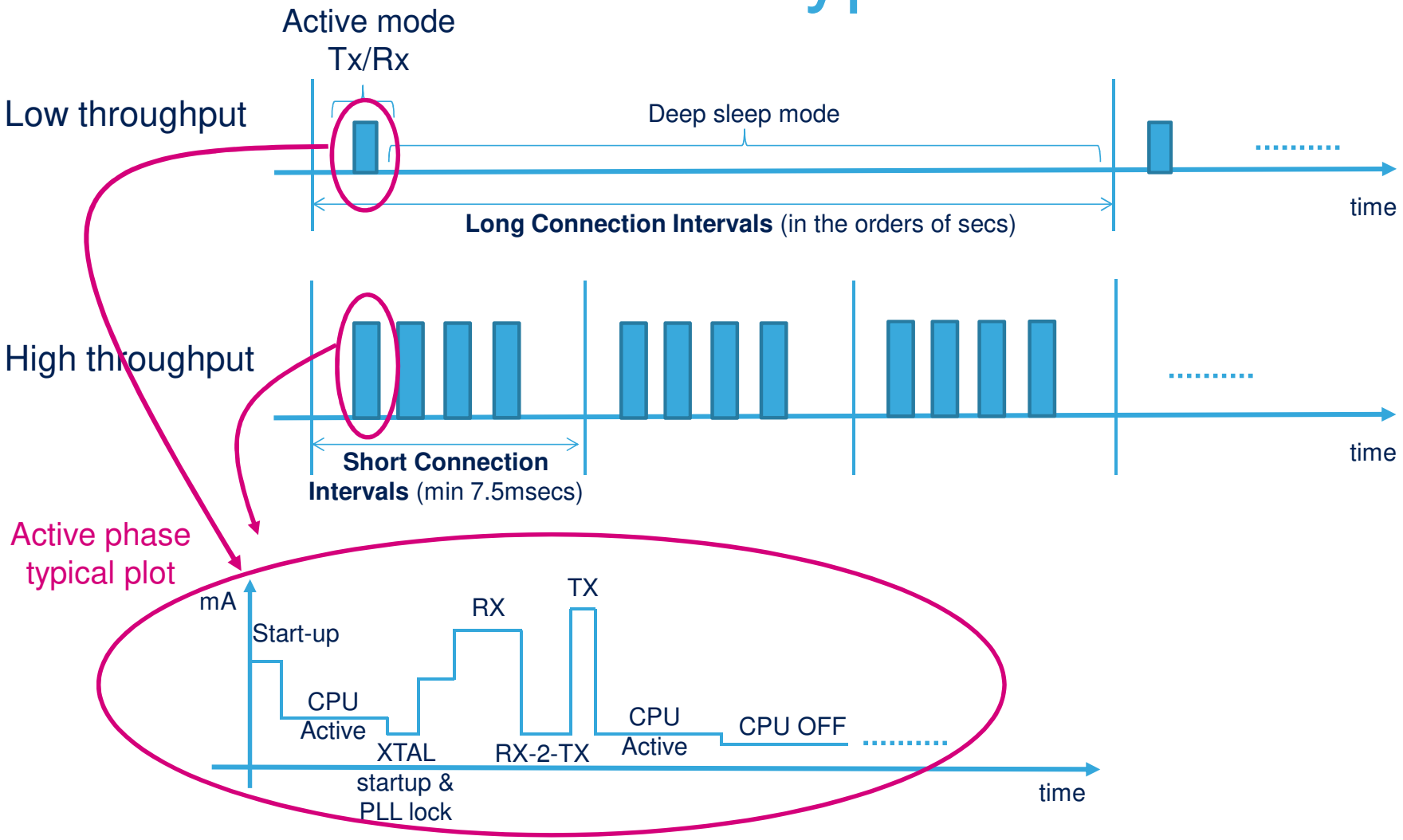


- **Sleep mode** managed through specific a **library** and **API**
  - *BlueNRG_Sleep(sleepMode, gpioWakeBitMask, gpioWakeLevelMask)*
- **4 wake-up sources**: **timers**, **DIO11**, **DIO12** and **DIO13**
  - **Timers**: 4 virtual timers mapped on 2 physical timers managed by the radio IP
  - **DIO11**: Interrupt line from UART RX or SW1 button
  - **DIO12**: Interrupt line from Magnetometer
  - **DIO13**: Interrupt line from 6-axis Accelerometer & Gyroscope



life.augmented

Typical scenarios



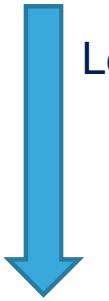
Low Power Modes

- **Deep sleep mode** can represent most of the application time.
- Efficient management of sleep mode significantly lowers the avg power consumption.

BlueNRG-2 flexible low power architecture

Sleeping Mode	Consumption	Notes
RUNNING	1.9mA	Core running
CPU HALT	1.5mA	WFI instruction
WAKETIMER	900nA	GPIOs and Timer Wakeup
NOTIMER	**500nA**	GPIOs only Wakeup

Highest pwr



Low Power library combines requests coming from the application with the radio operating mode

Lowest pwr

life.augmented

Context Save/Restore

- BlueNRG-2 Low Power Library **handles autonomously entering and exiting to/from the sleep mode**.

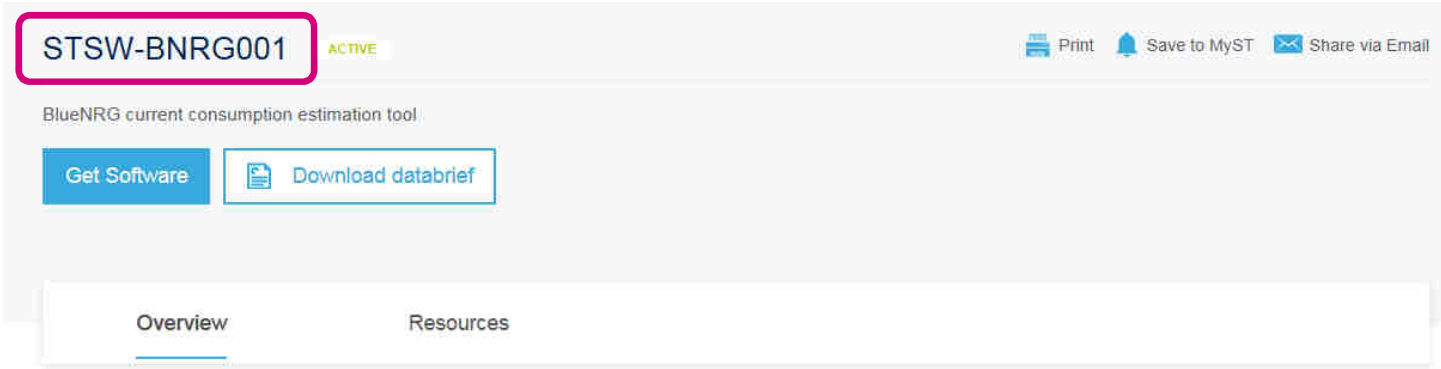
- **The library saves peripherals configurations and application context** before entering deep sleep, and **restores the context** on exiting from low power state:
  - **CPU returns to execute the next instruction after the low power function call**
  - **No need to re-initialize peripheral and radio stack**
  - **RAM retention is guaranteed**

NO need for the application to worry about sleep management and RAM retention!

BlueNRG Current Consumption Estimation Tool

- On st.com at this link:

  http://www.st.com/content/st_com/en/products/embedded-software/wireless-connectivity-software/stsw-bnrg001.html

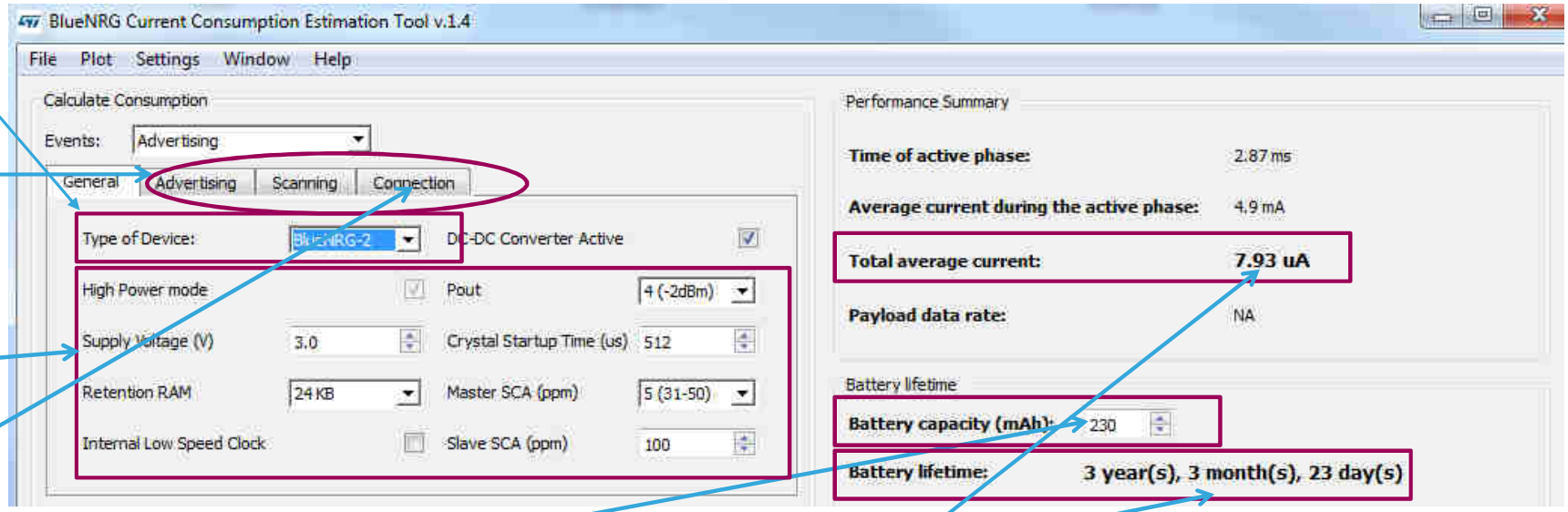


life.augmented

BlueNRG Current Consumption Estimation Tool

- **Accurate estimate** of average **current consumption** and **battery lifetime**

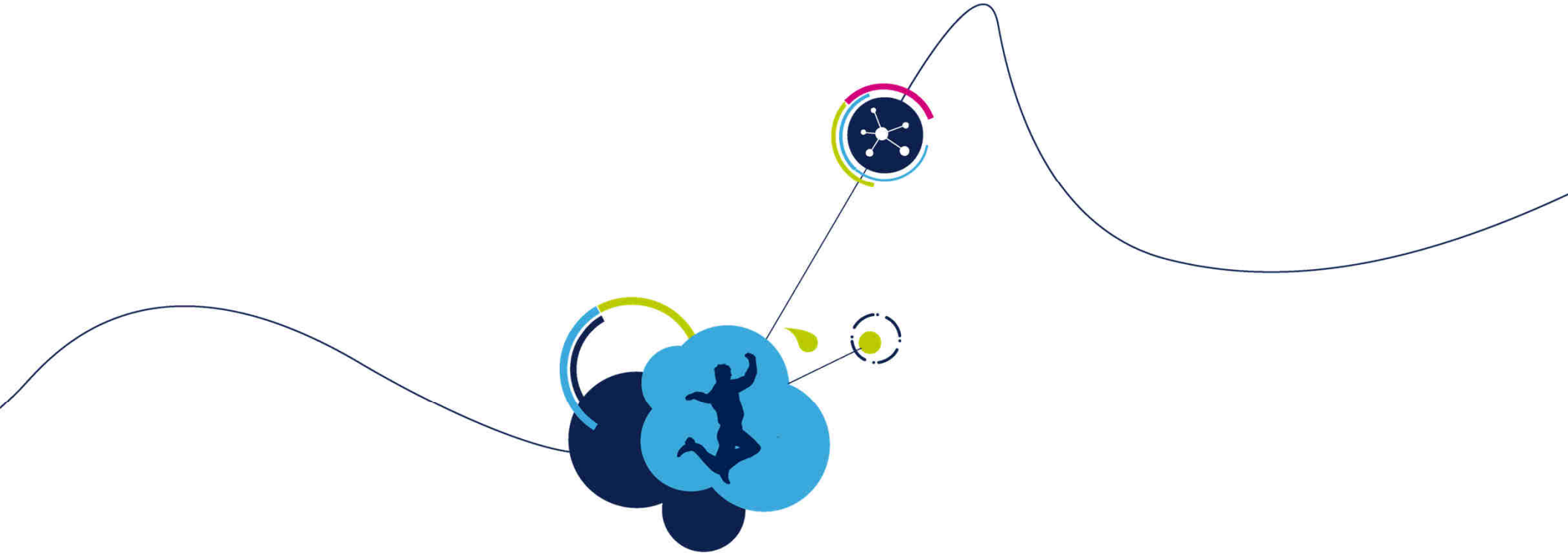
1. **Select** the device
2. **Select** the type of event
3. **Configure** the HW operating conditions
4. **Configure** the application use case scenario



5. **Specify** the Battery capacity

6. **Read** the **average current** and the estimated **battery lifetime**

life.augmented

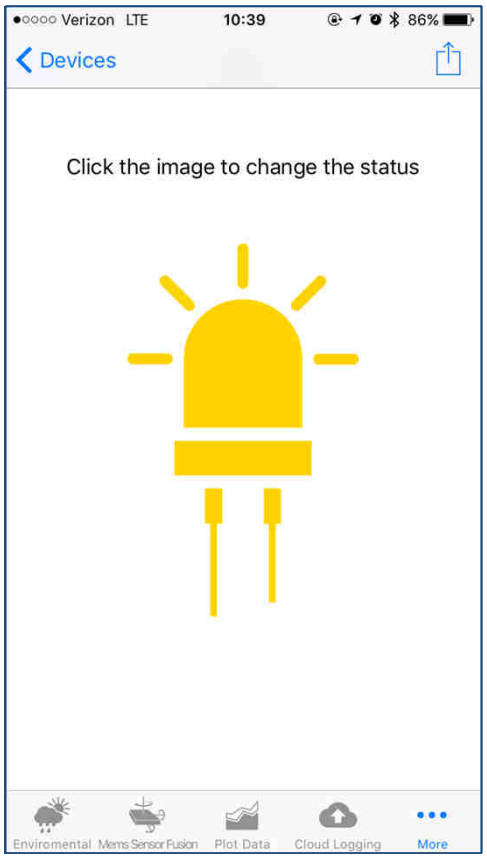


Lab 3

LED characteristic

life.augmented

Enable LED toggling

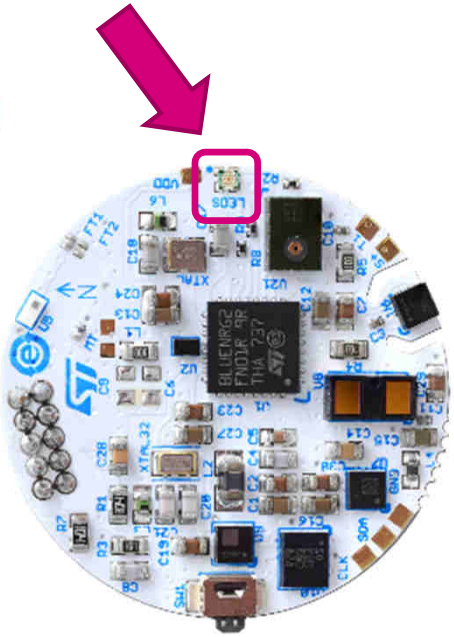


Write Packet

'1' LED ON
'0' LED OFF

Notification Packet

'1' LED ON
'0' LED OFF



1. **Tap** the image on the screen
2. Send **BLE notification packet**
3. Image changes color and the **RED LED toggles**

life.augmented

Code flow

1. Client **enables LED characteristic notifications**
2. Client **Writes in Config Char**
3. Parse **write command** from the client and **send BLE notification** on LED status from the server

ST BlueMS Protocol

BlueMS

4 Bytes

Len	Type	Protocol version	Device ID	Feature Mask	Device MAC	Len	Type	Local Name	Len	Type	Pwr

31	30	29	28	27	26	25	24	
RFU	ADPCM	Switch	DoA	ADPC	MicLevel	Proximity	Lux	0x20
23	22	21	20	19	18	17	16	
Acc	Gyro	Mag	Pressure	Humidity	Temperat	Battery	2nd Temp	0x1E
15	14	13	12	11	10	9	8	
RFU	RFU	RFU	RFU	Beamform	AccEvent	FreeFall	SensFusC	0x00
7	6	5	4	3	2	1	0	
SensFus	Compass	MotionInt	Activity	Carry Pos	MemsGes	ProxGes	Pedo	0x00

life.augmented

Client enables LED characteristic notifications

1. In the file **gatt_db.c** **go to** the **line 670**
2. **Client writes** in the **LED Characteristics Client Configurator Descriptor (CCCD)** and **Server enables notifications** through the ***xFeatureNotification*** **structure** in **line 673** by ***xFeatureNotification.LedNotification = true;***

```
670     } else if (attr_handle == (LedCharHandle + 2)) {
671
672         if (att_data[0] == 01) {
673             xFeatureNotification.LedNotification = true;
674             PRINTF("Led ON\n\r");
675         } else if (att_data[0] == 0) {
676             xFeatureNotification.LedNotification = false;
677             PRINTF("Led OFF\n\r");
678         }
```

Client Writes in Config Char

1. In the file **gatt_db.c** **go to** the **line 806**
2. Client **writes** in the **Config Char** and **Server parses** the command through the function ***ConfigCommandParsing(att_data, data_length)***

```
806     } else if (attr_handle == ConfigCharHandle + 1) {
807
808       /* Received one write command from Client on Configuration characteristic */
809       ConfigCommandParsing(att_data, data_length);
810
811     }
```

life.augmented

Parse write command and send BLE notification on LED status

1. In file **Sensor.c** at line **1040**
2. **Check** on the **FeatureMask** (line **1046**)
3. If it's the LED feature mask (line **1152**)
4. **Check** on the command value. If **0x01** (case1)
5. **Turn ON LED3**
6. **Send Notification** on the Config char on the command received and parsed
7. **Send Notification** on the **LED char** with the **LED status**

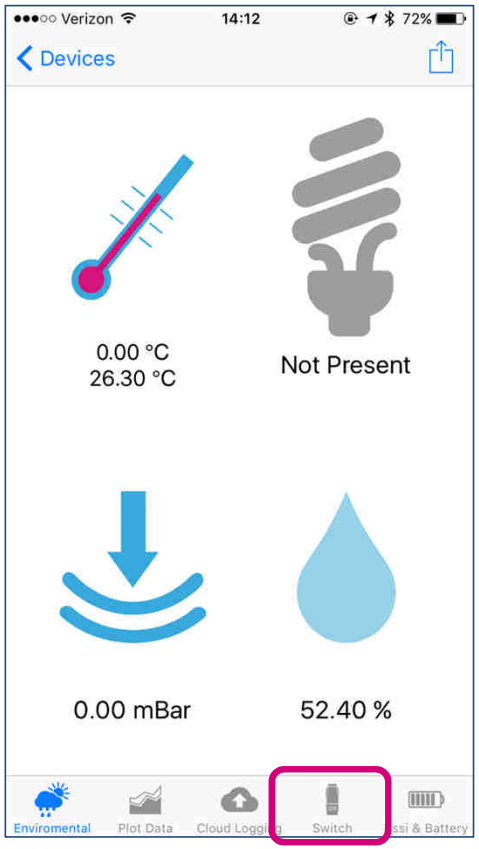
```
1040  uint32_t ConfigCommandParsing(uint8_t
1041      uint8_t FeatureMask = att_data[3];
1042      uint8_t Command = att_data[4];
1043      uint8_t Data = att_data[5];
1044      uint32_t SendItBack = 1;
1045
1046      switch (FeatureMask)
```

```
1152  case FEATURE_MASK_LED:
1153      switch (Command) {
1154      case 1:
1155          SdkEvalLedOn(LED1);
1156          PRINTF("Enabled: LED\n\r");
1157          Config_Notify(FEATURE_MASK_LED, Command, Data);
1158          break;
1159      case 0:
1160          SdkEvalLedOff(LED1);
1161          PRINTF("Disabled: LED\n\r");
1162          Config_Notify(FEATURE_MASK_LED, Command, Data);
1163          break;
1164      }
1165
1166      if (xFeatureNotification.LedNotification) {
1167          if (SdkEvalLedGetState(LED1))
1168              Led_Update(ENABLE);
1169          else
1170              Led_Update(DISABLE);
1171      }
1172      break;
```

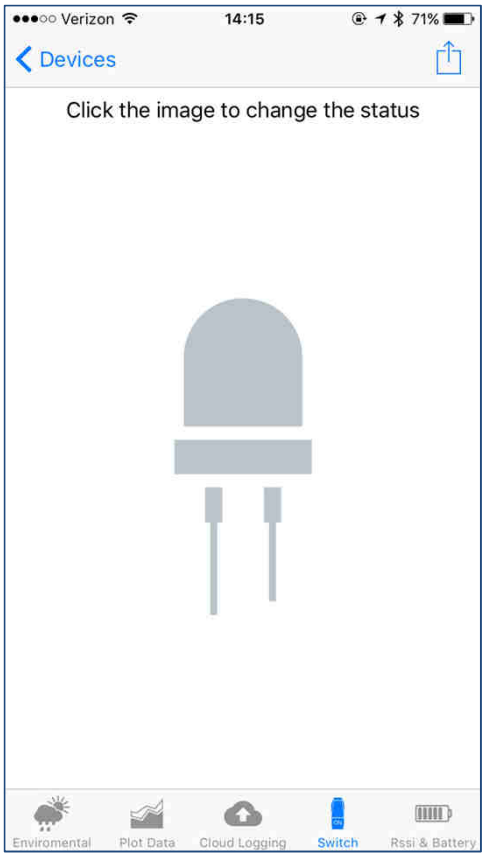
life.augmented

Enabling LED toggling

Click on “**Switch**”

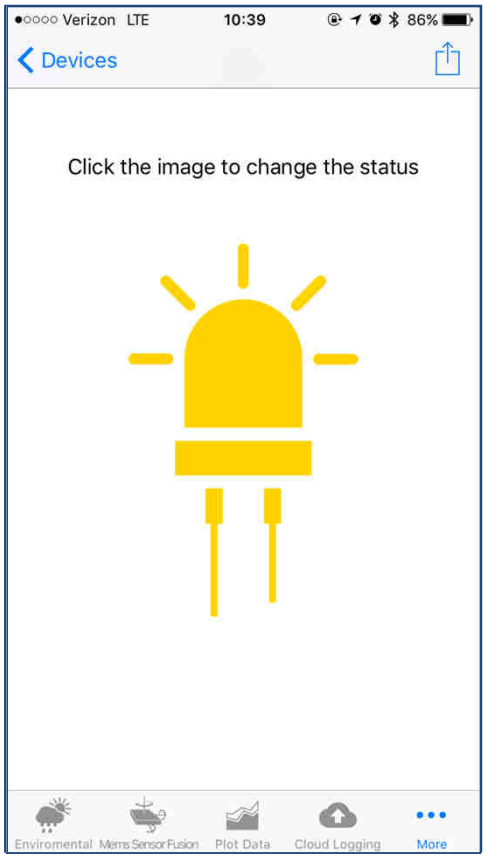


A new page will appear



life.augmented

Enable LED toggling

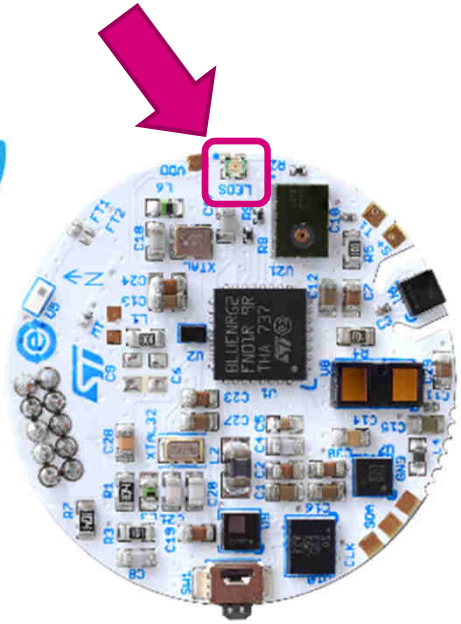


Write Packet

'1' LED ON
'0' LED OFF

Notification Packet

'1' LED ON
'0' LED OFF

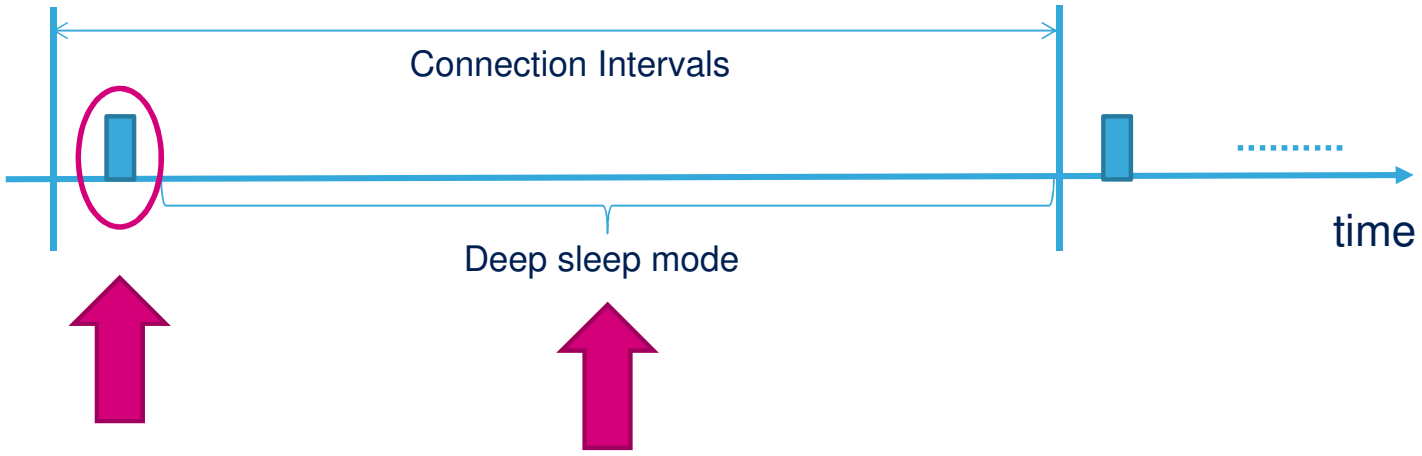


1. **Tap the image** on the screen
2. **Send BLE notification packet**
3. **Image changes color** and **RED LED toggles**

life.augmented

LED fast blinking due to Sleep Mode

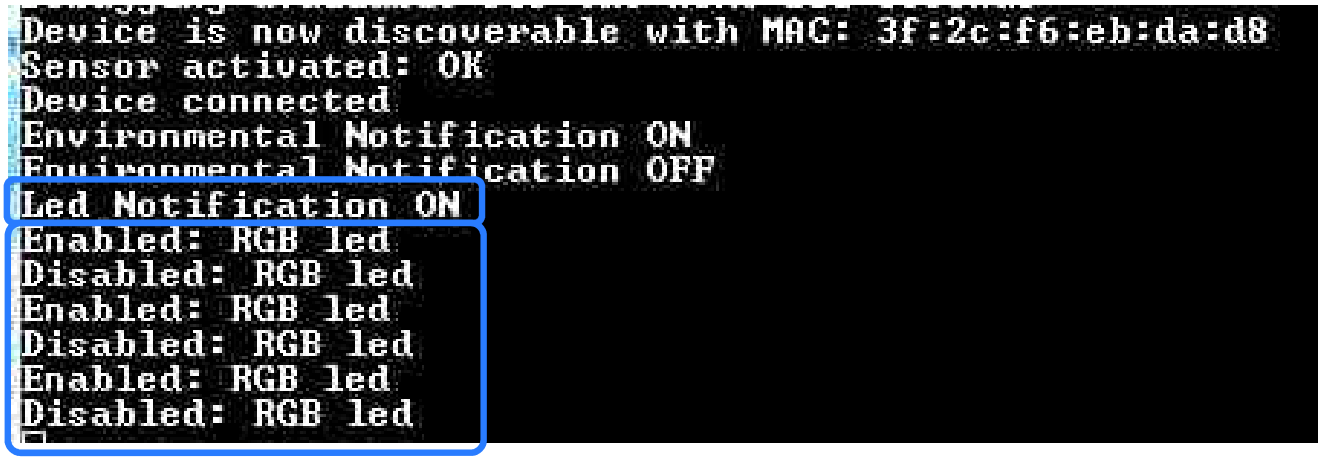
- **Red LED** fast toggling shows the entering/exiting to/from Sleep mode



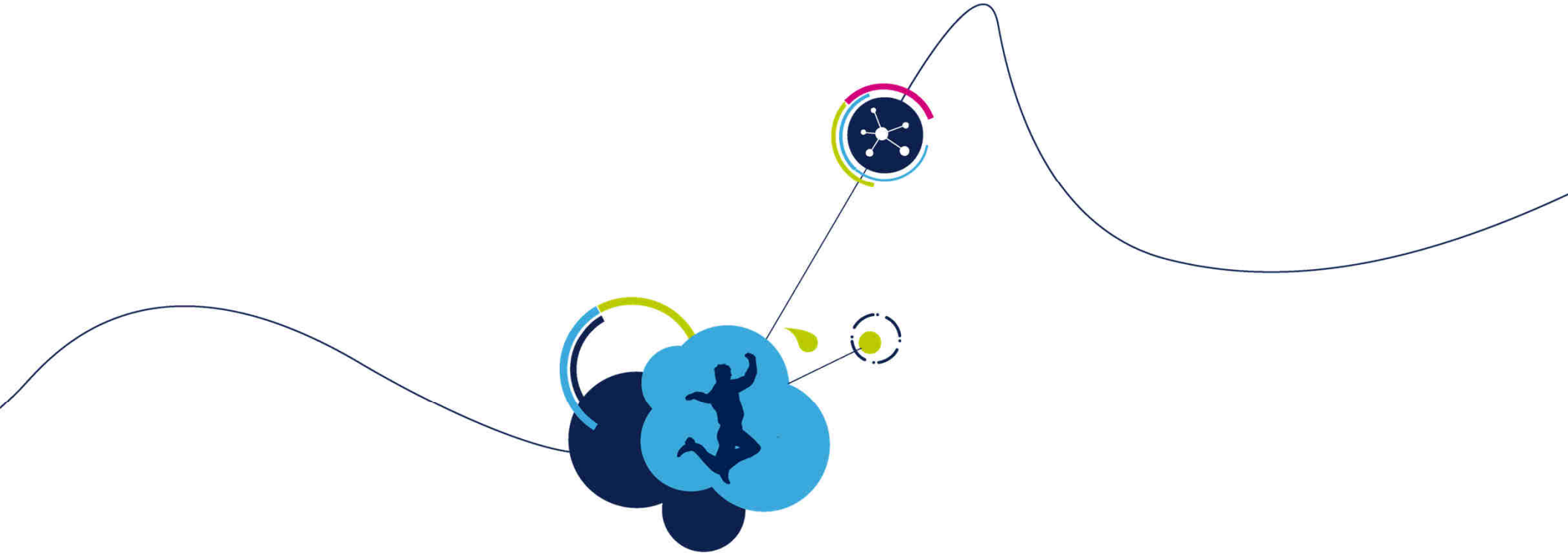
life.augmented

Tera Term output

- String ***"LED Notification ON"*** will appear as the app tab is enabled
- Each time user **click on the LED icon** in the **BLE Sensor app**, the **Write command is sent** to the board and the **LED toggles** accordingly ("Enabled: RGB led" = ON, "Disabled: RGB led" = OFF)



life.augmented



Lab 4

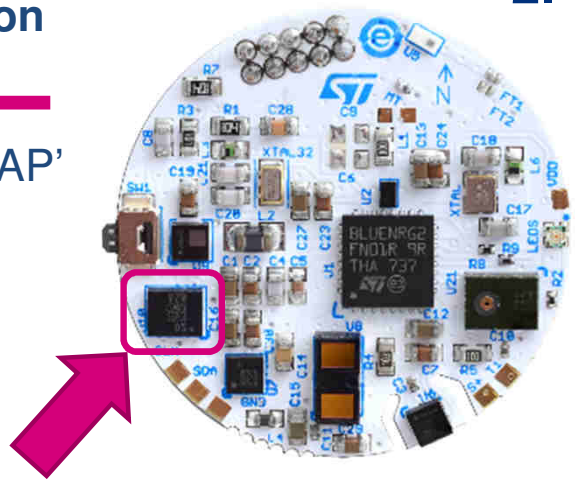
Accelerometer embedded events detection

life.augmented

Example - Single Tap

Notification Packet

‘SINGLE TAP’



1. On the board **perform a Single Tap**
2. **Send BLE notification packet**

NOTE: this is just an example for SingleTap. Other events will be displayed later in the lab!

life.augmented

ST BlueMS Protocol

BlueMS

4 Bytes

Len	Type	Protocol version	Device ID	Feature Mask	Device MAC	Len	Type	Local Name	Len	Type	Pwr

31	30	29	28	27	26	25	24	
RFU	ADPCM	Switch	DoA	ADPC	MicLevel	Proximity	Lux	0x20
23	22	21	20	19	18	17	16	
Acc	Gyro	Mag	Pressure	Humidity	Temperat	Battery	2nd Temp	0x9E
15	14	13	12	11	10	9	8	
RFU	RFU	RFU	RFU	Beamform	AccEvent	FreeFall	SensFusC	0x04
7	6	5	4	3	2	1	0	
SensFus	Compass	MotionInt	Activity	Carry Pos	MemsGes	ProxGes	Pedo	0x00

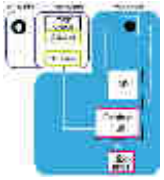
life.augmented

LSM6DSO at a glance

1. PERFORMANCE AND FEATURE

- **Accuracy:** Tango and Daydream compliance
- **0.55mA** current consumption is **HP combo**
  -15% vs. LSM6DSL/M @ same performance

- New **ultra low power mode: 14uA (@100Hz ODR) Axl only**

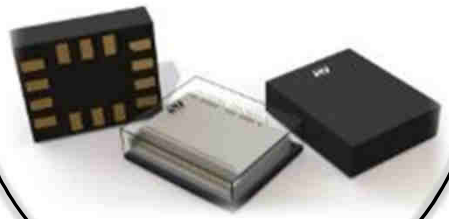


2. FLEXIBILITY: algorithm @ silicon level

- **Pedometer 2.x**
  WeChat Precision

- **FSM** build custom sensors for XL and Gyro

LSM6DSO



2.5x3x0.86 mm

3. NEW STANDARD

- **I3C**

- **FIFO tag**

4. INNOVATION

- MEMS: **Axl T-structure** for mechanical shock robustness
- **Compressed FIFO (3x)**
- **10 patents protecting LSM6DSO innovation**

life.augmented

Client enables Acc Event characteristic notifications

1. In the file **gatt_db.c** **go to** the **line 703**
2. **Client writes** in the **Acc Event Characteristic Client Configurator Descriptor (CCCD)** and **Server enables notifications** through the ***xFeatureNotification*** **structure** in **line 714** by ***xFeatureNotification.AccEventNotification = true***

```
705   } else if (attr_handle == (AccEventCharHandle + 2)) {
706
707     if (xFeatureNotification.MotionNotification == false && xFeatureNotification.iNemoEngineNotification == false) {
708       if (att_data[0] == 01) {
709         lsm6dso_xl_data_rate_set(0, LSM6DSO_XL_ODR_52Hz);
710         GPIO_WriteBit(GPIO_Pin_7, Bit_RESET); // Proximity OFF
711         EnableHWMultipleEvents();
712         ResetHWPedometer();
713         GPIO_EXTICmd(GPIO_Pin_13, ENABLE);
714         Config_Notify(FEATURE_MASK_ACC_EVENTS, 'm', 1);
715         AccEvent_Notify(0, 3);
716         xFeatureNotification.AccEventNotification = true;
717         PRINTF("AccEvent Notification ON\n\r");
718
719       } else if (att_data[0] == 0) {
720         lsm6dso_xl_data_rate_set(0, LSM6DSO_XL_ODR_OFF);
721         xFeatureNotification.AccEventNotification = false;
722         GPIO_EXTICmd(GPIO_Pin_13, DISABLE);
```

life.augmented

Read Event Status and send BLE notification


```
1299  void MEMSCallback(void) {
1300
1301      lsm6dso_all_sources_get(0, &all_source);
```

```
1334    /* Check if the interrupt is due to Single Tap */
1335    if (xHardwareFeaturePresence.HwSingleTAP || xHardwareFeaturePresence.MultipleEvent) {
1336      if (all_source.reg.all_int_src.single_tap)
1337        SdkEvalLedOn(LED1);
1338        SdkEvalLedOn(LED3);
1339        PRINTF("Event: Single Tap\n\r");
1340        AccEvent_Notify(ACC_SINGLE_TAP, 2);
1341      }
1342    }
```

In the file **sensor.c**

1. **line 1299** ***void MEMSCallback(void)***

   Callback triggered by IO13

2. **line 1301** ***lsm6dso_all_sources_get***

   Read accelerometer status registers

3. **line 1336** ***if (all_source.reg.all_int_src.single_tap)***

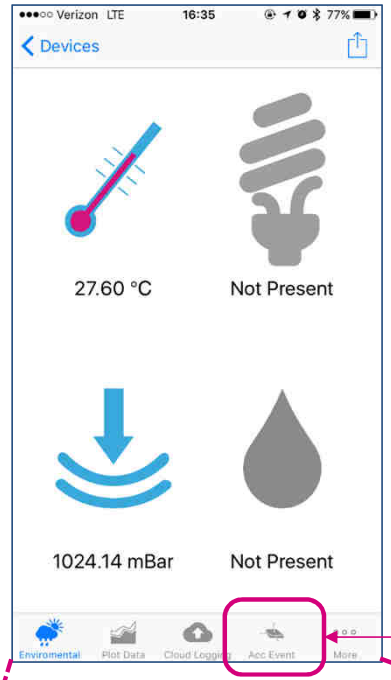
   Check vs. single tap event

4. **line 1340** ***AccEvent_Notify***

   Send BLE notification

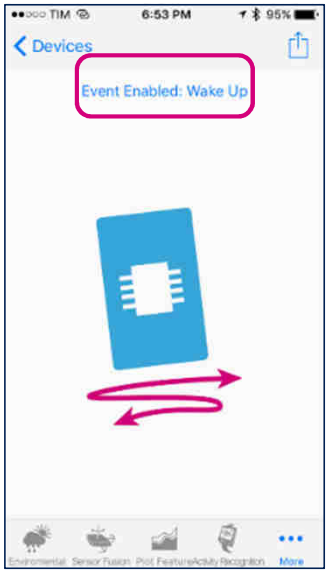
life.augmented

Visualize Single Tap Event in ST BLE Sensor



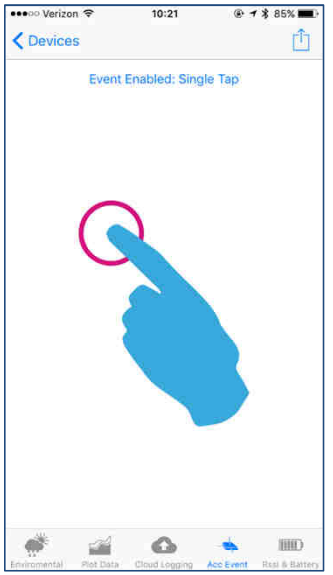
Select
“Acc Event”

Tap
“Event Enabled”



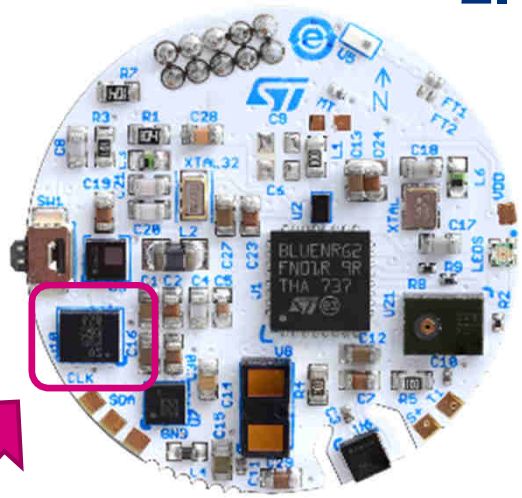
Select
“Single Tap”
and hit **“Select”**

Enabled
“Single Tap”



Event Detection

1. On the board **perform** a **Single Tap**
2. **Send BLE notification packet**



Tap **gently** on the board

Notification Packet

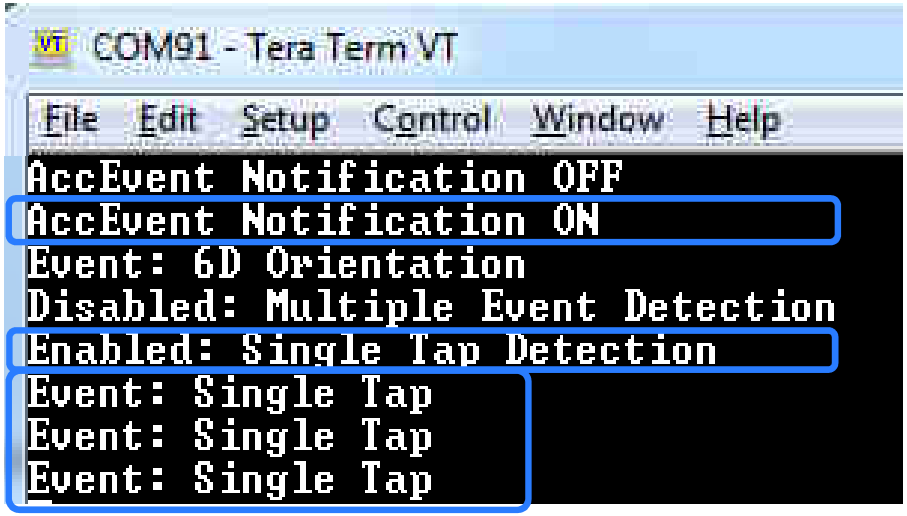
‘SINGLE TAP’

Single Tap

life.augmented

Tera Term output

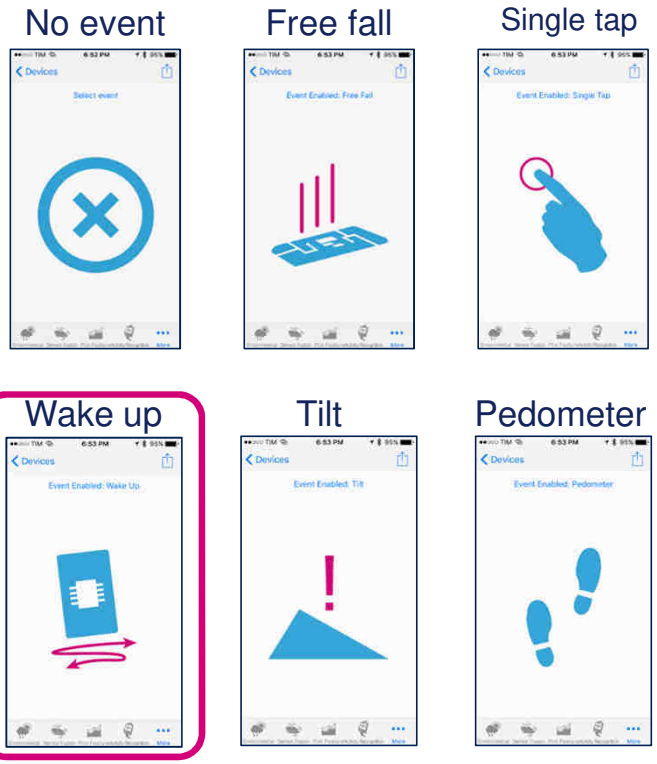
- ***"AccEvent Notification ON":*** **enable notifications** on the Accelerometer Event characteristic
- ***"Enabled Single Tap Detection":*** user **enables Single Tap event detection**
- ***"Event: Single Tap":*** the actual **BLE notification packet** is sent upon detection of the Single Tap event.



NOTE: if the phone display enters sleep mode, the notifications are not sent

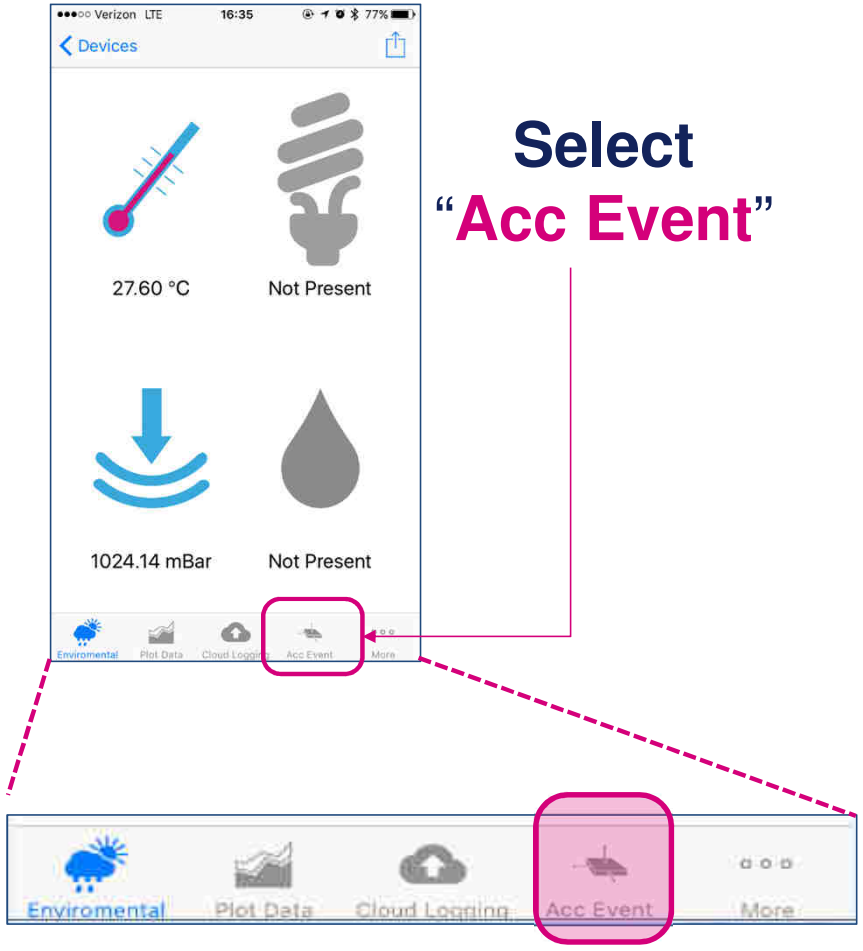
LSM6DSO Embedded Events

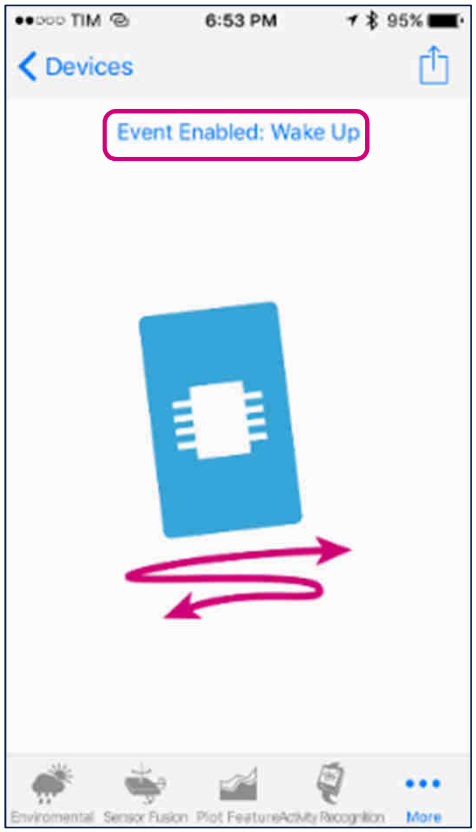
Other possible embedded events



WakeUp example

Visualize Hardware Wakeup Event in BLE Sensor

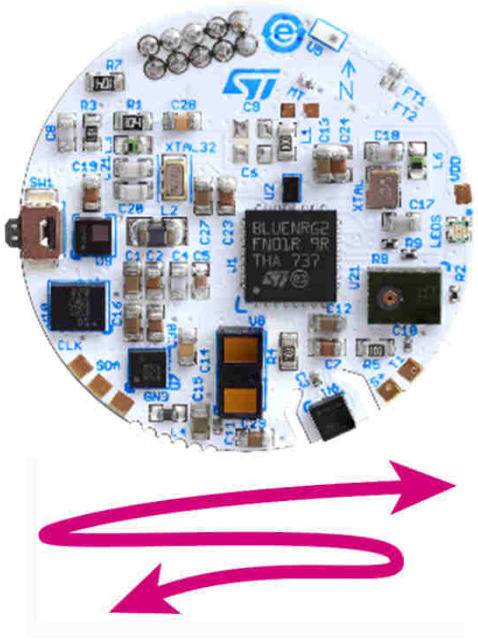




life.augmented

Event Detection

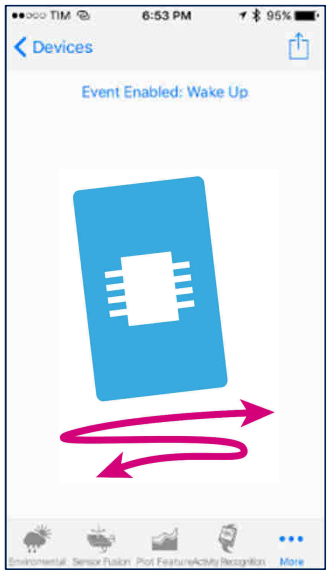
1. **Leave** the board **still** for a few seconds
2. **Shake** the board
3. **Send BLE notification packet**



Notification Packet

‘WAKE UP’

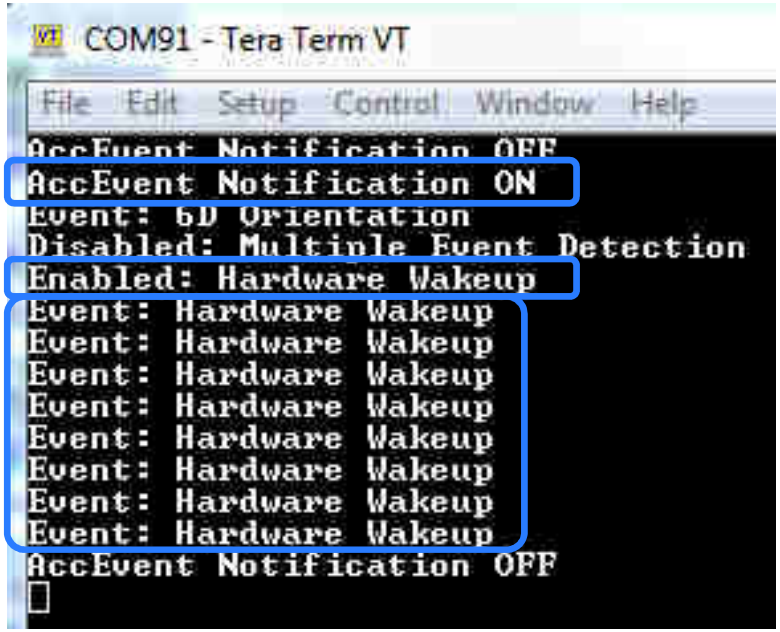
Wake Up



life.augmented

Tera Term output

- ***"AccEvent Notification ON":*** **enable notifications** on the Accelerometer Event characteristic
- ***"Enabled Hardware Wakeup":*** user enable hardware **wakeup event detection**
- ***"Event: Hardware Wakeup":*** the actual **BLE notification packet sent** upon detection of the Hardware Wakeup event.



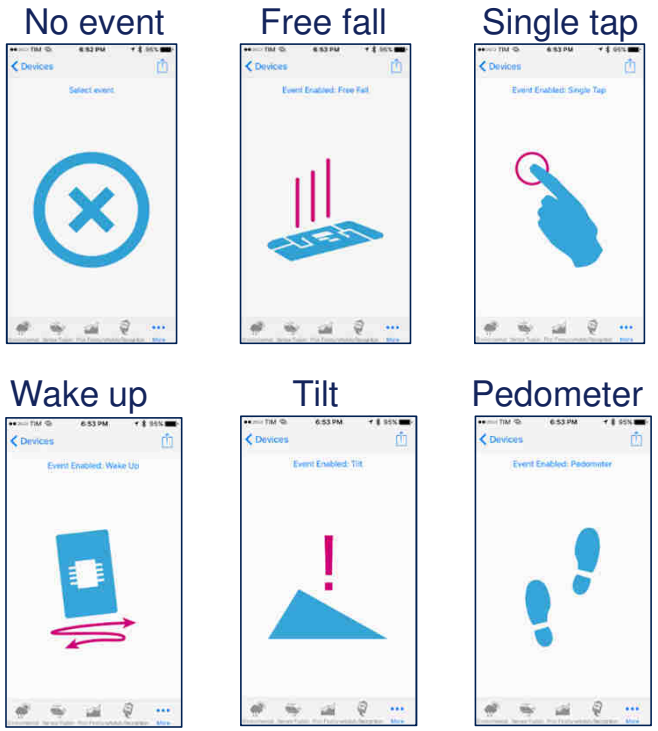
NOTE: if the phone display enters sleep mode, notifications are not sent

life.augmented

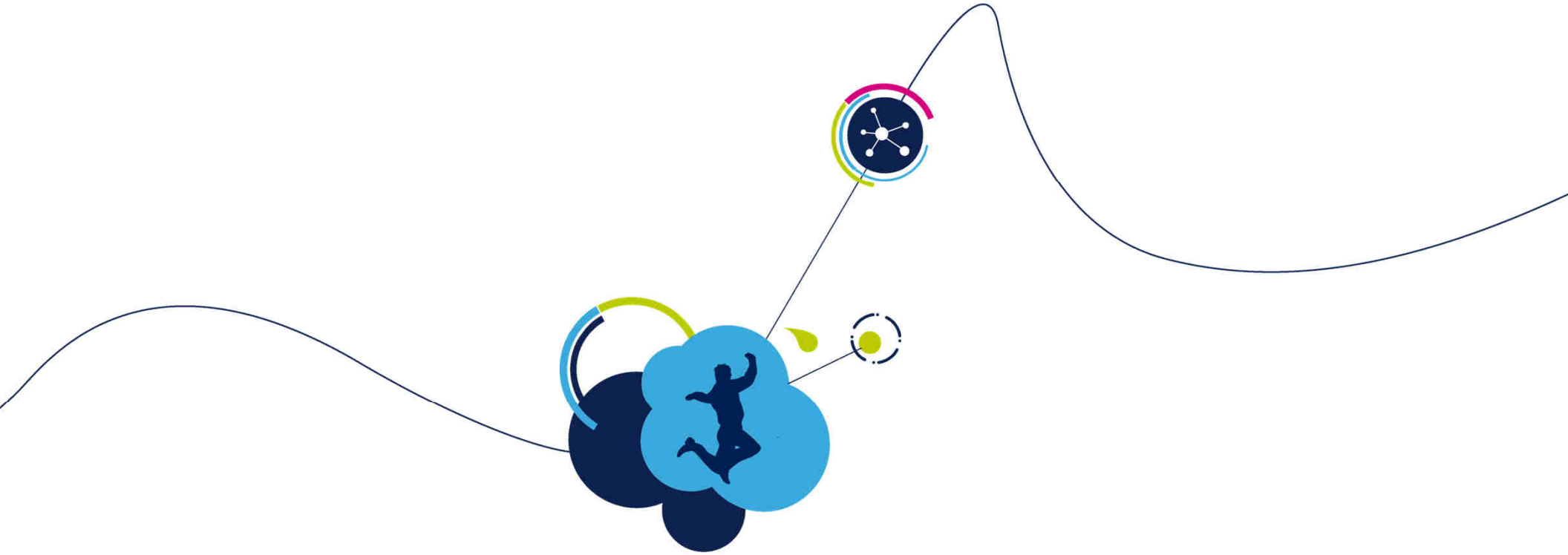
LSM6DSO Embedded Events

You can go ahead later and test other events…

Other possible embedded events



life.augmented



Lab 5

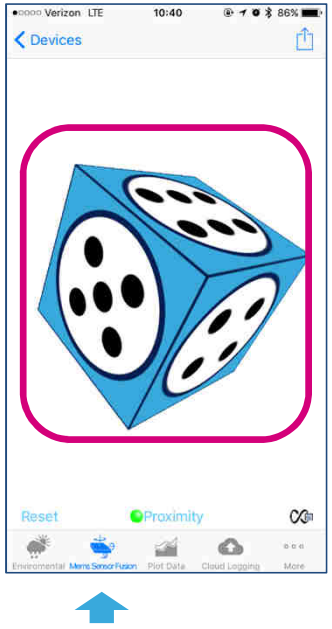
9-axis Acc+Gyro+Mag Sensor Data Fusion

life.augmented

Goal

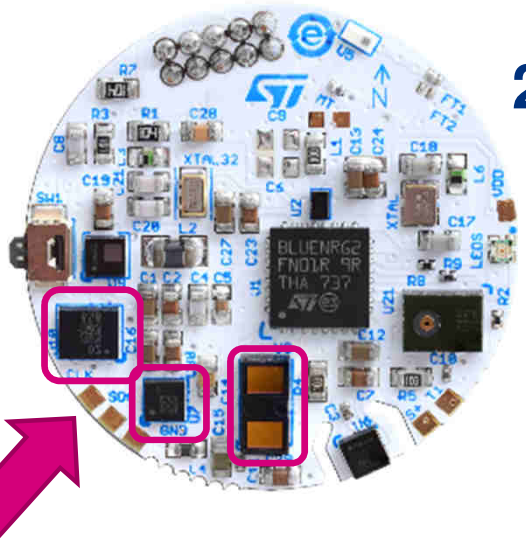
- In this example we are going to demonstrate how to:
  - **Enable** the embedded sensor data fusion library
    - Input: raw Acc+Gyro+Mag sensor data
    - Output: quaternions
  - **Scale the quaternions value** by a scaling factor proportional to proximity detection
  - **Send** scaled data fusion information through **BLE notifications packets** to the ST BLE Sensor app client

Sensor Data Fusion



Notification Packet

QUATERNIONS



Sensors used by data fusion:
Acc+Gyro+Mag and Proximity

1. **Enable** Sensor Fusion library
2. **Send** quaternions information through **BLE notification packet**

life.augmented

Code modifications

1. **Enable Sensor Data fusion library**
2. **Enable proximity sensor**
3. **Send quaternions data** – scaled by proximity sensor data - through **BLE notifications packets**

ST BlueMS Protocol

BlueMS

4 Bytes

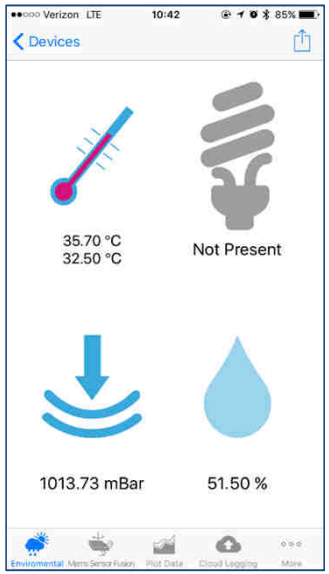
Len	Type	Protocol version	Device ID	Feature Mask	Device MAC	Len	Type	Local Name	Len	Type	Pwr

31	30	29	28	27	26	25	24	
RFU	ADPCM	Switch	DoA	ADPC	MicLevel	Proximity	Lux	0x22
23	22	21	20	19	18	17	16	
Acc	Gyro	Mag	Pressure	Humidity	Temperat	Battery	2nd Temp	0xFE
15	14	13	12	11	10	9	8	
RFU	RFU	RFU	RFU	Beamform	AccEvent	FreeFall	SensFusC	0x05
7	6	5	4	3	2	1	0	
SensFus	Compass	MotionInt	Activity	Carry Pos	MemsGes	ProxGes	Pedo	0x40

life.augmented

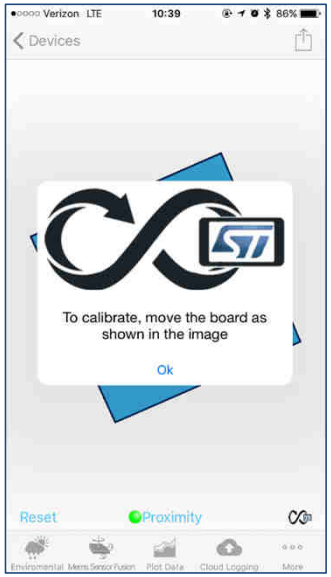
Sensor Data Fusion

Swipe left to **view** the **Mems Sensor Fusion**



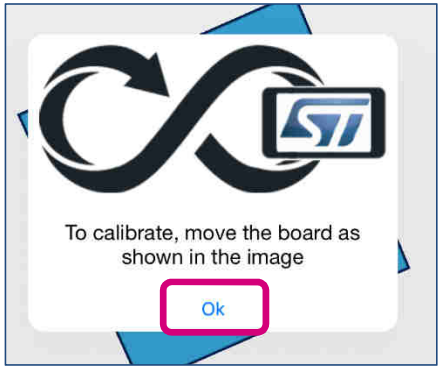
1

"Mems Sensor Fusion" tab



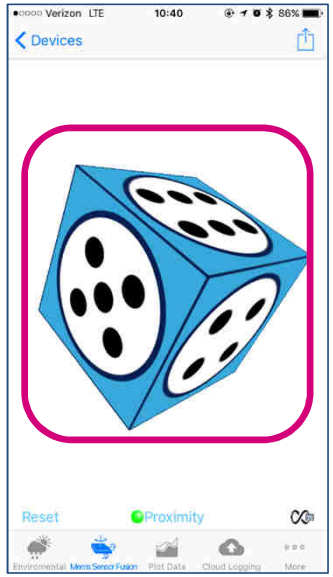
2

Click on **OK**



3

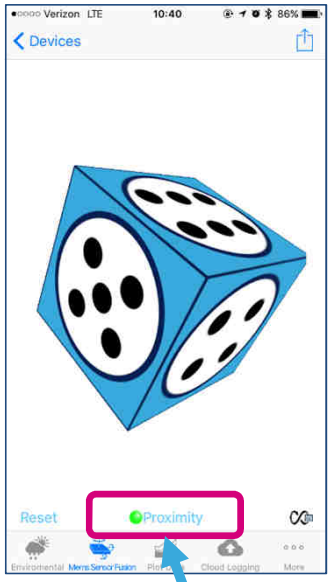
Sensor Fusion **enabled**



life.augmented

Proximity Sensor

Enable Proximity Sensor

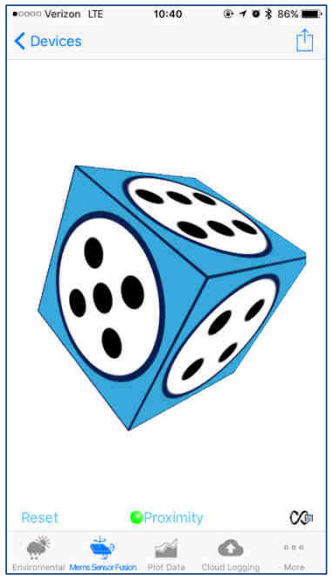


ON Proximity

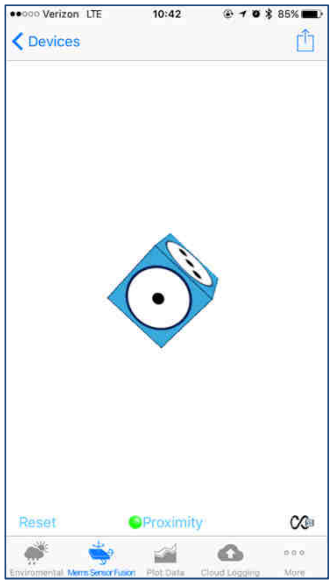
OFF Proximity

Move your hand far and close to the BlueNRG-Tile

1 Far from BlueTile



2 Near to BlueTile



Note: ONLY for rendering purposes, the cube is scaled proportional to a clipped 30 cm distance

life.augmented

Proximity Sensor - VL53L1X

3rd gen **ToF sensor** with lens for **long distance** ranging & ROI selection

Product highlights

OLGA: **4.9 x 2.5 x 1.56 mm**
FoV : 27°
Compatible footprint with VL53L0X

Enhanced performances:

- Full FoV ranging : **400cm+** (white target, no IR)
- **Programmable FOV:** SPAD zone selection – full-screen to 4x4

Cutting-edge module and silicon :

- **Fastest miniature ToF product in the market** up to 50Hz
- **Integrated lens** for longer range and better ambient light immunity
- **Programmable settings** to best fit customer's application:
  - Low power with interrupts for user / object detection
  - Long distance ranging
  - High accuracy for small movement detection

Applications

Presence user detection

- Autonomous mode with interrupts
- Low-power
- Long distance 400cm+
- PC, tablets, IoT, portable handsets, security

Obstacle detection:

- Robots: Obstacle avoidance
- Vacuum cleaners: Wall following, cliff detection
- Drones: Take-off and landing, Ceiling detection

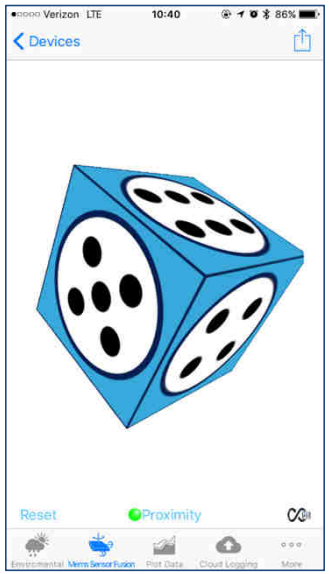
Accurate objects distance scanning

- Vending machines: control of objects in racks
- Coins dispensers: coins counting
- Smart shelves: Consumer scanning

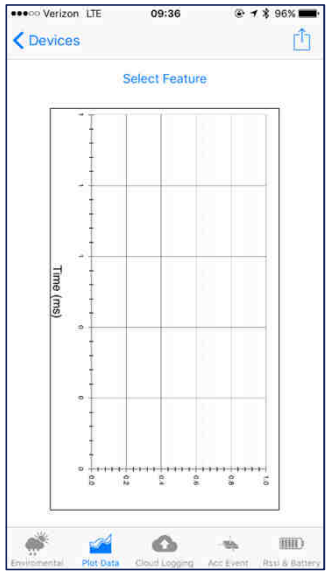
life.augmented

Full-range Proximity Real-time Data Plot

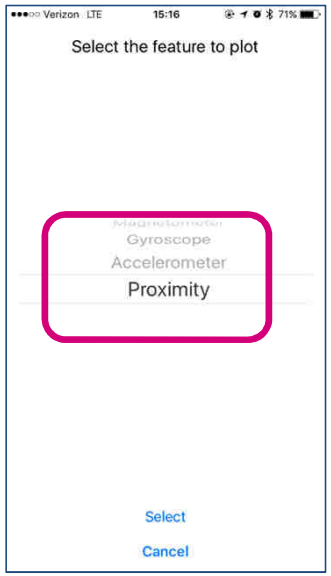
1 **Swipe left** to **view** the real-time data plot



2 **"Plot Feature"** tab

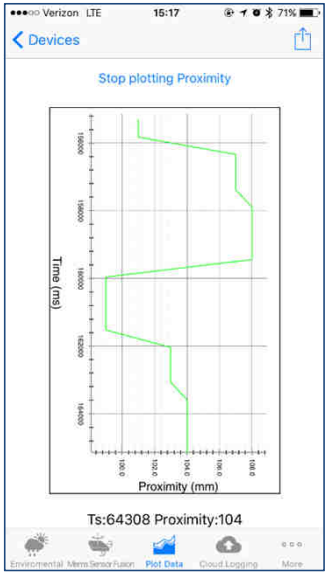


3 **Select Proximity**



4 **Select** the **time frame**

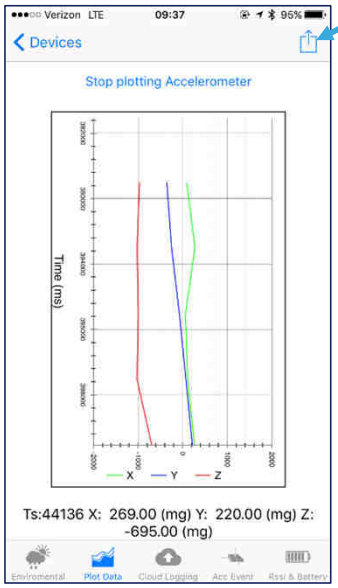
5 **View** the **real-time data plot**



life.augmented

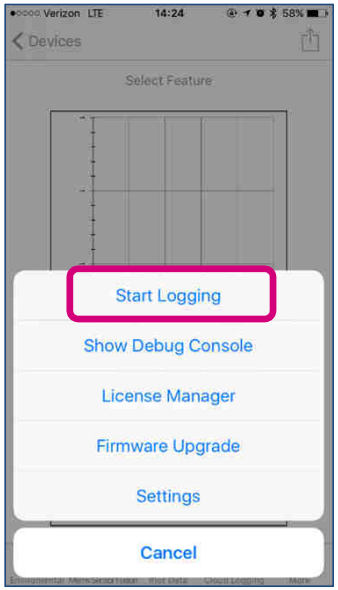
Logging Real-time Data Plot

Click on
top right corner



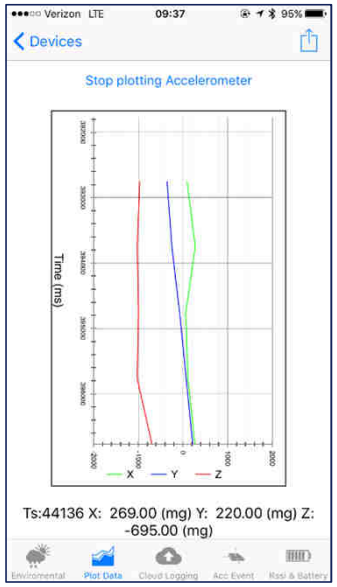
1

Select
"**Start logging**"



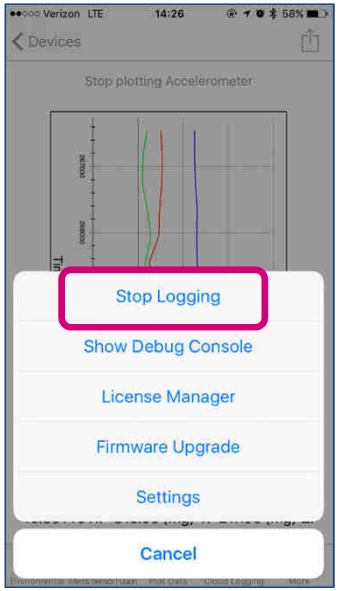
2

After some time
re-click on the
right-top corner



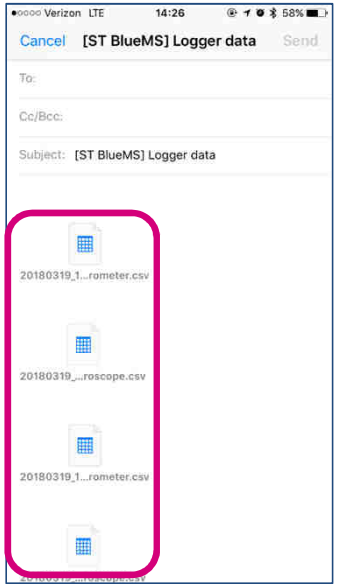
3

Select
"**Stop logging**"



4

App opens your
email client

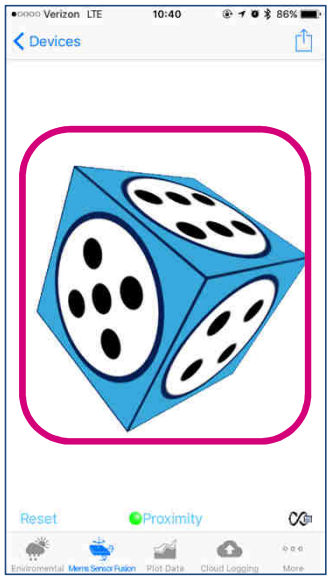


.csv files attached
to the email

life.augmented

Sensor Data Fusion

Sensor Fusion **enabled**



Try **static** position: no drift, the cube is perfectly still

- In steady static position the **acc** and **mag** enable an accurate orientation estimate
- acc vibrations and mag bias and interferences are **rejected or compensated by the motionFX library**

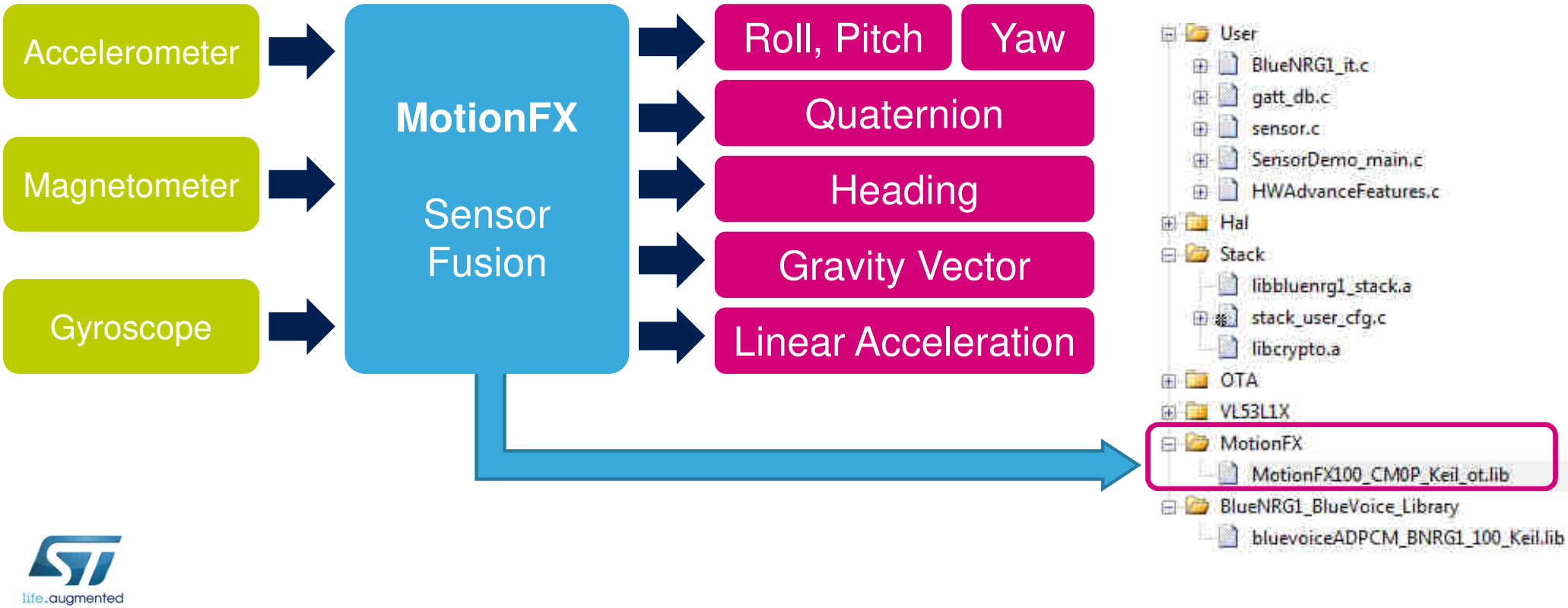
Try highly **dynamic** motion: perfect tracking

- In high dynamic motion the acc cannot be used, therefore the **gyro** is used to update the orientation
- gyro bias is **estimated and compensated at runtime** by the **motionFX** library

life.augmented

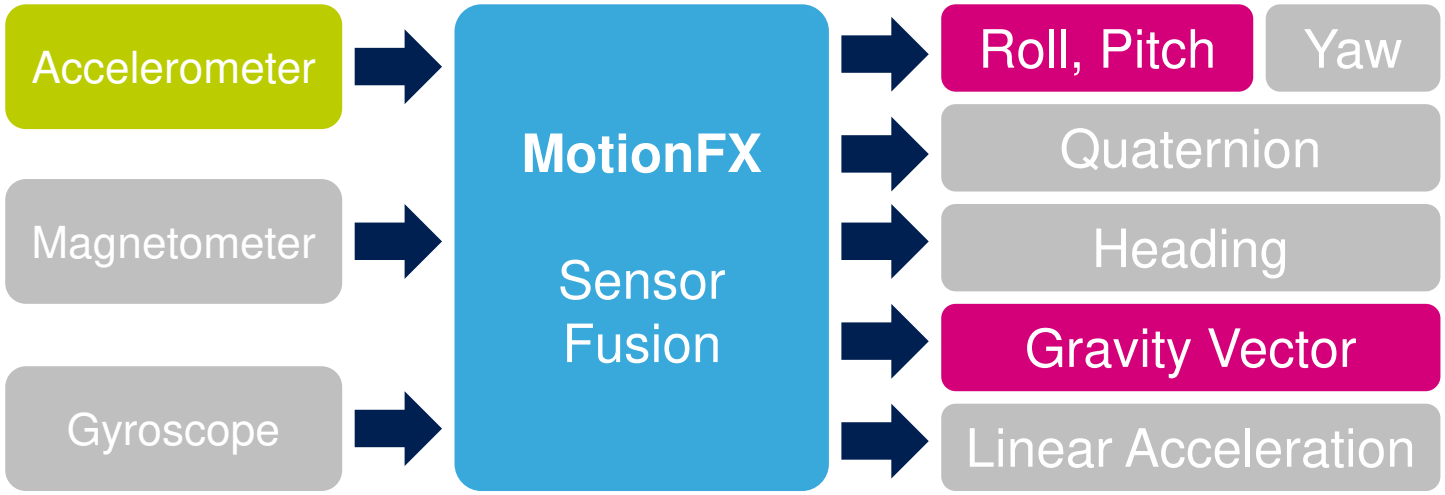
Sensor Data Fusion

MotionFX library



Sensor Data Fusion

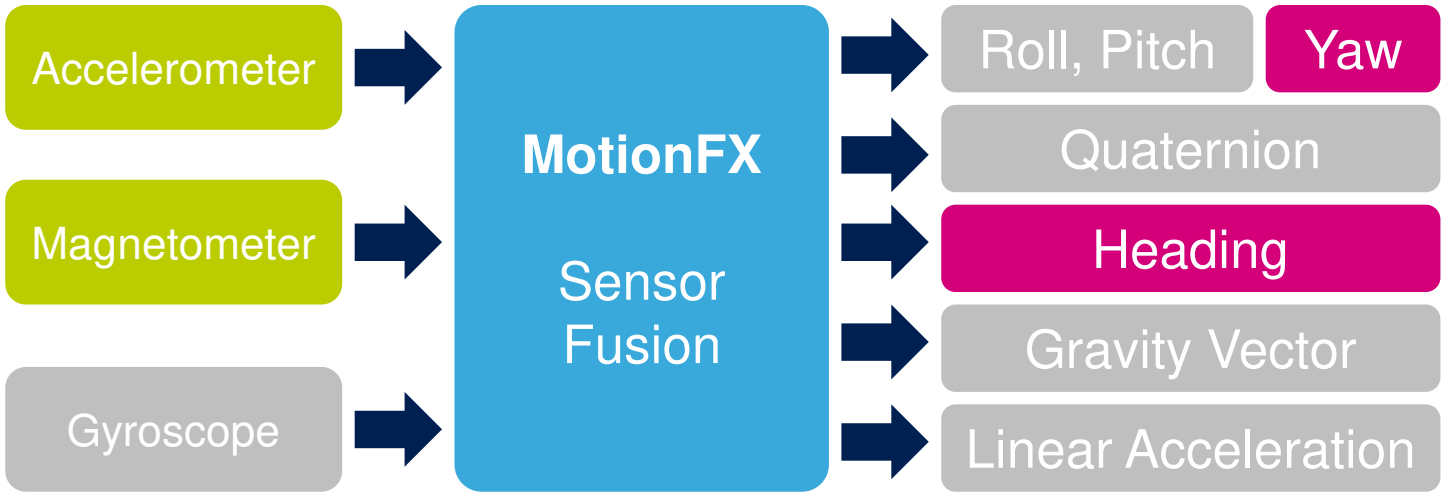
MotionFX library



- **Accelerometer gives roll** and **pitch angles** and the **gravity vector**…but **only in static conditions**! (or low dynamics)

Sensor Data Fusion

MotionFX library



- **Magnetometer gives yaw angle** and **heading**

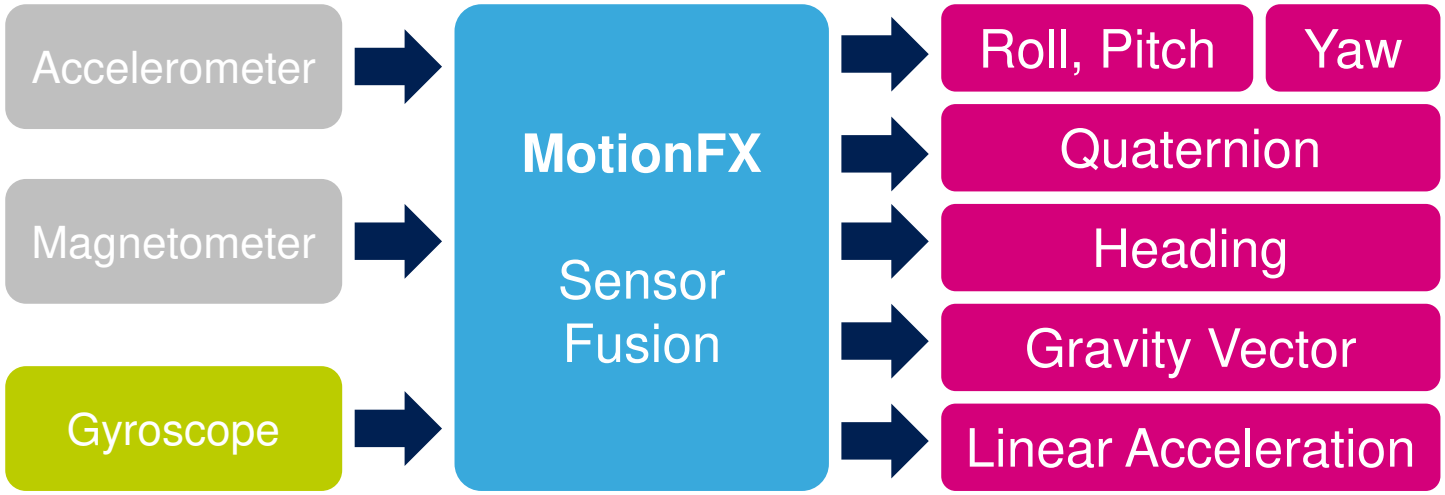
…but **only if tilt is compensated**: the accelerometer is needed!

…and **mag bias** (hard-iron) **must be compensated!**

life.augmented

Sensor Data Fusion

MotionFX library

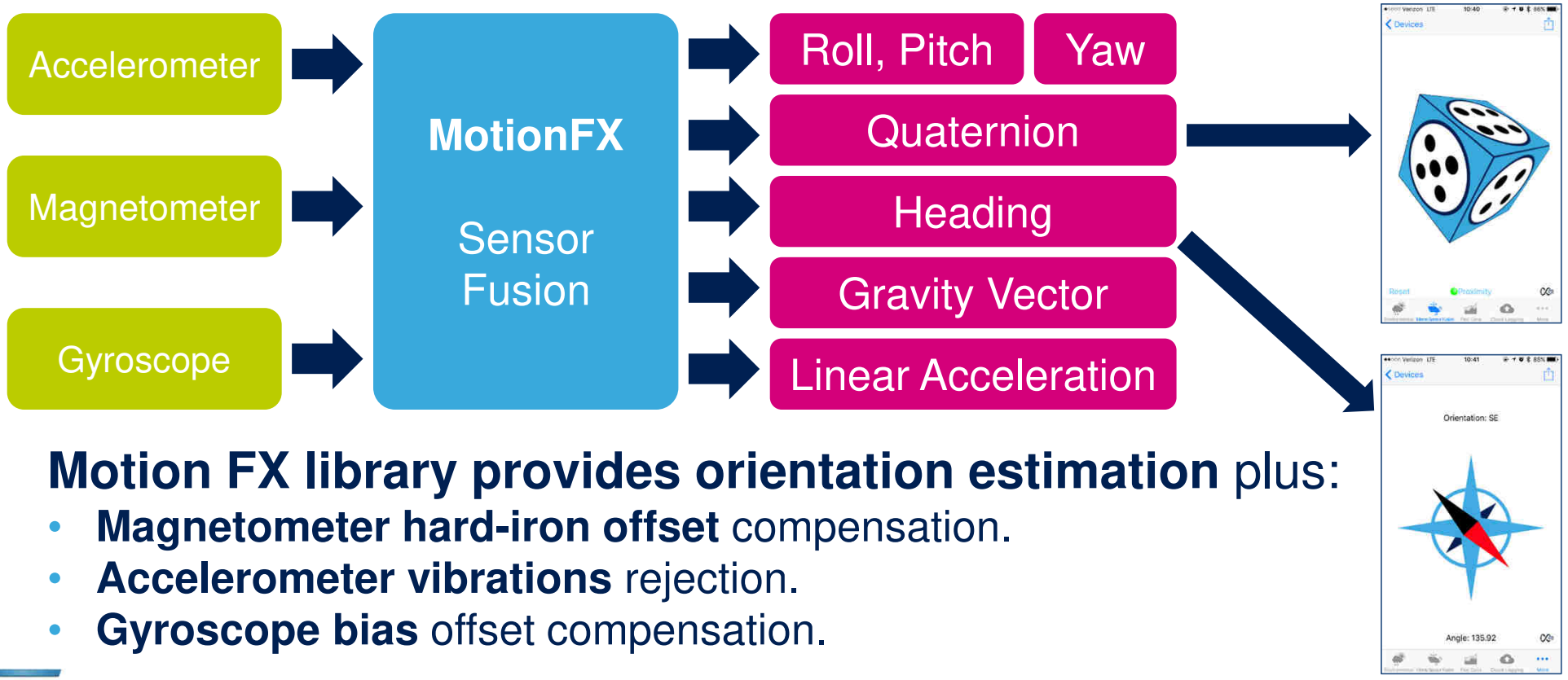


- **Gyroscope gives the new orientation** based on previous orientation

…but **gyro bias offset needs to be compensated** for accuracy!

Sensor Data Fusion

MotionFX library



Motion FX library provides orientation estimation plus:

- **Magnetometer hard-iron offset** compensation.
- **Accelerometer vibrations** rejection.
- **Gyroscope bias** offset compensation.

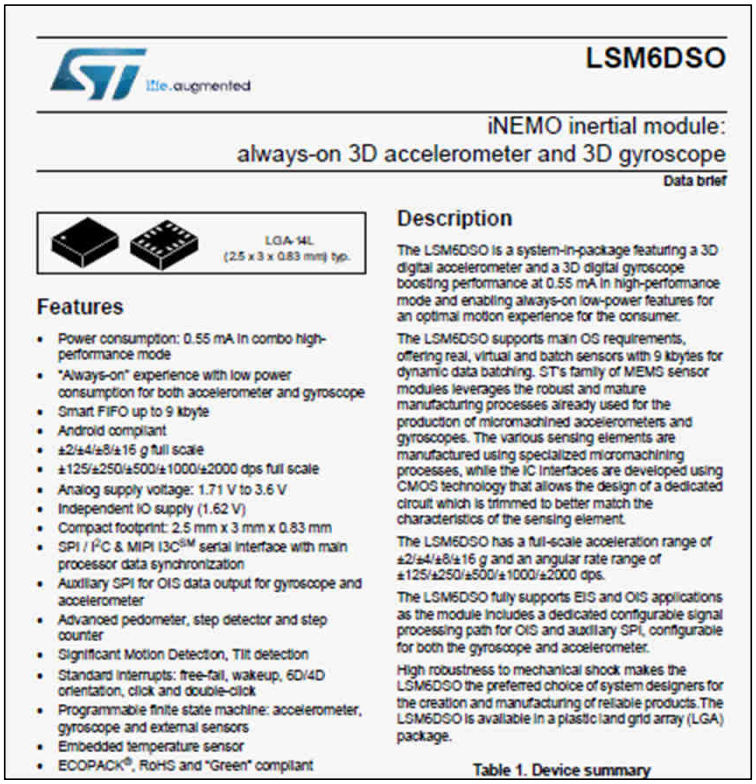
life.augmented

Gyroscope

The gyroscope is the most important sensor in the system
…works in highly dynamic conditions when the Acc cannot be used
…works with magnetic anomalies when the Mag cannot be used

LSM6DSO is an inertial module with **Acc** and **Gyro**

- **Exceptional noise performance** for maximum accuracy in orientation estimation
  - 90 ug/sqrtHz acc noise density
  - 3.8 mdps/sqrtHz gyro noise density
- **Extremely low-power**
  - **650 uA peak power** at ODR 6.66kS/sec with Acc+Gyro in high performance mode!



life.augmented

LSM6DSO

iNEMO inertial module:
always-on 3D accelerometer and 3D gyroscope

Data brief

LGA-14L
(2.5 x 3 x 0.83 mm) typ.

Features

- Power consumption: 0.55 mA in combo high-performance mode
- "Always-on" experience with low power consumption for both accelerometer and gyroscope
- Smart FIFO up to 9 kbyte
- Android compliant
- ±2/±4/±8/±16 *g* full scale
- ±125/±250/±500/±1000/±2000 dps full scale
- Analog supply voltage: 1.71 V to 3.6 V
- Independent IO supply (1.62 V)
- Compact footprint: 2.5 mm x 3 mm x 0.83 mm
- SPI / I²C & MIPI I3C℠ serial interface with main processor data synchronization
- Auxiliary SPI for OIS data output for gyroscope and accelerometer
- Advanced pedometer, step detector and step counter
- Significant Motion Detection, Tilt detection
- Standard interrupts: free-fall, wakeup, 6D/4D orientation, click and double-click
- Programmable finite state machine: accelerometer, gyroscope and external sensors
- Embedded temperature sensor
- ECOPACK®, RoHS and "Green" compliant

Description

The LSM6DSO is a system-in-package featuring a 3D digital accelerometer and a 3D digital gyroscope boosting performance at 0.55 mA in high-performance mode and enabling always-on low-power features for an optimal motion experience for the consumer.

The LSM6DSO supports main OS requirements, offering real, virtual and batch sensors with 9 kbytes for dynamic data batching. ST's family of MEMS sensor modules leverages the robust and mature manufacturing processes already used for the production of micromachined accelerometers and gyroscopes. The various sensing elements are manufactured using specialized micromachining processes, while the IC interfaces are developed using CMOS technology that allows the design of a dedicated circuit which is trimmed to better match the characteristics of the sensing element.

The LSM6DSO has a full-scale acceleration range of ±2/±4/±8/±16 *g* and an angular rate range of ±125/±250/±500/±1000/±2000 dps.

The LSM6DSO fully supports EIS and OIS applications as the module includes a dedicated configurable signal processing path for OIS and auxiliary SPI, configurable for both the gyroscope and accelerometer.

High robustness to mechanical shock makes the LSM6DSO the preferred choice of system designers for the creation and manufacturing of reliable products. The LSM6DSO is available in a plastic land grid array (LGA) package.

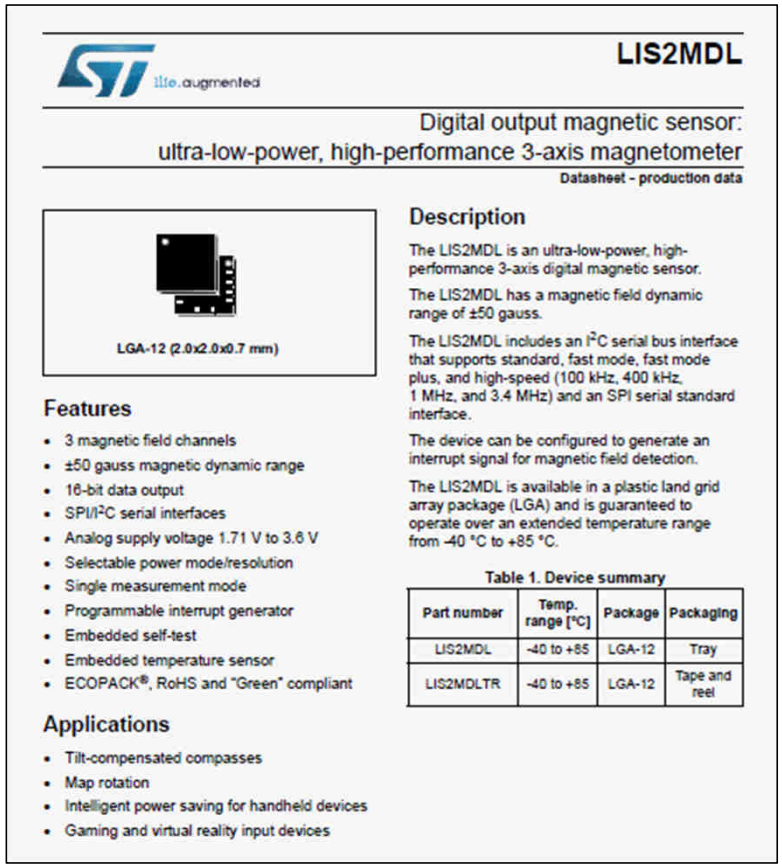
Table 1. Device summary

life.augmented

Magnetometer

LIS2MDL is **digital Mag**

- **Exceptional noise performance and dynamic range** for maximum accuracy in orientation estimation
  - 50 Gauss dynamic range
  - 3 mgauss RMS noise
- **Embedded offset compensation**
  - **Intrinsic offset is estimated and compensated automatically**
  - Extrinsic offset (hard-iron) must be estimated by host and can be compensated internally



life.augmented

LIS2MDL

Digital output magnetic sensor:
ultra-low-power, high-performance 3-axis magnetometer

Datasheet - production data

LGA-12 (2.0x2.0x0.7 mm)

Features

- 3 magnetic field channels
- ±50 gauss magnetic dynamic range
- 16-bit data output
- SPI/I²C serial interfaces
- Analog supply voltage 1.71 V to 3.6 V
- Selectable power mode/resolution
- Single measurement mode
- Programmable interrupt generator
- Embedded self-test
- Embedded temperature sensor
- ECOPACK®, RoHS and "Green" compliant

Applications

- Tilt-compensated compasses
- Map rotation
- Intelligent power saving for handheld devices
- Gaming and virtual reality input devices

Description

The LIS2MDL is an ultra-low-power, high-performance 3-axis digital magnetic sensor.

The LIS2MDL has a magnetic field dynamic range of ±50 gauss.

The LIS2MDL includes an I²C serial bus interface that supports standard, fast mode, fast mode plus, and high-speed (100 kHz, 400 kHz, 1 MHz, and 3.4 MHz) and an SPI serial standard interface.

The device can be configured to generate an interrupt signal for magnetic field detection.

The LIS2MDL is available in a plastic land grid array package (LGA) and is guaranteed to operate over an extended temperature range from -40 °C to +85 °C.

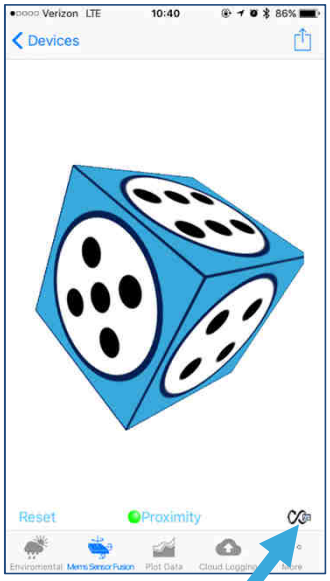
Table 1. Device summary

Part number	Temp. range [°C]	Package	Packaging
LIS2MDL	-40 to +85	LGA-12	Tray
LIS2MDLTR	-40 to +85	LGA-12	Tape and reel

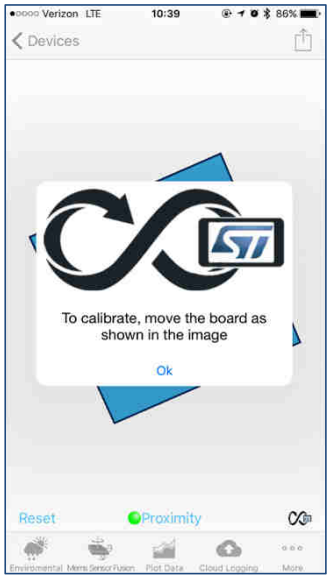
life.augmented

Magnetometer

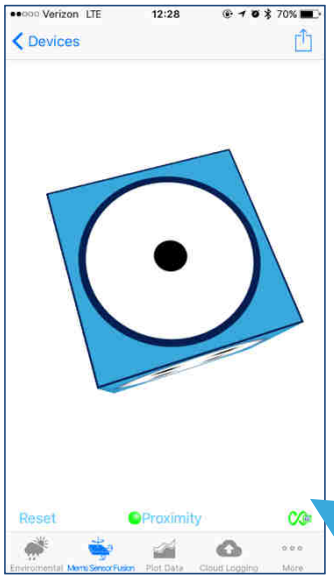
Calibration



Tap



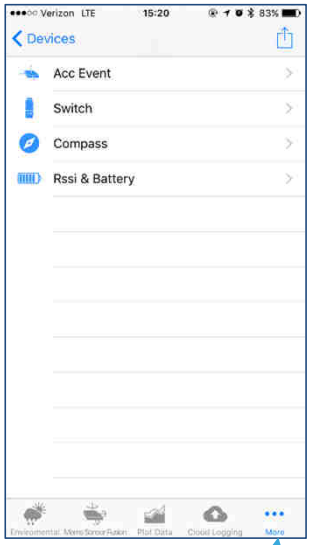
Move the BlueTile with the “**8 pattern**” shown in the figure to **calibrate the magnetometer**



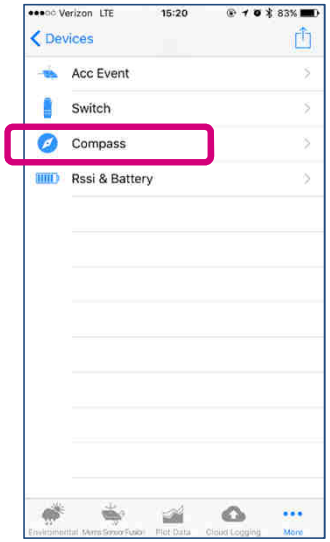
Calibration is **Completed** when the icon becomes **green**.

life.augmented

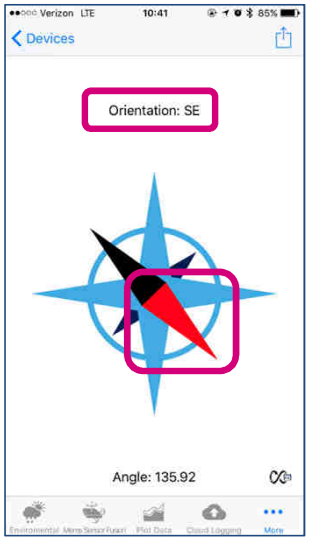
Magnetometer eCompass



Tap

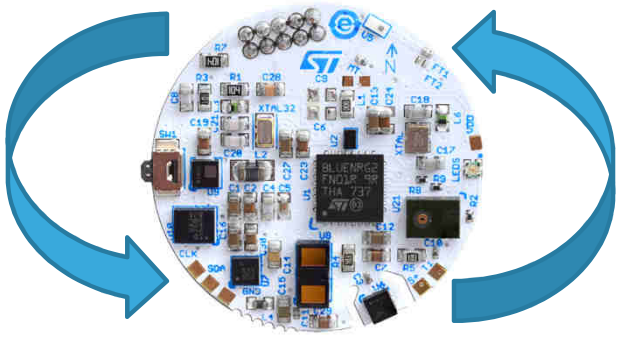


Select “Compass”



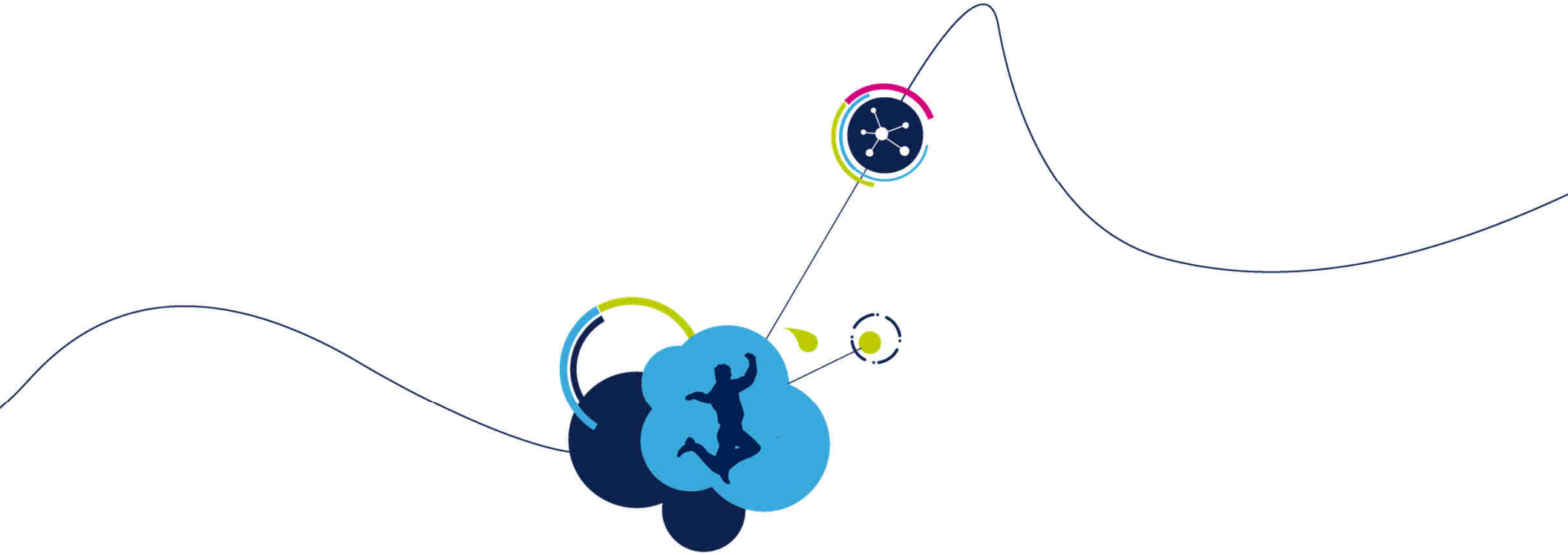
Red arrow is related to the current orientation of the BlueTile

Rotate the BlueTile



(you can check against the phone eCompass)

life.augmented



Lab 6

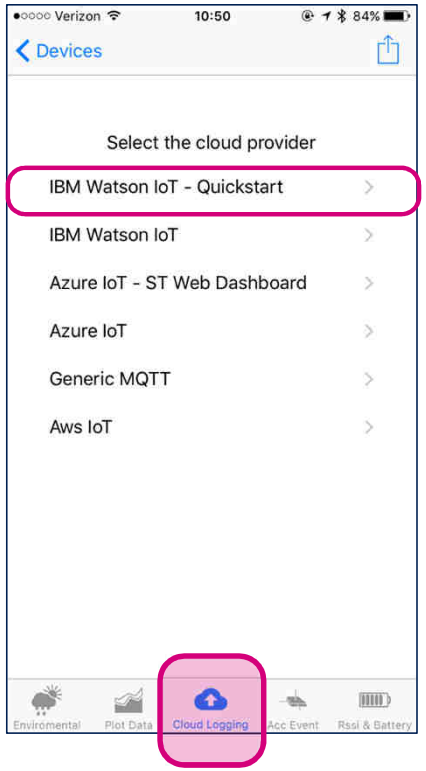
Cloud data logging on IBM Watson

life.augmented

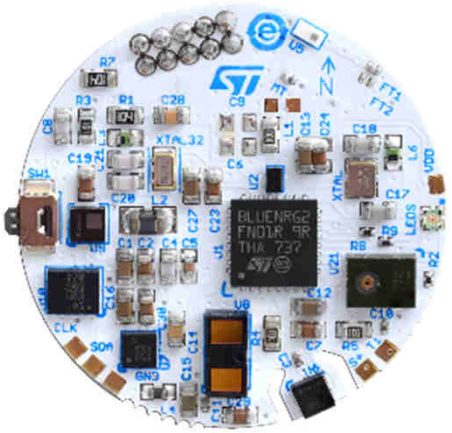
Cloud Logging

1. **Enable** IBM Watson IoT – Cloud Logging
2. **Send** BLE notification packets on Sensor status
3. **Visualize** the data

IBM Watson



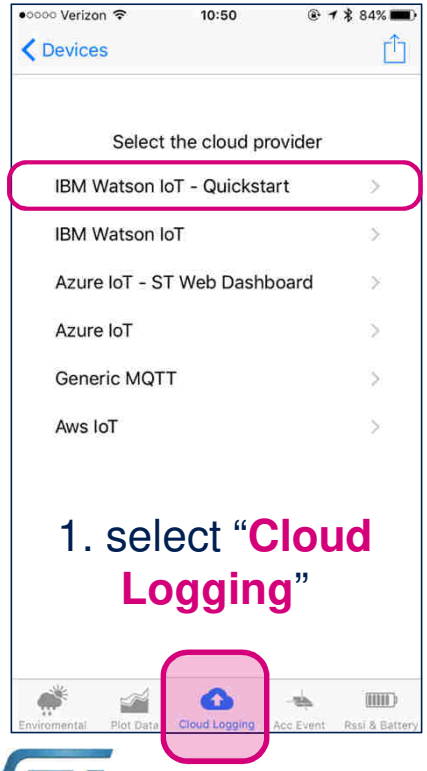
Notification Packets



life.augmented

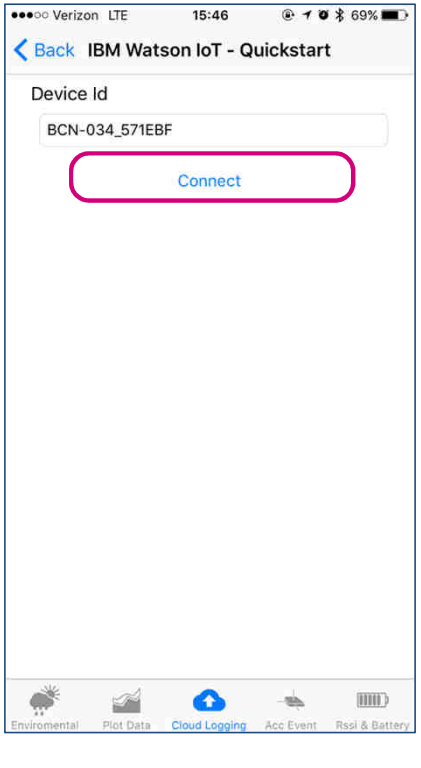
IBM Watson IoT Quickstart

2. **Select "IBM Watson IoT - Quickstart"**

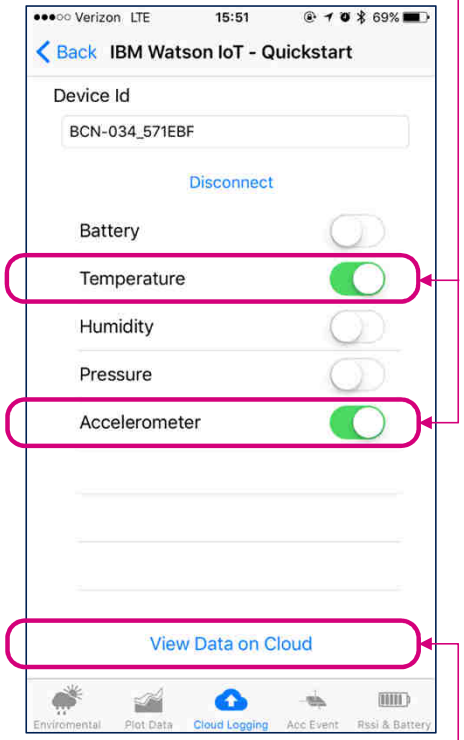


1. select "**Cloud Logging**"

Click "Connect"

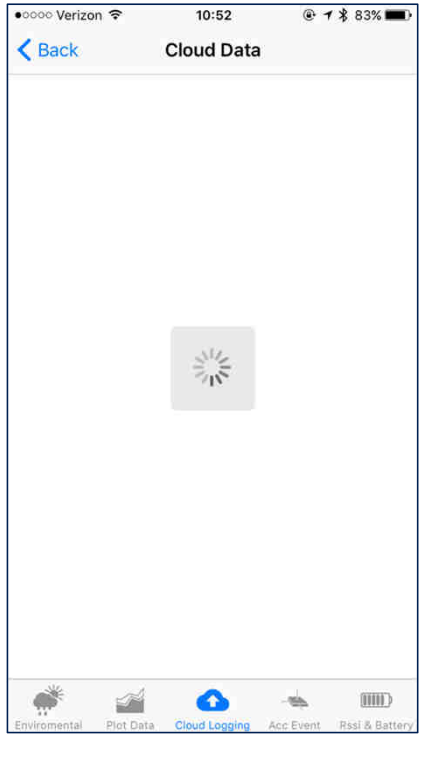


1. **Select** one or multiple features



2. **Click "View Data on Cloud"**

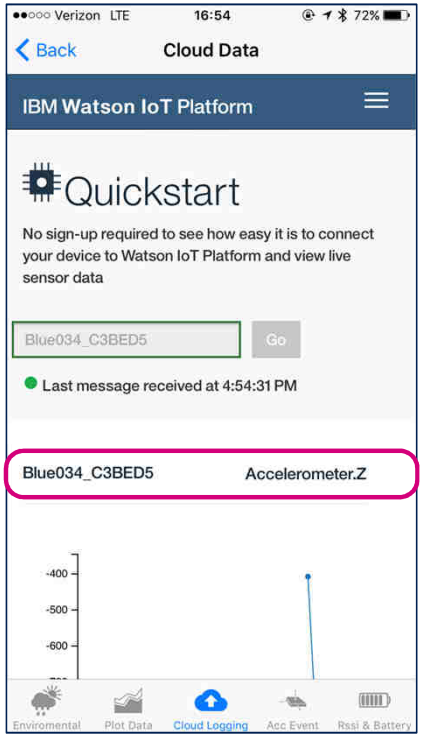
Wait a few seconds



life.augmented

IBM Watson IoT Quickstart

Quickstart will appear



Scroll down to see your selected sensor plot.

You will see the **Plot** of selected feature

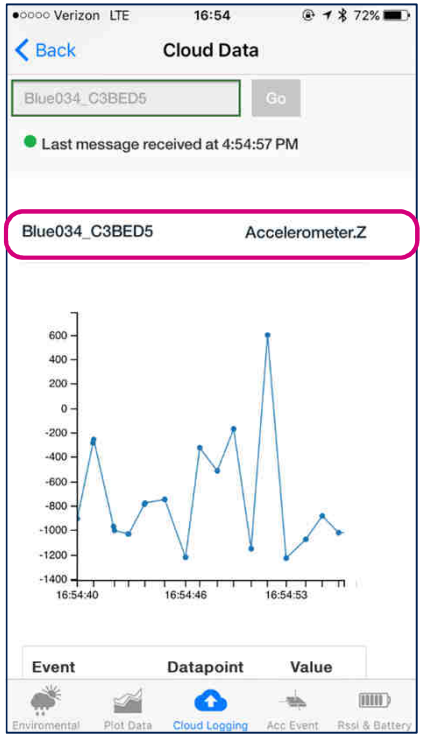
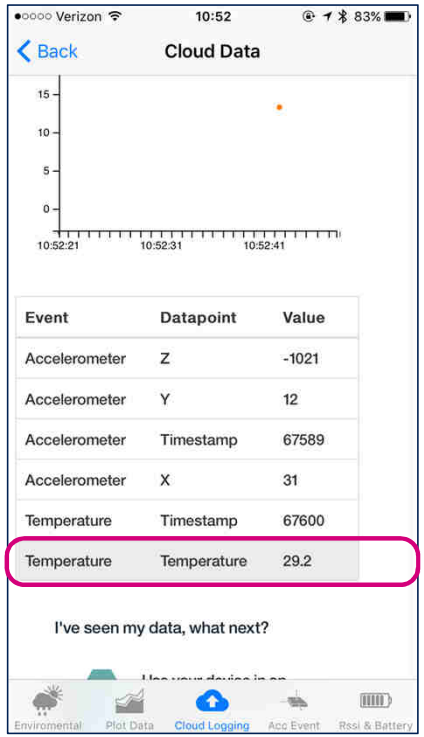
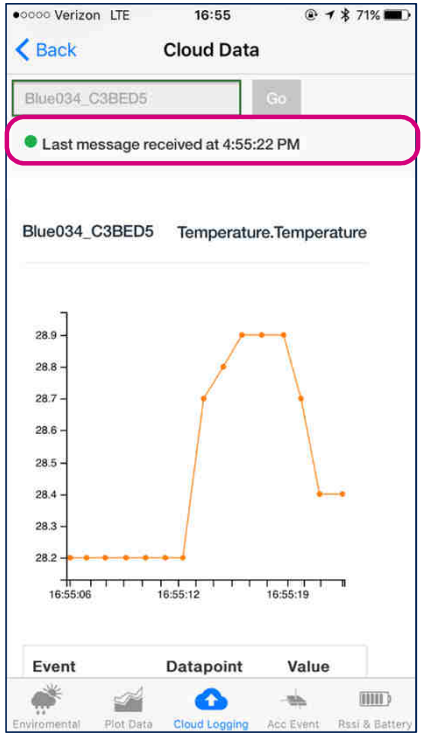


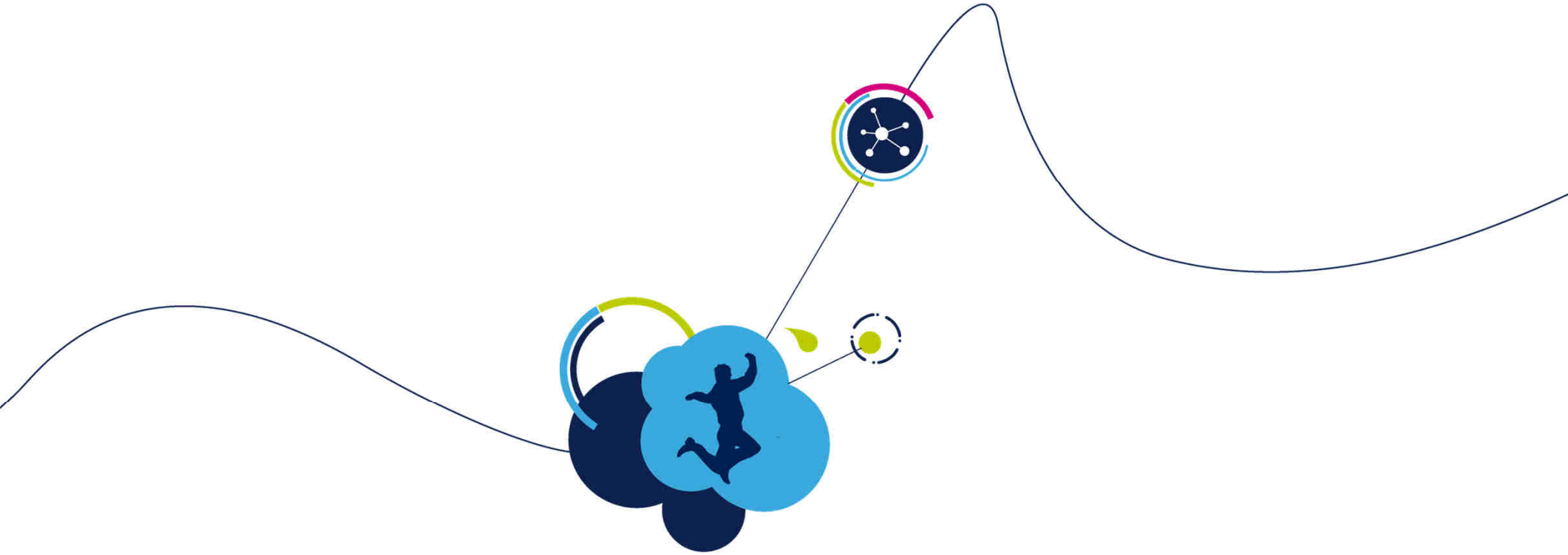
Table of **available features**





Scroll down again to change sensor data or axes

life.augmented



Lab 7

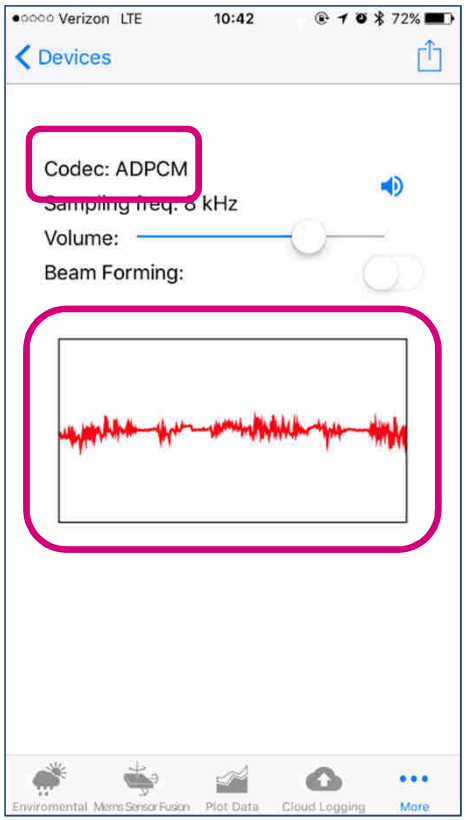
Voice over BLE

life.augmented

Goal

- In this example we are going to demonstrate how to:
  - **Enable the BlueVoice library** for Voice Over BLE streaming
    - Input: raw data from the MEMS microphone
    - Output: ADPCM data streaming @8kHz
  - **Send ADPCM data** through **BLE notifications packets** to the ST BLE Sensor app

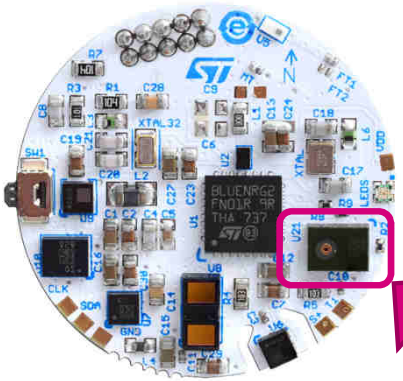
BlueVoice library



1. **Enable BlueVoice library**
2. **Send voice** to the ST BLE Sensor app client through **BLE notification packet**

Notification Packet

ADPCM compressed audio



MEMS Microphone

life.augmented

ST BlueMS Protocol

BlueMS

4 Bytes

Len	Type	Protocol version	Device ID	Feature Mask	Device MAC	Len	Type	Local Name	Len	Type	Pwr

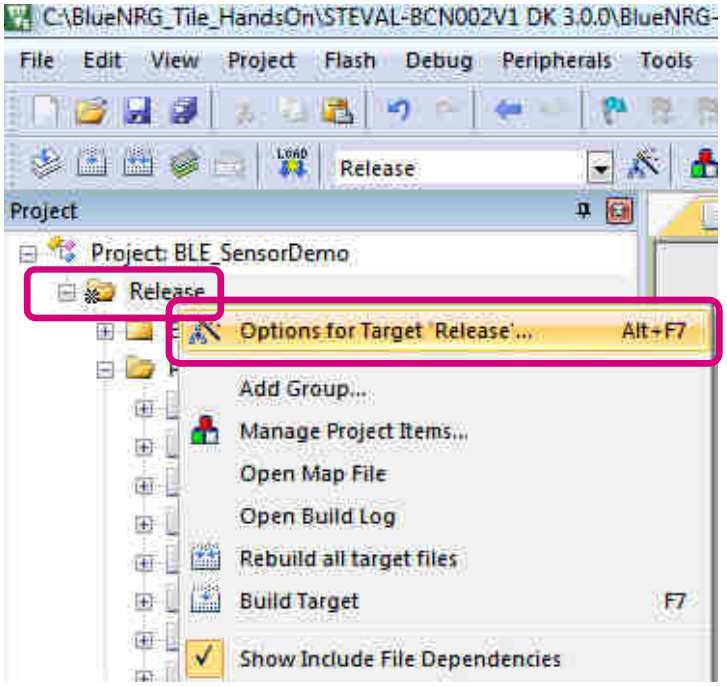
31	30	29	28	27	26	25	24	
RFU	ADPCM	Switch	DoA	ADPC	MicLevel	Proximity	Lux	0x6A
23	22	21	20	19	18	17	16	
Acc	Gyro	Mag	Pressure	Humidity	Temperat	Battery	2nd Temp	0xFE
15	14	13	12	11	10	9	8	
RFU	RFU	RFU	RFU	Beamform	AccEvent	FreeFall	SensFusC	0x05
7	6	5	4	3	2	1	0	
SensFus	Compass	MotionInt	Activity	Carry Pos	MemsGes	ProxGes	Pedo	0x40

life.augmented

Enable BlueVoice library

Add the preprocessor symbol **ENABLE_AUDIO** to the project

1. **Right-click** on **Release**
2. **Select** “**Options for Target ‘Release’…**”

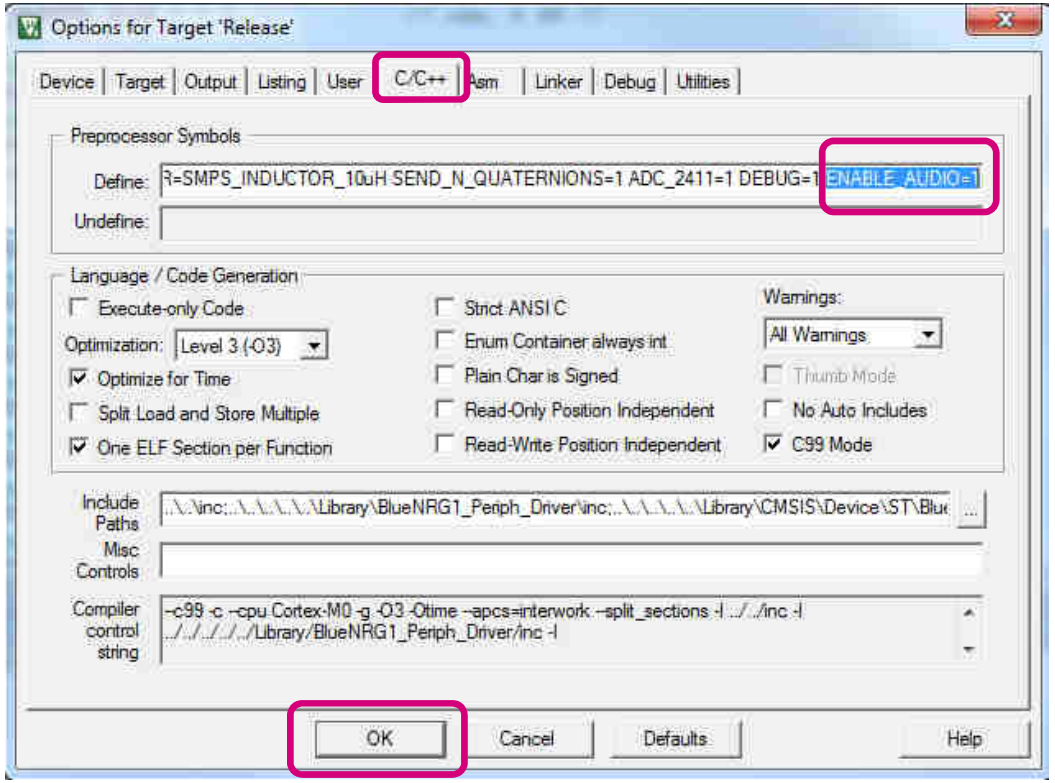


life.augmented

Enable BlueVoice library

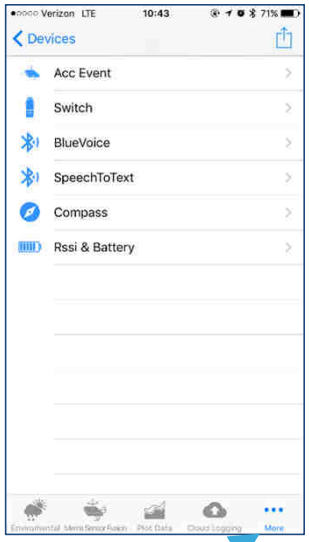
In the project options:

1. **Go to** the tab **C/C++** and then in the **Preprocessor Symbols**
2. **Set the symbol ENABLE_AUDIO=1** and then **click** on **OK**



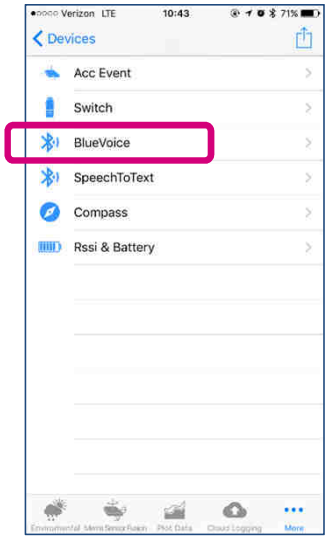
life.augmented

BlueVoice: voice over Bluetooth LE

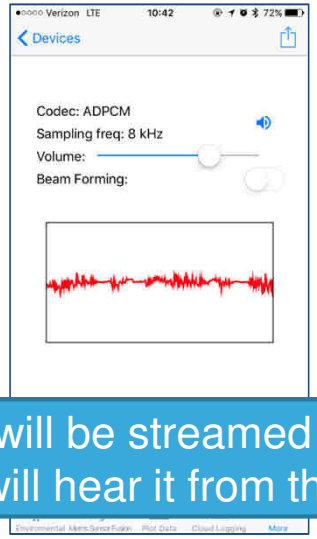


Tap

Select
"BlueVoice"

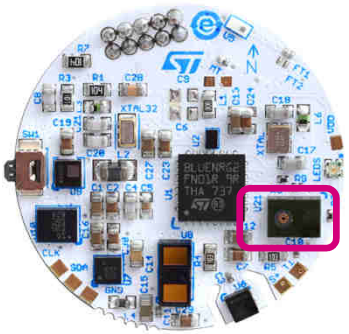


Speak close to the BlueNRG-Tile



Voice will be streamed over BLE
You will hear it from the phone

Do not silence your phone,
must **NOT** be vibration only!

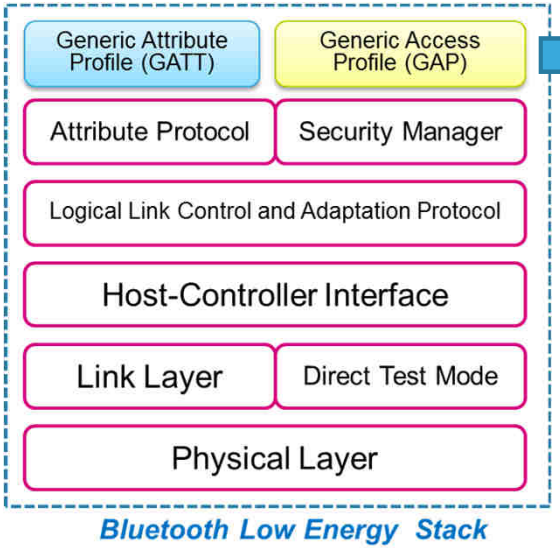


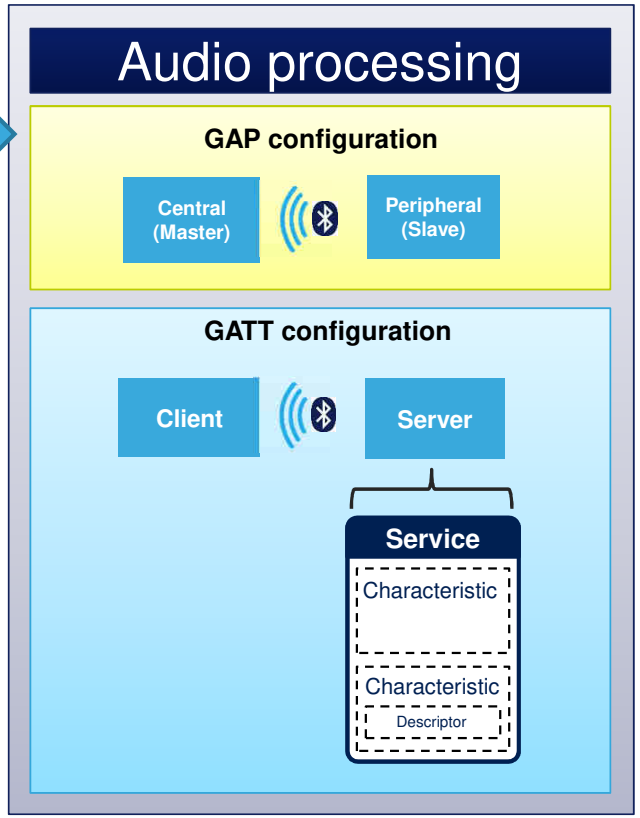
MEMS Microphone

(if the mic captures the audio from the phone speaker, a very high pitch sound can happen!)

life.augmented

Voice over Bluetooth LE





It's a Vendor Specific Service!

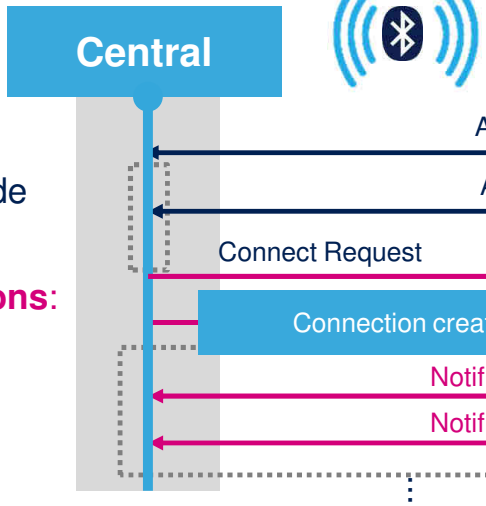
Audio: exported by the Server through **2 dedicated BLE characteristics**

Voice streaming over BLE:
Audio @8kHz
Codec: ADPCM
Bitrate: 32kbps

life.augmented

BlueVoice ADPCM - Audio 8kHz

Discovery Mode



Advertisement Mode

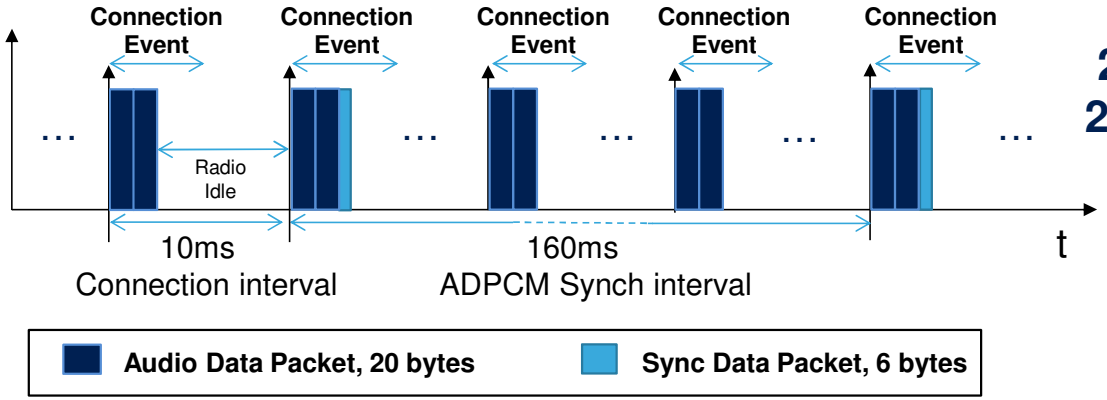
No acknowledge required for **Notifications**:

- OK for Audio streaming
- **Minimal Latency**
- **Optimal Bandwidth**

Audio can be half-duplex

Voice Streaming

- Audio Format: **PCM 16 bit @ 8 kHz**
- **ADPCM Compression** @ 32 Kbps: Low latency and low complexity
- Side Information at low data rate enhances error resilience



2 packets/event
20 bytes/packets
100 events/sec

=

32kbps

life.augmented

BlueVoice architecture

"Packetization"

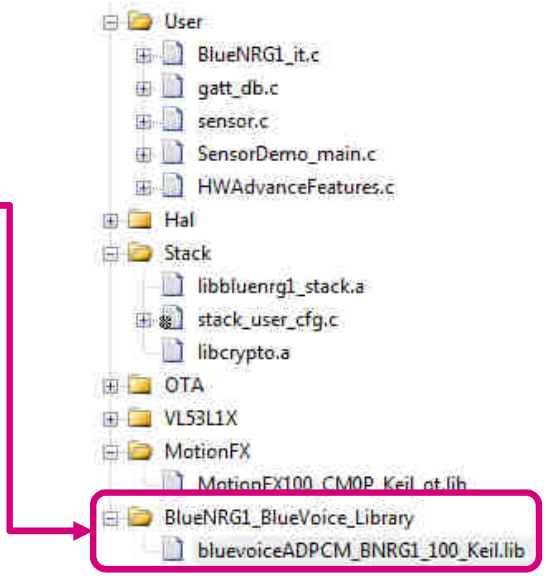
ST digital MEMS microphone

BlueNRG-2

Audio Acquisition → PDM → PDM to PCM conversion → PCM 8kHz → ADPCM Compression → Raw Data → Server - TX Bluetooth SMART

HW implementation

Bitrate 32 kbps

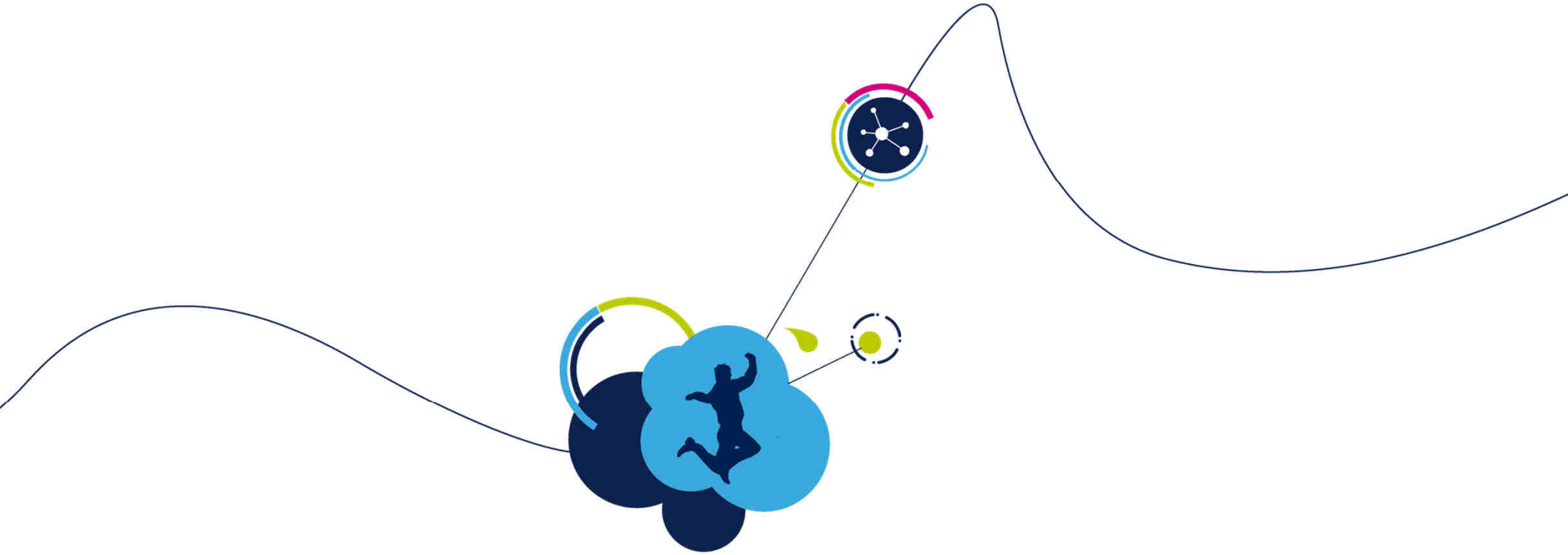


"De-Packetization"

Smartphone app (ST BLE Sensor app)

Client - RX Bluetooth SMART → Raw Data → Audio Decompression → PCM → Serial Audio Out USB, I2S, …

life.augmented



Lab 8

Over-The-Air (OTA) FW upgrade

life.augmented

Flash memory layout

0x1007E800

Empty

0x1005F800

BLE_SensorDemo_
LowerApp_OTA

0x10040800

BLE_OTA_ResetManager

0x10040000

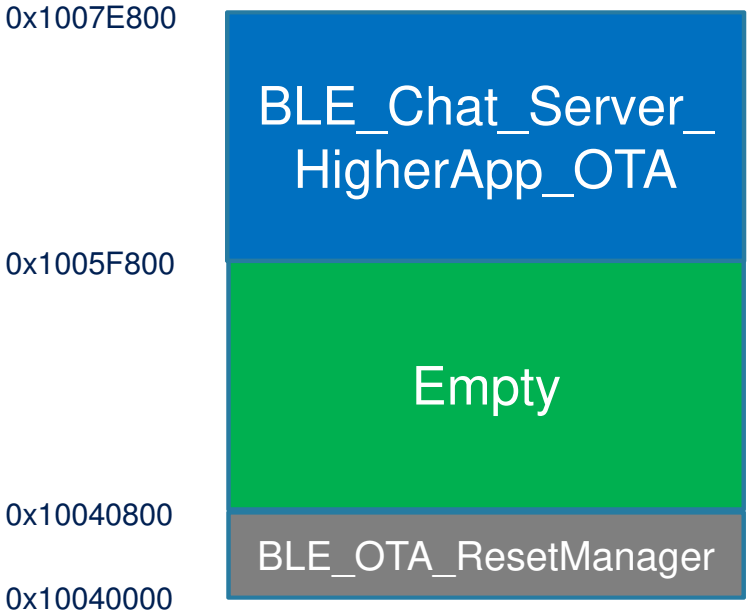
OTA

BLE_Chat_Server_HigherApp_OTA.bin

REMEMBER TO LOAD THE BLE OTA Reset manager!!!

life.augmented

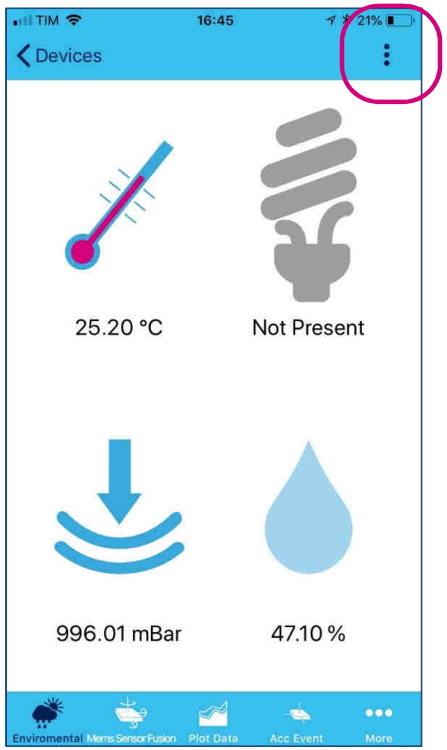
Once OTA is completed…



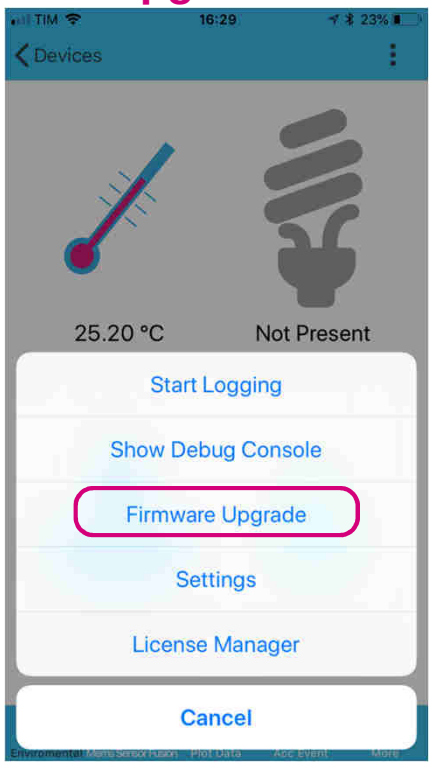
life.augmented

OTA FW upgrade (1/2)

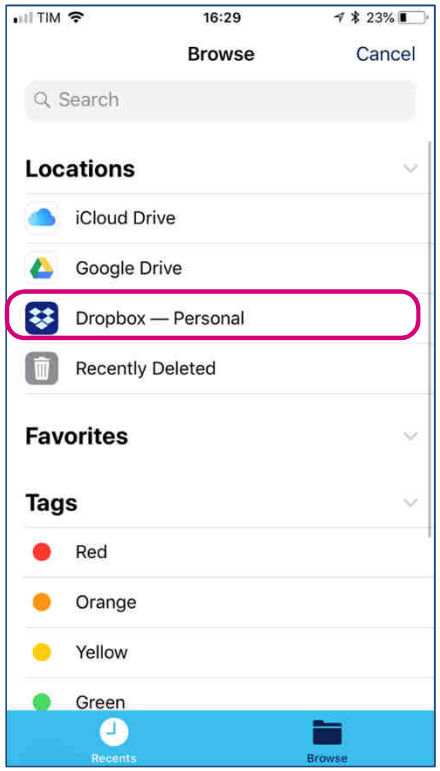
Click on the top right corner



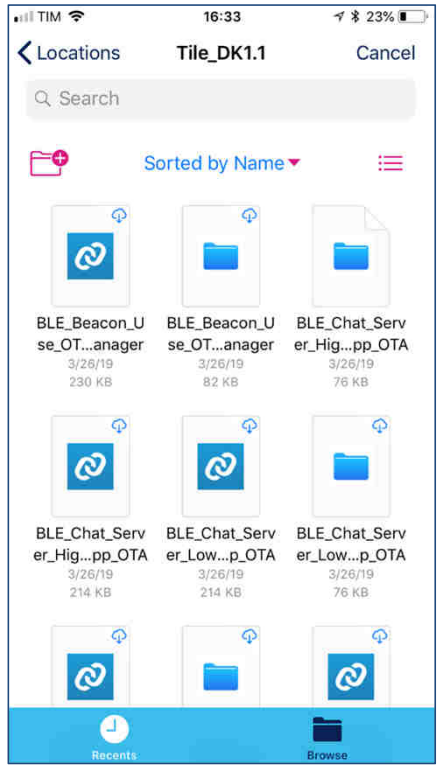
Select "**Firmware Upgrade**"



Select your remote repository



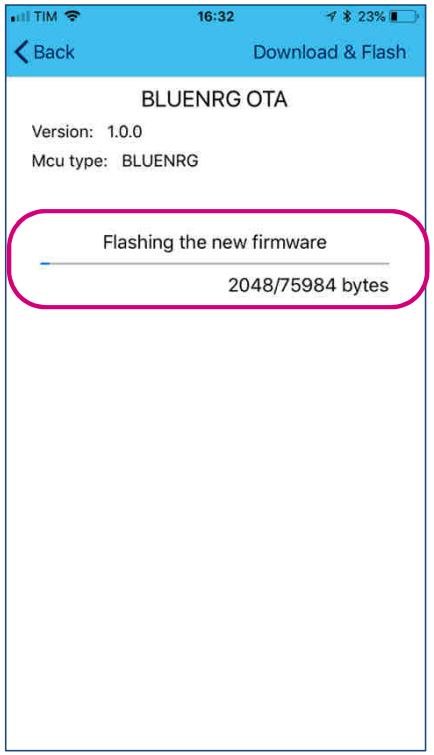
Select the binary file



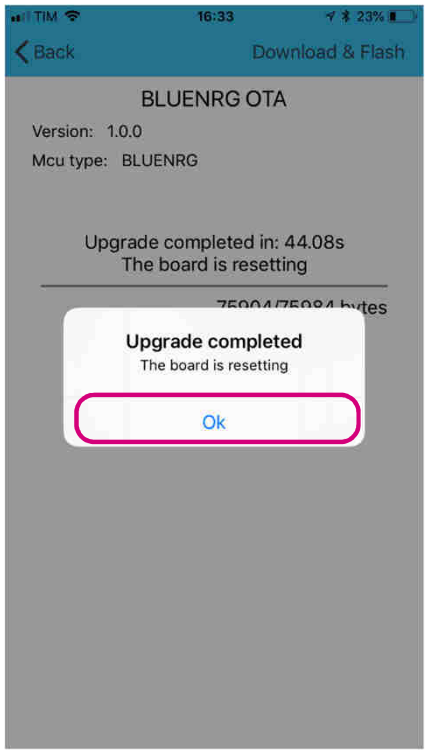
life.augmented

OTA FW upgrade (2/2)

OTA data transfer begins

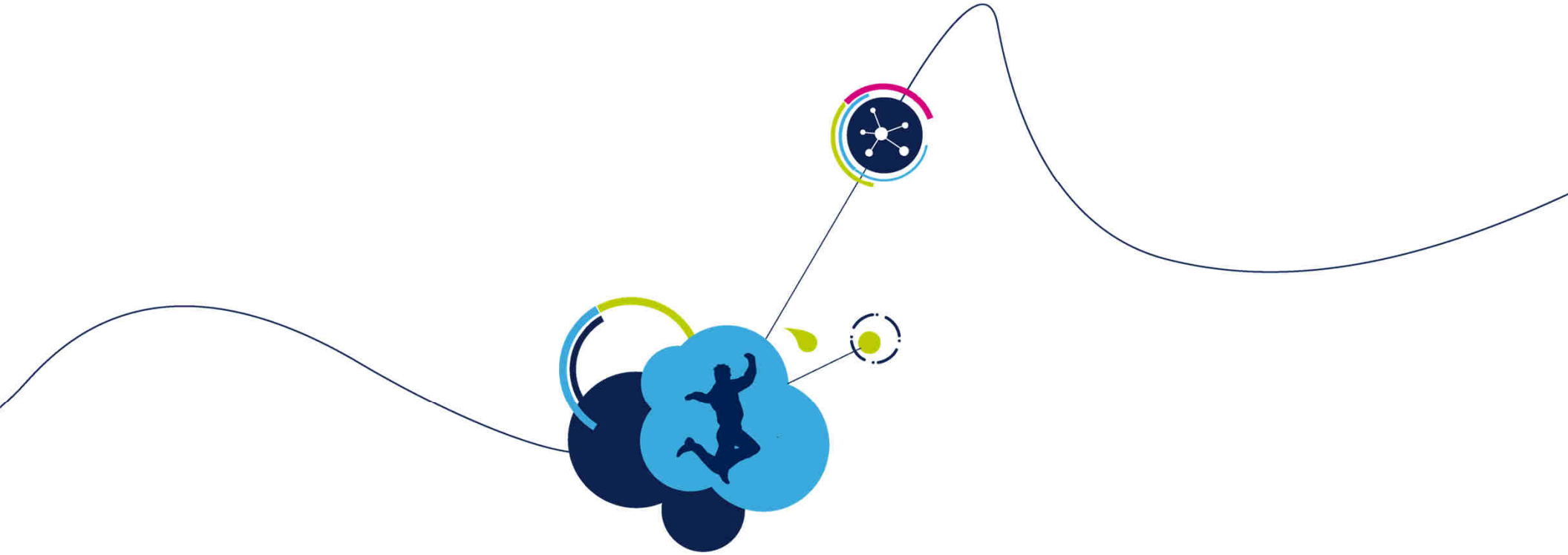


Click OK as completed



Re-Connect to BlueTile, now running a new application

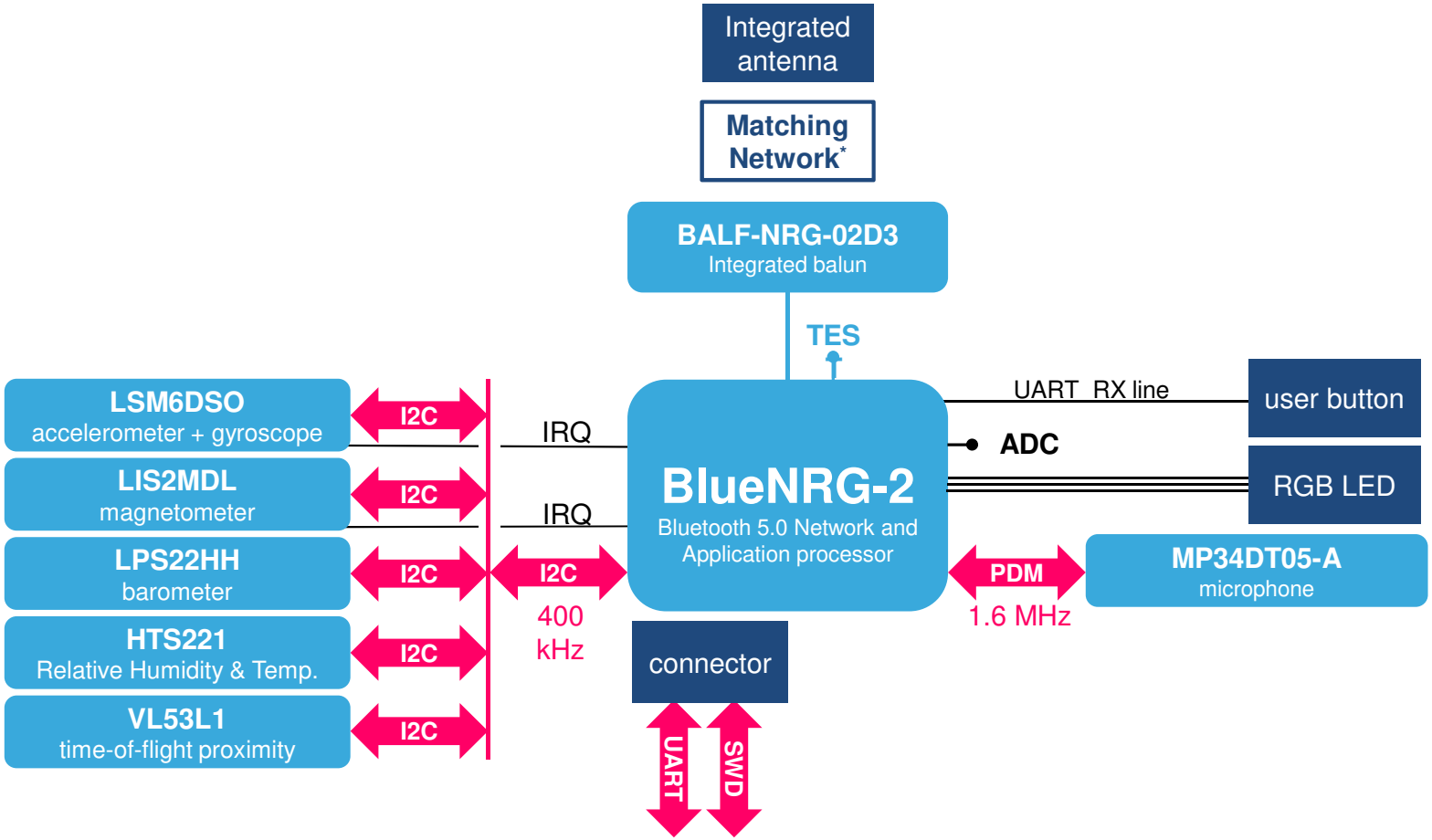
life.augmented



Customizing your design

life.augmented

STEVAL-BCN002V1B Block Diagram



life.augmented

Enable/Disable Sensors & Libraries

- **Dedicated structure** “**FeaturePresence**” for enabling/disabling sensors & libraries individually. File **sensor.h**

```
typedef struct {
  bool AccelerometerGyroscopePresence;
  bool MagnetometerPresence;
  bool HumidityTemperaturePresence;
  bool PressurePresence;
  bool ProximityLightPresence;
  bool iNemoEngine;
  bool Pedometer;
} FeaturePresence;
```

- **SensorScan() function** set each field of the structure to **false** or **true**. File **sensor.c**

Disable the unneeded sensor for optimizing the power consumption!

```
// Check sensor list
SensorsScan();

// Configure discovered sensors
if (xFeaturePresence.PressurePresence)
  Init_Pressure_Temperature_Sensor();
if (xFeaturePresence.HumidityTemperaturePresence)
  Init_Humidity_Sensor();
if (xFeaturePresence.MagnetometerPresence)
  Init_Magnetometer();
if (xFeaturePresence.AccelerometerGyroscopePresence)
  Init_Accelerometer_Gyroscope();
if (xFeaturePresence.ProximityLightPresence)
  Init_Proximity_Sensor();

// Configure sensors in low power mode
SensorsLowPower();
```

BLE timings

- **Advertising intervals**:
  - Dedicated API *aci_gap_set_discoverable(Advertising_Type, Advertising_Interval_Min, Advertising_Interval_Max, …)*
  - In file **sensor.c**
- **Connection intervals**:
  - Dictated by the Central device. Peripheral has no full control on this.
- **Notifications frequency**
  - Dedicated Virtual Timers (mapped on HW physical timers) for different functionalities
  - Timeouts defined in **sensor.h**

```
#define BATTERY_UPDATE_RATE     1000    // Fixed ODR @  1 Hz
#define ENV_SENSOR_UPDATE_RATE   100     // Fixed ODR @  10 Hz
#define MOTION_SENSOR_UPDATE_RATE   40    // Fixed ODR @ 25 Hz
```

HW modifications

- In this case redesign is of course necessary
- **Schematics** and **Gerbers** files:
  - online at www.st.com/bluetile
- BlueNRG-2 **pin mapping**
  - Check **BlueNRG-2 DS** at **Table 129**

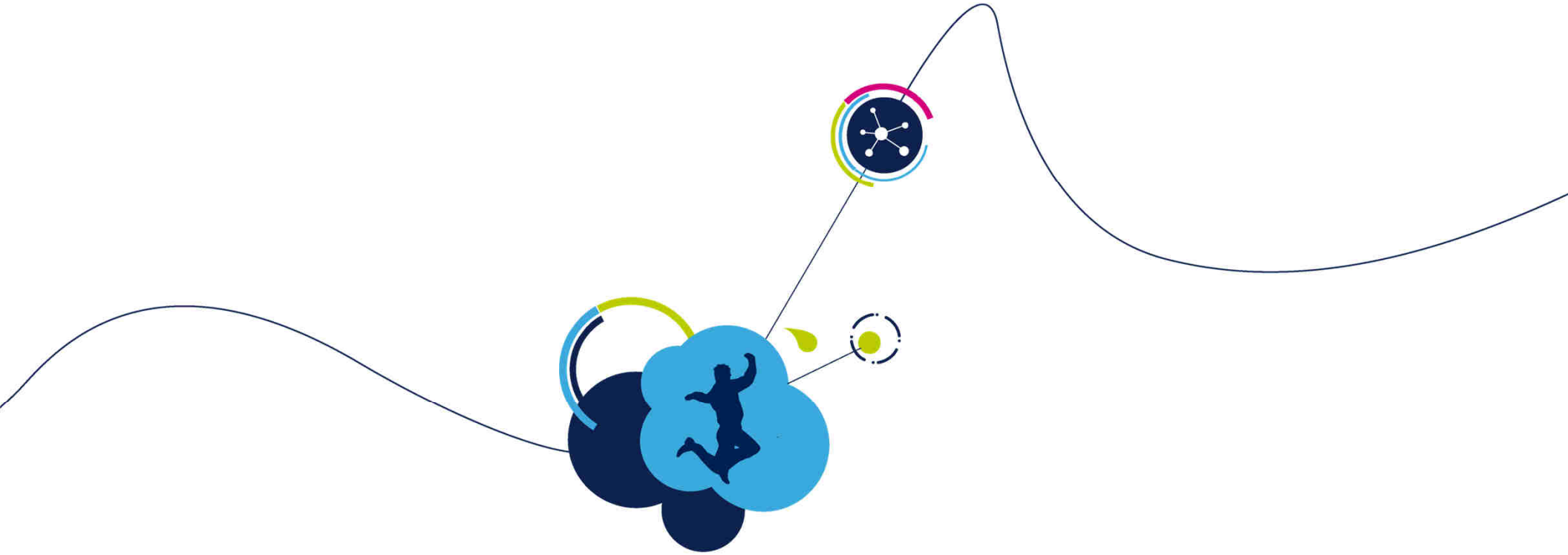
3.12.2 **Functional description**

The table below shows the GPIO configuration table where each IO pin is associated with related functions.

Table 129. **IO functional map**

Pin name(1)	GPIO mode "000"		Serial1 mode '001"		Serial0 mode '100'		Serial2 mode '101'	
	Type	Signal	Type	Signal	Type	Signal	Type	Signal
IO0	I/O	GPIO 0	I	UART_CTS	I/O	SPI_CLK	O	CPUCLK
IO1	I/O	GPIO 1	O	UART_RTS	I/O	SPI_CS1	I	PDM_DATA
IO2	I/O	GPIO 2	O	PWM0	O	SPI_OUT	O	PDM_CLK
IO3	I/O	GPIO 3	O	PWM1	I	SPI_IN	-	-
IO4	I/O	GPIO 4	I	UART_RXD	I/O	I2C2_CLK	O	PWM0
IO5	I/O	GPIO 5	O	UART_TXD	I/O	I2C2_DAT	O	PWM1
IO6	I/O	GPIO 6	O	UART_RTS	I/O	I2C2_CLK	I	PDM_DATA
IO7	I/O	GPIO 7	I	UART_CTS	I/O	I2C2_DAT	O	PDM_CLK
IO8	I/O	GPIO 8	O	UART_TXD	I/O	SPI_CLK	I	PDM_DATA
IO9	I/O	GPIO 9	I	SWCLK	I	SPI_IN	O	XO16/32M
IO10	I/O	GPIO 10	I	SWDIO	O	SPI_OUT	O	CLK_32K
IO11	I/O	GPIO 11	I	UART_RXD	I/O	SPI_CS1	O	CLK_32K
IO12	OD	GPI 12 (2)		-	I/O	I2C1_CLK	-	-
IO13	OD	GPI 13(2)	I	UART_CTS	I/O	I2C1_DAT	-	-
IO14	I/O	GPIO 14	I/O	I2C1_CLK	I/O	SPI_CLK	-	-

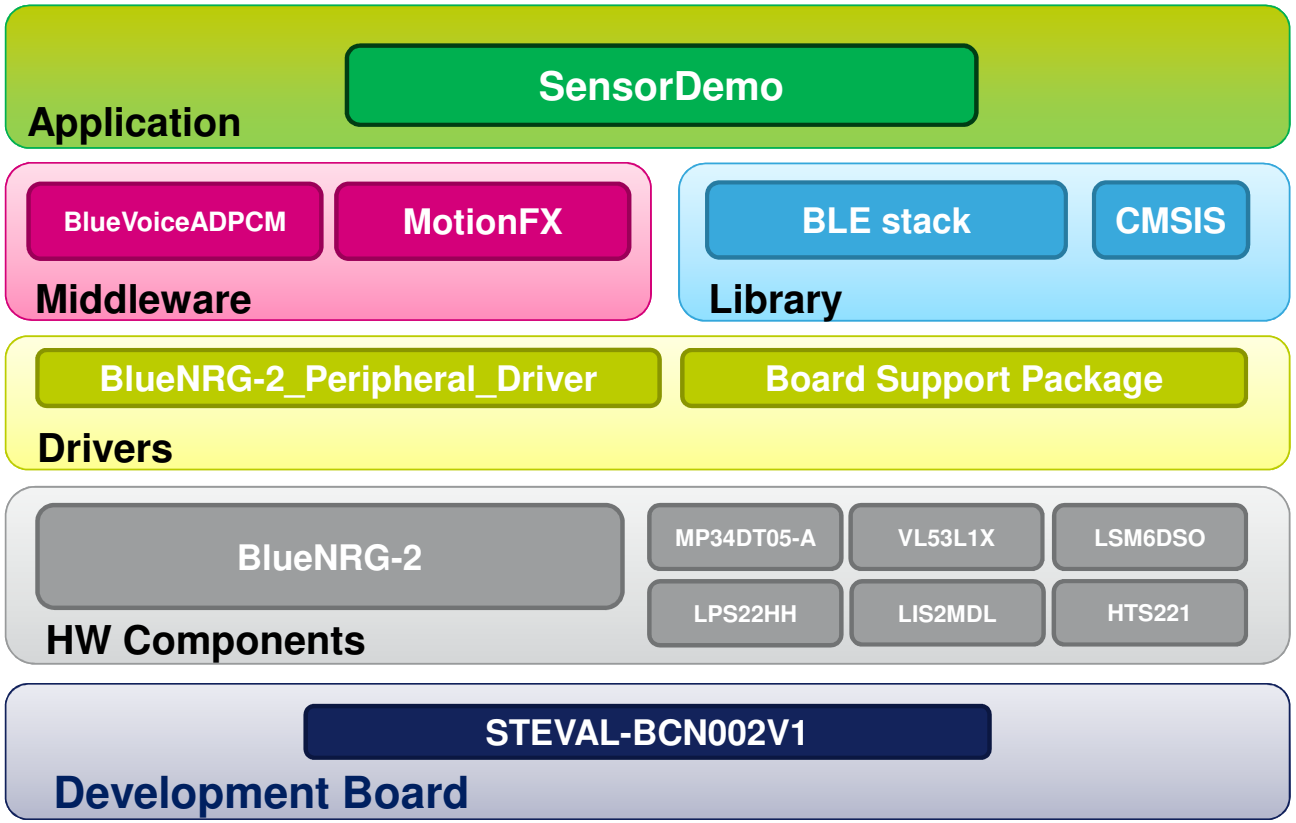
life.augmented



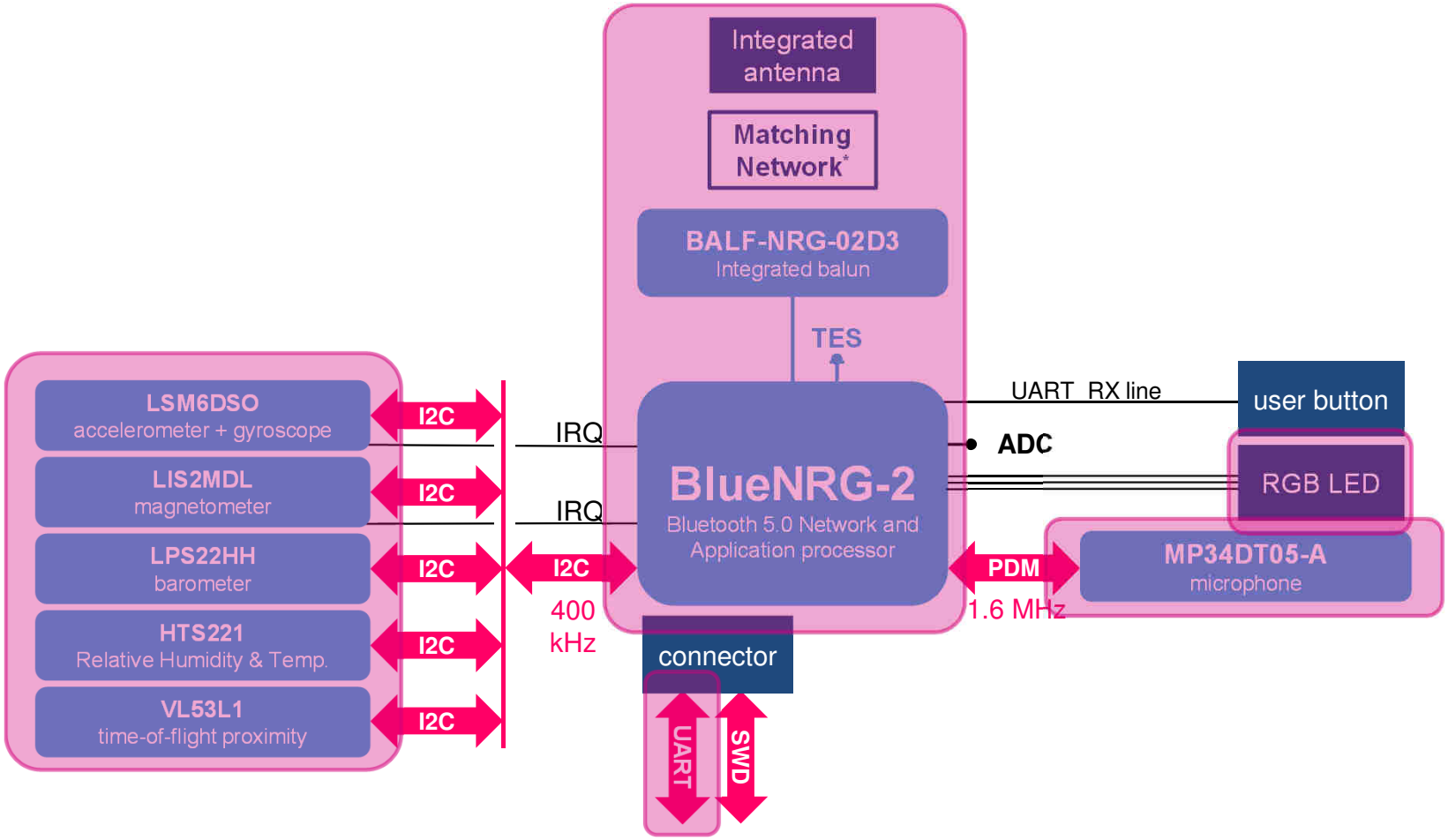
Quick recap

life.augmented

DK 3.0.0 SW architecture



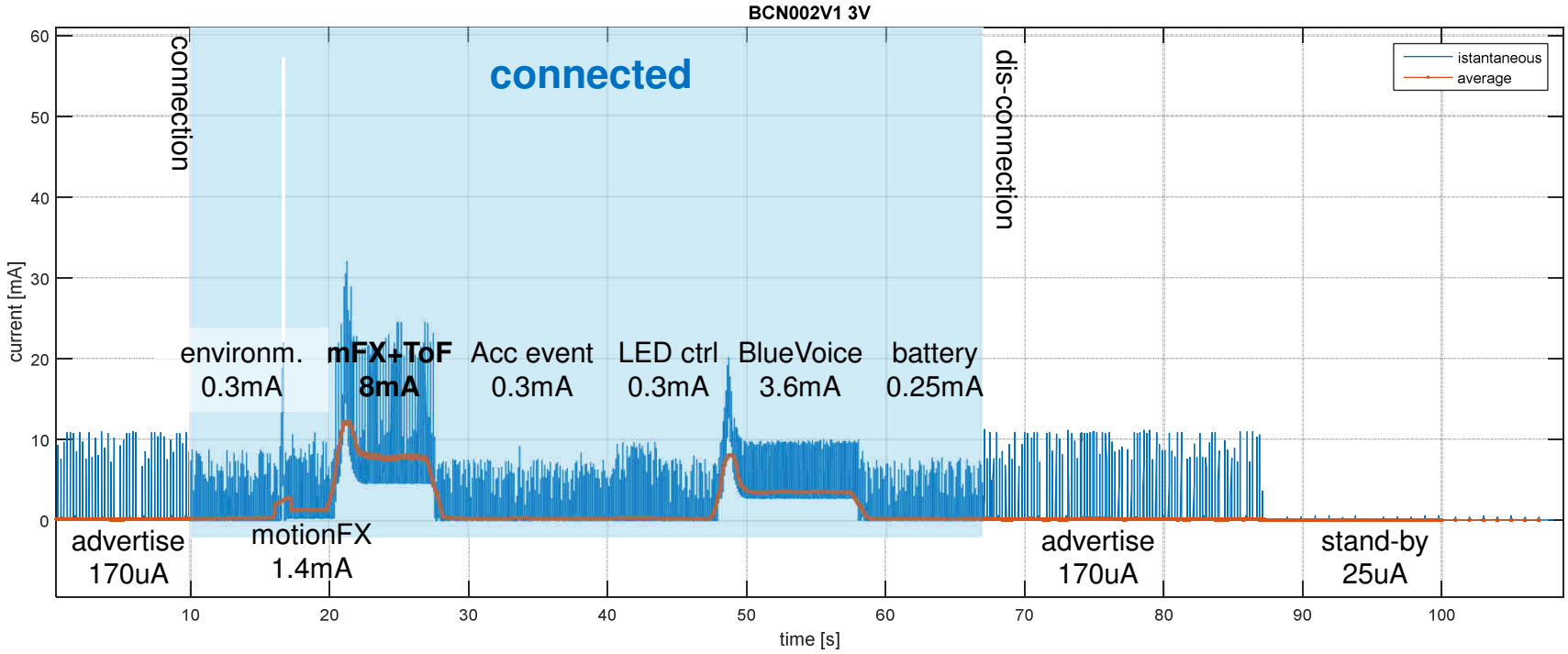
STEVAL-BCN002V1B Block Diagram



life.augmented

Power consumption

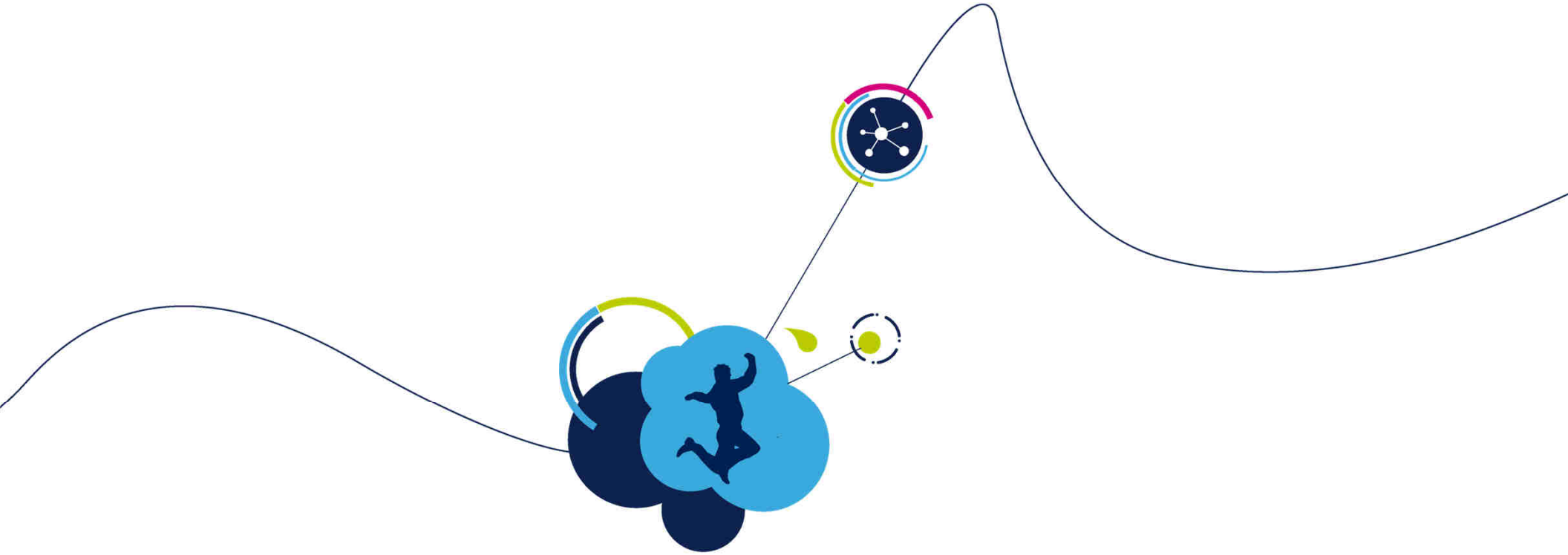
State	Avg power cons
stand-by	25uA
advertise 250ms	170uA
Battery notification	0.25mA
Environmental / AccEvents / LED control	0.3mA
Motion FX (Inertial)	1.4mA
BlueVoice	3.6mA
Motion FX plus Time Of Flight	8mA



life.augmented

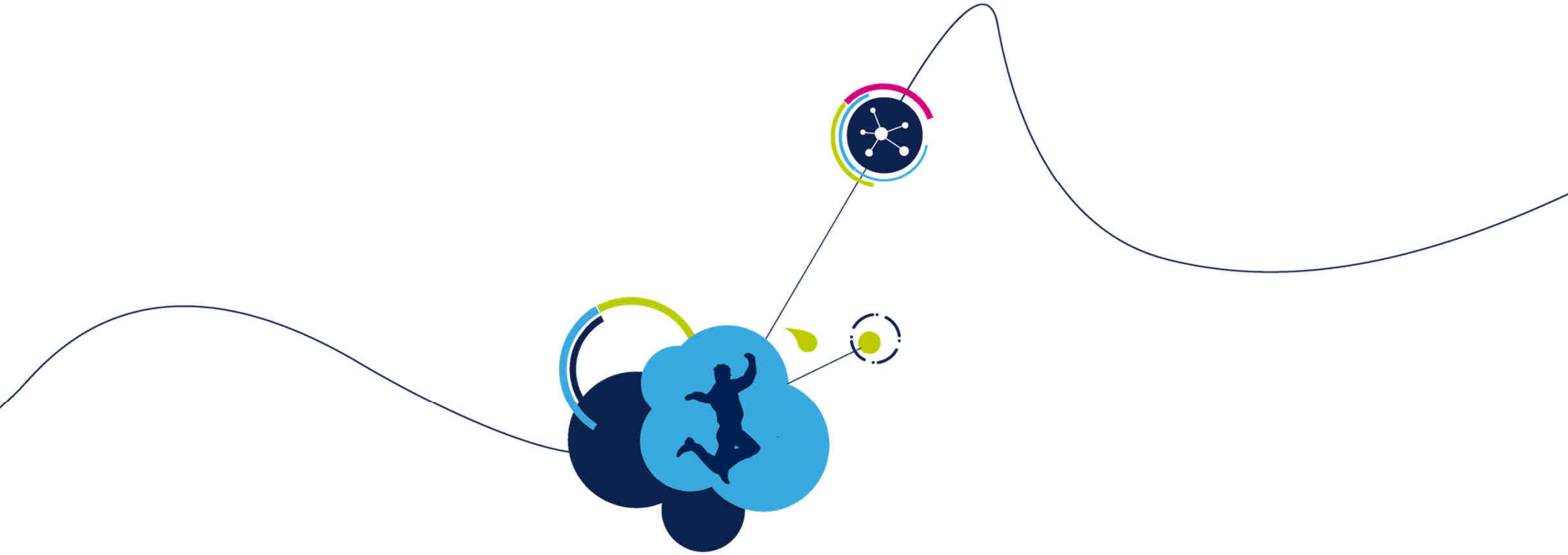
Battery lifetime: CR2032

State	Avg Power Cons	Battery lifetime
stand-by	25uA	8800 hrs / 367 days
advertise 250ms	170uA	1294 hrs / 54 days
Battery notification	0.25mA	880 hrs / 37 days
Environmental / AccEvents / LED control	0.3mA	733 hrs / 30 days
Motion FX (Inertial)	1.4mA	157 hrs / 6.5 days
BlueVoice	3.6mA	61.1 hrs / 2.5 days
Motion FX plus Time Of Flight	8mA	27.5 hrs



The end: Q&A

life.augmented

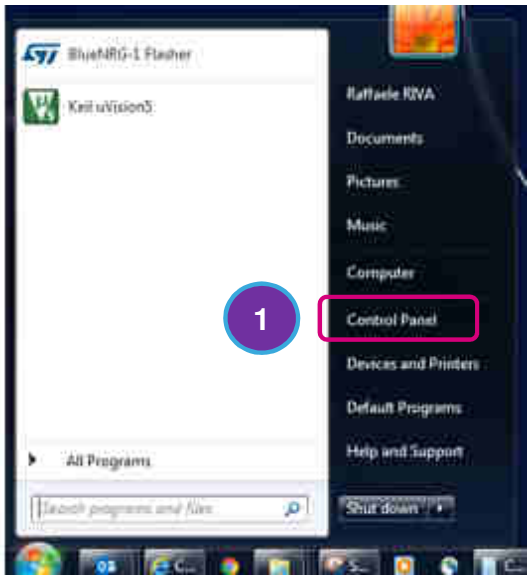


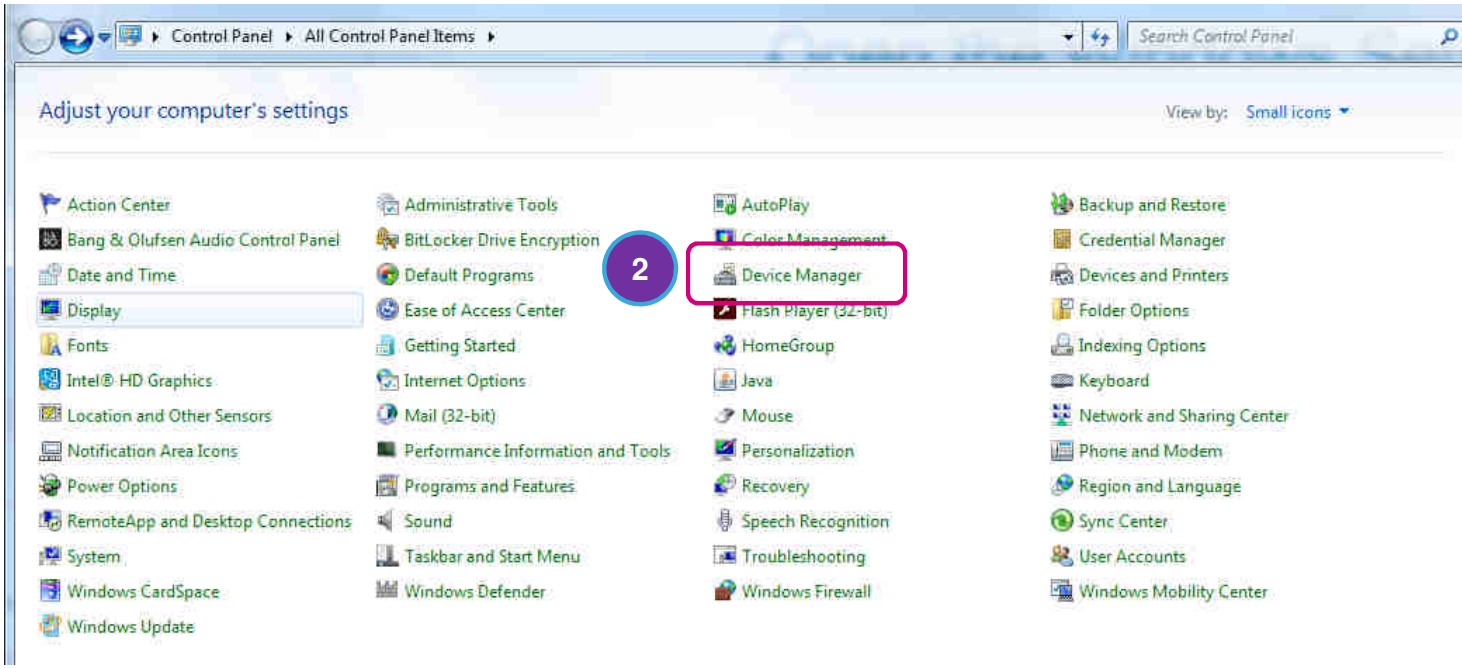
Virtual COM driver installation Win7

life.augmented

Open the Windows Control Panel

1. From **Start Menu** select **Control Panel**
2. Select **Device Manager**

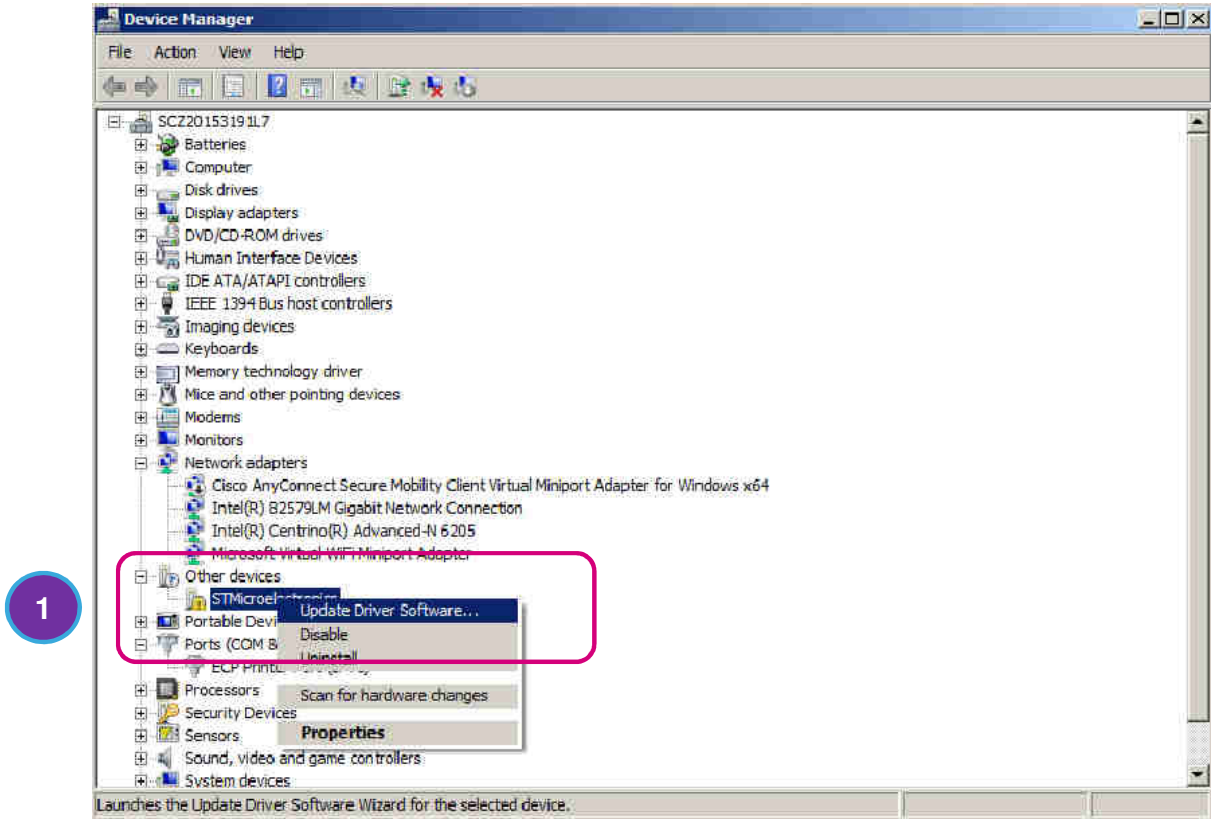




life.augmented

Open the Device Manager

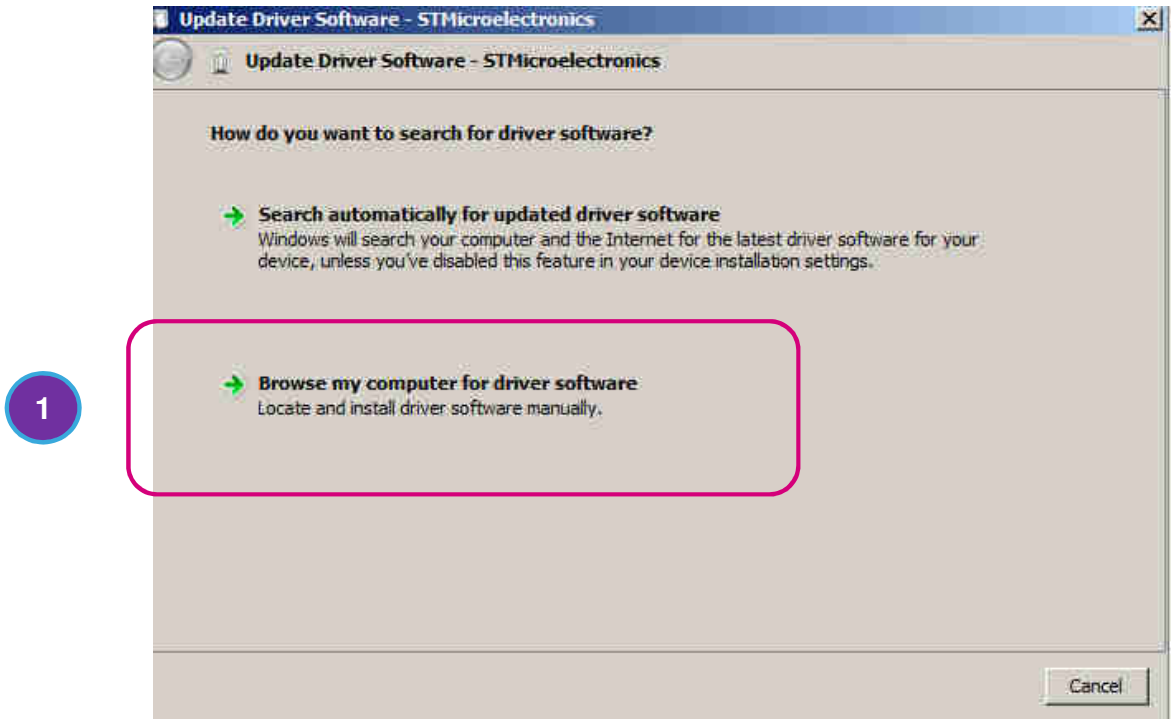
1. Look for **Other devices** and right click and then select **Update Driver Software…**



life.augmented

Look for the VCOM Driver

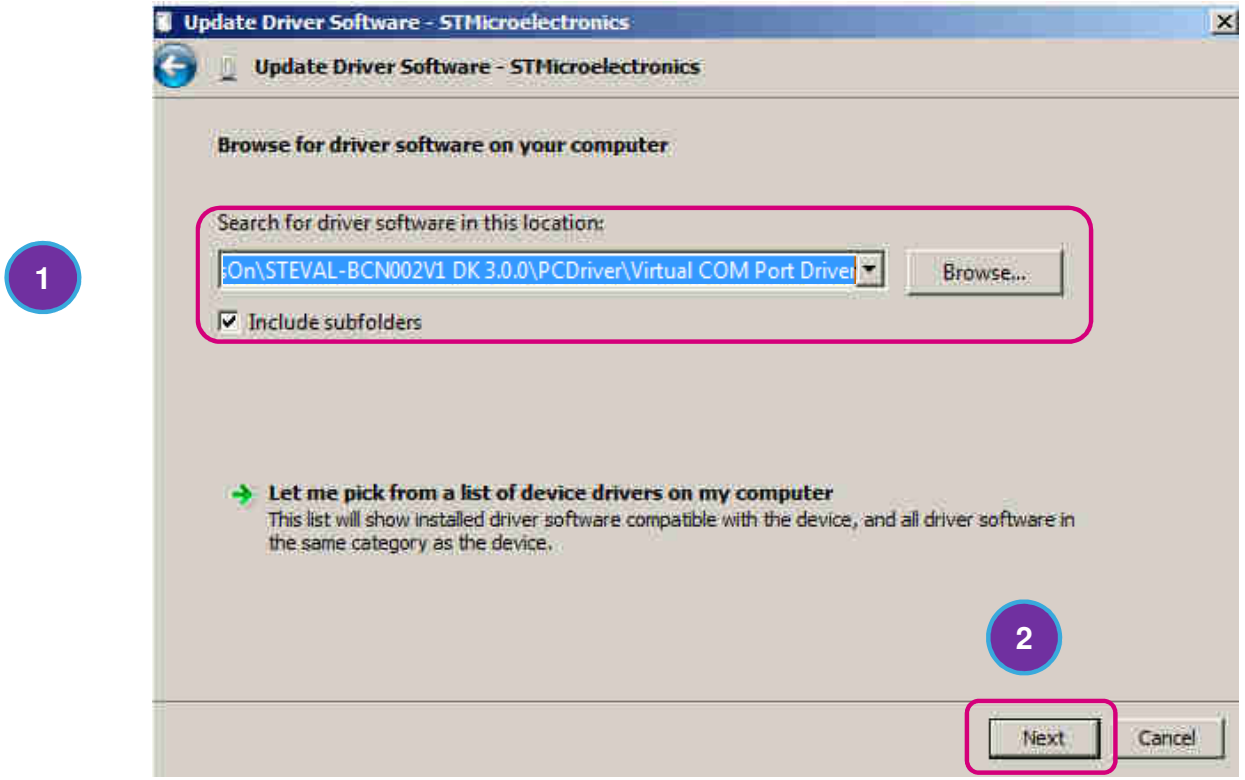
1. Select **Browse my computer for driver software**



Look for the VCOM Driver

1. Click on **Browse** button and **go to** the folder **"\STEVAL-BCN002V1 DK 1.0.9\PCDriver\Virtual COM Port Driver"**

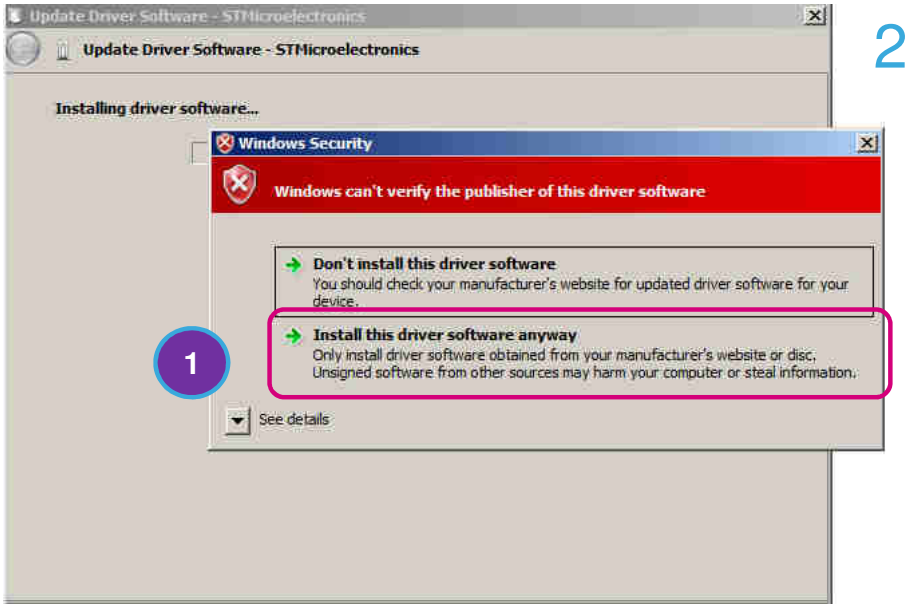
2. Click on **Include subfolder** and then on **Next**



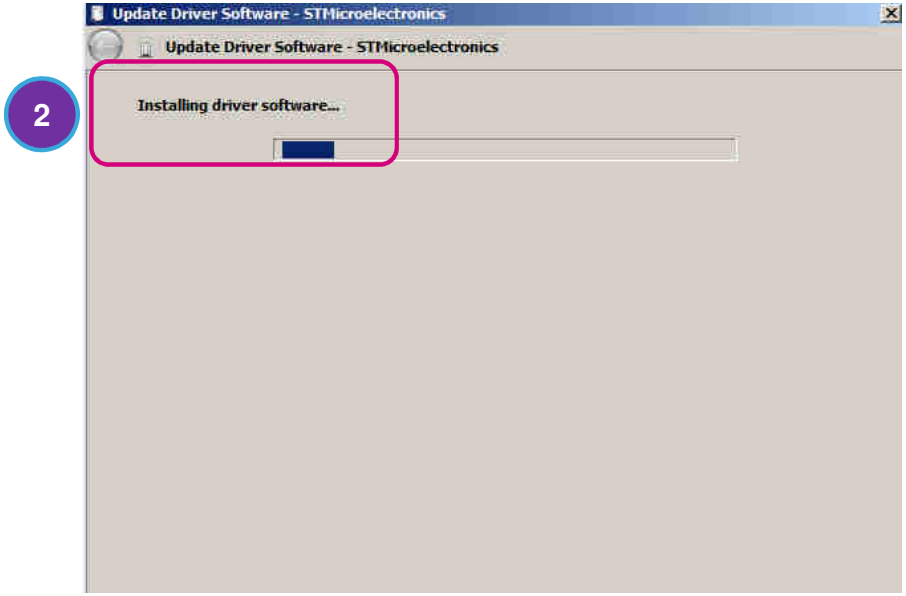
life.augmented

Allow the driver installation

1. Click on **Install driver software anyway**



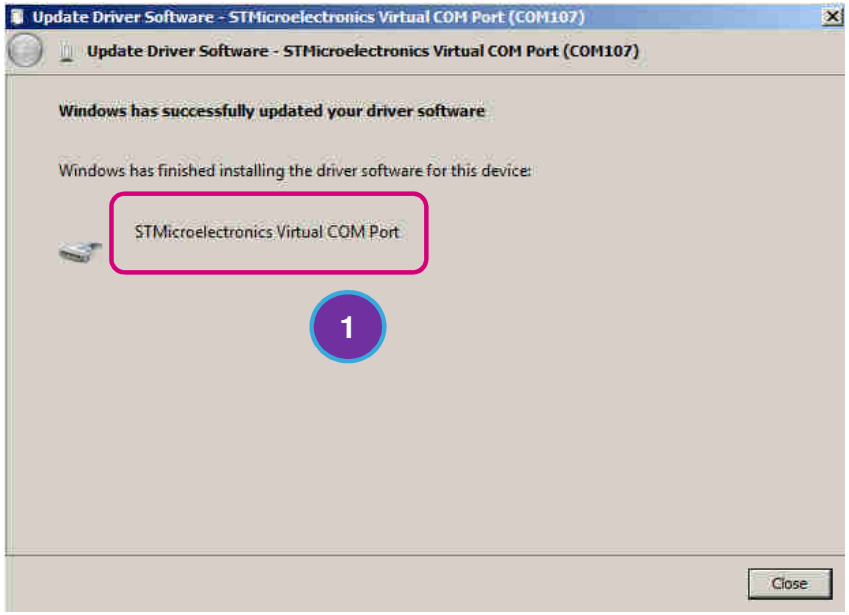
2. Installation starts



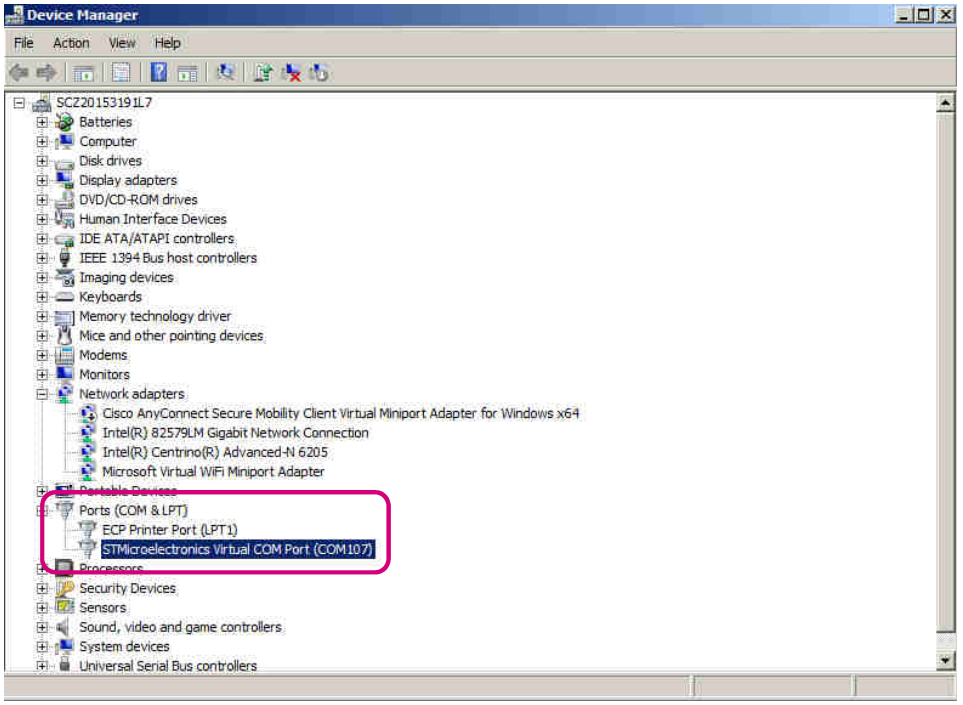
life.augmented

1. Installation completed

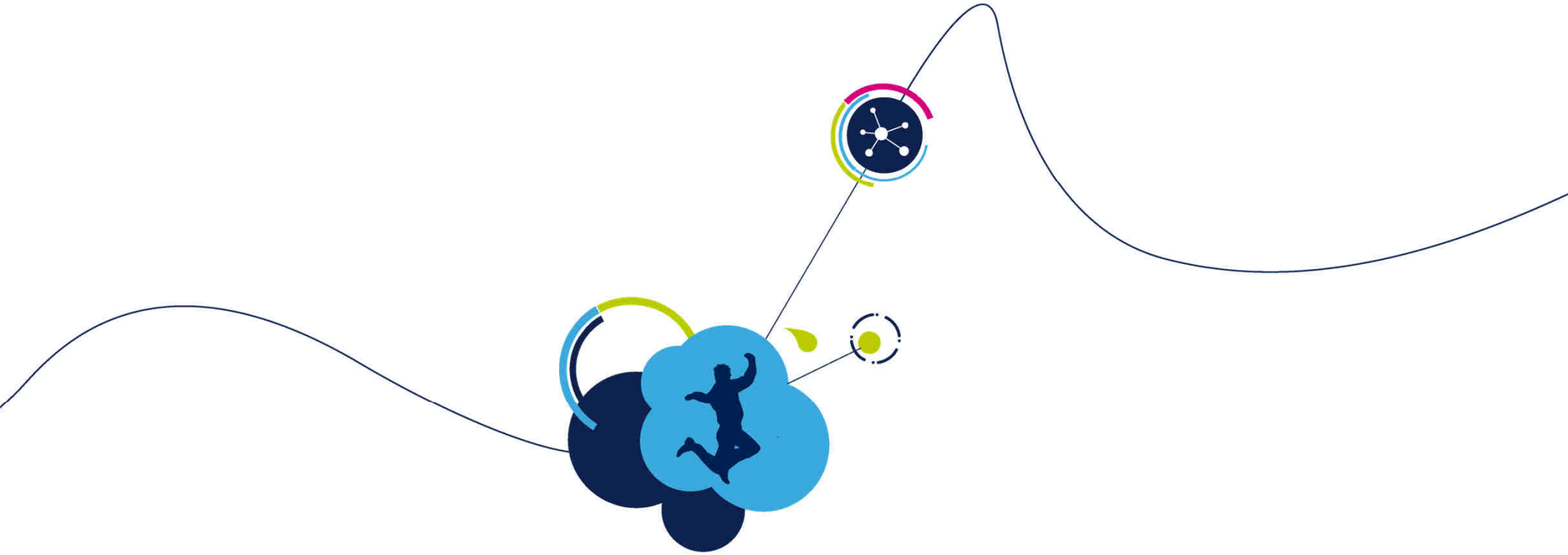
2. Device is in the **COM Ports** list



Done! Now please go back from here!



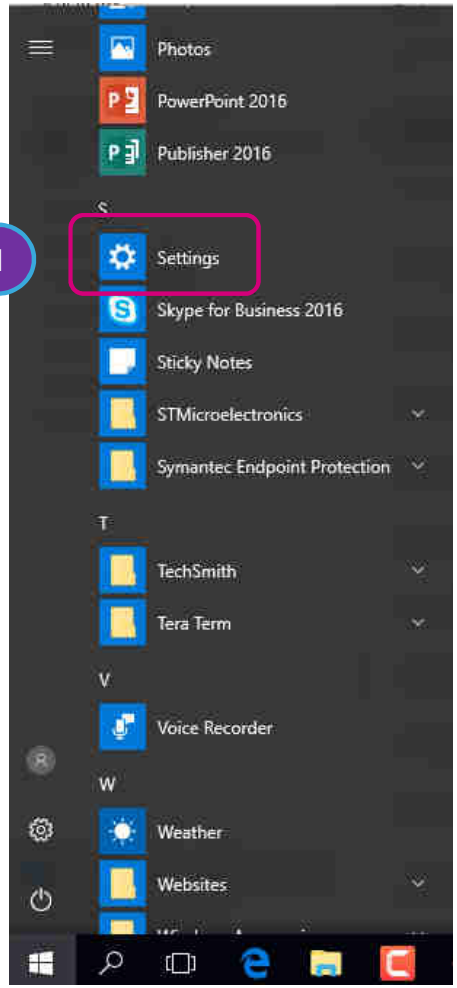
life.augmented



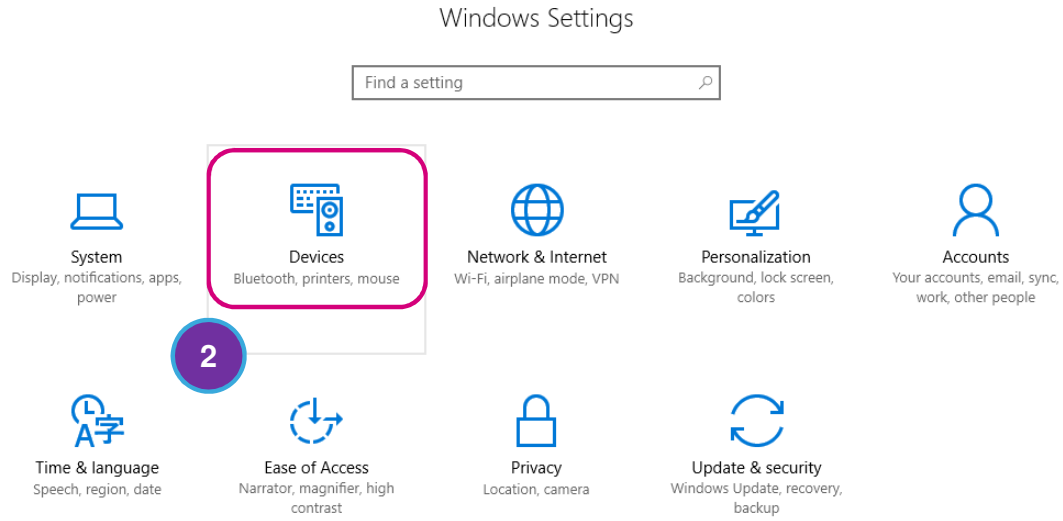
Virtual COM driver installation Win10

life.augmented

Open the Windows Settings



1. From **Start Menu** select **Settings**
2. Select **Devices**

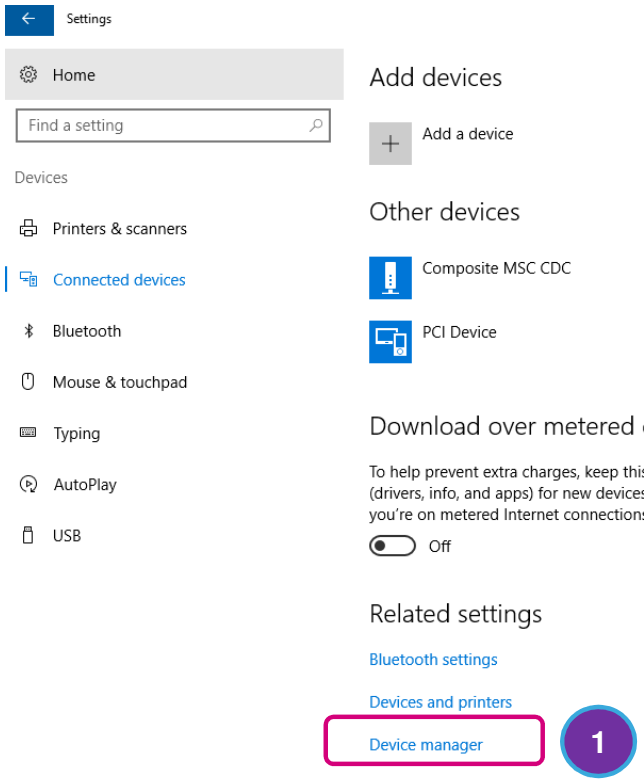


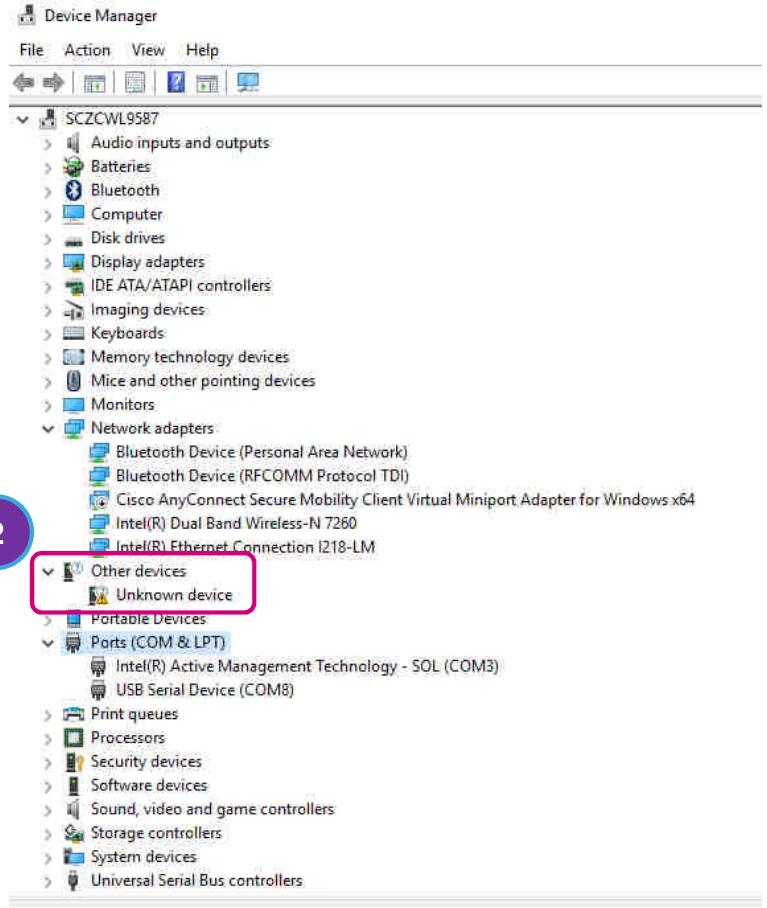
life.augmented

Open the Device Manager

1. Select **Device manager**

2. Select **Other Devices** and **Unknown device**



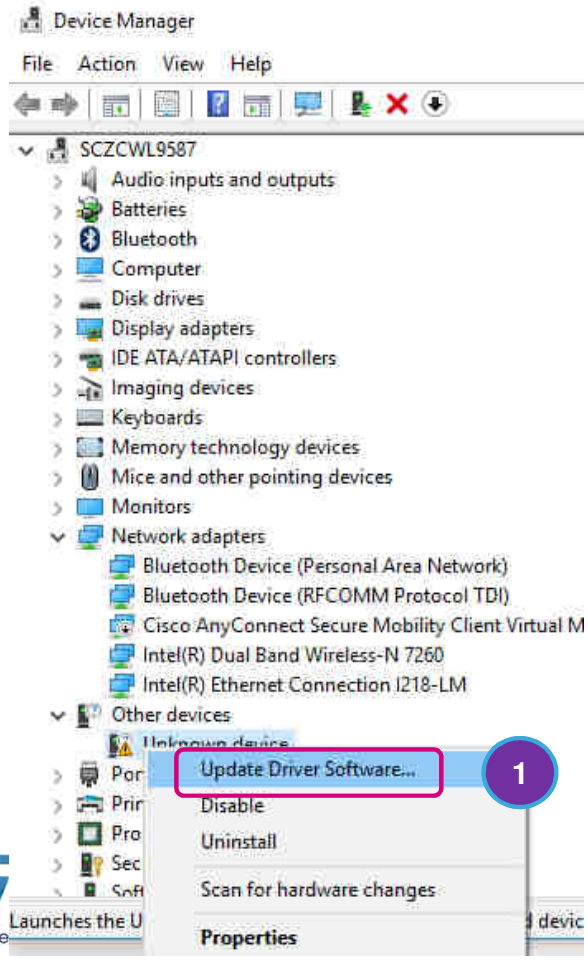


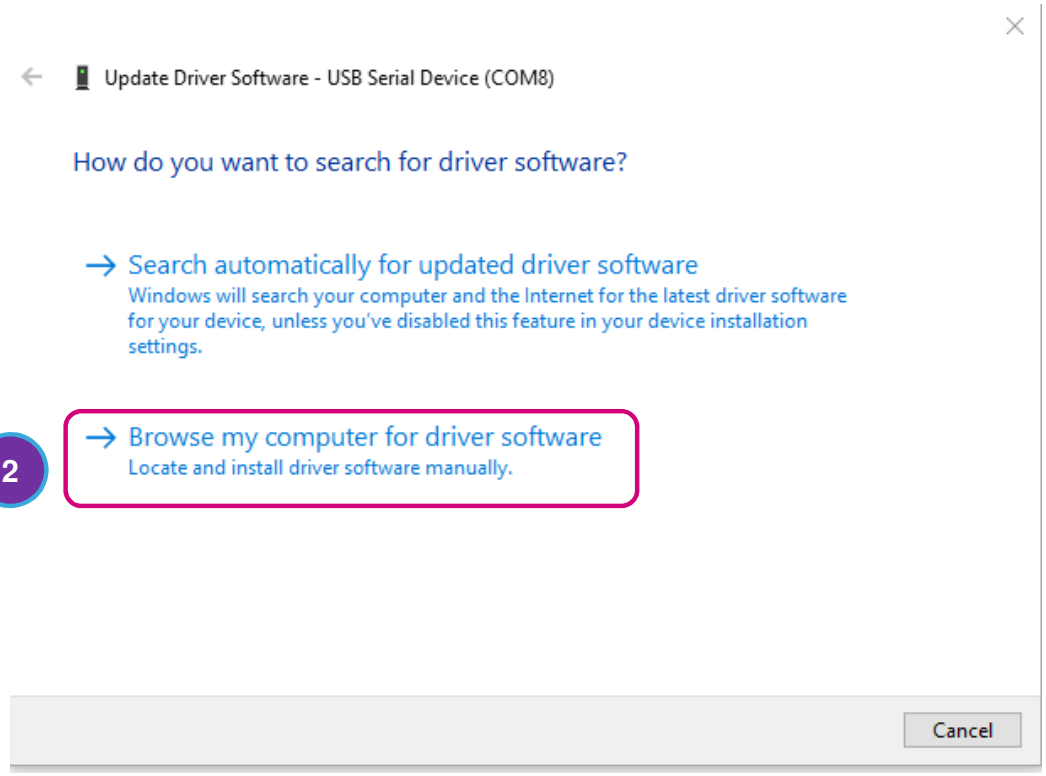
life.augmented

Update Driver Software

1. Right click and then select **Update Driver Software…**

2. Select **Browse my computer for driver software**



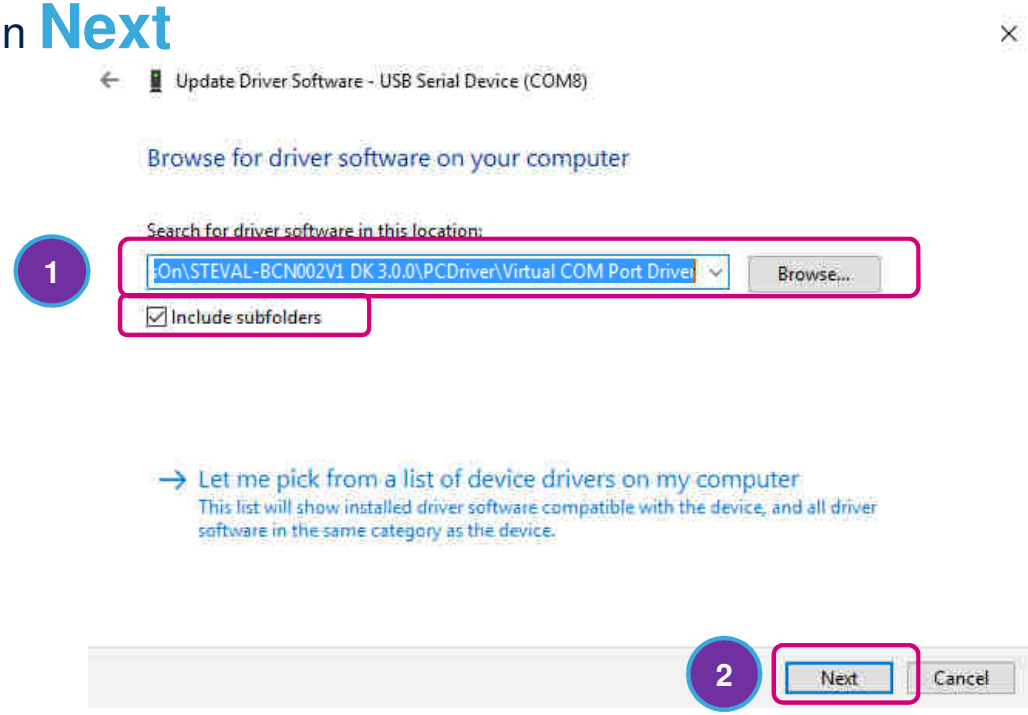


Look for the VCOM Driver

1. Click on **Browse** button and **go to** the folder

   "\STEVAL-BCN002V1 DK 1.0.9\PCDriver\Virtual COM Port Driver"

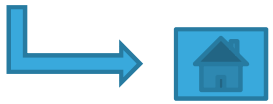
2. Click on **Include subfolder** and then on **Next**

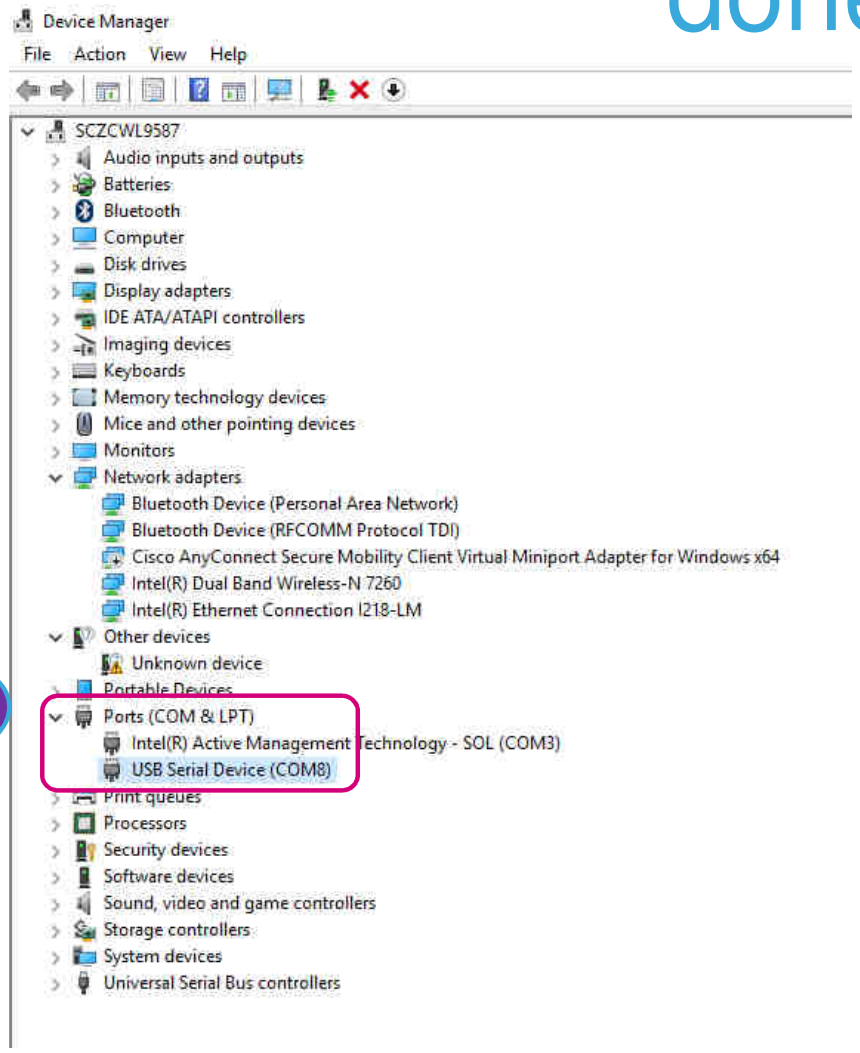


life.augmented

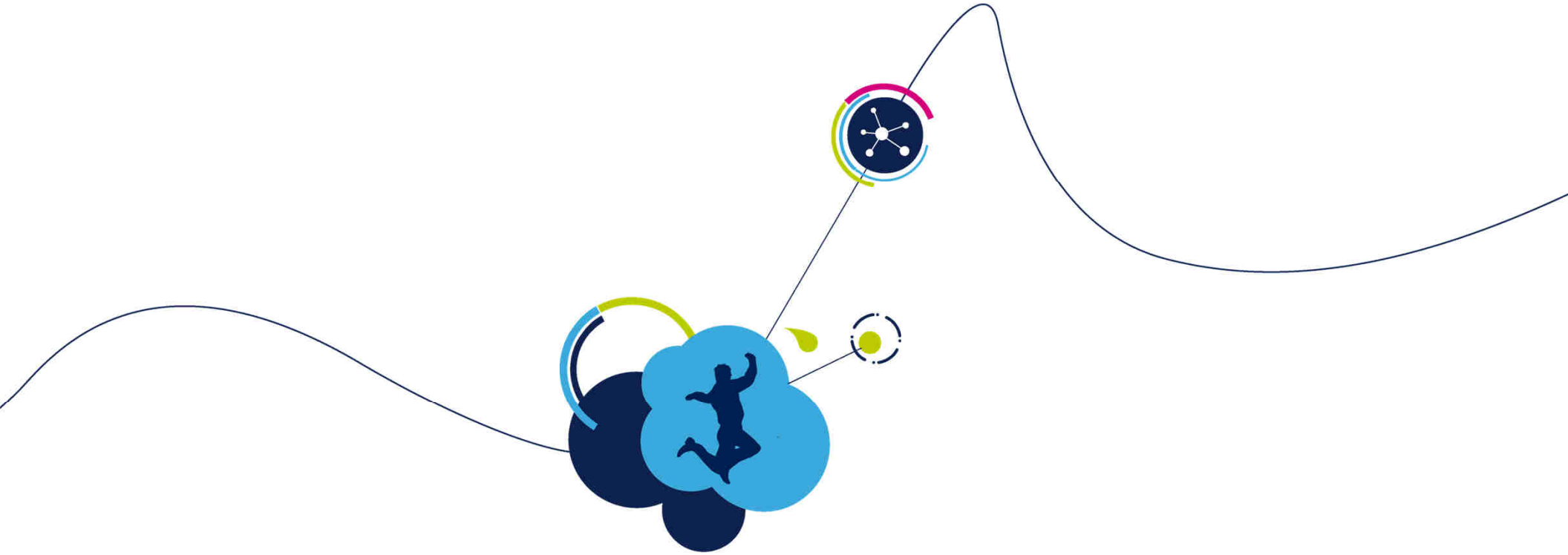
1. Installation completed. Device is in the **COM Ports** list

Done! Now please go back from here!





life.augmented

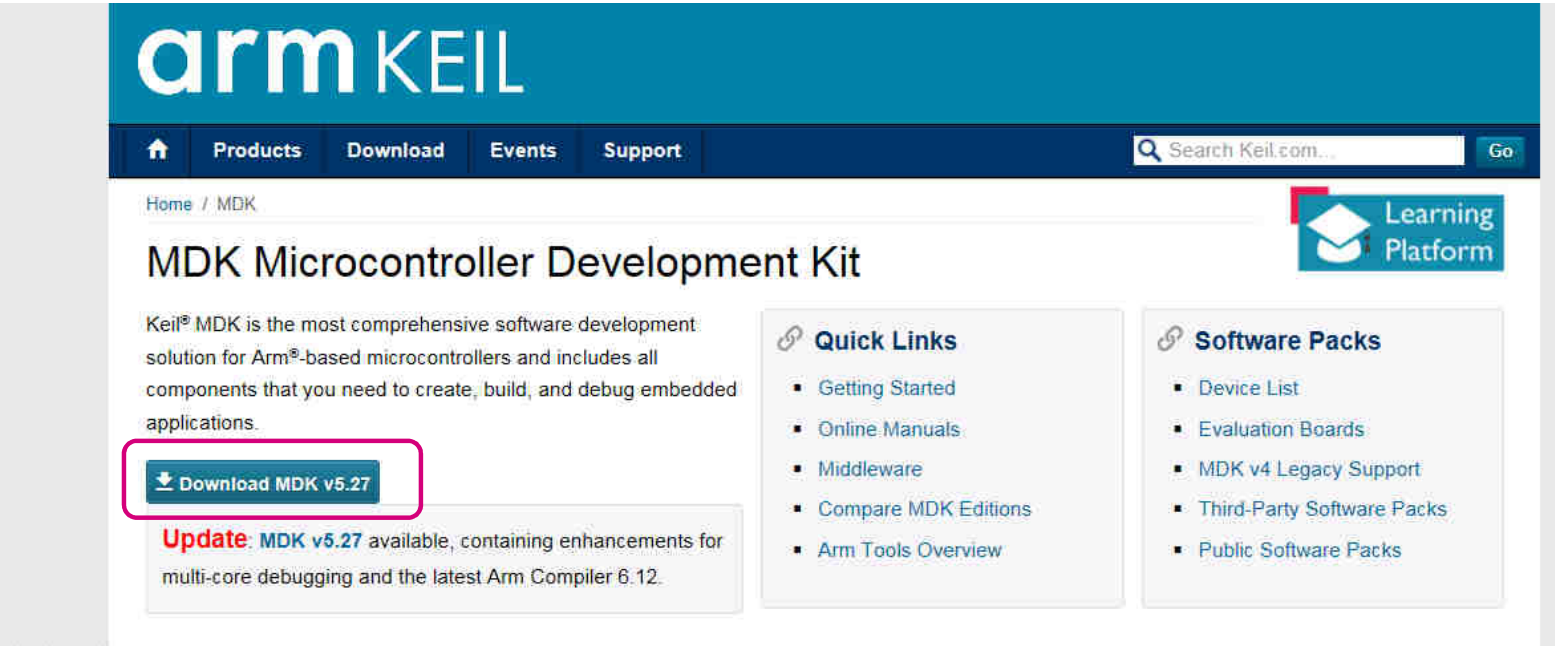


ARM Keil MDK Installation

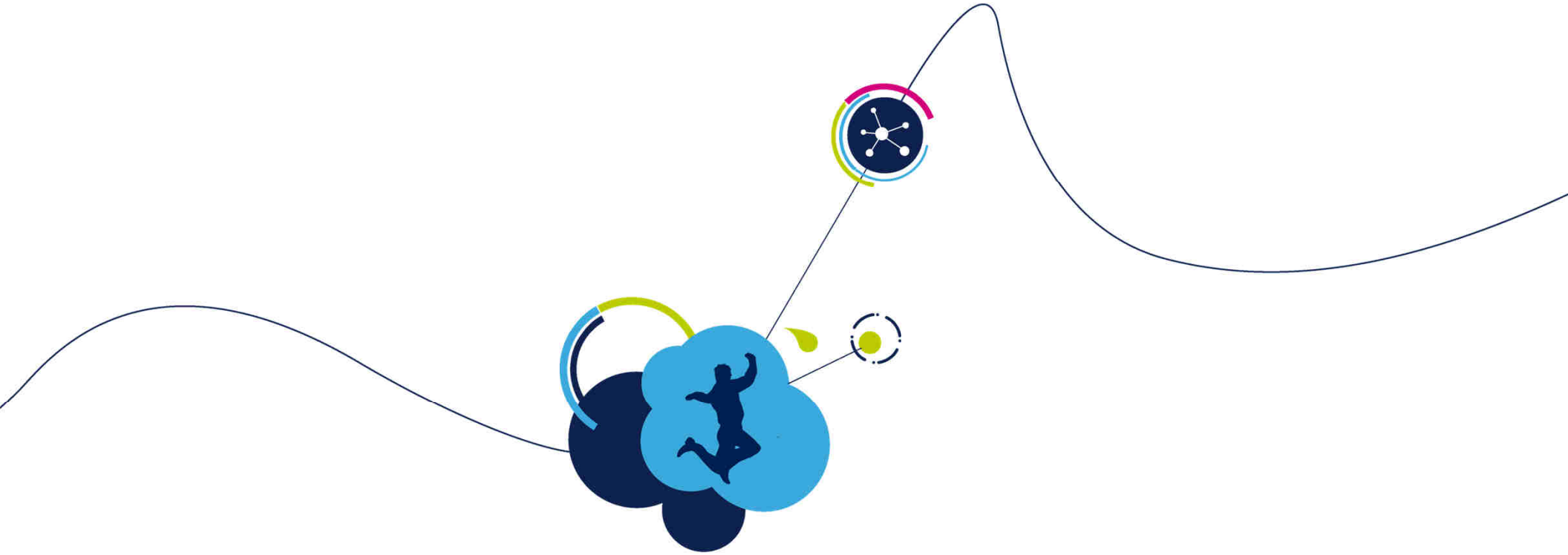
life.augmented

Keil ARM MDK Installation

- Download and install the tool from the following webpage:
  - http://www2.keil.com/mdk5/



life.augmented

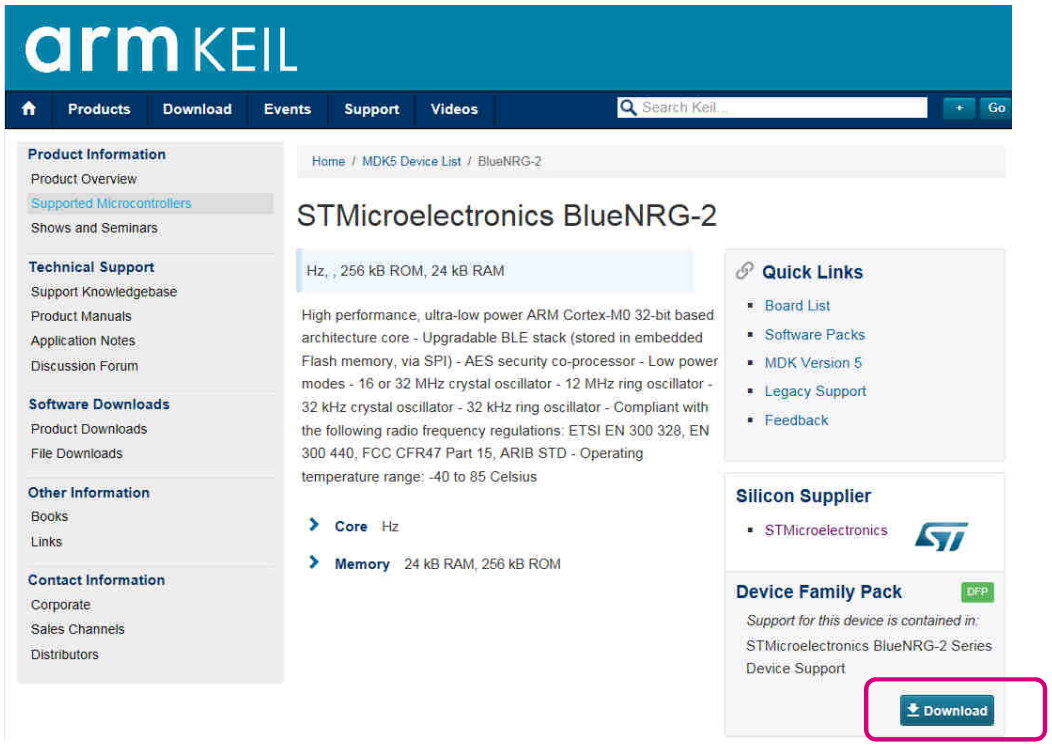


Arm Keil MDK Configuration

life.augmented

Keil ARM MDK Configuration

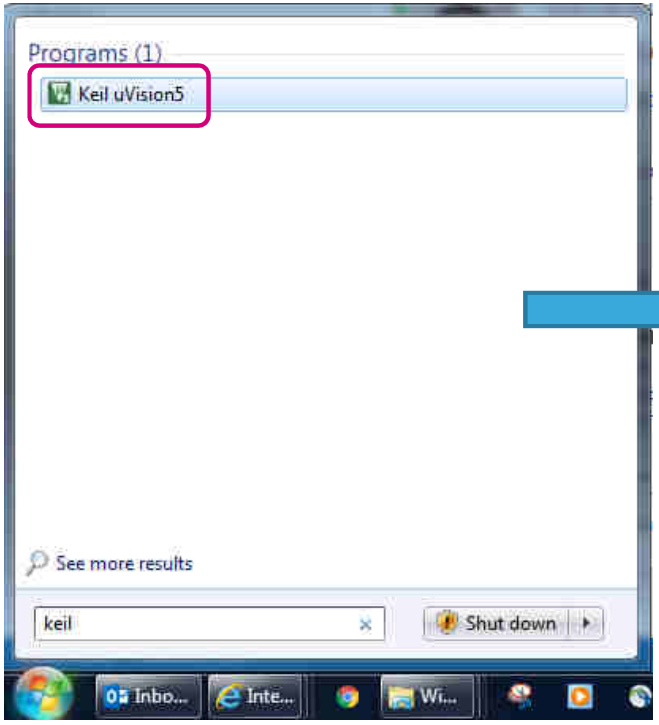
Download and save on your PC the .zip file "**Keil.STBlueNRG-2_DFP.1.0.1.zip**" from the following webpage: https://www.keil.com/dd2/stmicroelectronics/bluenrg_2/

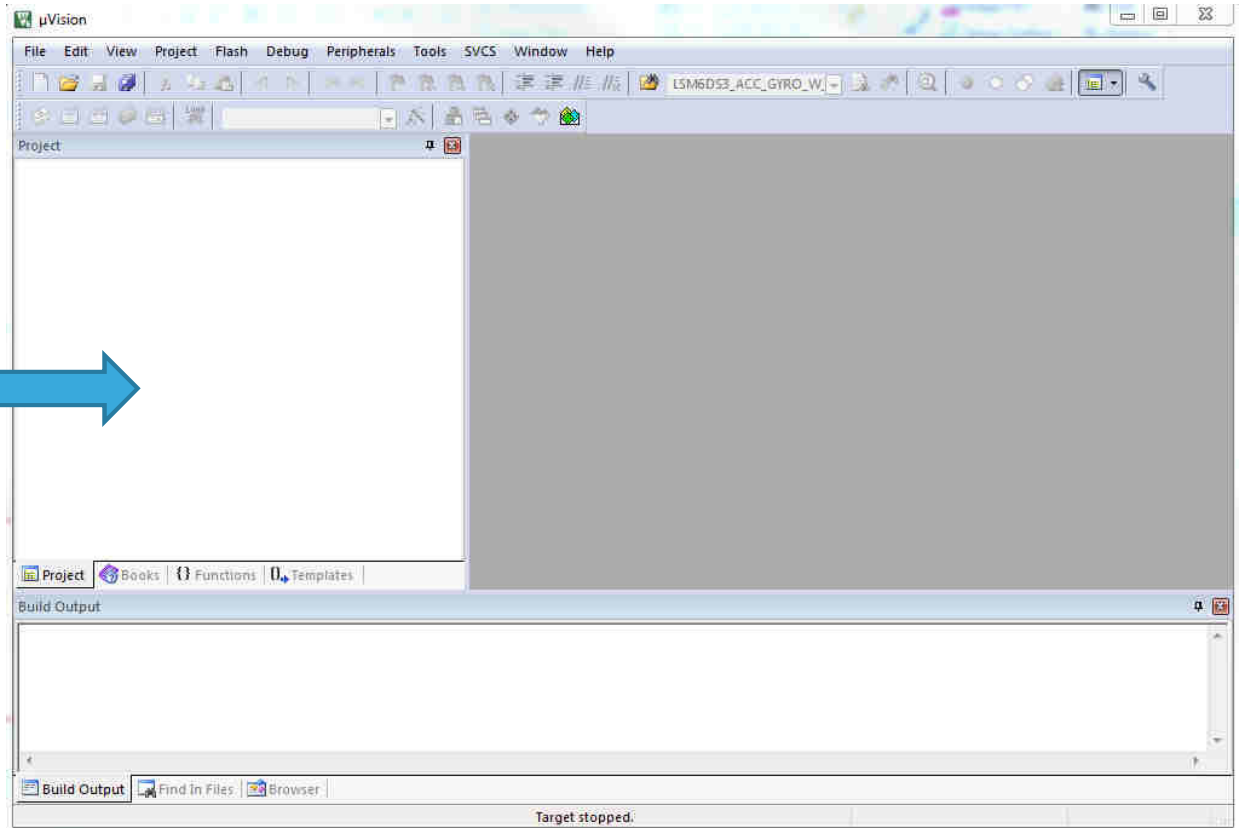


life.augmented

Keil ARM MDK Configuration

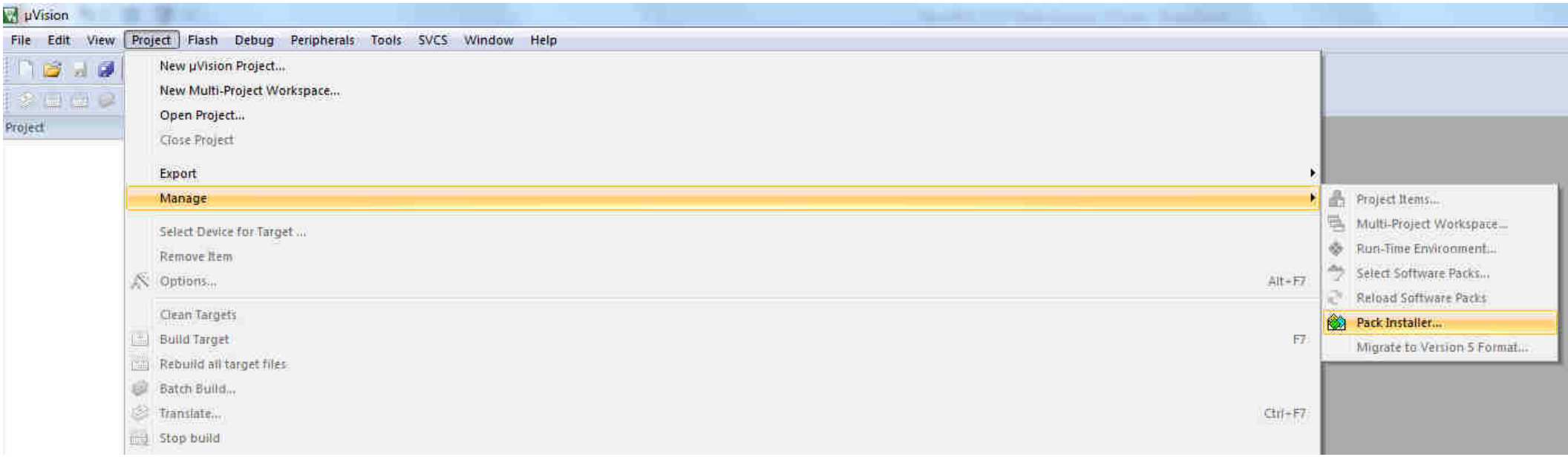
- **Open Keil uVision5 IDE**
- This is the main GUI





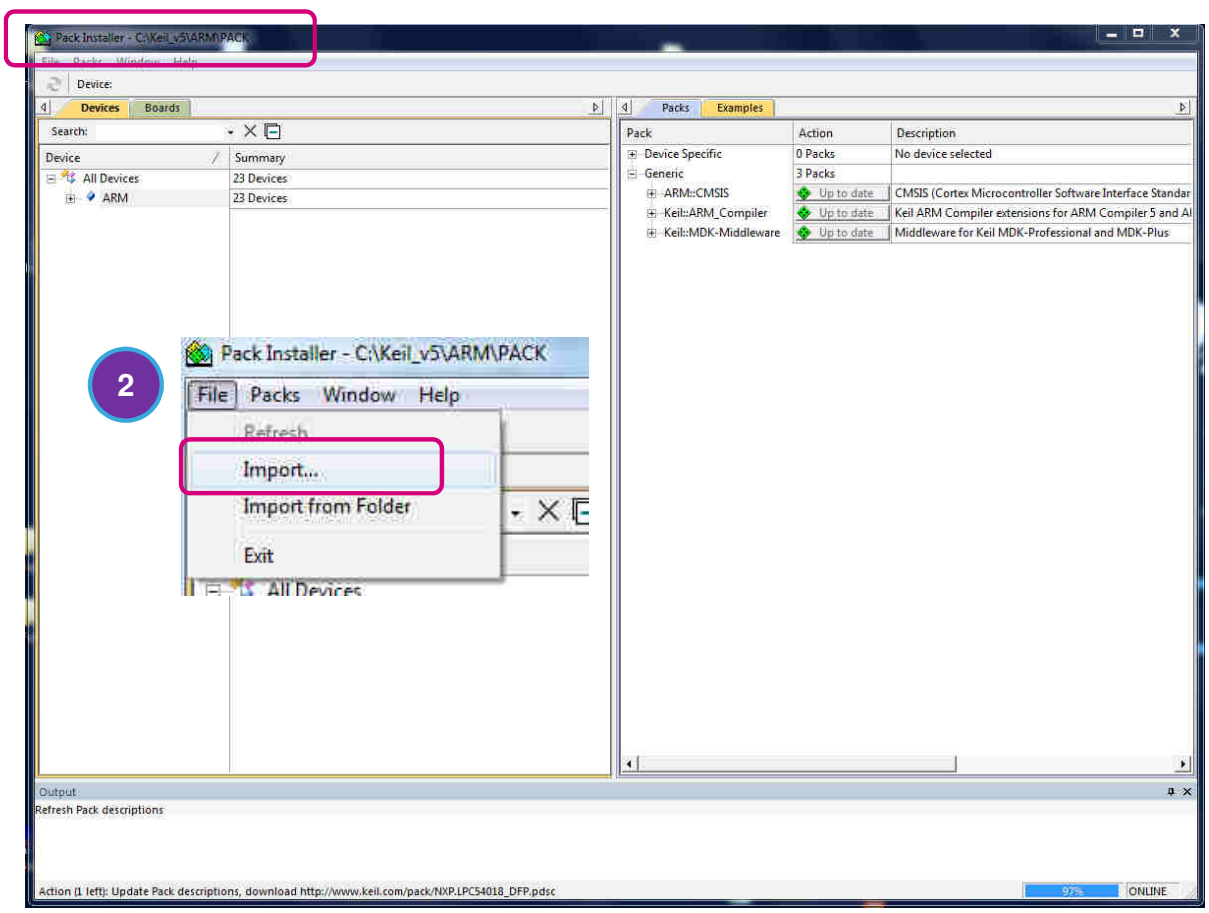
Keil ARM MDK Configuration

- **Open** the **Pack Installer**
  - **Go to Project->Manage->Pack Installer**



Keil ARM MDK Configuration

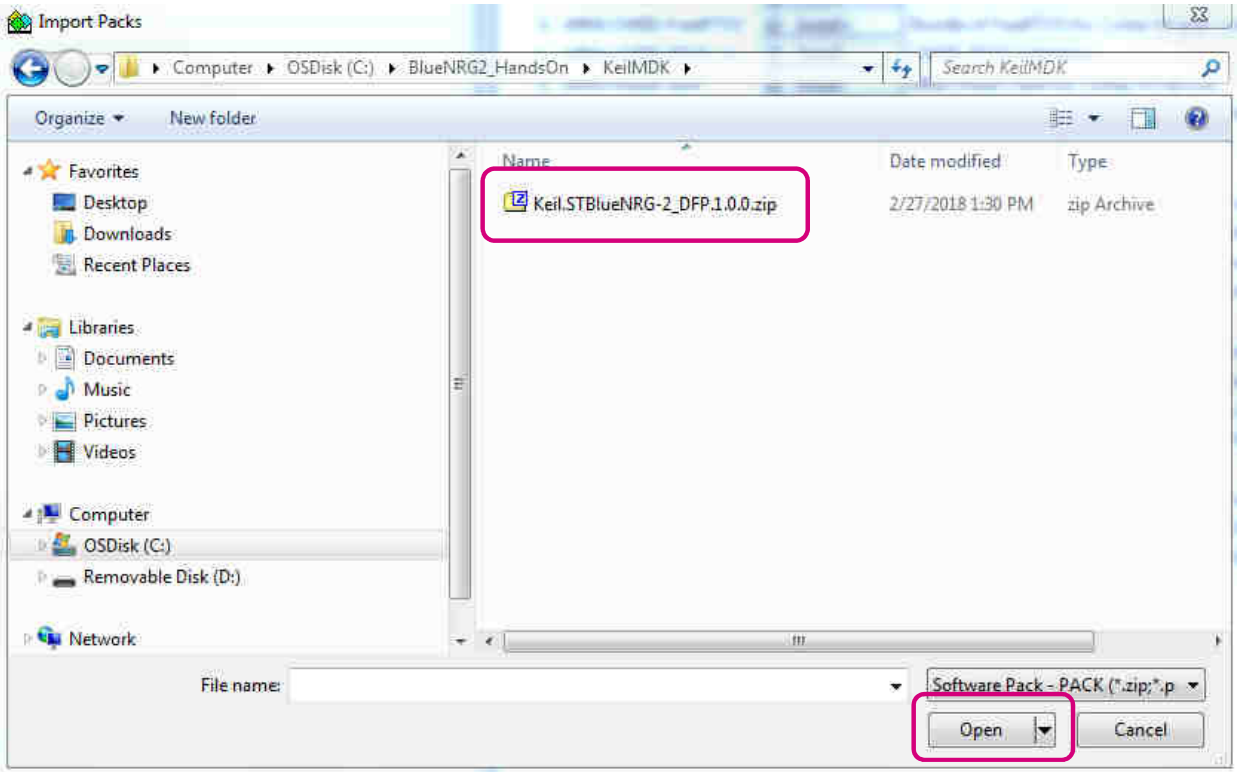
1. In the **Pack Installer…**
2. **Go to File->Import**



life.augmented

Keil ARM MDK Configuration

1. **Open** the previously saved .zip file "**Keil.STBlueNRG-2_DFP.1.0.1.zip**"



life.augmented

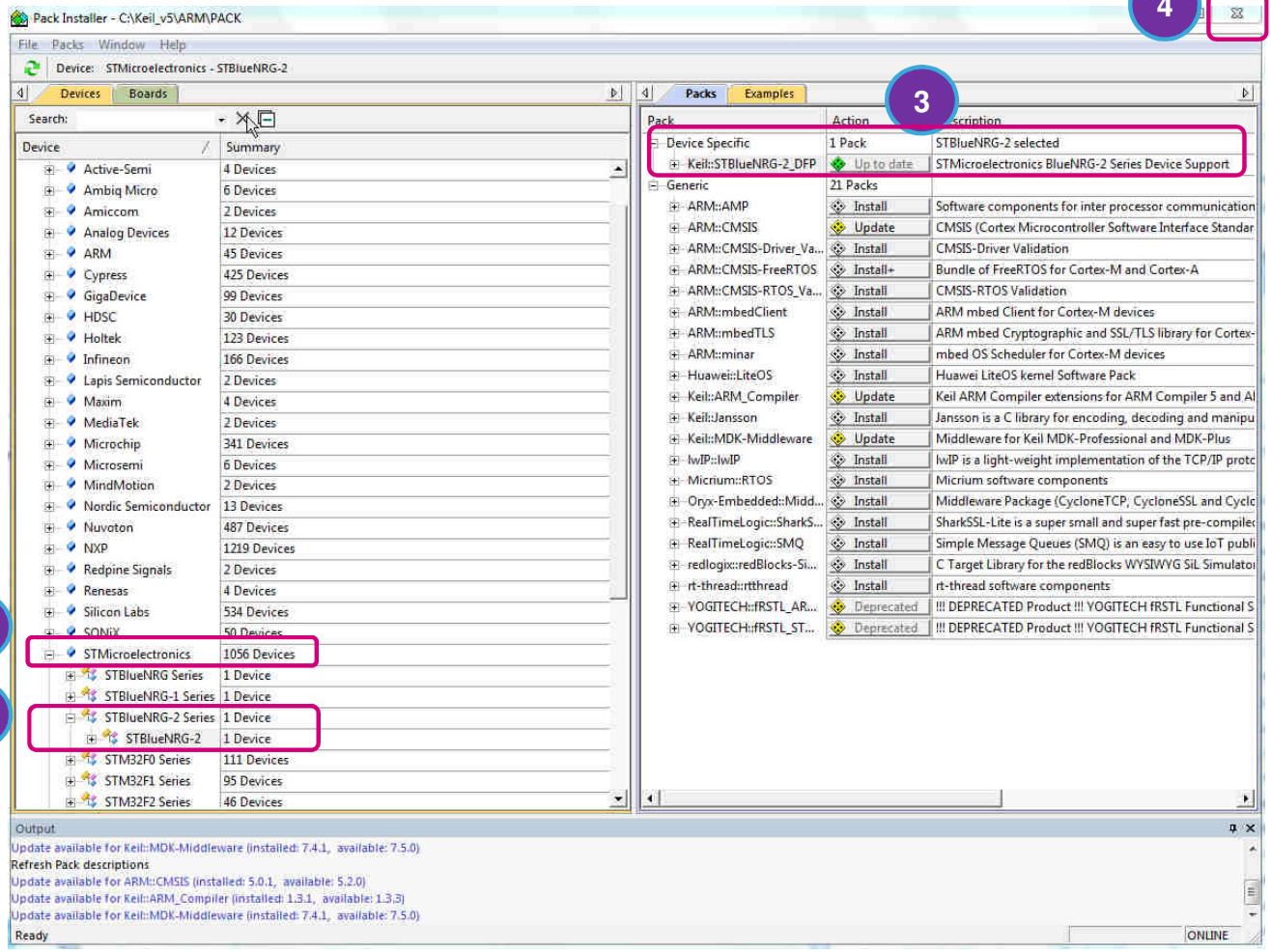
Keil ARM MDK Configuration

1. Once the Device Support pack has been installed, on the **Devices** list in the **left column** select the vendor **STMicroelectronics**

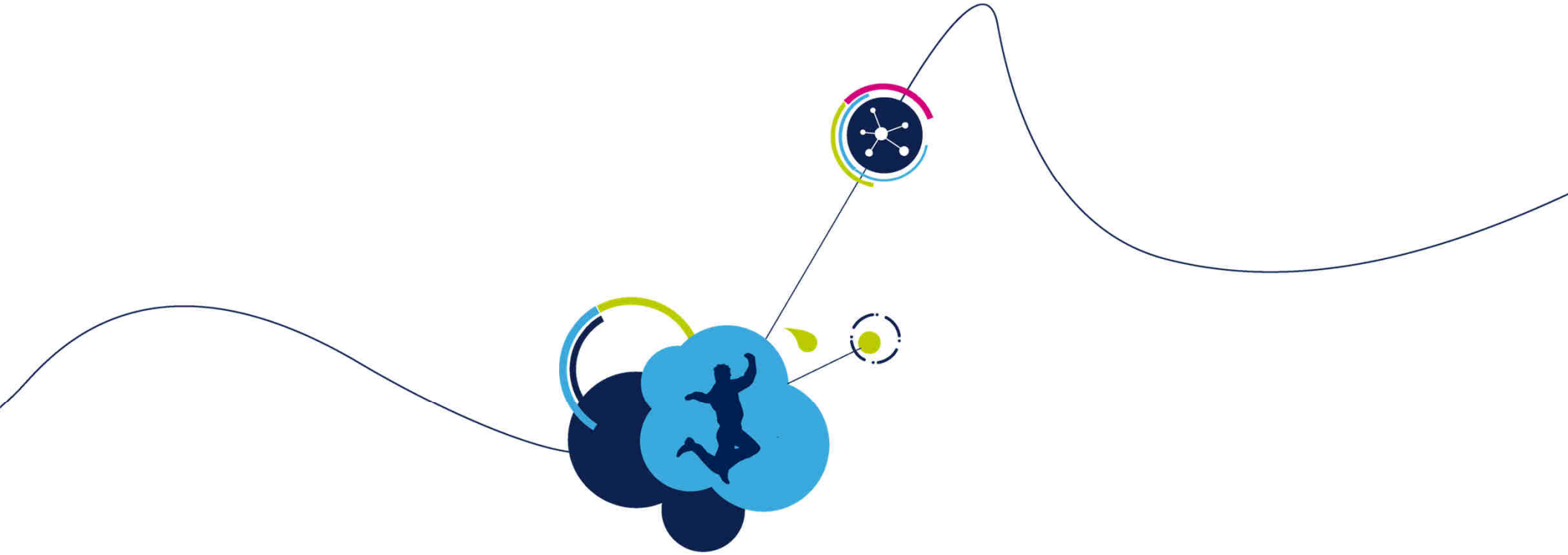
2. **Go to STBlueNRG-2 Series** and **select STBlueNRG-2**

3. Now it will show in green **"Up to date"**

4. **Close** the **Pack Installer**



life.augmented



BlueNRG-2 free license for Keil ARM MDK

life.augmented

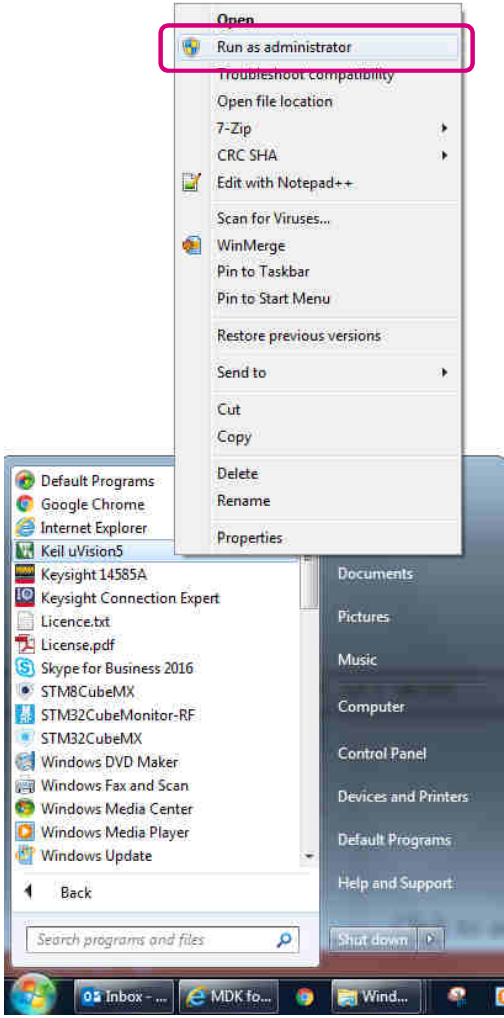
BlueNRG-2 MDK license Activation

- You can refer to the following webpage:
  - http://www2.keil.com/stmicroelectronics-stm32/mdk

- Note: it is mentioned STM32L0 and STM32F0, but the same procedure applies to BlueNRG-2 device

BlueNRG-2 License Installation 1/6

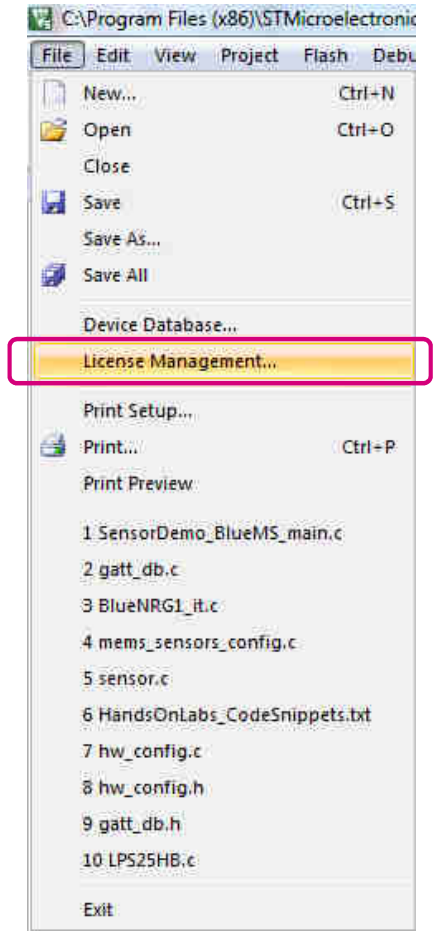
- Login with an account that has **administration rights**.
- Right-click the µVision icon and select **Run as Administrator...** from the context menu.



life.augmented

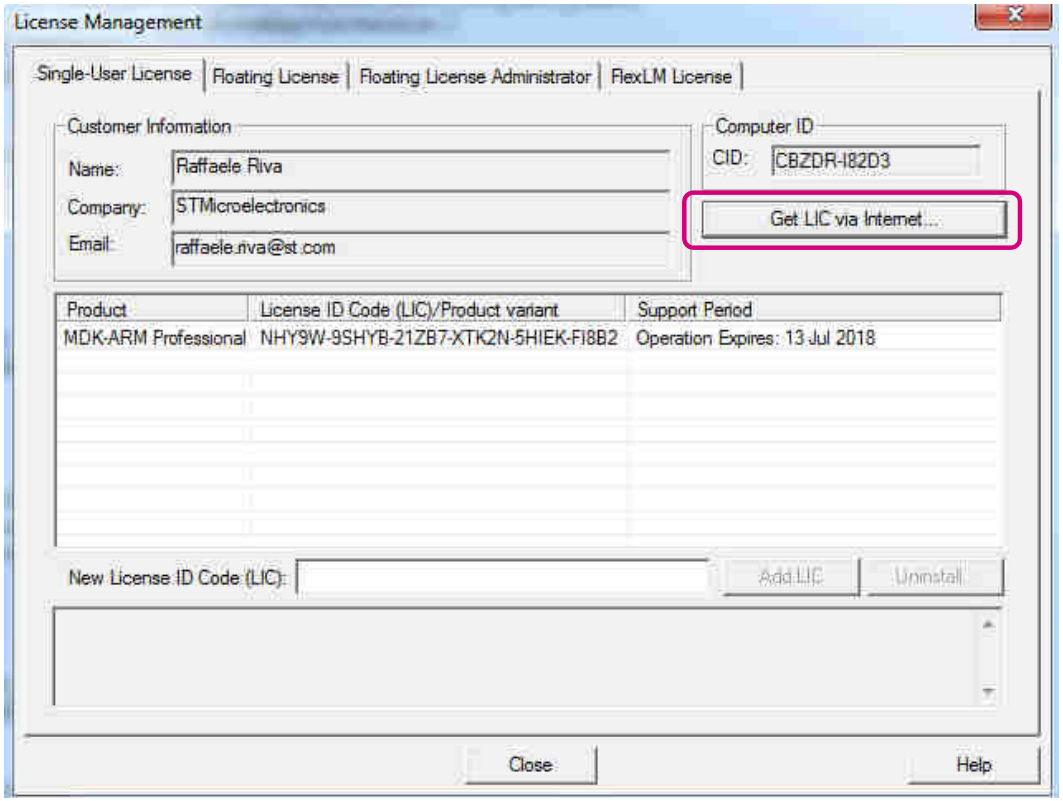
BlueNRG-2 License Installation 2/6

1. Open the dialog **File — License Management...**



BlueNRG-2 License Installation 3/6

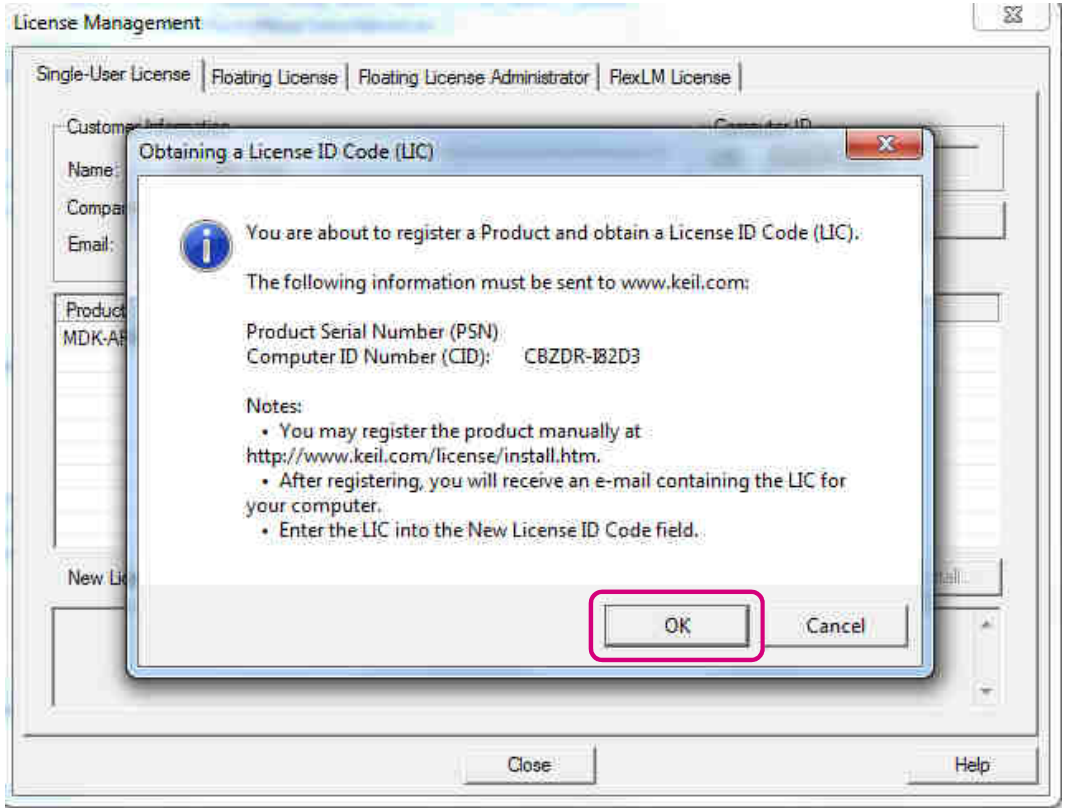
Select the **Single-User License** tab and click the button **Get LIC via Internet...**,



BlueNRG-2 License Installation 4/6

Click the button **OK** to register the product.

- This action opens the License Management page on the Keil web site.

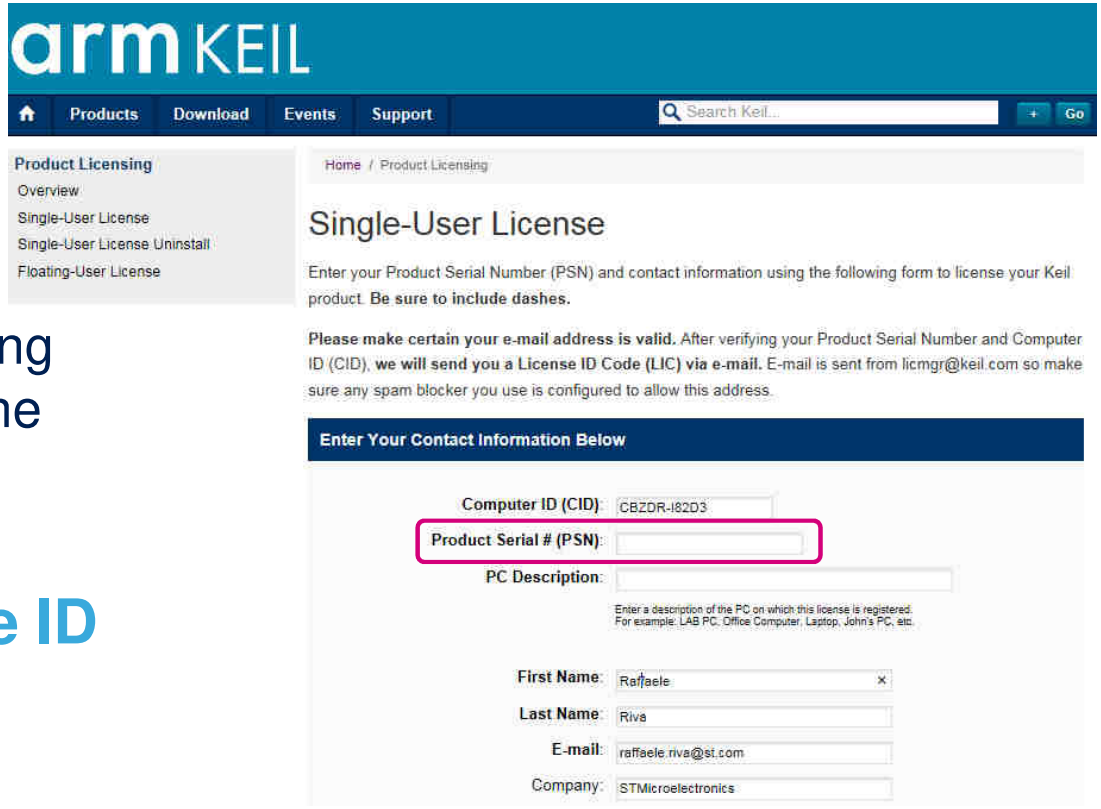


life.augmented

BlueNRG-2 License Installation 5/6

For the latest **Product Serial Number** please check on the Keil website at this URL:

http://www2.keil.com/stmicroelectronics-stm32/mdk



Then enter the PSN in this webpage along with your contact information and click the button **Submit**.

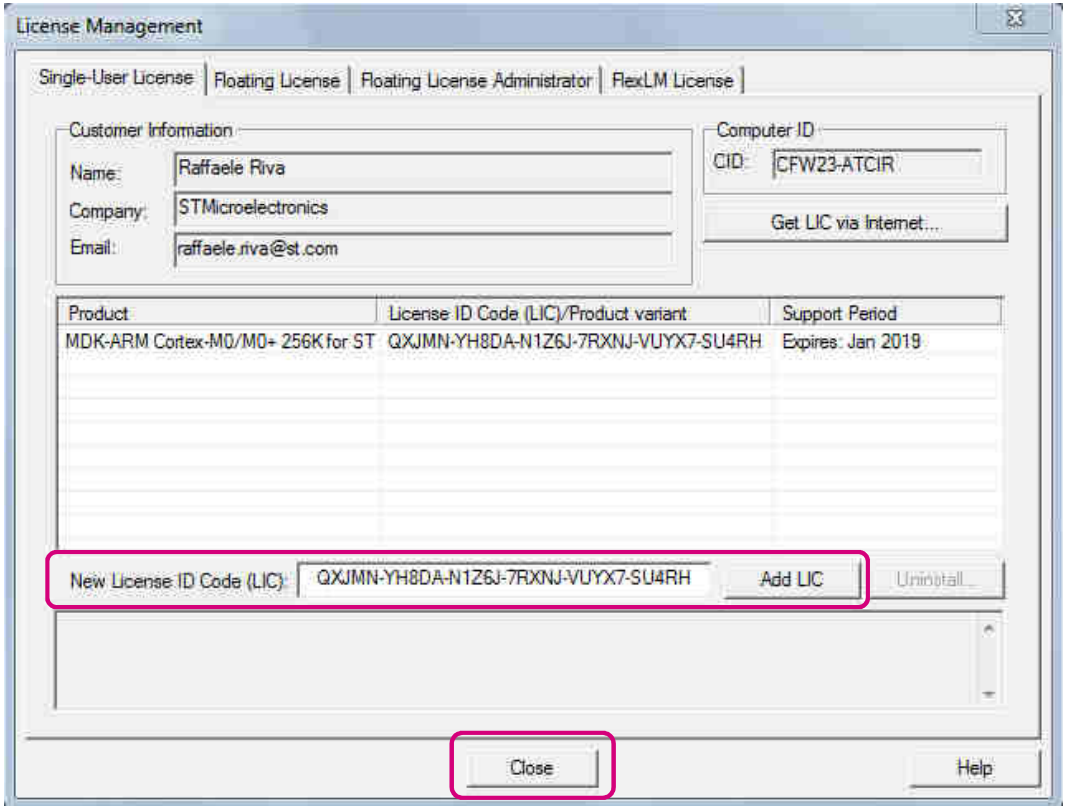
An e-mail is sent back with the **License ID Code (LIC)** within a few minutes.

life.augmented

BlueNRG-2 License Installation 6/6

To activate the Software Product, enter the **LIC** you received by email in the field **New License ID Code (LIC)** and click **Add LIC**.

Then click on **Close**



life.augmented