# Getting started with Zigbee on STM32WB

**Rev0.1**

## Introduction

This document gives a quick overview of Zigbee on STM32WB.

# Table of Contents

# 1. Acronyms and Definitions

| | |
|---|---|
| HAL | Hardware Abstraction Layer |
| ZCL | ZigBee Cluster Library |
| APS | Application Support Sub-Layer |
| BDB | Base device behavior |
| API | Application Programming Interface |
| IPCC | Inter-Processor Communication Controller IP |
| MAC | Media Access Control |
| BDB | Base Device Behavior |
| APS | Application support sub-layer |
| ZDO | Zigbee Device Object |

# 2. Reference documents

[1] AN5289 Building Wireless Application with STM32WB microcontroller series

# 3. Introduction

## 3.1. Zigbee overview

ZigBee is an IEEE 802.15.4-based communication protocols used to create personal area networks. It is used for applications like home automation, medical device data collection, and any other type of applications with low-power and low-bandwidth constraints using wireless connection. The data throughput is 250 kbps in 2.4 GHz band and the typical range is 10-20 meters.

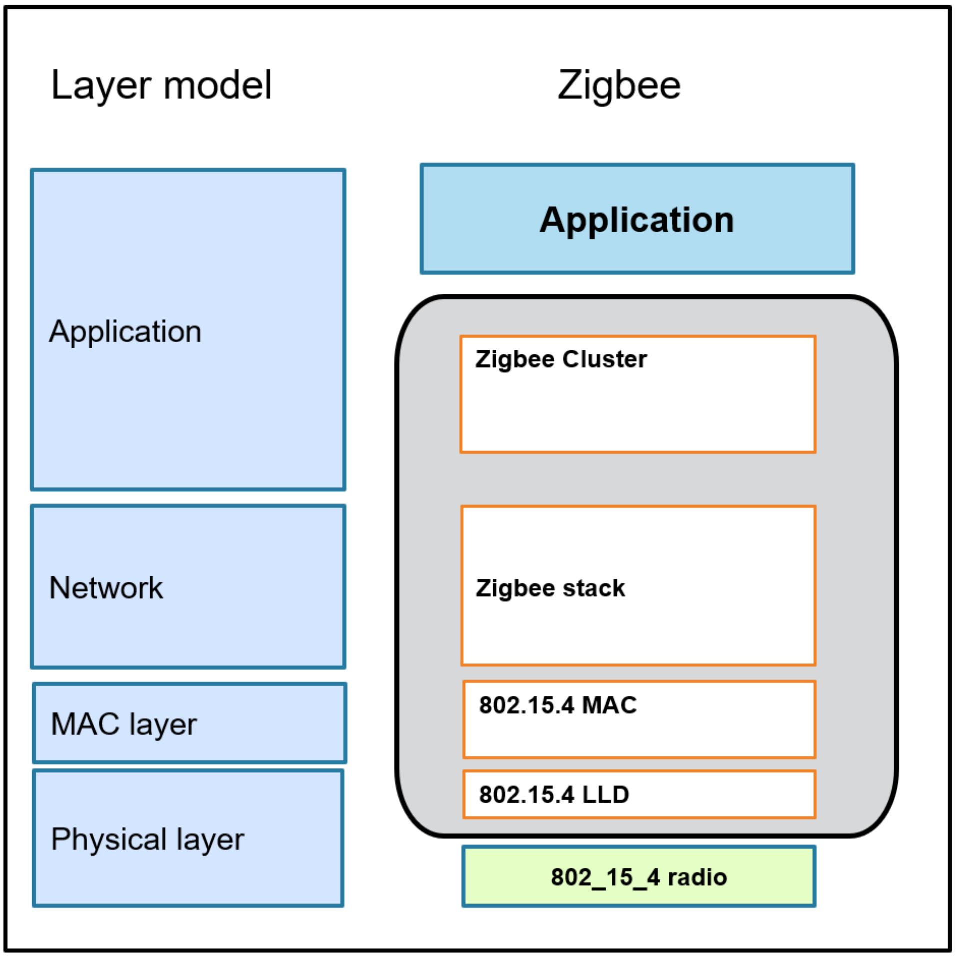


Fig 1 : Zigbee layer model

### 3.1.1. Type of devices

In Zigbee, different types of device are defined:

- **Coordinator** (ZC): This is the first node to be started. The coordinator is responsible for forming the network by allowing other nodes to join the network through it. The coordinator is responsible for starting the network and for choosing certain key network parameters. Once the network is established, the coordinator has a routing role. In a centralized network, every Zigbee mesh network must have one and only one coordinator.

- **Router** (ZR): This is a node with a routing capability which is also able to send and receive data. It also allows other nodes to join the network through it. A Zigbee mesh network can have multiple routers.

- **End Device** (ZED): This is a node which is only capable of sending and receiving data. It has no routing capability. A Zigbee mesh network can have multiple end devices. Some end device can also be sleepy end device allowing very low power consumptions.

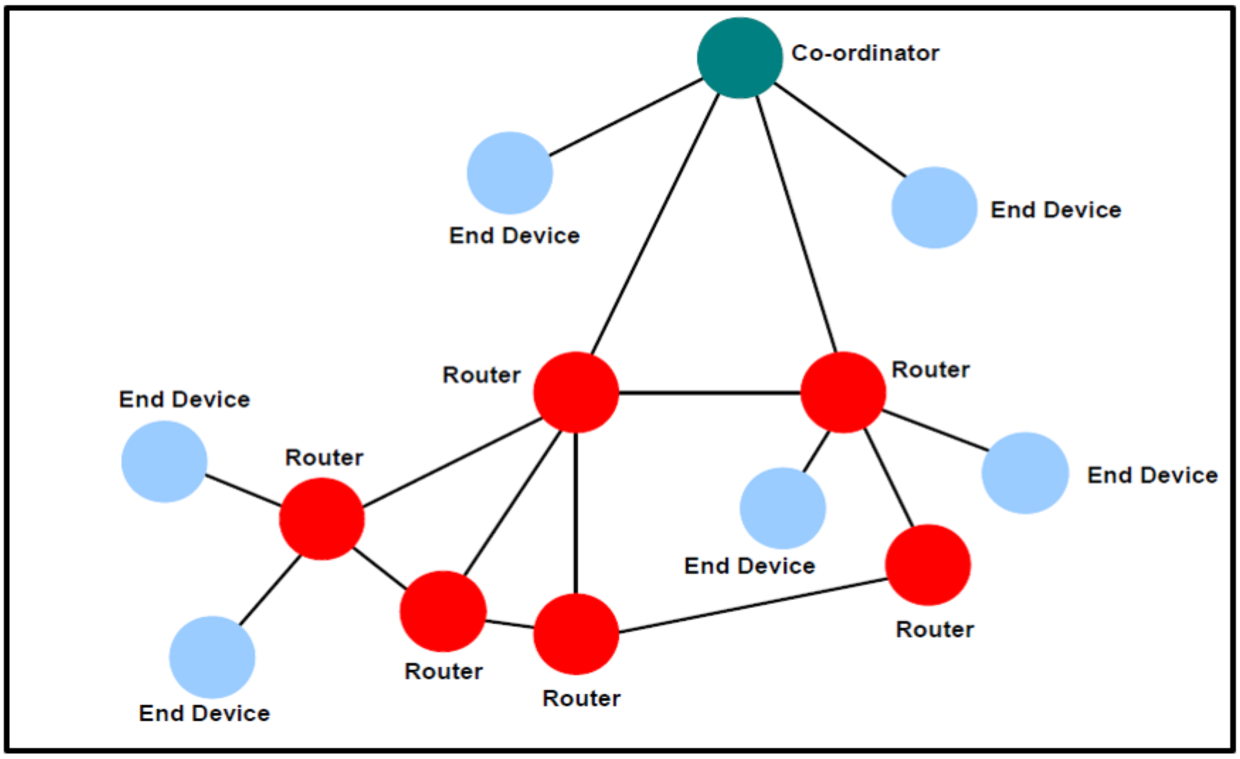


Fig 2 : Zigbee mesh network

### 3.1.2. Type of network

To satisfy a wide range of applications and to ensure the optimal balance of security, Zigbee offers two types of network: **distributed** and **centralized**

- In a distributed network, there are no coordinator. In this configuration, any router can issue network security keys. As more routers and end devices join the network, a router that is already on the network securely sends the network key. All devices on the network use the same network key to encrypt messages.

- In a centralized network, there is an entity named Trust Center (TC), which is typically the coordinator. The TC forms a centralized network and allows routers and end devices to join the network if they have proper credentials. In a centralized network, only the TC can issue encryption keys. The TC also establishes a unique TC Link Key for each device on the network as they join and link keys for each pair of devices as requested.

For obvious reasons, the centralized network is much more secure than the distributed one. Most of the Zigbee examples provided inside the STM32WB firmware package are using a centralized network.

### 3.1.3. Application framework

The application framework in ZigBee is very rich and it defines the environment in which application objects are hosted on devices.
Data exchange between Zigbee device is performed in a client server model.
It relies on an Application Profile, Cluster, Attribute model.

The Application profile is a collection of device descriptions, which together form a cooperative application. The Profile defines the data exchange form for the application functions of a ZigBee physical device. A Profile consists of one or more Endpoints, each with one or more clusters associated.

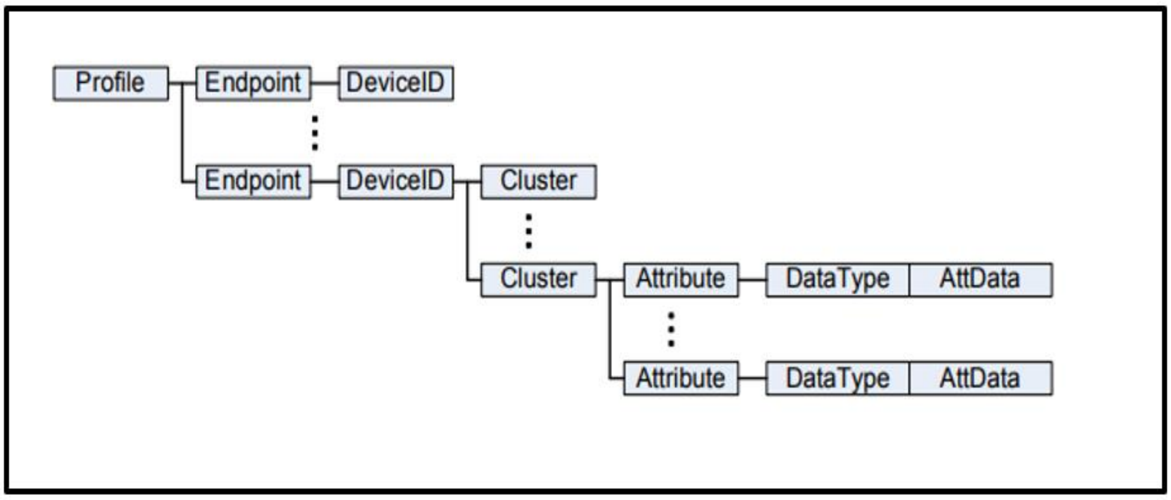


Fig 3 : Zigbee application profile organization

Clusters are a group of commands and attributes that define what a device can do. Clusters are managed by the ZCL (ZigBee Cluster Library).

## 3.2. Zigbee on STM32WB

### 3.2.1. Zigbee firmware supported

Two flavors of the stack are supported on the STM32WB55 device. Both stacks are Zigbee PRO 2017 (revision 22) certified.

| **Stacks supported** | **Firmware associated** |
|---|---|
| Zigbee FFD (Full feature device) | stm32wb5x_Zigbee_FFD_Full_fw.bin |
| Zigbee RFD (Reduced feature device | stm32wb5x_Zigbee_RFD_fw.bin |

- An **FFD** can accept any role in the network. It can be a router, a coordinator or an end device.
- An **RFD** can support only end device role. An **RFD** has a smaller footprint compared to an **FFD.** (Approximatively 254KB versus 300KB).

ST is also providing within a single binary a firmware which support both BLE and Zigbee protocol.

| **Stacks supported** | **Firmware associated** |
|---|---|
| BLE and Zigbee (static mode) | stm32wb5x_BLE_Zigbee_FFD_static_fw.bin |

This binary is used for static concurrent mode applications. An example of such application is provided under:
*Projects\P-NUCLEO-WB55.Nucleo\Applications\BLE_Zigbee* directory.
When using this binary, it is possible to switch from BLE to Zigbee and vice-versa. When the BLE protocol is running, the Zigbee stack is no more running. When the BLE is stopped, the system can switch back to Zigbee. In this case, the Zigbee stack is fully re-initialized.

**Important note:**

Before running any Zigbee application on STM32WB, you need to ensure that the proper firmware is downloaded on the M0. If it is not the case, you need to use STM32CubeProgrammer to load the appropriate binary.

All available Zigbee binaries are located under:
*/Projects/STM32WB_Copro_Wireless_Binaries/STM32WB5x.*

Refer to
*/Projects/STM32WB_Copro_Wireless_Binaries/STM32WB5x/Release_Notes.html* for the detailed procedure on how to change the Wireless Coprocessor binary.

### 3.2.2. Zigbee clusters supported

The Zigbee ecosystem available on STM32WB supports **Zigbee 3.0**. As a matter of fact, it includes BDB (base device behavior), Zigbee Green Power and several specific ZCL clusters as listed below:

| **Nb** | **Cluster ID** | **Cluster name** |
|---|---|---|
| 1 | 0x0000 | Basic |
| 2 | 0x0001 | Power Configuration |
| 3 | 0x0003 | Identify |
| 4 | 0x0004 | Groups |
| 5 | 0x0005 | Scenes |
| 6 | 0x0006 | On/Off |
| 7 | 0x0008 | Level Control |
| 8 | 0x000a | Time |
| 9 | 0x0019 | OTA Upgrade |
| 10 | 0x0020 | Poll Control |
| 11 | 0x0021 | Green Power Proxy |
| 12 | 0x0102 | Window Covering |
| 13 | 0x0202 | Fan Control |
| 14 | 0x0204 | Thermostat User Interface Configuration |
| 15 | 0x0300 | Color control |
| 16 | 0x0301 | Ballast Configuration |
| 17 | 0x0400 | Illuminance Measurement |
| 18 | 0x0402 | Temperature Measurement |
| 19 | 0x0406 | Occupancy Sensing |
| 20 | 0x0502 | IAS Warning Device (WD) |
| 21 | 0x0b05 | Diagnostics |
| 22 | 0x1000 | Touchlink |
| 23 | 0x0002 | Device Temperature Configuration |
| 24 | 0x0007 | On/Off Switch Configuration |
| 25 | 0x0009 | Alarms |
| 26 | 0x000b | RSSI Location |
| 27 | 0x0015 | Commissioning |
| 28 | 0x001a | Power Profile Cluster |
| 29 | 0x0024 | Nearest Gateway Cluster |
| 30 | 0x0101 | Door Lock |
| 31 | 0x0200 | Pump Configuration and Control |
| 32 | 0x0201 | Thermostat |
| 33 | 0x0203 | Dehumidification Control |

| | | |
|---|---|---|
| 34 | 0x0401 | Illuminance Level Sensing |
| 35 | 0x0403 | Pressure Measurement |
| 36 | 0x0405 | Relative Humidity Measurement |
| 37 | 0x0500 | IAS Zone |
| 38 | 0x0501 | IAS Ancillary Control Equipment (ACE) |
| 39 | 0x0700 | Price |
| 40 | 0x0701 | Demand Response and Load Control |
| 41 | 0x0702 | Metering |
| 42 | 0x0703 | Messaging |
| 43 | 0x0704 | Smart Energy Tunneling (Complex Metering) |
| 44 | 0x0800 | Key Establishment |
| 45 | 0x0904 | Voice Over ZigBee |
| 46 | 0x0b01 | Meter Identification |
| 47 | 0x0b04 | Electrical Measurement |

- All these 47 clusters are available through the **STM32_WPAN** Middleware. This Middleware is common to BLE and Thread. For specific needs, a customer may create its own 'proprietary' cluster' if needed.

- The APIs relative to these clusters can be found under the following directory: *\Middlewares\ST\STM32_WPAN\zigbee\stack\include*

# 4. Architecture

## 4.1. Architecture overview

The figure below gives an overview of the overall architecture. It shows in particular the split between the M4 and the M0. All the code running on the M0 is delivered as a binary library (*refer to 3.2.1 for more details*).

The customer has access only to the M4 core and sees the firmware running on M0 as a black box. All the intercommunication between the M4 and M0 is hidden by the framework. Dedicated IPCC channels are allocated for Zigbee.

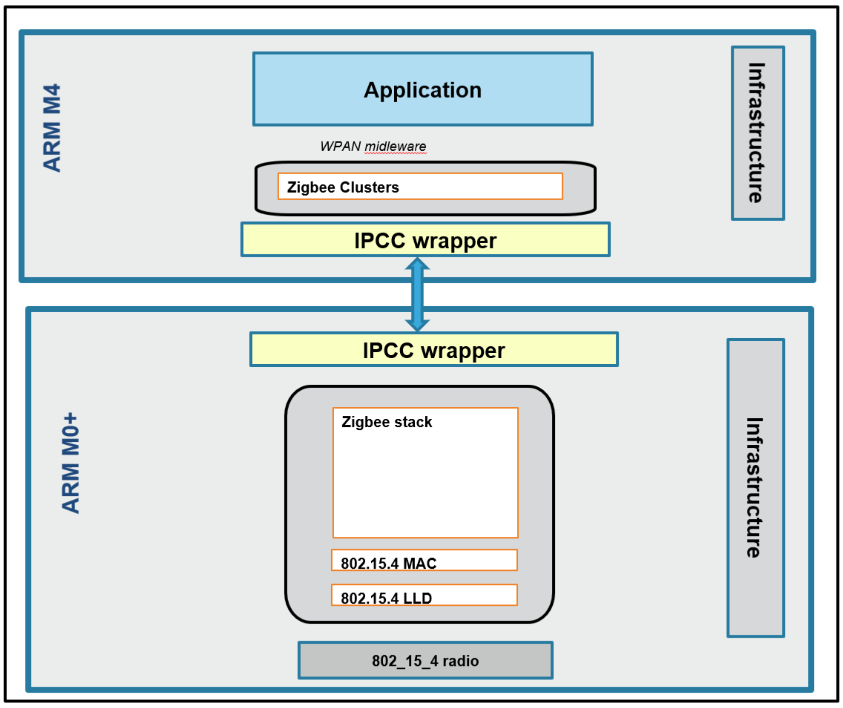


Fig 4 : Zigbee architecture overview on STM32WB

The Zigbee stack is running on top of the 802_15_4 MAC layer which itself use services provided by the 802_15_4 low level driver in charge of controlling the radio.

## 4.2. Zigbee stack layers

The figure shown in more details the different Zigbee layers

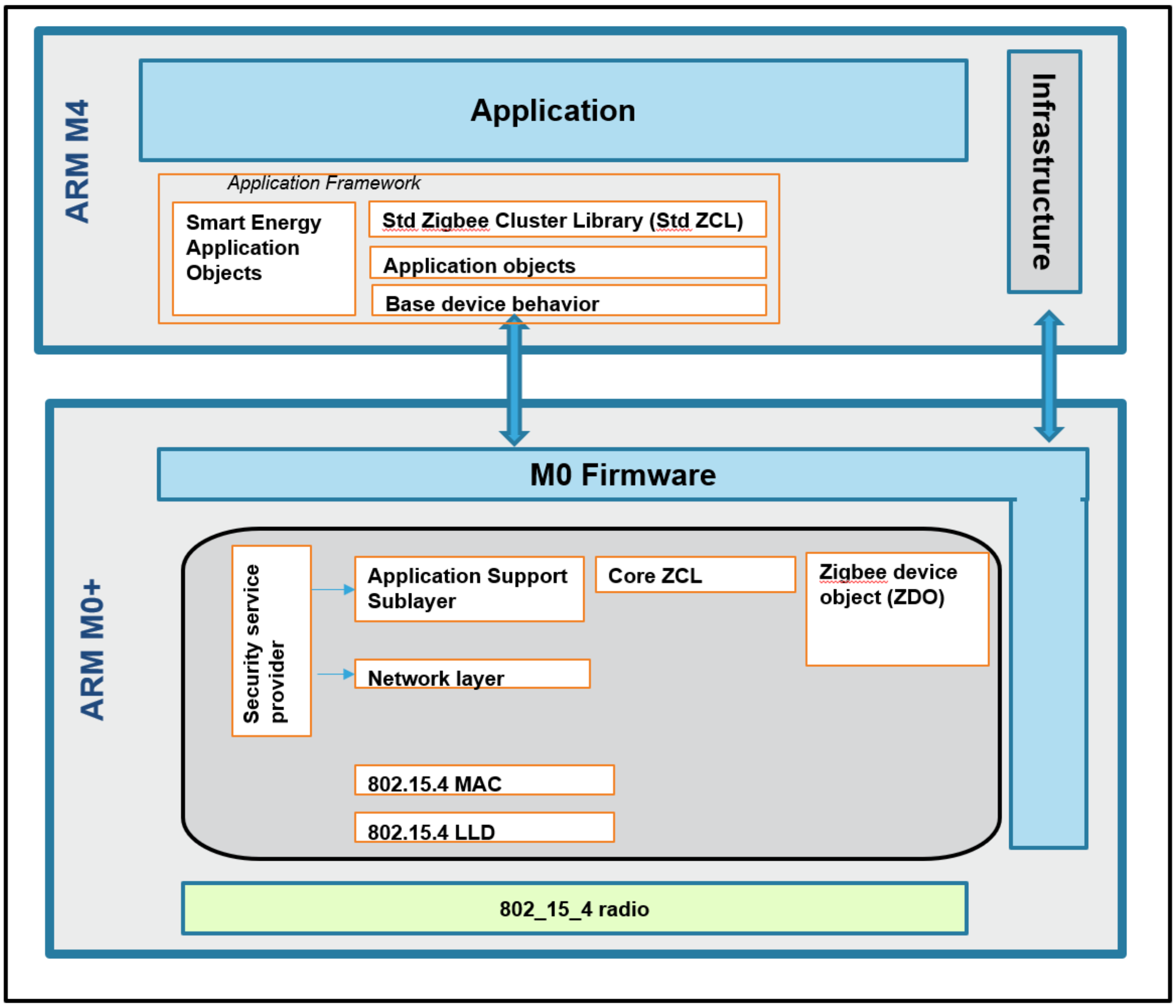


Fig 5 : Zigbee layers and modules

Some of the main modules shown on this figure are described briefly hereafter:

- **ZCL:**
  ZCL is the library which manage the Clusters (**Z**igBee **C**luster **L**ibrary).
  Clusters can be considered as a group of commands and attributes specific to a dedicated kind of application (DoorLock, OnOff, etc.).
  ZCL is defined by the ZigBee Alliance in order to speed the development and

standardization of public profiles. With ZCL, manufacturers are able to quickly build ZigBee products with consistency and compatibility.
A Cluster is a related collection of Commands and Attributes, which together defines an interface to specific functionality. Commands are actions that a cluster can take. Attributes are data or states within a cluster.

- **BDB:**
  BDB stands for **B**ase **D**evice **B**ehavior. It is a standard device type which handles fundamental operations such as commissioning, network security and persistent data management. This device does not need an endpoint.

- **APS**
  APS stands for application support sub-layer. It provides an interface between the network layer (NWK) and the application layer through a general set of services that are used by both the ZDO and the manufacturer-defined application objects.

- **ZDO:**
  ZDO stands for **Z**igbee **D**evice **O**bject. ZDO communicates over endpoint 0. The Profile used by ZDO is the ZigBee Device Profile with provides the following communication functions:
    - Device Discovery
    - Service Discovery
    - Network Management
    - Binding management

- **NWK:**
  The network layer is required to provide functionality to ensure correct operation of the IEEE 802.15.4-2003 MAC sub-layer and to provide a suitable service interface to the application layer.

# 5. STM32WB Zigbee applications

## 5.1. Applications available

Several Zigbee application are delivered inside the STM32WB firmware package. These applications are available on P-NUCLEO-WB55.Nucleo boards and on P-NUCLEO-WB55.USBDongle as well.

The full list of applications is available in STM32CubeProjectsList.html file (under *Projects* directory).

Here are some examples of such applications:

| **Projects name** | **Description** |
|---|---|
| Zigbee_Commissioning_Client_Coord<br>Zigbee_Commissioning_Server_Router | How to use Commissioning cluster on a centralized Zigbee network. |
| Zigbee_DevTemp_Server_Coord<br>Zigbee_DevTemp_Client_Router | How to use Device Temperature cluster on a Centralized Zigbee network. |
| Zigbee_Diagnostic_Server_Coord<br>Zigbee_Diagnostic_Client_Router | How to use Diagnostic on a centralized Zigbee network. |
| Zigbee_DoorLock_Server_Coord<br>Zigbee_DoorLock_Client_Router | How to use Door Lock cluster on a centralized Zigbee network. |
| Zigbee_IAS_WD_Server_Coord<br>Zigbee_IAS_WD_Client_Router | How to use IAS WD cluster on a centralized Zigbee network. |
| Zigbee_MeterId_Server_Coord<br>Zigbee_MeterId_Client_Router | How to use Meter Identification cluster on a centralized Zigbee network. |
| Zigbee_OnOff_Client_Distrib<br>Zigbee_OnOff_Server_Distrib | How to use OnOff cluster on a distributed Zigbee network. |
| Zigbee_OnOff_Server_Coord<br>Zigbee_OnOff_Client_Router | How to use OnOff cluster on a centralized Zigbee network. |
| Zigbee_PollControl_Client_Coord<br>Zigbee_PollControl_Server_SED | How to use Poll Control cluster on a centralized Zigbee network. |
| Zigbee_PowerProfile_Client_Coord<br>Zigbee_PowerProfile_Server_Router | How to use Power Profile cluster on a centralized Zigbee network. |

| | |
|---|---|
| Zigbee_PressMeas_Server_Coord<br>Zigbee_PressMeas_Client_Router | How to use Pressure Measurement cluster on a Centralized Zigbee network |
| Zigbee_SE_Msg_Client_Coord<br>Zigbee_SE_Msg_Server_Router | How to use SE Messaging cluster on a Centralized Zigbee network. |

The purpose of these applications is mainly to provide simple examples highlighting the usage of specific clusters.

The overall Zigbee application framework is based on a Client Server model. Each Zigbee application provided inside the STM32WB firmware package is delivered as two separate projects: one project handling the server part, and the other one handling the client part. To run these applications, it is required to have one board configured in client/coordinator mode, and all the other ones configured in server/router mode.

**Note:** It is possible to map a Client and or Server on any Zigbee, independently from the role it supports (ZC/ZR/ZED). Moreover, a single device can be a client and a server at the same time.

## 5.2. Zigbee application framework

### 5.2.1. Application framework

All projects are built using the same framework.
In all projects, the zigbee use case is set, defined and implemented inside the **app_zigbee.c** file.
Under:*Projects\Board_X\Applications\Zigbee\Zigbbe_Y_app\STM32_WPAN\App*.

All the other files present in the application projects are mainly used for the global infrastructure management (Interrupt management, IPCC wrapper, system startup and configuration, etc…)

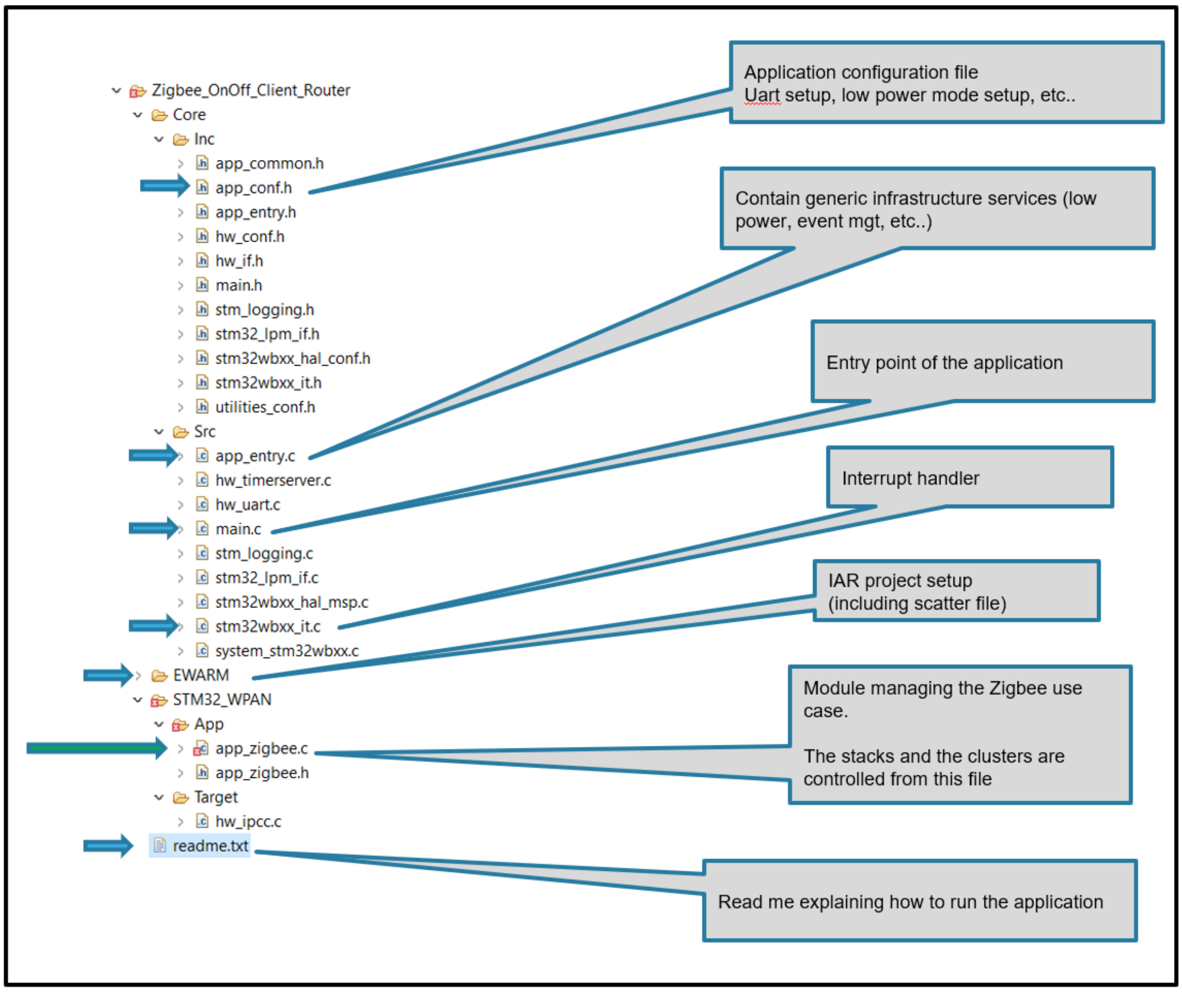

Zigbee_OnOff_Client_Router
Core
Inc
app_common.h
app_conf.h
app_entry.h
hw_conf.h
hw_if.h
main.h
stm_logging.h
stm32_lpm_if.h
stm32wbxx_hal_conf.h
stm32wbxx_it.h
utilities_conf.h
Src
app_entry.c
hw_timerserver.c
hw_uart.c
main.c
stm_logging.c
stm32_lpm_if.c
stm32wbxx_hal_msp.c
stm32wbxx_it.c
system_stm32wbxx.c
EWARM
STM32_WPAN
App
app_zigbee.c
app_zigbee.h
Target
hw_ipcc.c
readme.txt
Application configuration file
Uart setup, low power mode setup, etc..
Contain generic infrastructure services (low power, event mgt, etc..)
Entry point of the application
Interrupt handler
IAR project setup (including scatter file)
Module managing the Zigbee use case.
The stacks and the clusters are controlled from this file
Read me explaining how to run the application

Fig 6 : Zigbee OnOff cluster application

### 5.2.2. Traces

Both traces coming from the stack itsef (running on M0 core) and  from the application itlsef (running on M4 side) are managed by the M4 and are routed through an  UART (configured at compilation time using **app_conf.h** file).
To get the traces you need to connect your Board to the Hyperterminal (through the STLink Virtual COM Port). The UART must be configured as follows:

- BaudRate = 115200 baud
- Word Length = 8 Bits
- Stop Bit = 1 bit
- Parity = none
- Flow control = none

## 5.3. Zigbee OnOff cluster application

One of the easiest ways to start with Zigbee on STM32WB is to play with the **Zigbee_OnOff** application available in centralized network.
To do so is requested to have the following setup:

- 1 STM32WB55xx board loaded with:
  - wireless coprocessor: stm32wb5x_Zigbee_FFD_fw.bin
  - application: Zigbee_OnOff_Server_Coord

- 1 or more STM32WB55xx board loaded with:
  - wireless coprocessor: stm32wb5x_Zigbee_FFD_fw.bin
  - application: Zigbee_OnOff_Client_Router

The purpose of this application is to show how to create a Zigbee centralized network, and how to communicate from one node to another one using the OnOff cluster. Once the Zigbee mesh network is created, the user can send requests from the client to the server through the push button in order to make a LED toggling.

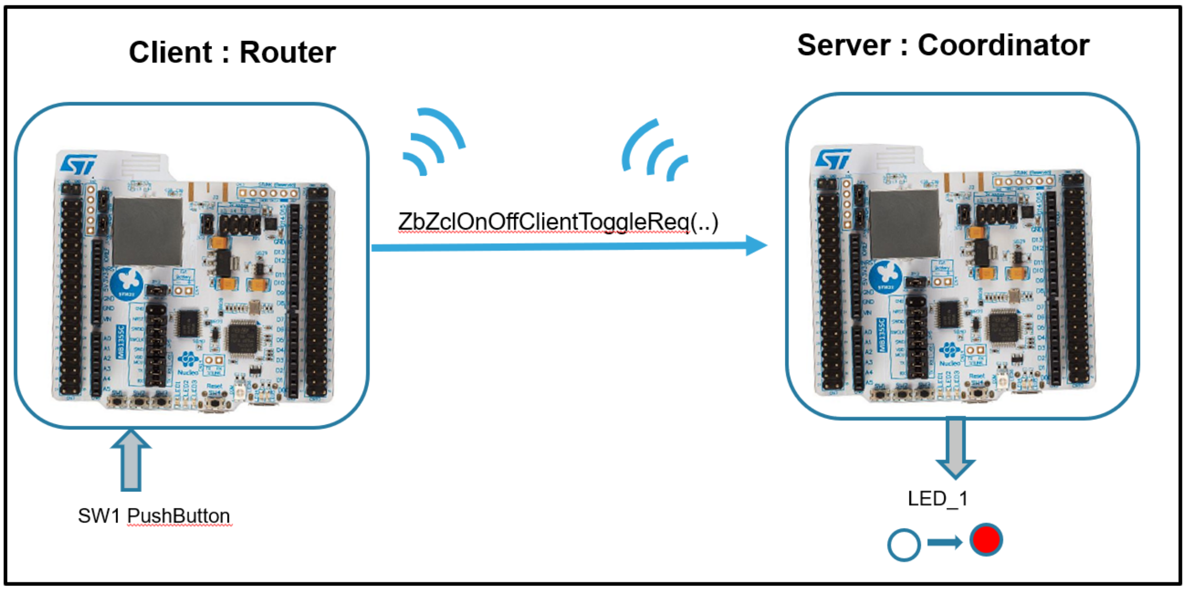


Fig 7 : Zigbee OnOff cluster application

- **Joining procedure**

  The joining procedure is managed by a specific task named (TASK_ZIGBEE_NETWORK_FORM). This task launched as soon as the Zigbee stack is initialized. The blue led light on when the joining procedure has been successful. When for any reason, the joining procedure fails, the joining task is rescheduled.

- **End point management**

  For any Zigbee application, a Profile consists of one or more EndPoints, each with one or more clusters and a vertical structure of the attributes. The generic links between all these entities is the following:

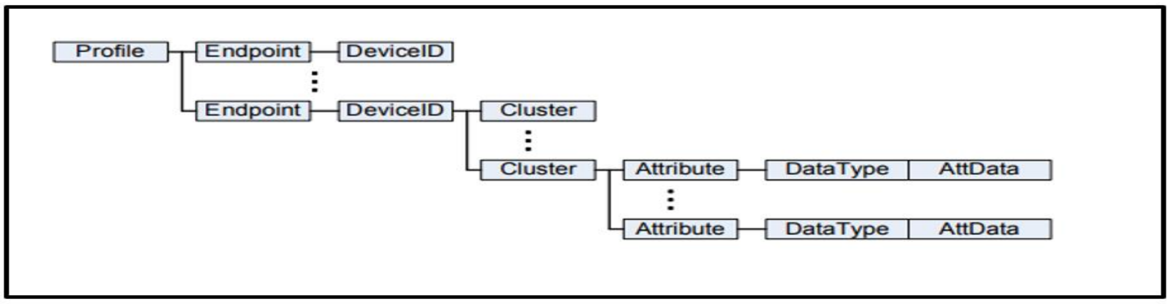
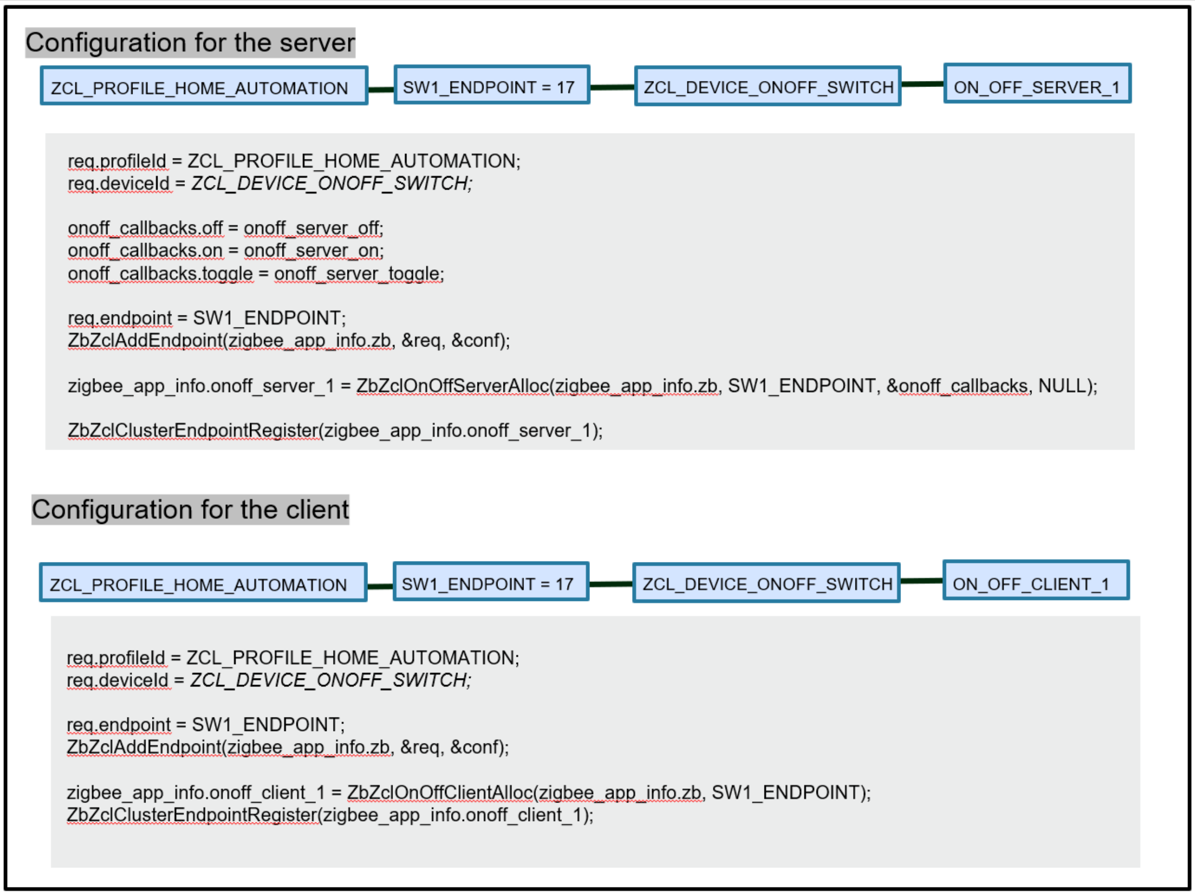


Fig 8 : Zigbee endpoint/cluster relationship

For the OnOff application, the end point configuration is managed as follow :



Configuration for the server

```
req.profileId = ZCL_PROFILE_HOME_AUTOMATION;
req.deviceId = ZCL_DEVICE_ONOFF_SWITCH;

onoff_callbacks.off = onoff_server_off;
onoff_callbacks.on = onoff_server_on;
onoff_callbacks.toggle = onoff_server_toggle;

req.endpoint = SW1_ENDPOINT;
ZbZclAddEndpoint(zigbee_app_info.zb, &req, &conf);

zigbee_app_info.onoff_server_1 = ZbZclOnOffServerAlloc(zigbee_app_info.zb, SW1_ENDPOINT, &onoff_callbacks, NULL);

ZbZclClusterEndpointRegister(zigbee_app_info.onoff_server_1);
```

Configuration for the client

```
req.profileId = ZCL_PROFILE_HOME_AUTOMATION;
req.deviceId = ZCL_DEVICE_ONOFF_SWITCH;

req.endpoint = SW1_ENDPOINT;
ZbZclAddEndpoint(zigbee_app_info.zb, &req, &conf);

zigbee_app_info.onoff_client_1 = ZbZclOnOffClientAlloc(zigbee_app_info.zb, SW1_ENDPOINT);
ZbZclClusterEndpointRegister(zigbee_app_info.onoff_client_1);
```

Fig 9 : Zigbee OnOff application end point configuration

- **Group management**

  On this example, multiple OnOff clients can interact with the unique OnOff Server which act as coordinator. The STM32WB Zigbee framework allows accessing the APS layer. Through this layer, it is possible to manage groups. A group being a collection of nodes inside a network. Some APS primitives allow the higher layer to request that group membership for a particular group to be added for a particular endpoint.

- **Persistent data storage**

  The purpose of the **Zigbee OnOff** example described in this section, was to provide an application as simple as possible to start with. The persistent data storage management feature is available in the STM32WB Zigbee framework but is not used in this simple Zigbee OnOff example. A dedicated example will be available in the next firmware package highlighting the usage of persistent data. In the meantime, you can have a look to the *BLE_Zigbee_Static* application to see how it is possible to start the stack from persistence.

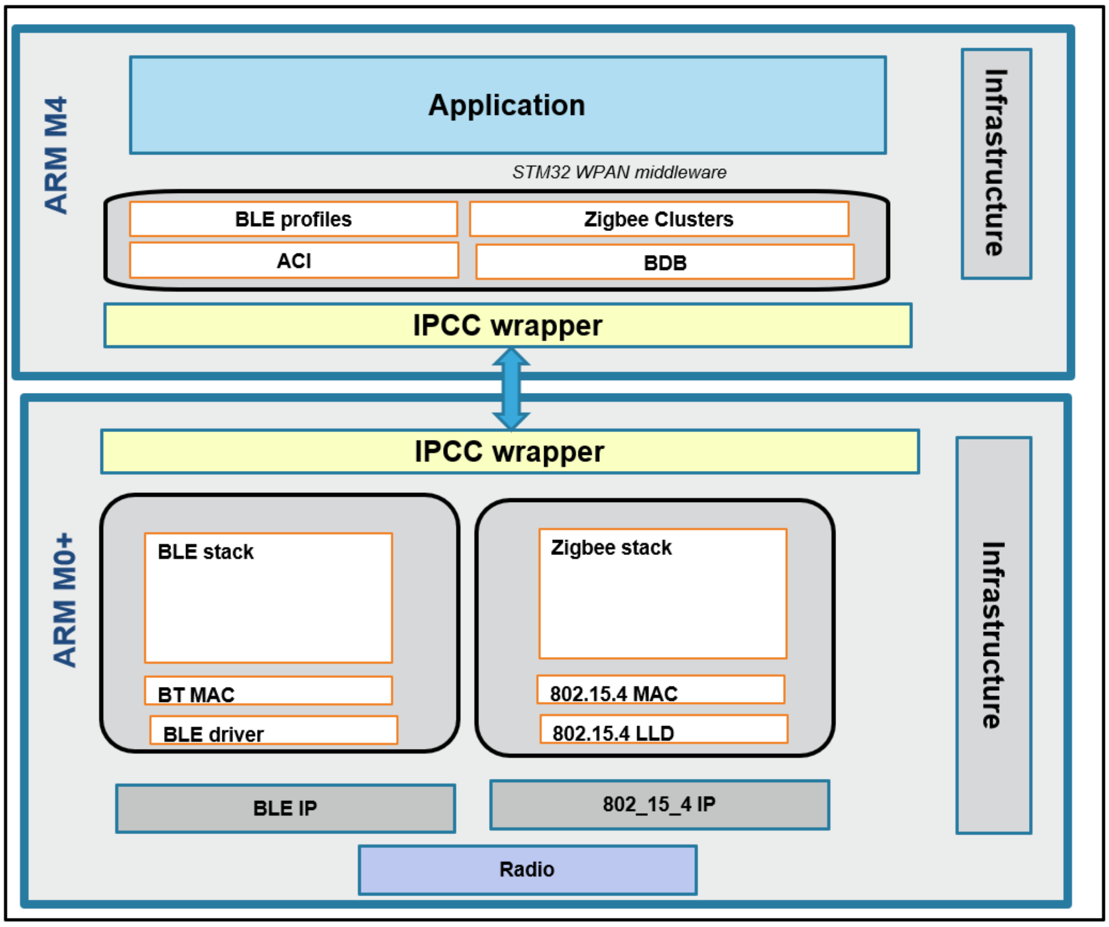
  In the **Zigbee OnOff** example described in this section, all data needed by the network node are stored only in RAM. They are maintained only while the node is powered. After a shutdown, all these data are lost. These data can be stack context values needed by the network and/or application data linked to cluster attributes. On a real system is it important to be able to restart from persistence. This feature is fully available using the STM32WB Zigbee framework.

# 6. ANNEXE

## 6.1. Static concurrent mode

An example of static concurrent mode (BLE/Zigbee) is provided in the STM32WB firmware package. This application is located under *Projects\P-NUCLEO-WB55.Nucleo\Applications\BLE_Zigbee* directory.

When running this use case, the ‘static concurrent mode’ device can switch from BLE to Zigbee and vice-versa. This device can be connected through BLE to a smartphone running the "ST BLE Sensor" Application and once the BLE activity is stopped, it can join a Zigbee network. Then, once the Zigbee application has been fully stopped, it is possible to go back to BLE again.



# 7. Revision history

| Date | Revision | Changes |
|---|---|---|
| 7 April 2020 | V0.1 | Initial release |
| | | |