AN4642

Application note

Overview of the BLE Profiles application for X-CUBE-BLE1

Introduction

The X-CUBE-BLE1 software expansion for STM32Cube includes sample applications for STM32 Nucleo development boards when connected to the X-NUCLEO-IDB05A2 expansion board.

One of the included applications is the BLE Profiles that supports BLE peripheral and central roles, and Apple notification center service.

The application is based on the standard GATT-based profiles described in the Bluetooth specifications and runs on the STM32 Nucleo development board that acts as the GAP peripheral device. It sends profile-related data to the GAP central device (Android™ or iOS™ smartphone).

Note: The BLE Profiles application is also available when connecting an STM32 Nucleo development board to the X-NUCLEO-IDB05A1 expansion board.

1 Acronyms and abbreviations

Table 1. List of acronyms

Acronym	Description
ACI	Application controller interface
ANS	Alert notification server
ATT	Attribute protocol
BLE	Bluetooth low energy
BLS	Blood pressure sensor
BSP	Board support package
BT	Bluetooth
FMT	Find me target
GAP	Generic access profile
GATT	Generic attribute profile
GLS	Glucose sensor
GUI	Graphical user interface
HAL	Hardware abstraction layer
HCI	Host controller interface
HRS	Heart rate sensor
IDE	Integrated development environment
L2CAP	Logical link control and adaptation protocol
LED	Light emitting diode
LL	Link layer
LPM	Low power manager
MCU	Microcontroller unit
PCI	Profile command interface
PXP	Proximity profile
PXR	Proximity reporter
PHY	Physical layer
SIG	Special interest group
SM	Security manager
SPI	Serial peripheral interface
TIP	Time profile
TS	Time server
UUID	Universally unique identifier

2 BLE Profiles application

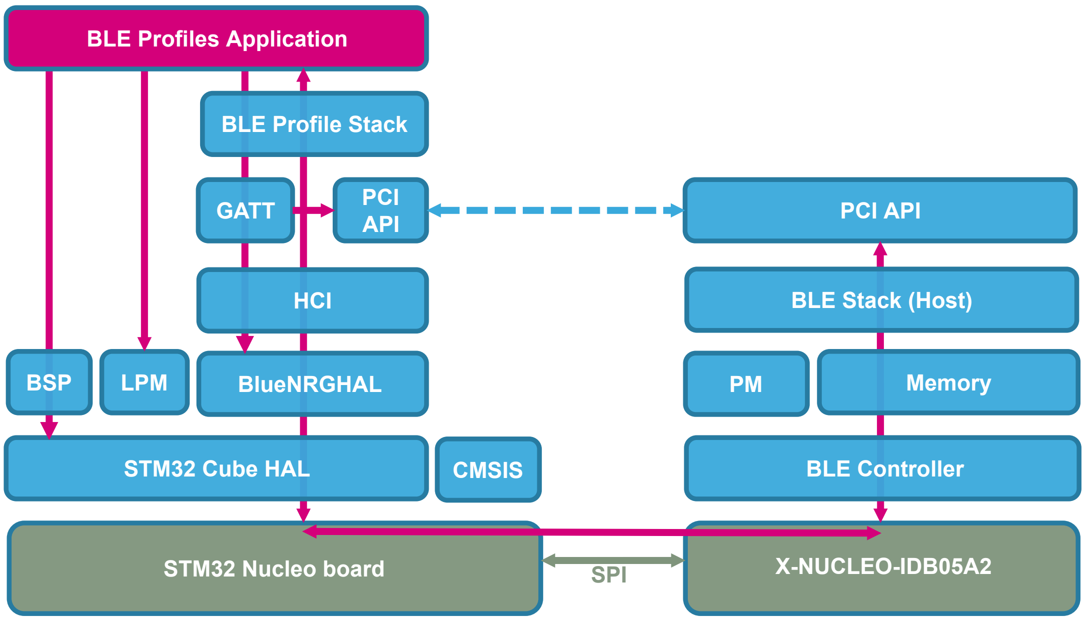
2.1 BLE Profiles application layers

The BLE Profiles application layers are:

- STM32Cube HAL
- Board support package (BSP)
- BlueNRG-MS or BlueNRG-M0 HAL
- Profile command interface (PCI)
- GATT profile
- BLE profile stack

The GATT profile uses PCI API and HCI to send BLE commands to the BlueNRG-MS or BlueNRG-M0 device.

Figure 1. BLE Profiles application architecture



2.1.1 Profile command interface (PCI)

The PCI is the main API for all applications that use the GATT-based profiles on the BlueNRG device.

The PCI exposes the APIs used to send commands to the BlueNRG-MS/BlueNRG-M0 device.

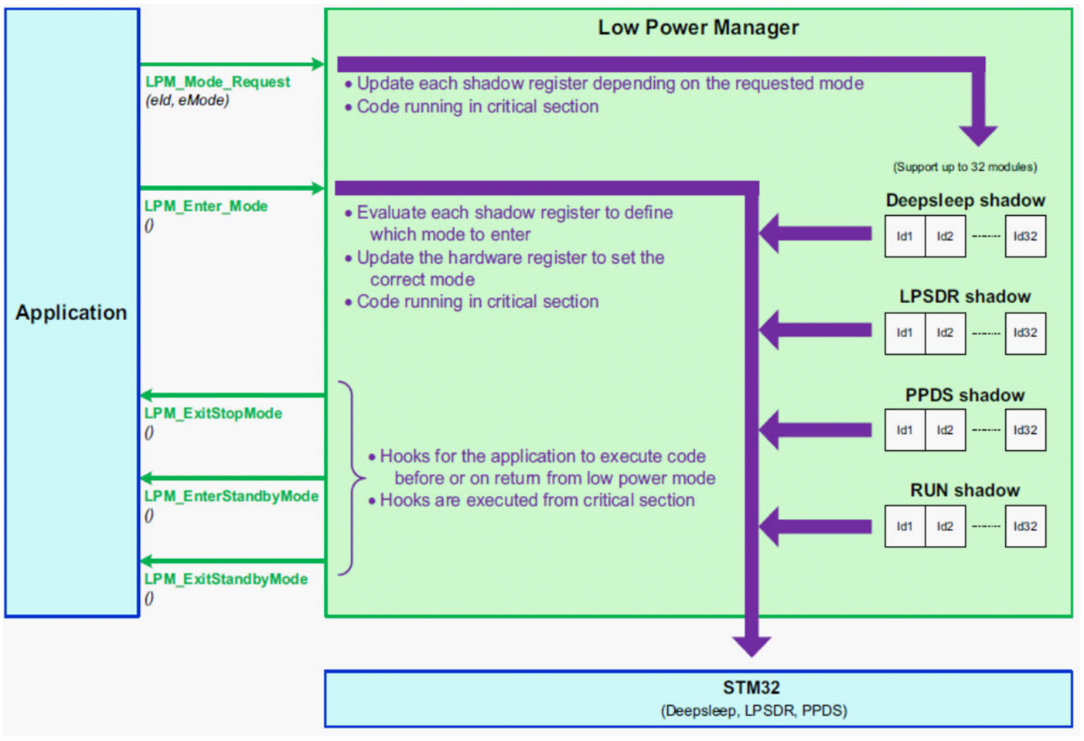
2.1.2 Low power manager (LPM)

The LPM lets the application allow the system drop to low-power states whenever necessary conditions are met. The low power manager provides a set of APIs that enable the application to support active power management.

It provides support for:

- the application to establish the lowest allowable power mode
- informing the system that no more power is required
- callbacks for the implementation of specific routines when coming out of, or going into low power modes (Stop mode or Standby mode)

Figure 2. **Low power manager structure overview**


Low Power Manager
Application
LPM_Mode_Request
(eId, eMode)
LPM_Enter_Mode
()
LPM_ExitStopMode
()
LPM_EnterStandbyMode
()
LPM_ExitStandbyMode
()
• Update each shadow register depending on the requested mode
• Code running in critical section
(Support up to 32 modules)
Deepsleep shadow
• Evaluate each shadow register to define which mode to enter
• Update the hardware register to set the correct mode
• Code running in critical section
LPSDR shadow
PPDS shadow
RUN shadow
Id1
Id2
Id32
• Hooks for the application to execute code before or on return from low power mode
• Hooks are executed from critical section
STM32
(Deepsleep, LPSDR, PPDS)

2.1.3 Low power manager API

The low power manager provides the following APIs to put the system in a specific low power mode:

- `LPM_Mode_Request()`: specifies the lowest supported power mode
- `LPM_Enter_Mode()`: makes the system enter the low power mode
- `LPM_ExitStopMode()`:, called by the LPM in a critical section, allows the application to implement dedicated code before exiting the critical section
- `LPM_Enter_StandbyMode()`:, called by the LPM in a critical section, allows the application to implement dedicated code before entering standby mode

2.1.4 BLE peripheral profile stack

The BLE profile stack layer implements the Bluetooth Low Energy profiles used by the applications. The PCI allows communicating with the controller and defines the necessary functions and callbacks for the applications to communicate with the profiles.

2.1.5 BSP layer

The BSP software layer supports the peripherals on the STM32 Nucleo development board, excluding the MCU, and provides a limited set of APIs which provides a programming interface for certain board-specific peripherals like the LED, the user button, etc. The interface also helps to identify the specific board version.

2.2 MCU power modes

The BLE Profiles application leverages the advanced STM32 MCU low-power features by the following MCU modes:

- RUN mode that is the standard execution mode where all the MCU features are available.
- STOP mode with RTC:
  - that achieves the lowest power consumption while retaining the RAM and register contents and real-time clock
  - stops all clocks in the VCORE domain (the PLL, MSI RC, HSE crystal and HSI RC oscillators are disabled)
  - lets LSE or LSI still running
  - puts the voltage regulator in low-power mode
  - wakes the device up from Stop mode through any of the EXTI lines, in 3.5 µs (the MCU can serve the interrupt or resume the code)

The low power manager is used to exploit low power features of the STM32 MCU (see Section 2.1.3 Low power manager API for details).

2.3 Role overview

The BLE Profiles application supports BLE peripheral and central roles, and Apple notification center service.

The supported slave profiles (peripheral role) are:

- Alert notification service
- Blood pressure service
- Find me target
- Glucose service
- Health thermometer service
- Heart rate service
- Human interface device service (not supported by NUCLEO-L053R8)
- Proximity reporter
- Time server

The supported master profiles (central role) are:

- Alert notification client
- Blood pressure collector
- Find me locator
- Glucose collector
- Health thermometer collector
- Heart rate collector
- Time client

2.4 Peripheral role

2.4.1 Slave profiles

2.4.1.1 Alert notification service

The alert notification service (ANS) runs the Alert Notification Profile described in the Bluetooth Profile Specifications. A client device can receive different types of alerts and event information as well as information on the count of new alerts and unread items in the server device (the STM32 Nucleo board).

To demonstrate the function, the ANS exposes the "New e-mail" alert with a sample text message. Any central device running the Alert Notification Client is notified when a new e-mail is received.

The ANS device initializes the profile and starts advertising its address and services, while the client device scans for an ANS device and sends a connection request if it detects one, enabling alert notification.

2.4.1.2 *Blood pressure service*

The blood pressure monitor service runs the Blood Pressure Profile (BPP) as described in the Bluetooth Profile Specifications. It enables the role of the blood pressure sensor (BLS) in the STM32 Nucleo device.

The application sends periodic data to simulate blood pressure readings (and related data) as an actual blood pressure sensor is not present on the STM32 Nucleo board.

Any central device running the Blood Pressure Profile can collect data from the blood pressure monitor service.

The BLS initializes the Blood Pressure Profile and begins advertising its address and services while the collector scans for a BLS and sends a connection request if it detects one.

Once connection is established, the BLS sends data over the connection link.

2.4.1.3 *Find me target*

The Find Me Target (FMT) application runs the Find Me Profile described in the Bluetooth Profile Specifications. It causes the STM32 Nucleo board to produce an alert signal (blinking LED) when a button is pressed on a Find Me Locator device (when acting as the central device).

The FMT device initializes the profile and starts advertising its address and services while the Locator scans for an FMT and sends a connection request if it detects one.

2.4.1.4 *Glucose service*

The Glucose Monitor (GLM) application runs the Glucose Profile as described in the Bluetooth Profile Specifications. It enables the role of the Glucose Sensor (GLS) in the STM32 Nucleo.

The application sends periodic data to simulate glucose concentration readings as an actual glucose sensor is not present on the STM32 Nucleo board. Other information describing the sensor position and its supported features is also transmitted.

Any central device running the Glucose Profile can collect data from the Glucose Monitor.

The GLS initializes the Glucose Profile and starts advertising its address and services while the Collector scans for a GLS and sends a connection request if it detects one. Once connection is established, the GLS sends data over the connection link.

2.4.1.5 *Health thermometer service*

The Health Thermometer Monitor (HTM) application runs the Health Thermometer Profile described in the Bluetooth Profile Specifications. It enables the role of the Health Thermometer Sensor (HTS) in the STM32 Nucleo.

The application sends periodic data to simulate body temperature readings as an actual temperature sensor is not present on the STM32 Nucleo board.

Any central device running the Health Thermometer Profile can collect data from the Health Thermometer Monitor.

The HTS device initializes the Health Thermometer Profile and starts advertising its address and services while the Collector scans for an HTS and sends a connection request if it detects one. Once connection is established, the HTS sends data over the connection link.

2.4.1.6 *Heart rate service*

The Heart Rate Monitor (HRM) application runs the Heart Rate Profile described in the Bluetooth Profile Specifications. It enables the role of the Heart Rate Sensor (HRS) in the STM32 Nucleo.

The application sends periodic data to simulate heart rate measurements as the actual sensor is not present on the STM32 Nucleo board.

Any central device running the Heart Rate Profile can collect data from the Heart Rate Monitor.

The HRS initializes the Heart Rate Profile and starts advertising its address and services while the Collector scans for a HRS device and sends a connection request if it detects one. Once connection is established, the HRS sends data over the connection link.

2.4.1.7 *Human interface device service*

The Human Interface Device (HID) application runs the HID Profile described in the Bluetooth Profile Specifications. It enables the role of a virtual keyboard on the STM32 Nucleo.

By pressing the user button, the application sends text data ("AB") to simulate the pressing of "A" and "B" on the keyboard.

Any BLE-enabled central device paired with the STM32 Nucleo device running the HID Profile is able to receive the "AB" string.

2.4.1.8 *Proximity reporter*

The Proximity (PXP) application runs the Proximity Profile described in the Bluetooth Profile Specifications. It enables the role of the Proximity Reporter in the STM32 Nucleo.

A device running the Proximity Monitor can immediately signal when a peer device moves away and the connection drops or when the path loss exceeds a pre-defined threshold.

The Proximity Reporter initializes the profile and starts advertising its address and services while the monitor scans for a PXR and sends a connection request, if it detects one, to enable alert notifications and/or set the path loss level.

2.4.1.9 *Time server*

The Time (TIP) application runs the Time Profile described in the Bluetooth Profile Specifications. It enables the role of the Time Server (TS) in the STM32 Nucleo device.

A client device (when acting as the central device) can obtain date and time, and related information exposed by the Current Time Service in the STM32 Nucleo.

The TS device initializes the profile and then starts advertising its address and services while the Time Client scans for a TS and sends a connection request, if it detects one, to collect date and time data.

2.4.2 Peripheral profile application files

The directory structure of the Profiles application is shown below. You can find application-specific source code in the following files:

- $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\Applications\\Profiles_LowPower\\Project\\Src\\main.c:
  - performs STM32 Nucleo board initialization and calls the generic profile layer initialization defined in the profile_application.c source file
  - defines the loop handling the main processes:
    - `HCI_Process()`, generic for all profiles
    - `profileStateMachineFunc()` which is specific for each profile state machine
    - `profileApplicationProcessFunc()` which implements a specific profile according to the standard specifications described in Section 2.4 Peripheral role
- $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\Applications\\Profiles_LowPower\\Project\\Src\\profile_application.c:
  - contains the source code to initialize the BlueNRG-MS/BlueNRG-M0 Profile stack, set TX power and security parameters common to all profiles
  - defines the callback for handling generic BLE events

According to the list in Section 2.4 Peripheral role, the following files contain source code for profile context initialization, application event handling, profile initialization, advertisement and application state machine management:

- **Alert Notification Service** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\ Applications\\Profiles_LowPower\\Project\\Src\\ans_profile_application.c
- **Blood Pressure Service** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\ Applications\\Profiles_LowPower\\Project\\Src\\bps_profile_application.c
- **Find Me Target** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\ Applications\\Profiles_LowPower\\Project\\Src\\fmt_profile_application.c
- **Glucose Service** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\ Applications\\Profiles_LowPower\\Project\\Src\\gs_profile_application.c
- **Health Thermometer Service** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\ Applications\\Profiles_LowPower\\Project\\Src\\htm_profile_application.c
- **Heart Rate Service** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\ Applications\\Profiles_LowPower\\Project\\Src\\hrm_profile_application.c
- **Human Interface Device** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\ Applications\\Profiles_LowPower\\Project\\Src\\hid_profile_application.c
- **Proximity Reporter** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\ Applications\\Profiles_LowPower\\Project\\Src\\pr_profile_application.c
- **Time Server** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\ Applications\\Profiles_LowPower\\Project\\Src\\ts_profile_application.c

2.4.3 Peripheral profile initialization

The application must initialize various hardware blocks and data structures before using them. The following sections describe two required types of initialization.

2.4.3.1 Generic initialization

The application must correctly initialize various hardware blocks on the STM32 Nucleo board before using them (for further details, refer to UM1873 available at www.st.com).

2.4.3.2 BLE profile specific initialization

The application initializes the BLE stack and BLE profile to use via the following APIs:

- `BLE_Profile_Init()`: performs generic initialization for BLE profiles (see profile_application.c file).
- Profile specific initialization: the application must initialize the individual profile that it intends to use, the PCI layer provides the API to perform this initialization.
- Each *_profile_application.c source file defines and implements the specific `Init_Profile()` API.

2.4.4 Peripheral profile application state machine

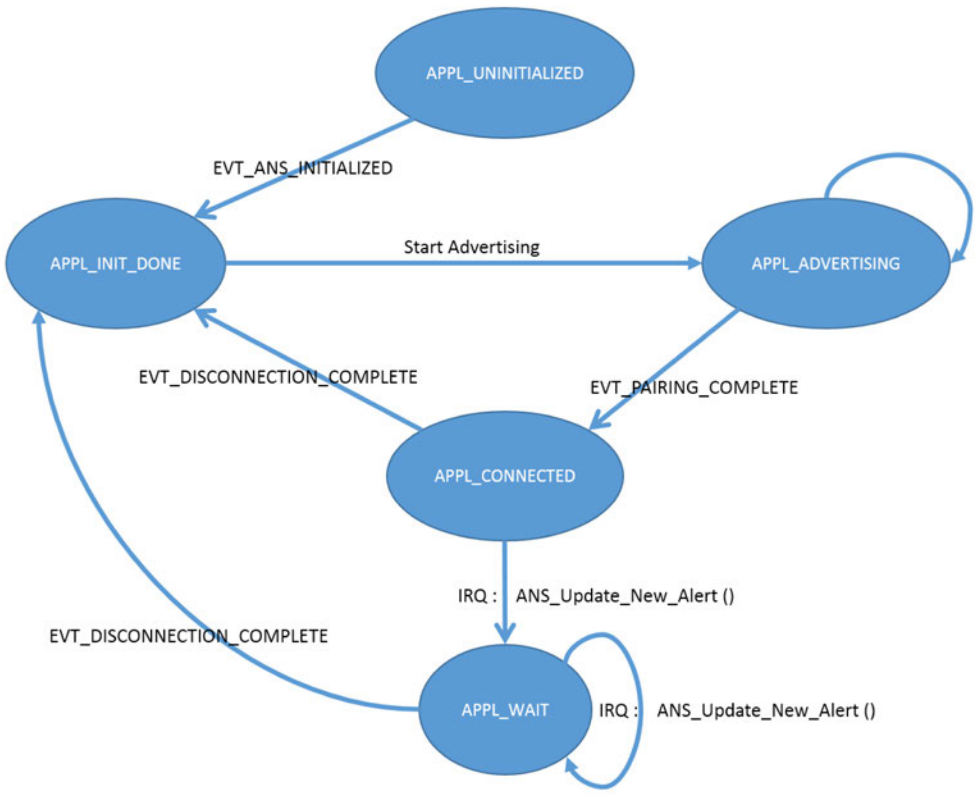
2.4.4.1 ANS application state machine

When the STM32 Nucleo board is powered on, the ANS application is in `APPL_UNINITIALIZED` state and initializes the hardware and the BLE Alert Notification Server (Section 2.4.3 Peripheral profile initialization).

Once the initialization is complete, the ANS application receives the `EVT_ANS_INITIALIZED` event and enters APPL_INIT_DONE state. The application and starts advertising its characteristics so that the central device can discover and choose whether to connect to it, upon which the peripheral enters the `APPL_CONNECTED` state.

In `APPL_CONNECTED` state, the new alert (e.g., a new e-mail alert) is sent to the central device when an interrupt (timer) is received, after which the peripheral enters `APPL_WAIT` state and waits for the interrupt (timer).

Figure 3. ANS application state machine



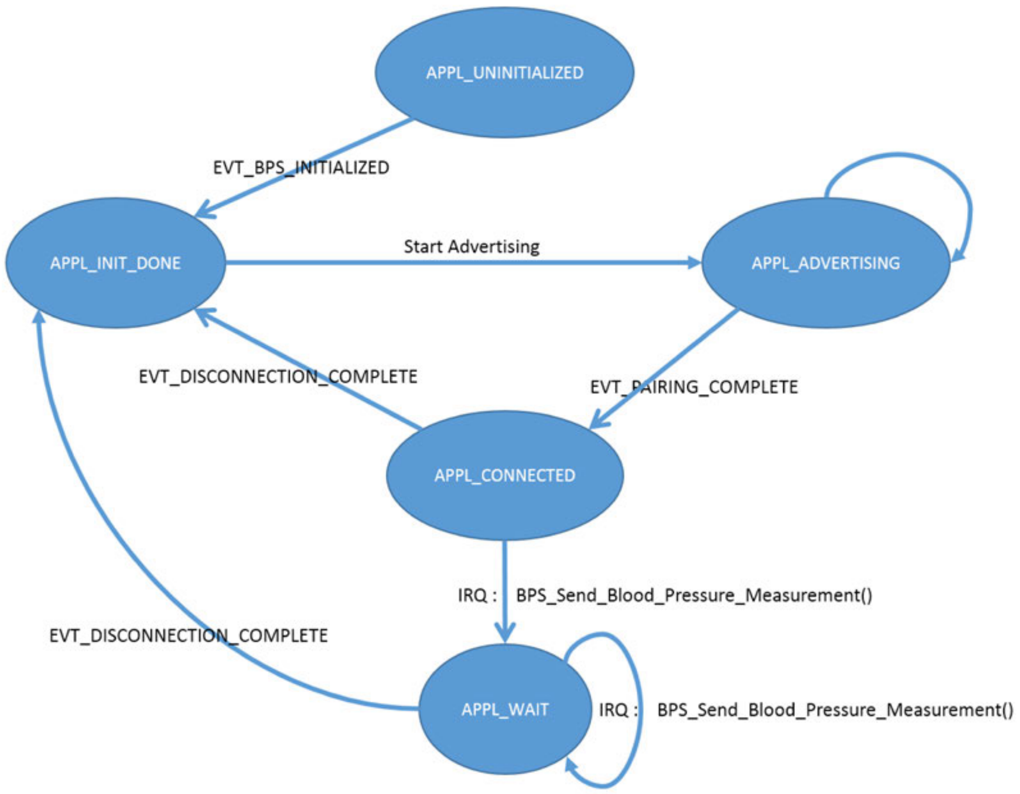
2.4.4.2 *BLP application state machine*

When the STM32 Nucleo board is powered on, the BLP application is in the `APPL_UNINITIALIZED` state and initializes the hardware and the BLE Blood Pressure profile (Section 2.4.3 Peripheral profile initialization).

Once the initialization is complete, the BLP application receives the `EVT_BPS_INITIALIZED` event, enters `APPL_INIT_DONE` state and starts advertising its characteristics so that the central device can detect and choose whether to connect to it, upon which the peripheral enters `APPL_CONNECTED` state.

In the `APPL_CONNECTED` state, the Blood Pressure measurement is sent to the central device when an interrupt (timer) is received, after which the peripheral device enters the `APPL_WAIT` state and waits for the interrupt (timer).

Figure 4. BLP application state machine


APPL_UNINITIALIZED
EVT_BPS_INITIALIZED
APPL_INIT_DONE
Start Advertising
APPL_ADVERTISING
EVT_DISCONNECTION_COMPLETE
EVT_PAIRING_COMPLETE
APPL_CONNECTED
IRQ : BPS_Send_Blood_Pressure_Measurement()
EVT_DISCONNECTION_COMPLETE
APPL_WAIT
IRQ : BPS_Send_Blood_Pressure_Measurement()

2.4.4.3 *FMT application state machine*

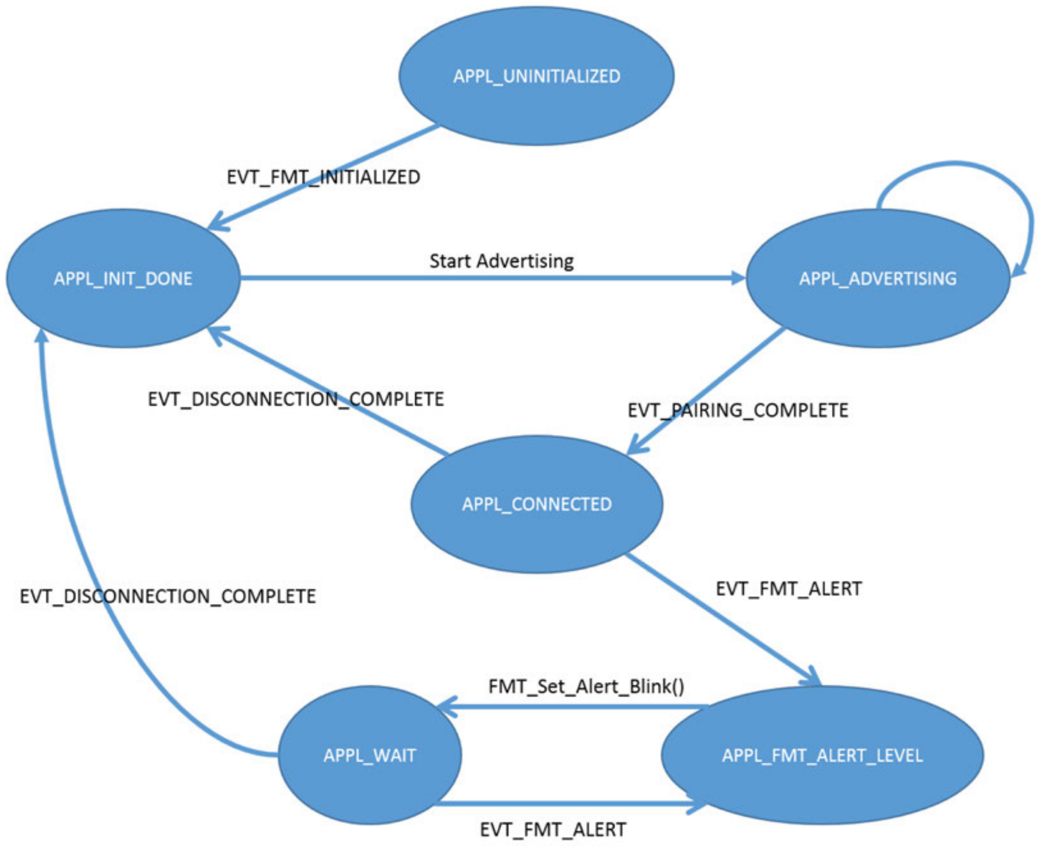
When the STM32 Nucleo board is powered on, the FM application is in `APPL_UNINITIALIZED` state and initializes the hardware and the BLE Find Me Target (Section 2.4.3 Peripheral profile initialization).

Once the initialization is complete, the FMT application receives the `EVT_FMT_INITIALIZED` event and enters `APPL_INIT_DONE` state. The application then starts advertising its characteristics so that the central device can detect and choose whether to connect to it, upon which the peripheral enters the `APPL_CONNECTED` state.

From the `APPL_CONNECTED` state, the application enters the `APPL_FMT_ALERT_LEVEL` state when an `EVT_FMT_ALERT` event is received, which is indicated by a blinking yellow LED on the STM32 Nucleo board at a rate determined by the alert level set by the central device. If the central device sets the "No Alert" option, the LED is turned off.

From the `APPL_FMT_ALERT_LEVEL` state, the application enters the `APPL_WAIT` state and waits for another alert event to occur.

Figure 5. **FMT application state machine**



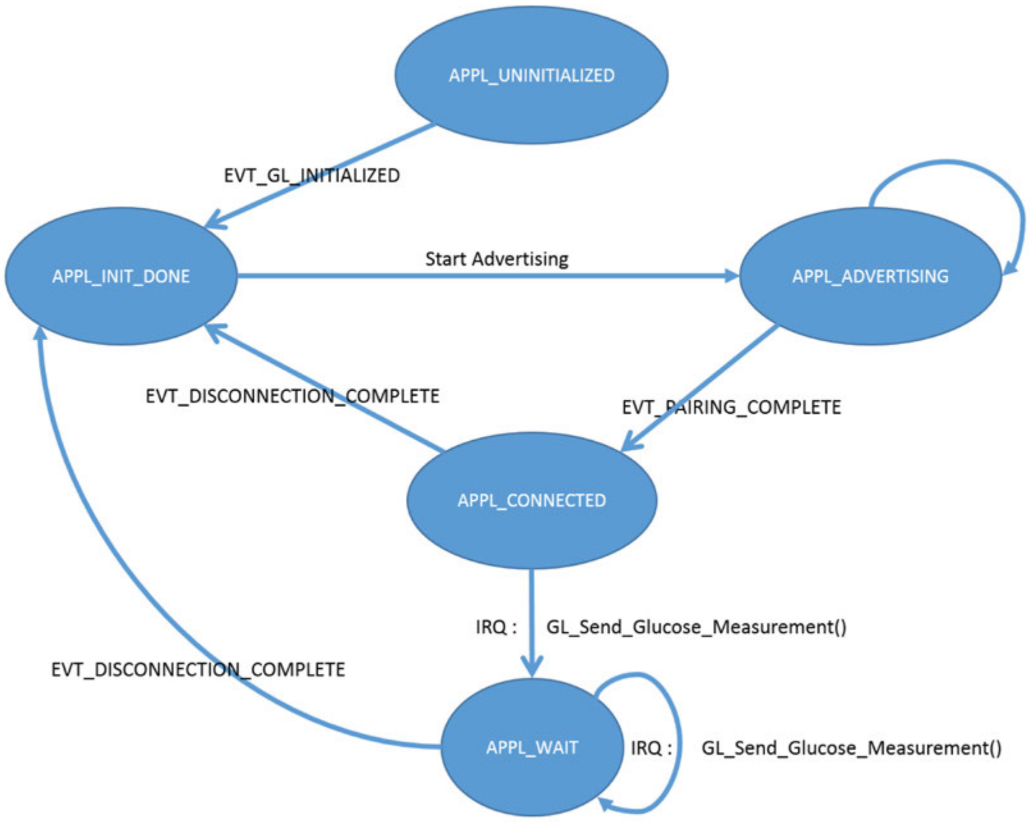
2.4.4.4 *GLM application state machine*

When the STM32 Nucleo board is powered on, the GLM application is in `APPL_UNINITIALIZED` state and initializes the hardware and the BLE Glucose profile (Section 2.4.3 Peripheral profile initialization).

Once initialization is complete, the GLM application receives the `EVT_GL_INITIALIZED` event and enters `APPL_INIT_DONE` state. The application then starts advertising its characteristics so that the central device can discover it and choose whether to connect to it, upon which the peripheral enters the `APPL_CONNECTED` state.

In the `APPL_CONNECTED` state, the Glucose measurement is sent to the central device when an interrupt (timer) is received, after which the peripheral device enters `APPL_WAIT` state and waits for the interrupt (timer).

Figure 6. **GLM application state machine**



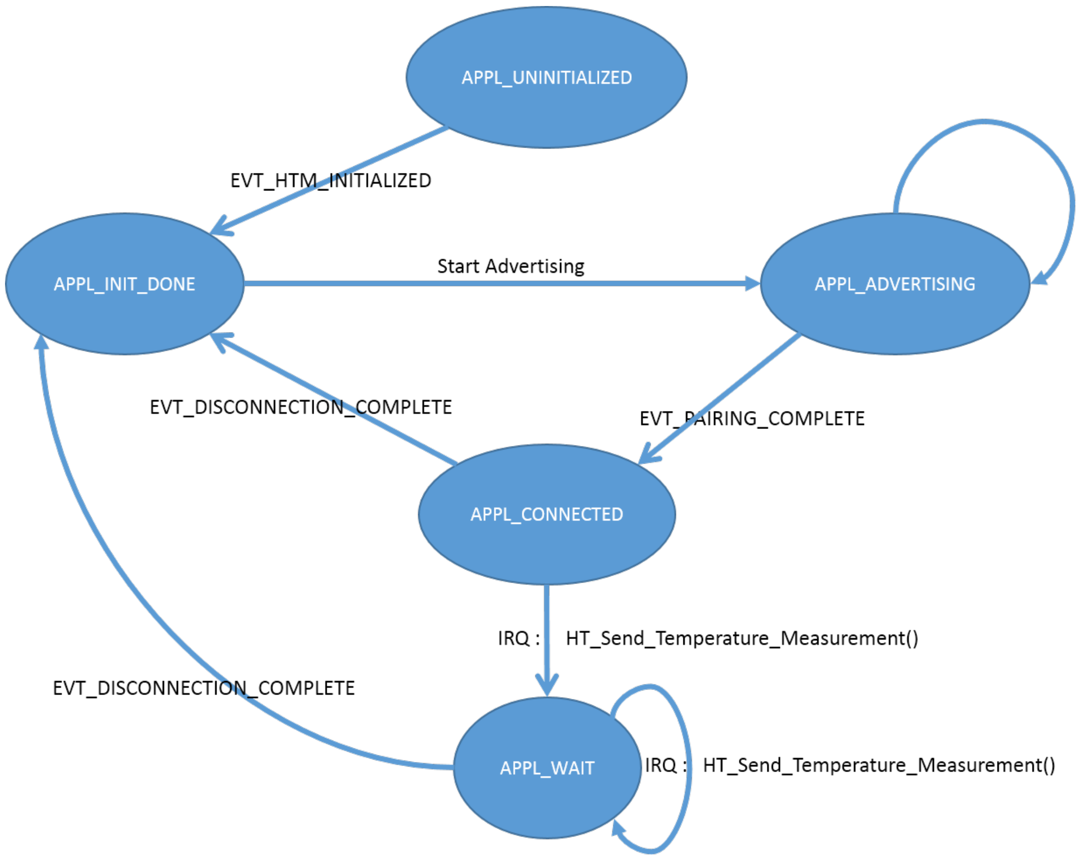
2.4.4.5 *HTM application state machine*

When the STM32 Nucleo board is powered on, the HTM application is in `APPL_UNINITIALIZED` state and initializes the hardware and the BLE Health Thermometer profile (Section 2.4.3 Peripheral profile initialization).

On completion, the HTM application receives the `EVT_HT_INITIALIZED` event and enters `APPL_INIT_DONE` state. The application then starts advertising its characteristics so that the central device can discover it and choose whether to connect to it, upon which the peripheral device enters `APPL_CONNECTED` state.

In the `APPL_CONNECTED` state, the Measurement Interval is set and the Health Thermometer measurement is sent to the central device when an interrupt (timer) is received, after which the peripheral device enters `APPL_WAIT` state and waits for the interrupt (timer).

Figure 7. **HTM application state machine**


APPL_UNINITIALIZED
EVT_HTM_INITIALIZED
APPL_INIT_DONE
Start Advertising
APPL_ADVERTISING
EVT_DISCONNECTION_COMPLETE
EVT_PAIRING_COMPLETE
APPL_CONNECTED
IRQ : HT_Send_Temperature_Measurement()
EVT_DISCONNECTION_COMPLETE
APPL_WAIT
IRQ : HT_Send_Temperature_Measurement()

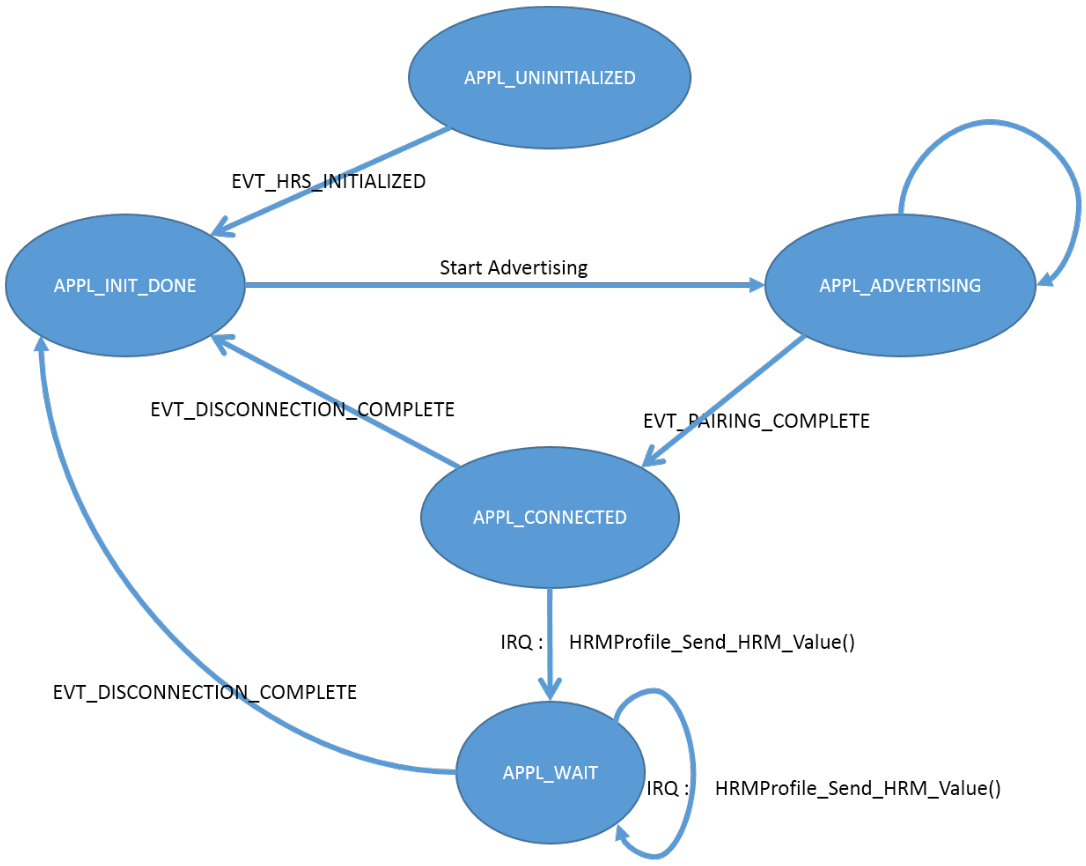
2.4.4.6 *HRM application state machine*

When the STM32 Nucleo board is turned on, the HRM application is in the `APPL_UNINITIALIZED` state, and the application initializes the hardware and the BLE heart rate profile (Section 2.4.3 Peripheral profile initialization).

Once this is complete, the HRS application receives the `EVT_HRS_INITIALIZED` event and enters the `APPL_INIT_DONE` state, and the application starts advertising its characteristics so that the central device can detect it and choose whether to connect to it, upon which the peripheral device enters the `APPL_CONNECTED` state.

In `APPL_CONNECTED` state, the Heart Rate measurement is sent to the central device when an interrupt (timer) is received, after which the peripheral device enters the `APPL_WAIT` state and waits for the interrupt (timer).

Figure 8. **HRM application state machine**



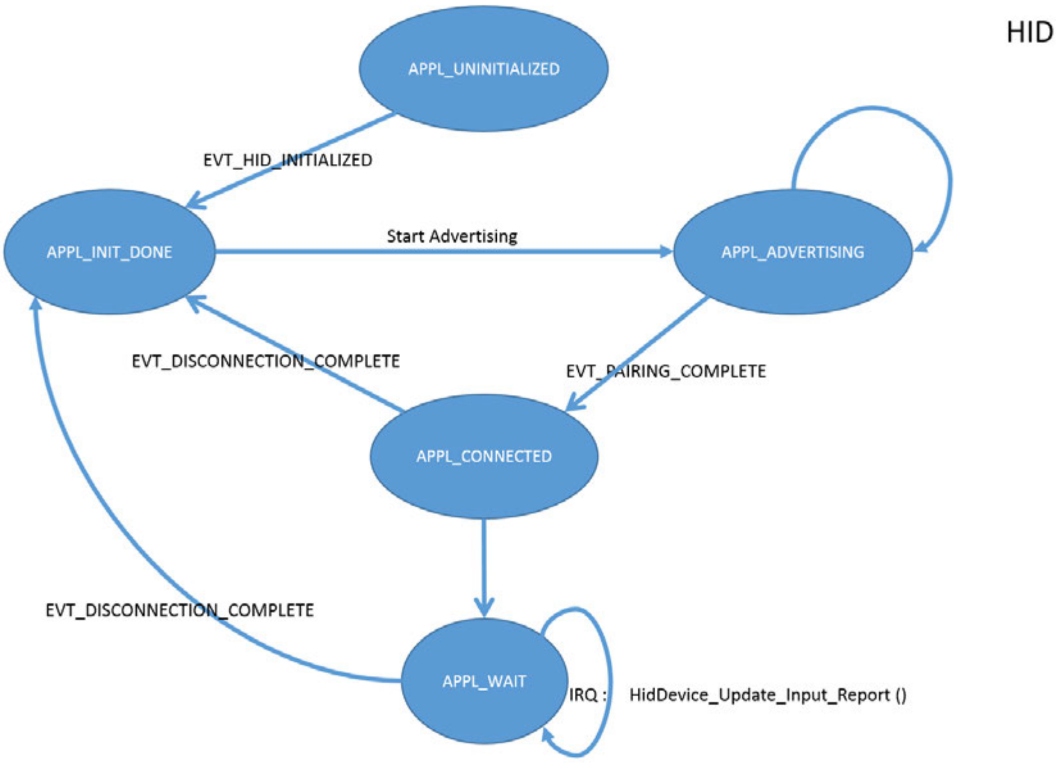
2.4.4.7 *HID application state machine*

When the STM32 Nucleo board is turned on, the HID application is in the `APPL_UNINITIALIZED` state, and the application initializes the hardware and the BLE HID profile (Section 2.4.3 Peripheral profile initialization).

Once this is complete, the HID application receives the `EVT_HID_INITIALIZED` event and enters the `APPL_INIT_DONE` state, and the application starts advertising its characteristics so that the central device can detect it and choose whether to connect to it, upon which the peripheral device enters the `APPL_CONNECTED` state and then moves to `APPL_WAIT` state.

In `APPL_WAIT` state, the sequence of "AB" characters is sent to the central each time the user button is pressed.

Figure 9. **HID application state machine**



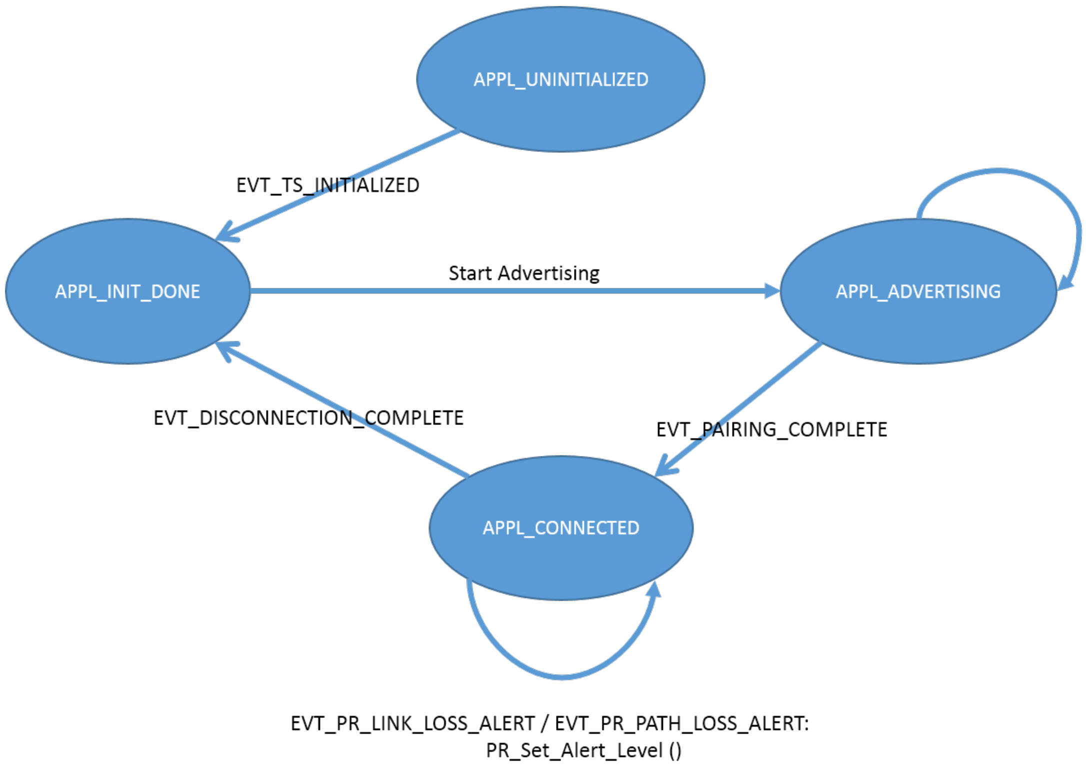
2.4.4.8 *PXP application state machine*

When the STM32 Nucleo board is turned on, the Proximity Reporter application is in `APPL_UNINITIALIZED` state. and initializes the hardware and the BLE Proximity Reporter (Section 2.4.3 Peripheral profile initialization).

Once the initialization is complete, the PXP application receives the `EVT_PR_INITIALIZED` event and enters `APPL_INIT_DONE` state. The application then starts advertising its characteristics so that the central device can detect and choose whether to connect to it, upon which the peripheral enters the `APPL_CONNECTED` state.

In the `APPL_CONNECTED` state, the application waits for an `EVT_PR_PATH_LOSS_ALERT` or an `EVT_PR_LINK_LOSS_ALERT` event to occur, which is indicated by a blinking yellow LED on the STM32 Nucleo board at a rate which is determined by the alert level set by the central device.

Figure 10. **PXP application state machine**



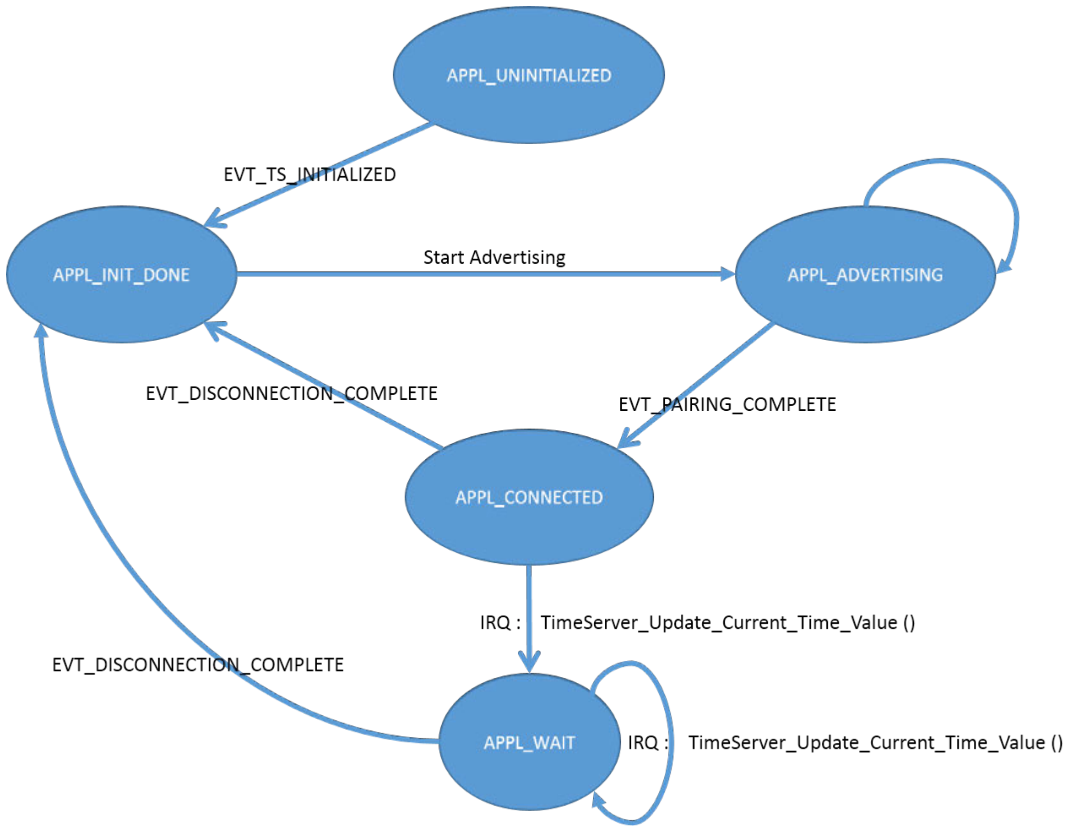
2.4.4.9 *Time server application state machine*

When the STM32 Nucleo board is turned on, the Time Server application is in `APPL_UNINITIALIZED` state and initializes the hardware and the BLE Time Server (Section 2.4.3 Peripheral profile initialization).

Once the initialization is complete, the Time Server application receives the `EVT_TS_INITIALIZED` event and enters `APPL_INIT_DONE` state. The application then starts advertising its characteristics so that the central device can detect and choose whether to connect to it, upon which the peripheral enters `APPL_CONNECTED` state.

In the `APPL_CONNECTED` state, the current time is updated and sent to the central device when an interrupt (timer) is received, after which the peripheral device enters the `APPL_WAIT` state and waits for the interrupt (timer).

Figure 11. Time server application state machine


APPL_UNINITIALIZED
EVT_TS_INITIALIZED
APPL_INIT_DONE
Start Advertising
APPL_ADVERTISING
EVT_DISCONNECTION_COMPLETE
EVT_PAIRING_COMPLETE
APPL_CONNECTED
IRQ : TimeServer_Update_Current_Time_Value ()
EVT_DISCONNECTION_COMPLETE
APPL_WAIT
IRQ : TimeServer_Update_Current_Time_Value ()

2.4.5 Peripheral profile event handling

The following events are generated by the underlying BLE stack during the lifetime of the profile application.

2.4.5.1 *Generic events*

The following generic BLE events, not specific to any profile, are used by the BLE Profiles application:

- `EVT_MP_BLUE_INITIALIZED`: sent to the application by the main profile when the controller has been initialized.
- `EVT_MP_CONNECTION_COMPLETE`: sent to the application by the main profile when a connection has been successfully established with the peer.
- `EVT_MP_PASSKEY_REQUEST`: sent to the application by the main profile when there is a request for passkey during pairing process from the controller. This event has no parameters. The application has to call the function `BLE_Profile_Send_Pass_Key()`, and pass the passkey to the controller.
- `EVT_MP_PAIRING_COMPLETE`: sent to the application by the main profile when the device is successfully paired with the peer.
- `EVT_MP_DISCONNECTION_COMPLETE`: sent to the application by the main profile to notify the result of a disconnection procedure initiated either by master or slave.
- `EVT_MP_ADVERTISING_TIMEOUT`: sent by child profiles when the limited discoverable mode times out or the profile-specific advertising timeout occurs. The application has to to restart the advertising.

2.4.5.2 *Alert notification profile events*

The alert notification profile (server role) events are specific to Alert Notification profile (server role) which uses the following one:

- `EVT_ANS_INITIALIZED`: sent to the application when the alert notification server has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed.

2.4.5.3 Blood pressure profile events

The blood pressure profile events used by the BLP application are:

- `EVT_BPS_INITIALIZED`: sent to the application when the blood pressure profile has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed due to some reason.
- `EVT_BPS_BPM_CHAR_UPDATE_CMPLT`: sent to the application whenever it has completed the characteristic update procedure to update blood pressure measurement.
- `EVT_BPS_ICP_CHAR_UPDATE_CMPLT`: sent to the application whenever it has completed the characteristic update procedure to update intermediate cuff pressure.

2.4.5.4 Find me profile events

The following find me profile (target role) events are used by the FMT application:

- `EVT_FMT_INITIALIZED`: sent to the application when the find me target has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed due to some reason.
- `EVT_FMT_ALERT`: sent to the application whenever an alert signaling request has been received from the Locator.

2.4.5.5 Glucose profile events

These events are specific to glucose profile. The following glucose profile events are used by the GLM application:

- `EVT_GL_INITIALIZED`: sent to the application when the glucose profile has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed.

2.4.5.6 Health thermometer profile events

The following health thermometer profile events are used by the HTM application:

- `EVT_HT_INITIALIZED`: sent to the application when the health thermometer profile has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed.
- `EVT_HT_TEMPERATURE_CHAR_UPDATE_CMPLT`: sent to the application whenever it has completed the characteristic update procedure to update the temperature measurement.
- `EVT_HT_INTERMEDIATE_TEMP_CHAR_UPDATE_CMPLT`: sent to the application whenever it has completed the characteristic update procedure to update the intermediate temperature measurement.
- `EVT_HT_MEASUREMENT_INTERVAL_RECEIVED`: sent to the application whenever it has started the characteristic update procedure to update the temperature measurement interval according to the value received from the collector.
- `EVT_HT_MEASUREMENT_INTERVAL_UPDATE_CMPLT`: sent to the application whenever it has completed the characteristic update procedure to update the temperature measurement interval.

2.4.5.7 Heart rate profile events

The following heart rate profile events are used by the HRM application:

- `EVT_HRS_INITIALIZED`: sent to the application when the heart rate profile has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed.
- `EVT_HRS_CHAR_UPDATE_CMPLT`: sent to the application whenever it has started the characteristic update procedure to update heart rate measurement or body sensor location.
- `EVT_HRS_RESET_ENERGY_EXPENDED`: sent to the application when the peer writes a value of 0x01 to the control point characteristic. This event has no parameters. The application has to restart accumulating the energy expended value from 0.

2.4.5.8 *Human interface device profile events*

The following human interface device profile events are used by the HID application:

- `EVT_HID_INITIALIZED`: sent to the application when the human interface device profile has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed.
- `EVT_BATT_LEVEL_READ_REQ`: this event is sent to the application when the client requests a battery level reading. On receiving this event, the application can update the battery level characteristic and then call the function Allow_BatteryLevel_Char_Read. If the process takes more than 30 minutes, the GATT channel is closed.
- `EVT_HID_CHAR_UPDATE_CMPLT`: this event is sent to the application when an update previously started by the application finishes. The status indicates whether the update was successful or not. The evt data also contains the service handle and the characteristic handle.

2.4.5.9 *Proximity profile events*

The following proximity profile (reporter role) events are used by the PXP application:

- `EVT_PR_INITIALIZED`: sent to the application when the proximity profile has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed.
- `EVT_PR_PATH_LOSS_ALERT`: sent to the application whenever an alert signaling is requested due to path loss increasing over a preset level.
- `EVT_PR_LINK_LOSS_ALERT`: sent to the application whenever an alert signaling is requested due link loss.

2.4.5.10 *Time profile events*

The following time profile (server role) events are used by the TIP application:

- `EVT_TS_INITIALIZED`: sent to the application when the time profile has completed its initialization sequence and is ready to enable the advertising or the initialization sequence failed.
- `EVT_TS_START_REFTIME_UPDATE`: sent to the application whenever it has requested to update its reference time.
- `EVT_TS_CHAR_UPDATE_CMPLT`: sent to the application whenever it has completed the characteristic update procedure to update any time server characteristic.
- `EVT_TS_CURTIME_READ_REQ`: sent to the application whenever that the current time is being requested for read.

2.4.6 Running the peripheral profile application

To run the BLE Profiles application on the STM32 Nucleo connected to a BlueNRG-MS/BlueNRG-M0 expansion board and then use it from an Android application running on a smartphone, you have to select the specific profile by unchecking the respective profile macro within the file: $BASE_DIR\\Middlewares\ST\BlueNRG-MS\profiles \Peripheral\Inc\\host_config.h

2.4.6.1 *STM32 Nucleo software setup*

Download the firmware onto the STM32 Nucleo by "drag and drop" or via the ST-LINK utility.

The folder “$BASE_DIR\\Projects\\STM32L476RG-Nucleo\\Applications\\Profiles_LowPower" contains the following subfolders:

- **Binary**, containing all the pre-built binary files;
- **EWARM**, containing the IAR project file (.eww);
- **MDK-ARM**, containing the uVision Keil project file (.uvprojx);
- **STM32CubeIDE**, containing the STM32CubeIDE project file (.project).

Note: *The subfolder files listed above are used for all the STM32 Nucleo development boards supported.*

2.4.6.2 *Starting the STM32 Nucleo and app*

Reset the STM32 Nucleo board to start running the BLE Profiles application. When the application is running, the LED on the STM32 Nucleo board starts blinking.

Note: *The application firmware should be uploaded to the board before performing this step.*

2.4.6.3 *Smartphone software setup*

For Android devices, you can retrieve the ST BLE Profiles application from the software package or download it from Google Play store (searching for "STM32 BLE Profiles").

For iOS devices, you can download the STM32 BLE Profiles app from iTunes.

For further details, see Software requirements.

2.4.6.4 *BLE Profiles application for Android*

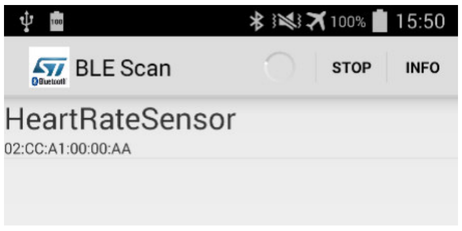
Figure 12. BLE Profiles application for Android: main page

An automatic background scan is initiated: the scan can be started/stopped by pressing the [**Start/Stop**] button.

A brief description about running the app and the link to the ST product web page can be accessed through the "Info" button.

The following screenshot shows a peripheral detected by the application (in this case, a heart rate sensor has been detected).

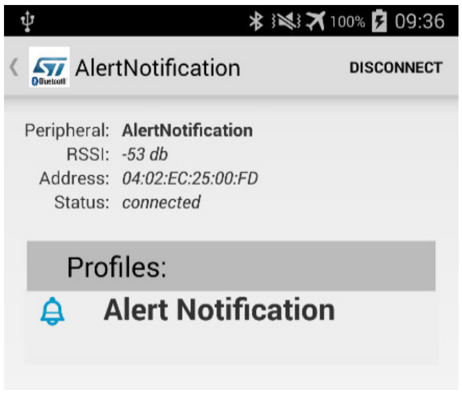
Figure 13. BLE Profiles application: detection of a heart rate sensor



2.4.6.5 *Reading ANS Data on smartphone*

The following screenshot shows the successful pairing with an Alert Notification Server (ANS).

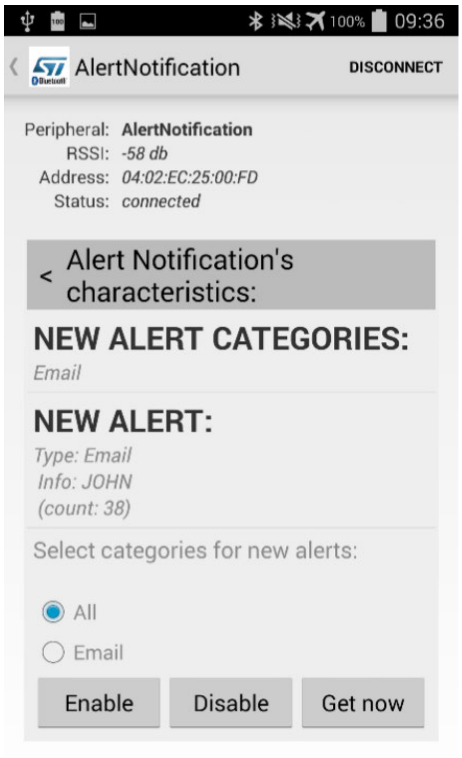
Figure 14. BLE Profiles application: successful pairing with an alert notification server



Peripheral: AlertNotification
RSSI: -53 db
Address: 04:02:EC:25:00:FD
Status: connected

To perform actions on the ANS device, select the [**Alert Notification**] service. The following figure shows the list of alert categories provided by the ANS, the alert details and the selectable options for notifications about a specific set of alert categories. In this case, the ANS provides the Email alert category.

Figure 15. BLE Profiles application: alert categories



Peripheral: AlertNotification
RSSI: -58 db
Address: 04:02:EC:25:00:FD
Status: connected

Alert Notification's characteristics:

NEW ALERT CATEGORIES:
Email

NEW ALERT:
Type: Email
Info: JOHN
(count: 38)

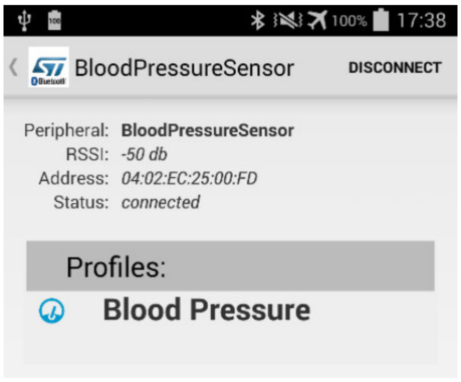
Select categories for new alerts:

[✓ All]
Email

2.4.6.6 *Reading BLP Data on smartphone*

The following figure shows the successful pairing with a blood pressure sensor.

Figure 16. BLE Profiles application paired with BLP device

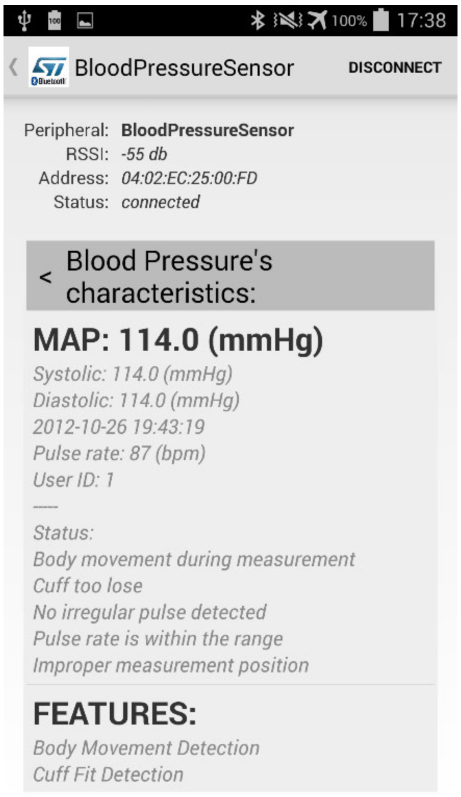


Peripheral: BloodPressureSensor
RSSI: -50 db
Address: 04:02:EC:25:00:FD
Status: connected

Profiles:
Blood Pressure

To obtain blood pressure data, select the [**Blood Pressure**] service. The following figure shows blood pressure characteristics (e.g., mean arterial pressure, systolic pressure, diastolic pressure, timestamp, pulse rate, etc.) being obtained from the BLP sensor.

Figure 17. BLE Profiles application with BLP device and being notified



Peripheral: BloodPressureSensor
RSSI: -55 db
Address: 04:02:EC:25:00:FD
Status: connected

< Blood Pressure's characteristics:

MAP: 114.0 (mmHg)
Systolic: 114.0 (mmHg)
Diastolic: 114.0 (mmHg)
2012-10-26 19:43:19
Pulse rate: 87 (bpm)
User ID: 1

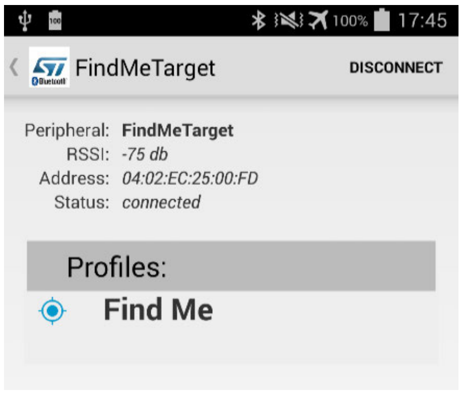
Status:
Body movement during measurement
Cuff too lose
No irregular pulse detected
Pulse rate is within the range
Improper measurement position

FEATURES:
Body Movement Detection
Cuff Fit Detection

2.4.6.7 *Reading FMT Data on smartphone*

The following figure shows the successful pairing with a Find Me Target device.

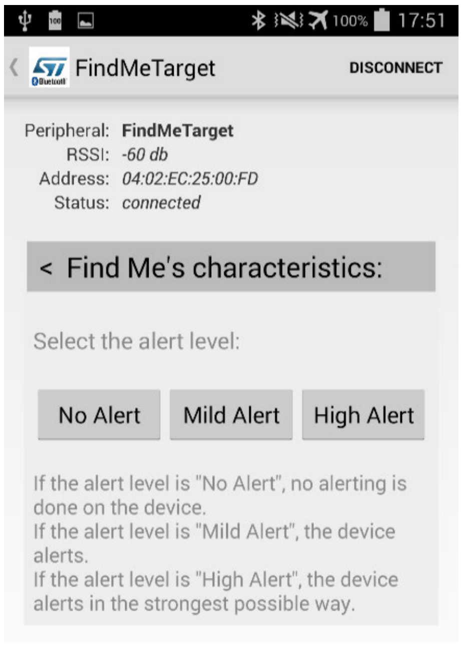
Figure 18. BLE Profiles application paired with FMT device



Peripheral: FindMeTarget
RSSI: -75 db
Address: 04:02:EC:25:00:FD
Status: connected

After selecting the [**Find Me**] service, you can generate an alert signal on the remote device by pressing the related button as shown below.

Figure 19. BLE Profiles application paired with FMT device and respective actions



Peripheral: FindMeTarget
RSSI: -60 db
Address: 04:02:EC:25:00:FD
Status: connected

< Find Me's characteristics:

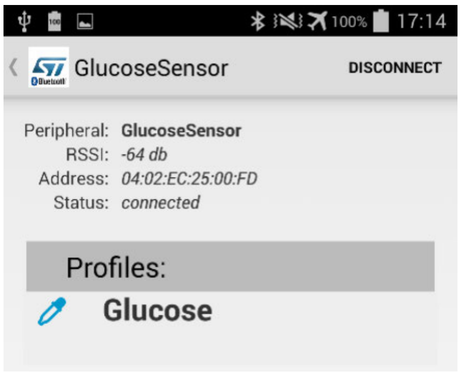
Select the alert level:

If the alert level is "No Alert", no alerting is done on the device.
If the alert level is "Mild Alert", the device alerts.
If the alert level is "High Alert", the device alerts in the strongest possible way.

2.4.6.8 *Reading GL Data on smartphone*

The following figure shows the successful pairing with a glucose sensor.

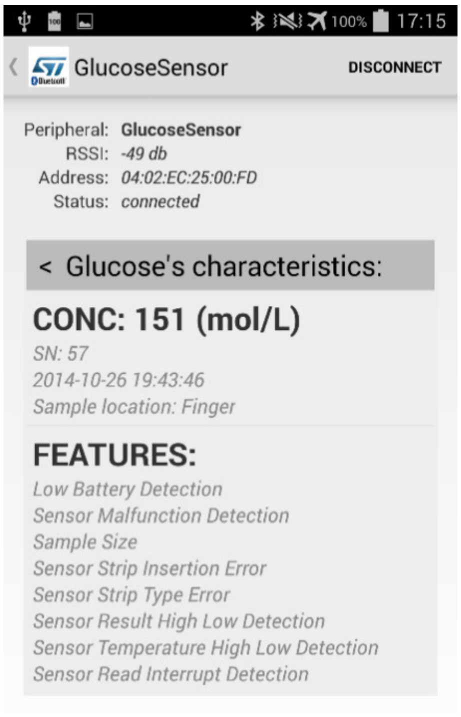
Figure 20. BLE profiles application paired with GL device



Peripheral: GlucoseSensor
RSSI: -64 db
Address: 04:02:EC:25:00:FD
Status: connected

To obtain glucose related measurements, select the [**Glucose**] service. The following figure shows glucose characteristics (e.g., glucose concentration, time-stamp, sensor location, etc.) obtained from the GL sensor.

Figure 21. BLE profiles application paired with GL device and notifications



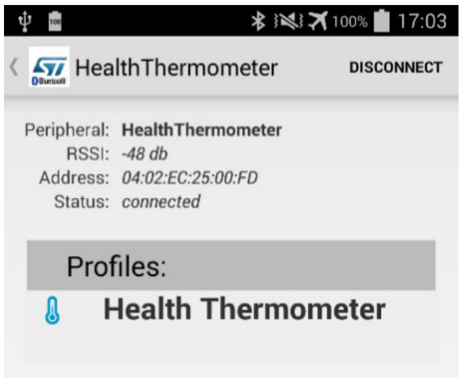
Peripheral: GlucoseSensor
RSSI: -49 db
Address: 04:02:EC:25:00:FD
Status: connected

CONC: 151 (mol/L)
SN: 57
2014-10-26 19:43:46
Sample location: Finger

FEATURES:
Low Battery Detection
Sensor Malfunction Detection
Sample Size
Sensor Strip Insertion Error
Sensor Strip Type Error
Sensor Result High Low Detection
Sensor Temperature High Low Detection
Sensor Read Interrupt Detection

2.4.6.9 *Reading HTM Data on smartphone*

The following figure shows the successful pairing with a health thermometer sensor.

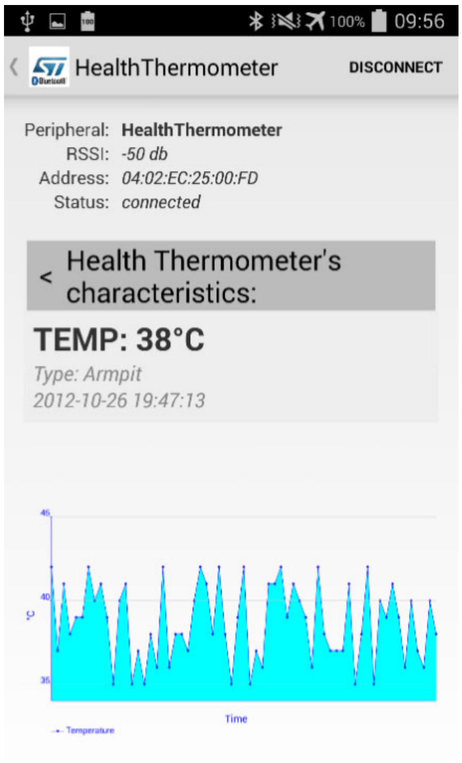


Peripheral: HealthThermometer
RSSI: -48 db
Address: 04:02:EC:25:00:FD
Status: connected

Profiles:
Health Thermometer

Figure 22. BLE Profiles application paired with an HTS device

To obtain temperature measurements, you should select [**Health Thermometer**] service. The following figure shows health thermometer characteristics being obtained from the HTM sensor. You can read the real-time temperature value and the graph displaying the latest temperature values.



Peripheral: HealthThermometer
RSSI: -50 db
Address: 04:02:EC:25:00:FD
Status: connected

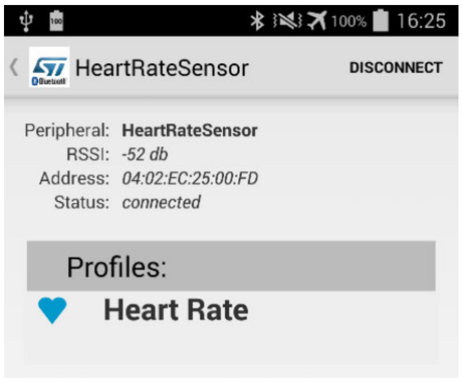
Health Thermometer's characteristics:
TEMP: 38°C
Type: Armpit
2012-10-26 19:47:13

Figure 23. BLE Profiles application paired with HTS device and notifications

2.4.6.10 *Reading HRM data on a smartphone*

Once the heart rate sensor is detected, you can select the device by tapping on it. The Android BLE pairing procedure initiates the pairing with the heart rate sensor if it is being used for the first time. The following figure shows the successful pairing process.

Figure 24. **BLE Profiles application paired with HRS device**



16:25

HeartRateSensor DISCONNECT

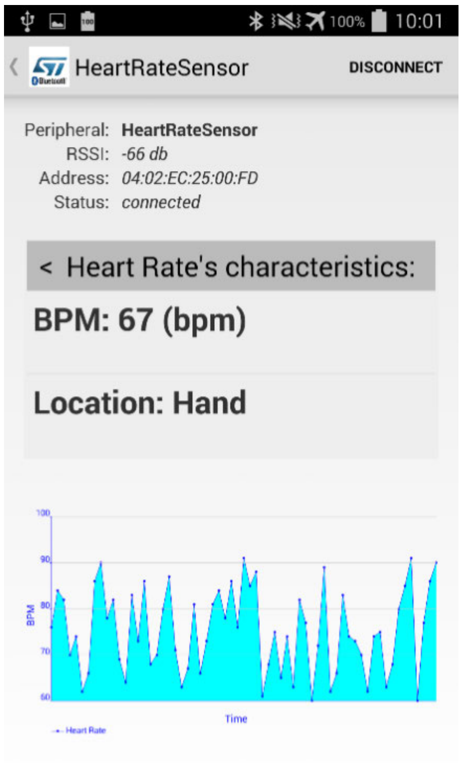
Peripheral: **HeartRateSensor**
RSSI: *-52 db*
Address: *04:02:EC:25:00:FD*
Status: *connected*

Profiles:

Heart Rate

To obtain heart rate measurements, select [**Heart Rate**] service. The following figure shows heart rate characteristics obtained from the heart rate sensor. You can read the real-time heart rate value, the sensor location and the graph displaying the latest heart rate values.

Figure 25. **BLE Profiles application paired with HRS device and notifications**



10:01

HeartRateSensor DISCONNECT

Peripheral: **HeartRateSensor**
RSSI: *-66 db*
Address: *04:02:EC:25:00:FD*
Status: *connected*

< Heart Rate's characteristics:

BPM: 67 (bpm)

Location: Hand

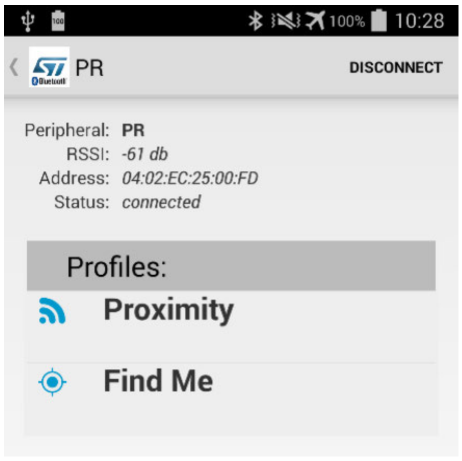
2.4.6.11 *Reading data from HID peripheral*

The behaviour of the HID profile is described in Section 2.4.1.7 Human interface device service.

2.4.6.12 *Reading PXP Data on smartphone*

The following figure shows the successful pairing with a Proximity Reporter device.

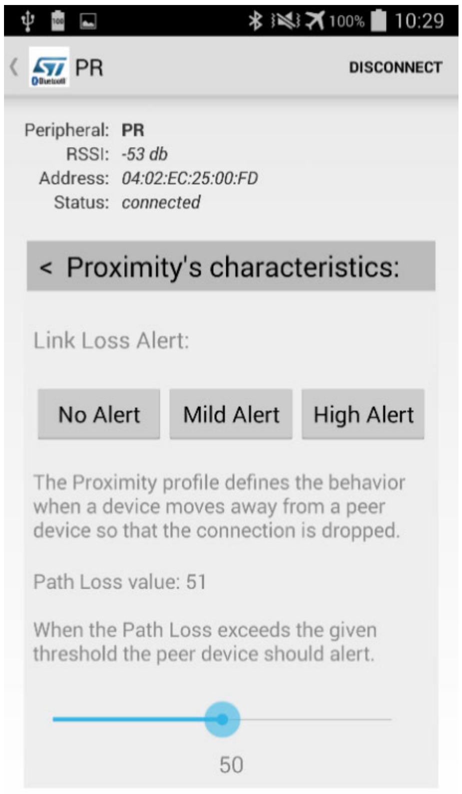
Figure 26. BLE Profiles application paired with PXR device



Peripheral: PR
RSSI: -61 db
Address: 04:02:EC:25:00:FD
Status: connected

In this case, two services are exposed: [**Find Me**] (see Section 2.4.6.7 Reading FMT Data on smartphone) and [**Proximity**]. After the latter, you can set the alert level to be generated remotely in case of disconnection or link loss. You can also set the path loss threshold: if the path loss exceeds a certain level, an alert signal is remotely generated.

Figure 27. BLE Profiles application paired with PXR device and respective actions



Peripheral: PR
RSSI: -53 db
Address: 04:02:EC:25:00:FD
Status: connected

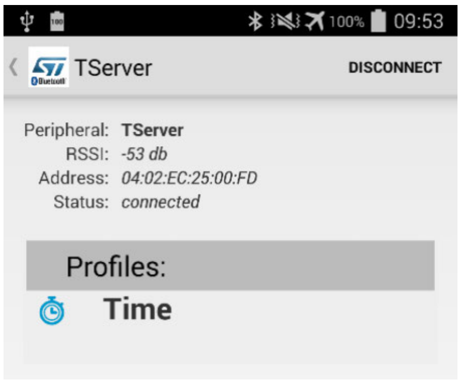
The Proximity profile defines the behavior when a device moves away from a peer device so that the connection is dropped.

Path Loss value: 51

When the Path Loss exceeds the given threshold the peer device should alert.

2.4.6.13 *Reading TS Data on smartphone*

The following figure shows the successful pairing with a Time Server device.

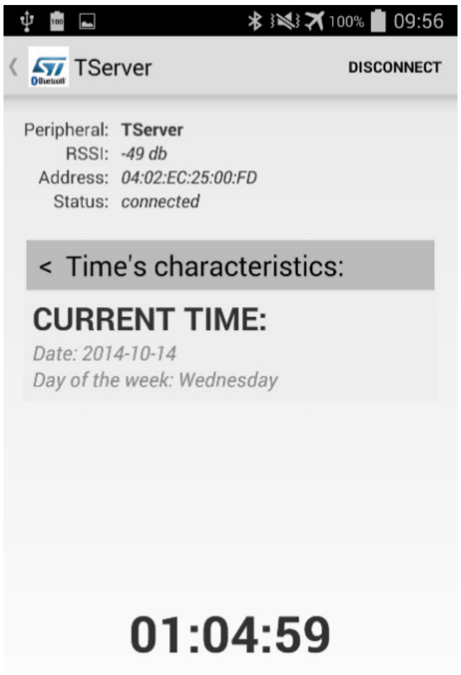

TServer DISCONNECT

Peripheral: TServer
RSSI: -53 db
Address: 04:02:EC:25:00:FD
Status: connected

Profiles:
Time

Figure 28. BLE Profiles application paired with TS device

To obtain the current time, select [**Time**] service. The following figure shows time characteristics obtained from the Time Server.


TServer DISCONNECT

Peripheral: TServer
RSSI: -49 db
Address: 04:02:EC:25:00:FD
Status: connected

< Time's characteristics:

CURRENT TIME:
Date: 2014-10-14
Day of the week: Wednesday

01:04:59

Figure 29. BLE Profiles application paired with TS device and notifications

2.4.6.14 *DMA support for BLE profile application*

The BLE Profiles application uses STM32Cube low level/low power optimizations.

You can find DMA optimization related projects and binaries in "$BASE_DIR\\Projects\\STM32L476RG-Nucleo\\Applications\\Profiles_DMA_LowPower".

The current consumption can be monitored through an amperemeter connected to JP6.

2.5 Central profile

The BLE profile central application is included in the STM32Cube firmware for X-CUBE-BLE1 and runs on the STM32 Nucleo development board when connected to an X-NUCLEO-IDB05A2 expansion board.

You can find the application-specific source code in the main.c and in each *_central_application.c file.

The $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\Applications\\Profiles_Central\\Project\\Src\\main.c performs the STM32 Nucleo board initialization, calls the `Host_Profile_Test_Application()` specific for each profile and other three main processes:

- `HCI_Process()` which is profile independent;
- `Master_Process()` that is the profile master role state machine;
- the state machine management function specific to each profile.

According to the list in Section 2.4 Peripheral role, the following files contain source code for profile context initialization:

- **Alert Notification Client** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\Applications\\Profiles_Central\\Project\\Src\\anc_central_application.c
- **Blood Pressure Collector** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\Applications\\Profiles_Central\\Project\\Src\\bp_central_application.c
- **Find Me Locator** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\Applications\\Profiles_Central\\Project\\Src\\fml_central_application.c
- **Glucose Collector** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\Applications\\Profiles_Central\\Project\\Src\\gc_central_application.c
- **Health Thermometer Collector** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\Applications\\Profiles_Central\\Project\\Src\\ht_central_application.c
- **Heart Rate Collector** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\Applications\\Profiles_Central\\Project\\Src\\hr_central_application.c
- **Time Client** $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\Applications\\Profiles_Central\\Project\\Src\\tc_central_application.c

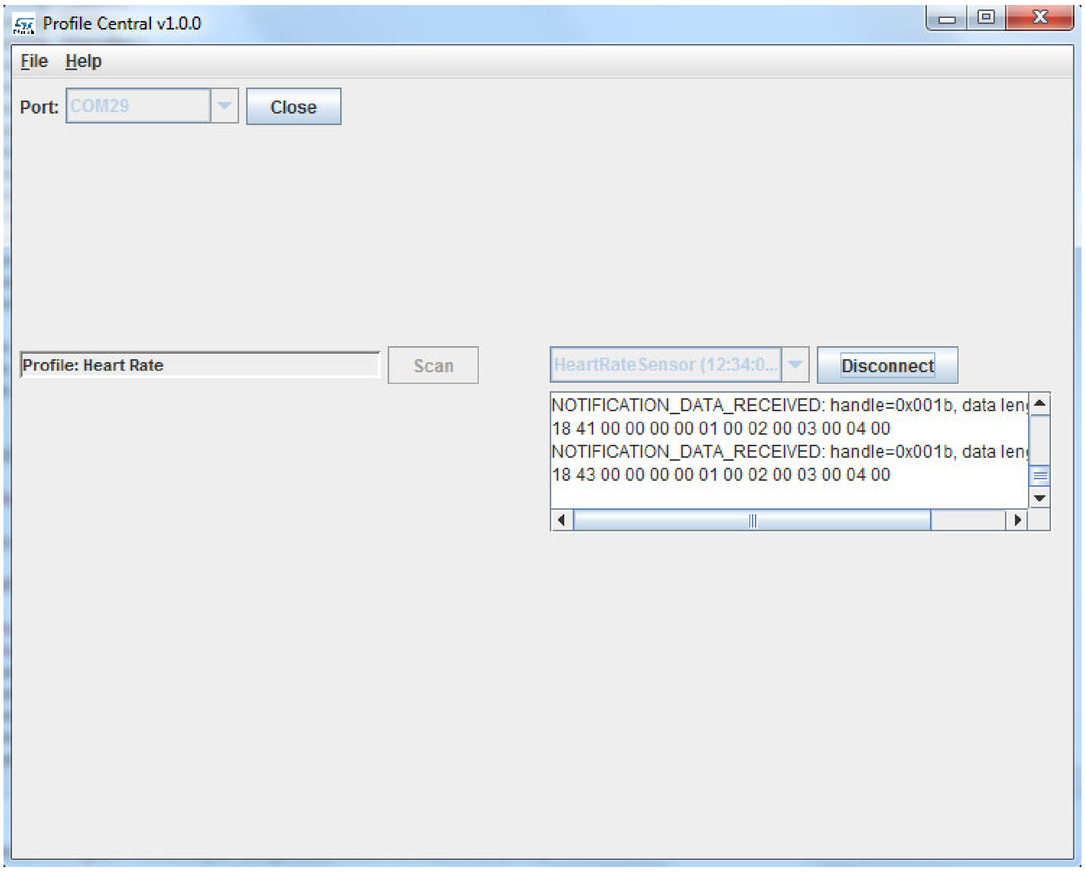
2.5.1 Running the Profile Central application

To set/change the BLE central profile to test, change the value of the macro `BLE_CURRENT_PROFILE_ROLES` (in the "active profile" section) in the $BASE_DIR\\Middlewares\ST\BlueNRG-MS\profiles\Central\Inc\\host_config.h file.

For example, if the `HEART_RATE` profile is set, once the connection between the two boards has been established, the central profile tool shows the information (services and characteristics discovered as well as HR value notifications) coming from the STM32 Nucleo board running the HR peripheral role.

The figure below shows the Java tool connected to the central device.

Figure 30. Profile Central: Java tool



```
NOTIFICATION_DATA_RECEIVED: handle=0x001b, data len
18 41 00 00 00 00 01 00 02 00 03 00 04 00
NOTIFICATION_DATA_RECEIVED: handle=0x001b, data len
18 43 00 00 00 00 01 00 02 00 03 00 04 00
```

2.6 Apple notification center service overview

The BLE_ANCS is a demo of the Apple notification center service (ANCS) which works only with the BlueNRG-MS/BlueNRG-M0 chip and shows how to configure it as a notification consumer device.

You can find the ANCS application in the folder $BASE_DIR\\Projects\\STM32L476RG-Nucleo\\Applications\\BLE_ANCS.

The ANCS profile (BLE notification consumer role) allows Bluetooth devices to easily access many kinds of notifications generated on a notification provider.

After reset, the BLE_ANCS demo puts the BlueNRG-MS/BlueNRG-M0 in advertising using "ANCSdemo" as device name and it sets the BlueNRG-MS/BlueNRG-M0 authentication requirements to enable connection. When the device is connected and linked to a notification provider, the demo configures the BlueNRG-MS/BlueNRG-M0 notification consumer device to explore the notification provider service and characteristics.

Once the setup phase has been completed, the BlueNRG-MS/BlueNRG-M0 device is configured as a notification consumer and is able to receive every notification sent by the notification provider.

3 System setup guide

3.1 Hardware description

The BLE Profiles application runs on the STM32 Nucleo boards connected to the X-NUCLEO-IDB05A2 expansion board.

The application can be easily tailored to any supported device and development board.

3.1.1 STM32 Nucleo

STM32 Nucleo development boards provide an affordable and flexible way for users to test solutions and build prototypes with any STM32 microcontroller line.

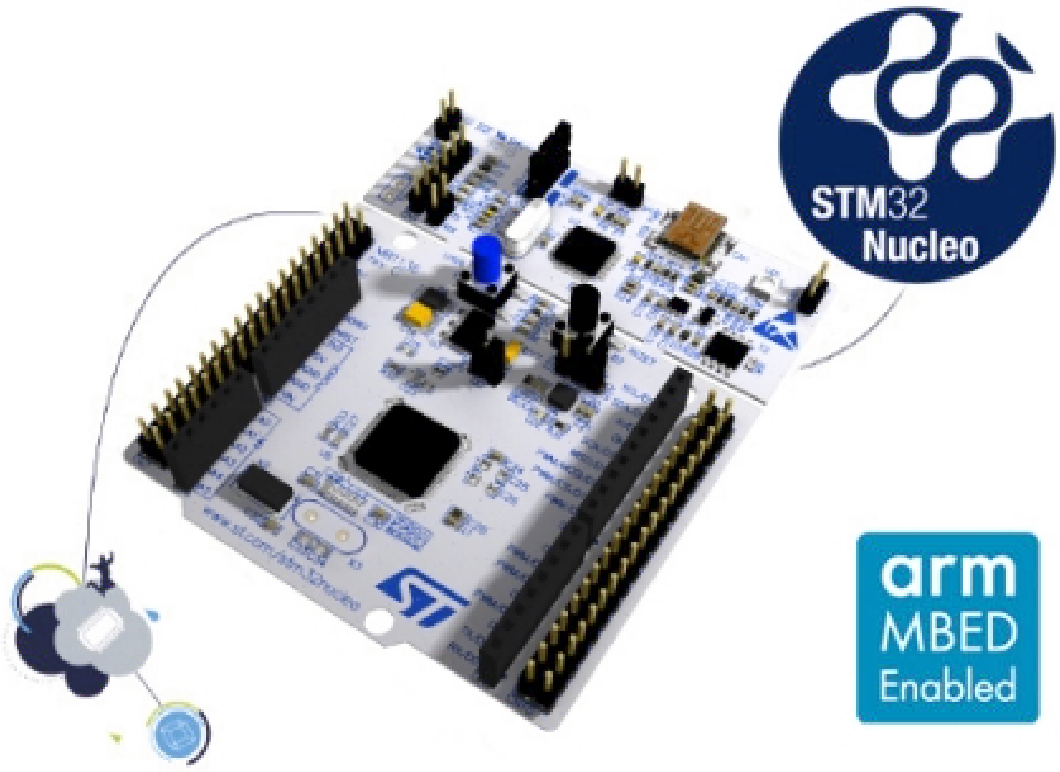
The Arduino™ connectivity support and ST morpho connectors make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide range of specialized expansion boards to choose from.

The STM32 Nucleo board does not require separate probes as it integrates the ST-LINK/V2-1 debugger/ programmer.

The STM32 Nucleo board comes with the comprehensive STM32 software HAL library together with various packaged software examples for different IDEs (IAR EWARM, Keil MDK-ARM, STM32CubeIDE, mbed and GCC/ LLVM).

All STM32 Nucleo users have free access to the mbed online resources (compiler, C/C++ SDK and developer community) at www.mbed.org to easily build complete applications.

Figure 31. **STM32 Nucleo board**



Information regarding the STM32 Nucleo board is available at www.st.com/stm32nucleo

3.1.2 X-NUCLEO-IDB05A2 expansion board

The X-NUCLEO-IDB05A2 Bluetooth low energy expansion board is based on the BlueNRG-M0 BLE network processor module.

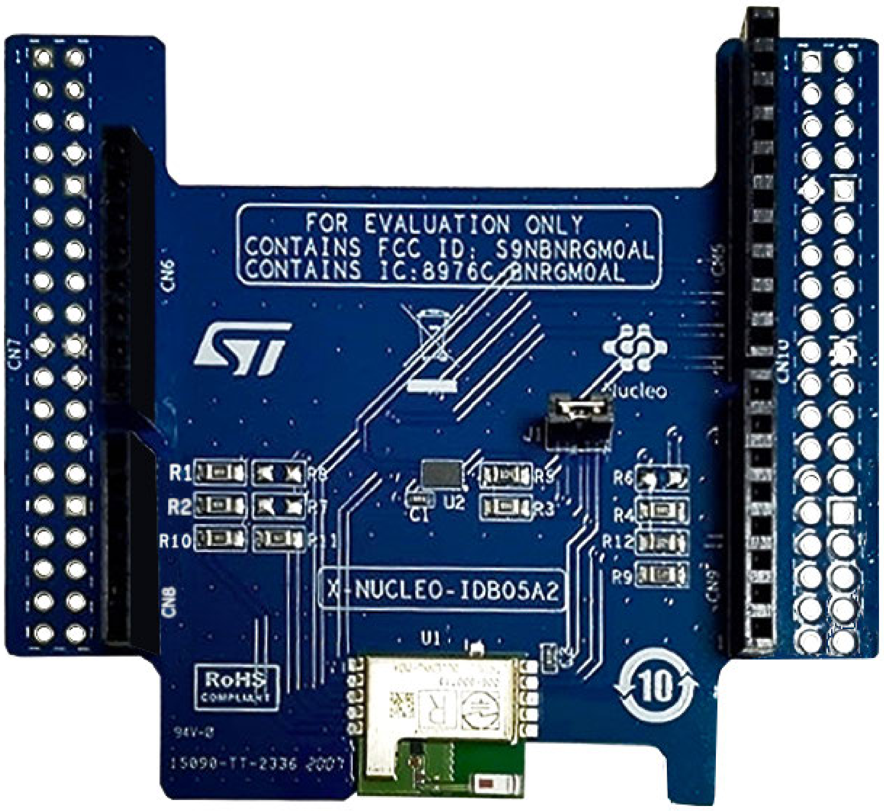
The BlueNRG-M0 is Bluetooth v4.2 compliant, FCC and IC certified (FCC ID: S9NBNRGM0AL; IC: 8976C-BNRGM0AL). It supports simultaneous master/slave roles and can behave as a Bluetooth low energy sensor and hub device at the same time.

The BlueNRG-M0 provides a complete RF platform in a tiny form factor, with integrated radio, antenna, high frequency and LPO oscillators.

The X-NUCLEO-IDB05A2 is compatible with the ST morpho (not mounted) and Arduino UNO R3 connector layout.

The X-NUCLEO-IDB05A2 interfaces with the STM32 microcontroller via the SPI pin and allows changing the default SPI clock, SPI chip select and SPI IRQ by replacing a resistor on the expansion board.

Figure 32. **X-NUCLEO-IDB05A2 expansion board**



3.1.3 Android™/iOS™ smartphone

- Android version 4.3 or above
- iOS version 8.0 or above
- Bluetooth Low Energy support

3.2 BLE Profiles app for Android/iOS

The BLE Profiles app for Android devices is available on Google Play at https://play.google.com/store/apps/details?id=com.stm.bluetoothlevalidation.

The BLE Profiles app for iOS devices is available on iTunes at https://itunes.apple.com/it/app/stm32-ble-toolbox/id1081331769?mt=8.

Revision history

Table 2. **Document revision history**

Date	Revision	Changes
18-Mar-2015	1	Initial release.
12-Sep-2016	2	Updated Section 6: "References".
20-Feb-2017	3	Updated Section 2: "BLE profile application for X-CUBE-BLE1, expansion for STM32Cube", Figure 1: "Profile application SW architecture", Section 2.2.1: "Profile command interface (PCI)"Section 2.3: "Peripheral profile application files", Section 2.4.2: "BLE profile specific initialization", Section 2.8: "Running the peripheral profile application", Section 2.8.1: "STM32 Nucleo software setup", Section 2.8.3: "Smartphone software setup", Section 5.1: "Hardware requirements", Section 5.1.2: "X-NUCLEO-IDB04A1 STM32 expansion board", Section 5.1.4: "Android™/iOS™ smartphone" and Section 5.2.1: "Software application". Added Section 2.1.7: "Human interface device service", Section 2.5.7: "HID application state machine", Section 2.7.8: "Human interface device profile events", Section 2.8.11: "Reading data from HID peripheral", Section 2.8.14: "DMA support for BLE profile application", Section 3: "Central profile application overview", Section 3.1: "Running the Profile Central application", Section 4: "Apple notification center service overview" and Section 5.1.3: "X-NUCLEO-IDB05A1 STM32 expansion board".
28-Apr-2020	4	Updated Introduction, Section 2.1 BLE Profiles application layers and Section 2.4.2 Peripheral profile application files. Added BlueNRG-M0 module and X-NUCLEO-IDB05A2 expansion board compatibility information. Text changes throughout the document.

Contents

List of tables

List of figures