
Getting started with the software package for Bluetooth low energy, sensor and NFC tag software in BLUEMICROSYSTEM3

Introduction

BLUEMICROSYSTEM3 is an expansion software package for STM32Cube. The software runs on the STM32 and includes drivers that recognize the Bluetooth low energy (BlueNRG), Dynamic NFC tag, four sensor devices (HTS221, LPS25HB, LSM6DS0, LIS3MDL) and proximity and ambient light sensing module (VL6180X). It uses the NDEF standard for writing the information for simple and secure Bluetooth pairing by storing the information on the NFC tag.

The expansion is built on STM32Cube software technology to ease portability across different STM32 microcontrollers. The software comes with sample implementations of the drivers running on the X-NUCLEO-NFC01A1, X-NUCLEO-IKS01A1, X-NUCLEO-6180XA1 and the X-NUCLEO-IDB04A1 (or X-NUCLEO-IDB05A1), when connected to a NUCLEO-F401RE or NUCLEO-L476RG.

The software is based on the STM32Cube technology.

Information regarding STM32Cube is available on www.st.com at: <http://www.st.com/stm32cube>.

Contents

1 Acronyms and abbreviations 5

2 BLUEMICROSYSTEM3 software description 6

 2.1 Overview 6

 2.2 Architecture 7

 2.3 Folder structure 8

 2.4 APIs 9

 2.5 Sample application description..... 9

 2.6 Android and iOS sample client application 15

 2.7 OPEN.MEMS Licenses activation 26

 2.7.1 OPEN.MEMS license activation using the OSX License Wizard 32

3 System setup guide..... 35

 3.1 Hardware description 35

 3.1.1 STM32 Nucleo platform..... 35

 3.1.2 X-NUCLEO-IDB04A1 expansion board..... 36

 3.1.3 X-NUCLEO-IDB05A1 expansion board..... 36

 3.1.4 X-NUCLEO-NFC01A1 expansion board 37

 3.1.5 X-NUCLEO-IKS01A1 expansion board..... 38

 3.1.6 X-NUCLEO-6180XA1 expansion board 39

 3.2 Software description..... 41

 3.3 Hardware and software setup 41

 3.3.1 Hardware setup 41

 3.3.2 Software setup..... 42

 3.3.3 System setup guide 42

4 Revision history 43

List of tables

Table 1: List of acronyms.....	5
Table 2: Document revision history	43

List of figures

Figure 1: BLUEMICROSYSTEM3 software architecture	8
Figure 2: BLUEMICROSYSTEM3 package folder structure.....	8
Figure 3: Terminal settings	9
Figure 4: Initialization phase	10
Figure 5: UART console output when the BLE services are started	11
Figure 6: UART console output when device first connects with the board	13
Figure 7: UART console output when a device is already trusted	14
Figure 8: BlueMS (Android version) start page following BLE connection	15
Figure 9: BlueMS (Android version) osxMotionFX sensor fusion page	16
Figure 10: BlueMS (Android version) calibration (1 of 2).....	17
Figure 11: BlueMS (Android version) calibration (2 of 2).....	17
Figure 12: BlueMS (Android version) example plot	18
Figure 13: BlueMS (Android version) menu selection	19
Figure 14: BlueMS (Android version) serial console (stdout/stderr)	20
Figure 15: BlueMS (Android version) debug console (stdin/stdout/stderr)	21
Figure 16: BlueMS (Android version) osxMotionAR activity recognition page	22
Figure 17: BlueMS (Android version) osxMotionCP carry position recognition page.....	23
Figure 18: BlueMS (Android version) osxMotionGR gesture recognition page.....	24
Figure 19: BlueMS (Android version) gesture recognition page.....	25
Figure 20: BlueMS (Android version) License Status page	26
Figure 21: BlueMS (Android version) Approve license page for osxMotionCP	27
Figure 22: BlueMS (Android version) license request e-mail generated for osxMotionCP	28
Figure 23: BlueMS (Android version) license upload for osxMotionCP	29
Figure 24: BlueMS (Android version) osxMotionCP license load	30
Figure 25: BlueMS (Android version) License Status page showing osxMotionCP license enabled	31
Figure 26: Licenses request tool.....	32
Figure 27: License request email generated by the OSX License Wizard	33
Figure 28: License activation email received.....	34
Figure 29: STM32 Nucleo board.....	35
Figure 30: X-NUCLEO-IDB04A1 expansion board.....	36
Figure 31: X-NUCLEO-IDB05A1 expansion board.....	37
Figure 32: X-NUCLEO-NFC01A1 M24SR64-Y dynamic NFC tag expansion board	38
Figure 33: X-NUCLEO-IKS01A1 expansion board	39
Figure 34: X-NUCLEO-6180XA1 expansion board	40
Figure 35: STM32 Nucleo + X-NUCLEO-IDB05A1 + X-NUCLEO-NFC01A1 + X-NUCLEO-IKS01A1 + X-NUCLEO-6180XA1 stack	41

1 Acronyms and abbreviations

Table 1: List of acronyms

Acronym	Description
BLE	Bluetooth low energy
NFC	Near Field Communication
NDEF	NFC Data Exchange Format

2 BLUEMICROSYSTEM3 software description

2.1 Overview

The key features of the BLUEMICROSYSTEM3 package include:

- osxMotionFX (iNEMOEngine PRO) real-time motion sensor data fusion (under OPEN.MEMS license) to combine the output from multiple MEMS sensors
- osxMotionAR (iNEMOEngine PRO) real-time activity-recognition algorithm (under OPEN.MEMS license) based only on accelerometer data
- osxMotionCP (iNEMOEngine PRO) carry position detection algorithm (under OPEN.MEMS license) based only on accelerometer data
- osxMotionGR (iNEMOEngine PRO) gesture recognition middleware (under OPEN.MEMS license) based on VL6180X proximity sensors
- Proximity-based hand gesture recognition middleware
- Complete middleware to build applications using temperature and humidity sensors (HTS221), pressure sensor (LPS25HB), motion sensors (LIS3MDL and LSM6DS0), VL6180X proximity and ambient light sensing module and M24SR64-Y Dynamic NFC/RFID tag (using the NDEF standard)
- The package is compatible with the motion sensor LSM6DS3 DIL24 expansion component
- Very low power Bluetooth low energy (BlueNRG) single-mode network processor, compliant with Bluetooth specifications core 4.1 for transmitting information to one client
- Easy portability across different MCU families, thanks to STM32Cube
- Compatible with BlueMS application for Android/iOS (Version 2.1.0 or higher) available on respective online markets (Play store/iTunes)
- Free, user-friendly license terms
- Sample implementation available on board X-NUCLEO-NFC01A1, X-NUCLEO-IKS01A1, X-NUCLEO-6180XA1 and X-NUCLEO-IDB04A1 (or X-NUCLEO-IDB05A1) when connected to NUCLEO-F401RE or NUCLEO-L476RG

This software creates four Bluetooth services:

The first service exposes all the HW features and contains the following characteristics:

- temperature
- pressure
- humidity
- lux
- proximity
- 3D gyroscope, 3D magnetometer, 3D accelerometer.

The second service exposes all the SW features and contains the following characteristics:

- Quaternions produced by osxMotionFX algorithm in short and floating precision
- The activity recognized by the osxMotionAR algorithm, among:
 - stationary
 - walking
 - fast walking
 - jogging
 - biking
 - driving
- The carried position recognized by osxMotionCP algorithm, among:
 - on desk

- in hand
- near head
- shirt pocket
- trousers pocket
- arm swing
- The gesture recognized by osxMotionGR algorithm, among:
 - wake up
 - glance
 - pick up
- The hand gesture recognized by the gesture recognition middleware, among:
 - single tap (from top to bottom)
 - swipe from left to right
 - swipe from right to left

The third service is the console service with two characteristics:

- stdin/stdout to implement bi-directional communication between client and server
- stderr to implement a mono-directional channel from the STM32 Nucleo board to an Android/iOS device

The last service is the configuration service used for communicating and controlling the status of the magneto calibration.

This package is compatible with the BlueMS Android/iOS application (Version 2.1.0 and above) available at respective Play/iTunes stores. This application can be used to display information sent with the Bluetooth low energy protocol.

2.2 Architecture

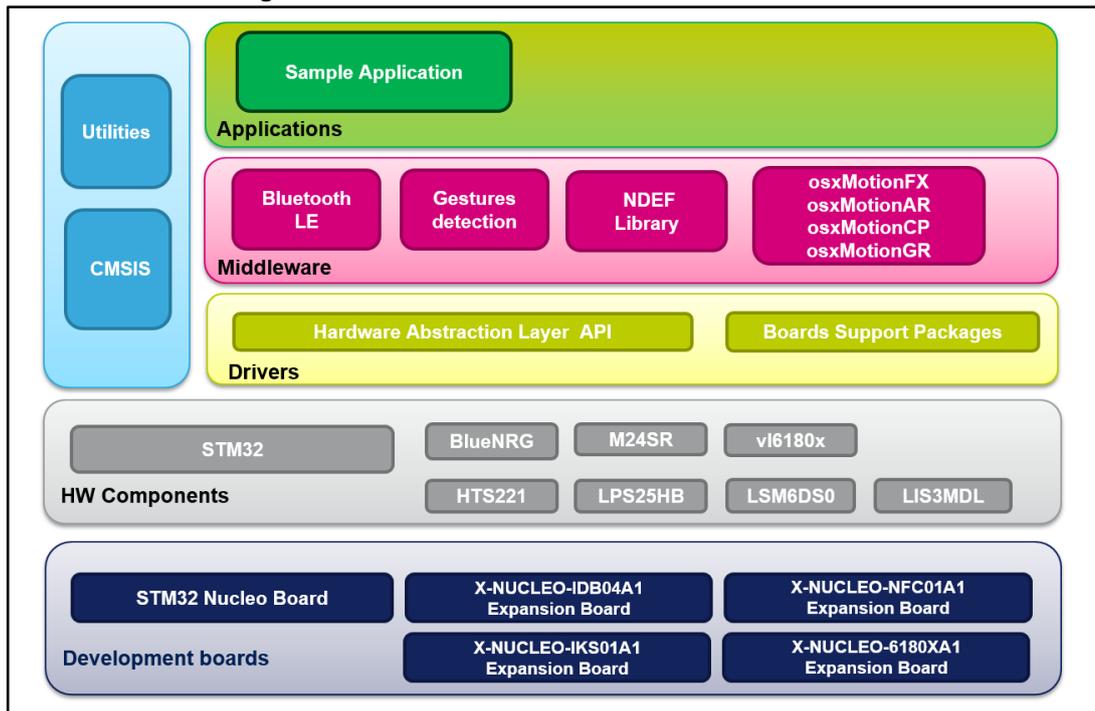
The software is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller.

The package provides a Board Support Package (BSP) for the sensor expansion board and some middleware components for serial communication with a PC.

The software layers used by the application software to access and use the sensor expansion boards are:

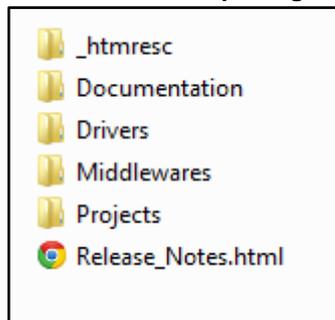
- STM32Cube HAL Layer: consists of a set of simple, generic, multi-instance APIs (application programming interfaces) which interact with the upper layer applications, libraries and stacks. These generic and extension APIs are based on a common framework which allows any layers they built on, such as the middleware layer, to implement their functions without requiring specific hardware information for a given microcontroller unit (MCU). This structure improves library code reusability and guarantees easy portability across other devices.
- Board Support Package (BSP) Layer: provides software support for the STM32 Nucleo board peripherals, excluding the MCU. These specific APIs provide a programming interface for certain board specific peripherals like LEDs, user buttons, etc and can also be used to fetch individual board version information. It also provides support for initializing, configuring and reading data.

Figure 1: BLUEMICROSYSTEM3 software architecture



2.3 Folder structure

Figure 2: BLUEMICROSYSTEM3 package folder structure



The following folders are included in the software package:

- Documentation: contains a compiled HTML file generated from the source code with software component and API details.
- Drivers: this folder contains the HAL drivers, the board specific drivers for each supported board or hardware platform, including the on-board components ones and the CMSIS layer which is a vendor-independent hardware abstraction layer for the Cortex-M processor series.
- Middlewares: this folder contains the BlueNRG Bluetooth low energy protocols and the libraries for NDEF, osxMotionFX sensor fusion, for the osxMotionAR activity recognition algorithm, for the osxMotionCP carry position recognition algorithm, and for the gesture recognition algorithm.
- Projects: this folder contains a sample application used for transmitting the sensors values and the output of osxMotionFX, osxMotionAR, osxMotionCP and gesture recognition algorithms by using the Bluetooth low energy protocol, provided for the NUCLEO-F401RE/NUCLEO-L476RG platforms with three development environments

(IAR Embedded Workbench for ARM, RealView Microcontroller Development Kit (MDK-ARM), System Workbench for STM32).

2.4 APIs

Detailed technical information about the APIs available to the user can be found in a compiled HTML file located inside the “Documentation” folder of the software package, where all the functions and parameters are fully described.

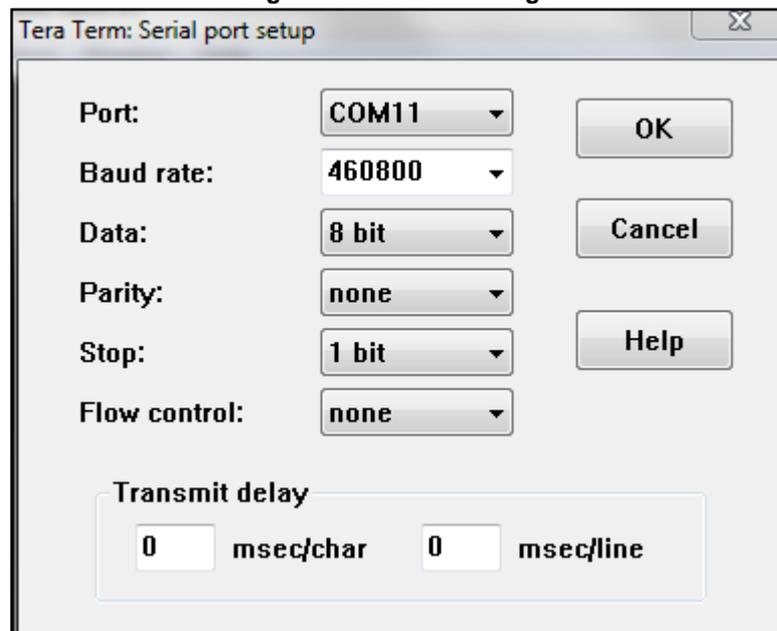
2.5 Sample application description

A sample application using the X-NUCLEO-NFC01A1, X-NUCLEO-IKS01A1, X-NUCLEO-6180XA1 and X-NUCLEO-IDB04A1 (or X-NUCLEO-IDB05A1) expansion boards with the NUCLEO-F401RE or NUCLEO-L476RG board is provided in the “Projects” directory. Ready to be built projects are available for multiple IDEs.

The user can control all application behavior via UART by launching a terminal application and setting the UART port to:

- Baud rate = 460800
- Data = 8 bit
- Parity = none
- Stop = 1 bit.

Figure 3: Terminal settings



Initially, when the user presses the reset button, the application begins initializing the UART and the I²C interfaces and, using the NDEF standard, writes the URI www.st.com/stm32ode on the M24SR dynamic NFC tag on the X-NUCLEO-NFC01A1 expansion board (see [Figure 4: "Initialization phase"](#)). So, when an Android device reads the content of the NFC tag, the browser automatically launches and tries to connect to this URI.

Figure 4: Initialization phase

```

COM12:460800baud - Tera Term VT
File Edit Setup Control Window Help
UART Initialized
I2C Initialized

STMicroelectronics BlueMicrosystem3:
STM32F401RE-Nucleo board (HAL 1.4.0_0)
Compiled Nov 24 2015 08:36:17 (IAR)
Send Every 30mS 3 Short precision Quaternions
Send Every 20mS 1 Float precision Quaternions
Send Every 500mS Temperature/Humidity/Pressure/Lux
Send Every 100mS Acc/Gyro/Magneto
Send Every 50mS Proximity

Press the User button for starting the BlueNRG
M24SR Initialized
M24SR URI written = "www.st.com/bluemicrosystem"

```

When the user presses the blue user button (see [Figure 5: "UART console output when the BLE services are started"](#)), the program:

- initializes the SPI interface used for communicating with the BlueNRG expansion board
- identifies which BlueNRG expansion board (X-NUCLEO-IDB04A1 or X-NUCLEO-IDB05A1) is connected to the STM32 Nucleo and the corresponding HW and FW version
- creates a random BLE MAC address and PIN necessary for establishing the connection
- initializes the BLE Console service, adding the stdin/stdout and stderr characteristics
- changes the content of the M24SR dynamic NFC tag (always using the NDEF standard) for writing all the necessary information to automatically run the BlueMS Android application (application name, BLE advertise data, BLE MAC address and BLE connection PIN).
- Checks whether the osxMotionFX, osxMotionAR, osxMotionCP and osxMotionGR library are initialized properly, with valid licenses.
- initializes the sensors on the 6180X expansion board, including satellites.

Therefore, when an Android device with the BlueMS application reads the NFC content, it can automatically launch the BlueMS application and form a connection with the STM32 Nucleo board, without any manual intervention.

iOS does not allow using NFC for this purpose, so you need to manually launch the application, scan for the STM32 Nucleo board and add the connection PIN.

There is a define called `OSX_BMS_SECURE_CONNECTION` in the `Projects\Multi\Applications\BLUEMICROSYSTEM3\Inc\osx_bms_config.h` file that can control whether the STM32 Nucleo board only accepts secure connections (default setting) or accepts any connection (define is commented out). In the latter case, you do not need a BLE connection PIN to connect a device to the STM32 Nucleo board.

Figure 5: UART console output when the BLE services are started

```

COM12:460800baud - Tera Term VT
File Edit Setup Control Window Help
UART Initialized
I2C Initialized

STMicroelectronics BlueMicrosystem3:
  STM32F401RE-Nucleo board (HAL 1.4.0_0)
  Compiled Nov 24 2015 08:36:17 (IAR)
  Send Every 30mS 3 Short precision Quaternions
  Send Every 20mS 1 Float precision Quaternions
  Send Every 500mS Temperature/Humidity/Pressure/Lux
  Send Every 100mS Acc/Gyro/Magneto
  Send Every 50mS Proximity
Press the User button for starting the BlueNRG
M24SR Initialized
M24SR URI written ="www.st.com/bluemicrosystem"
M24SR Bluetooth NDEF Table written
Debug Connection Enabled
Debug Notify Transmission Enabled
UL6180x_Prepare ok device=0
Device S0 Ready
SPI Initialized
EUT_Vendor =1
SERVER: BLE Stack Initialized
  Board type=IDB05A1 HWver=49, FWver=1811
  BoardName= BlueMS3
  BoardMAC = 4a:59:32:5f:0:3f
  Pin=000257

  Only Secure connection allowed

HW Service W2ST added successfully
SW Service W2ST added successfully
Console Service W2ST added successfully
Config Service W2ST added successfully
OK Temperature Sensor
OK Pressure Sensor
OK 6 AxesSensor
OK Magneto Sensor
DS3 on board
Enabled ST osxMotionFX x9/x6 v1.0.7
Enabled ST osxMotionAR v1.0.1
Enabled ST osxMotionCP v1.0.0
Calibration Not present in RAM
EUT_Vendor =1

```

As displayed in the console output, the application sends:

- every 30 ms - three quaternions in short precision
- every 500 ms - the temperature/humidity/pressure and lux values
- every 100 ms - the 3D accelerometer, 3D gyroscope and 3D magnetometer values
- every 50 ms - the proximity value
- every 10 ms - the gesture code result of the gesture recognition

This application reads the sensor data values from the accelerometer, magnetometer and gyroscope at a frequency of 100 samples/s. The osxMotionFX (iNEMOEngine PRO) library combines these data sensor values to produce quaternions at the same 100 quaternions/s rate, which are transmitted to the client via the Bluetooth low energy protocol to display real motion data using a vendor-specific BLE service. There are two definitions in the osx_bms_config.h file which control how many quaternions the application sends to the Bluetooth client:

- `QUAT_UPDATE_MUL_10MS`: defines the transmission rate for each set of quaternions by multiple of 10 ms.
- `SEND_N_QUATERNIONS`: defines how many quaternions are sent in each Bluetooth packet.

By default, the application sends three quaternions every 30 ms. In the same `osx_bms_config.h` file, there are the following other defines:

- `ENV_UPDATE_MUL_10MS` - transmission rate for temperature/pressure and humidity/lux
- `ACC_GYRO_MAG_UPDATE_MUL_10MS` - transmission rate for accelerometer/gyroscope and magnetometer values
- `OSX_BMS_DEBUG_CONNECTION` and `OSX_BMS_DEBUG_NOTIFY_TRANSMISSION` - to enable some debugging information for BLE communication
- `OSX_BMS_MOTIONAR` - to enable the `osxMotionAR` activity recognition algorithm
- `OSX_BMS_MOTIONCP` - to enable the `osxMotionCP` carry position recognition algorithm
- `OSX_BMS_MOTIONGR` - to enable the `osxMotionGR` motion-based gesture recognition algorithm
- `OSX_BMS_GESTURE` - to enable the proximity-based gesture recognition algorithm

The `osxMotionFX` (iNEMOEngine PRO) library has an auto-calibrating procedure, with the calibration status transmitted by BLE to the client. By pressing the user button on the NUCLEO-F401RE (or NUCLEO-L476RG) board, you can reset the library calibration status to force a new auto-calibration procedure.

The `osxMotionAR` (iNEMOEngine PRO) library can recognize the following activities:

- stationary
- walking
- fast walking
- jogging
- biking
- driving

The `osxMotionAR` can be enabled by defining the `OSX_BMS_MOTIONAR` define in `osx_bms_config.h`.

The `osxMotionCP` (iNEMOEngine PRO) library recognizes and provides real-time information about how the user is carrying the board; i.e., the phone carry position:

- on desk
- in hand
- near head
- shirt pocket
- trouser pocket
- arm swing

The `osxMotionCP` can be enabled by defining the `OSX_BMS_MOTIONCP` define in `osx_bms_config.h`.

The `osxMotionGR` (iNEMOEngine PRO) library can recognize the following gestures:

- pick up – indicating the raising/lifting of the board from a table
- glance – corresponding to the rotation of the board of approximately 30° (simulating the gesture of rotating a phone to glance at it)
- wakeup – a shaking action

The `osxMotionGR` can be enabled by defining the `OSX_BMS_MOTIONGR` define in `osx_bms_config.h`.

The proximity-based gesture recognition can be enabled by defining the `OSX_BMS_GESTURE` define in `osx_bms_config`. The algorithm is able to recognize single

tap (moving hand from top to bottom) and bi-directional swipes (moving hand from left to right and right to left).

When an Android/iOS device wants to connect to the STM32 Nucleo board, it starts the secure pairing procedure and sends ping information to the stdout console BLE characteristics (see [Figure 6: "UART console output when device first connects with the board"](#)).

Figure 6: UART console output when device first connects with the board

```

COM12:460800baud - Tera Term VT
File Edit Setup Control Window Help
Debug Connection Enabled
Debug Notify Trasmission Enabled
UL6180x_Prepate ok device=0
Device S0 Ready
SPI Initialized
EUT_Vendor =1
SERUER: BLE Stack Initialized
      Board type=IDB05A1 HWver=49, FWver=1811
      BoardName= BlueMS3
      BoardMAC = d9:5b:45:47:11:73
      Pin=000271

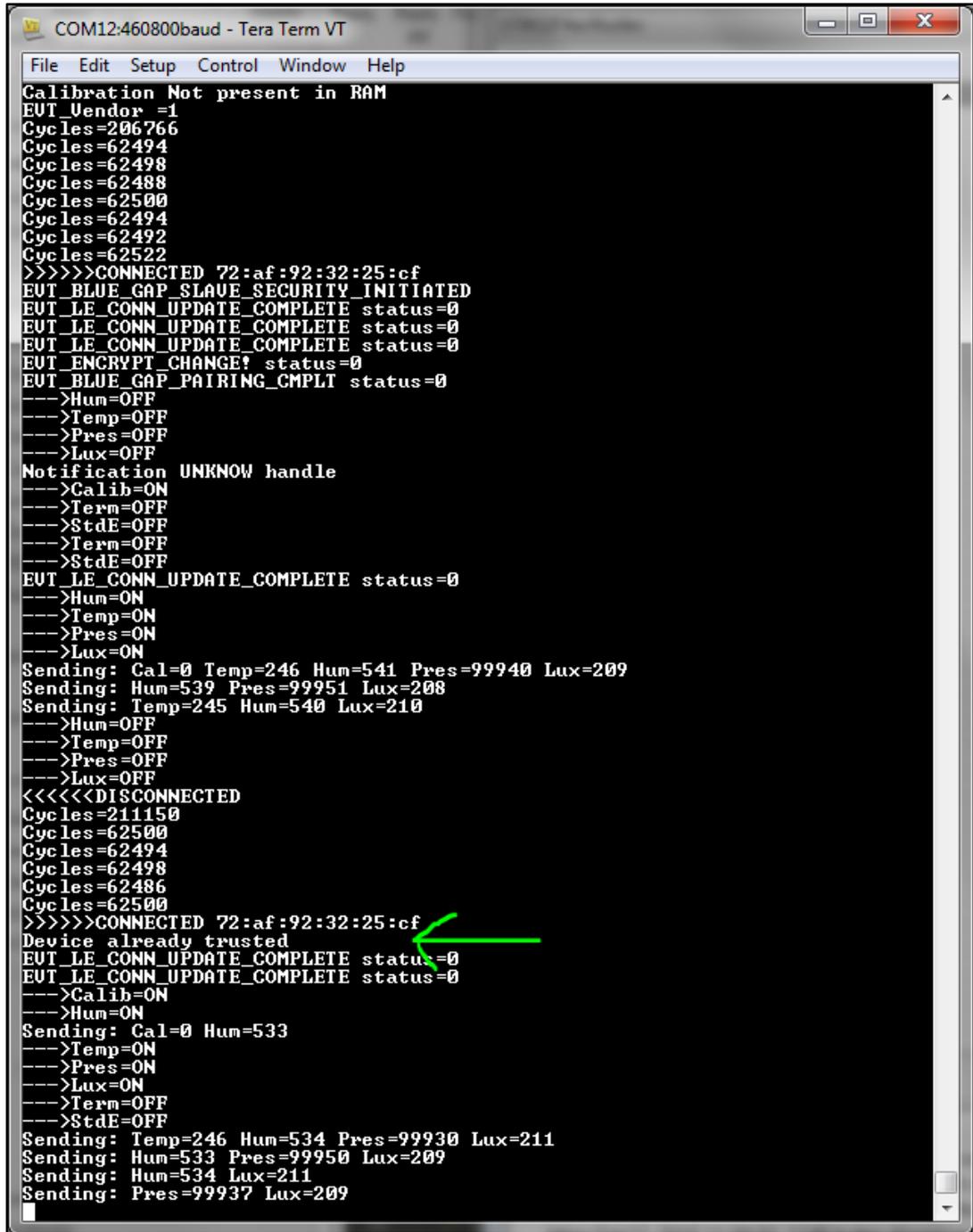
      Only Secure connection allowed

HW      Service W2ST added successfully
SW      Service W2ST added successfully
Console Service W2ST added successfully
Config Service W2ST added successfully
OK Temperature Sensor
OK Pressure Sensor
OK 6 AxesSensor
OK Magneto Sensor
DS3 on board
Enabled ST osxMotionFX x9/x6 v1.0.7
Enabled ST osxMotionAR v1.0.1
Enabled ST osxMotionCPR v1.0.0
Calibration Not present in RAM
EUT_Vendor =1
Cycles=206766
Cycles=62494
Cycles=62498
Cycles=62488
Cycles=62500
Cycles=62494
Cycles=62492
Cycles=62522
>>>>>CONNECTED 72:af:92:32:25:cf
EUT_BLUE_GAP_SLAVE_SECURITY_INITIATED
EUT_LE_CONN_UPDATE_COMPLETE status=0
EUT_LE_CONN_UPDATE_COMPLETE status=0
EUT_LE_CONN_UPDATE_COMPLETE status=0
EUT_ENCRYPT_CHANGE! status=0
EUT_BLUE_GAP_PAIRING_CMPLT status=0
---->Hum=OFF
---->Temp=OFF
---->Pres=OFF
---->Lux=OFF
Notification UNKNOW handle
---->Calib=ON
---->Term=OFF
---->StdE=OFF
---->Term=OFF
---->StdE=OFF
EUT_LE_CONN_UPDATE_COMPLETE status=0
---->Hum=ON
---->Temp=ON
---->Pres=ON
---->Lux=ON
Sending: Cal=0 Temp=246 Hum=541 Pres=99940 Lux=209
Sending: Hum=539 Pres=99951 Lux=208
Sending: Temp=245 Hum=540 Lux=210
---->Hum=OFF
---->Temp=OFF

```

The application has a whitelist of one element, so the next time that the same device wants to make a connection, it is not necessary to trust it again (see [Figure 7: "UART console output when a device is already trusted"](#)).

Figure 7: UART console output when a device is already trusted

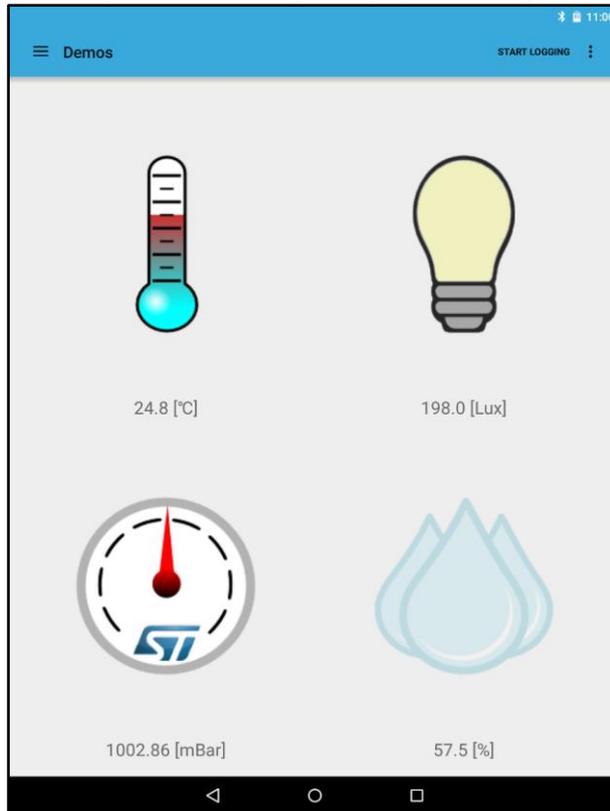


2.6 Android and iOS sample client application

The BLUEMICROSYSTEM3 software for STM32Cube is compatible with the BlueMS Android/iOS applications (Version 2.1.0 or above) available at the respective Android Play/Apple iOS stores. This section illustrates how the Android application works.

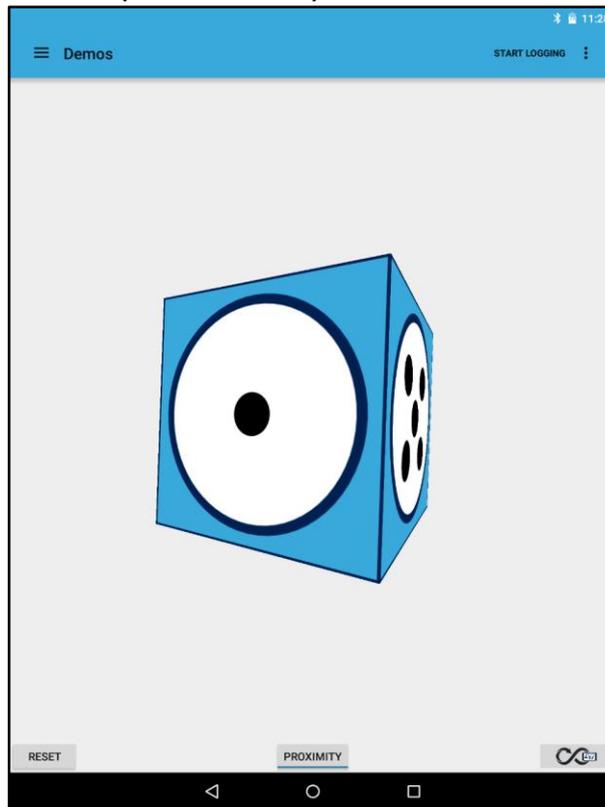
After connection, BlueMS opens the main page ([Figure 8: "BlueMS \(Android version\) start page following BLE connection"](#)) which displays the temperature, luminosity, pressure and humidity values.

Figure 8: BlueMS (Android version) start page following BLE connection



The following page ([Figure 9: "BlueMS \(Android version\) osxMotionFX sensor fusion page"](#)) shows the output of the osxMotionFX sensor fusion library with a cube that rotates according to board movements.

Figure 9: BlueMS (Android version) osxMotionFX sensor fusion page



There are three buttons on this page:

- The central button enables or disables the proximity sensor, which triggers the cube zooming out/in as a function of the proximity measured by the X-NUCLEO-6180XA1 expansion board.
- The left button resets the cube position
- The right button shows the calibration status of the osxMotionFX library (black/green for not calibrated/calibrated). When pressed, it forces a fresh magnetometer calibration.

When the left or right button is pressed, the application shows a pop-up window advising how to position the board for correct cube rotation and how to move the board to facilitate calibration (See *BlueMSca_1* and *Figure 11: "BlueMS (Android version) calibration (2 of 2)"*)

Figure 10: BlueMS (Android version) calibration (1 of 2)

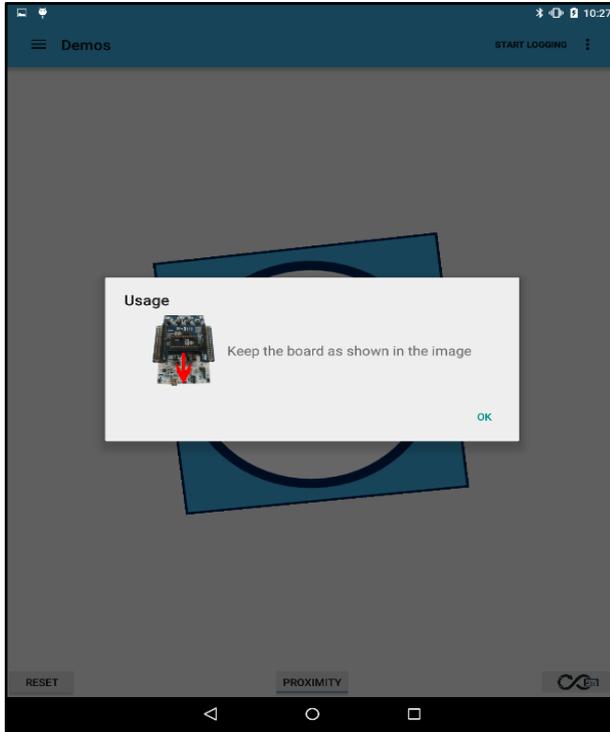
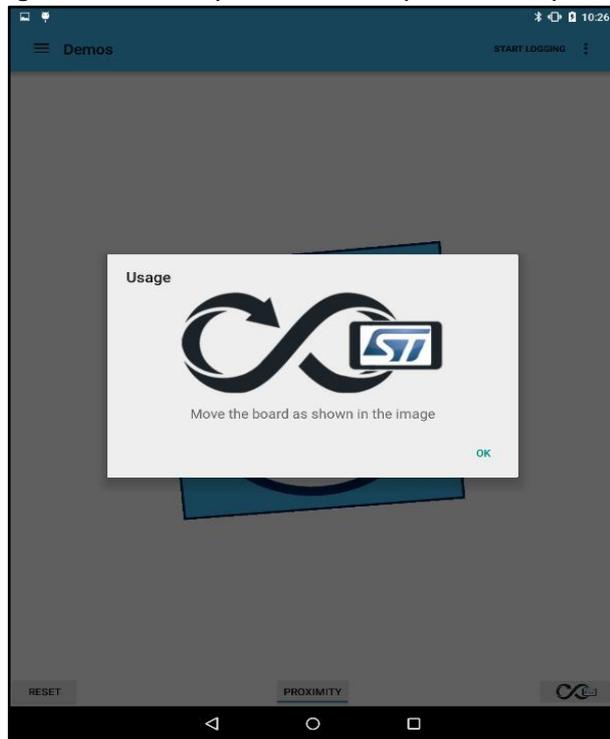
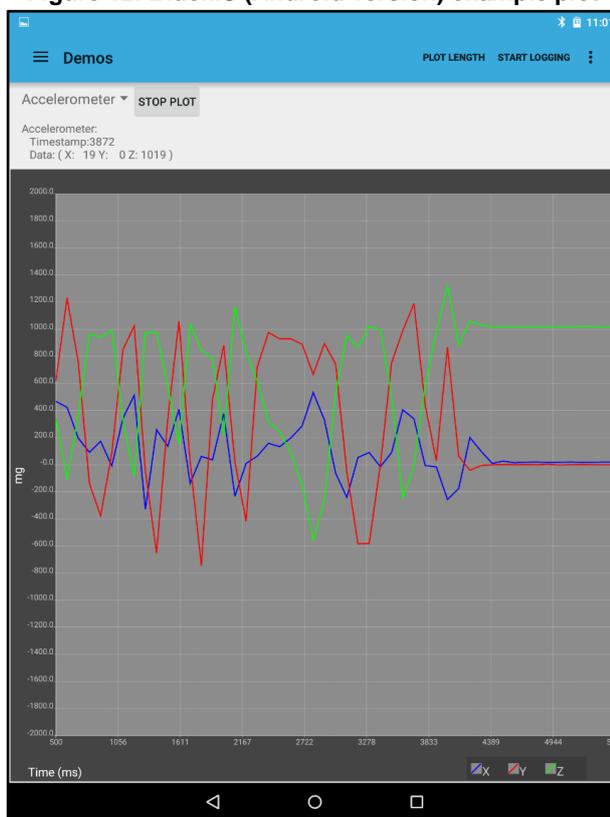


Figure 11: BlueMS (Android version) calibration (2 of 2)



Moving to the next page ([Figure 12: "BlueMS \(Android version\) example plot"](#)), you can plot data read from the sensor expansion boards.

Figure 12: BlueMS (Android version) example plot



By selecting the option menu ([Figure 13: "BlueMS \(Android version\) menu selection"](#)) you can enable the serial console (to display the stdout/stderr) or the debug console (to also display the stdin). If you write something to the debug console, the board will return the same test message ([Figure 14: "BlueMS \(Android version\) serial console \(stdout/stderr\)"](#) and [BlueMS_debug](#)).

Figure 13: BlueMS (Android version) menu selection

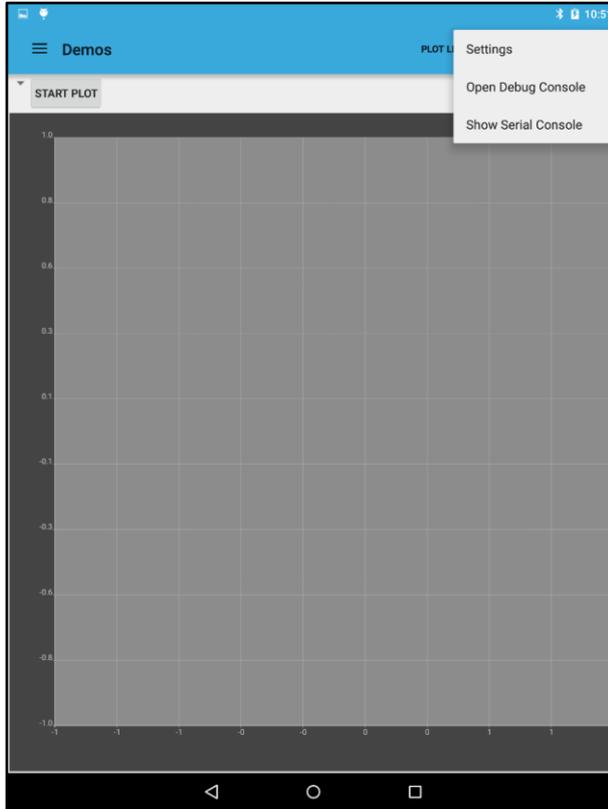


Figure 14: BlueMS (Android version) serial console (stdout/stderr)

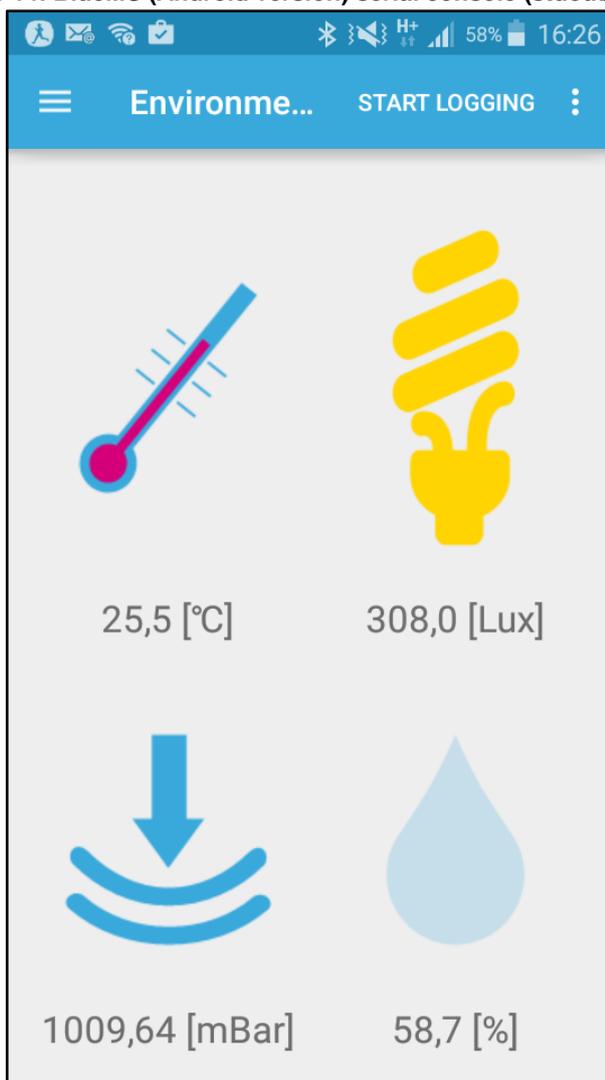
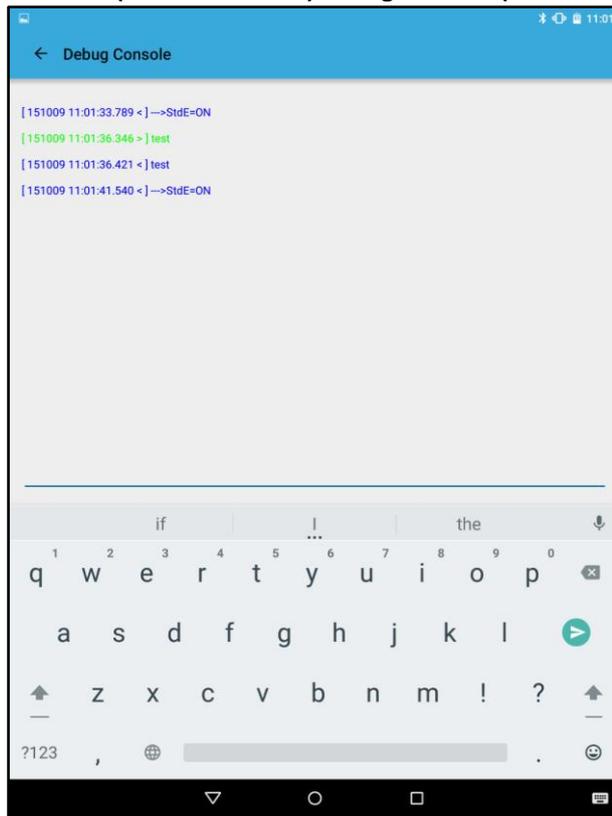


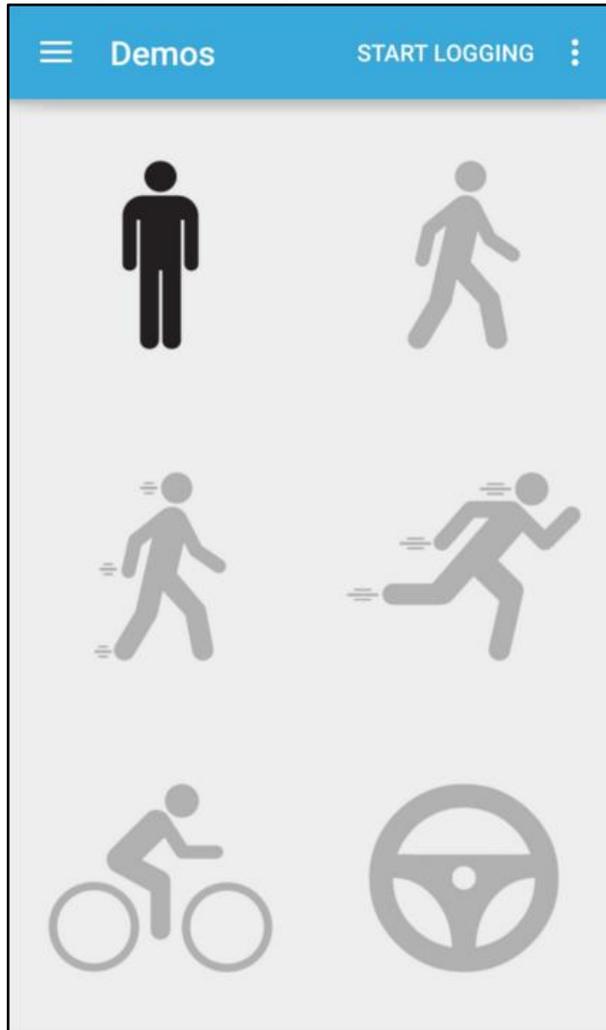
Figure 15: BlueMS (Android version) debug console (stdin/stdout/stderr)



If the osxMotionAR algorithm is enabled (`OSX_BMS_MOTIONAR` define in `osx_bms_config.h`) another page is available to display one of the following activities:

- stationary
- walking
- fast walking
- jogging
- biking
- driving

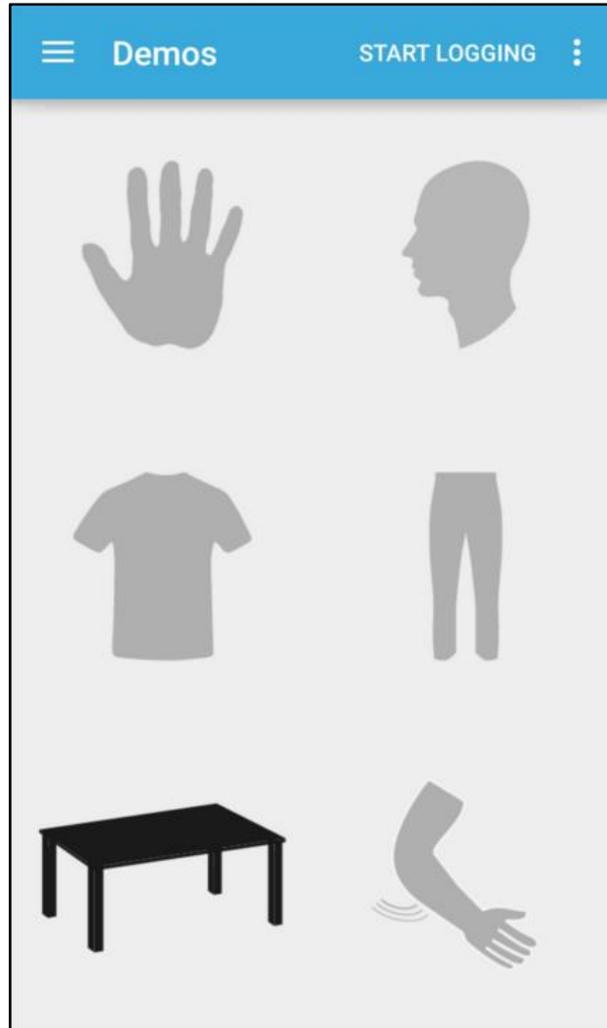
Figure 16: BlueMS (Android version) osxMotionAR activity recognition page



If the osxMotionCP algorithm is enabled (`OSX_BMS_MOTIONCP` define in `osx_bms_config.h`), another page is available to display the information about how the user is carrying the board; i.e., the phone carry position:

- on desk
- in hand
- near head
- shirt pocket
- trouser pocket
- arm swing

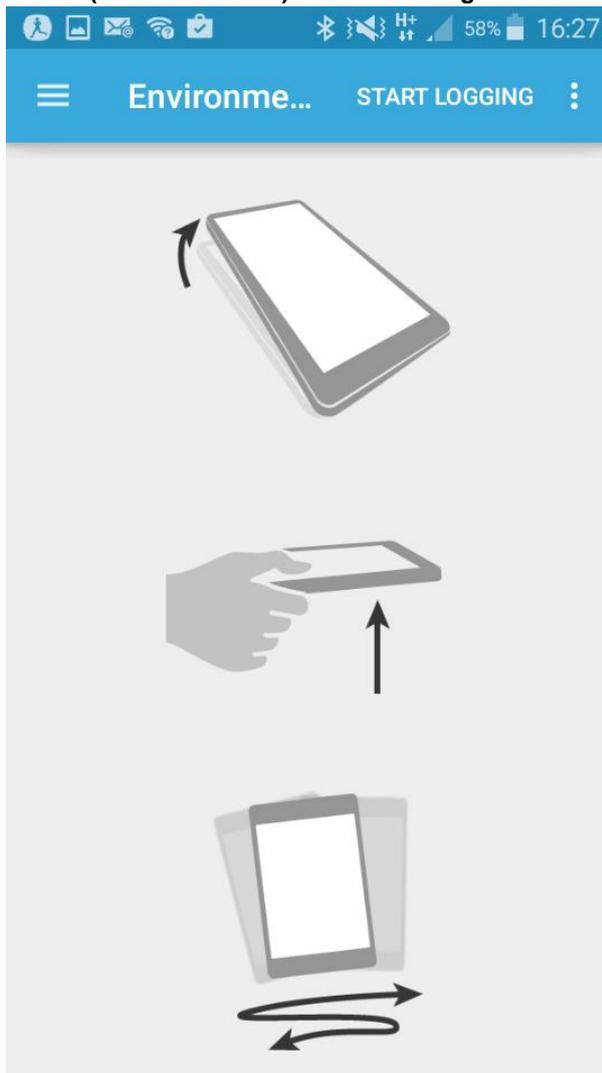
Figure 17: BlueMS (Android version) osxMotionCP carry position recognition page



If the osxMotionGR algorithm is enabled (`OSX_BMS_MOTIONGR` define in `osx_bms_config.h`), another page appears regarding phone gestures like

- pick up
- glance
- wake up

Figure 18: BlueMS (Android version) osxMotionGR gesture recognition page



If gesture recognition is enabled, the page shown below displays the results of the detected gestures, which can be a single tap (indicated by the circular symbol) or directional swipes (indicated by the double arrows).

Figure 19: BlueMS (Android version) gesture recognition page

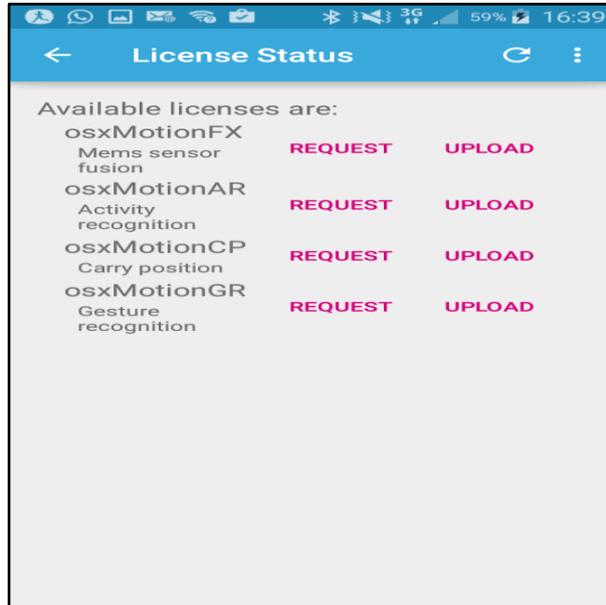


2.7 OPEN.MEMS Licenses activation

If the `OSX_BMS_LICENSE_H_FILE` in `Inc/osx_bms_config.h` is not defined, you must request and enable the OPEN.MEMS licenses using the BlueMS Android/iOS application. Initially, when no OPEN.MEMS licenses are activated, BLUEMICROSYSTEM3 can still be used to read and transmit sensor data values to the Android/iOS BlueMS application.

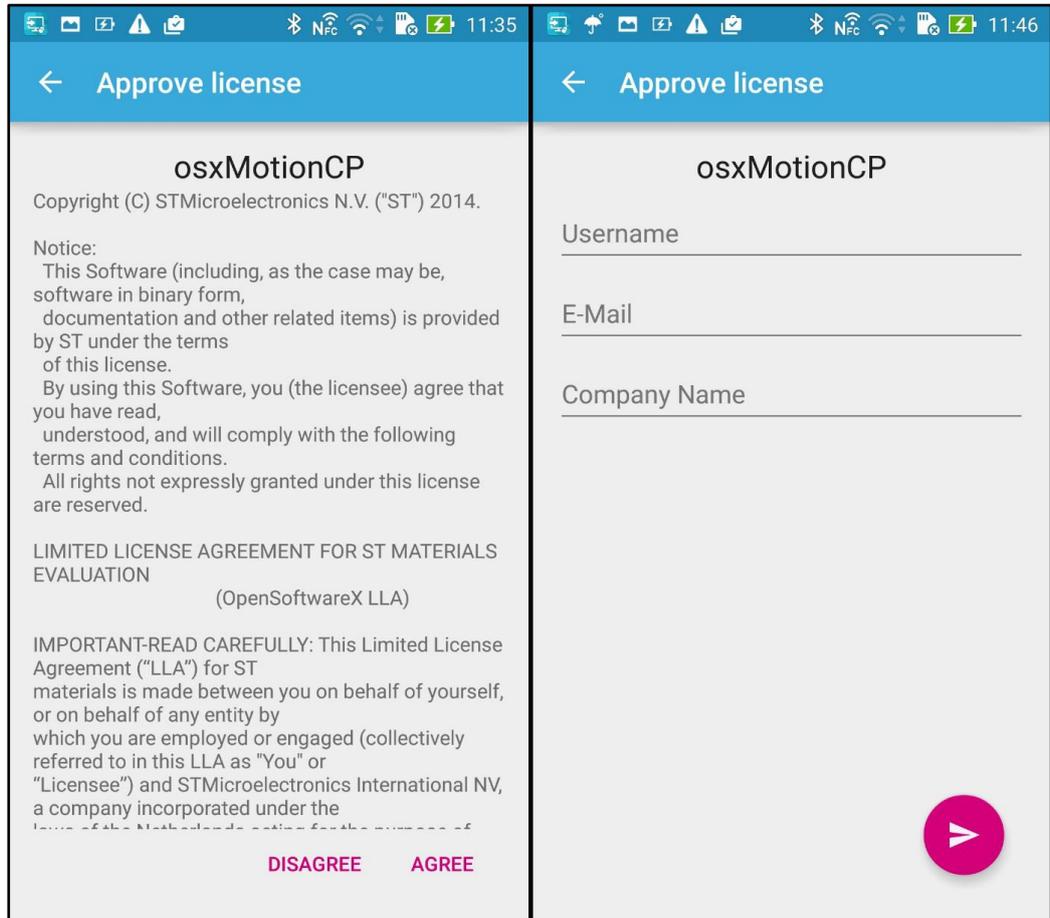
- 1 Select the “License Manager” menu option to open the page shown below.

Figure 20: BlueMS (Android version) License Status page



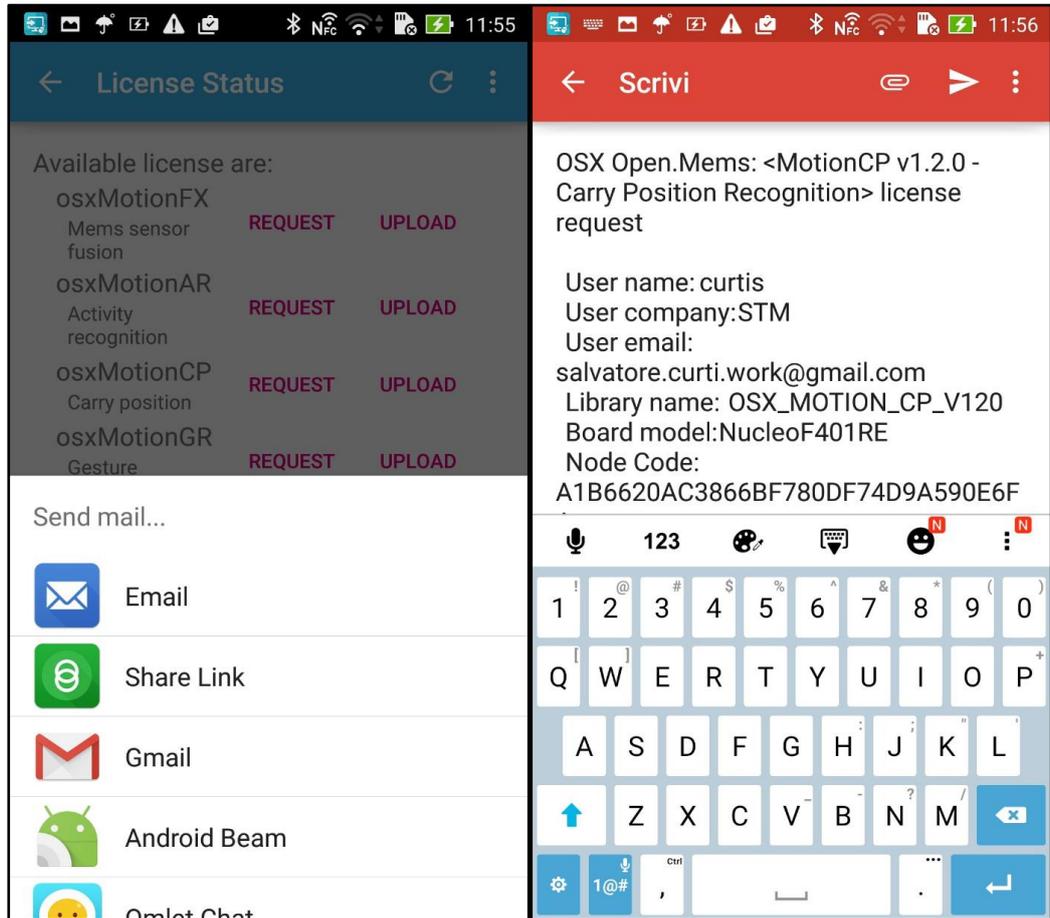
- 2 Select "request" for the license that we want to activate and after selecting "AGREE", the related "Approve license" page will be open.

Figure 21: BlueMS (Android version) Approve license page for osxMotionCP



- 3 Enter the Username, (valid) E-Mail and Company Name fields and click the arrow icon to generate the license request e-mail.

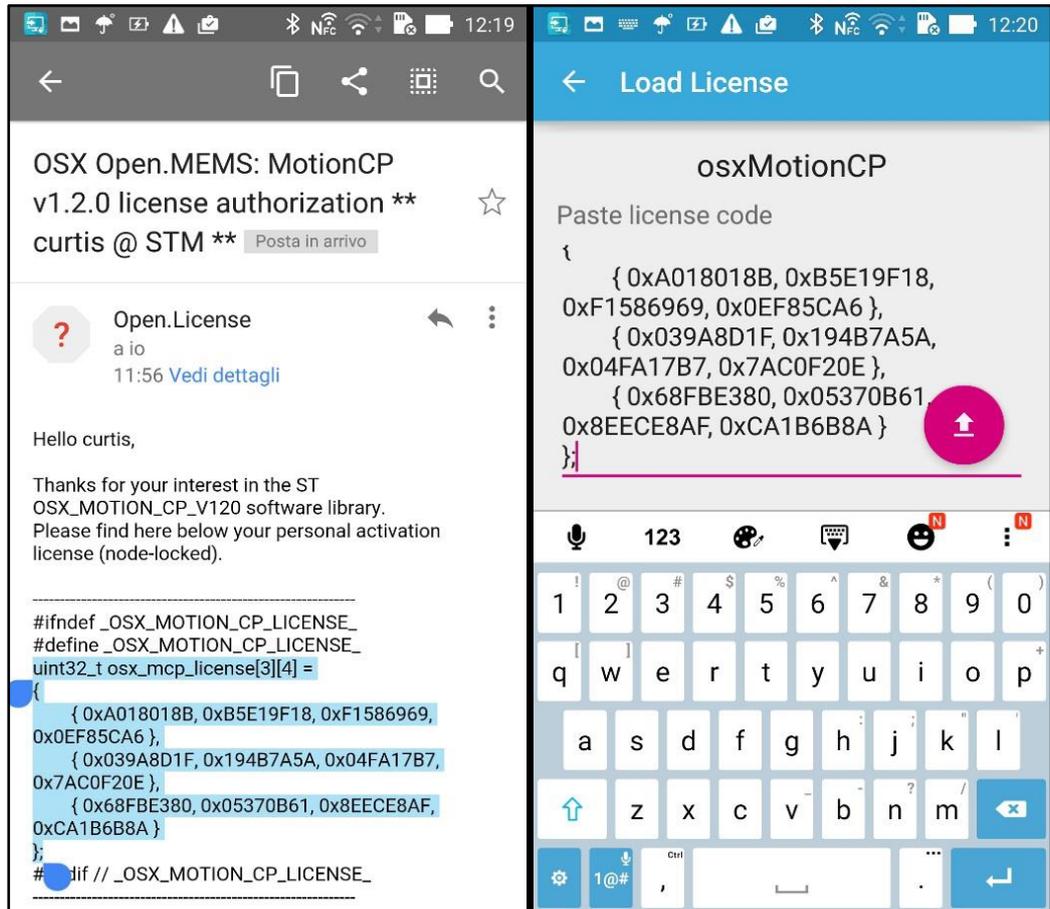
Figure 22: BlueMS (Android version) license request e-mail generated for osxMotionCP



- 4 The application returns to the initial License Status page (Figure 20: "BlueMS (Android version) License Status page") after the email request is sent.
- 5 Open the received e-mail and copy the license text.
- 6 Click the appropriate UPLOAD button in the License Status page Figure 20: "BlueMS (Android version) License Status page".

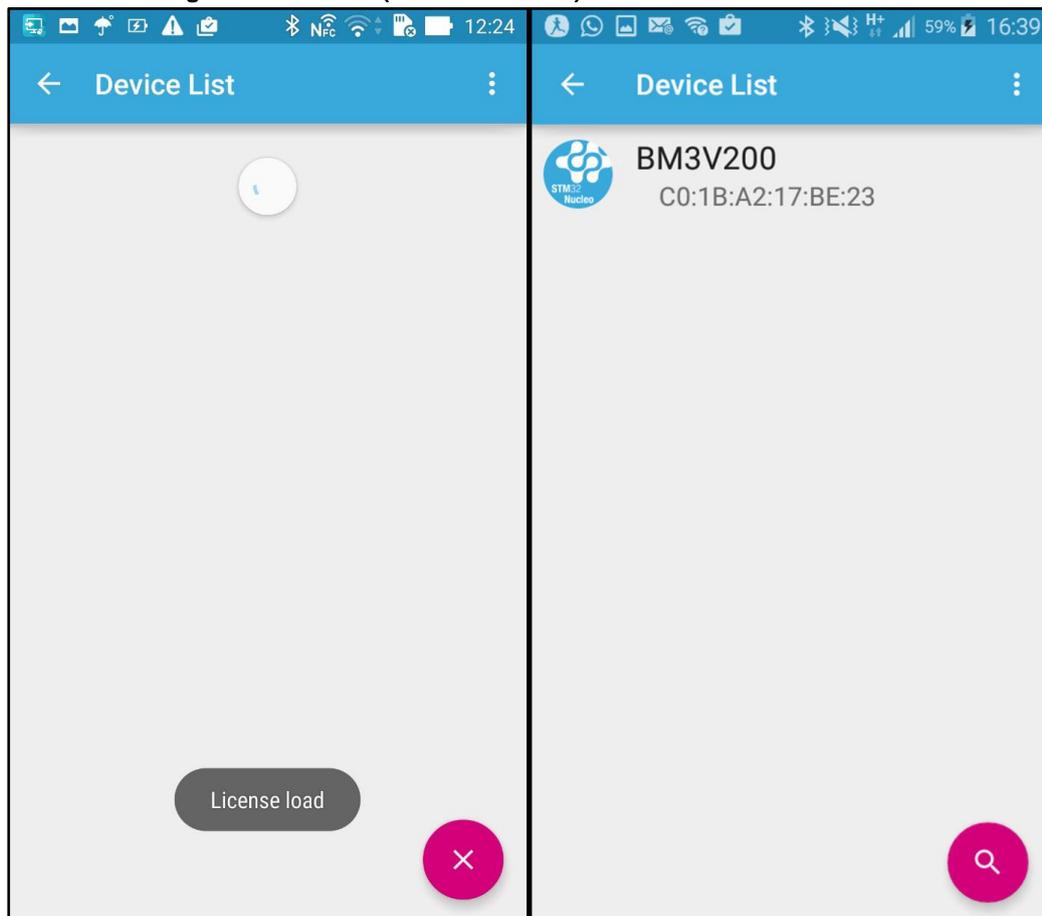
- Paste the license text in the Load License page and press the red upload arrow icon.

Figure 23: BlueMS (Android version) license upload for osxMotionCP



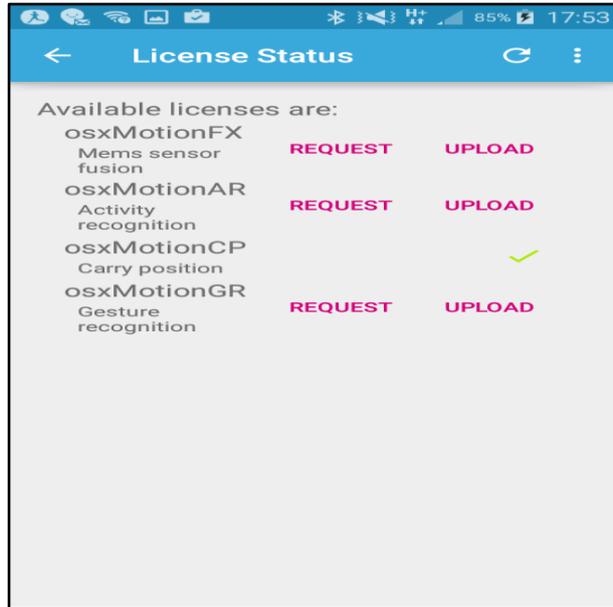
- 8 A board reboot is not necessary, but a new Bluetooth connection must be created.

Figure 24: BlueMS (Android version) osxMotionCP license load



9 Finally, the License Status page is updated.

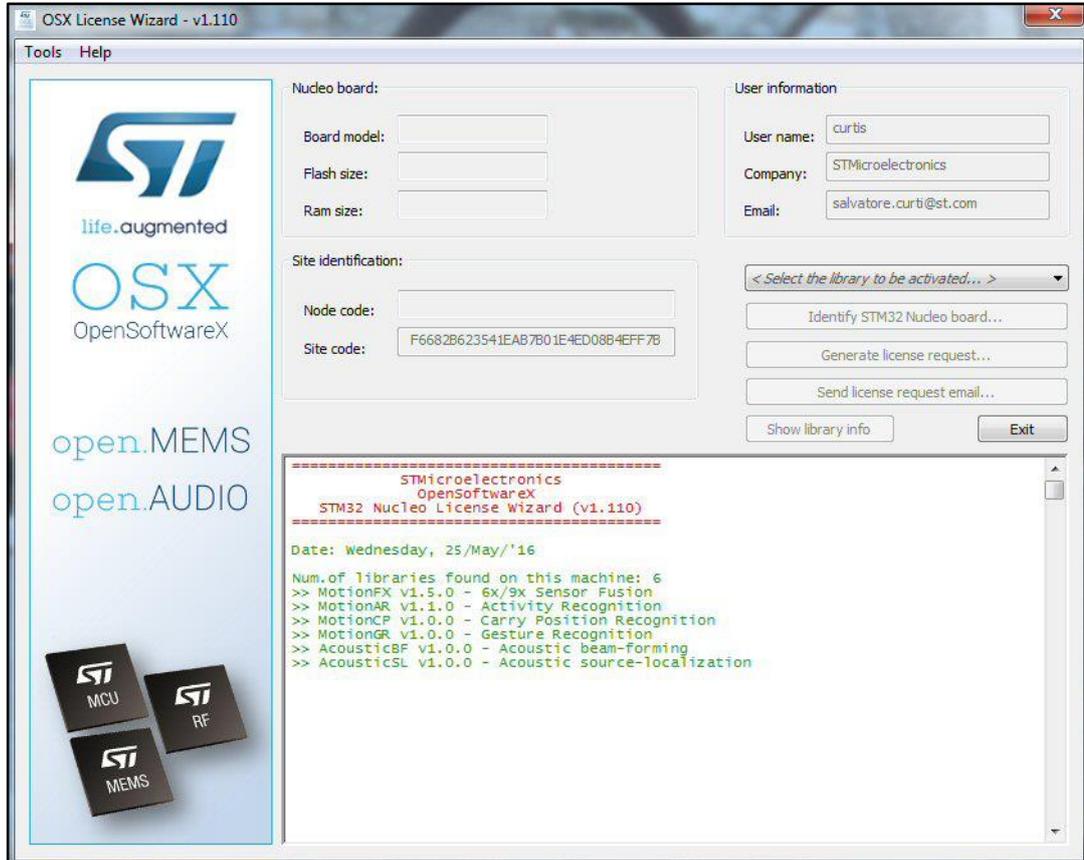
Figure 25: BlueMS (Android version) License Status page showing osxMotionCP license enabled



2.7.1 OPEN.MEMS license activation using the OSX License Wizard

If the `OSX_BMS_LICENSE_H_FILE` define in `Inc/osx_bms_config.h` is defined, you must request the OPEN.MEMS licenses using the OSX License wizard available for download on www.st.com.

Figure 26: Licenses request tool

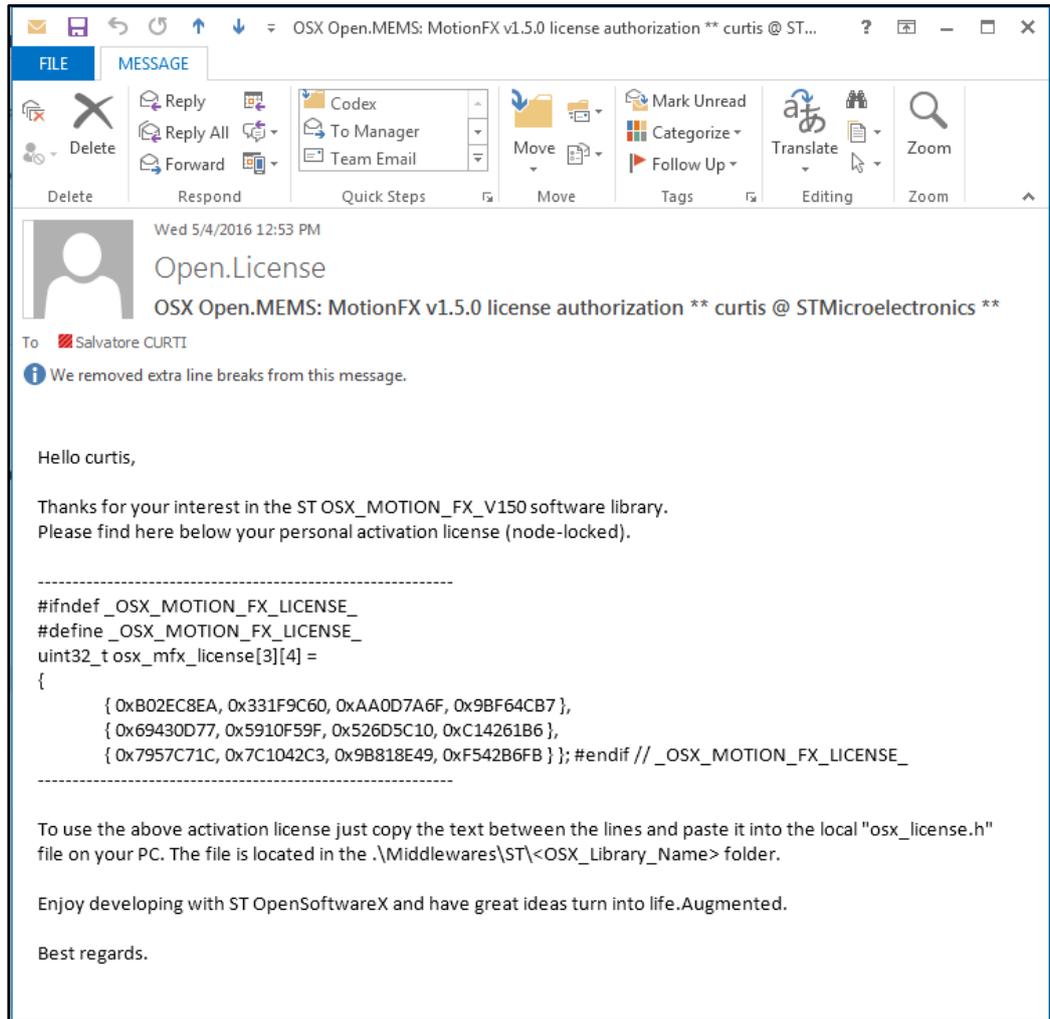


For each license:

- 1 Open OSX License Wizard
- 2 Selected the library to be activated (osxMotionFX, osxMotionAR, osxMotionCp, osxMotionGR)
- 3 Click identify STM32 Nucleo board
- 4 Click generate License Request and accept the terms in the license agreement
- 5 Click send License request email

6 Send the email

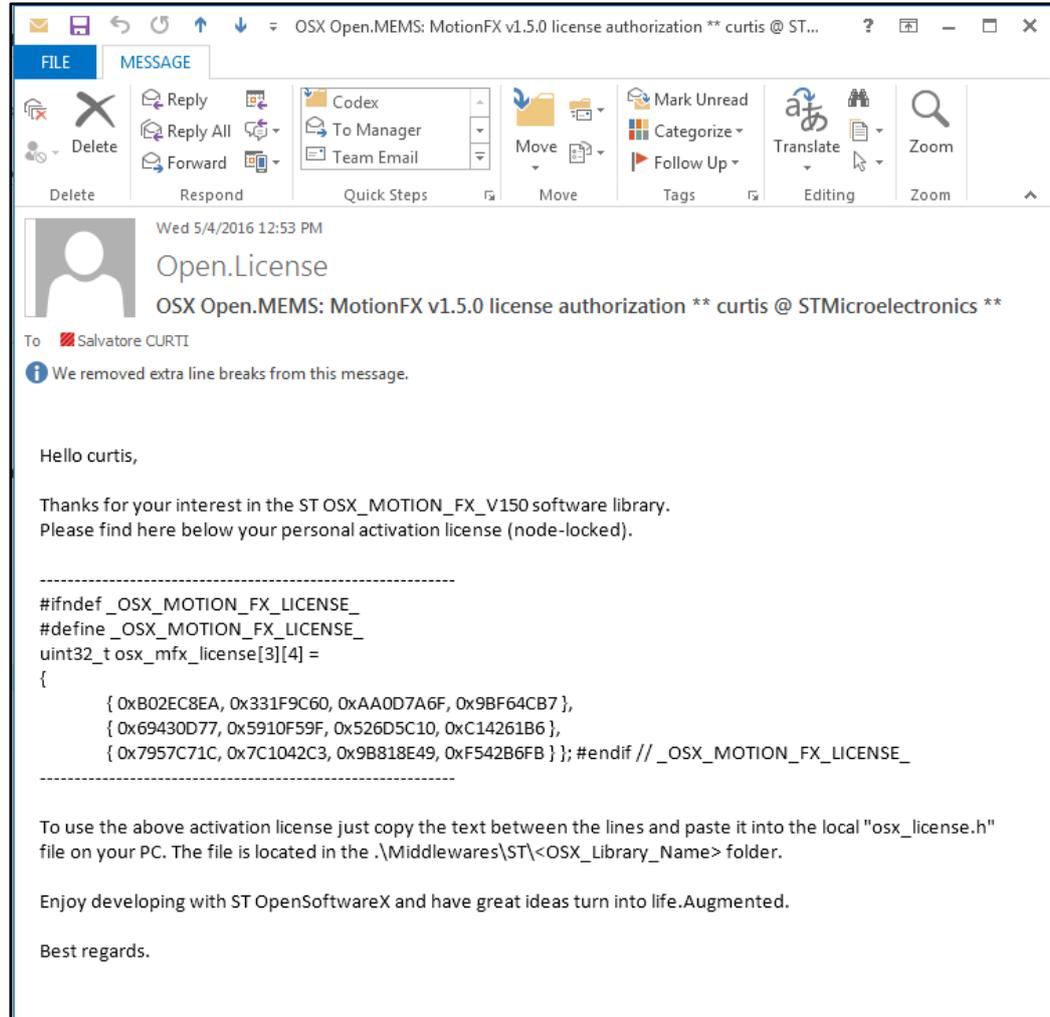
Figure 27: License request email generated by the OSX License Wizard



- 7 Open the received email and copy the received license codes into in the respective osx_license.h files.

License files are located in the Middleware directory. For example for enabling the osxMotionFX is necessary to put the license in Middlewares/ST/STM32_OSX_MotionFX_Library/osx_license.h file.

Figure 28: License activation email received



- 8 Compile BLUEMICROSYSTEM3 again to enable the licenses.

3 System setup guide

3.1 Hardware description

This section describes the hardware components needed for developing a sensor-based application.

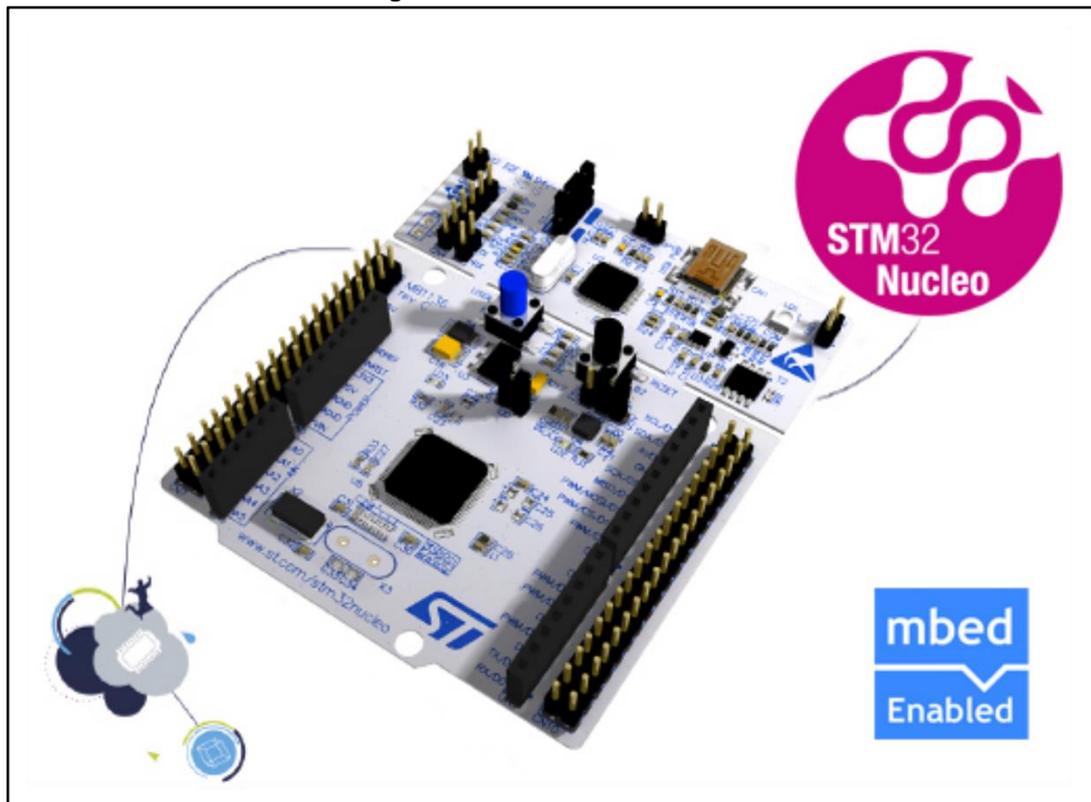
The individual components are described below.

3.1.1 STM32 Nucleo platform

The STM32 Nucleo boards provide an affordable and flexible way for users to try out new ideas and build prototypes with any STM32 microcontroller lines. The Arduino™ connectivity support and ST morpho headers 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 any separate probe 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.

Information regarding the STM32 Nucleo board is available on www.st.com at <http://www.st.com/stm32nucleo>

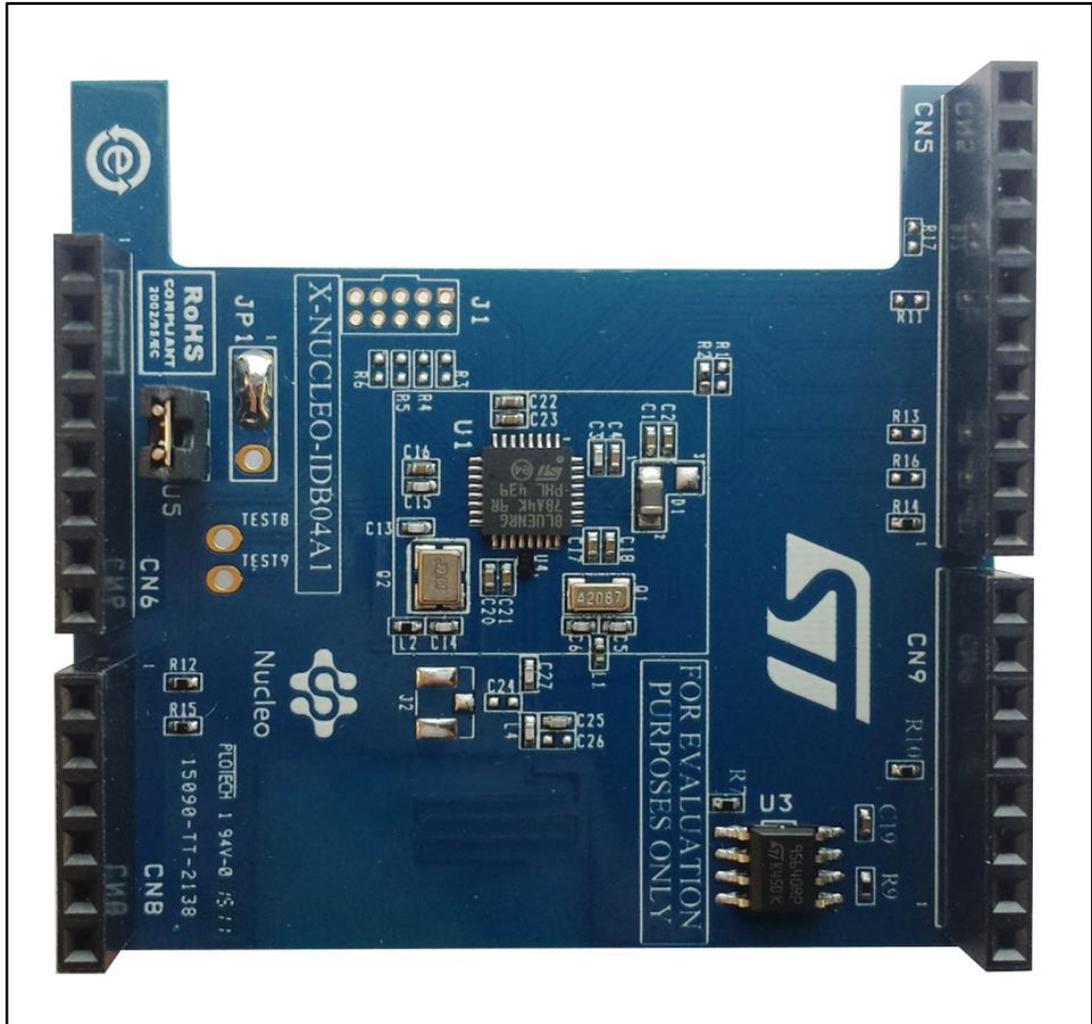
Figure 29: STM32 Nucleo board



3.1.2 X-NUCLEO-IDB04A1 expansion board

The X-NUCLEO-IDB04A1 is a Bluetooth BlueNRG expansion board usable with the STM32 Nucleo system. The BlueNRG is a very low power Bluetooth low energy (BLE) single-mode network processor, compliant with Bluetooth specifications core 4.0.

Figure 30: X-NUCLEO-IDB04A1 expansion board

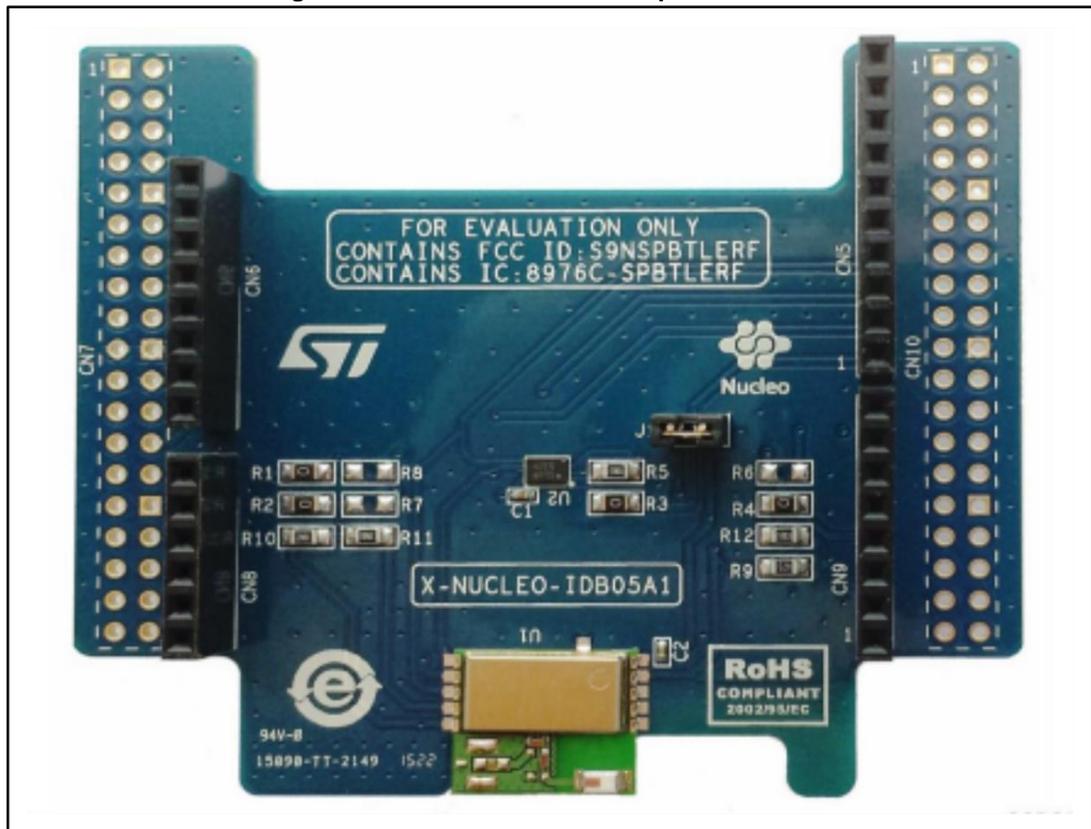


Information regarding the X-NUCLEO-IDB04A1 expansion board is available on www.st.com at <http://www.st.com/x-nucleo>.

3.1.3 X-NUCLEO-IDB05A1 expansion board

The X-NUCLEO-IDB05A1 is a Bluetooth low energy evaluation board based on the SPBTLE-RF BlueNRG-MS RF module to allow expansion of the STM32 Nucleo boards. The SPBTLE-RF module is FCC (FCC ID: S9NSPBLERF) and IC certified (IC: 8976C-SPBLERF). The BlueNRG-MS is a very low power Bluetooth low energy (BLE) single-mode network processor, compliant with Bluetooth specification v4.2. X-NUCLEO-IDB05A1 is compatible with the ST morpho and Arduino™ UNO R3 connector layout. This expansion board can be plugged into the Arduino UNO R3 connectors of any STM32 Nucleo board.

Figure 31: X-NUCLEO-IDB05A1 expansion board



Information about the X-NUCLEO-IDB05A1 expansion board is available on www.st.com at <http://www.st.com/x-nucleo>

3.1.4 X-NUCLEO-NFC01A1 expansion board

The X-NUCLEO-NFC01A1 is an expansion board based on the M24SR64-Y device. This expansion board can be plugged on the Arduino UNO R3 connectors of any STM32 Nucleo board.

The M24SR64-Y device is a dynamic NFC/RFID tag IC with a dual interface. It embeds 64 Kbit EEPROM memory, and can be operated from:

- an I²C interface
- a 13.56 MHz RFID reader or a NFC phone.

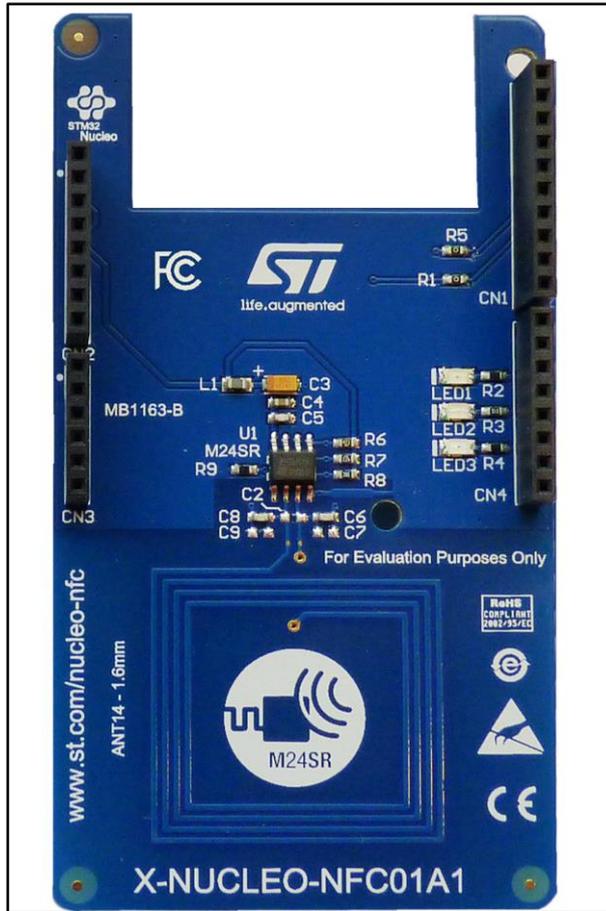
The I²C interface uses a two-wire serial interface, consisting of a bidirectional data line and a clock line. It behaves as a slave with respect to the I²C protocol.

The RF protocol is compatible with:

- ISO/IEC 14443 Type A
- NFC Forum Type 4 Tag.

The board is powered through the Arduino UNO R3 connectors and includes three general purpose LEDs.

Figure 32: X-NUCLEO-NFC01A1 M24SR64-Y dynamic NFC tag expansion board



Information regarding the X-NUCLEO-NFC01A1 expansion board is available on www.st.com at <http://www.st.com/x-nucleo>

3.1.5 X-NUCLEO-IKS01A1 expansion board

The X-NUCLEO-IKS01A1 is a sensor expansion board for the STM32 Nucleo board. It is also compatible with Arduino UNO R3 connector layout and is designed around humidity (HTS221), pressure (LPS25HB) and motion (LIS3MDL and LSM6DS0) sensing devices. The X-NUCLEO-IKS01A1 interfaces with the STM32 MCU via the I²C pin, and the user can change the default I²C port and the device IRQ by changing a resistor on the evaluation board.

You can attach the LSM6DS3 DIL24 expansion component and use it instead of the one of the LSM6DS0 sensors.

Figure 33: X-NUCLEO-IKS01A1 expansion board



Information about the X-NUCLEO-IKS01A1 expansion board is available on www.st.com at <http://www.st.com/x-nucleo>.

3.1.6 X-NUCLEO-6180XA1 expansion board

The X-NUCLEO-6180XA1 is an expansion board for the STM32 Nucleo system, also compatible with Arduino UNO R3 connector layout and designed around STMicroelectronics VL6180X proximity, gesture and ALS sensor, based on the ST FlightSense™ Time-of-Flight technology.

The board allows the user to test VL6180X functionality and develop relevant applications. It includes:

- a 4-Digit display to render either the range value in mm or the ambient light value in lux
- a switch to select the value type to be displayed
- a 2.8 V regulator to supply the VL6180X
- two level shifters to adapt the I/O level to the microcontroller main board
- the necessary connectivity for the application

For applications which implement gesture recognition, you need to plug two additional satellite sensors to the connectors marked LEFT and RIGHT. The third connector (BOTTOM) is not used.

Figure 34: X-NUCLEO-6180XA1 expansion board

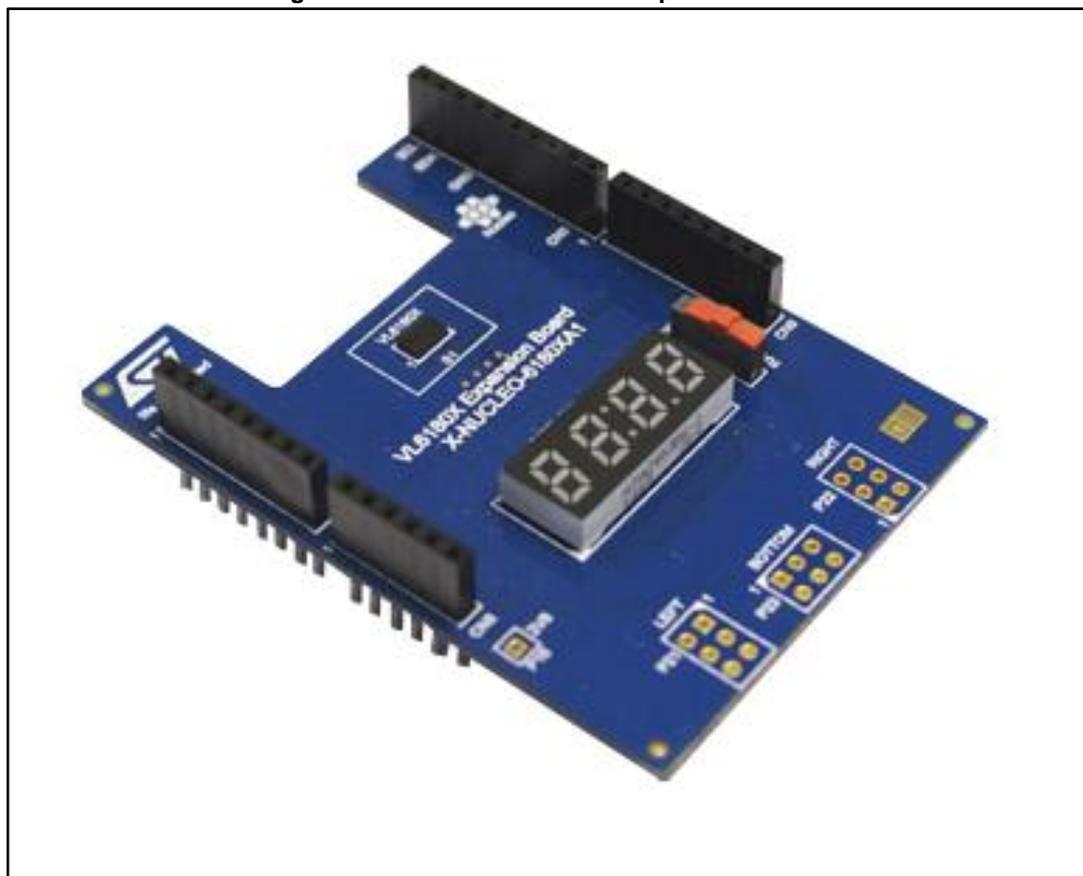
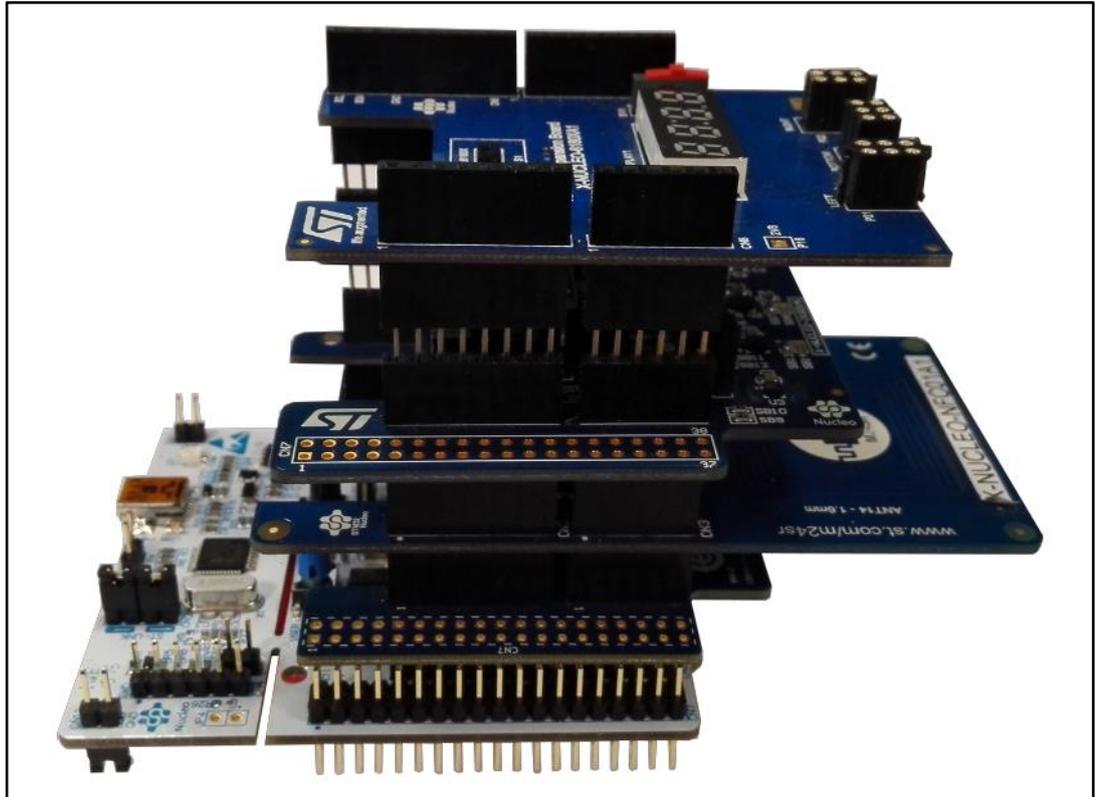


Figure 35: STM32 Nucleo + X-NUCLEO-IDB05A1 + X-NUCLEO-NFC01A1 + X-NUCLEO-IKS01A1 + X-NUCLEO-6180XA1 stack



3.2 Software description

The following software components are required in order to set up the suitable development environment for creating applications for the STM32 Nucleo equipped with the NFC, sensors, FlightSense and BlueNRG expansion boards:

- BLUEMICROSYSTEM3: a Bluetooth low energy, sensors and NFC tag software for STM32Cube. The BLUEMICROSYSTEM3 firmware and related documentation is available on st.com.
- Development tool-chain and Compiler: the STM32Cube expansion software supports the three following environments:
 - IAR Embedded Workbench for ARM® (EWARM) toolchain + ST-Link
 - RealView Microcontroller Development Kit (MDK-ARM) toolchain + ST-LINK
 - System Workbench for STM32 + ST-LINK

3.3 Hardware and software setup

This section describes the separate hardware and software setup procedures, as well as the combined system setup.

3.3.1 Hardware setup

The following hardware components are required:

1. one STM32 Nucleo Development platform (order code: NUCLEO-F401RE or NUCLEO-L476RG)
2. one NFC expansion board (order code: X-NUCLEO-NFC01A1)

3. one sensors expansion board (order code: X-NUCLEO-IKS01A1)
4. one FlightSense expansion board (order code: X-NUCLEO-6180XA1)
5. two mini-PCB VL6180X satellites (order code: VL6180X-SATEL)
6. one BlueNRG Bluetooth low energy expansion board (order code: X-NUCLEO-IDB04A1 or X-NUCLEO-IDB05A1)
7. one USB type A to Mini-B USB cable to connect the STM32 Nucleo to the PC

3.3.2 Software setup

This section lists the minimum requirements to set up the SDK, run the sample testing scenario based on the GUI utility and customize applications.

3.3.2.1 Development tool-chains and compilers

Choose one of the Integrated Development Environments supported by the STM32Cube expansion software and follow the system and setup details provided by the selected IDE provider.

3.3.3 System setup guide

This section describes how to setup different hardware parts before writing and executing an application on the STM32 Nucleo board with the sensors expansion board.

3.3.3.1 STM32 Nucleo and sensor expansion boards setup

The STM32 Nucleo board integrates the ST-LINK/V2-1 debugger/programmer. You can download the relevant version of the ST-LINK/V2-1 USB driver by searching STSW-LINK008 or STSW-LINK009 on www.st.com (based on your version of Microsoft Windows).

Connect the following boards via the Arduino UNO R3 extension connector

4 Revision history

Table 2: Document revision history

Date	Version	Changes
25-Nov-2015	1	Initial release.
25-Jan-2016	2	Throughout document: added reference to additional "gesture recognition" algorithm, and minor text edits. Updated Figure 1: "BLUEMICROSYSTEM3 software architecture" Updated Section 2.5: "Sample application description" Updated Section 2.6: "Android and iOS sample client application" and added Figure 18: "BlueMS (Android version) gesture recognition page" Updated Section 3.1.6: "X-NUCLEO-6180XA1 expansion board" Updated Section 3.3.1: "Hardware setup" Updated Section 3.3.3.1: "STM32 Nucleo and sensor expansion boards setup"
09-Sep-2016	3	Minor text and formatting edits throughout document Updated Section "Introduction" Updated Section 2.1: "Overview" Updated Figure 1: "BLUEMICROSYSTEM3 software architecture" Updated Section 2.5: "Sample application description" Updated Section 2.6: "Android and iOS sample client application" Updated Figure 14: "BlueMS (Android version) serial console (stdout/stderr)" Added Section 2.7: "OPEN.MEMS Licenses activation" Added Section 2.7.1: "OPEN.MEMS license activation using the OSX License Wizard" Removed section "References"

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2016 STMicroelectronics – All rights reserved