UM1936
User manual

Getting started with osxMotionAR activity recognition library for X-CUBE-MEMS1 expansion for STM32Cube

Introduction

This document describes how get started with the osxMotionAR software package.

The osxMotionAR activity recognition algorithm is provided in static library format and is designed for use in combination with the NUCLEO-F401RE or with the NUCLEO-L476RG(STM32 Nucleo development boards) and X-NUCLEO-IKS01A1 (motion MEMS and environmental sensor expansion board) or X-NUCLEO-IKS01A1 plus STEVAL-MKI160V1 (LSM6DS3 adapter board for standard DIL24 socket). The osxMotionAR library is integrated in a software package providing implementation examples for the above platforms.

The software is based on STM32Cube technology. Information about STM32Cube is available on www.st.com at *http://www.st.com/stm32cube*

Contents

List of tables

List of figures

1 osxMotionAR library add-on to X-CUBE-MEMS1 software expansion for STM32Cube

1.1 osxMotionAR overview

The osxMotionAR library is a complete middleware solution aimed at building applications strictly for 3D accelerometer sensors.

The software runs on the STM32 microcontroller and includes drivers to recognize the available inertial sensors (currently LSM6DS0 or LSM6DS3).

It provides real-time information on the type of activity performed by the user:

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

It is built on the STM32Cube software platform to facilitate portability across different STM32 microcontrollers.

The key package features include:

- complete middleware to build applications specifically for LSM6DS0 or LSM6DS3 motion sensor accelerometer sections;
- osxMotionAR activity recognition middleware (under open.MEMS license)
- easy portability across different MCU families, thanks to STM32Cube
- sample applications to transmit activity data to a PC
- sample implementations available for the X-NUCLEO-IKS01A1 board (optionally with the LSM6DS3 adapter board) when connected to NUCLEO-F401RE or to NUCLEO-L476RG
- the software comes with driver implementation examples, running on NUCLEO-F401RE and NUCLEO-L476RG mounted with the X-NUCLEO-IKS01A1 expansion.

1.2 osxMotionAR architecture

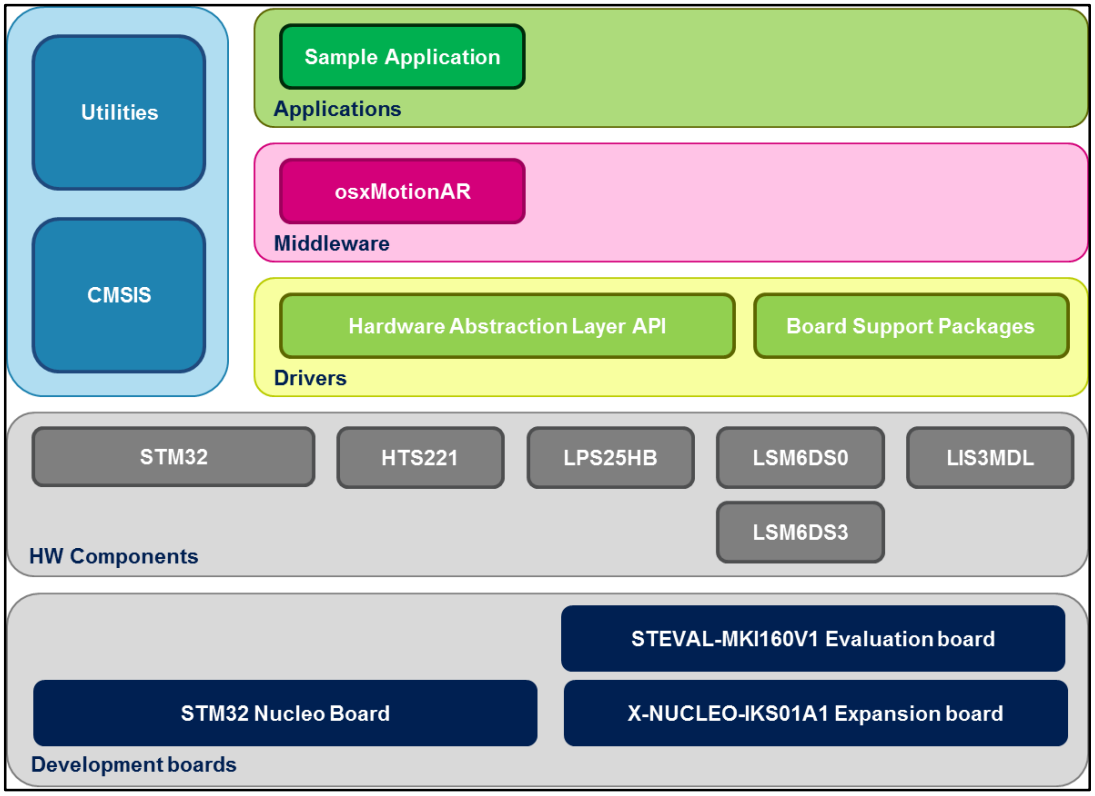
The following software layers are used by the application to access and use the sensor expansion board:

- STM32Cube HAL layer: the HAL driver layer provides a generic, multi-instance, simple set of APIs (application programming interfaces) to interact with the upper layers (application, libraries and stacks). It is composed of generic and extension APIs. It is directly built around a generic architecture and allows the layers that are built upon, such as middleware layers, to implement their functionalities avoiding dependencies on the specific hardware configuration for a given microcontroller unit (MCU). This structure improves library code reusability and guarantees easy portability to other devices.
- Board support package (BSP) layer: the software package needs to support all the available peripherals on the STM32 Nucleo board apart from the MCU. This software is included in the board support package (BSP). This is a limited set of APIs which provides a programming interface for certain board specific peripherals; e.g., the LED, user button, etc. This interface also helps in identifying the specific board version. If

the sensor expansion board is used, it provides the programming interface for various inertial and environmental sensors. It provides support for initializing and reading sensor data.

The diagram below outlines the software architecture of the package:

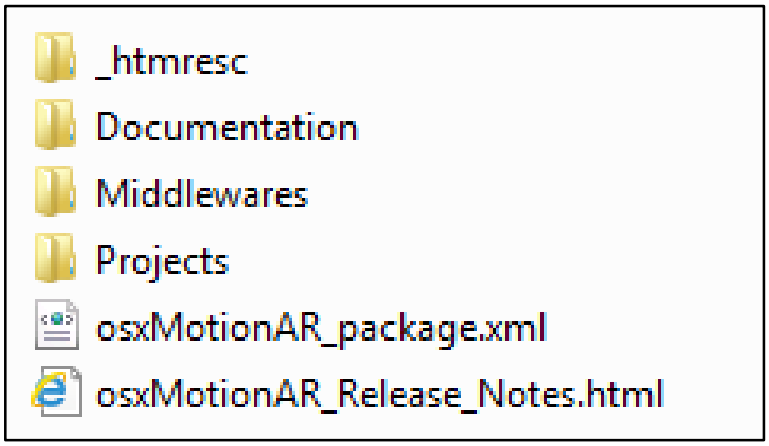
Figure 1: osxMotionAR plus X-CUBE-MEMS1 software architecture



1.3 osxMotionAR folder structure

The image below outlines the package file system architecture:

Figure 2: osxMotionAR package folder structure



The following folders are included in the package:

- **Documentation**: this folder contains a compiled HTML file generated from the source code and documenting in detail the software components and APIs (doxygen).
- **Middlewares**: this folder contains the osxMotionAR static library binary code, the library header file, documentation, license information plus header file for node-locked license validation.
- **Projects**: this folder contains a sample application used to access sensors and activity data, provided for the NUCLEO-F401RE and NUCLEO-L476RG platforms according to three IDE (Integrated Development Environments) proprietary formats (IAR Embedded Workbench for ARM, µVision (MDK-ARM) toolchain, System Workbench for STM32).

1.4 osxMotionAR APIs

Detailed technical information fully describing the functions and parameters of the osxMotionAR APIs can be found in the osxMotionAR_Package.chm compiled HTML file located in the Documentation folder of the software package.

The osxMotionAR is provided as a node-locked library which allows derivative firmware images to run on a specific STM32 Nucleo device only. Licensing activation codes must be requested from ST and included in the project (and become part of the build process) prior to attempting its usage. The resulting firmware binary image will therefore be node-locked.

For complete information about the open.MEMS license agreement, please refer to the license file located in the Middlewares/ST/STM32_OSX_MotionAR_Library folder.

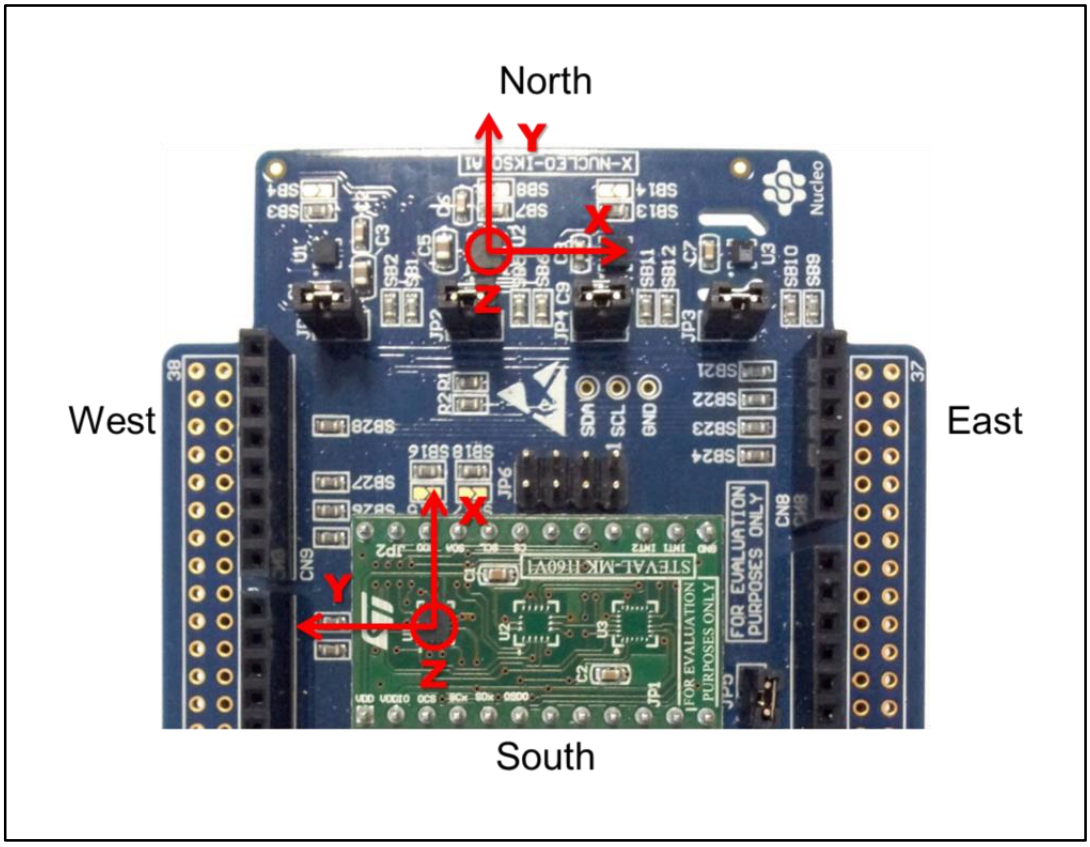
1.4.1 osxMotionAR library

The osxMotionAR is a real time activity recognition software solution. The algorithm only manages the data acquired from the accelerometer, at the low sampling frequency of 16 Hz to reduce host platform power consumption.

The exposed APIs of the osxMotionAR library are listed below:

- `uint8_t osx_MotionAR_GetLibVersion(char *version);`
  - retrieves the revision of the included core engine;
- `uint8_t osx_MotionAR_Initialize (void);`
  - performs osxMotionAR initialization and setup of the internal mechanism used for node-locking (See *Section 3.5.3: "osxMotionAR installer setup"*). The output for correct or incorrect initialization is 1 or 0 respectively (e.g., 0 is returned for license errors);
- `void osx_MotionAR_SetOrientation_Acc (const char *acc_orientation);`
  - this function is used to set the accelerometer data orientation; library configuration is usually performed immediately after the `osx_MotionAR_Initialize` function call.
    The required input is a pointer to a string of three characters indicating the direction of each of the positive orientations of the reference frame used for accelerometer data output, in the sequence x, y, z. Valid values are: n (north) or s (south), w (west) or e (east), u (up) or d (down).
    As shown in the figure below, the X-NUCLEO-IKS01A1 accelerometer sensor has an ENU orientation (x - East, y - North, z - Up), so the string is: “enu”, while the accelerometer sensor in STEVAL-MKI160V1 is NWU (x-North, y-West, z-Up): “nwu”.

Figure 3: Example x, y, z axes values



- `osx_MAR_output_t osx_MotionAR_Update (osx_MAR_input_t *data_in);`
  - The required input is a pointer to a structure containing the three axis accelerometer data expressed in [g]; the output is osxMAR_output, indicating the activity code and is an enum.

1.5 Sample application

The osxMotionAR middleware can be easily manipulated to build user applications; an application example is provided in the Projects folder.

It is designed to run on a NUCLEO-F401RE or a NUCLEO-L476RG board connected to an X-NUCLEO-IKS01A1 board (based on LSM6DS0), or a NUCLEO-F401RE or a NUCLEO-L476RG board connected to an X-NUCLEO-IKS01A1 plus a STEVAL-MKI160V1 board (based on LSM6DS3).

The application recognizes the performed activities in real time and stores them in the board memory for offline analysis or displays them in specific GUI. The algorithm recognizes stationary, walking, fast walking, jogging, bike riding and driving activities.

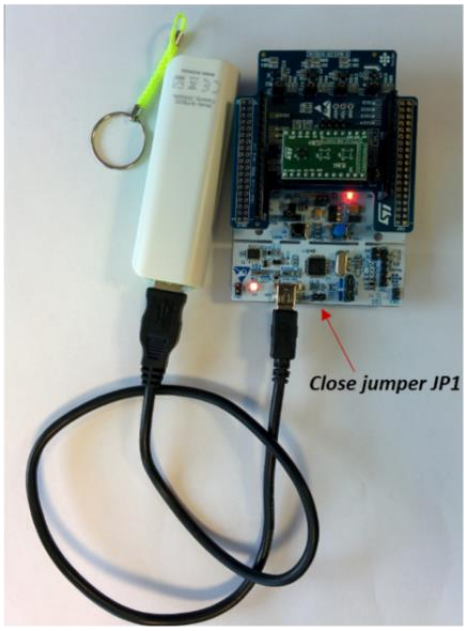
The detected activities are stored in the board for offline analysis or may be displayed in real time through a GUI. In the former case, the application runs in stand-alone mode, while in the latter, it runs in PC GUI driven mode.

Stand-alone mode

In stand-alone mode, the STM32 Nucleo board may be supplied by a portable battery pack (to make the user experience more comfortable, portable and free of any PC connections. The user can choose to display latest recognized activity through an on board LED. During

acquisition and algorithm operation, recognized activities are stored in the microcontroller memory.

Figure 4: STM32 Nucleo board connected to battery pack; jumper JP1 closed



PC GUI driven mode

In this mode, a USB cable connection is required to monitor real time data. Once connected, launch the dedicated *Sensors_DataLog* GUI and connect it to the STM32 Nucleo board; for more details, refer to *Section 1.5.2: "PC GUI driven mode"*.

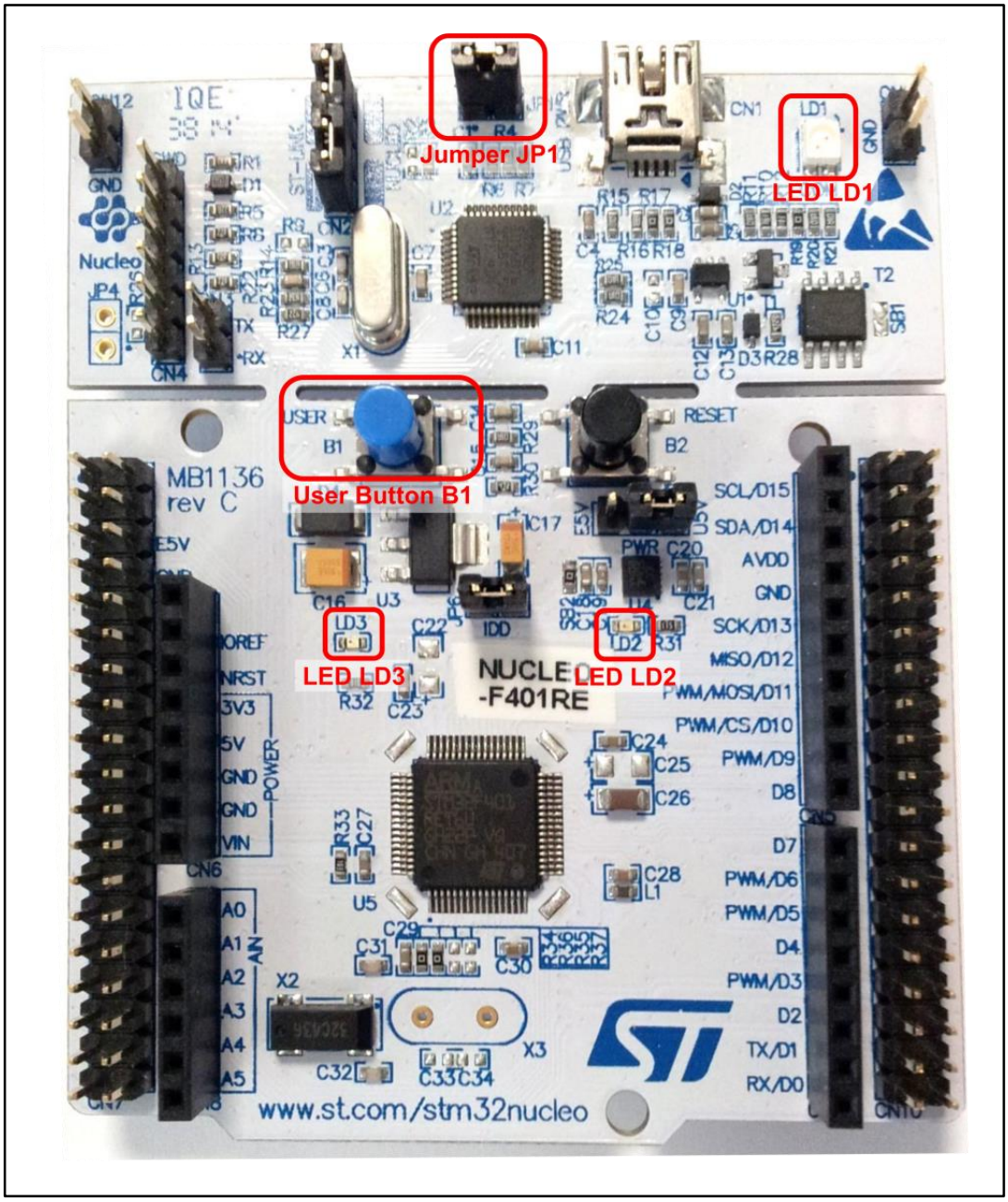
Table 1: Power supply scheme

Power Source	NUCLEO-F401RE settings	Working Mode
USB PC cable	JP1 open	Data upload to PC
Battery pack	JP1 closed	Activity Acquisition (Normal)

1.5.1 Stand-alone working mode

In stand-alone working mode, the user can power the board by means of an external battery pack (*Figure 4: "STM32 Nucleo board connected to battery pack; jumper JP1 closed"*), ensuring that jumper JP1 is fitted.

Figure 5: NUCLEO-F401RE board details



The above figure shows the user button B1 and the three LEDs of the NUCLEO-F401RE board. Once the board is powered, LED LD3 (PWR) turns ON and the tricolor LED LD1 (COM) begins blinking slowly due to the missing USB enumeration (refer to user manual UM1724 *STM32 Nucleo boards* for further details).

The NUCLEO-L476RG board presents similar layout.

When user button B1 is first pressed and LED LD2 (USER) is OFF, the system starts acquiring data from the accelerometer sensor and detects the performed activity; during this acquisition mode, rapid LED LD2 blinking indicates that the algorithm is running. During this phase the activity detected is stored in the MCU's internal flash memory.

Pressing button B1 a second time stops the algorithm (and the relative data storage session) and the LED LD2 displays the activity code according to a sequence of flashes described in *Table 2: "LED LD2 activity codes"*.

By pressing the button B1 a third time, the system goes in standby mode; i.e., the algorithm is not running and LED LD2 is off.

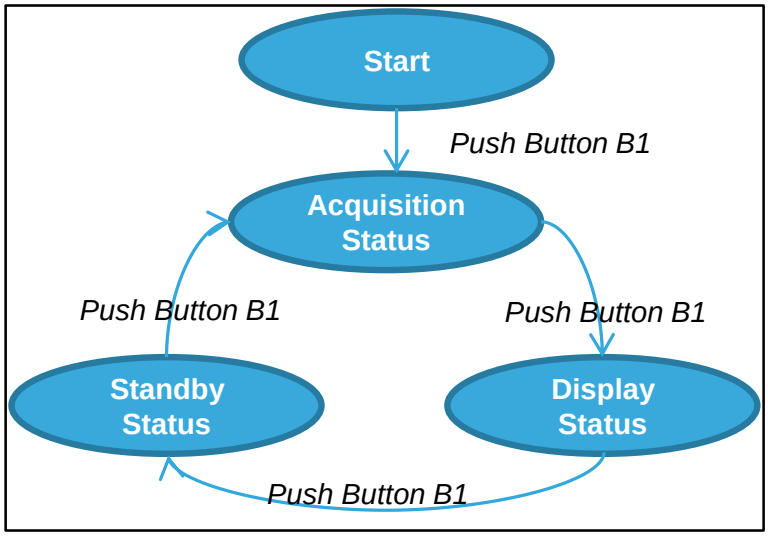
Pressing the button again initiates the algorithm and data storage once more (see *Figure 6: "State machine"*).

If the LED LD2 is ON after powering the board, this represents a warning message indicating that the flash memory is full or almost full (see *Section 1.5.4: "Data storage"*) or the library has an incorrect embedded license number.

Table 2: LED LD2 activity codes

Activity	LED LD2 blinking sequence (5 s interval)
Stationary	1
Walking	2
Fast walking	3
Running	4
Biking	5
Driving	6

Figure 6: State machine



During acquisition status, the algorithm is performed and recognized activity data are stored in the microcontroller flash memory. In order to retrieve those data, the NUCLEO-F401RE board (or the NUCLEO-L476RG board) has to be connected to a specific PC GUI (*Section 1.5.3: "Sensors_DataLog utility"* and *Section 1.5.4: "Data storage"*).

1.5.2 PC GUI driven mode

In this working mode, the NUCLEO-F401RE (or NUCLEO- L476RG) board is powered by PC via USB connection and controlled by a specific PC GUI.

This working mode allows the user to display the activity detected, accelerometer data, time stamp and eventually other sensors data, in real-time using the Sensors_DataLog utility GUI.

Once board is powered, launch the *Sensors_DataLog.exe* and drive the example application as described in *Section 1.5.3: "Sensors_DataLog utility"*.

In this working mode the data are not stored on MCU memory flash.

1.5.3 Sensors_DataLog utility

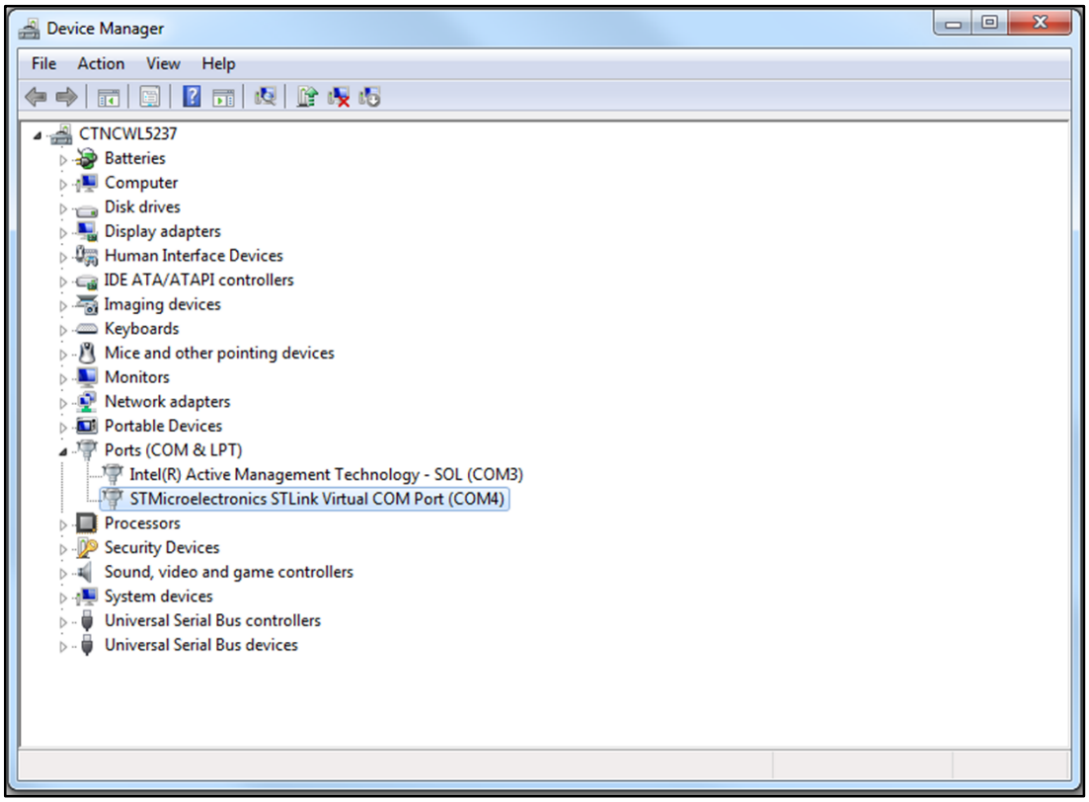
The osxMotionAR software package for STM32Cube uses the Windows *Sensors_DataLog* utility of the STM32CubeExpansion_MEMS1_ VX.X.X package in the ROOT_DIR\Utilities\PC_software folder.

Before using it, ensure that the necessary drivers are installed and the expansion board and the STM32 Nucleo board is connected to the PC.

Please follow these steps:

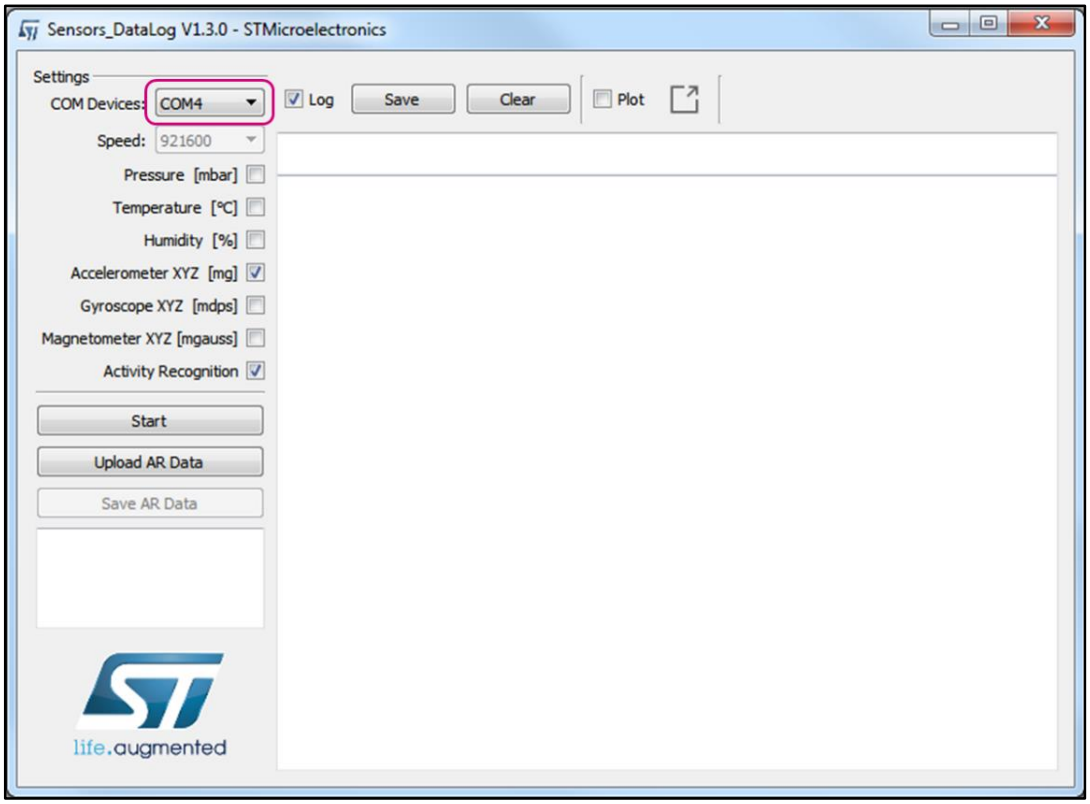
- **1.** Connect the STM32 Nucleo board to the PC with a USB cable (jumper JP1 not fitted); find the ST COM port in Windows Device Manager (COM4 in the figure below)

Figure 7: Windows Device Manager



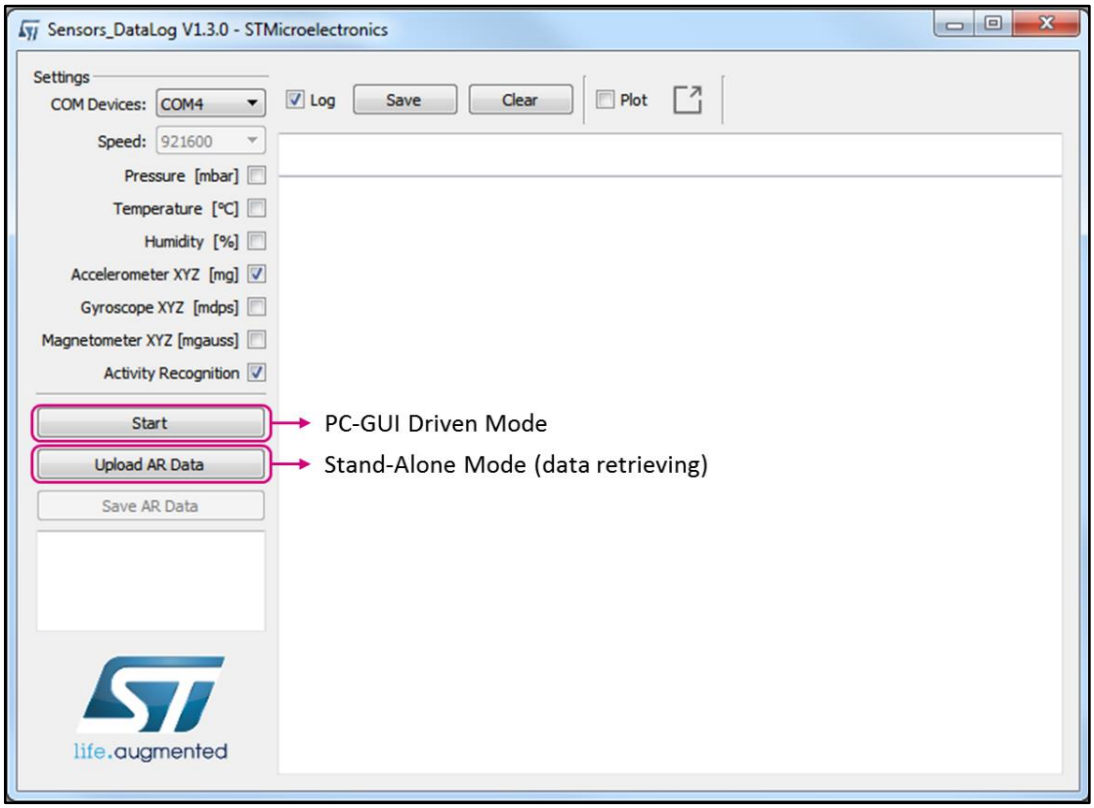
- **2.** Launch Sensors_DataLog.exe and check that the COM Device number is correct.

Figure 8: Sensors_DataLog utility screenshot 1



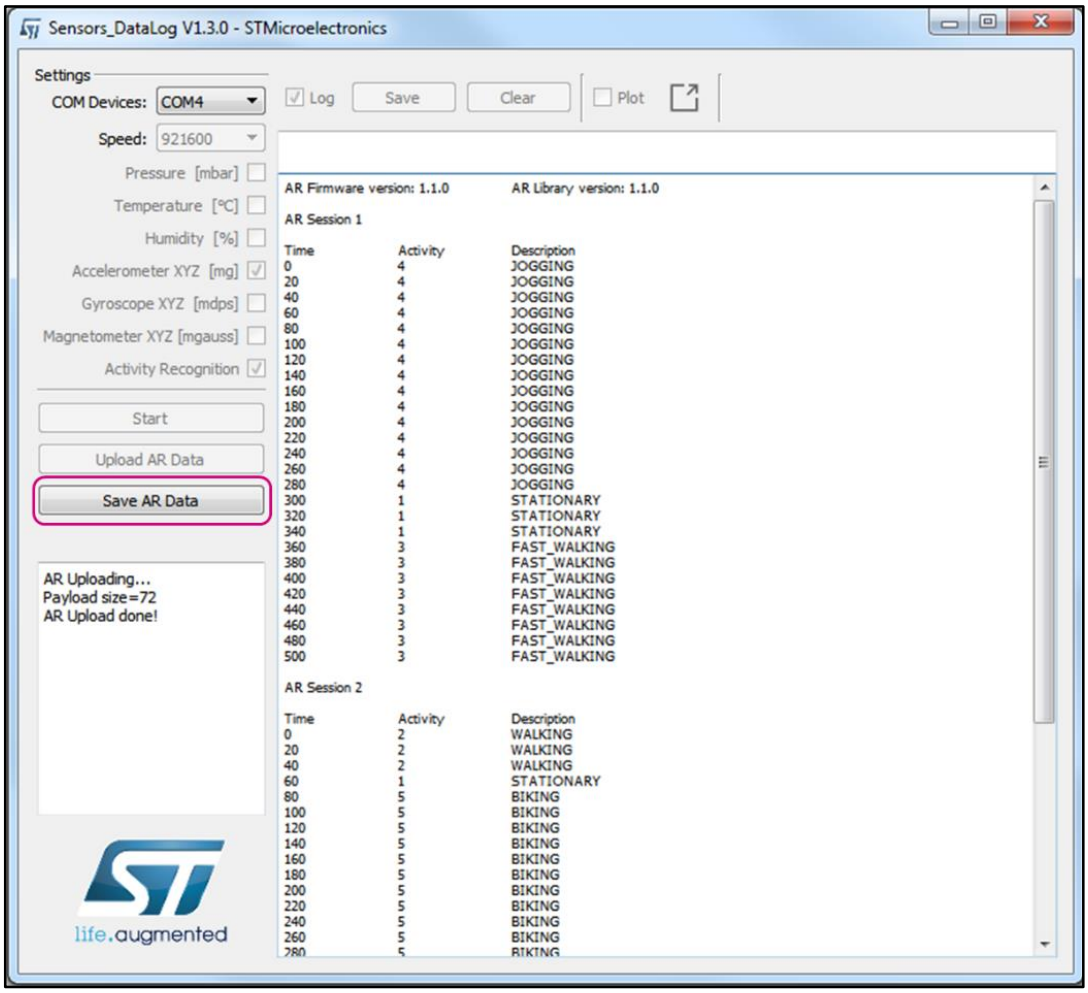
- **3.** At this point, you need to choose from the two possible operating modes. The two push button on the left “Start” and “Upload AR Data” (see Figure below) are to be used to run the AR algorithm, i.e. to start the acquisition in PC GUI driven mode, or to retrieve stored data from MCU’s flash after stand-alone mode session, respectively.

Figure 9: Sensors_DataLog utility screenshot 2



- **4a.** If the board has been working in stand-alone mode and user wants to retrieve stored data, press *Upload AR Data* to upload the stored activities data to the PC GUI. This operation automatically deletes acquired data from microcontroller. The figure below shows how the data is arranged in *Time*, *Activity* and *Description* columns. Activities with a duration of less than 20 seconds are not memorized.

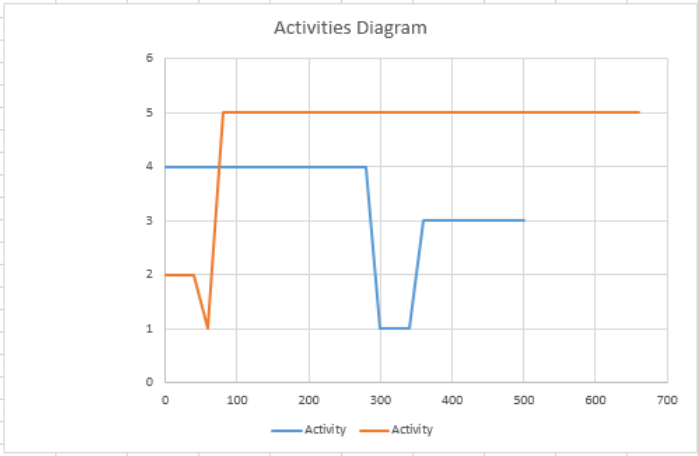
Figure 10: Sensors_DataLog utility screenshot: data upload



- **5a.** Press the *"Save AR Data"* button in the figure above to save the uploaded data in a .tsv file (named in the format *AR_log_yyyymmdd_hh.mm.ss.tsv*) located in a new folder called *SensorsDataLog*. A closer look at our example reveals two recording sessions; in the first one jogged for five minutes, then rested for one minute before start a fast walk recorded for almost three minutes. In the second session, the user walked for one minute and, after a very short rest (approximately 20 seconds), started cycling until end of session. The acquired data may be exported and plotted in an Excel file, as shown below.

Figure 11: Acquisition details – Excel screenshot

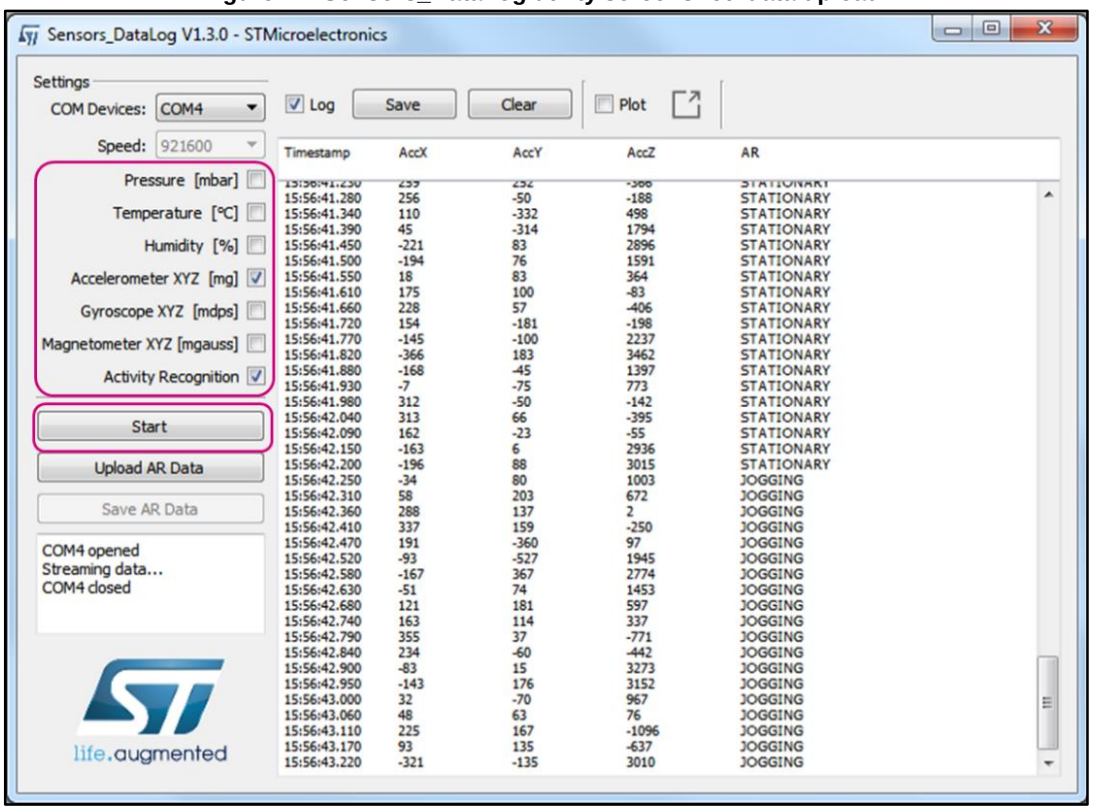
	A	B	C
1	AR Firmware version: 1.1.0		AR Library version: 1.0.1
2			
3	AR Session 1		
4			
5	Time	Activity	Description
6	0	4	JOGGING
7	20	4	JOGGING
8	40	4	JOGGING
9	60	4	JOGGING
10	80	4	JOGGING
11	100	4	JOGGING
12	120	4	JOGGING
13	140	4	JOGGING
14	160	4	JOGGING
15	180	4	JOGGING
16	200	4	JOGGING
17	220	4	JOGGING
18	240	4	JOGGING
19	260	4	JOGGING
20	280	4	JOGGING
21	300	1	STATIONARY
22	320	1	STATIONARY
23	340	1	STATIONARY
24	360	3	FAST_WALKING
25	380	3	FAST_WALKING
26	400	3	FAST_WALKING
27	420	3	FAST_WALKING
28	440	3	FAST_WALKING
29	460	3	FAST_WALKING
30	480	3	FAST_WALKING
31	500	3	FAST_WALKING
32			
33	AR Session 2		
34			
35	Time	Activity	Description
36	0	2	WALKING
37	20	2	WALKING
38	40	2	WALKING
39	60	1	STATIONARY
40	80	5	BIKING
41	100	5	BIKING
42	120	5	BIKING
43	140	5	BIKING
44	160	5	BIKING
45	180	5	BIKING



OR

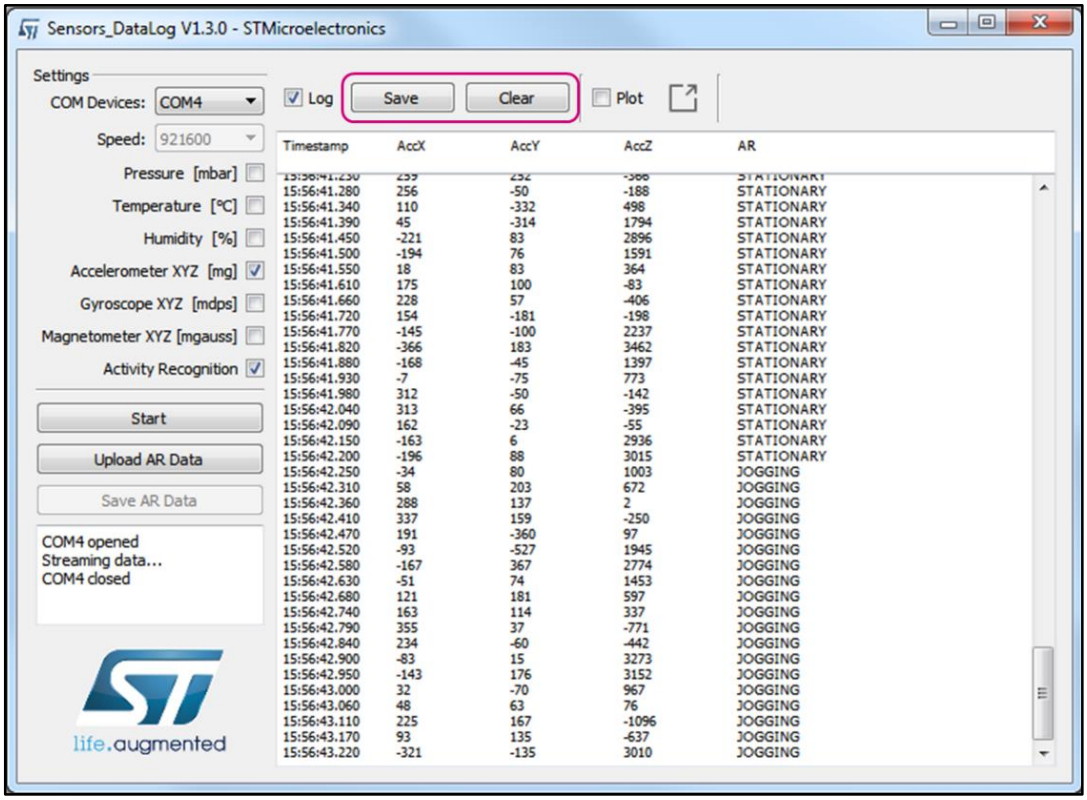
- **4b.** In case of PG GUI driven working mode, select between the various sensors (pressure, temperature, humidity, accelerometer, gyroscope, magnetometer) available on the expansion board (please note that accelerometer has to be selected to make Activity Recognition check box available), and press the “Start” button to commence the acquisition and activity computation session. Sensor data and Activity Recognition information will be shown as in the figure below.

Figure 12: Sensors_DataLog utility screenshot: data upload



- **5b.** Once acquisition has been stopped, the user can save the log in a file in the "SensorsDataLog" folder with the "Save" button, or clear it with the "Clear" button, as shown below.

Figure 13: Sensors_DataLog utility screenshot: data upload



1.5.4 Data storage

The example application allows the user to detect activity performed and store it in MCU flash memory. Data is automatically saved every 5 minutes to avoid excessive data loss in case of an unforeseen power fault. Data is also stored when the user stops acquisition by pressing button B1 to display data by LED.

When stored data is retrieved via the GUI (see *Section 1.5.3: "Sensors_DataLog utility"*), the MCU flash sector dedicated to this purpose is cleared.

LED LD2 should be OFF at power-on, unless:

- the flash memory is full or almost full
- the license number is incorrect

In the first case, the user can continue using the board for normal activity recognition and data storage (if enough space is available) as described in *Section 1.5.1: "Stand-alone working mode"*, by pushing the user button (LED switches off in 5 seconds). Data continues to be stored until the reserved sector is full, at which time the algorithm keeps running, but data is no longer stored.

The flash sector dedicated to data storage is 128 KB, allowing memorization of more than 16,000 data sets, this equates to:

- a minimum of 91 hours (activity changes and a new buffer is stored every 20 seconds)
- a maximum of 1365 hours (activity never changes and data is automatically stored every 5 minutes).

If LED LD2 switches ON at reset, no more than 1400 Bytes are free; i.e., from one hour recording time if data is stored every 20 seconds to 15 hours if data is automatically saved every five minutes.

If a large amount of data needs to be stored and no PC is available for data download and flash memory clean up, the MCU memory can be erased when LED LD2 is ON by holding the user push button down for at least 5 seconds. LED LD2 switches OFF and then starts blinking to indicate that acquisition mode has been activated and the activity data stored in the MCU has been erased.

If LED LD2 does not switch OFF after this operation, it means that the license number is wrong. In this case no action is allowed and the user must program the STM32 Nucleo board with the correct license number (see *Section 3.5.4: "osxMotionAR license wizard"* for details on how to obtain a license).

2 Algorithm performance

The activity recognition (AR) algorithm only uses data from the accelerometer and runs at a low frequency (16 Hz) to reduction power consumption.

The table below shows the performance of the AR algorithm in terms of recognition success rates. The data is based on 600 data sets collected from 71 individuals.

Table 3: Performance data

Activity	Detection probability	Best performance	Vulnerable	Carry positions
Stationary	91.41%		holding in hand and heavy texting	all: trouser pocket, shirt pocket, back pocket, near the head, etc.
Walking	98.85%	step rate ≥ 1.4 step/s	step rate ≤ 1.2 step/s	all
Fast walking	98.88%	step rate ≥ 2.0 step/s		all
Jogging	98.76%	step rate ≥ 2.2 step/s		trouser Pocket, arm swing, in-hand
Biking	86.41%.	outdoor speed ≥ 11 Km/h	duration < 1 minute; speed < 8 Km/h	backpack, shirt pocket, trouser pocket
Driving	83.57%	speed ≥ 48 Km/h	passenger seat, glove compartment	cup holder, dash board, shirt pocket, trouser pocket

3 System setup guide

3.1 Hardware description

This section describes the hardware components required to develop a sensor-based application.

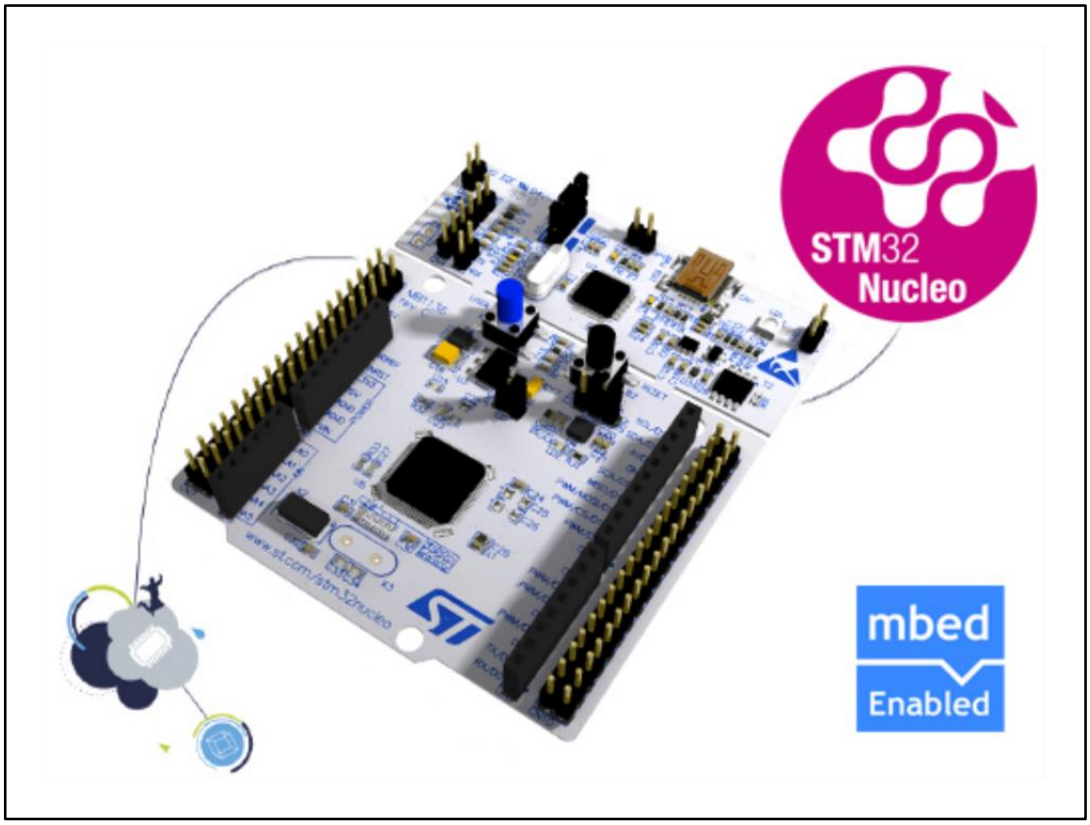
3.1.1 STM32 Nucleo platform

The STM32 Nucleo boards represent an affordable and flexible solution for users to implement new ideas and build prototypes with any STM32 microcontroller lines. 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.

The STM32 Nucleo board does not require any separate probes as it integrates the ST-LINK/V2-1 debugger/programmer, and it is bundled with the STM32 comprehensive software HAL library together with various packaged software examples.

Information about the STM32 Nucleo boards is available on www.st.com at: *http://www.st.com/stm32nucleo*

Figure 14: STM32 Nucleo board

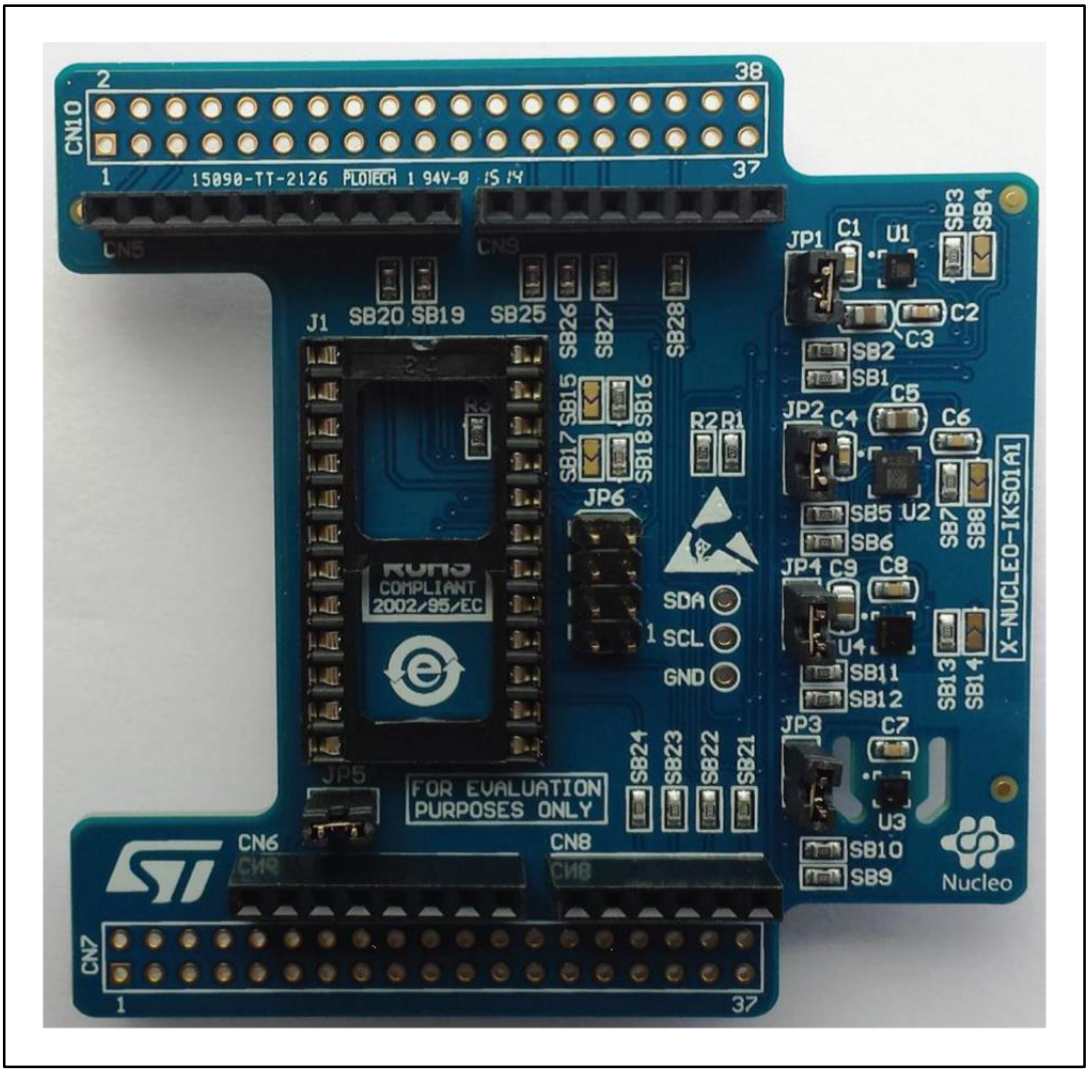


3.1.2 X-NUCLEO-IKS01A1 expansion board

The X-NUCLEO-IKS01A1 figured below is a sensor expansion board for use with the STM32 Nucleo system. It is also compatible with the Arduino UNO R3 connector layout, and is designed around the STMicroelectronics humidity (HTS221), pressure (LPS25HB)

and motion sensors (LIS3MDL and LSM6DS0). The X-NUCLEO-IKS01A1 interfaces with the STM32 MCU via an I²C pin, and the user can change the default I²C address and the device IRQ by changing one resistor on the evaluation board.

Figure 15: STM32 Nucleo board



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

The LSM6DS3 adapter board in the figure below can be plugged on top of the X-NUCLEO-IKS01A1 expansion board, as shown in *Figure 17: "Sensor expansion board and adapter connected to the STM32 Nucleo"*.

Figure 16: LSM6DS3 adapter board

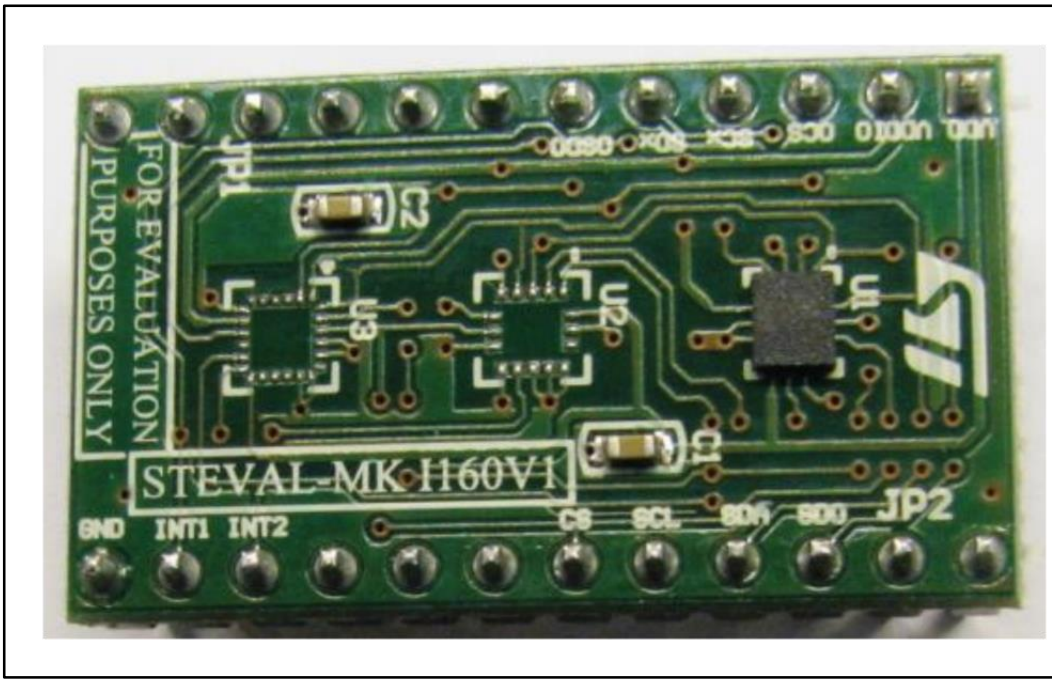
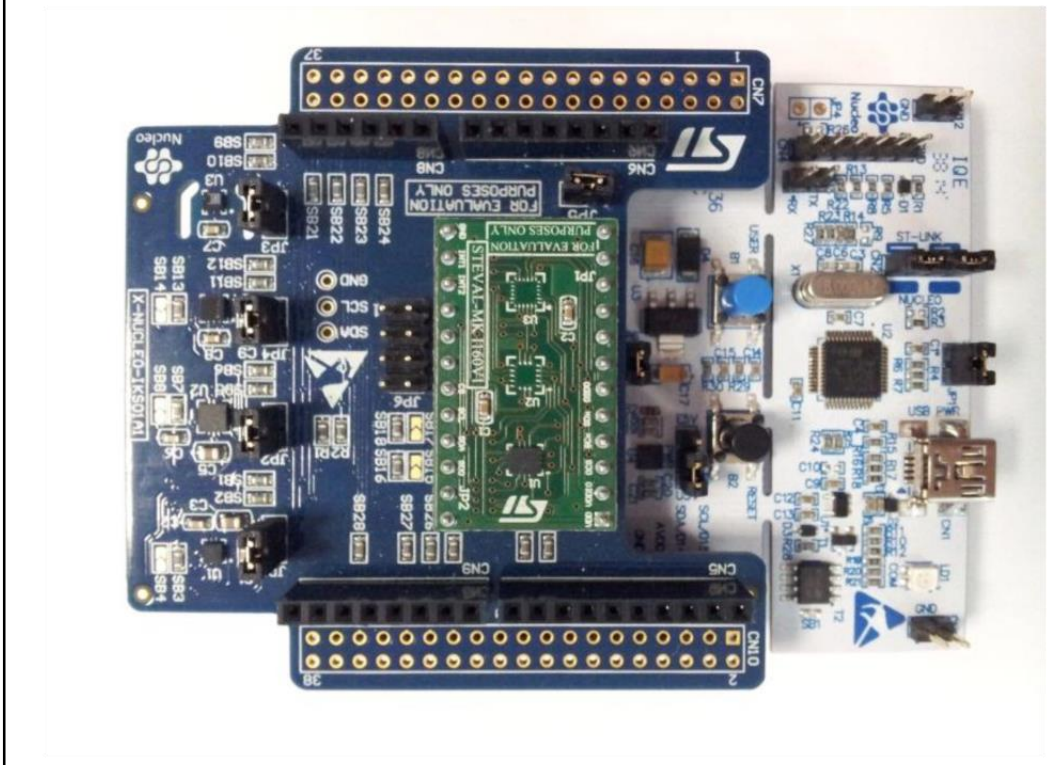


Figure 17: Sensor expansion board and adapter connected to the STM32 Nucleo



3.2 Software description

The following software components are required to set up a suitable development environment for creating applications using the osxMotionAR library for the STM32 Nucleo, equipped with the sensor expansion board:

- X-CUBE-MEMS1: an expansion for STM32Cube dedicated to sensor application development. The X-CUBE-MEMS1 firmware and related documentation is available on st.com.
- osxMotionAR: an add-on software package for X-CUBE-MEMS1 that provides activity recognition APIs. The osxMotionAR firmware and related documentation is available on st.com.
- Development tool-chain and compiler: The STM32Cube expansion software supports the following three IDEs:
  - IAR Embedded Workbench for ARM® (EWARM) toolchain + ST-LINK
  - µVision(MDK-ARM) toolchain + ST-LINK
  - System Workbench for STM32

3.3 Hardware setup

The following hardware components are required:

- One STM32 Nucleo development platform (order code: NUCLEO-F401RE or NUCLEO-L476RG)
- One sensor expansion board (order code: X-NUCLEO-IKS01A1) optionally mounted with the LSM6DS3 adapter board (order code: STEVAL-MKI160V1)
- One USB type A to Mini-B USB cable to connect the STM32 Nucleo to the PC
- One battery pack to run the application example

3.4 Software setup

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

3.4.1 Development tool-chains and compilers

Please select one of the integrated development environments supported by the STM32Cube expansion software, and read the system requirements and setup information provided by the selected IDE provider.

3.4.2 PC utility

The Sensors DataLog utility for PC has following minimum requirements:

- PC with Intel or AMD processor running one of the following Microsoft operating systems:
- Windows XP SP3
- Windows Vista
- Windows 7
- At least 128 MBs of RAM
- 2 X USB ports
- 40 MB of hard disk space

3.5 System setup

This section describes how to set up different hardware components before writing and executing an application on the STM32 Nucleo board with the sensor expansion board.

3.5.1 STM32 Nucleo and sensor expansion boards setup

The STM32 Nucleo board includes the ST-LINK/V2-1 debugger/programmer. The developer can download the relevant version of the ST-LINK/V2-1 USB driver by accessing the STSW-LINK008 or STSW-LINK009 on www.st.com (according to which MS Windows OS is used).

The sensor expansion board X-NUCLEO-IKS01A1 can be easily connected to the STM32 Nucleo motherboard through the Arduino UNO R3 extension connector (see *Figure 17: "Sensor expansion board and adapter connected to the STM32 Nucleo"*). The LSM6DS3 adapter board (STEVAL-MKI160V1) is plugged on top of the expansion board. The sensor expansion boards are capable of interfacing with the external STM32 microcontroller on the STM32 Nucleo via an inter-integrated circuit (I^2C) transport layer.

3.5.2 Sensors_DataLog GUI setup

The Sensors_DataLog GUI included in the X-CUBE-MEMS1 software package is a graphical user interface that can be used to obtain activity data previously stored on MCU. Please see for a description of the how this PC utility works.

This utility retrieves data from the connected STM32 MCU flash memory and stores it in a table format which can be exported for offline analysis.

In order to use the Sensors_DataLog GUI, the user has to correctly set up the hardware and software.

The utility can be launched by simply double-clicking on the Sensors_DataLog.exe file, located in the "Utilities\PC_software\Sensors_DataLog" folder.

3.5.3 osxMotionAR installer setup

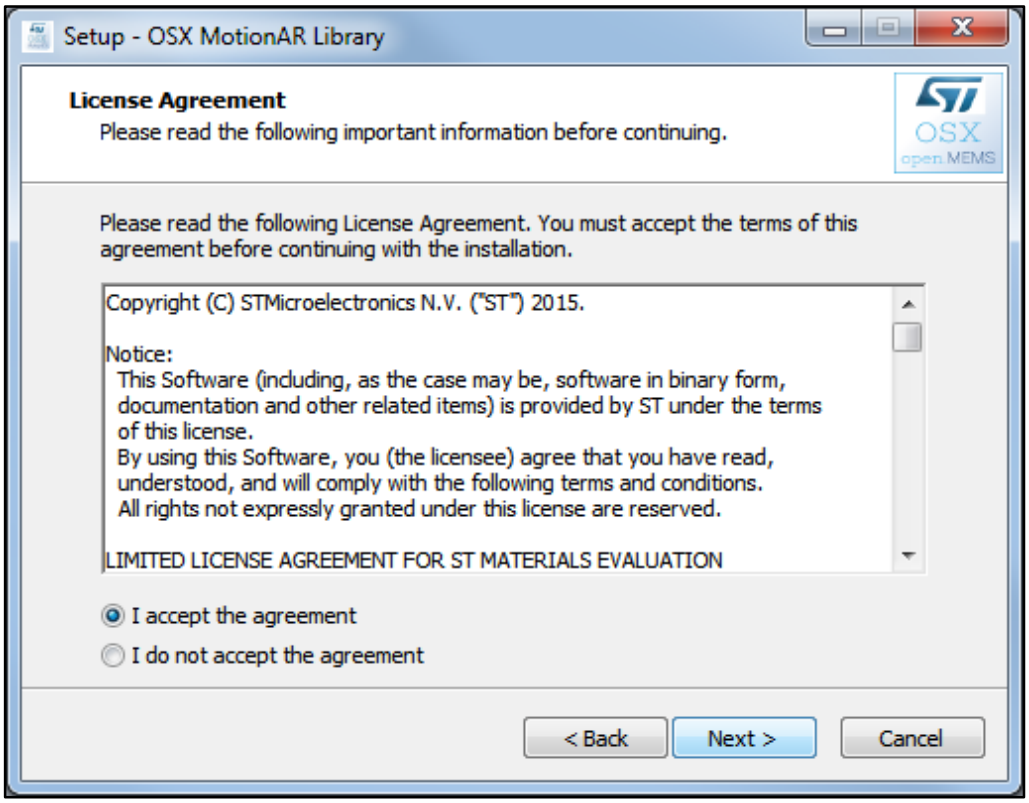
The osxMotionAR software package is equipped with a Windows installer (osxMotionAR_Setup_vXXX.exe), which guides the user through the software package installation. The installer also includes the OSX License Wizard tool, which allows the user to automatically obtain a valid node-locked license for their STM32 Nucleo board. The installer for this wizard is described in *Section 3.5.4: "osxMotionAR license wizard"*.

1 Before running the installer, download the latest release of the X-CUBEMEMS1 software package and unzip this package in the relative workspace. For example, if the X-CUBE-MEMS1 package is located in "C:\workspace\STM32CubeExpansion_MEMS1_VX.X.X", the installer will display welcome window shown in the figure below.

Figure 18: osxMotionAR installer screenshot 1

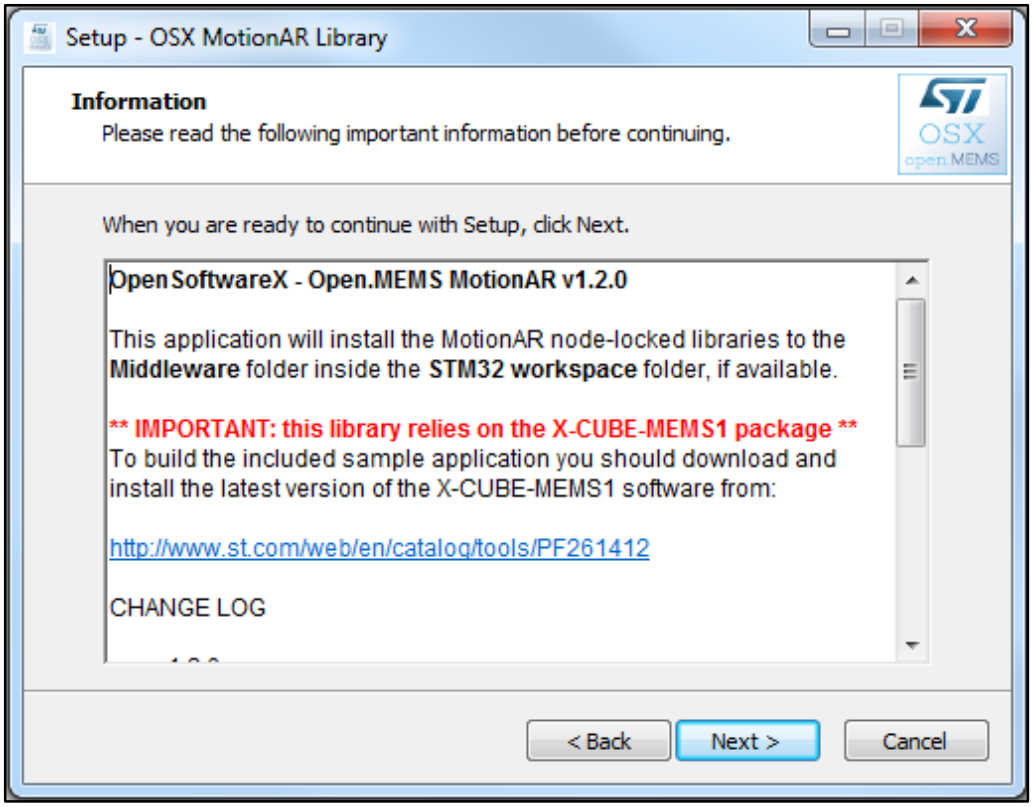
2 The license agreement must be accepted, as shown in the figure below.

Figure 19: osxMotionAR installer screenshot 2



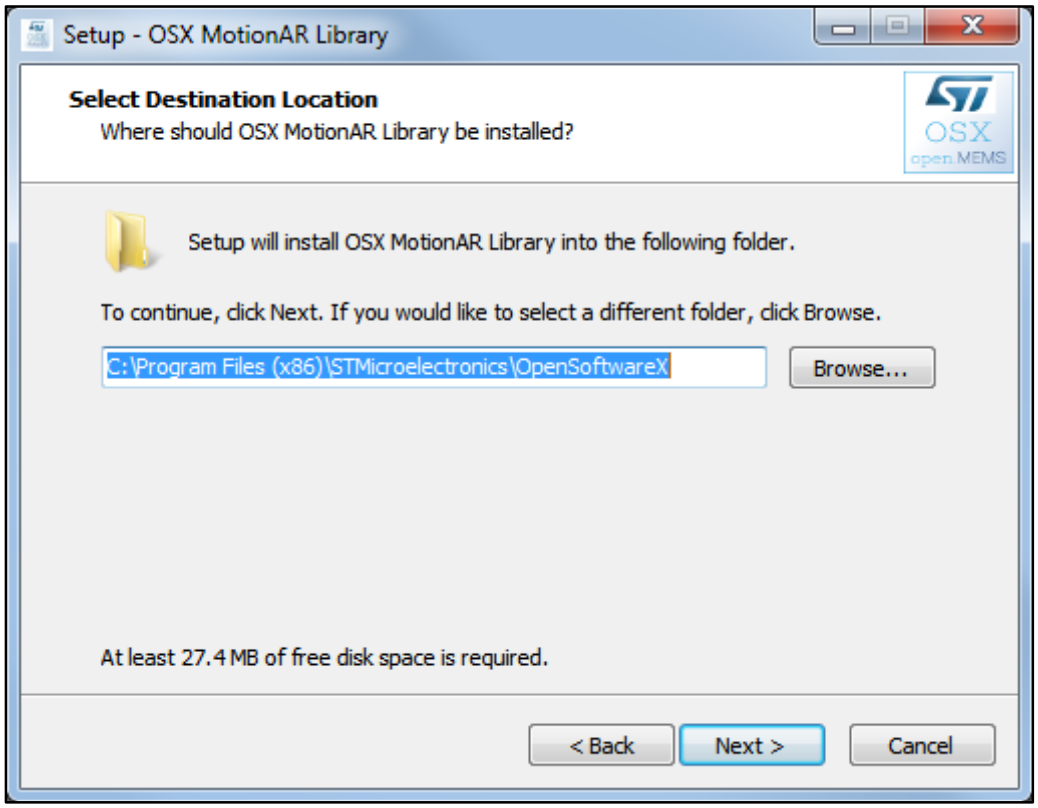
3 The installer provides information such as the requirements of the latest release of the X-CUBE-MEMS1 software package.

Figure 20: osxMotionAR installer screenshot 3



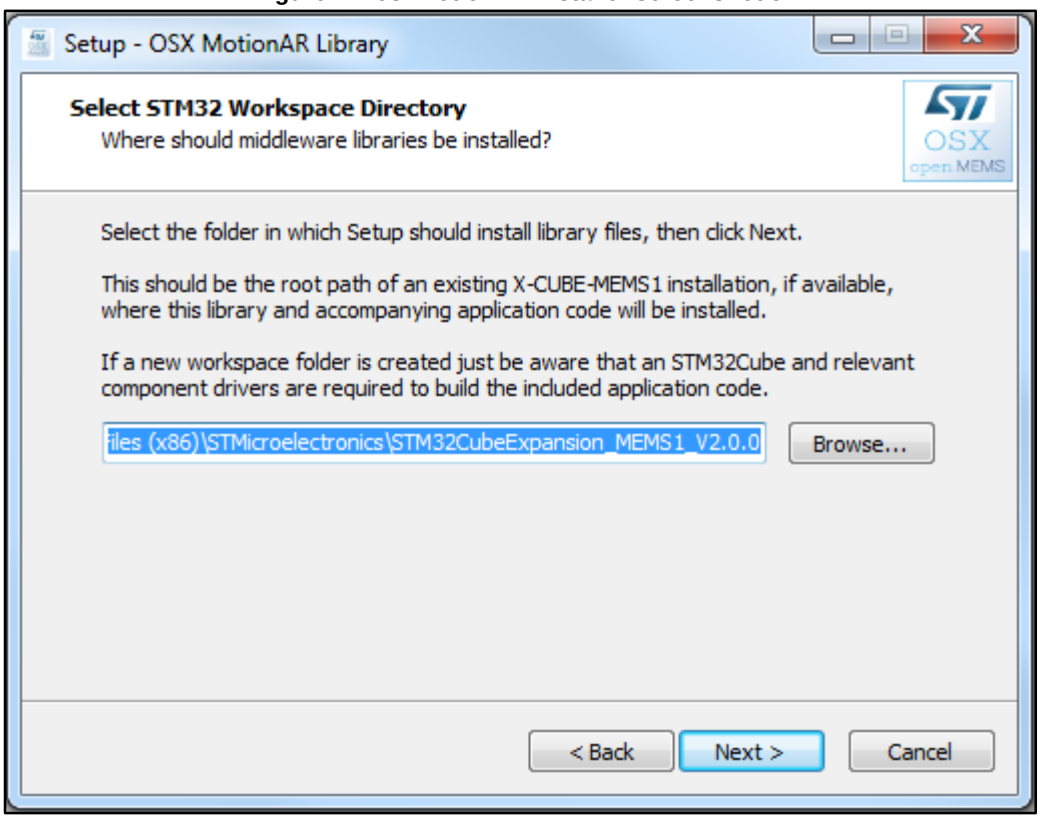
4 The installer then asks for the destination location of the osxMotionAR library.

Figure 21: osxMotionAR installer screenshot 4



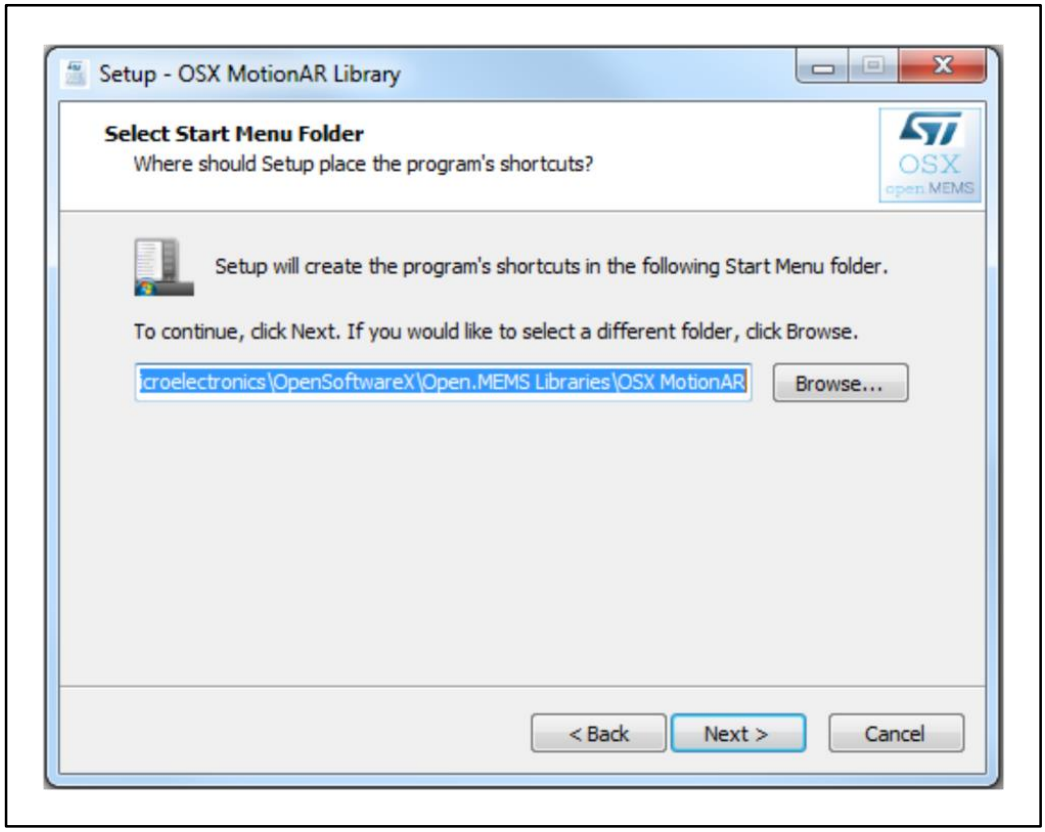
5 The user must then provide the path to the X-CUBE-MEMS1 package. It is imperative to insert the correct path to the X-CUBEMEMS1 package, otherwise the sample application will not compile. It is the path where the folders "Documentation", "Projects", "Drivers", "Utilities", etc.., contained in the X-CUBE-MEMS1 software package are located. In this example, it is "C:\Program Files (x86)\STMicroelectronics\STM32CubeExpansion_MEMS1_V2.0.0".

Figure 22: osxMotionAR installer screenshot 5



6 The user must then provide the path to the Start menu folder where the shortcuts are installed. This completes the osxMotionAR installation.

Figure 23: osxMotionAR installer screenshot 6



3.5.4 osxMotionAR license wizard

The osxMotionAR engine is provided as a node-locked library which allows derivative firmware images to run on a specific STM32 Nucleo device only. License activation codes must to be requested to ST and included in the project (and becomes part of the build process) prior to attempting its usage. The resulting firmware binary image will therefore be node-locked.

The License wizard allows the user to automatically obtain a valid node-locked license for their STM32 Nucleo board. Before running this tool, connect the STM32 Nucleo to be linked to the node-locked license via USB cable.

The user will receive an email with the license to be included in the Middlewares\ST\STM32_OSX_MotionAR_Library folder of the workspace.

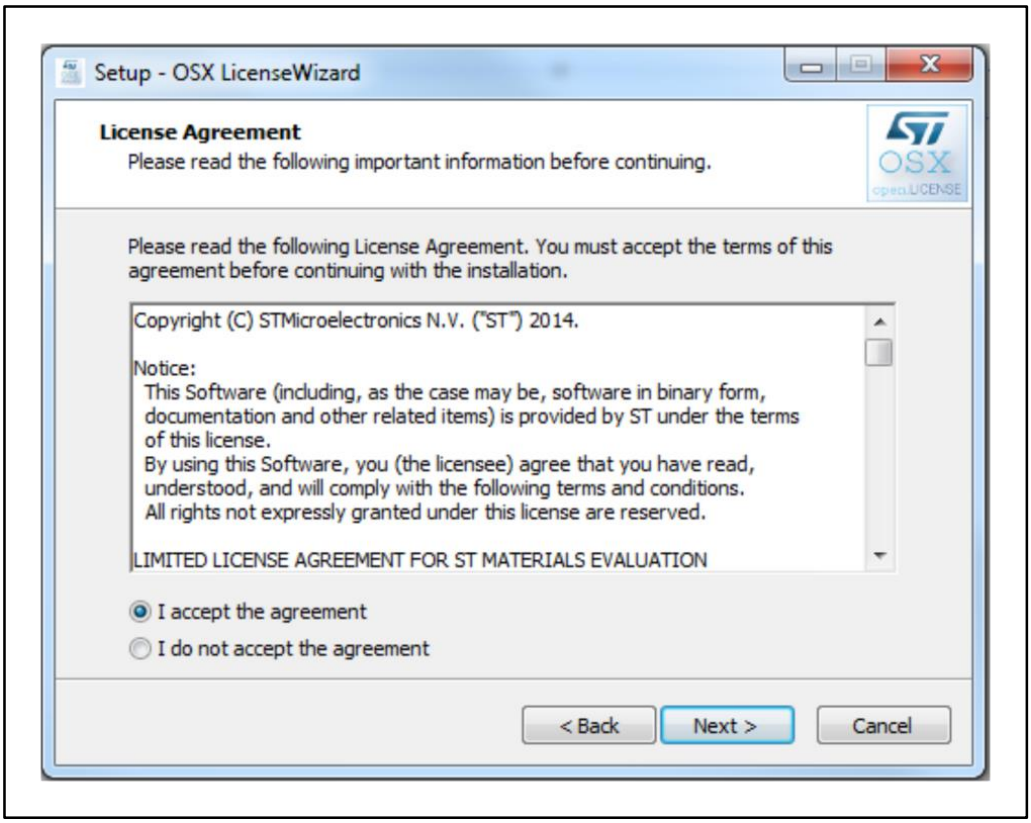
The steps below describe the licensing procedure in detail.

1 The figure below shows the welcome screen.

Figure 24: OSX License Wizard screenshot 1

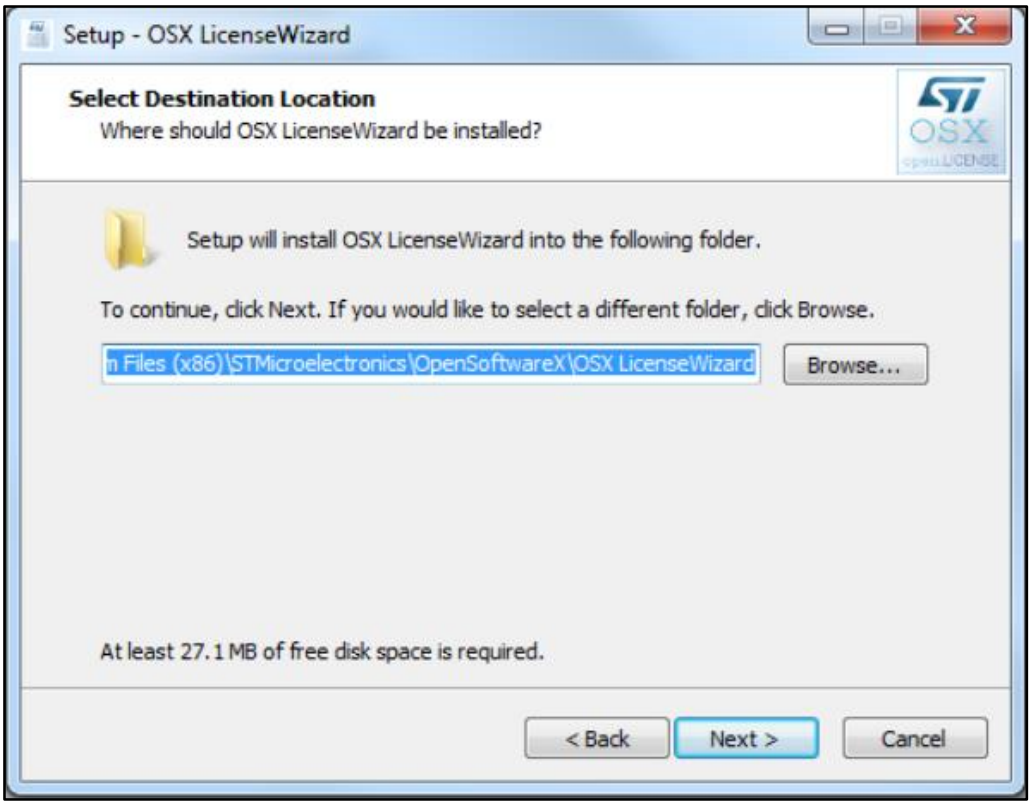
2 The user needs to accept the license agreement to proceed with the installation.

Figure 25: OSX License Wizard screenshot 2



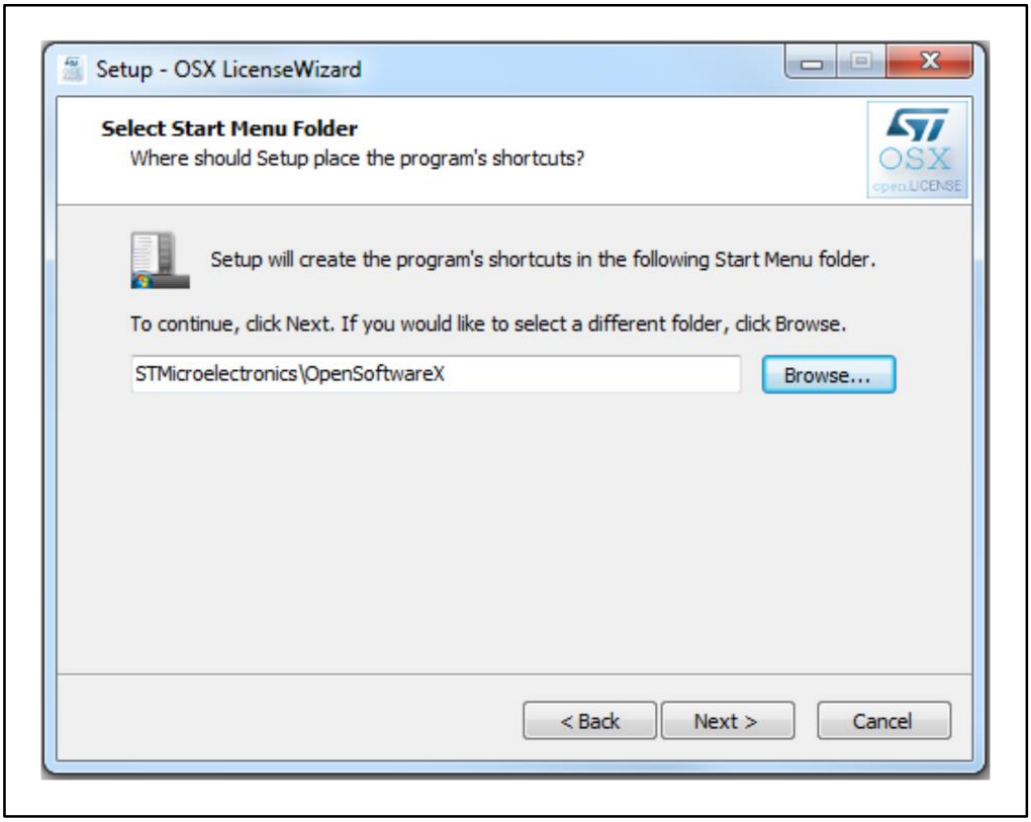
3 The wizard then asks for the destination location of the OSX License Wizard.

Figure 26: OSX License Wizard screenshot 3



4 The user must then provide the path to the Start menu folder where the shortcuts are installed. This step completes the OSX License Wizard software installation.

Figure 27: OSX License Wizard screenshot 4

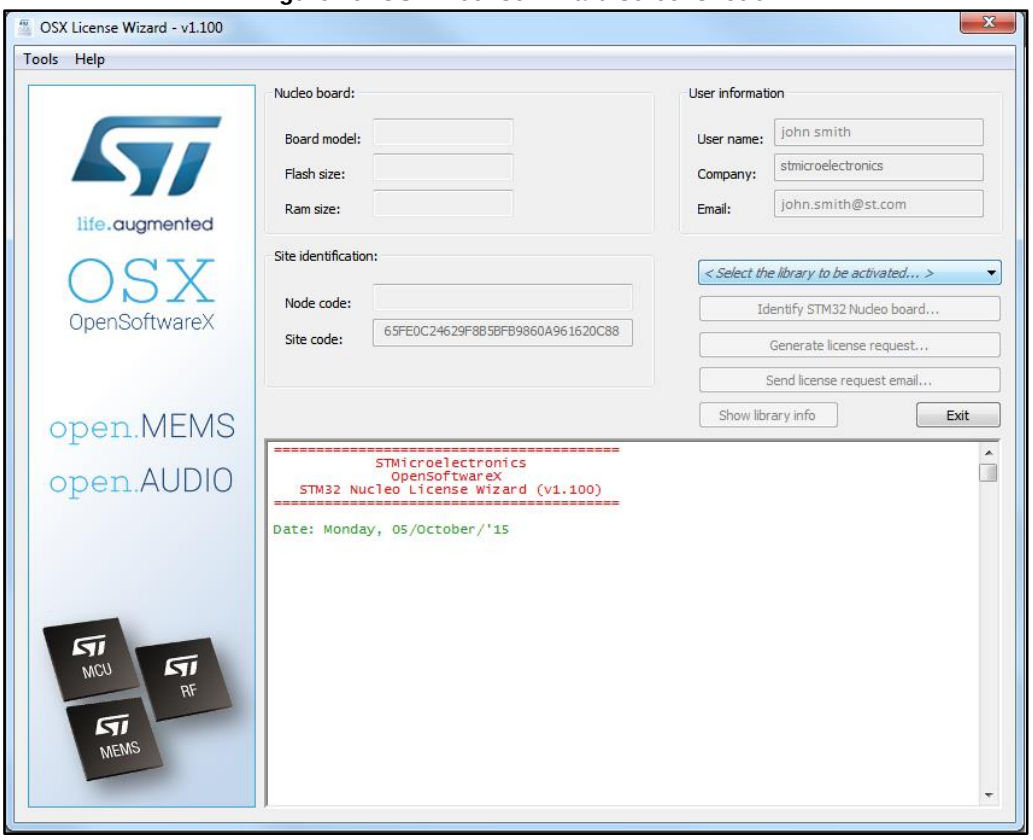


5 The OSX License Wizard needs to access some information on the STM32 Nucleo board in order to create the request for the node-locked license. For this reason, the OSX License Wizard requires the installation of some additional packages ("Microsoft VS 2013 redistributable" and "STM32 ST-LINK Utility"), if the user has not already done so

Figure 28: OSX License Wizard screenshot 5

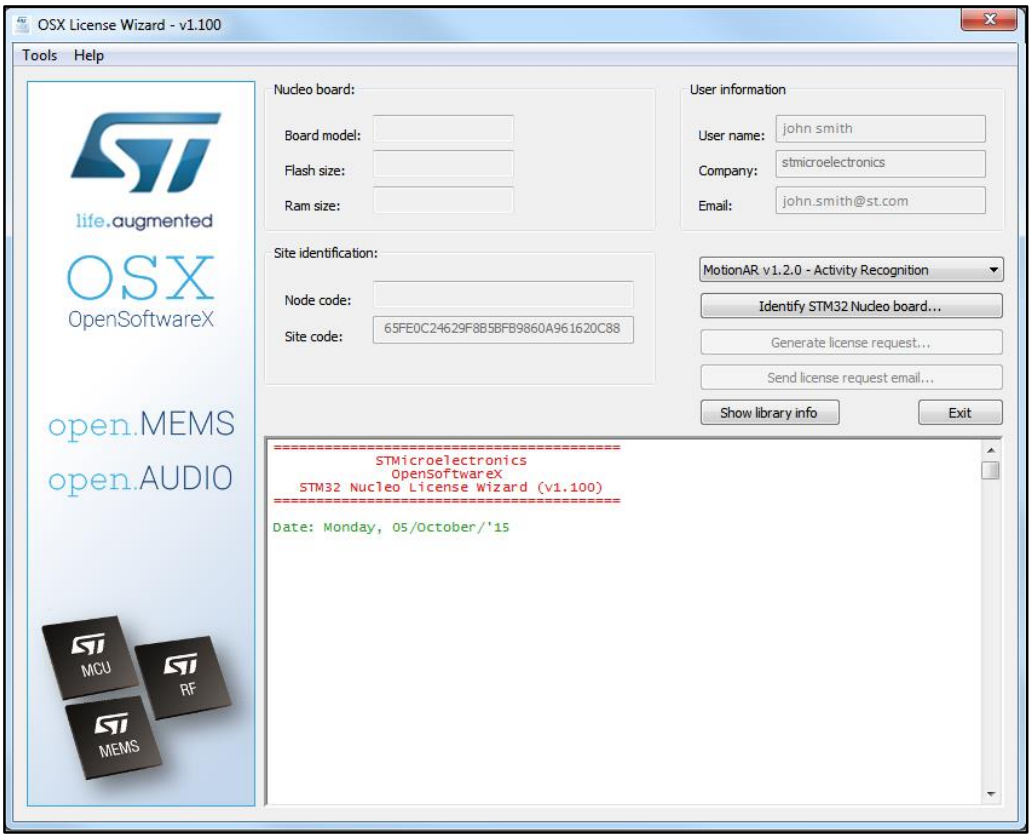
6 Before launching the OSX License Wizard, connect the STM32 Nucleo board to be linked to the node-locked license via USB cable. First, select the library to be activated as per the figure below.

Figure 29: OSX License Wizard screenshot 6



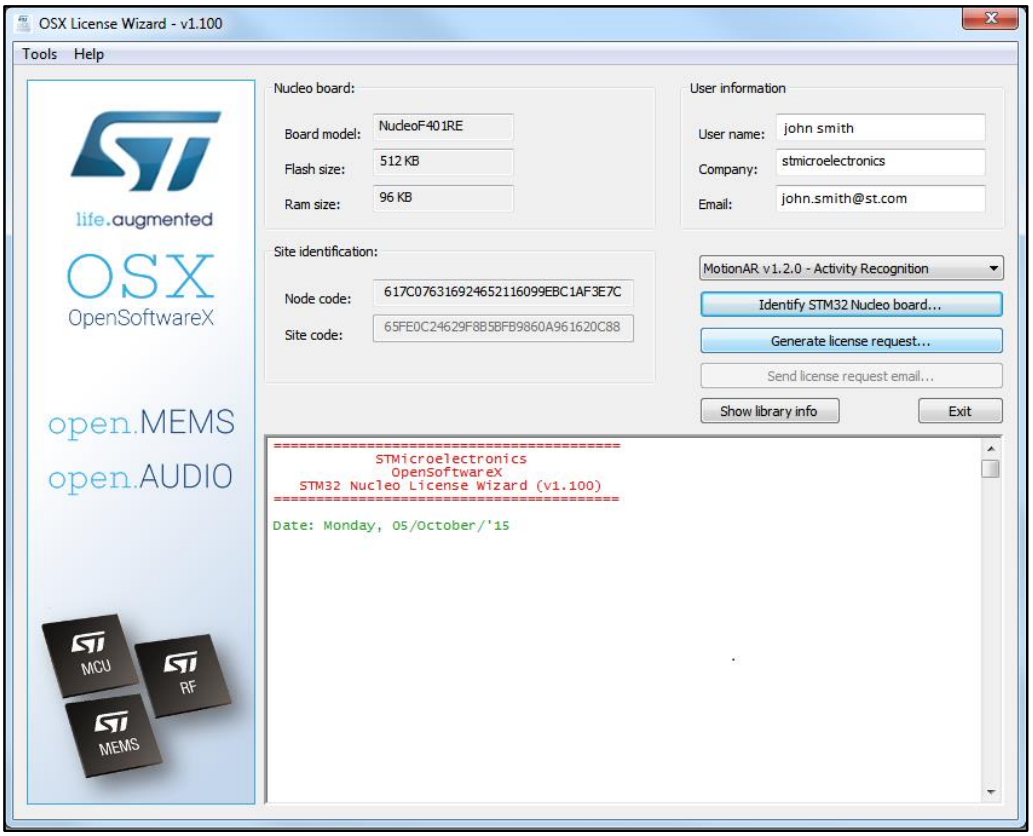
7 Next click on the "Identify STM32 Nucleo board" button.

Figure 30: OSX License Wizard screenshot 7



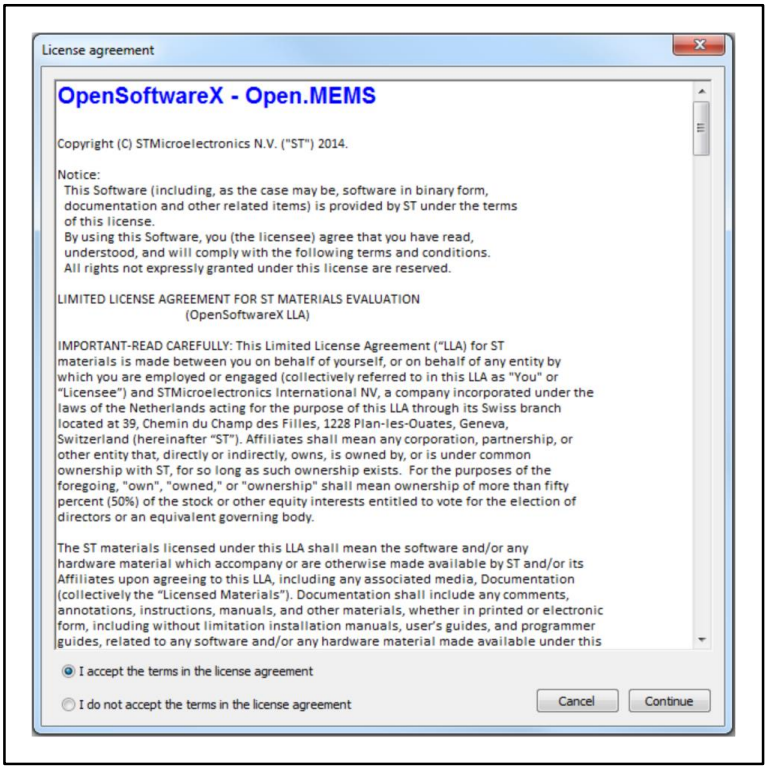
8 At this point, information about the connected STM32 Nucleo board is available to the user together with a node code associated with the same board. Complete the form, editing the "User name", "Company" and "Email" and click on "Generate license request".

Figure 31: OSX License Wizard screenshot 8



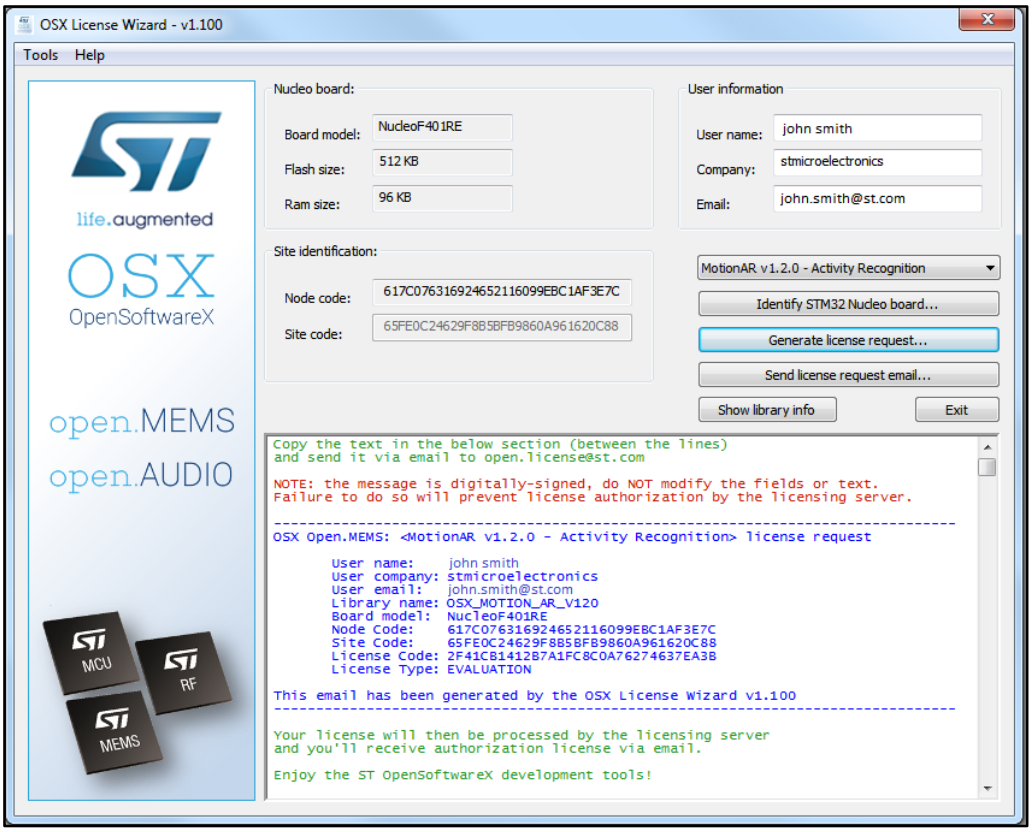
9 A new window concerning the license agreement will appear and user can accept the conditions by clicking the Continue button to proceed.

Figure 32: OSX License Wizard screenshot 9



10 Click the "Send license request email" button shown in the figure below.

Figure 33: OSX License Wizard screenshot 10



11 Once the license activation codes have been received, edit the content of the osx_license.h file with the license number. You must also delete or comment the "#error...." line, found in the Middlewares\ST\STM32_OSX_MotionAR_Library folder of your workspace.

4 Appendix

4.1 Production license

Additionally a *production* (i.e. node-free) license can be obtained after ST specific authorization. This license is usable on any STM32-based platform, provided the Cortex core compatibility.

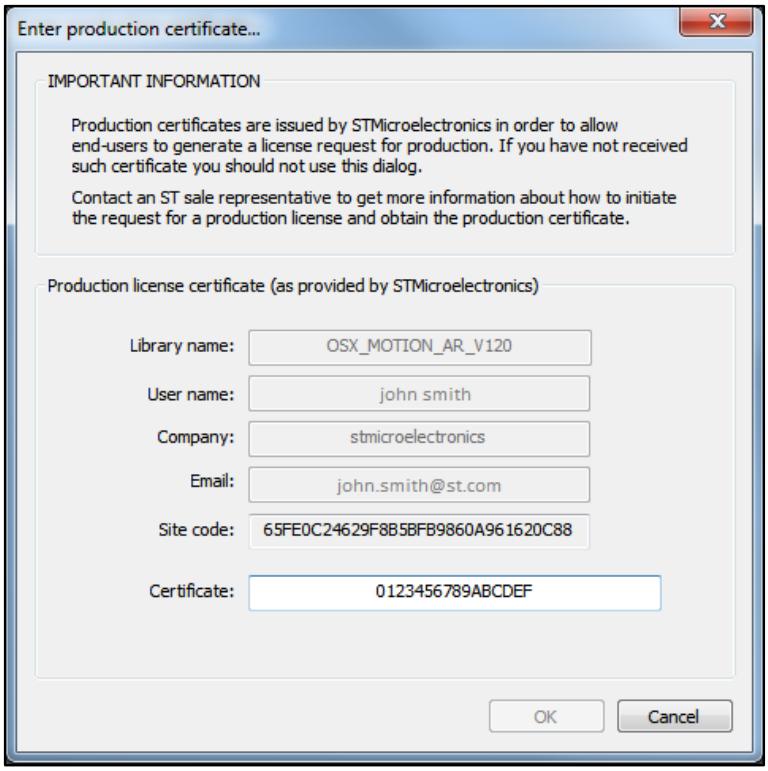
The request of this license type must be first authorized by ST through a user-specific certificate. Ownership of a valid certificate will enable the end-user to initiate a production license request – the request will then be automatically processed by the server.

ST will authorized end-users to request a *production* license only after a two-parties License User Agreement (LUA) has been signed.

After the LUA signing and a valid certificate has been obtained, proceeds as follows:

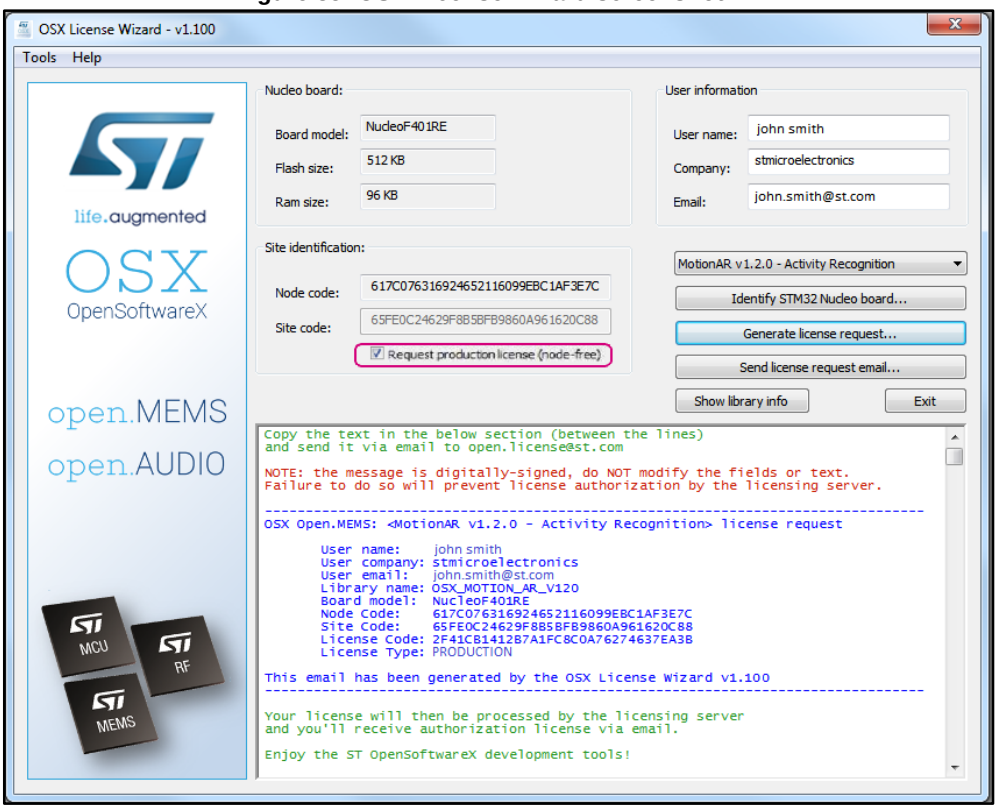
1 In the tools menu, select “Enter Production Certificate…” and insert a valid code in the relative window, as shown below.

Figure 34: OSX License Wizard screenshot 11



2 After a valid certificate has been inserted, the node-free license request flag becomes visible and the production license can be requested. Once received, the new license can replace the old node-locked license in the *osx_license.h* file.

Figure 35: OSX License Wizard screenshot 12



5 Acronyms and abbreviations

Table 4: Acronyms

Acronym	Description
API	Application Programming Interface
BSP	Board Support Package
ENU	x-East, y-North, z-Up
GUI	Graphical User Interface
HAL	Hardware Abstraction Layer
IDE	Integrated Development Environments
LUA	Licence User Agreement
NED	x-North, y-East, z-Down
SEU	x-South, y-East, z-Up
SDK	Software Development Kit

6 References

- *UM1859: Getting started with the X-CUBE-MEMS1 motion MEMS and environmental sensor software expansion for STM32Cube*
- *DB2613: Real-time activity-recognition software expansion for STM32Cube*
- *UM1724: STM32 Nucleo boards*

7 Revision history

Table 5: Document revision history

Date	Revision	Changes
05-Aug-2015	1	First release.
21-Jan-2016	2	Added section *Section 1.5.2: "PC GUI driven mode"* Added section *Section 4: "Appendix"* with subsection *Section 4.1: "Production license"* Added references to NUCLEO-L476RG board compatibility Minor text edits throughout document
25-Jan-2016	3	Minor edits

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.