UM1866
User manual

Getting started with the osxMotionFX fusion and compass library for X-CUBE-MEMS1 expansion for STM32Cube

Introduction

osxMotionFX is an add-on software package for X-CUBE-MEMS1. The software runs on the STM32 and includes drivers that recognize the sensors and provide real-time motion-sensor data fusion. The add-on is built on STM32Cube software technology to ease portability across different STM32 microcontrollers. The software comes with examples of implementation of the drivers running on the X-NUCLEO-IKS01A1 or X-NUCLEO-IKS01A2, when connected to a NUCLEO-F401RE or NUCLEO-L476RG.

Contents

List of tables

List of figures

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

1.1 Overview

The osxMotionFX library expands the functionality of the X-CUBE-MEMS1 software.

The key features of the package are:

- osxMotionFX (iNEMOEngine PRO) real-time motion-sensor data fusion (under OPEN.MEMS license)
- Complete software to build applications using temperature and humidity sensor (HTS221 for both X-NUCLEO-IKS01A1 and X-NUCLEO-IKS01A2), pressure sensor (LPS25HB for X-NUCLEO-IKS01A1 and LPS22HB for X-NUCLEO-IKS01A2) and motion sensors (LIS3MDL and LSM6DS0 for X-NUCLEO-IKS01A1 and LSM303AGR and LSM6DSL for X-NUCLEO-IKS01A2)
- Gyroscope bias and magnetometer calibration routine
- Easy portability across different MCU families, thanks to STM32Cube
- Sample application to transmit real-time both sensor data and sensor fusion data to a PC
- Sample implementation available on board X-NUCLEO-IKS01A1 or X-NUCLEO-IKS01A2 when connected to NUCLEO-F401RE or NUCLEO-L476RG

The osxMotionFX (iNEMOEngine PRO) filtering and predictive software uses advanced algorithms to intelligently integrate outputs from multiple MEMS sensors, independent of environmental conditions, for optimum performance.

The software comes with a sample application running on an X-NUCLEO-IKS01A1 or X-NUCLEO-IKS01A2 expansion boards connected to a NUCLEO-F401RE or a NUCLEO-L476RG development board.

1.2 Architecture

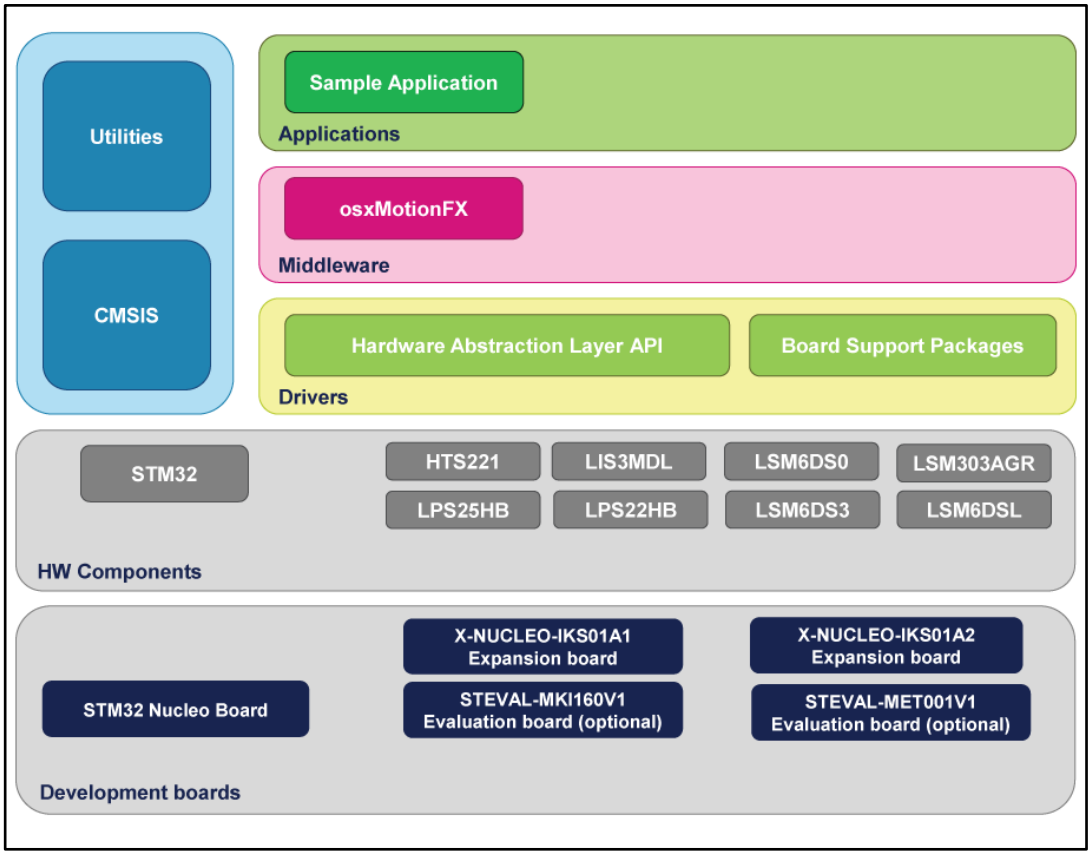
This software is an expansion for X-CUBE-MEMS1 software.

The package extends STM32Cube by providing a middleware component for the osxMotionFX library.

The application software uses the layers below to access the sensor expansion board.

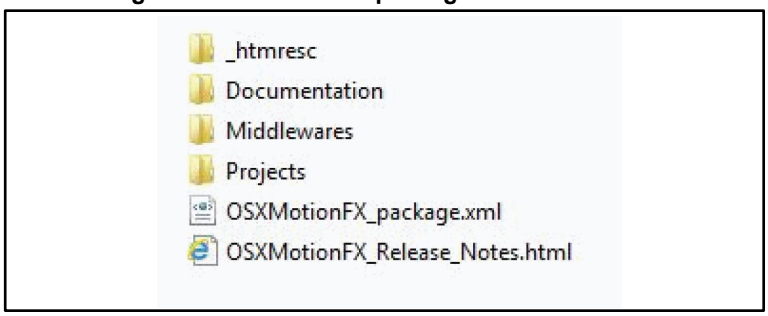
- STM32Cube HAL layer: provides a generic multi-instance simple set of APIs (application programming interfaces) to interact with the upper layers (application, libraries and stacks). The generic and extension APIs are based on generic architecture so higher layers like the middleware layer can function without requiring specific microcontroller unit (MCU) hardware information. This structure improves library code reusability and guarantees easy portability to other devices.
- Board support package (BSP) layer: supports the peripherals (except the MCU) on the STM32 Nucleo board. This limited set of APIs provides a programming interface for certain board specific peripherals like the LED and user button. It 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 and provides support for initializing and reading sensor data.

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



1.3 Folders structure

Figure 2: osxMotionFX package folder structure



The following folders are included in the package:

- **Documentation**: contains a compiled HTML file generated from the source code which details the software components and APIs.
- **Middlewares**: contains the osxMotionFX library binary code, documentation and license information.
- **Projects**: contains a sample application used to access sensor and sensor fusion data, provided for the NUCLEO-F401RE and the NUCLEO-L476RG platforms with three development environments (IAR Embedded Workbench for ARM, RealView Microcontroller Development Kit (MDK-ARM), System Workbench for STM32).

1.4 APIs

Detailed function and parameter descriptions of the osxMotionFX APIs and compass calibration libraries can be found in the OSXMotionFX_Package.chm compiled HTML file located in the package Documentation folder.

The osxMotionFX engine 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 to be requested from ST and included in the project (and will 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_MotionFX_Library folder.

1.5 Sample application description

The Projects folder contains an osxMotionFX fusion and compass calibration sample application using the X-NUCLEO-IKS01A1 expansion board with the NUCLEO-F401RE or NUCLEO-L476RG development board.

In this application, sensor and sensor fusion data is streamed from the STM32 Nucleo board to PC in real time via the UART interface and can be viewed using the Unicleo GUI, a PC-based application developed by STMicroelectronics and provided in binary code with a separated package, or the Sensors_DataLog PC application included in the X-CUBE-MEMS1 package (see).

When the board is powered on, the compass is not calibrated and LED2 (green LED) on the STM32 Nucleo board is consequently off (see *Section 1.5.2.2: "Magnometer calibration"*). You can only calibrate the compass when the sensor fusion is activated (press Start Sensor Fusion on the PC GUI). Following calibration, LED2 turns on and the compass points north.

The default storage location of calibration data is in the RAM memory so that it is preserved when the board is reset, but you can also store calibration data in Flash memory. To do this, compile the sample DataLogFusion application with the `OSXMOTIONFX_STORE_CALIB_FLASH` define in the project.

By default, the application uses fusion 9X (accelerometer + gyroscope + magnetometer), but you can switch from fusion 9X to fusion 6X (accelerometer + gyroscope) at any time by clicking the appropriate button on the Sensors_DataLog GUI. When using fusion 6X, the compass calibration is no longer required.

1.5.1 6-axis and 9-axis sensor fusion osxMotionFX library

1.5.1.1 6-axis and 9-axis sensor fusion osxMotionFX library

The osxMotionFX library implements sensor fusion algorithm for estimation of 3D orientation in space. It uses a digital filter based on the Kalman theory to fuse data from several sensors and compensate for limitations of single sensors. For instance, gyroscope data can drift, which impacts the orientation estimation. This can be fixed by using the magnetometer to provide absolute orientation information. Similarly, the magnetometer does not have a very high bandwidth and suffers of magnetic disturbance; but these weaknesses can be compensated with a gyroscope.

9-axis sensor fusion utilizes data from accelerometer, gyroscope and magnetometer (A+G+M) and provides absolute orientation in 3D space including heading; i.e., the magnetic North direction.

6-axis sensor fusion uses accelerometer and gyroscope (A+G) data only. It has lower computational requirements, but does not provide information about the absolute orientation of the device. 6-axis sensor fusion is good for fast movements (e.g., for gaming) and when absolute orientation is not necessary.

1.5.1.2 osxMotionFX library operation

The osxMotionFX library integrates 6 and 9-axis sensor fusion algorithms in one library; they can even run simultaneously. The osxMotionFX library implements the following very important internal functions associated with sensor fusion computation:

1. osx_MotionFX_propagate is a prediction function used to estimate the orientation in 3D space; gyroscope data is given more weight in this phase.
2. The osx_MotionFX_update is the corrective function which correct the predicted value when necessary; accelerometer and magnetometer data are given more weight in this phase.

The osx_MotionFX_update function can be called every time the osx_MotionFX_propagate is invoked, or less in systems that have less computation power. The osx_MotionFX_update function takes approximately three times more MCU computation time than the osx_MotionFX_propagate function.

The typical operation sequence of the osxMotionFX library is:

1. Initialize sensors
2. Initialize the sensor fusion library and calibration library (osx_MotionFX_initialize and osx_MotionFX_compass_init)
3. Setup parameters of the library in a Knobs structure (osx_MotionFX_getKnobs and osx_MotionFX_setKnobs)
4. Start 6-axis or 9-axis engine (osx_MotionFX_enable_6X or osx_MotionFX_enable_9X)
5. At a specific output sensor fusion data rate (e.g., using an MCU timer or data ready signal from one of the sensors), regularly perform:
   - read and convert data from sensors
   - call osx_MotionFX_propagate
   - call osx_MotionFX_update

Both osx_MotionFX_propagate and osx_MotionFX_update functions take sensor data for their input and they both also provide the Sensor Fusion library output data.

1.5.1.3 osxMotionFX library parameters

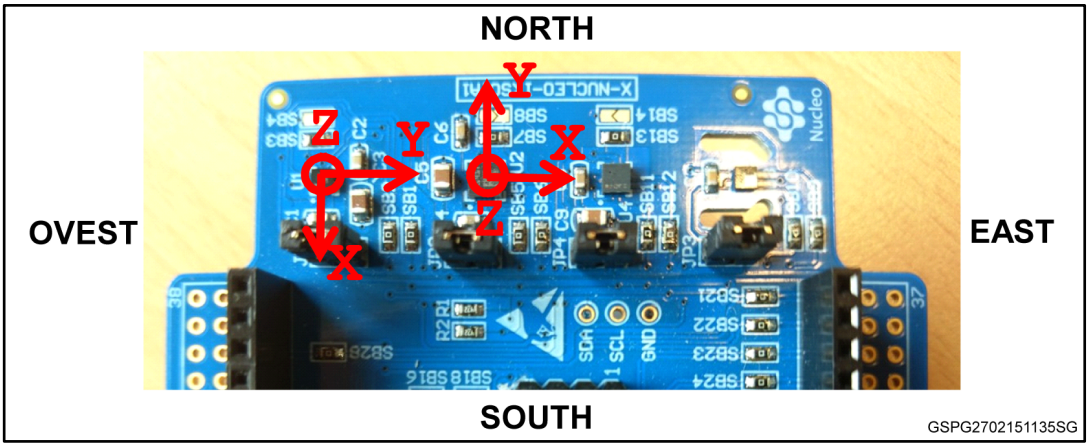
As she library osxMFX_knobs parameter structure and the osxMFX_output output structure contain several fields that are differentiated for 6-axis and 9-axis operation, so all you need to do is call the desired osx_MotionFX_enable_6X or osx_MotionFX_enable_9X engine (magnetometer data is not required for 6-axis sensor fusion operation).

The osxMotionFX library is parameterized using a Knobs structure:

```
typedef struct
{
  float ATime;                              /*  merge rate to the accel                  */
  float MTime;                              /*  merge rate to the mag                    */
  float FrTime;                             /*  merge rate to the accel when external
accelerations occurs  */
  unsigned char LMode;                      /*  gyro bias learn mode,
                                                                        1-static learning
                                                                        2-dynamic learning
*/
  float gbias_mag_th_sc_6X;          /*  scaler for the gyro bias mag threshold nominal    */
  float gbias_acc_th_sc_6X;          /*  scaler for the gyro bias mag threshold nominal    */
  float gbias_gyro_th_sc_6X;         /*  scaler for the gyro bias mag threshold nominal    */
  float gbias_mag_th_sc_9X;          /*  scaler for the gyro bias mag threshold nominal    */
  float gbias_acc_th_sc_9X;          /*  scaler for the gyro bias mag threshold nominal    */
  float gbias_gyro_th_sc_9X;         /*  scaler for the gyro bias mag threshold nominal    */
  unsigned char modx;                           /*  setting to indicate the decimation,
                                                                        set to 1 in
smartphone/tablet
                                                                  set to >=1 in embedded solutions */
  char acc_orientation[QNUM_AXES];           /*  accelerometer data orientation           */
  char gyro_orientation[QNUM_AXES];          /*  gyroscope data orientation               */
  char mag_orientation[QNUM_AXES];           /*  magnetometer data orientation            */
  osxMFX_Engine_Output_Ref_Sys output_type;  /*  0: NED, 1: ENU                           */
  int start_automatic_gbias_calculation;
} osxMFX_knobs;
```

- *ATime*, *MTime* and *FrTime* represent the weighting stability of sensors for prediction (trust factor), from 0 to 1; we recommend maintaining the default values
- *LMode* represents the gyroscope bias learning mode; the library automatically tracks and calibrates gyro zero-rate bias drift; the possible values of this parameter are:
  - LMode = 0 – learning off; use this mode if the gyro is already calibrated
  - LMode = 1 – static learning; learning is only performed when the system is not moving
  - LMode = 2 – dynamic learning: learning is also performed when the system is moving
- *gbias_xxx_th_sc* represent the thresholds below which the gbias algorithm automatically starts. These values should be established through testing (they are different for different part numbers). Values in example project are usually correct
- *modx* represents the decimation of osx_MotionFX_update call frequency. See resources optimization paragraph below for more details
- *acc_orientation*, *gyro_orientation*, *mag_orientation* represent the three-character arrays describing the orientation of the sensor X, Y and Z sensor axes. The possible values are: n (north), e (east), s (south), w (west), u (up) and d (down).
- *output_type* represents the Sensor Fusion library output orientation: 0 means NED, 1 means ENU
- *start_automatic_gbias_calculation* represents a flag that restarts gyroscope bias calibration when set to 1

Figure 3: Example X, Y and Z axis values



As shown in *Figure 3: "Example X, Y and Z axis values"*, the first sensor (magnetometer) is oriented SEU (X - South, Y - East, Z - Up), while the second (accelerometer + gyroscope) is ENU (X - East, Y - North, Z - Up).

The osx_MotionFX_propagate and the osx_MotionFX_update functions expect input from sensors in the osxMFX_input structure:

```
typedef struct
{
  float mag[NUM_AXES];              /*  calibrated mag [uT]/50    */
  float acc[NUM_AXES];              /*  acc [g]                   */
  float gyro[NUM_AXES];             /*  gyro [dps]                */
} osxMFX_input;
```

- *mag* represents magnetometer data after calibration in µT/50
- *acc* represents accelerometer data in g
- *gyro* represents gyroscope data in dps

The osx_MotionFX_propagate and the osx_MotionFX_update functions provide output of the sensor fusion in the osxMFX_output structure:

```
typedef struct
{
  float rotation_9X[NUM_AXES];              /*  9 axes yaw, pitch and roll     */
  float quaternion_9X[QNUM_AXES];           /*  9 axes quaternion              */
  float gravity_9X[NUM_AXES];               /*  9 axes device frame gravity    */
  float linear_acceleration_9X[NUM_AXES];   /*  9 axes device frame lin accel  */
  float heading_9X;                         /*  9 axes heading                 */
… similarly for 6-axis SF

} osxMFX_output;
```

- *rotation* represents the orientation of the system in three angle format: yaw, pitch and roll
- *quaternion* represents orientation of the system in four number format; this format gives the same information as rotation; it has advantages for computation and is therefore usually used by other algorithms (which can follow the sensor fusion)
- *gravity* represents the static acceleration (i.e., Earth's gravity) vector extracted from acceleration data

- *linear_acceleration* represents the dynamic acceleration (i.e., movement) vector extracted from acceleration data
- *heading* represents the direction of the magnetic North

1.5.1.4 osxMotionFX library output data rate

It is important to set up the output data rate of the sensors properly.

We recommend 100 Hz for the sensor fusion library output data rate, so the output data rate for:

- the gyroscope and the accelerometer should be greater than 100 Hz.
- the magnetometer can be lower - 20/40 Hz is typically good for a magnetic field sensor.

It is possible to scale the system requirements of the library in terms of MCU/MPU load. As the osx_MotionFX_update function requires approximately three times more computation power than osx_MotionFX_propagate function, it be called at a lower frequency than the library output data rate of the library if systems resources are limited (e.g., in embedded systems).

Use the `modx` parameter in osxMFX_knobs structure to decrease the frequency of osx_MotionFX_update function calls. For example, setting modx to 2 calls the osx_MotionFX_update function once every two osx_MotionFX_propagate function calls.

Recommended settings:

- for tablets or other systems with MCU/MPU, set modx to 1
- for STM32F4, set modx to 1
- for STM32F1, set the modx to 2

The library can also be used on Cortex-M0 (e.g., STM32L0).

1.5.2 Sensor calibration in the osxMotionFX library

1.5.2.1 Gyroscope and accelerometer calibration

Accelerometer calibration is not necessary for sensor fusion except for applications demanding very high orientation precision; it involves aligning the system in several positions according to the direction of gravity.

Gyroscope calibration is handled automatically by the osxMotionFX library by continuously compensating the zero-rate offset effect.

1.5.2.2 Magnometer calibration

The osxMotionFX library contains routines for magnetometer calibration, which is affected by the hard-iron and soft-iron phenomena described below.

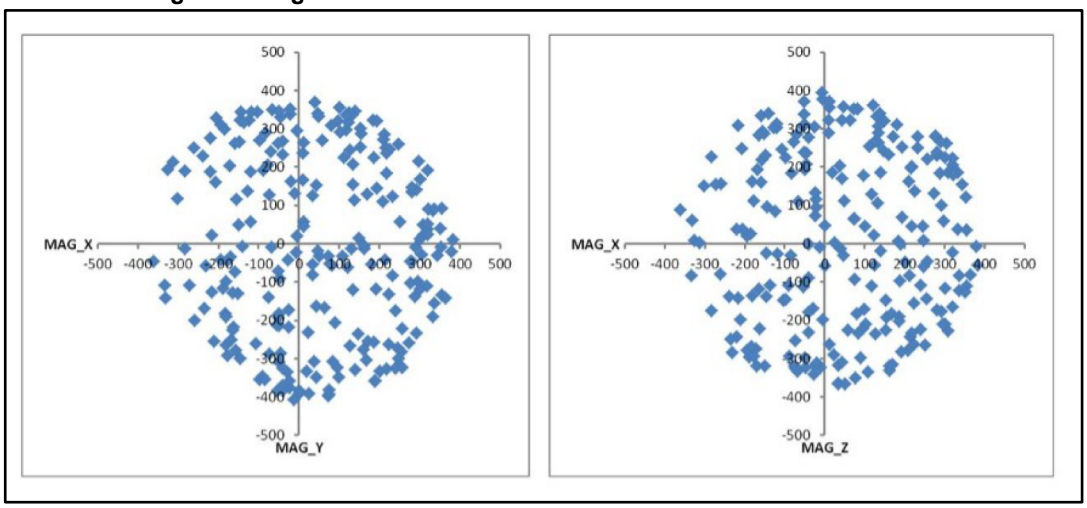
Hard-iron distortion

Hard-iron distortion is normally generated by ferromagnetic material with permanent magnetic fields that are part of the object (e.g., a tablet) in use. These materials can be permanent magnets or magnetized iron or steel. They are time invariant and deform the local geomagnetic field with different offsets on different directions.

As each board can be magnetized differently, the hard iron effect is specific to the individual board.

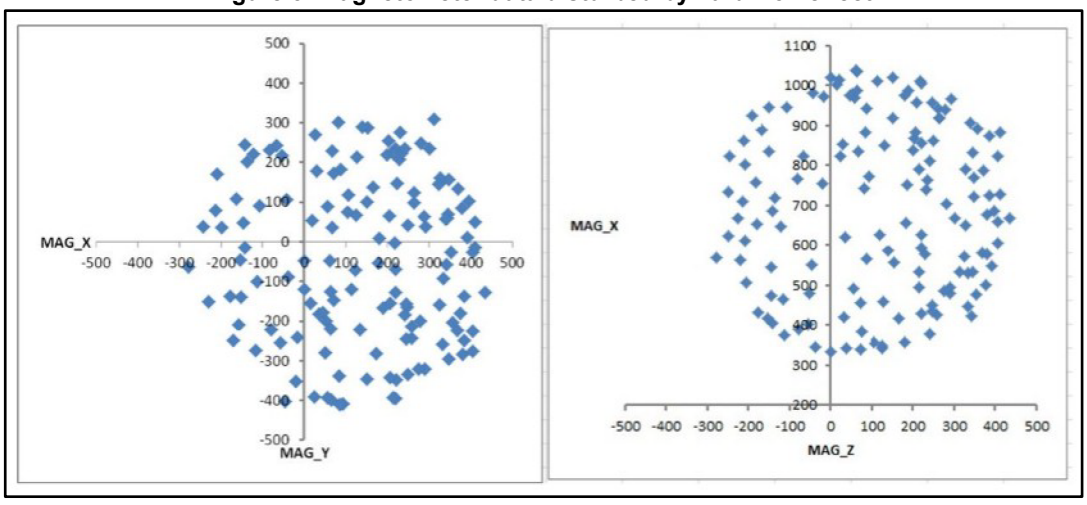
If you move the board around a space approximating (as much as possible) a 3D sphere in an ideal environment (no hard-iron/soft-iron distortion) and plot the collected magnetic sensor raw data, the result is a perfect sphere with no offset.

Figure 4: Magnetometer data in ideal environment or after calibration



The hard-iron distortion effect offsets the sphere along the x, y and z axes; in the x-y plane, the hard-iron distortion is identified by an offset of the origin of the ideal circle from (0, 0), scatter plots for XY and XZ axes are sufficient to determine if there is an offset.

Figure 5: Magnetometer data disturbed by hard-iron effect



Soft-iron distortion

Soft-iron distortion is generated by magnetically soft materials or current carrying PCB traces. While hard-iron distortion is constant regardless of the orientation, the soft-iron distortion changes with the orientation of the object in the Earth's field (soft magnetic materials change their magnetization direction).

The local geomagnetic field is deformed with different gain on different directions. The effect of the soft-iron to distort the ideal full round sphere into a tilted ellipsoid; in the x-y plane, the soft-iron distortion is identified by a tilted ellipse with the origin at (0,0) for the XY axis (XZ).

The soft iron effect is the same across all boards of the same design, which is why a good PCB design which takes magnetometer placement (high current traces/other component clearance) into account can generally avoid any soft iron effects (valid for X-NUCLEO-IKS01A1 and X-NUCLEO-IKS01A2).

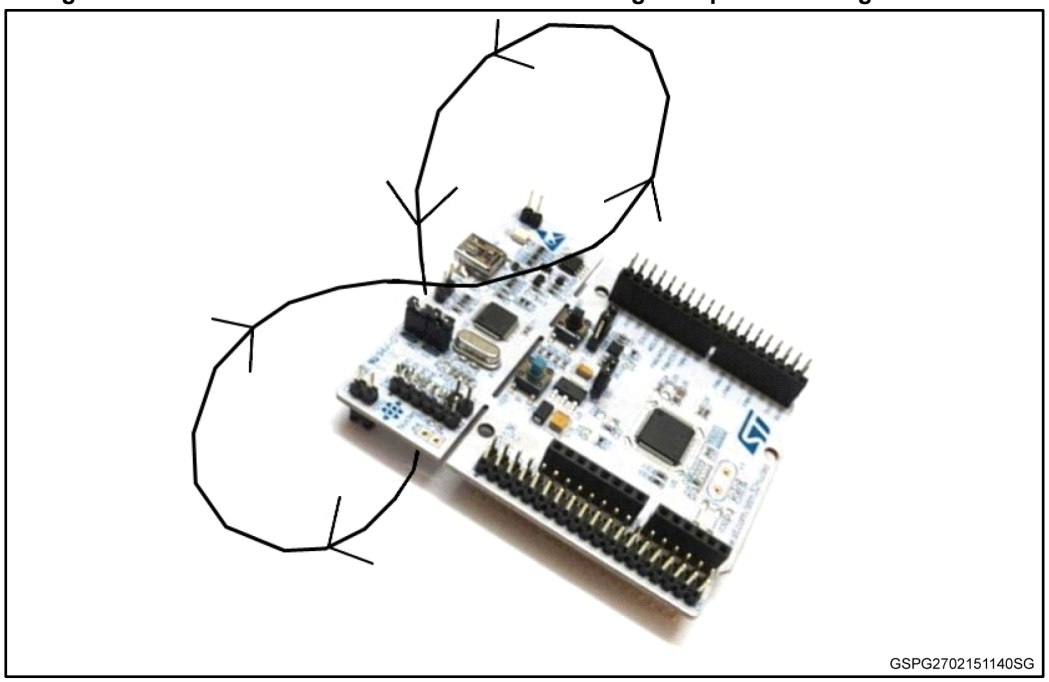
1.5.2.2.1 Calibration procedure

The osxMotionFX library magnetometer calibration library compensates for hard-iron distortions. The magnetometer calibration can be performed at a slower frequency than the sensor fusion output data rate (e.g., 25 Hz). The sequence is described below.

1. Set input data from accelerometer and magnetometer to calibration library (osx_MotionFX_compass_saveAcc and osx_MotionFX_compass_saveMag functions)
2. Call calibration function (osx_MotionFX_compass_run function)
3. Check if calibration was successful (osx_MotionFX_compass_isCalibrated function):
   - If it returns 1, then the calibration was successful and the results (magnetometer offset) can be read by osx_MotionFX_getCalibrationData function
   - Otherwise the process needs to be repeated from step 1.
4. Apply calibration results:
   - MAG_Calibrated.AXIS_X = MAG_Value.AXIS_X - magOffset.magOffX;
   - MAG_Calibrated.AXIS_Y = MAG_Value.AXIS_Y - magOffset.magOffY;
   - MAG_Calibrated.AXIS_Z = MAG_Value.AXIS_Z - magOffset.magOffZ;
5. After initiating the calibration routine, the Nucleo board should be slowly rotated performing a figure 8 movement in the 3D space.

   While performing this pattern, keep the Nucleo board clear of other magnetic objects such as cell phones, computers and other steel objects.

Figure 6: Rotation of the STM32 Nucleo board in a figure 8 pattern during calibration



6 To check that the calibration was performed correctly, you should check magnetometer data (after applying calibration results) using two 2D plots.

Figure 4: "Magnetometer data in ideal environment or after calibration" is an example of good calibration and *Figure 5: "Magnetometer data disturbed by hard-iron effect"* is an example of bad calibration).

7 The magnetometer calibration can be restarted by calling the osx_MotionFX_compass_forceReCalibration function.

1.6 Sensors data logging utility

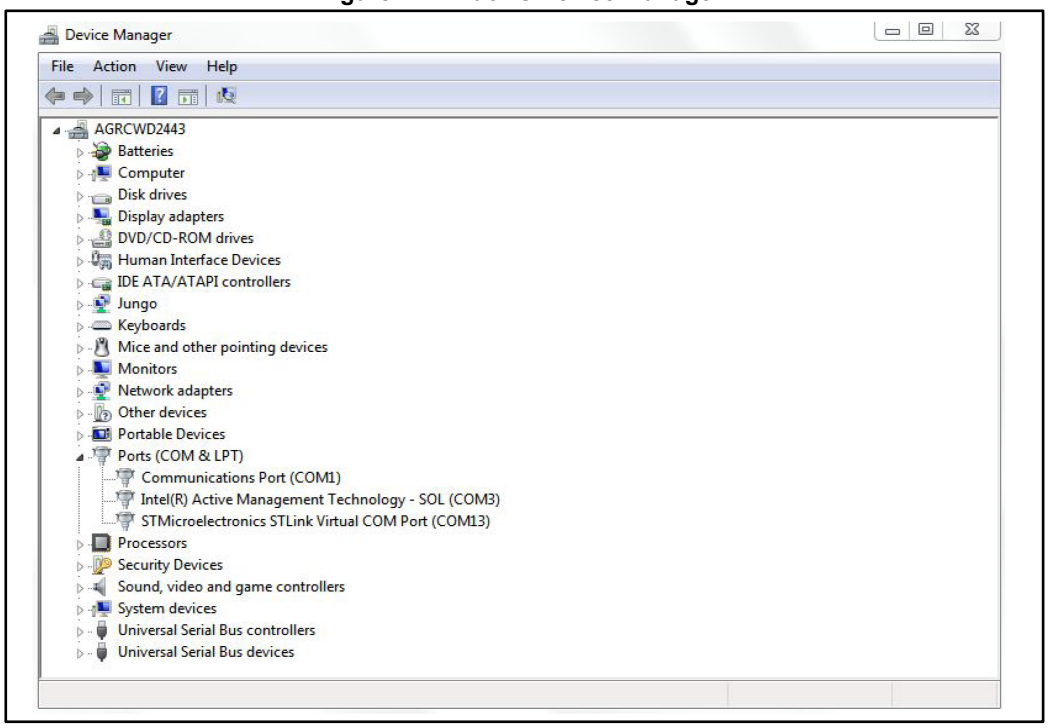
The X-CUBE-MEMS1 expansion for STM32Cube contains an utility for Windows PCs called "Sensors_DataLog", available in the ROOT_DIR\Utilities\PC_software folder.

Before using this utility, ensure that the necessary drivers described in the system setup sections are installed and the STM32 Nucleo development board plus expansion board assembly is connected to the PC.

1 Check the Windows Device Manager for the ST COM port number.

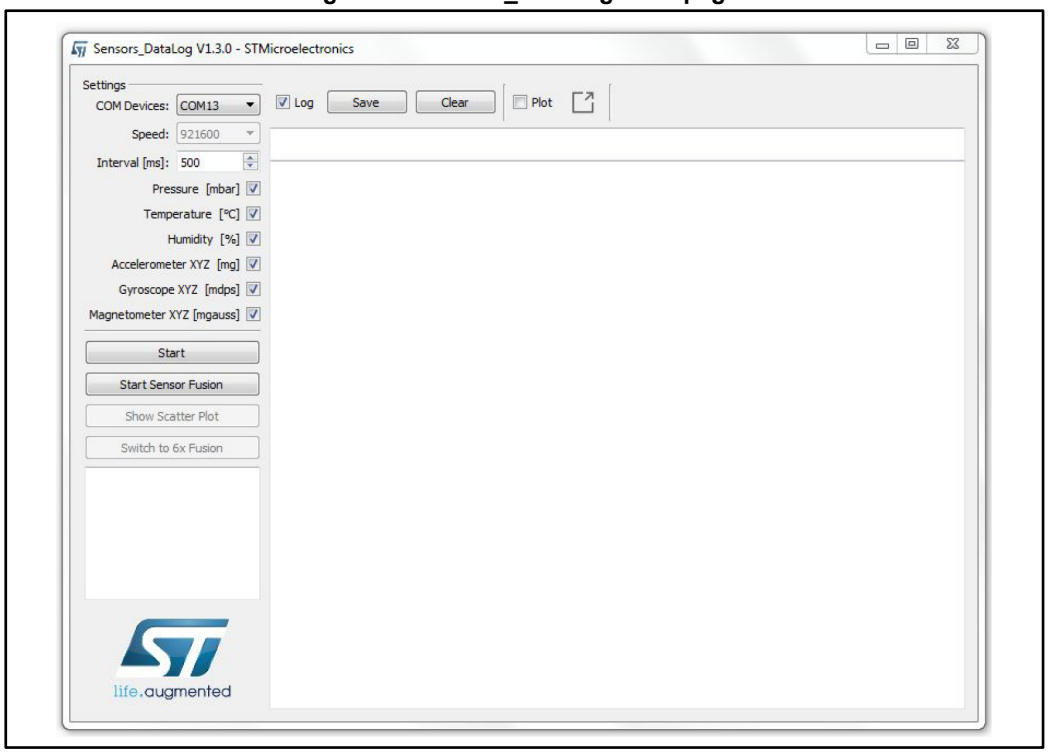
in the example below, it is COM13.

Figure 7: Windows Device Manager



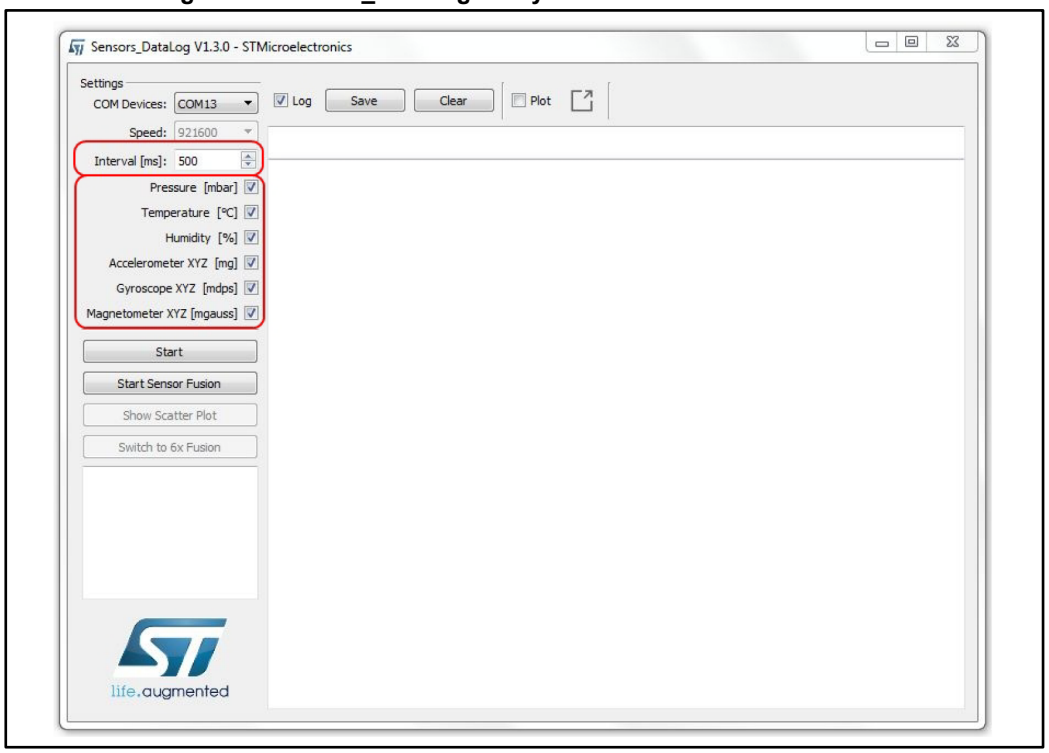
2 Launch Sensors_DataLog.exe and ensure the COM port number for the current expansion board is correct.

Figure 8: Sensors_DataLog main page



3 Select between various sensors (e.g., pressure, temperature, humidity, accelerometer, gyroscope, magnetometer) available on the expansion board and set appropriate delay/interval in milliseconds between consecutive data points; the default is 500 ms

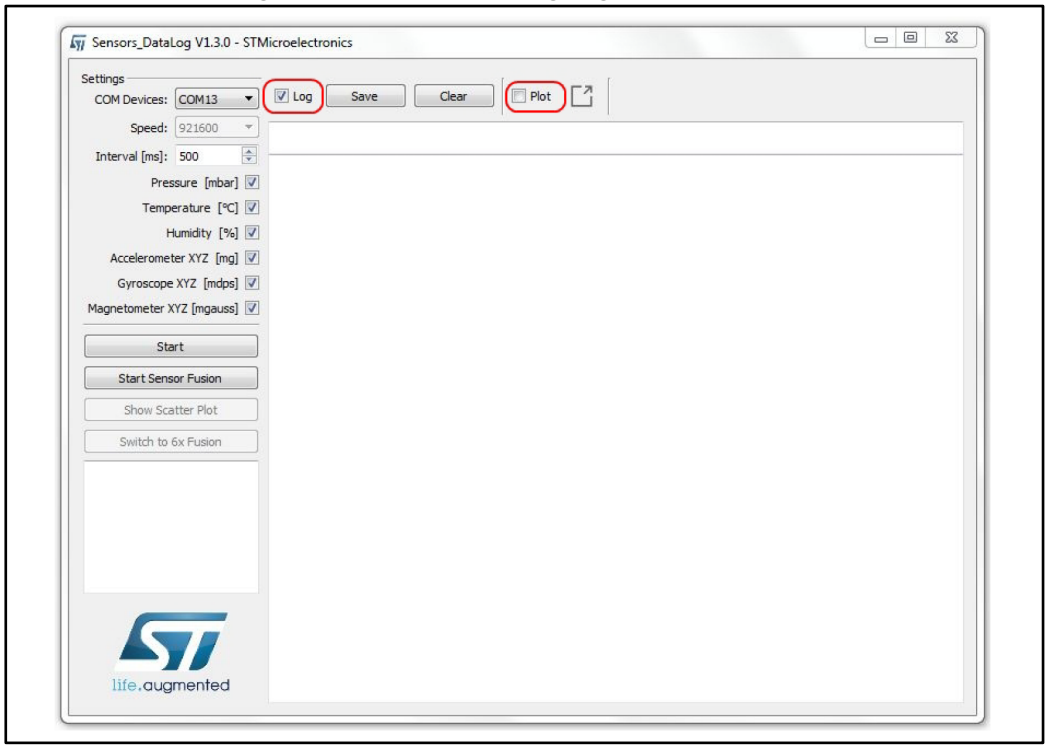
Figure 9: Sensors_DataLog Utility sensor and interval selection



4 Click Save to record the data in a log file in the SensorsDataLog folder, or remove the data with the Clear button.

5 Check the Plot CheckBox to view the data log for the selected sensors.

Figure 10: Sensors_DataLog Log and Plot options



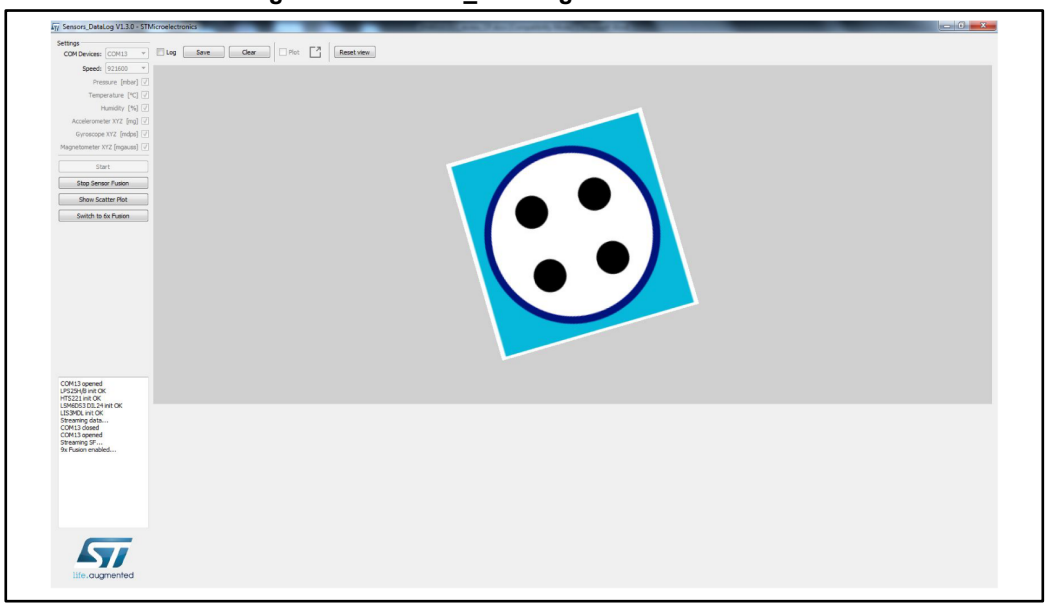
6 Press Start to display the data.

Figure 11: Sensors_DataLog Plot view

1.6.1 Sensor data logging utility with Sensor Fusion

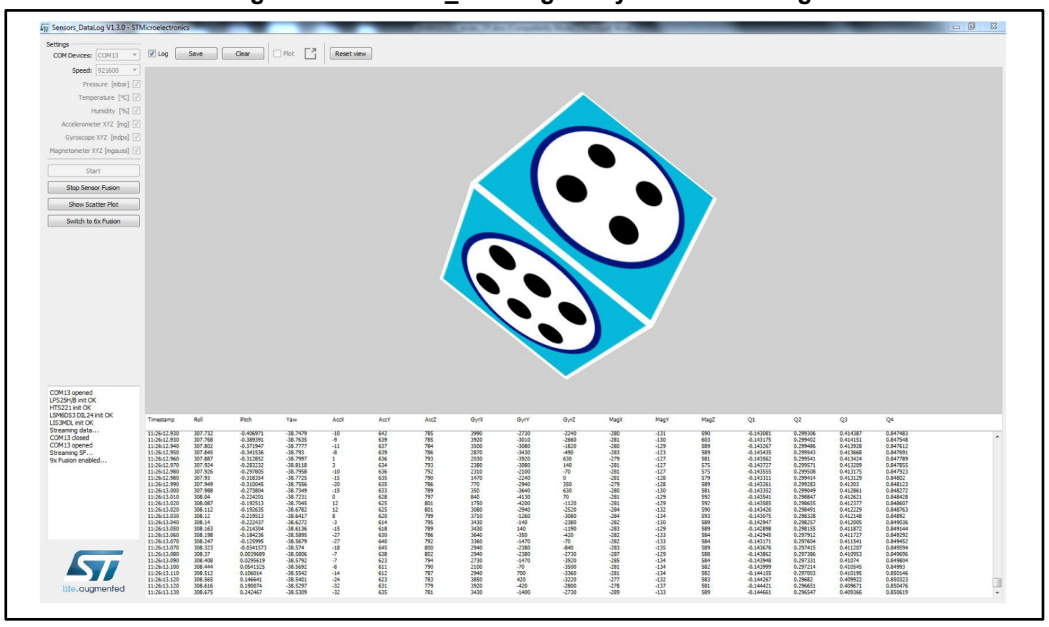
1 Click Start Sensor Fusion to display a 3D cube in the main window.
The cube animation is generated using sensor fusion quaternions.

Figure 12: Sensors_DataLog 3D cube animation



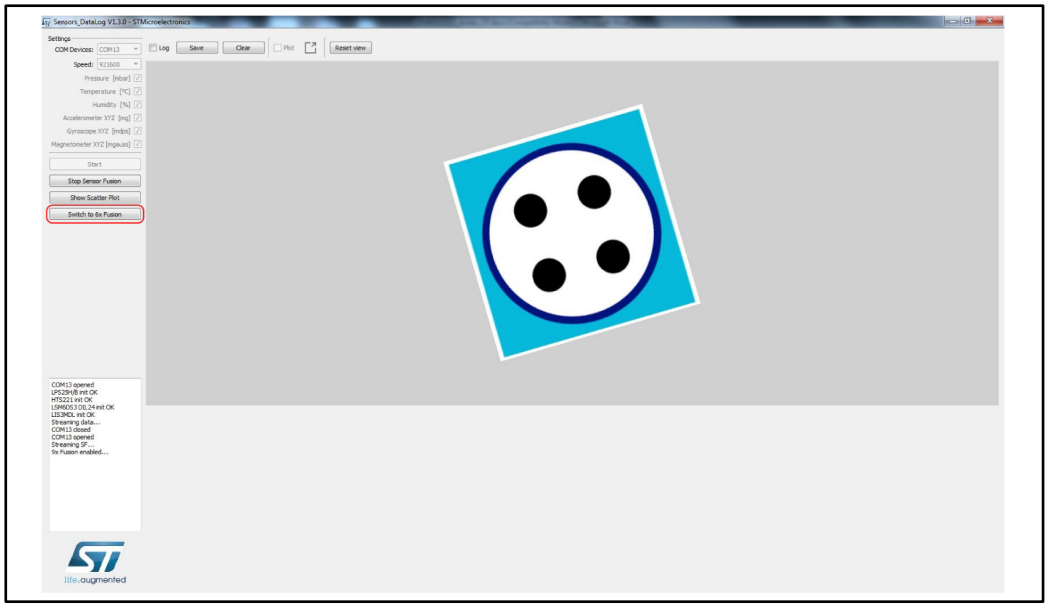
2 Check the Log CheckBox to see the MEMS sensor data and the sensor fusion data (quaternions and Euler angles).

Figure 13: Sensors_DataLog Utility sensor data log



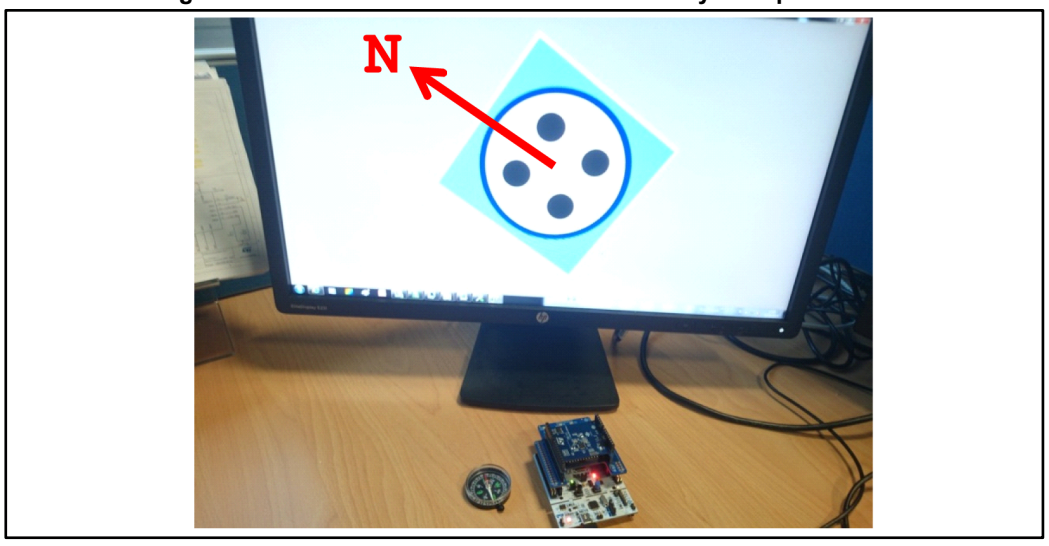
3 Press Switch to 6x Fusion to pass dynamically from 9x fusion to 6x fusion.
Switch to 9x Fusion returns you to 9x fusion.

Figure 14: Sensors_DataLog Utility Switch to 6x Fusion



4 Ensure the STM32 Nucleo board is powered on.
When the STM32 Nucleo board is powered, the compass is not calibrated (green LED2 on the Nucleo board is OFF) and the cube may not point to magnetic north.

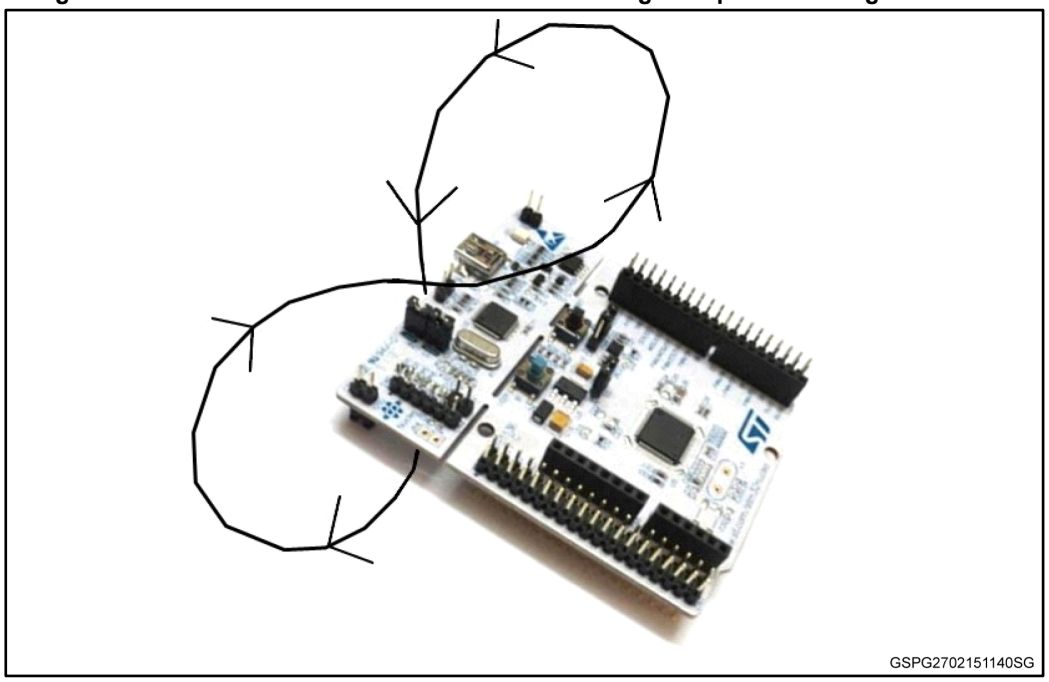
Figure 15: View of the board and Windows utility after power on



5 To calibrate the compass, press the user button on the Nucleo board and perform the figure 8 calibration movement.

You can only calibrate the compass when the sensor fusion is activate (press the "Start Sensor Fusion" button on the PC GUI).

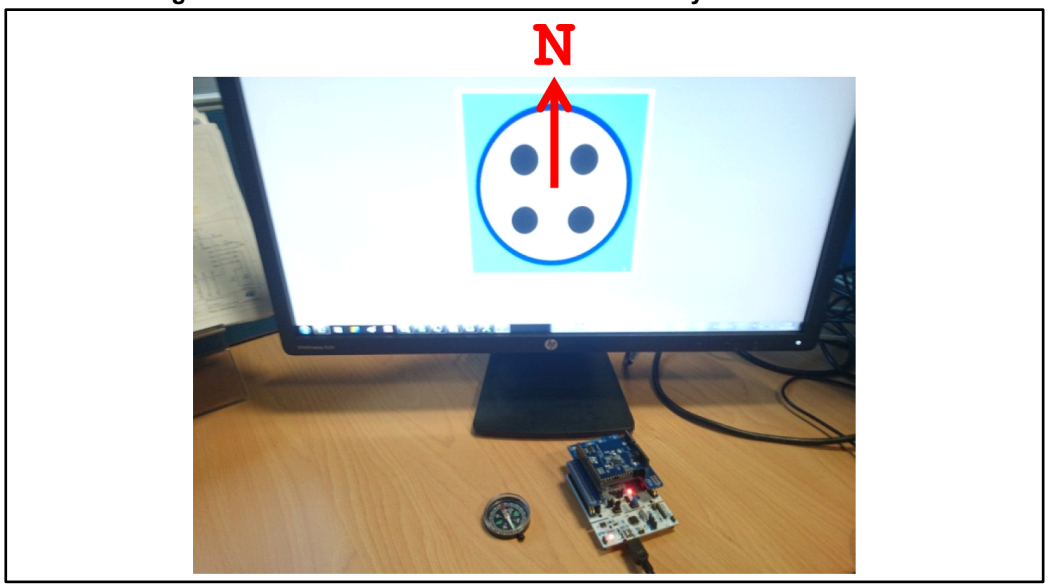
Figure 16: Rotation of the STM32 Nucleo board in a figure 8 pattern during calibration



6 When the calibration is completed, wait a few seconds until the cube shown in the GUI points north.

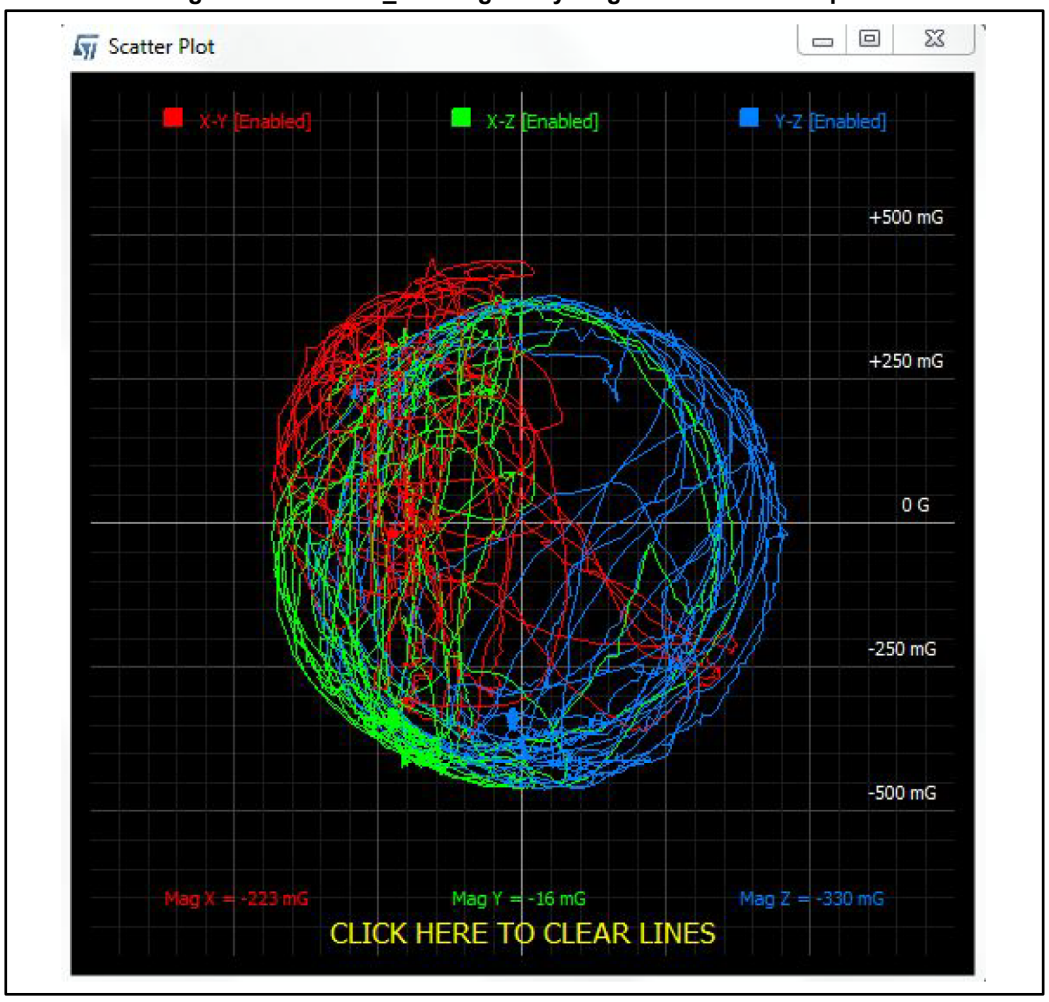
When the calibration is done, LED2 turns on.

Figure 17: View of the board and Windows utility after calibration



7 Click Show Scatter Plot in the GUI to check the quality of the magnetometer calibration in a scatter plot.

Figure 18: Sensors_DataLog Utility magnetometer scatter plot



8 If you wish to store calibration data in Flash memory, compile the sample application "DataLogFusion" defining `OSXMOTIONFX_STORE_CALIB_FLASH` in the project.

By default, the calibration data is stored in RAM memory so it is preserved when you reset the board.

2 System setup guide

2.1 Hardware Description

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

2.1.1 STM32 Nucleo platform

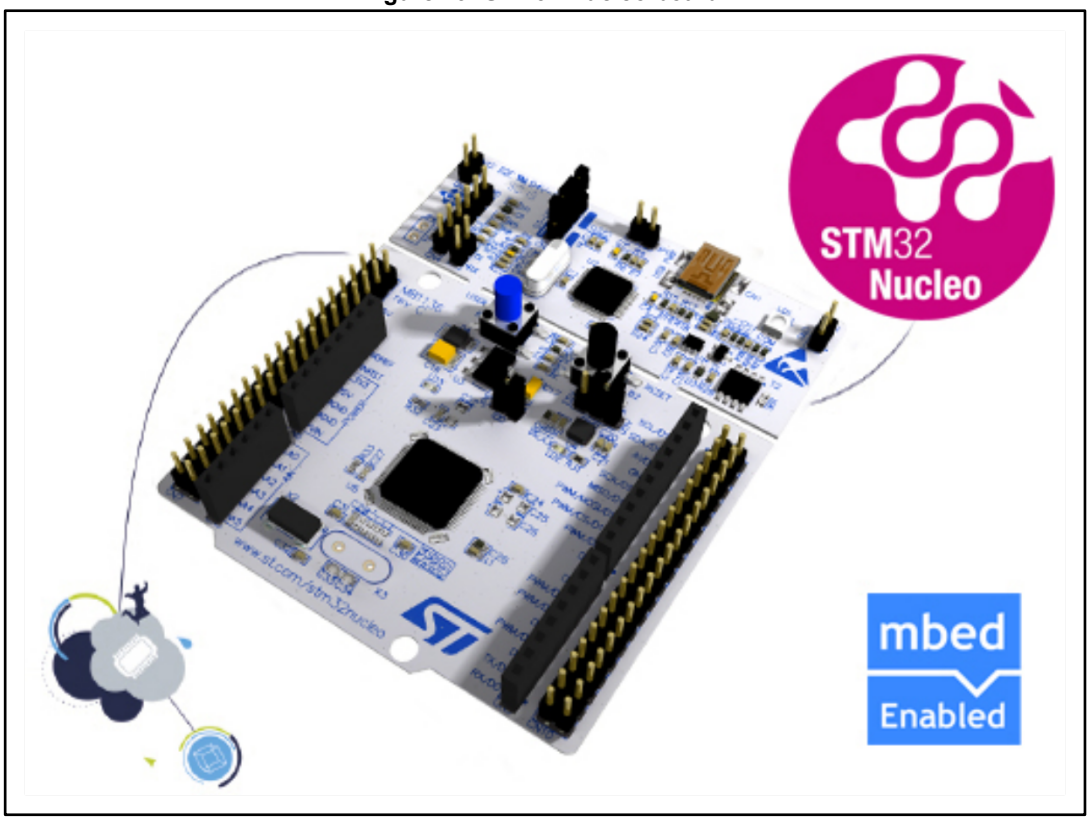
STM32 Nucleo development boards provide an affordable and flexible way for users to test solutions and build prototypes with any STM32 microcontroller line.

The Arduino™ connectivity support and ST morpho 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 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.

Figure 19: STM32 Nucleo board



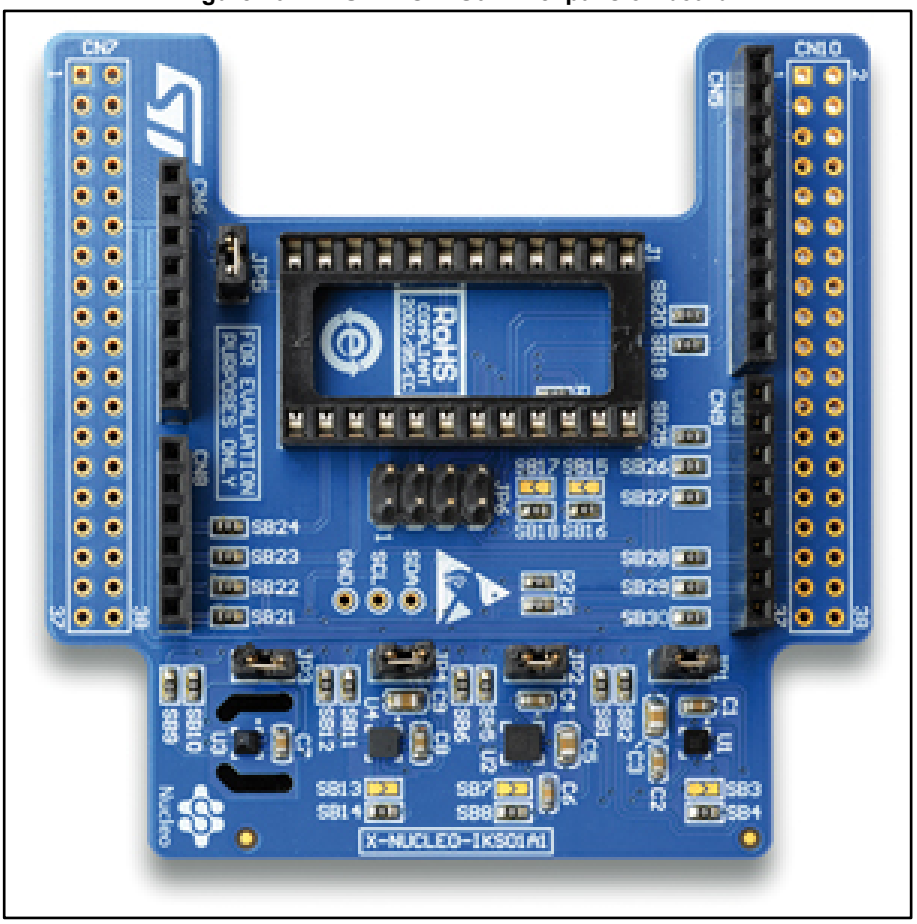
Information regarding the STM32 Nucleo board is available at *www.st.com/stm32nucleo*

2.1.2 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 20: X-NUCLEO-IKS01A1 expansion board



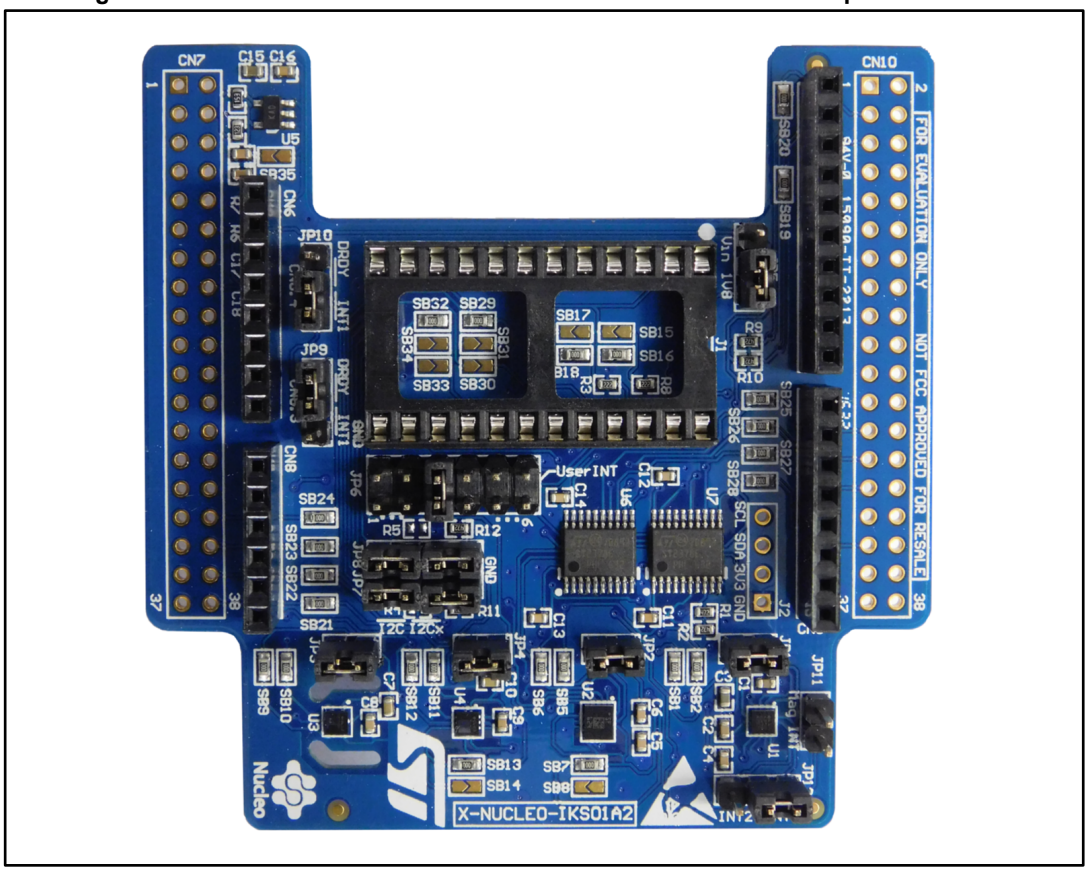
2.1.3 X-NUCLEO-IKS01A2 expansion board

The X-NUCLEO-IKS01A2 is a motion MEMS and environmental sensor expansion board for the STM32 Nucleo.

It is equipped with Arduino UNO R3 connector layout, and is designed around the LSM6DSL 3D accelerometer and 3D gyroscope, the LSM303AGR 3D accelerometer and 3D magnetometer, the HTS221 humidity and temperature sensor and the LPS22HB pressure sensor.

The X-NUCLEO-IKS01A2 interfaces with the STM32 microcontroller via the I²C pin, and it is possible to change the default I²C port.

Figure 21: X-NUCLEO-IKS01A2 MEMS and environmental sensor expansion board



2.2 Software description

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

- X-CUBE-MEMS1: STM32Cube expansion for sensor application development; the firmware and related documentation is available on st.com.
- osxMotionFX: X-CUBE-MEMS1 add-on with fusion and compass calibration APIs; the firmware and related documentation is available on st.com.
- One of the following development tool-chain and compilers supported by the STM32Cube expansion software:
  - IAR Embedded Workbench for ARM® (EWARM) toolchain + ST-LINK
  - RealView Microcontroller Development Kit (MDK-ARM) toolchain + ST-LINK
  - System Workbench for STM32

2.3 Hardware and software setup

This section describes the hardware and software setup procedures and required system setup.

2.3.1 Hardware setup

The following hardware components are required:

1. One STM32 Nucleo development board (suggested order code: NUCLEO-F401RE or NUCLEO-L476RG)
2. One sensor expansion board (order code: X-NUCLEO-IKS01A1 or X-NUCLEO-IKS01A2)
3. One USB type A to Mini-B USB cable to connect the STM32 Nucleo to the PC

2.3.2 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.

2.3.3 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.

2.3.4 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

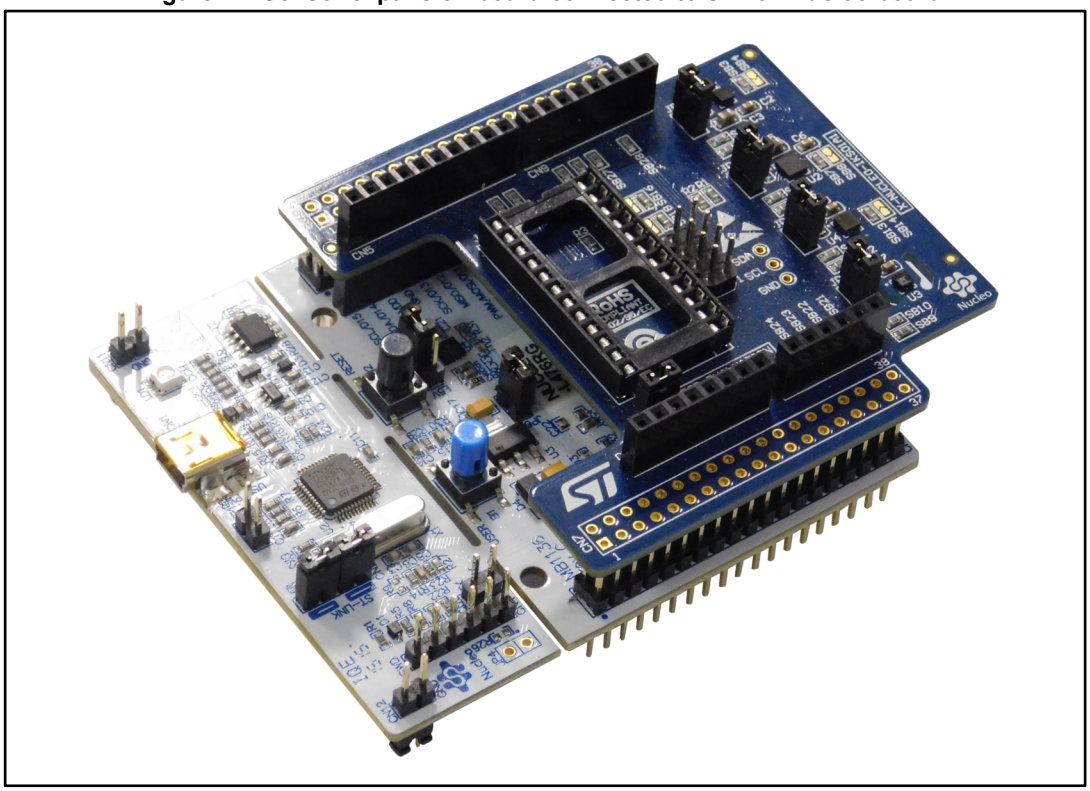
2.3.5 System setup guide

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.

2.3.6 STM32 Nucleo and Sensor expansion boards setup

The STM32 Nucleo board integrates 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 (depending on your version of Microsoft Windows).

Figure 22: Sensor expansion board connected to STM32 Nucleo board



The X-NUCLEO-IKS01A1 or X-NUCLEO-IKS01A2 expansion boards are easily connected to the STM32 Nucleo development board through the Arduino UNO R3 extension connector. The sensor expansion boards are capable of interfacing with the external STM32 microcontroller on the STM32 Nucleo board via an inter-integrated circuit (I²C) transport layer.

2.3.7 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 interact and obtain sensor and sensor fusion data. Please see for a description of the how this PC utility works.

This utility retrieves sensor data from the connected STM32 Nucleo board and displays it in a tabular form, and graphically as well.

In order to use the Sensors_DataLog GUI, make sure you have correctly set up your 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.

2.3.8 osxMotionFX installer setup

The osxMotionFX software package is provided with a Windows installer (osxMotionFX_Setup_vXXX.exe), that guides the user in the correct installation of the software package. The installer also includes the osx License Wizard tool which allows the user to automatically obtain a valid node-locked license for their Nucleo board. The installer of this wizard is described in *Section 2.3.9: "osx License wizard"*.

1 Download the latest release of the X-CUBE-MEMS1 software package and unzip this package in your workspace.

e.g., C:\workspace\STM32CubeExpansion_MEMS1_VX.X.X"

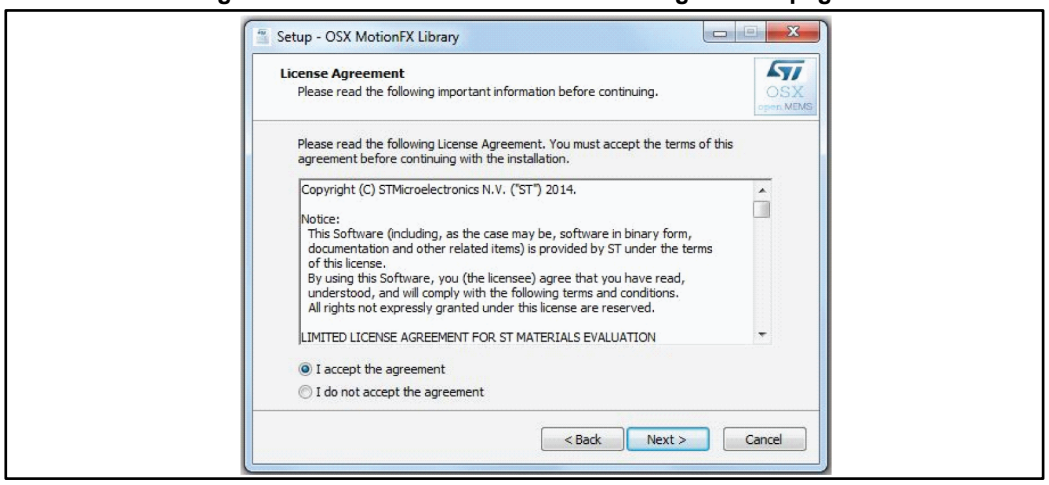
2 Run the installer to open the welcome page

Figure 23: osxMotionFX installer welcome page

3 Read and accept the license terms if you agree to them.

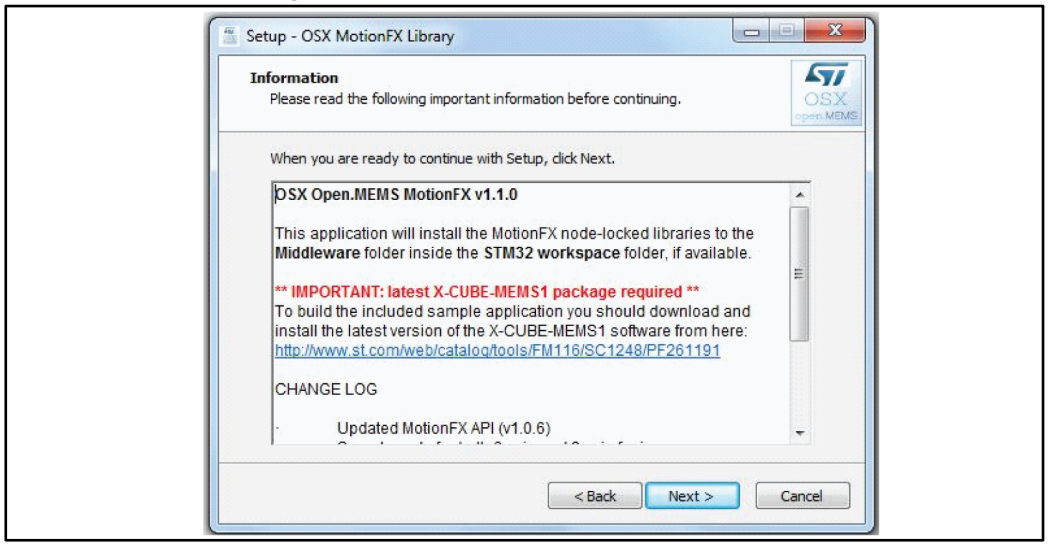
Installation will not proceed if you do not accept the license terms.

Figure 24: osxMotionFX installer license agreement page



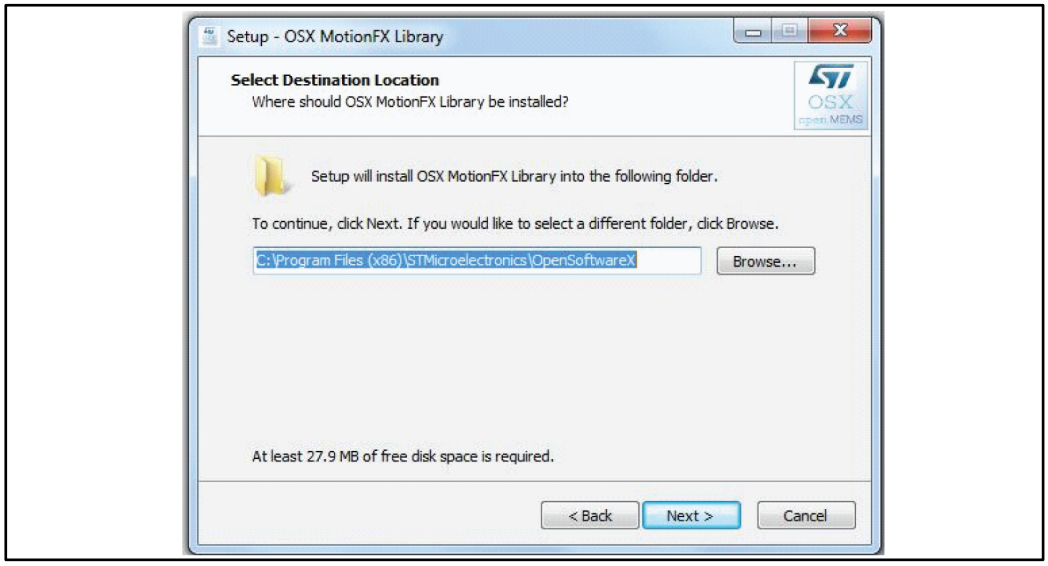
4 The installer provides some information, such as the requirements of the latest release of the X-CUBE-MEMS1 software package.

Figure 25: osxMotionFX installer information



5 Insert the destination location of the osxMotionFX library

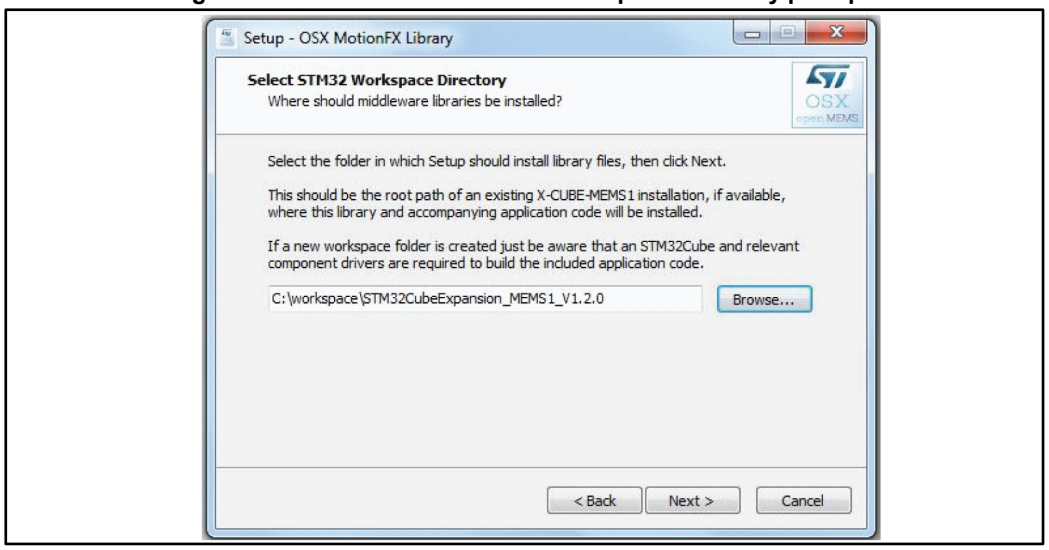
Figure 26: osxMotionFX installer destination folder prompt



6 Insert the path to the X-CUBE-MEMS1 package.

Remember to insert the correct path to the X-CUBE-MEMS1 package or the sample application will not compile. It is the path where you can locate the folders "Documentation", "Projects", "Drivers", "Utilities", etc. contained in the X-CUBE-MEMS1 software package. In our example it is "C:\workspace\STM32CubeExpansion_MEMS1_V1.2.0".

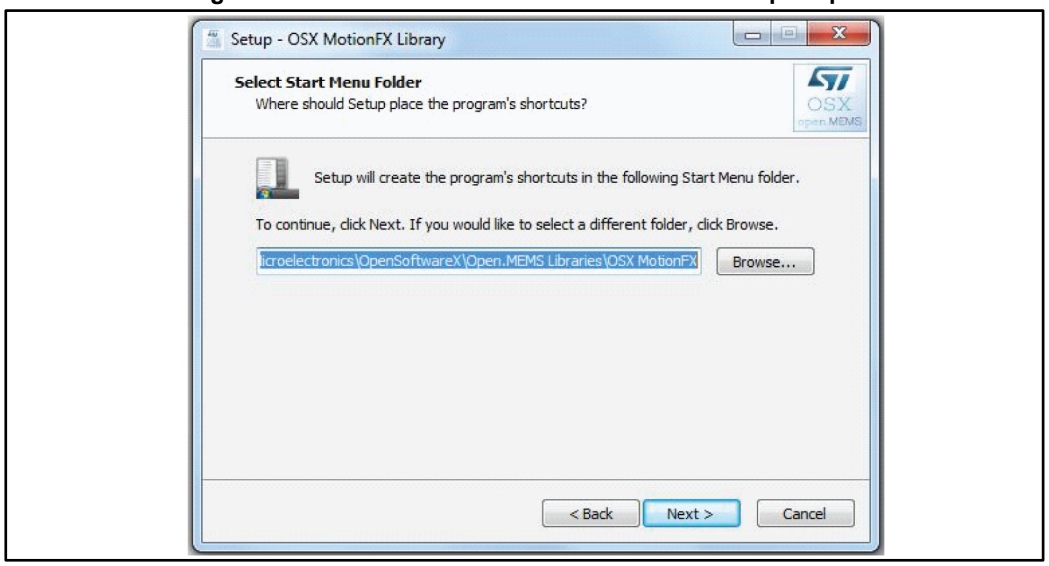
Figure 27: osxMotionFX installer workspace directory prompt



7 Insert the path to the Start menu folder where the shortcuts are installed.

After this step, the osxMotionFX installer will complete the software installation.

Figure 28: osxMotionFX installer start menu folder prompt



2.3.9 osx License wizard

This tool allows you to automatically obtain a valid node-locked license for a specific STM32 Nucleo board.

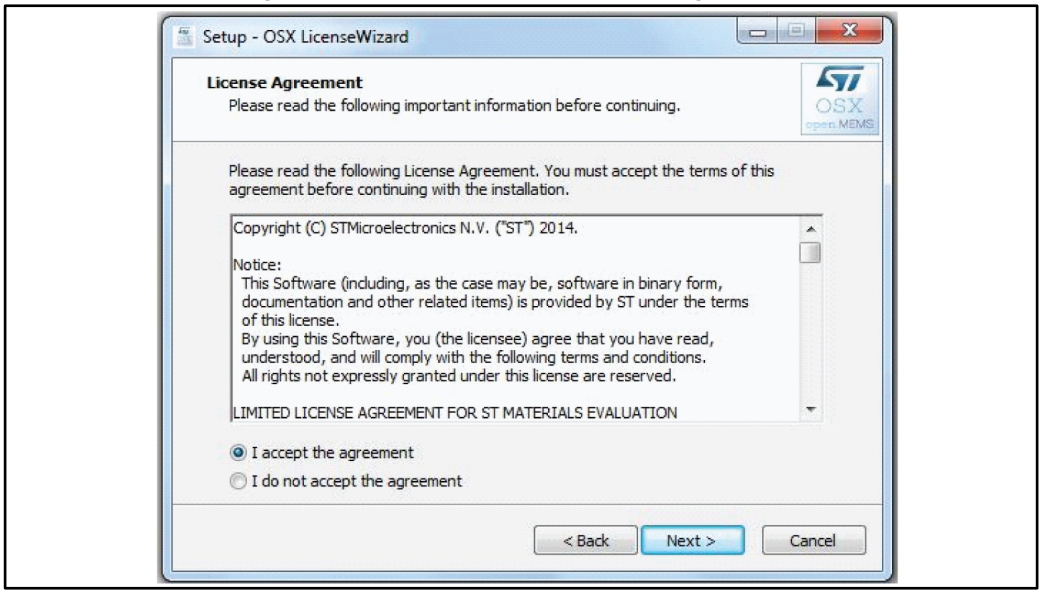
1 Download and run the osxLicenseWizard available on www.st.com

Figure 29: osx License Wizard welcome page

2 Read and accept the license agreement if you agree to the terms.

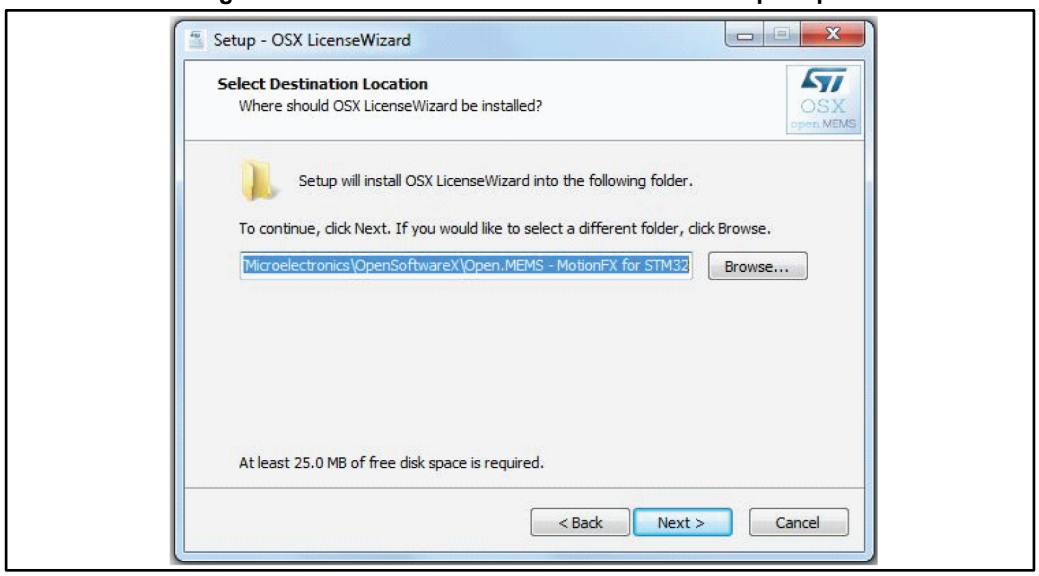
The license wizard installation will not proceed if you do not agree to the terms.

Figure 30: osx License Wizard license agreement



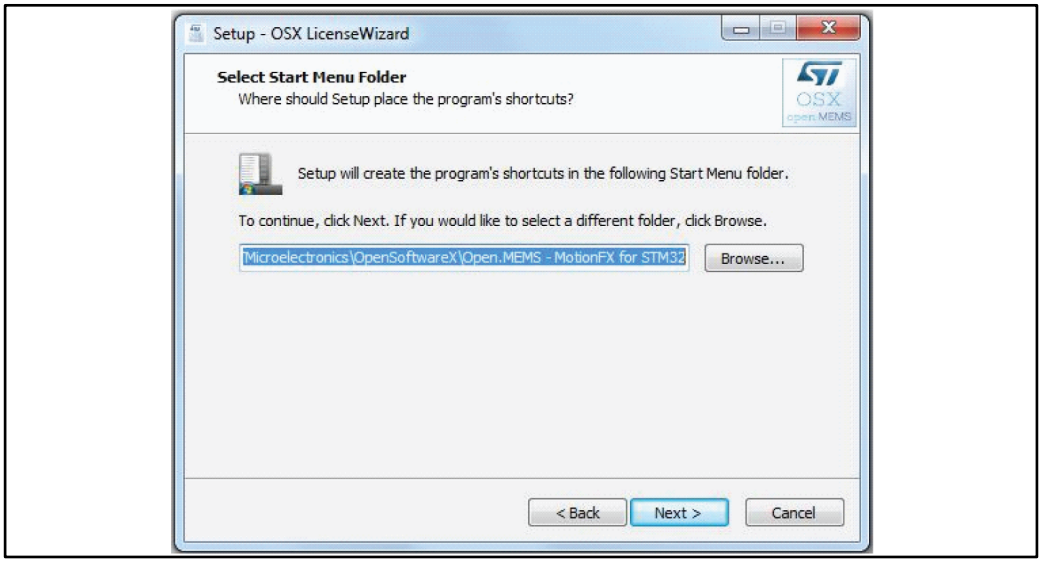
3 Insert the destination location of the osx License Wizard.

Figure 31: osx License Wizard destination folder prompt



4 Insert the path to the Start menu folder where the shortcuts are installed.
After this step, the osx License Wizard can finish the software installation.

Figure 32: osx License Wizard start menu folder location prompt



5 Accept the osx License Wizard prompts for the installation of some additional packages ("Microsoft VS 2013 redistributable" and "STM32 ST-LINK Utility"), if you have not already installed them.

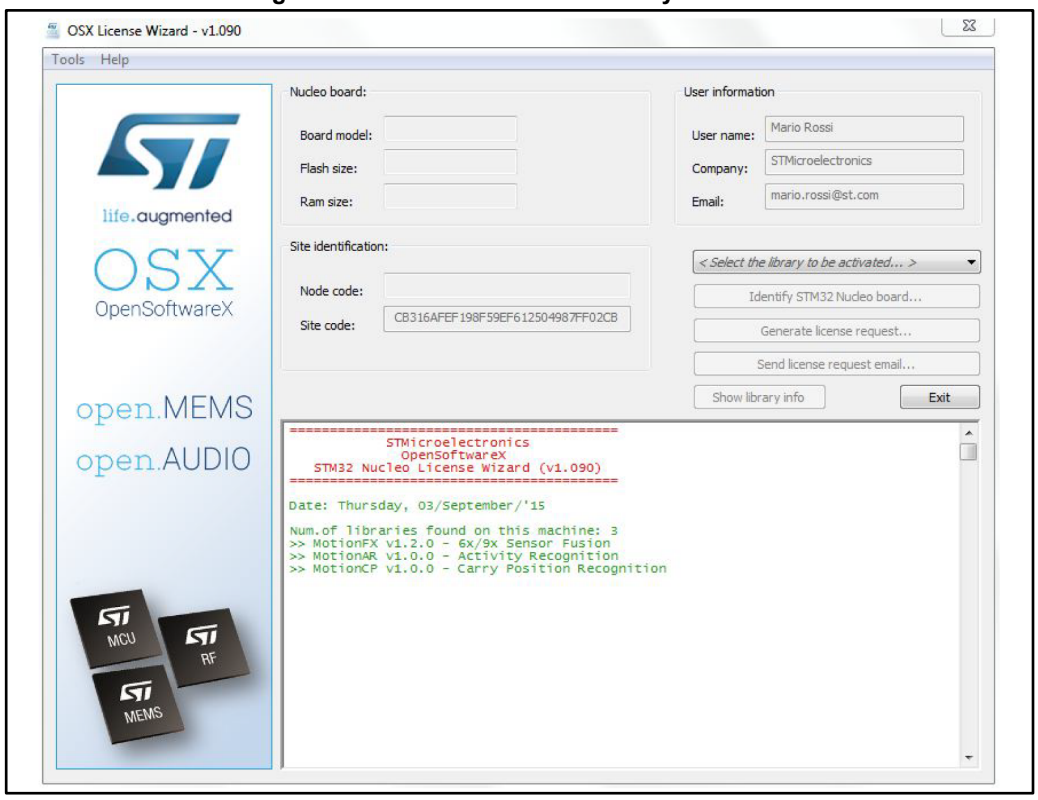
The osx License Wizard needs to access some information on the Nucleo board in order to create the request for the node-locked license.

Figure 33: osx License Wizard additional packages prompt

6 Connect the Nucleo board you want to associate with the node-locked license via USB cable.

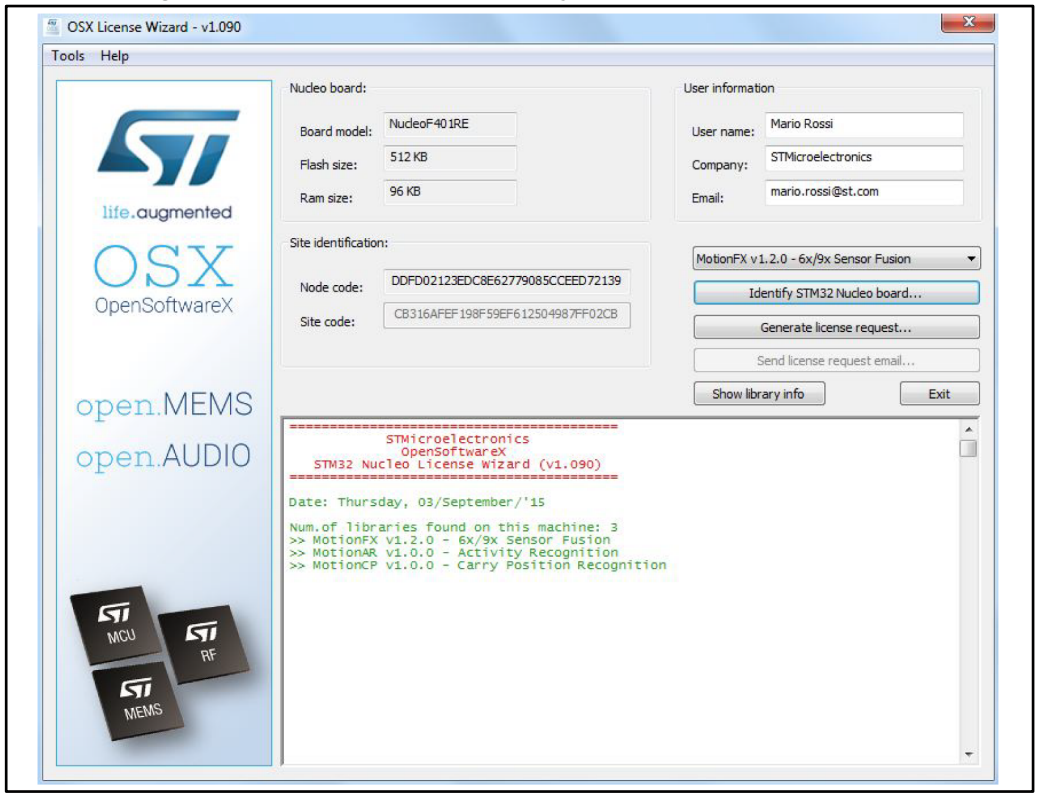
7 Select the library you want to activate.

Figure 34: osx License Wizard library selection



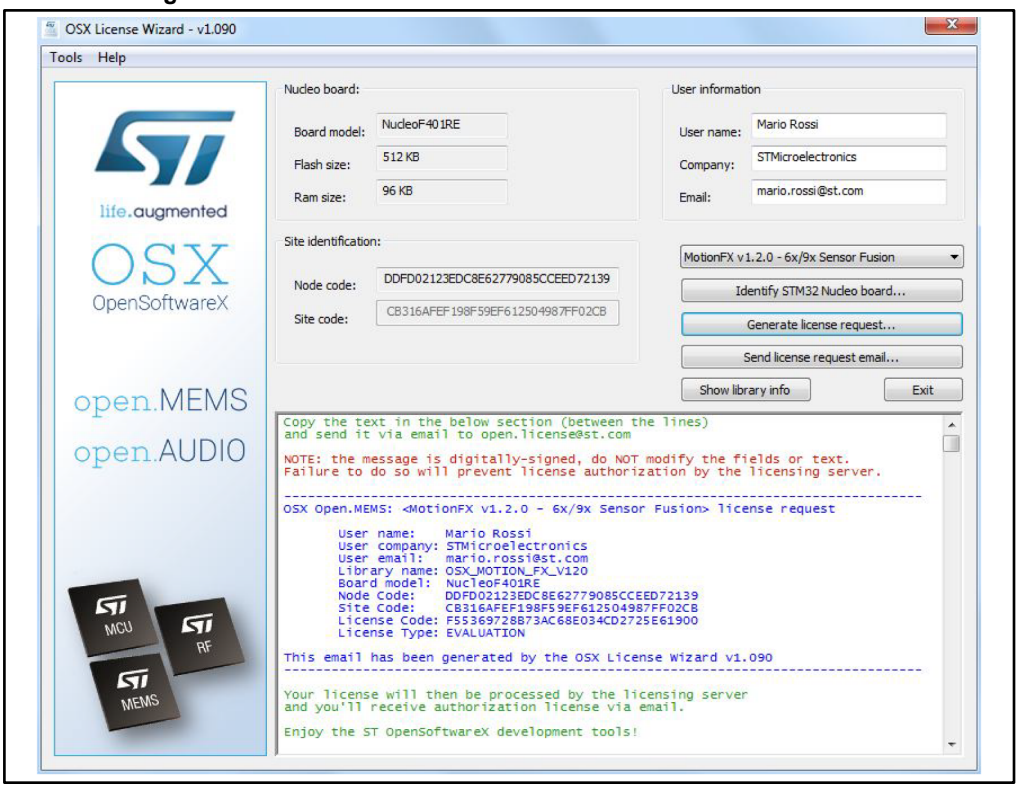
8 Identify the Nucleo board plugged into your PC by clicking the "Identify STM32 Nucleo board" button.

Figure 35: osx License Wizard identify STM32 Nucleo board button



9 You will see information about your Nucleo board.

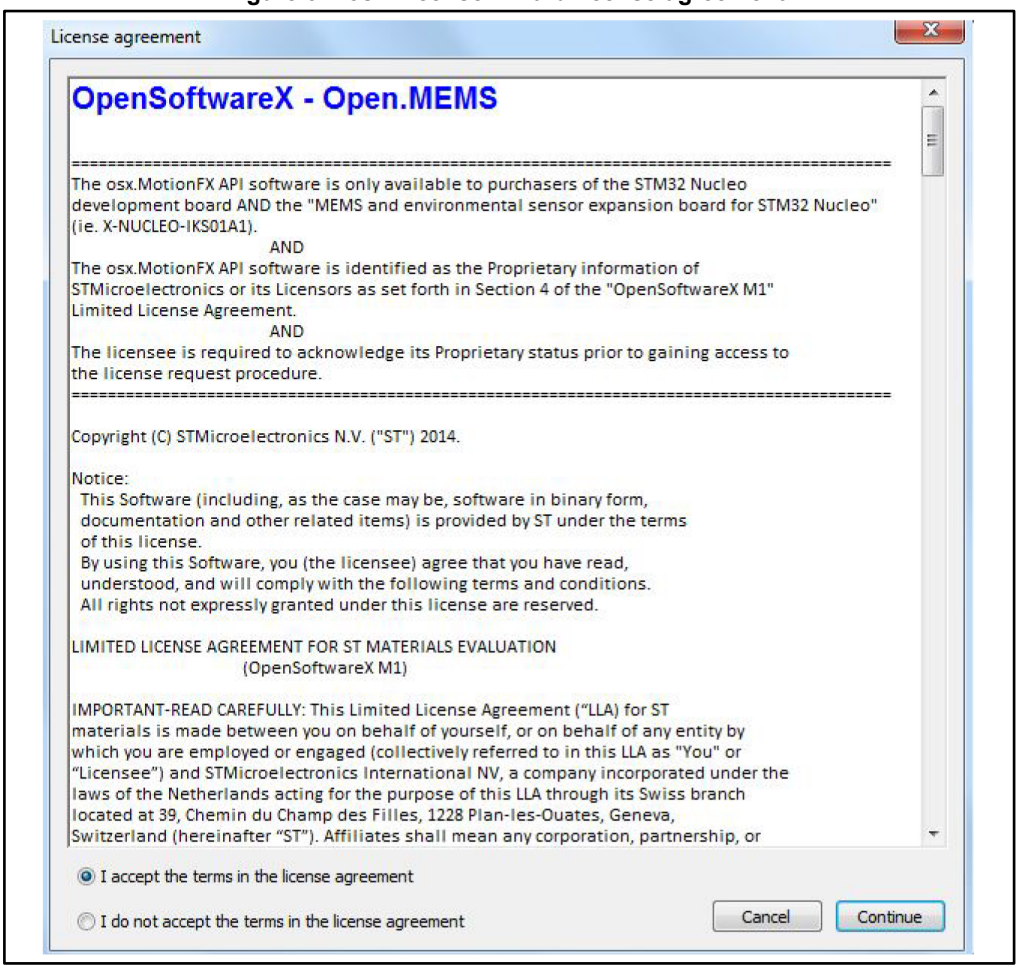
Figure 36: osx License Wizard STM32 Nucleo board information



10 Enter the "User name", "Company" and "Email" fields and press "Generate license request".

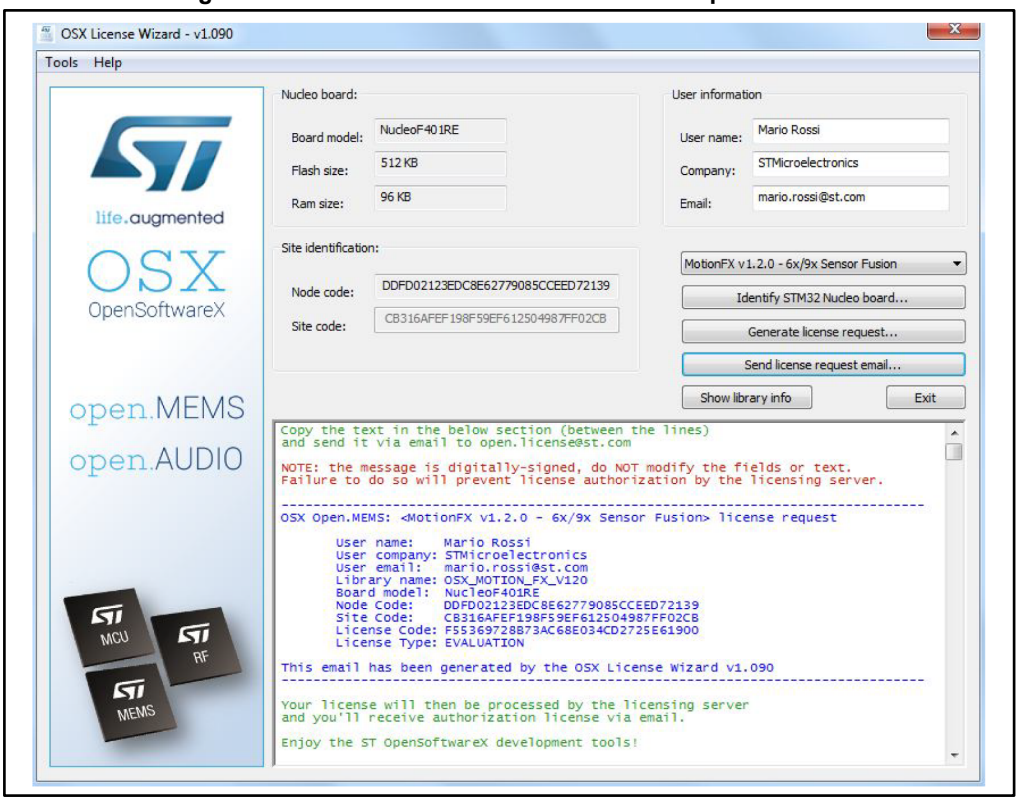
11 Read and accept the license agreement if you agrre to the terms.
Installation cannot proceed if you do not accept the terms.

Figure 37: osx License Wizard license agreement



12 Click the "Send license request email" button.

Figure 38: osx License Wizard send license request email



13 Once the license activation codes are received, edit the content of the osx_license.h file

Also delete or comment the "#error...." line, found in the Middlewares\ST\STM32_OSX_MotionFX_Library folder of your workspace.

3 Acronyms and abbreviations

Table 1: Acronyms and abbreviations

Acronym	Description
NED	x-North, y-East, z-Down
ENU	x-East, y-North, z-Up
SEU	x-South, y-East, z-Up

4 References

- UM1859: Getting started with the X-CUBE-MEMS1 motion MEMS and environmental sensor software expansion for STM32Cube.

5 Revision history

Table 2: Document revision history

Date	Revision	Changes
15-Apr-2015	1	Initial release.
12-Jun-2015	2	Updated: Title on the cover page and osxMotionFX plus X-CUBE-MEMS1 software architecture figure.
05-Oct-2015	3	Updated: Section 1.5.1 6-axis and 9-axis sensor fusion osxMotionFX library.
21-Dec-2015	4	Updated Figure 1: "osxMotionFX plus X-CUBE-MEMS1 software architecture"
27-Oct-2016	5	Text and formatting changes throughout document
24-Nov-2016	6	Updated *Figure 1: "osxMotionFX plus X-CUBE-MEMS1 software architecture"*. Added *Section 2.1.3: "X-NUCLEO-IKS01A2 expansion board"*. Added X-NUCLEO-IKS01A2 compatibility throughout document.

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.