UM2126
User manual

Getting started with osxMotionPE pose estimation library for X-CUBE-MEMS1 expansion for STM32Cube

Introduction

osxMotionPE is an add-on software package for X-CUBE-MEMS1. The software runs on the STM32 and includes drivers that recognize ST inertial sensors LSM6DS0, LSM6DS3 or LSM6DSL.

The pose estimation algorithm differentiates between stationary user poses like standing, sitting and lying down, so you can build applications that may, for example, signal the user to stand up after prolonged sitting, activate sleep monitoring if the user is lying down, and even calorie consumption when used in conjunction with activity detection algorithms like osxMotionAW.

The algorithm manages the data acquired exclusively from the accelerometer at the low sampling frequency of 16 Hz to reduce host platform power consumption.

The software comes with sample implementations of the drivers, exploiting STM32Cube software technology and running on X-NUCLEO-IKS01A2 board or X-NUCLEO-IKS01A1 with optional STEVAL-MKI160V1 board, mounted on a NUCLEO-F401RE or NUCLEO-L476RG development board.

Contents

List of tables

List of figures

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

1.1 osxMotionPE overview

The osxMotionPE library is a complete middleware solution aimed at building applications strictly for 3D accelerometer sensors and specifically designed for wrist devices.

The software runs on the STM32 microcontroller and includes drivers to recognize the available inertial sensors (currently LSM6DS0, LSM6DS3 or LSM6DSL).

The pose estimation algorithm differentiates between the following different stances when the user is stationary:

- standing
- sitting
- lying down

This may be used to signal the user to stand up after prolonged sitting, to activate sleep monitoring mode if the user is lying down, or even calorie burning calculations when run in conjunction with activity detection algorithms such as osxMotionAW.

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

The algorithm manages data acquired exclusively from the accelerometer at the low, 16 Hz sampling frequency to reduce the power consumption of the host platform.

The key package features include:

- Real-time pose estimation (under OpenSoftwareX license) based exclusively on accelerometer data, for wrist applications.
- Complete middleware to build applications on top of X-CUBE-MEMS1.
- Libraries for ARM Cortex-M3 and ARM Cortex-M4 MCU cores.
- Easy portability across different MCU families, thanks to STM32Cube.
- PC-based Windows application to log data.
- Sample implementations available on X-NUCLEO-IKS01A2 and X-NUCLEO-IKS01A1 (with optional STEVAL-MKI160V1 board) expansion boards, mounted on a NUCLEO-F401RE or NUCLEO-L476RG development board.

The osxMotionPE is an add-on software package for X-CUBE-MEMS1, provided in the form of node-locked library whose license activation codes must be requested to ST and included in the project (thus becoming part of the build process) prior to attempting its usage. The resulting firmware binary image will therefore be node-locked.

For details on system setup and installation as well as details on hardware components, refer to *2*.

1.2 osxMotionPE architecture

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

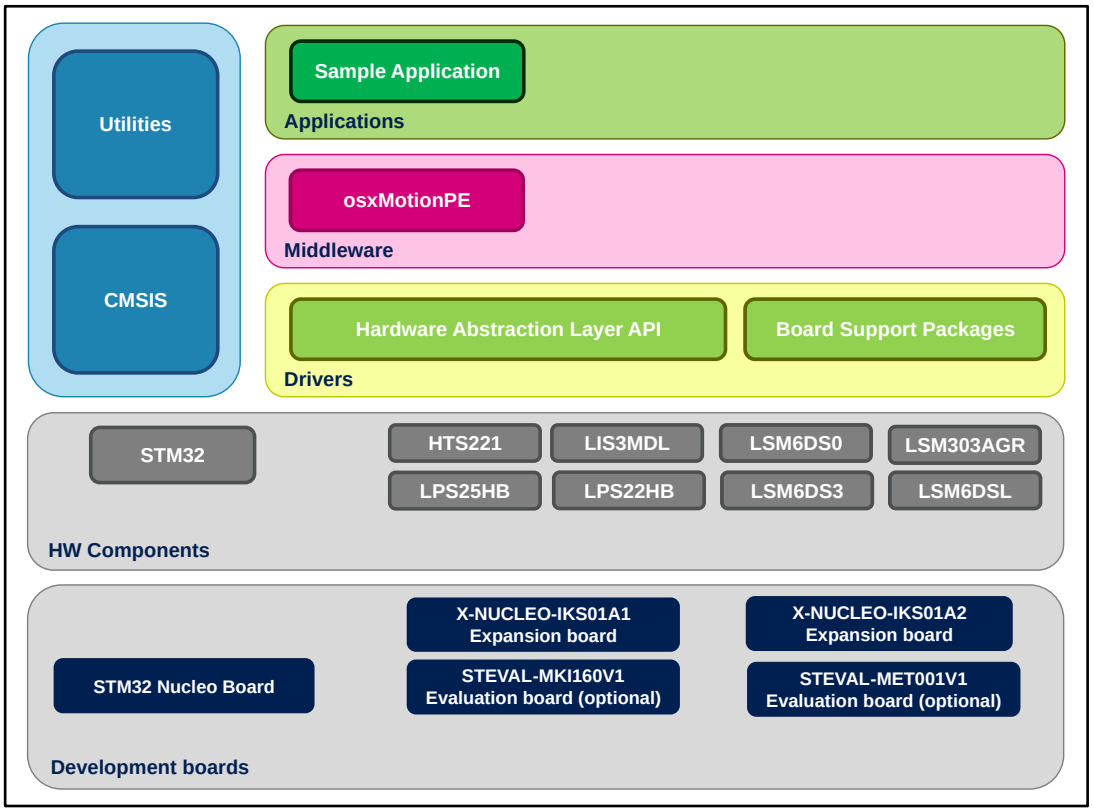
- STM32Cube HAL layer: provides a generic, multi-instance, simple set of APIs (application programming interfaces) to interact with the upper layers (application, libraries and stacks). These generic and extension APIs are built on a generic architecture so overlying layers like middleware can implement their functions without

requiring 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: supports all the available peripherals on the STM32 Nucleo board apart from the MCU. This limited set of APIs provides a programming interface for certain board specific peripherals like the LED and user button, and 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:

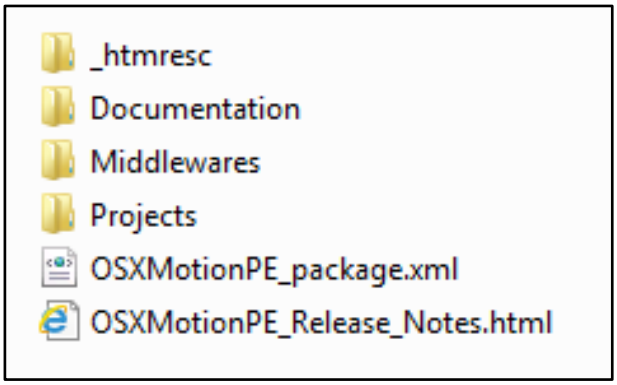
Figure 1: osxMotionPE plus X-CUBE-MEMS1 software architecture



1.3 osxMotionPE folder structure

The image below outlines the package file system architecture:

Figure 2: osxMotionPE package folder structure



The following folders are included in the package:

- **Documentation**: contains a compiled HTML file detailing the software components and APIs.
- **Middlewares**: contains the osxMotionPE static library binary code, the library header file, documentation, license information plus header file for node-locked license validation.
- **Projects**: contains a sample application used to access sensor and activity data with the NUCLEO-F401RE and NUCLEO-L476RG development boards under the IAR Embedded Workbench for ARM, µVision (MDK-ARM) toolchain and System Workbench for STM32 integrated development environments.

1.4 osxMotionPE APIs

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

The osxMotionPE 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_MotionPE_Library folder.

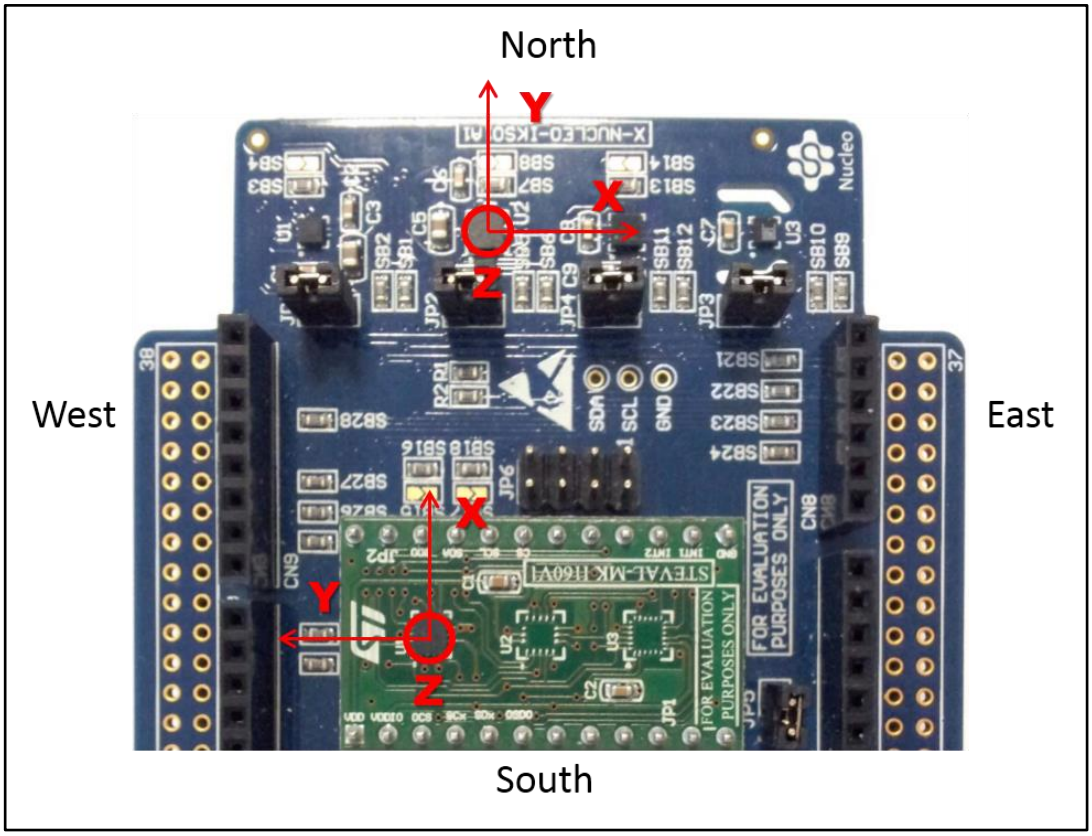
1.4.1 osxMotionPE library

The osxMotionPE is a real time pose estimation software solution specifically designed for wrist supports. 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 osxMotionPE library are listed below:

- `uint8_t osx_MotionPE_GetLibVersion(char *version);`
  - retrieves the revision of the included core engine;
- `uint8_t osx_MotionPE_Initialize(void);`
  - performs osxMotionPE initialization and setup of the internal mechanism used for node-locking (See *2*). The output for correct or incorrect initialization is 1 or 0 respectively (e.g., 0 is returned for license errors);
- `void osx_MotionPE_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_MotionPE_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_MPE_output_t osx_MotionPE_Update (osx_MPE_input_t *data_in);`
  - The required input is a pointer to a structure containing the three-axis accelerometer data expressed in g-force units; the output is provided in the osx_MPE_output_t structure, whose fields are the possible estimated poses.
- `void osx_MotionPE_ResetLib(void);`
  - Resets the algorithm.

1.5 Sample application

The osxMotionPE middleware can be easily manipulated to build user applications.

A sample application provided in the Projects folder runs on either:

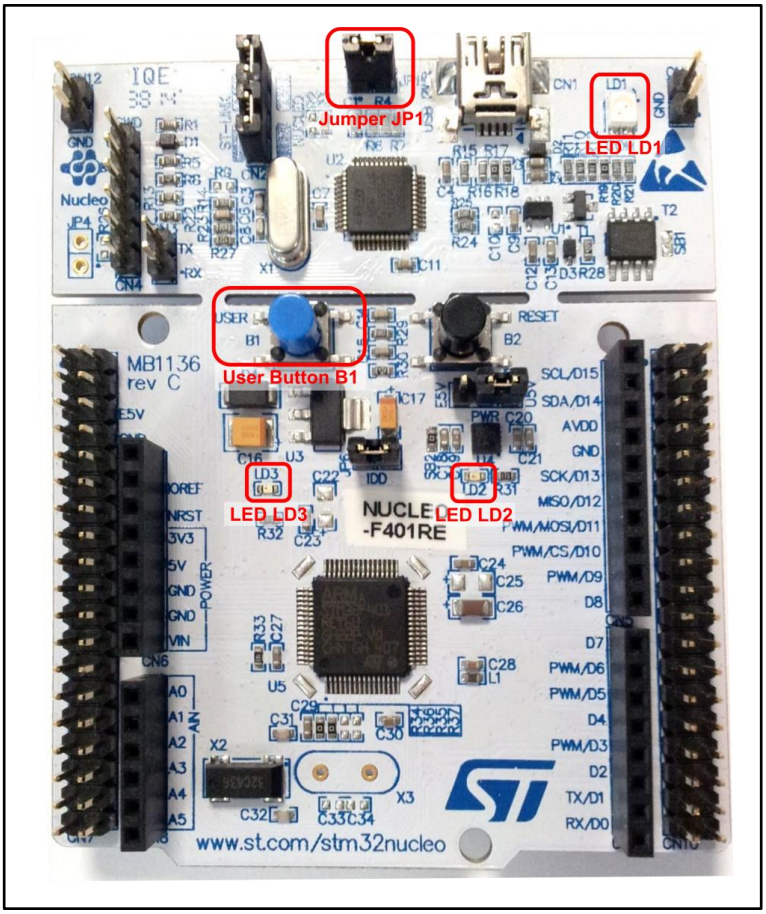
- a NUCLEO-F401RE or a NUCLEO-L476RG development board connected to an X-NUCLEO-IKS01A1 expansion board (based on LSM6DS0) with optional STEVAL-MKI160V1 board (based on LSM6DS3) connected
- a NUCLEO-F401RE or a NUCLEO-L476RG development board connected to an X-NUCLEO-IKS01A2 expansion board (based on LSM6DSL).

The application estimates the real-time user pose such as standing, sitting and lying down.

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 to make the user experience more comfortable, portable and free of any PC connections (*Figure 4: "NUCLEO-F401RE board details"*), ensuring that jumper JP1 is fitted.

Figure 4: NUCLEO-F401RE board details



The above figure shows the NUCLEO-F401RE board which has a similar layout to the NUCLEO-L476RG board. Once the board is powered, LED LD3 (PWR) turns on and the tricolour LED LD1 (COM) begins blinking slowly due to the missing USB enumeration (refer to *4*).

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 estimated pose; during this acquisition mode, rapid LED LD2 blinking indicates that the algorithm is running.

Pressing button B1 a second time stops the algorithm (and the relative data storage session) and the LED LD2 displays the pose code according to a sequence of flashes described in *Table 1: "LED LD2 pose 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 5: "State machine"*).

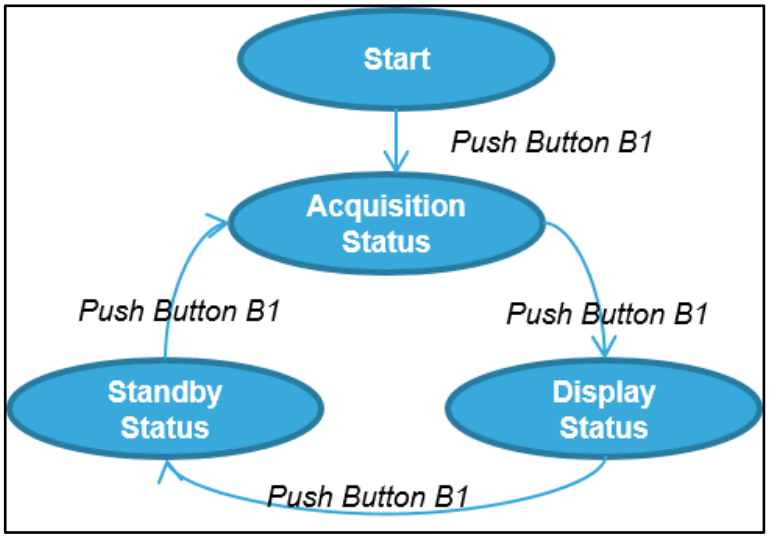
If the LED LD2 is ON after powering the board, this indicates that the Flash memory is full or almost full (see *Section 1.5.4: "Data storage"*) or the library has an incorrect embedded license number.

In order to retrieve acquired data stored in Flash, the STM32 Nucleo board has to be connected to a specific PC GUI (*Section 1.5.3: "Unicleo-GUI utility"*, *Section 1.5.4: "Data storage"*).

Table 1: LED LD2 pose codes

Pose	LED LD2 blinking sequence (5 s interval)
Sitting	1
Standing	2
Lying down	3

Figure 5: State machine



1.5.2 PC GUI driven mode

In this working mode the Nucleo board is powered by PC via USB connection, and controlled by a specific PC GUI.

This working mode allows the user to display the detected pose, accelerometer data, time stamp and any other sensor data, in real-time using the Unicleo-GUI utility GUI.

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

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

1.5.3 Unicleo-GUI utility

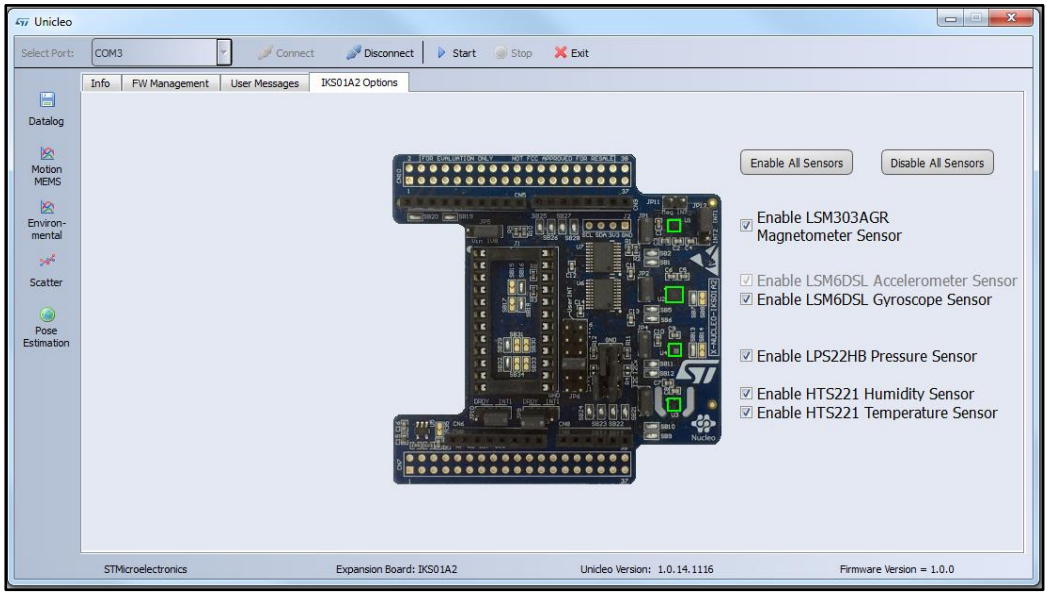
The osxMotionPE software package for STM32Cube uses the Windows Unicleo-GUI utility, which can be downloaded from *www.st.com* (see *5* for instructions).

1 Ensure that the necessary drivers are installed and the STM32 Nucleo board with appropriate expansion board is connected to the PC.

2 launch the Unicleo-GUI application to open the main application window.

If an STM32 Nucleo board with supported firmware is connected to the PC, it will automatically be detected and the appropriate COM port will be opened.

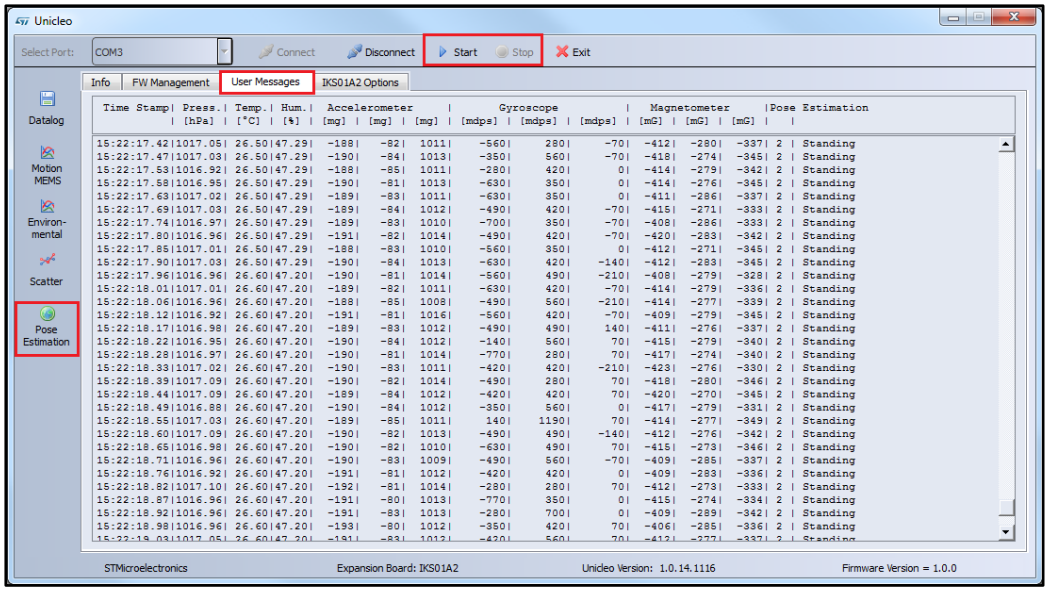
Figure 6: Unicleo main window



3 Start and stop data streaming by using the appropriate buttons on the vertical tool bar.

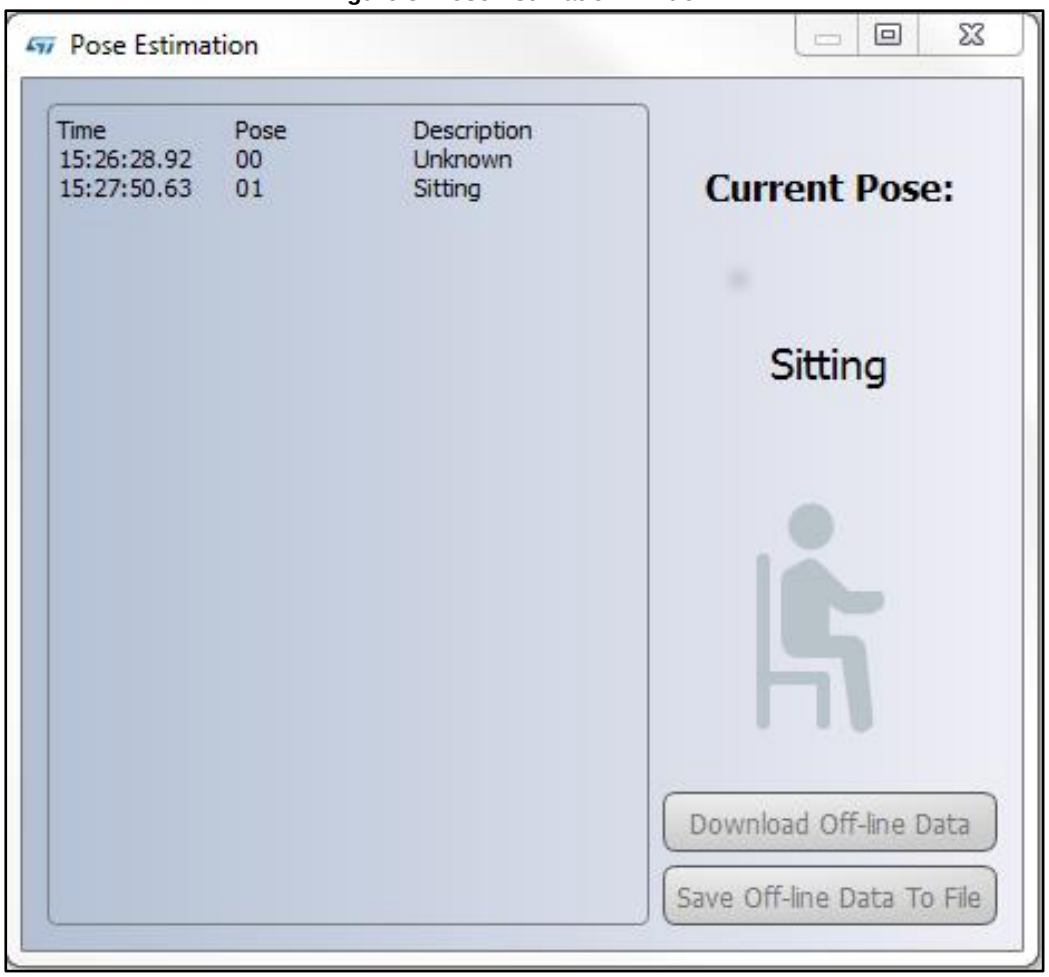
The data coming from the connected sensor can be viewed in the User Messages tab.

Figure 7: User Messages tab



4 Click on the Pose Estimation icon in the vertical tool bar to open the dedicated application window.

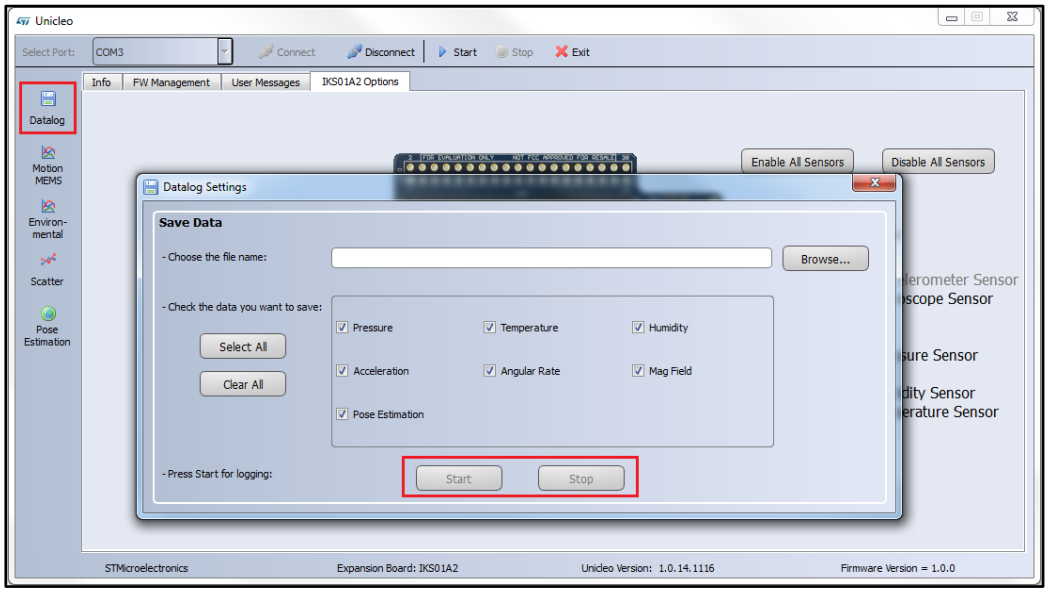
Figure 8: Pose Estimation window



5 Click on the Datalog icon in the vertical tool bar to open the datalog configuration window.

Here, you can select which sensor and activity data to save in files. Saves can be started or stopped by clicking on the corresponding button.

Figure 9: Datalog window



1.5.4 Data storage

The sample application allows the user to estimate the current pose of the user and store it in MCU Flash memory. Data is automatically saved every 10 minutes to avoid losing too much due to 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, 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 pose detection 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 275 hours (user changes pose and a new buffer is stored every 60 seconds)
- a maximum of more than 2700 hours (pose never changes during acquisition sessions and data is automatically stored every 10 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 60 seconds to 30 hours if data is automatically saved every ten 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 five seconds. LED LD2 switches OFF and then starts blinking to indicate that acquisition mode has been activated and the pose data stored in the MCU has been erased.

If LED LD2 does not switch OFF after this operation, the license number is wrong and no action is allowed. The user must program the STM32 Nucleo board with the correct license number (see *2* for details on how to obtain a license).

2 Acronyms and abbreviations

Table 2: 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 environment
NED	x-North, y-East, z-Down
SEU	x-South, y-East, z-Up

3 References

1. *UM1859: Getting started with the X-CUBE-MEMS1 motion MEMS and environmental sensor software expansion for STM32Cube*
2. *UM2012: osxMotionXX system setup*
3. *DB3057: Real-time pose estimation software expansion for STM32Cube*
4. *UM1724: STM32 Nucleo-64 boards*
5. *UM2128: Unicleo-GUI*

4 Revision history

Table 3: Document revision history

Date	Revision	Changes
08-Nov-2016	1	First release.

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.