UM2144
User manual

Getting started with osxMotionAC accelerometer calibration library for X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The osxMotionAC add-on software package for X-CUBE-MEMS1 software runs on the STM32 and includes drivers that recognize the inertial sensors. It provides real-time accelerometer calibration through offset and scale factor coefficients used to correct accelerometer data.

The algorithm is provided in static library format and is designed to be used on STM32 microcontrollers based on the ARM Cortex-M3 or ARM Cortex-M4 architecture.

It is built on top of the STM32Cube software technology for portability across different STM32 microcontrollers.

The software comes with sample implementations running on the X-NUCLEO-IKS01A1 (with optional STEVAL-MKI160V1) or X-NUCLEO-IKS01A2 expansion board on a NUCLEO-F401RE or NUCLEO-L476RG development board.

Contents

List of figures

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

1.1 osxMotionAC overview

This software is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller. It extends STM32Cube with a board support package (BSP) for the sensor expansion board and middleware components for serial communication with a PC.

The drivers abstract low-level details of the hardware and allow the middleware components and applications to access sensor data in a hardware independent fashion.

The osxMotionAC real-time software acquires data from the accelerometer and counts the offset and scale factor coefficients together with the calibration quality value. The Offset and scale factor coefficients are then used to compensate raw data coming from accelerometer.

The osxMotionAC package includes a sample application that developers can use to start experimenting with code that enables sensor data logging on a PC.

The key package features include:

- Real-time accelerometer calibration algorithm (under Open.MEMS license) based exclusively on accelerometer data.
- Complete middleware to build applications on top of X-CUBE-MEMS1.
- Sample application to transmit real time sensor data to a PC.
- Easy portability across different MCU families, thanks to STM32Cube.
- PC-based Windows application to log sensor data.
- Free user-friendly license terms
- Sample implementations available on X-NUCLEO-IKS01A2 and X-NUCLEO-IKS01A1 (with optional STEVAL-MKI160V1) expansion boards, mounted on a NUCLEO-F401RE or NUCLEO-L476RG development board.

The osxMotionAC 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 be requested from ST and included in the project (and become part of the build process) prior to usage. The resulting firmware binary image is therefore node-locked.

For complete information about the open.MEMS license agreement, please refer to the license file located in the Middlewares/ST/STM32_OSX_MotionAC_Library folder.

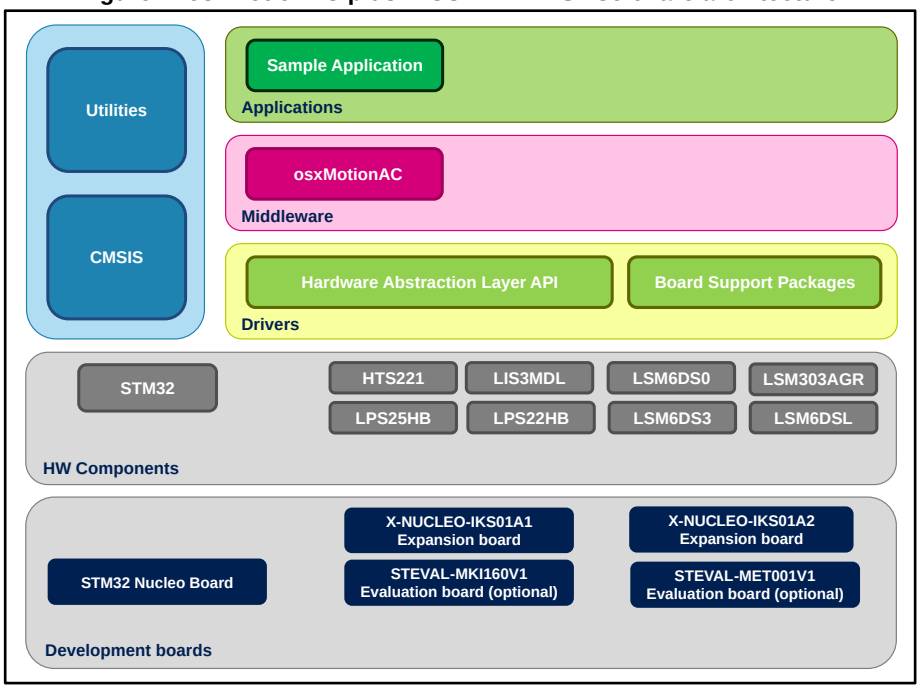
1.2 osxMotionAC architecture

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

- **STM32Cube HAL layer**: consists of simple, generic and multi-instance APIs (application programming interfaces) which interact with the upper layer applications, libraries and stacks. These generic and extension APIs are based on a common framework so that overlying layers like middleware can function without requiring specific microcontroller unit (MCU) hardware information. This structure improves library code reusability and guarantees easy portability across other devices.
- **Board support package (BSP) layer**: provides software support for the STM32 Nucleo board peripherals, excluding the MCU. These specific APIs provide a programming interface for certain board specific peripherals like LEDs, user buttons,

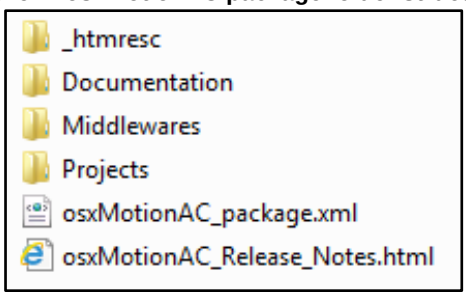
etc. and can also be used to fetch individual board version information. It also provides support for initializing, configuring and reading data.

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



1.3 osxMotionAC folder structure

Figure 2: osxMotionAC 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 osxMotionAC static library binary code, the library header file, documentation, license information plus header file for node-locked license validation.
- **Projects**: contains a sample application for the NUCLEO-F401RE or NUCLEO-L476RG development platforms to access sensor data and calibration coefficients in the IAR Embedded Workbench for ARM, µVision (MDK-ARM) toolchain and System Workbench for STM32 integrated development environments.

1.4 osxMotionAC library

Detailed technical information fully describing the functions and parameters of the osxMotionAC APIs can be found in the osxMotionAC_Package.chm compiled HTML file the package Documentation folder.

The osxMotionAC 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 be requested from ST and included in the project (and become part of the build process) prior to usage. The resulting firmware binary image is therefore node-locked.

For complete information about the open.MEMS license agreement, please refer to the license file located in the Middlewares/ST/STM32_OSX_MotionAC_Library folder.

1.4.1 osxMotionAC library description

The osxMotionAC accelerometer calibration library manages data acquired from accelerometer; it features:

- offset compensation up to 0.2 g
- scale factor compensation, in range from 0.8 to 1.2 in every direction
- update frequency from 20 to 100 Hz
- occupies 15 kB of code and 3 kB of data memory
- the library is available for ARM Cortex-M3 and Cortex-M4 architectures

1.4.2 osxMotionAC APIs

The exposed APIs of the osxMotionAC library are listed below:

- `uint8_t osx_MotionAC_GetLibVersion(char *version)`
  - retrieves the version of the library
  - `*version` is a pointer to an array of 35 characters
  - returns number of characters in the version string
- `uint8_t osx_MotionAC_Initialize(intsampleTime_ms)`
  - performs osxMotionAC library initialization and setup of the internal mechanism used for node-locking (see *Section 4: "References"*). This function must be called before using the accelerometer calibration library.
  - parameter `sampleTime_ms` is the 10 to 50 ms interval between update function calls.
  - returns 1 for correct initialization or 0 otherwise (e.g., 0 is returned for license errors)
- `void osx_MotionAC_Update(intx_mG, inty_mG, intz_mG, inttimeStamp_ms)`
  - executes accelerometer calibration algorithm
  - this function must be called at the same frequency indicated in the initialization function
  - parameters `x_mG, y_mG, z_mG` represent X, Y and Z axis acceleration in 10^{-3} g (milli-g)
  - parameter `timeStamp_ms` is the timestamp for accelerometer output values
- `osx_MAC_CalQuality_t osx_MotionAC_GetCalParams(int *bias_mG, float SF[][3])`
  - retrieves the accelerometer calibration coefficients for offset and scale factor compensation and calibration quality factor
  - `*bias_mG` is a pointer to an array of 3 elements containing the offset for each axis, the unit is 10^{-3} g

  - `float SF` is the scale factor correction 3x3 matrix (diagonal matrix)
  - returns calibration quality factor:
    - OSX_MAC_CALQSTATUSUNKNOWN = 0; accuracy of calibration parameters are unknown
    - OSX_MAC_CALQSTATUSPOOR = 1; accuracy of calibration parameters are poor, cannot be trusted
    - OSX_MAC_CALQSTATUSOK = 2; accuracy of calibration parameters are OK
    - OSX_MAC_CALQSTATUSGOOD = 3; accuracy of calibration parameters are good.

1.4.3 Storing and loading calibration parameters

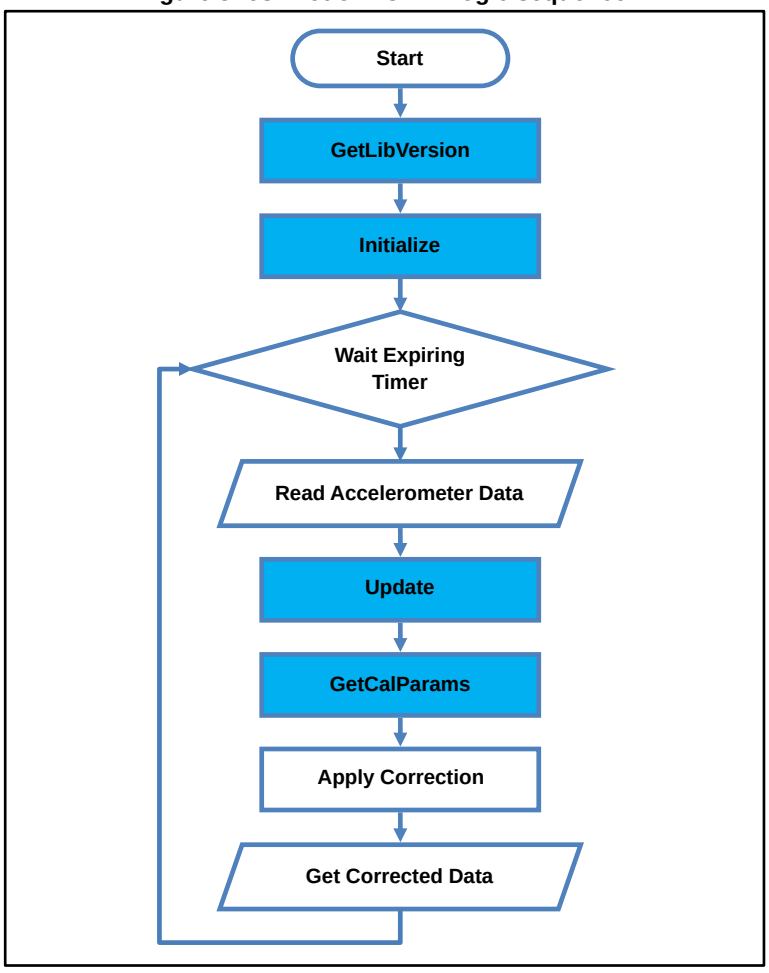
The following functions have to be implemented specifically for each target platform:

- `char osx_MotionAC_LoadCalFromNVM(unsigned short intdataSize, unsigned int *data)`
  - the function is used to retrieve the accelerometer calibration parameters from storage
  - `dataSize` is the data size
  - `*data` is the data location pointer
  - returns 0 for correct loading, 1 otherwise
- `char osx_MotionAC_SaveCalInNVM(unsigned short intdataSize, unsigned int *data)`
  - the function is used to save the accelerometer calibration parameters in storage
  - `dataSize` is the data size
  - `*data` is the data location pointer
  - returns 0 for correct loading, 1 otherwise

These functions need to be implemented but should not be called; the accelerometer calibration library decides when to call these functions. They may be implemented as empty (always return 0) if saving and loading calibration coefficients is not needed.

1.4.4 API flow chart

Figure 3: osxMotionAC API logic sequence



1.4.5 Accelerometer calibration demo code

The following demonstration code reads data from accelerometer sensor in 10^{-3} g (milli-g) (`raw_x, raw_y, raw_z`) and calculate compensated data in 10^{-3} g (`cal_x, cal_y, cal_z`).

```
/****************** Init phase ******************/
// Set update period, 40 ms -> 25 Hz
osx MotionAC Initialize(40);
/************************************************/

Timer_OR_DataRate_Interrupt_Handler()
{
  int16 raw x, raw y, raw z;
  int16 cal x, cal y, cal z;
  osx_MAC_CalQuality_t goodness;
  float sf[3][3];  int16 bias[3];

  //Get x,y,z in mg
  MEMS_Read_AccValue(&raw_x, &raw_y, &raw_z);

  //Update algorithm
  osx MotionAC Update(raw x, raw y, raw z, timestamp ms);
```

```
  //Get correction
  goodness = osx MotionAC GetCalParams(bias, sf);

  //Apply correction
  cal x = (int16) ((raw x - bias[0])* sf[0][0]);
  cal y = (int16) ((raw y - bias[1])* sf[1][1]);
  cal z = (int16) ((raw z - bias[2])* sf[2][2]);
}
```

1.4.6 Calibration process

This calibration algorithm uses the normal motion of the three orthogonal axes of a stationary accelerometer sensor exposed to Earth's gravitation field.

1 Hold the device firmly as shown in position 1.

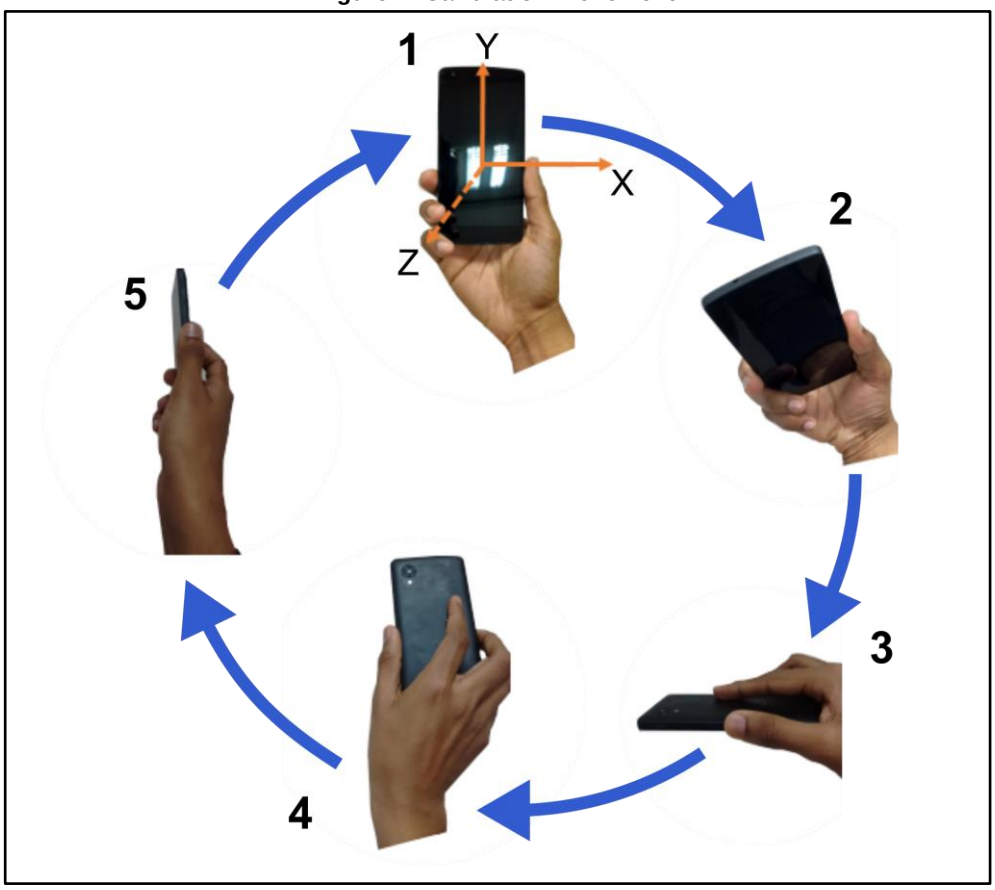
2 Gently rotate the device by 180° around the YZ plane such that in position 4, the device is flipped to its back side.

3 Rotate the device by 180° in a clockwise fashion around the XZ plane to reach position 1.

Try to rotate the device along a smooth path and at a constant speed.

You can also perform standard six point calibration, holding the module stationary in six different directions (positive and negative X,Y and Z directions).

Figure 4: Calibration movement



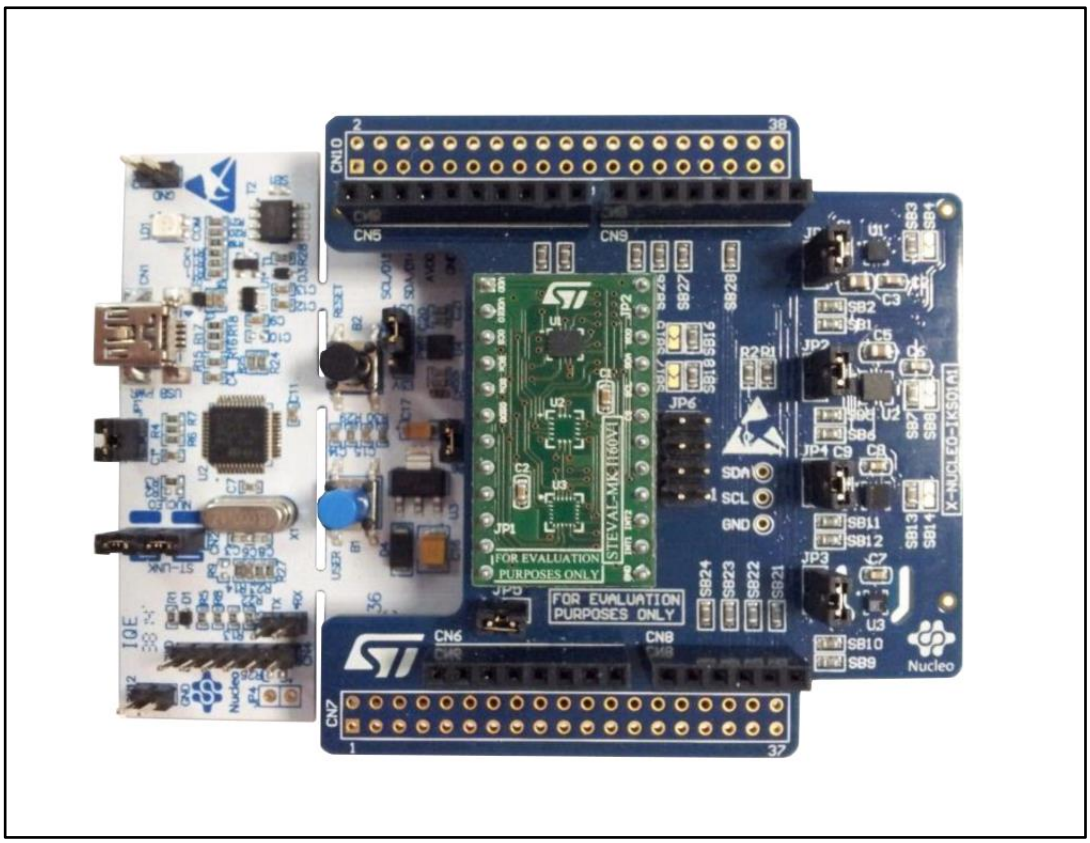
2 Sample application

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

A sample application example in the Projects folder is designed to run on either:

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

Figure 5: STM32 Nucleo development board plus X-NUCLEO-IKS01A1 and STEVAL-MKI160V1 boards



Accelerometer algorithm output data may be displayed in real time through a specific GUI.

2.1 Unicleo-GUI utility

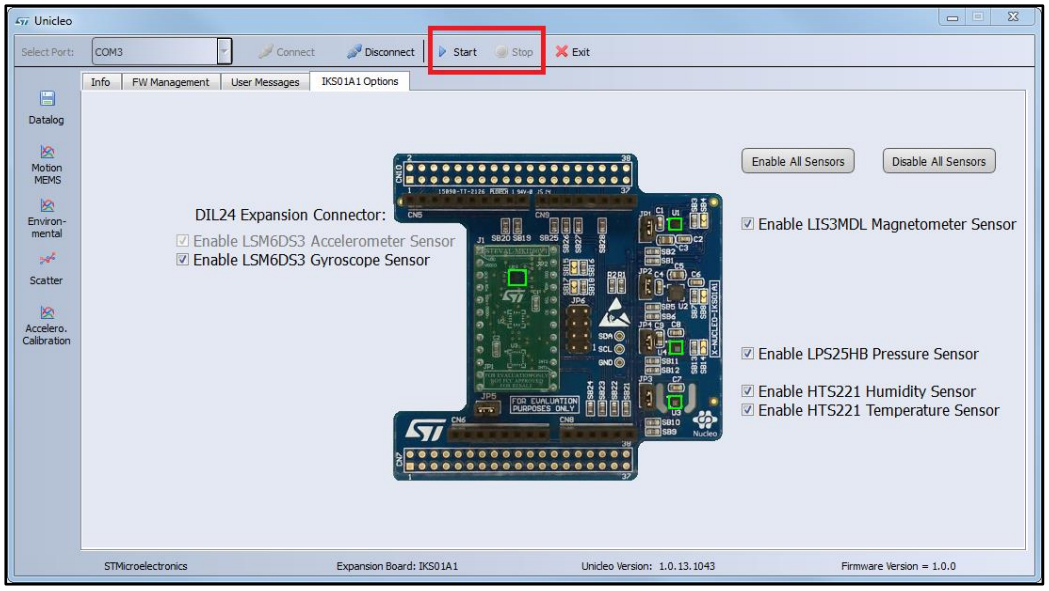
The osxMotionAC software package for STM32Cube uses the Windows Unicleo-GUI utility, which can be downloaded from *www.st.com* (see *5*).

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.

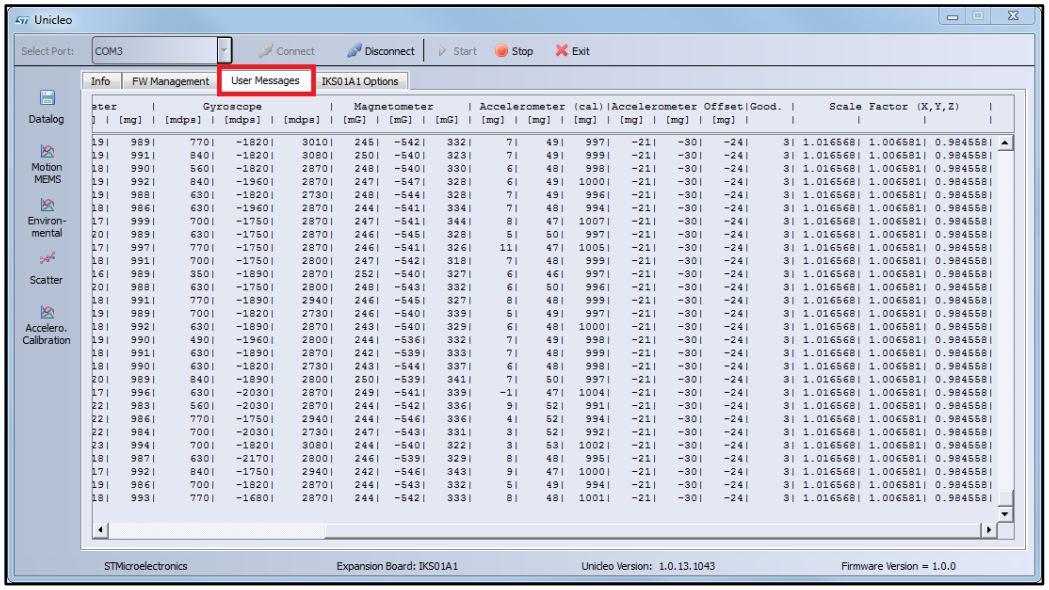
Figure 6: Unicleo main window



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.

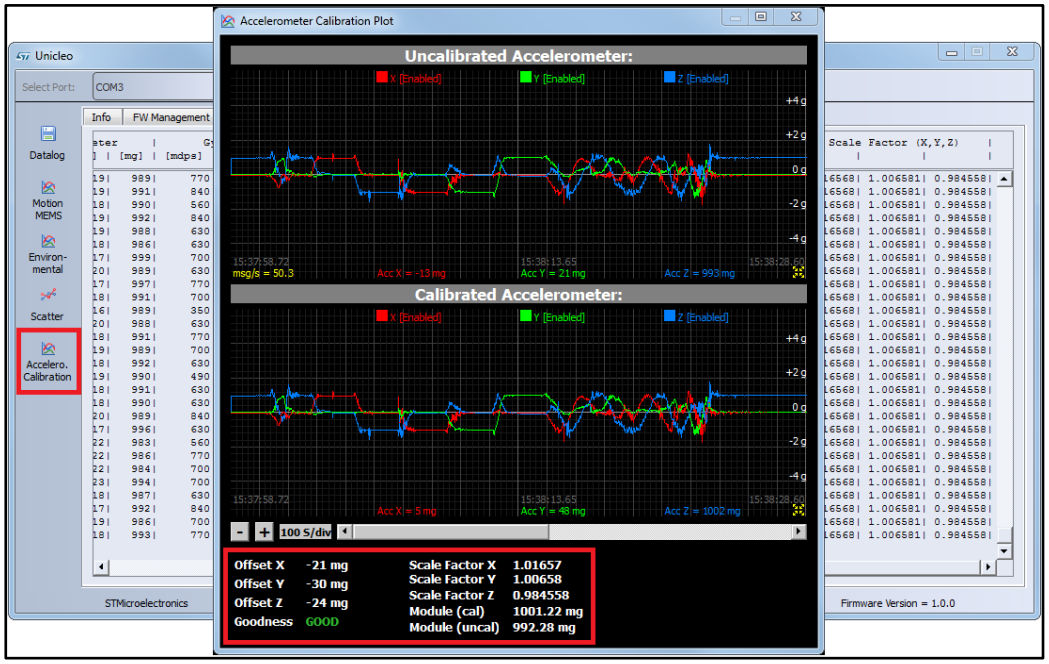
Figure 7: User Messages tab



4 Click on the Accelerometer Calibration icon in the vertical tool bar to open the dedicated application window.

The window is split into one section with uncalibrated data, another with the calibrated data and another section with offset, scale factor and quality of calibration information.

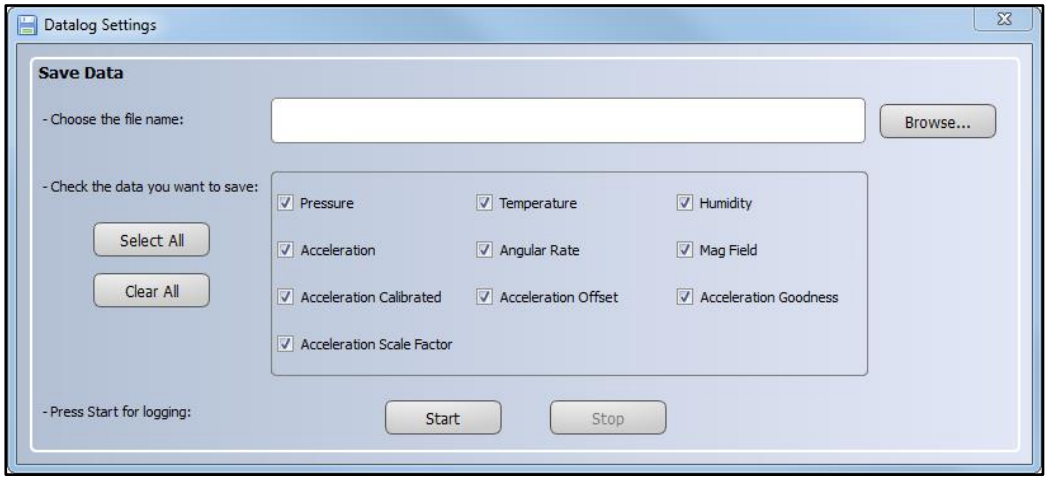
Figure 8: Activity recognition for Wrist window



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



3 References

1. UM1859: Getting started with the X-CUBE-MEMS1 motion MEMS and environmental sensor software expansion for STM32Cube
2. DB3121: Real-time accelerometer calibration software expansion for STM32Cube
3. UM2012: osxMotionXX system setup
4. UM1724: STM32 Nucleo-64 boards
5. UM2128: Unicleo-GUI

4 Revision history

Table 1: Document revision history

Date	Version	Changes
18-Nov-2016	1	Initial 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.