Getting started with osxMotionID motion intensity detection library for X-CUBE-MEMS1 expansion for STM32Cube

Introduction

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

By outputting an intensity value in the range [0:10] (i.e., from doing nothing to sprinting), this library may be used for real-time evaluation of activities for the accurate calculation of calories or to distinguish between active and non-active stationary status.

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

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

Contents

List of tables

List of figures

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

1.1 osxMotionID overview

The osxMotionID 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 motion intensity detection algorithm quantifies how active the user is, which can for instance be used to accurately determine calorie consumption. The output is an integer ranging from zero, or no activity, to ten, or maximum effort, like sprinting.

This library can be used in conjunction with other motion algorithms like osxMotionAW to actually identify the type of motion being performed.

The package is built on the STM32Cube software platform to facilitate portability across different STM32 microcontrollers. The algorithm manages the data acquired exclusively from the accelerometer at the low sampling frequency of 16Hz to reduce the power consumption of the host platform.

The key package features include:

- Real-time motion intensity detection algorithm (under OpenSoftwareX license) based on accelerometer data only, designed 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) expansion boards, mounted on a NUCLEO-F401RE or NUCLEO-L476RG development board.

The osxMotionID 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, please refer to *2*.
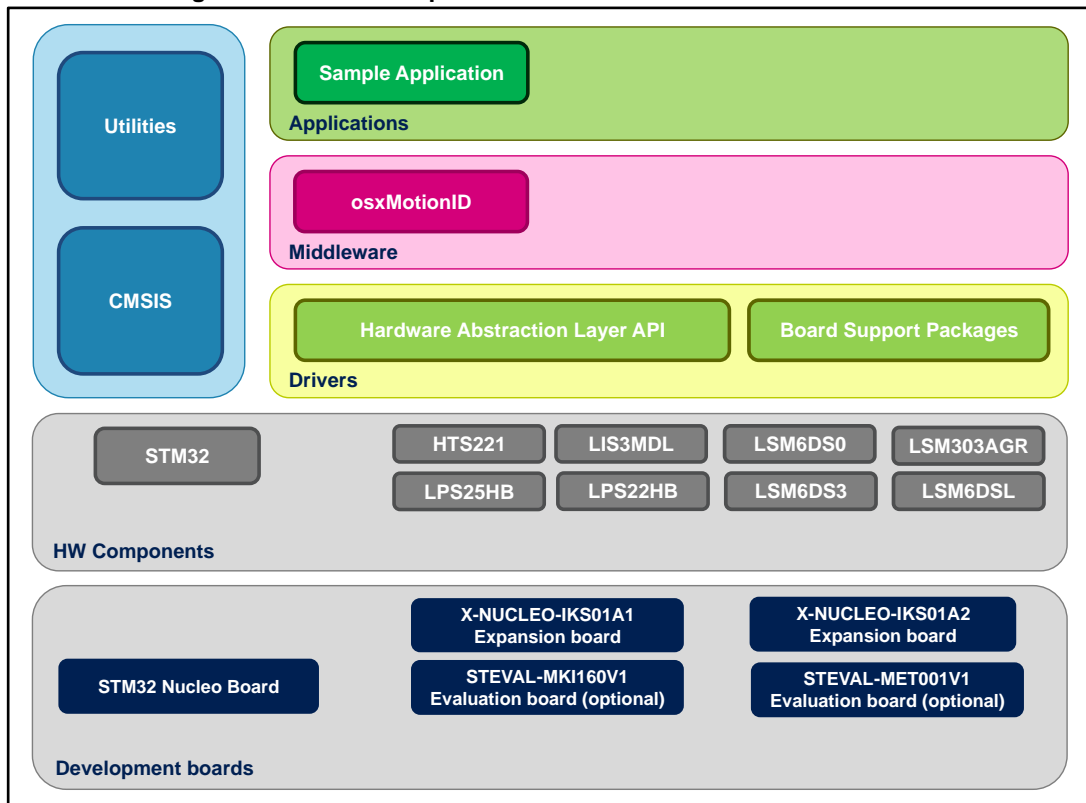
1.2 osxMotionID architecture

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). These generic and extension APIs are built on a generic architecture and allows overlying layers like middleware to function without the specific hardware configuration of 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 the 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:
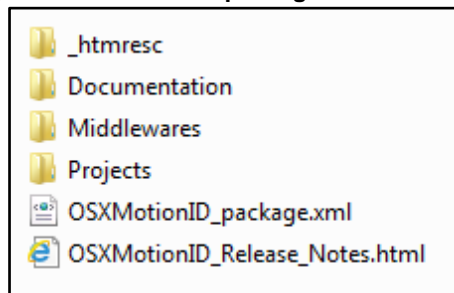
Figure 1: osxMotionID plus X-CUBE-MEMS1 software architecture

1.3 osxMotionID folder structure

The image below outlines the package file system architecture:

Figure 2: osxMotionID 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 osxMotionID 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 osxMotionID APIs

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

The osxMotionID 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_MotionID_Library folder.

1.4.1 osxMotionID library

The osxMotionID is a real time motion intensity detection 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 osxMotionID library are listed below:

- `uint8_t osx_MotionID_GetLibVersion(char *version);`
  - retrieves the revision of the included core engine;
- `uint8_t osx_MotionID_Initialize(void);`
  - performs osxMotionID 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);
- `osx_MPE_output_t osx_MotionID_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 detected intensity values.
- `void osx_MotionID_ResetLib(void);`
  - Resets the algorithm.

1.5 Sample application

The osxMotionID 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) optionally with STEVAL-MKI160V1 board (based on LSM6DS3) plugged in, or a NUCLEO-F401RE or a NUCLEO-L476RG board connected to an X-NUCLEO-IKS01A2 (based on LSM6DSL).

The application estimates in real-time the intensity of the activity performed by the user, and output a value in the range [0:10].

Two working modes are available for the application example: the *stand-alone* mode and the *PC GUI driven* mode, described below.

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 3: "NUCLEO-F401RE board details"*), ensuring that jumper JP1 is fitted.

Figure 3: NUCLEO-F401RE board details
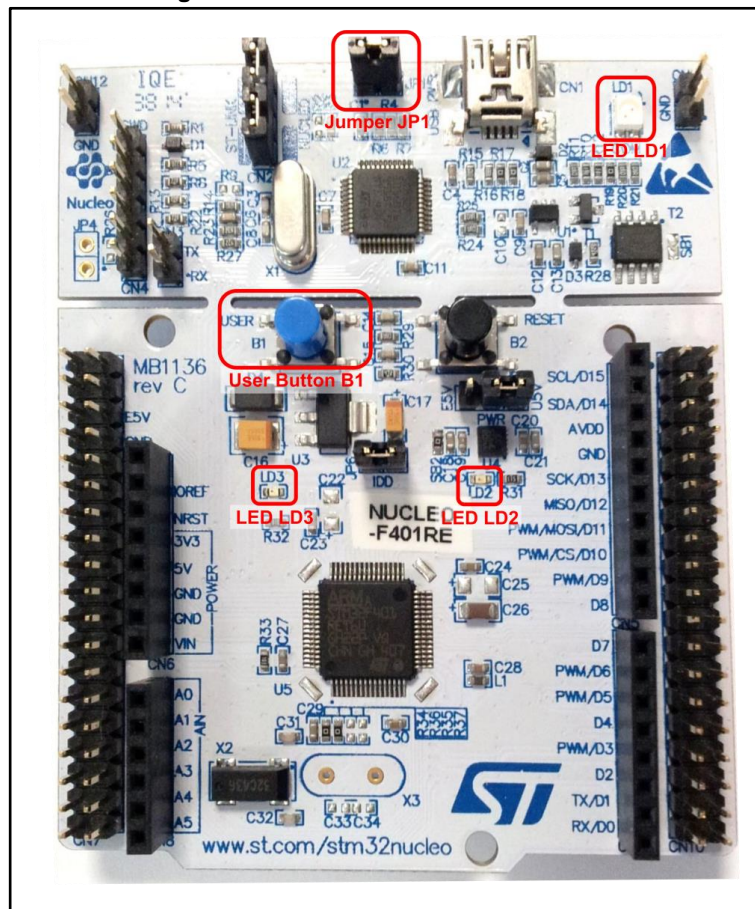
The above figure shows details of the NUCLEO-F401RE board, which has a similar layout to the NUCLEO-L476RG board. Once the board is powered, LED LD3 (PWR) turns on and the tricolour LED LD1 (COM) begins blinking slowly due to the missing USB enumeration (refer to User manual "UM1724: STM32 Nucleo-64 boards" for further details).

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 motion intensity; 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 intesity code according to a sequence of flashes described in *Table 1: "Intensity value 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 4: "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.

Intensity value are grouped for storage per ranges as follows:

- intensity values in the range [0:2] are stored as *low intensity*;
- intensity values in the range [3:6] are stored as *medium intensity*;
- intensity values in the range [7:8] are stored as *high intensity*;
- intensity values in the range [9:10] are stored as *sprinting*.

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: Intensity value codes

Intensity value	LED LD2 blinking sequence	intensity value
0	0	low intensity
1	1	low intensity
2	2	low intensity
3	3	medium intensity
4	4	medium intensity
5	5	medium intensity
6	6	medium intensity
7	7	high intensity
8	8	high intensity
9	9	sprinting
10	10	sprinting

Figure 4: State machine
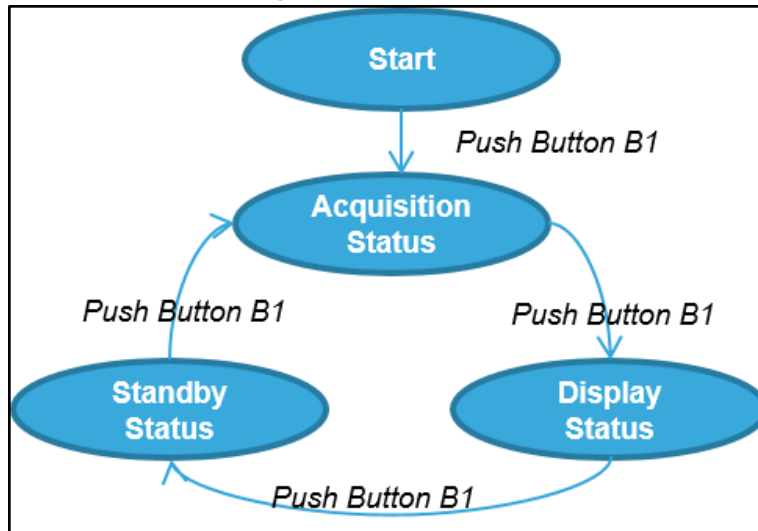
1.5.2 PC GUI driven mode

In this working mode, the STM32 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 motion intensity, accelerometer data, time stamp and any other real-time sensor data using the Unicleo-GUI utility.

Once 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 osxMotionID 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 5: Unicleo main window
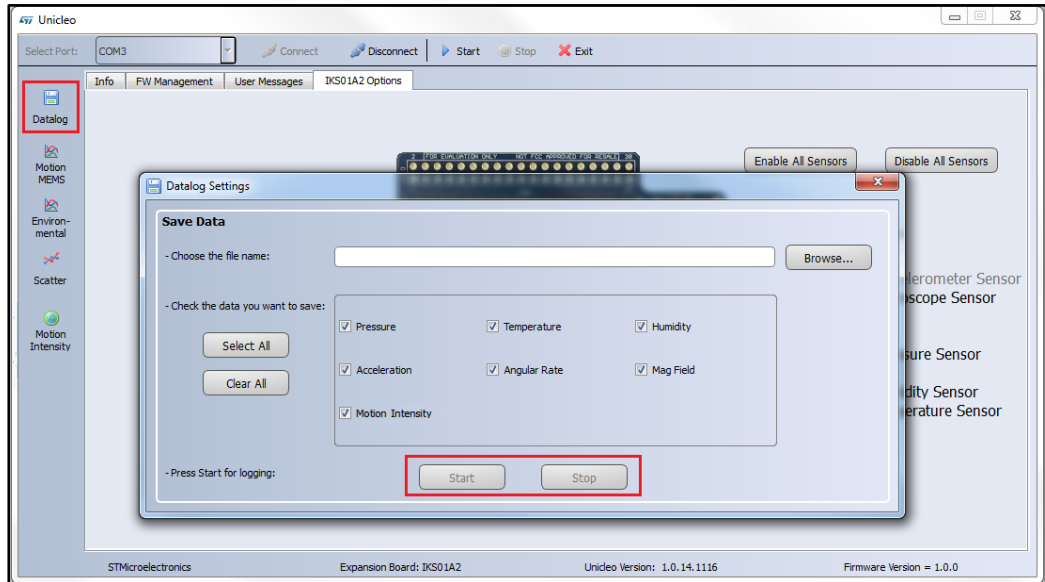
3   Start and stop data streaming by using the appropriate buttons on the vertical tool bar.

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

Figure 6: User Messages tab
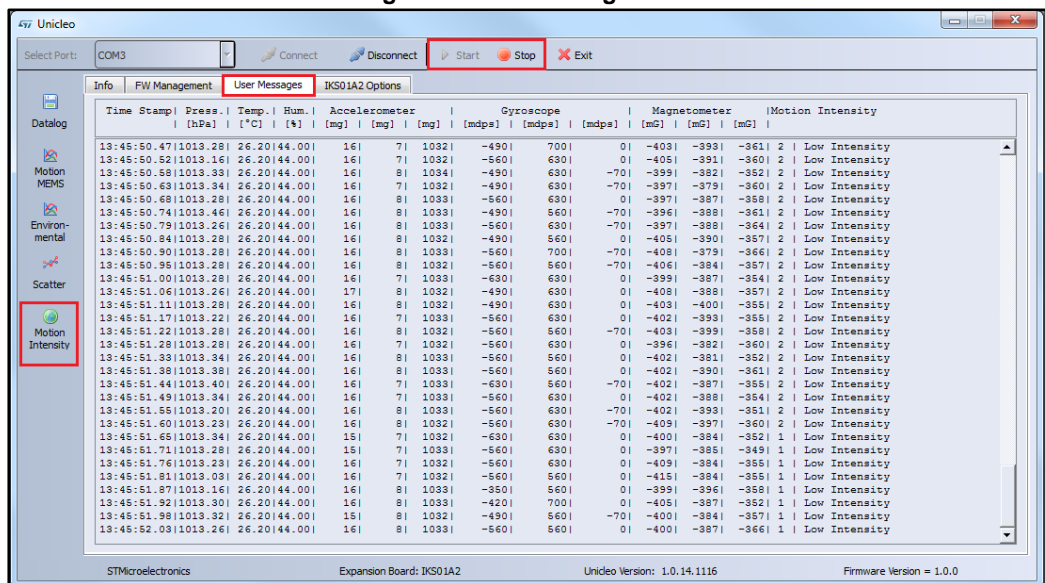
4   Click on the Motion Intensity icon in the vertical tool bar to open the dedicated
application window.

Figure 7: Motion Intensity Detection window
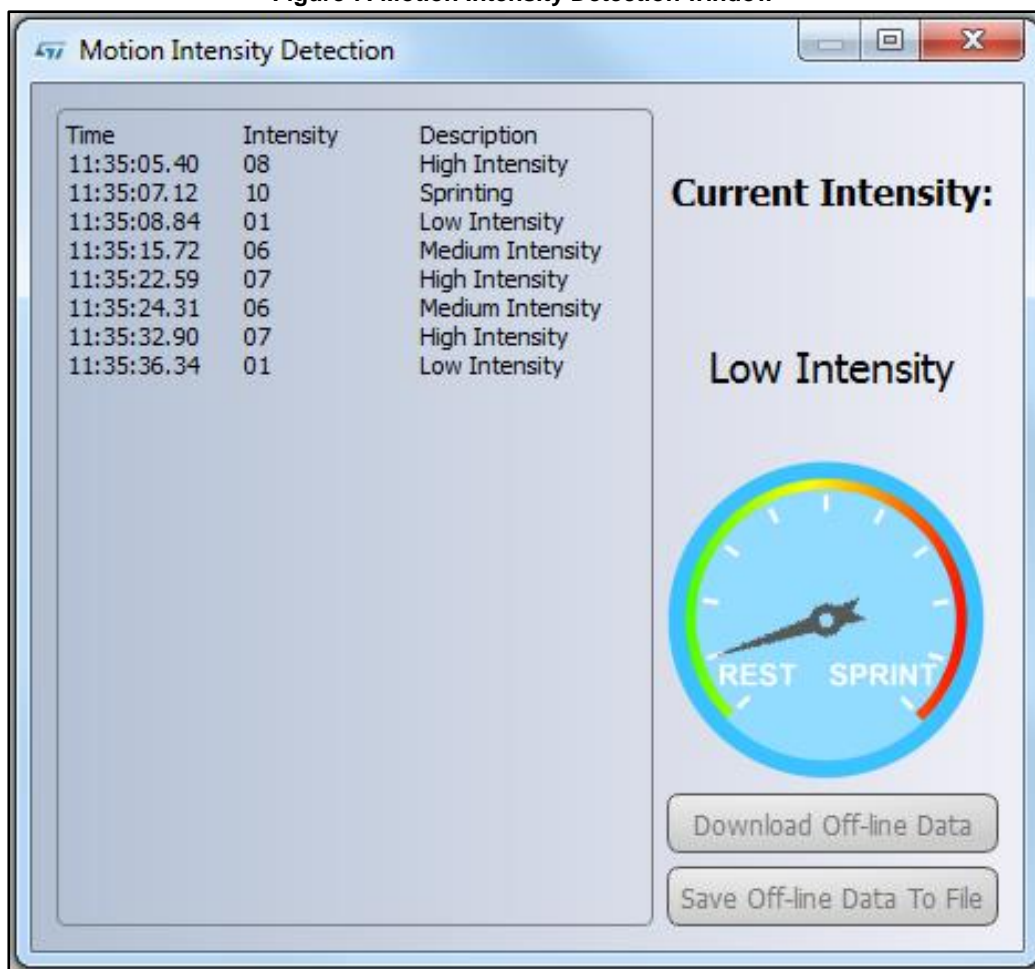
5   Click on the Datalog icon in the vertical tool bar to open the datalog configuration
window.

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

Figure 8: Datalog window
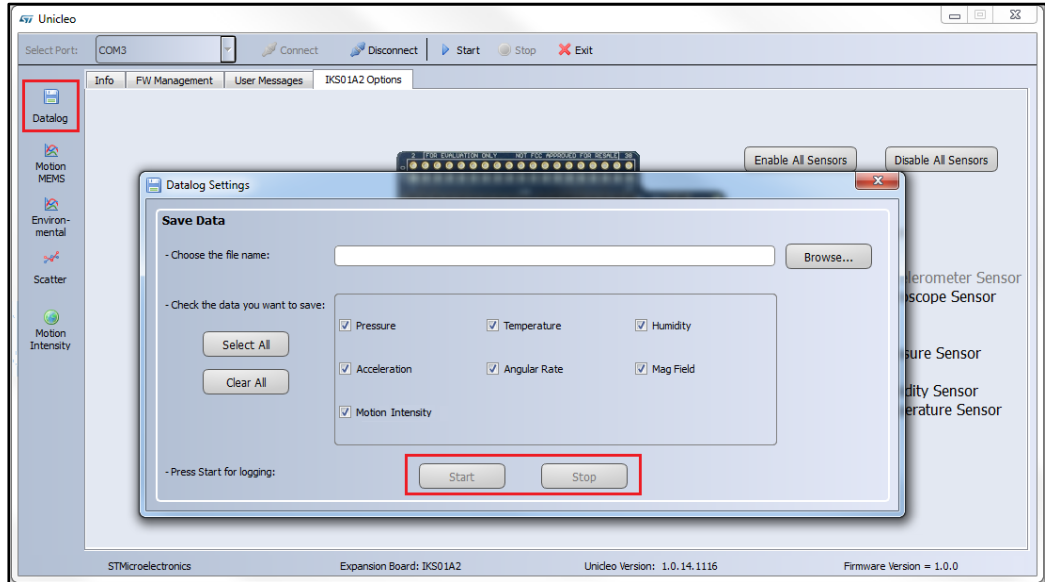
6   Click Download Off-line Data to retrieve stored data.

Figure 9: Download Off-line Data
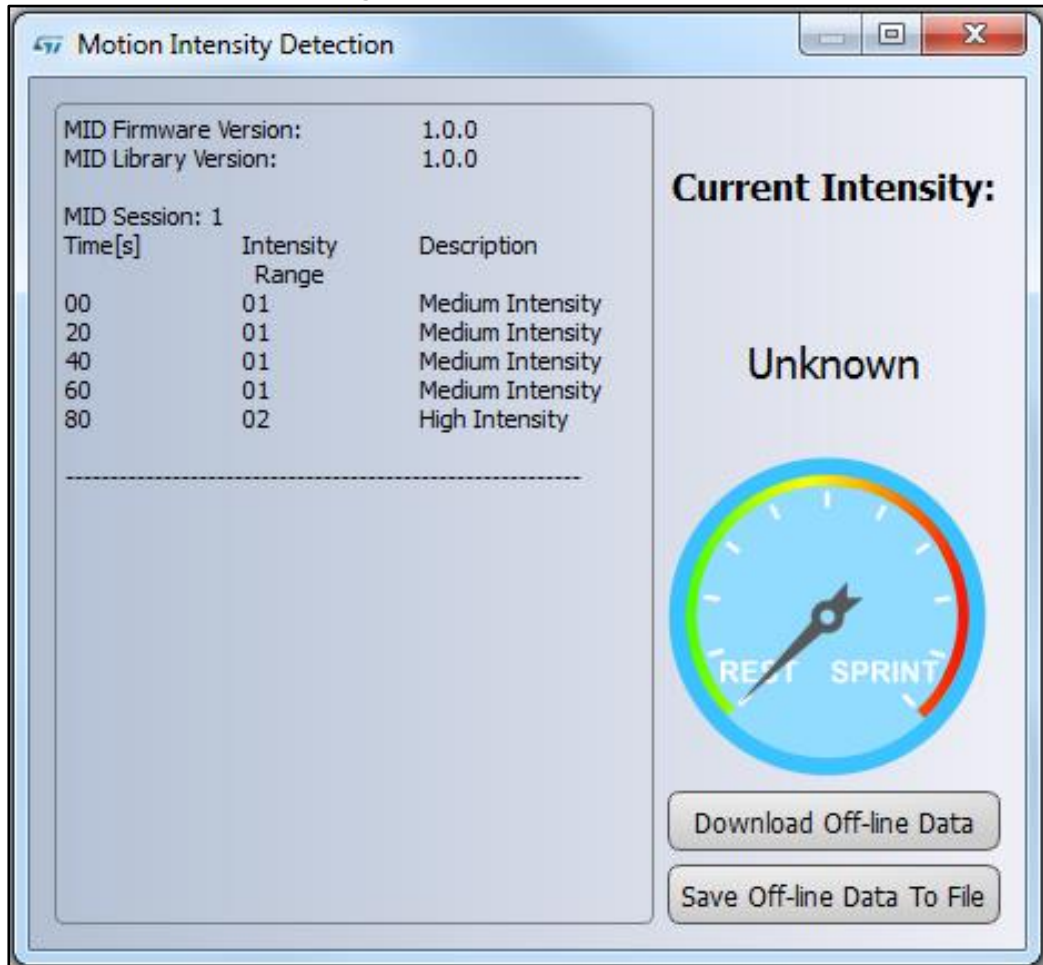
1.5.4   Data storage

The sample application allows the user to estimate the intensity of the activity of the user and store it in MCU Flash memory. Data is automatically saved every five minutes to avoid losing too much data in 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 five 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 for data storage is 128 KB, allowing memorization of more than 16,000 data sets, this equates to:

- a minimum of 130 hours (user modifies intensity and a new buffer is stored every 20 seconds)
- a maximum of more than 1300 hours (intensity 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 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 five seconds. LED LD2 switches OFF and then starts blinking to indicate that acquisition mode has been activated and the intensity 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

- *UM1859: Getting started with the X-CUBE-MEMS1 motion MEMS and environmental sensor software expansion for STM32Cube*
- *UM2012: osxMotionXX system setup*
- *DB3069: Real-time motion intensity detection software expansion for STM32Cube*
- *UM1724: STM32 Nucleo-64 boards*
- *UM2128: Unicleo-GUI*

4 Revision history

Table 3: Document revision history

Date	Revision	Changes
09-Nov-2016	1	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY