UM2162
User manual

Getting started with osxMotionGC gyroscope calibration library for X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The osxMotionGC add-on software package for X-CUBE-MEMS1 software runs on STM32 Nucleo and includes drivers for recognizing inertial sensors. It provides real-time gyroscope calibration through angular zero-rate level coefficients (offset) used to correct gyroscope 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 STM32Cube software technology that ease portability across different STM32 microcontrollers.

The software comes with sample implementation running on 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 tables

List of figures

1 Acronyms and abbreviations

Table 1: List of acronyms

Acronym	Description
API	Application programming interface
BSP	Board support package
GUI	Graphical user interface
HAL	Hardware abstraction layer
IDE	Integrated development environments

2 osxMotionGC library add-on to X-CUBE-MEMS1 software expansion for STM32Cube

2.1 osxMotionGC overview

This software is based on the STM32CubeHAL hardware abstraction layer for the STM32 microcontroller. The package extends STM32Cube by providing a board support package (BSP) for the sensor expansion board and some 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 osxMotionGC real-time software acquires data from the accelerometer and gyroscope and counts the angular zero-rate level coefficient (offset) for each axis. The angular zero-rate level coefficients are then used to compensate raw data coming from gyroscope.

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

The key package features include:

- Real-time gyroscope calibration algorithm (under OPEN.MEMS license) based on gyroscope and accelerometer data
- Complete middleware to build applications on top of X-CUBE-MEMS1
- Sample application for transmitting 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 implementation 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

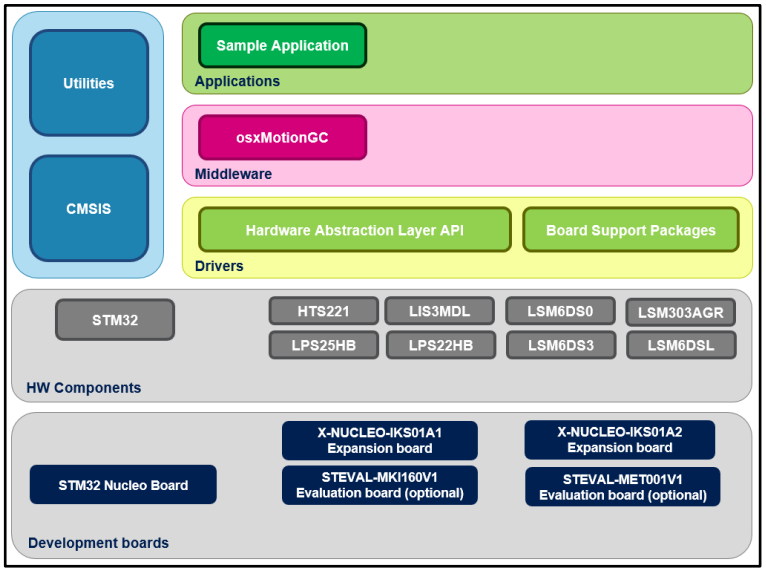
For further details on system setup, installation and hardware components, please refer to *Section 3: "References".*

2.2 osxMotionGC architecture

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

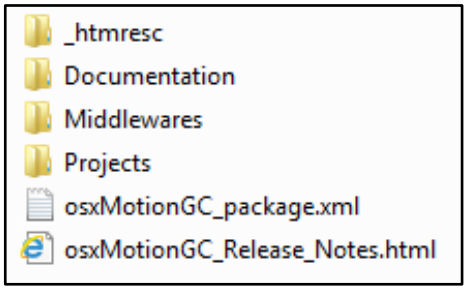
- STM32Cube HAL layer: the HAL driver layer provides a generic, multi-instance, simple set of APIs (application programming interfaces) to interact with the upper layers (application, libraries and stacks). It is composed of generic and extension APIs. It is directly built around a generic architecture and allows overlying layers like middleware to implement functions without depending on the specific hardware configuration for a given microcontroller unit (MCU). This structure improves library code reusability and guarantees easy portability across devices.
- Board support package (BSP) layer: the software package supports all the available peripherals on the STM32 Nucleo board, except the MCU. A limited set of APIs provides a programming interface for specific peripherals (LED, user button, etc.), which helps to identify the specific board version. If the sensor expansion board is used, it provides the programming interface for various inertial and environmental sensors and supports sensor data initialization and reading.

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



2.3 osxMotionGC folder structure

Figure 2: osxMotionGC package folder structure



The folders included in the package are:

- Documentation: contains a compiled HTML file generated from the source code and detailing the software components and APIs (doxygen).
- Middlewares: contains the osxMotionGC static library binary code, the library header file, documentation, license information and the header file for node-locked license validation.
- Projects: contains a sample application used to access sensor data and calibration coefficients, provided for the NUCLEO-F401RE and NUCLEO-L476RG platforms according to IDE (Integrated Development Environments) proprietary formats (IAR Embedded Workbench for ARM, µVision (MDK-ARM) toolchain, System Workbench for STM32 Nucleo).

2.4 osxMotionGC library

Technical information fully describing the functions and parameters of the osxMotionGC APIs can be found in the osxMotionGC_Package.chm compiled HTML file located in the 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, refer to the license file located in the Middlewares/ST/STM32_OSX_MotionGC_Library folder.

2.4.1 osxMotionGC library description

The osxMotionGC gyroscope calibration library manages data acquired from accelerometer and gyroscope; it features:

- the angular zero-rate level (offset) compensation
- adjustable thresholds to detect device in still state
- adjustable maximum angular zero-rate level to be compensated
- possibility to set previously saved coefficients as the initial values
- fast start-up option
- able to run with multiple sample rate from 25 Hz to 200 Hz.
- occupies 2.8kByte of code memory and 0.2kByte of data memory

Real size might differ for different IDE (toolchain) library versions

- available for Cortex-M3 and Cortex-M4 architectures.

2.4.2 osxMotionGC APIs

The osxMotionGC library APIs are:

- `uint8_t osx_MotionGC_GetLibVersion(char *version)`
  - retrieves the version of the library
  - `*version` is a pointer to an array of 35 characters
  - returns the number of characters in the version string

- `uint8_t osx_MotionGC_Initialize(float freq)`
  - performs osxMotionGC library initialization and setup of the internal mechanism used for node-locking (see *Section 3: "References"*)

This function must be called before using the gyroscope calibration library.

  - `freq` parameter indicates the sampling frequency
  - returns 1 for correct initialization or 0 otherwise (e.g., 0 is returned for license errors)

- `void osx_MotionGC_SetKnobs(osx_MGC_knobs_t *knobs)`
  - sets the library knobs
  - `*knobs` parameter is a pointer to a structure with settings
  - the parameters for structure type `osx_MGC_knobs_t` are the following:
    - `acc_thr_mg` is the accelerometer threshold to detect steady state in mgs. The default value is 10 mg. The input range is 3 to 50 mg. For higher accuracy, set the value low; for vibration or noisy sensor platforms, set the value high.

  - `gyro_thr_mdps` is the gyroscope threshold to detect steady state in mdps. The default value is 200 mdps. The input range is 8 to 400 mdps. For higher accuracy, set the value low.
  - `k` is the decay constant for the internal filter (from 0 to1). Setting a higher value can cause a variation in the gyro offset but allows a quick convergence. The default value is 0.002.
  - `fast_start` set to 1 immediately starts computing the gyro offset and quickly converges to gyro offset. By setting the value to 1, it is possible that the initial gyro offset value is not accurate but converges over a period of time.
  - `max_gyro_mdsp` is the maximum expected angular zero-rate level when still in mdps. The default value is 15 mdps
  - `max_acc_mg` is the maximum acceleration module when still in mg. The default value is 1.3 mg.

- `void osx_MotionGC_GetKnobs(osx_MGC_knobs_t *knobs)`:
  - gets the library knob setting
  - `*knobs` is a pointer to the setting structure

This function needs to be called before making changes in the settings

- `void osx_MotionGC_Update(int *acc_mg, int *gyro_mdps, int *gyro_bias_mdps, int* bias_update)`:
  - runs the gyroscope calibration algorithm and returns compensation parameters
  - `*acc_mg` is a pointer to the array of accelerometer sensor value is in mg
  - `*gyro_mdps` is a pointer to the the array of gyroscope sensor value in mdps
  - `*gyro_bias_mdps` ia a pointer to the array of actual gyroscope offset value in mdps
  - `*bias_update` is a pointer to an integer set to 1 if the device is in stable state and the gyroscope bias is updated (0 otherwise)

This function has to be called periodically as indicated in the initialization function.

- `void osx_MotionGC_GetCalParams(int *gyro_bias_mdps)`:
  - retrieves the gyroscope compensation parameters
  - `*gyro_bias_mdps` is a pointer to an array of three elements containing the offset for each axis in mdps

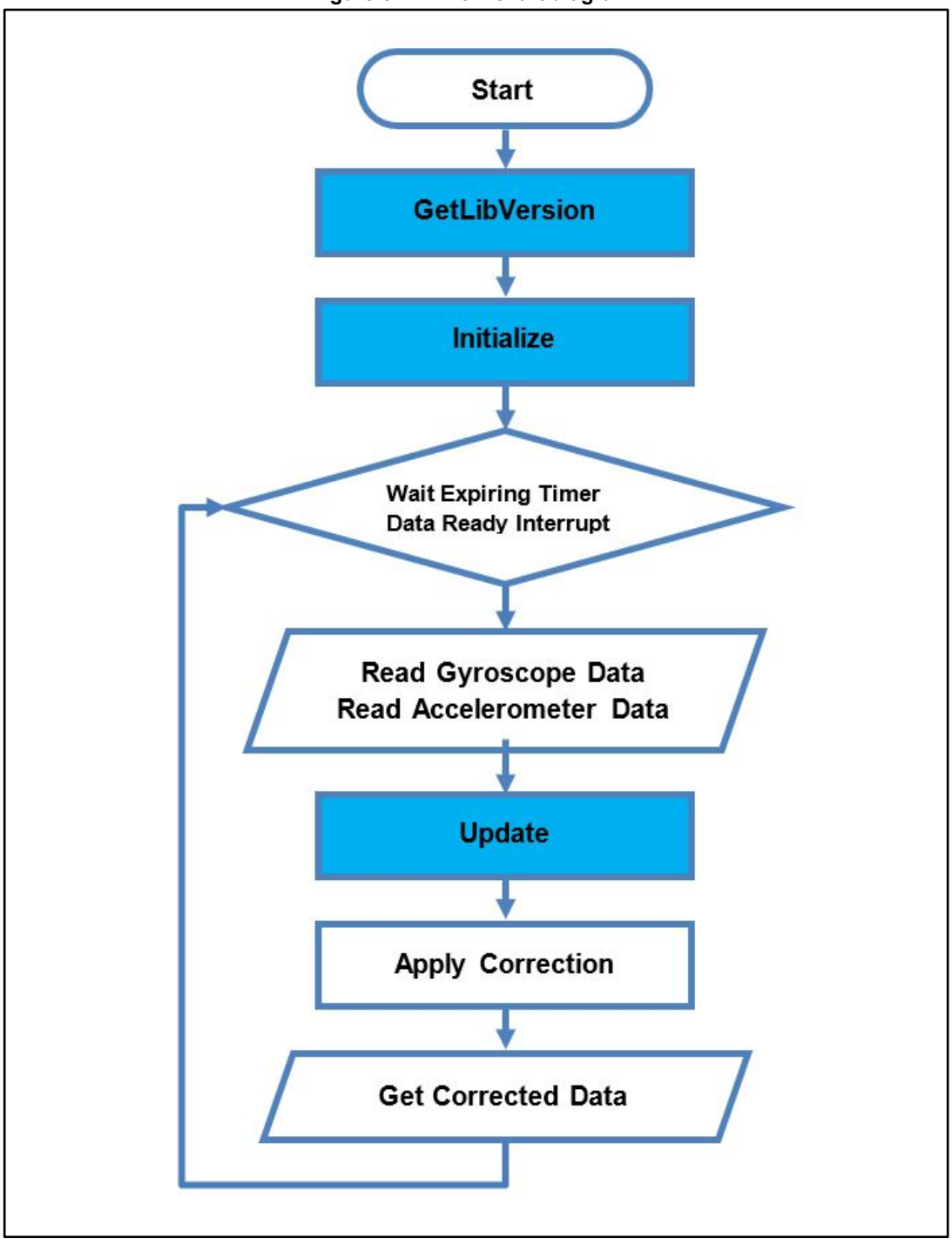
- `void osx_MotionGC_SetCalParams(int *gyro_bias_mdps)`
  - sets the initial gyroscope compensation parameters, helping to quickly converge to the gyro offset right value
  - `*gyro_bias_mdps` is a pointer to an array of three elements containing the offset for each axis in mdps

This function should be called just after `osx_MotionGC_Initialize`

- `void osx_MotionGC_SetFrequency(float freq)`
  - applies the new sample frequency to the library without resetting
  - `freq` indicates the new sample frequency in Hertz

2.4.3 API flow chart

Figure 3: API flow chart diagram



2.4.4 Demo code

The following is an example of the demonstration code that reads data from the gyroscope sensor in mdps `(gyr_raw[3])` and from the accelerometer sensor in mg

`(acc_raw[3])` and calculates compensated data in mdps `(gyr_cal_x, gyr_cal_y, gyro_cal_z)`.

```
[…]

#define VERSION_STR_LENG        35
#define SAMPLE_FREQUENCY     50.0f

[…]

/*** Initialization ***/
char strVersion[VERSION_STR_LENG];
osx_lm_result_t lic_res;
osx_MGC_knobs_t knobs;
int gyro init offset[3];
float sample freq;

/* Gyroscope calibration API initialization function */
lic_res = osx_MotionGC_Initialize(SAMPLE_FREQUENCY);

/* Optional: Get version */
osx MotionGC GetLibVersion(strVersion);

/* Optional: Get knobs settings */
osx_MotionGC_GetKnobs(&knobs);

/* Optional: Adjust knobs settings */
knobs.acc_trh_mg = 8;
knobs.gyro_thr_mdps = 150;
osxMotionGC_Set_knobs(&knobs);

/* Optional: Set initial gyroscope offset */
gyro init offset[0] = 0;
gyro_init_offset[1] = 0;
gyro_init_offset[2] = 0;
osxMotionGC_SetCalParams(gyro_init_offset);

/* Optional: Set sample frequency */
sample freq = SAMPLE FREQUENCY;
osxMotionGC_SetFrequency(sample_freq);

[…]

/*** Using gyroscope calibration algorithm ***/
{
Timer_OR_DataRate_Interrupt_Handler()
  int acc_raw[3];
  int gyr raw[3];
  int cal raw x, cal raw y, cal raw z;

  int gyro_bias[3];        /* Offset */
  int bias_update;         /* Update flag */
  /* Get acceleration X/Y/Z in mG */
  MEMS Read AccValue(acc raw[0], acc raw[1], acc raw[2]);

  /* Get angular rate X/Y/Z in mdps */
  MEMS_Read_GyroValue(gyr_raw[0], gyr_raw[1], gyr_raw[2]);

  /* Gyroscope calibration algorithm update */
  osx MotionGC Update(acc raw, gyro raw, gyro bias, &bias update);

  /* Apply correction */
  gyr_cal_x = (int)((gyr_raw[0] - gyro_bias[0]);
  gyr_cal_y = (int)((gyr_raw[1] - gyro_bias[1]);
  gyr cal z = (int)((gyr raw[2] - gyro bias[2]);
}
```

2.4.5 The calibration process

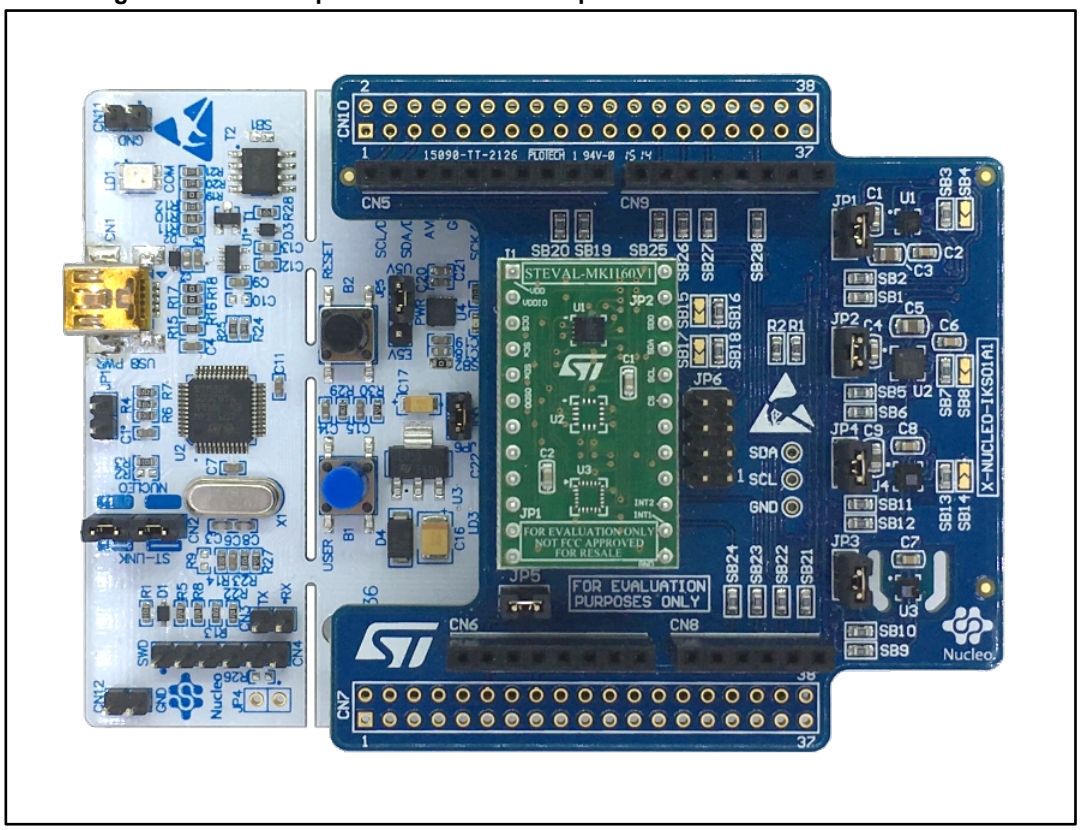
The calibration process does not need any sensor motion as the offset coefficient is calculated and updated when the sensor is still.

2.5 Sample application

The osxMotionGC middleware can be easily manipulated to build user applications; a sample application is provided in the Projects folder.

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

Figure 4: Sensor expansion board and adapter connected to an STM32 Nucleo



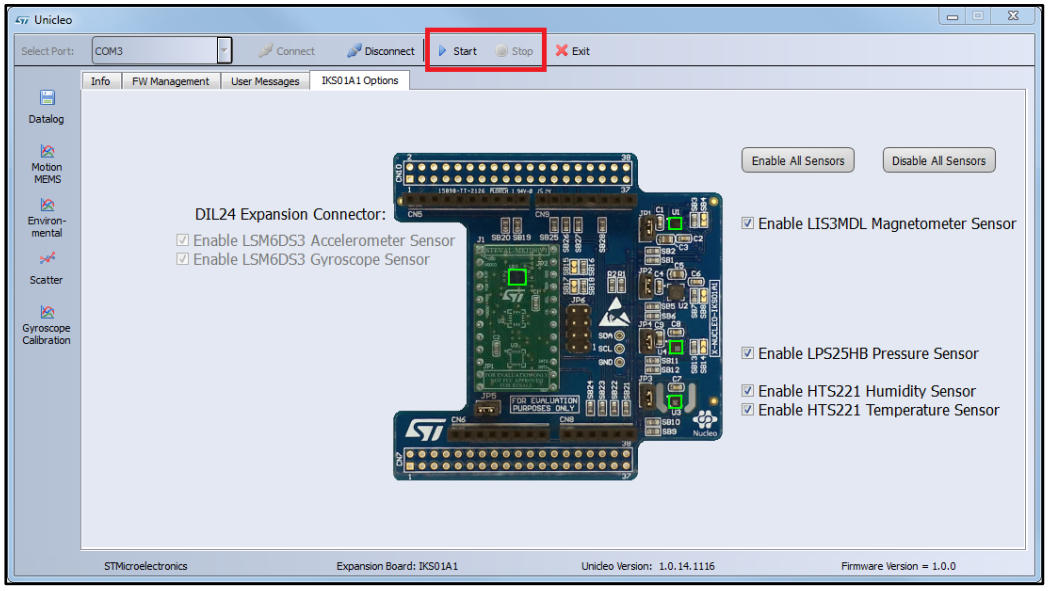
The accelerometer algorithm output data may be displayed in real-time through a specific GUI.

2.5.1 Unicleo-GUI application

The osxMotionGC software package for STM32Cube uses the Windows Unicleo-GUI utility, which can be downloaded from *www.st.com*.

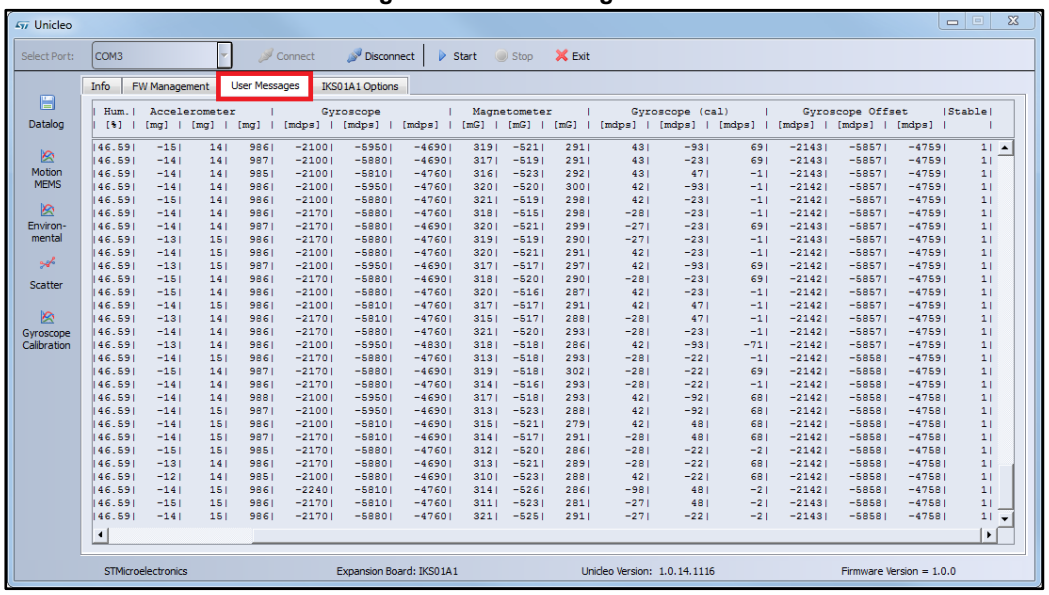
1. Ensure that the necessary drivers are installed and the STM32 Nucleo board with the appropriate expansion board is connected to the PC.
2. Launch the Unicleo-GUI application to open the main application window. If an STM32 Nucleo board with the supported firmware is connected to the PC, it is automatically detected and the appropriate COM port is opened.

Figure 5: Unicleo main window



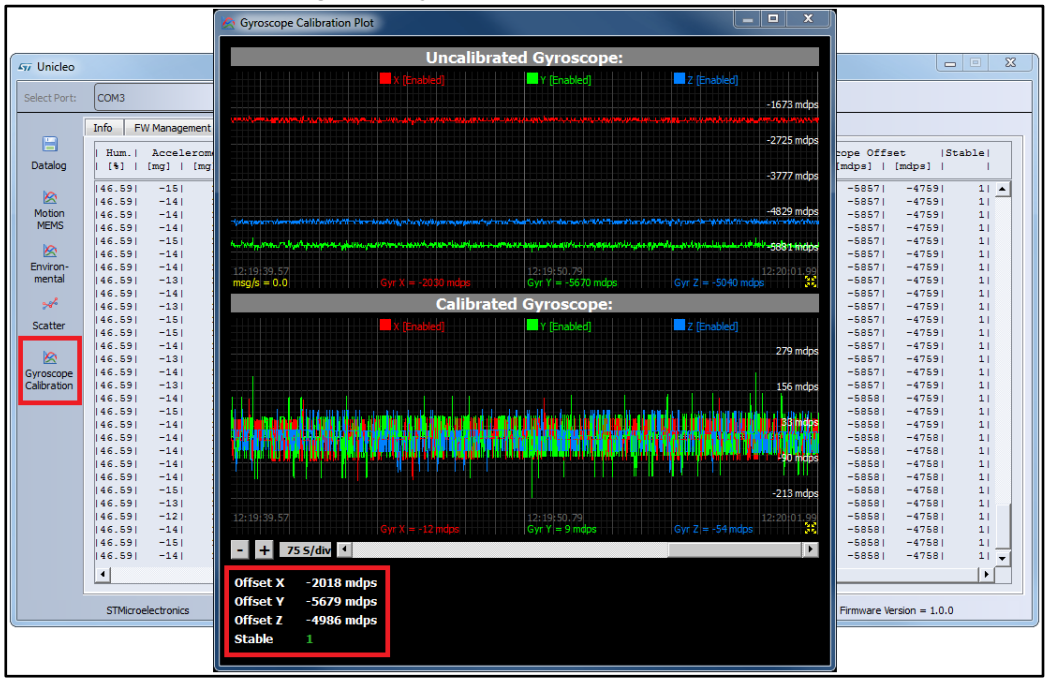
3 Start and stop data streaming by using the appropriate buttons on the vertical toolbar. The data coming from the connected sensor can be viewed in the User Messages tab.

Figure 6: User Messages tab



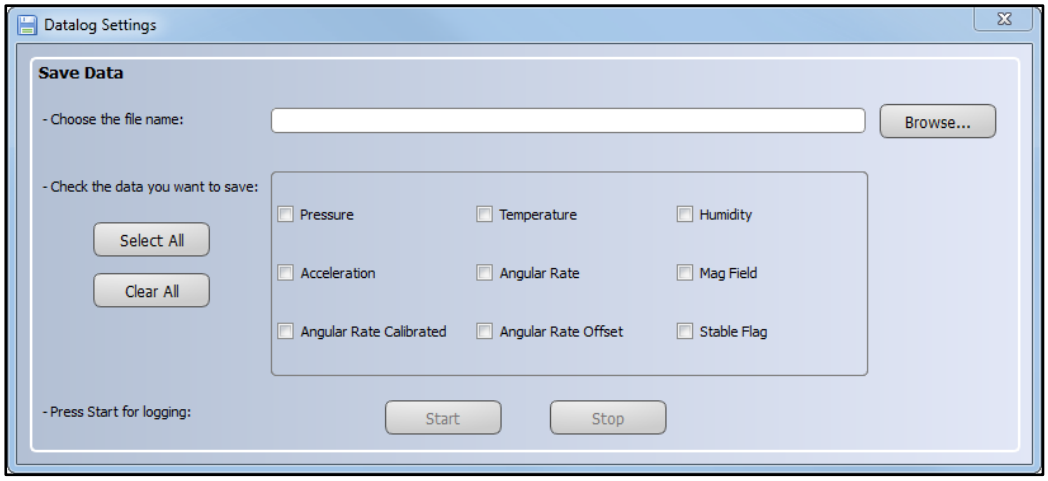
4 Click on the Gyroscope Calibration icon in the vertical toolbar to open the dedicated application window. The window is split in three sections: the first one containing uncalibrated data, the second one containing calibrated data and the last one containing offset and stable flag information.

Figure 7: Gyroscope calibration window



5 Click on the Datalog icon in the vertical toolbar to open the datalog configuration window: you can select the sensor and activity data to be saved in the files. You can start or stop saving by clicking on the corresponding button.

Figure 8: Datalog window



3 References

1. UM1859: Getting started with the X-CUBE-MEMS1 motion MEMS and environmental sensor software expansion for STM32Cube
2. DB3168: Real-time gyroscope calibration software expansion for STM32Cube
3. UM2012: osxMotionXX system setup
4. UM1724: STM32 Nucleo-64 board
5. UM2128: Getting started with Unicleo-GUI for motion MEMS and environmental sensor software expansion for STM32Cube

4 Revision history

Table 2: Document revision history

Date	Version	Changes
16-Jan-2017	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.