UM2724

User manual

Getting started with MotionDI dynamic inclinometer library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionDI middleware library is part of the X-CUBE-MEMS1 software and runs on STM32. It provides real-time information about device orientation, including tilt information, and can also perform accelerometer and gyroscope calibration.

This library is intended to work with ST MEMS only.

The algorithm is provided in static library format and is designed to be used on STM32 microcontrollers based on the ARM® Cortex®-M3, ARM® Cortex®-M4 or ARM® Cortex®-M7 architecture.

It is built on top of STM32Cube software technology to ease portability across different STM32 microcontrollers.

The software comes with a sample implementation running on X-NUCLEO-IKS01A2 or X-NUCLEO-IKS01A3 expansion board on a NUCLEO-F401RE, NUCLEO-L476RG or NUCLEO-L152RE development board.

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 environment

2 MotionDI middleware library in X-CUBE-MEMS1 software expansion for STM32Cube

2.1 MotionDI overview

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

The library acquires data from the accelerometer and gyroscope and provides information about the device position (quaternions, Euler angles, linear acceleration, gravity vector).

The MotionDI filtering and predictive software uses advanced algorithms to intelligently integrate outputs from multiple MEMS sensors for optimum performance regardless of environmental conditions.

The library is designed for ST MEMS only. Functionality and performance when using other MEMS sensors have not been tested and can vary significantly from documented behavior.

A sample implementation is available for X-NUCLEO-IKS01A2 and X-NUCLEO-IKS01A3 expansion boards, mounted on a NUCLEO-F401RE, NUCLEO-L476RG or NUCLEO-L152RE development board.

2.2 MotionDI library

Technical information fully describing the functions and parameters of the MotionDI APIs can be found in the MotionDI_Package.chm compiled HTML file located in the Documentation folder.

2.2.1 MotionDI library description

The MotionDI dynamic inclinometer library manages the data acquired from the accelerometer and gyroscope sensor; it features:

- real-time 6-axis motion sensor data fusion (accelerometer, gyroscope)
- computation of rotation vector, quaternions, gravity and linear acceleration data
- gyroscope bias calibration
- accelerometer bias and scale calibration
- recommended sensor data sampling frequency of 100 Hz

Note: Real size might differ for different IDEs (toolchain)

- resource requirements:
  - Cortex-M3: 56.1 kB of code and 6.2 kB of data memory
  - Cortex-M4: 48.7 kB of code and 6.2 kB of data memory
  - Cortex-M7: 46.5 kB of code and 6.2 kB of data memory
- available for ARM® Cortex®-M3, ARM® Cortex®-M4 and ARM® Cortex®-M7 architectures

2.2.2 MotionDI library operation

The MotionDI library implements a sensor fusion algorithm for the estimation of 3D orientation in space. It uses a digital filter based on the Kalman filter theory to merge data from several sensors and compensate for the limitations of the single sensors. For instance, as gyroscope data drift can impact the orientation estimates, the accelerometer can be used to provide absolute tilt orientation information.

The MotionDI library integrates sensor fusion and calibration algorithms in one library; the calibration functionality can be enabled or disabled using the knob setting.

2.2.3 MotionDI library parameters

The library is based on parameters pertaining to an `MDI_knobs_t` structure.

The parameters MDI_fusion_knobs_t for the structure are:

- ATime, FrTime: represent the weighting stability of sensors for prediction (trust factor), from 0.5 to 5. Default values are recommended.
  - ATime: lowering the value increases the accelerometer weight and sensitivity towards external acceleration.
  - FrTime: lowering the value increases the trust on gyroscope data and lower the correction from the accelerometer.
- modx: represents the decimation of MotionDI_update call frequency
- output_type: represents the sensor fusion library output orientation: 0 = NED, 1 = ENU

The parameters MDI_acc_cal_knobs_t for the structure are:

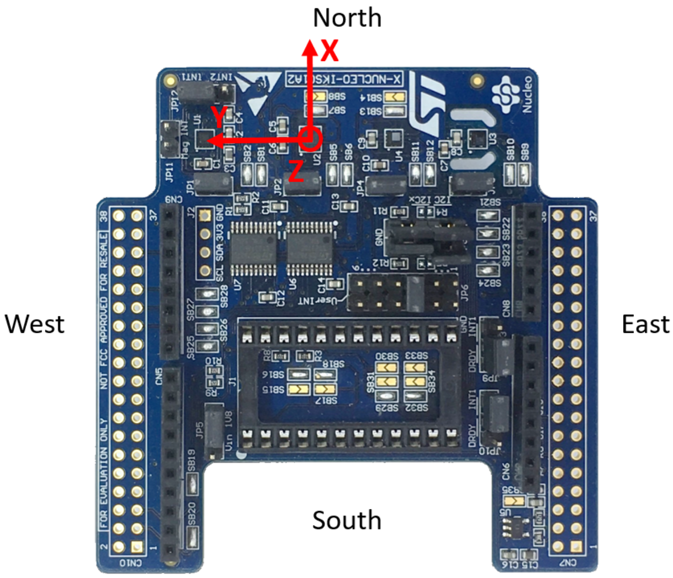
- MoveThresh_g: the maximum motion allowed during the calibration. Recommended in the range of 0.1-0.30 g. The expected accuracy of bias is the same as MoveThresh_g parameters
- CalType: the library can be configured to run the calibration and the corresponding frequency.
  - MDI_CAL_NONE: disables the calibration
  - MDI_CAL_ONETIME: runs the calibration one time and then disables it when it reaches a good calibration quality
  - MDI_CAL_CONTINUOUS: continuously runs the calibration.

The parameters MDI_gyro_cal_knobs_t for the structure are:

- AccThr: is the accelerometer threshold to detect steady state in g. The default value is 0.001 g. The input range is 0.0005 to 0.01 g. For higher accuracy, set the value low; for platforms inherent vibration or noisy sensors, set the value high.
- GyroThr: is the gyroscope threshold to detect steady state in degrees per second. The default value is 0.25 dps. The input range is 0.008 to 0.4 dps. For higher accuracy, set the value low.
- MaxGyro: is the maximum expected angular rate offset when still in [dps]. The default value is 15 dps.
- GBiasDiffAVTh: is the gyroscope threshold in dps unit. The default value of this variable is 0.33, the range of the input is 0.2 to 0.5. For higher accuracy, set the value low.
- GBiasDiffVelTh: is the accelerometer threshold in g unit. The default value of this variable is 0.033, the range of the input is 0.02 to 0.05. For higher accuracy, set the value low. Both variables (GBiasDiffAVTh and GBiasDiffVelTh) represent the minimum change in acceleration and angular velocity required to detect stationary condition.
- CalType: calibration type

As shown in the figure below, the X-NUCLEO-IKS01A2 accelerometer sensor has an NWU (x-North, y-West, z-Up), so the string is: “nwu”.

Figure 1. **Example of sensor orientations**



2.2.4 MotionDI library output data rate

Set up the dynamic inclinometer library output data rate (ODR) properly. Although the ODR can be set to a higher value, 100 Hz is recommended.

2.2.5 Sensor calibration in the MotionDI library

Accelerometer calibration

Accelerometer calibration can improve the accuracy of tilt angle and is recommended for applications which demand a very accurate orientation. It aligns the system in six positions according to the gravity direction.

The accelerometer calibration can be configured to run one time since the calibration parameter drift is very low. Calibration can be done by placing the device in 6 different directions with respect to gravity. For example, device can be placed with ±X, ±Y, ±Z.

Gyroscope calibration

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

2.2.6 MotionDI APIs

The MotionDI APIs are:

- `uint8_t MotionDI_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
- `void MotionDI_Initialize(float *freq)`
  - performs MotionDI library initialization and setup of the internal mechanism
  - `freq` is frequency at which the IMU data is available to library
  - the CRC module in STM32 microcontroller (in RCC peripheral clock enable register) has to be enabled before using the library

Note: This function must be called before using the dynamic inclinometer library.

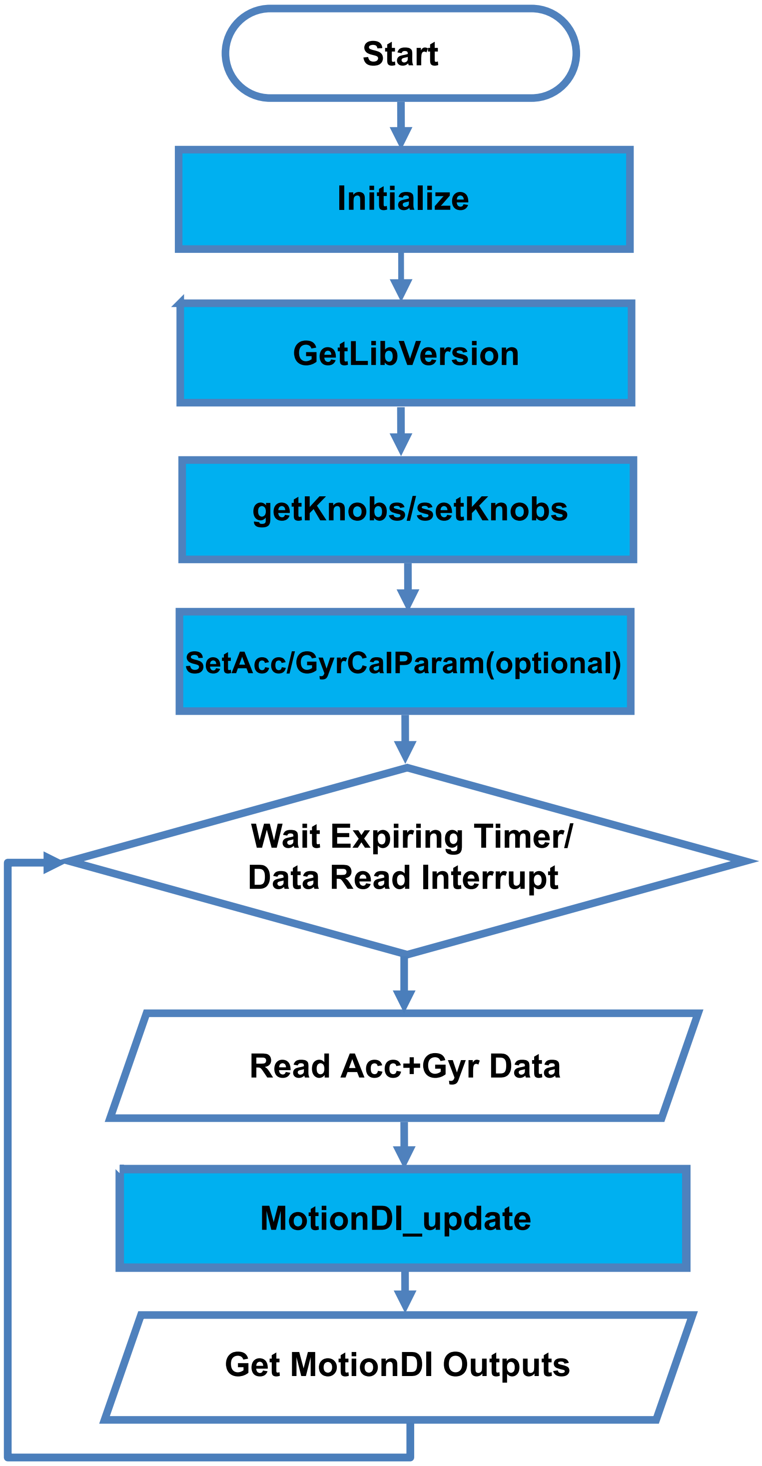
- `void MotionDI_setKnobs(MDI_knobs_t *knobs)`
  – sets the internal knobs
  – `knobs` is a pointer to a structure with knobs
- `void MotionDI_getKnobs(MDI_knobs_t *knobs)`
  – gets the current internal knobs
  – `knobs` is a pointer to a structure with knobs
- `void MotionDI_AccCal_getParams(MDI_cal_output_t *acc_cal)`
  – gets the accelerometer calibration parameters
  – `acc_cal` is a pointer to a structure with accelerometer calibration parameters. The accelerometer calibration parameters contain bias, scale factor and calibration quality.
- `void MotionDI_AccCal_setParams(MDI_cal_output_t *acc_cal)`
  – sets the accelerometer calibration parameters
  – `acc_cal` is a pointer to a structure with accelerometer calibration parameters. The accelerometer calibration parameters contain bias, scale factor and calibration quality.
    This function should be called after MotionDI_Initialize but before calling MotionDI_update function.
- `void MotionDI_AccCal_reset(void)`
  – restarts accelerometer calibration algorithm
- `void MotionDI_GyrCal_getParams(MDI_cal_output_t*gyro_cal)`
  – gets the gyroscope calibration parameters
  – `gyro_cal` is a pointer to a structure with gyroscope calibration parameters. The gyroscope calibration parameters contain bias, and calibration quality. The scale factor remains 1.
- `void MotionDI_GyrCal_setParams(MDI_cal_output_t *gyro_cal)`
  – sets the gyroscope calibration parameters
  – `gyro_cal` is a pointer to a structure with gyroscope calibration parameters. The gyroscope calibration parameters contain bias, and calibration quality. The scale factor remains 1.
    This function should be called after MotionDI_Initialize but before calling MotionDI_update function.
- `void MotionDI_GyrCal_reset(void)`
  – restarts gyroscope calibration algorithm
- `void MotionDI_update(MDI_output_t *data_out, MDI_input_t *data_in)`
  – runs the update
  – `data_out` is a pointer to output data structure
  – `data_in` is a pointer to input data structure

2.2.7 Portability to a generic microcontroller

The MotionDI library does not have any dependency on external libraries. It requires an FPU enabled on the MCU.

2.2.8 API flow chart

Figure 2. MotionDI API logic sequence



2.2.9 Demo code

The following demonstration code reads data from the accelerometer and gyroscope sensors and gets the rotation, quaternions, gravity and linear acceleration.

```
[…]

#define VERSION_STR_LENG 35

[…]

/*** Initialization ***/
char lib_version[VERSION_STR_LENG];
float freq = 100.0f;
MDI_knobs_t iKnobs;

/* Dynamic Inclinometer API initialization function */
MotionDI_Initialize(&freq);

/* Optional: Get version */
MotionDI_GetLibVersion(lib_version);

/* Optional: Modify knobs settings & set the knobs */
MotionDI_getKnobs(&iKnobs);

[…]

MotionDI_setKnobs(&iKnobs);

[…]

/*** Using Dynamic Inclinometer algorithm ***/
Timer_OR_DataRate_Interrupt_Handler()
{
    MDI_input_t data_in;
    MDI_output_t data_out;

    /* Get acceleration X/Y/Z in g */
    MEMS_Read_AccValue(data_in.Acc[0], data_in.Acc[1], data_in.Acc[2]);

    /* Get angular rate X/Y/Z in dps */
    MEMS_Read_GyroValue(data_in.Gyro[0], data_in.Gyro[1], data_in.Gyro[2]);

    data_in.Timestamp = CurrentTime;

    /* Run Dynamic Inclinometer algorithm */
    MotionDI_update(&data_out, &data_in);

    /* Use data_out – output data from Dynamic Inclinometer algorithm */

    […]
}
```

2.2.10 Algorithm performance

The dynamic inclinometer algorithm uses data from the accelerometer and gyroscope sensors and runs at 100 Hz frequency.

Table 2. Cortex-M3: elapsed time (µs) algorithm

Cortex-M3 STM32L152RE at 32 MHz								
STM32CubeIDE 1.3.1			IAR EWARM 8.32.3			Keil µVision 5.29		
Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
1512	6027	38085	1080	3618	24907	1942	4849	45138

Table 3. **Cortex-M4: elapsed time (µs) algorithm**

Cortex-M4 STM32F401RE at 84 MHz								
STM32CubeIDE 1.3.1			IAR EWARM 8.32.3			Keil µVision 5.29		
Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
1391	1449	2884	1079	1111	2268	1538	1619	4450

Table 4. **Cortex-M7: elapsed time (µs) algorithm**

Cortex-M7 STM32F767ZI at 96 MHz								
STM32CubeIDE 1.3.1			IAR EWARM 8.32.3			Keil µVision 5.29		
Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
1794	1820	2976	1600	1620	2531	1254	1277	2069

2.3 Sample application

The MotionDI middleware can be easily manipulated to build user applications. A sample application is provided in the Application folder.

It is designed to run on a NUCLEO-F401RE, NUCLEO-L476RG or NUCLEO-L152RE development board connected to an X-NUCLEO-IKS01A2 or X-NUCLEO-IKS01A3 expansion board.

The application provides motion sensor data fusion (quaternions, Euler angles, linear acceleration, gravity vector) and tilt information in real-time or via received offline data. It also performs accelerometer and gyroscope calibration. Data can be displayed through a GUI.

USB cable connection is required to monitor real-time data or feed library with offline data. The board is powered by the PC via USB connection.

This working mode allows the user to display motion sensor fusion data and tilt information, accelerometer, and gyroscope data, timestamp and eventually other sensor data, in real time, using the Unicleo-GUI.

2.4 Unicleo-GUI application

The sample application uses the Windows Unicleo-GUI utility, which can be downloaded from www.st.com.

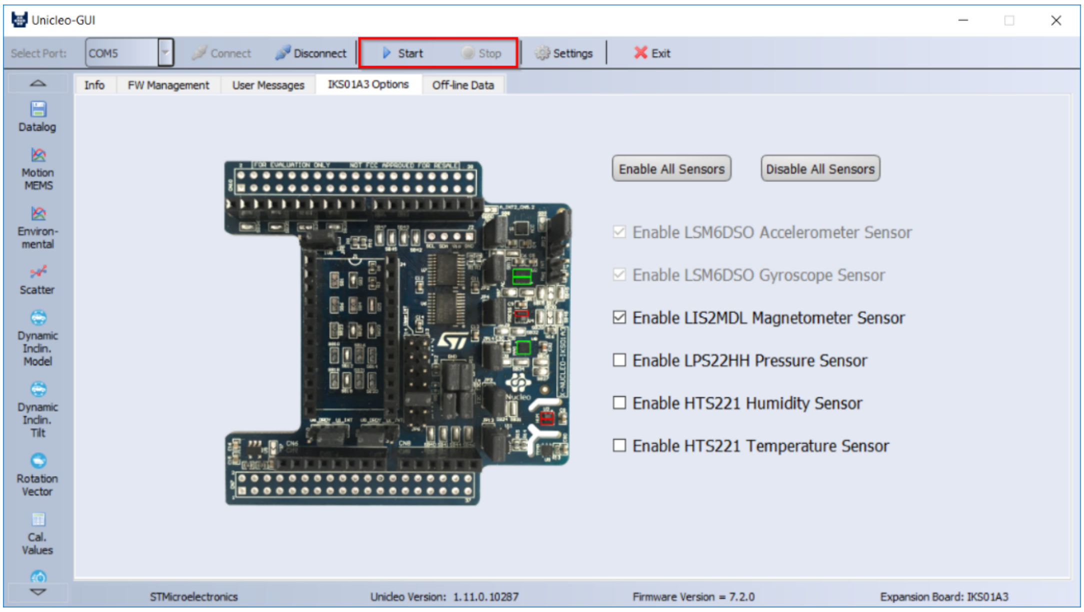
Step 1. Ensure that the necessary drivers are installed and the STM32 Nucleo board with appropriate expansion board is connected to the PC.

Step 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 is automatically detected and the appropriate COM port is opened.

Note: It is necessary to scroll the vertical tool bar using the mouse wheel or clicking on up/down arrows to see all icons.

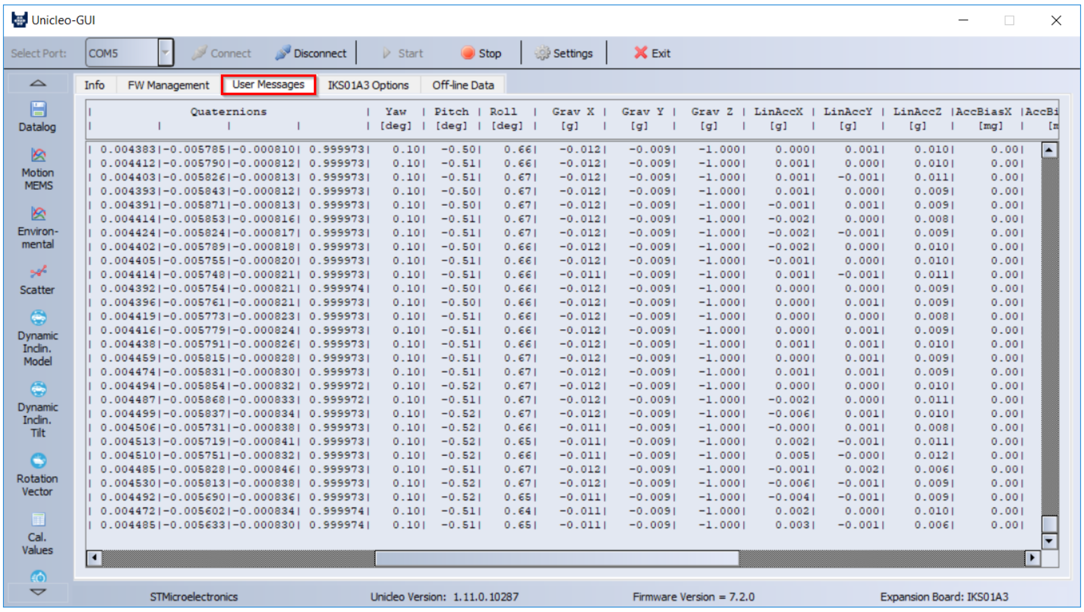
Figure 3. Unicleo main window



Step 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.

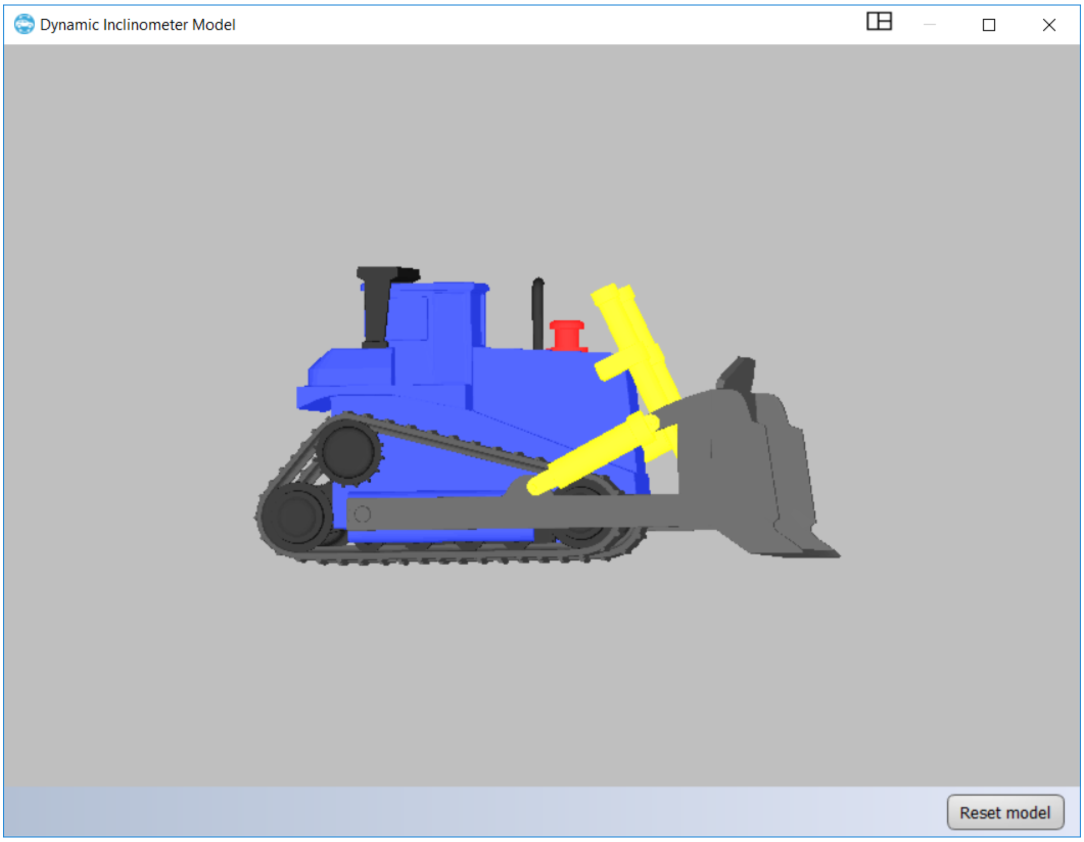
Figure 4. User Messages tab



Step 4. Click on the Dynamic Inclin. Model icon in the vertical toolbar to open the application window displaying the vehicle model.

To align the vehicle model, point the STM32 Nucleo board towards the screen and press the Reset model button.

Figure 5. Dynamic Inclinometer Model window



Step 5. Click on the Dynamic Inclin. Tilt icon in the vertical toolbar to open the application window displaying tilt.

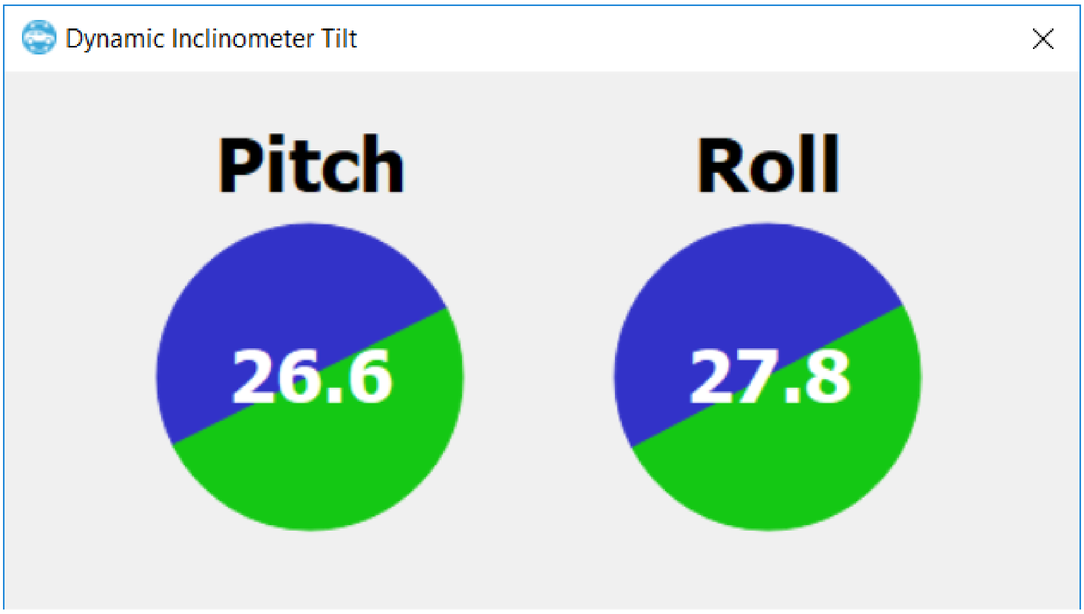


Figure 6. **Dynamic Inclinometer Tilt window**

Step 6. Click on the Rotation Vector icon in the vertical toolbar to open the application window displaying the rotation vector graph.

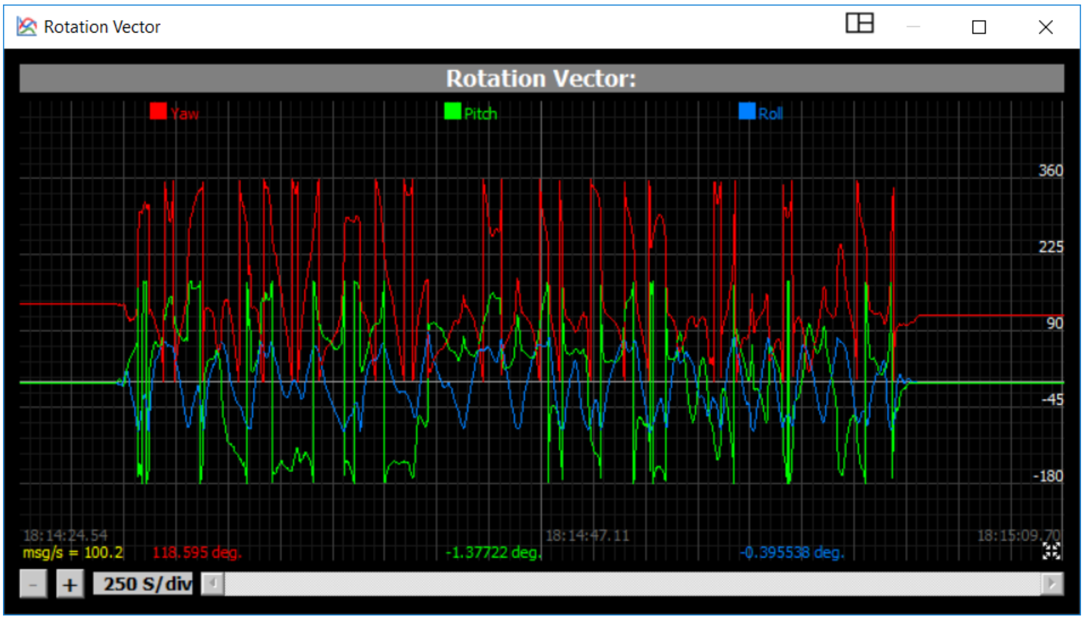


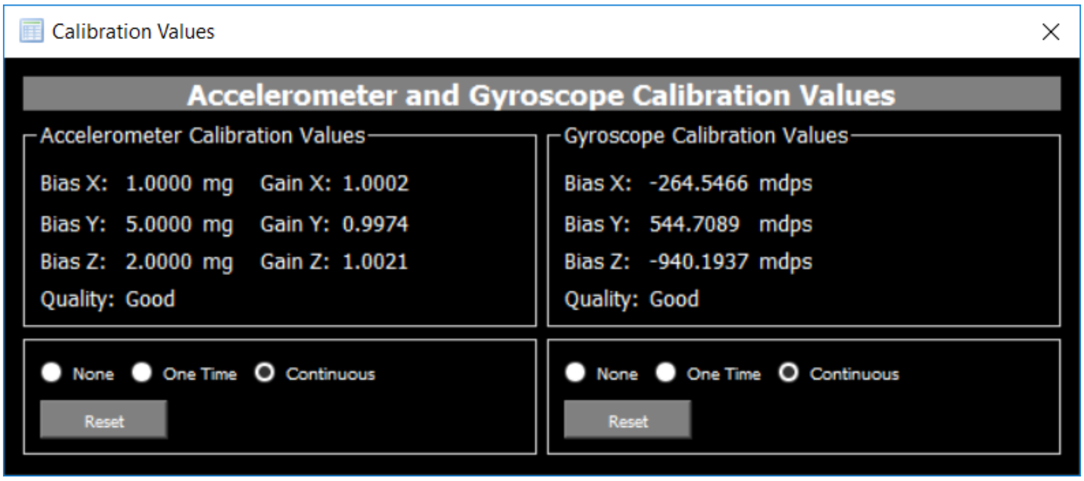
Figure 7. **Rotation Vector window**

Step 7. Click on the Cal. Values icon in the vertical toolbar to open the application window displaying accelerometer and gyroscope calibration status.

To switch between calibration modes click on the appropriate button.

To reset calibration press the Reset button.

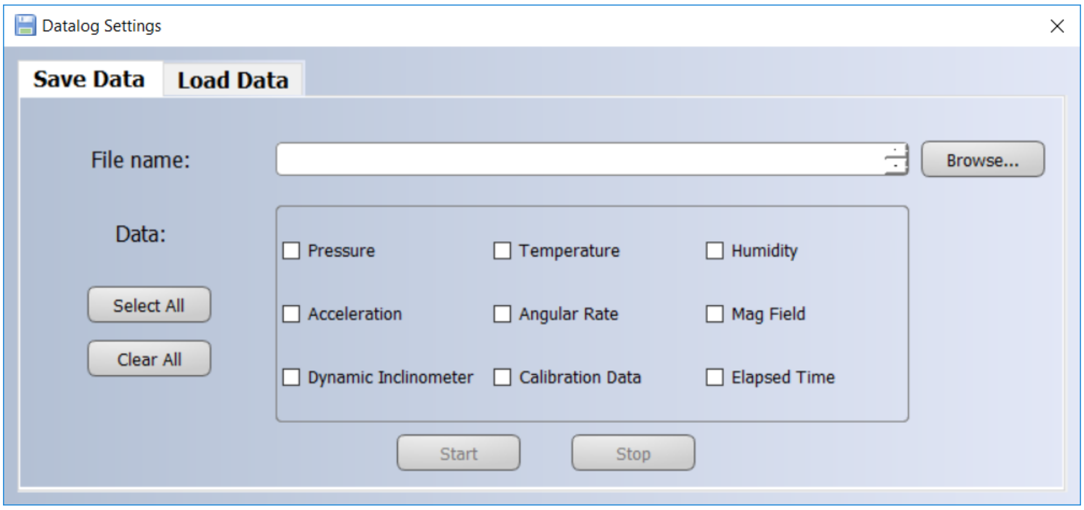
Figure 8. Calibration Values window



Step 8. Click on the Datalog icon in the vertical tool bar to open the datalog configuration window:

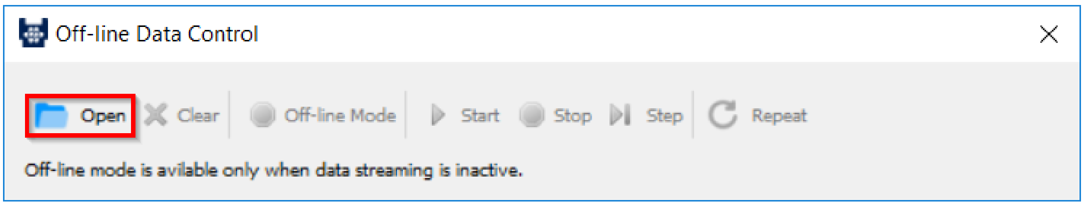
you can select the various data to be saved in files. You can start or stop saving by clicking on the corresponding button.

Figure 9. Datalog window



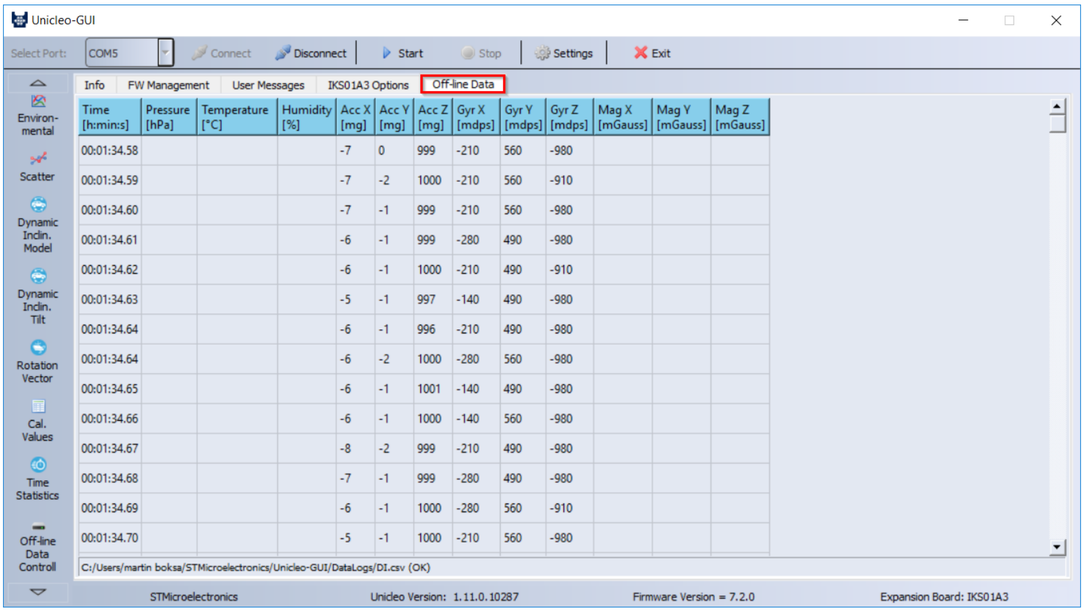
Step 9. To process previously captured data, click on the Off-line Data Control icon in the vertical toolbar to open the dedicated window. The data are processed by the MCU firmware.

Figure 10. Off-line Data Control window



Step 10. Click on the Open button to select the file with offline data in CSV format.
The data are loaded into the Offline Data tab.

Figure 11. Off-line Data tab

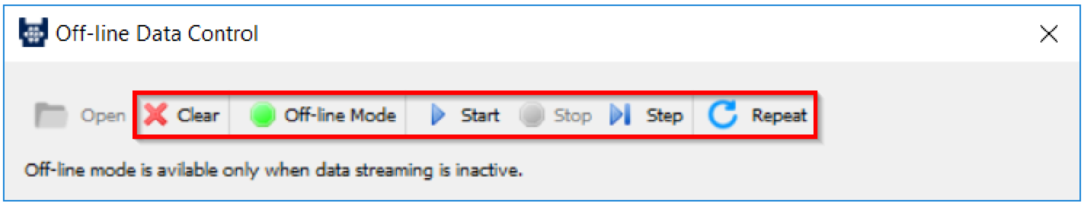


Other buttons in the Off-line Data Control window become active. You can click on:

– Off-line Mode button to switch the offline mode of the firmware on/off .
– Start/Stop/Step/Repeat buttons to control the data feed from Unicleo-GUI to the firmware.
– Clear button to remove the data from Unicleo-GUI.

Note: You have to stop data streaming from real sensors by using the Stop button on the top vertical tool bar to switch to offline mode.

Figure 12. Off-line Data Control window - offline mode



3 References

All of the following resources are freely available on www.st.com.

1. UM1859: Getting started with the X-CUBE-MEMS1 motion MEMS and environmental sensor software expansion for STM32Cube
2. UM1724: STM32 Nucleo-64 boards (MB1136)
3. UM2128: Getting started with Unicleo-GUI for motion MEMS and environmental sensor software expansion for STM32Cube

Revision history

Table 5. Document revision history

Date	Version	Changes
12-May-2020	1	Initial release.

Contents

List of tables

List of figures