UM2192

User manual

Getting started with MotionMC magnetometer calibration library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionMC is a middleware library part of X-CUBE-MEMS1 software and runs on STM32. It provides real-time magnetometer calibration using hard iron (HI) and scale factor (SF) coefficients to correct magnetometer data.

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®-M0+, ARM® Cortex®-M3, ARM® Cortex®-M33, ARM® Cortex®-M4 or ARM® Cortex®-M7 architecture.

It is built on top of STM32Cube software technology that eases portability across different STM32 microcontrollers.

The software comes with sample implementation running on an X-NUCLEO-IKS4A1, X-NUCLEO-IKS01A3 or X-NUCLEO-IKS02A1 expansion board on a NUCLEO-F401RE, NUCLEO-U575ZI-Q, NUCLEO-L152RE or NUCLEO-L073RZ 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 MotionMC middleware library in X-CUBE-MEMS1 software expansion for STM32Cube

2.1 MotionMC overview

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

Objects near the surroundings of a magnetometer can cause distortion of Earth magnetic field. Based on the distortion character, we can categorize it as two types: hard iron and soft iron effect.

Hard iron distortion arises from permanent magnets, magnetized iron or steel located nearby the magnetometer. This distortion remains constant and in fixed location related to the magnetometer for all heading orientations. Hard iron effect adds a constant magnitude field component along each magnetometer axis.

Soft iron distortion arises from the interaction of the Earth magnetic field and the material which surrounds the magnetometer sensor distorting the Earth magnetic field. The distortion magnitude and direction depend on the incident angle of the Earth magnetic field on the material. Hence, it varies with the magnetometer orientation.

The library acquires data from the magnetometer and calculates the hard iron (HI) and soft iron (SI) coefficients or, in the case of the library for Cortex-M3, M33, M4 and M7 only the scale factor (SF - diagonal elements of soft iron matrix) coefficients together with the calibration quality value. The calibration coefficients are then used to compensate raw data from the magnetometer and reduce hard iron and soft iron effects.

The library is designed for ST MEMS only. Functionality and performance when using other MEMS sensors are not analyzed and can be significantly different from what described in the document.

Sample implementation is available on X-NUCLEO-IKS4A1, X-NUCLEO-IKS01A3 and X-NUCLEO-IKS02A1 expansion boards, mounted on a NUCLEO-F401RE, NUCLEO-U575ZI-Q, NUCLEO-L152RE or NUCLEO-L073RZ development board.

2.2 MotionMC library

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

2.2.1 MotionMC library description

The MotionMC magnetometer calibration library manages data acquired from a magnetometer; it features:

- update frequency up to 100 Hz
- hard iron compensation, theoretically unlimited (within sensor range)
- wide scale factor compensation range from 0.65 to 1.35 in every direction
- support for all ST magnetometer devices
- resources requirements:
  - Cortex-M0+: 9.6 kB of code and 0.1 kB of data memory
  - Cortex-M3: 45.8 kB of code and 2.4 kB of data memory
  - Cortex-M33: 34.9 kB of code and 2.4 kB of data memory
  - Cortex-M4: 35.0 kB of code and 2.4 kB of data memory
  - Cortex-M7: 34.6 kB of code and 2.4 kB of data memory
- available for ARM® Cortex®-M0+, ARM® Cortex®-M3, ARM® Cortex®-M33, ARM® Cortex®-M4 and ARM® Cortex®-M7 architectures

2.2.2 MotionMC APIs

The MotionMC APIs are:

- **Cortex-M3 and Cortex-M4**
  - `uint8_t MotionMC_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 MotionMC_Initialize(int sampletime, unsigned short int enable)`
    - performs MotionMC library initialization and setup of the internal mechanism.
    - `sampletime` parameter is the interval between update function calls in ms.
    - `enable` parameter enables (1) or disables (0) the library

Note: This function must be called before using the magnetometer calibration library and the CRC module in the STM32 microcontroller (in RCC peripheral clock enable register) has to be enabled.

  - `void MotionMC_Update (MMC_Input_t *data_in)`
    - runs the magnetometer calibration algorithm
    - `*data_in` parameter is a pointer to a structure with input data
    - the parameters for the structure type `MMC_Input_t` are:
      - `Mag[3]` is an array of magnetometer sensor value in µT
      - `TimeStamp` parameter is the timestamp value in ms for current magnetometer sensor data

Note: This function has to be called periodically at the same period that is indicated in the initialization function.

  - `void MotionMC_GetCalParams (MMC_Output_t *data_out)`
    - retrieves the magnetometer hard iron (HI) and scale factor (SF) coefficients
    - `*data_out` parameter is a pointer to a structure with ouput data
    - the parameters for the structure type `MMC_Output_t` are:
      -
        - `HI_Bias[3]` is an array of magnetometer hard iron (HI) coefficients in µT
        - `SF_Matrix[3][3]` is a 3x3 matrix of magnetometer scale factor (SF) coefficients
        - `CalQuality` is the calibration quality factor.

- **Storing and loading magnetometer calibration parameters**
  The following functions have to be implemented specifically for each target platform in the user's application due to hardware dependent load/store mechanisms. The functions are automatically called by the library. The `MotionMC_LoadMagCalFromNVM` function is called when the magnetometer calibration is enabled, by calling `MotionMC_Initialize` with parameter `enable` set to 1. The `MotionMC_SaveMagCalInNVM` function is called when the magnetometer calibration is disabled, by calling `MotionMC_Initialize` with parameter `enable` set to 0. If functions for storing and loading the magnetometer calibration coefficients are not needed the function should only return 1.
  - `char MotionMC_LoadMagCalFromNVM(unsigned short int datasize, unsigned int *data)`
    - the function is used to retrieve the calibration parameters from storage, the function is called when magnetometer calibration library is enabled
    - `datasize` is the size of the data in bytes
    - `*data` is the data location pointer
    - returns 0 for correct loading, 1 otherwise
  - `char MotionMC_SaveMagCalInNVM(unsigned short int datasize, unsigned int *data)`
    - the function is used to store the calibration parameters and is called when the magnetometer calibration library is disabled
    - `datasize` is the size of the data in bytes
    - `*data` is the data location pointer
    - returns 0 for correct saving, 1 otherwise

- **Cortex-M0+:**
  - `uint8_t MotionMC_CM0P_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 *MotionMC_CM0P_Initialize(MMC_CM0P_mcu_type_t mcu_type void *mmc_algo, MMC_CM0P_Mode_t mode, unsigned short int enable)`
    - performs MotionMC library initialization and setup of the internal mechanism.
    - `mcu_type` is the type of MCU:
      - `MMC_CM0P_MCU_STM32` is a standard STM32 MCU
      - `MMC_CM0P_MCU_BLUE_NRG1` is BlueNRG-1
      - `MMC_CM0P_MCU_BLUE_NRG2` is BlueNRG-2
      - `MMC_CM0P_MCU_BLUE_NRG_LP` is BlueNRG-LP
      - `MMC_CMOP_MCU_STM32WB0` is STM32WB0
    - `*mmc_algo` is a pointer to the instance of MotionMC algorithm
    - `mode` parameter defines calibration mode type
      - `MMC_CM0P_HI_ONLY` hard-iron only mode (faster)
      - `MMC_CM0P_HI_AND_SI` hard-iron + soft-iron (slower)
    - `enable` parameter enables (1) or disables (0) the library

Note: This function must be called before using the magnetometer calibration library and the CRC module in the STM32 microcontroller (in RCC peripheral clock enable register) has to be enabled.

  - `void MotionMC_CM0P_Update(void *mmc_algo, MMC_CM0P_Input_t *data_in)`
    - runs the magnetometer calibration algorithm
    - `*mmc_algo` is a pointer to the instance of MotionMC algorithm
    - `*data_in` parameter is a pointer to a structure with input data
    - the parameters for the structure type `MMC_CM0P_Input_t` are:
      - `DTime` parameter is the interval between update function calls in seconds
      - `Mag[3]` is an array of magnetometer sensor value in µT

Note: This function has to be called periodically at the same period that is indicated in the `DTime` parameter.

  - `void MotionMC_CM0P_GetCalParams (MMC_CM0P_Output_t *data_out)`
    - retrieves the magnetometer hard iron (HI) and soft iron (SI) coefficients
    - `*data_out` parameter is a pointer to a structure with ouput data
    - the parameters for the structure type `MMC_CM0P_Output_t` are:
      - `HI_Bias[3]` is an array of magnetometer hard iron (HI) coefficients in µT
      - `SI_Matrix[3][3]` is a 3x3 matrix of magnetometer soft iron (SI) coefficients
      - `B` is the estimated magnetic field in µT (to be used in `MMC_CM0P_HI_ONLY` mode)
      - `CalQuality` is the calibration quality factor.

2.2.3 API flow chart

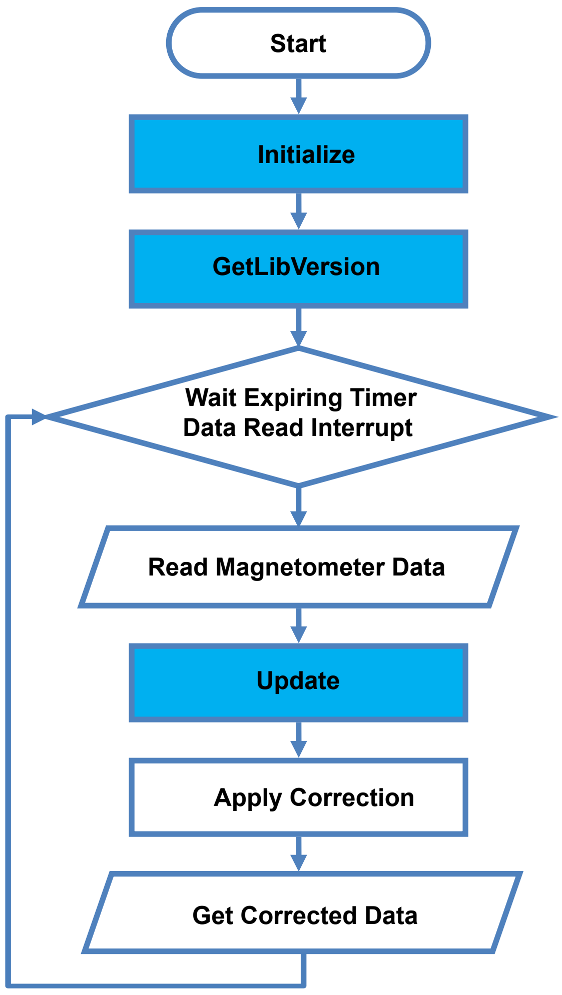


Figure 1. MotionMC API logic sequence

2.2.4 Demo code

The following demonstration code reads data from magnetometer sensor and calculates compensated data.

```
[…]

#define VERSION_STR_LENG 35
#define SAMPLE_TIME 20 /* Set sample timer overflow to 20 ms (i.e.: 50 Hz) */

[…]

/* Initialization */
volatile uint32_t timestamp = 0; /* Increments when sample timeroverflows */

char lib_version[VERSION_STR_LENG];

/* Magnetometer calibration API initialization function */
MotionMC_Initialize(SAMPLE_TIME, 1);

/* Optional: Get version */
```

```
MotionMC_GetLibVersion(lib_version);

[…]

/* Using magnetometer calibration algorithm */
Timer_overflow_Interrupt_Handler()
{
  MMC_input_t data_in;
  MMC_output_t data_out;
  float mag_cal_x, mag_cal_y, mag_cal_z;

  /* Get magnetic field X/Y/Z in [uT] */
  MEMS_Read_MagValue(&data_in.Mag[0], &data_in.Mag[1], &data_in.Mag[2]);

  /* Get current sample time in [ms] */
  data_in.TimeStamp = timestamp * SAMPLE_TIME;

  /* Magnetometer calibration algorithm update */
  MotionMC_Update(&data_in);

  /* Get the magnetometer calibration coefficients */
  MotionMC_GetCalParams(&data_out);

  /* Apply calibration coefficients */
  mag_cal_x = (int)((data_in.Mag[0] - data_out.HI_Bias[0]) * SF_Matrix[0][0]
                + (data_in.Mag[1] - data_out.HI_Bias[1]) * SF_Matrix[0][1]
                + (data_in.Mag[2] - data_out.HI_Bias[2]) * SF_Matrix[0][2]);

  mag_cal_y = (int)((data_in.Mag[0] - data_out.HI_Bias[0]) * SF_Matrix[1][0]
                + (data_in.Mag[1] - data_out.HI_Bias[1]) * SF_Matrix[1][1]
                + (data_in.Mag[2] - data_out.HI_Bias[2]) * SF_Matrix[1][2]);

  mag_cal_z = (int)((data_in.Mag[0] - data_out.HI_Bias[0]) * SF_Matrix[2][0]
                + (data_in.Mag[1] - data_out.HI_Bias[1]) * SF_Matrix[2][1]
                + (data_in.Mag[2] - data_out.HI_Bias[2]) * SF_Matrix[2][2]);
}
```

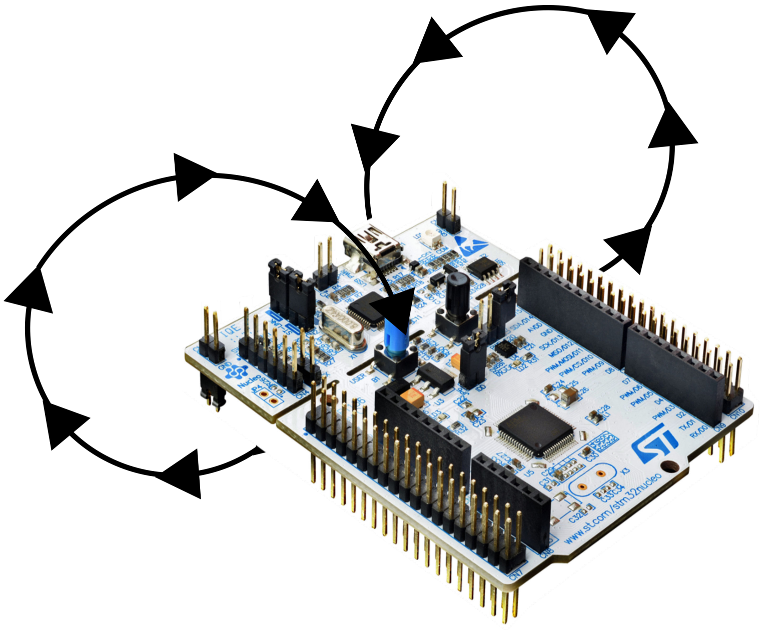
2.2.5 Calibration process

This calibration algorithm uses the specific motion of the magnetometer sensor exposed to Earth magnetic field.

Step 1. Rotate the STM32 Nucleo board slowly in a figure eight pattern through 3D space.

This is not a planar movement; it is necessary to tilt and rotate the device to cover as much different 3D space positions as possible.

Figure 2. STM32 Nucleo board rotation during calibration



Step 2. Use the 3D Plot (see Section 3.1: MEMS Studio application) during calibration; the measured data points should cover as much of the 3D sphere as possible.

Important: *While performing the calibration pattern, keep the STM32Nucleo board clear of other magnetic objects such as cell phones, computers and other steel objects.*

2.2.6 Algorithm performance

Table 2. Algorithm elapse time (µs) Cortex-M4, Cortex-M3 and Cortex-M0+

Cortex-M4 STM32F401RE at 84 MHz			Cortex-M3 STM32L152RE at 32 MHz			Cortex-M0+ STM32L073RZ at 32 MHz		
Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
1	6	1256	16	88	30344	<1	470	6000

Table 3. Algorithm elapse time (µs) Cortex-M33 and Cortex-M7

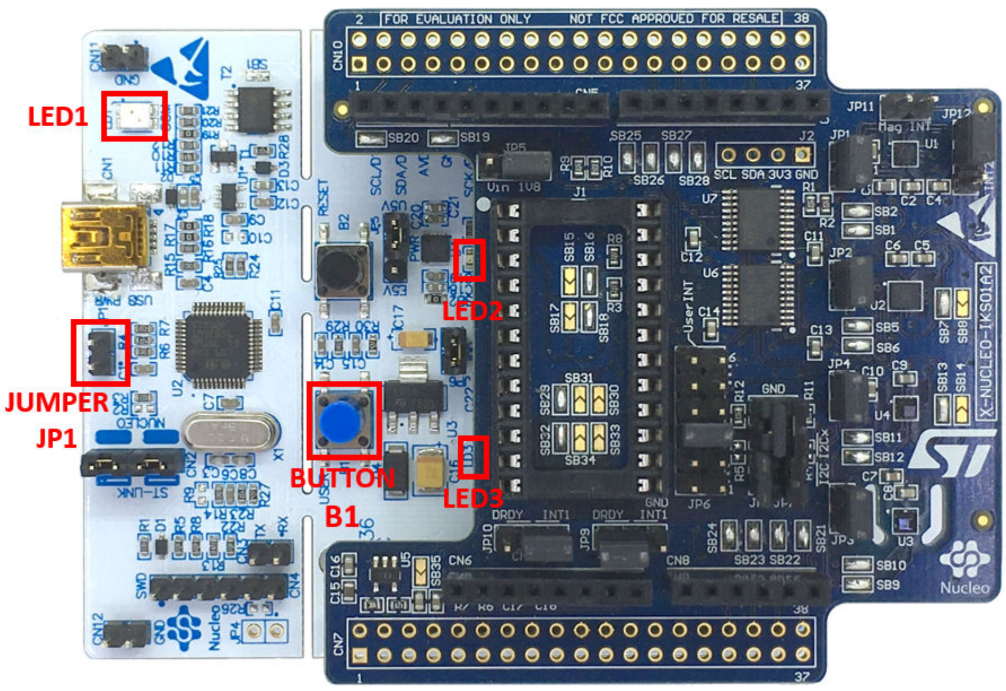
Cortex- M33 STM32U575ZI-Q at 160 MHz			Cortex- M7 STM32F767ZI at 96 MHz		
Min	Avg	Max	Min	Avg	Max
<1	2	584	2	8	1715

3 Sample application

The MotionMC 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-U575ZI-Q, NUCLEO-L152RE or NUCLEO-L073RZ development board connected to an X-NUCLEO-IKS4A1, X-NUCLEO-IKS01A3 or X-NUCLEO-IKS02A1 expansion board.

Figure 3. **STM32 Nucleo: LEDs, button, jumper**



The above figure shows the user button B1 and the three LEDs of the NUCLEO-F401RE board. Once the board is powered, LED LD3 (PWR) turns ON.

A USB cable connection is required to monitor real-time data. The board is powered by the PC via USB connection. This working mode allows the user to display magnetometer calibration algorithm data and eventually other sensor data, in real-time, using the MEMS-Studio.

3.1 MEMS Studio application

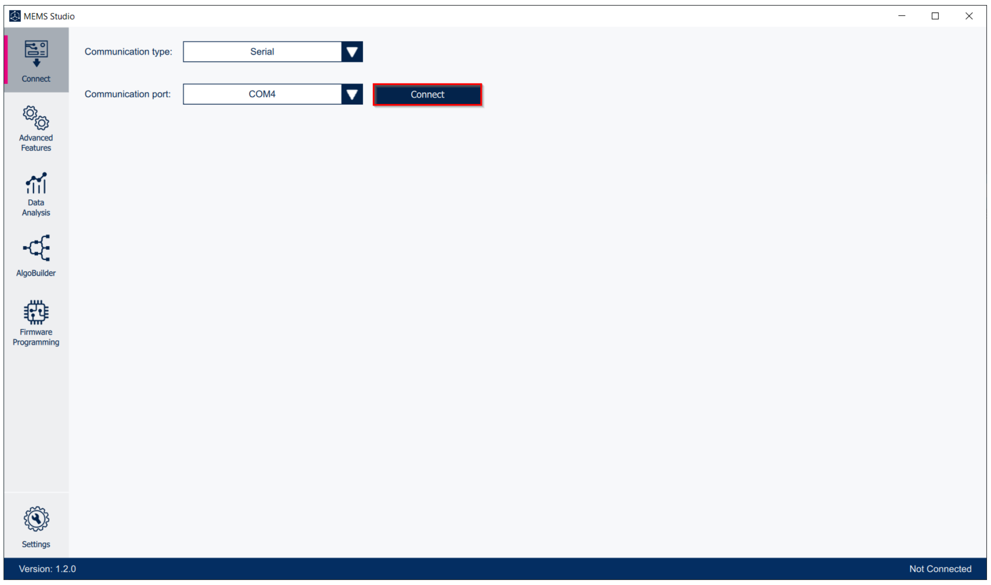
The sample application uses MEMS-Studio application, which can be downloaded from www.st.com.

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 MEMS-Studio application to open the main application window.

If an STM32 Nucleo board with supported firmware is connected to the PC, it is automatically detected. Press the [**Connect**] button to establish connection to the evaluation board.

Figure 4. MEMS-Studio - Connect



Communication type: Serial

Communication port: COM4

Version: 1.2.0

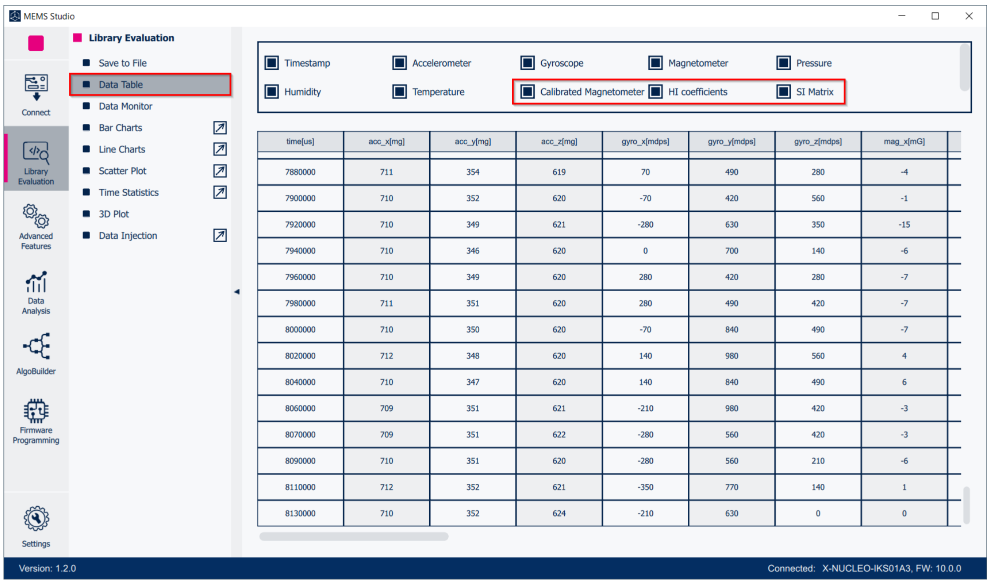
Not Connected

Step 3. When connected to a STM32 Nucleo board with supported firmware [**Library Evaluation**] tab is opened.

To start and stop data streaming, toggle the appropriate [**Start**] ▶ or [**Stop**] ■ button on the outer vertical tool bar.

The data coming from the connected sensor can be viewed selecting the [**Data Table**] tab on the inner vertical tool bar.

Figure 5. MEMS-Studio - Library Evaluation - Data Table



■ Timestamp ■ Accelerometer ■ Gyroscope ■ Magnetometer ■ Pressure
■ Humidity ■ Temperature ■ Calibrated Magnetometer ■ HI coefficients ■ SI Matrix

time[us]	acc_x[mg]	acc_y[mg]	acc_z[mg]	gyro_x[mdps]	gyro_y[mdps]	gyro_z[mdps]	mag_x[mG]
7880000	711	354	619	70	490	280	-4
7900000	710	352	620	-70	420	560	-1
7920000	710	349	621	-280	630	350	-15
7940000	710	346	620	0	700	140	-6
7960000	710	349	620	280	420	280	-7
7980000	711	351	620	280	490	420	-7
8000000	710	350	620	-70	840	490	-7
8020000	712	348	620	140	980	560	4
8040000	710	347	620	140	840	490	6
8060000	709	351	621	-210	980	420	-3
8070000	709	351	622	-280	560	420	-3
8090000	710	351	620	-280	560	210	-6
8110000	712	352	621	-350	770	140	1
8130000	710	352	624	-210	630	0	0

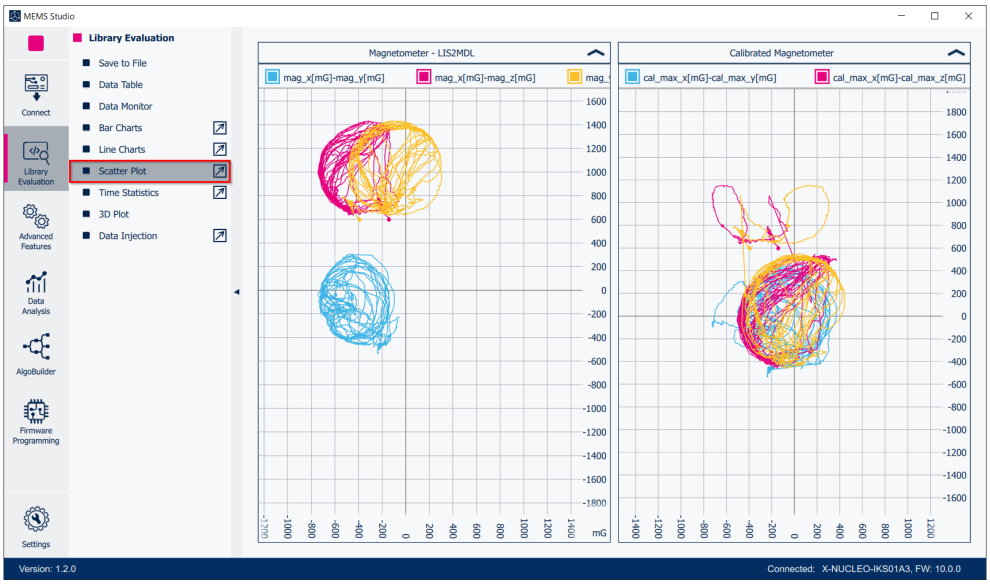
Version: 1.2.0

Connected: X-NUCLEO-IKS01A3, FW: 10.0.0

Step 4. Click on [**Scatter Plot**] .

The uncalibrated data window shows the hard-iron distortion. The calibrated data window shows all the data centered on the origin (0, 0, 0) as the hard-iron distortion has been removed through calibration.

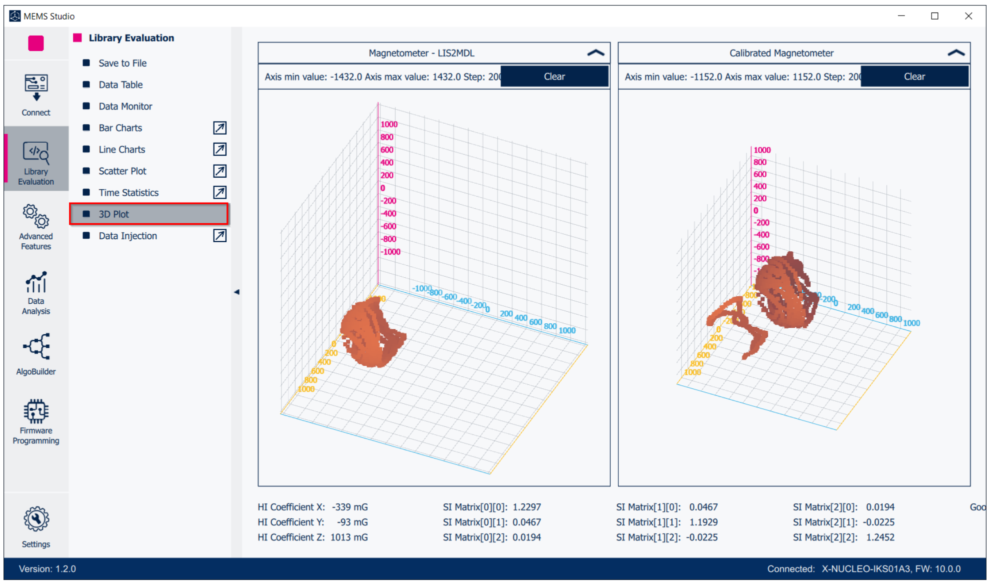
Figure 6. Scatter Plot - Uncalibrated/Calibrated data



Step 5. Click on [**3D plot**].

The uncalibrated data window shows the hard-iron distortion: the 3D sphere is subject to offsets in x, y and z directions. The calibrated data window shows the 3D sphere centered on the origin (0, 0, 0) as the hard-iron distortion has been removed through calibration. The 3D Plot page also shows the hard-iron and soft-iron coefficients and calibration quality (goodness).

Figure 7. 3D Plot - Uncalibrated/Calibrated data



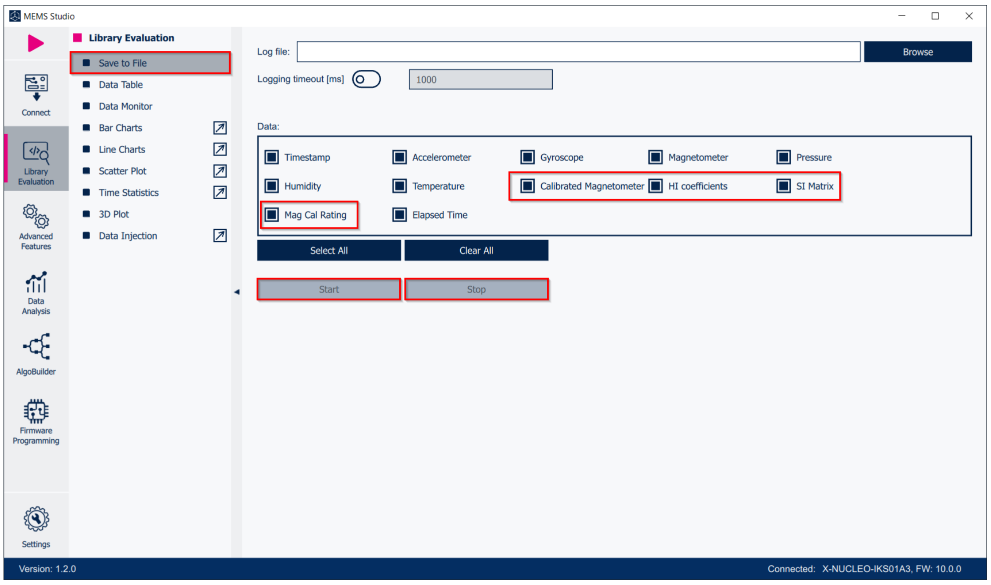
Axis min value: -1432.0 Axis max value: 1432.0 Step: 20
Axis min value: -1152.0 Axis max value: 1152.0 Step: 20

```
HI Coefficient X:  -339 mG    SI Matrix[0][0]: 1.2297    SI Matrix[1][0]:  0.0467    SI Matrix[2][0]:  0.0194    Goo
HI Coefficient Y:   -93 mG    SI Matrix[0][1]: 0.0467    SI Matrix[1][1]:  1.1929    SI Matrix[2][1]: -0.0225
HI Coefficient Z:  1013 mG    SI Matrix[0][2]: 0.0194    SI Matrix[1][2]: -0.0225    SI Matrix[2][2]:  1.2452
```

Connected: X-NUCLEO-IKS01A3, FW: 10.0.0

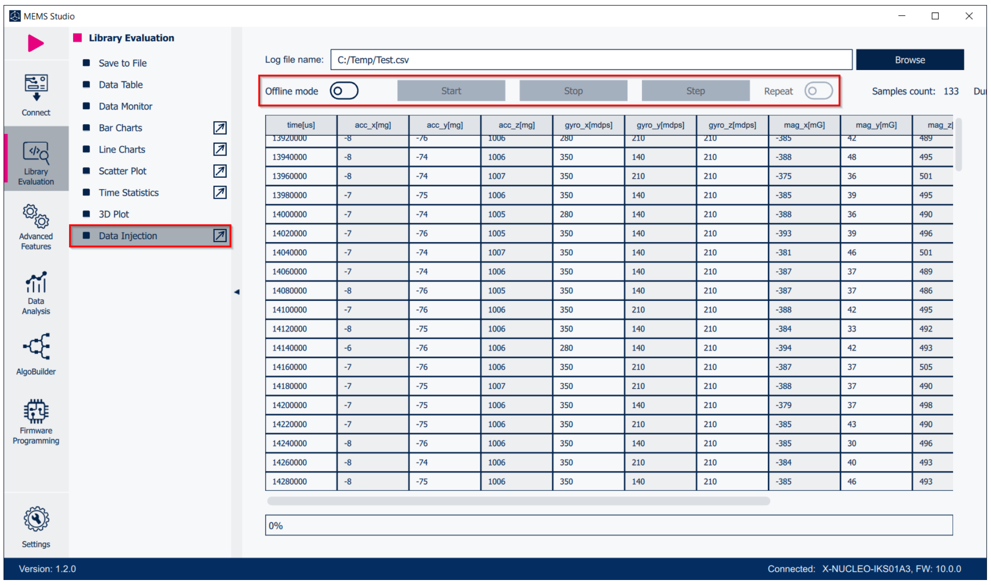
Step 6. Click on the [**Save To File**] to open the dataloging configuration window. Select the sensor and magnetometer calibration data to be saved in the file. You can start or stop saving by clicking on the corresponding button.

Figure 8. MEMS-Studio - Library Evaluation - Save To File



Step 7. Data Injection mode can be used to send the previously acquired data to the library and receive the result. Select the [**Data Injection**] tab on the vertical tool bar to open the dedicated view for this functionality.

Figure 9. MEMS-Studio - Library Evaluation - Data Injection



Log file name: C:/Temp/Test.csv

Samples count: 133

time[us]	acc_x[mg]	acc_y[mg]	acc_z[mg]	gyro_x[mdps]	gyro_y[mdps]	gyro_z[mdps]	mag_x[mG]	mag_y[mG]	mag_z
13920000	-8	-76	1006	280	210	210	-385	42	489
13940000	-8	-74	1006	350	140	210	-388	48	495
13960000	-8	-74	1007	350	210	210	-375	36	501
13980000	-7	-75	1006	350	140	210	-385	39	495
14000000	-7	-74	1005	280	140	210	-388	36	490
14020000	-7	-76	1005	350	140	210	-393	39	496
14040000	-7	-74	1007	350	140	210	-381	46	501
14060000	-7	-74	1006	350	140	210	-387	37	489
14080000	-8	-76	1005	350	140	210	-387	37	486
14100000	-7	-76	1006	350	210	210	-388	42	495
14120000	-8	-75	1006	350	140	210	-384	33	492
14140000	-6	-76	1006	280	140	210	-394	42	493
14160000	-7	-76	1006	350	210	210	-387	37	505
14180000	-7	-75	1007	350	210	210	-388	37	490
14200000	-7	-75	1006	350	140	210	-379	37	498
14220000	-7	-75	1006	350	210	210	-385	43	490
14240000	-8	-76	1006	350	140	210	-385	30	496
14260000	-8	-74	1006	350	210	210	-384	40	493
14280000	-8	-75	1006	350	140	210	-385	46	493

0%

Connected: X-NUCLEO-IKS01A3, FW: 10.0.0

Step 8. Click on the [**Browse**] button to select the file with the previously captured data in CSV format.

The data will be loaded into the table in the current view.
Other buttons will become active. You can click on:

– [**Offline Mode**] button to switch the firmware offline mode on/off (mode utilizing the previously captured data).
– [**Start**]/[**Stop**]/[**Step**]/[**Repeat**] buttons to control the data feed from MEMS-Studio to the library.

4 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. UM3233: Getting started with MEMS-Studio

Revision history

Table 4. **Document revision history**

Date	Version	Changes
10-Apr-2017	1	Initial release.
26-Jan-2018	2	Updated Section 2.1 MotionMC overview. Added references to NUCLEO-L152RE development board and Section 2.2.6 Algorithm performance.
20-Mar-2018	3	Updated Introduction and Section 2.1 MotionMC overview.
03-May-2018	4	Added Table 3. Cortex-M0+: elapsed time (µs) algorithm.. Added references to Cortex-M0+ and NUCLEO-L073RZ development board.
21-Feb-2019	5	Updated Figure 3. Sensor expansion board and adapter connected to the STM32 Nucleo, Table 2. Cortex-M4, Cortex-M3: elapsed time (µs) algorithm, Table 3. Cortex-M0+: elapsed time (µs) algorithm, Figure 4. Unicleo main window, Figure 5. User Messages tab, Figure 6. 2D Plot Uncalibrated/Calibrated windows, Figure 7. 3D Plot Uncalibrated/Calibrated windows and Figure 8. Datalog window. Added X-NUCLEO-IKS01A3 expansion board compatibility information.
24-Mar-2020	6	Updated Introduction, Section 2.2.1 MotionMC library description and Section 2.2.6 Algorithm performance. Added ARM Cortex-M7 architecture compatibility information.
07-Apr-2021	7	Updated Introduction, *Section 2.1: MotionMC overview, Section 2.2.1: MotionMC library description,* *Section 2.2.2: MotionMC APIs* and *Section 3: Sample application.* *Added X-NUCLEO-IKS02A1 compatibility information.*
05-Jun-2025	8	Updated Section Introduction, Section 2.1: MotionMC overview, Section 2.2: MotionMC library, Section 3: Sample application and Section 4: References.
21-Aug-2025	9	Updated Section 2.2.2: MotionMC APIs.

Contents

List of tables

List of figures