
Getting started with MotionFA fitness activity library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionFA middleware library is a component of the [X-CUBE-MEMS1](#) software and runs on STM32. It provides real-time information about the repetition quantity of various fitness activities performed by a user. It is able to count the number of bicep curls, squats and push-ups. The library is intended for wrist-worn devices.

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[®]-M33, ARM Cortex-M4 or ARM[®] Cortex[®]-M7 architecture.

It is built on [STM32Cube](#) software technology for portability across different STM32 microcontrollers.

The software comes with a sample implementation running on [X-NUCLEO-IKS4A1](#) or [X-NUCLEO-IKS01A3](#) expansion board on a [NUCLEO-F401RE](#), [NUCLEO-U575ZI-Q](#) 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 MotionFA middleware library in X-CUBE-MEMS1 software expansion for STM32Cube

2.1 MotionFA overview

The MotionFA library expands the functionality of the [X-CUBE-MEMS1](#) software.

The library acquires data from the accelerometer, gyroscope and pressure sensor and provides information about the repetition quantity of various fitness activities performed by a user.

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.

A sample implementation is available for [X-NUCLEO-IKS4A1](#) and [X-NUCLEO-IKS01A3](#) expansion boards, mounted on a [NUCLEO-F401RE](#), [NUCLEO-U575ZI-Q](#) or [NUCLEO-L152RE](#) development board.

2.2 MotionFA library

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

2.2.1 MotionFA library description

The MotionFA fitness activity library manages the data acquired from the accelerometer, gyroscope and pressure sensor; it features:

- possibility to count the number of bicep curls, squats and push-ups
- bicep curl and squat recognition based on accelerometer and pressure sensor data
- push-up recognition based on accelerometer and gyroscope sensor data
- required accelerometer, gyroscope and pressure sensor data sampling frequency of 25 Hz
- resources requirements:
 - Cortex-M3: 12.6 kB of code and 17.0 kB of data memory
 - Cortex-M33: 11.3 kB of code and 17.0 kB of data memory
 - Cortex-M4: 11.6 kB of code and 17.0 kB of data memory
 - Cortex-M7: 11.3 kB of code and 17.0 kB of data memory
- available for ARM® Cortex®-M3, ARM® Cortex®-M33, ARM® Cortex®-M4 and ARM® Cortex®-M7 architectures

2.2.2 MotionFA APIs

The MotionFA library APIs are:

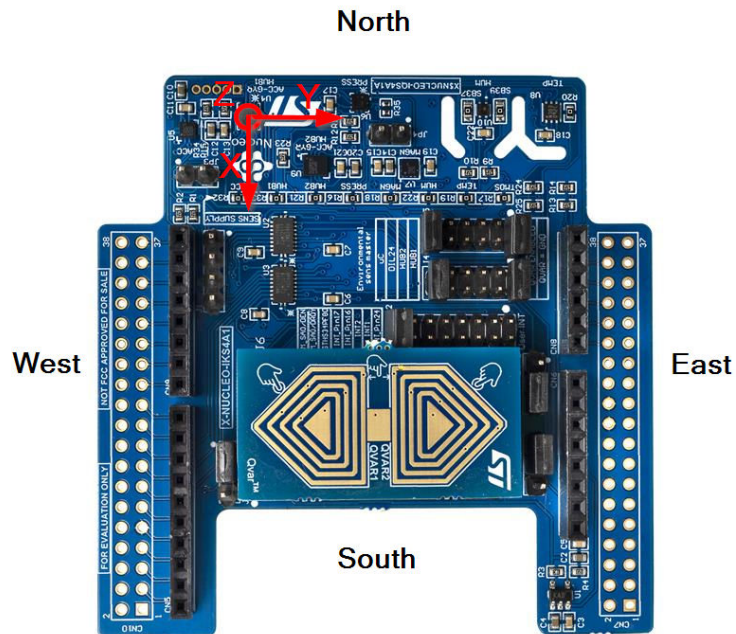
- `uint8_t MotionFA_GetLibVersion(char *version)`
 - retrieves the library version
 - `*version` is a pointer to an array of 35 characters
 - returns the number of characters in the version string
- `void MotionFA_Initialize(void)`
 - performs MotionFA library initialization and setup of the internal mechanism

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

- `void MotionFA_BicepCurl_Update(MFA_input_t *data_in, MFA_output_t *data_out)`
 - executes bicep curl detection and counter algorithm
 - `*data_in` parameter is a pointer to a structure with input data
 - the parameters for the structure type `MFA_input_t` are:
 - `AccX` is the accelerometer sensor value in X axis in g
 - `AccY` is the accelerometer sensor value in Y axis in g
 - `AccZ` is the accelerometer sensor value in Z axis in g
 - `GyrX, GyrY, GyrZ` is not used during bicep curl recognition
 - `Press` is the atmospheric pressure in hPa
 - `*data_out` parameter is a pointer to a structure `MFA_output_t` with the following parameter:
 - `Counter` is the number of bicep curls
- `void MotionFA_Squat_Update(MFA_input_t *data_in, MFA_output_t *data_out)`
 - executes squat detection and counter algorithm
 - `*data_in` parameter is a pointer to a structure with input data
 - the parameters for the structure type `MFA_input_t` are the following:
 - `AccX` is the accelerometer sensor value in X axis in g
 - `AccY` is the accelerometer sensor value in Y axis in g
 - `AccZ` is the accelerometer sensor value in Z axis in g
 - `GyrX, GyrY, GyrZ` is not used during bicep curl recognition
 - `Press` is the atmospheric pressure in hPa
 - `*data_out` is a pointer to a structure `MFA_output_t` with the following parameter:
 - `Counter` is the number of squats
- `void MotionFA_Pushup_Update(MFA_input_t *data_in, MFA_output_t *data_out)`
 - executes push-up detection and counter algorithm
 - `*data_in` parameter is a pointer to a structure with input data
 - the parameters for the structure type `MFA_input_t` are the following:
 - `AccX` is the accelerometer sensor value in X axis in g
 - `AccY` is the accelerometer sensor value in Y axis in g
 - `AccZ` is the accelerometer sensor value in Z axis in g
 - `GyrX` is the gyroscope sensor value in X axis in dps
 - `GyrY` is the gyroscope sensor value in Y axis in dps
 - `GyrZ` is the gyroscope sensor value in Z axis in dps
 - `Press` is not used during push-up recognition
 - `*data_out` is a pointer to a structure `MFA_output_t` with the following parameter:
 - `Counter` is the number of push-ups
- `void Motion_BicepCurl_Reset(void)`
 - resets bicep curl counter
- `void MotionFA_Squat_Reset(void)`
 - resets squat counter
- `void MotionFA_Pushup_Reset(void)`
 - resets push-up counter

- `void MotionFA_SetOrientation_Acc (const char *acc_orientation)`
 - this function is used to set the accelerometer data orientation
 - configuration is usually performed immediately after the `MotionFA_Initialize` function call
 - `*acc_orientation` parameter is a pointer to a string of three characters indicating the direction of each of the positive orientations of the reference frame used for accelerometer data output, in the sequence x, y, z. Valid values are: n (north) or s (south), w (west) or e (east), u (up) or d (down).
 - As shown in the figure below, the **X-NUCLEO-IKS4A1** accelerometer sensor has an SEU (x - South, y - East, z - Up), so the string is: "seu".

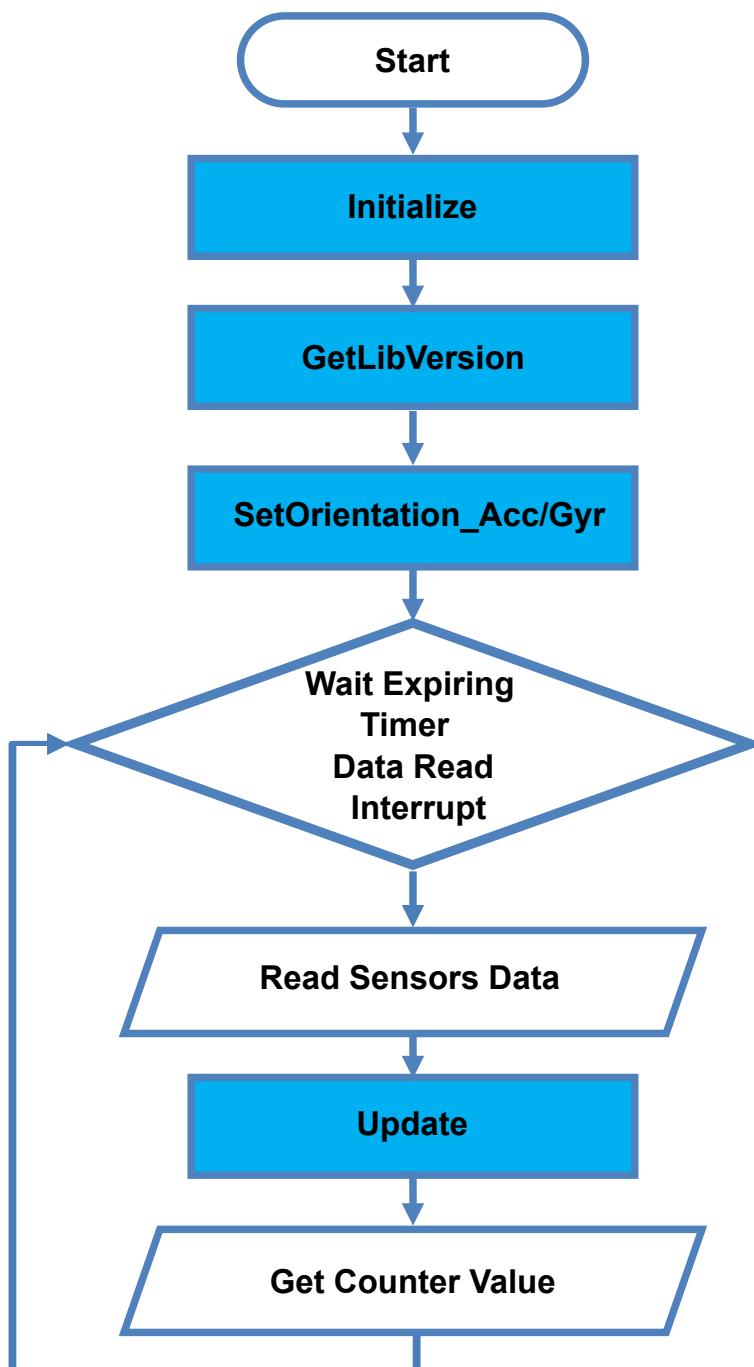
Figure 1. Example of sensor orientations



- `void MotionFA_SetOrientation_Gyr (const char *gyr_orientation)`
 - this function is used to set the gyroscope data orientation
 - configuration is usually performed immediately after the `MotionFA_Initialize` function call
 - `*gyr_orientation` parameter is a pointer to a string of three characters indicating the direction of each of the positive orientations of the reference frame used for gyroscope data output, in the sequence x, y, z. Valid values are: n (north) or s (south), w (west) or e (east), u (up) or d (down).
 - As shown in the figure above, the **X-NUCLEO-IKS4A1** gyroscope sensor has an SEU (x - South, y - East, z - Up), so the string is: "seu".

2.2.3 API flow chart

Figure 2. MotionFA API logic sequence



2.2.4

Demo code

The following demonstration code reads data from the accelerometer, gyroscope and pressure sensor and gets the activity code.

```
[...]
#define VERSION_STR LENG 35
[...]

/* Initialization */
char lib_version[VERSION_STR LENG];
char acc_orientation[3];
char gyr_orientation[3];
MFA_activity_t activity;

/* Fitness Activities API initialization function */
MotionFA_Initialize();

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

/* Set accelerometer orientation */
acc_orientation[0] = 'n';
acc_orientation[1] = 'w';
acc_orientation[2] = 'u';
MotionFA_SetOrientation_Acc(acc_orientation);

/* Set gyroscope orientation */
gyr_orientation[0] = 'n';
gyr_orientation[1] = 'w';
gyr_orientation[2] = 'u';
MotionFA_SetOrientation_Gyr(gyr_orientation);

/* Select activity i.e. bicep curl */
activity = MFA_BICEPCURL;

[...]

/* Using Fitness Activities algorithm */
Timer_OR_DataRate_Interrupt_Handler()
{
    MFA_input_t data_in;
    MFA_output_t data_out;

    /* Get acceleration X/Y/Z in g */
    MEMS_Read_AccValue(&data_in.AccX, &data_in.AccY, &data_in.AccZ);

    /* Get angular velocity X/Y/Z in dps */
    MEMS_Read_GyrValue(&data_in.GyrX, &data_in.GyrY, &data_in.GyrZ);

    /* Get atmospheric pressure in hPa */
    MEMS_Read_PressValue(&data_in.Press);

    /* Fitness Activities algorithm update */
    if (activity == MFA_BICEPCURL)
    {
        MotionFA_BicepCurl_Update(&data_in, &data_out);
    }
    else if (activity == MFA_SQUAT)
    {
        MotionFA_Squat_Update(&data_in, &data_out);
    }
    else if (activity == MFA_PUSHUP)
    {
        MotionFA_Pushup_Update(&data_in, &data_out);
    }
}
```

2.2.5 Algorithm performance

The fitness activity algorithm uses data from the accelerometer, gyroscope and pressure sensor and runs at a low frequency (25 Hz) to reduce power consumption.

When replicating fitness activity with the [STM32 Nucleo](#) board, ensure the board is oriented perpendicularly to the forearm, to simulate the wristband position.

Figure 3. Orientation system for wrist-worn devices

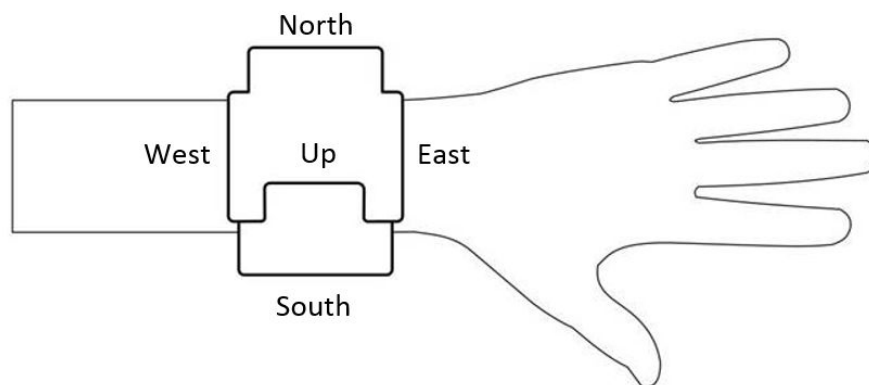


Table 2. Algorithm elapse time (μs) Cortex-M4, Cortex-M3

Cortex-M4 STM32F401RE at 84 MHz			Cortex-M3 STM32L152RE at 32 MHz		
Min	Avg	Max	Min	Avg	Max
2	58	7885	30	700	91886

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	31	4055	4	55	5104

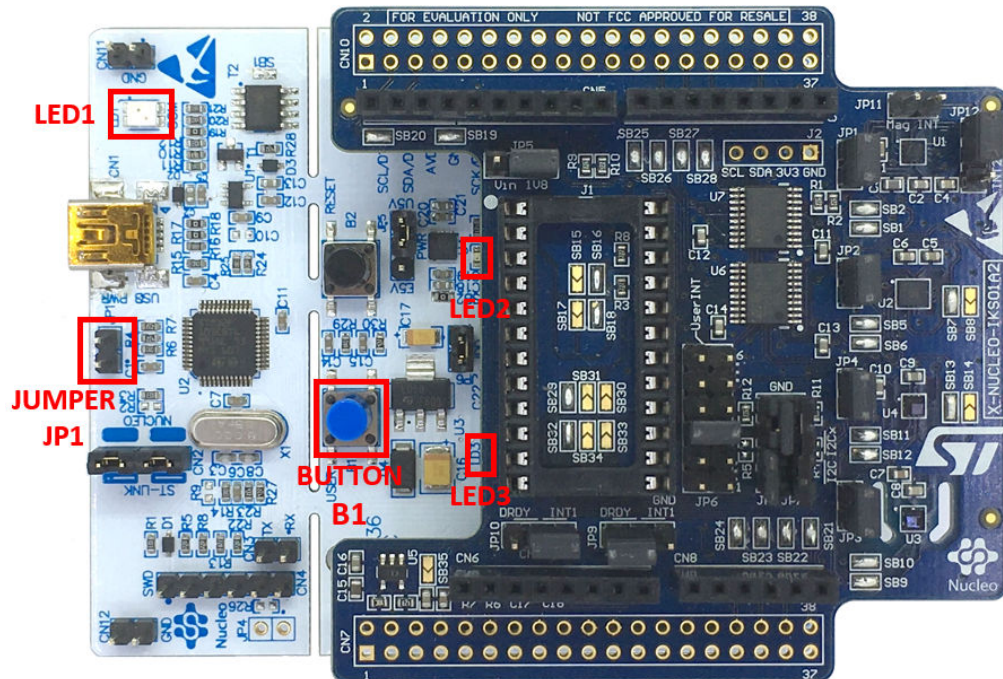
2.3 Sample application

The MotionFA 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](#) or [NUCLEO-L152RE](#) development board connected to an [X-NUCLEO-IKS4A1](#) or [X-NUCLEO-IKS01A3](#) expansion board.

The application counts the number of bicep curls, squats or push-ups in real-time.

Figure 4. 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 real-time counter values, accelerometer, gyroscope and pressure data, time stamp and any other sensor data, using the [MEMS-Studio](#).

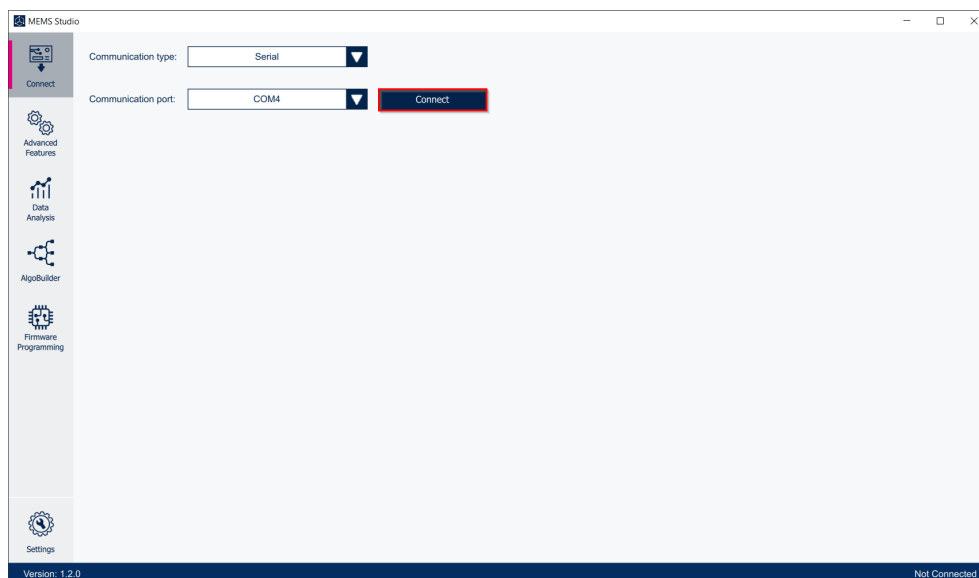
2.4 MEMS Studio application

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 5. MEMS-Studio - Connect

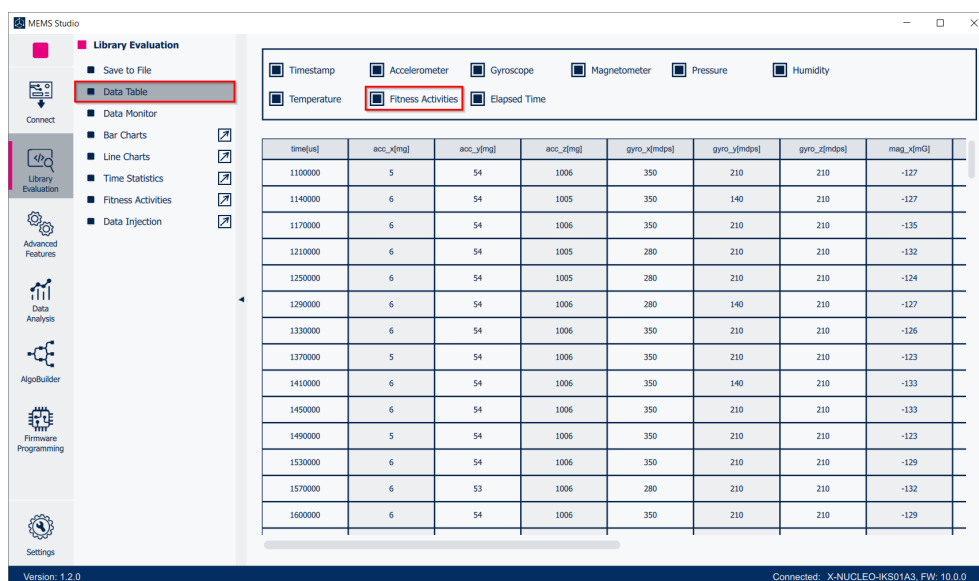


- Step 3.** When connected to a **STM32 Nucleo** board with supported firmware **[Library Evaluation]** tab is opened.

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

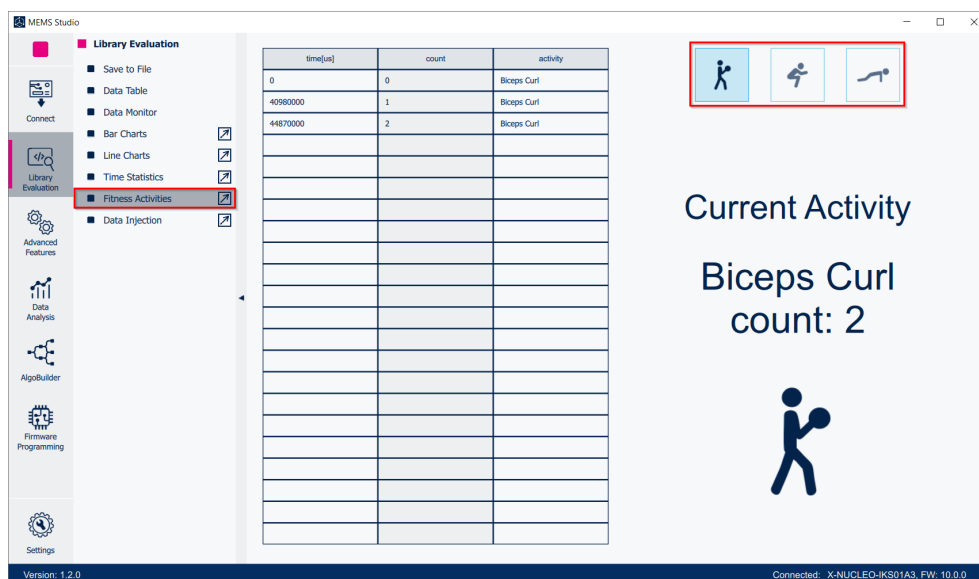
The data coming from the connected sensor can be viewed selecting the **[Data Table]** tab on the inner vertical tool bar.

Figure 6. MEMS-Studio - Library Evaluation - Data Table



Step 4. Click on the **[Fitness Activities]** to open the dedicated application window.

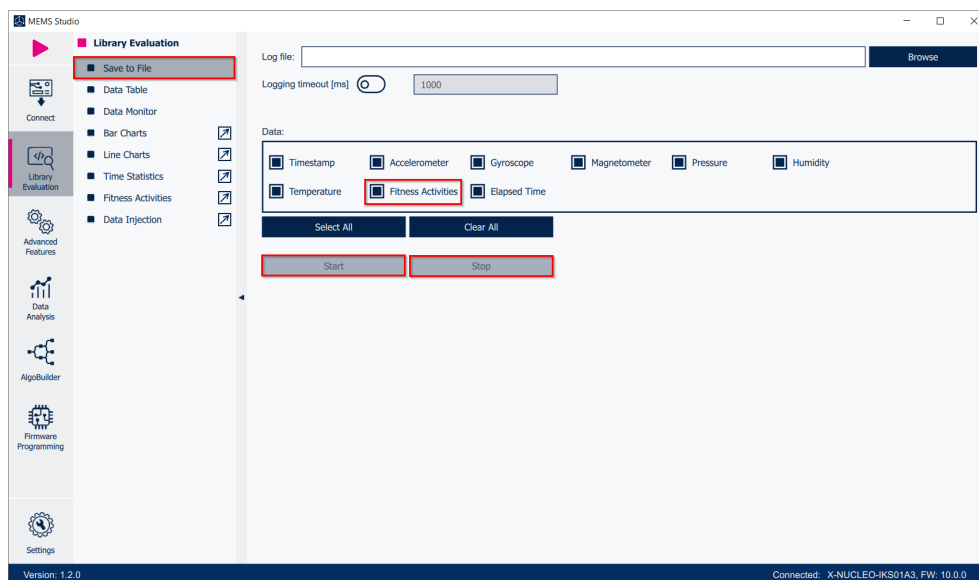
Figure 7. MEMS-Studio - Library Evaluation - Fitness Activities



To switch between bicep curl, squat and push-up counting click on the appropriate icon.

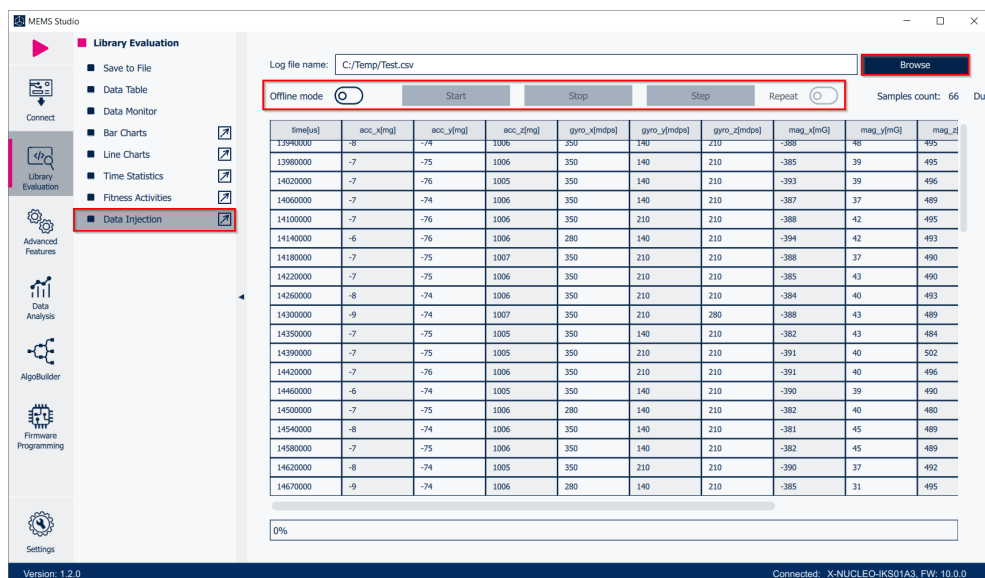
Step 5. Click on the **[Save To File]** to open the datalogging configuration window. Select the sensor and fitness activities data to be saved in the file. You can start or stop saving by clicking on the corresponding button.

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



Step 6. 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



Step 7. 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.

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. [UM3233](#): Getting started with MEMS-Studio

Revision history

Table 4. Document revision history

Date	Version	Changes
02-May-2017	1	Initial release.
06-Feb-2018	2	Added references to NUCLEO-L152RE development board and Table 2. Elapsed time (μ s) algorithm.
20-Mar-2018	3	Updated Introduction and Section 2.1 MotionFA overview.
01-Oct-2018	4	Removed references to X-NUCLEO-IKS01A1 expansion board throughout document. Updated Section 2.2.1 MotionFA library description, Section 2.2.2 MotionFA APIs, Section 2.2.3 API flow chart, Section 2.2.4 Demo code, Figure 3. STM32 Nucleo: LEDs, button, jumper and Section 2.4 Unicleo-GUI application.
21-Feb-2019	5	Updated <i>Section 2.2.5: Algorithm performance</i> . Added X-NUCLEO-IKS01A3 expansion board compatibility information.
17-Sep-2024	6	Updated <i>Section Introduction</i> , <i>Section 2.1: MotionFA overview</i> , <i>Section 2.2.1: MotionFA library description</i> , <i>Section 2.2.2: MotionFA APIs</i> , <i>Section 2.2.5: Algorithm performance</i> , <i>Section 2.3: Sample application</i> , <i>Section 2.4: MEMS Studio application</i>

Contents

1	Acronyms and abbreviations	2
2	MotionFA middleware library in X-CUBE-MEMS1 software expansion for STM32Cube	3
2.1	MotionFA overview	3
2.2	MotionFA library	3
2.2.1	MotionFA library description	3
2.2.2	MotionFA APIs	3
2.2.3	API flow chart	6
2.2.4	Demo code	7
2.2.5	Algorithm performance	8
2.3	Sample application	8
2.4	MEMS Studio application	9
3	References	13
	Revision history	14

List of tables

Table 1.	List of acronyms	2
Table 2.	Algorithm elapse time (μ s) Cortex-M4, Cortex-M3	8
Table 3.	Algorithm elapse time (μ s) Cortex-M33 and Cortex-M7	8
Table 4.	Document revision history	14

List of figures

Figure 1.	Example of sensor orientations	5
Figure 2.	MotionFA API logic sequence	6
Figure 3.	Orientation system for wrist-worn devices	8
Figure 4.	STM32 Nucleo: LEDs, button, jumper	9
Figure 5.	MEMS-Studio - Connect	10
Figure 6.	MEMS-Studio - Library Evaluation - Data Table.	10
Figure 7.	MEMS-Studio - Library Evaluation - Fitness Activities	11
Figure 8.	MEMS-Studio - Library Evaluation - Save To File	11
Figure 9.	MEMS-Studio - Library Evaluation - Data Injection	12

IMPORTANT NOTICE – 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 acknowledgment.

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. For additional information about ST trademarks, refer to www.st.com/trademarks. 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.

© 2024 STMicroelectronics – All rights reserved