UM2223

User manual

Getting started with MotionPE real-time pose estimation library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionPE middleware library is part of the X-CUBE-MEMS1 software and runs on STM32. It provides real-time information about the user current pose based on data from a device.

It is able to distinguish the following poses: sitting, standing and lying down. 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 top of STM32Cube software technology to ease portability across different STM32 microcontrollers.

The software comes with sample implementation running on X-NUCLEO-IKS01A3 or X-NUCLEO-IKS4A1expansion 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 MotionPE middleware library in X-CUBE-MEMS1 software expansion for STM32Cube

2.1 MotionPE overview

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

The library acquires data from the accelerometer and provides information about the user current pose based on data from a device.

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-IKS01A3 and X-NUCLEO-IKS4A1 expansion boards, mounted on a NUCLEO-F401RE, NUCLEO-U575ZI-Q or NUCLEO-L152RE development board.

2.2 MotionPE library

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

2.2.1 MotionPE library description

The MotionPE pose estimation library manages the data acquired from the accelerometer; it features:

- possibility to distinguish the following user poses: sitting, standing, lying down
- intended for wrist-worn devices
- recognition based on the accelerometer data only
- required accelerometer data sampling frequency of 16 Hz
- resources requirements:
  – Cortex-M3: 12.0 kB of code and 2.8 kB of data memory
  – Cortex-M33: 12.5kB of code and 2.8kB of data memory
  – Cortex-M4: 12.9 kB of code and 2.8 kB of data memory
  – Cortex-M7: 12.9 kB of code and 2.8kB of data memory
- available for ARM® Cortex®-M3, ARM Cortex®-M33, ARM® Cortex®-M4 or ARM® Cortex®-M7 architectures.

2.2.2 MotionPE APIs

The MotionPE library APIs are:

- `uint8_t MotionPE_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 MotionPE_Initialize(void)`
  – performs MotionPE library initialization and setup of the internal mechanism

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

- `void MotionPE_ResetLib(void)`
  – reset the library

- `void MotionPE_Update(MPE_input_t *data_in, MPE_output_t *data_out)`
  - executes pose estimation algorithm
  - `*data_in` parameter is a pointer to a structure with input data
  - the parameters for the structure type `MPE_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
  - `*data_out` parameter is a pointer to an enum with the following items:
    - `MPE_UNKNOWN = 0`
    - `MPE_SITTING = 1`
    - `MPE_STANDING = 2`
    - `MPE_LYING_DOWN = 3`
- `void MotionPE_SetOrientation_Acc(const char *acc_orientation)`
  - this function is used to set the accelerometer data orientation
  - configuration is usually performed immediately after the `MotionPE_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 orientation (x - South, y - East, z - Up), so the string is: “seu”.

Figure 1. Example of sensor orientations

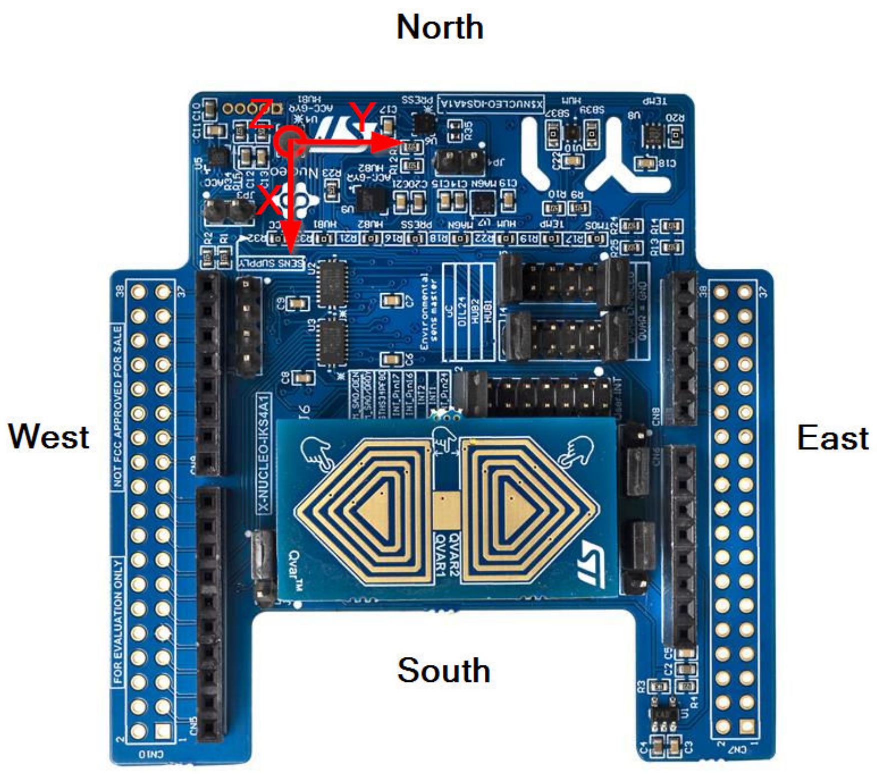
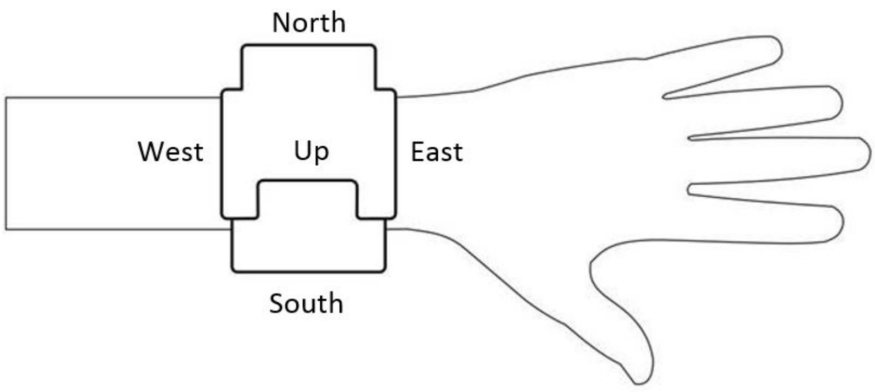
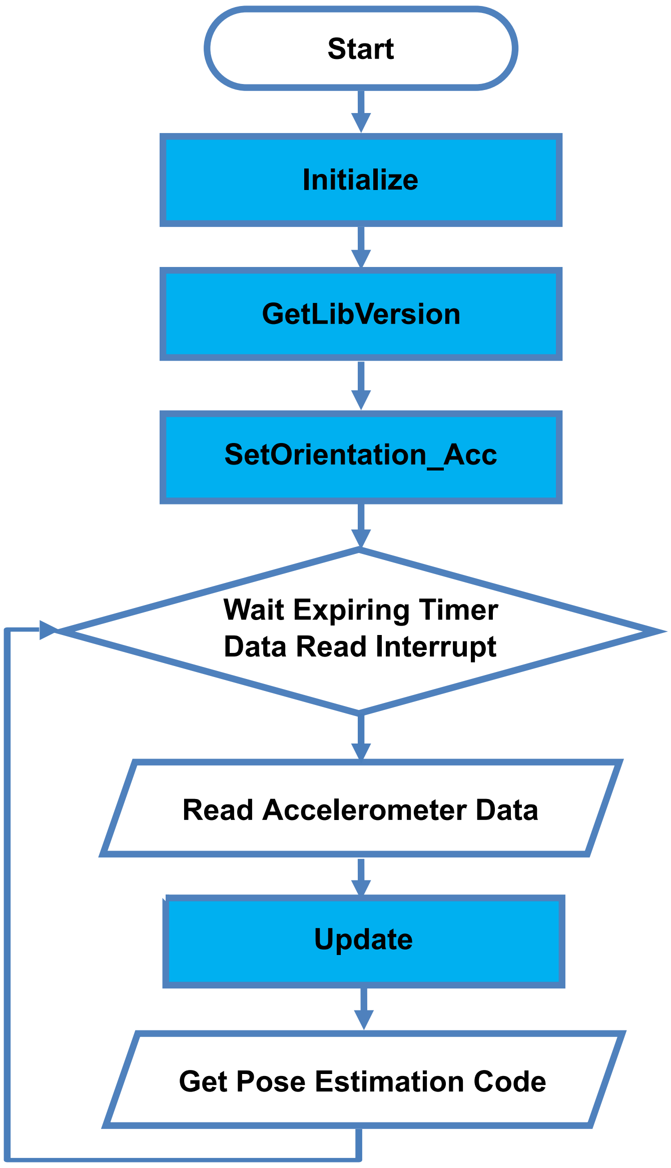


Figure 2. Orientation system for wrist-worn devices



2.2.3 API flow chart

Figure 3. MotionPE API logic sequence



2.2.4 Demo code

The following demonstration code reads data from the accelerometer sensor and gets the estimated pose.

```
[…]
#define VERSION_STR_LENG 35
[…]

/* Initialization */
char lib_version[VERSION_STR_LENG];
char acc_orientation[3];

/* Pose Estimation API initialization function */
MotionPE_Initialize();

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

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

[…]

/* Using Pose Estimation algorithm */
Timer_OR_DataRate_Interrupt_Handler()
{
  MPE_input_t data_in;
  MPE_output_t data_out;

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

  /* Pose Estimation algorithm update */
  MotionPE_Update(&data_in, &data_out);
}
```

2.2.5 Algorithm performance

The pose estimation algorithm only uses data from the accelerometer and runs at a low frequency (16 Hz) to reduce power consumption.

The table below shows the performance of the pose estimation algorithm in terms of recognition success rates.

Table 2. **Algorithm performance data**

Pose	Detection probability (typical)(1)	Minimum latency	Typical latency
Sitting	83.30%	15 s	30 s
Standing	75.05%	15 s	30 s
Lying	89.73%	3 min	-

1. *Typical specifications are not guaranteed.*

Table 3. **Algorithm elapse time (µs) Cortex-M3 and Cortex-M33**

Cortex-M3 STM32L152RE at 32 MHz			Cortex- M33 STM32U575ZI-Q at 160 MHz		
Min	Avg	Max	Min	Avg	Max
205	400	6050	25	29	143

Table 4. **Algorithm elapse time (µs) Cortex-M4 and Cortex-M7**

Cortex-M4 STM32F401RE at 84 MHz			Cortex- M7 STM32F767ZI at 96 MHz		
Min	Avg	Max	Min	Avg	Max
50	57	286	106	118	385

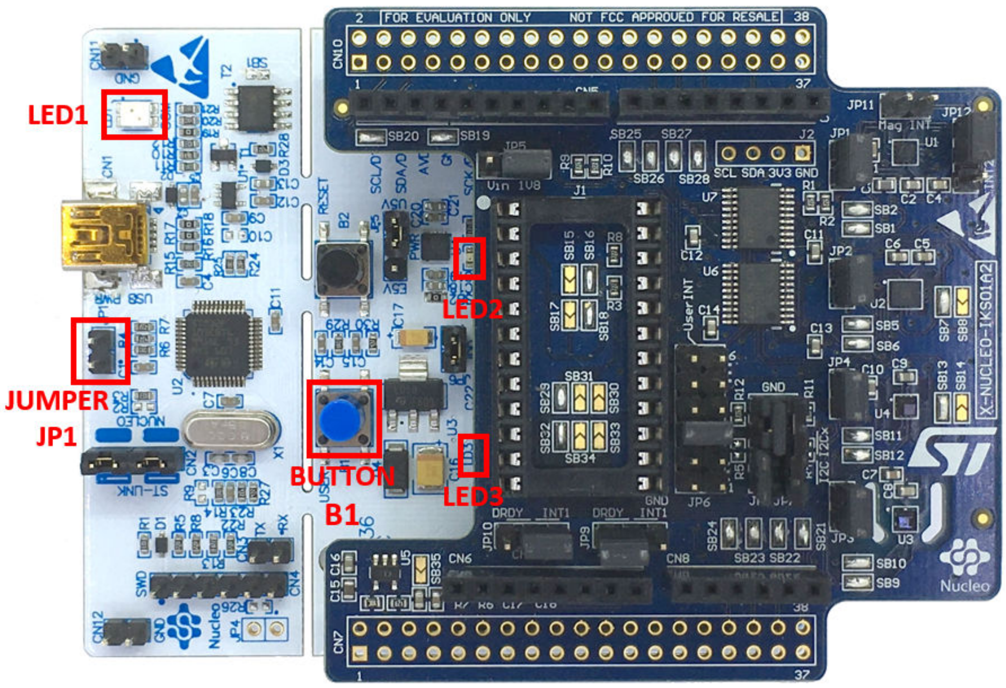
2.3 Sample application

The MotionPE 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-IKS01A3 or X-NUCLEO-IKS4A1 expansion board.

The application recognizes the current user pose 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.

Note: *After powering the board, LED LD2 blinks once indicating the application is ready.*

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 detected user pose, accelerometer data, time stamp and any other sensor data, using the MEMS-Studio.

2.4 MEMS-Studio application

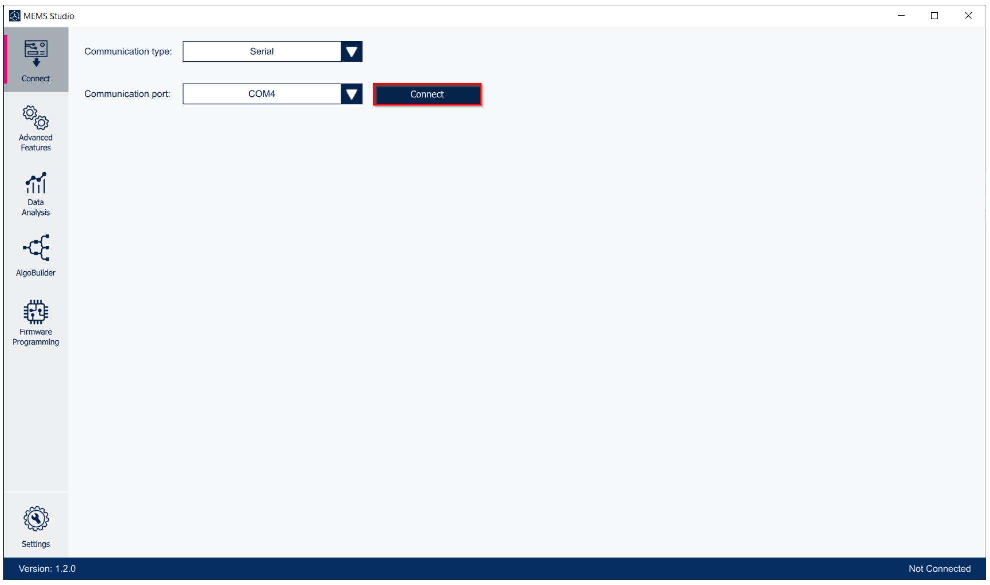
The sample application uses the MEMS-Studio application, which can be downloaded from www.st.com.

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, the appropriate COM port is automatically detected. Press the [**Connect**] button to establish connection to the evaluation board.

Figure 5. MEMS-Studio - Connect

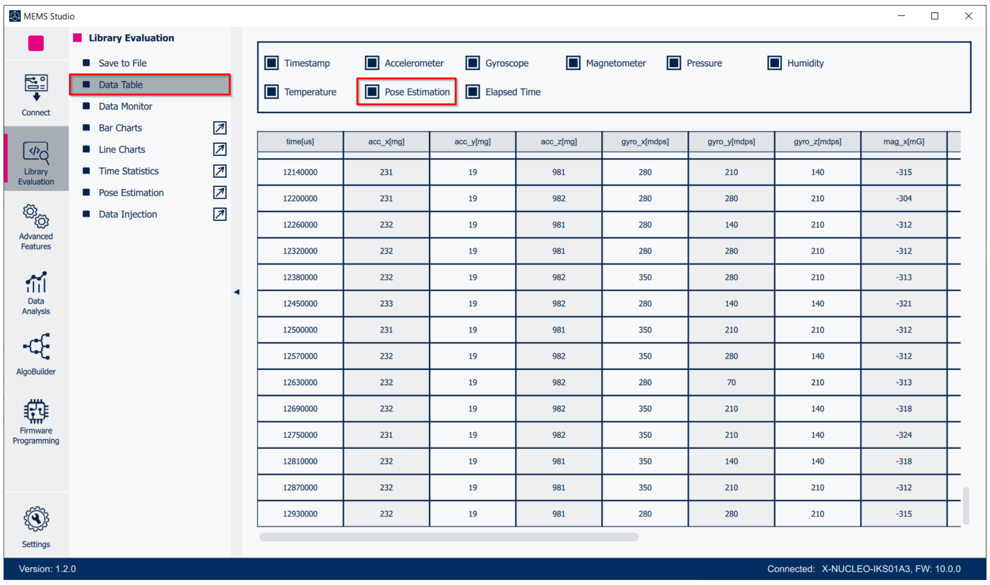


Step 3. When connected to a STM32 Nucleo board with supported firmware [**Library Evaluation**] tab is opened.

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.

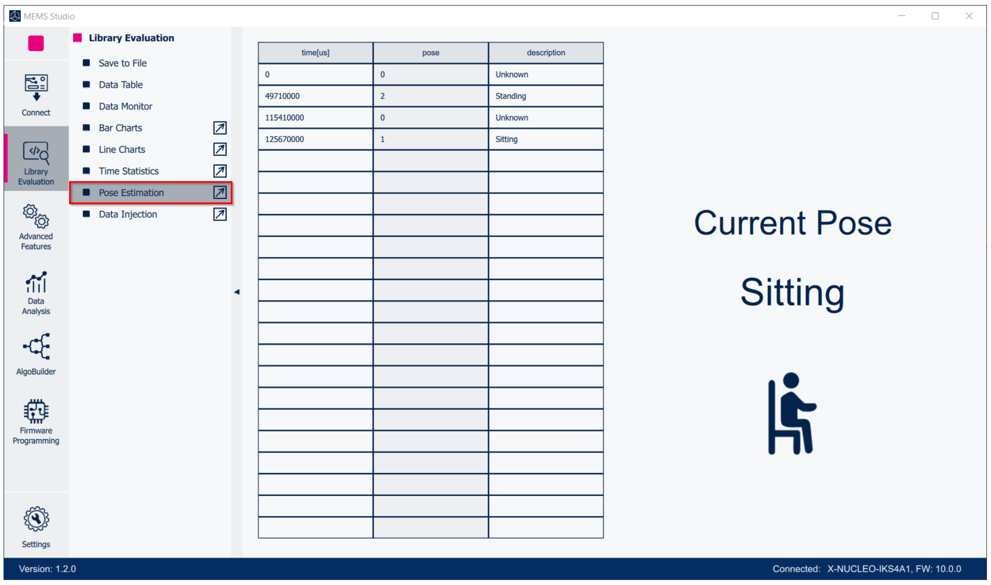
Figure 6. MEMS-Studio - Library Evaluation - Data Table



time[us]	acc_x[mg]	acc_y[mg]	acc_z[mg]	gyro_x[mdps]	gyro_y[mdps]	gyro_z[mdps]	mag_x[mG]
12140000	231	19	981	280	210	140	-315
12200000	231	19	982	280	280	210	-304
12260000	232	19	981	280	140	210	-312
12320000	232	19	981	280	280	210	-312
12380000	232	19	982	350	280	210	-313
12450000	233	19	982	280	140	140	-321
12500000	231	19	981	350	210	210	-312
12570000	232	19	982	350	280	140	-312
12630000	232	19	982	280	70	210	-313
12690000	232	19	982	350	210	140	-318
12750000	231	19	982	350	210	140	-324
12810000	232	19	981	350	140	140	-318
12870000	232	19	981	350	210	210	-312
12930000	232	19	981	280	280	210	-315

Step 4. Click on the [**Pose Estimation**] to open the dedicated page for this library.

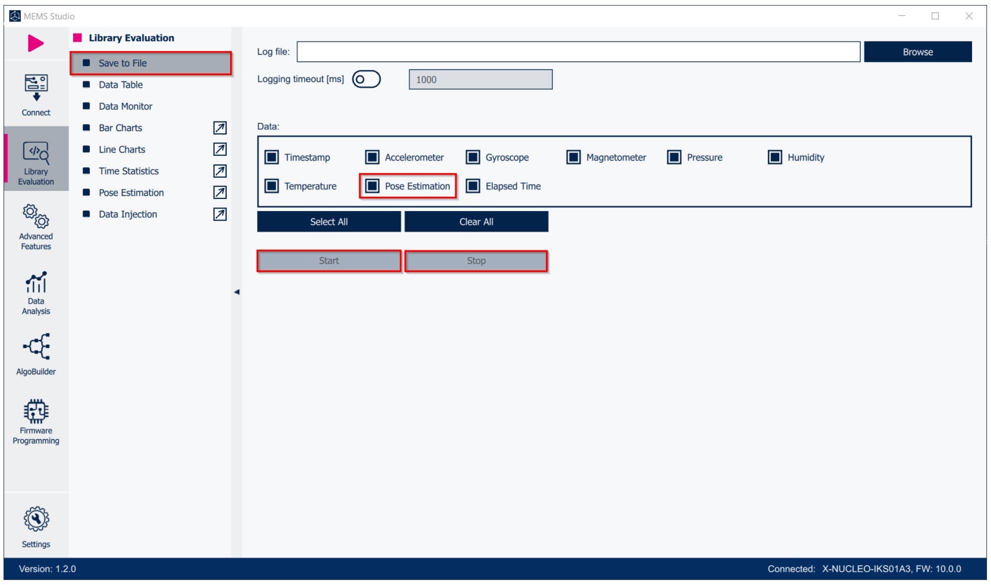
Figure 7. MEMS-Studio - Library Evaluation - Pose Estimation



time[us]	pose	description
0	0	Unknown
49710000	2	Standing
115410000	0	Unknown
125670000	1	Sitting

Step 5. Click on [**Save to File**] to open the dataloging configuration window. Select the sensor and pose estimation 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



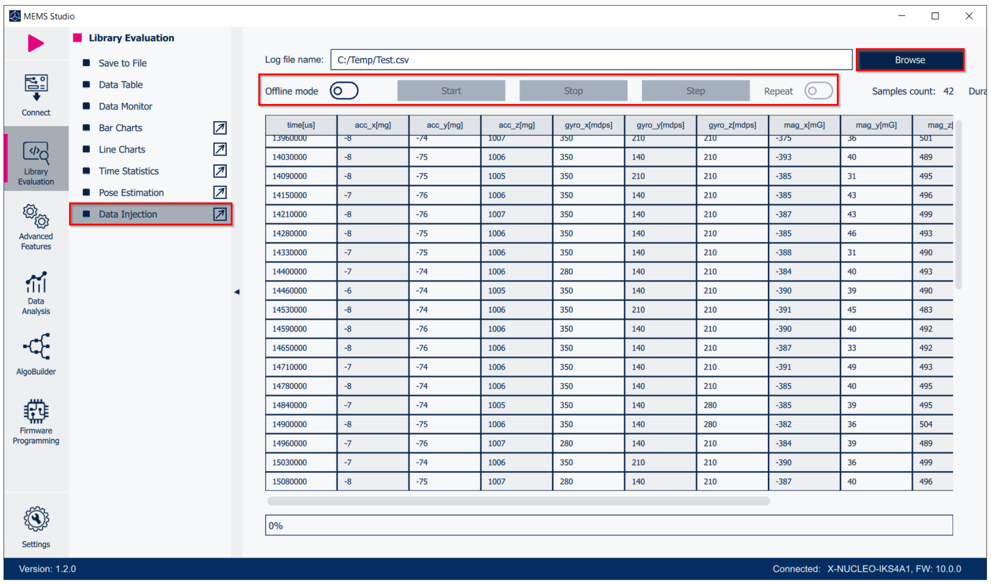
Log file:

Logging timeout [ms] 1000

Data:

Timestamp, Accelerometer, Gyroscope, Magnetometer, Pressure, Humidity, Temperature, Pose Estimation, Elapsed Time

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.



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

Offline mode

Samples count: 42 Dura

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
13960000	-8	-74	1007	350	210	210	-375	36	501
14030000	-8	-75	1006	350	140	210	-393	40	489
14090000	-8	-75	1005	350	210	210	-385	31	495
14150000	-7	-76	1006	350	140	210	-385	43	496
14210000	-8	-76	1007	350	140	210	-387	43	499
14280000	-8	-75	1006	350	140	210	-385	46	493
14330000	-7	-75	1006	350	140	210	-388	31	490
14400000	-7	-74	1006	280	140	210	-384	40	493
14460000	-6	-74	1005	350	140	210	-390	39	490
14530000	-8	-74	1006	350	210	210	-391	45	483
14590000	-8	-76	1006	350	140	210	-390	40	492
14650000	-8	-76	1006	350	140	210	-387	33	492
14710000	-7	-74	1006	350	140	210	-391	49	493
14780000	-8	-74	1006	350	140	210	-385	40	495
14840000	-7	-74	1005	350	140	280	-385	39	495
14900000	-8	-75	1006	350	140	280	-382	36	504
14960000	-7	-76	1007	280	140	210	-384	39	489
15030000	-7	-74	1006	350	210	210	-390	36	499
15080000	-8	-75	1007	280	140	210	-387	40	496

0%

Version: 1.2.0

Connected: X-NUCLEO-IKS4A1, FW: 10.0.0

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 5. **Document revision history**

Date	Version	Changes
18-May-2017	1	Initial release.
06-Feb-2018	2	Added references to NUCLEO-L152RE development board, Figure 2. Orientation system for wrist-worn devices and Table 3. Elapsed time (µs) algorithm.
21-Mar-2018	3	Updated Introduction, Section 2.1 MotionPE overview and Section 2.2.5 Algorithm performance.
21-Feb-2019	4	Updated Figure 1. Example of sensor orientations, Section 2.2.5: Algorithm performance, Figure 4. STM32 Nucleo: LEDs, button, jumper, Figure 5. MEMS-Studio - Connect, Figure 6. MEMS-Studio - Library Evaluation - Data Table, Figure 7. MEMS-Studio - Library Evaluation - Pose Estimation and Figure 8. MEMS-Studio - Library Evaluation - Save to File. Added X-NUCLEO-IKS01A3 expansion board compatibility information.
06-Aug-2024	5	Updated Section Introduction, Section 2.1: MotionPE overview, Section 2.2.1: MotionPE library description, Section 2.2.2: MotionPE APIs, Section 2.2.5: Algorithm performance, Section 2.3: Sample application, Section 2.4: MEMS-Studio application

Contents

List of tables

List of figures