Getting started with MotionAT active time library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionAT middleware library is part of the X-CUBE-MEMS1 software and runs on STM32. It combines results from activity recognition for wrist, motion intensity detection and pedometer for wrist algorithms. It also provides real-time information about the number of active seconds (that is how long the user was active with the wearable device as a smart watch).

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 or ARM® Cortex®-M4 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-IKS01A1 (with optional STEVAL-MKI160V1) or X-NUCLEO-IKS01A2 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 MotionAT middleware library in X-CUBE-MEMS1 software expansion for STM32Cube

2.1 MotionAT overview

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

The library acquires data from the accelerometer and provides information about the number of active seconds (how long the user was active) with the wearable 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-IKS01A2 and X-NUCLEO-IKS01A1 (with optional STEVAL-MKI160V1) expansion boards, mounted on a NUCLEO-F401RE, NUCLEO-L476RG or NUCLEO-L152RE development board.

2.2 MotionAT library

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

2.2.1 MotionAT library description

The MotionAT active time library manages the data acquired from the accelerometer; it features:

- possibility of detecting the number of active seconds
- recognition based on accelerometer data only
- required accelerometer data sampling frequency of 50 Hz
- measurement based on the accelerometer data only
- 16.5 kByte of code memory and 5.6 kByte of data memory usage

  Note: Real size might differ for different IDEs (toolchain)
- available for ARM® Cortex®-M3 and ARM® Cortex®-M4 architectures

2.2.2 MotionAT APIs

The MotionAT library APIs are:

- `uint8_t MotionAT_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 MotionAT_Initialize(void)`
  - performs MotionAT library initialization and setup of the internal mechanism including the dynamic memory allocation
  - 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 active time library.
- `void MotionAT_Update(MAT_input_t *data_in, MAT_output_t *data_out)`
  - executes active time algorithm
  - `*data_in` parameter is a pointer to a structure with input data
  - the parameters for the structure type `MAT_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

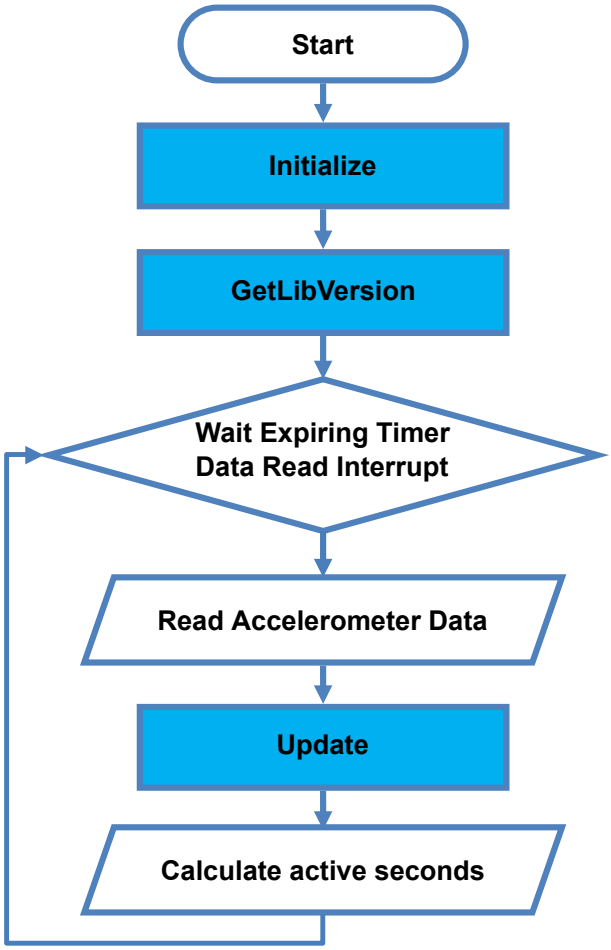
– *data_out parameter is a pointer to a structure with output data

– the parameter for the structure type MAT_output_t is:

- active equal to 1 if the user was active during the last algorithm run, otherwise it is equal to 0

2.2.3 API flow chart

Figure 1. MotionAT API logic sequence



2.2.4 Demo code

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

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

/* Pedometer API initialization function */
MotionAT_manager_init(ACCELERO_handle);

/* OPTIONAL */
/* Get library version */
  MotionAT_manager_get_version(lib_version, &lib_version_len);

[…]
```

```
/*** Using Active Time algorithm ***/
Timer_OR_DataRate_Interrupt_Handler()
{
  MAT_input_t MAT_data_in;
  MAT_output_t MAT_data_out;

  /* Get acceleration X/Y/Z in g */
  MEMS_Read_AccValue(&MAT_data_in.acc_x, &MAT_data_in.acc_y, &MAT_data_in.acc_z);

  /* Run Active Time algorithm */
  MotionAT_manager__run(&MAT_data_in, &MAT_data_out);

  /* Count active seconds */
  /* 1/50Hz = 0.02s per 1 algorithm run */
  active_seconds += MAT_data_out.active * 0.02f;

}
```

2.2.5 Algorithm performance

The active time algorithm only uses data from the accelerometer and runs at a low frequency (50 Hz) to reduce power consumption.

It detects and provides real-time information about the number of active seconds (how long the user was active with his wearable device).

Table 2. Elapsed time (µs) algorithm

Cortex-M4 STM32F401RE at 84 MHz									Cortex-M3 STM32L152RE at 32 MHz								
SW4STM32 1.13.1 (GCC 5.4.1)			IAR EWARM 7.80.4			Keil µVision 5.22			SW4STM32 1.13.1 (GCC 5.4.1)			IAR EWARM 7.80.4			Keil µVision 5.22		
Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
1	178	885	38	70	816	38	146	1877	15	955	10192	284	555	7607	311	894	8235

2.3 Sample application

The MotionAT 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-IKS01A1 (based on LSM6DS0) or an X-NUCLEO-IKS01A2 (based on LSM6DSL) expansion board, with optional STEVAL-MKI160V1 board (based on LSM6DS3).

The application recognizes the active seconds in real-time. The data can be displayed through a GUI or stored in the board for offline analysis.

Stand-alone mode

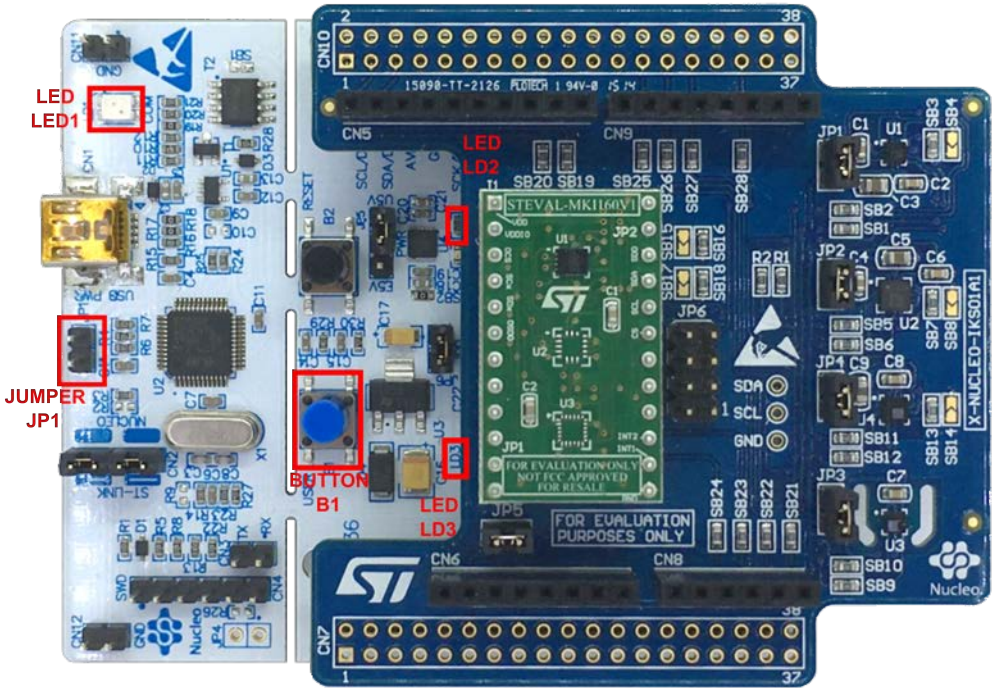
In stand-alone mode, the sample application allows the user to detect active seconds and store them in the MCU flash memory.

The STM32 Nucleo board may be supplied by a portable battery pack (to make the user experience more comfortable, portable and free of any PC connections).

Table 3. Power supply scheme

Power source	JP1 settings	Working mode
USB PC cable	JP1 open	PC GUI driven mode
Battery pack	JP1 closed	Stand-alone mode

Figure 2. **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 and the tricolor LED LD1 (COM) begins blinking slowly due to the missing USB enumeration (refer to UM1724 on www.st.com for further details).

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

When the user button B1 is pressed, the system starts acquiring data from the accelerometer sensor and detects the active seconds; during this acquisition mode, a fast LED LD2 blinking indicates that the algorithm is running. During this phase, the detected steps and cadence are stored in the MCU internal flash memory. Data are automatically saved every 5 minutes to avoid excessive data loss in case of an unforeseen power fault.

Pressing button B1 a second time stops the algorithm and data storage and LED LD2 switches off.

Pressing the button again starts the algorithm and data storage once again.

The flash sector dedicated to data storage is 128 KB, allowing memorization of more than 16,000 data sets.

To retrieve those data, the board has to be connected to a PC, running Unicleo-GUI. When stored data is retrieved via the GUI, the MCU flash sector dedicated to this purpose is cleared.

If LED LD2 is ON after powering the board, it represents a warning message indicating the flash memory is full.

Note: Optionally, the MCU memory can be erased by holding the user push buttondown for at least 5 seconds. LED LD2 switches OFF and then blinks 3 times to indicate that the data stored in the MCU has been erased. This option is available only after power ON or reset of the board while LED LD2 is ON indicating the flash memory is full.

When the application runs in stand-alone mode and the flash memory is full, the application switches to PC GUI drive mode and LED LD2 switches OFF.

The flash memory must be erased by downloading data via the Unicleo-GUI or the user push button (see the above note).

PC GUI drive mode

In this mode, 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 whether the user was active during last single

algorithm run, the amount of active seconds, accelerometer data, time stamp and eventually other sensor data, in real-time, using the Unicleo-GUI.

In this working mode, data are not stored in the MCU flash memory.

2.4 Unicleo-GUI application

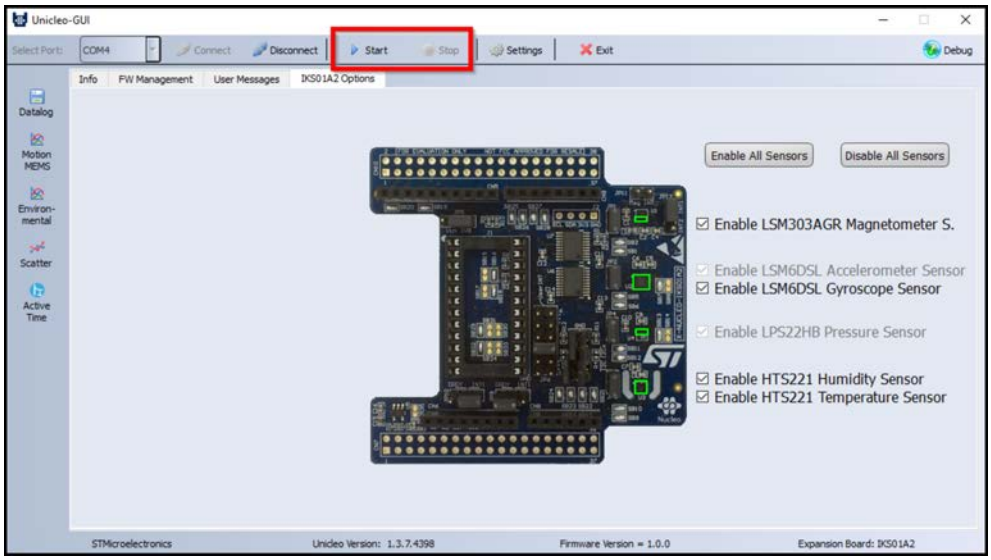
The sample application uses the Windows Unicleo-GUI utility, which can be downloaded from www.st.com.

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.

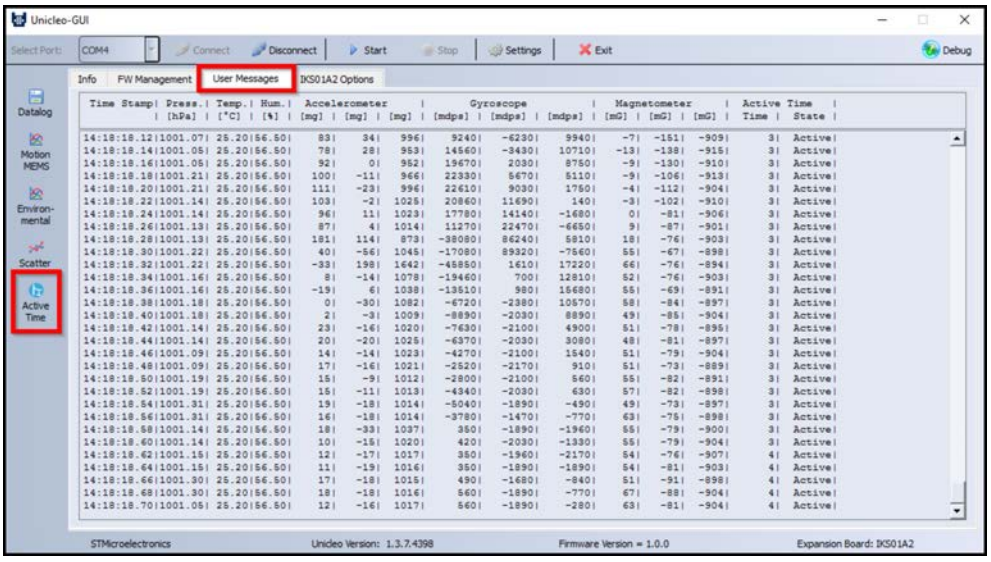
Figure 3. Unicleo main window



Step 3. Start and stop data streaming by using the appropriate buttons on the vertical tool bar.

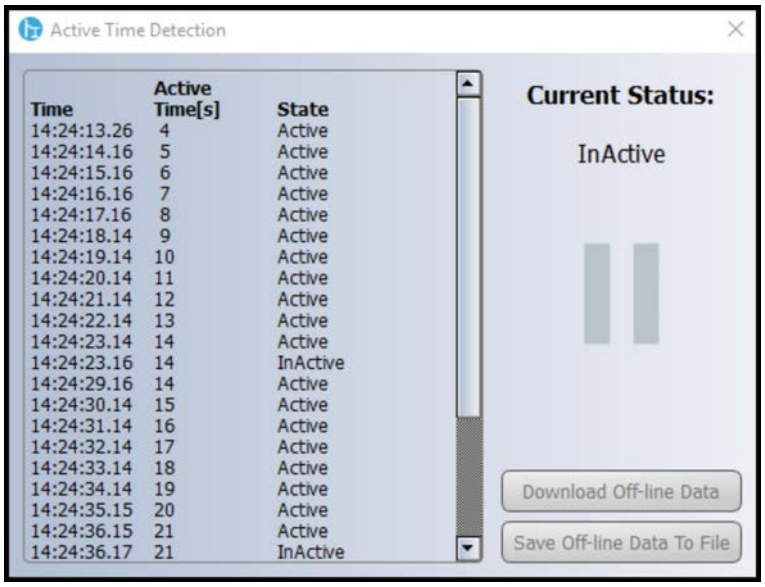
The data coming from the connected sensor can be viewed in the User Messages tab.

Figure 4. User Messages tab



Step 4. Click on the Active Time icon in the vertical tool bar to open the dedicated application window.

Figure 5. Active Time window



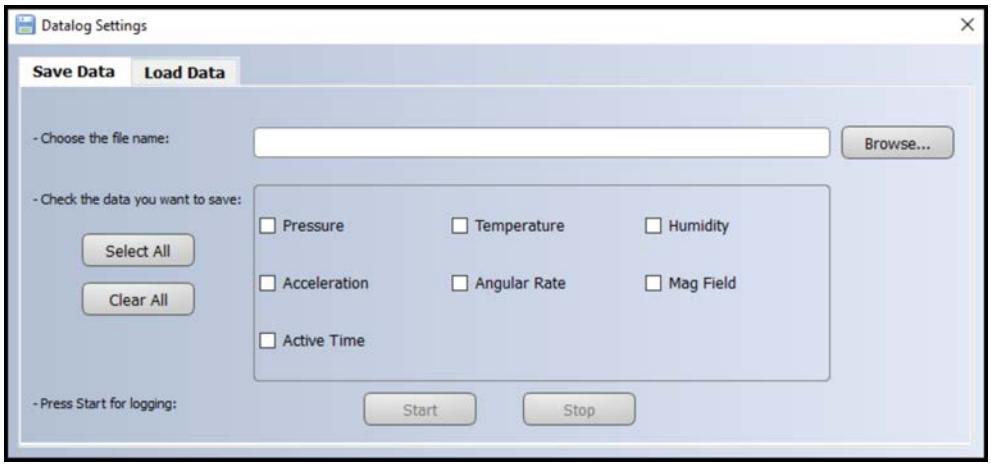
If the board has been working in stand-alone mode and the user wants to retrieve stored data, press **Download Off-line Data** button to upload the stored activities data to the application. This operation automatically deletes acquired data from microcontroller.

Press the **Save Off-line Data to File** button to save the uploaded data in a .tsv file.

Step 5. Click on the Datalog icon in the vertical tool bar to open the datalog configuration window:

you can select which sensor and activity data to save in files. You can start or stop saving by clicking on the corresponding button.

Figure 6. Datalog window



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 board
3. UM2128: Getting started with Unicleo-GUI for motion MEMS and environmental sensor software expansion for STM32Cube

Revision history

Table 4. **Document revision history**

Date	Version	Changes
31-Jan-2018	1	Initial release.
21-Mar-2018	2	Updated Section ● Introduction and Section 2.1 MotionAT overview.

Contents

List of tables

List of figures

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.

© 2018 STMicroelectronics – All rights reserved