Getting started with MotionAR activity recognition library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionAR is a middleware library part of X-CUBE-MEMS1 software and runs on STM32. It provides real-time information on the type of activity performed by the user. It is able to distinguish the following activities: stationary, walking, fast walking, jogging, biking, driving.

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 Cortex®-M4 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 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 MotionAR middleware library in X-CUBE-MEMS1 software expansion for STM32Cube

2.1 MotionAR overview

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

The library acquires data from the accelerometer and provides information on the type of activity performed by the 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.

Sample implementation is available on 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 MotionAR library

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

2.2.1 MotionAR library description

The MotionAR activity recognition library manages data acquired from accelerometer; it features:

- possibility to distinguish the following activities: stationary, walking, fast walking, jogging, biking, driving
- recognition based on accelerometer data only
- required accelerometer data sampling frequency: 16 Hz
- occupies 7.2 kByte of code memory and 1.6 kByte of data memory

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

2.2.2 MotionAR APIs

The MotionAR APIs are:

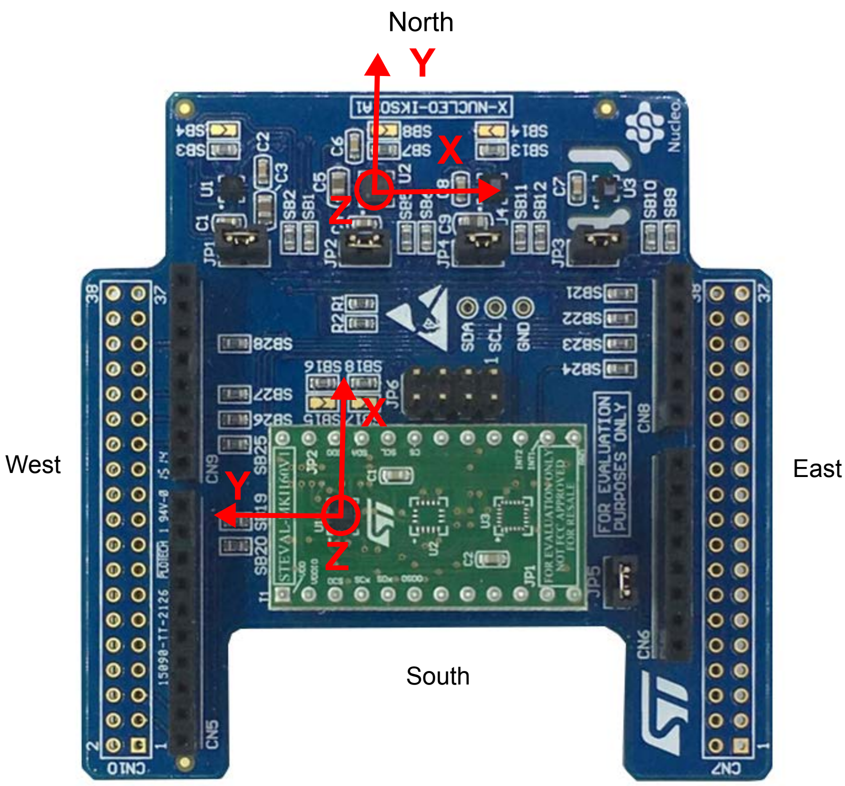
- `uint8_t MotionAR_GetLibVersion(char *version)`
  - retrieves the version of the library
  - `*version` is a pointer to an array of 35 characters
  - returns the number of characters in the version string
- `void MotionAR_Initialize(void)`
  - performs MotionAR library initialization and setup of the internal mechanism
  - 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 accelerometer calibration library.
- `void MotionAR_Reset(void)`
  - resets activity recognition algorithms
- `void MotionAR_Update (MAC_input_t *data_in, MAR_output_t *data_out)`
  - executes activity recognition algorithm
  - `*data_in` parameter is a pointer to a structure with input data
  - the parameters for the structure type `MAR_input_t` are:
    - `AccX` is accelerometer sensor value in X axis in g

        - `AccY` is accelerometer sensor value in Y axis in g
        - `AccZ` is accelerometer sensor value in Z axis in g
    - `*data_out` parameter is a pointer to enum with the following items:
        - `MAR_NOACTIVITY = 0`
        - `MAR_STATIONARY = 1`
        - `MAR_WALKING = 2`
        - `MAR_FASTWALKING = 3`
        - `MAR_JOGGING = 4`
        - `MAR_BIKING = 5`
        - `MAR_DRIVING = 6`

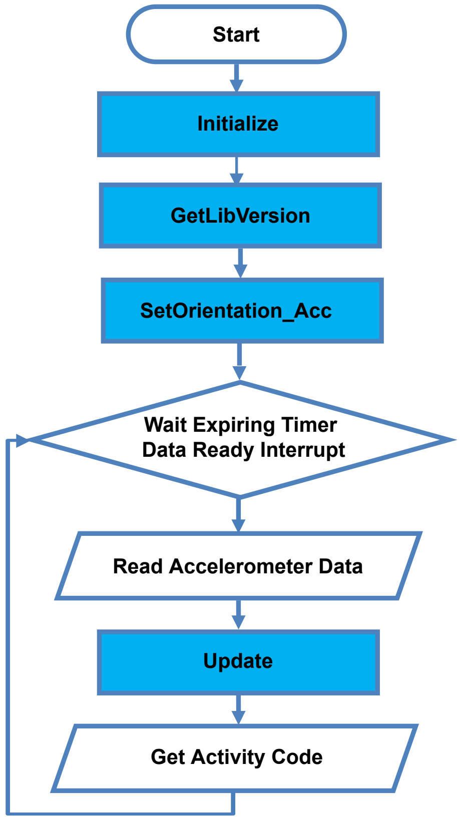
- `void MotionAR_SetOrientation_Acc(const char *acc_orientation)`
    - sets the accelerometer data orientation
    - configuration is usually performed immediately after the `MotionAR_Initialize` function call
    - `*acc_orientation` parameter is a pointer to a string of three characters indicating 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-IKS01A1 accelerometer sensor has an ENU orientation (x - East, y - North, z - Up), so the string is: “enu”, while the accelerometer sensor in STEVAL-MKI160V1 is NWU (x-North, y-West, z-Up): “nwu”.

Figure 1. Sensor orientation example



2.2.3 API flow chart

Figure 2. MotionAR API logic sequence



2.2.4 Demo code

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

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

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

/* Activity recognition API initialization function */
MotionAR_Initialize();

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

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

```
[…]

/*** Using activity recognition algorithm ***/
Timer_OR_DataRate_Interrupt_Handler()
{
MAR_input_t data_in;
MAR_output_t activity;

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

/* Activity recognition algorithm update */
MotionAR_Update(&data_in, &activity);
}
```

2.2.5 Algorithm performance

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

Table 2. Algorithm performance

Activity	Detection probability (typical)(1)	Best performance	Susceptible	Carry positions
Stationary	92.27%		Holding in hand and heavy texting	All: trouser pocket, shirt pocket, back pocket, near the head, etc.
Walking	99.44%	Step rate ≥ 1.4 step/s	Step rate ≤ 1.2 step/s	all
Fast walking	95.94%	Step rate ≥ 2.0 step/s		All
Jogging	98.49%	Step rate ≥ 2.2 step/s	Duration < 1 minute; speed < 8 Km/h	Trouser pocket, arm swing, in-hand
Biking	91.93%	Outdoor speed ≥11 Km/h	Passenger seat, glove compartment	Backpack, shirt pocket, trouser pocket
Driving	78.65%	Speed ≥ 48 Km/h	Passenger seat, glove compartment	Cup holder, dash board, shirt pocket, trouser pocket

1. *Typical specifications are not guaranteed*

Table 3. 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
5	43	801	5	44	547	12	90	1374	113	574	7923	120	435	5462	118	694	8991

3 Sample application

The MotionAR 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 performed activities in real-time. Data can be displayed through a GUI or stored in the board for offline analysis. The algorithm recognizes stationary, walking, fast walking, jogging, bike riding and driving activities.

Stand-alone mode

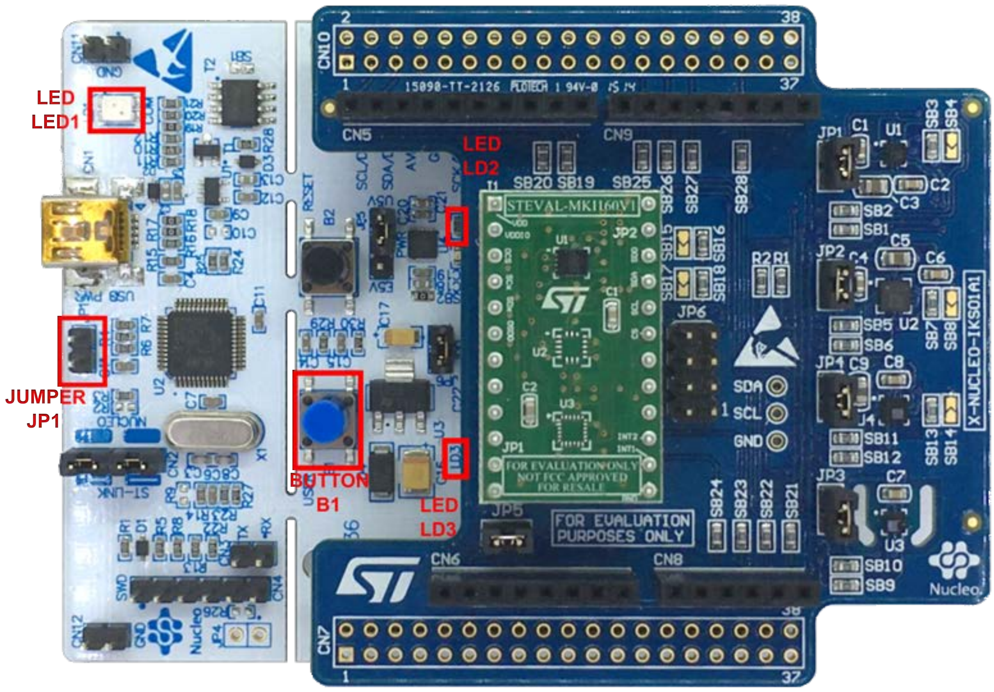
In stand-alone mode, the sample application allows the user to detect performed gesture and store it in the MCU flash memory.

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 4. **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 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 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).

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 performed activity; during this acquisition mode, a fast LED LD2 blinking indicates that the algorithm is running. During this phase, the detected device gesture is stored in the MCU internal flash memory.

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, theapplication 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 theuser 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 the activity detected, 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.

4 Unicleo-GUI application

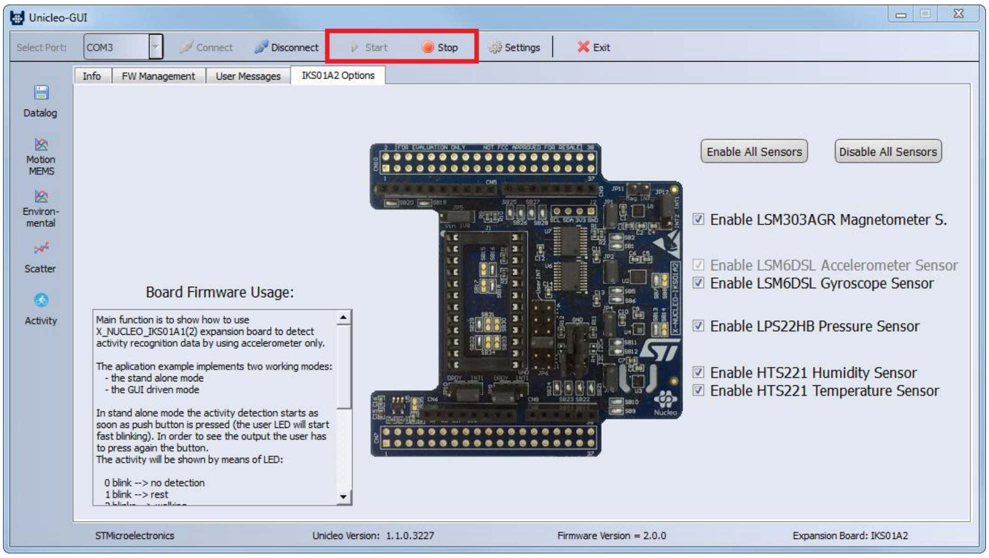
The sample application uses the Windows Unicleo-GUI utility, which can be downloaded from www.st.com.

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.

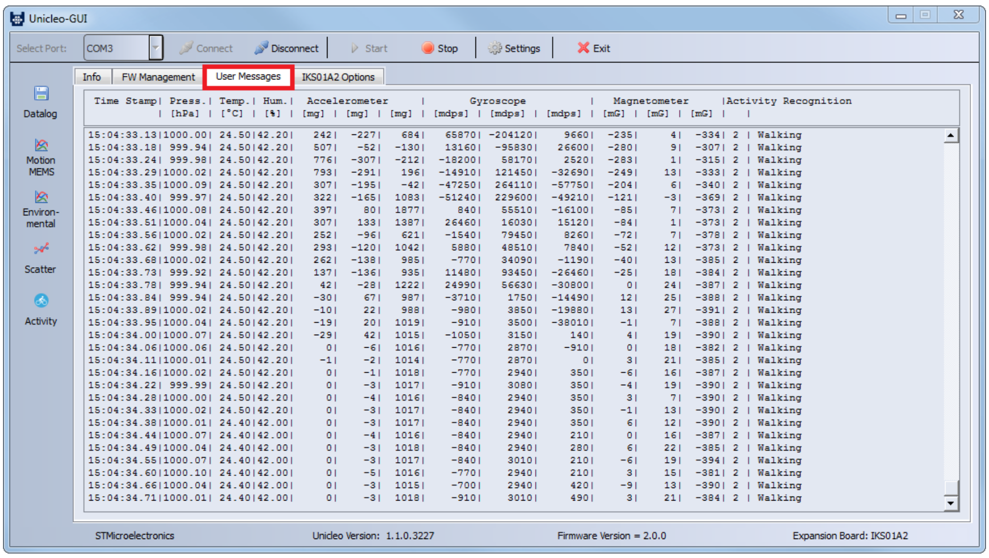
Figure 4. Unicleo main window



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 5. User Messages tab



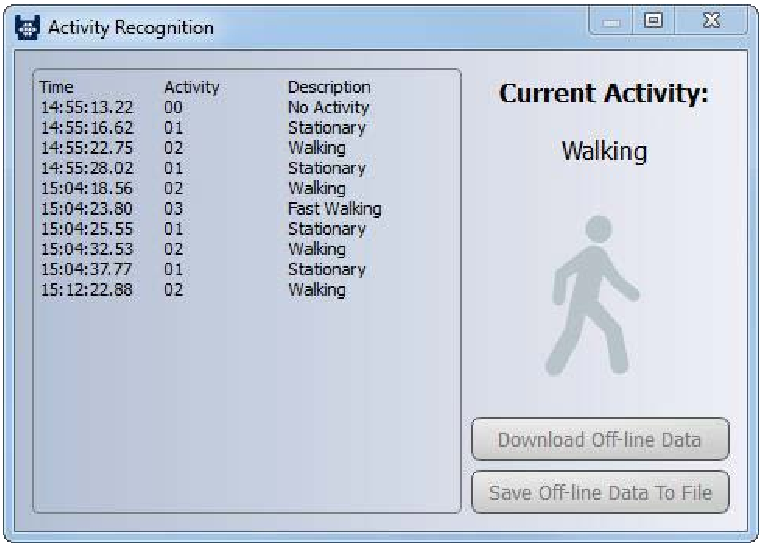
Step 4. Click on the Activity icon in the vertical tool bar to open the dedicated application window.

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.

Note: Activities with a duration of less than 20 seconds are not memorized.

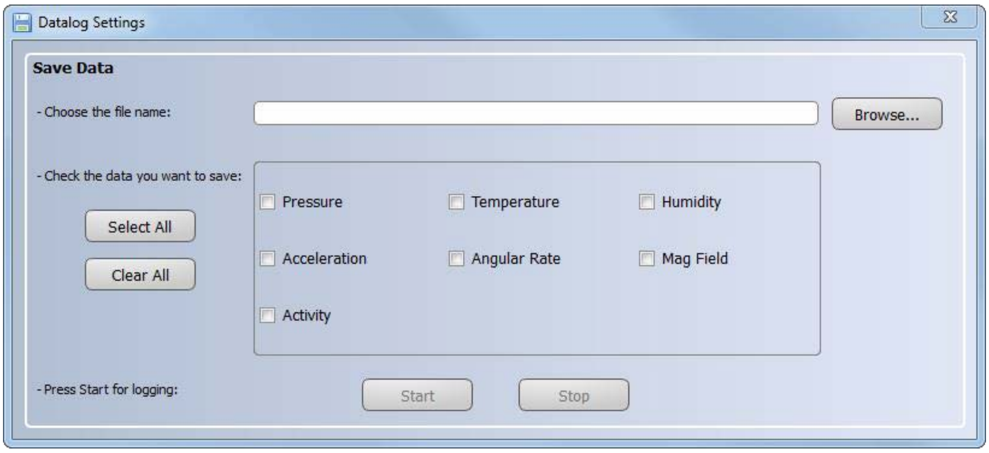
Press the Save Off-line Data to File button to save the uploaded data in a .tsv file.

Figure 6. Activity recognition window



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 7. Datalog window



5 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 5. **Document revision history**

Date	Version	Changes
10-Apr-2017	1	Initial release.
26-Jan-2018	2	Updated Section 3 Sample application. Added references to NUCLEO-L152RE development board and Table 3. Elapsed time (µs) algorithm.
19-Mar-2018	3	Updated Section • Introduction, Section 2.1 MotionAR overview and Section 2.2.5 Algorithm performance.

Contents

List of tables

List of figures