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 an X-NUCLEO-IKS01A2 or X-NUCLEO-IKS01A3 expansion board on a NUCLEO-F401RE, NUCLEO-L476RG or NUCLEO-L152RE development board.
## 1 Acronyms and abbreviations

**Table 1. List of acronyms**

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application programming interface</td>
</tr>
<tr>
<td>BSP</td>
<td>Board support package</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical user interface</td>
</tr>
<tr>
<td>HAL</td>
<td>Hardware abstraction layer</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated development environment</td>
</tr>
</tbody>
</table>
2 MotionAR middleware library in X-CUBE-MEMS1 software expansion

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-IKS01A3 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
- resources requirements:
  - Cortex-M3: 8.3 kB of code and 1.4 kB of data memory
  - Cortex-M4: 7.8 kB of code and 1.4 kB of data memory
- 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, long int timestamp)**
  - 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
- *timestamp* is a relative time for actual sample in ms

- 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-IKS01A2 accelerometer sensor has an NWU orientation (x-North, y-West, z-Up), so the string is: "nwu".

*Figure 1. Sensor orientation example*
2.2.3 API flow chart

![MotionAR API logic sequence]

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.

```c
#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);
```

### 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

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

1. Typical specifications are not guaranteed

#### Table 3. Elapsed time (µs) algorithm

<table>
<thead>
<tr>
<th></th>
<th>Cortex-M4 STM32F401RE at 84 MHz</th>
<th>Cortex-M3 STM32L152RE at 32 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SW4STM32 1.13.1 (GCC 5.4.1)</td>
<td>IAR EWARM 7.80.4</td>
</tr>
<tr>
<td></td>
<td>Keil µVision 5.22</td>
<td>SW4STM32 1.13.1 (GCC 5.4.1)</td>
</tr>
<tr>
<td></td>
<td>Keil µVision 5.22</td>
<td>IAR EWARM 7.80.4</td>
</tr>
<tr>
<td>Min</td>
<td>Min</td>
<td>Min</td>
</tr>
<tr>
<td>Avg</td>
<td>Avg</td>
<td>Avg</td>
</tr>
<tr>
<td>Max</td>
<td>Max</td>
<td>Max</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>14</td>
<td>19</td>
</tr>
<tr>
<td>158</td>
<td>149</td>
<td>229</td>
</tr>
<tr>
<td>117</td>
<td>282</td>
<td>101</td>
</tr>
<tr>
<td>292</td>
<td>3547</td>
<td>5092</td>
</tr>
</tbody>
</table>
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-IKS01A2 or X-NUCLEO-IKS01A3 expansion board.

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>
<thead>
<tr>
<th>Power source</th>
<th>JP1 settings</th>
<th>Working mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>USB PC cable</td>
<td>JP1 open</td>
<td>PC GUI driven mode</td>
</tr>
<tr>
<td>Battery pack</td>
<td>JP1 closed</td>
<td>Stand-alone mode</td>
</tr>
</tbody>
</table>

**Table 4. Power supply scheme**

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 button down 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 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.

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.
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**

![Activity recognition window](image)

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**

![Datalog window](image)
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
### Table 5. Document revision history

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-Apr-2017</td>
<td>1</td>
<td>Initial release.</td>
</tr>
<tr>
<td>26-Jan-2018</td>
<td>2</td>
<td>Updated Section 3 Sample application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added references to NUCLEO-L152RE development board and Table 3. Elapsed time (μs) algorithm.</td>
</tr>
<tr>
<td>19-Mar-2018</td>
<td>3</td>
<td>Updated Introduction, Section 2.1 MotionAR overview and Section 2.2.5 Algorithm performance.</td>
</tr>
<tr>
<td>14-Feb-2019</td>
<td>4</td>
<td>Updated Figure 1. Sensor orientation example, Table 3. Elapsed time (μs) algorithm and Figure 3. STM32 Nucleo: LEDs, button, jumper.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Added X-NUCLEO-IKS01A3 expansion board compatibility information.</td>
</tr>
<tr>
<td>20-Mar-2019</td>
<td>5</td>
<td>Updated Section 2.2.2 MotionAR APIs, Figure 4. Unicleo main window, Figure 5. User Messages tab, Figure 6. Activity recognition window and Figure 7. Datalog window.</td>
</tr>
</tbody>
</table>
Contents

1 Acronyms and abbreviations ........................................................................................................2

2 MotionAR middleware library in X-CUBE-MEMS1 software expansion ........................................3
   2.1 MotionAR overview ........................................................................................................3
   2.2 MotionAR library ...........................................................................................................3
      2.2.1 MotionAR library description .................................................................................3
      2.2.2 MotionAR APIs ......................................................................................................3
      2.2.3 API flow chart ........................................................................................................4
      2.2.4 Demo code ................................................................................................................5
      2.2.5 Algorithm performance .........................................................................................6

3 Sample application ..................................................................................................................7

4 Unicleo-GUI application ..........................................................................................................9

5 References ................................................................................................................................11

Revision history ..........................................................................................................................12
List of tables

Table 1. List of acronyms ................................................................. 2
Table 2. Algorithm performance ...................................................... 6
Table 3. Elapsed time (µs) algorithm ........................................... 6
Table 4. Power supply scheme ....................................................... 7
Table 5. Document revision history .............................................. 12
List of figures

Figure 1. Sensor orientation example ........................................................... 4
Figure 2. MotionAR API logic sequence ......................................................... 5
Figure 3. STM32 Nucleo: LEDs, button, jumper ................................................ 7
Figure 4. Unicreo main window ................................................................ 9
Figure 5. User Messages tab .................................................................... 9
Figure 6. Activity recognition window. ........................................................ 10
Figure 7. Datalog window ...................................................................... 10