UM2194

User manual

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

Introduction

The MotionAW 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, standing, sitting, lying, walking, fast walking, jogging, biking. 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 that eases portability across different STM32 microcontrollers.

The software comes with sample implementation running on an X-NUCLEO-IKS01A3 or X-NUCLEO-IKS4A1 expansion board on a NUCLEO-F401RE, NUCLEO-L152RE or NUCLEO-U575ZI-Q 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 MotionAW middleware library in X-CUBE-MEMS1 software expansion

2.1 MotionAW overview

The MotionAW 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.

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

2.2 MotionAW library

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

2.2.1 MotionAW library description

The MotionAW real-time activity recognition library manages the data acquired from the accelerometer; it:

- can distinguish the following activities: stationary, standing, sitting, lying, walking, fast walking, jogging, biking
- is intended for wrist-worn devices
- is based only on accelerometer data
- requires accelerometer data sampling frequency of 16 Hz
- resources requirements:
  - Cortex-M3: 13.7 kB of code and 3.1 kB of data memory
  - Cortex-M33: 13.7 kB of code and 3.1 kB of data memory
  - Cortex-M4: 13.9 kB of code and 3.1 kB of data memory
  - Cortex-M7: 13.9 kB of code and 3.1 kB of data memory
- available for ARM Cortex-M3, Cortex-M33, Cortex-M4 and Cortex-M7 architectures.

2.2.2 MotionAW APIs

The MotionAW library APIs are:

- `uint8_t MotionAW_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 MotionAW_Initialize(void)`
  - performs MotionAW 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 MotionAW_Reset(void)`
  - resets activity recognition algorithms

- `void MotionAW_Update(MAW_input_t *data_in, MAW_output_t *data_out, int64_t t)`
  - executes activity recognition for wrist algorithm
  - `*data_in` parameter is a pointer to a structure with input data
  - the parameters for the structure type `MAW_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 parameters for the structure type `MAW_output_t` are:
    - `CurrectActivity` contains the following items
      - `MAW_NOACTIVITY = 0`
      - `MAW_STATIONARY = 1`
      - `MAW_STANDING = 2`
      - `MAW_SITTING = 3`
      - `MAW_LYING = 4`
      - `MAW_WALKING = 5`
      - `MAW_FASTWALKING = 6`
      - `MAW_JOGGING = 7`
      - `MAW_BIKING = 8`
    - `Confidence` is the confidence of the current activity from 0 to 5
    - `CurrentActivityDuration` is the duration in seconds for the current ongoing activity since the last activity change
    - `ActivityTotalDuration` is an array of the total duration in seconds for each activity since the last reset
    - t is a relative time for actual sample in ms
- `void MotionAW_Reset_Activity_Duration(void)`
  - resets total activity duration counters
- `void MotionAW_SetOrientation_Acc (const char *acc_orientation)`
  - this function is used to set the accelerometer data orientation
  - configuration is usually performed immediately after the `MotionAW_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

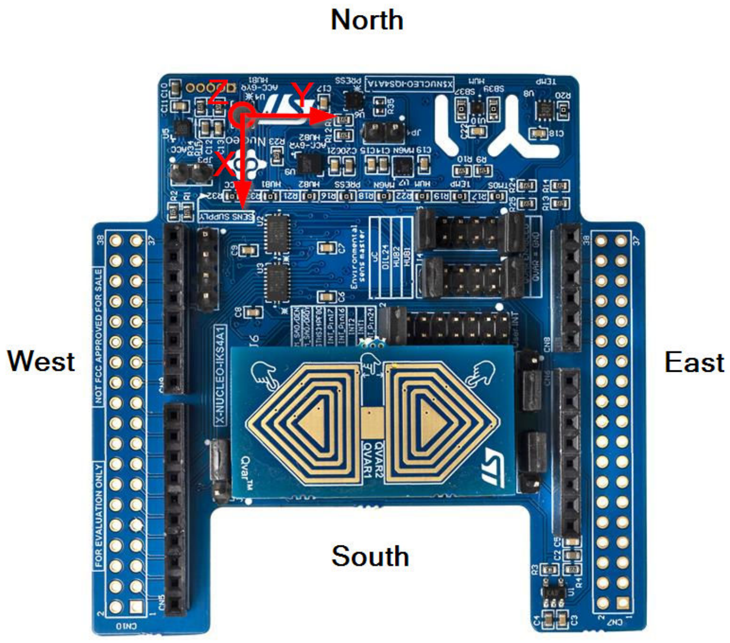
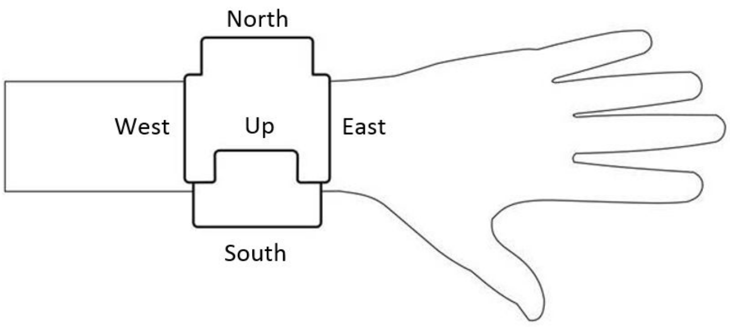
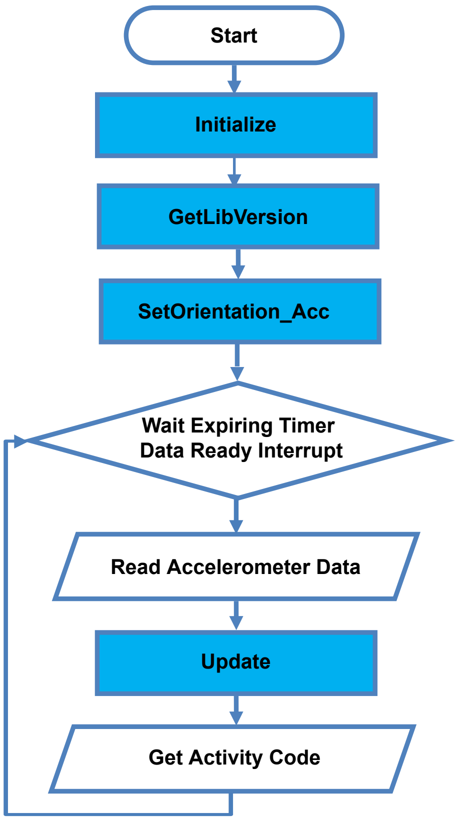


Figure 2. Orientation system for wrist-worn devices



2.2.3 API flow chart

Figure 3. MotionAW 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[] = "seu";

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

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

/* Set accelerometer orientation */
MotionAW_SetOrientation_Acc(acc_orientation);

[…]

/*** Using activity recognition algorithm ***/
Timer_OR_DataRate_Interrupt_Handler()
{
MAW_input_t data_in;
MAW_output_t activity;

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

/* Get current time in ms */
TIMER_Get_TimeValue(&timestamp_ms);

/* Activity recognition algorithm update */
MotionAW_Update(data_in, data_out, timestamp_ms);
}
```

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)	Minimum Latency	Typical Latency
Stationary	98.48%	3 s	5 s
• Sitting	83.30%	15 s	30 s
• Standing	75.05%	15 s	30 s
• Lying	89.73%	3 min	-
Walking	93.22%	5 s	7 s
Fast walking	85.92%	5 s	7 s
Jogging	98.30%	3 s	6 s
Biking	89.35%	10 s	20 s

1. *Typical specifications are not guaranteed*

Table 3. Cortex-M4 and Cortex-M3: elapsed time (µs) algorithm

Cortex-M4 STM32F401RE at 84 MHz			Cortex-M3 STM32L152RE at 32 MHz		
Min	Avg	Max	Min	Avg	Max
10	17	248	157	370	6559

Table 4. Cortex-M33 and Cortex-M7: elapsed time (µs) algorithm

Cortex-M33 STM32U575ZI-Q at 160 MHz			Cortex-M7 STM32F767ZI at 96 MHz		
Min	Avg	Max	Min	Avg	Max
5	9	124	15	23	281

2.2.6 Sample application

The MotionAW 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-L152RE or NUCLEO-U575ZI-Q development board connected to an X-NUCLEO-IKS01A3 or X-NUCLEO-IKS4A1 expansion board.

The application recognizes performed activities in real-time. Data can be displayed through a GUI. The algorithm recognizes stationary, walking, fast walking, jogging, and bike riding activities.

USB cable connection is required to monitor real-time data. The board is powered by the PC via the USB connection. This allows the user to display the activity detected, accelerometer data, time stamp and eventually other sensor data, in real-time, using the MEMS-Studio GUI application.

2.2.7 MEMS-Studio application

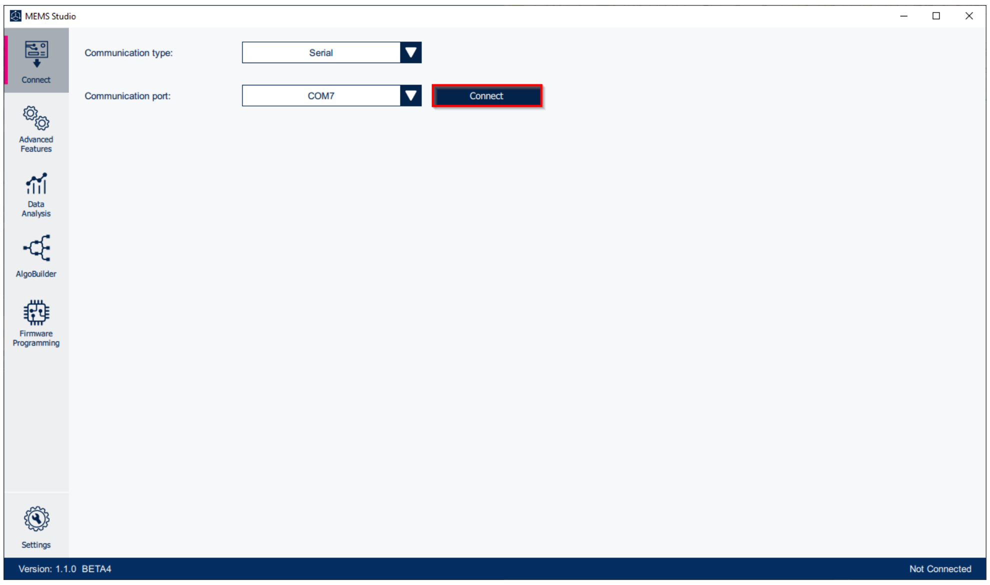
The sample application uses the MEMS-Studio GUI application, which can be downloaded from www.st.com.

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 the appropriate COM port. Press **Connect** button to open this port.

Figure 4. MEMS-Studio - Connect



Communication type: Serial

Communication port: COM7

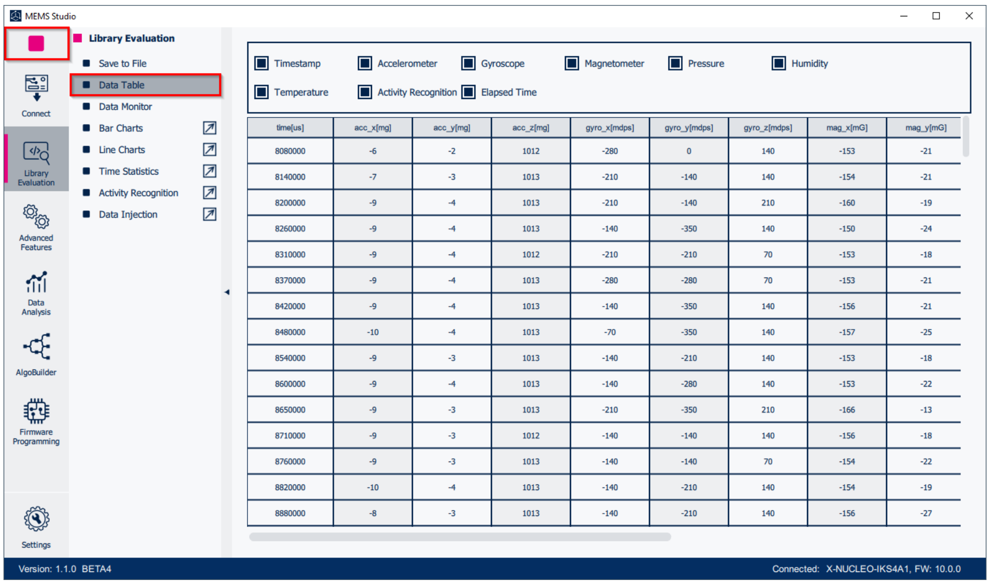
Version: 1.1.0 BETA4

Not Connected

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

To start and stop data streaming toggle the appropriate start / 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 5. MEMS-Studio - Library Evaluation - Data Table



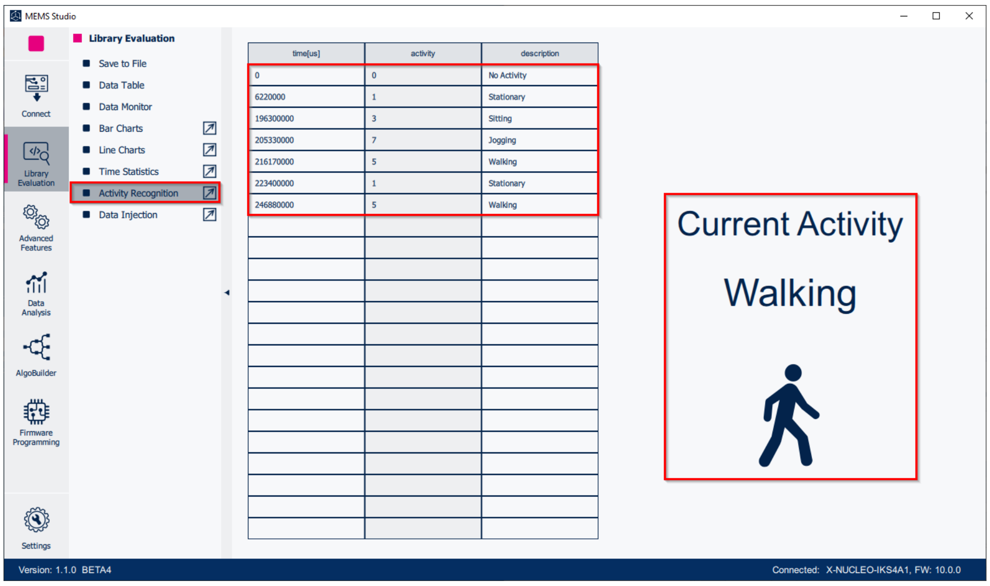
time[us]	acc_x[mg]	acc_y[mg]	acc_z[mg]	gyro_x[mdps]	gyro_y[mdps]	gyro_z[mdps]	mag_x[mG]	mag_y[mG]
8080000	-6	-2	1012	-280	0	140	-153	-21
8140000	-7	-3	1013	-210	-140	140	-154	-21
8200000	-9	-4	1013	-210	-140	210	-160	-19
8260000	-9	-4	1013	-140	-350	140	-150	-24
8310000	-9	-4	1012	-210	-210	70	-153	-18
8370000	-9	-4	1013	-280	-280	70	-153	-21
8420000	-9	-4	1013	-140	-350	140	-156	-21
8480000	-10	-4	1013	-70	-350	140	-157	-25
8540000	-9	-3	1013	-140	-210	140	-153	-18
8600000	-9	-4	1013	-140	-280	140	-153	-22
8650000	-9	-3	1013	-210	-350	210	-166	-13
8710000	-9	-3	1012	-140	-140	140	-156	-18
8760000	-9	-3	1013	-140	-140	70	-154	-22
8820000	-10	-4	1013	-140	-210	140	-154	-19
8880000	-8	-3	1013	-140	-210	140	-156	-27

Version: 1.1.0 BETA4

Connected: X-NUCLEO-IKS4A1, FW: 10.0.0

Step 4. Select the *Activity Recognition* tab on the inner vertical tool bar to open the dedicated application status view.

Figure 6. MEMS-Studio - Library Evaluation - Activity Recognition



time[us]	activity	description
0	0	No Activity
6220000	1	Stationary
196300000	3	Sitting
205330000	7	Jogging
216170000	5	Walking
223400000	1	Stationary
246880000	5	Walking

Current Activity

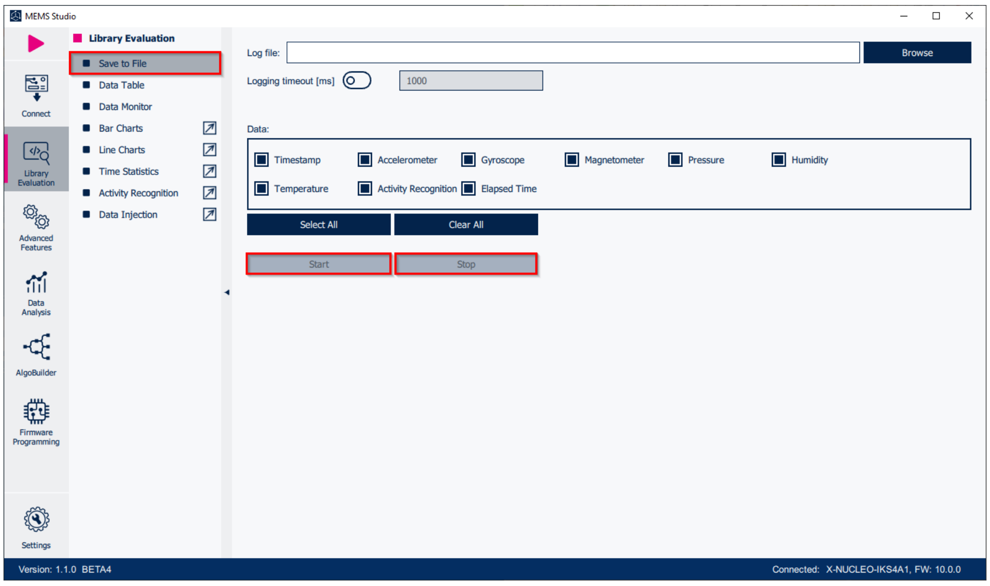
Walking

Version: 1.1.0 BETA4

Connected: X-NUCLEO-IKS4A1, FW: 10.0.0

Step 5. Select the *Save to File* tab on the inner vertical tool bar to open the data logging configuration window. Select which sensor and activity data to save to log file. You can start or stop saving by clicking on the corresponding Start / Stop button.

Figure 7. MEMS-Studio - Library Evaluation - Save to File



Log file:

Logging timeout [ms] 1000

Data:

Timestamp, Accelerometer, Gyroscope, Magnetometer, Pressure, Humidity, Temperature, Activity Recognition, Elapsed Time

Version: 1.1.0 BETA4

Connected: X-NUCLEO-IKS4A1, FW: 10.0.0

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
10-Apr-2017	1	Initial release.
26-Jan-2018	2	Updated Section 2.3 Sample application. Added references to NUCLEO-L152RE development board, Figure 2. Orientation system for wrist-worn devices and Table 3. Elapsed time (µs) algorithm.
20-Mar-2018	3	Updated Introduction, Section 2.1 MotionAW overview and Section 2.2.5 Algorithm performance.
14-Feb-2019	4	Updated *Figure 1. Example of sensor orientations, Table 3. Elapsed time (µs) algorithm* and *Figure 4. STM32 Nucleo: LEDs, button, jumper.* Added X-NUCLEO-IKS01A3 expansion board compatibility information.
16-Apr-2024	5	Updated Section Introduction, Section 2.1: MotionAW overview, Section 2.2.1: MotionAW library description, Section 2.2.2: MotionAW APIs, Section 2.2.4: Demo code, Section 2.2.4: Demo code and Section 2.2.7: MEMS-Studio application.

Contents

1 **Acronyms and abbreviations** 2
2 **MotionAW middleware library in X-CUBE-MEMS1 software expansion** 3
2.1 MotionAW overview 3
2.2 MotionAW library 3
2.2.1 MotionAW library description 3
2.2.2 MotionAW APIs 3
2.2.3 API flow chart 6
2.2.4 Demo code 6
2.2.5 Algorithm performance 7
2.2.6 Sample application 7
2.2.7 MEMS-Studio application 8
3 **References** 10
Revision history 11

List of tables

Table 1. List of acronyms . . . 2
Table 2. Algorithm performance . . . 7
Table 3. Cortex-M4 and Cortex-M3: elapsed time (µs) algorithm . . . 7
Table 4. Cortex-M33 and Cortex-M7: elapsed time (µs) algorithm . . . 7
Table 5. Document revision history . . . 11

List of figures

Figure 1. Example of sensor orientations 5
Figure 2. Orientation system for wrist-worn devices 5
Figure 3. MotionAW API logic sequence 6
Figure 4. MEMS-Studio - Connect 8
Figure 5. MEMS-Studio - Library Evaluation - Data Table 8
Figure 6. MEMS-Studio - Library Evaluation - Activity Recognition 9
Figure 7. MEMS-Studio - Library Evaluation - Save to File 9