Getting started with MotionPW real-time pedometer for wrist library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionPW middleware library is part of the X-CUBE-MEMS1 software and runs on STM32 Nucleo. It provides real-time information about the number of steps and cadence which the user just performed with the wearable device (e.g. 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, ARM Cortex®-M33, ARM® Cortex®-M4, 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-IKS4A1 or X-NUCLEO-IKS01A3 expansion 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 MotionPW middleware library in X-CUBE-MEMS1 software expansion for STM32Cube

2.1 MotionPW overview

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

The library acquires data from the accelerometer and provides information about the number of steps and cadence the user just performed 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-IKS4A1 and X-NUCLEO-IKS01A3 expansion boards, mounted on aNUCLEO-F401RE, NUCLEO-U575ZI-Q or NUCLEO-L152RE development board.

2.2 MotionPW library

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

2.2.1 MotionPW library description

The MotionPW pedometer library manages the data acquired from the accelerometer; it features:

- possibility of detecting the number of steps, cadence and confidence
- recognition based on accelerometer data only
- required accelerometer data sampling frequency of 50 Hz
- resources requirements:
  - Cortex-M3: 3.7 kB of code and 1.8 kB of data memory
  - Cortex-M33: 3.5 kB of code and 1.8 kB of data memory
  - Cortex-M4: 3.5 kB of code and 1.8 kB of data memory
  - Cortex-M7: 3.6 kB of code and 1.8 kB of data memory
- available for ARM® Cortex®-M3, ARM® Cortex®-M33, ARM® Cortex®-M4 and ARM® Cortex®-M7 architectures

2.2.2 MotionPW APIs

The MotionPW library APIs are:

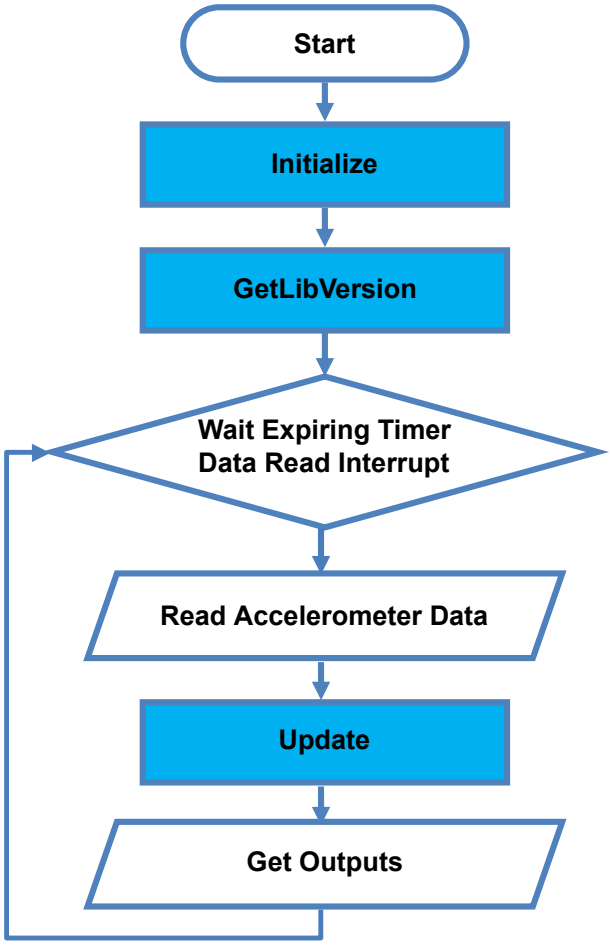
- `uint8_t MotionPW_GetLibVersion(char *version)`
  - retrieves the library version
  - `*version` is a pointer to an array of 35 characters
  - returns the number of characters in the version string
- `void MotionPW_Initialize(void)`
  - performs MotionPW library initialization and setup of the internal mechanism including the dynamic memory allocation

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

- `void MotionPW_Update(MPW_input_t *data_in, MPW_output_t *data_out)`
  - executes pedometer for wrist algorithm
  - `*data_in` parameter is a pointer to a structure with input data
  - the parameters for the structure type `MPW_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
  - `CurrentActivity` is the enumerated input type `MPW_activity_t` with the following values:
    - MPW_UNKNOWN_ACTIVITY = 0x00
    - MPW_WALKING = 0x01
    - MPW_FASTWALKING = 0x02
    - MPW_JOGGING = 0x03
  - `*data_out` parameter is a pointer to a structure with output data
  - the parameters for the structure type `MPW_output_t` are:
    - `Nsteps` is number of steps performed by user
    - `Cadence` is the cadence of user steps
    - `Confidence` is the confidence of calculated output parameter
- `void MotionPW_ResetPedometerLibrary(void)`
  - resets the library internal variables and mechanism into default values (including current step count)
- `void MotionPW_ResetStepCount(void)`
  - resets the current step count
- `void MotionPW_UpdateEnergyThreshold(float *energy_threshold)`
  - updated energy threshold to fine tune step detection algorithm
  - `*energy_threshold` parameter is a pointer to energy threshold value

2.2.3 API flow chart

Figure 1. MotionPW API logic sequence



2.2.4 Demo code

The following demonstration code example reads data from the accelerometer sensor, obtains the current activity from MotionAW library and gets the number of steps, cadence and confidence from MotionPW library.

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

/* Initialization */
char lib_version[VERSION_STR_LENG];

/* Pedometer API initialization function */
MotionPW_Initialize();

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

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

[…]

/* Using Pedometer for wrist algorithm */
Timer_OR_DataRate_Interrupt_Handler()
{
  MPW_input_t MPW_data_in;
  MPW_output_t MPW_data_out;
```

```
  MAW_input_t MAW_data_in;
  MAW_output_t MAW_data_out;

  /* Get acceleration X/Y/Z in g */
  MEMS_Read_AccValue(&MAW_data_in.Acc_X, &MAW_data_in.Acc_Y, &MAW_data_in.Acc_Z);

  /* Get current activity */
  MotionAW_Update(&MAW_data_in, &MAW_data_out, Timestamp);

  MPW_data_in.Acc_X = MAW_data_in.Acc_X;
  MPW_data_in.Acc_Y = MAW_data_in.Acc_Y;
  MPW_data_in.Acc_Z = MAW_data_in.Acc_Z;

  if (MAW_data_out.current_activity == MAW_WALKING)
  {
    MPW_data_in.currentActivity = MPW_WALKING;
  }
  else if (MAW_data_out.current_activity == MAW_FASTWALKING)
  {
    MPW_data_in.currentActivity = MPW_FASTWALKING;
  }
  else if (MAW_data_out.current_activity == MAW_JOGGING)
  {
    MPW_data_in.currentActivity = MPW_JOGGING;
  }
  else
  {
    MPW_data_in.currentActivity = MPW_UNKNOWN_ACTIVITY;
  }

  /* Run pedometer for wrist algorithm */
  MotionPW_Update(&MPW_data_in, &MPW_data_out);
}
```

2.2.5 Algorithm performance

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

When replicating fitness activity with the STM32 Nucleo board, ensure the board is oriented perpendicularly to the forearm, to simulate the wristband position.

Figure 2. Orientation system for wrist-worn devices

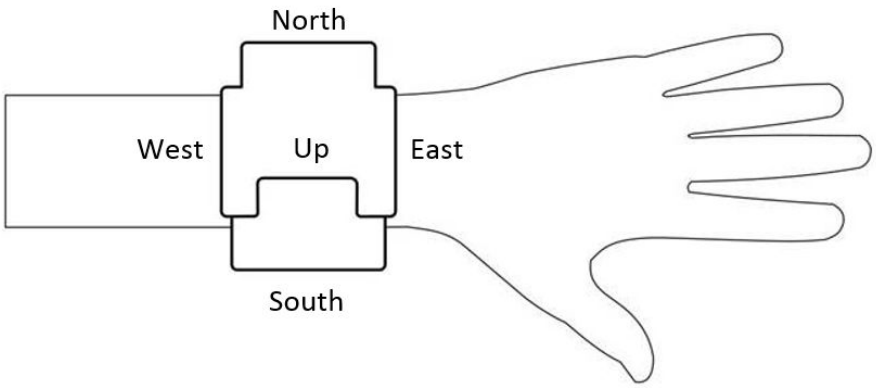


Table 2. Algorithm elapse time (µs) Cortex-M4, Cortex-M3

Cortex-M4 STM32F401RE at 84 MHz			Cortex-M3 STM32L152RE at 32 MHz		
Min	Avg	Max	Min	Avg	Max
38	49	616	296	390	3314

Table 3. **Algorithm elapse time (µs) Cortex-M33 and Cortex-M7**

Cortex- M33 STM32U575ZI-Q at 160 MHz			Cortex- M7 STM32F767ZI at 96 MHz		
Min	Avg	Max	Min	Avg	Max
57	63	359	61	88	1301

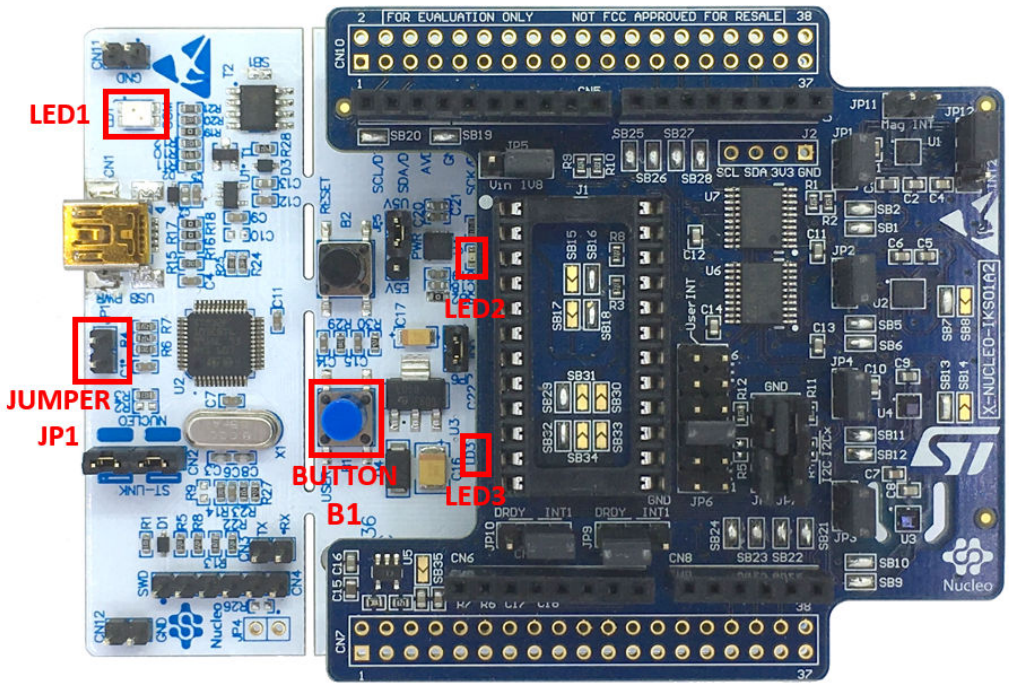
2.3 Sample application

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

The application recognizes the steps, cadence and confidence in real-time. The data can be displayed through a GUI.

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.

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 detected steps, cadence and confidence, accelerometer data, time stamp and eventually other sensor data, in real-time, using the MEMS-Studio.

2.4 MEMS Studio application

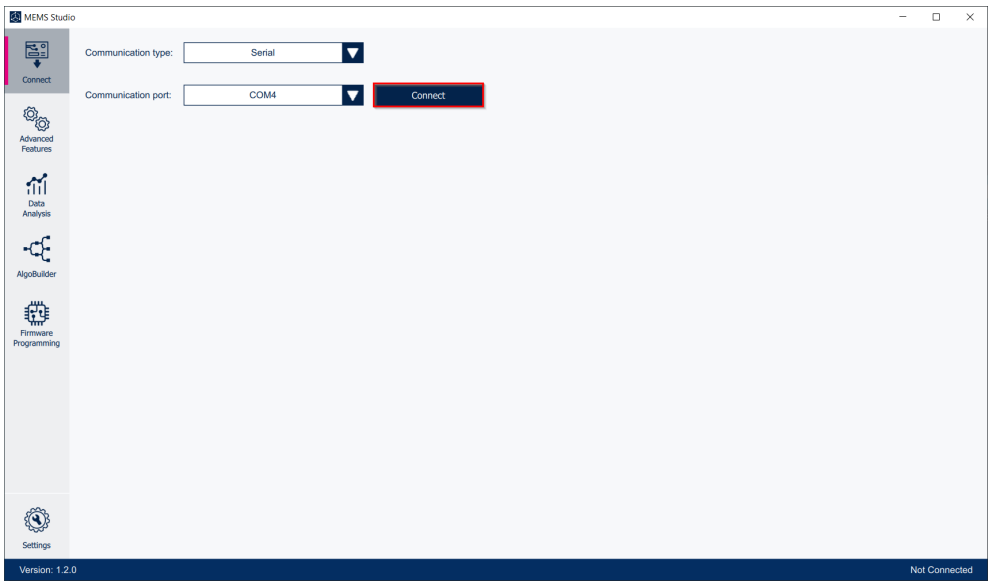
The sample application uses MEMS-Studio application, which can be downloaded from www.st.com.

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

Figure 4. MEMS-Studio - Connect

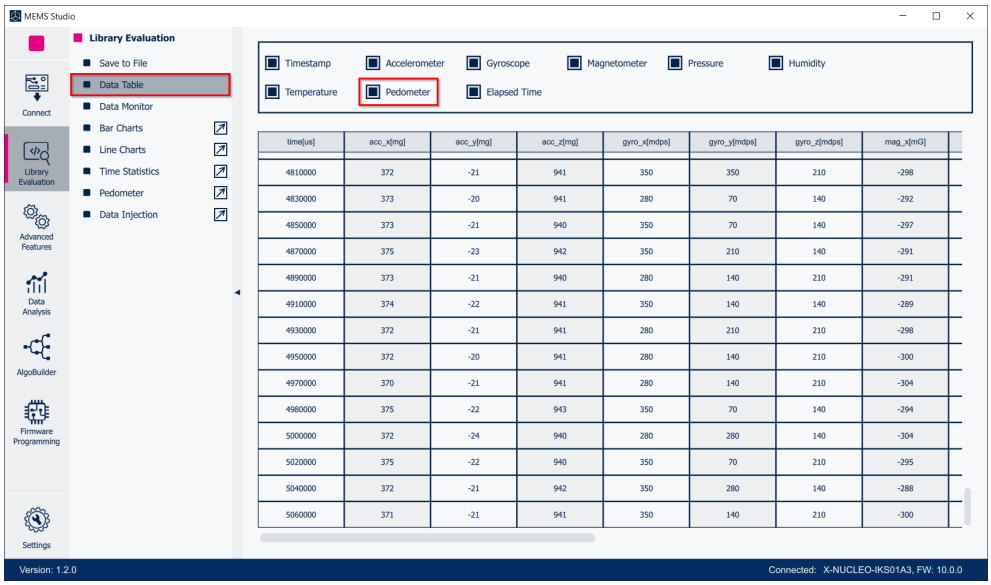


Step 3. When connected to a STM32 Nucleo board with supported firmware [**Library Evaluation**] tab is opened.

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 5. MEMS-Studio - Library Evaluation - Data Table



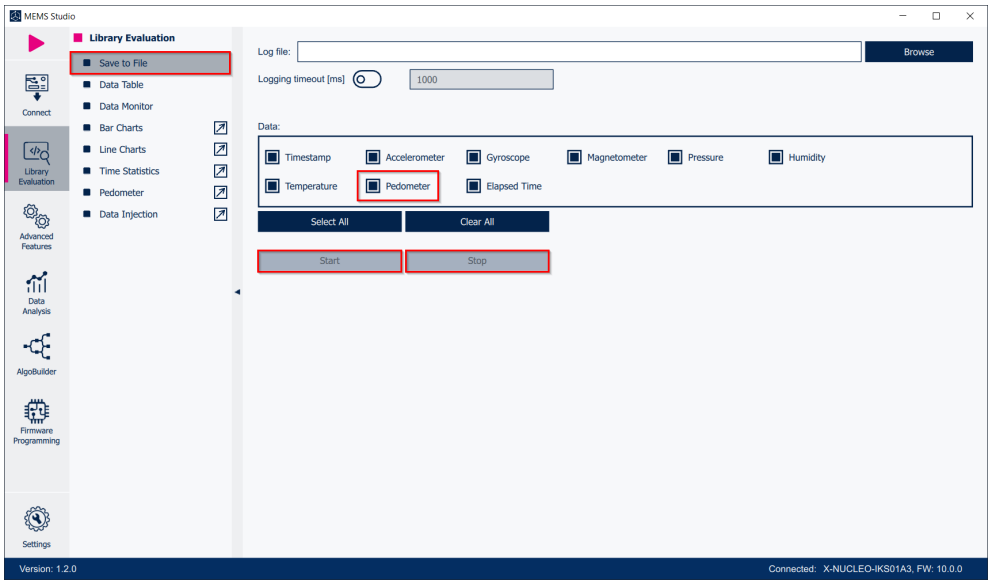
time[us]	acc_x[mg]	acc_y[mg]	acc_z[mg]	gyro_x[mdps]	gyro_y[mdps]	gyro_z[mdps]	mag_x[mG]
4810000	372	-21	941	350	350	210	-298
4830000	373	-20	941	280	70	140	-292
4850000	373	-21	940	350	70	140	-297
4870000	375	-23	942	350	210	140	-291
4890000	373	-21	940	280	140	210	-291
4910000	374	-22	941	350	140	140	-289
4930000	372	-21	941	280	210	210	-298
4950000	372	-20	941	280	140	210	-300
4970000	370	-21	941	280	140	210	-304
4980000	375	-22	943	350	70	140	-294
5000000	372	-24	940	280	280	140	-304
5020000	375	-22	940	350	70	210	-295
5040000	372	-21	942	350	280	140	-288
5060000	371	-21	941	350	140	210	-300

Step 4. Click on the [**Pedometer**] to open the dedicated application window.

Figure 6. MEMS-Studio - Library Evaluation - Pedometer

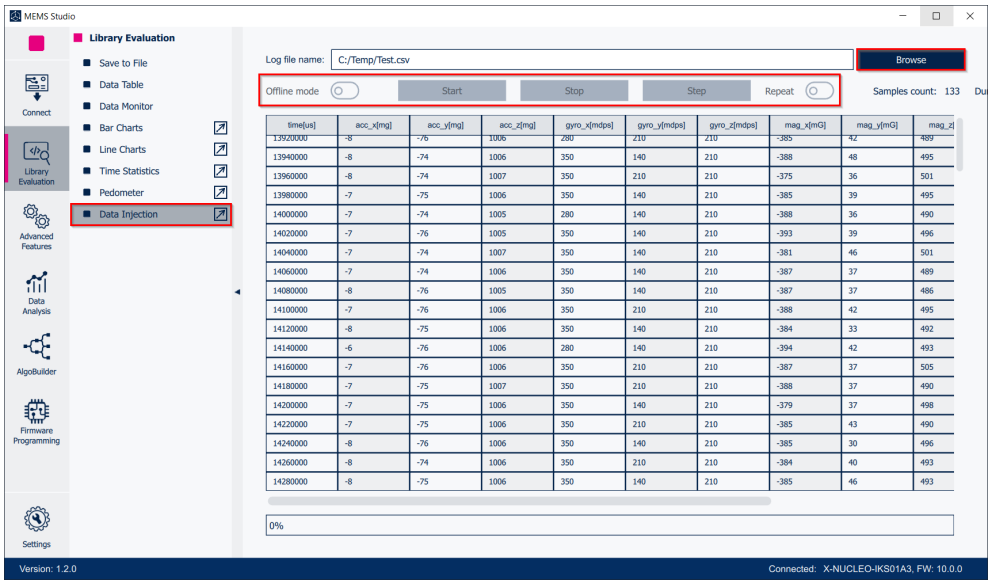
Step 5. Click on the [**Save To File**] to open the dataloging configuration window. Select the sensor and pedometer data to be saved in the file. You can start or stop saving by clicking on the corresponding button.

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



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.

Figure 8. 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 4. **Document revision history**

Date	Version	Changes
24-Jan-2018	1	Initial release.
21-Mar-2018	2	Updated Introduction and Section 2.1 MotionPW overview.
20-Feb-2019	3	Updated *Section 2.2.5: Algorithm performance* and F*igure 3. STM32 Nucleo: LEDs, button, jumper.* Added X-NUCLEO-IKS01A3 expansion board compatibility information.
20-May-2025	4	Updated Section Introduction, Section 2.1: MotionPW overview, Section 2.2.1: MotionPW library description, Section 2.2.2: MotionPW APIs, Section 2.2.4: Demo code, Section 2.2.5: Algorithm performance, Section 2.3: Sample application, Section 2.4: MEMS Studio application

Contents

List of tables

List of figures