UM2216

User manual

Getting started with MotionFA fitness activity library in X-CUBE-MEMS1 expansion for STM32Cube

Introduction

The MotionFA middleware library is a component of the X-CUBE-MEMS1 software and runs on STM32. It provides real-time information about the repetition quantity of various fitness activities performed by a user. It is able to count the number of bicep curls, squats and push-ups. 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 or ARM Cortex-M4 architecture.

It is built on STM32Cube software technology for portability across different STM32 microcontrollers.

The software comes with a sample implementation running on 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 MotionFA middleware library in X-CUBE-MEMS1 software expansion for STM32Cube

2.1 MotionFA overview

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

The library acquires data from the accelerometer, gyroscope and pressure sensor and provides information about the repetition quantity of various fitness activities performed by a 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-IKS01A2 expansion board, mounted on a NUCLEO-F401RE, NUCLEO-L476RG or NUCLEO-L152RE development board.

2.2 MotionFA library

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

2.2.1 MotionFA library description

The MotionFA fitness activity library manages the data acquired from the accelerometer, gyroscope and pressure sensor; it features:

- possibility to count the number of bicep curls, squats and push-ups
- bicep curl and squat recognition based on accelerometer and pressure sensor data
- push-up recognition based on accelerometer and gyroscope sensor data
- required accelerometer, gyroscope and pressure sensor data sampling frequency of 25 Hz
- 17.6 kByte of code memory and 17.4 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 MotionFA APIs

The MotionFA library APIs are:

- `uint8_t MotionFA_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 MotionFA_Initialize(void)`
  - performs MotionFA 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 fitness activity library
- `void MotionFA_BicepCurl_Update(MFA_input_t *data_in, MFA_output_t *data_out)`
  - executes bicep curl detection and counter algorithm
  - `*data_in` parameter is a pointer to a structure with input data
  - the parameters for the structure type `MFA_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
    - `GyrX`, `GyrY`, `GyrZ` is not used during bicep curl recognition

    - `Press` is the atmospheric pressure in hPa
  - `*data_out` parameter is a pointer to a structure `MFA_output_t` with the following parameter:
    - `Counter` is the number of bicep curls

- `void MotionFA_Squat_Update(MFA_input_t *data_in, MFA_output_t *data_out)`
  - executes squat detection and counter algorithm
  - `*data_in` parameter is a pointer to a structure with input data
  - the parameters for the structure type `MFA_input_t` are the following:
    - `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
    - `GyrX`, `GyrY`, `GyrZ` is not used during bicep curl recognition
    - `Press` is the atmospheric pressure in hPa
  - `*data_out` is a pointer to a structure `MFA_output_t` with the following parameter:
    - `Counter` is the number of squats

- `void MotionFA_Pushup_Update(MFA_input_t *data_in, MFA_output_t *data_out)`
  - executes push-up detection and counter algorithm
  - `*data_in` parameter is a pointer to a structure with input data
  - the parameters for the structure type `MFA_input_t` are the following:
    - `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
    - `GyrX` is the gyroscope sensor value in X axis in dps
    - `GyrY` is the gyroscope sensor value in Y axis in dps
    - `GyrZ` is the gyroscope sensor value in Z axis in dps
    - `Press` is not used during push-up recognition
  - `*data_out` is a pointer to a structure `MFA_output_t` with the following parameter:
    - `Counter` is the number of push-ups

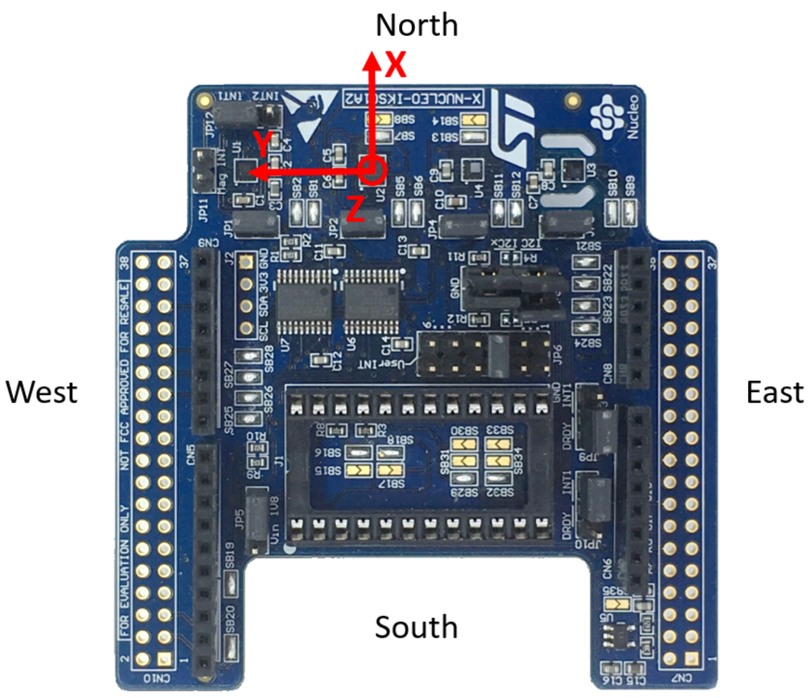
- `void Motion_BicepCurl_Reset(void)`
  - resets bicep curl counter

- `void MotionFA_Squat_Reset(void)`
  - resets squat counter

- `void MotionFA_Pushup_Reset(void)`
  - resets push-up counter

- `void MotionFA_SetOrientation_Acc (const char *acc_orientation)`
  - this function is used to set the accelerometer data orientation
  - configuration is usually performed immediately after the `MotionFA_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”.

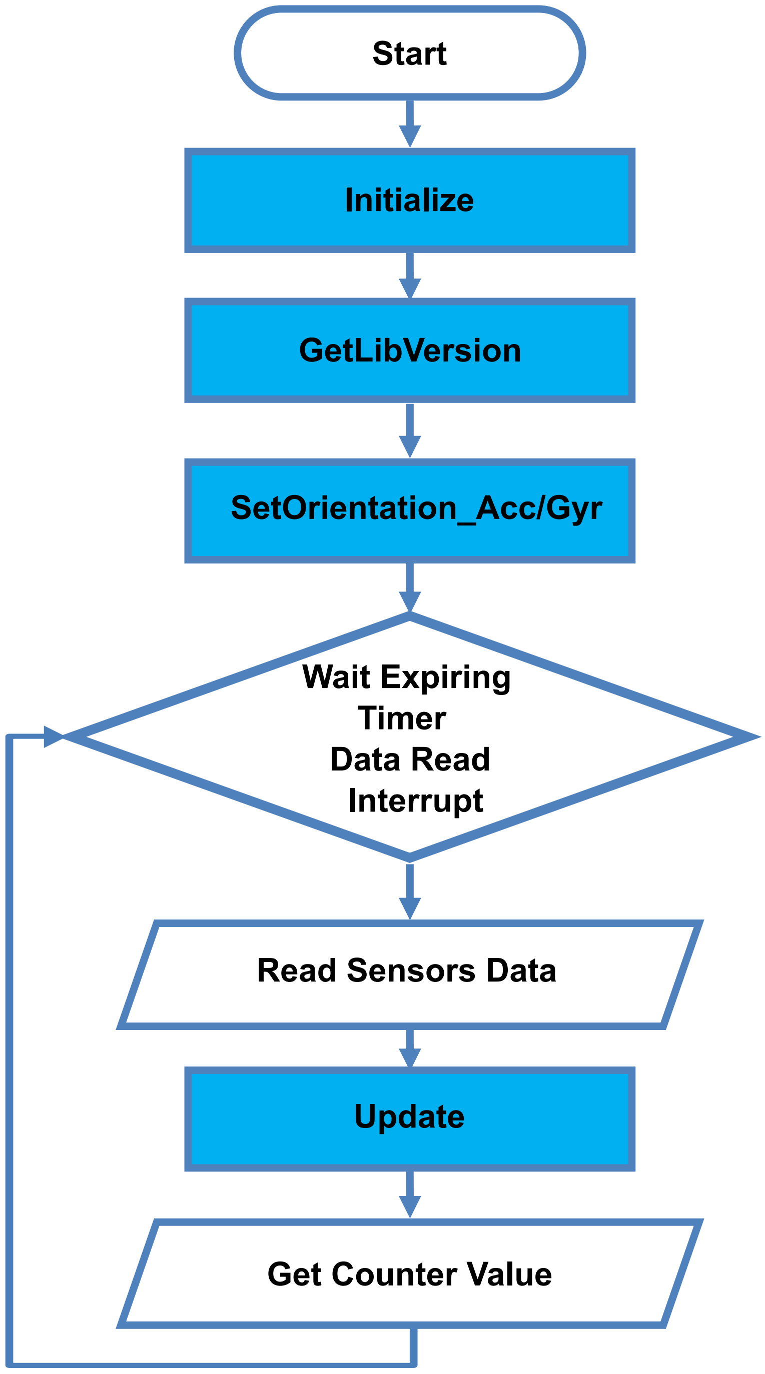
Figure 1. Example of sensor orientations



- `void MotionFA_SetOrientation_Gyr (const char *gyr_orientation)`
  - this function is used to set the gyroscope data orientation
  - configuration is usually performed immediately after the `MotionFA_Initialize` function call
  - `*gyr_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 gyroscope 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 above, the X-NUCLEO-IKS01A2 gyroscope sensor has an NWU orientation (x - North, y - West, z - Up), so the string is: “nwu”.

2.2.3 API flow chart

Figure 2. MotionFA API logic sequence



2.2.4 Demo code

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

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

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

/* Fitness Activities API initialization function */
MotionFA_Initialize();

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

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

/* Set gyroscope orientation */
gyr_orientation[0] ='n';
gyr_orientation[1] ='w';
gyr_orientation[2] ='u';
MotionFA_SetOrientation_Gyr(gyr_orientation);

[…]

/* Select activity i.e. bicep curl */
activity = MFA_BICEPCURL;

[…]

/* Select activity i.e. bicep curl */
activity = MFA_BICEPCURL;
{
MFA_input_t data_in;
MFA_output_t data_out;

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

/* Get angular velocity X/Y/Z in dps */
MEMS_Read_GyrValue(&data_in.GyrX, &data_in.GyrY, &data_in.GyrZ);

/* Get atmospheric pressure in hPa */
MEMS_Read_PressValue(&data_in.Press);

/* Fitness Activities algorithm update */
if (activity == MFA_BICEPCURL)
{
MotionFA_BicepCurl_Update(&data_in, &data_out);
}
else if (activity == MFA_SQUAT)
{
MotionFA_Squat_Update(&data_in, &data_out);
}
else if (activity == MFA_PUSHUP)
{
MotionFA_Pushup_Update(&data_in, &data_out);
}
}
```

2.2.5 Algorithm performance

The fitness activity algorithm uses data from the accelerometer, gyroscope and pressure sensor and runs at a low frequency (25 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.

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
2	91	9901	2	72	7692	3	15796	378	33	697	73800	32	608	68514	31	844	112055

2.3 Sample application

The MotionFA 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 (based on LSM6DSL) expansion board.

The application counts the number of bicep curls, squats or push-ups in real-time. Data can be displayed through a GUI or stored in the board for offline analysis.

Stand-alone mode

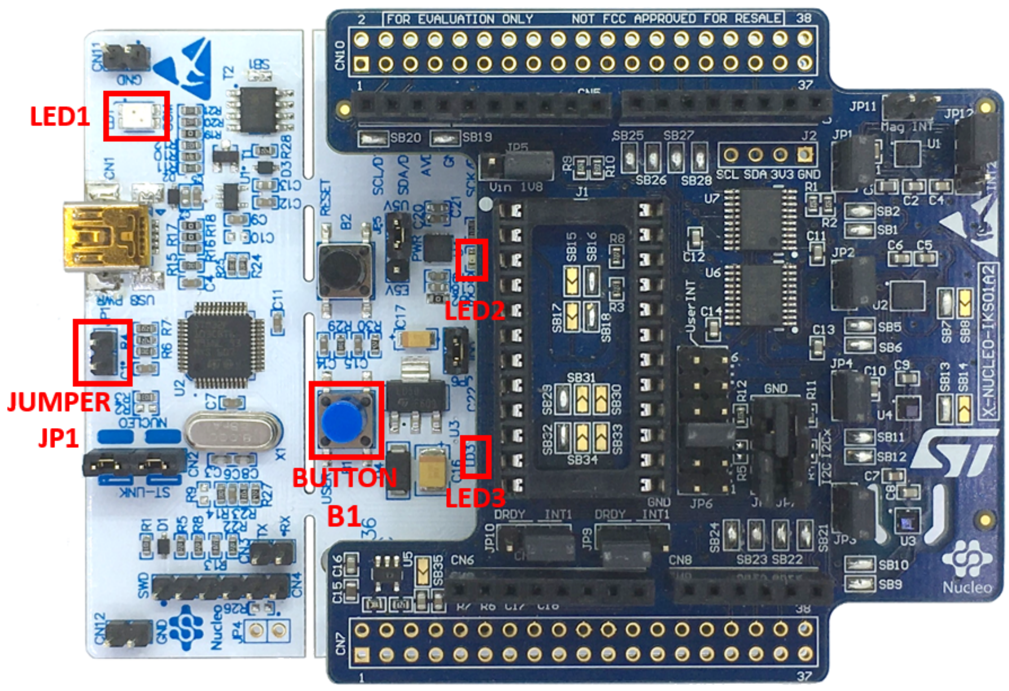
In stand-alone mode, the sample application allows the user to detect the performed gesture and store it in the MCU flash memory.

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 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 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 gesture; 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. 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 8,000 data sets.

To retrieve this data, the board must 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 real-time counter values, accelerometer and pressure data, time stamp and any other sensor data, using the Unicleo-GUI.

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

2.4 Unicleo-GUI application

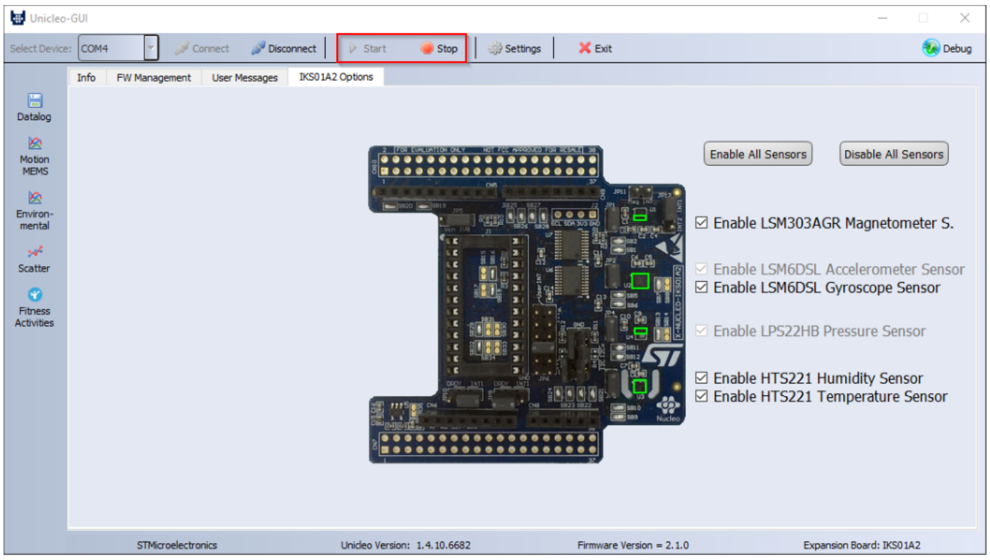
The sample application uses the Windows Unicleo-GUI utility, which can be downloaded from www.st.com.

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 4. Unicleo main window



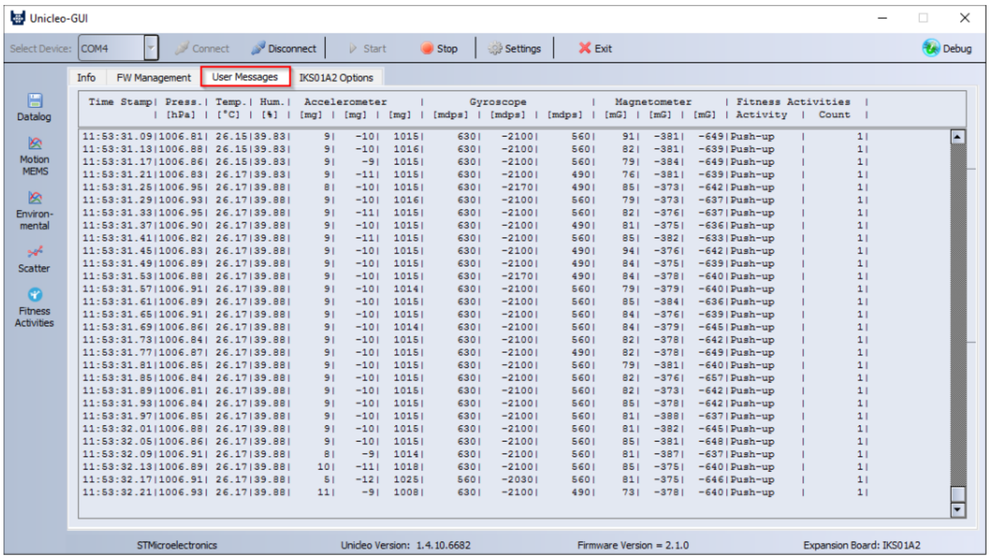
Enable All Sensors
Disable All Sensors
☑ Enable LSM303AGR Magnetometer S.
☑ Enable LSM6DSL Accelerometer Sensor
☑ Enable LSM6DSL Gyroscope Sensor
☑ Enable LPS22HB Pressure Sensor
☑ Enable HTS221 Humidity Sensor
☑ Enable HTS221 Temperature Sensor

STMicroelectronics Unicleo Version: 1.4.10.6682 Firmware Version = 2.1.0 Expansion Board: IKS01A2

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

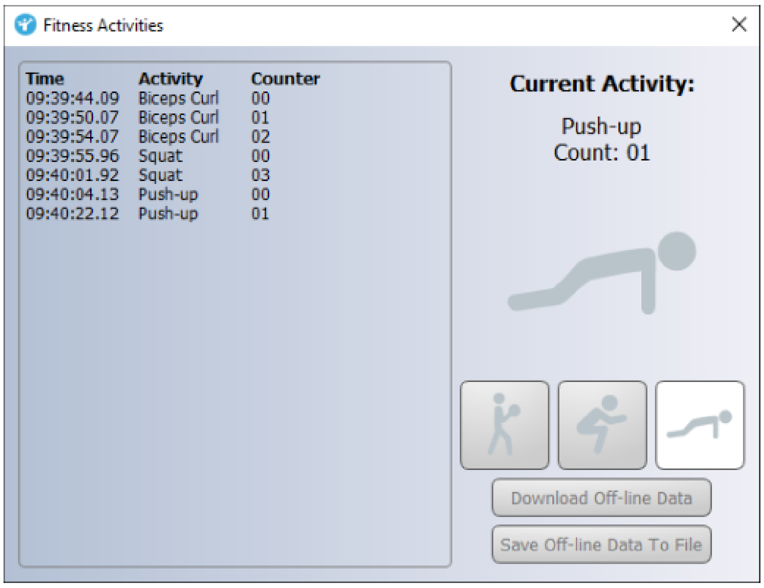


```
  Time Stamp| Press.| Temp.| Hum.| Accelerometer     |     Gyroscope           |  Magnetometer      | Fitness Activities |
            | [hPa] | [°C] | [%] | [mg] | [mg] | [mg] | [mdps] | [mdps] | [mdps] | [mG] | [mG] | [mG] | Activity | Count  |
11:53:31.09|1006.81| 26.15|39.83|    9|  -10|  1015|    630|  -2100|    560|   91|  -381|  -649|Push-up   |      1|
11:53:31.13|1006.88| 26.15|39.83|    9|  -10|  1016|    630|  -2100|    560|   82|  -381|  -639|Push-up   |      1|
11:53:31.17|1006.86| 26.15|39.83|    9|   -9|  1015|    630|  -2100|    560|   79|  -384|  -649|Push-up   |      1|
11:53:31.21|1006.83| 26.17|39.83|    9|  -11|  1015|    630|  -2100|    490|   76|  -381|  -639|Push-up   |      1|
11:53:31.25|1006.95| 26.17|39.88|    8|  -10|  1015|    630|  -2170|    490|   85|  -373|  -642|Push-up   |      1|
11:53:31.29|1006.93| 26.17|39.88|    9|  -10|  1016|    630|  -2100|    560|   79|  -373|  -637|Push-up   |      1|
11:53:31.33|1006.95| 26.17|39.88|    9|  -11|  1015|    630|  -2100|    560|   82|  -376|  -637|Push-up   |      1|
11:53:31.37|1006.90| 26.17|39.88|    9|  -10|  1015|    630|  -2100|    490|   81|  -375|  -636|Push-up   |      1|
11:53:31.41|1006.82| 26.17|39.88|    9|  -11|  1015|    630|  -2100|    560|   85|  -382|  -633|Push-up   |      1|
11:53:31.45|1006.83| 26.17|39.88|    9|  -10|  1015|    630|  -2100|    490|   94|  -376|  -642|Push-up   |      1|
11:53:31.49|1006.89| 26.17|39.88|    9|  -10|  1015|    630|  -2100|    490|   84|  -375|  -639|Push-up   |      1|
11:53:31.53|1006.88| 26.17|39.88|    9|  -10|  1015|    630|  -2170|    490|   84|  -378|  -640|Push-up   |      1|
11:53:31.57|1006.91| 26.17|39.88|    9|  -10|  1014|    630|  -2100|    560|   79|  -379|  -640|Push-up   |      1|
11:53:31.61|1006.89| 26.17|39.88|    9|  -10|  1015|    630|  -2100|    560|   85|  -384|  -636|Push-up   |      1|
11:53:31.65|1006.91| 26.17|39.88|    9|  -10|  1015|    630|  -2100|    560|   84|  -376|  -639|Push-up   |      1|
11:53:31.69|1006.86| 26.17|39.88|    9|  -10|  1014|    630|  -2100|    560|   84|  -379|  -645|Push-up   |      1|
11:53:31.73|1006.84| 26.17|39.88|    9|  -10|  1015|    630|  -2100|    560|   82|  -378|  -642|Push-up   |      1|
11:53:31.77|1006.87| 26.17|39.88|    9|  -10|  1015|    630|  -2100|    490|   82|  -378|  -649|Push-up   |      1|
11:53:31.81|1006.85| 26.17|39.88|    9|  -10|  1015|    630|  -2100|    560|   79|  -381|  -640|Push-up   |      1|
11:53:31.85|1006.84| 26.17|39.88|    9|  -10|  1015|    630|  -2100|    560|   82|  -376|  -657|Push-up   |      1|
11:53:31.89|1006.81| 26.17|39.88|    9|  -10|  1015|    630|  -2100|    560|   82|  -373|  -642|Push-up   |      1|
11:53:31.93|1006.84| 26.17|39.88|    9|  -10|  1015|    630|  -2100|    560|   85|  -378|  -642|Push-up   |      1|
11:53:31.97|1006.85| 26.17|39.88|    9|  -10|  1015|    630|  -2100|    560|   81|  -388|  -637|Push-up   |      1|
11:53:32.01|1006.88| 26.17|39.88|    9|  -10|  1015|    630|  -2100|    560|   81|  -382|  -645|Push-up   |      1|
11:53:32.05|1006.86| 26.17|39.88|    9|  -10|  1015|    630|  -2100|    560|   85|  -381|  -648|Push-up   |      1|
11:53:32.09|1006.91| 26.17|39.88|    8|   -9|  1014|    630|  -2100|    560|   81|  -387|  -637|Push-up   |      1|
11:53:32.13|1006.89| 26.17|39.88|   10|  -11|  1018|    630|  -2100|    560|   85|  -375|  -640|Push-up   |      1|
11:53:32.17|1006.91| 26.17|39.88|    5|  -12|  1025|    560|  -2030|    560|   81|  -375|  -646|Push-up   |      1|
11:53:32.21|1006.93| 26.17|39.88|   11|   -9|  1008|    630|  -2100|    490|   73|  -378|  -640|Push-up   |      1|
```

STMicroelectronics Unicleo Version: 1.4.10.6682 Firmware Version = 2.1.0 Expansion Board: IKS01A2

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

Figure 6. **Fitness Activities window**



Time	Activity	Counter
09:39:44.09	Biceps Curl	00
09:39:50.07	Biceps Curl	01
09:39:54.07	Biceps Curl	02
09:39:55.96	Squat	00
09:40:01.92	Squat	03
09:40:04.13	Push-up	00
09:40:22.12	Push-up	01

Current Activity:

Push-up
Count: 01

To switch between bicep curl, squat and push-up counting click on the appropriate icon.

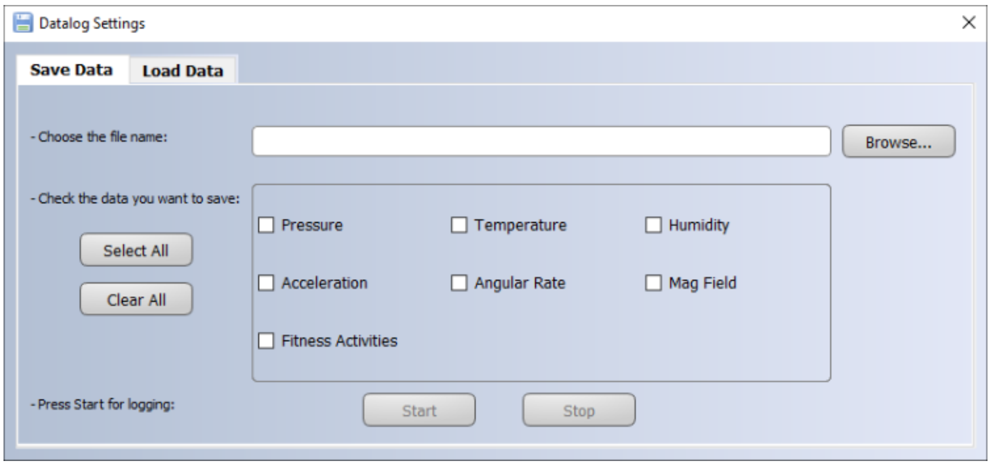
If the board has been working in standalone mode and you wants to retrieve stored data, press **Download Off-line Data** button to upload the stored activities data to the application. This operation automatically deletes acquired data from microcontroller.

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

Note: ***Download Off-line Data*** *button is not available while data streaming is active*

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**



- Choose the file name:

- Check the data you want to save:

☐ Pressure	☐ Temperature	☐ Humidity
☐ Acceleration	☐ Angular Rate	☐ Mag Field
☐ Fitness Activities		

- Press Start for logging:

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
02-May-2017	1	Initial release.
06-Feb-2018	2	Added references to NUCLEO-L152RE development board and Table 2. Elapsed time (µs) algorithm.
20-Mar-2018	3	Updated Introduction and Section 2.1 MotionFA overview.
01-Oct-2018	4	Removed references to X-NUCLEO-IKS01A1 expansion board throughout document. Updated Section 2.2.1 MotionFA library description, Section 2.2.2 MotionFA APIs, Section 2.2.3 API flow chart, Section 2.2.4 Demo code, Figure 3. STM32 Nucleo: LEDs, button, jumper and Section 2.4 Unicleo-GUI application.

Contents

List of tables

List of figures