Software User Guide

For

Tungsten eMD

TABLE OF CONTENTS

TABLE OF FIGURES/DIAGRAMS

USEFUL LINKS

InvenSense website:

http://www.invensense.com/

Arduino website:

https://www.arduino.cc/en/Guide/Environment

https://www.arduino.cc/en/Main/ArduinoBoardZero

https://www.arduino.cc/en/uploads/Main/Arduino-Zero-schematic.pdf

1. OVERVIEW

The purpose of this document is to give an overview of the Tungsten embedded Developer Kit in order to begin creating a custom sensor application. This document also serves as a quick start guide to present Invensense ICM-30630 package and its elements, how to setup the development environment and how to use the sample applications provided.

1.1. INTRODUCTION

The Tungsten-eMD solution is compatible with Arduino Zero and based on GCC development tools. The purpose of this solution is to allow sensors management and algorithms processing using a standalone microcontroller. The Tungsten-eMD solution was created as an advanced sensor hub easy to integrate for user developing in wearable and IoT space. The developer kit includes a fully sensor software solution and additional memory space available on chip for the user to program with their own logic.

1.2. TUNGSTEN-EMD BASICS

1.2.1. Overview

Tungsten-eMD provides the ICM-30630 Invensense chip. The ICM-30630 chip uses three different processors available: an ARM Cortex-M0, the DMP3 and the DMP4.

The DMP3 (Digital Motion Processor 3) is dedicated to offload motion processing from the CPU and provides ready-to-use physical and virtual sensors based on Android L solution. The DMP image has 6-axis (Accelerometer + Gyroscope) support including calibration.

The DMP4 (Digital Motion Processor 4) is higher functioning processor with specialties in fixed point Q30 processing and 16-bit FFT. The DMP4 complements the CPU by offloading computationally-intensive operations.

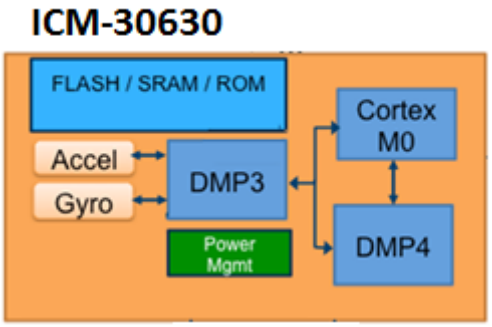

Figure 1 – Tungsten-eMD architecture

1.2.2. Tungsten-eMD hardware layout

The Arduino Zero board communicates to the ICM-30630 through SPI, to provide a fast communication pathway. It's also needed to have two interrupt lines from ICM-30630 connected to a wake-up pin and to a non-wake-up pin on application. On Arduino Zero, the wake-up interrupt (GPIO0) is connected to pin J102-D5 and the non-wake-up interrupt (GPIO1) is connected to the pin J102-D6. The two interrupt lines signal wake-up and non-wake-up events according to Android L specification and can allow the application to sleep to saving power. External sensors are available on Sensor Daughter board and are connected to ICM-30630 using I2C. These sensors include, but are not limited to, magnetometer, proximity sensor, and barometer.

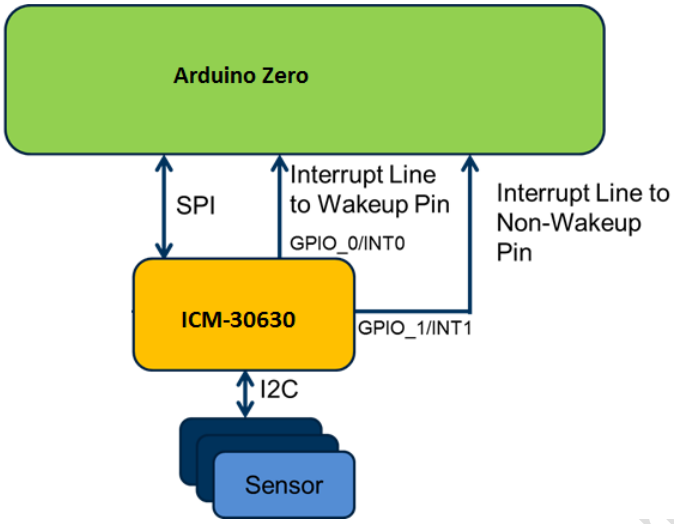

Figure 2 – Recommended hardware layout for Tungsten-eMD

2. SOFTWARE DEVELOPMENT PLATFORM

2.1. SOLUTION PLATFORM

This section will describe the hardware components needed to setup the Tungsten-eMD solution. The proposed solution platform includes:

- Application board with a standalone microcontroller, here the **Arduino Zero** board. For more information about the Arduino Zero board, please refer to Arduino website in section Useful links above.

- The **Tungsten-eMD** shield including the ICM-30630 with three combined processors, an accelerometer and a gyrometer sensor. All the connectors on the bottom of the shield will be connected to Arduino Zero pins.
  The board can support two additional daughter boards to connect to the front connectors.
  The board also supports a power supply connected to the jack but it's not providing with Tunsgten-eMD Developer Kit.

Be careful to jumpers' configuration:

- JP1 : connect VDD to VDDIO
- JP2 : connect VDD to 3V3
- JP3 : close

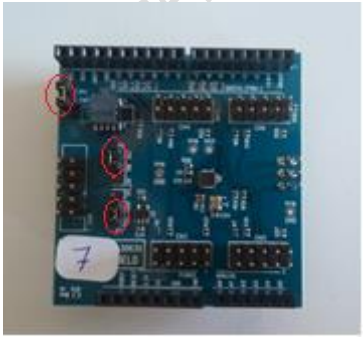

Figure 3 – Tungsten-eMD shield

- The **Sensor daughter board**, are optional but add available external sensors: Magnetometer, Proximity sensor and Pressure sensor. The connectors on the bottom of the board will be connected to Tunsgten-eMD shield.

Be careful to jumpers' configuration:

- J100 : [1<->2] close, [3<->4] close, [5<->6] open, [7<->8] open

Figure 4 – Sensor daughter board

2.2. DIRECTORY STRUCTURE

The Invensense Tungsten-eMD package includes all the necessary files to get started and creating custom application. There are two parts in the installed package:

- **examples**: Application sample codes using Invensense library to communicates with Tungsten-eMD. All the sample codes contain a .ino file compatible with Arduino IDE (for more information about sample code, please refer to section 4).

- **src**: Library sources needed to interface with Tungsten-eMD.
  - **Bridge:** Library sources needed to encapsulate protocol to communicate with Invensense Studio, a suite of tools to help developing applications or new features on Invensense devices.
  - **Invn:** Libraries to communicate with Tungsten-eMD
    - **Devices:** Contains Invensense Device Driver library that defined API methods to control and retrieve data from Invensense devices.
    - **FifoProtocol:** Contains Invensense FIFO Protocol to communicate with Tungsten-eMD.
    - **Utils:** Contains helper sources for these libraries.
  - **Arduino Adapter:** Contains adapter layout for SPI communication between Arduino Zero and Tungsten-eMD.
  - **DeviceIoHal:** Contains HAL for SPI communication using Arduino API.
  - **EasyDevice:** Contains simplify layout to access to the Invensense Device Driver library.

2.3. PROJECTS

There Invensense Tungsten-eMD package includes projects for very basic functionalities. Basic projects should be used as a starting point for application development and are briefly described below. For more details on these included projects, please refer to section 4.

- The **Bridge** project is used to interface with Invensense Studio tool (please refer to Invensense Studio documentation for more information).

- The **AccAndGyrCalibration** project consists of sample code to compute calibration coefficients in order to have accurate bias computed for accelerometer and gyrometer sensors. This sample code generates calibration tables that could be directly used in other application. Please, refer to "DoorOpenningDetection" sample code (section 4.2) that explains how to use calibration tables.

The "detecting" projects consist of sample codes that demonstrate how to use the Tungsten-eMD Developer Kit to detect wanted events. They can be use as reference for developing a detecting application.

- The **DoorOpeningDetection** project shows how to detect a door opening. When the gesture is detected a LED is blinking on Arduino Zero to inform the user.

- The **GestureDetection** project demonstrates how to detect gestures; tilting gesture or "SMD" gesture (for Significant Motion Detection). For each event detected a message is printed on the Arduino Serial Monitor.

The "GetData" projects demonstrate very simple uses of Invensense Device Driver library to get different sensor data from the Tungsten-eMD and the Sensor Daughter board. They can be use as reference for example to develop sensor data processing application.

- **GetAccelerometerData**, this sample code shows how to get calibrated accelerometer data. Data are read from the accelerometer sensor integrated in Tungsten-eMD and printed on Arduino Serial Monitor.

- **GetExternalSensors**, this sample code shows how to get data from external sensors available on Sensor Daughter board: Proximity, Pressure and Magnetometer sensor. All data are printed on Arduino Serial Monitor.

- **GetQuaternion**, this sample code shows how to get quaternion data, which represents orientation of the device based on accelerometer and gyrometer sensor data. Data are read from sensors available on Tungsten-eMD, computed by the processors and printed on Arduino Serial Monitor.

2.4. SETTING UP THE DEVELOPMENT ENVIRONMENT

Before processing further it is necessary to set up the Integrated Development Environment (IDE) in order to browse through the relevant projects and view code. All embedded software is developed using Arduino IDE. This section provides information on where to find this software and how to properly configure the environment.

2.4.1. Installing the Arduino IDE

- Download the Arduino Software (IDE) from https://www.arduino.cc/en/Main/Software. Run the installer "Arduino-xxxx-windows.exe". Where "xxxx" is the IDE revision number. The oldest IDE revision number supported by the Invensense Tungsten-eMD Developer Kit is 1.6.5-r2.
  In the Arduino Setup Installation options select "Install Arduino Software" and also "Install USB driver". The default IDE install path is "C:\Program Files (x86)\Arduino".

- Once the software is installed, run Arduino.exe
  The Arduino IDE software also creates an Arduino folder by default in "Documents library". After installation this folder is empty, but will contain custom .ino sketches and custom libraries.

- Configure the Board Manager in order to have access to the needed Arduino Zero package.
  Click on Tools->Board: "xxxxxxxxx" -> Board Manager

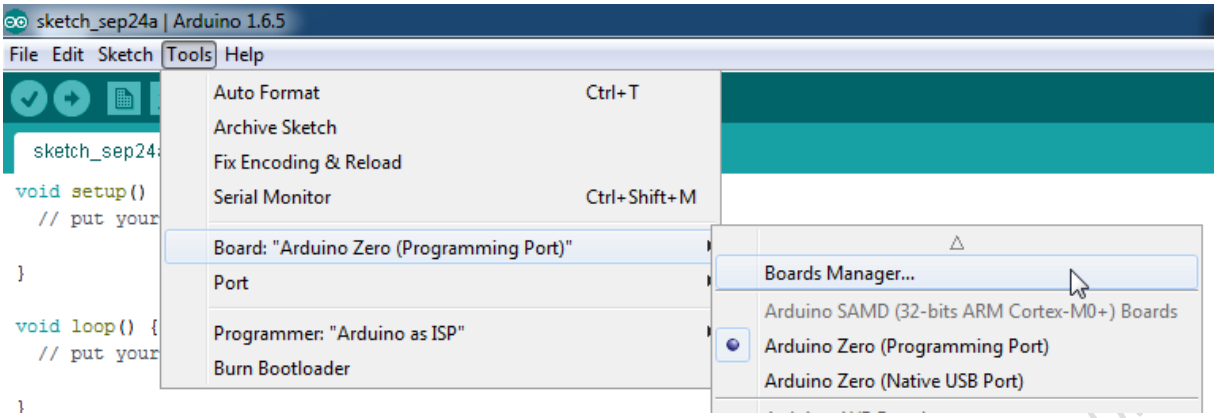

Figure 5 – Click on Board Manager

- Select **Arduino SAMD Boards** and click Install.

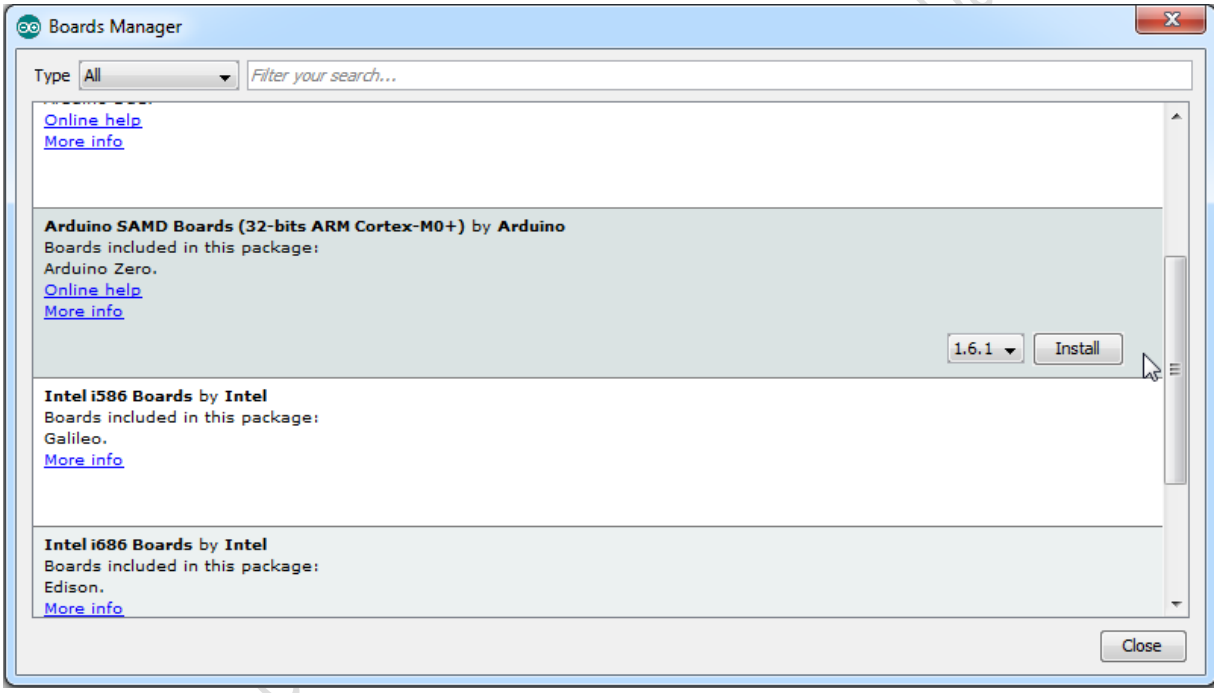

Figure 6 – Install Arduino SAMD Boards package

Once the package for Arduino Zero is installed, click close.

- Exit Arduino IDE.

2.4.2. PC connection

- Connect Arduino Zero board through Micro USB (**PROGRAMMING port**) and let the windows install the driver.

Note: Arduino Zero comes with 2 USB ports. The bottom side of the board has labels for identification. Please refer to the Figure 7 below.

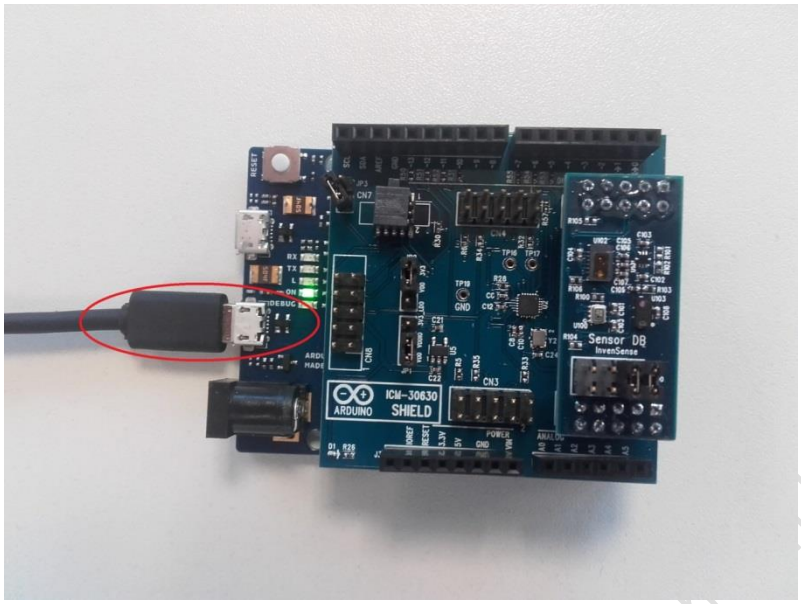

Figure 7 – Connect to PC using USB Programming Port

- On Arduino IDE, select the Arduino Zero programming port. Click on Tools->Boards->Arduino Zero (Programming Port)

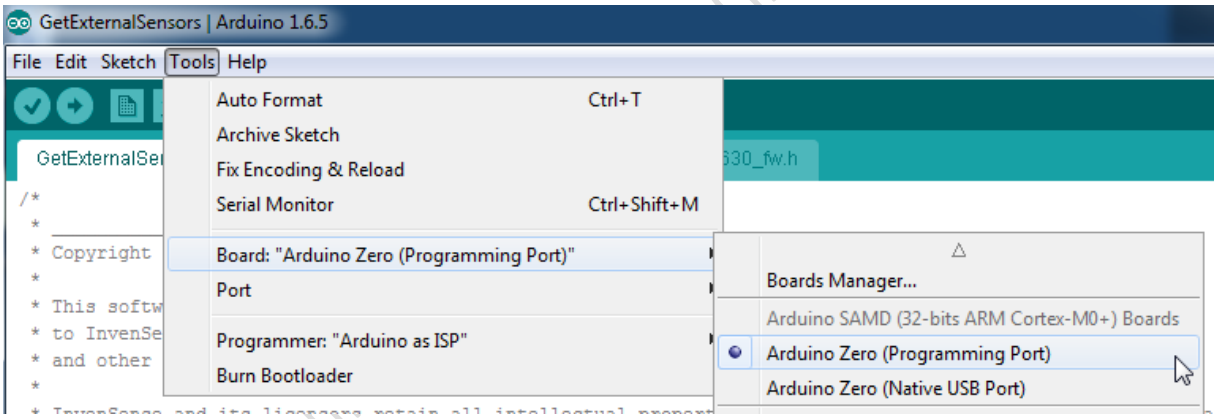

Figure 8 – Select Board: Arduino Zero (Programming Port)

- Select the correct COM port. Click on Tools->Port-> COMxx (Arduino Zero (Programming Port)). To find the COM port number, open the Device Manager on your computer and look for the label “Atmel Corp. EDBG CMSIS-DAP” and the associated COM number. The Arduino Zero Programming port is connected to Atmel chip EDBG (please refer to Arduino Zero datasheet for more information in section “Useful links”).

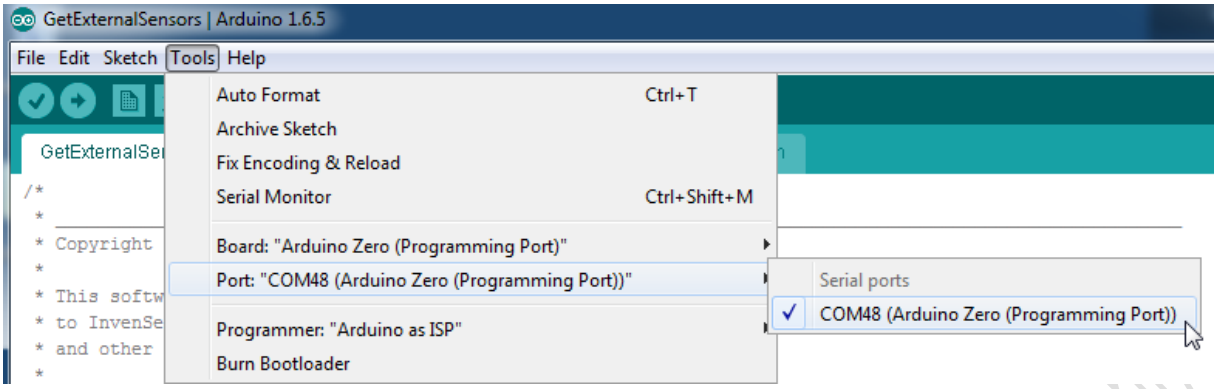

Figure 9 – Select Port: COMxx (Arduino Zero (Programming Port))

2.4.3. Package installation

- Install the Invensense Tungsten-eMD Developer Kit. Click on Sketch->Include Library -> Add .ZIP Library…

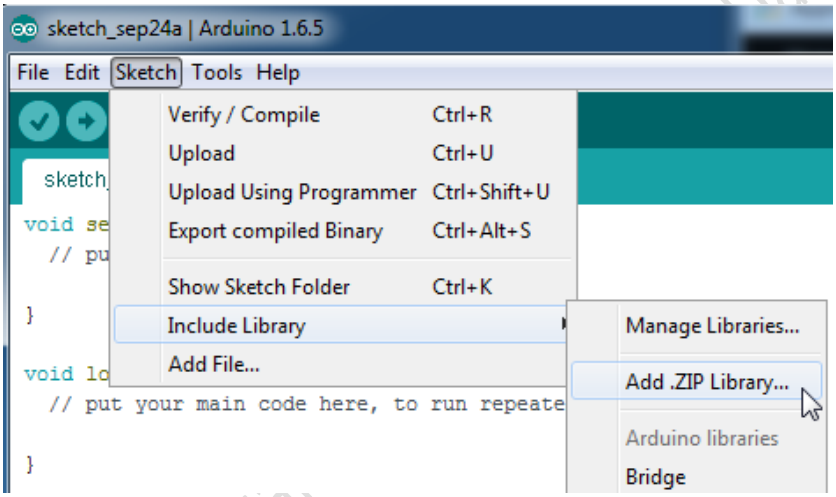

Figure 10 – Add the .ZIP library

Select the package **InvenSense-Tungsten-eMD_V_xxxx.zip**, where "xxxx" is the version number. And you will see this message displayed to confirm the library is correctly add to custom library.

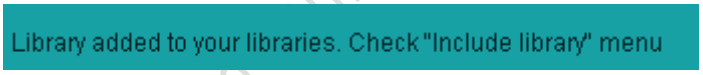

Figure 11 – Displayed message when adding the Invensense ICM-30630 library

The Invensense Tungsten-eMD package is installed in the local Arduino documents directory (in "Documents library" by default). Inside you will find a directory named "Libraries". The imported library directory is there. You will find the main Invensense ICM-30630 library directory containing sources and examples (for more information about directory structure, please refer to section 2.2).

- Update the package version if available by reproducing the procedure above

When updating the ICM-30630 library version, the Arduino IDE installs the library at the same path defined above but in a different folder; the libraries are duplicated but give access separately to the two versions. The Arduino IDE will always automatically load the most recent library. If ever you want to load to a previous version, you should remove the latest library directory directly from Arduino documents library location.

2.5. COMPILING AND DOWNLOAD

- Now you can select a sample code.
  Click on File->Examples->Invensense ICM-30630 library ->"xxxxxxxxxx"

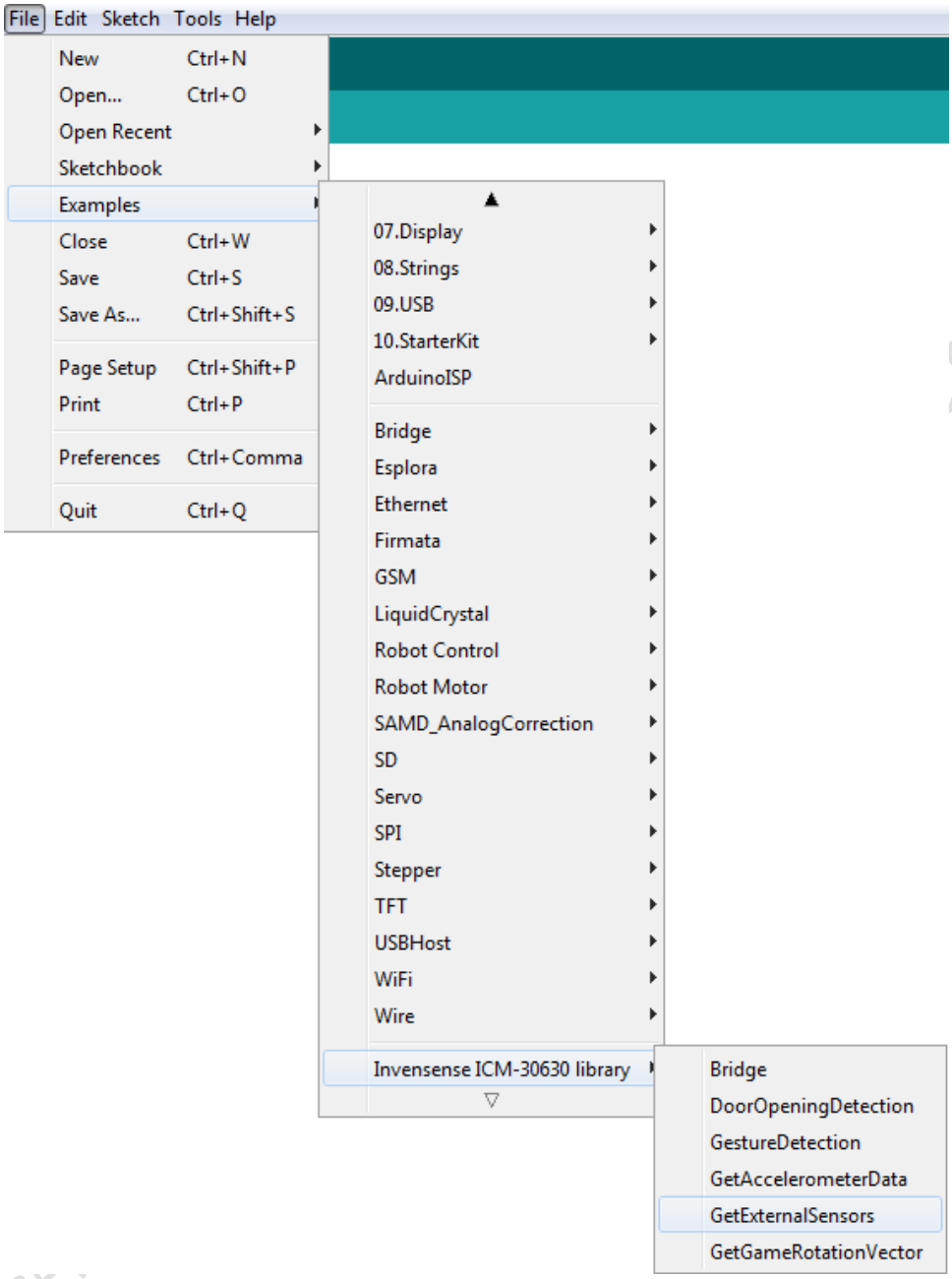

Figure 12 – Invensense ICM-30630 library examples tree

- Build the sketch by clicking on “Verify” button

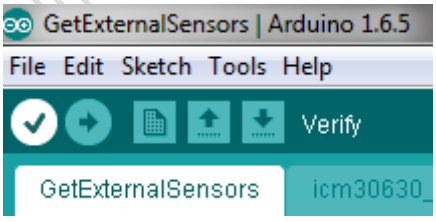

Figure 13 - Click on Verify

- Build and download the sketch. Click on “Upload” button

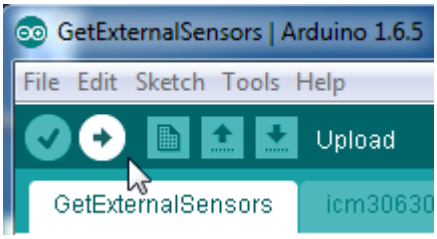

Figure 14 - Click on Upload

3. THE APPLICATION

This section will described the application portion of projects using Easy Device library to communicate with the Tungsten-eMD.

3.1. START-UP IN .INO FILE

The .ino file is the starting point at runtime. There is no difference with a .cpp file about the file format and the programming language. Every project must have a .ino or sketch file that containing the setup() and loop() function.

3.2. GENERAL APPLICATION ARCHITECTURE

3.2.1. Application Setup Function

The *setup()* function is where the board peripherals are initialized using Arduino API. For example the serial interface setup is called to get traces.

Also in this function, Easy device library is initialized to start the communication with Tungsten-eMD. In the Easy device structure parameters, there are callbacks function (refer to 3.2.3 Callbacks) and all the images needed for setup.

There are three major image loading steps needed to bring-up the Tungsten-eMD: the firmware programming, the DMP3 and the DMP4 image loading. These images are given to the Easy device structure and are available in header files:

- icm_30630_fw.h,
- icm_30630_dmp3.h,
- icm_30630_dmp4.h

The firmware programming step is optional, you can use DISABLE_FW_M0_PROG define to avoid reprogramming the Tungsten-eMD at each Easy device initialization.

Another requirement for the Easy Device setup is to provide a minimum of 4Kbytes buffer needed to read Tungsten-eMD FIFOs before parsing data read through SPI. More the allocate buffer is big, more the communication will be fast.

It is possible to directly start a sensor after setup the Easy device library by calling *inv_easy_device_start_sensor().* You should specify parameters:

- the device structure, already defined for the Easy device initialization,
- the sensor ID, the full list of available sensors is available in Invensense Device Driver library, refer to inv_sensor_type structure in SensorType.h header,
- the requested data period in milliseconds,
- the allowed timeout in milliseconds.

3.2.2. Application Loop Function

The *loop()* function checks notified events.

For the Easy device, you may check flag reporting new data available through Easy device library callback and calls data processing. Each data available event notified should be followed by the polling function *inv_device_poll()* in order to get all available data.

3.2.3. Callbacks

There are two callbacks to pass in parameters of the Easy device initialization function. The interrupt to notify there are data available in FIFOs, and sensor event callback to notify data reported.

The interrupt callback is called under interrupt, so the code inside must be thread safe. The data processing should be done in another safe context.

The sensor event callback is called when sensor data received. For each sensor data you have access to a sensor structure containing sensor ID, status event, and all sensor data. This callback has two parameters available:

- event, the sensor event structure. Refer to inv_sensor_event structure in Invensense Device Driver library, see SensorType.h header.
- arg, listener context.

3.3. SERIAL MONITOR

How can I verify my code correctly running? You could use the Arduino Serial interface. You may refer to every sample codes printing traces on the Serial Monitor interface. This Arduino tool is a terminal window that communicates by receiving and sending serial data over USB. There are two serial interfaces available defined by Arduino on Arduino Zero, “Serial” class for USB Programming port and “SerialUSB” class for USB Native port. It’s recommended to use the USB Programming port. Refer to SERIAL_TRACES define in sample code in order to use the other port. The bellow step shows how to use the serial monitor in Arduino IDE:

- Open the Arduino Serial Monitor

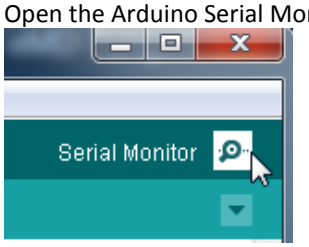

Figure 15 – Serial monitor interface shortcut

- Select the speed: 250000baud.
  Note: This baudrate is chosen and configured for the “Serial” class using USB Programming port for every sample code. Refer to SERIAL_TRACES.begin() function to change this speed parameter.

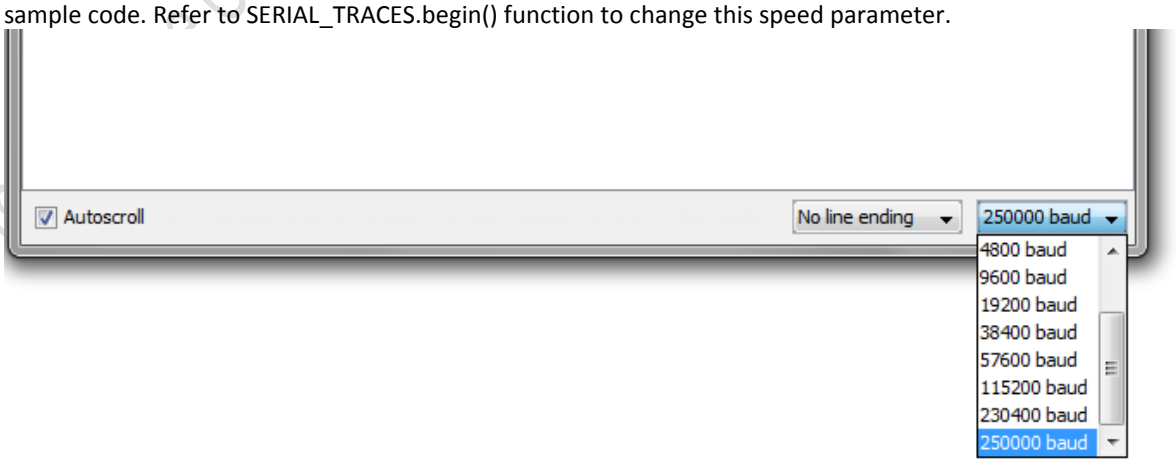

Figure 16 - Serial monitor speed configuration

Then click on the reset button of the Arduino Zero board. You can watch traces generated by sketches on the serial interface.

4. SAMPLES APPLICATIONS

4.1. ACCELEROMETER AND GYROMETER CALIBRATION

This sample code starts communication with Tungsten-eMD through Easy device library. And then starts accelerometer and gyrometer sensors at 200Hz and waits for a high accuracy value for these sensors. Once the calibration is done, c-style float tables with calibrated coefficient values are displayed on serial monitor.

At start, you'll see these messages:

```
[SKETCH] Sketch start
[SKETCH] Setup msg level info
[SKETCH] Easy device init
```

Figure 17 - Traces at start

For more information about how Easy device calls the Invensense Device Driver library and the hiding process, additional traces can be enabling by a defined option MSG_LEVEL.

At start, the accelerometer and gyrometer sensors are started, you will see data reported. You can notice for that the accuracy flag (AF) is set to '0' that means the values reported by the sensor are unreliable, calibration is needed.

```
[SKETCH] ACC : X   mg =   -19 : Y   mg =   +19 : Z   mg =  +959 AF = 0
[SKETCH] GYR : X mdps =  +125 : Y mdps =    +0 : Z mdps = +1187 AF = 0
```

Figure 18 – Non-calibrated accelerometer and gyrometer data events traces

The calibration process will be notify be the accuracy flag value. There are several states:

- '0', the sensor needs to be calibrated,
- '1', the sensor is reporting data with low accuracy,
- '2', the sensor is reporting a medium accuracy, the calibration may improve the reading values,
- '3', the sensor is reporting a high and maximum accuracy.

The gyrometer calibration is static; you need to let the board static during several seconds in order to allow the gyrometer angular calcul to define the offset values that can be adjust by the calibration.

The accelerometer calibration is also static; you need to put the board on the 3 different axes (x/y/z) during several seconds in order to get a uniformed gravity value.

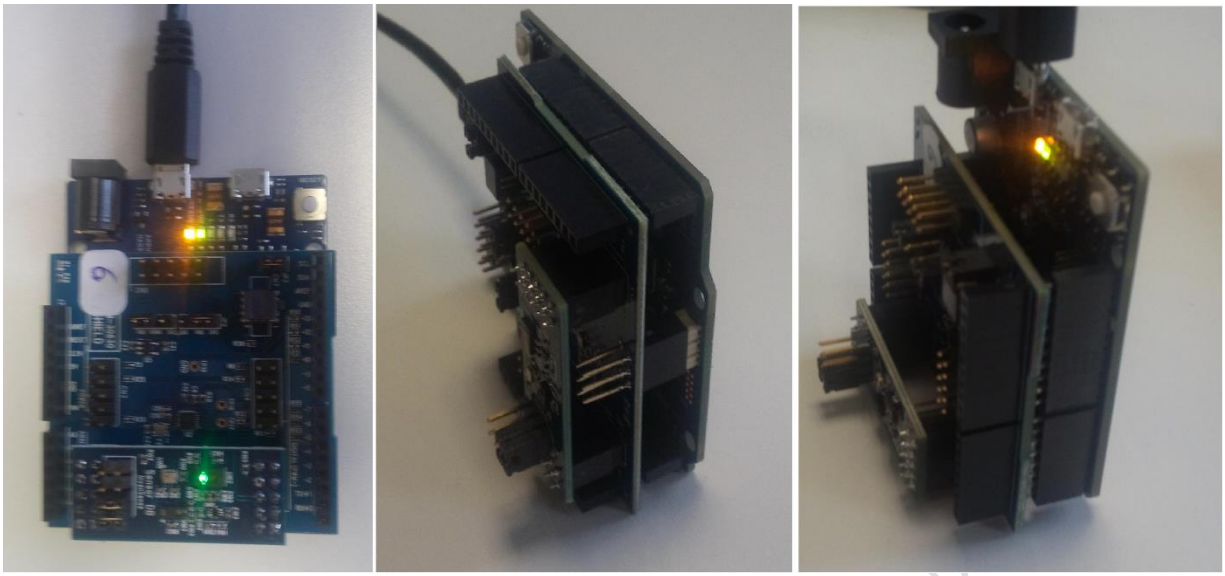

Figure 19 - Calibration process on 3 axes for accelerometer

After these procedures, the accuracy flag of the accelerometer and gyrometer sensors should be set to '3'. Then the sketch stops and you will see the calibration tables that may be used to have accurate bias. These tables could be copied/paste directly to another sketch in order to have calibrated value.

```
[SKETCH] BIAS COMPUTED : Copy/Paste this value in sketch which need this values
static float acc_bias[3] = { -0.020020, 0.004089, -0.041748 };
static float gyr_bias[3] = { -0.854492, 0.976562, 1.403809 };
[SKETCH] Sketch STOPPED !
[SKETCH] Send a char for restarting sketch
```

Figure 20 - Configuration tables traces

As displayed, the sensors can restart when sending a character on serial monitor allowing getting another calibration coefficient tables.

4.2. DOOR OPENING DETECTION

This sample code reports an event when a door is opening. It could be displayed by a LED blinking, here on Arduino Zero the LED_L will be on when the event is triggered for 5s. But if you connect a buzzer to the pin 13, you can hear a sound signal at 1 kHz tone on it. The detection is based on Game Rotation Vector (GRV) sensor that computes quaternions based on accelerometer and gyrometer sensors data. The sketch used GRV values to compute angle of move. When the angle value is more than a predefined threshold, the event occurs. Take a look into the code to easily modify the threshold defined by DOOR_OPENING_DETECTION_THRESHOLD_ANGLE.

The Easy device initialization starts the communication with the Tunsgten-eMD board and enables GRV sensor at 10Hz. In order to get accurate quaternions computed, we used Easy device library to set bias for the accelerometer and gyrometer sensors. To have an accurate detection, it's recommended to use the calibration sketch to compute the configuration coefficient tables for accelerometer and gyrometer sensors before run the opening door detection sketch (refer to Figure 20 above).

When the gesture wanted is triggered, the LED comes on and you'll see printed on serial monitor:

```
[SKETCH] DOOR OPENING DETECTED
```

Figure 21 - Door opening detection trace

4.3. GESTURE DETECTION

This sample code reports events when specific gestures are detected. For triggered a TILT gesture, you have to hold the device and have a 180 degrees move. To trigger a “SMD” gesture (**S**ignificant **M**otion **D**etected), you have to move the device at least 10s.

At start, the “Easy Device” initialization starts communication with the Tungsten-eMD board and enables events for TILT and SMD gestures. When an interrupt coming from the Tungsten-eMD is triggered, the “Easy Device” poll the data FIFO to reports the event.

After seeing the start messages:

- For a tilt gesture, put the device on your hand and then reverse the device to point towards the ground. You’ll see on serial monitor:

```
[SKETCH] Poll device
[SKETCH] Tilt gesture detected
```

Figure 22 - Tilt gesture detected traces

- For a SMD gesture, put the device on your hand and for example shake it during 10s.

```
[SKETCH] Poll device
[SKETCH] SMD gesture detected
```

Figure 23 - SMD gesture detected traces

4.4. GET ACCELEROMETER DATA

This sample code starts the Invensense Device Driver library using Easy device, enables the accelerometer sensor at 20Hz with data buffering for a maximum of 500ms. The data polling appends when the data FIFOs are full or when 500ms elapsed.

You’ll see on serial monitor for each data reported; the associated timestamp in microsecond, the event number and the acceleration value in milli-G for each axis x/y/z.

```
[SKETCH] Poll device
[SKETCH] timestamp us :   50731080  : evt cnt :       1021 : X mg =   -11 : Y mg =   -13 : Z mg =  +943
[SKETCH] timestamp us :   50780220  : evt cnt :       1022 : X mg =    -4 : Y mg =   -17 : Z mg =  +954
[SKETCH] timestamp us :   50829360  : evt cnt :       1023 : X mg =    -3 : Y mg =   -17 : Z mg =  +948
[SKETCH] timestamp us :   50878500  : evt cnt :       1024 : X mg =   -10 : Y mg =    -7 : Z mg =  +943
[SKETCH] timestamp us :   50927640  : evt cnt :       1025 : X mg =    +3 : Y mg =   -12 : Z mg =  +938
[SKETCH] timestamp us :   50976780  : evt cnt :       1026 : X mg =    -3 : Y mg =   -10 : Z mg =  +951
[SKETCH] timestamp us :   51025920  : evt cnt :       1027 : X mg =   -10 : Y mg =    -2 : Z mg =  +966
[SKETCH] timestamp us :   51075060  : evt cnt :       1028 : X mg =    -4 : Y mg =   -23 : Z mg =  +956
[SKETCH] timestamp us :   51124200  : evt cnt :       1029 : X mg =    +0 : Y mg =   -21 : Z mg =  +944
[SKETCH] timestamp us :   51173340  : evt cnt :       1030 : X mg =    +2 : Y mg =   -16 : Z mg =  +950
[SKETCH] Poll device
[SKETCH] timestamp us :   51222480  : evt cnt :       1031 : X mg =    -3 : Y mg =   -13 : Z mg =  +964
[SKETCH] timestamp us :   51271620  : evt cnt :       1032 : X mg =    -9 : Y mg =   -10 : Z mg =  +942
[SKETCH] timestamp us :   51320760  : evt cnt :       1033 : X mg =    -9 : Y mg =    -9 : Z mg =  +951
[SKETCH] timestamp us :   51369900  : evt cnt :       1034 : X mg =    -1 : Y mg =    -8 : Z mg =  +926
[SKETCH] timestamp us :   51419040  : evt cnt :       1035 : X mg =    -6 : Y mg =   -13 : Z mg =  +942
[SKETCH] timestamp us :   51468180  : evt cnt :       1036 : X mg =   -10 : Y mg =   -20 : Z mg =  +950
[SKETCH] timestamp us :   51517320  : evt cnt :       1037 : X mg =    -5 : Y mg =    -5 : Z mg =  +960
[SKETCH] timestamp us :   51566460  : evt cnt :       1038 : X mg =    +0 : Y mg =   -12 : Z mg =  +949
```

Figure 24 - Accelerometer data events reported at 20Hz traces

It’s possible to stop the accelerometer sensor by writing a character on the serial monitor, and you’ll see:

```
[SKETCH] Sketch STOPPED !
[SKETCH] Send a char for restarting sketch
```

Figure 25 - Stop sketch traces

As displayed, the accelerometer sensor can restart when sending a character on serial monitor.

4.5. GET EXTERNAL SENSORS

This sample code reports externals sensors data: Proximity, Pressure and Magnetometer. During the execution, data and traces are printed on serial monitor. As those sensors are externals, at startup the Easy device sends a ping request to each sensor to verify if they are connected to the Tungsten-eMD shield before starting the sensor.

Verify that you see on Serial Monitor:

```
[SKETCH] Ping proximity sensor
[SKETCH] Start proximity sensor
```

Figure 26 - Ping & Start Proximity sensor traces

And equivalent traces for Pressure and Magnetometer sensors.

Sensors are started at 1Hz, every second you'll see data reported on serial monitor. For each, there is an associated timestamp in microseconds. The magnetometer data are displayed in microTesla for each axis x/y/z. The pressure data are displayed in hectoPascal.

```
[SKETCH] Poll device
[SKETCH] MAG  timestamp us :    3921600  : microTesla : X = -35437 : Y = +11375 : Z = -37250
[SKETCH] PRES timestamp us :    3921870  :        994 hPa
[SKETCH] Poll device
[SKETCH] MAG  timestamp us :    4904640  : microTesla : X = -36625 : Y = +9562 : Z = -34250
[SKETCH] PRES timestamp us :    4904910  :        994 hPa
[SKETCH] Poll device
[SKETCH] MAG  timestamp us :    5887680  : microTesla : X = -36625 : Y = +10750 : Z = -35437
[SKETCH] PRES timestamp us :    5887950  :        994 hPa
[SKETCH] Poll device
[SKETCH] MAG  timestamp us :    6870720  : microTesla : X = -36000 : Y = +10750 : Z = -34250
[SKETCH] PRES timestamp us :    6870990  :        994 hPa
[SKETCH] Poll device
[SKETCH] MAG  timestamp us :    7853760  : microTesla : X = -36625 : Y = +11937 : Z = -36000
[SKETCH] PRES timestamp us :    7854030  :        994 hPa
```

Figure 27 - Magnetomter and Pressure sensor data events at 1Hz traces

The proximity sensor is an "on-change" sensor that means the data are reported only if there is a different distance to the closest surface of the sensor reported. To well see proximity data printed in serial monitor, move your finger close to the Sensor Daughter board connected to the Tungsten-eMD shield. The proximity sensor will detect distance variation and reports sensor data in millimeters.

```
[SKETCH] Poll device
[SKETCH] MAG  timestamp us :  240834210  : microTesla : X = -34812 : Y = +10750 : Z = -36625
[SKETCH] PRES timestamp us :  240834510  :        994 hPa
[SKETCH] PROX timestamp us :  240834420  :         25 mm
[SKETCH] Poll device
[SKETCH] MAG  timestamp us :  241817250  : microTesla : X = -36000 : Y = +10187 : Z = -37250
[SKETCH] PRES timestamp us :  241817550  :        994 hPa
[SKETCH] PROX timestamp us :  241817460  :         33 mm
[SKETCH] Poll device
[SKETCH] MAG  timestamp us :  242800290  : microTesla : X = -34812 : Y = +9562 : Z = -36625
[SKETCH] PRES timestamp us :  242800590  :        994 hPa
[SKETCH] PROX timestamp us :  242800500  :         37 mm
[SKETCH] Poll device
[SKETCH] MAG  timestamp us :  243783330  : microTesla : X = -36000 : Y = +11375 : Z = -35437
[SKETCH] PRES timestamp us :  243783630  :        994 hPa
[SKETCH] PROX timestamp us :  243783540  :         25 mm
```

Figure 28 - External sensors traces

4.6. GET QUATERNION

This sample code gets quaternion data printed on serial monitor. The Easy device initializes the communication with the Tungsten-eMD board and starts the Game Rotation Vector sensor at 20Hz; this sensor name is an Android name to orientation based on accelerometer and gyrometer. Data are bufferized for a maximum of 500ms. The data are reported on Tungsten-eMD interrupt when the sensor data FIFOs are full or when 500ms elapsed.

For each data reported, there are an associated timestamp, an event number and the normalized unit value for each axis x/y/z.

```
[SKETCH] Poll device
[SKETCH] timestamp us :    3473160  : evt cnt :         60 : W =  +999 X =   -11 : Y =   +22 : Z =   +33
[SKETCH] timestamp us :    3522300  : evt cnt :         61 : W =  +999 X =   -11 : Y =   +22 : Z =   +33
[SKETCH] timestamp us :    3571440  : evt cnt :         62 : W =  +999 X =   -11 : Y =   +22 : Z =   +33
[SKETCH] timestamp us :    3620580  : evt cnt :         63 : W =  +999 X =   -11 : Y =   +22 : Z =   +34
[SKETCH] timestamp us :    3668790  : evt cnt :         64 : W =  +999 X =   -11 : Y =   +22 : Z =   +35
[SKETCH] timestamp us :    3717930  : evt cnt :         65 : W =  +999 X =   -11 : Y =   +23 : Z =   +35
[SKETCH] timestamp us :    3767070  : evt cnt :         66 : W =  +998 X =   -12 : Y =   +23 : Z =   +36
[SKETCH] timestamp us :    3816210  : evt cnt :         67 : W =  +998 X =   -12 : Y =   +23 : Z =   +36
[SKETCH] timestamp us :    3865350  : evt cnt :         68 : W =  +998 X =   -12 : Y =   +23 : Z =   +37
[SKETCH] timestamp us :    3914490  : evt cnt :         69 : W =  +998 X =   -12 : Y =   +23 : Z =   +38
[SKETCH] Poll device
[SKETCH] timestamp us :    3963630  : evt cnt :         70 : W =  +998 X =   -12 : Y =   +24 : Z =   +38
[SKETCH] timestamp us :    4012770  : evt cnt :         71 : W =  +998 X =   -12 : Y =   +24 : Z =   +38
[SKETCH] timestamp us :    4061910  : evt cnt :         72 : W =  +998 X =   -12 : Y =   +24 : Z =   +39
[SKETCH] timestamp us :    4111050  : evt cnt :         73 : W =  +998 X =   -12 : Y =   +24 : Z =   +39
[SKETCH] timestamp us :    4160190  : evt cnt :         74 : W =  +998 X =   -12 : Y =   +24 : Z =   +40
[SKETCH] timestamp us :    4209330  : evt cnt :         75 : W =  +998 X =   -13 : Y =   +24 : Z =   +41
[SKETCH] timestamp us :    4257540  : evt cnt :         76 : W =  +998 X =   -13 : Y =   +25 : Z =   +41
[SKETCH] timestamp us :    4306680  : evt cnt :         77 : W =  +998 X =   -13 : Y =   +25 : Z =   +42
[SKETCH] timestamp us :    4355820  : evt cnt :         78 : W =  +998 X =   -13 : Y =   +25 : Z =   +42
[SKETCH] timestamp us :    4404960  : evt cnt :         79 : W =  +998 X =   -13 : Y =   +25 : Z =   +43
```

Figure 29 - Quaternion data events traces

The sketch allows the user to start and stop the sensor in sending a character on serial monitor, as described above in the GetAccelerometerData sample code.