InvenSense

Software User Guide

For

ICM-30630 eMD

InvenSense Inc.
1745 Technology Drive, San Jose, CA 95110 U.S.A
+1(408) 988–7339
www.invensense.com

Document Number: AN-000077
Revision: 1.0
Rev Date: 10/28/2015

1. TABLE OF CONTENTS

TABLE OF FIGURES/DIAGRAMS

ADDITIONAL 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

ADDITIONAL DOCUMENTS

ICM-30630 Datasheet
ICM-30630 System Hardware Design Guide
ICM-30630 Application Note: Using Eclipse IDE with J-Link Debugger
ICM-30630 Command Protocol and Architecture
ICM-30630 Shield Hardware Guide
SensorStudio User Documentation

1 OVERVIEW

The purpose of this document is to give an overview of the ICM-30630 embedded Developer's Kit that will allow users to create a custom sensor application. This document may also serve as a quick start guide for the ICM-30630 package and its elements, including setup and how use the sample applications provided.

1.1. INTRODUCTION

The ICM-30630 eMD solution is compatible with the Arduino Zero and is based on GCC development tools. The purpose of this solution is to allow sensor management and algorithm processing by using a standalone microcontroller. The ICM-30630 eMD solution was created as an advanced sensor hub, easy to integrate for users developing in wearable and IoT space. The Developer's Kit's includes a full sensor software solution; additional memory space is available on-chip for users to program with their own logic.

1.2. ICM-30630 EMD BASICS

1.2.1. Overview

The ICM-30630 eMD provides the user with the ICM-30630 InvenSense chip. The ICM-30630 chip uses three different processors: 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 the Android L solution. The DMP image has 6-axis (Accelerometer + Gyroscope) support including calibration.

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

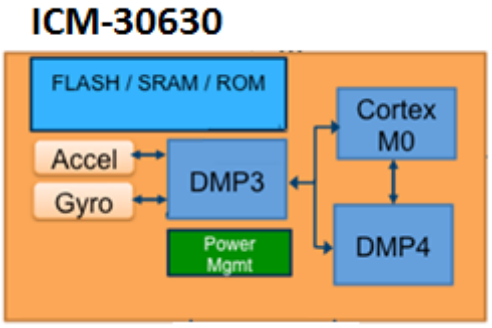

Figure 1 – ICM-30630 eMD Architecture

1.2.2. ICM-30630 eMD Hardware Layout

The Arduino Zero board communicates to the ICM-30630 through SPI, providing a fast communication pathway. Two interrupt lines are needed from the ICM-30630 to connect to a wake-up pin and a non-wake-up pin. 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. These two interrupt lines signal wake-up and non-wake-up events according to the Android L specification, and allow the application to sleep in order to save power. External sensors are available on the sensor daughter board and connect to the ICM-30630 using I2C. These sensors include, but are not limited to, magnetometer, proximity sensor, and barometer.

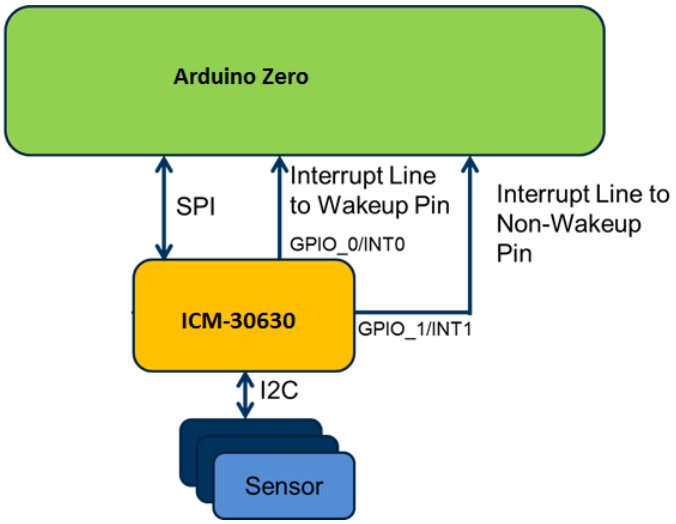

Figure 2 – Recommended Hardware Layout for ICM-30630 eMD

2. SOFTWARE DEVELOPMENT PLATFORM

2.1. SOLUTION PLATFORM

This section will describe the hardware components needed to setup the ICM-30630 eMD solution. The proposed solution platform includes:

- The **Arduino Zero Board** - Application board with a standalone microcontroller. For more information about the Arduino Zero board, please refer to Arduino website (see *Useful Links* section above.)

- The **ICM-30630 eMD Shield**- Includes the ICM-30630 with three combined processors, an accelerometer, and a gyrometer sensor. All the connectors at the back 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, not provided with the ICM-30630 eMD Developer's Kit.

Be careful of the Jumpers' configuration: they must be set as below:

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

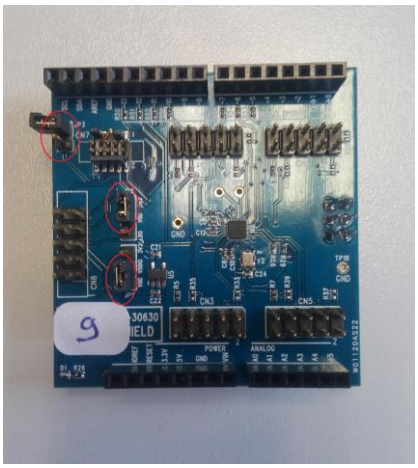

Figure 3 –ICM-30630 eMD Shield

- The **Sensor Daughter Board** is optional but adds available external sensors -- magnetometer, proximity sensor and pressure sensor. The connectors at the back of the board will be connected to ICM-30630 eMD Shield. The SD Board pin1 marked as a square pin will be connected to pin1 labelled on ICM-30630 Shield.

Jumpers' configuration:

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

Figure 4 – Sensor Daughter Board

Find below a view of the three boards connected together.

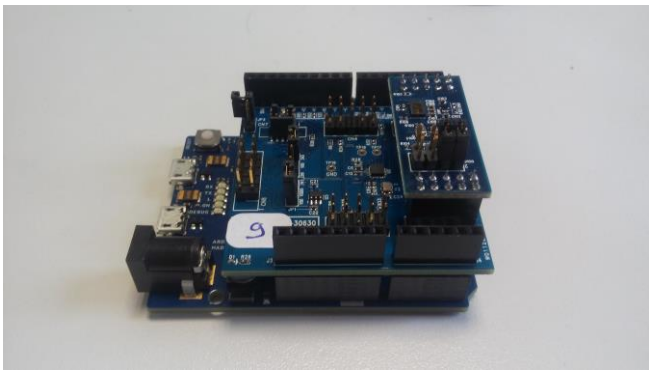

Figure 5 - Development Platform

2.2. DIRECTORY STRUCTURE

The InvenSense ICM-30630 eMD package includes all the necessary files to create a custom application. There are two parts in the installed package:

- **Examples**: Included are application sample codes that use the InvenSense library to communicate with ICM-30630 eMD. All the sample codes contain an .ino file that is compatible with Arduino IDE. (For more information about sample code, please refer to *Section 4*).

- **src**: Library sources needed to interface with ICM-30630 eMD.
  - **Bridge:** Library sources needed to encapsulate protocol to communicate with SensorStudio.
  - **INVN:** Libraries to communicate with ICM-30630 eMD include:
    - **Devices:** Contains InvenSense Device Driver library that defines API methods to control and retrieve data from InvenSense devices
    - **FifoProtocol:** Contains InvenSense FIFO Protocol to communicate with ICM-30630 eMD
    - **Utils:** Contains helper sources for these libraries
  - **Arduino Adapter:** Contains adapter layout for SPI communication between Arduino Zero and ICM-30630 eMD
  - **DeviceIoHal:** Contains HAL for SPI communication using Arduino API
  - **EasyDevice:** Contains simplified layout to access the InvenSense Device Driver library

2.3. PROJECTS

The InvenSense ICM-30630 eMD package includes projects for very basic functionalities. The basic projects listed below can be used as starting points for application developments. For more details, please refer to *Section 4*.

- The **Bridge** project is used to interface with the SensorStudio tool (please refer to *SensorStudio 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 can be used directly in other application. Please, refer to "DoorOpeningDetection" sample code (*Section 4.2*) that explains how to use calibration tables.

The "Detection" projects consist of sample codes that demonstrate how to use the ICM-30630 eMD Developer's 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, an LED blinks on Arduino Zero to inform the user.

- The **GestureDetection** project demonstrates how to detect two types of gestures – a tilting gesture or a "SMD" gesture (for Significant Motion Detection). For each event detected, a message is printed on the Arduino Serial Monitor (for more information on this monitor, please refer to *Section 3.3*).

The "GetData" project demonstrates very simple uses of the InvenSense Device Driver library to get different sensor data from the ICM-30630 eMD and the Sensor Daughter board. They can be used as references for examples to develop sensor data processing applications.

- **GetAccelerometerData** -- this sample code shows how to get calibrated accelerometer data. Data is read from the accelerometer sensor integrated in ICM-30630 eMD and printed on Arduino Serial Monitor.

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

- **GetGameRotationVector** -- this sample code shows how to get quaternion data, which represents orientation of the device based on accelerometer and gyrometer sensor data. Data is read from the sensors available on ICM-30630 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 ICM-30630 eMD Developer's 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 will be empty. But it will contain custom programs written using Arduino IDE that are called sketches and custom libraries. These sketches are saved with the specific Arduino file extension .ino.

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

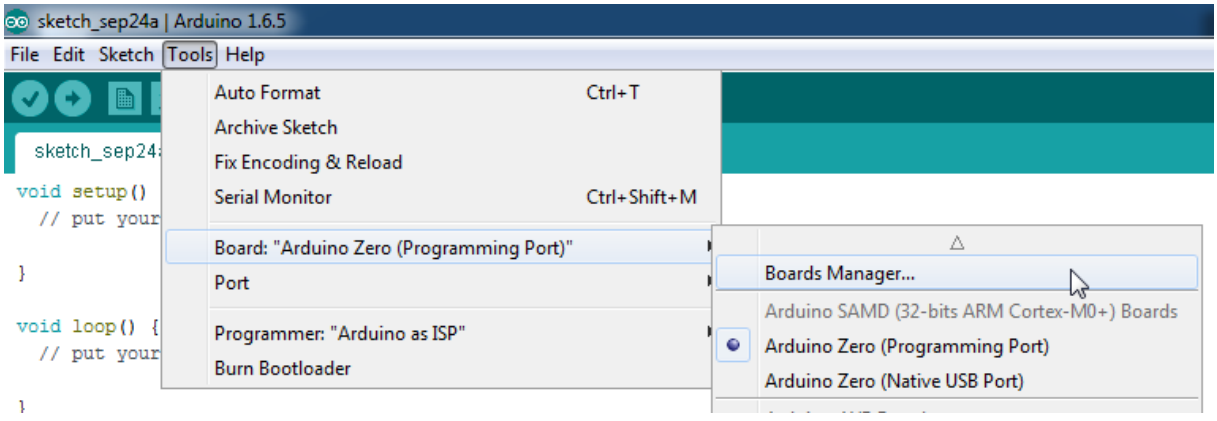

Figure 6 – Click on Board Manager

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

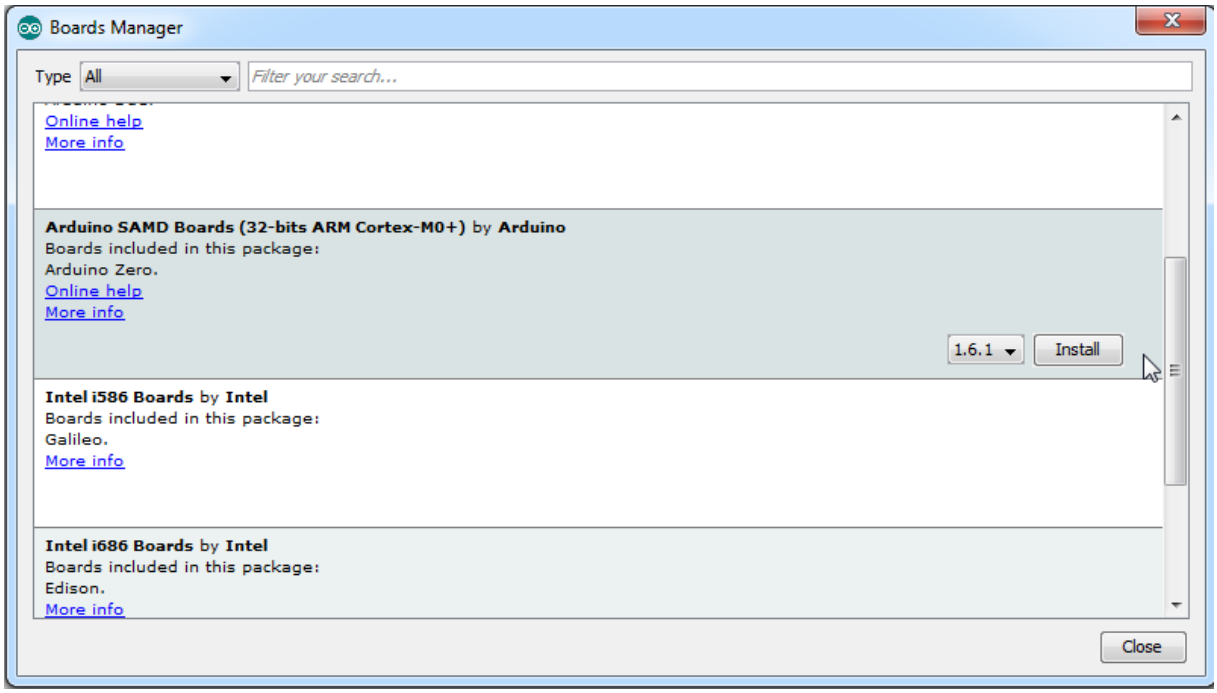

Figure 7 – Install Arduino SAMD Boards Package

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 backside of the board has labels for identification. Please refer to the *Figure 8* below.

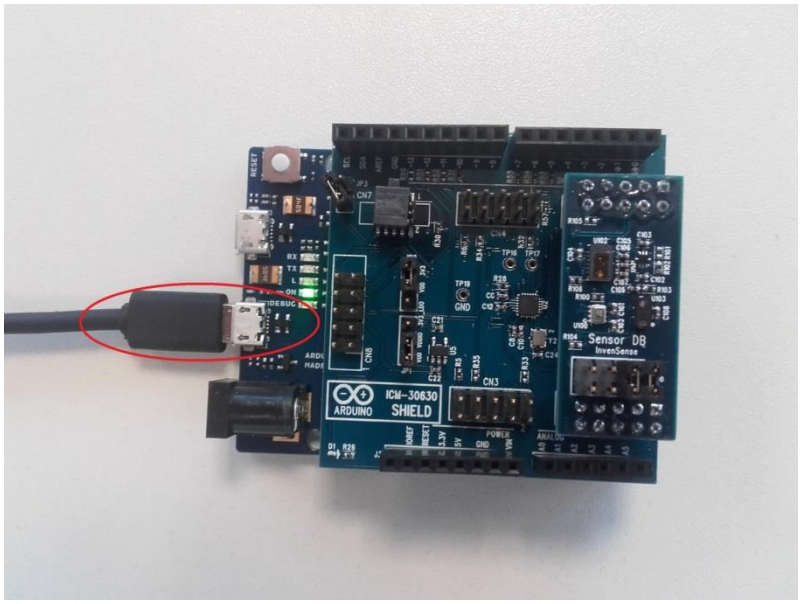

Figure 8 – Connect to PC using USB Programming Port

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

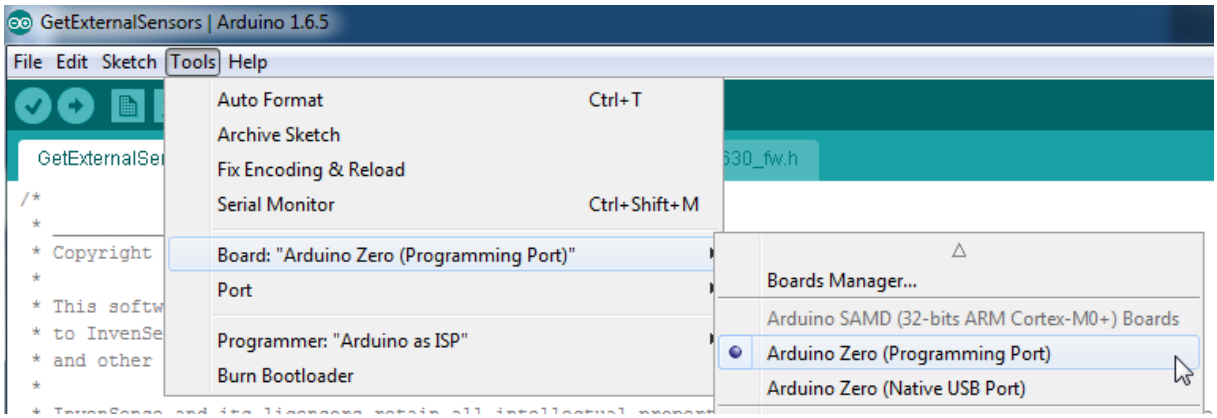

Figure 9 – Select Board: Arduino Zero (Programming Port)

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

Note: The "Programmer" setting is not used in our configuration. This setting defines the programmer type able to program without using the Arduino bootloader.

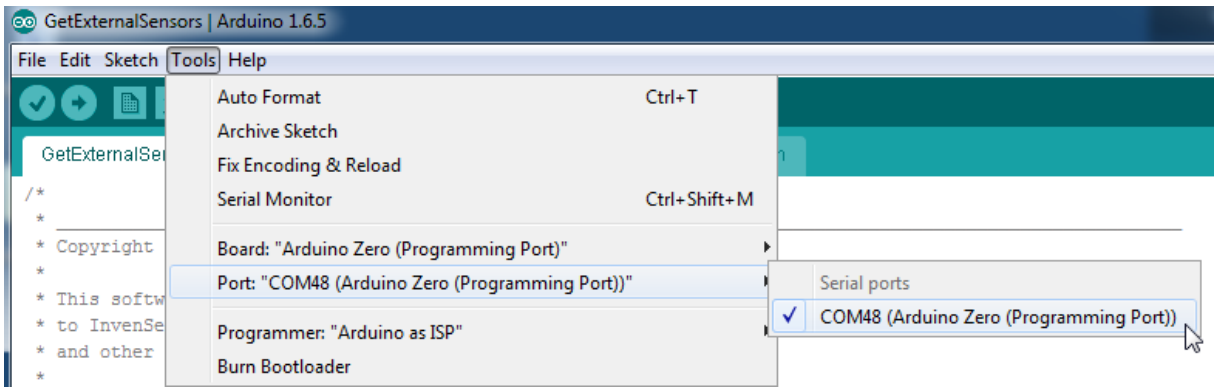

Figure 10 – Select Port: COMxx (Arduino Zero (Programming Port))

The Arduino Zero includes the Atmel Embedded Debugger (EDBG) to program the chip directly. The programmer setting (from Tools menu) allows using an alternate bootloader to flash the Arduino firmware. In our case, no additional step is required.

2.4.3. Package installation

- Install the InvenSense ICM-30630 eMD Developer's Kit. Click on Sketch->Include Library -> Add .ZIP Library…

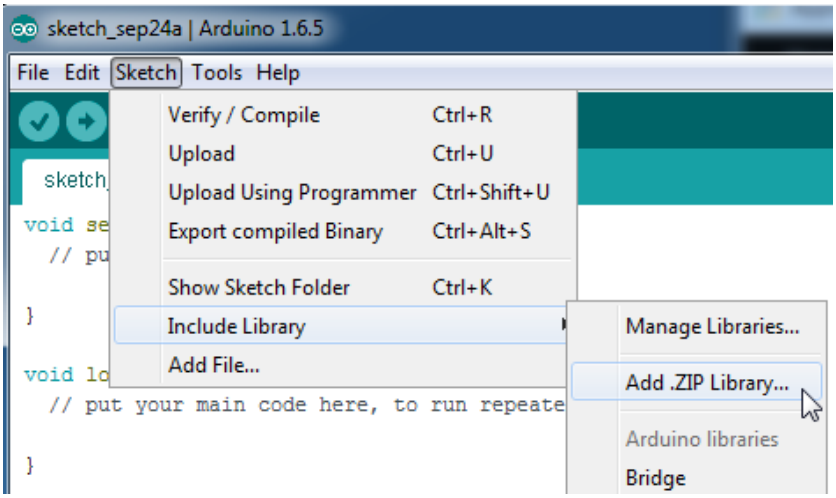

Figure 11 – Add the .ZIP Library

Select the package **Arduino_Zero_ICM-30630-eMD_xxxx.zip**, where "xxxx" is the version number. You will see this message displayed to confirm the library is correctly added to custom library.

Figure 12 – Displayed Message when Adding the InvenSense ICM-30630 Library

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

2.5. COMPILING AND DOWNLOADING

2.5.1. Compiling and downloading a sample from Arduino

In order to verify the Arduino configuration, Arduino IDE provides a set of samples (also called sketches). For example, one of these samples will blink a LED.

- Select the Blink example
  Click on File->Examples->01.Basics->Blink

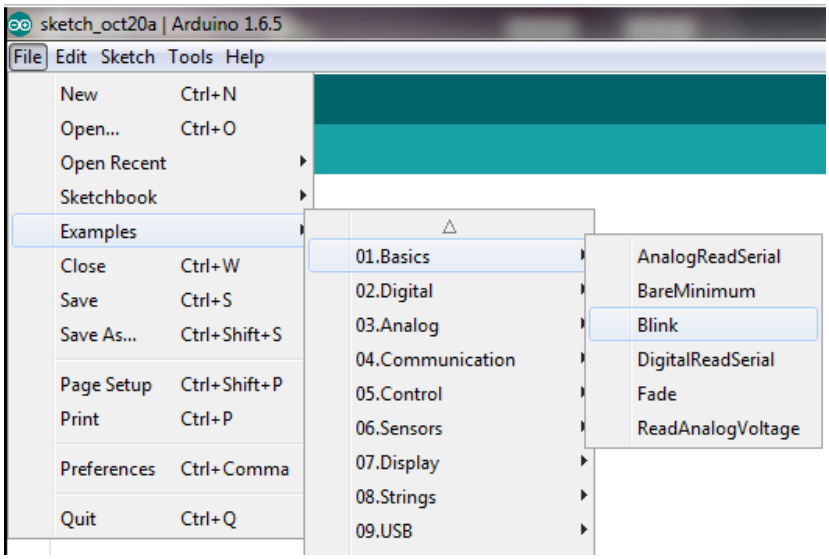

Figure 13 - Select Arduino Basic Example: Blink

- Build the sketch by clicking on “Verify” button

Figure 14 - Click on Verify

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

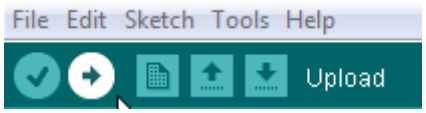

Figure 15 - Click on Upload

If the upload succeeds, you will see the yellow LED_L blinking every second.

Troubleshooting: Please refer to Arduino website https://www.arduino.cc/en/Guide/Environment.

2.5.2. Compiling and download a sample from InvenSense ICM-30630 library

- Now you can select a sample code dedicated to ICM-30630.
  Click on File->Examples->InvenSense ICM-30630 library ->”xxxxxxxxxx”

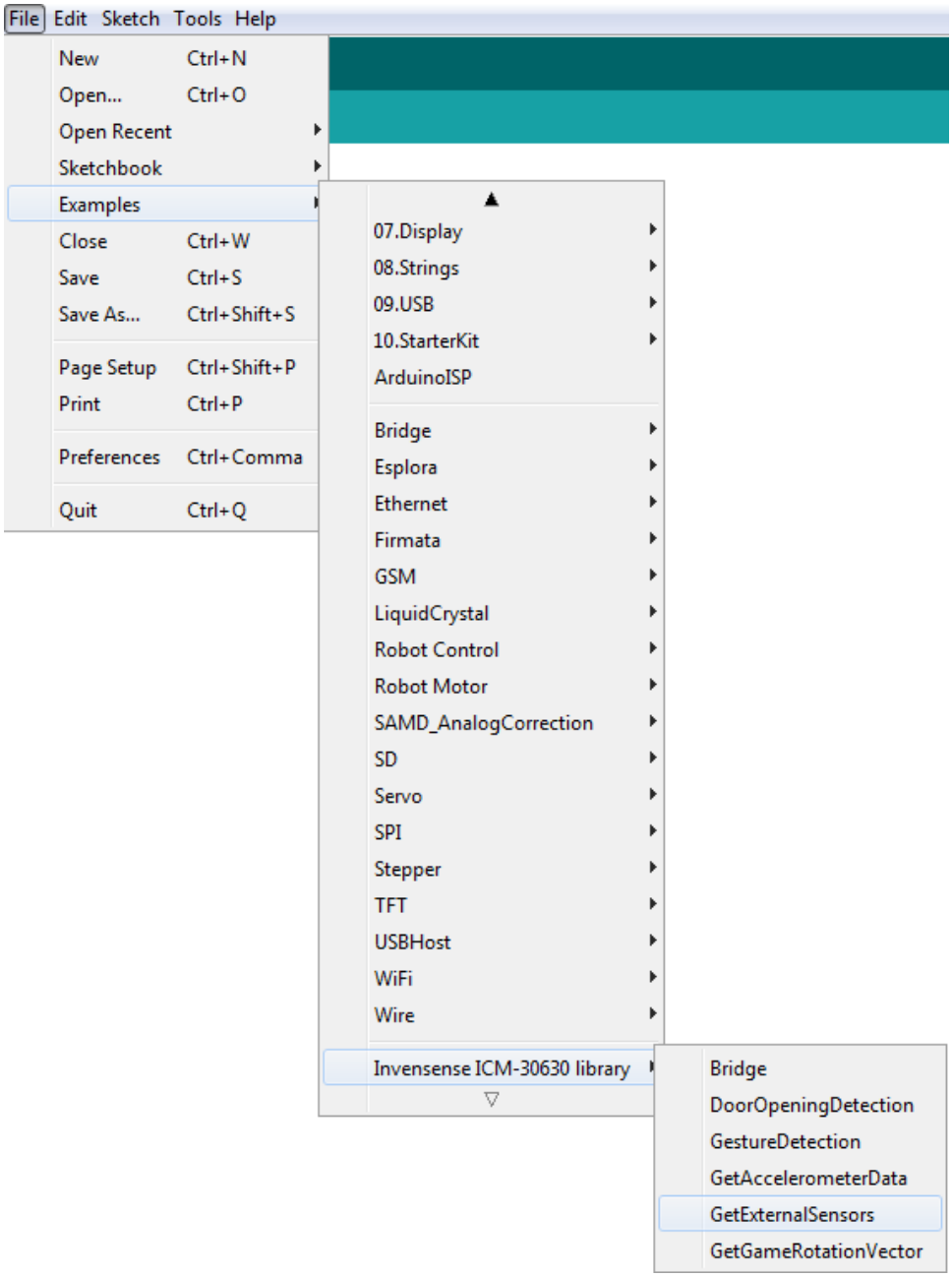

Figure 16 – InvenSense ICM-30630 library Examples Tree

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

The ‘GetExternalSensors’ sketches requires external sensors to be connected. Please make sure to have the sensor daughter-board plugged-in.

3. THE APPLICATION

This section describes the application portion of projects using Easy Device library to communicate with the ICM-30630 eMD. This architecture is common to all sample application provided by InvenSense.

3.1. START-UP IN .INO FILE

The .ino file is the starting point at runtime. There is no difference with a .cpp file and the file format and the programming language. Every project must have a .ino or sketch file that contains 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, the Easy Device library is initialized to start the communication with ICM-30630 eMD. In the Easy Device structure parameters, there are callbacks functions (refer to *Section 3.2.3*) and all the images needed for setup.

There are three major **image loading steps** needed to bring-up the ICM-30630 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

Note: The firmware programming step is optional, you can use DISABLE_FW_M0_PROG define to avoid reprogramming the ICM-30630 eMD at each Easy Device initialization.

Another requirement for the Easy Device setup is to provide a minimum of **4Kbytes buffer** needed to read ICM-30630 eMD FIFOs before parsing data read through SPI. The bigger the allocate buffer is, the faster the communication will be.

One last important step to bring-up the ICM-30630 eMD device is to **set the reference frame**. The reference frame describes the X/Y/Z axis reference and the convention sign for sensor data for the device. It depends on how the hardware sensors (accelerometer, gyrometer and magnetometer) are mounting on the board. The X/Y/Z axis reference on ICM-30630 eMD shield for accelerometer and gyrometer sensors is on the figure below:

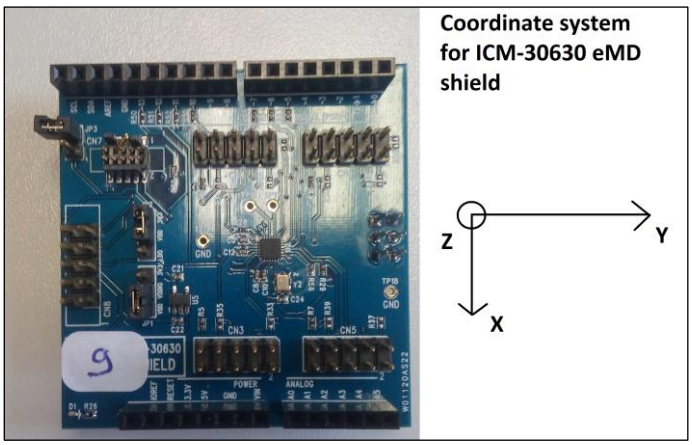

Figure 17 - Coordinate System for Reference Frame

The reference frame is computed according to the mounting matrix settings provided in the Easy Device structure. For the accelerometer and gyrometer (both being physically aligned), an identity matrix will set the default board orientation as described in Figure 17. For the external magnetometer on the Sensor Daughter-Board, the mounting matrix has to be aligned with the appropriate coefficients to match the desired orientation.

It is possible to directly start a sensor after setting up the Easy Device library by calling *inv_easy_device_start_sensor().* The user should specify the following parameters:

- The device structure, already defined for the Easy Device initialization

- The sensor ID, (the full list of sensors is available in the 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 the Easy Device library callback and calls data processing. Each data available event notification 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 callback, to notify there are data available in FIFOs,
- the 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 is received. For each sensor data the user has access to, there is 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 the InvenSense Device Driver library, see SensorType.h header
- Arg: listener context

3.3. SERIAL MONITOR

How can the user verify that their code is correctly running? The Arduino Serial interface. You may refer to the 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: the “Serial” class for the 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 below step shows how to use the serial monitor in Arduino IDE:

- Open the Arduino Serial Monitor

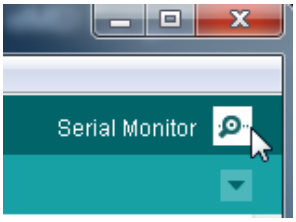

Figure 18 – Serial Monitor Interface Shortcut

- 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.

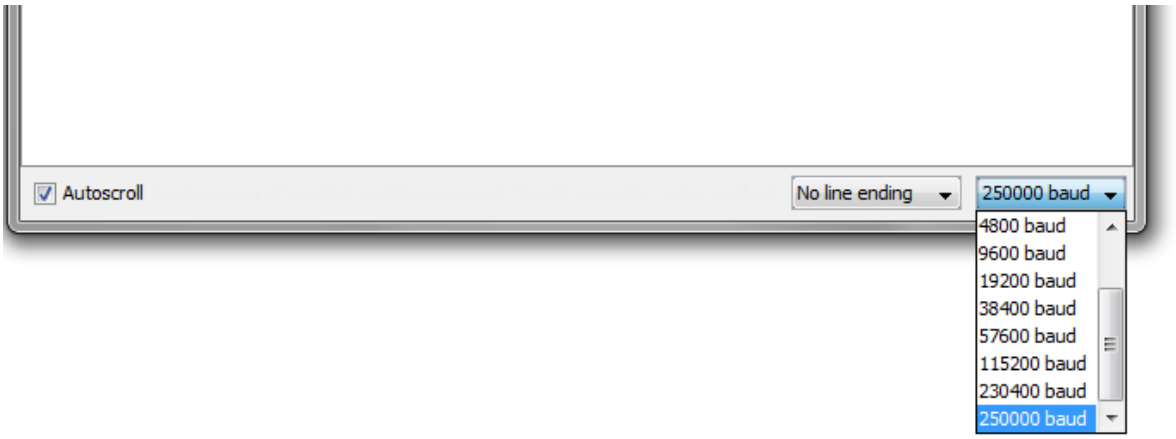

Figure 19 - Serial Monitor Speed Configuration

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 communicating with ICM-30630 eMD through the Easy Device library, starts the accelerometer and gyrometer sensors at 200H, 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, the user will see these messages:

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

Figure 20 - Traces at Start

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

When the accelerometer and gyrometer sensors start, the user will see the data reported. If the accuracy flag (AF) is set to '0', that means the values reported by the sensor are unreliable; calibration is needed.

```
[SKETCH] 0d 00h00m00s.765ms.434us : ACC : X   mg =    -3 : Y   mg =    +7 : Z   mg =  +941 AF = 0
[SKETCH] 0d 00h00m00s.763ms.390us : GYR : X mdps =  -562 : Y mdps =  -250 : Z mdps =   +62 AF = 0
```

Figure 21 – Non-Calibrated Accelerometer and Gyrometer Data Events Traces

The calibration process will notify 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 medium accuracy, the calibration may improve the reading values
- '3'-- the sensor is reporting a high and maximum accuracy

If the gyrometer calibration is static, you need to let the board static for several seconds in order to allow the gyrometer angular calcul to define the offset values that can be adjusted by the calibration.

If the accelerometer calibration is also static, you need to put the board on the 4 different axes for several seconds in order to get a uniformed gravity value.

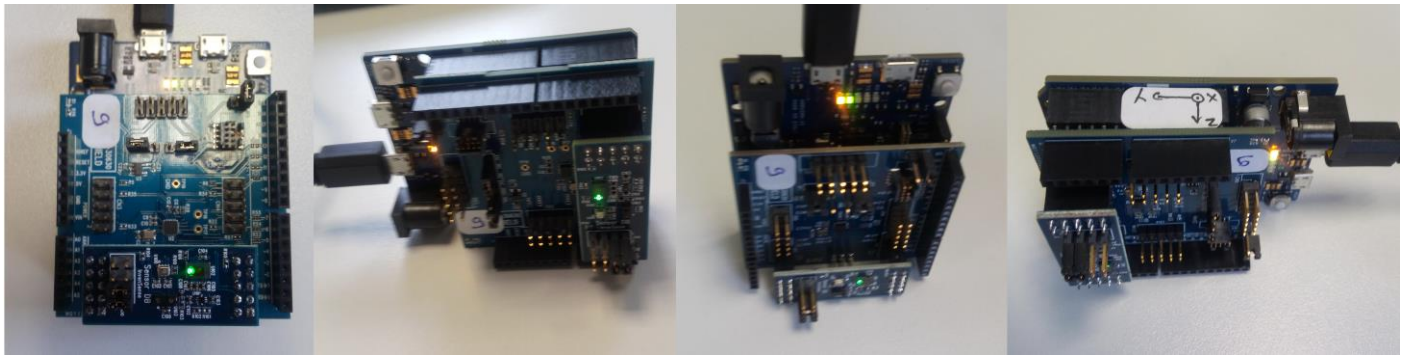

Figure 22 - Calibration Process on 4 Axes for Accelerometer

After these procedures, the accuracy flag of the accelerometer and gyrometer sensors should be set to '3', the sketch then stops and you will see the calibration tables that are used to get the accurate bias. These tables could be copied and pasted 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 23 - Configuration Tables Traces

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

4.2. DOOR OPENING DETECTION

This sample code reports an event when a door is opening, displayed by an LED blinking. On Arduino Zero, the LED_L will be on when the event is triggered for 5s, but if you connect a buzzer to Pin 13, you will hear a sound signal at 1 kHz tone. The detection is based on the custom sensor generated from SensorStudio Flow "Door opening detection". First, you must program the ICM-30630 eMD board with this Studio flow. Refer to SensorStudio documentation for more information about programming the ICM-30630 eMD through Studio flow. SensorStudio updates the firmware image and when starting the Door Opening Detection Arduino sketch, the ICM-30630 firmware image is not re-programmed.

```
[SKETCH] Sketch start
[SKETCH] Setup msg level as warning
[SKETCH] Easy device init
[SKETCH] Ping custom sensor
[SKETCH] Start custom sensor
[SKETCH] Door open detected
[SKETCH] Door close detected
[SKETCH] Door open detected
```

Figure 24 – Door Opening Detection Sketch

The Easy Device initialization starts the communication with the ICM-30630 eMD board and enables custom sensor 0 at 10Hz. In order to get accurate detection of angle 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 the accelerometer and gyrometer sensors before running the opening door detection sketch (refer to Figure 22 above).

When the gesture wanted is triggered, the LED turns on and you will see the following message printed on serial monitor:

```
[SKETCH] Door open detected
```

Figure 25 - Door Opening Detection Trace

4.3. GESTURE DETECTION

This sample code reports events when specific gestures are detected. For a triggered TILT gesture, you have to hold the device and move 180 degrees. 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 ICM-30630 eMD board and enables events for TILT and SMD gestures. When an interrupt coming from the ICM-30630 eMD is triggered, the "Easy Device" polls the data FIFO to report 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 26 - 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 27 - SMD Gesture Detected Traces

4.4. GET ACCELEROMETER DATA

This sample code starts the InvenSense Device Driver library using Easy Device, enabling 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 will see on the serial monitor for each data reported the associated timestamp, the event number, and the acceleration value in milli-G for each axis X/Y/Z.

```
[SKETCH] Poll device
[SKETCH] 0d 00h00m34s.560ms.416us : evt cnt :        680 : X mg =   -11 : Y mg =    -5 : Z mg =  +952
[SKETCH] 0d 00h00m34s.610ms.391us : evt cnt :        681 : X mg =   -13 : Y mg =    +2 : Z mg =  +942
[SKETCH] 0d 00h00m34s.660ms.366us : evt cnt :        682 : X mg =   -14 : Y mg =    -4 : Z mg =  +962
[SKETCH] 0d 00h00m34s.710ms.342us : evt cnt :        683 : X mg =   -18 : Y mg =    +7 : Z mg =  +948
[SKETCH] 0d 00h00m34s.760ms.317us : evt cnt :        684 : X mg =   -19 : Y mg =    -9 : Z mg =  +957
[SKETCH] 0d 00h00m34s.810ms.292us : evt cnt :        685 : X mg =   -20 : Y mg =    -9 : Z mg =  +949
[SKETCH] 0d 00h00m34s.860ms.268us : evt cnt :        686 : X mg =    -9 : Y mg =   -16 : Z mg =  +951
[SKETCH] 0d 00h00m34s.910ms.243us : evt cnt :        687 : X mg =   -14 : Y mg =    -8 : Z mg =  +949
[SKETCH] 0d 00h00m34s.960ms.219us : evt cnt :        688 : X mg =   -13 : Y mg =    -4 : Z mg =  +960
[SKETCH] 0d 00h00m35s.010ms.194us : evt cnt :        689 : X mg =   -39 : Y mg =    -1 : Z mg =  +945
[SKETCH] 0d 00h00m35s.060ms.169us : evt cnt :        690 : X mg =    -3 : Y mg =    -3 : Z mg =  +939
[SKETCH] Poll device
[SKETCH] 0d 00h00m35s.110ms.145us : evt cnt :        691 : X mg =    -6 : Y mg =    -8 : Z mg =  +941
[SKETCH] 0d 00h00m35s.160ms.120us : evt cnt :        692 : X mg =   -26 : Y mg =    +1 : Z mg =  +954
[SKETCH] 0d 00h00m35s.210ms.096us : evt cnt :        693 : X mg =    +0 : Y mg =   -20 : Z mg =  +929
[SKETCH] 0d 00h00m35s.260ms.071us : evt cnt :        694 : X mg =   -17 : Y mg =    -1 : Z mg =  +961
[SKETCH] 0d 00h00m35s.310ms.046us : evt cnt :        695 : X mg =   -23 : Y mg =    -5 : Z mg =  +961
[SKETCH] 0d 00h00m35s.360ms.022us : evt cnt :        696 : X mg =   -17 : Y mg =    +0 : Z mg =  +937
[SKETCH] 0d 00h00m35s.409ms.997us : evt cnt :        697 : X mg =   -27 : Y mg =    +4 : Z mg =  +949
[SKETCH] 0d 00h00m35s.459ms.972us : evt cnt :        698 : X mg =   -12 : Y mg =    -3 : Z mg =  +936
[SKETCH] 0d 00h00m35s.509ms.948us : evt cnt :        699 : X mg =    -8 : Y mg =    +0 : Z mg =  +946
[SKETCH] 0d 00h00m35s.559ms.923us : evt cnt :        700 : X mg =   -25 : Y mg =   -11 : Z mg =  +958
```

Figure 28 - Accelerometer Data Events Reported at 20Hz Traces

It is possible to stop the accelerometer sensor by writing a character on the serial monitor:

```
[SKETCH] Sensor STOPPED !
[SKETCH] Send a char for restarting sensor
```

Figure 29 - 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 the 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 ICM-30630 eMD shield before starting the sensor.

Verify that you see Serial Monitor:

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

Figure 30 - 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. The magnetometer data is displayed in microTesla for each axis X/Y/Z. The pressure data is displayed in hectoPascal.

```
[SKETCH] Poll device
[SKETCH] MAG  0d 00h03m30s.936ms.041us : microTesla : X = -12000 : Y = -34812 : Z = -34250
[SKETCH] PRES 0d 00h03m30s.936ms.376us :        992 hPa
[SKETCH] Poll device
[SKETCH] MAG  0d 00h03m31s.935ms.792us : microTesla : X = -11437 : Y = -33625 : Z = -34250
[SKETCH] PRES 0d 00h03m31s.936ms.128us :        992 hPa
[SKETCH] Poll device
[SKETCH] MAG  0d 00h03m32s.935ms.544us : microTesla : X = -12000 : Y = -36000 : Z = -34812
[SKETCH] PRES 0d 00h03m32s.935ms.880us :        992 hPa
[SKETCH] Poll device
[SKETCH] MAG  0d 00h03m33s.935ms.296us : microTesla : X = -12625 : Y = -35437 : Z = -35437
[SKETCH] PRES 0d 00h03m33s.935ms.631us :        992 hPa
```

Figure 31 - Magnetometer and Pressure Sensor Data Events at 1Hz Traces

The proximity sensor is an "on-change" sensor that means the data is reported only if there is a different distance to the closest surface of the sensor reported. To see proximity data printed in the serial monitor, move your finger close to the Sensor

Daughter board connected to the ICM-30630 eMD shield. The proximity sensor will detect distance variation and reports sensor data in millimeters.

```
[SKETCH] Poll device
[SKETCH] MAG  0d 00h00m09s.985ms.923us : microTesla : X = -13250 : Y = -34812 : Z = -34250
[SKETCH] PRES 0d 00h00m09s.986ms.258us :        992 hPa
[SKETCH] PROX 0d 00h00m09s.986ms.289us :         10 mm
[SKETCH] Poll device
[SKETCH] MAG  0d 00h00m10s.985ms.674us : microTesla : X = -10812 : Y = -35437 : Z = -34812
[SKETCH] PRES 0d 00h00m10s.986ms.040us :        992 hPa
[SKETCH] PROX 0d 00h00m10s.986ms.040us :         16 mm
[SKETCH] Poll device
[SKETCH] MAG  0d 00h00m11s.985ms.426us : microTesla : X = -12000 : Y = -34250 : Z = -36000
[SKETCH] PRES 0d 00h00m11s.985ms.761us :        992 hPa
[SKETCH] PROX 0d 00h00m11s.985ms.792us :         11 mm
[SKETCH] Poll device
[SKETCH] MAG  0d 00h00m12s.985ms.178us : microTesla : X = -12000 : Y = -34812 : Z = -33625
[SKETCH] PRES 0d 00h00m12s.985ms.513us :        992 hPa
[SKETCH] PROX 0d 00h00m12s.985ms.544us :         30 mm
```

Figure 32 - External Sensors Traces

4.6. GET GAME ROTATION VECTOR

This sample code gets quaternion data printed on serial monitor. The Easy Device initializes the communication with the ICM-30630 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 buffered for a maximum of 500ms. The data is reported on ICM-30630 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] 0d 00h00m20s.019ms.014us : evt cnt :       391 : W = 0.929044 : X = -0.353061 : Y = 0.108841 : Z = -0.019453
[SKETCH] 0d 00h00m20s.068ms.989us : evt cnt :       392 : W = 0.929044 : X = -0.353061 : Y = 0.108841 : Z = -0.019453
[SKETCH] 0d 00h00m20s.118ms.965us : evt cnt :       393 : W = 0.929055 : X = -0.353021 : Y = 0.108896 : Z = -0.019320
[SKETCH] 0d 00h00m20s.168ms.940us : evt cnt :       394 : W = 0.929070 : X = -0.352999 : Y = 0.108843 : Z = -0.019336
[SKETCH] 0d 00h00m20s.218ms.915us : evt cnt :       395 : W = 0.929191 : X = -0.352694 : Y = 0.108793 : Z = -0.019368
[SKETCH] 0d 00h00m20s.268ms.891us : evt cnt :       396 : W = 0.929197 : X = -0.352675 : Y = 0.108801 : Z = -0.019404
[SKETCH] 0d 00h00m20s.317ms.981us : evt cnt :       397 : W = 0.929217 : X = -0.352646 : Y = 0.108722 : Z = -0.019397
[SKETCH] 0d 00h00m20s.367ms.957us : evt cnt :       398 : W = 0.929198 : X = -0.352705 : Y = 0.108680 : Z = -0.019472
[SKETCH] 0d 00h00m20s.417ms.932us : evt cnt :       399 : W = 0.929257 : X = -0.352535 : Y = 0.108721 : Z = -0.019512
[SKETCH] 0d 00h00m20s.467ms.908us : evt cnt :       400 : W = 0.929312 : X = -0.352405 : Y = 0.108665 : Z = -0.019532
[SKETCH] Poll device
[SKETCH] 0d 00h00m20s.517ms.883us : evt cnt :       401 : W = 0.929329 : X = -0.352374 : Y = 0.108627 : Z = -0.019489
[SKETCH] 0d 00h00m20s.567ms.858us : evt cnt :       402 : W = 0.929317 : X = -0.352378 : Y = 0.108721 : Z = -0.019441
[SKETCH] 0d 00h00m20s.617ms.834us : evt cnt :       403 : W = 0.929255 : X = -0.352558 : Y = 0.108687 : Z = -0.019349
[SKETCH] 0d 00h00m20s.667ms.809us : evt cnt :       404 : W = 0.929234 : X = -0.352639 : Y = 0.108603 : Z = -0.019355
[SKETCH] 0d 00h00m20s.717ms.784us : evt cnt :       405 : W = 0.929222 : X = -0.352709 : Y = 0.108472 : Z = -0.019385
[SKETCH] 0d 00h00m20s.767ms.760us : evt cnt :       406 : W = 0.929206 : X = -0.352781 : Y = 0.108415 : Z = -0.019189
[SKETCH] 0d 00h00m20s.817ms.735us : evt cnt :       407 : W = 0.929194 : X = -0.352837 : Y = 0.108346 : Z = -0.019130
[SKETCH] 0d 00h00m20s.867ms.711us : evt cnt :       408 : W = 0.929111 : X = -0.353057 : Y = 0.108329 : Z = -0.019189
[SKETCH] 0d 00h00m20s.916ms.954us : evt cnt :       409 : W = 0.929166 : X = -0.352935 : Y = 0.108261 : Z = -0.019167
[SKETCH] 0d 00h00m20s.966ms.929us : evt cnt :       410 : W = 0.929106 : X = -0.353121 : Y = 0.108155 : Z = -0.019233
```

Figure 33 - Quaternion Data Events Traces

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

5. REVISION HISTORY

REVISION DATE	REVISION	DESCRIPTION
10/28/2015	1.0	Initial Release

InvenSense