InvenSense

Quick Start Guide

For

Generic Sensor Hub

Powered by icm-20690

1 TABLE OF CONTENTS

USEFUL LINKS

InvenSense website:

http://www.InvenSense.com/

ST website:

http://www.st.com/web/catalog/tools/FM116/SC959/SS1532/LN1847?sc=stm32nucleo

http://www.st.com/web/en/catalog/tools/PF258168

Eclipse website:

https://eclipse.org/downloads/packages/eclipse-ide-cc-developers/mars2

https://sourceforge.net/projects/gnuarmeclipse/

FreeRTOS website

https://sourceforge.net/projects/freertos/files/FreeRTOS/V7.6.0/

1 OVERVIEW

The purpose of this document is to give an overview of the various steps required to run the Generic Sensor Hub (GSH) firmware. This document covers:

- How to setup the hardware for the GSH (running on Nucleo)
- How to build and flash the GSH firmware
- How to debug with Eclipse
- How to use sensor-cli application.

1.1 INTRODUCTION

The purpose of GSH is to have a complete sensor hub solution portable on any MCU. The code is provided as open-source with motion algorithms only delivered as libraries. The proposed implementation supports the ST Nucleo board (embedding a STM32F411RE) with an ICM-20690 MEMS sensor connected. The ICM-20690 MEMS is a 6-axis combo (accelerometer and gyroscope).

1.2 ICM-20690 BASICS

The ICM-20690 is accessible through SPI or I2C.

A secondary I2C (master) is also available in order to connect external sensors to the chip and a secondary SPI (slave) is used to output data for OIS (Optical Image Stabilization). An FSYNC pin is also available for EIS (Electronical Image Stabilization) use case.

These additional capabilities will not be showcased within the GSH implementation

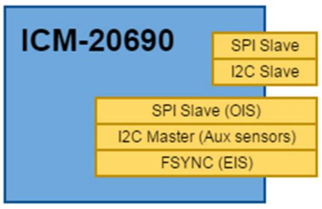


Figure 1 – ICM-20690 Architecture

2 HARDWARE PLATFORM

The SH reference kit consists of of the following components:

- ST Nucleo F411-RE board
- InvenSense Nucleo Carrier Board
- ICM-20690 6-axis sensors
- *Optionally, sensor Daughter Boards for AK09911 magnetometer*

2.1 ST NUCLEO F411-RE SETUP

The **ST Nucleo F411-RE** includes a STM32F411 microcontroller. For more information about the ST Nucleo board, please refer to ST website (see *Useful Links* section above.)

You will find the ST Link drivers to install on your PC here: http://www2.st.com/content/st_com/en/products/embedded-software/development-tool-software/stsw-link009.html

Required jumper configuration for NUCLEO is as follows:

JP1	Open
JP5 (PWR)	(U5V)
JP6 (IDD)	Closed
CN2	Closed - on (NUCLEO)

Table 1. ST Nucleo jumper configuration

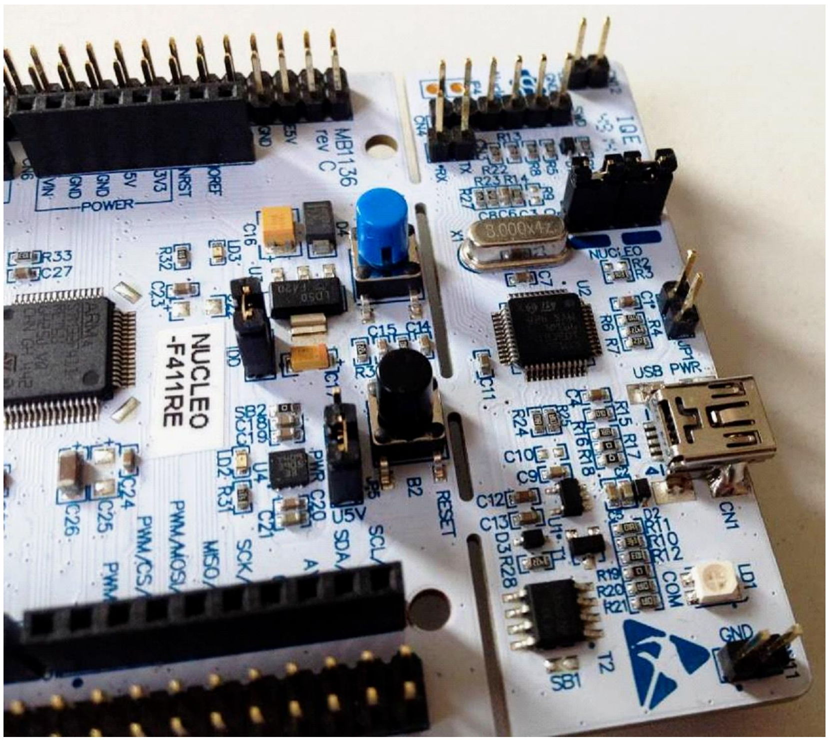


Figure 2 – ST Nucleo board jumper configuration

2.2 NUCLEO CARRIER BOARD SETUP

The **Nucleo Carrier Board** offers a convenient way to connect ST Nucleo and ICM-20690 sensor daughter boards together.

Please use the following jumper configurations:

JP1	(VDD=VDDIO)
JP2	(VDD=3V3)
JP8	(Nucleo)

Table 2. Nucleo Carrier board jumper configuration

Warning	The ST Nucleo pin headers have to match the Nucleo Carrier board ones. Please verify that the two boards are correctly connected (pin1 aligned).

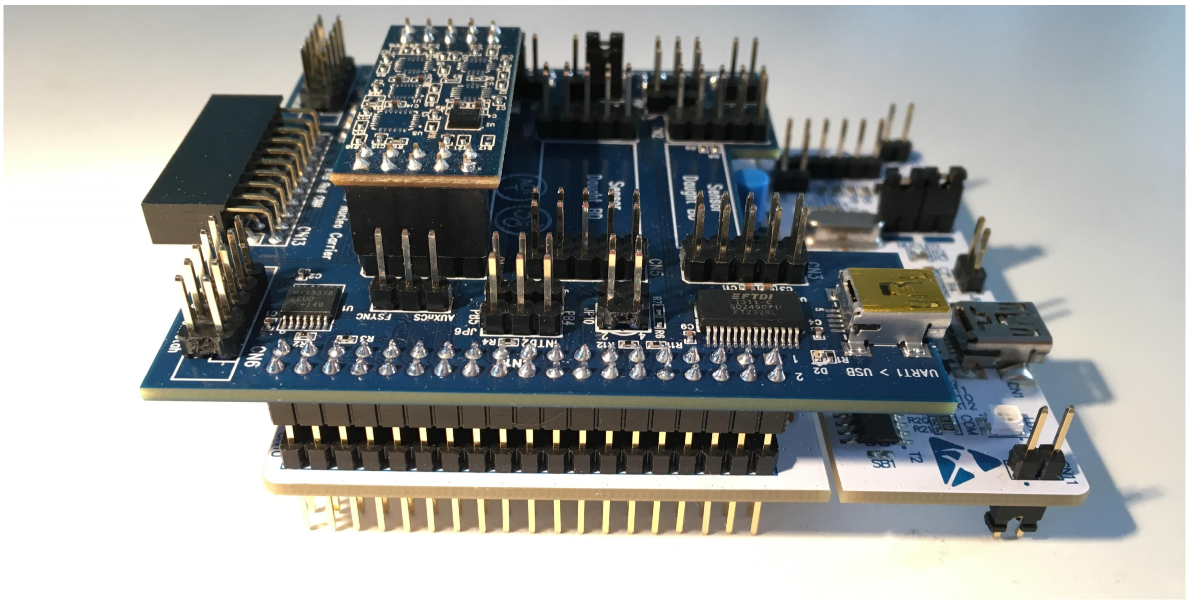


Figure 3- Nucleo Carrier Board rev C Setup

The Nucleo Carrier Board has an additional jumper JP10 needed to multiplex the SPI and I2C output lines to communicate to ICM devices. This Quick Start Guide only covers the SPI configuration. Jumper on JP10 should remain open.

JP10	Open (1!=3)
	Open (2!=4)

Table 3. SPI/I2C Nucleo Carrier board jumper configuration

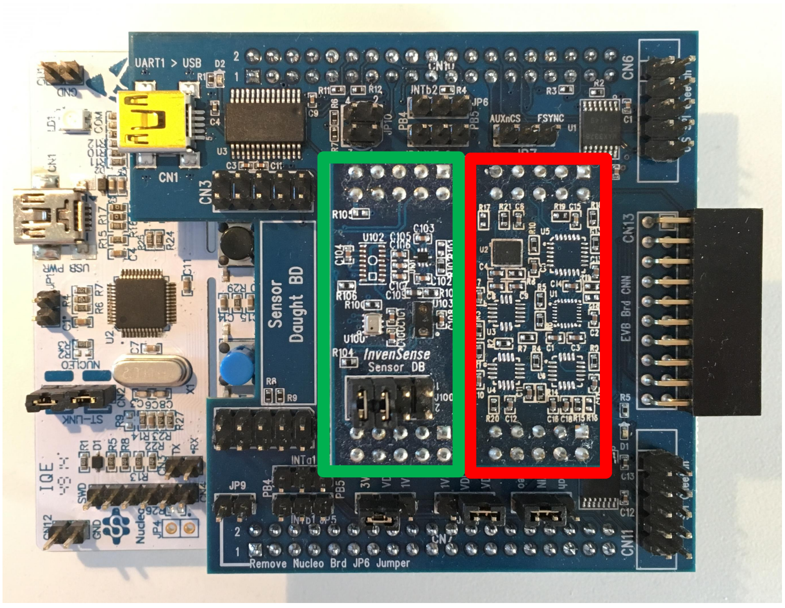
2.2.1 Pluging-in Sensor Daughter Boards

In order to connect the **ICM-20690** daughter board to the SPI bus, plug it into the slot labelled 'INV Sensor DB'.

To use AK09911, BMP280 and VCNL4040 as auxiliary sensors, you can plug in a **Sensor Daughter Board** into the slot labelled "Sensor Daught DB".

Warning	The jumpers on Sensor Daughter Board has to be placed on 5-6 and 7-8 pins

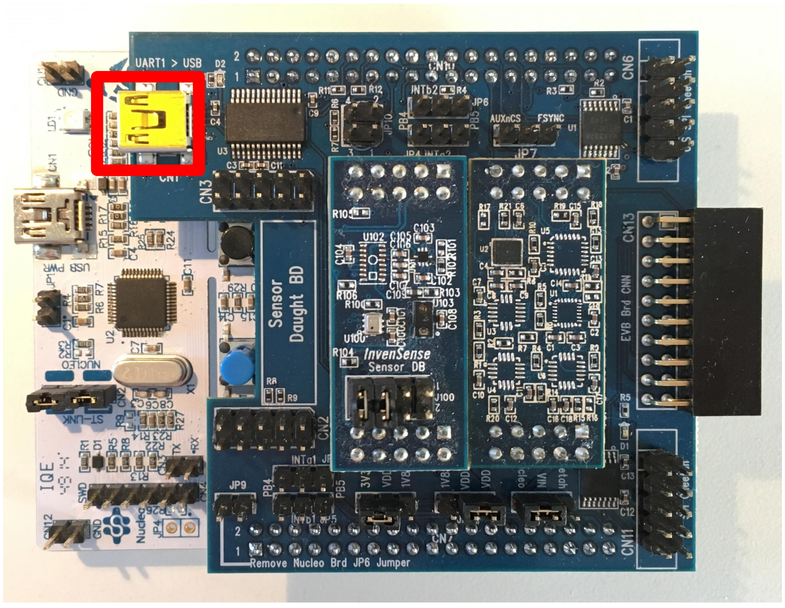
J100	[1 :2] Open
	[3 :4] Open
	[5 : 6] Close
	[7 : 8] Close



2.2.2 Connect Nucleo to computer through ST-Link

Connect the Nucleo to a PC using a mini-USB cable (ST-Nucleo CN1: ST-Link).The ST-Link allows you to:

- Power-up the platform
- Flash the firmware and debug it
- Display some traces from the firmware (sensors data will not be printed here, only debug traces)

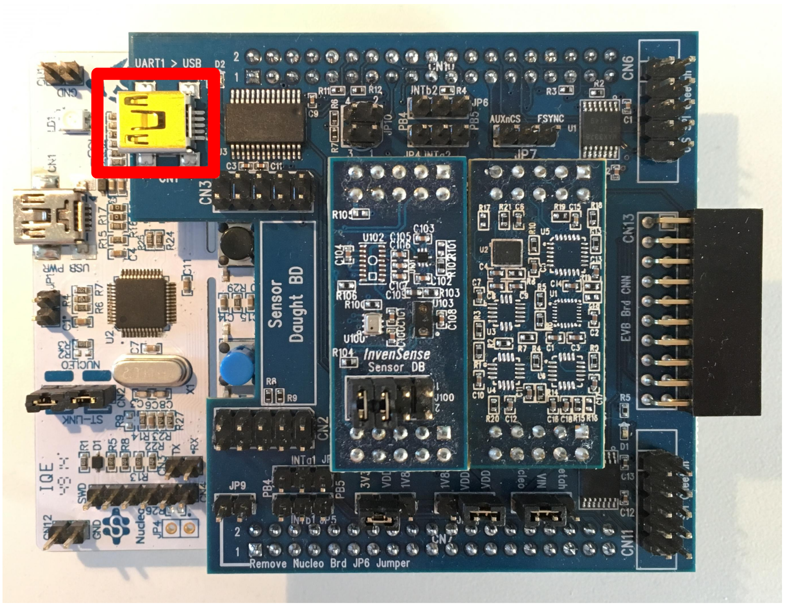


Warning	This USB port is only one capable to provide power to the Nucleo, you must keep it plugged-in at all time.

2.2.3 Connect the GSH to the PC

In order to control the firmware and to retrieve data, you will have to connect your computer to the on-board FTDI UART/USB converter (CON1 on the Carrier board).

This converter requires the following drivers to operate properly: http://www.ftdichip.com/Drivers/VCP.htm



3 SOFTWARE ENVIRONMENT

3.1 PREREQUISITE

To build and use sample applications provided as part of the Generic Sensor Hub packages, the following 3rd parties software are required:

- Eclipse IDE: https://eclipse.org/downloads/packages/eclipse-ide-cc-developers/mars2 (see appendix A for installation steps and usage)
    - Used to build and debug provided FW application
- A RS232 terminal emulator (such as Putty: http://www.putty.org/)
    - Used to retrieve traces from provided FW application
- ST Link utilities: http://www.st.com/web/en/catalog/tools/PF258168
    - Used to load FW binaries and to access the USB VCOM port of the NUCLEO

3.2 PACKAGE CONTENT

- **bin/**
    - **linaro-cm4-nucleo/** : Contains prebuilt firmware and prebuilt library (closed source)
- **config/**
    - **linaro-cm4-nucleo/** : Contains configuration file for linaro toolchain
        - **eclipse/**
            - **template/** : Contains a templated version of the Eclipse project (can be re-used to build your own project)
            - **.cprojet** : Ready to use Eclipse project
            - **.projet** : Ready to use Eclipse project
            - ***.launch** : Ready to use Eclipse project
            - **Makefile** : Ready to use Makefile
        - **main.mk** : Main makefile used to build the firmware
- **doc/** : Contains documentation
- **sources/** : Sources of the GSH
- **tools/** : Helpful tools and dll running under windows only
- **README.md**

3.3 AVAILABLE SENSORS

The following features are currently supported:
Please note that below frequencies are internal frequency and can be modified by the decimation mode. Please refer to the Software Guide for more information.

Sensor	Reporting Frequencies (Hz)		Reporting mode	Required Frequencies (Hz)	
	min	max		Accel	Gyro
Raw Accelerometer	4	1000	*Continuous*	=	x
Accelerometer	4	1000	*Continuous*	=	x
Raw Gyroscope	4	1000	*Continuous*	x	=
Gyroscope	4	1000	*Continuous*	x	=
Uncal Gyroscope	4	1000	*Continuous*	x	=
Raw Temperature	4	1000	*Continuous*	x	x
Raw Magnetometer	4	100	*Continuous*	x	x
Magnetometer	4	100	*Continuous*	x	x
Uncal Magnetometer	4	100	*Continuous*	x	x
Gravity	50	1000	*Continuous*	=	=
Linear Acceleration	50	1000	*Continuous*	=	=
Game Rotation Vector	50	1000	*Continuous*	=	=
Geomag Rotation Vector	4	100	*Continuous*	=	x
Rotation Vector	50	1000	*Continuous*	=	=
Orientation	50	1000	*Continuous*	=	=
SMD			*One shot*	50	x
Step Detector			*One shot*	50	x
Step counter			*One shot*	50	x
Tilt			*One shot*	50	x
Pickup			*One shot*	56.25	x
BAC			*One shot*	50	x
Light (VCNL4040)	1	8	*Continuous*	x	x
Proximity (VCNL4040)	1	8	*Continuous*	x	x
Pressure (BMP280)	1	16	*Continuous*	x	x

'=' means that the frequency will be the same as the corresponding sensor.
'x' means that it doesn't use it.

4 FREERTOS INSTALLATION

The GSH firmware is based on FreeRTOS 7.6.0 and the operating system is not provided in the SDK package. Please download FreeRTOSV7.6.0.zip from https://sourceforge.net/projects/freertos/files/FreeRTOS/V7.6.0/ .

Once the archive is downloaded, unzip it. Inside you will find two folders, copy the folder FreeRTOS in the folder `C:\invn\firmware\gsh\sdk\nucleo\win32\1.1.0\sources` in SDK package.

In `\invn\firmware\gsh\sdk\nucleo\win32\1.1.0\sources\GSH\FreeRtos,` rename the file 'InvnFreeRTOSConfig.h' into 'FreeRTOSConfig.h'

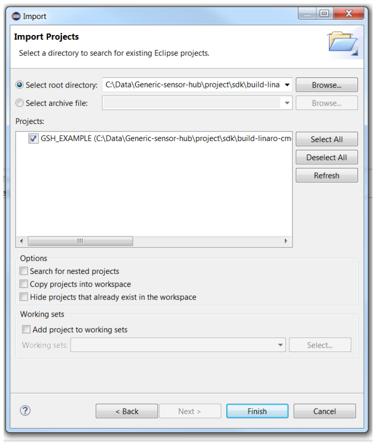
5 BUILDING A PROJECT USING ECLIPSE IN STANDALONE

5.1 INSTALLING ECLIPSE WORKBENCH IDE

- Download Eclipse C/C++ from https://eclipse.org/downloads/packages/eclipse-ide-cc-developers/ .
- Once the software has been installed, you should be able to open it and a window will ask you to open/create a workspace.

5.2 SETUP PROJECT

You are now able to import into the Eclipse workspace the project available in the SDK package. Select 'File import/General/Existing Projects' into Workspace and click next.


Import

Import Projects

Select a directory to search for existing Eclipse projects.

[✓ Select root directory:] C:\Data\Generic-sensor-hub\project\sdk\build-lina

Select archive file:

Projects:

[✓] GSH_EXAMPLE (C:\Data\Generic-sensor-hub\project\sdk\build-linaro-cm

Options

[ ] Search for nested projects

[ ] Copy projects into workspace

[ ] Hide projects that already exist in the workspace

Working sets

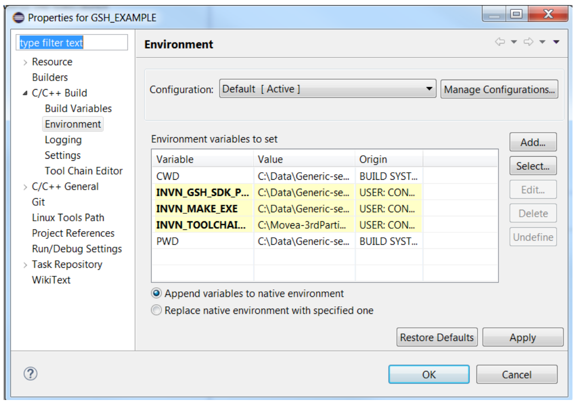
[ ] Add project to working sets

Working sets:

Figure 4 – Eclipse import Project

Browse to the location where you extracted the GSH package, then to `config\linaro-cm4-nucleo\eclipse` (and select *eclipse* folder*)*. Eclipse will automatically find the existing project. Click on Finish to import the project.

In the upper left hand side of Eclipse main window, the project explorer will display all the opened projects. Right click on the project you just added and select *properties*. In the 'Properties' window, go to 'C/C++ Build/Environment' panel.


Properties for GSH_EXAMPLE

Environment

Configuration: Default [Active]

Environment variables to set

Variable	Value	Origin
CWD	C:\Data\Generic-se...	BUILD SYST...
INVN_GSH_SDK_P...	C:\Data\Generic-se...	USER: CON...
INVN_MAKE_EXE	C:\Data\Generic-se...	USER: CON...
INVN_TOOLCHAI...	C:\Movea-3rdParti...	USER: CON...
PWD	C:\Data\Generic-se...	BUILD SYST...

[✓ Append variables to native environment]

Replace native environment with specified one

Figure 5 –Project properties

Now you will have to set some environment variables in order to be able to compile the project:

- **INVN_TOOLCHAIN_PATH** : should point to toolchain location (the folder where *arm-none-eabi-gcc.exe* is located)
  - ⇨ Example: `C:\path\to\toolchain\win32\linaro-5.2-2015.q4\bin\`
- **INVN_MAKE_EXE** : should point to make executable to be used (including executable name). For convenience, mingw32 make is available in the *tools/* folder.
  - ⇨ Example: `C:\invn\firmware\gsh\sdk\nucleo\win32\1.1.0\`*`tools\mingw32-make.exe`*
- **INVN_GSH_SDK_PATH** : should point to the SDK location
  - ⇨ Example: `C:\invn\firmware\gsh\sdk\nucleo\win32\1.1.0\`

After all the above steps are completed, you will be able to build the project. You will find the generated binaries in *config/linaro-cm4-nucleo/eclipse/GSH_EXAMPLE.bin.*

For reference, the Linaro toolchain version used for testing is **5.2-2015 q4** (available here: https://launchpad.net/gcc-arm-embedded/5.0/5-2015-q4-major/+download/gcc-arm-none-eabi-5_2-2015q4-20151219-win32.exe)

6 DEBUGGING A PROJECT USING ECLIPSE IN STANDALONE

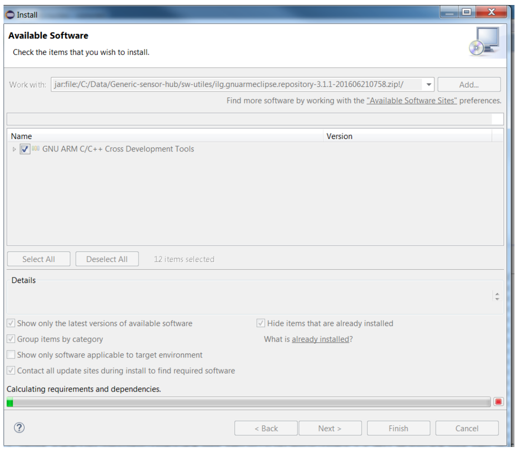
In order to debug the GSH project with Eclipse, you will have to download Openocd https://github.com/gnuarmeclipse/openocd/releases and install it.

Next, assuming your project set up is done (please refers to **part 4.1** and **4.2)**, set another environment variable in the project's properties:

- **INVN_OPENOCD_PATH** : should point to the directory where *openocd.exe* is located
  - ⇨ Example: `C:\Program Files\GNU ARM Eclipse\OpenOCD\0.10.0-201601101000-dev\bin`

Now download Eclipse plugin **GNU ARM Eclipse** available at https://sourceforge.net/projects/gnuarmeclipse/ and then follow these steps:

- In Eclipse, go to 'Help' > 'Install New Software'…' menu
- 'n 'Work w'th' section, click t'e ''dd' button on the right
- Cli'k 'Archive'..' and select the file previously downloaded. Cli'k 'Ok'
- Sele't 'GNU ARM C/C++ Cross Development To'ls' plugin and proceed to its installation


Install

Available Software

Check the items that you wish to install.

Work with: jar:file:/C:/Data/Generic-sensor-hub/sw-utiles/ilg.gnuarmeclipse.repository-3.1.1-201606210758.zip!/ Add...

Find more software by working with the "Available Software Sites" preferences.

Name	Version
☑ GNU ARM C/C++ Cross Development Tools	

Select All | Deselect All | 12 items selected

Details

☑ Show only the latest versions of available software ☑ Hide items that are already installed

☑ Group items by category What is already installed?

☐ Show only software applicable to target environment

☑ Contact all update sites during install to find required software

Calculating requirements and dependencies.

< Back | Next > | Finish | Cancel

Figure 6 –Install window

Once the aforementioned plugin is installed and OpenOCD is available on your computer and project environment variables are correctly set, two debugging configurations should be available:

- _Debug-GSH_EXAMPLE-(load)_: will reset the board and re-flash the FW
- _Debug-GSH_EXAMPLE-(no-load)_: will run the debugger without resetting the board and flashing the FW

To download/debug you have to click on the top of the Debug-GSH_EXAMPLE-(load) button. The debug mode will be launched and you will be able to step into the code.

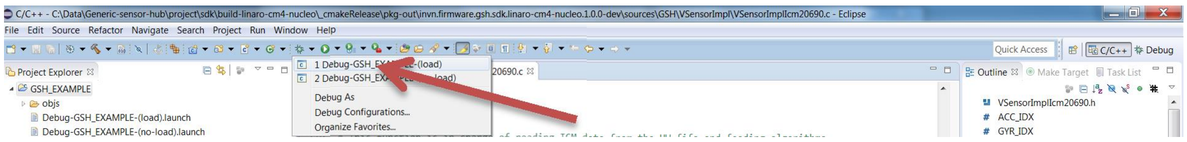

C/C++ - C:\Data\Generic-sensor-hub\project\sdk\build-linaro-cm4-nucleo_cmakeRelease\pkg-out\invn.firmware.gsh.sdk.linaro-cm4-nucleo.1.0.0-dev\sources\GSH\VSensorImpl\VSensorImplIcm20690.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
C/C++
Debug
Project Explorer
GSH_EXAMPLE
objs
Debug-GSH_EXAMPLE-(load).launch
Debug-GSH_EXAMPLE-(no-load).launch
1 Debug-GSH_EXAMPLE-(load)
2 Debug-GSH_EXAMPLE-(no-load)
Debug As
Debug Configurations...
Organize Favorites...
Outline
Make Target
Task List
VSensorImplIcm20690.h
ACC_IDX
GYR_IDX

Figure 7 –Launch debug session

If those launchers files are not available, please manually load them:

⇨ *File > Import... > Run/Debug > Launch Configurations*

7 BUILD AND DEBUG A PROJECT USING ECLIPSE AND SENSORSTUDIO

7.1 BUILD AND FLASH THE PROJECT

To build the Generic Sensor Hub sample project you generated with SensorStudio, click on Build button on GSH toolbar.

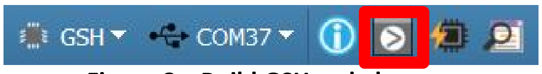
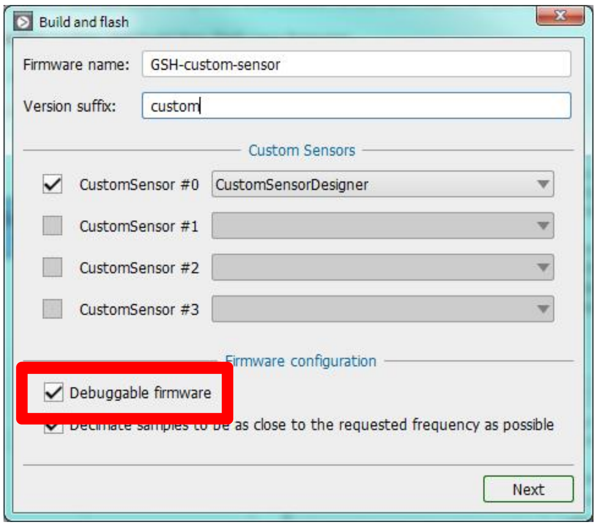


Figure 8 – Build GSH code button

In the build window, make sure to check the “Debuggable firmware” checkbox.


Build and flash

Firmware name: GSH-custom-sensor

Version suffix: custom

Custom Sensors

- [✓] CustomSensor #0: CustomSensorDesigner
- [ ] CustomSensor #1
- [ ] CustomSensor #2
- [ ] CustomSensor #3

Firmware configuration

- [✓] Debuggable firmware
- [✓] Decimate samples to be as close to the requested frequency as possible

Figure–9 - Debuggable firmware checkbox

Once the build is successful (this can take a few minutes for the first build), the next step in the wizard will flash the image. You can also flash the image, or the factory image, by clicking on the flash button from the toolbar.

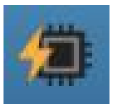
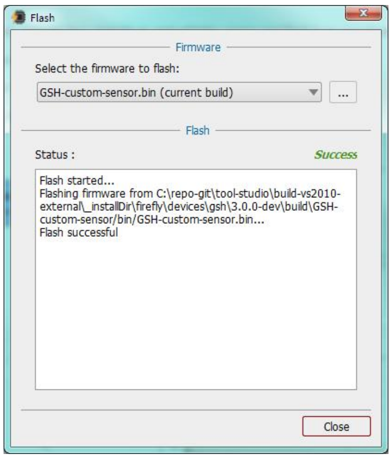


Figure 10 – Flash GSH code button

A window will appear and you will have to click on flash button to launch the firmware download.


Flash

Firmware

Select the firmware to flash: GSH-custom-sensor.bin (current build)

Flash

Status : *Success*

```
Flash started...
Flashing firmware from C:\repo-git\tool-studio\build-vs2010-
external\_installDir\firefly\devices\gsh\3.0.0-dev\build\GSH-
custom-sensor/bin/GSH-custom-sensor.bin...
Flash successful
```

Figure 11 – Flash GSH window

7.2 RUNNING THE PROJECT FROM DEBUGGER

Once you start your flow on SensorStudio, you can start the debugging session on Eclipse IDE. Your Eclipse project is generated automatically. Before running the debug session, make sure you connect the system properly. Then click on Debug button in SensorStudio.

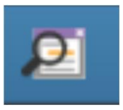
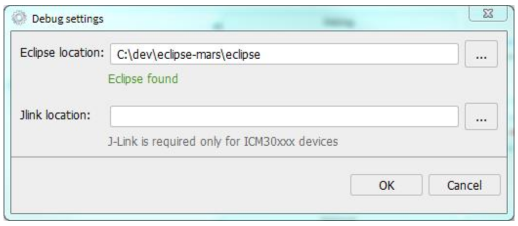


Figure –2 - Debug firmware button

The first time you launch the debug session, SensorStudio will ask for the location of Eclipse. Notice that you don't need to fill J-Link location for GSH.



Debug settings

Eclipse location: C:\dev\eclipse-mars\eclipse

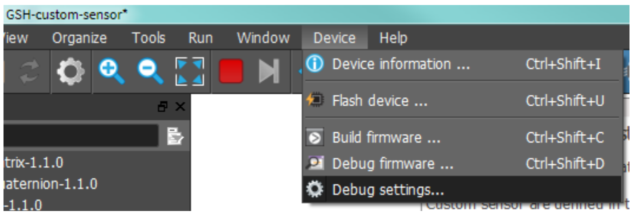
Eclipse found

Jlink location:

J-Link is required only for ICM30xxx devices

Figure –3 - Debug settings

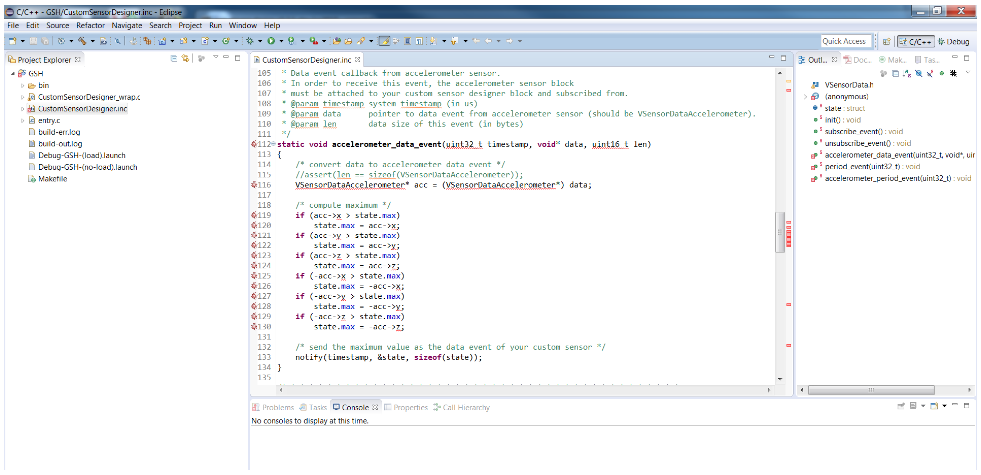
These settings can be modified at any time via Device -> Settings… menu:



GSH-custom-sensor*
iew
Organize
Tools
Run
Window
Device
Help
Device information ... Ctrl+Shift+I
Flash device ... Ctrl+Shift+U
Build firmware ... Ctrl+Shift+C
Debug firmware ... Ctrl+Shift+D
Debug settings...
trix-1.1.0
aternion-1.1.0
-1.1.0

Figure –4 - Accessing settings

Once the settings set, the Eclipse IDE is automatically launched (this can take up to a minute on the first launch). At the first time you start Eclipse IDE, there is a Welcome page that you can close. Now you have the C/C++ Project view, with your project files available.



```
 * Data event callback from accelerometer sensor.
 * In order to receive this event, the accelerometer sensor block
 * must be attached to your custom sensor designer block and subscribed from.
 * @param timestamp system timestamp (in us)
 * @param data      pointer to data event from accelerometer sensor (should be VSensorDataAccelerometer).
 * @param len       data size of this event (in bytes)
 */
static void accelerometer_data_event(uint32_t timestamp, void* data, uint16_t len)
{
    /* convert data to accelerometer data event */
    //assert(len == sizeof(VSensorDataAccelerometer));
    VSensorDataAccelerometer* acc = (VSensorDataAccelerometer*) data;

    /* compute maximum */
    if (acc->x > state.max)
        state.max = acc->x;
    if (acc->y > state.max)
        state.max = acc->y;
    if (acc->z > state.max)
        state.max = acc->z;
    if (-acc->x > state.max)
        state.max = -acc->x;
    if (-acc->y > state.max)
        state.max = -acc->y;
    if (-acc->z > state.max)
        state.max = -acc->z;

    /* send the maximum value as the data event of your custom sensor */
    notify(timestamp, &state, sizeof(state));
}
```

No consoles to display at this time.

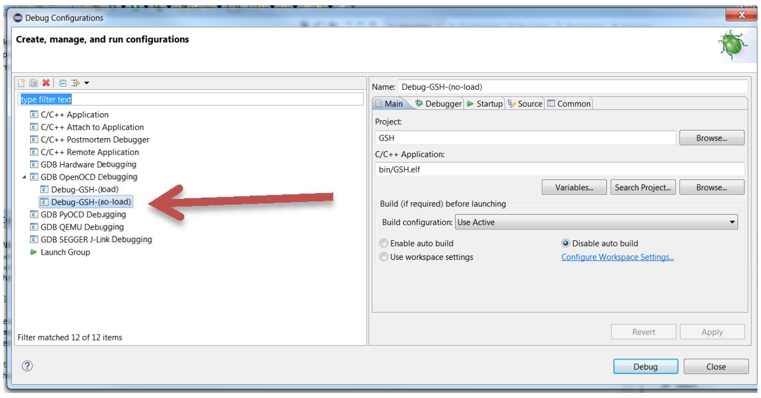
Figure 15 – Eclipse IDE project view

The debug session is already configured for the project in the GSH debug configuration. The debugger connects to the target on the running using OpenOCD.

Click Run->Debug Configuration…

On the new windows you have select to use GDB OpenOcd for debug, either in no-load mode or in load mode.

Warning	In load mode, the firmware will be re-flashed into the Nucleo. The board will be reset and the program will start at the beginning of main() function.


Debug Configurations

Create, manage, and run configurations

type filter text
- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- GDB Hardware Debugging
- GDB OpenOCD Debugging
  - Debug-GSH-(load)
  - Debug-GSH-(no-load)
- GDB PyOCD Debugging
- GDB QEMU Debugging
- GDB SEGGER J-Link Debugging
- Launch Group

Filter matched 12 of 12 items

Name: Debug-GSH-(no-load)

Main | Debugger | Startup | Source | Common

Project: GSH

C/C++ Application: bin/GSH.elf

Build (if required) before launching

Build configuration: Use Active

Enable auto build [✓ Disable auto build]

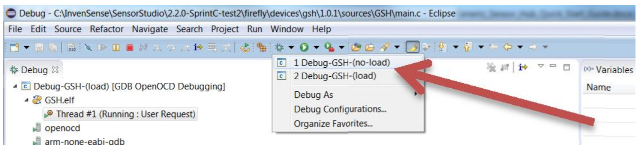
Use workspace settings

Configure Workspace Settings...

Figure 16 – Debug configuration

Then click on Debug button to launch the debug session. Please note it is not possible to connect to the target if the "Debuggable firmware" option is not checked when building your firmware.

You can also launch easily the debug session by clicking on the little green bug on the top toolbar.


Debug - C:\InvenSense\SensorStudio\2.2.0-SprintC-test2\firefly\devices\gsh\1.0.1\sources\GSH\main.c - Eclipse

Debug
- Debug-GSH-(load) [GDB OpenOCD Debugging]
  - GSH.elf
    - Thread #1 (Running : User Request)
  - openocd
  - arm-none-eabi-gdb

Figure 17 – Launch debug session

Now, you can see the Eclipse Debug perspective. The execution of the application can now be controlled from the debugger in Eclipse IDE. Press Suspend/Resume button to stop/start running the application, press Terminate button to close the debug session. You can also put breakpoints in your code by double-clicking on the desired line number.

8 BUILD AND DOWNLOAD A DEFAULT PROJECT

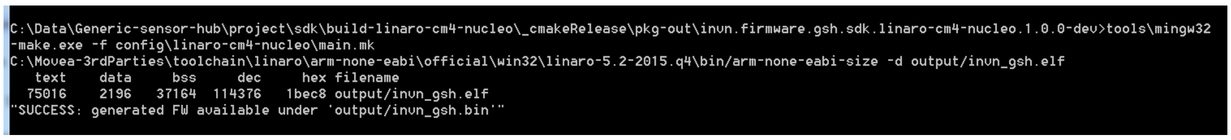
8.1 BUILD USING MAKEFILE DIRECTLY

To generate the GSH firmware using GCC you will have to use **mingw32-make** available in the SDK in the tools repository if gcc is not installed on your computer.

Please set **INVN_TOOLCHAIN_PATH** in a console to point to the toolchain location and then run mingw32-cmake:

```
set INVN_TOOLCHAIN_PATH=path\to\the\toolchain\bin
```

```
mingw32-make.exe -f config\linaro-cm4-nucleo\main.mk
```



```
C:\Data\Generic-sensor-hub\project\sdk\build-linaro-cm4-nucleo\_cmakeRelease\pkg-out\invn.firmware.gsh.sdk.linaro-cm4-nucleo.1.0.0-dev>tools\mingw32
-make.exe -f config\linaro-cm4-nucleo\main.mk
C:\Movea-3rdParties\toolchain\linaro\arm-none-eabi\official\win32\linaro-5.2-2015.q4\bin/arm-none-eabi-size -d output/invn_gsh.elf
   text    data     bss     dec     hex filename
  75016    2196   37164  114376   1bec8 output/invn_gsh.elf
"SUCCESS: generated FW available under 'output/invn_gsh.bin'"
```

Figure 18 –Successful build

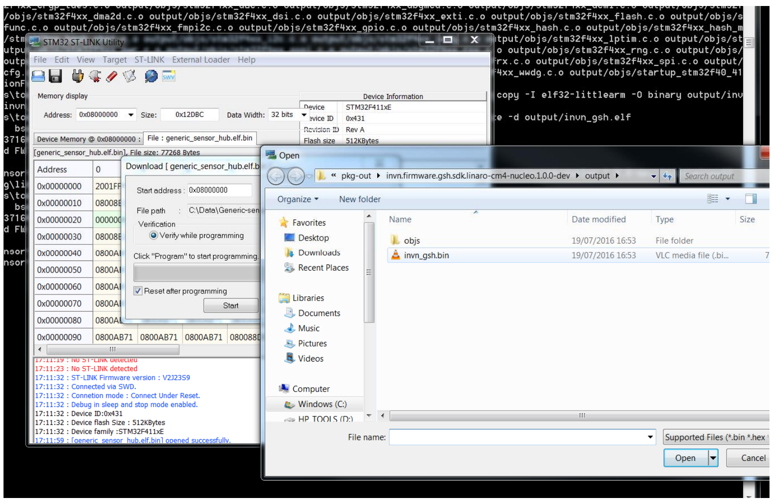
The firmware is generated into *output* folder from the current path.

8.2 DOWNLOAD WITH ST-LINK UTILITIES

Firstly, you have to download ST link utilities from http://www.st.com/web/en/catalog/tools/PF258168 . This software allows you to flash STM32 Nucleo board directly.

Launch ST-Link Utilities and then :

- 'Connect' to the target
- Click on 'Program verify', a windows will be open
- Browse your file path to the previously built GSH firmware binaries
- Click on 'start' to download it



Address: 0x08000000 Size: 0x12DBC Data Width: 32 bits

Device STM32F411xE
Device ID 0x431
Revision ID Rev A
Flash size 512KBytes

Device Memory @ 0x08000000 : File : generic_sensor_hub.elf.bin
[generic_sensor_hub.elf.bin], File size: 77268 Bytes

Download [generic_sensor_hub.elf.bin]
Start address : 0x08000000
File path : C:\Data\Generic-sen
Verification: Verify while programming
Click "Program" to start programming.
Reset after programming [✓]

« pkg-out ▸ invn.firmware.gsh.sdk.linaro-cm4-nucleo.1.0.0-dev ▸ output ▸

Name	Date modified	Type	Size
objs	19/07/2016 16:53	File folder	
invn_gsh.bin	19/07/2016 16:53	VLC media file (.bi...	7

File name: Supported Files (*.bin *.hex

```
17:11:19 : No ST-LINK detected
17:11:23 : No ST-LINK detected
17:11:32 : ST-LINK Firmware version : V2J23S9
17:11:32 : Connected via SWD.
17:11:32 : Connetion mode : Connect Under Reset.
17:11:32 : Debug in sleep and stop mode enabled.
17:11:32 : Device ID:0x431
17:11:32 : Device flash Size : 512KBytes
17:11:32 : Device family :STM32F411xE
17:11:59 : [generic_sensor_hub.elf.bin] opened successfully.
```

Figure 19 –Download binary with ST link Utilities

9 SENSOR-CLI

9.1 OVERVIEW

sensor-cli is a command line application used to control an InvenSense device from a PC for evaluation and testing purpose. It is available in the tools repository of the GSH package.

Included features, non-exhaustive list:

- Start/stop/configure sensors for an InvenSense devices
- Display data on the screen
- Log sensor data to file

9.2 USAGE

Being a command line application, *sensor-cli* should be launched from a console with a few mandatory arguments.

`sensor-c- --help` will provide brief information about sensor-cli invocation.

The main option to pass to sensor-cli are the target device (--*target*) and adapter to use (--*adapter*).

Another useful option – --*level* to enable diagnostic messages (eg: --level=debug)

Example: `sensor-c- --target=gsh,port=\\.\COM- --adapter=dum- --level=debug`

Where *port* is the UART COM port number used for the communication between the computer and the system (connected to CON1 micro-USB on the Carrier Board).
Use the *Device Manager* to see which port is used on your computer. To quickly start Device Manager, you can press 'Windows+R' shortcut to run a command and execute 'devmgmt.msc'. Look under 'Ports (COM & LPT)', the device is called "USB Serial Port (COMXX)".

If the device is correctly connected and successfully setup, *sensor-cli* will display a prompt and wait for commands to be input by the user.

The '*help*' command will provide a list of all supported commands with a short description. Adding the command name to the 'help' command will give detailed information about a specific command.

```
sensor-cli> help en

Start a sensor designated by its short name or its id.

‘f 'per’od' a‘d 'time’ut' are given, this command will also configure the sensor period and sensor batch
timeout.

Synopsys: enable sensor_id [period_ms] [timeout_ms]

          enable sensor_name [period_ms] [timeout_ms]
```

The '*quit*' or '*exit*' command should be used to properly exit *sensor-cli*

9.2.1 Controlling sensors

Sensor can be enabled or disabled using the '*en*' (for enable) and '*dis*' (for disable) commands. Output data rate can be changed using the '*odr*' command.

Those commands expect as a first argument, the sensor id or name to be controlled. Those can be obtained with the '*ids*' command:

```
sensor-cli> ids

  1 [0x01] a-c - SENSOR_ACCELEROMETER

 13 [0x0d] ate-p - SENSOR_AMBIENT_TEMPERATURE

 29 [0x1d] ax-s - SENSOR_3AXIS

 28 [0x1c] b-s - SENSOR_B2S

 26 [0x1a] b-c - SENSOR_BAC

 20 [0x14] geo-v - SENSOR_GEOMAG_ROTATION_VECTOR

    1    24 [0x18] glan-e - SENSOR_GLANCE_GESTU 9 [0x09] g-a - SENSOR_GRAVITY
 15 [0x0f] g-v - SENSOR_GAME_ROTATION_VECTOR
  4 [0x04] g-r - SENSOR_GYROSCOPE
 21 [0x15] h-m - SENSOR_HEART_RATE
 12 [0x0c] humidi-y - SENSOR_HUMIDITY
    1     5 [0x05] lig-t - SENSOR_LIG10 [0x0a] lina-c - SENSOR_LINEAR_ACCELERATI 2 [0x02] m-g -
       SENSOR_MAGNETOMETER
    1     0 [0x00] reserv-d - SENSOR_RESERV 3 [0x03] o-i - SENSOR_ORIENTATION
 27 [0x1b] p-r - SENSOR_PDR
...
```

In order to start the calibrated accelerometer, one would use the command '*en acc*' and in order to start the RAW gyroscope '*en rgyr*'.

In order to find out if your device supports a certain sensor, use the '*ping*' command.

9.2.2 Display quaternion as a cube

The 'cube' command allows the user to display a rotating cube based on the quaternion data.

To display orientation data corresponding to GAME ROTATION VECTOR sensor, the command would be '*cube on grv*' and '*cube off grv*' to close the cube windows.

Important note: the cube display is independent of the sensor control. Notice that grv sensor must be started separately with command *'en grv'*.

9.2.3 Redirecting sensor events

By default, sensor events are displayed in the main windows (the one that invocates sensor-cli), which can make it difficult to enter commands while events are reported.

It is possible to redirect sensor-events to:

- a file with '*disp > file_name*'
- the void with '*disp off*'
- to another window with '*disp >| named_pipe*'

The later command relies on operating system named-pipe. The *pipe-cat* is provided with *sensor-cli* to easily create named pipe under Windows. See *pipe-cat –help*

9.2.4 Examples

We will describe three examples in this section :

- how to display a cube using Game Rotation Vector ?

- how to log data from accelerometer?

- how to run custom sensor?

Both examples will use GSH with Icm20690 device connected through SPI. The hardware needs to be fully set up and the Nucleo flashed with the GSH firmware image previously built.

Run sensor-cli , available in tools repository from the GSH package:

```
sensor-cli.e- --target=gsh,port=\\.\COM- --adapter=dummy
```

9.2.4.1 How to display a cube using Rotation Vector

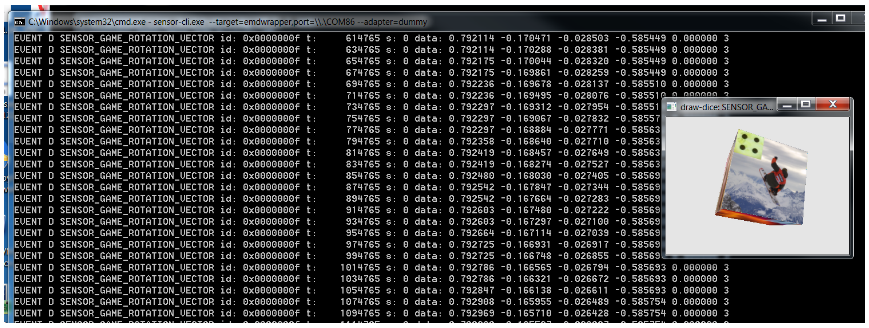
In order to have a good quaternion behavior, it is recommended to calibrate accelerometer and gyroscope before. Now enable the cube display for Game Rotation Vector Sensor (GRV) :

```
sensor-cli> cube on grv
```

A new window displaying a cube will be opened.

Now enable the GRV with a data output period of 20 ms by typing:

```
sensor-cli> en grv 20
```


C:\Windows\system32\cmd.exe - sensor-cli.exe --target=emdwrapper,port=\\.\COM86 --adapter=dummy
draw-dice: SENSOR_GA...

```
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     614765 s: 0 data: 0.792114 -0.170471 -0.028503 -0.585449 0.000000 3
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     634765 s: 0 data: 0.792114 -0.170288 -0.028381 -0.585449 0.000000 3
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     654765 s: 0 data: 0.792175 -0.170044 -0.028320 -0.585449 0.000000 3
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     674765 s: 0 data: 0.792175 -0.169861 -0.028259 -0.585449 0.000000 3
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     694765 s: 0 data: 0.792236 -0.169678 -0.028137 -0.585510 0.000000 3
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     714765 s: 0 data: 0.792236 -0.169495 -0.028076 -0.585510
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     734765 s: 0 data: 0.792297 -0.169312 -0.027954 -0.58551
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     754765 s: 0 data: 0.792297 -0.169067 -0.027832 -0.58557
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     774765 s: 0 data: 0.792297 -0.168884 -0.027771 -0.58563
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     794765 s: 0 data: 0.792358 -0.168640 -0.027710 -0.58563
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     814765 s: 0 data: 0.792419 -0.168457 -0.027649 -0.58563
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     834765 s: 0 data: 0.792419 -0.168274 -0.027527 -0.58563
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     854765 s: 0 data: 0.792480 -0.168030 -0.027405 -0.58569
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     874765 s: 0 data: 0.792542 -0.167847 -0.027344 -0.58569
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     894765 s: 0 data: 0.792542 -0.167664 -0.027283 -0.58569
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     914765 s: 0 data: 0.792603 -0.167480 -0.027222 -0.58569
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     934765 s: 0 data: 0.792603 -0.167297 -0.027100 -0.58569
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     954765 s: 0 data: 0.792664 -0.167114 -0.027039 -0.58569
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     974765 s: 0 data: 0.792725 -0.166931 -0.026917 -0.58569
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:     994765 s: 0 data: 0.792725 -0.166748 -0.026855 -0.58569
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:    1014765 s: 0 data: 0.792786 -0.166565 -0.026794 -0.585693 0.000000 3
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:    1034765 s: 0 data: 0.792786 -0.166321 -0.026672 -0.585693 0.000000 3
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:    1054765 s: 0 data: 0.792847 -0.166138 -0.026611 -0.585693 0.000000 3
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:    1074765 s: 0 data: 0.792908 -0.165955 -0.026489 -0.585754 0.000000 3
EVENT D SENSOR_GAME_ROTATION_VECTOR id: 0x0000000f t:    1094765 s: 0 data: 0.792969 -0.165710 -0.026428 -0.585754 0.000000 3
```

Data will be displayed and the cube will move according to the board motion.

Then you can stop the sensor:

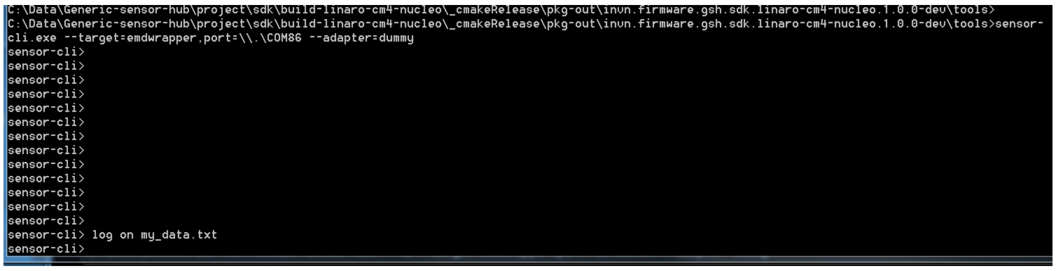
```
sensor-cli> dis grv
```

Data output will be stopped.

9.2.4.2 How to log data

Firstly, enable log and set a destination file. If you do not set a path, the file will be created in the current directory.

```
sensor-cli> log on my_data.txt
```



```
C:\Data\Generic-sensor-hub\project\sdk\build-linaro-cm4-nucleo\_cmakeRelease\pkg-out\invn.firmware.gsh.sdk.linaro-cm4-nucleo.1.0.0-dev\tools>
C:\Data\Generic-sensor-hub\project\sdk\build-linaro-cm4-nucleo\_cmakeRelease\pkg-out\invn.firmware.gsh.sdk.linaro-cm4-nucleo.1.0.0-dev\tools>sensor-
cli.exe --target=emdwrapper,port=\\.\COM86 --adapter=dummy
sensor-cli>
sensor-cli>
sensor-cli>
sensor-cli>
sensor-cli>
sensor-cli>
sensor-cli>
sensor-cli>
sensor-cli>
sensor-cli>
sensor-cli>
sensor-cli>
sensor-cli>
sensor-cli> log on my_data.txt
sensor-cli>
```

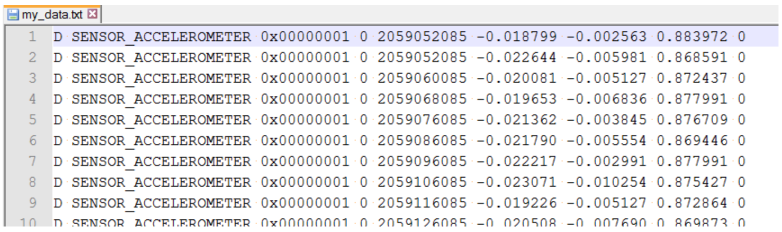
Then enable sensors, eg: accelerometer with 10 ms data rate.

```
sensor-cli> en acc 10
```

After you performing your tests you can disable the sensor.

```
sensor-cli> dis acc
```

Finally, you can check data from your log file:



```
D SENSOR_ACCELEROMETER 0x00000001 0 2059052085 -0.018799 -0.002563 0.883972 0
D SENSOR_ACCELEROMETER 0x00000001 0 2059052085 -0.022644 -0.005981 0.868591 0
D SENSOR_ACCELEROMETER 0x00000001 0 2059060085 -0.020081 -0.005127 0.872437 0
D SENSOR_ACCELEROMETER 0x00000001 0 2059068085 -0.019653 -0.006836 0.877991 0
D SENSOR_ACCELEROMETER 0x00000001 0 2059076085 -0.021362 -0.003845 0.876709 0
D SENSOR_ACCELEROMETER 0x00000001 0 2059086085 -0.021790 -0.005554 0.869446 0
D SENSOR_ACCELEROMETER 0x00000001 0 2059096085 -0.022217 -0.002991 0.877991 0
D SENSOR_ACCELEROMETER 0x00000001 0 2059106085 -0.023071 -0.010254 0.875427 0
D SENSOR_ACCELEROMETER 0x00000001 0 2059116085 -0.019226 -0.005127 0.872864 0
D SENSOR_ACCELEROMETER 0x00000001 0 2059126085 -0.020508 -0.007690 0.869873 0
```

9.2.4.3 How to run custom sensor

By default, custom sensors examples are build-out. You will first need to enable them.

Add the files located under ‘sources/GSH/ExtFwk/examples’ to the ***main.mk*** :

```
# List of files and dependencies (automatically-generated)
IDIRS   +=  \
   sources \
   sources/board-hal \
   […]
   sources/FreeRTOS/Source/portable/MemMang\
+  sources/GSH/ExtFwk/examples
LDIRS   +=  \
   bin/linaro-cm4-nucleo
DEPS    +=  \
   sources/board-hal/dbg_gpio.h \
   sources/board-hal/delay.h \
   […]
   sources/stm32f4x/STM32F4xx_StdPeriph_Driver/inc/stm32f4xx_wwdg.h \
+  sources/GSH/ExtFwk/examples/Ak09911Driver.h
CSRCS   +=  \
   sources/board-hal/dbg_gpio.c \
   sources/board-hal/delay.c \
   […]
   sources/stm32f4x/CMSIS/Device/gcc_ride7/startup_stm32f40_41xxx.s \
+  sources/GSH/ExtFwk/examples/Ak09911Driver.c \
+  sources/GSH/ExtFwk/examples/CustomEntry.c \
+  sources/GSH/ExtFwk/examples/CustomSensor0.c \
+  sources/GSH/ExtFwk/examples/CustomSensor1.c \
+  sources/GSH/ExtFwk/examples/CustomSensor2.c \
+  sources/GSH/ExtFwk/examples/CustomSensor3.c
```

You will then need to rebuild the project.

You can then run *sensor-cli* and type *ping* command to see your custom sensors available.

```
sensor-cli> ping
Ping -K - SENSOR_ACCELEROMETER (id: 1)
Ping -K - SENSOR_GYROSCOPE (id: 4)
Ping -K - SENSOR_GRAVITY (id: 9)
Ping -K - SENSOR_LINEAR_ACCELERATION (id: 10)
Ping -K - SENSOR_GAME_ROTATION_VECTOR (id: 15)
Ping -K - SENSOR_UNCAL_GYROSCOPE (id: 16)
Ping -K - SENSOR_RAW_ACCELEROMETER (id: 32)
Ping -K - SENSOR_RAW_GYROSCOPE (id: 33)
Ping -K - SENSOR_CUSTOM0 (id: 49)
Ping -K - SENSOR_CUSTOM1 (id: 50)
Ping -K - SENSOR_CUSTOM3 (id: 52)
```

The Custom sensor 0, 1 and 3 are now available. Custom Sensor 2 answers KO to ping because it tries to connect to a magnetometer, physically not available.

10 DOCUMENT INFORMATION

10.1 REVISION HISTORY

REVISION	DATE	DESCRIPTION	AUTHOR
1.0	July 19, 2016	Initial version.	Axel Zbitak
1.1	July 29, 2016	Rewording and update following latest changes	Loic Michallon
1.2	August 17, 2016	Clarifications following feedback from Paul and Rajesh	Loic Michallon
1.3	September 15, 2016	Update the document according to 1.1.0 scope	Axel Zbitak
1.4	September 16, 2016	Add feedback from Zoran	Axel Zbitak
1.5	October 3, 2016	Add supported frequencies Update to Carrier board revC	Loic Michallon
1.6	October 12, 2016	Add FreeRTOS sources installation	Axel Zbitak
1.7	October 13, 2016	Updated SensorStudio documentation	Thomas Muguet
1.8	October 17, 2016	Add warning concerning load/no-load with Eclipse	Axel Zbitak

Table 6. Revision History