

# Using Eclipse IDE with J-Link Debugger

## ICM-30630 Embedded Application Note

## Table OF CONTENTSs

Abstract .....	4
1. Introduction.....	4
2. Prerequisites.....	4
2.1. Platform.....	4
2.2. Hardware .....	4
2.3. Software .....	5
3. Software installation instructions .....	6
3.1. J-Link Debugger .....	6
3.2. Eclipse IDE.....	7
3.1. SensorStudio.....	9
4. Build and debug project .....	9
4.1. Build the project .....	9
4.2. Running the project from debugger .....	9
5. Revision History .....	13

## TABLE OF FIGURES

Figure 1 – ICM-30630 Embedded Jumper's Configuration for Debug .....	5
Figure 2 - J-Link Debugger and its SWD Connector .....	5
Figure 3 - Run the J-Link Flash Programmer Installer.....	6
Figure 4 - Installing J-Link Flash Programmer .....	7
Figure 5 - Install New Software Plug-Ins.....	8
Figure 6 - Add the GNU ARM Eclipse Plug-Ins Archive .....	8
Figure 7 - Select all GNU ARM Eclipse plug-Ins .....	8
Figure 8 - Installing GNU ARM Eclipse Plug-Ins .....	9
Figure 9 – Build Firefly Code Button.....	9
Figure 10 - Debug Firmware Button .....	9
Figure 11 - Firefly Settings .....	10
Figure 12 - Accessing Firefly Settings .....	10
Figure 13 – Eclipse IDE Project View .....	11
Figure 14 - Select the Firefly Debug Configuration .....	11
Figure 15 - Running the Project from Debugger .....	12

## 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 eMD Software User Guide

ICM-30630 Command Protocol and Architecture

ICM-30630 Shield Hardware Guide

SensorStudio User Documentation

## **ABSTRACT**

This application note describes how to set up a software-based debugging environment for the ICM-30630 embedded shield with SensorStudio, using Eclipse Integrated Development Environment (IDE) and the SEGGER J-Link debugger.

## **1. INTRODUCTION**

The ICM-30630 embedded shield using with the SensorStudio is based on GCC development tools. The purpose of this solution is to allow sensor management and algorithm development. The embedded ICM-30630 running through SensorStudio was created as an advanced sensor hub, easy to modify and upgrade for developers. The ICM-30630 embedded and SensorStudio suite tool include a full sensor software solution for the user to develop new features and applications in order to program their own logic.

This document describes which software packages are necessary, and provides installation and configuration instructions for each.

Section 2 lists the software packages and hardware required. Installation procedures for the software packages are provided in Section 3. The next section describes how to use the Eclipse IDE through SensorStudio to build the project and run a debugging session.

## **2. PREREQUISITES**

### **2.1. PLATFORM**

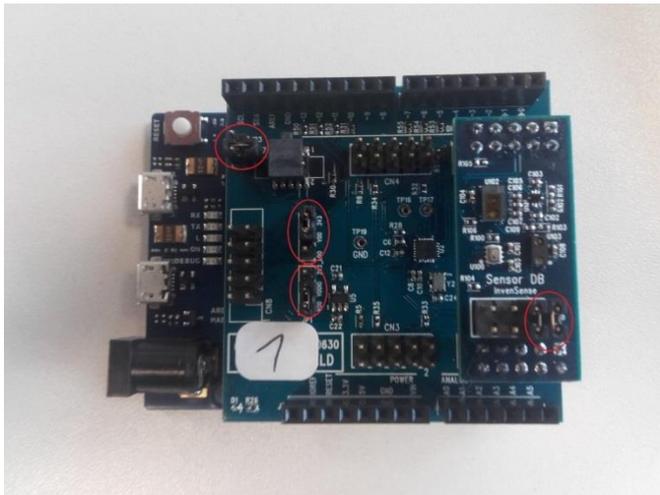
The tool setup has been tested on Windows 7. Linux platforms are not supported in this application note.

### **2.2. HARDWARE**

The following hardware pieces are required:

- USB cable
- Arduino Zero board, connected to PC through USB programming port
- ICM-30630-embedded shield

During debugging session; please be careful of the jumper's configuration.



On the ICM-30630 shield:

- JP1 connects VDD to VDDIO
- JP2 connects VDD to 3V3
- **JP3 is open**

On Sensor DB

- J100 connects  
1<->2 and 3<->4

Figure 1 – ICM-30630 Embedded Jumper's Configuration for Debug

- SEGGER J-Link Debugger
  - SWD cable to connect to ICM-30630 embedded shield through CN7 connector (for more information refer to ICM-30630 embedded shield datasheet).
- The connector is sold with a white full pin (pin 7) to be used as keyed, it is not needed for the ICM-30630 shield, the user can safely remove it.



Figure 2 - J-Link Debugger and its SWD Connector

The Sensor DB is optional and adds available external sensors: magnetometer, proximity sensor and barometer.

## 2.3. SOFTWARE

This section describes the software packages that are required for this application note. The below packages might be used, as well as the newer version which became available after this document was written.

- The Tool Suite: SensorStudio
- The IDE : Eclipse and the following components
  - o Java Runtime Environment – JRE
  - o Eclipse Mars (the oldest version supporting GNU ARM Eclipse plug-ins is Eclipse Kepler 4.4 SR2)GNU Toolchain for ARM Embedded Processors linaro-4.7-2013 included in the SensorStudio tool suite package

- GNU ARM Eclipse plug-ins
- The SEGGER J-Link software

## 3. SOFTWARE INSTALLATION INSTRUCTIONS

### 3.1. J-LINK DEBUGGER

Download the J-Link installer available here <https://www.segger.com/jlink-software.html> . Unzip the setup folder and run **Setup\_JLink\_xxxx.exe**, where “xxx” is the software version. The version package tested during writing this document is v5.02c.



Figure 3 - Run the J-Link Flash Programmer Installer

Install the USB Driver for J-Link. The default path for the J-Link software is C:\Program Files (x86)\SEGGER\JLink\_V502c. Click on Next to install the J-Link Debugger.

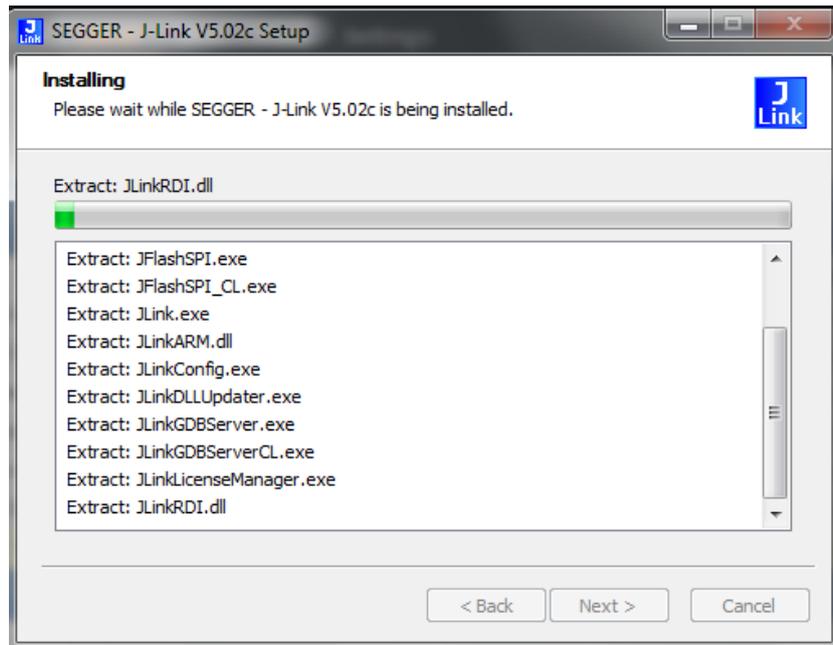


Figure 4 - Installing J-Link Flash Programmer

### 3.2. ECLIPSE IDE

Preamble: The Eclipse Integrated Development Environment (IDE) is dependent on the Java Runtime Environment (JRE) being installed on the machine. Both must be downloading on the same platform, for example, the Eclipse IDE x64 version needs the JRE x64 version. Make sure the correct JRE version is installed on your system before running Eclipse installer. You can also download it here:

<http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html> .

Download Eclipse IDE from the eclipse website <https://eclipse.org/downloads/> . Please select “**Eclipse IDE for C/C++ Developers**” that contains C/C+ Development Tooling (CDT) needed.

Unzip the download package, and run **eclipse.exe**. Eclipse will ask for a folder to use for workspace location when started.

Please follow these instructions to install Eclipse needed components for GNU ARM Eclipse plug-ins:

1. Download GNU ARM Eclipse plug-ins on website:[http://sourceforge.net/projects/gnuarmeclipse/?source=typ\\_redirect](http://sourceforge.net/projects/gnuarmeclipse/?source=typ_redirect)
2. Install GNU ARM Eclipse plug-ins; navigate to Help->Install New Software

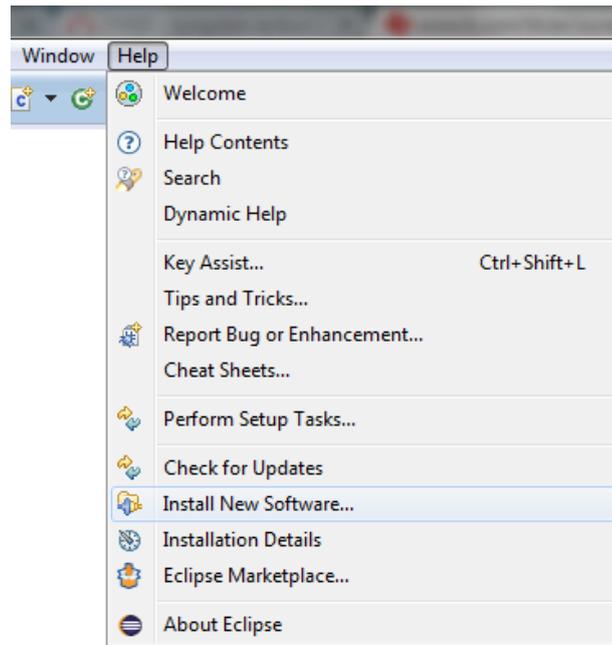


Figure 5 - Install New Software Plug-Ins

3. Click to Add. To select GNU ARM Eclipse plug-ins, select the Archive to directly load the zip folder.

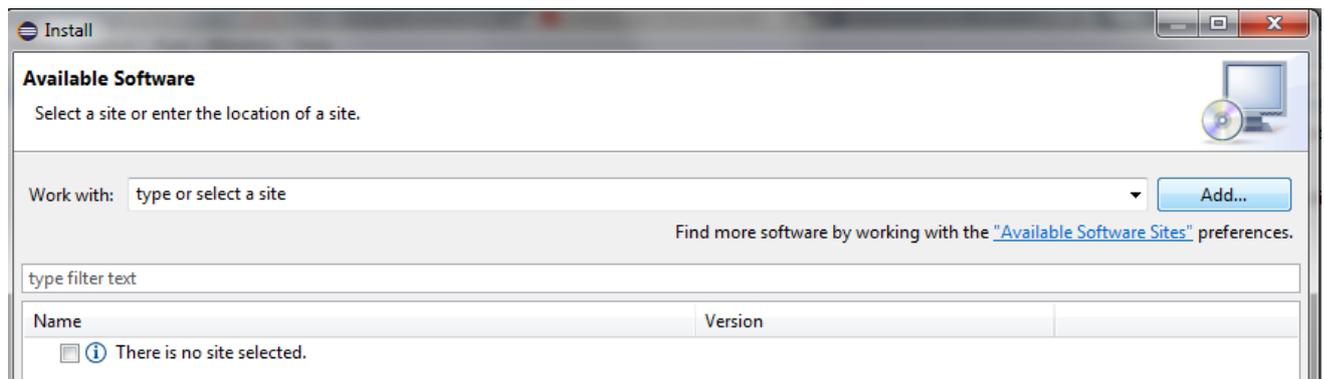


Figure 6 - Add the GNU ARM Eclipse Plug-Ins Archive

4. Select all items available

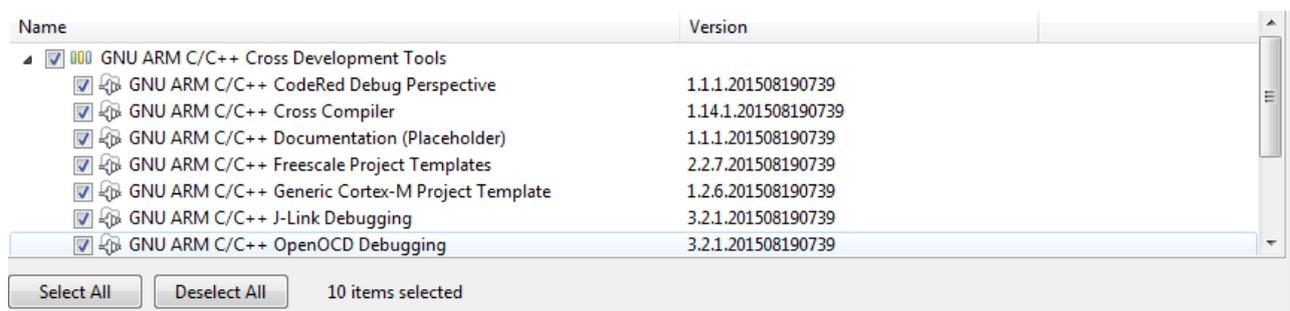


Figure 7 - Select all GNU ARM Eclipse plug-Ins

## 5. Finish the installation

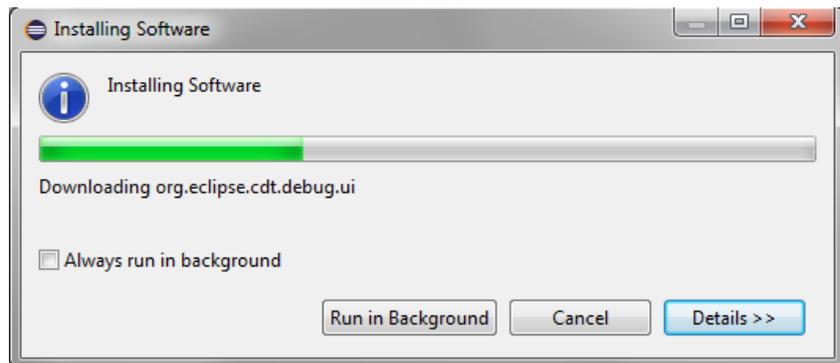


Figure 8 - Installing GNU ARM Eclipse Plug-Ins

You will need to restart Eclipse, take into account the plug-ins installation.

Once these package installations are completed, you can verify that the CDT and the GNU ARM Eclipse plug-ins are correctly installed. Navigate to Help-> Installation Details. Make sure that the C/C++ Development Platform, The GNU ARM C/C++ Cross Compiler and the GNU ARM C/C++ J-Link Debugging are installed.

### 3.1. SENSORSTUDIO

Please refer to SensorStudio documentation.

## 4. BUILD AND DEBUG PROJECT

### 4.1. BUILD THE PROJECT

To build the Firefly sample project you generated with SensorStudio, click on “Build” button on Firefly toolbar.



Figure 9 – Build Firefly Code Button

### 4.2. 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 automatically generated and available. Before running the debug session, make sure you connect the J-Link SWD connector to the ICM-30630 embedded shield. Then click on “Debug” button on SensorStudio.

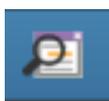


Figure 10 - Debug Firmware Button

The first time you launch “Debug”, SensorStudio asks for the location of Eclipse and J-Link debugger:

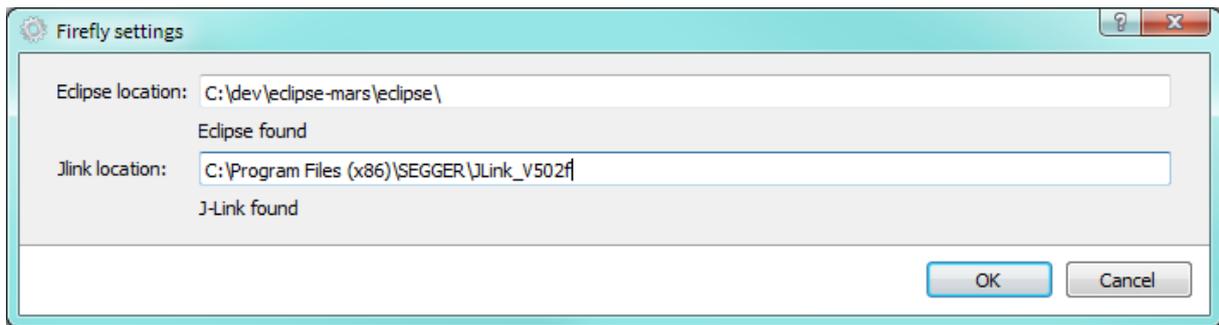


Figure 11 - Firefly Settings

These settings can be modified at any time via FireFly -> Settings... menu:

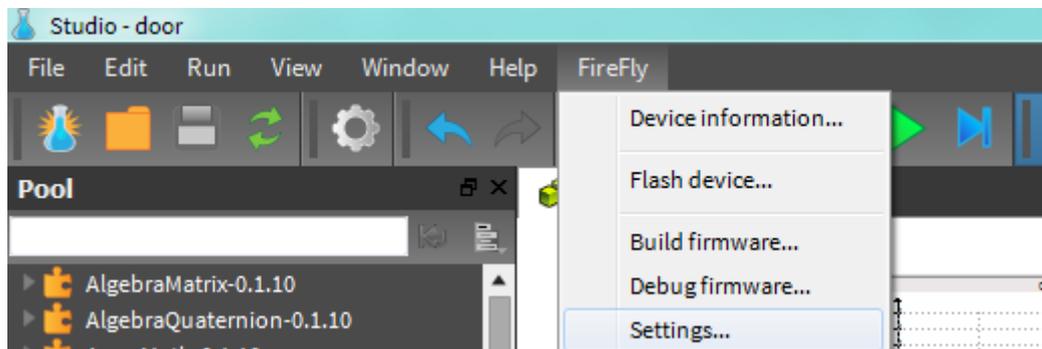


Figure 12 - Accessing Firefly Settings

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

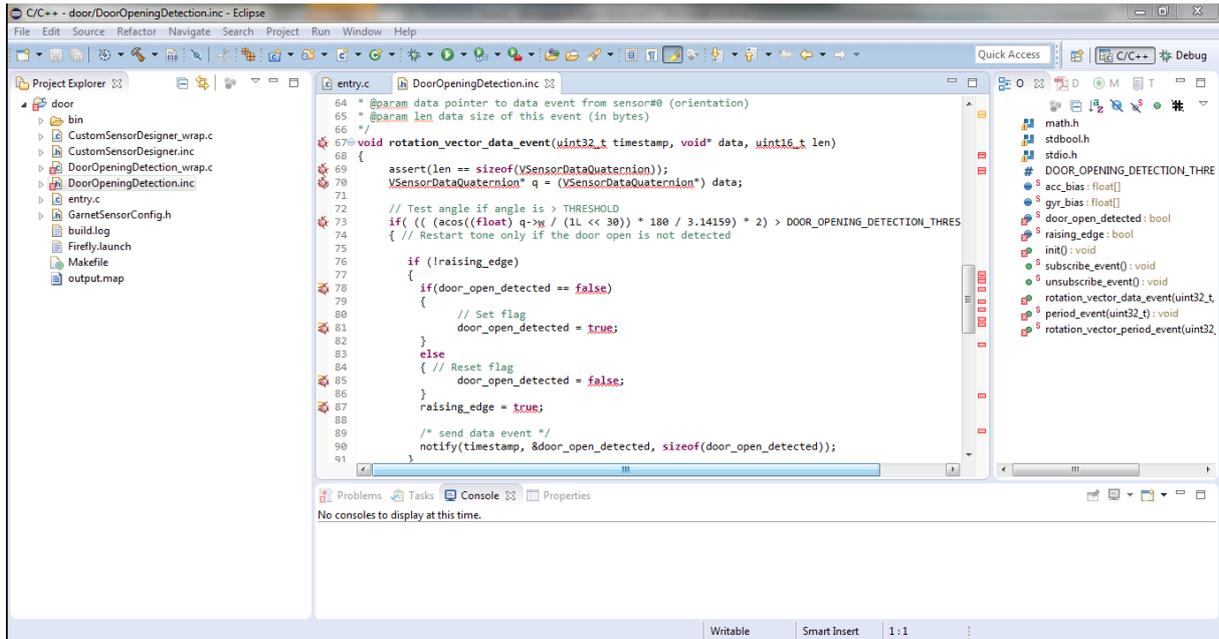


Figure 13 – Eclipse IDE Project View

The debug session is already configured for the project in the Firefly debug configuration. The debugger connects to the target through the J-Link GDB server on the running target. To access the predefined debug configuration, you have to select it the first time you start the debugging.

Click Run->Debug Configuration On the left side of the window, select GDB SEGGER J-Link Debugging -> **Firefly**, then **Debug**.

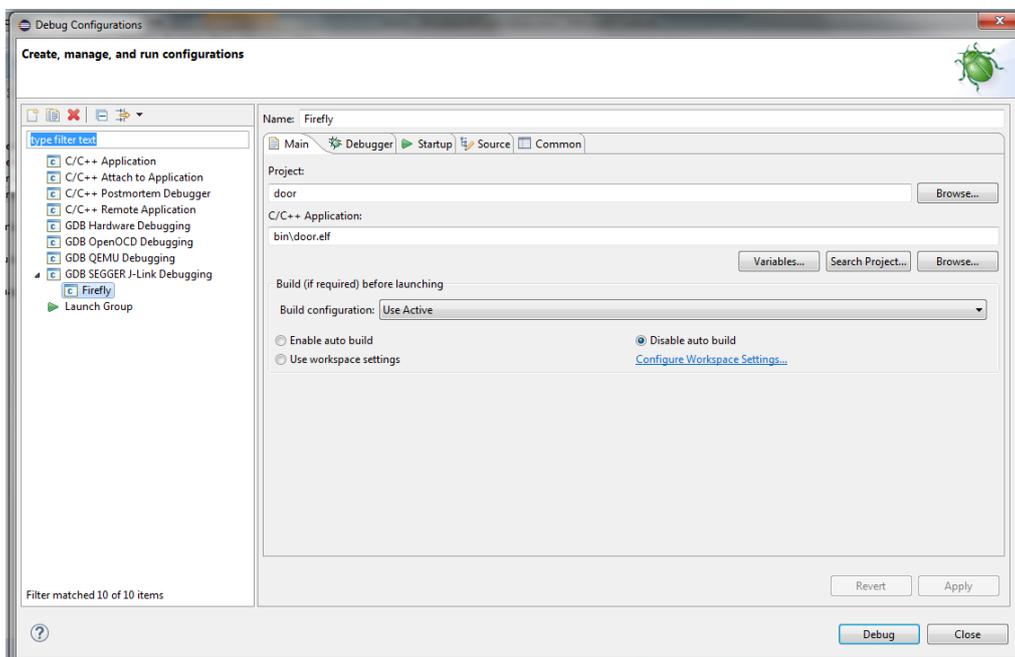


Figure 14 - Select the Firefly Debug Configuration

Now you can see the Eclipse Debug perspective. The execution of the application can now be controlled from the J-Link debugger in Eclipse IDE. Press Suspend/Resume button to stop/start

running the application. Press the Terminate button to close the debug session. You can also put breakpoints in your code in double-click in code margin.

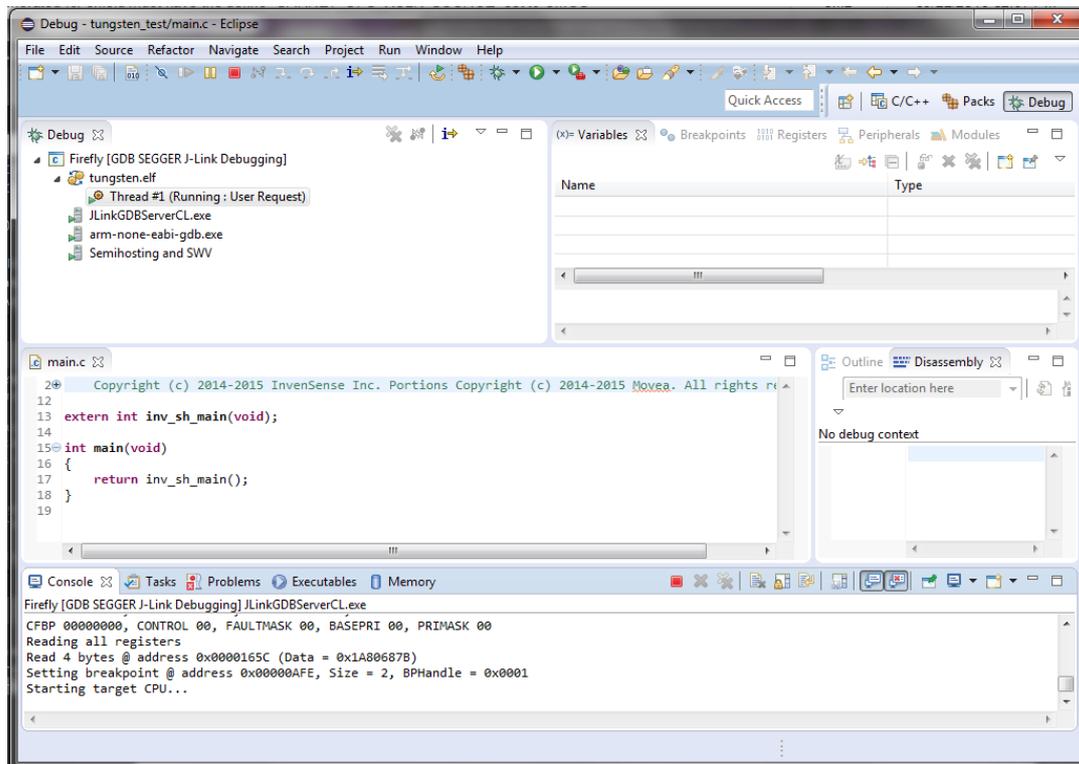


Figure 15 - Running the Project from Debugger

## 5. REVISION HISTORY

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

- This information furnished by InvenSense is believed to be accurate and reliable. However, no responsibility is assumed by InvenSense for its use, or for any infringements of patents or other rights of third parties that may result from its use. Specifications are subject to change without notice. InvenSense reserves the right to make changes to this product, including its circuits and software, in order to improve its design and/or performance, without prior notice. InvenSense makes no warranties, neither expressed nor implied, regarding the information and specifications contained in this document. InvenSense assumes no responsibility for any claims or damages arising from information contained in this document, or from the use of products and services detailed therein. This includes, but is not limited to, claims or damages based on the infringement of patents, copyrights, mask work and/or other intellectual property rights.
- Certain intellectual property owned by InvenSense and described in this document is patent protected. No license is granted by implication or otherwise under any patent or patent rights of InvenSense. This publication supersedes and replaces all information previously supplied. Trademarks that are registered trademarks are the property of their respective companies. InvenSense sensors should not be used or sold in the development, storage, production or utilization of any conventional or mass-destructive weapons or for any other weapons or life threatening applications, as well as in any other life critical applications such as medical equipment, transportation, aerospace and nuclear instruments, undersea equipment, power plant equipment, disaster prevention and crime prevention equipment.
- ©2015 InvenSense, Inc. All rights reserved. InvenSense, Sensing Everything, MotionTracking, MotionProcessing, MotionProcessor, MotionFusion, MotionApps, DMP, and the InvenSense logo are trademarks of InvenSense, Inc. Other company and product names may be trademarks of the respective companies with which they are associated.

©2015 InvenSense, Inc. All rights reserved.