



STMicroelectronics SensorTile Tutorial: Introduction to Motion Data Acquisition via BLE Communication



Table of Contents

1. INTRODUCTION TO THIS TUTORIAL.....	3
1.1. LIST OF REQUIRED EQUIPMENT AND MATERIALS.....	3
1.2. PREREQUISITE TUTORIALS.....	3
2. INTRODUCTION TO FIRMWARE FP-SNS-ALLMEMS1	4
3. REQUEST MOTION DATA USING GATTTOOL.....	14
4. SAVE REQUESTED MOTION DATA TO TEXT FILE.....	18



1. Introduction to This Tutorial

In *STMicroelectronics SensorTile Tutorial: BLE Communication via BlueZ and the GATT Profile*, we introduced how to use BlueZ gatttool utility to read environmental data from SensorTile Bluetooth Generic Attribute Profile.

In this tutorial, we will first introduce a new firmware, FP-SNS-ALLMEMS1, to support motion data acquisition via BLE communication.

The Tutorial steps provide:

1. An introduction to download, modify and flash advanced firmware, FP-SNS-ALLMEMS1.
2. An introduction to request motion data using gatttool
3. An introduction to save requested motion data to text file

For more information regarding the SensorTile board, please open the following link.

www.st.com/sensortile

For more information regarding Bluetooth, please refer to Bluetooth official website.

1.1. List of Required Equipment and Materials

- 1) 1x STMicroelectronics SensorTile kit.
- 2) 1x STMicroelectronics Nucleo Board.
- 3) 1x Personal Computer with two USB type-A inputs OR you must have a powered USB hub.
- 4) 1x USB 2.0 A-Male to Micro-B Cable (micro USB cable).
- 5) 1x USB 2.0 A-Male to Mini-B Cable (mini USB cable).
- 6) Network access to the Internet.
- 7) A Linux machine (BeagleBone, Intel Edison, Rpi, and etc)

1.2. Prerequisite Tutorials

It is recommended that users have completed and are familiar with the contents of the following tutorials before proceeding.

1. STMicroelectronics SensorTile Tutorial: BLE Communication via BlueZ and the GATT Profile



2. Introduction to Firmware FP-SNS-ALLMEMS1

FP-SNS-ALLMEMS1 firmware package enables real-time motion data streaming via BLE communication. To download FP-SNS-ALLMEMS1 firmware, please refer to the following link: www.st.com/content/st_com/en/premium-content/sensortile-curriculum-fp-sns-allmems1_firmware_zip.html

Fill out the form and a download link will be sent to your email.

1. Download FP-SNS-ALLMEMS1 firmware v3.1 from the google drive link above.



SensorTile Curriculum - stsw-stlkt01_firmware_v1_3_1_rc1_zip

Email Address *

example@ucla.edu

Country *

United States

First Name

Last Name

Zip Code

90024

☐ Please keep me informed about ST's latest news

☐ I have read and accepted the [Terms of Use](#) and [Privacy Policy](#) *


 SUBMIT

Figure 1: Download FP-SNS-ALLMEMS1 Firmware



2. Extract *en.fp-sns-allmems1_firmware.zip* to proper directory and you should get a folder named *STM32CubeFunctionPack_ALLMEMS1_V3.1.0*, which contains the FP-SNS-ALLMEMS1 firmware. See Figure 2.

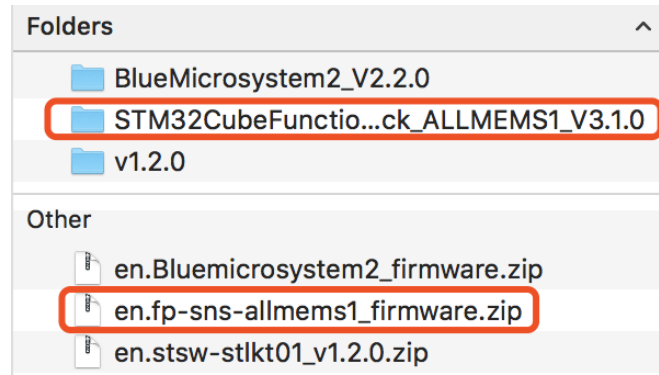


Figure 2: Extract FP-SNS-ALLMEMS1 firmware

3. Open the IDE (Eclipse or System WorkBench) on your personal computer. Once the IDE is open, **remove** your current active project in your IDE and then **import** the FP-SNS-ALLMEMS1 project by selecting *STM32CubeFunctionPack_ALLMEMS1_V3.1.0* as root directory and **unchecking** the first four projects, so only the project with 'SensorTile' in the name will be imported. See Figure 3.

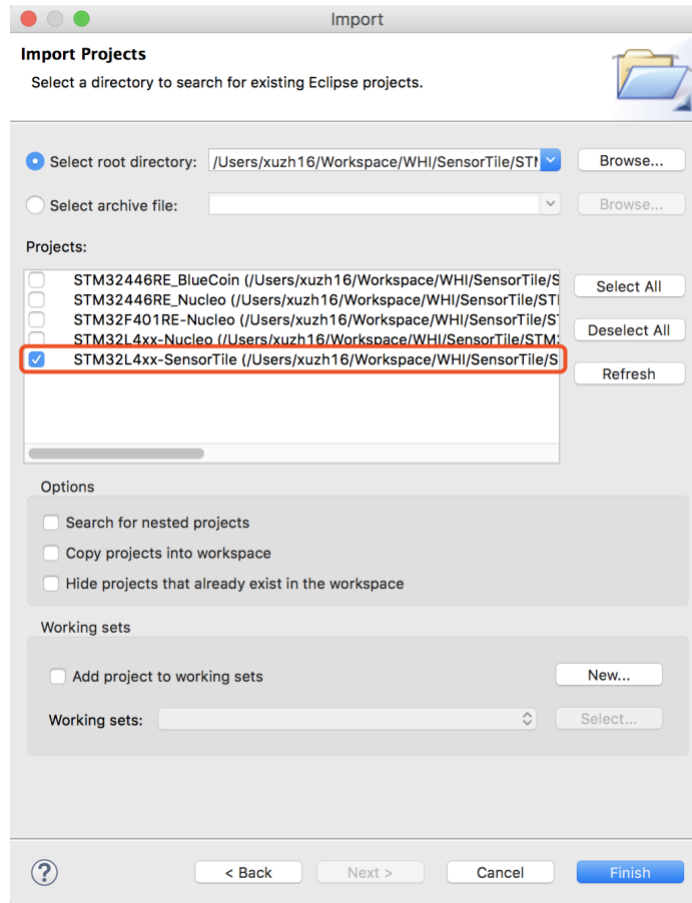


Figure 3: Import FP-SNS-ALLMEMS1 project from folder
STM32CubeFunctionPack_ALLMEMS1_V3.1.0

4. Once you successfully import the FP-SNS-ALLMEMS1 project. Open *main.c* and *main.h* according to Figure 4. Ignore the Error message in the Problems window.

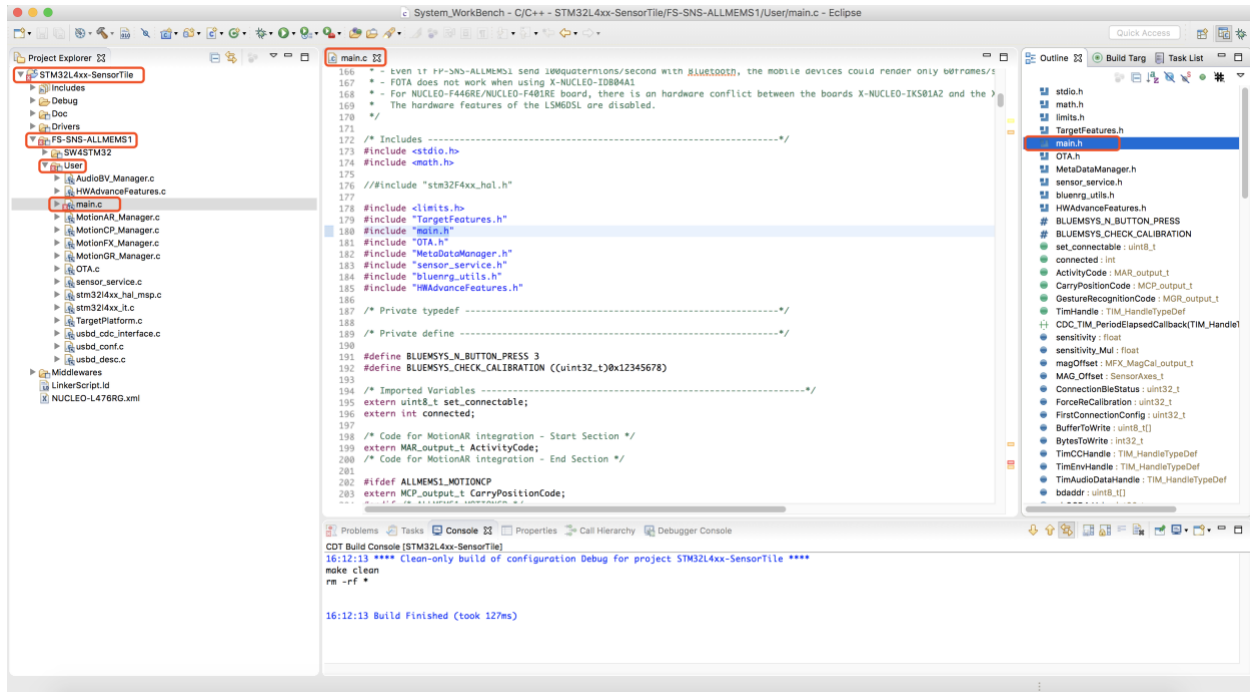


Figure 4: Open main.c and main.h in FP-SNS-ALLMEMS1 project

5. Navigate to function *SendMotionData()* at line 940 in *main.c*. Add code as highlighted in Figure 5. New added code will support motion data in USB debug interface. **Save** the mortification.



```

main.c  main.h  ALLMEMS1_config.h
933  BSP_ACCELER0_Get_Axes(TargetBoardFeatures.HandleAccSensor,&ACC_Value);
934
935  /* Read the Magneto values */
936  BSP_MAGNETO_Get_Axes(TargetBoardFeatures.HandleMagSensor,&MAG_Value);
937
938  /* Read the Gyro values */
939  BSP_GYRO_Get_Axes(TargetBoardFeatures.HandleGyroSensor,&GYR_Value);
940
941  #ifdef ALLMEMS1_DEBUG_NOTIFY_TRANSMISSION
942      int16_t acc_x_to_send, acc_y_to_send, acc_z_to_send;
943      int16_t gyr_x_to_send = 0, gyr_y_to_send = 0, gyr_z_to_send = 0;
944      int16_t mag_x_to_send, mag_y_to_send, mag_z_to_send;
945
946      acc_x_to_send = ACC_Value.AXIS_X;
947      acc_y_to_send = ACC_Value.AXIS_Y;
948      acc_z_to_send = ACC_Value.AXIS_Z;
949      ALLMEMS1_PRINTF("ACC_X=%d ",acc_x_to_send);
950      ALLMEMS1_PRINTF("ACC_Y=%d ",acc_y_to_send);
951      ALLMEMS1_PRINTF("ACC_Z=%d ",acc_z_to_send);
952
953      gyr_x_to_send = GYR_Value.AXIS_X / 100;
954      gyr_y_to_send = GYR_Value.AXIS_Y / 100;
955      gyr_z_to_send = GYR_Value.AXIS_Z / 100;
956      ALLMEMS1_PRINTF("GYR_X=%d ",gyr_x_to_send);
957      ALLMEMS1_PRINTF("GYR_Y=%d ",gyr_y_to_send);
958      ALLMEMS1_PRINTF("GYR_Z=%d ",gyr_z_to_send);
959
960      mag_x_to_send = MAG_Value.AXIS_X;
961      mag_y_to_send = MAG_Value.AXIS_Y;
962      mag_z_to_send = MAG_Value.AXIS_Z;
963      ALLMEMS1_PRINTF("MAG_X=%d ",mag_x_to_send);
964      ALLMEMS1_PRINTF("MAG_Y=%d ",mag_y_to_send);
965      ALLMEMS1_PRINTF("MAG_Z=%d ",mag_z_to_send);
966
967      ALLMEMS1_PRINTF("\r\n");
968  #endif /* ALLMEMS1_DEBUG_NOTIFY_TRANSMISSION */
969
970  AccGyroMag_Update(&ACC_Value,&GYR_Value,&MAG_Value);

```

Figure 5: Modify SendMotionData() in main.c



6. Open *ALLMEMS1_config.h* from *main.h* according to Figure 6.

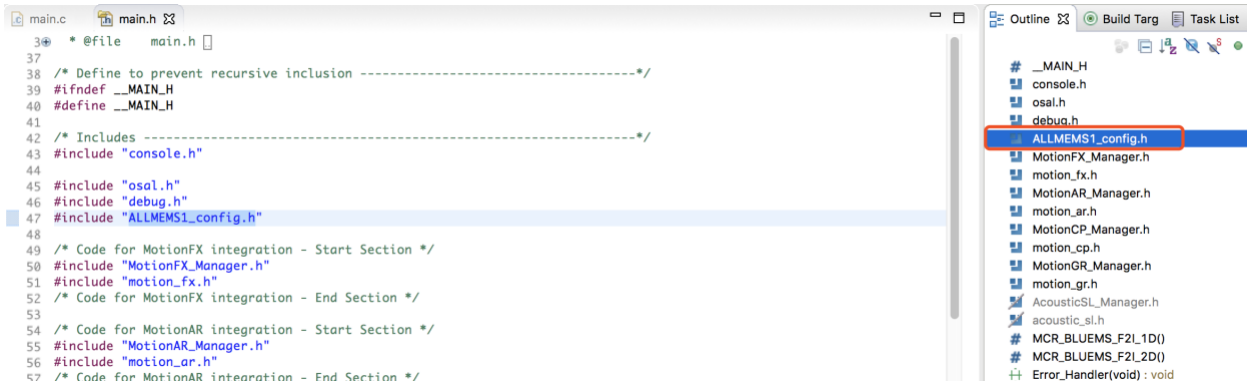


Figure 6: Open *ALLMEMS1_config.h* in FP-SNS-ALLMEMS1 project

7. Modify **line 96** in *ALLMEMS1_config.h* such that it matches Figure 7. This will enable the USB debug interface for the FP-SNS-ALLMEMS1 firmware.

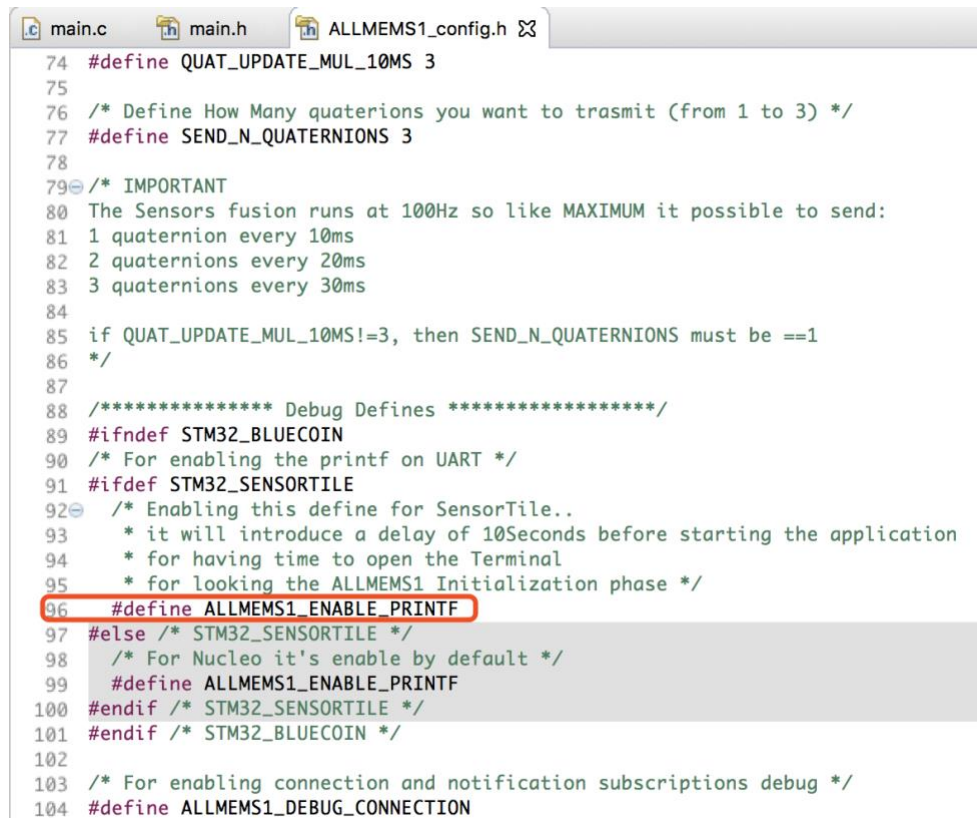


Figure 7: Modify *ALLMEMS1_config.h* to enable USB debug interface for FP-SNS-ALLMEMS1.

8. **Save** the modification. **Terminate** and **remove** all previous applications from the SensorTile board. Compile and run the FP-SNS-ALLMEMS1 application on the SensorTile board in debug mode.



9. Examine the LED on SensorTile. Your SensorTile should start to blink once every second, which indicates that SensorTile is running FP-SNS-ALLMEMS1 firmware properly.
10. Connect your SensorTile with your computer and use terminal screen command (MAC user) or Putty serial connection (Windows user) to access the USB debug interface of FP-SNS-ALLMEMS1 firmware as we accessed the USB debug interface of BLE_SampleAPP firmware in ***STMicroelectronics SensorTile Tutorial: BLE Communication via BlueZ and the GATT Profile***. The USB debug interface is indicated as Figure 8. Make sure you do not close the debug interface because you will use it in the following session.



```

Debug — screen /dev/cu.usbmodem144131 9600 ▶ SCREEN — 80x...
STMicroelectronics FP-SNS-ALLMEMS1:
  Version 3.1.0
  STM32476RG-SensorTile board

OK Accelero Sensor
OK Gyroscope Sensor
OK Magneto Sensor
Error Humidity Sensor
OK Temperature Sensor1
Error Temperature Sensor2
OK Pressure Sensor
Enabled Accelero Sensor
Enabled Gyroscope Sensor
Enabled Magneto Sensor
Enabled Temperature Sensor1
Enabled Pressure Sensor
Battery not present

Meta Data Manager read from Flash
Meta Data Manager version=0.9.0
  Generic Meta Data found:
    CALIBRATION Size=120 [bytes]

  (HAL 1.7.1_0)
  Compiled Sep 26 2017 16:35:09 (openstm32)
  Send Every 30ms 3 Short precision Quaternions
  Send Every 500ms Temperature/Humidity/Pressure
  Send Every 50ms Acc/Gyro/Magneto
  Send Every 50ms dB noise

Debug Connection Enabled
Debug Notify Transmission Enabled

SERVER: BLE Stack Initialized
  Board type=IDB05A1 HWver=49, FWver=7.2.c
  BoardName= AM1V310
  BoardMAC = c0:83:2c:31:25:48

HW & SW Service W2ST added successfully
Console Service W2ST added successfully
Config Service W2ST added successfully

ERROR: BootLoader NOT Compliant with FOTA procedure

Initialized ST MotionFX v2.0.0
Magneto Calibration Not present
Initialized ST MotionAR v2.0.0
Initialized ST MotionCP v2.0.0
Initialized ST MotionGR v2.0.0
Initialized ST BlueVoiceADPCM v2.0.0
>>>>>CONNECTED 5c:f8:21:d6:9d:2d

```

Figure 8: FP-SNS-ALLMEMS1 USB debug interface

11. Connect your BeagleBone with your computer and log into BeagleBone.
12. In BeagleBone, type command **\$ bluetoothctl** to use BlueZ in interactive mode.



13. As instructed in *STMicroelectronics SensorTile Tutorial: Introduction to Bluetooth Low Energy (BLE) Interfaces*, discover, pair, and **disconnect** your SensorTile device in bluetoothctl. See Figure 9. The device MAC address with name **AM1V310** is your SensorTile MAC address. You should always use this MAC address for BLE communication with SensorTile.

```
[bluetooth]# scan on
Discovery started
[CHG] Controller 5C:F8:21:D6:9D:2D Discovering: yes
[NEW] Device C0:83:2C:31:25:48 AM1V310
[NEW] Device 61:E5:99:7F:1A:BE 61-E5-99-7F-1A-BE
[NEW] Device 7A:25:D5:60:F5:4F 7A-25-D5-60-F5-4F
[NEW] Device 14:99:E2:17:33:1F 14-99-E2-17-33-1F
[bluetooth]# scan off
[CHG] Device 14:99:E2:17:33:1F RSSI is nil
[CHG] Device 7A:25:D5:60:F5:4F RSSI is nil
[CHG] Device 61:E5:99:7F:1A:BE RSSI is nil
[CHG] Device C0:83:2C:31:25:48 RSSI is nil
Discovery stopped
[CHG] Controller 5C:F8:21:D6:9D:2D Discovering: no
[bluetooth]# pair C0:83:2C:31:25:48
Attempting to pair with C0:83:2C:31:25:48
[CHG] Device C0:83:2C:31:25:48 Connected: yes
[CHG] Device C0:83:2C:31:25:48 UUIDs:
00000000-0001-11e1-9ab4-0002a5d5c51b
00000000-000e-11e1-9ab4-0002a5d5c51b
00000000-000f-11e1-9ab4-0002a5d5c51b
00001800-0000-1000-8000-00805f9b34fb
00001801-0000-1000-8000-00805f9b34fb
[CHG] Device C0:83:2C:31:25:48 Paired: yes
Pairing successful
[bluetooth]# disconnect C0:83:2C:31:25:48
Attempting to disconnect from C0:83:2C:31:25:48
Successful disconnected
[CHG] Device C0:83:2C:31:25:48 Connected: no
```

Figure 9: Use bluetoothctl to discover, pair, and disconnect SensorTile (AM1V310)



14. Use command **info** in bluetoothctl to check that your SensorTile is paired but **disconnected**. See Figure 10.

```
[bluetooth]# info C0:83:2C:31:25:48
Device C0:83:2C:31:25:48
  Name: AM1V310
  Alias: AM1V310
  Paired: yes
  Trusted: no
  Blocked: no
  Connected: no
  LegacyPairing: no
  UUID: Vendor specific (00000000-0001-11e1-9ab4-0002a5d5c51b)
  UUID: Vendor specific (00000000-000e-11e1-9ab4-0002a5d5c51b)
  UUID: Vendor specific (00000000-000f-11e1-9ab4-0002a5d5c51b)
  UUID: Generic Access Profile (00001800-0000-1000-8000-00805f9b34fb)
  UUID: Generic Attribute Profile (00001801-0000-1000-8000-00805f9b34fb)
```

Figure 10: Use bluetoothctl to check if the SensorTile is paired and disconnected

15. Copy the MAC address of SensorTile and **exit** bluetoothctl. Meanwhile, you can switch to USB debug interface to see debug information.
16. In BeagleBone, use command **\$ gatttool -b <MAC_ADDRESS_SENSORTILE> -t random -I** to run gatttool in interactive mode as instructed in *STMicroelectronics SensorTile Tutorial: BLE Communication via BlueZ and the GATT Profile*. See Figure 11.

```
root@beaglebone:~# gatttool -b C0:83:2C:31:25:48 -t random -I
```

Figure 11: Use gatttool with proper options in interactive mode

17. In gatttool, use command **connect** to connect BeagleBone with SensorTile. If the connection is successful, the MAC address should turn into color of blue. See Figure 12. You can also see the successful connection information in USB debug interface.

```
[C0:83:2C:31:25:48][LE]> connect
Attempting to connect to C0:83:2C:31:25:48
Connection successful
[C0:83:2C:31:25:48][LE]>
```

Figure 12: Connect with SensorTile in gatttool interactive mode

18. In gatttool, 1) use command **primary** to discover all the primary services; 2) use command **characteristics** to discover all the characteristics; 3) use command **char-desc** to discover all the characteristic descriptors as instructed in *STMicroelectronics SensorTile Tutorial: BLE Communication via BlueZ and the GATT Profile*.



3. Request Motion Data Using Gatttool

In this session, you will use gatttool to request motion data.

In the previous session, you have discovered all the characteristic descriptors. Handle **0x0012** is client characteristic configuration handle to request motion data.

1. Make sure that the SensorTile is connected with BeagleBone in gatttool.
2. In gatttool, use command **char-write-req 0012 0100** to modify the characteristic value of handle 0x0012 and therefore enable the notification of motion data (handle 0x0011) on SensorTile. After you enter the command, SensorTile will start to send motion data every 50ms as indicated in the USB debug interface and BeagleBone will listen to the data transmission in gatttool. See Figure 13.

```
[C0:83:2C:31:25:48][LE]> char-write-req 0012 0100
Characteristic value was written successfully
Notification handle = 0x0011 value: d7 32 e5 ff a3 ff fa 03 00 00 f2 ff fc ff 0a
ff 04 00 bb fe
Notification handle = 0x0011 value: de 32 e6 ff a1 ff f4 03 ff ff f2 ff fc ff 09
ff fc ff b9 fe
Notification handle = 0x0011 value: e4 32 e5 ff a0 ff f2 03 fe ff f2 ff fc ff 0c
ff fa ff b8 fe
Notification handle = 0x0011 value: ea 32 e6 ff 9f ff f7 03 fe ff f2 ff fc ff 0f
ff 07 00 b8 fe
Notification handle = 0x0011 value: f0 32 e5 ff a0 ff f8 03 ff ff f2 ff fc ff 09
ff 03 00 bb fe
Notification handle = 0x0011 value: f7 32 e5 ff a0 ff f6 03 ff ff f2 ff fc ff 0a
ff 0a 00 c2 fe
Notification handle = 0x0011 value: fd 32 e5 ff a1 ff f4 03 ff ff f1 ff fc ff 12
ff 01 00 be fe
Notification handle = 0x0011 value: 03 33 e5 ff a0 ff f5 03 ff ff f1 ff fc ff 09
ff 06 00 be fe
Notification handle = 0x0011 value: 09 33 e6 ff a0 ff f5 03 fe ff f1 ff fb ff 0c
ff 00 00 b9 fe
```

Figure 13: char-write-req to enable motion data notification in gatttool

3. In gatttool, use command **char-write-req 0012 0000** to disable the notification of motion data. See Figure 14.

```
[C0:83:2C:31:25:48][LE]> char-write-req 0012 0000
```

Figure 14: char-write-req to disable motion data notification in gatttool



4. Switch to the USB debug interface. You will see similar debug information that as Figure 15. You may only see the numbers and “--->Acc/Gyro/Mag = OFF” because SensorTile sends motion data every 50 ms.

```

--->Acc/Gyro/Mag= ON
ACC_X=-27 ACC_Y=-93 ACC_Z=1018 GYR_X=0 GYR_Y=-14 GYR_Z=-4 MAG_X=-246 MAG_Y=4 MAG_Z=-325
ACC_X=-26 ACC_Y=-95 ACC_Z=1012 GYR_X=-1 GYR_Y=-14 GYR_Z=-4 MAG_X=-247 MAG_Y=-4 MAG_Z=-327
ACC_X=-27 ACC_Y=-96 ACC_Z=1010 GYR_X=-2 GYR_Y=-14 GYR_Z=-4 MAG_X=-244 MAG_Y=-6 MAG_Z=-328
ACC_X=-26 ACC_Y=-97 ACC_Z=1015 GYR_X=-2 GYR_Y=-14 GYR_Z=-4 MAG_X=-241 MAG_Y=7 MAG_Z=-328
ACC_X=-27 ACC_Y=-96 ACC_Z=1016 GYR_X=-1 GYR_Y=-14 GYR_Z=-4 MAG_X=-247 MAG_Y=3 MAG_Z=-325
ACC_X=-27 ACC_Y=-96 ACC_Z=1014 GYR_X=-1 GYR_Y=-14 GYR_Z=-4 MAG_X=-246 MAG_Y=10 MAG_Z=-318
ACC_X=-27 ACC_Y=-95 ACC_Z=1012 GYR_X=-1 GYR_Y=-15 GYR_Z=-4 MAG_X=-238 MAG_Y=1 MAG_Z=-322
ACC_X=-27 ACC_Y=-96 ACC_Z=1013 GYR_X=-1 GYR_Y=-15 GYR_Z=-4 MAG_X=-247 MAG_Y=6 MAG_Z=-322
ACC_X=-26 ACC_Y=-96 ACC_Z=1013 GYR_X=-2 GYR_Y=-15 GYR_Z=-5 MAG_X=-244 MAG_Y=0 MAG_Z=-327
ACC_X=-28 ACC_Y=-94 ACC_Z=1014 GYR_X=0 GYR_Y=-14 GYR_Z=-4 MAG_X=-253 MAG_Y=-3 MAG_Z=-328
ACC_X=-29 ACC_Y=-93 ACC_Z=1014 GYR_X=0 GYR_Y=-15 GYR_Z=-4 MAG_X=-246 MAG_Y=0 MAG_Z=-319
ACC_X=-28 ACC_Y=-96 ACC_Z=1014 GYR_X=-1 GYR_Y=-14 GYR_Z=-4 MAG_X=-246 MAG_Y=7 MAG_Z=-327
ACC_X=-27 ACC_Y=-97 ACC_Z=1015 GYR_X=-1 GYR_Y=-15 GYR_Z=-4 MAG_X=-238 MAG_Y=1 MAG_Z=-325
ACC_X=-27 ACC_Y=-96 ACC_Z=1014 GYR_X=-1 GYR_Y=-14 GYR_Z=-4 MAG_X=-243 MAG_Y=4 MAG_Z=-327
ACC_X=-27 ACC_Y=-99 ACC_Z=1016 GYR_X=-3 GYR_Y=-14 GYR_Z=-5 MAG_X=-243 MAG_Y=6 MAG_Z=-324
ACC_X=-23 ACC_Y=-96 ACC_Z=1009 GYR_X=0 GYR_Y=-15 GYR_Z=-4 MAG_X=-249 MAG_Y=6 MAG_Z=-325
--->Acc/Gyro/Mag= OFF
  
```

Figure 15: USB debug interface for enable/disable motion data notification

5. The motion data sent from SensorTile are parsed in similar method as environmental data in the BLE_SampleApp firmware. We can still compare Figure 13 (motion data in gatttool) and Figure 15 (motion data in USB debug interface).

In Figure 13, all the data are sent in the format of hexadecimal byte quantities. The first 2 bytes indicated in the red rectangle (e1 95) are timestamp values, where e1 is the first byte and 95 is the second byte. The next 6 bytes in the blue rectangle (e5 ff a3 ff fa 03) are the 3-axis accelerometer data. The second to last 12 bytes in the yellow rectangle (00 00 f2 ff fc ff) are 3-axis gyroscope data. The last 6 bytes in the green rectangle (0a ff 04 00 bb fe) are 3-axis magnetometer data. In each case, the X-axis sensor data value appears in the first two



bytes (corresponding to a 16b encoding of the signal) while the Y and Z axis values appear in the following two byte pairs, respectively.

Accelerometer data, gyroscope data and magnetometer data have similar data formats composed of the 3-axis data values for each sensor. Therefore, we will use accelerometer data as an example to interpret the motion data. For example, 3-axis accelerometer data, the first 2 bytes (e5 ff) are x-axis data. The second 2 bytes (a3 ff) are y-axis data. The last 2 bytes (fa 03) are z-axis data.

Similar to BLE_SampleAPP firmware environmental data, motion data sent by notification should be interpreted for purposes of determining signal value with the rule that the second byte is the high order byte and the first byte of the pair is the low order byte. This means that the x-axis acceleration value is computed from this data as 0xffe5, the y-axis acceleration value is 0xffa3, and the z-axis acceleration value is 0x03fa.

Environmental data are **unsigned** because all data are positive. However, motion data is encoded to represent both positive and negative values. Also, for the acceleration gain configuration here, the magnitude of acceleration is represented in milli-g. Specifically, the most positive acceleration value indicated by the accelerometer will be decimal 32767 (corresponding to the hexadecimal value of 0x7FFF and represented in two's complement binary encoding as 0111 1111 1111 1111.) The most negative acceleration value indicated by the accelerometer will be decimal -32768 (corresponding to the hexadecimal value of 0x8000 and represented in as 1000 0000 0000 0000.)

Therefore, in the example above, the x-axis acceleration decimal value is -27 (0xffe5); the y-axis acceleration decimal value is -93 (0xffa3); the z-axis acceleration decimal value is 1018 (0x03fa), which is exactly in agreement with the motion data observed from the USB debug interface value above.

The x-axis gyroscope data are 0 (0x0000); the y-axis gyroscope data are -14 (0xff2); the z-axis gyroscope data are -4 (0xffc).

The x-axis magnetometer data are -246 (0xff0a); the y-axis magnetometer data are 4 (0x0004); the z-axis magnetometer data are -325 (0xfebb).

The acceleration data are provided in units of **milli-g**. The gyroscope data are provided in units of **degree per second** (dps) and the magnetometer data are provided in units of **milli-Gauss**.

The accelerometer and magnetometer physical sensor signal quantities are directly indicated by their respective integer data values and physical units as above. However, the



gyroscope data signal quantities are encoded such these are equal to the corresponding data value integer multiplied by a gain factor of 10. This provides additional resolution for indicating the physical signal with the corresponding integer data value. For example, if the x-axis gyroscope data value is 155, then this represents a physical signal of 15.5 degree per second angular velocity of the x axis.

Thus, the motion data may be computed as below:

x-axis acceleration data: -27 milli-g
y-axis acceleration data: -93 milli-g
z-axis acceleration data: 1018 milli-g

x-axis gyroscope data: 0.0 dps
y-axis gyroscope data: -1.4 dps
z-axis gyroscope data: -0.4 dps

x-axis magnetometer data: -246 milli-Gauss
y-axis magnetometer data: 4 milli-Gauss
z-axis magnetometer data: -325 milli-Gauss

Conversion between the hexadecimal value and decimal signal for accelerometer, gyroscope, and magnetometer are accomplished in this way. First, the properly ordered hexadecimal two byte sensor values may be converted to decimal. For decimal values that are greater than 32767, correction of the offset due to two's complement encoding is performed by subtracting 65536. For decimal values less than or equal to 32767 no offset correction is applied.



4. Save Requested Motion Data to Text File

In this session, you will use `gatttool` to request motion data and then save the data to text file for future development.

1. In `gatttool`, use command **`disconnect`** and **`exit`** to disconnect BeagleBone and SensorTile and exit `gatttool` as Figure 16.

```
[C0:83:2C:31:25:48][LE]> disconnect
[C0:83:2C:31:25:48][LE]> exit
root@beaglebone:~#
```

Figure 16: Disconnect SensorTile and exit `gatttool`

2. Make sure that the SensorTile is successfully paired but disconnected. In BeagleBone, use command **`$ gatttool -b <MAC_ADDRESS_SENSORTILE> -t random --char-write-req --handle=0x0012 --value=0100 --listen > motion_data.txt`**, wait 5 seconds, use **`Ctrl + C`** to terminate the `gatttool`. See Figure 17.

```
root@beaglebone:~# gatttool -b C0:83:2C:31:25:48 -t random --char-write-req --ha
ndle=0x0012 --value=0100 --listen > motion_data.txt
^C
root@beaglebone:~#
```

Figure 17: Request motion data and save data in `motion_data.txt`

3. Use command **`$ ls`** to check if `motion_data.txt` is saved. Now, you have saved the 5 seconds motion data in `motion_data.txt` file for future use.
4. You can use command **`$ cat motion_data.txt`** to quickly check the data in `motion_data.txt`.