TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Vladimir Semjonov 142944IALB

MSP432 BASED BLUETOOTH DATALOGGER FOR ANDROID SMART DEVICES

Bachelor's thesis

Supervisor: Eero Haldre

Certified Engineer

Tallinn 2018

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Vladimir Semjonov 142944IALB

MSP432 PÕHINE BLUETOOTH ANDMELOGGER ANDROID NUTISEADMETELE

Bakalaureuse töö

Juhendaja: Eero Haldre

Dipl. Insener

Tallinn 2018

Author’s declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Vladimir Semjonov

21.05.2018

Abstract

This thesis is developed for Thomas Johann Seebeck electronics institute as a Bluetooth Low Energy communication solution between Android smart device and Texas Instruments microcontroller combination. The thesis project consists of an example program from the "Texas Instruments" organization for MSP432 microcontroller and companion booster packs, an Android application for smart devices and a guide for students' successful understanding and learning of the communication solution. The thesis gives an overview the principles of Android application development for communication with microcontroller combination. The Android application is an essential tool for issuing sensor information, which makes possible to monitor the information coming from the sensors and examine the application's operation. This graduation thesis is useful for students, who are interested in exploring the world of microcontrollers and want to develop the system set up to achieve institute's goals.

This thesis is written in English and is 64 pages long, including 5 chapters, 33 figures and 4 tables.

Annotatsioon

MSP432 PÕHINE BLUETOOTH ANDMELOGGER ANDROID NUTISEADMETELE

Antud lõputöö on loodud Thomas Johann Seebecki elektroonikainstituudile Texas Instruments mikrokontrolleri MSP432 ja Android nutiseadme vahelise side katsetamiseks Bluetooth Low Energy interfeisi kaudu. Lõputöö projekt koosneb firma „Texas Instruments“ poolt loodud näidisprogrammist mikrokontrollerile, Android rakendusest ja koostatud juhendist üliõpilastele. Töös antakse ülevaade mikrokontrolleri tegevusest ja loodud Android rakendusest. Android rakendus on andurite informatsiooni esitamiseks loodud töörist, millega on võimalik jälgida anduritest tulevat informatsiooni ja uurida rakenduse töötamist. Antud lõputöö tuleb kasuks üliõpilastele, kes on huvitatud mikrokontrollerite arendustöödest ja soovivad loodud vahendit edasi arendada.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 64 leheküljel, 5 peatükki, 33 joonist, 4 tabelit.

List of abbreviations and terms

ARIB	*Association of Radio Industries and Businesses*
ARM	*Advanced RISC Machine*
BLE	*Bluetooth Low Energy*
CCS	*Code Composer™ Studio*
CE	*Conformité Européenne*
CPU	*Central Processing Unit*
DSP	*Digital Signal Processor*
EMC	*Electromagnetic Compatibility*
EVM	*Evaluation Module*
FCC	*Federal Communications Committee*
I/O	*Input / Output*
I^2C	*Inter-Integrated Circuit*
IC	*Industry Canada*
IDE	*Integrated Development Environment*
IR	*Infrared*
IrDA	*Infrared Data Association*
JTAG	*Joint Test Action Group (named after group which codified it)*
LED	*Light Emitting Diode*
LGA	*Land Grid Array*
LLC	*Limited Liability Company*
LPM	*Low Power Mode*
MCU	*Micro Controller Unit*
OS	*Operating System*
PWM	*Pulse-Width-Modulation*
RISC	*Reduced Instruction Set Computer*
RTC	*Real-Time Clock*
SDK	*Software Development Kit*
SNP	*Simple Network Processor*
SPI	*Serial Peripheral Interface*
TI	*Texas Instruments*
UART	*Universal Asynchronous Receiver/Transmitter*
UI	*User Interface*
USB	*Universal Serial Bus*

Table of contents

List of figures

List of tables

1 Introduction

The development of microelectronics and widespread use of its products in industrial production and variety of management systems is one of the main science and technology development directions today. The use of microcontrollers in production increases the economic value (cost, reliability, energy consumption) of products and allows shortening the development stage and delaying the moral aging.

In the recent years, production of smart devices has become one of the main trends in microelectronics. The smart device has become relatively more comparable to personal computers performance wise and now allows people to execute activities, which were possible only on stationary device. Each smart device holds a host of microprocessors and microcontrollers. The largest part of smart devices in the world is controlled by an operating system called Android.

The main task of this bachelor thesis is to create a Bluetooth Low Energy communication solution between the microcontroller platform and a smart device with integrated Android operation system for Thomas Johann Seebeck Institute of Electronics at Tallinn University of Technology, which will be useful for further development by other students. This solution will provide a more intuitive and accessible way to learn on how connect the devices with each other and read the sensor data from application's user interface.

The "Texas Instruments MSP-EXP432P401R SimpleLink™ Microcontroller LaunchPad™ Development Kit" microcontroller is introduced as the primary hardware component. The "Texas Instruments SimpleLink™ CC2650 BoosterPack™ plug-in module" provides the existence of BLE technology in the system and "Texas Instruments SimpleLink™ Sensors BoosterPack™" provides sensors that will be used for our thesis.

2 Technical overview

The task of the technical overview is to introduce the technology used in the thesis project and review the important technical information that will concern the practical part of the thesis.

2.1 Texas Instruments MSP-EXP432P401R SimpleLink™ Microcontroller LaunchPad™ Development Kit

The SimpleLink MSP-EXP432P401R LaunchPad™ development kit is an easy-to-use evaluation module for the MSP432P401R microcontroller. It contains everything needed to start developing on the MSP432 LowPower and Performance ARM 32-bit Cortex-M4F microcontroller (MCU), including onboard debug probe for programming, debugging, and energy measurements. [1]

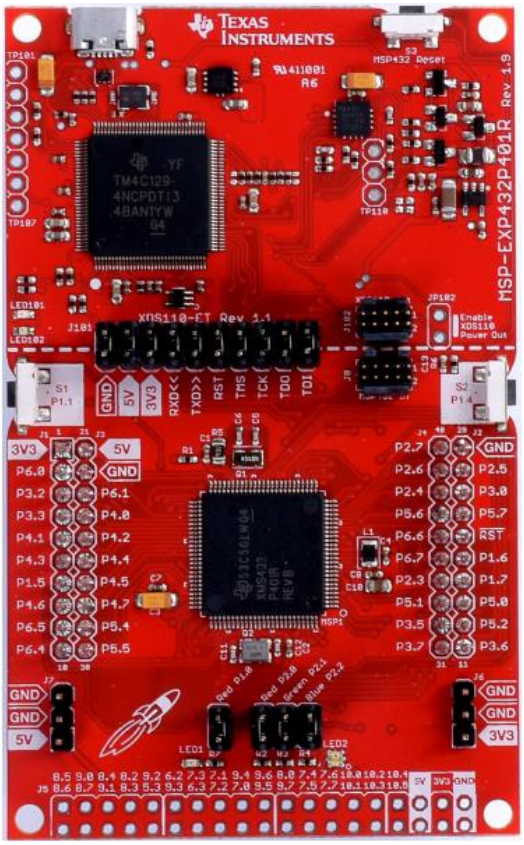


Figure 1: MSP -EXP432P401R LaunchPad™ Development Kit [1]

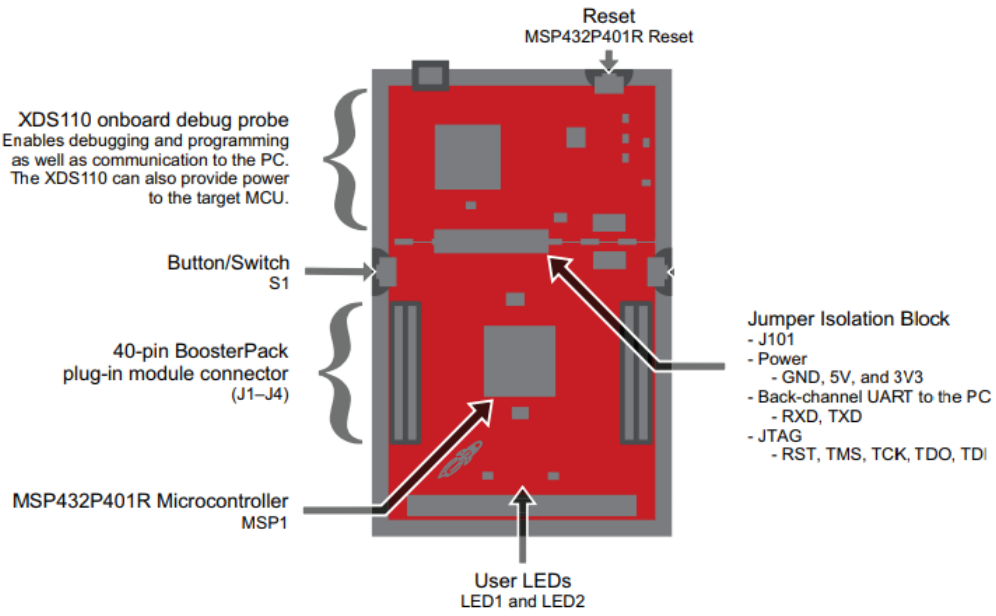


Figure 2: Overview of the EVM hardware [1]

The SimpleLink MSP432P401R MCU is the first MSP432 family device to feature ARM Cortex-M4F core. This device's features important for the thesis include:

- Up to 48-MHz system clock 32-bit ARM Cortex M4F with Floating Point Unit and DSP acceleration
- Memory: 256KB Flash, 64KB SRAM and 32KB of ROM with SimpleLink MSP432 SDK libraries
- Two buttons and two LEDs for User Interaction
- Back-channel UART via USB to PC
- Possibility to use BoosterPack™ plug-in modules to achieve environmental sensing and wireless connectivity for the thesis project

The MSP43x family started from MSP430 MCU family, which led to the development and production of MSP432. The MSP brand and architecture has always been based on low-power optimization, but the MSP432 introduced the support of high performance features.

2.1.1 XDS110-ET Onboard Debug Probe

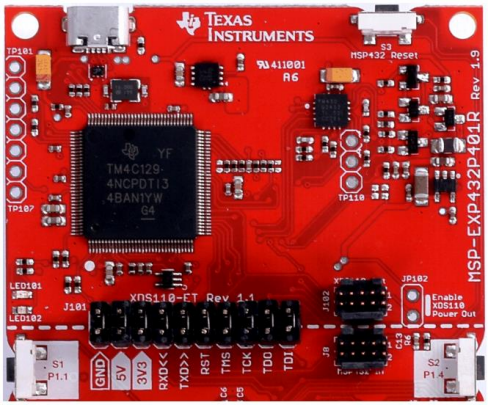


Figure 3: XDS110-ET Debug Probe [1]

XDS110-ET Debug Probe is an additional interface that Texas Instruments implemented in order to gain low-cost debug support, which satisfies the programming needs. The programmers used in the past were overly expensive and XDS110-ET Debug Probe makes development easy and very cost effective.

The Debug Probe contains an Isolation Block J101, which allows the user to connect or disconnect signals that cross from the XDS110-ET domain into MSP432P401R target domain.

In this thesis the XDS110-ET Debug Probe is being used for connecting and flashing „Texas Instruments SimpleLink™ CC2650 BoosterPack™ plug-in module" to meet the onboard software requirements of the "Texas Instruments SimpleLink™ Sensors BoosterPack™" example. The flashing process will be described in the practical part of the thesis.

2.2 Texas Instruments SimpleLink™ CC2650 BoosterPack™ plug-in module

The SimpleLink Bluetooth low energy CC2650 BoosterPack™ plug-in module offers a expedited way to provide an integrated hardware solution quickly, without having to develop a new hardware board, integrate an antenna, and obtain approval from regulatory agencies. [2]

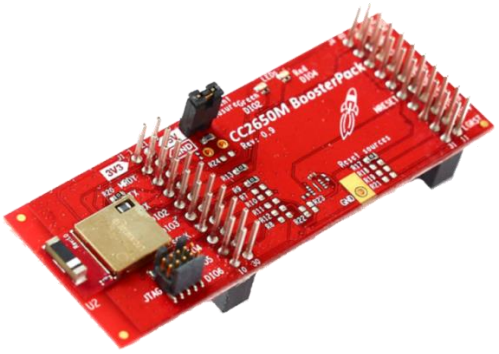


Figure 4: Texas Instruments SimpleLink™ CC2650 BoosterPack™ plug-in module [2]

This particular module contains:

- CC2650 wireless microcontroller with an integrated antenna
- ARM Cortex-M3 32-bit processor
- In-system flash memory
- Fifteen Inputs/Outputs
- Precertification for FCC/IC, CE and ARIB radio standards.

The CC2650 device contains a 32-bit ARM Cortex-M3 processor that runs at 48 MHz as the main processor and a peripheral feature set that includes ultralow power sensor controller. In featured thesis project this sensor controller is needed to enable and arrange output of analog and digital data out of sensors included in the "Texas Instruments SimpleLink™ Sensors BoosterPack™". In addition to the processor, the module has number of resistors required for balanced voltage, as well as JTAG connector for debugging and programming or flashing the device.

The guide included in thesis's appendix will include the use of JTAG connector in order to flash the CC2650 BoosterPack™ with the most recent software to achieve fault exclusion during example debugging.

2.3 Texas Instruments SimpleLink™ Sensors BoosterPack™

The Sensors BoosterPack™ kit (BOOSTXL-SENSORS) is an easy-to-use plug-in module for adding digital sensors to LaunchPad™ development kit design. [4]

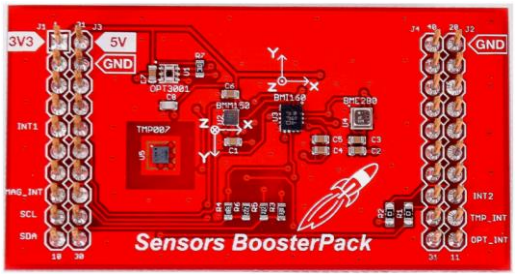


Figure 5:BOOSTXL-SENSORS BoosterPack™ Plug-in Module [4]

BOOSTXL-SENSORS BoosterPack™ Plug-in Module contains a compilation of sensors that are required for the thesis's practical part. The sensors are the following:

- Texas Instruments „OPT3001" Ambient Light Sensor
- Texas Instruments "TMP007" Contactless Temperature Sensor
- Bosch "BME280" Integrated Environmental Unit

2.3.1 Texas Instruments "OPT3001" Light Sensor

The OPT3001 is a digital ambient light sensor (ALS) that measures the intensity of light as visible by the human eye. Covering the sensor with a finger or shining a flashlight on it changes the output of the OPT3001. [4] The precision of response and intense IR rejection allows the sensor to accurately meter the intensity of light.

The OPT3001 is designed for systems that create light-based experiences for humans, and an ideal preferred replacement for photodiodes, photoresistors, or other ambient light sensors with less human eye matching and IR rejection. [5]

Important features for the thesis project:

- Precision Optical Filtering to Match Human Eye
- Measurements: 0.01 lux to 83000 lux

Parameters:

Table 1: OPT110 parameters

Spectral Bandwidth (nm)	460nm – 655nm
Supply Range (Nom)	1.6V to 3.6V
Signal Bandwidth (none)	10 samples/sec
Operating Temperature Range (C^o)	-40 to 85

2.3.2 Texas Instruments "TMP007" Temperature Sensor

The TMP007 Temperature sensor is a thermal infrared sensor, which measures the temperature of the object by sensing the infrared radiation emitted by the object. The working principle of this sensor also contains measured voltage conversion to a digital reading of the temperature, which is then sent to CC2650 BoosterPack™ and, having the notifications and "enabling" code input on point, sends the information to Android smart device.

This particular temperature sensor is produced according to "touchless" technology, meaning that the sensor measures temperature without physical contact with the object. Having resolution of 14-bit, the sensor's precision converts to 0.03125 °C, what makes it very accurate. TMP007 radiation sensitivity in the IR spectrum is measured from approximately 4- to 16-µm wavelength.

The list applications for this sensor contains mainly the noncontact temperature sensing in power relays, laser printers, HVAC comfort optimization, but the "touchless" technology enables this sensor to be used in security systems for gas concentration and flame detection.

2.3.3 Bosch "BME280" Integrated Environmental Unit

The BME280 is an integrated environmental sensor developed specifically for mobile applications where size and low power consumption are key design constraints.

The unit combines individual high linearity, high accuracy sensors for pressure, humidity and temperature in an 8-pin metal-lid 2.5 x 2.5 x 0.93 mm³ LGA package, designed for low current consumption (3.6 μA @ 1Hz), long term stability and high EMC robustness. [7]

The BME280 sensor delivers humidity and barometric pressure output for our thesis project. By virtue of humidity sensor's fast response time feature, its application to our project supplies high accuracy and more precise access to information. The pressure sensor measures barometric pressure with an optimized very low noise and high resolution output.

2.4 Android and Bluetooth Low Energy

2.4.1 Definition of Android

Android is an operating system developed by Google LLC for mobile devices. The Android OS is based on modified version of Linux kernel and other open source software for touchscreen mobile devices. In addition to this, Android OS is being used in television, car production, game consoles and other electronics.

2.4.2 Android operating system importance in thesis project

The smart device with an Android operating system will be used as a client, which will be receiving sensor readings from MCU combination. Android system offers wide possibilities of Bluetooth Low Energy application and a modern way of developing apps.

2.4.3 Bluetooth Low Energy

Bluetooth Low Energy is a wireless network technology, which is applied in wide spectrum of industries, including healthcare, fitness, security and home entertainment. In comparison to Classic Bluetooth, Bluetooth Low Energy offers reduced power consumption, delivering similar communication range. All Bluetooth Low Energy devices use the Generic Attribute Profile, which terminology consists of the following terms: client, server, characteristic, service, descriptor and identifiers.

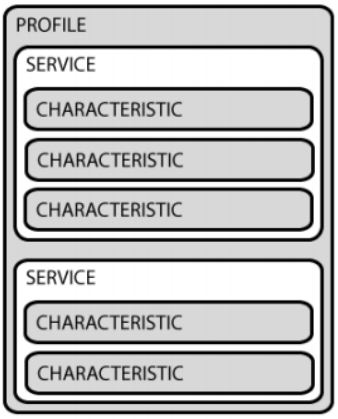


Figure 6: Profile, service and characteristic locations in system [9]

2.4.4 Application of Bluetooth Low Energy in thesis project

The "Texas Instruments SimpleLink™ CC2650 BoosterPack™ plug-in module" uses Bluetooth Low Energy in pursuit of best optimization and power consumption. In this thesis project, a role of the "Server" is given to the MCU combination and the smart device acts like a "Client" receiving sensor reading.

2.5 Overview of the system

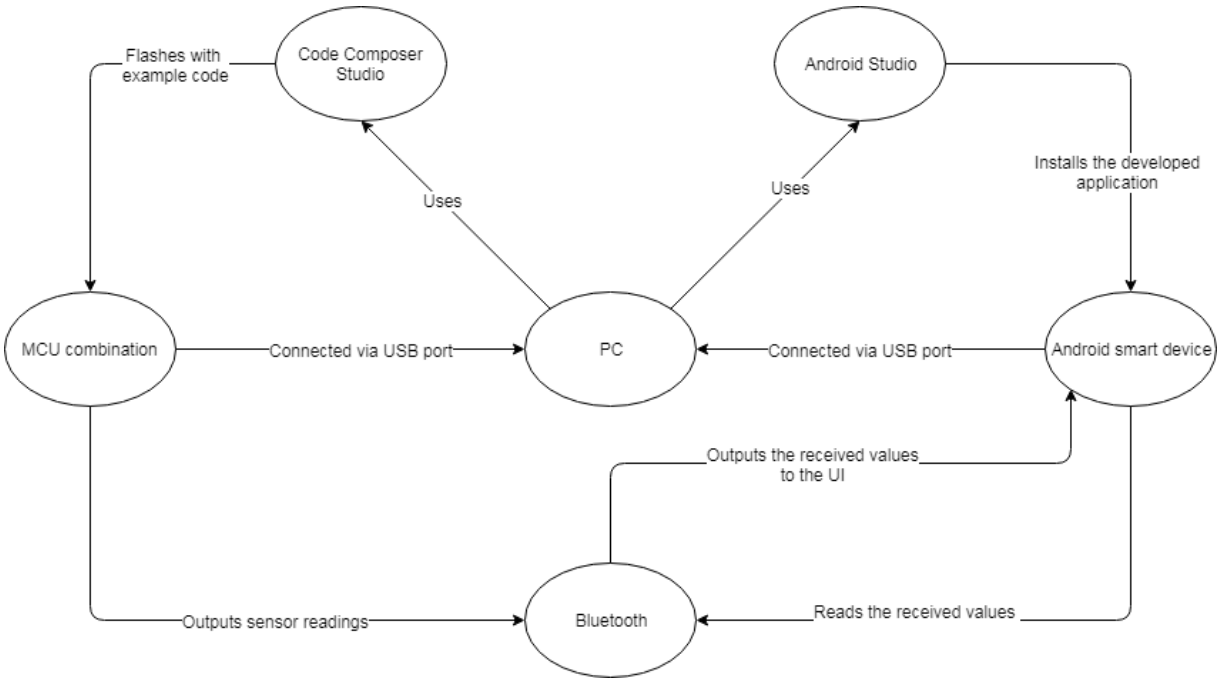


Figure 7: The thesis project's simplified system overview

3 Practical part

3.1 Setup of MCU hardware and environment

3.1.1 Choice of IDE

Code Composer™ Studio is an integrated development environment designed to support Texas Instruments microcontrollers and embedded processors. As of 09th March 2018 the latest version of Code Composer™ Studio is "v8", which was used in thesis project. In comparison to other IDE offered by IAR Systems – "IAR Embedded Workbench", Code Composer™ Studio offers a more comfortable and optimized working process, what does not require any work done to make factory examples perform. The CCS IDE makes it possible to run the "heavy" factory examples without any problems, while the IAR Embedded Workbench allows using only 32KB of code in the free version – this was the main reason for choosing CCS over IAR Embedded Workbench.

3.1.2 Setting up the Code Composer™ Studio IDE

The very first part of the thesis was environment setup. The Code Composer™ Studio IDE was chosen, because it delivers a wide spectrum of functionality in terms of developing software needed to operate the microcontroller and includes the functionality required for example project on-the-spot functionality. In order to achieve Bluetooth Low Energy communication and "Texas Instruments SimpleLink™ Sensors BoosterPack™" support, a compilation of add-ons and software development kits is required to be downloaded from Texas Instruments official sources for the IDE.

1. TI-RTOS for MSP43x series microcontrollers. TI-RTOS is a real time operating system developed by Texas Instruments for production microcontrollers. TI-RTOS for MSP43x series microcontrollers enables „Texas Instruments MSP-EXP432P401R SimpleLink™ Microcontroller LaunchPad™" capability and developing support with Code Composer™ Studio.

2. TI-RTOS for CC13xx and CC26xx series microcontrollers. This real time operating system enables Texas Instruments SimpleLink™ CC2650

BoosterPack™ plug-in module capability and developing support with Code Composer™ Studio.

3. Bluetooth Low Energy software stack, called "BLE-STACK". As of 28th March 2018 the version is "V2.2.2". BLE-STACK provides full-featured Bluetooth 4.2 and Bluetooth 5 certified stacks that include all necessary software to make example application used in the thesis to work.

4. SimpleLink™ MSP432P4 high-precision ADC MCU Software Development Kit. As of 16th March 2018 the version is "2.10.00.14". The SimpleLink™ MSP432P4 SDK includes a compilation of software to recognize other MCU's in the combination.

5. Bluetooth Plugin for SimpleLink™ MCU SDK. The SimpleLink™ SDK Bluetooth Plugin is an affiliate software package that enables the use of Bluetooth radio on „Texas Instruments MSP-EXP432P401R SimpleLink™ Microcontroller LaunchPad™". For the thesis project, this plugin is very important, because it contains the "Sensors BoosterPack™ Example" that will be used to output sensor data to Android smart device.

6. Sensors And Actuator Interface Library Plugin or SAIL. As of 02nd February 2018 the version is "v1.20.00.02". This plugin provides a set of application programming interfaces required for "Texas Instruments SimpleLink™ Sensors BoosterPack™" functionality of sensors.

3.1.3 Updating Texas Instruments SimpleLink™ CC2650 BoosterPack™ plug-in module with the latest SNP image

The SNP image for the CC2650 BoosterPack™ includes hardware configuration for this particular MCU and every BLE Plugin version may contain an updated SNP image. To make sure that example debug is successful, it is required to update the SNP image.

The SNP image on the initial CC2650 BoosterPack™ that was used for the thesis project, was outdated, what caused to move project completion to an impressive time frame. The problem was caused by the mistake that Texas Instruments's developers made in the documentation for CC2650 BoosterPack™ external programming topic.

The mistake was related with power supply of CC2650 BoosterPack™ during the process. The developers have not included the fact that CC2650 BoosterPack™ requires 3.3V and GND in order to be powered for reprogramming.

3.1.3.1 Hardware setup prerequisite for SNP image programming

Nevertheless, first of all, in order to update the CC2650 BoosterPack™ SNP image, it was required to install the "TI SmartRF™ Flash Programmer 2" software utility from Texas Instruments official source. Next, it was required to connect an external programmer to the MCU. As it was explained in the theoretical part of the thesis before, the "Texas Instruments MSP-EXP432P401R SimpleLink™ Microcontroller LaunchPad™" is equipped with XDS110 debugger and it simplifies the programming process, just by granting access to J102 "XDS110 OUT" connector. In order to provide power for CC2650 BoosterPack™, it was necessary to remove the jumpers from J101 Isolation Block, leaving the jumpers on "3.3V" and "GND" pins. Using the standard 10-pin ARM programmer cable included in the CC2650 BoosterPack™ packaging, it was required to connect MSP432 LaunchPad™ with the CC2650 BoosterPack™ and launch the software utility.

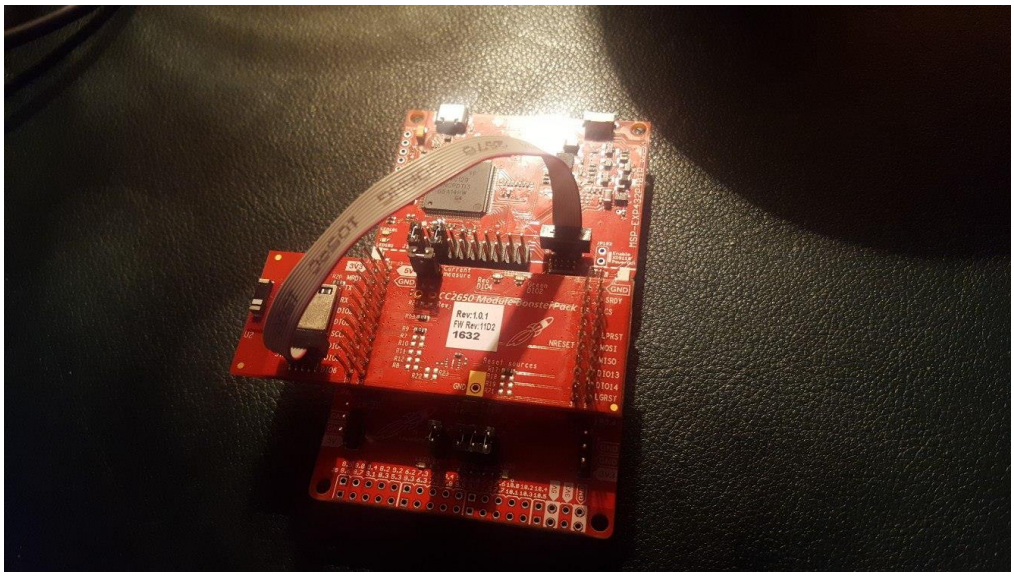


Figure 8: MCU combination setup for SNP image programming

3.1.3.2 Programming process

After connecting the MCU combination showed above to the computer with USB interface, the reprogramming process started from "TI SmartRF™ Flash Programmer 2" software utility. The version of the software utility used was "ver. 1.7.5". By successfully locating the required hex file, the CC2650 BoosterPack™ was programmed with updated SNP image.

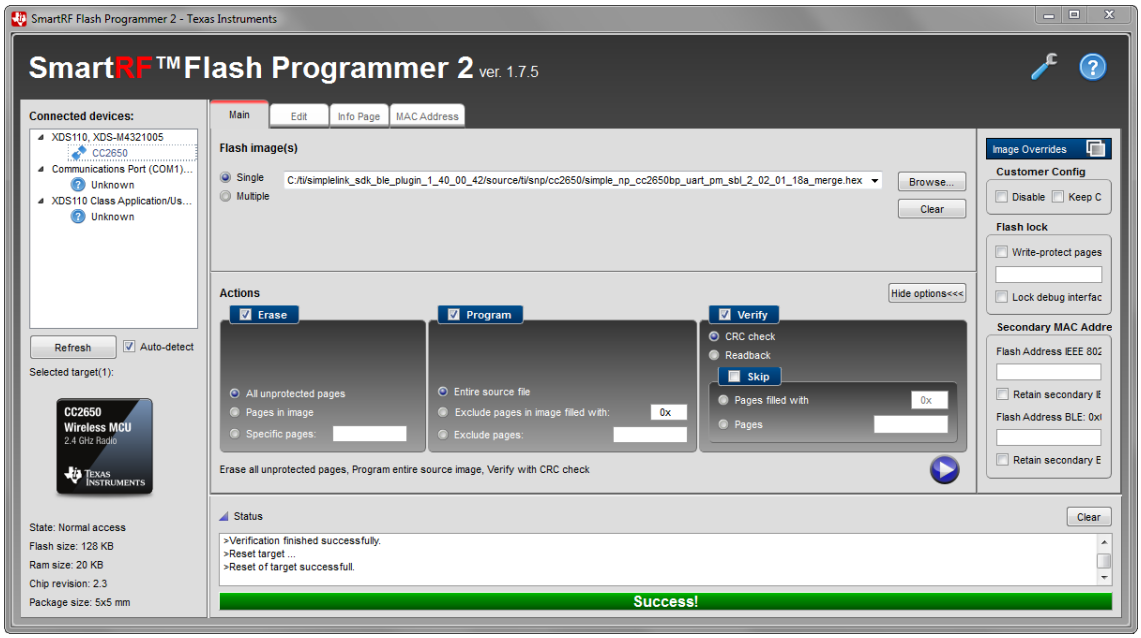


Figure 9: Snapshot from SmartRF™ Flash Programmer 2 including successful reprogramming of CC2650 BoosterPack™

3.1.3.3 Final hardware setup

Subsequently, the CC2650 BoosterPack™ SNP image was updated during programming process and the jumpers on J101 Isolation Block were returned back to default locations. For the example runtime it was required to run microcontrollers in following combination:

- Texas Instruments SimpleLink™ Sensors BoosterPack™
- Texas Instruments SimpleLink™ CC2650 BoosterPack™
- Texas Instruments MSP-EXP432P401R SimpleLink™ Microcontroller LaunchPad™

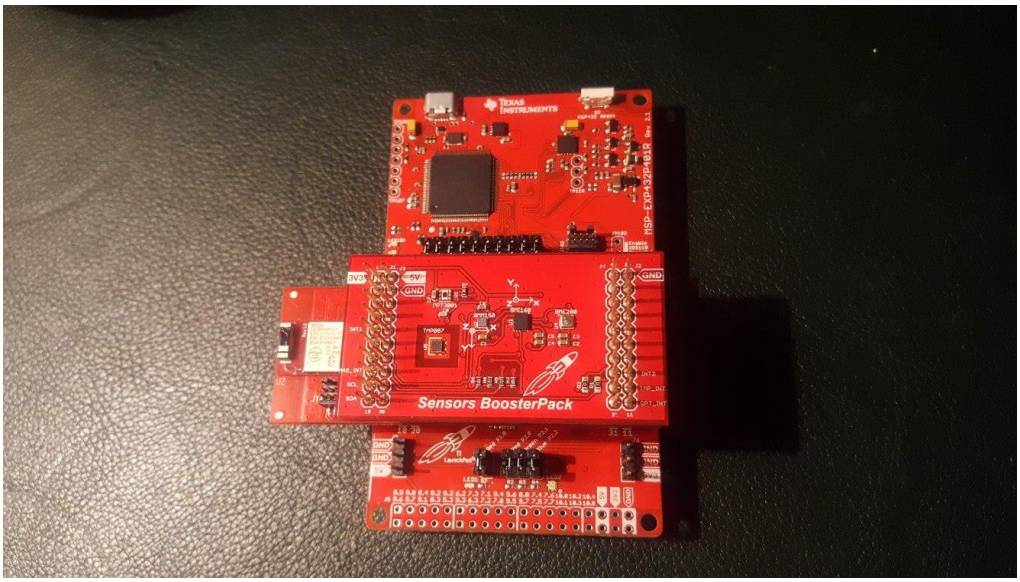


Figure 10: MCU combination

3.1.4 Running the example

The Sensors BoosterPack™ code example is a part of Bluetooth Low Energy plugin which enables users to try out the sensors in action. Each sensor has its unique BLE profile and provides the data from periodic sensor readings. The sensors used in the thesis project are the following:

- Texas Instruments „OPT3001" Ambient Light Sensor
- Texas Instruments "TMP007" Contactless Temperature Sensor
- Bosch "BME280" Integrated Environmental Unit

Every sensor in this example, as it was said, has its own profile and set of services with characteristics, corresponding to the sensors. This example relies on Sensor and Actuator Interface Library (SAIL), which was required to install during IDE preparation.

Table 2: Sensor service output specification table

Purpose	UUID	Format	Unit	Properties
IR Temperature Data	F000AA01-0451-4000-B000-000000000000	IEEE-754 32-bit floating point	°C	Notify
IR Temperature Config (enable)	F000AA02-0451-4000-B000-000000000000	Integer	N/A	Read/Write
Humidity Data	F000AA21-0451-4000-B000-000000000000	IEEE-754 32-bit floating point	Percent	Notify
Humidity Config (enable)	F000AA22-0451-4000-B000-000000000000	Integer	N/A	Read/Write
Barometer Data	F000AA41-0451-4000-B000-000000000000	Integer	Pascals	Notify
Barometer Config (enable)	F000AA42-0451-4000-B000-000000000000	Integer	N/A	Read/Write
Optic Data	F000AA71-0451-4000-B000-000000000000	IEEE-754 32-bit floating point	lux	Notify
Optic Config (enable)	F000AA72-0451-4000-B000-000000000000	Integer	N/A	Read/Write

To run the example, it was required to import it to the IDE after installation of all software add-ons and plugins, and run the debug session. In order to track the MCU activity, the IDE offers a serial terminal what makes the working process impressively easier. After all the actions were performed, the microcontroller part was finished and project moved to Android application development.

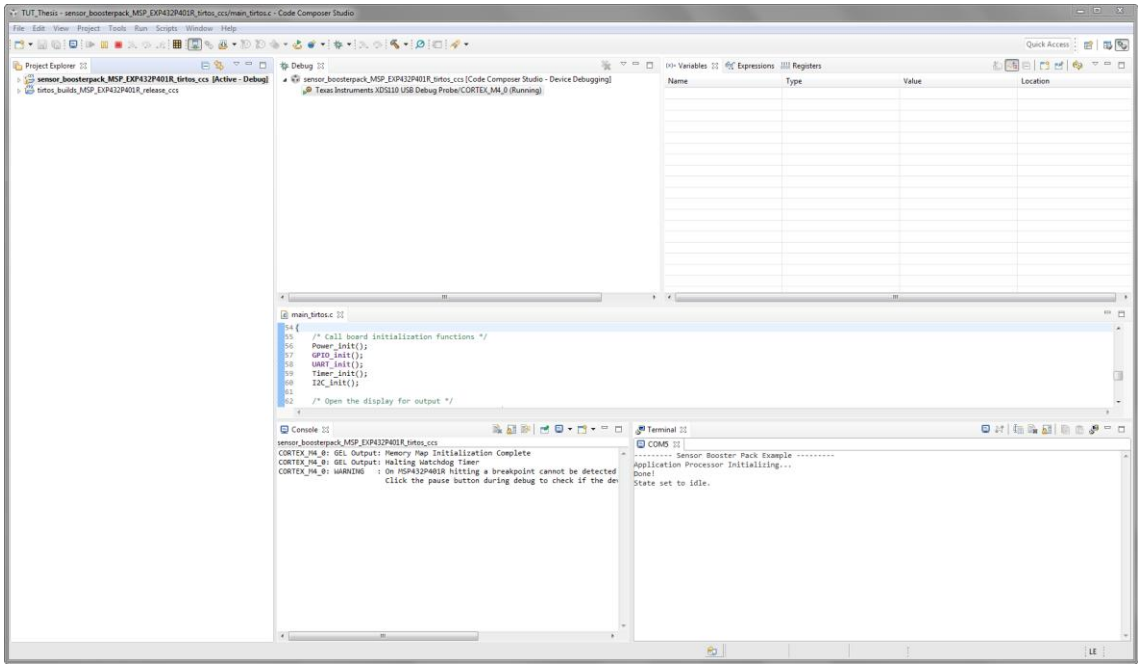


Figure 11: Snapshot from Code Composer™ Studio IDE with active debug session of Sensor BoosterPack™ example code

3.2 Android application development

Android application development was the main task of this thesis.

In order to understand the code in the created project, to run the application and MCU combination, it is required to have basic knowledge of language "C" and "Java". It is advised to read the book "Teach Yourself C in 24 hours" from author Tony Zhang and complete all tasks in the book, also to reference to "Google Developers" Android section, if the student has zero knowledge about programming.

The total working time devoted to this application development is approximately 600 hours in four months, what includes the learning time of the basics of language "C" and "Java" from knowing nothing about the world of programming, development of the application and working through debugging of the application to make it functional.

3.2.1 Choice of IDE

As the thesis project's idea was to develop an Android application, which uses BLE concept, the choice of IDE was known from the beginning of the working process. The choice of IDE fell on Android Studio.

Android Studio is an official Android development IDE created by Google LLC and announced on 16th May 2013. In comparison to other IDE's, the Android Studio offers very comfortable layout work and will be supported as long as Android applications are still being developed. The importance of this IDE for thesis project consists in the fact that it has fast functionality, friendly UI, comfortable layout, live logging possibility, fast debugging process and, most importantly, very big support and guides from active Google LLC. developers.

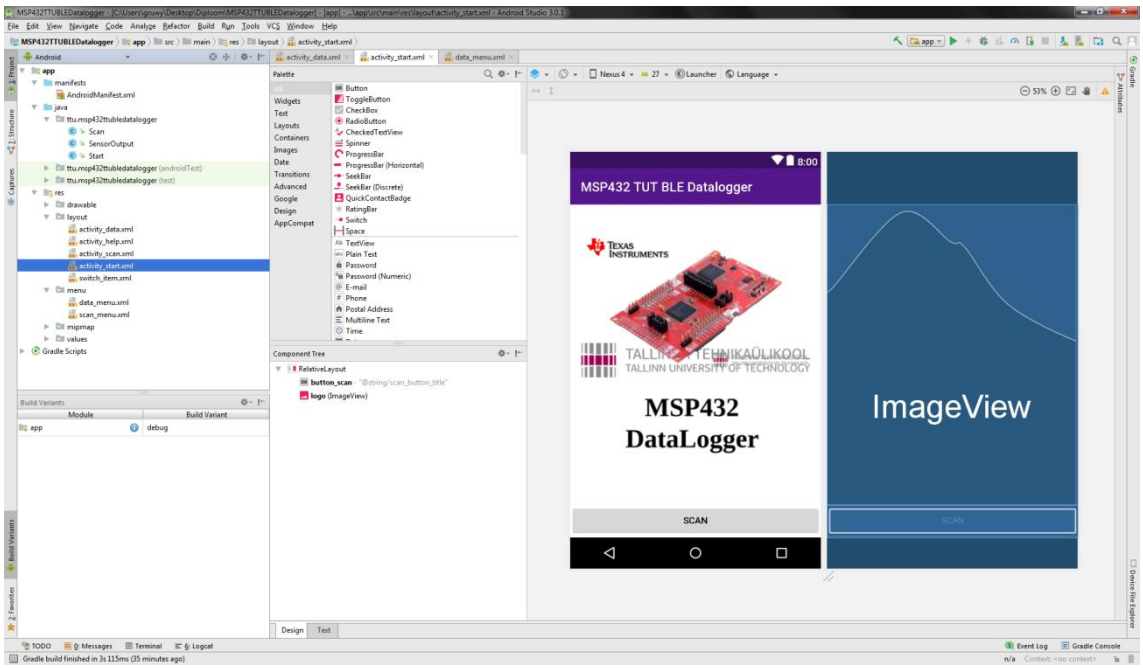


Figure 12: Android Studio snapshot

Android Studio turned out to be a very good tool for thesis project development and simplified a lot of questions by an implemented tip system, which solved most of the programming questions inside the IDE. The Android Studio IDE version used in thesis project was "3.1.2".

3.2.2 Testing smart device setup for development

The smart device used for testing purposes was Samsung Galaxy S6 Limited Edition.

Table 3: Technical information of testing smart device

Smart device name	Samsung Galaxy S6 Limited Edition
Model number	SM-G920F
Android version	7.0
Baseband version	G920FXXU6ERC2
Build number	NRD90M.G920FXXU6ERC1
Kernel version	3.10.61-13115714

Every IDE requires the "Developer options" to be enabled on the smart device in order to work with it successfully and enable debugging through this smart device. Developer options are enabled by tapping several times on build number in settings menu.

3.2.3 Planning of the application structure and creation of the project in Android Studio IDE

The overall structure of the application is shown in the following diagram:

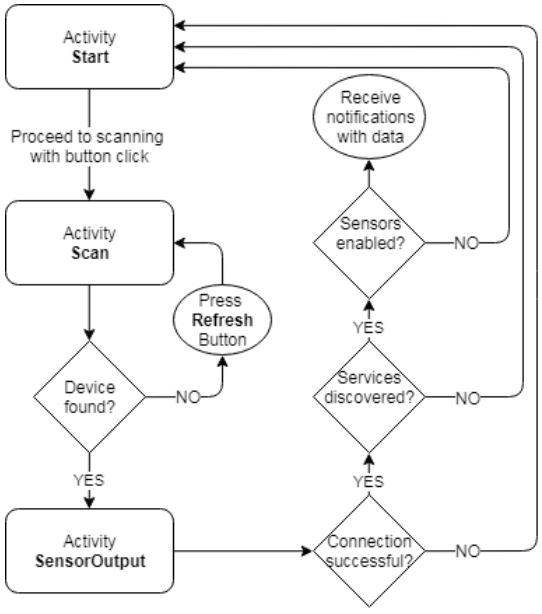


Figure 13: Structure of the Android application

The project is created by simply starting a new project in Android Studio IDE. After choosing the name for the project, it was required to choose the right API level for the application. The decision fell on API level 23, as it has more advanced BLE functions and has minor fixes, which made developing easier for thesis project completion.

3.2.4 First additions and "Start" activity

The first addition to the project was a combination of assigned permissions at AndroidManifest.xml file:

- BLUETOOTH permission is granted in order to use Bluetooth features in our thesis project
- BLUETOOTH_ADMIN permission is granted in order to initiate other device discovery.
- ACCESS_COARSE_LOCATION permission is granted, because API 21+ requires location coarse enabled to make other device discovery function correctly
- ACCESS_FINE_LOCATION is a companion permission for ACCESS_COARSE_LOCATION and is granted in order to illuminate possible malfunctions during application working process

```
<uses-permission
android:name="android.permission.BLUETOOTH" />
```

Figure 14: Example of the Bluetooth permission granted to the application

This compilation of permissions is essential for thesis project, because the fulfilled functionality of the application would not be realized.

The first, called "Start", activity is a very simple, introductory activity that includes an illustration of MSP432 LaunchPad™ and a button with a name "Scan" that moves the user to next activity.

The tasks of the activity are the following:

- Initialize the view and the “Scan” button
- Ask the user to enable Bluetooth, if not enabled
- Ask the user to grant access to coarse location

As the API level is “23”, this means that the minimum Android OS version is “6.0” and it is essential to grant coarse location permission for the scanning and connecting to work. The development of this activity was the first steps in learning Android developing and delivered general understanding on how the Java programming language works.

Figure 15: Snapshot of "Start" activity of the Android application

3.2.5 “Scan” activity

The “Scan” button click leads the user to activity, where the MCU combination searching is being executed. Even though the Bluetooth inquiry happened in the start of the application, the scanning activity asks the user to enable Bluetooth once again, if it is not enabled. The concept of this activity intends an immediate start of MCU combination discovery, as the activity initializes and outputs the results in predefined list view in the UI. The handling of discovered Bluetooth activity is realized through broadcast receiver. On list view click, the application parses the device address to intent and leads the user to next activity, where the main functionality is executed.

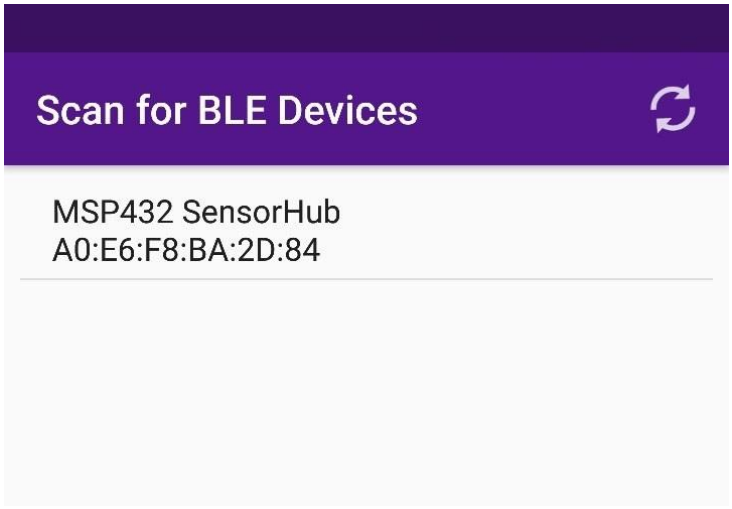


Figure 16: Successful finding of MSP432 SensorHub

3.2.6 “SensorOutput” activity

After successfully scanning for the MCU combination, receiving it’s name and address in list view and clicking on the item from list view, application brings the user to the final activity, where all the sensor outputs are received.

3.2.6.1 Connecting to the MCU combination

First of all, the application insures that the Bluetooth is enabled on the testing smart device and attempts to connect to the MCU combination with the help of intent that transferred MCU address to this activity. The connection to the testing smart device is executed with help of “connect” method.

```
public void connect(BluetoothDevice device) {
    if (mBluetoothGatt == null) {
        mBluetoothGatt =
device.connectGatt(SensorOutput.this, false,
mGattCallback);
    }
}
```

Figure 17: Connect method for sensor output activity

While the testing smart device attempts to connect to the MCU combination, a progress dialog is shown:

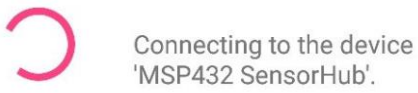


Figure 18: Connection progress dialog

If the connection succeeds, the "Sensor BoosterPack™ example code" reacts to the connection and logs the peer connection to the terminal in CCS IDE reporting the peer address.

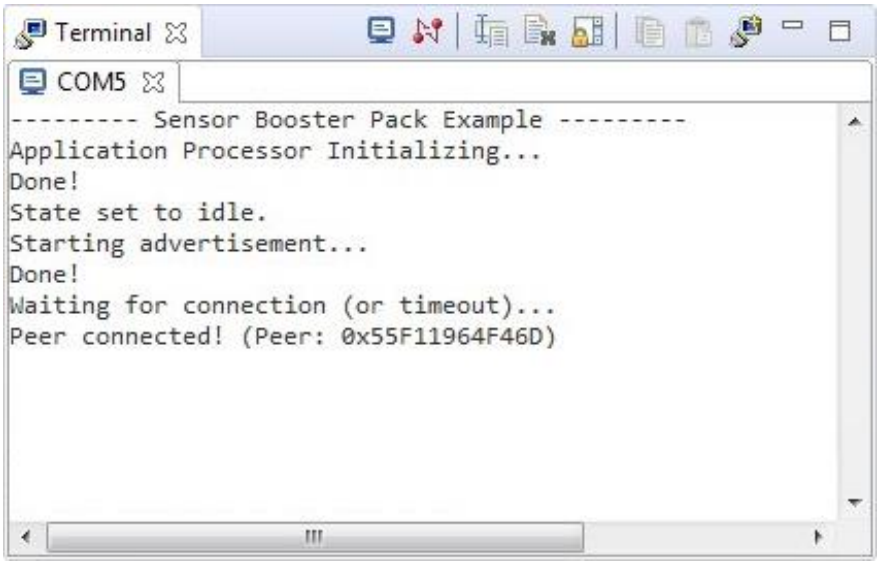


Figure 19: Sensor BoosterPack™ example code output in CCS terminal

3.2.6.2 Discovering MCU combination services

After connecting to the MCU combination, the testing smart device attempts to discover services, showing the following progress dialog:

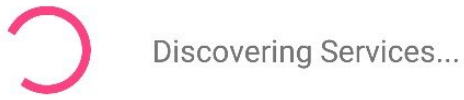


Figure 20: Service discovery dialog

The service discovery is being triggered by the "onConnectionStateChange" method of GATT callback that performs an action after a certain stage of connection is achieved by the Bluetooth GATT. In this method's case, if the Bluetooth GATT succeeded and the state changed to "STATE_CONNECTED" on the Bluetooth profile, then the application starts service discovery.

```
@Override
public void onConnectionStateChange(BluetoothGatt gatt, int
status, int newState) {
  Log.d(TAG, "Connection State Change: " + status + " -> "
+ connectionState(newState));
    if (status == BluetoothGatt.GATT_SUCCESS && newState ==
BluetoothProfile.STATE_CONNECTED) {
  gatt.discoverServices();
  mHandler.sendMessage(Message.obtain(null, MSG_PROGRESS,
"Discovering Services..."));  }
```

Figure 21: onConnectionStateChange method realization if the connection is successful

Thanks to the updated methods and functions in API 23, service discovery is realized in a single command.

3.2.6.3 Enabling sensors

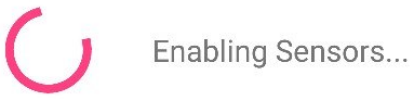


Figure 22: Sensor enabling dialog

In order to read and write the values to the characteristics, a client requires a "Client Characteristic Configuration Descriptor". The descriptor used for the needs of this project is defined with a value "0x2902", which grants the client a possibility to enable and disable server's notifications.

The next step for the application is to enable sensor broadcasting by writing a byte value "0x01" to each of sensors' configuration characteristic named "*_CONFIG" (where * is sensors' name), also as to read each sensor and subscribe to each of the sensors. This job is realized by "enableNextSensor", "readNextSensor" and "setNotifyNextSensor" methods which use the state machine to go through every case and enable every sensor to send data to the client smart device.

The process of whole sensor enabling process is described in the following figure:

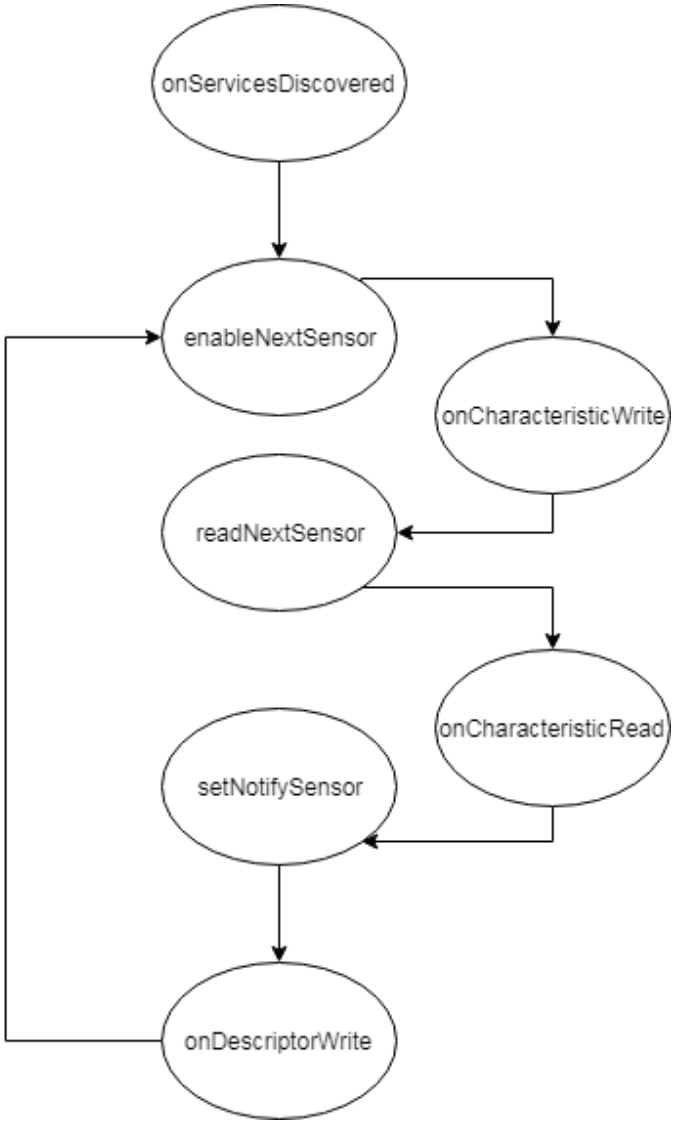


Figure 23: Sensor enabling process

Table 4: Case distribution

Case number	Sensor
0	Temperature
1	Humidity
2	Barometer
3	Optic

After the services were discovered, the application uses a "onServicesDiscovered" GATT callback, logs the service discovery status to console and initializes the progress dialog with the "Enabling Sensors…" context, transferring action to "enableNextSensor(gatt)" method.

```
@Override
public void onServicesDiscovered(BluetoothGatt gatt, int
status) {
        Log.d(TAG, "Service discovery status: " + status);
        mHandler.sendMessage(Message.obtain(null,
MSG_PROGRESS, "Enabling Sensors..."));
        reset();
        enableNextSensor(gatt);
}
```

Figure 24: onServicesDiscovered callback if the discovery is successful

Before attempting to set the value of chosen sensor, the system logs the beginning of the writing process, then sets the byte value of "0x01" to the characteristic. After setting the value of characteristic, the "gatt.writeCharacteristic(characteristic)" is called to make GATT write of the characteristic. To move to the next method, the GATT callback "onCharacteristicWrite" is called to execute the "readNextSensor" method.

```
@Override
      public void onCharacteristicWrite(BluetoothGatt gatt,
BluetoothGattCharacteristic characteristic, int status) {
          readNextSensor(gatt);
      }
```

Figure 25: onCharacteristicWrite GATT callback

"readNextSensor" method reads the initial value of the "*_CONFIG" characteristic and this is when the "onCharacteristicRead" GATT callback triggers.

```
@Override
public void onCharacteristicRead(BluetoothGatt gatt,
BluetoothGattCharacteristic characteristic, int status) {
    if (TEMP_DATA.equals(characteristic.getUuid())) {
        mHandler.sendMessage(Message.obtain(null, MSG_TEMP,
characteristic));
    }
    setNotifyNextSensor(gatt);
}
```

Figure 26: onCharacteristicRead GATT callback if the TEMP_DATA characteristic equals the gotten characteristic UUID

Finally, after the reading the written value of configuration characteristics, the application moves to subscribing to the sensor's data stream. This task is executed by "setNotifyNextSensor" method. This is where CONFIG_DESCRIPTOR is used.

```
private void setNotifyNextSensor(BluetoothGatt gatt) {
    BluetoothGattCharacteristic characteristic;
    switch (mState) {
        case 0:
            Log.d(TAG, "Set notify temperature");
            characteristic = gatt.getService(TEMP_SERVICE)
                    .getCharacteristic(TEMP_DATA); }
    gatt.setCharacteristicNotification(characteristic,
true);
    BluetoothGattDescriptor desc =
characteristic.getDescriptor(CONFIG_DESCRIPTOR);
desc.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_V
ALUE);
         gatt.writeDescriptor(desc);
}
```

Figure 27: setNotifyNextSensor method example on temperature notification enabling

The GATT callback for writing a value is "onDescriptorWrite", where the application advances to the next case and begins from "enableNextSensor" method.

```
@Override
public void onDescriptorWrite(BluetoothGatt gatt,
BluetoothGattDescriptor descriptor, int status) {
    advance();
    enableNextSensor(gatt);
}
```

Figure 28: onDescriptorWrite GATT callback

After enabling and subscribing to all the sensors, the progress dialog is being dismissed and application displays the values in predefined layout. The value updating in the UI happens through "onCharacteristicChanged" GATT callback, where every UUID is being checked for equality to current chosen characteristic and sent to a handler, where the "update*Value(characteristic)", where "*" is sensor type, is being executed.

```
@SuppressLint("HandlerLeak")
private Handler mHandler = new Handler() {
    @Override
    public void handleMessage(Message msg) {
        BluetoothGattCharacteristic characteristic;
        switch (msg.what) {
            case MSG_TEMP:
                characteristic =
(BluetoothGattCharacteristic) msg.obj;
                if (characteristic.getValue() == null) {
                    Log.w(TAG, "Error obtaining temperature
value.");

Toast.makeText(getApplicationContext(),"Error obtaining
temperature value. Returning to main
screen.",Toast.LENGTH_SHORT).show();
                    finish();
                    return;
                }
                updateTemperatureValue(characteristic);
                Log.i(TAG, "Characteristic " +
characteristic.getUuid() + " changed value!");
                break;
     }
};
```

Figure 29: Handler realization on temperature sensor case

The UI is being refreshed every second with new sensor readings, which pass in their respectable formats. The format handling and representation is being executed in "update*Value" method.

```
@Override
public void onCharacteristicChanged(BluetoothGatt gatt,
BluetoothGattCharacteristic characteristic) {

    if (TEMP_DATA.equals(characteristic.getUuid())) {
        mHandler.sendMessage(Message.obtain(null,
MSG_TEMP, characteristic));
    }
```

Figure 30: onCharacteristicChanged GATT callback on example of temperature sensor handling

```
@SuppressLint("DefaultLocale")
private void
updateTemperatureValue(BluetoothGattCharacteristic
characteristic) {
    final byte[] data = characteristic.getValue();
    float f1 =
ByteBuffer.wrap(data).order(ByteOrder.LITTLE_ENDIAN).getFlo
at();
    mTemperature.setText(String.format("%.1f\u00B0C", f1));
}
```

Figure 31: updateTemperatureValue method

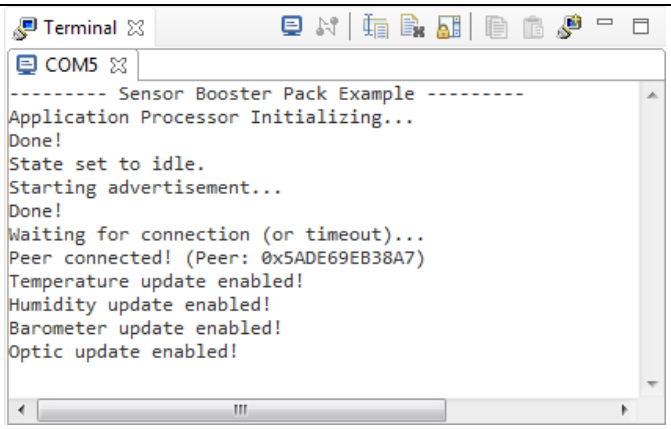


Figure 32: CCS terminal output after successful connection and sensor enabling

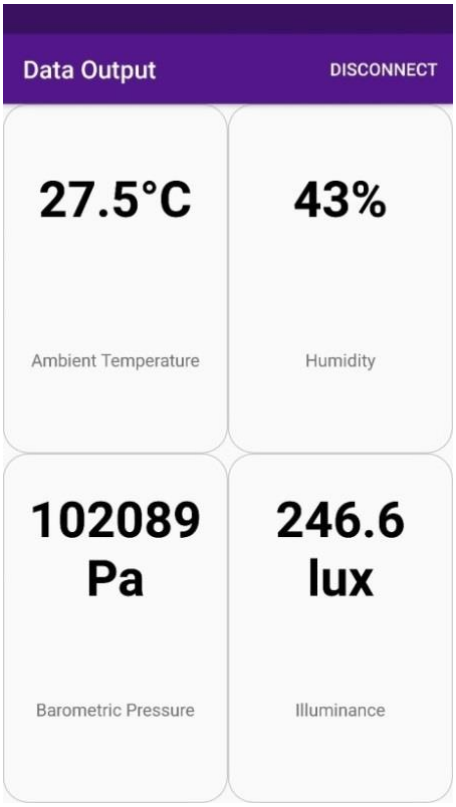


Figure 33: Data Output UI

4 Summary

The MSP432 based Bluetooth Low Energy solution got a fully working example of communication with Android smart device. The author of this thesis has significantly gained knowledge in Android development, CCS IDE application, Java and C language and showed an example of what is possible to achieve with “zero” knowledge about the world of programming.

The main task of the created system was to output sensor readings from MCU combination and provide a solution that would display the readings in “user-friendly” interface. The task completion was successfully achieved.

This thesis’s project is useful for further development, education purposes, application in individual projects.

The further development possibilities could be the following:

- Create a solution to output readings from accelerometer and gyroscope
- Create a solution to output signal strength between MCU combination and Android device in “Sensor Output” activity
- Create a solution to graph the received sensor outputs and save them for further analysis on external memory
- Create a solution to support sensors not only from plug-in module, used in the thesis project, but for sensors from other manufacturers
- Create a solution to review full device information from a menu item

It is possible to integrate a lot more functionality to this project and it may be a good start for developing an universal MSP432 data logging Android smart device application.

5 References

[1] "MSP432P401R SimpleLink™ Microcontroller LaunchPad™ Development Kit User's Guide (Rev. F)," March 2018. [Online]. Available: http://www.ti.com/lit/ug/slau597f/slau597f.pdf. [Accessed 05 May 2018].

[2] "Access Control Panel With Bluetooth® low energy, Capacitive Touch, and Software Integration Reference Design," Texas Instruments, February 2017. [Online]. Available: http://www.ti.com/lit/ug/tiducp8/tiducp8.pdf. [Accessed 05 May 2018].

[3] N. Siegel, "Using TI Certified Bluetooth® low energy Module (CC2650MODA) as Single-Chip Wireless MCU," Texas Instruments, January 2017. [Online]. Available: http://www.ti.com/lit/an/swra534a/swra534a.pdf. [Accessed 05 May 2018].

[4] "BOOSTXL-SENSORS Sensors BoosterPack™ Plug-in," March 2017. [Online]. Available: http://www.ti.com/lit/ug/slau666a/slau666a.pdf. [Accessed 05 May 2018].

[5] "OPT3001 Ambient Light Sensor (ALS) datasheet (Rev. C)," Texas Instruments, November 2017. [Online]. Available: http://www.ti.com/lit/ds/symlink/opt3001.pdf. [Accessed 05 May 2018].

[6] Texas Instruments, "TMP007 Infrared Thermopile Sensor with Integrated Math Engine," 2015.

[7] Bosch, "BME280 Product Flyer," [Online]. Available: https://ae-bst.resource.bosch.com/media/_tech/media/product_flyer/BME280_Productflyer_BST_20170109.pdf. [Accessed 06 May 2018].

[8] Bosch, "BME280," 2018. [Online]. Available: http://www.bosch-sensortec.com/en/bst/products/all_products/bme280. [Accessed 06 May 2018].

[9] K. Townsend, "Introduction to Bluetooth Low Energy," 2014.

Appendix 1 – MSP432 Datalogger User's Guide

In order to understand the microcontroller combination example code running process and the Android application project structure, it is required to have basic knowledge of programming languages "C" and "Java". Suggested book is "Teach Yourself C in 24 Hours" by Tony Zhang.

1 Hardware preparation

1.1 Installation and configuration of Code Composer™ Studio IDE

Step 1: To download Code Composer™ Studio IDE visit this link (http://processors.wiki.ti.com/index.php/Download_CCS), refer to "Download the latest CCS" section and download latest version of the IDE. It is advisable to install all the IDE files at the destination predefined by manufacturer (C:\ti).

Step 2: After downloading and installing Code Composer™ Studio, it is required to download and install a compilation of add-ons and software development kits:

1. TI RTOS for MSP43x.
   http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/tirtos/index.html

2. TI RTOS for CC26xx
   http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/tirtos/index.html

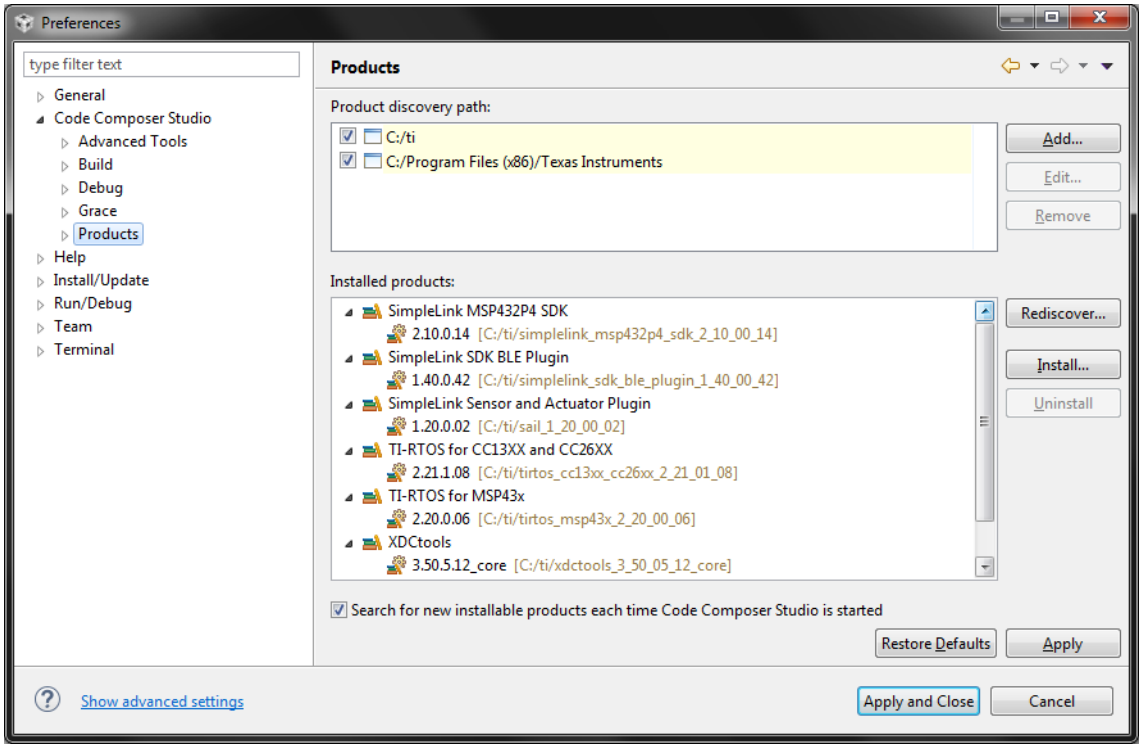
3. BLE STACK (Support for CC2640/CC2650)
   http://www.ti.com/tool/ble-stack

4. SimpleLink MSP432P4 High-precision ADC MCU Software Development Kit
   http://www.ti.com/tool/download/SIMPLELINK-MSP432-SDK

5. Bluetooth Plugin for SimpleLink™ MCU SDK
   http://www.ti.com/tool/download/SIMPLELINK-SDK-BLUETOOTH-PLUGIN

6. Sensor and Actuator Plugin for SimpleLink™ MCU SDKs
   http://www.ti.com/tool/download/SIMPLELINK-SDK-SENSOR-ACTUATOR-PLUGIN

In order to check, if the all software is installed in IDE, user has to navigate to "Window → Preferences" and have the preferences window opened respectively:



User's Guide Figure 1: CCS IDE preferences window

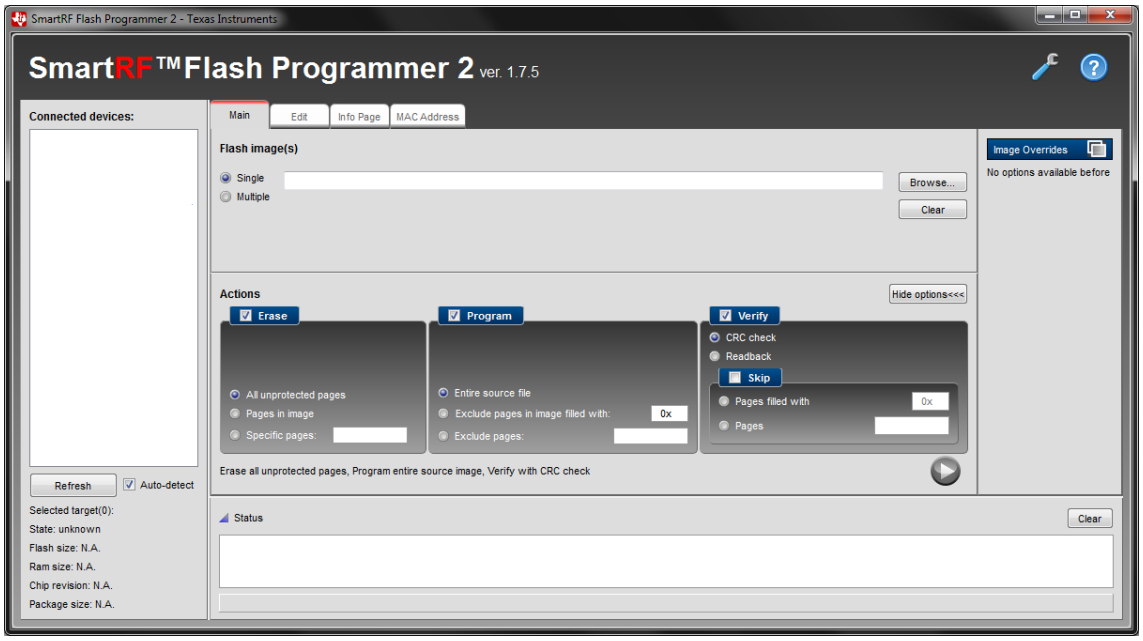
If all of the products are installed, it is possible to move to the next step.

1.2 Updating SNP image of the Texas Instruments CC2650 BoosterPack™

Step 1: To start with updating SNP image of CC2650 BoosterPack™ it is required to install the following software

- TI SmartRF™ Flash Programmer 2 (v2)
  http://www.ti.com/tool/FLASH-PROGRAMMER

After installing the software, user has to open the flash programmer.



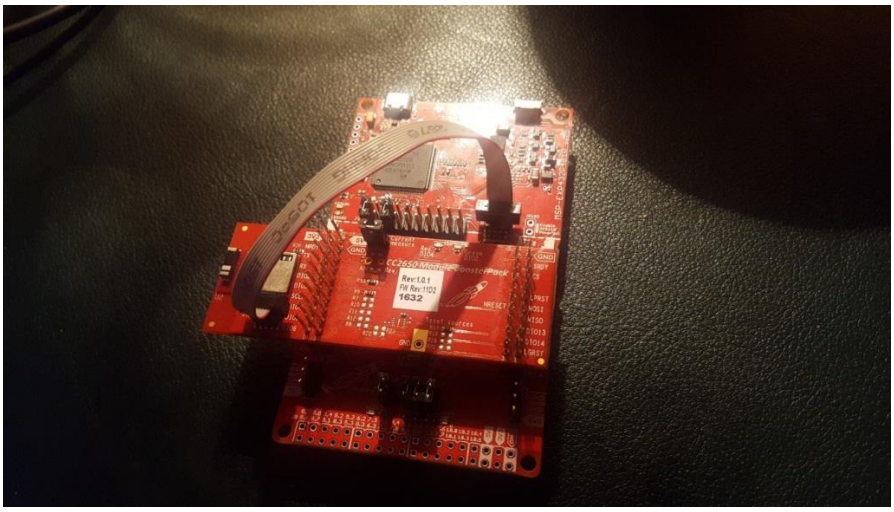
User's Guide Figure 2: Flash Programmer 2 snapshot

Step 2: Connect the microcontroller stack with USB and reprogram the SNP image.

For the example runtime it is required to run microcontrollers in following combination:

- Texas Instruments SimpleLink™ Sensors BoosterPack™
- Texas Instruments SimpleLink™ CC2650 BoosterPack™
- Texas Instruments MSP-EXP432P401R SimpleLink™ Microcontroller LaunchPad™

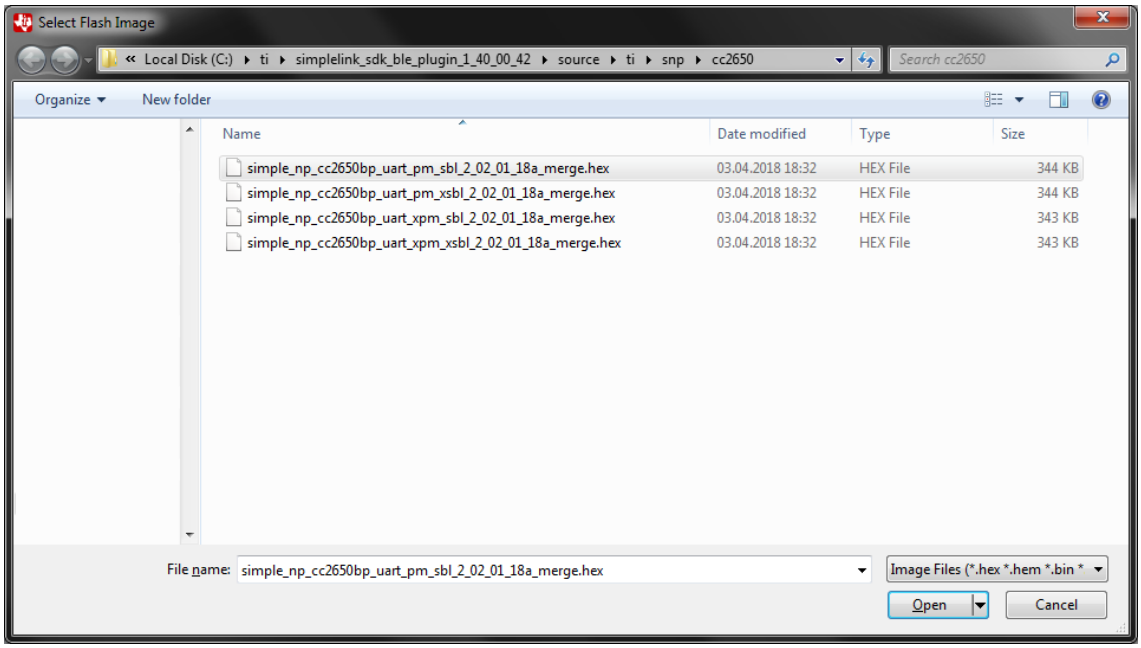
Using the standard 10-pin ARM programmer cable included in the CC2650 BoosterPack™ packaging, it is required to connect MSP432 LaunchPad™ with the CC2650 BoosterPack™ and launch the software utility through JTAG connectors.



User's Guide Figure 3: MCU combination setup for reprogramming

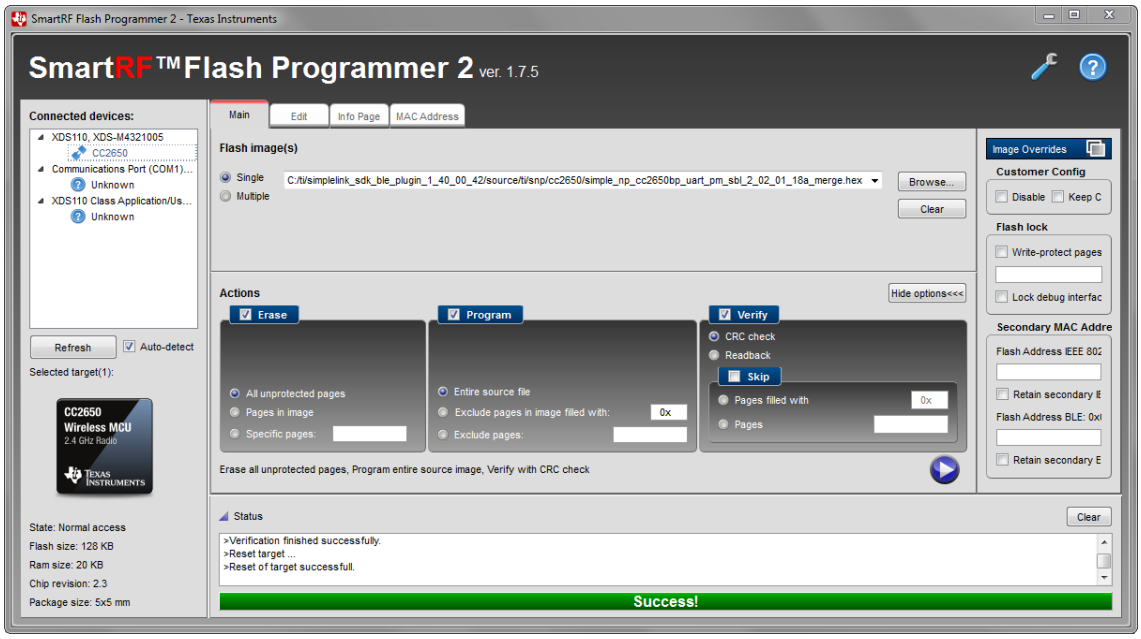
In order to provide power for CC2650 BoosterPack™, it is necessary to remove the jumpers from J101 Isolation Block, leaving the jumpers on “3.3V” and “GND” pins.

After doing this work, user needs to find the HEX image for reprogramming. The latest HEX image is located in the BLE plugin folder (C:\ti\simplelink_sdk_ble_plugin_1_40_00_42\source\ti\snp\cc2650).



User's Guide Figure 4: SNP image location

After locating the HEX image, user has to just push the “Play” button and software does all the work itself. Note: check the settings “Erase”, “Program” and “Verify” before programming to suit the settings set on the figure.



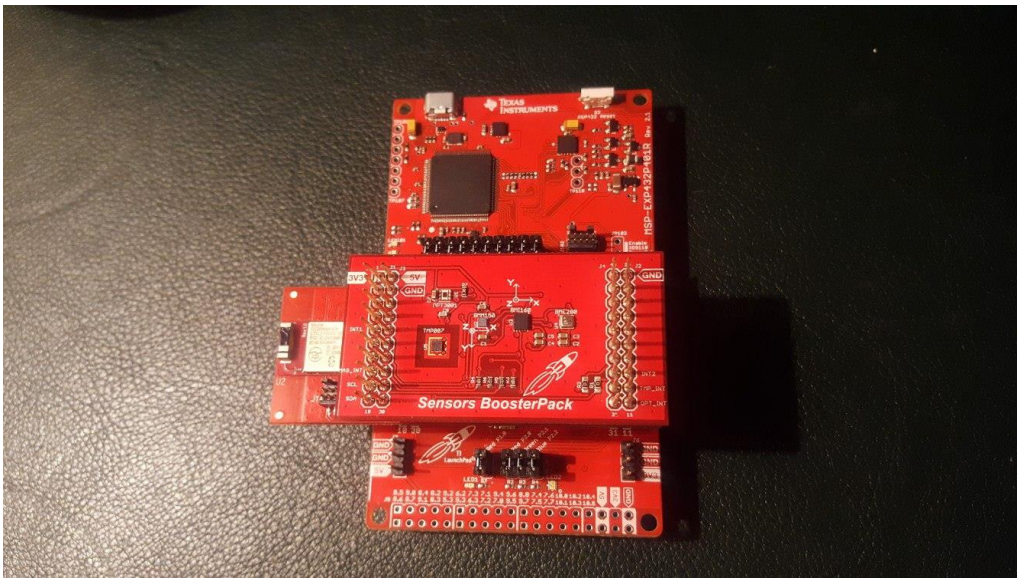
User's Guide Figure 5: Successful reprogramming in Flash Programmer 2

Place the jumpers back at default location after reprogramming.

1.3 Running the example code

Step 1: In order to run the example code it is required to assemble a MCU combination:

- Texas Instruments SimpleLink™ Sensors BoosterPack™
- Texas Instruments SimpleLink™ CC2650 BoosterPack™
- Texas Instruments MSP-EXP432P401R SimpleLink™ Microcontroller LaunchPad™



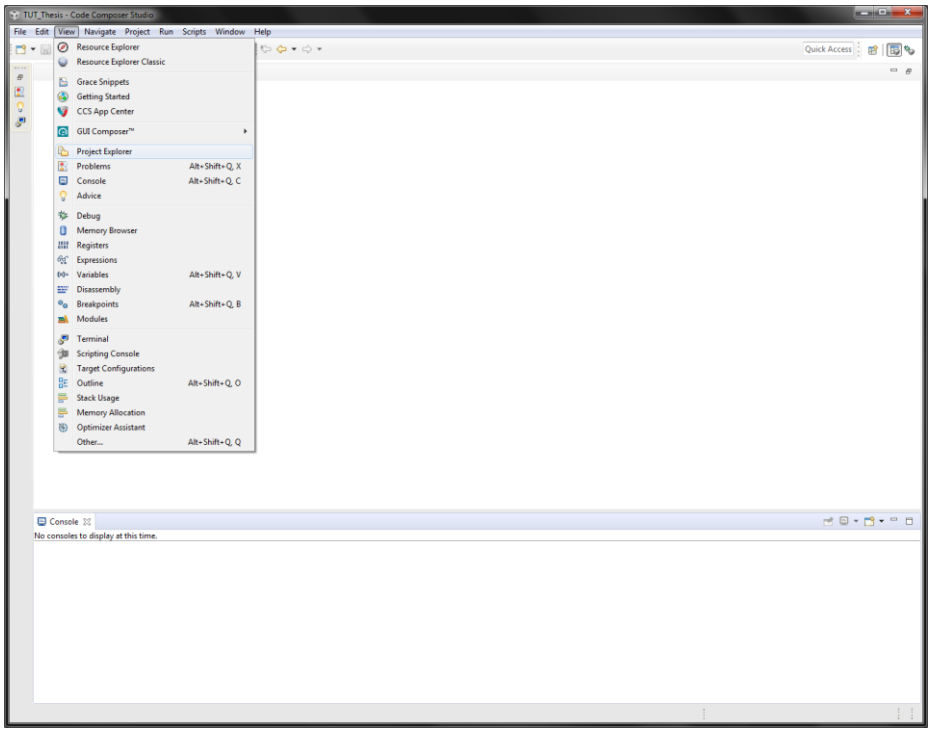
User's Guide Figure 6: Final MCU combination setup

As this step is completed, the user can move to the next step.

Step 2: Importing and running the example code in the IDE

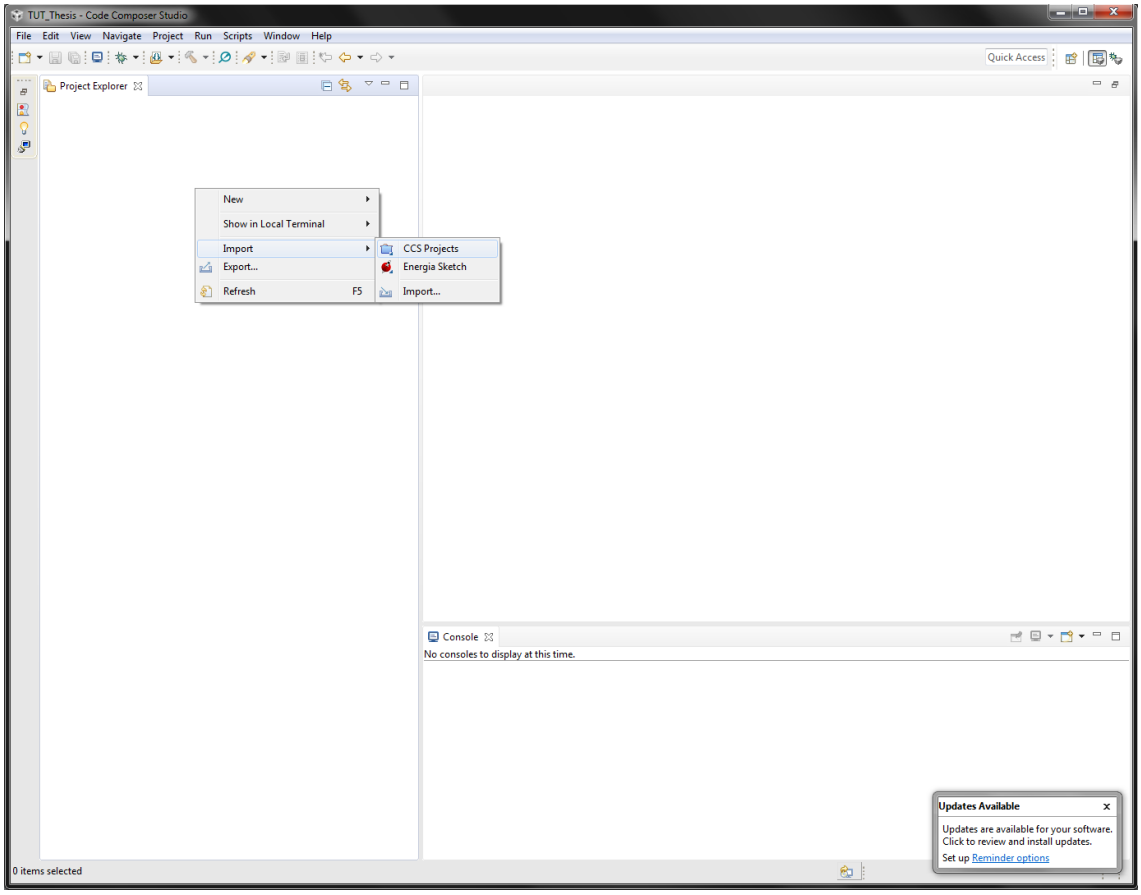
The steps are shown in the figures below.

Step 2.1: Go to View → Project Explorer



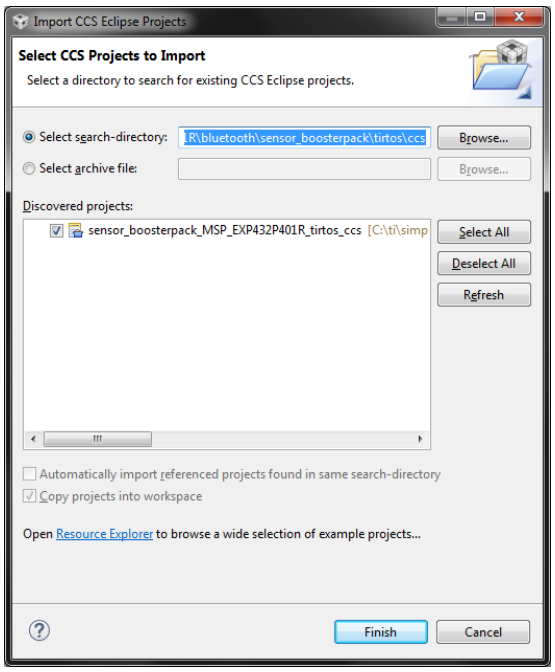
User's Guide Figure 7: Project Explorer location in "View" field

Step 2.2: Right click the Project Explorer window and do the following Import → CCS projects



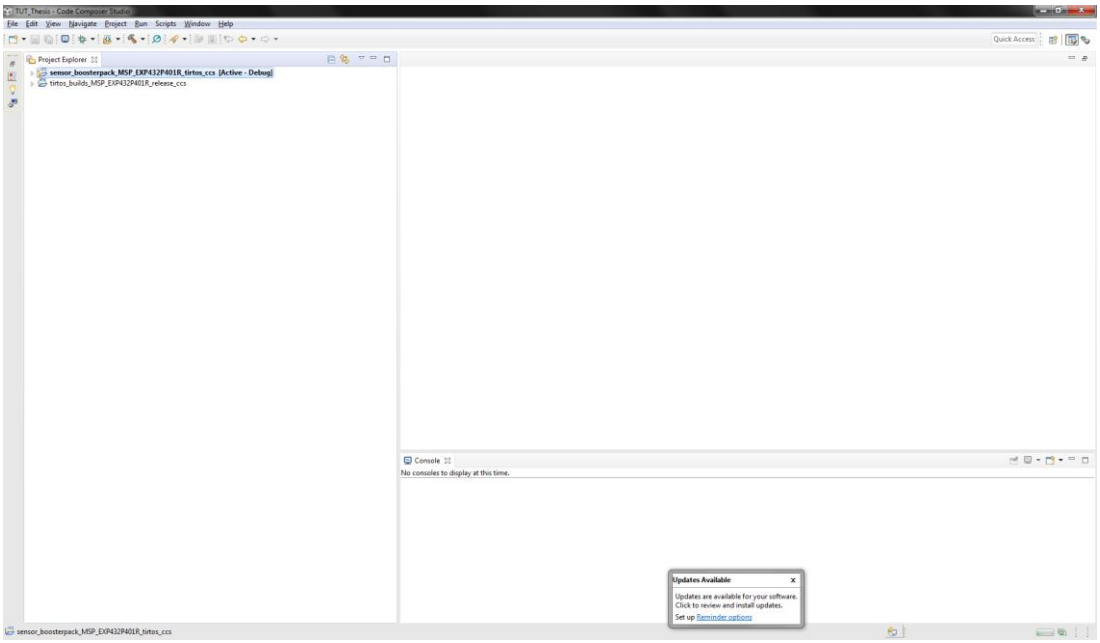
User's Guide Figure 8: CCS project import location

Step 2.3: As the user enters the import window, it is required to click the "Browse…" button and locate the project at the plugin location (C:\ti\simplelink_sdk_ble_plugin_1_40_00_42\examples\rtos\MSP_EXP432P401R\bluetooth\sensor_boosterpack\tirtos\ccs)



User's Guide Figure 9: CCS project import window

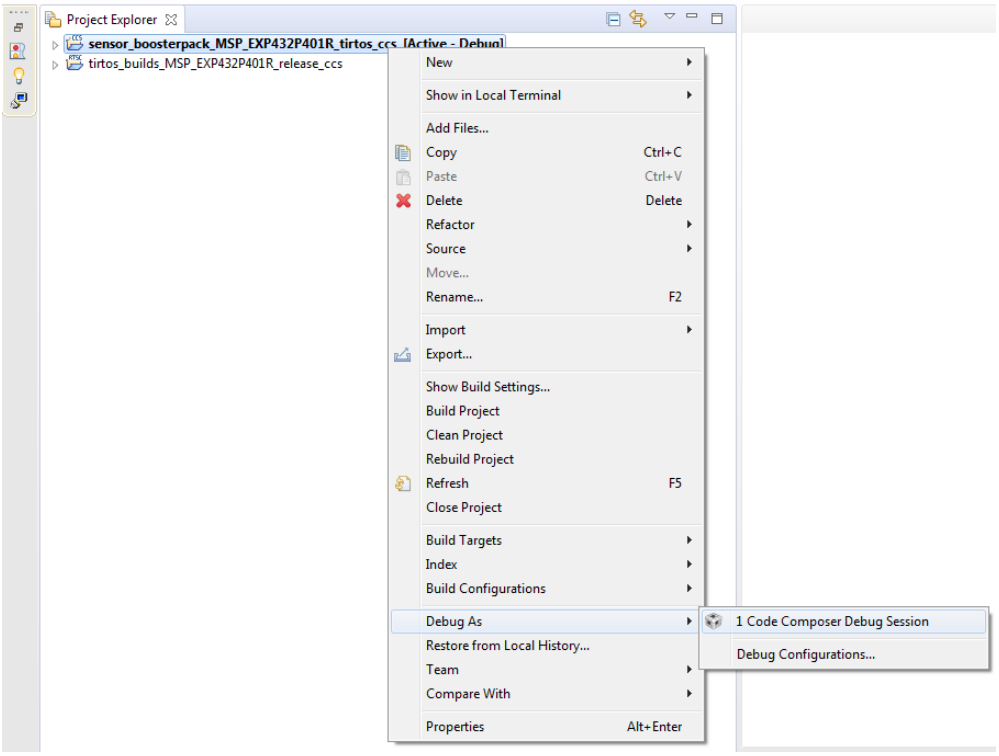
After clicking finish, the Sensor BoosterPack example will be imported to the IDE.



User's Guide Figure 10: Output of importing the example code project to the IDE

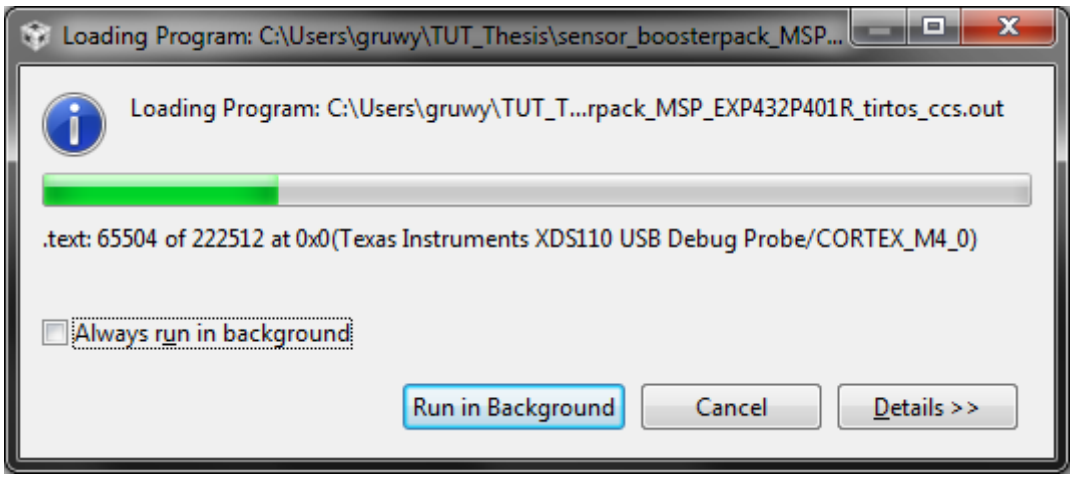
Step 2.4: Launching debug session.

In order to launch the debug session, user has to do the following: Right click on the "sensor_boosterpack_MSP_EXP432P401R_tirtos_ccs" → Debug As → "1 Code Composer Debug Session".



User's Guide Figure 11: Debug Session location

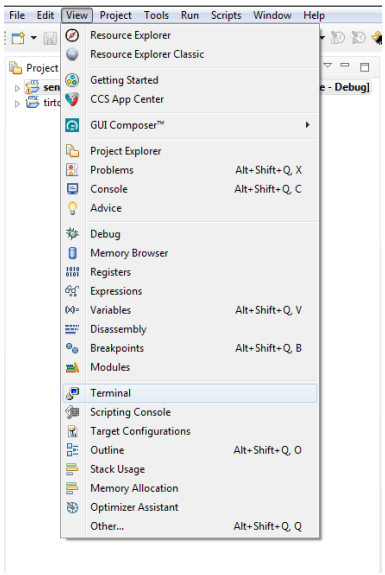
After this step, the IDE will launch the debug session. It may take some time for the personal computer to launch the example.



User's Guide Figure 12: Debug launching process

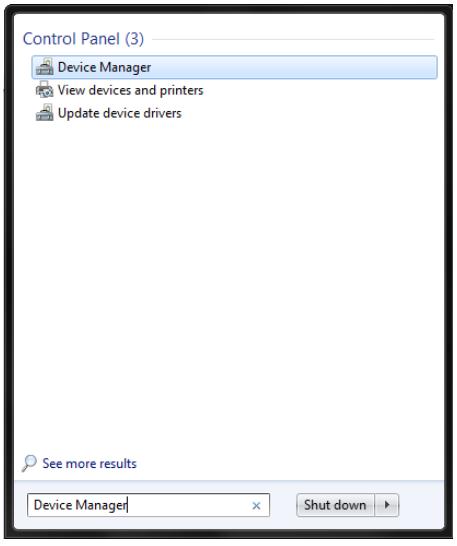
Step 2.5: Connecting to the MCU combination via CCS IDE terminal.

In order to locate the terminal in the IDE, user has to do the following: View → Terminal. The terminal window will pop up at the lower right corner of the screen.



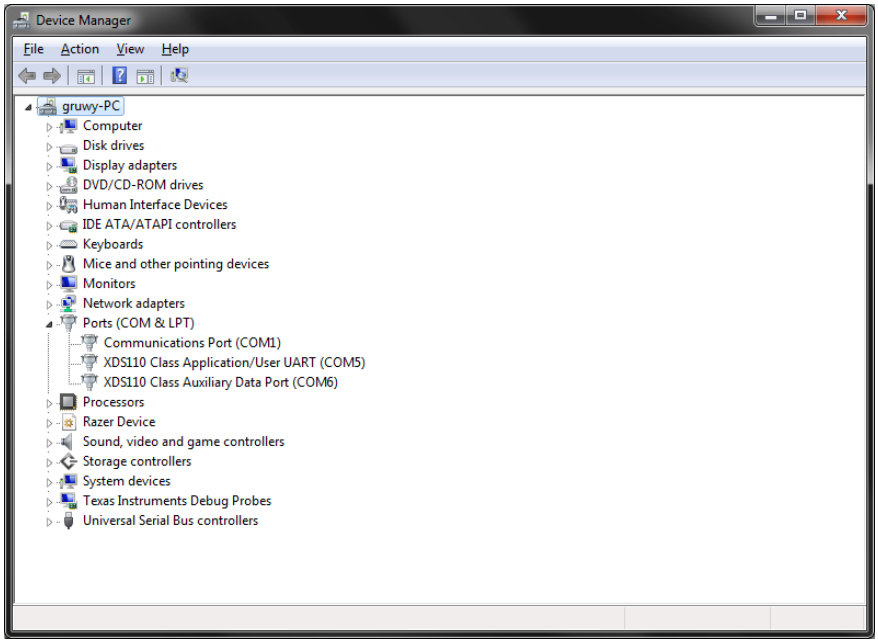
User's Guide Figure 13: Terminal location in CCS IDE

The serial port may be different due to personal computer's port allocation and settings. In order to find out the serial port required, user has to navigate the port at Device Manager application. To do this, following actions are required: Push "Windows" key → Write "Device Manager" in search window. The Device Manager should be acquired.



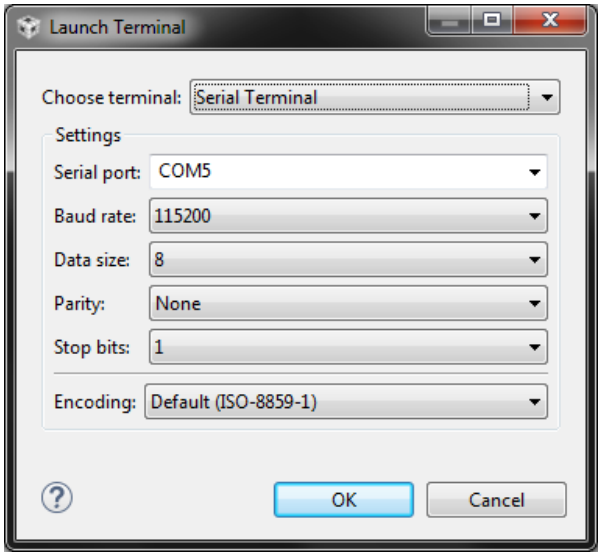
User's Guide Figure 14: Device Manager location in Windows OS

In order to find the correct port, the user has to navigate to “Ports (COM & LPT” tab and find the “XDS110 Class Application/User UART (COM5).



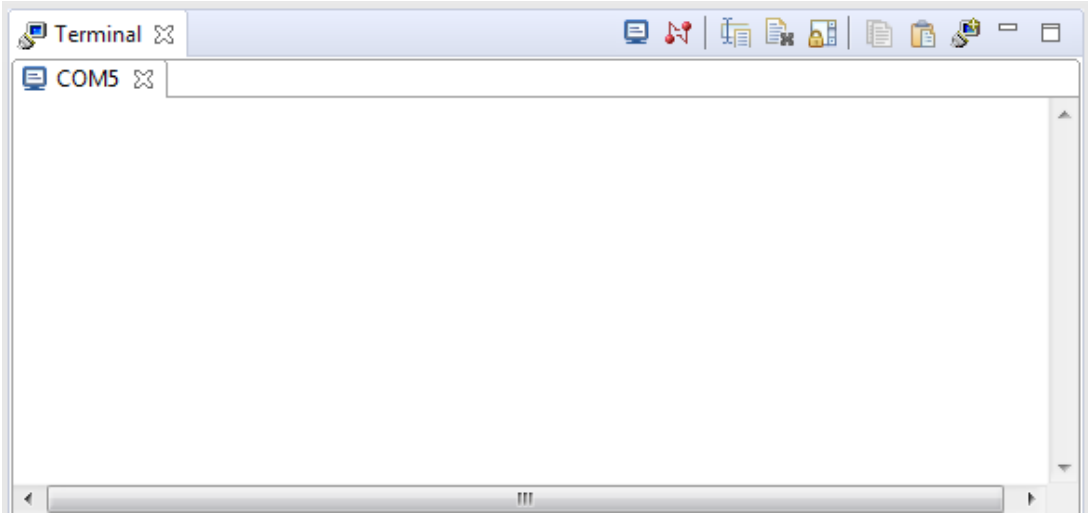
User's Guide Figure 15: Device Manager snapshot

To open the terminal settings, the user has to do the following keyboard combination: Ctrl + Alt + Shift + T. After opening the settings menu, the settings must be set to the same as the figure below.



User's Guide Figure 16: Terminal settings

After successfully connecting to the COM port, the terminal window should look like this.

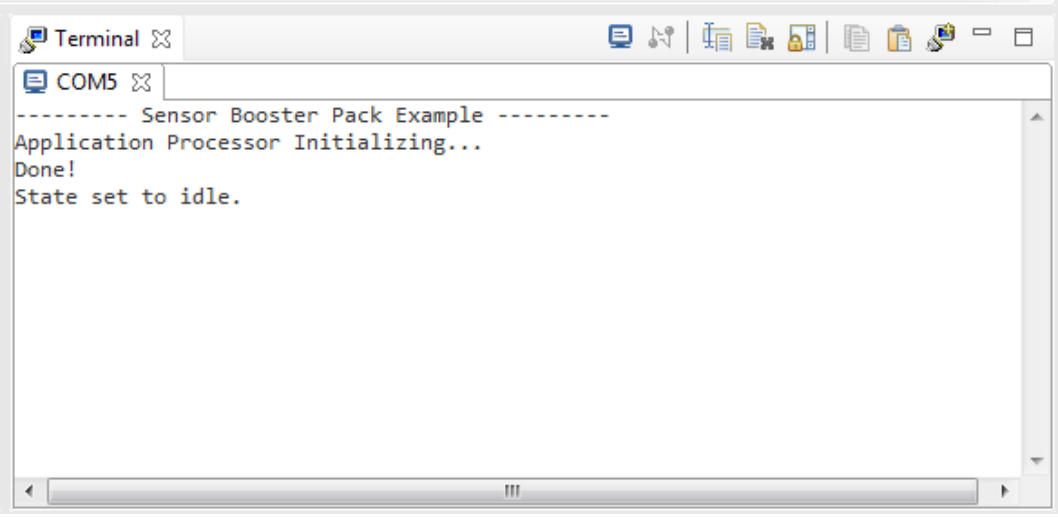


User's Guide Figure 17: Initial terminal view after successful connection via COM port

The IDE may contain errors during connection and it is advised to relaunch the IDE, if any errors occur.

Step 2.6: Running the example

To run the example, user has to push the “F8” button, when the system is focused on the IDE window. The successful running of the example is shown in the terminal window below.



User's Guide Figure 18: Successful example code execution in terminal

At this point, the example is successfully running, setting the MCU combination to idle.

2 Software preparation

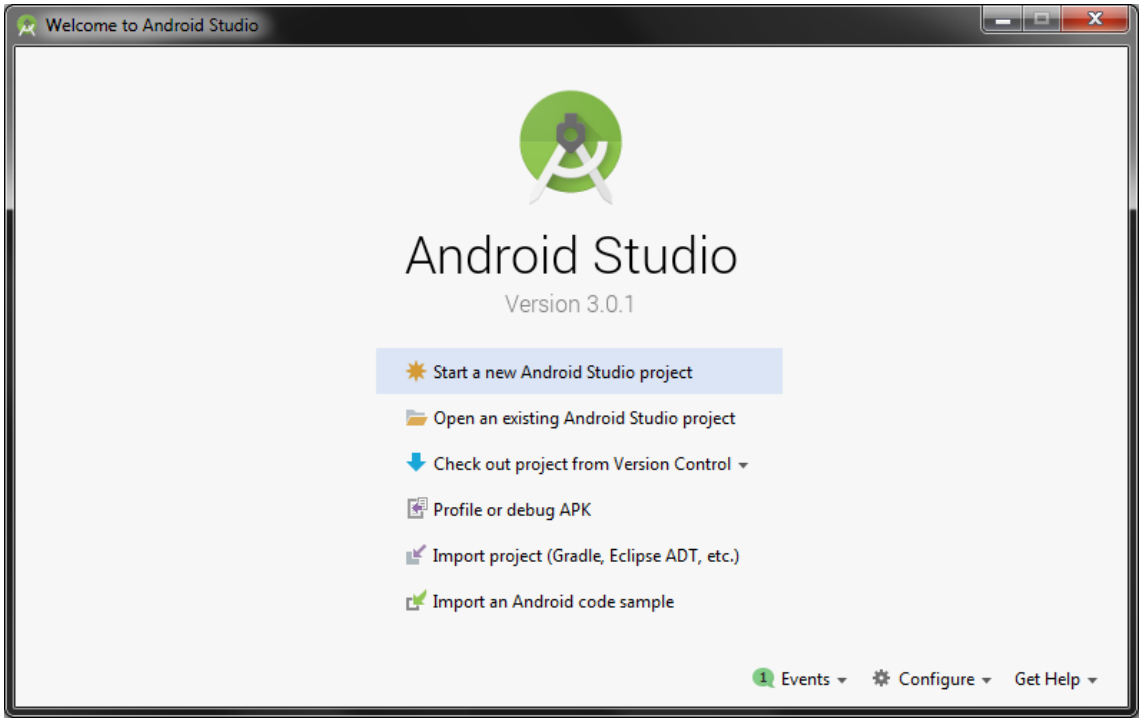
The software preparation includes running the Android application project in Android Studio IDE. The Android application is created for smart devices supporting API 23 (Android 6.0). Lower version Android smart devices will not be able to run the application.

2.1 Installing the Android Studio IDE

Step 1: The user has to acquire the latest version of Android Studio IDE, referencing to this link: https://developer.android.com/studio/

Step 2: After successfully downloading the IDE, the user has to install it on his personal computer following instructions set by manufacturer defaults.

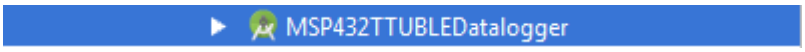
Note: the "Android Virtual Device" installation is not required for this project debugging. After installing the Android Studio IDE and launching it, the following "welcome" screen will appear.



User's Guide Figure 19: Snapshot from Android studio welcome screen

2.2 Importing the project to the IDE

Step 1: To import the Android application project the user has to do the following: "Open an existing Android Studio project" → locate the acquired project downloaded from the provided source.



User's Guide Figure 20: Look of the Android application project in the search window

Step 2: After choosing the project, user has to click "OK" to import the project. The IDE will build the project's gradle information and import it.

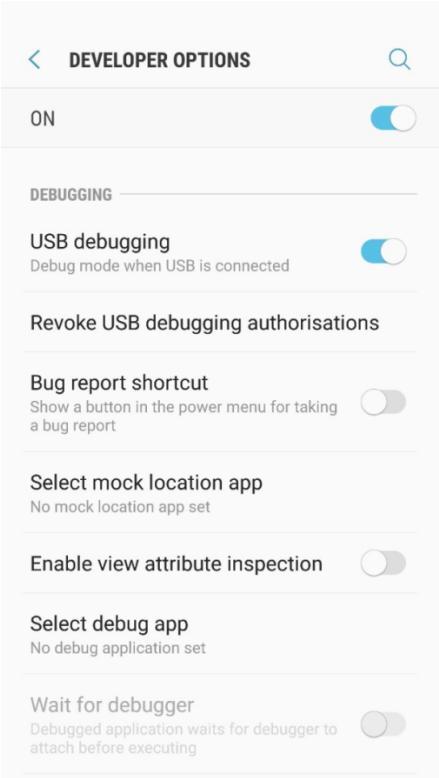
2.3 Running the project on the testing smart device

The testing smart device, as it was said before, has to have at least version 6.0 of Android OS and support Bluetooth Low Energy. The smart device must be connected to the personal computer via USB.

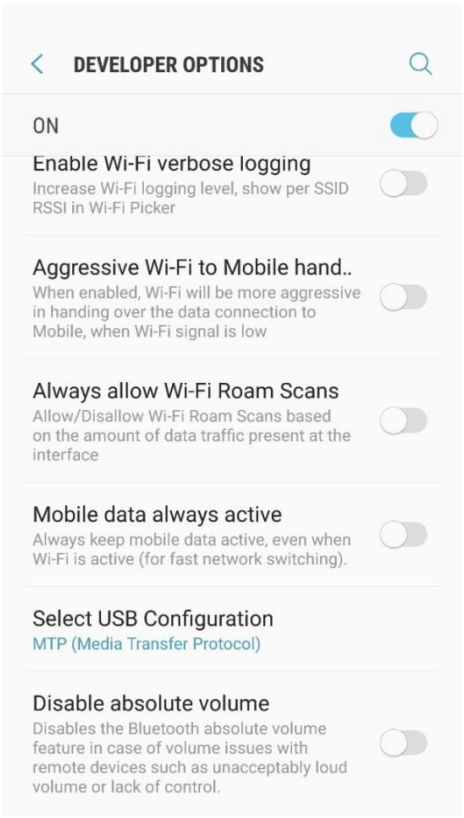
Step 1: Enabling "Developer options" mode on the testing smart device.

To successfully run the application on the testing smart device, it is required to enable "Developer options" mode. To enable the "Developer options" mode, the user has to reference to the testing smart device's documentation. After enabling the mode, it is required to enable two options:

- "USB debugging"
- "Select USB Configuration" → "MTP (Media Transfer Protocol)"



User's Guide Figure 21: USB debugging location in Android OS



User's Guide Figure 22: USB configuration location in Android OS

This will allow the IDE to find the testing smart device.

Step 2: Building and running the project.

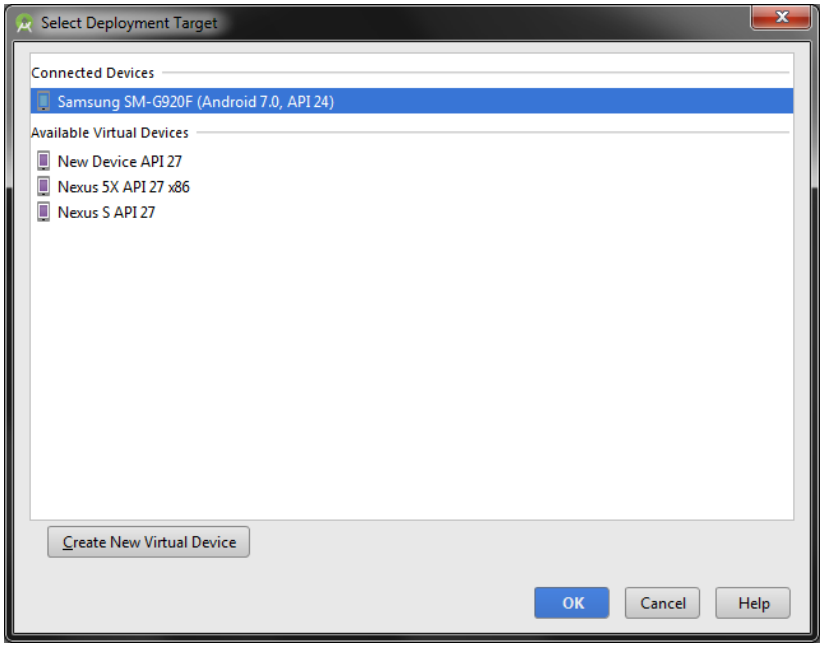
Step 2.1: In order to run the application, the IDE has to successfully build it. To build the project, the user has to execute following keyboard combination: “Ctrl + F8”. From now on, the IDE will start building the project and will show the finish of gradle build in the lower left corner.

User's Guide Figure 23: Successful gradle build

Step 2.2: To run the project, the user has to choose the running configuration by simply navigating to the location shown in the figure below.

User's Guide Figure 24: Running configuration location in Android Studio IDE

Clicking the green “play” button or executing following keyboard combination: “Shift + F10” will bring the user to the “Select Deployment Target” window.



User's Guide Figure 25: Deployment target selection window

If the testing smart device configuration was executed correctly, the user must see his smart device in “Connected Devices” tab. After executing all the steps above, the user has to push the “OK” button in order to let IDE import and install the APK and run the application.

The application may take some time in order to run on the device.

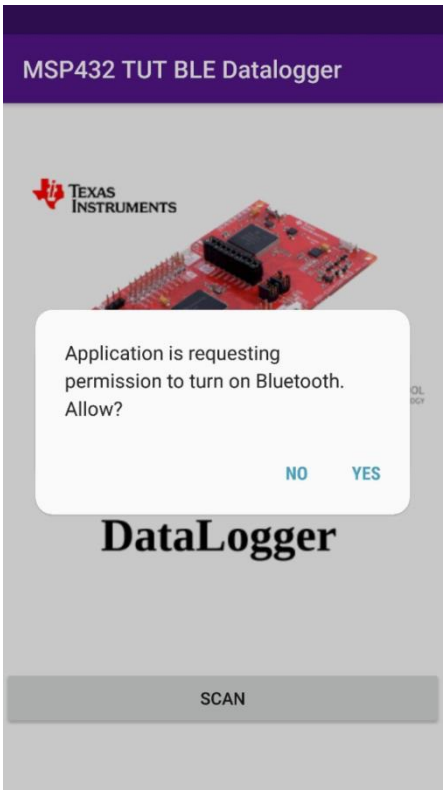
3 Working with the configured software and hardware

As the MCU combination debug session and Android application launched, the main process begins.

3.1 Meeting and allowing the Android application to use its permissions

The Android application requires Bluetooth adapter to be enabled and location permission to be granted by the user. As the Android application launches for the first time, it takes some time to execute the “cold” start, when all the dependencies and initialization are made for the first time.

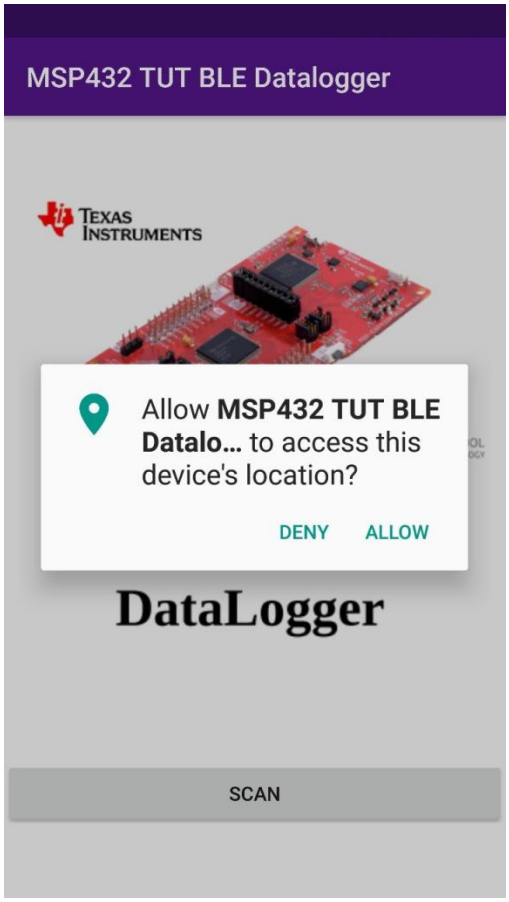
Step 1: The user has to wait for the Bluetooth permission request shown in the figure below.



User's Guide Figure 26: Snapshot from Android application with Bluetooth permission request

Clicking “Yes” enables the Bluetooth adapter on the testing smart device.

Step 2: Right after the Bluetooth permission granted, the Android application asks the user to allow to access the testing device's location.



User's Guide Figure 27: Snapshot from Android application with location permission request

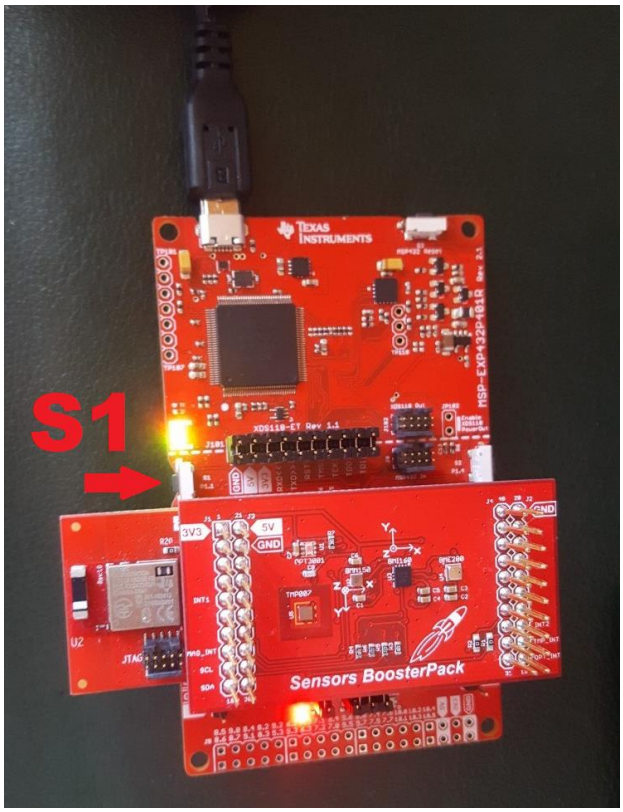
Clicking "Allow" grants access to use device's location to work with Bluetooth Low Energy technology. Note: this permission does not enable the "Location" option on the testing smart device nor track user's location by any means.

As the permissions are granted, the Android application represents the UI of "Start" activity.

3.2 Enabling advertising on the MCU combination

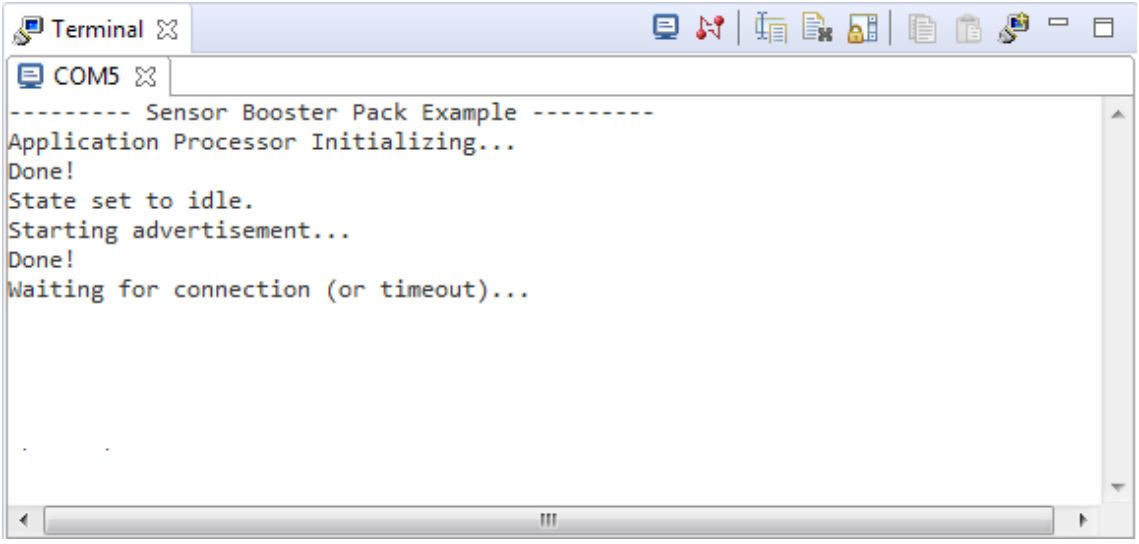
From this point the user has to move back to the CCS IDE and track his actions in the configured terminal.

To enable Bluetooth Low Energy advertising on the MCU combination, it is required to toggle "S1 (P1.1)" button on the MSP432 LaunchPad as shown on the figure below.



User's Guide Figure 28: S1 button location on MCU combination

If the advertisement is launched correctly, the LED1 on the MSP432 board will light up that will lead to the following output of the terminal.

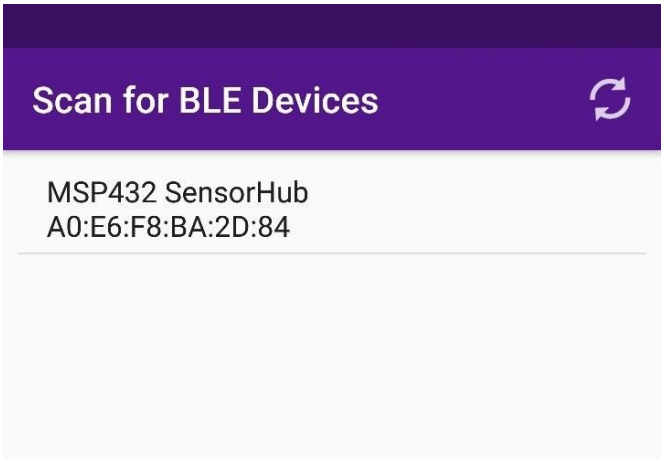


User's Guide Figure 29: Terminal output after successful advertisement enabling

From now on, user has 30 seconds in order to connect to the MCU combination before the example code cancels the advertisement.

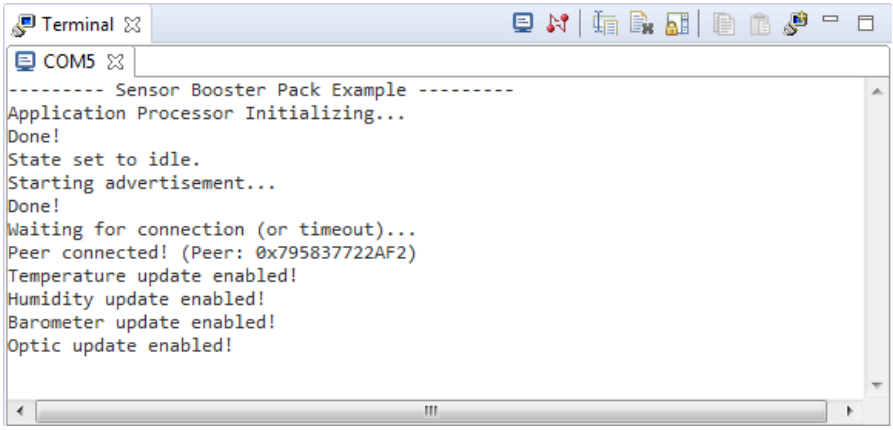
3.3 Scanning for the MCU combination and connecting

Step 1: In order to connect to the MCU combination, user has to navigate to the “Scan” activity using button “Scan”. After clicking the button, the application will transfer the user to scanning activity, where the testing smart device will search for the MCU combination. After successfully finding the MCU combination, the output of the scanner must be the following. The name, set by the example code, is “MSP432 SensorHub”.



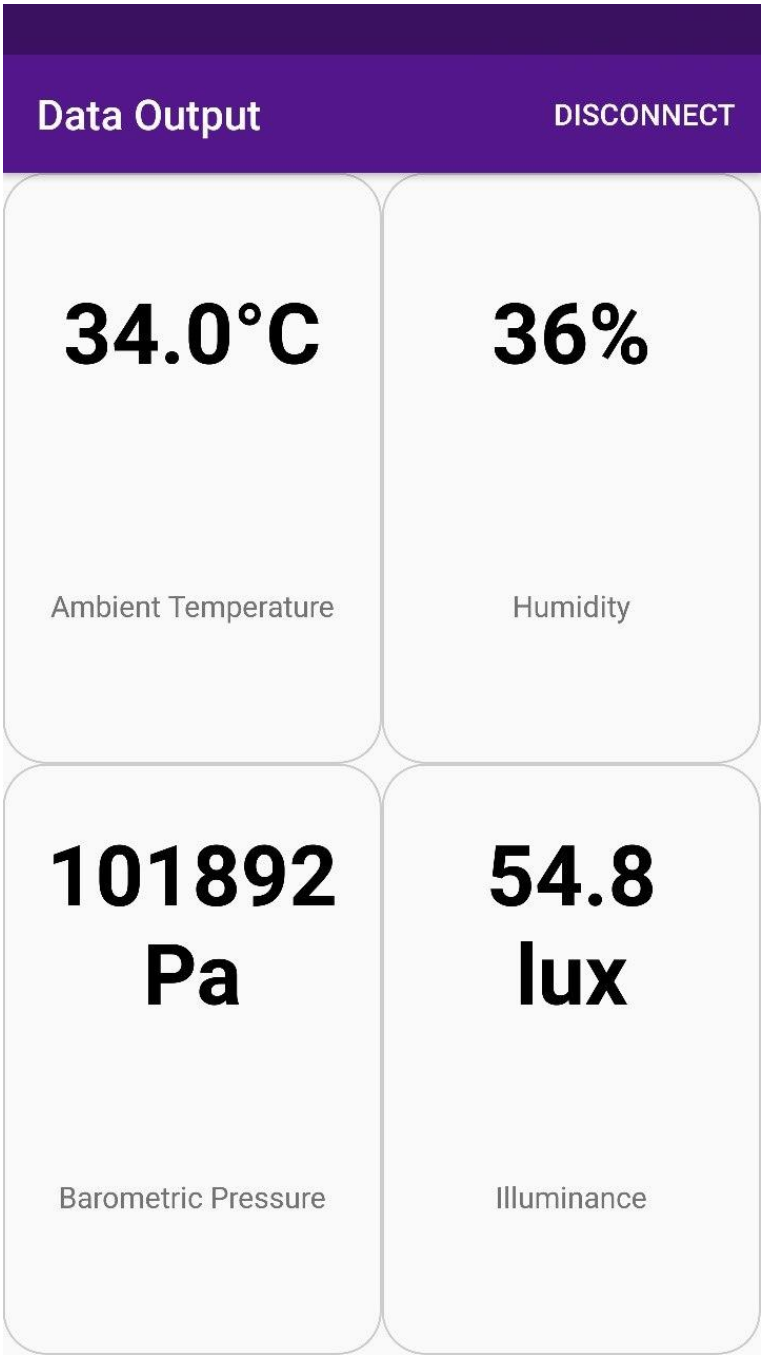
User's Guide Figure 30: Snapshot from Android application after successful find of MSP432 SensorHub

Step 2: In order to connect to the MCU combination, user has to click on the list view item and this will shift user to the “SensorOutput” activity, where the connection, sensor enabling and value representation occurs. As the smart device connects and enables the notifications on the MCU combination, the example code will output activity to the terminal.



User's Guide Figure 31: Terminal output after successful connection and sensor enabling

The final result of this activity will show an UI, where the readings of the sensor will be updating every second.



User's Guide Figure 32: Snapshot from Android application's "SensorOutput" activity

In order to disconnect from the MCU combination, the user has to simply click "DISCONNECT" menu item and he will be forced back to the "Start" activity.