

# Indoor Solar Development Kit with TI BLE (DEV-IN-BLE-TI) Operation Guide

## Introduction

Fully powered by indoor solar energy, the DEV-IN-BLE-TI collects data from a variety of sensors and sends it to the cloud via Bluetooth Low Energy (BLE). This development kit is meant to be the first building block of any IOT sensing project or idea. The design makes it simple to add solar to any Bluetooth sensing application. PowerFilm's Indoor Solar Panels power the DEV-IN-BLE-TI from indoor light sources, enabling completely self-sufficient IoT devices.

The DEV-IN-BLE-TI comes configured to measure temperature, light level, and battery level and includes a pin header to easily connect any additional external sensors.

This operation guide is a technical reference for all components and software related to the DEV-IN-BLE-TI (PFDEV-TI BLE device name) and the PFDEV-TI Data Monitor including setup and development instructions.

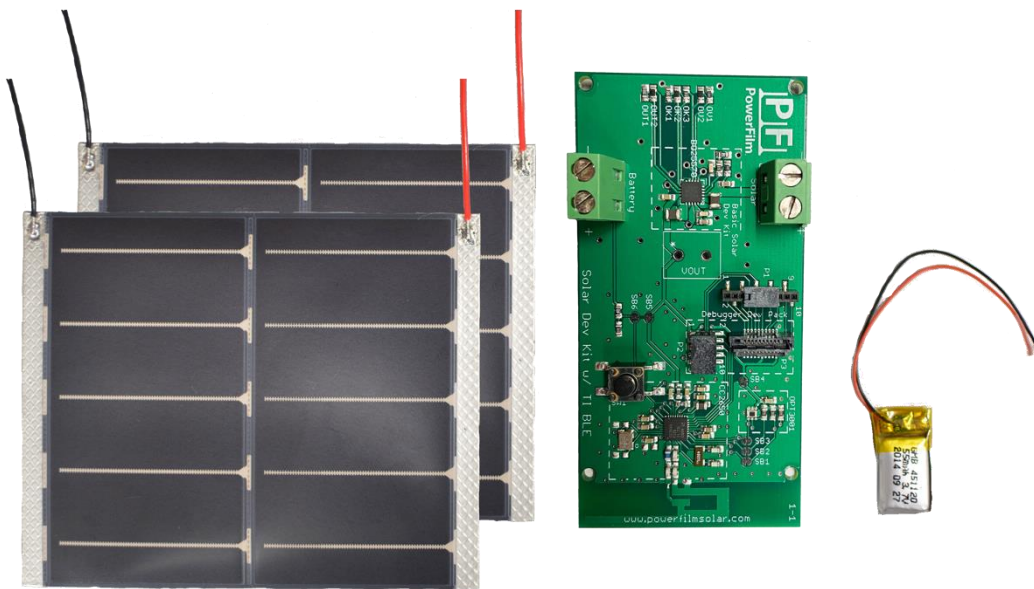


Figure 1, Product picture.

## Table of Contents

Introduction.....	1
Use Cases.....	4
Sensing Applications.....	4
IoT Applications .....	5
Included in Kit .....	5
System Overview.....	5
Technical Specifications .....	6
IC.....	6
Solar.....	6
Software and Compatible IDEs .....	6
Electrical .....	7
Quick Setup Instructions.....	7
Equipment .....	7
Setup .....	7
PFDEV-TI Data Monitor Notes .....	9
Home Page.....	9
Device Page.....	10
Battery Profile.....	12
Temperature Profile.....	13
Light Level Profile .....	14
Connection Control Service.....	14
Saving Settings.....	16
Software Development in Code Composer Studio .....	16
Additional Equipment .....	16
Installation Setup.....	16
Build and Download/Debug .....	29
Overview of CC2650 Code Structure .....	33

Adding a Custom Service or Profile .....	34
Profile Setup .....	34
External Sensor/Device Setup .....	34
Initialization .....	34
Reprogramming with Hex Files.....	35
Flashing HEX Files to the DEV-IN-BLE-TI CC2650 IC.....	35
Merging Compiled DEV-IN-BLE-TI Project Into a Single Hex File.....	37
Android Studio.....	39
Installation Setup .....	39
Build and Run/Debug PFDEV-TI Data Monitor.....	47
Overview of Android Code Structure.....	48
Structure .....	48
MainActivity.java .....	48
DeviceActivity.java .....	48
Bluetooth Profiles and Rows .....	48
Layouts .....	49
Other .....	49
Adding a Customized Service or Profile .....	49
SensorTagGatt.java modification.....	50
gatt_uuid.xml modification.....	50
Creating a Custom Bluetooth Profile Class .....	51
DeviceActivity.java modification .....	51
Custom Profile Row .....	52
Block Diagram .....	52
Diagram .....	52
Block Diagram Notes .....	53
BQ25570 Energy Harvester .....	53
CC2650 Wireless MCU .....	53
PFDEV-TI Data Monitor.....	53

Schematic.....	53
Overview .....	53
DEV-IN-BASIC BQ25570 Energy Harvester .....	54
Bluetooth MCU .....	55
I/O .....	57
Ports and Connectors.....	57
Switches .....	57
Layout.....	58
Top.....	58
Bottom .....	58
Part Placement .....	59
DEV-IN-BLE-TI Application Notes .....	59
Power Consumption Optimization.....	59
Hardware Configurations .....	59
I/O Access .....	59
Charge Control Configuration .....	60
Capacitor/Super Capacitor Storage Element Operation .....	60
Theory of Operation.....	60
Specifications .....	61
CC2650 Microcontroller Reset .....	62

## Use Cases

### Sensing Applications

Temperature	Humidity	Pressure
Light Level	Vibration	Motion
Acoustic	Air Flow	Water Flow
Strain	Moisture	Air Quality
Occupancy	Acceleration	Water Level

## IoT Applications

Home Automation  
Smart City  
Asset Tracking  
Beacons

Wearables  
Smart Agriculture  
Smart Transportation  
Equipment Monitoring

Smart Buildings  
Wireless Nodes  
Gateways  
Industrial Automation

## Included in Kit

- DEV-IN-BLE-TI Board: fully assembled (battery and solar not connected)
- Solar Panel: LL200-2.4-75 Amorphous Silicon.
- Battery: 3.7V nominal Lithium Ion Polymer, 40mAh, circuit protection

## System Overview

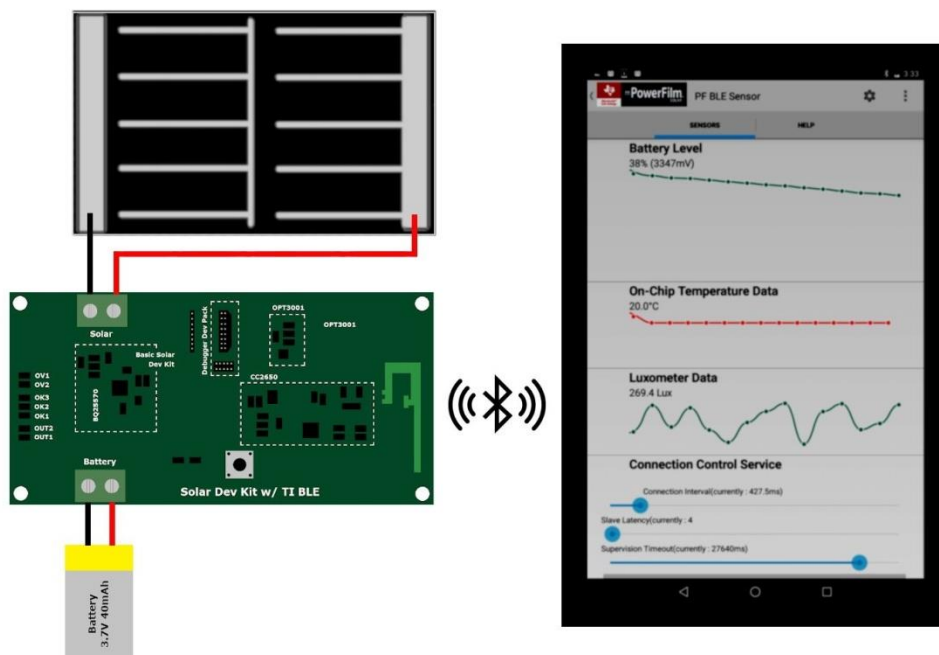


Figure 2, Connected System

The DEV-IN-BLE-TI dev board system extends the TI CC2650 SensorTag Bluetooth and data collection by integrating energy harvesting and an indoor solar panel. It is capable of operating at extremely low light levels, down to 200 lux and below.

DEV-IN-BLE-TI is designed to be fully customizable. DevPack and JTAG connectors are present for debugging and adding TI DevPack plug-in modules. All IO pins can be accessed via a 10-pin female header so that external sensor can easily be added. The current onboard sensors can be disconnected by removing SMD jumpers to allow access to all 10 IO pins.

Software development can be done using Code Composer Studio and Android Studio, which are both free to use. Hardware modifications can be done using EAGLE CAD software.

The CC2650 device is a wireless MCU targeting Bluetooth, ZigBee® and 6LoWPAN, and ZigBee RF4CE remote control applications. The protocol provided with the development kit is for Bluetooth LE, however other protocols could be loaded as well.

## Technical Specifications

### IC

- CC2650 SimpleLink Multi-Standard Wireless MCU.
- BQ25570 Nano Power Boost Charger and Buck Converter for Energy Harvesting Applications.
- OPT3001 Digital Ambient Light Sensor.

### Solar

- LL200-2.4-75 Two Cell Tandem Amorphous Silicon 0.3 – 1.9mW @ 200 – 1000 lux.
- Thickness: 10mil
- Bend Radius: 2cm

### Software and Compatible IDEs

- Android Studio.
- Code Composer Studio v6.2.0.00048.
- TI BLE 2.2.1 Bluetooth Low Energy Stack.

## Electrical

- Pre-configured for 3.7V Lithium battery charging.
- <10uA system sleep current.
- Operational down to 200 lux.
- 10mA peak transmission current (typical).
- See CC2650 Datasheet.
- See BQ25570 Datasheet.

## Quick Setup Instructions

The following instructions will outline how to set up the DEV-IN-BLE-TI (PFDEV-TI BLE device) to connect and send data to the PFDEV-TI Data Monitor application using the preloaded programs and configuration.

## Equipment

- DEV-IN-BLE-TI Development board.
- LL200-2.4-75 Indoor Solar Panel.
- Lithium Ion battery (4.2V Max) or compatible capacitor/supercapacitor.
- Android Device with Bluetooth Low Energy capability.
- Small flathead Screwdriver.

## Setup

Refer to Figure 2 to help find noted location on assembled circuit board. References to PFDEV-TI Data Monitor Figures 5,6, and 7 are located in the section "PFDEV-TI Data Monitor Notes".

1. Open all screw terminals.
  - a. Using the flathead screwdriver, turn each screw terminal counterclockwise until it is in the open position.

NOTE: The terminals must clamp down on bare metal and not the insulation.  
Strip insulation off the end of wires if needed.

2. Connect the provided lithium ion battery to the DEV-IN-BLE-TI board.
  - a. Insert the positive red battery wire into the Bat+ terminal. Screw the terminal down until the wire is tightly held.

- b. Insert the negative black battery wire into the Bat- terminal. Screw the terminal down until the wire is tightly held.
3. Connect the LL200-2.4-75 Indoor Solar Panel to the Solar input.
  - a. Insert the positive red solar wire into the Solar+ terminal. Screw the terminal down until the wire is tightly held.
  - b. Insert the negative black solar wire into the Solar- terminal. Screw the terminal down until the wire is tightly held.

NOTE: If using a capacitor or supercapacitor, allow adequate charge up time in decent indoor light (300 lux). See Figures 63 and 64 in the Capacitor Storage Element Operation section in the Application Notes for estimated charge times.



Figure 3, Assembled DEV-IN-BLE-TI Circuit Board.

4. Download the PFDEV-TI Data Monitor app from Google Play store,

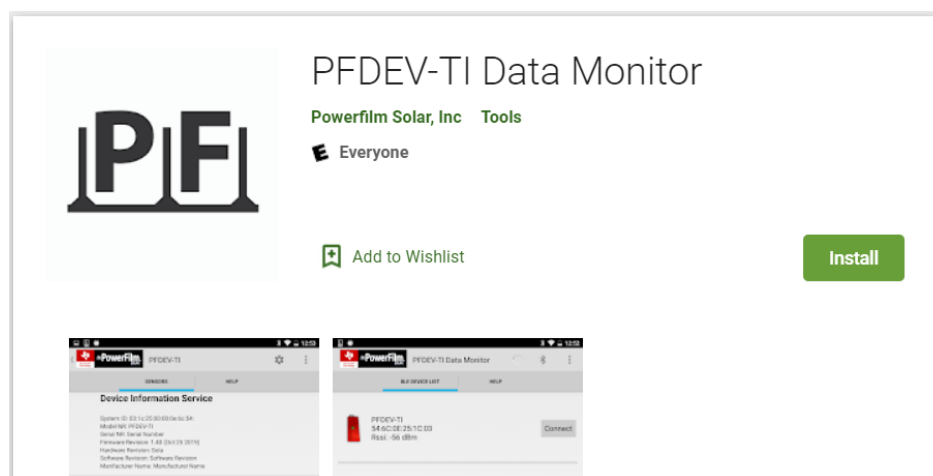


Figure 4, PFDEV-TI Data Monitor Google Play Store Listing.



5. Open the PFDEV-TI Data Monitor and scan for PFDEV-TI devices.
  - a. From the home screen click Scan (Figure 5). Bluetooth devices found will be displayed on the screen.
  - b. Once the PFDEV-TI device is found, Click Connect (Figure 6).
6. Once connected, observe light level, battery level, and temperature displays (Figure 7).
7. Configure sensors as desired. Changes to connection parameters are automatically saved (Figure 7).
8. If connection is lost, the PFDEV-TI Data Monitor will automatically reconnect to the last connected device if found, as long as Bluetooth scanning is not reset (turned off then back on).

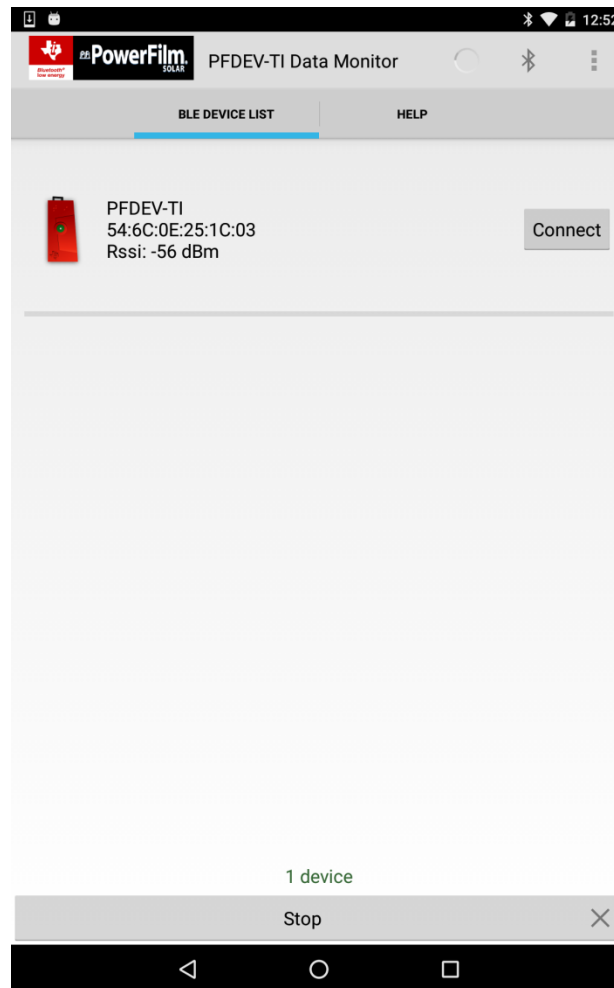
## PFDEV-TI Data Monitor Notes

### Home Page

The PFDEV-TI Data Monitor home page can enable/disable Bluetooth scanning and displays Bluetooth devices found in a scrollable list.

To enable/disable Bluetooth scanning, click the Start Scanning or Stop button at the bottom of the screen (Figure 5).

To connect to a found PFDEV-TI device, click the Connect button next to the desired device (Figure 6).



*Figure 5, DEV-IN-BLE-TI Data Monitor Home Screen*

The PFDEV-TI Data Monitor remembers the last device it was connected too so if connection is lost it will automatically reconnect if it finds it again. Clicking the Stop scanning button will reset the device memory so the app will not automatically reconnect.

## Device Page

After successfully connecting to a PFDEV-TI device, the device home page will be displayed. All of the PFDEV-TI Bluetooth profiles and services will be displayed as

separate rows on the page (Figure 7). Each Profile or Service performs a different function such as displaying collected sensor data or adjusting Bluetooth connection parameter.

The PFDEV-TI Bluetooth profiles and services include the Battery Level Profile, the Temperature Profile, the Light Level Profile, the Connection Control Service, and the Device Information Service.

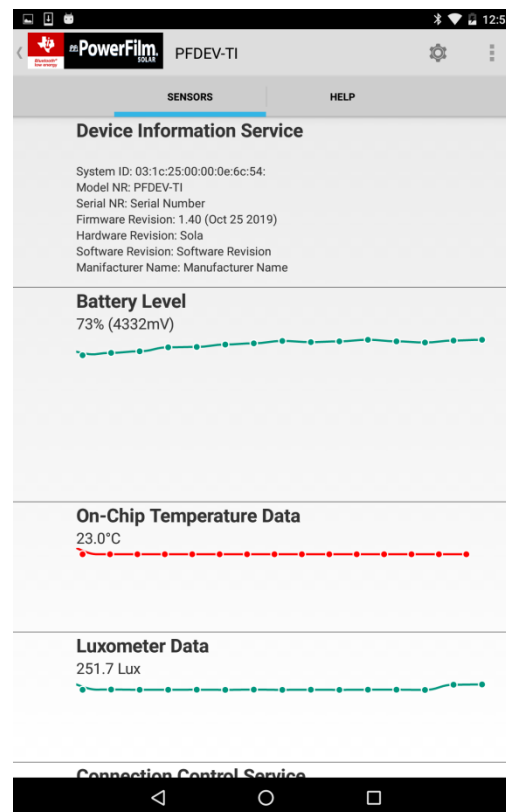


Figure 7, DEV-IN-BLE-TI Data Monitor Device Screen

## Battery Profile

The Battery Profile displays battery level data collected by the battery monitoring circuitry. Data is plotted on a line chart while the measured battery state of charge (SOC) [%] and actual battery voltage is displayed in real time above it (Figure 8).

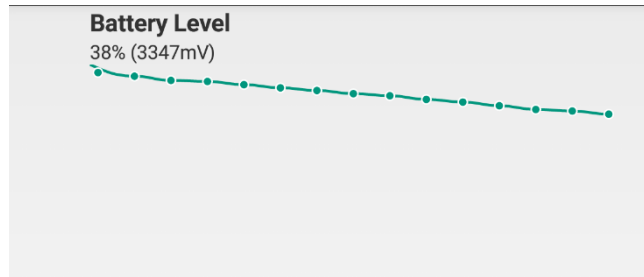


Figure 7, Battery Level Data Display Row

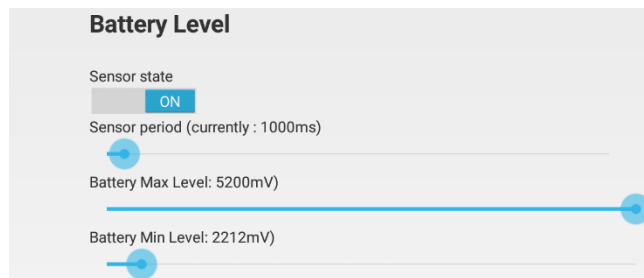


Figure 9, Battery Level Configuration.

Clicking on the Battery Profile row will reveal its configurable settings (Figure 9).

The battery level sensor can be configured ON/OFF and the sensing period can be adjusted from 100ms – 25.5s.

Battery Max Level and Battery Min Level determine state of charge percentage that is displayed above the line chart and can also be adjusted to accommodate for different battery types and setups. The state of charge is calculated by comparing the current battery voltage to the maximum and minimum battery voltage settings.

$$SOC( \ ) = \frac{(BatteryMaxLevel - CurrentBatteryLevel)}{(BatteryMaxLevel - BatteryMinLevel)} \cdot 100$$

Battery level is measured by a resistive divider  $R_{BM\_1}$  and  $R_{BM\_2}$ . The measured voltage is then sent to the PFDEV-TI Data Monitor where it is scaled to reflect the actual battery voltage.

The Battery Level calculation is based a linear charge scheme like those seen in capacitors. It does not reflect the true state of charge for storage elements with non-linear charge curves, such as rechargeable lithium batteries.

NOTE: Changing Battery Max Level and Battery Min Level has no effect on the charging scheme of the PFDEV-TI device, it is only used to calculate SOC. Connecting batteries to the PFDEV-TI device that are incompatible with the hardware configured charging scheme could result in damage to the battery and the PFDEV-TI device.

## Temperature Profile

The Temperature Profile displays temperature data collected by the internal CC2650 sensor. Data is plotted on a line chart while the real time measured value is displayed above it (Figure 10).

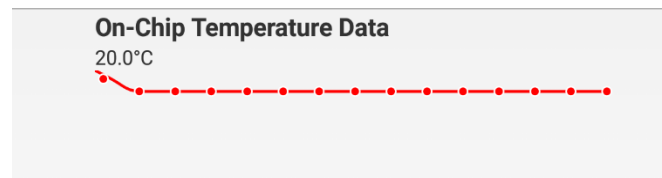


Figure 10, Temperature Configuration

Due to the low resolution of the internal 6-bit ADC, the temperature measurements are accurate to  $\pm 2^{\circ}\text{C}$ .

Clicking on the Temperature Profile row will reveal its configurable settings (Figure 11).

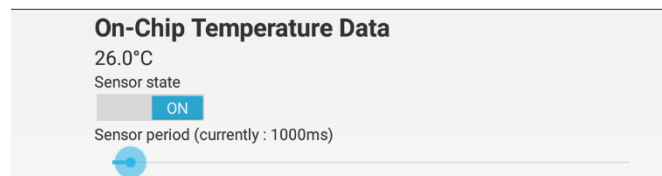


Figure 11, Temperature Configuration

The Temperature Profile can configure the temperature sensor ON/OFF and adjust the sensor period from 100ms to 25.5 seconds.

## Light Level Profile

The Light Level Profile displays data collected by the OPT3001 lux-meter. Data is plotted on a line chart while the real time measured value is displayed above it (Figure 12).

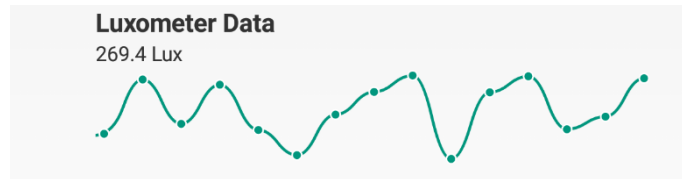


Figure 12, Light Level Data Display Row

The OPT3001 is capable of measuring light levels from .01-83,000 Lux. For perspective, an indoor room with no windows will be illuminated at approximately 300 Lux. Direct sunlight will measures approximately 100,000 Lux.

Clicking on the Light Level Profile row will reveal its configurable settings.

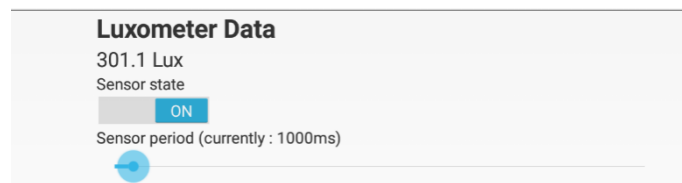


Figure 13, Light Level Configuration

The Light Level Profile can configure the sensor ON/OFF and the adjust the sensor period from 100ms-25.5s.

## Connection Control Service

The Connection Control Profile can adjust the Bluetooth Connection Interval, Slave Latency, and Timeout. These parameters define how the DEV-IN-BLE-TI and PFDEV-TI Data Monitor behave after connection. A connection event refers to a specific time when the two devices send data back and forth. This could be data collected from sensors or a handshake to verify the other device is still present.

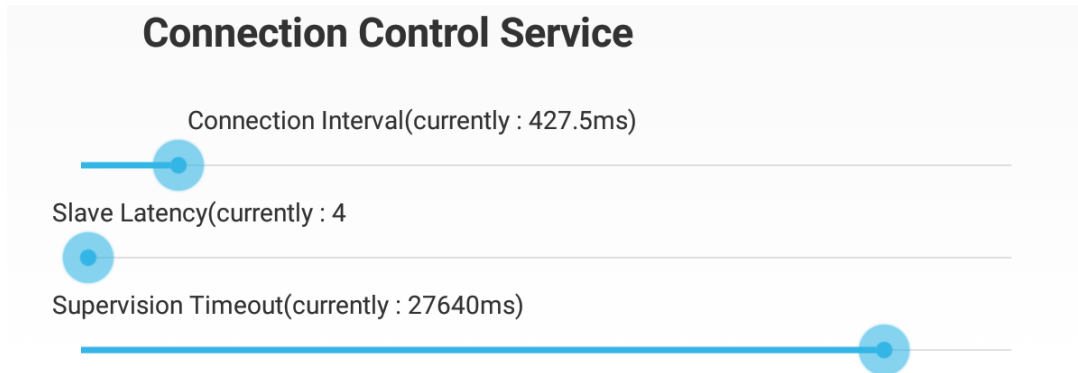


Figure 14, Connection Control Service Row

The Connection Interval sets the time between connection events. After connecting and after connection events, each device will wait exactly one Connection Interval and then attempt a connection event with the other device. If the other device is not found it will wait another connection interval and try again. If the device continues to not find the other device a timeout will occur and the connection will be lost.

The Slave Latency is the maximum consecutive number of times a device can skip a connection event without the connection being lost. At the time of a connection event, if a device has no data to transfer it can skip the event and stay asleep to conserve power.

The Supervision Timeout is the maximum amount of time allowed between two successful connection events. If a successful connection event does not occur within the Supervision Timeout, the connection will be severed.

According to the Bluetooth Low Energy protocol, the maximum allowed time between connection events is 16 seconds. This restricts the values that the parameters can be set at.

In General;

- The Connection Interval multiplied by the Slave Latency + 1 must be less than 16seconds.
- The Supervision Timeout must be greater than the previous bullet point and must be less than 32 seconds.

To modify the connection settings, adjust the parameters to the desired levels and click the Send Parameters button.

Sending connection parameters does not guarantee that they will be accepted. Once received, the PFDEV-TI device can accept or reject the sent parameters or may wait until a more convenient time to change connection parameters.

To read the current connection settings click the Read Parameters button.

These settings greatly impact the power consumption of the PFDEV-TI device. These parameters, along with the sensor time period, can be optimized so that the device can operate down to 200 lux and below.

For more information on the Bluetooth connection parameters see Chapter 5 of the CC2640 and CC2650 SimpleLink Bluetooth Low Energy Software Stack 2.2.1 Developer's Guide.

NOTE: In this guide and in the PFDEV-TI Data Monitor, the parameter units are expressed in seconds whereas the CC2640 and CC2650 SimpleLink Bluetooth Low Energy Software Stack 2.2.1 Developer's Guide expresses the parameters by their actual integer values. For conversion from seconds to actual integer value see Chapter 5 of mentioned guide.

### Saving Settings

All settings in the app save automatically after they are changed. This includes the on/off and periods of all sensors, the minimum and maximum Battery levels, and the connection parameters.

When the app is closed and reopened, all settings will be configured to the last used values.

## Software Development in Code Composer Studio

### Additional Equipment

- SimpleLink SensorTag Debugger DevPack (Sold Separately).
- Micro USB to USB cable (Sold Separately).

### Installation Setup



Modification and customization of the original TI SensorTag code was implemented using Code Composer Studio v6.2.0.00048 and TI BLE v2.2.1 Bluetooth stack, which is free to download and use.

NOTE: Instructions are based on the Windows 10 OS.

- 1) Install TI BLE 2.2.1 Bluetooth stack.
  - a. Open the provided DEV-IN-BLE-TI Software folder.
  - b. Run the ble\_sdk\_2\_02\_01\_18\_setup.exe executable.
  - c. Follow the standard recommended installation process which will include installation of TI-RTOS for CC13xx and CC26xx Wireless MCUs 2.20.01.08.
    - i. If a previous version of TI-RTOS is already installed, TI-RTOS 2.20.01.08 may not be installed automatically and will need to be installed manually. Installation executable can be found in the included software.
    - ii. Verify that TI-RTOS 2.20.01.08 is installed into the TI home directory (default: C:\ti). See Figure 15.

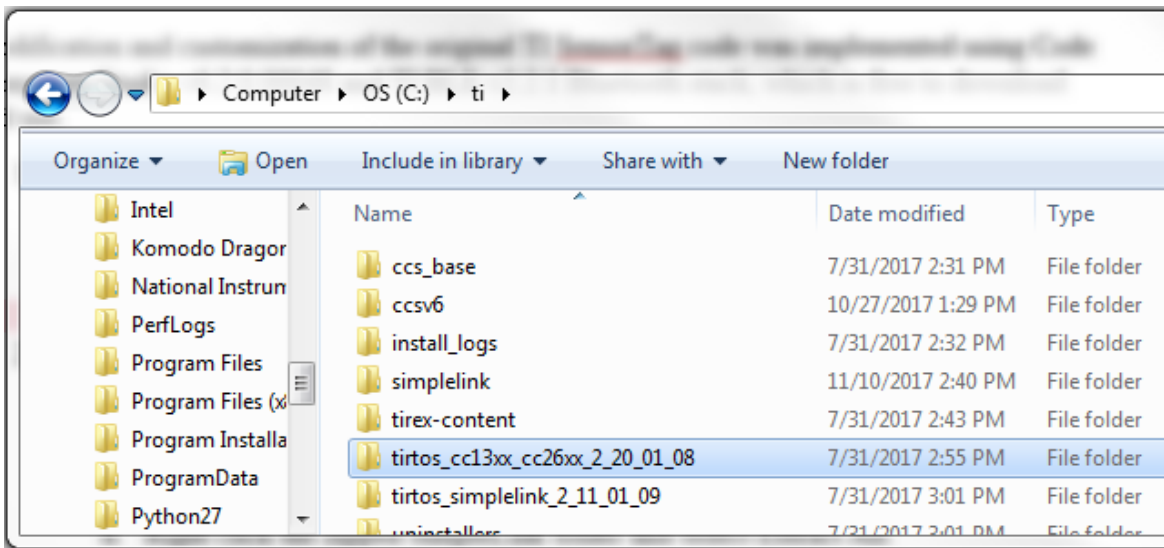


Figure 15, TI-RTOS 2.20.01.08 installation folder.

- 2) Update SimpleLink directory.
  - a. Navigate to the TI home directory (default C:\ti).
  - b. Rename the old SimpleLink folder to save a copy of the original. Ex. "C:\ti\SimpleLink\_Original"

- c. Open the provided DEV-IN-BLE-TI Software folder.
- d. Right click the zipped SimpleLink folder and Select Extract All.
- e. Click Browse.
- f. Navigate to and select the TI home directory (default C:\ti).
- g. Click Extract.
- h. Wait for the Extraction to finish. The TI home directory should now contain both the original simplelink folder and the provided simplelink folder (Figure 16).

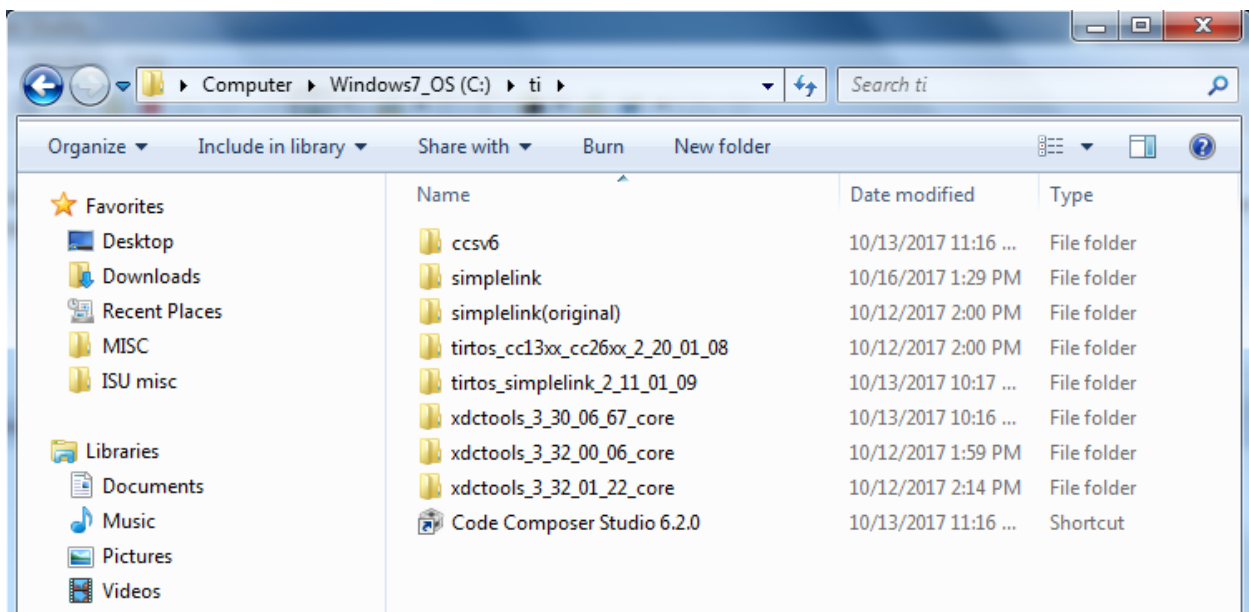


Figure 16, TI home directory after updating SimpleLink folder.

### 3) Install CCS v6.2.00048

- a. Download CCS v6.2.00048 from TI CCS downloads webpage:  
[https://software-dl.ti.com/ccs/esd/documents/ccs\\_downloads.html#code-composer-studio-version-6-downloads](https://software-dl.ti.com/ccs/esd/documents/ccs_downloads.html#code-composer-studio-version-6-downloads)
- b. Turn off virus protection and firewall before installation. They can interfere and cause installation errors
- c. Run the ccs\_setup\_6.2.0.00048.exe file.
- d. Accept terms and agreement. Press Next
- e. Select CCS Installation/Home folder (default is c:\ti, Figure 17). Press Next.

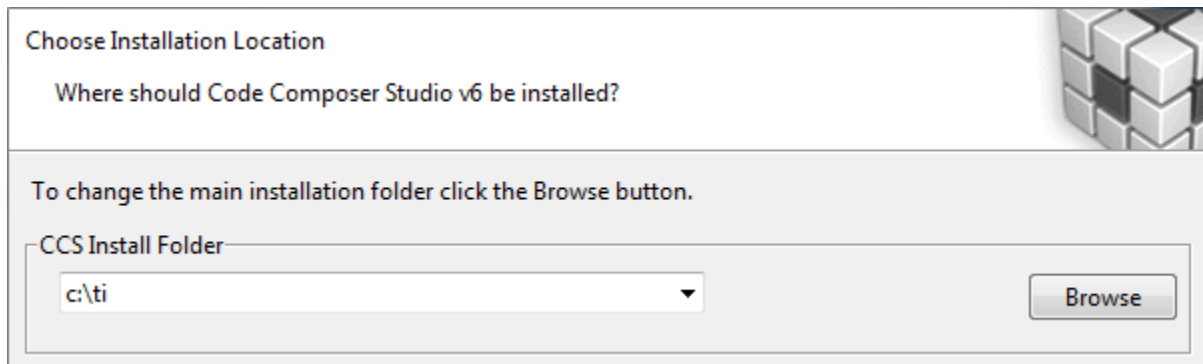


Figure 17, TI Installation Location

- f. For Processor Support: Check SimpleLink Wireless MCUs. CC2538 device support, CC26xx device support and TI ARM Compiler options are required (Figure 18). Press Next

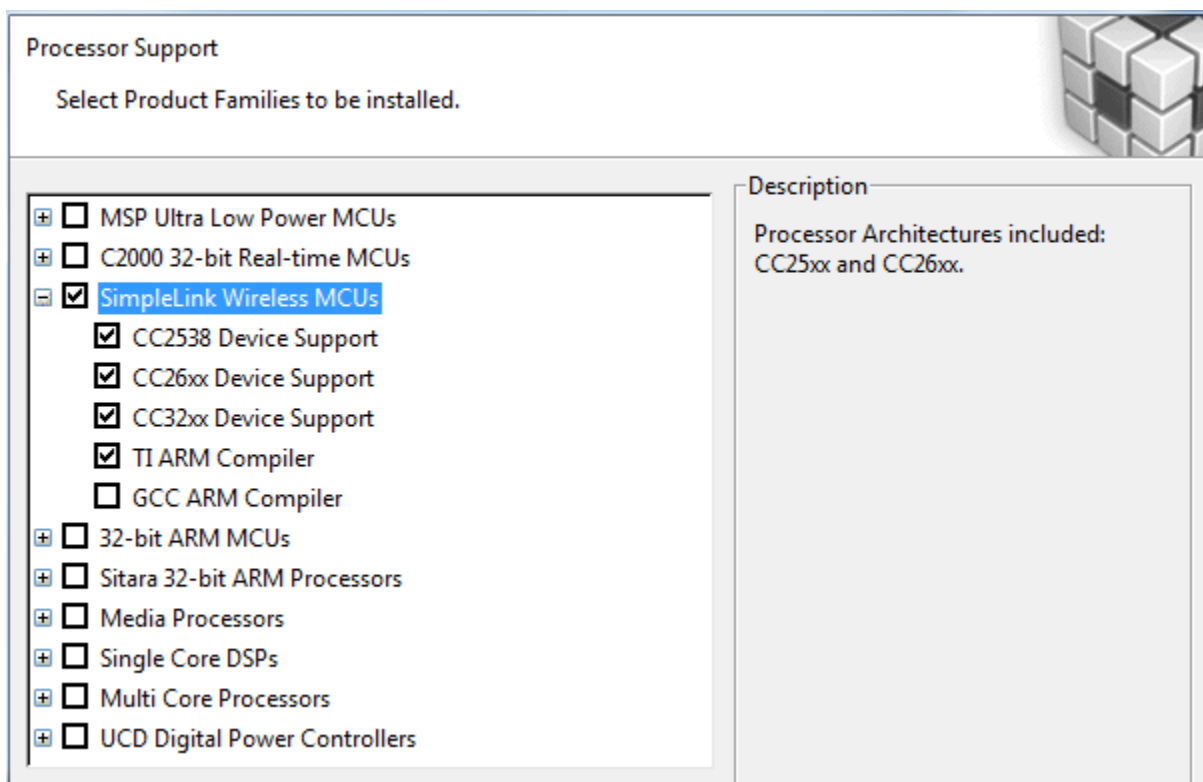


Figure 18, Processor Support Options

- g. For Debug Probe Options, TI XDS and Tiva/Stellaris ICDI debug probe support are required (Figure 19). Press Next.

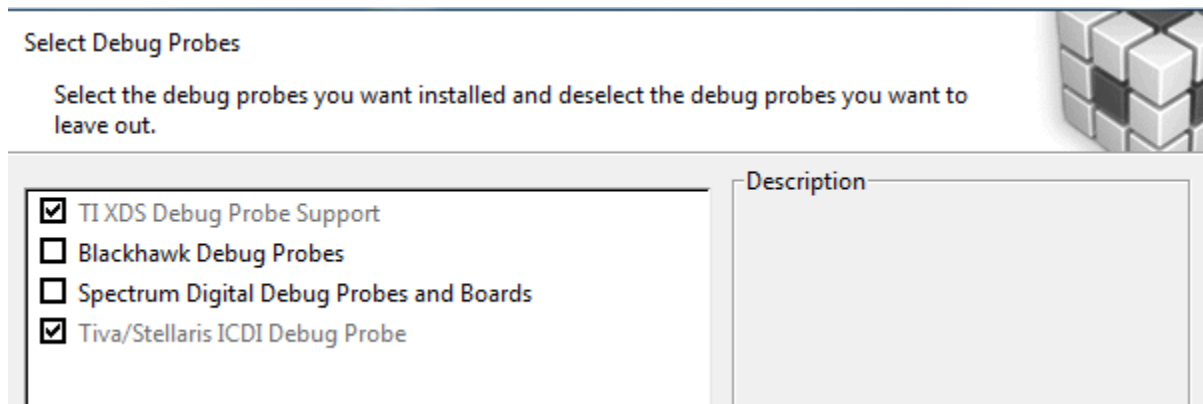


Figure 19, Debug Probe Options

- h. No App Center options are required (Figure 20). Press Next.

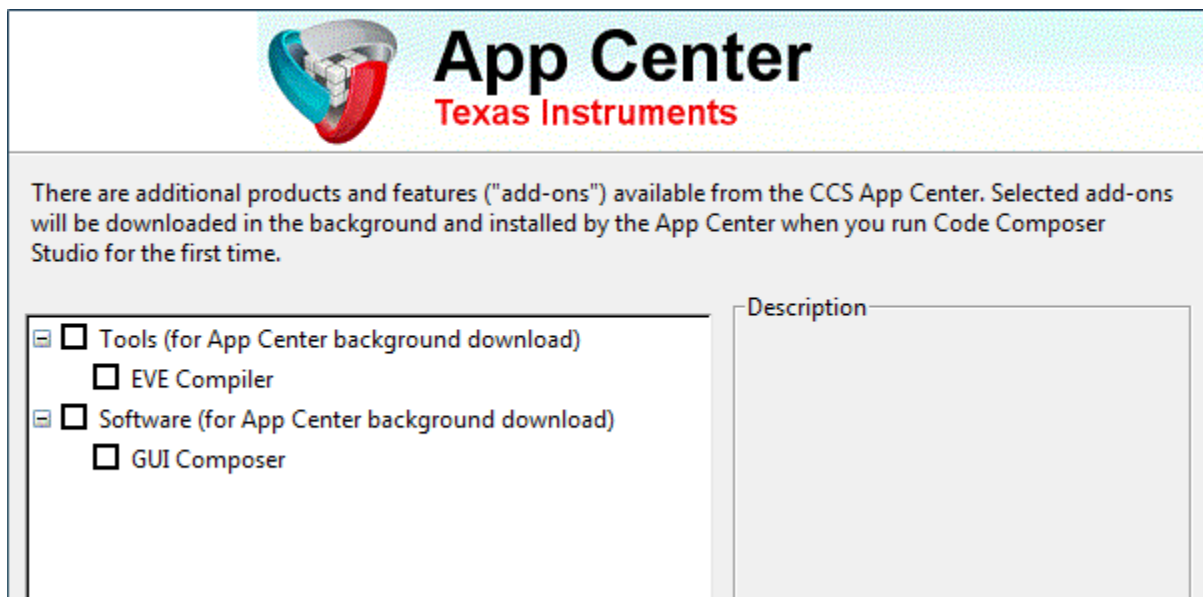


Figure 20, App Center Options

- i. You may need to allow the program firewall access.
- j. Press Finish.
- 4) Update the CCS flashCC26xx.dll file.
- Open the provided software files.
  - Right click and copy FlashCC26xx.dll.
  - Navigate to the <CCS62 INSTALL DIR>\ccsv6\ccs\_base\DebugServer\bin directory (default install directory is C:\ti) and find FlashCC26xx.dll.

- d. Rename the old FlashCC26xx.dll to FlashCC26xxVOID.dll.
- e. Paste the new FlashCC26xx.dll copied from the provided software.
- f. The directory should now resemble Figure 21.

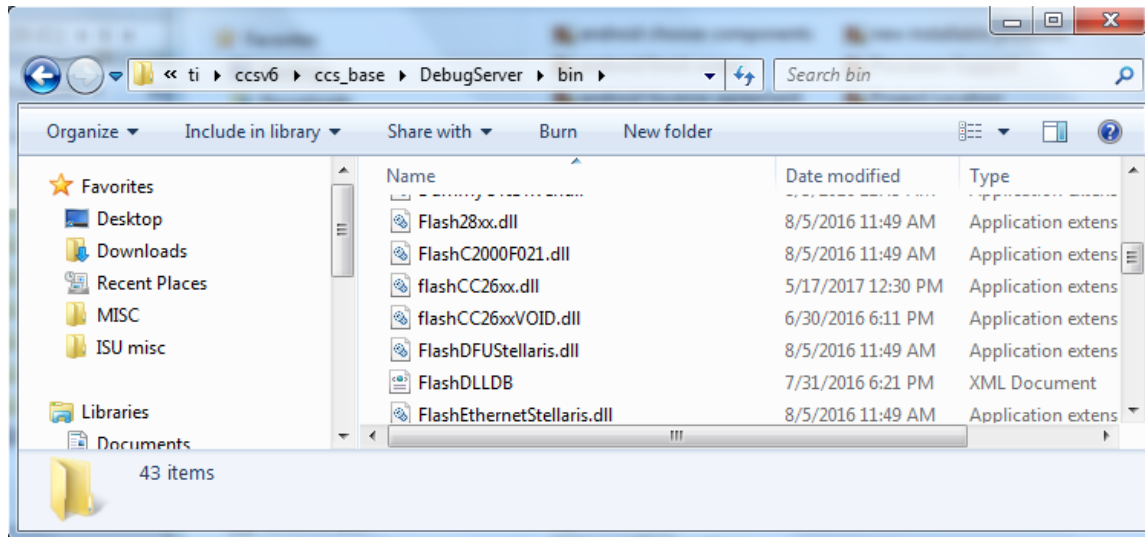


Figure 21, Updated <CCS62 INSTALL DIR>\ccsv6\ccs\_base\DebugServer\bin directory.

- 5) Install TI 5.2.6 ARM compiler. Refer to Figure 22 and 23 for help identifying mentioned fields.
  - a. Open CCS (Code Composer Studio)
  - b. On the top menu bar click Help – Install New Software (Figure 22). This will take you to the Available Software Installation page (Figure 23).

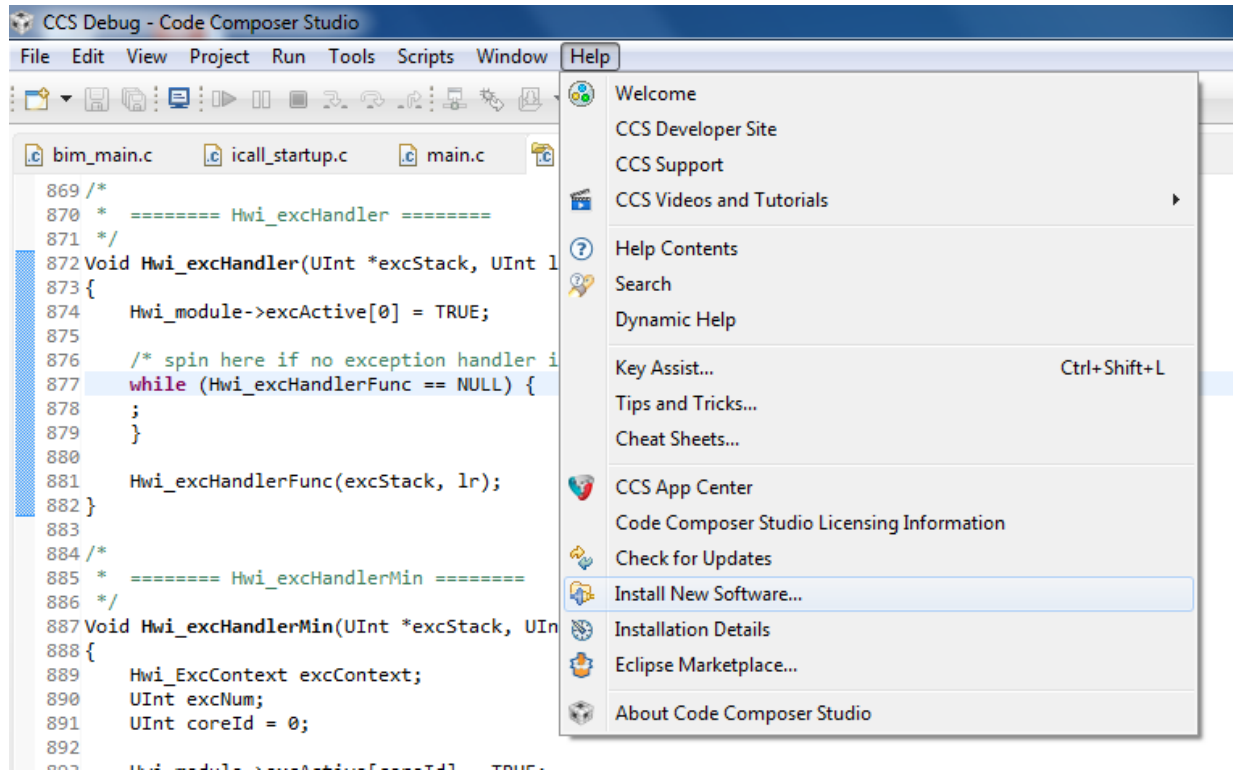


Figure 22, Install New Software

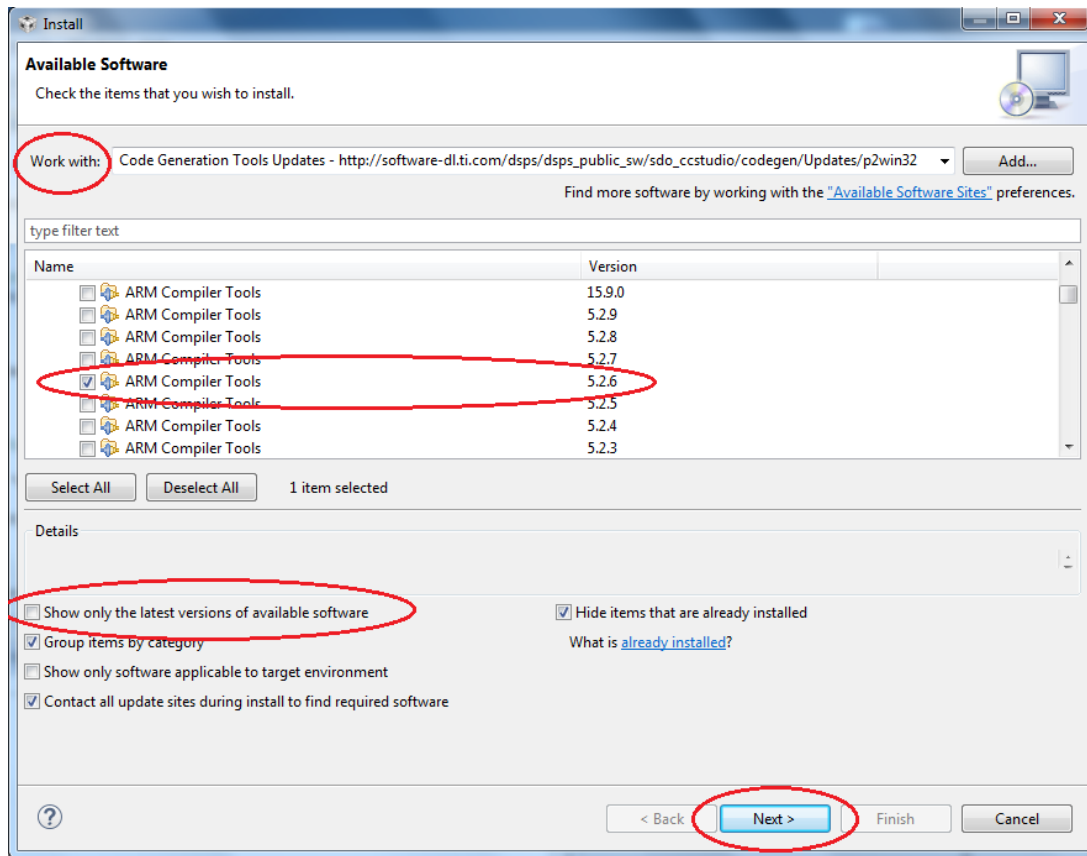


Figure 23, Install Available Software Page.

- c. In the bottom left hand corner, uncheck the "Show only the latest versions of available software" box.
- d. In "Work with" field, type:

[http://software-](http://software-dl.ti.com/dsps/dsps_public_sw/sdo_ccstudio/codegen/Updates/p2win32/)

[dl.ti.com/dsps/dsps\\_public\\_sw/sdo\\_ccstudio/codegen/Updates/p2win32/](http://software-dl.ti.com/dsps/dsps_public_sw/sdo_ccstudio/codegen/Updates/p2win32/)

- e. Under TI Compiler Updates, select TI ARM Compiler Tools Version 5.2.6. Click Next. (Figure 23)
- f. The Review Licenses screen should appear. Accept and Click Finish.
- g. Restart CCS to finish installation.
- 6) Install the Debugger Dev Pack interface.
  - a. Close out of CCS if it is open.
  - b. Plug the Debugger Dev Pack into a USB port on your computer Via a B-Male Micro USB to an A-Male USB cable and wait for the driver to finish installing. (Figure 24)

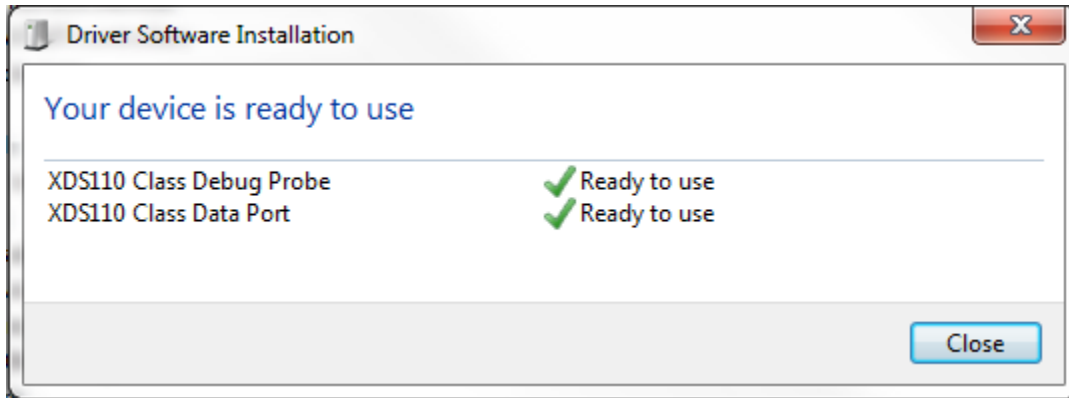


Figure 24, Debugger Driver Installation.

- 7) Install TIRTOS v2.11.01.9
  - a. Open the provided software folder.
  - b. Run the `tirtos_simplelink_setupwin32_2_11_01_09` executable.
  - c. After the setup wizard loads, click Next (Figure 25)

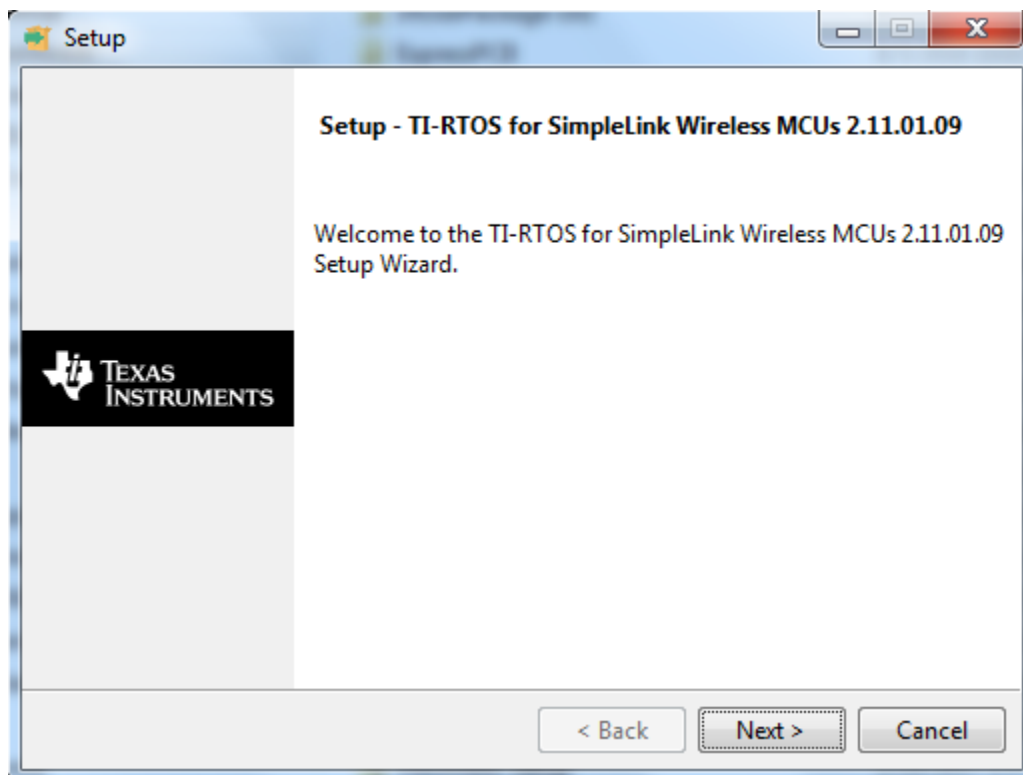


Figure 25, TI-RTOS 2.11.01.09 setup wizard.



- d. Accept the license agreement. Click next
- e. Select the proper home directory (default is C:\ti). Click Next
- f. After Installation Click Finish.
- g. Open CCS or close and restart it if it is already open.
- h. CCS should detect new installable products (Figure 26). Click Install.

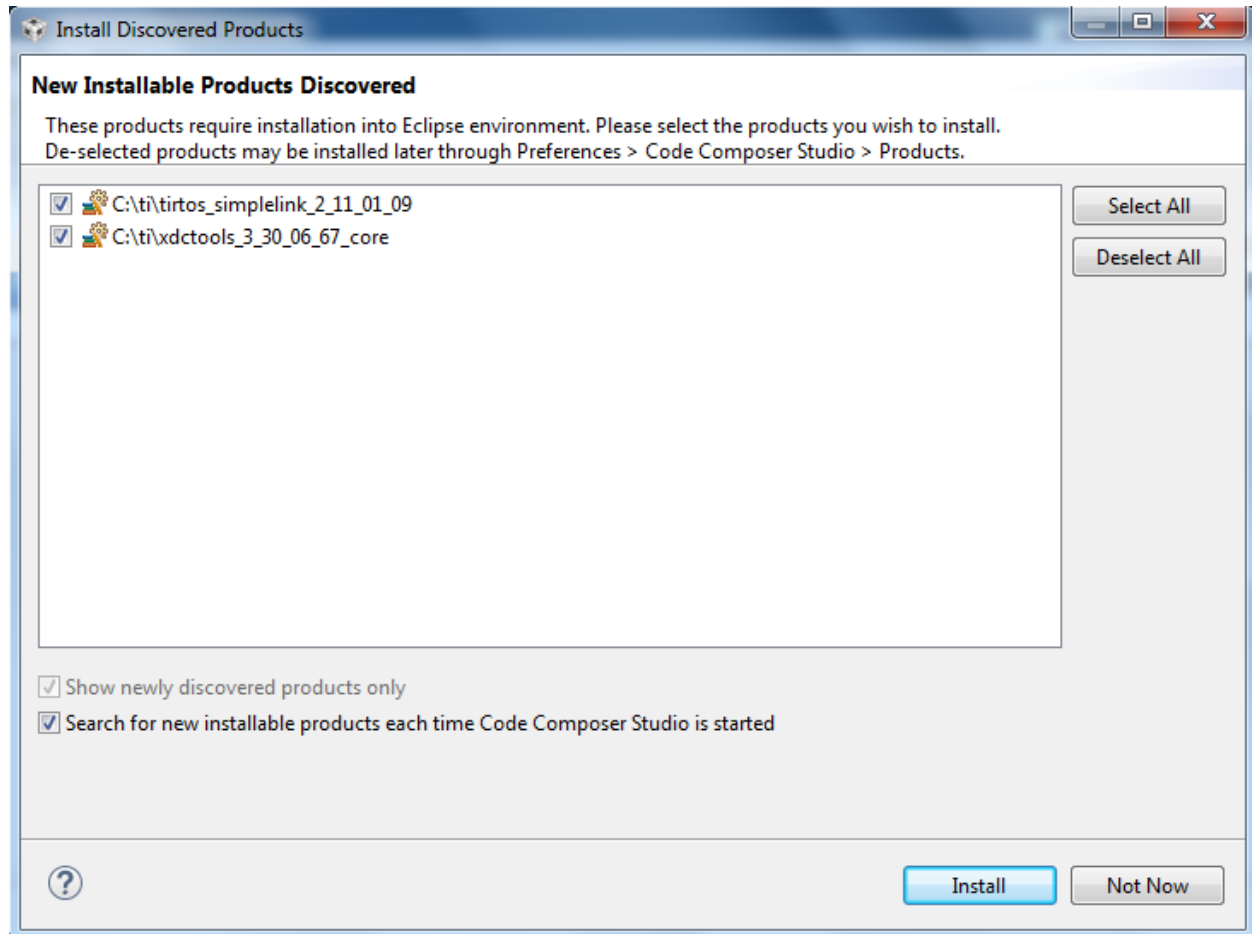


Figure 26, Install Discovered Products.

- 8) Import the DEV-IN-BLE-TI CCS app and stack projects.
  - a. Open CCS.
  - b. Make sure that the "CCS Edit" button in the upper right corner is pressed.
  - c. In the Project menu tab click "Import CCS projects..." (Figure 27).

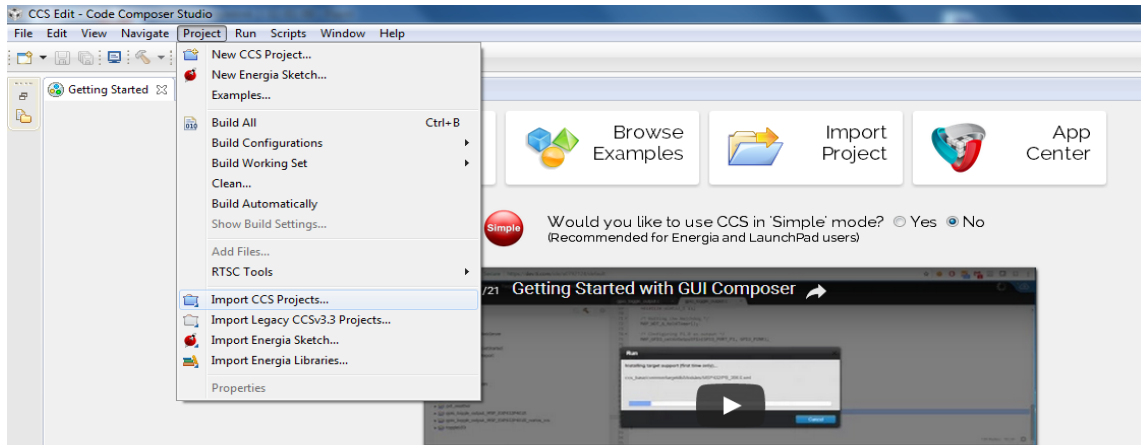


Figure 27, Import CCS Project tab.

- d. The Select CCS Project to Import page should now be open. (Figure 28)

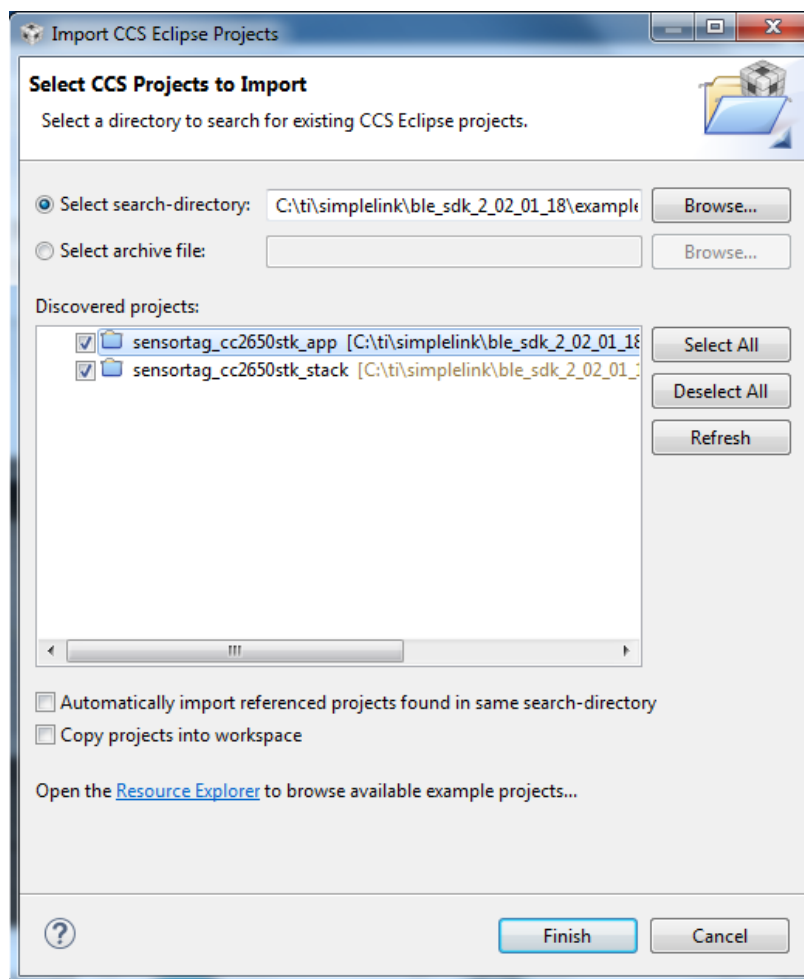


Figure 28, Select CCS Projects to import

- e. Under "Select search-directory", Click Browse
- f. Navigate to <TI home directory (default:  
C:\ti)>\simplelink\ble\_sdk\_2\_02\_01\_18\examples\cc2650stk\sensortag\ and  
select the ccs folder (Figure 29). Click OK

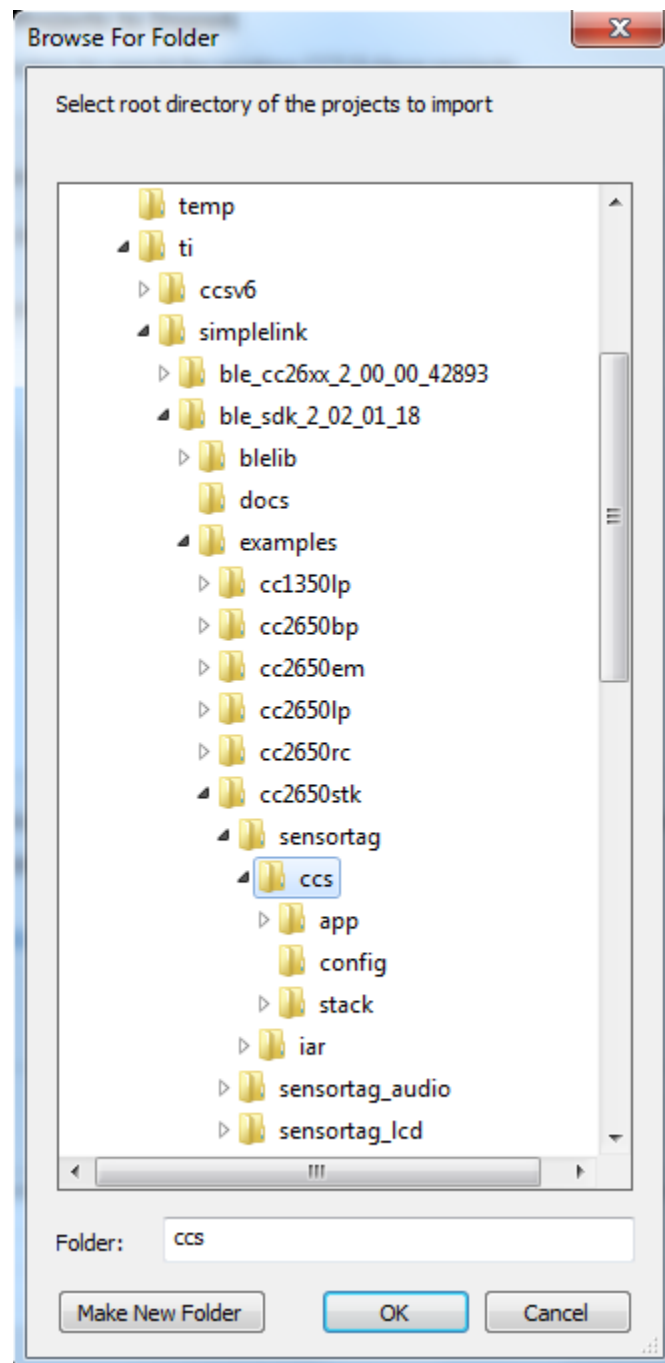


Figure 29, DEV-IN-BLE-TI Project Location

- g. The sensortag\_cc2650stk\_app and sensortagcc2650stk\_stack projects should appear in the Discovered projects box (Figure 28).
  - h. Check the box for each project (Figure 28).
  - i. Make sure that "copy projects into workspace" box is unchecked (Figure 28).
  - j. "Automatically import referenced projects found in same search-directory" box can be checked or unchecked (Figure 28).
  - k. Click Finish.
- 9) Import the DEV-IN-BLE-TI CCS bim\_extflash project.
- a. Return to the Import CCS Projects page by clicking Projects – Import CCS Projects...
  - b. Under "Select search-directory:", navigate to <TI home directory (default: C:\ti)>\simplelink\ble\_sdk\_2\_02\_01\_18\examples\util\bim\_extflash\cc2640\ and select the ccs folder. Click OK
  - c. The bim\_extflash project should appear in the Discovered projects: box.
  - d. Check the box next to the bim\_extflash project.
  - e. Make sure the "Copy projects into workspace" box is unchecked.
  - f. The "Automatically import reference projects found in same search-directory" box can be checked or unchecked.
  - g. Click Finish. The bim\_extflash project should appear in the Project Explorer along with the sensortag\_cc2650stk\_app and sensortag\_cc2650stk\_stack projects. (Figure 30).

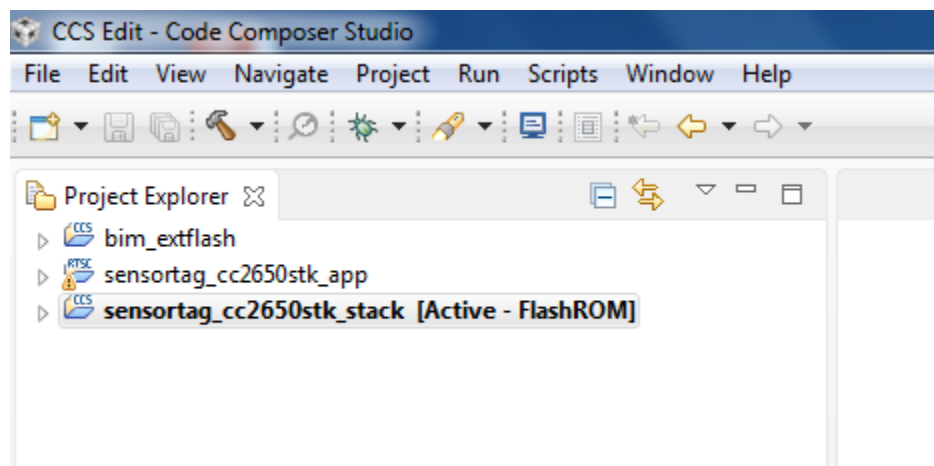


Figure 30, Project Explorer with imported projects.

See instructions below for build and download/debug instructions.

## Build and Download/Debug

The build and download process is more involved due to the multi project structure. The build configurations of each project depend on each other so the order in which projects are built is important. Some projects also share memory segments so the order of download is also important. For more information on the build and debug process see the CC2640 and CC2650 SimpleLink Bluetooth Low Energy Software Stack 2.2.1 Developer's Guide sections 2.6. (<http://www.ti.com/lit/ug/swru393d/swru393d.pdf> )

Follow these steps to build, download and debug code on the DEV-IN-BLE-TI.

- 1) Clean all projects.
  - a. Under Projects menu tab, Click Clean. (Figure 31).

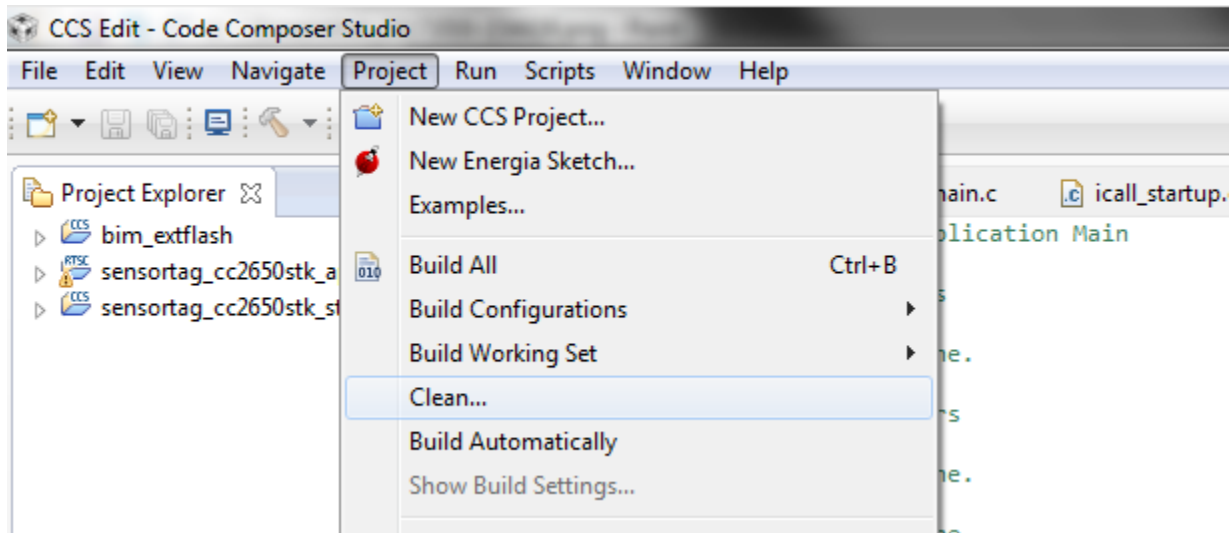


Figure 31, Project - Clean...

- b. The Clean form should appear (Figure 32).

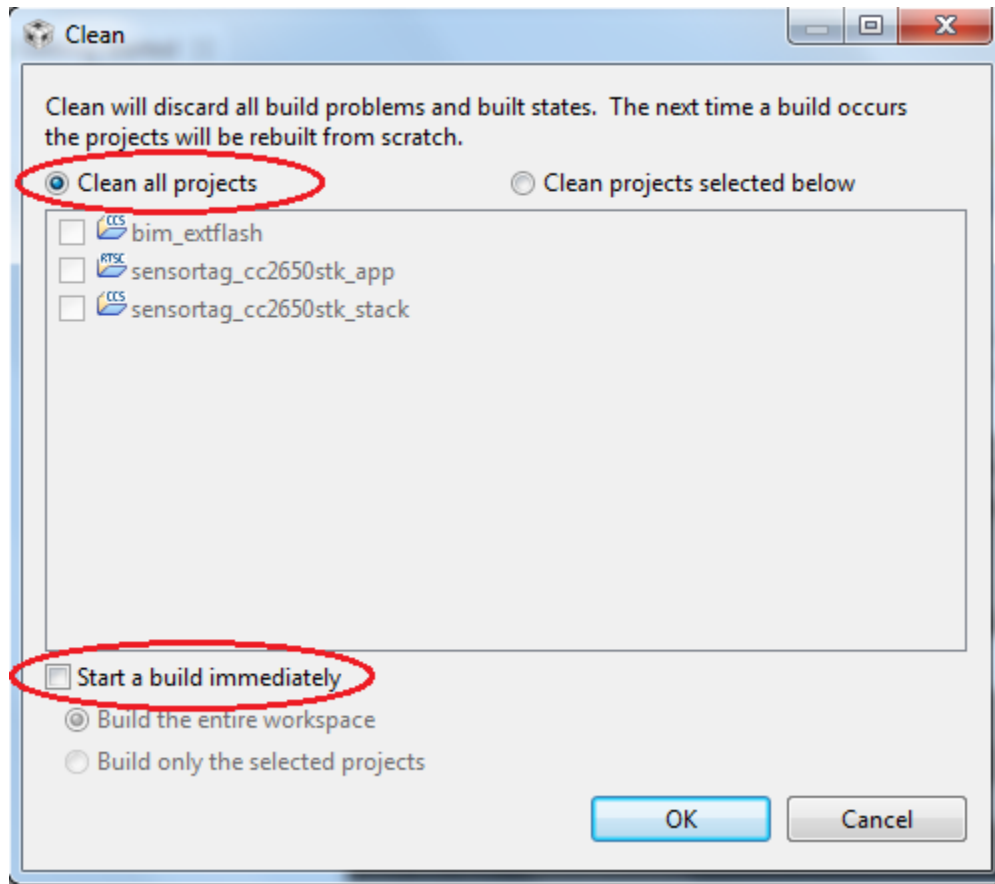


Figure 32, Clean Projects Screen.

- c. At the top right of the form select the "Clean all projects" radio button.
  - d. UNCHECK the "Start a build immediately" box at the bottom right of the form.
  - e. Click OK. Wait for the cleaning process to finish.
- 2) Build bim\_extflash project.
- a. Right Click the bim\_extflash project in the Project Explorer (Figure 33).

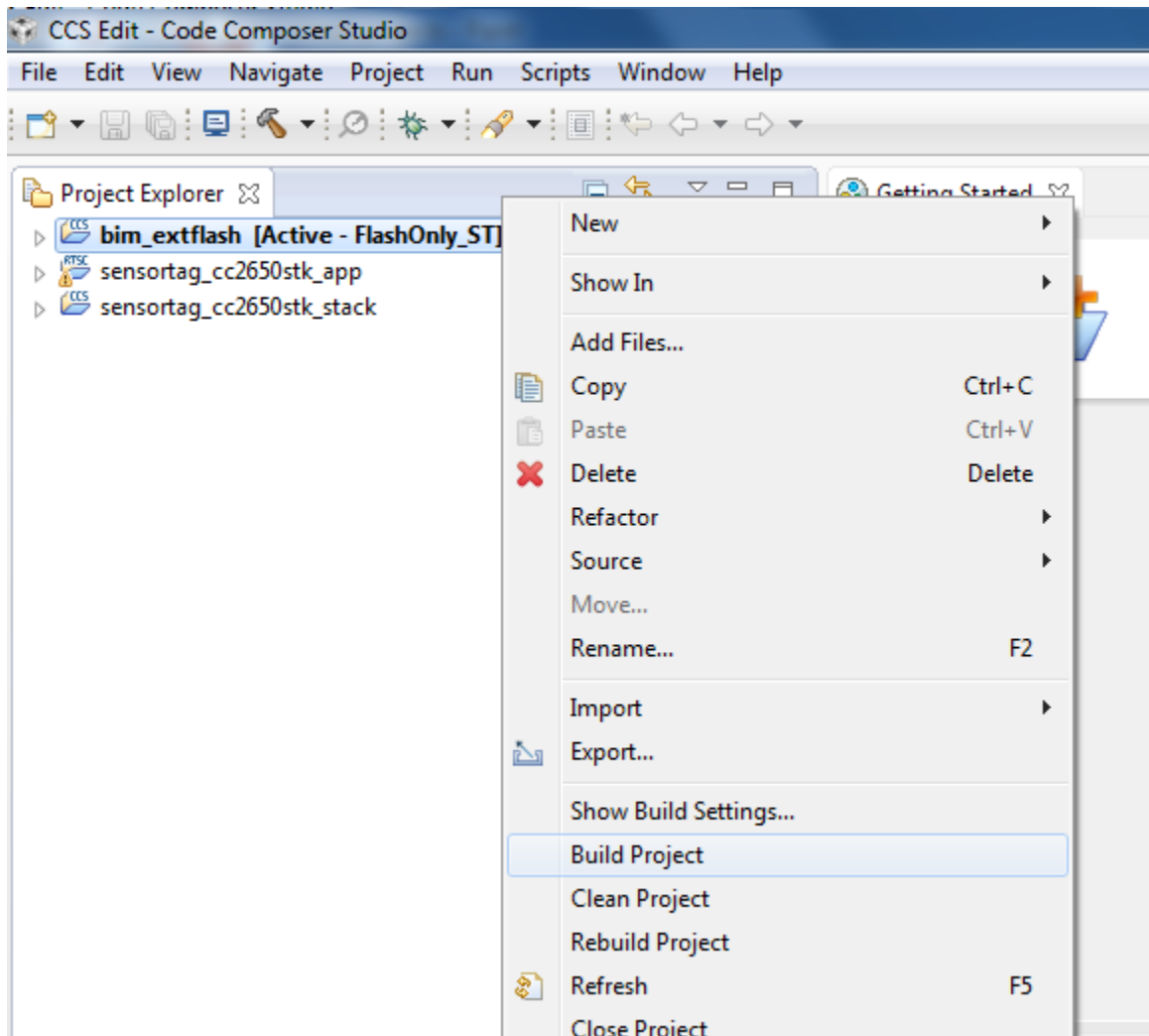


Figure 33, Right Click - Build Project.

- b. Select "Build Project". Wait for the build process to finish.
- 3) Repeat the build process (Step 2) for the sensortag\_cc2650stk\_stack project
- 4) Repeat the build process (Step 2) for the sensortag\_cc2650stk\_app project
- 5) Use a microUSB cable to connect the Debugger Dev Pack to a USB port on your computer. Wait for the driver software to finish installing.
- 6) Attach the Debugger Dev Pack to the DEV-IN-BLE-TI using the Dev Pack Connectors P1 and P2. (Figure 34)



Figure 34, Circuit Board Dev Pack Connector.

- 7) Download/Debug the bim\_extflash project. (Figure 35)
  - a. Right Click the bim\_extflash project.
  - b. Select Debug As – 1 Code Composer Debug Session.
  - c. Wait for the program to load.

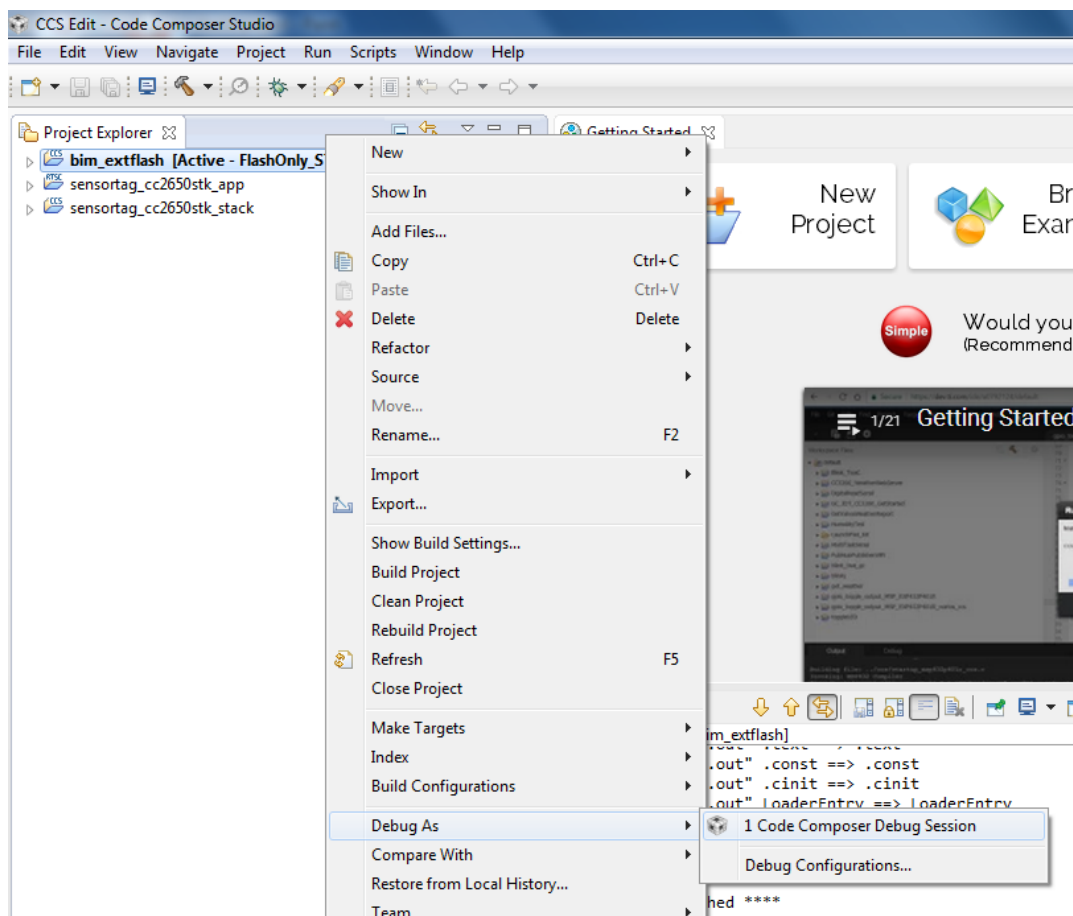


Figure 35, Download/Debug.



- d. Click the red stop square on the toolbar to stop the debug session. (Figure 36)

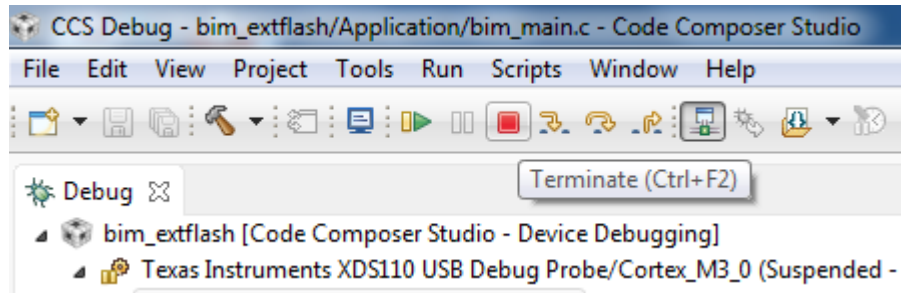


Figure 36, Terminate Debug Session.

- 8) Repeat the Download/Debug process (Step 7) for the sensortag\_cc2650stk\_stack project.
- 9) Repeat the Download/Debug process (Step 7) for the sensortag\_cc2650stk\_app
  - a. To debug the application, click the green run arrow instead of the red stop square after program loads.

## Overview of CC2650 Code Structure

The code structure of the DEV-IN-BLE-TI is broken up into three separate projects; bim\_extflash, sensortag\_cc2650stk\_stack, and sensortag\_cc2650stk\_app. These projects are loaded into separate segments of flash memory and work together to produce the final functionality of the device.

The bim\_extflash project is the Boot Image Manager which handles Over Air Downloads. This gives users the ability to update device software through the BLE interface. For more information on using the BIM refer to the CC2640 BLE OAD User's Guide.

[https://e2e.ti.com/cfs-file/\\_key/communityserver-discussions-components-files/538/CC2640-BLE-OAD-User\\_2700\\_s-Guide.pdf](https://e2e.ti.com/cfs-file/_key/communityserver-discussions-components-files/538/CC2640-BLE-OAD-User_2700_s-Guide.pdf)).

The sensortag\_cc2650stk\_stack contains the BLE software stack and is responsible for implementing the Bluetooth Low Energy protocol. For details on the BLE stack see the CC2640 and CC2650 SimpleLink Bluetooth Low Energy Software Stack 2.2.1 Developer's Guide.

The `sensortag_cc2650stk_app` contains the application. This includes the main application loops and threads, Bluetooth profiles, startup procedures, hardware interfaces, and the interfacing the BLE stack.

The majority of customization will only affect the `sensortag_cc2650stk_app` project. Adding sensors or profiles can be done by modifying the application project (See Adding a Custom Service or Profile).

More information, examples, and documentation for TI SimpleLink TIRTOS can be found in the TI Resource Explorer found at <http://dev.ti.com/tirex/#/>

Particularly helpful is the SimpleLink™ Academy 1.14.02 for SimpleLink CC2640R2 SDK 1.50 found under Software – SimpleLink CC2640R2 SDK -v:1.50.00.58 – SimpleLink Academy

It provides thorough explanation of the TI BLE stack, TIRTOS operating system, interfacing with sensor drivers, and examples.

### Adding a Custom Service or Profile

New sensors or functionality added to the DEV-IN-BLE-TI will have to be integrated into the application code structure.

### Profile Setup

A tutorial on how to set up a custom profile in a TI BLE application can be found in the TI Resource Explorer under Software – SimpleLink CC2640R2 SDK -v:1.50.00.58 – SimpleLink Academy – Lab Overview – Bluetooth 4.2 – Custom Profile.

### External Sensor/Device Setup

A tutorial on setting up sensors and an example setting up a UART connection can be found in the TI Resource Explorer under Software – SimpleLink CC2640R2 SDK -v:1.50.00.58 – SimpleLink Academy – Lab Overview – TI Drivers – Configuring TI Drivers.

### Initialization

Once the sensor is configured and the profile is set up the last step is to initialize the profile in the application code. This can be done in the `sensortag_init()` function located in the `sensortag.c` file.

Figure 37 shows where an example initialization function, `SensorCustom_init()`, should be placed. The initialization function should include required components outlined by the profile setup section.

```
496 // Add device info service.  
497 DevInfo_AddService();  
498  
499 // Add application specific device information  
500 SensorTag_setDeviceInfo();  
501  
502 // Add battery monitor  
503 SensorTagBatt_init();  
504  
505 /*  
506 SensorCustom_init();  
507 */  
508  
509 // Power on self-test for sensors, flash and DevPack
```

Figure 37, Location in code to initialize a custom service.

## Reprogramming with Hex Files

Device firmware can be compiled into HEX files and flashed onto the CC2650 chip using TI's SmartRF Flash Programming tool and a compatible programmer. For detailed information on compiling and using HEX files see the CC2640 and CC2650 SimpleLink Bluetooth Low Energy Software Stack 2.2.1 Developer's Guide section 2.7 and the CC2640 BLE OAD User's Guide.

### Flashing HEX Files to the DEV-IN-BLE-TI CC2650 IC

Follow the instruction below to reprogram the DEV-IN-BLE-TI Dev Kit to its factory default program. Instructions assume user has access to the TI Debugger Dev Pack or equivalent programmer. Instructions are based on Windows 10 OS.

- 1) Install the latest version of TI's FLASH-PROGRAMMER-2 PC application:  
<http://www.ti.com/tool/FLASH-PROGRAMMER>
- 2) Follow standard installation instructions.
- 3) When finished, open application to the 'Main' tab (Figure 40).

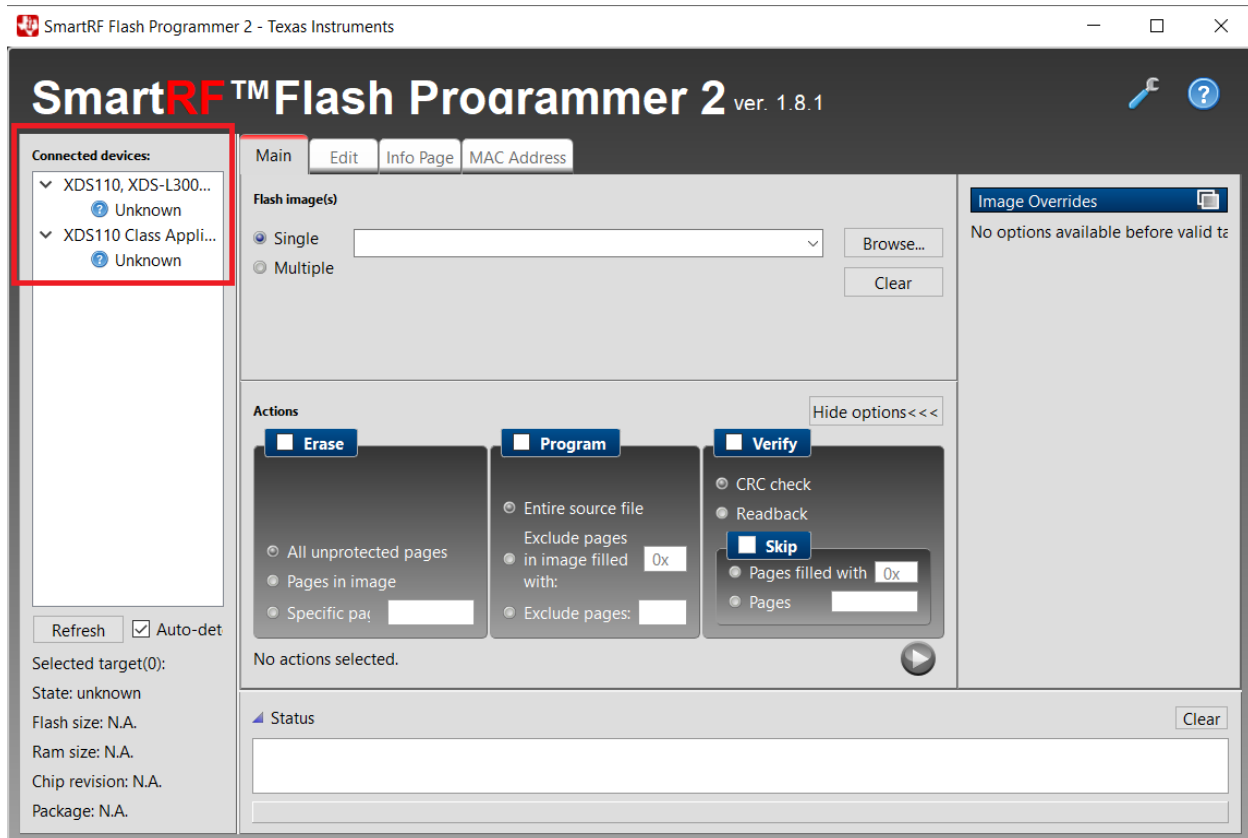


Figure 38, SmartRF Flash Programmer 2 PC Application

- 4) Connect the Debugger Dev pack to a USB port on the PC. It should appear in the connected device display.
- 5) Connect the Debugger Dev pack to the DEV-IN-BLE-TI PCB. The connected devices list should update to read “CC2650” (Figure 39)
- 6) Click on “CC2650” in the connected devices list to select it.

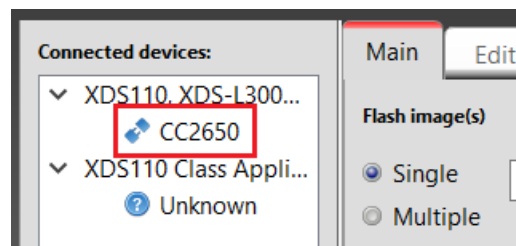


Figure 39, DEV-IN-BLE-TI device connected to the SmartRF Flash Programmer 2

- 7) On the “Main” tab, select Single and click Browse.
- 8) Navigate to the DEV-IN-BLE-TI Software folder in the provided Software and Resources Package. Select sensortag\_cc2650stk\_all.hex (Figure 40)

- 9) In the Actions list check Program. (Figure 40)
- 10) Click the Play button to Flash the code on the CC2650 IC, when complete the progress bar should turn green and read Success!
- 11) If failure occurs, you may need to erase the Flash and try again. Click the wrench in the top right corner and Select CC26xx/CC13xx Forced Mass Erase.

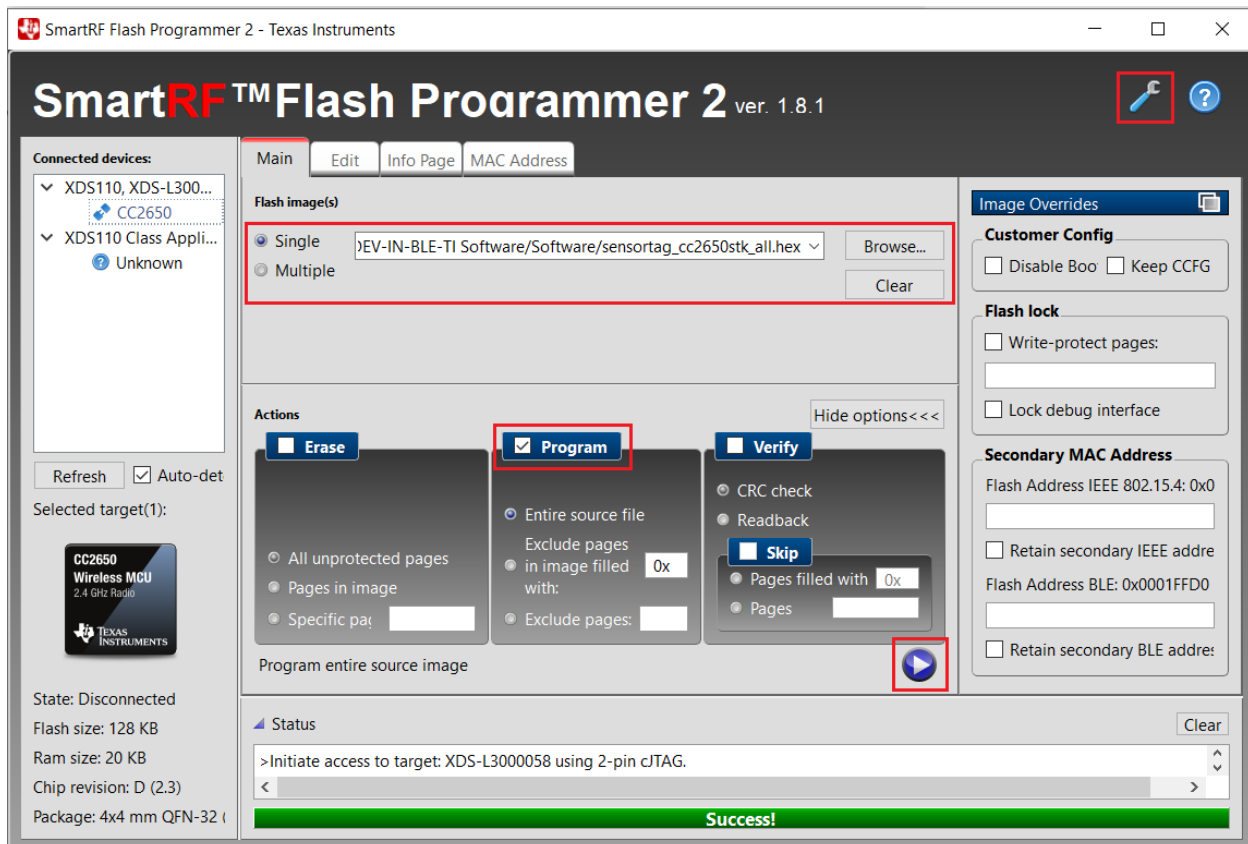


Figure 40, SmartRF Flash Programmer 2 Successful Operation.

After successful flashing, the DEV-IN-BLE-TI kit will now operate in its default original state.

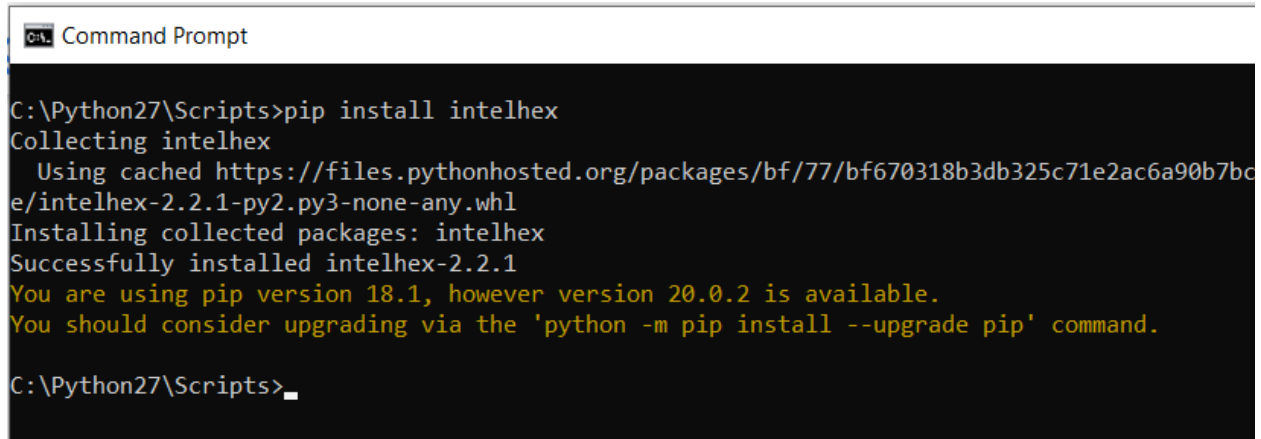
## Merging Compiled DEV-IN-BLE-TI Project Into a Single Hex File

The DEV-IN-BLE-TI firmware consists of three components; the application, the BLE stack and Over-Air-Download. Code Composer Studio compiles each of these projects separately and is configured to output 3 separate Hex code files.

Follow the instructions below to combine the 3 project hex files into a single file able to be flashed on the CC2650 IC, as described in the previous section.

- 1) Download and install python 2.7.0 from <https://www.python.org/download/releases/2.7/>

- 2) Open Command Prompt on your PC and Navigate to the C:\Python27\Scripts. (Figure 41)
- 3) Type “pip install intelhex” and execute. (Figure 41)
- 4) Make sure output reads “successfully installed intelhex-2.2.1” or equivalent version.

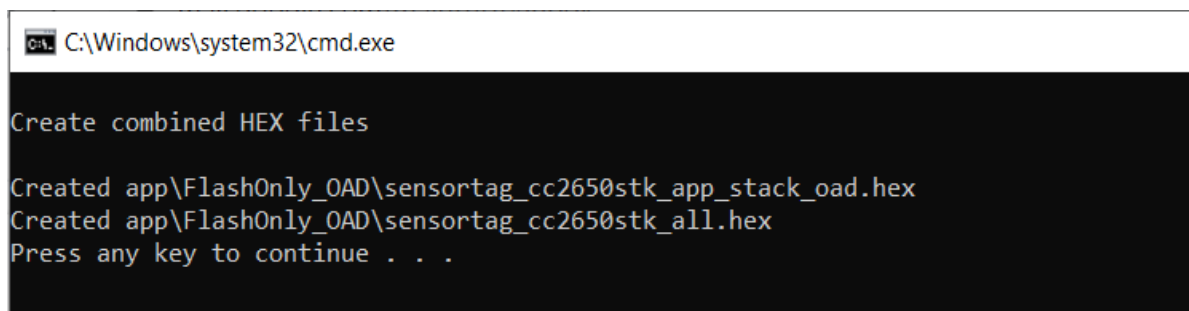


```
C:\Python27\Scripts>pip install intelhex
Collecting intelhex
  Using cached https://files.pythonhosted.org/packages/bf/77/bf670318b3db325c71e2ac6a90b7bc
e/intelhex-2.2.1-py2.py3-none-any.whl
Installing collected packages: intelhex
Successfully installed intelhex-2.2.1
You are using pip version 18.1, however version 20.0.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Python27\Scripts>
```

Figure 41, Installing intelhex python library.

- 5) In file explorer, open the ti home directory (Default C:\ti).
- 6) Navigate to  
...\ti\simplelink\ble\_sdk\_2\_02\_01\_18\examples\cc2650stk\sensortag\ccs\app and  
double click the merge.bat file.



```
C:\Windows\system32\cmd.exe

Create combined HEX files

Created app\FlashOnly_OAD\sensortag_cc2650stk_app_stack_oad.hex
Created app\FlashOnly_OAD\sensortag_cc2650stk_all.hex
Press any key to continue . . .
```

Figure 42, execution of the merge.bat batch file.

The 3 project components have now been successfully merged into a single hex file, sensortag\_cc2650stk\_all.hex. This file is located in the app\FlashOnly\_OAD folder of the ccs project directory. It can be flashed to the CC2650 IC following the instructions provided in the previous section.

## Android Studio

### Installation Setup

- 1) Install Android Studio.
  - a. Download the latest version of Android Studio from <https://developer.android.com/studio/index.html>
  - b. Run the installation executable after it has finished downloading.
  - c. The Android Studio Setup welcome screen should appear (Figure 38). Click Next.

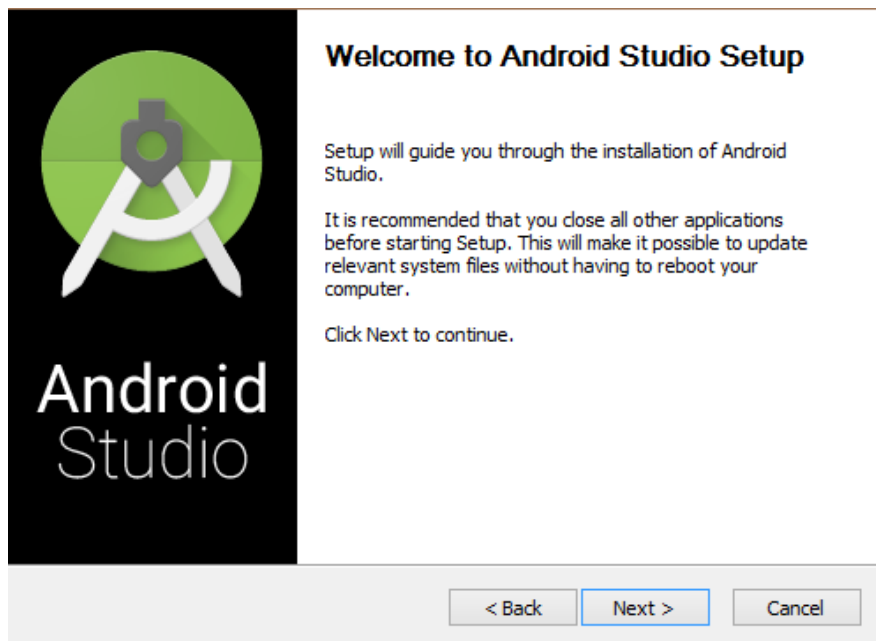


Figure 43, Android Studio Installation Wizard Welcome Screen.

- d. For "Choose Components", check all boxes (Figure 43). Click Next.

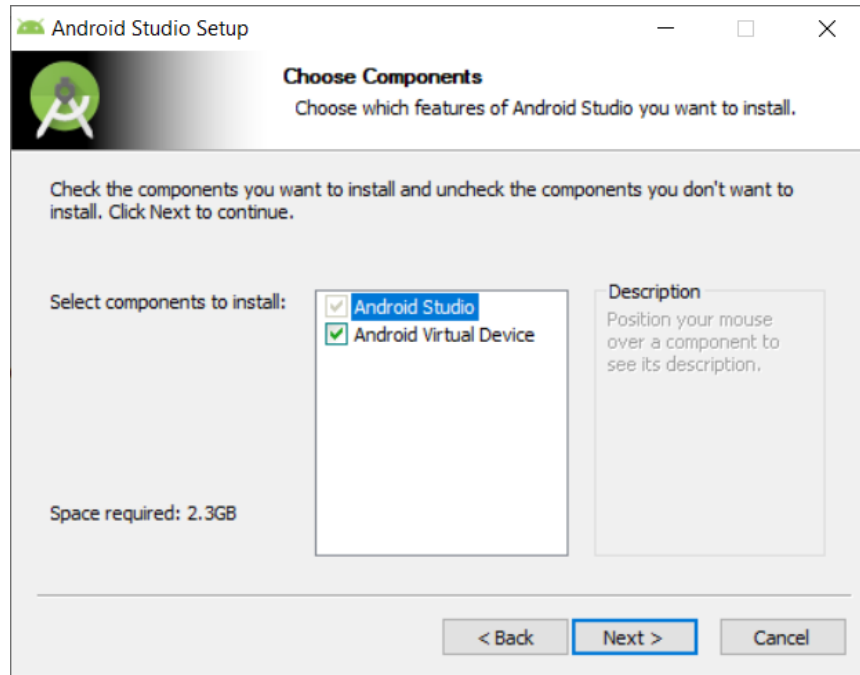


Figure 44, Choose Android Studio Components.

- e. For "License Agreement", Click I Agree.
- f. For "Configuration Settings", Choose a desired Android installation location or use the default locations. Click Next.
- g. Select desired Start Menu Folder location. Click Install.
- h. On the "Android Studio Setup" screen, check the "Start Android Studio" box (Figure 45). Click Finish.



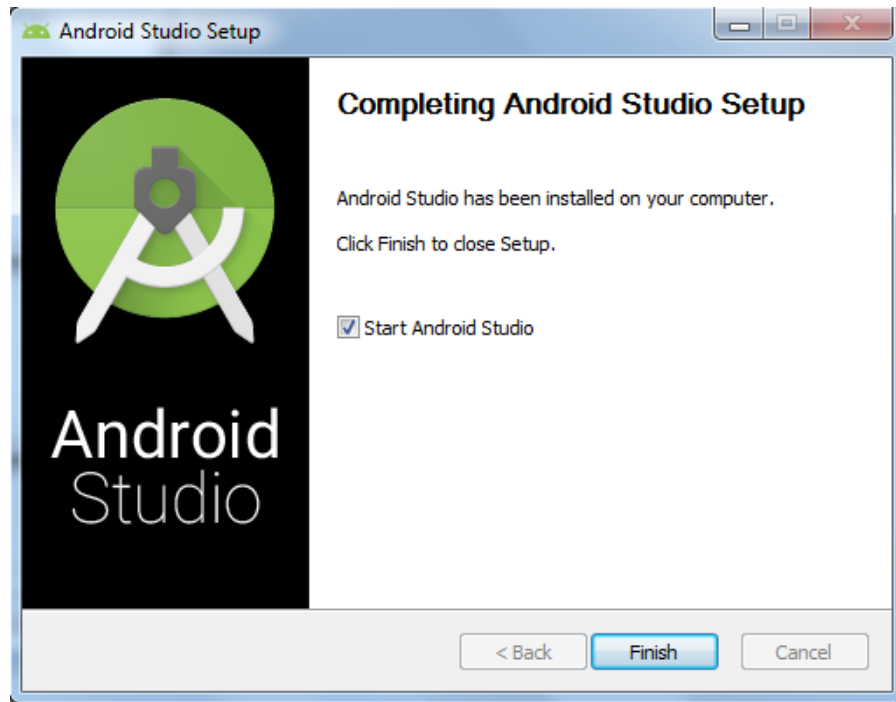


Figure 45, Android Studio Installation Completion Screen.

- i. The first time running Android Studio, it will go through the preferences setup. Choose standard settings as desired or configure from an installation folder.
- 2) Import the PFDEV-TI Data Monitor project.
  - a. Open Provided DEV-IN-BLE-TI Software folder.
  - b. Extract the sensortag-20-android project to desired working directory on your local machine.
  - c. Open Android Studio if it is not already open.
  - d. Close out of any open projects and navigate to the Android Studio welcome screen (Figure 46).

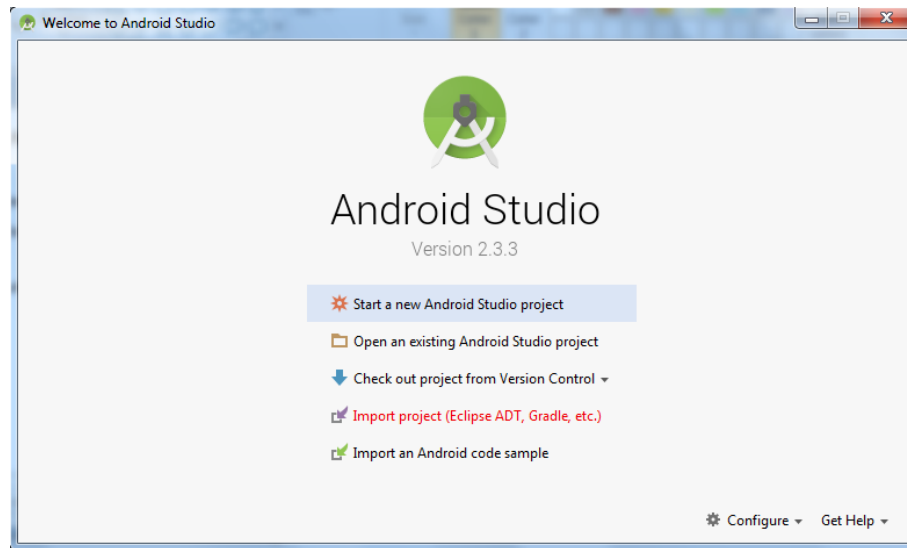


Figure 46, Android Studio Home Screen.

- e. Select "Open an existing Android Studio project".
- f. Browse to the local working directory where the sensortag-20-android project was extracted too.
- g. Expand sensortag-20-android folder and select the sensortag-20-android project (Figure 47). Click OK.

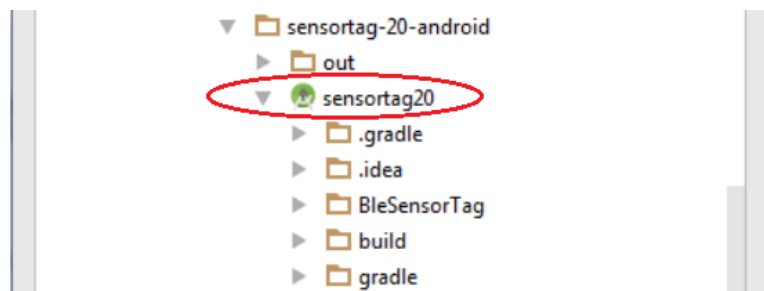


Figure 47, sensortag project.

- h. Android may prompt to change the Android SDK path to reflect your local machine (Figure 48), Click OK.



Figure 48, Sync Android SDKs

- i. Allow project to build and sync. Ignore “obsolete” warning.
- j. If SDK is not present, sync may fail. Install missing SDK package and resync (Figure 49)

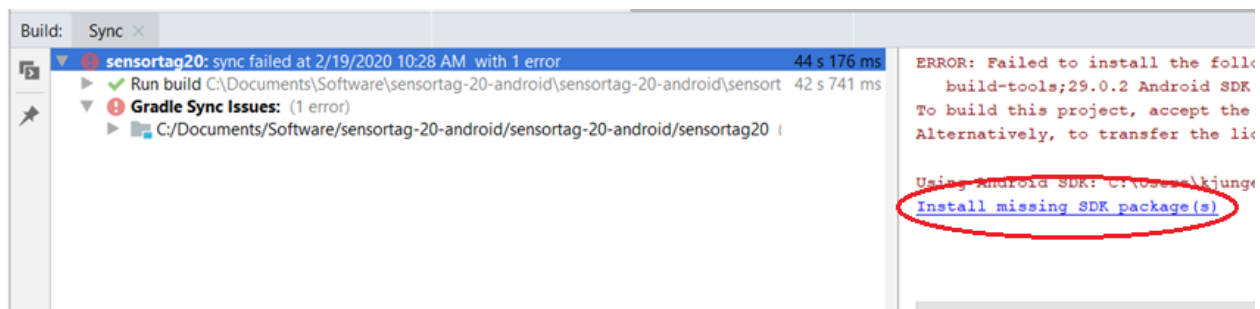


Figure 49, Sync Android SDKs

- 3) Install USB ADB driver for the specific Android device you will be using for development. As an example, the set-up process for a Kindle Fire Tablet will be shown.
  - a. Using the Windows Device manager, uninstall any drivers the computer has installed for the Kindle Fire (Figure 50).

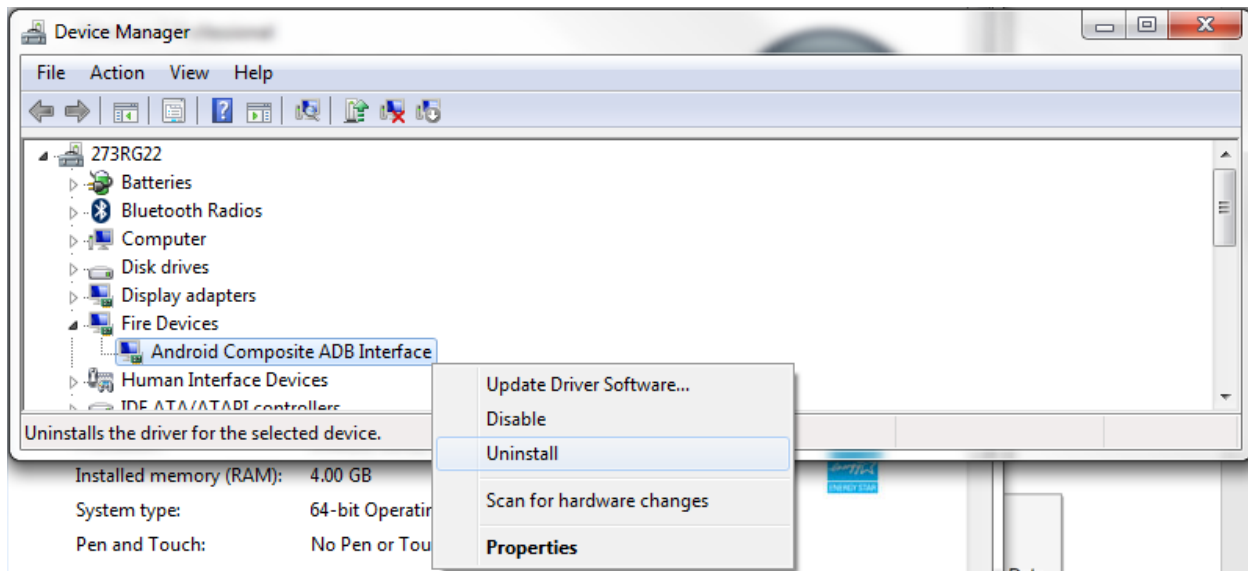


Figure 50, Uninstall Any Drivers present for your Android Device.

- b. Make sure Android Device is not plugged into your computer.
- c. Enable USB Debugging on your Android device. Android device developer settings are hidden by the manufacturer and need to be enabled.
  - i. To enable Developer settings.
    - 1. Open Settings – About. (or find where the build number is located on your device)
    - 2. Click Build number 7 times. (For Kindle Fire, Serial Number is tapped 7 times instead of build number and is located in Settings – Device Options)
    - 3. After the 7<sup>th</sup> tap, the developer menu should appear in the settings menu.
    - 4. Although the procedure should be the same for all android devices, the “Build number” setting may have a different name and be located in different places for different devices.
  - ii. In Developer settings, make sure “Enable ADB” or “USB debugging” are enabled.
- d. Obtain the USB ADB driver for you specific Android Device. The following webpage provides instructions on installing OEM USB Drivers for a variety of Android device manufacturers.  
<https://developer.android.com/studio/run/oem-usb.html>

- i. For Kindle Fire, USB ADB driver can be downloaded from this URL;  
[https://s3.amazonaws.com/android-sdk-manager/redist/kindle\\_fire\\_usb\\_driver.zip](https://s3.amazonaws.com/android-sdk-manager/redist/kindle_fire_usb_driver.zip)
- e. Follow the manufactures instructions on installing the ADB driver for your android device.
  - i. For Kindle Fire, a setup executable is provided.
    1. After running the executable and following the installation prompts, the driver is ready to use.
    2. After connecting the Kindle Fire to the computer via USB it was able to successfully recognize and install the correct driver for the Kindle Fire (Figure 51).

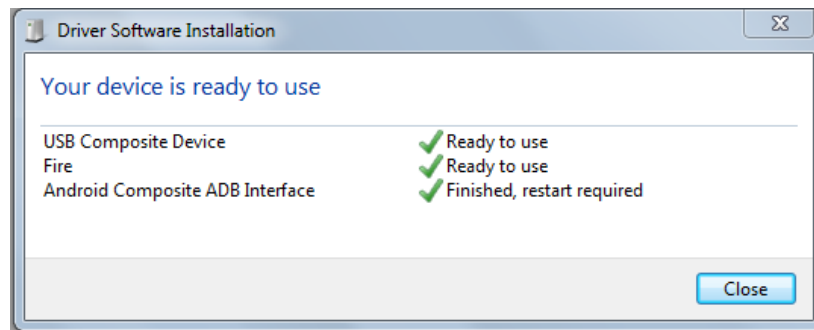


Figure 51, Kindle Fire Driver Installation.

- f. Connect to Android Device in Android Studio
  - i. Unplug Android device from computer.
  - ii. Open Android Studio.
  - iii. When no devices are present the device toolbar should read "No Devices" (Figure 52)

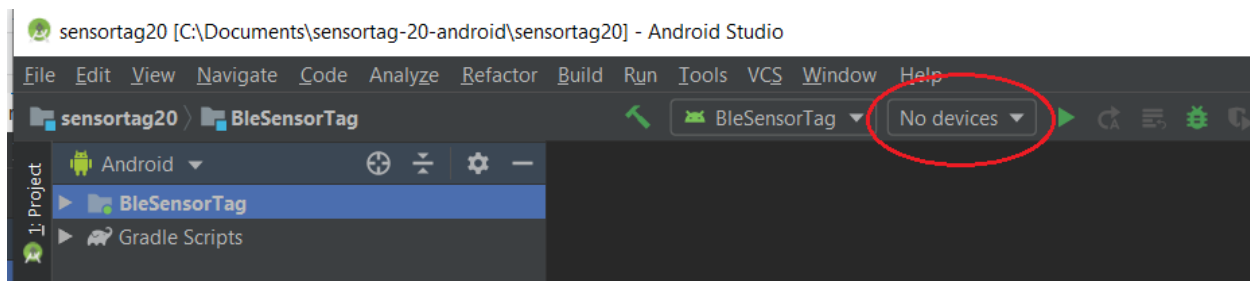


Figure 52, Device Toolbar.

- iv. Plug Android device into computer and popup to enable USB debugging should appear on your device (Figure 53). Click OK.

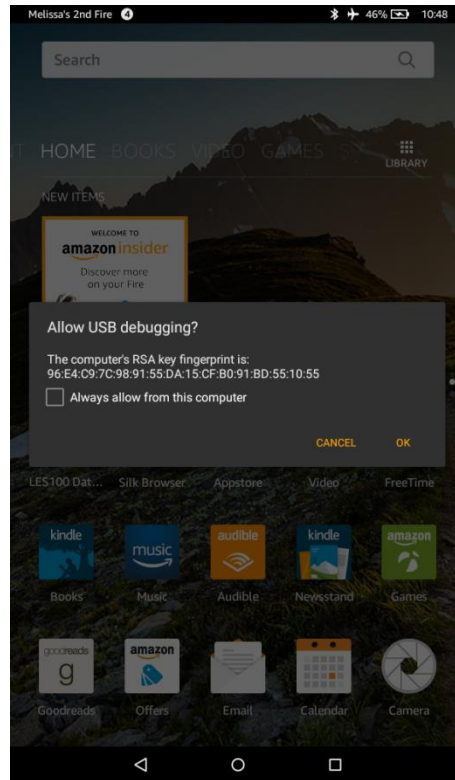


Figure 53, Allow USB Debugging on Android Device.

- g. Android should now detect your device (Figure 54).

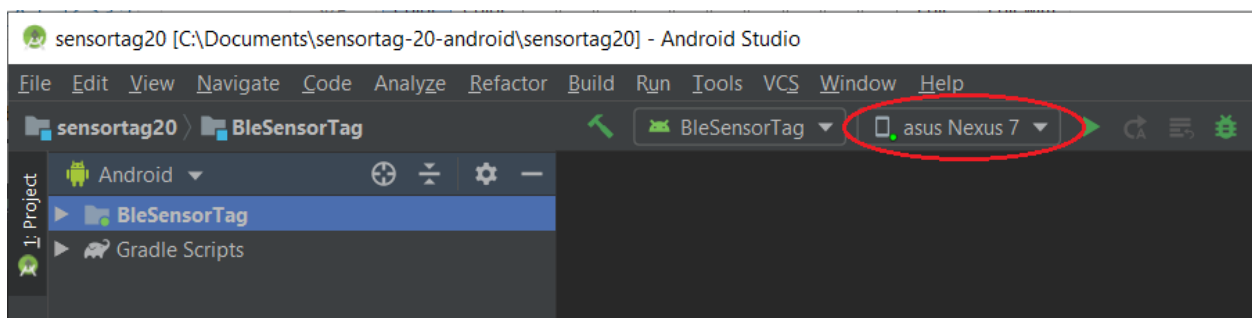


Figure 54, Allow USB Debugging on Android Device.

## Build and Run/Debug PFDEV-TI Data Monitor

The following steps will outline how to build and debug the DEV-IN-BLE-TI project on a connected Android device. Assuming you have already gone through the Android Installation and setup procedure.

- 1) Open the DEV-IN-BLE-TI Android Studio project.
- 2) Connect your Android device to the computer via USB. (follow instructions in previous section to connect device to Android Studio)
- 3) To build the project, select Build on the top menu bar and click Make Project (Figure 55)

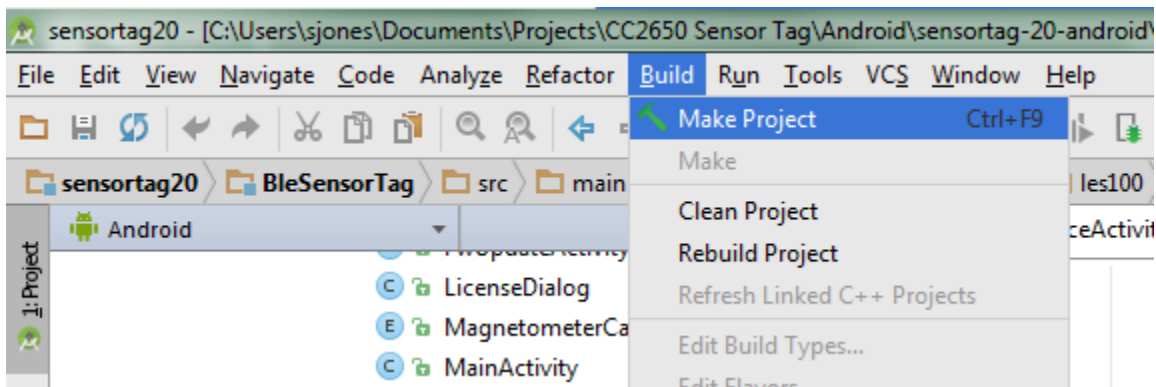


Figure 55, Make Project.

- 4) To download and run/debug the project on your Android device, select Run on the top menu bar and click Run 'BleSensorTag' or Debug 'BleSensorTag'. (Figure 56).

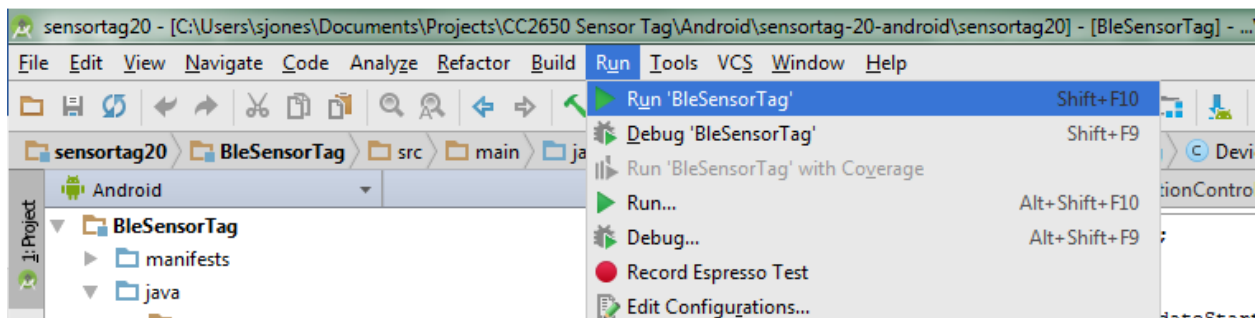


Figure 56, Run Program.

- 5) If a deployment target has not been set, a prompt will ask you to select a target device. Choose your connected Android device. Click OK.

NOTE: Bluetooth features are incompatible with Virtual devices, therefore the PFDEV-TI Data Monitor can only be run/debugged on real Android devices.

## Overview of Android Code Structure

### Structure

The PFDEV-TI Data Monitor utilizes both java and xml to create the functionality and User Interface (UI) of the application. In general, java activities handle the flow of the application while xml files are used to define the various layouts, constants and other UI components.

Apart from the build files, the project code is separated into the java, assets, and res (resource) folders. (located at \sensortag20\BLESensorTag\src\)

The java folder (located at

\sensortag20\BLESensorTag\src\main\java\com\powerfilm\les100\ble\sensortag) contains the main application activities which includes the MainActivity.java, DeviceActivity.java and all of the Bluetooth profiles and row java files.

### MainActivity.java

The MainActivity.java code is run at the start of the application and is responsible for handling user interaction with the home screen. It handles scanning for Bluetooth devices and displaying devices found to the user.

### DeviceActivity.java

The DeviceActivity.java code runs when the main activity selects a device to connect too. This activity handles the Bluetooth connection, data transfer and interaction with the device home page.

After discovering all Profiles and Services of the connected device, the DeviceActivity initializes a row for each one. If new Bluetooth profiles or service are added to the application, this activity will need to be modified to include those as well.

### Bluetooth Profiles and Rows

Each Bluetooth Profile or Service has a java activity associated with it. This code is run after the profile or service is discovered. These activities handle data specific to their functionality.



The PFDEV-TI Data Monitor Profiles and services functionality are outlined in the section “PFDEV-TI Data Monitor Notes”.

Rows are defined by the GenericCharacteristicTableRow activity and display data and information on the UI. If extra features are required, profiles may have their own specific row activity which is an extension of the GenericCharacteristicTableRow. For example, the Battery Level Profile needs extra features to define the Max and Min Battery levels. These features are created in the BatteryTableRow activity.

### Layouts

The UI is defined in the xml files located in the layout section of the res folder. The xml files define the UI elements for the various pages and screens throughout the application.

### Other

The res folder also contains xml files with string constants, gatt\_uuids, and preferences.

The Android Manifest file located in the root directory somewhat ties all application components together. This is where the project launch location is stored, application permissions, and other critical settings are.

Android Studio projects use the Gradle build system to save build settings and configurations.

### Adding a Customized Service or Profile

After adding a new sensor along with a service or profile to the DEV-IN-BLE-TI device firmware, the PFDEV-TI Data Monitor must also be updated to be able to discover and display its data.

For general information about android BLE development and SensorTag specific Android development see the SensorTag wiki guide at

[http://processors.wiki.ti.com/index.php/SensorTag\\_User\\_Guide#SensorTag\\_Android\\_Development](http://processors.wiki.ti.com/index.php/SensorTag_User_Guide#SensorTag_Android_Development)

To integrate a new Bluetooth service into the existing Android project, the DeviceActivity.java, SensorTagGatt.java and the gatt\_uuid.xml files will have to be modified. Also, a new Bluetooth profile class will have to be created. If the service

requires more than a generic row to display its data then a new row class will have to be created.

### SensorTagGatt.java modification

The SensorTagGatt.java class contains the UUIDs for all Bluetooth services used. The UUIDs of the new service must be added to this class. Figure 57 shows an example entry for a new custom profile commented out.

```

    UUID_BATT_SERV = fromString("0000180F-0000-1000-8000-00805f9b34fb"),
    UUID_BATT_LEVEL = fromString("00002A19-0000-1000-8000-00805f9b34fb"),
    UUID_BATT_CONFIG = fromString("f0002A1A-0451-4000-b000-000000000000"),
    UUID_BATT_PERIOD = fromString("f0002A1B-0451-4000-b000-000000000000");

    //UUID_CUSTOM_SERV = fromString("0000XXXX-0000-1000-8000-00805f9b34fb"),
    //UUID_CUSTOM_DATA = fromString("0000XXXX-0000-1000-8000-00805f9b34fb"),
    //UUID_CUSTOM_CONFIG = fromString("f000XXXX-0451-4000-b000-000000000000"),
    //UUID_CUSTOM_PERIOD = fromString("f000XXXX-0451-4000-b000-000000000000");
}

```

Figure 57, Example template of the UUID entries for a custom service commented out in the SensortagGatt.java class.

### gatt\_uuid.xml modification

The gatt\_uuid.xml file contains descriptions and names of the UUIDs of each service and its characteristics and are used by the user interface. Descriptions and names will need to be given for a new service or profile. The commented-out code in the gatt\_uuid.xml file, seen in Figure 58, can be used as a template.

```

<item uuid="0xCCC3" descr="Change the value to disconnect">Disconnect request</item>
<item uuid="0x180F">Battery Service</item>
<item uuid="0x2A19" descr="returns 12 bit adc reading in a 2 byte long array">Battery Level</item>
<item uuid="0x2A1A">Battery Config</item>
<item uuid="0x2A1B">Battery Period</item>
<!--
<item uuid="0xXXXX">Custom Service</item>
<item uuid="0xXXXX" descr="Custom service or functionality">Battery Level</item>
<item uuid="0xXXXX">Battery Config</item>
<item uuid="0xXXXX">Battery Period</item>
-->
</profile>
<profile>

```

Figure 58, Example template of UUID name and description entries commented out in gatt\_uuid.xml

## Creating a Custom Bluetooth Profile Class

A custom Bluetooth profile class will be an extension of the generic Bluetooth profile class. These together will handle reading and writing data to the service characteristics, initializing the table row on the user interface, and any functionality related to the custom Bluetooth profile.

To create a new custom Bluetooth class, use the SensorTagCustomProfile template found in the source, BleSensorTag-java-com.powerfilm.les100-ble-sensortag-SensorTagCustomProfile.java

Uncomment code sections during implementation.

## DeviceActivity.java modification

The DeviceActivity.java code runs when the main activity selects a device to connect too. The new profile must be added to the existing code structure so it is discoverable.

Use the commented-out code, seen in Figure 59, as a template. This code is run in the mGattUpdateReceiver broadcast receiver when the onReceive method is triggered.



```

1
if (SensorTagBatteryProfile.isCorrectService(s)) {
    Batt = new SensorTagBatteryProfile(context,mBluetoothDevice,s,mBtLeService);
    mProfiles.add(Batt);
    BattProfileIndex = mProfiles.size() -1;
    if (nrNotificationsOn < maxNotifications) {
        Batt.configureService();
        nrNotificationsOn++;
    }
    else {
        Batt.grayOutCell(true);
    }
}

//if (SensorTagCustomProfile.isCorrectService(s)) {
//    Custom = new SensorTagCustomProfile(context,mBluetoothDevice,s,mBtLeService);
//    mProfiles.add(Custom);
//    CustomProfileIndex = mProfiles.size() -1;
//    if (nrNotificationsOn < maxNotifications) {
//        Custom.configureService();
//        nrNotificationsOn++;
//    }
//    else {
//        Custom.grayOutCell(true);
//    }
//}
if (SensorTagSimpleKeysProfile.isCorrectService(s)) {

```

Figure 59, Example Custom Service initialization entry commented out in DeviceActivity.java

The Custom profile will also need to be added to the code that enables each row, use the commented-out line as a template (Figure 60).

```

for (final GenericBluetoothProfile p : mProfiles) {
    runOnUiThread((Runnable) () -> {
        if (p.equals(mProfiles.get(LuxProfileIndex)) || p.equals(mProfiles.get(CCSPProfileIndex)) || p.equals(mProfiles.get(
        //if (p.equals(mProfiles.get(CustomProfileIndex)) || p.equals(mProfiles.get(LuxProfileIndex)) || p.equals(mProfiles
        mDeviceView.addRowToTable(p.getTableRow());
        p.enableService();
        progressDialog.setProgress(progressDialog.getProgress() + 1);
    } else {
        p.disableService();
        mDeviceView.addRowToTable(p.getTableRow());
        progressDialog.setProgress(progressDialog.getProgress() + 1);
    }
    });
    p.onResume();
}
runOnUiThread((Runnable) () -> {
    mProgressDialog.hide();
}

```

Figure 60, location to enable a Custom Service during initialization in DeviceActivity.java, example commented out.

## Custom Profile Row

By default, a new profile will use the GenericCharacteristicTableRow to display data using a Sparkline, turn the service or sensor on/off, and adjust the service period.

If the custom profile requires addition elements on the row a designated row class that extends the GenericCharacteristicTableRow will be required.

See the profiles that use designated row classes for examples on how to implement a custom row class.

## Block Diagram

### Diagram

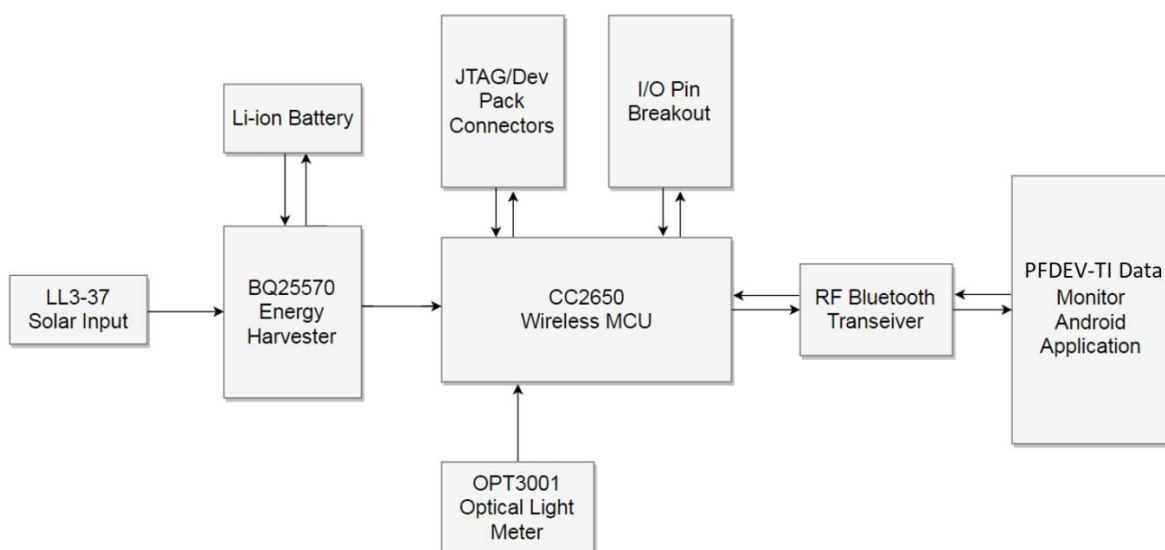


Figure 61, DEV-IN-BLE-TI system block diagram.

## Block Diagram Notes

### BQ25570 Energy Harvester

- Responsible for collecting solar energy from the indoor solar panel and storing that energy into a Li-ion battery or other rechargeable storage element.
- Provides steady power to the entire system. Output voltage is adjustable from 1.3V to Storage Element voltage via  $R_{out\_1}$  and  $R_{out\_2}$ .

### CC2650 Wireless MCU

- Responsible for collecting data and configuring sensors.
- Handles Bluetooth Low Energy connection and communication with external Bluetooth devices.

### PFDEV-TI Data Monitor

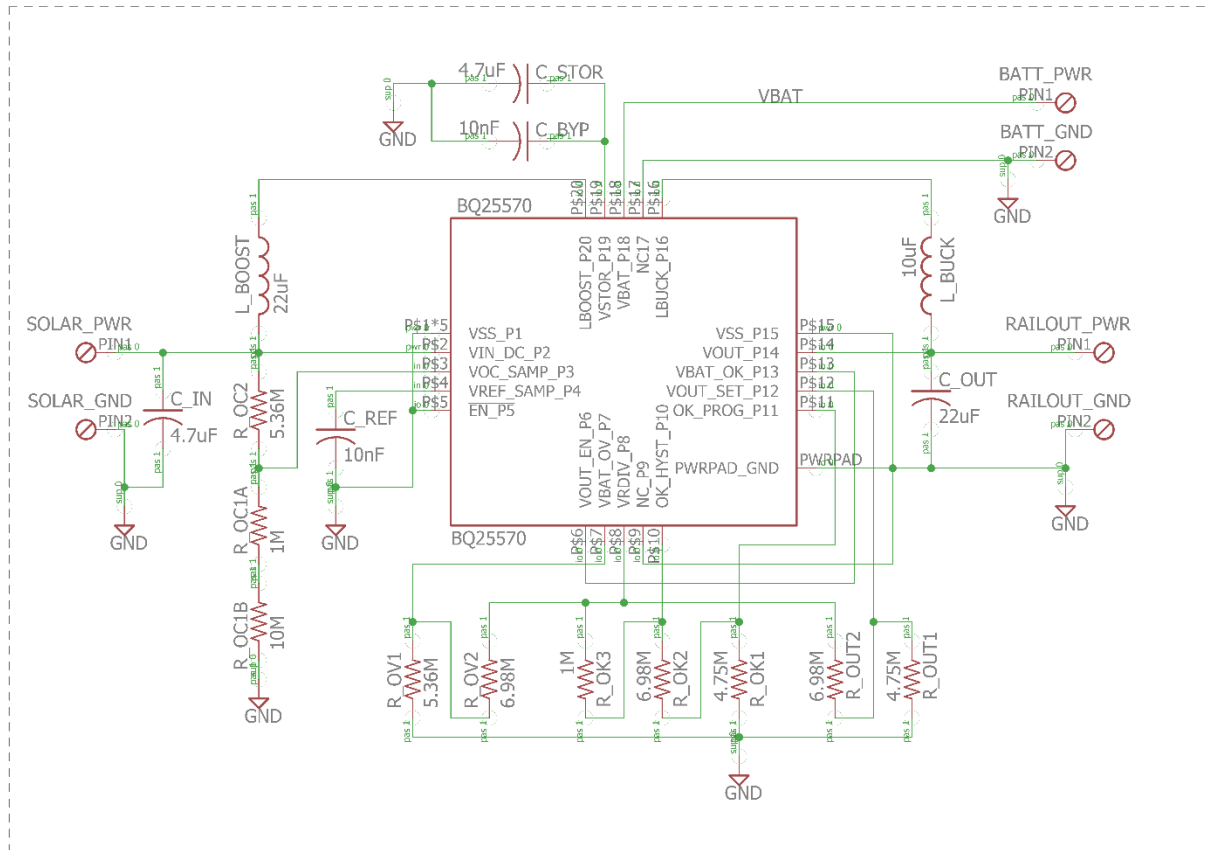
- Receives and displays data from connected PFDEV-TI devices.
- Provides a user interface for configuring sensor and Bluetooth connection settings.

## Schematic

### Overview

The DEV-IN-BLE-TI board (PFDEV-TI device) design can be broken down into the energy harvesting portion (DEV-IN-BASIC) and the Bluetooth MCU portion. These components are physically separated on different halves of the circuit board and are only connected by ground, the power supply rail, and the battery level signal.

# DEV-IN-BASIC



The BQ25570 is an ultra-low-power boost-buck charge controller and battery management dedicated IC. Solar is connected to the SOLAR\_GND and SOLAR\_PWR pins and the storage element is connected to BATT\_GND and BATT\_PWR pins. The BQ25570 will provide steady power to the RAILOUT\_PWR rail using energy collected from the solar or from energy stored in the battery.

The energy harvesting circuit is connected to the Bluetooth MCU circuit through RAILOUT\_PWR, RAILOUT\_GND, and the VBATT\_SIGNAL pins.

## Bluetooth MCU CC2650 MCU

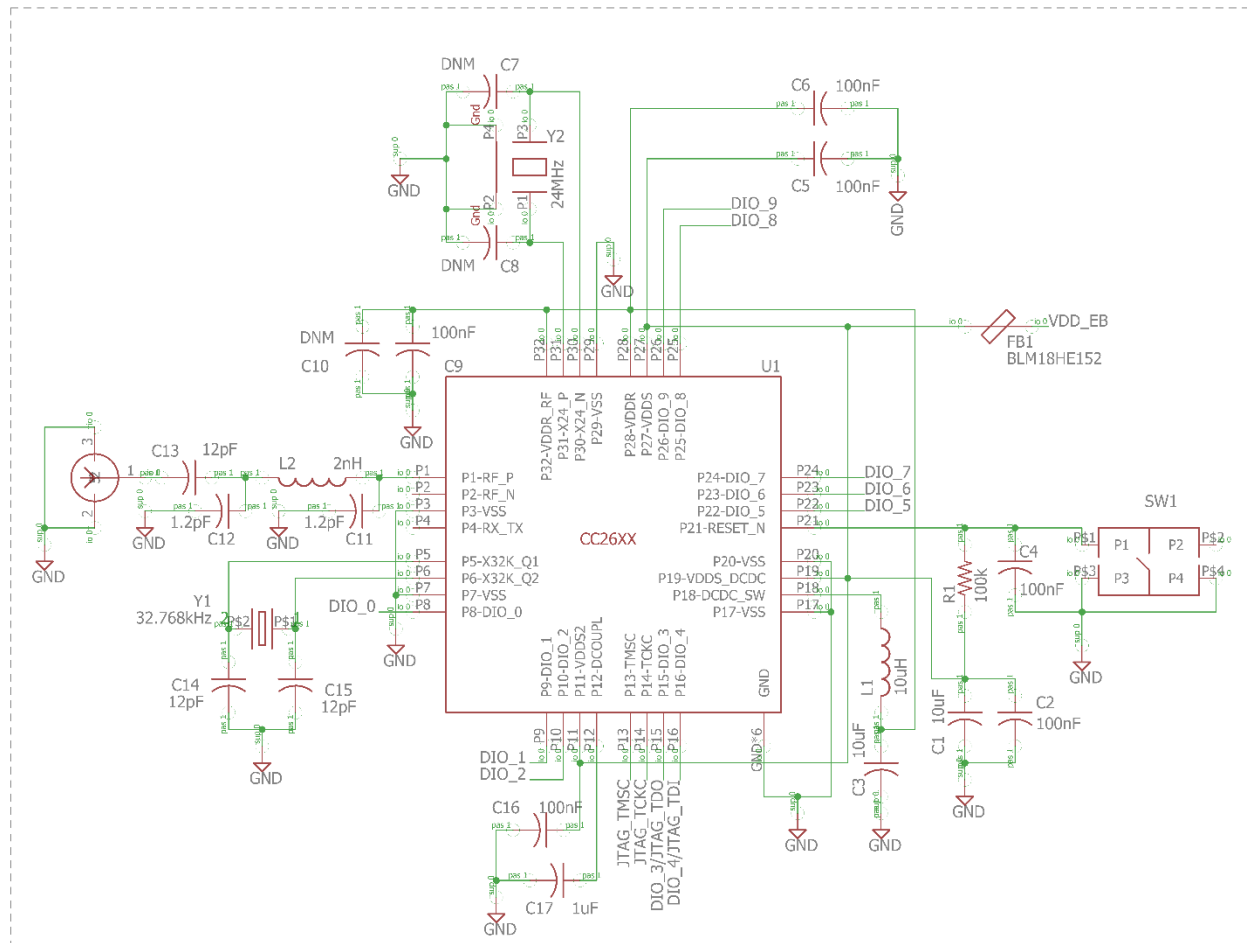
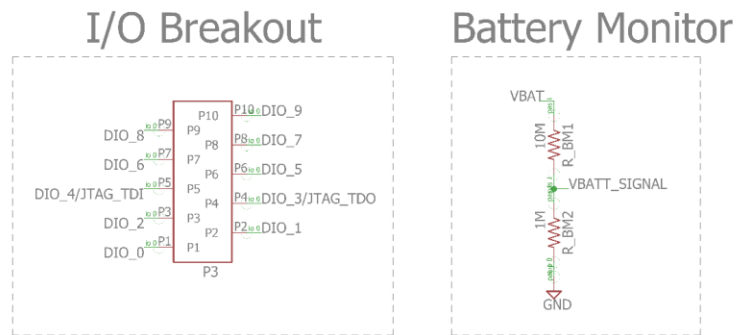


Figure 63, DEV-IN-BLE-TI Bluetooth MCU



## Solder Bridge Connections

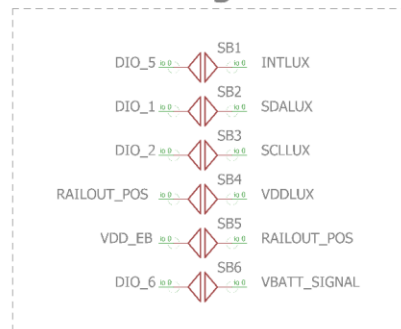
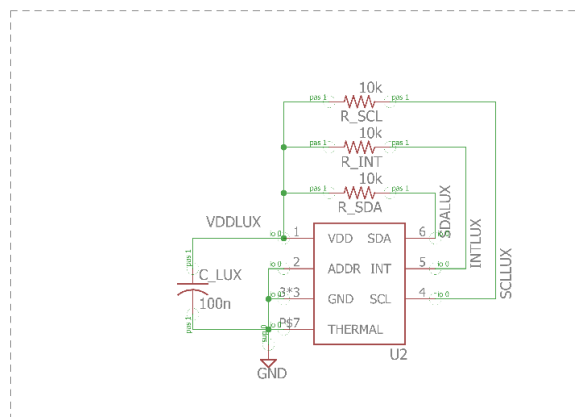


Figure 64, DEV-IN-BLE-TI IO

## OPT3001 Lux Meter



## Debugger Dev Pack

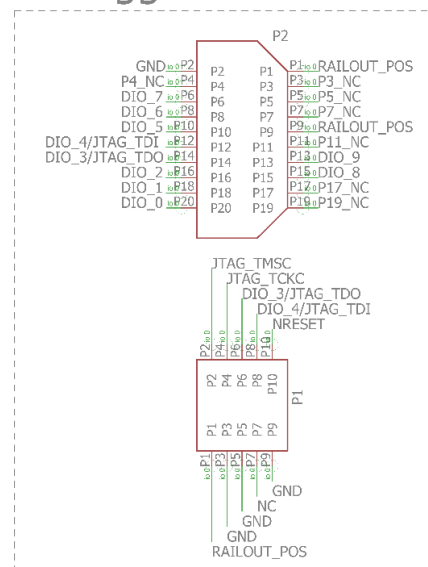


Figure 65, DEV-IN-BLE-TI Lux Meter and Debugger Dev Pack.



The CC2650 Wireless MCU runs the DEV-IN-BLE-TI BLE application, collects data from sensors, and handles Bluetooth communication.

### I/O

Sensors are connected to IO of the CC2650. The OPT3001 uses three IO pins to set up I2C serial communication with the MCU; DIO\_1, DIO\_2, and DIO\_5. The Battery Level signal is routed to pin DIO\_6.

### Ports and Connectors

P1 is the programming JTAG interface.

P2 is the 20 pin TI Dev Pack connector. This enables TI sensor Dev Packs to be plugged in and integrated into the DEV-IN-BLE-TI system.

P1 and P2 are physically oriented to allow the TI Debugger DevPack to be connected and used for programming and debugging. (See Code Composer section for more information on setting up the Debugger Dev Pack)

Access to all 10 IO pins is available via P3 female pin header.

### Switches

S1 is the reset button switch. Under normal operation, CC2650 NRESET pin is pulled high. When S1 is depressed then released, the NRESET pin is pulled to ground and the CC2650 will reset.

## Layout

### Top

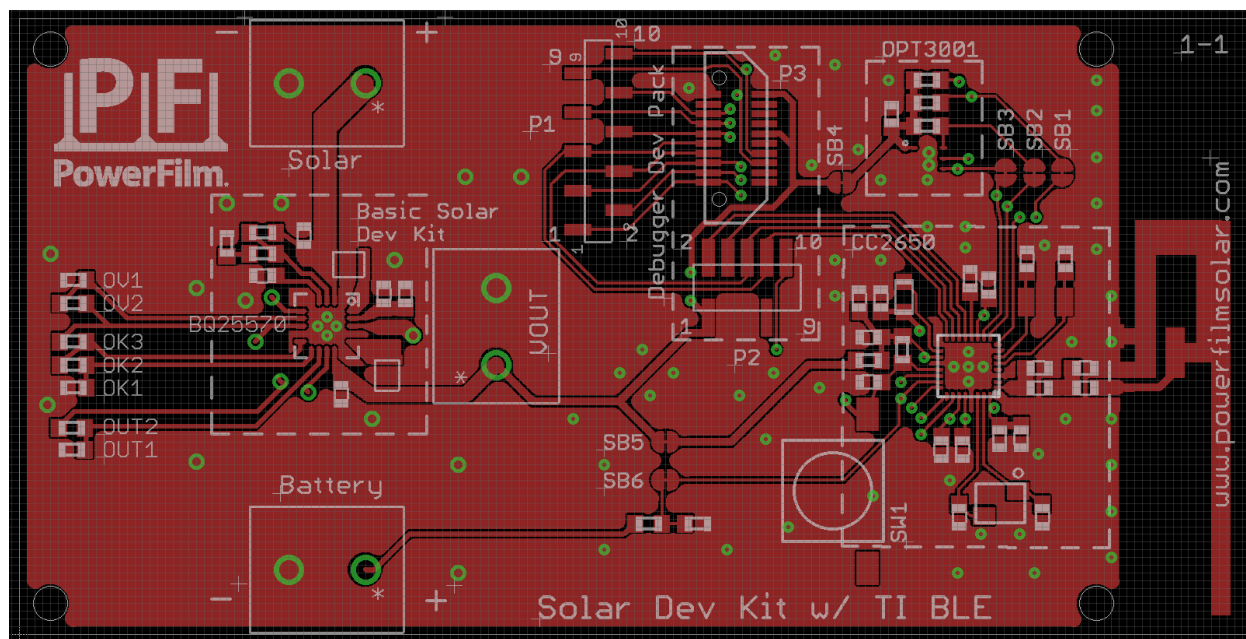


Figure 66, DEV-IN-BLE-TI Top Board Layer

### Bottom

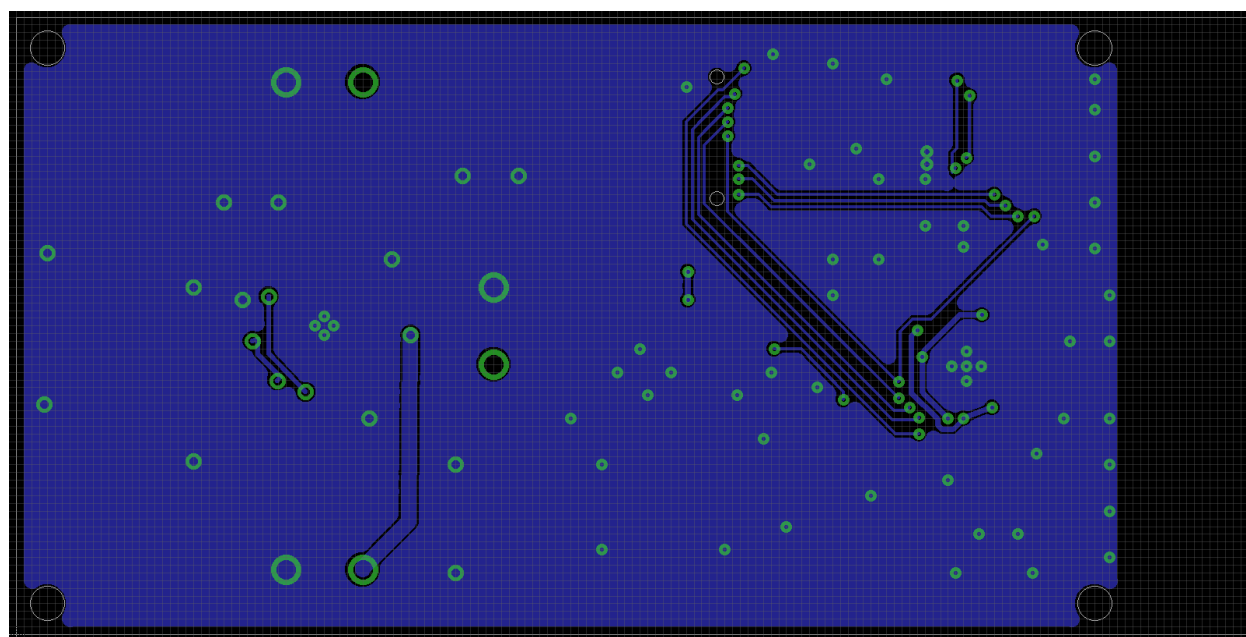


Figure 67, DEV-IN-BLE-TI Bottom Board Layer

## Part Placement

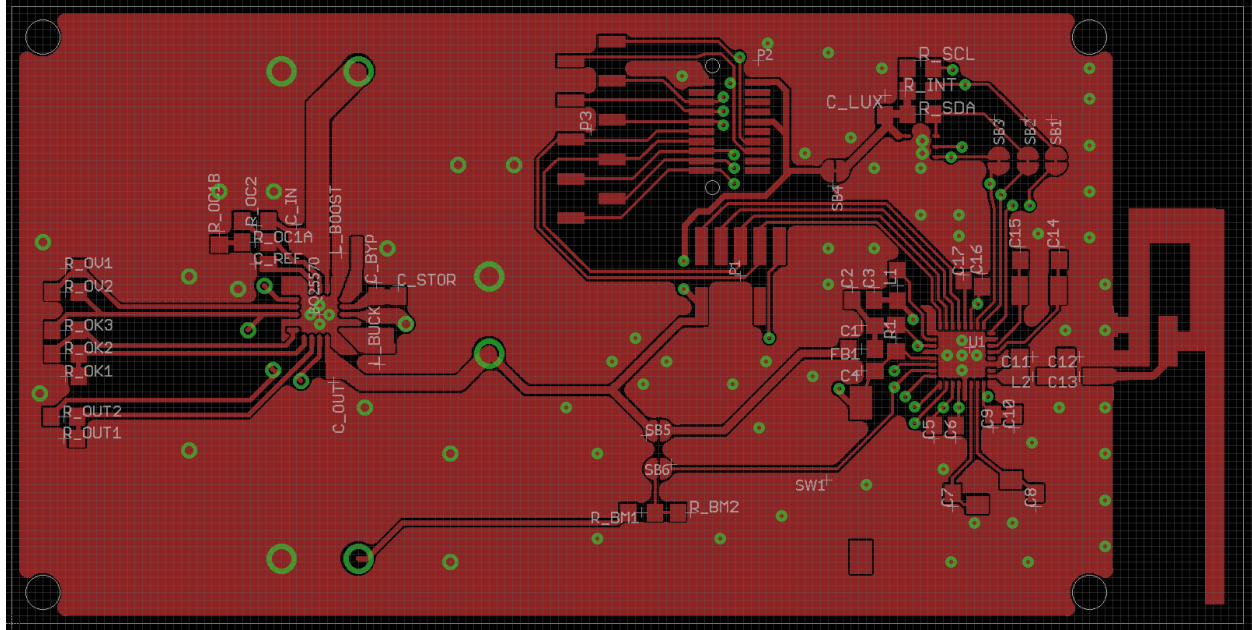


Figure 68, DEV-IN-BLE-TI Part Placement

## DEV-IN-BLE-TI Application Notes

### Power Consumption Optimization

The DEV-IN-BLE-TI is able to adjust Bluetooth connection parameters, which can greatly reduce power consumption. The power consumption of the device is heavily dependent on the Bluetooth connection interval and the data period of the sensors. Both parameters can be adjusted within the application. Using long connection intervals (16 seconds max) and long time periods between sensor measurements will allow the device to operate at very low light levels (< 100 lux).

### Hardware Configurations

#### I/O Access

External sensors can be added to the system via the 10 pin I/O breakout header. On board sensors can be disconnected by removing SMD jumpers so that all I/O pins can be accessed.

The Luxmeter can be disconnected by removing resistors R3, R7, R8, and R10. This will free I/O pins DIO\_1, DIO\_2, and DIO\_3 and disconnect the OPT3001 from the power supply rail.

The Battery monitor circuitry can be disconnected by removing resistor R9. This will free DIO\_6.

### Charge Control Configuration

The DEV-IN-BLE-TI hardware is currently configured to charge a Li-ion type battery with max voltage of 4.2V and the output voltage is set to 3.0V. The configuration can be customized by modifying SMD resistor dividers per the BQ25570 datasheet specifications which can be found at <http://www.ti.com/lit/ds/symlink/bq25570.pdf> section 7.3.

- Output voltage can be set using R<sub>OUT\_1</sub> and R<sub>OUT\_2</sub>.
- Max Battery Voltage can be set using R<sub>OV\_1</sub> and R<sub>OV\_2</sub>.
- The minimum battery voltage and output turn on voltage can be set using resistors R<sub>OK\_1,2,3</sub>.

### Capacitor/Super Capacitor Storage Element Operation

The DEV-IN-BLE-TI is capable of running and operating with a capacitor as the storage element instead of the li-ion battery. Except the capacitor, the hardware configuration of the board can remain the same.

### Theory of Operation

A capacitor will behave similarly to a small capacity battery when used as a storage element. It will charge and discharge quicker and may not hold as much energy as a li-ion battery.

Starting from fully discharged, when the is placed in a decently lit room the capacitor will begin to charge up. Once the voltage is above 3.2V the Bluetooth radio will be enabled and ready to connect to the PFDEV-TI Data Monitor Android Application.

The capacitor will maintain steady power to the system while light is available. If the lights are turned off, the capacitor will discharge. When it reaches 3V the radio will shut down.

The PFDEV-TI Data Monitor comes configured to remember the last connected device and will automatically reconnect once the device turns back on.

### Specifications

- Capacitor must be at least 1800uF or greater.
- Capacitor must be rated for 6V or greater.

Charge and discharge rate will be greatly affected by the size of the capacitor. If the capacitor is completely discharged (0V) the charge rate will be slower because the harvester chip is not yet fully functional. Figures 69 and 70 show charge up times vs storage capacitance size for 0V-3V and 3V-4.2V.

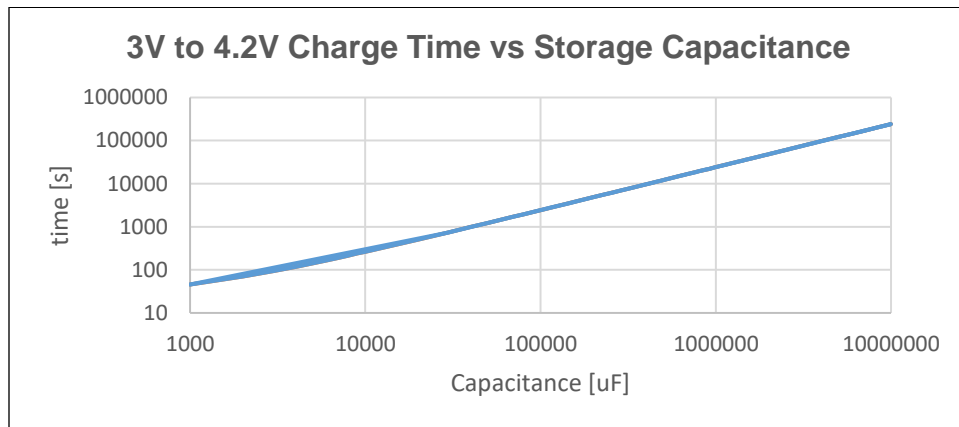


Figure 69, 3V to 4.2V charge time vs storage capacity at 300 lux.

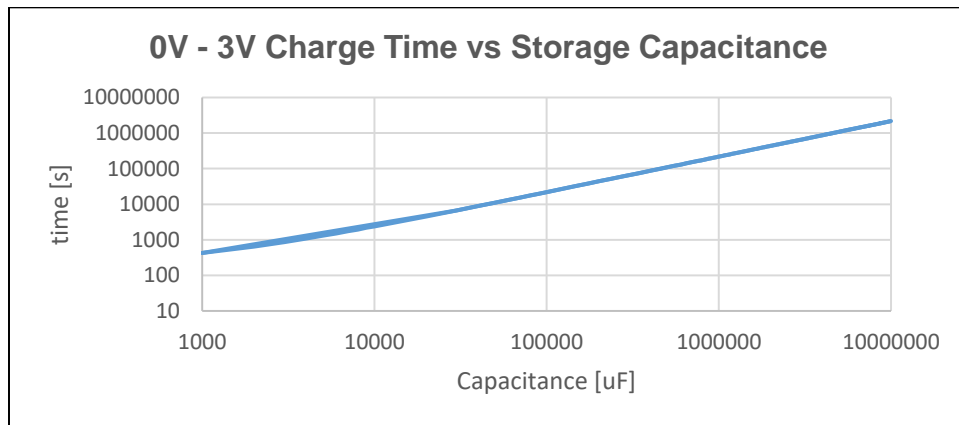


Figure 70, 0V-3V charge time vs storage capacitance at 300 lux.

## CC2650 Microcontroller Reset

If the PFDEV-TI device experiences a Bluetooth connection failure it may enter lengthy software reset loops depending on the severity and circumstances of the failure. For this reason, a hardware reset button is present on the circuit board to quickly reset the Bluetooth functionality of the PFDEV-TI.