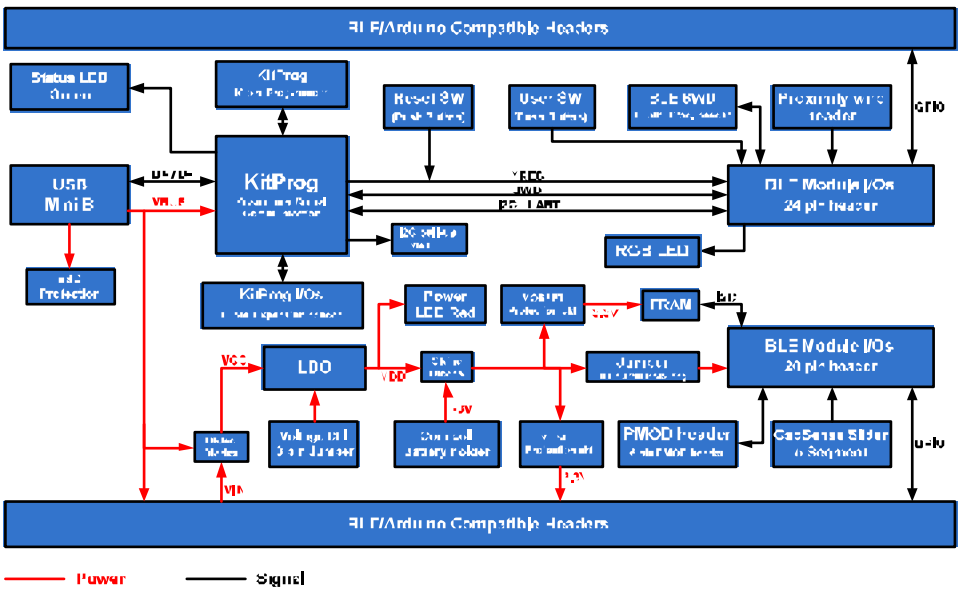
3. Kit Operation

This chapter introduces you to the BLE Pioneer Kit and the features that will be used as part of its operation. We will discuss features such as USB connection, programming/debugging, and programmer firmware update. The chapter also describes the USB-UART and USB-I²C bridges along with the PC tools that can be used to communicate with the BLE device on the BLE Pioneer Kit.

3.1 Theory of Operation

Figure 3-1, Figure 3-2, and Figure 3-3 show the block diagrams for the BLE Pioneer Baseboard, PSoC 4 BLE PRoC BLE Module, and BLE Dongle.

Figure 3-1. BLE Pioneer Baseboard Block Diagram

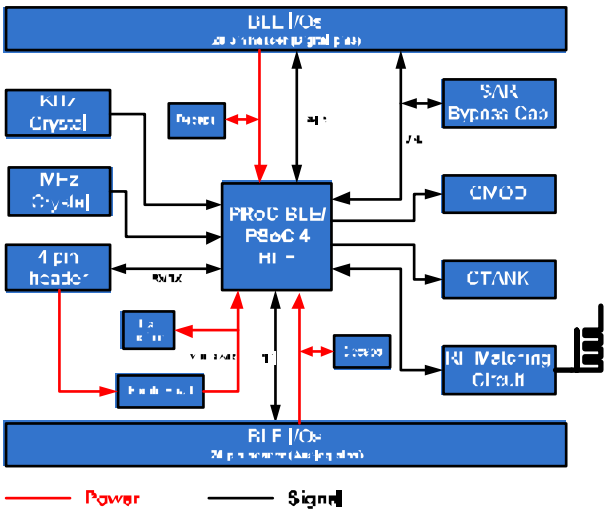


The BLE Pioneer Baseboard acts as the baseboard for the PSoC 4 BLE (red module) and PRoC BLE (black module). The BLE Pioneer Baseboard contains a PSoC 5LP device, that has KitProg firmware, used as an onboard programmer or debugger, and for the USB-Serial interface.

The baseboard is Arduino form-factor compatible, enabling Arduino shields to be connected on top of the board to extend the functionality of BLE modules. The board also features a 1-Mb F-RAM, an RGB LED, a five-segment CapSense slider, a proximity header, a user switch, and a reset switch for the PSoC 4 BLE and PRoC BLE devices on the module. The Pioneer board supports three voltage levels: 1.8 V, 3.3 V, and 5 V.

The BLE Pioneer Baseboard can also be used as a standalone programmer to program and debug other BLE devices using SWD, and as a USB-Serial interface. The KitProg firmware on PSoC 5LP device enables bootloading PSoC 5LP over USB to upgrade the firmware.

Figure 3-2. PSoC 4 BLE PRoC BLE Module Block Diagram

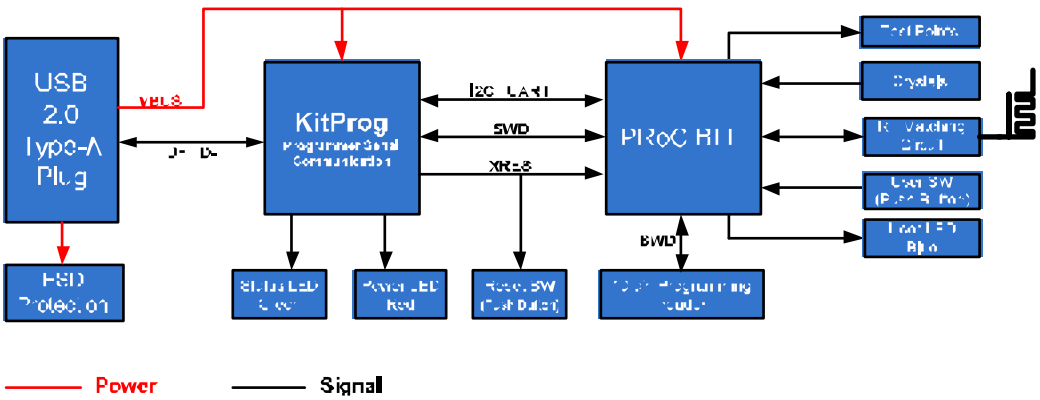


This BLE Pioneer Kit includes two modules. These modules act as a basic breakout board for the CY8C4248LQI-BL583 (PSoC 4 BLE) and CYBL11573-56_QXI (PRoC BLE) device. The PSoC 4 BLE and PRoC BLE Modules are identical except for the BLE device. Besides these two modules, there are additional modules available, which can be ordered separately. The complete list is available in BLE Modules and BLE Dongles Compatible with the BLE Pioneer Kit on page 130.

The BLE Dongle is the wireless interface for the CySmart Central Emulation Tool. It has a PRoC BLE device for BLE communication and KitProg for onboard programming, debugging, and for the USB-Serial interface, as shown in Figure 3-3.

The BLE Dongle has a USB Type-A plug to connect the KitProg to the USB port of the host computer. The KitProg then communicates with the PRoC BLE device over UART or multiplexed I^2C or an SPI bus. The BLE Dongle also features a user LED, a user switch, and a reset switch for the PRoC BLE device. The dongle is powered directly through the USB port (VBUS) at 5.0 V.

Figure 3-3. BLE Dongle Block Diagram



3.2 KitProg

KitProg is the hardware/firmware block for onboard programming, debugging, and bridge functionality. It is a common reusable hardware/firmware block used across many Cypress kit platforms. It consists of a PSoC 5LP which connects to the computer over an USB interface and connects to the PSoC 4 BLE or PRoC BLE device over SWD, I^2C, and UART pins.

The KitProg communicates with PSoC Programmer and PSoC Creator software to program/debug the target PSoC 4 BLE or PRoC BLE over the SWD interface. The main advantage of an onboard programmer/debugger is that users do not have to buy an extra programmer debugger hardware.

3.3 BLE Pioneer Kit USB Connection

The BLE Pioneer Kit powers from a computer over the USB interface (J13). It enumerates as a composite device, as shown in Table 3-1. USB drivers required for this enumeration are part of the kit installer. The kit should be installed properly for its correct operation.

Visit www.cypress.com/CY8CKIT-042-BLE for the latest kit installer.

Table 3-1. BLE Pioneer Kit Enumerated Interfaces

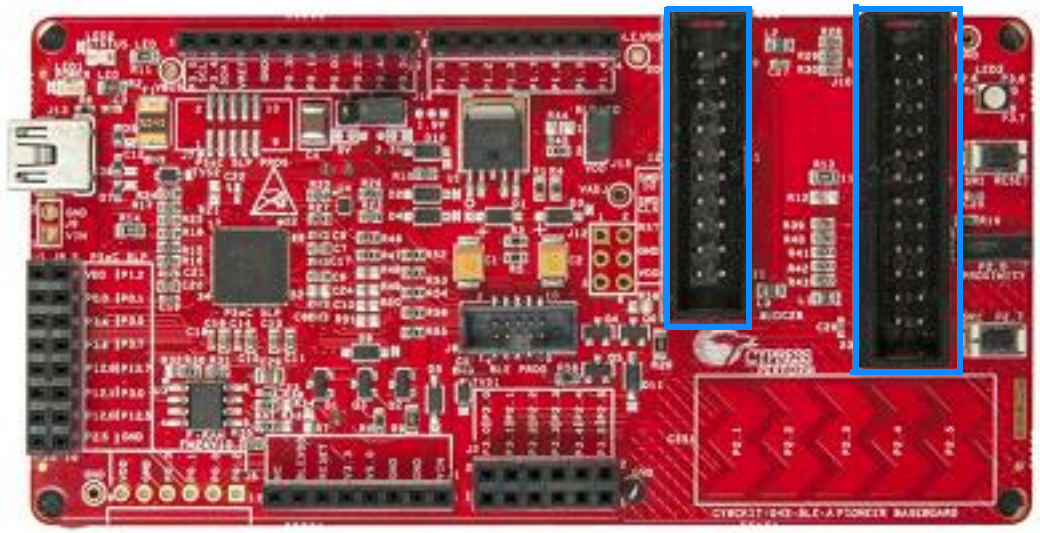
Port	Description
USB Composite Device	Composite device
USB Input Device	USB-I^2C bridge, KitProg command interface
KitProg	USB-I^2C bridge, programmer, and debugger
KitProg USB-UART	USB-UART bridge, which appears as a COM# port

Figure 3-1. KitProg Driver Installation (appearance may differ depending on Windows version)

3.4 Placing Modules on Baseboard

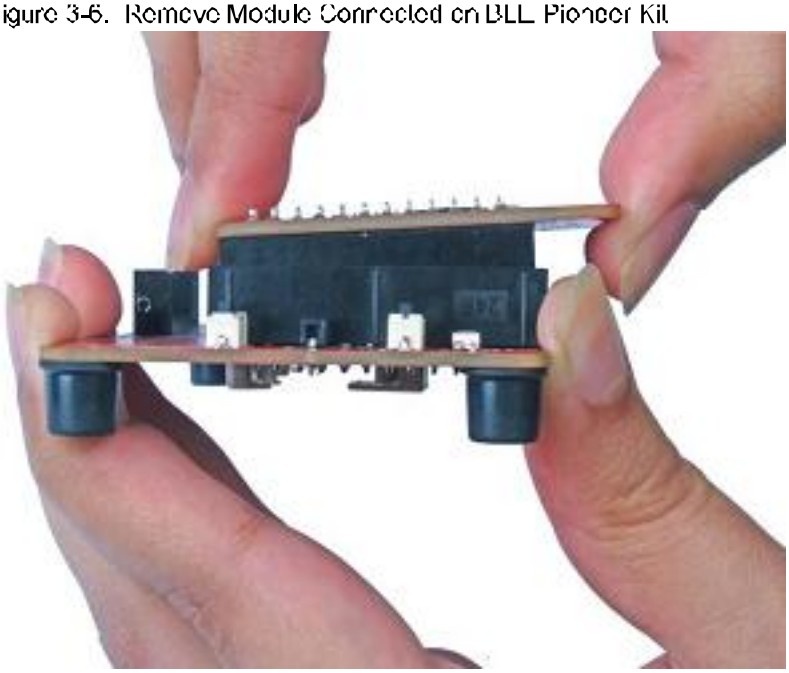
Plug the module into the BLE Pioneer Baseboard on headers J10 and J11, while keeping the antenna directed outside. Note that the two parallel headers J10 and J11 are not equal (24-pin and 20-pin, respectively) and will not allow the module to be inserted in the opposite direction.

Figure 3-5. Baseboard with J10 and J11 Headers to Connect Modules



To remove the modules from the BLE Pioneer Kit, hold the BLE Pioneer Kit in one hand and the module in the other, as shown in Figure 3-6, and pull it out using a rocking motion.

Figure 3-6. Remove Module Connected on BLE Pioneer Kit



3.5 Programming and Debugging BLE Device

The BLE Pioneer Kit and BLE Dongle can be programmed and debugged using the KitProg. Before programming the device, ensure that PSoC Creator and PSoC Programmer are installed on the computer. See Install Software on page 21 for more information.

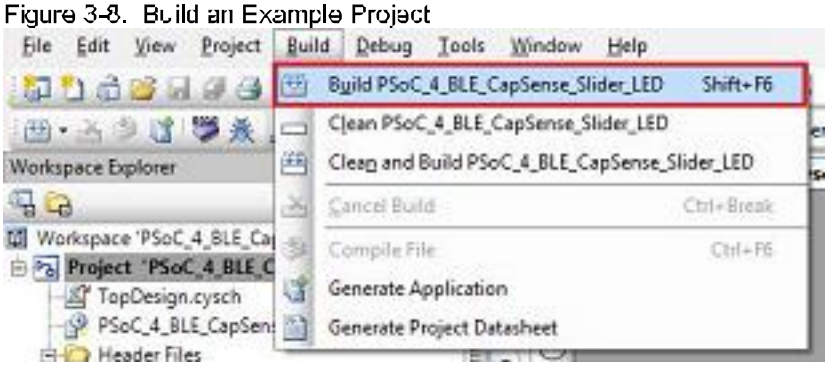
3.5.1 Programming using PSoC Creator

1. Connect the BLE Pioneer Kit/BLE Dongle to the computer's USB port, as shown in Figure 3-7.

Figure 3-7. Connect USB Cable to J13

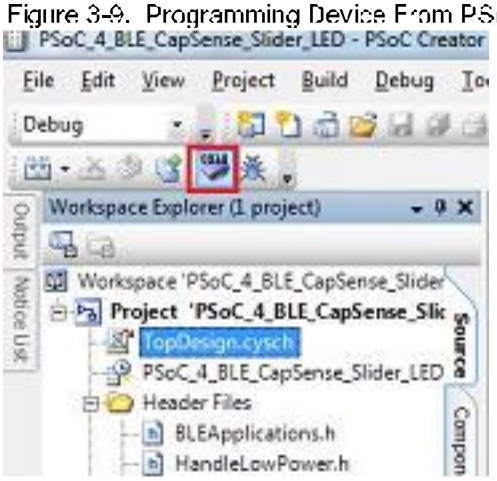
2. Load the desired example project in PSoC Creator from **File** > **Open** > **Project/Workspace**.
3. Build the project by choosing **Build** > **Build <Project Name>** or **[Shift] [F6]**, as shown in Figure 3-8.

Figure 3-8. Build an Example Project



4. If there are no errors during build, program the firmware by clicking the **Program** button on the tool bar or pressing **[Ctrl] [F5]**, as shown in Figure 3-9. This will program the device on the BLE Pioneer Kit BLE Dongle and it will be ready for use.

Figure 3-9. Programming Device From PSoC Creator

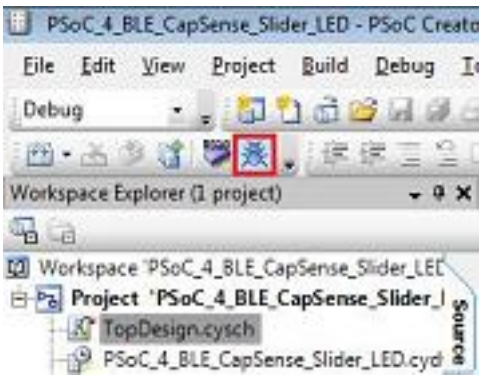


3.5.2 Debugging using PSoC Creator

For debugging the project using PSoC Creator, follow steps 1 to 5 from Programming using PSoC Creator on page 29 followed by:

1. Click the **Debug** icon or press **[F5]**, as shown in Figure 3-10.

Figure 3-10. Start Debug on PSoC Creator



2. When PSoC Creator opens in debug mode, use the buttons on the toolbar for debugging.

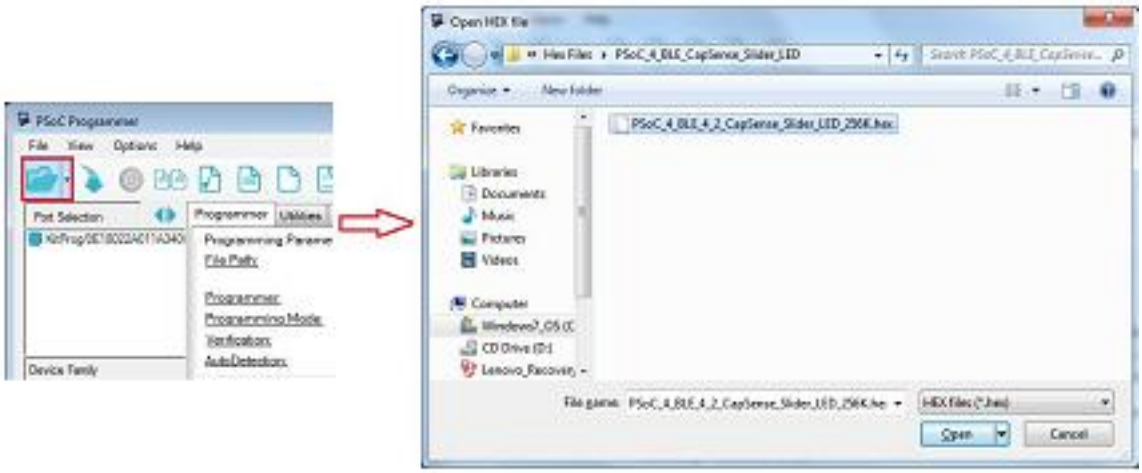
For more details on using the debug features, see the Cypress application note Getting Started with PSoC 4 BLE.

3.5.3 Programming using PSoC Programmer

PSoC Programmer (3.24 or later) can be used to program existing hex files into both BLE Pioneer Kit or BLE Dongle. To do this, follow these steps.

1. Connect the BLE Pioneer Kit or BLE Dongle to a computer and open PSoC Programmer from **Start > All Programs > Cypress > PSoC Programmer <version> > PSoC Programmer <version>**.
2. Click the **File Load** button at the top left corner of the window. Browse for the desired hex file and click **Open**.

Figure 3-11. Select Hex File



3. Go to **File > Program** to start programming the kit with the selected file.

   Note: If the hex file does not match the device selected, then PSoC Programmer will throw an error of device mismatch and terminate programming.

Figure 3-12. Program Hex File to Kit

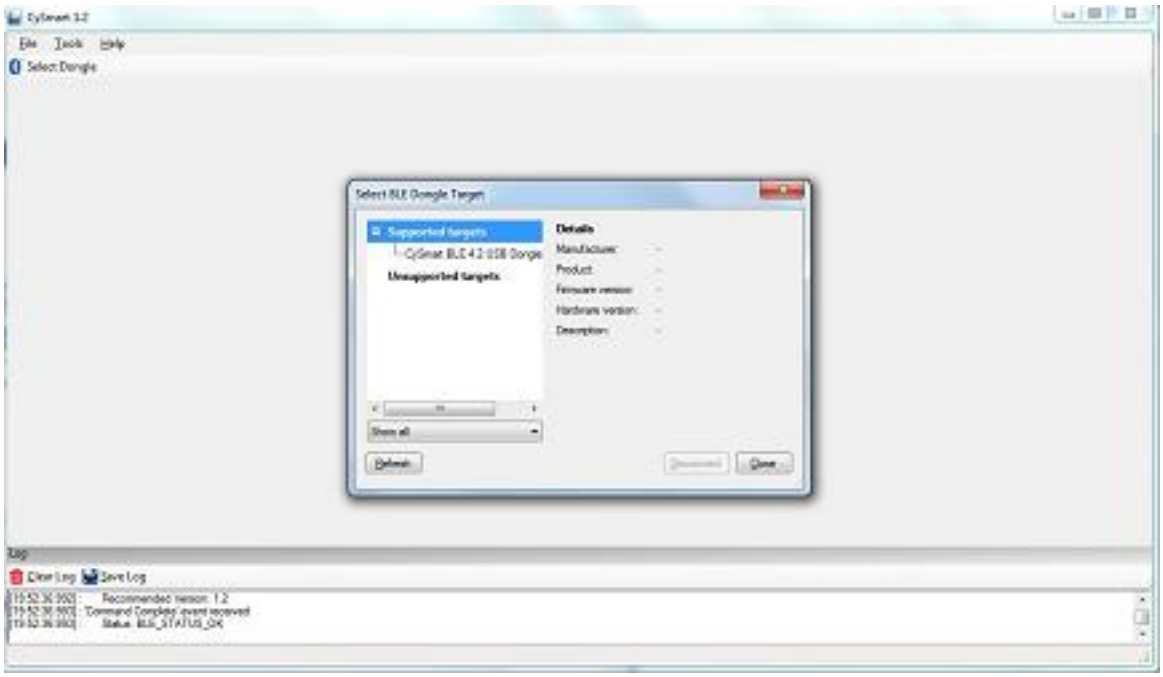
4. When the programming is finished successfully, indicated by a PASS message on the status bar, the BLE Pioneer Kit/BLE Dongle is ready for use. Close PSoC Programmer.

3.6 Updating BLE Dongle for CySmart Central Emulation Tool

The BLE Dongle provides a BLE Central mode capability using the CySmart Central Emulation Tool on the computer. The CySmart Central Emulation Tool on the PC is the interface with which to configure the BLE Dongle and analyze the data transferred after connecting with a BLE Peripheral.

The BLE Dongle works along with the CySmart Central Emulation Tool, as shown in Figure 3-13. The CySmart Central Emulation Tool is installed as part of the BLE Pioneer Kit installation and can be opened from **Start** > **All Programs** > **Cypress** > **CySmart <version>** > **CySmart <version>**. The tool operation is explained in the user guide, which can be accessed from **Help** > **Help Topics**.

Figure 3-13. BLE Dongle Interface on CySmart Central Emulation Tool



If the BLE Dongle contains custom firmware on PRoC BLE, the original CySmart firmware can be programmed back to restore the CySmart functionality. It must be connected through the USB and enumerated as KitProg. To do this, follow these steps:

1. Connect the BLE Dongle to the USB port of the computer.
2. Open PSoC Programmer by going to **Start** > **All Programs** > **Cypress** > **PSoC Programmer <version>** > **PSoC Programmer <version>**.
3. Click the **File Load** button and browse to the location of the *BLE_4_2_Dongle_CySmart_256K.hex* file. The hex file is located at:
   <Install Directory>\CY8CKIT-042-BLE-A\<version>\Firmware\BLE Dongle Hex Files

   Note: If Cypress releases new versions of the CySmart Central Emulation Tool and the BLE Dongle firmware, then the CySmart Central Emulation Tool will display a message requesting to update the firmware, as shown in the following figures.

Figure 3-14. Update BLE Dongle Firmware with Hex from Latest Kit Installer

Choose the .hex file from the respective location and update the BLE Dongle firmware.

Figure 3-15. Open Hex File

4. Ensure the other settings match as shown in Figure 3-15. Click the **Program** button to start programming. The status bar at the bottom of the PSoC Programmer window will show the programming status and the result (Pass/Fail).

Figure 3-16. Programming Hex File to Dongle

5. After programming is completed successfully, the BLE Dongle firmware is updated and can be used to connect to the CySmart Central Emulation Tool.

3.7 USB-UART Bridge

The KitProg on both the BLE Pioneer Baseboard and BLE Dongle acts as a USB-UART bridge. When connected to a computer, a device named **KitProg USB-UART** is available under **Ports (COM & LPT)** in the Device Manager. The UART lines between modules and KitProg are hard-wired onboard, with UART_RX assigned to **P1_4** and UART_TX assigned to P1_5 on PSoC 4 BLE/PRoC BLE device.

COM terminal software, such as Hyperterminal or Teraterm, can be used to send and receive data. UART data sent from PSoC 4 BLE/PRoC BLE device on UART_TX line will be received by the software. Data entered in the software will be received by PSoC 4 BLE/PRoC BLE on UART_RX line. See CY8CKIT-042 PSoC 4 Pioneer Kit User Guide for more details.

Table 3-2 lists the specifications supported by the USB-UART bridge.

Table 3-2. Specifications Supported by USB-UART Bridge

Parameter	Supported Values
Baud Rate	1200, 2400, 4800, 9600, 19200, 38400, 57600, and 115200
Data Bits	8
Parity	None
Stop Bits	1
Flow Control	None
File Transfer Protocols supported	Xmodem, 1K Xmodem, Ymodem, Kermit, and Zmodem (only speeds greater than 2100 baud)

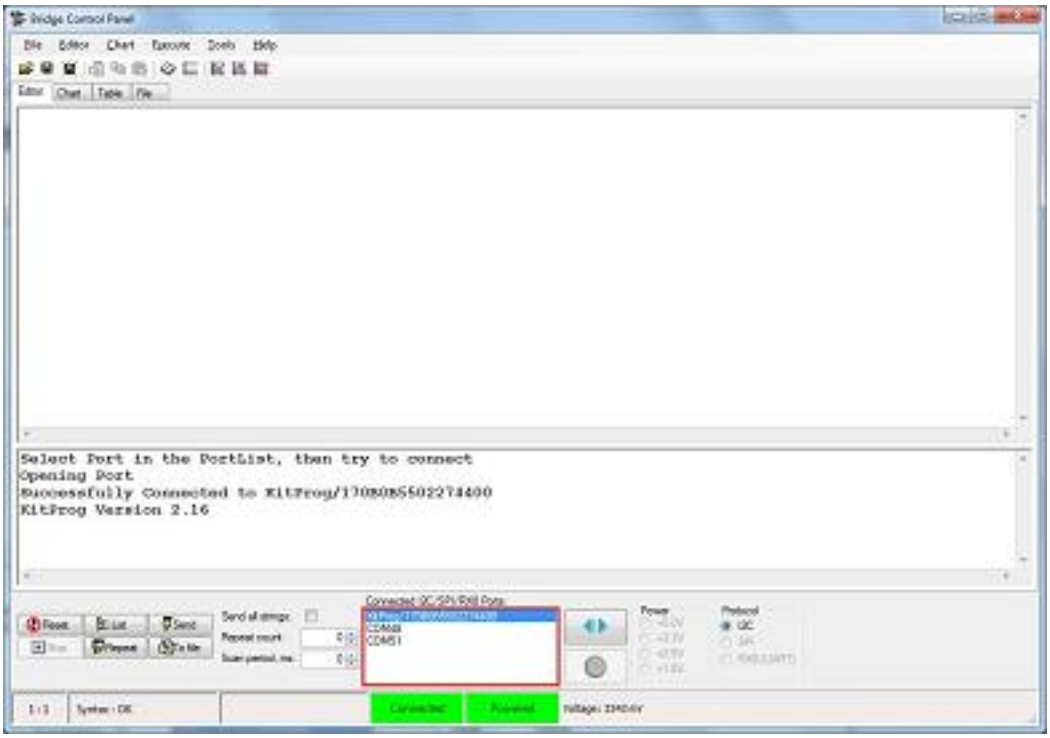
3.8 USB-I²C Bridge

The KitProg can function as USB-I²C bridge and communicate with the Bridge Control Panel (BCP) software utility. When connected to BCP, the **KitProg\<serial number>** is available under **Connected I2C/SPI/RX8 Ports** in the BCP. The I²C connection between PSoC 4 BLE/PRoC BLE device and KitProg is used to transfer data between BCP and the PSoC 4 BLE/PRoC BLE device. The I²C lines on PSoC 4 BLE/PRoC BLE device are **P3_4 (SDA)** and **P3_5 (SCL)**, which are hard-wired onboard to I²C lines of KitProg. The USB-I²C supports I²C speed of 50 kHz, 100 kHz, 400 kHz and 1 MHz.

BCP is installed as part of the PSoC Programmer installation and can be accessed from **Start > All Programs > Cypress > Bridge Control Panel**. Refer to the Advanced section in the CY8CKIT-042 PSoC® 4 Pioneer Kit User Guide for more details.

To use the USB-I²C functionality, select the **KitProg\<serial number>** in the BCP. On successful connection, the **Connected** and **Powered** status box turn green, as shown in Figure 3-17.

Figure 3-17. KitProg USB-I²C Connected in Bridge Control Panel



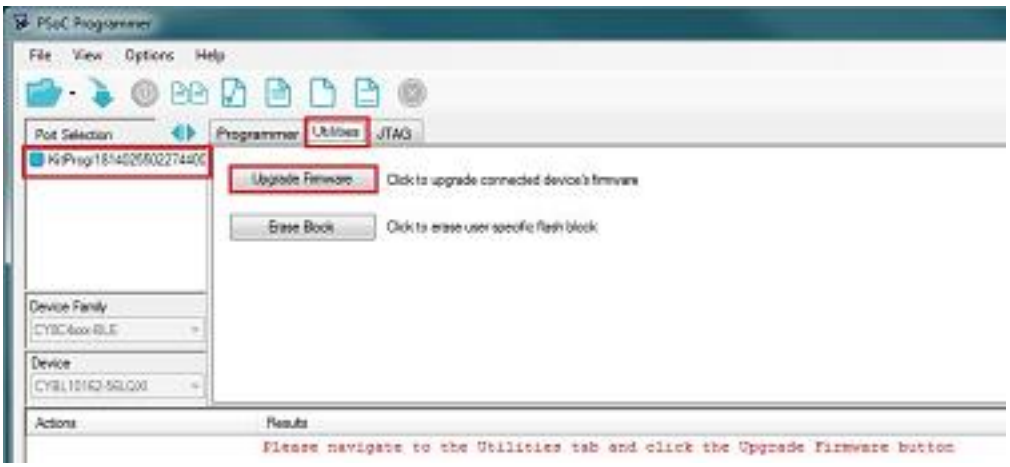
3.9 Updating the KitProg Firmware

The KitProg firmware normally does not require any update. If an update is required, then PSoC Programmer will display a warning message when the kit is connected to it, as shown in Figure 3-18.

Figure 3-18. Update KitProg

To update the KitProg, go to the **Utilities** tab on PSoC Programmer and click **Upgrade Firmware**, as shown in Figure 3-19.

Figure 3-19. Update KitProg from PSoC Programmer



4. Example Projects

This chapter demonstrates the functionality of PSoC 4 BLE and PRoC BLE devices using the BLE Pioneer Kit example projects. Download and install the kit setup file from the kit web page. The example projects can be accessed on the Start Page of PSoC Creator under **Examples and Kits**.

4.1 Using Example Projects

Follow these steps to open and use the example projects:

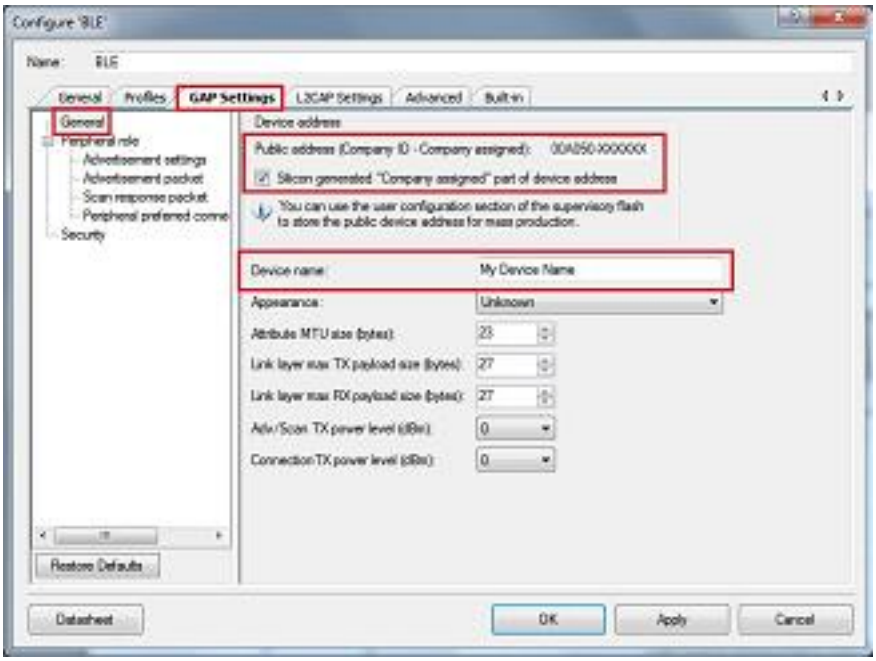
1. Launch PSoC Creator from **Start** > **All Programs** > **Cypress** > **PSoC Creator 3.3** > **PSoC Creator 3.3**.
2. On the Start Page, under the **Examples and Kits** section, choose **Kits** > **CY8CKIT-042-BLE-A**. A list of example projects appears, as shown in Figure 4-1. Projects named with the prefix '*PSoC_4_BLE_*' work on the BLE Pioneer Kit with the PSoC 4 BLE Module; projects named with the prefix '*PRoC_BLE_*' work on the BLE Pioneer Kit with the PRoC BLE Module.
3. Click the desired example project.

Figure 4-1. Open Example Project from PSoC Creator

4. Select the folder where you want to save the project and click **OK**.
5. Every BLE project use a public address set in the BLE Component GUI to advertise and scan depending on the role - Peripheral or Central mode. If multiple kits in close proximity have the same public address, then wrong devices may be connected or connections may fail. To prevent this, change the **Public address** (and preferably **Device name**) in the BLE Component **GAP Settings** tab as shown in Figure 4-2. Click **OK**.

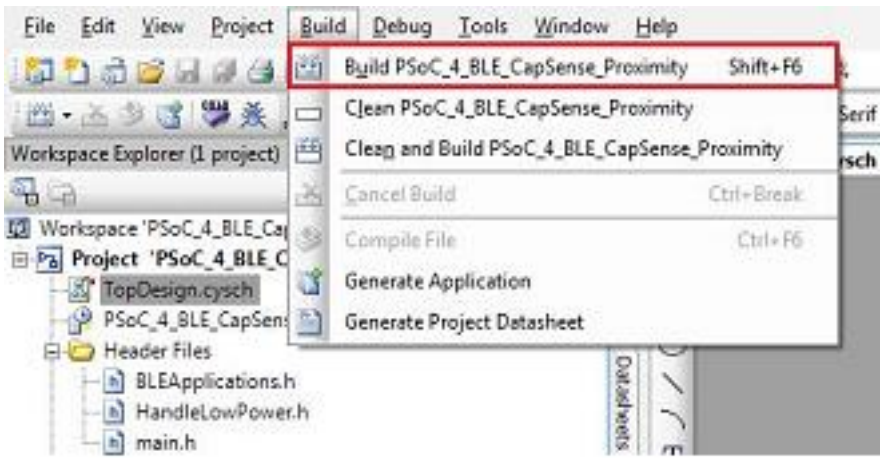
   Alternatively, you can select the 'Silicon generated' device address by selecting the check box. This way, the Bluetooth device (BD) address is generated using the silicon ID, unique to each device. Click **OK**.

Figure 4-2. Change BLE Public Address and Name



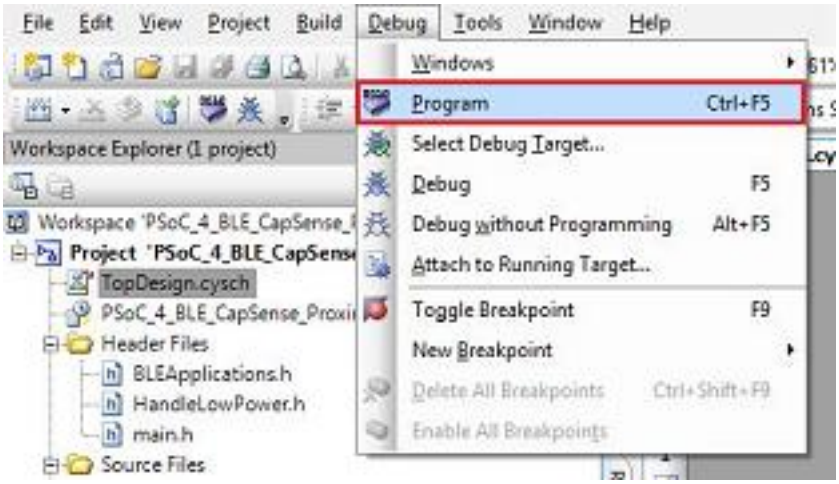
6. Build the example project by choosing **Build** > **Build <Project Name>**, as shown in Figure 4-3. A hex file will be generated.

Figure 4-3. Build Project from PSoC Creator



7. Connect the BLE Pioneer Baseboard to the computer through the USB Mini-B connector J13. Ensure that the correct module (PSoC 4 BLE or PRoC BLE) is placed on the baseboard, depending on the project opened.
8. Choose **Debug** > **Program** in PSoC Creator as shown in Figure 4-4.

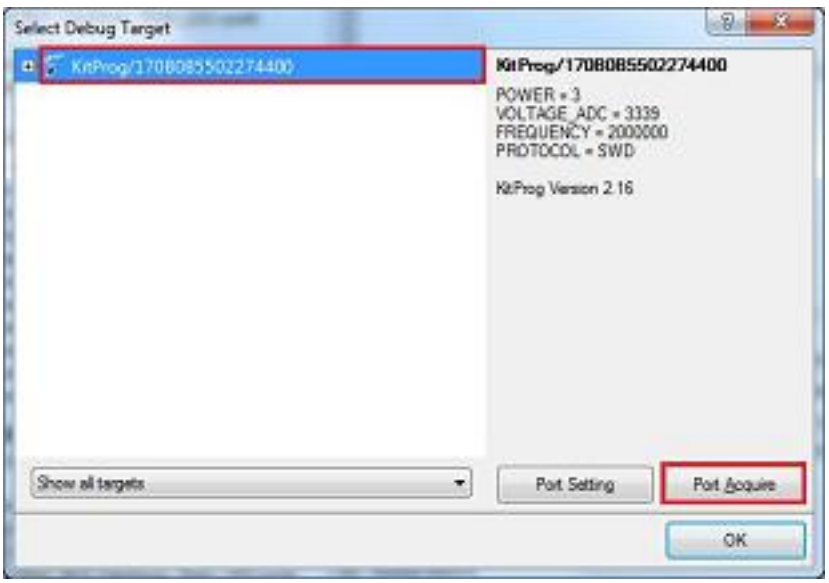
Figure 4-4. Program Device in PSoC Creator



9. If the device is not yet acquired, PSoC Creator will open the programming window. Select KitProg and click the **Port Acquire** button, as shown in Figure 4-5.

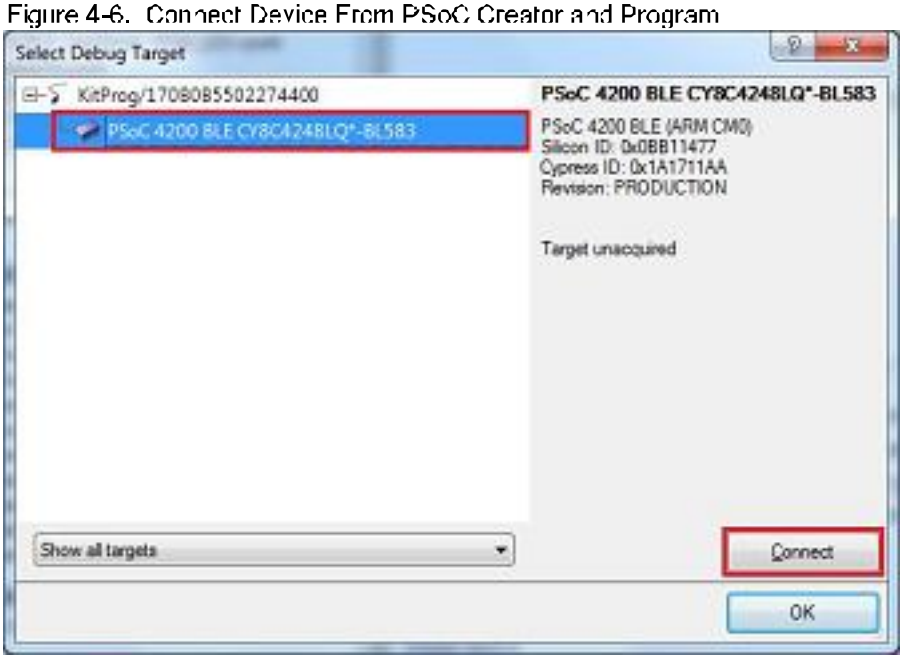
   Note: The serial ID starting with *'BLE'* belongs to the BLE Dongle (see Updating BLE Dongle for CySmart Central Emulation Tool on page 32).

Figure 4-5. Port Acquire



10. After the device is acquired, it is shown in a structure below the KitProg. Click the **Connect** button and then **OK** to exit the window and start programming, as shown in Figure 4-6.

Figure 4-6. Connect Device From PSoC Creator and Program



Note: As stated previously, the BLE Pioneer Kit supports both Cypress BLE devices: PSoC 4 BLE and PRoC BLE. The description, hardware configurations, and verification method of the example projects explained in the following sections are valid for both these devices. Unless explicitly mentioned, the theory and usability for these example projects should be considered the same for both the modules.

This document refers to the BLE Pioneer Kits, BLE Dongle, and PC/mobile as Central or Peripheral devices. A Central device is normally the master and requests/commands data from the Peripheral device. BLE-enabled phones and computers are one such example. Peripheral devices store the actual data and send it to the Central device when requested. Examples include BLE-enabled sensors, proximity beacons, and so on.

If you are a beginner in BLE, refer to the PSoC Creator code examples such as BLE_FindMe and BLE_Device_Information_Service as described in PSoC Creator Code Examples on page 16. You may also refer the application note Getting Started with PSoC 4 BLE.

The four KIT code examples viz. CapSense Slider and LED, CapSense Proximity, BLE Central Mode and Eddystone are intermediatory level examples that will help to design a system around the kit. Refer to the 4.2 CapSense Slider and LED and later for details.

The CySmart Dongle code example is an advanced level example that will demonstrate a complete solution around the kit. Refer to 4.6 BLE Dongle and LED Control for details.

4.2 CapSense Slider and LED

4.2.1 Project Description

This project demonstrates connectivity between the BLE Pioneer Kit (acting as a Peripheral and GATT server device) and CySmart Central Emulation tool or mobile device running the CySmart mobile application (acting as a Central and GATT client device). This project demonstrates the following:

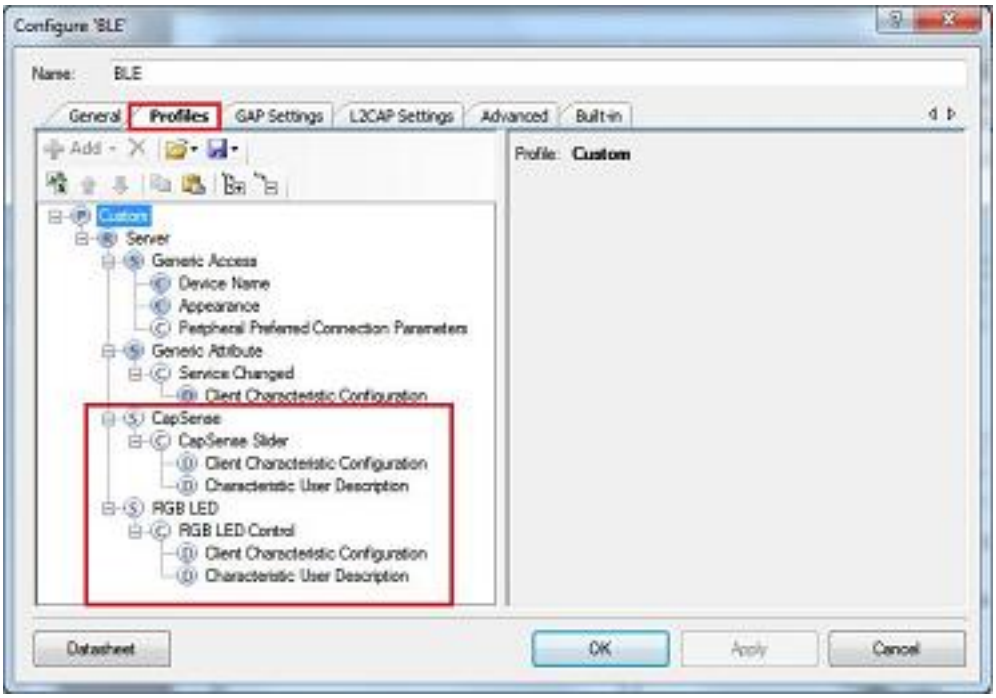
- Advertisement with timeout
- Connection with any Central device
- Two custom services in single profile
- Data transfer over BLE custom service using notifications, read, and write
- Low-power mode implementation for coin-cell operation

The BLE profile in this project consists of two BLE custom services: CapSense and RGB LED. The **CapSense service** consists of one custom characteristic, termed as CapSense Slider. The CapSense slider characteristic is used to send one byte data, ranging from 0 to 100, as notification to the GATT client device. This data is the finger location read by the CapSense Component on the five-segment slider (CSS1) present on the kit. This characteristics supports notification, which allows the GATT server to send data to the connected client device whenever new data is available.

The **RGB LED service** also consists of one custom characteristic, termed as RGB LED Control. This characteristic supports two operations, read and write, through which the connected GATT client device can read data as well as write a new value to the characteristic. This data has four byte values indicating red, green, blue, and the intensity values to control the onboard RGB LED.

The properties for the custom service/characteristics are configured in the BLE Component under the Profiles tab, as shown in Figure 4-7.

Figure 4-7. Attributes Configuration in BLE Component for Custom Services



The project consists of the following files:

- **main.c/.h**

  These files contain the main function, which is the entry point and execution of the firmware application. They also contain the function definition for initialization of the system and reading the CapSense slider data from the CapSense Component.

- **BLEApplications.c/.h**

  These files contain all the macros and function definitions related to BLE communication and operation. They include the event callback function definition that is registered with the BLE Component startup and used to send BLE-related events from the BLE stack to the application layer for processing. These files contain a method to send CapSense notifications to the GATT client device and process the Read and Write commands on the RGB LED characteristic by the GATT client device. They update the BLE Connection parameter, which is important for low-power mode usage.

- **HandleLowPower.c/.h**

  These files contain the function to handle low-power mode. This function is continuously called in the main loop and is responsible for pushing the BLE hardware block (BLESS) as well as the CPU to Deep Sleep mode as much as possible. The wakeup source is either the BLE hardware block link layer internal timer or the interrupt from the user button press (**SW2**). This allows for very low power mode implementation and operation using a coin cell.

Additionally, the PRoC BLE version of this project consists of the *RGB_PrISm.c/.h* file, which contains the function to drive the software-based PrISM method; it also drives the color and intensity on the RGB LED.

This is the default firmware that comes in the modules shipped with the kit.

Two projects demonstrate this functionality on two different devices.

- **PSoC_4_BLE_CapSense_Slider_LED** works with the PSoC 4 BLE Module.
- **PRoC_BLE_CapSense_Slider_LED** works with the PRoC BLE Module.

The PSoC 4 BLE project implements RGB color and intensity control using the PrISM Component; whereas the PRoC BLE uses the software implementation of the PrISM mode.

Figure 4-8. TopDesign for PSoC_4_BLE_CapSense_Slider_LED Project

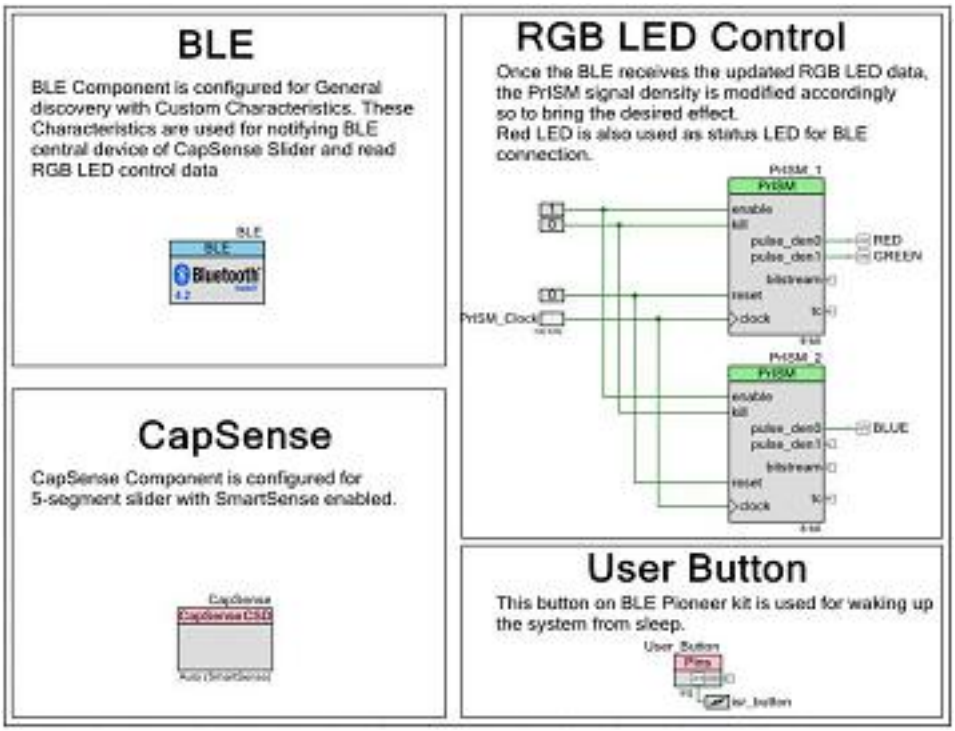
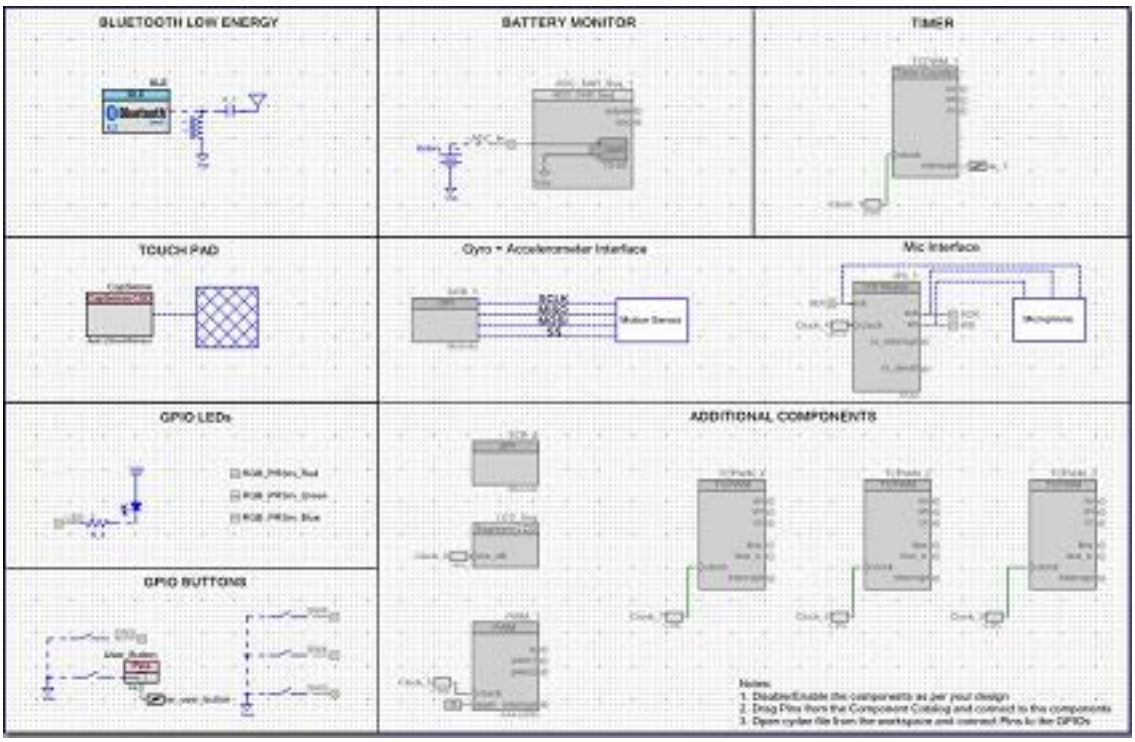


Figure 4-9. TopDesign for PRoC_BLE_CapSense_Slider_LED Project



4.2.2 Hardware Connections

No specific hardware connections are required for this project because all connections are hardwired on the BLE Pioneer Baseboard. Ensure that the correct module is placed on the baseboard corresponding to the project being used. PSoC_4_BLE_CapSense_Slider_LED works with the PSoC 4 BLE Module. PRoC_BLE_CapSense_Slider_LED works with the PRoC BLE Module.

The pin assignment for this project is in **PSoC_4_BLE_CapSense_Slider_LED.cydwr** **PRoC_BLE_CapSense_Slider_LED.cydwr** in the Workspace Explorer, as shown in Figure 4-10.

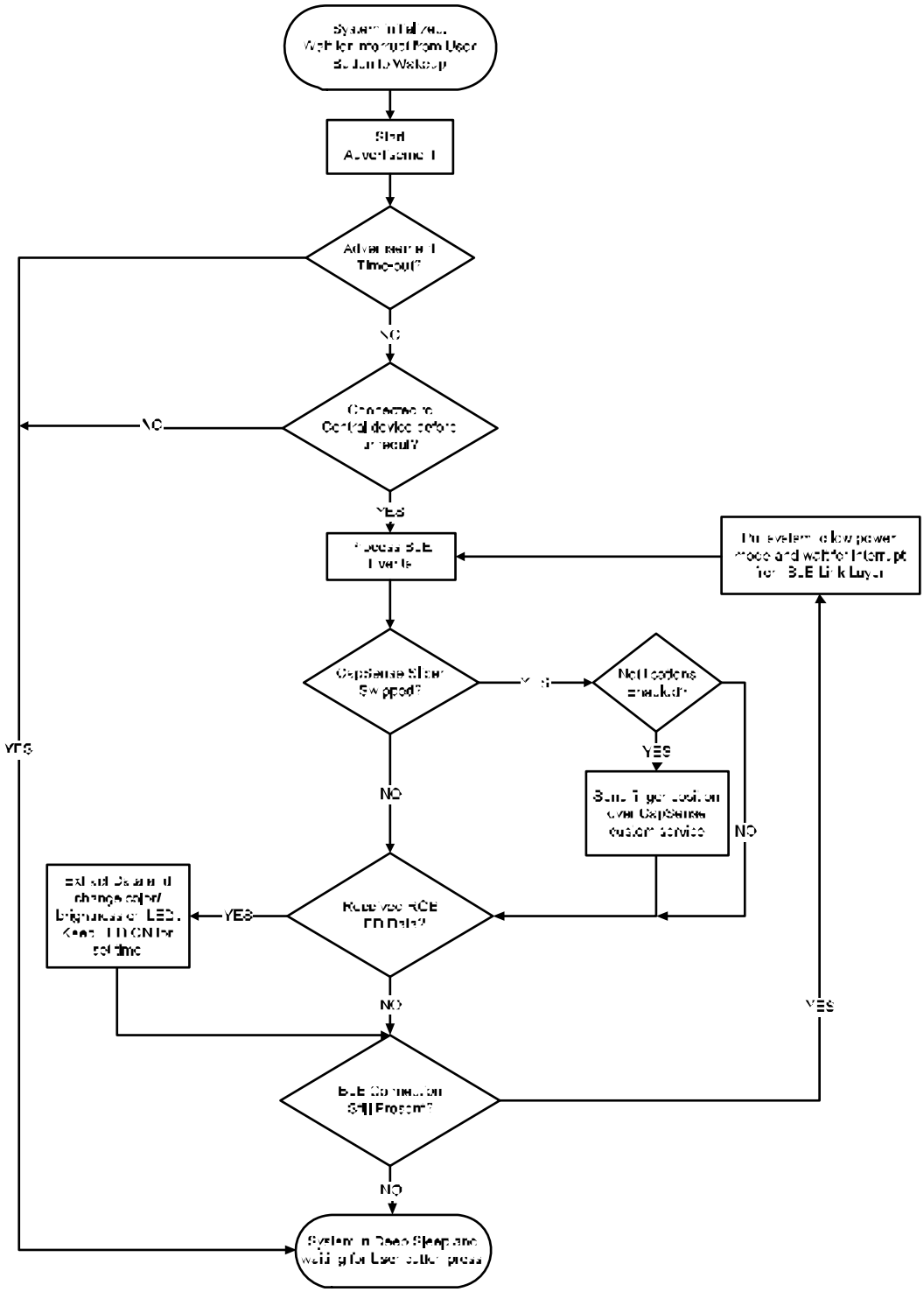
Figure 4-10. Pin Selection for CapSense Slider and LED Project

	Name	Port	Pin	Lock
	\CapSense:Cmod\ (Cmod)	P4[0]	5	
	\CapSense:Sns[0]\ (LinearSlider0_e0__LS)	P2[1]	38	
	\CapSense:Sns[1]\ (LinearSlider0_e1__LS)	P2[2]	39	
	\CapSense:Sns[2]\ (LinearSlider0_e2__LS)	P2[3]	40	
	\CapSense:Sns[3]\ (LinearSlider0_e3__LS)	P2[4]	41	
	\CapSense:Sns[4]\ (LinearSlider0_e4__LS)	P2[5]	42	
	BLUE	P3[7]	54	
	GREEN	P3[6]	53	
	RED	P2[6]	43	
	User_Button	P2[7]	44	

4.2.3 Flow Chart

Figure 4-11 shows the flow chart of the code implemented.

Figure 4-11. CapSense Slider and LED Project Flow Chart



4.2.4 Verify Output

The project can be verified by two methods: using the CySmart Central Emulation Tool and BLE Dongle or using the CySmart mobile application.

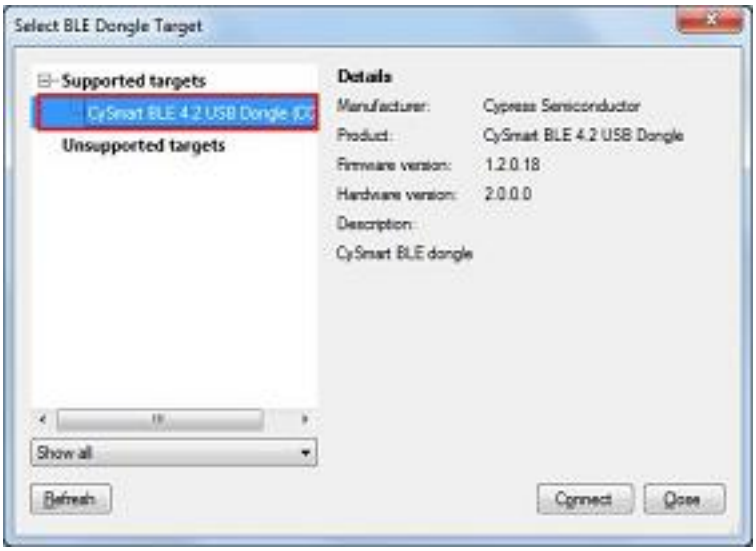
4.2.4.1 CySmart Central Emulation Tool

To verify the CapSense and LED project using the CySmart Central Emulation Tool, follow these steps:

Note: Refer CySmart Central Emulation tool to learn how to use the tool.

1. Connect the BLE Dongle to one of the USB ports on the computer.
2. Start the CySmart Central Emulation Tool on the computer by going to **Start** > **All Programs** > **Cypress** > **CySmart <version>** > **CySmart <version>**. You will see a list of BLE Dongles connected to it. If no dongle is found, click **Refresh**. Select the BLE Dongle and click **Connect**.

Figure 4-12. Connect to BLE Dongle



3. Place the module on the BLE Pioneer Kit, depending on the project chosen.
4. Power the BLE Pioneer Kit through the USB connector J13.
5. Program the BLE Pioneer Kit with the CapSense and LED example project. Follow steps in Using Example Projects on page 37 to program the device.
6. After programming successfully, press the user button (**SW2**) on the BLE Pioneer Kit to start the advertisement. Advertisement is indicated by a blinking red LED on the baseboard.

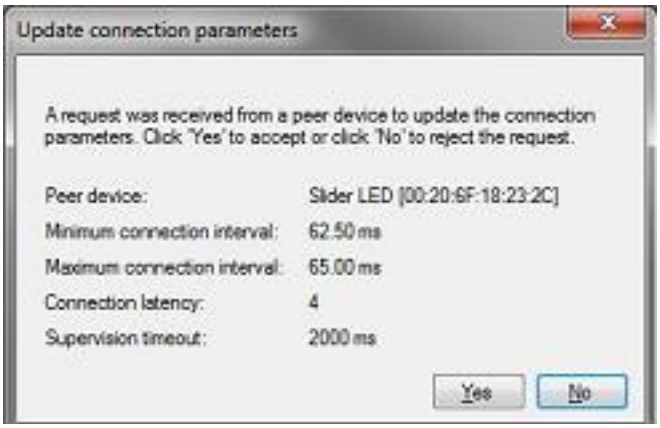
   Note: The project has an advertisement timeout of 30 seconds after which it returns to Deep Sleep mode. Press **SW2** again to restart the advertisement.
7. On the CySmart Central Emulation Tool, click **Start Scan** to see the list of available BLE Peripheral devices.

8. Double-click the **Slider LED** device to connect, or click **Slider LED** and then click **Connect**.

Figure 4-13. Connect to BLE Slider and LED Peripheral

9. When connected, the CySmart Central Emulation Tool will display a message for the **Update connection parameters**. Select **Yes**, as shown in Figure 4-14.

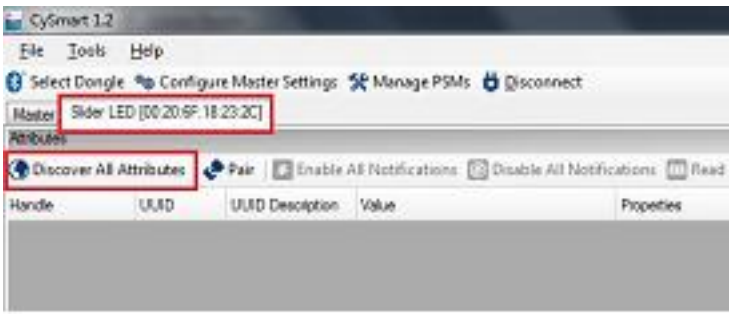
Figure 4-14. Update Connection Parameter Option



Note: If you select **No**, the project will still work. However, the current consumption will be higher due to faster connection interval.

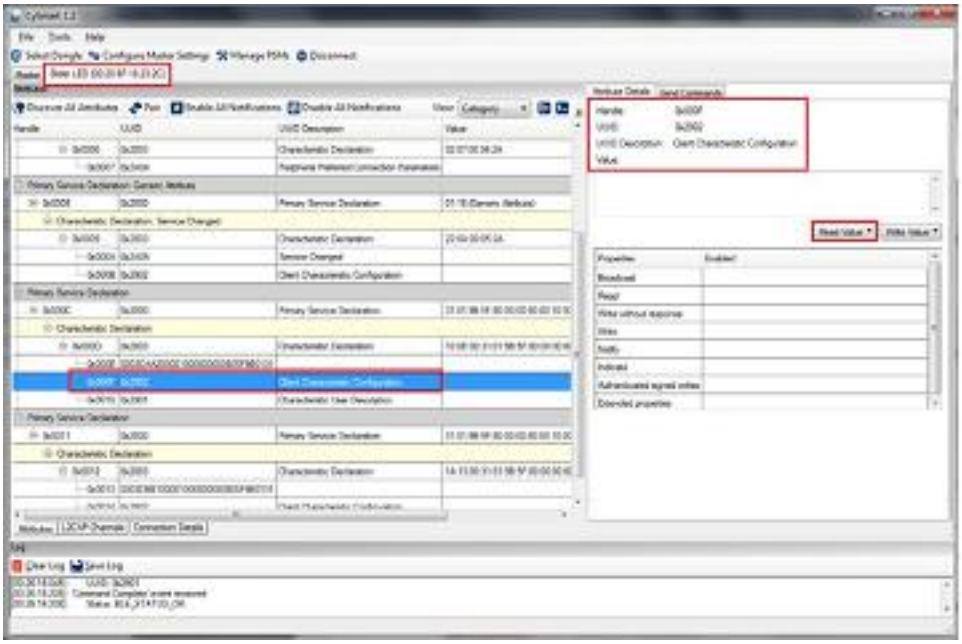
10. Click **Discover All Attributes** to find all attributes supported.

Figure 4-15. Discover All Attributes



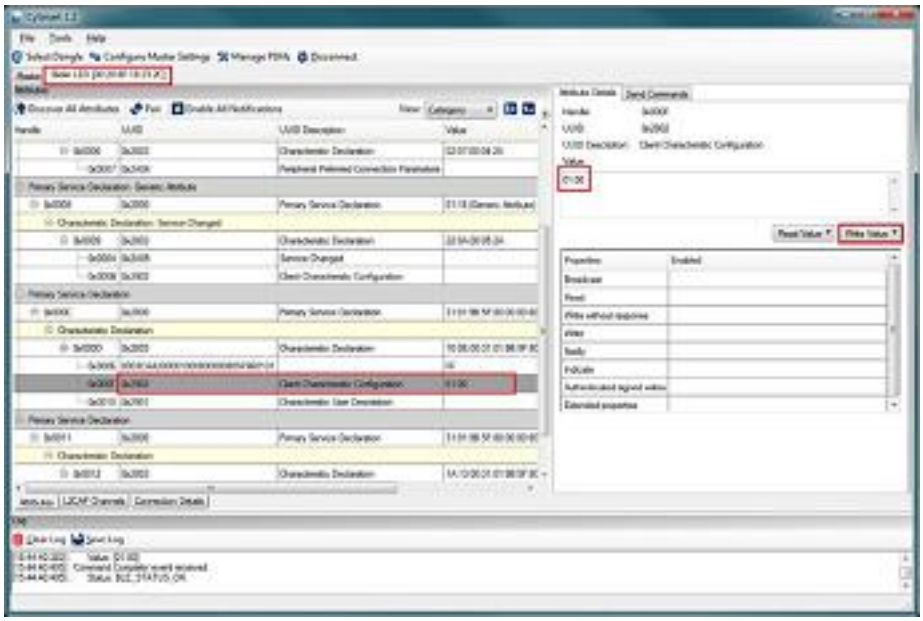
11. Locate the attribute **Client Characteristic Configuration** descriptor (UUID 0x2902) under CapSense slider characteristic (**UUID 0x0003CAA2-0000-1000-8000-00805F9B0131**). Click **Read Value** to read the existing Client Characteristic Configuration Descriptor (CCCD) value as shown in Figure 4-16.

Figure 4-16. Read CCCD for CapSense Slider Characteristic



12. Modify the **Value** field of CCCD to '01:00' and click **Write Value**. This enables the notifications on the CapSense slider characteristic.

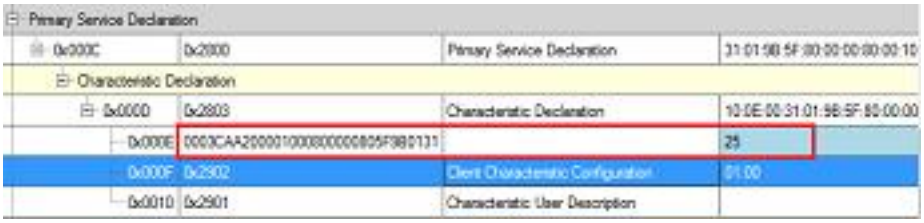
Figure 4-17. Write CCCD to Enable Notifications



13. Swipe your finger on the CapSense slider on the BLE Pioneer kit, as shown in Figure 4-18 and see the notification values in the CapSense Slider value field, as shown in Figure 4-19.

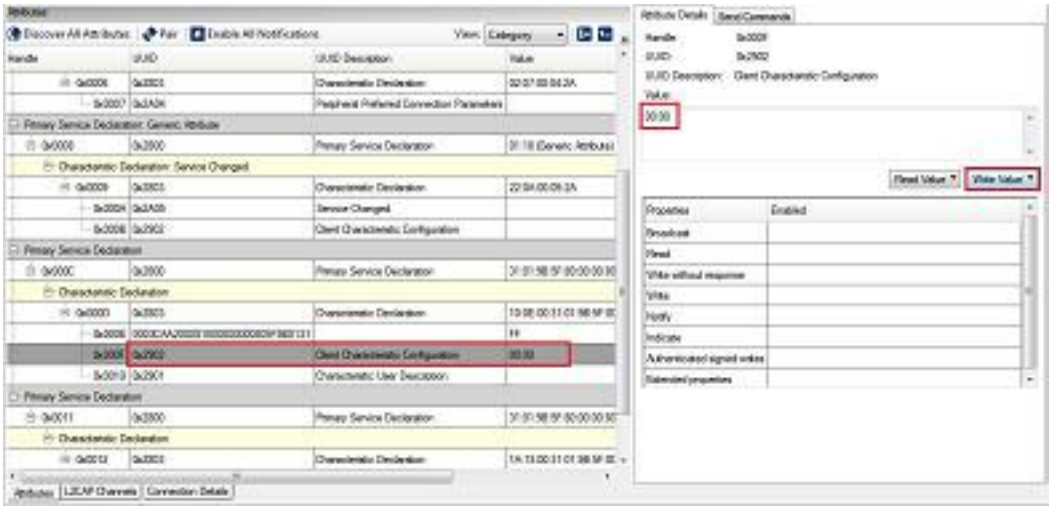
Figure 4-18. CapSense Slider

Figure 4-19. CapSense Slider Notification Received



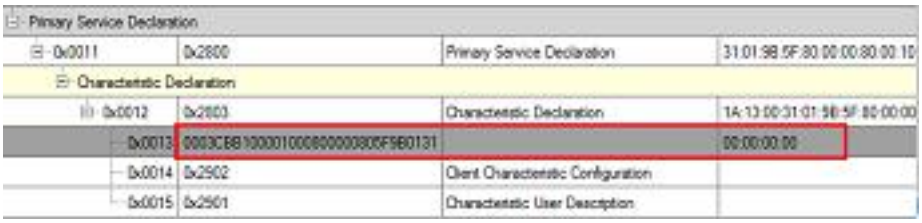
14. To disable notifications, modify the **Value** field of the **Client Characteristic Configuration** descriptor to '00:00' and click **Write Value**.

Figure 4-20. Disable Notifications



15. Locate the **RGB LED Control** characteristic (**UUID 0x0003CBB1-0000-1000-8000-00805F9B0131**). Click **Read Value** to read the existing 4-byte onboard RGB LED color information, as shown in Figure 4-21. The four bytes indicate red, green, blue, and the overall brightness, respectively.

Figure 4-21. Read RGB LED Control Characteristic Value



16. Modify the four bytes of data in the **Value** field and click **Write Value**. You will see the corresponding change in the color and intensity of the RGB LED on the BLE Pioneer Kit, as shown in Figure 4-22. The RGB LED will be on for 3 seconds before switching off to conserve power.

Note: If the kit is powered from a coin cell and not the USB Vbus, then the color mixing and intensity will vary. This is because the coin cell provides a lower driving voltage for RGB LEDs.

Figure 4-22. Write RGB LED Control Characteristic Value

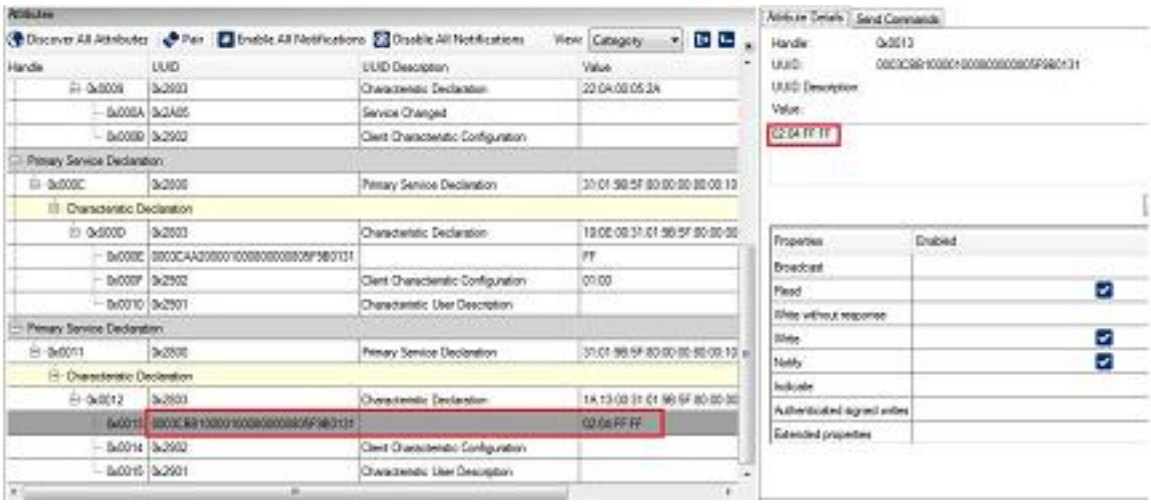
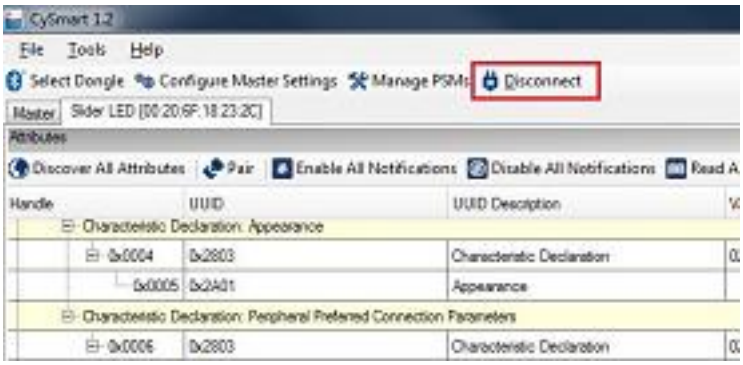


Figure 4-23. RGB LED Control with PSoC 4 BLE Module and PRoC BLE Module

17. To disconnect from the device, click **Disconnect**, as shown in Figure 4-24.

Figure 4-24. Disconnect from the Device



18. To connect to this peripheral again, restart advertising by pressing the user button (**SW2**) on the BLE Pioneer Kit. Advertising is indicated by the blinking red LED.

4.2.4.2 *CySmart Mobile Application*

To verify the CapSense and LED project using the CySmart mobile application (refer CySmart Mobile App webpage), follow these steps:

1. Plug the desired module on the BLE Pioneer Baseboard.
2. Connect the BLE Pioneer Kit to the computer using the J13 USB connector.
3. Program the kit with the CapSense Slider and LED example project. See Using Example Projects on page 37 for programming instructions.
4. Press the user button (**SW2**) on the BLE Pioneer Kit to start the advertisement. This is indicated by the blinking red LED on the BLE Pioneer Kit.
5. Open the application on the mobile device. If Bluetooth is not enabled on the device, the application will ask to enable it.
6. After Bluetooth is enabled, the CySmart mobile application will automatically search for available peripherals and list them. Select the **Slider LED** peripheral as shown in Figure 4-25.

Figure 4-25. Slider LED Peripheral

7. When connected, the CySmart mobile application will list the profiles supported by the peripherals. Scroll and select the CapSense Icon, as shown in Figure 4-26.

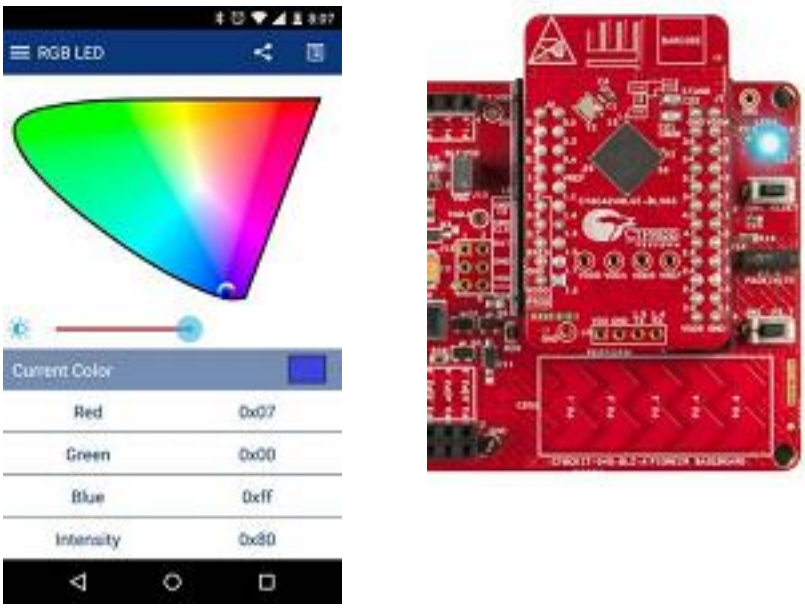
Figure 4-26. CapSense Service Page

8. Swipe your finger on the CapSense slider on the BLE Pioneer Kit and see a similar response on the **CapSense** page in the CySmart application (Figure 4-27).

Figure 4-27. CapSense Slider

9. Press the back button to return to the service selection page. Scroll and tap on the RGB LED service.
10. On the RGB LED service page, swipe over the color gamut to see a similar color response on the BLE Pioneer Kit RGB LED. The slider below the color gamut controls the intensity of the RGB LED color. The RGB LED will be on for 3 seconds before switching off. This is done to conserve power.

Figure 4-28. RGB LED Control with CySmart Mobile Application



11. To disconnect from the BLE Pioneer Kit, return to the CySmart mobile application home screen by pressing the back button.
12. To reconnect to the Peripheral, press the user button (**SW2**) on the BLE Pioneer Kit again and then scan for devices using CySmart mobile application.

4.3 CapSense Proximity

4.3.1 Project Description

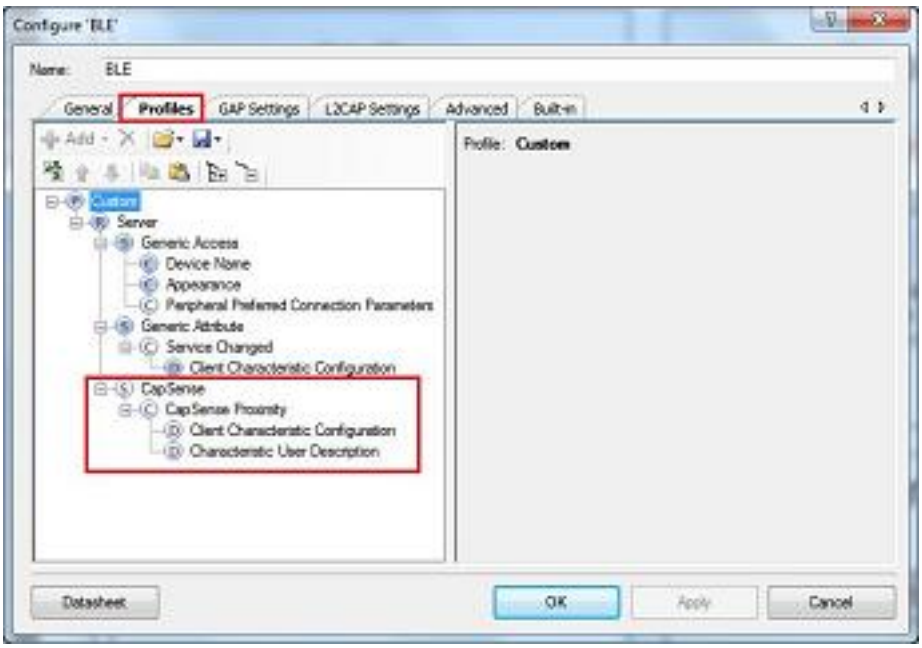
This project demonstrates connectivity between the BLE Pioneer Kit (acting as a Peripheral and GATT server device) and the CySmart Central Emulation Tool or mobile device running the CySmart mobile application (acting as a Central and GATT client device). This project demonstrates the following:

- Advertisement with the default
- Connection with any Central device
- One custom service
- Data transfer over BLE custom service using notifications
- Low-power mode implementation for coin cell operation

The BLE profile in this project consists of a single BLE custom service, called CapSense. The **CapSense service** consists of a custom characteristic, termed as **CapSense Proximity**. The CapSense proximity characteristic is used to send one byte data, ranging from 0 to 255, as notification to the GATT client device. This data is the difference count read by the CapSense Component on the one-wire proximity sensor (J14) connected on the kit. This characteristic supports notification, which allows the GATT server to send data to the connected GATT client device whenever new data is available.

The properties for the custom attributes are configured in the BLE Component under the Profiles tab, as shown in Figure 4-29.

Figure 4-29. Attributes Configuration in BLE Component for CapSense Proximity



The project consists the following files:

- **main.c/.h**

  These files contain the main function, which is the entry point and execution of the firmware application. They contain function definition for initialization of the system and reading the CapSense proximity data from the CapSense Component.

- **BLEApplications.c/.h**

  These files contain all the macros and function definitions related to BLE communication and operation. They include the event callback function definition that is registered with the BLE Component startup and used to send BLE-related events from the BLE stack to the application layer for processing. These files contain a method to send CapSense notifications to the GATT client device. They update the BLE Connection parameter, which is important for low-power mode usage.

- **HandleLowPower.c/.h**

  These files contain the function to handle low-power mode. This function is continuously called in the main loop and is responsible for putting the BLE hardware block (BLESS) as well as the CPU to Deep Sleep mode as much as possible. The wakeup source is either the BLE hardware block link layer internal timer or the interrupt from the user button press (**SW2**). This allows for very low-power mode implementation and operation using a coin cell.

The red LED is used as the status LED and provides visual confirmation of advertising or connection states. A blinking red LED indicates advertising state.

Two projects demonstrate this functionality on two different devices:

- **PSoC_4_BLE_CapSense_Proximity** works with the PSoC 4 BLE Module.
- **PRoC_BLE_CapSense_Proximity** works with the PRoC BLE Module.

Figure 4-30. Top Design for PSoC_4_BLE_CapSense_Proximity Project

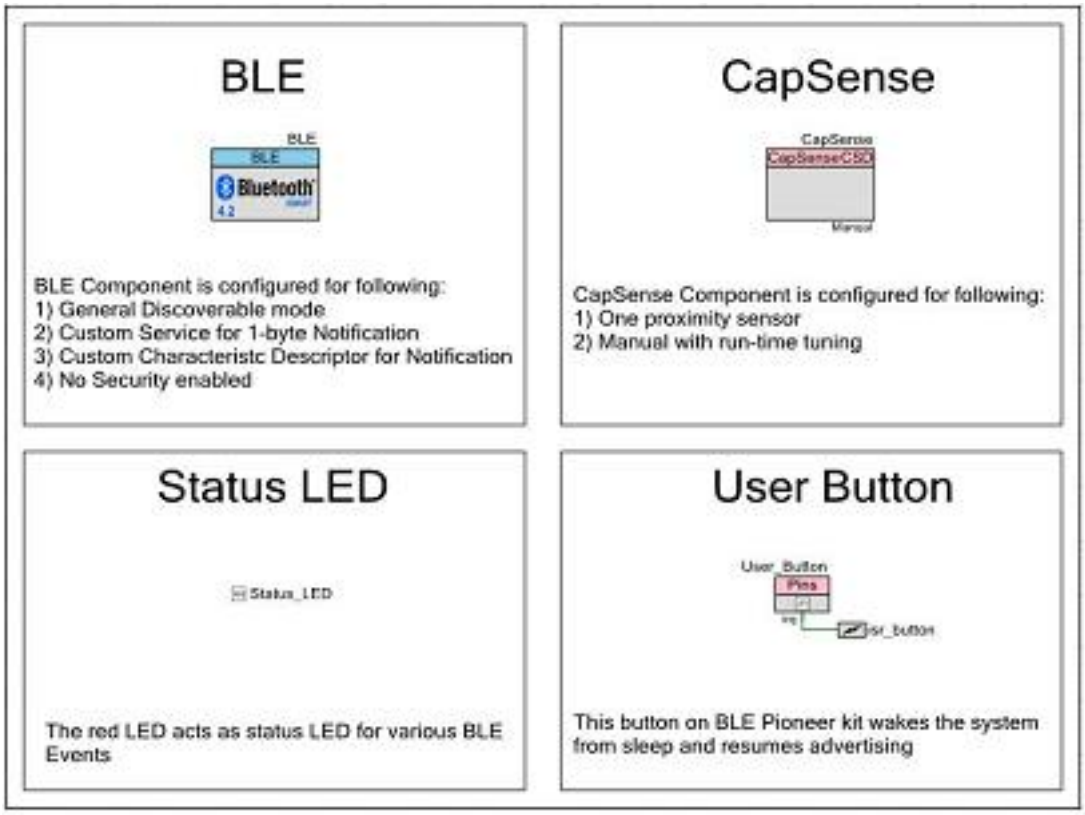
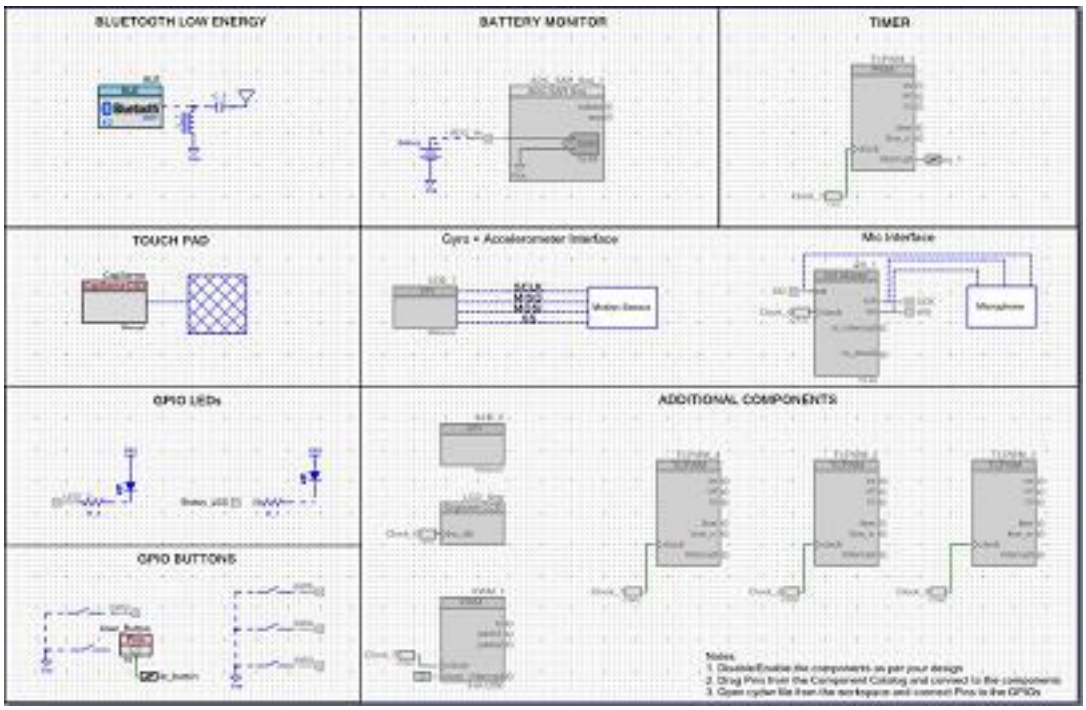


Figure 4-31. Top Design for PRoC_BLE_CapSense_Proximity Project



4.3.2 Hardware Connections

- Ensure that the correct module is placed on the baseboard corresponding to the project being used. PSoC_4_BLE_CapSense_Proximity works with the PSoC 4 BLE Module. PRoC_BLE_CapSense_Proximity works with the PRoC BLE Module.
- Connect a five-inch wire (provided as part of this kit) to the proximity connector J14 on the baseboard. Loop the wire as shown in Figure 4-32.

  The proximity range is approximately equal to the diameter of the proximity loop. For more information, refer to AN92239 – Proximity Sensing with CapSense.

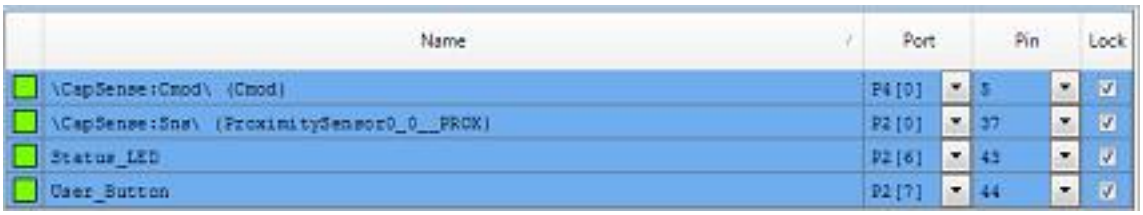
  Note: Ensure that the proximity sensor loop wire is kept away as much as possible from the BLE antenna on the modules.

Figure 4-32. Proximity Sensor Connection on BLE Pioneer Kit with PSoC 4 BLE Module

Figure 4-33. Proximity Sensor Connection on BLE Pioneer Kit with PRoC BLE Module

The pin assignment for this project is in **PSoC_4_BLE_CapSense_Proximity.cydwr** / **PRoC_BLE_CapSense_Proximity.cydwr** in the Workspace Explorer, as shown in Figure 4-34.

Figure 4-34. Pin Selection for CapSense Proximity Project

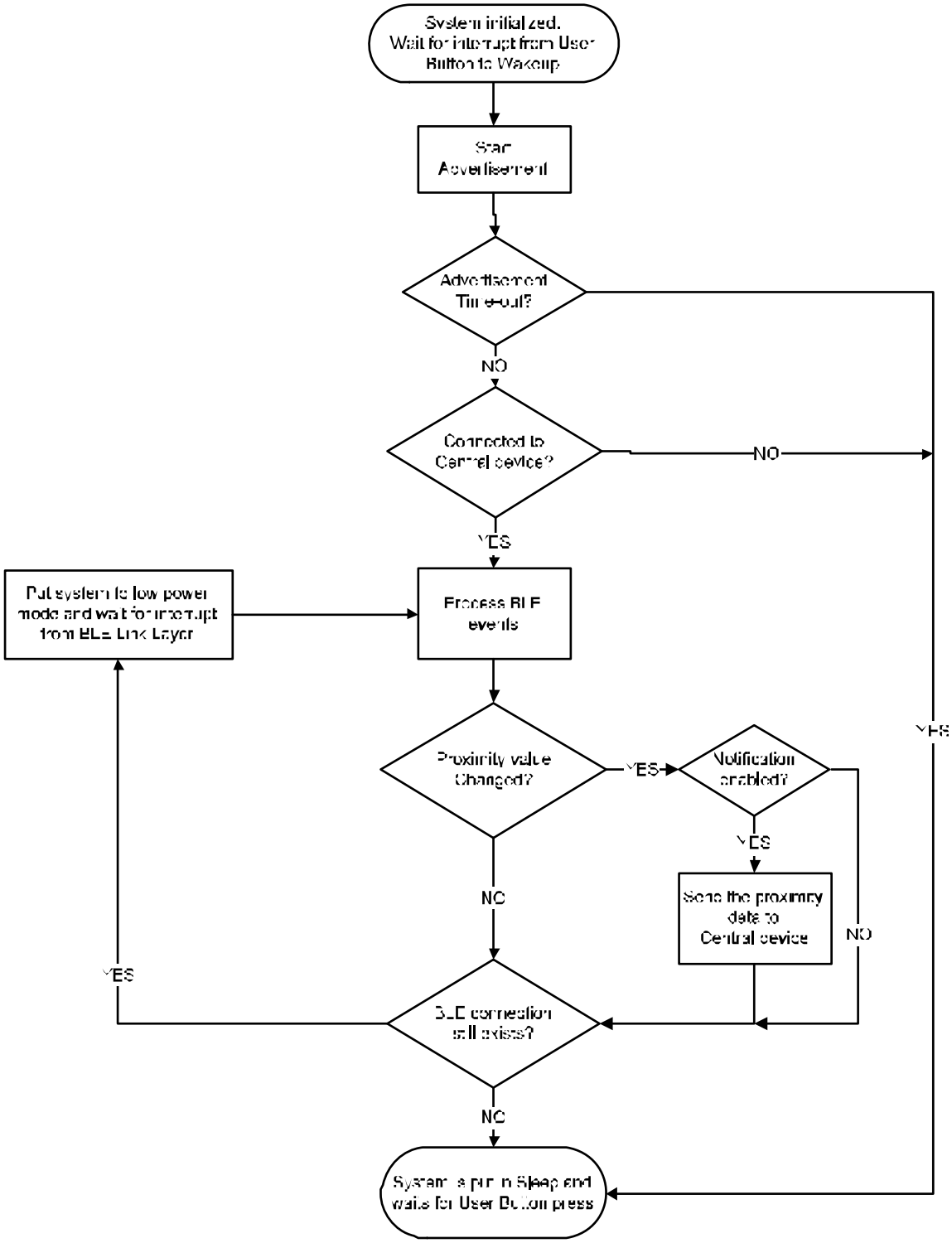


	Name	Port	Pin	Lock
	\CapSense:Cmod\ (Cmod)	P4[0]	5	☑
	\CapSense:Sns\ (ProximitySensor0_0__PROX)	P2[0]	37	☑
	Status_LED	P2[6]	43	☑
	User_Button	P2[7]	44	☑

4.3.3 Flow Chart

Figure 4-35 shows the flow chart of code implemented.

Figure 4-35. CapSense Proximity Project Flow Chart



4.3.4 Verify Output

The project can be verified by two methods: using the CySmart Central Emulation Tool and BLE Dongle or using the CySmart iOS/Android app.

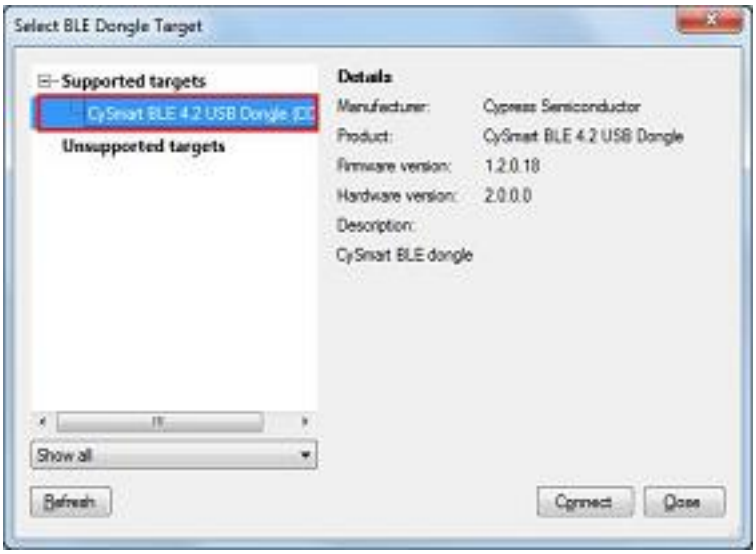
4.3.4.1 CySmart Central Emulation Tool

To verify the CapSense Proximity project using the CySmart Central Emulation Tool, follow these steps:

Note: Refer CySmart Central Emulation tool to learn how to use the tool.

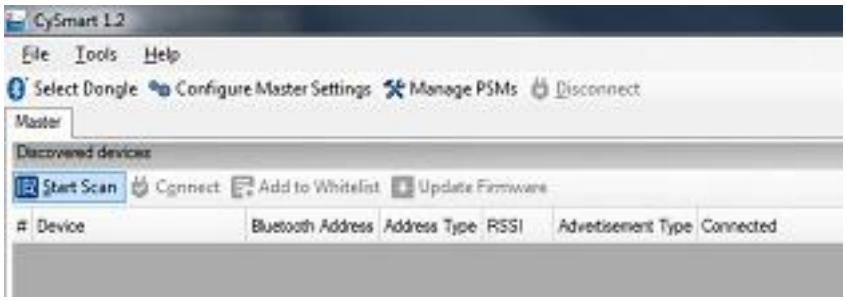
1. Connect the BLE Dongle to one of the USB ports on the computer.
2. Start the CySmart Central Emulation Tool on the computer by going to **Start > All Programs > Cypress > CySmart <version> > CySmart <version>**. You will see a list of dongles connected to it. If no dongle is found, click **Refresh**. Select the BLE Dongle and click **Connect**.

Figure 4-36. Connect to BLE Dongle



3. Connect a five-inch wire (included in the kit) to the proximity sensor connector J14 and make a loop of it.
4. Power the BLE Pioneer Kit through the USB connector J13.
5. Program the BLE Pioneer Kit with the CapSense proximity example project. Follow the steps in Using Example Projects on page 37 to program the device.
6. After programming successfully, press the user button (**SW2**) on the BLE Pioneer kit to start the advertisement. This is indicated by a blinking red LED on the baseboard.
   Note: The project has an advertisement timeout of 30 seconds after which it returns to Deep Sleep mode. Press **SW2** again to restart the advertisement.
7. On the CySmart Central Emulation Tool, click **Start Scan** to see the list of available BLE Peripheral devices.

Figure 4-37. Start Scanning

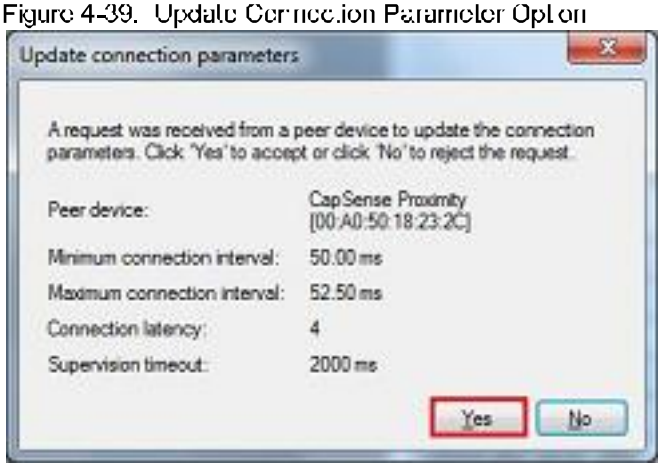


8. Double-click **CapSense Proximity** to connect or click **Connect** to connect to the BLE Pioneer Kit.

Figure 4-38. Connect to CapSense Proximity Peripheral

9. When connected, the CySmart Central Emulation Tool will display a message for the **Update connection parameters**. Select **Yes**, as shown in Figure 4-39.

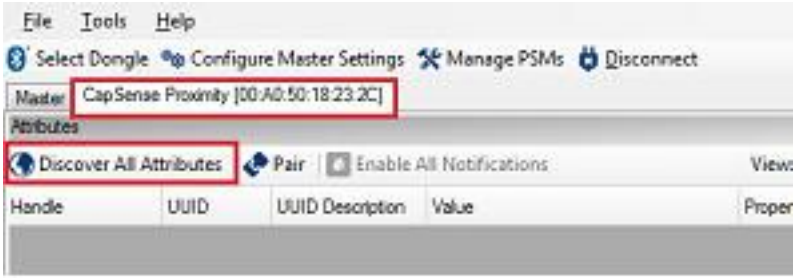
Figure 4-39. Update Connection Parameter Option



Note: If you select **No**, the project will still work. However, the current consumption will be higher due to faster connection interval.

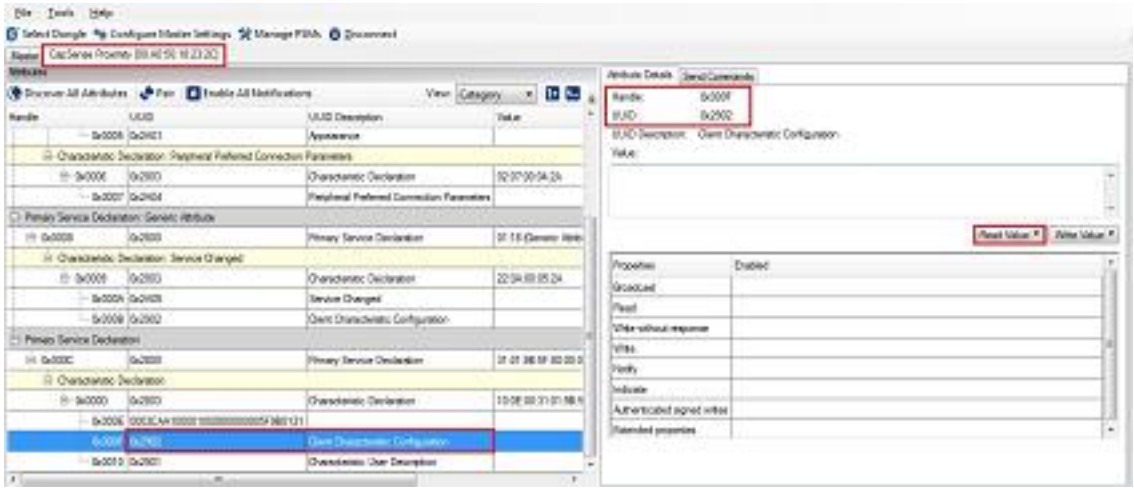
10. Click **Discover All Attributes** to find all attributes supported by the Peripheral.

Figure 4-40. Discover All Attributes



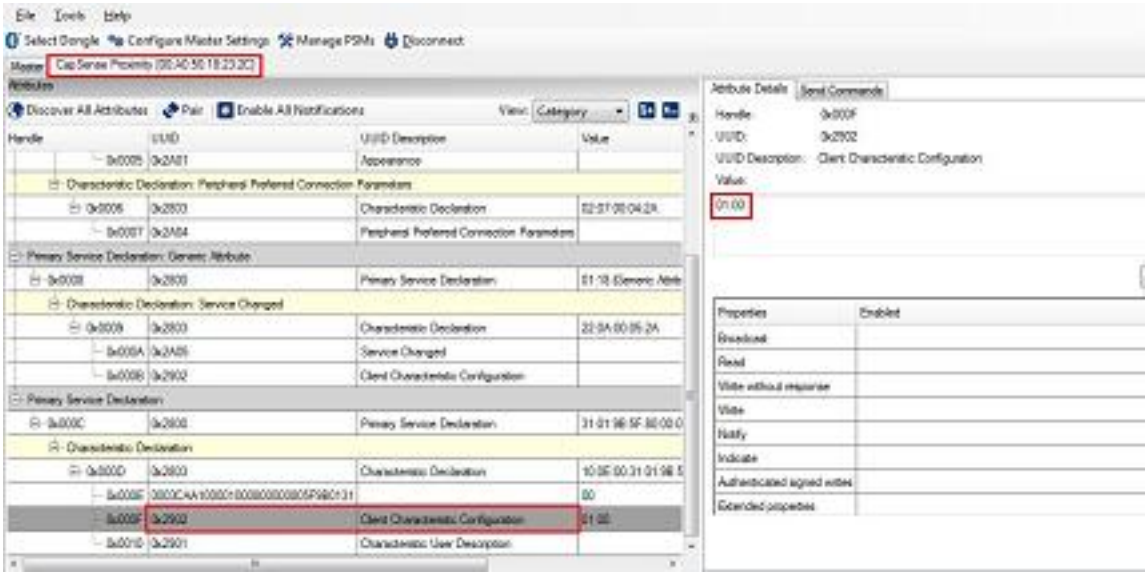
11. When all the attributes are listed, locate the **Client Characteristic Configuration** descriptor (UUID 0x2902) under CapSense Proximity characteristic (**UUID 0x0003CAA1-0000-1000-8000-00805F9B0131**). Click **Read Value** to read the existing CCCD value as shown in Figure 4-41.

Figure 4-41. Read CapSense Proximity CCCD



12. Modify the **Value** field to '01:00' and click **Write Value**. This enables the notifications on the CapSense proximity characteristic.

Figure 4-42. Write CCCD to Enable Notifications



13. Bring your hand closer to the proximity sensor on the BLE Pioneer Kit, as shown in Figure 4-43 and observe the value changing in the characteristic value field, as shown in Figure 4-44.

Figure 4-43. CapSense Proximity Sensing with PSoC 4 BLE Module

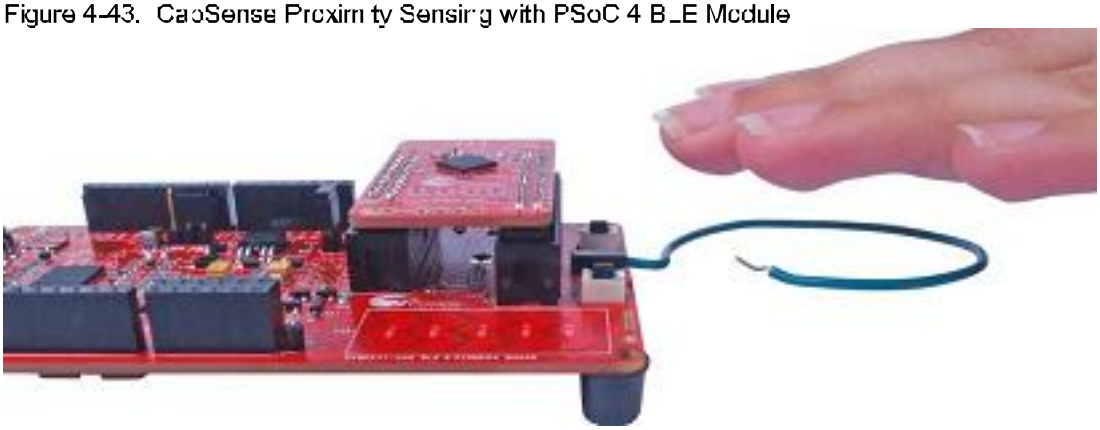
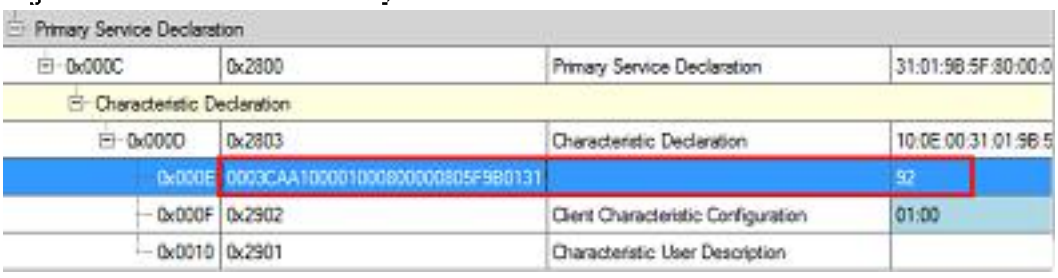


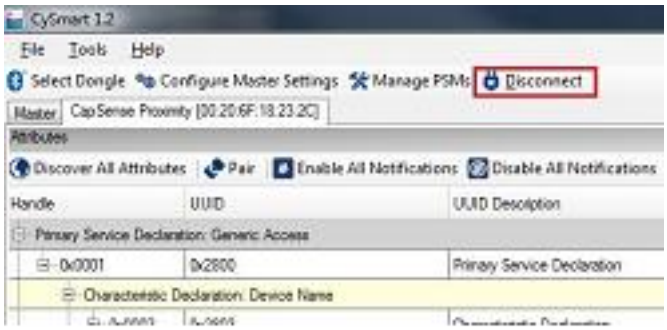
Figure 4-44. CapSense Proximity Notification Received



14. Modify the **Value** field of the Client Characteristic Configuration descriptor to '00:00' to disable notifications.

15. To disconnect from the device, click **Disconnect**, as shown in Figure 4-45.

Figure 4-45. Disconnect from the Device



16. Press user button (**SW2**) to wake up from sleep and restart the advertisement for the next connection.

4.3.4.2 *CySmart Mobile Application*

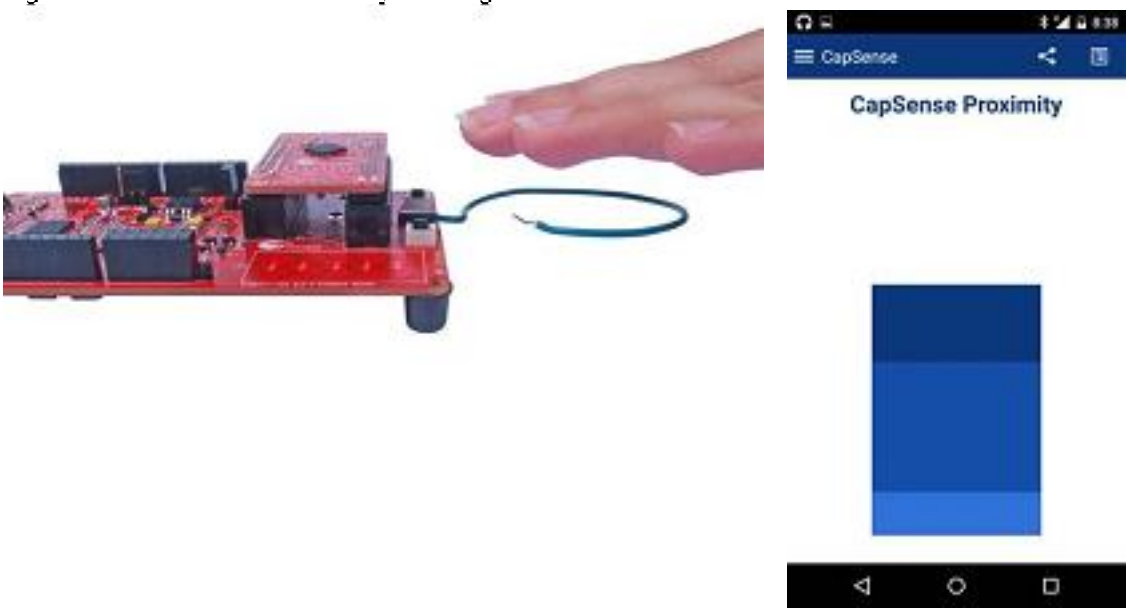
To verify the CapSense Proximity project using the CySmart mobile application (refer CySmart Mobile App webpage), follow these steps:

1. Place the desired module on BLE Pioneer Baseboard.
2. Connect the five-inch wire as a loop to the proximity connector J14 on the baseboard.
3. Plug the BLE Pioneer Kit into the computer using the J13 USB connector.
4. Program the kit with the CapSense proximity example project. Follow steps in Using Example Projects on page 37 to program the device.
5. Press the user button (**SW2**) on the BLE Pioneer Kit to start the advertisement.
6. Open the CySmart mobile application on the mobile device. If Bluetooth is not enabled on the device, the app will ask to enable it.
7. The app will automatically search for available peripherals and list them. Select the **CapSense Proximity** peripheral, as shown in Figure 4-46.

Figure 4-46. Connect to CapSense Proximity Peripheral

8. When connected, the app will list the services supported by the peripherals. Scroll and select the **CapSense Proximity** service.
9. When the CapSense service page opens, bring your hand near the sensor wire on the BLE Pioneer Kit; the proximity level is represented in the app as a bar graph, as shown in Figure 4-47. The darker shade in the graph indicates closer proximity. The app beeps when the graph fills up by 50 percent. Refer to CySmart iOS App User Guide for details.

Figure 4-47. CapSense Proximity Sensing with PSoC 4 BLE Module



10. To disconnect from the BLE Pioneer Kit, return to the device selection screen on the CySmart mobile application.
11. To reconnect to the Peripheral, press the user button (**SW2**) on the BLE Pioneer Kit to restart the advertisement and scan for the device in the CySmart mobile application.

4.4 BLE Central Mode

4.4.1 Project Description

The BLE projects described above have been functioning as Peripheral devices. This means that the firmware role was set to be a Peripheral and GATT server; another device such as the CySmart Central Emulation Tool or CySmart mobile application will connect to it and collect the data.

This example project demonstrates the Central and GATT client mode where it will scan for a Peripheral device, connect to it, and send commands. In this project, the BLE Pioneer Kit scans and auto-connects to a particular Peripheral device supporting **Immediate Alert Service (IAS)**. Whenever the Peripheral with a predetermined public address is found, a connection request is sent followed by discovering the attributes. When the discovery is over, you can send one of the three alert levels to the Peripheral device over the IAS. This is done by pressing the **SW2** button on the BLE Pioneer Kit and cycling through the alert levels.

The BLE Central project supports low-power mode operation, where the firmware supports BLE Hardware block and CPU Deep Sleep mode whenever possible. The system remains in Deep Sleep when disconnected. Press **SW2** to wake up the system and start scanning (blinking blue LED). The scanning timeout interval is set to 30 seconds. If a particular Peripheral device is found advertising

before timeout, a connection is made (blue LED always ON). If no such device is found, then the system stops scanning and returns to Deep Sleep mode (LED OFF). Press **SW2** again to wake the system and restart scanning.

To aid in evaluation, the Peripheral project with the particular public address is provided in the same workspace. This Peripheral project supports IAS and has fixed public address that the Central device will recognize and auto-connect to. The project should be programmed on the BLE Dongle and powered through the USB port of the computer. The received alert levels (No, Mid, and High alert) on the BLE Dongle are represented by different LED status. No Alert is represented by LED OFF. Mid Alert by blinking LED, and High Alert with LED always ON. Upon each successive button press on the BLE Pioneer Kit, the LED state on the BLE Dongle changes in a circular fashion.

Two projects demonstrate the BLE Central functionality on the two devices:

- **PSoC_4_BLE_Central_IAS** works with the PSoC 4 BLE Module.
- **PRoC_BLE_Central_IAS** works with the PRoC BLE Module.

Additionally, the **BLE_Dongle_Peripheral_IAS** project is to be programmed on the BLE Dongle. This project is present in both the PSoC_4_BLE_Central_IAS and the PRoC_BLE_Central_IAS workspace and can be used to program the BLE Dongle separately.

Note: If the BLE Dongle is programmed with the **BLE_Dongle_Peripheral_IAS** example, it will not work with the CySmart PC utility. Reprogram the BLE Dongle with the CySmart firmware according to Updating BLE Dongle for CySmart Central Emulation Tool on page 32 to use the CySmart Central Emulation Tool.

Figure 4-48. PSoC_4_BLE_Central_IAS TopDesign

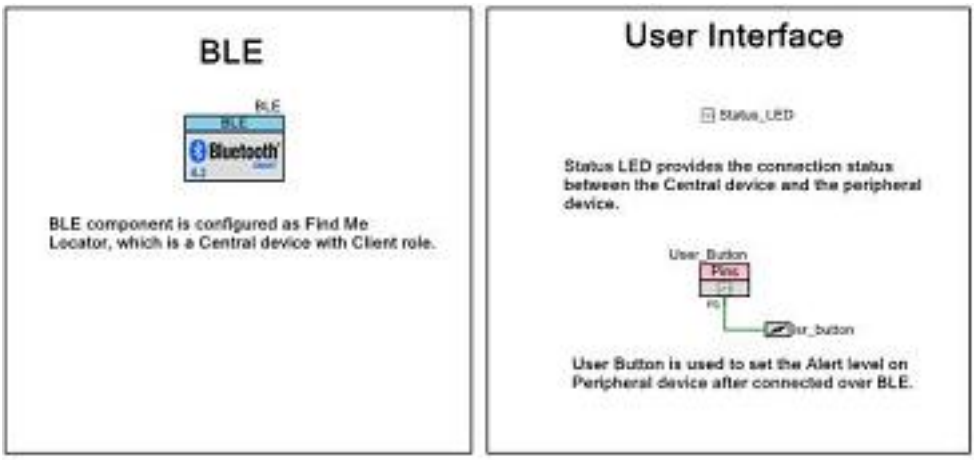


Figure 4-49. PRoC_BLE_Central_IAS TopDesign

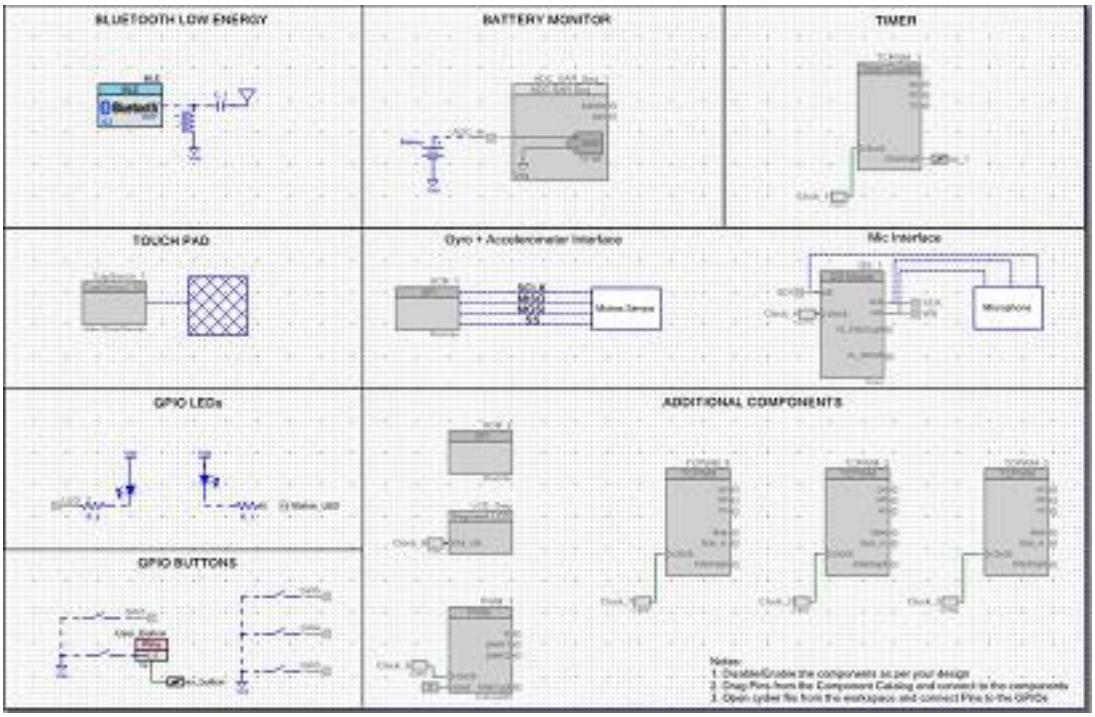
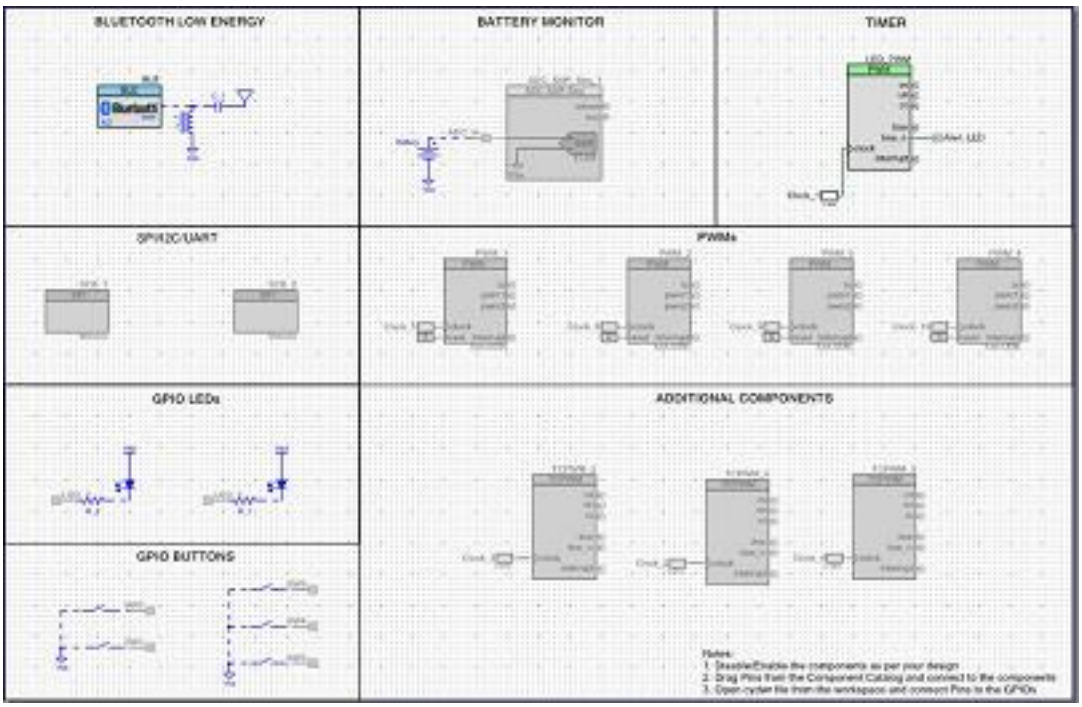


Figure 4-50. BLE_Dongle_Peripheral_IAS TopDesign



4.4.2 Hardware Connections

No specific hardware connections are required for this project because all connections are hardwired on the BLE Pioneer Baseboard.

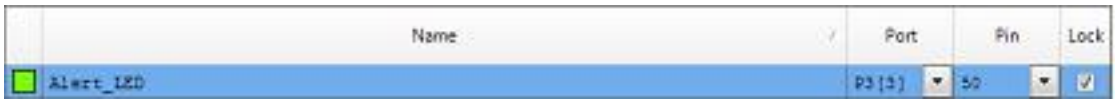
Ensure that the correct module is placed on the BLE Pioneer Baseboard corresponding to the project being used. PSoC_4_BLE_Central_IAS works with the PSoC 4 BLE Module. PRoC_BLE_Central_IAS works with the PRoC BLE Module. BLE_Dongle_Peripheral_IAS is the common project for both workspaces and programs the BLE Dongle with Peripheral mode firmware.

The pin assignment for this project is in **PSoC_4_BLE_Central_IAS.cydwr/ PRoC_BLE_Central_IAS.cydwr** in the Workspace Explorer, as shown in Figure 4-51.

Figure 4-51. Pin Selection for BLE IAS Central Example Project

Similarly, the pin assignment for the BLE Dongle Peripheral project is in **BLE_Dongle_Peripheral_IAS.cydwr** in the Workspace Explorer as shown in Figure 4-52.

Figure 4-52. Pin Selection for BLE IAS Peripheral Example Project



4.4.3 Flow Chart

Figure 4-53 shows the flow chart for the IAS GATT client mode example project.

Figure 4-53. IAS GATT Client Mode Flow Chart

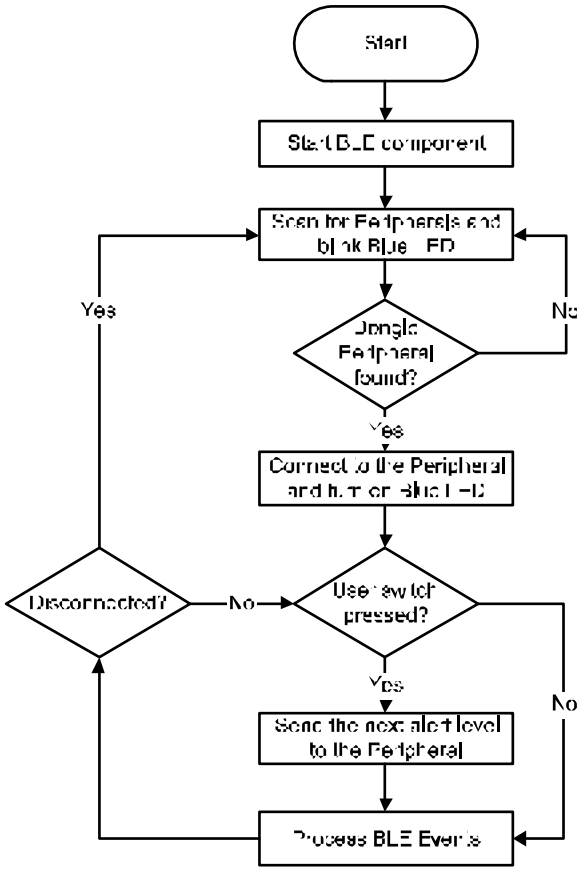
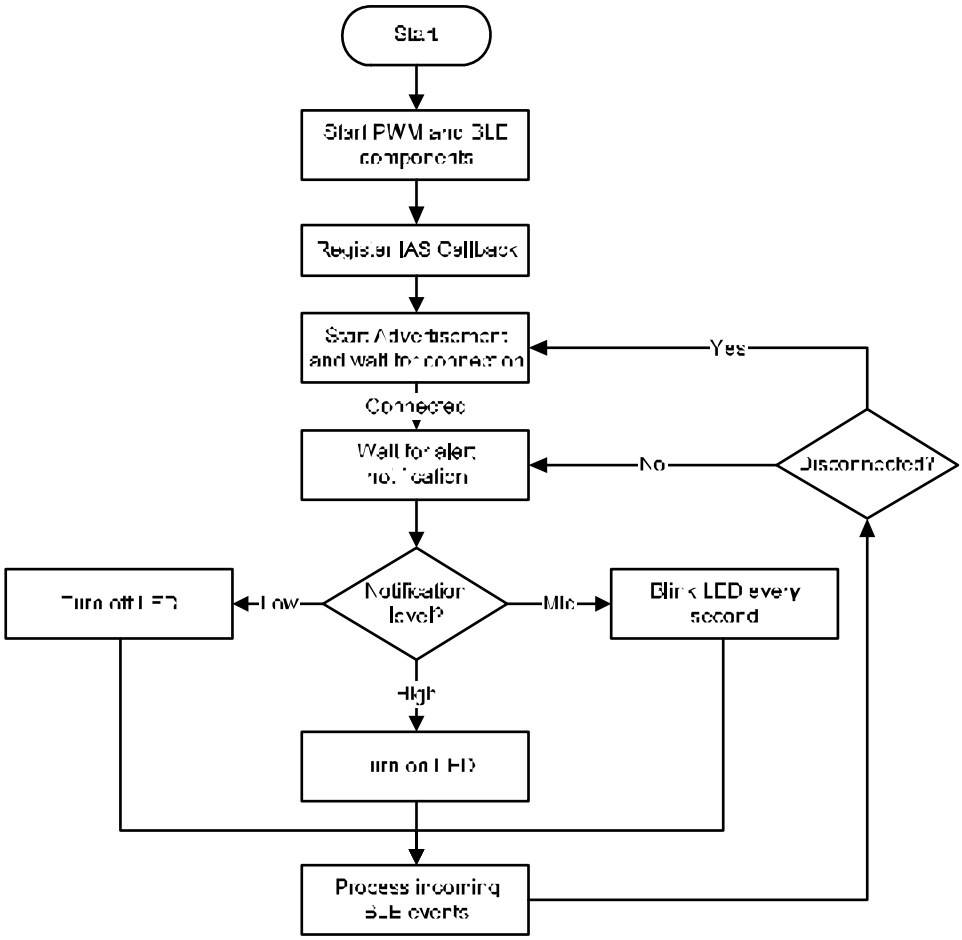


Figure 4-54 shows the flow chart for the IAS GATT server mode example project.

Figure 4-54. IAS GATT Server Mode Flow Chart



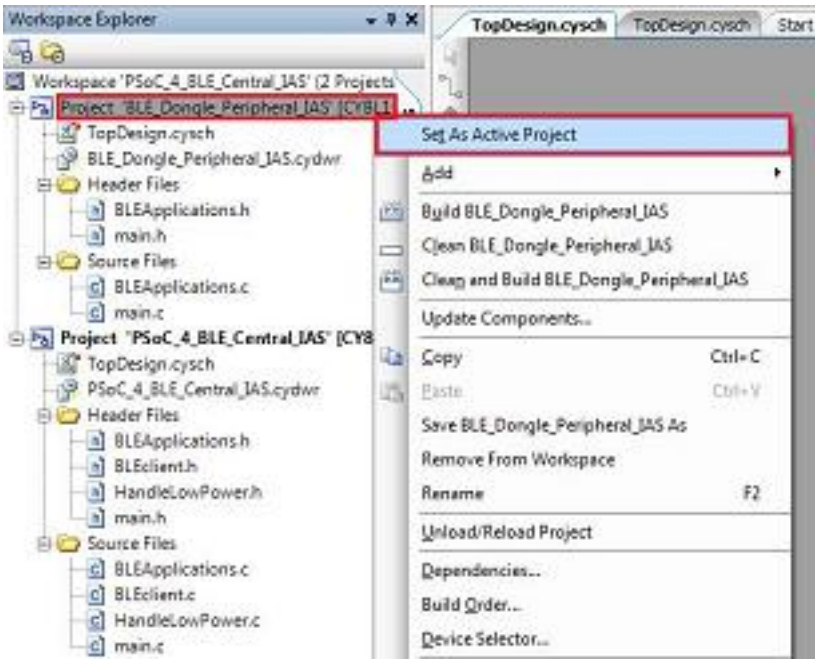
4.4.4 Verify Output

1. Connect the BLE Dongle to one of the USB ports on the computer.

Figure 4-55. Connect BLE Dongle to USB Port

2. In the PSoC Creator Workspace Explorer, right-click the **BLE_Dongle_Peripheral_IAS** project and select **Set As Active Project**, as shown in Figure 4-56.

Figure 4-56. Set Dongle Peripheral Project as Active

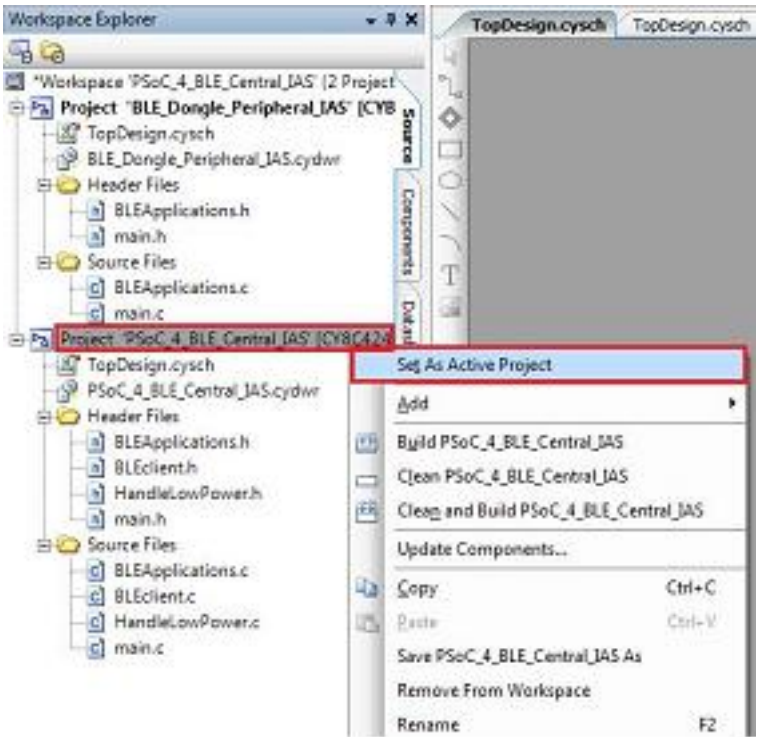


3. Program the BLE Dongle with the BLE_Dongle_Peripheral_IAS project described in Using Example Projects on page 37.

   Note: Do not update the public device address (inside the BLE Component) for the **BLE_Dongle_Peripheral_IAS** example project. Changing the **BLE_Dongle_Peripheral_IAS** example project public address will lead to no connection with the BLE Central device on the BLE Pioneer kit.

4. Power the BLE Pioneer Kit through USB connector J13.
5. In the Workspace Explorer, right-click the **PSoC_4_BLE_Central_IAS** project and select **Set As Active Project**, as shown in Figure 4-57.

Figure 4-57. Set Central IAS Project as Active



6. Program the BLE Pioneer Kit with either the **PSoC_4_BLE_Central_IAS** or the **PRoC_BLE_Central_IAS** project, depending on the module placed on the BLE Pioneer Kit.
7. Press the **SW2** button on the BLE Pioneer Kit to wake the system and start scanning. Scanning is indicated by a blinking LED.
8. Wait for the BLE connection between the BLE Dongle and the BLE Pioneer Kit. The connection success status is indicated on the baseboard in the following three stages:
   a. Fast blinking blue LED represents scanning mode. During this mode, the BLE Pioneer Kit is scanning for Peripheral devices.
   b. Slow blinking blue LED represents discovery mode. During this mode, the BLE Pioneer Kit has found the BLE Dongle Peripheral device and has started the connection procedure.
   c. The blue LED remains on, representing the connected mode. This mode indicates that the Peripheral device is connected and the application can now send alert levels.
9. Press the **SW2** button on the BLE Pioneer Kit to send the next alert level to the BLE Dongle. The alert level will rotate from No Alert to Mid Alert to High Alert.

Figure 4-68. User Button on BLE Pioneer Kit with PSoC 4 BLE Module

10. Check if the LED behavior changes for each alert notification on the BLE Dongle according to the following table:

Alert Level	LED State
No Alert	LED Off
Mild Alert	LED Blinking
High Alert	LED On

Note: To revert the CySmart functionality to the dongle, program the dongle hex file, as described in Updating BLE Dongle for CySmart Central Emulation Tool on page 32.

4.5 Eddystone

4.5.1 Project Description

This project demonstrates a BLE beacon based on Google's Eddystone™ protocol on the BLE Pioneer Kit. A beacon is a wireless device that broadcasts data (such as temperature) over a periodic radio signal from a known location. BLE-based beacons use the BLE advertisement packets to broadcast data. This project demonstrates the following:

- Beacon (non-connectable advertisement)
- Beacon configuration (connectable advertisement)

Google's Eddystone is a protocol that defines BLE advertisement data formats for beacons. In Eddystone parlance, the advertisement data is called a frame and the advertisement data format is called a frame type. The protocol supports multiple frame types that may be used individually or in combinations to create beacons for a variety of applications.

Figure 4-69. Characteristic Configuration in BLE Component for Eddystone Beacon

The BLE profile in this project consists of two BLE custom services, **Eddystone** and **Eddystone Configuration** (see Figure 4-59). The project broadcasts one of the following beacon frames in a non-connectable advertisement packet.

- **Eddystone-UID** frame broadcasts a unique 16-byte Beacon ID composed of a 10-byte namespace and a 6-byte instance. The Beacon ID may be useful in mapping a device to a record in external storage. The namespace portion of the ID may be used to group a particular set of beacons, while the instance ID identifies individual devices in the group. The division of the ID into namespace and instance components may also be used to optimize BLE scanning strategies, for example, by filtering only on the namespace. More details are available here.
- **Eddystone-URL** frame broadcasts a URL using a compressed encoding format to fit more within the limited advertisement packet. More details are available here.
- **Eddystone-TLM** frame broadcasts telemetry information about the beacon itself such as battery voltage, device temperature and counts of broadcast packets. More details are available here.

The Eddystone Configuration Service consists of nine characteristics as follows.

- **Lock State** – Read returns true if the device is locked.
- **Lock** – Locks the beacon and sets the single-use lock-code.
- **Unlock** – Unlocks the beacon and clears the single-use lock-code.
- **URI Data** – Reads/writes the URI.
- **URI Flags** – Reads/writes the flags.
- **Advertised Tx Power Levels** – Reads/writes the Advertised Power Levels array.
- **Tx Power Mode** – Reads writes the TX Power Mode.
- **Beacon Period** – The period in milliseconds that an Eddystone-URL packet is transmitted.
- **Reset** – Resets to default values.

More details on the Eddystone Configuration service can be found here.

The project consists of the following files:

- **main.c/.h**

  These files contain the main function, which is the entry point and execution of the firmware application. They contain function definitions for initialization and handling low power mode of the system. By default on power-on-reset, the system will do a non-connectable advertisement of URL/URI and TLM packets. When the **SW2** button is pressed, the device will enter connectable mode and various characteristics can be configured.

- **Eddystone.c/.h**

  These files contain the functions required to implement the Eddystone protocol including advertisement packet creation (for UID/URL and TLM), advertisement scheduling, and configuration read write. The battery and temperature data are also updated before the TLM packets are constructed and advertised.

- **WatchdogTimer.c/.h**

  These files contain functions that provide necessary timing for the operation of the system.

- **Battery.c/.h**

  These files contain a function that will measure the battery voltage of the kit. This information is advertised as part of the TLM packets.

- **Temperature.c/.h**

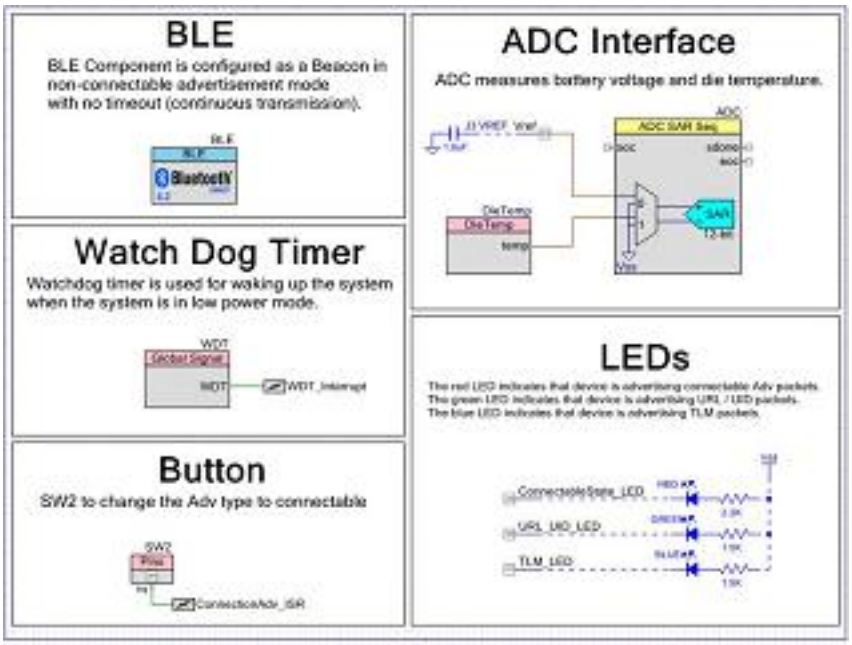
  These files contain a function that will measure the die temperature of the PSoC or PRoC device. An external sensor may be attached to measure ambient temperature. This information is advertised as part of the TLM packets.

The green LED indicates UID URL packet transmissions, the blue LED indicates TLM packet transmissions, and the red LED indicates that the device is in connectable mode. In this mode, when connected, the beacon configuration can be modified.

Two projects demonstrate this functionality on two different devices:

- PSoC_4_BLE_Eddystone works with the PSoC 4 BLE Module
- PRoC_BLE_Eddystone works with the PRoC BLE Module

Figure 4-60. Top Design for PSoC_4_BLE_Eddystone and PRoC_BLE_Eddystone Project



4.5.2 Hardware Connection

- Ensure that the correct module is placed on the baseboard corresponding to the project being used. PSoC_4_BLE_Eddystone works with the PSoC 4 BLE Module. PRoC_BLE_Eddystone works with the PRoC BLE Module.
- Connect a wire (provided as part of this kit) between VREF (connector J3) and P3.0 (connector J2) on the baseboard (see Figure 4-61).

Figure 4-61. VREF (J3) and P3.0 (J2) Connectors on BLE Pioneer Kit with PSoC 4 BLE Module

The pin assignment for this project is in *PSoC_4_BLE_Eddystone.cydwr* or *PRoC_BLE_Eddystone.cydwr* in the Workspace Explorer, as shown in Figure 4-62.

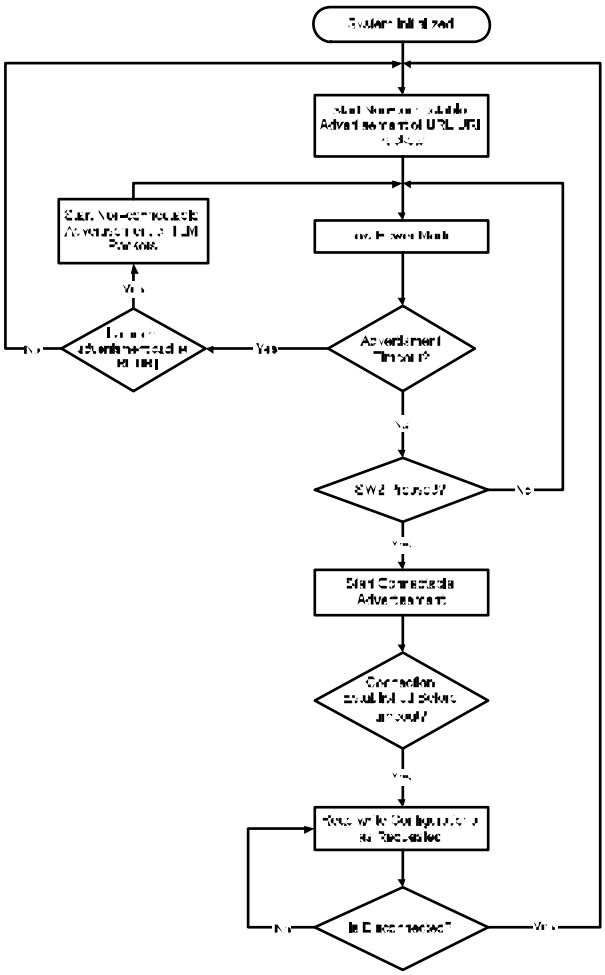
Figure 4-62. Pin Selection for Eddystone Project

	Name	Port	Pin	Lock
	ConnectableState_LED	P2[6]	43	✓
	SW2	P2[7]	44	✓
	TLM_LED	P3[7]	54	✓
	URL_UID_LED	P3[6]	53	✓
	Vref	P3[0]	47	✓

4.5.3 Flow Chart

Figure 4-63 shows the flow chart of code implemented.

Figure 4-63. Eddystone Project Flow Chart



4.5.4 Verify Output

The project can be verified using an Android smart phone or an iOS device (iPhone or iPad). Before proceeding further make sure that a beacon application (such as Locate Beacon for Android or Chrome browser for iOS) and the CySmart app are installed on the smart phone/device. Also make sure that Bluetooth is turned on in the device, and you have a working internet connection.

To verify the Eddystone project in iOS, using Chrome for iOS browser, follow these steps. Before you start, make sure the Chrome browser for iOS is installed on your iOS device and the Chrome widget is added to the widgets view.

1. Place the module on the BLE Pioneer Kit, depending on the project chosen.
2. Power the BLE Pioneer Kit through the USB connector J13.
3. Program the BLE Pioneer Kit with the Eddystone example project. Follow steps in Using Example Projects on page 37 to program the device.
4. After programming successfully, the firmware starts the non-connectable advertisement of UID/URL and TLM packets. Advertisement of UID/URL packets is indicated by the green LED and TLM packets is indicated by blue LED on the baseboard.
5. Pull down the notification area. You should be able to see the link in the notification area as shown in Figure 4-64 on page 78

Figure 4-64. iOS Notification Shade Showing the Beacon Advertised Web Link

6. Touching the link will take you to the website.

To verify the Eddystone project in Android devices using the Locate Beacon application, follow these steps.

1. Place the module on the BLE Pioneer Kit, depending on the project chosen.
2. Power the BLE Pioneer Kit through the USB connector J13.
3. Program the BLE Pioneer Kit with the Eddystone example project. Follow steps in Using Example Projects on page 37 to program the device.
4. After programming successfully, the firmware starts the non-connectable advertisement of UID/URL and TLM packets. Advertisement of UID/URL packets is indicated by the green LED and TLM packets is indicated by blue LED on the baseboard.
5. Launch the **Locate** application on the smart phone/device.

6. Click the **Locate Beacons** button to start scanning for beacons (see Figure 4-65).

Figure 4-65. Locate Beacon App Home Screen

7. In the next screen a list of **Visible Beacons** will be displayed. Click the desired beacon (see Figure 4-66) to display its status and information (see Figure 4-67).

Figure 4-66. Locate Beacon App Showing Discovered Beacons

Figure 4-67. Locate Beacon Showing Details about Selected Beacon

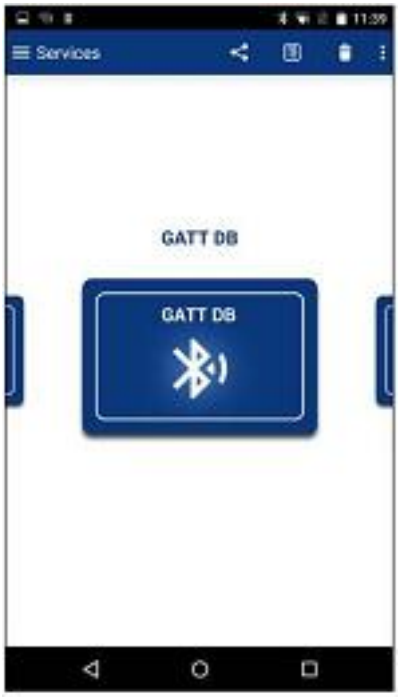
To read and write characteristics (using an Android or iOS device, and CySmart app) of the beacon follow these steps.

1. Press **SW2** on the BLE Pioneer Kit baseboard to start the connectable advertisement. The advertisement state is indicated by the red LED.
2. Launch the CySmart app on your Android smart phone device.
3. Connect to the **CY Eddystone** device from the list by clicking on it (see Figure 4-68).

Figure 4-68. CySmart App Showing the CY Eddystone Beacon

1. Click the **GATT DB** menu option (see Figure 4-69).

Figure 4-69. CySmart Showing GATT DB Option for CY Eddystone Device



5. Click on **Unknown Service** (see Figure 4-70) to see all available characteristics.

Figure 4-70. CySmart Showing Unknown Service in GATT DB

6. Click on any of the characteristics to read and modify it. For reference on Eddystone Configuration characteristics, refer to the Google Eddystone page. To understand more about the CySmart app, refer to the CySmart user guide.
7. Disconnect from the device to go back to the beacon mode.

4.6 BLE Dongle and LED Control

4.6.1 Project Description

This firmware supports the CySmart debug tool (refer CySmart Central Emulation Tool) by acting as the BLE host emulator. This is the default firmware that comes in the BLE Dongle shipped with the kit.

This project additionally demonstrates LED brightness control via a custom BLE profile, which works with the CapSense slider example explained in CapSense Slider and LED on page 41.

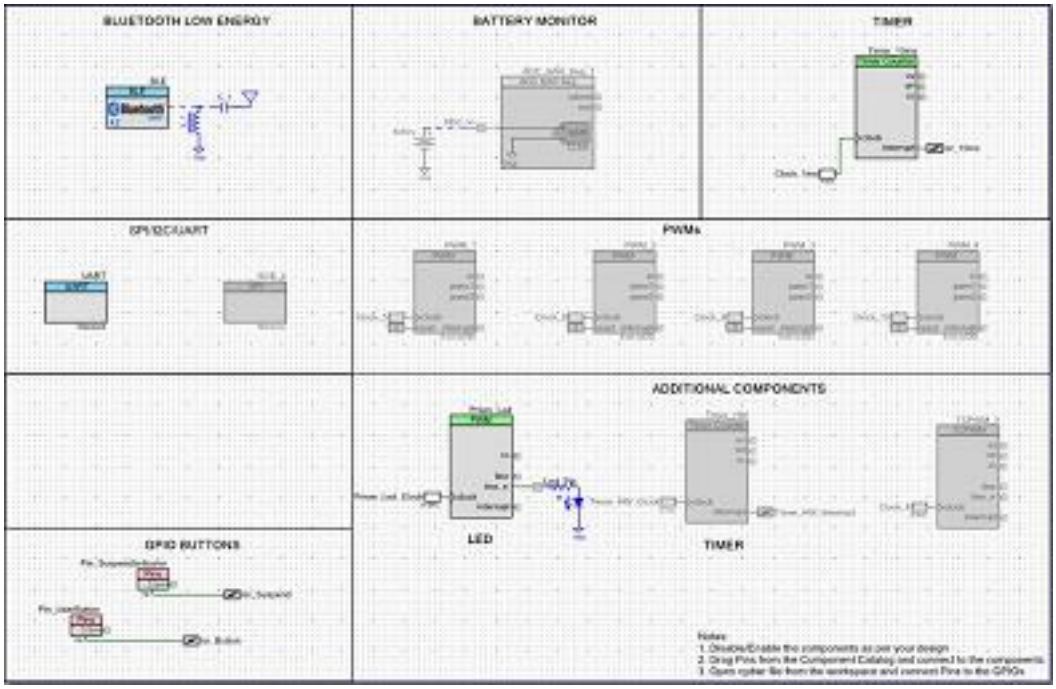
The device will scan for the Peripheral, which acts as a CapSense slider and LED device, and connect to it automatically. This is achieved by filtering the advertisement packets for the CapSense slider service data. Then, it will enable slider notifications and process the received notifications. Whenever CapSense detects activity, it will notify the finger location to the BLE Dongle; it will update the LED brightness using PWM.

The custom GATT client LED control will be stopped if the CySmart Central Emulation Tool acquires the dongle. The dongle will enter the CySmart emulator mode, in which it will process all BLE commands as triggered by the user via the tool. The project uses custom command/event protocol to exchange data between the CySmart Central Emulation Tool and the BLE device via a USB-CDC

interface. It uses the Cypress USB-UART bridge functionality from the PSoC 5LP-based KitProg module.

Note: This project is meant only for the PRoC BLE device and works on the CY5677 CySmart BLE 4.2 USB Dongle hardware. The example project for CY5670 CySmart USB Dongle is available in the CY8CKIT-042-BLE Pioneer Kit.

Figure 4-71. Top Design for BLE_4_2_Dongle_CySmart_256K Project



4.6.2 Hardware Connections

No specific hardware connections are required for this project because all connections are hardwired on the BLE Dongle.

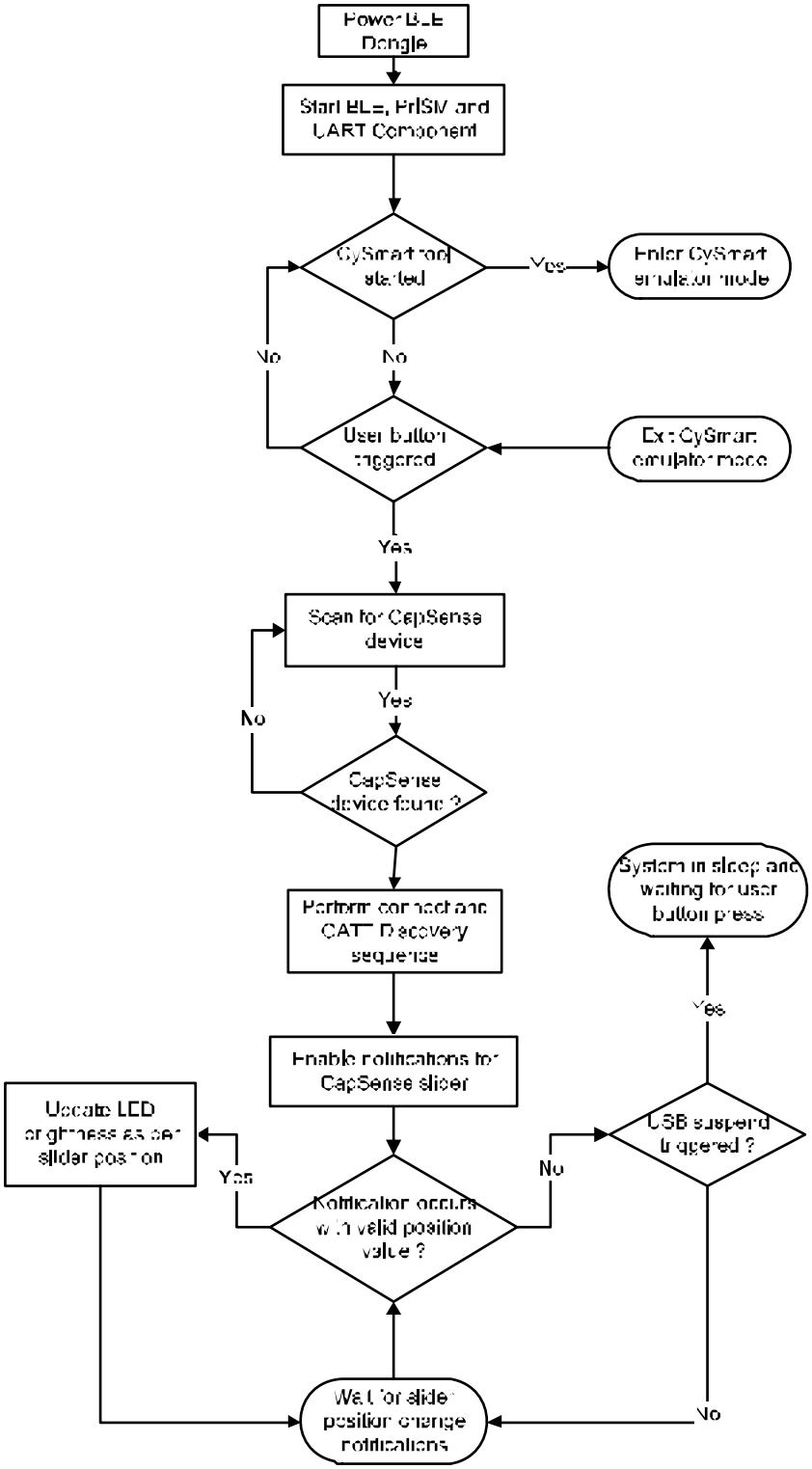
The pin assignment for this project is in **BLE_4_2_Dongle_CySmart_256K.cydwr** in the Workspace Explorer, as shown in Figure 4-72.

Figure 4-72. Pin Selection for BLE Dongle Project

Name	Port	Pin	Lock
\UART:rx_wake\	P1[4]	32	☑
\UART:tx\	P1[5]	33	☑
Led_Pin	P3[3]	50	☑
Pin_SuspendIndicator	P3[2]	49	☑
Pin_UserButton	P2[6]	43	☑

4.6.3 Flow Chart

Figure 4-73. Flow Chart for BLE_4_2_Dongle_CySmart_256K Project



4.6.4 Verify Output

This project will be used when the CySmart Central Emulation Tool is invoked for testing other example projects. In addition, the LED control operation can be verified as follows.

1. Power the BLE Pioneer kit through the USB connector J13.
2. Program the BLE Pioneer kit with the CapSense and LED example project described in CapSense Slider and LED on page 41.
3. Connect the BLE Dongle to one of the USB ports on the computer.
4. Program the BLE Dongle with the **BLE_4_2_Dongle_CySmart_256K** project. See Using Example Projects on page 37 for programming instructions.
5. Press the user button **SW2** on both the BLE Dongle and the BLE Pioneer Kit. The BLE Dongle will start scanning and the BLE Pioneer Kit will start advertising.
6. Wait for the BLE connection between the BLE Dongle and the BLE Pioneer Baseboard. The connection success status will be indicated by a 3-second ON state of the red LED followed by the OFF state on the BLE Pioneer Baseboard.
7. Swipe your finger on the CapSense slider and check the LED brightness variation on the BLE Dongle.

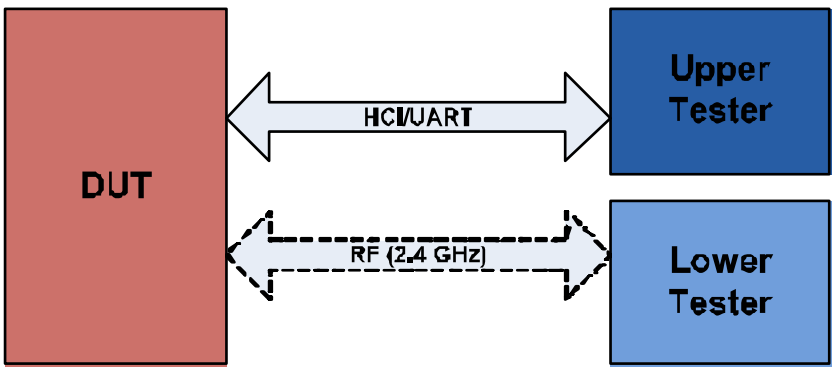
4.7 Direct Test Mode (DTM)

4.7.1 Project Description

Bluetooth Core specification (v4.0 and later), Volume 6, Part F defines Direct Test Mode (DTM) as a method to test the BLE PHY layer and provide a report back to the tester. It uses a **Host Controller Interface (HCI)** with a **two-wire UART** as the communication protocol.

The Device Under Test (DUT) is the BLE system that is to be tested (for example, BLE Pioneer Kit). With DTM, the RF performance of the BLE system can be verified during development or on a production line. The environment consists of the DUT and a tester. The tester has two parts; the **upper tester** sends commands through the HCI and the **lower tester** performs the corresponding action over the RF link. The tester compares the command sent over the HCI and the response received over RF, and provides a result of the performance.

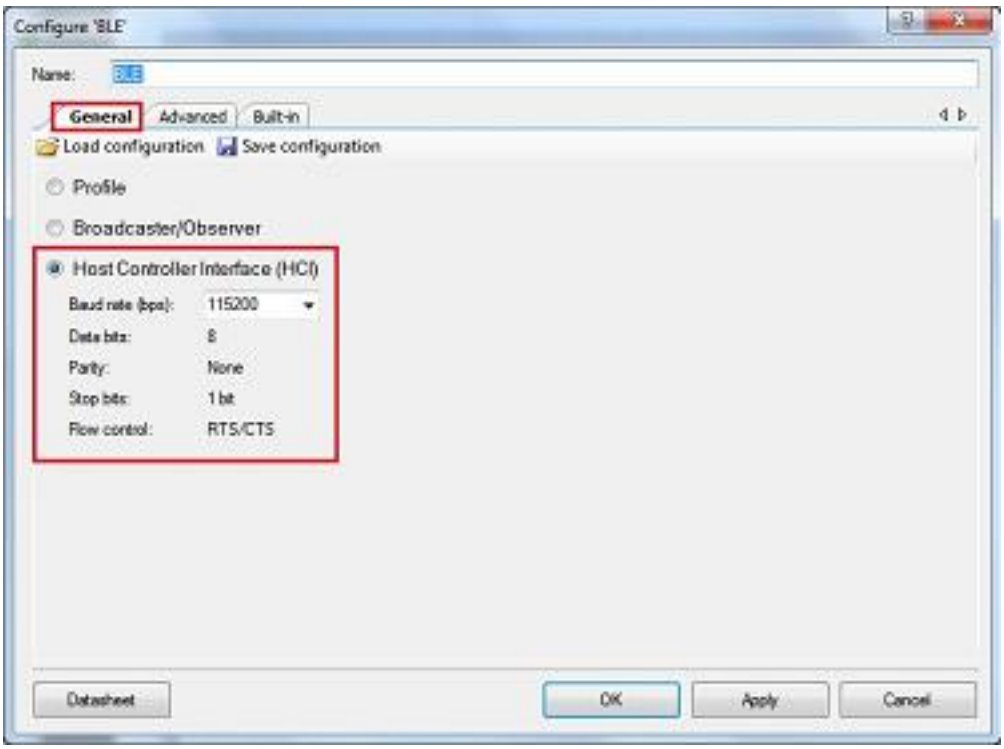
Figure 4-74. Direct Test Mode (DTM) Setup



The BLE Component allows configuring the device in DTM by enabling the HCI. The appropriate responses to commands from the tester are performed by the BLE protocol stack and does not involve separate application handling. The only task required are to start the BLE Component and call the API to process the events.

The HCI is enabled in the BLE Component under the General settings. Note that when the HCI mode is selected, all other tabs are hidden and cannot be configured. This is because in HCI mode, there are no upper layer processes. On enabling HCI mode, the component automatically reserves a UART block to allow communication between the tester and BLE stack. The UART exposes the pins that can be assigned in *<project.cydwr>* file under the Pins tab. The only options to be configured for HCI mode are the baud rate and the pins for communication with the tester.

Figure 4-75. HCI Mode in BLE Component

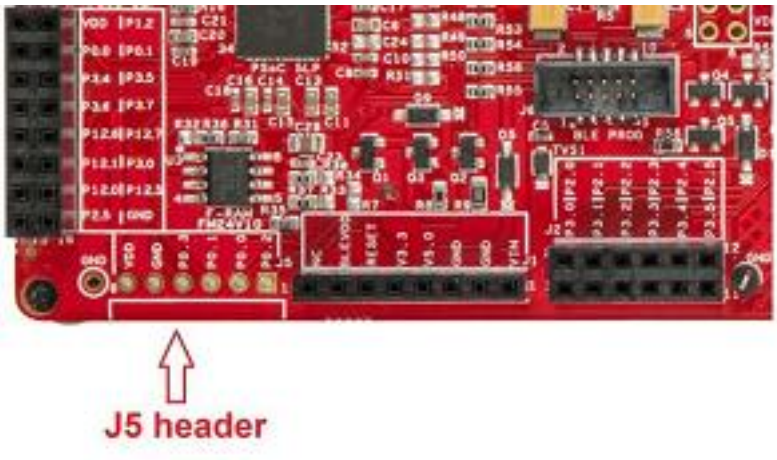


Many companies develop BLE testers for Direct Test Mode. It is also possible to create PC-based software tools that will send HCI commands over serial communication links.

For PC-based software, the serial communication link is the COM port, which is enumerated by the KitProg on the PSoC 5LP of the BLE Pioneer kit. In such a case, the UART pins on the PSoC 4 BLE/ PRoC BLE should be assigned to P1_4 and P1_5. These pins are hardwired to pins on the PSoC 5LP which allows USB-UART data communication between the PC-based software and the BLE device.

For external BLE testers, the serial communication is mostly over RS232. To test with that type of system, an external RS232 voltage translator is required, such as Digilent's PmodRS232. This translator will modify the signal levels of the serial communication between the BLE device and the RS232 port on the tester. The UART pins of the BLE device can be assigned to P0_0 and P0_1 and header J5 can be used to connect to the RS232 translator.

Figure 4-76. J5 Header to Interface RS232 Translator



4.7.2 Hardware Connection

For DTM test mode, it is recommended to use SMA connectors and connect the tester and DUT using an SMA to SMA connector cable. This ensures that there is minimum interference to RF communication between the DUT and tester, and the performance measured is the true RF performance of the device. The BLE Pioneer Kit module with SMA connector (CY8CKIT-141 PSoC 4 BLE) is available separately and can be ordered from the Cypress web page.

Four UART pins are exposed when HCI mode is selected in the BLE Component. These pins should be assigned to allow communication with the external tester. The connection depends on the tester being used.

If the tester is PC-based software that communicates with HCI over a serial link, then the onboard PSoC 5LP on the BLE Pioneer Kit can act as the USB-UART bridge. The KitProg on the PSoC 5LP enumerates as a USB-UART interface and opens a COM port on the computer. This COM port is then used by the software tool to communicate commands to the BLE device. In this case, the UART pins should be assigned as follows.

Table 4-1. UART Pin Assignment for PC Software Tester

UART Pins	Pin Assigned
RX	P1_4
TX	P1_5
RTS	P1_6
CTS	P1_7

The UART for HCI communication exposes hardware flow control lines CTS and RTS. They can either be connected to the hardware control lines of the tester or CTS connected to ground for operation without hardware flow control.

If the tester is an external hardware tester (CBT), then connect any of the RS232 voltage translators to header J5 on the BLE Pioneer Kit. The UART pins should be assigned as follows.

Table 4-2. UART Pin Assignment for RS232 Voltage Translator

UART Pins	Pin Assigned
RX	P0_0
TX	P0_1
RTS	P0_2
CTS	P0_3

4.7.3 Verify Output

1. Connect the BLE Pioneer Kit through the USB connector J13.
2. Program the BLE Pioneer Kit with the PSoC_4_BLE_DTM or PRoC_BLE_DTM project, depending on the module used (PSoC 4 BLE or PRoC BLE), as described in Using Example Projects on page 37. Programming should complete successfully.
3. Connect the serial link, UART, or RS232 to the tester.
4. On the software tool for tester, configure the UART communication with the correct COM port and baud rate, as set in the BLE Component.
5. Start the test. The tool will generate the report after the end of the test. This depends on the tester tool being used.