

## Objective

This example demonstrates one of the methods to implement BLE Mesh network using Flooding mechanism. The data between nodes is transferred after BLE connection, where each node switches between GAP Central and Peripheral roles.

## Overview

This project demonstrates how to transform BLE Pioneer Kits with PSoC 4 BLE into nodes of a Mesh Network and relay common data over that network using flooding mechanism. This project employs connection method, where a BLE connection is made between two nodes before data is transferred. Once the data has been received by a node, the node switches GAP role to connect to other nodes and relay the same data. The data relayed in this project is of RGB LED color and intensity control and the Mesh Network allows it to be relayed to each node in that network. The project can be easily modified to relay any other form of data.

**Note:** From here on, the BLE Pioneer Kits with PSoC 4 BLE Module will be referred to as nodes of this Mesh network, unless explicitly specified.

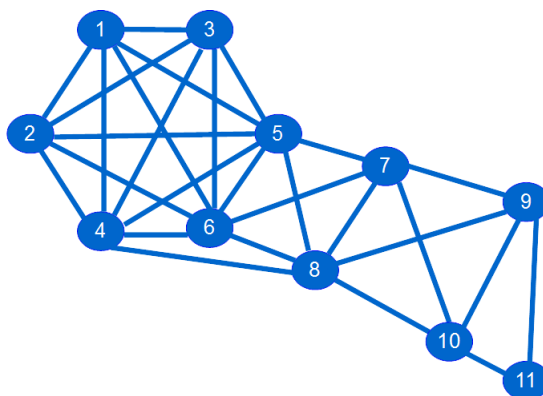
Some of the features of this project are:

- Same firmware for each node: Each node runs same firmware, making it easy to maintain the identity of nodes in the network
- Scalability of network: As the network employs Flooding mechanism, there is no separate processing required to deal with changes in size of network, such as when nodes are added or removed from the network.
- Size of network: There are (theoretically) no limits on number of nodes that can be added as part of this network, though limit may arise due to interference on BLE channels from increasing number of nodes.
- Prevention of multiple connection to same nodes: The application prevents a node from connecting to other node which has either the new relayed data or has received data from other node, increasing reliability of data relay.

## Theory and Operation

Mesh network is a communication topology where all devices can connect to all other devices (in range) in network. These network model do not employ a Central hub.

Figure 1. Mesh Network Topology



For BLE, Mesh network is becoming an important feature that will allow devices to be integrated into IoT and effectively increases the range and usability of BLE communication. One such use case is home automation.

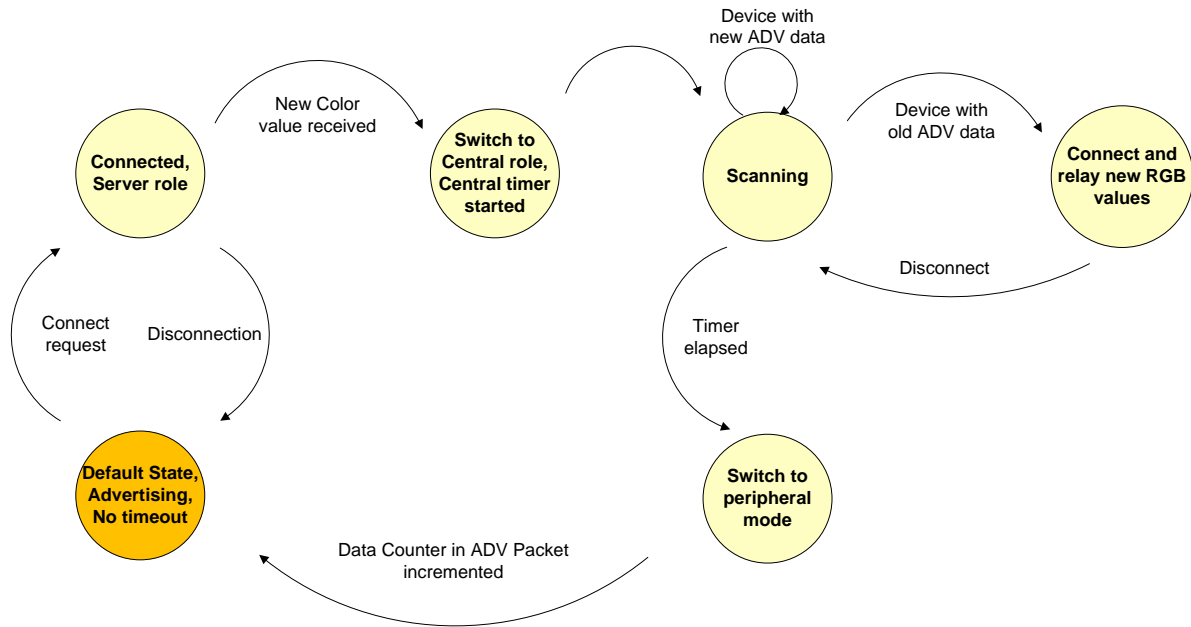
As there are no BLE SIG specifications for BLE Mesh network yet, there are varied ways to implement BLE Mesh. Each method has its pros and cons and suits one application better than other.

We implement, in this project, Mesh network with Flooding mechanism over BLE connections. Flooding means that there is no routing involved while transferring data from one node to another and each node can (and will) receive the relayed data. BLE connections imply that the nodes connect over BLE before transferring the data. This way, there is no limitation on the size of payload that can be transferred between the nodes.

**Note:** The project is configured for lowest latency involved in transferring data between nodes. This involves frequent BLE TX/RX activity, causing higher current consumption. Do not operate the project on battery/coin cell and use only USB power.

Following diagram shows the various states at application layer and there transitions.

Figure 2. Application State Diagram



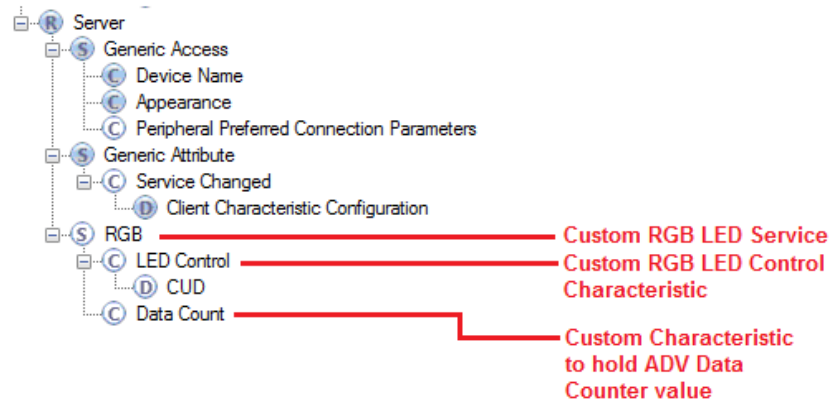
The project supports both GAP Central and GAP Peripheral role. The default state is GAP Peripheral where the nodes advertise. The advertisement data contains custom ADV data counter which is a value from 0-255. Each time a new RGB LED data is relayed to the node, this counter value is incremented and later advertised. A scanning node will read this counter value to determine whether the advertising node has new RGB LED data or not. If the node is advertising with old data counter, then this node will be connected to.

Once the scanning node connects to the advertising node, the scanning node becomes the GATT Client and advertising node becomes the GATT Server. The GATT Client writes the RGB LED data to the GATT Server, after which the GATT Server disconnects to switch role to Central device. The node acting as GAP Central/GATT Client has internal timer which triggers a GAP role switch whenever the timer crosses the set value.

The new RGB LED data is sent by BLE Central device such as CySmart PC Tool or similar, after connecting to any of the node.

The GATT server has following services and characteristic:

Figure 3. Server Attributes



## Requirements

**Design Tool:** PSoC Creator 3.1 SP3, CySmart 1.0

**Programming Language:** C (GCC 4.8.4 – included with PSoC Creator)

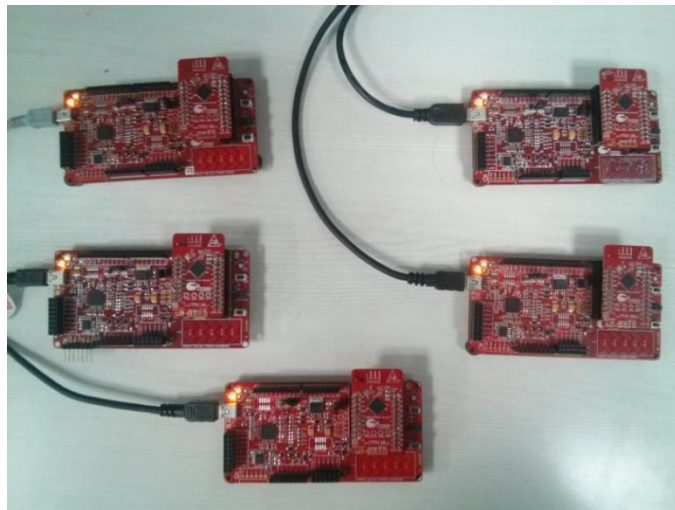
**Associated Devices:** All PSoC 4 BLE devices

**Required Hardware:** CY8CKIT-042-BLE Bluetooth® Low Energy (BLE) Pioneer Kit

## Hardware Setup

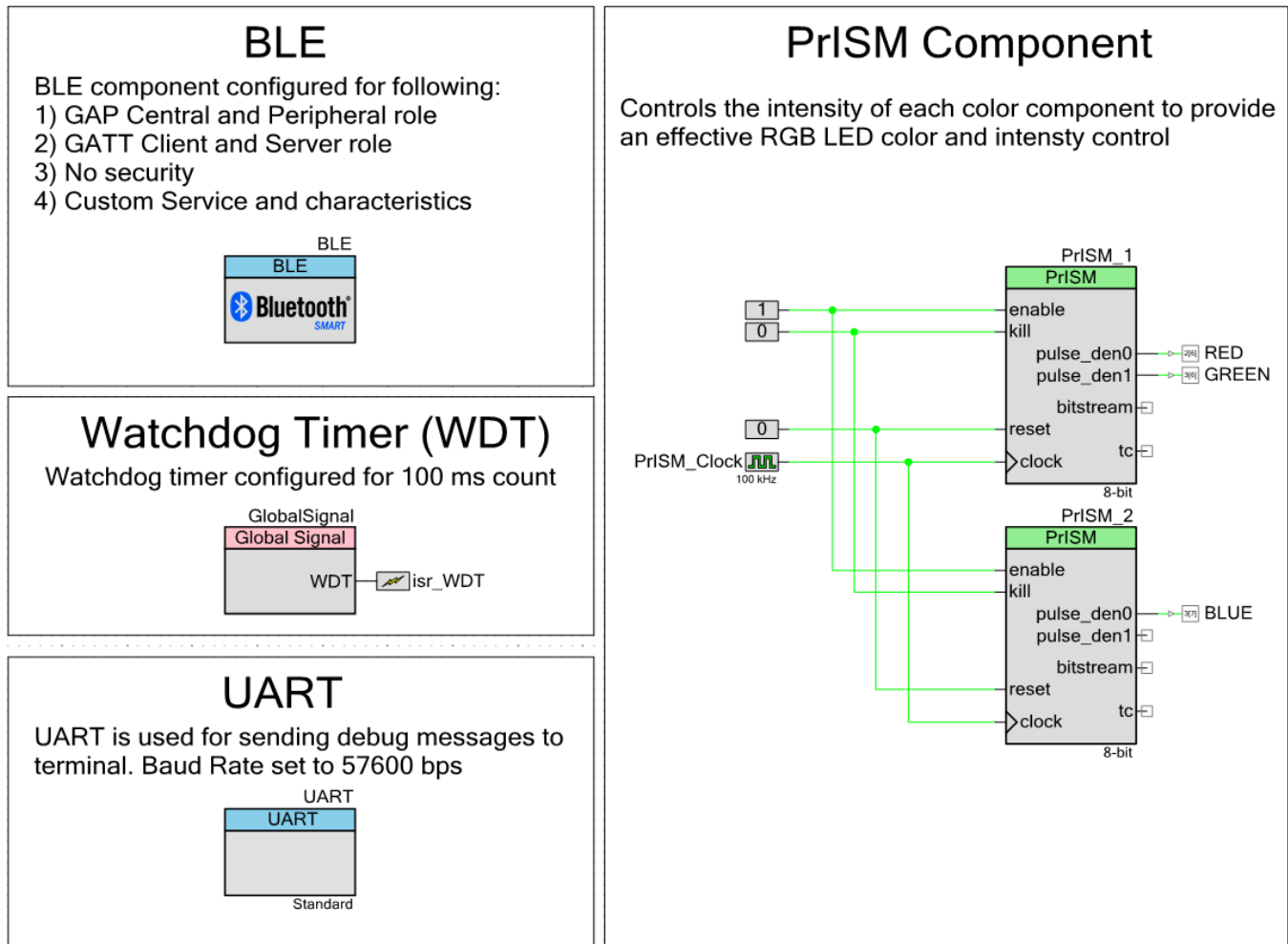
The BLE Pioneer Kit has all of the necessary hardware required for this example. For demonstration of this project, you will require 2 or more (say 10) BLE Pioneer Kits. Figure 4 shows the hardware setup for this example (with 5 kits). Connect the kits to PC using USB cable.

Figure 4: Kits Setup



## PSoC Creator Schematic

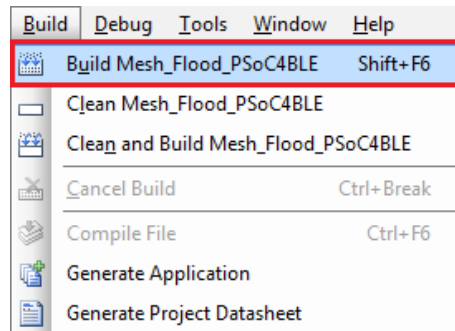
Figure 5. PSoC Creator Schematic



## Testing

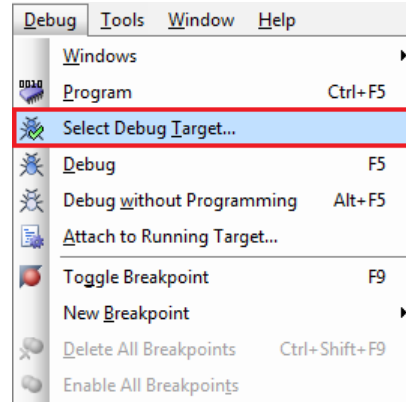
- 1) Open the associated example project *Mesh\_Flood\_PSoC4BLE*. Using PSoC Creator 3.1 SP3 or later.
- 2) Build the project by going to Menu bar -> **Build** -> **Build Mesh\_Flood\_PSoC4BLE**

Figure 6. Build the project



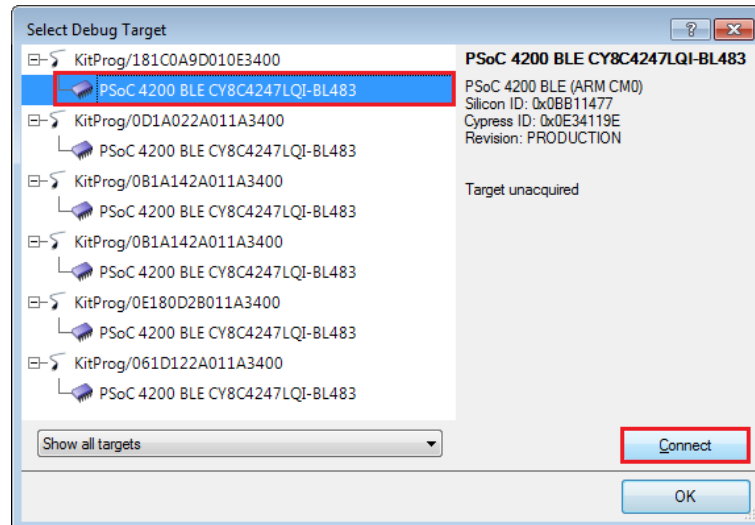
- 3) Once the build is completed successfully, program each kit with the built firmware, one by one. For this, go to **Debug** menu -> **Select Debug Target**. This will open up a dialogue box with KitProg of all connected kits.

Figure 7. Select the kit to be programmed



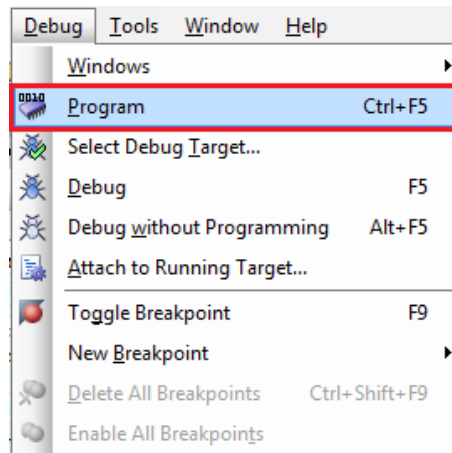
- 4) Select the BLE Pioneer kit, starting from first one, and click **Connect** -> **OK**

Figure 8. Select the Kit to be programmed



- 5) Go to Debug -> Program to program the select BLE Pioneer Kit.

Figure 9. Program the kit with built firmware



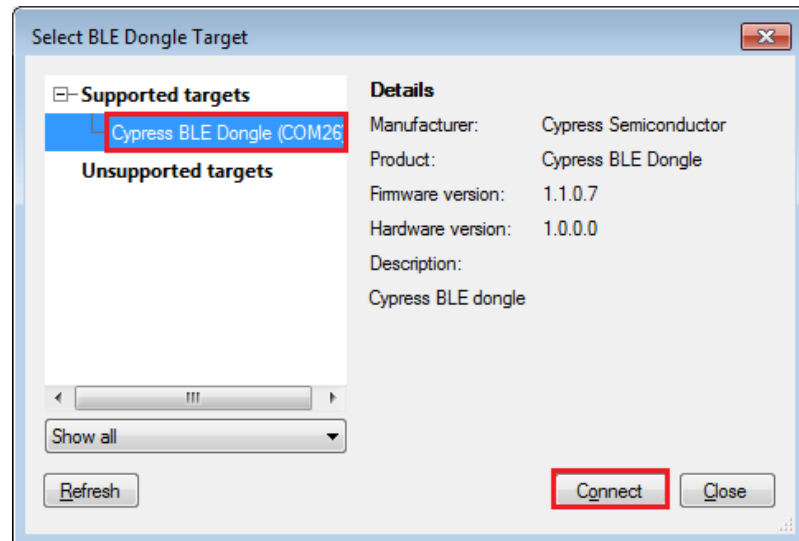
- 6) Repeat steps 3-5 above to program the remaining connected kits.
- 7) Connect BLE Dongle (part of the BLE Pioneer Kit) to PC and open [CySmart PC Tool](#) from **Start -> All Programs -> Cypress -> CySmart 1.0 -> CySmart 1.0**.

Figure 10. Connect BLE Dongle to PC



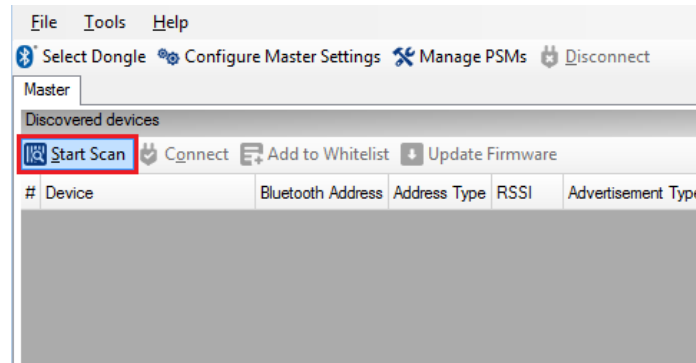
- 8) Click on the BLE Dongle name under **Supported targets** and click **Connect**. If the BLE Dongle is listed under **Unsupported targets**, click on **Refresh**.

Figure 11. Select BLE Dongle on CySmart PC Tool

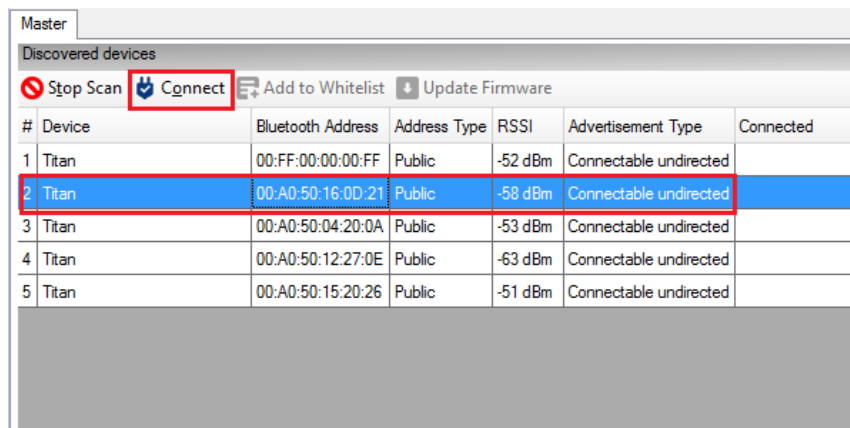


- 9) Click on **Start Scan** to start BLE scanning.

Figure 12. Start BLE Scanning

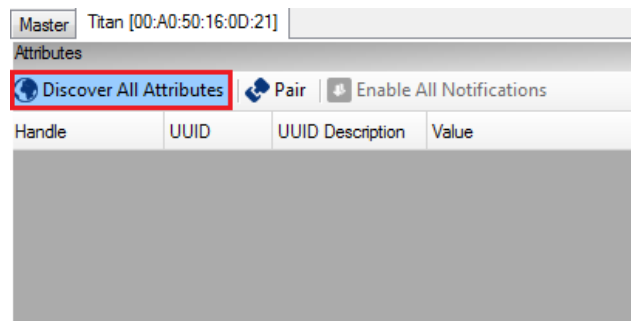


- 10) The programmed BLE Pioneer Kits will advertise as *Titans*. Double click on any one of the *Titan* to connect to it or simply select any one of it and click on **Connect**.

Figure 13. Connect to *Titan*


- 11) Click on **Discover All Attributes** to list the supported services and characteristics.

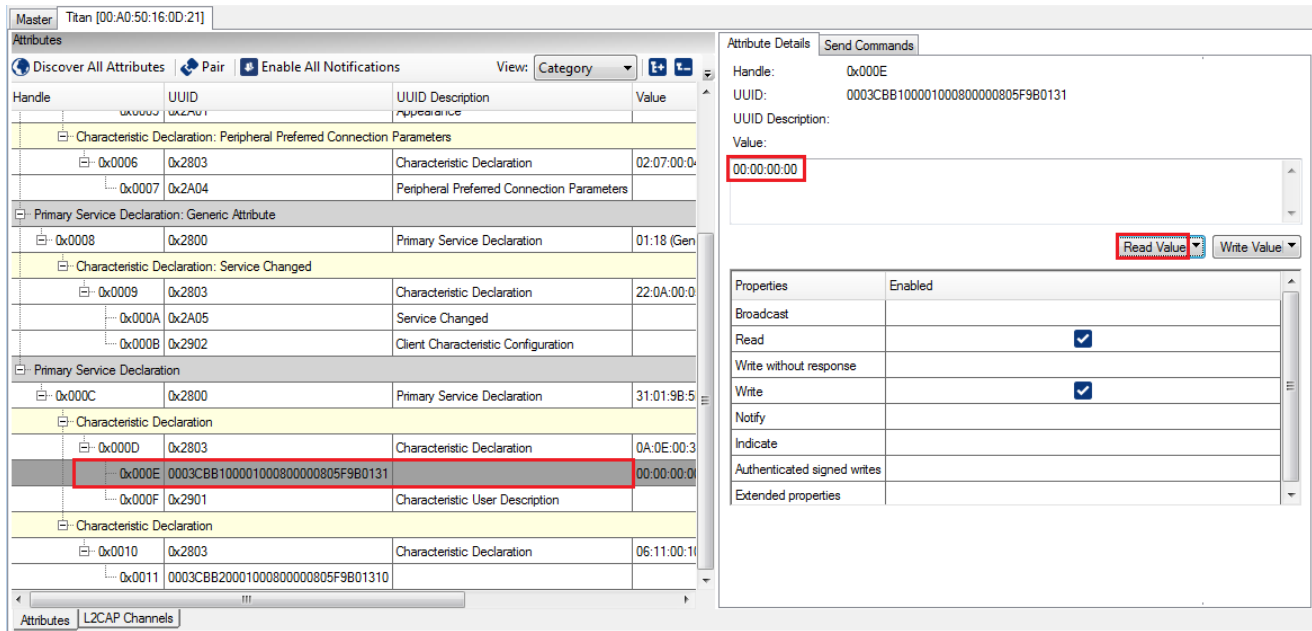
Figure 14. Discover All Attributes



- 12) Locate the RGB LED characteristic and read its current value as shown below. This will be a 4 byte value.



Figure 15. Read existing RGB LED color value

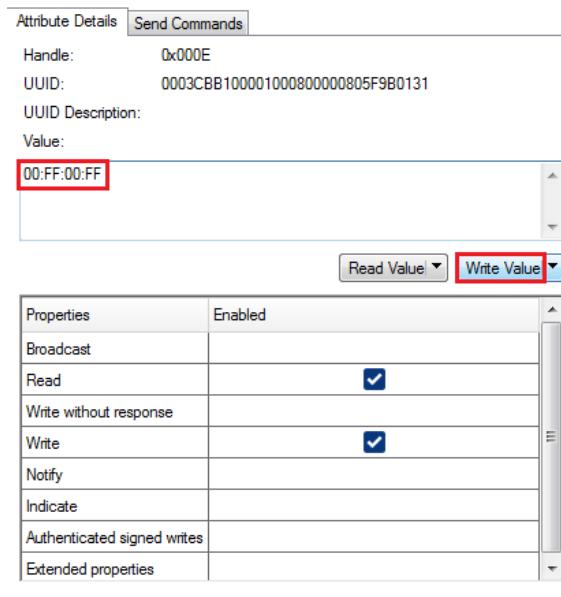


The screenshot shows the CySmart PC tool interface. The 'Attributes' tab is active, displaying a list of attributes. The 'Characteristic Declaration: Peripheral Preferred Connection Parameters' is selected. The 'Value' field shows '00:00:00:00'. The 'Read Value' button is highlighted.

Handle	UUID	UUID Description	Value
0x0006	0x2803	Characteristic Declaration	02:07:00:00
0x0007	0x2A04	Peripheral Preferred Connection Parameters	
0x0008	0x2800	Primary Service Declaration	01:18:9B:5
0x0009	0x2803	Characteristic Declaration	22:0A:00:0
0x000A	0x2A05	Service Changed	
0x000B	0x2902	Client Characteristic Configuration	
0x000C	0x2800	Primary Service Declaration	31:01:9B:5
0x000D	0x2803	Characteristic Declaration	0A:0E:00:3
0x000E	0003CBB100001000800000805F9B0131	Characteristic Declaration	00:00:00:0
0x000F	0x2901	Characteristic User Description	
0x0010	0x2803	Characteristic Declaration	06:11:00:1
0x0011	0003CBB200010008000000805F9B0131	Characteristic Declaration	

- 13) Modify the four byte to your desired value. The format is [Red:Green:Blue:Intensity], with 0x00 being least and 0xFF being highest value for each byte. Click **Write Value**.

Figure 16. Write Green color on with full intensity

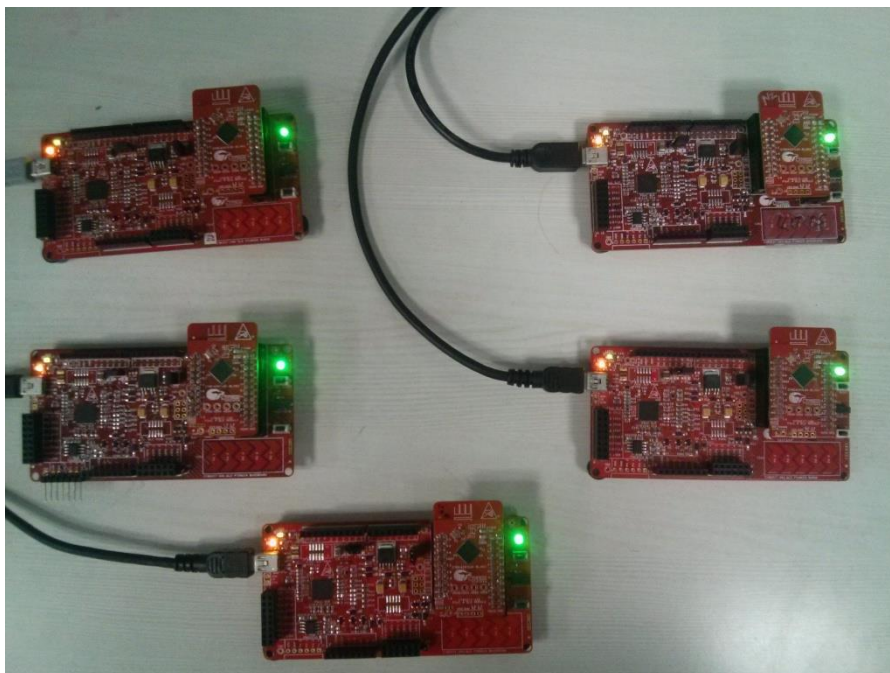


The screenshot shows the 'Attribute Details' tab. The 'Value' field is set to '00:FF:00:FF'. The 'Write Value' button is highlighted.

Properties	Enabled
Broadcast	
Read	<input checked="" type="checkbox"/>
Write without response	
Write	<input checked="" type="checkbox"/>
Notify	
Indicate	
Authenticated signed writes	
Extended properties	

- 14) Along with instant disconnection of the device from CySmart PC tool, it will be observed that the RGB LED color has changed and all other *Titan* nodes will be having the same color.

Figure 17. RGB LED data relayed to all nodes



- 15) Scan again in CySmart PC tool and you will see list of Titans re-appearing after about 5-6 seconds (timeout period).
- 16) Connect again to any of the node and modify its color to reflect the color on all other nodes.
- 17) Also, you can bring a new kit and change color as part of this network.

## Related Documents

Table 1 lists all relevant application notes, code examples, knowledge base articles, device datasheets, and Component datasheets.

Table 1. Related Documents

Document	Title	Comment
<a href="#">AN91267</a>	Getting Started with PSoC 4 BLE	Provides an introduction to PSoC 4 BLE device that integrates a Bluetooth Low Energy radio system along with programmable analog and digital resources.
<a href="#">AN91445</a>	Antenna Design Guide	Provides guidelines on how to design an antenna for BLE applications.
<a href="#">AN91162</a>	Creating a BLE Custom Profile	Provides methodology to create your own BLE custom profile on PSoC 4 BLE/PROC BLE device.