

## BLE Mesh Network (addressed) using GATT Connections

### Objective

This example demonstrates one of the methods to implement BLE Mesh network using Flooding mechanism, where nodes are addressed to send unicast or broadcast data. 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 either common data or data to a particular node 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 all nodes or a particular node in that network, depending on address selected. 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.

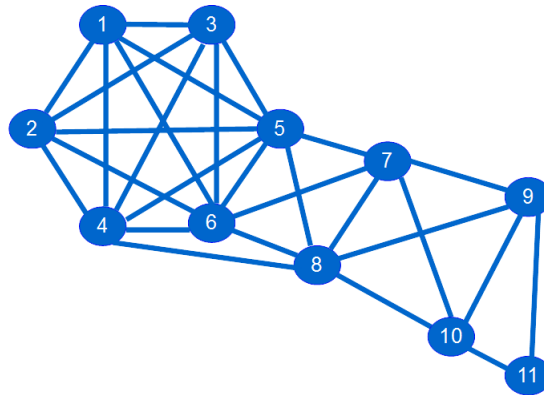
Some of the features of this project are:

- Same firmware for each node: Each node runs same firmware, making it easy to maintain the identity of nodes in the network
- Node Addressing: This network allows the data to be sent to a particular node of the network or broadcast it to all 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.
- Preventing duplicate connection to a 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 and minimizing interference.

### 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.

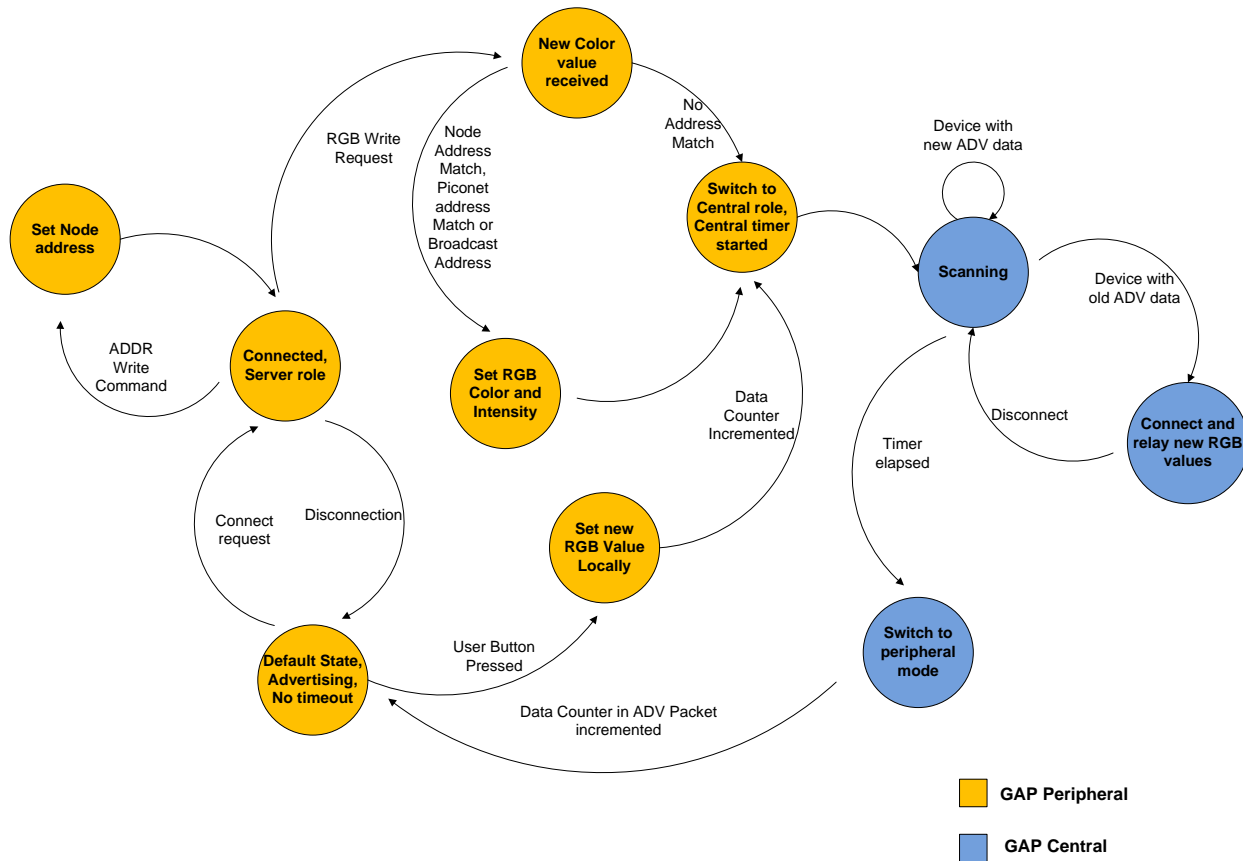
As there are no BLE SIG specifications for BLE Mesh network yet, there are varied ways to implement BLE Mesh. Each method has its pros and cons and suits one application better than other.

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. Though the project can be modified to support Low Power, it is not part of this project. 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.

## Requirements

**Design Tool:** [PSoC Creator 3.3](#), [CySmart 1.1](#)

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

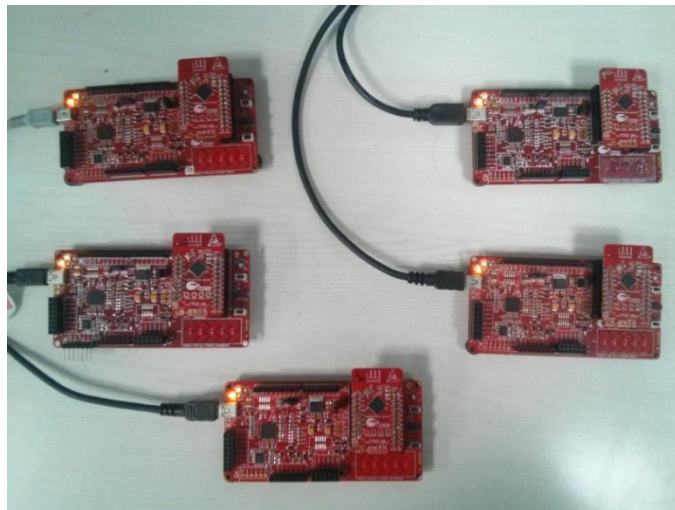
**Associated Devices:** All PSoC 4 BLE devices

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

## Hardware Setup

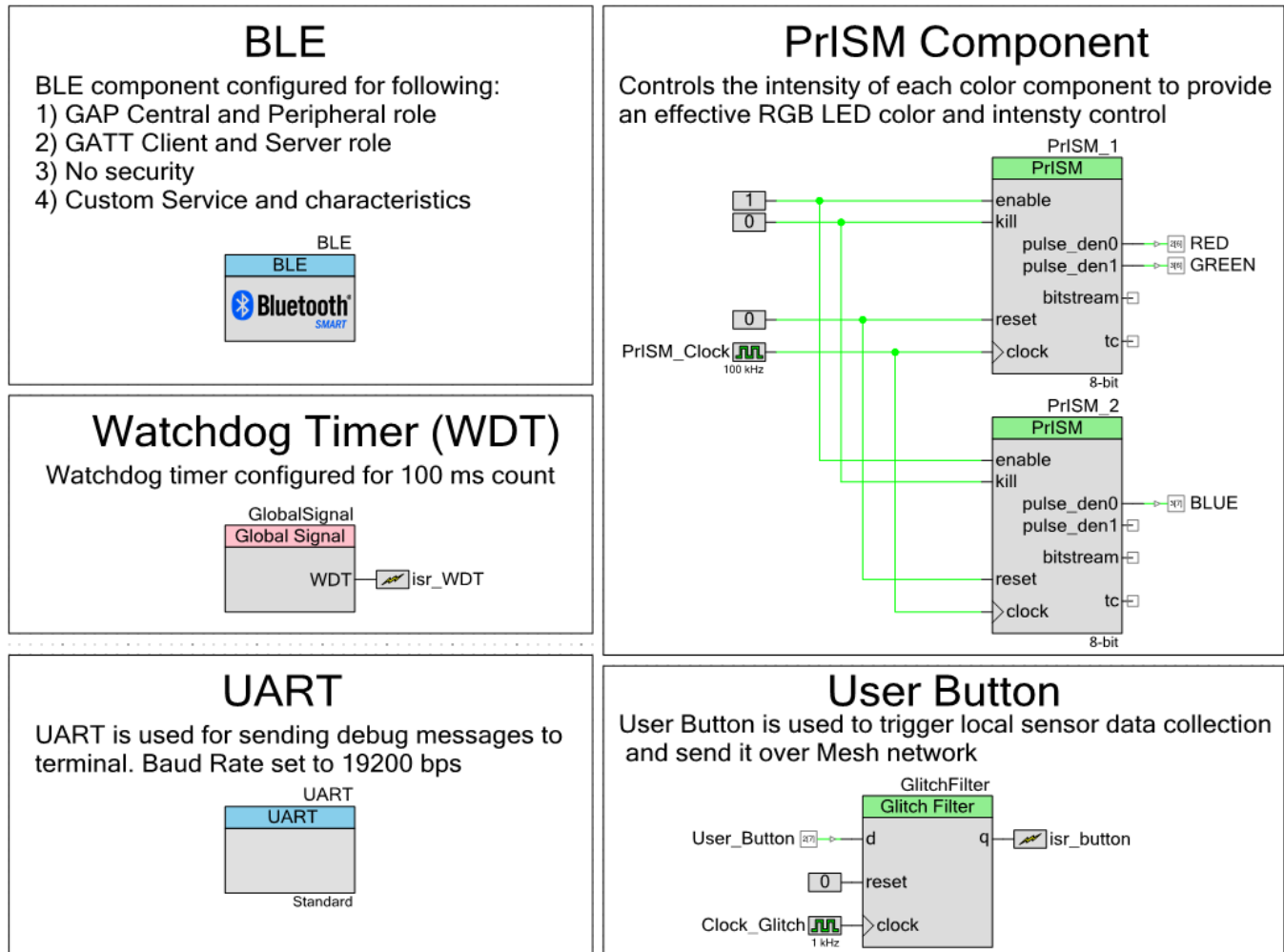
The BLE Pioneer Kit has all of the necessary hardware required for this example. For demonstration of this project, you will require 2 or more (say 10) BLE Pioneer Kits. Figure 3 shows the hardware setup for this example (with 5 kits). Connect the kits to PC using USB cable.

Figure 3: Kits Setup



## PSoC Creator Schematic

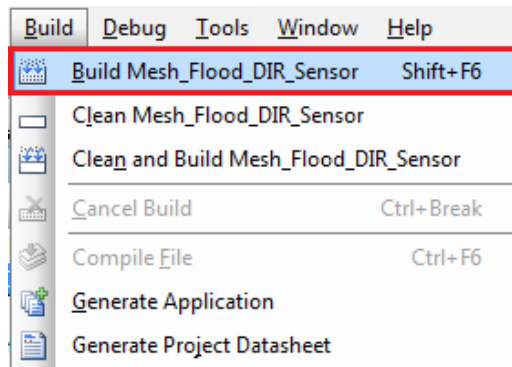
Figure 4. PSoC Creator Schematic



## Testing

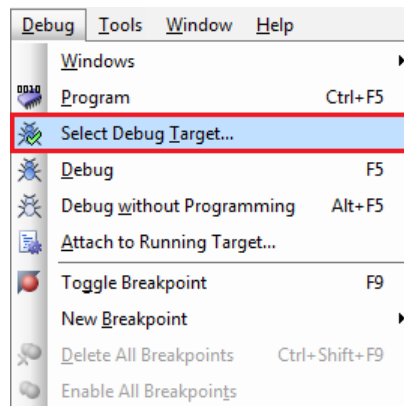
- 1) Open the associated example project *Mesh\_Flood\_DIR\_Sensor* using PSoC Creator 3.3 or later.
- 2) Build the project by going to Menu bar -> **Build** -> **Build Mesh\_Flood\_DIR\_Sensor**.

Figure 5. Build the project



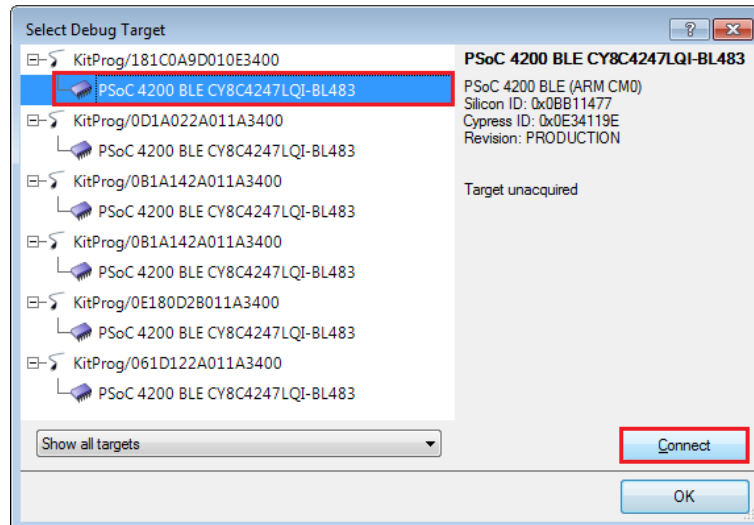
- 3) Once the build is completed successfully, program each kit with the built firmware, one by one. For this, go to **Debug** menu -> **Select Debug Target**. This will open up a dialogue box with KitProg of all connected kits.

Figure 6. Select the kit to be programmed



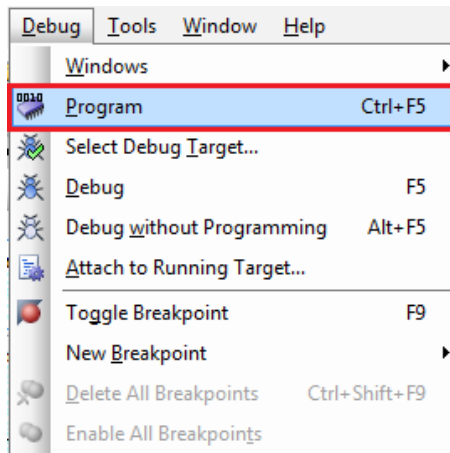
- 4) Select the BLE Pioneer kit, starting from first one, and click **Connect** -> **OK**

Figure 7. Select the Kit to be programmed



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

Figure 8. Program the kit with built firmware



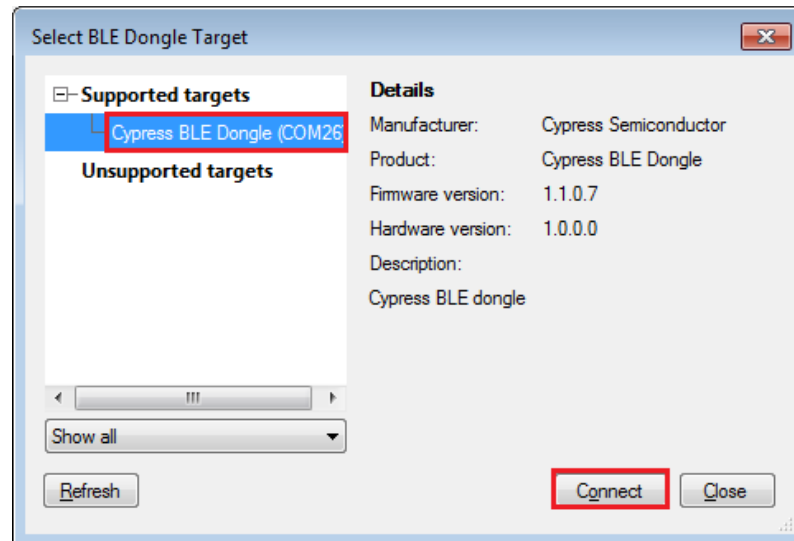
- 6) Repeat steps 3-5 above to program the remaining connected kits.
- 7) Connect BLE Dongle (part of the BLE Pioneer Kit) to PC and open [CySmart PC Tool](#) from **Start -> All Programs -> Cypress -> CySmart 1.1 -> CySmart 1.1**.

Figure 9. 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**.

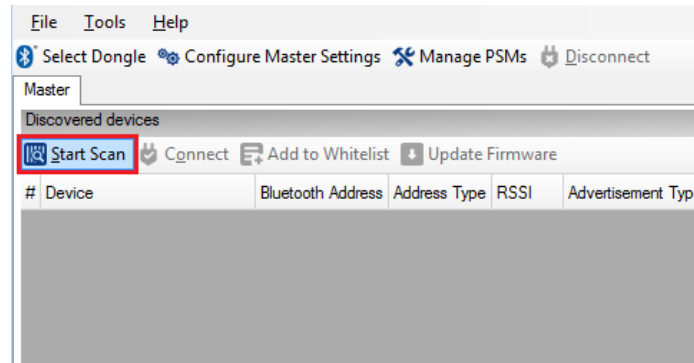
Figure 10. Select BLE Dongle on CySmart PC Tool



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

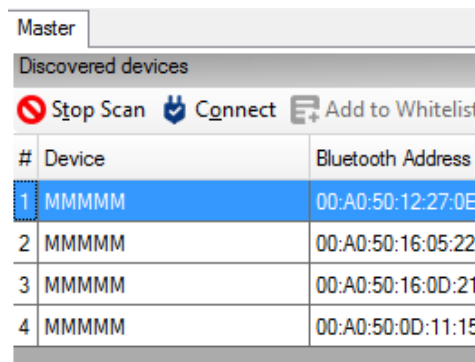


Figure 11. Start BLE Scanning



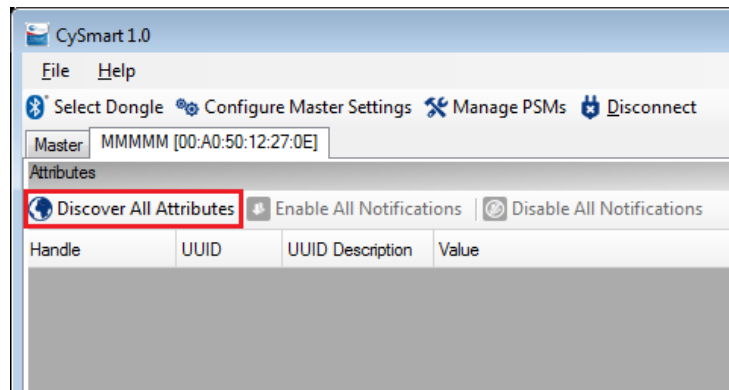
- 10) *MMMMM* indicates that the *Mesh\_Flood\_DIR\_Sensor* project is advertising but has not been assigned a unique address. This address, a **2 byte value**, is to be assigned to each node to allow directed data transfer on this mesh network. Without this address, the node will not accept or relay any data from any other node/Central device.

Figure 12. Connect to MMMMM devices



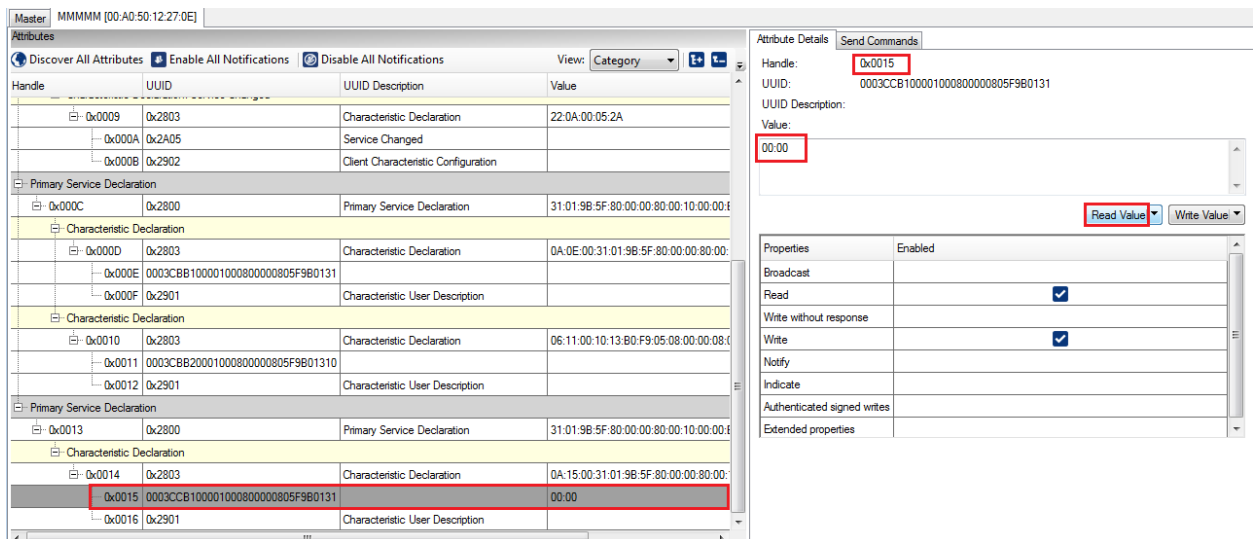
- 11) To assign address to the node, click on the device name and then click **Connect**.
- 12) Once the connection is done, click on **Discover All Attributes** to discover the supported services by the node.

Figure 13. Discover All Attributes of connected device



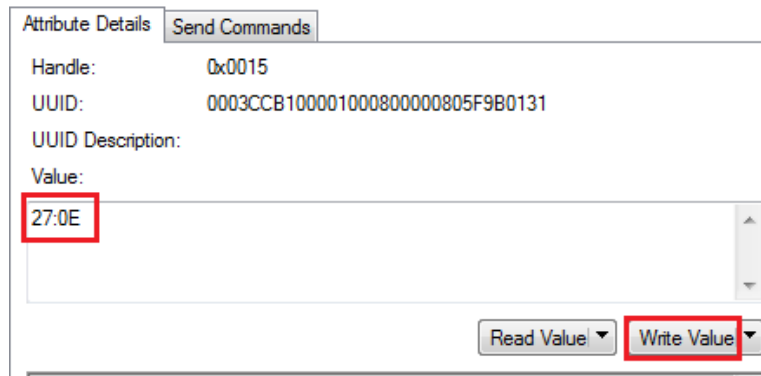
- 13) At the end of the list, locate the custom characteristic with attribute handle **0x0015**. This is the custom characteristic to set the node address. Select it and click on **Read Value** at right to read the existing value. This will be 00:00

Figure 14. Read Node address characteristic



- 14) Modify the address bytes to any non-zero value and click on **Write Value**. This will set the node address for that particular node. 00:00 and FF:FF are not allowed.

Figure 15 Write new Node address for the device



Attribute Details Send Commands

Handle: 0x0015

UUID: 0003CCB100001000800000805F9B0131

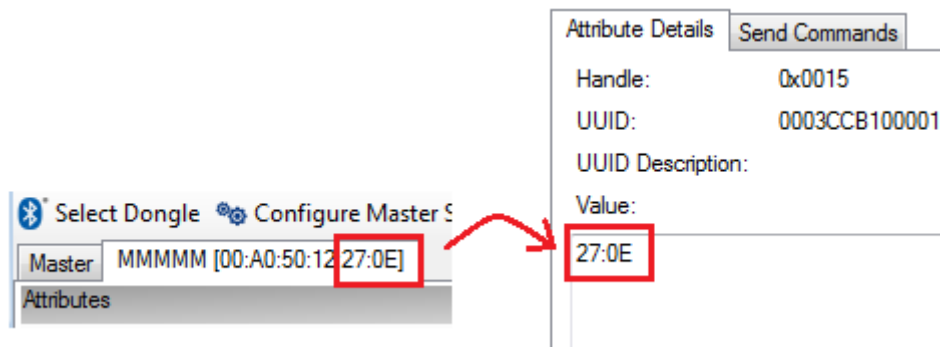
UUID Description:

Value: 27:0E

Read Value Write Value

**Note:** The node address can be any non-zero value. For quick setting, you can set the node address as the last two bytes of the BD address of the node. This ensures some level of address uniqueness followed by easy recognition of the node you want to send data to.

Figure 16. Use Last two bytes of BD address for node address



Select Dongle Configure Master

Master MMMMM [00:A0:50:12:27:0E]

Attributes

Attribute Details Send Commands

Handle: 0x0015

UUID: 0003CCB100001000800000805F9B0131

UUID Description:

Value: 27:0E

- 15) Click on **Disconnect** to disconnect from the node.
- 16) Follow steps 11-15 for all other nodes which have been not assigned a node address (device name MMMMM).
- 17) As each node has been assigned a node address, they will start advertising with name **Titan**. These are nodes that can be connected to send or relay data.

Figure 17. Devices with assigned addresses advertising as *Titan*

Master						
Discovered devices						
<div> <span>Stop Scan</span> <span>Connect</span> <span>Add to Whitelist</span> <span>Update Firmware</span> </div>						
#	Device	Bluetooth Address	Address Type	RSSI	Advertisement Type	Connected
1	Titan	00:A0:50:16:0D:21	Public	-49 dBm	Connectable undirected	
2	Titan	00:A0:50:16:05:22	Public	-46 dBm	Connectable undirected	
3	Titan	00:A0:50:0D:11:15	Public	-33 dBm	Connectable undirected	
4	Titan	00:A0:50:12:27:0E	Public	-39 dBm	Connectable undirected	

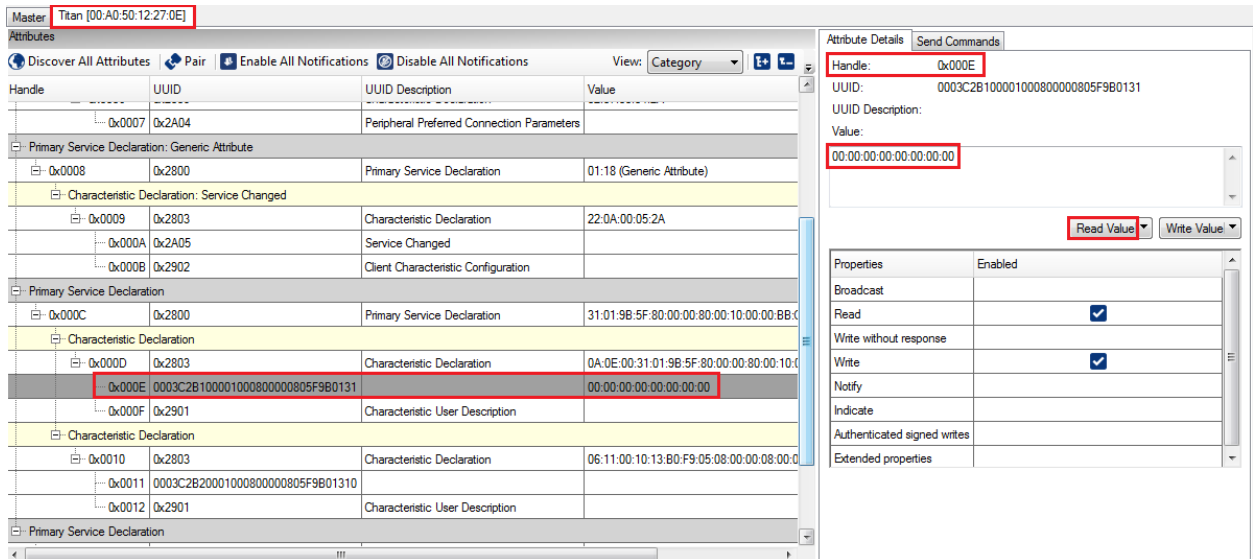
- 18) Connect to any of these nodes by double clicking on it or selecting the name and clicking on **Connect**.

Figure 18. Connect to *Titan* device

Master			
Discovered devices			
<div> <span>Stop Scan</span> <span>Connect</span> <span>Add to Whitelist</span> <span>Update Fi</span> </div>			
#	Device	Bluetooth Address	Address Type
1	Titan	00:A0:50:16:0D:21	Public
2	Titan	00:A0:50:16:05:22	Public
3	Titan	00:A0:50:0D:11:15	Public
4	Titan	00:A0:50:12:27:0E	Public

- 19) Click on **Discover All Attributes** and locate the custom characteristic with attribute handle **0x000E**. This is the RGB LED custom characteristic. Click on **Read Value** to read the existing 8 bytes value.

Figure 19. Discover all attributes of connected device



- 20) The first byte indicates if the address in the payload is general addressing (0x00) or Piconet addressing (0x01). If general addressing is set, then the node will look for either 0x00 0x00 or its own 2 byte node address at ADDR1 and ADDR2 location of payload before it processes RGB data. If Piconet Addressing is set, then the node will look for only ADDR1 byte and process the RGB data if it matches the first byte of its own node address.

The next byte is reserved for future use. The next two bytes are the node address of the desired kit to which the data has to be sent over mesh network. If this value is set to broadcast address (00:00), then the data is accepted by each node of the mesh network. Else, the data is accepted by the node whose node address matches the address set in this data.

The next four bytes are the RGB color and intensity value. This is the data that is to be transmitted/relayed over the mesh network.

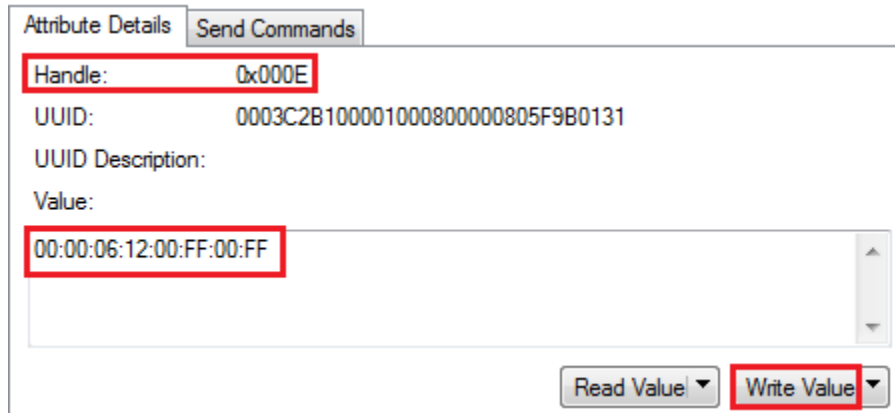
Figure 20. Data packet format

0	1	2	3	4	5	6	7
Address Scheme	Reserve	ADDR 1	ADDR 2	Red	Green	Blue	Intensity

The color and intensity value can range from **0x00 to 0xFF**.

- 21) Write the required Address scheme value and 2 byte node address followed by 4 bytes of color value and click on **Write Value**.

Figure 21. Sending Full Intensity Green Color data to node with address 06:12



Attribute Details Send Commands

Handle: 0x000E

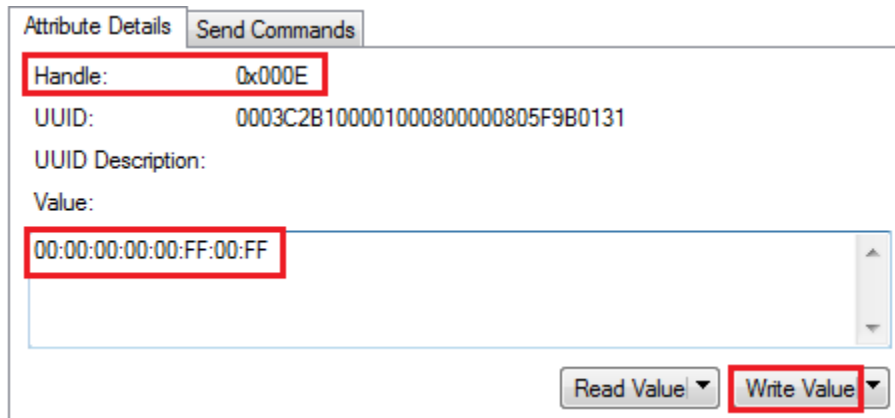
UUID: 0003C2B100001000800000805F9B0131

UUID Description:

Value: 00:00:06:12:00:FF:00:FF

Read Value Write Value

Figure 22. Sending Full Intensity Green Color data to all nodes (Broadcasting with 00:00)



Attribute Details Send Commands

Handle: 0x000E

UUID: 0003C2B100001000800000805F9B0131

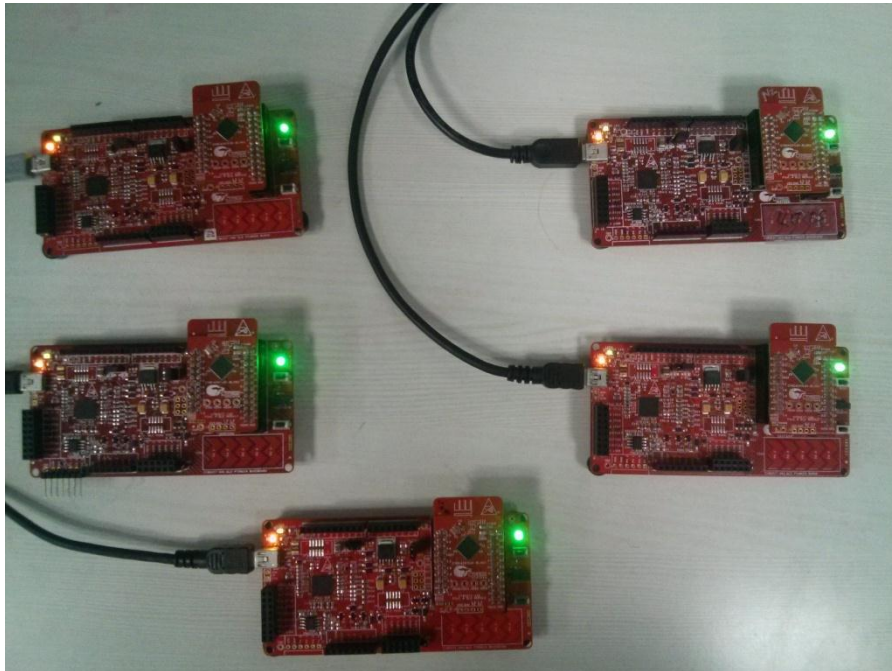
UUID Description:

Value: 00:00:00:00:00:FF:00:FF

Read Value Write Value

- 22) As soon as the color value is written, the device will automatically disconnect. According to the node address sent, the color will be set on a particular node of the network or to all the nodes.

Figure 23. RGB LED data relayed to all nodes



23) The devices will re-appear on the list on CySmart PC Tool after about 5 seconds. Reconnect to change the color and send new data.

24) Bring a new kit with mesh project and change color as part of this network.

### Using Button press:

The project also supports sending new RGB color data to all the nodes using the user button (SW2) on the BLE Pioneer Kit. It simulates local sensor trigger when it has a new data and wishes to send it over the mesh network.

- 1) For new Kits, follow step 1-15 as mentioned above to assign node address to each node. Once node addresses are set for the kits, these steps need not be repeated as the node address is stored in SLFASH and remains valid even after reset.

**Note:** Unless all the nodes are not assigned a node address, the node will not relay any data in mesh network.

- 2) Once all the kits have been powered, press the User Button (SW2) on any of the BLE Pioneer Kit/node. The color on that node will change along with same color propagated to all other nodes of the mesh network (broadcast). Wait for the Central timeout (about ~4-5 seconds) and press the user button again to send a new RGB color value.

3) The color sequence on button press is as following:

*Red -> Green -> Blue -> Yellow -> Cyan -> Purple -> White -> White with Half Intensity -> Red.*



## 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.