ECE9407– PSoC 4 BLE Lab #4
CapSense Design with BLE Connectivity

Name: An He, Amr Gaballab
Number: 250847871, 250776065
Data: 2016-06-22
Instructor: Dr. Ahmed Hussein

1. Lab4 Task Description

This lab aims to hands on how to create a Custom Profile by implementing an RGB LED controller through BLE. We will also learn how to combine CapSense and BLE in a system, by designing a slider application. Finally, we can use mobile phone to test the result by sliding the CapSense and to check the LED by touch the RGB App.

In this lab, we will try to use two Custom Service called RGB LED and CapSense Slider respectively.

2. Equipment and Objectives

Equipment:
Hardware: BLE Pioneer Kit (CY8CKIT-042-BLE)
Software : PSoC Creator 3.3 SP1, CySmart 1.2, and iOS app

Objectives:
There are two main tasks RGB LED and CapSense.

1. Adjust RGB LED color and intensity using the PRiSM Component
2. Implement a custom BLE Profile with a custom Service to send RGB LED color and intensity over BLE
3. Implement a Custom Service to send CapSense slider data over BLE
4. Use the CySmart tool and mobile app to validate the operation

Block Diagram:

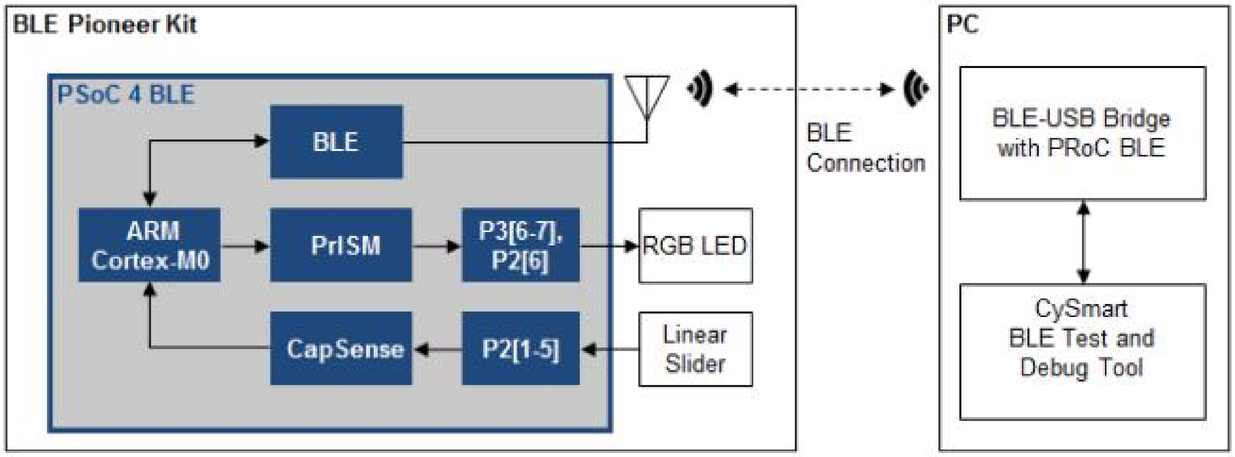


Figure 1: Block Diagram and Overview

Preparing and Understanding

This lab implements two Custom Services – one to control the RGB LED on the kit over BLE, and the other to send CapSense slider information over BLE.

Custom Service for RGB LED

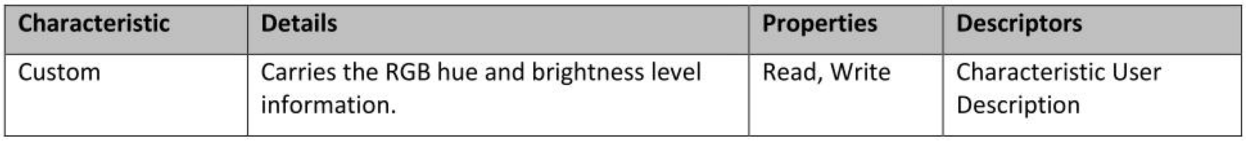
The Service has one Characteristic, which is given in Table 1. This Characteristic contains 4 bytes as described in Table 2.

One byte each for the Red,Green and Blue LEDs and another byte for the overall LED intensity.

The following information for RGB LED is very important.

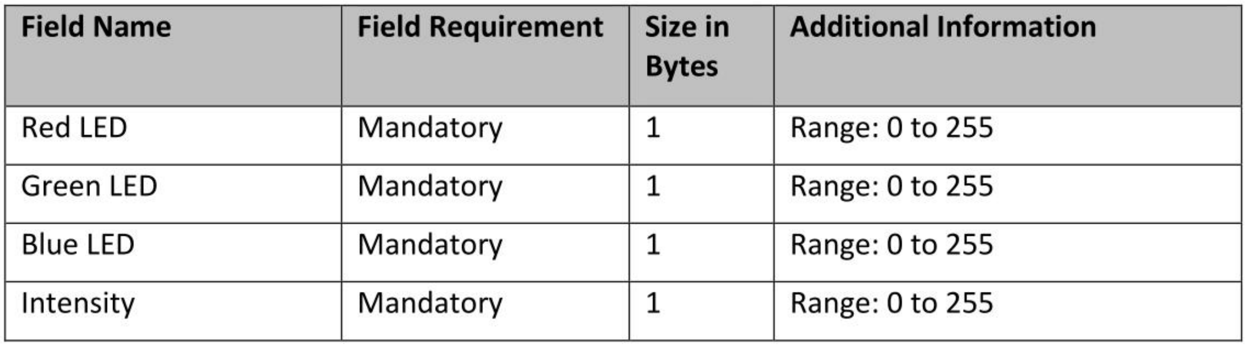
Custom Service UUID (16-bit): 0xCBBB

Custom Characteristic UUID (16-bit): 0xCBB1



Characteristic	Details	Properties	Descriptors
Custom	Carries the RGB hue and brightness level information.	Read, Write	Characteristic User Description

Table 1: Custom Service for RGB LED

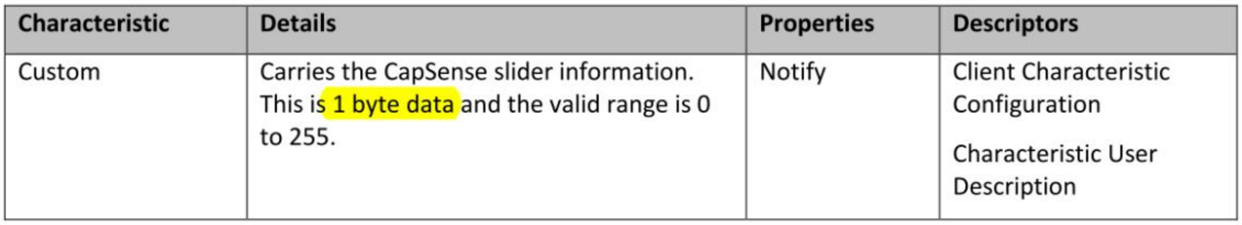


Field Name	Field Requirement	Size in Bytes	Additional Information
Red LED	Mandatory	1	Range: 0 to 255
Green LED	Mandatory	1	Range: 0 to 255
Blue LED	Mandatory	1	Range: 0 to 255
Intensity	Mandatory	1	Range: 0 to 255

Table 2: Custom Characteristic Fields for RGB LED

Custom Service for CapSense Slider

The CapSense Service has one Characteristic for implementing sliders. See Table 3 for details.



Characteristic	Details	Properties	Descriptors
Custom	Carries the CapSense slider information. This is 1 byte data and the valid range is 0 to 255.	Notify	Client Characteristic Configuration Characteristic User Description

Table 3: Custom Service for CapSense Slider

The following information for CapSense Slider is very important.

Custom Service UUID (16-bit): 0xCAB5

Custom Characteristic UUID (16-bit): 0xCAA2

Also, the GATT Client Characteristic Configuration Descriptor (or CCCD) is used by the GATT Server to identify whether notifications are enabled or not. If the CCCD has a value of ‘1’, the notifications are enabled. A value of ‘0’ indicates disabled notifications.

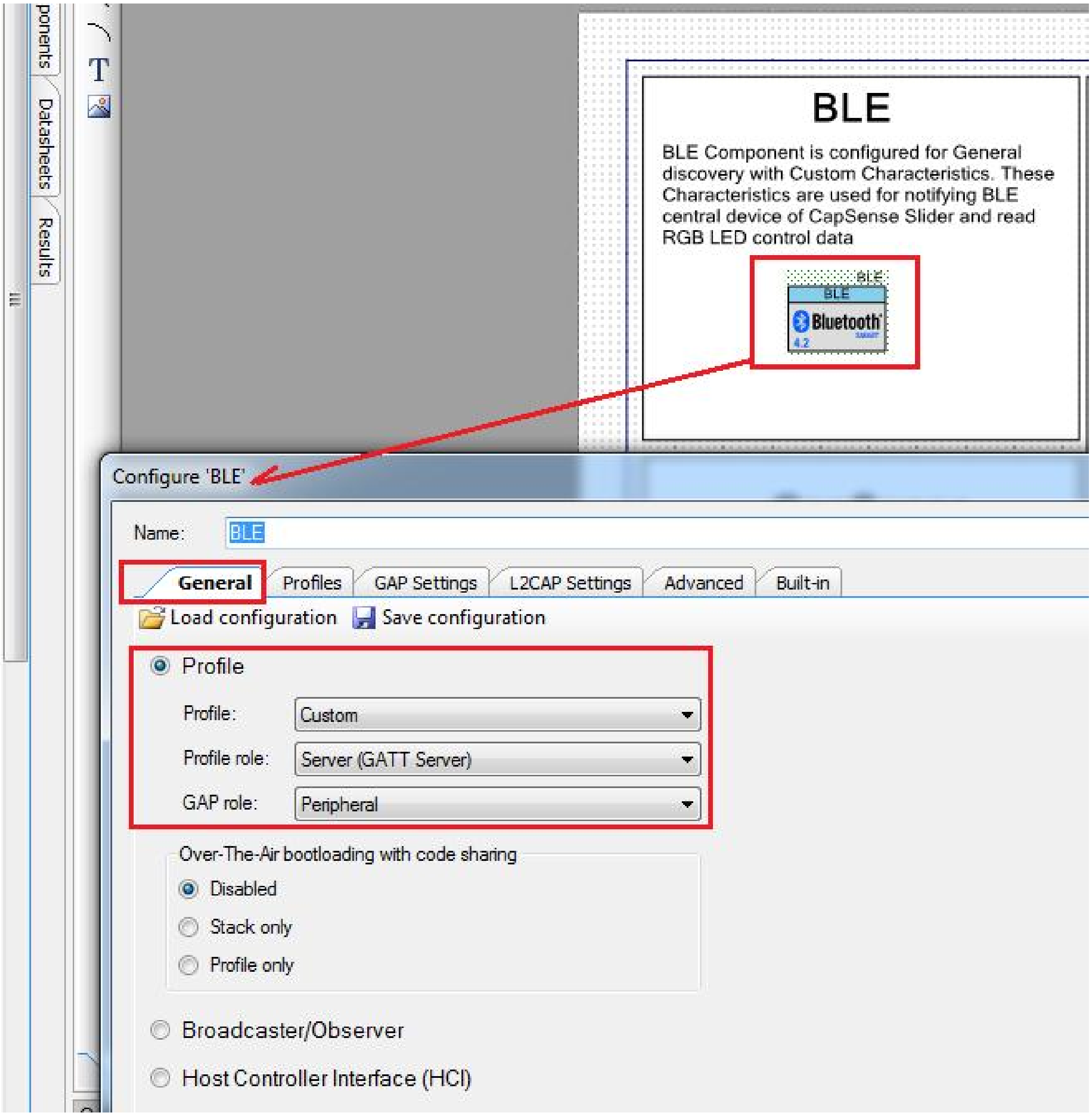
To control the LED, the Universal Digital Blocks (UDBs) are used. A UDB can be configured as a PrISM Component (short for Precision Illumination Signal Modulation) which can drive two outputs with a Linear Feedback Shift Register. Two PrISM blocks are used to drive the three LEDs on the kit at any intensity desired. Together, these PrISM blocks control the hue and brightness of the RGB LED. This Component provides better EMI performance than using a standard PWM to control intensity.

3. Steps and Results

3.1 Open the General Tab of BLE component

Set the Profile to Custom and the Profile role to Server (GATT Server).

Set the GAP role to Peripheral as Figure 2.



BLE

BLE Component is configured for General discovery with Custom Characteristics. These Characteristics are used for notifying BLE central device of CapSense Slider and read RGB LED control data

Configure 'BLE'

Name: BLE

● Profile

Profile: Custom

Profile role: Server (GATT Server)

GAP role: Peripheral

Over-The-Air bootloading with code sharing

● Disabled

○ Stack only

○ Profile only

○ Broadcaster/Observer

○ Host Controller Interface (HCI)

Figure 2: BLE Component - General Tab

3.2 BLE - Profiles Tab

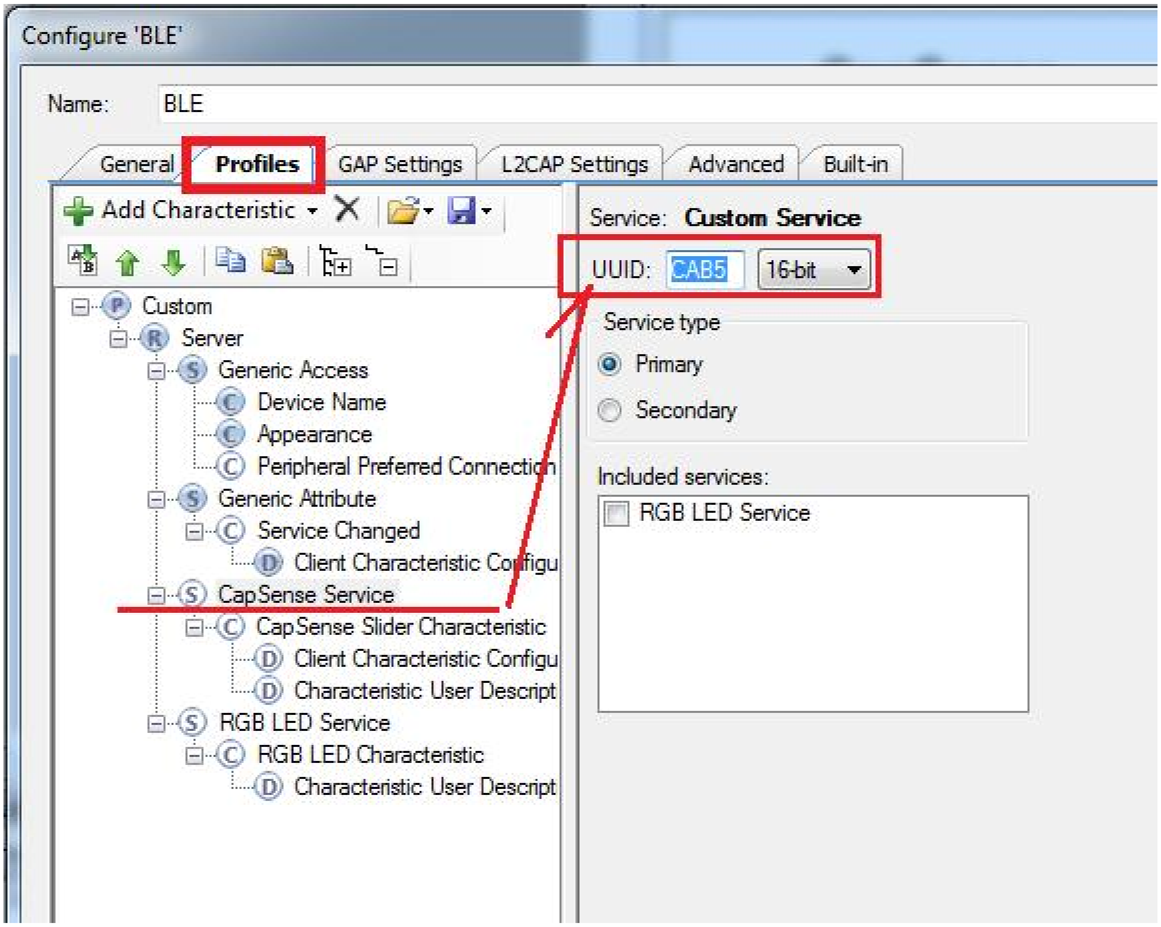
Configure that for CapSense Slider, and add another Custom Service for RGB LED. Follow these steps:

Custom Service for CapSense Slider

a. Configure the Custom Service for CapSense Slider.

b. Custom Service - Rename this service to CapSense Service by right-clicking on Custom Service and choosing Rename.

c. Set the UUID length as 16-bit and value as **0xCAB5**



Name: BLE

- Custom
  - Server
    - Generic Access
      - Device Name
      - Appearance
      - Peripheral Preferred Connectio
    - Generic Attribute
      - Service Changed
        - Client Characteristic Configu
    - CapSense Service
      - CapSense Slider Characteristic
        - Client Characteristic Configu
        - Characteristic User Descript
    - RGB LED Service
      - RGB LED Characteristic
        - Characteristic User Descript

Service: **Custom Service**

UUID: CAB5 16-bit

Service type

● Primary

○ Secondary

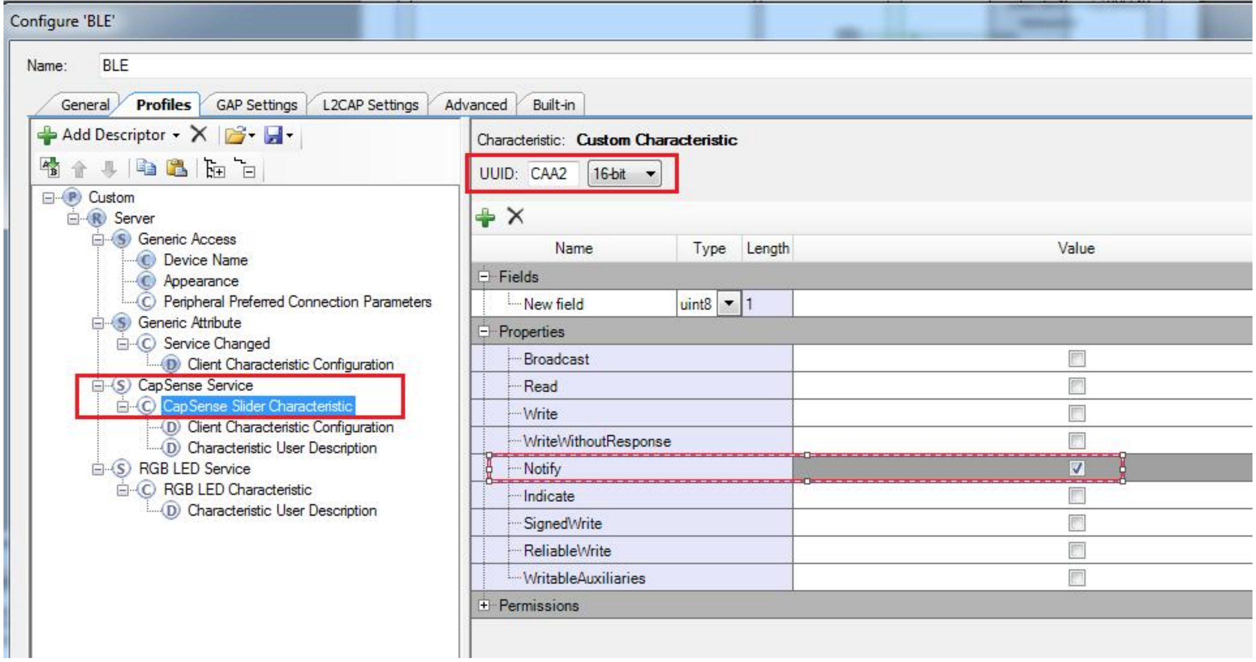
Included services:

□ RGB LED Service

Figure 3: Configuring Custom Service UUID for CapSense Slider

d. Custom Characteristic

Rename Custom Characteristic to CapSense Slider Characteristic.
Set the UUID (16-bit) to 0xCAA2.
Enable Notify in the Properties for the Characteristic. See Figure 4.



Name: BLE

- Custom
  - Server
    - Generic Access
      - Device Name
      - Appearance
      - Peripheral Preferred Connection Parameters
    - Generic Attribute
      - Service Changed
        - Client Characteristic Configuration
    - CapSense Service
      - CapSense Slider Characteristic
        - Client Characteristic Configuration
        - Characteristic User Description
    - RGB LED Service
      - RGB LED Characteristic
        - Characteristic User Description

Characteristic: **Custom Characteristic**

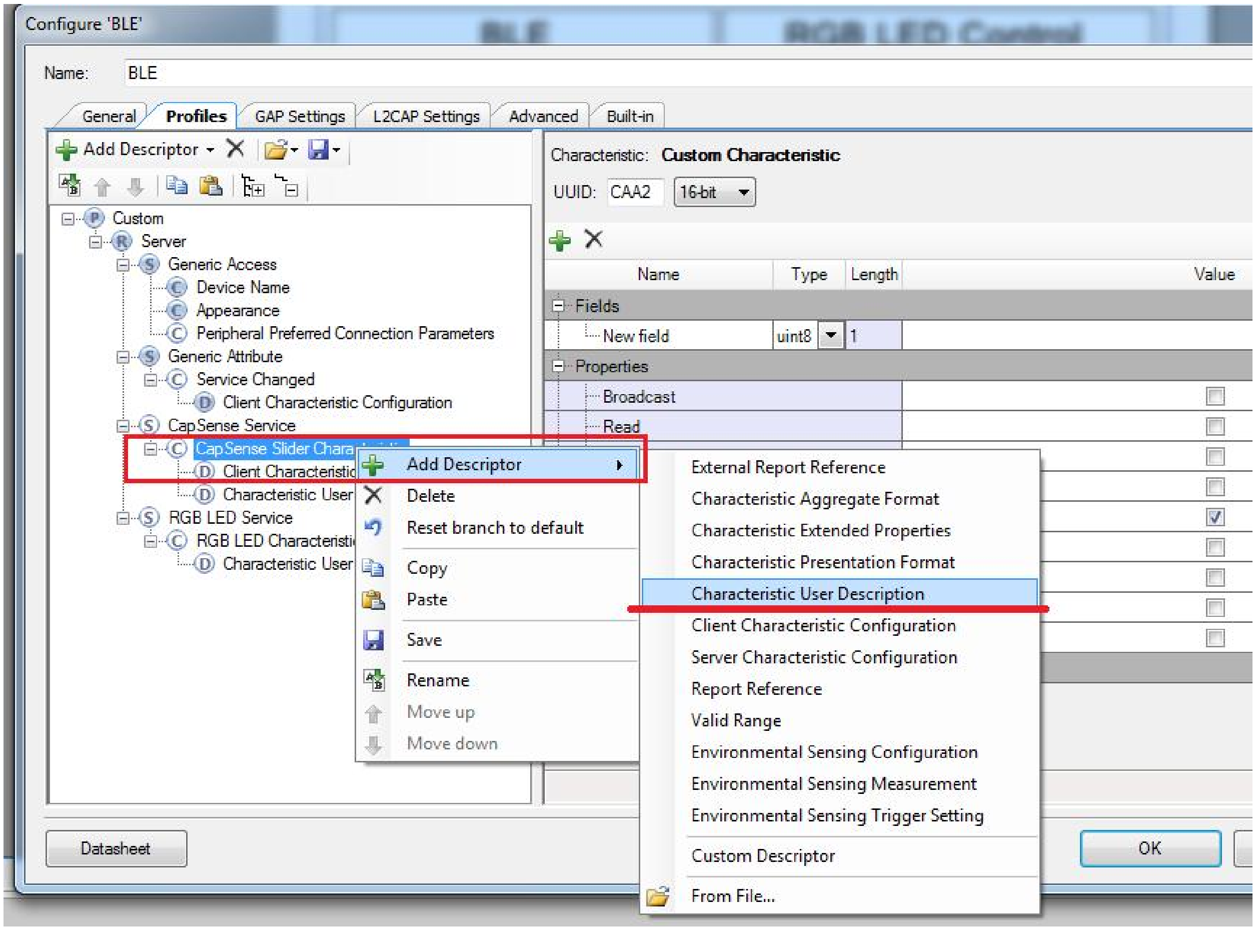
UUID: CAA2 16-bit

Name	Type	Length	Value
Fields			
New field	uint8	1	
Properties			
Broadcast			□
Read			□
Write			□
WriteWithoutResponse			□
Notify			☑
Indicate			□
SignedWrite			□
ReliableWrite			□
WritableAuxiliaries			□
Permissions			

Figure 4: Configuring Custom Characteristic for CapSense Slider

e. Delete the Custom Descriptor by right-clicking on it and selecting Delete.

f. Click on the **CapSense Slider Characteristic**, and using the **Add Descriptor** drop down on the top, add the **Characteristic User Description Descriptor**. See Figure 5.



Name: BLE

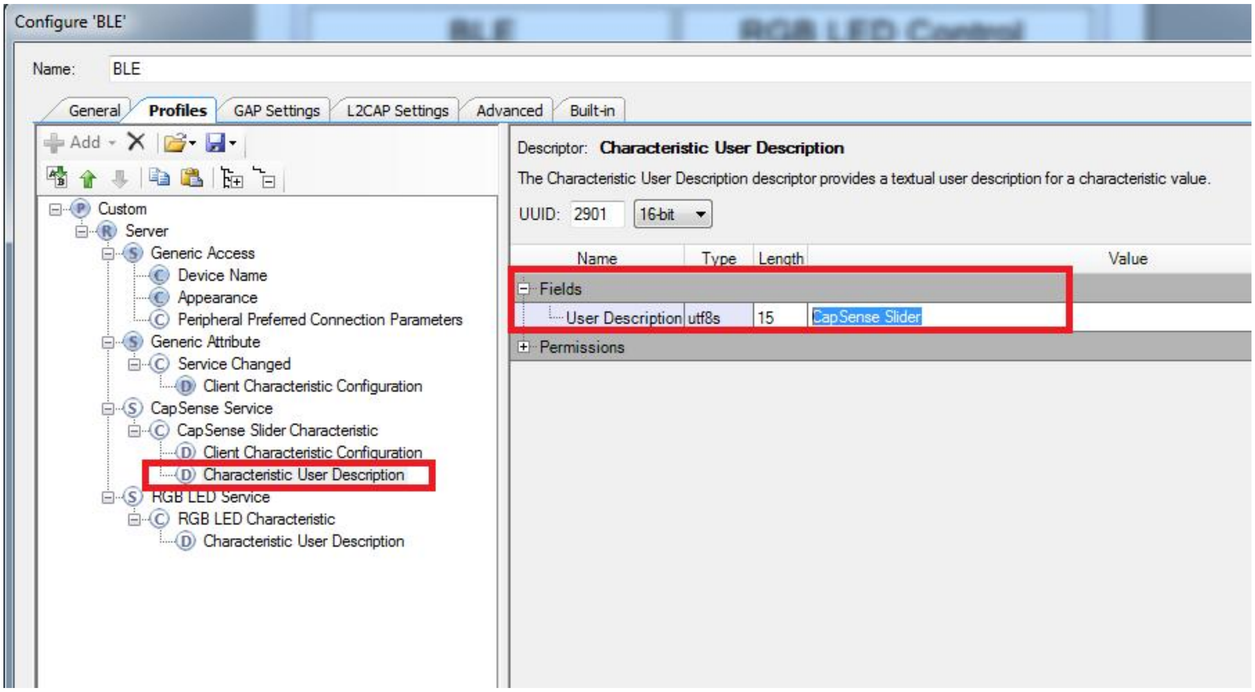
Characteristic: Custom Characteristic

UUID: CAA2 16-bit

Figure 5: Add Descriptor for the CapSense Service

g. Set the value of **Characteristic User Description** to **CapSense Slider**.

See Figure 6.



Name: BLE

Descriptor: **Characteristic User Description**

The Characteristic User Description descriptor provides a textual user description for a characteristic value.

UUID: 2901 16-bit

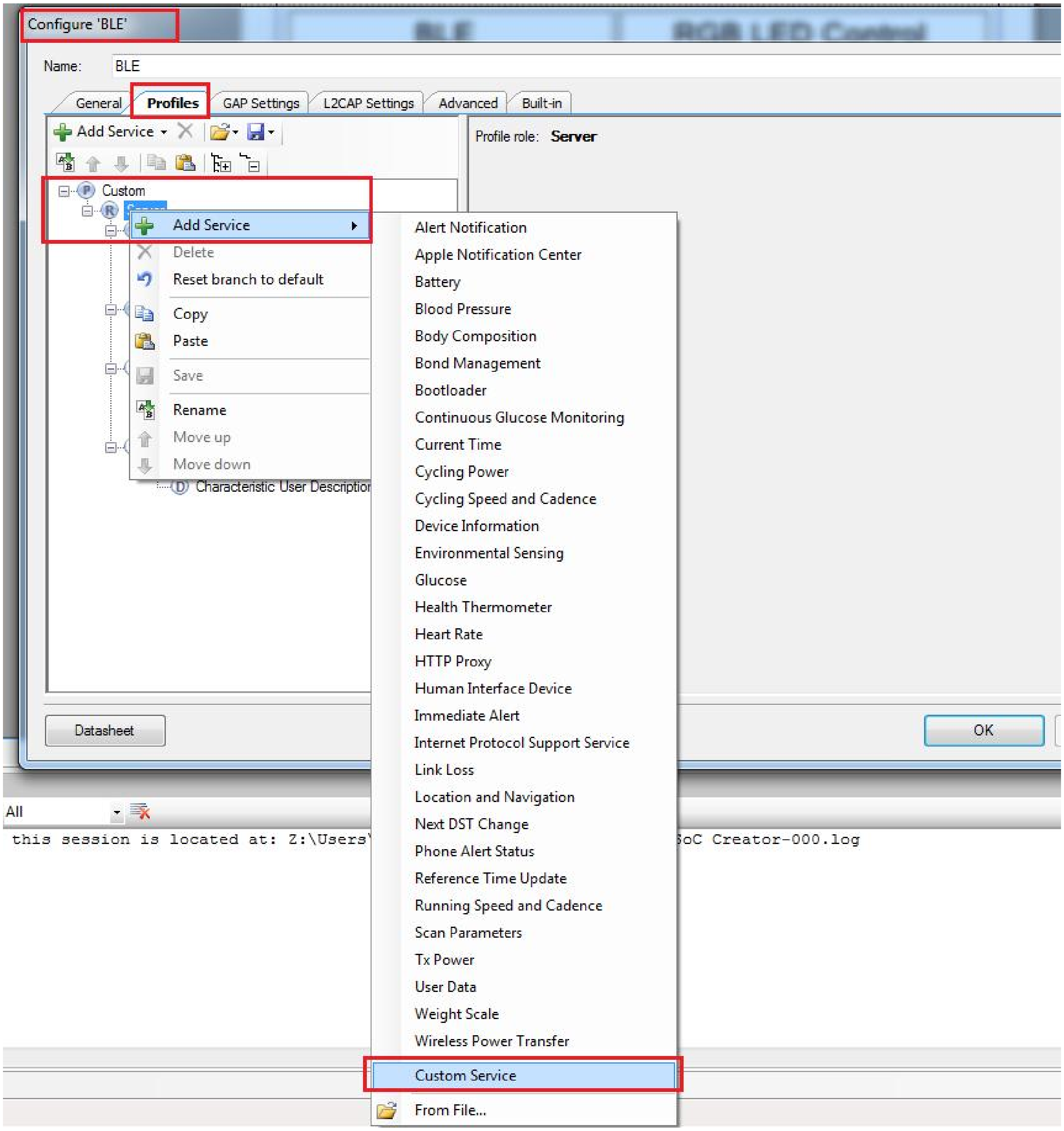
Name	Type	Length	Value
Fields			
User Description	utf8s	15	CapSense Slider
Permissions			

Figure 6: Characteristic User Description for CapSense Slider Characteristic

Till this step, the setting for CapSense Slider is finished

3.3 Custom Service for RGB LED

a. Add a new Custom Service to the Profiles page by right-clicking on Server and then selecting **Add Service -> Custom Service**. See Figure 7.



```
this session is located at: Z:\Users\                                    SoC Creator-000.log
```

Figure 7: Add Custom Service for RGB LED

b. Rename this **Custom Service to RGB** LED Service.

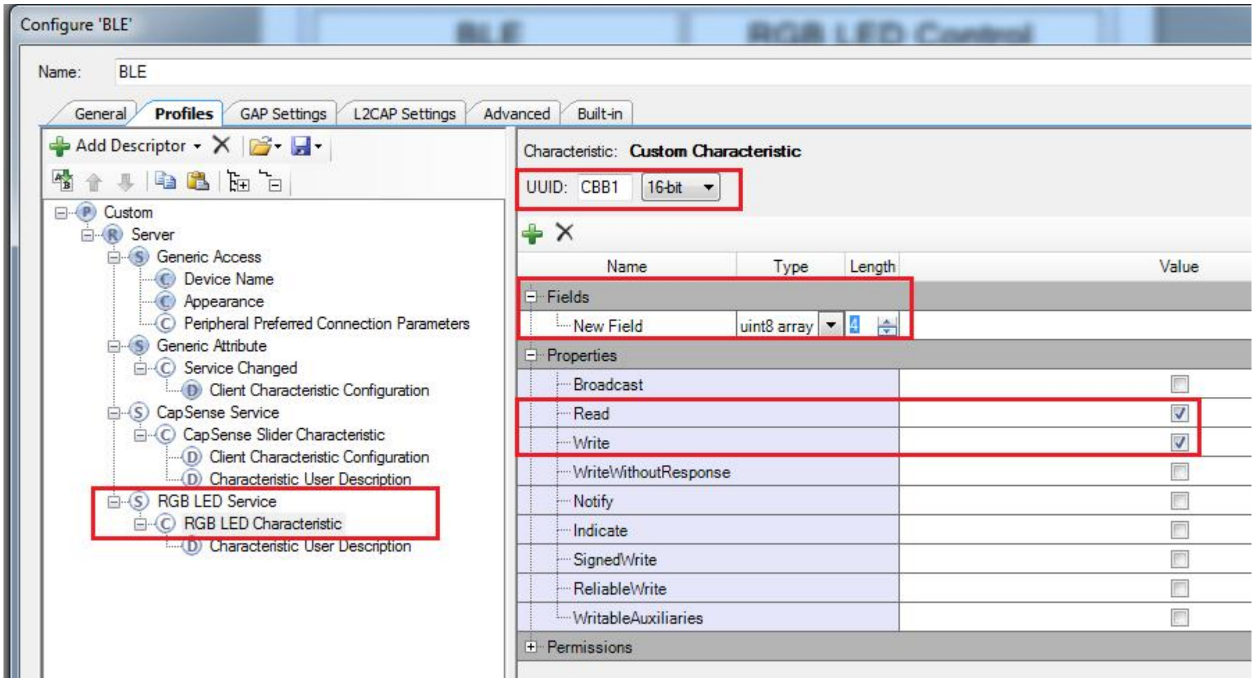
Set its **UUID (16-bit) to 0xCBBB**.

c. Rename the **Custom Characteristic** in this service to **RGB LED Characteristic**. Set its **UUID (16-bit) to 0xCBB1.** See Figure 8.

d. For the **RGB LED Characteristic**, change the Type of **New Field** under the Fields column to **uint8 array** and the **Length to 4**, see Figure 8. This is because the Characteristic contains
4 bytes – Red LED, Green LED, Blue LED, and overall intensity.

Field Name	Field Requirement	Size in Bytes	Additional Information
Red LED	Mandatory	1	Range: 0 to 255
Green LED	Mandatory	1	Range: 0 to 255
Blue LED	Mandatory	1	Range: 0 to 255
Intensity	Mandatory	1	Range: 0 to 255

e. Enable **Read and Write** in the Characteristic properties. See Figure 8.



Configure 'BLE'

Name: BLE

Characteristic: Custom Characteristic

UUID: CBB1 16-bit

Name	Type	Length	Value
Fields			
New Field	uint8 array	4	
Properties			
Broadcast			□
Read			☑
Write			☑
WriteWithoutResponse			□
Notify			□
Indicate			□
SignedWrite			□
ReliableWrite			□
WritableAuxiliaries			□
Permissions			

General
Profiles
GAP Settings
L2CAP Settings
Advanced
Built-in
Add Descriptor
Custom
Server
Generic Access
Device Name
Appearance
Peripheral Preferred Connection Parameters
Generic Attribute
Service Changed
Client Characteristic Configuration
CapSense Service
CapSense Slider Characteristic
Client Characteristic Configuration
Characteristic User Description
RGB LED Service
RGB LED Characteristic
Characteristic User Description

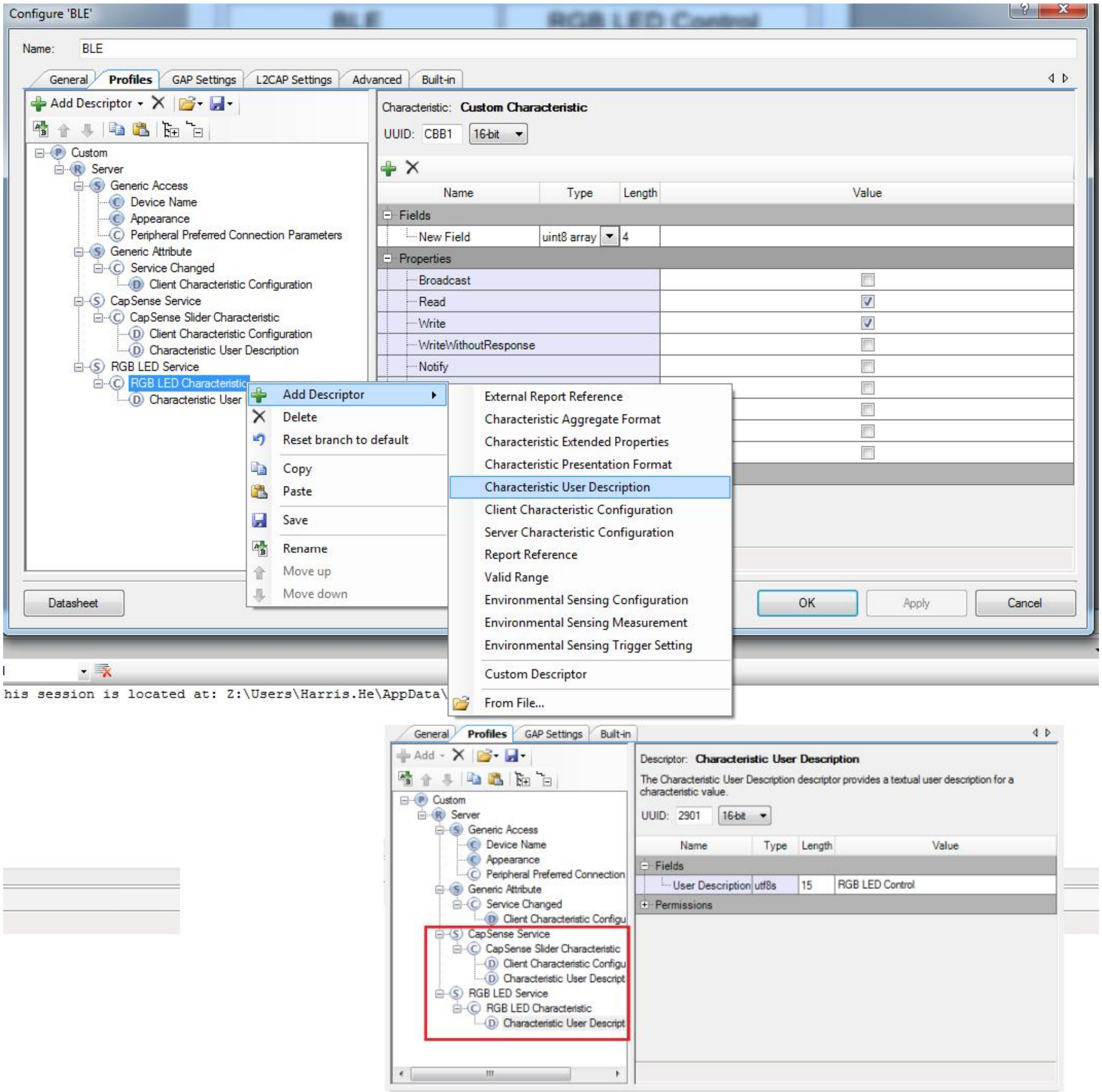
Figure 8: Configuring Characteristic for RGB LED Service

f. **Delete** the **Custom Descriptor.**

g. **Add** the **Characteristic User Description Descriptor**.

Give it a value of **RGB LED Control**.

h. The final result can be found from Figure 9.


Configure 'BLE'

Name: BLE

Characteristic: **Custom Characteristic**

UUID: CBB1 16-bit

Name	Type	Length	Value
Fields			
New Field	uint8 array	4	
Properties			
Broadcast			☐
Read			☑
Write			☑
WriteWithoutResponse			☐
Notify			☐

```
his session is located at: Z:\Users\Harris.He\AppData\
```

Descriptor: **Characteristic User Description**

The Characteristic User Description descriptor provides a textual user description for a characteristic value.

UUID: 2901 16-bit

Name	Type	Length	Value
Fields			
User Description	utf8s	15	RGB LED Control
Permissions			

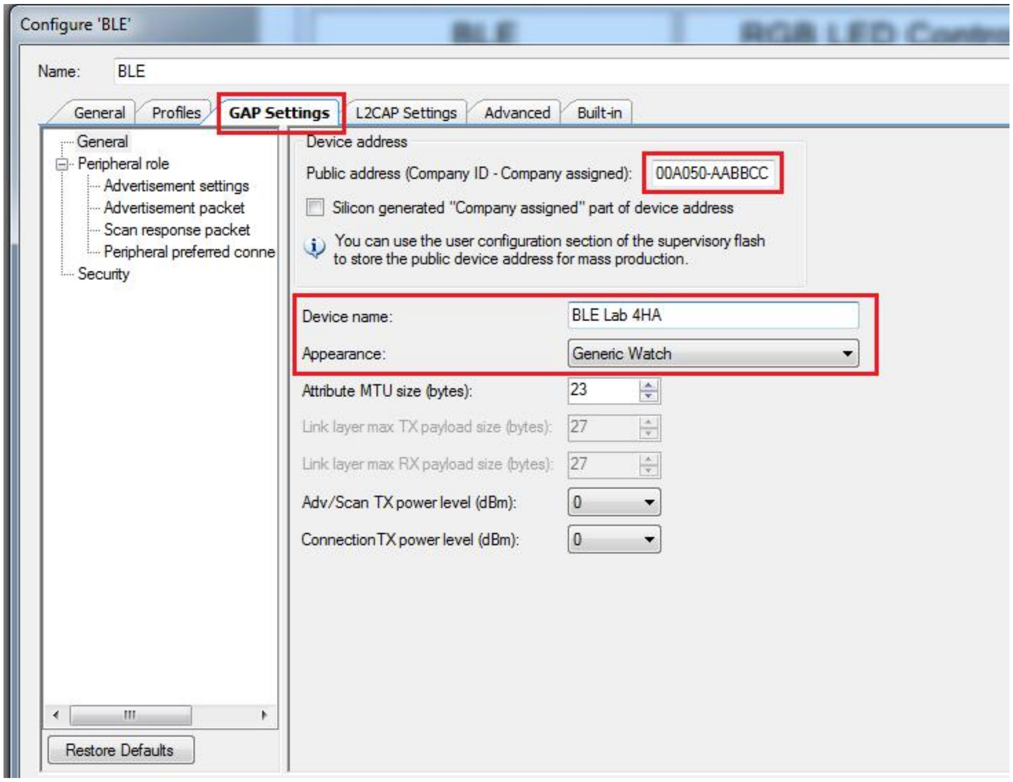
General
Profiles
GAP Settings
L2CAP Settings
Advanced
Built-in
Add Descriptor
Custom
Server
Generic Access
Device Name
Appearance
Peripheral Preferred Connection Parameters
Generic Attribute
Service Changed
Client Characteristic Configuration
CapSense Service
CapSense Slider Characteristic
Client Characteristic Configuration
Characteristic User Description
RGB LED Service
RGB LED Characteristic
Characteristic User
Add Descriptor
Delete
Reset branch to default
Copy
Paste
Save
Rename
Move up
Move down
External Report Reference
Characteristic Aggregate Format
Characteristic Extended Properties
Characteristic Presentation Format
Characteristic User Description
Client Characteristic Configuration
Server Characteristic Configuration
Report Reference
Valid Range
Environmental Sensing Configuration
Environmental Sensing Measurement
Environmental Sensing Trigger Setting
Custom Descriptor
From File...
Datasheet
OK
Apply
Cancel

Figure 9: Profiles Tab for Lab 4

Till this step, the setting of Cap Sense and RGB LED in BLE Profiles page has been finished.

3.4 Now the ‘Profile’ of BLE setting has been finished. The GAP Settings Tab will be the focus.

3.4.1 General - Set the Device Address, Device name, and Appearance as shown in Figure 10.



Configure 'BLE'

Name: BLE

Device address

Public address (Company ID - Company assigned): 00A050-AABBCC

☐ Silicon generated "Company assigned" part of device address

You can use the user configuration section of the supervisory flash to store the public device address for mass production.

Device name: BLE Lab 4HA

Appearance: Generic Watch

Attribute MTU size (bytes): 23

Link layer max TX payload size (bytes): 27

Link layer max RX payload size (bytes): 27

Adv/Scan TX power level (dBm): 0

Connection TX power level (dBm): 0

Figure 10: GAP Settings - General

3.4.2 Security – Configure as below:

a. Security mode: Select Mode 1 security

b. Security level: Select No Security (No Authentication, No Encryption)

c. I/O Capabilities: Set this to No Input No Output

d. Pairing method: Select Just works

e. Bonding requirement: Set this to No Bonding

f. Encryption key size (bytes): Leave this parameter to the default value of 16

Till this step, the BLE setting has been finished.

The next step will turn to CapSense CSE Component setting.

3.5 Add the **CapSense CSD Component** to the schematic.

This Component is used to sense finger position on the slider on the BLE Pioneer Kit.

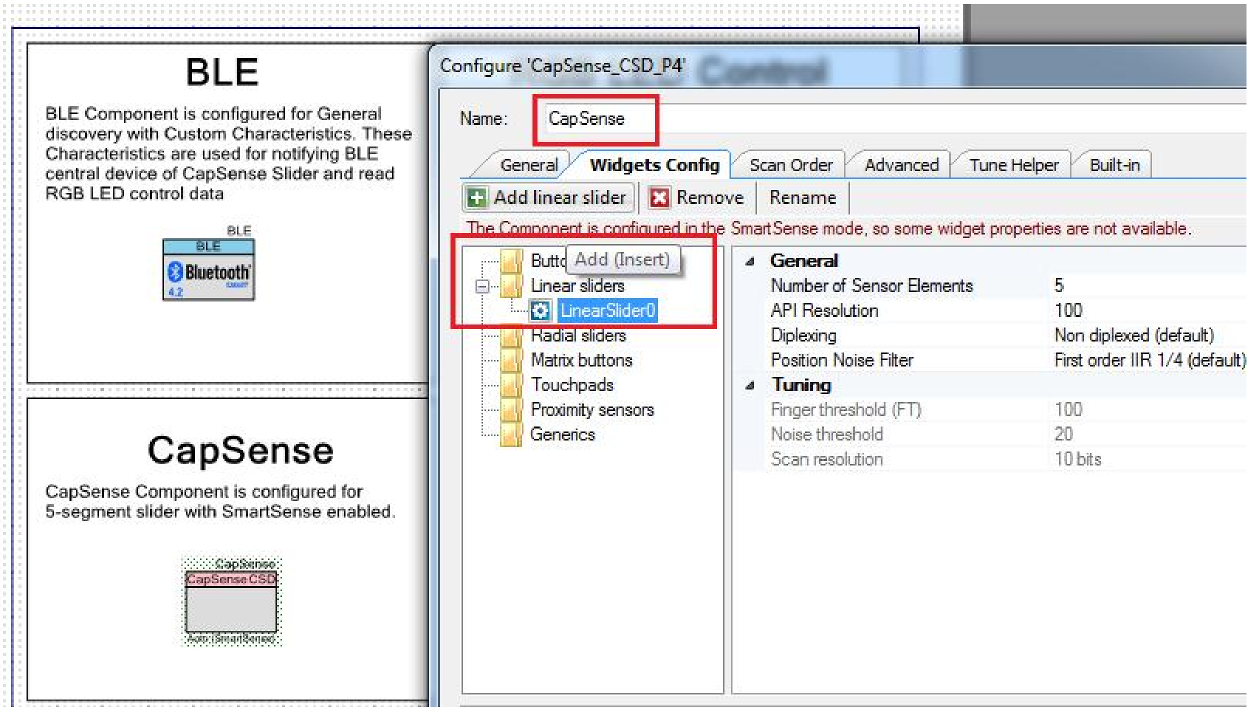
3.5.1 Configure the CapSense CSD Component's

Set its name to be **CapSense.**

On the General tab, leave the default settings.

The Component automatically tunes its parameters for the best performance, using **SmartSense Auto-Tuning** – an algorithm that sets, monitors and continuously maintains optimal capacitive sensor performance.

3.5.2 On the **Widgets Config tab**, add a linear slider by clicking Linear sliders and then the Add linear slider button, as shown in Figure 12.



BLE

BLE Component is configured for General discovery with Custom Characteristics. These Characteristics are used for notifying BLE central device of CapSense Slider and read RGB LED control data

CapSense

CapSense Component is configured for 5-segment slider with SmartSense enabled.

Configure 'CapSense_CSD_P4'

Name: CapSense

The Component is configured in the SmartSense mode, so some widget properties are not available.

General	
Number of Sensor Elements	5
API Resolution	100
Diplexing	Non diplexed (default)
Position Noise Filter	First order IIR 1/4 (default)
Tuning	
Finger threshold (FT)	100
Noise threshold	20
Scan resolution	10 bits

Figure 12: CapSense Component Configuration - Adding a Linear Slider

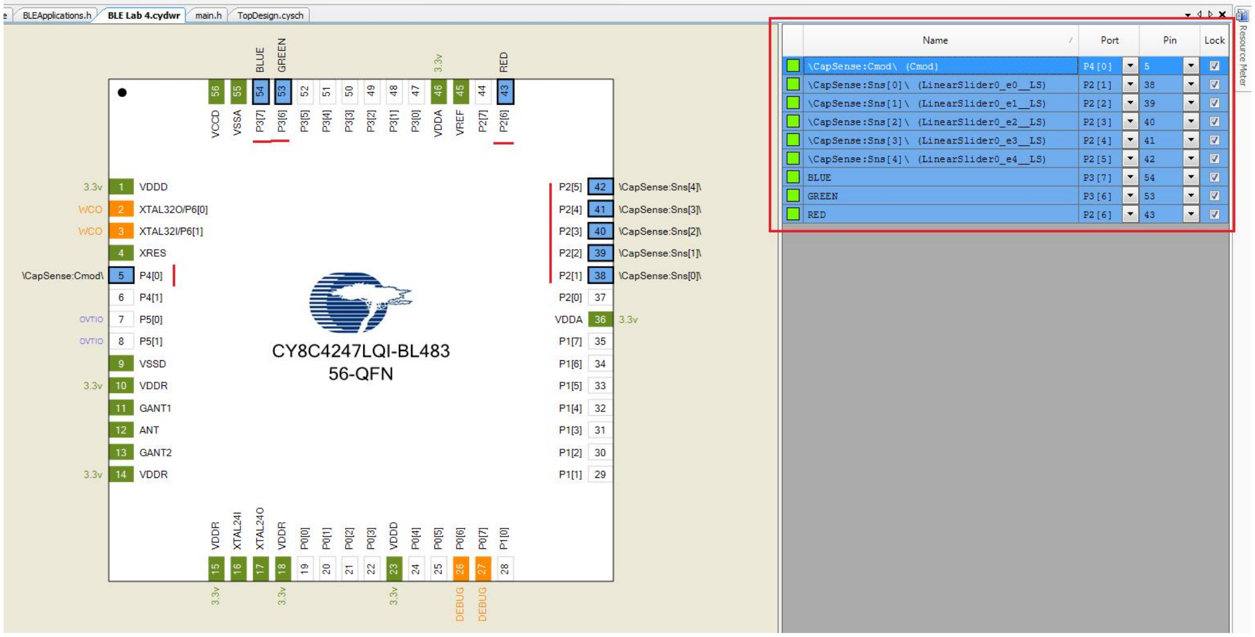
Keep other setting as defaults.

Till this step, all the configuration of components (BLE & CapSense) has been finished.

3.6 DWR Configuration

Open the DWR file. On the Pins tab, assign the segments of the linear slider to pins P2[1], P2[2], P2[3], P2[4], and P2[5], in increasing order.

The CMOD capacitor is on P4[0]. The three LEDs are already set to the correct device pins.



	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	☑

3.7 Firmware and Program

The firmware consists of these high-level blocks:

CapSense Slider

RGB LED

BLE

The firmware flow is shown in Figure 15.

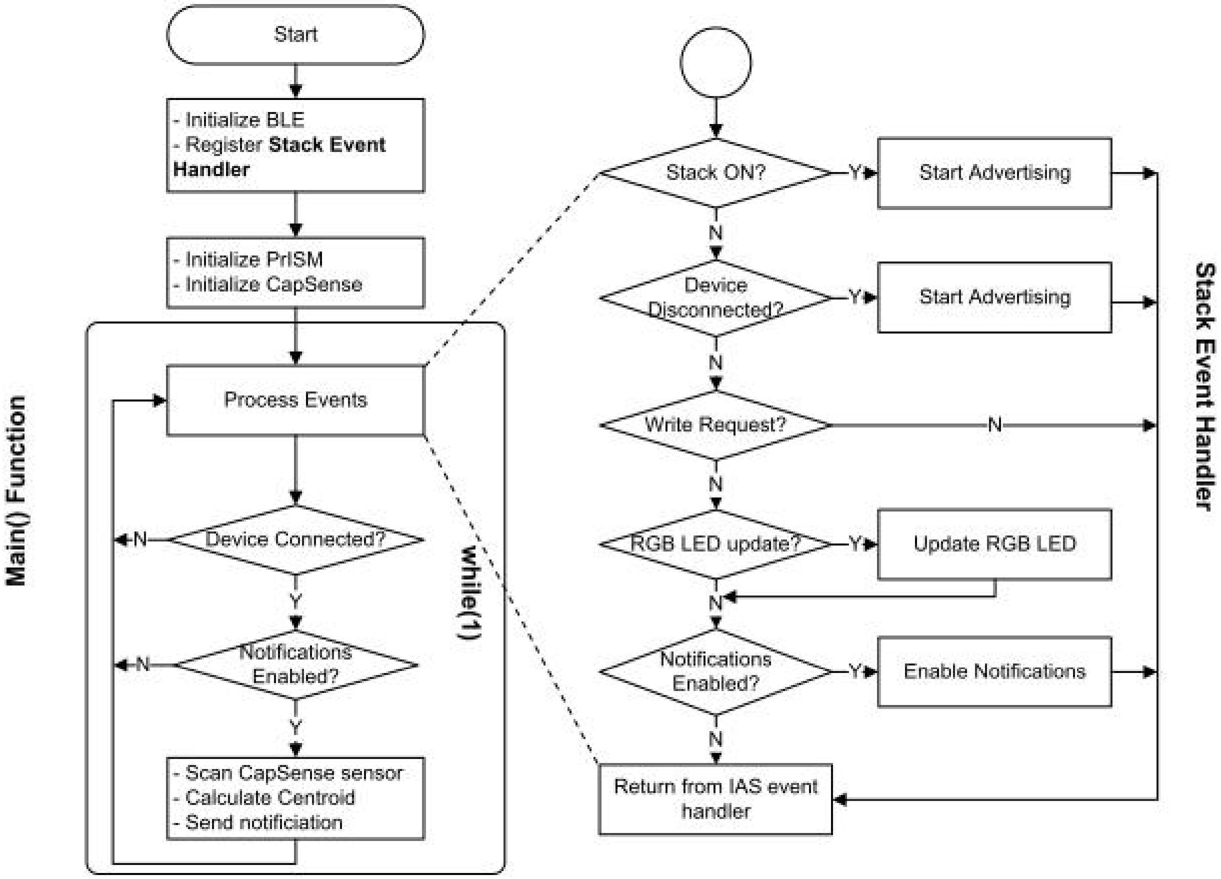


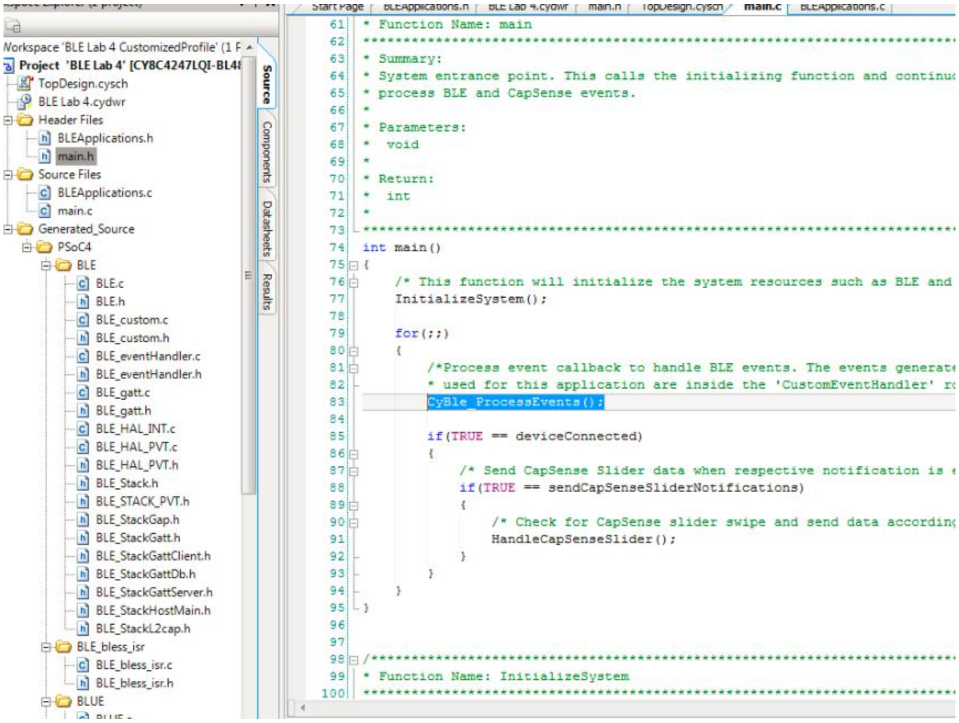
Figure 15: Firmware Firmware

3.8 Evaluation and Test

Build the project to generate the hex file, and **Program** it to the kit.

Testing with **CySmart Central Emulation Tool**

- Open **CySmart 1.2** and Connect it to the **BLE-USB Bridge.**
- **Start Scan and Connect** to your GATT Server device.
- **Discover all Attributes** and then scroll down the Attribute list to the RGB LED Characteristic (it has the UUID 0xCBB1). See Figure 16,17



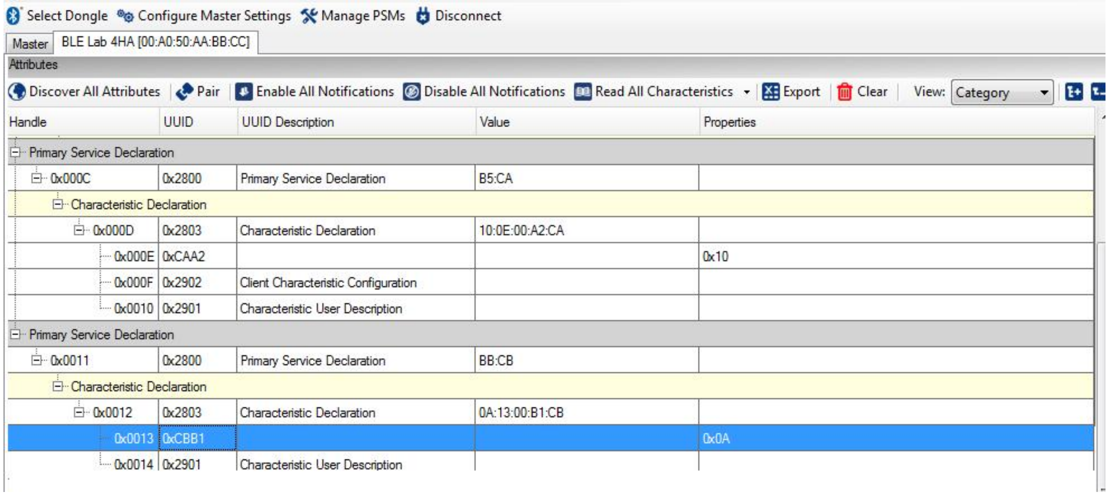
```
* Function Name: main
*************************************************************************
* Summary:
* System entrance point. This calls the initializing function and continu
* process BLE and CapSense events.
*
* Parameters:
*  void
*
* Return:
*  int
*
*************************************************************************
int main()
{
    /* This function will initialize the system resources such as BLE and
    InitializeSystem();

    for(;;)
    {
        /*Process event callback to handle BLE events. The events generate
        * used for this application are inside the 'CustomEventHandler' ro
        CyBle_ProcessEvents();

        if(TRUE == deviceConnected)
        {
            /* Send CapSense Slider data when respective notification is e
            if(TRUE == sendCapSenseSliderNotifications)
            {
                /* Check for CapSense slider swipe and send data accordin
                HandleCapSenseSlider();
            }
        }
    }
}


/*************************************************************************
* Function Name: InitializeSystem
*************************************************************************
```

Figure 16. Main program



Handle	UUID	UUID Description	Value	Properties
Primary Service Declaration				
0x000C	0x2800	Primary Service Declaration	B5:CA	
Characteristic Declaration				
0x000D	0x2803	Characteristic Declaration	10:0E:00:A2:CA	
0x000E	0xCAA2			0x10
0x000F	0x2902	Client Characteristic Configuration		
0x0010	0x2901	Characteristic User Description		
Primary Service Declaration				
0x0011	0x2800	Primary Service Declaration	BB:CB	
Characteristic Declaration				
0x0012	0x2803	Characteristic Declaration	0A:13:00:B1:CB	
0x0013	0xCBB1			0x0A
0x0014	0x2901	Characteristic User Description		

Figure 17. CySmart - RGB LED Characteristic

Write a 4 byte value to this Characteristic on the right and notice the corresponding color and intensity of the RGB LED on the kit.

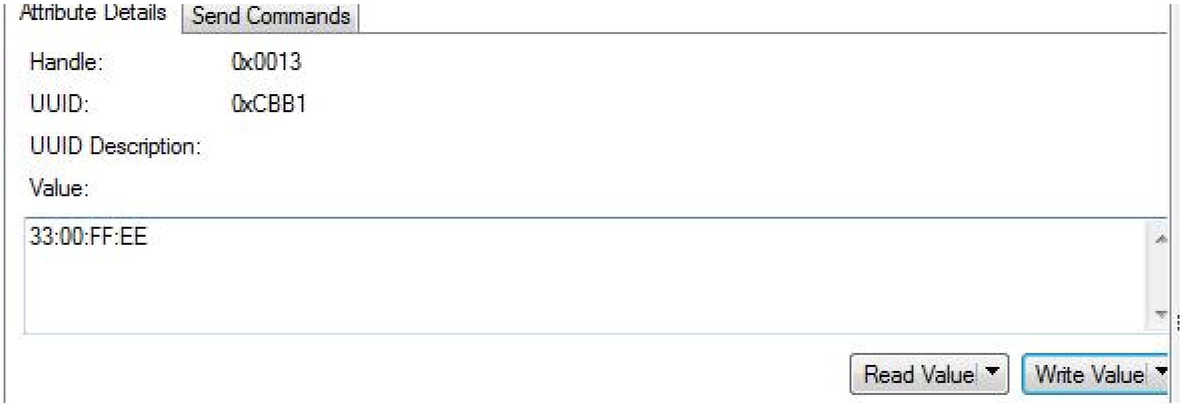
Byte 0 corresponds to the Red color,

Byte 1 corresponds to the Green color,

Byte 2 corresponds to the Blue color, and

Byte 3 corresponds to the intensity.

As the Figure 18 shows, writing 33:00:FF:EE to this Characteristic turns on the Red and Blue LED with almost full(EE) intensity. See Figure 17.



Handle: 0x0013

UUID: 0xCBB1

UUID Description:

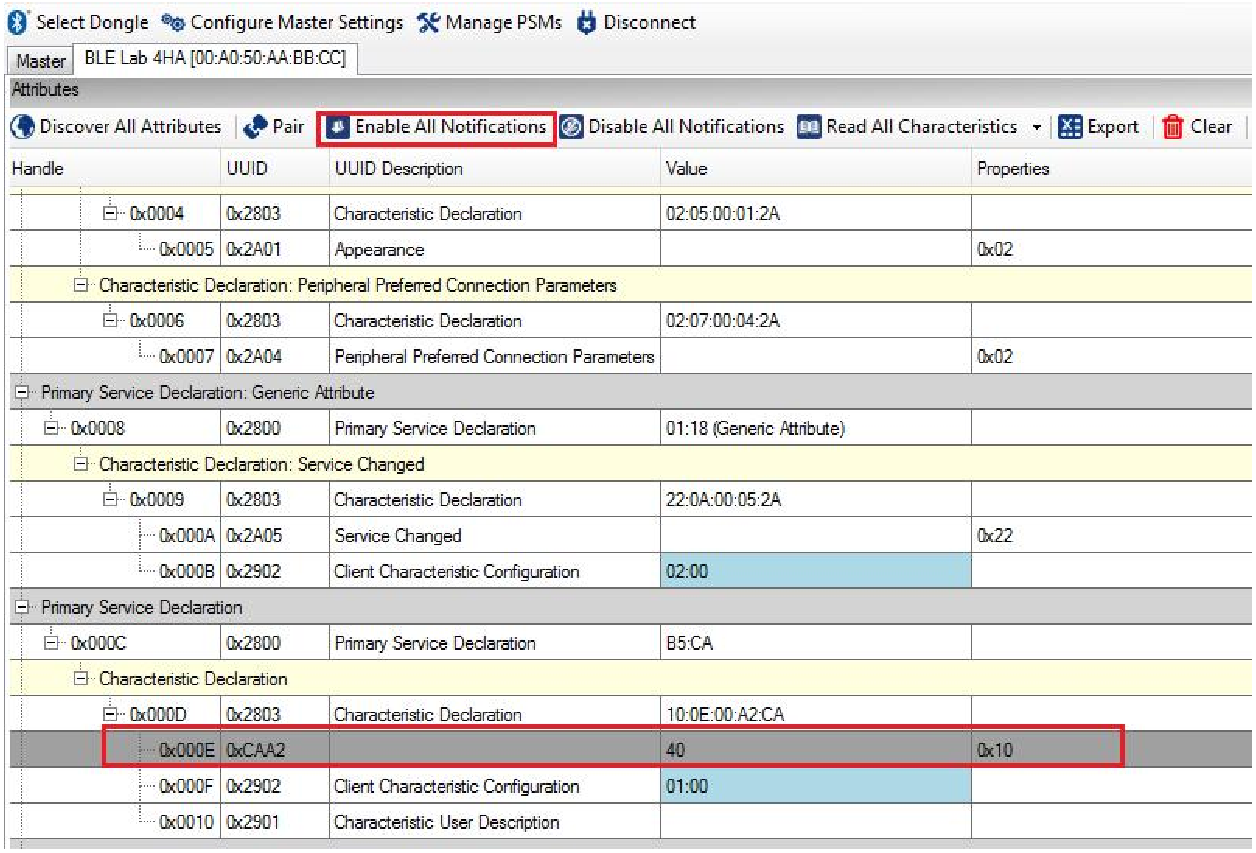
Value:

33:00:FF:EE

Figure 17: CySmart - Write Attribute

3.9 To test CapSense Slider, locate the CapSense Slider Characteristic (UUID = 0xCAA2) and enable notifications for it, either by clicking **Enable All Notifications** or by **writing 1** to its **CCCD descriptor (UUID=0x2902).**

Move finger over the slider on the kit and observe that the value of the Characteristic changes in the CySmart tool, while the tool's log shows notification packets being received. See Figure 19



Handle	UUID	UUID Description	Value	Properties
0x0004	0x2803	Characteristic Declaration	02:05:00:01:2A	
0x0005	0x2A01	Appearance		0x02
Characteristic Declaration: Peripheral Preferred Connection Parameters				
0x0006	0x2803	Characteristic Declaration	02:07:00:04:2A	
0x0007	0x2A04	Peripheral Preferred Connection Parameters		0x02
Primary Service Declaration: Generic Attribute				
0x0008	0x2800	Primary Service Declaration	01:18 (Generic Attribute)	
Characteristic Declaration: Service Changed				
0x0009	0x2803	Characteristic Declaration	22:0A:00:05:2A	
0x000A	0x2A05	Service Changed		0x22
0x000B	0x2902	Client Characteristic Configuration	02:00	
Primary Service Declaration				
0x000C	0x2800	Primary Service Declaration	B5:CA	
Characteristic Declaration				
0x000D	0x2803	Characteristic Declaration	10:0E:00:A2:CA	
0x000E	0xCAA2		40	0x10
0x000F	0x2902	Client Characteristic Configuration	01:00	
0x0010	0x2901	Characteristic User Description		

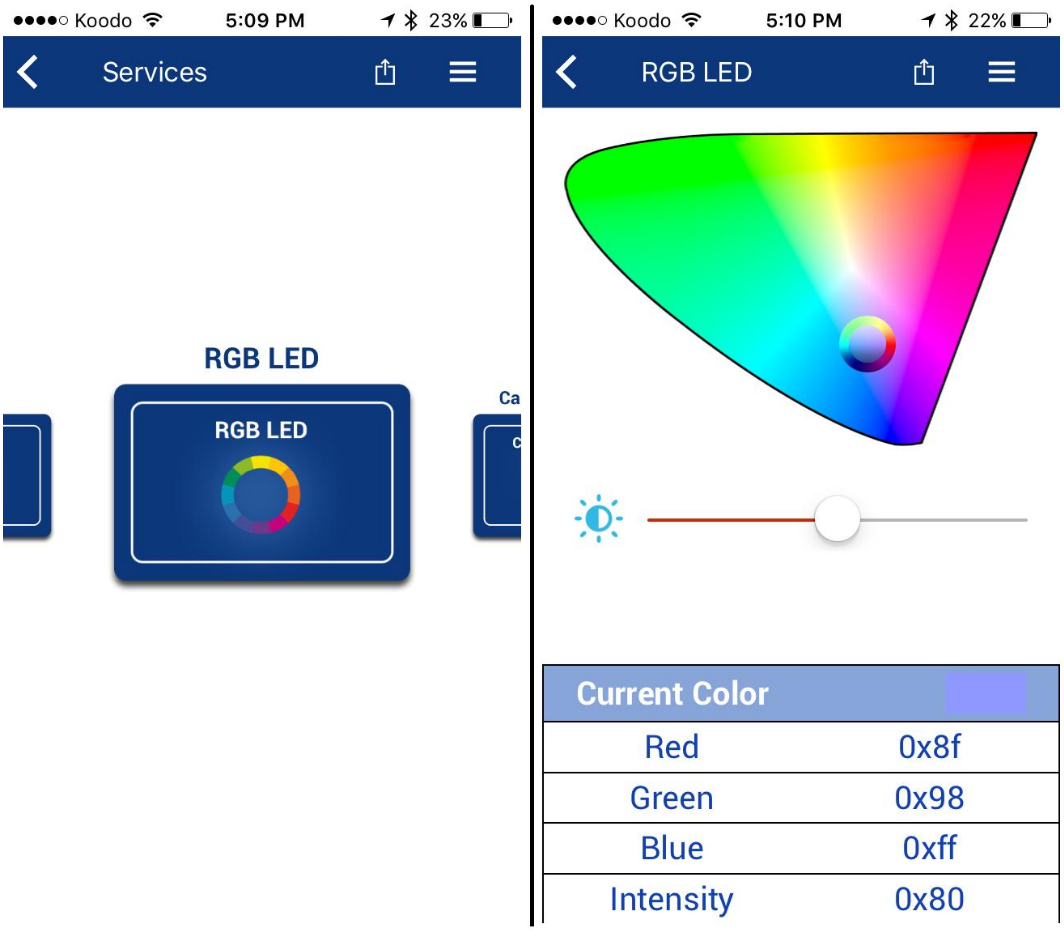
Figure 18: CySmart - Slider Notifications

.

3.10 Testing with CySmart Mobile App

Open the CySmart Mobile App on the apple phone. Then Connect to the GATT Server device on the app. Once connected, the app shows us all the Services exposed by the GATT Server. It automatically detects the Custom Services for RGB LED and CapSense

Select the RGB LED Service. We will see that a color gamut is available. Tap anywhere on the gamut to see the corresponding color on the RGB LED on the kit. See Figure 19.



Current Color	
Red	0x8f
Green	0x98
Blue	0xff
Intensity	0x80

Figure 19: CySmart iOS Mobile App – RGB LED Control

Then move the slider position on the app page to change the brightness level of the LED.

Once on the CapSense Slider page, move finger on the slider on the kit. We will see a corresponding slider update on the app page. See Figure 20.

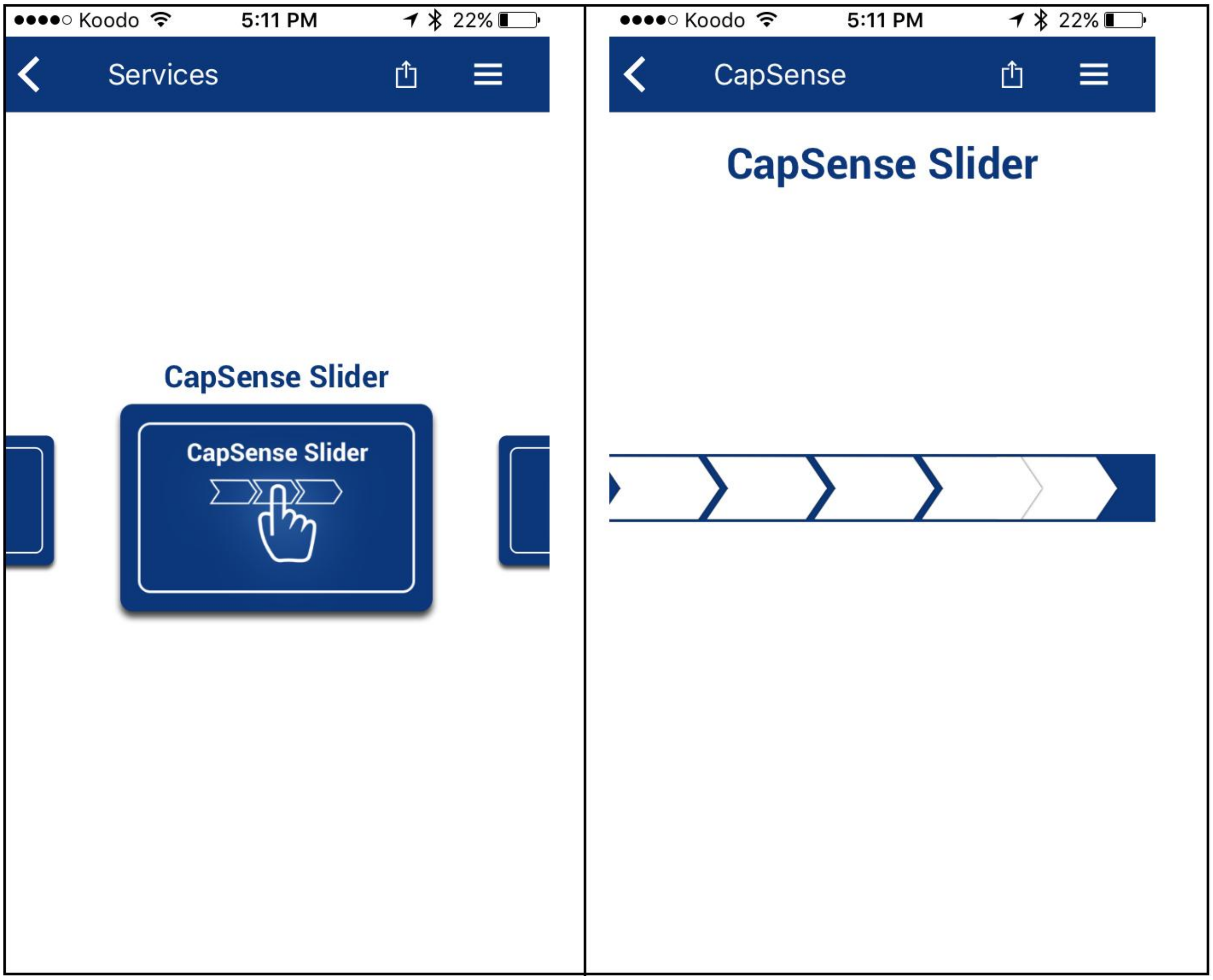


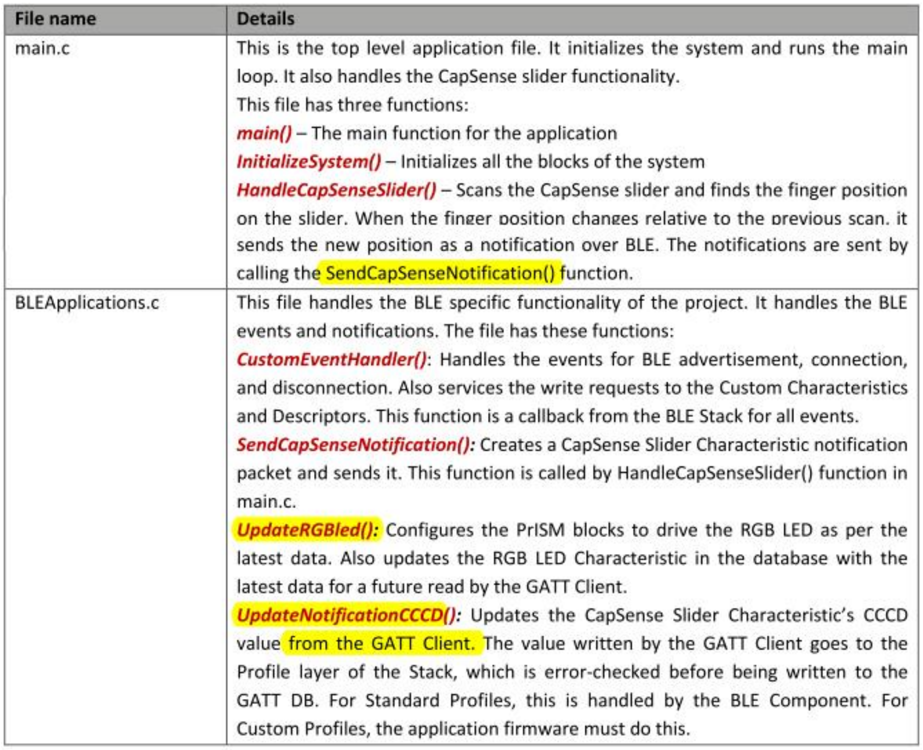
Figure 20: CySmart iOS Mobile App - CapSense Slider

Till Now, all the lab has been finished.

4. Conclusion:

Though this lab, we learn how to create Custom Profile by implementing RGB LEB function through BLE control. The Custom Service is user-defined. Each Service, Characteristic, and Descriptor should have their own UUID. Those information has been shown during the implementation. Also this lab teach us how to combine CapSense and BLE in this lab by designing a slider application.

After finished this lab, we found that we should take care about the thread and theory in the setting of component tool part and make a full preparing. Meanwhile, we should also pay more effort on understanding the firmware code part. The template of this lab provide us a good foundation to do further test. Base on this, we also analyzed the firmware flowchart and mechanism like the following picture.



File name	Details
main.c	This is the top level application file. It initializes the system and runs the main loop. It also handles the CapSense slider functionality. This file has three functions: *main()* – The main function for the application *InitializeSystem()* – Initializes all the blocks of the system *HandleCapSenseSlider()* – Scans the CapSense slider and finds the finger position on the slider. When the finger position changes relative to the previous scan, it sends the new position as a notification over BLE. The notifications are sent by calling the SendCapSenseNotification() function.
BLEApplications.c	This file handles the BLE specific functionality of the project. It handles the BLE events and notifications. The file has these functions: *CustomEventHandler()*: Handles the events for BLE advertisement, connection, and disconnection. Also services the write requests to the Custom Characteristics and Descriptors. This function is a callback from the BLE Stack for all events. *SendCapSenseNotification():* Creates a CapSense Slider Characteristic notification packet and sends it. This function is called by HandleCapSenseSlider() function in main.c. *UpdateRGBled():* Configures the PrISM blocks to drive the RGB LED as per the latest data. Also updates the RGB LED Characteristic in the database with the latest data for a future read by the GATT Client. *UpdateNotificationCCCD():* Updates the CapSense Slider Characteristic's CCCD value from the GATT Client. The value written by the GATT Client goes to the Profile layer of the Stack, which is error-checked before being written to the GATT DB. For Standard Profiles, this is handled by the BLE Component. For Custom Profiles, the application firmware must do this.

After the lab , we also test the blue tooth communication length for the default setting. It shown that the mobile phone could communicate with kit within 20m(a wall obstacle) or 50m (direct, no obstacle). The power of communication could be enhanced by setting of BLE. This scenario and parameter could provide us a reference on final project.