ELEC5514 Lab 3: BLE Service and Commands

After two weeks of practice, you have a good understanding of embedded programming and its development environment. Starting this week, we will really start to use the wireless communication protocols commonly used in IoT area.

This week's experiment will be divided into two parts:

1. Learn about the concept of servers in BLE communication and learn how to add services.
2. Add a function to your development board that sends commands from the phone to control the LED switches on the board.

Section 1

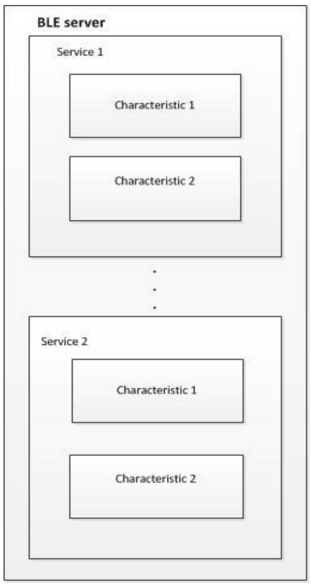
If you have completed Section 3 of Lab 2 last week, you can skip this Section 1 and start directly with Section 2.

Please use the code in "**Lab3_SampleApp**":

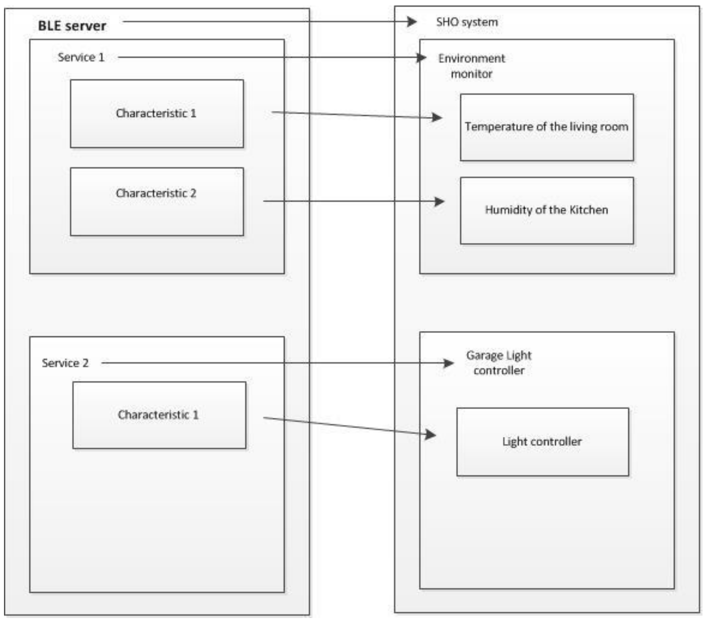
Source Code/ Lab3_SampleApp\Projects\Multi\Applications\SampleAppThT\MDK-ARM\STM32L053R8-Nucleo\SampleAppThT.uvprojx

In this section, you are beginning to touch the BLE world. Here we will use a specific example to explain how BLE works. To achieve the data exchange, a BLE device can be configured as either the client or the server.

The client plays a role of sending the data request to the server. The server plays a role of responding the data request from the client, e.g., data provider. Each server may have more than one services, each service can have at least one characteristic. The service is a predefined application. The characteristic can be regarded as the sub-function within a service. The exchanged data between server and client is called the attribute in BLE protocol. Based on the brief explanation above, a BLE service structure can be described as the graph below:



To make it easy to understand, we will use an example to explain.
Tom wants to set a SHO (Smart Home) system in his house. He prefers to add the environment monitor (service 1) and the light controller in garage (service 2) to the SHO system. For the environment monitoring, he wants the system to report the temperature state in the living room (characteristic 1 in service 1) and the humidity of the kitchen (characteristic 2 in service 1). After Tom left his house in the morning, he will get the corresponding temperature (attribute from characteristic 1) and the humidity (attribute from characteristic 2) every 5 hours. When Tom comes back home, he wants to use his phone to switch on (characteristic 1 in service 2) or off (characteristic 2 in service 2) the light in his garage. When Tom sends the number "1" (attribute of characteristic 1, service 2), the light will be turned on. In contrast, "2" (attribute of characteristic 2, service 2) indicates the light turned off.



The introduction for the "service", "characteristic" and "attribute" in detail can be found from "*BLE_stacks programming guidelines.pdf*" in the folder of Help Manual.
To avoid the connection confliction among groups, we will allocate each group a specific address. Please follow the form below (We strongly recommend that each group uses the corresponding local name, device name and mac address in the following labs):

Group N0.	Local name	Device name	MAC address
1	BLE_LOC_G1	Dev1	0xF1, 0x33, 0x01, 0xE1, 0x80, 0x03
2	BLE_LOC_G2	Dev2	0xF2, 0x33, 0x01, 0xE1, 0x80, 0x03
3	BLE_LOC_G3	Dev3	0xF3, 0x33, 0x01, 0xE1, 0x80, 0x03
4	BLE_LOC_G4	Dev4	0xF4, 0x33, 0x01, 0xE1, 0x80, 0x03
5	BLE_LOC_G5	Dev5	0xF5, 0x33, 0x01, 0xE1, 0x80, 0x03
6	BLE_LOC_G6	Dev6	0xF6, 0x33, 0x01, 0xE1, 0x80, 0x03
7	BLE_LOC_G7	Dev7	0xF7, 0x33, 0x01, 0xE1, 0x80, 0x03
8	BLE_LOC_G8	Dev8	0xF8, 0x33, 0x01, 0xE1, 0x80, 0x03

9	BLE_LOC_G9	Dev9	0xF9, 0x33, 0x01, 0xE1, 0x80, 0x03
10	BLE_LOC_G10	Dev10	0x10, 0x33, 0x01, 0xE1, 0x80, 0x03
11	BLE_LOC_G11	Dev11	0x11, 0x33, 0x01, 0xE1, 0x80, 0x03
12	BLE_LOC_G12	Dev12	0x12, 0x33, 0x01, 0xE1, 0x80, 0x03
13	BLE_LOC_G13	Dev13	0x13, 0x33, 0x01, 0xE1, 0x80, 0x03

After getting familiar with the principle of service, characteristic and attribute, in this section, you are required to add：

1) A blank service, the service name can be determined by yourself. But in this lab, please use the following UUID:

0x1b,0xc5,0xd5,0xa5,0x02,0x00,0xb4,0x9a,0xe1,0x11,0x3a,0xcf,0x80,0x6e,0x36,0x5d

2) The service must include one blank characteristic, the characteristic name can be determined by yourself but please use the UUID

TX: *0x1b,0xc5,0xd5,0xa5,0x02,0x00,0xb4,0x9a,0xe1,0x11,0x3a,0xcf,0x80,0x6e,0x36,0x5e*

RX: *0x1b,0xc5,0xd5,0xa5,0x02,0x00,0xb4,0x9a,0xe1,0x11,0x3a,0xcf,0x80,0x6e,0x36,0x5f*

Hints: In the files “sample_service.c” and “sample_service.h”, the source code has defined an example service and the corresponding characteristics, like “sampleServHandle” and “TXCharHandle”, you can use the “Ctrl” + “F” to find the corresponding operations, like how to define the service and characteristic. Please note that after you finish defining your own service in “sample_service.c”, you need to relate this “adding” action to the “sample_service.h”, and then, in the “main.c” file, you need to execute it so that the application is added online.

1. In the “**sample_service.c**” file, you need to customize your own handle or variables, and then define your own service and add your own service. In order to figure out some functions, you can select them and click the right button and click “Go to definition of xxx” to find the definition of this code.
2. In the “**sample_service.h**” file, you also need to register the function you add.
3. In the “**main.c**” file, you need to execute the function to add your service.

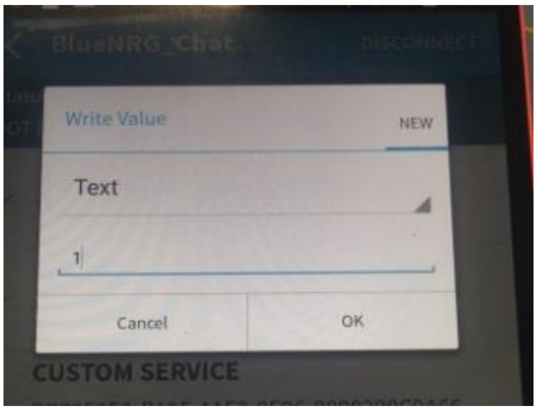
When finish this part, call your tutor to check your result. You can use Arduino to record the serial feedback from the board. Remember to take the screenshot, you need to use them in your lab report.

Section 2

In this section, you need to increase the service you added in last section to realize the following function. The required contents are presented as follows:

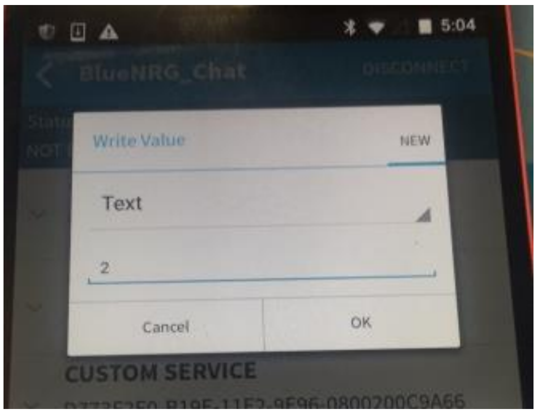
Build the connection from the board and the apps in the smart phone, and the LED2 will be turned on if mobile phone sends “1”, and turned off if “2” is sent. And if you input neither “1” nor “2”, the string will be shown on the screen through the serial monitor of “Arduino”. For example:

Input symbol “1” from the mobile phone:

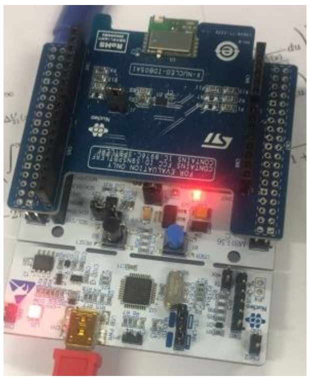


Then the green LED light on the board is turned on:

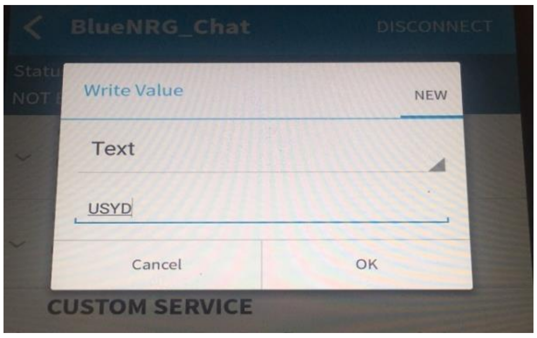
Input symbol “2” to the borad:



Then the light is turned off:

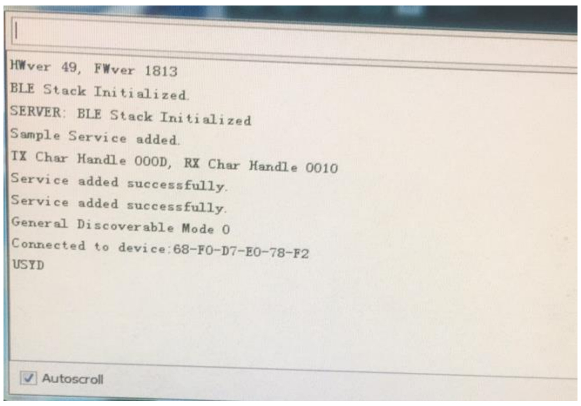


Input the symbol neither “1” nor “2”, for example “USYD”:



USYD

Then the input message is showed on the desktop:



```
HWver 49, FWver 1813
BLE Stack Initialized.
SERVER: BLE Stack Initialized
Sample Service added.
TX Char Handle 000D, RX Char Handle 0010
Service added successfully.
Service added successfully.
General Discoverable Mode 0
Connected to device:68-F0-D7-E0-78-F2
USYD
```

To complete this task, you need to modify the blank service, which was added in the last section. In details, you need to do the following steps:

In the **sample_service.c** file, the codes have shown an example that how to add a sample service and two characteristics through **Add_Sample_Service**. You can read the code and find the meaning of each line. Please pay attention to the code below:

```
ret = aci_gatt_add_serv(UUID_TYPE_128, service_uuid, PRIMARY_SERVICE, 7,
&sampleServHandle);
```

Please try to understand each part included in this function.

```
ret = aci_gatt_add_char(sampleServHandle, UUID_TYPE_128, charUuidRX, 20,
CHAR_PROP_WRITE|CHAR_PROP_WRITE_WITHOUT_RESP,
ATTR_PERMISSION_NONE, GATT_NOTIFY_ATTRIBUTE_WRITE, 16, 1,
&RXCharHandle);
```

Please try to understand each part included in this function.

If you don't understand the meaning of any code, you can select them and right click to find the definition of the function. Please make sure you have understood it before using it.

In the file "**sample_service.c**", you can find the following code:

```
void Attribute_Modified_CB(uint16_t handle, uint8_t data_length, uint8_t *att_data)
```

This is the **call-back function** whose role is: once the function is triggered, this function will be called automatically. Please read it and figure out how it works, then have a try to edit it so as to achieve the application. This function provides the developers with the solution of the communication between two BLE devices.

In the code below, we can find the function of the sample service, such as the "**RXCharHandle**"; so, in this way, you can modify your own function of the adding service. For example: when receive "1", turn the light on.

Tips: the command to control LED on/off is as described below:

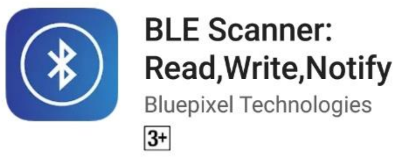
(The default led is Led2, please do not edit it)

BSP_LED_ON(Led2) ---Turn on the light

BSP_LED_OFF(Led2) ---Turn off the light

Now the codes written in the **sample_service.c** is done, you then need to complete the service registering in **main.c** and **sample_service.h**. The tips are to find the key codes "**Add_sample_service**" and write your own service by changing some of words.

You use the following App to test your code by yourself.



BLE Scanner:
Read,Write,Notify
Bluepixel Technologies
3+

After you finished the lab, please rise your hand and call the tutor to check the result. Remember to save the screenshot and the photo of your result then you can use them in your report.