Bluetooth 5 Lab Manual

This practical exercise will demonstrate how to use the Bluetooth 5 features supported in our BLE SDK 2.10 using the GCC compiler and SimplicityStudio IDE.

The focus of this training session will be on BT5 features supported in BLE SDK 2.10

- LE Coded and 2M PHY
- Multiple Advertisement sets
- Extended advertisements
- Anonymous advertising

Topics Covered

- Multiple advertisement sets
- Extended advertising
- 2 M and LE Coded PHY
- Scan request reporting
- Anonymous advertising

Getting Started

Review the following material before starting the Bluetooth 5 labs. Ensure that you have the correct hardware and software to successfully complete the labs.

0.1 Hardware Requirements

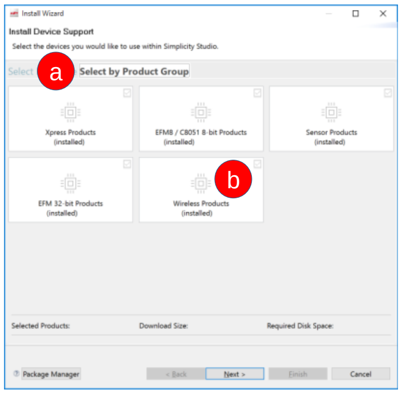
- 2 Wireless Starter Kits (EFR32BG13/MG13 Radio Boards installed)
- 2 USB mini B cables
- iOS or Android Mobile device

0.2 Software Requirements

- **Simplicity Studio v4**
- **Gecko Platform – 2.4**
- **Bluetooth SDK – 2.10**
- **Blue Gecko v1.5.0 Mobile App**
- **Project source files (main-client.c, main-server.c)**

0.3 Software Installation

1. Install **Simplicity Studio v4** (www.silabs.com/simplicity-studio)
    a. Install Device Support by clicking on tab "**Select by Product Group**".
    b. Click on **Wireless Products**. Click **Next**.



Install Device Support

Select the devices you would like to use within Simplicity Studio.

Install Wizard
Select
a
Select by Product Group
Xpress Products (installed)
EFM8 / C8051 8-bit Products (installed)
Sensor Products (installed)
EFM 32-bit Products (installed)
b
Wireless Products (installed)
Selected Products:
Download Size:
Required Disk Space:
Package Manager
< Back
Next >
Finish
Cancel

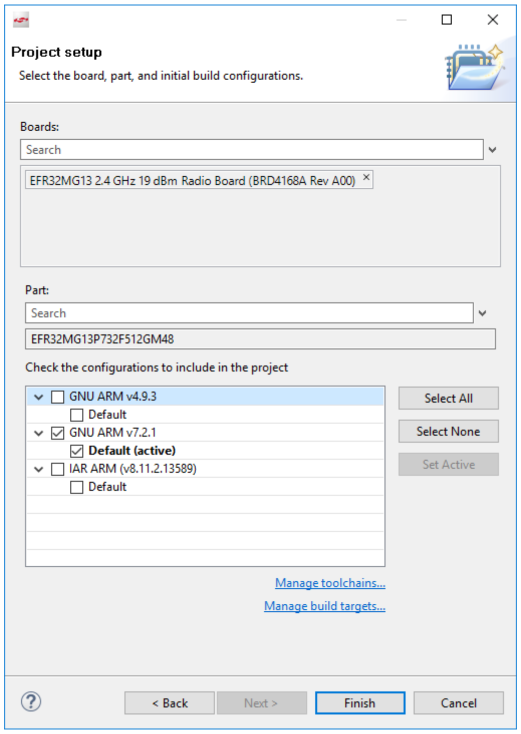
2. Install Protocol SDKs by clicking menu bar **Help** -> **Update Software**.
   Click on tab for "SDKs" in Package Manager window
    a. Select and Install Gecko Platform – 2.4
    b. Select and Install Bluetooth SDK – 2.10.0
3. Install Silicon Labs Blue Gecko WSTK App at a mobile app store: Google Play or Apple App Store

1 ADD 2M PHY CONNECTIONS TO AN APPLICATION

1.1 Create a Project using Simplicity Studio IDE

Start by creating an soc-empty sample application.

1. Click on the **Simplicity IDE** button in the upper right corner to return to the IDE perspective.
2. Select **File** ->**New -> Project** in the menu bar.
3. In the Create Project window, Click on Silicon Labs AppBuilder project and click next.
4. Click Bluetooth SDK and click next
5. Click Bluetooth SDK 2.10.0.0 and click next
6. Click SOC-Empty and click next
7. Enter soc-bt5-server for the project name and click next
8. Select your target board from the list and turn off all configurations except GNU ARM v7.2.1 and click finish



Project setup

Select the board, part, and initial build configurations.

Boards:

Search

EFR32MG13 2.4 GHz 19 dBm Radio Board (BRD4168A Rev A00)

Part:

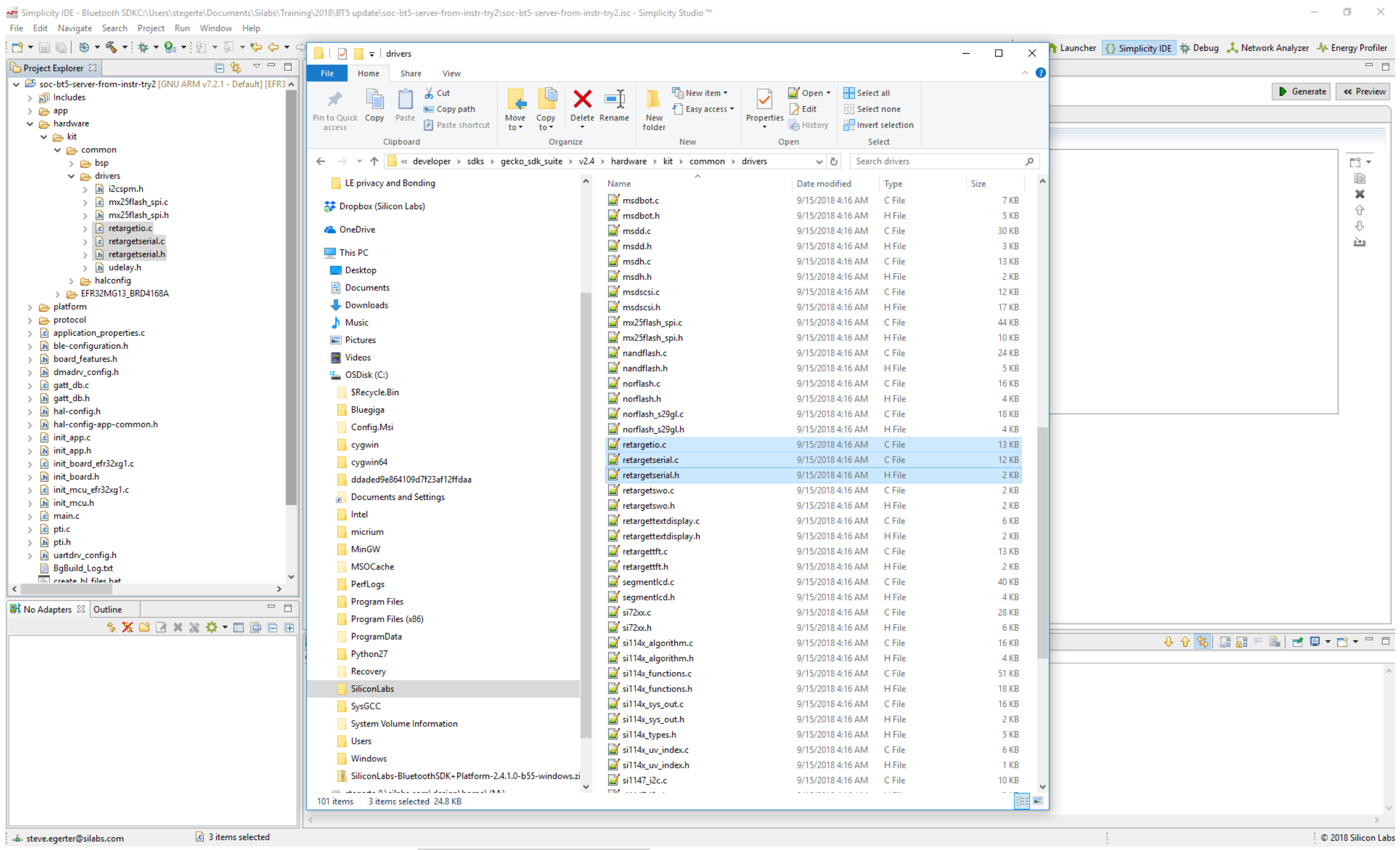
Search

EFR32MG13P732F512GM48

Check the configurations to include in the project

- ☐ GNU ARM v4.9.3
  - ☐ Default
- ☑ GNU ARM v7.2.1
  - ☑ **Default (active)**
- ☐ IAR ARM (v8.11.2.13589)
  - ☐ Default

9. Drag and drop main-server.c into the project folder and delete main.c
10. Drag and drop retargetio.c, retargetserial.c and retargetserial.h into the project's hardware\kit\common\drivers folder



Simplicity IDE - Bluetooth SDK:C:\Users\stegerte\Documents\Silabs\Training\2018\BT5 update\soc-bt5-server-from-instr-try2\soc-bt5-server-from-instr-try2.isc - Simplicity Studio ™

« developer › sdks › gecko_sdk_suite › v2.4 › hardware › kit › common › drivers

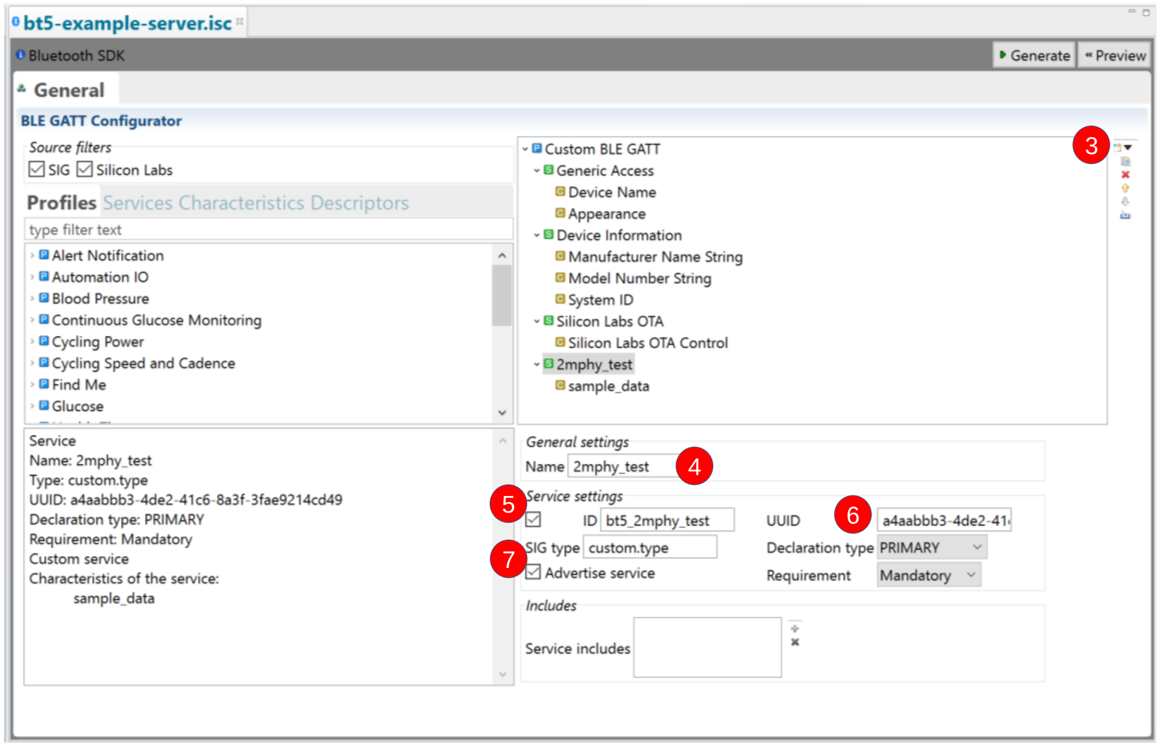
Name	Date modified	Type	Size
msdbot.c	9/15/2018 4:16 AM	C File	7 KB
msdbot.h	9/15/2018 4:16 AM	H File	5 KB
msdd.c	9/15/2018 4:16 AM	C File	30 KB
msdd.h	9/15/2018 4:16 AM	H File	3 KB
msdh.c	9/15/2018 4:16 AM	C File	13 KB
msdh.h	9/15/2018 4:16 AM	H File	2 KB
msdscsi.c	9/15/2018 4:16 AM	C File	12 KB
msdscsi.h	9/15/2018 4:16 AM	H File	17 KB
mx25flash_spi.c	9/15/2018 4:16 AM	C File	44 KB
mx25flash_spi.h	9/15/2018 4:16 AM	H File	10 KB
nandflash.c	9/15/2018 4:16 AM	C File	24 KB
nandflash.h	9/15/2018 4:16 AM	H File	5 KB
norflash.c	9/15/2018 4:16 AM	C File	16 KB
norflash.h	9/15/2018 4:16 AM	H File	4 KB
norflash_s29gl.c	9/15/2018 4:16 AM	C File	18 KB
norflash_s29gl.h	9/15/2018 4:16 AM	H File	4 KB
retargetio.c	9/15/2018 4:16 AM	C File	13 KB
retargetserial.c	9/15/2018 4:16 AM	C File	12 KB
retargetserial.h	9/15/2018 4:16 AM	H File	2 KB
retargetswo.c	9/15/2018 4:16 AM	C File	2 KB
retargetswo.h	9/15/2018 4:16 AM	H File	2 KB
retargettextdisplay.c	9/15/2018 4:16 AM	C File	6 KB
retargettextdisplay.h	9/15/2018 4:16 AM	H File	2 KB
retargettft.c	9/15/2018 4:16 AM	C File	13 KB
retargettft.h	9/15/2018 4:16 AM	H File	2 KB
segmentlcd.c	9/15/2018 4:16 AM	C File	40 KB
segmentlcd.h	9/15/2018 4:16 AM	H File	4 KB
si72xx.c	9/15/2018 4:16 AM	C File	28 KB
si72xx.h	9/15/2018 4:16 AM	H File	6 KB
si114x_algorithm.c	9/15/2018 4:16 AM	C File	16 KB
si114x_algorithm.h	9/15/2018 4:16 AM	H File	4 KB
si114x_functions.c	9/15/2018 4:16 AM	C File	51 KB
si114x_functions.h	9/15/2018 4:16 AM	H File	18 KB
si114x_sys_out.c	9/15/2018 4:16 AM	C File	16 KB
si114x_sys_out.h	9/15/2018 4:16 AM	H File	2 KB
si114x_types.h	9/15/2018 4:16 AM	H File	5 KB
si114x_uv_index.c	9/15/2018 4:16 AM	C File	6 KB
si114x_uv_index.h	9/15/2018 4:16 AM	H File	1 KB
si1147_i2c.c	9/15/2018 4:16 AM	C File	10 KB

101 items   3 items selected 24.8 KB

11. Open hal-config.h and set `HAL_VCOM_ENABLE to 1`

1.2 Add 2M PHY Service to GATT

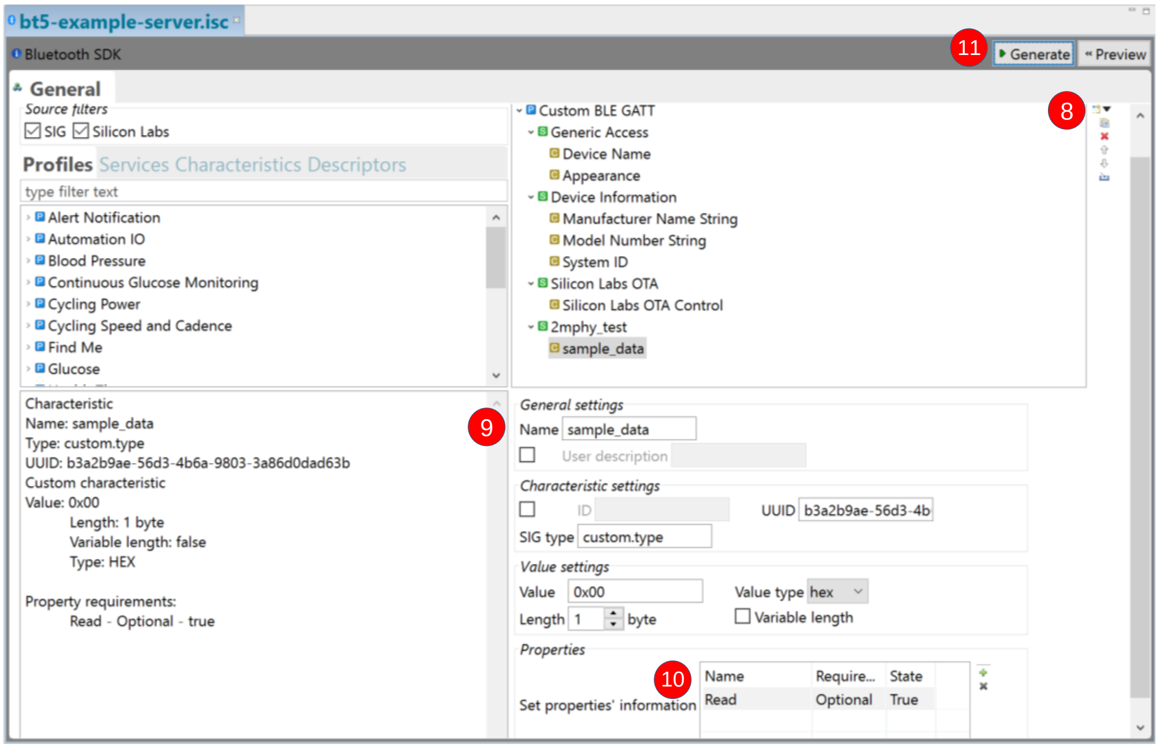
1. Open the visual GATT editor by double clicking project's **.isc** file.
2. Create a new service by clicking on the **Create new item** icon in upper right of configurator. Select **New Service**.
3. Under ***General settings***, set the Name of the service to "**2mphy_test**".
4. Give it the following **UUID**:
   a4aabbb3-4de2-41c6-8a3f-3fae9214cd49
   This is an important step because the client will look for a server advertising a service with this custom UUID.
5. Check the **Advertise service** box.



Service
Name: 2mphy_test
Type: custom.type
UUID: a4aabbb3-4de2-41c6-8a3f-3fae9214cd49
Declaration type: PRIMARY
Requirement: Mandatory
Custom service
Characteristics of the service:
sample_data

Name 2mphy_test
ID bt5_2mphy_test
UUID a4aabbb3-4de2-41
SIG type custom.type
Declaration type PRIMARY
Advertise service
Requirement Mandatory

6. Create a characteristic for this service by clicking on the **Create new item** icon and selecting **New Characteristic**.
7. Under ***General settings***, set the Name of the characteristic to "**sample_data**".
8. Under ***Properties***, set the information to only include **Read** with Requirement = **Optional** and State = **True**.
9. Click **Generate** to update changes to the GATT files.



Characteristic
Name: sample_data
Type: custom.type
UUID: b3a2b9ae-56d3-4b6a-9803-3a86d0dad63b
Custom characteristic
Value: 0x00
Length: 1 byte
Variable length: false
Type: HEX

Property requirements:
Read - Optional - true

Name sample_data
ID
UUID b3a2b9ae-56d3-4b
SIG type custom.type
Value 0x00
Value type hex
Length 1 byte
Variable length
Set properties' information

Name	Require...	State
Read	Optional	True

1.3 Add event handler code for 2M PHY connections

1. Add an event handler in main-server.c for the "le_connection_phy_status" event.

Insert the following code at the location marked by comment: /* LAB SECTION 1.3 STEP 1 INSERT CODE BELOW */

```
case gecko_evt_le_connection_phy_status_id:
        printf("now using the %s PHY\r\n", getPhyName(evt-
>data.evt_le_connection_phy_status.phy).string);
        break;
```

1.4 Add event handler for scan request reporting

Another feature of Bluetooth 5 is to add the ability to report scan requests. To add this event handler paste in the following code after the comment which reads /*LAB Section 1.4*/

```
case gecko_evt_le_gap_scan_request_id:
        printf("received a scan request from %X:%X:%X:%X:%X:%X:\r\n",
                    evt->data.evt_le_gap_scan_request.address.addr[5],
                          evt->data.evt_le_gap_scan_request.address.addr[4],
                          evt->data.evt_le_gap_scan_request.address.addr[3],
                          evt->data.evt_le_gap_scan_request.address.addr[2],
                          evt->data.evt_le_gap_scan_request.address.addr[1],
                          evt->data.evt_le_gap_scan_request.address.addr[0]);
        break;
```

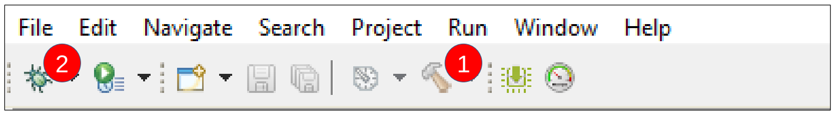
. This is a purely informative event and no action is required here. This event handler simply prints out the address of the device requesting the scan response. To enable this reporting you must also add a call to the following :

```
gecko_cmd_le_gap_set_advertise_report_scan_request(0,1);
```

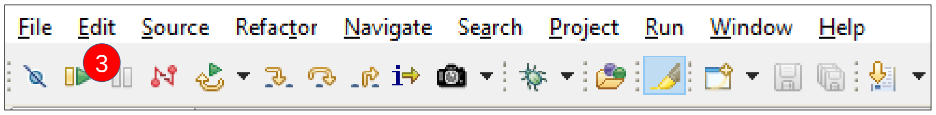
add this call after the comment /* LAB Section 1.4 enable scan request*/

1.5 Program Starter Kit with 2M PHY Server Application

1. Click **Build** icon in the tool bar at the top of the Simplicity IDE.
   The project will compile based on its build configuration. (NOTE: You need to select the project in the Project explorer view on left)
2. Press the **Debug** icon in the tool bar.
   This will flash the project onto the board if it was successfully built. This only updates the program memory.



3. Once the project is flashed, the Debug perspective will be shown.
   Press the **Resume** icon in the tool bar to have the application start.



1.6 Program client with 2M Capable Client Application

Now that we have a Server(aka peripheral) application, we need to build a client which can connect to this server.

As with the server application, the source code for the client is provided (main-client.c).

1. Create a new project as you did in section 1.1 and name it bt5-client
2. Replace the sample app's main.c with main-client.c. A few helper functions have been added to this template, they are not related to Bluetooth 5 features.
3. Drag and drop retargetio.c, retargetserial.c and retargetserial.h into the project's hardware\kit\common\drivers folder
4. Open hal-config.h and set `HAL_VCOM_ENABLE to 1`
5. `Make a copy of platform\emdrv\gpiointerrupt\src\ (including the source file it contains)` in your SDK and copy it to the `platform\emdrv\gpiointerrupt\` folder in your project.
6. System Boot event handler. We'll start by adding a startup message, so that we can tell when the device has started, and the setup for scanning. Paste in the following code for the system boot event, right after the comment /*Section 1.5 step 6*/ in main-client.c

```
printf("BT5 test client started\r\n");
/*set the discovery type for both PHYs to passive */
gecko_cmd_le_gap_set_discovery_type(le_gap_phy_1m|le_gap_phy_coded,0);
/* 200 ms scan window min/max*/
gecko_cmd_le_gap_set_discovery_timing(le_gap_phy_1m|le_gap_phy_coded,320,320);
/* start listening for devices to connect to */
/* need some way to set the PHY*/
gecko_cmd_le_gap_start_discovery(scanning_phy,le_gap_discover_generic);
gecko_cmd_le_gap_set_discovery_extended_scan_response(true);
```

7. The client will look for server by the service it advertises. Add the following code just after the comment /*Section 1.5 step 7*/

```
static const uint8 silabs_2mphy_uuid[16] =
    {0xa4,0xaa,0xbb,0xb3,0x4d,0xe2,0x41,0xc6,0x8a,0x3f,0x3f,0xae,0x92,0x14,0xcd,0x49};
```

8. Now we'll had a scan response event handler. The event used here is a new event available starting in SDK v2.10. This event is enabled by calling
   gecko_cmd_le_gap_set_discovery_extended_scan_response(true);
   as done above in step 6. Add the following code just after the comment /*Section 1.5 step 8*/

```
      case gecko_evt_le_gap_extended_scan_response_id:
              {
                                      for(int index = 0; index < evt-
>data.evt_le_gap_extended_scan_response.data.len;){
                      if(evt->data.evt_le_gap_extended_scan_response.data.data[in-
dex+1] == 0x07){
                        if(check_uuid(&evt->data.evt_le_gap_extended_scan_re-
sponse.data.data[index+2], sizeof(scan_uuid)) == true){

                              /*stop scanning for now*/
                              gecko_cmd_le_gap_end_procedure();
                              /* and connect to the advertising device using what-
ever PHY we're scanning on*/
                              gecko_cmd_le_gap_connect(evt->data.evt_le_gap_ex-
tended_scan_response.address, evt->data.evt_le_gap_extended_scan_response.ad-
dress_type,scanning_phy);
                              break;
                        }
                      }
                        index += evt->data.evt_le_gap_extended_scan_re-
sponse.data.data[index]+1;
                }

              }
          break;
```

9. We have two buttons on the WSTK and we'll use these to control the following parameters
   PB0 to control the PHY used for scanning
   PB1 to control the desired PHY once a connection has been established.
   To accomplish this will use the Bluetooth stack's external sign mechanism which requires an event handler. Paste in the following immediately after the comment /* LAB Section 1.5 step 9*/

```
case gecko_evt_system_external_signal_id:
        /*
         * PB0 is used to toggle the PHY used for scanning
         * PB1 is used to change the PHY used once a connection is established
         *
         * */
      if(1 == evt->data.evt_system_external_signal.extsignals){
        printf("Switch scanning PHY\r\n");
        gecko_cmd_le_gap_end_procedure();
        if(scanning_phy == le_gap_phy_1m){
           scanning_phy = le_gap_phy_coded;
           phy_index = 2;
        }
        else if(scanning_phy == le_gap_phy_coded){
           scanning_phy = le_gap_phy_1m;
        }
        printf("start scanning on %s PHY\r\n",getPhyName(scanning_phy).string);
        gecko_cmd_le_gap_start_discovery(scanning_phy, le_gap_discover_ge-
neric);
       }
       else if(2 == evt->data.evt_system_external_signal.extsignals){

          //use PB1 to increment the desired PHY
          phy_index = (phy_index+1)%4;
          desired_phy = supported_phys[phy_index];
          gecko_cmd_le_connection_set_phy(connection, desired_phy);
       }

       break;
```

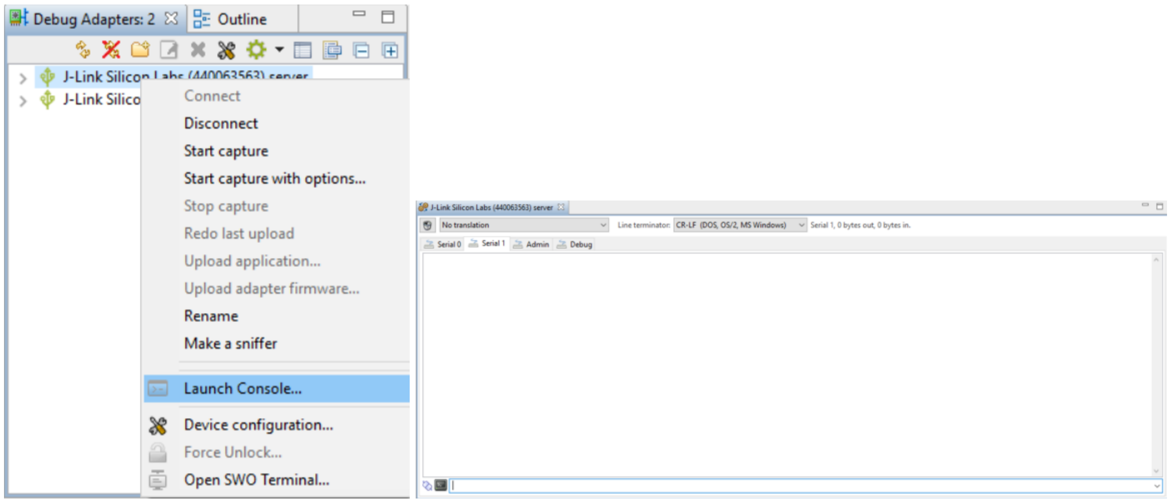
10. Build the project and flash the resulting hex file to the second WSTK.

1.7 Open Consoles to display 2M Status

The two Bluetooth 5 devices will establish a 1M PHY connection. Once a connection is established the PHY can be changed by pressing PB1 on the client WSTK. If the selected PHY is not supported by the remote server, the connection will fall back to the 1M PHY.

The status of the connections is printed to their debug terminals.

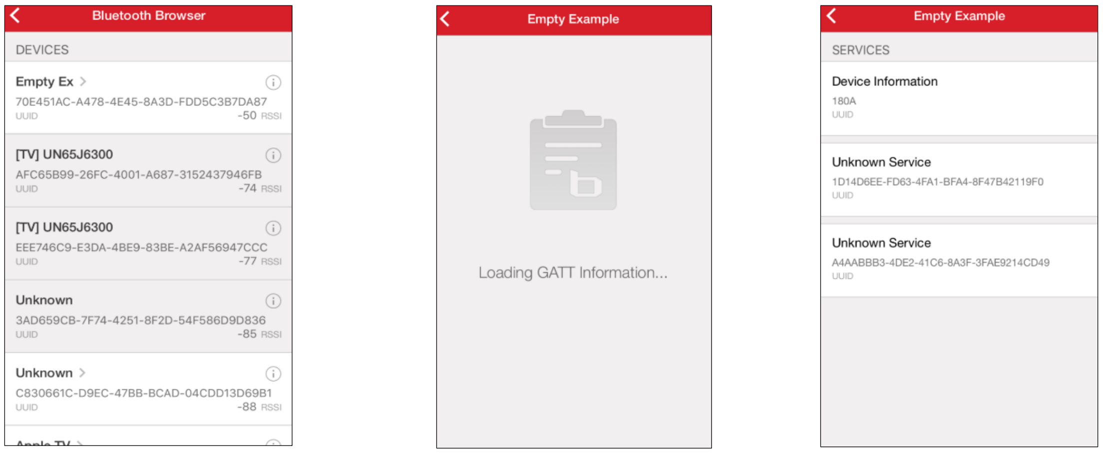
1. Open up 2 consoles to show these status messages. Right click the WSTK in the debug adapters perspective as shown, then select the serial1 tab, place the cursor in the text entry field at the bottom and hit enter



1. Press reset on both boards to see the startup messages and confirm that both applications are running.
2. Once a connection is formed, press PB1 on the Client board to switch to the 2 Mbps PHY.
3. The client will now display a message indicating that the 2Mbps PHY is in use.

1.8 Compare Server PHY Connection of Mobile Device vs. 2M PHY Client

1. To establish a 4.2 connection with server, switch off the client WSTK then restart the server and connect to it using the Blue gecko app in the mobile. Click on the **Bluetooth Browser** and select the “Empty Ex” server. A 1M PHY connection is established as it loads the GATT information.


Bluetooth Browser
DEVICES
Empty Ex
70E451AC-A478-4E45-8A3D-FDD5C3B7DA87
UUID -50 RSSI
[TV] UN65J6300
AFC65B99-26FC-4001-A687-3152437946FB
UUID -74 RSSI
[TV] UN65J6300
EEE746C9-E3DA-4BE9-83BE-A2AF56947CCC
UUID -77 RSSI
Unknown
3AD659CB-7F74-4251-8F2D-54F586D9D836
UUID -85 RSSI
Unknown
C830661C-D9EC-47BB-BCAD-04CDD13D69B1
UUID -88 RSSI
Empty Example
Loading GATT Information...
Empty Example
SERVICES
Device Information
180A
UUID
Unknown Service
1D14D6EE-FD63-4FA1-BFA4-8F47B42119F0
UUID
Unknown Service
A4AABBB3-4DE2-41C6-8A3F-3FAE9214CD49
UUID

2 ADD MULTIPLE ADVERTISEMENT SETS TO AN APPLICATION

Bluetooth 5 makes it possible to have multiple advertisements each with its own data, transmit power and timing parameters. In this part we will examine how an application can simultaneously send out a non-connectable iBeacon advertisement as well as a connectable Eddystone advertisement.

The project loaded in Section 1, **bt5-server**, will be modified to demonstrate the new features for Advertisement sets.

2.1 Increase Number of Advertisers

1. Open main-server.c and locate the definition of gecko_configuration.
   Make the change below at the location marked by comment: /* LAB SECTION 2.1 STEP 1 MODIFY MAX ADVERTISERS */
   This will initialize the Bluetooth stack to support the specific amount of advertisements.

```
/* Gecko configuration parameters (see gecko_configuration.h) */
static const gecko_configuration_t config = {
  .config_flags=0,
  .sleep.flags=SLEEP_FLAGS_DEEP_SLEEP_ENABLE,
  .bluetooth.max_connections=MAX_CONNECTIONS,
  .bluetooth.heap=bluetooth_stack_heap,
  .bluetooth.max_advertisers = 3,
  .bluetooth.heap_size=sizeof(bluetooth_stack_heap),
  .bluetooth.sleep_clock_accuracy = 100, // ppm
  .gattdb=&bg_gattdb_data,
  .ota.flags=0,
  .ota.device_name_len=3,
  .ota.device_name_ptr="OTA",
  #ifdef FEATURE_PTI_SUPPORT
  .pti = &ptiInit,
  #endif
};
```

2.2 Configure Beacons and Enable after Boot Event

1. In **main-server.c** go to the location marked by comment: /* first advertising set – iBeacon . Do this first because it sets the power */ and uncomment the following line
   bcnSetupAdvBeaconing();
   If you examine the contents of this function you'll see there is a new API gecko_cmd_le_gap_set_advertise_tx_power(). This API allows you to set the tx power used in just this advertiser without affecting other advertisers nor the tx power used in connections.

2. In **main-server.c** go to the location marked by comment: /* LAB SECTION 2.2 STEP 2 AND 3 MODIFY BELOW */
3. Modify the characters for a custom URL you would like for the physical web advertisement.
4. Determine the new value with the new URL and modify the length and the characters for the URL
   NOTE: The default overhead and silabs.com requires the length of service data to be 16 bytes (length = 0x10)

```
static uint8_t eddystone_data[EDDYSTONE_DATA_LEN] = {

  0x03,           //Length of service list
  0x03,           //service list
  0xAA, 0xFE,     //Eddystone ID
  0x10,           //length of service data
  0x16,           //service data
  0xAA, 0xFE,    //Eddystone ID
  0x10,           //frame type Eddystone-URL
  0x00,          // tx power
  0x00,          //http://www.
  's','i','l','a','b','s','.','c','o','m'
};
```

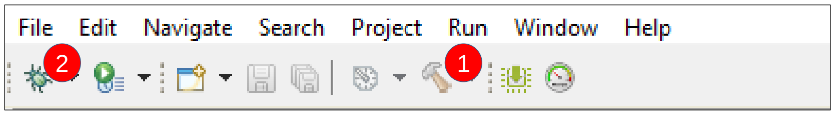
5. Navigate to the Event Handler section of code in the **main()** routine.
6. Modify the case statement for **gecko_evt_system_boot_id**, after the existing iBeacon instructions.

Insert the following code at the location marked by comment: /* LAB SECTION 2.2 STEP 5 INSERT CODE BELOW */

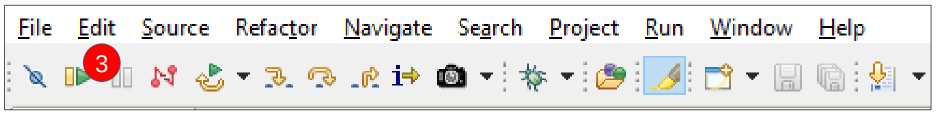
```
/* EddyStone*/
  gecko_cmd_le_gap_bt5_set_adv_data( 2, ADV_PKT, 30, eddystone_data);
  gecko_cmd_le_gap_set_advertise_configuration(2,0x01);
  gecko_cmd_le_gap_set_advertise_timing(2,160,160,0,0);
  gecko_cmd_le_gap_set_advertise_channel_map(2,0x07);
  gecko_cmd_le_gap_start_advertising(2,le_gap_user_data, le_gap_non_connectable);
```

2.3 Build and Flash the Application

1. Click the **Build** icon in the tool bar at the top of the Simplicity IDE.
   The project will compile based on its build configuration. (NOTE: You need to select the project in the Project explorer view on left)
2. Press the **Debug** icon in the tool bar.
   This will flash the project onto the board if it was successfully built. This only updates the program memory.

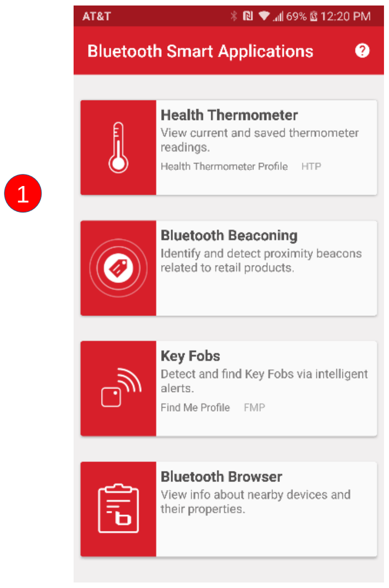


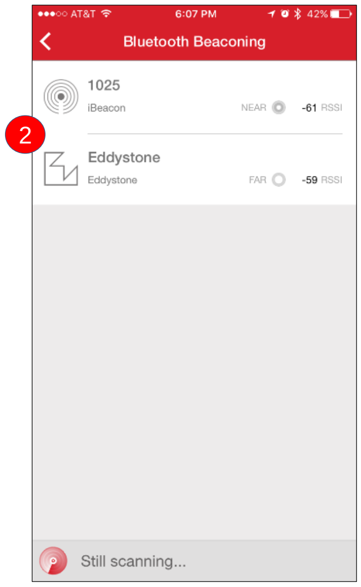
3. Once the project is flashed, the Debug perspective will be shown.
   Press the **Resume** icon in the tool bar to have the application start.



2.4 Use Mobile Apps to View Multiple Advertisements

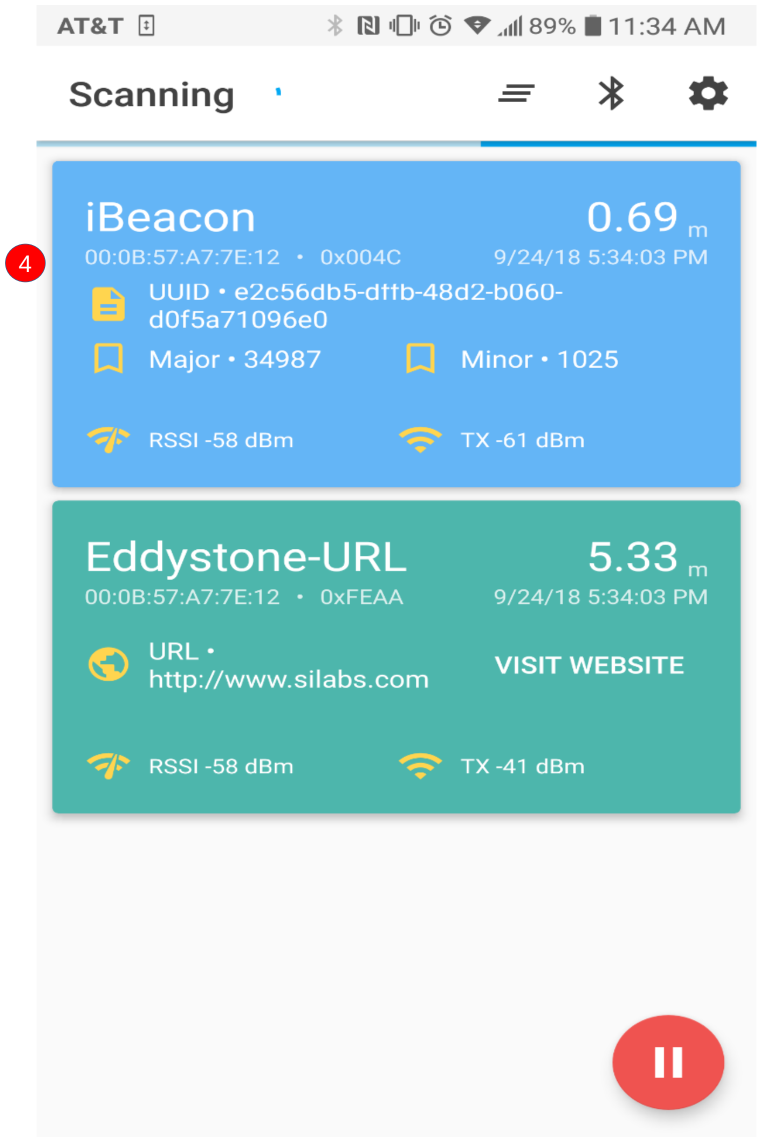
1. Open the BlueGecko app on a smartphone and tap the Bluetooth Beaconing tab.
2. There are multiple beacons broadcasting from the one device. An iBeacon and Eddystone beacon are displayed.


1
AT&T 69% 12:20 PM
Bluetooth Smart Applications
Health Thermometer
View current and saved thermometer readings.
Health Thermometer Profile HTP
Bluetooth Beaconing
Identify and detect proximity beacons related to retail products.
Key Fobs
Detect and find Key Fobs via intelligent alerts.
Find Me Profile FMP
Bluetooth Browser
View info about nearby devices and their properties.



3. Install the mobile App **Beacon Scanner by Nicolas Bridoux** from the Google Playstore if you have an Android phone or eBeacon by Lifestyle from the Apple Appstore.
   These apps scan for Physical Web advertisements i.e. Eddystone.

4. Open the Beacon app or enable notifications. The Eddystone URL is shown. You can tap the 'visit website' link to visit ww.silabs.com.



iBeacon 0.69 m
00:0B:57:A7:7E:12 • 0x004C 9/24/18 5:34:03 PM
UUID • e2c56db5-dffb-48d2-b060-d0f5a71096e0
Major • 34987 Minor • 1025
RSSI -58 dBm TX -61 dBm

Eddystone-URL 5.33 m
00:0B:57:A7:7E:12 • 0xFEAA 9/24/18 5:34:03 PM
URL • http://www.silabs.com
RSSI -58 dBm TX -41 dBm

3 Extended Advertisements

3.1 Getting Started with Extended Advertisements

In this section you'll learn about extended advertisements. This type of advertisement is used by both the LE coded PHY for any length of advertising packet and by the 1M PHY for packets longer than 31 bytes. We'll look at the LE coded PHY first.

1. Paste in the following code after the comment marked /*LAB Section 3.1 LE Coded PHY*/,

```
gecko_cmd_le_gap_set_advertise_timing(3,160,160,0,0);
gecko_cmd_le_gap_set_advertise_phy(3,le_gap_phy_coded,le_gap_phy_coded);
gecko_cmd_le_gap_start_advertising(3,le_gap_general_discoverable,
le_gap_undirected_connectable);
```

2. Change the maximum number of advertisers setting, as you did in section 2.1, to 4
3. Build the project and download it to the target board as you have in previous sections. Now the server application should be broadcasting 4 beacons : a connectable 1M advertisement, an iBeacon, an Eddystone beacon and a connectable 125k coded PHY advertisement.

4. However, this section of code has an intentional error built into it. The task here is to find out what's wrong and fix it. (hint : each of the BGAPI functions returns an error code, you can output the error code using printf(). Refer to the BGAPI reference manual for the meaning of the error codes and for information about acceptable advertising configurations). Hint: you may want to turn off scan request reporting here to limit the number of messages being sent to the console, comment out the code added in section 1.4.
5. Once you've fixed the bug, you can move on to the client. Here you can add code to test the packet type to check if the advertisement was of the extended type and print the PHY used for the primary and secondary advertisements. This information is used to determine which PHYs are actually being used by the advertiser. Paste in the following at the very beginning of the extended scan response event handler

```
                                if(0x80 & evt->data.evt_le_gap_extended_scan_re-
sponse.packet_type){

                        printf("extended advertising packet primary advertisement
on %s PHY and secondary advertisment on %s PHY\r\n",
                                                       getPhyName(evt-
>data.evt_le_gap_extended_scan_response.primary_phy).string,
                                                       getPhyName(evt-
>data.evt_le_gap_extended_scan_response.secondary_phy).string);
        /*LAB Section 3.2 Step 3*/
                    }
```

6. As a bonus exercise, you can also add code here to print out information about the address type and the bond handle. This is left to the user to do.
7. Once the code in step 5 (and optionally step 6) has been added, build the project and flash it to client board.
8. Temporarily power down the server WSTK to prevent the client from connecting to it automatically.
9. Build the application and download to the target as before
10. Press PB0. You'll see a message on the console indicating that we are now scanning on the LE Coded PHY.
11. Power up the server WSTK.
12. The client's console will now connect when it finds the desired service UUID being advertised.
13. Press PB1 to switch the PHY, any of the supported PHYs can be selected by pressing PB1 to cycle through the list. The server and client consoles each display the PHY currently in use. Note that the 500k PHY is not supported in this example.

3.2 Long Advertising Packet and the 2Mbps PHY

Another feature of extended advertising is the ability to set the PHY for the primary and secondary advertisements independently. This gives the user the ability to use the 2Mbps PHY to save energy.

1. Go to the Visual GATT Editor and change the device name to "Extended Advertising Test Device" and change the length parameter to 40.
2. In main-server.c find the comment /*section 3.1 LE Coded PHY*/ and make the following changes (the call to `gecko_cmd_le_gap_set_advertise_phy()` in this section replaces the previous one)

```
gecko_cmd_le_gap_set_advertise_phy(3,le_gap_phy_coded, le_gap_phy_2m);
gecko_cmd_le_gap_clear_advertise_configuration(3,0x01);

gecko_cmd_le_gap_set_advertise_configuration(3,0x08);

gecko_cmd_le_gap_set_advertise_tx_power(3,100);
```

Since we are re-using advertiser set 0, we'll just change the connectability mode from le_gap_undirected_connectable to le_gap_connectable_non_scannable.

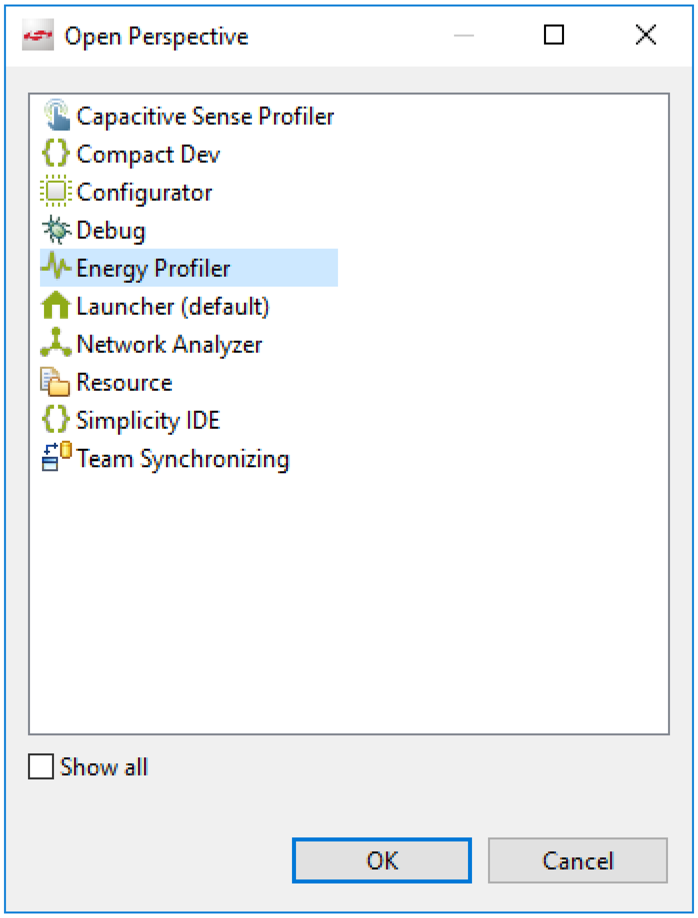
The configuration used here adds the tx power used in the advertisement set to the advertising data.

3. Now lets' modify the client to look for this advertisement and print out the name. Add the following code to the end of the if block you added in Section 3.1 Step 5

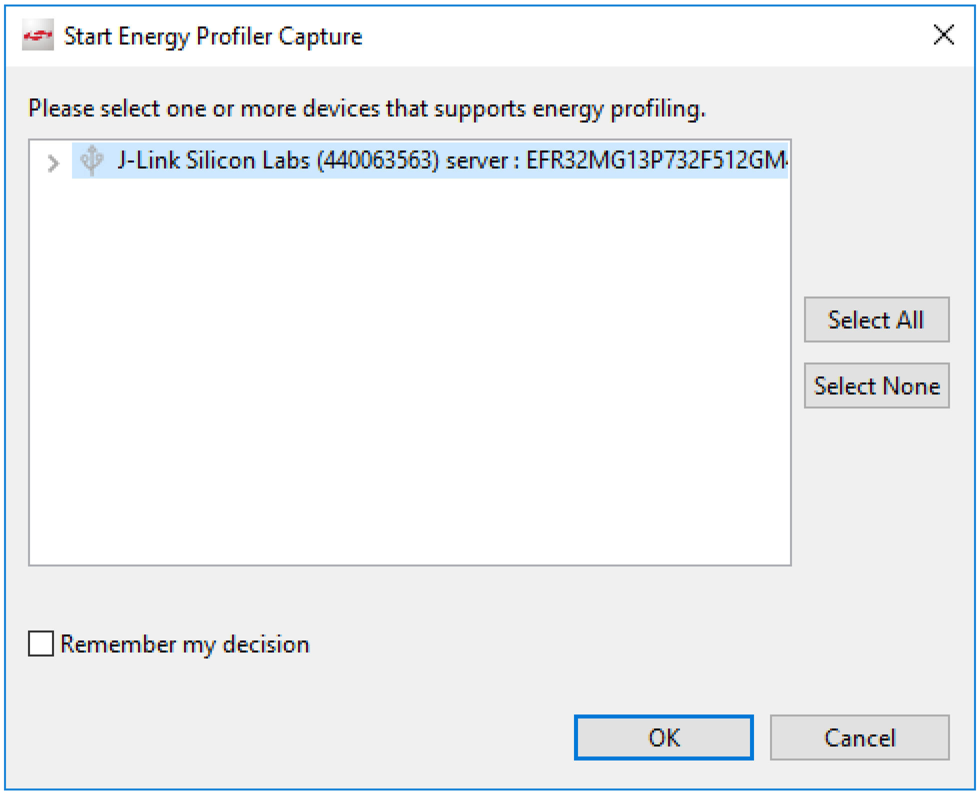
```
for(int index=0;index<evt->data.evt_le_gap_extended_scan_response.data.len;)
                          { /* find start of device name */

                                if(evt->data.evt_le_gap_extended_scan_re-
sponse.data.data[index+1] == 8||
                                            evt->data.evt_le_gap_extended_scan_re-
sponse.data.data[index+1] == 9)
                                {
                                      uint8 adv_name[35];
                                      printf("adv packet size %d\r\n", evt-
>data.evt_le_gap_extended_scan_response.data.len);
                                      printf("device name: ");
                                      snprintf(adv_name,
                                                  evt->data.evt_le_gap_ex-
tended_scan_response.data.data[index],
                                                  "%s",
                                                  &evt->data.evt_le_gap_ex-
tended_scan_response.data.data[index+2]);
                                      printf("%s\r\n",adv_name);

                                      if(127 == evt->data.evt_le_gap_ex-
tended_scan_response.tx_power){
                                            printf("advertised tx power is : unavail-
able\r\n");
                                      }
                                      else {
                                            printf("advertised tx power is : %d and
RSSI is %d\r\n",
                                                        evt->data.evt_le_gap_ex-
tended_scan_response.tx_power,
                                                        evt->data.evt_le_gap_ex-
tended_scan_response.rssi);
                                      }
                                      break;
                                } else{
                                      index += evt->data.evt_le_gap_extended_scan_re-
sponse.data.data[index]+1;
                                }

                          }
```

4. Build the client application and flash to the target as before.
5. Observe in the client's console that the client now reports the server's tx power as well as the RSSI of the advertisement.

6. To see the effects of the advertising PHY being used switch to the Energy Profiler perspective



7. In the Energy Profiler click the Quick Access menu -> Start capture then select your server device as shown below and click OK

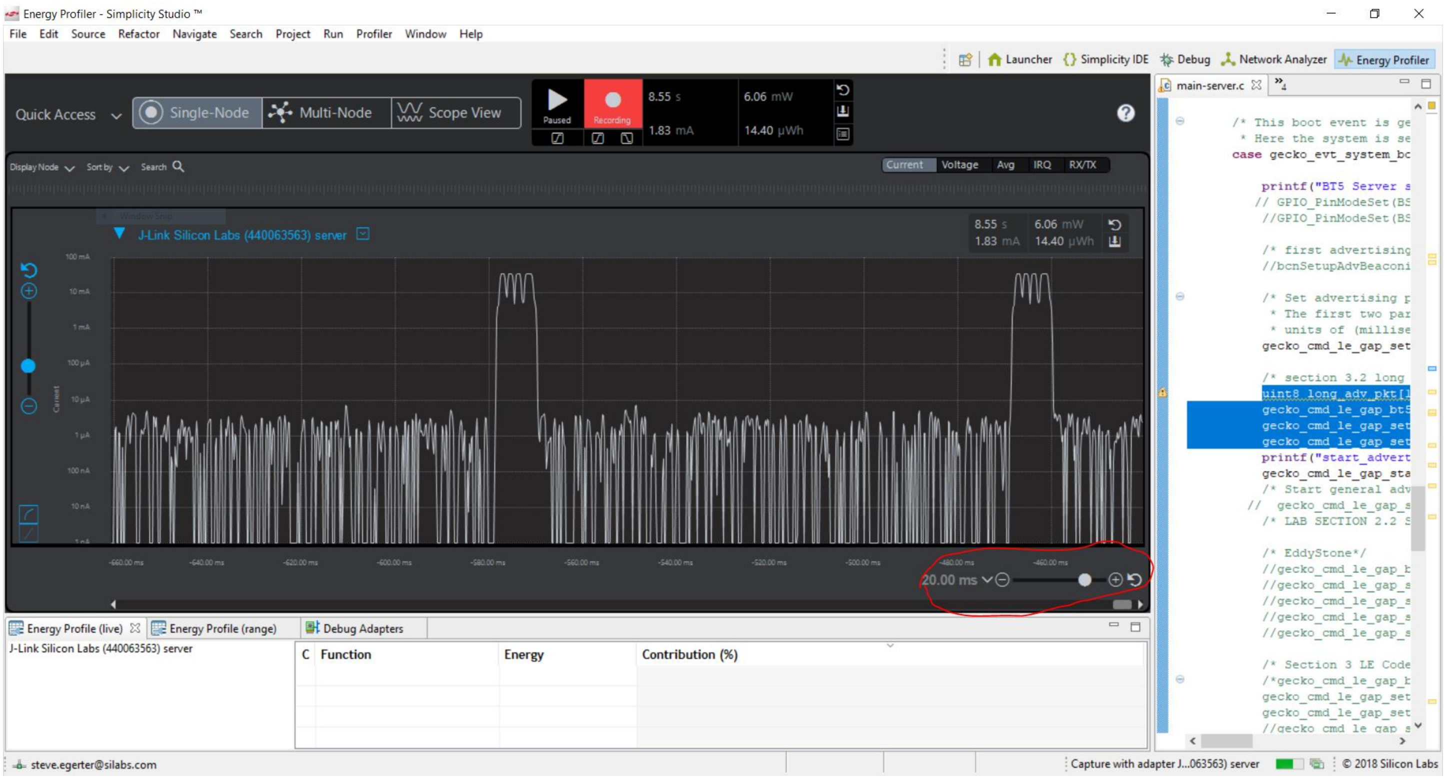


Please select one or more devices that supports energy profiling.

J-Link Silicon Labs (440063563) server : EFR32MG13P732F512GM

☐ Remember my decision

8. Let the energy capture run for a few seconds then use the slider in the bottom right corner of the screen to adjust the timebase to 20 ms so that you can see only 2 advertisements. Note the average current. (Hint: it may be necessary to power cycle the WSTK to get the results shown below)



8.55 s 6.06 mW
1.83 mA 14.40 µWh

20.00 ms

J-Link Silicon Labs (440063563) server

C	Function	Energy	Contribution (%)

```
/* This boot event is ge
 * Here the system is se
case gecko_evt_system_bc

   printf("BT5 Server s
  // GPIO_PinModeSet(BS
   //GPIO_PinModeSet(BS

   /* first advertising
   //bcnSetupAdvBeaconi

   /* Set advertising p
    * The first two par
    * units of (millise
   gecko_cmd_le_gap_set

   /* section 3.2 long
   uint8 long_adv_pkt[1
   gecko_cmd_le_gap_bt5
   gecko_cmd_le_gap_set
   gecko_cmd_le_gap_set
   printf("start_advert
   gecko_cmd_le_gap_sta
   /* Start general adv
 //  gecko_cmd_le_gap_s
   /* LAB SECTION 2.2 S

   /* EddyStone*/
   //gecko_cmd_le_gap_b
   //gecko_cmd_le_gap_s
   //gecko_cmd_le_gap_s
   //gecko_cmd_le_gap_s
   //gecko_cmd_le_gap_s

   /* Section 3 LE Code
   /*gecko_cmd_le_gap_b
   gecko_cmd_le_gap_set
   gecko_cmd_le_gap_set
   //gecko_cmd_le_gap_s
```

steve.egerter@silabs.com Capture with adapter J...063563) server © 2018 Silicon Labs

9. Now stop the energy capture using the Quickstart menu and Switch back to the SimplicityIDE perspective. Change the PHY used for the secondary advertisement to the 125k coded PHY like this

gecko_cmd_le_gap_set_advertise_phy(0,*le_gap_phy_coded*, *le_gap_phy_coded*);

Rebuild and flash to the target board again and switch to the energy profiler and start capture as before. Adjust the timebase as in the previous step and note the new average current. You should see the average current go up significantly in the second measurement..

3.3 Anonymous Advertising

In this section you'll learn about anonymous advertising. This is a new feature in Bluetooth 5 that allows an advertisement to omit the device address. Anonymous advertising is only available in extended advertisements.

1. To make the extended advertisement anonymous, change the call to gecko_cmd_le_gap_set_advertise_configuration(0,0x0A) just after the comment section 3.2, before the call to gecko_cmd_le_gap_start_advertising(). This sets the anonymous advertising flag
2. Build the project and download to the target as before and you'll see that the client reports the address as 00:00:00:00:00:00.

This is a simple feature and it completes the lab portion of the training exercise.