ECEN 5823 Spring 2025
Assignment 7 - Bluetooth BLE Client

Objective: Modify your project to support a Bluetooth BLE Client implementation, which will enable Health Thermometer indications in (from) your BLE Server, and receive and display those indications on a second Blue Gecko.

Note: This assignment will start with the completed code from Assignment 6.

References required:

Silicon Labs Bluetooth Software API Reference Manual: see https://docs.silabs.com/bluetooth/latest/

Instructions:

1. Start by creating your assignment repository using the assignment link at https://classroom.github.com/a/MWiweTqM. You will not need to clone this from the GitHub repo you just created into a new local directory since you will be starting with your code from the previous assignment. Instead, follow these instructions and run these commands with git bash within your assignment directory. Ensure you are setup to use SSH keys before proceeding, see the instructions from github at https://help.github.com/articles/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent/ and be sure to use Git Bash For Windows or Cygwin if using a Windows environment. (Cygwin .ssh directory will be different than Git Bash.)

    a. On your local PC, duplicate the folder created in Assignment #6 to a new folder, name it something like ecen5823-assignment7-<username> where <username> is your GitHub username.
    b. Change your current directory to this new folder ecen5823-assignment7-<username> and execute the following git commands.
    c. git remote remove origin
    d. git remote add origin <url>
        i. Where <url> is the URL for the repository created with the link above
        ii. This adds the new submission repository created in the link above as your new origin.
    e. git push origin master
    f. Import ecen5823-assignment7-<username> into Simplicity Studio
        i. Remember to perform a **Clean…** immediately after you import the project into Simplicity Studio.

2. Use the build switch in ble_device_type.h to control whether you are building Server or Client code. Setting **DEVICE_IS_BLE_SERVER to 1** should build your **Server** assignment from assignment 6. Setting **DEVICE_IS_BLE_SERVER to 0** should build the **Client** implementation for this assignment, which is hard coded to connect to the server address set by the **#define SERVER_BT_ADDRESS** also in ble_device_type.h,

where SERVER_BT_ADDRESS contains bytes 0 through 5 of the bd_addr struct returned by the server in sl_bt_system_get_identity_address(), you'll get a data type of bd_addr returned. You will need to set this #define value with the address of the Server you want to connect to for temperature readings. Both Server and Client builds should complete successfully on your submission. It should not be necessary to set any values other than this single build switch (#define DEVICE_IS_BLE_SERVER) to change between compiling Server or Client implementations.

3. In your boot event handler, set the connection parameters to:
    a. min_interval = 75 ms
    b. max_interval = 75 ms
    c. slave_latency = 4
    d. Supervision timeout = (1 + slave_latency) * (max_interval * 2) + max_interval
    e. min_ce_length = 0 (recommended for current Gecko 3.x SDK)
    f. max_ce_length = 4

Pay attention to how the values of the parameters passed to this routine are scaled to produce final values in ms!

4. Add support for each of the events and commands included in your "Assignment 6 - Client Command Table for A7" spreadsheet for the case where DEVICE_IS_BLE_SERVER is set to 0. You must maintain the complete and full functionality of your assignment 6 Server code. If you discover that you need to make changes to your command table, please update your command table to keep it in sync with your Client code implementation.
    a. The Client firmware shall support on each open event:
        i. 1-time discovery of the HTM service using its service UUID
        ii. 1-time discovery of the Temperature Measurement characteristic using the HTM thermometer characteristic UUID
        iii. 1-time enabling of the HTM temperature indications. Disabling indications for the HTM characteristic is not required.
        iv. Receiving the temperature indications from the Server and displaying it on the Client's LCD.
    b. Extra credit (+10 points) is available if you include a bubble state diagram of your discovery state machine in your command table spreadsheet for A7.
    c. Ways to lose points:
        i. Your new discovery state machine implementation if implemented fully or partially in your ble.c code (handle_ble_event()) will result in 0 points on this assignment. Your BLE event handler should set variables/ flags that your new discovery state machine examines, for example: Did the Client receive a HTM Service handle in response to sl_bt_gatt_discover_primary_services_by_uuid() ?

   ii. Your discovery state machine transitions shall be driven by events from the BT stack, just like our Server's temperature taking state machine was driven by events. **Do not burden your discovery state machine with any other functionality other than the discovery process!** The events for your new discovery state machine shall be:
      1. Open
      2. Gatt Completed
      3. Close
   iii. If you convert events in handle_ble_event() into flags that your state machine solely transitions on, you shall receive 0 points on this assignment.
   iv. If you leave your Server's temperature state machine running in your Client build, you will loose 50% of your grade on this assignment.

5. For scanning, use passive scanning and set the scan interval to 50ms and the scan window to 25ms.

6. Update your LCD display with new status information:
   a. Add displayPrintf() calls to set the "DISPLAY_ROW_CONNECTION" row with the display states you will now add in the "LCD Display State" column of the command table for the new Client design.
      i. Booted up and **not** in a connection, DISPLAY_ROW_CONNECTION should display Discovering
      ii. Once connected, DISPLAY_ROW_CONNECTION should display Connected. You may not see this displayed on your LCD as the state change from "Connected" to "Handling Indications" happens quite quickly.
      iii. Once your Client code has enabled HTM indications to the Server, DISPLAY_ROW_CONNECTION should display Handling Indications
   b. Add displayPrintf() calls to set the DISPLAY_ROW_NAME with value "Client" or "Server" depending on the build selected. In either case, the DISPLAY_ROW_BTADDR should use the value of the bluetooth address obtained from gecko_cmd_system_get_bt_address().
   c. For the Client build, the DISPLAY_ROW_BTADDR2 value should be set with the hard coded address to the Server Blue Gecko used for testing (SERVER_BT_ADDRESS). This row should be cleared when not in a connection.
   d. For the Client build, the DISPLAY_ROW_TEMPVALUE row should show the temperature indicated from the connected Server, or be blank if no Server is connected.

i. For printing or displaying the temperature you will need the inverse of the FLT_TO_UINT32 macro you used to write the value to the buffer in your Server code. You can use the function below or one like it for this purpose.

```
/ -----------------------------------------------

// Private function, original from Dan Walkes. I fixed a sign extension bug.

// We'll need this for Client A7 assignment to convert health thermometer

// indications back to an integer. Convert IEEE-11073 32-bit float to signed integer.

// -----------------------------------------------

static int32_t FLOAT_TO_INT32(const uint8_t *buffer_ptr)

{

  uint8_t     signByte = 0;

  int32_t     mantissa;


    // input data format is:

    // [0]       = flags byte, bit[0] = 0 -> Celsius; =1 -> Fahrenheit

    // [3][2][1] = mantissa (2's complement)

    // [4]       = exponent (2's complement)


  // BT buffer_ptr[0] has the flags byte

  int8_t exponent = (int8_t)buffer_ptr[4];


  // sign extend the mantissa value if the mantissa is negative

    if (buffer_ptr[3] & 0x80) { // msb of [3] is the sign of the mantissa

      signByte = 0xFF;

    }


  mantissa = (int32_t) (buffer_ptr[1]  << 0)  |

                       (buffer_ptr[2]  << 8)  |

                       (buffer_ptr[3]  << 16) |

                       (signByte       << 24) ;


  // value = 10^exponent * mantissa, pow() returns a double type

  return (int32_t) (pow(10, exponent) * mantissa);


} // FLOAT_TO_INT32
```

e. Add displayPrintf() to display A7 on row DISPLAY_ROW_ASSIGNMENT for both your Client and Server builds for this assignment.

7. Test your firmware implementation as shown in the "Testing" section below to ensure your implementation is complete. You will require a second Blue Gecko board for this assignment. Set the value of SERVER_BT_ADDRESS for your Client build to match the address of the Server as displayed on the LCD of the Server Gecko.

8. For the code you submit for grading, there should be no calls to __WFI() or __BKPT() or LOG_***() functions. No calls to Systick routines are allowed. However, do log errors returned from API calls.

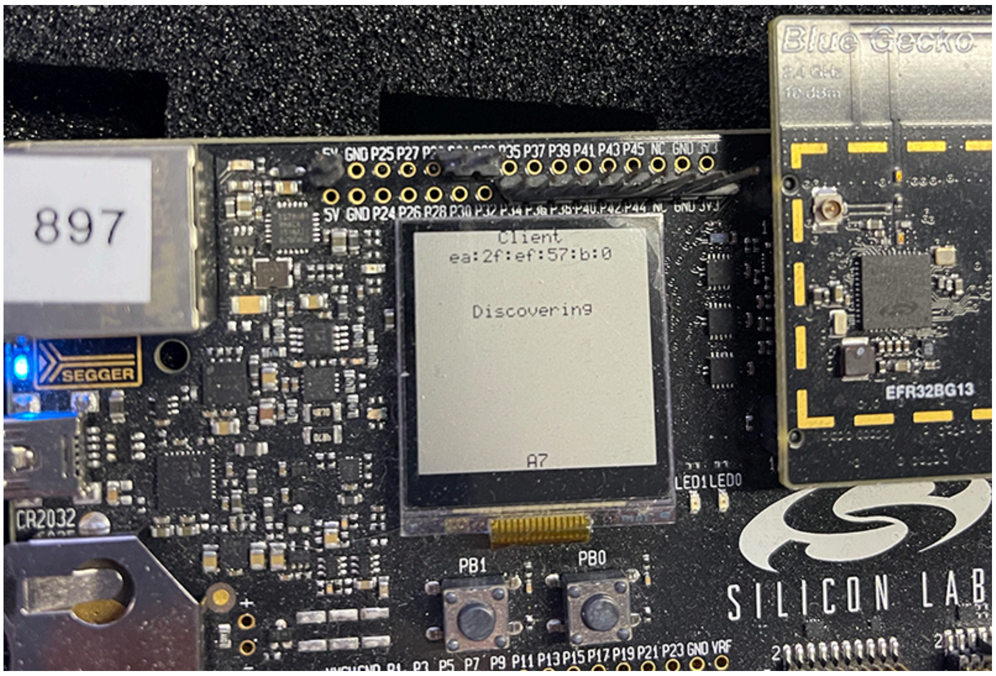
Testing:

Your Client firmware should pass the following tests:

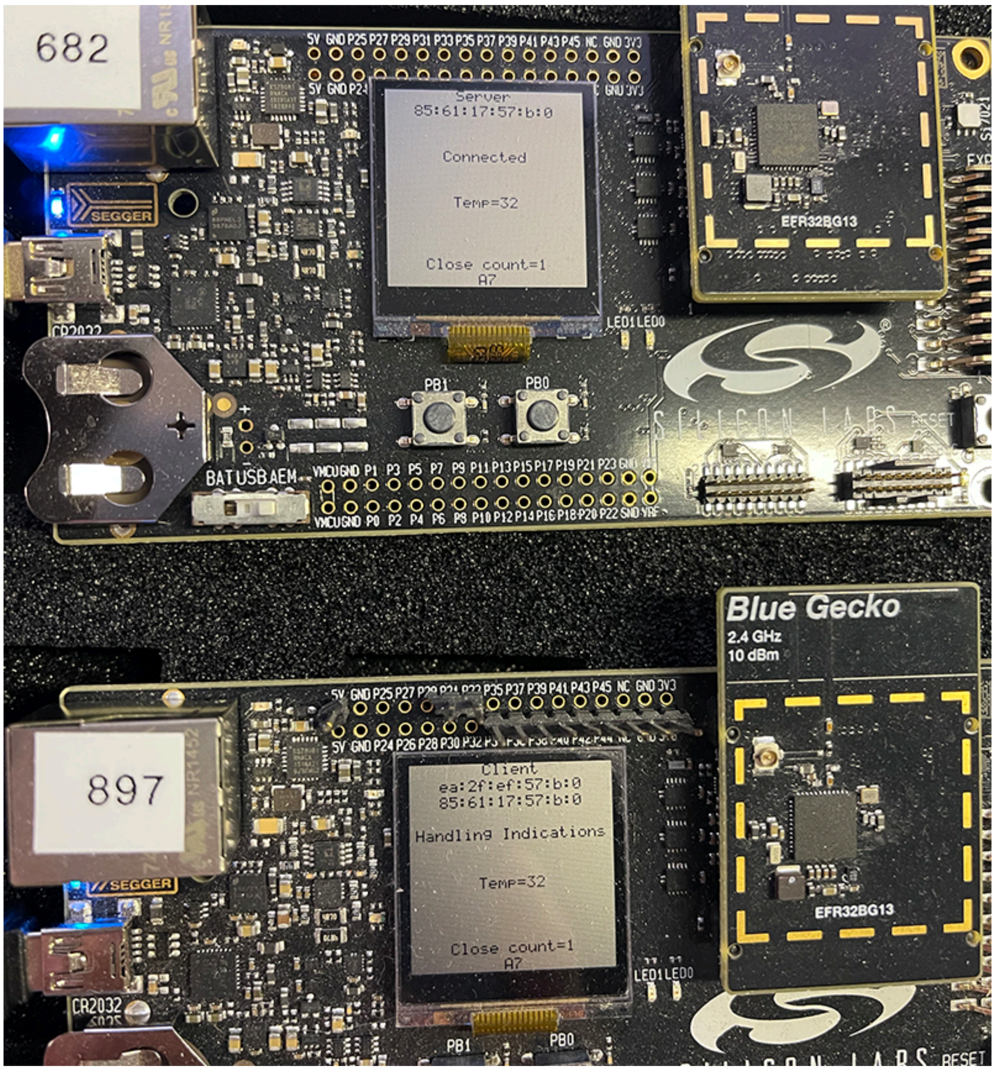
1. When powered on the first line should display Client. The Client's bd_addr should be shown in the second line of the display.
2. On initial power up the 5th line of the display should show Discovering when no Server is present and while your new state machine is "Discovering", and row DISPLAY_ROW_TEMPVALUE should be blank.
3. When first connected to the Server the 5th line of the display should briefly show Connected and then show Handling Indications once your Client has enabled HTM indications in your Server. The connected Server's bd_addr should be displayed on the third line of the display. The temperature should be displayed and updated continuously on the display using HTM Indications. The temperature displayed on the Client should be the temperature of the Server's Si7012 and not the Client's!
4. When the Server is powered off, the 5th line of the display should show Discovering and row DISPLAY_ROW_TEMPVALUE should be blank. The DISPLAY_ROW_BTADDR2 row of the display should be blank.
5. Powering the Server or Client off and back on should automatically reconnect and restart display updates. It should not be necessary to reset your Client device to restart HTM indications and display updates on either the Server of the Client.
6. It should be possible to change your Client implementation to access a different Server by changing the value of SERVER_BT_ADDRESS in ble_device_type.h.
7. It should be possible to build your working Server implementation by changing the value DEVICE_IS_BLE_SERVER to 1 in ble_device_type.h. Again, both A7 builds of Client and Server should show A7 on row DISPLAY_ROW_ASSIGNMENT.

Here are pictures of the Client and the Server.

Client powered up looking for your Server and performing the discovery process:



Client connected to your Server and receiving HTM indications from your Server:



Note: For my own debugging purposes I used the LCD row above the assignment number as a counter of closed events. **This is not required for this assignment, however, it can be helpful to use this row to display diagnostic values**. For the build configuration you submit for grading, please disable all diagnostic LCD messages and logging statements.

Deliverables:

- An updated (if it was necessary as per instructions above) Assignment 6 - Client Command Table for A7 spreadsheet saved in the "Questions" folder. The content should match your implementation.
- The completed assignment 7 program project to be run on the instructing team's computer for grading.
- Submission via **github classroom** and your repository setup via the classroom link at above. Use the following git command to submit your assignment: git push origin master
- Verify your ecen5823-assignment7-<username> repository contains your latest/ intended code and files in the questions folder. It is your responsibility to push the correct version of your project/code. Use a web browser to verify the correct version of your project/code has been uploaded to GitHub.
- In **Canvas**, submit the URL to your github repository and in **GitHub** create a tag of your GitHub submission prior to the due date with the text of A7-final. The date and time of the tag creation is what we use to determine whether your submission was on time or late. These 2 steps will complete your assignment submission.

Approach/Guidance

The overall approach I used to work my way through the events as they occur in roughly chronological order.

```
sl_bt_evt_system_boot_id

sl_bt_evt_scanner_scan_report_id

sl_bt_evt_connection_opened_id

sl_bt_evt_gatt_service_id

sl_bt_evt_gatt_characteristic_id

sl_bt_evt_gatt_procedure_completed_id

sl_bt_evt_gatt_characteristic_value_id

sl_bt_evt_connection_closed_id
```

Similar to assignment 6:

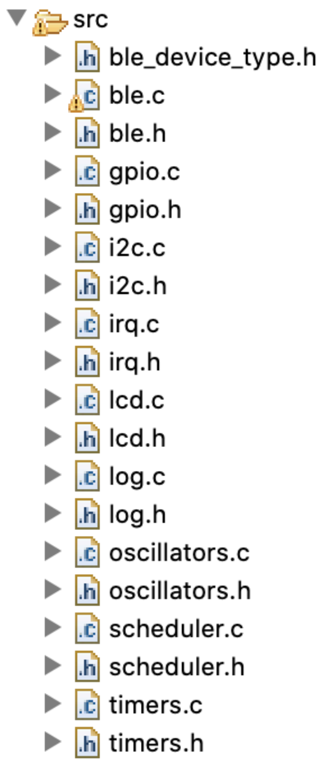
1- Start with your app.c/.h and scheduler.c/.h code. Go through your code line by line, function call by function call, adding switches for DEVICE_IS_BLE_SERVER. Build your new discovery state machine and call that from app.c in function sl_bt_on_event(). Use #if to compile-out your Server's temperature taking state machine in sl_bt_on_event().

2- Now work your way through ble.c, again adding switches for DEVICE_IS_BLE_SERVER and implement the events your Client has to respond to. Get your code working for the BT Boot event. Make sure you are displaying: Client, the Client's BT address and Discovering display state. Ideally you would organize your BLE event handler function into 3 sections: one section handles events common to both Servers and Clients, one section handles events for Servers only, and one section handles events for Clients only.

3- As you work on your Client code, periodically flip the DEVICE_IS_BLE_SERVER switch back to SERVER to make sure your server code still compiles and functions correctly. Use your A7 Server build, which should function identically to your A6 Server code, to test your A7 Client code.

Also see the **Bluetooth - SoC Thermometer Client** example in Simplicity Studio. This example program can be very useful to learn from. Take what you learn there and apply it to the guidance I've provided.

Files in your src/ directory should look like:



```
src
  ble_device_type.h
  ble.c
  ble.h
  gpio.c
  gpio.h
  i2c.c
  i2c.h
  irq.c
  irq.h
  lcd.c
  lcd.h
  log.c
  log.h
  oscillators.c
  oscillators.h
  scheduler.c
  scheduler.h
  timers.c
  timers.h
```