ECEN 5823 Spring 2025

Assignment 10: Bluetooth Mesh

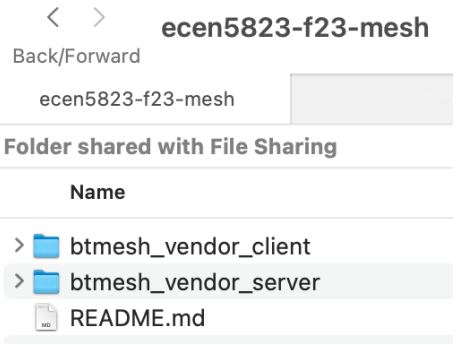
Objective: Create firmware to support two unique Bluetooth Mesh nodes, including a Server node and a Client node. The Server node contains a synthetically created (made up) temperature measurement. The Client node can send requests to the Server to read the temperature and send some other requests to the Server as well. The Server node acts as a Friend Node, and the Client node acts as a Low Power Node (LPN). When the Client node is initialized after reset it will start searching for a Friend node to establish friendship by sending out friendship requests. The Server node will respond to the friendship requests and become a friend for the Client/LPN.

This code also demonstrates vendor models, and a type of provisioning referred to as "Self Provisioning". Vendor models can support their own set of vendor defined messages (think: commands). Self provisioning is simply having the model call a few API calls that bind hard-coded keys for the NetKey and the AppKey to the main model in each node. This is a great method to use when trying out new ideas in the lab, or in academia. It is a terrible idea to use in production as the NetKey and AppKey are hard-coded into the firmware. It would not take long for a sophisticated hacker to discover these key values and use them to mount an attack against your Mesh Network. So never use Self Provisioning in production products!

Note: This assignment will start with a new set of starter code. The starter code in the GitHub repository contains 2 Simplicity Studio project folders:

- btmesh_vendor_server/ (Server node project code)
- btmesh_vendor_client/ (Client node project code)

The contents of the GitHub repo looks like this:



The README.md at the top-level of the repo contains a brief summary of the project. There are more extensive README files in both btmesh_vendor_client/ and btmesh_vendor_server/ sub-folders. Please read these README files. These README files describe the design and the messages that can be sent from one node to another node when the push buttons on the Gecko are pressed.

Each of these projects needs to be imported into SSv5 for this assignment. When you complete the required code editing, all edited files in both projects need to be pushed back to your GitHub repo.

Looking inside the btmesh_vendor_server/ folder we can see:

Note: There are no src/ or questions/ subdirectories in these projects. All edits required for this assignment are in the app.c files: btmesh_vendor_server/app.c and btmesh_vendor_client/app.c. In each app.c file, see the comment: “Student edit:”, implement what the instructions say there. This configuration of the starter code limits the Friend to support up to 4 LPNs.

Remember: Bluetooth Mesh is a networking technology and Bluetooth Low Energy is a wireless technology.

Instructions:

1. Similar to assignment #1, start by creating your assignment repository using the GitHub Classroom assignment link at https://classroom.github.com/a/BiwHieMn.

Click the link to accept the assignment and then navigate in your web browser to your newly created empty repository.

2. Clone that repository to your simplicity studio workspace directory (~/SimplicityStudio/v5_workspace/ where ~ is your home directory or C:\Users\username on windows).
    a. Use the simplicity studio workspace directory since when located outside this directory some features **such as project rename** are not supported in the IDE.
    b. See https://help.github.com/articles/cloning-a-repository/ for clone instructions from github.

Linux example:

```
% cd ~/SimplicityStudio/v5_workspace
% git clone https://github.com/CU-ECEN-5823/ecen5823-assignment10-cchoi22915.git
```

cchoi22915 is my GitHub username. A new directory will be created in ~/SimplicityStudio/v5_workspace/ called ecen5823-assignment10-cchoi22915/

Populate the local directory with code from the Mesh starter code repository:

```
% cd ecen5823-assignment10-cchoi22915
% git remote add assignments-base https://github.com/CU-ECEN-5823/ecen5823-mesh.git
% git fetch assignments-base
% git merge assignments-base/master [1] (may need –allow-unrelated-histories)
% git push origin master
```

Use a web browser to look at the files in your repo on GitHub. You should see:

```
btmesh_vendor_client/
btmesh_vendor_server/
.gitignore
README.md
```

You should see the same files in your local directory of this repo as well.

    c. Import projects btmesh_vendor_client and btmesh_vendor_server into Simplicity Studio - **you will have to import by "Projects from Folder or Archive"**
        i. Remember to perform a **Clean…** immediately after you import the projects into Simplicity Studio.

[1] Some Windows users get a "Filename too long" error on the merge command. If this happens try these to commands prior to the merge:

```
% git config --system core.longpaths true
% git fsck --name-object
```

3. These projects are set for Gecko SDK 4.3.2 and Mesh 5.0.2. You can right-click on the project in the Project Explorer pane, goto C/C++ Build / Board / Part / SDK. Near the bottom of that pane is an SDK: pop-up menu, check it there.

4. Prior to editing, verify each project builds successfully with the default project starter files. Ignore warnings about variables set but not used. Those warnings should go away after you make your required edits.

5. Open each project's app.c file, search for "Student Edit", and make the edit described there.

6. You should not need to make any changes or modify the .slcp files as they already contain the required Software Components and Mesh models. Please explore the Software Components and the Bluetooth Mesh Configurator to see what is available in there, and how the Mesh model is configured.

7. In order for the nodes to function correctly a "Factory Reset" is required. There was a bug in the code from SiLabs for the Factory Reset, which I have fixed in this code. See app.c, sl_bt_on_event(), event = sl_bt_evt_system_boot_id. To perform a factory reset, do this on each node:
    a. Press and hold down PB0.
    b. Press and release the Reset button on the Gecko.
    c. Release PB0. You should briefly see "Factory Reset" display on the LCD.
    d. The nodes will then establish friendship and will be in a state where they can start exchanging messages.

8. Use this assignment to study how SiLabs has implemented call backs, vendor models, defined message types (opcodes and data payloads), group addresses, and the API calls to Self Provision, and send & receive messages. This code can serve as a nice starting point for your Course Project, because with vendor models you are able to define a wide range of behavior and messages that are not required to conform to the BT SIG defined Mesh Models.

9. Use the testing steps below to verify your implementation.

10. Helpful resources:
    a. QSG148 bluetooth mesh quick start guide
    b. AN1098 understanding bluetooth mesh lighting demo
    c. UG366 bluetooth mesh node configuration users guide
    d. Mesh Technology Overview
    e. Mesh Model Specification
    f. Mesh glossary: https://www.bluetooth.com/learn-about-bluetooth/bluetooth-technology/topology-options/le-mesh/mesh-glossary/

g. Mesh Learning Center: https://www.silabs.com/products/wireless/learning-center/bluetooth/bluetooth-mesh?cid=eml-nur-ble-021418&mkt_tok=eyJpIjoiTVRaalpESTVNRFV5TnpsaiIsInQiOiJ6cDZFejF6aWhJZSt1QUdJeVJsMkdsZE1SZjRORW1rY3pKSmNnNlZ4Q2R6SnUrbEF2T3duMmlcLzJUNzdHZWhISEJFSDI0aExGWlQ2WTdvdFR5V1wvNVJXdko3UFdjc3JRTWttVUREcCtYU1ByYXJlM2lhZEFCRWVBRGxhUTQ3MWtUIn0%3D

Testing

Messages transmitted by the nodes are based on either short or long presses of the push buttons. The duration of what constitutes short and long is somewhat arbitrary, but is documented in the Software Component: App / Utility / Button Press, and in the code in the function app_button_press_cb(). The messages transmitted by short vs. long button presses are explained in the README files in each project subdirectory. Download the Client code to one of your Geckos, and download the Server code to your other Gecko. Your firmware should pass the following tests.

Tests on the Client/LPN

1. PB0 Press (a short press) should send a GetTemperature message to the group address that the Server has subscribed to, and will be received by the Server. The Server will respond with an Acknowledgement to the Client containing the temperature measurement. The Client will display the temperature both in the terminal.
2. There is no need to test the periodic temperature update functionality by PB0 Long Press. You can play around with that if you want, but it is not graded. There could be applicability to your Course Project, and therefore there may something for you to learn by studying this mechanism.
3. PB1 Press (a short press) should send a GetUnits message to the group address. The Server should respond with the units (Celsius or Fahrenheit). The Client will display the units in a terminal.
4. PB1 Long Press should send a ToggleUnits message to the group address. The Server should respond with the new units (Celsius or Fahrenheit). Subsequent GetTemperature messages sent from the Client should be displayed in the new (toggled) units.

Tests on the Server/Friend

1. PB0 Press (there's no long/short implemented) should publish the temperature to the group address that the Client(s) subscribes to. The Client will display the temperature in a terminal.
2. PB1 Press (there's no long/short implemented) should publish the current units (Celsius or Fahrenheit) to the group address that the Client subscribes to. The Client will display the units in a terminal.

Deliverables:

- The completed programs for both projects in the repo, 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-assignment10-<username> repository contains your latest/ intended code and files. 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. Do this immediately after accepting the GitHub Classroom assignment - there is no reason to wait! In **GitHub** create a tag of your GitHub submission prior to the due date with the text of A10-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.