University of Colorado Boulder

ECEN 5823 – IoT Embedded Firmware (Spring-2020)

Final Project Individual Report

Project Details	Description
Created By:	**Rushi James Macwan**
Project Title:	**Greenhouse Plant Monitoring System using Bluetooth Mesh**
Platform Used:	**Silicon Labs EFR32 Blue Gecko 13**
Report Submission Date:	***April 30th, 2020***
GitHub Repository	***https://github.com/CU-ECEN-5823/final-project-assignment-MacRush7***
Provisioner Repository	***https://github.com/CU-ECEN-5823/mesh-provisioner-MacRush7***

Credits: *All content presented in this proposal is my own work and all references are duly credited.*

Please Note: *This final project report has been created based on the documentation that was originally created by me for my two project updates spread out across the project development cycle.*

University of Colorado Boulder

Table of Contents

Subsystem Overview

Description:

My project subsystem is focussed on the design and development of a Bluetooth Mesh Friend Node (FN). The FN has been interfaced with sensors/actuators for interacting with the external environment properties within the greenhouse (e.g. temperature, ambient light level, etc.). The FN showcases a communication network with three Bluetooth Mesh Low-Power Nodes (LPNs). The LPNs that are part of the rest of the project will send fixed-interval notifications to the FN based on the model states of the services available on the LPNs. The LPNs send alarm notifications during its fixed-interval notifications duty-cycle when the set thresholds existing as persistent data on the LPNs are violated by the sensor readings acquired from the sensors interfaced with the LPNs.

The FN will deal with the notifications arriving from the LPNs and will display the same in a user-friendly format on its LCD. The FN will further derive its own external environment properties using the MCP9808 I2C High-accuracy temperature sensor available on it. Based on the model states updated by the external sensor connected with the FN and the model state alarm notifications received from the communicating LPNs, the FN will display sensor level information fed from the LPNs and will also report alarms when thresholds are violated on the LPNs and is conveyed to the FN.

The FN stores the alarm statuses as persistent data using the on-board flash memory. These alarm statuses are triggered by the threshold violations and alarms cleared thereafter by the LPNs for the information that they are acquiring from the sensors interfaced with the LPNs.

Apart from handling the model state notifications arising out of its own sensor or the LPNs, the FN will also maintain a Bluetooth Mesh network with the LPNs by providing a BTM friendship established through the Mesh architecture created through a Publish-Subscribe/Client-Server system. The FN acts as a Subscriber/Server for all communication purposes with active friendships established with the LPNs.

Subsystem High Level Design

How does it alleviate/solve the problem?

- The subsystem realized by the FN in this project proposal solves the problem of tedious monitoring using human resources of the greenhouse plants.
- Through sophisticated Bluetooth Mesh based connected IoT devices providing a real-time, reliable and accurate solution, the operational costs to hire human resources is eliminated while increasing the operational efficiency by overcoming the human response latencies and associated risks.
- It will alleviate the problem by further providing real-time support for monitoring the greenhouse environmental properties while also automating the cause and act relationship to produce desirable results.
- The FN through collective network interfaces from the LPNs, in general, will provide equipment control and remote management abilities for the greenhouse by reporting the environmental properties (e.g. 31% moisture).
- The FN would also provide data logging, reporting of essential model states and alarm statuses. In a practical setting, this prototype would be able to provide machine learning abilities in commercial product deployments to increase operational efficiencies, reduce operational costs, and extend flexibilities for plant operations while increasing revenues, profits and quality of throughput.
- From a systems perspective of the subsystem design, the FN responds to the services available on the LPNs that are updated at a lower frequency than its own (FN). While doing that, it manages its own service and uses its state machine to service its own sensor interface (i.e. MCP 9808 Temperature Sensor).

Credits: *Please note, the above solutions to alleviating/solving the problem has been sourced by the same resources as used for the "example use cases" section of the group project proposal.*

List of sensors/actuators required:

The list of sensors/actuators required for this subsystem are as follows:

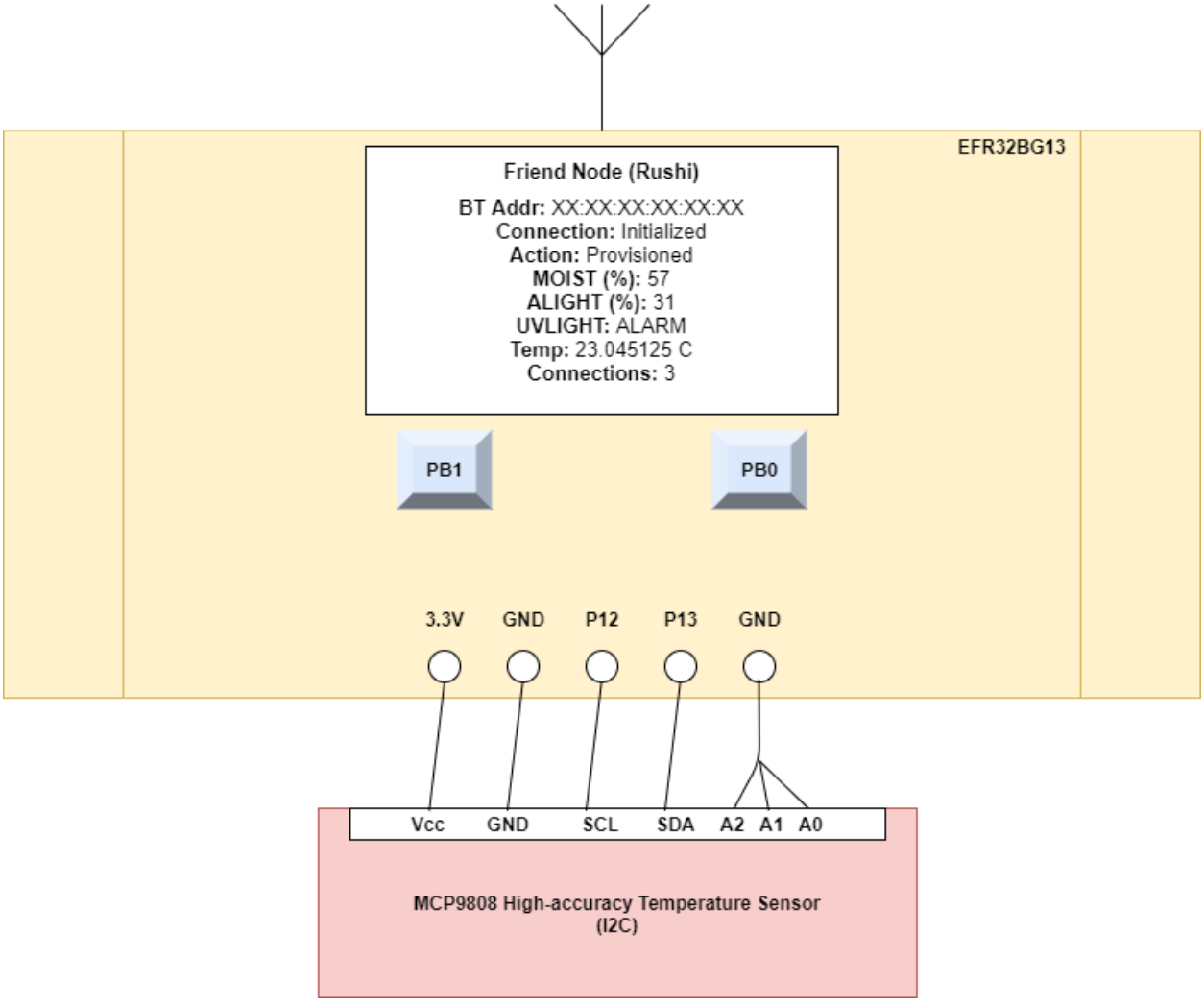
Sensors:

1. MCP9808 I2C High-accuracy temperature sensor – *already have access to this sensor*

The MCP9808 I2C High-accuracy temperature sensor will be connected via the I2C protocol with the EFR32BG13. The underlying firmware for this will be sourced from the one that was created for the on-board Si7021 I2C temperature sensor.

Functional block diagram of the subsystem with additional information:

The block diagram of the subsystem is provided as under:



The MCP9808 sensor will be interfaced with the EFR32BG13 Port-C pins (SCL – pin 10 and SDA – pin 11). The temperature sensor will connect with the I2C0 port on the EFR32BG13. Besides that, the A2-A1-A0 addressing pins on the sensor will be all connected to ground to acquire a slave address of 0x18 on the sensor for I2C communications.

BTM Command Table:

Please find the command table here.

List of exposed services:

The list of exposed services on the FN will essentially include the following:

1. Temperature Service (provided using the on-board connected MCP9808 temperature sensor)
2. Moisture Service (provided via LPN transactions)
3. Ambient Light Service (provided via LPN transactions)
4. UV Light Service (provided via LPN transactions)

The above services include one (Temperature Service) that is completely implemented on the FN for direct accessing and controlling. The remaining three services are acquired using the Generic On/Off or Generic Level Models used to acquire sensor information from the LPNs on the FN.

Description of persistent data:

This project make use of persistent data in the form of storage of alarm statuses using an alarm buffer in the flash memory as part of persistent data. Whenever the FN is provisioned, it establishes friendships with provisioned LPNs and then it begins the process of acquiring alarm statuses for the threshold violations on the LPNs. After every update/notification that is acquired from a LPN, the FN BTM stack event handler updates the alarm buffer on the FN with the alarm status as updated by the newly acquired notification. Further, the firmware stores this updated alarm buffer to the flash memory as persistent data. Under unexpected power cycles, the FN can retain the alarm status that was updated/stored earlier and can avoid missing or creating false alarms.

User interfaces:

The LCD interface provides a very useful way to interact with the FN. The LCD display on the FN exhaustively provides the following parameters (as per the order of LCD display rows):

1. Device Name (e.g. Friend Node (Rushi))
2. Device BT Address (e.g. XX:XX:XX:XX:XX:XX)
3. Device Connection Status (e.g. Initialized)
4. Device Action Status (e.g. Provisioning/Provisioned)
5. Moisture Level (e.g. 39%)
6. Ambient Light Level (e.g. 51%)
7. UV Light Level (e.g. 23% or ALARM)
8. Temperature Reading (e.g. 23.913094 C)
9. Active Connections (e.g. 3)

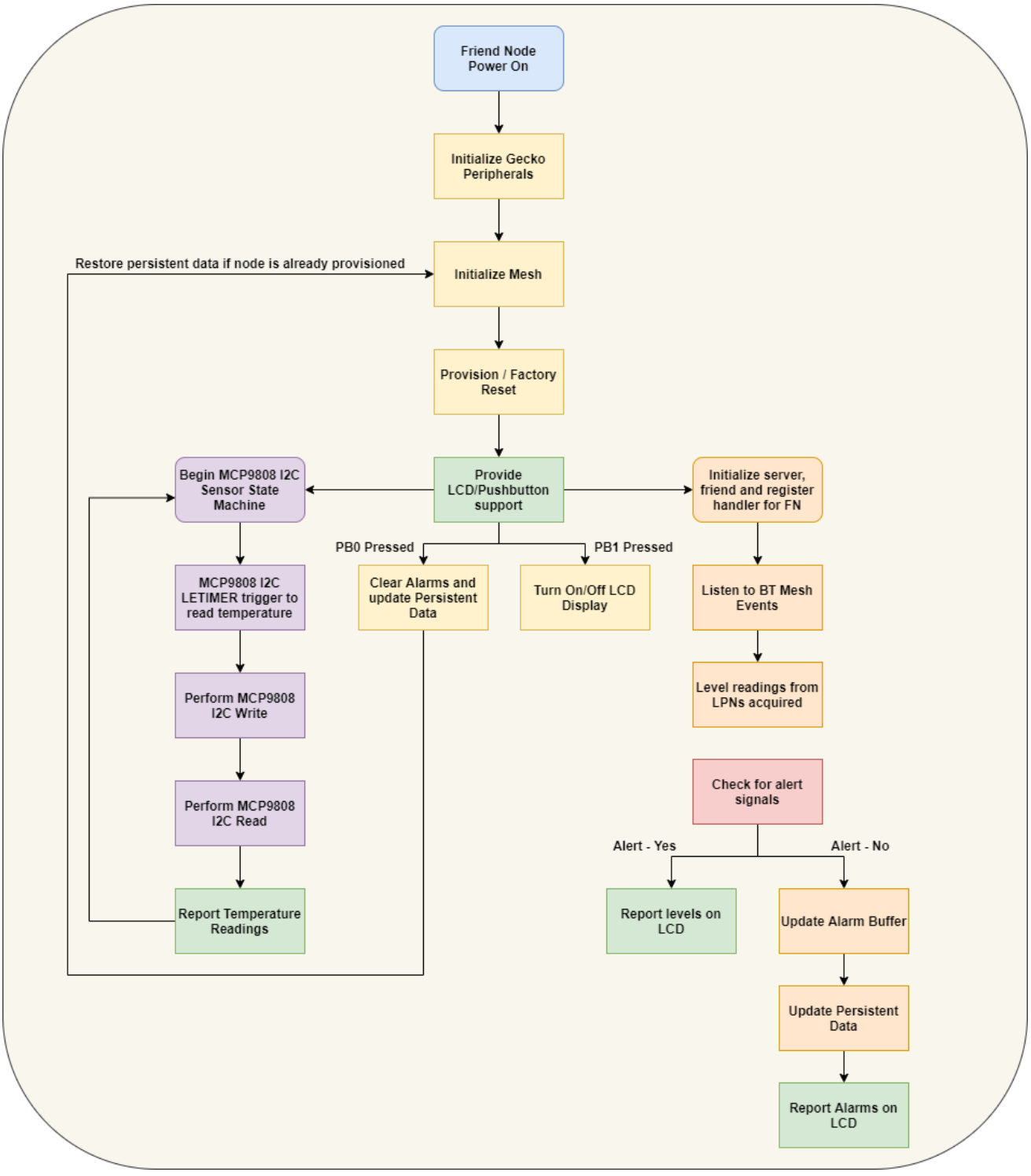
Besides the above provided exhaustive display features, the FN allows a quick interaction with the LCD display using the pushbuttons PB0/PB1. By pressing pushbutton PB0, the alarm statuses are all cleared and the alarm buffer is therefore cleared. The alarm buffer copy stored as persistent data in the flash memory is

also updated. This means that all alarms are cleared and the display is refreshed to either specify an absence of data or new data levels that are available but no alarms from the previous iteration when the alarms were not cleared.

In addition to the above feature, pressing pushbutton PB1 once the device is powered on and set (after moving forth the device factory reset interval), the PB1 presses allows turning on/off the LCD display. This helps to save power although the FN is not on an energy budget and also increases the flexibility to interact with a user.

Block Diagram for Firmware Architecture

As an additional section, this sub-section throws some light on the subsystem firmware architecture that runs the entire firmware on the FN. The below firmware architecture block diagram provides an exhaustive view of the firmware development that has taken place in this subsystem for the project:



Final Project Development Schedule

Based on the development history of this project and the information shared by our Team Validation Plan, the final project development schedule is briefly provided below:

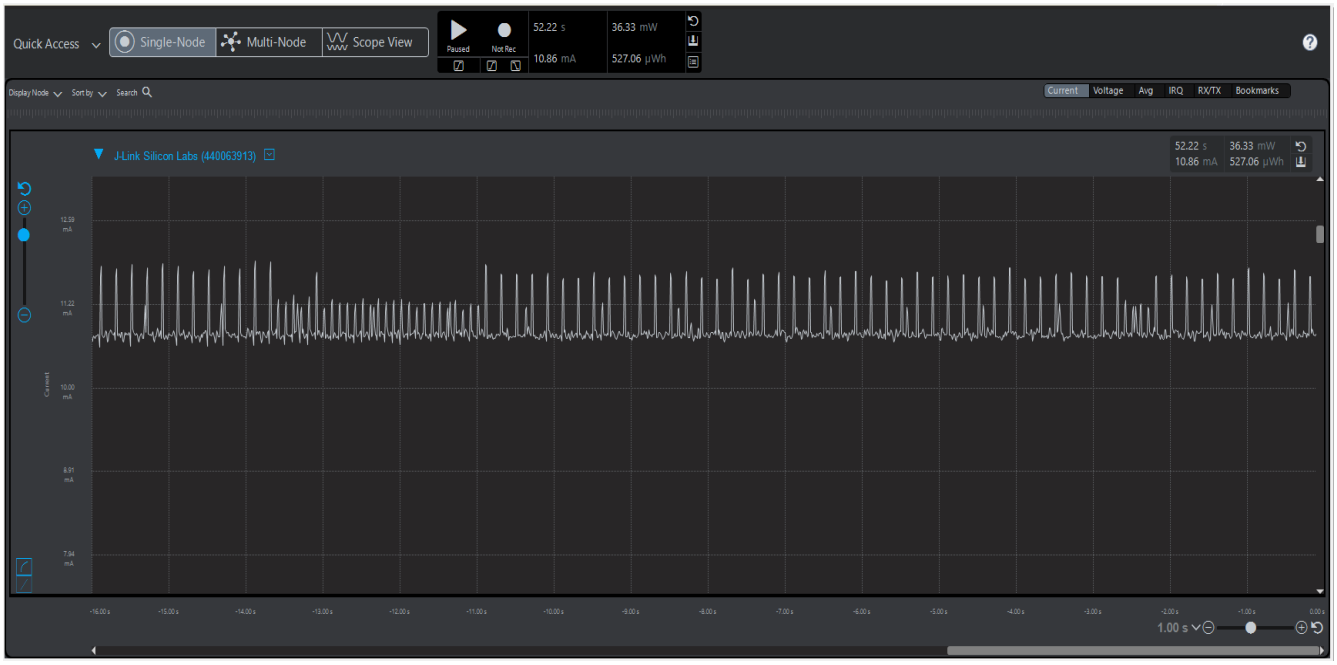
Target Dates	Targeted Implementation
14th April 2020	Implemented fundamental FN BTM functionality (Added factory reset, provisioning and BTM initialization features)
14th April 2020	Implemented BTM FN – LPN relationships
21st April 2020	Interfaced MCP9808 temperature sensor
22nd April 2020	Added LCD/pushbutton functionality
23rd April 2020	Adding support for persistent data
24th April 2020	Implemented Generic On/Off Server Model for the FN
25th April 2020	Project Integration & Validation (for Generic On/Off Model)
26th April 2020	Implemented Generic Level Server Model for the FN
28th April 2020	Project Integration & Validation (for Generic Level Model)

(Glossary – BTM: Bluetooth Mesh; FN: Friend Node; LPN: Low Power Node; SM – State Machine)

Low Power Validation

A Bluetooth Mesh (BTM) Friend Node (FN) is not inherently designed to be low-power. This is because the BTM FN would be continuously providing support to the Low-Power Nodes that are connected to the FN via established friendships. Consistent with this viewpoint, and having learnt low-power firmware design principles in class, an attempt has been however made at reducing the power consumed by the device (FN). This attempt involves turning on/off the on-board LCD display which would provide some relaxation in the power budget to the FN although it will not be resource constrained in a practical setting.

A review of the power consumed by the FN can be obtained using the Energy Profiler output available in the Simplicity Studio. The screenshot below shows that the FN consumes a current of approximately 10-11 mA on an average and that implies that the FN is not at all implementing any low-power designs. This is true because the FN does not make use of any sleep levels (e.g. EM1, EM2, etc.) as it is continuously involved in reception of information arriving from the LPNs and while also providing services to the LPNs as necessary.



Five Lessons Learnt (+1)

Some of the best learning taken away from this class and especially this project experience are based on the following top five lessons that come to my mind:

1. During the project development and validation cycle, I significantly learnt about the strong importance of consistency and like-design throughout the system subunits. For example, it was very important that all the Low Power Nodes communicated in the same consistent fashion with the Friend Node such that their state machines would be identical, repetitive and reliable despite the flexibility in duty cycles of the Low Power Nodes.

2. Through this project experience I learnt how projects would grow in the industry and how they can rapidly look different than the original plan with different kinds of modification. Throughout this project experience, I realized how the use of different firmware pieces made me realize the gaps in my assumptions/expectations and the true implementation that was possible within the given time.

3. This project experience taught me the importance of managing/controlling source version. We encountered several instances while working on the project where we had to move back and forth in the firmware versions that we had created and to take away the best source code fragments from the different versions we had to create the working version that we have now.

4. Most importantly, I learnt a great lesson on the necessity and extreme importance of testing at every step of the development cycle. Through continuous testing iterations, we saved a lot of time in project development and that has made me realize how much testing and debugging is important in the industry when it comes to massive project that have innumerable moving and isolated parts that still rely on each other.

5. Through this project experience, I learnt more about the relevance and practical importance of using state machines. Although, we used state machines initially in the class assignments, it was not so much clear until I actively dived into the project development where I realized the importance of state machines for individual exposed services when multiple moving parts were playing an active role in the project execution.

6. I wanted to add this one more lesson that I specially learnt in this class and through this experience is the importance of low-power design. Although our focus was majorly on designing low-power firmware, I realized the important implications it would have on the hardware and how hardware can leverage the applications when they are not CPU intensive and therefore the product battery lifetime would substantially increase/improve.

Final Project Status

The individual final project status for the project development is COMPLETED.

The final project FN development includes a complete list of features that were proposed in the original project proposal – excluding a few. An exhaustive list of the features that have been implemented in the final project submission for my individual portion has been provided below:

- The project firmware that would run on the EFR32 Blue Gecko 13 platform and would provide Bluetooth Mesh Friend Node (Server Model) capabilities.
- The firmware provides support for handling external events arising from the state machine that runs the MCP9808 temperature sensor that provides temperature readings at an interval of every 1 second.
- The firmware allows the device to be provisioned using a GATT device (e.g. smartphone) or the embedded provisioner for which the repository link has been provided in this file.
- The firmware supports simultaneous connections of up to three Low Power Nodes with successful friendship establishment.
- The firmware has source files that run the EFR32BG13 LETIMER and I2C peripherals for interacting with the external temperature sensor and for also providing timestamps for logging.
- The firmware includes a GPIO source file that manages all the GPIO requirements – e.g. running the on-board LEDs and providing support for the LCD display.
- The firmware includes logging support using the log source file.
- The firmware also logs the error cases for BTM event and gecko APIs/commands run in the application source file.
- The firmware provides support for Generic Level Server Model as the FN is a subscriber and utilizes the server model.
- The MCP9808 temperature sensor make use of I2C0 port and operates at both 3.3V/5V power supply. The MCP9808 temperature sensor provides an operational I2C frequency of 400 kHz. The sensor uses its own external event state machine.
- The firmware provides support for storing persistent data. The firmware essentially uses the Generic Level Server Model to acquire level information from the Low-power Nodes for the sensor that are interfaced with those nodes. Upon threshold violations on these individual LPNs, the FN acquires alarm signals from the LPNs and uses an alarm buffer to set bit flags corresponding to each LPN. The alarm buffer would be then stored in the flash memory to provide persistent data support and at the same time would allow the retention of alarm statuses in case of power cycles on the FN.
- The firmware on the FN also allows power-saving features by providing the ability to the user to turn off the LCD display and turn it on back when necessary by interacting with the device using PB1 pushbutton.
- The firmware on the FN also allows a user to override all the alarm signals and refresh the LCD display with no alarm signals. This feature also reflects the new changes in the persistent data (i.e. alarm buffer stored in the flash memory).
- The firmware also provides support for using Generic On/Off Server Model. However, this feature is available as a separate firmware (generic_on_off branch of my repository).

  The firmware **does not provide** full DFU support and this feature has been excluded and mentioned as "**NOT IMPLEMENTED**" in the project validation plan.

  All the above feature have been implemented - except as noted otherwise - and have been included in the firmware created.

*Please see the glossary below.

LPN – Low Power Node
FN – Friend Node
BTM – Bluetooth Mesh Network

Credits: *Please note, the above documentation pertaining to project completion status has been directly borrowed from the work that I have created and submitted to my project repository ReadMe file. I duly credit this section that I had originally created for my GitHub repository.*