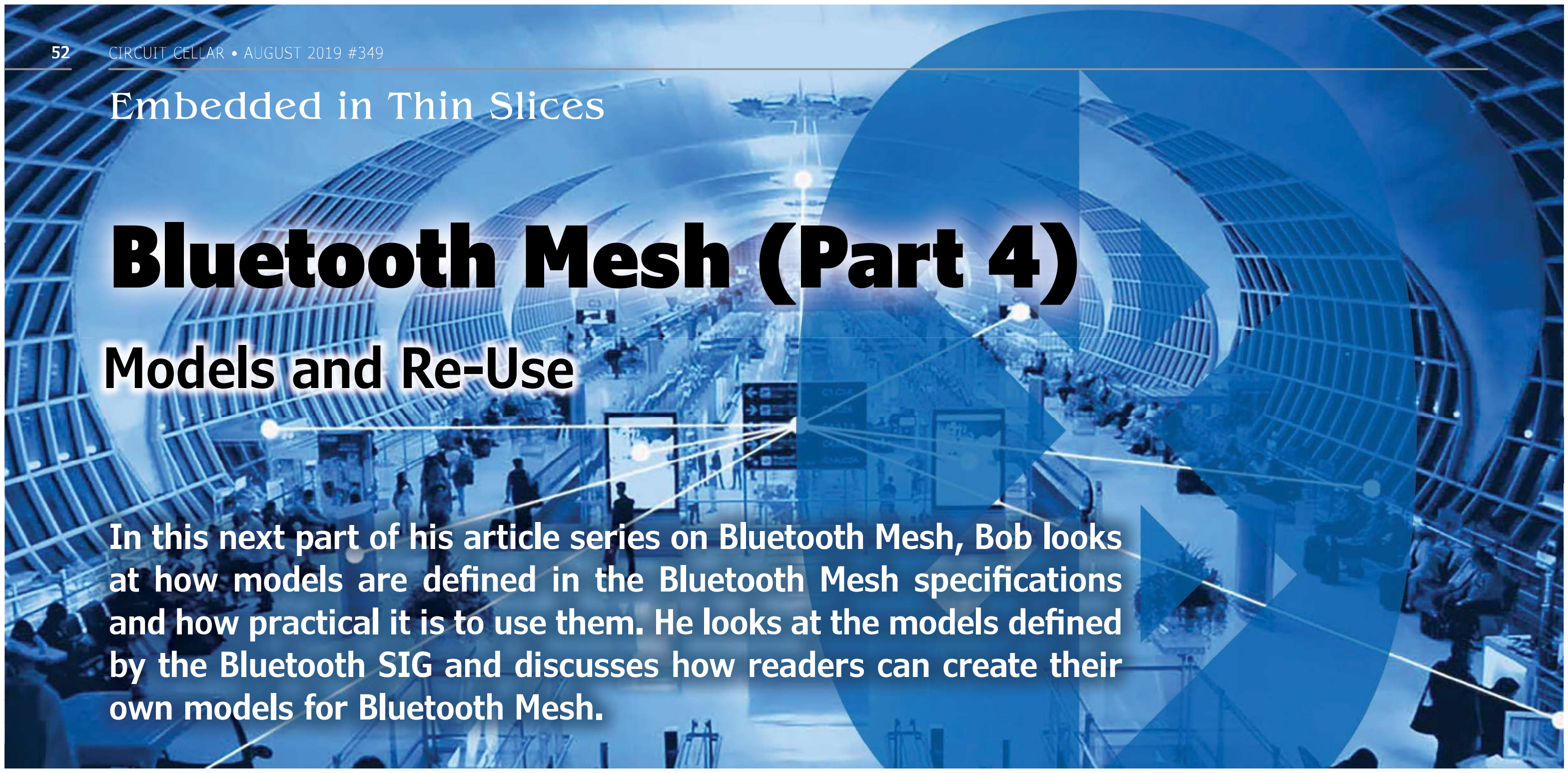
Embedded in Thin Slices



# Bluetooth Mesh (Part 4)

## Models and Re-Use

In this next part of his article series on Bluetooth Mesh, Bob looks at how models are defined in the Bluetooth Mesh specifications and how practical it is to use them. He looks at the models defined by the Bluetooth SIG and discusses how readers can create their own models for Bluetooth Mesh.

By
*Bob Japenga*

Some of you might remember Grady Booch. Grady is well known for his part in developing the Universal Modeling Language (UML) that many of us use to create quality software specifications. I remember Grady from his days when he was promoting software re-use using the programming language Ada. As a software engineering visionary, he saw the re-use of Ada packages as the silver bullet to slay the vampire of software development cost and schedule overruns.

Grady envisioned a world of Ada packages that would revolutionize software development. Perhaps he never read that Fred Brooks, in his book *The Mythical Man-Month*, said that the silver bullet would never be found. (I am sure he did read this classic.) Is anyone out there re-using Ada packages today? Let me know! Perhaps his move to develop a better way of specifying and developing software through UML may have been motivated by the failure of the re-use of Ada packages. Certainly, creating clear and more deterministic specifications—which UML does—has improved our ability to create reliable software on-time and on-budget.

The promise of re-use remains elusive to those of us in the real world of software development. Certainly, some things get re-used, but we are far from Grady's vision. Most re-use takes place in the rich libraries that are provided by our operating systems and our Software Development Kits (SDK).

During the past few months, we've been looking at Bluetooth Mesh in this article series. One of the things that the Bluetooth SIG created in the Bluetooth Mesh specifications is a set of models for communication to a variety of Bluetooth devices. Hundreds of pages of the specifications are devoted to defining how to communicate to lights, sensors, batteries and generic on/off devices. It also includes models for timing and scheduling. This month we will look at what the specification provides and what is actually available from the Bluetooth community to keep us from re-inventing the wheel and to perhaps fulfill a tiny bit of Grady's expansive vision for re-use.

## MODELS IN BLUETOOTH MESH

The Bluetooth SIG created a paradigm in the specification to describe what a device can do on the mesh network. Two of the building blocks in that specification are the concept of the element and the model. An element is that part of the device that has been assigned a unique address at the time of provisioning. A given device on the mesh network has one or more elements each with a unique address. Each element must support one or more models.

Meanwhile, a model defines a collection of functionality and behaviors for the element. There are three categories of models: Server, Client and Controller. A Controller contains both Client and Server functionality as well as control functionality. There are four types of models defined in the specification: the generic model, the sensor model, the lighting model and a model of various timing functions and behaviors. Much like inheritance in object-oriented programming languages, new models can inherit behavior from other models. A model that doesn't inherit from any other model is called a root model.

With all that in mind, each device on the network can have multiple elements in them and each element can have multiple models. For example, imagine a light fixture that has three lamps. One lamp is a simple on/off. Another lamp is dimmable. The third lamp can provide a range of colors as well as being turned on or off and dimmable.

In 1999, the International Standards Organization proposed a standard set of layers of network functionality (**Figure 1**) [1]. The Bluetooth SIG created a slightly different set of layers for looking at Bluetooth Mesh network functionality (**Figure 2**) [2]. The Foundation Model Layer is responsible for implementing the models that are needed to configure and manage the network. The Model Layer is defined by the specifications and is an integral part of all Bluetooth Mesh Networks.

Another concept that is very closely related to the Model is the Scene. A Scene is a collection of states of the device. Using our light fixture example, let's assume that the light fixture is used in a theatrical play. Each scene in the play would have different lamp intensity, different lamps on and different colors. The message structure for this is present in the specification to allow you to create different Scenes.

A Bluetooth Mesh Scene would correspond to a scene in the play. Another example of using the Scene concept would be a device containing a number of sensors. The sensors may be configured for three different modes: one full-power, one half-power and one off. Each of these power levels could be described as a Scene. The specification defines a Scene Server, a Scene Client and a Scene Setup Server, which includes all the commands needed to create Scenes.

## DRILLING DOWN

Bluetooth SIG has done a good job and defined a number of models and scenes that will cover a lot of the devices we will be developing. **Table 1** provides a list of the comprehensive and extensible models that are defined. Bluetooth module vendors can create other models. As implementers, we are free to create other models.

Let's look more in depth at our light fixture with three different types of lamps (**Figure 3**) [3]. Our light fixture has a single Bluetooth LE radio. Each of the separate lamps is called an element. Each element uses one or more different models defined by the Bluetooth Mesh Specification. Each element has a unique address assigned to it during provisioning. Each of the three lights has states, messages and models as defined in the Bluetooth Mesh Model specification.

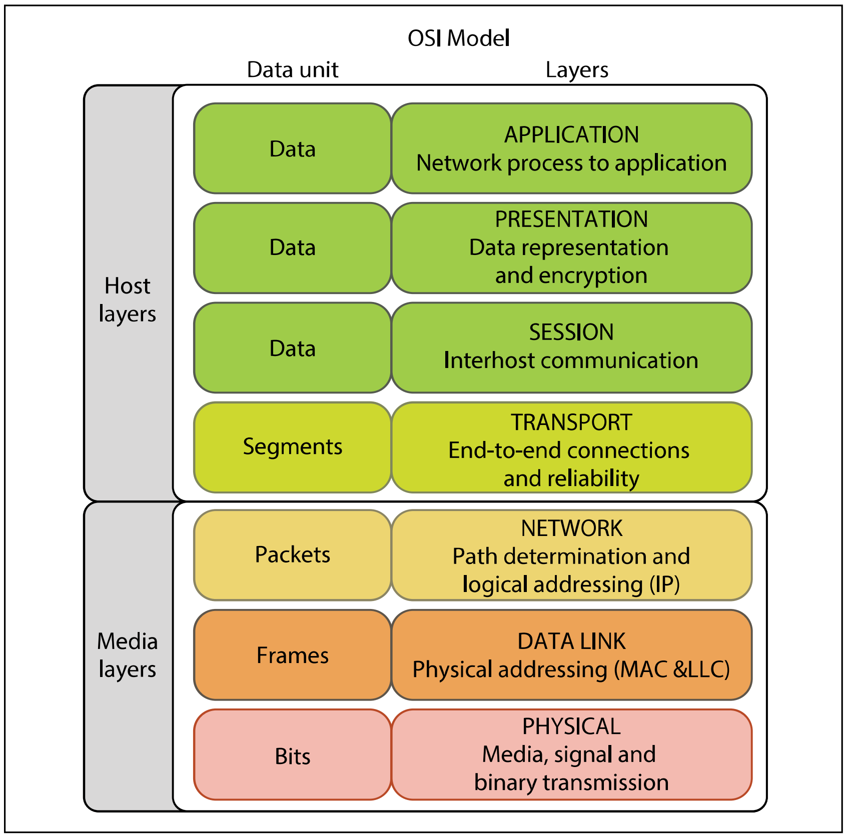

OSI Model
Data unit
Layers
Host layers
Media layers
Data
APPLICATION Network process to application
Data
PRESENTATION Data representation and encryption
Data
SESSION Interhost communication
Segments
TRANSPORT End-to-end connections and reliability
Packets
NETWORK Path determination and logical addressing (IP)
Frames
DATA LINK Physical addressing (MAC &LLC)
Bits
PHYSICAL Media, signal and binary transmission

**FIGURE 1**
ISO created this Open Systems Interconnection (OSI) standard, a set of layers of network functionality.

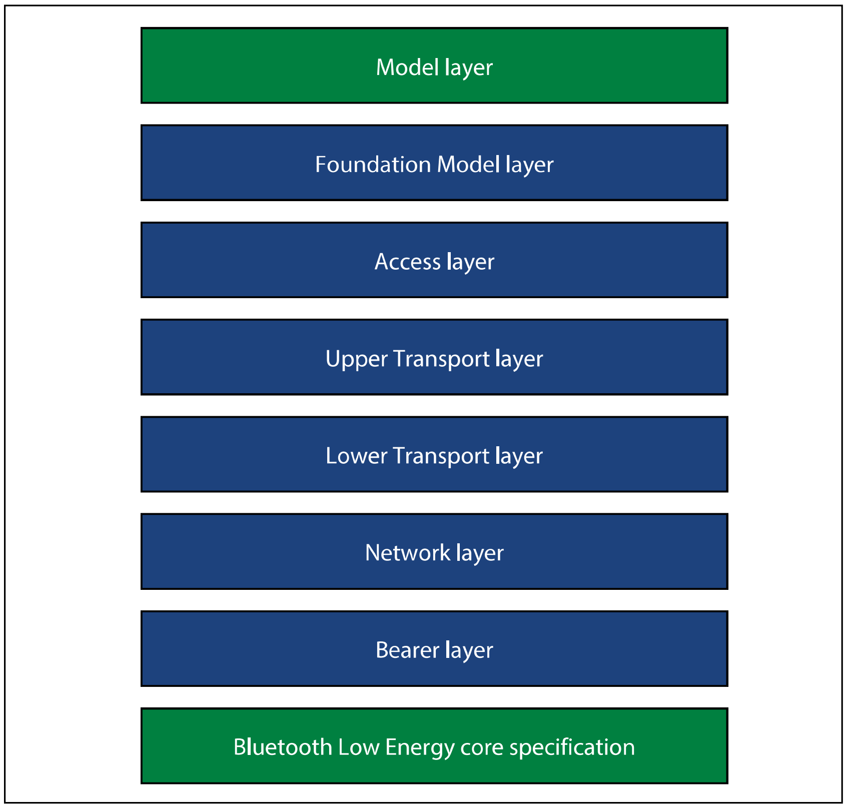


**FIGURE 2**
In contrast to the ISO's set of layers shown in Figure 1, the Bluetooth SIG created a slightly different set of layers for looking at Bluetooth Mesh network functionality.

The functionality of the simple lamp element is defined by the Generic OnOff Server Model. It can be controlled by another device on the mesh network that uses the Generic OnOff Client Model. The functionality of the dimmable light element in the light fixture is defined by the Light Lightness Server model. Lightness is similar to brightness—lightness is perceived reflectance, brightness is perceived luminance. It seems to me that they are controlling brightness. This model inherits functionality from several generic models like the Generic OnPowerup model, the Generic OnOff model and the Generic Level model.

The Light Lightness Model also picks up some unique features like the default settings,

| Model Group | Model Name | Description |
|---|---|---|
| Generic | Generic OnOff Server | Used for responding to the request to turn an element on or off |
| | Generic OnOff Client | Used for turning an element on or off |
| | Generic Level Server | Used for responding to the request to set an element to a certain level |
| | Generic Level Client | Used for setting an element to a certain level |
| | Generic Default Transition Time Server | Used for responding to the request to set how long an element shall take to transition from a present state to a new state |
| | Generic Default Transition Time Client | Used for setting how long an element shall take to transition from a present state to a new state |
| | Generic Power OnOff Server | An extension of the Generic OnOff Server to encapsulate the powering on/off of the element in the node |
| | Generic Power OnOff Setup Server | Used to set up the timing characteristics of power on/off for the element in the node |
| | Generic Power OnOff Client | An extension of the Generic OnOff Client to encapsulate the powering on/off of the element in the node |
| | Generic Power Level Server | An extension of the Generic Level Server to encapsulate the powering on/off of the element in the node |
| | Generic Power Level Setup Server | Used to set up the timing characteristics of power levels for the element in the node |
| | Generic Power Level Client | An extension of the Generic Level Client to encapsulate the powering on/off of the element in the node |
| | Generic Battery Server | Used for sending battery status and battery parameters |
| | Generic Battery Client | Used for requesting battery status and battery parameters |
| | Generic Location Server | Used for responding to the request for latitude, longitude and altitude |
| | Generic Location Setup Server | Used for responding to requests to set latitude, longitude and altitude |
| | Generic Location Client | Used to send or request latitude, longitude and altitude |
| | Generic Admin Property Server | Used to respond to requests for administrator level properties |
| | Generic Manufacturer Property Server | Used to respond to requests for manufacturing level properties |
| | Generic User Property Server | Used to respond to requests for user properties |
| | Generic Client Property Server | Used to obtain properties of the client |
| | Generic Property Client | Used to set and request properties for all levels |
| Sensors | Sensor Server | Used for responding to requests for sensor data |
| | Sensor Setup Server | Used for responding to requests to setup sensors |
| | Sensor Client | Used for setting sensor parameters and obtain sensor readings |

the range of brightness and the last state of brightness. The specification provides all of the messages needed to control a dimmable light. All of the parameters needed to set up this dimmable light are handled by the Light Lightness Setup Server model. The light can be controlled by a device that implements the Light Lightness Client model.

The functionality of the colored light element in our light fixture is also handled via models defined in the specification. Various hues, saturation levels and brightness of the colors of the light are all controllable with the HSL (Hue, Saturation and Lightness) model. Both client and server models are defined for this model.

## WHERE'S THE RE-USE?

Before we look at this question, let me provide our history in using these models. Last year, we were involved with a company that was implementing a very simple lighting device using Bluetooth Mesh. The Bluetooth vendor (who will remain anonymous)

| Model Group | Model Name | Description |
|---|---|---|
| Times, Schedulers and Scenes | Time Server | Note: Mesh uses International Atomic Time, which is different than UTC. The server delivers time, time zone, time authority information to Time clients. |
| | Time Setup Server | Used for setting up time, time zone, known uncertainty, time authority and other time related parameters. |
| | Time Client | Used for obtaining time, time zone and other time related parameters |
| | Scene Server | Scenes are used to group setup parameters to create a particular scene (e.g., Lighting controls that set a particular mood or ambiance). The server used for responding to requests for particular scenes |
| | Scene Setup Server | Used for setting the parameters for particular scenes |
| | Scene Client | Used for setting and requesting the particular scenes |
| | Scheduler Server | Used to schedule changes to the states and properties of an element at a particular time and date |
| | Scheduler Setup Server | Used to set up schedules |
| Lighting | Light Lightness Server | Used to respond to requests for the level of brightness |
| | Light Lightness Setup Server | Used to set up the Lightness settings |
| | Light Lightness Client | Used to control the level of brightness in a server |
| | Light CTL Server | Used to respond to requests for the color temperature |
| | Light CTL Setup Server | Used to set up the color temperature settings |
| | Light CTL Client | Used to control the color temperature in a server |
| | Light CTL Temperature Server | Used to respond to setting of the color temperature of tunable white light |
| | Light HSL Server | Used to respond to requests for the Hue, Saturation and Lightness settings |
| | Light HSL Setup Server | Used to setup the Hue, Saturation, and Lightness settings |
| | Light HSL Client | Used to control the Hue, Saturation and Lightness in a server |
| | Light HSL Hue Server | Used to respond to requests for the Hue settings |
| | Light HSL Saturation Server | Used to respond to requests for the Saturation settings |
| | Light xyL Server | Used to respond to requests for the color |
| | Light xyL Setup Server | Used to set up the color settings |
| | Light xyL Client | Used to control the color in a server |
| | Light LC Server | Used to respond to requests for a Lighting Controller |
| | Light LC Setup Server | Used to set up the Lighting Controller |
| | Light LC Client | Used to control the Lighting Controller |

**TABLE 1**
The Bluetooth SIG has defined a number of models and scenes that will cover a lot of devices. This table provides a list of the comprehensive and extensible models that are defined.

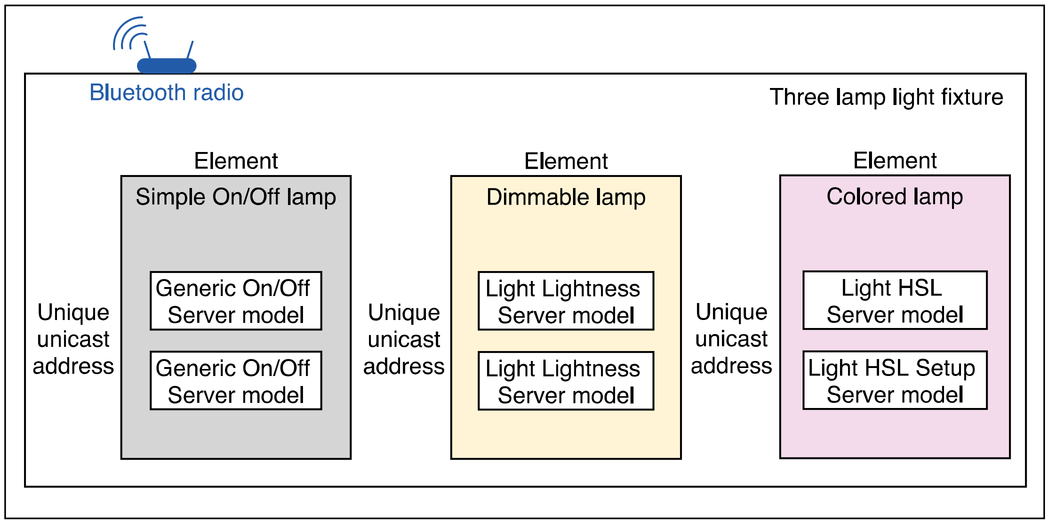


**FIGURE 3**
This example light fixture has a single Bluetooth Low Energy (BLE) radio. Each of the separate lamps is called an element. Each element uses one or more different models defined by the Bluetooth Mesh Specification. Each element has a unique address assigned to it during provisioning. And each of the three lights has states, messages and models as defined in the Bluetooth Mesh Model specification.

agreed to implement the Client side of the project and help us with the Server side of the project. The project started with a pilot program to demonstrate feasibility to a very large customer. After several months of glacial progress, we initially thought that the problem was that the developers did not have enough time to put into the project. However, when our deadline was approaching and the work was still not getting done, it became very clear to us that the problem was that the developers had to develop all of the lighting models themselves.

Very few of the models needed were implemented in the vendor's Software Development Kit (SDK). None of the lighting models were implemented—although there was a simple lighting example that did not use any of the lighting models! In fact, only 12 of the 51 models were implemented in the SDK. Most of the models needed for our implementation needed to be developed. So much for re-use!

The Bluetooth SIG site lists 139 products that have Bluetooth Mesh networking capability and have successfully completed the Bluetooth Qualification Process. A few of the vendors have implemented all of the models. Some, like the one we used, implemented a small subset. One vendor is clear on their webpage that they only support some of the models—foundation, generic and some of the lighting models—but not the sensor model nor the timing models. This is confirmed in their SDK. Others are not clear and you have to drill down into the SDK to find out what is provided. If we want to minimize development time and cost, the bottom line for us as developers is to find a vendor that provides the models that we need.

## CONCLUSION

Re-use and interoperability are wonderful in concept but are extremely difficult to implement. Re-use takes a different form today than was envisioned by Grady Booch. Today, re-use happens when we can draw upon rich libraries provided by the vendors for the chips we are using. In an ideal world, I would love to have a Bluetooth Mesh network with lights and sensors from a variety of suppliers and have them all be interoperable. I would love to be able to create my own light fixture and have the Bluetooth chip supplier's SDK provide most of the code. The Bluetooth SIG has laid out a paradigm for that to happen. Right now, just two years after the specification was released, it may be too early to tell. But two years is a long time in the world of IoT.

In the past few articles I've introduced you to Bluetooth Mesh. Clearly, we have explored it in thin slices. Although there is much more that I could write about, I want to move on next time to explore the very, very low bandwidth wireless data communication technology known as LoRa.

***ABOUT THE AUTHOR***

Bob Japenga has been designing embedded systems since 1973. In 1988, along with his best friend, he started MicroTools, which specializes in creating a variety of real-time embedded systems. MicroTools has a combined embedded systems experience base of more than 200 years. They love to tackle impossible problems together. Bob has been awarded 11 patents in many areas of embedded systems and motion control. You can reach him at rjapenga@microtoolsinc.com.

**For detailed article references and additional resources go to:**
**www.circuitcellar.com/article-materials**
**References [1] through [3] as marked in the article can be found there.**