Chapter 7A: Bluetooth Mesh Topology and Client Applications

This chapter covers the basics of the Bluetooth mesh network topology.

7A.1 Overview

Traditional Bluetooth LE devices use point-to-point communication. That is, each pair of devices send data back and forth to each other. Each of these connections has a GAP Central and a GAP Peripheral.

In contrast, in a mesh network every device in the mesh can communicate (either directly or indirectly) with every other device in the network. Some devices in the network can relay messages that they receive so that the overall communication range is extended beyond the radio range of each individual device. In theory, the range of a mesh network is unlimited as long as you have at least one relay device within range of every device in the network.

In Bluetooth Mesh, messages are sent using advertising packets. That is, no connections are made. Rather, data is broadcast by a sending device using advertising packets which can be received by any devices that are in range of the sender.

Devices in a mesh network are called "nodes". Devices that are not part of a mesh network (yet) are called "unprovisioned devices". The process of provisioning a node will also be covered later.

A mesh network can have one or more subnets that enable isolation of related groups of nodes. A subnet is a group of nodes that can communicate with each other at the network layer because they share a network key. The difference between a network and a subnet is that a node may belong to more than one subnet by having more than one network key.

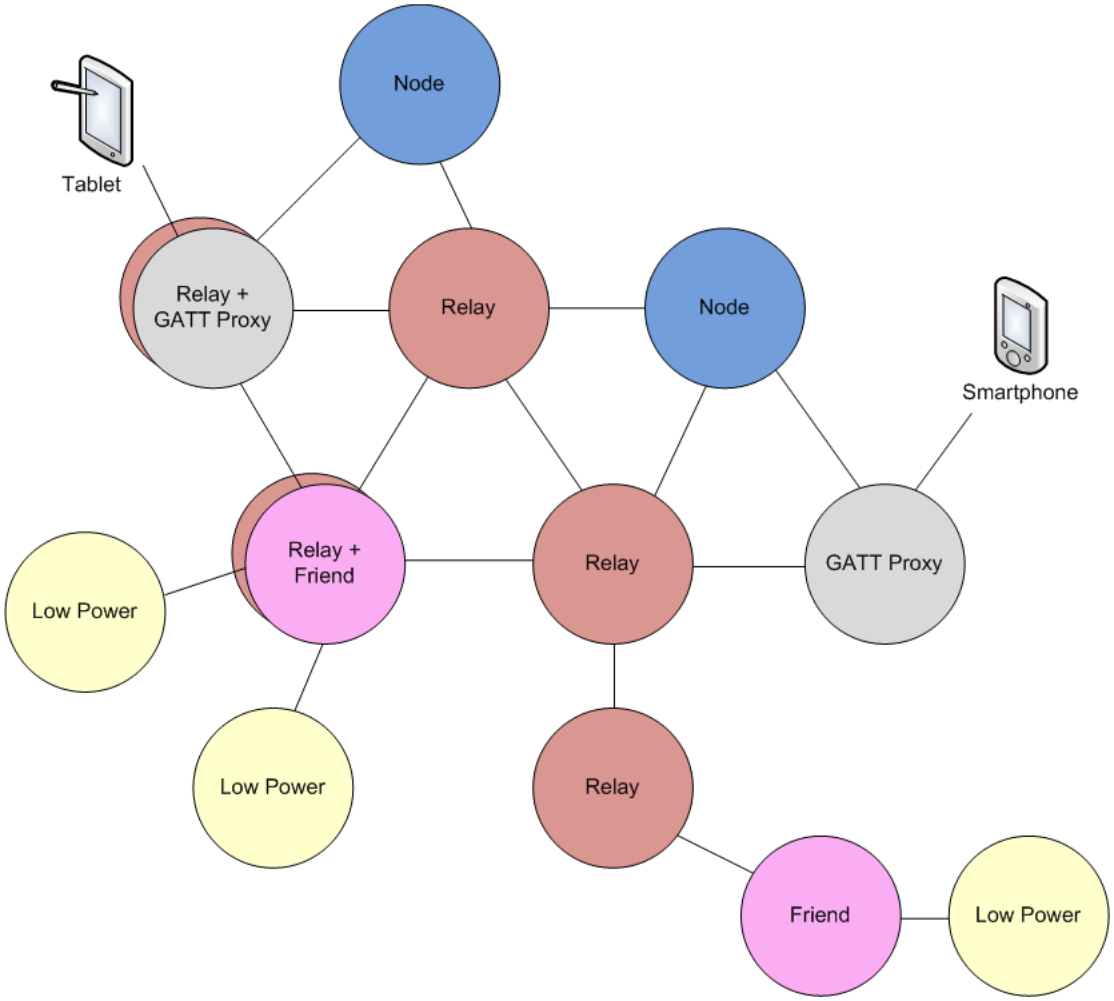
7A.2 Mesh Specs

Before going into more detail, it is worth noting that the Bluetooth SIG provides three specifications that contain every detail of the mesh protocol. These are:

1. Mesh Profile – defines fundamental requirements for mesh networking
2. Mesh Model Specification – defines models which are used to define basic functionality of nodes in a mesh network
3. Mesh Device Properties – defines device properties required for the mesh model spec

7A.3 Nodes

The following figure shows an example mesh network topology. Each of the types of node will be discussed in detail in the following sections. It is suggested that you refer to this figure while reading the descriptions.



Each node in a mesh network can send and receive messages. Each node may also implement one or more of the following features depending on its capabilities:

1. Relay
2. GATT Proxy
3. Friend
4. Low Power

Relay, GATT Proxy and Friend features can all be implemented on the same node. Typically, it doesn't make sense for a Low Power node to implement any of the other features as you will see in a minute.

7A.3.1 Standard Node

The standard node functionality involves sending and receiving mesh messages. Every node in the network must be able to act as a standard node.

Message Caching

Each node must maintain a message cache containing all recently received messages. If a message is received more than once, it is immediately discarded. In this way, if a message is relayed by multiple nodes to a final destination, the destination only acts on the message one time.

7A.3.2 Relay Node

Relay nodes can receive a message for the network and then retransmit it to other devices in range. This is the method by which mesh networks can cover larger distances than the range of any single device. For a network to operate, every node must be within range of at least one relay so that its messages can be forwarded on to nodes that it cannot directly communicate with.

It is common for all except low power nodes to implement a relay feature in order to maximize the possible paths through a mesh network.

Due to the message cashing described above, a relay node will only relay a given message one time.

TTL

Each message has a field called the Time To Live (TTL). This is used to determine how many times a given message will be retransmitted. By understanding the basic topology of a mesh network, the TTL can be used to prevent messages from being retransmitted too many times. This allows the mesh network to be more efficient.

In fact, there are heartbeat messages sent periodically which include, among other things, information that allow receiving nodes to determine how many hops away the sender is. Networks can use this information to adapt TTL settings to optimize the network.

Security

A relay node only decodes enough of the message to decide what to do with it. For example, it decodes the addresses for the message but not the payload if it is not intended for that node. In fact, due to the security architecture, the relay node cannot decode the payload for any messages that are not from the same network application (e.g. lighting). Security will be discussed in detail later.

7A.3.3 GATT Proxy Node

Many existing BLE devices support traditional BLE GATT communication but not mesh communication. Most smartphones and tablets fall into this category. Since you may want to interact with a mesh network from one of those devices, the GATT proxy was created. A GATT proxy node has both a mesh interface and a GATT interface. The GATT interface is used to communicate with BLE devices that don't possess a mesh stack and then relay those messages to/from the mesh network. That is, the GATT proxy acts as a bridge between the mesh network and the traditional BLE GATT device.

7A.3.4 Friend and Low Power Nodes

Friend and Low Power Nodes are used to optimize power consumption for constrained devices. Devices that are power constrained (e.g. a battery powered device) are designated as low power nodes. Every low power node in the network must be associated with exactly one friend node. Friend nodes are devices which are not power constrained (e.g. a device plugged into AC power) that support 1 or more low power nodes depending on its capabilities (e.g. available RAM).

When a low power node is added to a mesh network it broadcasts a request for a friend. Each friend in range that can handle a new low power node replies and the low power node selects the best friend based on how many messages the friend can store; the RSSI and the timing accuracy.

Once the relationship is established, the friend node will receive and store messages for any low power nodes that it is associated with. The low power node will periodically ask the friend node for any messages that the friend has stored for it. In this way, the low power node does not need to listen continuously for mesh packets. Instead, it can be in a low power mode most of the time and can wake up only periodically for a very short time.

For example, consider a battery powered mesh connected thermostat. It will measure the actual temperature and may send a mesh message with the temperature once per minute. This can be done with very low power consumption since the device can be sleeping all the time except for a short period each minute to send the value. However, it must also be possible to change the set point of the thermostat. In this case, instead of sending messages, the thermostat must be listening for messages. If it listens constantly for messages the power consumption will be unacceptably high, but if it only listens occasionally for messages it will likely miss messages. By making the thermostat a low power node we get the best of both worlds - it can send messages once a minute and receive any stored messages regarding the set point from its friend node. No messages are missed even though the thermostat is awake only a very small percentage of the time.

7A.4 Provisioning and Configuration/Management

To get a new device up and running on a Bluetooth Mesh network, it must be provisioned and configured. These are often thought of as a single step, but they are unique processes with different protocols. These will each be described separately below.

7A.4.1 Provisioning

Provisioning is the process by which a device is made a member of the mesh network and becomes a node. To be a node on a mesh network, a device needs to have the network key (and other associated network security information like the IV index) and it needs to have a unicast address assigned to its primary element. Provisioning can be done using either a GATT connection (PB-GATT) or an advertising channel (PB-ADV) as the bearer. (PB = Provisioning Bearer).

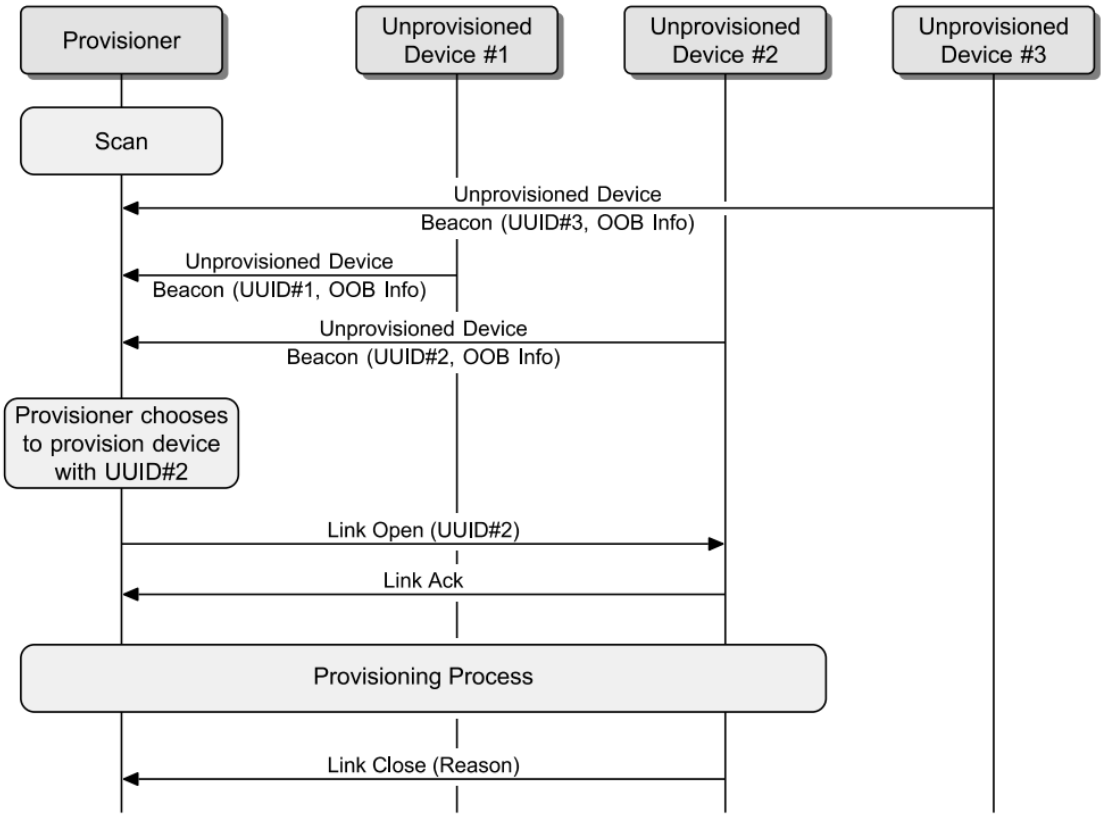
Provisioning is most commonly done using an application on a smartphone or a tablet. Note that smartphones currently do not support provisioning over an advertising channel, so from a practical standpoint, all devices should support provisioning over GATT. The Bluetooth Mesh spec strongly recommends that unprovisioned devices support both.

Beaconing

Any unprovisioned device will indicate its availability to be provisioned by sending out advertising packets of the type "Mesh Beacon".

Link Establishment

A provisioner will scan for unprovisioned devices and will choose (usually via input from the user) which device to provision. The provisioner sends a Link Open message to the device to be provisioned which will in turn respond with a Link ACK message. Once provisioning completes, the provisioner sends a Link Close message. These steps are illustrated in the figure below.



(This figure is taken from the Bluetooth Mesh Profile Specification)

The remaining steps detailed below occur within the box labeled "Provisioning Process" in the figure above.

Invitation

The provisioner sends an invitation to the device being provisioned in the form of a provisioning invite protocol data unit (PDU). The device being provisioned responds with information about itself in the form of a provisioning capabilities PDU.

Exchanging Public Keys

The provisioner and the device to be provisioned exchange public keys either directly or using an out-of-band (OOB) method.

Authentication

Authentication is performed using an OOB method that depends on the capabilities of the device being provisioned. For example, if the device to be provisioned has some output mechanism, it creates a random number and indicates that number to the user (e.g. it may flash an LED a random number of times, beep a random number of times, or show the number on a display). The user then enters that number into the provisioner.

If the device has some input mechanism, then the provisioner creates a random number and presents it to the user. The user then inputs that number on the device (e.g. by pressing a button the specified number of times or entering the number using a keypad).

Either way, once the random number has been generated on one side and entered on the other, a cryptographic exchange happens between the two devices using that random number.

Distribution of Provisioning Data

Once authentication is done, a session key is derived by each device from its private key and the public key from the other device. The session key is used to secure subsequent distribution of the data needed to complete provisioning. Once provisioning is completed, the provisioned device has the network's key (NetKey), a security parameter called the IV index, and its Unicast address which was allocated by the provisioner. The device is now a node and is a part of the network. The provisioner then sends a Link Close message as described previously.

7A.4.2 Configuration/Management

Once provisioning is done, the same smartphone or tablet (i.e. the provisioner) then uses the mesh network to configure the new node. This includes distribution of application keys, assigning group addresses to models, etc.

Note that smartphones currently do not support Bluetooth mesh directly so at least one device should be configured as a GATT Proxy to allow a smartphone to do configuration once provisioning is done. The only alternative currently is to have a gateway on the mesh network that allows the smartphone to access the mesh network indirectly.

7A.5 Client Applications

There are several applications that can be used for provisioning, configuration, and communication for mesh networks. There are:

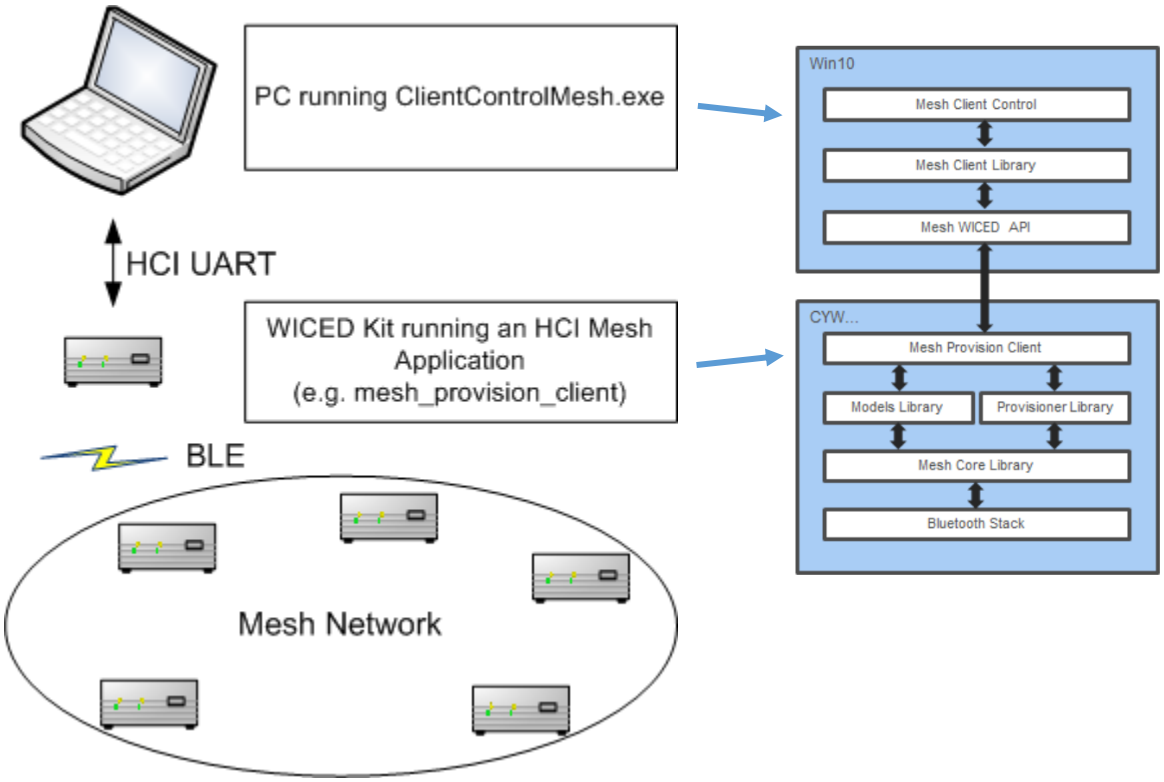
- Two Windows applications (Client Control Mesh and Mesh Client) written by Cypress
- One Android application written by Cypress
- One iOS application written by Cypress
- 3rd party mesh applications such as the LedVance Sylvania Smart Home Android app

7A.5.1 Client Control Mesh (Windows)

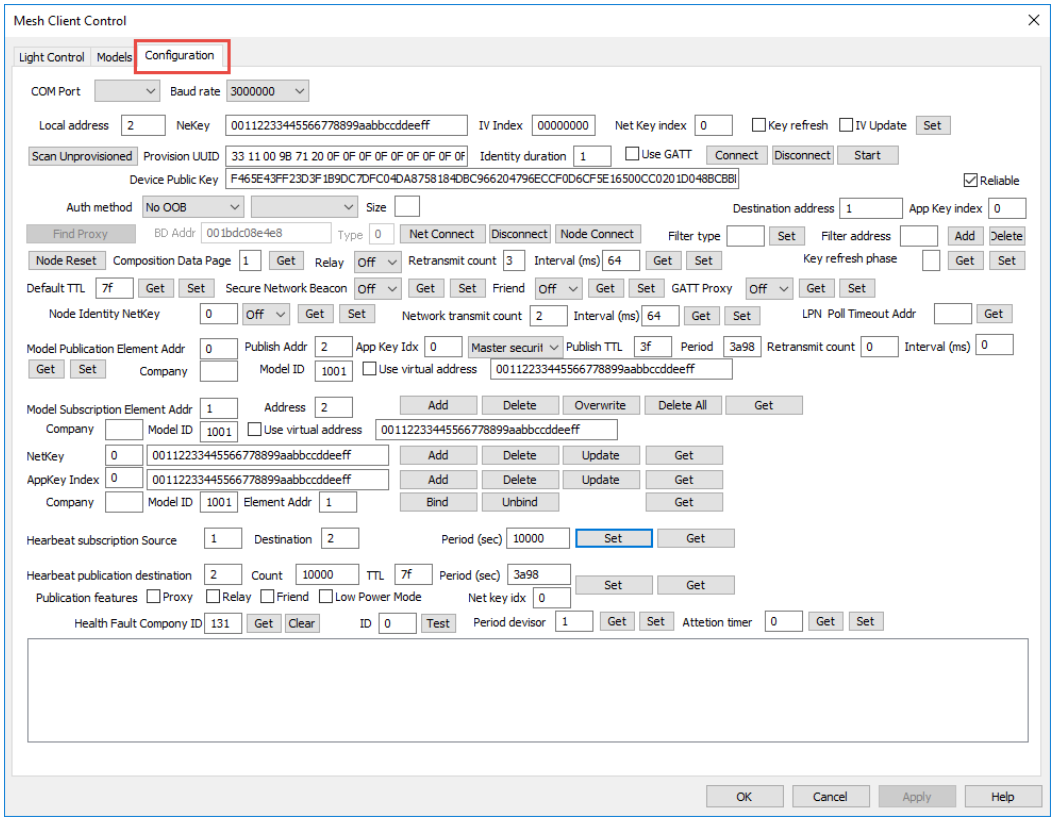
This application is an extension of the Client Control application that can be used for other WICED Bluetooth applications. It is located at:

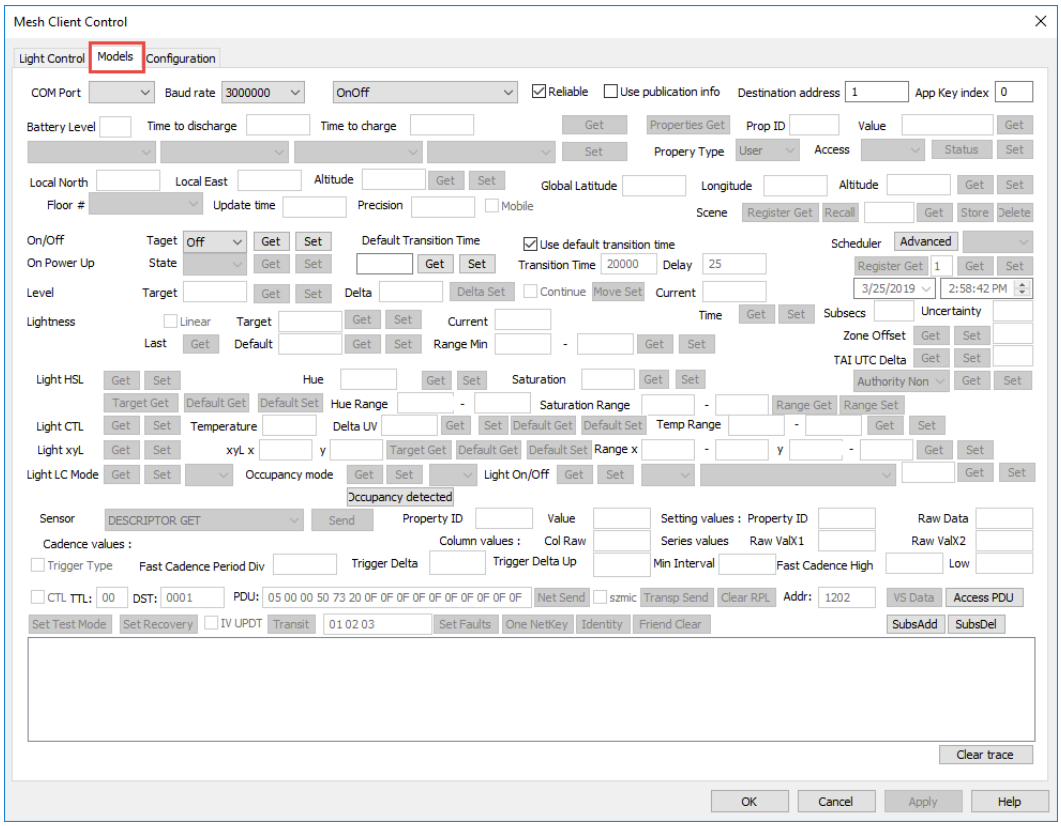
> <Install Folder>/ModusToolbox_1.1/libraries/bt_sdk-1.1/components/BT-SDK/common/apps/snip/mesh/ClientControl/Release/ClientControlMesh.exe

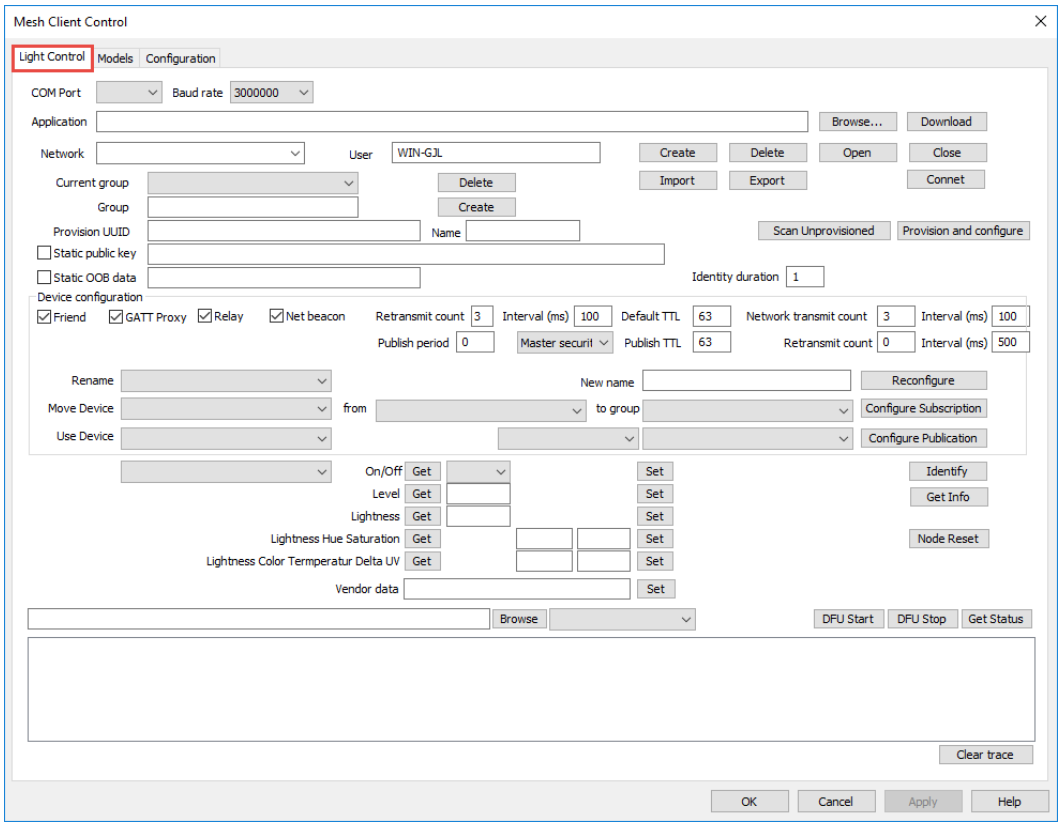
To use it, you must first program a kit with an application that does mesh operations such as provisioning, OnOff client, etc. This kit will act as an interface from the PC to the mesh devices in the network. For example, the mesh_provision_client application supports a provisioning client, OnOff client, level client, light lightness client, etc. That kit will receive HCI commands from the PC over UART and in turn sends the appropriate messages to the mesh network.



The application has tabs for Configuration (provisioning, etc.), Models (for interacting with any model), and Light Control (specific for lighting applications). It allows very low-level interaction with the mesh network as you can see in the screenshots below.

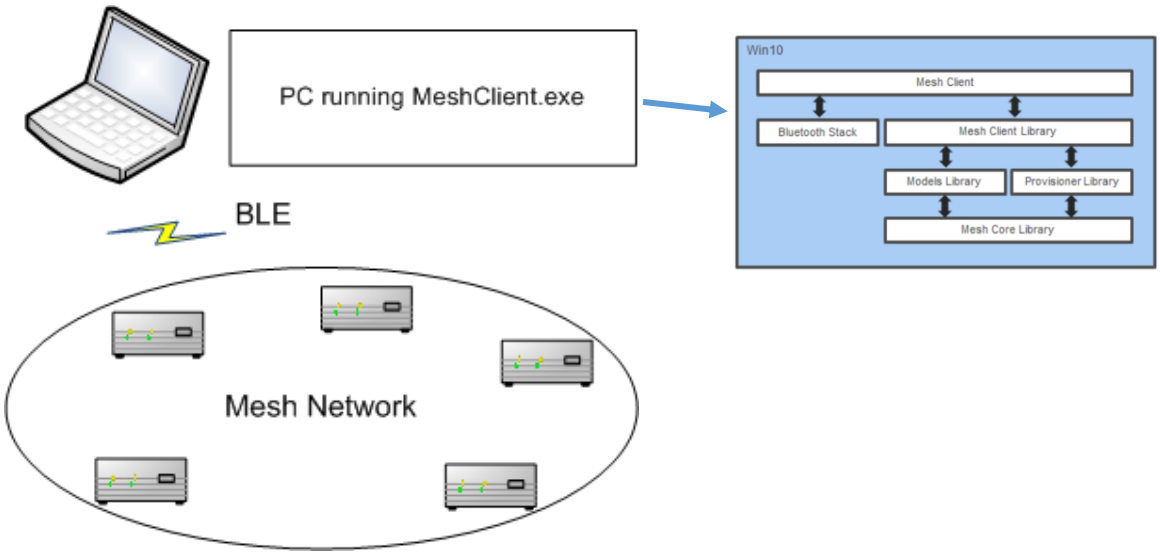



Mesh Client Control
Light Control
Models
Configuration
COM Port
Baud rate
3000000
OnOff
Reliable
Use publication info
Destination address
App Key index
Battery Level
Time to discharge
Time to charge
Prop ID
Value
Propery Type
User
Access
Status
Local North
Local East
Altitude
Global Latitude
Longitude
Floor #
Update time
Precision
Mobile
Scene
Register Get
Recall
Store
Delete
On/Off
Taget
Off
On Power Up
State
Default Transition Time
Use default transition time
Transition Time
20000
Delay
25
Scheduler
Advanced
Level
Target
Delta
Delta Set
Continue
Move Set
Current
3/25/2019
2:58:42 PM
Lightness
Linear
Time
Subsecs
Uncertainty
Last
Default
Range Min
Zone Offset
TAI UTC Delta
Light HSL
Hue
Saturation
Authority Non
Target Get
Default Get
Default Set
Hue Range
Saturation Range
Range Get
Range Set
Light CTL
Temperature
Delta UV
Temp Range
Light xyL
xyL x
y
Range x
Light LC Mode
Occupancy mode
Light On/Off
Occupancy detected
Sensor
DESCRIPTOR GET
Send
Property ID
Setting values :
Raw Data
Cadence values :
Column values :
Col Raw
Series values
Raw ValX1
Raw ValX2
Trigger Type
Fast Cadence Period Div
Trigger Delta
Trigger Delta Up
Min Interval
Fast Cadence High
Low
CTL TTL:
00
DST:
0001
PDU:
05 00 00 50 73 20 0F 0F 0F 0F 0F 0F 0F 0F 0F
Net Send
szmic
Transp Send
Clear RPL
Addr:
1202
VS Data
Access PDU
Set Test Mode
Set Recovery
IV UPDT
Transit
01 02 03
Set Faults
One NetKey
Identity
Friend Clear
SubsAdd
SubsDel
Clear trace
OK
Cancel
Apply
Help


Mesh Client Control
Light Control
Models
Configuration
COM Port
Baud rate
3000000
Application
Browse...
Download
Network
User
WIN-GJL
Create
Delete
Open
Close
Import
Export
Connet
Current group
Group
Provision UUID
Name
Scan Unprovisioned
Provision and configure
Static public key
Static OOB data
Identity duration
1
Device configuration
Friend
GATT Proxy
Relay
Net beacon
Retransmit count
Interval (ms)
Default TTL
Network transmit count
Publish period
Master securit
Publish TTL
Rename
New name
Reconfigure
Move Device
from
to group
Configure Subscription
Use Device
Configure Publication
On/Off
Level
Lightness
Identify
Get Info
Lightness Hue Saturation
Lightness Color Termperatur Delta UV
Node Reset
Vendor data
Browse
DFU Start
DFU Stop
Get Status
Clear trace
OK
Cancel
Apply
Help

7A.5.2 Mesh Client (Windows)

This application communicates with the specified mesh network directly using the BLE radio of the computer.



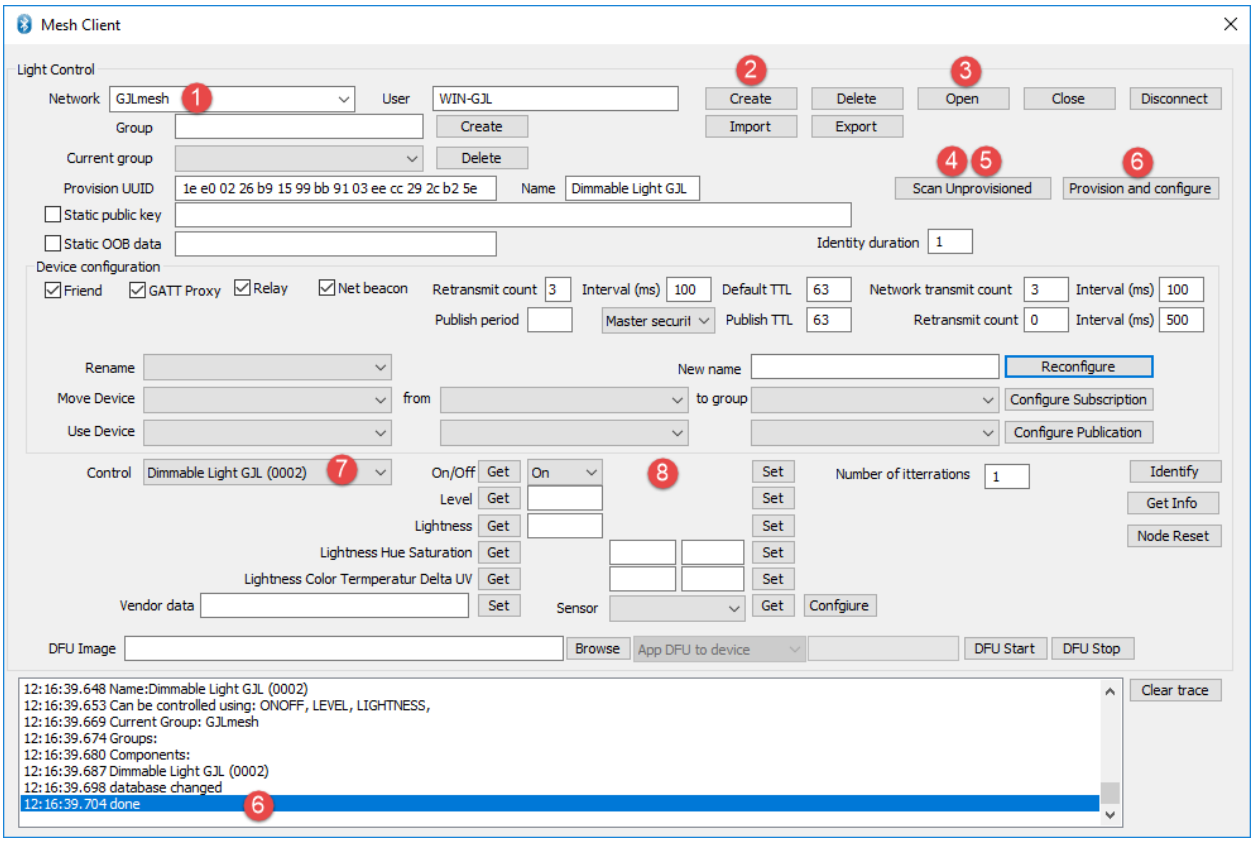
Note that support for BLE was added in Windows 10 so you can't use this with earlier versions of Windows. It can create mesh networks, provision devices and control lighting devices. It is in the SDK at:

> <Install Folder>/ModusToolbox_1.1/libraries/bt_sdk-1.1/components/BT-SDK/common/apps/snip/mesh/peerapps/Windows/MeshClient/Release/x86/MeshClient.exe

Note: To get the latest version, you should refer to the BT example repository located on GitHub. The path to the file in the repository is:

> https://github.com/cypresssemiconductorco/Code-Examples-BT-SDK-for-ModusToolbox/blob/master/Mesh-Peer-Apps/Windows.zip

Unzip the file and go to Windows/MeshClient/Release/x86 to find MeshClient.exe.



The basic flow for using the application is:

1. Enter a name for your network
2. Click *Create*
3. Click *Open*
4. Click *Scan Unprovisioned*
5. Wait until your device appears in the list and click *Stop Scanning*
    a. If there are multiple unprovisioned devices you may need to stop and restart multiple times until you see the device you are looking for.
6. Click *Provision and configure*
    a. This step will take a few seconds – wait until it is complete before continuing.
7. Select your device in the *Control* dropdown.
8. Use *On/Off Get*, *On/Off Set*, *Level Get*, *Level Set*, etc. to control your device.

7A.5.3 Mesh Lighting App (Android)

The Cypress Android app is provided with the BTSDK. Source code is available in the SDK for those who want an example to create their own custom Android mesh app.

The app communicates with the mesh network using the device's BLE capabilities. Since smartphones don't (yet) have mesh capability, the app uses GATT connections for provisioning and relies on the presence of a GATT proxy for mesh configuration and communication.

The app can create mesh networks, provision/configure devices and can control lighting devices. The installable file is located at:

> <Install Folder>/ModusToolbox_1.1/libraries/bt_sdk-1.1/components/BT-SDK/common/apps/snip/mesh/peerapps/Android/src/bin/MeshLightingController.apk

Note: To get the latest version, you should refer to the BT example repository located on GitHub. The path to the file in the repository is:

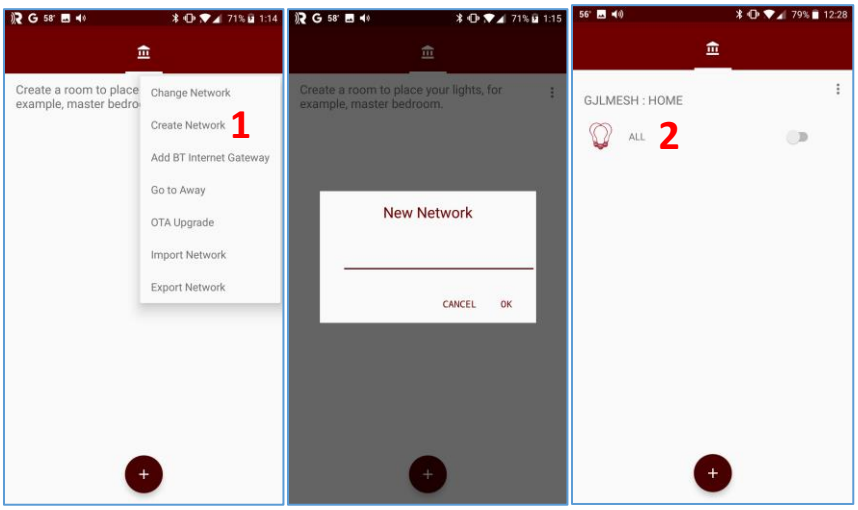
> https://github.com/cypresssemiconductorco/Code-Examples-BT-SDK-for-ModusToolbox/blob/master/Mesh-Peer-Apps/Android.zip

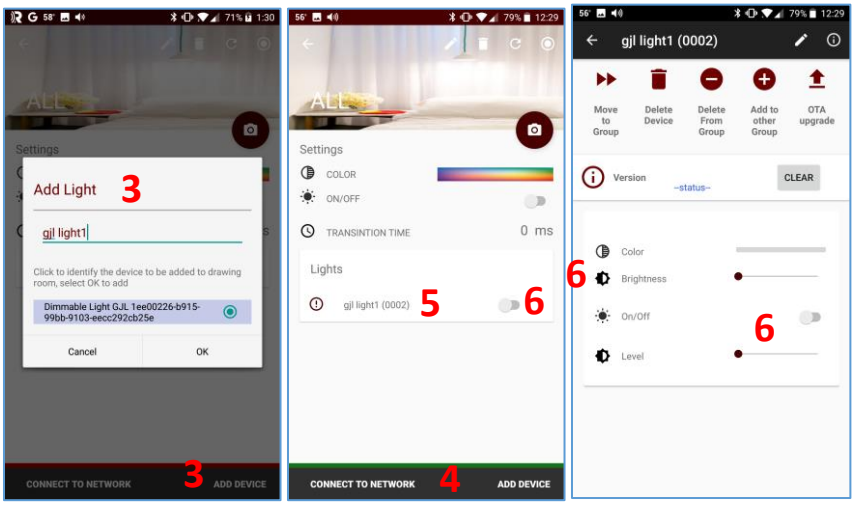
Unzip the file and go to Andoird/src/bin to find MeshLightingController.apk.

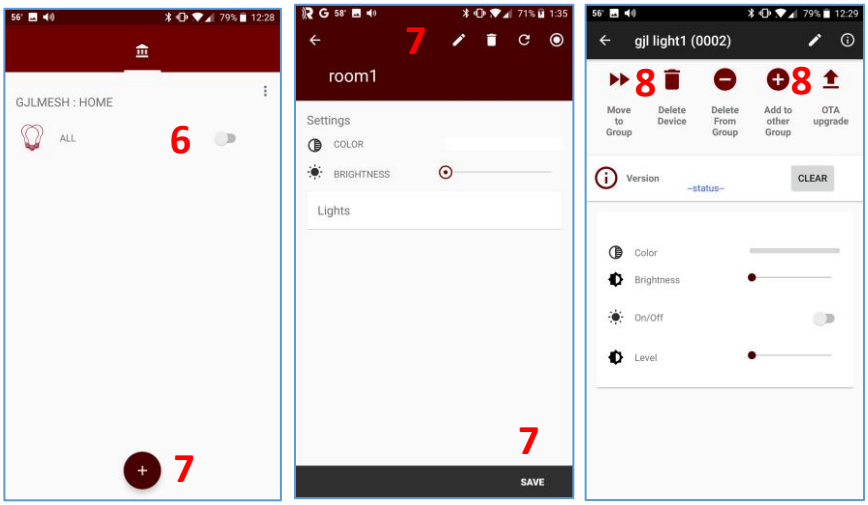
Since the app is not (yet) in the Android Play Store, it is necessary to install it manually by dragging the .apk file onto the phone's filesystem and then executing it to install the app. You will need to allow installation of 3rd party applications for this to work.

The basic flow for using the application is:

1. Create a network
2. Select a group (ALL is created by default)
3. Add a device to the group
    a. This will take a few seconds. Wait until it completes before proceeding
4. Connect to the network if it doesn't happen automatically (bar will be green when connected)
5. Select the device
6. Control the device
    a. Note: you can control all devices simultaneously at the group level or individually at the device level
7. Optional: Add additional Groups (i.e. Rooms)
8. Optional: Move or Add devices to other Groups


Create a room to place
example, master bedro
Change Network
Create Network
1
Add BT Internet Gateway
Go to Away
OTA Upgrade
Import Network
Export Network
Create a room to place your lights, for example, master bedroom.
New Network
CANCEL
OK
GJLMESH : HOME
ALL
2


Settings
Add Light
3
gjl light1
Click to identify the device to be added to drawing room, select OK to add
Dimmable Light GJL 1ee00226-b915-99bb-9103-eecc292cb25e
Cancel
OK
CONNECT TO NETWORK
3
ADD DEVICE
Settings
COLOR
ON/OFF
TRANSINTION TIME
0 ms
Lights
gjl light1 (0002)
5
6
CONNECT TO NETWORK
4
ADD DEVICE
gjl light1 (0002)
Move to Group
Delete Device
Delete From Group
Add to other Group
OTA upgrade
Version
--status--
CLEAR
Color
Brightness
On/Off
Level
6
6
6


GJLMESH : HOME
ALL
6
7
7
room1
Settings
COLOR
BRIGHTNESS
Lights
7
SAVE
gjl light1 (0002)
8
8
Move to Group
Delete Device
Delete From Group
Add to other Group
OTA upgrade
Version
--status--
CLEAR
Color
Brightness
On/Off
Level

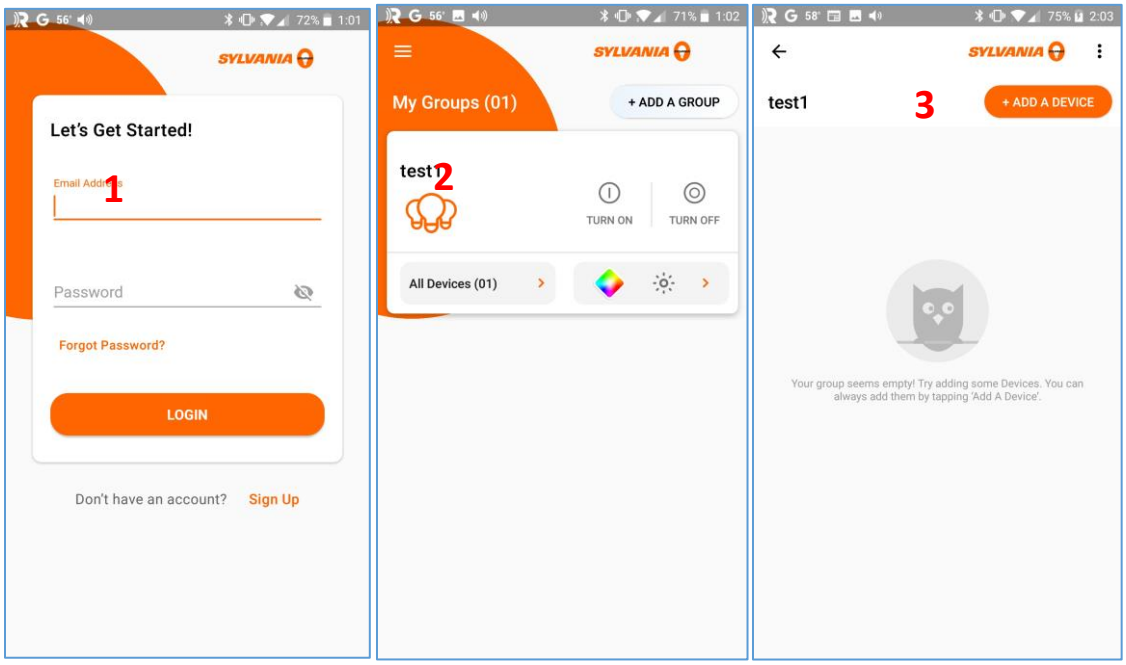
7A.5.4 Sylvania Smart Home App by LedVance (Android)

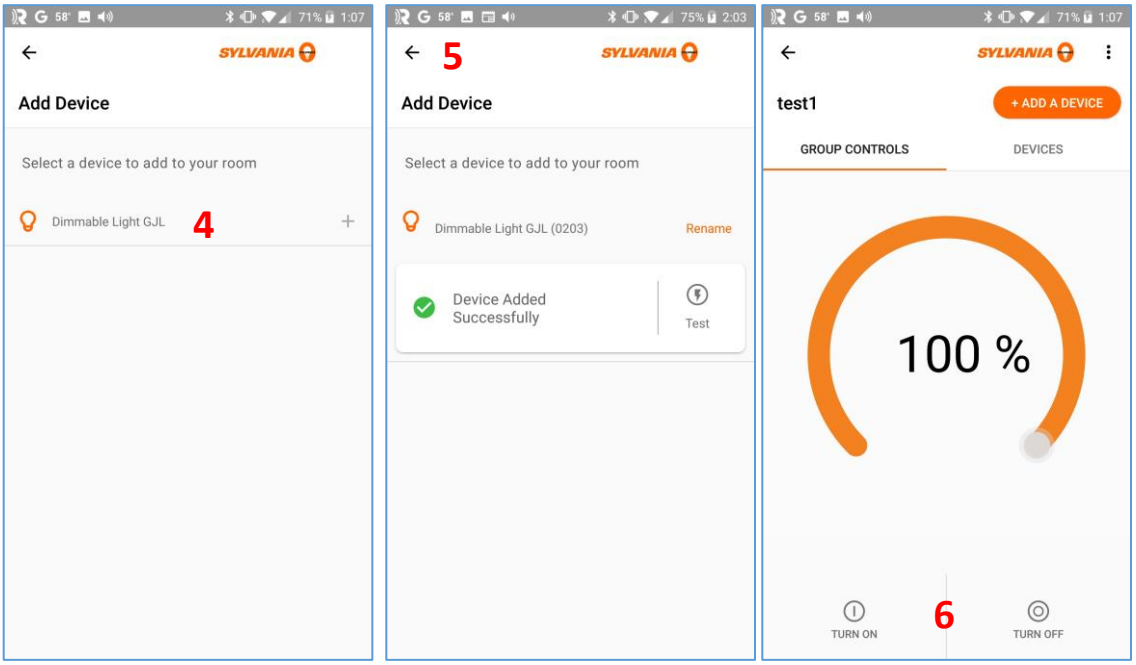
The Sylvania Smart Home application created by LedVance is available for Android from the Google Play Store. It can create mesh networks, provision devices and can control lighting devices. Note: there is a version of the app for iOS, but it is based on the Apple HomeKit solution, so it won't work with our starter applications.

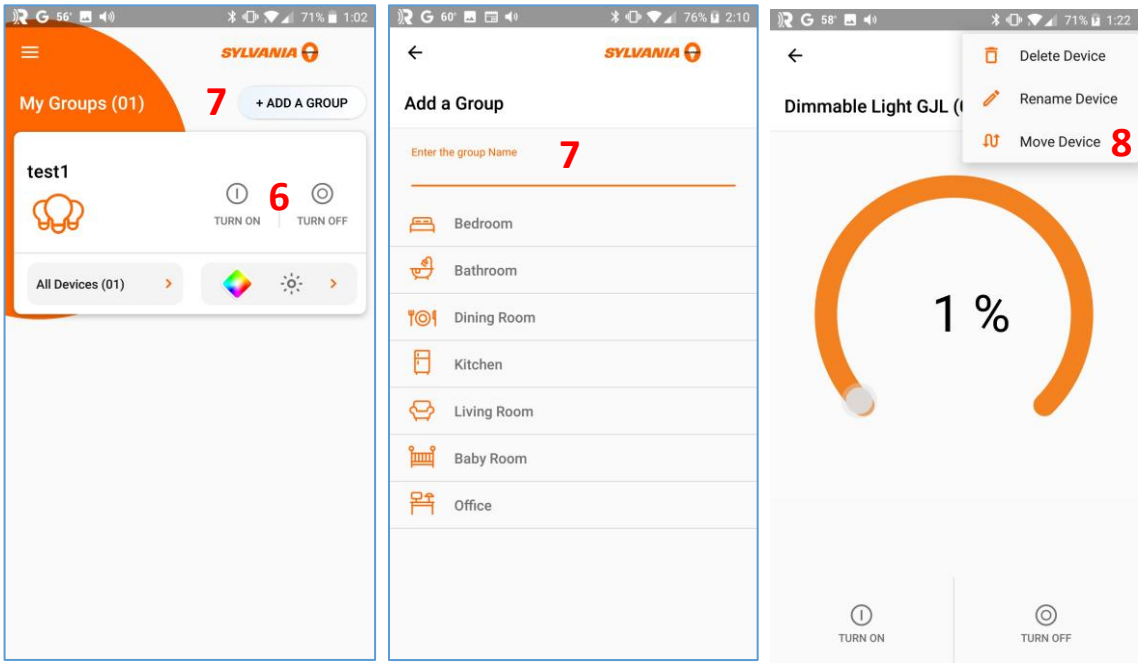
The basic flow for using the application is:

1. Register for an account
2. Select the *test1* group
3. Click *ADD A DEVICE*
4. Select your device from the list
    a. This will take a few seconds. Wait until it completes before proceeding
5. Go back to the test 1 group page
6. Control the device
    a. Note: you can control all devices simultaneously at the group level or individually at the device level
7. Optional: Add additional Groups (i.e. Rooms) from the My Groups screen.
    a. You can choose from a predefined list of rooms or enter your own name.
8. Optional: Move devices to other Groups

Note: <u>Do NOT use the dimmer controls in the app since it is not compatible with the demo firmware. If you try dimming from the Sylvania Smart Home app, you will need to factory reset and re-provision your kit to get it to control the LED again</u>.




Add Device
Select a device to add to your room
Dimmable Light GJL
4
5
Dimmable Light GJL (0203)
Rename
Device Added Successfully
Test
test1
+ ADD A DEVICE
GROUP CONTROLS
DEVICES
100 %
TURN ON
6
TURN OFF


My Groups (01)
7
+ ADD A GROUP
test1
TURN ON
6
TURN OFF
All Devices (01)
Add a Group
Enter the group Name
7
Bedroom
Bathroom
Dining Room
Kitchen
Living Room
Baby Room
Office
Delete Device
Rename Device
Move Device
8
1 %
TURN ON
TURN OFF

7A.5.5 Mesh App (Android)

The Cypress iOS app is similar to the Android app. The app communicates with the mesh network using the device's BLE capabilities. Since smartphones don't (yet) have mesh capability, the app uses GATT connections for provisioning and relies on the presence of a GATT proxy for mesh configuration and communication.

The app can create mesh networks, provision/configure devices and can control lighting devices. The project is located at:

> <Install Folder>/ModusToolbox_1.1/libraries/bt_sdk-1.1/components/BT-SDK/common/apps/snip/mesh/peerapps/iOS/MeshApp

Note: To get the latest version, you should refer to the BT example repository located on GitHub. The path to the file in the repository is:

> https://github.com/cypresssemiconductorco/Code-Examples-BT-SDK-for-ModusToolbox/blob/master/Mesh-Peer-Apps/iOS.zip

Since the app is not (yet) in the Apple Store, it is necessary to install it using a MAC that has the iOS development environment installed. The details of building and installing this app on an iOS device are left as an exercise for the reader.

7A.6 Demo

A mesh network with several lights and rooms will be demonstrated to the class at this point.

7A.7 Exercises

Exercise 7A.1 Create Network with a LightDimmable Device

In this exercise you will create your own (very small) mesh network.

1. In ModusToolbox IDE, create a new application for:
    a. Target Hardware: CYBT-213043-MESH
    b. Starter Application: BLE_Mesh_LightDimmable
2. Open the file "light_dimmable.c" and find the "mesh_dev_name". Change the name so that it has your initials in it (e.g. "<Inits> Dimmable Light").
3. Program the project to one of the CYBT-213043-MESH kits.
    a. Hint: You should open a terminal window for the PUART to see messages. **The default PUART baud rate for the mesh applications is 921600**.
        i. Hint: If your terminal emulator does not support 921600, from ModusToolbox, open the file in libraries/mesh_app_lib/mesh_app_hci.c and search for 921600. Change that value to one that is supported and rebuild/reprogram.
4. Run the Mesh Lighting application to provision the device.
    a. Hint: If you don't have an Android smartphone, the instructor can provide a tablet with the app pre-installed.
    b. Hint: If you don't see any devices listed after ~10 seconds, exit the app, stop/restart BLE and then try again.