AN1438: Networked Lighting Control

Networked lighting control (NLC) systems feature an intelligent network of individually addressable and sensor-rich luminaires and control devices that allows each component of the system to send and receive data. Specifically designed to meet the scale, reliability, and security demands required in commercial settings, Bluetooth® NLC is the only full-stack standard for wireless lighting control. By offering standardization from the radio through the device layer, Bluetooth® NLC enables true multi-vendor interoperability and mass adoption of wireless lighting control.

The Basic Lightness Controller NLC Profile specifies the requirements for a networked lighting control (NLC) product acting as a luminaire controller in a Bluetooth mesh system. The Basic Lightness Controller NLC Profile standardizes the use cases and implementation patterns of luminaire controllers to help improve interoperability and performance of systems based on Bluetooth mesh, such as NLC systems.

A common use case for the Basic Lightness Controller NLC Profile is a luminaire re-acting to information published by occupancy and/or ambient light sensors as well as reacting to override events (e.g., manually dimming/brightening the lights or turning them on/off) in NLC systems.

AVAILABLE PROFILES

- Basic Lightness Controller NLC Profile
- Occupancy Sensor NLC Profile
- Ambient Light Sensor NLC Profile
- Dimming Control NLC Profile
- Basic Scene Selector NLC Profile

1 Introduction

Part of the reason that Bluetooth technology has become such a prevalent technology in people's lives is its reliance on standards. Through standards, the Bluetooth connections made between phones, PCs, headphones, game controllers, vehicle entertainment systems, and countless others all work because they all have built and established trust in the device profiles that initiate these connections, and they are universally used and available.

The Bluetooth SIG is taking that to the next step with the release of their new Network Lighting Control (NLC) bundle of device profiles. These standardized profiles will improve interoperability and scalability, simplify integration in the field, and grow the Bluetooth ecosystem. The same Silicon Labs Bluetooth SoCs and modules that support the new feature enhancements can also support the following NLC profiles: ambient light sensor, basic scene selector, dimming control, basic lightness controller, and occupancy sensor.

1.1 Lighting Control Device Profiles

Device profiles define which options and features of the Bluetooth Mesh specifications are mandatory for a certain kind of end product. The first suite of mesh device profiles, collectively referred to as NLC profiles, build on Bluetooth Mesh to enable the world's first full-stack, multi-vendor interoperable wireless standard for wireless lighting control, Bluetooth® NLC.

1.2 NLC Profile Relationships

A device implementing the *Basic Lightness Controller NLC Profile* interacts with devices implementing the following NLC profiles: *Occupancy Sensor NLC Profile*, *Ambient Light Sensor NLC Profile*, *Dimming Control NLC Profile*, *Basic Scene Selector NLC Profile* (see Figure 1).

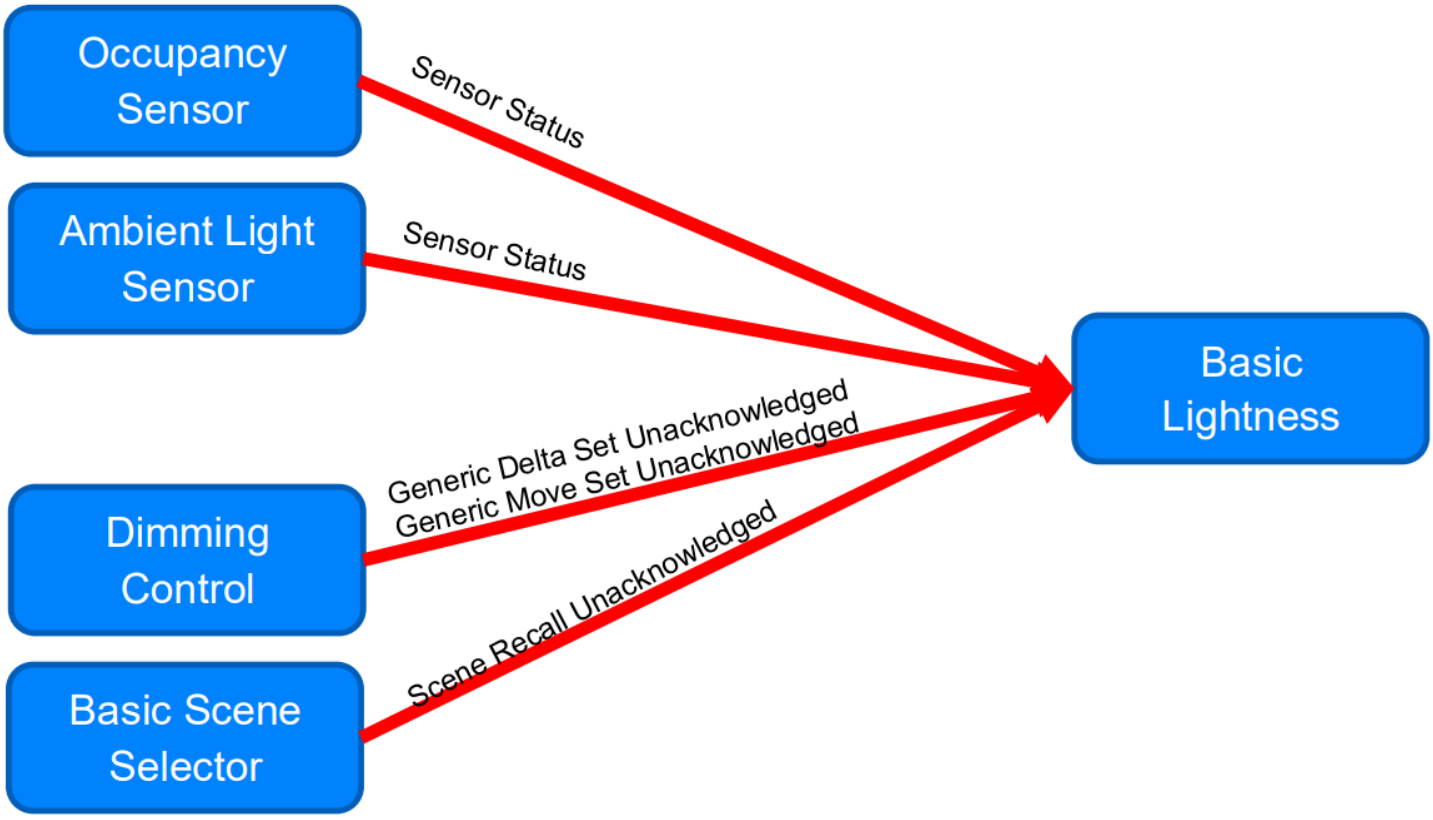


Figure 1. Interaction of a Basic Lightness Controller with other NLC Profiles

2 Implementation

Silicon Labs provides five examples to demonstrate this feature, one for every NLC Profile (except the Energy Monitor NLC Profile). All the examples and their main functionality logics are detailed below.

2.1 Bluetooth Mesh - NLC Basic Lightness Controller

btmesh_soc_nlc_basic_lightness_controller - This is an example of a Bluetooth Mesh application providing the functionality of the Basic Lightness Controller NLC Profile. It demonstrates how to control a light source, an LED mounted on a mainboard, and a radio board or similar hardware, connected to a Bluetooth Mesh network. The light source lightness can be controlled with a Generic Level Client, e.g. another radio board running the Bluetooth Mesh - NLC Dimming Control application, or with Bluetooth Mesh smartphone application.

The device listens to messages from other NLC devices, namely Occupancy Sensor, Ambient Light Sensor, Dimming Control, and Basic Scene Selector nodes. Only a very simple implementation is needed on the application level.

```
/***************************************************************************
 * Callback for setting Light Lightness by PWM level (0x0001 - FFFE)
 **************************************************************************/
void sl_btmesh_lighting_level_pwm_cb(uint16_t level)
{
  app_led_set_level(level);
}

/***************************************************************************
 * Callback for setting Light Color by PWM level (0x0001 - FFFE)
 **************************************************************************/
void sl_btmesh_lighting_color_pwm_cb(uint16_t color)
{
  app_led_set_color(color);
}
```

2.2 Bluetooth Mesh - NLC Basic Scene Selector

btmesh_soc_nlc_basic_scene_selector - This is an example of a Low Power Node-enabled Bluetooth Mesh application providing the functionality of the Basic Scene Selector NLC Profile. Once the node is provisioned and a Basic Lightness Controller subscribes to the client, the two buttons of the mainboard are used to publish the messages that will recall stored scenes on the server. The example provides this recall functionality for scenes #1 and #2. The scene store functionality is provided by the mobile application.

Push Button presses control Basic Lightness Controllers in the network by scene recall requests. This requires the scene selection logic to be implemented in the button handler function.

```
/***************************************************************************
 * Button press Callbacks
 **************************************************************************/
void app_button_press_cb(uint8_t button, uint8_t duration)
{
  (void)duration;
  char scene[SCENE_BUF_LEN];
#if SL_SIMPLE_BUTTON_COUNT == 1
  if (button == BUTTON_PRESS_BUTTON_0) {
    // Switch between scene 1 and 2
    current_scene = current_scene == 1 ? 2 : 1;
    sl_btmesh_select_scene(current_scene);
  }
#endif
#if SL_SIMPLE_BUTTON_COUNT >= 2
  // Select scene by pushed button
  current_scene = button == BUTTON_PRESS_BUTTON_0 ? 1 : 2;
  sl_btmesh_select_scene(current_scene);
#endif
  // Create unique device name using the last two bytes of the device UUID
  snprintf(scene, SCENE_BUF_LEN, "Selected scene: %d", current_scene);
  lcd_print(scene, SL_BTMESH_WSTK_LCD_ROW_CURRENT_SCENE_CFG_VAL);
}
```

2.3 Bluetooth Mesh - NLC Dimming Control

btmesh_soc_nlc_dimming_control - This is an example of a Low Power Node-enabled Bluetooth Mesh application providing the functionality of the Dimming Control NLC Profile. Once the node is provisioned and a Basic Lightness Controller subscribes to the client, the two buttons of the mainboard are used to publish the messages that will change the state of the Basic Lightness Controller node.

Buttons can control the following models:

- Generic On/Off Client model can turn the light on and off or toggle
- Generic Level Client model can control the light brightness

Push Button presses control Basic Lightness Controllers in the network by Generic Level Delta or Generic On/Off messages. A slightly more complex logic is needed for the dimming in the button handler function.

```
/**************************************************************************
 * Button press Callbacks
 *************************************************************************/
void app_button_press_cb(uint8_t button, uint8_t duration)
{
#if SL_SIMPLE_BUTTON_COUNT == 1
  if (button == BUTTON_PRESS_BUTTON_0) {
    switch (duration) {
      // Handling of button press less than 0.25s
      case APP_BUTTON_PRESS_DURATION_SHORT: {
        sl_btmesh_generic_level_client_ext_delta_set_unack(delta);
      } break;
      // Handling of button press greater than 0.25s and less than 1s
      case APP_BUTTON_PRESS_DURATION_MEDIUM: {
        sl_btmesh_generic_level_client_ext_delta_set_unack(-delta);
      } break;
      // Handling of button press greater than 1s
      case APP_BUTTON_PRESS_DURATION_LONG:
      case APP_BUTTON_PRESS_DURATION_VERYLONG: {
        sl_btmesh_change_switch_position(SL_BTMESH_LIGHTING_CLIENT_TOGGLE);
      } break;
      default:
        break;
    }
  }
#endif
#if SL_SIMPLE_BUTTON_COUNT >= 2
  // Selecting action by duration
  switch (duration) {
    // Handling of button press less than 0.25s
    case APP_BUTTON_PRESS_DURATION_SHORT: {
      int32_t delta_set;
      delta_set = button == BUTTON_PRESS_BUTTON_0 ? -delta : delta;
      sl_btmesh_generic_level_client_ext_delta_set_unack(delta_set);
      break;
    }
    // Anything more than 0.25s
    case APP_BUTTON_PRESS_DURATION_MEDIUM:
    case APP_BUTTON_PRESS_DURATION_LONG:
    case APP_BUTTON_PRESS_DURATION_VERYLONG: {
      uint8_t   on_off   =   button   ==   BUTTON_PRESS_BUTTON_0   ?   SL_BTMESH_LIGHTING_CLIENT_OFF   :
SL_BTMESH_LIGHTING_CLIENT_ON;
      sl_btmesh_change_switch_position(on_off);
      break;
    }
    default:
      break;
  }
#endif
}
```

2.4 Bluetooth Mesh - NLC Ambient Light Sensor

btmesh_soc_nlc_sensor_ambient_light - This example makes ambient light sensor measurements and forwards them to another node implementing the Basic Lightness Controller NLC Profile.

The device measures ambient light and sends these measurements to the network. Properly configured NLC Basic Lightness Controllers then can act on the received data. This example contains a Mock sensor logic in the button handler, in case there is no actual light sensor present on the board, thus revealing a possible logic to implement.

```
/*******************************************************************************
 * Callback for button press
 ******************************************************************************/
void app_button_press_cb(uint8_t button, uint8_t duration)
{
  #if defined(SL_CATALOG_SENSOR_LIGHT_LUX_MOCK_PRESENT)
  float uvi;
  float lux;
  sl_status_t sc;
  sc = sl_sensor_light_get(&lux, &uvi);
  app_assert_status_f(sc, "Failed to get lux and uvi");
  #if (SL_SIMPLE_BUTTON_COUNT >= 2)
  // button pressed
  if (duration == APP_BUTTON_PRESS_DURATION_SHORT) {
    if (button == BUTTON_PRESS_BUTTON_0) {
      app_log("PB0 pressed" APP_LOG_NL);
      sl_sensor_light_set(lux - LUX_SMALL_CHANGE, uvi);
    } else if (button == BUTTON_PRESS_BUTTON_1) {
      app_log("PB1 pressed" APP_LOG_NL);
      sl_sensor_light_set(lux + LUX_SMALL_CHANGE, uvi);
    }
  } else if (duration == APP_BUTTON_PRESS_DURATION_MEDIUM) {
    if (button == BUTTON_PRESS_BUTTON_0) {
      app_log("PB0 medium pressed" APP_LOG_NL);
      sl_sensor_light_set(lux - LUX_LARGE_CHANGE, uvi);
    } else if (button == BUTTON_PRESS_BUTTON_1) {
      app_log("PB1 medium pressed" APP_LOG_NL);
      sl_sensor_light_set(lux + LUX_LARGE_CHANGE, uvi);
    }
  }
  #elif (SL_SIMPLE_BUTTON_COUNT == 1)
  (void)duration;
  if (button == BUTTON_PRESS_BUTTON_0) {
    app_log("PB0 pressed" APP_LOG_NL);
    sl_sensor_light_set(lux + LUX_LARGE_CHANGE, uvi);
  }
  #endif // SL_SIMPLE_BUTTON_COUNT
  #else
  (void)duration;
  (void)button;
  #endif // SL_CATALOG_SENSOR_LIGHT_MOCK_PRESENT
}
```

2.5 Bluetooth Mesh - NLC Occupancy Sensor

btmesh_soc_nlc_sensor_occupancy – This example makes Occupancy Sensor measurements and forwards them to another node implementing the Basic Lightness Controller NLC Profile. This measurement can be faked by pressing the buttons on the mainboard:

- BTN1 increases people count
- BTN0 decreases people count

In case of only one button, a shorter press increases, while a longer one decreases the people count.

Push Button presses imitate people count changes which can control a properly configured NLC Basic Lightness Controller. Here as well, buttons simulate the increase/decrease of people, implemented in the button press handler function.

```
/******************************************************************************
 * Callback for button press
 *****************************************************************************/
void app_button_press_cb(uint8_t button, uint8_t duration)
{
  (void)duration;
  // button pressed
  if (duration == APP_BUTTON_PRESS_DURATION_SHORT) {
    if (button == BUTTON_PRESS_BUTTON_0) {
      app_log("PB0 pressed" APP_LOG_NL);
      sl_btmesh_people_count_decrease();
    } else if (button == BUTTON_PRESS_BUTTON_1) {
      app_log("PB1 pressed" APP_LOG_NL);
      sl_btmesh_people_count_increase();
    }
  } else if (duration == APP_BUTTON_PRESS_DURATION_MEDIUM) {
    if ( (button == BUTTON_PRESS_BUTTON_0)) {
      app_log("PB0 medium pressed" APP_LOG_NL);
      sl_btmesh_people_count_increase();
    }
  }
}
```

3 Network Setup

To create a demonstration network, at least five WSTK + radio boards or Thunderboards are needed, preferably ones equipped with display, and at least one of them with light sensor for the *btmesh_soc_nlc_sensor_ambient_light* example and another one with RGB LED for the *btmesh_soc_nlc_basic_lightness_controller* example.

3.1 Flashing

You may use the provided demo binaries or create your own project and flash it to the board with the bootloader of your choice. Doing these should lead to the terminal outputs below:

3.1.1 Bluetooth Mesh - NLC Basic Lightness Controller

```
.BT Mesh NLC Basic Lightness Controller initialized
> Device name: 'NLC Light 9356'
BT mesh node is unprovisioned, started unprovisioned beaconing...
```

3.1.2 Bluetooth Mesh - NLC Basic Scene Selector

```
BT Mesh NLC Basic scene selector initialized
> Device name: 'NLC Scene node 88f8'
BT mesh node is unprovisioned, started unprovisioned beaconing...
```

3.1.3 Bluetooth Mesh - NLC Dimming Control

```
.BT Mesh NLC Dimming Control initialized
> Device name: 'NLC Dimmer 3006'
BT mesh node is unprovisioned, started unprovisioned beaconing...
```

3.1.4 Bluetooth Mesh - NLC Ambient Light Sensor

```
.BT mesh NLC Ambient Light Sensor initialized
> Device name: 'NLC Amb. Light 52bb'
BT mesh node is unprovisioned, started unprovisioned beaconing...
```

3.1.5 Bluetooth Mesh - NLC Occupancy Sensor

```
BT mesh NLC Occupancy Sensor initialized
> Device name: 'NLC Occupancy bd74'
BT mesh node is unprovisioned, started unprovisioned beaconing...
```

3.2 Bluetooth Mesh App

With the help of Silicon Labs' own Bluetooth Mesh App, scan for the Nodes and start to provision them one by one into your own NLC Mesh network.

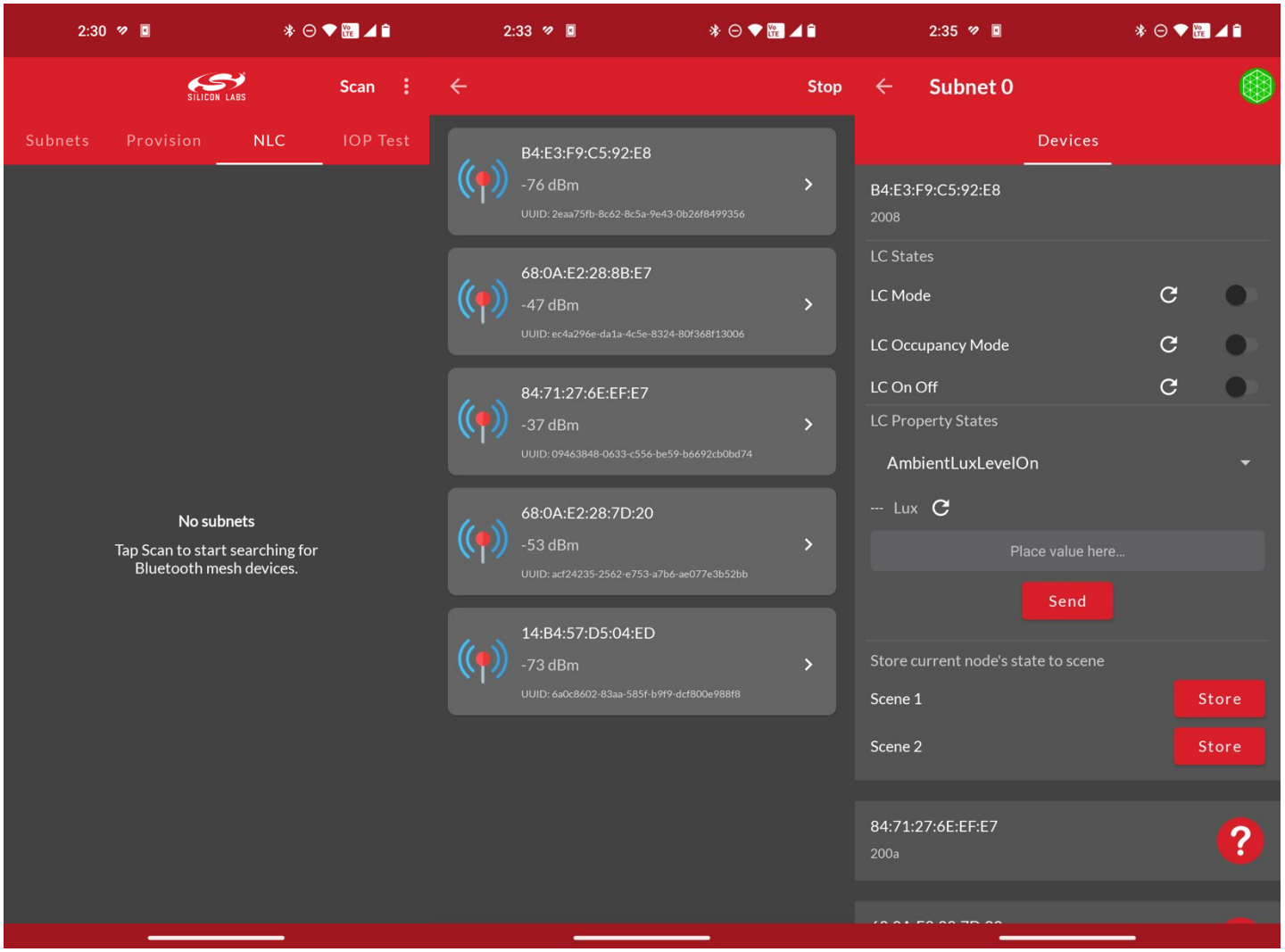


Figure 2. Before and After the Provisioning Process

3.2.1 Bluetooth Mesh - NLC Basic Lightness Controller

```
Connected
BT mesh node provisioning is started (result: 0x0000)
BT mesh node is provisioned (address: 0x2008, iv_index: 0x0)
[E] Status: sc = 0x041c (?) Lighting server lightbulb state load from PS
failed or nvm is empty, use defaults.
[I] On power up state is OFF
BT mesh Lightness:     0%
[D] Status: sc = 0x0514 (?) Lighting server state publish failed
(mdl=0x1006,elem=0,state=0x0001)
[D] Status: sc = 0x0514 (?) Lighting server state publish failed
(mdl=0x1000,elem=0,state=0x0000)
[D] Status: sc = 0x0514 (?) Lighting server state publish failed
(mdl=0x1300,elem=0,state=0x0080)
[E] Status: sc = 0x041c (?) LC server lc_state load from PS failed or nvm
is empty, use defaults.
[E] Status: sc = 0x041c (?) LC server lc_property_state load from PS
failed or nvm is empty, use defaults.
[I] Friend mode initialization
[I] Dummy LC ON/OFF change - same state as before
[I] Lightness update -same value (0)
BT mesh Friendship established with LPN (netkey idx: 0, lpn addr: 0x2007)
```

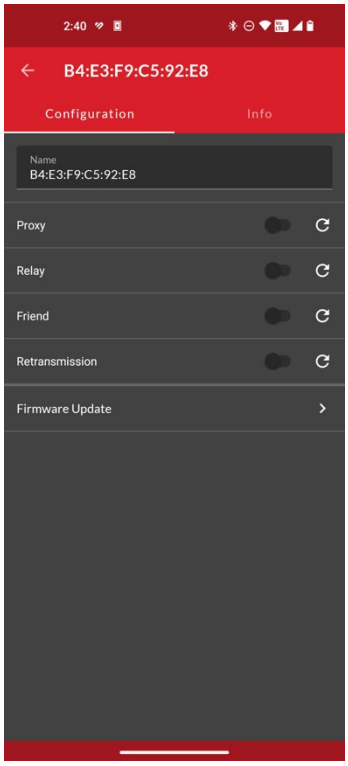


Figure 3. Provisioning and Configuration of the NLC Basic Lightness Controller Node

3.2.2 Bluetooth Mesh - NLC Basic Scene Selector

```
Connected
BT mesh node provisioning is started (result: 0x0000)
BT mesh node is provisioned (address: 0x2007, iv_index: 0x0)
[I] Trying to initialize lpn...
Disconnected
[I] LPN initialized
BT mesh LPN on
[I] Trying to find a friend...
```

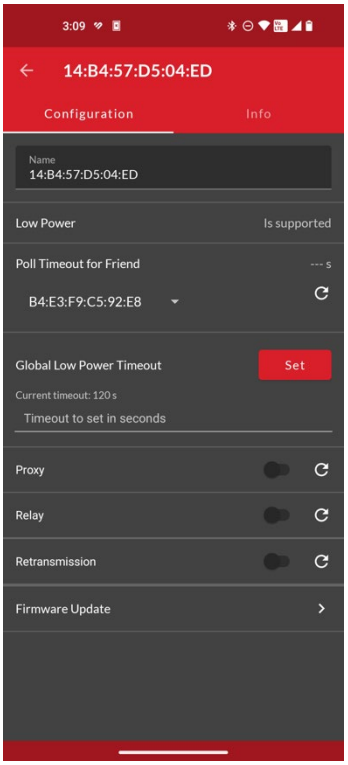
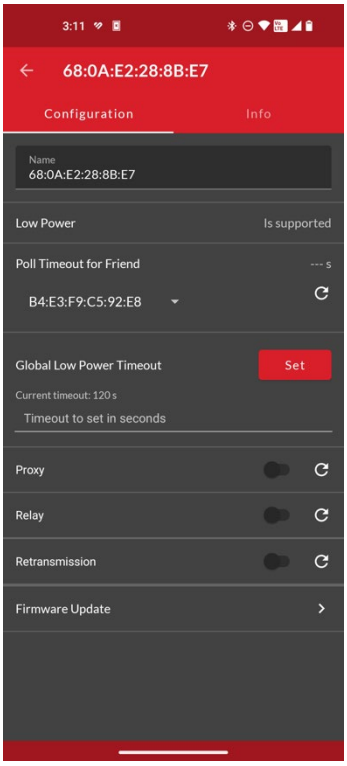


Figure 4. Provisioning and Configuration of the NLC Basic Scene Selector Node

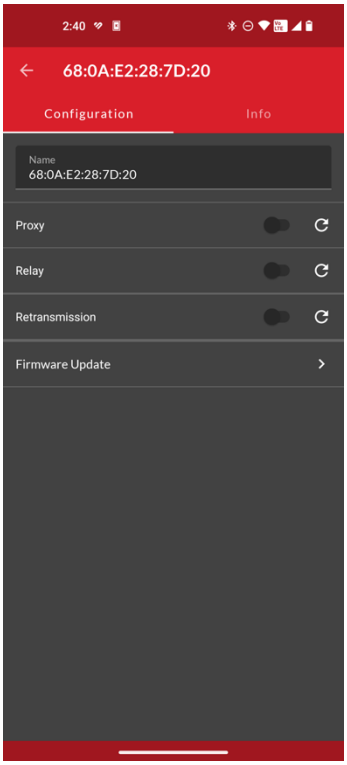
3.2.3 Bluetooth Mesh - NLC Dimming Control




```
Connected
BT mesh node provisioning is started (result: 0x0000)
BT mesh node is provisioned (address: 0x200a, iv_index: 0x0)
[I] Trying to initialize lpn...
Disconnected
[I] LPN initialized
BT mesh LPN on
[I] Trying to find a friend...
```

Figure 5. Provisioning and Configuration of the NLC Dimming Control Node

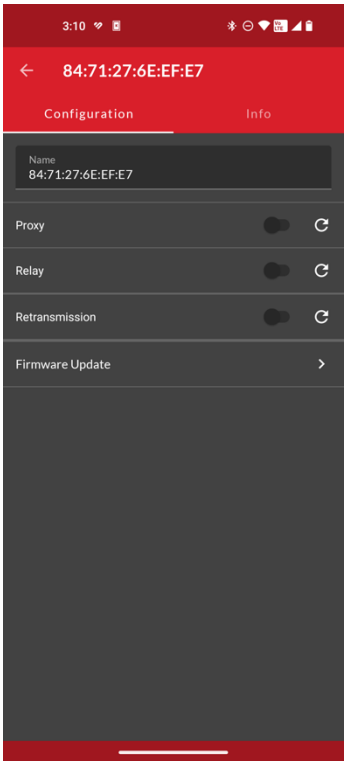
3.2.4 Bluetooth Mesh - NLC Ambient Light Sensor




```
Connected
BT mesh node provisioning is started (result: 0x0000)
BT mesh node is provisioned (address: 0x2006, iv_index: 0x0)
Illuminance:      0.00 lx
Disconnected
```

Figure 6. Provisioning and Configuration of the NLC Ambient Light Sensor Node

3.2.5 Bluetooth Mesh - NLC Occupancy Sensor

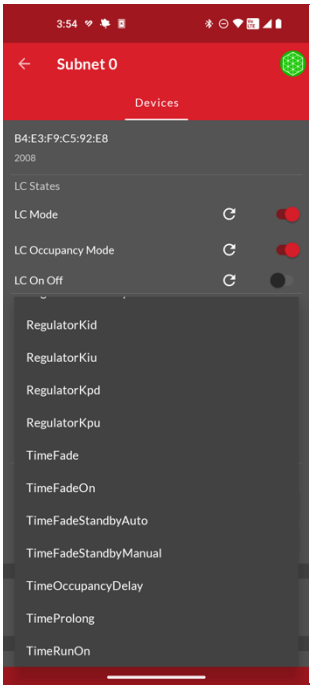



```
Connected
BT mesh node provisioning is started (result: 0x0000)
BT mesh node is provisioned (address: 0x2005, iv_index: 0x0)
People count:     0
Disconnected
```

Figure 7. Provisioning and Configuration of the NLC Occupancy Sensor Node

3.3 Altering the Network

After the full setup, the network parameters are adjustable by changing the LC Property States, and the network can be tested by triggering changes with the push buttons. Switching LC Mode OFF, the NLC Dimming Control Node can be used, and by switching it ON, the NLC Ambient Light Sensor going to be active. By switching Occupancy Mode ON, the NLC Occupancy Sensor Node will take over.




```
People count:     1
PB0 medium pressed
People count:     2
PB0 medium pressed
People count:     3
PB1 medium pressed
Illuminance:     50.00 lx
PB1 medium pressed
Illuminance:    100.00 lx
PB1 medium pressed
Illuminance:    150.00 lx
BT mesh Lightness:    95%
[I] lightness_change, kind 129, value 60323
[I] Primary level update: from 30105 to 30107
[I] Dummy ON/OFF change - same state as before
[I] Lightness update: from 62873 to 62875
[I] lightness_change, kind 130, value 62875
[I] LC ON/OFF state changed 0 to 1
BT mesh Lightness:   100%
[I] lightness_change, kind 129, value 65535
[I] Primary level update: from 30107 to 32767
[I] Dummy ON/OFF change - same state as before
[I] Lightness update: from 62875 to 65535
[I] lightness_change, kind 130, value 65535
[I] LC ON/OFF state changed 1 to 0
[I] LC ON/OFF state changed 0 to 1
```

Figure 8. Configuration of the Network and Node Logs

4 Custom Project

To create a custom NLC Profile project, you have to add the appropriate NLC component to your project, which in turn will install the corresponding NLC Profile component. This is only valid for SoC projects. In the case of NCP, only the NLC Profile component is required.

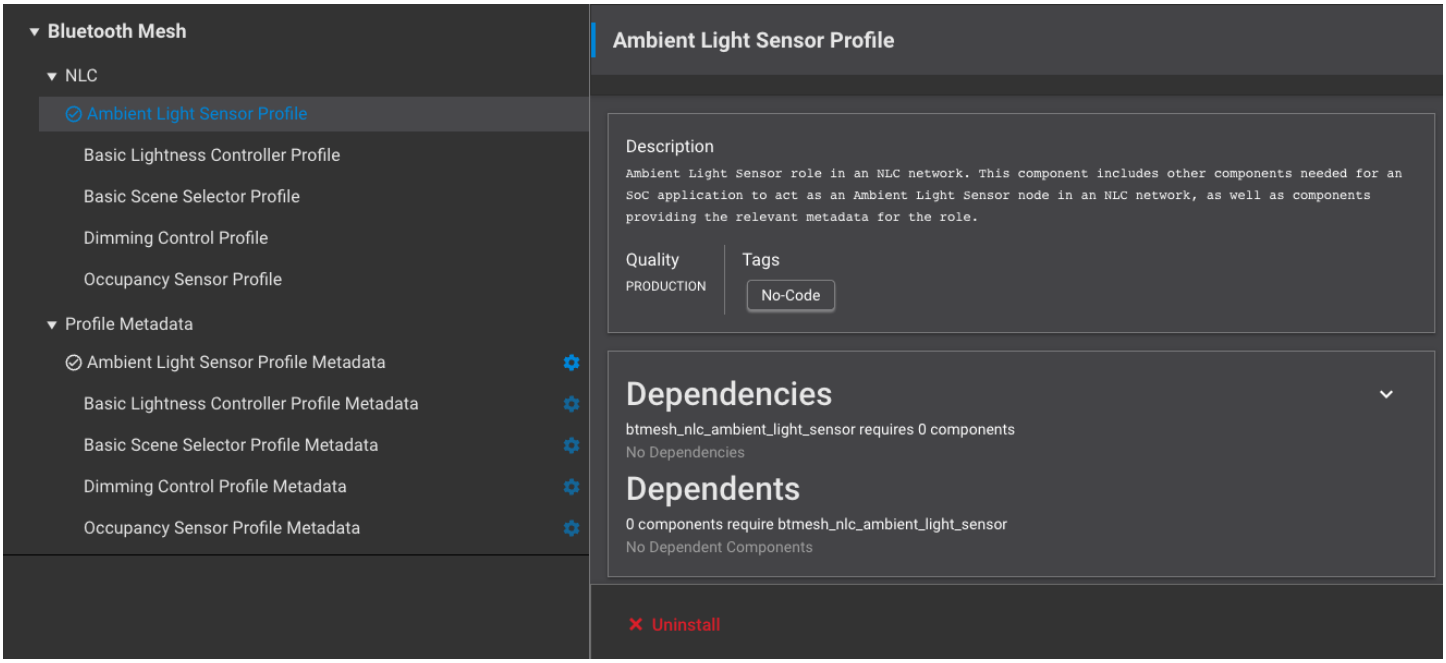


Figure 9. Simplicity Studio 5 Component Manager

Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support & Community
www.silabs.com/community

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

www.silabs.com