# BLE Mesh Demo

|  |  |
| --- | --- |
| **Document version** | **Author** |
| Nov14-2018 | MQ |
|  |  |
|  |  |

## Purpose

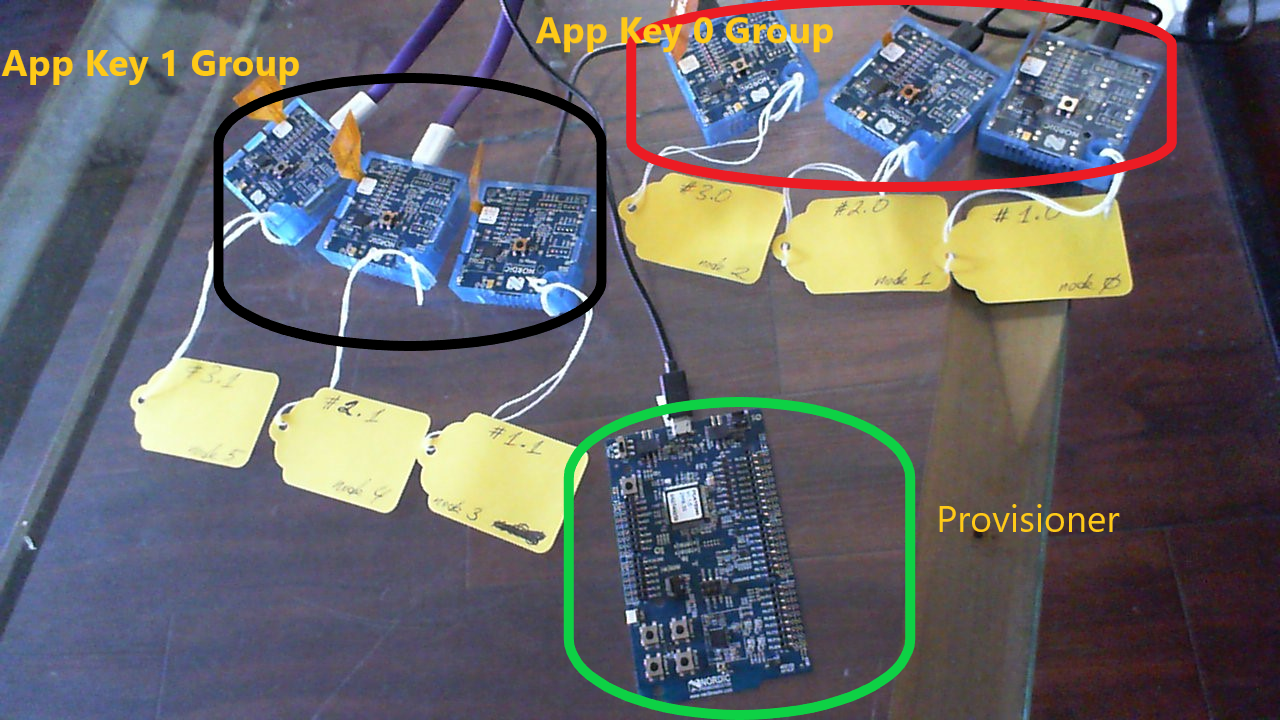
This document will explain the Mesh Demo and the motivations behind the design decisions.

## Architecture

The mesh network realized in this demo has the following points to show:

1. Create a mesh network with two groups of Thingy52s. The two groups are on the same Mesh network (common network key) but they cannot decode the other group’s messages because they have separate application keys. The purpose of creating and bifurcating two groups such as this is to simulate two separate drawers on a station.
2. Demonstrate one Thingy52 from one group to be able to “move” to the other group and be part of the new group. This would demonstrate the cubies switching drawers by wirelessly being reconfigured to belong in the new group.

The above two points should demonstrate isolation of traffic between two drawers while allowing for any cubie to be reconfigured for re-use with either drawer.



Picture above shows the division of the mesh network. The Provisioner is attached to the PC via USB and allows for PyACI to send commands to it.

Below is a diagram of a node and the capabilities built into it via the firmware:

Please see BLE Mesh profile document for definitions of Elements.

1. Configuration Server
2. Health Server
3. Generic On/Off Server
4. Generic On/Off Client
5. Generic On/Off Server
6. Generic On/Off Client

Element 1

Element 2

In this demo, we only use the Generic On/Off Server on Element 1. In a mesh network, a server controls access to a resource. In this case, the resource is the LED. We will command the Generic On/Off Server via a Generic On/Off Client. In this case, the Client will be the same as the Provisioner but they do not have to be. In other words, another node in the mesh network with the proper network and application key can successfully command the server to switch the LED on and off. The Provisioner is a special node that assigns each node its role. So, in this instance, a provisioner assigns which application key and which group an unprovisioned node belongs to. The demo comes as pre-provisioned nodes but nodes can be erased and re-provisioned as needed.

## Running demo out-of-the-box

### To use the demo as-is without the need to re-flash, the database file holding the necessary keys must be used.

The database file that holds the provisioning information resides in nrf5\_SDK\_for\_Mesh\_v2.2.0\_src/scripts/interactive\_pyaci/database/ and it is called “bd\_demo\_database\_final.json”. Please follow steps in the Restoring a mesh network after reset of provisioner section and replace the following command

In **[**2**]:** db **=** MeshDB**(**"database/example\_database.json"**)**

With

In **[**2**]:** db **=** MeshDB**(**"database/bd\_demo\_database\_final.json"**)**

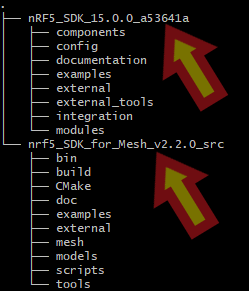
This will load the Provisioner with proper values to be able to access the Mesh network as was demonstrated on-site.

If it is desired to create a mesh network from scratch, please follow instructions in the Provisioning section.

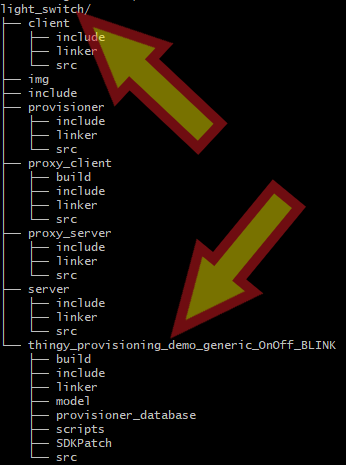
## Installation

The demo is distributed with the Mesh SDK 2.2 and [SDK 15](http://developer.nordicsemi.com/nRF5_SDK/nRF5_SDK_v15.x.x/) and it lives in a subdirectory in the Mesh SDK 2.2 source code. Please follow the instructions here to install the [toolchain](http://infocenter.nordicsemi.com/topic/com.nordic.infocenter.meshsdk.v2.2.0/md_doc_getting_started_how_to_toolchain.html?cp=4_1_0_1_0) and [SES](http://infocenter.nordicsemi.com/topic/com.nordic.infocenter.gs.ses/dita/gs/developing/install_segger_tools.html?cp=1_1_0_6_2).

The following diagram illustrates the directory structure:

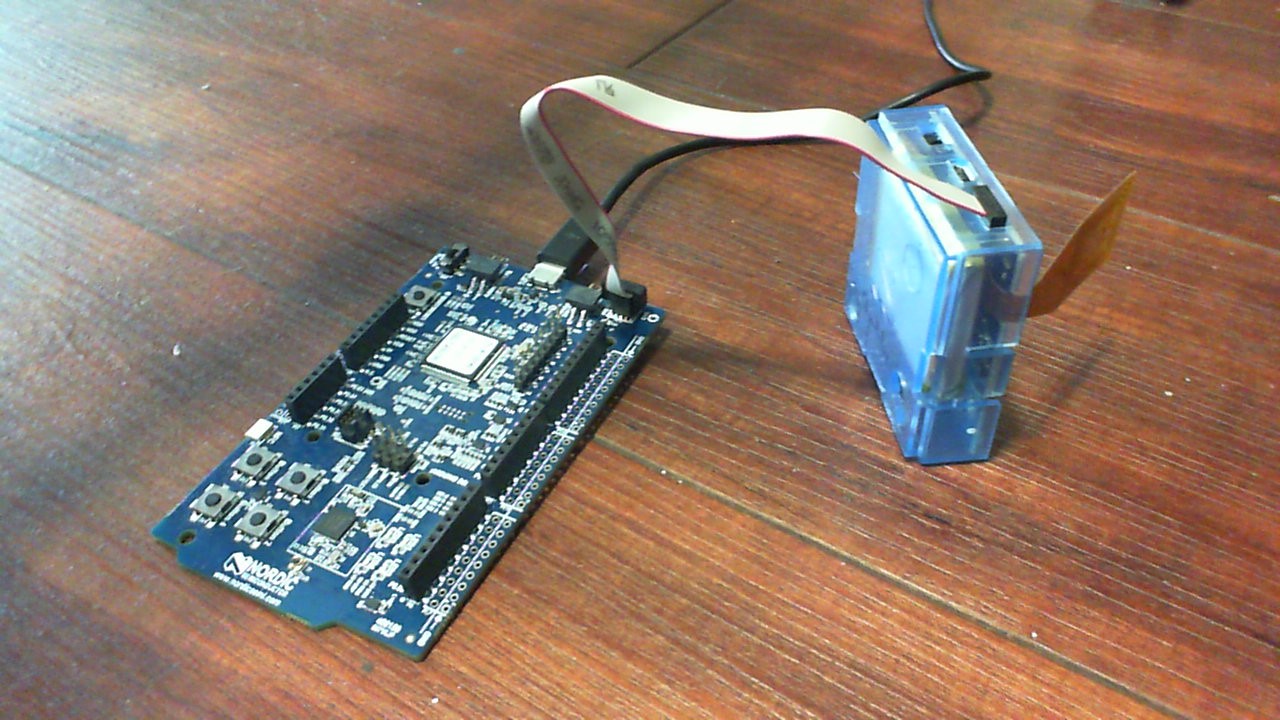


Within the Mesh SDK 2.2 source code, the demo source code lies in the nrf5\_SDK\_for\_Mesh\_v2.2.0\_src/examples/light\_switch/thingy\_provisioning\_demo\_generic\_OnOff\_BLINK/:



Open the Segger Embedded Studio (SES) project file in the Mesh-nrf52840/nrf5\_SDK\_for\_Mesh\_v2.2.0\_src/examples/light\_switch/thingy\_provisioning\_demo\_generic\_OnOff\_BLINK.

Then connect the DK to the Thingy52 node with the cable supplied as shown here:



With the DK connected to the PC, erase the Thingy52 with the following command:



Now launch SES and flash the project to the Thingy52. When the cable is attached to the Thingy52, the DK programs the Thingy52 rather its own flash.

Repeat steps for all Thingy52 nodes.

## Provisioning

### This section will provide information how to set up a mesh network from scratch.

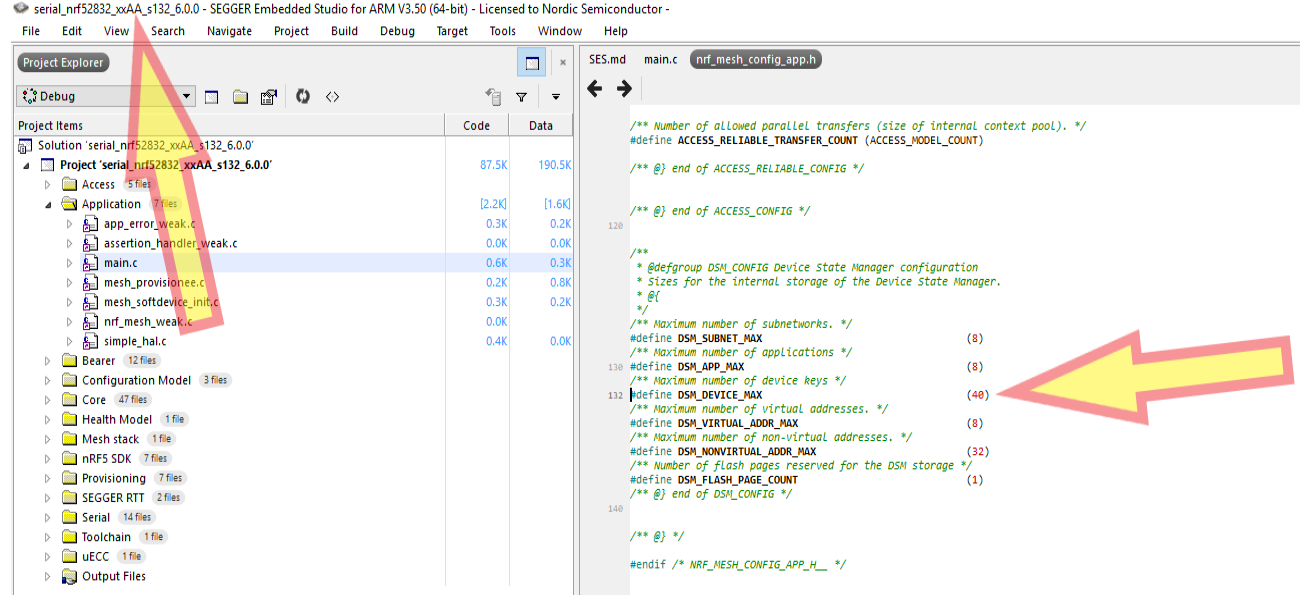
### These steps will guide one to set up a 6 node mesh network with 3 nodes bound to Application Key 0 and 3 nodes bound to Application Key 1. However, this methodology can be extended to any number of nodes.

1. All nodes should be programmed with the firmware as shown in the Installation section and they should come up blinking red:

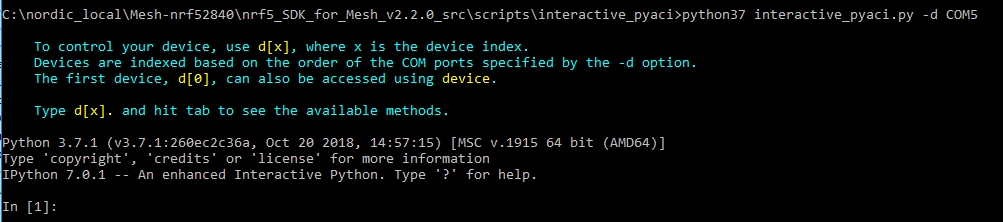


Please keep in mind that the nodes save their provisioning data on the on-board flash. This information can be erased so the node is unprovisioned again by holding down the button on top and powering the switch off and then on.

1. Connect a nRF52-DK to the PC; launch SES with the project in <Mesh SDK 2.2>/examples/serial/. Open the nrf\_mesh\_config\_app.h file and modify DSM\_DEVICE\_MAX to surpass or equal the maximum number of nodes you intend to provision using the nRF52-DK provisioner:



1. Compile, flash, and run the firmware on the nRF52-DK.
2. Follow instructions [here](http://infocenter.nordicsemi.com/topic/com.nordic.infocenter.meshsdk.v2.2.0/md_scripts_interactive_pyaci_README.html?cp=4_1_0_3_0) to install Python 3 and launch the PyACI tool under <Mesh SDK 2.2>/scripts/interactive\_pyaci:



1. Now we will provision each node one-at-a-time. The easiest way is to turn off all the nodes except one which is the one that shall be provisioned. So turn off all nodes except one which would be blinking red indicating that it is unprovisioned.
2. We need to start with a fresh JSON file which will hold the database of the mesh network. This file resides in the nrf5\_SDK\_for\_Mesh\_v2.2.0\_src/scripts/interactive\_pyaci/database/ directory.

Note that this file can be edited by humans since it is human-readable. Copy “bd\_example\_database.json” to “example\_database.json” in the same folder. Overwrite, if needed.

1. Now we will provision the first node. In PyACI shell, type in the highlighted input; example output shown for clarity (not highlighted):

In **[**1**]:** db **=** MeshDB**(**"database/example\_database.json"**)**

In **[**2**]:** db**.**provisioners

Out**[**2**]:** **[{**'name'**:** 'BT Mesh Provisioner'**,** 'UUID'**:** \_UUID**(**b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'**),** 'allocated\_unicast\_range'**:** **[{**'low\_address'**:** 0010**,** 'high\_address'**:** 7fff**}],** 'allocated\_group\_range'**:** **[{**'low\_address'**:** c000**,** 'high\_address'**:** feff**}]}]**

In **[**3**]:** p **=** Provisioner**(**device**,** db**)**

In **[**4**]:** 2018**-**11**-**13 02**:**31**:**25**,**175 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 02**:**31**:**25**,**181 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 02**:**31**:**25**,**186 **-** INFO **-** COM5**:** SubnetAdd**:** **{**'subnet\_handle'**:** 0**}**

2018**-**11**-**13 02**:**31**:**25**,**197 **-** INFO **-** COM5**:** AppkeyAdd**:** **{**'appkey\_handle'**:** 0**}**

2018**-**11**-**13 02**:**31**:**25**,**202 **-** INFO **-** COM5**:** AppkeyAdd**:** **{**'appkey\_handle'**:** 1**}**

In **[**4**]:**

In **[**4**]:** p**.**scan\_start**()**

In **[**5**]:** 2018**-**11**-**13 02**:**31**:**40**,**555 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 02**:**31**:**51**,**343 **-** INFO **-** COM5**:** Received UUID 0059ffff00000000a5cb9cc547cdcea9 **with** RSSI**:** **-**48 dB

In **[**5**]:**

In **[**5**]:** p**.**scan\_stop**()**

In **[**6**]:** 2018**-**11**-**13 02**:**32**:**03**,**714 **-** INFO **-** COM5**:** Success

In **[**6**]:**

In **[**6**]:** p**.**provision**(**name**=**"Light bulb #1"**)**

In **[**7**]:** 2018**-**11**-**13 02**:**32**:**15**,**090 **-** INFO **-** COM5**:** Provision**:** **{**'context'**:** 0**}**

2018**-**11**-**13 02**:**32**:**15**,**102 **-** INFO **-** COM5**:** Link established

2018**-**11**-**13 02**:**32**:**15**,**144 **-** INFO **-** COM5**:** Received capabilities

2018**-**11**-**13 02**:**32**:**15**,**146 **-** INFO **-** COM5**:** Number of elements**:** 3

2018**-**11**-**13 02**:**32**:**15**,**149 **-** INFO **-** COM5**:** OobUse**:** **{**'context'**:** 0**}**

2018**-**11**-**13 02**:**32**:**15**,**368 **-** INFO **-** COM5**:** ECDH request received

2018**-**11**-**13 02**:**32**:**15**,**384 **-** INFO **-** COM5**:** EcdhSecret**:** **{**'context'**:** 0**}**

2018**-**11**-**13 02**:**32**:**15**,**749 **-** INFO **-** COM5**:** Provisioning complete

2018**-**11**-**13 02**:**32**:**15**,**753 **-** INFO **-** COM5**:** Address**(**es**):** 0x10**-**0x12

2018**-**11**-**13 02**:**32**:**15**,**754 **-** INFO **-** COM5**:** Device key**:** ef48f9898373a1cef29db185ee4058b4

2018**-**11**-**13 02**:**32**:**15**,**756 **-** INFO **-** COM5**:** Network key**:** 18eed9c2a56add85049ffc3c59ad0e12

2018**-**11**-**13 02**:**32**:**15**,**759 **-** INFO **-** COM5**:** Adding device key to subnet 0

2018**-**11**-**13 02**:**32**:**15**,**766 **-** INFO **-** COM5**:** Adding publication address of root element

2018**-**11**-**13 02**:**32**:**15**,**783 **-** INFO **-** COM5**:** DevkeyAdd**:** **{**'devkey\_handle'**:** 8**}**

2018**-**11**-**13 02**:**32**:**15**,**786 **-** INFO **-** COM5**:** AddrPublicationAdd**:** **{**'address\_handle'**:** 0**}**

2018**-**11**-**13 02**:**32**:**15**,**853 **-** INFO **-** COM5**:** Provisioning link closed

In **[**7**]:**

In **[**7**]:** cc **=** ConfigurationClient**(**db**)**

In **[**8**]:** cc**.**force\_segmented **=** **True**

In **[**9**]:** device**.**model\_add**(**cc**)**

In **[**10**]:** cc**.**publish\_set**(**8**,** 0**)**

In **[**11**]:** cc**.**composition\_data\_get**()**

In **[**12**]:** 2018**-**11**-**13 02**:**33**:**01**,**420 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 02**:**33**:**01**,**521 **-** INFO **-** COM5**.**ConfigurationClient**:** Received composition data **(**page 0x00**):** **{**

"cid"**:** "0059"**,**

"pid"**:** "0000"**,**

"vid"**:** "0000"**,**

"crpl"**:** 40**,**

"features"**:** **{**

"relay"**:** 0**,**

"proxy"**:** 2**,**

"friend"**:** 2**,**

"low\_power"**:** 2

**},**

"elements"**:** **[**

**{**

"index"**:** 0**,**

"location"**:** "0000"**,**

"models"**:** **[**

**{**

"modelId"**:** "0000"

**},**

**{**

"modelId"**:** "0002"

**},**

**{**

"modelId"**:** "1000"

**},**

**{**

"modelId"**:** "1001"

**}**

**]**

**},**

**{**

"index"**:** 1**,**

"location"**:** "0000"**,**

"models"**:** **[**

**{**

"modelId"**:** "1000"

**},**

**{**

"modelId"**:** "1001"

**}**

**]**

**}**

**]**

**}**

In **[**12**]:**

In **[**12**]:** cc**.**appkey\_add**(**0**)**

In **[**13**]:** 2018**-**11**-**13 02**:**33**:**09**,**562 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 02**:**33**:**09**,**642 **-** INFO **-** COM5**.**ConfigurationClient**:** Appkey status**:** AccessStatus**.**SUCCESS

2018**-**11**-**13 02**:**33**:**09**,**645 **-** INFO **-** COM5**.**ConfigurationClient**:** Appkey add 0 succeded **for** subnet 0 at node 0010

In **[**13**]:**

In **[**13**]:** cc**.**appkey\_add**(**1**)**

In **[**14**]:** 2018**-**11**-**13 02**:**33**:**13**,**850 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 02**:**33**:**13**,**936 **-** INFO **-** COM5**.**ConfigurationClient**:** Appkey status**:** AccessStatus**.**SUCCESS

2018**-**11**-**13 02**:**33**:**13**,**939 **-** INFO **-** COM5**.**ConfigurationClient**:** Appkey add 1 succeded **for** subnet 0 at node 0010

In **[**14**]:**

In **[**14**]:** cc**.**model\_app\_bind**(**db**.**nodes**[**0**].**unicast\_address**,** 0**,** mt**.**ModelId**(**0x1000**))** #bind to appkey0

In **[**15**]:** 2018**-**11**-**13 02**:**33**:**24**,**401 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 02**:**33**:**25**,**361 **-** INFO **-** COM5**.**ConfigurationClient**:** Model app bind status**:** AccessStatus**.**SUCCESS

2018**-**11**-**13 02**:**33**:**25**,**367 **-** INFO **-** COM5**.**ConfigurationClient**:** Appkey bind 0 to model 1000 at 0010

In **[**15**]:**

In **[**15**]:** cc**.**model\_subscription\_add**(**db**.**nodes**[**0**].**unicast\_address**,** 0xc001**,** mt**.**ModelId**(**0x1000**))** #Add to group 0xC001

2018**-**11**-**13 02**:**33**:**35**,**254 **-** INFO **-** COM5**:** Success

In **[**16**]:** 2018**-**11**-**13 02**:**33**:**36**,**299 **-** INFO **-** COM5**.**ConfigurationClient**:** Model subscription status**:** AccessStatus**.**SUCCESS

2018**-**11**-**13 02**:**33**:**36**,**306 **-** INFO **-** COM5**.**ConfigurationClient**:** Added subscription 'c001' to model 1000 at element 0010

1. We will now provision the second node. We will leave the first node switched on since it has already been provisioned and its LED should be off. Switch on the second node and observe its red LED breathing to signify that it is unprovisioned.
2. In the same PyACI shell from step 6 above, we type in the highlighted Python commands:

In **[**16**]:** p**.**scan\_start**()**

In **[**17**]:** 2018**-**11**-**13 03**:**01**:**21**,**404 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 03**:**01**:**22**,**906 **-** INFO **-** COM5**:** Received UUID 0059ffff00000000d4164b7a7253618c **with** RSSI**:** **-**46 dB

In **[**17**]:**

In **[**17**]:** p**.**scan\_stop**()**

2018**-**11**-**13 03**:**01**:**32**,**809 **-** INFO **-** COM5**:** Success

In **[**18**]:**

In **[**18**]:** p**.**provision**(**name**=**"Light bulb #2"**)**

In **[**19**]:** 2018**-**11**-**13 03**:**01**:**55**,**598 **-** INFO **-** COM5**:** Provision**:** **{**'context'**:** 0**}**

2018**-**11**-**13 03**:**01**:**55**,**609 **-** INFO **-** COM5**:** Link established

2018**-**11**-**13 03**:**01**:**55**,**665 **-** INFO **-** COM5**:** Received capabilities

2018**-**11**-**13 03**:**01**:**55**,**666 **-** INFO **-** COM5**:** Number of elements**:** 3

2018**-**11**-**13 03**:**01**:**55**,**669 **-** INFO **-** COM5**:** OobUse**:** **{**'context'**:** 0**}**

2018**-**11**-**13 03**:**01**:**55**,**879 **-** INFO **-** COM5**:** ECDH request received

2018**-**11**-**13 03**:**01**:**55**,**890 **-** INFO **-** COM5**:** EcdhSecret**:** **{**'context'**:** 0**}**

2018**-**11**-**13 03**:**02**:**00**,**318 **-** INFO **-** COM5**:** Provisioning complete

2018**-**11**-**13 03**:**02**:**00**,**321 **-** INFO **-** COM5**:** Address**(**es**):** 0x13**-**0x15

2018**-**11**-**13 03**:**02**:**00**,**325 **-** INFO **-** COM5**:** Device key**:** f2efd4734a036568fda5d6f499292071

2018**-**11**-**13 03**:**02**:**00**,**331 **-** INFO **-** COM5**:** Network key**:** 18eed9c2a56add85049ffc3c59ad0e12

2018**-**11**-**13 03**:**02**:**00**,**337 **-** INFO **-** COM5**:** Adding device key to subnet 0

2018**-**11**-**13 03**:**02**:**00**,**340 **-** INFO **-** COM5**:** Adding publication address of root element

2018**-**11**-**13 03**:**02**:**00**,**361 **-** INFO **-** COM5**:** DevkeyAdd**:** **{**'devkey\_handle'**:** 9**}**

2018**-**11**-**13 03**:**02**:**00**,**362 **-** INFO **-** COM5**:** AddrPublicationAdd**:** **{**'address\_handle'**:** 1**}**

2018**-**11**-**13 03**:**02**:**00**,**422 **-** INFO **-** COM5**:** Provisioning link closed

In **[**19**]:**

In **[**19**]:** cc**.**publish\_set**(**9**,** 1**)**

In **[**20**]:** cc**.**composition\_data\_get**()**

In **[**21**]:** 2018**-**11**-**13 03**:**02**:**23**,**056 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 03**:**02**:**23**,**143 **-** INFO **-** COM5**.**ConfigurationClient**:** Received composition data **(**page 0x00**):** **{**

"cid"**:** "0059"**,**

"pid"**:** "0000"**,**

"vid"**:** "0000"**,**

"crpl"**:** 40**,**

"features"**:** **{**

"relay"**:** 0**,**

"proxy"**:** 2**,**

"friend"**:** 2**,**

"low\_power"**:** 2

**},**

"elements"**:** **[**

**{**

"index"**:** 0**,**

"location"**:** "0000"**,**

"models"**:** **[**

**{**

"modelId"**:** "0000"

**},**

**{**

"modelId"**:** "0002"

**},**

**{**

"modelId"**:** "1000"

**},**

**{**

"modelId"**:** "1001"

**}**

**]**

**},**

**{**

"index"**:** 1**,**

"location"**:** "0000"**,**

"models"**:** **[**

**{**

"modelId"**:** "1000"

**},**

**{**

"modelId"**:** "1001"

**}**

**]**

**}**

**]**

**}**

In **[**21**]:**

In **[**21**]:** cc**.**appkey\_add**(**0**)**

In **[**22**]:** 2018**-**11**-**13 03**:**02**:**44**,**436 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 03**:**02**:**44**,**515 **-** INFO **-** COM5**.**ConfigurationClient**:** Appkey status**:** AccessStatus**.**SUCCESS

2018**-**11**-**13 03**:**02**:**44**,**519 **-** INFO **-** COM5**.**ConfigurationClient**:** Appkey add 0 succeded **for** subnet 0 at node 0013

In **[**22**]:**

In **[**22**]:** cc**.**appkey\_add**(**1**)**

In **[**23**]:** 2018**-**11**-**13 03**:**02**:**48**,**471 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 03**:**02**:**48**,**562 **-** INFO **-** COM5**.**ConfigurationClient**:** Appkey status**:** AccessStatus**.**SUCCESS

2018**-**11**-**13 03**:**02**:**48**,**568 **-** INFO **-** COM5**.**ConfigurationClient**:** Appkey add 1 succeded **for** subnet 0 at node 0013

In **[**23**]:**

In **[**23**]:** cc**.**model\_app\_bind**(**db**.**nodes**[**1**].**unicast\_address**,** 0**,** mt**.**ModelId**(**0x1000**))** #bind to appkey0

In **[**24**]:** 2018**-**11**-**13 03**:**03**:**02**,**567 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 03**:**03**:**02**,**632 **-** INFO **-** COM5**.**ConfigurationClient**:** Model app bind status**:** AccessStatus**.**SUCCESS

2018**-**11**-**13 03**:**03**:**02**,**636 **-** INFO **-** COM5**.**ConfigurationClient**:** Appkey bind 0 to model 1000 at 0013

In **[**24**]:**

In **[**24**]:** cc**.**model\_subscription\_add**(**db**.**nodes**[**1**].**unicast\_address**,** 0xc001**,** mt**.**ModelId**(**0x1000**))** #Add to group 0xC001

In **[**25**]:** 2018**-**11**-**13 03**:**03**:**11**,**058 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 03**:**03**:**11**,**189 **-** INFO **-** COM5**.**ConfigurationClient**:** Model subscription status**:** AccessStatus**.**SUCCESS

2018**-**11**-**13 03**:**03**:**11**,**192 **-** INFO **-** COM5**.**ConfigurationClient**:** Added subscription 'c001' to model 1000 at element 0013

1. Switch on the third node. Now we will provision it, same as above, now shown for brevity without the output:

In **[**27**]:** p**.**scan\_start**()**

In **[**28**]:** p**.**scan\_stop**()**

In **[**29**]:** p**.**provision**(**name**=**"Light bulb #3"**)**

In **[**30**]:** cc**.**publish\_set**(**10**,** 2**)**

In **[**31**]:** cc**.**composition\_data\_get**()**

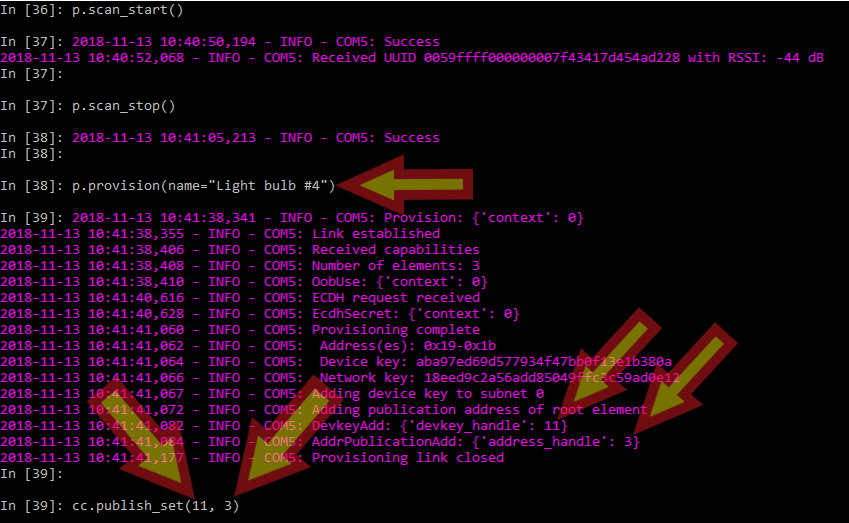
In **[**32**]:** cc**.**appkey\_add**(**0**)**

In **[**33**]:** cc**.**appkey\_add**(**1**)**

In **[**34**]:** cc**.**model\_app\_bind**(**db**.**nodes**[**2**].**unicast\_address**,** 0**,** mt**.**ModelId**(**0x1000**))** #bind to appkey0

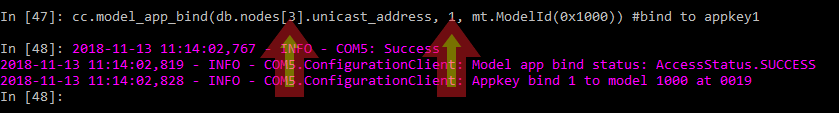
In **[**35**]:** cc**.**model\_subscription\_add**(**db**.**nodes**[**2**].**unicast\_address**,** 0xc001**,** mt**.**ModelId**(**0x1000**))** #Add to group 0xC001

1. Now we will provision the fourth node so switch that node on and it should signal unprovisioned state. However, we will bind it to Application Key 1 instead of Application Key 0. We will note a pattern here that we can use to provision any number of devices:



As seen above, the provision function opens a provisioning link with the device. As a result, we see the devkey\_handle of 11 and address\_handle of 3. We then plug in the devkey\_handle as the first parameter of the publish\_set function and address\_handle as the second parameter of the publish\_set function, as seen above. This is the same pattern that we followed to provision the previous three nodes. The handles here are used as a shortcut to refer to devices, as opposed to having to specify their Mesh address.

Similarly, when we call the model\_app\_bind function, we specify the unique increasing node index number and the application key to bind to:



The node index is simply a number which provides an index into the JSON database file that PyACI creates for a Mesh network. For all intents and purposes, when provisioning a number of devices, this index number simply increments by one for each new node and always starts with 0. In this case, we used 3 because we used 2 in the previous node provision. The second parameter of 1 specifies which application key to bind the node to so the node will respond only to messages encrypted with that particular application key.

Summary commands for provisioning the fourth node are:

In **[**36**]:** p**.**scan\_start**()**

In **[**37**]:** p**.**scan\_stop**()**

In **[**38**]:** p**.**provision**(**name**=**"Light bulb #4"**)**

In **[**39**]:** cc**.**publish\_set**(**11**,** 3**)**

In **[**40**]:** cc**.**appkey\_add**(**1**)**

In **[**41**]:** cc**.**appkey\_add**(**0**)**

In **[**46**]:** cc**.**composition\_data\_get**()**

In **[**47**]:** cc**.**model\_app\_bind**(**db**.**nodes**[**3**].**unicast\_address**,** 1**,** mt**.**ModelId**(**0x1000**))** #bind to appkey1

In **[**48**]:** cc**.**model\_subscription\_add**(**db**.**nodes**[**3**].**unicast\_address**,** 0xc001**,** mt**.**ModelId**(**0x1000**))** #Add to group 0xC001

1. Switch on the fifth node and provision as before. It will follow the same pattern explained in step 10. Here is the summary of commands:

In **[**49**]:** p**.**scan\_start**()**

In **[**50**]:** p**.**scan\_stop**()**

In **[**51**]:** p**.**provision**(**name**=**"Light bulb #4"**)**

In **[**52**]:** cc**.**publish\_set**(**12**,** 4**)**

In **[**53**]:** cc**.**composition\_data\_get**()**

In **[**54**]:** cc**.**appkey\_add**(**1**)**

In **[**55**]:** cc**.**appkey\_add**(**0**)**

In **[**56**]:** cc**.**model\_app\_bind**(**db**.**nodes**[**4**].**unicast\_address**,** 1**,** mt**.**ModelId**(**0x1000**))** #bind to appkey1

In **[**57**]:** cc**.**model\_subscription\_add**(**db**.**nodes**[**4**].**unicast\_address**,** 0xc001**,** mt**.**ModelId**(**0x1000**))** #Add to group 0xC001

1. Switch on the sixth node and provision as before. It will follow the same pattern explained in step 10. Here is the summary of commands:

In **[**58**]:** p**.**scan\_start**()**

In **[**59**]:** p**.**scan\_stop**()**

In **[**60**]:** p**.**provision**(**name**=**"Light bulb #6"**)**

In **[**61**]:** cc**.**publish\_set**(**13**,** 5**)**

In **[**62**]:** cc**.**composition\_data\_get**()**

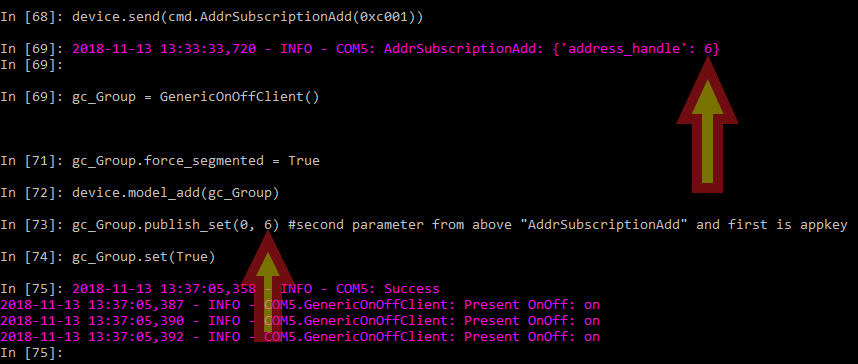
In **[**63**]:** cc**.**appkey\_add**(**0**)**

In **[**64**]:** cc**.**appkey\_add**(**1**)**

In **[**65**]:** cc**.**model\_app\_bind**(**db**.**nodes**[**5**].**unicast\_address**,** 1**,** mt**.**ModelId**(**0x1000**))** #bind to appkey1

In **[**67**]:** cc**.**model\_subscription\_add**(**db**.**nodes**[**5**].**unicast\_address**,** 0xc001**,** mt**.**ModelId**(**0x1000**))** #Add to group 0xC001

1. Now we will test the mesh network to see if it is working. To do this, we will send a group message out bound to Application Key 0 to turn on nodes 1, 2, and 3. The message will go to all the nodes that we have added to the group address 0xC001 but only the nodes with the correct application key will respond by turning on their LED:



As seen above, we use the address\_handle for the group message to pass to publish\_set function as its second parameter and the application key to bind to as its first parameter (Application Key 0, in this case).

Summary of commands and output for this step:

In **[**68**]:** device**.**send**(**cmd**.**AddrSubscriptionAdd**(**0xc001**))**

In **[**69**]:** 2018**-**11**-**13 13**:**33**:**33**,**720 **-** INFO **-** COM5**:** AddrSubscriptionAdd**:** **{**'address\_handle'**:** 6**}**

In **[**69**]:**

In **[**69**]:** gc\_Group **=** GenericOnOffClient**()**

In **[**71**]:** gc\_Group**.**force\_segmented **=** **True**

In **[**72**]:** device**.**model\_add**(**gc\_Group**)**

In **[**73**]:** gc\_Group**.**publish\_set**(**0**,** 6**)** #second parameter from above "AddrSubscriptionAdd" and first is appkey

In **[**74**]:** gc\_Group**.**set**(True)**

In **[**75**]:** 2018**-**11**-**13 13**:**37**:**05**,**358 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 13**:**37**:**05**,**387 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

2018**-**11**-**13 13**:**37**:**05**,**390 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

2018**-**11**-**13 13**:**37**:**05**,**392 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

1. Observe the LED on nodes bound to Application Key 0 to light up:



1. Turn off the LEDs on Application Key 0 nodes by issuing command:

In **[**75**]:** gc\_Group**.**set**(False)**

In **[**76**]:** 2018**-**11**-**13 14**:**11**:**13**,**148 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 14**:**11**:**13**,**170 **-** INFO **-** COM5**.**GenericOnOffClient**:** off

2018**-**11**-**13 14**:**11**:**13**,**179 **-** INFO **-** COM5**.**GenericOnOffClient**:** off

2018**-**11**-**13 14**:**11**:**13**,**227 **-** INFO **-** COM5**.**GenericOnOffClient**:** off

1. Now test nodes bound with Application Key 1 to see if they light up properly:

In **[**76**]:** gc\_Group**.**publish\_set**(**1**,** 6**)** #second parameter from above "AddrSubscriptionAdd" and first is appkey

In **[**77**]:** gc\_Group**.**set**(True)**

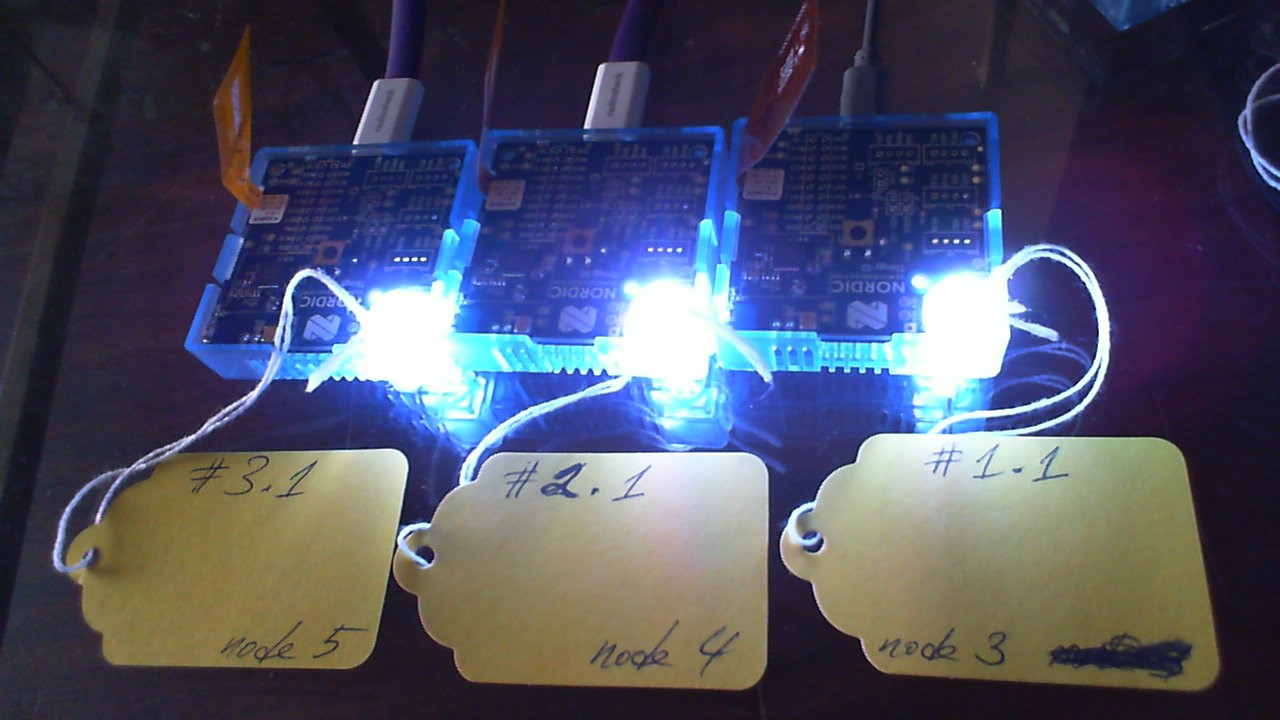
In **[**78**]:** 2018**-**11**-**13 14**:**15**:**10**,**181 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 14**:**15**:**10**,**203 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

2018**-**11**-**13 14**:**15**:**10**,**206 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

2018**-**11**-**13 14**:**15**:**10**,**230 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

1. Observe the LED on nodes bound to Application Key 1 to light up:



1. Turn off the LEDs on Application Key 1 nodes by issuing command:

In **[**78**]:** gc\_Group**.**set**(False)**

1. Now we shall look at “moving” one node from one group to another. Recall that we have provisioned each node with both Application Key 0 and Application Key 1. This means that both application keys are available to each node but we “bind” each node with either Application Key 0 or Application Key 1. We will unbind a node from Application Key 1 to Application Key 0:

****

Command above will point the Configuration Client object (cc) to node 3 because recall that 11 is its device\_key and 3 is its address\_handle and we want to perform a configuration operation on that node. Then we unbind node 3 from Application Key 1 to Application Key 0 thereby allowing the node to be part of Application Key 0 Group. Here are the commands and the outputs:

In **[**81**]:** cc**.**publish\_set**(**11**,** 3**)**

In **[**83**]:** cc**.**model\_app\_unbind**(**db**.**nodes**[**3**].**unicast\_address**,** 1**,** mt**.**ModelId**(**0x1000**))**

In **[**84**]:** 2018**-**11**-**13 16**:**39**:**23**,**402 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 16**:**39**:**23**,**450 **-** INFO **-** COM5**.**ConfigurationClient**:** Model app bind status**:** AccessStatus**.**SUCCESS

2018**-**11**-**13 16**:**39**:**23**,**455 **-** INFO **-** COM5**.**ConfigurationClient**:** Appkey unbind 1 to model 1000 at 0019

In **[**84**]:**

In **[**84**]:** cc**.**model\_app\_bind**(**db**.**nodes**[**3**].**unicast\_address**,** 0**,** mt**.**ModelId**(**0x1000**))**

In **[**85**]:** 2018**-**11**-**13 16**:**39**:**37**,**512 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 16**:**39**:**37**,**547 **-** INFO **-** COM5**.**ConfigurationClient**:** Model app bind status**:** AccessStatus**.**SUCCESS

2018**-**11**-**13 16**:**39**:**37**,**552 **-** INFO **-** COM5**.**ConfigurationClient**:** Appkey bind 0 to model 1000 at 0019

1. Now we will send a group message to illuminate nodes with Application Key 1:

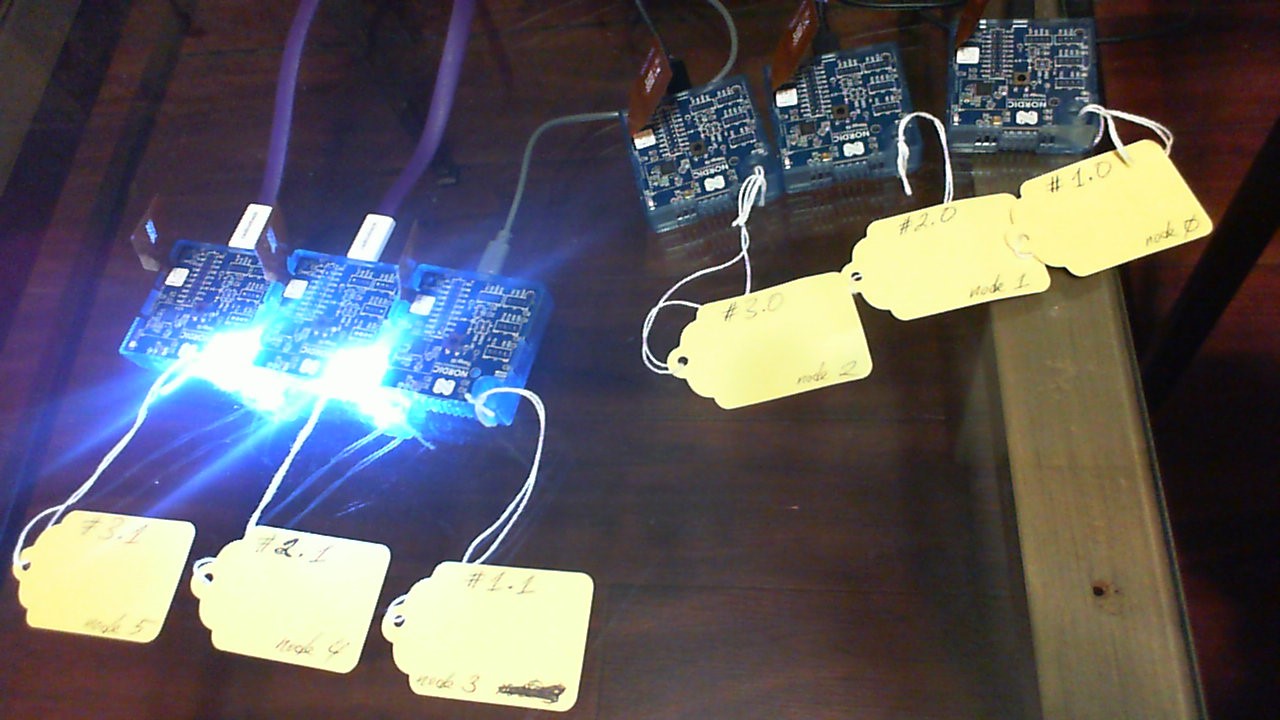
In **[**85**]:** gc\_Group**.**set**(True)**

In **[**86**]:** 2018**-**11**-**13 16**:**39**:**54**,**319 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 16**:**39**:**54**,**333 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

2018**-**11**-**13 16**:**39**:**54**,**340 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

Here we can see that node 3 is no longer responding to the command we sent to illuminate its LED because we have changed its binding from Application Key 1 to Application Key 0:



1. Now we will switch off Application Key 1 nodes and switch on Application Key 0 nodes:

In **[**88**]:** gc\_Group**.**set**(False)** #Turn off AppKey1 nodes’ LED

In **[**89**]:** 2018**-**11**-**13 16**:**42**:**31**,**181 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 16**:**42**:**31**,**202 **-** INFO **-** COM5**.**GenericOnOffClient**:** off

2018**-**11**-**13 16**:**42**:**31**,**207 **-** INFO **-** COM5**.**GenericOnOffClient**:** off

2018**-**11**-**13 16**:**42**:**31**,**209 **-** INFO **-** COM5**.**GenericOnOffClient**:** off

2018**-**11**-**13 16**:**42**:**31**,**254 **-** INFO **-** COM5**.**GenericOnOffClient**:** off

In **[**91**]:** gc\_Group**.**publish\_set**(**0**,** 6**)** #Switch Generic On/Off Client object to AppKey 0

In **[**92**]:** gc\_Group**.**set**(True)** #Turn on AppKey0 nodes’ LED

In **[**93**]:** 2018**-**11**-**13 17**:**36**:**37**,**824 **-** INFO **-** COM5**:** Success

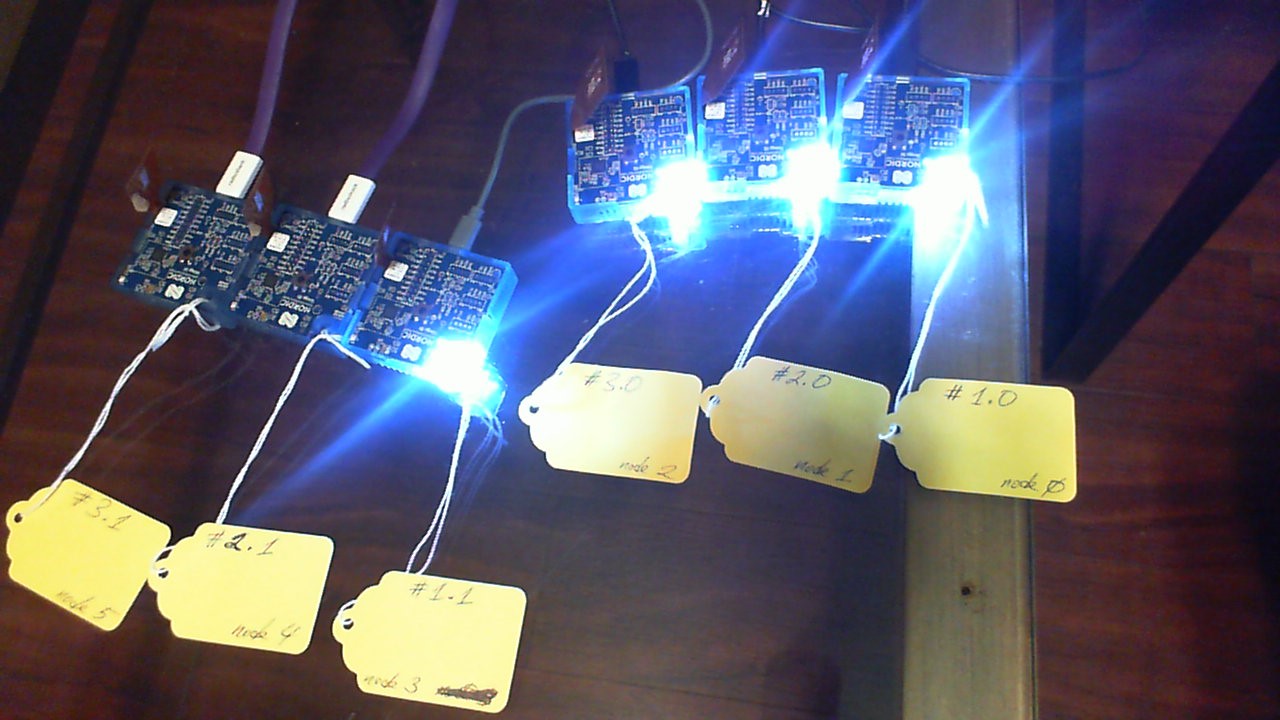
2018**-**11**-**13 17**:**36**:**37**,**856 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

2018**-**11**-**13 17**:**36**:**37**,**858 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

2018**-**11**-**13 17**:**36**:**37**,**867 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

2018**-**11**-**13 17**:**36**:**37**,**874 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

Observe now that node 3 is now grouped together with Application Key 0 Group:



1. Now we bind node 3 back to Application Key 1, as it was originally, and then turn on Application Key 0 Group:

In **[**94**]:** cc**.**model\_app\_unbind**(**db**.**nodes**[**3**].**unicast\_address**,** 0**,** mt**.**ModelId**(**0x1000**))** #unbind node 3 from Appkey0

2018**-**11**-**13 18**:**00**:**09**,**068 **-** INFO **-** COM5**:** Success

In **[**95**]:** 2018**-**11**-**13 18**:**00**:**09**,**132 **-** INFO **-** COM5**.**ConfigurationClient**:** Model app bind status**:** AccessStatus**.**SUCCESS

2018**-**11**-**13 18**:**00**:**09**,**139 **-** INFO **-** COM5**.**ConfigurationClient**:** Appkey unbind 0 to model 1000 at 0019

In **[**95**]:**

In **[**95**]:** cc**.**model\_app\_bind**(**db**.**nodes**[**3**].**unicast\_address**,** 1**,** mt**.**ModelId**(**0x1000**))** #bind node 3 to Appkey1

In **[**96**]:** 2018**-**11**-**13 18**:**00**:**29**,**008 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 18**:**00**:**29**,**072 **-** INFO **-** COM5**.**ConfigurationClient**:** Model app bind status**:** AccessStatus**.**SUCCESS

2018**-**11**-**13 18**:**00**:**29**,**077 **-** INFO **-** COM5**.**ConfigurationClient**:** Appkey bind 1 to model 1000 at 0019

In **[**96**]:** gc\_Group**.**set**(True)** #Illuminate Appkey0 Group

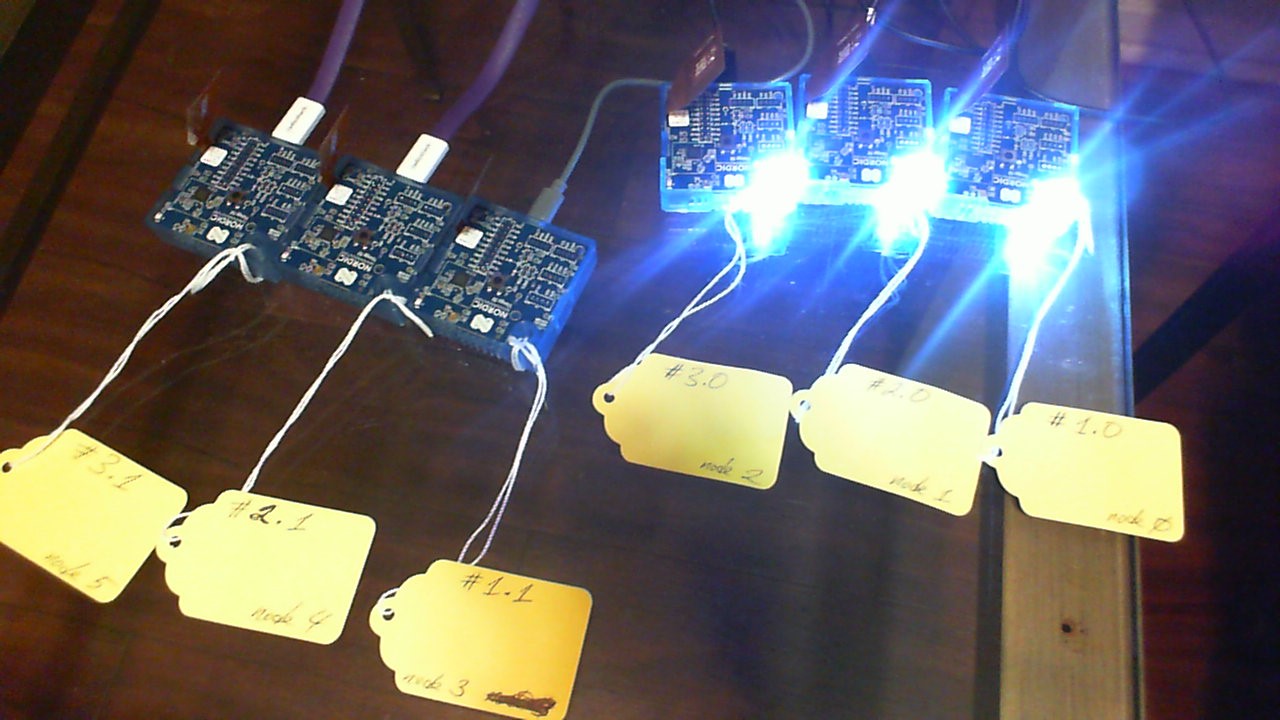
In **[**97**]:** 2018**-**11**-**13 18**:**00**:**38**,**334 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 18**:**00**:**38**,**356 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

2018**-**11**-**13 18**:**00**:**38**,**358 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

2018**-**11**-**13 18**:**00**:**38**,**361 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

As a result, we see that now node 3 is no longer illuminated because we have bound it back to Application Key 1:



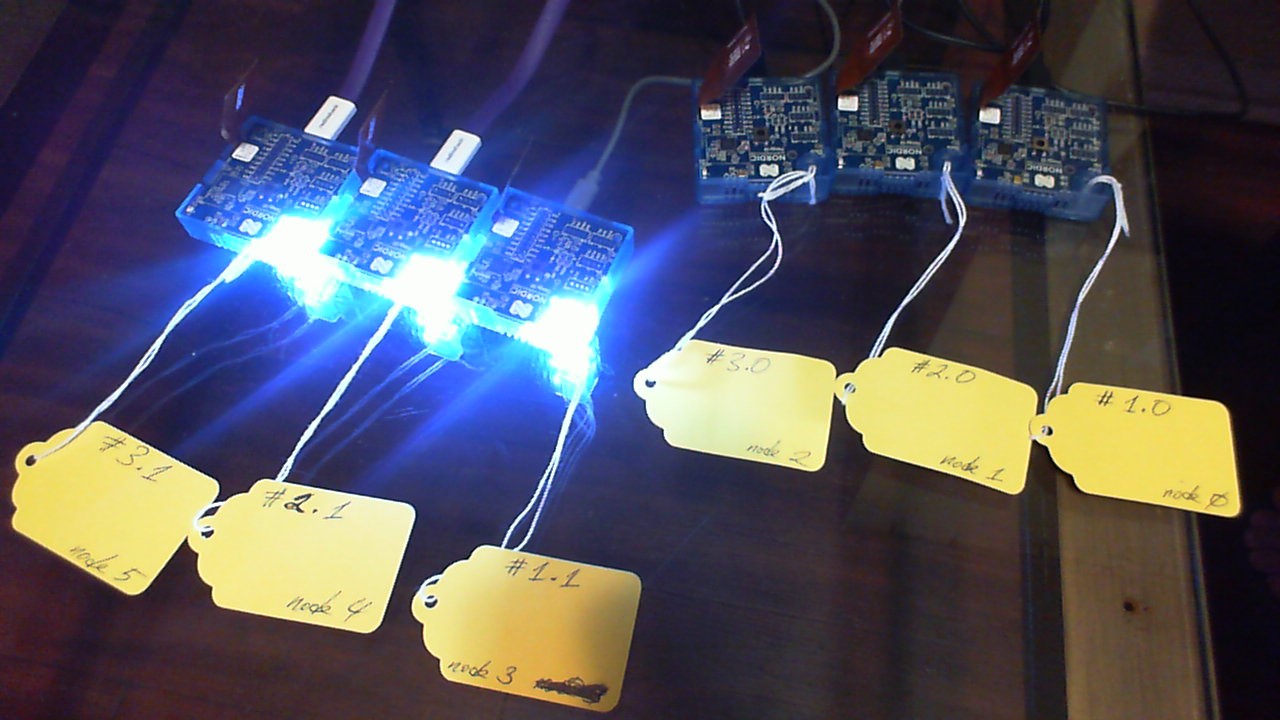
1. Now we turn off Application Key 0 Group LEDs and turn on Application Key 1 Group LEDs to show that node 3 is now back to being part of Application 0 Group:

In **[**97**]:** gc\_Group**.**set**(False)** #Turn off AppKey0 Group LEDs

In **[**98**]:** gc\_Group**.**publish\_set**(**1**,**6**)** #Set address to AppKey1 Group

In **[**99**]:** gc\_Group**.**set**(True)** #Turn on AppKey1 Group LEDs

As a result, we see that node 3 is back into the Application Key 1 Group:



1. Now we will demonstrate network access to one single node by illuminating its LED:

In **[**101**]:** device**.**send**(**cmd**.**AddrPublicationAdd**(**db**.**nodes**[**1**].**unicast\_address**))**

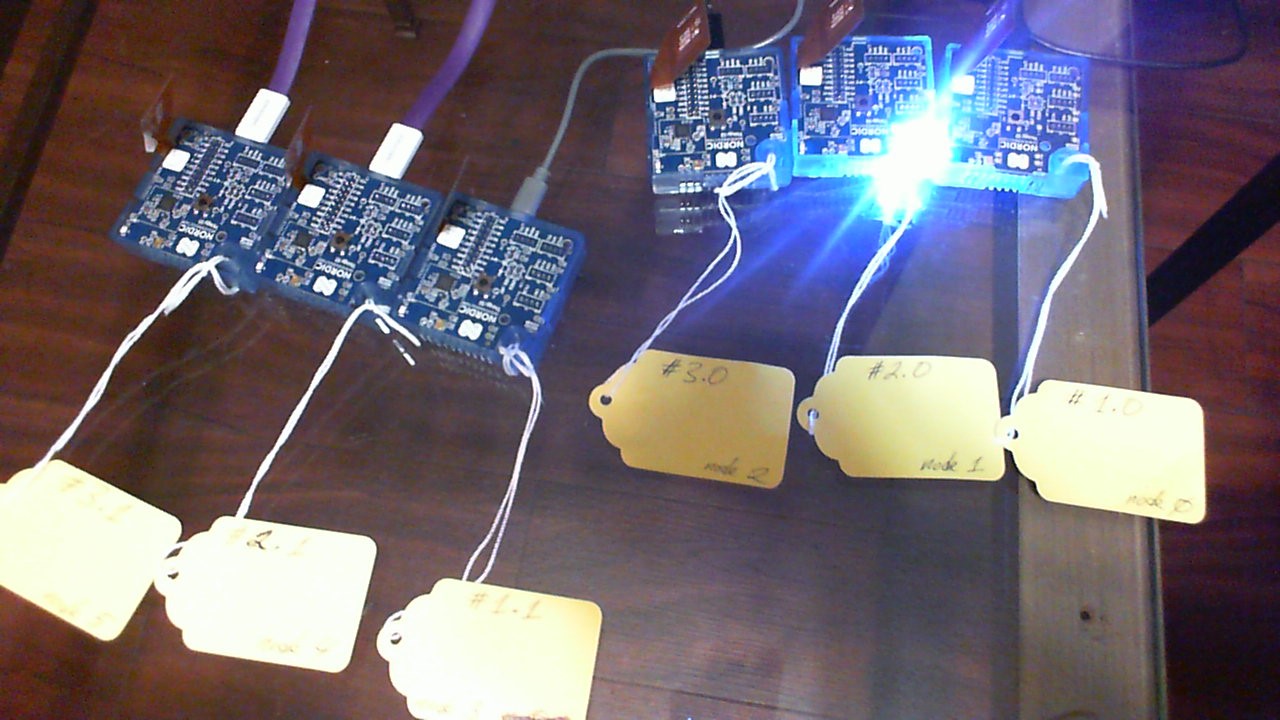
In **[**102**]:** 2018**-**11**-**13 19**:**14**:**47**,**860 **-** INFO **-** COM5**:** AddrPublicationAdd**:** **{**'address\_handle'**:** 1**}**

In **[**107**]:** gc\_Uni**.**publish\_set**(**0**,**1**)** #first is appkey index, second is from AddrPublicationAdd()

In **[**108**]:** gc\_Uni**.**set**(True)** #Turn on single node

In **[**109**]:** gc\_Uni**.**set**(False)** #Turn off single node

When switched on, we see only node 1 illuminated:



## Restoring a mesh network after reset of provisioner

### This section will explain how to restore a mesh network if the provisioner nRF52-DK is rebooted and/or power-cycled.

1. Please follow steps 2, 3, and 4 in the [Provisioning](#_Provisioning) section so that the PyACI shell is running on the PC, which is connected via USB to the nRF52-DK running the serial firmware.
2. Type the highlighted commands in PyACI (output shown is a sample):

In **[**1**]:** **import** restoring\_commands

In **[**2**]:** db **=** MeshDB**(**"database/example\_database.json"**)**

In **[**3**]:** db**.**load**()**

In **[**4**]:** restoring\_commands**.**database\_restore**(**db**,** device**)**

In **[**5**]:** 2018**-**11**-**13 22**:**30**:**52**,**154 **-** INFO **-** COM5**:** Success

2018**-**11**-**13 22**:**30**:**52**,**160 **-** INFO **-** COM5**:** SubnetAdd**:** **{**'subnet\_handle'**:** 0**}**

2018**-**11**-**13 22**:**30**:**52**,**165 **-** INFO **-** COM5**:** AppkeyAdd**:** **{**'appkey\_handle'**:** 0**}**

2018**-**11**-**13 22**:**30**:**52**,**170 **-** INFO **-** COM5**:** AppkeyAdd**:** **{**'appkey\_handle'**:** 1**}**

In **[**5**]:**

In **[**5**]:** restoring\_commands**.**provisioned\_device\_data\_restore**(**db**,** device**)**

2018**-**11**-**13 22**:**31**:**06**,**390 **-** INFO **-** COM5**:** DevkeyAdd**:** **{**'devkey\_handle'**:** 8**}**

2018**-**11**-**13 22**:**31**:**06**,**393 **-** INFO **-** COM5**:** AddrPublicationAdd**:** **{**'address\_handle'**:** 0**}**In **[**6**]:**

2018**-**11**-**13 22**:**31**:**06**,**406 **-** INFO **-** COM5**:** DevkeyAdd**:** **{**'devkey\_handle'**:** 9**}**

2018**-**11**-**13 22**:**31**:**06**,**408 **-** INFO **-** COM5**:** AddrPublicationAdd**:** **{**'address\_handle'**:** 1**}**

2018**-**11**-**13 22**:**31**:**06**,**420 **-** INFO **-** COM5**:** DevkeyAdd**:** **{**'devkey\_handle'**:** 10**}**

2018**-**11**-**13 22**:**31**:**06**,**422 **-** INFO **-** COM5**:** AddrPublicationAdd**:** **{**'address\_handle'**:** 2**}**

2018**-**11**-**13 22**:**31**:**06**,**432 **-** INFO **-** COM5**:** DevkeyAdd**:** **{**'devkey\_handle'**:** 11**}**

2018**-**11**-**13 22**:**31**:**06**,**445 **-** INFO **-** COM5**:** AddrPublicationAdd**:** **{**'address\_handle'**:** 3**}**

2018**-**11**-**13 22**:**31**:**06**,**449 **-** INFO **-** COM5**:** DevkeyAdd**:** **{**'devkey\_handle'**:** 12**}**

2018**-**11**-**13 22**:**31**:**06**,**458 **-** INFO **-** COM5**:** AddrPublicationAdd**:** **{**'address\_handle'**:** 4**}**

2018**-**11**-**13 22**:**31**:**06**,**462 **-** INFO **-** COM5**:** DevkeyAdd**:** **{**'devkey\_handle'**:** 13**}**

2018**-**11**-**13 22**:**31**:**06**,**472 **-** INFO **-** COM5**:** AddrPublicationAdd**:** **{**'address\_handle'**:** 5**}**

1. Test mesh network access by illuminating all nodes. Type in the following commands in PyACI:

In **[**6**]:** gc\_Group **=** GenericOnOffClient**()**

In **[**7**]:** gc\_Group**.**force\_segmented **=** **True**

In **[**8**]:** device**.**model\_add**(**gc\_Group**)**

In **[**9**]:** device**.**send**(**cmd**.**AddrPublicationAdd**(**address**=**0xC001**))**

In **[**10**]:** 2018**-**11**-**13 22**:**46**:**41**,**276 **-** INFO **-** COM5**:** AddrPublicationAdd**:** **{**'address\_handle'**:** 6**}**

In **[**10**]:**

In **[**10**]:** gc\_Group**.**publish\_set**(**0**,** 6**)**

In **[**12**]:** gc\_Group**.**set**(True)**

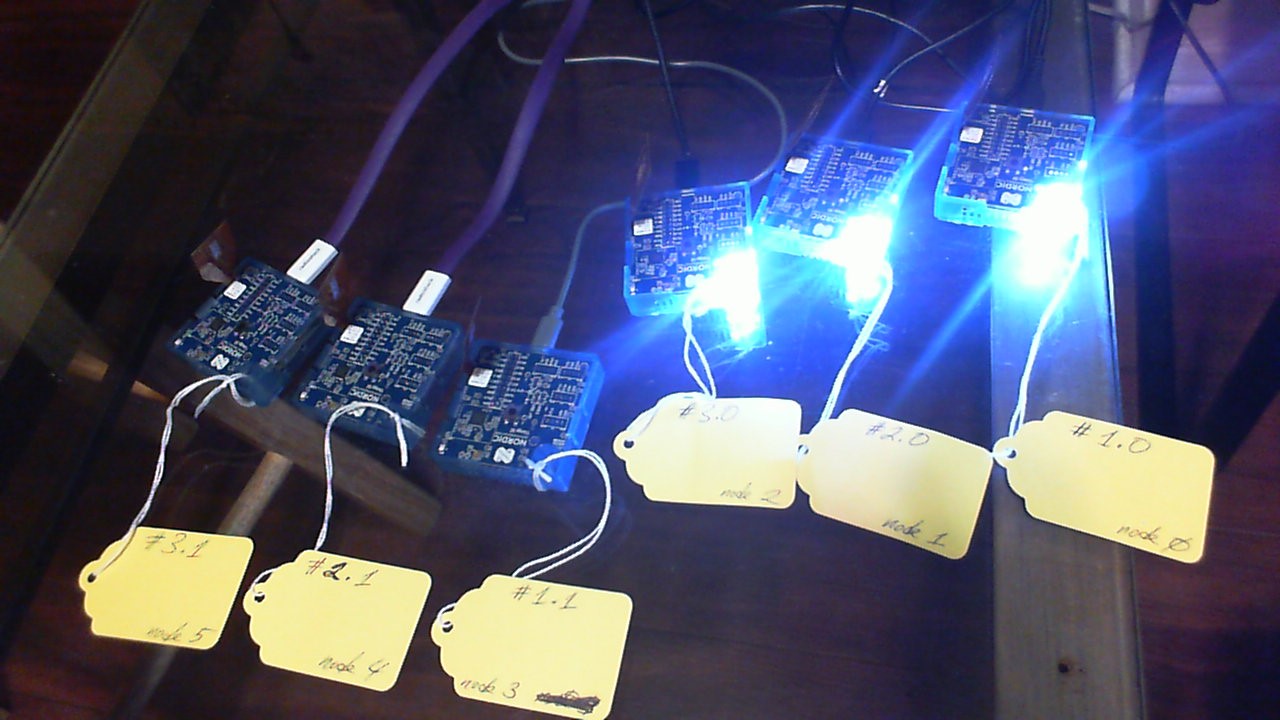
2018**-**11**-**13 22**:**48**:**38**,**886 **-** INFO **-** COM5**:** Success

In **[**13**]:** 2018**-**11**-**13 22**:**48**:**38**,**923 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

2018**-**11**-**13 22**:**48**:**38**,**928 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

2018**-**11**-**13 22**:**48**:**38**,**932 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

This should illuminate Application Key 0 Group:



1. Now illuminate Application Key 1 Group:

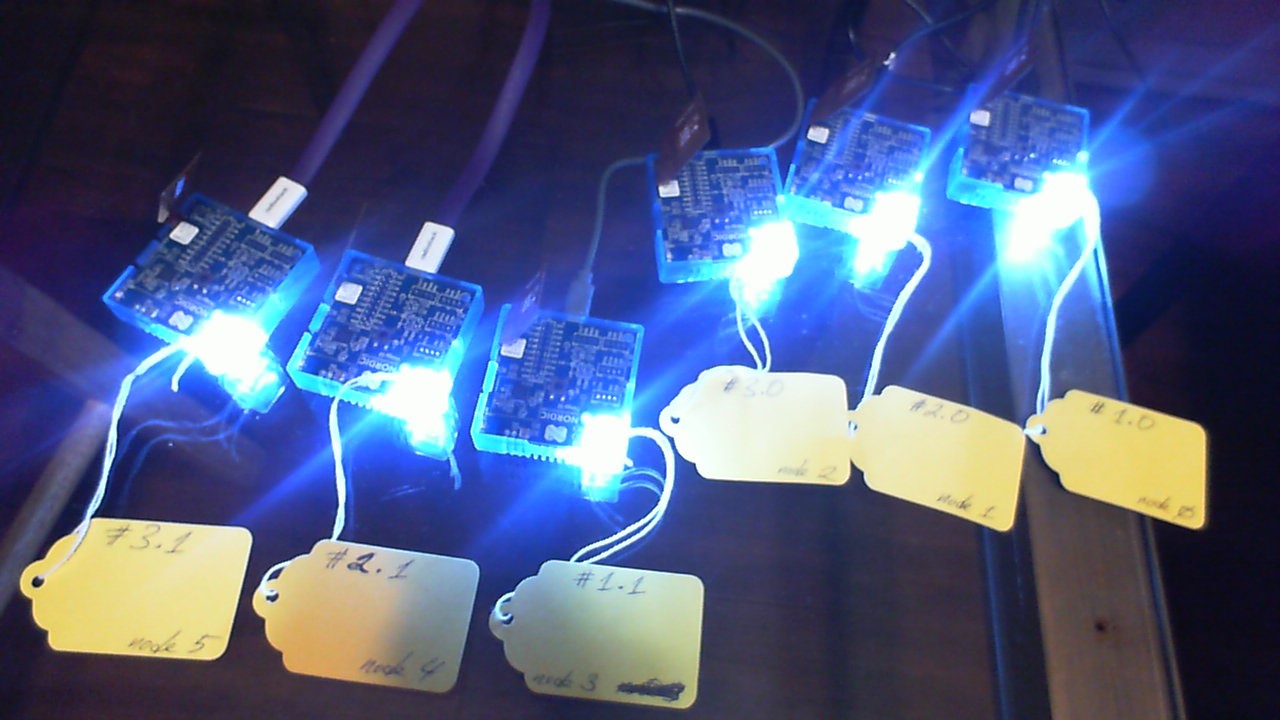
In **[**13**]:** gc\_Group**.**publish\_set**(**1**,** 6**)**

In **[**14**]:** gc\_Group**.**set**(True)**

2018**-**11**-**13 22**:**55**:**19**,**212 **-** INFO **-** COM5**:** Success

In **[**15**]:** 2018**-**11**-**13 22**:**55**:**19**,**245 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on

2018**-**11**-**13 22**:**55**:**19**,**257 **-** INFO **-** COM5**.**GenericOnOffClient**:** Present OnOff**:** on  
Now we see all the nodes illuminated:

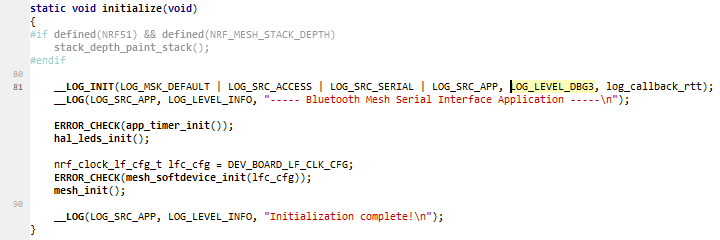


This shows we have mesh network access after Provisioner reboot.

# Profiling

## Profiling allows one to measure the message latency in the mesh network.

1. To enable profiling, enable debug logging in the Provisioner firmware referred to in the [Provisioning](#_Provisioning_1) section, step 2. This is done by editing main.c in the SES project and enabling debug logging (highlighted):



1. Launch the Segger RTT Client to see log output.
2. Compile and run the firmware on the DK in Debug mode.
3. Send a message to the mesh network (see Provisioning section for sending messages). For instance:

In **[**103**]:** gc\_Uni **=** GenericOnOffClient**()**

In **[**104**]:** gc\_Uni**.**force\_segmented **=** **True**

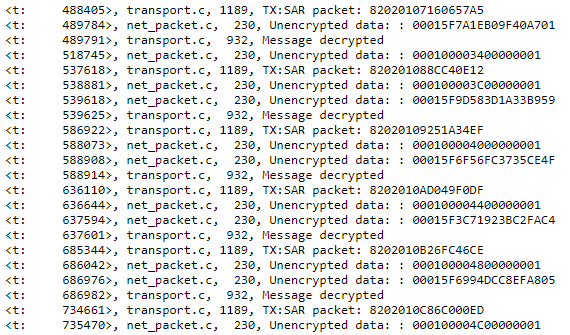
In **[**105**]:** device**.**model\_add**(**gc\_Uni**)**

In **[**107**]:** gc\_Uni**.**publish\_set**(**0**,**1**)**

In **[**108**]:** gc\_Uni**.**set**(True)**

Note that when we set force\_segmented to True, messages are sent with ACK requested whereas setting for\_segmented to False results in messages sent without an ACK requested.

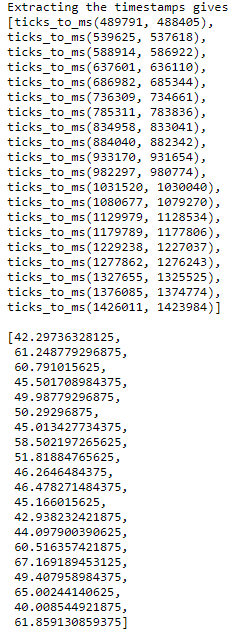
1. Observe RTT output with RTC tick count on the left:



1. For transforming ticks displayed in the RTT output to milliseconds, use the following Python function:

ticks\_to\_ms = lambda a, b: abs(a-b) \* 1000000 / 32768 / 1000

1. Extracting the timestamps will give output:



The last list are times in milliseconds.