



SILICON LABS



embeddedworld

Matter Workshop: Matter-over-Thread in Simplicity Studio

Martin Looker

15th March 2023, 10:00 - 12:30 (GMT+1)

NCC Ost, Floor 3, Room Neu-Delhi



The Leader in IoT Wireless Connectivity

ember

2012

Software ZigBee SoC

ENERGY^{micro}

2013

Low-power 32-bit
MCUs

blue giga

2015

BT Smart Modules

telegesis

2015

ZigBee/Thread
Modules

Micrium[®]

2016

Software RTOS

ZENTRI

2017

Cloud Connected Wi-Fi®

Z-WAVE

2018

Smart Home Protocol

**REDPINE
SIGNALS**

2020

Ultra Low Power Wi-Fi®

100%
IoT Focused



#1

Share in Mesh



1st

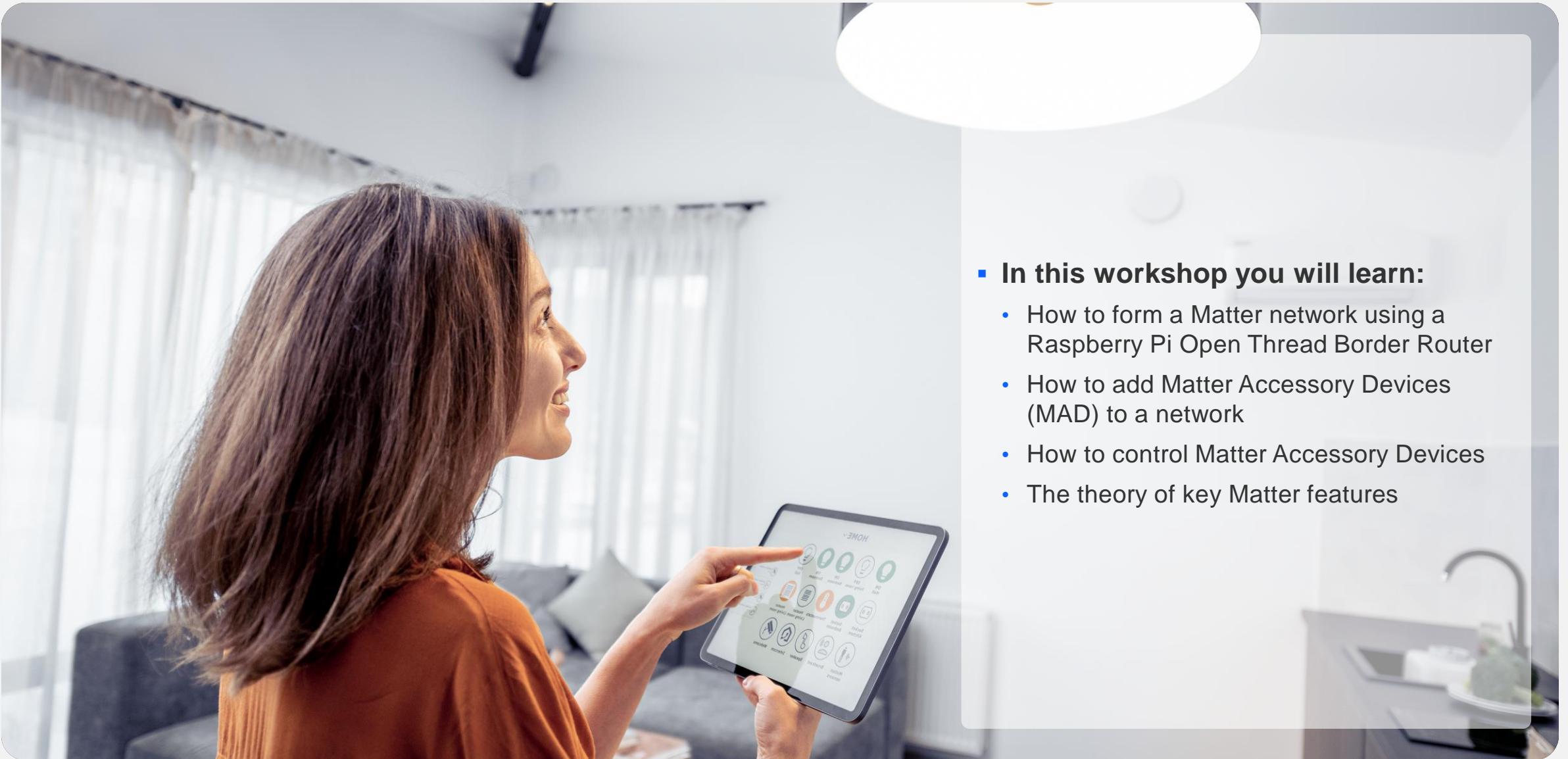
To Market with
Multiprotocol, Bluetooth LE,
Bluetooth mesh



Innovation

Performance, Power,
CoEx, Modules,
Secure Vault™

Overview

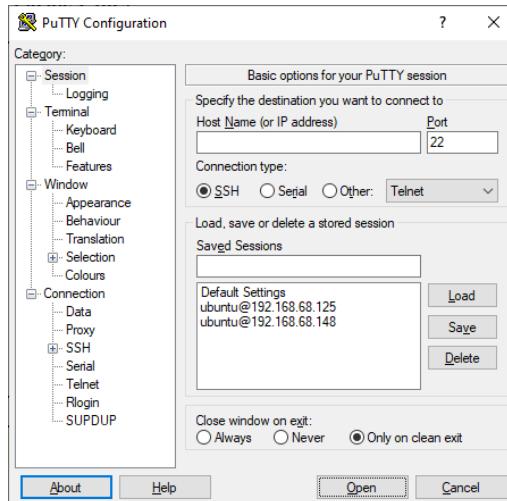


- **In this workshop you will learn:**

- How to form a Matter network using a Raspberry Pi Open Thread Border Router
- How to add Matter Accessory Devices (MAD) to a network
- How to control Matter Accessory Devices
- The theory of key Matter features

Pre-requisites – Software

- Simplicity Studio v5 installed and ready to use
 - Part 1 of this series shows how to download and install Simplicity Studio:
<https://www.brainshark.com/siliconlabs/EW23-Simplicity-Studio-Install>
- PuTTY or other SSH application
 - Download from:
<https://www.chiark.greenend.org.uk/~sgtatham/putty/>



Simplicity Studio is the unified development environment for all Silicon Labs technologies, SoCs, and modules. It provides you with access to the target device-specific web and SDK resources, software and hardware configuration tools, and an integrated development environment (IDE) featuring industry-standard code editors, compilers, and debuggers. With Simplicity Studio, you get a complete set of advanced value-add tools for network analysis and code-correlated energy profiling.

No matter your experience level, Simplicity Studio takes you through an optimized workflow, enabling quicker project progression, device configuration, and application optimization. Simplicity Studio 5 is built on Eclipse and C/C++ Development Tooling (CDT), adding robustness, improving performance, and allowing you to customize your development experience using plugins from the Eclipse Marketplace.

Simplicity Studio version 5 supports Silicon Labs Secure Vault, the most advanced security software suite with the highest PSA Certification Level 3. With Secure Vault, you can protect your IoT devices against escalating threats while conforming to the quickly evolving cyber-security regulations. The IDE also includes a UI engine for modern, responsive, web-like user interfaces.

Download the Full Online Installer Version of Simplicity Studio 5

[Windows Installer >](#) [Mac Installer >](#) [Linux Installer >](#)

*SS 5 User Guide >

Looking for Release Notes? Visit our [Gecko SDK \(GSDK\)](#) page or the relevant technology SDK page and look under the Tech Docs tab.

Pre-requisites – Hardware (Individual)

- **Silicon Labs Wireless Kit**

- Programmed with:

**Bootloader – SoC Internal Storage
(single image on 1MB device)**

- Part 2 of this series provides an overview of Simplicity Studio and shows how to create, compile and program a bootloader:

[https://www.brainshark.com/siliconlabs/
EW23-Bootloader-SS](https://www.brainshark.com/siliconlabs/EW23-Bootloader-SS)



Pre-requisites – Hardware (Shared)

- **Ethernet / Wi-Fi Router**
- **Raspberry Pi 3 or 4**
 - Programmed with **Matter Hub Raspberry Pi Image**:
https://github.com/SiliconLabs/matter/blob/release_1.0.2-1.0/docs/silabs/general/ARTIFACTS.md
 - Connected to router with ethernet cable
 - Power supply
 - Instructions:
https://github.com/SiliconLabs/matter/blob/release_1.0.2-1.0/docs/silabs/thread/RASPI_IMG.md
- **Silicon Labs Wireless Board**
 - Programmed with **Bootloader Binary and Radio Co-Processor (RCP) Image**:
<https://github.com/SiliconLabs/matter/releases/tag/v1.0.2-1.0>
 - Connected to Raspberry Pi with USB cable
 - Instructions:
https://github.com/SiliconLabs/matter/blob/release_1.0.2-1.0/docs/silabs/thread/RCP.md



Current Landscape of IoT Industry

Consumers

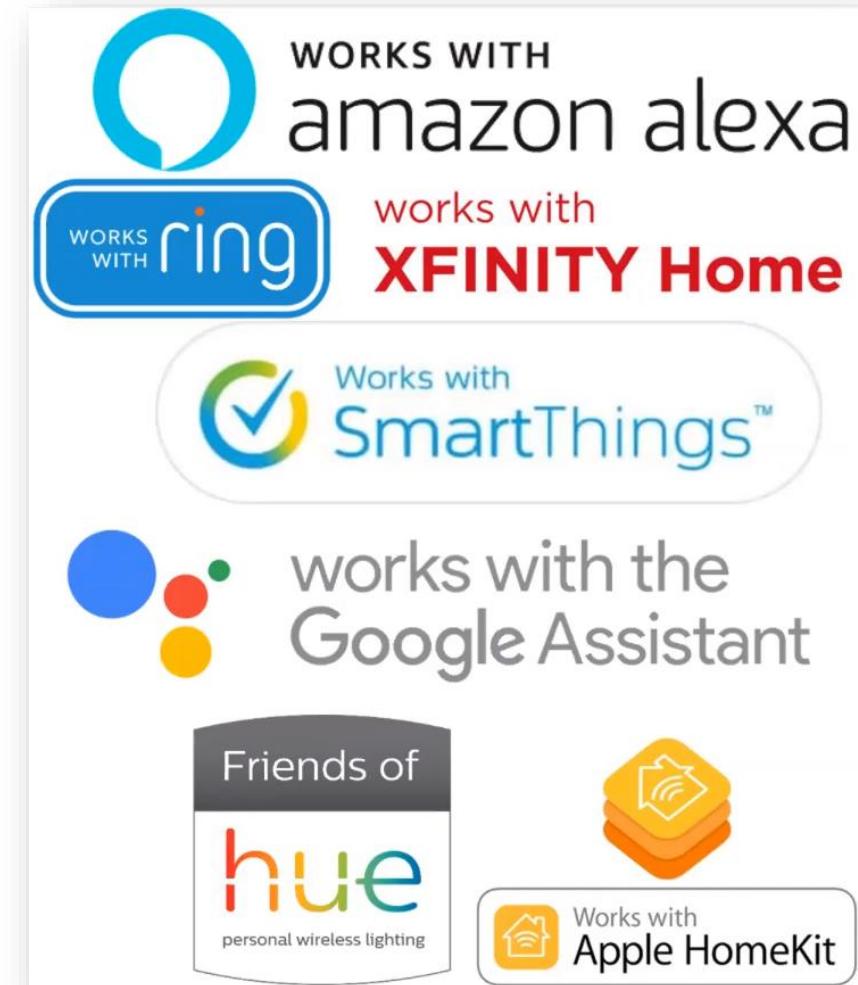
- Extremely hard to mix and match the product they want with their preferred ecosystem
- Very difficult to change once selected

Developers

- Developers are forced to pick what ecosystem integrations they support
- Often need to ship multiple SKUs for all connectivity standards
- Need to learn different IoT technologies and ecosystems

Retailers

- Too difficult to provide expert advice to answer consumer questions
- High return rates due to interoperability or incompatibility issues

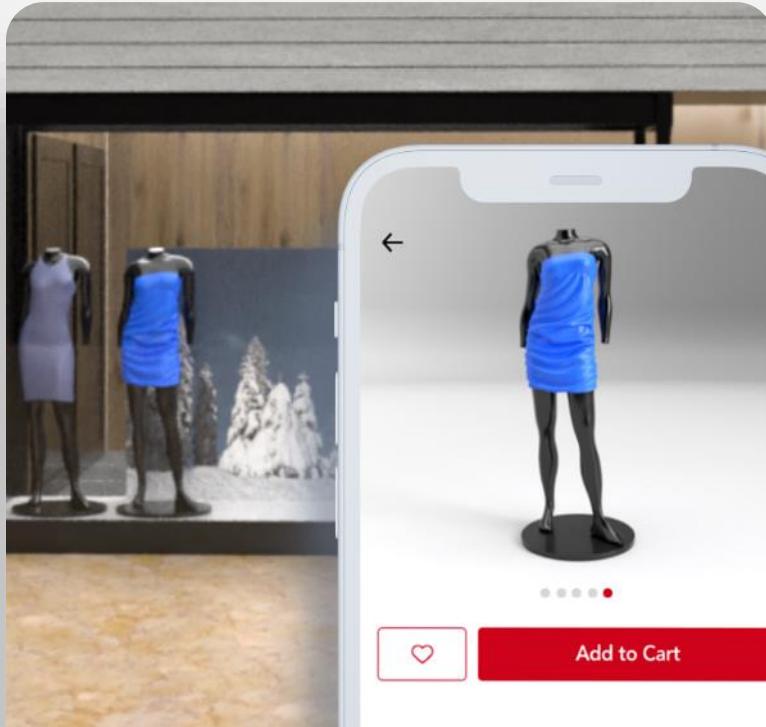


Simplifying IoT Connectivity



DEVELOPER/MANUFACTURER

- Single SKU**
- Lower development & operational cost**
- More time for innovation**



RETAILER

- Requires less shelf space**
- Lowers inventory cost**
- Minimizes returns**



CONSUMER

- Simplifies purchasing experience**
- Simplifies setup & control**
- Provides better user experience**

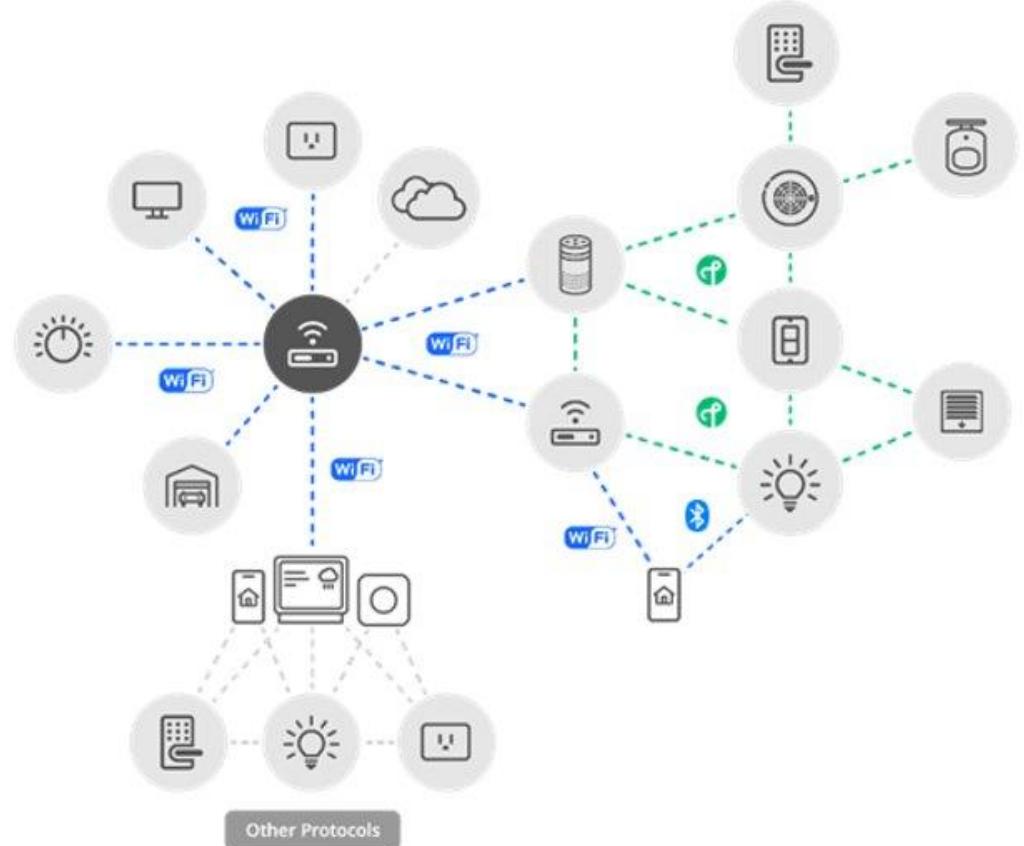
Network Topology

- **Matter wireless protocols:**

- Matter has native support for Thread and Wi-Fi
- Matter uses Bluetooth for commissioning
- Other protocols, like Zigbee and Z-Wave, can be bridged into a Matter network

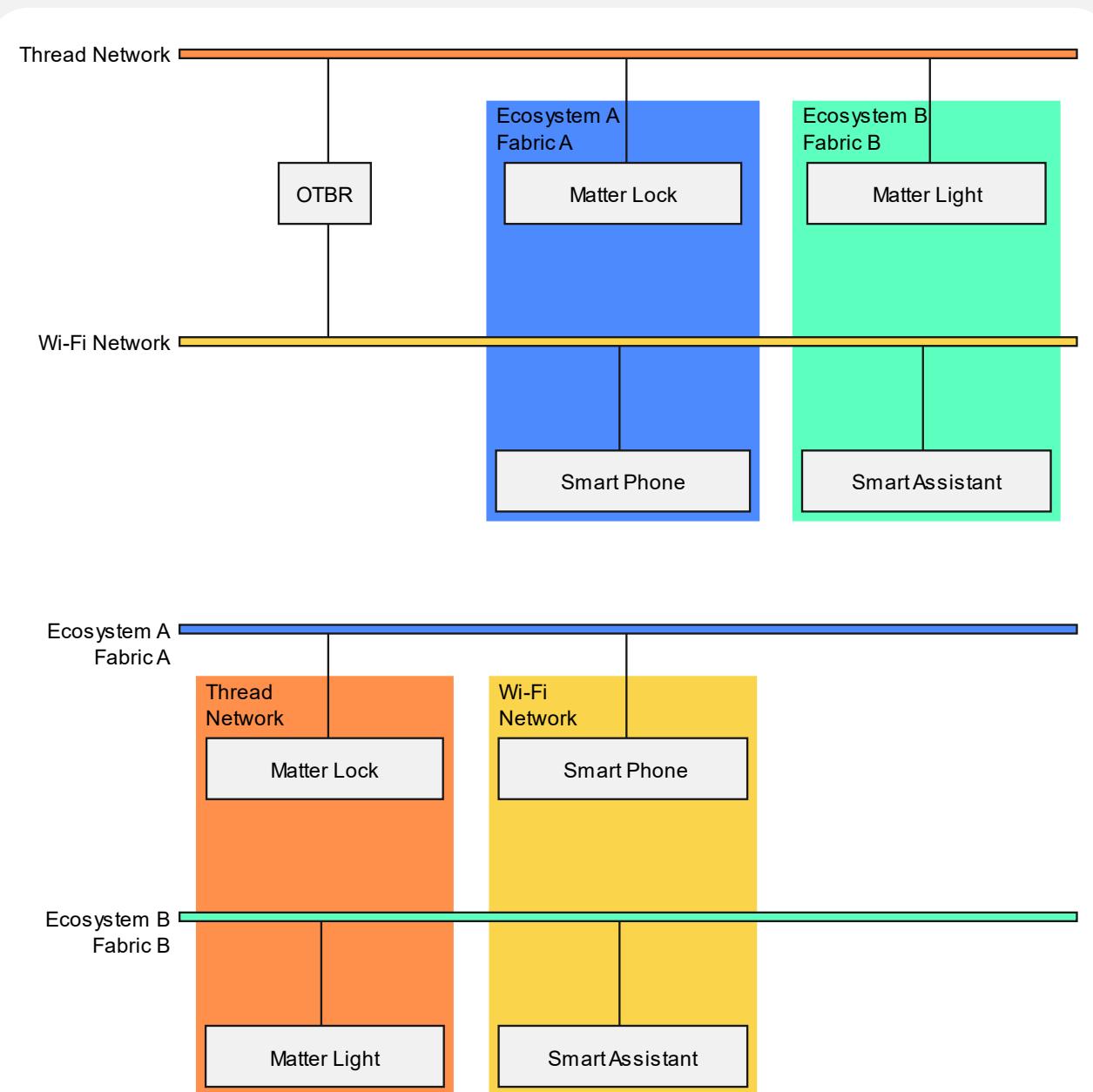
- **Matter device types:**

- Matter Accessory Devices (MADs) provide end-node functionality such as lights and switches
- OpenThread Border Routers (OTBRs) provide a connection between Thread devices and the local IP network
 - The Raspberry Pi and OpenThread Radio Co-Processor together form an OTBR
- Matter Controllers are used to control MADs, including commissioning them into the network
 - Typically, these are smart phones
 - The Raspberry Pi image includes a command-line Matter Controller
- Bridges can be used to link to other protocols, like ZigBee and Z-Wave



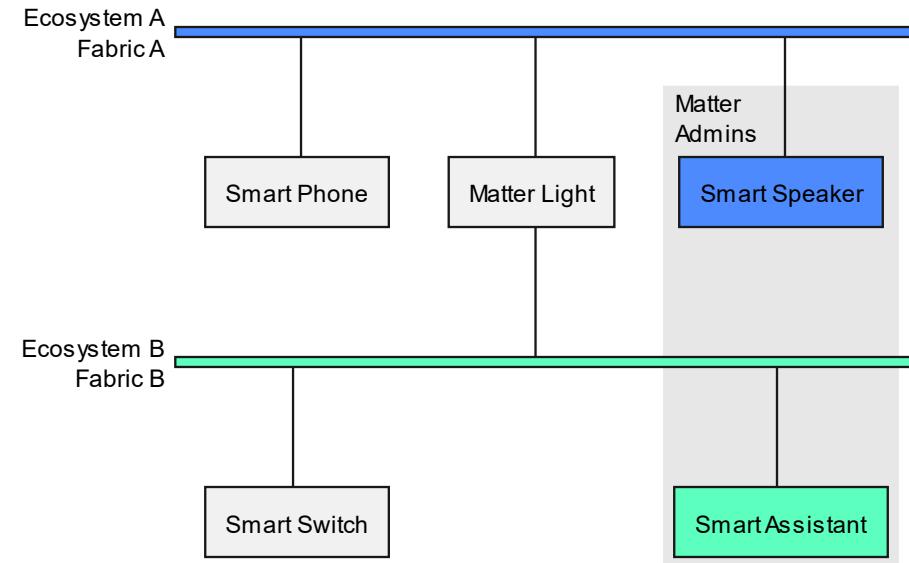
Matter Fabric

- Multiple transports
 - Matter can work on top of multiple wireless or wired technologies to transport the IP packets
- Fabric
 - A collection of Matter devices sharing a trusted root
 - A fabric is identified by a fabric ID which is a 64-bit number
- Node
 - In a Matter fabric, each physical device is called a node
 - Each node is identified by a node ID which is a 64-bit number



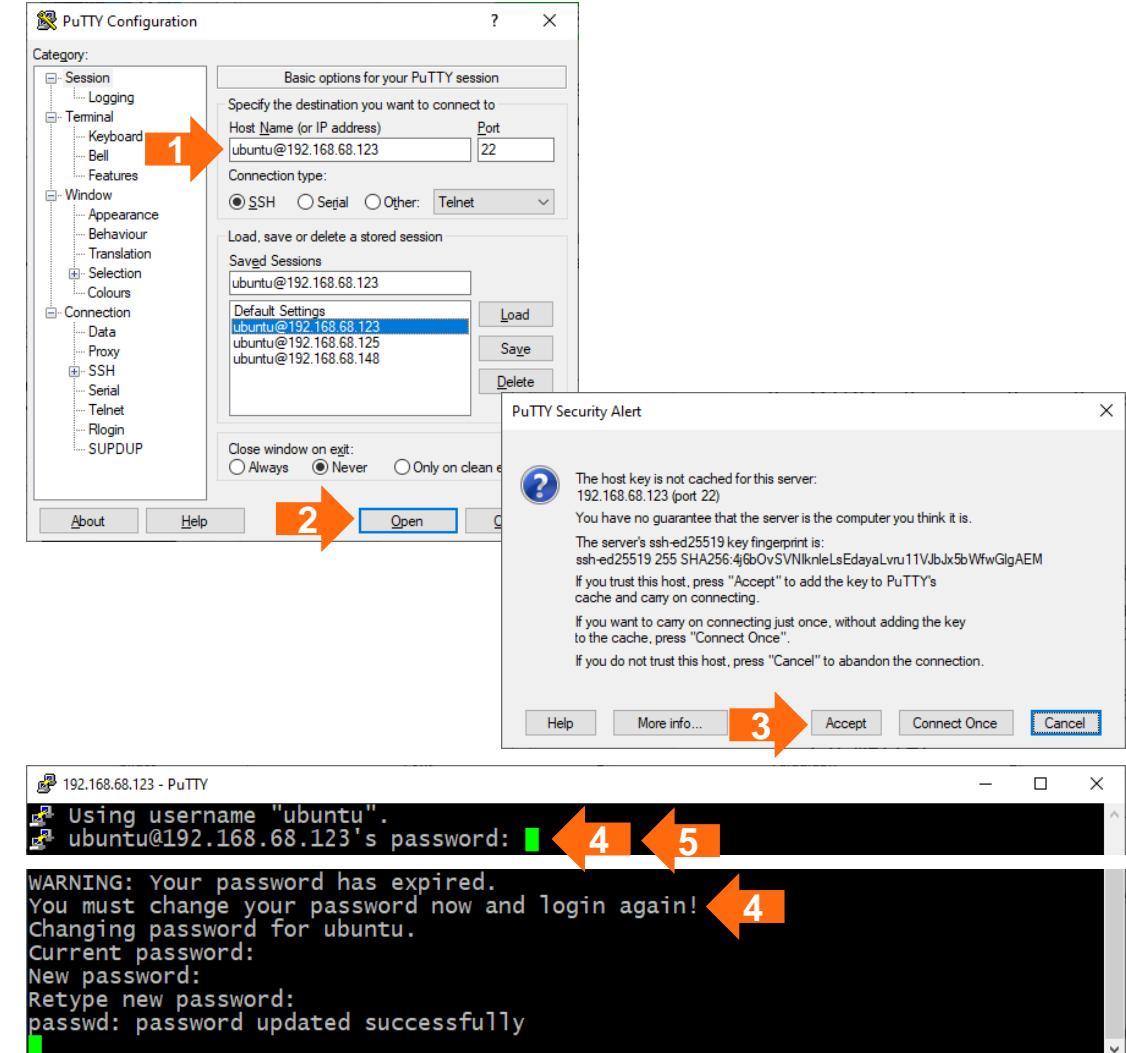
Matter Multi-Admin

- Provides a means for multiple Matter Fabrics and their administrators to manage devices
- Each Matter Fabric can have unique root authority
- Devices must support multiple Matter Admins
- Matter admins dictate the access control lists for their Matter fabric, and thus the devices can access the device
- **Example:**
 - Smart Speaker, Admin for Fabric A, can grant control privileges to Smart Phone on Fabric A
 - Smart Assistant, Admin for Fabric B, can grant control privileges to Smart Switch on Fabric B
- **Access Control is separate for both fabrics**



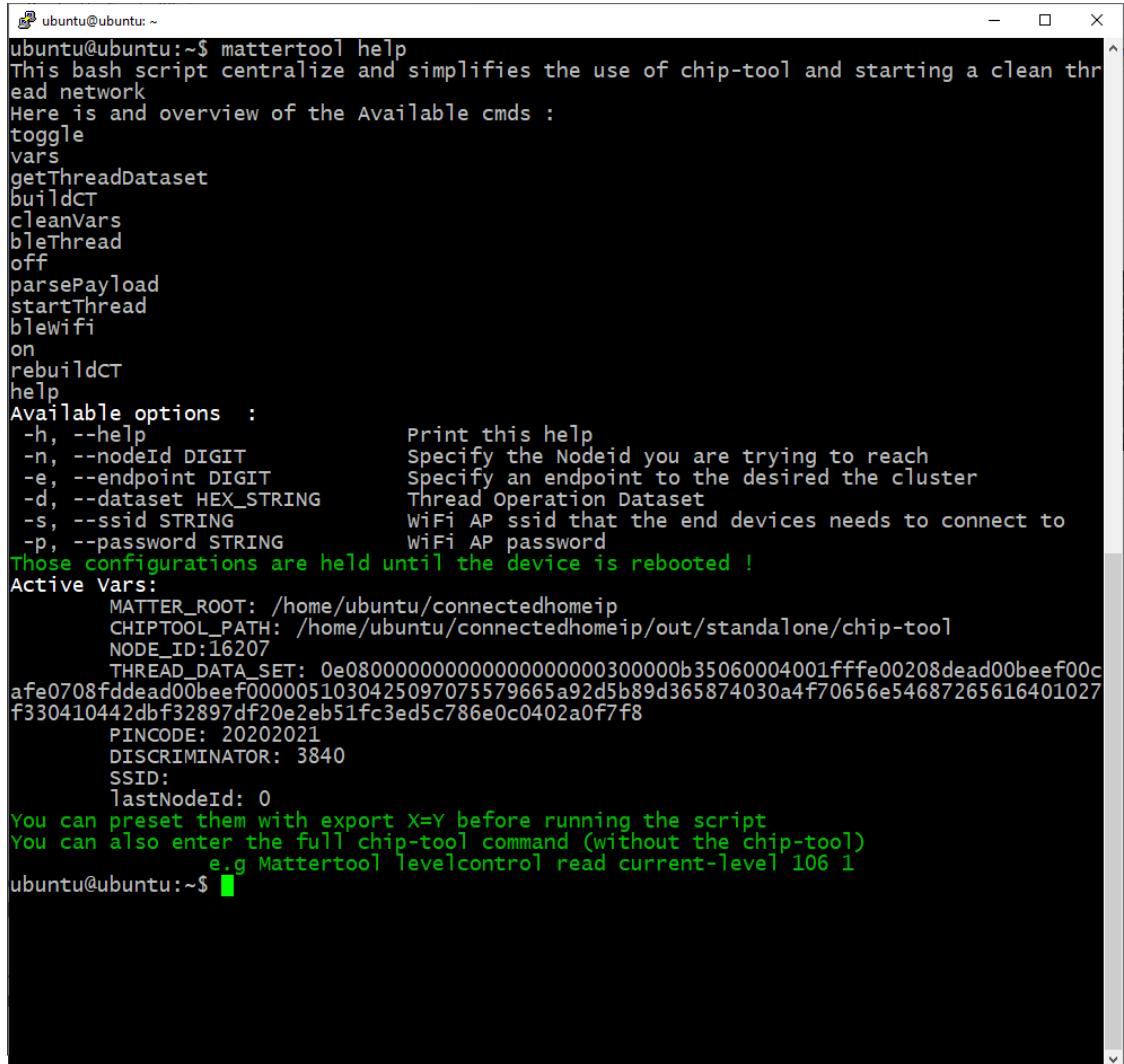
SSH / PuTTY

- The devices will be manipulated via a SSH connection to the OTBR using PuTTY or the command-line
- To open a connection using PuTTY:
 - Enter `ubuntu@192.168.68.123` in the Host Name (or IP address) edit box where:
`ubuntu` is the username for the connection
`192.168.68.123` is the IP address of the OTBR
 - Click the **Open** button
 - The first time connecting to a device using PuTTY, click the **Accept** button on the **PuTTY Security Alert** window
 - The first time the OTBR is connected to, enter the password `ubuntu`, you will then be prompted to select a new password, we will use `matter`
 - If you changed the password, reconnect and use the new password, `matter`



Mattertool and Discriminator

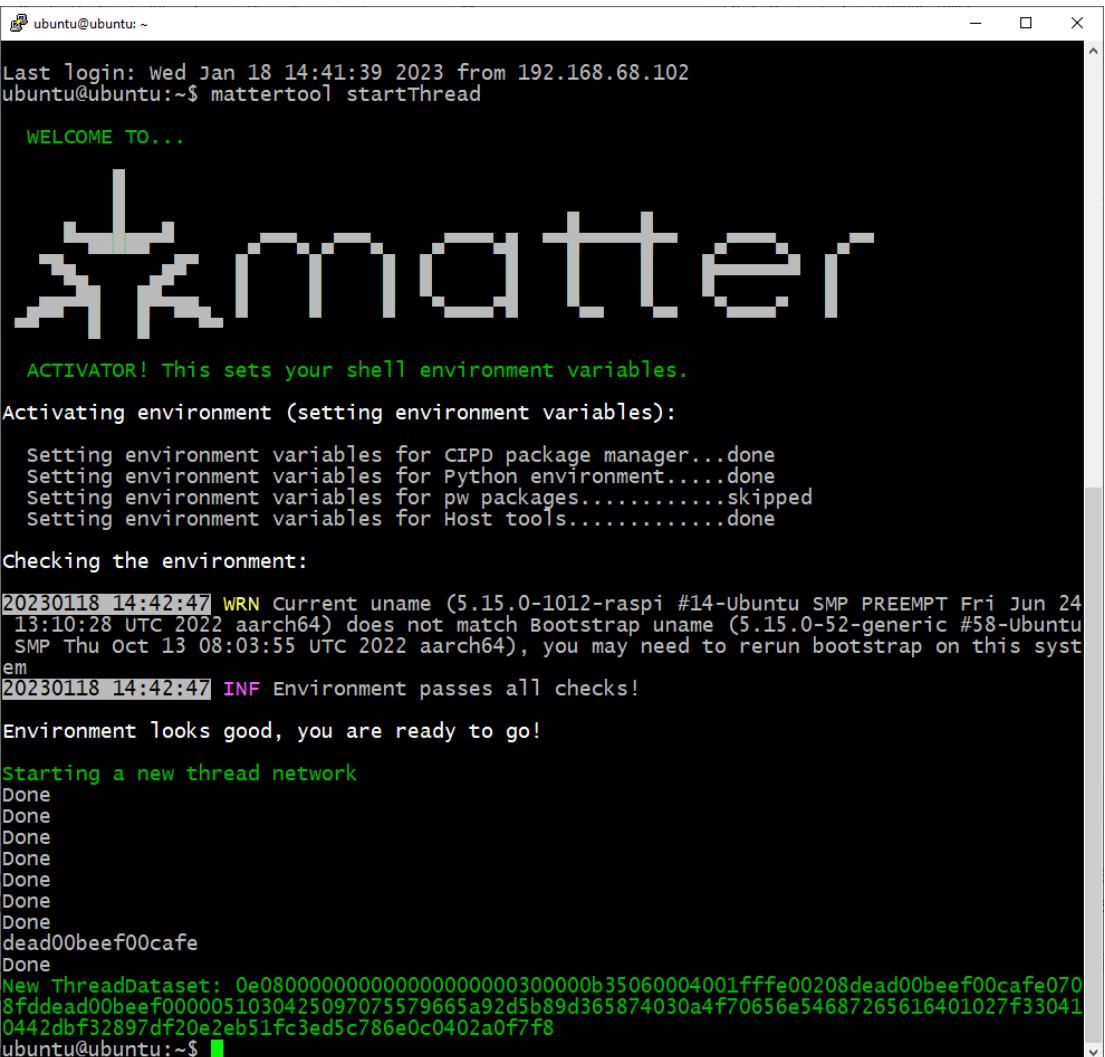
- The `mattertool` command is used to interact with the OTBR and network devices
 - This tool serves as a wrapper around the `chip-tool` command
 - In some cases, `mattertool` issues multiple `chip-tool` commands to make interacting with the system simpler
 - Help can be obtained by using `mattertool help`
 - Contextual help is provided for incomplete commands
- The discriminator can be used to control which devices join which networks
 - If working in groups each group has been assigned a discriminator
 - To set the discriminator for your SSH session enter:
`export DISCRIMINATOR=1234`
where:
1234 is the discriminator assigned to your group
 - Check the discriminator is set correctly by entering:
`mattertool help`



```
ubuntu@ubuntu:~$ mattertool help
This bash script centralize and simplifies the use of chip-tool and starting a clean thread network
Here is an overview of the Available cmds :
toggle
vars
getThreadDataset
buildDCT
cleanVars
bleThread
off
parsePayload
startThread
blewifi
on
rebuildDCT
help
Available options :
-h, --help          Print this help
-n, --nodeId DIGIT Specify the Nodeid you are trying to reach
-e, --endpoint DIGIT Specify an endpoint to the desired the cluster
-d, --dataset HEX_STRING Thread Operation Dataset
-s, --ssid STRING   WiFi AP ssid that the end devices needs to connect to
-p, --password STRING WiFi AP password
Those configurations are held until the device is rebooted !
Active Vars:
    MATTER_ROOT: /home/ubuntu/connectedhomeip
    CHIPTOOL_PATH: /home/ubuntu/connectedhomeip/out/standalone/chip-tool
    NODE_ID:16207
    THREAD_DATA_SET: 0e0800000000000000000000300000b35060004001ffffe00208dead00beef00c
afe0708fddead00beef0000051030425097075579665a92d5b89d365874030a4f70656e54687265616401027
f330410442dbf32897df20e2eb51fc3ed5c786e0c0402a0f7f8
    PINCODE: 20202021
    DISCRIMINATOR: 3840
    SSID:
    lastNodeId: 0
You can preset them with export X=Y before running the script
You can also enter the full chip-tool command (without the chip-tool)
  e.g Mattertool levelcontrol read current-level 106 1
ubuntu@ubuntu:~$ █
```

Network Commands

- Network commands are used to create and recover networks
- To create a new Thread network:
 - `mattertool startThread`
 - When the network is created a Thread dataset will be output
- To recover a network after a power cycle the Thread dataset needs to be read:
 - `mattertool getThreadDataset`

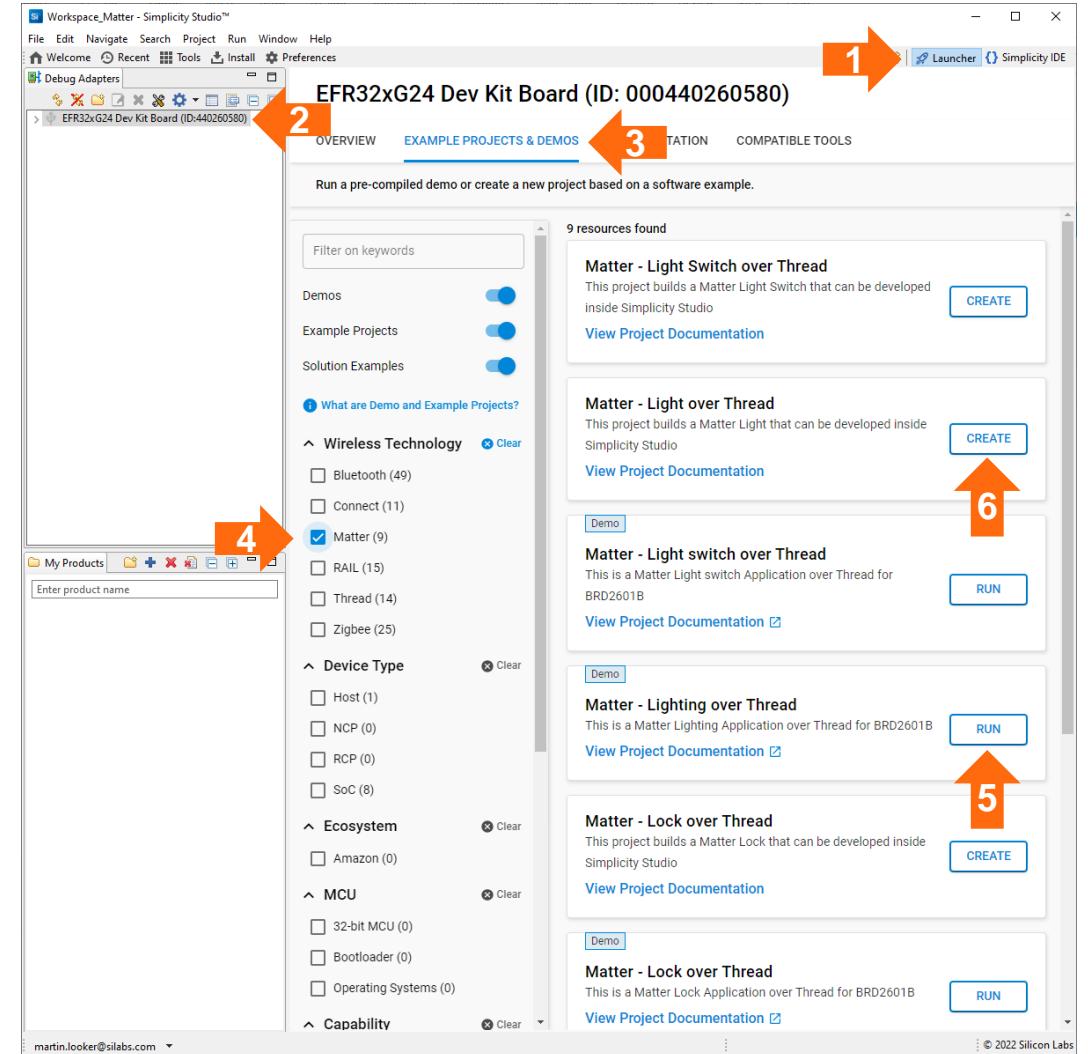


```
ubuntu@ubuntu: ~
Last login: Wed Jan 18 14:41:39 2023 from 192.168.68.102
ubuntu@ubuntu:~$ mattertool startThread
WELCOME TO...
matter
ACTIVATOR! This sets your shell environment variables.
Activating environment (setting environment variables):
Setting environment variables for CIPD package manager...done
Setting environment variables for Python environment.....done
Setting environment variables for pw packages.....skipped
Setting environment variables for Host tools.....done
Checking the environment:
20230118 14:42:47 WRN Current uname (5.15.0-1012-raspi #14-Ubuntu SMP PREEMPT Fri Jun 24
13:10:28 UTC 2022 aarch64) does not match Bootstrap uname (5.15.0-52-generic #58-Ubuntu
SMP Thu Oct 13 08:03:55 UTC 2022 aarch64), you may need to rerun bootstrap on this syst
em
20230118 14:42:47 INF Environment passes all checks!
Environment looks good, you are ready to go!
Starting a new thread network
Done
Done
Done
Done
Done
Done
Done
dead00beef00cafe
Done
New ThreadDataset: 0e0800000000000000000000300000b35060004001ffffe00208dead00beef00cafe070
8fddead00beef0000051030425097075579665a92d5b89d365874030a4f70656e54687265616401027f33041
0442dbf32897df20e2eb51fc3ed5c786e0c0402a0f7f8
ubuntu@ubuntu:~$
```

Light Demo / Example

- Connect board via USB

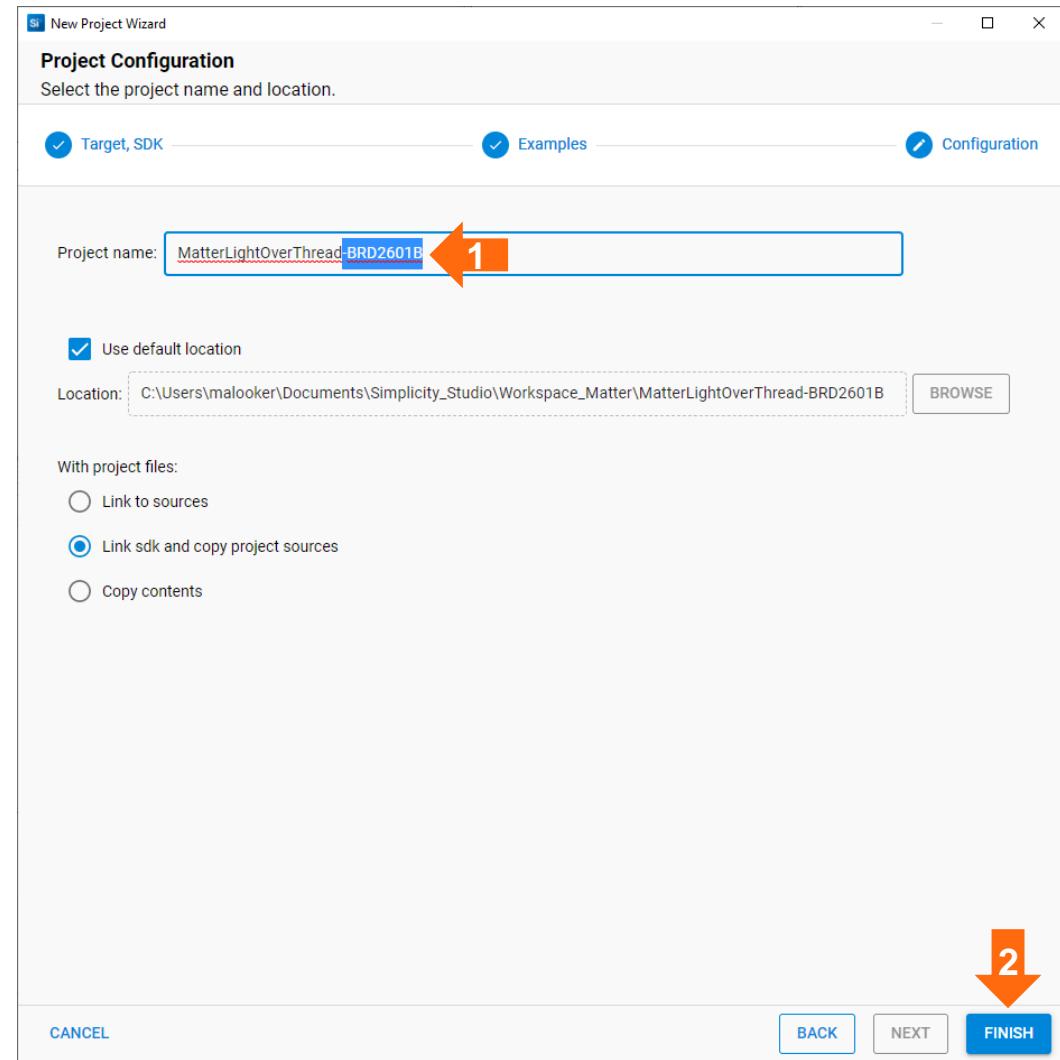
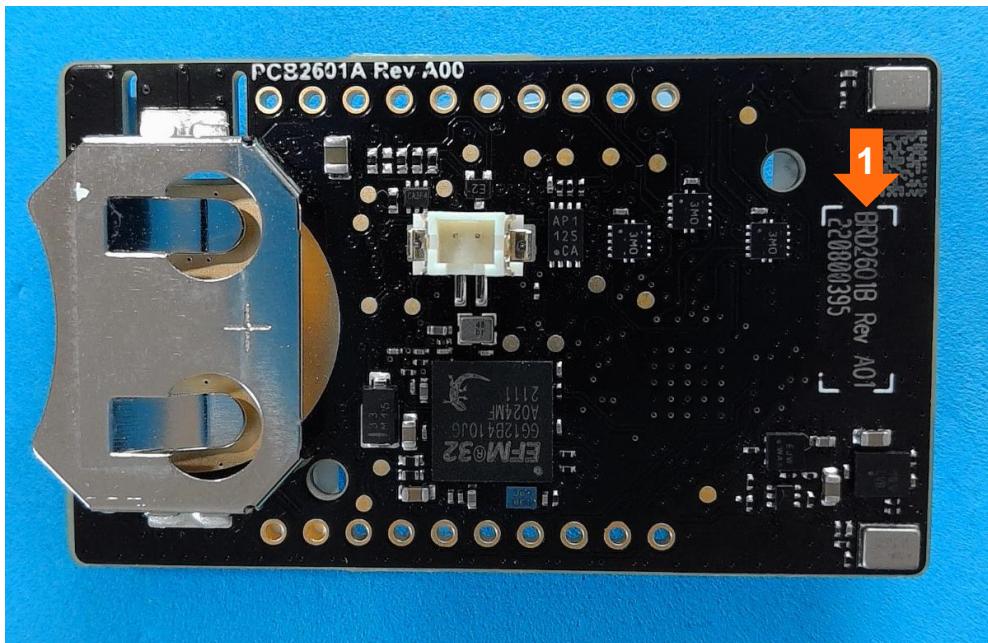
1. Switch to **Launcher** perspective
2. Select **EFR32xG24 Dev Kit Board** from **Debug Adapters** panel
 - ▶ Make a note of the last 4 digits of the device ID (we will use this as the network ID later)
3. Select **Example Projects and Demos**
4. Select **Matter** filter
5. To run a pre-compiled binary, find the **Matter – Lighting over Thread** demo and click the **RUN** button to flash the binary into the board
6. To create a project with code, find the **Matter – Light over Thread** example and click the **CREATE** button
 - ▶ Note that the build can take over 20 minutes



Light Example – New Project Wizard

- To complete the wizard:

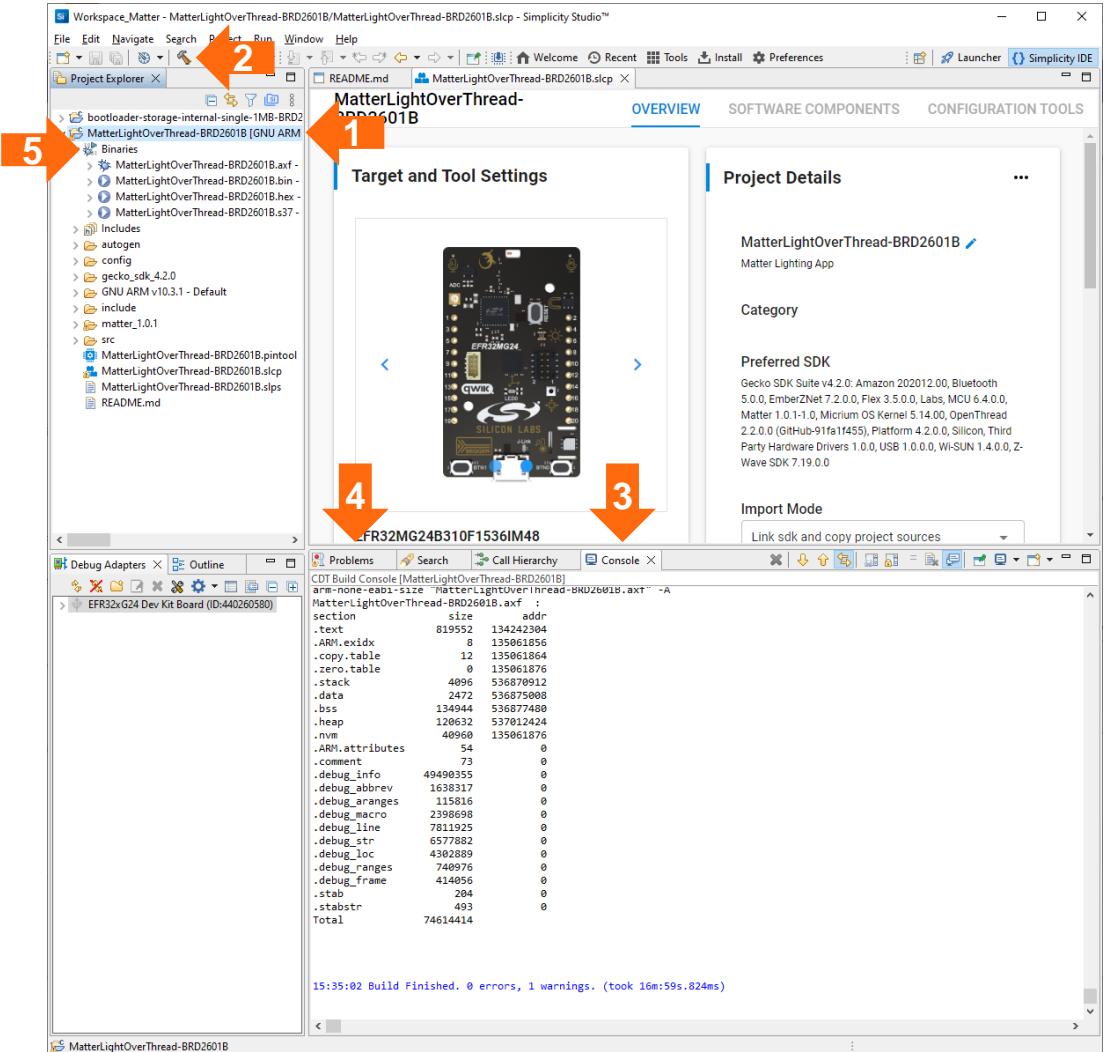
- Append the board number in the **Project Name** edit box
 - Board numbers are printed on each board
 - This helps when the same example is created for different boards
- Click the **FINISH** button



Light Example – Build

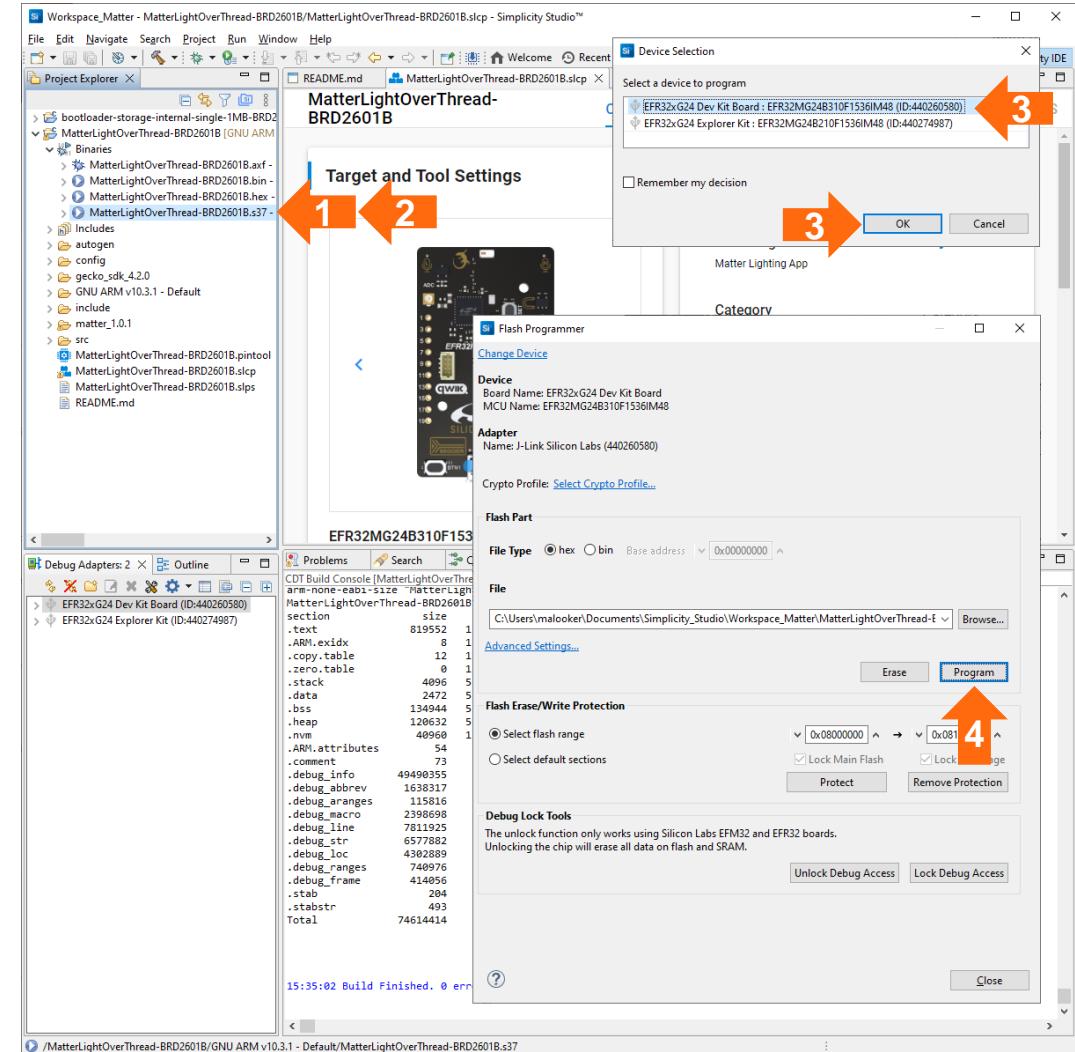
- To build the firmware:

- In the **Project Explorer** panel, select the top-level project file **MatterLightOverThread-BRD2601B**
- Click the **Build** button in the toolbar
- The **Console** panel will display progress as the firmware builds
- The **Problems** panel will display any errors or warnings encountered during the build
- The **Binaries** folder in the **Project Explorer** panel contains the built binaries
 - We will be using the **.s37** output file to program the device



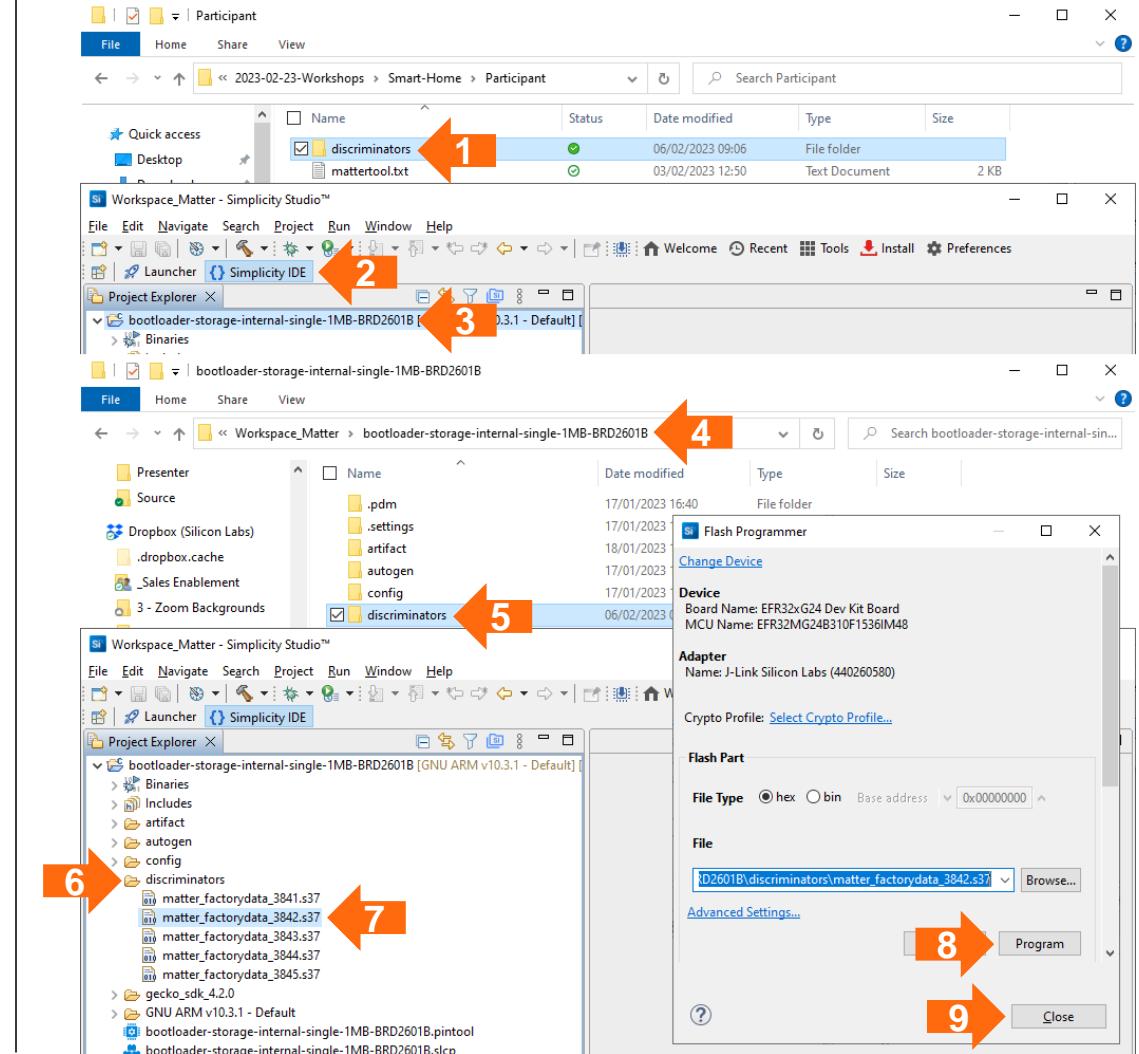
Light Example – Flash

- Ensure the correct bootloader binary is loaded into the device
- To flash the light firmware:
 1. In the **Project Explorer** panel, select and right-click the **.s37** binary to be flashed
 2. Click **Flash to device...** to open the Flash Programmer
 3. If you have more than one board connected, the **Device Selection** window will open, select the device to be programmed and click the **OK** button
 4. To flash the device from the **Flash Programmer** window, click the **Program** button



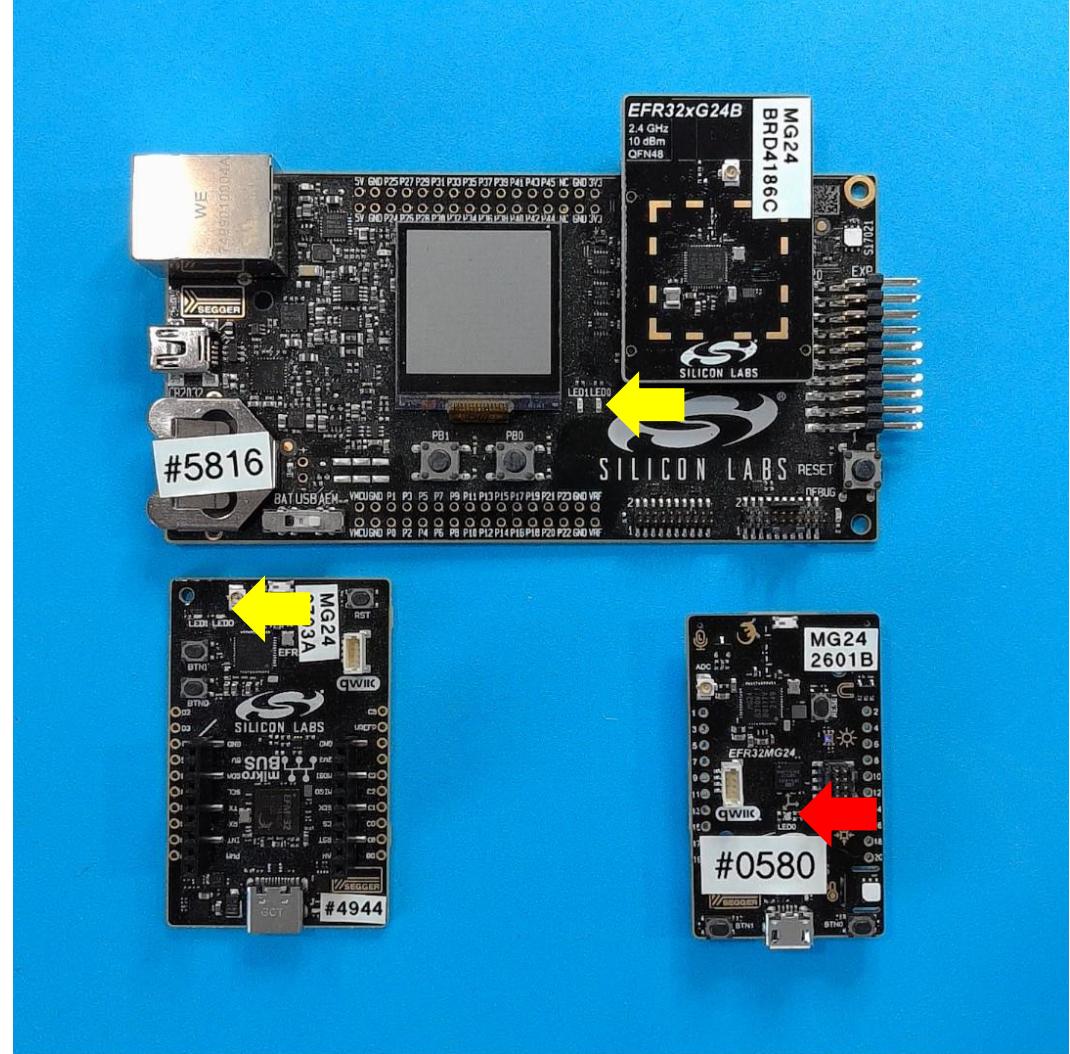
Light – Change Discriminator

- The discriminator and associated certificates are stored in NVM in the light device
- To change the discriminator in the light device:
 - In **Simplicity Studio**, expand the **discriminators** folder (in the `ew23-matter-main\bin` folder) in the **Project Explorer** panel
 - Right-click the `.s37` file for your group's discriminator and select **Flash to device...**
 - In the **Flash Programmer**, check the **File** is correct and click the **Program** button
 - Use the **Close** button to exit the Flash Programmer



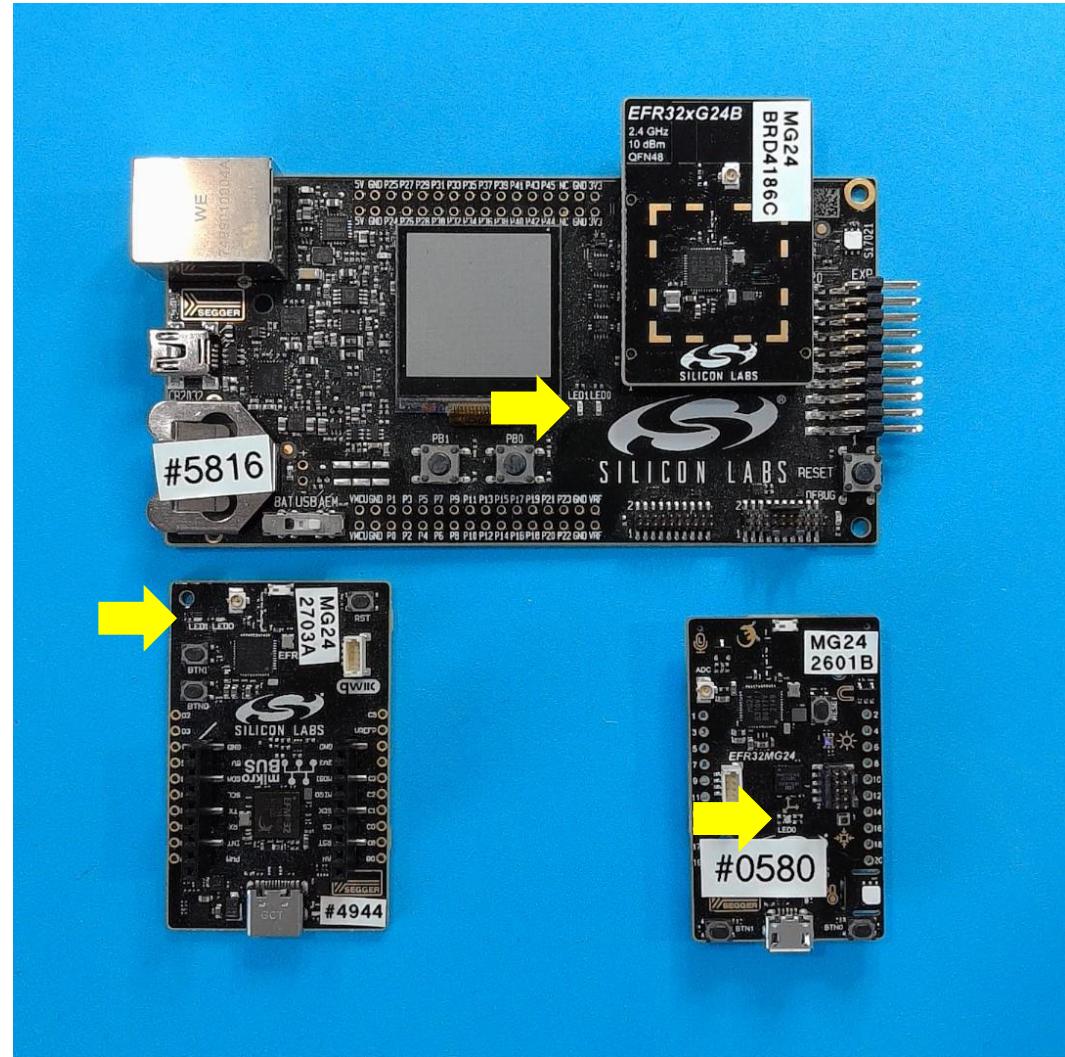
Light – Commissioning

- To commission a new device into the network using BLE (the nodeid for the new device needs to be specified with the -n option):
 - `mattertool bleThread -n 100`
where:
100 is the nodeid for the new device (use the last 4 digits of the device ID, noted earlier, to ensure uniqueness)
- The devices that join the network use LED0 to indicate the status of the device in the network:
 - Short flash: advertising to join the network
 - Long flash: commissioning in progress
 - On: in the network
 - On BRD2601B the red channel of LED0 is used to indicate the network status
- Devices in a network can be factory reset using BTN0:
 - Hold down BTN0 until LED0 flashes equally on and off
 - Continue to hold down BTN0 until the LED returns to a short flash



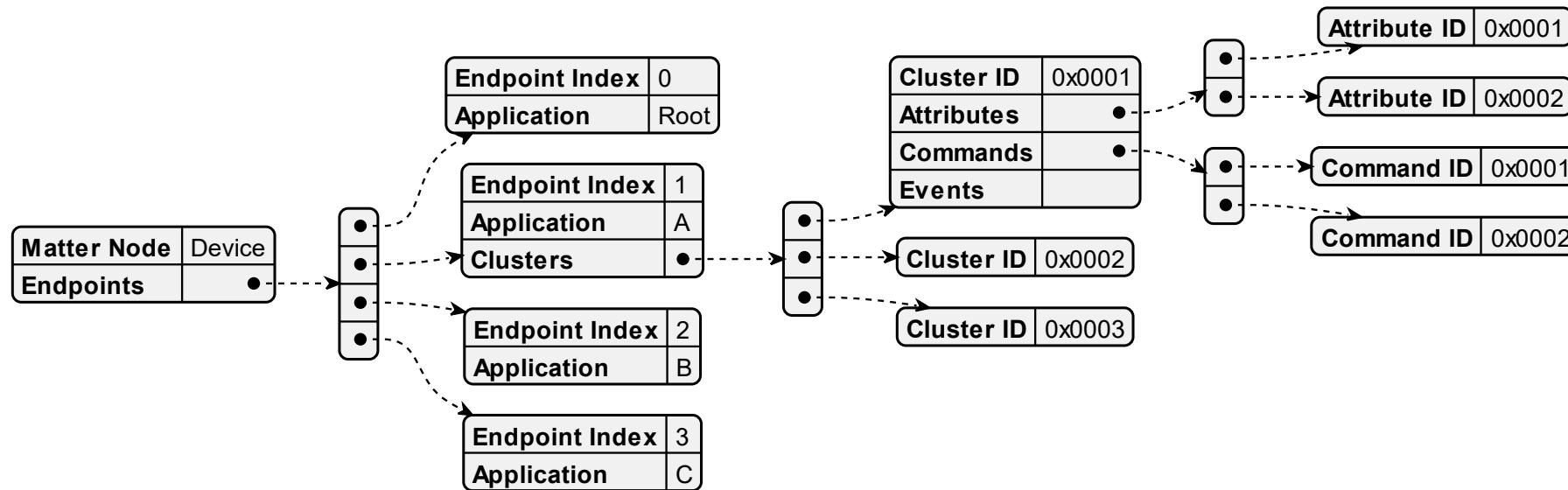
Light – On/Off Cluster Commands

- To turn on a light:
 - `mattertool on -n 100`
where:
100 is the nodeid of the light to be turned on
- To turn off a light:
 - `mattertool off -n 100`
where:
100 is the nodeid of the light to be turned off
- To toggle a light between on and off:
 - `mattertool toggle -n 100`
where:
100 is the nodeid of the light to be toggled
- Lights use LED1 (and the LCD, where available) to indicate the light's state
 - On BRD2601 the green channel of LED indicates the light's state (making it yellow when in the on state and in the network)
- Lights use BTN1 to toggle the light locally



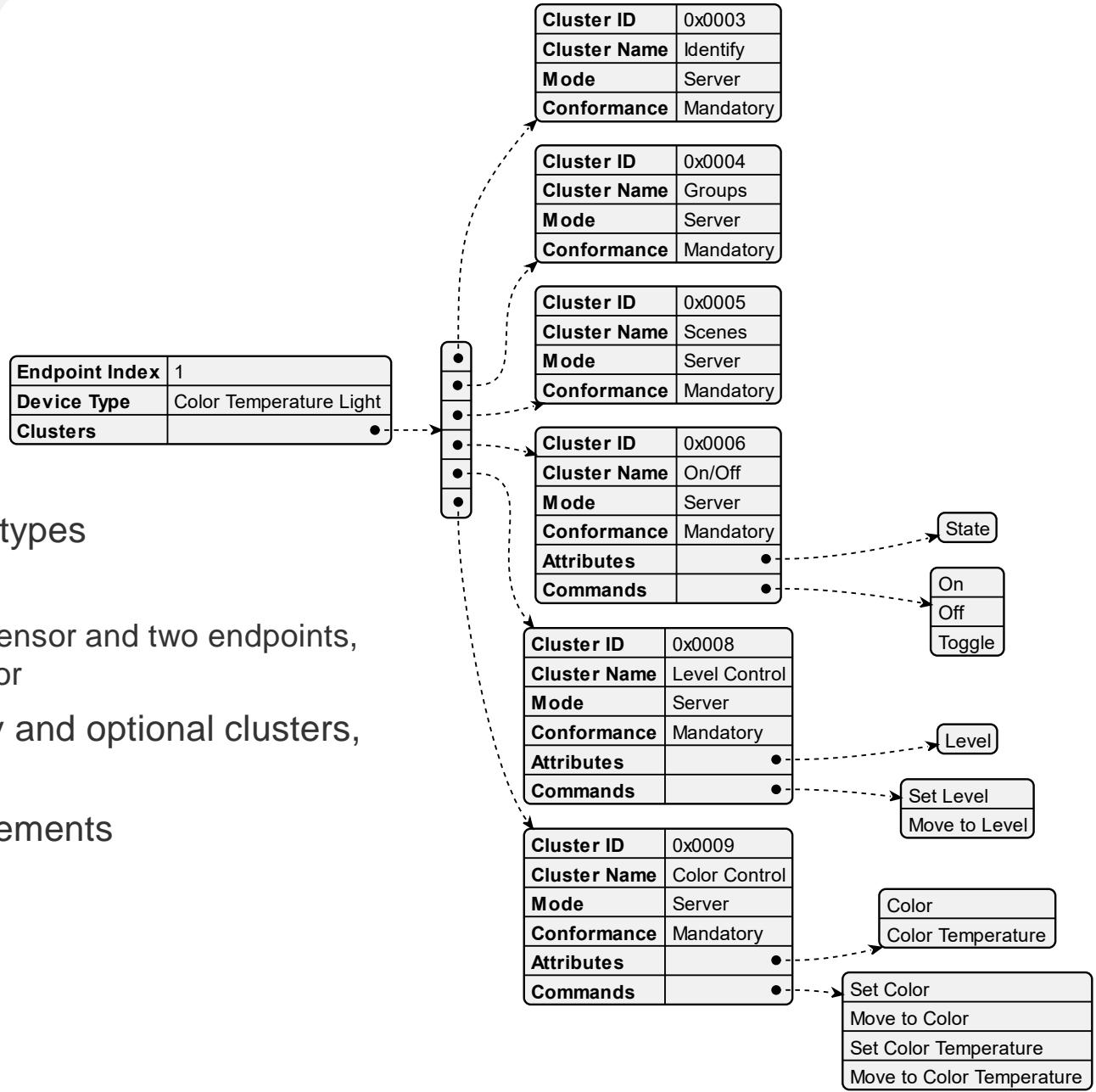
Zigbee Cluster Library

- Matter leverages the Zigbee Cluster Library (ZCL)
 - Provides a set of field-proven methods to form and control devices
 - Does not perfectly match the ZCL, has been extended with new functionality as needed
- An endpoint is generally a single device type
 - Interactions happen between local endpoints and remote endpoints in a client/server model
- Specific functions are described by clusters. They contain:
 - Attributes (state or modes)
 - Commands (operations to be acted on)
 - Events (transitions and alarms)



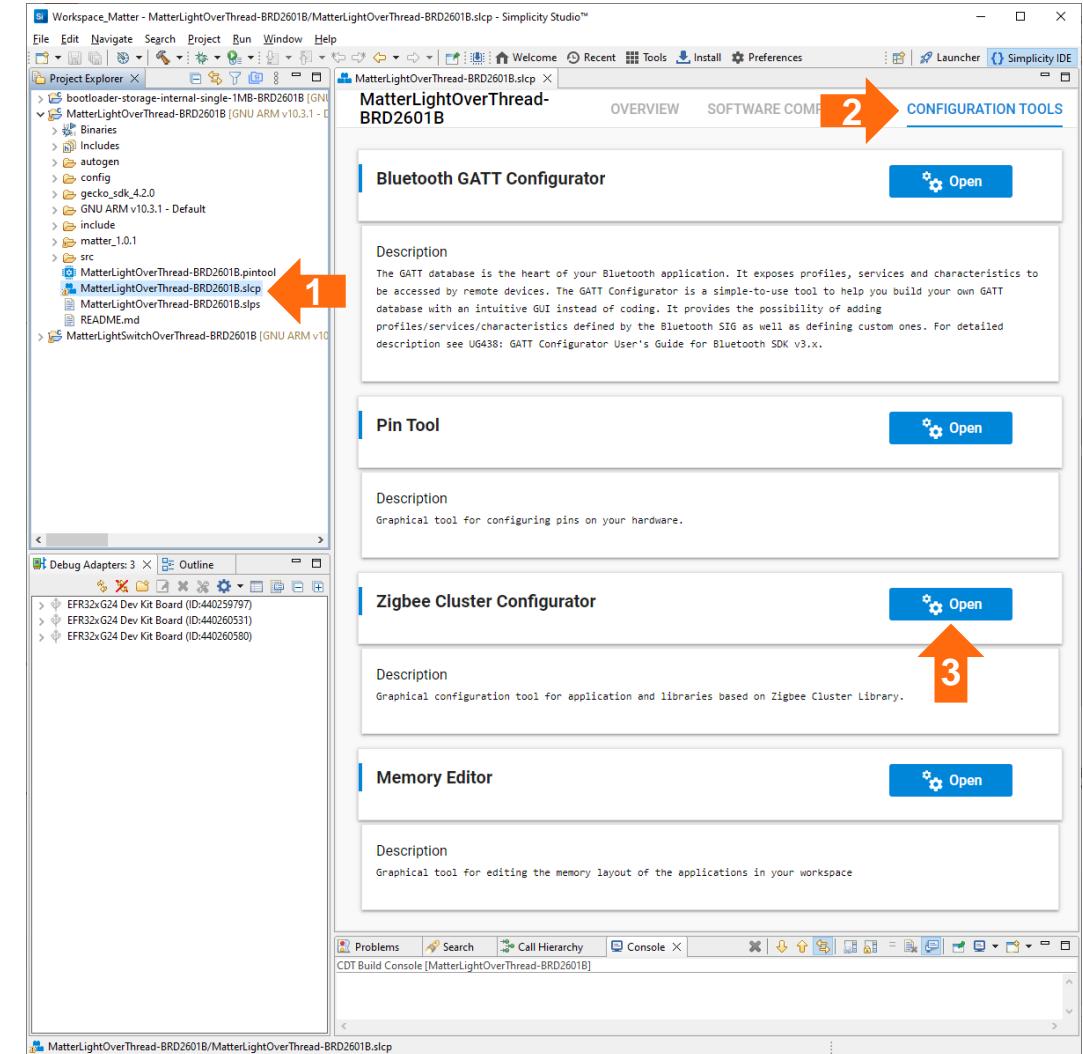
Device Types

- A Matter product may contain one or more device types
 - Each endpoint represents a logical device
 - You could have a bulb with an integrated occupancy sensor and two endpoints, one for the light and a second for the occupancy sensor
- Device Types encapsulate a number of mandatory and optional clusters, attributes, and features
- Example: Color Temperature Light Device Requirements



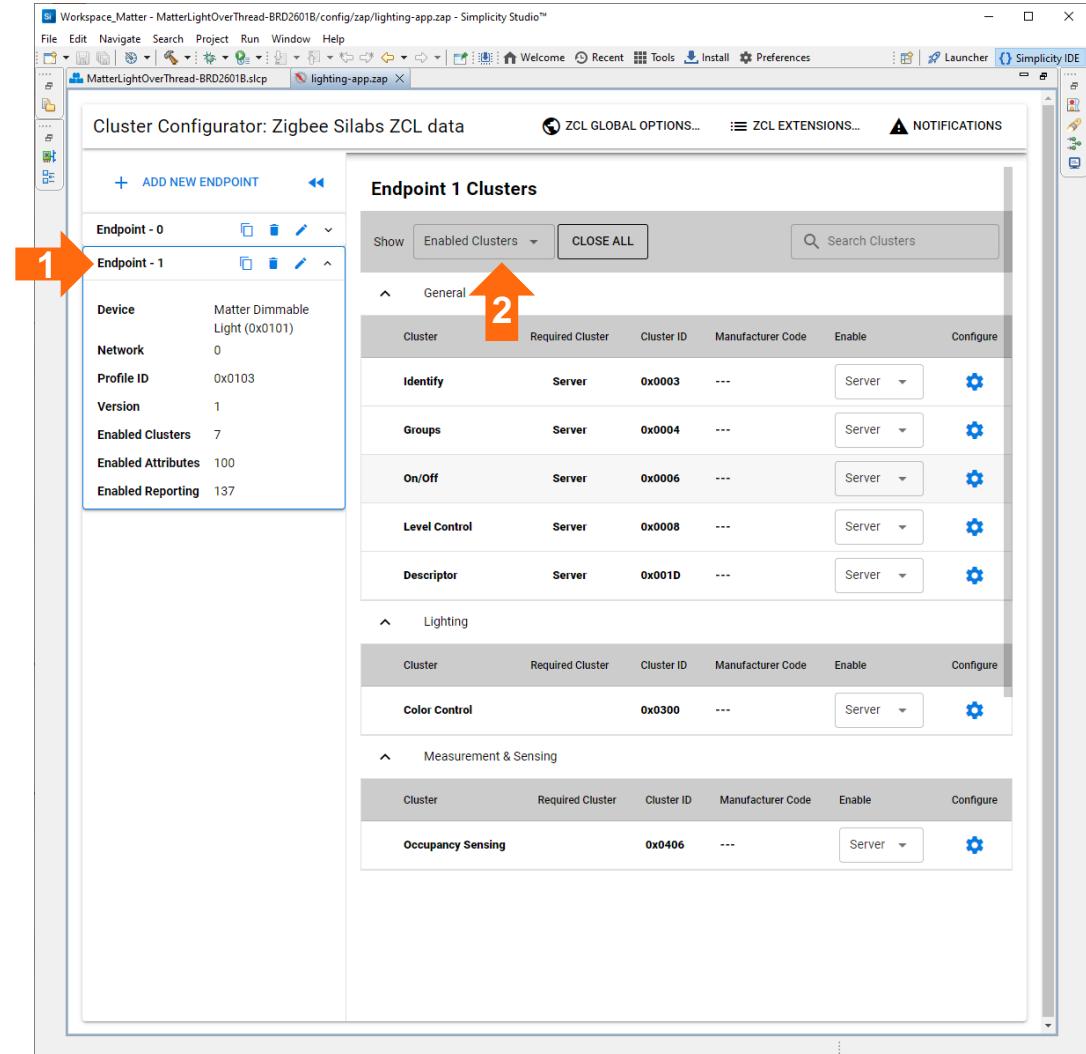
Zigbee Cluster Configurator – Open

- The Zigbee Cluster Configurator allows the endpoints and clusters in the firmware to be edited
- To open the Zigbee Cluster Configurator:
 - Double-click the `.slcp` file in the project
 - Select **Configuration Tools**
 - In the **Zigbee Cluster Configurator** box, click the **Open** button



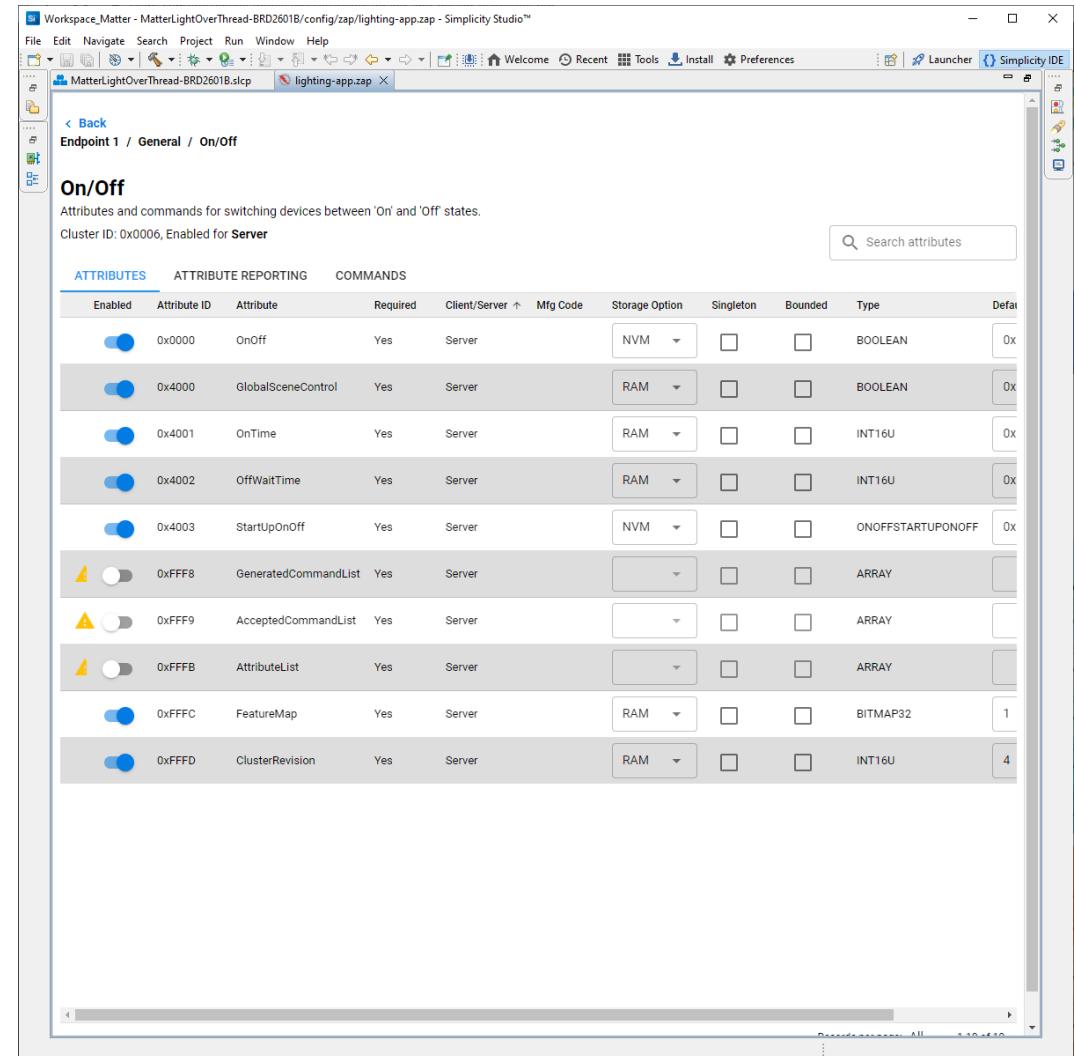
Zigbee Cluster Configurator – View Enabled Clusters

- The Zigbee Cluster Configurator allows the endpoints and clusters in the firmware to be viewed and/or edited
- Endpoint 0 contains clusters to manage the Matter Device
- Endpoint 1 contains clusters to operate the light application in the device, to view the enabled clusters:
 - Click the **Endpoint – 1** box
 - From the **Show** drop-box, select **Enabled Clusters**
- The clusters built into the application are displayed:
 - Each cluster has an ID for cluster, these are allocated by the CSA for the standard cluster they define
 - Note that the On/Off cluster uses an ID of 0x0006 (we will use this later)
 - The **Enable** column specifies how the cluster will be used in the application
 - Not Enabled, Client, Server or Client and Server can be selected



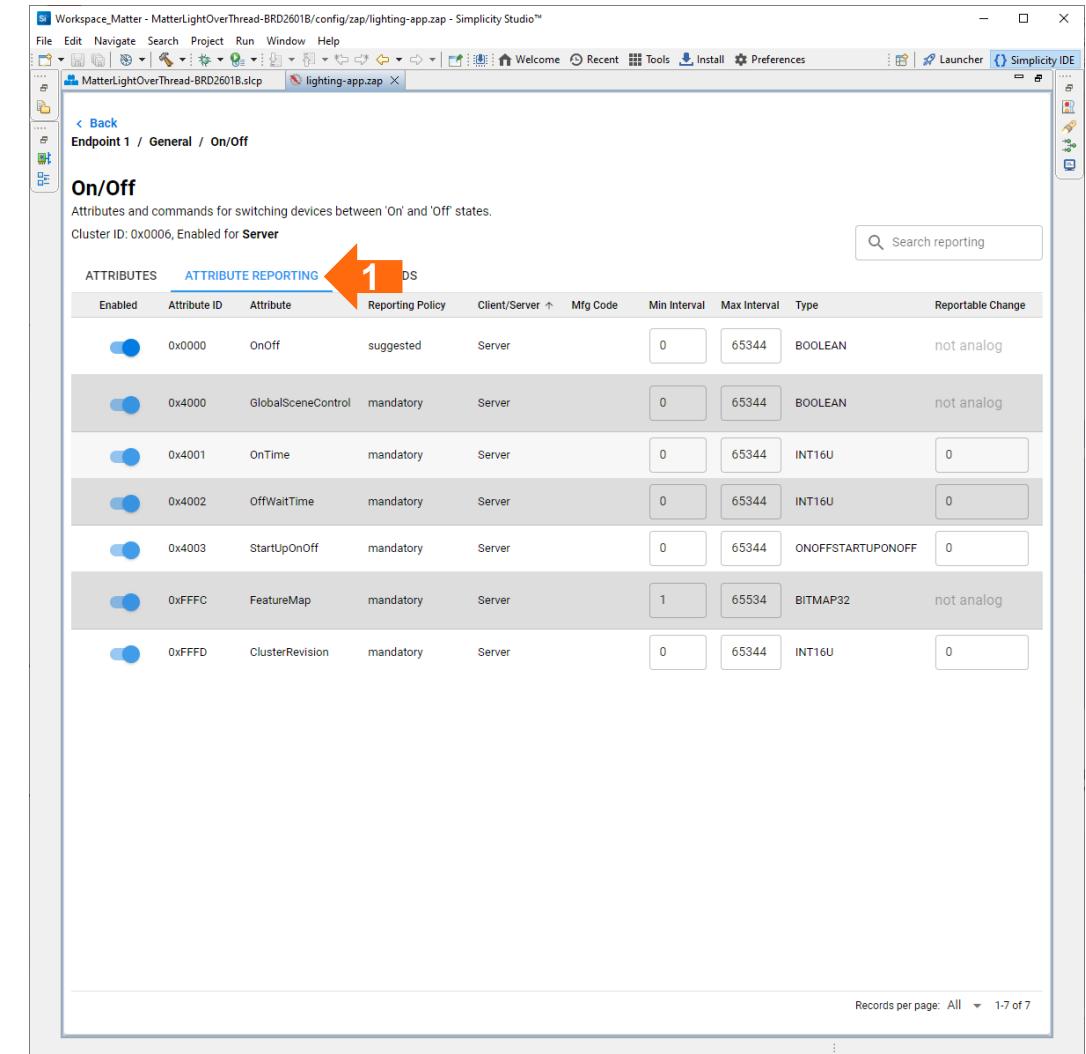
Zigbee Cluster Configurator – View On/Off Cluster Attributes

- To view the attribute settings in the On/Off cluster:
 - In the **Zigbee Cluster Configurator**, click the **Settings (Cog)** button in the **On/Off Cluster** row
- Information on the attributes is shown and some can be edited :
 - Attribute IDs for standard clusters are allocated by the CSA
 - Warning icons are shown where the attribute does not confirm to the CSA specification



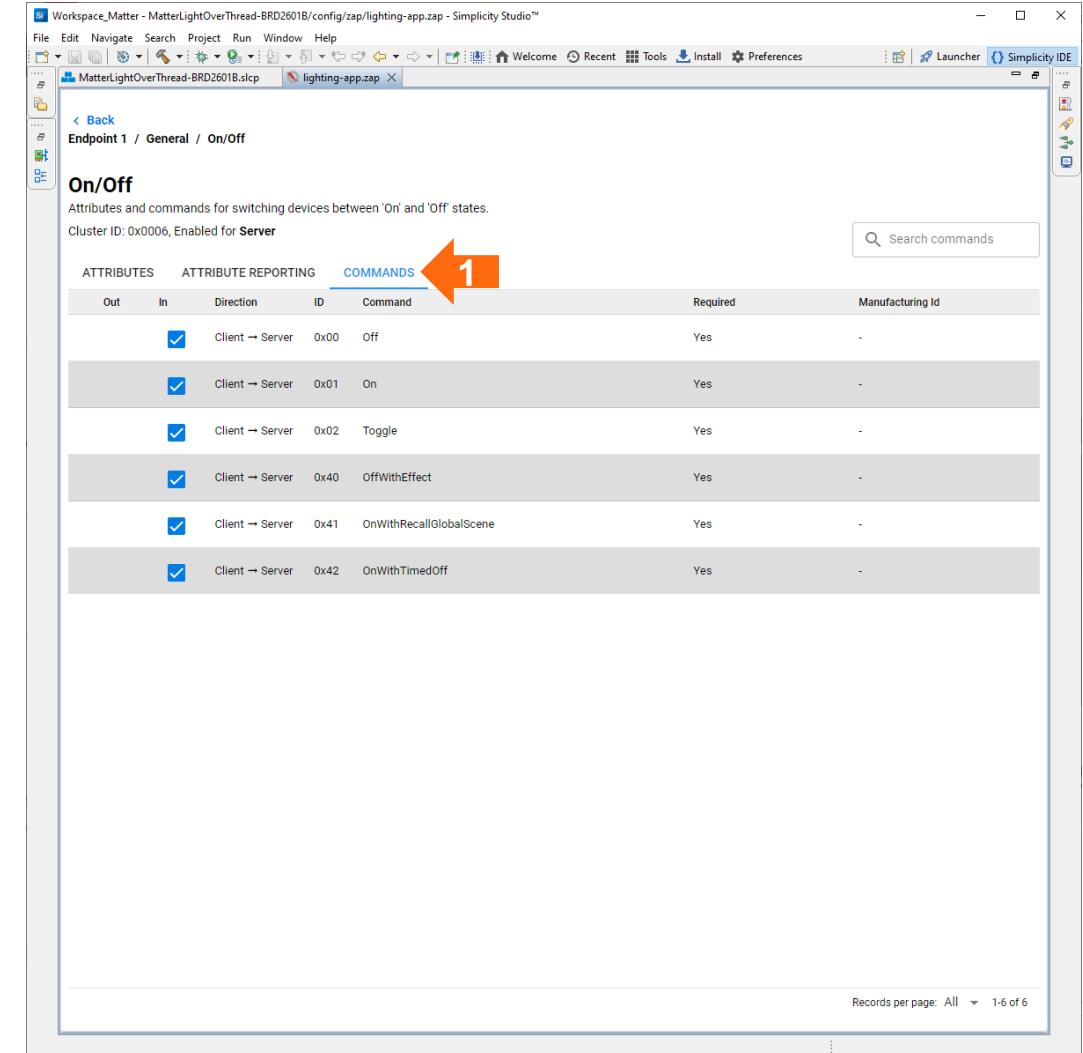
Zigbee Cluster Configurator – View On/Off Cluster Attribute Reporting

- To view the Attribute Reporting settings in the On/Off cluster:
 - Click **Attribute Reporting**
- Devices can request reports either on a timed basis or when values change:
 - Attribute reporting settings can be viewed and/or edited here



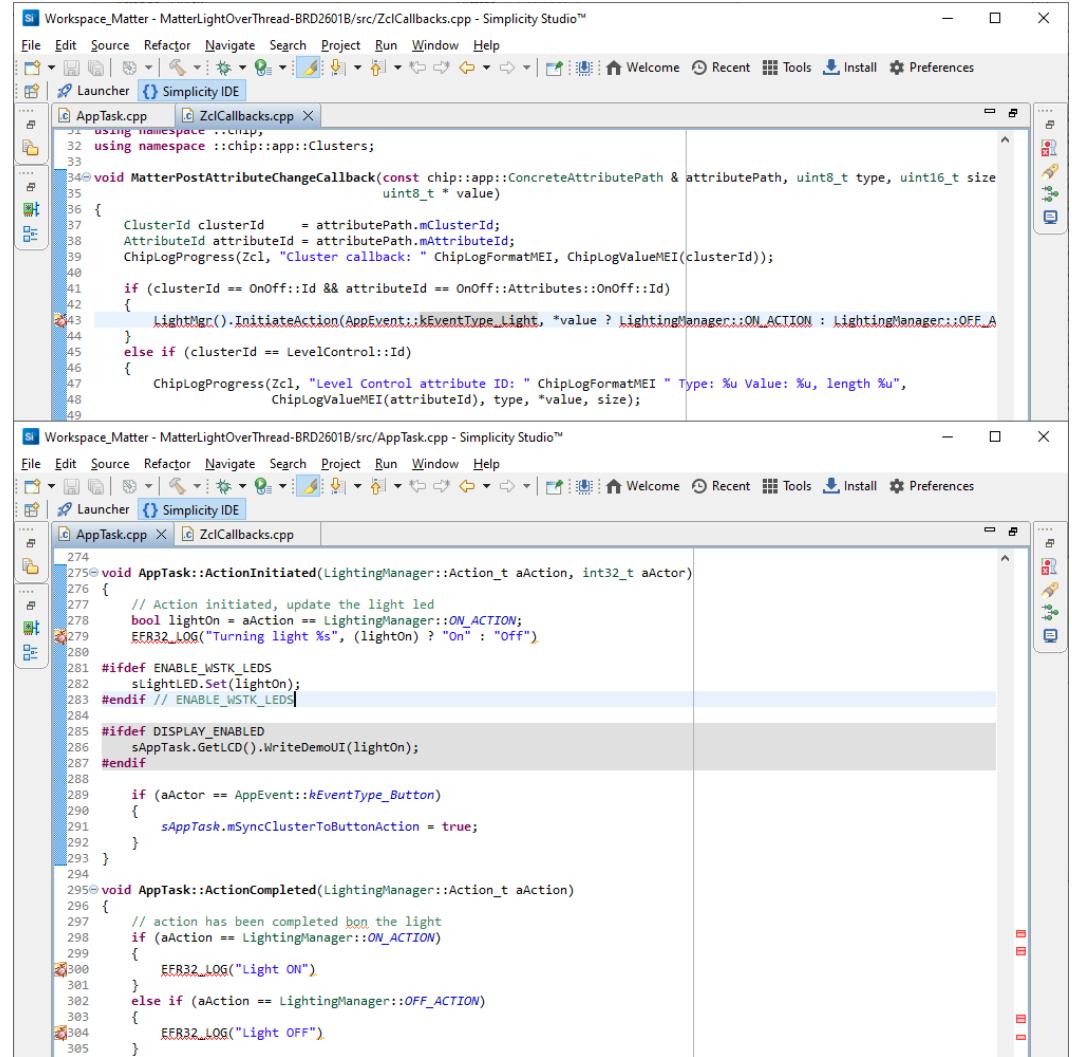
Zigbee Cluster Configurator – View On/Off Cluster Commands

- To view the Cluster Command settings in the On/Off cluster:
 1. Click **Commands**
- Commands can be sent to devices to control them:
 - Writing to attributes can also control devices
 - Using commands can also add additional functionality such as ramping up brightness when a light is turned on
- The On, Off and Toggle commands were issued by mattertool when we controlled the light



Code – On/Off Cluster Commands

- In addition to configuring the clusters that will appear in the device code needs to be added to implement appropriate functions
- ZclCallbacks.cpp**
MatterPostAttributeChangeCallback()
 - This function is called after an attribute is changed
 - It could be changed using a write attribute operation or a received command
 - For the On/Off attribute in the On/Off cluster this code initiates an action in a **LightMgr** class to control the LED
- AppTask.cpp ActionInitiated()**
 - The **LightMgr** class calls this function to control the LED
 - This provides a single point in the software for hardware control regardless of whether the change was initiated by a wireless command or by local control using the switch



The image displays two side-by-side screenshots of the Simplicity Studio IDE interface. Both windows have a title bar 'Workspace_Matter - MatterLightOverThread-BRD2601B/src/ZclCallbacks.cpp - Simplicity Studio™' and a menu bar with File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help.

The top window shows the 'ZclCallbacks.cpp' file with the following code snippet:

```
31 using namespace ::chip;
32 using namespace ::chip::app::Clusters;
33
34 void MatterPostAttributeChangeCallback(const chip::app::ConcreteAttributePath & attributePath, uint8_t type, uint16_t size
35                                         uint8_t * value)
36 {
37     ClusterId clusterId = attributePath.mClusterId;
38     AttributeId attributeId = attributePath.mAttributeId;
39     ChipLogProgress(Zcl, "Cluster callback: " ChipLogFormatMEI, ChipLogValueMEI(clusterId));
40
41     if (clusterId == OnOff::Id && attributeId == OnOff::Attributes::OnOff::Id)
42     {
43         LightMgr().InitiateAction(AppEvent::kEventType_Light, *value ? LightingManager::ON_ACTION : LightingManager::OFF_A
44     }
45     else if (clusterId == LevelControl::Id)
46     {
47         ChipLogProgress(Zcl, "Level Control attribute ID: " ChipLogFormatMEI " Type: %u Value: %u, length %u",
48                         ChipLogValueMEI(attributeId), type, *value, size);
49 }
```

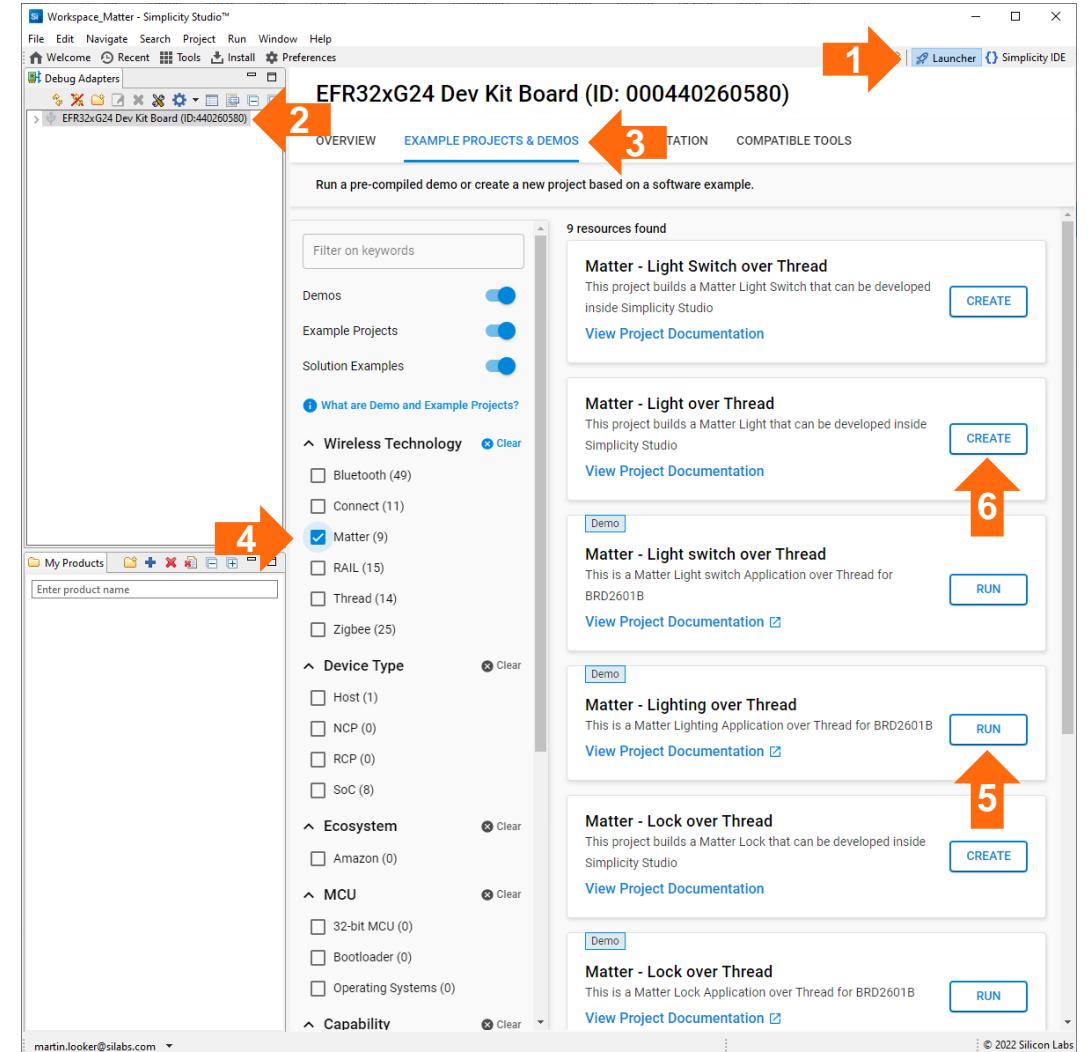
The bottom window shows the 'AppTask.cpp' file with the following code snippet:

```
274
275 void AppTask::ActionInitiated(LightingManager::Action_t aAction, int32_t aActor)
276 {
277     // Action initiated, update the light led
278     bool lightOn = aAction == LightingManager::ON_ACTION;
279     EFR32_LOG("Turning light %s", (lightOn) ? "On" : "Off");
280
281 #ifdef ENABLE_WSTK_LEDS
282     sLightLED.Set(lightOn);
283 #endif // ENABLE_WSTK_LEDS
284
285 #ifdef DISPLAY_ENABLED
286     sAppTask.GetLCD().WriteDemoUI(lightOn);
287 #endif
288
289     if (aActor == AppEvent::kEventType_Button)
290     {
291         sAppTask.mSyncClusterToButtonAction = true;
292     }
293 }
294
295 void AppTask::ActionCompleted(LightingManager::Action_t aAction)
296 {
297     // action has been completed on the light
298     if (aAction == LightingManager::ON_ACTION)
299     {
300         EFR32_LOG("Light ON");
301     }
302     else if (aAction == LightingManager::OFF_ACTION)
303     {
304         EFR32_LOG("Light OFF");
305     }
306 }
```

Switch Demo / Example

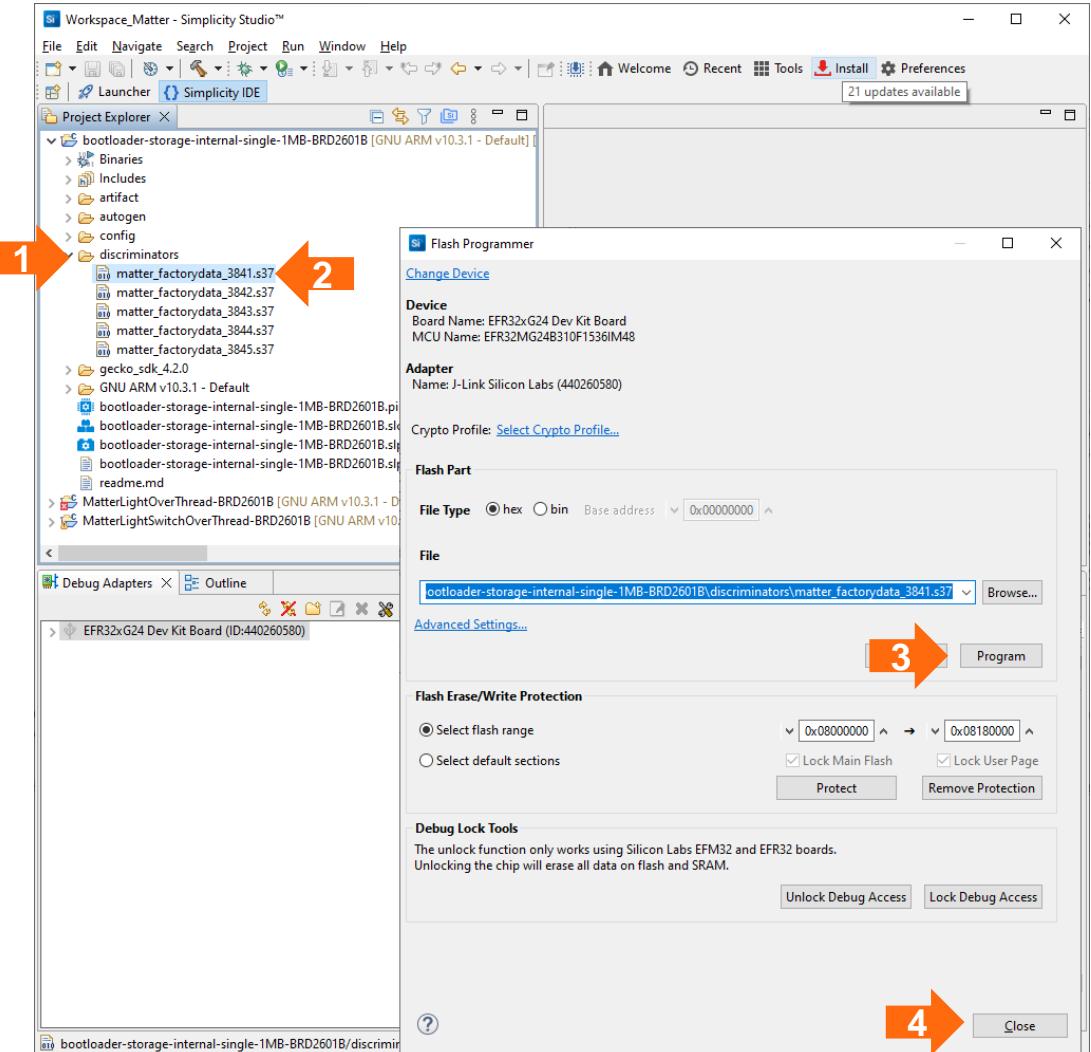
- Connect board via USB

1. Switch to **Launcher** perspective
2. Select **EFR32xG24 Dev Kit Board** from **Debug Adapters** panel
 - ▶ Make a note of the last 4 digits of the device ID (we will use this as the network ID later)
3. Select **Example Projects and Demos**
4. Select **Matter** filter
5. To run a pre-compiled binary, find the **Matter – Light Switch over Thread** demo and click the **RUN** button to flash the binary into the board
6. To create a project with code, find the **Matter – Light Switch over Thread** example and click the **CREATE** button
 - ▶ Note that the build can take over 20 minutes
 - ▶ Follow the slides for the Light Example to complete the New Project Wizard, Build and Flash the Switch Example



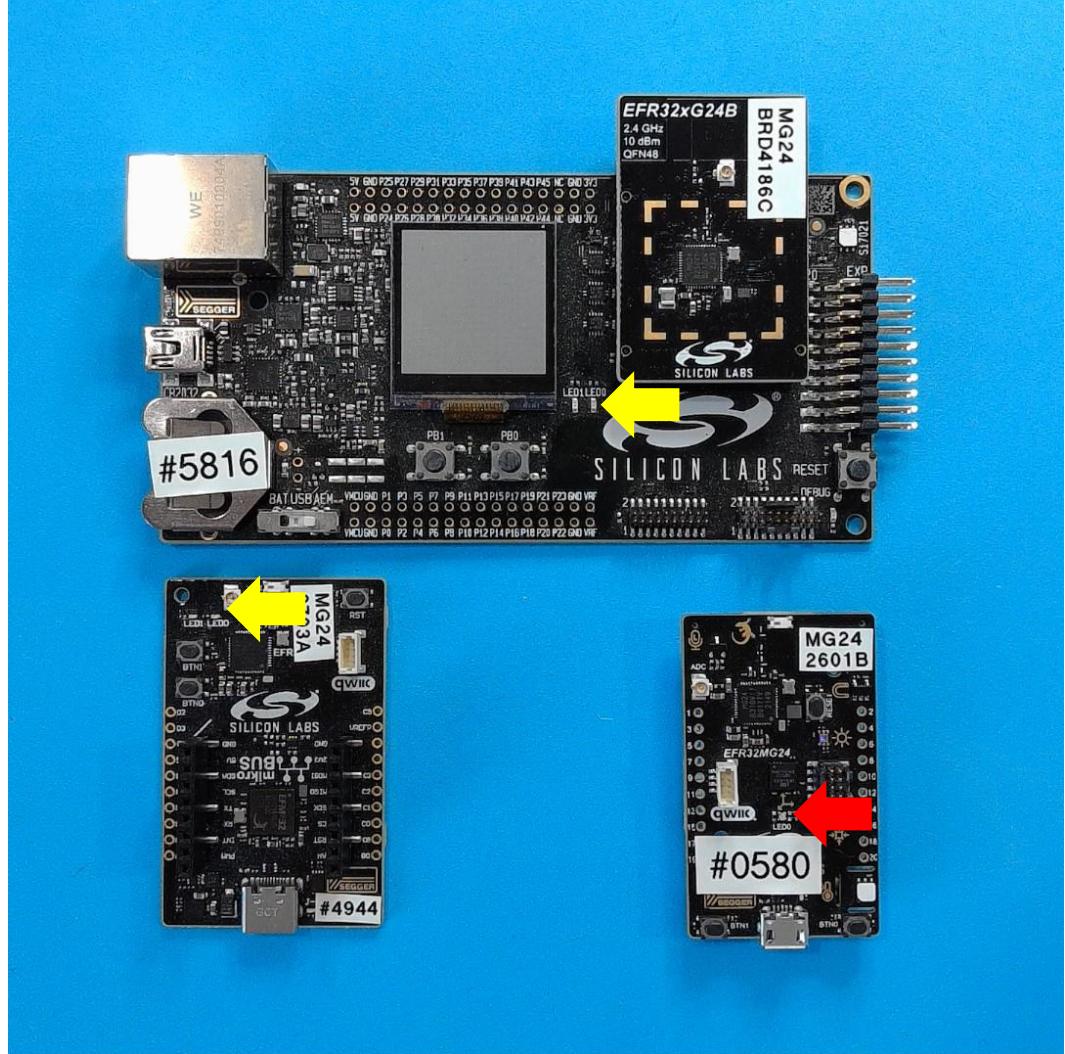
Switch – Change Discriminator

- The discriminator and associated certificates are stored in NVM in the light device
- To change the discriminator in the switch device:
 - In **Simplicity Studio**, expand the **discriminators** folder (copied in earlier) in the **Project Explorer** panel
 - Right-click the **.s37** file for your group's discriminator and select **Flash to device...**
 - In the **Flash Programmer**, check the **File** is correct and click the **Program** button
 - Use the **Close** button to exit the Flash Programmer



Switch – Commissioning

- To commission a new device into the network using BLE (the nodeid for the new device needs to be specified with the -n option):
 - `mattertool bleThread -n 200`
where:
200 is the nodeid for the new device (use the last 4 digits of the device ID, noted earlier, to ensure uniqueness)
- The devices that join the network use LED0 to indicate the status of the device in the network:
 - Short flash: advertising to join the network
 - Long flash: commissioning in progress
 - On: in the network
 - On BRD2601B the red channel of LED0 is used to indicate the network status
- Devices in a network can be factory reset using BTN0:
 - Hold down BTN0 until LED0 flashes equally on and off
 - Continue to hold down BTN0 until the LED returns to a short flash



Access Control List – Write Command

- Each device has an Access Control List (ACL), this list defines the transmitting devices in the network that the device with the ACL, the receiving device, will react to
- To write to the ACL in a device:

1. `mattertool accesscontrol write acl '[{ "fabricIndex" : 1 , "privilege" : 5 , "authMode" : 2 , "subjects" : [112233] , "targets" : null } , { "fabricIndex" : 1 , "privilege" : 3 , "authMode" : 2 , "subjects" : [200] , "targets" : null }]' 100 0`

Where:

`112233` is the nodeid of the controller (OTBR) being placed in the ACL

`200` is the nodeid of the switch being placed in the ACL

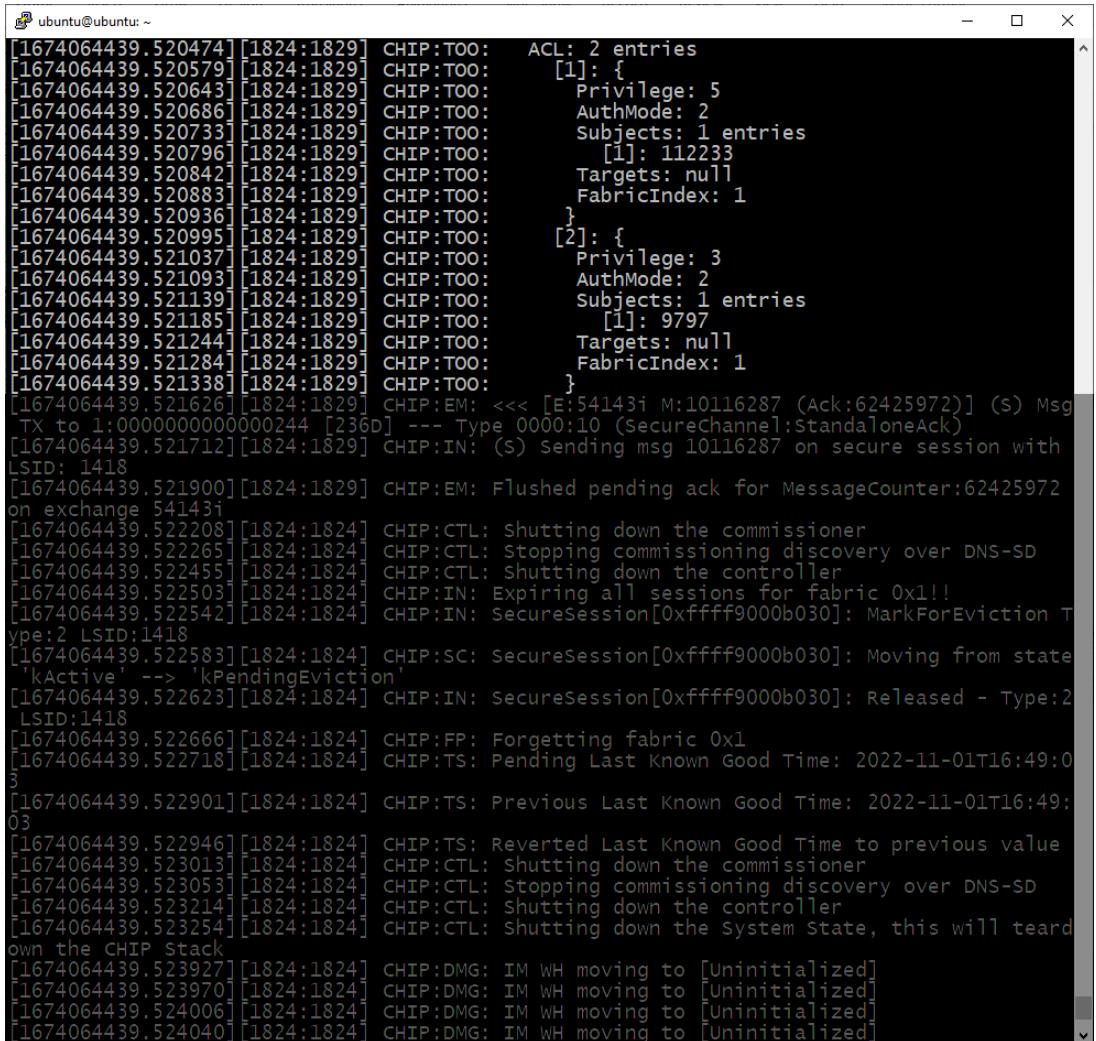
`100` is the nodeid of the light containing the ACL to be written

`0` is the endpoint in the light that holds the ACL, this is always 0

- When using this command, the whole ACL is replaced, take care not to lose the entry for the controller (OTBR) which always has nodeid 112233
- Multiple devices with the same settings can be written by specifying a comma-separated list for the `subjects`

Access Control List – Read Command

- To read the ACL from a device:
 1. `mattertool accesscontrol read acl 100 0`
Where:
100 is the nodeid of the device containing the ACL to be read
0 is the endpoint in the device that holds the ACL, this is always 0
- Scroll up through the output to find the output ACL



```
ubuntu@ubuntu: ~
[1674064439.520474][1824:1829] CHIP:TOO: ACL: 2 entries
[1674064439.520579][1824:1829] CHIP:TOO: [1]: {
[1674064439.520643][1824:1829] CHIP:TOO:   Privilege: 5
[1674064439.520686][1824:1829] CHIP:TOO:   AuthMode: 2
[1674064439.520733][1824:1829] CHIP:TOO:   Subjects: 1 entries
[1674064439.520796][1824:1829] CHIP:TOO:     [1]: 112233
[1674064439.520842][1824:1829] CHIP:TOO:   Targets: null
[1674064439.520883][1824:1829] CHIP:TOO:   FabricIndex: 1
[1674064439.520936][1824:1829] CHIP:TOO: }
[1674064439.520995][1824:1829] CHIP:TOO: [2]: {
[1674064439.521037][1824:1829] CHIP:TOO:   Privilege: 3
[1674064439.521093][1824:1829] CHIP:TOO:   AuthMode: 2
[1674064439.521139][1824:1829] CHIP:TOO:   Subjects: 1 entries
[1674064439.521185][1824:1829] CHIP:TOO:     [1]: 9797
[1674064439.521244][1824:1829] CHIP:TOO:   Targets: null
[1674064439.521284][1824:1829] CHIP:TOO:   FabricIndex: 1
[1674064439.521338][1824:1829] CHIP:TOO: }
[1674064439.521626][1824:1829] CHIP:EM: << [E:54143i M:10116287 (Ack:62425972)] (s) Msg
TX to 1:0000000000000244 [236D] --- Type 0000:10 (SecureChannel:StandaloneAck)
[1674064439.521712][1824:1829] CHIP:IN: (s) Sending msg 10116287 on secure session with
LSID: 1418
[1674064439.521900][1824:1829] CHIP:EM: Flushed pending ack for MessageCounter:62425972
on exchange 54143i
[1674064439.522208][1824:1824] CHIP:CTL: shutting down the commissioner
[1674064439.522265][1824:1824] CHIP:CTL: Stopping commissioning discovery over DNS-SD
[1674064439.522455][1824:1824] CHIP:CTL: Shutting down the controller
[1674064439.522503][1824:1824] CHIP:IN: Expiring all sessions for fabric 0x1!!
[1674064439.522542][1824:1824] CHIP:IN: SecureSession[0xfffff9000b030]: MarkForEviction T
ype:2 LSID:1418
[1674064439.522583][1824:1824] CHIP:SC: SecureSession[0xfffff9000b030]: Moving from state
'kActive' --> 'kPendingEviction'
[1674064439.522623][1824:1824] CHIP:IN: SecureSession[0xfffff9000b030]: Released - Type:2
LSID:1418
[1674064439.522666][1824:1824] CHIP:FP: Forgetting fabric 0x1
[1674064439.522718][1824:1824] CHIP:TS: Pending Last Known Good Time: 2022-11-01T16:49:0
3
[1674064439.522901][1824:1824] CHIP:TS: Previous Last Known Good Time: 2022-11-01T16:49:
03
[1674064439.522946][1824:1824] CHIP:TS: Reverted Last Known Good Time to previous value
[1674064439.523013][1824:1824] CHIP:CTL: Shutting down the commissioner
[1674064439.523053][1824:1824] CHIP:CTL: Stopping commissioning discovery over DNS-SD
[1674064439.523214][1824:1824] CHIP:CTL: Shutting down the controller
[1674064439.523254][1824:1824] CHIP:CTL: Shutting down the System State, this will tear
down the CHIP Stack
[1674064439.523927][1824:1824] CHIP:DMG: IM WH moving to [Uninitialized]
[1674064439.523970][1824:1824] CHIP:DMG: IM WH moving to [Uninitialized]
[1674064439.524006][1824:1824] CHIP:DMG: IM WH moving to [Uninitialized]
[1674064439.524040][1824:1824] CHIP:DMG: IM WH moving to [Uninitialized]
```

Binding Table – Write Single Command

- The switch devices contain a binding table, when BTN1 is pressed On/Off cluster toggle commands will be sent to the lights in this table
- To write a single entry to the binding table in a switch:

1. `mattertool binding write binding '[{ "fabricIndex" : 1 , "node" : 100 , "endpoint" : 1 , "cluster" : 6 }]' 200 1`

Where:

`100` is the nodeid of the light to be bound

`1` is the application endpoint in the light, this is always 1

`6` is the on/off cluster in the light, this is always 6

`200` is the nodeid of the switch for the binding table entry

`1` is the application endpoint in the switch that holds the binding table, this is always 1

- When using this command, the whole binding table is replaced
- When BTN1 is pressed the switch transmits the on/off toggle command to all devices in the binding table.

Binding Table – Read Command

- To read the binding table from a switch:
 - `mattertool binding read binding 200 1`
Where:
9797 is the nodeid of the switch containing the binding table to be read
1 is the application endpoint in the switch that holds the binding table, this is always 1
- Scroll up through the output to find the output binding table

```
ubuntu@ubuntu: ~
[1674065267.938583][1873:1878] CHIP:TOO: Binding: 1 entries
[1674065267.938715][1873:1878] CHIP:TOO: [1]: {
[1674065267.938755][1873:1878] CHIP:TOO:     Node: 580
[1674065267.938792][1873:1878] CHIP:TOO:     Endpoint: 1
[1674065267.938825][1873:1878] CHIP:TOO:     Cluster: 6
[1674065267.938860][1873:1878] CHIP:TOO:     FabricIndex: 1
[1674065267.938894][1873:1878] CHIP:TOO:   }
[1674065267.939103][1873:1878] CHIP:EM: <<< [E:371851 M:265971183 (Ack:91981266)] (S) Ms
g TX to 1:0000000000002645 [236D] --- Type 0000:10 (SecureChannel:StandaloneAck)
[1674065267.939166][1873:1878] CHIP:IN: (S) Sending msg 265971183 on secure session with
LSID: 48033
[1674065267.939337][1873:1878] CHIP:EM: Flushed pending ack for MessageCounter:91981266
on exchange 371851
[1674065267.939602][1873:1873] CHIP:CTL: Shutting down the commissioner
[1674065267.939653][1873:1873] CHIP:CTL: Stopping commissioning discovery over DNS-SD
[1674065267.939825][1873:1873] CHIP:CTL: Shutting down the controller
[1674065267.939868][1873:1873] CHIP:IN: Expiring all sessions for fabric 0x1!!
[1674065267.939905][1873:1873] CHIP:IN: SecureSession[0xfffff9400b030]: MarkForEviction T
ype:2 LSID:48033
[1674065267.939940][1873:1873] CHIP:SC: SecureSession[0xfffff9400b030]: Moving from state
'kActive' --> 'kPendingEviction'
[1674065267.939974][1873:1873] CHIP:IN: SecureSession[0xfffff9400b030]: Released - Type:2
LSID:48033
[1674065267.940012][1873:1873] CHIP:FP: Forgetting fabric 0x1
[1674065267.940134][1873:1873] CHIP:TS: Pending Last Known Good Time: 2022-11-01T16:49:0
3
[1674065267.940308][1873:1873] CHIP:TS: Previous Last Known Good Time: 2022-11-01T16:49:
03
[1674065267.940348][1873:1873] CHIP:TS: Reverted Last Known Good Time to previous value
[1674065267.940410][1873:1873] CHIP:CTL: Shutting down the commissioner
[1674065267.940445][1873:1873] CHIP:CTL: Stopping commissioning discovery over DNS-SD
[1674065267.940593][1873:1873] CHIP:CTL: Shutting down the controller
[1674065267.940629][1873:1873] CHIP:CTL: Shutting down the System State, this will tear
down the CHIP Stack
[1674065267.941301][1873:1873] CHIP:DMG: IM WH moving to [Uninitialized]
[1674065267.941343][1873:1873] CHIP:DMG: IM WH moving to [Uninitialized]
[1674065267.941473][1873:1873] CHIP:DMG: IM WH moving to [Uninitialized]
[1674065267.941508][1873:1873] CHIP:DMG: IM WH moving to [Uninitialized]
[1674065267.941543][1873:1873] CHIP:DMG: All ReadHandler-s are clean, clear GlobalDiry
st
[1674065267.941687][1873:1873] CHIP:BLE: BleConnectionDelegate::CancelConnection is not
implemented.
[1674065267.942032][1873:1873] CHIP:DL: writing settings to file (/tmp/chip_counters.ini
-E2kTK2)
[1674065267.942942][1873:1873] CHIP:DL: renamed tmp file to file (/tmp/chip_counters.ini
)
[1674065267.943096][1873:1873] CHIP:DL: NVS set: chip-counters/total-operational-hours =
0 (0x0)
```

Binding Table – Write Multiple Command

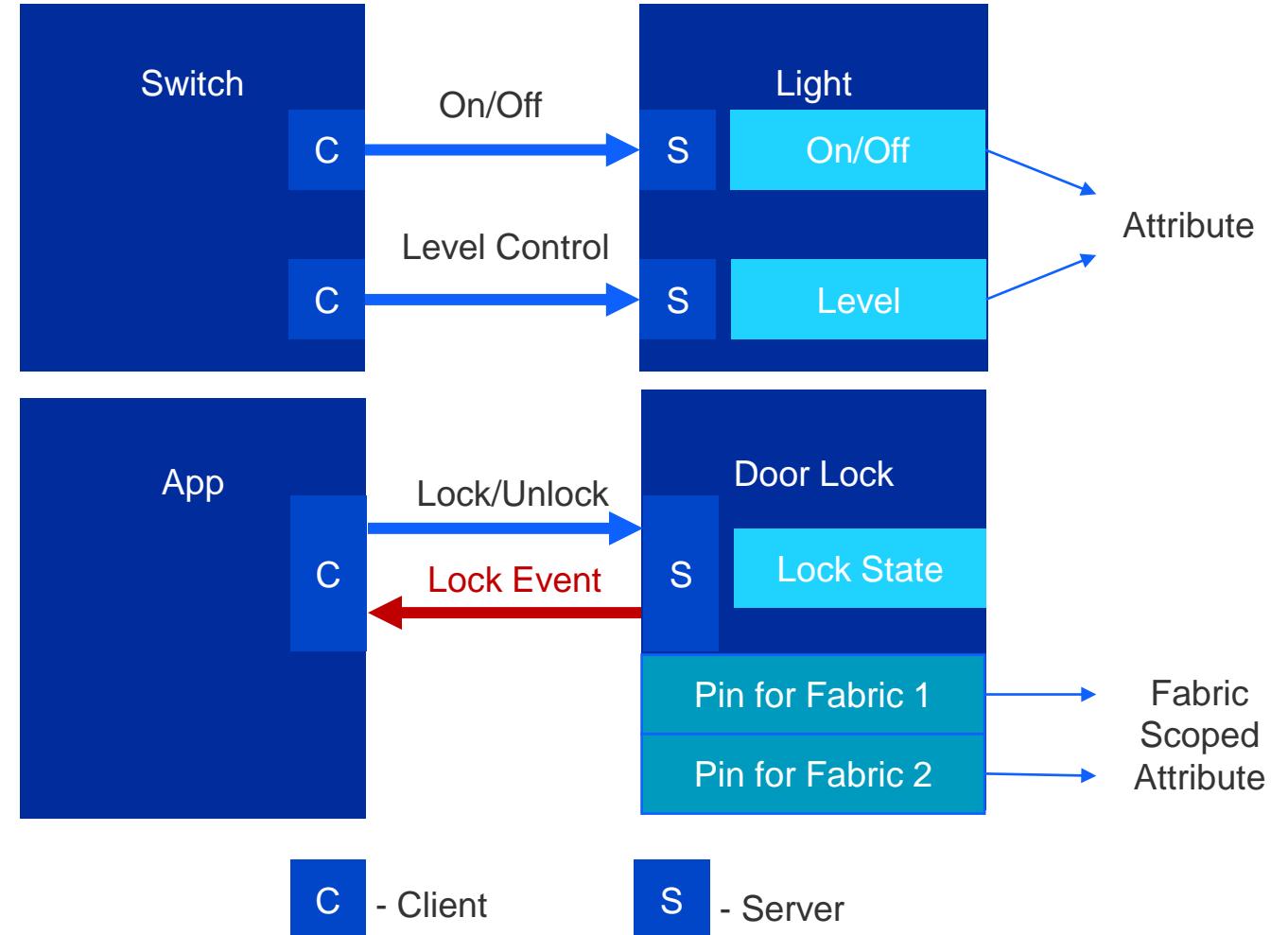
- To write multiple entries to the binding table in a switch:

- `mattertool binding write binding '[{ "fabricIndex" : 1 , "node" : 100 , "endpoint" : 1 , "cluster" : 6 } , { "fabricIndex" : 1 , "node" : 101 , "endpoint" : 1 , "cluster" : 6 }]' 200 1`

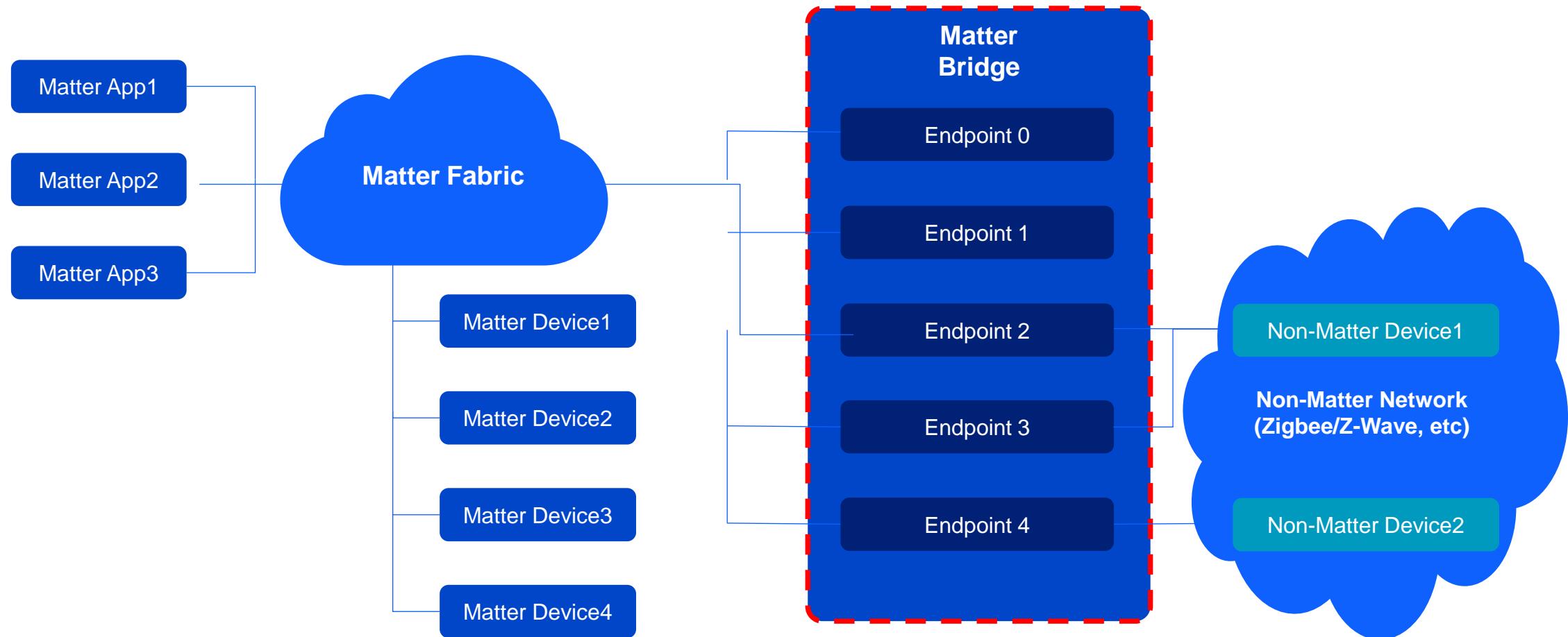
- Where:
 - **100** and **101** are the nodeids of the lights to be bound
 - **1** is the application endpoint in the light, this is always 1
 - **6** is the on/off cluster in the light, this is always 6
 - **200** is the nodeid of the switch for the binding table entry
 - **1** is the application endpoint in the switch that holds the binding table, this is always 1
- When using this command, the whole binding table is replaced
- When BTN1 is pressed the switch transmits the on/off toggle command to all devices in the binding table

Data Model - Cluster

- Client/Server communication model
 - Attributes
 - Commands
 - Events
- Inherited from Zigbee Cluster Library (ZCL)
- Security related attributes are fabric scoped

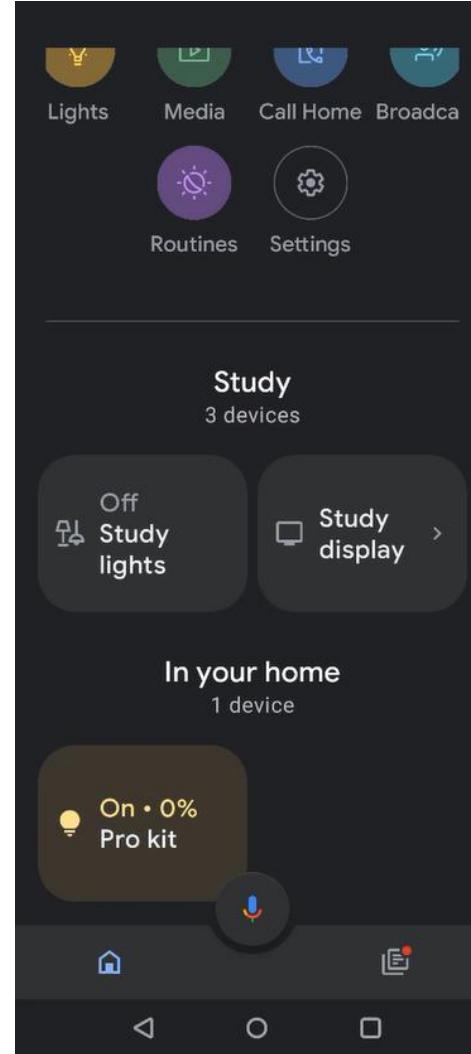
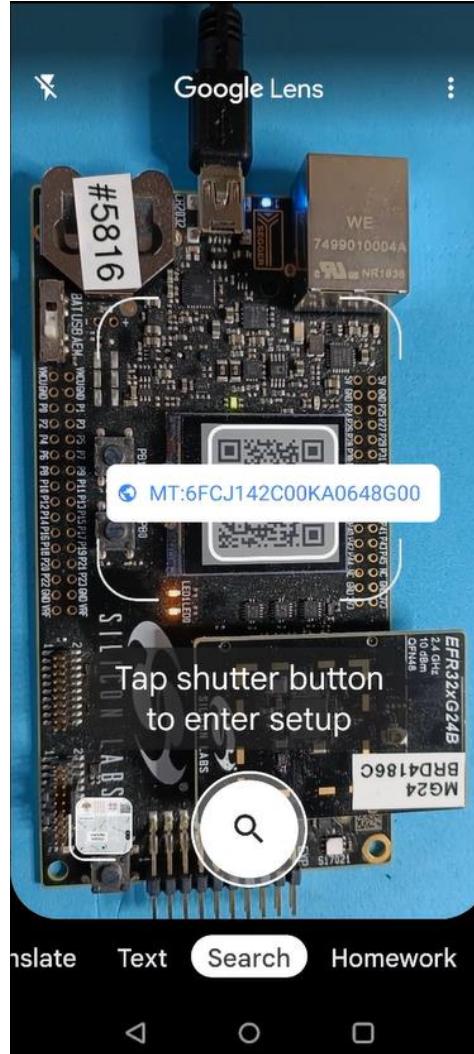


Bridge for Non-Matter Devices



Google Home

- To add to Google Home:
- Ensure the board is in advertising mode
- Scan the QR code using Google Lens
- Follow the prompts to add the board to the Home network
- Used voice control to control the board
- You may need to enable your account to use test devices, instructions can be found at:
[https://
developers.home.google.com/
matter/project/create](https://developers.home.google.com/matter/project/create)

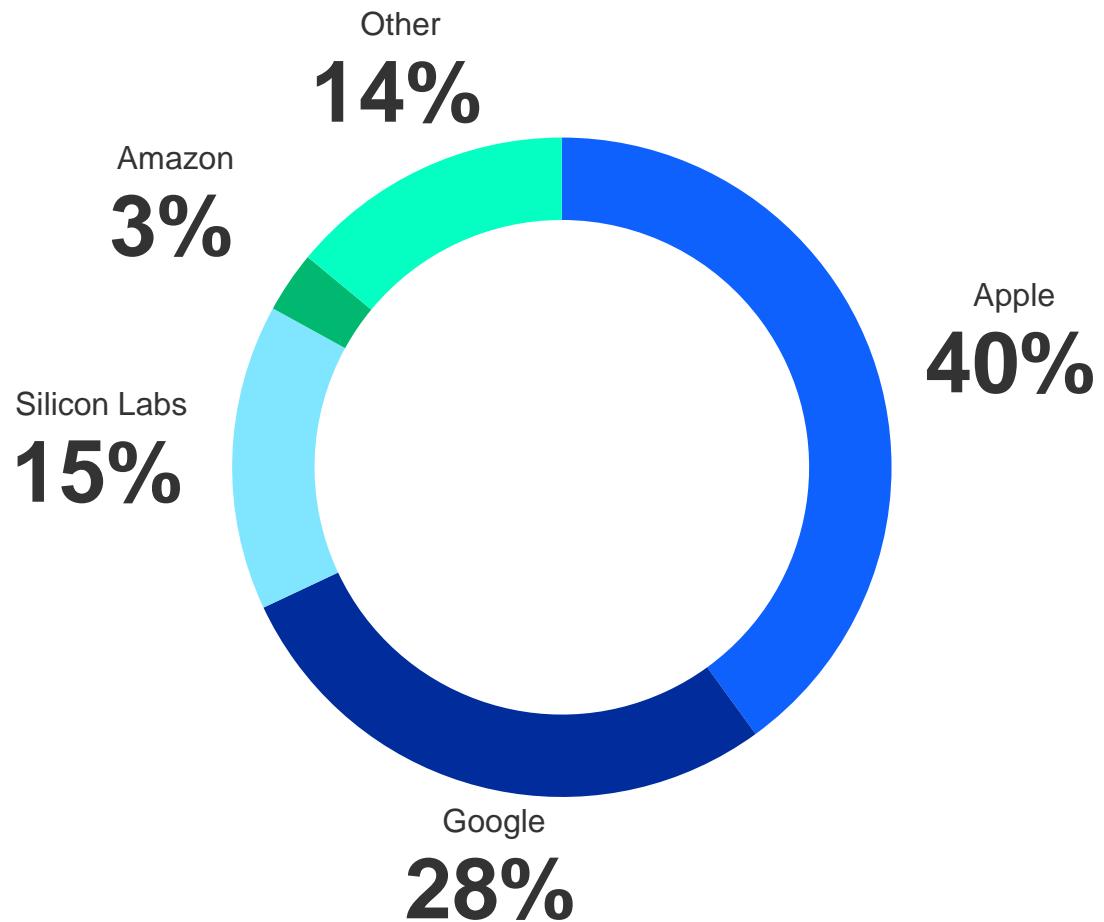


Matter GitHub and GSDK Offerings – Our Plan

	Item	 CSA matter GitHub	 SILICON LABS matter GitHub	 Si ⁵ Studio & GSDK Matter
Development	Thread Part Support	Yes	Yes	Yes
	Wi-Fi Part Support	Yes	Yes	Yes (Studio)
	Developer Platforms	MacOS, Linux	MacOS, Linux	Windows, MacOS Linux
	Studio Tools Support		Limited	Full
	Memory Optimizations		Limited	Full
	Core Protocol Stack	Source Code	Source Code	Pre-built Compliant Library
QA	Production Testing		Yes	Yes
	Performance Testing		Limited	Yes
Certification	Thread Certified Libraries	Yes	Yes	Yes
	Matter Compliance Testing		Yes	Yes
Support	Application Engineering Support	Limited	Full	Full

Silicon Labs Proven IoT Expertise

Matter Contributors by Lines of Code Submitted
as of 07-05-2022

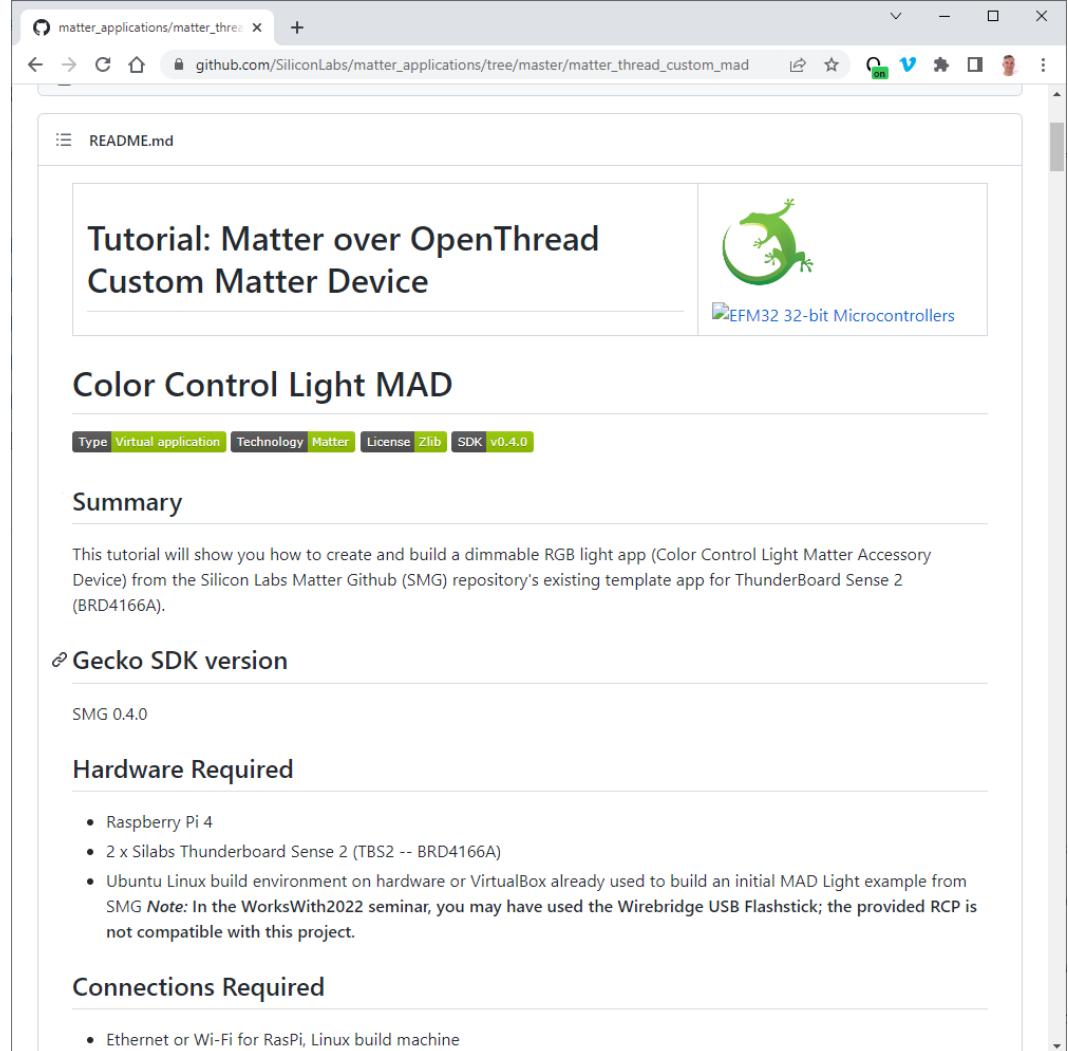


Silicon Labs is the 3rd largest contributor to Matter GitHub

- [**ZCL parser**](#) (ZAP tool)
 - Replaces Application Builder
 - Standalone, no longer tied to Studio
- **Ported Zigbee application framework**
- **Added support for Matter multicast and OTA Bootload, and Light Switch sample apps**
- [**Light, Switch, lock, and window covering examples**](#)
 - MG12 and MG24 support
 - Wi-Fi and Thread

Next Steps

- **Simplicity Studio** has additional examples and documentation
- **Silicon Labs Matter Github:**
 - <https://github.com/SiliconLabs/matter>
 - Further examples and documentation
 - Pre-built bootloaders and binaries
 - Raspberry Pi OTBR image
- **Silicon Labs Github Matter Applications:**
 - https://github.com/SiliconLabs/matter_applications
 - A growing selection of Matter devices presented in tutorial format
- **Community**
 - <https://community.silabs.com/> > Forums > Matter
 - There is lots of useful information in the **Articles** tab



The screenshot shows a web browser window displaying a GitHub repository page for a Matter application template. The URL in the address bar is github.com/SiliconLabs/matter_applications/tree/master/matter_thread_custom_mad. The page title is "matter_thread_custom_mad". The main content is a README.md file titled "Tutorial: Matter over OpenThread Custom Matter Device". It features a green lizard logo and the text "EFM32 32-bit Microcontrollers". Below the title, there are sections for "Color Control Light MAD" (Virtual application, Technology: Matter, License: Zlib, SDK: v0.4.0), "Summary" (describing how to create a dimmable RGB light app), "Gecko SDK version" (SMG 0.4.0), "Hardware Required" (Raspberry Pi 4, 2 x Silabs Thunderboard Sense 2 (TBS2 -- BRD4166A), Ubuntu Linux build environment), and "Connections Required" (Ethernet or Wi-Fi for RasPi, Linux build machine). A note at the bottom states: "SMG Note: In the WorksWith2022 seminar, you may have used the Wirebridge USB Flashstick; the provided RCP is not compatible with this project."

Please help us improve future training by completing our short survey:
<https://www.surveymonkey.com/r/matter-ew23>



Thank You

- This is part 3/3 of the Silicon Labs Matter-over-Thread workshop series for Embedded World 2023
View online at:
 1. <https://www.brainshark.com/siliconlabs/EW23-Simplicity-Studio-Install>
 2. <https://www.brainshark.com/siliconlabs/EW23-Bootloader-SS>
 3. <https://www.brainshark.com/siliconlabs/EW23-Matter-SS>
- Presentations and other files can be found alongside the online videos in the **Attachments** tab
- Learn more at the Silicon Labs in Hall 4A, Stands 128 & 129