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

100%
IoT Focused

Bluetooth® Multiprotocol Proprietary
THREAD Wi Fi Wi SUN zigbee
Z-WAVE amazon sidewalk matter
Breadth and Depth of Wireless IoT Protocols

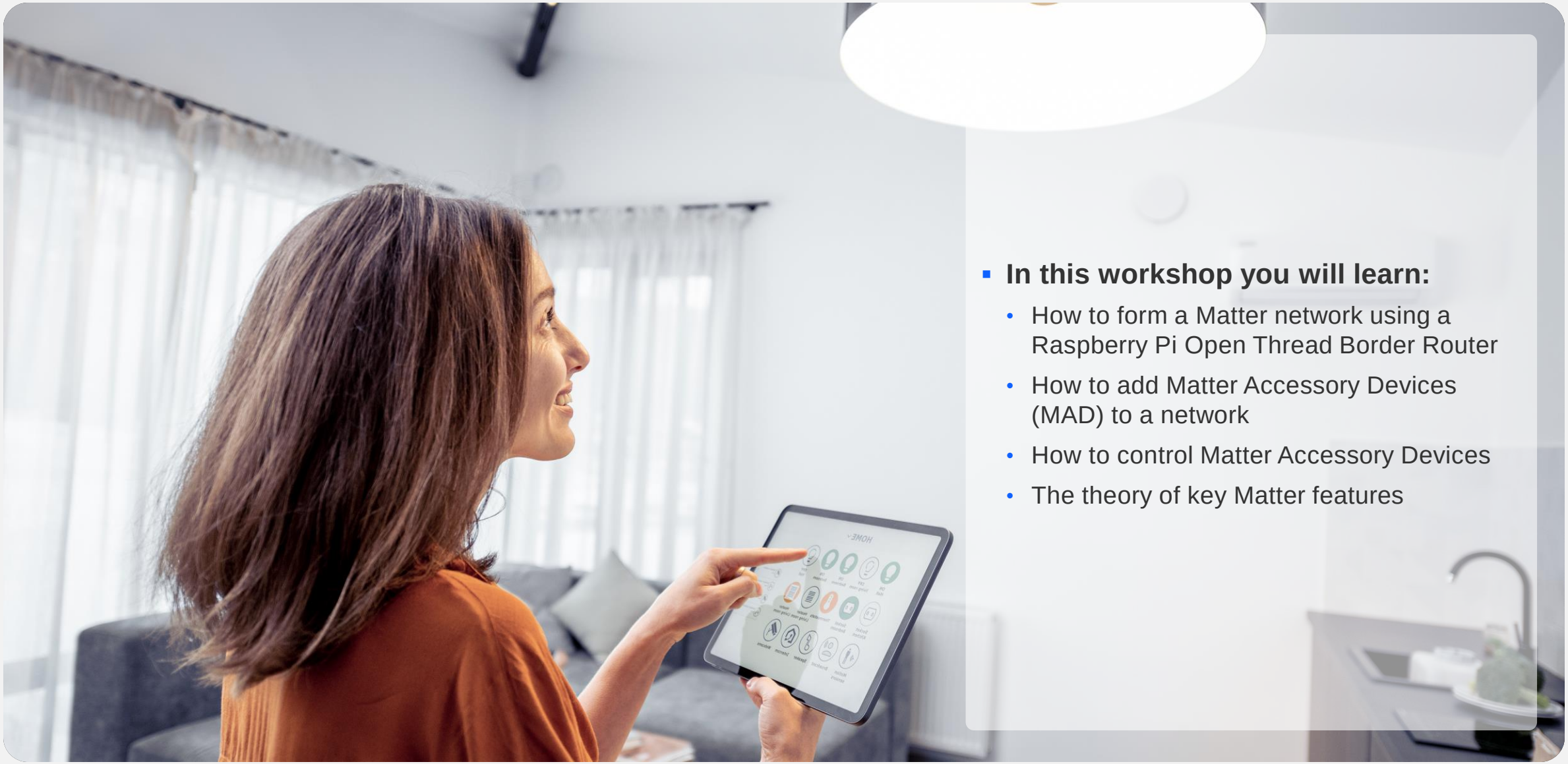
#1
Share in Mesh

1st
To Market with Multiprotocol, Bluetooth LE, Bluetooth mesh

Innovation
Performance, Power, CoEx, Modules, Secure Vault™

ember	ENERGY micro	blue giga	telegesis	Micriμm®	ZENTRI	Z-WAVE	REDPINE SIGNALS
2012	**2013**	**2015**	**2015**	**2016**	**2017**	**2018**	**2020**
Software ZigBee SoC	Low-power 32-bit MCUs	BT Smart Modules	ZigBee/Thread Modules	Software RTOS	Cloud Connected Wi-Fi®	Smart Home Protocol	Ultra Low Power Wi-Fi®

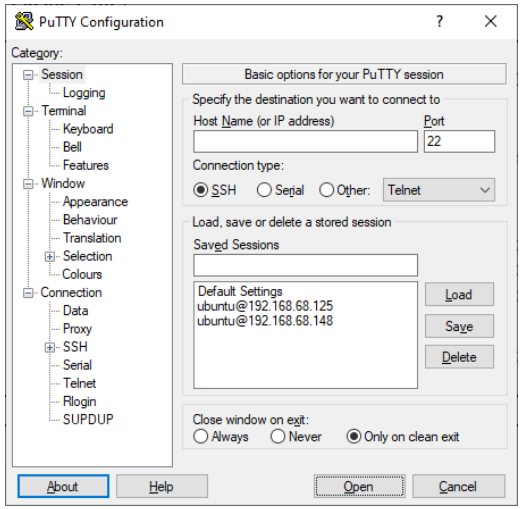
Overview

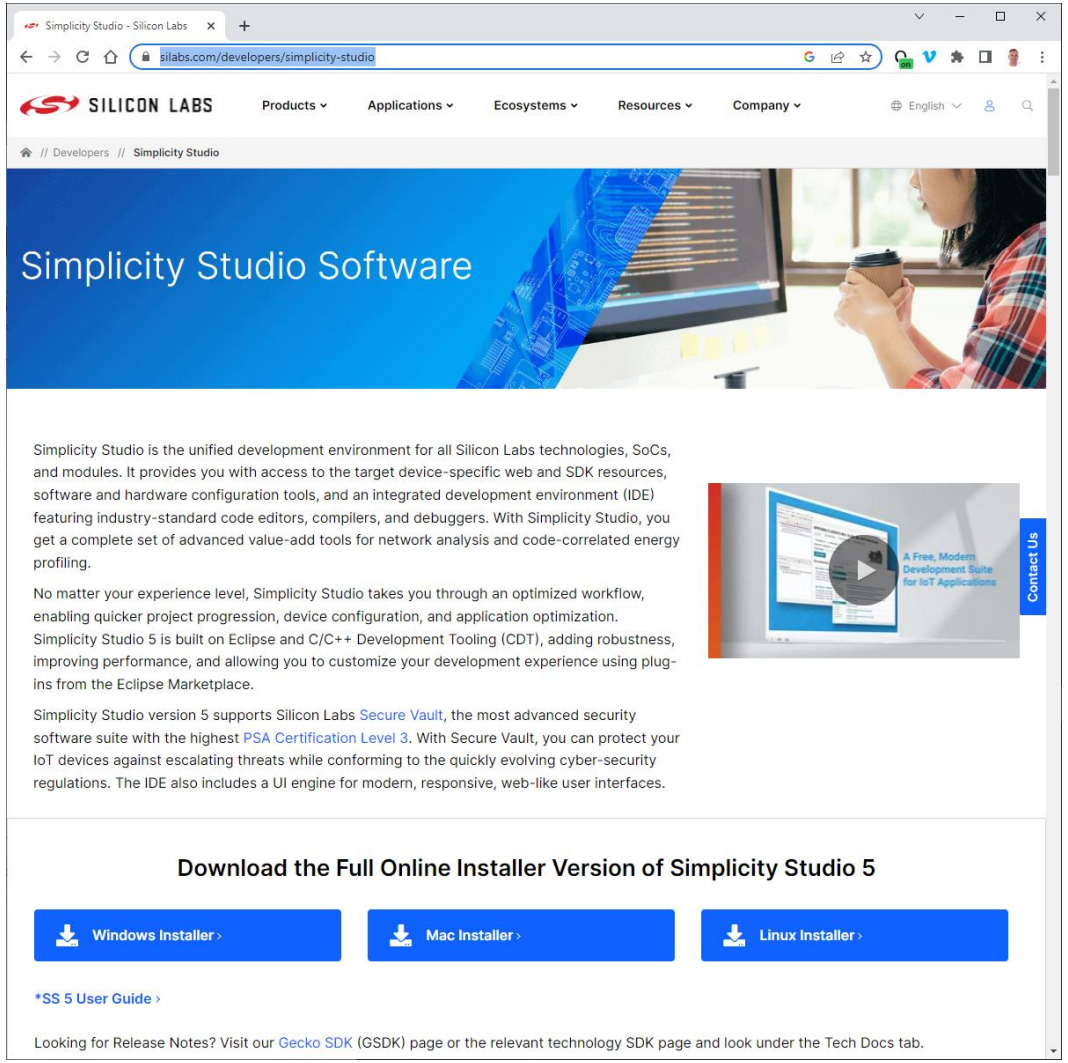


- **In this workshop you will learn:**
  - How to form a Matter network using a Raspberry Pi Open Thread Border Router
  - How to add Matter Accessory Devices (MAD) to a network
  - How to control Matter Accessory Devices
  - The theory of key Matter features

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/





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

SILICON LABS

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

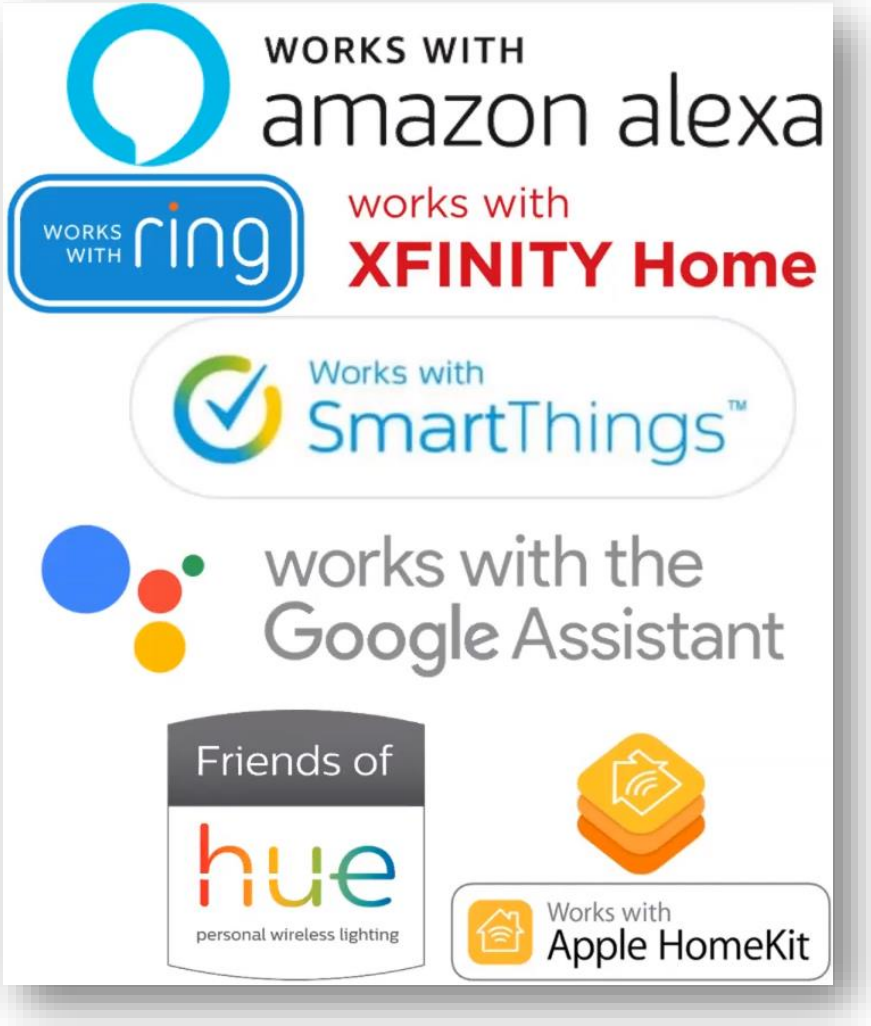
- Extremely hard to mix and match the product they want with their preferred ecosystem
- Very difficult to change once selected

Developers

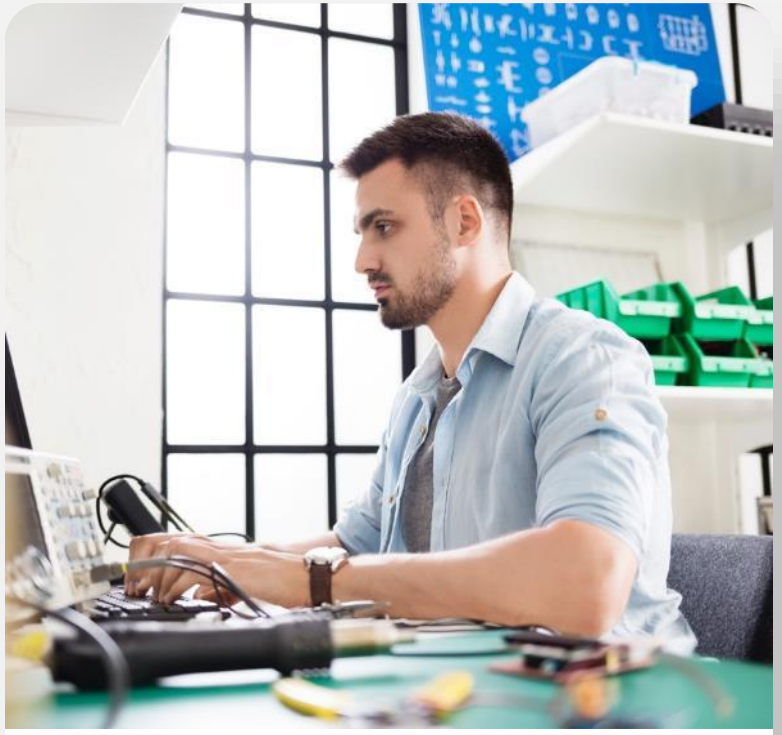
- Developers are forced to pick what ecosystem integrations they support
- Often need to ship multiple SKUs for all connectivity standards
- Need to learn different IoT technologies and ecosystems

Retailers

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



Simplifying IoT Connectivity

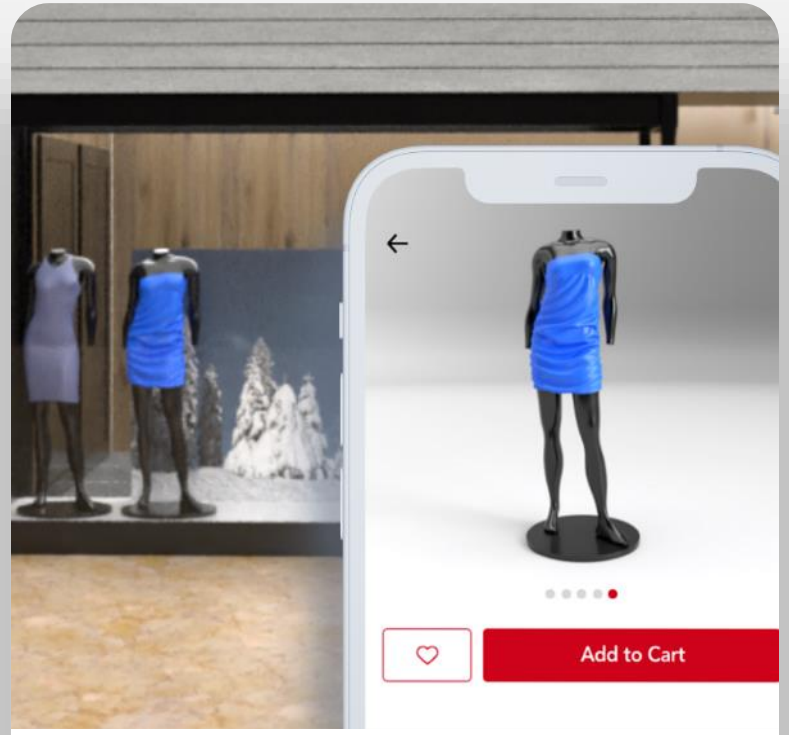


DEVELOPER/MANUFACTURER

Single SKU

Lower development & operational cost

More time for innovation



RETAILER

Requires less shelf space

Lowers inventory cost

Minimizes returns

CONSUMER

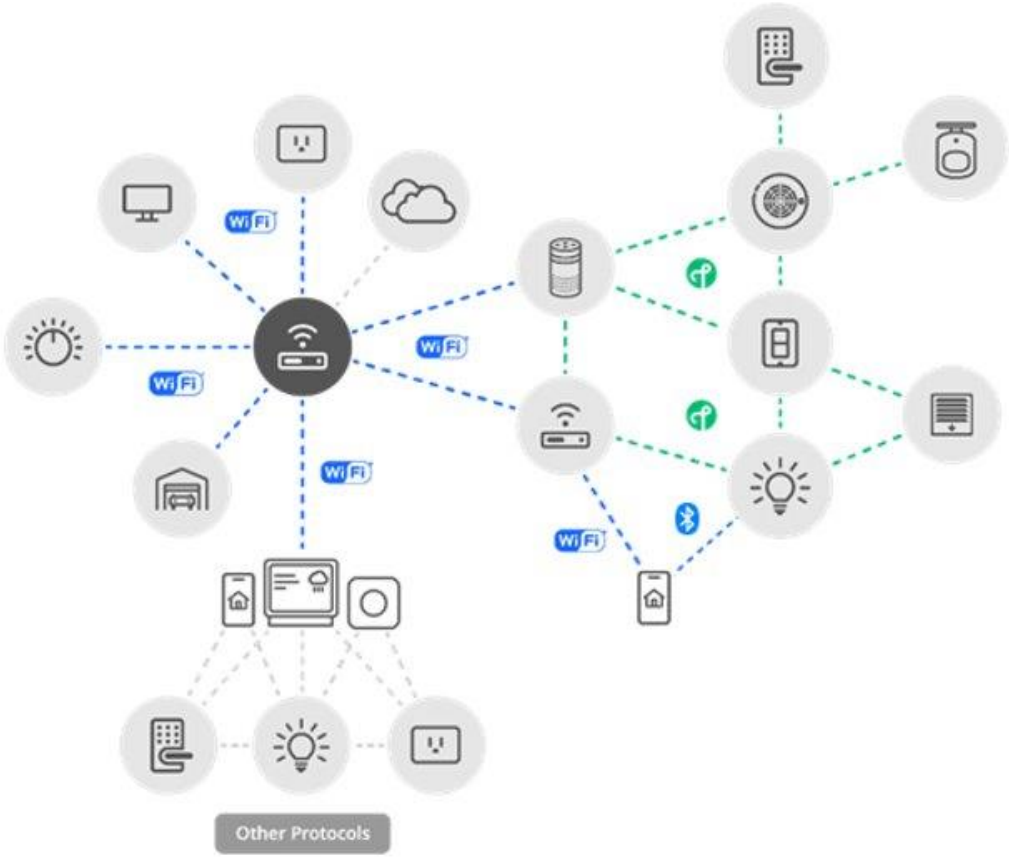
Simplifies purchasing experience

Simplifies setup & control

Provides better user experience

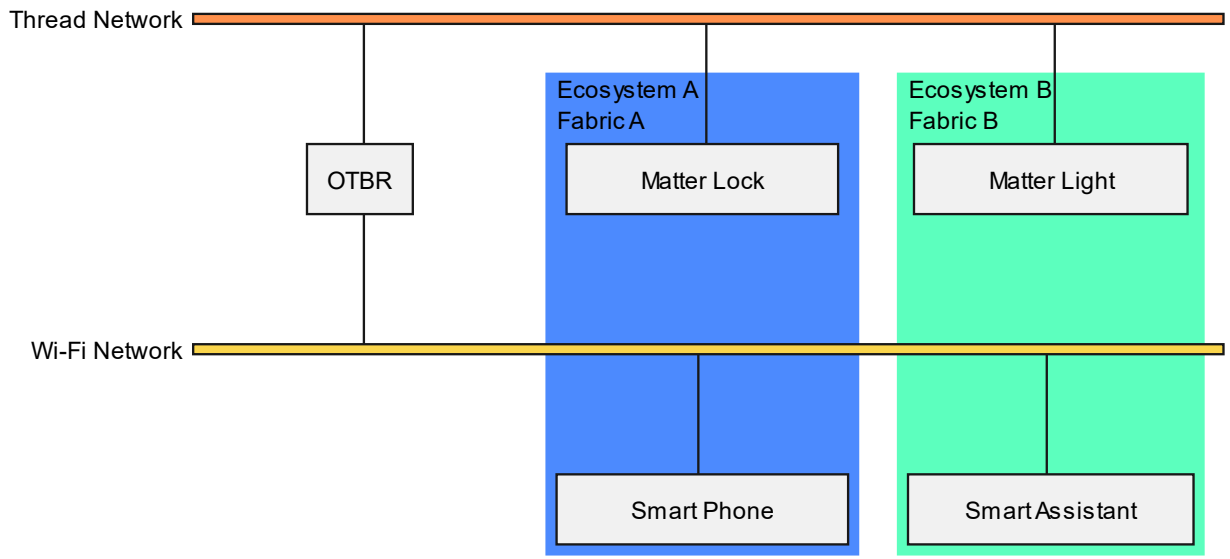
Network Topology

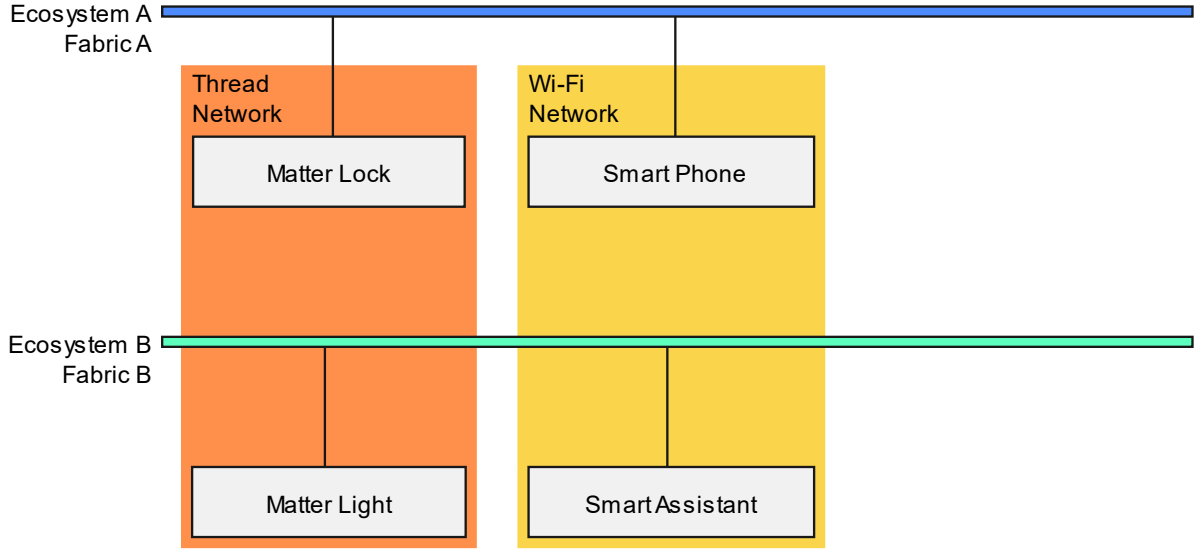
- **Matter wireless protocols:**
  - Matter has native support for Thread and Wi-Fi
  - Matter uses Bluetooth for commissioning
  - Other protocols, like Zigbee and Z-Wave, can be bridged into a Matter network
- **Matter device types:**
  - Matter Accessory Devices (MADs) provide end-node functionality 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

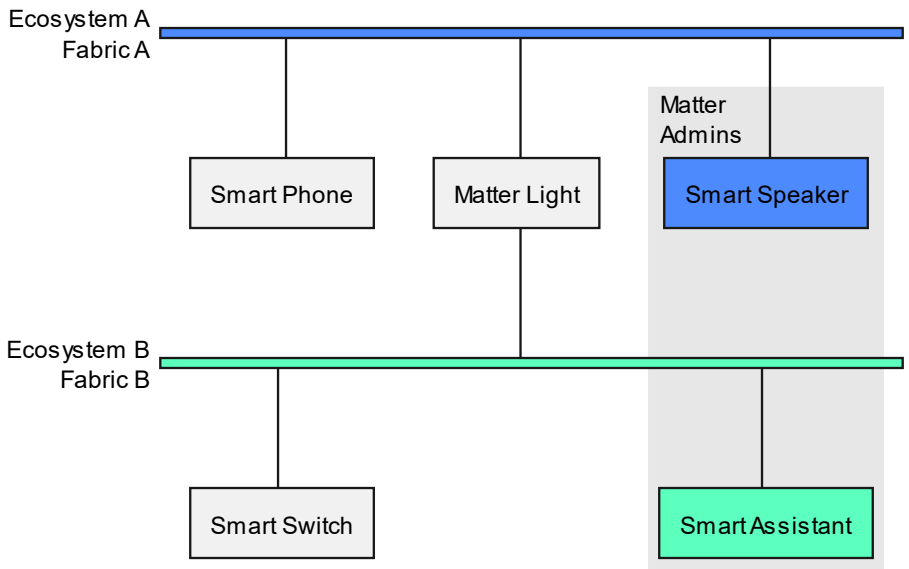
- Multiple transports
  - Matter can work on top of multiple wireless or wired technologies to transport the IP packets
- Fabric
  - A collection of Matter devices sharing a trusted root
  - A fabric is identified by a fabric ID which is a 64-bit number
- Node
  - In a Matter fabric, each physical device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





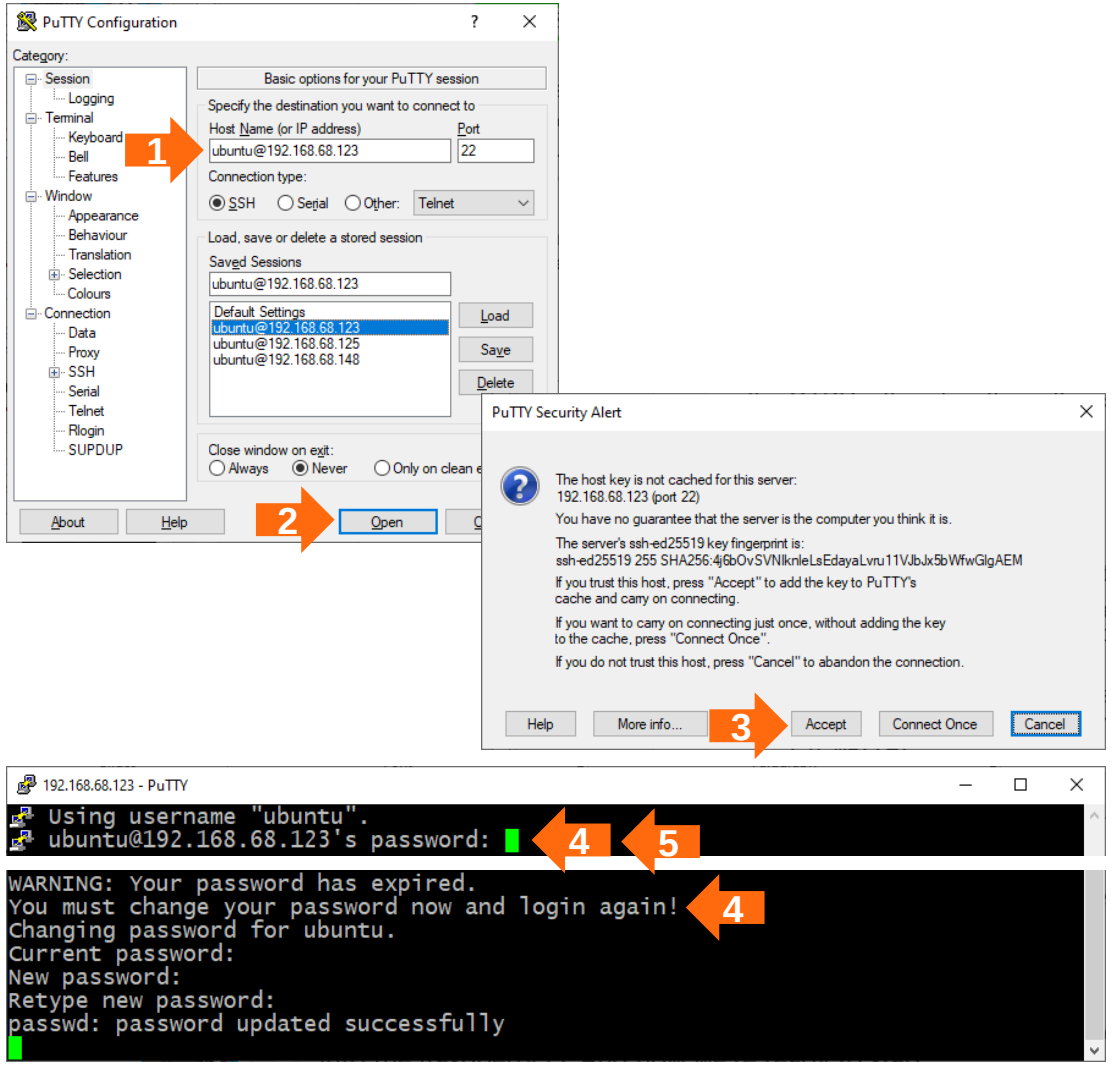
Matter Multi-Admin

- **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**



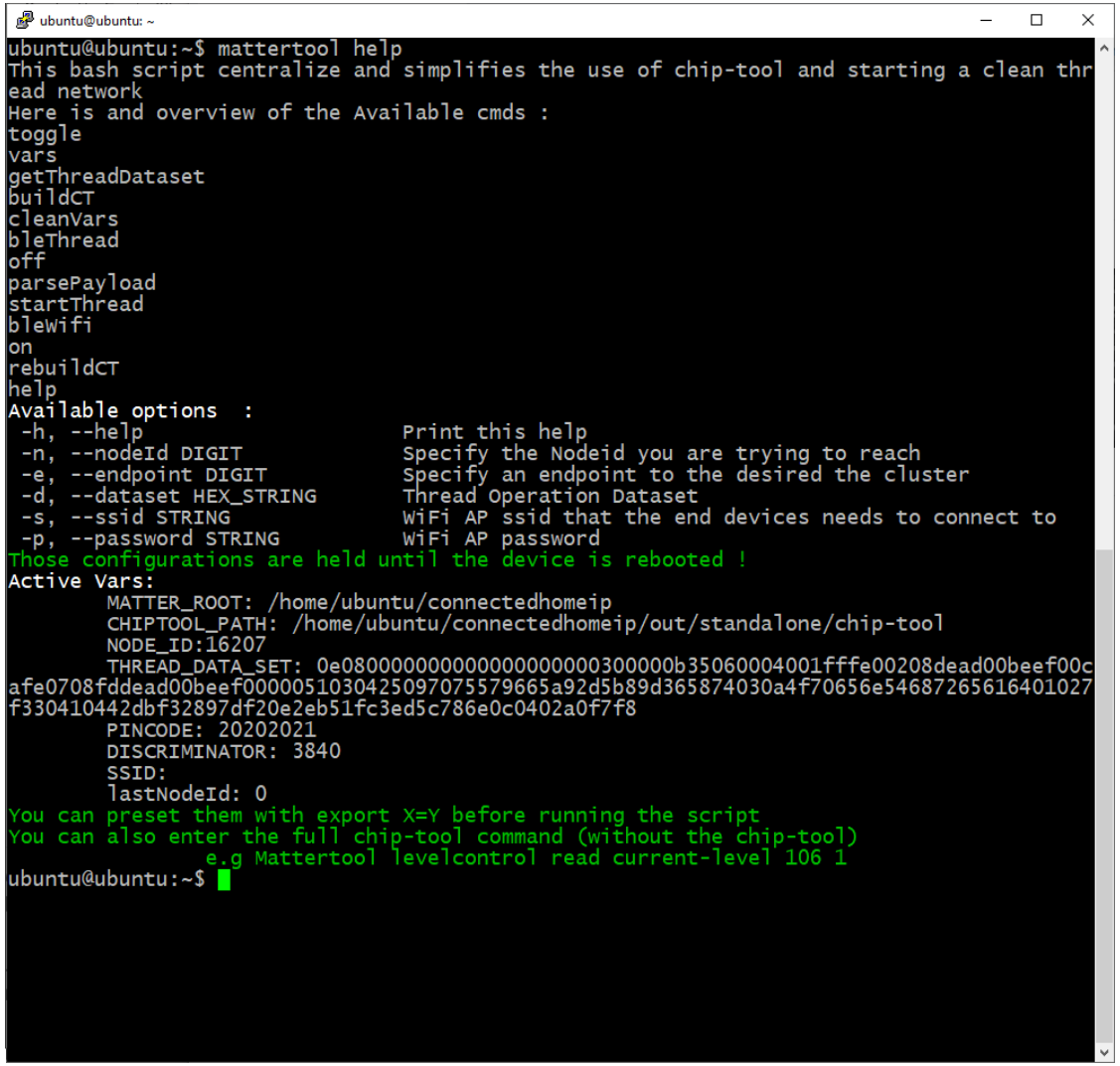
SSH / PuTTY

- The devices will be manipulated via a SSH connection to the OTBR using PuTTY or the command-line
- To open a connection using PuTTY:
  1. 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
  2. Click the **Open** button
  3. The first time connecting to a device using PuTTY, click the **Accept** button on the **PuTTY Security Alert** window
  4. 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`
  5. If you changed the password, reconnect and use the new password, `matter`



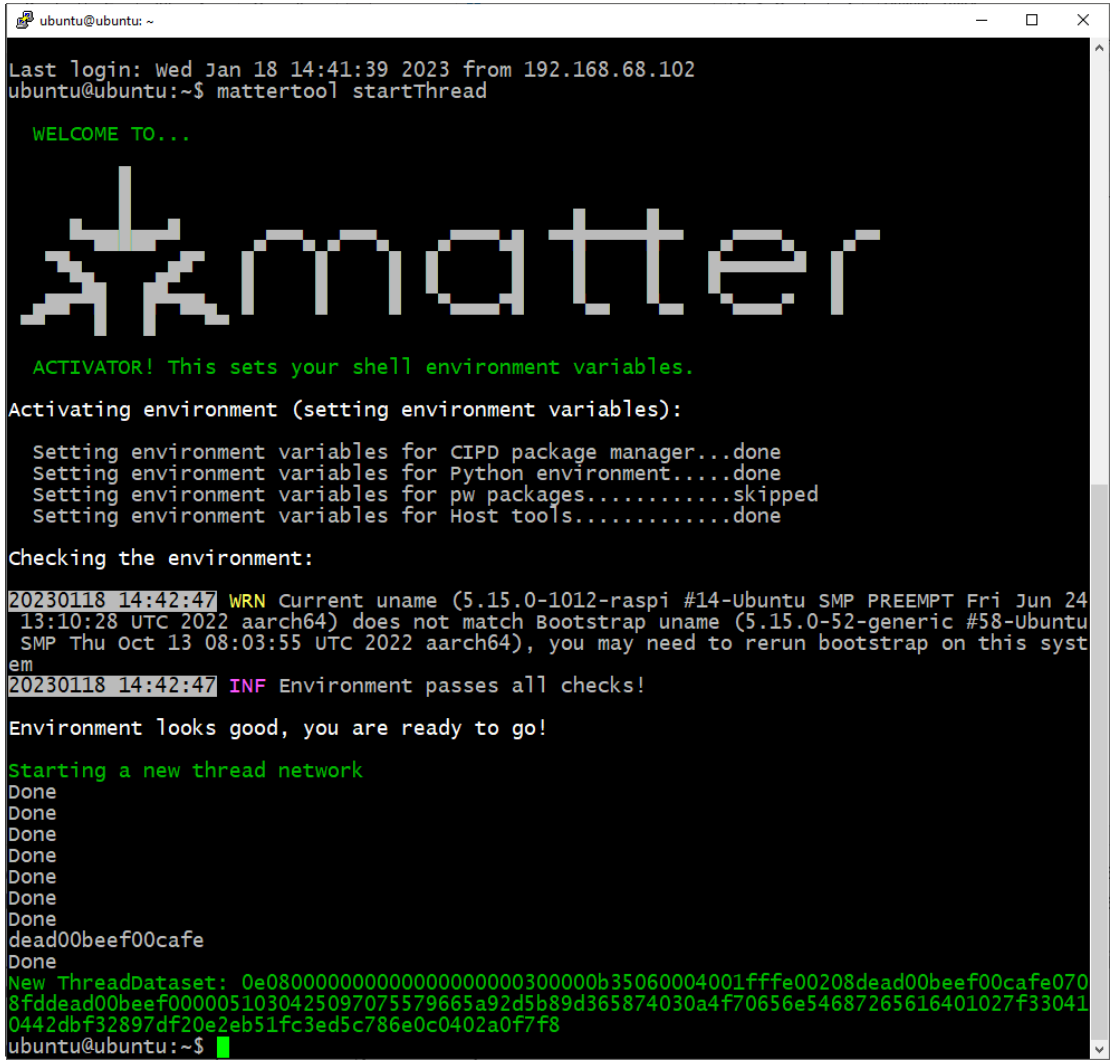
Mattertool and Discriminator

- 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`



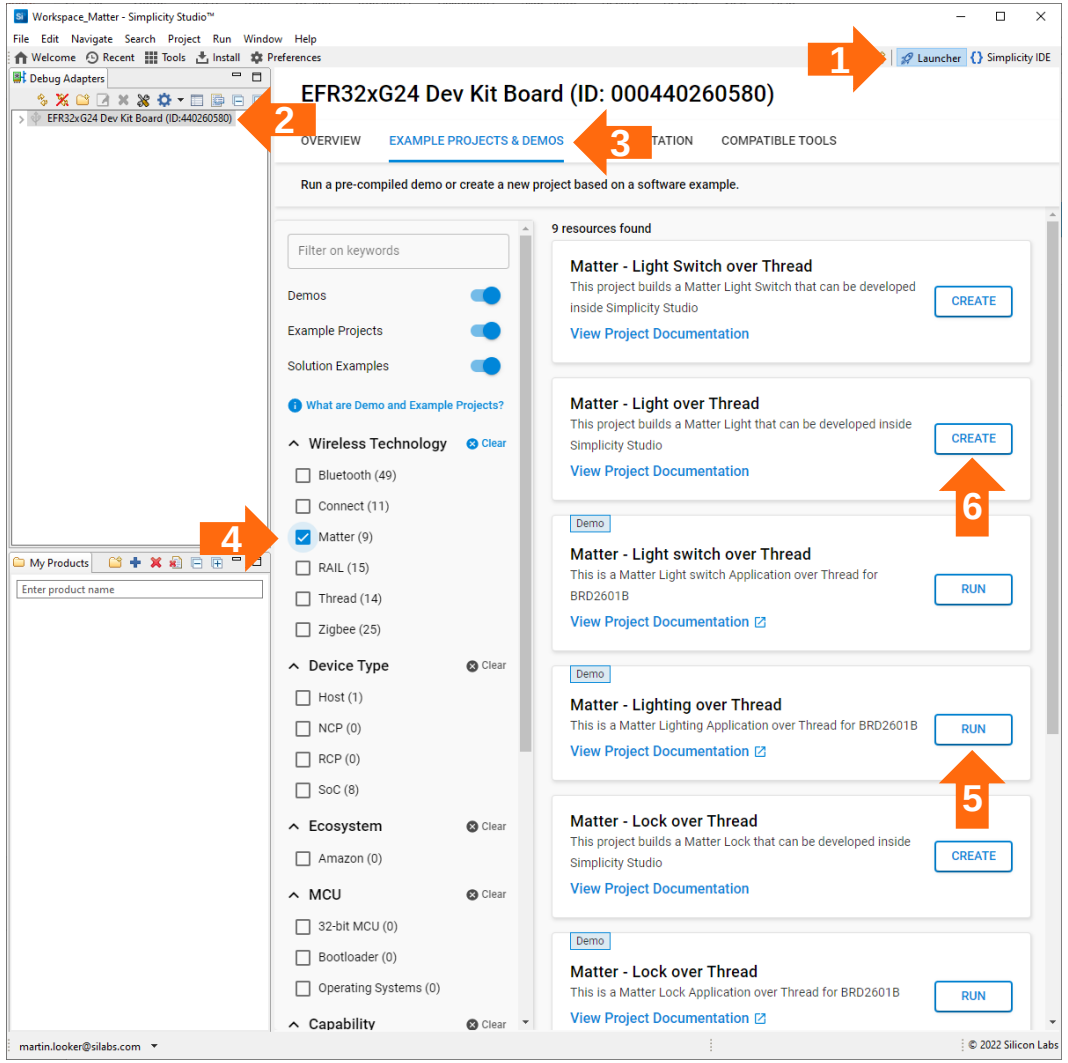
Network Commands

- 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`



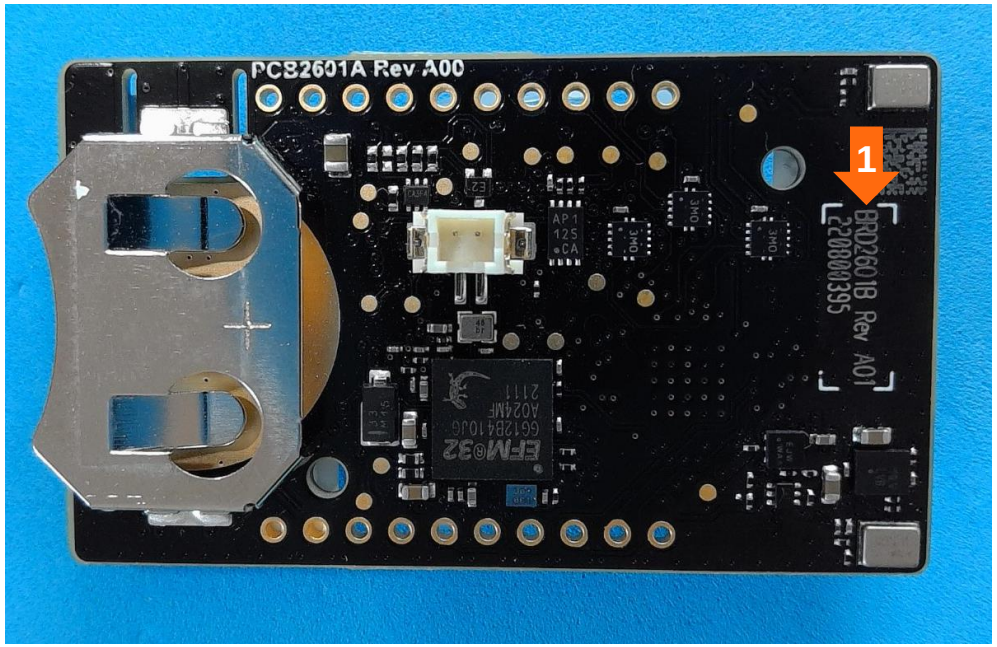
Light Demo / Example

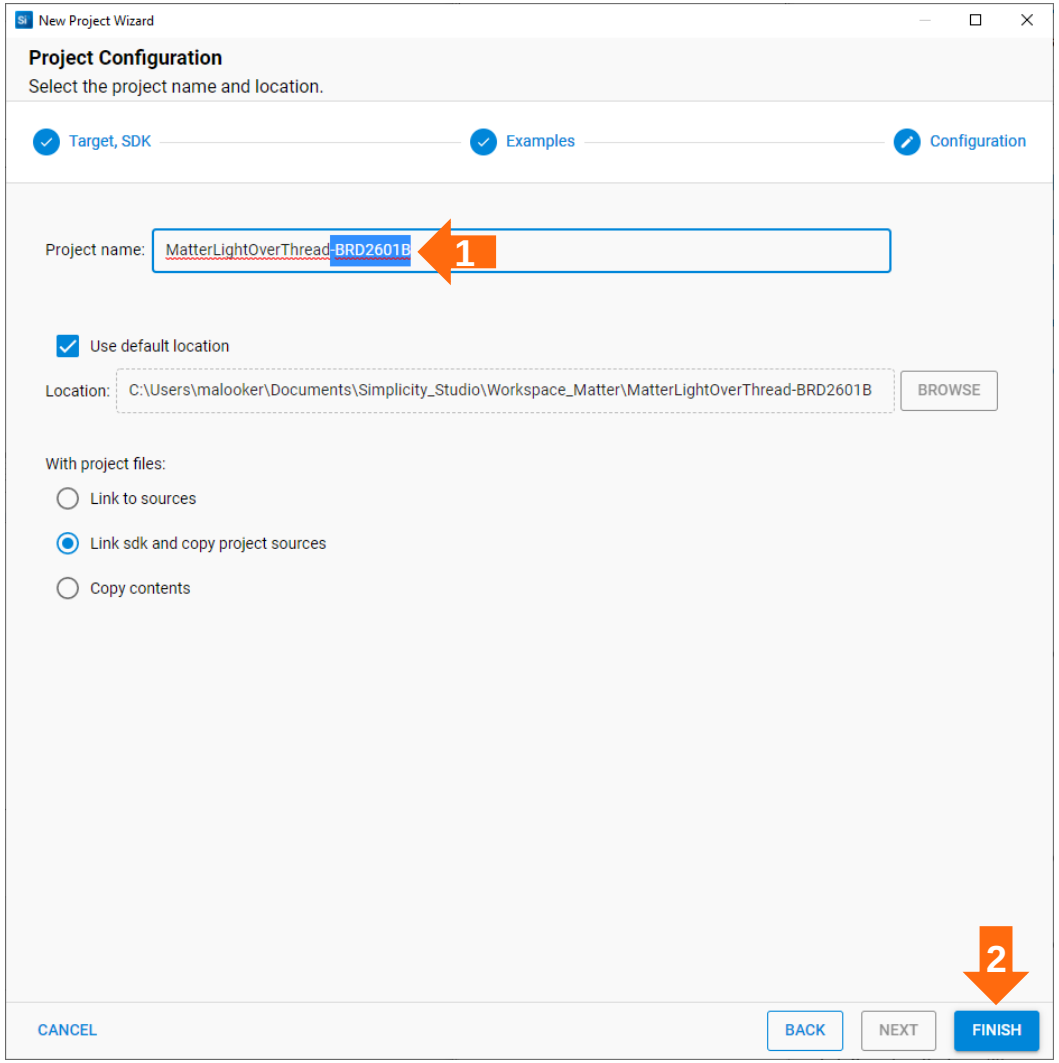
- Connect board via USB
  1. Switch to **Launcher** perspective
  2. Select **EFR32xG24 Dev Kit Board** from **Debug Adapters** panel
     - Make a note of the last 4 digits of the device ID (we will use this as the network ID later)
  3. Select **Example Projects and Demos**
  4. Select **Matter** filter
  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

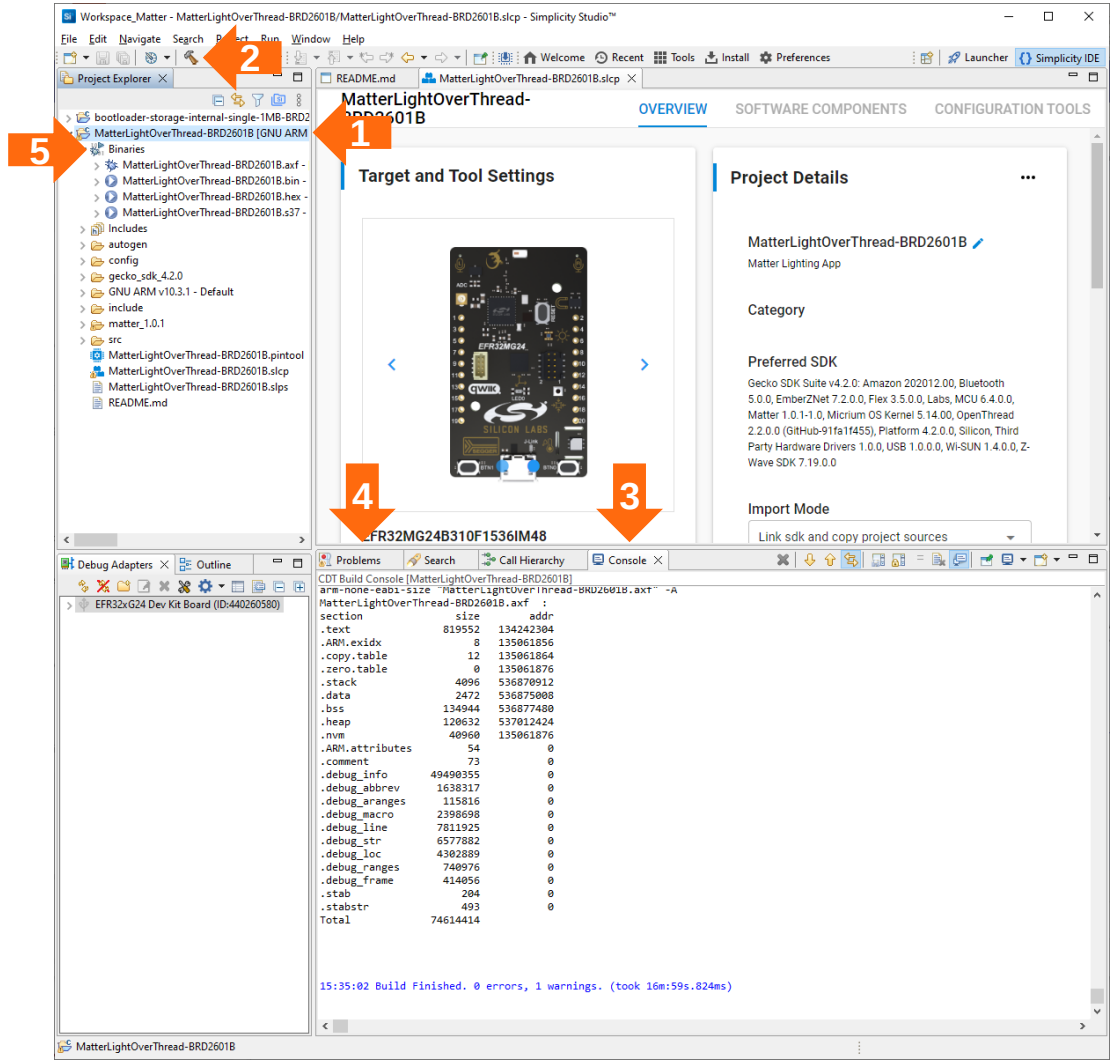
- To complete the wizard:
  1. Append the board number in the **Project Name** edit box
     - Board numbers are printed on each board
     - This helps when the same example is created for different boards
  2. Click the **FINISH** button





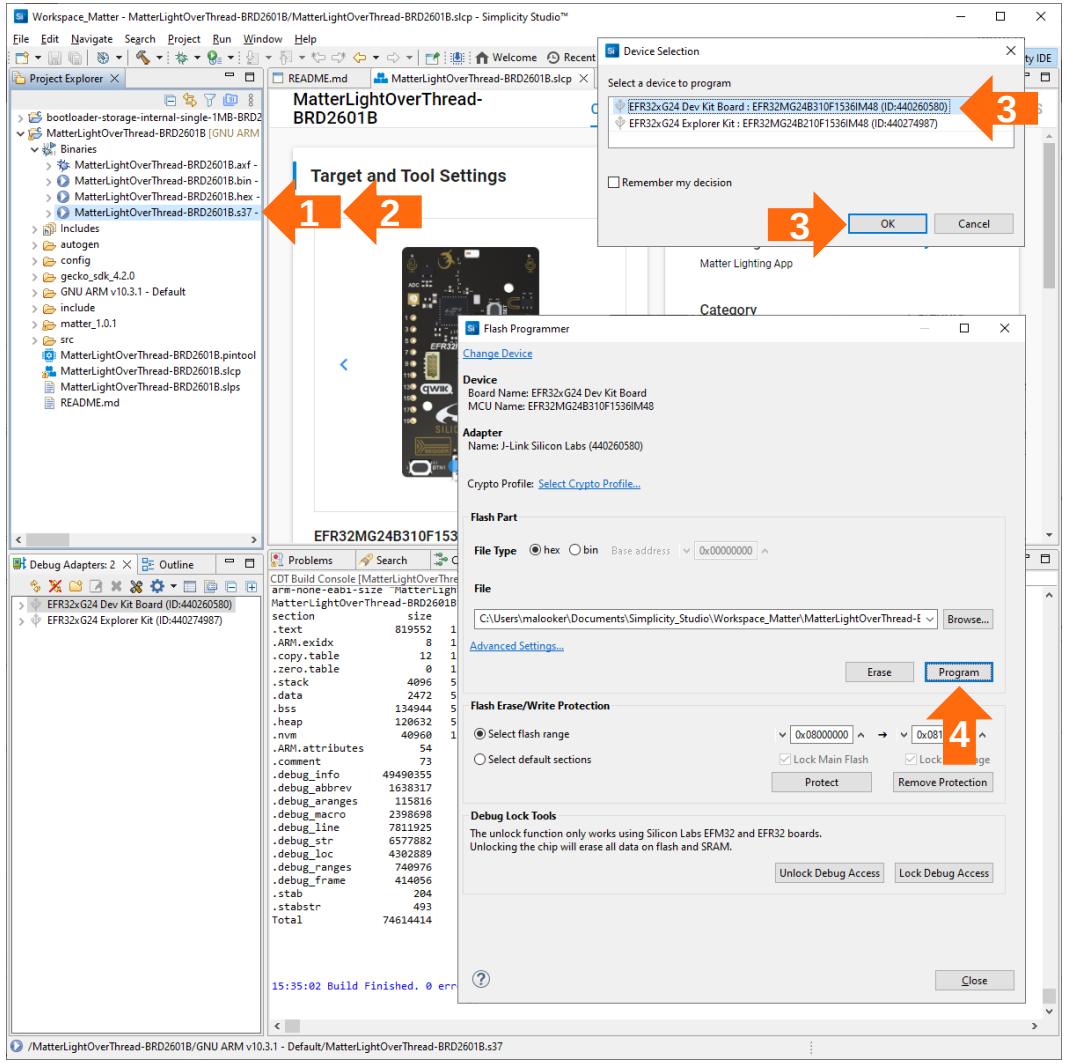
Light Example – Build

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



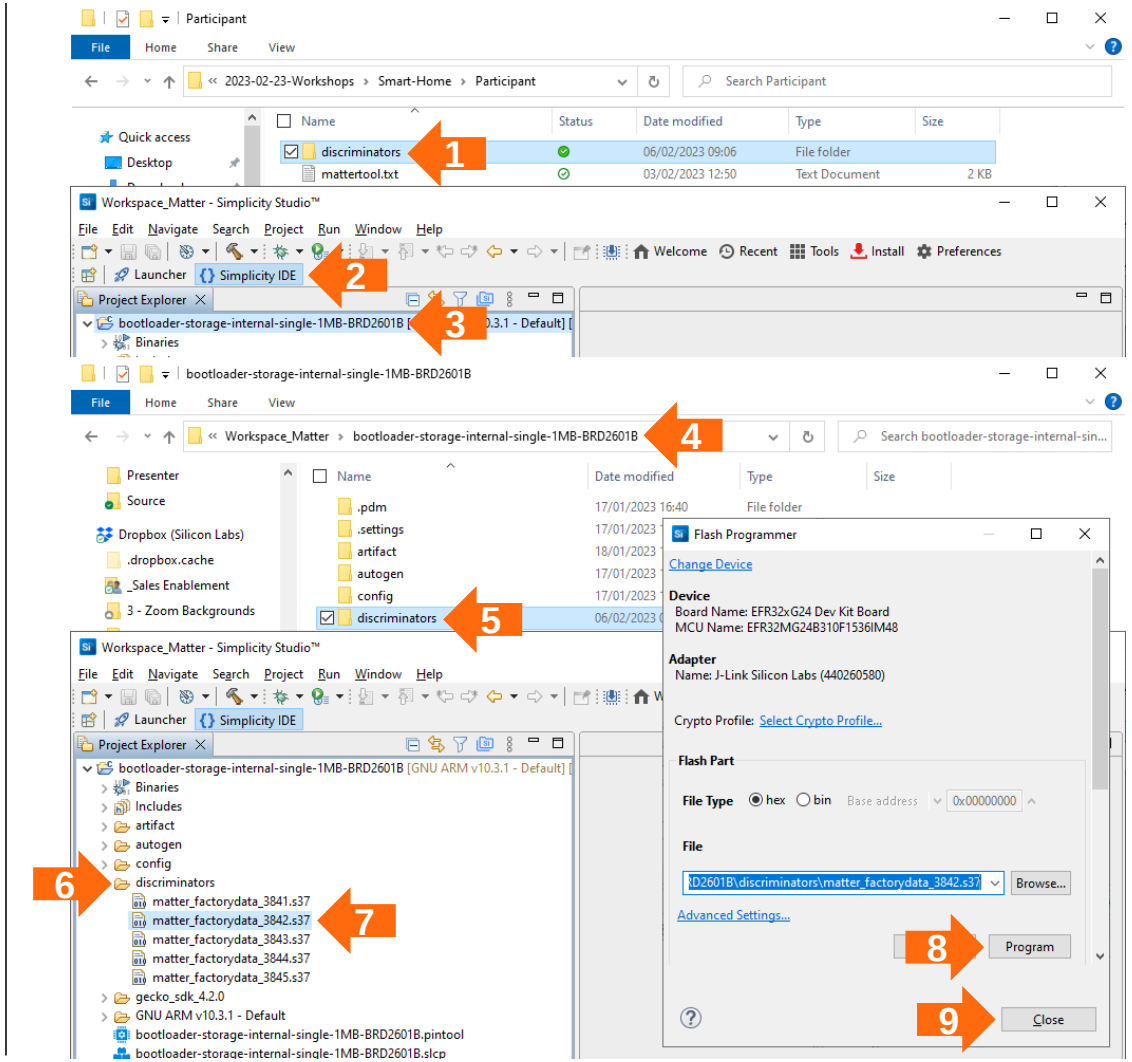
Light Example – Flash

- 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



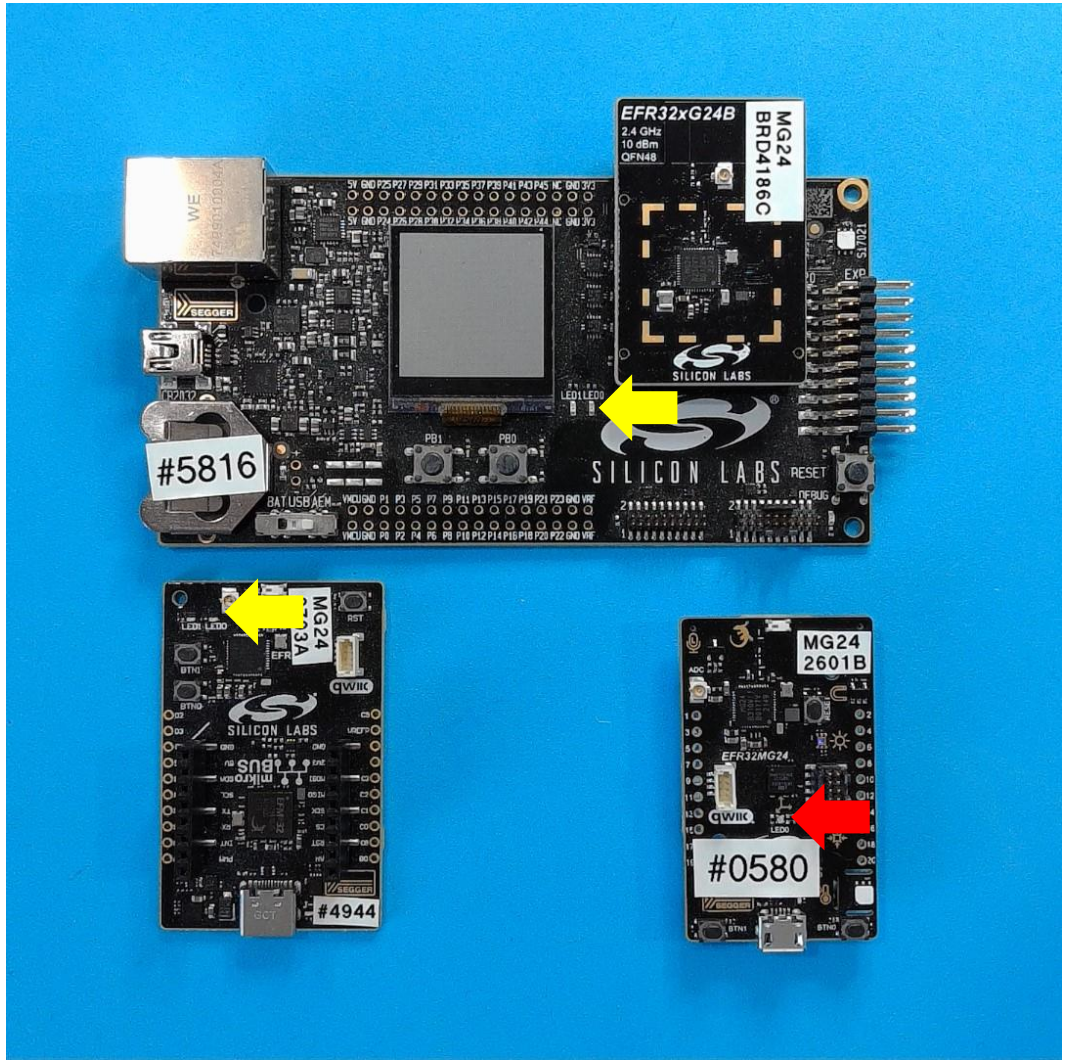
Light – Change Discriminator

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



Light – Commissioning

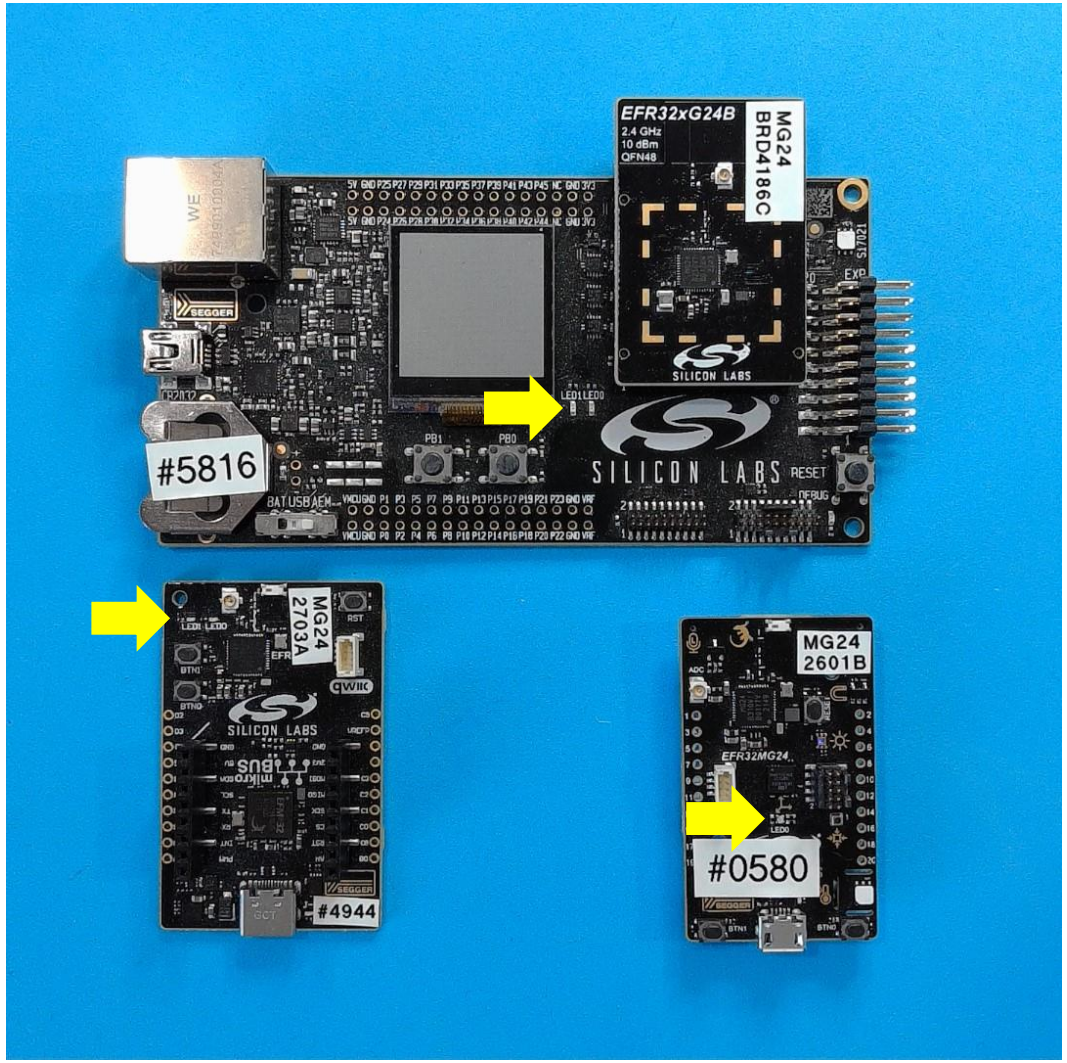
- To commission a new device into the network using BLE (the nodeid for the new device needs to be specified with the -n option):
  1. `mattertool bleThread -n 100`
     where:
     100 is the nodeid for the new device (use the last 4 digits of the device ID, noted earlier, to ensure uniqueness)
- The 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



SILICON LABS

Light – On/Off Cluster Commands

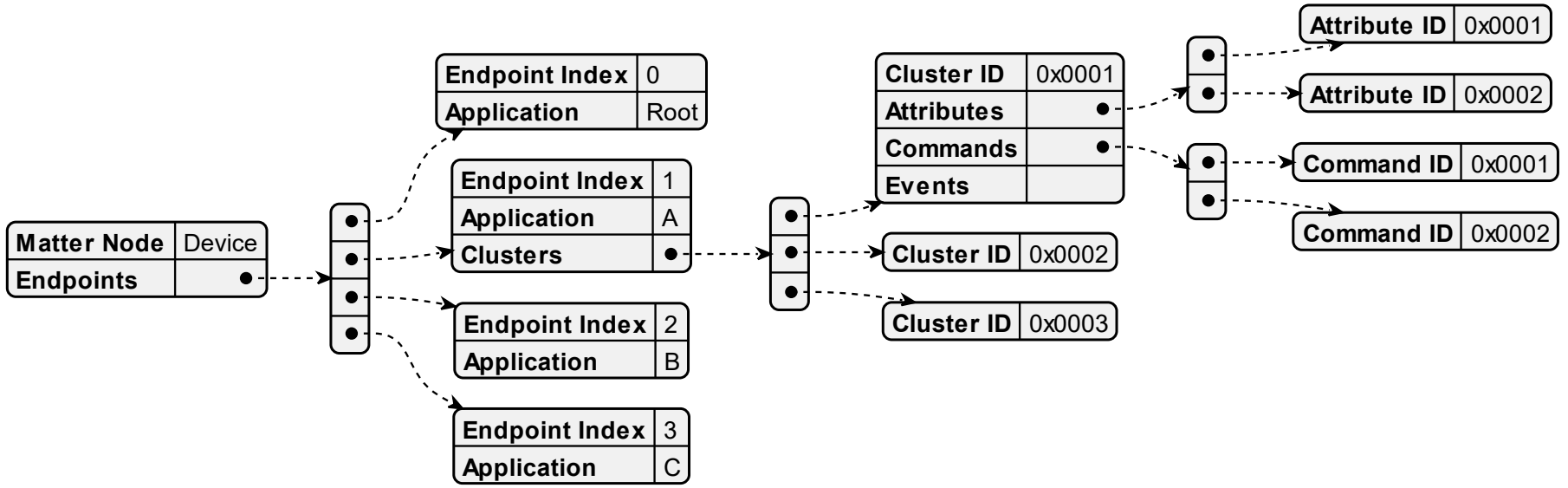
- To turn on a light:
  - `mattertool on -n 100`
    where:
    100 is the nodeid of the light to be turned on
- To turn off a light:
  - `mattertool off -n 100`
    where:
    100 is the nodeid of the light to be turned off
- To toggle a light between on and off:
  - `mattertool toggle -n 100`
    where:
    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



SILICON LABS

Zigbee Cluster Library

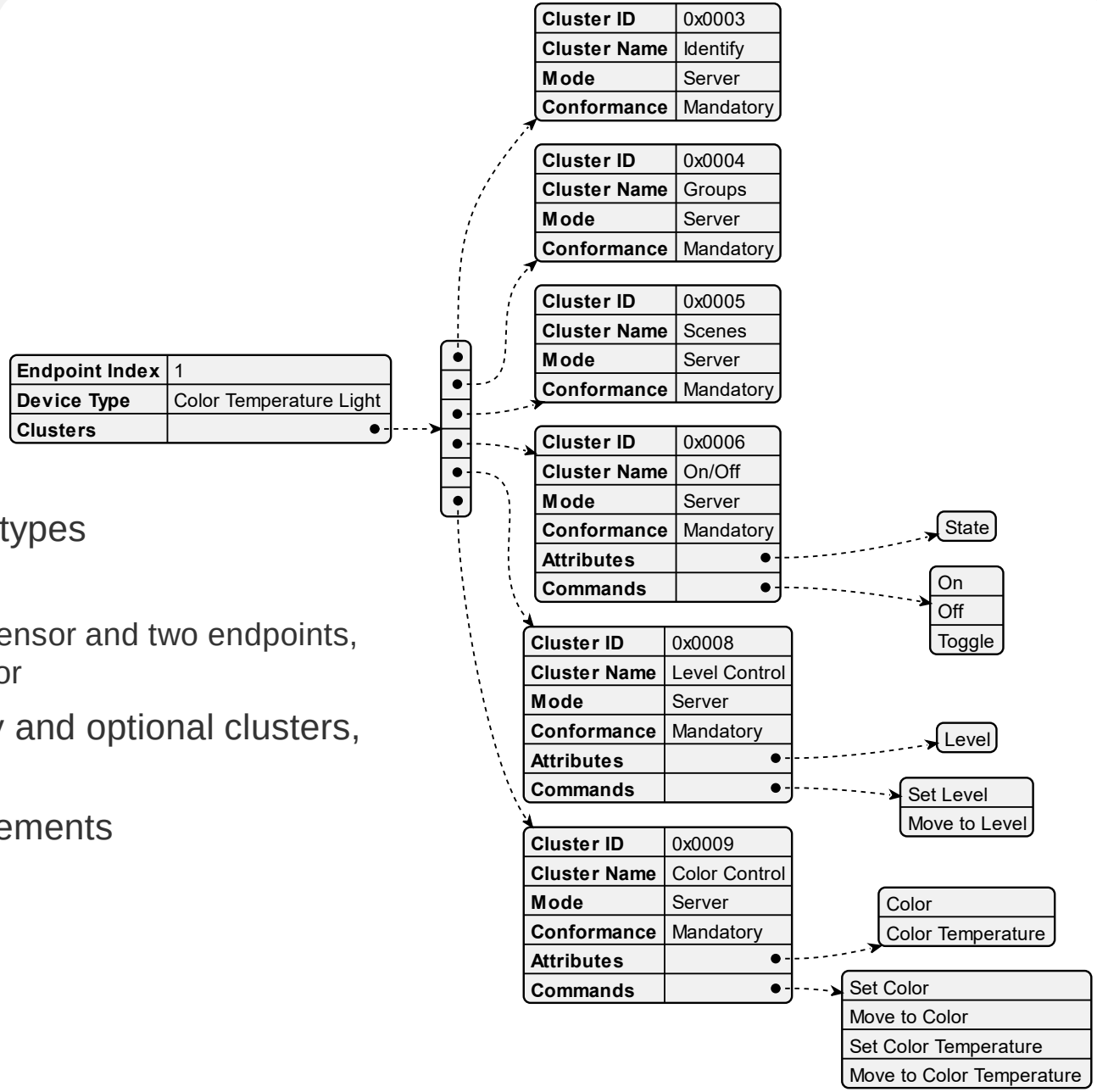
- Matter leverages the Zigbee Cluster Library (ZCL)
  - Provides a set of field-proven methods to form and control devices
  - Does not perfectly match the ZCL, has been extended with new functionality as needed
- An endpoint is generally a single device type
  - Interactions happen between local endpoints 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)



SILICON LABS

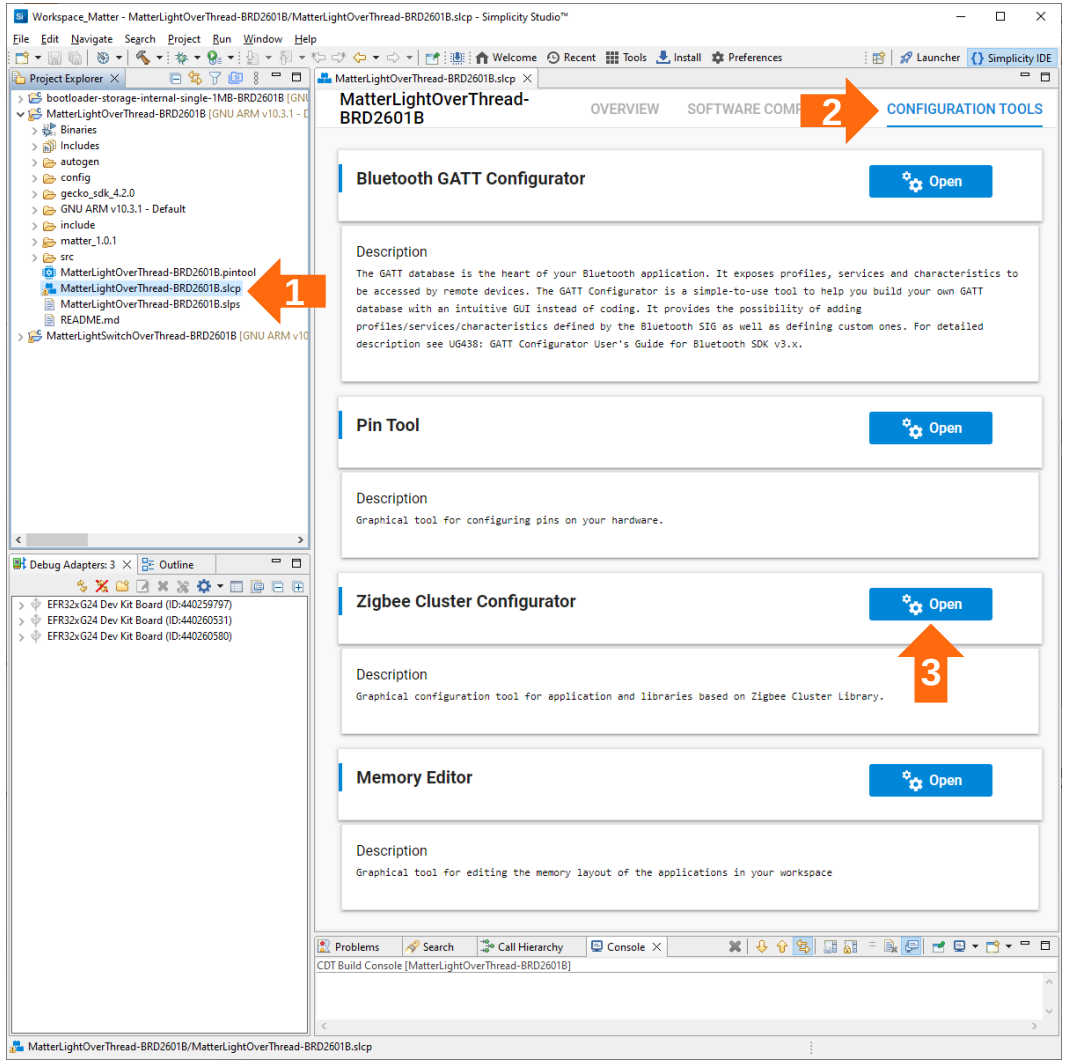
Device Types

- 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



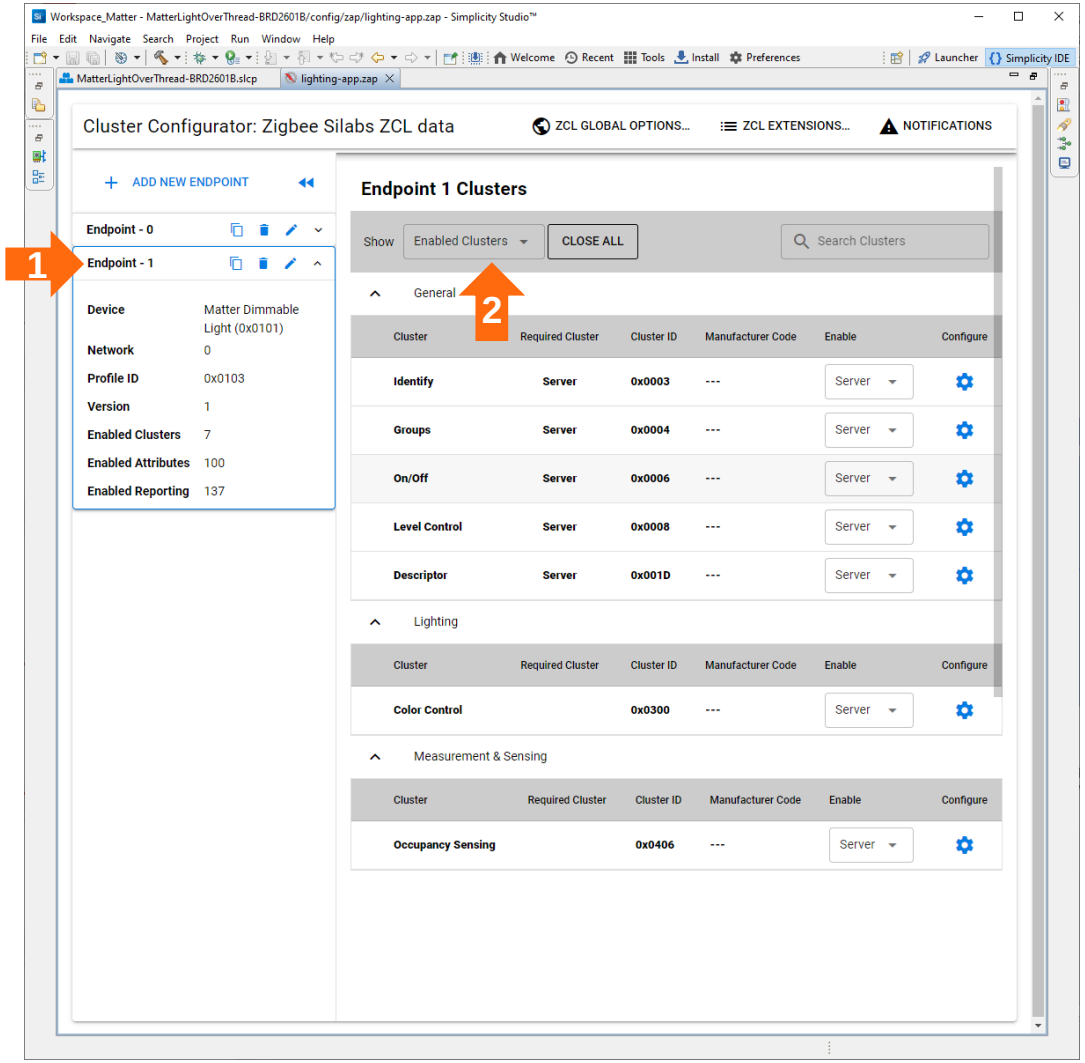
Zigbee Cluster Configurator – Open

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



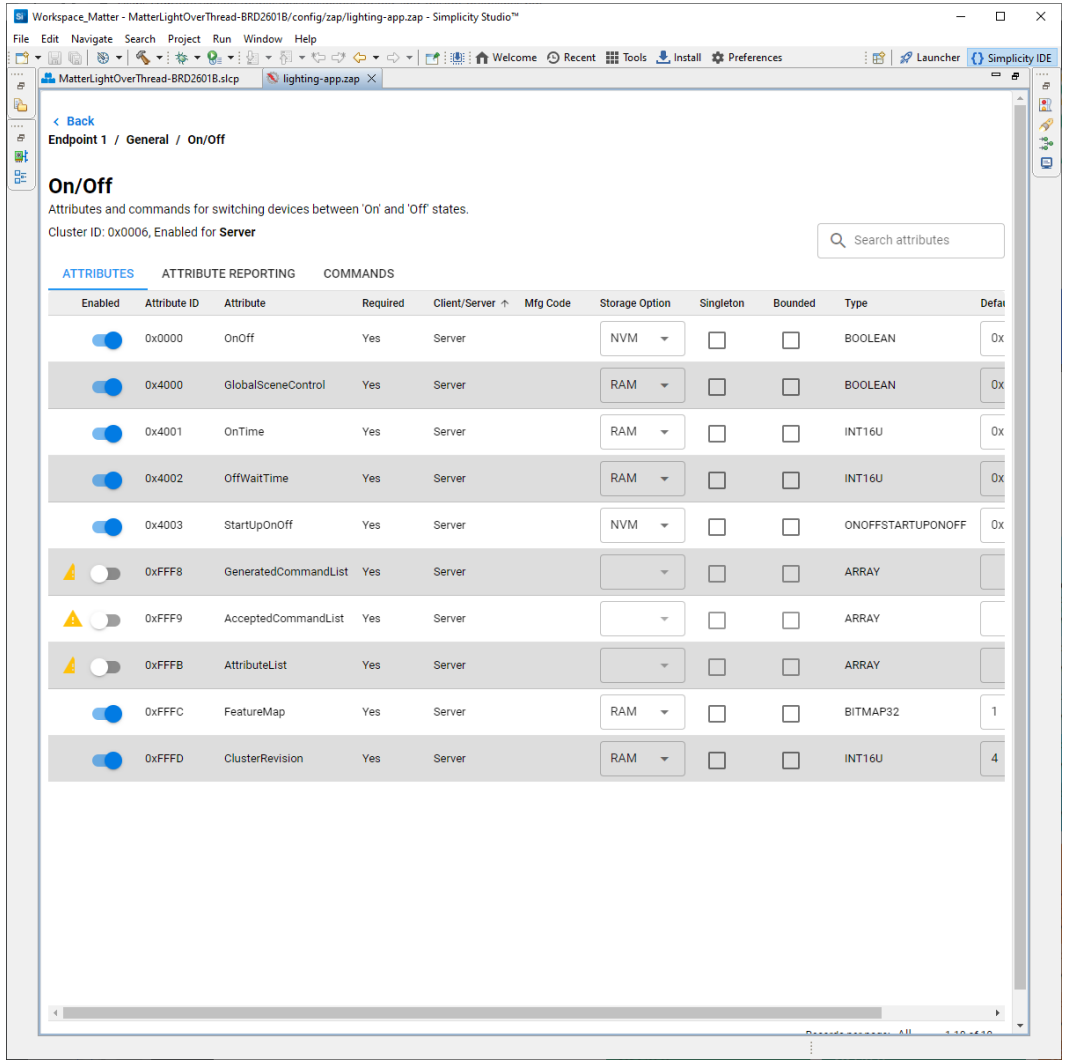
Zigbee Cluster Configurator – View Enabled Clusters

- 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:
  1. Click the **Endpoint – 1** box
  2. 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



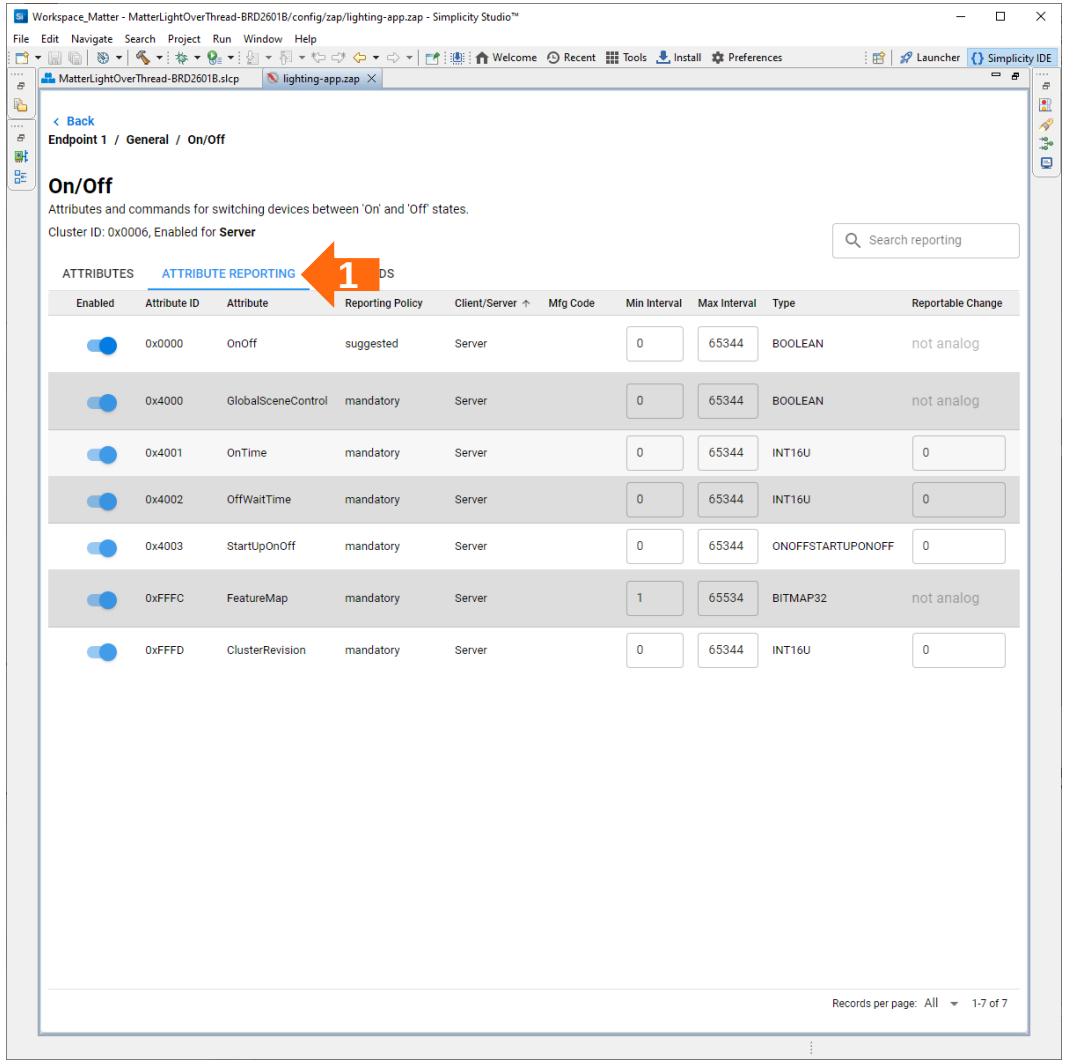
Zigbee Cluster Configurator – View On/Off Cluster Attributes

- To view the attribute settings in the On/Off cluster:
  1. 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



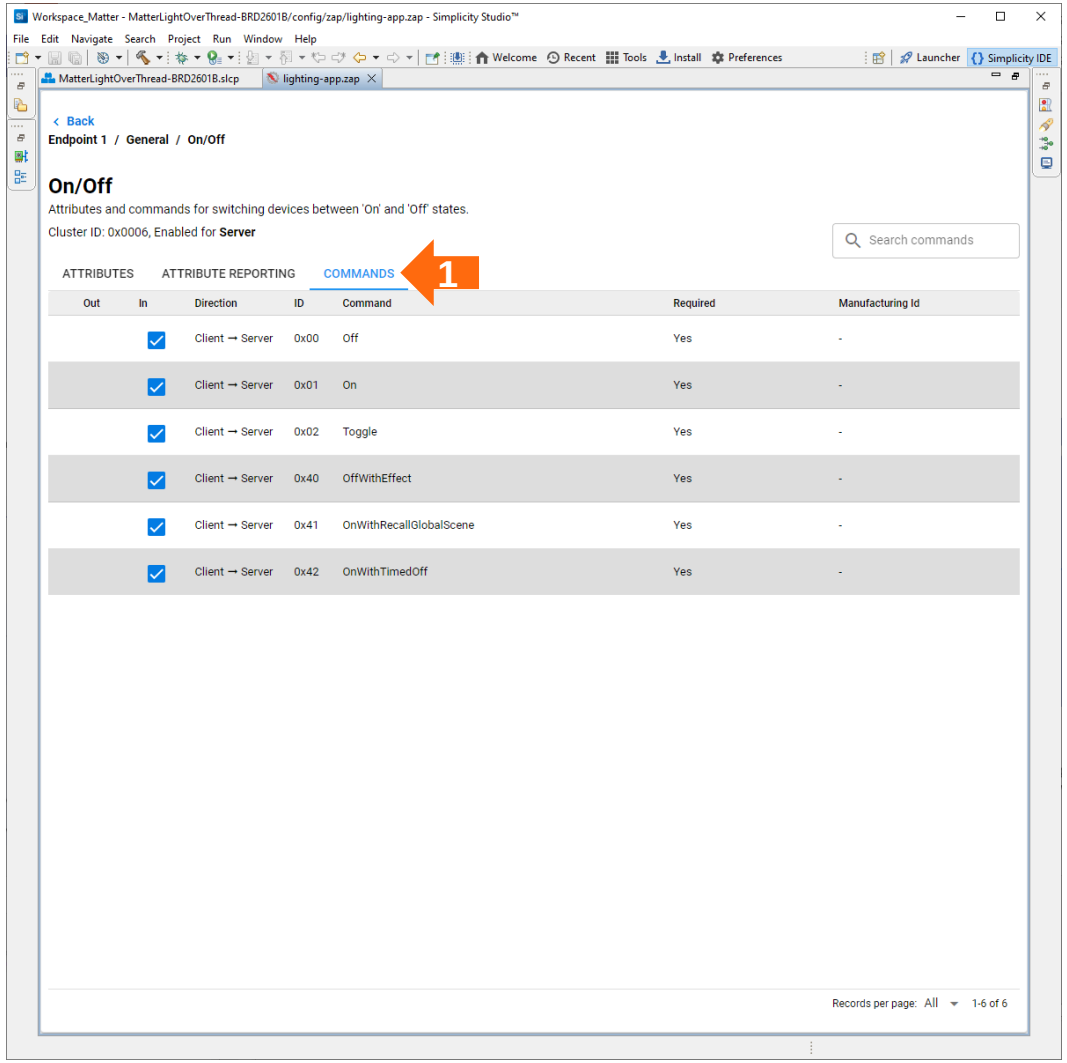
Zigbee Cluster Configurator – View On/Off Cluster Attribute Reporting

- To view the Attribute Reporting settings in the On/Off cluster:
  1. 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



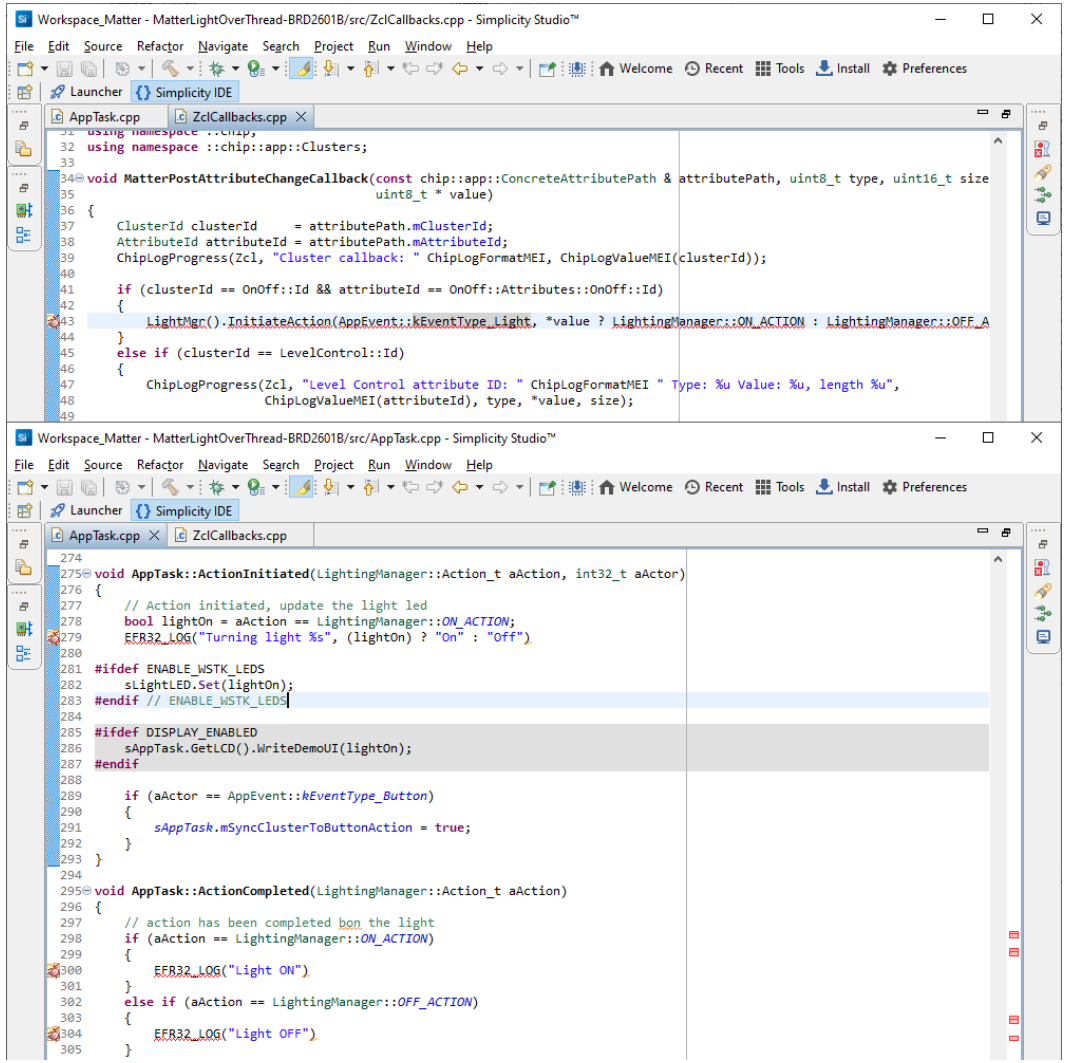
Zigbee Cluster Configurator – View On/Off Cluster Commands

- 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



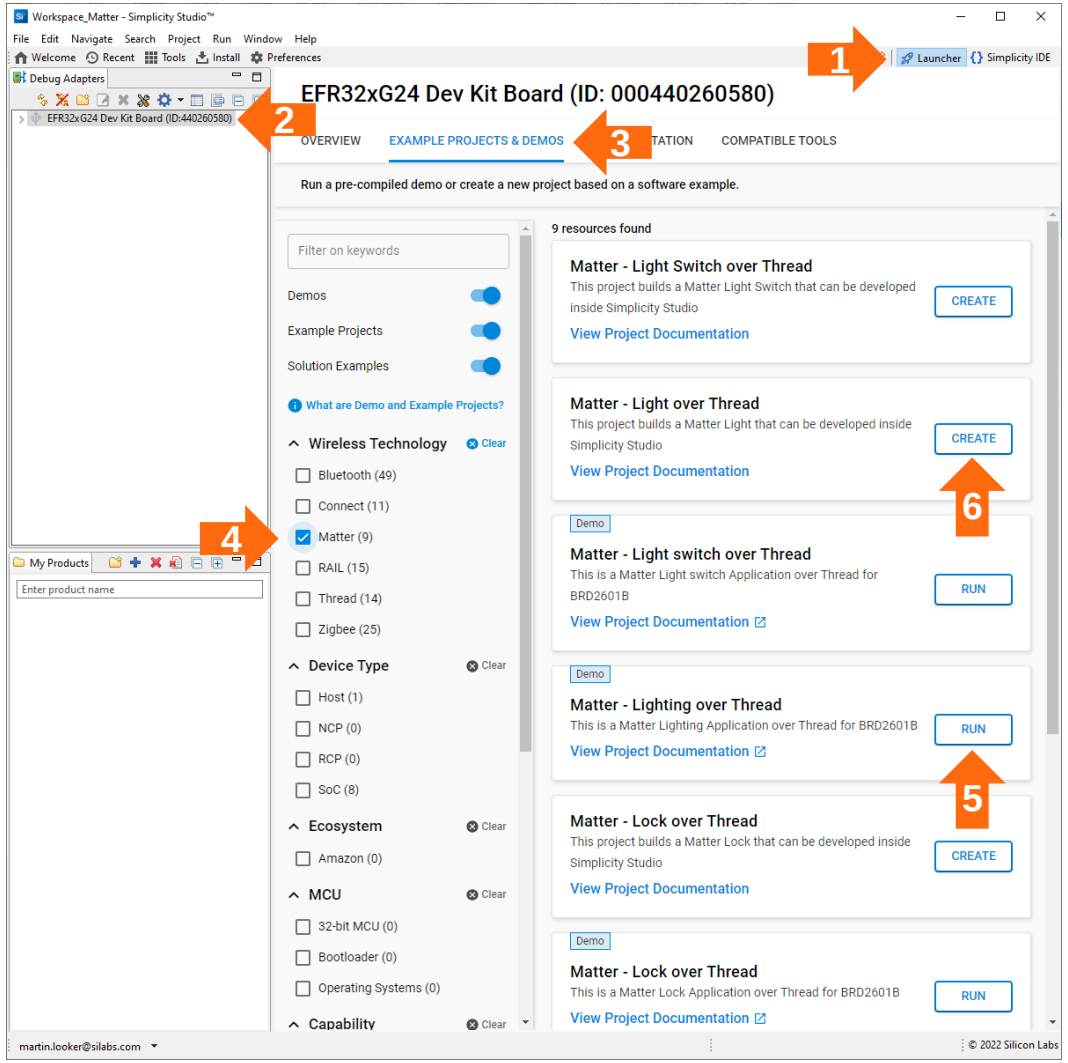
Code – On/Off Cluster Commands

- 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 intitiates 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



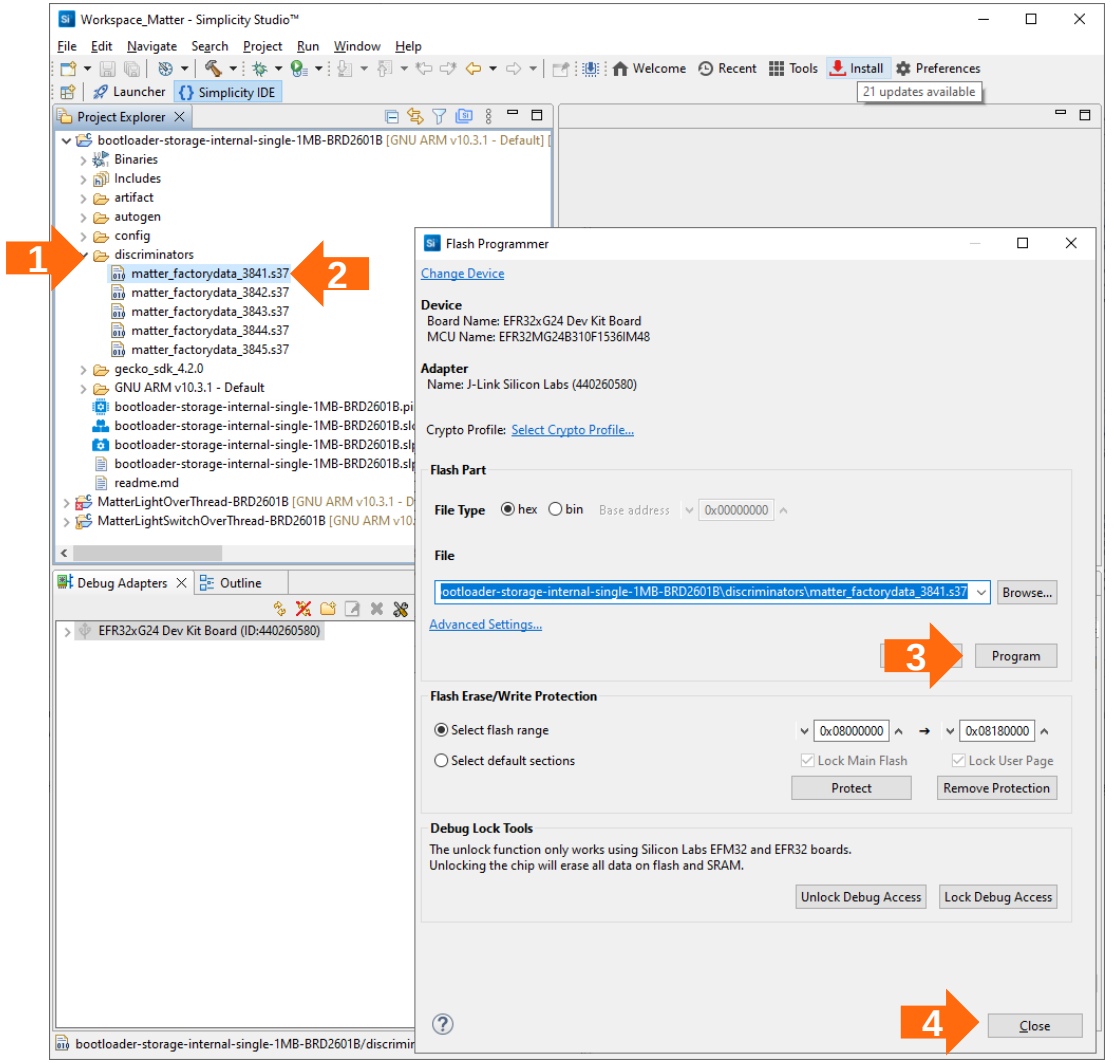
Switch Demo / Example

- 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



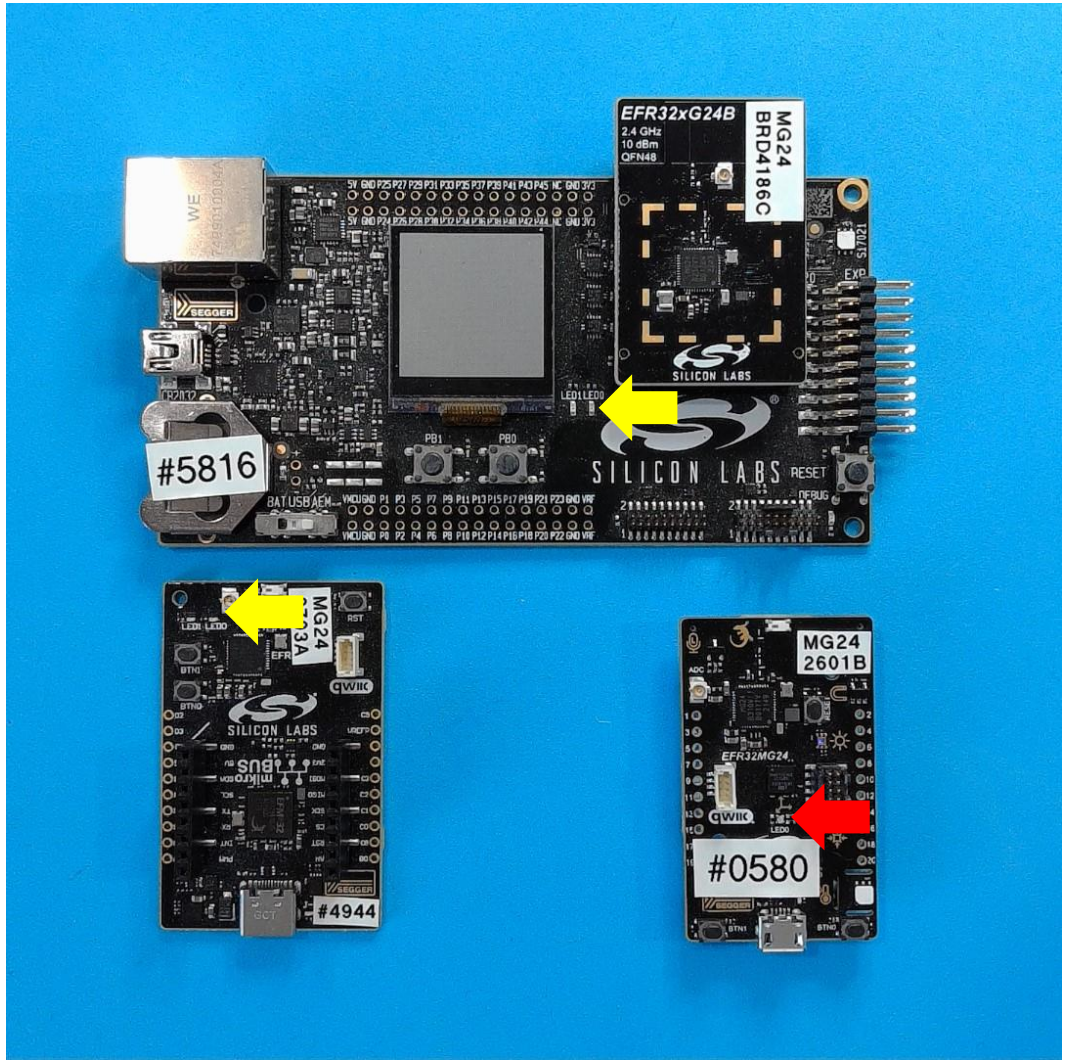
Switch – Change Discriminator

- The discriminator and associated certificates are stored in NVM in the light device
- To change the discriminator in the switch device:
  1. In **Simplicity Studio**, expand the `discriminators` folder (copied in earlier) in the **Project Explorer** panel
  2. Right-click the `.s37` file for your group's discriminator and select **Flash to device…**
  3. In the **Flash Programmer**, check the **File** is correct and click the **Program** button
  4. 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):
  1. `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:

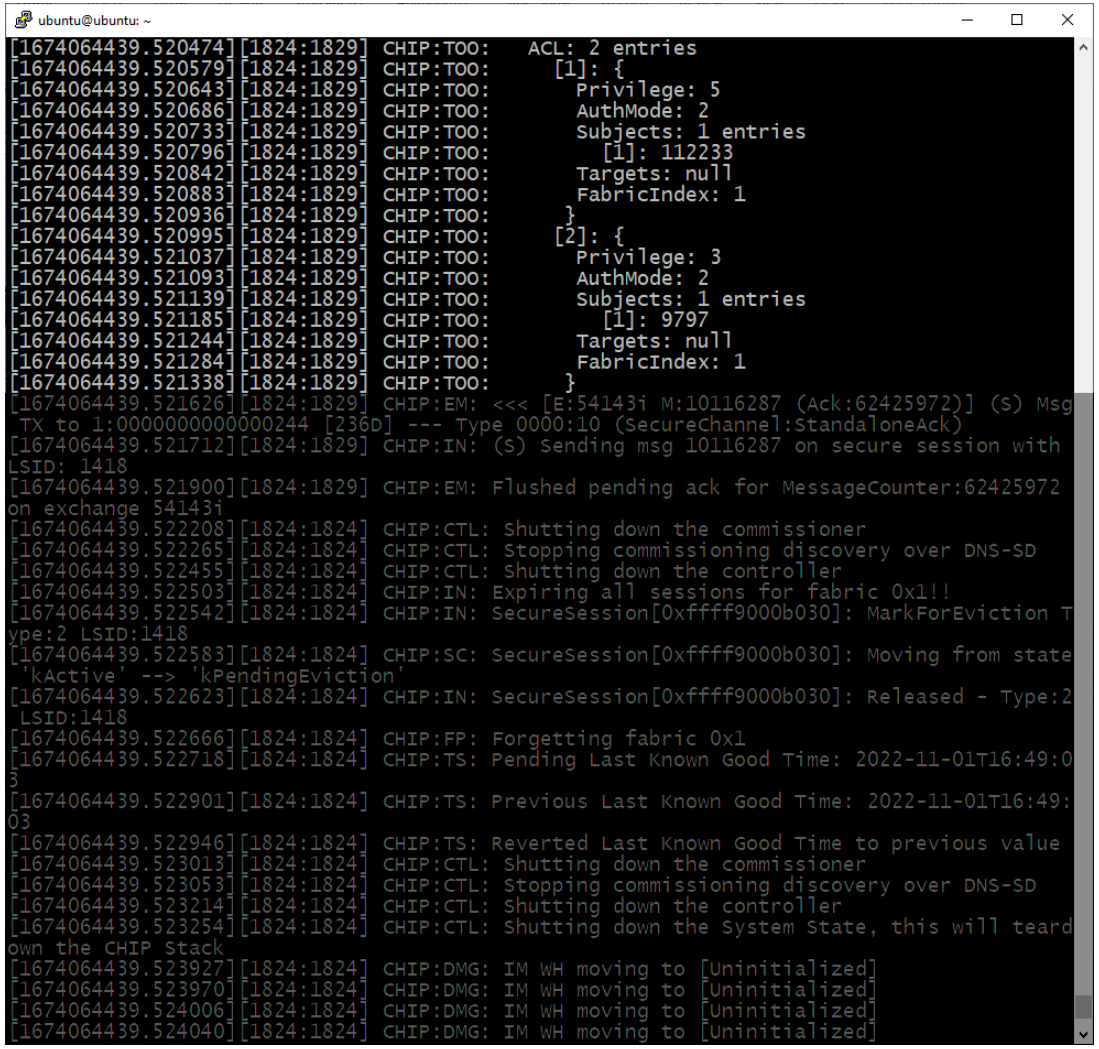
1. `mattertool accesscontrol write acl '[ { "fabricIndex" : 1 , "privilege" : 5 , "authMode" : 2 , "subjects" : [ 112233 ] , "targets" : null } , { "fabricIndex" : 1 , "privilege" : 3 , "authMode" : 2 , "subjects" : [ 200 ] , "targets" : null } ]' 100 0`

   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



SILICON LABS

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:

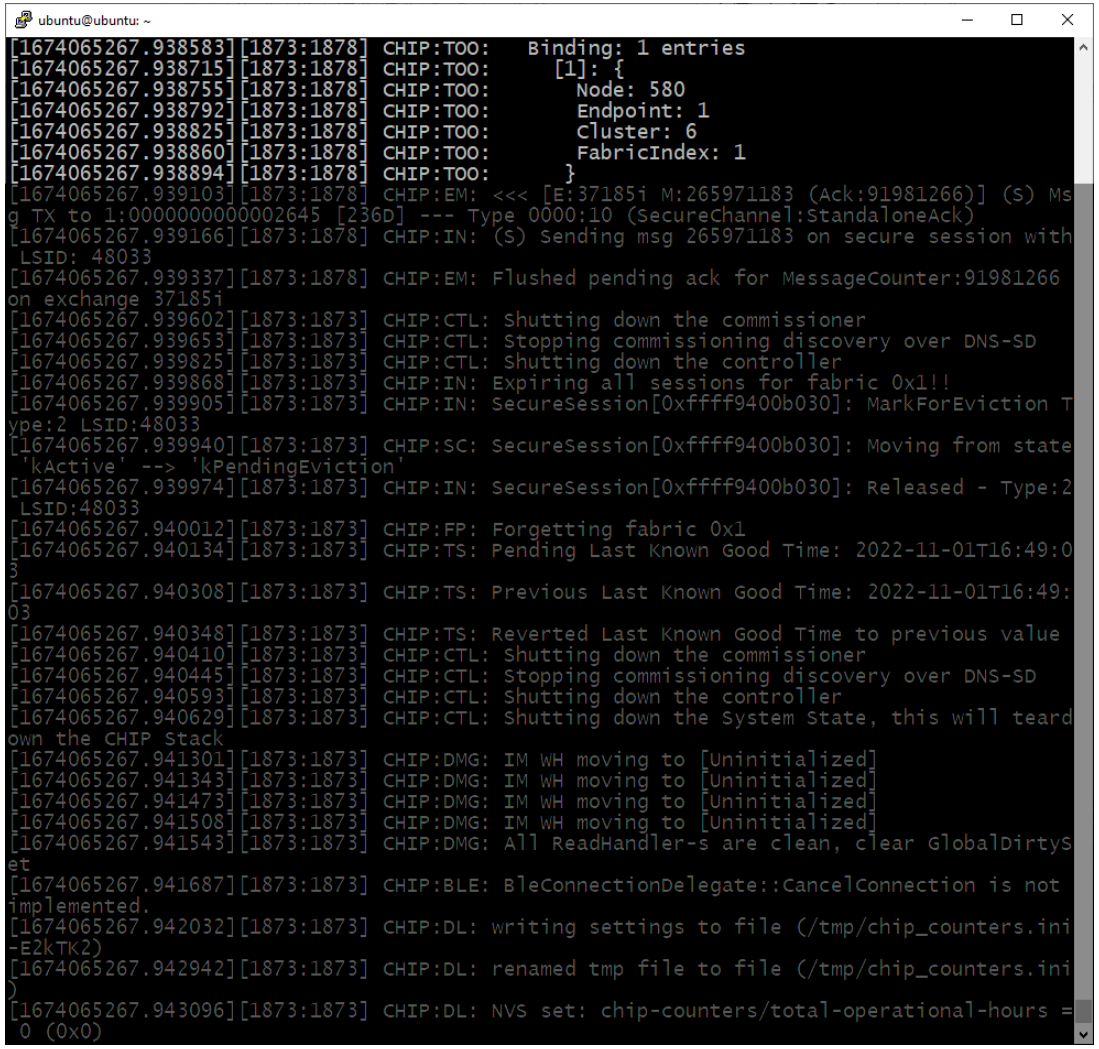
1. `mattertool binding write binding '[ { "fabricIndex" : 1 , "node" : 100 , "endpoint" : 1 , "cluster" : 6 } ]' 200 1`

   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

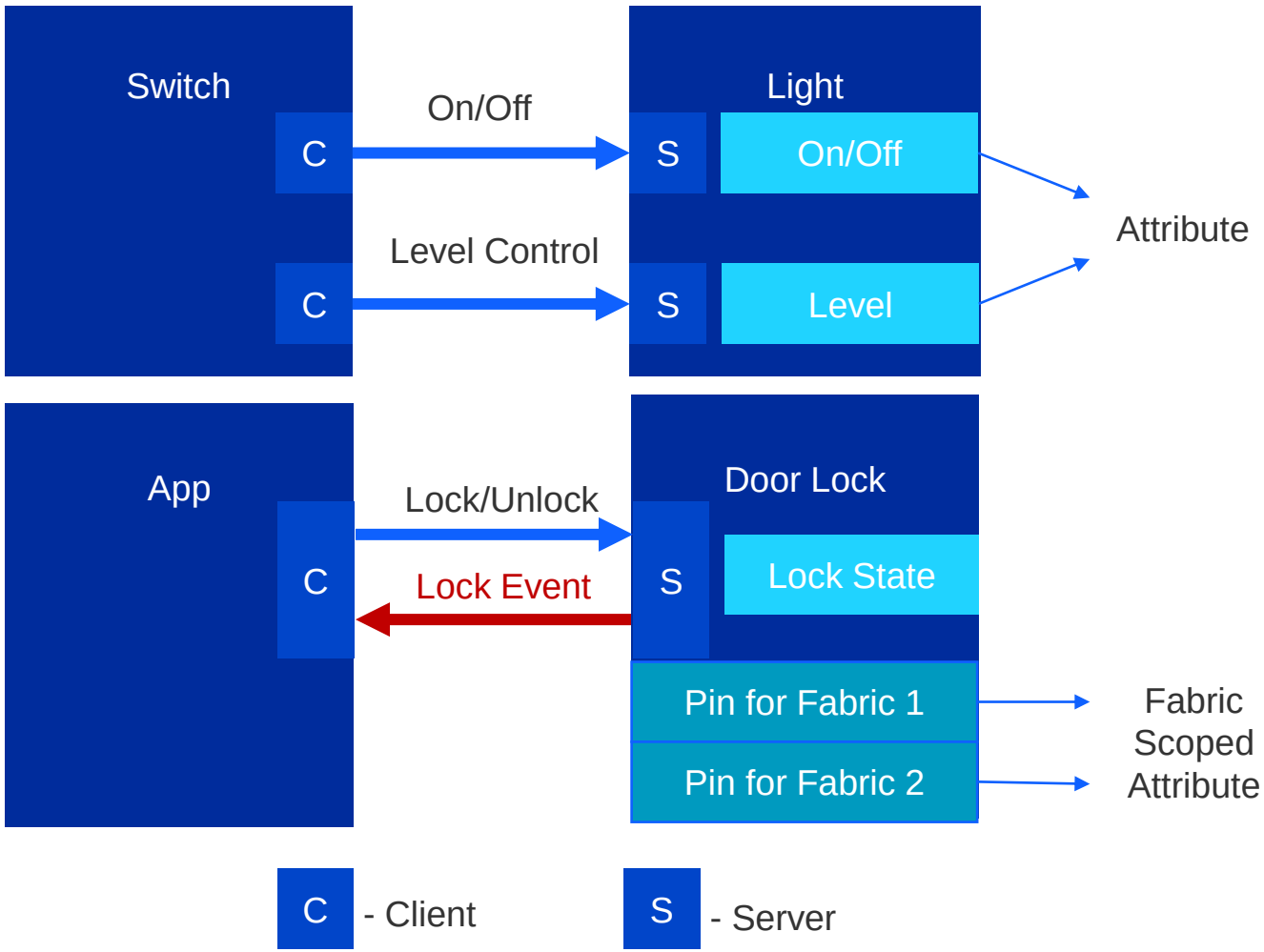


Binding Table – Write Multiple Command

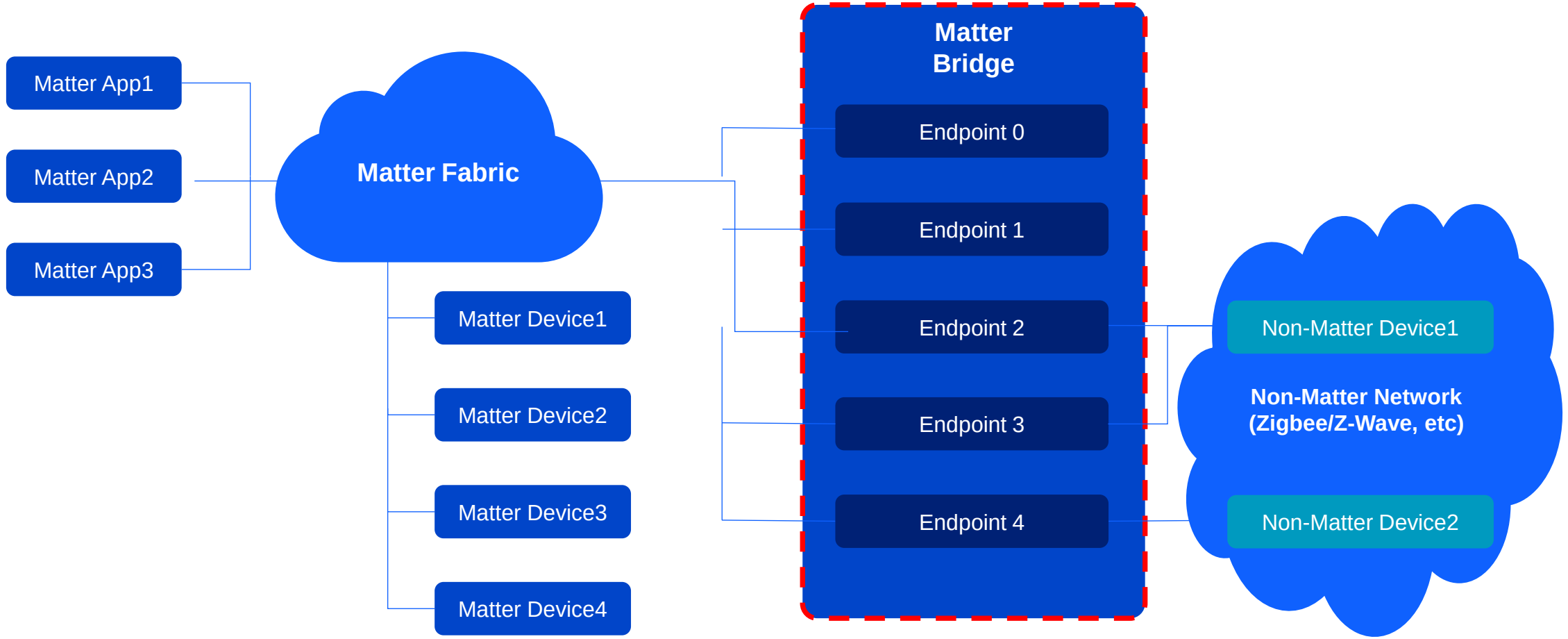
- To write multiple entries to the binding table in a switch:
  - `mattertool binding write binding '[ { "fabricIndex" : 1 , "node" : 100 , "endpoint" : 1 , "cluster" : 6 } , { "fabricIndex" : 1 , "node" : 101 , "endpoint" : 1 , "cluster" : 6 } ]' 200 1`
  - Where:
  - **100** and **101** are the nodeids 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

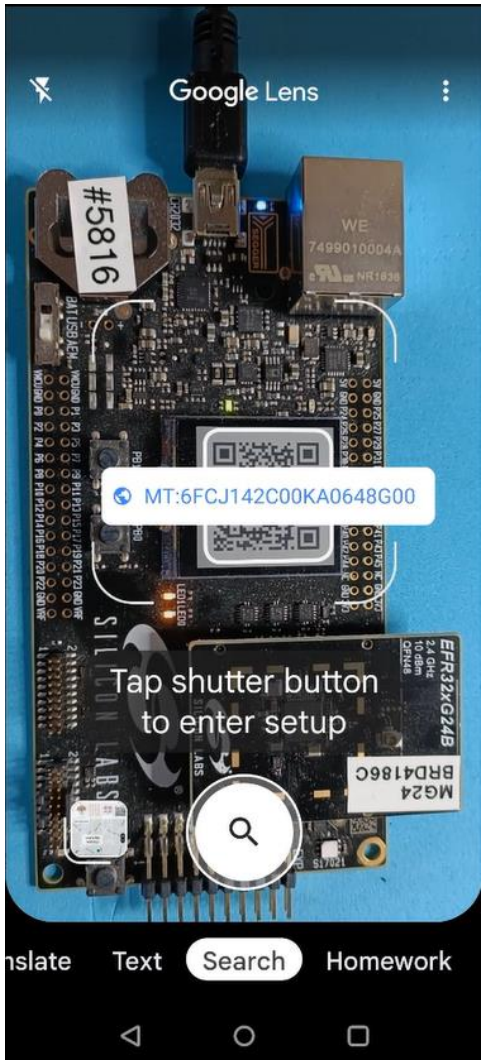


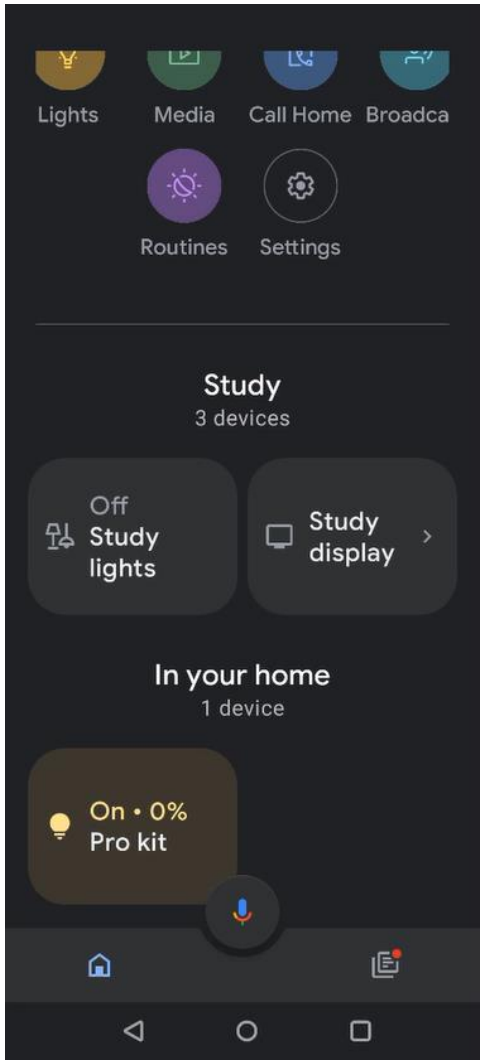
Bridge for Non-Matter Devices

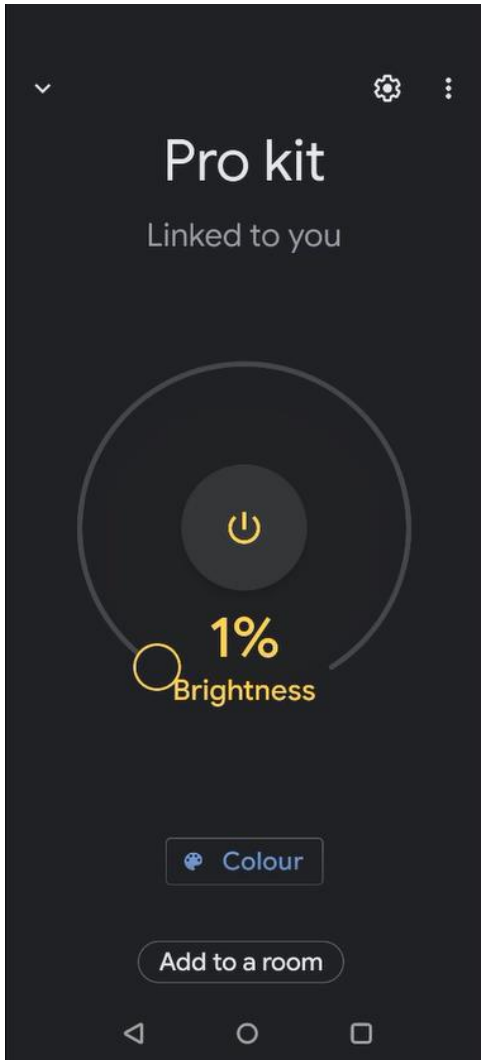


Google Home

- 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







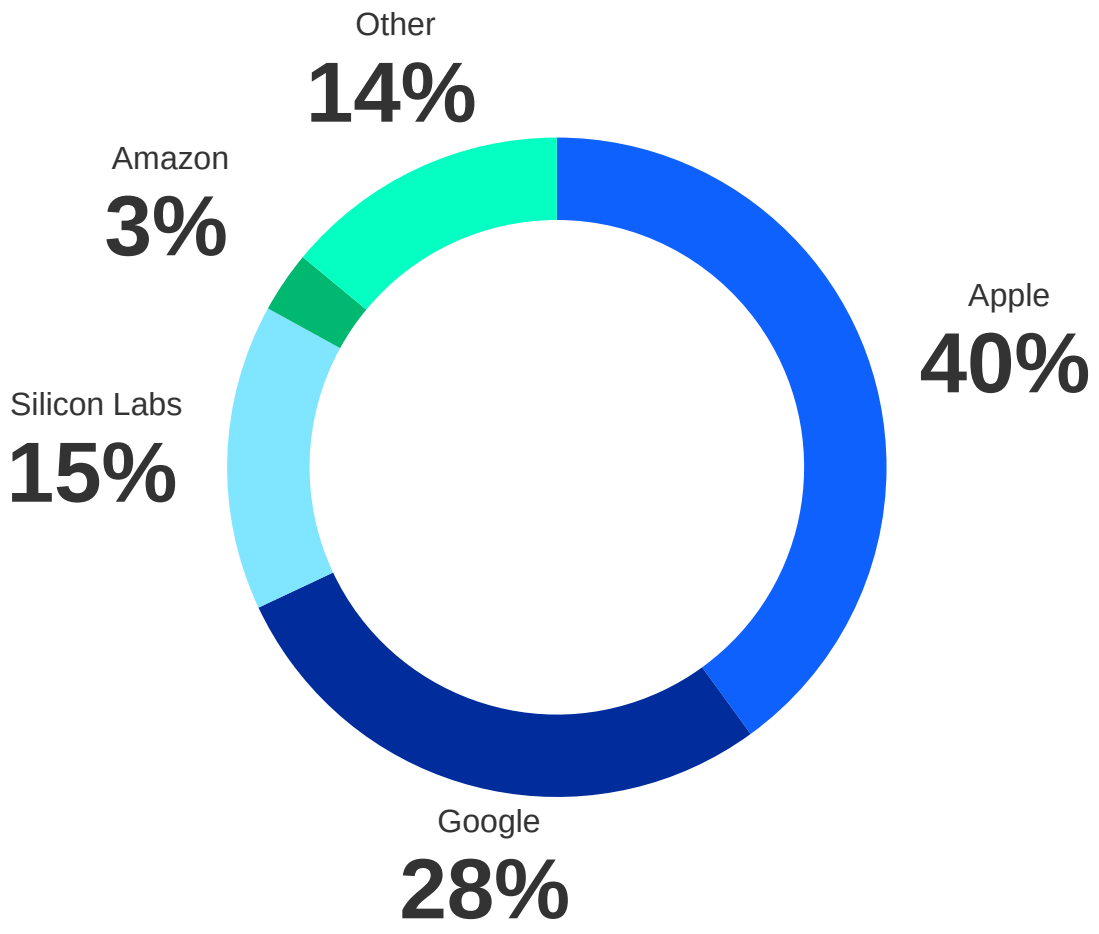
Matter GitHub and GSDK Offerings – Our Plan

	Item	csa matter GitHub For cutting edge Matter Protocol Developers	SILICON LABS matter GitHub Stable Silicon Labs Matter Source on GitHub	Si5 Studio & GSDK Matter Silicon Labs Simplicity Studio and GSDK support (Rolling out now)
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

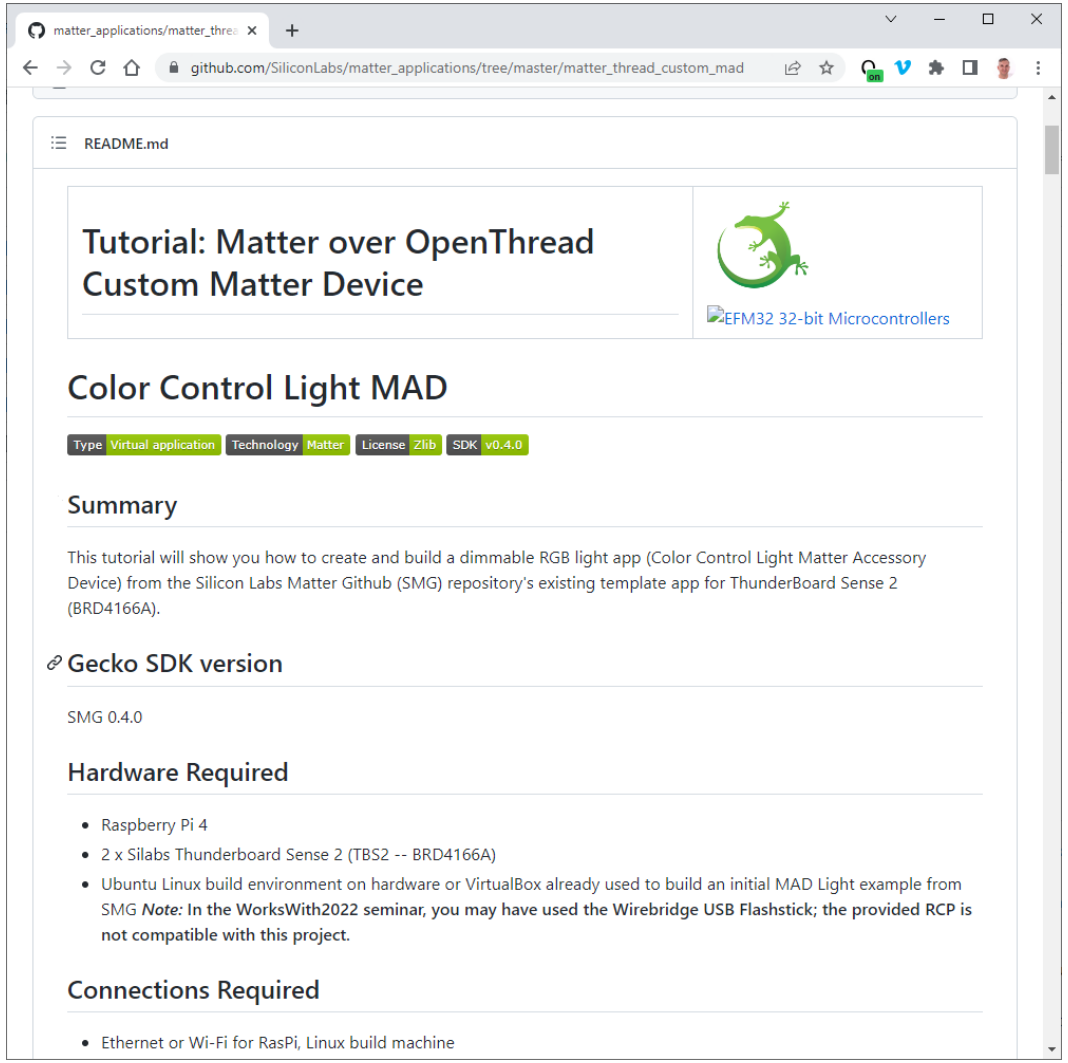


Silicon Labs is the 3rd largest contributor to Matter GitHub

- **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



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