

SE-103

Getting Started with the Arduino Nano Matter



Martin Looker

Applications Training Manager

Agenda

- 01** How Silicon Labs, Arduino and Matter make IoT device development available to all
- 02** Configuring the Arduino IDE to work with Silicon Labs powered boards
- 03** Example applications available in the IDE
- 04** The structure and APIs used to create Arduino Matter applications
- 05** Enhancing, compiling, flashing, commissioning and using your first application
- 06** Next steps, Q&A

The Leader in IoT Wireless Connectivity



100%
IoT Focused

amazon sidewalk

Bluetooth

matter

Multiprotocol

Proprietary

THREAD

WiFi

WiSUN

zigbee

ZWAVE

**Breadth and
Depth of
Wireless IoT
Protocols**



#1
Share in Mesh



1st
To Market with Multiprotocol,
BLE Mesh, BLE 5.1



Innovation
Performance, Power,
CoEx, Modules,
SecureVault™

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

ZWAVE

2018

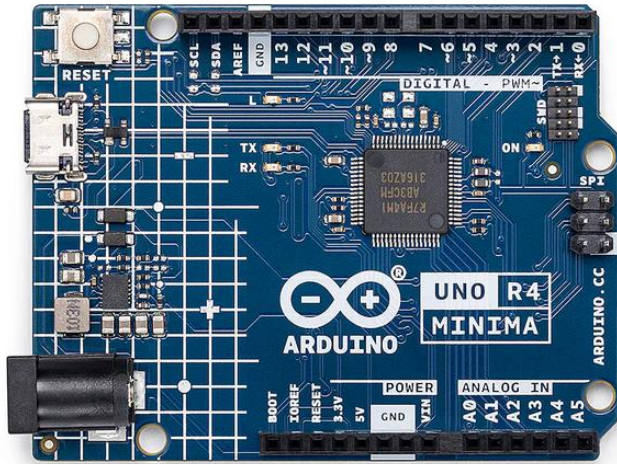
Smart Home
Protocol

**REDPINE
SIGNALS**

2020

Ultra Low
Power Wi-Fi

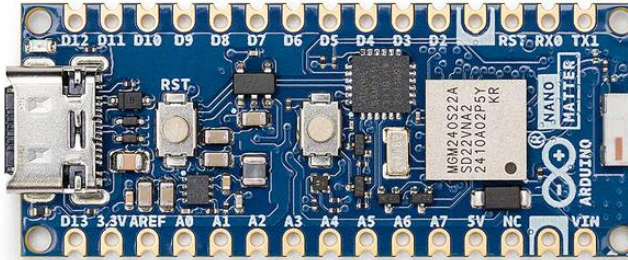
About Arduino



WHAT IS ARDUINO?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It's intended for anyone making interactive projects.

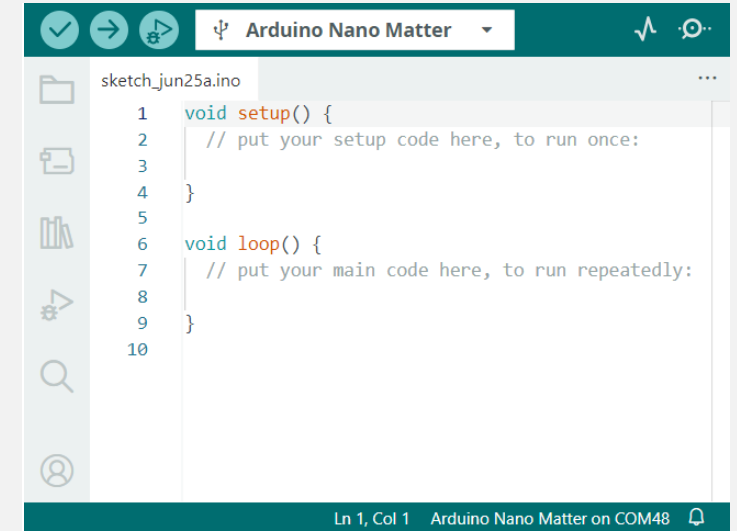
[Learn more about Arduino](#)



ARDUINO HARDWARE

Arduino senses the environment by receiving inputs from many sensors and affects its surroundings by controlling lights, motors, and other actuators.

[Discover the official Arduino boards](#)



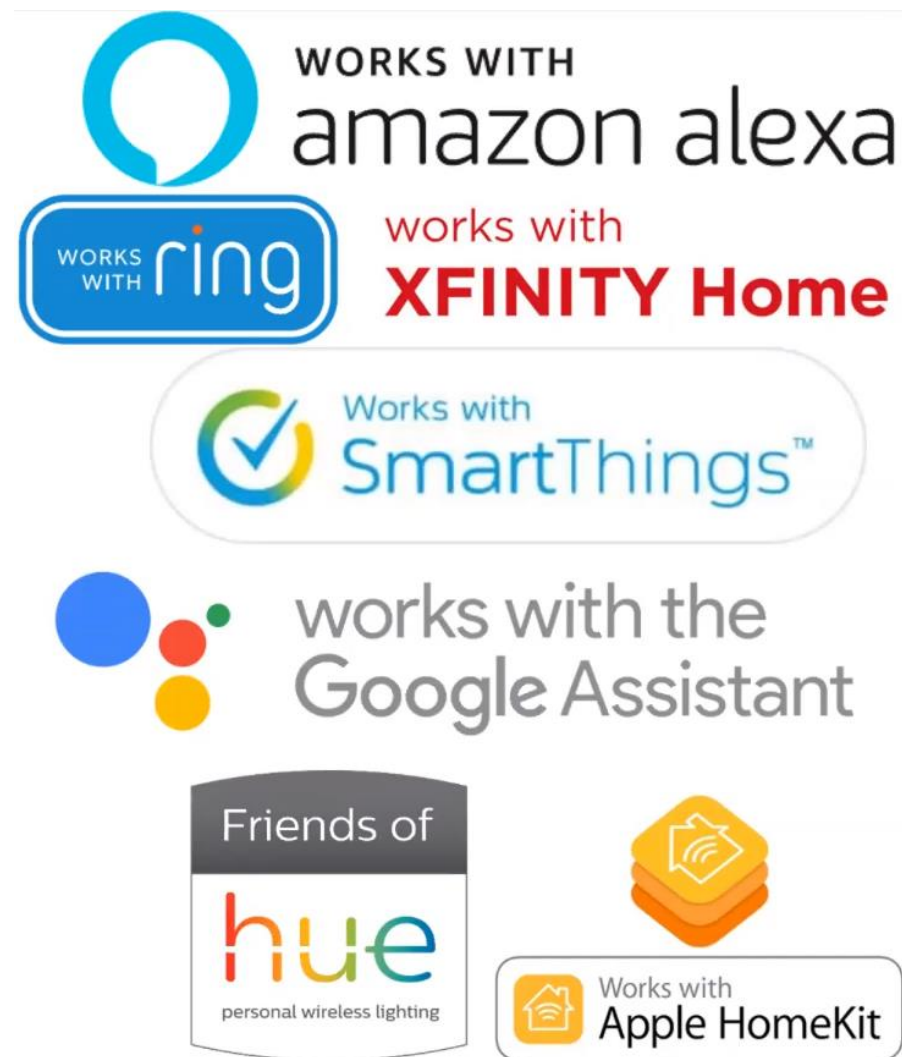
ARDUINO SOFTWARE

You can tell your Arduino what to do by writing code in the Arduino programming language and using the Arduino development environment.

[Download the Arduino software](#)

Historical Landscape of the 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

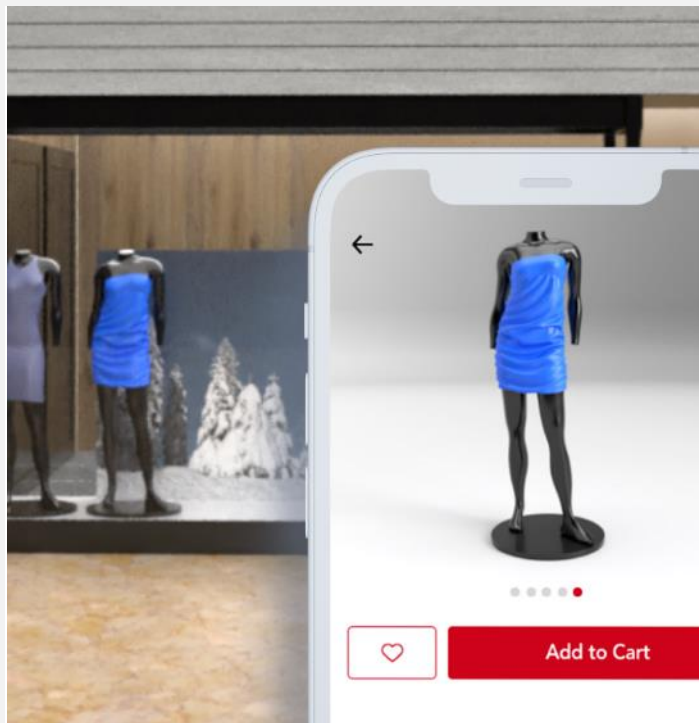


Simplifying IoT Connectivity with Matter



MANUFACTURER

Single SKU
Lower development
& operational cost
More time for innovation



RETAILER

Requires less shelf space
Lowers inventory cost
Minimizes returns

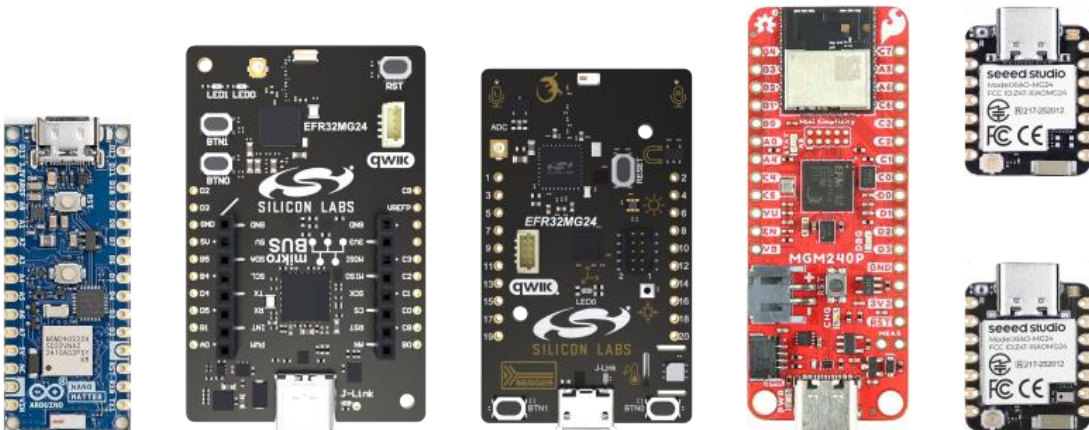


CONSUMER

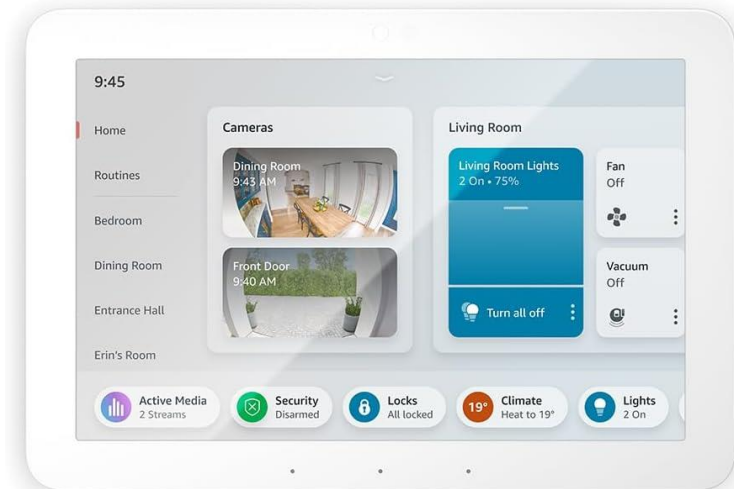
Simplifies purchasing experience
Simplifies setup & control
Provides better user experience

Hardware

- There are six boards that allow the development of Matter over Thread applications in the Arduino IDE:
 - [Arduino Nano Matter](#)
 - [Silicon Labs xG24 Explorer Kit](#)
 - [Silicon Labs xG24 Dev Kit](#)
 - [SparkFun Thing Plus Matter MGM240P](#)
 - [Seeed Studio XIAO MG24](#)
 - [Seeed Studio XIAO MG24 Sense](#)

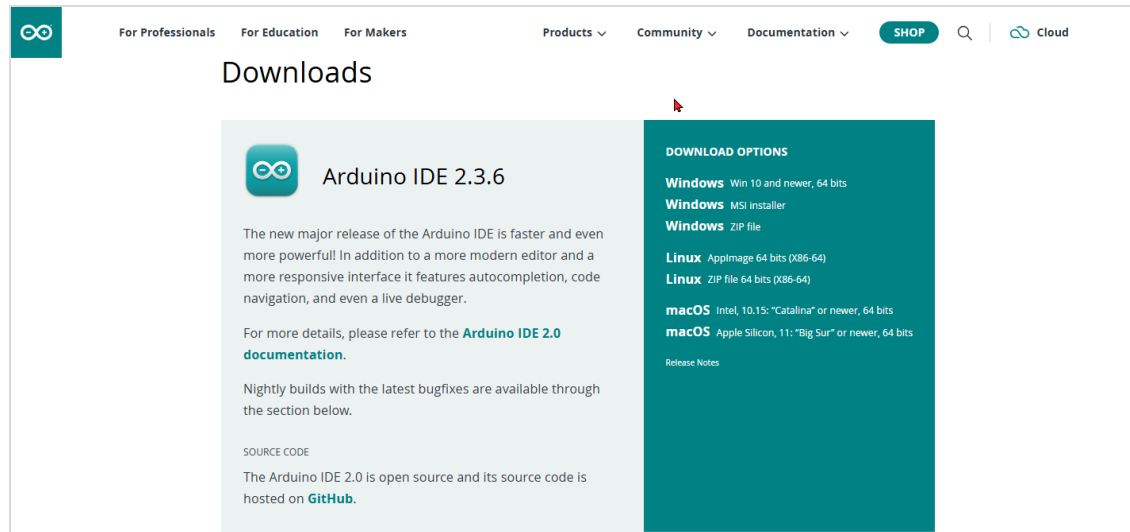


- Hubs that include an OpenThread Border Router (required for Matter over Thread applications) are available from a wide range of ecosystem manufacturers including:
 - [Amazon](#)
 - [Google](#)
 - [Apple](#)
 - Home Assistant (open source)
 - [OpenThread Border Router](#)
 - [Matter over Thread](#)

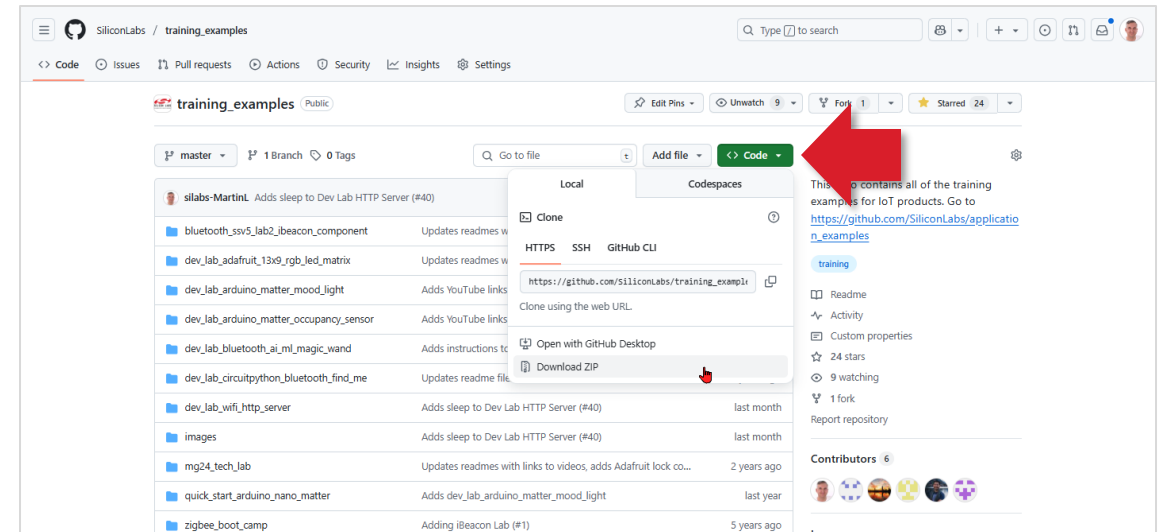


Software

- To develop Matter over Thread applications you will need the Arduino IDE:
 - [Downloads for Windows, Linux and macOS](#)
 - [Installation instructions](#)



- We will be using enhanced code during this session which is available in the [Silicon Labs Training Examples repository on GitHub](#) to obtain the files using the **Code** dropdown either:
 - Clone the repository using your favourite Git client
 - Download a ZIP file of the repository contents

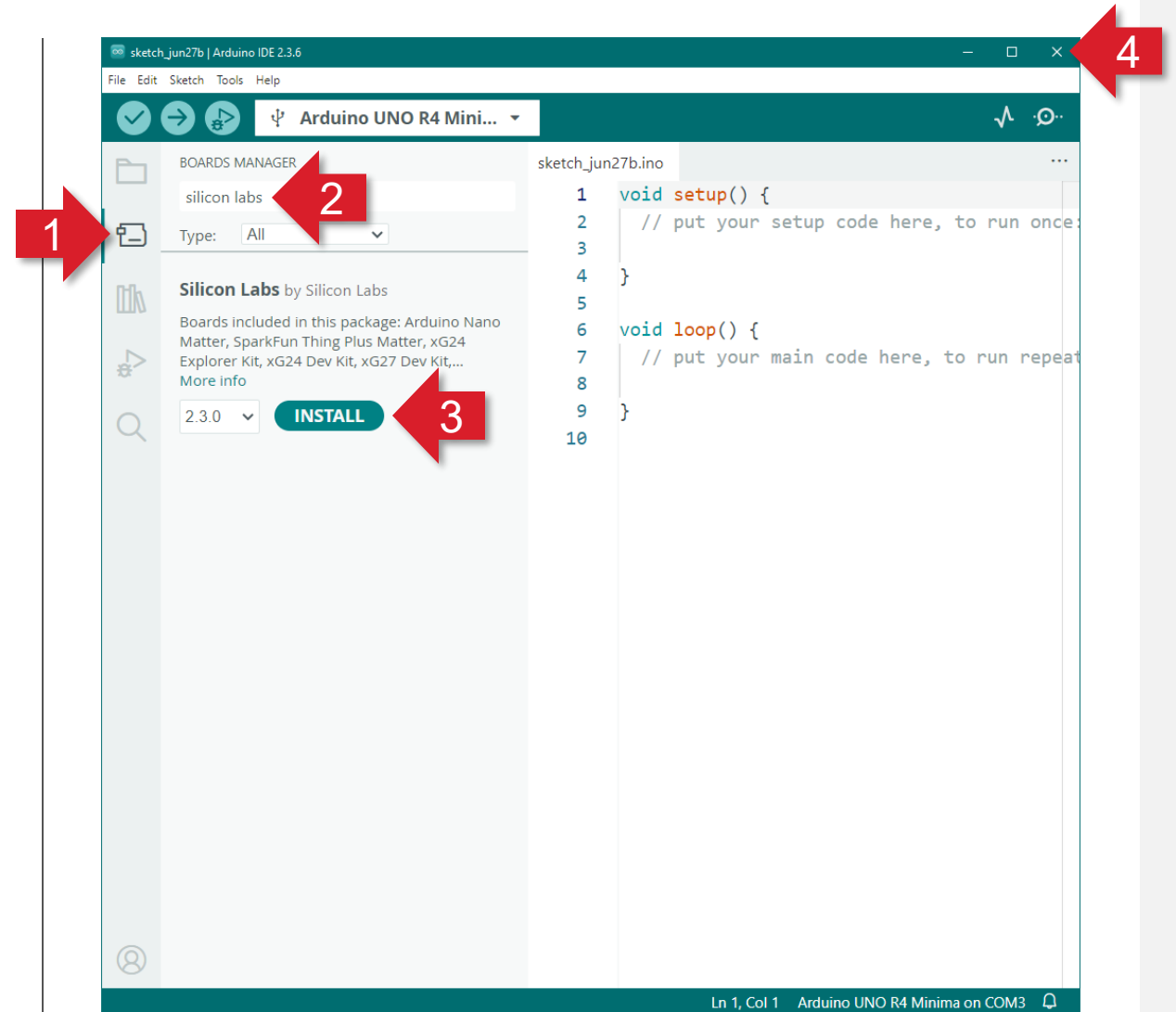


Arduino IDE and Board Setup



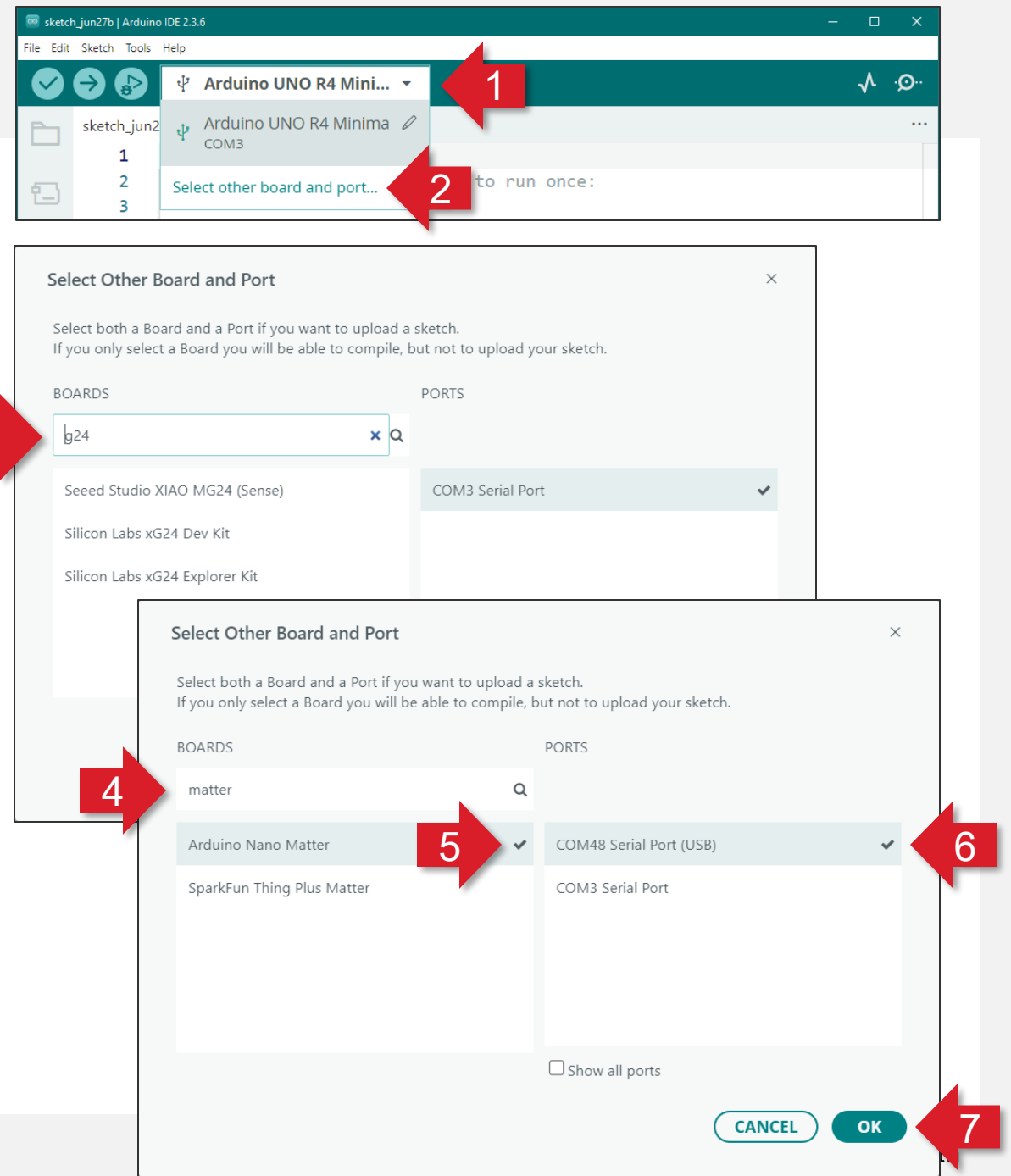
Boards Manager

- The [Arduino Nano Matter User Manual](#) has instructions for setting the Arduino IDE to work with the Arduino Nano Matter board
- The [Silicon Labs Arduino Core repository on GitHub](#) contains additional information, including for non-Arduino boards
- To install the Silicon Labs Arduino Core in the IDE:
 1. Open the **Boards Manager** by clicking the icon
 2. Enter **Silicon Labs** in the search box
 3. Click the **Install** button
 4. Restart the Arduino IDE



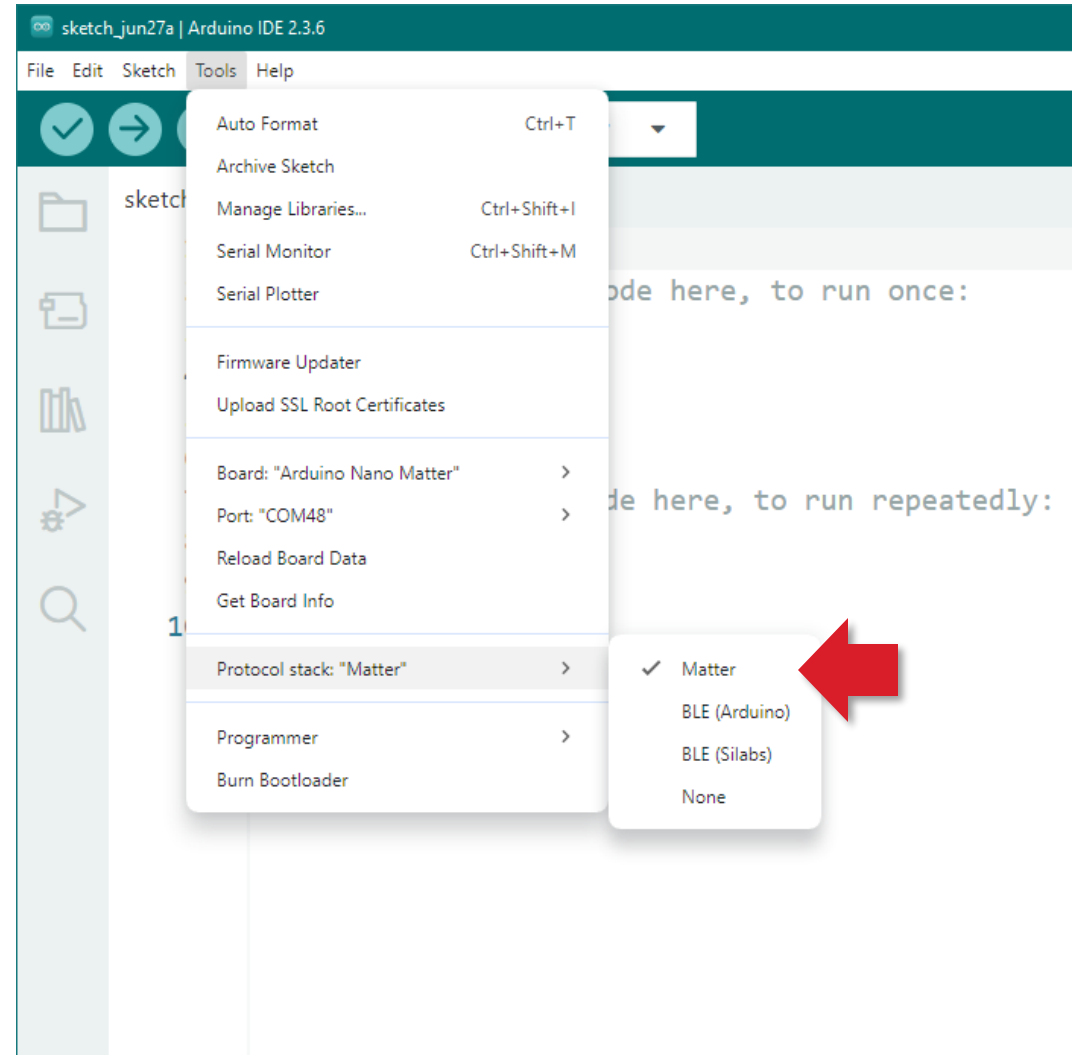
Board Configuration

- To configure a board for use in the IDE:
 1. Click the **Board** dropdown
 2. Click **Select other board an port...**
 3. Search for **g24** for Silicon Labs or Seeed Studio boards
 4. Search for **matter** for Arduino or SparkFun boards
 5. Select the correct board in the search results
 6. Connect board via USB and select the COM port that appears
 7. Click the **OK** button



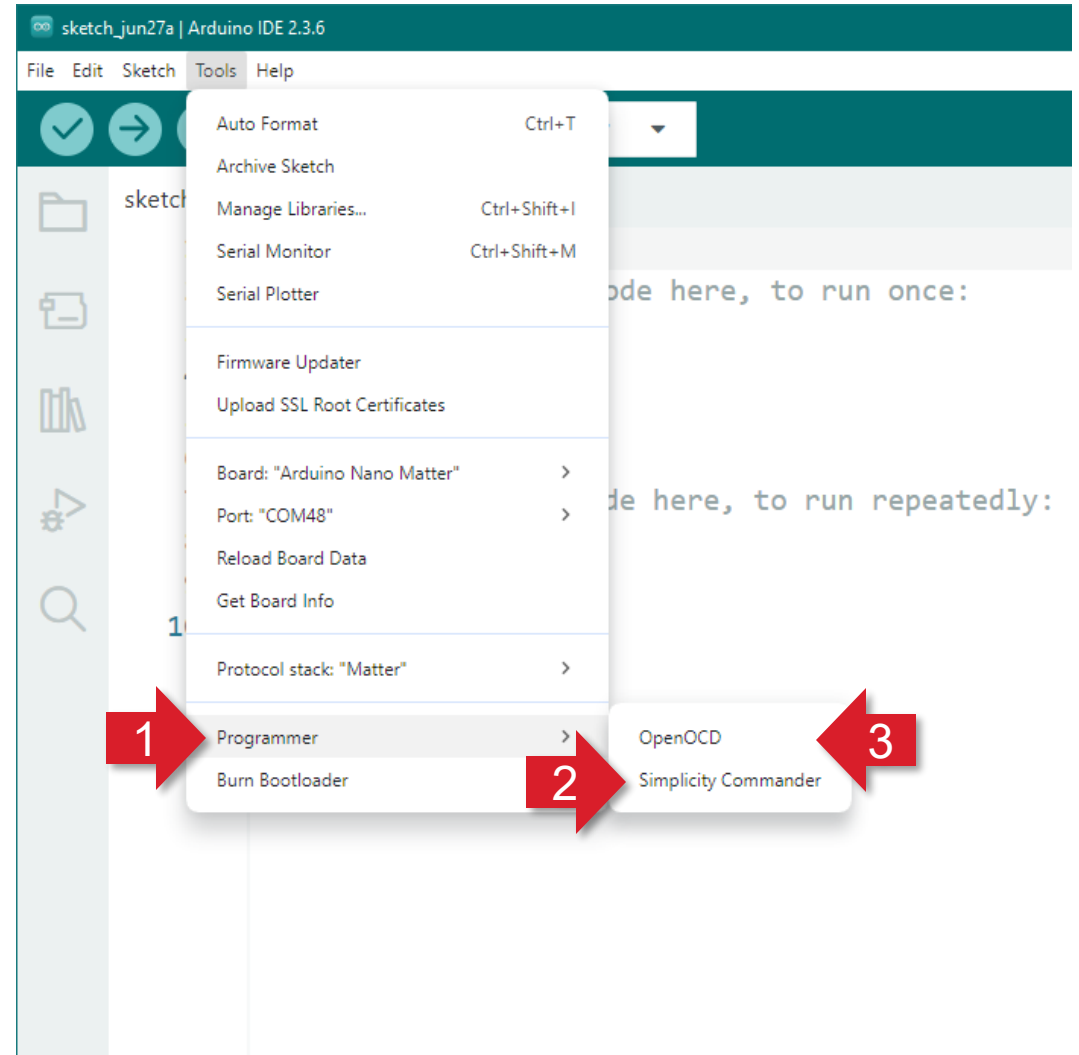
Protocol Stack

- Four different protocol stacks can be selected from the **Tools > Protocol stack** menu:
 - Matter** for developing Matter over Thread applications
 - BLE (Arduino)** for developing Bluetooth LE applications using the Arduino BLE APIs
 - BLE (Silabs)** for developing Bluetooth LE applications using the Silicon Labs APIs
 - None** for developing applications without radio comms



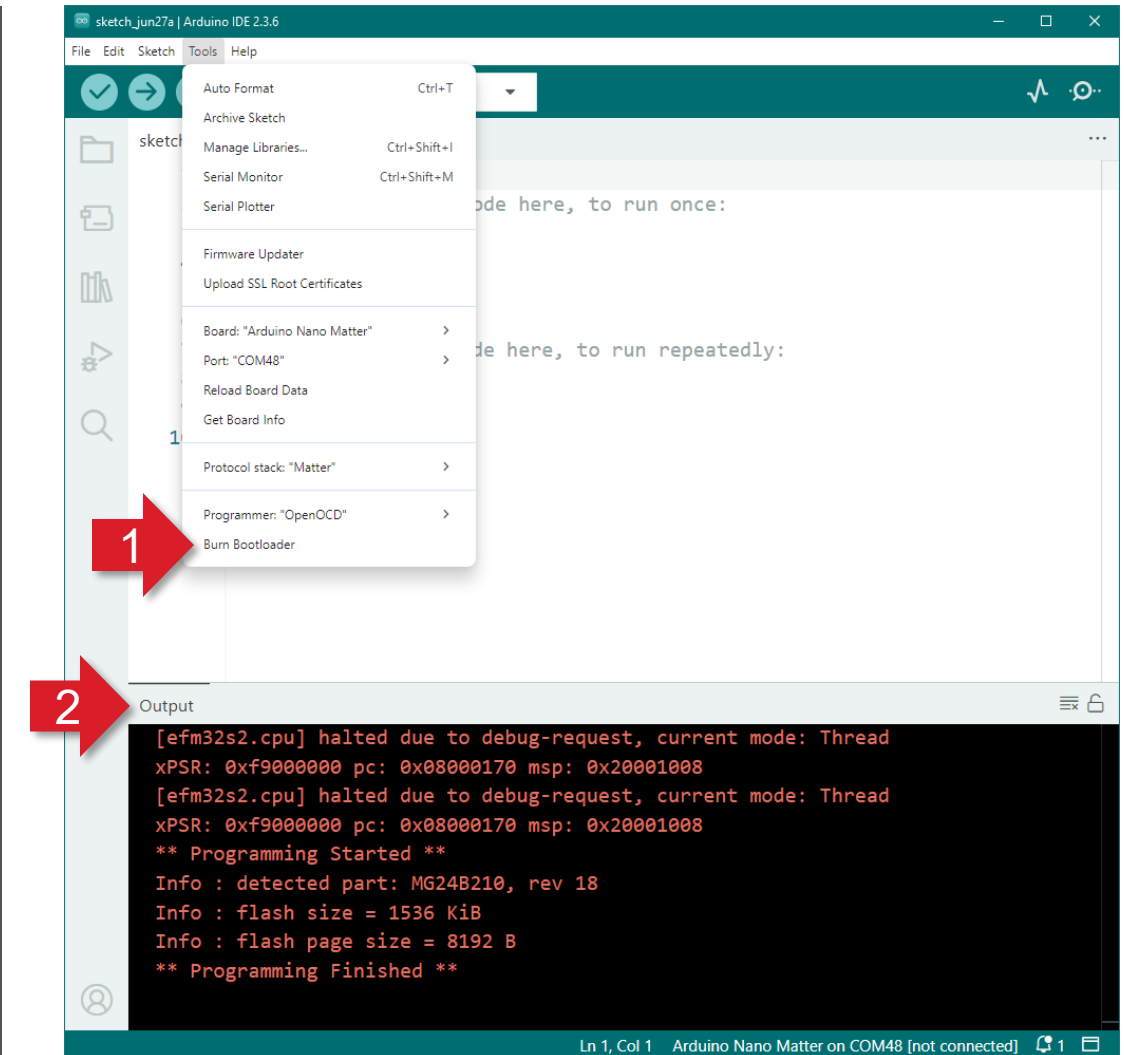
Programmer

- Different boards use different flash programmers:
 1. From the menu click **Tools > Programmer**
 2. For Silicon Labs and SparkFun boards, select **Simplicity Commander**
 3. For the Arduino and Seeed Studio boards, select **OpenOCD**



Bootloader

- All boards require a bootloader to be programmed:
 1. From the menu click **Tools > Burn Bootloader**
 2. Progress is displayed in the **Output** panel
- Burning the bootloader also erases the whole flash
 - Including Matter credentials and application
 - Recommended if changing Matter device types

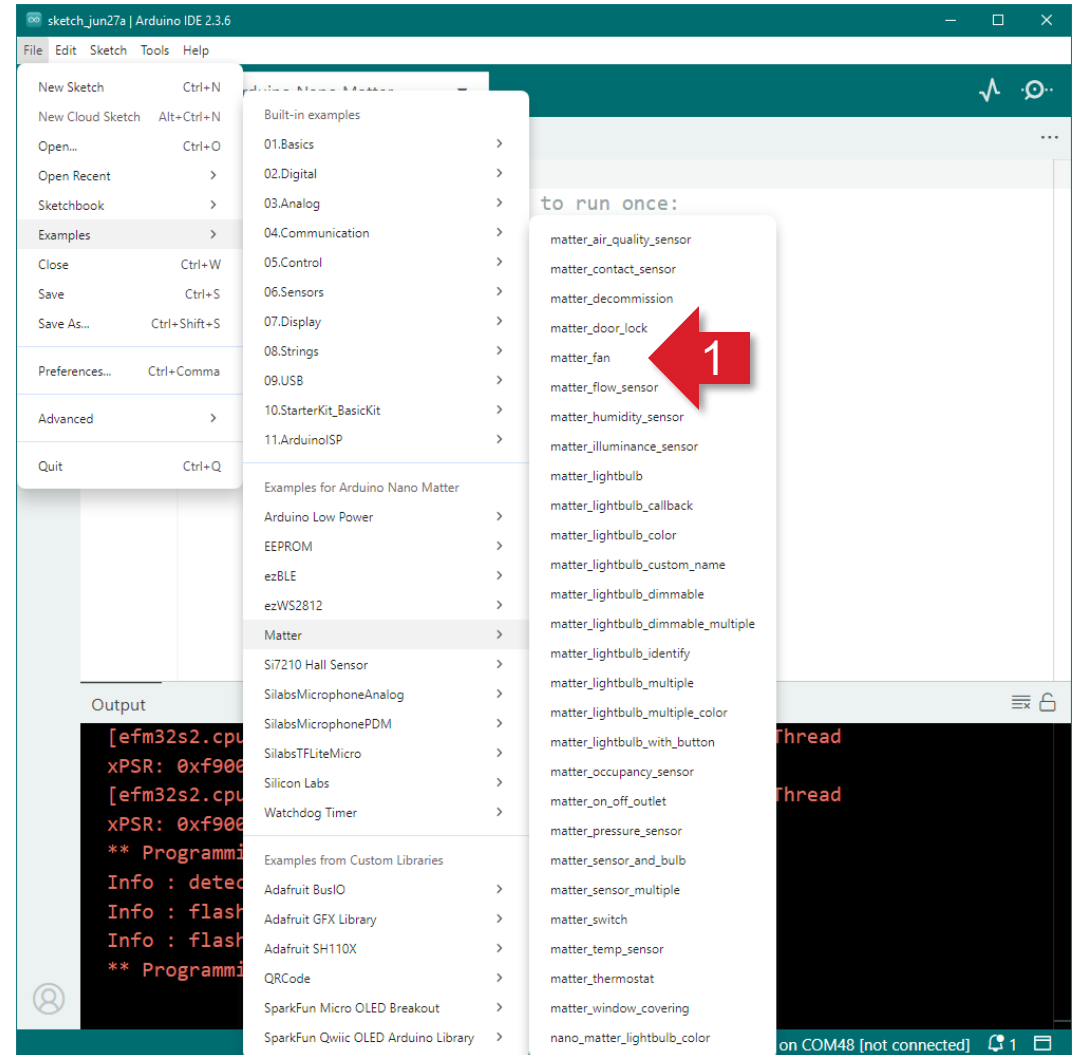


Example Applications



Example Applications

- The Silicon Labs Arduino Core includes a wide range of example applications including 28 Matter over Thread applications
- To create the Matter Fan example application:
 1. From the menu select **File > Examples > Matter > matter_fan**
 2. The example sketch will open in a new window



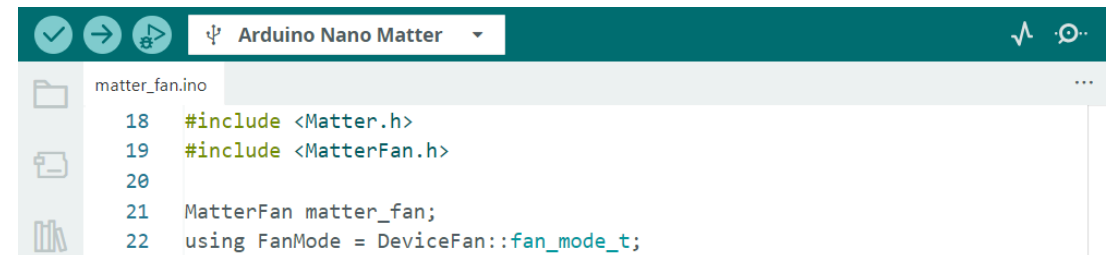
Matter Fan – Source Code

- All Arduino code is structured in a similar way:
 - The **setup()** function is called once at startup to allow application initialization
 - The **loop()** function is called repeatedly to allow main application processing
 - This function should be allowed to return in case other background processing needs to take place



```
sketch_jun27b.ino
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
10
```

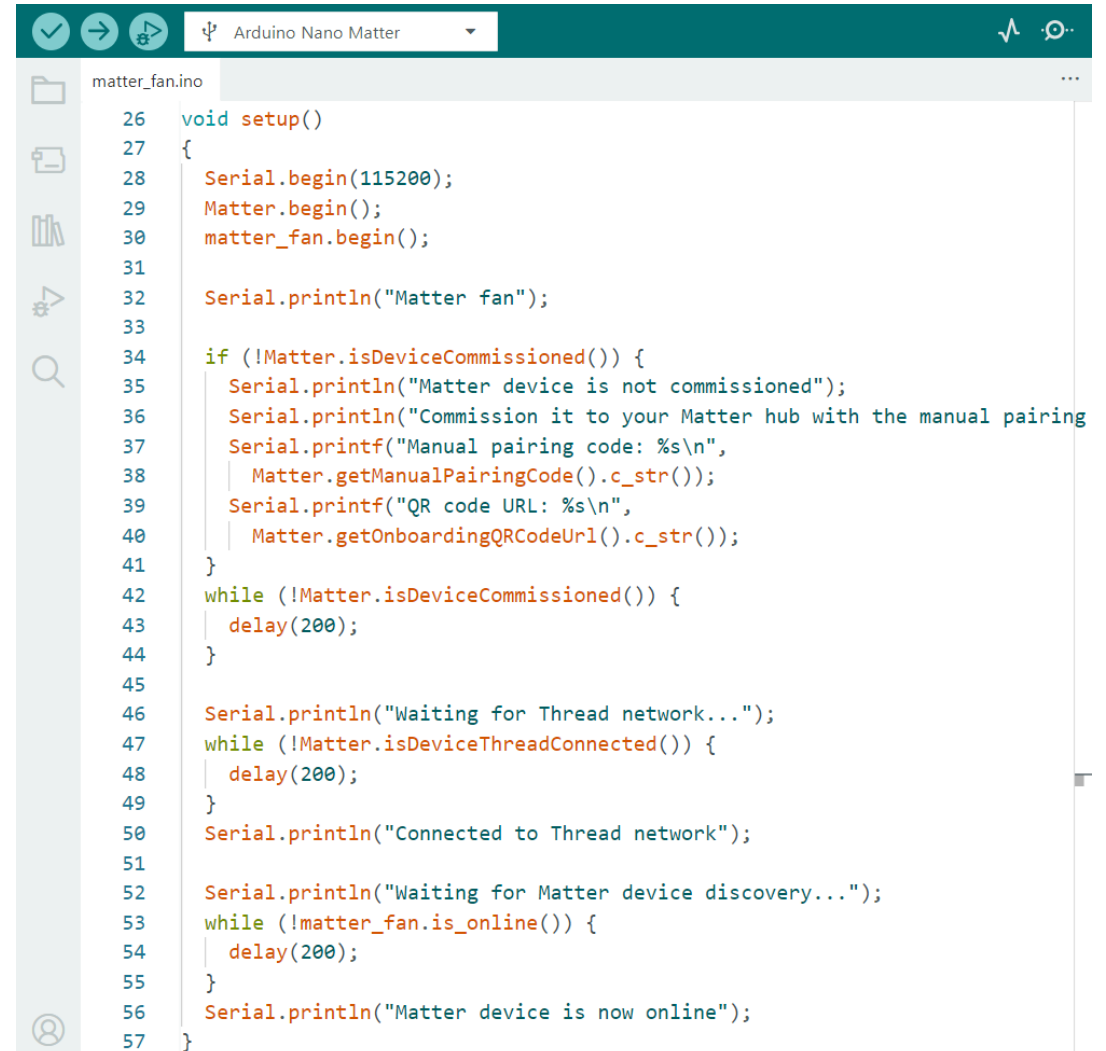
- All Matter examples are structured in a similar way:
 - **Matter.h** provides access to APIs common to all device types
 - **MatterFan.h** provides access to APIs specific to the Matter Fan device type
 - Similar headers for other device types are also available
 - The **MatterFan** object encapsulates the functionality of a Matter Fan
 - The **DeviceFan::fan_mode_t** enumeration is brought in to provide the various fan modes and used as **FanMode**



```
matter_fan.ino
18 #include <Matter.h>
19 #include <MatterFan.h>
20
21 MatterFan matter_fan;
22 using FanMode = DeviceFan::fan_mode_t;
```

Matter Fan – Setup() Function

- Matter examples include code in this function to bring the device into a Matter network before returning:
 - **begin()** functions are called to initialize the common Matter object and the Matter Fan object
 - **Matter.isDeviceCommissioned()** checks whether the device has been commissioned into a network
 - When not in a network, commissioning data is output to the serial port
 - A device only needs to be commissioned once
 - A loop waits for commissioning to be completed
 - **Matter.isDeviceThreadConnected()** checks whether the device is running in a Thread network
 - The device will need to rejoin the Thread network each time it is started
 - The device may form an isolated Thread network until it can rejoin the original network
 - A loop waits for the device to be in a Thread network
 - **Matter.is_online()** returns true when a controller device, such as a hub, queries the device
 - Confirms the device can be controlled by a hub
 - A loop waits for the device to be confirmed online



```
26 void setup()
27 {
28     Serial.begin(115200);
29     Matter.begin();
30     matter_fan.begin();
31
32     Serial.println("Matter fan");
33
34     if (!Matter.isDeviceCommissioned()) {
35         Serial.println("Matter device is not commissioned");
36         Serial.println("Commission it to your Matter hub with the manual pairing");
37         Serial.printf("Manual pairing code: %s\n",
38             Matter.getManualPairingCode().c_str());
39         Serial.printf("QR code URL: %s\n",
40             Matter.getOnboardingQRCodeUrl().c_str());
41     }
42     while (!Matter.isDeviceCommissioned()) {
43         delay(200);
44     }
45
46     Serial.println("Waiting for Thread network...");
47     while (!Matter.isDeviceThreadConnected()) {
48         delay(200);
49     }
50     Serial.println("Connected to Thread network");
51
52     Serial.println("Waiting for Matter device discovery...");
53     while (!matter_fan.is_online()) {
54         delay(200);
55     }
56     Serial.println("Matter device is now online");
57 }
```

Matter Fan – Loop() Function For Controlled Devices

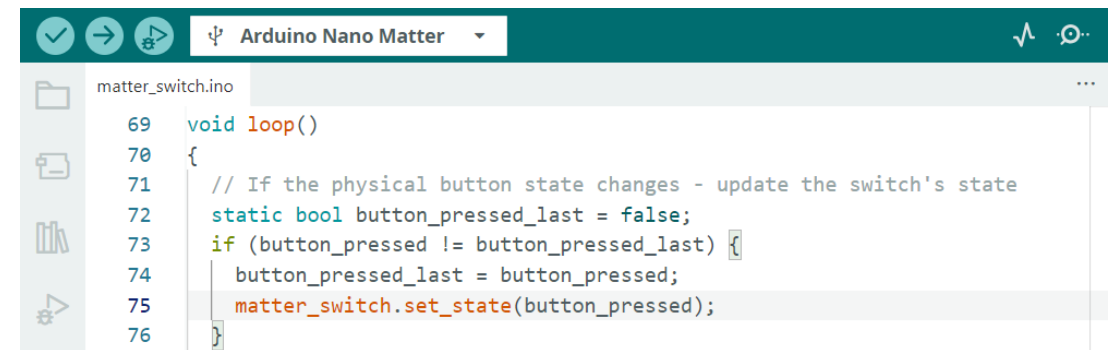
- For device types like the fan, which are controlled, the flow is to monitor the Matter device model and apply changes to hardware:
 - `matter_fan.get_percent()` is called to determine if the fan's speed has been changed remotely
 - `matter_fan.get_mode()` is called to determine if the fan's mode has been changed remotely:
 - ▶ The `matter_fan.set_percent()` function is used to set a new percentage value in the data model
 - ▶ The new percentage will be picked up in the next call to the `loop()` function
 - This example simply outputs changes to these values to the serial port
 - ▶ Code can be added here to control hardware devices appropriately
 - ▶ Some examples already have code to control hardware such as the light examples controlling the on-board LED



```
55 void loop()
56 {
57     static uint8_t fan_last_speed = 0u;
58     uint8_t fan_current_speed = matter_fan.get_percent();
59
60     if (fan_current_speed != fan_last_speed) {
61         fan_last_speed = fan_current_speed;
62         Serial.print("Fan speed: ");
63         Serial.print(fan_current_speed);
64         Serial.println("");
65     }
66
67     static FanMode fan_last_mode = FanMode::Off;
68     FanMode fan_current_mode = matter_fan.get_mode();
69
70     if (fan_current_mode != fan_last_mode) {
71         fan_last_mode = fan_current_mode;
72         switch (fan_current_mode) {
73             case FanMode::Off:
74                 Serial.println("Fan mode: Off");
75                 matter_fan.set_percent(0);
76                 break;
77             case FanMode::Low:
78                 Serial.println("Fan mode: Low");
79                 matter_fan.set_percent(20);
80                 break;
81             case FanMode::Med:
82                 Serial.println("Fan mode: Medium");
83                 matter_fan.set_percent(50);
84                 break;
85             case FanMode::High:
86                 Serial.println("Fan mode: High");
87                 matter_fan.set_percent(100);
88                 break;
89             case FanMode::On:
90                 Serial.println("Fan mode: On");
91                 matter_fan.set_percent(fan_last_speed);
92                 break;
```

Matter Switch – Loop() Function For Controller Devices

- For other device types, that act as controllers, the flow is reversed to monitor the hardware device model and apply changes to the Matter device model
 - This applies to devices like switches and sensors
 - For the switch example, the `matter_switch.set_state()` function is called to update the Matter state when hardware button presses are detected



```
69 void loop()
70 {
71     // If the physical button state changes - update the switch's state
72     static bool button_pressed_last = false;
73     if (button_pressed != button_pressed_last) {
74         button_pressed_last = button_pressed;
75         matter_switch.set_state(button_pressed);
76     }
```

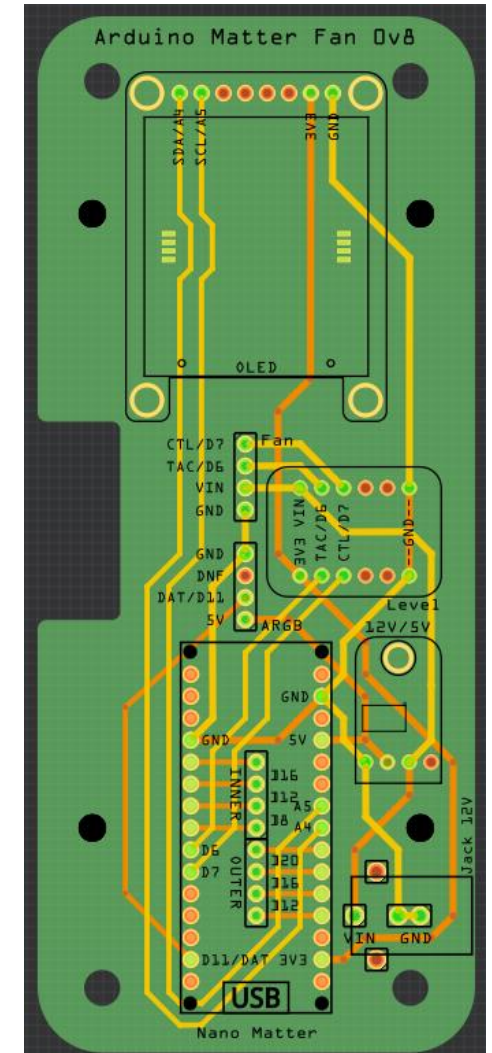
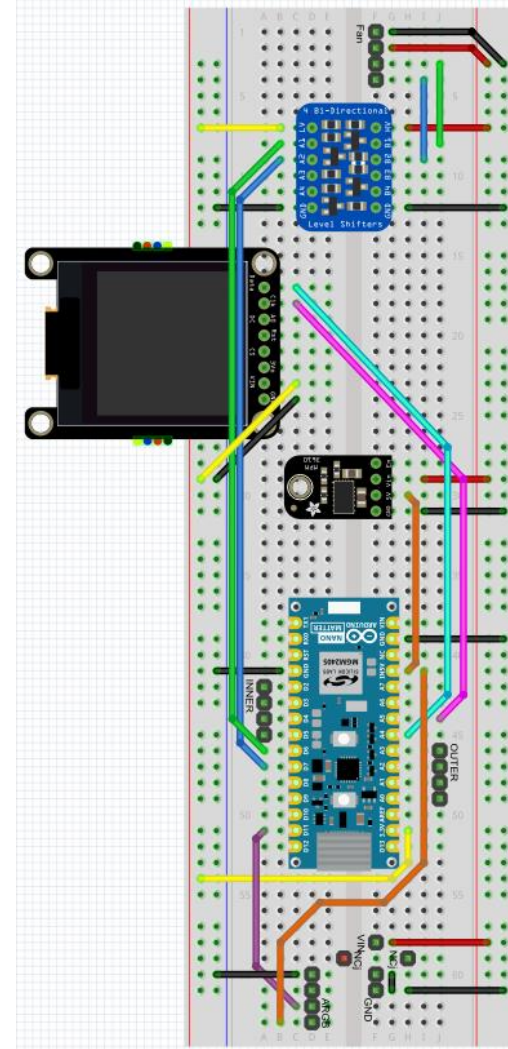
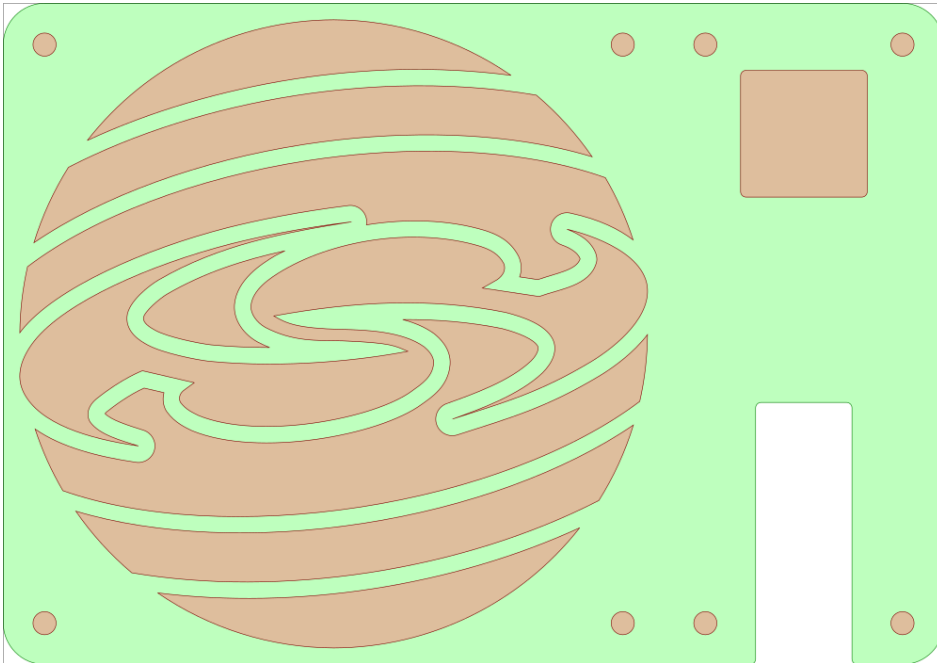
Matter Fan – Adding a Hardware Fan

- The [dev_lab_arduino_matter_fan](#) folder in the [Silicon Labs Training Examples](#) repository contains an enhanced version of the Matter Fan code to control and monitor a hardware fan along with supporting components:
 - A PWM 4-wire fan, such as the [Cooler Master MasterFan MF120 Halo 2](#)
 - ▶ The fan's speed is controlled via a PWM driven input
 - ▶ The fan's tachometer output can be monitored to calculate the fan's RPM
 - ▶ If the fan has addressable RGB LEDs, these are also controlled
 - An [Adafruit 128x128 Monochrome OLED](#) display
 - ▶ Displays the commissioning QR Code on start-up when not commissioned into a network
 - ▶ Otherwise displays the fan's connection state, on/off state, speed and RPM values
 - An [Adafruit BSS138 Logic Level Converter](#)
 - ▶ To convert the input and output lines between the fan's 12V and the Arduino Nano Matter's 3.3V levels
 - An [Adafruit MPM3610 Buck Converter 12V to 5V](#)
 - ▶ Powering the Arduino Nano Matter and fan's ARGB LEDs
 - A 12V power supply



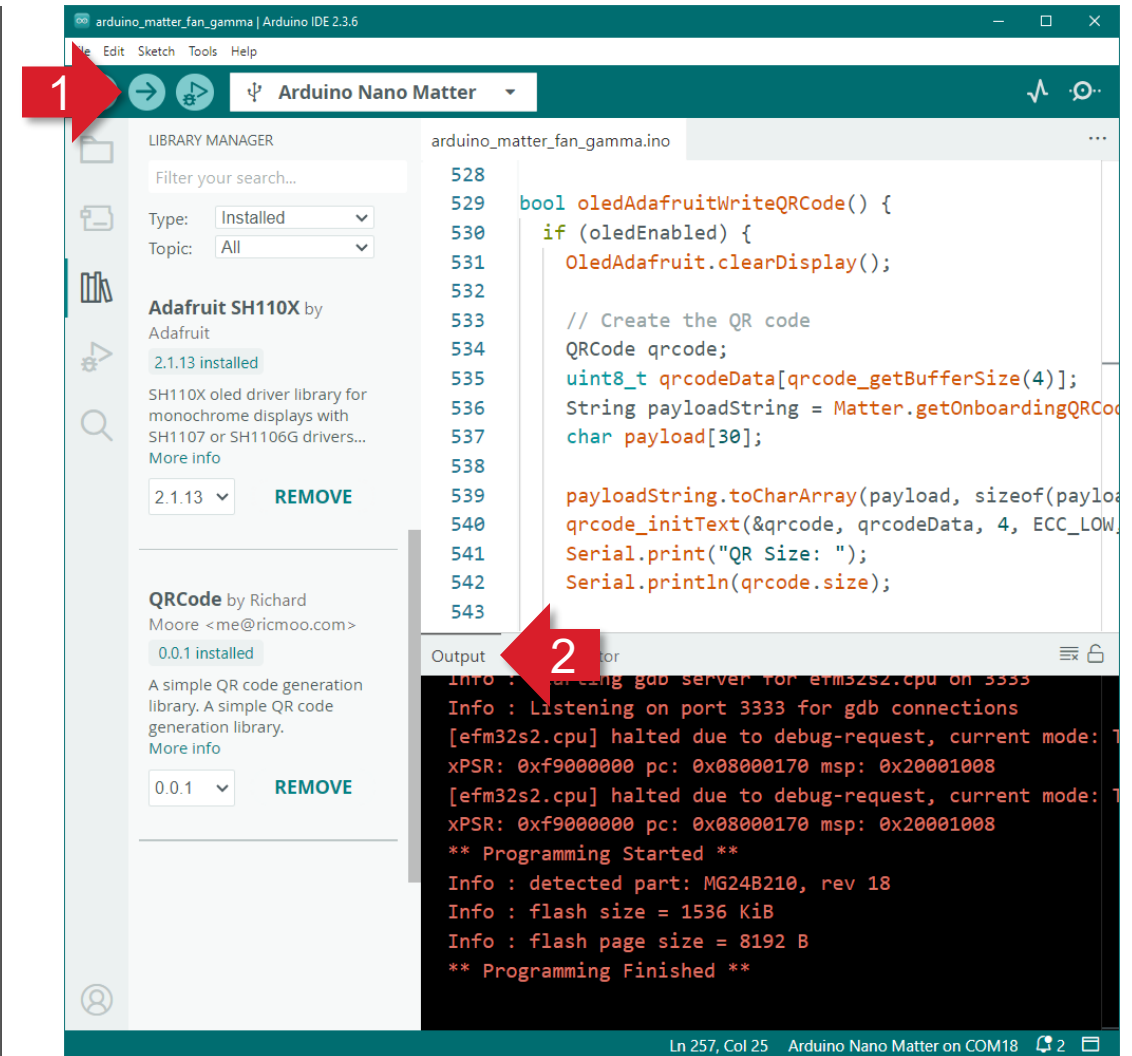
Matter Fan – Breadboard, Schematic, PCB and Enclosure

- The `dev_lab_arduino_matter_fan` folder in the [Silicon Labs Training Examples](#) repository also contains hardware files:
 - A Fritzing file with layouts for a breadboard, schematic and PCB, plus a separate Gerber ZIP file for the PCB
 - SVG files to create a laser-cut case from acrylic



Matter Fan – Compile and Flash

- To compile and flash the enhanced application:
 1. Click the **Upload** button in the IDE
 2. Compilation and upload progress is shown in the **Output** panel

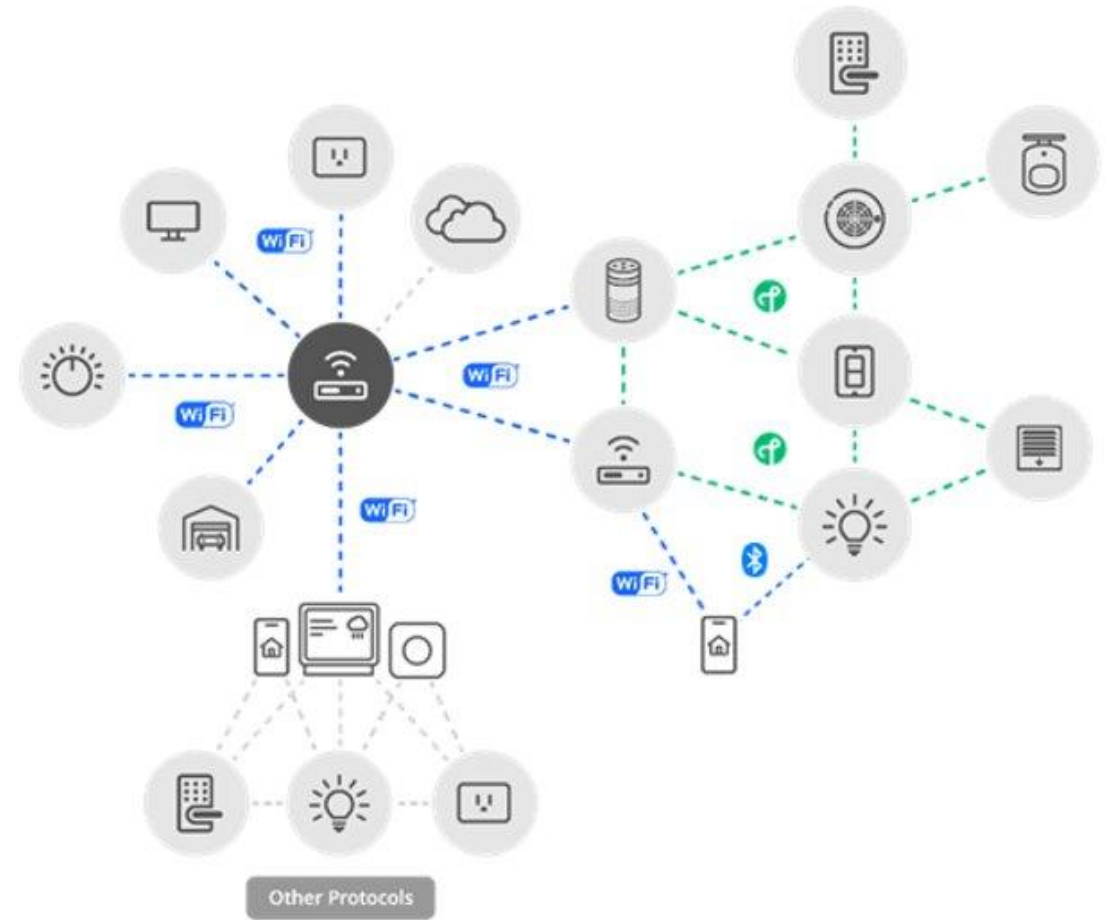


Matter Theory



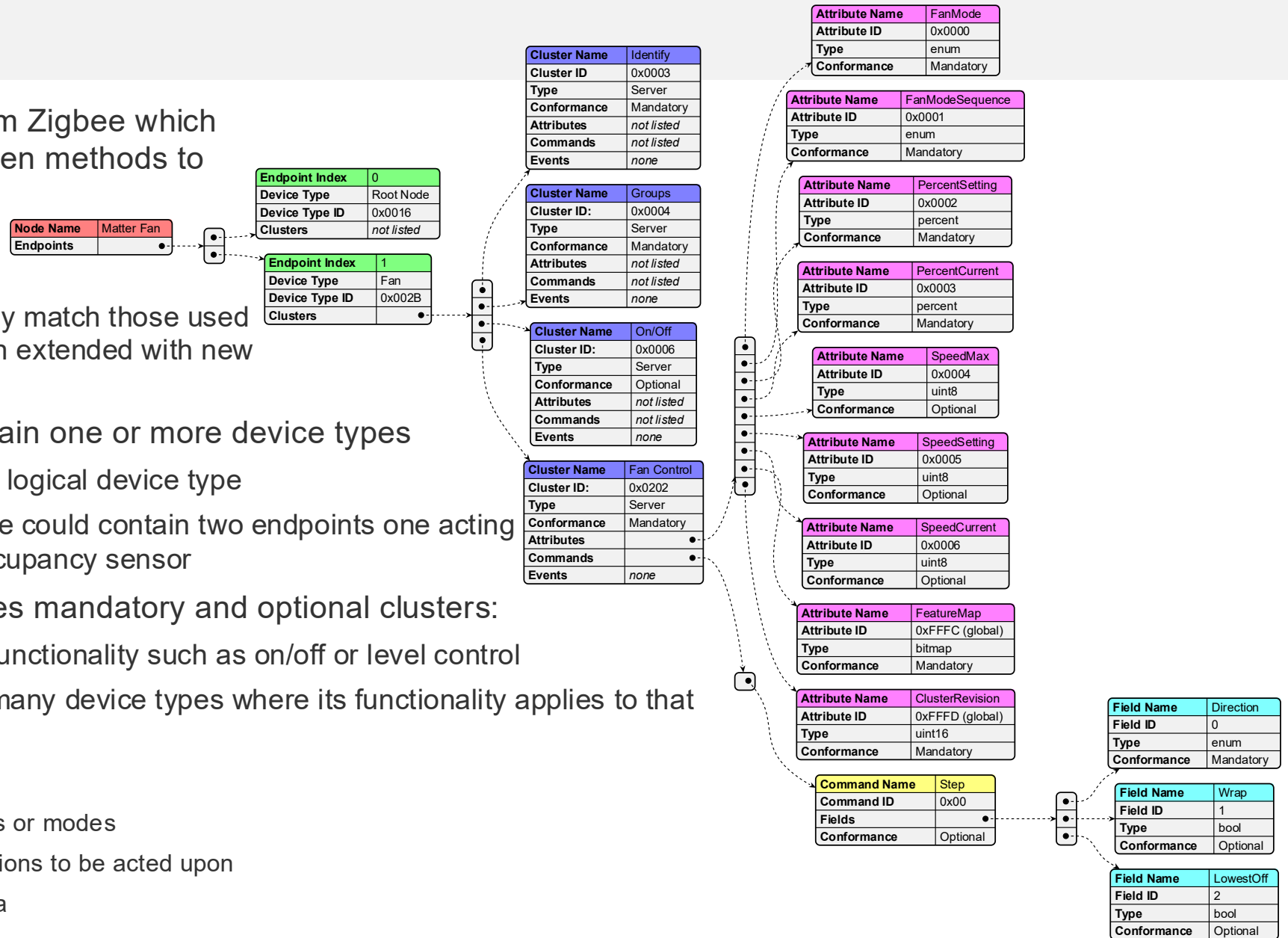
Network Topology

- 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
- Matter Controllers are used to control MADs, including commissioning them into the network
 - Typically, these are smart phones
- Bridges can be used to link to other protocols, like ZigBee and Z-Wave



Data Model

- Matter borrows heavily from Zigbee which provides a set of field-proven methods to form, control and monitor devices
 - Device Types and Clusters used in Matter do not exactly match those used in Zigbee as they have been extended with new functionality as needed
- A Matter product may contain one or more device types
 - Each endpoint represents a logical device type
 - For example, a single device could contain two endpoints one acting as a light and one as an occupancy sensor
- A Device Type encapsulates mandatory and optional clusters:
 - A cluster provides specific functionality such as on/off or level control
 - A cluster can be reused in many device types where its functionality applies to that device type
 - A cluster is made up of
 - Attributes, representing states or modes
 - Commands, which are operations to be acted upon
 - Events, that log historical data



Commissioning and Operation



Commissioning Codes

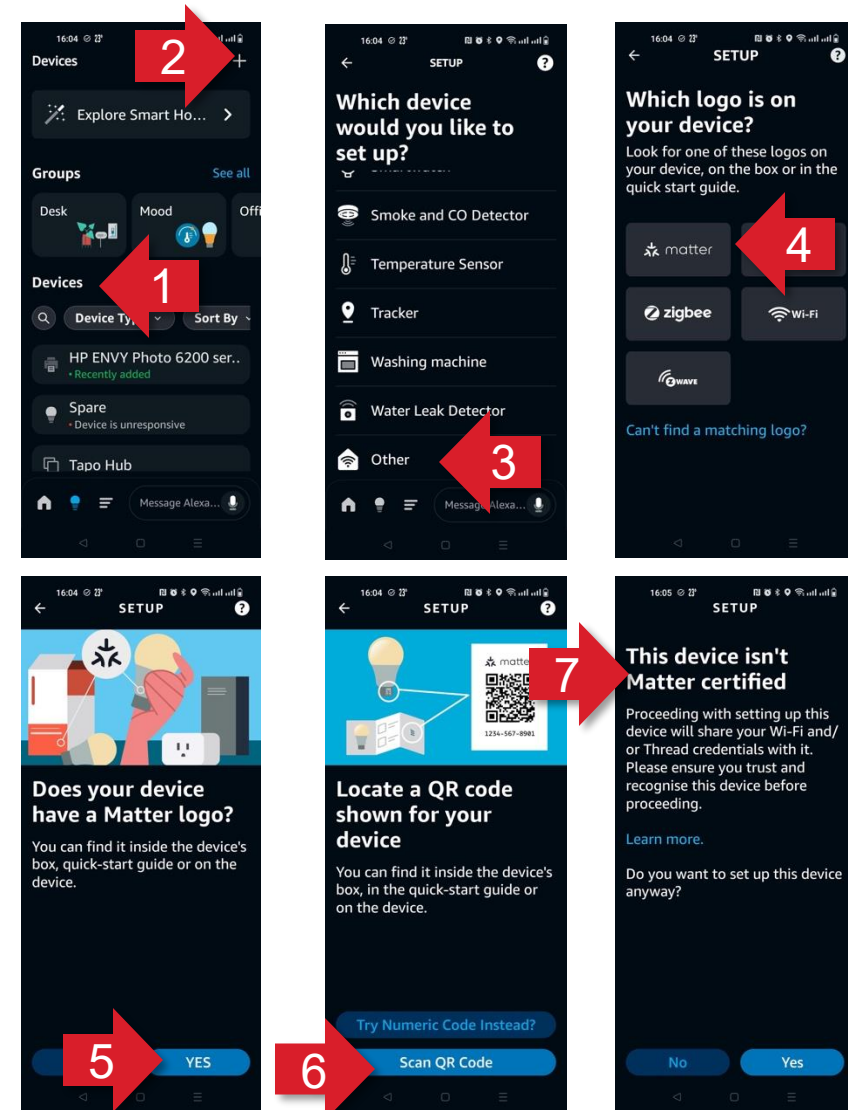
- Commissioning devices into Matter network has been designed to be simple:
 - A QR Code is scanned in the ecosystem app
 - ▶ A manual pairing code can be entered as an alternative
 - This allows the app to discover the device advertising on Bluetooth and connect to it to provide Thread network credentials
 - The joining device then uses the credentials to join the Thread network and authenticate itself
- To access the commissioning QR Code for an Arduino Matter device:
 1. Open the serial monitor
 2. Ensure the baud rate is set to 115200
 3. Reset the board
 4. Copy the QR code URL from the debug output
 - ▶ The manual pairing code is also output to the Serial Monitor
 - ▶ For our enhanced fan, the QR Code is displayed on the OLED
 5. Paste the URL into a browser

```
arduino_matter_fan_gamma | Arduino IDE 2.3.6
File Edit Sketch Tools Help
Arduino Nano Matter
arduino_matter_fan_gamma.ino
529 bool oledAdafruitWriteQRCode() {
530     if (oledEnabled) {
531         OledAdafruit.clearDisplay();
532     }
533     // Create the QR code
Output Serial Monitor X
Message (Enter to send message to 'Arduino Nano Matter' on 'COM18') New Line 115200 baud
MATTER FAN (beta)
OLED: 0
Matter device is not commissioned
Commission it to your Matter hub with the manual pairing code or QR code
Manual pairing code: 34970112332
QR code URL: https://project-chip.github.io/connectedhomeip/qrcode.html?data=MT%3A6FCJ142C00KA0648G00
Waiting for commissioning...
```



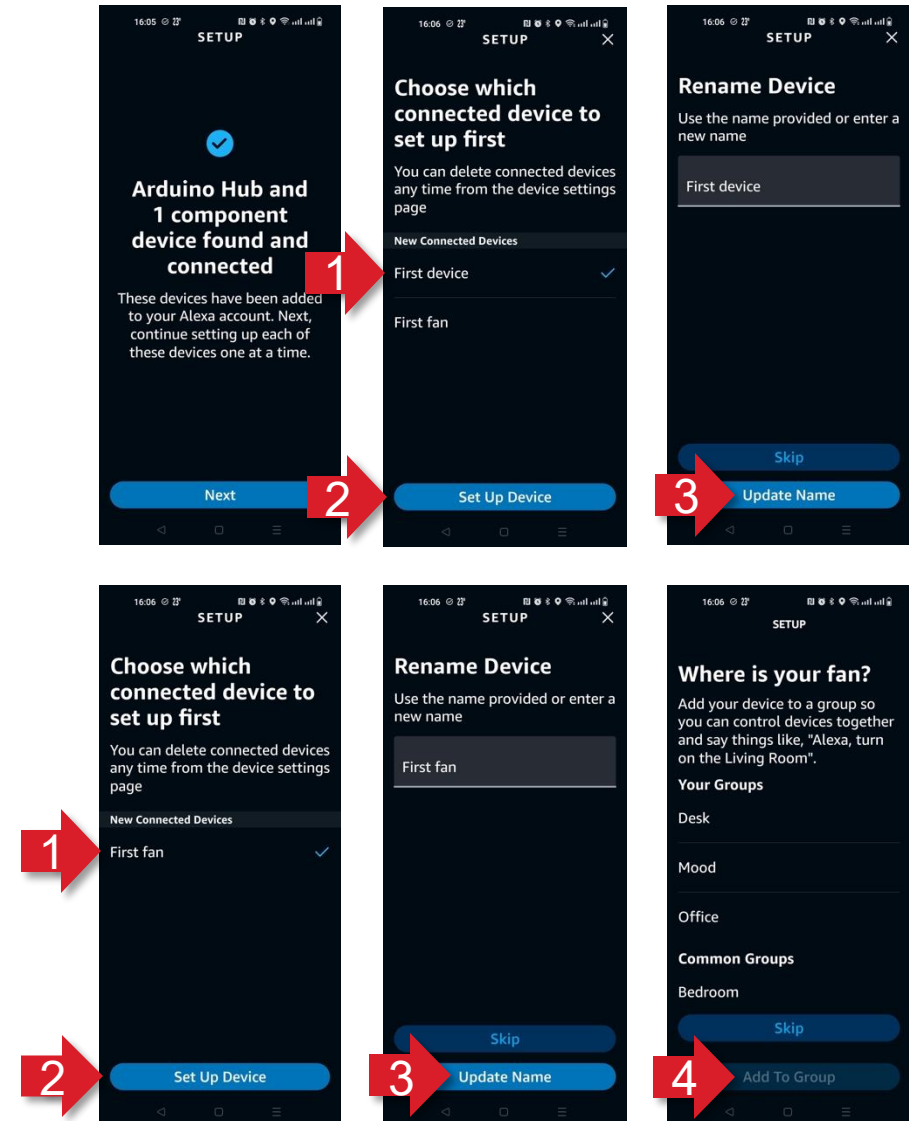
Commissioning Amazon Alexa

- During commissioning the OLED is updated to display the Matter network state
- The RGB LED also indicates Matter network state:
 - Red, during commissioning, joining and discovery, flashing more quickly when moving through the steps
 - Green, when in the network and in the off mode
 - Blue when in the network and in an on mode
- The commissioning process is similar in all major ecosystems, for Amazon Alexa:
 1. Go to the **Device** page
 2. Click the **Add** button
 3. Select **Other**
 4. Select **Matter**
 5. Click **Yes** when asked if the device has a Matter logo
 6. Select **Scan QR Code**
 7. The Arduino Matter examples use test certificates, you may be prompted to accept these



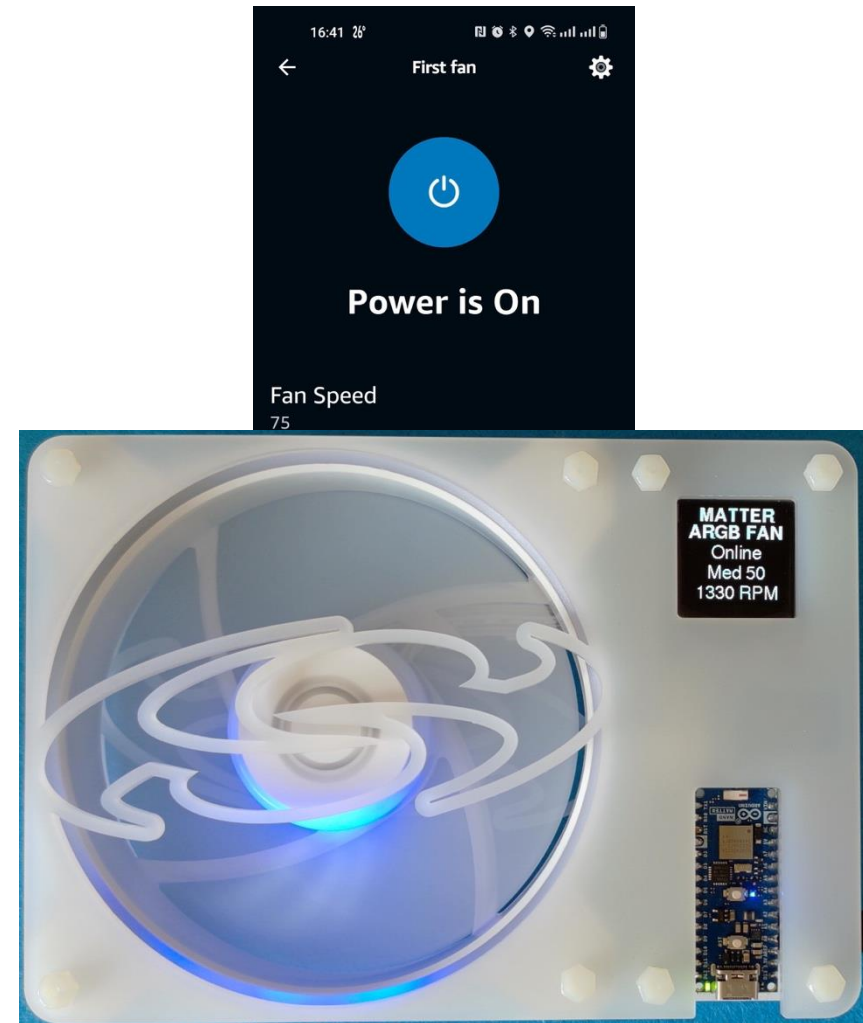
Device Setup Amazon Alexa

- Two devices are found:
 - The First Fan device is the fan application
 - The First Device is a hub and appears as a side effect of creating the fan endpoint at runtime
- To set up the devices:
 - Select a device
 - Click **Set Up Device**
 - Devices can be renamed
 - The Fan endpoint can be added to a group to allow group control



Device Operation Amazon Alexa

- The fan can be controlled using controls in the app, via a voice assistant and the on-board button:
 - Turning the fan on and off
 - Setting the fan mode
 - Setting the fan speed
- The fan displays information on the OLED:
 - Matter state, fan mode and speed, RPM reading
- The LEDs indicate the fan mode and RPM:
 - Green when in an off mode, blue when in an on mode
 - The on-board LED flashes relative to the measured RPM
 - The fan's LEDs rotate relative to the measured RPM




Next Steps



Next Steps – Matter over Thread in Arduino IDE

- There are lots more Matter examples to explore in the Arduino IDE
- [Quick Start – Arduino Nano Matter](#) video:
 - Walks through the setup of the Arduino IDE for Matter
 - Adapts the Matter Dimmable Lightbulb example to animate LEDs
- [Dev Lab – Arduino Matter Mood Light](#) video:
 - Creates a mood light using a single device with two color bulb endpoints, creating plasma effects between the two colors
- [Dev Lab – Arduino Matter Occupancy Sensor](#) video:
 - Walks through creating an occupancy sensor and how to control other devices based on its state
- [Silicon Labs Arduino Core on GitHub](#):
 - Contains a useful readme, including pin outs for all the boards
 - Contains the source code for the Arduino Core
- [Arduino Nano Matter User Manual](#):
 - Walks through setting up the board
 - Has tutorials including commissioning into the major Matter ecosystems





QUICK START
ARDUINO
NANO
MATTER

DEV LAB
ARDUINO
MATTER
MOOD LIGHT

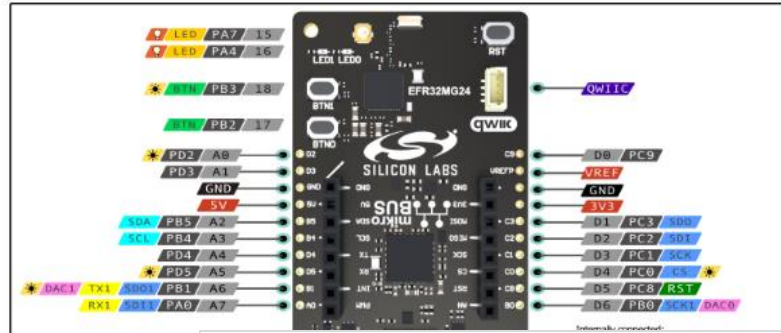
DEV LAB
ARDUINO MATTER
OCCUPANCY
SENSOR

README

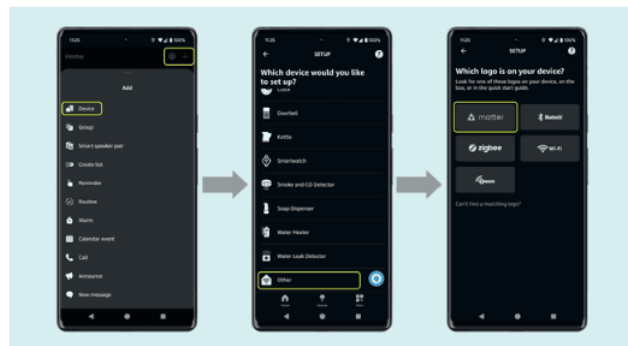
xG24 Explorer Kit  

[Product page](#) | [User guide](#)

Pinout diagram



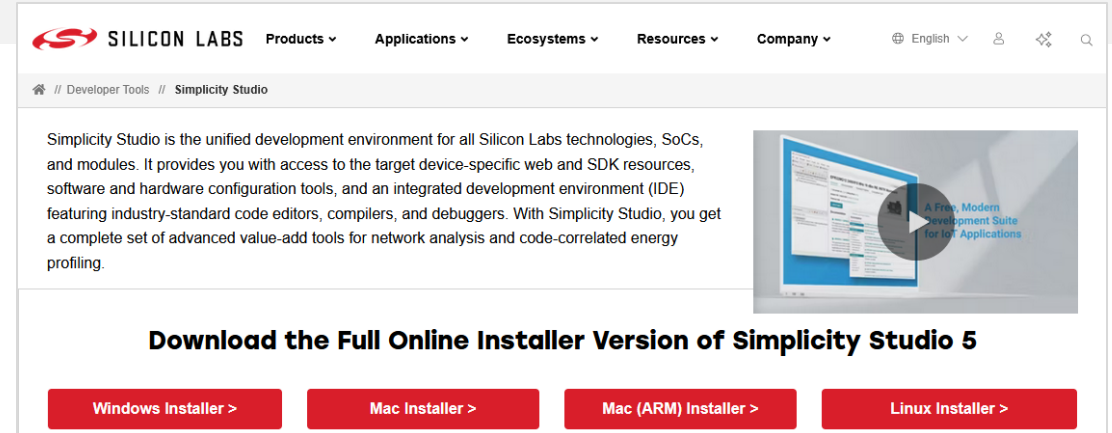
To commission your device, open the Amazon Alexa app, click on the upper right + symbol, select **device** and select the **Matter** logo option:



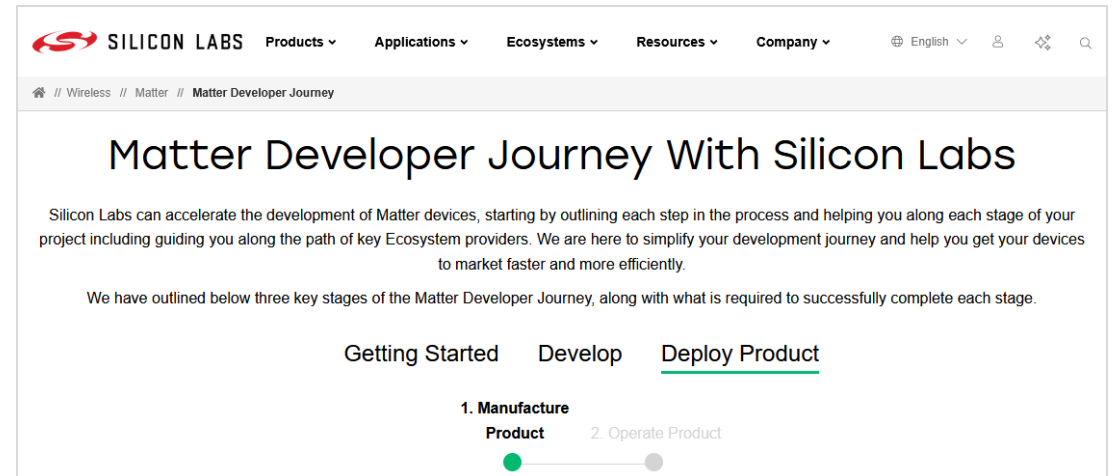
Adding a new Matter device to Amazon Alexa

Next Steps – Matter in Simplicity Studio

- Simplicity Studio is the IDE for all Silicon Labs boards and technologies including Matter over Thread and Matter over Wi-Fi:
 - [Simplicity Studio download](#)
 - Example applications are built into the IDE
 - Additional example applications are available on [GitHub](#)
 - General information is available from the [Silicon Labs Matter webpage](#)
 - The [Matter Developer Journey](#) walks through the process of creating a Matter product
 - The [Matter documentation site](#) has detailed information on developing Matter applications
 - There are a series of Quick Start videos on YouTube covering Matter over Thread development in Simplicity Studio, begin with [Quick Start – Matter over Thread Light](#)
 - A series of Quick Start videos also covers the installation and use of the Simplicity Studio IDE, begin with [Quick Start – Simplicity Studio 5 Installation](#)



The screenshot shows the Silicon Labs website's page for downloading Simplicity Studio. The header includes the Silicon Labs logo and navigation links for Products, Applications, Ecosystems, Resources, and Company. The breadcrumb trail indicates the path: Developer Tools // Simplicity Studio. The main content area describes Simplicity Studio as a unified development environment for all Silicon Labs technologies, SoCs, and modules. It lists features like access to target device-specific web and SDK resources, software and hardware configuration tools, and an integrated development environment (IDE) with industry-standard code editors, compilers, and debuggers. A call to action, "Download the Full Online Installer Version of Simplicity Studio 5", is followed by four red buttons: "Windows Installer >", "Mac Installer >", "Mac (ARM) Installer >", and "Linux Installer >".



The screenshot shows the Silicon Labs website's page for the Matter Developer Journey. The header includes the Silicon Labs logo and navigation links for Products, Applications, Ecosystems, Resources, and Company. The breadcrumb trail indicates the path: Wireless // Matter // Matter Developer Journey. The main heading is "Matter Developer Journey With Silicon Labs". The text describes how Silicon Labs can accelerate the development of Matter devices by outlining each step in the process and helping along each stage. It mentions guiding users along the path of key Ecosystem providers to simplify their development journey and help them get their devices to market faster and more efficiently. Below this, it states that three key stages of the Matter Developer Journey are outlined: "Getting Started", "Develop", and "Deploy Product". A progress bar shows "1. Manufacture Product" as the current stage, with "2. Operate Product" as the next stage.





SILICON LABS

CONNECTED INTELLIGENCE