



基于Simplicity Studio v5的 Matter GSDK开发指南

余发明

2023-02-21

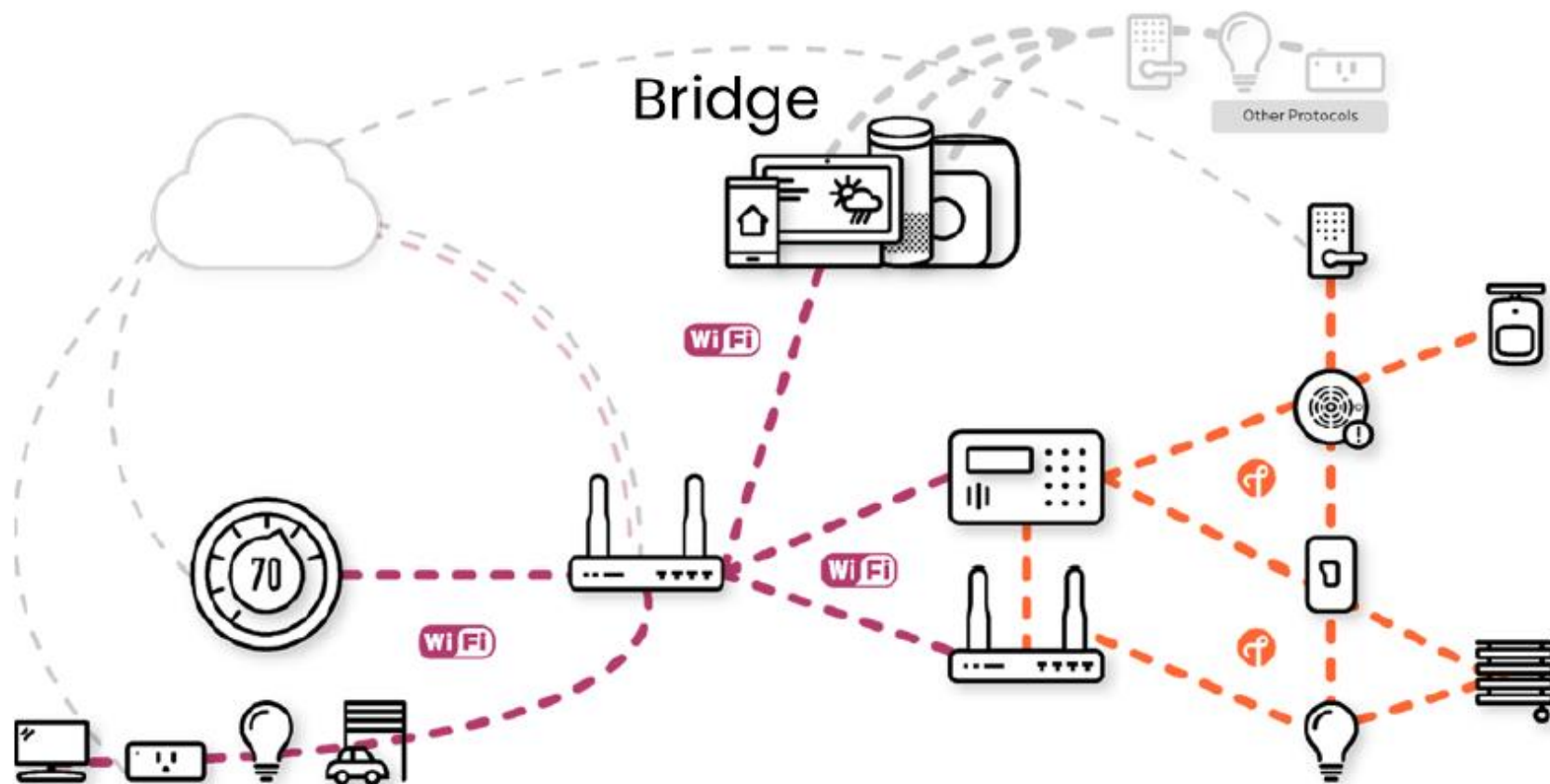


大纲 (45')

- 为什么要开发**Matter**
- **Matter**开发方式介绍
- **SSv5(Simplicity Studio v5)**介绍
 - 下载与安装
 - 在SSv5内安装工具包和SDK
 - 使用SSv5内安装的资源
- 基于**SSv5**创建**Matter**工程
 - 工程目录文件介绍
 - 软件组件应用介绍

为什么要开发Matter

- Matter over Thread
- Matter over Wi-Fi
- Other Protocols Bridge to Matter



有哪些方式开发Matter产品

- 基于**CHIP(connectedhomeip)** 开源SDK
 - SDK位置: <https://github.com/project-chip/connectedhomeip>
 - 所有平台或方案的开发者共同维护
 - 需要VPN连接国外网站
 - Linux/MacOS系统命令行操作
- 基于芯科科技**Simplicity Studio v5**集成开发平台
 - Simplicity Studio v5平台优势
 - 自动识别配套的硬件开发板所支持的工程
 - 通过平台界面选择安装支持Matter的GSDK及编译环境, 不需要VPN连接外网
 - 通过界面配置&编译工程, 无需复杂的命令行
 - 一系列配套的开发辅助工具
 - GSDK优势: 在CHIP开源SDK基础上针对芯科科技Matter方案结合自身的开发工具进行优化
 - SSv5开发环境: 支持多种操作系统(Windows/Linux/MacOS)

SSv5(Simplicity Studio v5)介绍

- 下载&安装
- 在SSv5内安装工具包和SDK
- 使用SSv5内安装的资源

- 对电脑空间要求

System Requirements

Operating Systems

Operating System	Tested Version
Windows	Windows 10 (64-bit)
macOS	10.14 Mojave 10.15 Catalina* 11.x Big Sur* 12.x Monterey* * If trying to use the Keil 8051 or IAR toolchains, Click Here
Linux	Ubuntu 20.04 LTS

Hardware

Hardware Component	Item
CPU	1 GHz or better
Memory	1 GB RAM minimum, 8 GB recommended for Wireless Protocol development
Disk Space	600 MB disk space for minimum FFD installation 7 GB for Wireless Dynamic Protocol support

SSv5——下载&安装

- 下载地址: <https://www.silabs.com/developers/simplicity-studio>

Download the Full Online Installer Version of Simplicity Studio 5



Windows Installer >



Mac Installer >



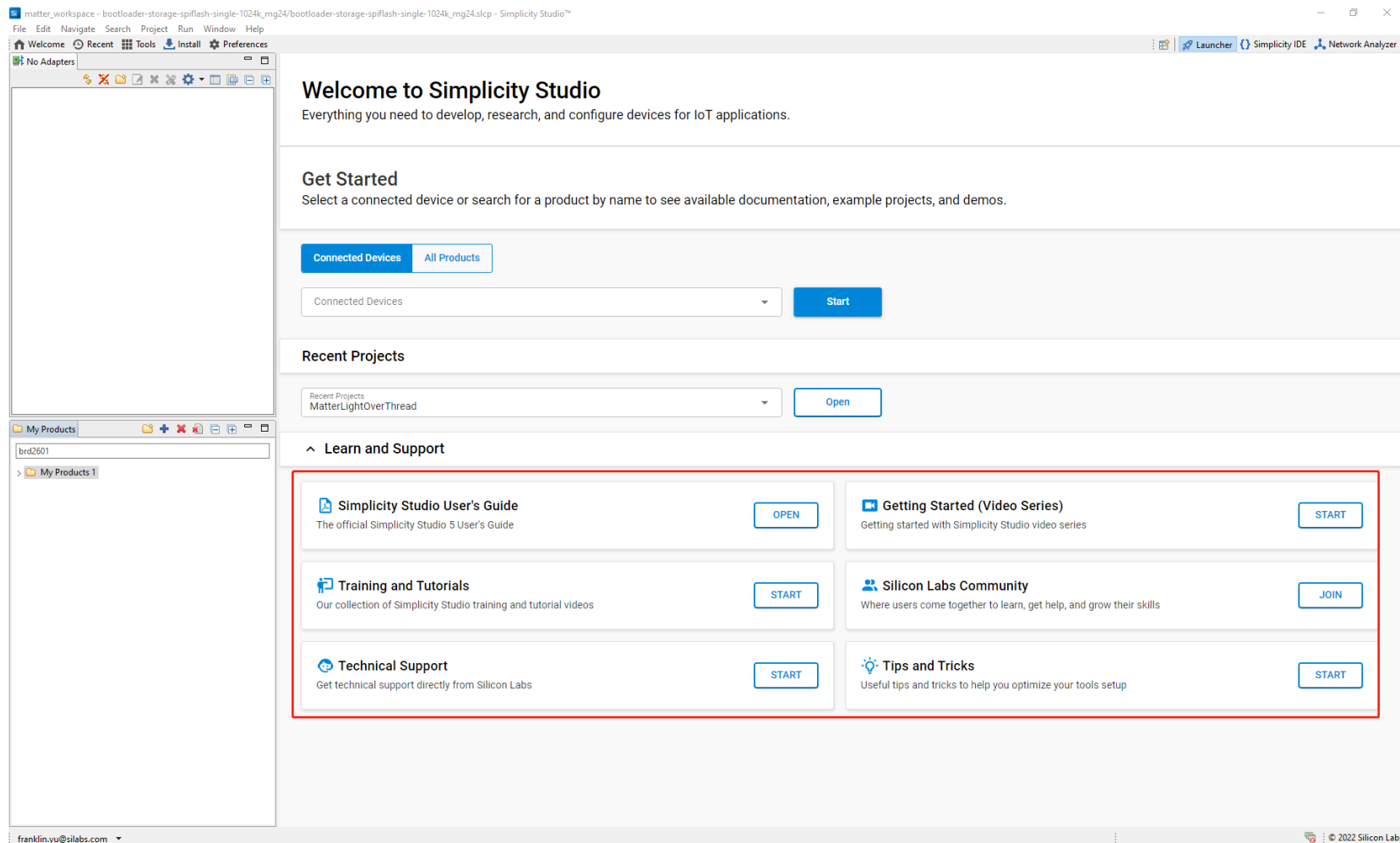
Linux Installer >

[*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.

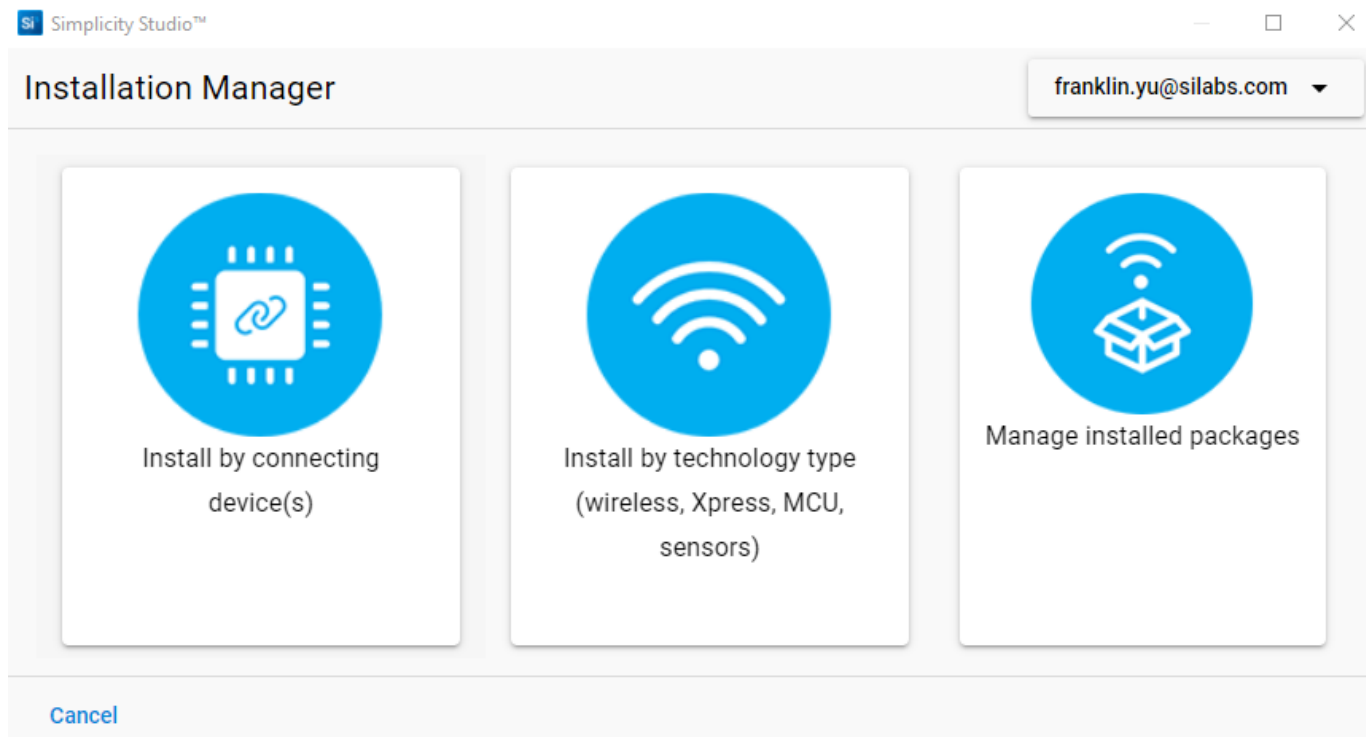
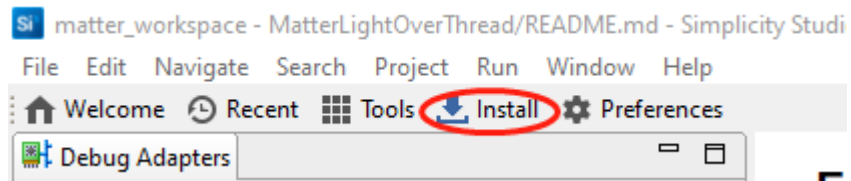
SSv5——启动界面

- 安装完成后，启动SSv5界面如下



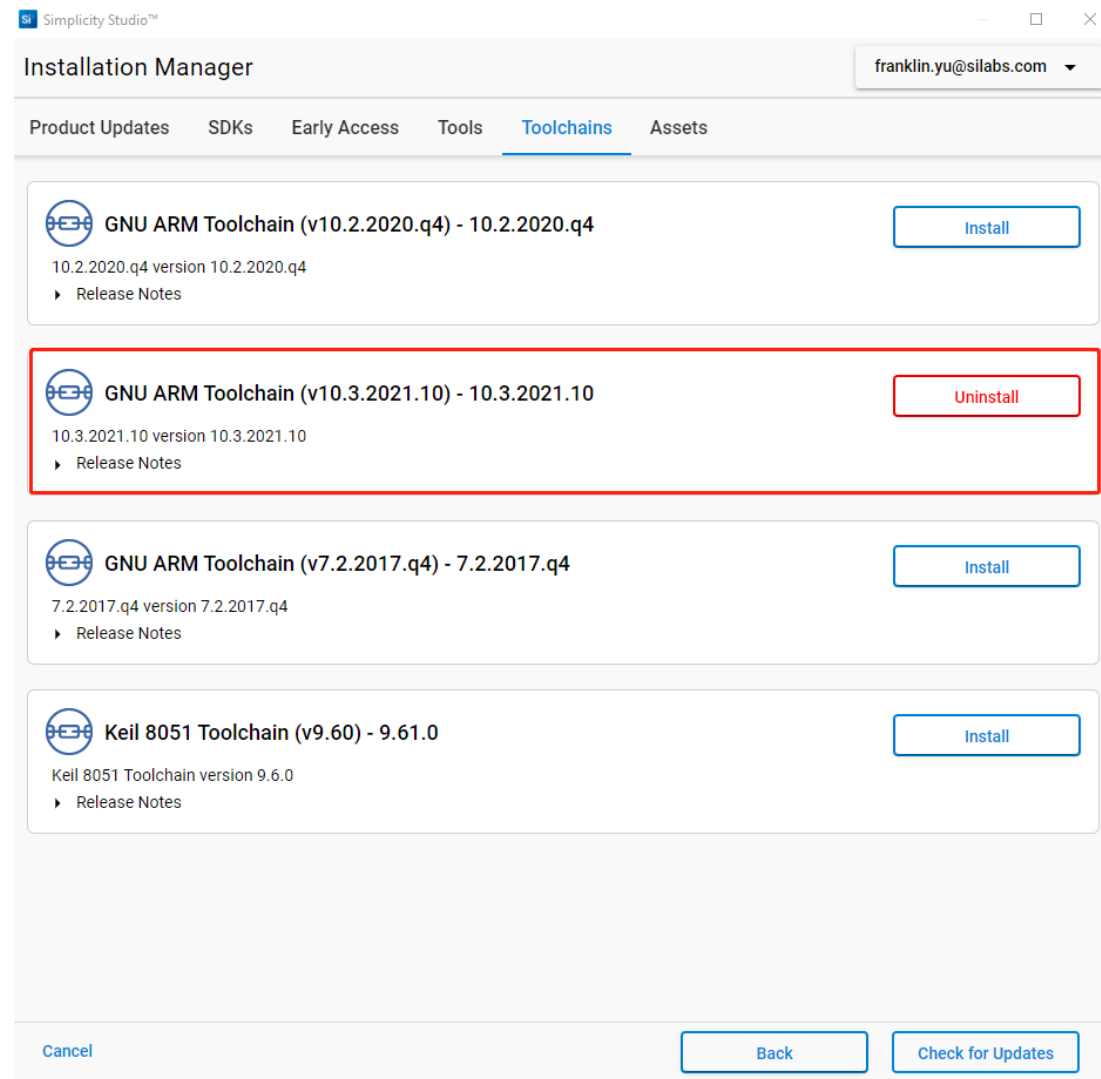
SSv5——安装工具包和SDK

- 点击**Install**按钮，安装所需要的工具、**SDK**等组件



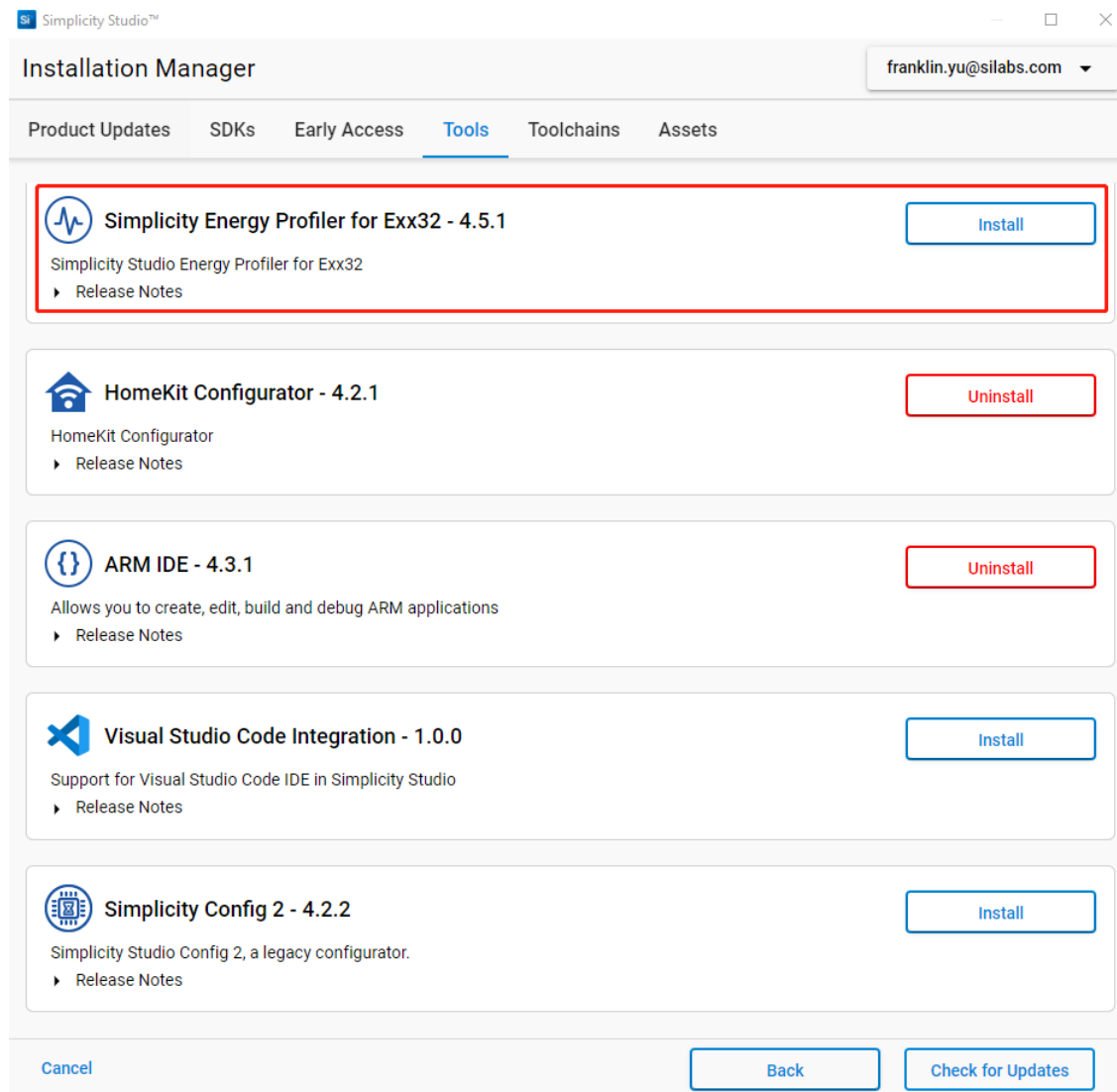
SSv5——安装工具链

- 在**Toolchains**页面选择工具链进行安装
- 通常我们选择最新版



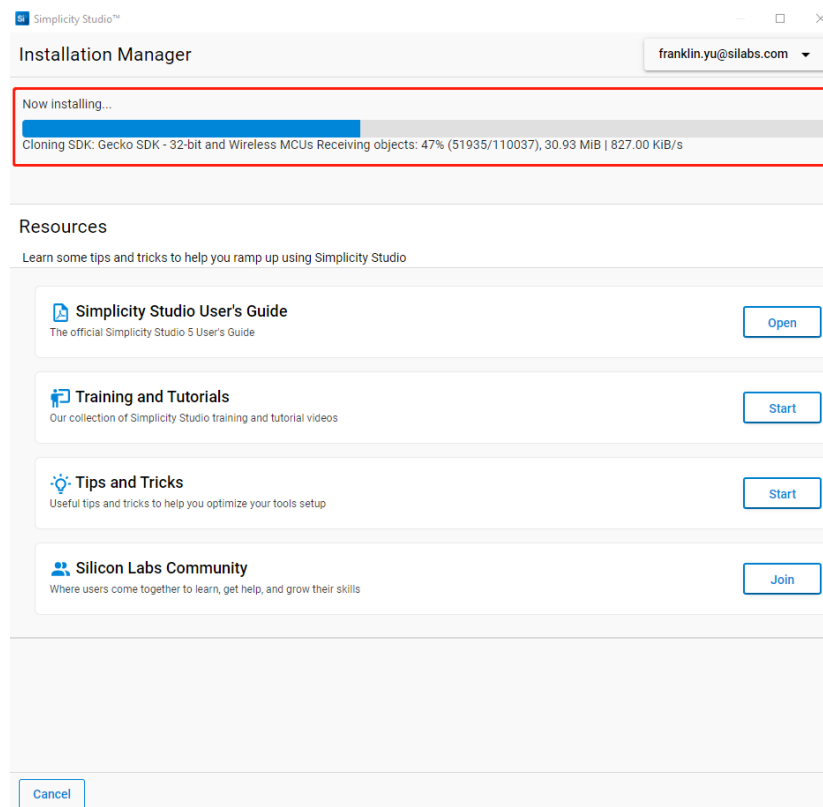
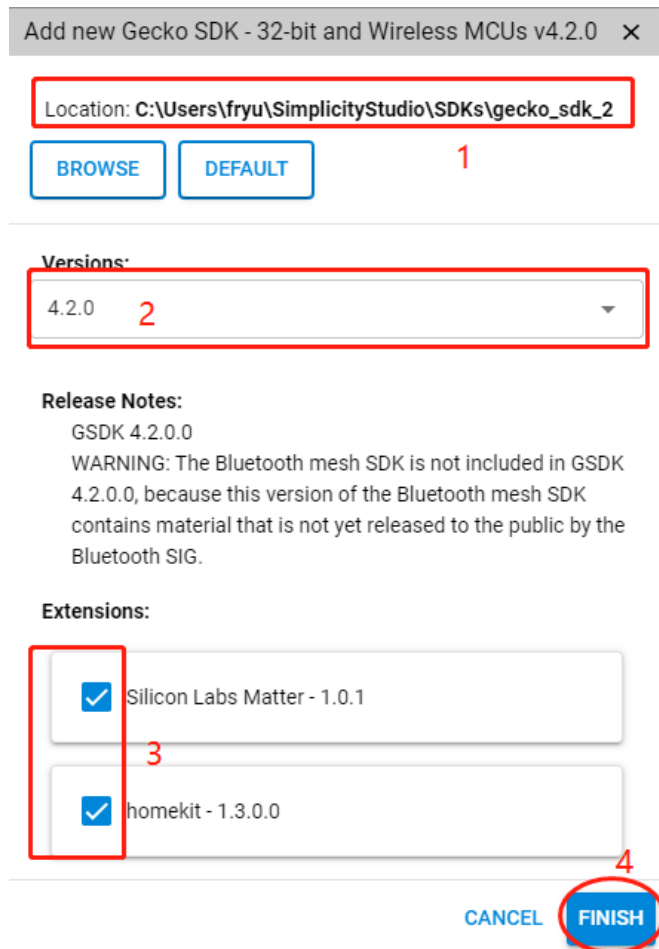
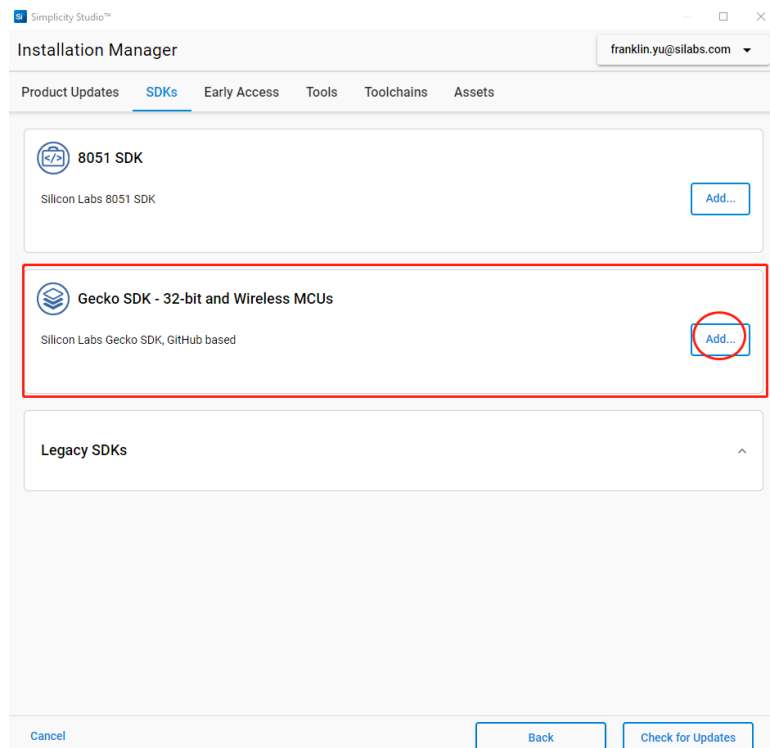
SSv5——安装工具

- 在**Tools**页面选择工具进行安装
- 如右图红框部分的功耗测试工具



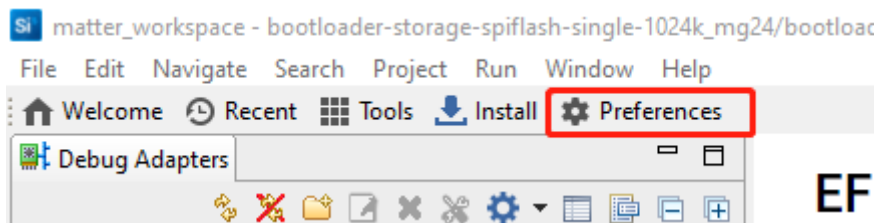
SSv5——直接安装SDK

- 选择SDKs页面
- 选择Gecko SDK安装

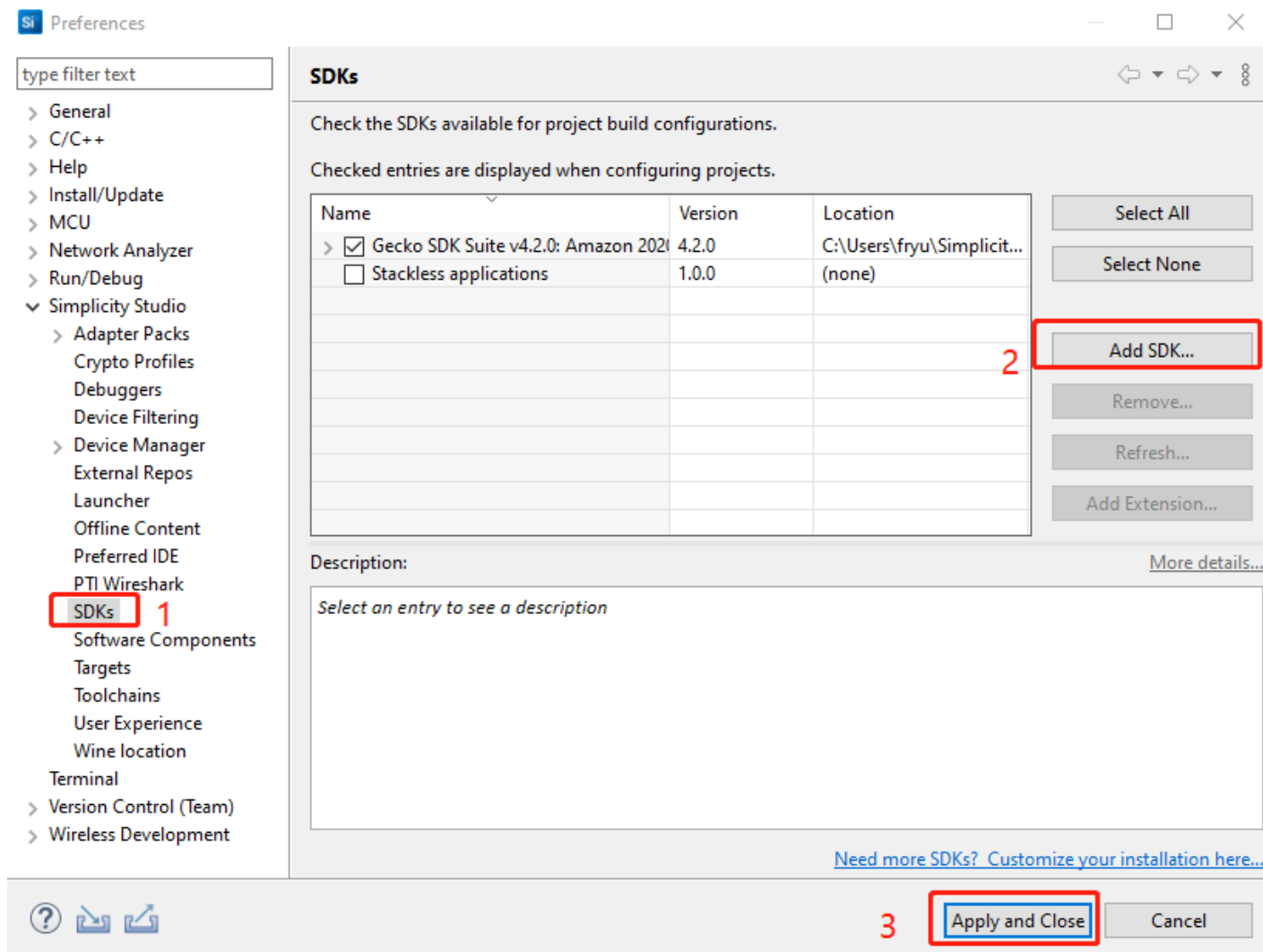


SSv5——手动添加SDK

- 通过菜单Window/Preferences
- 或者直接点击Preferences按钮
- 选择Simplicity Studio/SDKs



EF



SSv5——添加基础开发板

- 在左下角**My Products**处选择**BRD2601 Dev Kit Board**作为基础开发板

The screenshot shows the Simplicity Studio interface for the EFR32xG24 Dev Kit Board (BRD2601B Rev A02). The interface is divided into several sections:

- My Products (Left Sidebar):** A list of products where "EFR32xG24 Dev Kit Board (BRD2601B Rev A02)" is selected and highlighted with a red box.
- General Information (Top Right):** Contains a "Preferred SDK" section where "Gecko SDK Suite v4.2.0" is selected, highlighted with a red box. Below it is a "Manage SDKs" button.
- Recommended Quick Start Guides (Top Right):** A list of links to guides, including "AN1255: Transitioning from the v2.x to the v3.x Bluetooth SDK", "QSG168: Proprietary Flex SDK v3.x Quick Start Guide", "QSG169: Bluetooth Quick Start Guide for SDK 3.x and Higher", and "QSG175: Silicon Labs' Direction Finding Solution Quick-Start Guide". A red box highlights this entire section, and an "All Quick Start Guides" button is at the bottom.
- Board (Bottom Left):** A section with an image of the board and the text "EFR32xG24 Dev Kit Board (BRD2601B Rev A02)", highlighted with a red box. Below it is a "View Documents" button.
- Target Part (Bottom Right):** A section with an image of the chip and the text "EFR32MG24B310F1536IM48", highlighted with a red box. Below it is a "View Documents" button.

SSv5——查看Matter示例工程

■ 切换到EXAMPLE PROJECTS & DEMOS页面

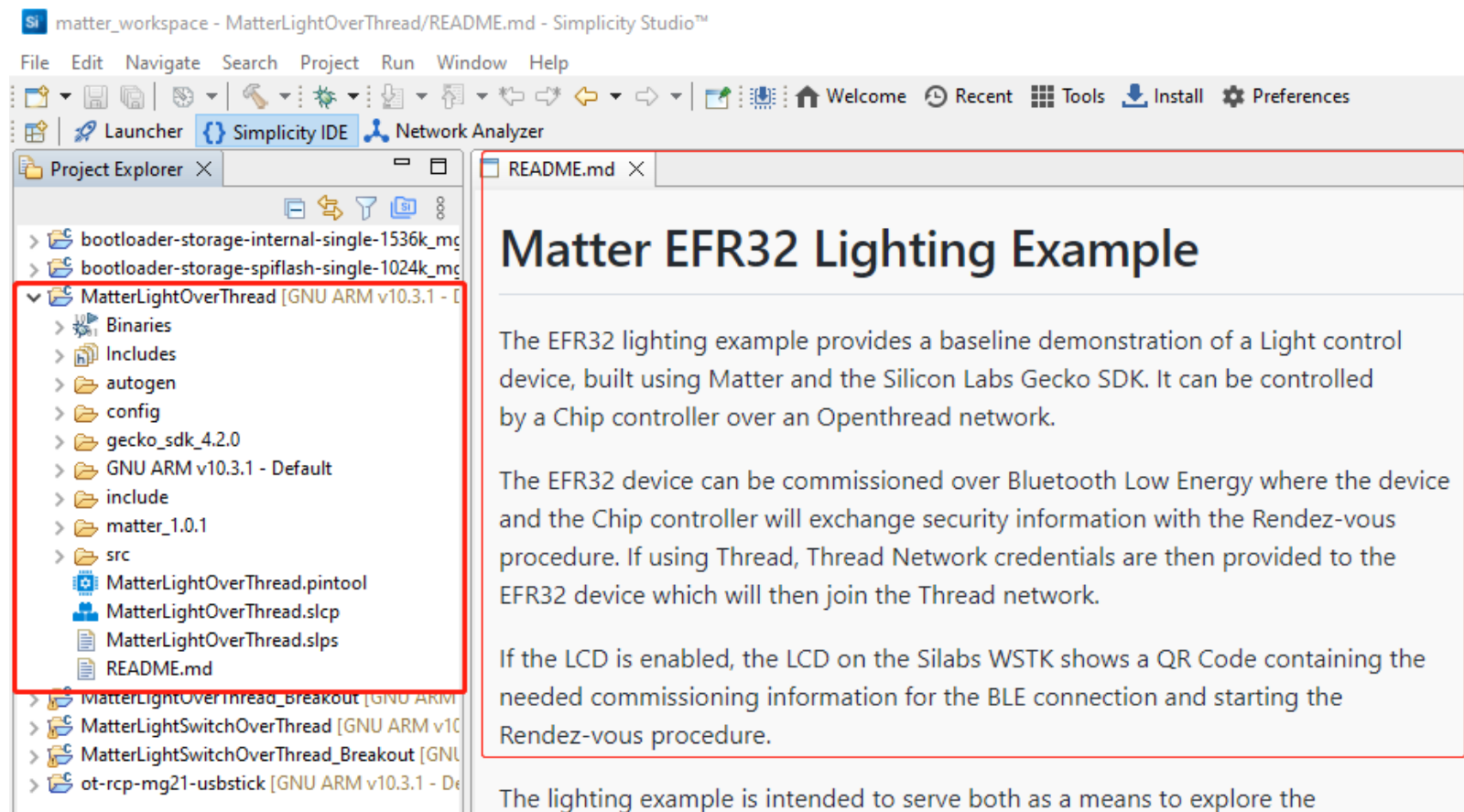
The screenshot shows the Simplicity Studio interface with the 'EXAMPLE PROJECTS & DEMOS' page selected for the 'EFR32xG24 Dev Kit Board (BRD2601B Rev A02)'. The left sidebar shows the 'My Products' list with 'EFR32xG24 Dev Kit Board (BRD2601B Rev A02)' selected. The main content area displays a grid of project cards, each with a title, description, and a 'CREATE' or 'RUN' button. The projects are categorized under 'Matter' and 'OpenThread'.

9 resources found

- Matter - Light Switch over Thread**
This project builds a Matter Light Switch that can be developed inside Simplicity Studio
[View Project Documentation](#) [CREATE](#)
- Matter - Light over Thread**
This project builds a Matter Light that can be developed inside Simplicity Studio
[View Project Documentation](#) [CREATE](#)
- Matter - Light switch over Thread**
This is a Matter Light switch Application over Thread for BRD2601B
[View Project Documentation](#) [RUN](#)
- Matter - Lighting over Thread**
This is a Matter Lighting Application over Thread for BRD2601B
[View Project Documentation](#) [RUN](#)
- Matter - Lock over Thread**
This is a Matter Lock Application over Thread for BRD2601B
[View Project Documentation](#) [RUN](#)
- Matter - Lock over Thread**
This project builds a Matter Lock that can be developed inside Simplicity Studio
[View Project Documentation](#) [CREATE](#)
- Matter - OpenThread Border Router Documentation**
This is an Empty Project that is intended to guide the user to our main documentation for the OpenThread Border Router
[View Project Documentation](#) [CREATE](#)
- Matter - Window Cover over Thread**
This project builds a Matter Window Cover that can be developed inside Simplicity Studio
[View Project Documentation](#) [CREATE](#)
- Matter - Window over Thread**
This is a Matter Window Application over Thread for BRD2601B
[View Project Documentation](#) [RUN](#)

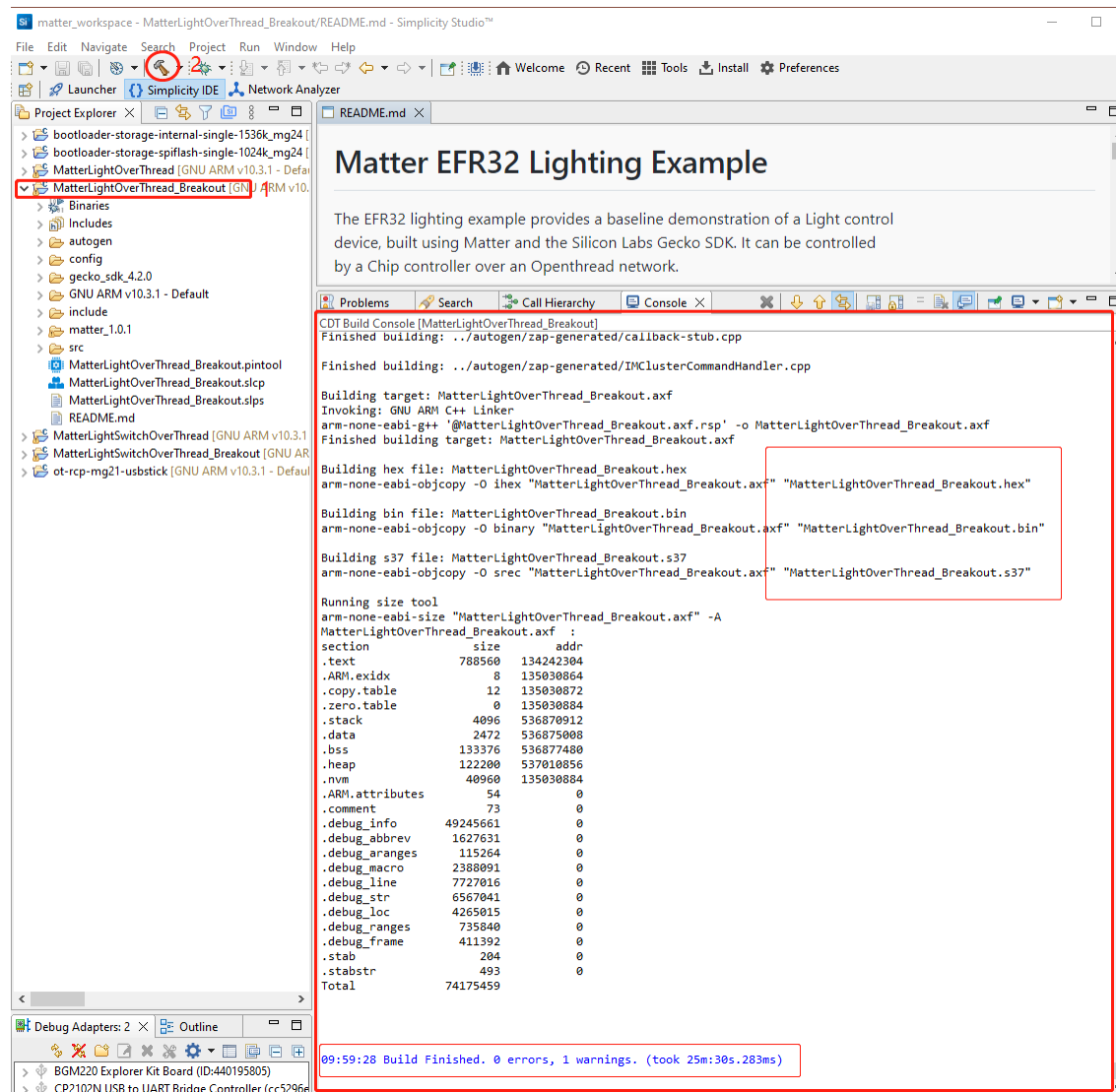
SSv5——创建Matter GSDK工程

■ 创建Matter – Light over Thread工程



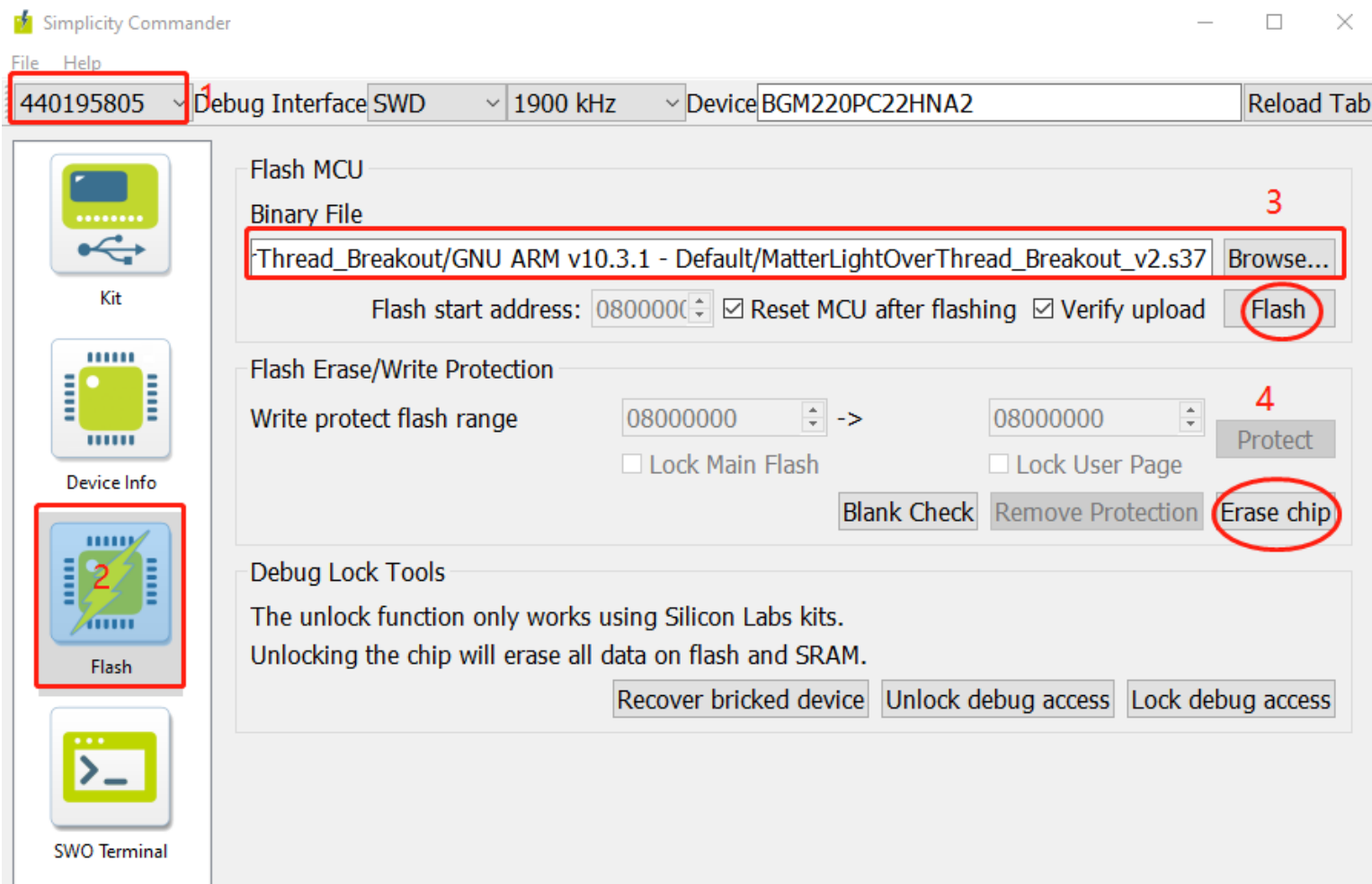
SSv5——编译Matter GSDK工程

- 选择工程目录
- 点击编译按钮
- 编译完成后会生成s37/hex/bin三种格式的固件
- 最下面显示了本次编译的耗时时间



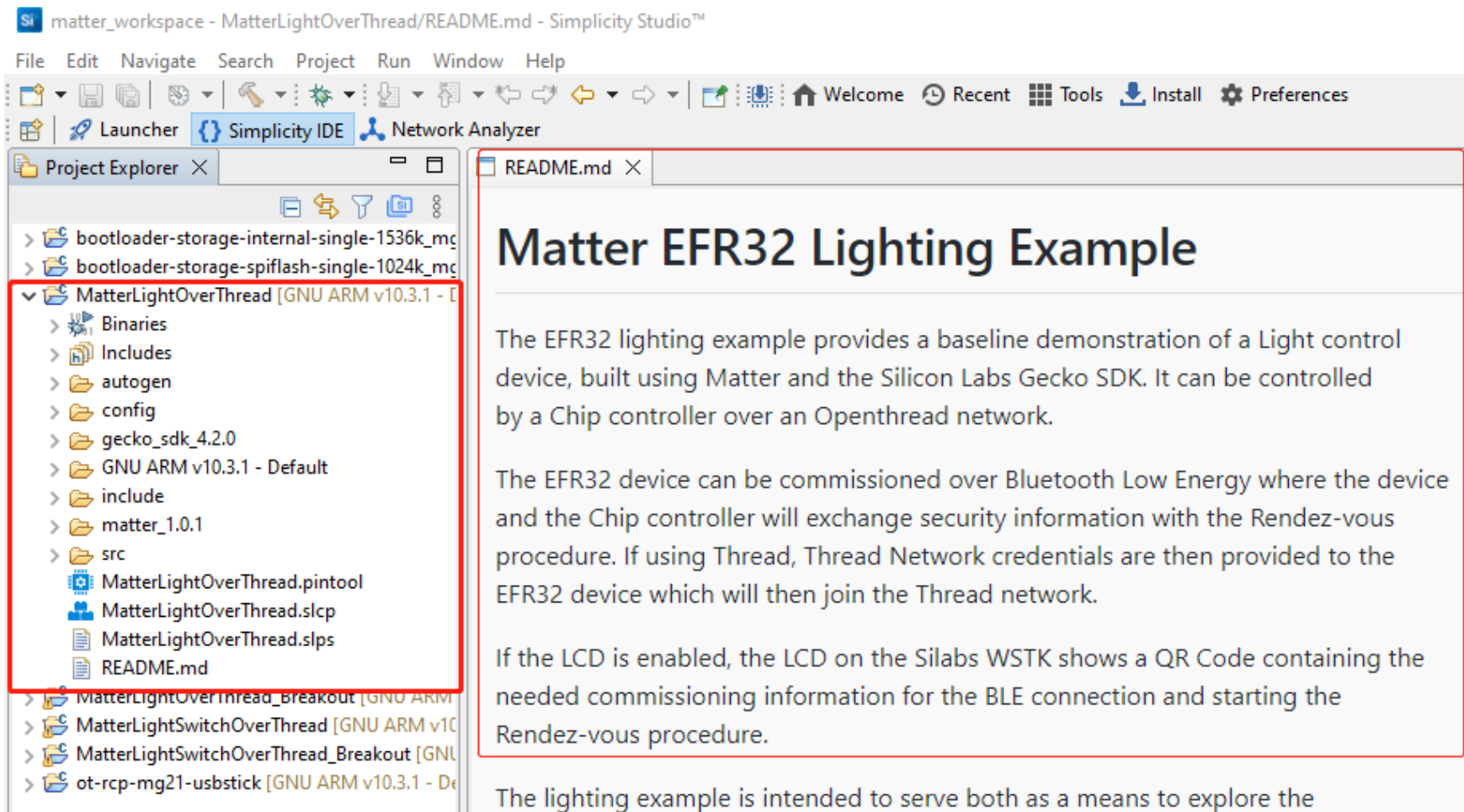
SSv5——烧录固件

- 打开**Simplicity Commander**工具
- 选择对应的**J-Link**工具
- 选择**Flash**页面
- 选择要烧录的固件
- 点击“**Flash**”进行烧录
- 如需擦除芯片，可点击“**Erase chip**”



GSDK工程介绍

- 工程目录文件
- **GSDK**目录文件
- **Matter**目录文件



The screenshot displays the Simplicity Studio IDE interface. The top menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. Below the menu is a toolbar with icons for various functions. The main workspace is divided into two panes. The left pane, titled 'Project Explorer', shows a hierarchical view of the project structure. The 'MatterLightOverThread' project is expanded, revealing subfolders like Binaries, Includes, autogen, config, gecko_sdk_4.2.0, GNU ARM v10.3.1 - Default, include, matter_1.0.1, and src. It also lists files: MatterLightOverThread.pintool, MatterLightOverThread.slcp, MatterLightOverThread.slps, and README.md. The right pane, titled 'README.md', displays the content of the file. The title is 'Matter EFR32 Lighting Example'. The text describes the example as a baseline demonstration of a Light control device built using Matter and the Silicon Labs Gecko SDK, controlled by a Chip controller over an OpenThread network. It also mentions commissioning over Bluetooth Low Energy and the use of the LCD on the Silabs WSTK for commissioning information.

Matter EFR32 Lighting Example

The EFR32 lighting example provides a baseline demonstration of a Light control device, built using Matter and the Silicon Labs Gecko SDK. It can be controlled by a Chip controller over an OpenThread network.

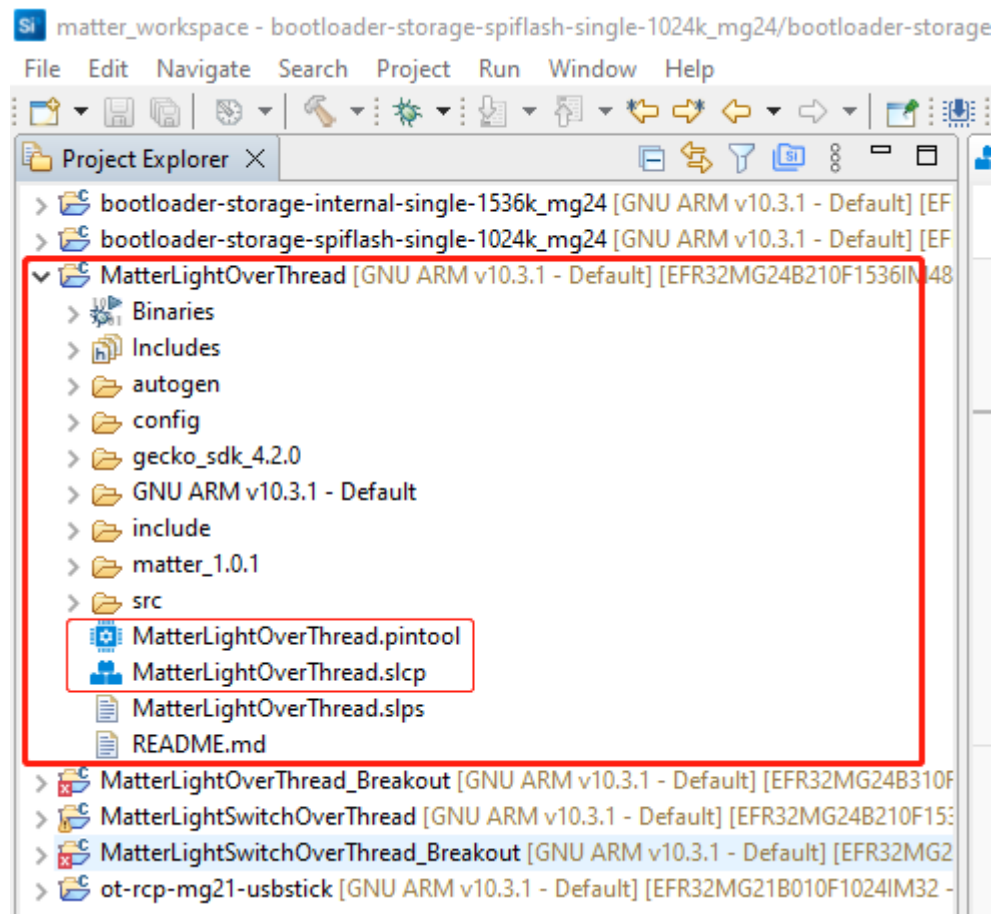
The EFR32 device can be commissioned over Bluetooth Low Energy where the device and the Chip controller will exchange security information with the Rendez-vous procedure. If using Thread, Thread Network credentials are then provided to the EFR32 device which will then join the Thread network.

If the LCD is enabled, the LCD on the Silabs WSTK shows a QR Code containing the needed commissioning information for the BLE connection and starting the Rendez-vous procedure.

The lighting example is intended to serve both as a means to explore the

GSDK工程——工程目录文件介绍

- **Binaries:** 编译后产生的虚拟目录。列出了编译产生的固件
- **Includes:** 列出了该工程用到的头文件
- **autogen目录:** 通过配置工具自动生成的代码，不可修改
- **config目录:** 工程配置文件，可通过此目录中的文件修改软件配置
- **gecko_sdk_4.2.0目录:** GSDK目录。若需要修改到里面的内容，建议创建副本到工程目录进行修改
- **GNU ARM v10.3.1 – Default目录:** 编译后产生的目录。用于存放编译产生的临时文件和固件
- **include目录:** 工程头文件
- **matter_1.0.1目录:** Matter相关代码。若需要修改到里面的内容，建议创建副本到工程目录进行修改
- **src目录:** 工程相关应用代码目录
- **slcp文件:** 工程配置文件。通过此工具添加软件功能组件并进行配置
- **pintool文件:** 芯片引脚配置工具。可通过此工具配置芯片引脚功能，并能够直观查看每个引脚的配置
- **README.md:** 该工程说明文档



GSDK工程——GSDK目录文件介绍

■ hardware

- board: 硬件板级初始化框架

■ platform

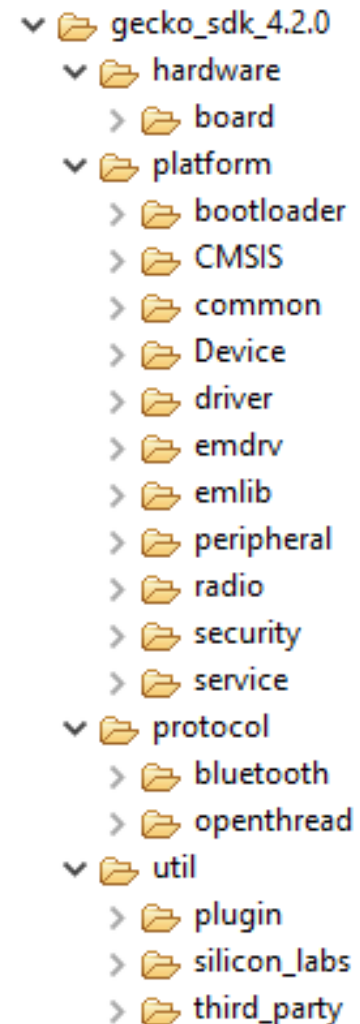
- bootloader: 启动引导程序
- Device: 芯片启动代码
- driver: 驱动程序。比如安装按键驱动、LED驱动等。
- emdrv: 芯片外设驱动框架
- emlib: 芯片外设驱动实现
- radio: 无线控制
- security: 加密算法
- service: 独立于设备和线程安全的服务组件，并具有硬件抽象层。

■ protocol

- bluetooth: 蓝牙协议栈
- openthread: Thread协议栈

■ util

- plugin: 插件。比如安全管理
- third_party: 第三方开源库。比如: freertos/mbdrtls/openthread/segger



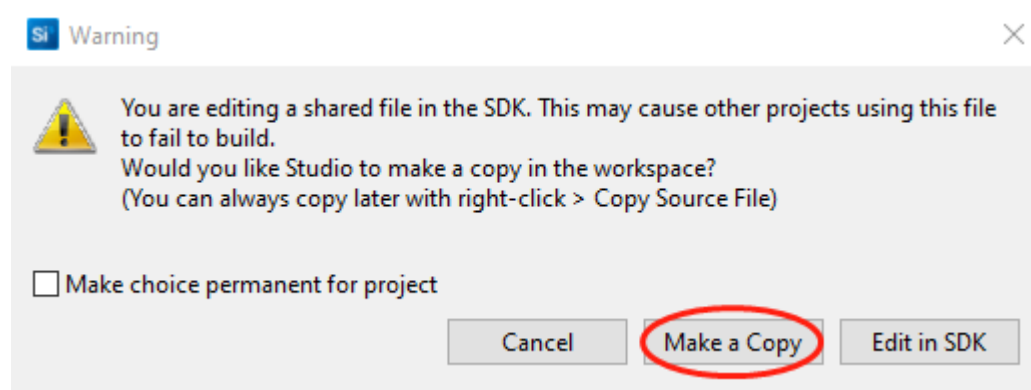
GSDK工程——Matter目录文件介绍

- **example** : 与平台相关的例程代码
- **src**
 - app: 应用层。ZCL(Zigbee Cluster Library)
 - ble:: BLE相关协议
 - controller: 控制器配网流程相关API。
 - crypto: 加密库
 - inet: TCP/UDP相关网络层接口
 - lib: 核心支持库
 - message: 消息分发管理接口
 - platform: 可移植性平台适配层
 - setup_payload: 二维码编解码相关库
 - system: 系统控制公共接口
 - Transport: 数据传输接口
- **zzz_generated**: zap生成的模板代码

```
▼ matter_1.0.1
  ▼ examples
    > platform
    > providers
  ▼ src
    > access
    > app
    > ble
    > controller
    > credentials
    > crypto
    > inet
    > lib
    > messaging
    > platform
    > protocols
    > setup_payload
    > system
    > transport
  ▼ zzz_generated
    > app-common
```

GSDK工程——修改代码

- 修改用户代码
 - 直接修改即可
- 修改**GSDK**代码
 - 修改时会弹出如下图所示的一个对话框
 - 这里我们建议不要直接在**SDK**中修改，
 - 点击“**Make a Copy**”，就会创建一个副本文件到工程目录里进行修改
 - 这样在修改时不会影响到原始**GSDK**目录中的代码



GSDK工程——Matter代码main.cpp

- 这里我们简单查看一下Matter代码main.cpp

```
int main(void)
{
    // Initialize Silicon Labs device
    sl_system_init();
    // Initialize the application.
    app_init();
    // Start the kernel.
    sl_system_kernel_start();

    chip::Platform::MemoryShutdown();
    EFR32_LOG("vTaskStartScheduler() failed");
    appError(CHIP_ERROR_INTERNAL);
}

void sl_button_on_change(const sl_button_t * handle)
{
    AppTask::GetAppTask().ButtonEventHandler(handle,
    sl_button_get_state(handle));
}
```

```
void app_init(void)
{
    init_efrPlatform();
    EFR32MatterConfig::InitMatter(BLE_DEV_NAME)

    gExampleDeviceInfoProvider.SetStorageDelegate(&Server::GetInstance().GetPersistentStorage());
    chip::DeviceLayer::SetDeviceInfoProvider(&gExampleDeviceInfoProvider);

    chip::DeviceLayer::PlatformMgr().LockChipStack();
    // Initialize device attestation config
    SetDeviceAttestationCredentialsProvider(Examples:
    :GetExampleDACProvider());
    chip::DeviceLayer::PlatformMgr().UnlockChipStack(
    );

    EFR32_LOG("Starting App Task");
    AppTask::GetAppTask().StartAppTask();
}
```


软件组件应用实例

■ 实例功能定义

- 使用开发板上的PB02来做一个按键功能
 - 当操作按键时，打印出按键状态
 - 当操作按键时，控制LED灯亮灭
 - 当操作按键时，控制RGB灯亮灭
- 通过本实例，看看实现此功能需要手动添加的代码有多少？

■ 需要用到的组件

- 添加按键功能组件
- 添加GPIO LED功能组件
- 添加RGB PWM LED功能组件

软件组件应用实例一：安装按键组件

- 安装组件：`Platform\Driver\Button\Simple Button`

The screenshot displays the 'SOFTWARE COMPONENTS' tab of the MatterLightOverThread application. The interface includes a search bar at the top right with the text 'butt' entered, circled in red and labeled '1'. On the left sidebar, the 'Platform' section is expanded, and 'Simple Button' is selected, also circled in red and labeled '2'. In the main panel, the 'Simple Button' component is shown with an 'Install' button circled in red and labeled '3'. The component's description, quality (PRODUCTION), and dependencies (simple_button requires 0 components) are visible.

MatterLightOverThread.slc

MatterLightOverThread OVERVIEW SOFTWARE COMPONENTS CONFIGURATION TOOLS

Filter components by ☐ Configurable ☐ Installed ☐ Installed by you ☐ SDK Extensions ☐ Quality butt 1

▼ Application

▼ Utility

Button Press

▼ Platform

▼ Board

▼ Starter Kit

BRD2703A rev A02

▼ Driver

▼ Button

Generic Button API

Simple Button 2

Simple Button Configuration

▼ Zigbee

▼ Utility

Application Framework Common

Simple Button

Install 3

Description

The Simple Button Driver provides an implementation of the Generic Button API. This driver supports both active high and low buttons and configurable debouncing.

This component is instantiable, meaning that several named instances can be created. For each instance a configuration is generated along with an initialization struct set according to this configuration. These instance defines and variables are available in a generated header file, `sl_simple_button_instances.h`.

If the Services->Runtime->System Setup->System Init component is included in a project, the driver instances will be initialized automatically, using the instance configurations, during the `sl_system_init()` call in `main.c`.

Selecting this component will also include the Simple Button Core component, which is the implementation of the Simple Button driver itself.

Quality

PRODUCTION

Dependencies

simple_button requires 0 components

软件组件应用实例一：安装按键组件

- 点击安装，出现下图框，默认命名为btn0
- 点击Done完成安装

Close X

Create A Component Instance

This component allows multiple instances. Create a name for this instance in the field below. The name will be used to construct #defines in the source files of the instance.

INSTANCE NAME

btn0

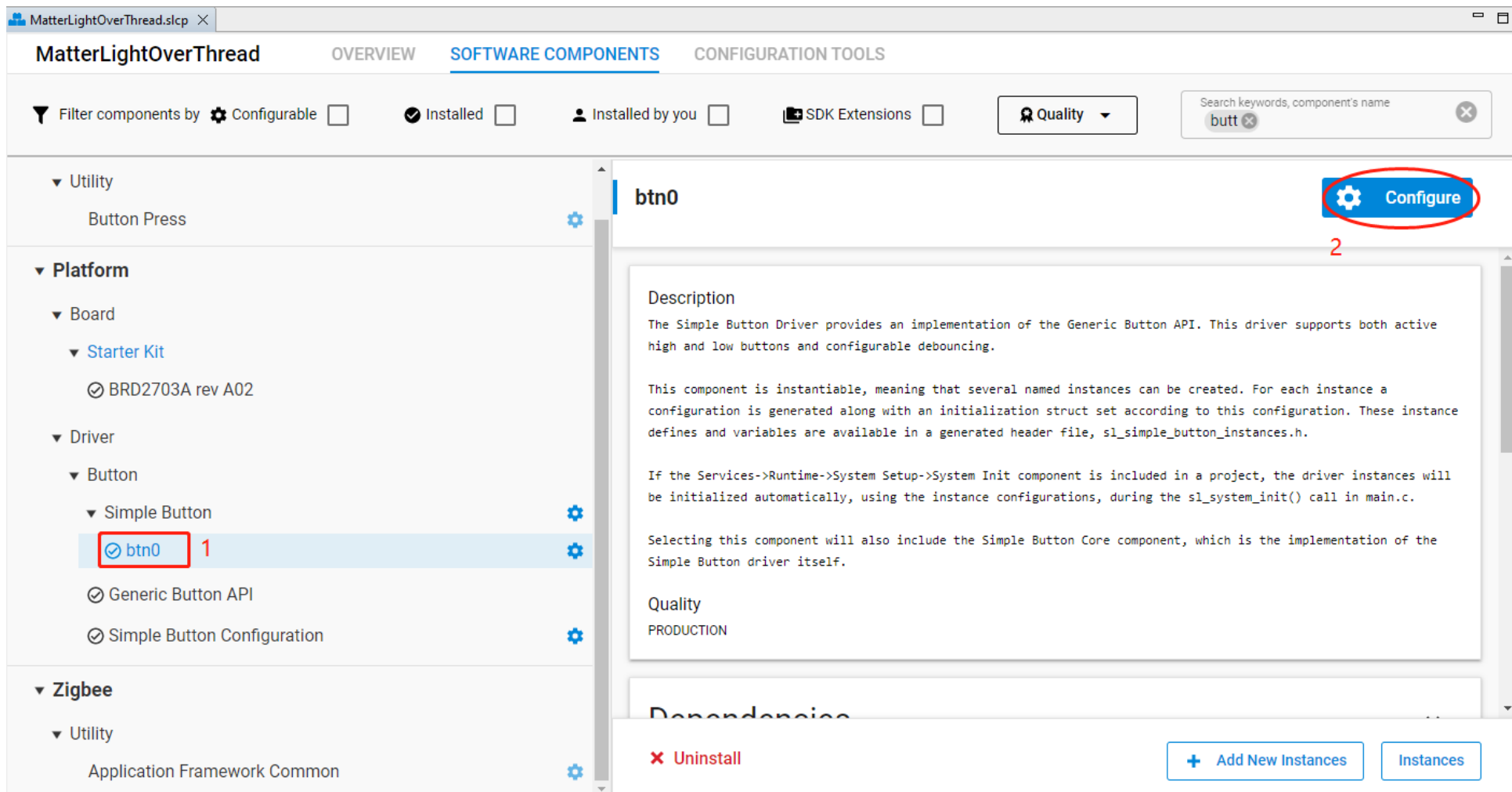
Names must adhere to both C and POSIX filename rules

▼ Recommended Instance Names

Cancel Done

软件组件应用实例一：配置按键

- 安装完成后可以看到btn0组件，可以点击Configure进行配置



软件组件应用实例一：配置按键

- 配置按键工作模式
 - Interrupt
 - Poll and Debounce
 - Poll
- 配置GPIO
 - 根据硬件原理图配置
- 自定义引脚名称（可选）

Simple Button (btn0)

Documentation Pin Tool View Source

General

SL_SIMPLE_BUTTON_BTN0_MODE

Interrupt

配置按键工作模式

SL_SIMPLE_BUTTON_BTN0

Selected Module

PB02 (FUNC_KEY)

配置GPIO

Port Pin

Custom Pin Name

FUNC_KEY

定义名称（可选）

软件组件应用实例一：配置按键

- 打开pintool文件
- 可清楚的看到刚刚配置的按键

MatterLightOverThread.sclpSimple Button (btn0)MatterLightOverThread.pintool

DefaultMode Port I/O: PORTIO

(top view)

48-pin QFN

6x6

VSS

1/PC00

2/PC01

3/PC02

4/PC03

5/PC04

6/PC05

7/PC06

8/PC07

9/PC08

10/PC09

11/HFXTAL1

12/HFXTAL2

13/RES10

14/RES11

15/RES12

16/RES13

17/RES14

18/RES15

19/RES16

20/RES17

21/RES18

22/RES19

23/RES20

24/RES21

25/RES22

26/RES23

27/RES24

28/RES25

29/RES26

30/RES27

31/RES28

32/RES29

33/RES30

34/RES31

35/RES32

36/RES33

37/RES34

38/RES35

39/RES36

40/RES37

41/RES38

42/RES39

43/RES40

44/RES41

45/RES42

46/RES43

47/RES44

48/RES45

PTI_DOUT

PTI_DFRAME

35/EUSART1_CTS

34/EUSART1_RTS

33/GPIO

32/EUSART1_RX

31/EUSART1_TX

30/GPIO

Configure

Search:

Pins	Functions	Peripherals	Custom Pin Name	Software Component
26	PA00			
27	PA01			
28	PA02			
29	PA03			
30	PA04	GPIO mode		Simple LED (led0) : SL_SIMPLE_LED_LED0
31	PA05	EUSART1_TX		UARTDRV EUSART (vcom) : SL_UARTDRV_EUSART_VCOM
32	PA06	EUSART1_RX		UARTDRV EUSART (vcom) : SL_UARTDRV_EUSART_VCOM
33	PA07	GPIO mode		Simple LED (led1) : SL_SIMPLE_LED_LED1
34	PA08	EUSART1_RTS		UARTDRV EUSART (vcom) : SL_UARTDRV_EUSART_VCOM
35	PA09	EUSART1_CTS		UARTDRV EUSART (vcom) : SL_UARTDRV_EUSART_VCOM
23	PB00			
22	PB01			
21	PB02	GPIO mode	FUNC_KEY	Simple Button (btn0) : SL_SIMPLE_BUTTON_BTN0
20	PB03			
19	PB04			
18	PB05			
1	PC00			
2	PC01			
3	PC02			
4	PC03			
5	PC04			
6	PC05			
7	PC06	PTI_DOUT		RAIL Utility, PTI : SL_RAIL_UTIL_PTI
8	PC07	PTI_DFRAME		RAIL Utility, PTI : SL_RAIL_UTIL_PTI
9	PC08			
10	PC09			
48	PD00			
47	PD01			
46	PD02			
45	PD03			
44	PD04			
43	PD05			

软件组件应用实例一：添加按键相关代码

- autogen目录下自动生成以下文件
 - sl_simple_button_instances.c
 - sl_simple_button_instances.h
- config目录下自动生成以下文件
 - sl_simple_button_btn0_config.h
 - sl_simple_button_config.h
- 添加用户代码打印按键状态

//重写callback函数:

```
#include "sl_simple_button_btn0_config.h"
```

```
void sl_button_on_change(const sl_button_t *handle)
```

```
{
```

```
    sl_simple_button_context_t* ctx = handle->context;
```

```
    // 打印按键状态
```

```
    printf("Port:%d PIN:%d state:%d\r\n", ctx->port, ctx->pin, ctx->state);
```

```
}
```

软件组件应用实例二：安装LED组件

- 安装组件：Platform\Driver\LED\Simple LED

The screenshot displays the 'MatterLightOverThread' software interface with the 'SOFTWARE COMPONENTS' tab selected. The left sidebar shows a tree view under 'Platform' > 'Driver' > 'LED', where 'Simple LED' is highlighted with a red box and a red '2'. The main panel shows the 'Simple LED' component details, including a description and a quality level of 'PRODUCTION'. A red circle highlights the 'Install' button in the top right corner of the component details panel, with a red '3' next to it. A search bar at the top right contains the text 'led' and a red '1' next to it.

MatterLightOverThread.slcip ×

MatterLightOverThread OVERVIEW **SOFTWARE COMPONENTS** CONFIGURATION TOOLS

Filter components by ☒ Configurable ☒ Installed ☐ Installed by you ☐ SDK Extensions ☐ Quality 1

▼ Debug
Debug Logging Disabled

▼ Platform

▼ Board

▼ Starter Kit

BRD2703A rev A02

▼ Driver

▼ LED

Generic LED API

Simple LED 2

Simple RGB PWM LED

Simple LED 3

Description

The Simple LED Driver provides an implementation of the Generic LED API. This driver supports controlling GPIO based on/off type LEDs.

This component is instantiable, meaning that several named instances can be created. For each instance a configuration is generated along with an initialization struct set according to this configuration. These instance defines and variables are available in a generated header file, `sl_simple_led_instances.h`.

If the Services->Runtime->System Setup->System Init component is included in a project, the driver instances will be initialized automatically, using the instance configurations, during the `sl_system_init()` call in `main.c`.

Selecting this component will also include the Simple LED Core component, which is the implementation of the Simple LED driver itself.

Quality

PRODUCTION

软件组件应用实例二：配置LED组件

- 配置LED工作电平
 - Active low
 - Active high
- 配置GPIO
 - 根据硬件原理图配置
- 自定义引脚名称（可选）

MatterLightOverThread_matter.slcp Simple LED (led0) X

Simple LED (led0)

>>

Simple LED configuration

SL_SIMPLE_LED_LED0_POLARITY

Active low ▼

SL_SIMPLE_LED_LED0

Selected Module

PD02 (LED0) ▼

Port Pin

Custom Pin Name

LED0

软件组件应用实例二：添加LED相关代码

- autogen目录下自动生成以下文件
 - sl_simple_led_instances.c
 - sl_simple_led_instances.h
- config目录下自动生成以下文件
 - sl_simple_led_led0_config.h

- 添加用户代码打印按键状态

```
// 在按键的callback函数中改变LED灯的状态
#include "sl_simple_button_btn0_config.h"
#include "sl_simple_led_instances.h"

void sl_button_on_change(const sl_button_t *handle)
{
    sl_simple_button_context_t* ctx = handle->context;
    // 打印按键状态
    printf("Port:%d PIN:%d state:%d\r\n", ctx->port, ctx->pin, ctx->state);
    // 改变LED灯的状态
    sl_led_toggle(SL_SIMPLE_LED_INSTANCE(0));
}
```

软件组件应用实例三：安装RGB组件

- 安装组件：Platform\Driver\LED\Simple RGB PWM LED

The screenshot displays the 'MatterLightOverThread' software interface. The 'SOFTWARE COMPONENTS' tab is active. In the top right, a search bar contains the text 'rgb', which is circled in red. On the left sidebar, the 'Platform' > 'Driver' > 'LED' path is expanded, and 'Simple RGB PWM LED' is highlighted with a red box. On the right, the details for 'Simple RGB PWM LED' are shown, including a description and an 'Install' button circled in red.

MatterLightOverThread OVERVIEW SOFTWARE COMPONENTS CONFIGURATION TOOLS

Filter components by ☒ Configurable ☐ Installed ☐ Installed by you ☐ SDK Extensions ☐ Quality rgb

Bluetooth

- GATT
 - RGB LED GATT Service

Platform

- Driver
 - LED
 - Simple RGB PWM LED**

Simple RGB PWM LED

Description

The Simple RGB PWM LED Driver provides an implementation of the Generic LED API. This driver supports controlling Red/Green/Blue LED sets and uses Pulse Width Modulation (PWM) to set the intensity of each LED.

This component is instantiable, meaning that several named instances can be created. For each instance a configuration is generated along with an initialization struct set according to this configuration. These instances, defines, and variables are available in a generated header file, `sl_simple_rgb_pwm_led_instances.h`.

If the Services->Runtime->System Setup->System Init component is included in a project, the driver instances will be initialized automatically, using the instance configurations, during the `sl_system_init()` call in `main.c`.

Selecting this component will also include the Simple LED Core component, which is the implementation of the Simple RGB PWM LED driver itself.

Quality

软件组件应用实例三：配置RGB组件

- 配置LED工作电平
 - Active low
 - Active high
- 配置PWM
 - 频率
 - 定时器
 - R/G/B各通道
- 配置GPIO
 - 根据硬件原理图配置
- 自定义引脚名称（可选）

bt_soc_empty_2.sldp Simple RGB PWM LED (inst) X

Simple RGB PWM LED (inst) Documentation Pin Tool </> View Source

>>

Simple RGB PWM LED Configuration

PWM frequency [Hz]	PWM resolution	Red LED Polarity	Green LED Polarity	Blue LED Polarity
10000	256	Active low	Active low	Active low

SL_SIMPLE_RGB_PWM_LED_INST

Selected Module	CC0	CC0 name	CC1	CC1 name
TIMER0	PB00	BLUE	PA04	GREEN
CC2	CC2 name	CDTI0	CDTI1	CDTI2
PD02	RED	None	None	None

TIMER

Custom Peripheral Name	BLUE channel	GREEN channel	RED channel
	Channel 0	Channel 1	Channel 2

软件组件应用实例三：添加RGB相关代码

- autogen目录下自动生成以下文件
 - sl_simple_rgb_pwm_led_instances.c
 - sl_simple_rgb_pwm_led_instances.h
- config目录下自动生成以下文件
 - sl_simple_rgb_pwm_led_inst_config.h

- 添加用户代码打印按键状态

```
#include "sl_simple_button_btn0_config.h"
#include "sl_simple_rgb_pwm_led_instances.h"

void sl_button_on_change(const sl_button_t *handle)
{
    uint16_t red = 65535; // max red
    uint16_t green = 0; // no green
    uint16_t blue = 65535; // max blue
    sl_led_set_rgb_color(&sl_simple_rgb_pwm_led_inst, red, green, blue);
    sl_led_toggle(&sl_simple_rgb_pwm_led_inst);
}
```

参考资料

- 芯科科技**Simplicity-studio**集成开发环境: <https://www.silabs.com/developers/simplicity-studio>。
- 芯科科技开发者文档: <https://docs.silabs.com/>
- 芯科科技**Matter**方案介绍: <https://www.silabs.com/wireless/matter>
- 芯科科技**Matter**开发文档: <https://docs.silabs.com/matter/1.0.1/matter-start/>
- **Matter**协议规格书: <https://csa-iot.org/developer-resource/specifications-download-request/>
- **OpenThread**参考资料: <https://openthread.google.cn/>
- **SSv5**用户指南: <https://docs.silabs.com/simplicity-studio-5-users-guide/5.6.0/ss-5-users-guide-overview/>



—
谢谢!

Silicon Labs
官方网站



Silicon Labs
微信公众号



Silicon Labs
在线社区

