

Works With 2021

WIR-301: Exploring RTOS Options for Wireless IoT Projects

Overview

A real-time operating system (RTOS), or, to be more specific, a real-time kernel, can help to mitigate the complexity of a wireless IoT project. Although getting started with a kernel can seem like a challenge to developers who are accustomed to bare-metal system design, Simplicity Studio makes it easy to get up and running with a kernel-based project. This lab will walk through the steps both for leveraging Simplicity Studio's helpful kernel-based example projects and for adding a kernel to bare-metal examples.

Pre-requisites

Hardware: You'll need a Thunderboard Sense 2 to complete this lab. Combining a rich collection of sensors with a highly capable EFR32 wireless SoC, the Thunderboard is a great way to get started on your next IoT project. More information on the board is available from the [Silicon Labs Web site](#).



Development Tools: Simplicity Studio is a free, Eclipse-based IDE supporting Silicon Labs' SoCs and wireless modules. You'll need to download and install the latest version of the IDE in order to complete the steps outlined in this lab procedure. Installation requires a silabs.com account. There is no cost to set up an account, and you can create yours by [clicking here](#).

The Simplicity Studio installer covers both the IDE and the embedded software—wireless stacks, device drivers, and, *most importantly for this lab*, RTOSes—that Silicon Labs provides to support your development efforts. For the embedded software portion of the install, you'll be prompted to choose from **Install by connecting device(s)** or **Install by technology type**. Unless you are an advanced user with previous Silicon Labs experience, you should choose the former and then select the **Auto** option. This will automatically install the software supported by your Thunderboard, including the Bluetooth and Flex (proprietary wireless) SDKs needed by this lab. For more information on the Simplicity Studio installation process, you can consult the [IDE's documentation](#).

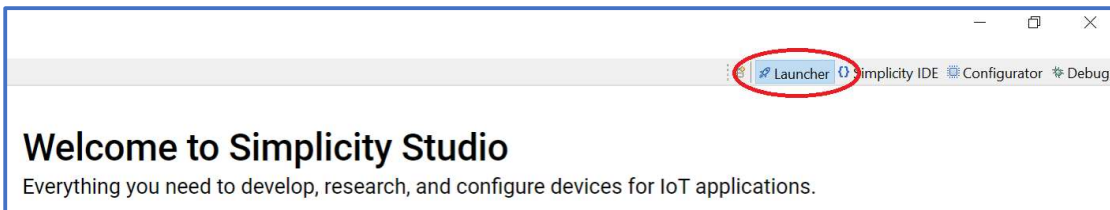
Mobile App: EFR Connect is Silicon Labs' easy-to-use mobile app for testing and debugging BLE projects. You'll use the app in the first part of this lab. You should take a few minutes to download the app from

either the [Apple Store](#) or [Google Play](#) before getting started. (Note that the screenshots in this guide were taken on iOS—the interface may be slightly different on Android.)

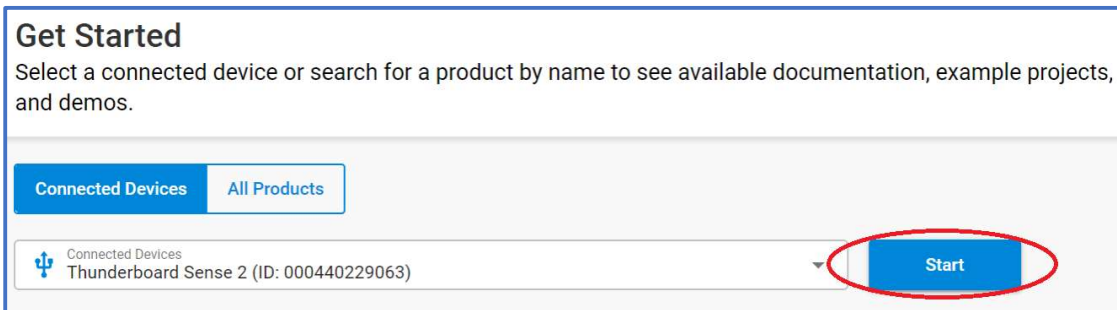
Lab Procedure

Part 1: A FreeRTOS-Based Bluetooth Example

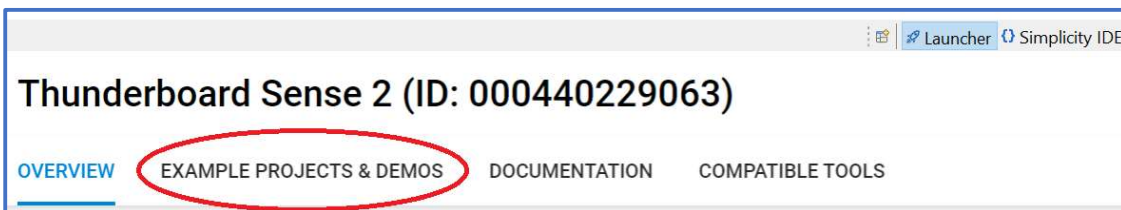
1. Establish a USB connection between your laptop and Thunderboard
2. Open Simplicity Studio and click on **Launcher** in the upper-right-hand corner



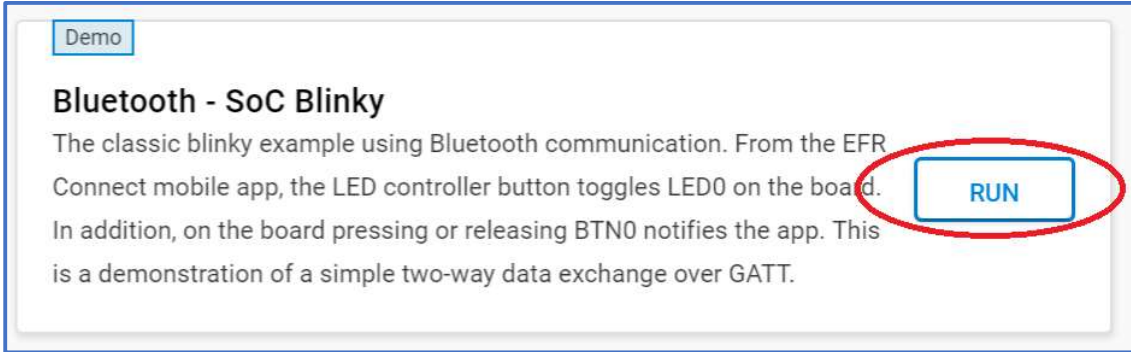
3. Beneath **Getting Started**, click **Connected Devices**, and, after confirming that your Thunderboard (**Thunderboard Sense 2**) appears in the **Connected Devices** dropdown, click **Start**



4. On the subsequent **Thunderboard Sense 2** page, open the **Example Projects & Demos** tab



- In the list of demos and projects appearing in the lower-right-hand corner of the page, scroll to the **Bluetooth SoC Blinky** demo (not the identically named example project, which lacks the blue **Demo** label) and click **Run** in order to flash your board with the bootloader that will be needed by the project you will create in the next steps



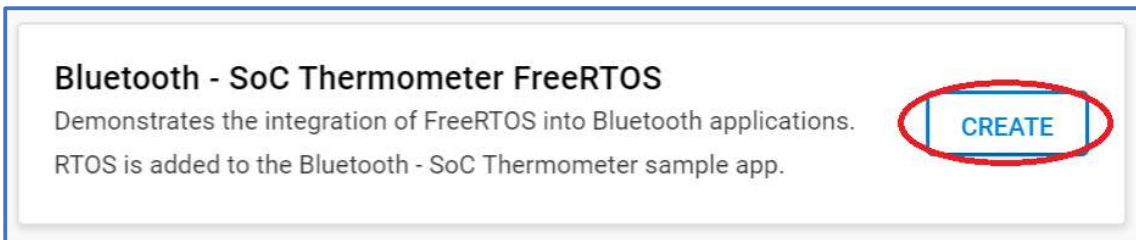
Demo

Bluetooth - SoC Blinky

The classic blinky example using Bluetooth communication. From the EFR Connect mobile app, the LED controller button toggles LED0 on the board. In addition, on the board pressing or releasing BTNO notifies the app. This is a demonstration of a simple two-way data exchange over GATT.

RUN

- Now scroll to the **Bluetooth – SoC Thermometer FreeRTOS** example project and click **Create**

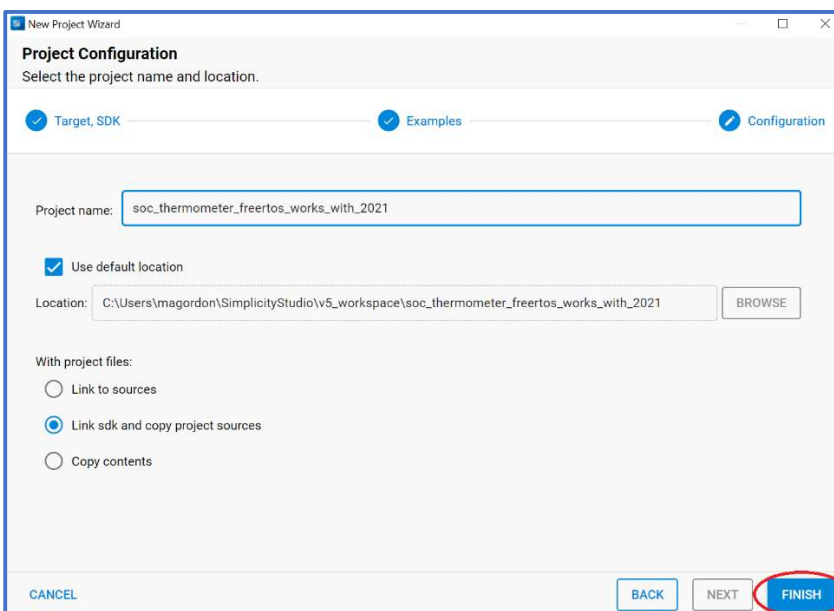


Bluetooth - SoC Thermometer FreeRTOS

Demonstrates the integration of FreeRTOS into Bluetooth applications. RTOS is added to the Bluetooth - SoC Thermometer sample app.

CREATE

- In the dialog that subsequently appears, leave the default options (updating the **Project** name according to your preference) and then click **Finish**



New Project Wizard

Project Configuration

Select the project name and location.

Target, SDK Examples Configuration

Project name: soc_thermometer_freertos_works_with_2021

Use default location

Location: C:\Users\magordon\SimlicityStudio\v5_workspace\soc_thermometer_freertos_works_with_2021 BROWSE

With project files:

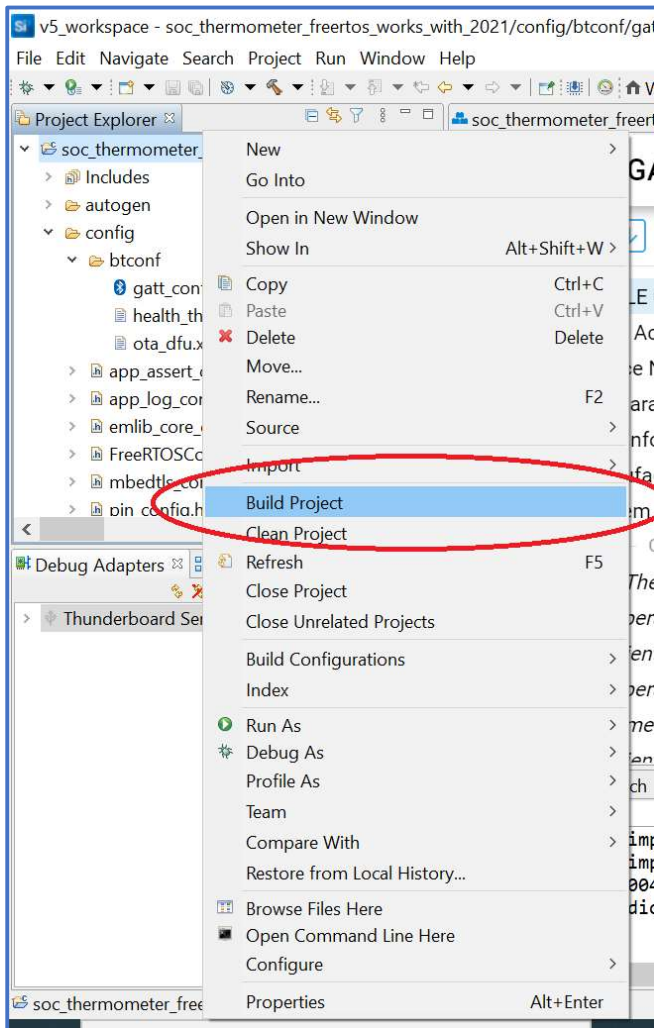
Link to sources

Link sdk and copy project sources

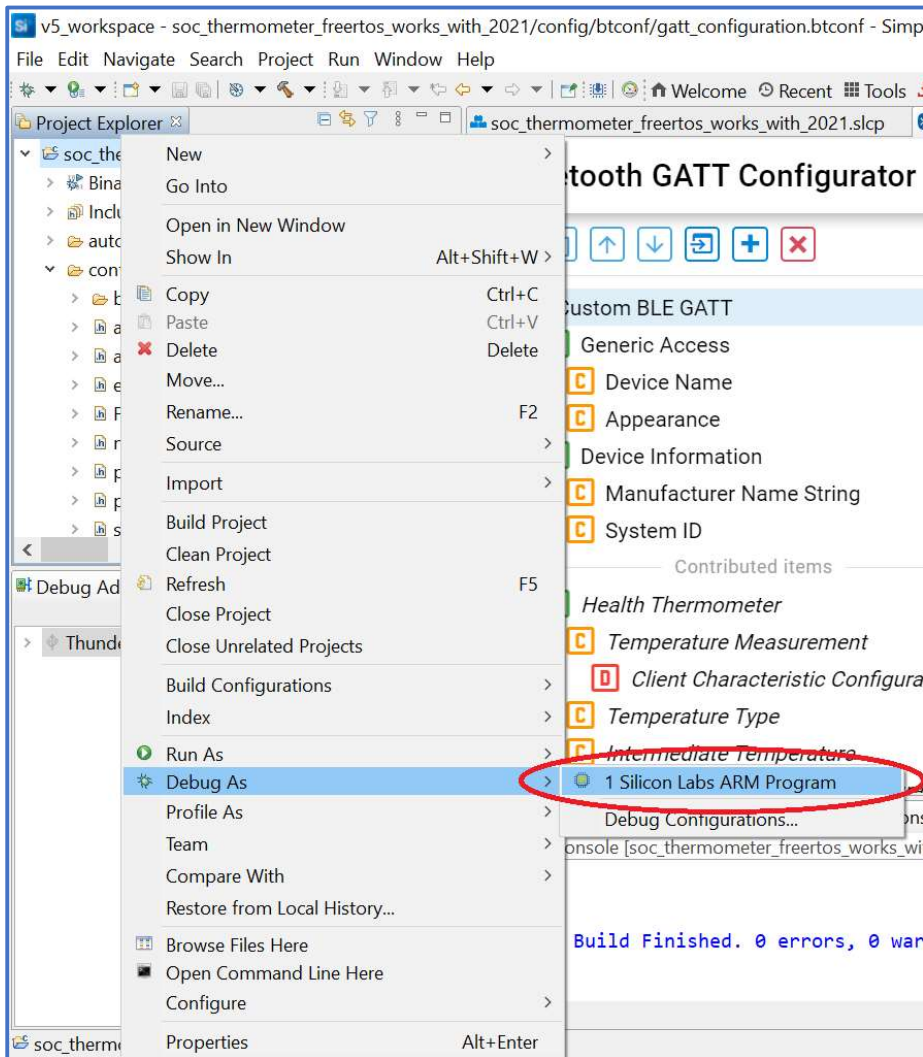
Copy contents

CANCEL BACK NEXT FINISH

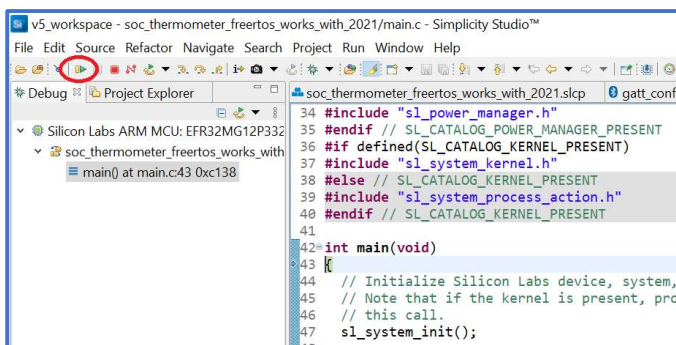
8. In the **Simplicity Studio IDE** perspective (which should have opened automatically when the project was created), right-click the project's name in the **Project Explorer** in the upper-left-hand corner of the screen and select **Build Project** from the menu that subsequently appears



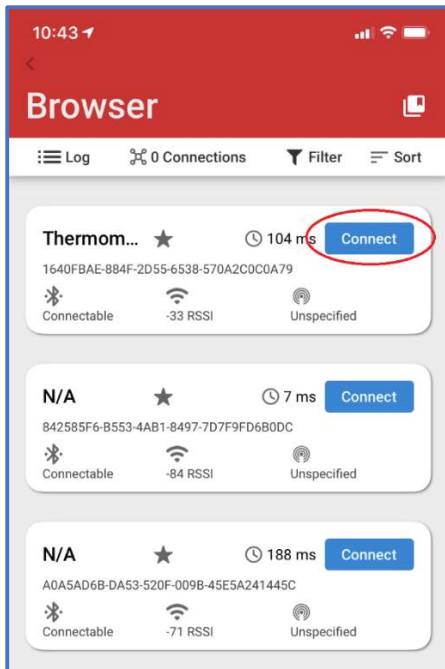
9. After the project has built successfully (with 0 errors and warnings reported in the **Console** at the bottom of the screen), right-click the project name again and select **Debug As > 1 Silicon Labs ARM Program**



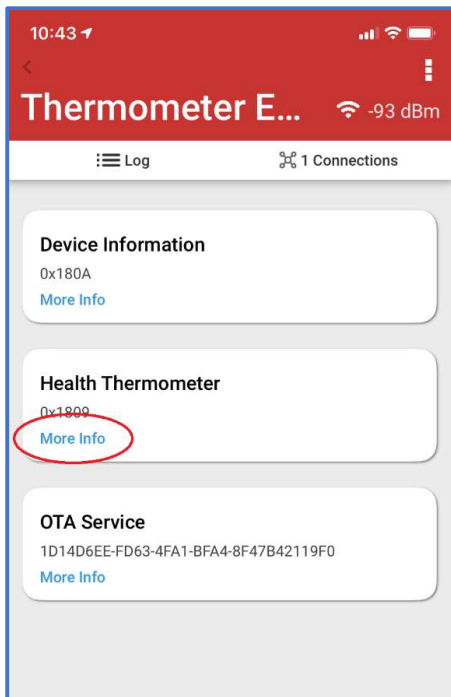
10. With program execution now halted at the first line of `main()`, click the **Resume** button to begin running the code



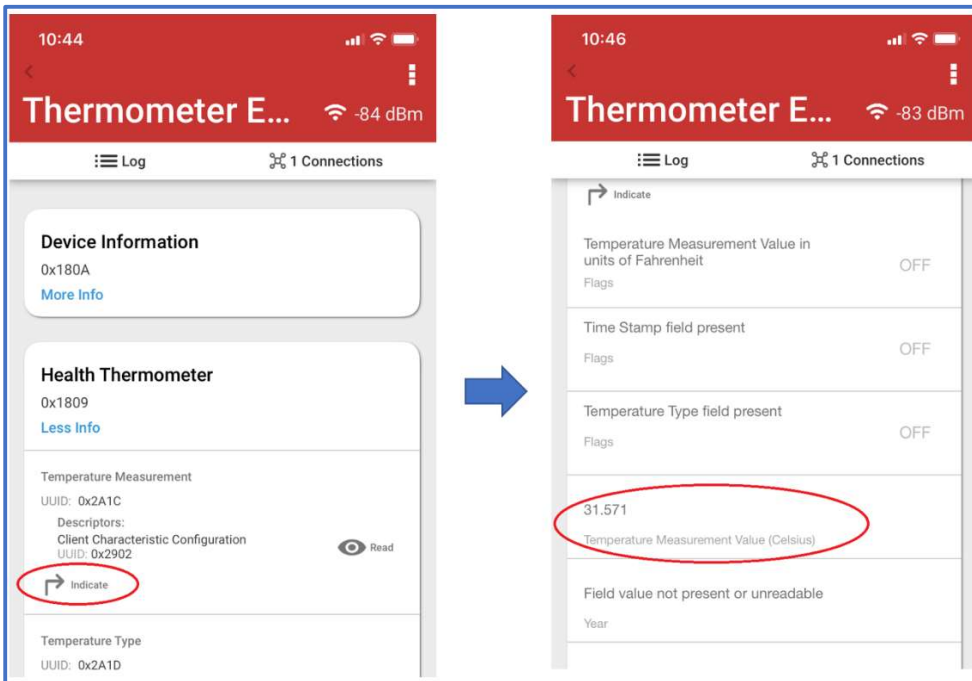
11. On your phone, open the EFR Connect app
12. In the Browser, find **Thermometer Example** and click **Connect**



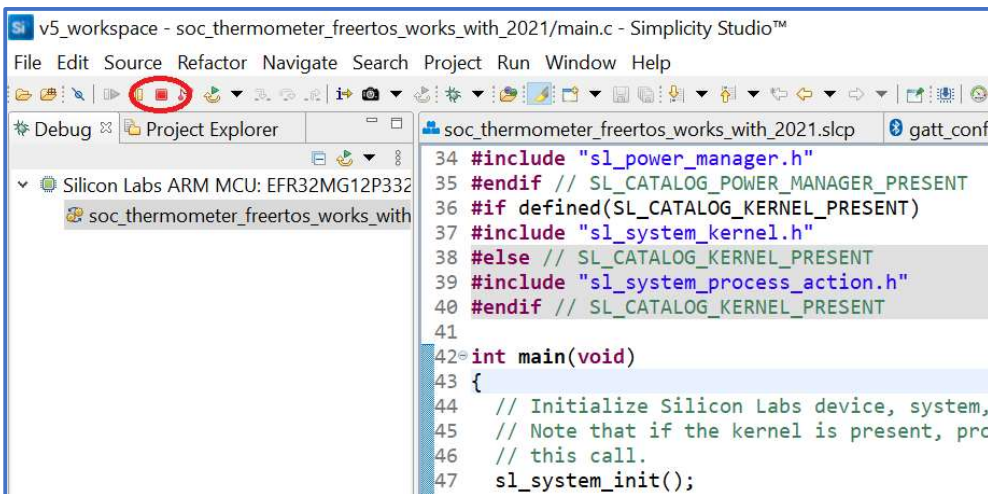
13. Under **Health Thermometer**, click **More Info**



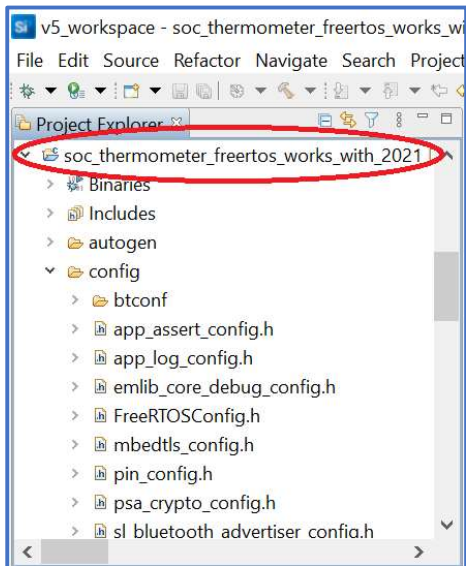
14. Under **Temperature Measurement**, click **Indicate** and confirm that updated temperature values are being periodically displayed in the **Temperature Measurement Value (Celsius)** field



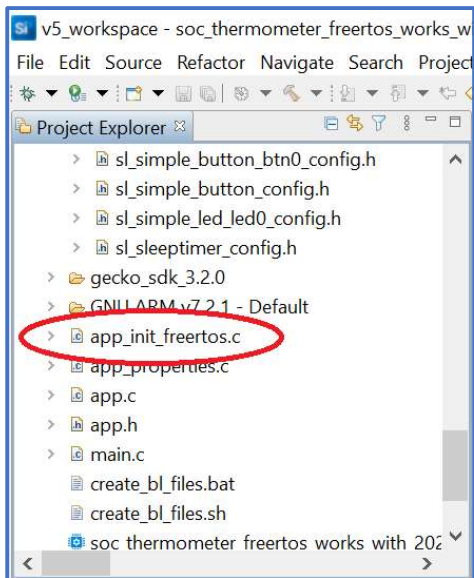
15. Return to Simplicity Studio and click the **Terminate** button to stop the debugger



16. In the **Simplicity IDE** perspective (which should open automatically when the debugger stops), click just below the arrow next to your project's name in **Project Explorer** to expand the list of source files comprising the project



17. Double-click *app_init_freertos.c*, which contains the FreeRTOS task that initializes the temperature sensor used by the example project, to open the file in the editor



18. Insert the below lines of code just after the include statements near the top of *app_init_freertos.c*

```
#include "sl_simple_led_instances.h"

#define APP_LED_TASK_PRIORITY      10u
#define APP_LED_TASK_STACK_SIZE   200

TaskHandle_t led_cb = NULL;

static void vTaskLED(void *pvParameters);
```

19. Insert the below function call (which will cause the example to create a new LED-blinking task using the parameters that you defined in the previous step) at the bottom of the existing function *app_init()* in *app_init_freertos.c*

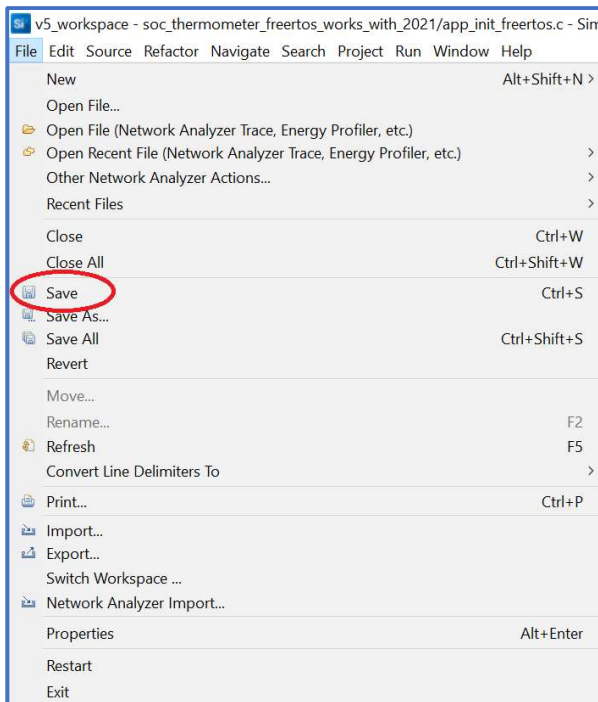
```
xTaskCreate(vTaskLED,
            "LED Task",
            APP_LED_TASK_STACK_SIZE,
            NULL,
            APP_LED_TASK_PRIORITY,
            &led_cb);
```

20. Add the below implementation of the LED-blinking task to the bottom of *app_init_freertos.c*

```
static void vTaskLED (void *pvParameters)
{
    (void)pvParameters;

    while (1) {
        vTaskDelay(1000);
        sl_led_toggle(&sl_led_led0);
    }
}
```

21. Save your changes to *app_init_freertos.c* by selecting **File>Save**



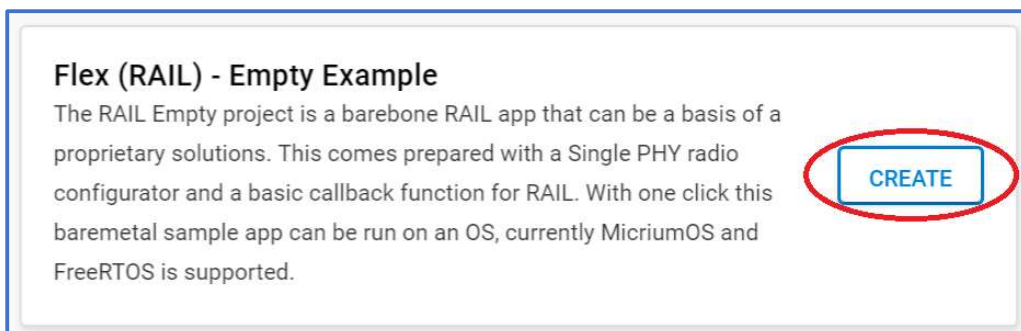
22. Build the updated example by once again right-clicking the project name and selecting **Build Project**

23. Start the debugger via **Debug As > 1 Silicon Labs Arm Program**, as in step 9, and click the **Resume** button to run the code

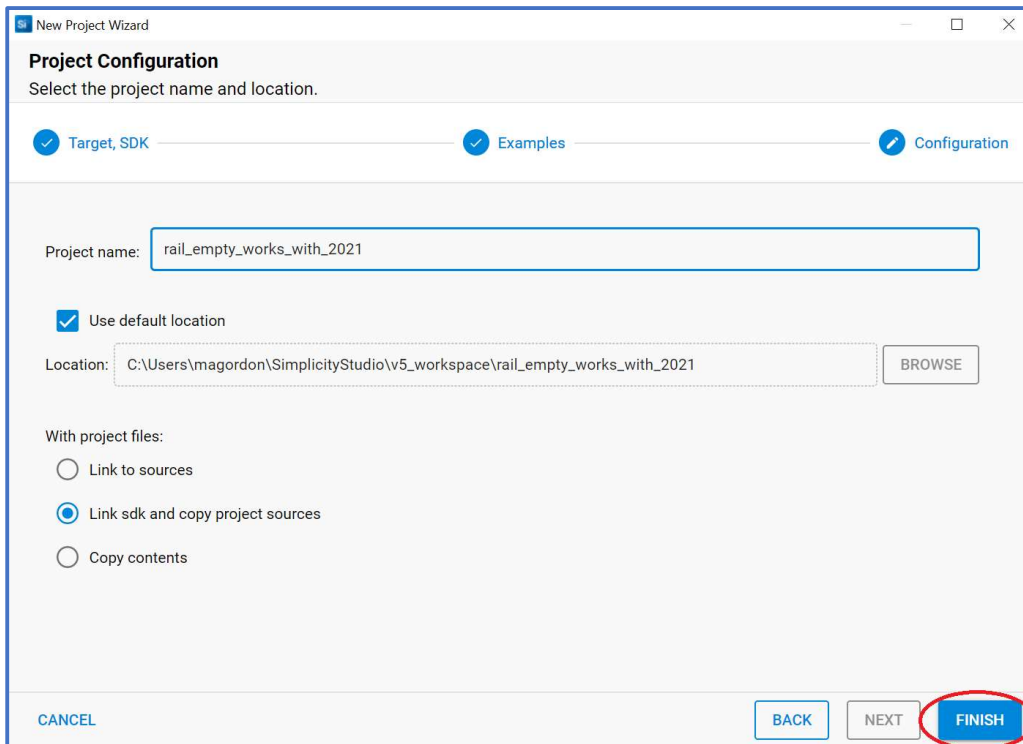
24. Confirm that the newly created FreeRTOS task is toggling your board's RG LED with a frequency of one second and reconnect to the project on your phone with EFR Connect to ensure that the temperature is still being updated

Part 2: Adding Micrium OS to a Flex (Proprietary Wireless) Example

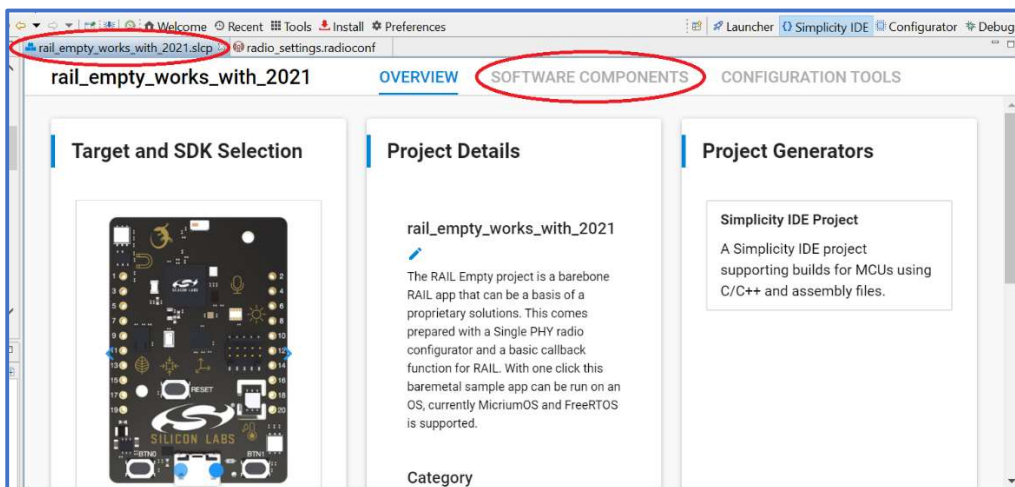
1. After terminating your most recent debug session, return to the Simplicity Studio **Launcher**, scroll to **Flex (RAIL) – Empty Example** and click **Create**



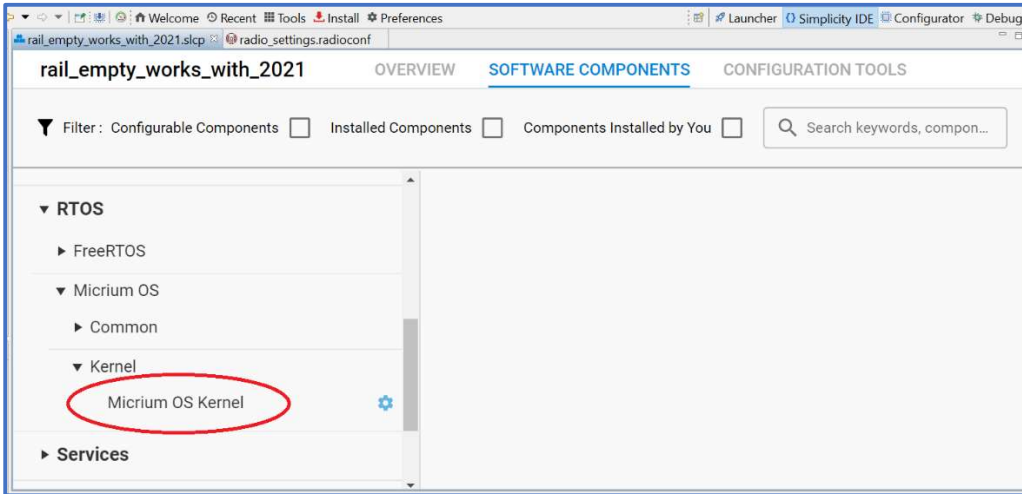
2. Update the new project's name according to your preferences and click **Finish**



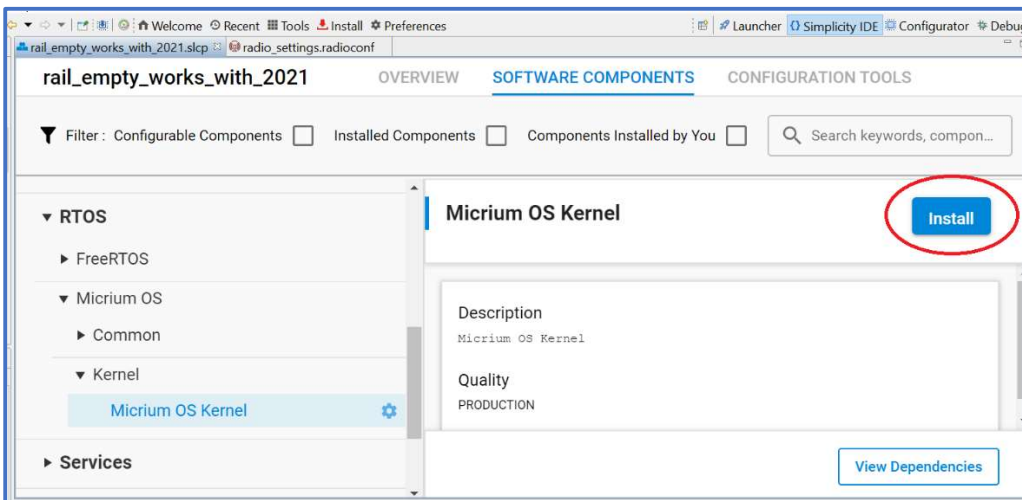
3. In the **Simplicity Studio IDE** perspective (which should have opened automatically when the project was created), open the tab corresponding to your new project's **.slcp* file and then click on **Software Components**



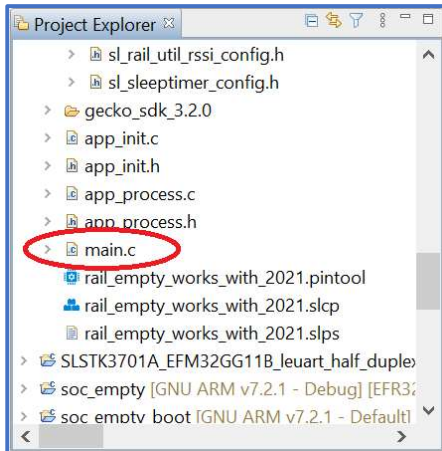
4. In the list on the left-hand side of the page, expand **RTOS > Micrium OS > Kernel** and then click **Micrium OS Kernel**



5. Click the **Install** button that should now appear on the right-hand side of the screen



- Expand the new project's listing in the **Project Explorer** and double-click *main.c* to open this file



- Insert the following code (which contains definitions and declarations needed by a Micrium OS task that you will declare in the next few steps) into *main.c*, just below the file's existing `#include` directives

```
#include "os.h"
#include "em_gpio.h"

#define APP_LED_TASK_PRIORITY          56u
#define APP_LED_TASK_STACK_SIZE      200u
#define APP_LED_PORT                  8u

static CPU_STK  AppTaskLED_Stk[APP_LED_TASK_STACK_SIZE];
static OS_TCB   AppTaskLED_TCB;

static void AppTaskLED (void *p_arg);
```

8. Insert the following code into the *main.c* declaration of `main()`, just below the call to `sl_system_init()`

```
RTOS_ERR err;

GPIO_PinModeSet(gpioPortD,
                APP_LED_PORT,
                gpioModePushPull,
                0);

OSTaskCreate(&AppTaskLED_TCB,
            "LED Task",
            AppTaskLED,
            (void *)0,
            APP_LED_TASK_PRIORITY,
            AppTaskLED_Stk,
            0,
            APP_LED_TASK_STACK_SIZE,
            1u,
            0u,
            0u,
            OS_OPT_TASK_STK_CHK | OS_OPT_TASK_STK_CLR |
            OS_OPT_TASK_SAVE_FP,
            &err);
```

9. Add the below code (which declares a periodic task that will blink your Thunderboard Sense 2's RG LED once every second) to the bottom of *main.c*

```
static void AppTaskLED (void *p_arg)
{
    RTOS_ERR err;

    (void)p_arg;

    while (1) {
        GPIO_PinOutToggle(gpioPortD, APP_LED_PORT);
        OSTimeDlyHMSM(0, 0, 1, 0, 0, &err);
    }
}
```

10. Save *main.c* (**File>Save**) and build the project by right-clicking its name in **Project Explorer** and selecting **Build Project**

11. To download the project's code to the board, again right-click the project name and select **Debug As > 1 Silicon Labs Arm Program**

12. Run the code by clicking the **Resume** button

13. Check that the RG LED is toggling once per second on your Thunderboard Sense 2