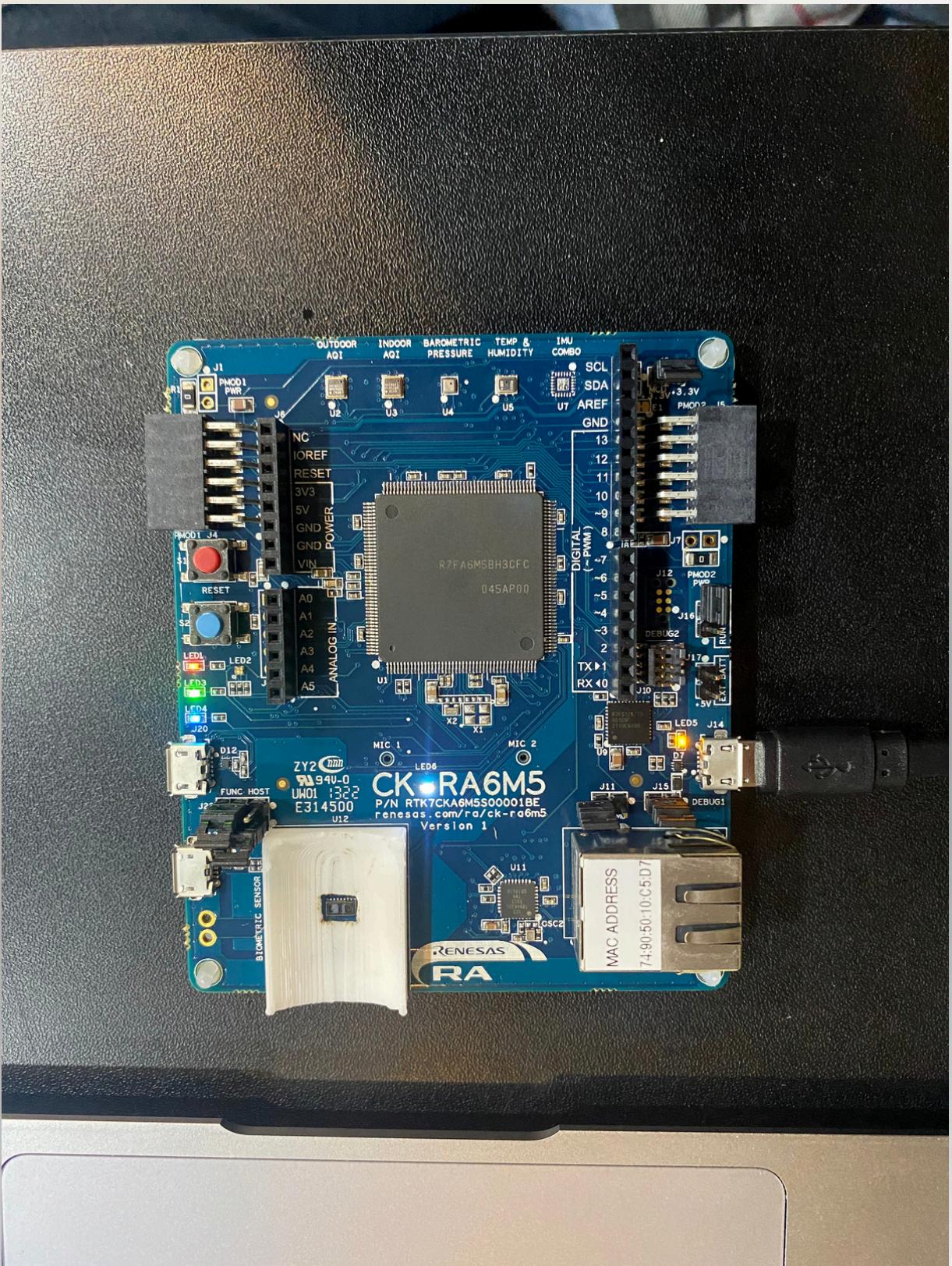


# Blinky LED with Renesas



Nhóm 1:

- Do Minh Quan - 20T2
- Truong Quang Chu - 20T2
- Le Van Dat - 20T2

# Tổng quan

- ▶ Giới thiệu
- ▶ Hướng dẫn cài đặt
- ▶ Giải thích code
- ▶ Demo



alamy

Image ID: 2M82DA1  
www.alamy.com

# I. GIỚI THIỆU

# Giới thiệu về Renesas Electronics Corporation

Renesas Electronics Corporation là một trong những công ty hàng đầu trong lĩnh vực bán dẫn, chuyên về các giải pháp microcontroller (MCU) và microprocessor (MPU).

Được thành lập vào năm 2010 sau khi sáp nhập NEC Electronics Corporation và Renesas Technology Corporation.



# Renesas cloud kit CK-RA6M5

RA6M5 được thiết kế cho các ứng dụng IoT, với khả năng bảo mật tích hợp và hiệu suất xử lý cao.

Thông số kỹ thuật:

- Vi điều khiển: Renesas RA6M5 với bộ xử lý ARM Cortex-M33.
- Tần số: 200 MHz.
- Bộ nhớ: 2 MB Code Flash và 512 KB SRAM.
- 176 pins, LQFP package
- Giao tiếp: Hỗ trợ nhiều giao thức như UART, SPI, I2C, CAN, và USB
- Tính năng bảo mật: Tích hợp Trusted Secure IP, hỗ trợ mã hóa phần cứng.



# E2Studio

Hỗ trợ đa nền tảng: Vi điều khiển RA, RX, RL78, RH850.

Công cụ gỡ lỗi mạnh mẽ: JTAG, SWD, Renesas E2 Emulator.

Cài từ <https://github.com/renesas/fsp> cho RA MCUs



The screenshot shows the E2Studio IDE interface. The central area is a code editor displaying C++ code for a plugin. The code includes methods like `_showConnectMethod()`, `_showUpdate()`, and `_showStatus()`. The code editor has syntax highlighting and a vertical scrollbar. To the left is a project tree view showing various source files and folders. On the right side, there are several toolbars and windows, including a 'Logs' window showing build logs and a 'Console' window. The top menu bar includes options like File, Edit, View, Start, Debug, Utilities, Subproject, Project, Refactoring, Run, Settings, Window, Resources, Plugins, Help. The bottom status bar shows file paths and line numbers.

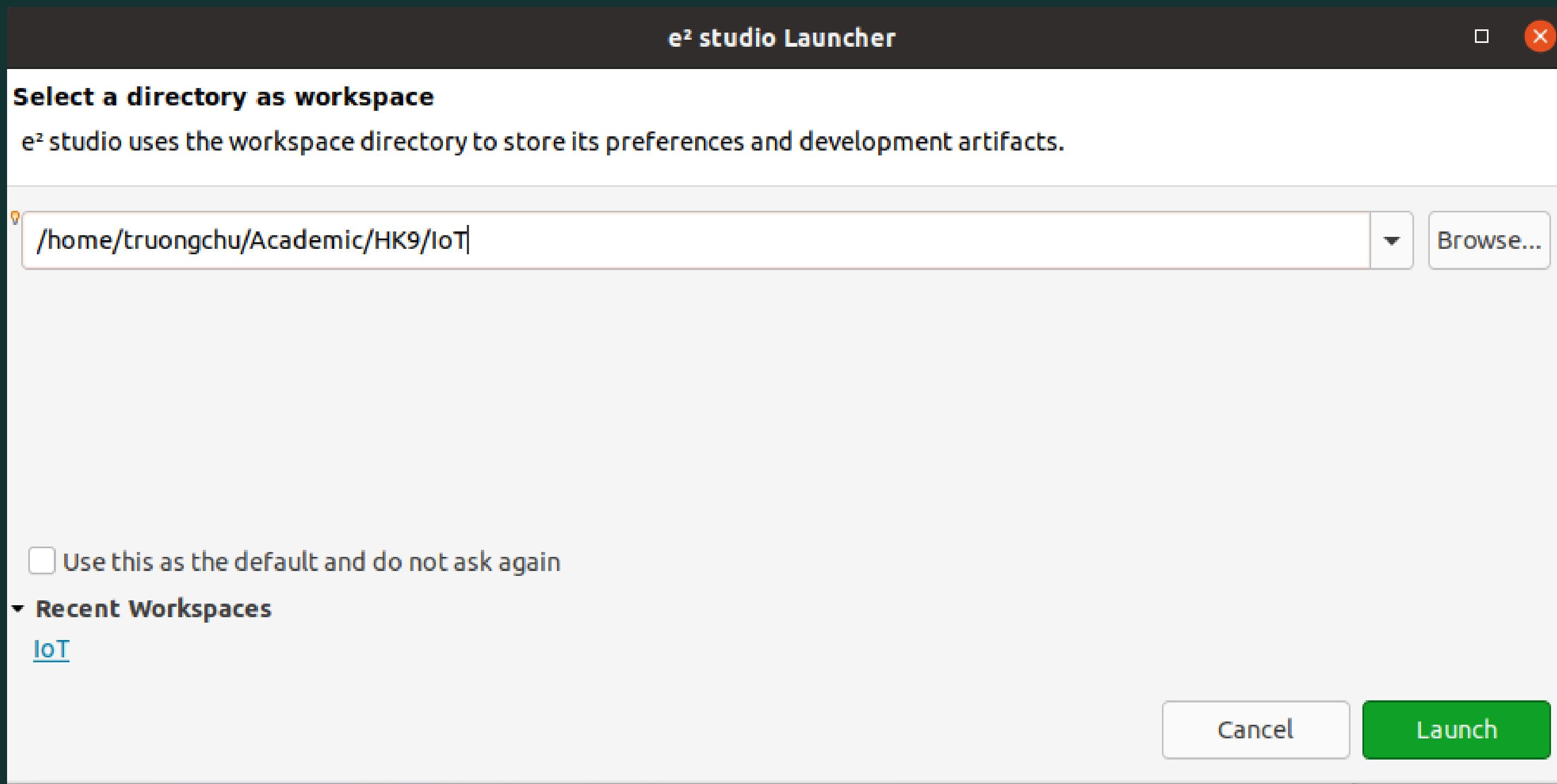
# e<sup>2</sup> studio



## II. Hướng dẫn cài đặt

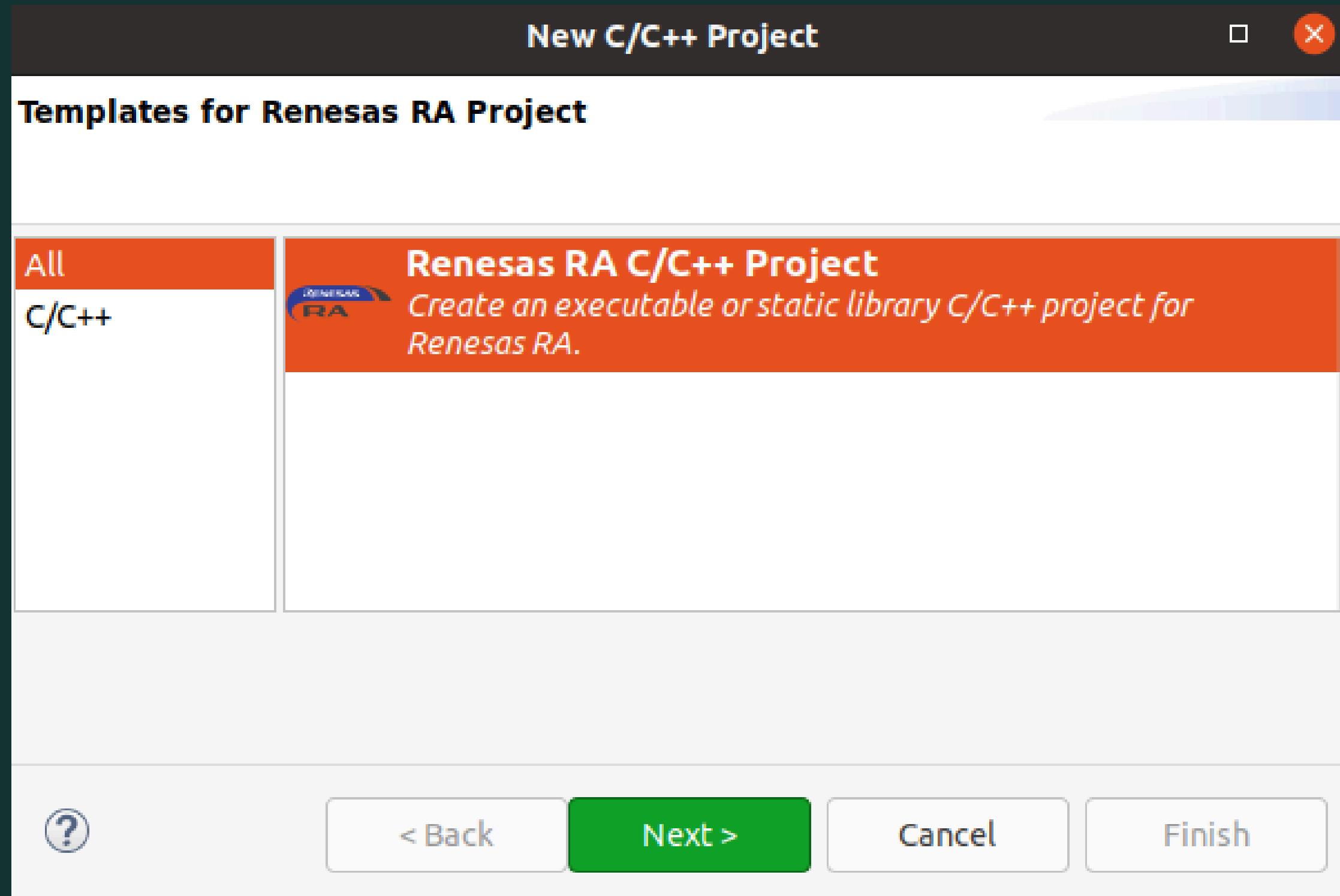
# Hướng dẫn cài đặt chương trình

Bước 1: Mở e2studio và tạo mới hoặc sử dụng 1 workspace có sẵn



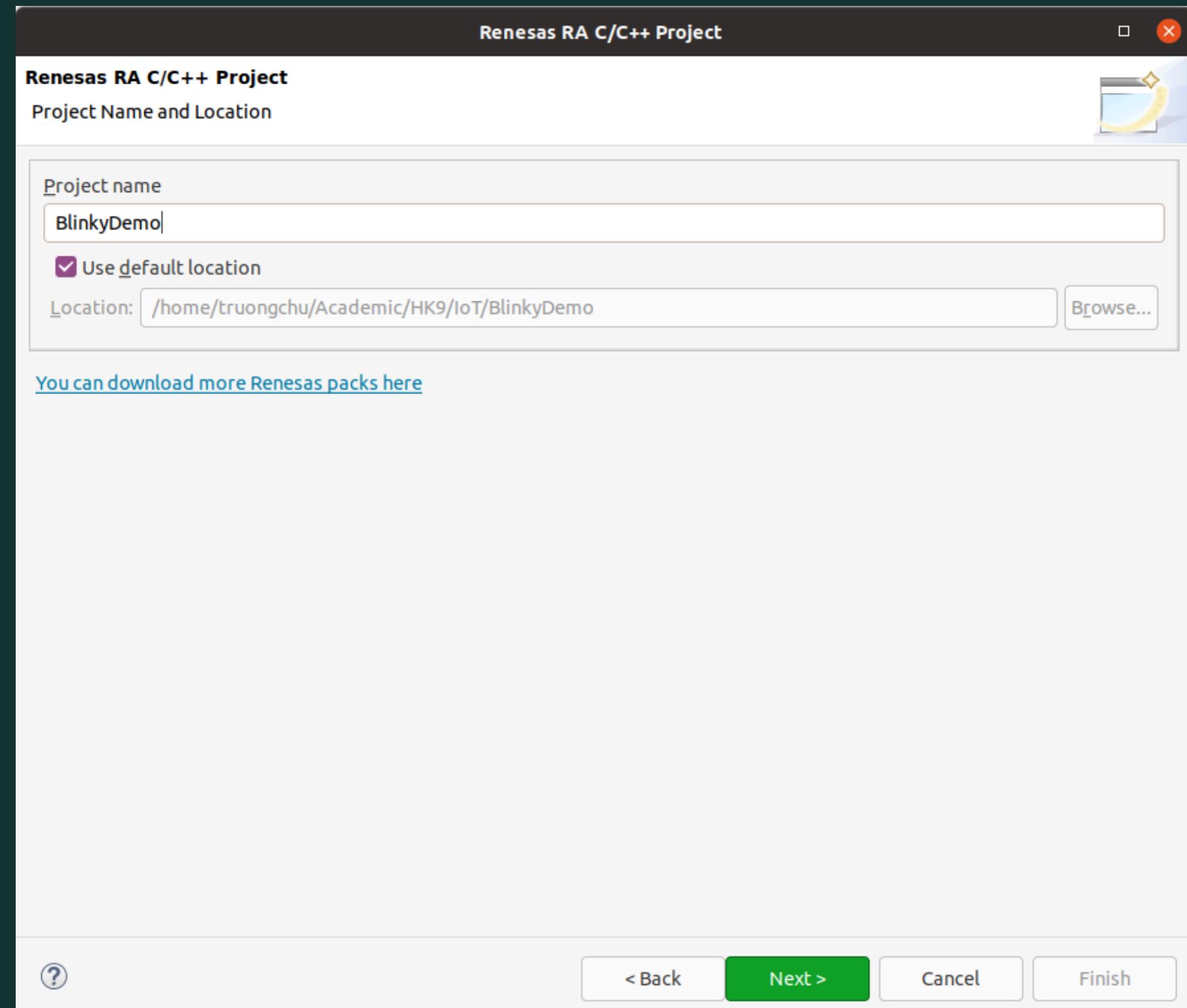
# Hướng dẫn cài đặt chương trình

Bước 2: File -> New -> Renesas C/C++ Project -> Renesas RA (CK RA6M5 MCU)



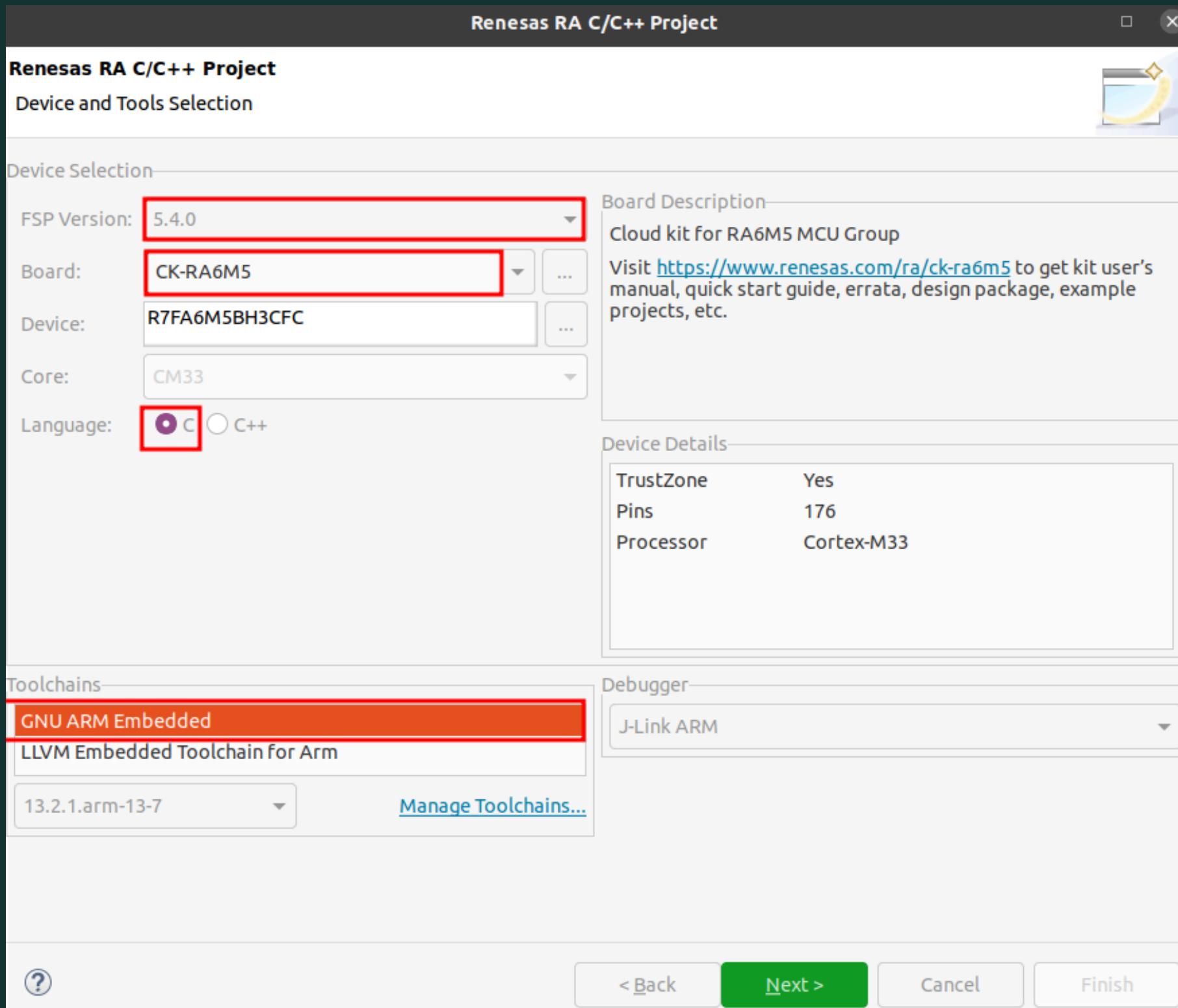
# Hướng dẫn cài đặt chương trình

## Bước 3: Nhập tên cho Project



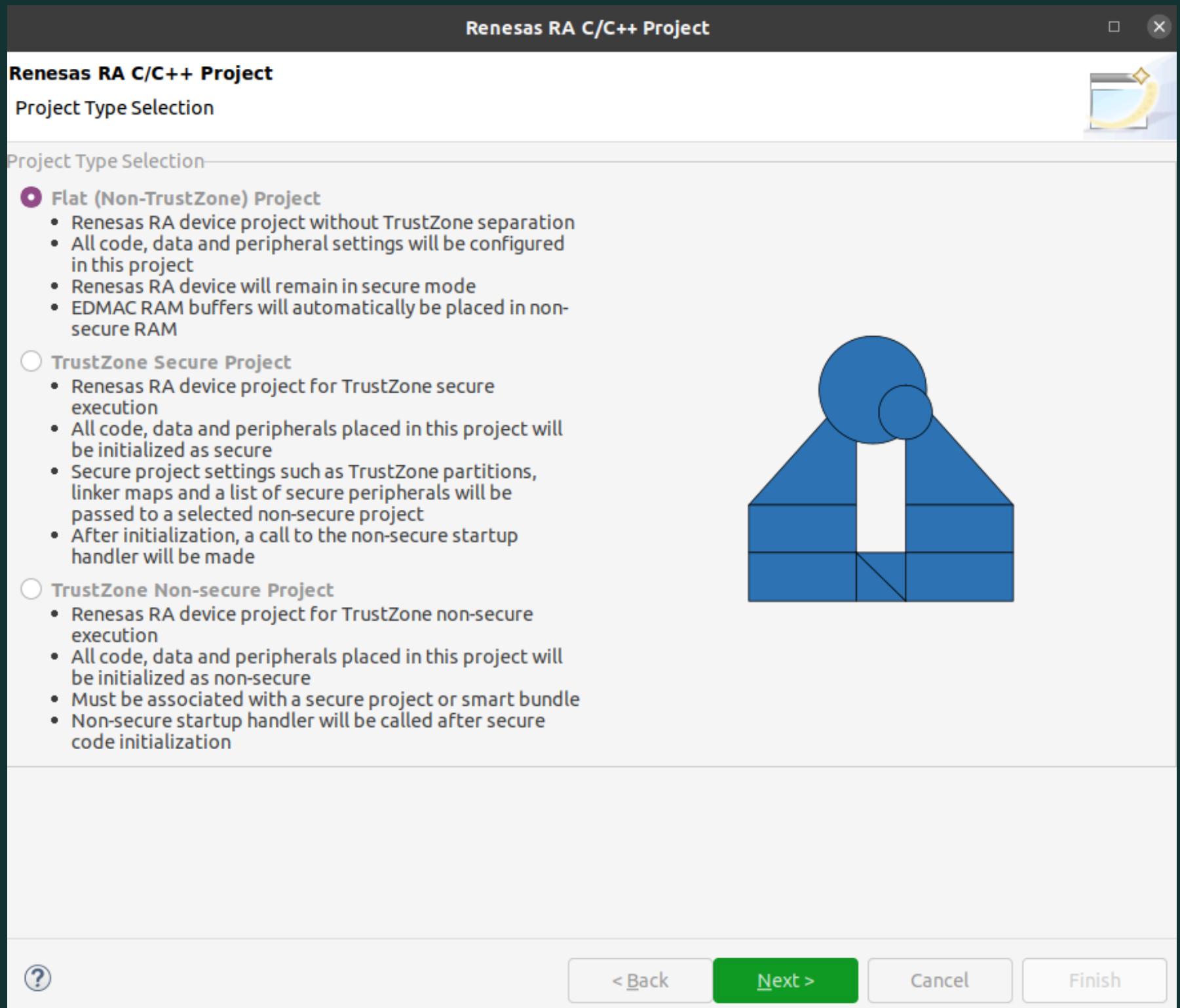
# Hướng dẫn cài đặt chương trình

- Bước 4: Chọn FSP version > Chọn Board > Chọn ngôn ngữ > Next



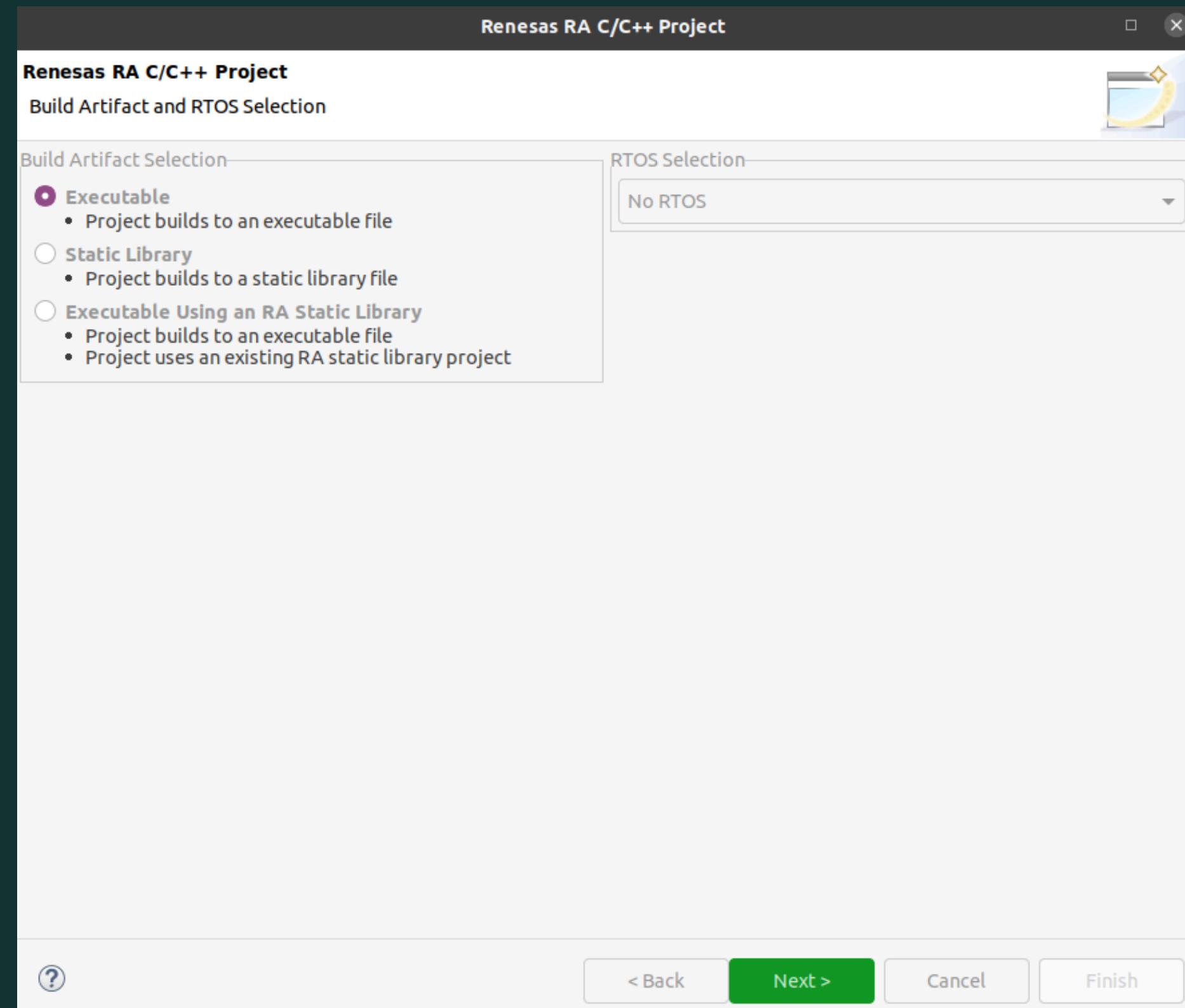
# Hướng dẫn cài đặt chương trình

## Bước 5: Flat (Non-TrustZone) Project => Chọn Next



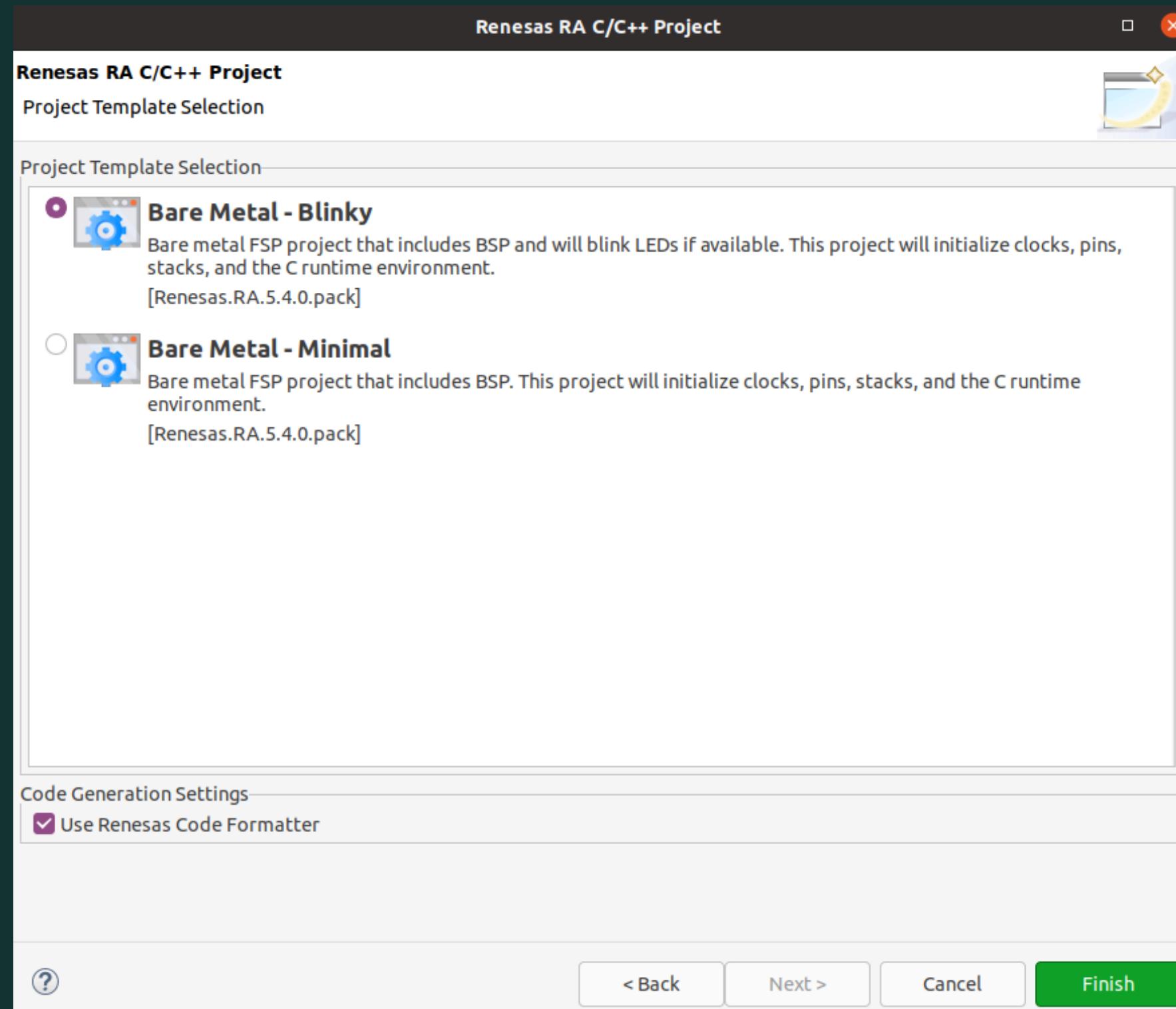
# Hướng dẫn cài đặt chương trình

## Bước 6: Chọn Executable => Next



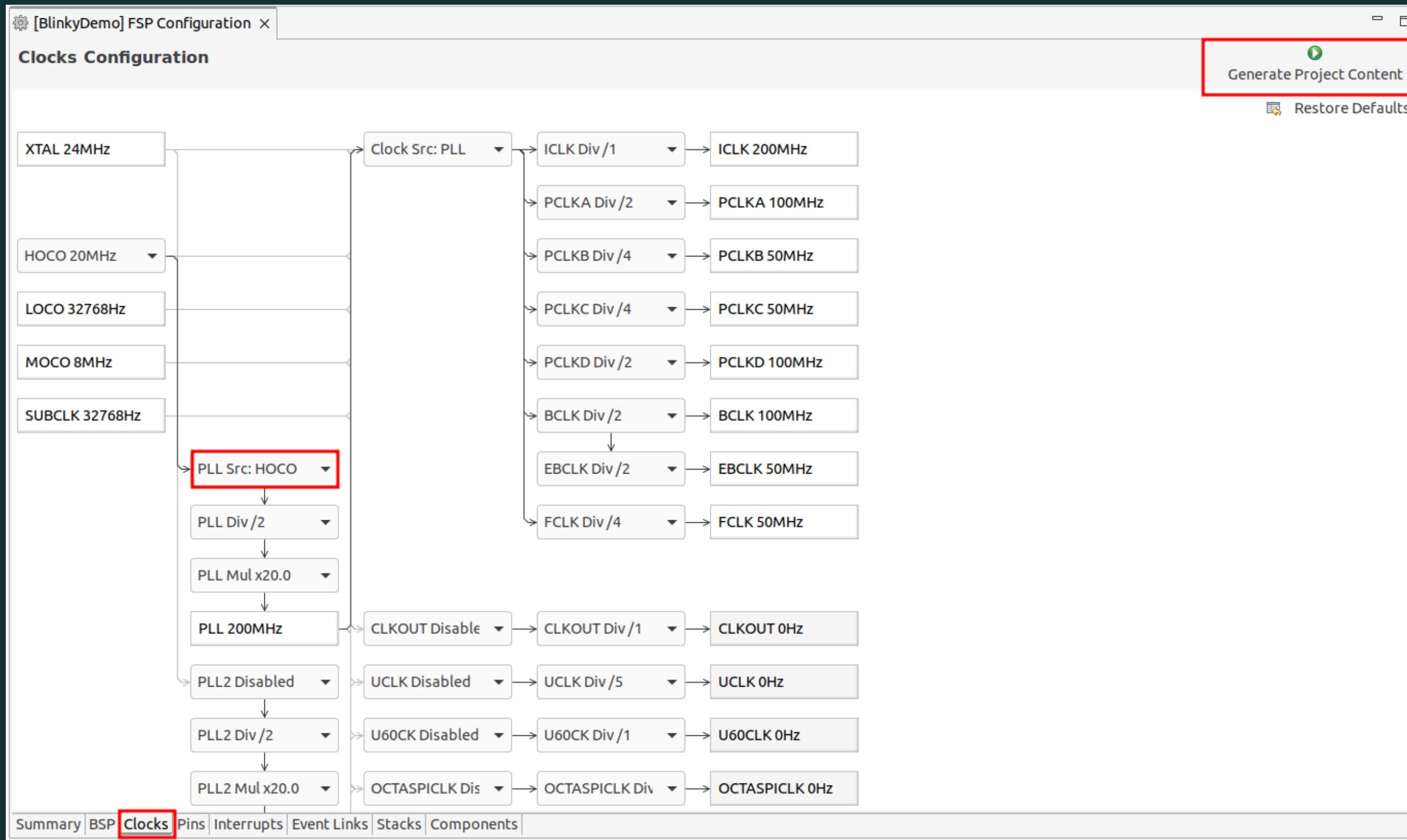
# Hướng dẫn cài đặt chương trình

Bước 7: Chọn Bare Metal - Blinky => Finish



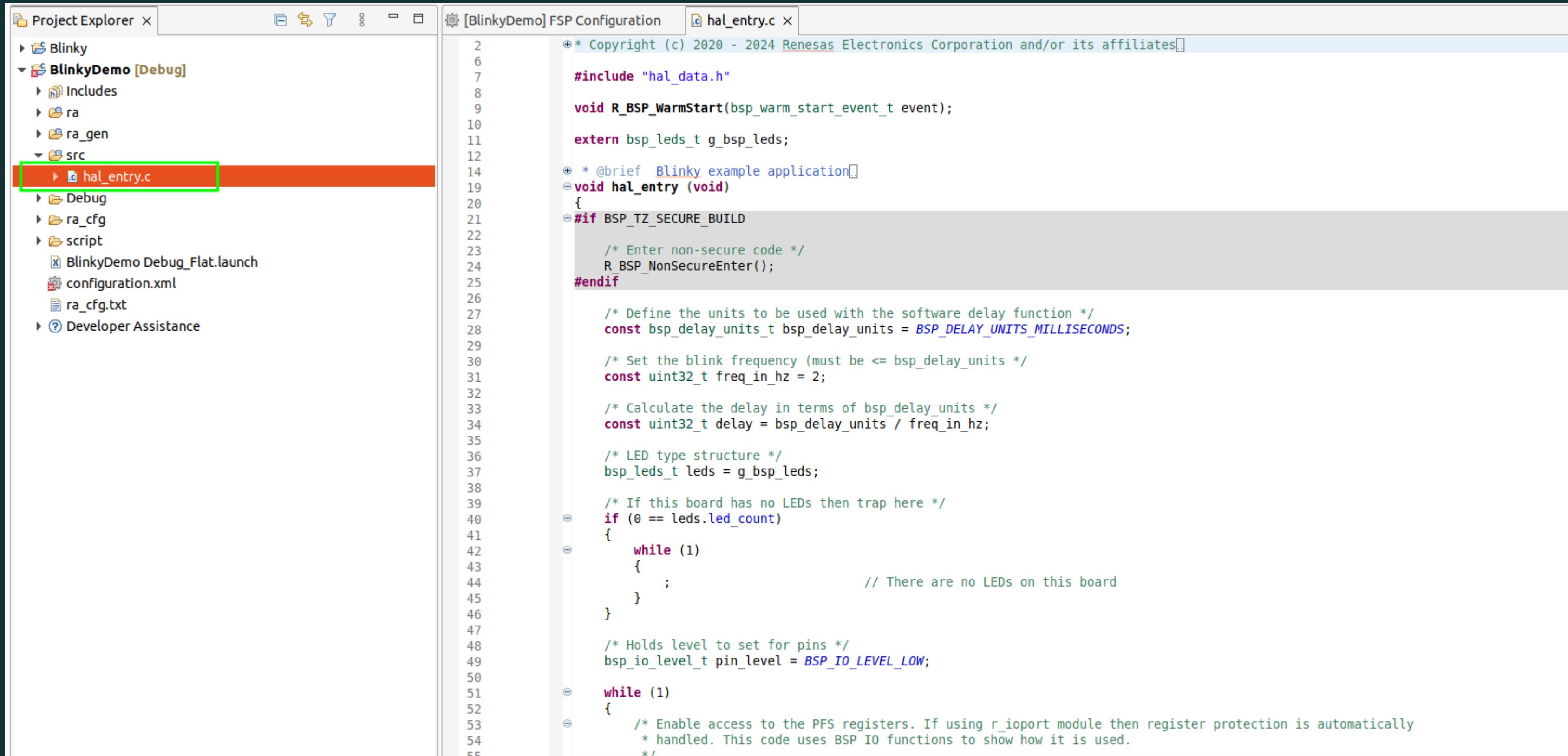
# Hướng dẫn cài đặt chương trình

## Bước 8: Chọn PPL Src: HOCO > Generate Project Content



# Hướng dẫn cài đặt chương trình

## Bước 9: File code được gen trong hal\_entry.c



The screenshot shows a software development environment with the following interface elements:

- Project Explorer:** Shows the project structure. The 'BlinkyDemo [Debug]' folder is expanded, revealing 'Includes', 'ra', 'ra\_gen', and 'src'. Inside 'src', the file 'hal\_entry.c' is selected and highlighted with a red box.
- [BlinkyDemo] FSP Configuration:** Shows the configuration settings for the project.
- Code Editor:** Displays the content of the 'hal\_entry.c' file. The code is written in C and includes comments explaining the purpose of various sections. The file starts with copyright information, includes 'hal\_data.h', defines a warm start function, and contains a main loop for blinking LEDs. It uses BSP functions like 'R\_BSP\_WarmStart', 'R\_BSP\_NonSecureEnter', and 'R\_BSP\_SetPinLevel'.

```
* Copyright (c) 2020 - 2024 Renesas Electronics Corporation and/or its affiliates.

#include "hal_data.h"

void R_BSP_WarmStart(bsp_warm_start_event_t event);

extern bsp_leds_t g_bsp_leds;

/* @brief Blinky example application */
void hal_entry (void)
{
#if BSP_TZ_SECURE_BUILD

    /* Enter non-secure code */
    R_BSP_NonSecureEnter();
#endif

    /* Define the units to be used with the software delay function */
    const bsp_delay_units_t bsp_delay_units = BSP_DELAY_UNITS_MILLISECONDS;

    /* Set the blink frequency (must be <= bsp_delay_units */
    const uint32_t freq_in_hz = 2;

    /* Calculate the delay in terms of bsp_delay_units */
    const uint32_t delay = bsp_delay_units / freq_in_hz;

    /* LED type structure */
    bsp_leds_t leds = g_bsp_leds;

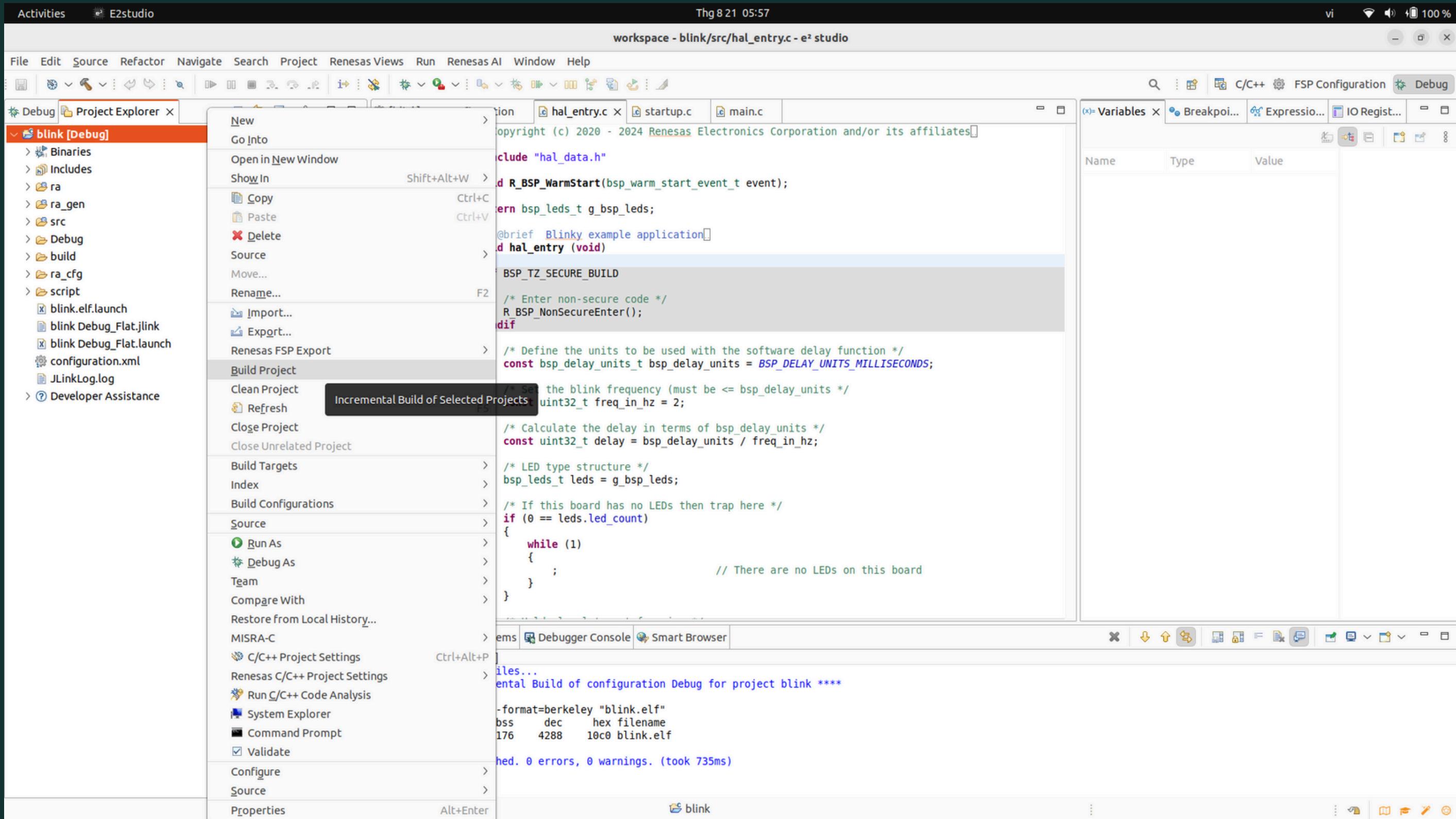
    /* If this board has no LEDs then trap here */
    if (0 == leds.led_count)
    {
        while (1)
        {
            ;
            // There are no LEDs on this board
        }
    }

    /* Holds level to set for pins */
    bsp_io_level_t pin_level = BSP_IO_LEVEL_LOW;

    while (1)
    {
        /* Enable access to the PFS registers. If using r_ioport module then register protection is automatically
         * handled. This code uses BSP IO functions to show how it is used.
        */
    }
}
```

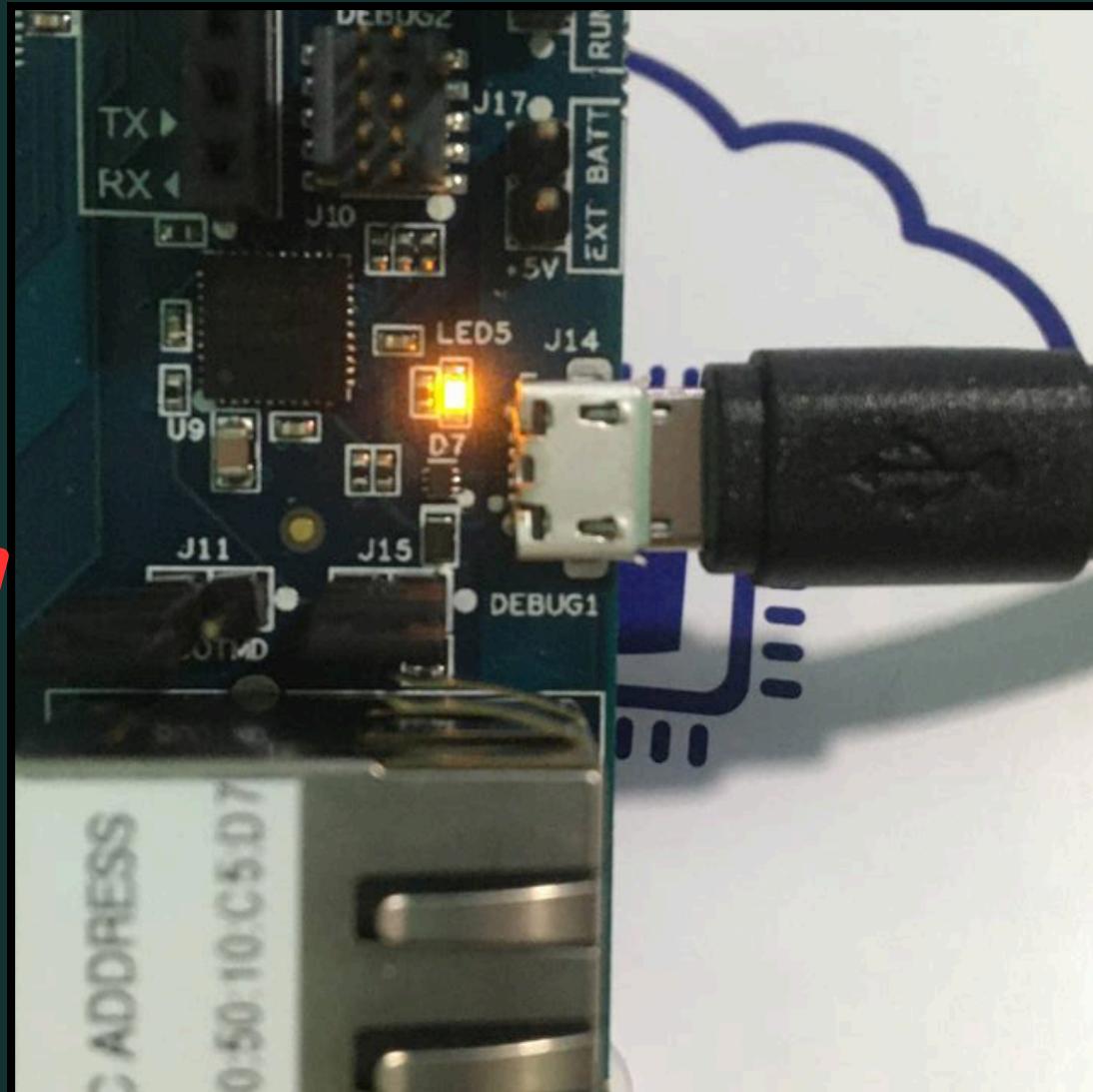
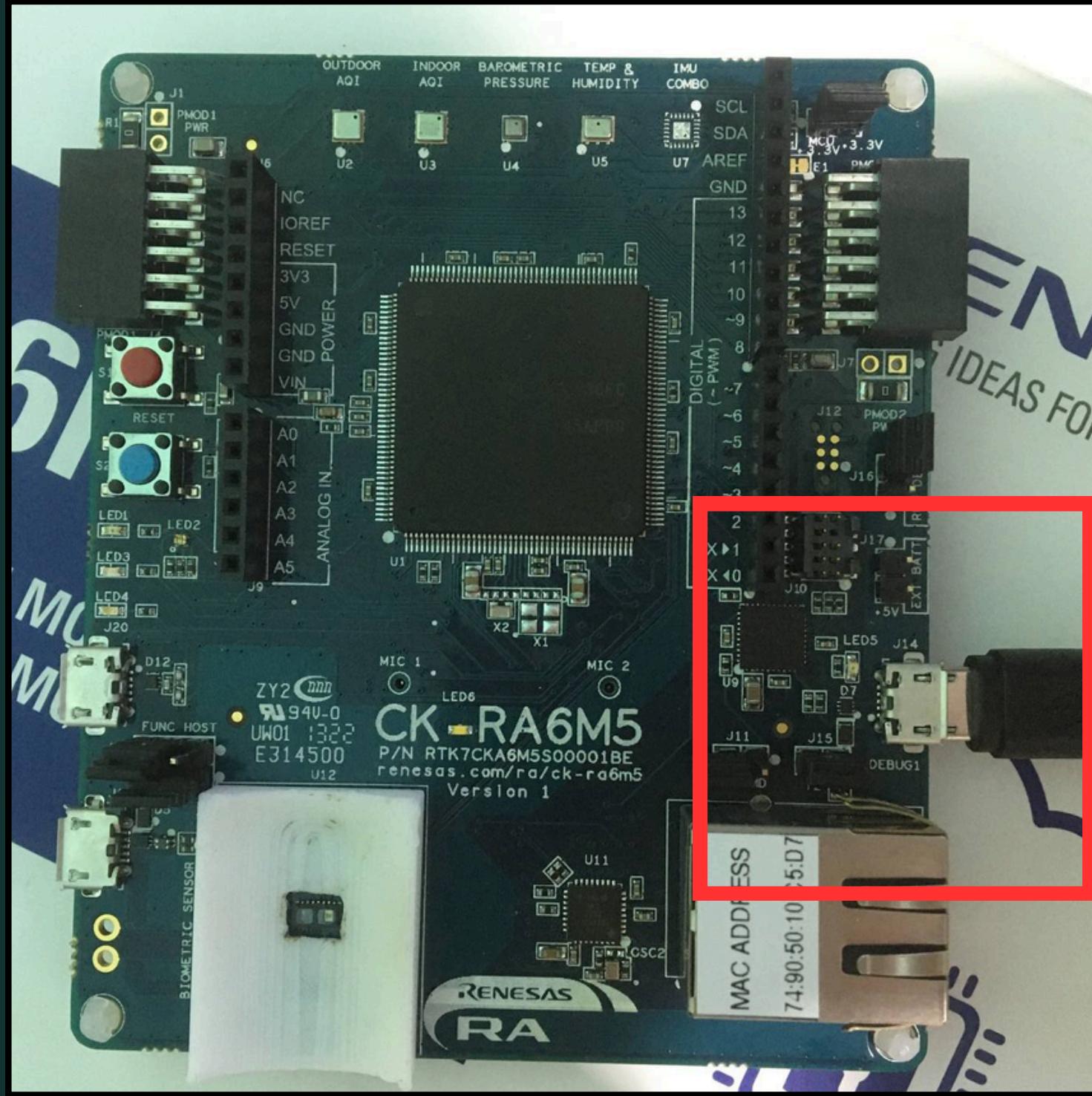
# Hướng dẫn cài đặt chương trình

## Bước 10: Nhấn chuột phải vào project => Build Project



# Hướng dẫn cài đặt chương trình

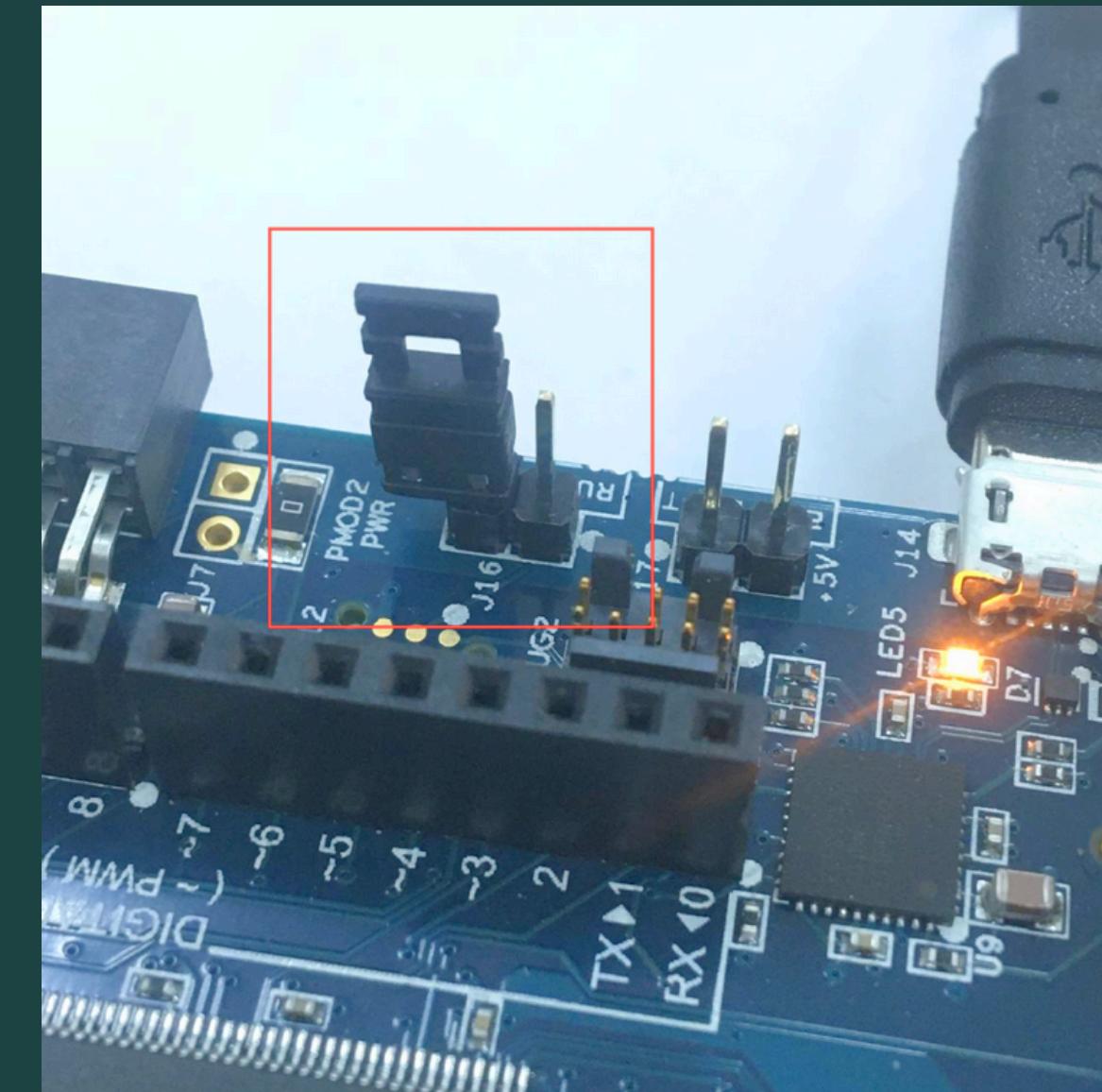
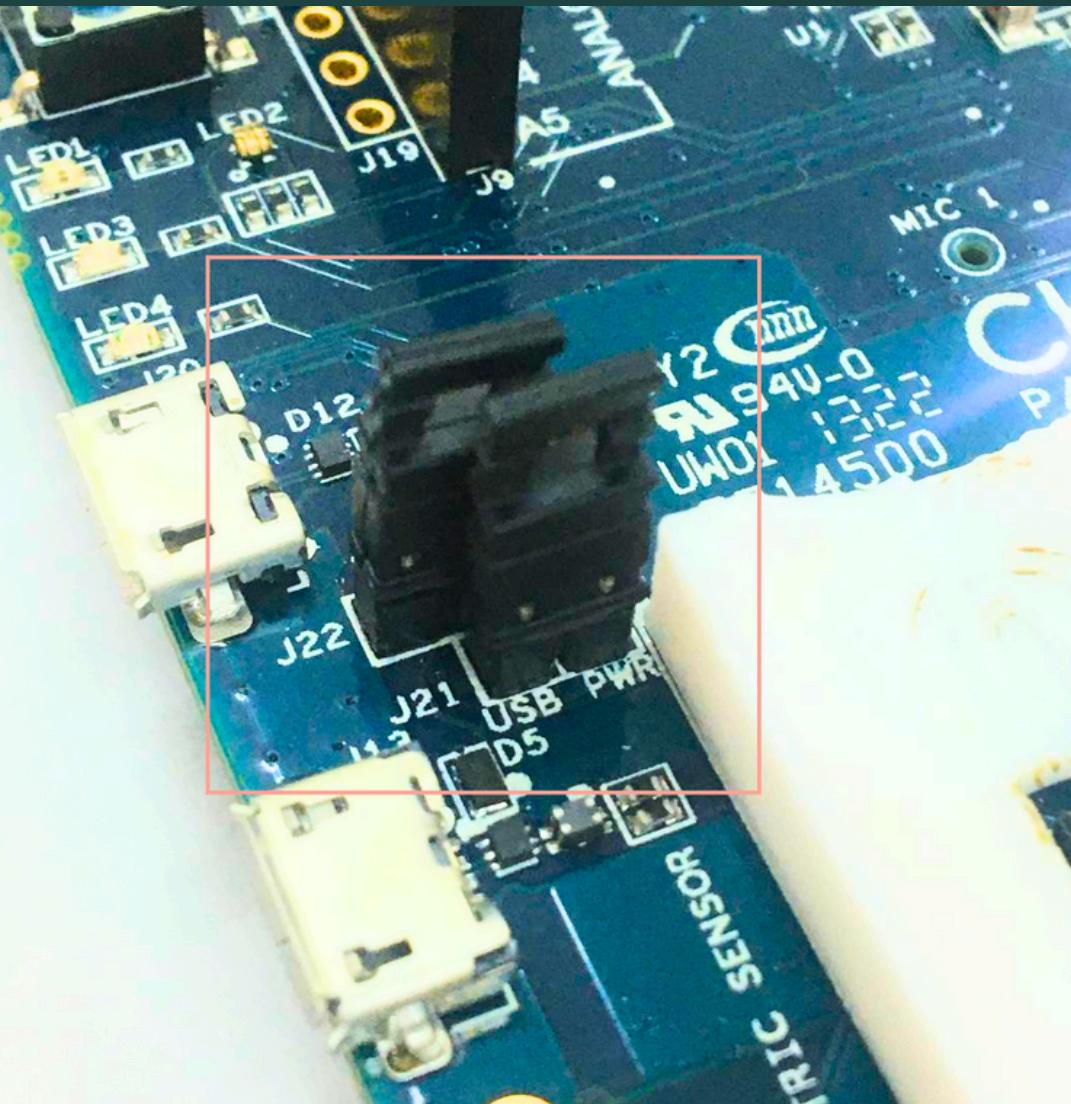
Bước 11: Kết nối dây vào cổng Debug 1



# Hướng dẫn cài đặt chương trình

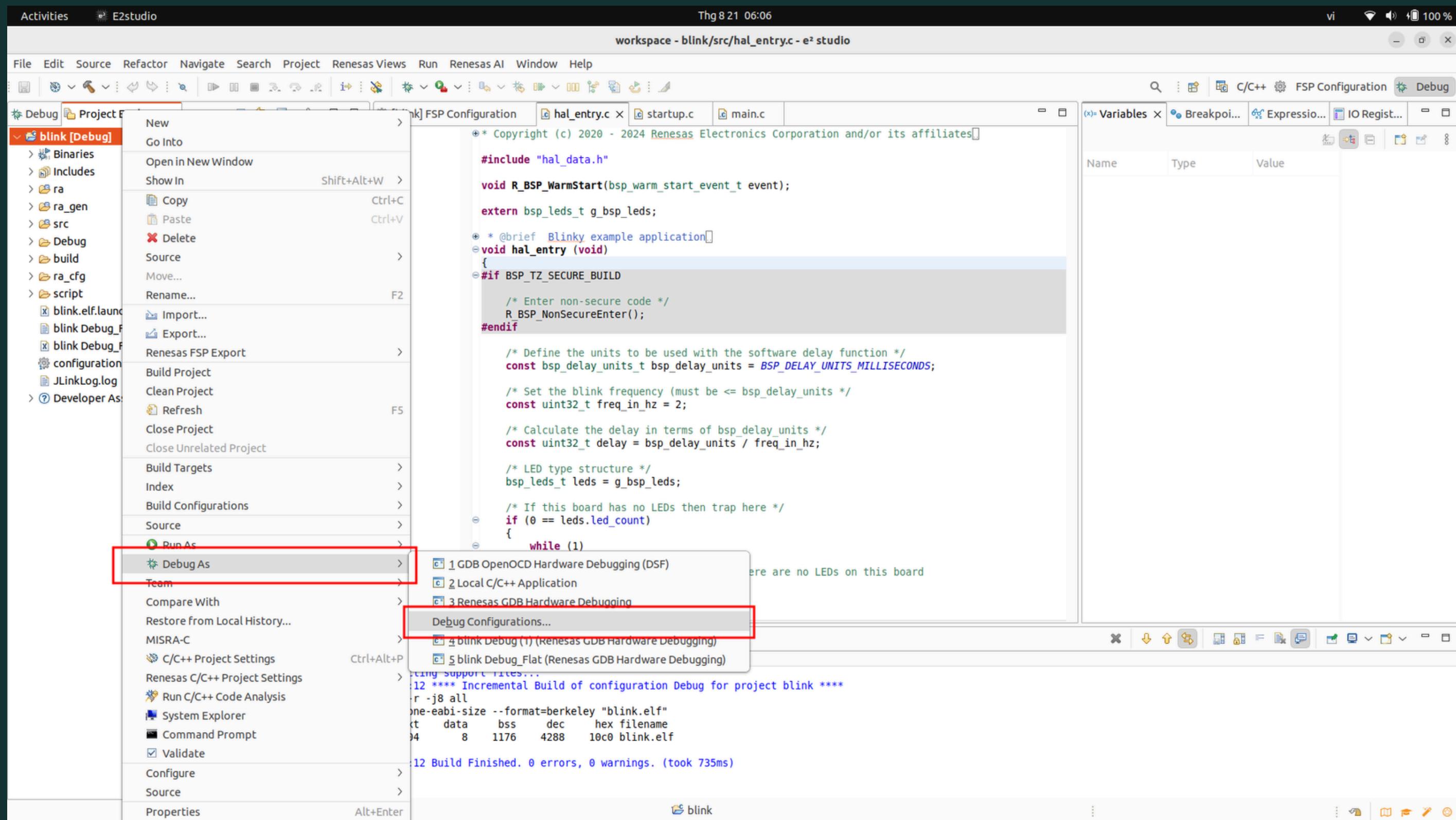
Lưu ý:

- J22 link pins 2-3
- J21 link đóng
- J16 link mở



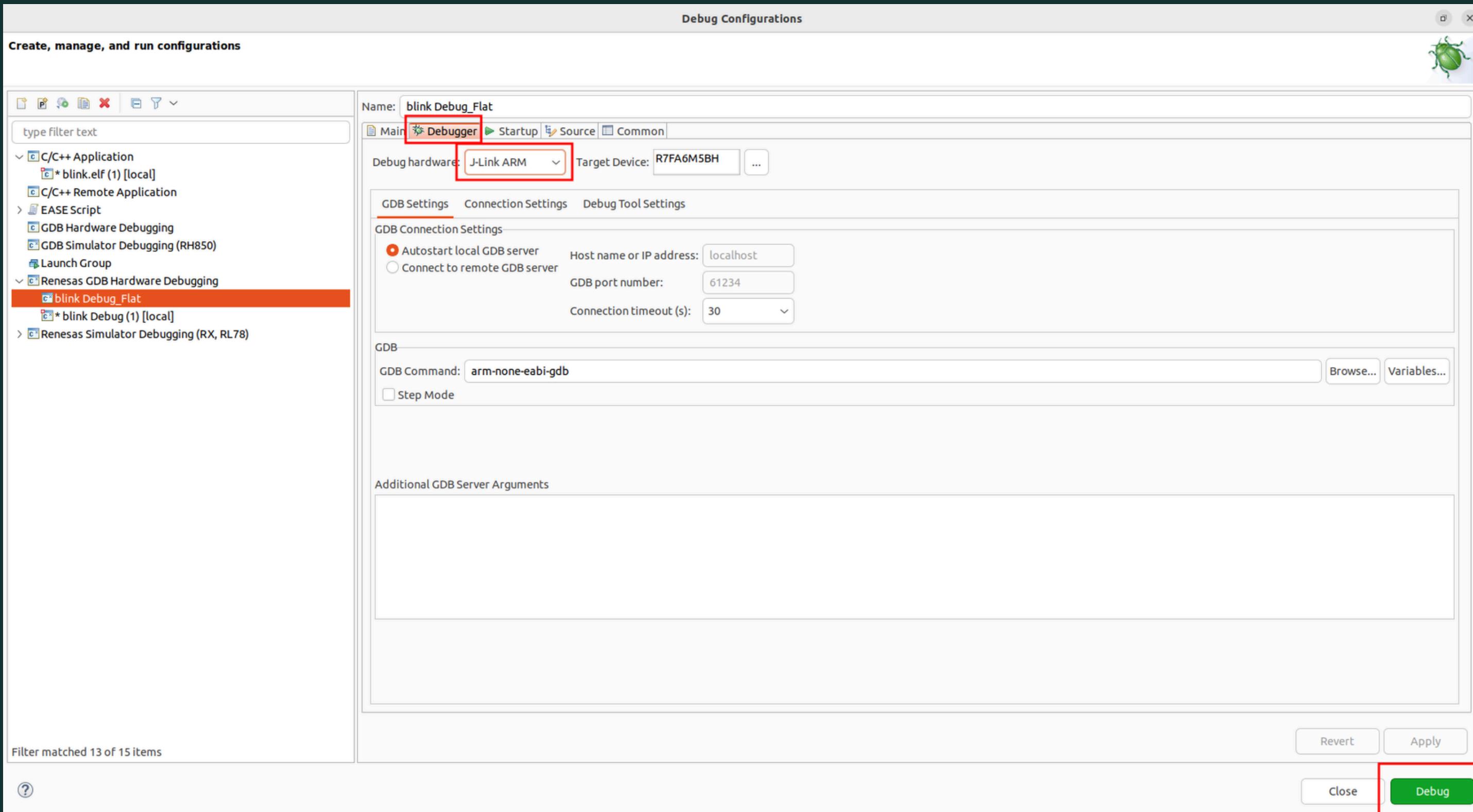
# Hướng dẫn cài đặt chương trình

## Bước 12: Nhấn chuột phải vào project > Debug As > Debug Configurations



# Hướng dẫn cài đặt chương trình

Bước 13: Chọn tab Debugger > Chọn J-Link ARM > Debug



# Kết quả



# MacOS

Nếu gặp lỗi: “could not determine gdb version using command: arm-none-eabi-gdb --version”

=> Kiểm tra Python đã được cài đặt trong /Library/Frameworks/Python.framework/3.10 chưa ?

=> Cài đặt python 3.10 theo link sau

<https://www.python.org/ftp/python/3.10.8/python-3.10.8-macos11.pkg>.



### III. Giải thích code

# Giải thích code

```
#include "hal_data.h"                                     hardware abstraction layer (HAL)
void R_BSP_WarmStart(bsp_warm_start_event_t event);      called at various points during the startup process.
extern bsp_leds_t g_bsp_leds;                            holds information about the LEDs
```

```
void hal_entry (void)
{
#if BSP_TZ_SECURE_BUILD
    /* Enter non-secure code */
    R_BSP_NonSecureEnter();
#endif

    /* Define the units to be used with the software delay function */
    const bsp_delay_units_t bsp_delay_units = BSP_DELAY_UNITS_MILLISECONDS;      Delay units

    /* Set the blink frequency (must be <= bsp_delay_units */
    const uint32_t freq_in_hz = 2;                                                 2 blinks per second

    /* Calculate the delay in terms of bsp_delay_units */
    const uint32_t delay = bsp_delay_units / freq_in_hz;                          the delay would be 500 ms

    /* LED type structure */
    bsp_leds_t leds = g_bsp_leds;                                                 load LED information into leds variable

    /* If this board has no LEDs then trap here */
    if (0 == leds.led_count)
    {
        while (1)
        {
            ;                                         // There are no LEDs on this board
        }
    }

    /* Holds level to set for pins */
    bsp_io_level_t pin_level = BSP_IO_LEVEL_LOW;
```

# Giải thích code

```
/* Holds level to set for pins */
bsp_io_level_t pin_level = BSP_IO_LEVEL_LOW; ————— init with low

while (1)
{
    /* Enable access to the PFS registers. If using r_ioport module then register protection is automatically
     * handled. This code uses BSP IO functions to show how it is used.
     */
    R_BSP_PinAccessEnable(); ————— enable pin access

    /* Update all board LEDs */
    for (uint32_t i = 0; i < leds.led_count; i++)
    {
        /* Get pin to toggle */
        uint32_t pin = leds.p_leds[i];

        /* Write to this pin */
        R_BSP_PinWrite((bsp_io_port_pin_t) pin, pin_level); ————— set pin_level for each led
    }

    /* Protect PFS registers */
    R_BSP_PinAccessDisable();

    /* Toggle level for next write */
    if (BSP_IO_LEVEL_LOW == pin_level)
    {
        pin_level = BSP_IO_LEVEL_HIGH;
    }
    else
    {
        pin_level = BSP_IO_LEVEL_LOW;
    }

    /* Delay */
    R_BSP_SoftwareDelay(delay, bsp_delay_units); ————— delay for pin level
}
```

# Giải thích code

```
+ * This function is called at various points during the startup process. This implementation uses the event that is
⊖ void R_BSP_WarmStart (bsp_warm_start_event_t event)
{
    if (BSP_WARM_START_RESET == event) ——— reset event
    {
#ife BSP_FEATURE_FLASH_LP_VERSION != 0

        /* Enable reading from data flash. */
        R_FACI_LP->DFLCTL = 1U; ——— enable data flash

        /* Would normally have to wait tDSTOP(6us) for data flash recovery. Placing the enable here, before clock and
         * C runtime initialization, should negate the need for a delay since the initialization will typically take more than 6
#endif
    }

    if (BSP_WARM_START_POST_C == event) ——— C runtime & clocks are setup
    {
        /* C runtime environment and system clocks are setup. */

        /* Configure pins. */
        R_IOPORT_Open(&IOPORT_CFG_CTRL, &IOPORT_CFG_NAME); ——— opening the I/O port
    }
}
```

# API Docs

The screenshot shows the RA Flexible Software Package Documentation website. The header includes the Renesas logo, the title "RA Flexible Software Package Documentation Release v5.4.0", and a search bar. The left sidebar contains a navigation tree with "RA Flexible Software Package Documentation" expanded, showing "Introduction", "Reference Materials", "Starting Development", "FSP Architecture", "API Reference", and "Copyright". The main content area is titled "BSP I/O access" under the "BSP" category. It features a "Functions" section with five listed functions: `_STATIC_INLINE uint32_t R_BSP_PinRead(bsp_io_port_pin_t pin)`, `_STATIC_INLINE void R_BSP_PinWrite(bsp_io_port_pin_t pin, bsp_io_level_t level)`, `_STATIC_INLINE void R_BSP_PinCfg(bsp_io_port_pin_t pin, uint32_t cfg)`, `_STATIC_INLINE void R_BSP_PinAccessEnable(void)`, and `_STATIC_INLINE void R_BSP_PinAccessDisable(void)`. A blue water drop icon is visible in the top right corner of the content area.

The screenshot shows the documentation for the `R_BSP_PinAccessEnable()` function. The function signature is displayed as `_STATIC_INLINE void R_BSP_PinAccessEnable ( void )`. Below the signature, a description states: "Enable access to the PFS registers. Uses a reference counter to protect against interrupts that could occur via multiple threads or an ISR re-entering this code." A blue water drop icon is visible in the top right corner of the content area.

<https://renesas.github.io/fsp/modules.html>

# Bài tập

1. Nhấp nháy nhanh/ chậm hơn

2. Đèn sáng tuần tự

Các đèn sáng tuần tự từ LED 1 đến LED 2, đến LED 3, LED 4

DEMO