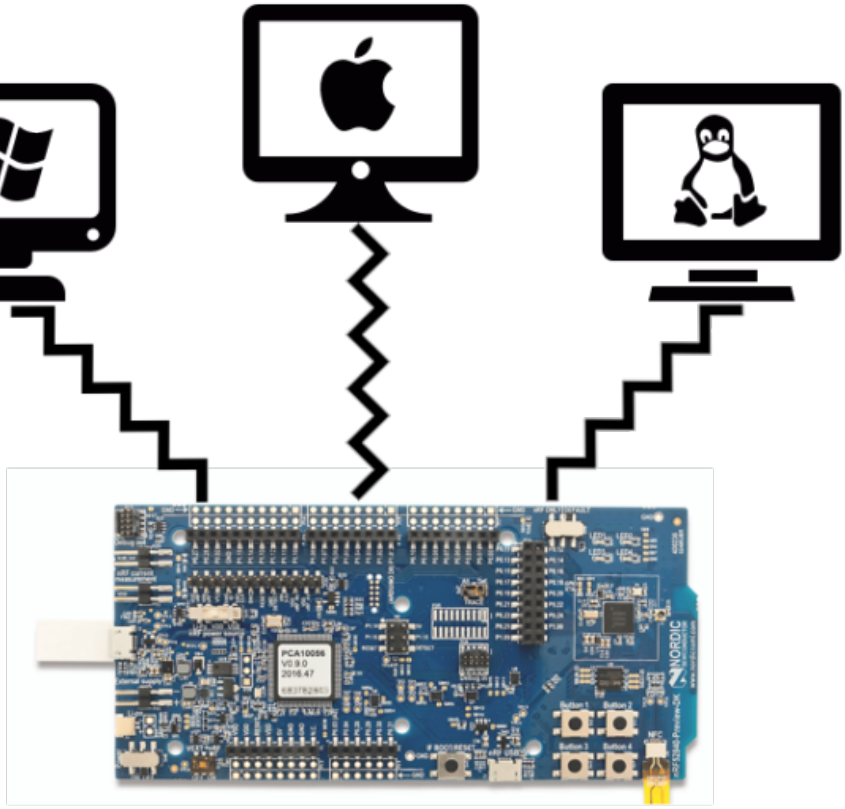


# The complete cross-platform nRF development tutorial

By Mohammad Afaneh (https://www.novelbits.io/author/mafaneh/) | April 10, 2018 | 16 (https://www.novelbits.io/cross-platform-nrf-development-tutorial/#comments)

In a previous blog post (The complete nRF Mac developmeent tutorial (https://www.novelbits.io/nrf52-mac-development-tutorial/)), I went over how to set up your nRF development environment using the NetBeans IDE (https://netbeans.org/) on a Mac. The beauty with NetBeans is that it's cross-platform and customizable to fit your needs. I had been using NetBeans for over a decade, so I was very comfortable with customizing and modifying the different settings and configurations to make it work. Unfortunately, the setup was quite lengthy and involved many changes to make it work, especially with on-target debugging. Even then, it wouldn't work quite as well as I would like it to. Not to mention, the setup was different for each of the three main operating systems (Windows, macOS, and Linux).

I started searching for a better, more sustainable solution and I stumbled upon Segger Embedded Studio (SES). SES is a professional cross-platform IDE for ARM Core microcontrollers, including the nRF5x series chipsets from Nordic Semiconductor. Best of all, Nordic and Segger had recently announced a partnership where nRF developers can get a FREE commercial license to use SES without limitations! You can read more about the partnership announcement here (https://www.segger.com/news/segger-embedded-studio-ide-now-free-for-nordic-sdk-users/).

In this blog post, I will go over:

- How to set up and install Segger Embedded Studio for nRF development
- How to enable logging and debug messages
- How to configure the CMSIS Configuration Wizard plugin for easier sdk_config.h configuration

# Segger Embedded Studio (SES)

Segger (https://www.segger.com/) is a well-known company in the embedded space that provides software, hardware, and development tools for embedded systems. Segger Embedded Studio (SES) is a professional cross-platform IDE for ARM Core microcontrollers. It compares to other professional IDEs in the space including ARM Keil and IAR (LINKs). Some of the advantages of SES include:

- It provides a turnkey solution for development on the nRF52 series platform with seamless integration for on-target (on the device) debugging using J-Link.
- It has integrated GCC C/C++ and Clang/LLVM compilers.
- It provides cross-platform support: runs on Windows, Linux, and macOS.
- There are no restrictions on code size and compiler optimization.
- It integrates analysis tools for memory use, static code analysis and more.
- It has tools for importing projects from other IDEs such as Keil and IAR.

For nRF5x developers, this is the most cost-efficient solution and provides features that compete with other professional IDEs such as Keil and IAR Embedded Workbench for ARM IDEs.

# Setup for Windows, Linux, and macOS

Let's go through the steps for setting up SES for nRF development on the different operating systems (Windows, macOS, and Linux). The steps below are pretty much identical for the three operating systems with minor differences.

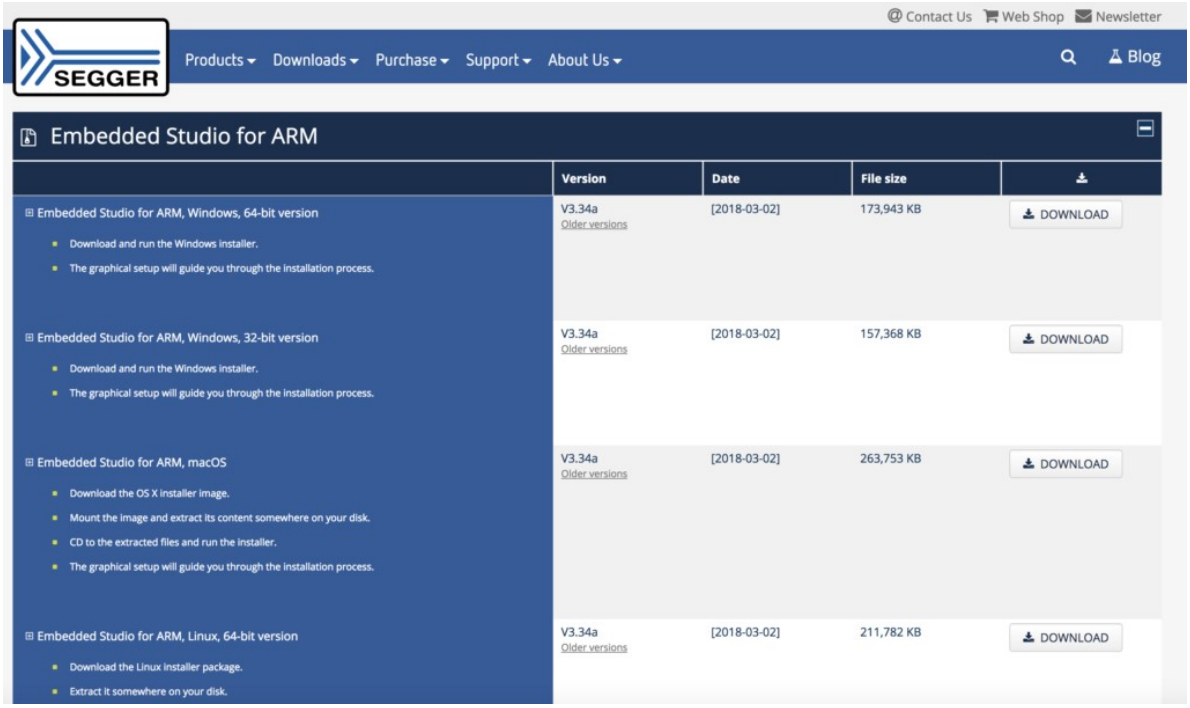
# Detailed Steps for Setting Up SES

(https://www.no

1. First, download Segger Embedded Studio from their website (look for the download that matches your operating system):

   https://www.segger.com/downloads/embedded-studio

   (https://www.segger.com/downloads/embedded-studio/)

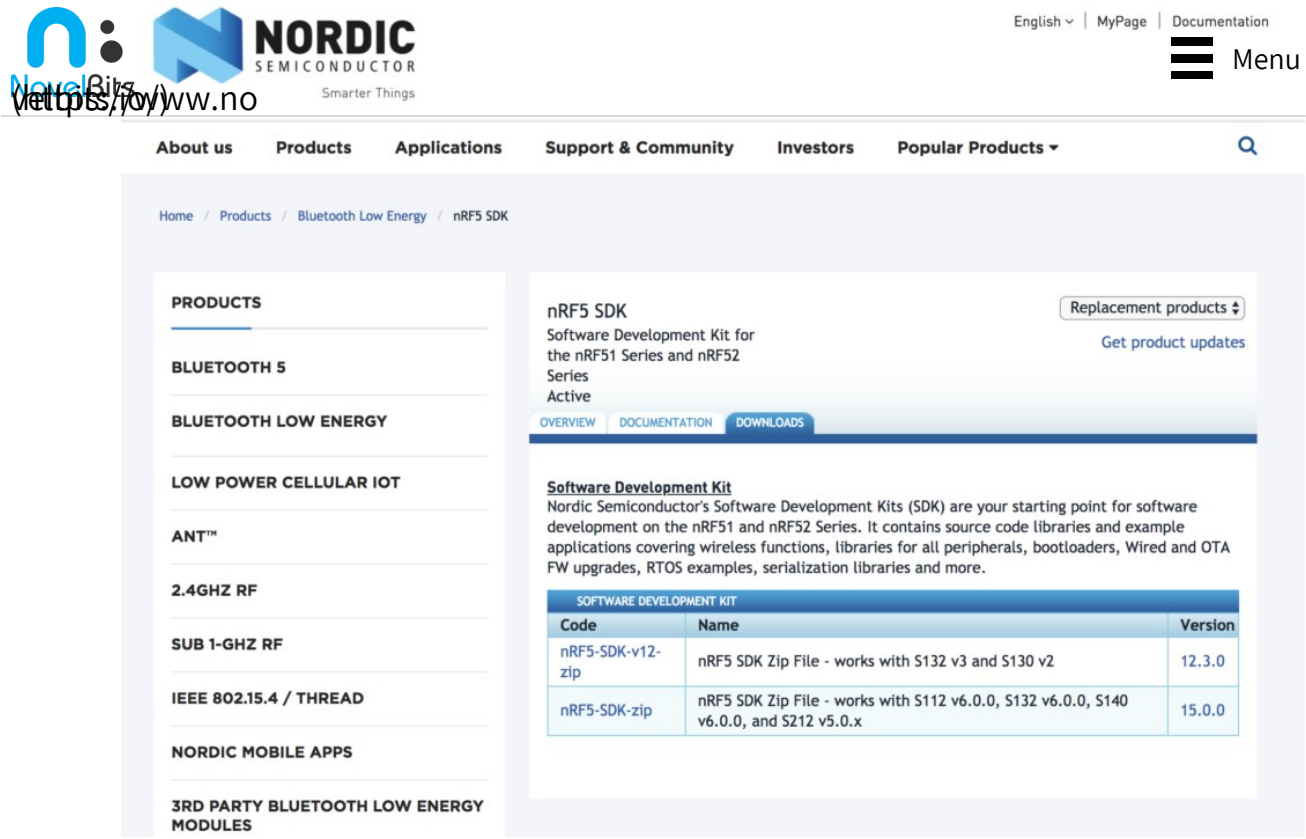



   (https://www.novelbits.io/wp-content/uploads/2018/04/SES-Download-website.png)

2. Launch the downloaded installation file and complete the installation.
3. Download the latest nRF5x SDK:

   https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF5-SDK

   (https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF5-SDK)




(https://www.novelbits.io/wp-content/uploads/2018/04/nRF5-SDK-Download.png)

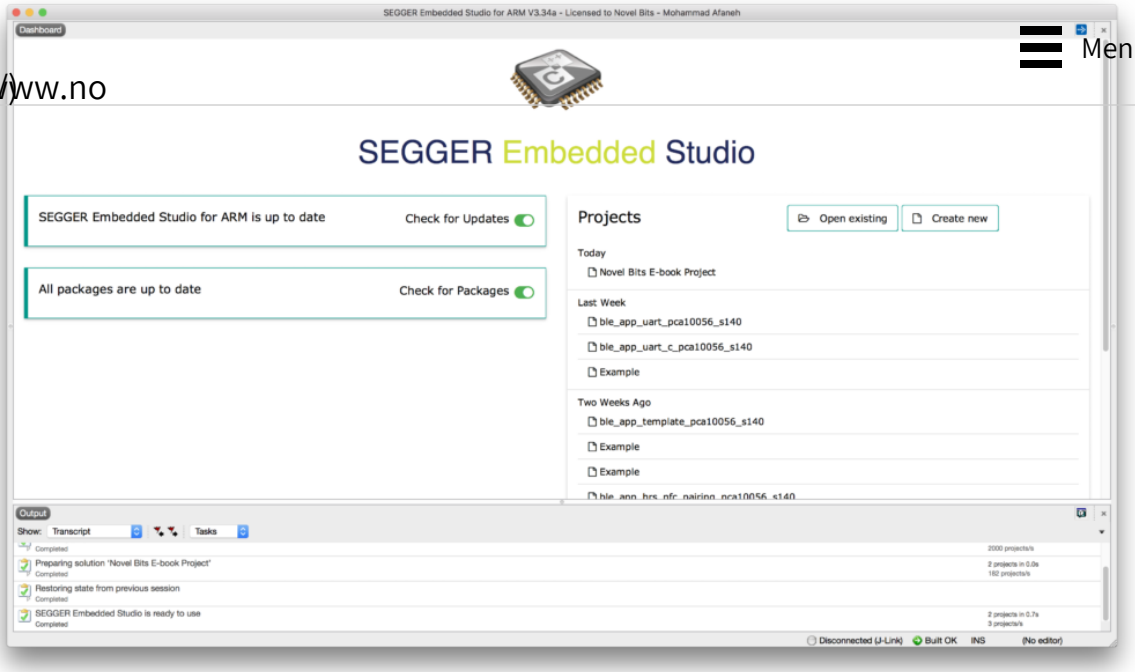
4. Unzip the SDK file to a known location on your computer.
5. Download the latest version of Java for your computer (for macOS and Windows):

   https://java.com/en/download/ (https://java.com/en/download/)

   For Linux, run the following command from a Terminal window:

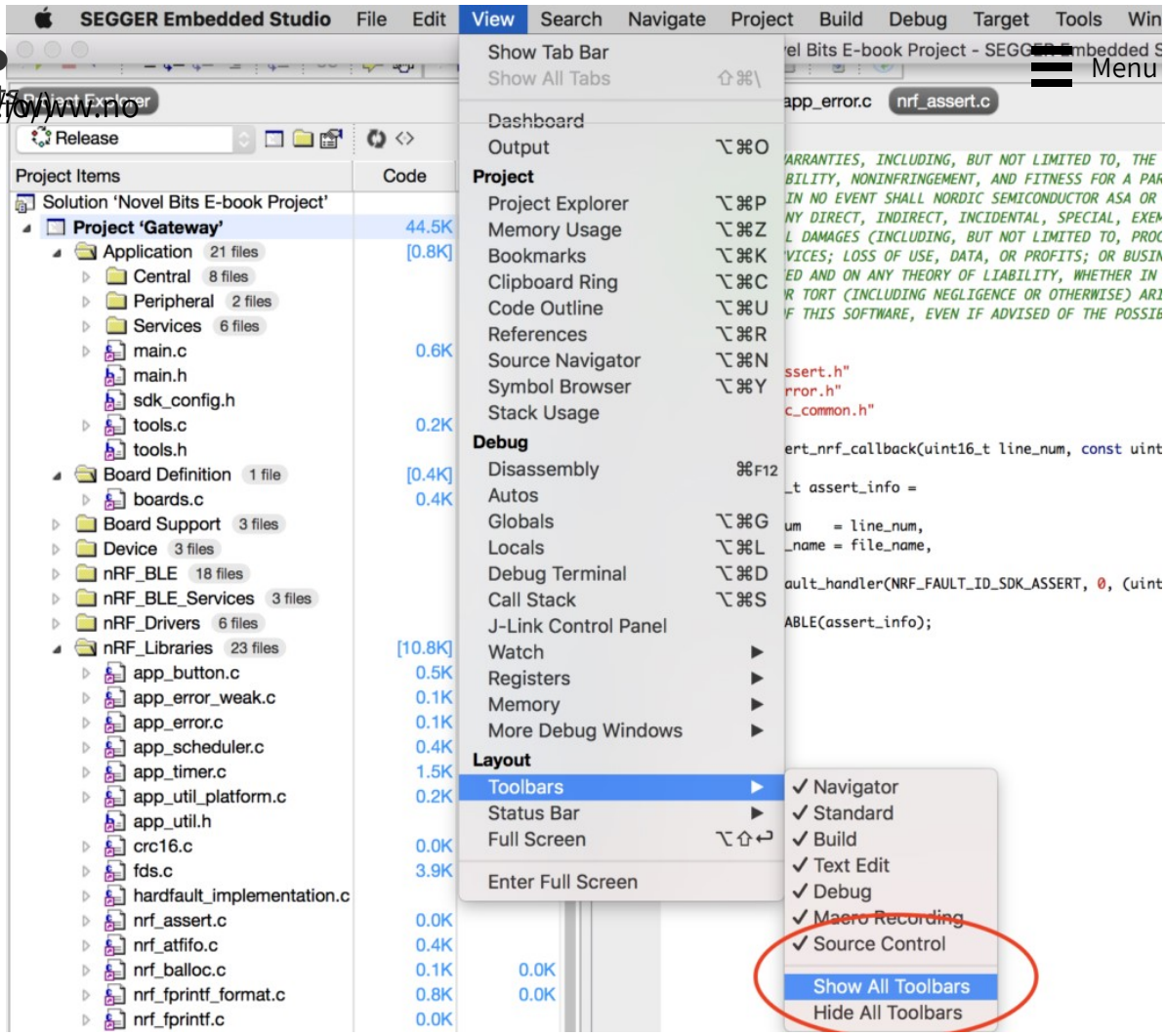
   **sudo apt-get install openjdk-8-jre**

   *We will need this for a plugin that runs within SES (the CMSIS Configuration Wizard).*
6. Once you have Java and SES installed, go ahead and launch SES.



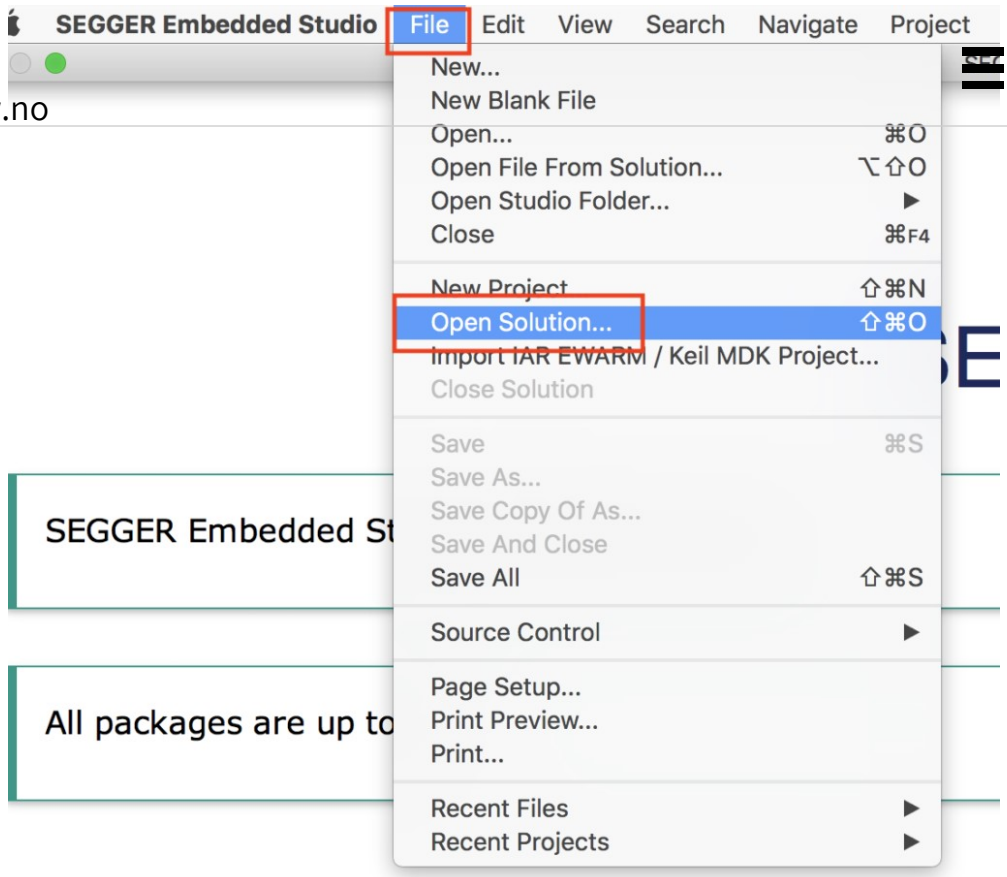

(https://www.novelbits.io/wp-content/uploads/2018/04/SES-screenshot-1.png)

7. Now that SES is launched, we will open one of the examples provided with the SDK under the **/examples** folder.
8. But before that, let's enable showing all toolbars in SES to have a better user interface and to be able to perform operations more efficiently:




(https://www.novelbits.io/wp-content/uploads/2018/04/SES-Show-all-toolbars.png)

9. Now let's open the example named **ble_app_hrs**. Navigate to File → Open Solution.

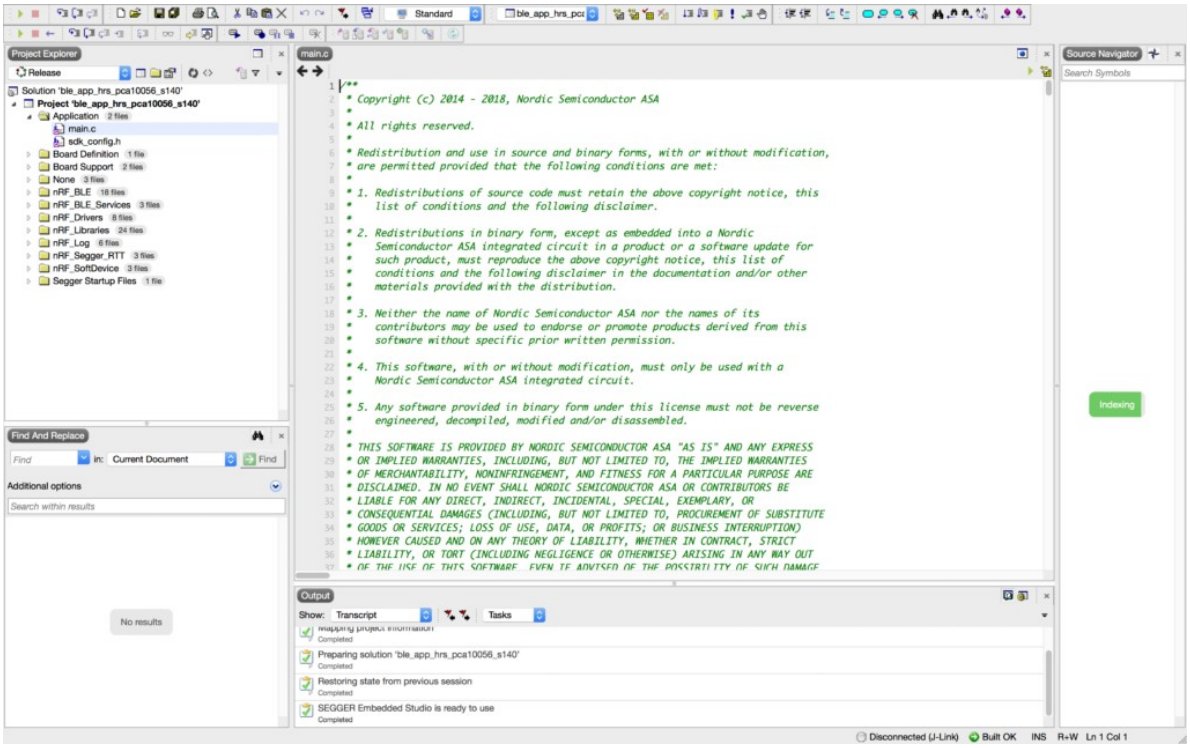



(https://www.novelbits.io/wp-content/uploads/2018/04/SES-Open-Solution-2.png)

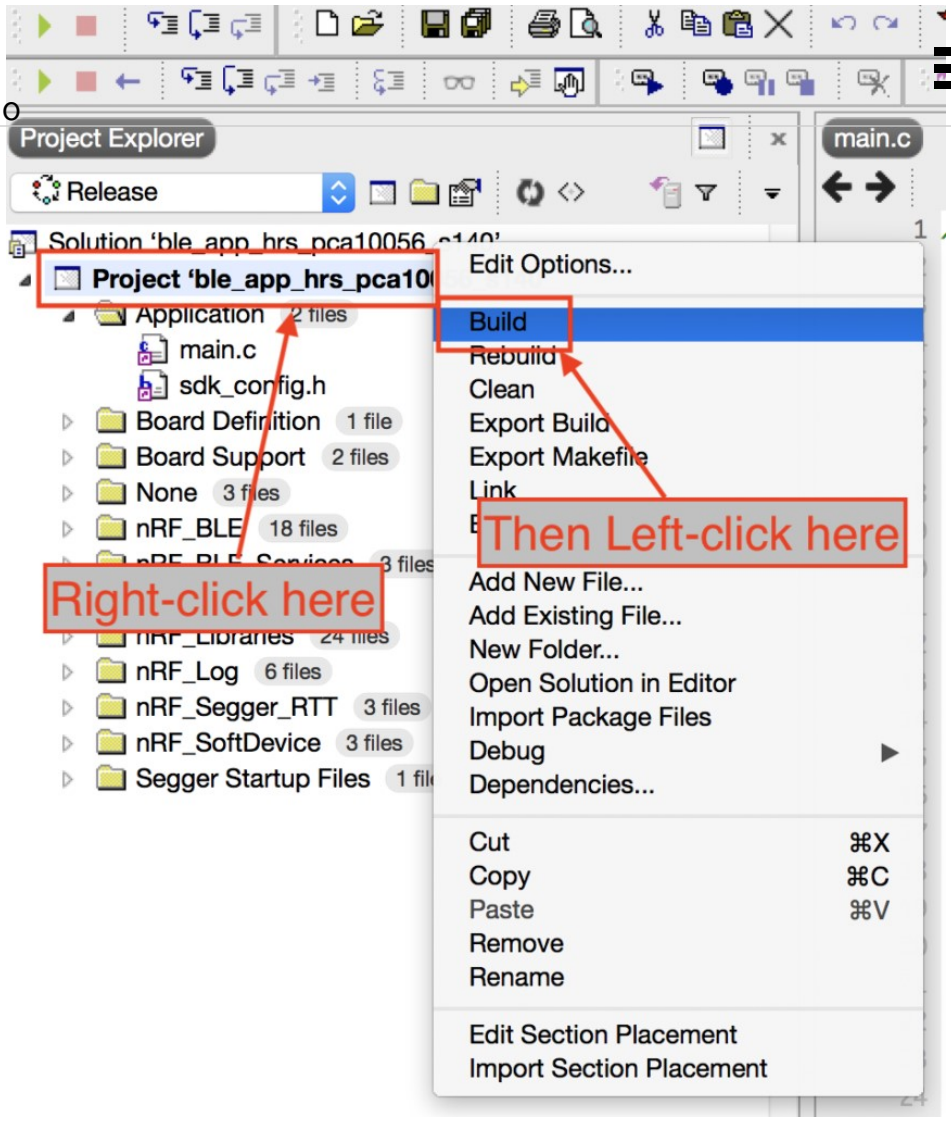
10. Navigate to this folder, and then select the folder for the nRF5 board that you are using. For this example, I'll be using the nRF52840 Preview Development Kit which corresponds to the **PCA10056** folder.
11. Inside the folder **PCA10056** (or whatever folder you need to choose for your development board), you will find a folder named **s1xx** which corresponds to the SoftDevice (http://infocenter.nordicsemi.com/topic/com.nordic.infocenter.softdevices52/dita/nrf52/softdevices.html?cp=2_3) used for the development kit's chipset. The SoftDevice is Nordic's proprietary BLE stack (it also includes interfaces for the different hardware components of the chip).

12. Inside the SoftDevice folder (**s140** in my case), there's a folder named **ses**. Open (https://www.novelbits.io/wp-content/uploads/2018/04/SES-hrs-example-open.png) this folder and click on the ***.emProject** file. Click **Open.**

13. Once you have the Solution open, you'll be presented with the following screen:

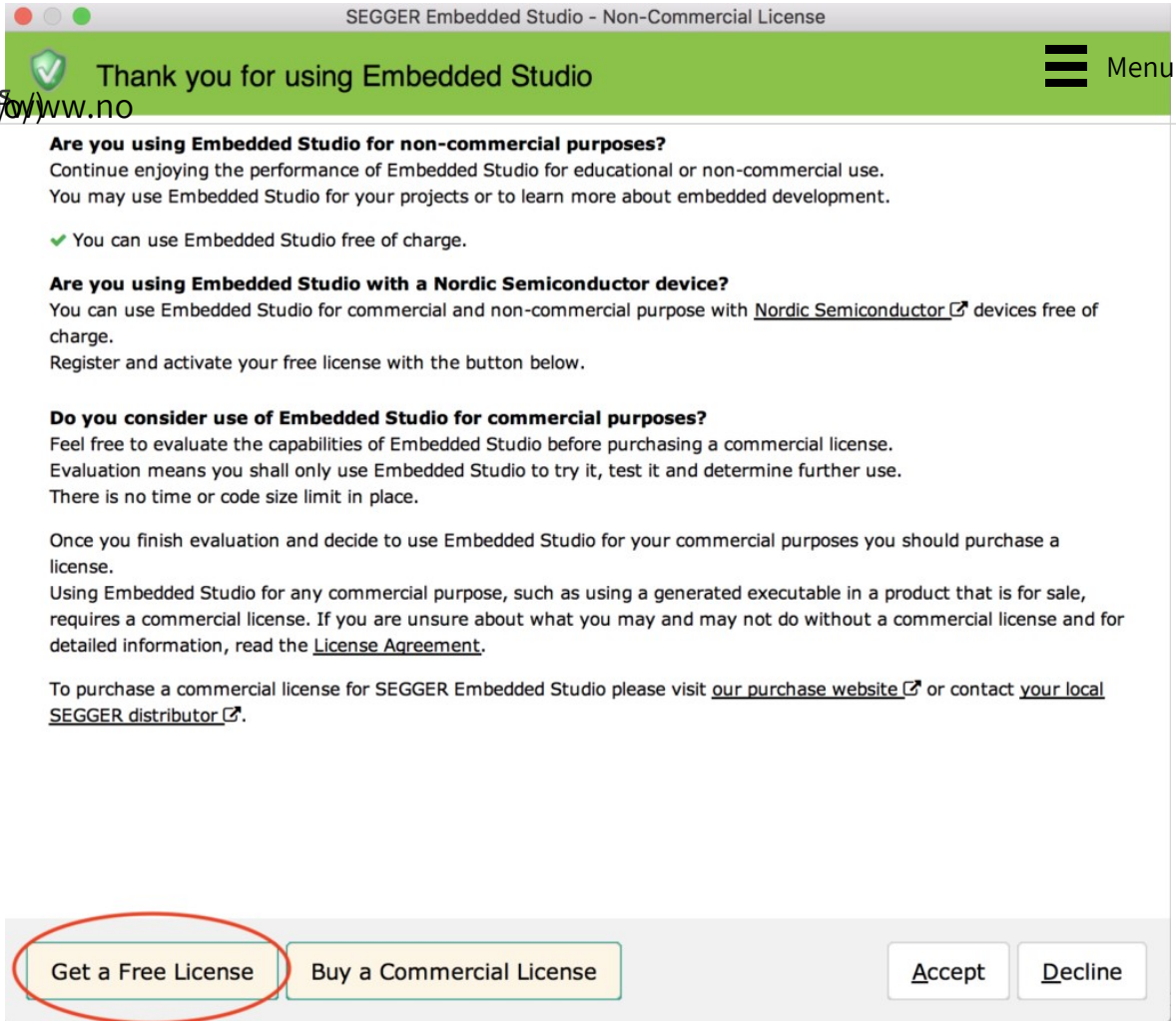



14. Now you can attempt to Build the project:



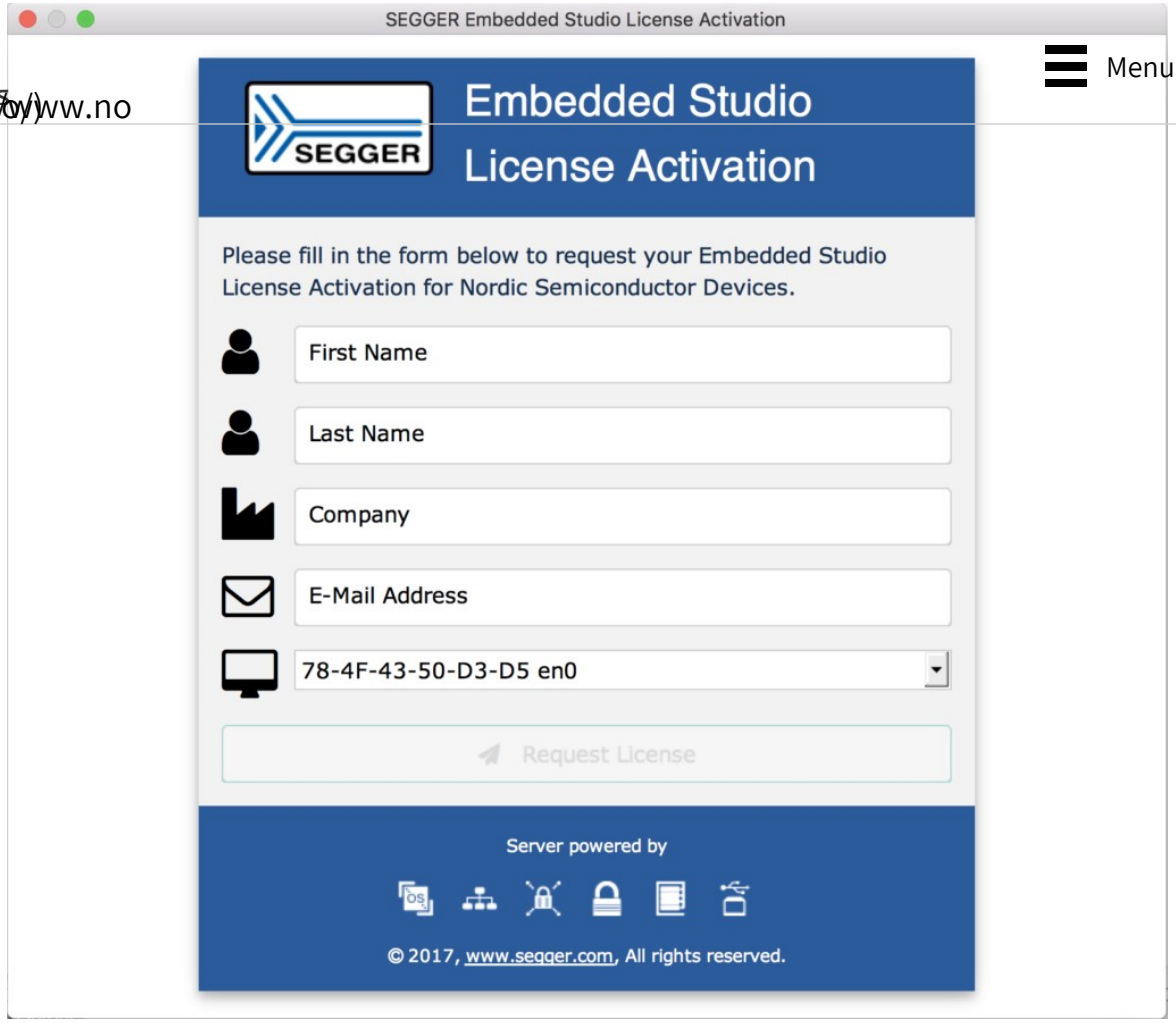

(https://www.novelbits.io/wp-content/uploads/2018/04/SES-build-hrs-example.png)

15. If you do not already have an SES license, you will be prompted to register for a FREE commercial license (*for nRF devices only*) before you are able to build:



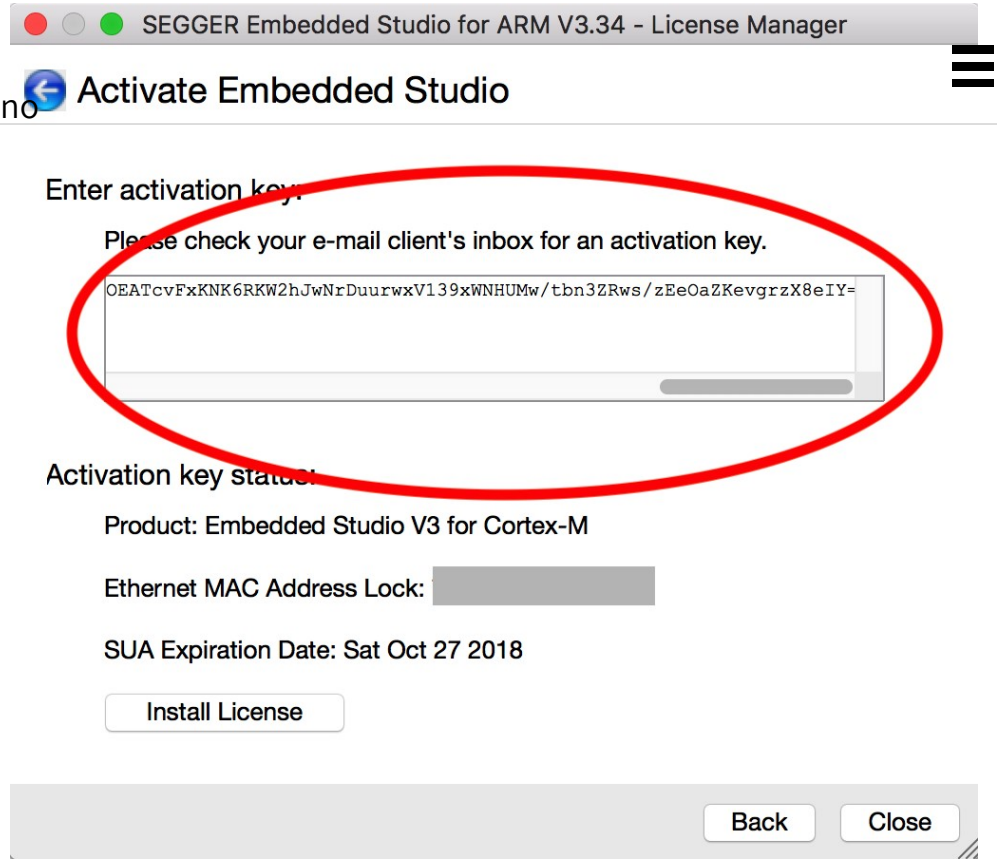

(https://www.novelbits.io/wp-content/uploads/2018/04/SES-FREE-License.png)

16. Click the **Get a Free License** option, and then enter your information needed:



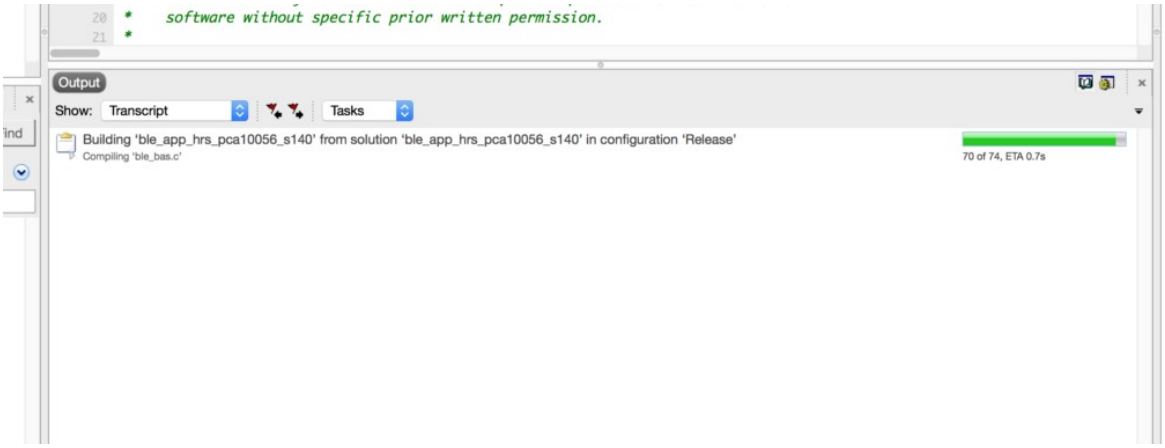

(https://www.novelbits.io/wp-content/uploads/2018/04/SES-License-activation.png)

17. Hit **Request License.**
18. You should then receive the License information in your email.
19. Copy the License key and Paste it into the Activation field:

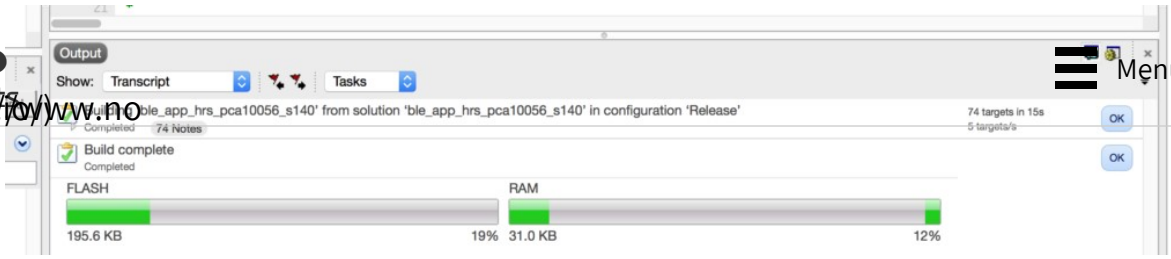



(https://www.novelbits.io/wp-content/uploads/2018/04/SES-License-enter-1.png)

20. Once you enter the License key, click **Close.**
21. The Project should now start building and (eventually) build successfully:



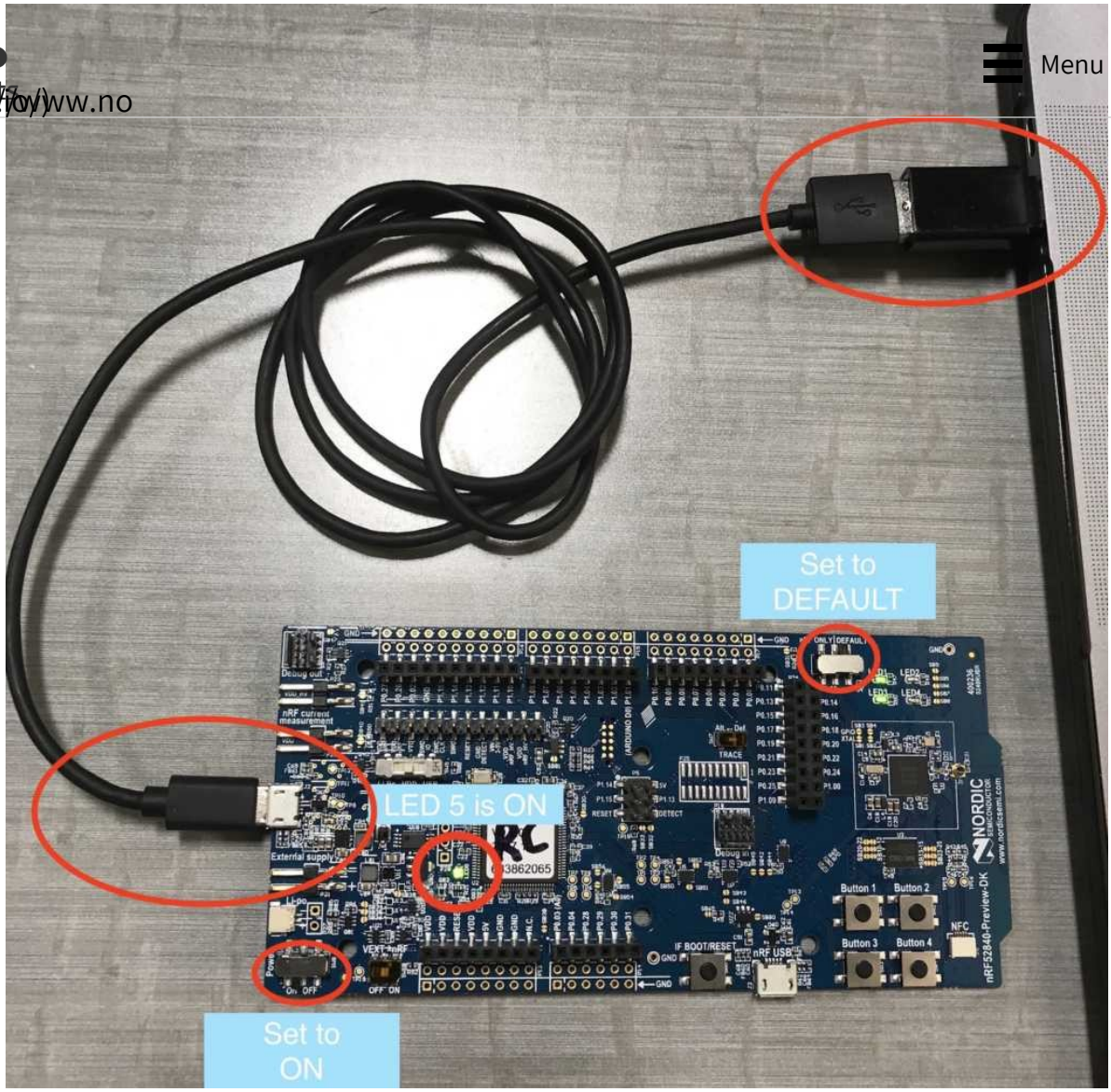

(https://www.novelbits.io/wp-content/uploads/2018/04/SES-building.png)




22. Now you should be able to flash and run the application on your nRF development kit (nRF52840 in my case).
23. Make sure you have the development kit connected to your computer with the following configuration:

    **Note:** ***I am using a MacBook Pro that does not have a regular USB connection, so the USB cable is connected to a USB-C adapter as you will see in the picture. In your case, you may have the USB cable connecting directly to your computer.***

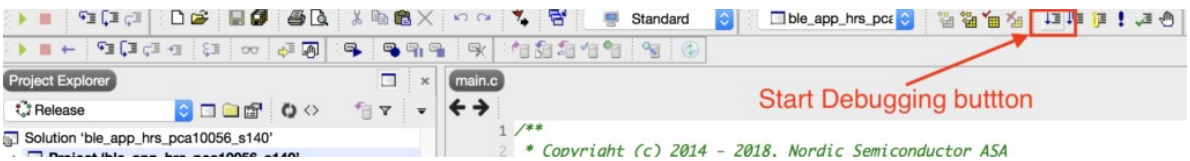
(https://www.no




(https://www.novelbits.io/wp-content/uploads/2018/04/nRF52-dev-kit-computer-connection.jpg)

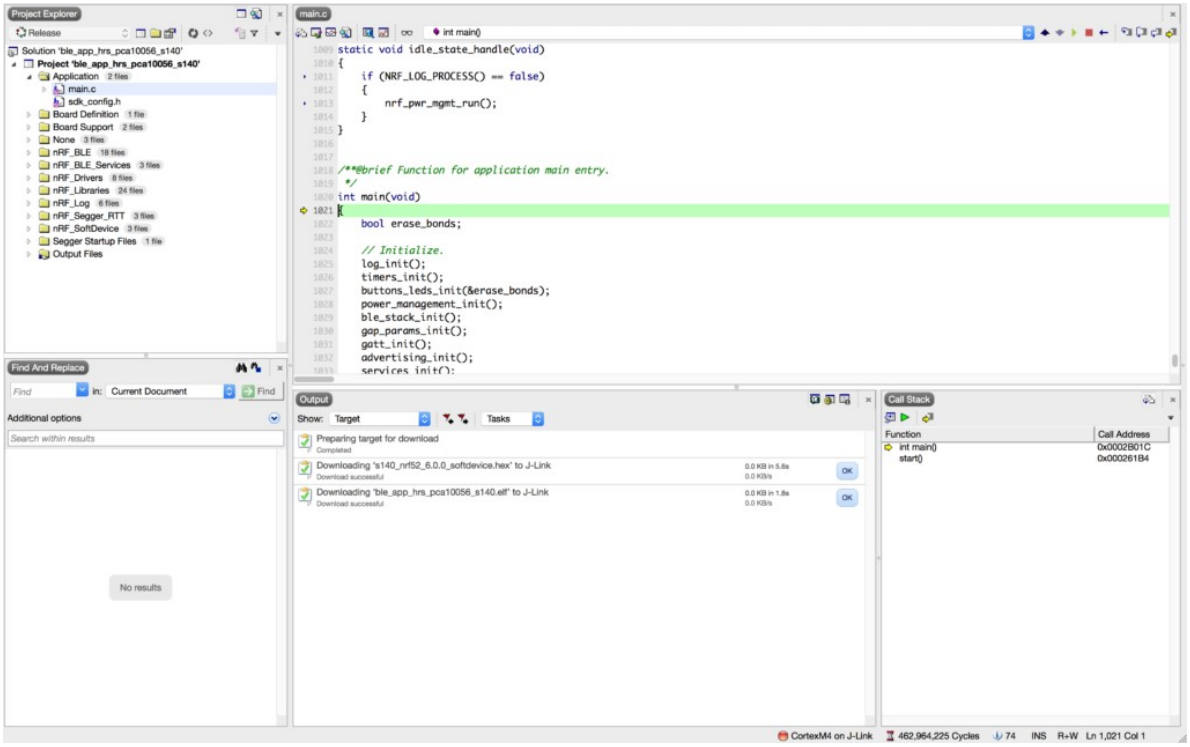
24. Once you have the development kit connected to the computer, you can run the application on the board.

    Click on the **Start Debugging** button:



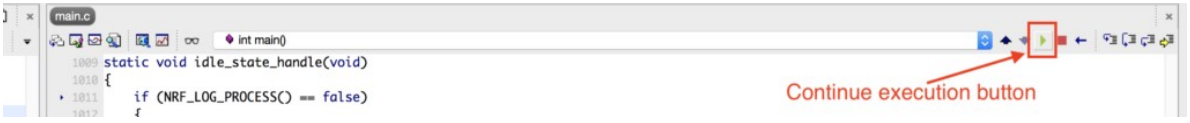
(https://www.novelbits.io/wp-content/uploads/2018/04/SES-Start-debugging.png)

25. When the application is flashed to your development kit, it will stop at a breakpoint in **main()**:




(https://www.novelbits.io/wp-content/uploads/2018/04/SES-Started-debugging.png)

26. Continue execution and run the application by clicking the **Play** button:




(https://www.novelbits.io/wp-content/uploads/2018/04/SES-continue-execution.png)

27. To get a better idea of the state of the running application, we'll enable logging so we can see any debug messages.

In order to do that, we'll first set up a plugin that helps enable different settings

within the SDK.

(https://www.no

## The sdk_config.h file

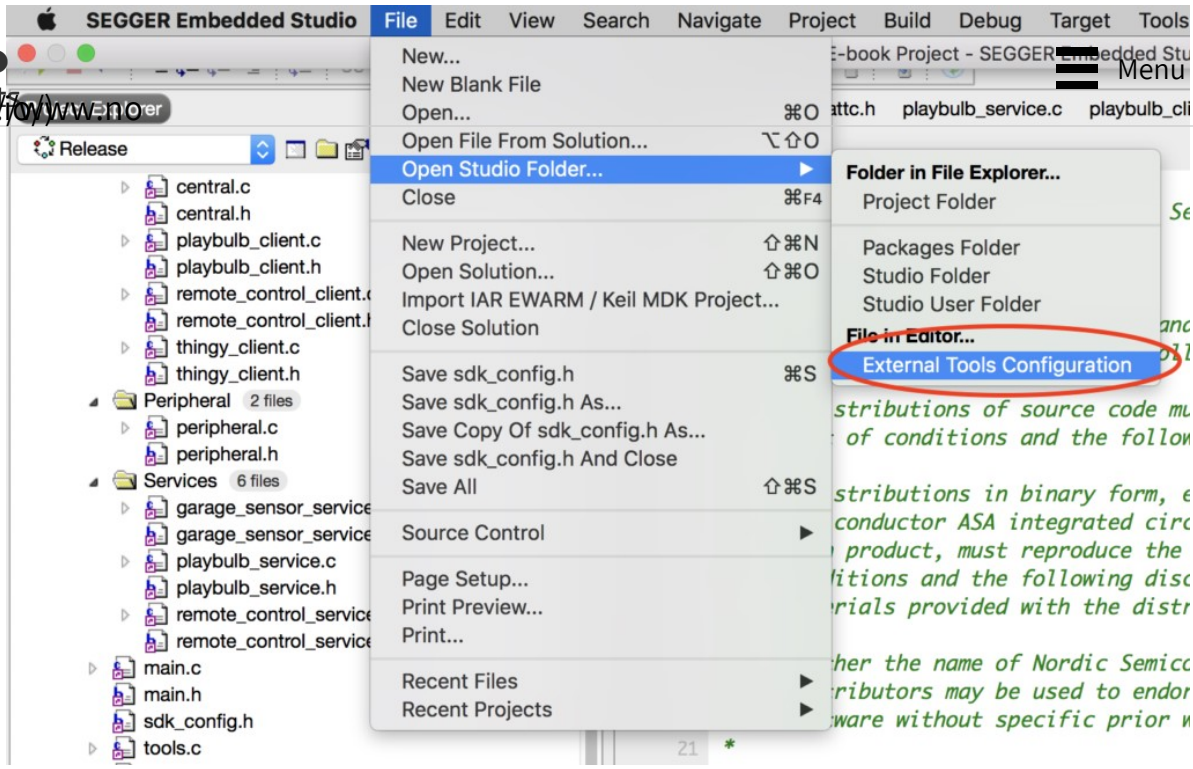
The **sdk_config.h** file includes different configurations and macro definitions that allow you to customize the SDK and application to enable various features. This file includes configurations such as:

- Flags to enable certain modules to be included in an application (e.g. Central role, Peripheral role, Battery Service Client, and more).
- Flags to configure and enable different hardware peripherals.
- Flags to configure and enable logging and debugging.
- Many others…

The file can be modified directly from the SES IDE Editor, using any generic text editor or via a plugin called the **CMSIS Configuration Wizard** (a Java-based add-on). This tool enables you to edit the options within the sdk_config.h file from a user-friendly interface instead of editing it within the raw source code. *You will only have to do this step once.*

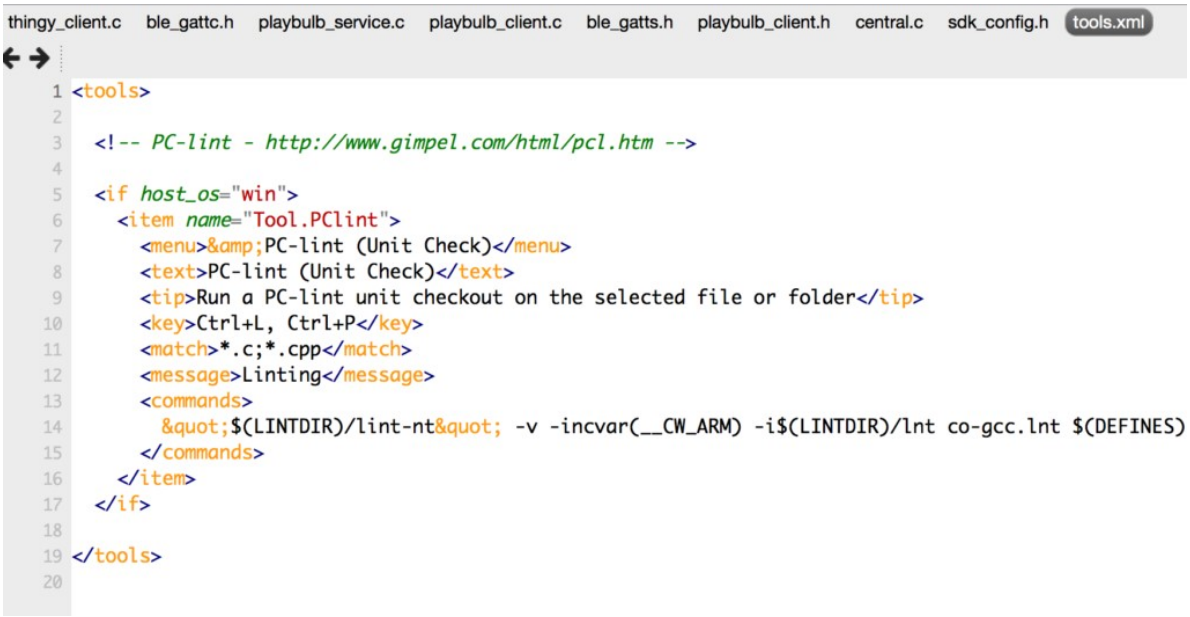
To enable the CMSIS Configuration Wizard, follow these steps:

1. Navigate to File → Open Studio File → External Tools Configuration




(https://www.novelbits.io/wp-content/uploads/2018/04/SES-CMSIS-Wizard-External-Tools-Config.png)

2. Once you've clicked on **External Tools Configuration**, the **tools.xml** file should now be open:



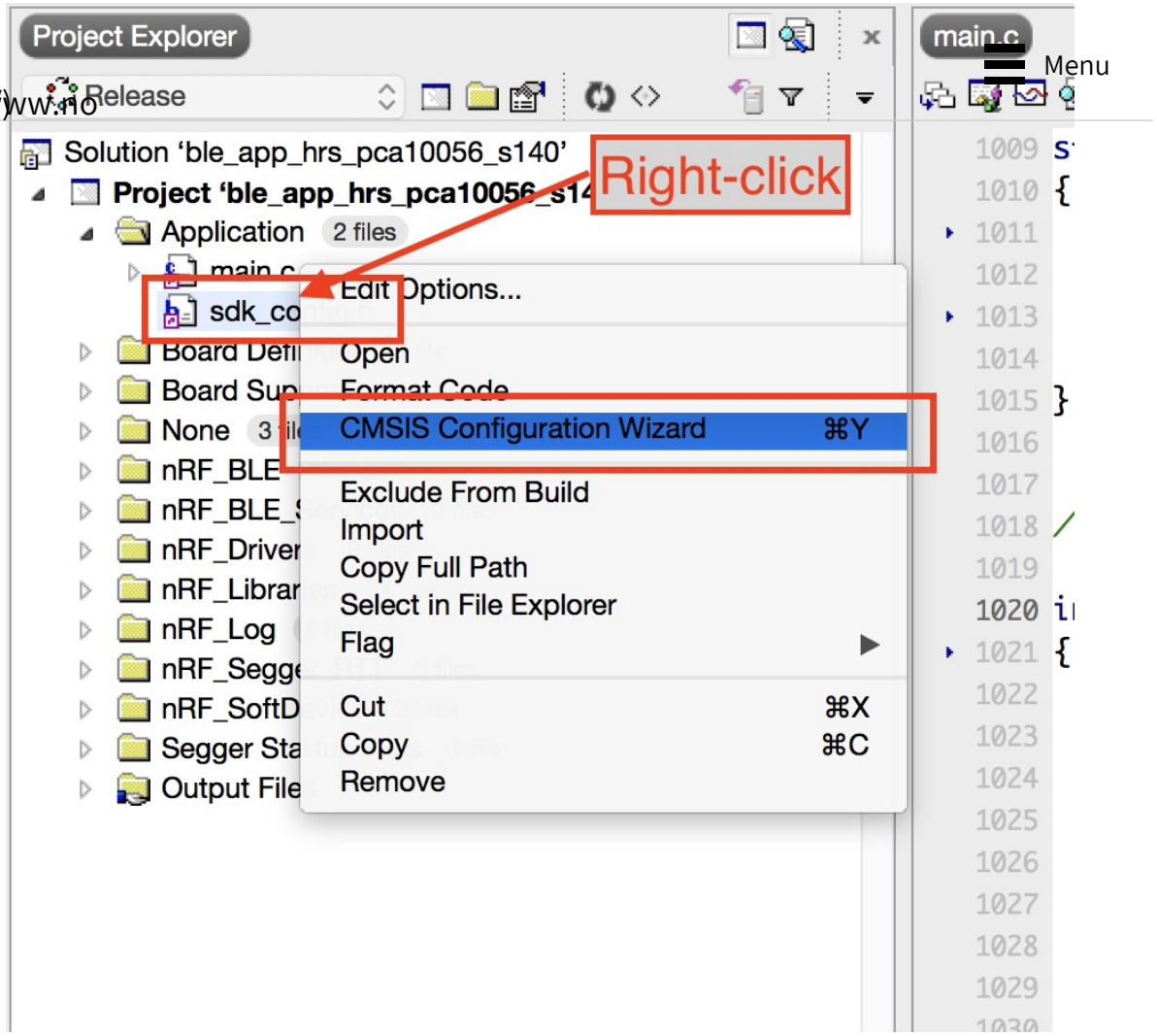

(https://www.novelbits.io/wp-content/uploads/2018/04/SES-tools-xml.png)

3. After you have the tools.xml file open, add the following XML code before the (https://www.novelbits.io/wp-content/uploads/2018/04/tools-xml-code.jpg) **</tools>** tag:

```xml
<item name="Tool.CMSIS_Config_Wizard" wait="no">
    <menu>&amp;CMSIS Configuration Wizard</menu>
    <text>CMSIS Configuration Wizard</text>
    <tip>Open a configuration file in CMSIS Configuration Wizard</tip>
    <key>Ctrl+Y</key>
    <match>*config*.h</match>
    <message>CMSIS Config</message>
    <commands>
      java -jar &quot;$(CMSIS_CONFIG_TOOL)&quot; &quot;$(InputPath)&quot;
    </commands>
</item>
```

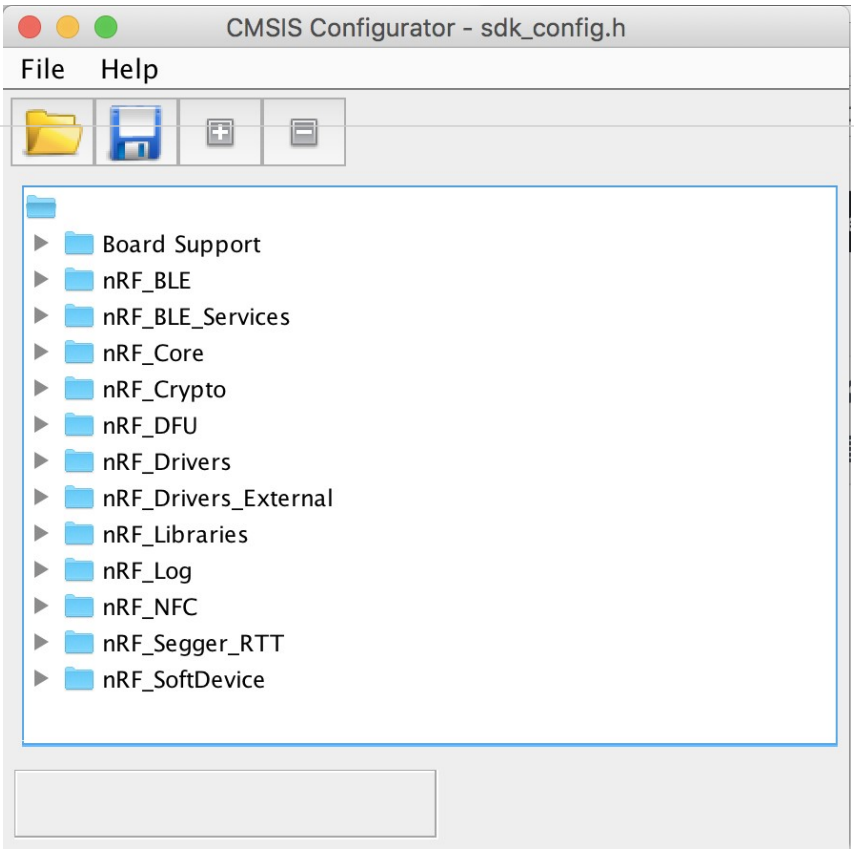
4. Make sure to place it outside the **<if host_os=....>...</if>** statement:


5. Now, quit and restart SES.
6. After SES restarts, you can now right-click on the **sdk_config.h** file within the **Application** folder and you will see the option “CMSIS Configuration Wizard”:




(https://www.novelbits.io/wp-content/uploads/2018/04/CMSIS-Configuration-Wizard-option-1.png)

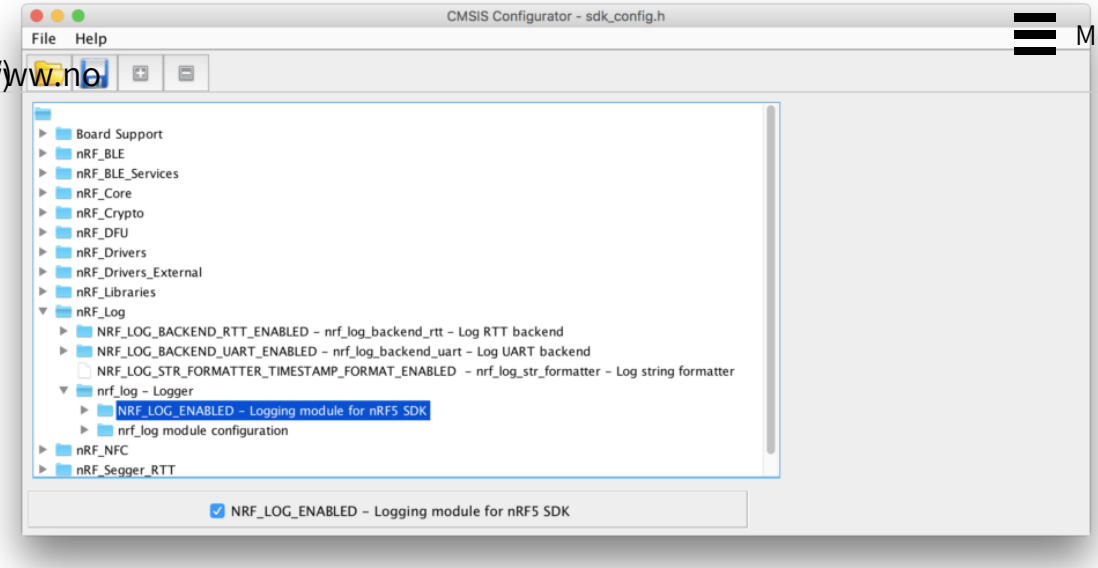
7. Click on “CMSIS Configuration Wizard” and it will present you with the following window:




(https://www.novelbits.io/wp-content/uploads/2018/04/CMSIS-Configuration-Wizard.png)

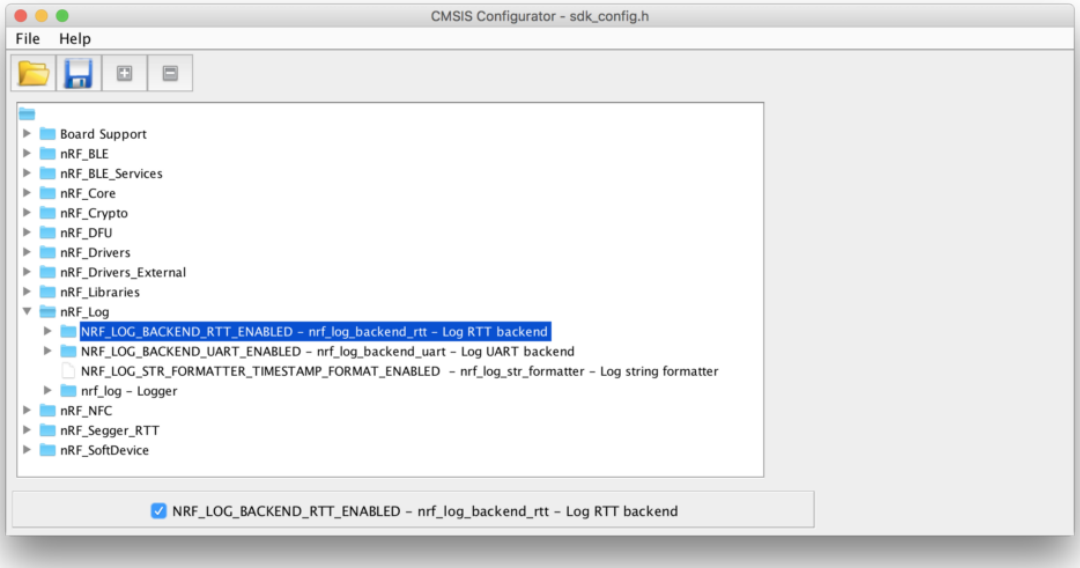
8. From here, we can enable logging. Make sure you have the following options enabled (checked)

   **NRF_LOG_ENABLED**



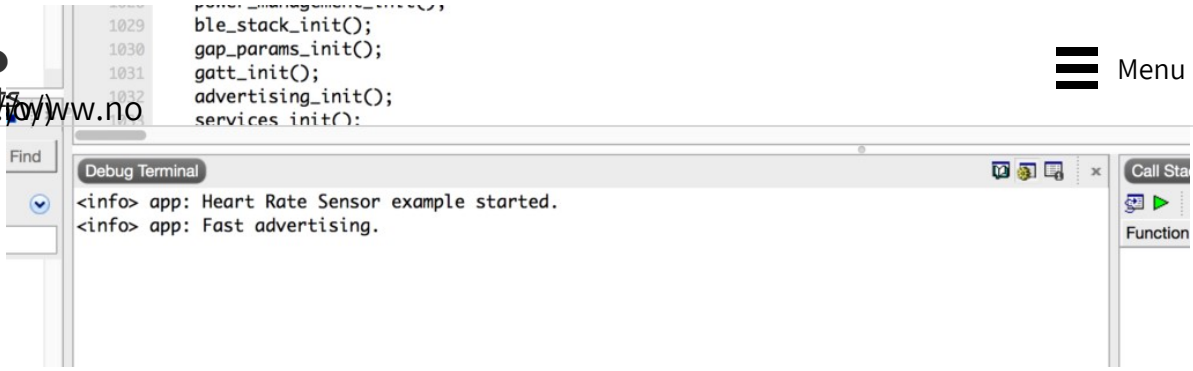

(https://www.novelbits.io/wp-content/uploads/2018/04/nRF-Log-Enabled.png)

**NRF_LOG_BACKEND_RTT_ENABLED**




(https://www.novelbits.io/wp-content/uploads/2018/04/RTT-enabled.png)

9. Now, rebuild the application, and start debugging again. You should now see debug messages displayed in the **Debug Terminal** window:




(https://www.novelbits.io/wp-content/uploads/2018/04/SES-Debugging.png)

Now, we're able to debug the application and see debug messages!

# Useful nRF development tips and tricks

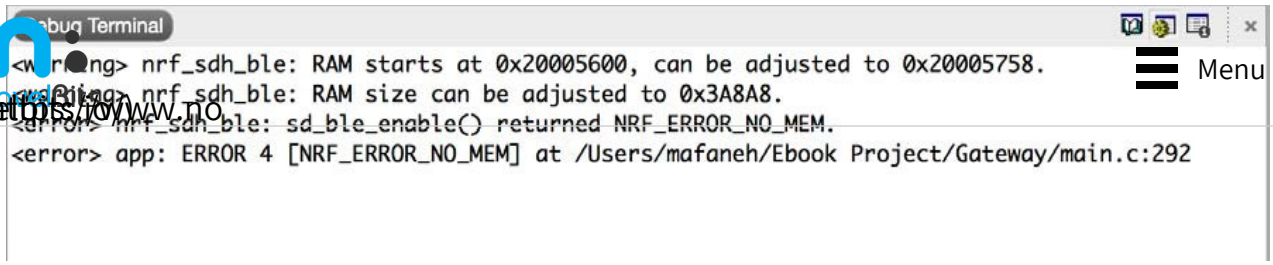
## RAM Size and RAM Start

For some of the modifications to the sdk_config.h file, the RAM size used by the SoftDevice will change causing the RAM offset and size of the user application to shift. This causes the application to halt and not work properly (sometimes causing an early exception). Fortunately, the SoftDevice prints out an error message in the debug messages log indicating the correct RAM start and size for the application. Make sure you have logging enabled with the following settings:

- NRF_LOG_DEFAULT_LEVEL set to ≥ 3.
- NRF_SDH_BLE_LOG_ENABLED set to 1.
- NRF_SDH_BLE_LOG_LEVEL set to ≥ 2.
- Logger configured to use the Segger RTT backend.

All of these can be set from the CMSIS Configuration Wizard plugin we mentioned earlier.

The error message will look like this:

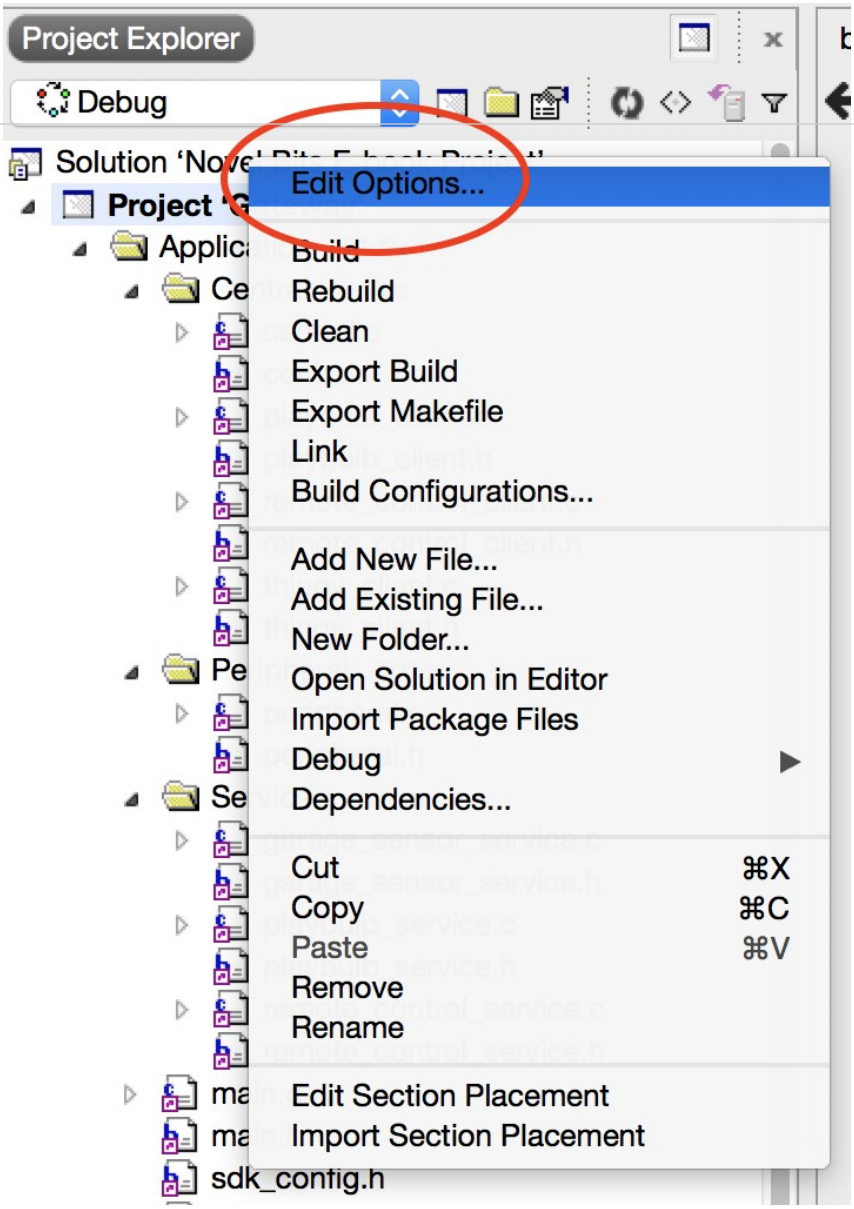



(https://www.novelbits.io/wp-content/uploads/2018/04/RAM-Size-Error.jpg)

The error can be fixed by modifying the RAM start and size of the application from within SES:

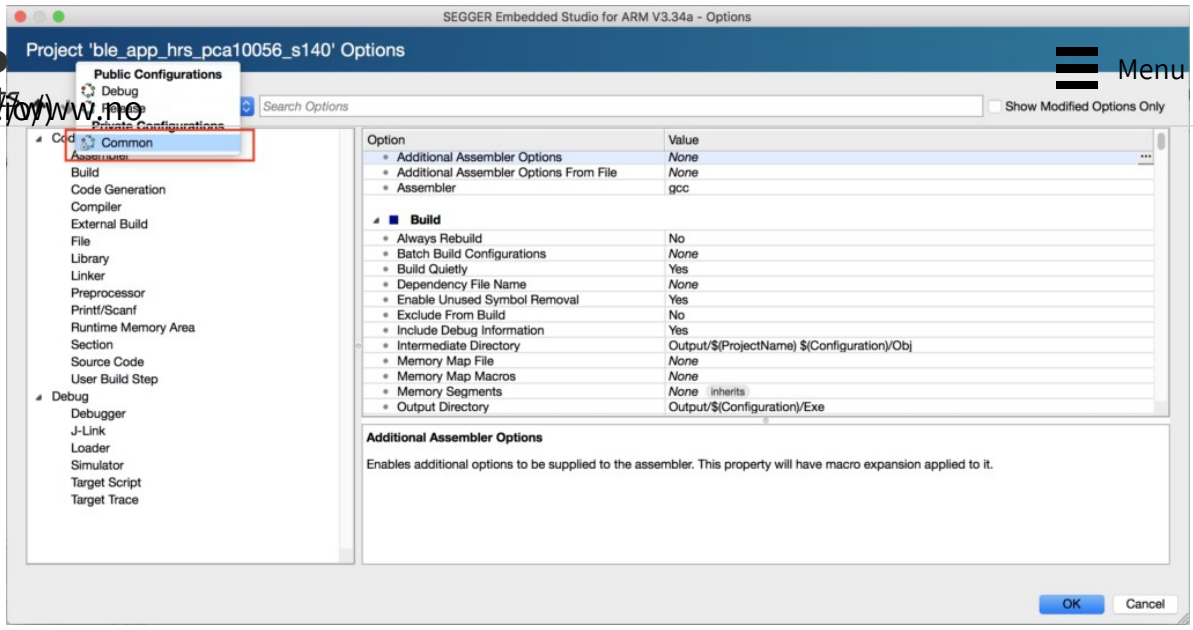
- Right-click on the Project, then click on “Edit options”.

  **Make sure you right-click on the Project name and not the Solution.**



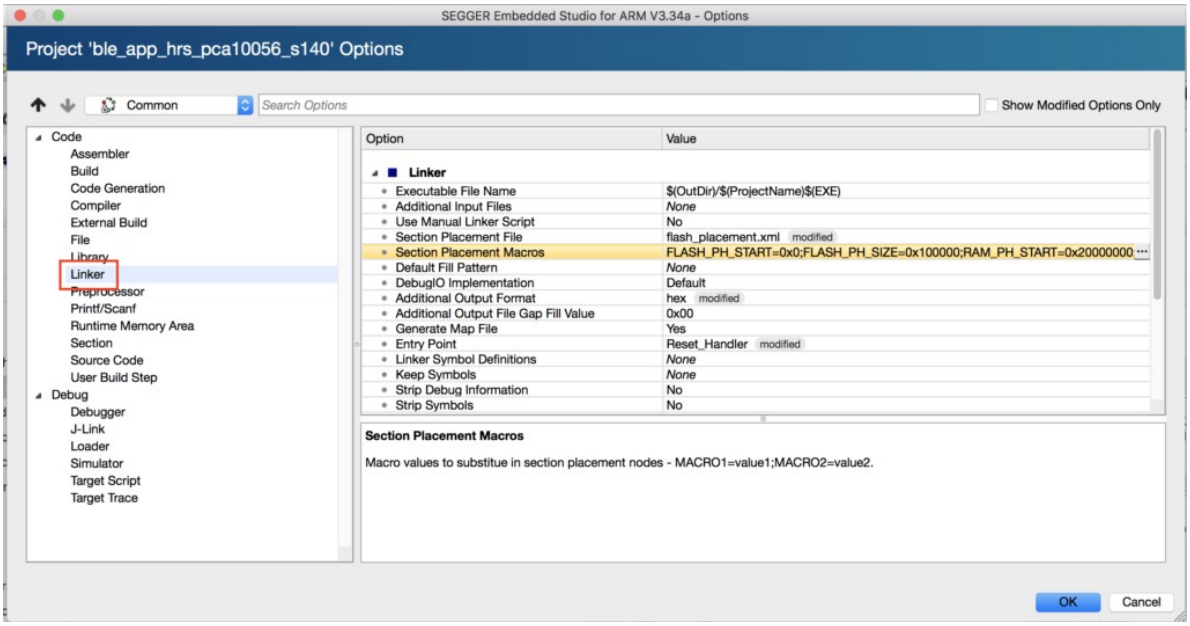

(https://www.novelbits.io/wp-content/uploads/2018/04/SES-Edit-Options.png)

- From the Configurations drop-down menu, choose “Common”:




(https://www.novelbits.io/wp-content/uploads/2018/04/SES-Common-configuration-1.png)

- Select “Linker”:




(https://www.novelbits.io/wp-content/uploads/2018/04/SES-Linker-option.png)

- Double click on “Section Placement Macros”, then modify the RAM start and RAM size to match the values provided by the SoftDevice in the Debug Terminal

output:

(https://www.novelbits.io/wp-content/uploads/2018/04/SES-Section-placement.png)

- Once you change the RAM Size and Start settings, go ahead and rebuild the application and flash it to your target development board.
- The application should now run correctly and it should not show any debug messages indicating a RAM Size/Start error.
- Here are a couple of YouTube videos by Nordic Semiconductor that are helpful to better understand the different **Build Options** of a project:
    - SES Build Configurations video (https://youtu.be/315-zJJ0i34)
    - SES Build Settings video (https://youtu.be/o_H1USWAM18)

## Projects and Solutions

A **Solution** in SES is a grouping of one or more **Projects.** A Project is a singular application that can be compiled and flashed to your device, and they contain the source files that are used within the application. Solutions, on the other hand, are

used to group related Projects together while still being able to switch between them within the same view. Other benefits of including multiple Projects within a single Solution is the ability to build all the included Projects at once.

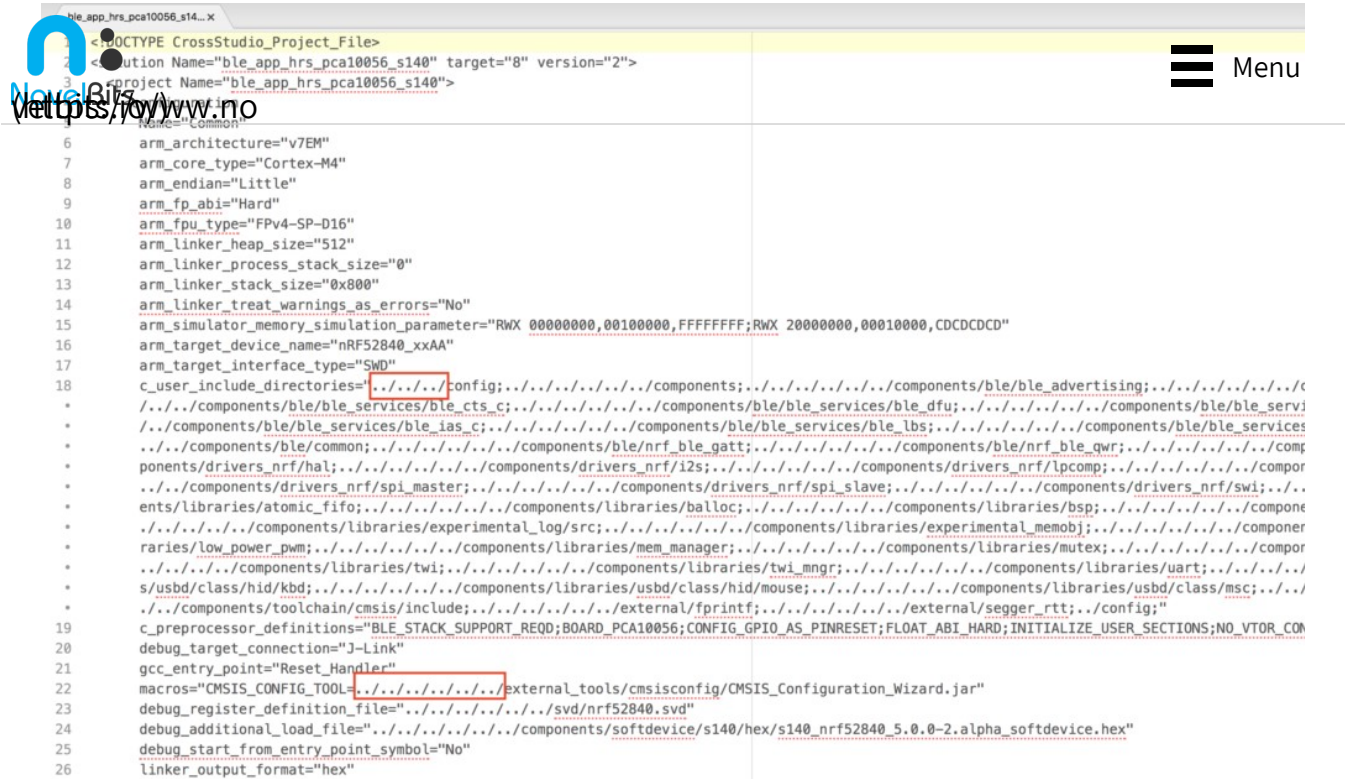
A Solution is stored as a **.emProject** file in your filesystem. Projects, on the other hand, do not have separate files. Instead, they are contained within the Solution file.

The one thing that I've found interesting is that all of Nordic's example Projects include the source files relative to their root directories in the filesystem. This causes issues when trying to move that folder (e.g. outside of the SDK folder) to another location in your filesystem. The only solution that I've found to this problem is to manually edit the ***.emProject** file and switch from using relative paths to absolute paths.

To do this, open the Solution (.emProject) file in an editor of your choice and change all the paths to absolute paths.

You could do this by searching for and placing certain length strings like "……" with the base SDK dir.

For example, looking at the Solution file for the **ble_app_hrs** example, we notice that all paths listed are relative paths instead of absolute:




(https://www.novelbits.io/wp-content/uploads/2018/04/emProject-editor.png)

Yes, it will be a pain to change all these paths to absolute, but I recommend doing this for a template Solution (or a Solution for the example that you will be using as your base Solution), and then re-using that Solution for the different applications that you will be developing from thereon.

# Summary

In this post, we covered how to set up Segger Embedded Studio for nRF development as well as a couple of tips and tricks. The steps we went through should work on all the major operating systems (Windows, macOS, and Linux). As mentioned previously, the best part is that you can get a **FREE License** for SES when developing nRF applications. This gives you access to a professional cross-platform IDE that compares to other (costly) IDEs at no charge!

If you're ready to dig deeper into BLE and Bluetooth 5, and learn the basics through detailed step-by-step tutorials, then check out my recently released book. To learn more about the book as well as access **the FREE sample content** visit:

**Bluetooth 5 & Bluetooth Low Energy: A Developer's Guide**

(https://www.novelbits.io/bluetooth-5-developers-e-book/)

In an upcoming blog post, we'll go through building a simple nRF BLE application and learn how to debug it using SES. We'll also go over some tips and tricks that can help you develop and test BLE applications faster.

If you've enjoyed this blog post and found it helpful, please share it with your co-workers, friends, or anyone who may find it useful.

---

*Posted in BLE technology (https://www.novelbits.io/category/ble/), Tutorials (https://www.novelbits.io/category/tutorials/) and tagged BLE development (https://www.novelbits.io/tag/ble-development/), cross-platform (https://www.novelbits.io/tag/cross-platform/), Linux (https://www.novelbits.io/tag/linux/), Mac (https://www.novelbits.io/tag/mac/), macOS (https://www.novelbits.io/tag/macos/), nRF development (https://www.novelbits.io/tag/nrf-development/), nRF52 (https://www.novelbits.io/tag/nrf52/), nRF52 development (https://www.novelbits.io/tag/nrf52-development/), Segger (https://www.novelbits.io/tag/segger/), Segger Embedded Studio*

*(https://www.novelbits.io/tag/segger-embedded-studio/), SES (https://www.novelbits.io/tag/ses/), Windows (https://www.novelbits.io/tag/windows/)*

← Bluetooth 5 speed: How to achieve maximum throughput for your BLE application (https://www.novelbits.io/bluetooth-5-speed-maximum-throughput/)

How to build the simplest nRF52 BLE Peripheral application (Lightbulb use case) → (https://www.novelbits.io/smart-ble-lightbulb-application-nrf52/)

## 16 Comments

**Joachim Willner** on April 11, 2018 at 1:16 am

Very helpfull

Reply

**Mohammad Afaneh (https://www.novelbits.io)** on April 11, 2018 at 1:17 am

Thanks, Joachim!

Reply

**Andreas Birkedal** on April 12, 2018 at 8:55 am

> Another great tutorial. Thank you, Mohammad Afaneh!

Reply

**Mohammad Afaneh (https://www.novelbits.io)** on April 18, 2018 at 2:08 pm

> Thanks, Andreas!

Reply

**Hannu Hirvonen (https://www.optiwatti.fi)** on May 9, 2018 at 6:41 am

> I have to recommend this tutorial for my co-workers 🙂

Reply

(https://www.no **Mohammad Afaneh (https://www.novelbits.io)** on May 9, 2018 at 8:27 am

Thanks, Hannu!

Reply

**Pritesh Mishra** on June 9, 2018 at 7:13 am

I really thanks for this, but i am trying to develop the BLE mesh with redbear BLE Nano V2 kit(DapLink) . So, is there any support with SES IDE.

Reply

**Mohammad Afaneh (https://www.novelbits.io)** on June 10, 2018 at 1:39 am

Thanks, Pritesh.

Unfortunately, SES does not (yet) support any debuggers other than J-Link. Refer to this GitHub issue for more information:

https://github.com/makerdiary/nrf52832-mdk/issues/9

(https://github.com/makerdiary/nrf52832-mdk/issues/9)

Reply

**Wagner Caetano** on July 11, 2018 at 9:20 pm

Great tutorial. Very helpful!!!

Reply

**Mohammad Afaneh (https://www.novelbits.io)** on July 12, 2018 at 12:53 pm

Thanks, Wagner!

Reply

**Shantanu** on July 12, 2018 at 4:54 pm

Hi Mohammad!
(https://www.no

Great tutorial. However, I'm stuck at one point. At 24) when I click on start debug, I get the error message "Cannot connect to J-Link via USB". Any way to fix that?

Reply

**Walter** on August 7, 2018 at 2:10 pm

thank you

Reply

**mwarren** on October 25, 2018 at 12:41 pm

Hi Mohammad,

Thank you for your tutorials. They are well written and documented. I am just starting a project with this board and Segger Embedded Studio was suggested for development. Zephyr was also recommended as an RTOS to use. Are you aware if it is possible to use SES with Zephyr (as well as any tutorials on setting this up :-)?

Regards,
(Mike)

Reply

**Mohammad Afaneh** on October 25, 2018 at 4:21 pm

Hi Mike,

Thanks for the feedback.

I haven't looked into Zephyr and setting it up (yet). But I will be doing this in the next few weeks and posting a tutorial documenting my findings. So stay tuned! 🙂

-Mohammad

Reply

**Bo Chen** on November 24, 2018 at 8:47 pm

Hi Mohammad,

try to enable the FLAG: “NRF_LOG_ENABLED logging module for nRF5 SDK” (I followed your instruction. However, I cannot find this option in CMSIS UI or any option that contains string “nRF5 SDK” in sdk_config.h file (with text editor). And I cannot see the log info in terminal’s output (But I do can see some output printed by “printf()” ). I guess there might be some version problems. My SDK version is 15.2.0. , SES version is 4.10a. So, how should I solve this problem? Is there any updating option for “NRF_LOG_ENABLED logging module for nRF5 SDK”? Thanks.

Best

Bo

Reply

**Mohammad Afaneh** on November 25, 2018 at 4:44 pm

Hi Bo,

I downloaded SDK 15.2.0 and I’m using SES v 4.10a as well. I can see the NRF_LOG_ENABLED macro at line #7451 (and it shows up in the CMSIS Configuration Wizard), not sure why it’s not there in your file. Can you maybe re-download the SDK and check it?

Reply

# Leave a Comment



**Comment**

**Name (required)**

**Email (will not be published) (required)**

**Website**

Submit Comment