RAK3172 Module Low Level Development Reference

Overview

The RAK3172 module uses RF System-on-Chip IC from STMicroelectronics, specifically the STM32WL series, which can be used on different LPWAN IoT devices.

STM32WL microcontrollers feature a sub-GHz radio based on Semtech SX126x to meet the requirements of a wide range of Low-Power Wide Area Network (LPWAN) wireless applications in industrial and consumer Internet-of-Things (IoT). The specific STM32WL microcontroller used in RAK3172 is the STM32WLE5CCU6.

While RAK3172 has a built-in default FW with a set of AT commands that can be interfaced to an external host like other microcontrollers, it can also be used by developing custom firmware directly on its chip using the STM32WL SDK from STMicroelectronics. Doing this approach will reduce the overall cost of the device because there will be no need for an external microcontroller but with the extra software development effort.

This guide will illustrate how to generate custom firmware for the STM32WLE5CCU6, which is inside the RAK3172 module. It supports two STM32WL SDK versions - v1.0.0 and v1.2.0.

- STM32CubeIDE guide with STM32WL SDK v1.0.0
- STM32CubeIDE guide with STM32WL SDK v1.2.0

Guide on Using STM32WL SDK Using STM32CubeIDE

Installation of STM32Cube IDE

1. Download the STM32Cube IDE from the STMicroelectronics website. Then select the latest version compatible with your computer.

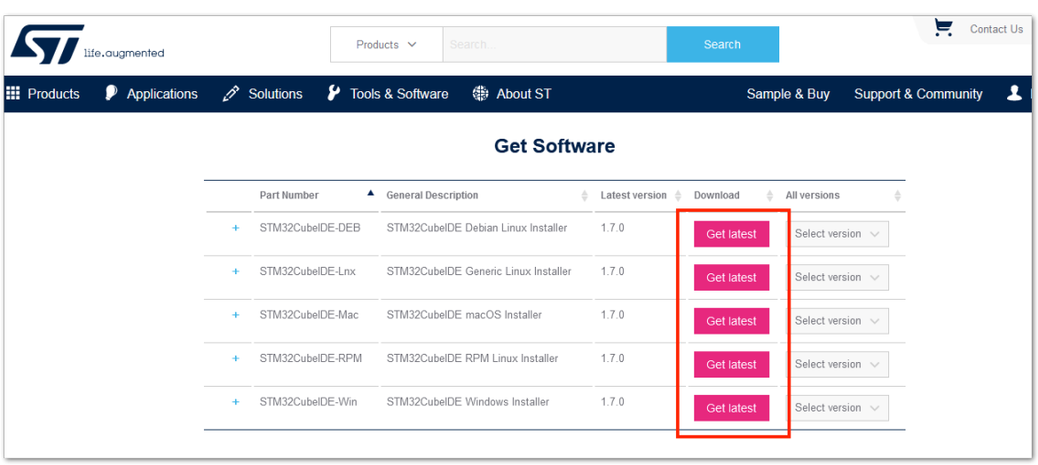

Figure 1: STM32CubeIDE Software Download

2. A license agreement from STMicroelectronics will show, and you must agree to log in using your STMicroelectronics account. Create an account if you do not have one.

3. After downloading, unzip the installer file and start the installation process. Just click next on the initial installation window and agree on the license agreement.

Figure 2: STM32CubeIDE Installation

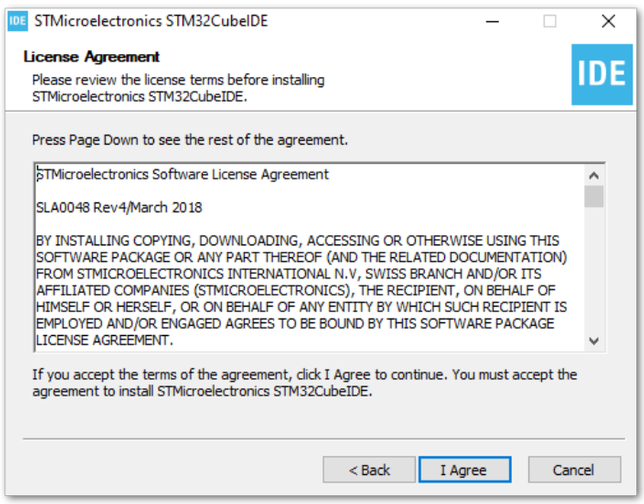

Figure 3: STM32CubeIDE License Agreement

3. Next step is to determine the directory where you want the STM32CubeIDE software to be placed. You can just click next to use the default folder or you can select a different location.

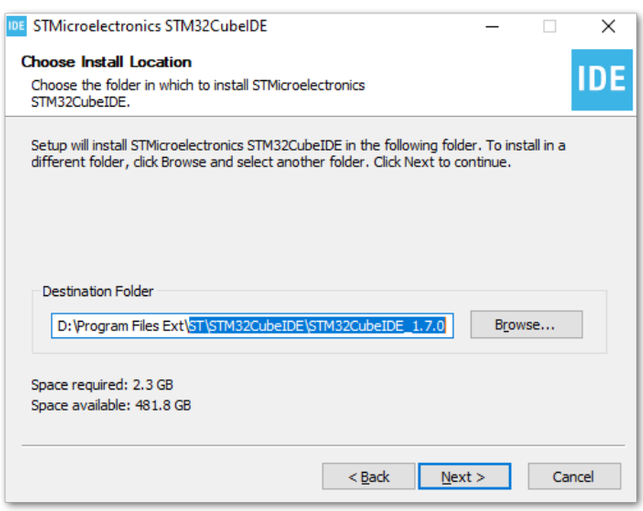

Figure 4: STM32CubeIDE Directory Location

4. Select optional components in the installation, like J-link and ST-Link drivers. It is highly recommended to include these drivers, which will be useful in debugging and uploading binary firmware files to the STM32WL chip.

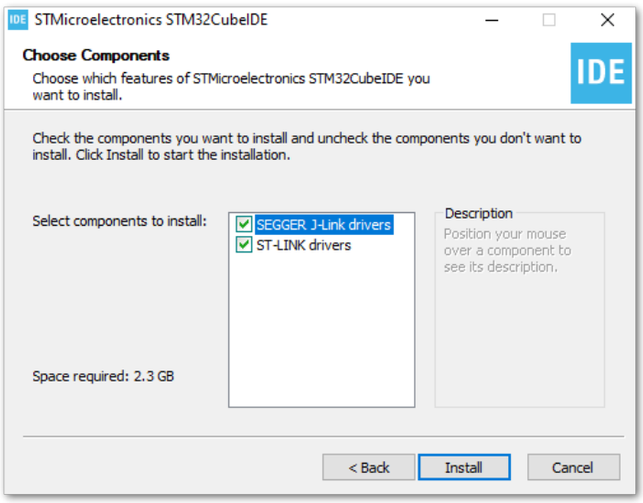

Figure 5: STM32CubeIDE Drivers

5. The progress bar will show as the installation begins.

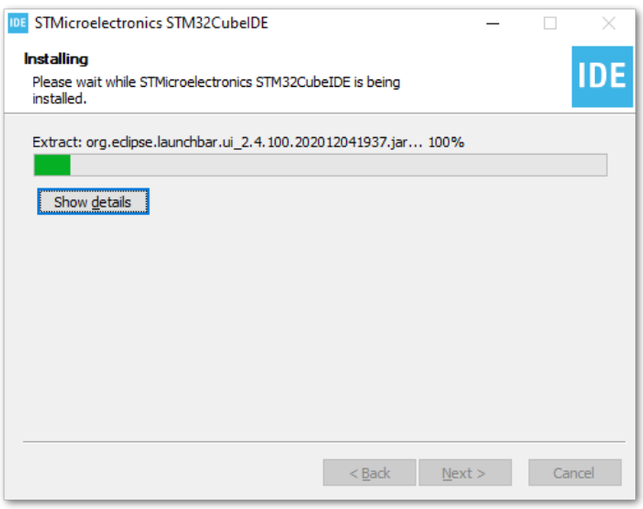

Figure 6: STM32CubeIDE On-going Installation

6. During the installation, there are STMicroelectronics device drivers that will pop up. These are not needed by STM32WL, so you can leave them uninstalled.

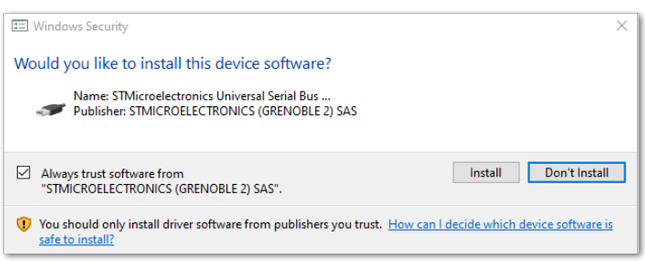

Figure 7: STM32CubeIDE Optional Device Drivers

7. If there are no problems in the installation process, you should be finished and can now create a desktop shortcut for the STM32CubeIDE application.

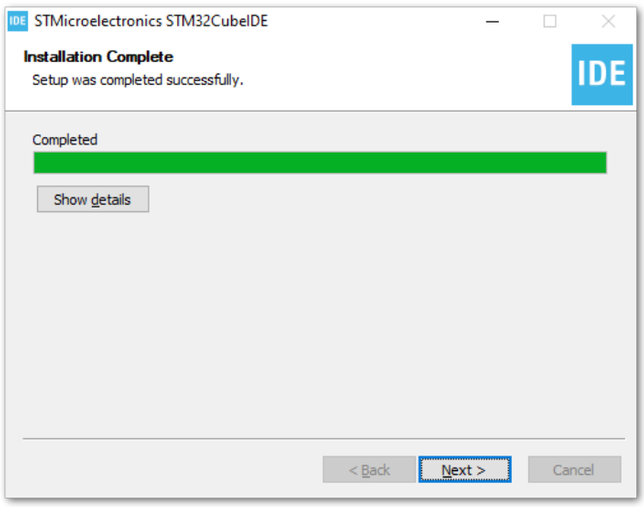

Figure 8: STM32CubeIDE Installation Successful

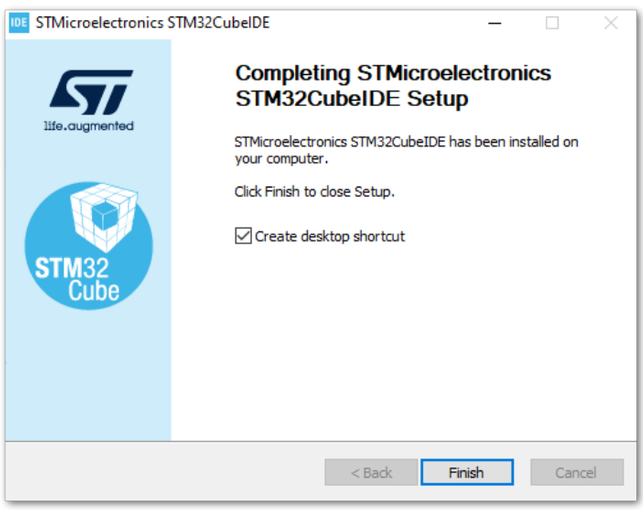

Figure 9: STM32CubeIDE Installation Finished

8. Check if the installation of the STM32CubeIDE is successful by trying to run the app. It should have no errors. It will ask you for the workspace and you can leave the default location if you don't want to put it in another location. You also have the option to create multiple workspaces but only one should be active.

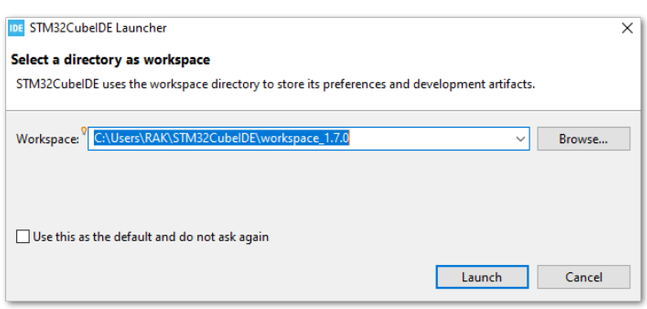

Figure 10: STM32CubeIDE Workspace Selection

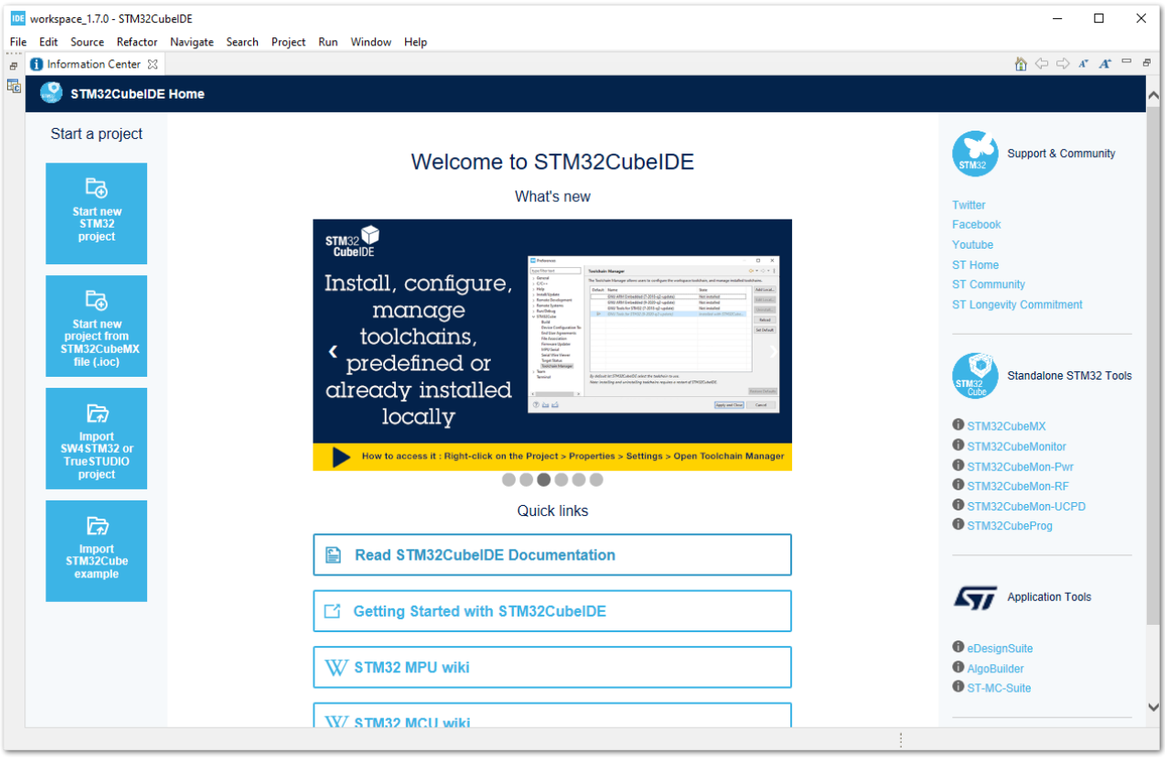

Figure 11: STM32CubeIDE Application Running

RAK3172 on STM32CubeIDE with STM32WL SDK v1.0.0

STM32CubeIDE is a free IDE from STMicroelectronics, which you can use to develop firmware for various STM32 microcontrollers including the STM32WL series. It is a complete software IDE based on Eclipse where you can debug easily your code with built-in integration to tools like ST-Link and other features like STM32CubeMX. It is multiplatform software that can run on Windows, Linux, and macOS.

You cannot select RAK3172 directly on the STM32CubeIDE, but you can use the STM32WLE5CCU6 inside it with STM32WL SDK from STMicroelectronics to start your own custom firmware. This guide is only applicable to STM32WL SDK v1.0.0.

Getting STM32WL SDK v1.0.0

This guide only works on v1.0.0 of the SDK.

1. If you already have the STM32CubeIDE running on your machine, the next step is to download the STM32WL SDK v1.0.0 from the STMicroelectronics website.

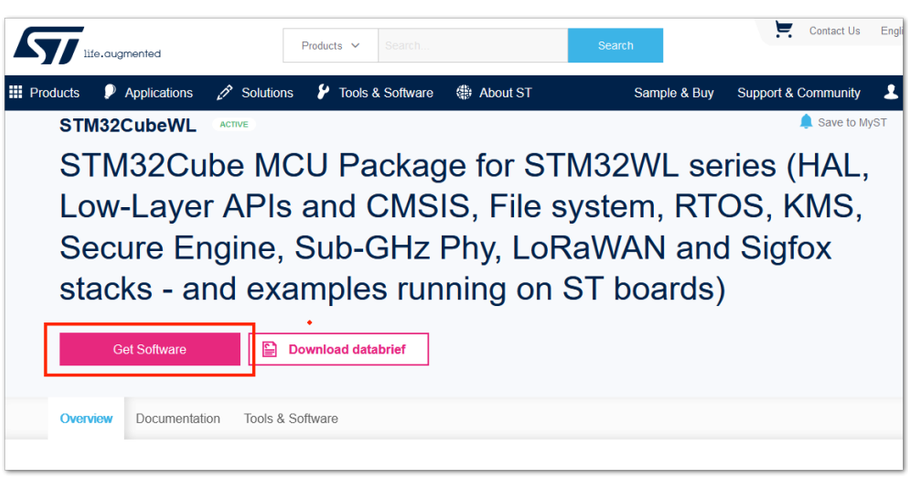

Figure 12: STM32WL SDK Download Page

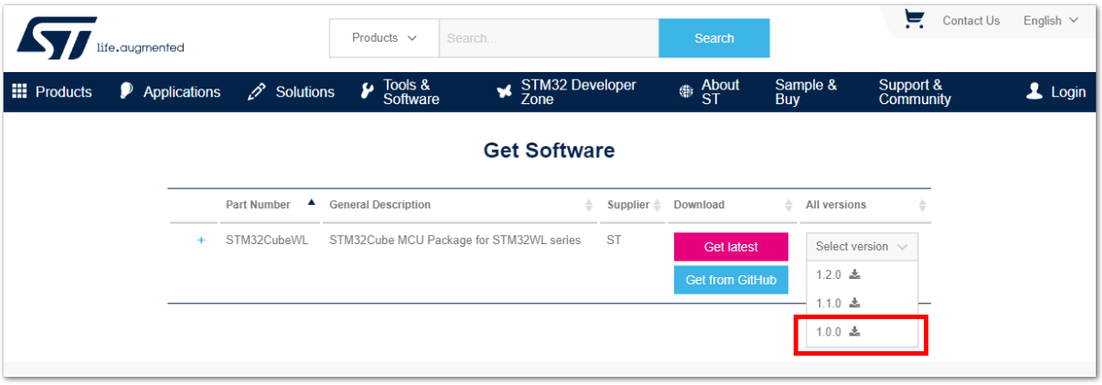

Figure 13: STM32WL V1.0.0 Download

2. The downloaded files usually go to the download folder. You need to extract it then you will see the STM32WL SDK firmware folder.

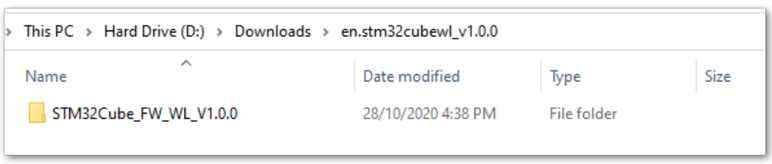

Figure 14: STM32WL V1.0.0

3. The structure of the extracted files should be the same, as shown in **Figure 15**. You cannot just change this folder structure. This contains many examples related to the STM32WL chip.

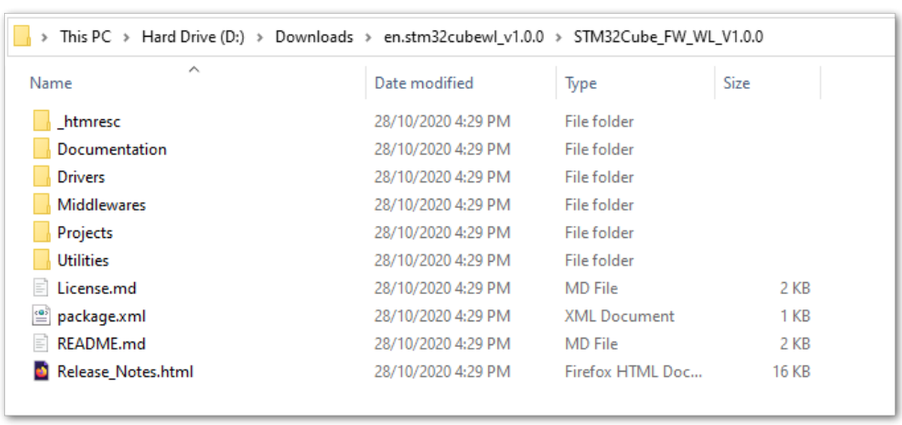

Figure 15: STM32WL V1.0.0 Folder Structure

Modifications for the RAK3172

Files Modification Needed to Run STM32WL SDK LoRaWAN Examples to RAK3172

If you already have the STM32WL V1.0.0 SDK folder, there are only a few files that you need to update to be able to create firmware that will run on RAK3172.

> 📝 **NOTE**
>
> STM32 microcontroller firmware created using STM32CubeIDE (with the help of STM32CubeMX) have **.ioc** file. This is a configuration file created by the STM32CubeMX tool. This is a helpful tool in setting up peripherals and drivers quickly in the STM32 development ecosystem. However, once you do the file modification mentioned in this guide, you cannot create a new **.ioc** file or modify it via STM32CubeMX. Else, those modified files needed by the RAK3172 will be overwritten and will go back to their original state or the **.ioc** file.
>
> In cases that you need to use STM32CubeMX to set up peripherals or drivers, you just need to do again the same modification as mentioned in this section.

1. Download the Low Level Development zip file from the RAK downloads site. Extract the zip file and inside the folder are four files that need to be copy-pasted on specific locations of the STM32WL V1.0.0 folder to make it compatible with RAK3172.

The majority of these files are for setting up the RF channel front end of the radio section on the STM32WL chip. Also, the startup file must be changed because the default startup on STM32WL SDK V1.0.0 is for the STM32WL55 series and not for STM32WLE5. The RAK3172 is based on STM32WLE5CCU6.

> 📝 **NOTE**
>
> This guide will demonstrate how to run the **LoRaWAN_End_Node** example of the STM32WL SDK to RAK3172. If you need to run other LoRaWAN-related examples like **LoRaWAN_AT_Slave**, you need to update the files on that folder.

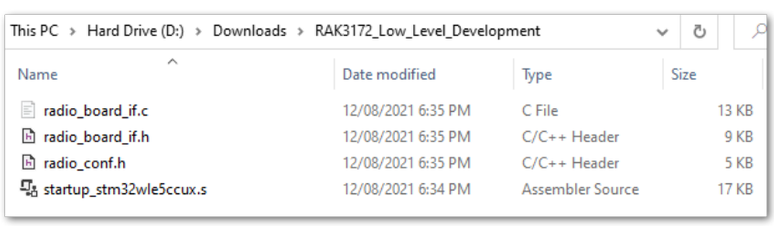

Figure 16: RAK3172 Low Level Development Files

2. The radio related files `radio_board_if.c`, `radio_board_if.h`, and `radio_conf` must be placed in this location of the STM32WL SDK folder `\STM32Cube_FW_WL_V1.0.0\Projects\NUCLEO-WL55JC\Applications\LoRaWAN\LoRaWAN_End_Node\LoRaWAN\Target`. You have to overwrite or replace the old files.

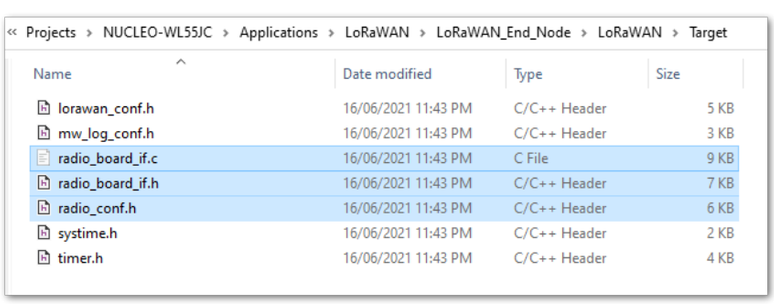

Figure 17: RAK3172 Radio Related Files for Modification

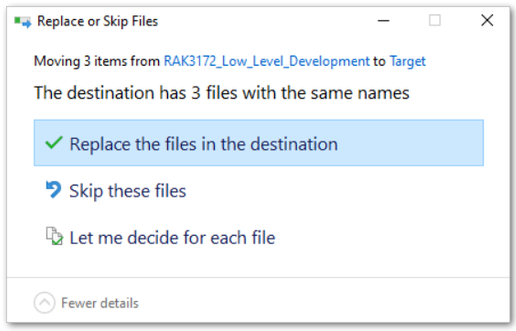

Figure 18: RAK3172 Radio Related Files Replaced

3. You also need to update the startup file. Place the `startup_stm32wle5ccux.s` file to this location `\STM32Cube_FW_WL_V1.0.0\Projects\NUCLEO-WL55JC\Applications\LoRaWAN\LoRaWAN_End_Node\STM32CubeIDE\Application\Startup`. There is a default startup file in that directory named `startup_stm32wl55jcix.s` and you need to delete that.

The updated startup folder should be the same, as shown in **Figure 19**.

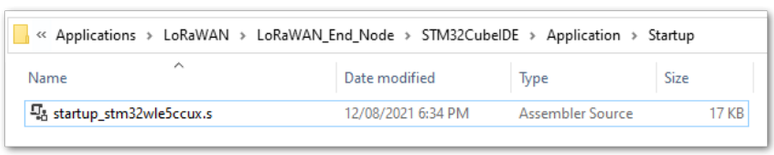

Figure 19: RAK3172 Radio Related Files Replaced

Initial Build Test for the RAK3172 Custom Firmware

1. After doing the file modifications, the next step is to test if the **LoRaWAN_End_Node** example can be built without errors.

> 📝 **NOTE**
>
> If this is your first time using STM32CubeIDE, it shows **Information Center** by default. Just close it and access the project on the left panel.

2. Open the STM32CubeIDE and click on `File` then `Open Projects from File System`.

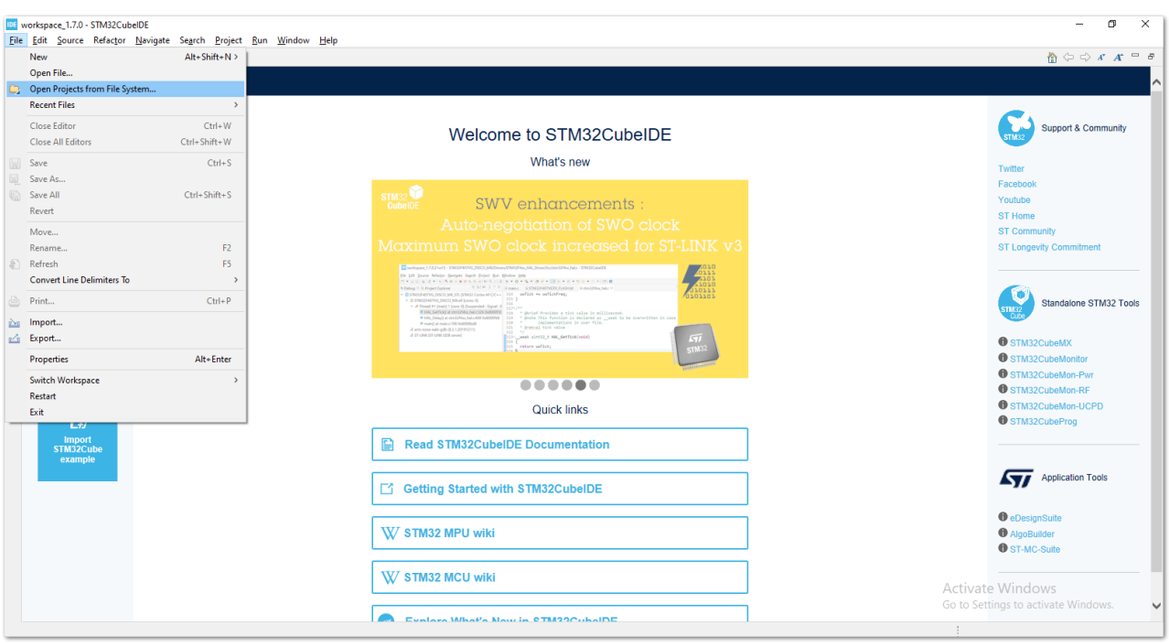

Figure 20: Open the Project in STM32CubeIDE

3. You then need to browse the project folder location by clicking the **Directory** button.

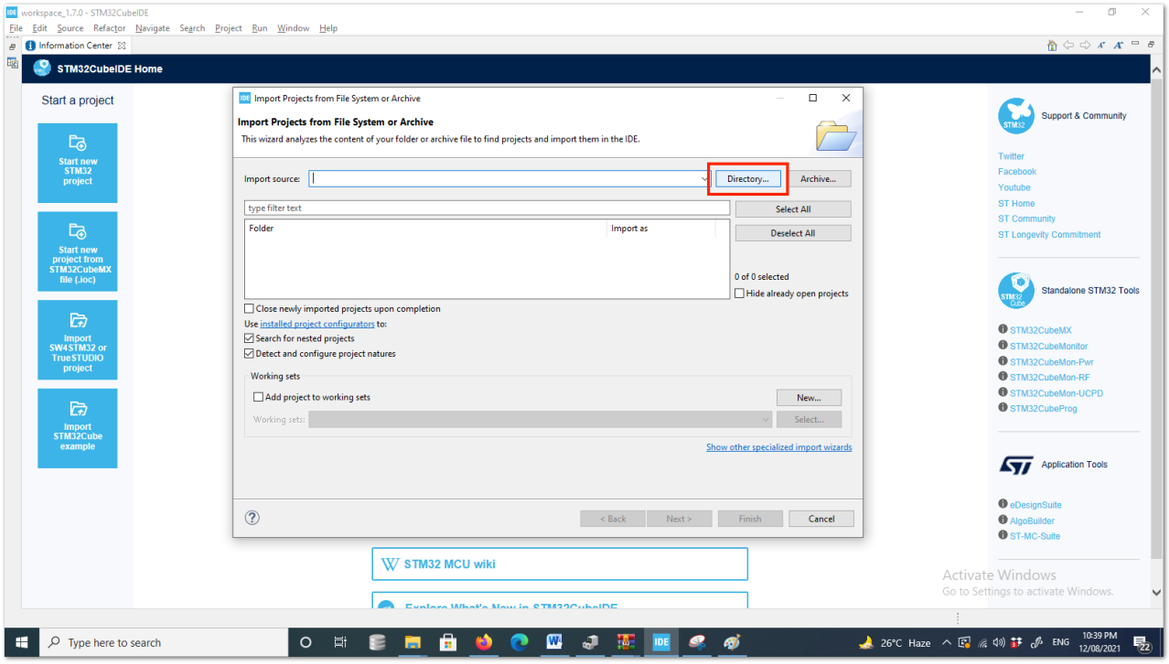

Figure 21: Locate the Project Directory in STM32CubeIDE

4. You should locate this directory `\STM32Cube_FW_WL_V1.0.0\Projects\NUCLEO-WL55JC\Applications\LoRaWAN\LoRaWAN_End_Node`. Click on **STM32CubeIDE** folder once, then click the **Select Folder**.

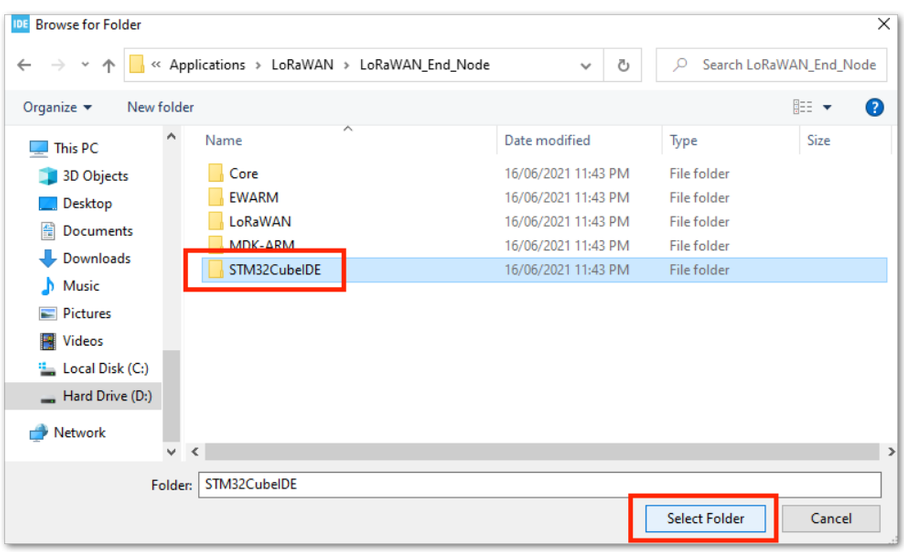

Figure 22: Select the STM32CubeIDE Project Folder

5. You should now see the **STM32CubeIDE** checked and ready to be imported as **Eclipse project**. If not checked, click the checkbox and then the **Finish** button. It will take some time to fully import the whole project.

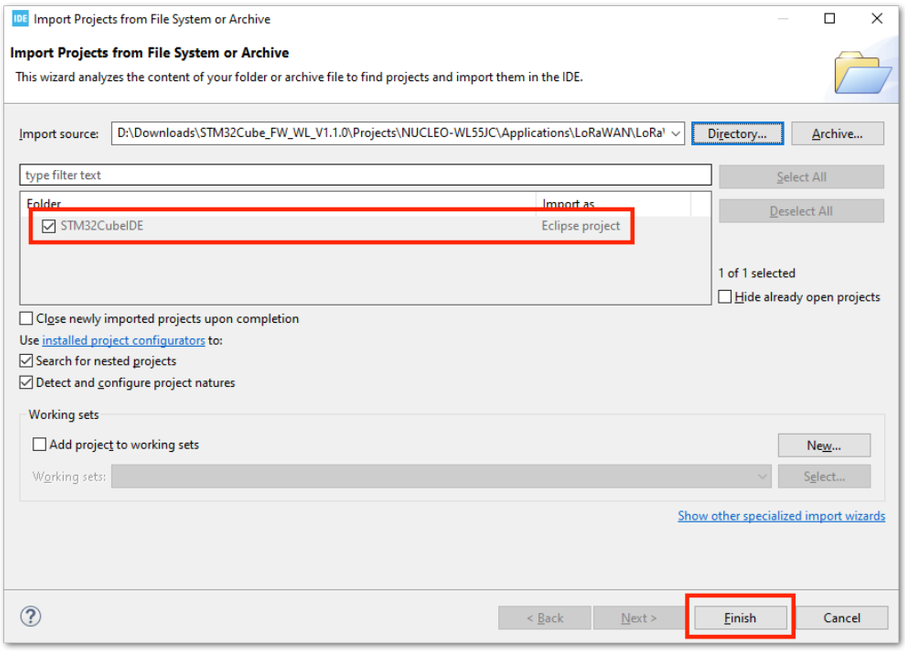

Figure 23: Open the LoRaWAN_End_Node Project

6. After the successful import, you should now see the **LoRaWAN_End_Node** project structure on the left side of the STM32CubeIDE.

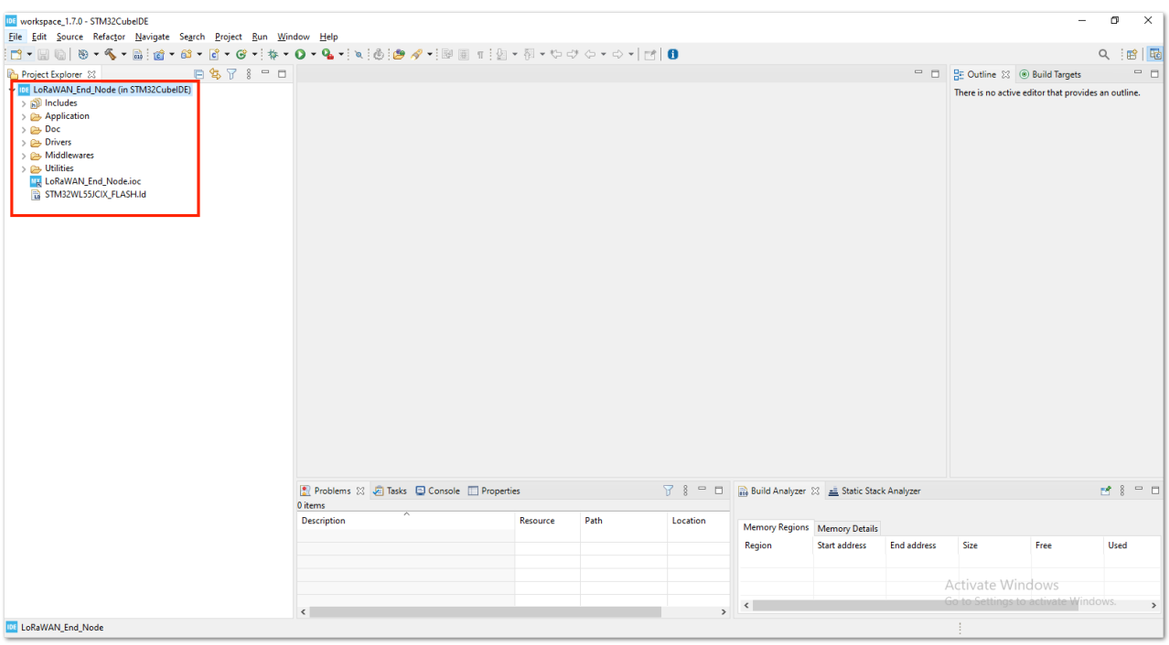

Figure 24: Open the STM32CubeIDE Project

7. With the modified files already implemented, you can check if the files are updated by checking the startup file `startup_stm32wle5ccux.s` and the `radio_board_if.c`. The startup file must be updated and show `startup_stm32wle5ccux.s`. You should see `#if defined(RAK3172_RF_CHANNEL_SWITCH)` in line 72 of `radio_board_if.c` file, as shown in **Figure 25**. If not, then you are not successful in changing these files with the Low Level Development required modification.

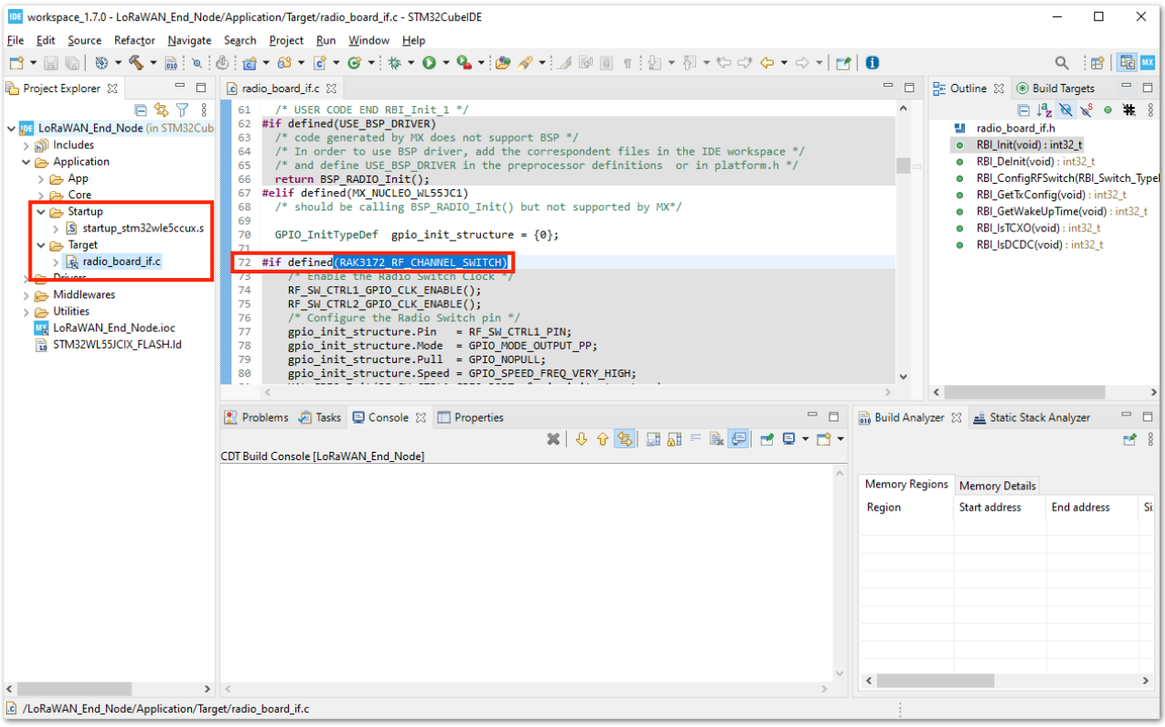

Figure 25: Open the STM32CubeIDE Project

8. You can now try to build the project by setting up the build configuration to release so that a `.bin` file will be generated.

> 📝 **NOTE**
>
> If you have an ST-LINK debugging tool, you can also choose **Debug** instead of **Release**.

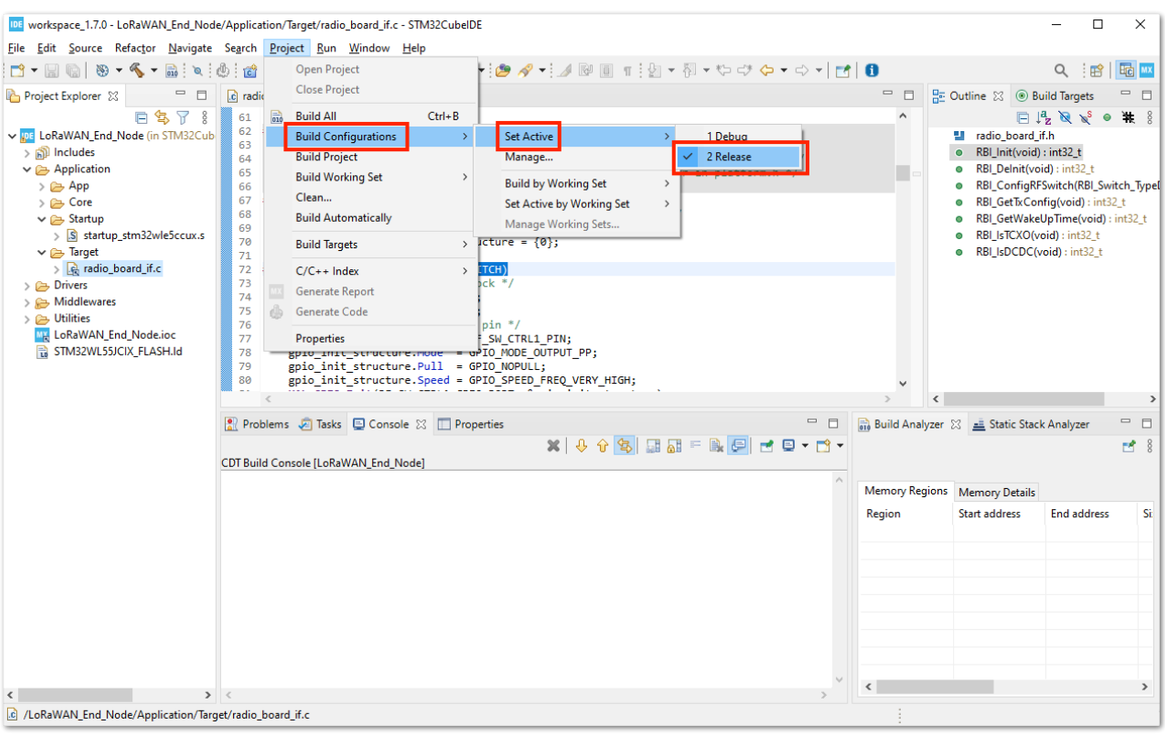

Figure 26: Configure Build to Release

9. After setting the build configuration, you are now ready to build the project. You should see a successful compilation and generation of a `.bin` file.

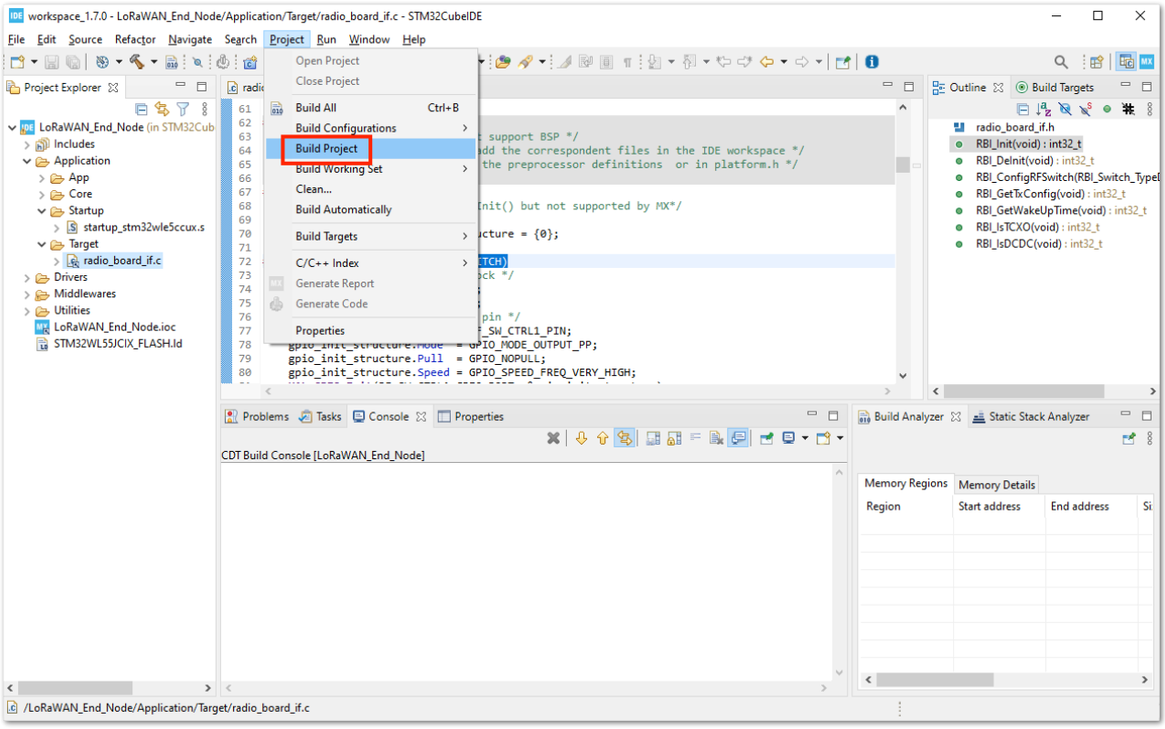

Figure 27: Build the Project

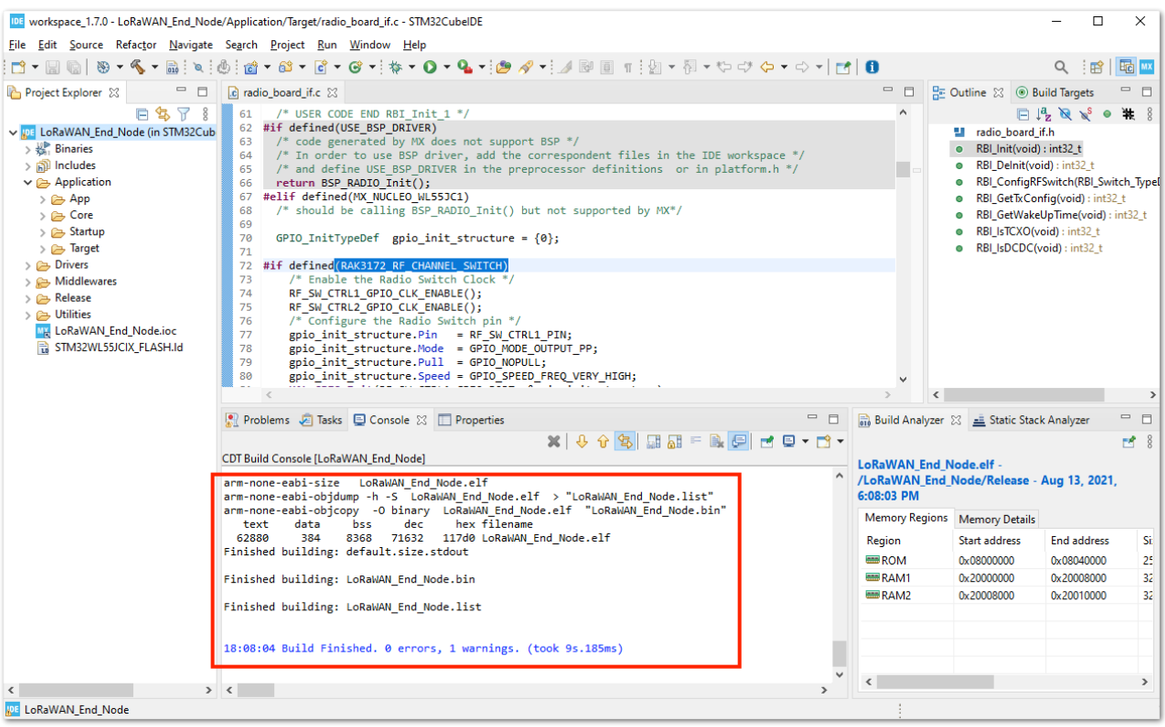

Figure 28: Successful Project Build

RAK3172 on STM32CubeIDE with STM32WL SDK v1.2.0

The previous guide is for STM32WL SDK version 1.0.0. This guide is compatible with STM32WL SDK v1.2.0.

Getting STM32WL SDK v1.2.0

1. If you already have the STM32CubeIDE running on your machine, the next step is to download the STM32WL SDK v1.2.0 from the STMicroelectronics website.

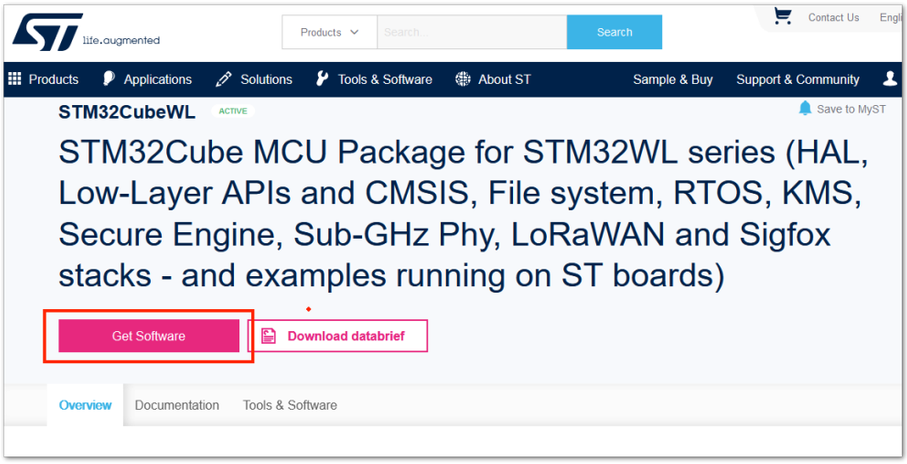

Figure 29: STM32WL SDK Download Page

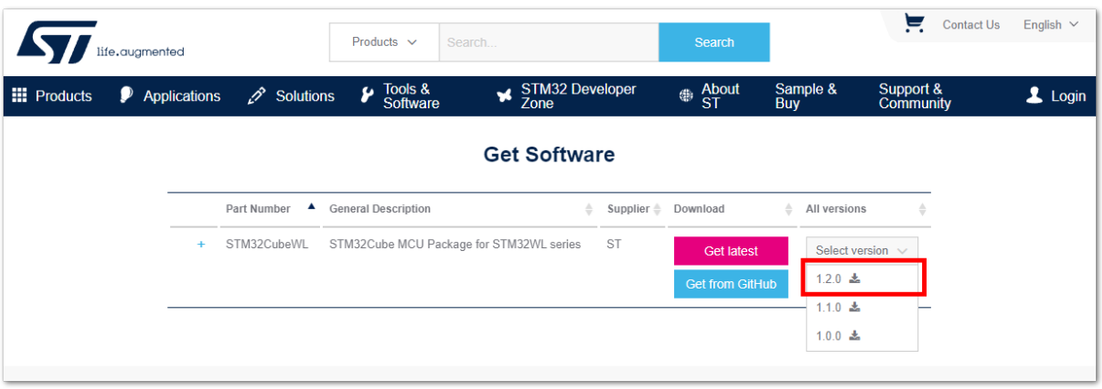

Figure 30: STM32WL V1.2.0 Download

2. The downloaded files usually go to the download folder. You need to extract it then you will see the STM32WL SDK firmware folder.

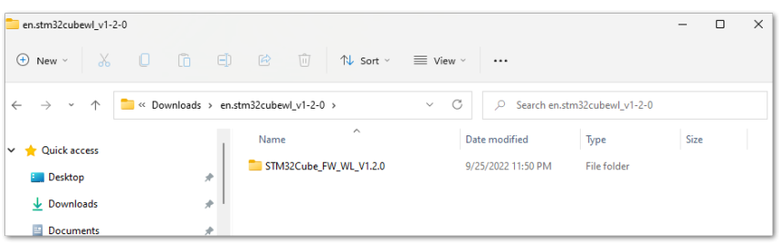

Figure 31: STM32WL V1.2.0

3. The structure of the extracted files should be the same, as shown in **Figure 32**. You cannot just change this folder structure. This contains many examples related to the STM32WL chip.

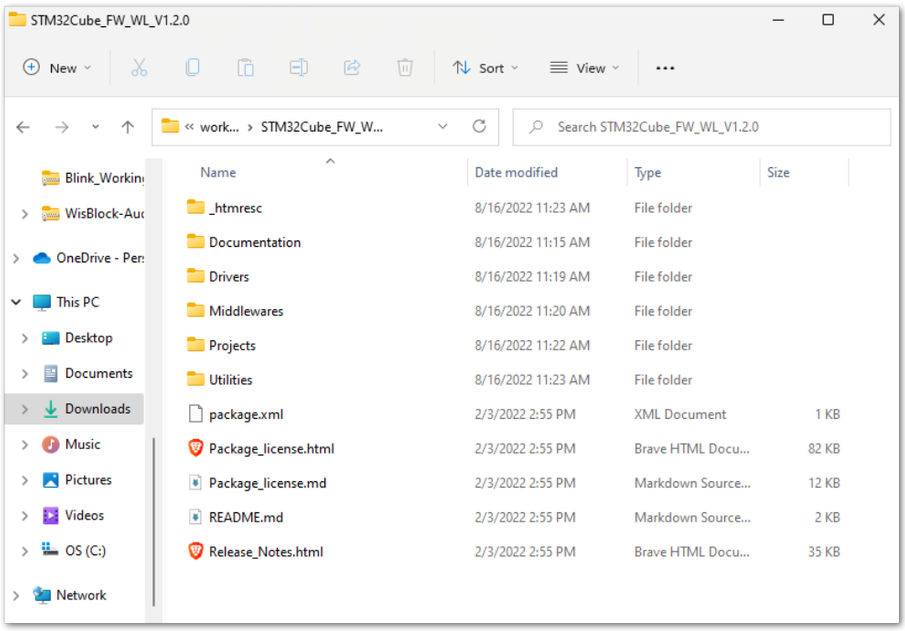

Figure 32: STM32WL V1.2.0 Folder Structure

Modifications for the RAK3172

Files Modification Needed to Run STM32WL SDK LoRaWAN Examples to RAK3172

If you already have the STM32WL V1.2.0 SDK folder, there are only a few files that you need to update to be able to create firmware that will run on RAK3172.

> 📝 **NOTE**
>
> STM32 microcontroller firmware created using STM32CubeIDE (with the help of STM32CubeMX) have **.ioc** file. This is a configuration file created by the STM32CubeMX tool. This is a helpful tool in setting up peripherals and drivers quickly in the STM32 development ecosystem. However, once you do the file modification mentioned in this guide, you cannot create a new **.ioc** file or modify it via STM32CubeMX. Else, those modified files needed by the RAK3172 will be overwritten and will go back to their original state or the **.ioc** file.
>
> In cases that you need to use STM32CubeMX to set up peripherals or drivers, you just need to do again the same modification as mentioned in this section.

1. Download the Low Level Development zip file for v1.2.0 from the RAK downloads site. Extract the zip file and inside the folder are four files that need to be copy-pasted on specific locations of the STM32WL V1.2.0 folder to make it compatible with RAK3172.

The majority of these files are for setting up the RF channel front end of the radio section on the STM32WL chip. Also, the startup file must be changed because the default startup on STM32WL SDK V1.2.0 is for the STM32WL55 series and not for STM32WLE5. The RAK3172 is based on STM32WLE5CCU6.

> 📝 **NOTE**
>
> This guide will demonstrate how to run the **LoRaWAN_End_Node** example of the STM32WL SDK to RAK3172. If you need to run other LoRaWAN-related examples like **LoRaWAN_AT_Slave**, you need to update the files on that folder.

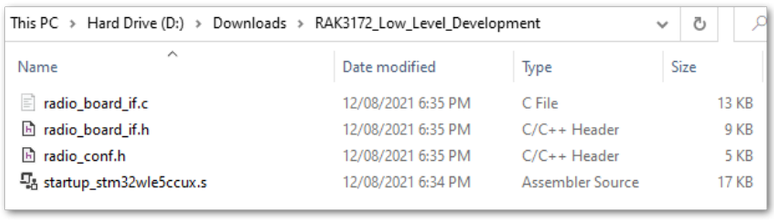

Figure 33: RAK3172 Low Level Development Files

2. The radio related files `radio_board_if.c`, `radio_board_if.h`, and `radio_conf` must be placed in this location of the STM32WL SDK folder `\STM32Cube_FW_WL_V1.2.0\Projects\NUCLEO-WL55JC\Applications\LoRaWAN\LoRaWAN_End_Node\LoRaWAN\Target`. You have to overwrite or replace the old files.

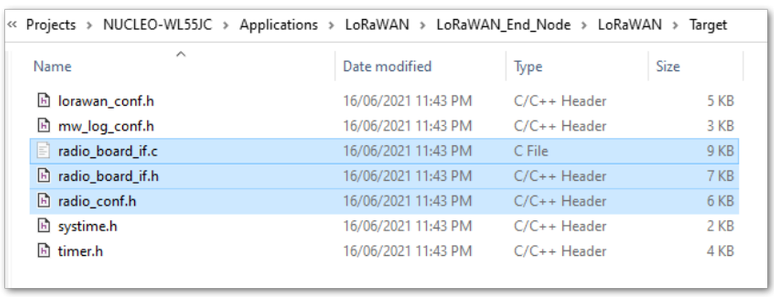

Figure 34: RAK3172 Radio Related Files for Modification

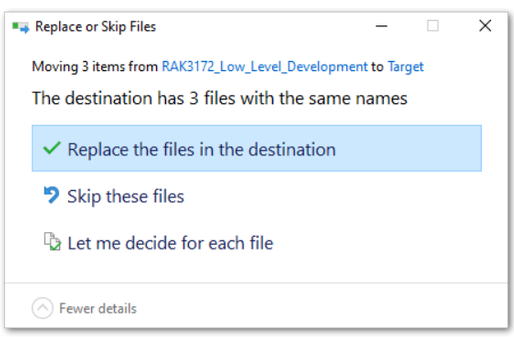

Figure 35: RAK3172 Radio Related Files Replaced

3. You also need to update the startup file. Place the `startup_stm32wle5ccux.s` file to this location `\STM32Cube_FW_WL_V1.2.0\Projects\NUCLEO-WL55JC\Applications\LoRaWAN\LoRaWAN_End_Node\STM32CubeIDE\Application\User\Startup`. There is a default startup file in that directory named `startup_stm32wl55jcix.s` and you need to delete that.

The updated startup folder should be the same, as shown in **Figure 36**.

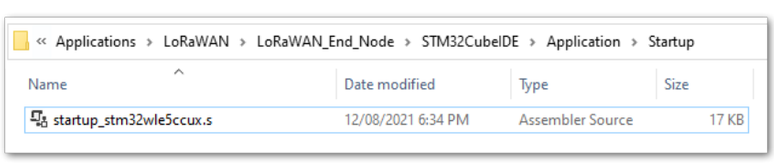

Figure 36: RAK3172 Radio Related Files Replaced

4. After the modifications above, there are minor changes needed to be adjusted on other source files.

- On *\STM32Cube_FW_WL_V1.2.0\Projects\NUCLEO-WL55JC\Applications\LoRaWAN\LoRaWAN_End_Node\Core\Inc\platform.h*, you need to comment out `#define USE_BSP_DRIVER`. It should be `//#define USE_BSP_DRIVER`.
- On *\STM32Cube_FW_WL_V1.2.0\Projects\NUCLEO-WL55JC\Applications\LoRaWAN\LoRaWAN_End_Node\LoRaWAN\Target\lorawan_conf.h*, you need to change the version of LoRaWAN to 1.0.3 which is the commonly used LNS version at the moment of this writing. To do this, you have to change LORAMAC_SPECIFICATION_VERSION to 0x01000300. It should look like this

`#define LORAMAC_SPECIFICATION_VERSION 0x01000300`. However, if you are using LoRaWAN version 1.0.4 on your LNS, you do not need to perform this step since the DevNonce will be handled correctly.

Initial Build Test for the RAK3172 Custom Firmware

1. After doing the file modifications, the next step is to test if the **LoRaWAN_End_Node** example can be built without errors.

> 📝 **NOTE**
>
> If this is your first time using STM32CubeIDE, it shows **Information Center** by default. Just close it and access the project on the left panel.

2. Open the STM32CubeIDE and click on `File` then `Open Projects from File System`.

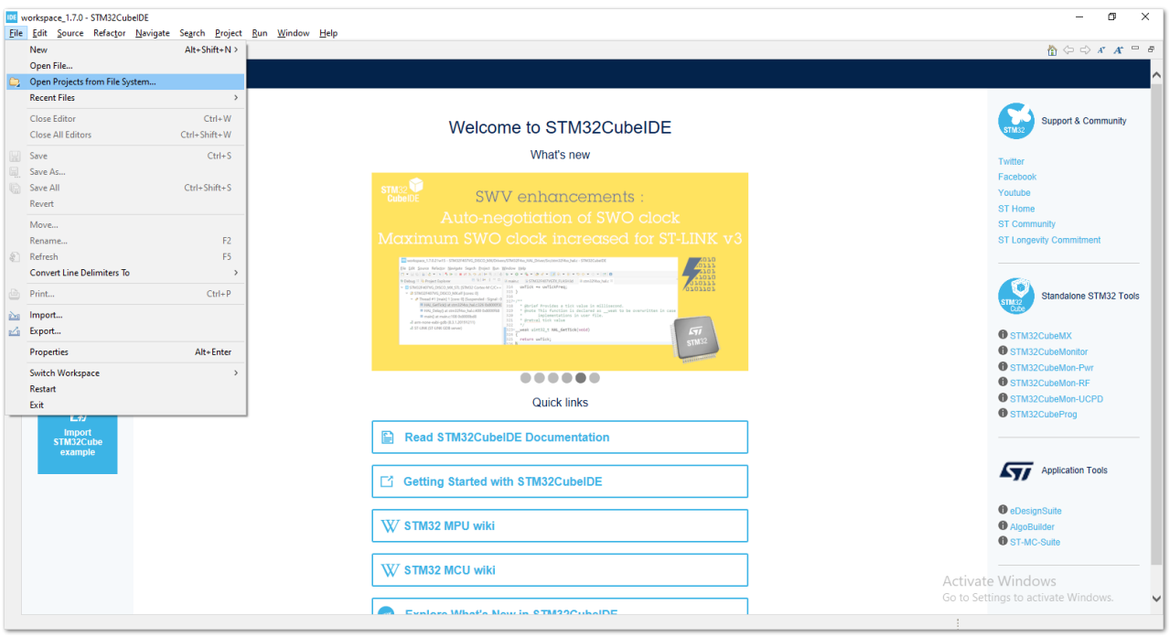

Figure 37: Open the Project in STM32CubeIDE

3. You then need to browse the project folder location by clicking the **Directory** button.

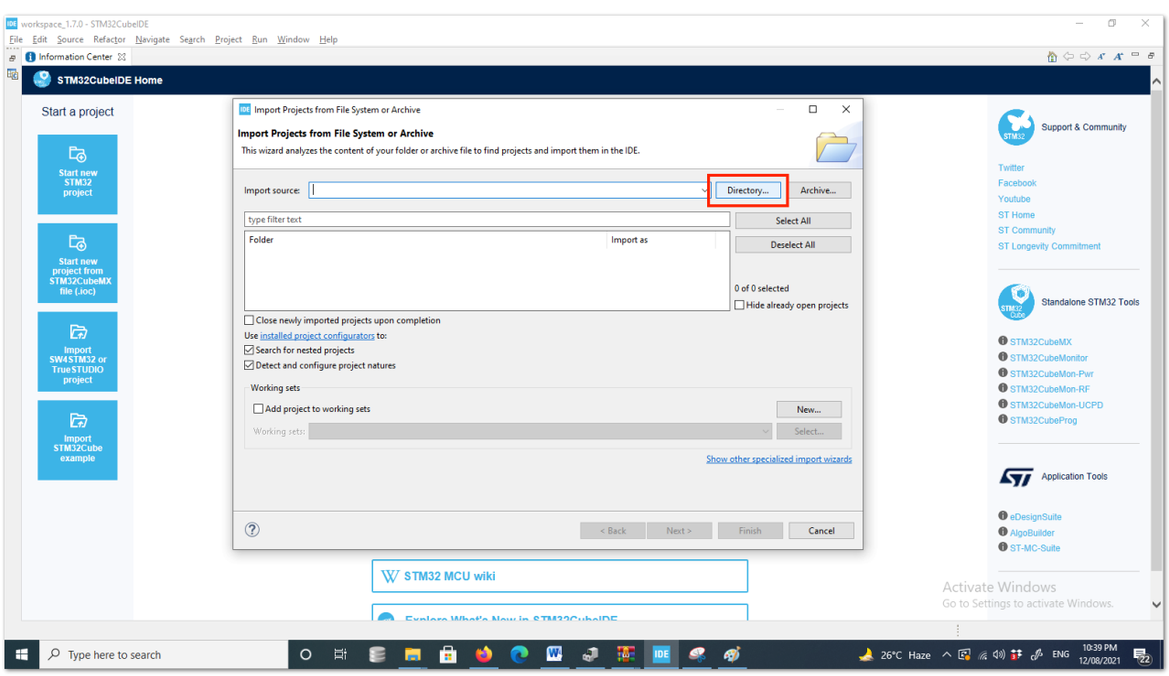

Figure 38: Locate the Project Directory in STM32CubeIDE

4. You should locate this directory `\STM32Cube_FW_WL_V1.2.0\Projects\NUCLEO-WL55JC\Applications\LoRaWAN\LoRaWAN_End_Node`. Click on **STM32CubeIDE** folder once, then click the **Select Folder**.

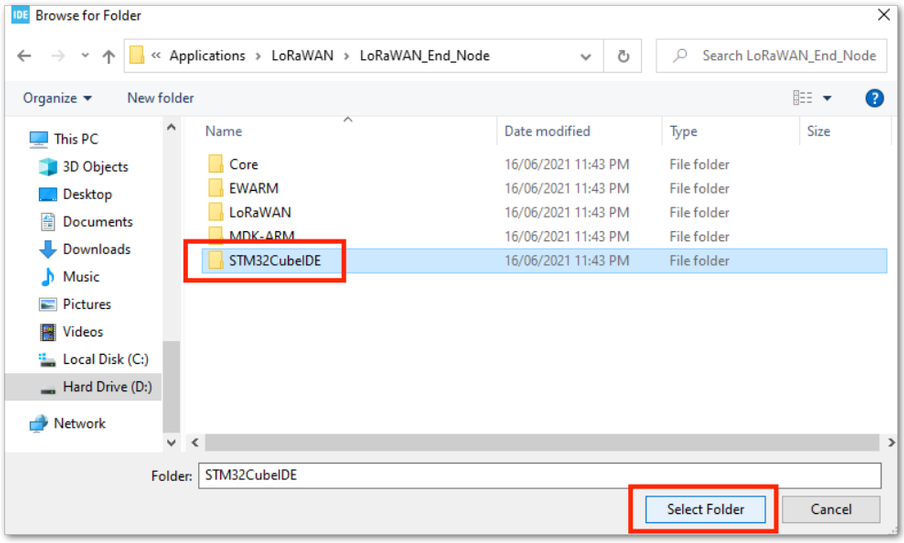

Figure 39: Select the STM32CubeIDE Project Folder

5. You should now see the **STM32CubeIDE** checked and ready to be imported as **Eclipse project**. If not checked, click the checkbox and then the **Finish** button. It will take some time to fully import the whole project.

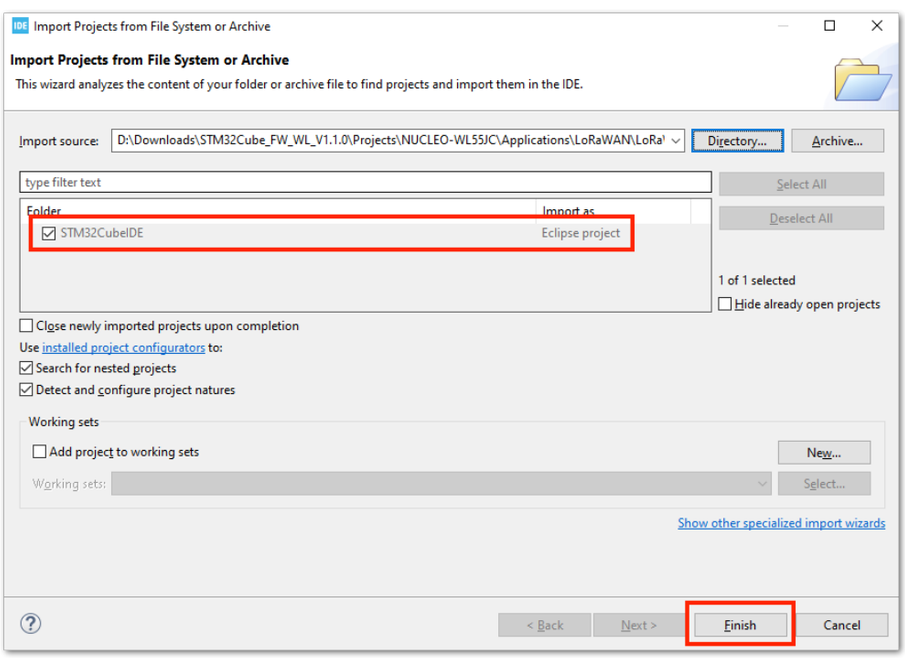

Figure 40: Open the LoRaWAN_End_Node Project

6. After the successful import, you should now see the **LoRaWAN_End_Node** project structure on the left side of the STM32CubeIDE.

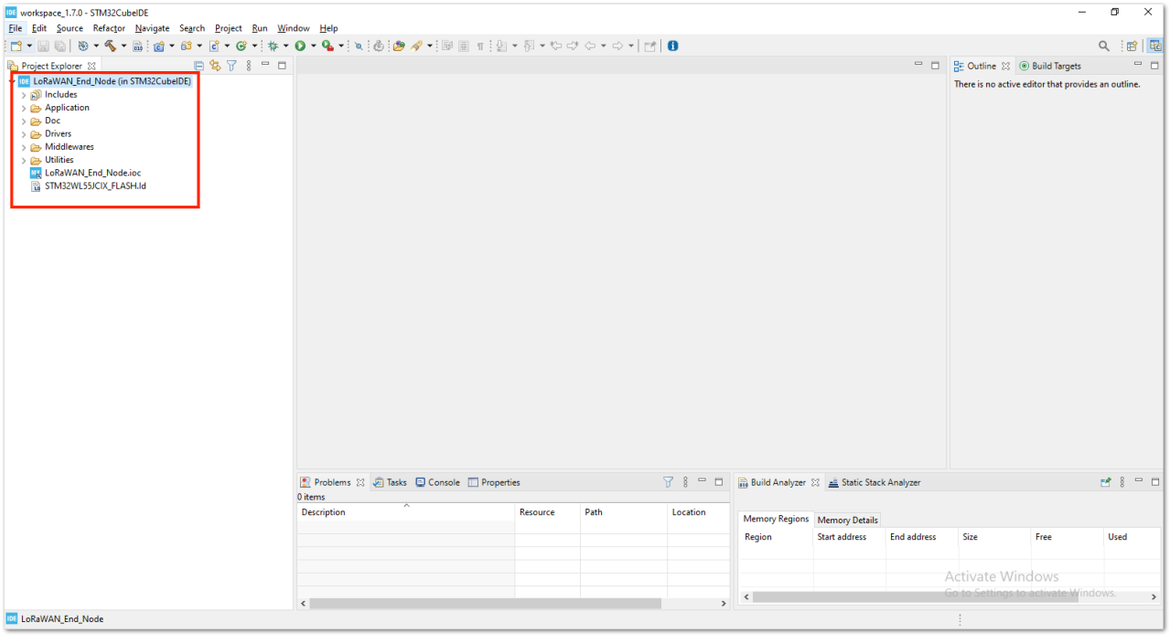

Figure 41: Open the STM32CubeIDE Project

7. With the modified files already implemented, you can check if the files are updated by checking the startup file `startup_stm32wle5ccux.s` and the `radio_board_if.c`. The startup file must be updated and show `startup_stm32wle5ccux.s`. You should see `#if defined(RAK3172_RF_CHANNEL_SWITCH)` in line 72 of `radio_board_if.c` file, as shown in **Figure 42**. If not, then you are not successful in changing these files with the Low Level Development required modification.

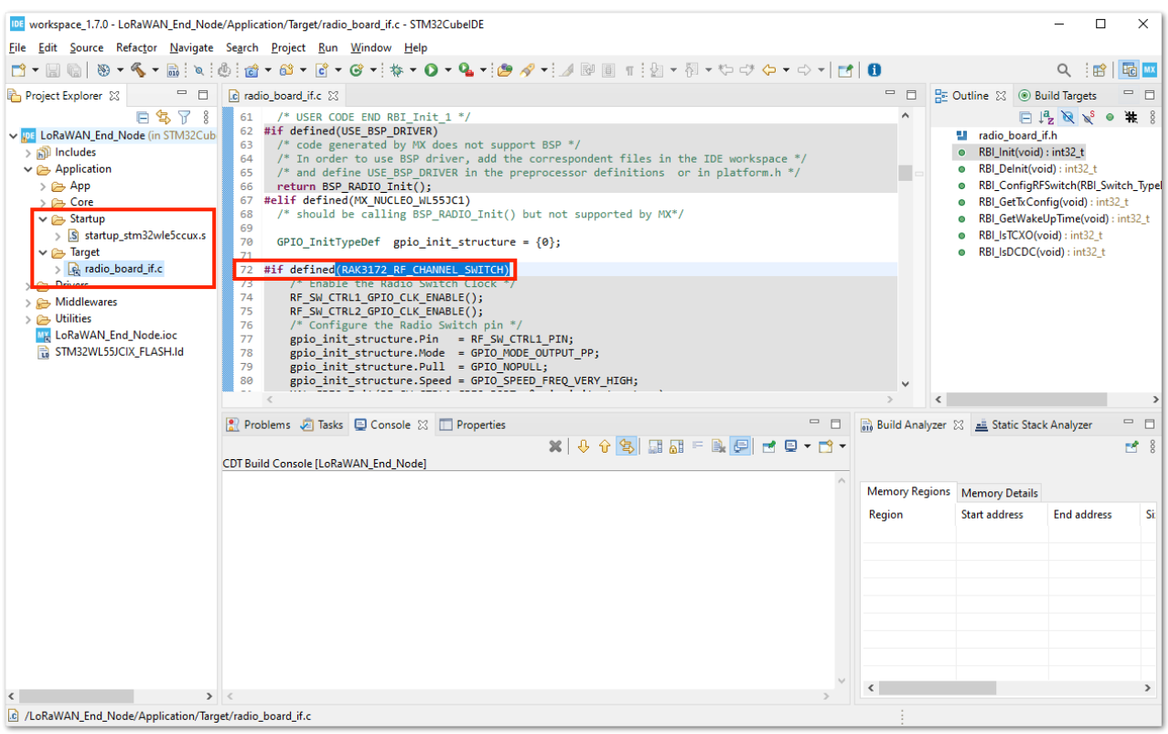

Figure 42: Open the STM32CubeIDE Project

8. The next step is to ensure that a bin file will be generated on your release build. Go to the properties and ensure that *Convert to binary file (-O binary)* is checked.

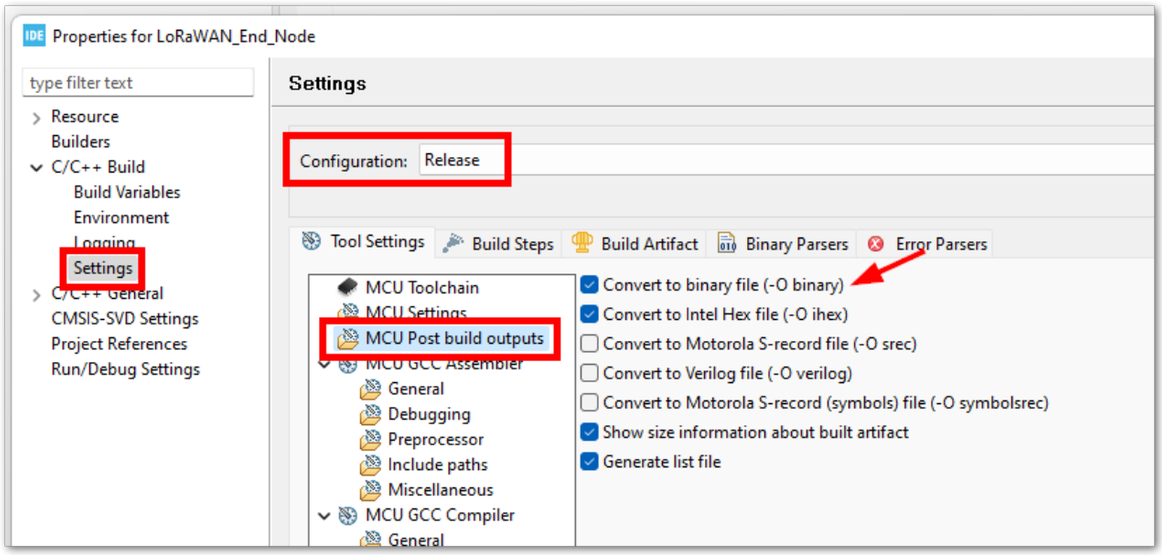

Figure 43: Bin generation settings

9. You can now try to build the project by setting up the build configuration to release so that a `.bin` file will be generated.

> 📝 **NOTE**
>
> If you have an ST-LINK debugging tool, you can also choose **Debug** instead of **Release**.

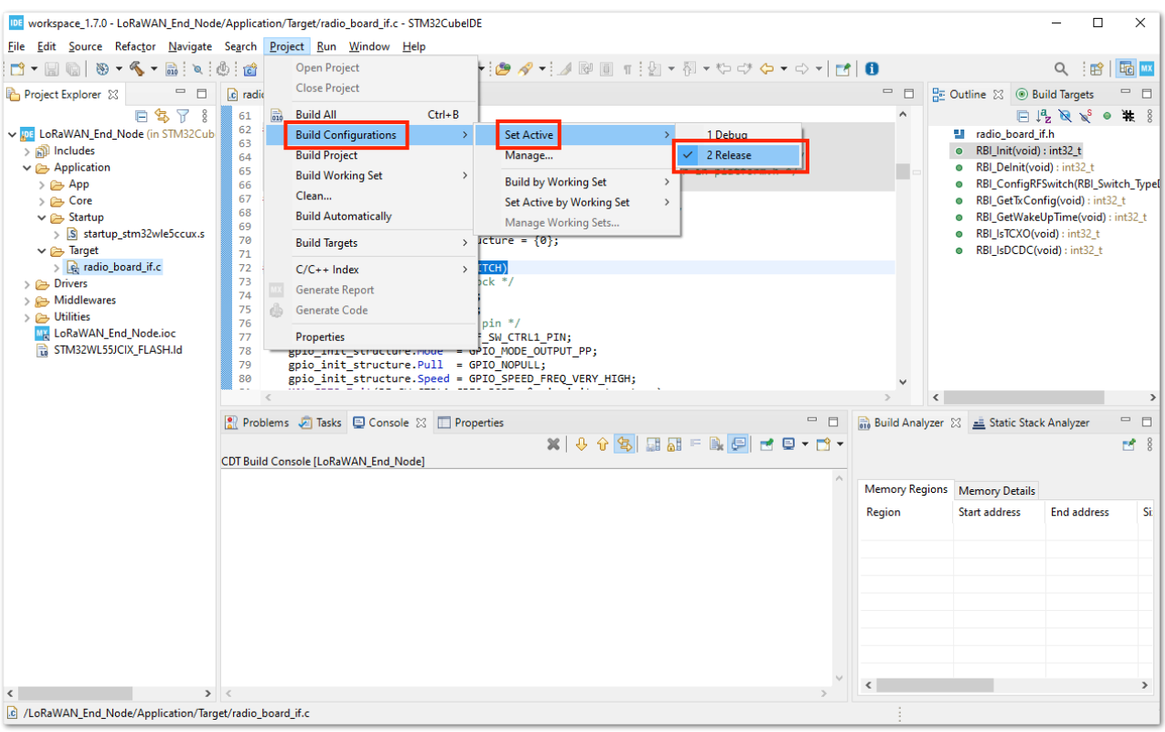

Figure 44: Configure Build to Release

10. After setting the build configuration, you are now ready to build the project. You should see a successful compilation and generation of a `.bin` file.

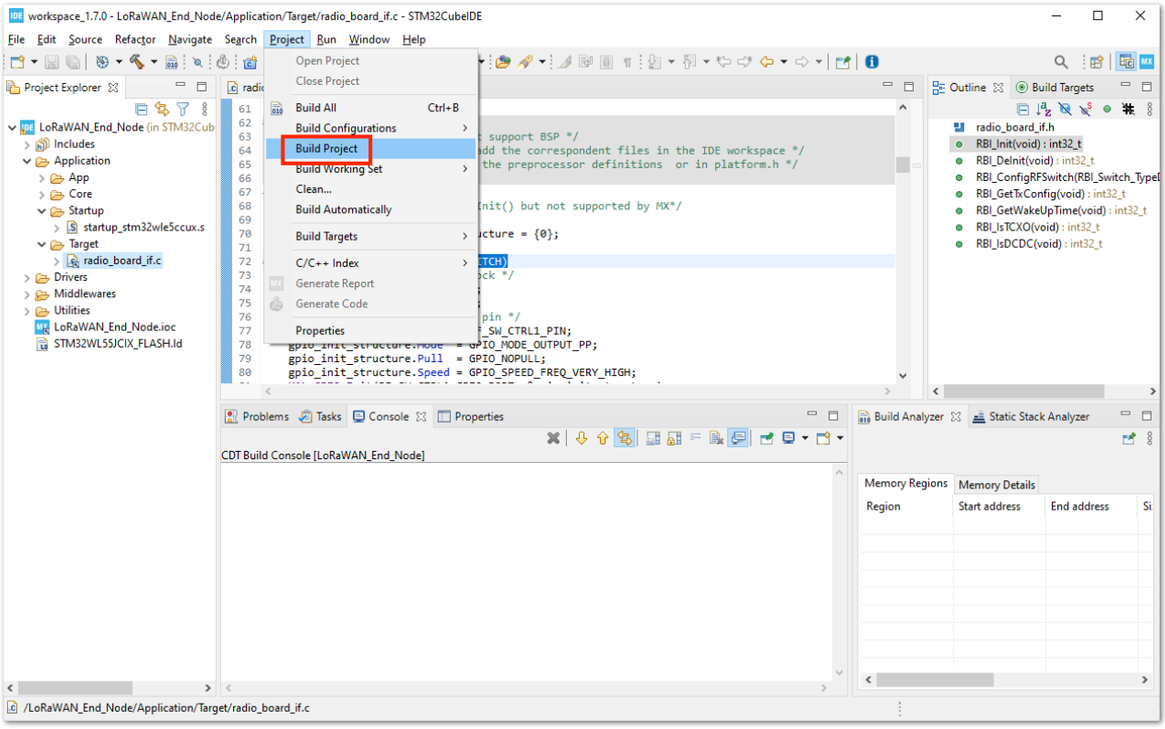

Figure 45: Build the Project

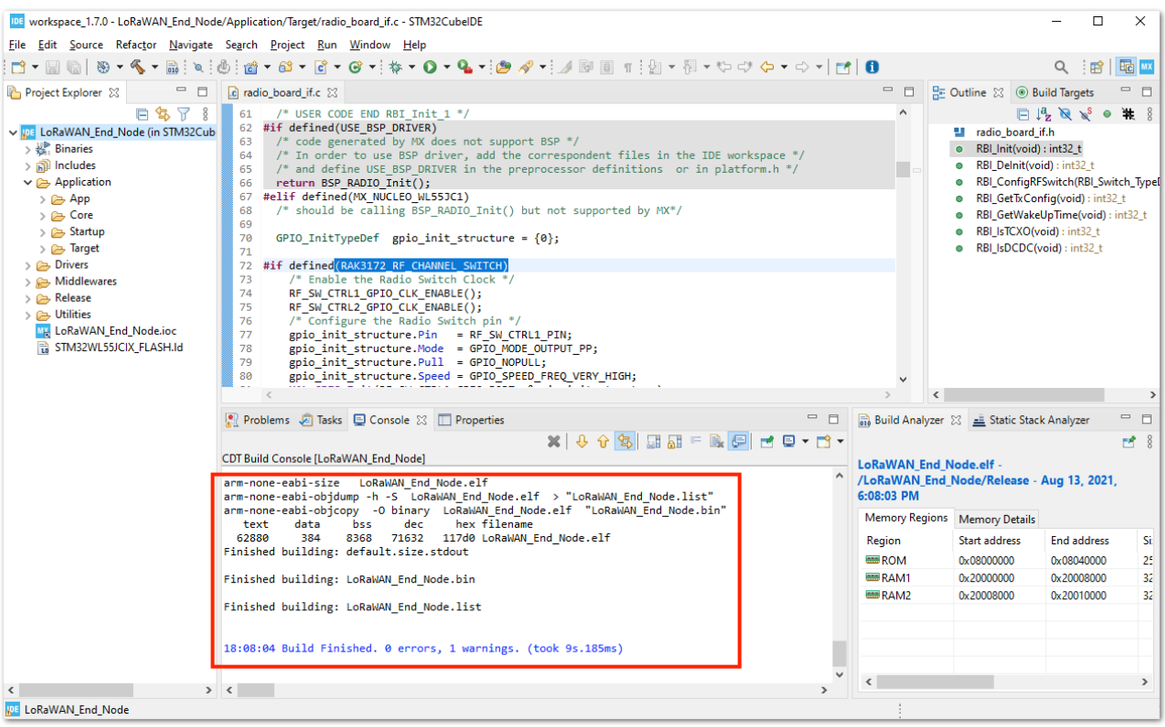

Figure 46: Successful Project Build

Running the LoRaWAN_End_Node Example of STM32WL SDK on RAK3172

Configuration to Connect the LoRaWAN Network Server

Once you have a working project and were able to build with no errors in the STM32CubeIDE, the next step is to configure the LoRaWAN parameters to be able to run **LoRaWAN_End_Node** example code with RAK3172.

1. First, you need to register the device to the network server. You can follow the guide on how to register a device in TTN V3 or Chirpstack using the procedure discussed in the RAK3172 TTN V3 OTAA Quick Start Guide or in the RAK3172 Chirpstack OTAA Quick Start Guide respectively.

> 📝 **NOTE**
>
> By default, the **LoRaWAN_End_Node** example will work on the EU868 region. This is set in the `lora_app.h` that can be found in this location `/STM32Cube_FW_WL_V1.0.0/Projects/NUCLEO-WL55JC/Applications/LoRaWAN/LoRaWAN_End_Node/LoRaWAN/App/` (for v1.0.0) and `/STM32Cube_FW_WL_V1.2.0/Projects/NUCLEO-WL55JC/Applications/LoRaWAN/LoRaWAN_End_Node/LoRaWAN/App/` (for v1.2.0).

2. To activate and connect your device via OTAA, you need to get the following parameters: **DEVEUI**, **APPEUI**, and **APPKEY**.

Once you successfully register your device to TTN V3, you should see those parameters, as shown in **Figure 47**.

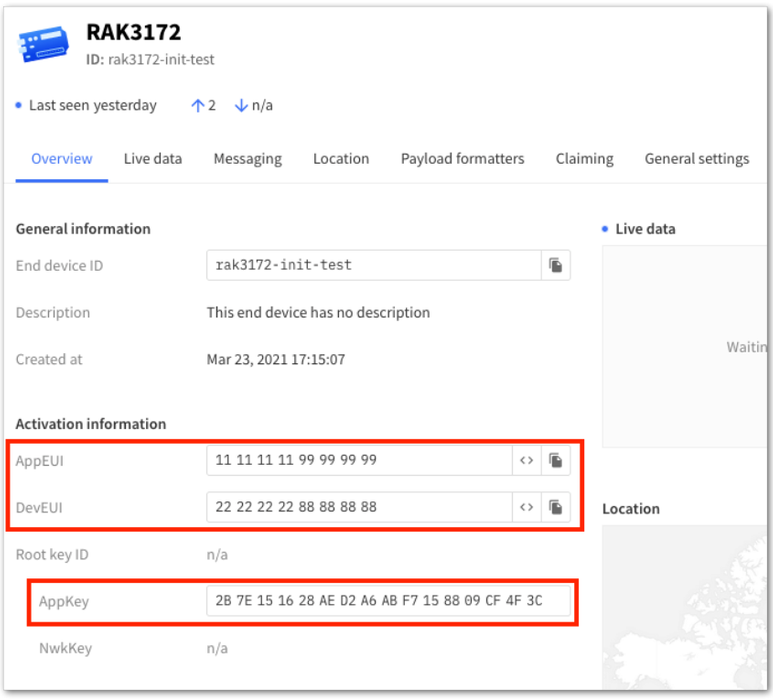

Figure 47: OTAA device registration in TTN V3

3. With the device registered to TTN V3, you should edit the `se-identity.h` file to update the needed OTAA parameters. On the STM32CubeIDE, click **File** and select **Open File..**. You should navigate in this directory `\STM32Cube_FW_WL_V1.0.0\Projects\NUCLEO-WL55JC\Applications\LoRaWAN\LoRaWAN_End_Node\LoRaWAN\App` (for v1.0.0) or `\STM32Cube_FW_WL_V1.2.0\Projects\NUCLEO-WL55JC\Applications\LoRaWAN\LoRaWAN_End_Node\LoRaWAN\App` (for v1.0.0) to find the `se-identity.h`, then click Open.

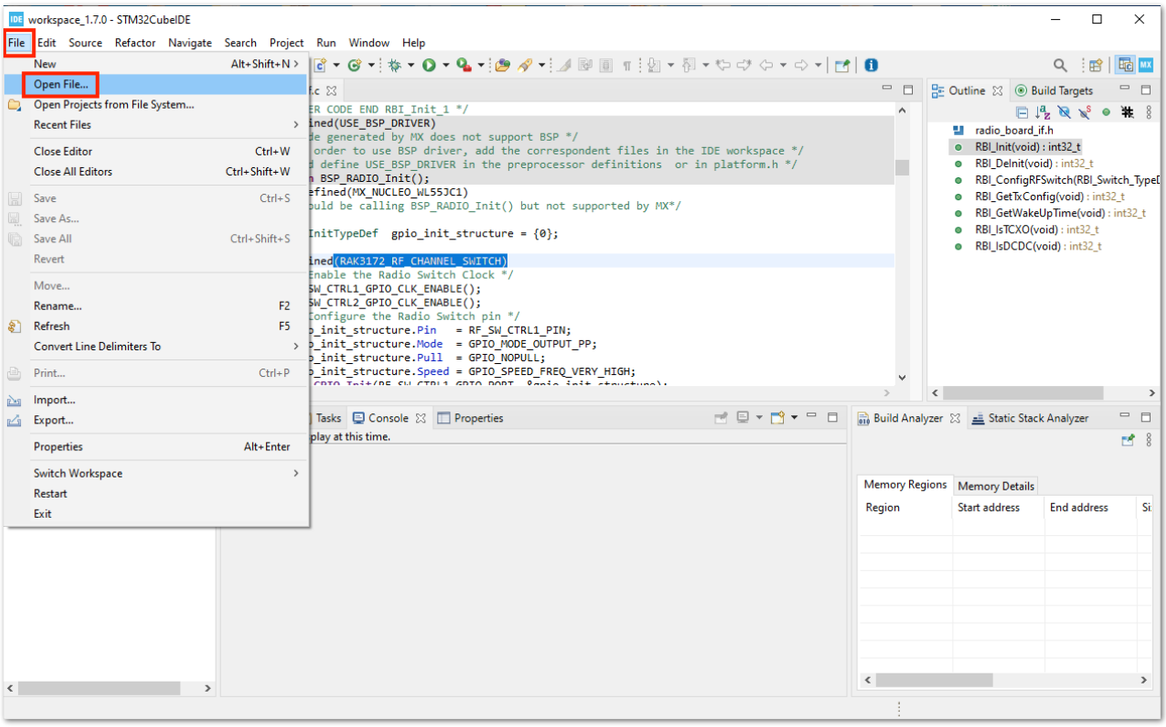

Figure 48: Open the File Needed to Modify with OTAA Parameters

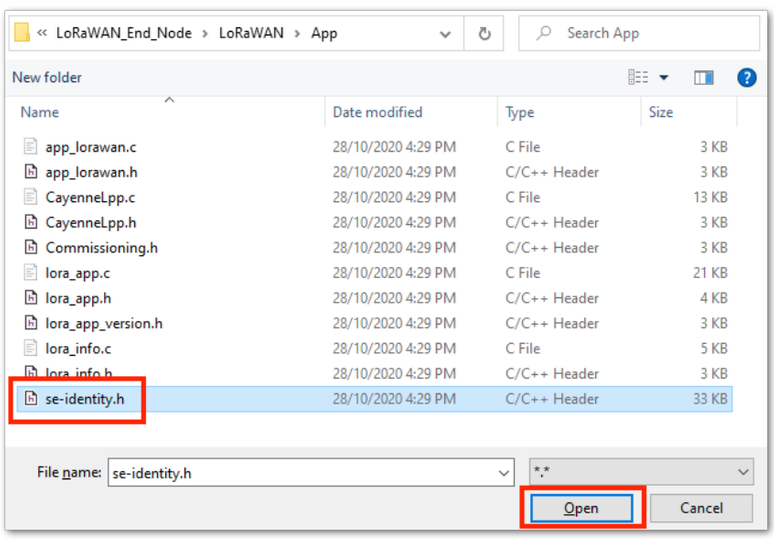

Figure 49: Open the se-identity.h File

4. The `se-identity.h` header file must be updated with the common DEVEUI, APPEUI, and APPKEY parameters in the device registration on the network server. In this example, you can see that the `LORAWAN_DEVEUI_EUI`, `LORAWAN_JOIN_EUI`, and `LORAWAN_APP_KEY` are updated, as shown in **Figure 50**. These values are based on the TTN V3 registration in **Figure 47**.

The `LORAWAN_JOIN_EUI` is the same as the `App_EUI` in this guide which is the term that adheres to the **LoRaWAN Specification V1.1**.

> 📝 **NOTE**
>
> To ensure that your device work on both LoRaWAN versions (**LoRaWAN Specifications V1.0.x and V1.1**), make sure that the application rook key `LORAWAN_APP_KEY` and the network root key `LORAWAN_NWK_KEY` of the `se-identity.h` file are exactly the same. Else, you might encounter MIC-related errors while joining the network.
>
> ```c
> /*!
> * Application root key
> */
> #define LORAWAN_APP_KEY                                2B,7E,15,16,28,AE,D2,A6,AB,F7,1
>
> /*!
> * Network root key
> */
> #define LORAWAN_NWK_KEY                                2B,7E,15,16,28,AE,D2,A6,AB,F7,1
> ```

The macro `STATIC_DEVICE_EUI` is also updated to `1` instead of `0` since a generated DEVEUI in TTN V3 is used in this guide instead of the embedded DEVEUI of the device.

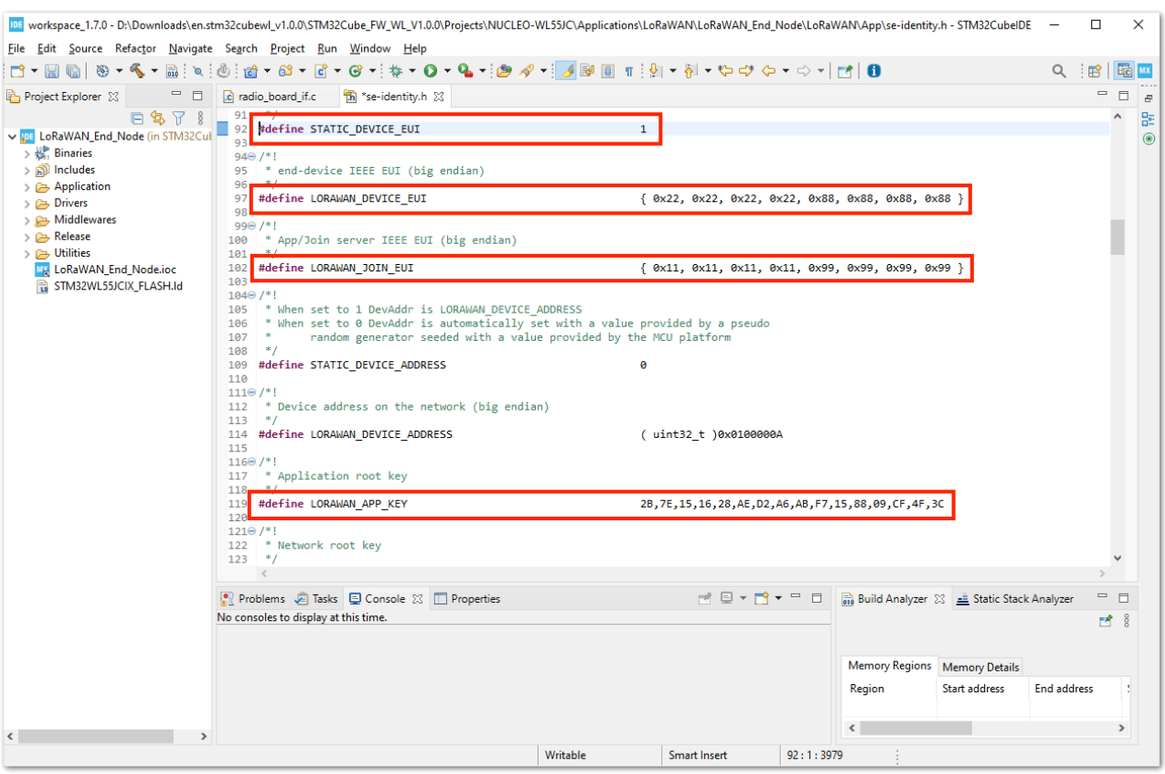

Figure 50: Modified se-identity.h File

> 📝 **NOTE**
>
> These parameters are usually set and code generated in the **.ioc** file via STM32CubeMX. But that method is not possible since a direct modification of the radio-related source files is done in this guide. Any further code generation via STM32CubeMX after the modifications in the previous steps in this guide will override all the changes that are required to run the LoRaWAN_End_Node project example RAK3172 module.

Generation of BIN file

With all the needed files modified and edited, you can now generate your `.bin` FW file and upload it to your RAK3172 module.

1. The first step is to clean first the project to remove any outdated binary files in the project folder then followed by building it. Sometimes **Build Project** is not clickable so you can use **Build All** as an alternative. You only have one project as of now so that should work fine as well.

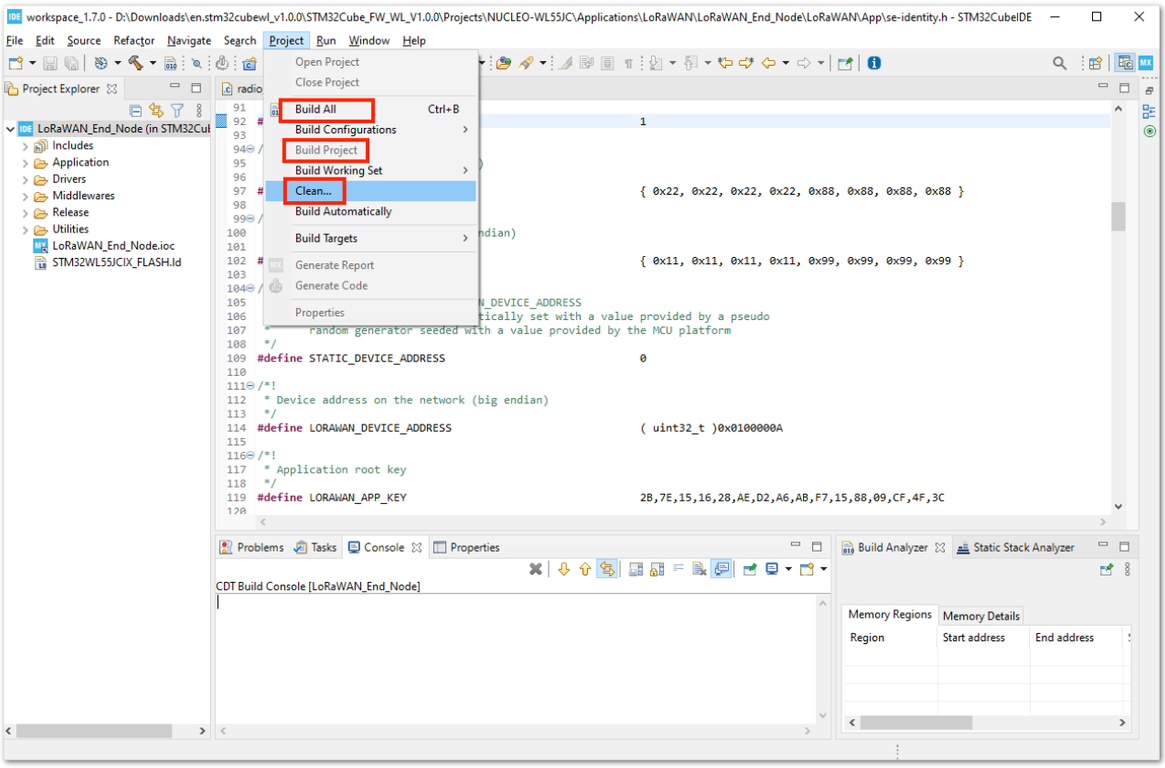

Figure 51: Clean and Build the STM32CubeIDE Project

2. After a successful build, you should see in console **Finished building: LoRaWAN_End_Node.bin**. You should be able to see the generated `LoRaWAN_End_Node.bin` firmware file in this folder location `\STM32Cube_FW_WL_V1.0.0\Projects\NUCLEO-WL55JC\Applications\LoRaWAN\LoRaWAN_End_Node\STM32CubeIDE\Release` (for v1.0.0) or `\STM32Cube_FW_WL_V1.2.0\Projects\NUCLEO-WL55JC\Applications\LoRaWAN\LoRaWAN_End_Node\STM32CubeIDE\Release` (for v1.2.0). That bin file is the firmware binary that you need to upload to your RAK3172 module.

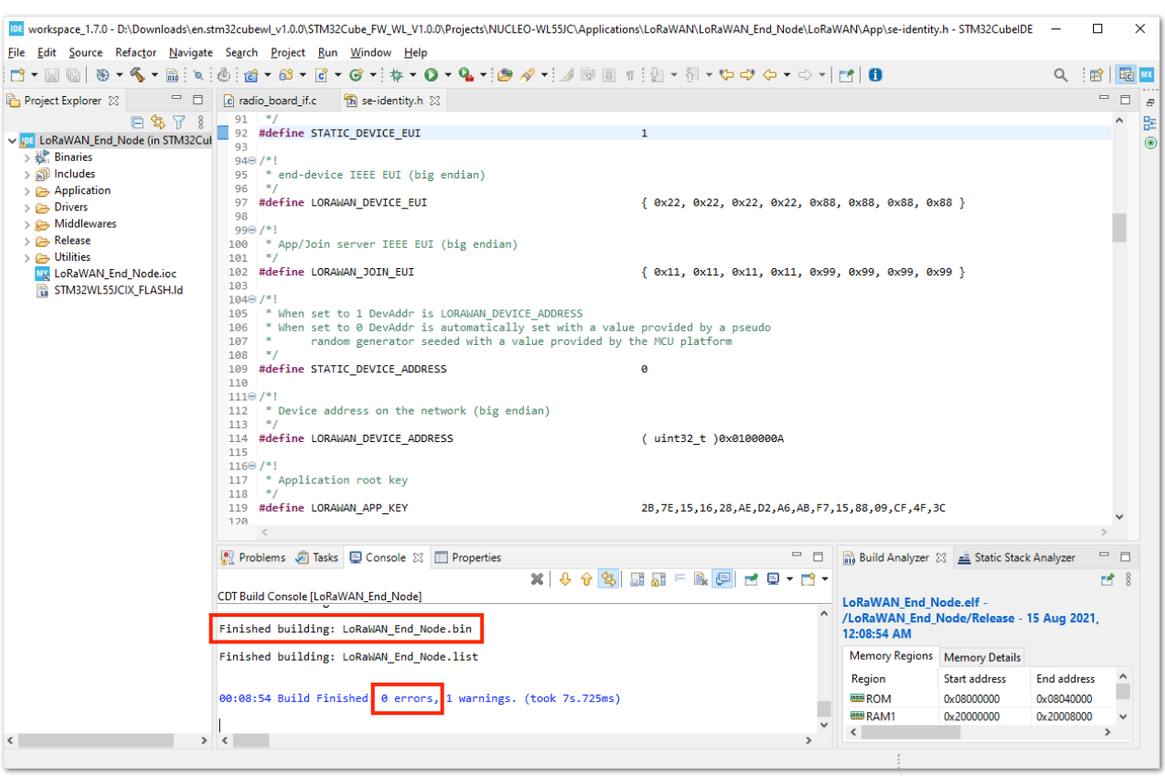

Figure 52: Successful Build with BIN File for RAK3172

Uploading the FW Generated Using STM32CubeProgrammer

The generated .bin FW file is ready to be uploaded to RAK3172.

To upload this binary file, you need to use STM32CubeProgrammer created by STMicroelectronics.

Download the latest version STM32CubeProgrammer and the one compatible with your computer.

In this guide, you will use the internal UART bootloader of the STM32WL and connect the RAK3172 to a USB to Serial converter tool like the RAKDAP1. You need to connect five pins: power supply pins (3.3 V and GND), UART2 pins (TX and RX), and the Boot0 pin (connected to 3.3 V), as shown in **Figure 53**.

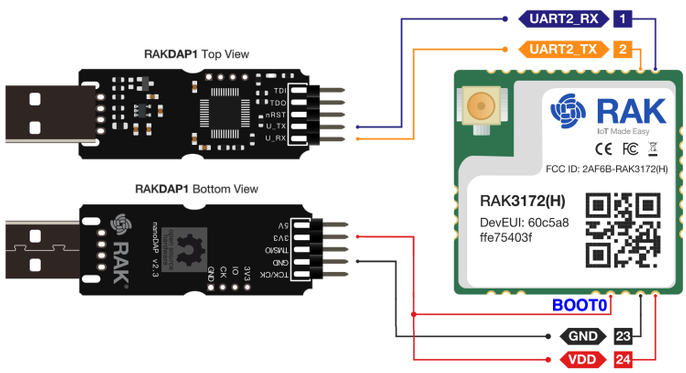

Figure 53: RAK3172 Connection to USB-UART Converter with Boot0 Pin at 3.3V Level

> 📝 **NOTE**
>
> You can also use an ST-LINK to upload the `.bin` file to RAK3172.

1. Once the hardware is now ready, you can open the STM32CubeProgrammer. Then you need to open the `LoRaWAN_End_Node.bin` FW file from this folder location `\STM32Cube_FW_WL_V1.0.0\Projects\NUCLEO-WL55JC\Applications\LoRaWAN\LoRaWAN_End_Node\STM32CubeIDE\Release` (for v1.0.0) or `\STM32Cube_FW_WL_V1.2.0\Projects\NUCLEO-WL55JC\Applications\LoRaWAN\LoRaWAN_End_Node\STM32CubeIDE\Release` (for v1.2.0).

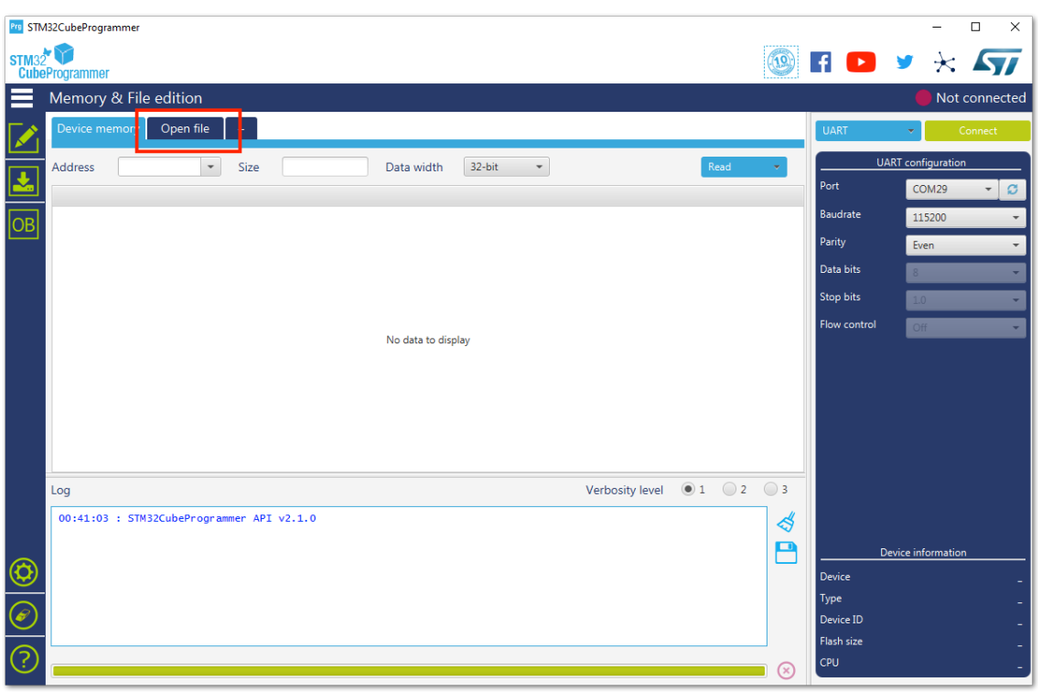

Figure 54: STM32CubeProgrammer Application

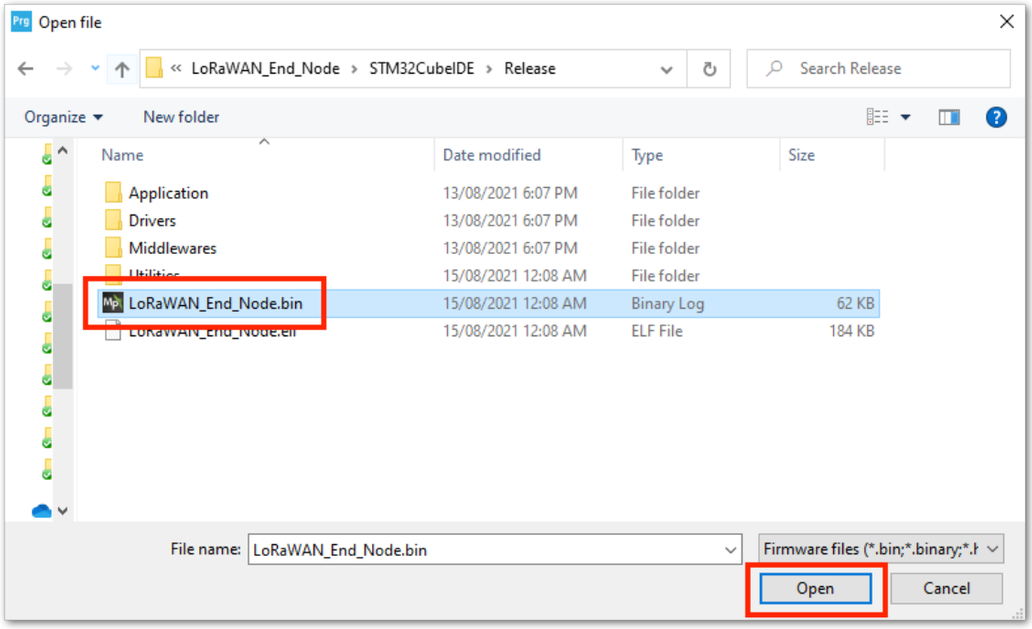

Figure 55: Locating the .bin Firmware File

2. If everything is successful, you should now see the `LoRaWAN_End_Node.bin` FW file in the STM32CubeProgrammer.

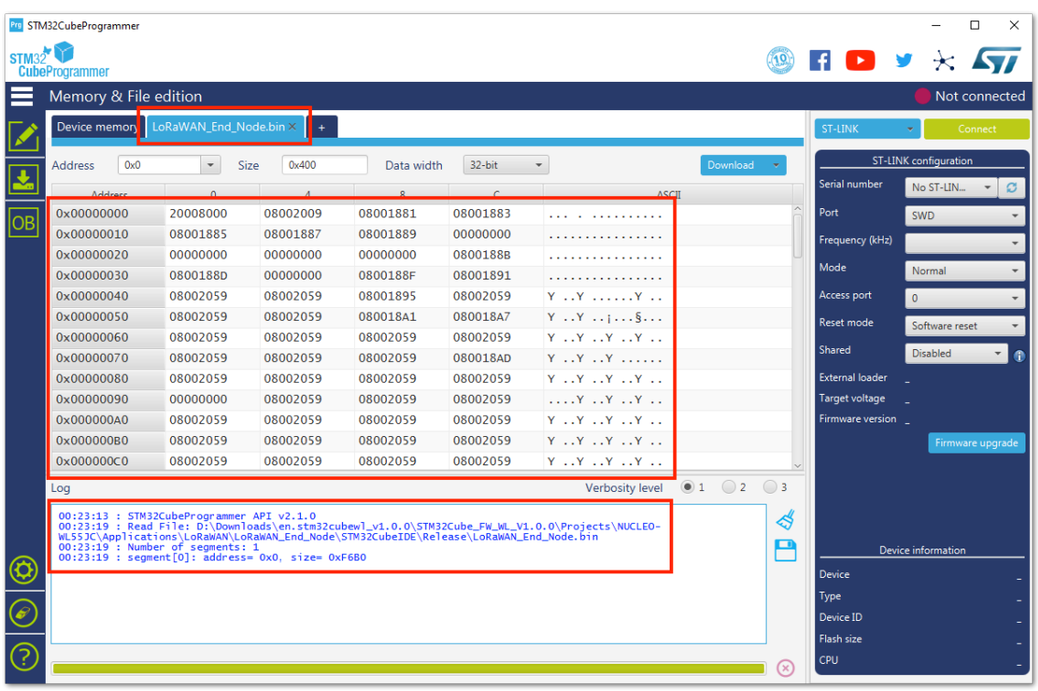

Figure 56: Loaded .bin Firmware File to the STM32CubeProgrammer

3. By default, the STM32CubeProgrammer chooses ST-LINK as the uploading interface, so you need to change it to UART and select the right COM port. After setting up the UART connection, you can now connect and see that the device is detected.

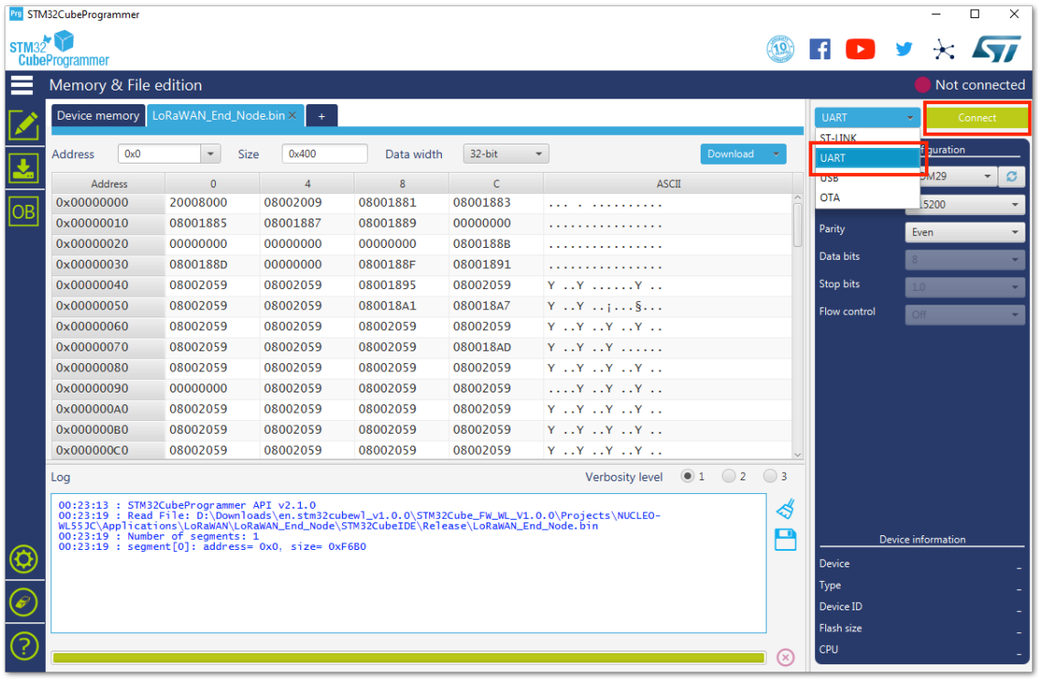

Figure 57: Selecting UART as Programming Interface

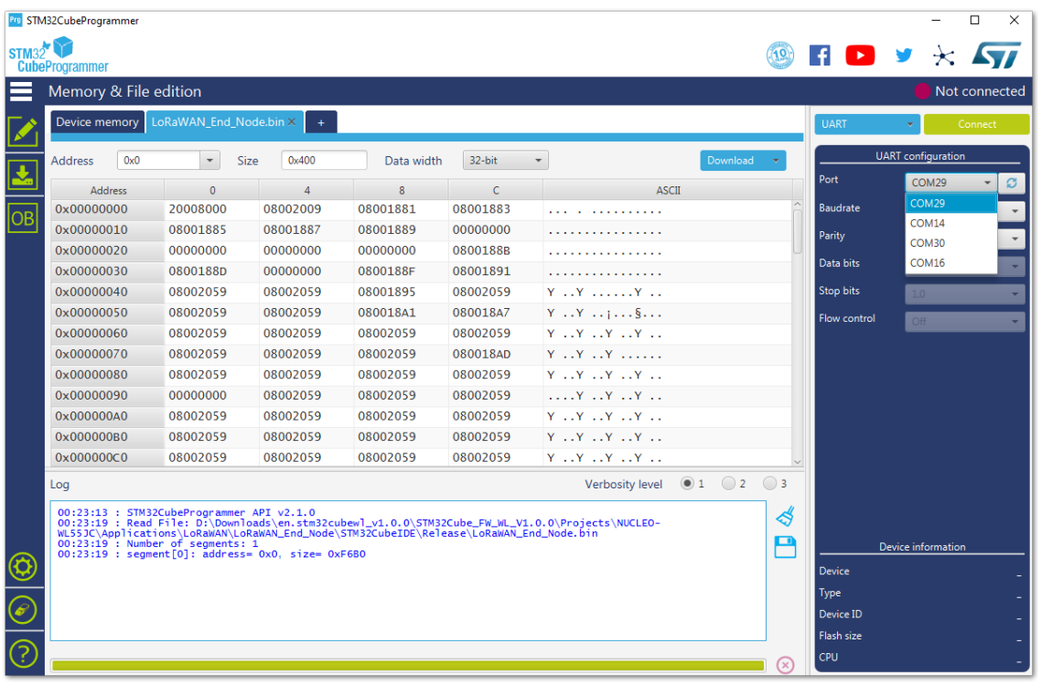

Figure 58: Selecting the Right COM Port

4. You need to ensure that the `Boot0` is connected to VDD (3.3 V) when the device is powered up, else, the STM32CubeProgrammer might not detect the device. The logs of a detected device are shown in **Figure 59**.

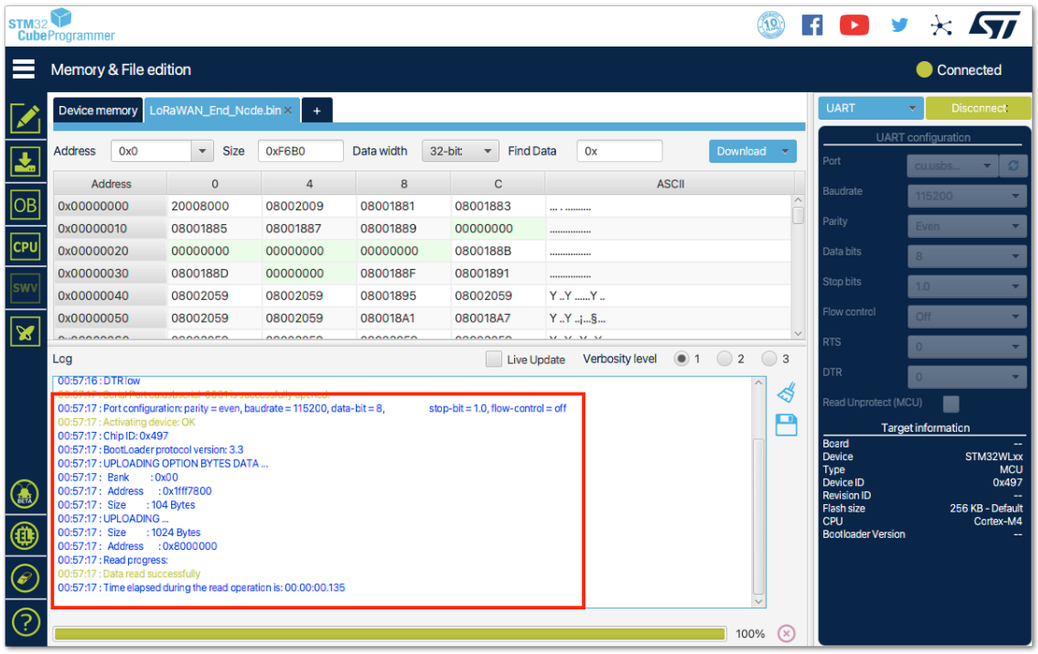

Figure 59: RAK3172 Detected by STM32CubeProgrammer

5. If the device is detected by the STM32CubeProgrammer, you can now upload the firmware by clicking **Download**.

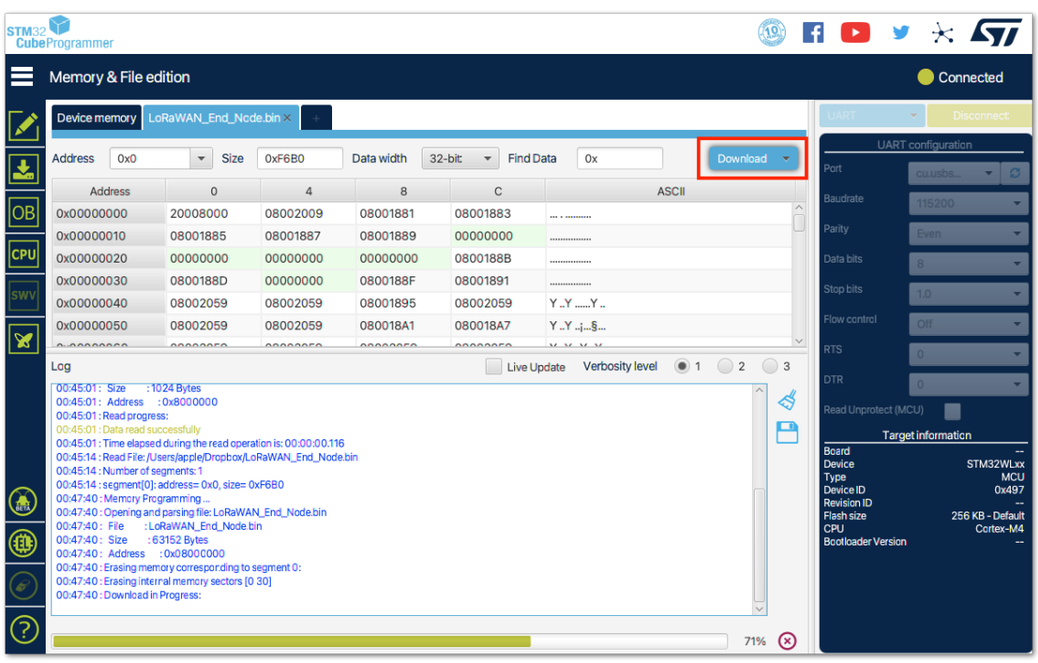

Figure 60: Firmware Uploading in Progress

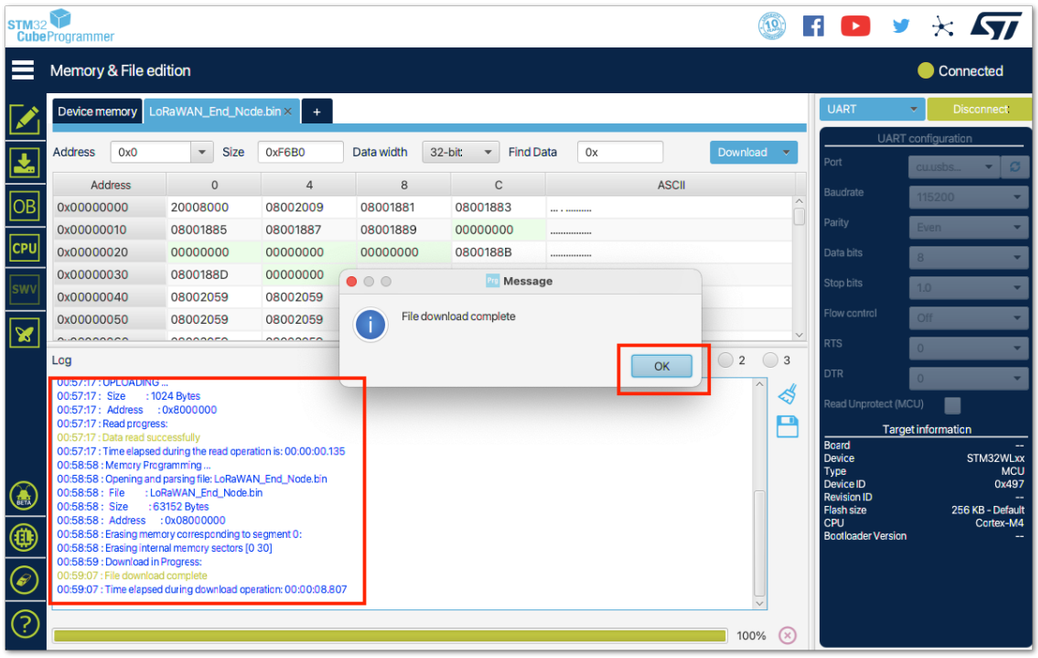

Figure 61: Firmware Successfully Uploaded

6. After the successful download, restart the device and remove the connection of the Boot0 pin to VDD (3.3 V), leaving you only with four-pin connections (power supply lines and UART2) as shown in **Figure 62**.

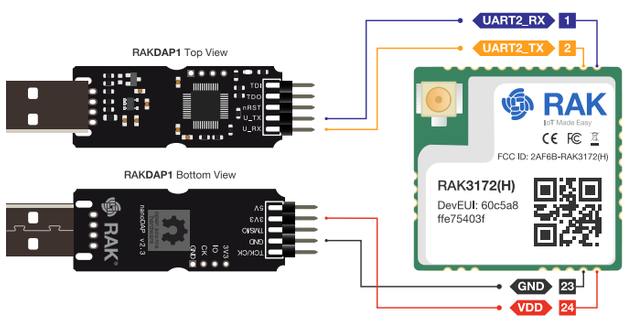

Figure 62: RAK3172 Connection to UART

7. By using Serial Terminal software, check the serial output logs of the RAK3172 with the newly uploaded FW with baud rate setting 115200. You should be able to see the serial logs, as shown in **Figure 63**.

```
APP_VERSION:        V1.0.0
MW_LORAWAN_VERSION: V2.2.1
MW_RADIO_VERSION:   V0.6.1
###### OTAA ######
###### AppKf±APP_VERSION:        V1.0.0
MW_LORAWAN_VERSION: V2.2.1
MW_RADIO_VERSION:   V0.6.1
###### OTAA ######
###### AppKey:  11 11 11 11 11 11 11 11 66 66 66 66 66 66 66 66
###### NwkKey:  2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C
###### ABP  ######
###### AppSKey: 2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C
###### NwkSKey: 2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C
###### DevEui:  22-22-22-22-88-88-88-88
###### AppEui:  11-11-11-11-99-99-99-99
0s012:TX on freq 868300000 Hz at DR 0
1s496:MAC txDone
6s518:RX_1 on freq 868300000 Hz at DR 0
6s645:PRE OK
7s191:HDR OK
8s337:MAC rxDone

###### = JOINED = OTAA ====================
10s014:temp= 29

10s014:VDDA= 254
10s015:TX on freq 867500000 Hz at DR 0
10s016:SEND REQUEST
11s664:MAC txDone
16s685:RX_1 on freq 867500000 Hz at DR 0
16s812:PRE OK
17s358:HDR OK
17s849:MAC rxDone

###### ========== MCPS-Confirm =============
20s014:temp= 30
```

Figure 63: RAK3172 UART2 Logs

8. With the device registered to TTN, you should now see a successful join and LoRaWAN device uplink.

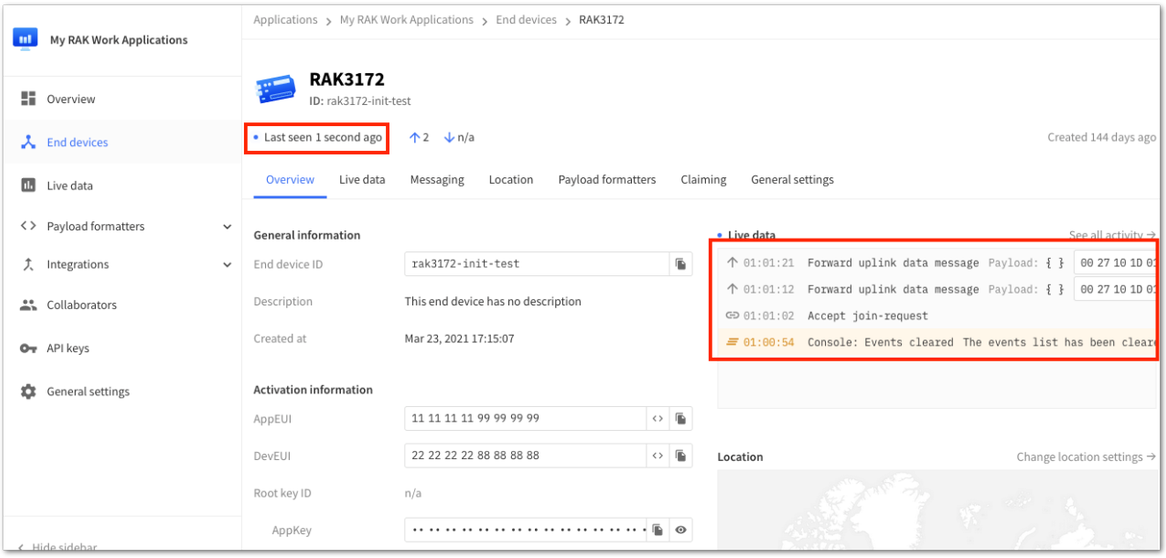

Figure 64: RAK3172 in TTN V3 Join and Uplink

Last Updated: 3/15/2023, 2:27:00 PM