Running the Semtech LoRa® Demo on a Nordic DevKit

This article describes how to setup the Nordic development environment, build the LoRa® demo application, and download it to the target board. The target board is a Connected Development SX1262 LoRa® shield attached to a Nordic nRF52840-DK board via the Arduino connector.

Overview

The development environment includes:

- Windows 10
- Nordic nRF Command Line Tools
- Nordic nRF Connect SDK
- Microsoft Visual Studio Code IDE (VS Code)
- Nordic nRF Extensions to VS Code
- git

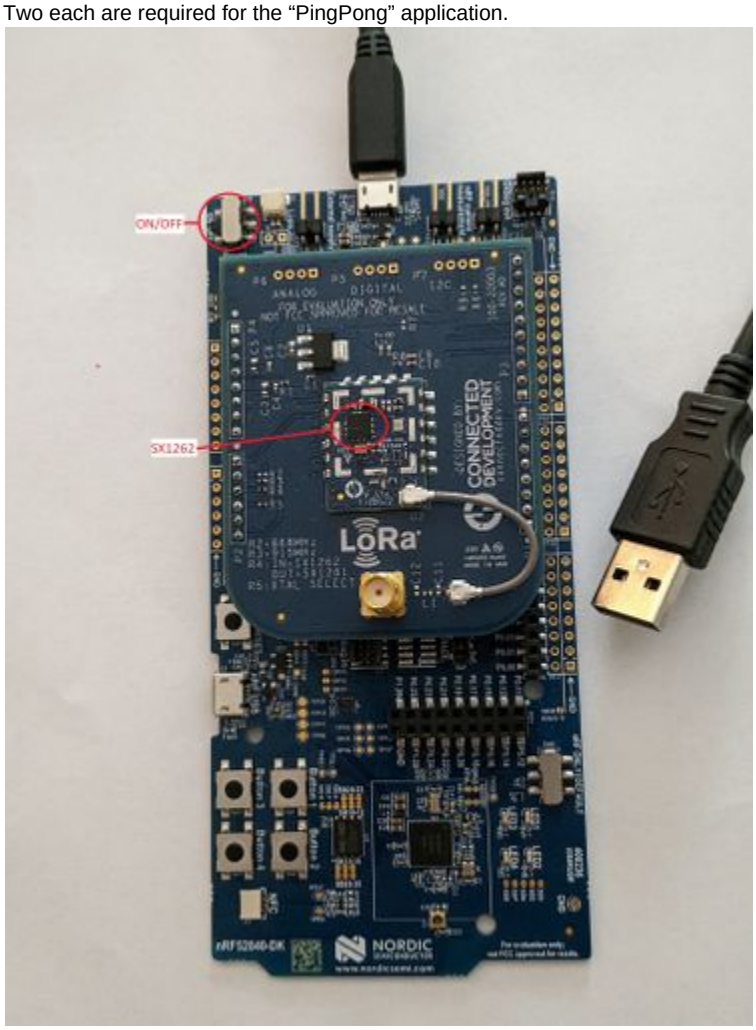
Two demo applications are supported:

1. PingPong – This is a point-to-point example. It was derived from a Semtech ping-pong sample application running on two ST Micro Nucleo boards, then ported to the Nordic nRF Connect SDK.
2. lorawan – This is a LoRaWAN® Class A example. It was derived from the sample in the Nordic nRF Connect SDK that is based on the Semtech LoRaMac implementation.

The target hardware includes:

- Nordic nRF52840-DK development board.
- Connected Development SX1262 LoRa® radio board mounted to a carrier board with Arduino connectors and attached to the nRF52840-DK (Figure 1).

Two each are required for the “PingPong” application.



References

1. Nordic self-paced hands-on online course: https://academy.nordicsemi.com/courses/nrf-connect-sdk-fundamentals
2. nRF Connect SDK Getting Started: https://www.nordicsemi.com/Products/Development-software/nRF-Connect-SDK/GetStarted#infotabs
3. Git version control system: https://git-scm.com
4. nRF Connect Workflow 4 – Application as the manifest repository: https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/dm_adding_code.html#workflow-4-application-as-the-manifest-repository
5. Working with GitHub forks: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks
6. Fork a repo quick start: https://docs.github.com/en/get-started/quickstart/fork-a-repo

Install the Nordic tools and VS Code

The Nordic instructions say that either the Segger IDE or Visual Studio Code can be used. Connected Development has chosen to use VS Code.

Follow the Nordic procedures to install the nRF Command Line Tools, nRF Connect SDK, and VS Code. If you are new to the Nordic tools, you can follow the online course (Reference 1) to install the tools and learn how to build a project. If you don't want to complete the whole course now, you can stop after building the first “blinky” project. This will confirm you have the tools installed correctly and can perform builds.

Alternately, the tools can be installed directly (Reference 2).

> The default installation folder for the SDK is under the user’s home folder. But it is recommended to override the default location to a directory close to the root, such as “C:\Nordic\v2.1.0\” (or whatever the SDK version is).

Install Git

If you don’t already have Git, install it for your Windows 10 system (Reference 3).

You can then run Git Bash, which gives the more traditional ‘Linux’ style interface, or Git CMD, which is a more ‘windows’ style command interface.

- Setup Git username and email (set to existing GitHub account):

```
> git config --global user.name "John Doe"
> git config --global user.email johndoe@connecteddev.com
```

- Make the working directory and navigate to it:

```
> mkdir C:\Source
> cd C:\Source
```

Git the application code

- Clone the repo branch to the working directory. The applications are in the GitHub CD repo “CD_Semtech_Demo". The latest version is on the default “main" branch:

```
C:\Source> git clone https://github.com/rdaubencd/CD_Semtech_Demo.git
```

- Command output sample:

```
Cloning into 'CD_Semtech_Demo'...
remote: Enumerating objects: 47, done.
Receiving objects: 100% (47/47), 16.73 KiB | 4.18 MiB/s, done.
Resolving deltas: 100% (14/14), done.)
remote: Counting objects: 100% (47/47), done.
remote: Compressing objects: 100% (34/34), done.
remote: Total 47 (delta 14), reused 32 (delta 5), pack-reused 0
```

West workspace explained

The build process uses a west workspace project which clones the Nordic SDK as part of the build process. This goes beyond the scope of the basic “blinky” freestanding project that was demonstrated during the initial environment setup above. This process is based on the Nordic Development Model Workflow 4, described in Reference 4.

The "PingPong" project contains the main manifest, which means it contains the top-level *west.yml* manifest file in its root. The main manifest file contains the list of other repos and their revisions for the project build. The first repo is the nRF Connect SDK “nrf-sdk”, which is itself a manifest repo that references all other repos to be used in the build. Next, the main manifest file references other repos that have been forked and modified by Connected Development for the project. These will override the unmodified repos that are referenced in the nRF Connect SDK manifest.

As of this writing, there is only one SDK repo that is modified by CD for this demo project:

Original Nordic	Repo	https://github.com/zephyrproject-rtos/loramac-node
	SDK File	modules/lib/loramac-node/src/radio/sx126x/radio.c
CD modified	Repo	https://github.com/rdaubencd/zephyr-loramac-node
	Local path	ConDev/modules/lib/loramac-node/src/radio/sx126x/radio.c

The entire forked repo will be included in the build, not just the modified files. This is because the repo is built as a library and all of its files must reside in the same folder.

For more information about forking and modifying a repo, see Appendix 1 below.

Update the workspace

Use the “west” command to clone all the repos that are referenced in the project's west.yml. The unmodified nRF Connect repos will be pulled down from Nordic's public repos on GitHub, and the CD modified repos will be pulled down from the CD repos. The update might take a few minutes!

If west.exe is not in your path, it is in the *toolchains* folder where the Nordic SDK is installed. For example: C:\Nordic\toolchains\v2.1.0 \opt\bin\Scripts\west.exe

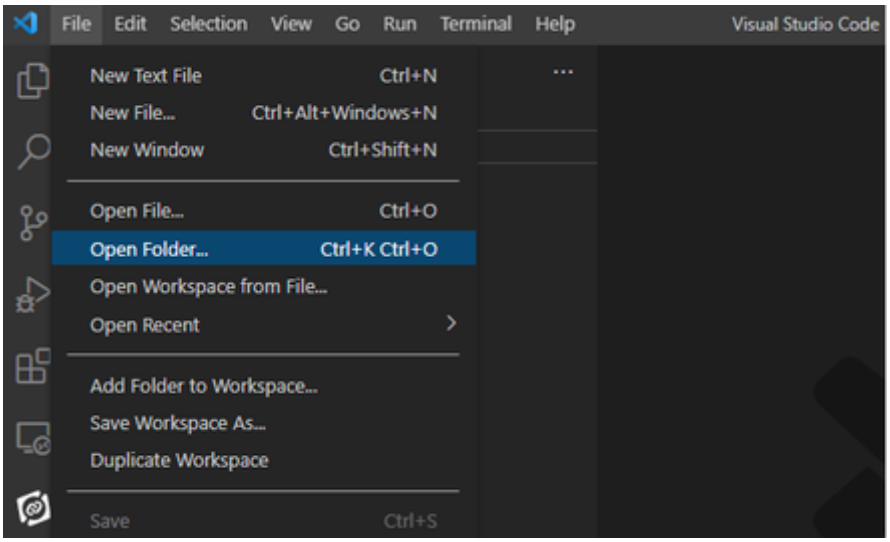
```
>cd CD_Semtech_Demo
>west update
```

Repeat the west update if any changes are ever made to the *west.yml* file.

Open the application folder in VS Code

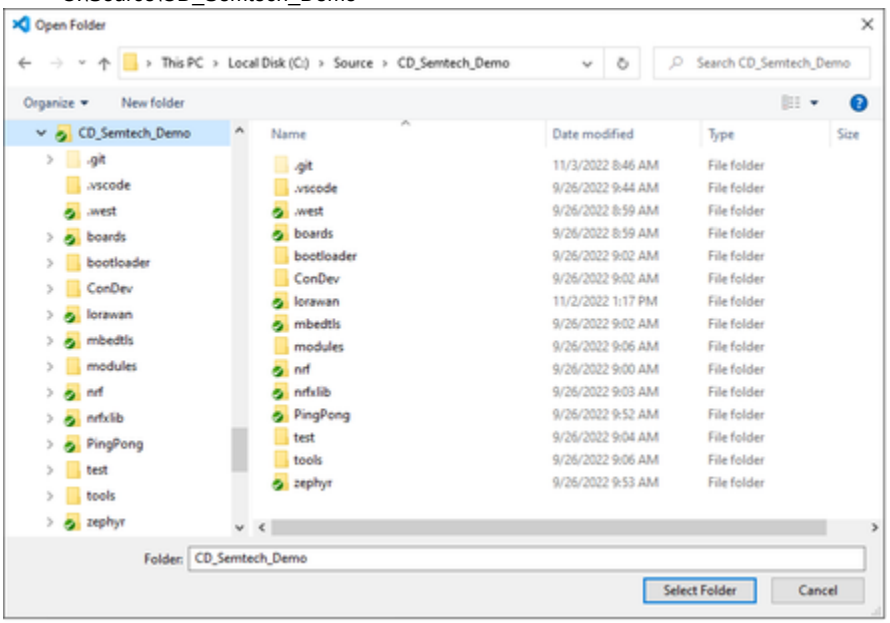
This step chooses the folder that VS Code will use as its workspace.

- Go to “File Open Folder…”



- Navigate to the sub-directory of the application at:

  C:\Source\CD_Semtech_Demo



- Hit “Select Folder”.

If you followed the Nordic online course to build the sample “blinky” application, the following steps are the same. Only the specific application is different.

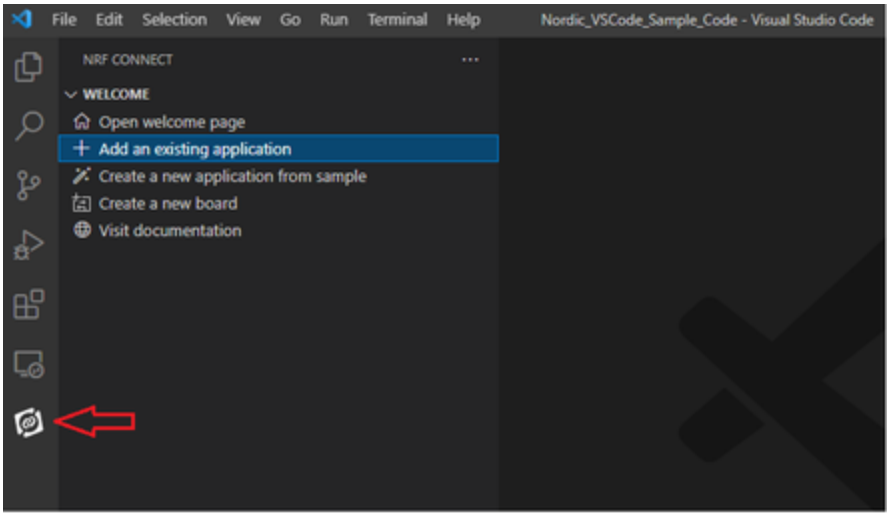
Confirm or set the following variables in the VS Code settings for the workspace (Ctrl+,) . Use your actual toolchains installation path:

- nrf-connect.toolchain.path: C:\Nordic\toolchains\v2.1.0
- nrf-connect.topdir: ${workspaceFolder}

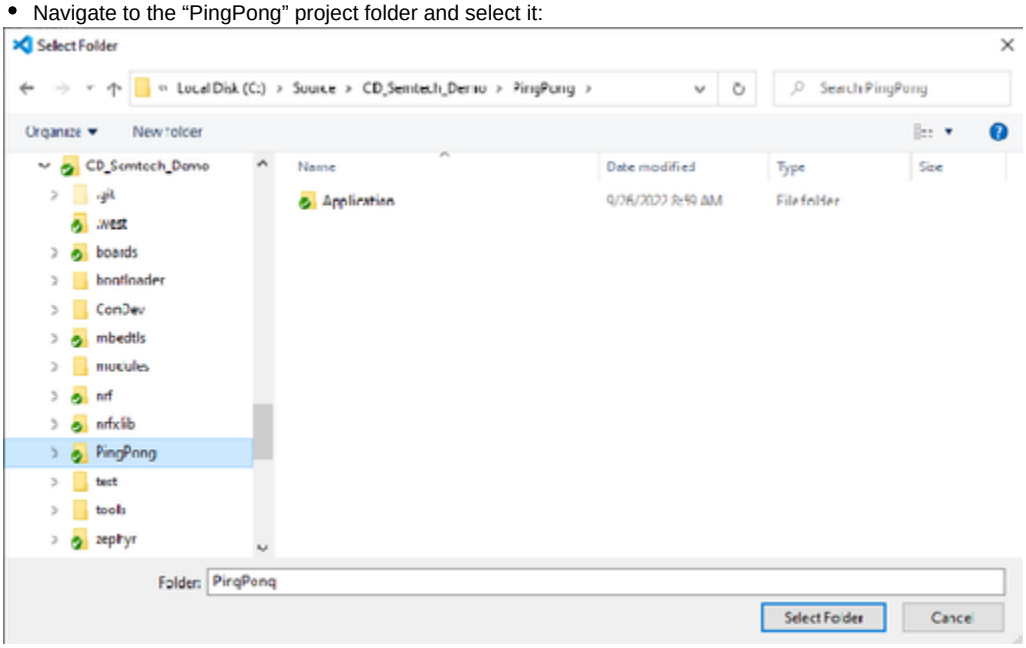
Add the application(s)

This step adds the “PingPong” project to the workspace.

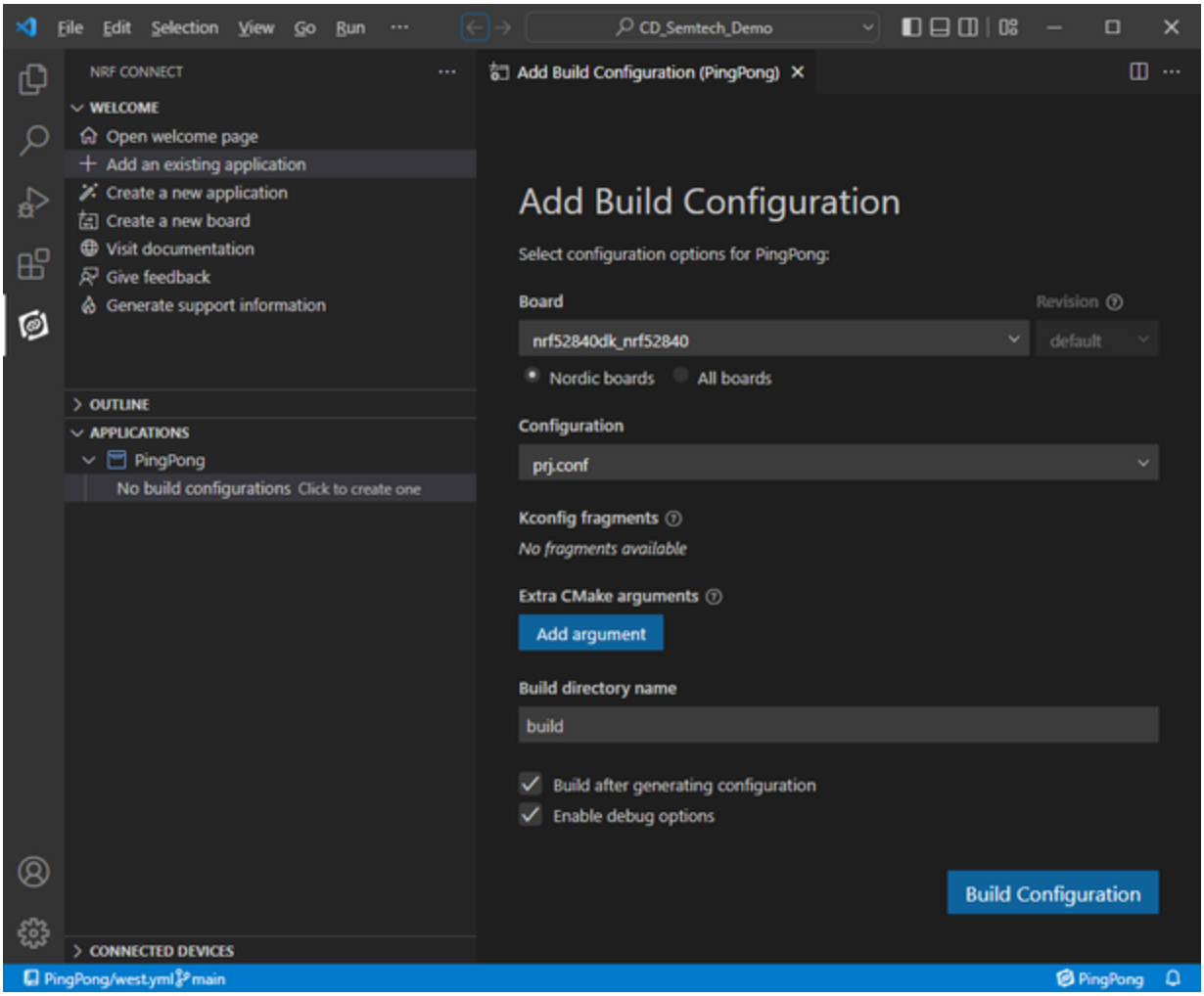
- Click on the “nRF Connect” tab in the left side Activity bar.
- Select “Add an existing application” under the WELCOME section:



- Navigate to the “PingPong” project folder and select it:



- Hit “Select Folder”. Ignore the warning about the west workspace being out of sync.
- Now the “PingPong” project shows up under the APPLICATIONS section:



LoRaWAN® Application

The “lorawan” application can be added in the same way. Navigate to the “lorawan/class_a” project folder and select it. Both applications can co-exist in the same workspace and share the same copy of the Nordic SDK.

Prior to building, the device credentials must be configured for Over-The-Air-Activation (OTAA). Open the src/*main.c* file and change the following values to your specific case:

```
#define LORAWAN_DEV_EUI       { 0x00, 0x80, 0x00, 0x00, 0x04, 0x01,
0xdd, 0x40 }
#define LORAWAN_JOIN_EUI      { 0xa2, 0xb3, 0x84, 0x25, 0xcf, 0xb6,
0xf7, 0xfe }
#define LORAWAN_APP_KEY       { 0x87, 0x47, 0xcc, 0xc8, 0xce, 0x01,
0xd2, 0x96, 0x2d, 0x5f, 0x94, 0x60, 0x0b, 0xcd, 0x38, 0xcf }
```

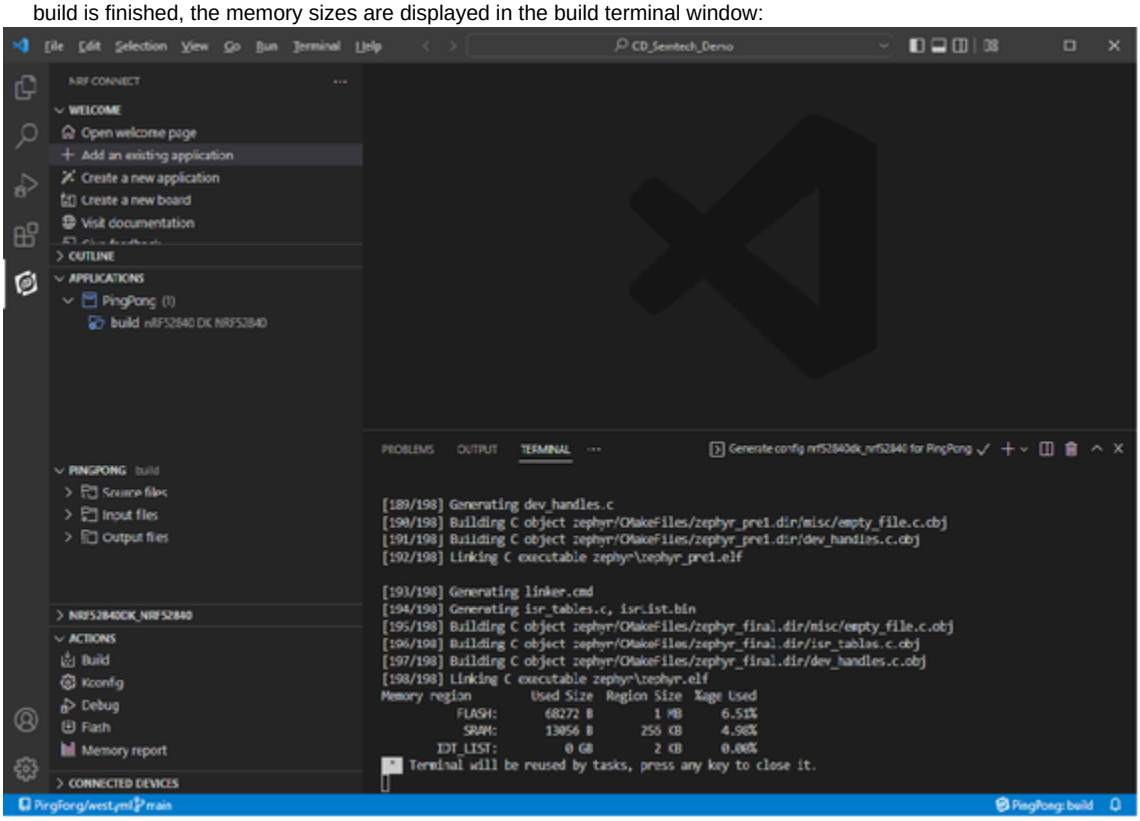
By default, the application region is configured for the US915 band. This can be changed in the project’s *prj.conf* file.

Create a new build configuration

There is no build configuration created by default. One must be created prior to building the project. Refer to Figure 6 above for the following steps.

- Click on “Click to create one” to add a new build configuration.

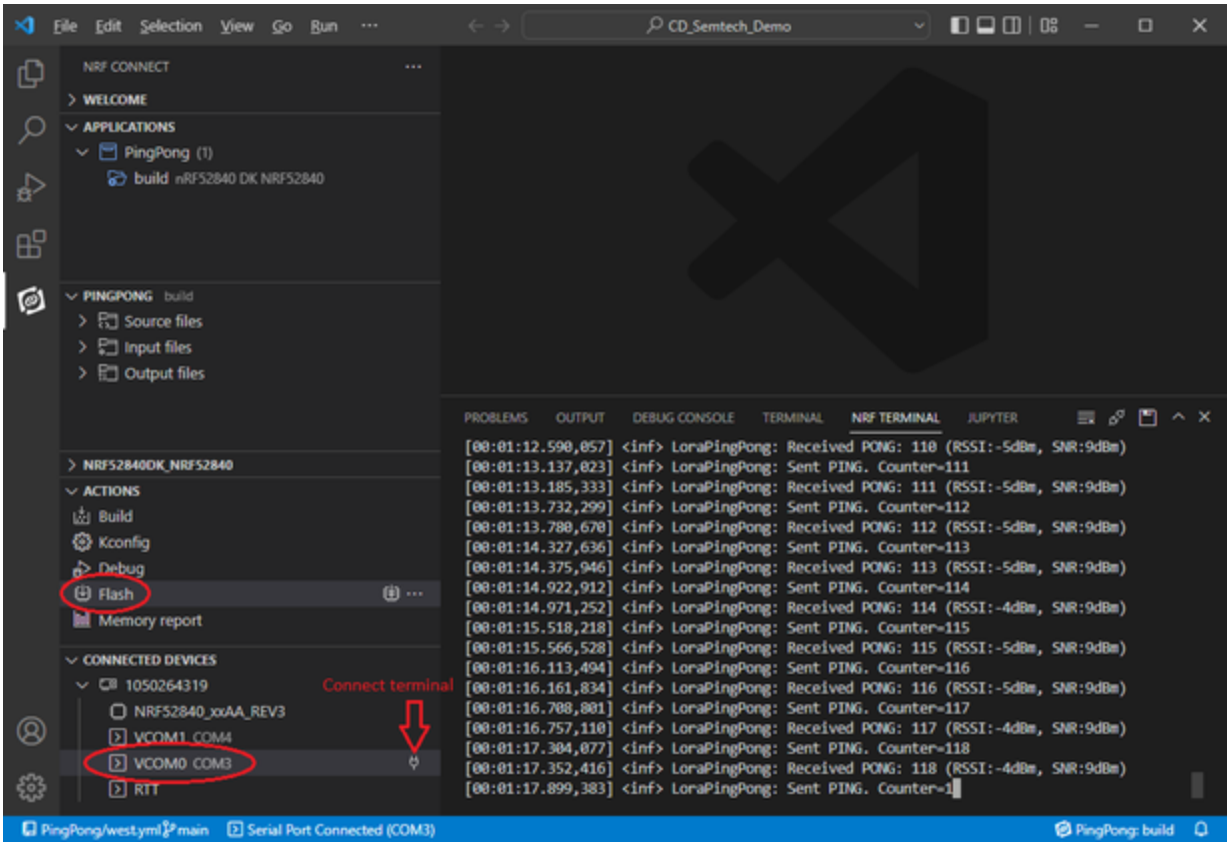
- On the right side, configure the build. Open the Board drop list and select the “nrf52840dk_nrf52840” board.
- Select other options as shown above. When completed, click on “Build Configuration” to finish.
- Since we selected the option to “Build after generating configuration”, the project will begin building immediately. A manual build can be started by clicking on “Build” under the ACTIONS section.
- Open the output terminal window to see the build progress by clicking on “building” in the build status line in the lower right corner. When the build is finished, the memory sizes are displayed in the build terminal window:



Download to the target

The last step is to download the build to the target board and run it.

- Connect a USB cable from the end connector on the Nordic nRF52840-DK board to the PC, as shown in Figure 1.
- Slide the ON/OFF switch to ON.
- In VS Code, open the CONNECTED DEVICES section and connect to the board:



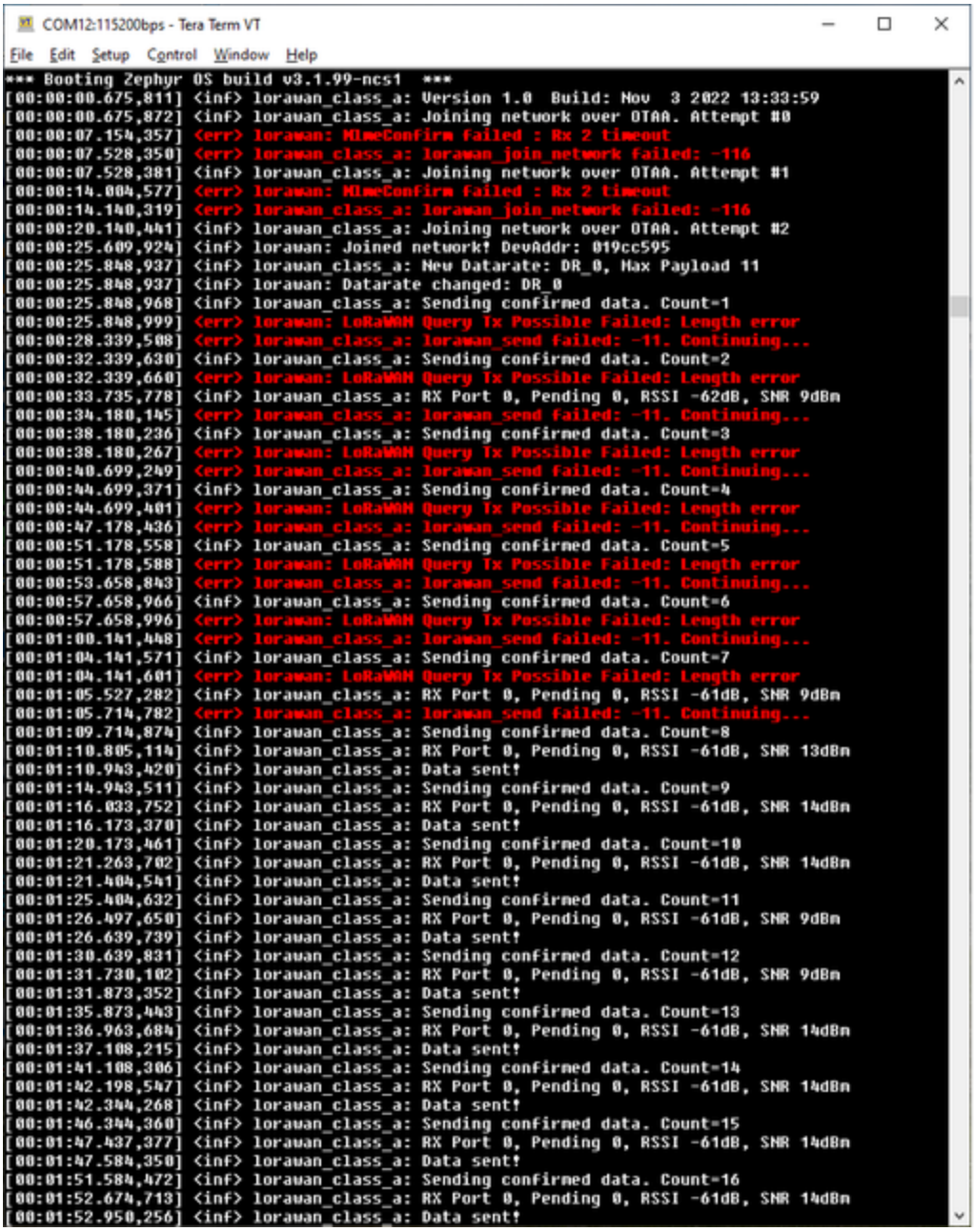
- Click on the small plug icon at right side of the “VCOM0” port to open an NRF TERMINAL window. Select the “115200 8n1 rtscts:off” configuration. The application’s log messages will be displayed here.
- Under the ACTIONS section, click on “Flash” to download the build to the board.
- After the download is finished, the board will be automatically reset and the application will run.

PingPong

- Repeat the download to the 2nd Nordic nRF52840-DK board and shield.
- If desired, open a 2nd NRF TERMINAL window to see the logs for the other board.
- A log is displayed in the NRF TERMINAL window for each LoRa® packet that is sent and received, including the packet counter (see Figure 8). Each receive log includes the packet counter, the signal strength (RSSI), and Signal-to-Noise Ratio (SNR).
- LED1 will toggle each time a packet is received, and LED2 will toggle each time a packet is sent.

LoRaWAN®

- The LoRa® Join Server must be configured with the device credentials that are configured in *main.c* (DEV_EUI, JOIN_EUI, and APP_KEY).
- Log messages are displayed in the NRF TERMINAL window (see Figure 9). The example application performs an OTAA Join, then sends out data packets containing "helloworldxxxx" once every 5 seconds. The "xxxx" data is an incrementing counter to make it easier to track a specific packet through the gateway, or look for dropped packets.



- The Join Request might take several attempts before success. This is because the device sends the Join Request on a random channel, which might not be supported by the gateway. Each attempt chooses a different random channel until it finds a channel that is supported by the gateway. The above sample failed the first two attempts, then succeeded on the third attempt.
- Once joined, data packets are sent out, using the lowest data rate, which does not support the payload length of the test data. After a few packets are sent, the gateway will send down a confirm that includes a new data rate that allows a larger payload size. After that, the data packets are sent out successfully and each is confirmed by the gateway.
- Each received packet confirm message is logged and includes the RSSI and SNR.

Appendix 1 – Create a GitHub fork copy of the repo to be modified

For every SDK file that needs to be modified, we must create a fork copy of its parent repo, branch it for our changes, and update the manifest file to point to the forked version. The entire forked repo will be included in the build, not just the modified files. This is because the repo is built as a library and all of its files must reside in the same folder.

Forking a public repo creates the fork with public access, so it can be shared with anyone. For background information, see Reference 5 – Working with GitHub forks.

Create a new fork using the GitHub web page. Browse to the parent repo to be forked. Follow the quick start instructions at Reference 6 – “Fork a repo”. Then the forked repo can be cloned to your local file system. Then you can branch, modify, commit, push, etc to the mirror repo.

A fork can be synced with its upstream parent to pick up new changes. This is done on the GitHub web page for the forked repo. For reference, see Reference 5, section “Syncing a fork”.

Attachments

This is the build output of the PingPong demo application. It can be flashed to the target board using the Nordic Programmer, which can be accessed from the nRF Connect for Desktop application.