XM126 Software

User Guide

XM126 Software

User Guide

Author: Acconeer AB

Version:a121-v1.10.0

Acconeer AB March 26, 2025

Contents

1 Acconeer SDK Documentation Overview

To better understand what SDK document to use, a summary of the documents are shown in the table below.

Table 1: SDK document overview.

Name	Description	When to use
	RSS API documentation (html)	
rss_api	The complete C API documentation.	- RSS application implementation - Understanding RSS API functions
	User guides (PDF)	
A121 Assembly Test	Describes the Acconeer assembly test functionality.	- Bring-up of HW/SW - Production test implementation
A121 Breathing Reference Application	Describes the functionality of the Breathing Reference Application.	- Working with the Breathing Reference Application
A121 Distance Detector	Describes usage and algorithms of the Distance Detector.	- Working with the Distance Detector
A121 SW Integration	Describes how to implement each integration function needed to use the Acconeer sensor.	- SW implementation of custom HW integration
A121 Presence Detector	Describes usage and algorithms of the Presence Detector.	- Working with the Presence Detector
A121 Smart Presence Reference Application	Describes the functionality of the Smart Presence Reference Application.	- Working with the Smart Presence Reference Application
A121 Sparse IQ Service	Describes usage of the Sparse IQ Service.	- Working with the Sparse IQ Service
A121 Tank Level Reference Application	Describes the functionality of the Tank Level Reference Application.	- Working with the Tank Level Reference Application
A121 Touchless Button Reference Application	Describes the functionality of the Touchless Button Reference Application.	- Working with the Touchless Button Reference Application
A121 Parking Reference Application	Describes the functionality of the Parking Reference Application.	- Working with the Parking Reference Application
A121 STM32CubeIDE	Describes the flow of taking an Acconeer SDK and integrate into STM32CubeIDE.	- Using STM32CubeIDE
A121 Raspberry Pi Software	Describes how to develop for Raspberry Pi.	- Working with Raspberry Pi
A121 Ripple	Describes how to develop for Ripple.	- Working with Ripple on Raspberry Pi
XM125 Software	Describes how to develop for XM125.	- Working with XM125
XM126 Software	Describes how to develop for XM126.	- Working with XM126
I2C Distance Detector	Describes the functionality of the I2C Distance Detector Application.	- Working with the I2C Distance Detector Application
I2C Presence Detector	Describes the functionality of the I2C Presence Detector Application.	- Working with the I2C Presence Detector Application
I2C Breathing Reference Application	Describes the functionality of the I2C Breathing Reference Application.	- Working with the I2C Breathing Reference Application
	A121 Radar Data and Control (PDF)	
A121 Radar Data and Control	Describes different aspects of the Acconeer offer, for example radar principles and how to configure	- To understand the Acconeer sensor - Use case evaluation
	Readme (txt)	
README	Various target specific information and links	- After SDK download

2 Introduction

The Acconeer Software Development Kit (SDK) enables customers to develop their own software that can be executed on the module. This enables full control of all the peripherals and to maximize the performance and power consumption for a specific use case.

The SDK comes with a number of example applications that can be used as a starting point when developing your own application. These applications can be downloaded and executed using the methods described in “Installing Software Image” at page 10.

When developing your own application we recommend that you setup a development environment as described in “Setting up a Development Environment” at page 11.

This guide has been verified in Ubuntu 20.04 and Windows with nRF Connect for Desktop v5.1.0 and nRF Connect SDK v2.9.1

3 Zephyr in XM126

The XM126 SDK is developed using the nRF Connect SDK from Nordic Semiconductor and integrates the Zephyr real-time operating system. The Zephyr RTOS is based on a small-footprint kernel designed for use on resource-constrained and embedded systems.

The XM126 SDK will use the build and configuration system in the nRF Connect SDK described here.

The XM126 SDK consists of several components that is standard in an nRF Connect based SDK as well as Acconeer specific components:

```
├── boards
├── doc
├── dts
├── include
├── integration
├── lib
├── LICENSES
├── samples
└── source
```

3.1 XM126 Device Definitions

The XM126 device and the XB122 shield is defined in KConfig, defconfig and device tree files where the most important files are “xm126.dts” and “acconeer_xb122.overlay”. Starting with nRF Connect SDK version 2.7.0, hardware model v2 is used to name SoCs and boards.

```
├── boards
│   ├── acconeer
│   │   └── xm126
│   │       ├── board.cmake
│   │       ├── board.yml
│   │       ├── Kconfig
│   │       ├── Kconfig.xm126
│   │       ├── pre_dt_board.cmake
│   │       ├── xm126.dts
│   │       └── xm126.yaml
│   └── shields
│       └── acconeer_xb122
│           ├── acconeer_xb122.overlay
│           └── Kconfig.shield
└── dts
    └── bindings
        ├── acc,a121.yaml
        ├── acc,a121_hal.yaml
        └── vendor-prefixes.txt
```

3.2 XM126 Sample Applications

Each sample application is configured to be used together with a bootloader. The bootloader in XM126 is based on MCUBoot and is described in detail here. The configuration of the bootloader can be found in “sysbuild/mcuboot/prj.conf”.

The XM126 module is delivered pre-flashed with the same bootloader that will be built in the SDK.

Configuration of each sample application is done in the provided “prj.conf” and “CMakeLists.txt” files.

```
└── samples
    └── example_detector_distance_ble_beacon
        ├── sysbuild
        │   ├── mcuboot
        │   └── prj.conf
        └── CMakeLists.txt
```

```
└─prj.conf
```

More information about Application Development in the nRF Connect SDK can be found here.

3.3 BLE beacons

Some of the examples include advertisement of BLE beacons. The beacons implemented in the XM126 SDK are all utilizing 'Extended Advertisements' with flags: 'General Discovery' and 'BR/EDR not supported' set. The advertisement type is 'Manufacturer Specific Data' (type 0xFF). Manufacturer specific data is used to add any custom data into advertising packets, using any format that is suitable for your application.

Table 2: Acconeer BLE beacon format.

Application	**Manufacturer**	**Acc Beacon Type**	**Application Data**
	2 bytes	1 byte	
example_detector_distance_ble_beacon	0xAC 0xC0	0x90	See Table 3
example_detector_presence_ble_beacon	0xAC 0xC0	0x91	See Table 4
ref_app_tank_level_ble_beacon	0xAC 0xC0	0x92	See Table 5
ref_app_parking_ble_beacon	0xAC 0xC0	0x94	See Table 7

Please note that all data types that consists of more than one byte is re-arranged in little-endian format. The Manufacturer id in the above table (0xACC0) is actually transmitted as 0xC0AC. This applies for the application data below as well.

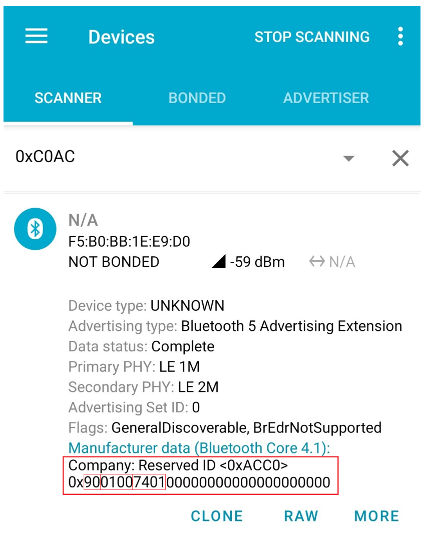
3.3.1 Distance

The application data for the distance example is structured in the BLE beacon in the following way:

Table 3: Application Data, Distance.

Nbr distances	**Distance 1 (mm)**	**Distance 2 (mm)**	**...**	**Distance x (mm)**
2 bytes	2 bytes	2 bytes	...	2 bytes

An example of a beacon with 1 measured distance at 372 mm (0x0174).



0xC0AC

N/A
F5:B0:BB:1E:E9:D0
NOT BONDED -59 dBm N/A

Device type: UNKNOWN
Advertising type: Bluetooth 5 Advertising Extension
Data status: Complete
Primary PHY: LE 1M
Secondary PHY: LE 2M
Advertising Set ID: 0
Flags: GeneralDiscoverable, BrEdrNotSupported
Manufacturer data (Bluetooth Core 4.1):
Company: Reserved ID <0xACC0>
0x900100740100000000000000000000

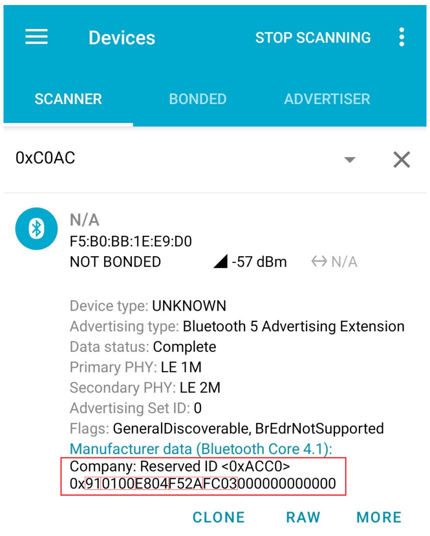
Figure 1: Distance Beacon

3.3.2 Presence

Table 4: Application Data, Presence.

Presence (bool)	Intra presence score (*1000)	Inter presence score (*1000)	Presence distance (mm)
2 bytes	2 bytes	2 bytes	2 bytes

An example of a beacon with presence detected at 1020 mm (0x03FC) with intra presence score 1.256 (0x04E8 / 1000) and inter presence score 10.997 (0x2AF5 / 1000).



N/A
F5:B0:BB:1E:E9:D0
NOT BONDED -57 dBm ↔ N/A

Device type: UNKNOWN
Advertising type: Bluetooth 5 Advertising Extension
Data status: Complete
Primary PHY: LE 1M
Secondary PHY: LE 2M
Advertising Set ID: 0
Flags: GeneralDiscoverable, BrEdrNotSupported
Manufacturer data (Bluetooth Core 4.1):
Company: Reserved ID <0xACC0>
0x910100E804F52AFC03000000000000

Figure 2: Presence Beacon

3.3.3 Tank Level

The application data for the tank level reference application is structured in the BLE beacon in the following way:

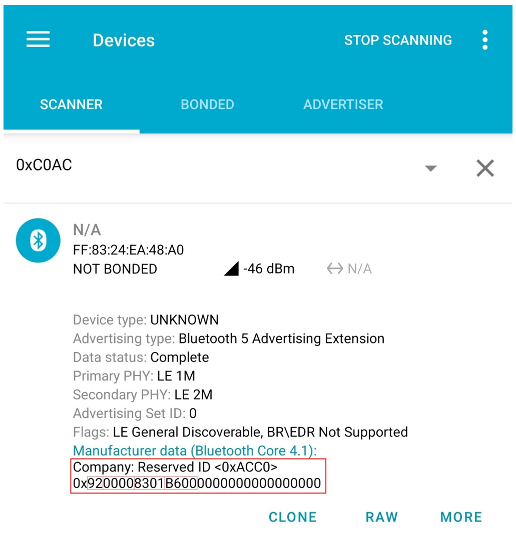
Table 5: Application Data, Tank Level.

Peak status	Level (‰)	Level (mm)
2 bytes	2 bytes	2 bytes

Table 6: Tank Level Peak Status.

Peak status	Name	Description
0x0000	In Range	Level detected within range
0x0001	No Detection	No level detected within range
0x0002	Overflow	Overflow detected
0x0003	Out Of Range	Level detected outside of range

An example of a beacon with a level detected at 182 mm (0x00B6) and 38.7 % (0x0183 / 10)



0xC0AC

N/A
FF:83:24:EA:48:A0
NOT BONDED -46 dBm N/A

Device type: UNKNOWN
Advertising type: Bluetooth 5 Advertising Extension
Data status: Complete
Primary PHY: LE 1M
Secondary PHY: LE 2M
Advertising Set ID: 0
Flags: LE General Discoverable, BR\EDR Not Supported
Manufacturer data (Bluetooth Core 4.1):
Company: Reserved ID <0xACC0>
0x9200008301B6000000000000000000

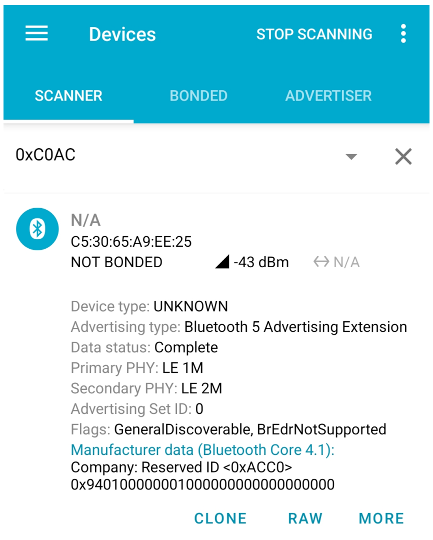
Figure 3: Tank Level Beacon

3.3.4 Parking

Table 7: Application Data, Parking.

Data Reliable (bool)	Obstruction Detected (bool, 0xFFFF if disabled)	Car Detected (bool)
2 bytes	2 bytes	2 bytes

An example of a beacon with car detected and no obstruction detected.



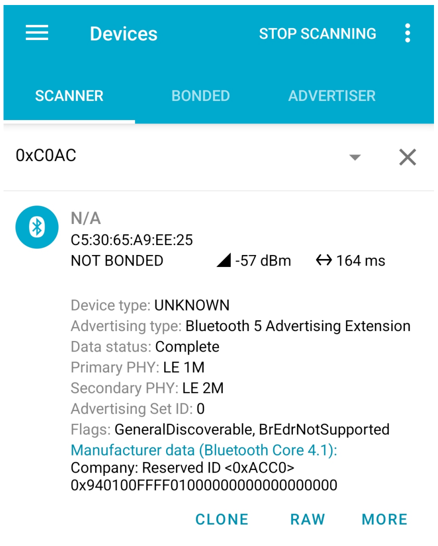
0xC0AC

N/A
C5:30:65:A9:EE:25
NOT BONDED -43 dBm N/A

Device type: UNKNOWN
Advertising type: Bluetooth 5 Advertising Extension
Data status: Complete
Primary PHY: LE 1M
Secondary PHY: LE 2M
Advertising Set ID: 0
Flags: GeneralDiscoverable, BrEdrNotSupported
Manufacturer data (Bluetooth Core 4.1):
Company: Reserved ID <0xACC0>
0x94010000000100000000000000000000

Figure 4: Parking Beacon

An example of a beacon with car detected and obstruction is disabled.



0xC0AC

N/A
C5:30:65:A9:EE:25
NOT BONDED -57 dBm ↔ 164 ms

Device type: UNKNOWN
Advertising type: Bluetooth 5 Advertising Extension
Data status: Complete
Primary PHY: LE 1M
Secondary PHY: LE 2M
Advertising Set ID: 0
Flags: GeneralDiscoverable, BrEdrNotSupported
Manufacturer data (Bluetooth Core 4.1):
Company: Reserved ID <0xACC0>
0x940100FFFF01000000000000000000

Figure 5: Parking Beacon, Obstruction disabled

3.4 Using BLE together with A121 on XM126

XM126 uses SPI on nRF52840 to communicate with the A121 radar sensor. When using BLE simultaneously as SPIM3, the transmit data sometimes gets corrupted.

There are a number of possible workarounds for this issue, each with their pros and cons, which will be shortly explained in the following subsections.

For more information regarding the anomaly, see "[198] SPIM: SPIM3 transmit data might be corrupted" in the Errata document for nRF52840 located at `https://infocenter.nordicsemi.com/`

3.4.1 Dedicate RAM for SPIM3 TX buffer when BLE is used (XM126 SDK default)

This is the default implemented solution in the XM126 SDK.

This solution allows BLE connections to remain open while still utilizing the 32MHz SPIM3 peripheral.
However, 8k bytes of RAM is dedicated to this.

3.4.2 Turn off BLE when using radar

The application can turn off BLE every time the radar is used, then enable BLE again once the radar is done.

This solution reduces RAM memory consumption and allows the radar to utilize the 32MHz SPIM3 peripheral.
However, the BLE connection can not remain open as long as the radar is used.

3.4.3 Use another SPI peripheral

The user can change SPI peripheral from SPIM3 to any of the other available SPI peripherals on nRF52840, i.e. SPIM0, SPIM1, SPIM2.

This solution reduces RAM memory consumption and BLE connections can remain open at all times.
However, SPI frequency is limited to 8MHz.

4 Installing Software Image

The Acconeer SDK for XM126 is configured to include a bootloader based on MCUboot. This will enable the user to upgrade the application over UART.

Another option is to use a SWD debugger, this requires additional hardware which is suitable when developing your own applications.

4.1 Windows COM port drivers

If running on Windows, you might need to install a driver for the USB to UART Bridge. It can be downloaded here.

4.2 Flash Over UART Using MCUmgr

Download and install MCUmgr.

4.2.1 Boot the XM126 in bootloader mode

1. Connect the XB122 to your PC with a USB-B Micro cable to the USB connector
2. Press and hold the “DFU” button on the board
3. Press the “RESET” button (still holding the “DFU” button)
4. Release the “RESET” button
5. Release the “DFU” button

Your XM126 device is now in “DFU” mode waiting for a software upgrade procedure to be started.

4.2.2 Program the XM126

1. Set the XM126 into bootloader mode, see above for how to do this
2. Create a connection with the correct port
3. Program the device
4. Press “RESET” or power cycle the device to start the embedded application

```
$ mcumgr conn add usb0 type="serial" connstring="dev=COM0,baud=115200,mtu
  =1024"
$ mcumgr -c usb0 image upload samples/example_bring_up/out/example_bring_up/
  zephyr/example_bring_up.signed.bin
```

5 Setting up a Development Environment

In order to develop your own applications you need to set up a development environment. The XM126 is based on a nRF52840 Bluetooth 5 SoC from Nordic Semiconductor.

5.1 Developing on XB122

When developing applications targeting XM126 it is recommended to do this using the XB122 Evaluation Board before switching over to the standalone XM126. The Acconeer SDK for XM126 is prepared to be compatible with XB122, meaning it will be possible to develop your application on XB122 and then seamlessly flash this application on an XM126.

5.2 Using a Debugger

In order to debug your applications it is recommended to use a SWD debugger. We recommend that you use a SEGGER JLink debug probe e.g. a J-Link BASE Compact debugger.

The debugger is connected to the Cortex 10-pin JTAG/SWD connector marked "SWD/JTAG" on XB122. The pin header has 1.27 mm pitch and an adapter typically called "ARM-JTAG-20-10" is needed between the debugger and XB122.

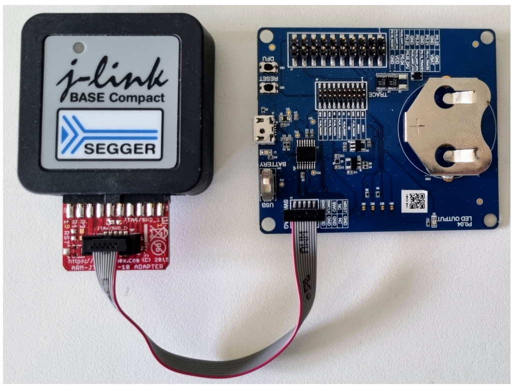


Figure 6: JTAG/SWD Connection

The J-Link BASE Compact can be used to set breakpoints and single step the program in an easy way.

5.3 nRF Connect SDK

Using nRF Connect SDK and nRF Command Line Tools is recommended when you develop applications for XM126. See www.nordicsemi.com

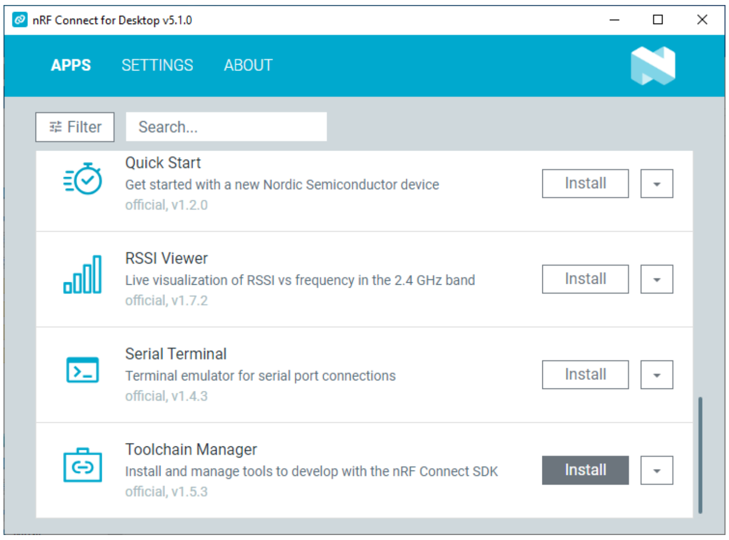
5.3.1 Install nRF Command Line Tools

The nRF Command Line Tools is used for development, programming and debugging. Download and install nRF Command Line Tools from www.nordicsemi.com

5.3.2 Install nRF Connect SDK

The most convenient way to install the nRF Connect SDK is through nRF Connect for Desktop.

1. Download nRF Connect for Desktop from www.nordicsemi.com
2. Start nRF Connect for Desktop and install Toolchain Manager.
3. Open Toolchain Manager and install the toolchain version of your choice. The Acconeer SDK for XM126 has been developed and verified with NCS v2.9.1



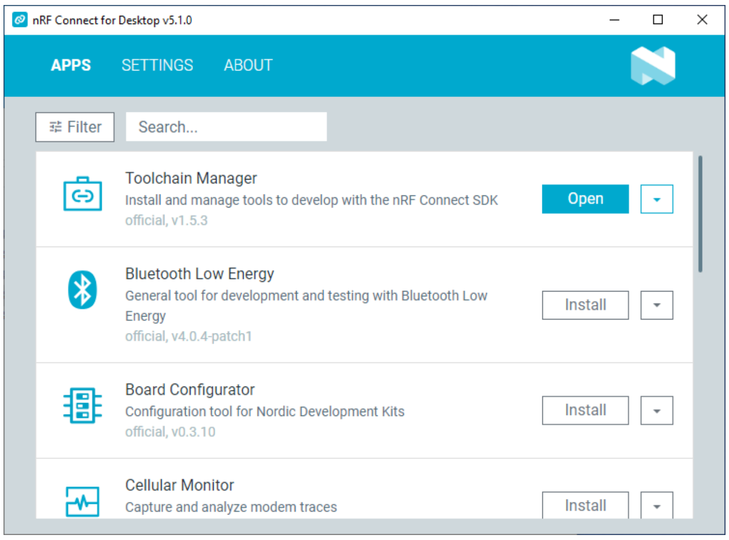
Quick Start
Get started with a new Nordic Semiconductor device
official, v1.2.0

RSSI Viewer
Live visualization of RSSI vs frequency in the 2.4 GHz band
official, v1.7.2

Serial Terminal
Terminal emulator for serial port connections
official, v1.4.3

Toolchain Manager
Install and manage tools to develop with the nRF Connect SDK
official, v1.5.3

Figure 7: nRF Connect for Desktop



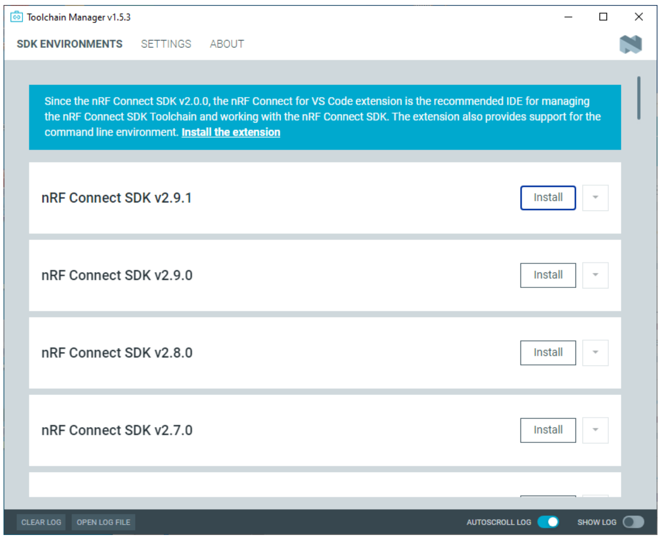
Toolchain Manager
Install and manage tools to develop with the nRF Connect SDK
official, v1.5.3

Bluetooth Low Energy
General tool for development and testing with Bluetooth Low Energy
official, v4.0.4-patch1

Board Configurator
Configuration tool for Nordic Development Kits
official, v0.3.10

Cellular Monitor
Capture and analyze modem traces

Figure 8: nRF Connect for Desktop - Open Toolchain Manager



Since the nRF Connect SDK v2.0.0, the nRF Connect for VS Code extension is the recommended IDE for managing the nRF Connect SDK Toolchain and working with the nRF Connect SDK. The extension also provides support for the command line environment. **Install the extension**

nRF Connect SDK v2.9.1

nRF Connect SDK v2.9.0

nRF Connect SDK v2.8.0

nRF Connect SDK v2.7.0

Figure 9: Toolchain Manager

5.4 Build using nRF Connect for VS Code

This section will cover how to open an Acconeer example application in Visual Studio Code and build, and run it on the XM126 module. We will use the nRF Connect plugin and assume that you have the XM126 module connected to a XB122 breakout board and a J-link debugger.

The nRF Connect for VS Code lets you develop, build and debug applications based on the nRF Connect SDK using the Visual Studio Code Integrated Development Environment (VS Code IDE). The nRF Connect for VS Code can be downloaded using the Toolchain Manager found in nRF Connect for Desktop. Alternatively, it can be downloaded directly from inside Visual Studio Code.

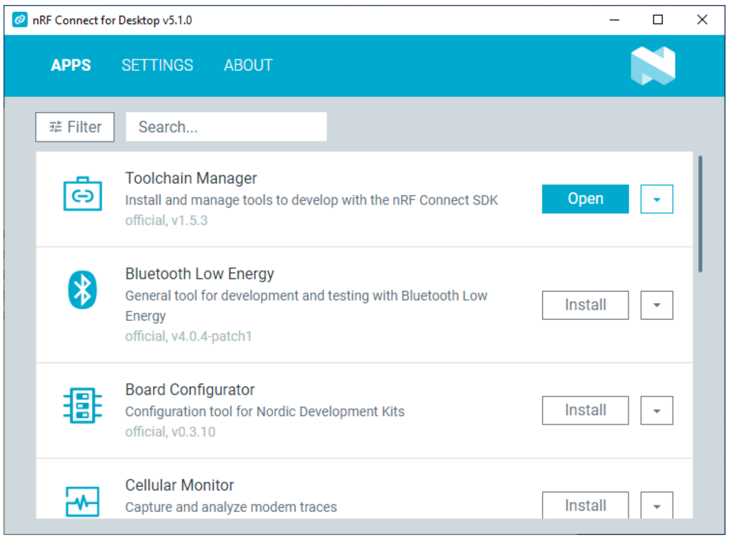
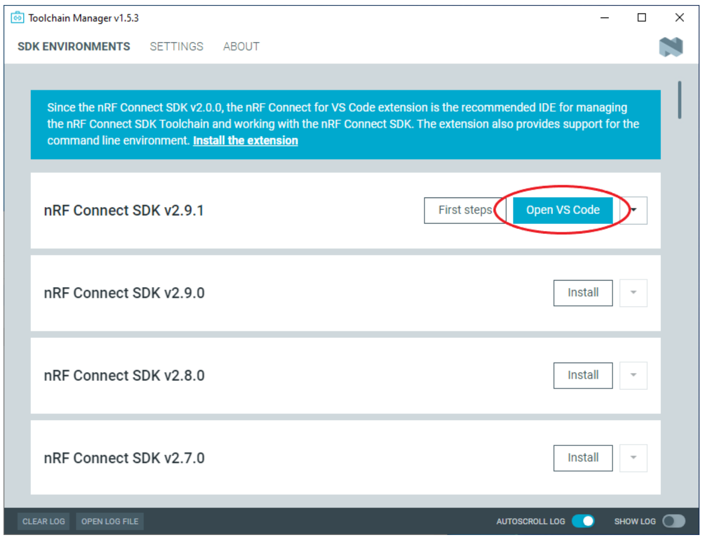
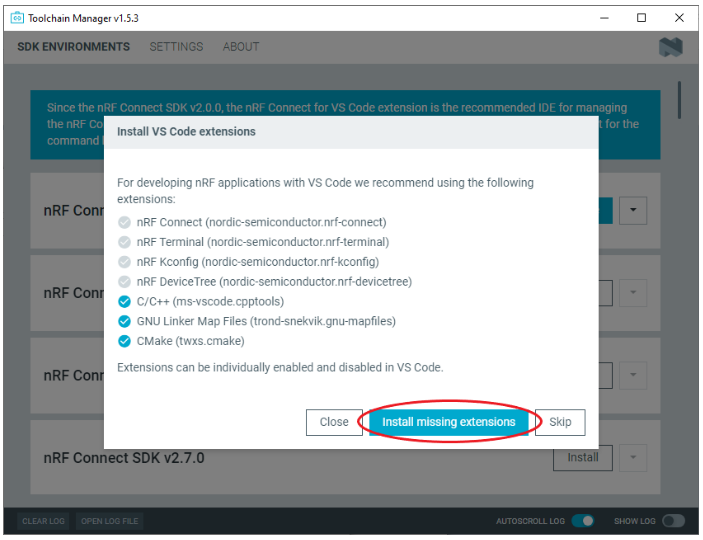

nRF Connect for Desktop v5.1.0
APPS
SETTINGS
ABOUT
Filter
Search...
Toolchain Manager
Install and manage tools to develop with the nRF Connect SDK
official, v1.5.3
Open
Bluetooth Low Energy
General tool for development and testing with Bluetooth Low Energy
official, v4.0.4-patch1
Install
Board Configurator
Configuration tool for Nordic Development Kits
official, v0.3.10
Install
Cellular Monitor
Capture and analyze modem traces
Install

Figure 10: nRF Connect for Desktop - Open Toolchain Manager



Since the nRF Connect SDK v2.0.0, the nRF Connect for VS Code extension is the recommended IDE for managing the nRF Connect SDK Toolchain and working with the nRF Connect SDK. The extension also provides support for the command line environment. **Install the extension**

Figure 11: Open nRF Connect for VS Code


Toolchain Manager v1.5.3

SDK ENVIRONMENTS SETTINGS ABOUT

Since the nRF Connect SDK v2.0.0, the nRF Connect for VS Code extension is the recommended IDE for managing the nRF Co … for the command …

Install VS Code extensions

For developing nRF applications with VS Code we recommend using the following extensions:

- nRF Connect (nordic-semiconductor.nrf-connect)
- nRF Terminal (nordic-semiconductor.nrf-terminal)
- nRF Kconfig (nordic-semiconductor.nrf-kconfig)
- nRF DeviceTree (nordic-semiconductor.nrf-devicetree)
- ✓ C/C++ (ms-vscode.cpptools)
- ✓ GNU Linker Map Files (trond-snekvik.gnu-mapfiles)
- ✓ CMake (twxs.cmake)

Extensions can be individually enabled and disabled in VS Code.

Close | Install missing extensions | Skip

nRF Connect SDK v2.7.0 Install

CLEAR LOG OPEN LOG FILE AUTOSCROLL LOG SHOW LOG

Figure 12: nRF Connect Extension for VS Code

If you use VS Code for other projects they may open up automatically when VS code is started. If that happens go to the file menu and close any open workspace or remote connection:

- “File → Close Workspace”
- “File → Close Remote Connection”

If these menu items don’t appear in the menu, it’s fine as you have nothing to close.

Open the nRF Connect Extension by clicking on the nRF Connect Extension icon on the left.

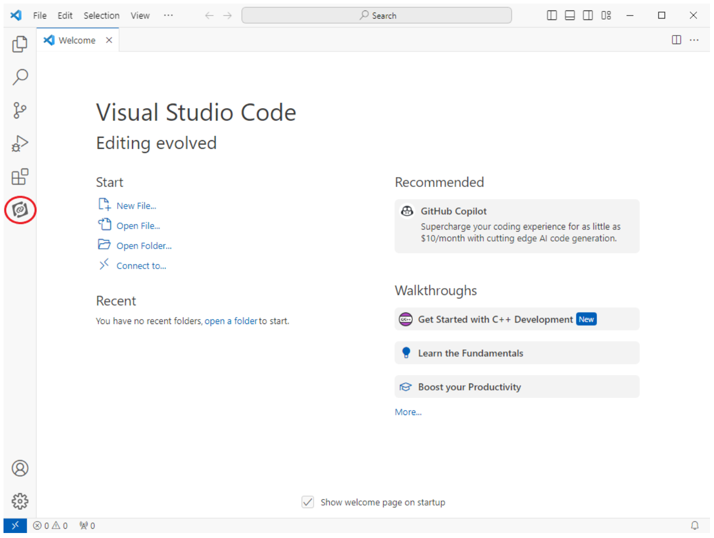
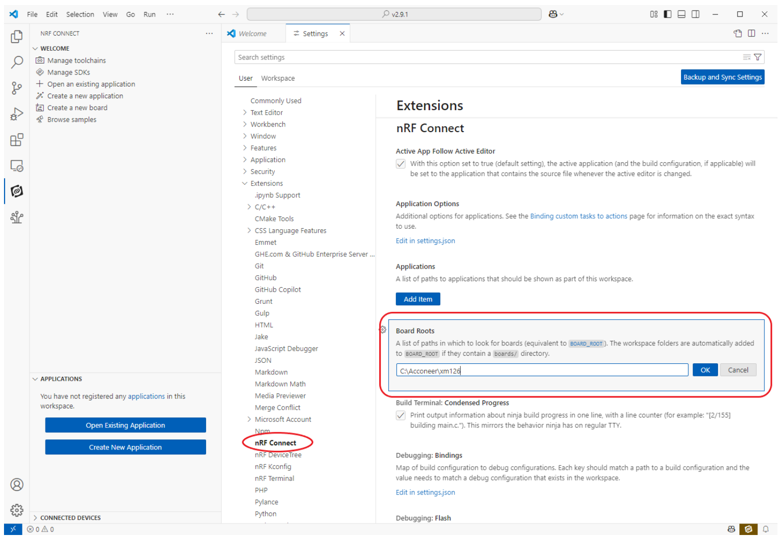

File Edit Selection View ···
Search
Welcome
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Connect to...
Recent
You have no recent folders, open a folder to start.
Recommended
GitHub Copilot
Supercharge your coding experience for as little as $10/month with cutting edge AI code generation.
Walkthroughs
Get Started with C++ Development New
Learn the Fundamentals
Boost your Productivity
More...
Show welcome page on startup

Figure 13: VS Code

5.4.1 Configure the nRF Connect extension

To be able to select XM126 as a custom board in the nRF Connect Extension the XM126 SDK package must be added as a board root.

- Open “File → Preferences → Settings”
- Go to “Extensions → nRF Connect → Board Roots”
- Press “Add Item”
- Add the path to the unzipped SDK package for example “c:\Acconeer\xm126”



Extensions

nRF Connect

Active App Follow Active Editor

✓ With this option set to true (default setting), the active application (and the build configuration, if applicable) will be set to the application that contains the source file whenever the active editor is changed.

Application Options

Additional options for applications. See the Binding custom tasks to actions page for information on the exact syntax to use.

Edit in settings.json

Applications

A list of paths to applications that should be shown as part of this workspace.

Add Item

Board Roots

A list of paths in which to look for boards (equivalent to `BOARD_ROOT`). The workspace folders are automatically added to `BOARD_ROOT` if they contain a `boards/` directory.

C:\Acconeer\xm126 OK Cancel

Build Terminal: Condensed Progress

✓ Print output information about ninja build progress in one line, with a line counter (for example: "[2/155] building main.c."). This mirrors the behavior ninja has on regular TTY.

Debugging: Bindings

Map of build configuration to debug configurations. Each key should match a path to a build configuration and the value needs to match a debug configuration that exists in the workspace.

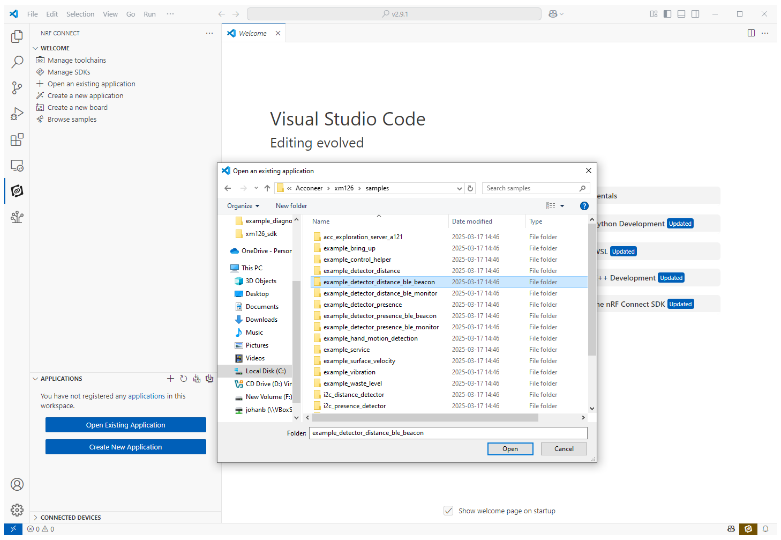
Edit in settings.json

Debugging: Flash

Figure 14: VS Code - Add Board Root

5.4.2 Open the Acconeer Example

Click the "Open Existing Application" button and open one of the sample folders in the XM126 SDK package.



Name	Date modified	Type
acc_exploration_server_a121	2025-03-17 14:46	File folder
example_bring_up	2025-03-17 14:46	File folder
example_control_helper	2025-03-17 14:46	File folder
example_detector_distance	2025-03-17 14:46	File folder
example_detector_distance_ble_beacon	2025-03-17 14:46	File folder
example_detector_distance_ble_monitor	2025-03-17 14:46	File folder
example_detector_presence	2025-03-17 14:46	File folder
example_detector_presence_ble_beacon	2025-03-17 14:46	File folder
example_detector_presence_ble_monitor	2025-03-17 14:46	File folder
example_hand_motion_detection	2025-03-17 14:46	File folder
example_service	2025-03-17 14:46	File folder
example_surface_velocity	2025-03-17 14:46	File folder
example_vibration	2025-03-17 14:46	File folder
example_waste_level	2025-03-17 14:46	File folder
i2c_distance_detector	2025-03-17 14:46	File folder
i2c_presence_detector	2025-03-17 14:46	File folder

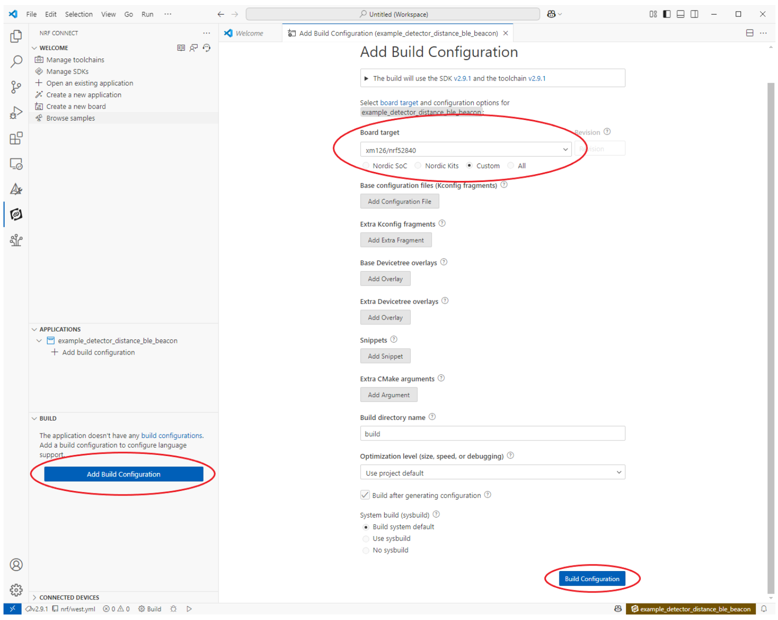
Folder: example_detector_distance_ble_beacon

Open Cancel

Figure 15: VS Code - Open Existing Application

After opening the Application we need to add a build configuration.

1. Click on the "Add Build Configuration" button.
2. Select "Custom Boards" and xm126/nrf52840 as the board type.



Add Build Configuration

▶ The build will use the SDK v2.9.1 and the toolchain v2.9.1

Select board target and configuration options for example_detector_distance_ble_beacon:

Board target: xm126/nrf52840

Nordic SoC Nordic Kits [✓ Custom] All

Build directory name: build

Optimization level (size, speed, or debugging): Use project default

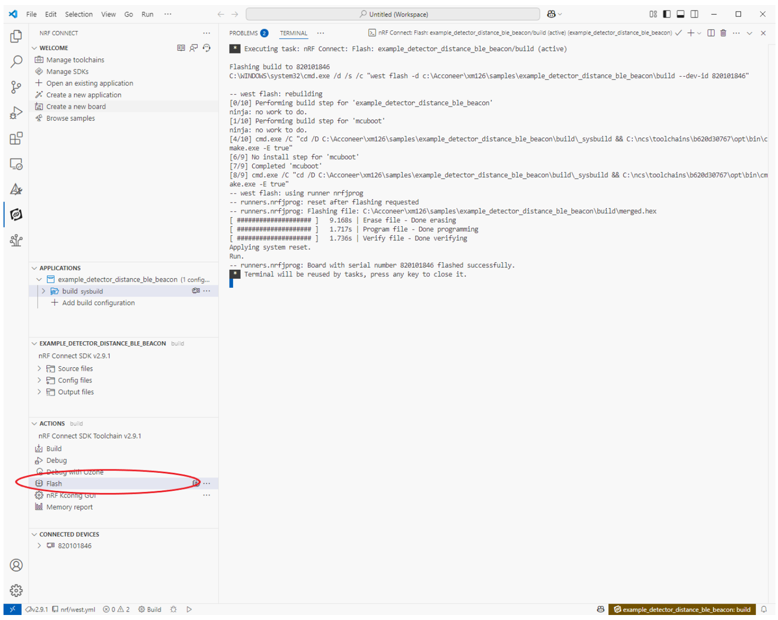
[✓] Build after generating configuration

System build (sysbuild): [✓ Build system default] Use sysbuild No sysbuild

Figure 16: Add a new build configuration

Scroll down and press the "Build Configuration" button to start the build.

If the build was successful and you have connected XB122/XM126 to a J-Link debugger, you can now flash the example program to the module by clicking "Flash".



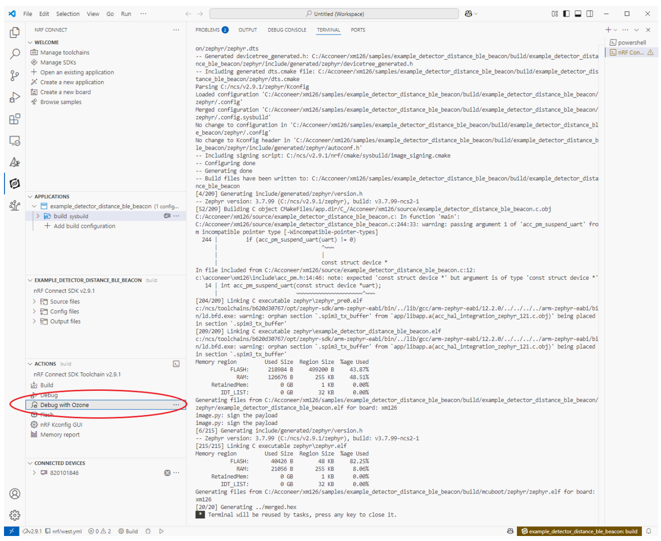
```
 * Executing task: nRF Connect: Flash: example_detector_distance_ble_beacon/build (active)

Flashing build to 820101846
C:\WINDOWS\system32\cmd.exe /d /s /c "west flash -d c:\Acconeer\xm126\samples\example_detector_distance_ble_beacon\build --dev-id 820101846"

-- west flash: rebuilding
[0/10] Performing build step for 'example_detector_distance_ble_beacon'
ninja: no work to do.
[1/10] Performing build step for 'mcuboot'
ninja: no work to do.
[4/10] cmd.exe /C "cd /D C:\Acconeer\xm126\samples\example_detector_distance_ble_beacon\build\_sysbuild && C:\ncs\toolchains\b620d30767\opt\bin\c
make.exe -E true"
[6/9] No install step for 'mcuboot'
[7/9] Completed 'mcuboot'
[8/9] cmd.exe /C "cd /D C:\Acconeer\xm126\samples\example_detector_distance_ble_beacon\build\_sysbuild && C:\ncs\toolchains\b620d30767\opt\bin\cm
ake.exe -E true"
-- west flash: using runner nrfjprog
-- runners.nrfjprog: reset after flashing requested
-- runners.nrfjprog: Flashing file: C:\Acconeer\xm126\samples\example_detector_distance_ble_beacon\build\merged.hex
[ #################### ]   9.168s | Erase file - Done erasing
[ #################### ]   1.717s | Program file - Done programming
[ #################### ]   1.736s | Verify file - Done verifying
Applying system reset.
Run.
-- runners.nrfjprog: Board with serial number 820101846 flashed successfully.
 * Terminal will be reused by tasks, press any key to close it.
```

Figure 17: Flash software to the board

5.4.3 Debug the Acconeer Example

If you have a JLink debugger the preferred way to debug your application is using Ozone from SEGGER which can be downloaded and installed from www.segger.com

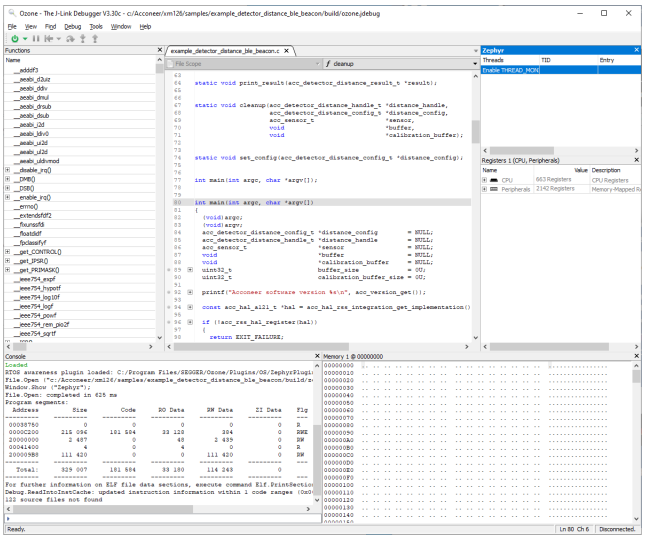
Once Ozone is installed there will be a new Debug option under ACTIONS, "Debug with Ozone".



```
on/zephyr/zephyr.dts
-- Generated devicetree_generated.h: C:/Acconeer/xm126/samples/example_detector_distance_ble_beacon/build/example_detector_dista
nce_ble_beacon/zephyr/include/generated/zephyr/devicetree_generated.h
-- Including generated dts.cmake file: C:/Acconeer/xm126/samples/example_detector_distance_ble_beacon/build/example_detector_dis
tance_ble_beacon/zephyr/dts.cmake
Parsing C:/ncs/v2.9.1/zephyr/Kconfig
Loaded configuration 'C:/Acconeer/xm126/samples/example_detector_distance_ble_beacon/build/example_detector_distance_ble_beacon/
zephyr/.config'
Merged configuration 'C:/Acconeer/xm126/samples/example_detector_distance_ble_beacon/build/example_detector_distance_ble_beacon/
zephyr/.config.sysbuild'
No change to configuration in 'C:/Acconeer/xm126/samples/example_detector_distance_ble_beacon/build/example_detector_distance_bl
e_beacon/zephyr/.config'
No change to Kconfig header in 'C:/Acconeer/xm126/samples/example_detector_distance_ble_beacon/build/example_detector_distance_b
le_beacon/zephyr/include/generated/zephyr/autoconf.h'
-- Including signing script: C:/ncs/v2.9.1/nrf/cmake/sysbuild/image_signing.cmake
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Acconeer/xm126/samples/example_detector_distance_ble_beacon/build/example_detector_dista
nce_ble_beacon
[4/209] Generating include/generated/zephyr/version.h
-- Zephyr version: 3.7.99 (C:/ncs/v2.9.1/zephyr), build: v3.7.99-ncs2-1
[52/209] Building C object CMakeFiles/app.dir/C_/Acconeer/xm126/source/example_detector_distance_ble_beacon.c.obj
C:/Acconeer/xm126/source/example_detector_distance_ble_beacon.c: In function 'main':
C:/Acconeer/xm126/source/example_detector_distance_ble_beacon.c:244:33: warning: passing argument 1 of 'acc_pm_suspend_uart' fro
m incompatible pointer type [-Wincompatible-pointer-types]
  244 |         if (acc_pm_suspend_uart(uart) != 0)
      |                                 ^~~~
      |                                 |
      |                                 const struct device *
In file included from C:/Acconeer/xm126/source/example_detector_distance_ble_beacon.c:12:
c:\acconeer\xm126\include\acc_pm.h:14:46: note: expected 'const struct device *' but argument is of type 'const struct device *'
   14 | int acc_pm_suspend_uart(const struct device *uart);
      |                         ~~~~~~~~~~~~~~~~~~~~~^~~~
[204/209] Linking C executable zephyr\zephyr_pre0.elf
c:/ncs/toolchains/b620d30767/opt/zephyr-sdk/arm-zephyr-eabi/bin/../lib/gcc/arm-zephyr-eabi/12.2.0/../../../../arm-zephyr-eabi/bi
n/ld.bfd.exe: warning: orphan section `.spim3_tx_buffer' from `app/libapp.a(acc_hal_integration_zephyr_121.c.obj)' being placed
in section `.spim3_tx_buffer'
[209/209] Linking C executable zephyr\example_detector_distance_ble_beacon.elf
c:/ncs/toolchains/b620d30767/opt/zephyr-sdk/arm-zephyr-eabi/bin/../lib/gcc/arm-zephyr-eabi/12.2.0/../../../../arm-zephyr-eabi/bi
n/ld.bfd.exe: warning: orphan section `.spim3_tx_buffer' from `app/libapp.a(acc_hal_integration_zephyr_121.c.obj)' being placed
in section `.spim3_tx_buffer'
Memory region         Used Size  Region Size  %age Used
           FLASH:      218984 B     499200 B     43.87%
             RAM:      126676 B       255 KB     48.51%
      RetainedMem:          0 GB         1 KB      0.00%
         IDT_LIST:          0 GB        32 KB      0.00%
Generating files from C:/Acconeer/xm126/samples/example_detector_distance_ble_beacon/build/example_detector_distance_ble_beacon/
zephyr/example_detector_distance_ble_beacon.elf for board: xm126
image.py: sign the payload
image.py: sign the payload
[6/215] Generating include/generated/zephyr/version.h
-- Zephyr version: 3.7.99 (C:/ncs/v2.9.1/zephyr), build: v3.7.99-ncs2-1
[215/215] Linking C executable zephyr\zephyr.elf
Memory region         Used Size  Region Size  %age Used
           FLASH:       40426 B        48 KB     82.25%
             RAM:       21056 B       255 KB      8.06%
      RetainedMem:          0 GB         1 KB      0.00%
         IDT_LIST:          0 GB        32 KB      0.00%
Generating files from C:/Acconeer/xm126/samples/example_detector_distance_ble_beacon/build/mcuboot/zephyr/zephyr.elf for board:
xm126
[20/20] Generating ../merged.hex
 * Terminal will be reused by tasks, press any key to close it.
```

Figure 18: Debug with Ozone

Press "Debug with Ozone" and start debugging using Ozone.



```
static void print_result(acc_detector_distance_result_t *result);


static void cleanup(acc_detector_distance_handle_t *distance_handle,
                    acc_detector_distance_config_t *distance_config,
                    acc_sensor_t                   *sensor,
                    void                           *buffer,
                    void                           *calibration_buffer);


static void set_config(acc_detector_distance_config_t *distance_config);


int main(int argc, char *argv[]);


int main(int argc, char *argv[])
{
  (void)argc;
  (void)argv;
  acc_detector_distance_config_t *distance_config        = NULL;
  acc_detector_distance_handle_t *distance_handle        = NULL;
  acc_sensor_t                   *sensor                 = NULL;
  void                           *buffer                 = NULL;
  void                           *calibration_buffer     = NULL;
  uint32_t                       buffer_size             = 0U;
  uint32_t                       calibration_buffer_size = 0U;

  printf("Acconeer software version %s\n", acc_version_get());

  const acc_hal_a121_t *hal = acc_hal_rss_integration_get_implementation()

  if (!acc_rss_hal_register(hal))
  {
    return EXIT_FAILURE;
```

```
Loaded
RTOS awareness plugin loaded: C:/Program Files/SEGGER/Ozone/Plugins/OS/ZephyrPlugi
File.Open ("c:/Acconeer/xm126/samples/example_detector_distance_ble_beacon/build/z
Window.Show ("Zephyr");
File.Open: completed in 625 ms
Program segments:
  Address        Size        Code     RO Data     RW Data     ZI Data    Flg
---------   ---------   ---------   ---------   ---------   ---------    ---
 00038750           0           0           0           0           0    R
 0000C200     215 096     181 584      33 128         384           0    RWE
 20000000       2 487           0          48       2 439           0    RW
 00041400           4           0           4           0           0    R
 200009B8     111 420           0           0     111 420           0    RW
---------   ---------   ---------   ---------   ---------   ---------    ---
   Total:     329 007     181 584      33 180     114 243           0
---------   ---------   ---------   ---------   ---------   ---------    ---
For further information on ELF file data sections, execute command Elf.PrintSectio
Debug.ReadIntoInstCache: updated instruction information within 1 code ranges (0x0
122 source files not found
```

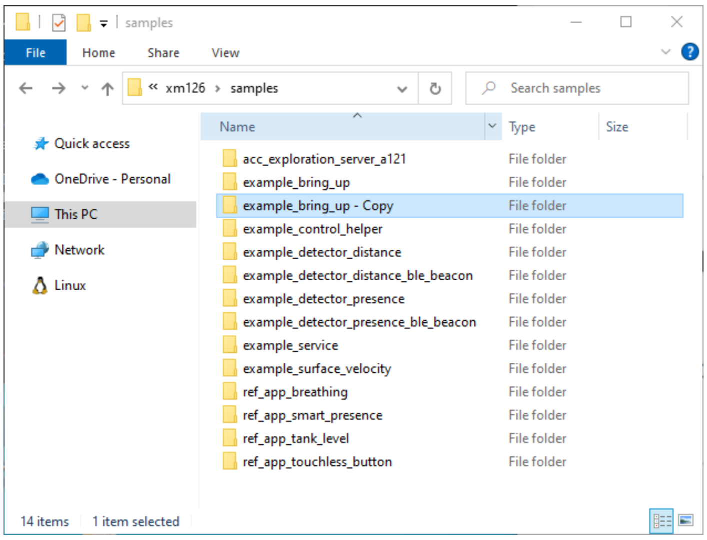
Figure 19: Ozone Debugger

5.4.4 Create New Acconeer Example

All Acconeer example applications are delivered as source code but not all of them are defined as sample applications from the start. To create a new sample application from an existing example one can copy one of the existing samples into a new directory inside “samples”.

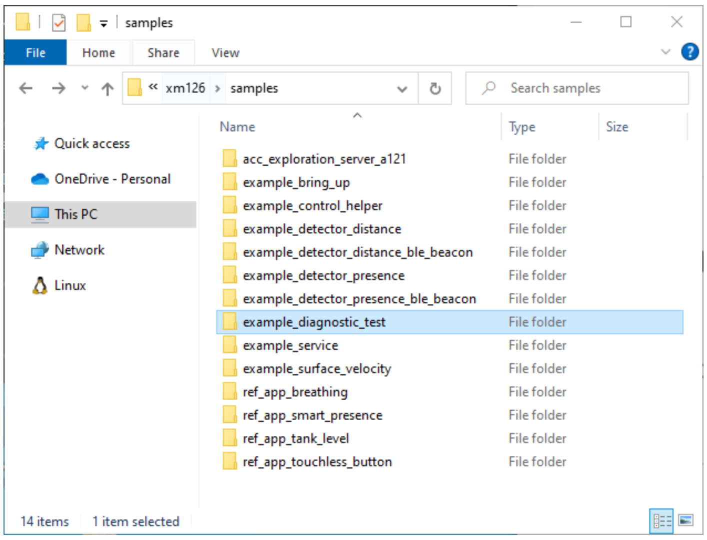
In this case a new example “example_diagnostic_test” is created from “example_bring_up”.

1. Copy “example_bring_up” directory and rename it as “example_diagnostic_test”
2. Open “prj.conf” and change the configuration “CONFIG_KERNEL_BIN_NAME” to “example_diagnostic_test”
3. Open “CMakeLists.txt” and change “example_bring_up.c” in “app_sources” to “example_diagnostic_test.c”
4. Remove “out” directory if it exists
5. You can now add the new sample application in VS code as described in “Open the Acconeer Example”



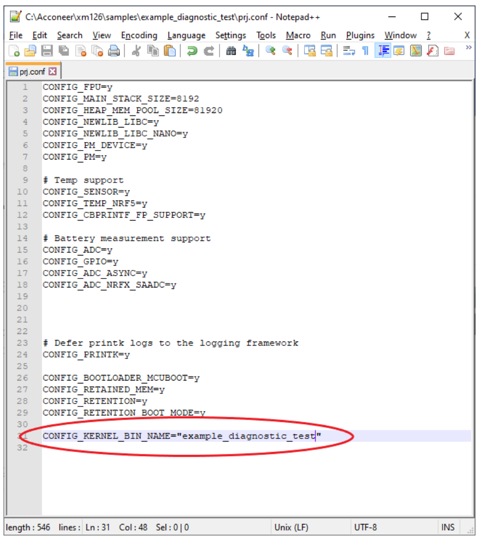
Name	Type	Size
acc_exploration_server_a121	File folder	
example_bring_up	File folder	
example_bring_up - Copy	File folder	
example_control_helper	File folder	
example_detector_distance	File folder	
example_detector_distance_ble_beacon	File folder	
example_detector_presence	File folder	
example_detector_presence_ble_beacon	File folder	
example_service	File folder	
example_surface_velocity	File folder	
ref_app_breathing	File folder	
ref_app_smart_presence	File folder	
ref_app_tank_level	File folder	
ref_app_touchless_button	File folder	

Figure 20: Copy sample application folder



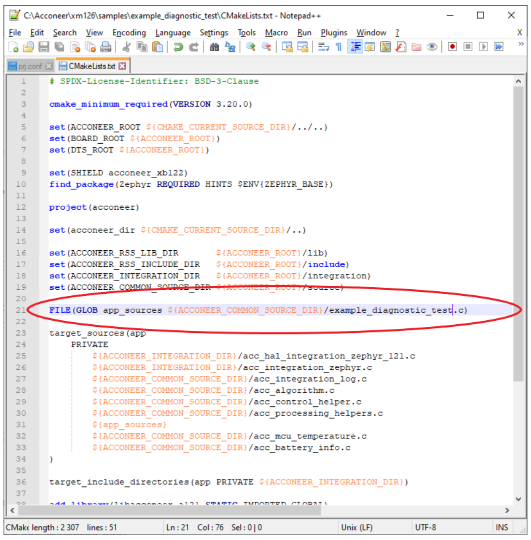
Name	Type	Size
acc_exploration_server_a121	File folder	
example_bring_up	File folder	
example_control_helper	File folder	
example_detector_distance	File folder	
example_detector_distance_ble_beacon	File folder	
example_detector_presence	File folder	
example_detector_presence_ble_beacon	File folder	
example_diagnostic_test	File folder	
example_service	File folder	
example_surface_velocity	File folder	
ref_app_breathing	File folder	
ref_app_smart_presence	File folder	
ref_app_tank_level	File folder	
ref_app_touchless_button	File folder	

Figure 21: Rename sample application folder


C:\Acconeer\xm126\samples\example_diagnostic_test\prj.conf - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
prj.conf
length : 546 lines : Ln : 31 Col : 48 Sel : 0 | 0
Unix (LF)
UTF-8
INS

```
CONFIG_FPU=y
CONFIG_MAIN_STACK_SIZE=8192
CONFIG_HEAP_MEM_POOL_SIZE=81920
CONFIG_NEWLIB_LIBC=y
CONFIG_NEWLIB_LIBC_NANO=y
CONFIG_PM_DEVICE=y
CONFIG_PM=y

# Temp support
CONFIG_SENSOR=y
CONFIG_TEMP_NRF5=y
CONFIG_CBPRINTF_FP_SUPPORT=y

# Battery measurement support
CONFIG_ADC=y
CONFIG_GPIO=y
CONFIG_ADC_ASYNC=y
CONFIG_ADC_NRFX_SAADC=y




# Defer printk logs to the logging framework
CONFIG_PRINTK=y

CONFIG_BOOTLOADER_MCUBOOT=y
CONFIG_RETAINED_MEM=y
CONFIG_RETENTION=y
CONFIG_RETENTION_BOOT_MODE=y

CONFIG_KERNEL_BIN_NAME="example_diagnostic_test"

```

Figure 22: Change kernel bin name of application



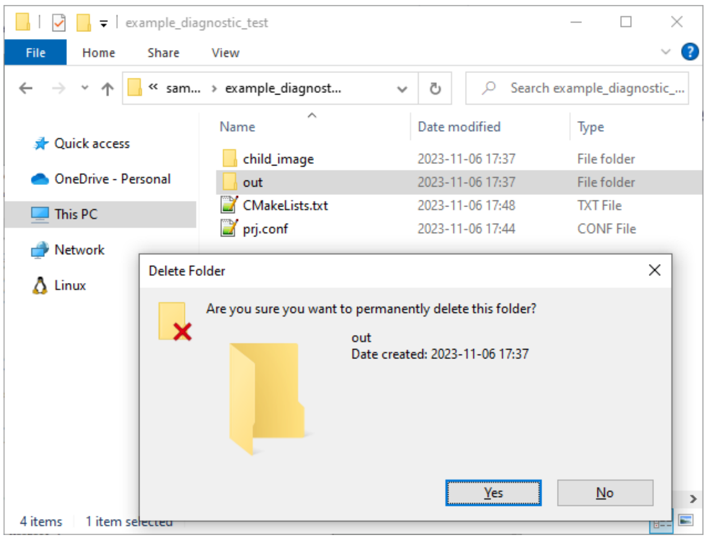
C:\Acconeer\xm126\samples\example_diagnostic_test\CMakeLists.txt - Notepad++

```
# SPDX-License-Identifier: BSD-3-Clause

cmake_minimum_required(VERSION 3.20.0)

set(ACCONEER_ROOT ${CMAKE_CURRENT_SOURCE_DIR}/../..)
set(BOARD_ROOT ${ACCONEER_ROOT})
set(DTS_ROOT ${ACCONEER_ROOT})

set(SHIELD acconeer_xb122)
find_package(Zephyr REQUIRED HINTS $ENV{ZEPHYR_BASE})

project(acconeer)

set(acconeer_dir ${CMAKE_CURRENT_SOURCE_DIR}/..)

set(ACCONEER_RSS_LIB_DIR        ${ACCONEER_ROOT}/lib)
set(ACCONEER_RSS_INCLUDE_DIR    ${ACCONEER_ROOT}/include)
set(ACCONEER_INTEGRATION_DIR    ${ACCONEER_ROOT}/integration)
set(ACCONEER_COMMON_SOURCE_DIR ${ACCONEER_ROOT}/source)

FILE(GLOB app_sources ${ACCONEER_COMMON_SOURCE_DIR}/example_diagnostic_test.c)

target_sources(app
    PRIVATE
        ${ACCONEER_INTEGRATION_DIR}/acc_hal_integration_zephyr_121.c
        ${ACCONEER_INTEGRATION_DIR}/acc_integration_zephyr.c
        ${ACCONEER_COMMON_SOURCE_DIR}/acc_integration_log.c
        ${ACCONEER_COMMON_SOURCE_DIR}/acc_algorithm.c
        ${ACCONEER_COMMON_SOURCE_DIR}/acc_control_helper.c
        ${ACCONEER_COMMON_SOURCE_DIR}/acc_processing_helpers.c
        ${app_sources}
        ${ACCONEER_COMMON_SOURCE_DIR}/acc_mcu_temperature.c
        ${ACCONEER_COMMON_SOURCE_DIR}/acc_battery_info.c
)

target_include_directories(app PRIVATE ${ACCONEER_INTEGRATION_DIR})
```

CMake length : 2 307 lines : 51 Ln : 21 Col : 76 Sel : 0 | 0 Unix (LF) UTF-8 INS

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
prj.conf
CMakeLists.txt

Figure 23: Change source file in app_sources



example_diagnostic_test

Name	Date modified	Type
child_image	2023-11-06 17:37	File folder
out	2023-11-06 17:37	File folder
CMakeLists.txt	2023-11-06 17:48	TXT File
prj.conf	2023-11-06 17:44	CONF File

Delete Folder

Are you sure you want to permanently delete this folder?

out
Date created: 2023-11-06 17:37

Figure 24: Remove out directory

5.5 Build using nRF Connect Command Line Tools

All sample applications can be built from the command line using “west”. Open a bash shell from Toolchain Manager.

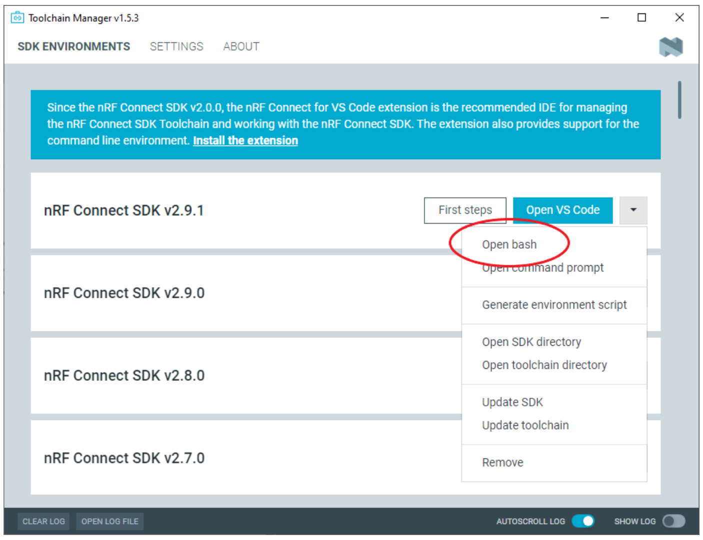
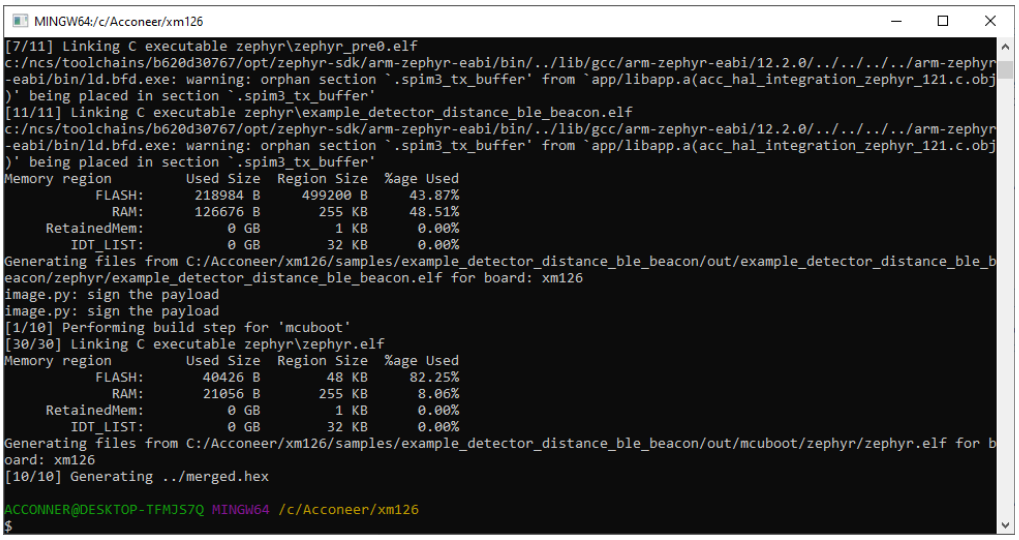

Since the nRF Connect SDK v2.0.0, the nRF Connect for VS Code extension is the recommended IDE for managing the nRF Connect SDK Toolchain and working with the nRF Connect SDK. The extension also provides support for the command line environment. Install the extension

Figure 25: Open nRF Connect for VS Code

Change working directory to the path to the unzipped SDK package for example “c:\Acconeer\xm126”.

```
$ cd /c/Acconeer/xm126
$ west build --build-dir samples/example_detector_distance_ble_beacon/out
   samples/example_detector_distance_ble_beacon --pristine --board xm126/
   nrf52840 -- -DNCS_TOOLCHAIN_VERSION=NONE -DBOARD_ROOT=/c/Acconeer/xm126
```



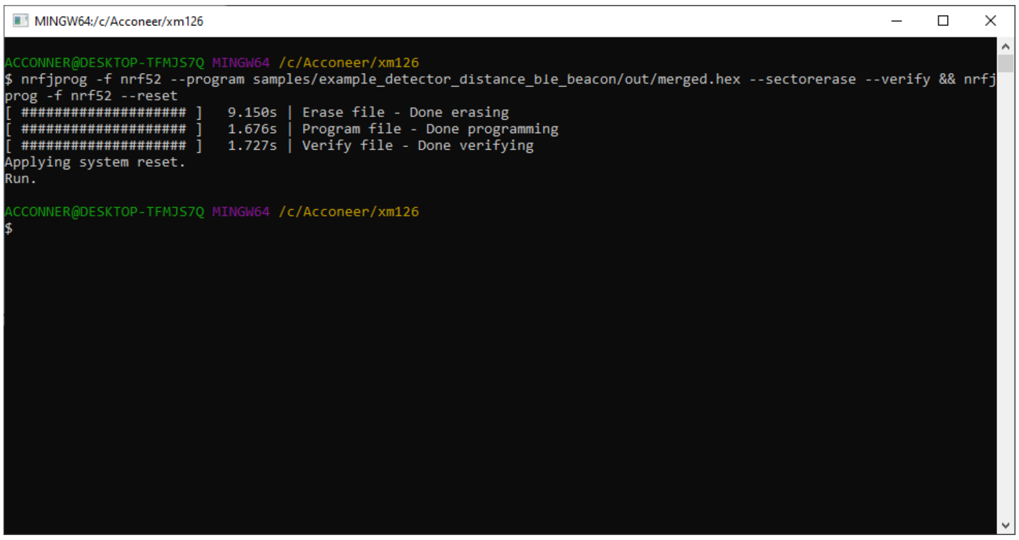
```
[7/11] Linking C executable zephyr\zephyr_pre0.elf
c:/ncs/toolchains/b620d30767/opt/zephyr-sdk/arm-zephyr-eabi/bin/../lib/gcc/arm-zephyr-eabi/12.2.0/../../../../arm-zephyr
-eabi/bin/ld.bfd.exe: warning: orphan section `.spim3_tx_buffer' from `app/libapp.a(acc_hal_integration_zephyr_121.c.obj
)' being placed in section `.spim3_tx_buffer'
[11/11] Linking C executable zephyr\example_detector_distance_ble_beacon.elf
c:/ncs/toolchains/b620d30767/opt/zephyr-sdk/arm-zephyr-eabi/bin/../lib/gcc/arm-zephyr-eabi/12.2.0/../../../../arm-zephyr
-eabi/bin/ld.bfd.exe: warning: orphan section `.spim3_tx_buffer' from `app/libapp.a(acc_hal_integration_zephyr_121.c.obj
)' being placed in section `.spim3_tx_buffer'
Memory region         Used Size  Region Size  %age Used
           FLASH:      218984 B     499200 B     43.87%
             RAM:      126676 B       255 KB     48.51%
     RetainedMem:          0 GB         1 KB      0.00%
        IDT_LIST:          0 GB        32 KB      0.00%
Generating files from C:/Acconeer/xm126/samples/example_detector_distance_ble_beacon/out/example_detector_distance_ble_b
eacon/zephyr/example_detector_distance_ble_beacon.elf for board: xm126
image.py: sign the payload
image.py: sign the payload
[1/10] Performing build step for 'mcuboot'
[30/30] Linking C executable zephyr\zephyr.elf
Memory region         Used Size  Region Size  %age Used
           FLASH:       40426 B        48 KB     82.25%
             RAM:       21056 B       255 KB      8.06%
     RetainedMem:          0 GB         1 KB      0.00%
        IDT_LIST:          0 GB        32 KB      0.00%
Generating files from C:/Acconeer/xm126/samples/example_detector_distance_ble_beacon/out/mcuboot/zephyr/zephyr.elf for b
oard: xm126
[10/10] Generating ../merged.hex

ACCONNER@DESKTOP-TFMJS7Q MINGW64 /c/Acconeer/xm126
$
```

Figure 26: Open nRF Connect for VS Code

5.5.1 Download Software Using a J-Link

You can flash the software using a J-Link debugger from the command line.

```
$ nrfjprog -f nrf52 --program samples/example_detector_distance_ble_beacon/
   out/merged.hex  --sectorerase --verify && nrfjprog -f nrf52 --reset
```



```
MINGW64:/c/Acconeer/xm126

ACCONEER@DESKTOP-TFMJS7Q MINGW64 /c/Acconeer/xm126
$ nrfjprog -f nrf52 --program samples/example_detector_distance_ble_beacon/out/merged.hex --sectorerase --verify && nrfj
prog -f nrf52 --reset
[ #################### ]   9.150s | Erase file - Done erasing
[ #################### ]   1.676s | Program file - Done programming
[ #################### ]   1.727s | Verify file - Done verifying
Applying system reset.
Run.

ACCONEER@DESKTOP-TFMJS7Q MINGW64 /c/Acconeer/xm126
$
```

close

Figure 27: Open nRF Connect for VS Code

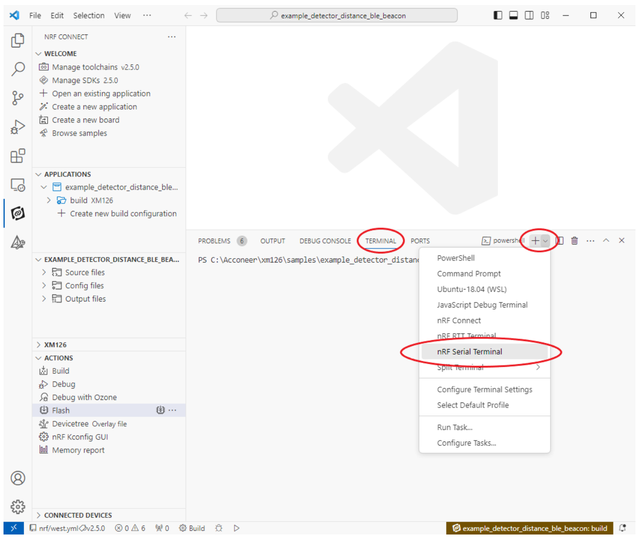
5.6 Debug Output

Debug logs will be outputted on the UART accessible through the USB connector on XB122 using a baud rate of 115200.

5.6.1 Debug Output in nRF Connect for VS Code

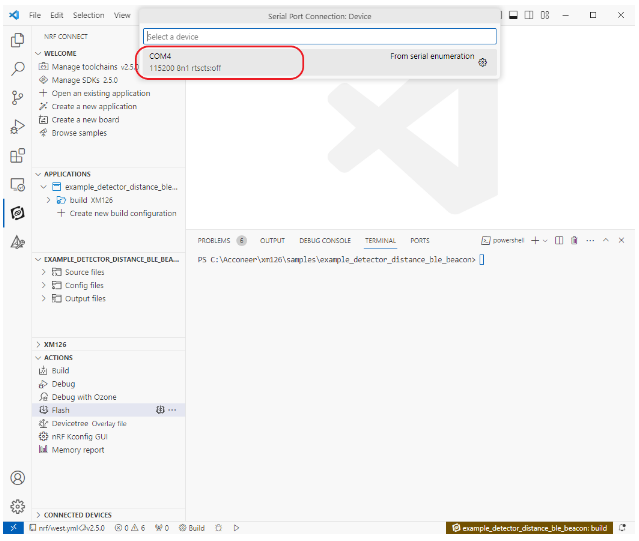
It is possible to get nRF Connect for VS Code to capture the debug logs.

- Open the “TERMINAL” View
- Press “Launch Profile...” and then “nRF Serial Terminal”
- Select the port, in this case “COM4”
- Debug logs will now be shown in the “TERMINAL” View



```
PS C:\Acconeer\xm126\samples\example_detector_distan
```

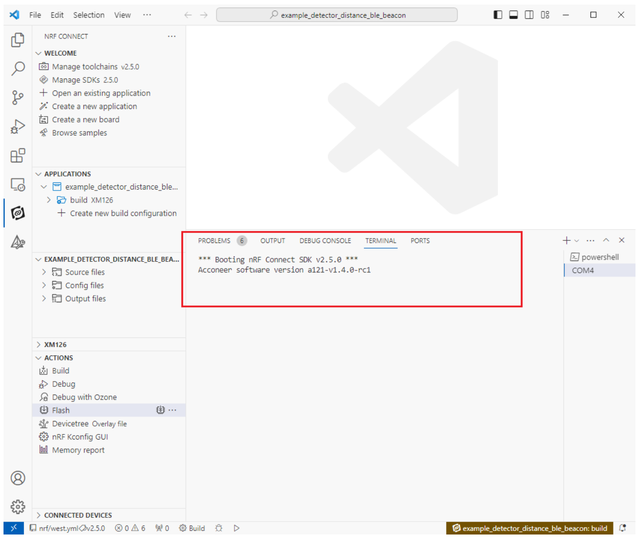
Figure 28: VS Code Debug Logs - Open nRF Serial Terminal



```
COM4
115200 8n1 rtscts:off
```

```
PS C:\Acconeer\xm126\samples\example_detector_distance_ble_beacon>
```

Figure 29: VS Code Debug Logs - Select port



```
*** Booting nRF Connect SDK v2.5.0 ***
Acconeer software version a121-v1.4.0-rc1
```

Figure 30: VS Code Debug Logs - Logs

6 Disclaimer

The information herein is believed to be correct as of the date issued. Acconeer AB ("Acconeer") will not be responsible for damages of any nature resulting from the use or reliance upon the information contained herein. Acconeer makes no warranties, expressed or implied, of merchantability or fitness for a particular purpose or course of performance or usage of trade. Therefore, it is the user's responsibility to thoroughly test the product in their particular application to determine its performance, efficacy and safety. Users should obtain the latest relevant information before placing orders.

Unless Acconeer has explicitly designated an individual Acconeer product as meeting the requirement of a particular industry standard, Acconeer is not responsible for any failure to meet such industry standard requirements.

Unless explicitly stated herein this document Acconeer has not performed any regulatory conformity test. It is the user's responsibility to assure that necessary regulatory conditions are met and approvals have been obtained when using the product. Regardless of whether the product has passed any conformity test, this document does not constitute any regulatory approval of the user's product or application using Acconeer's product.

Nothing contained herein is to be considered as permission or a recommendation to infringe any patent or any other intellectual property right. No license, express or implied, to any intellectual property right is granted by Acconeer herein.

Acconeer reserves the right to at any time correct, change, amend, enhance, modify, and improve this document and/or Acconeer products without notice.

This document supersedes and replaces all information supplied prior to the publication hereof.