



# Wireless IoT Entwicklung mit Nordic Semiconductor

Einleitung und SDK Überblick

*Nordic Semiconductor*

*Christian Kurz*

*Thomas Page*

*February 2026*

# Agenda des Workshops

- Einleitung von Nordic zum nRF Connect SDK
- Hands-on Kapitel 1: IDE Einführung & Blinky Applikation
- Hands-on Kapitel 2: Bluetooth Peripheral LED & Button Service (LBS)
- Hands-on Kapitel 3: Nordic PMIC & Fuel Gauge Integration



# We are Nordic

## Simplifying lives through all things connected



Fabless  
semiconductor  
company



Founded in  
Norway, 1983



~1,350  
employees



\$511 MUSD revenue,  
CY 2024  
OSEBX: NOD



ISO 9001  
certified

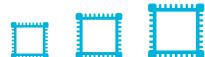


Global  
presence

# Complete solution

## Faster time-to-market

Next-gen  
hardware



ICs, SoCs, SiPs, PMICs



3rd party modules



Embedded SW stacks

Embedded  
software



nRF Connect SDK  
Unified software



Mobile Apps



Extensive SW/HW  
development tools

World-class  
support



Developer community



Online hands-on trainings



Extensive technology  
partner program

Customer  
device



Consumer



Healthcare



Industrial

Cloud lifecycle  
services



powered by Memfault



Device management



Embedded observability



Location services



# Nordic product overview

Cloud support across all our wireless connectivity solutions

Short-Range



Bluetooth® LE H-READ  
matter zigbee



Cellular IoT



LTE-M NB-IoT  
GNSS nFC



Wi-Fi 6 IoT



WiFi matter



Power Management



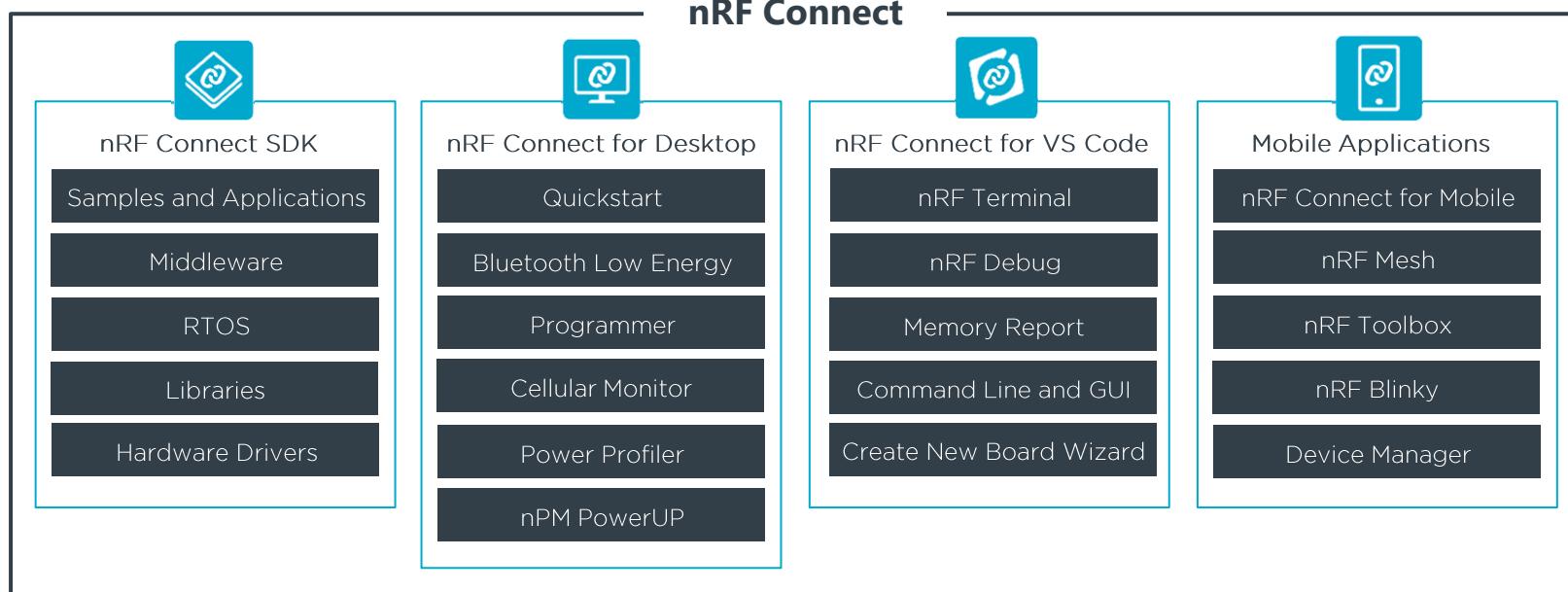
# nRF Connect SDK

## Introduction

```
191     pin->pin_value);
192 
193     // The TCI requires to check if other pin values did not change
194     // since the last framework comment, so values_to_update_and_pin_value
195     // (edit_values, pin) = pin->pin_value;
196 }
197 
198 // The gpiote lib channel
199 
200 gpiote_lib_channel_t *gpiote_channel_create(
201     void *lib,
202     gpiote_config_t config);
203 
204 gpiote_lib_uninit(gpiote_channel_t *channel);
205 
206 TEST_START
207 {
208     // this test is very heavy on tester communication, we want to decrease the communication pull
209     // period to the minimal value possible
210     tester_txs_set_communication_pull_min(1);
211 
212     before_txs_init();
213 
214     gpiote_shared_lib_test_config_t gpiote_test = {
215         polarity      = TEST_GPIO_PULSE_INVERTED,
216         mode          = TEST_GPIO_DIGITAL_VALUE,
217         gpioint       = TEST_GPIO_SHARED_LIB_MODE,
218         gpiostate     = TEST_GPIO_STATE_INIT,
219     };
220 
221     TEST_TASK_WAIT;
222 
223     set_all_tester_pins_framework_gpiote_channel();
224 
225     for (uint32_t gpiote_index = 0; gpiote_index < GPIOTE_SHARED_LIB_INSTANCE_COUNT - 1; gpiote_index++)
226     {
227         for (uint32_t channel_number = 0; channel_number < GPIOTE_SHARED_LIB_NUM_CHANNELS;
228              channel_number++)
229         {
230             gpiote_lib_channel_t gpiote_channel = { .instance_index = gpiote_index,
231                                                     .channel = channel_number };
232 
233             ok_of_channel_number_is_increased_by_one =
234                 gpiote_lib_iterator_instance_t::pin =
235                     gpiote_channel_create(lib, &gpiote_test);
236         }
237     }
238 }
239 
240 // The gpiote lib iterator
241 
242 gpiote_lib_iterator_t gpiote_lib_iterator_create(
243     void *lib,
244     gpiote_config_t config);
245 
246 gpiote_lib_iterator_t gpiote_lib_iterator_get(
247     void *lib,
248     gpiote_config_t config);
249 
```



# nRF Connect Platform Overview



# nRF Connect SDK

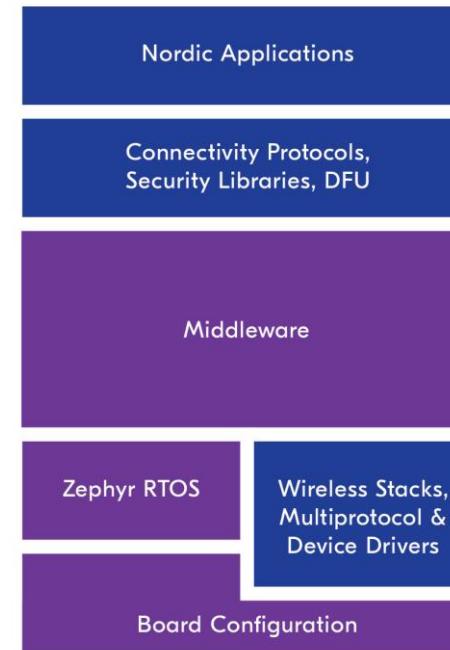
Unified code base and toolchain for all Nordic SoCs



- Modern tools and SDK
  - In the market since 2018
  - Based on Zephyr RTOS
  - Strong focus on open-source
  - Future-proof, modular approach
- 
- Single SDK for all Nordic product series
  - Bluetooth v6 qualified Host and Controller stack since v2.8.0

# nRF Connect SDK Code Base

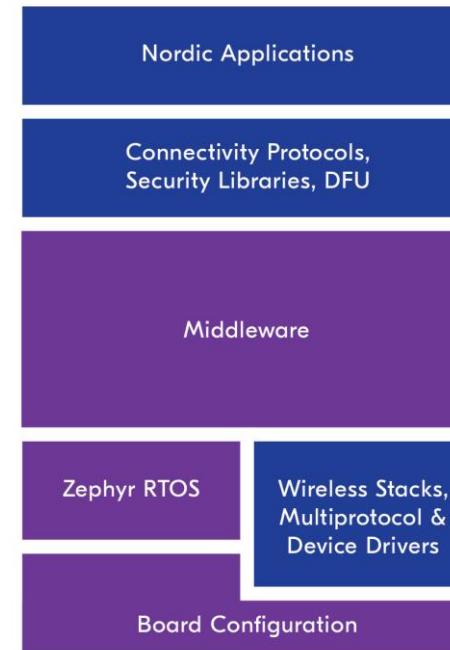
- Code base consists of several integrated repositories
  - Code coming from **Nordic** (blue)
  - Code coming from **Open Source** (OS) repositories (purple)
- Contains an RTOS, application code, connectivity protocols, wireless stacks, peripheral drivers & more



# nRF Connect SDK Repositories

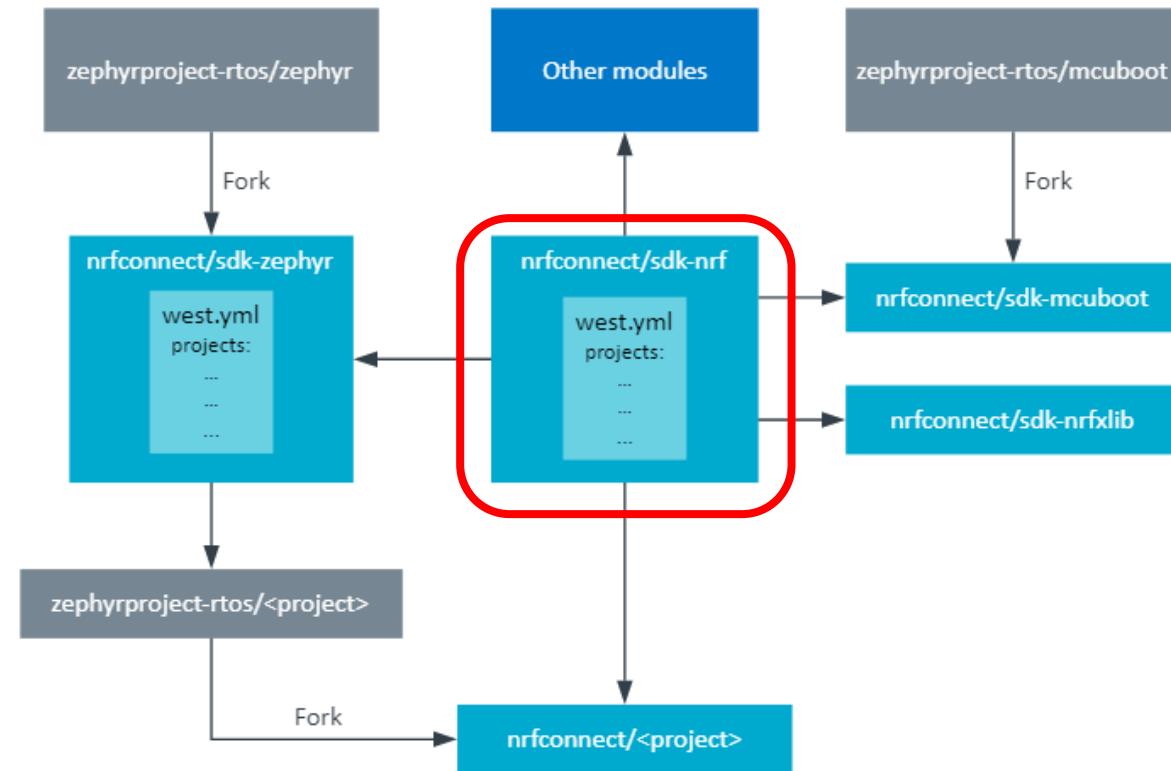
- **nRF**: Application & connectivity protocols
- **nrfxlib**: compiled libraries where Nordic cannot distribute source code
- **nrfx**: peripheral drivers
- **Zephyr**: RTOS & board configuration (OS)
- **MCUboot**: Secure Bootloader (OS)
- Other repositories
  - Trusted Firmware-M, Matter, etc.

Nordic and Open Source (OS) code

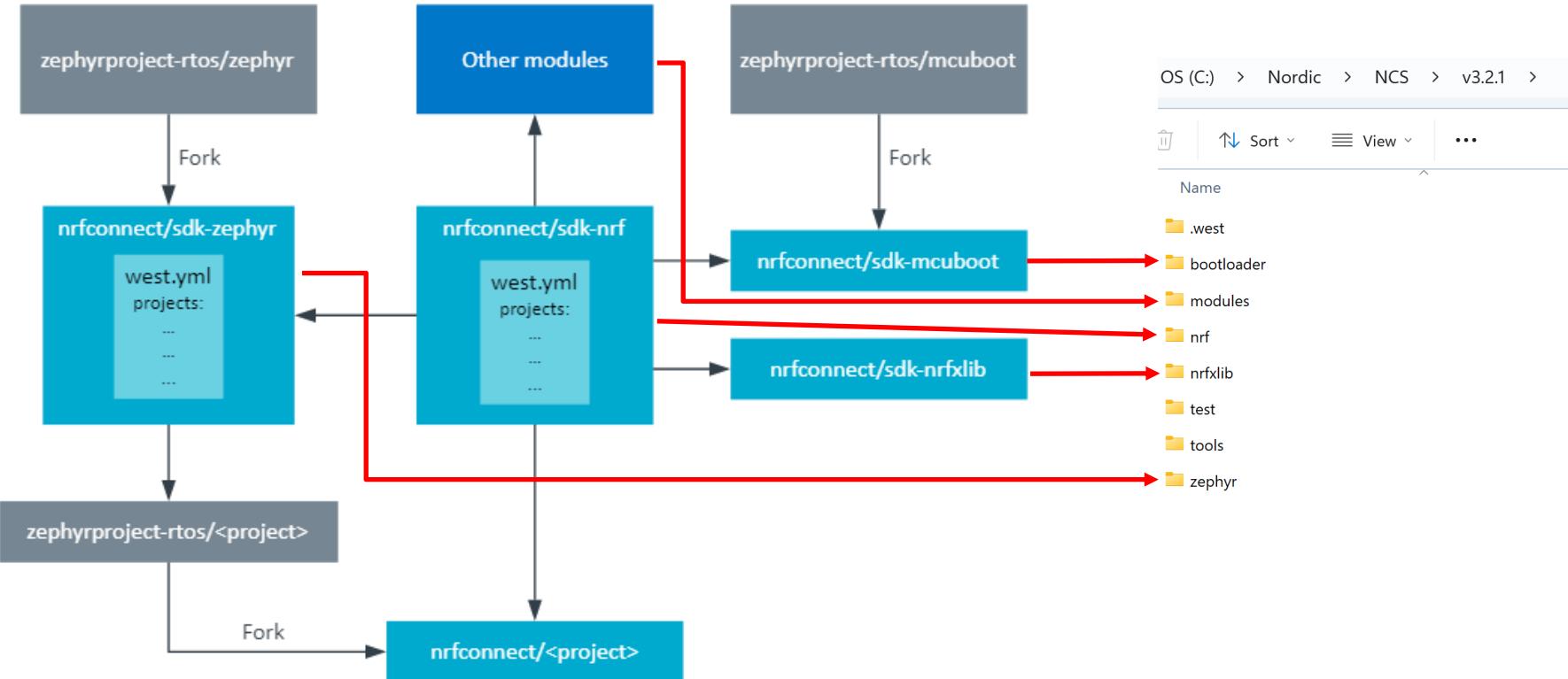


# nRF Connect SDK Structure

- The nRF Connect SDK is structured as star topology, with the sdk-nrf repository being in the center.
- The [sdk-nrf](#) repository contains the manifest file `west.yml` in its root folder, which lists and points to all other repositories included in the SDK.



# Repository Structure vs. Installation Paths



# Code Examples

Code Examples can be found from Nordic or Zephyr.

This PC > OS (C:) > Nordic_SDKs > ncs > v1.5.0 > nrf > samples		
Name	Date modified	Type
bluetooth	3/26/2021 3:05 PM	File folder
bootloader	3/26/2021 3:05 PM	File folder
connectedhomeip	3/26/2021 3:05 PM	File folder
debug	3/26/2021 3:05 PM	File folder
ei_data_forwarder	3/26/2021 3:05 PM	File folder
ei_wrapper	3/26/2021 3:05 PM	File folder
esb	3/26/2021 3:05 PM	File folder
event_manager	3/26/2021 3:05 PM	File folder
mpsl	3/26/2021 3:05 PM	File folder
nfc	3/26/2021 3:05 PM	File folder
nrf_pc	3/26/2021 3:05 PM	File folder
nrf5340	3/26/2021 3:05 PM	File folder
nrf9160	3/26/2021 3:05 PM	File folder
openthread	3/26/2021 3:05 PM	File folder
peripheral	3/26/2021 3:05 PM	File folder
profiler	3/26/2021 3:05 PM	File folder
sensor	3/26/2021 3:05 PM	File folder
spm	3/26/2021 3:05 PM	File folder
tfm	3/26/2021 3:05 PM	File folder
zigbee	3/26/2021 3:05 PM	File folder
CMakeLists.txt	3/26/2021 3:05 PM	Text Document
Kconfig	3/26/2021 3:05 PM	File

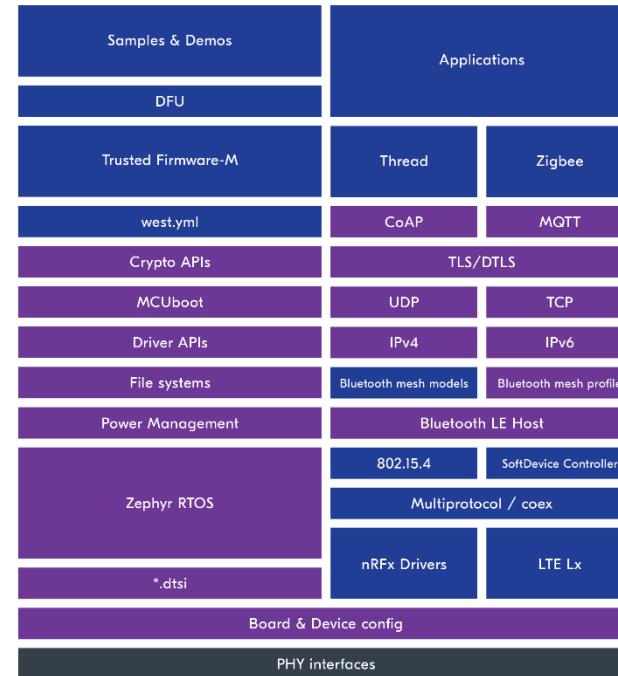
Nordic examples -> nrf/samples

This PC > OS (C:) > Nordic_SDKs > ncs > v1.5.0 > zephyr > samples		
Name	Date modified	Type
application_development	3/26/2021 3:09 PM	File folder
audio	3/26/2021 3:09 PM	File folder
basic	3/26/2021 3:09 PM	File folder
bluetooth	3/26/2021 3:09 PM	File folder
boards	3/26/2021 3:09 PM	File folder
cpp_synchronization	3/26/2021 3:09 PM	File folder
debug	3/26/2021 3:09 PM	File folder
display	3/26/2021 3:09 PM	File folder
drivers	3/26/2021 3:09 PM	File folder
hello_world	3/26/2021 3:09 PM	File folder
kernel	3/26/2021 3:09 PM	File folder
lorawan	3/26/2021 3:09 PM	File folder
mpu	3/26/2021 3:09 PM	File folder
net	3/26/2021 3:09 PM	File folder
philosophers	3/26/2021 3:09 PM	File folder
portability	3/26/2021 3:09 PM	File folder
posix	3/26/2021 3:09 PM	File folder
scheduler	3/26/2021 3:09 PM	File folder
sensor	3/26/2021 3:09 PM	File folder
shields	3/26/2021 3:09 PM	File folder
smp	3/26/2021 3:09 PM	File folder
subsys	3/26/2021 3:09 PM	File folder
synchronization	3/26/2021 3:09 PM	File folder
testing	3/26/2021 3:09 PM	File folder
tfm_integration	3/26/2021 3:09 PM	File folder
userspace	3/26/2021 3:09 PM	File folder
video	3/26/2021 3:09 PM	File folder
classic.rst	3/26/2021 3:09 PM	RST File
index.rst	3/26/2021 3:09 PM	RST File

Zephyr examples -> zephyr/samples

# nRF Connect SDK Code Base in Detail

- Code management, build and configuration tools allow developers to focus on the components required for their specific designs while having a powerful solution toolbox available
- Zephyr includes a lot of fundamental pieces of code and functionality
  - Middleware & messaging protocols (IP, HTTP, MQTT, CoAP, Modbus, ...)
  - Device drivers (e.g. sensors)
- → Use Zephyr as building block with the pieces that you need!

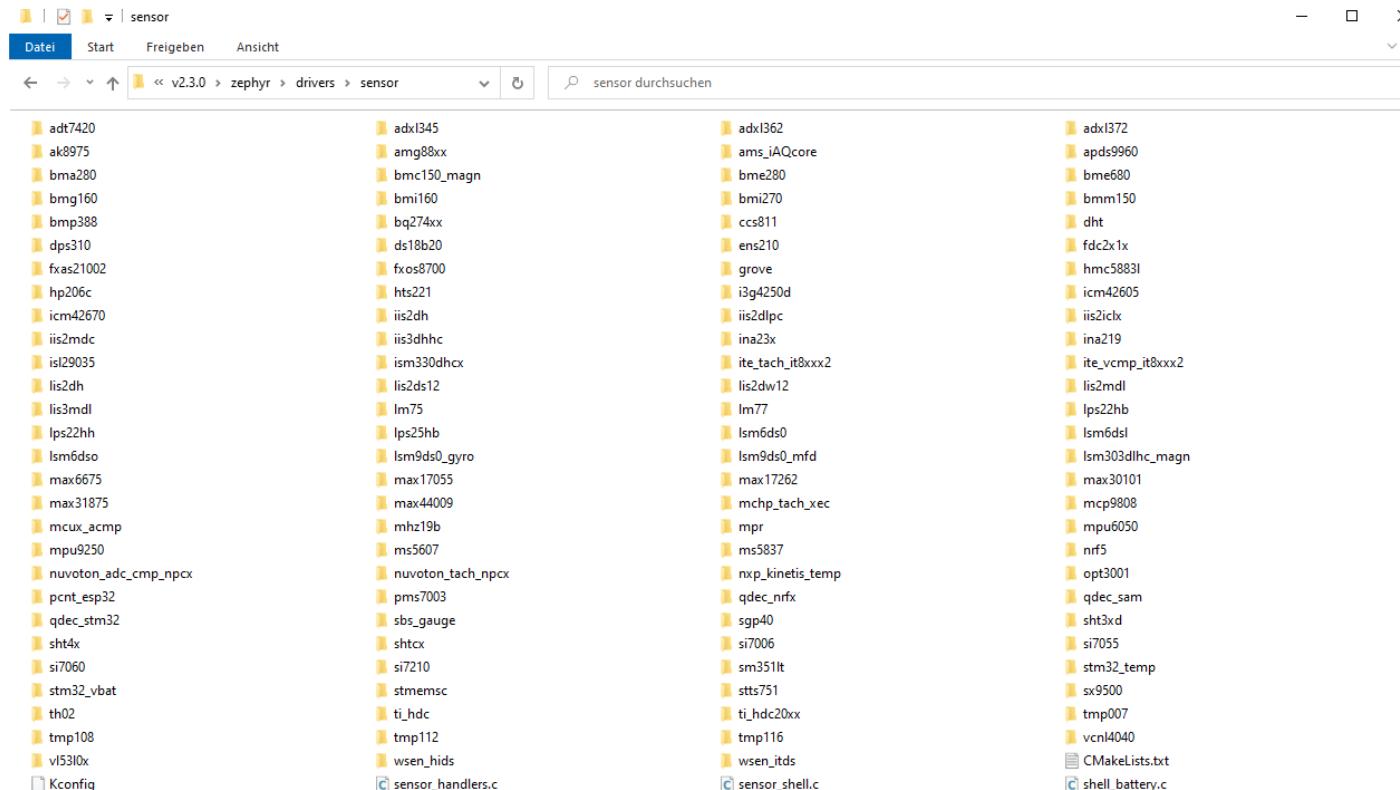


# Why the Zephyr RTOS?

- Zephyr is designed & built for low power wireless
- Zephyr is an independent technology project
  - Governed under the Linux Foundation
  - Contributions made by over 1000 industry embedded experts
- Zephyr is scalable
  - Very small configurations for memory constrained devices
  - Powerful, feature-rich, configurations for large memory, high-processing power devices (multiple MBs)
  - Designed for low power applications
- Zephyr includes a lot of fundamental pieces of code
  - Nordic can focus on other important features
- Nordic has been a member & contributor since 2016

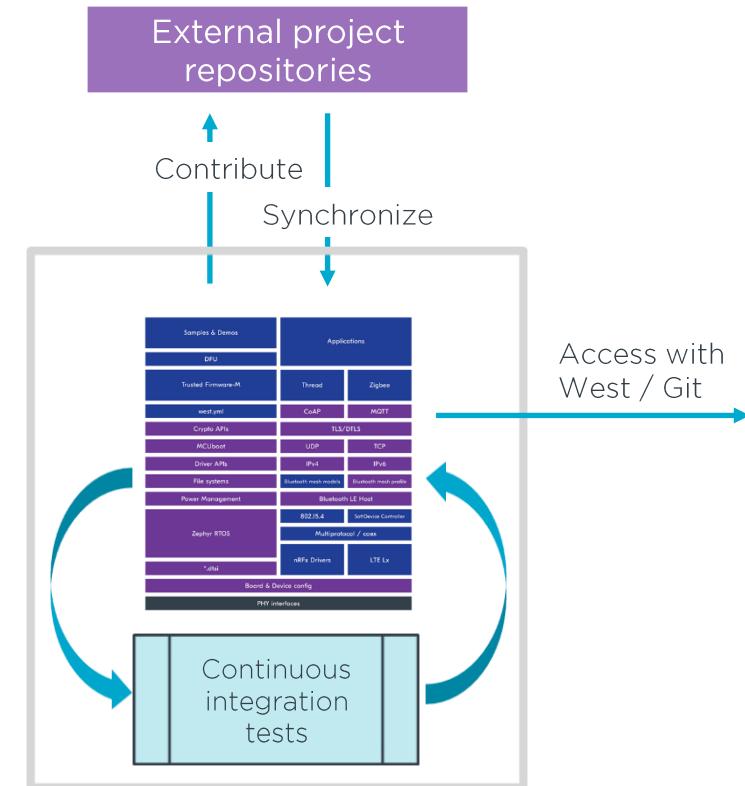


# Leverage code that is fully integrated



# Nordic Synchronizes with External Repos

- nRF Connect SDK is a single platform
- All source code is distributed by Nordic
- Includes open-source code from external projects
- Nordic contributes to, and synchronizes with external projects
- Nordic runs integration test on all source code and manages configuration
- Customers clone a tag using git and west



# nRF Connect SDK

## Toolchain

# Manage Source Code and Configurations

West  
Multi-repository  
management tool

Kconfig  
Source module / feature  
configuration for compile

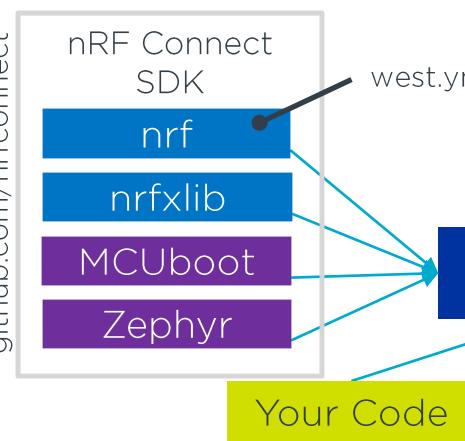
Device Tree  
Target Board / Device  
description

Clone / update

Configure features

Configure target

github.com/nrfconnect



Kconfig & prj.conf

Code  
Base

Application 1

Application 2

Application ...

\*.dts



Github

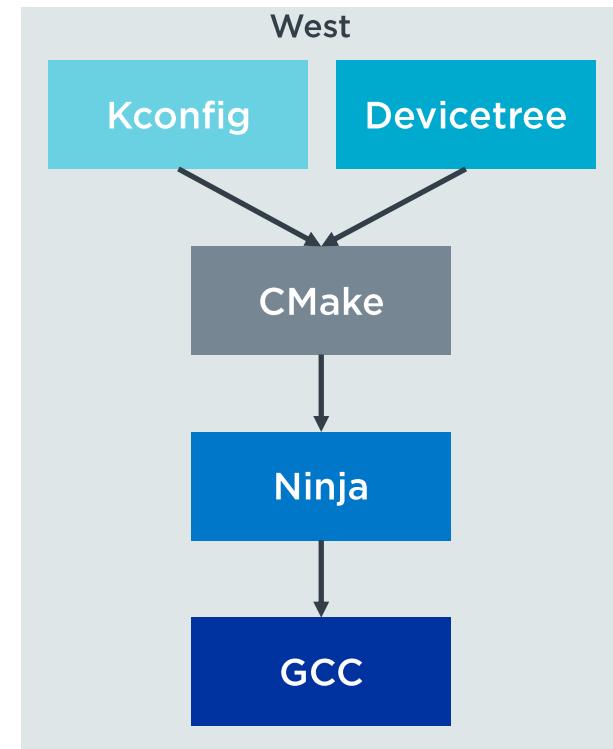
Your Development Machine

Hardware

# nRF Connect SDK

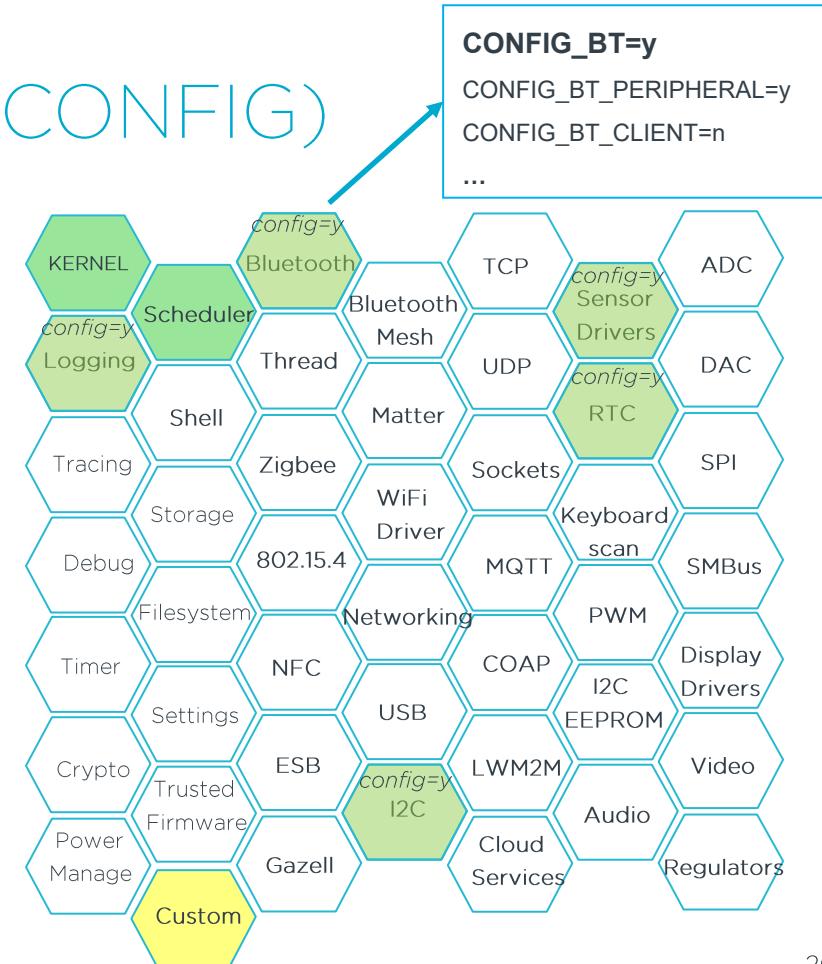
## Toolchain

- Kconfig
  - Describes SW and system features, generates C definitions to configure the system and include libraries without changing the source code (e.g. BLE, sensors, RTOS services (logging, shell))
  - Kconfig and prj.conf files are merged into one .config file for CMake (see *build/<proj.name>/zephyr/.config*)
- DeviceTree (dts,dtsi)
  - Describes HW, pin layout
  - Allows for flexible HW modification via an overlay file
  - → Build for different PCB designs and SoCs without changing source code
- [Zephyr Documentation: Device Tree Guide](#)



# Configuration System (KCONFIG)

- Zephyr Kernel and Sub-System can be configure at build time
  - Add software modules to your project
  - If a software module was added, further CONFIG symbols appear and allow you to configure the module
- Custom Kconfig is possible
- Goal: Configure software features without changing source code



# Configuration for an Application

Kconfig  
Configuration  
Options  
(Kconfig  
Symbols)



C Code  
(#define)

C Code header file  
autoconf.h

# nRF Connect SDK

## Toolchain cont.

### CMake

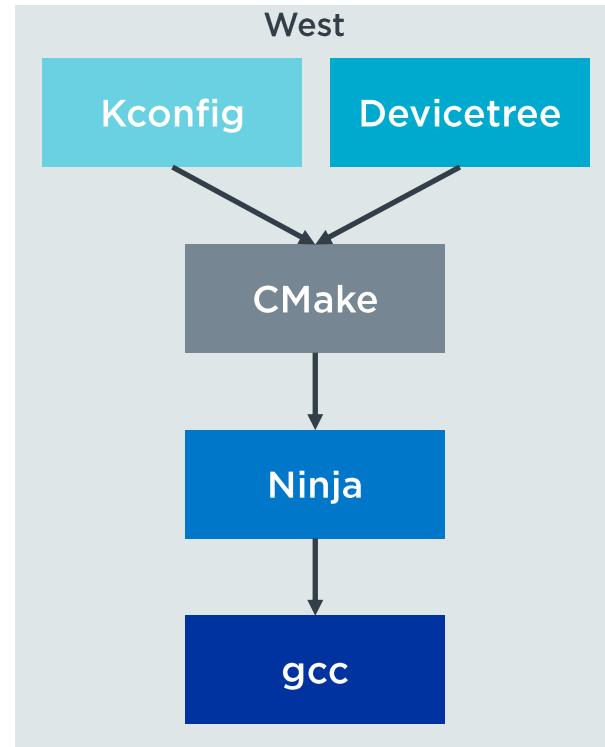
- Uses the information from Kconfig and the devicetree to generate build files.

### Ninja

- Similar to make
- Faster than make when performing incremental builds
- Requires CMake in order to generate build files

### gcc

- The gcc compiler is used to creates the executables (hex/elf files)



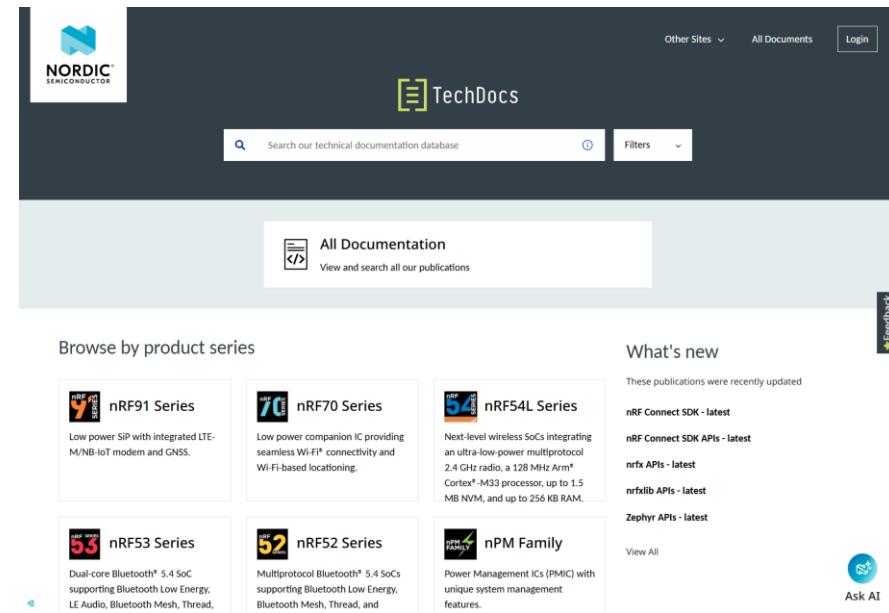
# Documentations & Help Guidance

```
190     pin_value = gpiote_get_pin_value(gpiote_channel);
191     if (pin_value != expected_final_value) {
192         // The TDD requires to check if other pin values did not change
193         // due to the pin being set to an expected value.
194         // This is not an expected behavior.
195         // It is used by the framework
196         // to make sure frameworks can handle multiple pins at once.
197     }
198 }
```

```
199 void gpiote_set_pin(gpiote_channel_t *gpiote_channel,
200                     uint32_t pin, uint32_t expected_final_value);
201
202 void gpiote_set_all_pins(gpiote_channel_t *gpiote_channel);
203
204 TEST_START
205 // this test is very heavy on tester communication, we want to decrease the communication and
206 // period to the minimal value possible
207 tester_ib_set_communication_timeout(1);
208
209 gpiote_ib_in_init();
210
211 gpiote_shared_ib_test_config_t gpiote_test = {
212     .initial_value = TEST_GPIOTE_INITIAL_VALUE,
213     .outinit = TEST_GPIOTE_INITIAL_VALUE,
214     .mode = GPIOTE_SHARED_IB_MODE_TEST,
215     .gpiote_init = TEST_GPIOTE_STATE_INIT,
216     .test_task_type = TEST_TASK_TYPE,
217 };
218
219 set_all_tester_mos_framework_pins_ib_out();
220
221 for (uint32_t gpiote_index = 0; gpiote_index < GPIOTE_SHARED_IB_INSTANCE_COUNT; gpiote_index++) {
222     for (uint32_t channel_number = 0; channel_number < GPIOTE_SHARED_IB_NUM_CHANNELS;
223          channel_number++) {
224         gpiote_ib_channel_t gpiote_channel = { .instance_index = gpiote_index,
225                                              .channel_number = channel_number };
226         gpiote_ib_set_pin(gpiote_channel, pin);
227     }
228 }
229
230 // Testing initial value
231 gpiote_ib_set_pin(gpiote_channel, pin);
232
233 gpiote_ib_out(gpiote_channel);
234
235 gpiote_ib_set_all_pins(gpiote_channel);
```

# Where do I find documentation?

- Nordic created a new, unified website for its documentation called TechDocs:
  - [www.docs.nordicsemi.com](http://www.docs.nordicsemi.com)
- **HW:** Tiles represent the product series on the main page
- **SW:** Navigation found on the right-hand side
  - Links to various repositories that are part of the nRF Connect SDK
  - (Entry Page: “nRF Connect SDK – latest”)



# nRF Connect SDK Documentation

- nRF Connect SDK: [Documentation link](#)
- Ensure to view the documentation corresponding to your NCS version in use
  - Click on the version field / drop down menu
  - “Latest” refers to the current development branch
  - Other versions listed as per SDK releases
- Note that not all sub pages list all SDK release versions, as the respective sub page might not exist in that SDK release

Home > nRF Connect SDK - latest > Introduction

## Introduction

The screenshot shows a documentation page for the nRF Connect SDK. At the top, there's a navigation bar with links for Home, nRF Connect SDK - latest, and Introduction. Below the navigation is a header with the title "Introduction". Under the title, there's a "Version:" dropdown set to "nRF Connect SDK latest", a "Last Updated" timestamp of "Jan 21, 2025", a "2 minute read" estimate, and a "Summarize" button. To the right of the summary button are links for various hardware components: nRF9151, nRF9161, nRF9160, Thingy:91, Thingy:91X, and nRF70 Series. The main content area starts with a heading "The nRF Connect SDK" followed by a detailed description of its purpose and supported platforms (Windows, Linux, macOS). Below the main content, there's a section titled "Based on Zephyr and open source".

The nRF Connect SDK is a unified software development kit for building low-power wireless applications based on the Nordic Semiconductor nRF52, nRF53, nRF54, nRF70, and nRF91 Series wireless devices. It supports Microsoft Windows, Linux, and macOS for development.

The nRF Connect SDK has the following distinguishing features:

Based on Zephyr and open source

# Have you seen this button?

It's found on following web pages:

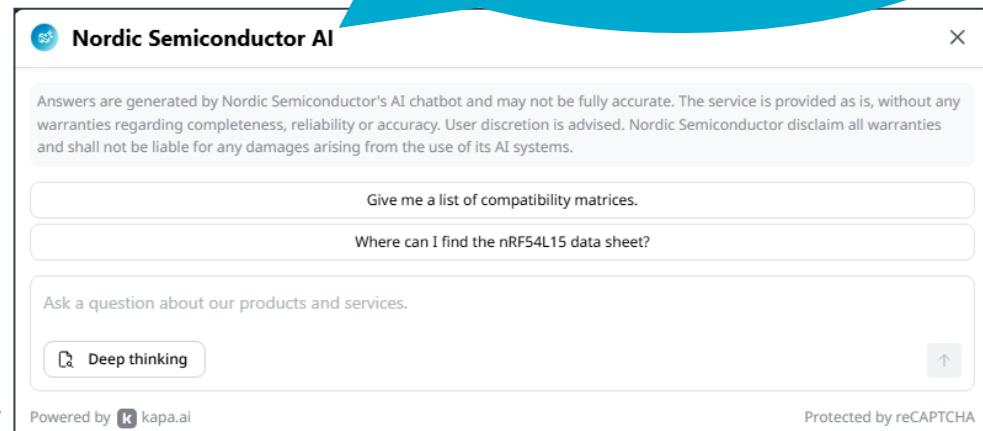
- [Nordic Semiconductor's TechDocs](#)
- [Nordic Semiconductor's DevZone](#)



# Nordic Semiconductor AI Agent

- Answering questions about Nordic Semiconductor's products, solutions, and documentation.
- AI Agent uses Nordic Semiconductor resources to look for an answer:
  - Official Nordic Semiconductor documentation
  - Nordic Semiconductor Blog posts
  - Nordic Semiconductor Forum Discussions
  - Nordic Semiconductor Support Resources
- Different Languages are supported

I am specialized in answering questions about Nordic Semiconductor's products, solutions, and documentation. If you have questions about Nordic products or technologies, I am happy to help!



# Recommendation: Nordic DevAcademy !



DevAcademy

- Interactive Learning Platform around Nordic's Software Solution and Products
- Start with [nRF Connect SDK Fundamentals](#)
- Enhance Knowledge with [BLE Fundamentals](#)
- Advanced topics in [nRF Connect SDK Intermediate](#)

# Nordic Support

Webinars



**NORDICTECH**  
WEBINARS

Regular webinars about our products and  
technologies

[www.nordicsemi.com/Events/Webinars](http://www.nordicsemi.com/Events/Webinars)

Tech Support & Forum



Dedicated Application Engineering Support from  
Norway and open community forum: DevZone Portal

<https://devzone.nordicsemi.com>