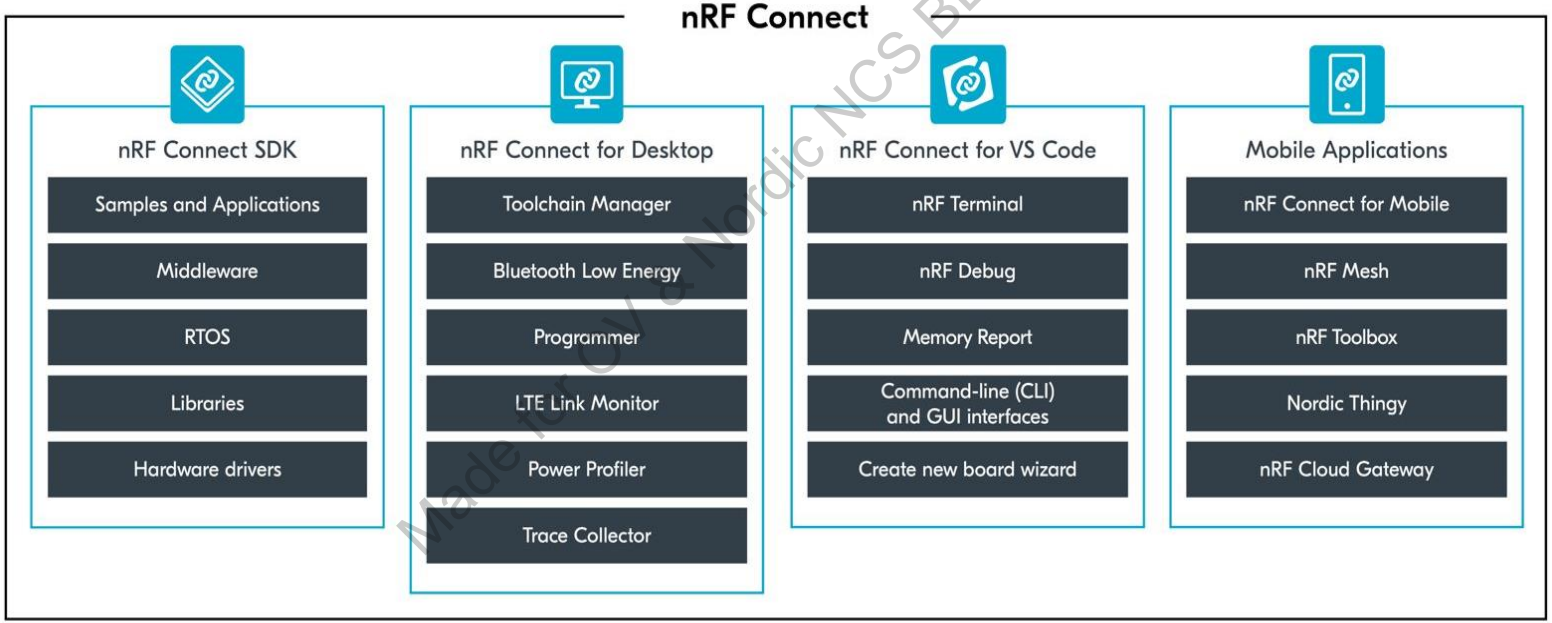
# nRF Connect SDK and tools

# nRF Connect



# nRF Connect SDK

## Unified code base and toolchain for all Nordic SoCs


Bluetooth · THREAD · matter · ANT · zigbee · Wi-Fi · LTE-M · NB-IoT · GPS

- 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

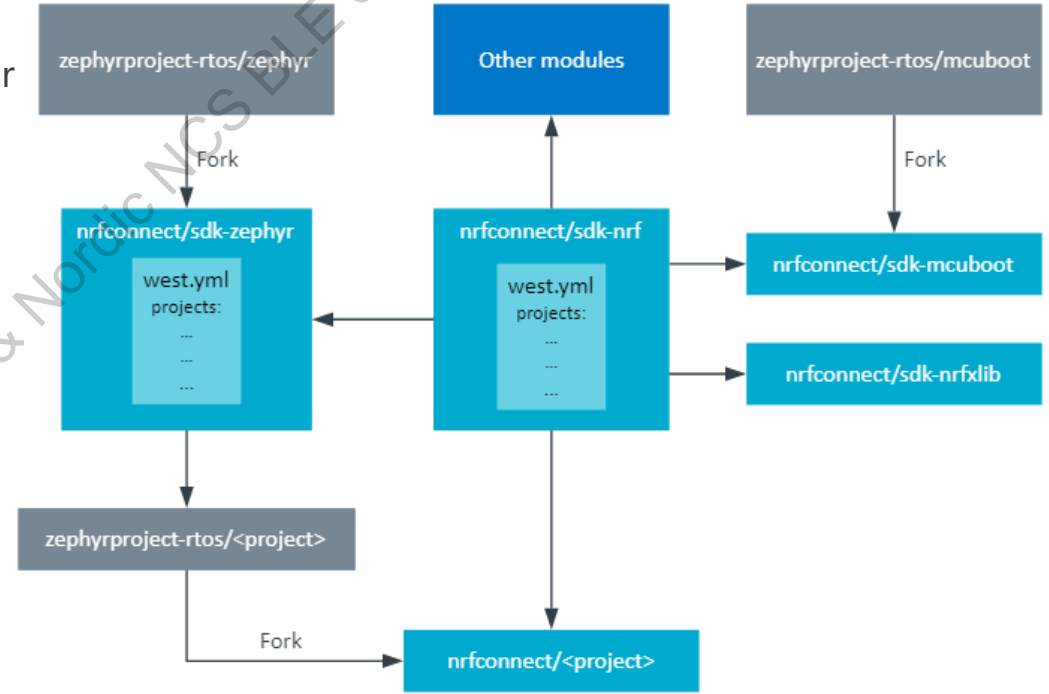
- Code base consists of several integrated repositories
  - Code coming from **Nordic** (blue)
  - Code coming from **Open Source** (OS) repositories (purple)
- Contains an RTOS, application code, connectivity protocols, wireless stacks, peripheral drivers & more

# nRF Connect SDK Repositories

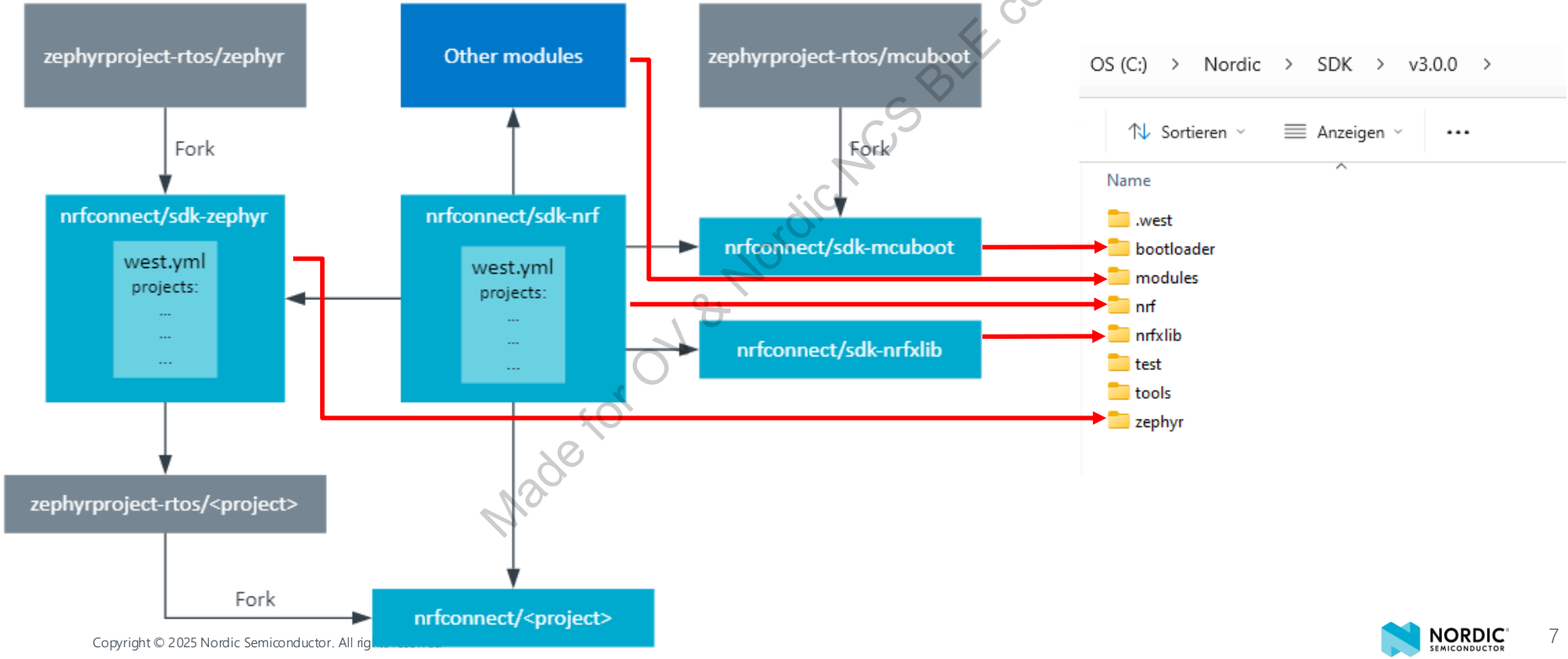
- **nRF**: application & connectivity protocols
- **nrfxlib**: compiled libraries where Nordic cannot distribute source code
- **nrfx**: peripheral drivers
- **Zephyr**: RTOS & board configuration (OS)
- **MCUboot**: Secure Bootloader (OS)
- Other repositories
  - Trusted Firmware-M, Matter, etc.

# 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.
- Use repositories as building blocks with the pieces that you need!



# Repository Structure vs. Installation Paths

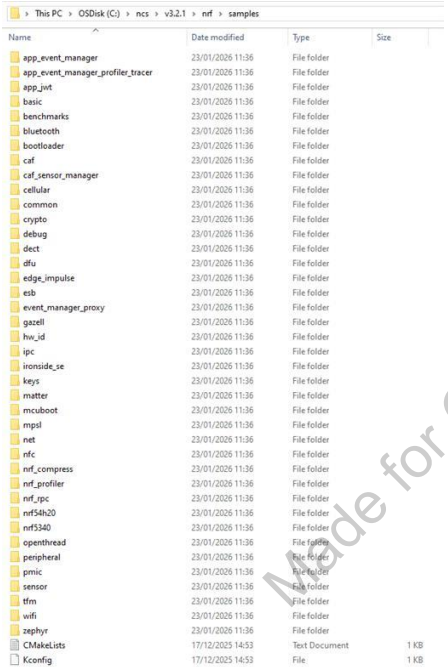


# Why the Zephyr RTOS?

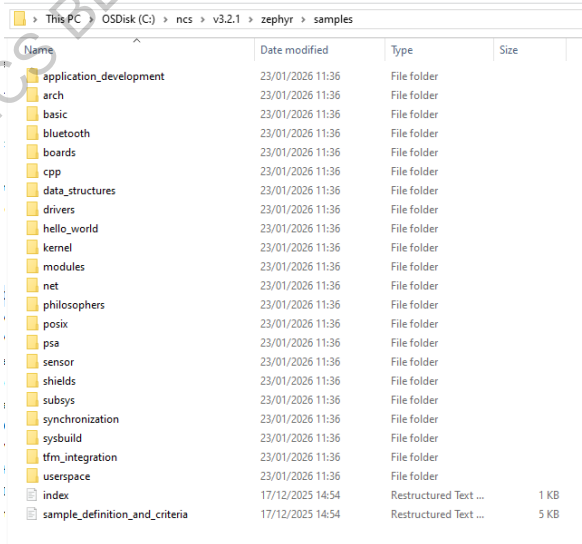

- Zephyr is designed & built for low power wireless
- Zephyr is an independent technology project
  - Governed under the Linux Foundation
  - Contributions made by over 1000 industry embedded experts
- Zephyr is scalable
  - Very small configurations for memory constrained devices
  - Powerful, feature-rich, 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

# Code Examples

- Code Examples can be found from Nordic or Zephyr.




Nordic examples **-> nrf/samples**




Zephyr examples **-> zephyr/samples**

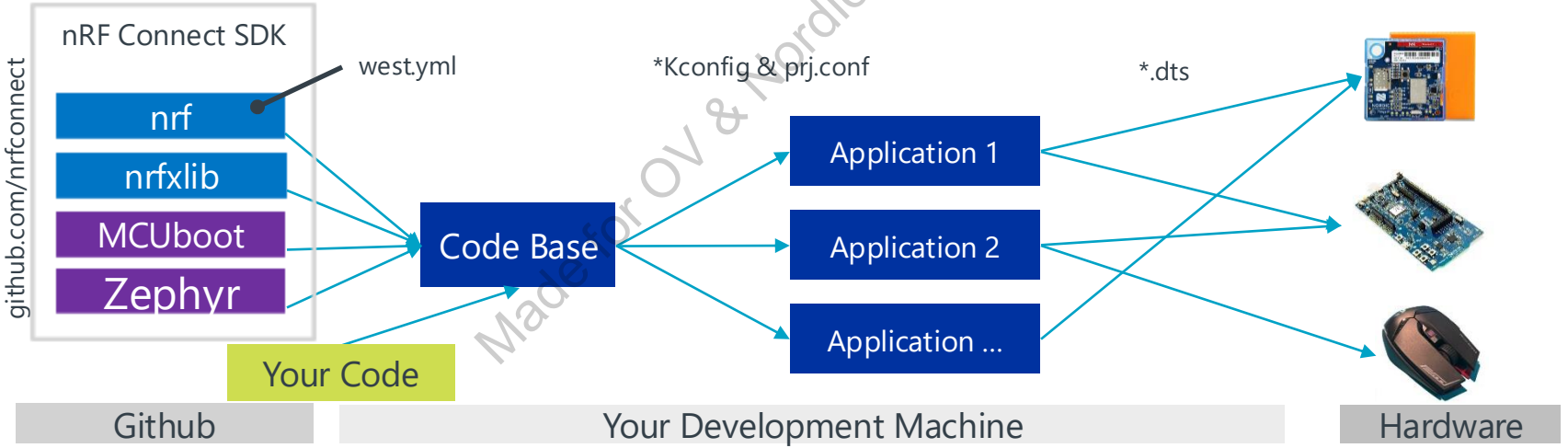
# Manage Source Code and Configurations

West
Multi-repository management tool

Kconfig
Source module / feature configuration for compile

Devicetree
Target Board / Device description

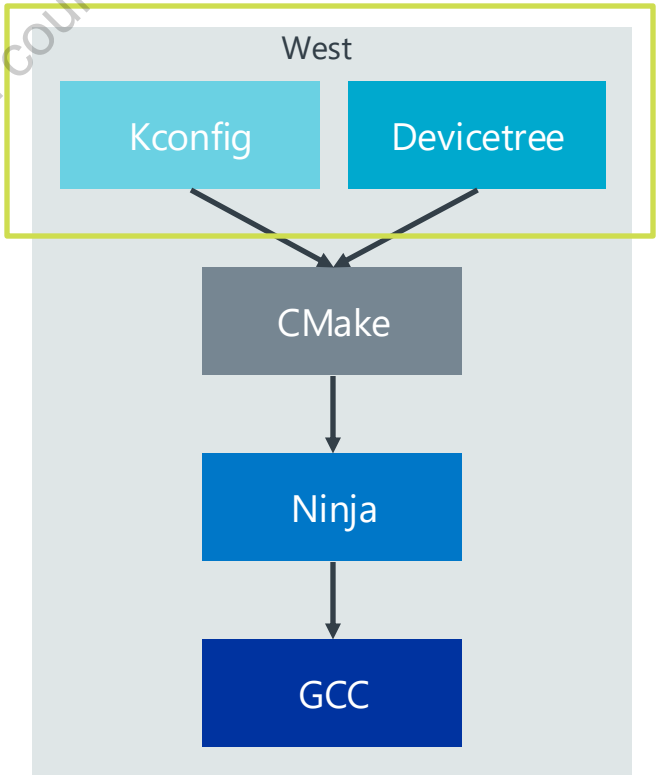
**Clone / update** → **Configure features** → **Configure target**



# nRF Connect SDK

## Kconfig and Devicetree

- 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



# nRF Connect SDK

## Kconfig and Devicetree cont.

prj.conf

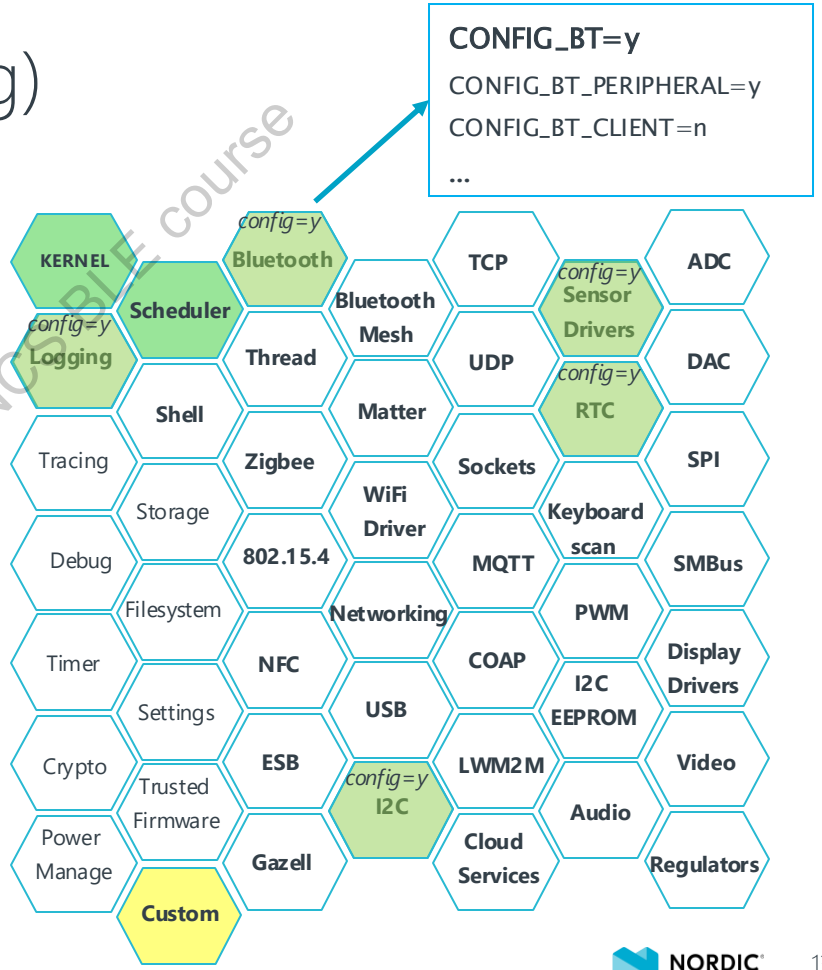
```
# Enable logging
CONFIG_LOG=y

# Set the logging level to debug
CONFIG_LOG_DEFAULT_LEVEL=4

# Enable the shell
CONFIG_SHELL=y

# Enable the Bluetooth subsystem
CONFIG_BT=y

# Enable the Bluetooth peripheral role
CONFIG_BT_PERIPHERAL=y

# Enable UART driver
CONFIG_SERIAL=y
```

.overlay / .dts

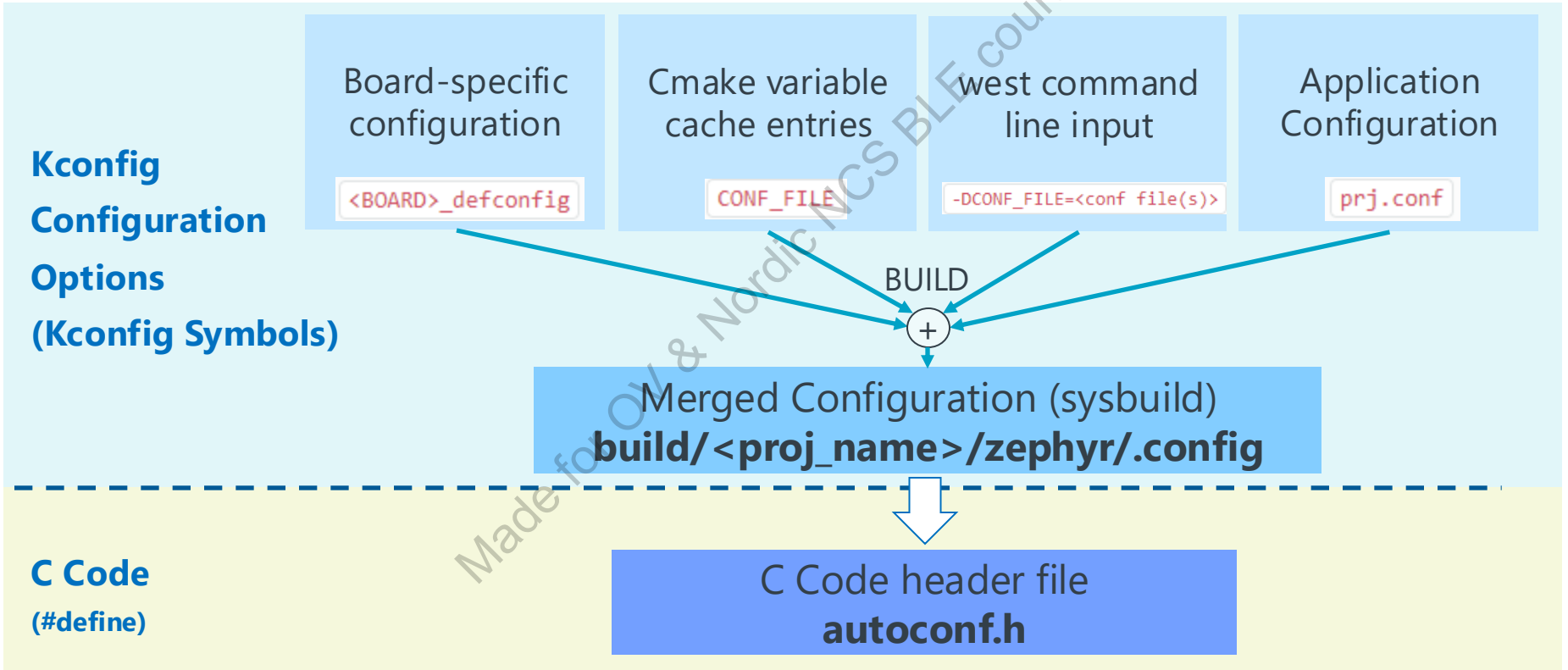
```
/ {
    chosen {
        zephyr,shell-uart = &uart0;
    };

    uart0: uart0 {
        status = "okay";
        current-speed = <115200>;
        # Configure pins
        pinctrl-0 = <&uart0_default>;
        pinctrl-1 = <&uart0_sleep>;
        pinctrl-names = "default", "sleep";
    };
};


[...]
```

# 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
- Enable with CONFIG_...
- **Goal**: Configure software features without changing source code



# Configuration for an Application (Kconfig flow)



# nRF Connect SDK Toolchain (CMake, Ninja, gcc)

- 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)