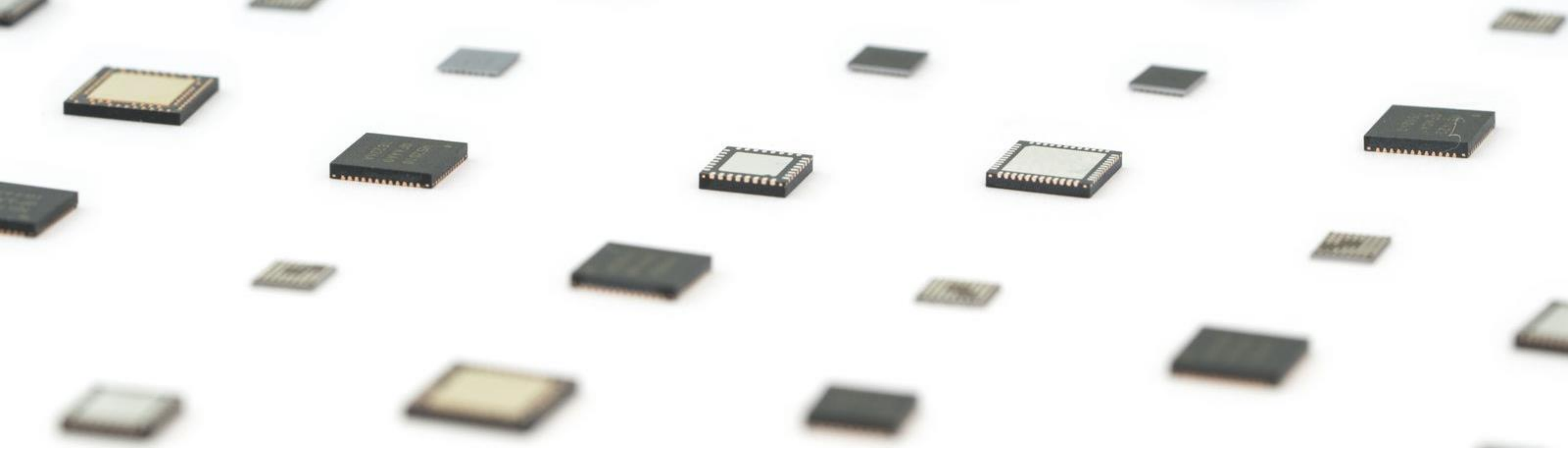


AVNET 4 in 1 Training – Nordic Semiconductor

Part 1

nRF Connect SDK

25. October 2022

Christian Kurz

Christian.Kurz@nordicsemi.no

Agenda – Nordic Part 1

- Introduction to *nRF Connect SDK*
- *Hands-on: Modem Firmware update*
- Hands-on: Create a project from scratch
- Hands-on: Adding a Sensor to our Project
  - Connect Sensor Board to nRF9160DK
  - Adding Sensor Driver via KCONFIG
  - DeviceTree: Use Zephyr BME280 Sensor Driver
- Summary

Nordic Semiconductor product offering

Short Range

nRF 52 SERIES

nRF 53 SERIES

nRF 21 SERIES

Bluetooth Low Energy

Matter

Thread/Zigbee

Proprietary

Wi-Fi

nRF 70 SERIES

Wi-Fi 6

2.4GHz +5GHz

Cellular

nRF 91 SERIES

LTE-M

NB-IoT

GPS

PMIC

nPM FAMILY

Li-Ion charger

LDO

DC/DC buck converter

nRF Cloud

Cloud connectivity

Location support

SCELL/MCELL

AGPS/PGPS

Device management

nRF Connect SDK

Introduction

nRF Connect

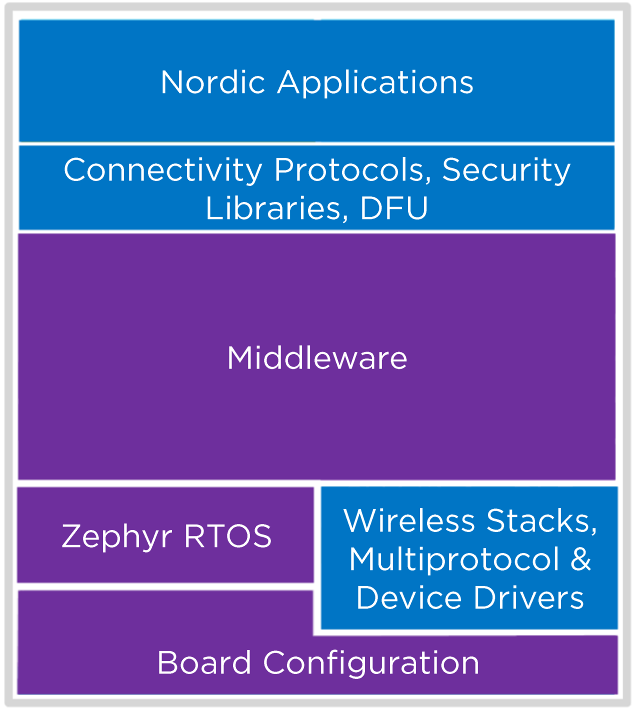
- nRF Connect is our umbrella of tools to help developers build and debug their applications quickly
- Consist of
  - nRF Connect SDK
  - nRF Connect for Desktop
  - nRF Connect for VS Code
  - nRF Cloud
  - Mobile applications

nRF Connect SDK

- One code base and toolchain for nRF70, nRF91, nRF53, nRF52 and nRF21 Series
  - Optional for nRF52 Series (>= v1.3.0)
- Includes LTE-M/NB-IoT/GPS, Bluetooth Low Energy, Bluetooth mesh, Thread/Zigbee, Matter, ESB, Gazell, NFC, WiFi
- Bluetooth v5.3 qualified Host and Controller stack since v2.0.0

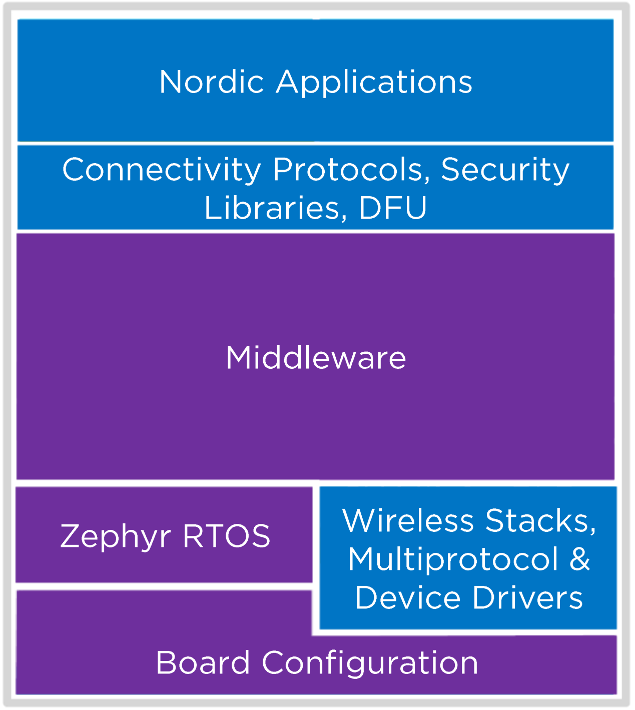
Code base

- Contains app code, connectivity protocols, wireless stacks and peripheral drivers
- Code is organized into several repositories (Nordic - blue and Open Source (OS) code - purple)


Nordic Applications
Connectivity Protocols, Security Libraries, DFU
Middleware
Zephyr RTOS
Wireless Stacks, Multiprotocol & Device Drivers
Board Configuration

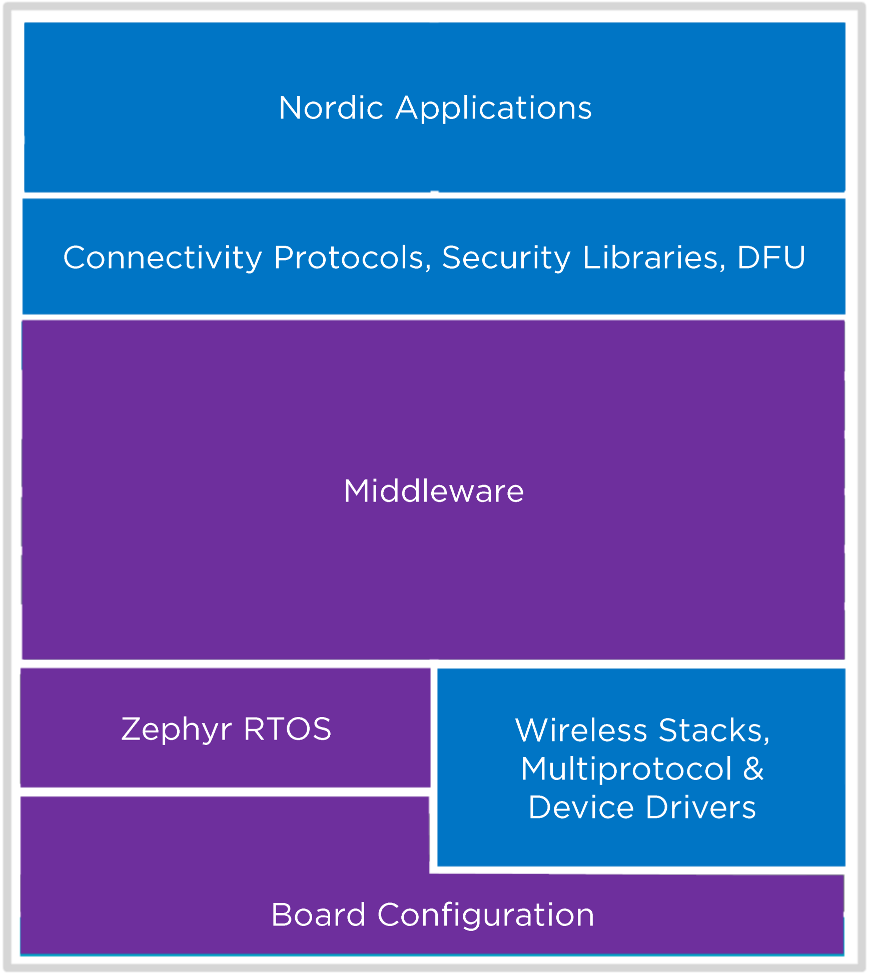
nRF Connect SDK repositories

- 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 Applications
Connectivity Protocols, Security Libraries, DFU
Middleware
Zephyr RTOS
Wireless Stacks, Multiprotocol & Device Drivers
Board Configuration

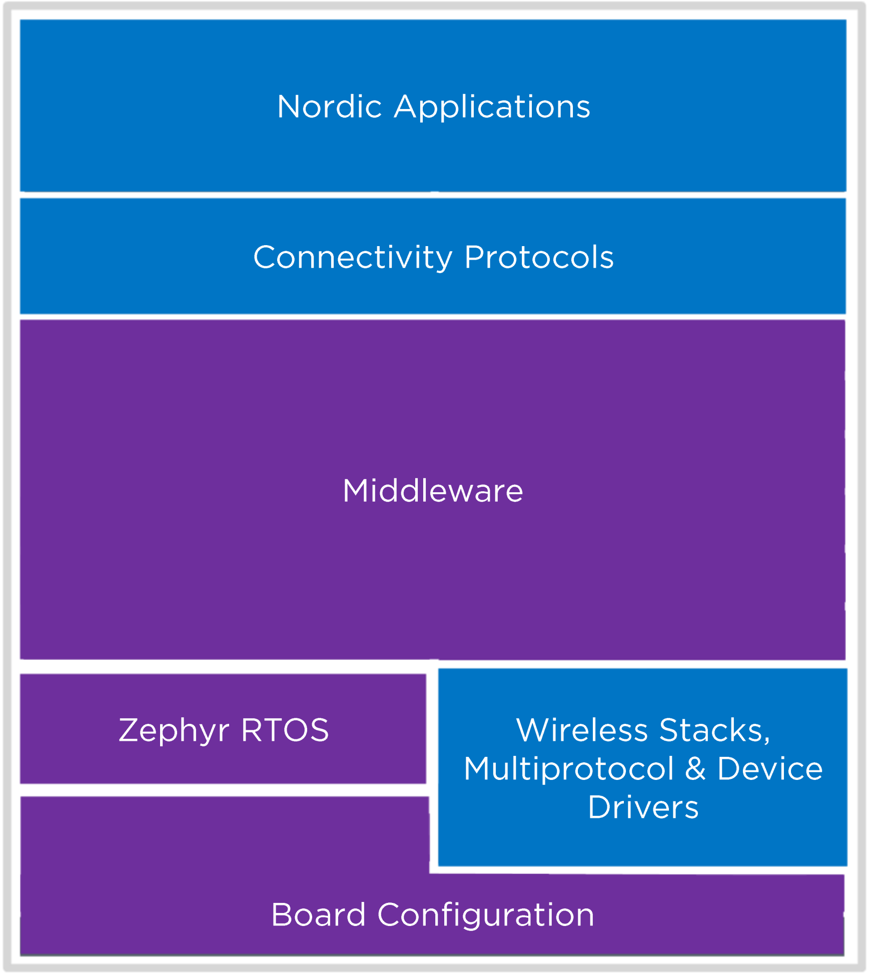
Nordic SW

- The main SDK repositories with samples, applications and connectivity stacks
- Source code exclusively written by Nordic Semiconductor
- Licensed with permissive Nordic 3-clause or 5-clause BSD license


Nordic Applications
Connectivity Protocols, Security Libraries, DFU
Middleware
Zephyr RTOS
Wireless Stacks, Multiprotocol & Device Drivers
Board Configuration

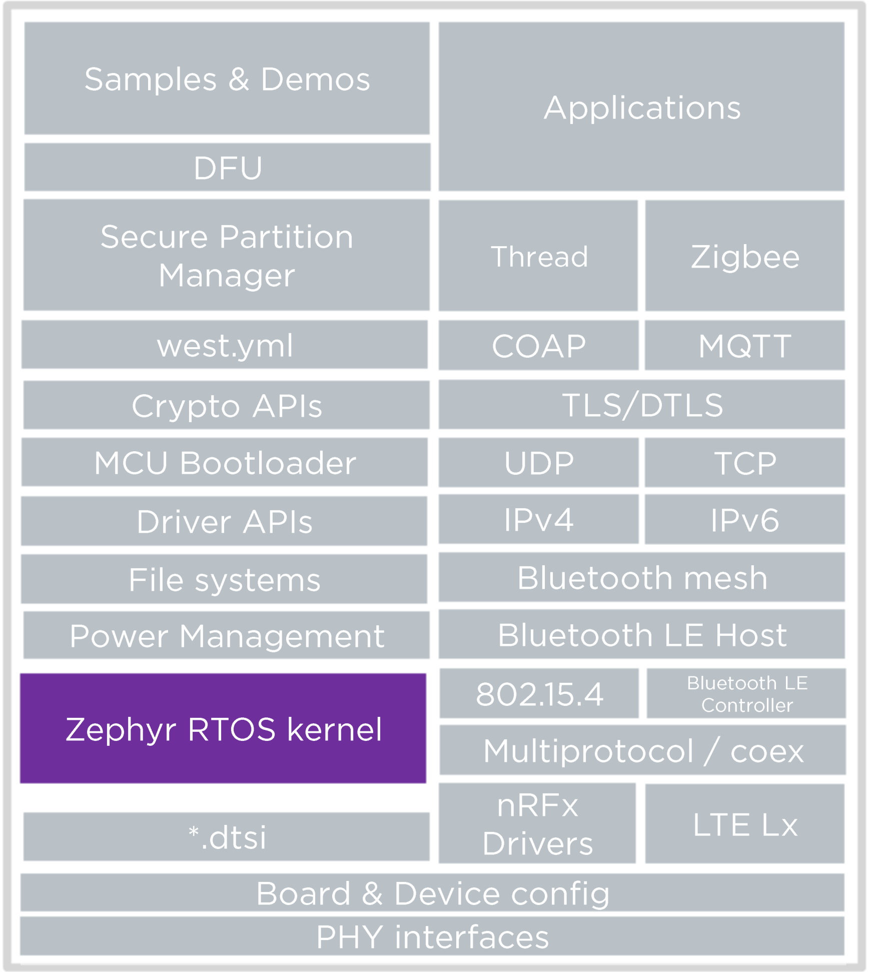
Open-source SW

- nRF Connect SDK re-distributes OS for standard platform components
- Nordic collaborates with communities of industry experts to deliver these components as part of the nRF Connect SDK
- OS repositories are licensed with permissive license (e.g. Apache 2.0)



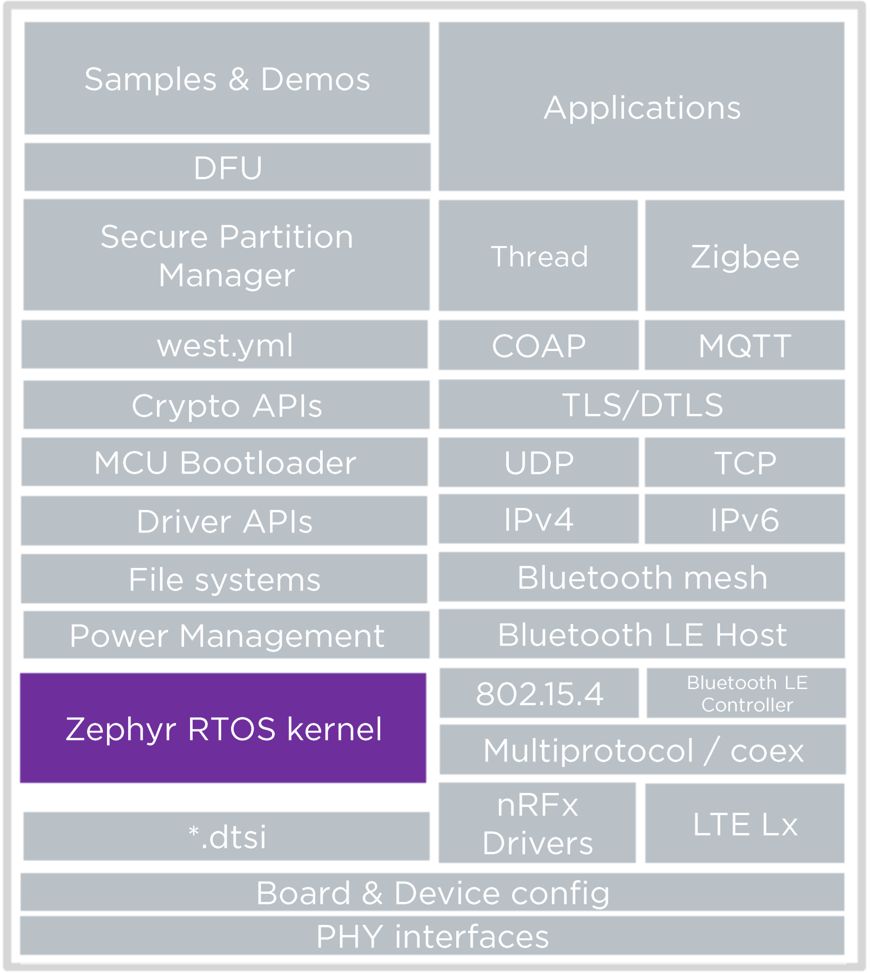
What is an RTOS?

- Real Time Operating System
  - Goal is to ensure predictable/deterministic execution pattern
  - Embedded systems often have strict timing requirements
  - Scheduler decides which task to execute at which time
  - Achievable by setting a priority for each execution thread



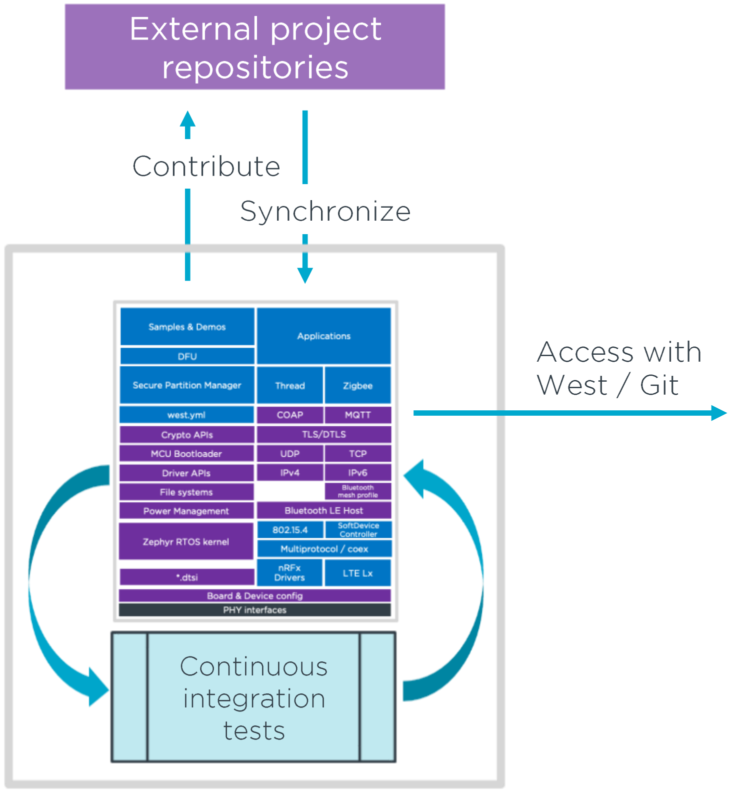
Why use an RTOS?

- How does an RTOS help a product developer?
  - Separation of Concern
  - More portable & re-usable applications
  - Controls application complexity in large memory devices
- Higher-level programming model
  - -> faster time to market



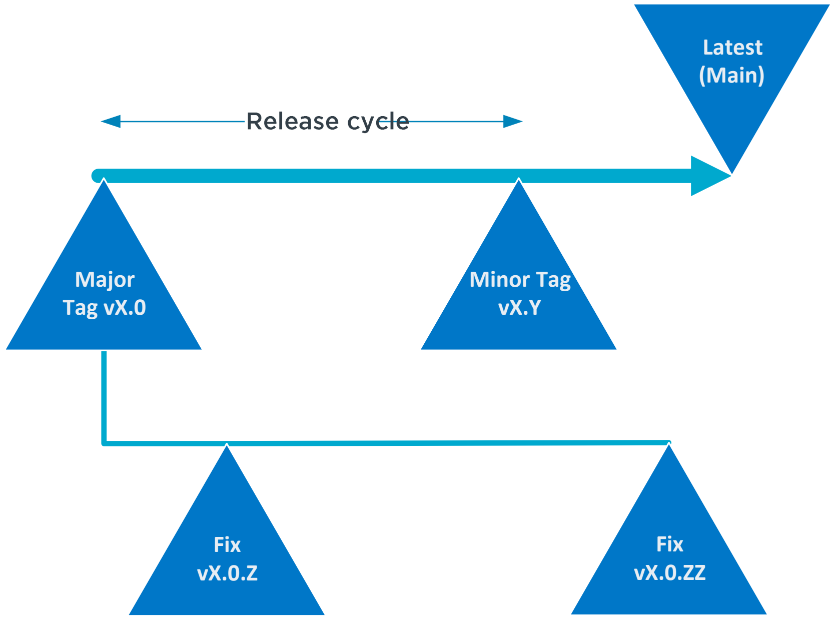
Nordic synchronizes with external repos

- 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

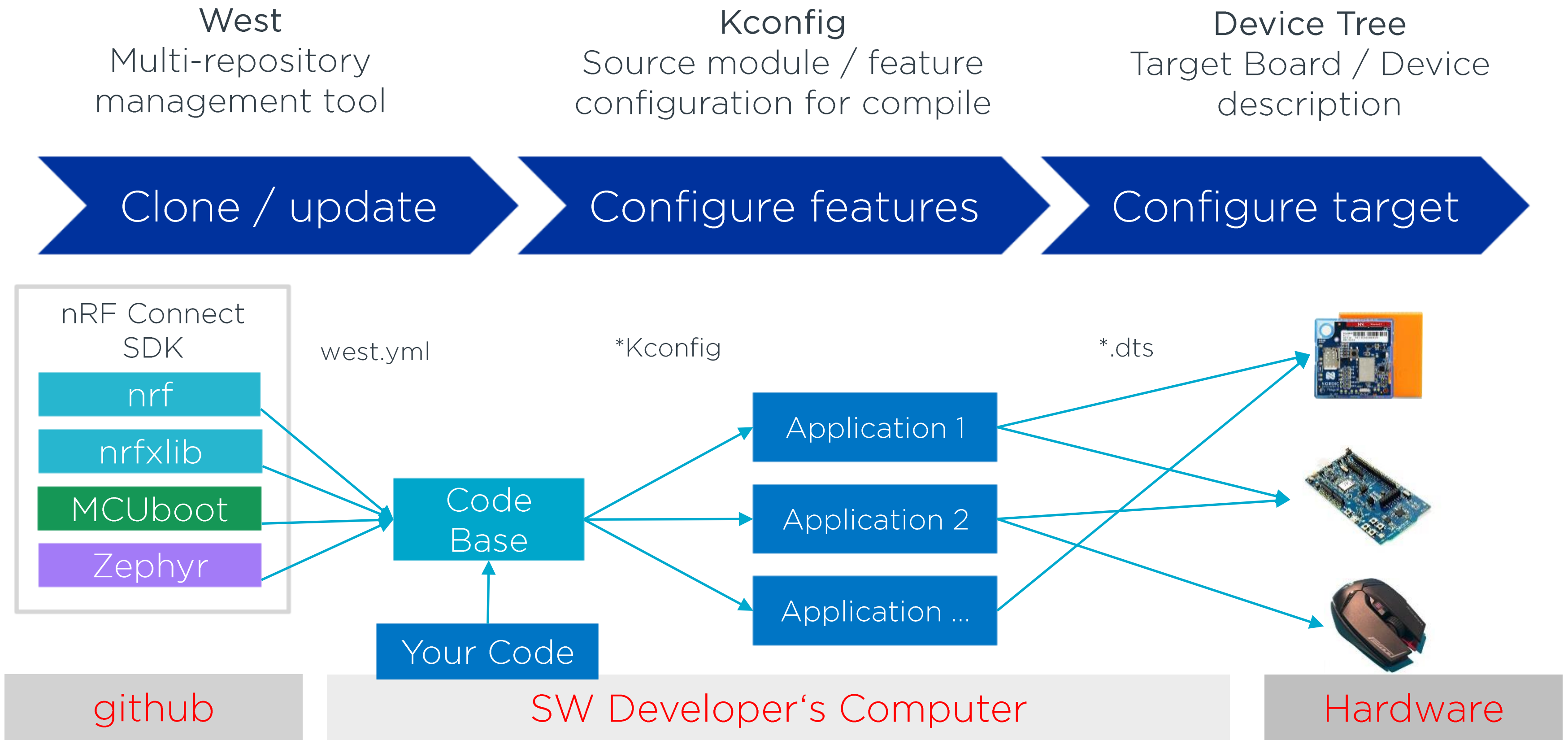


Release cycles

- Regular releases (e.g. quarterly)
- Publicly hosted on GitHub
- Fixes released as needed
  - Long term supported releases can have fixes applied and delivered after new releases
- Latest development version available
- Version control management with Git:
  - manage new version and fix adoption
  - tool supported merging

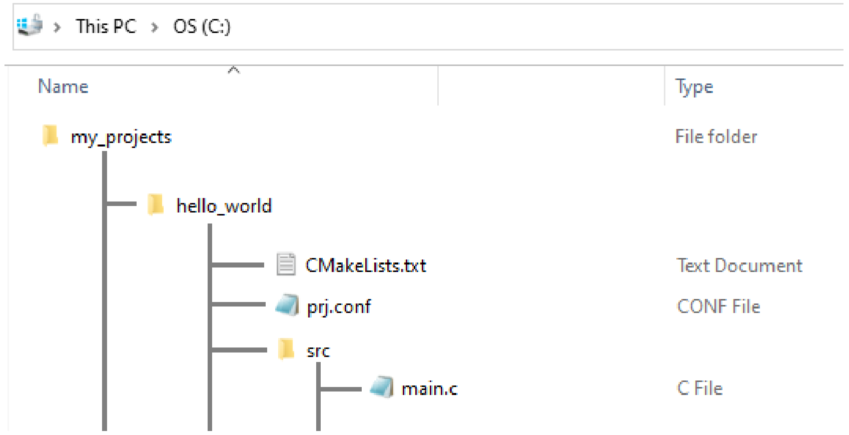


Manage Source Code and Configurations



Creating an own Project from Scratch (minimal)

In this simple example, the sample project will be put outside the nRF Connect SDK folder. This is a cleaner and more modular approach than building it directly in the nrf folder since you keep the project detached from the nRF Connect SDK.



Name	Type
my_projects	File folder
hello_world	
CMakeLists.txt	Text Document
prj.conf	CONF File
src	
main.c	C File

File: CMakeLists.txt

```
cmake_minimum_required(VERSION 3.8.2)
find_package(Zephyr REQUIRED HINTS $ENV{ZEPHYR_BASE})
project(NONE)
target_sources(app PRIVATE src/main.c)
```

File: prj.conf

```
CONIFG_SERIAL=y
```

File: src/main.c

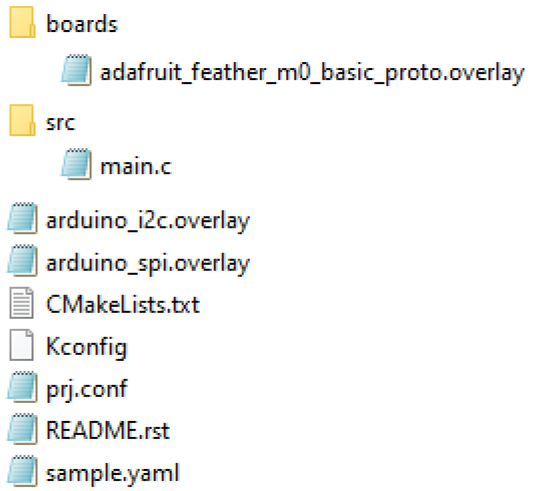
```
#include <zephyr.h>
#include <sys/printk.h>
void main(void){
            printk("Hello World!\n");
}
```

Typical NCS Project

- *.overlay file:
  - Configure HW features without changing source code
  - Optional DTS format files which override BOARD.dts
- Kconfig. file:
  - Configure Software Modules without changing source code

Example:

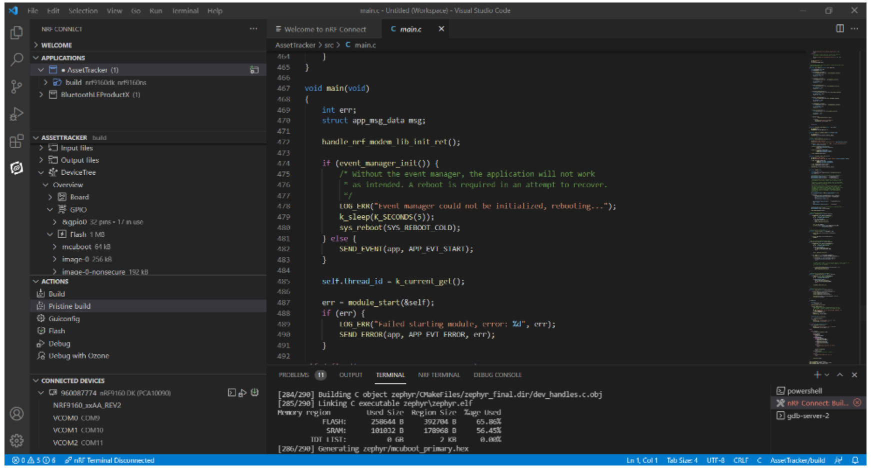
C:\Nordic_SDKs\ncs\v2.1.0\zephyr\samples\sensor\bme280



```
boards
    adafruit_feather_m0_basic_proto.overlay
src
    main.c
arduino_i2c.overlay
arduino_spi.overlay
CMakeLists.txt
Kconfig
prj.conf
README.rst
sample.yaml
```

IDE support

- nRF Connect for Visual Studio Code
  - Built from the ground up for nRF Connect SDK
  - Highly extendable and configurable
  - CLI and GUI Interfaces
  - Cross-platform support
    - Windows, macOS, Linux
  - Create new board wizard
  - Rich set of tutorial videos



```c
        }
}

void main(void)
{
    int err;
    struct app_msg_data msg;

    handle_nrf_modem_lib_init_ret();

    if (event_manager_init()) {
        /* Without the event manager, the application will not work
         * as intended. A reboot is required in an attempt to recover.
         */
        LOG_ERR("Event manager could not be initialized, rebooting...");
        k_sleep(K_SECONDS(5));
        sys_reboot(SYS_REBOOT_COLD);
    } else {
        SEND_EVENT(app, APP_EVT_START);
    }

    self.thread_id = k_current_get();

    err = module_start(&self);
    if (err) {
        LOG_ERR("Failed starting module, error: %d", err);
        SEND_ERROR(app, APP_EVT_ERROR, err);
    }

```

```
[284/290] Building C object zephyr/CMakeFiles/zephyr_final.dir/dev_handles.c.obj
[285/290] Linking C executable zephyr\zephyr.elf
Memory region         Used Size  Region Size  %age Used
           FLASH:      258644 B       392704 B     65.86%
            SRAM:      101032 B       178968 B     56.45%
        IDT_LIST:          0 GB         2 KB      0.00%
[286/290] Generating zephyr/mcuboot_primary.hex
```

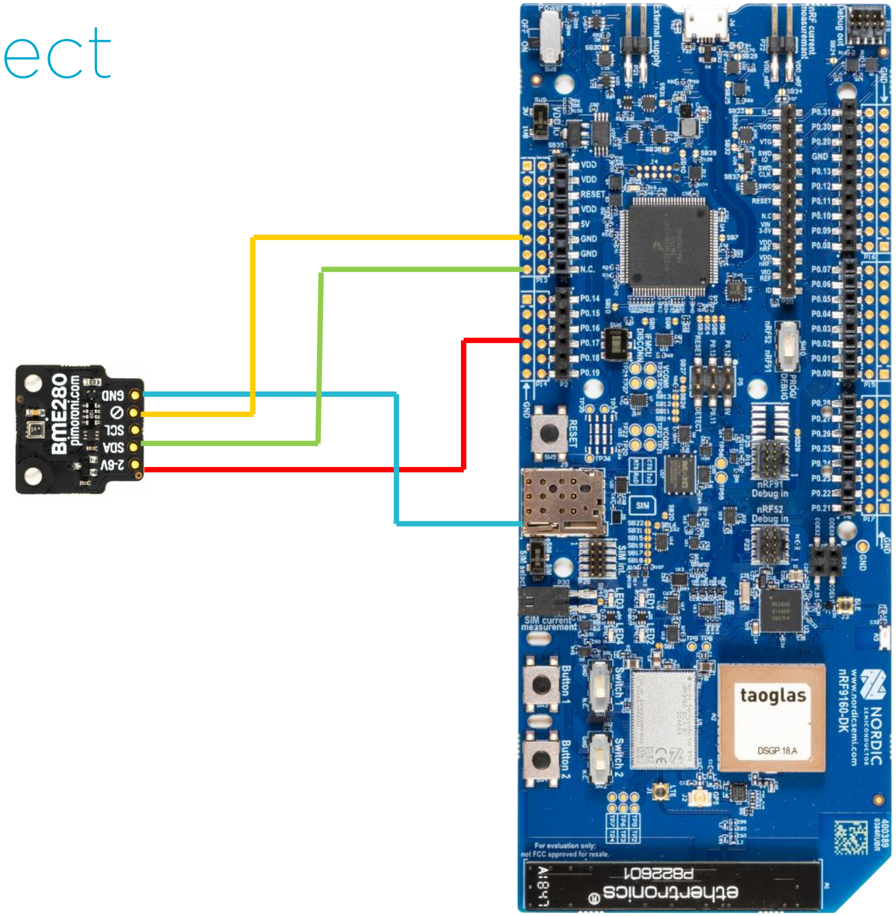
Hands-On: Our Project

Modem Firmware Update

1. Create Project from Scratch

2. Add Sensor

3. Send Data to UDP Server



Hands-On

Update Modem Firmware

Update Modem Firmware on nRF9160DK

Flashing the modem firmware:

1. Select and download the modem firmware version v1.3.2 from the download page.
   https://www.nordicsemi.com/Software-and-Tools/Development-Kits/nRF9160-DK/Download#infotabs
2. Open the **nRF Connect for Desktop** application and start the **Programmer**.
3. Click SELECT DEVICE in the upper left corner of the Programmer application and select the nRF9160DK.
4. Click **Add file** and select the previously downloaded modem firmware zip file.
5. Click **Write**.

The new modem firmware will now be flashed to the nRF9160 modem. This may take about 50 seconds. After flashing, press the Reset button on the board for the changes to take effect.

Hands-On

1. Create a Project from Scratch

github.com/ChrisKurz/AVNET_Training

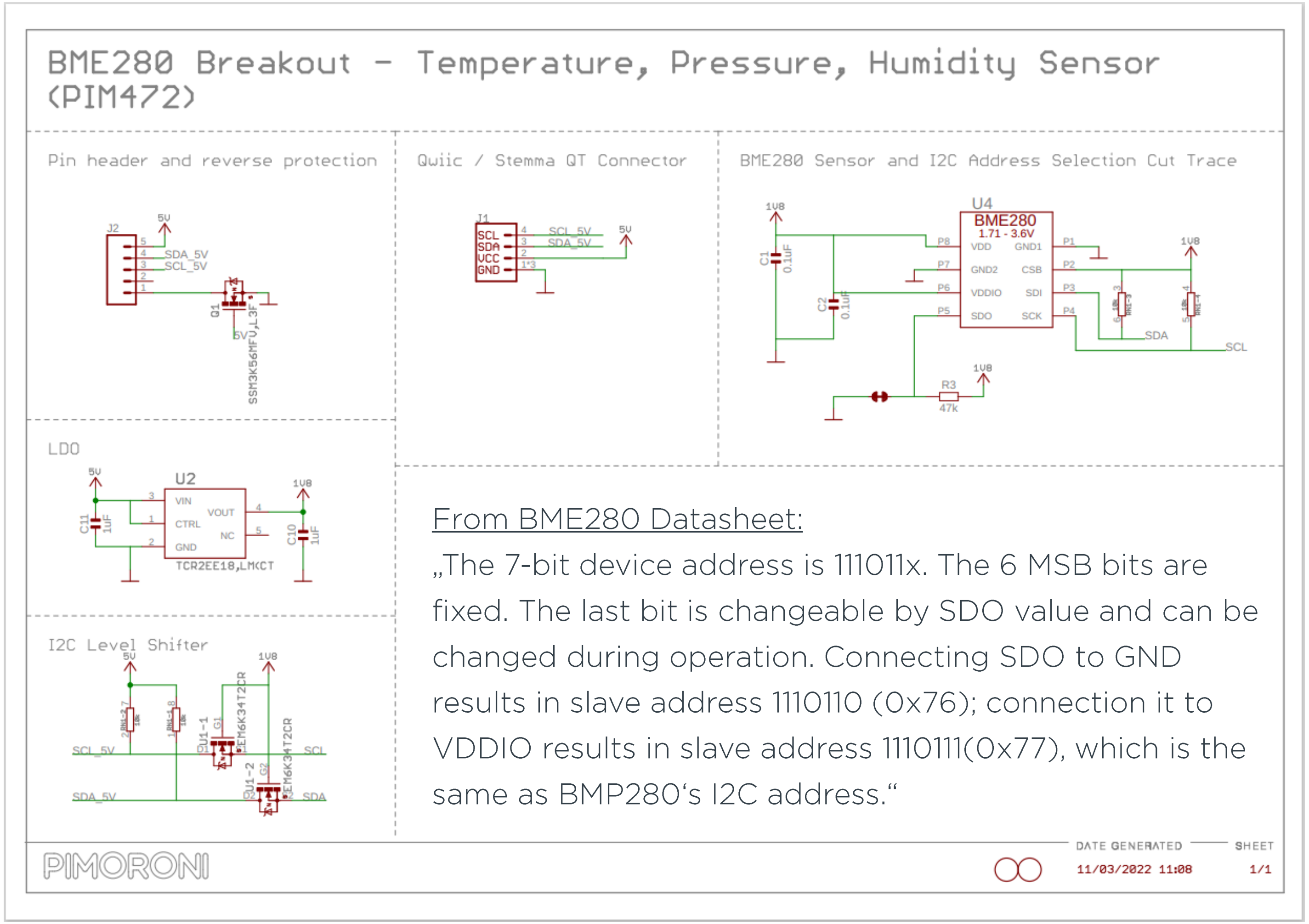
→ open file „HandsOn-1_Create a project from scratch.md“

Hands-On

2. Adding BME280 Sensor Driver

github.com/ChrisKurz/AVNET_Training

→ open file „HandsOn-2_Adding BME280 Sensor Driver.md“

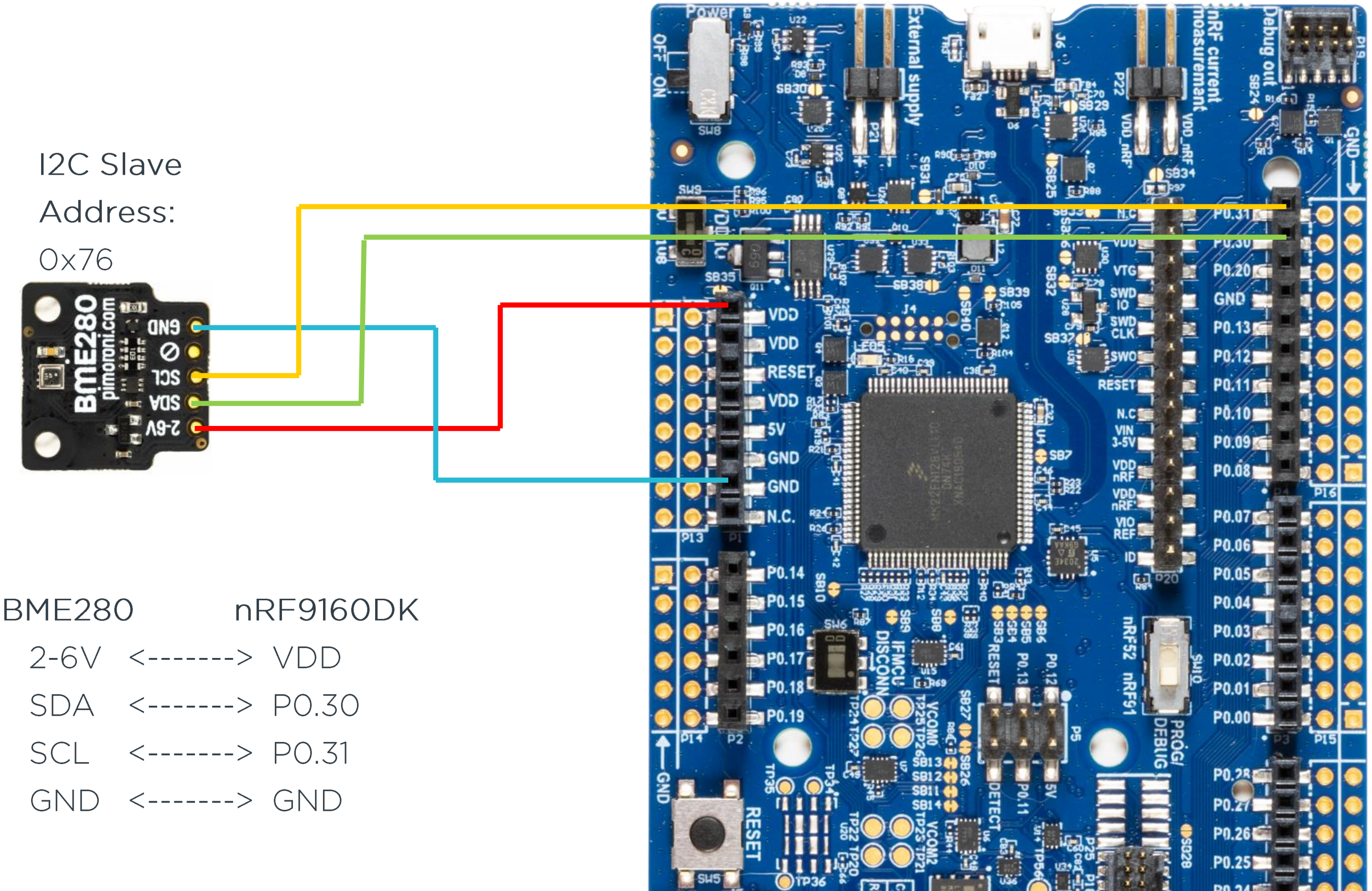


From BME280 Datasheet:

„The 7-bit device address is 111011x. The 6 MSB bits are fixed. The last bit is changeable by SDO value and can be changed during operation. Connecting SDO to GND results in slave address 1110110 (0x76); connection it to VDDIO results in slave address 1110111(0x77), which is the same as BMP280's I2C address."

nRF9160DK

I2C Slave Address: 0x76



BME280		nRF9160DK
2-6V	<------->	VDD
SDA	<------->	P0.30
SCL	<------->	P0.31
GND	<------->	GND

nRF Connect SDK fundamentals course

- Self-paced hands-on online course focusing on learning the essentials of nRF Connect SDK.
- Lots of hands-on exercises.
- Centralized up-to-date content.
- Protocol agnostic (cellular IoT, Bluetooth LE, Bluetooth mesh, etc..).
- Supports all our DKs and the Thingy:91.
- Ideal for new users of nRF Connect SDK or users switching from nRF5 SDK to nRF Connect SDK.
- Test your knowledge through interactive quizzes.

DevAcademy

What have we learned?

- *nRF Connect SDK* – One Code base for all Nordic Products
- Updating nRF9160 Modem Firmware
- Creating a Project from Scratch
- KCONFIG: Adding of Software Modules and configure the Software Modules
- DeviceTree: Abstraction of project allows Hardware Configuration in separate file

Thank you