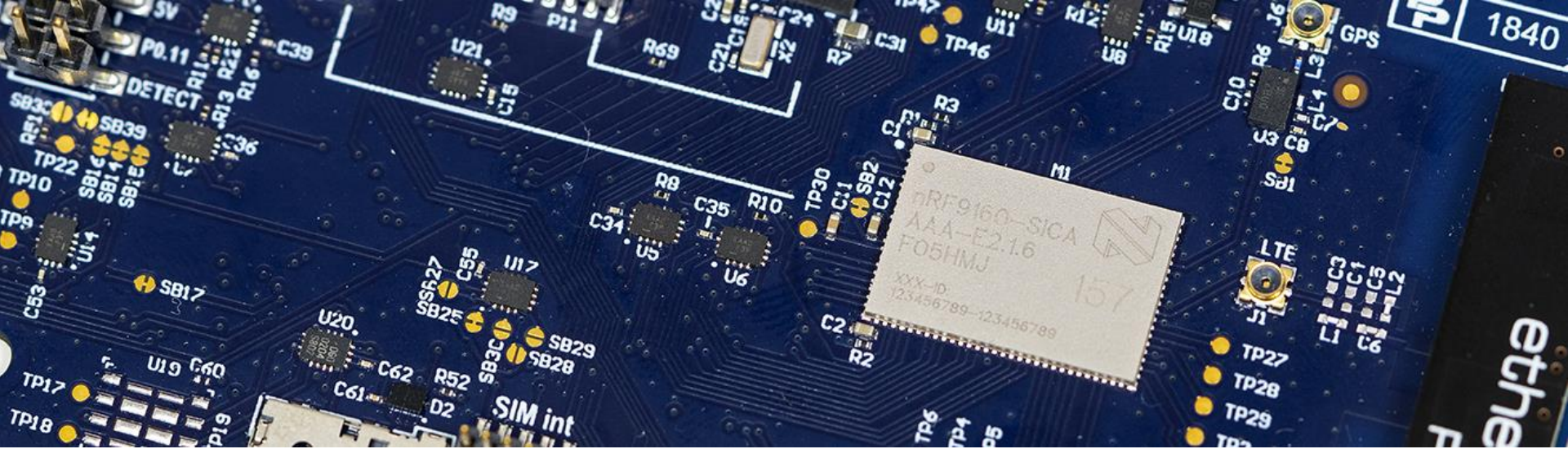



# Introduction to nRF Connect SDK

Cellular IoT made easy

*Webinar*

*Nordic Semiconductor*

*September 2019*

*Duration: approx 45 mins*

# Today's host

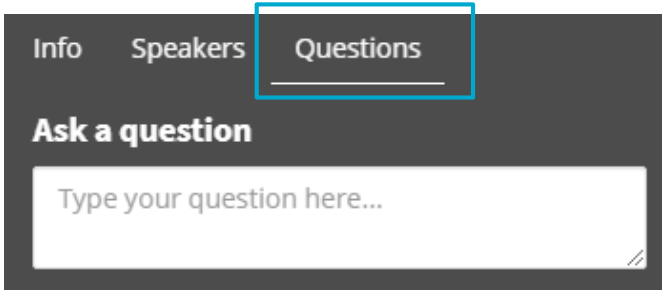
Bjørn Kvaale

Product Marketing Engineer

- Master of Science degree in Engineering Cybernetics & Robotics from the Norwegian University of Science & Technology
- Two years of experience working in the tech support group at Nordic
  - Focusing on SW related support cases, with a major focus on Bluetooth mesh & the nRF91 Series.

# This webinar

- Duration: 45 mins
- Questions are encouraged!
  Please type your question in the Questions tab on the right sidebar.
- All questions are anonymously posted.
- We will do our best to answer all questions at the end of the webinar
- We will share a Q&A after the webinar with updated answers if necessary



# Content

- What is nRF Connect SDK
- RTOS Intro
- Setting Up & Configuring nRF Connect SDK
- Overview of what is included in nRF Connect Platform

# What is nRF Connect SDK?

- Software development kit for the nRF9160
- Integrates the Zephyr RTOS
- Publicly hosted on GitHub
  - version control management with Git
- Free SEGGER Embedded Studio support
- Support for Desktop & Cloud apps

Nordic Application Frameworks
& Reference Designs

Zephyr kernel & middleware

IP protocols CoAP, HTTP, MQTT, LWM2M

System

Memory mng.
DFU
File systems

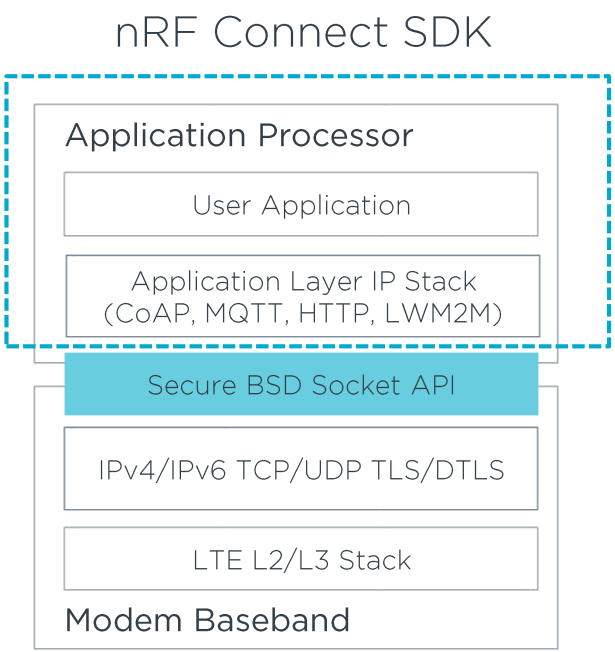
Nordic Low Layer FW

Root of Trust bootloader & secure domain arch.

MAC layers: BSD, Bluetooth LE, 802.15.4, Zigbee,Thread

Drivers inc. NFC, Crypto, SPI, UART, USB

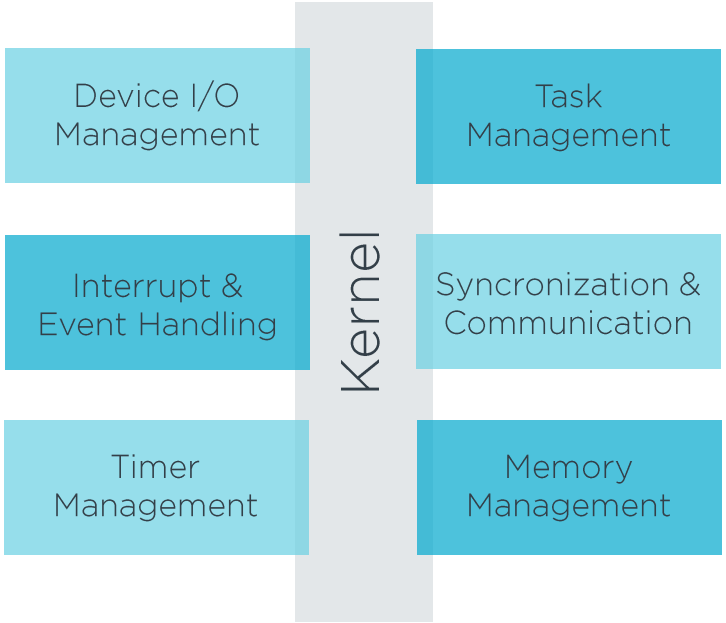
# Software architecture




- Complete and easy to use solution
- Connectivity and application software
- Support for OTA Modem and Application updates
- Reference design with multiple sensors
- Support for “thin modem” operation

# 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 open source RTOS?

Trend towards open RTOS for MCUs in IoT

- Greater number of open standards in applications
  - Vendor implementations not necessarily better, just different
- Free use & distribution
  - The price is right
- Security & stability for IoT devices
  - Proven success: open review, open information sharing
- 3rd-party expertise develop around open source

# Why the Zephyr RTOS?

- Zephyr designed & built for low power wireless
- Zephyr is independent
  - Linux Foundation Project
- Zephyr is scalable
  - Very small configurations for memory constrained devices (< 8 kB)
  - Powerful, feature-rich, configurations for large memory, high-processing power devices
- Zephyr includes popular IP standards
  - Nordic can focus on other important features

Zephyr™ Project

# Zephyr Project Members

- Nordic a member & contributor since 2016
- We will continue to work with Zephyr
  - -> ensure best in class features & performance

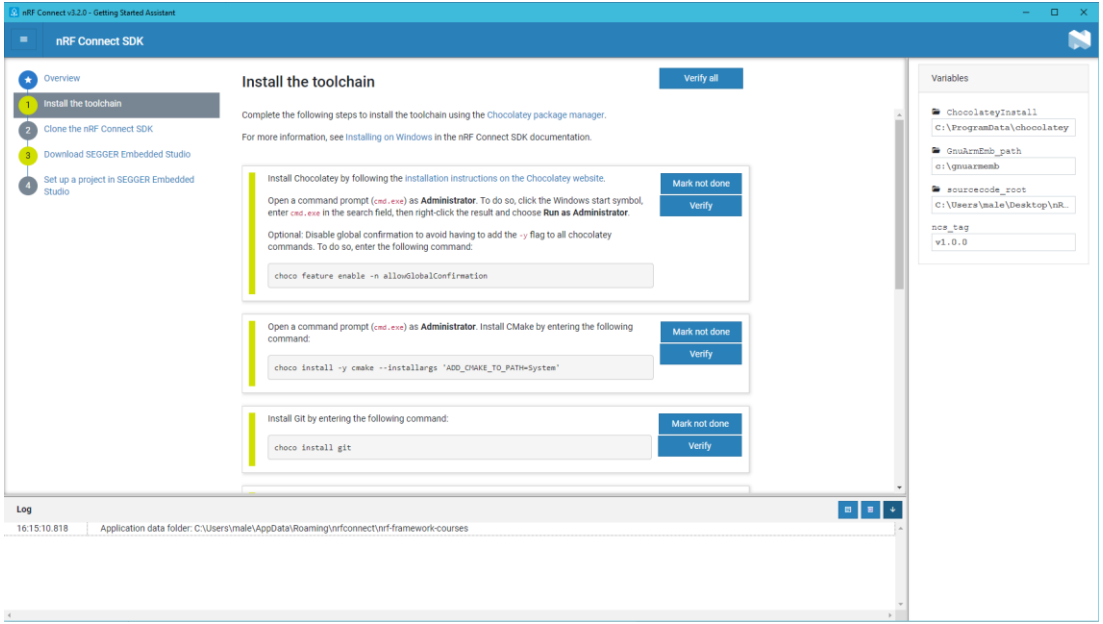
Platinum Members

Silver Members

# Getting and using nRF Connect SDK

- Hosted on GitHub, supported by git version control & west
  - West is a command line tool for programming & flashing, provides multiple repository management support
  - All code pre-integrated in one SDK
- Can fork nRF Connect SDK code & contribute to the SDK
- Allows syncing of nRF Connect SDK code for use in private repos
- Setup the toolchain & download the SDK version using nRF Connect for Desktop “Getting Started Assistant” app

# Getting and using nRF Connect SDK




- nRF Connect for Desktop
- nRF Getting Started Assistant app
- Platform dependent guides
- Windows, macOS, Linux

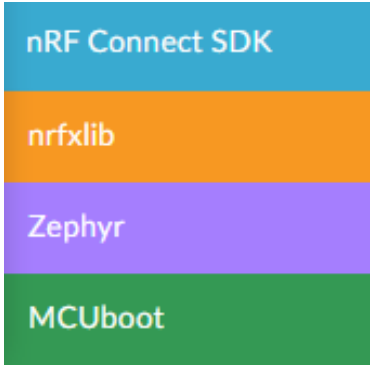
# Getting and using nRF Connect SDK

Configure, build & debug with Segger Embedded Studio

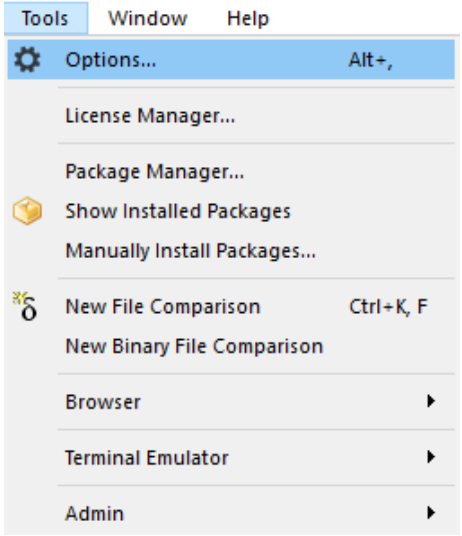
- Select which SDK features to auto-include
- Add your code
- Add configurations for your product PCB

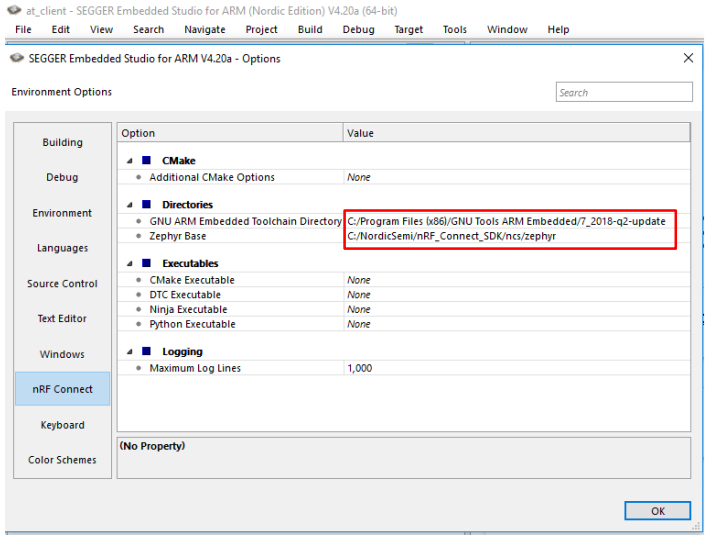
Four main repositories:

- nRF Connect SDK
- nrfxlib, Zephyr, MCUboot
  - Fork nRF repo into own branch
  - Update nRF repo using git
  - Update project repos using west

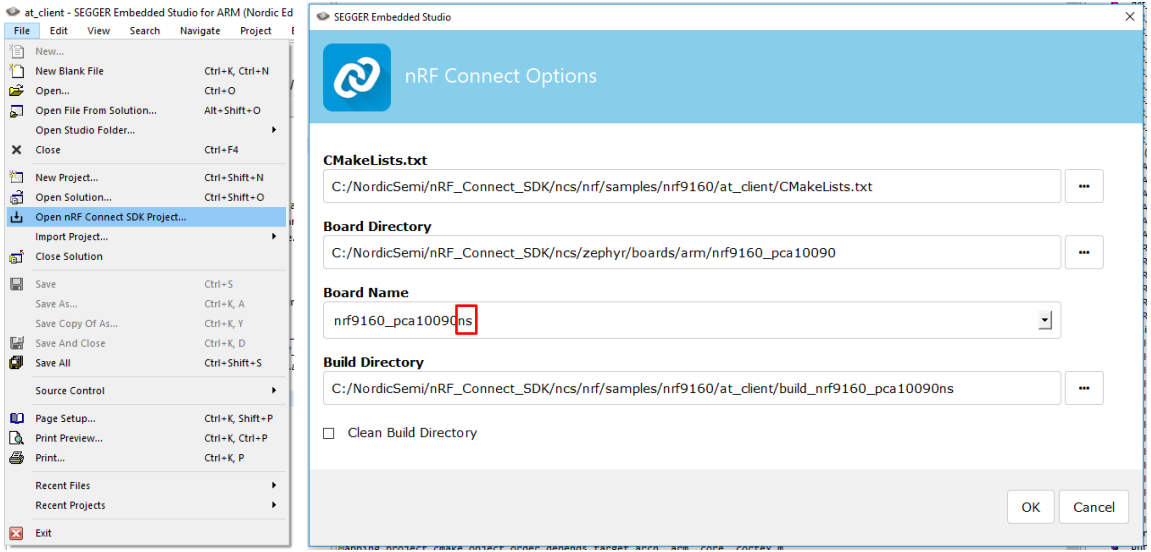
# Getting and using nRF Connect SDK



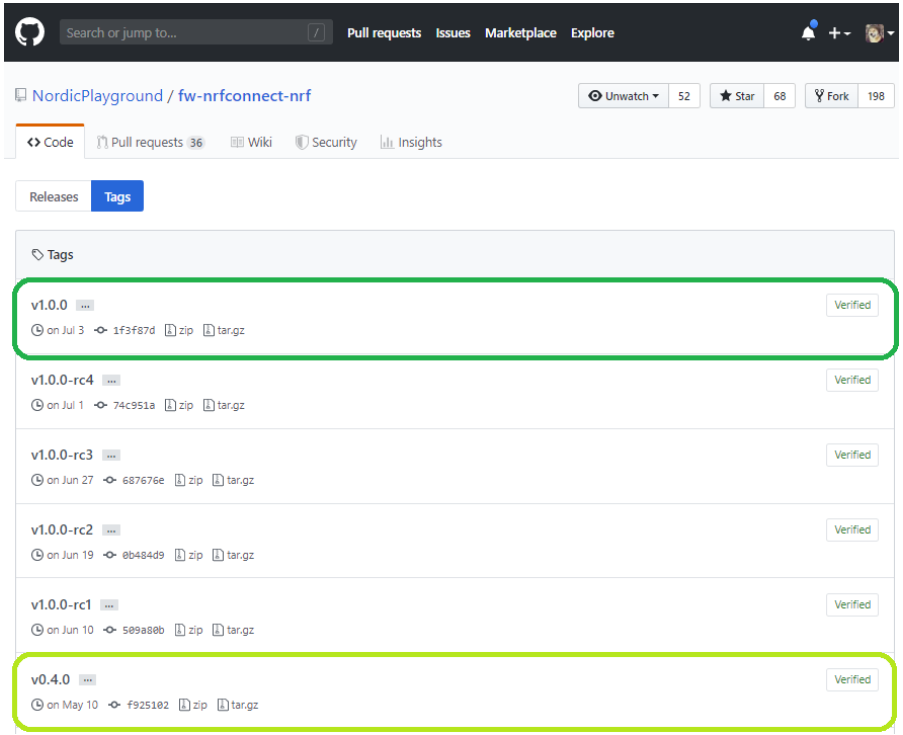

- Open Segger Embedded Studio
- Set GNU ARM Embedded Toolchain directory
- Use v7-2018-q2!
- Set Zephyr Base repository

# Getting and using nRF Connect SDK




- Setting up the asset tracker example in nRF Connect SDK
- Using the v1.0.0 tag
- Use the non-secure board name for application

# Getting and using nRF Connect SDK




- Start with a tagged release to ensure higher quality assurance
- https://github.com/NordicPlayground/fw-nrfconnect-nrf/tags
- Additional documentation
- https://developer.nordicsemi.com/nRF_Connect_SDK/doc/latest/nrf/getting_started.html

# Project Configuration

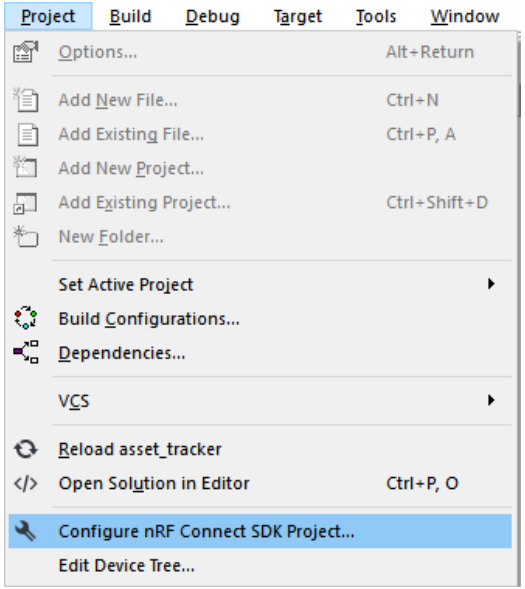
Current SES release: v4.20a

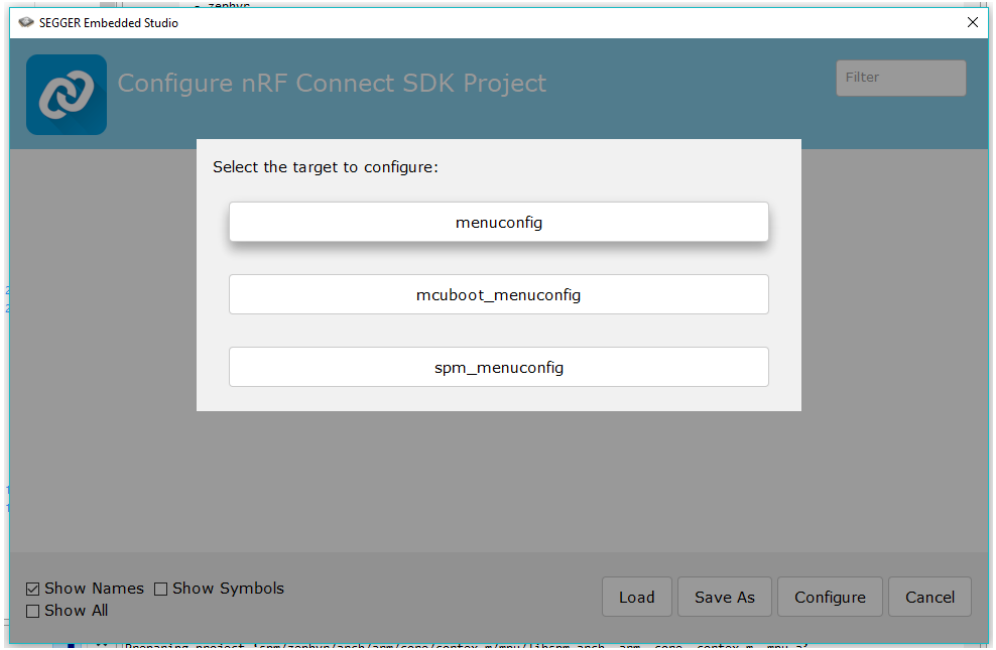
- Project configuration is controlled through Kconfig and DTS

- To configure project:
  Project -> ‘Configure nRF Connect SDK Project’ (menuconfig)

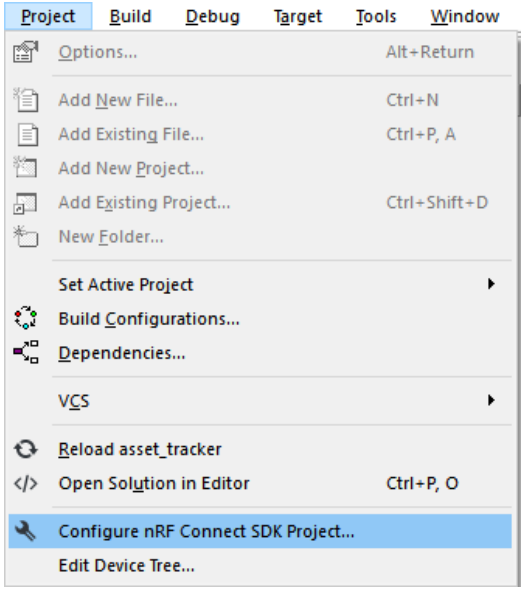
- To configure Device Tree:
  Project -> ‘Edit Device Tree’

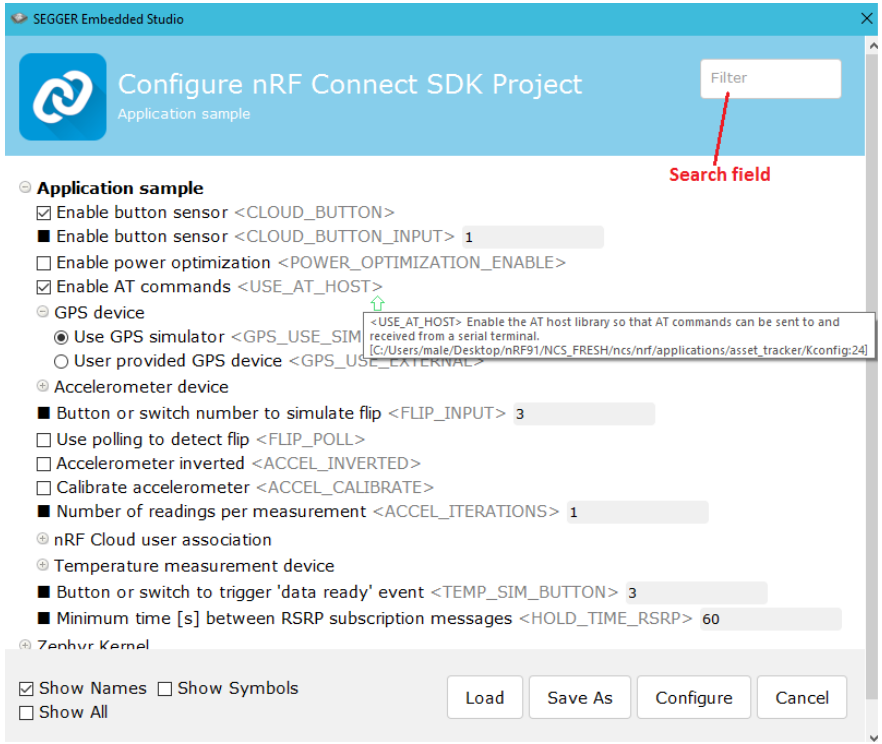
# Configuring an nRF Connect SDK project



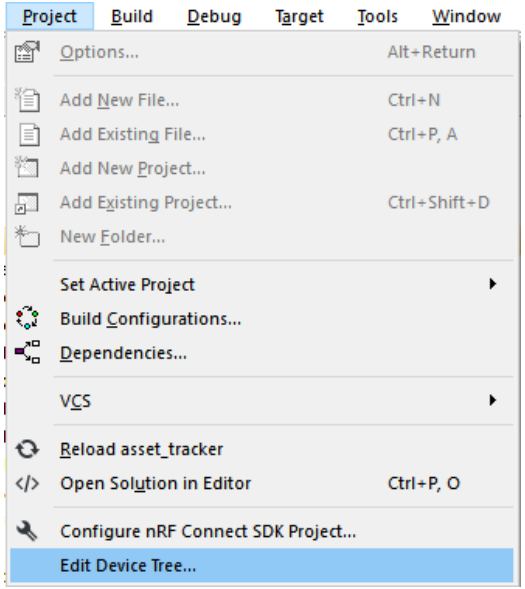



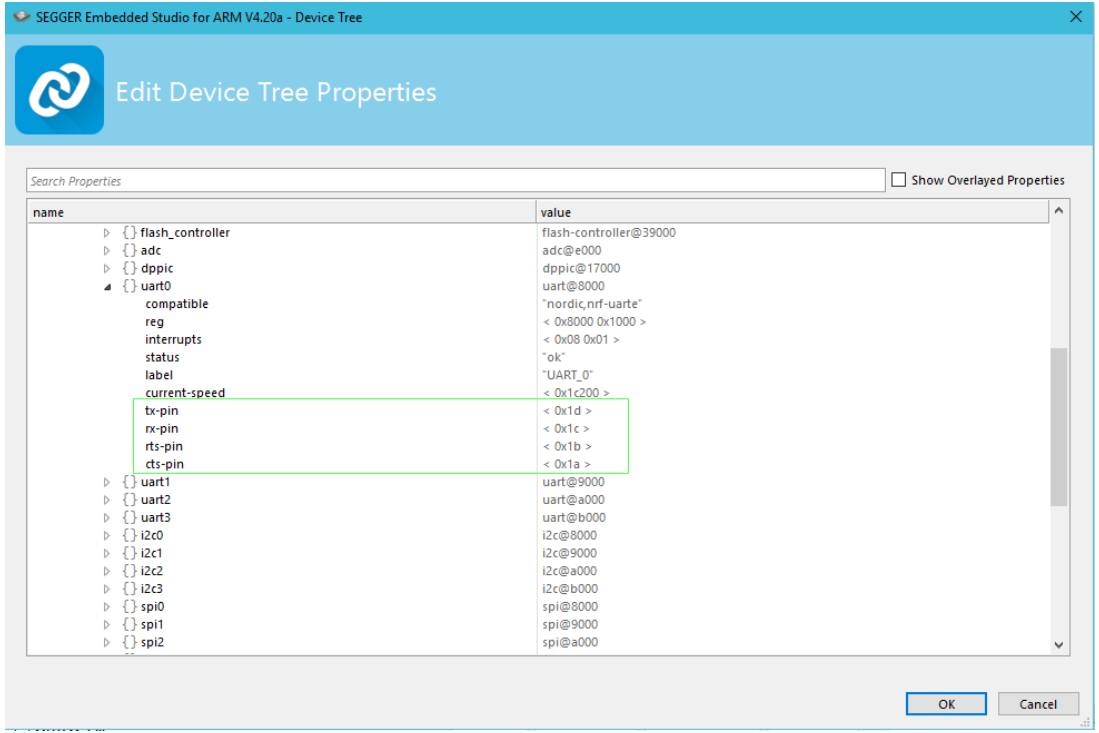
# Configuring an nRF Connect SDK project



# Configuring an nRF Connect SDK project



# nRF Connect family

## nRF Connect SDK

- Examples
- Application protocols
- RTOS
- Libraries
- Hardware drivers

## nRF Connect for Desktop

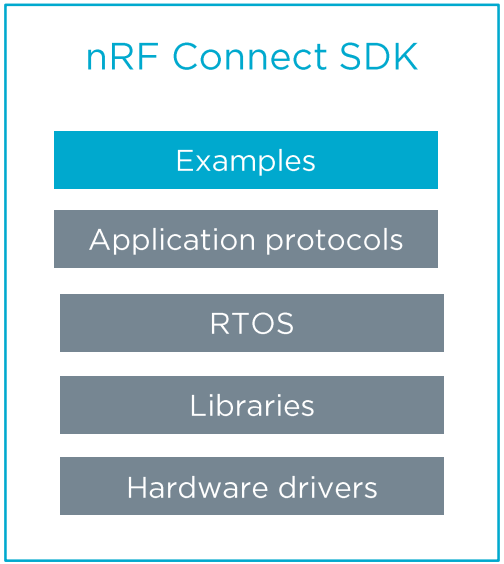
- Getting Started Assistant
- Programmer
- LTE Link Monitor
- Trace Collector

## nRF Connect for Cloud

- Examples
- Device management
- SIM management

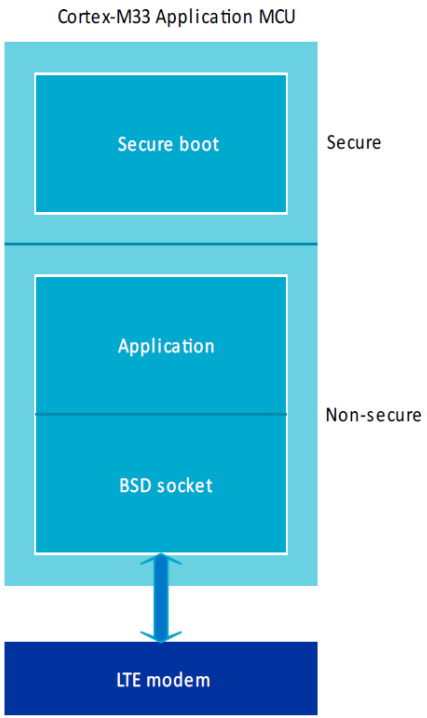
# nRF Connect SDK

- Asset tracker
- LTE sensor gateway
- AT client
- Simple MQTT
- nRF CoAP Client
- LwM2M Client
- HTTP Application Update
- AWS FOTA sample
- Secure Services Sample
- *Secure Partition Manager* (SPM)



# nRF9160: Secure Partition Manager

- Configures SPU & then jumps to application
  - SPU: system protection unit
  - No DFU
- **Secure:** CPU thinks the executed code is secure
  - Arm CryptoCell
  - Not restricted
- **Non-secure:** CPU thinks the executed code is NOT secure
  - Restricted
  - Modem only works in this mode



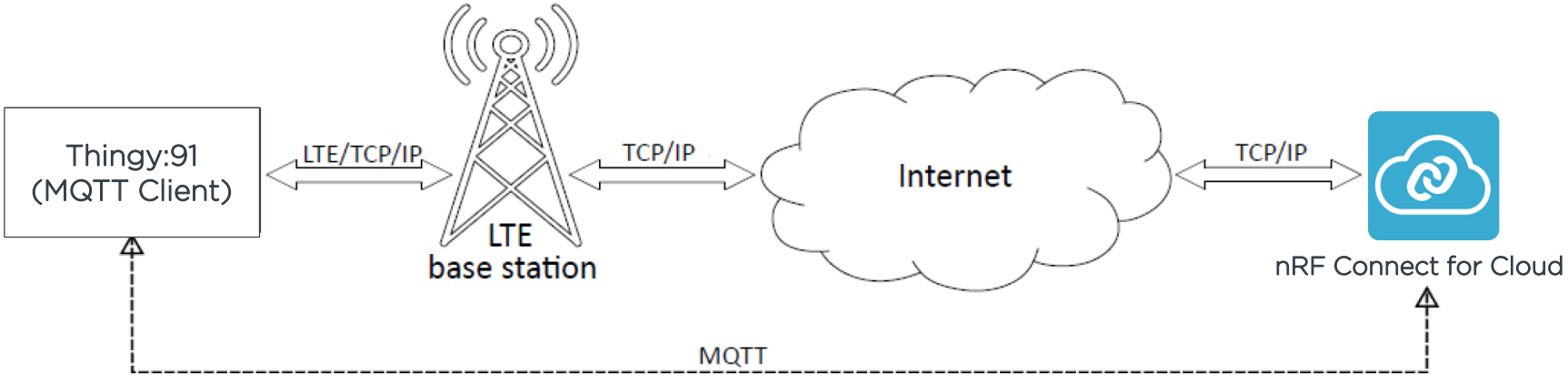
# nRF9160: Secure Partition Manager

- SPM sets security of all user peripherals & GPIOs to non-secure
- Peripherals need to be non-secure in order for application to use them
  - Application runs in non-secure domain

```
static void spm_config_peripherals(void)
{
        struct periph_cfg {
#ifndef CONFIG_SPM_BOOT_SILENTLY
                char *name;
#endif
                u8_t id;
                u8_t nonsecure;
        };

        /* - All user peripherals are allocated to the Non-Secure domain.
         * - All GPIOs are allocated to the Non-Secure domain.
         */
        static const struct periph_cfg periph[] = {
                PERIPH("NRF_P0", NRF_P0, CONFIG_SPM_NRF_P0_NS),
                PERIPH("NRF_CLOCK", NRF_CLOCK, CONFIG_SPM_NRF_CLOCK_NS),
                PERIPH("NRF_RTC1", NRF_RTC1, CONFIG_SPM_NRF_RTC1_NS),
                PERIPH("NRF_NVMC", NRF_NVMC, CONFIG_SPM_NRF_NVMC_NS),
                PERIPH("NRF_UARTE1", NRF_UARTE1, CONFIG_SPM_NRF_UARTE1_NS),
                PERIPH("NRF_UARTE2", NRF_UARTE2, CONFIG_SPM_NRF_UARTE2_NS),
                /* There is no DTS node for the peripherals below,
                 * so address them using nrfx macros directly.
                 */
                PERIPH("NRF_IPC", NRF_IPC_S, CONFIG_SPM_NRF_IPC_NS),
                PERIPH("NRF_VMC", NRF_VMC_S, CONFIG_SPM_NRF_VMC_NS),
                PERIPH("NRF_FPU", NRF_FPU_S, CONFIG_SPM_NRF_FPU_NS),
                PERIPH("NRF_EGU1", NRF_EGU1_S, CONFIG_SPM_NRF_EGU1_NS),
                PERIPH("NRF_EGU2", NRF_EGU2_S, CONFIG_SPM_NRF_EGU2_NS),
                PERIPH("NRF_TWIM2", NRF_TWIM2_S, CONFIG_SPM_NRF_TWIM2_NS),
                PERIPH("NRF_SPIM3", NRF_SPIM3_S, CONFIG_SPM_NRF_SPIM3_NS),
                PERIPH("NRF_TIMER0", NRF_TIMER0_S, CONFIG_SPM_NRF_TIMER0_NS),
                PERIPH("NRF_TIMER1", NRF_TIMER1_S, CONFIG_SPM_NRF_TIMER1_NS),
                PERIPH("NRF_TIMER2", NRF_TIMER2_S, CONFIG_SPM_NRF_TIMER2_NS),
                PERIPH("NRF_SAADC", NRF_SAADC_S, CONFIG_SPM_NRF_SAADC_NS),

                PERIPH("NRF_GPIOTE1", NRF_GPIOTE1_NS,
                                      CONFIG_SPM_NRF_GPIOTE1_NS),
        };

        PRINT("Peripheral\t\tDomain\t\tStatus\n");

        if (IS_ENABLED(CONFIG_SPM_NRF_P0_NS)) {
                /* Configure GPIO pins to be Non-Secure */
                NRF_SPU->GPIOPORT[0].PERM = 0;
        }

        for (size_t i = 0; i < ARRAY_SIZE(periph); i++) {
                int err;
```

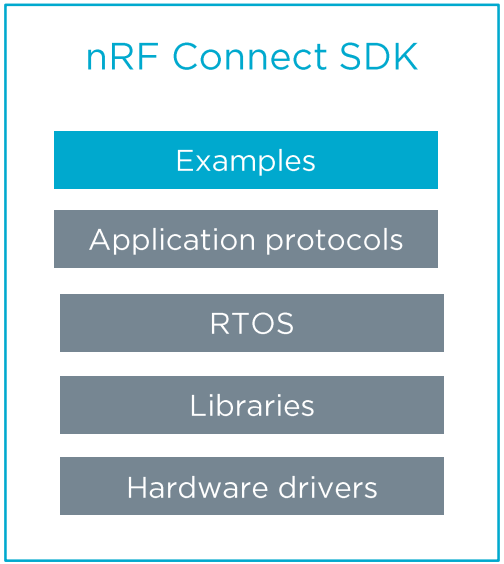
# nRF9160: Asset Tracker

- Sends real GPS coordinates, accelerometer, temperature, humidity & air pressure data to nRF Connect for Cloud
- Source code for connecting to server



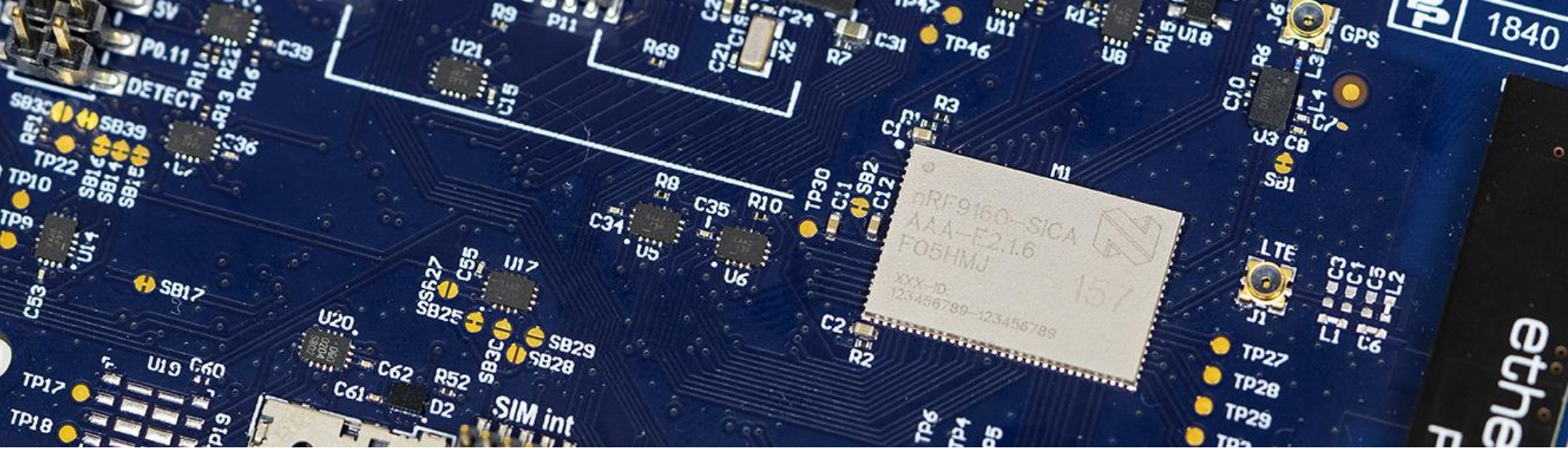
# nRF Connect SDK

- Application protocols
  - MQTT
  - CoAP
  - HTTP
  - LwM2M
- Zephyr RTOS
- Libraries
  - MCUboot secure bootloader
- Hardware drivers are available through nrfxlib
  - DPPI, GPIO, GPIOTE, I2S, PDM, SAADC, TIMER, UARTE, WDT, etc



# nRF Connect for Cloud

- Examples
- Asset tracker – displays GPS coordinates on map and switch state
- LTE sensor gateway – displays sensor data
- nRF9160 DK management
- SIM card activation




# Introduction to nRF Connect SDK

Cellular IoT made easy

*Webinar*

*Nordic Semiconductor*

*September 2019*

*Duration: approx 45 mins*

# Demo

# Q&A