Hes·so VALAIS WALLIS
Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Hes·so

Systems Engineering

Infotronics

Bachelor's thesis
Diploma 2022

Pompili Valérie

Performance Analysis of Nordic NRF under Zephyr

Professor
Rieder Medard

Expert
Mangold Stefan

Submission date of the report
cf. thesis form (19.08.2022)

HES-SO Valais

SYND	ETE	TEVI
X	X	X

Données du travail de diplôme
Aufgabenstellung der Bachelorarbeit

FO 1.2.02.07.EB
che/31/05/2021

Filière / Studiengang **SYND**	Année académique / *Studienjahr* **2021-22**	No TB / *Nr. BA* **IT/2022/79**
Mandant / *Auftraggeber* ☐ HES—SO Valais ☒ Industrie ☐ Etablissement partenaire *Partnerinstitution*	Etudiant / *Student* **Valérie Pompili** Professeur / *Dozent* **Medard Rieder**	Lieu d'exécution / *Ausführungsort* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*
Travail confidentiel / *vertrauliche Arbeit* ☐ oui / ja ☒ non / nein	Expert / *Experte* (*données complètes*) **Mangold Stefan** - stefan.mangold@ypsomed.com Ypsomed AG, Brunnmattstrasse 6, 3401 Burgdorf	

Titre / *Titel* **Performance Analysis of Nordic NRF under Zephyr**

Description / Goal:

This Bachelor thesis intents to establish a collection of sample projects that are useful to establish a performance profile of the NRF52xxx (maybe the NRF53xxx) Bluetooth device family. The thesis is embedded into a medical context given by Ypsomed AG. Another constraint is given by the use of the Zephyr RTOS.

Tasks:

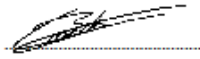
The following tasks must be solved for this thesis:

- Interrupt latency study: An IO pin is interrupted by a signal generator. The interruptions are transferred via Bluetooth to a host device, typically to a PC with a Nordic BTLE dongle.
- Interrupt priority study: Document and test the interrupt priority features under Zephyr / Nordic SDK.
- ADC performance study: An analog signal is sampled by the onboard ADC. The samples are transferred via Bluetooth to a host device.
- PWM Generation: Generation of a high frequency PWM signal to e.g. drive a motor driver.
- Bandwidth study: Measurement of the BTLE bandwidth under Zephyr.
- Extended Advertising study: Study and test of the "extended Advertising" feature of BT5 under Zephyr.
- FOTA / DFU study: Implement a sample FOTA / DFU scenario under Zephyr.

All these tasks are implemented under the NRF Connect environment using an NRF53xxx development kit. Each task must be entirely documented.

Deliverables:

- Any source code.
- Technical report
- Presentation slides

Signature ou visa / *Unterschrift oder Visum*	Délais / *Termine*
Responsable de l'orientation / *Leiter der Vertiefungsrichtung:*  ..... [1] Etudiant / Student : *Valérie Pompili*	Attribution du thème / *Ausgabe des Auftrags:* **16.05.2022** Présentation intermédiaire / *Zwischenpräsentation:* **20-21.06.2022** Remise du rapport final / *Abgabe des Schlussberichts:* **19.08.22, 12:00** Expositions / *Ausstellungen der Diplomarbeiten:* **24-26.08.2022** Défense orale / *Mündliche Verfechtung:* **Semaine/Woche 36 (5-9.09.2022)**

[1] *Par sa signature, l'étudiant-e s'engage à respecter strictement la directive DI.1.2.02.07 liée au travail de diplôme.*
Durch seine Unterschrift verpflichtet sich der/die Student/in, sich an die Richtlinie DI.1.2.02.07 der Diplomarbeit zu halten.

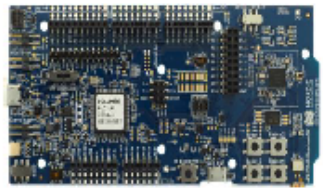
Rapport reçu le / *Schlussbericht erhalten am* ..... Visa du secrétariat / *Visum des Sekretariats* .....

Resume

Hes·so VALAIS WALLIS

Haute Ecole d'Ingénierie
Hochschule für Ingenieurwissenschaften

Zephyr™ NORDIC SEMICONDUCTOR Bluetooth 5.0



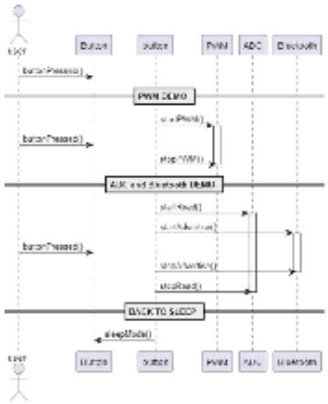
Edition 2022 |

Study Branch
Systems Engineering

Area of Application
Infotronics

Professor in Charge
Medard Rieder
medard.rieder@hes-so.ch

Partner
Yposmed AG



Performance Analysis of the Nordic nRF Hardware and Software under Zephyr RTOS

Graduate Valérie Pompili

Aim of the project

Ypsomed, "Ipso" means "self" and "med" stands for medication, is a Swiss medical technology company in Burgdorf. They develop products to allow patients to manage their own medication in a safe and simple way. Their main sector of activity concerns patients with diabetes. In order to include Nordic device in their product, Ypsomed wants to let realize a performance analysis of their nRF hardware and software under the Zephyr RTOS.

This bachelor's thesis provides a collection of sample projects that are useful to establish a performance profile of the nRF52xxx Bluetooth device family in a medical context, using nRF Connect SDK with Zephyr RTOS.

Methods | Experience | Results

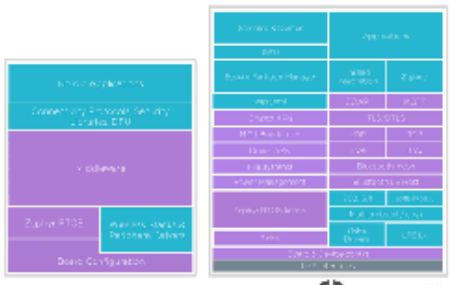
The first step was the parameterization of the development environment with the different software layers. From installing the nRF Connect SDK to how to create and implement projects with the management tools like West and the Microsoft Visual Code Studio IDE.

The second step was a real time performance analysis with a latency interrupt study. A response time of 21us has been determinate using an Analog Discovery 2 analyzer and specific calculations.

The third major step was the creation of a collection of sample project. First of all, a main example demonstrating how to implement a Bluetooth communication between 2 nRF52840 has been developed. The BTLE advertising has been prioritized in this example since Ypsomed wants to use this feature in its products.

Then, other useful examples showing how to use analog to digital conversion and generating pulse width modulated signals have been developed. These features will be used by Ypsomed to precisely control the functioning of their future products.

And finally, the last step was the development of an application that gathers different use cases in a medical context.



HES-SO Valais-Wallis • rue de l'Industrie 23 • 1950 Sion
+41 58 606 85 23 • hei@hevs.ch • www.hevs.ch/hei

swissuniversities

Abbreviations

BLE	Bluetooth Low Energy
DK	Development Kit
IoT	Internet of Things
RTOS	Real-Time Operating Systems
SDK	Software Development Kit
SoC	System on a Chip
TB	Bachelor’s Thesis

Abstract

The YpsoMate autoinjector is an automated disposable injection device created by Ypsomed. “Ipso” means “self” and “med” stands for medication. The aim of this Swiss medical technology company is to allow patients to administer their own medication in a safe and simple way. Their main sector of activity concern patients with diabetes.

With YpsoMate they seek to create a product that is as easy to use as possible. The use consists of two steps: Remove cap and inject. The second point of interest of Ypsomed is to create a product that is as ecological and environmentally friendly as possible. That is why, with Ypsomate, they have created a reusable injector. After using the product, the user only needs to change the syringe and the drug container.

In an effort to simplify patients' lives and avoid appointments and lost hours with medical staff, they included an electronics system capable of alerting them to the use of the autoinjector.

After Nordic's announcement to refocus their production directly in Norway, Ypsomed is very interested in working with their product and especially with the nRF5x product range.

The goal of this project is to study the new software environment of Nordic that includes Zephyr RTOS on the nRF52840 Development Kit (DK) to integrate it into their product. At the beginning of this project, a list of the different tasks to be carried out has been established. Each task will be entirely documented and a collection of samples projects will be established.

This Thesis is embedded into a medical context given by Ypsomed AG.

Table of Contents

Abbreviations 2
1 Introduction 8
1.1 Aim of Study 8
1.1.1 Introduction to TB 8
1.1.2 Description of the project 8
1.1.3 Objectives of the project 8
2 Nordic SoCs 9
2.1 State of the Art 9
2.1.1 Family of nRF Series SoCs 9
2.1.2 nRF5 SDK vs nRF Connect SDK 9
2.2 Materials – nRF52840 Development Kit 10
2.2.2 Hardware description 10
2.2.3 Specification 11
3 Setup of the Development Environment 12
3.1 Structure 12
3.2 Tools and configuration 14
3.3 Installation of the environment 15
3.4 About Workspace 16
3.5 Creation of a project with Workspace 18
3.6 Open an existing project using west 21
4 About Zephyr 22
4.1 What is an RTOS? 22
4.2 Threads 23
4.3 Create a Thread 23
5 Performance's analysis of the nRF52840 Nordic device under Zephyr 24
5.1 The interest of study latency and response time 24
5.2 The Cortex-M4 25
5.3 Set up and protocol 26
5.4 Test and Results 27
5.4.2 Toggle an output pin 27
5.4.3 Input to Output 29
5.4.4 Response time 30
6 Bluetooth communication on nRF52 under Zephyr 31
6.1 Introduction of Bluetooth Low Energy 31
6.1.2 GAP Role 32
6.1.3 Bluetooth LE Advertising 33

6.2 Bluetooth sample 34
6.2.2 Overview 34
6.2.3 Implementation of Broadcaster 34
6.2.4 Implementation of Observer 36
7 Samples 38
7.1 Use cases 38
7.2 SAADC 38
7.3 ADC with nfrx drivers 39
7.3.2 Overview 39
7.3.3 Implementation 39
7.4 ADC with Zephyr 41
7.4.2 Overview 41
7.4.3 Implementation 41
7.5 PWM 43
7.6 PWM sample 45
7.6.2 Overview 45
7.6.3 Implementation 45
8 Demonstration 46
9 Conclusion 47
Annexe 48
Annexe A: BT Specification V1 48
Annexe B: BT Specification V2 49
List of Figures 50
List of Tables 51

1 Introduction

1.1 Aim of Study

1.1.1 Introduction to TB

The Bachelor's Thesis (TB) is realised in collaboration with HES-SO Valais-Wallis and Ypsomed AG. All the project is realised at the HES-SO in Sion.
The work is supervised by Professor Medard Rieder.

1.1.2 Description of the project

This TB intents to establish a collection of samples projects that are useful to establish a performance profile of the nRF52xxx Bluetooth device family. All the tasks realised are implemented under the nRF Connect environment SDK. Another constraint is the use of the Zephyr RTOS. Zephyr is a recent open source RTOS. It is mainly used in embedded systems due to its low power consumption, stability and efficiency. It includes a scheduler that ensures predictable/deterministic execution patterns and abstracts out the timing requirements. The main element of this project will be the Bluetooth Low Energy (BLE) stack offered by Nordic. It will be important to study the impact of BLE communication on single core SoC's.

1.1.3 Objectives of the project

This project aims to cover the following goals:

1. Study of the nRF Family
2. Deployment of the development environment
3. Study the interrupt latency
4. Study Interrupt Priority
5. Study ADC performance
6. Study PWM Generation

All these tasks will be studied running in parallel with Bluetooth Low Energy in order to see how to prevent several tasks from being interrupted due to Bluetooth communication. For more details, please refer to the specifications in the annex. Be aware that there is two version of the specifications after discussion and modification of the points to be studied with the company.

2 Nordic SoCs

2.1 State of the Art

2.1.1 Family of nRF Series SoCs

Nordic Semiconductor is a world-leading specialist in ultra-low power wireless connectivity. Their technology enables millions of IoT devices to connect around the world. Everything changed in the summer of 2012 when Nordic launch its nRF51 Series of SoC on the market. It was a revolution in the wireless world.

"*For the first time, these chips uniquely separated the wireless protocol stack from the application software and included a 32-bit Arm Cortex-M based processor and enough embedded memory to run Bluetooth LE and proprietary 2.4GHz or ANT wireless technologies, enabling connected IoT products from a single chip.*[1]"

Then followed the nRF52 series in 2015. In this family, the nRF52840 and nRF52833 are particularly popular among users. Another big improvement was made in 2018 with the release of the nRF91 series. It included new tools as West and nRF Connect SDK. Finally, the latest product is the nRF5340. This is the first SoC in the world that possess a dual-core (ARM Cortex-M33).

2.1.2 nRF5 SDK vs nRF Connect SDK

In order to easily develop applications with their device, Nordic provides Software Development Kit which contains tools, libraries and a set of samples to help the developers. There are two of them: nRF5 SDK and nRF Connect SDK.

The nRF5 SDK is a standalone software used since 2012. The Bluetooth LE stack used is called SoftDevice and the source code for the library is confidential. The nRF5 SDK is a bare-metal SDK, which means that programming is based on sequential programming and does not depend on an RTOS. However. It is possible to use FreeRTOS to have some real-time operating system.

With the new generation of devices and due to the demand from product makers, it was logical to adopt an RTOS natively. Project member of Zephyr since 2016, Nordic provided the nRF Connect SDK fully RTOS-based using Zephyr RTOS. Independent, scalable and stable, Zephyr RTOS is open source that provides security and stability and build for low power wireless.

The nRF Connect SDK is the main SDK of Nordic. NRF5 SDK is now in maintenance mode. Any features beyond BLE 5 will not be added to the nRF5 SDK. Nordic requires its users and customers to use the nRF Connect SDK and Zephyr.

For more information, you can refer the following article.

[1] https://www.chipestimate.com/Nordic-Semiconductor-to-ship-its-billionth-Arm-1601622000/Nordic-Semiconductor/news/52681

2.2 Materials – nRF52840 Development Kit

In agreement with the company, all samples were developed with an nRF52840. You can find following description on Zephyr website.

2.2.2 Hardware description

For this bachelor thesis, the nRF52840 DK (PCA 10056) board was used.

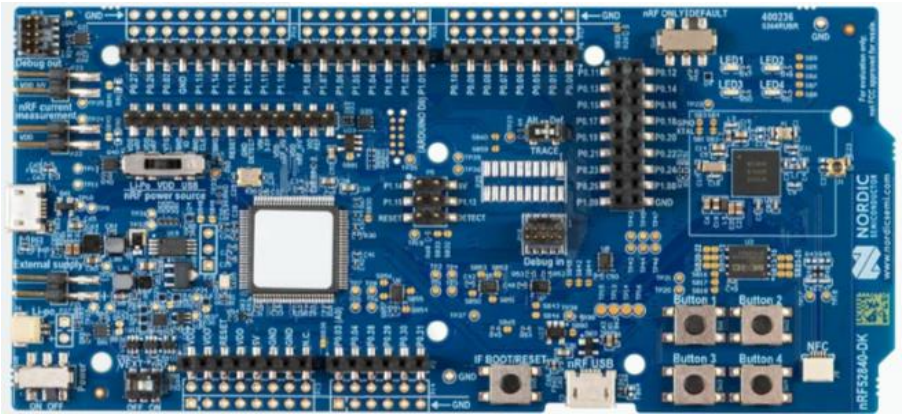


Figure 1: nRF5280 DK board

This board include a 32-bit ARM Cortex-M4 CPU with FPU running at 64 MHz. It has two external oscillators:

1. Slow clock: 32.768 kHz
2. Main Clock: 32 MHz

For connections and IOs, here are the PINs dedicated to GPIOs:

LED 1	P0.13
LED 2	P0.14
LED 3	P0.15
LED 4	P0.16

Button 1	P0.11
Button 2	P0.12
Button 3	P0.24
Button 4	P0.25

Tableau 1: Pin configuration for GPIO

2.2.3 Specification

The nRF52840 SoC provide Bluetooth 5.3 supporting Bluetooth Low Energy, Bluetooth Direction Finding, Bluetooth mesh, NFC, Thread and Zigbee. This SoC has the following specifications:

Flash	**1 MB**
RAM	256 KB
CPU	64 MHz ARM Cortex M4 with FPU
Supply voltage range	1.7 to 5.5 V from battery, external or USB
Hardware security	128-bit AES CCM
Hardware Security	Arm CryptoCell 310
Oscillators	64 MHz crystal oscillator (HFXO) controlled by a 32 MHz external crystal.

Tableau 2: nRF52840 DK specifications

3 Setup of the Development Environment

3.1 Structure

The nRF Connect SDK provides one code base and toolchain to develop application for nRF91, nRF53 and nRF52 Series. This code is organized in 4 main repositories:

- `Sdk-nrf` Contains application & Connectivity Protocols.
- `Sdk-nrfxlib` Contains Peripheral Drivers and Stacks
- `Sdk-Zephyr` Contains RTOS & Board Configuration
- `Sdk-MCUBoot` Contains an open-source Bootloader.

This code is an interesting combination between software developed by Nordic and open-source projects by Zephyr. The figure below shows the different layers available and the combination of Nordic and Zephyr code. In blue we find the code provided by Nordic Semiconductor and in purple the code provided by Zephyr. This allowed a wide range of wireless technologies and applications to be supported by one integrated code base.

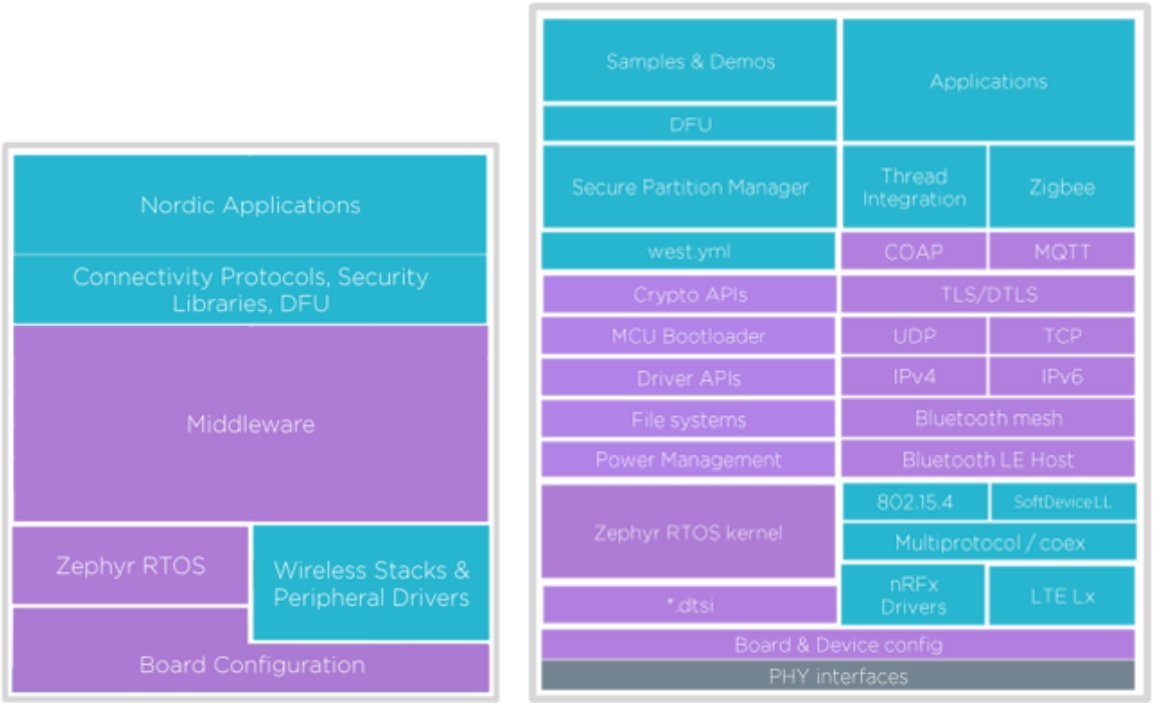


Figure 2: Layers of nRF Connect SDK

Nordic synchronizes with external repositories to always have the latest version of open source of Zephyr and MCU. You can see with the next figure the external repositories that would include Zephyr and MCU boot. As customer or developer, you don't need to worry about the open-source. Nordic also runs integration tests on all source and manages configuration.

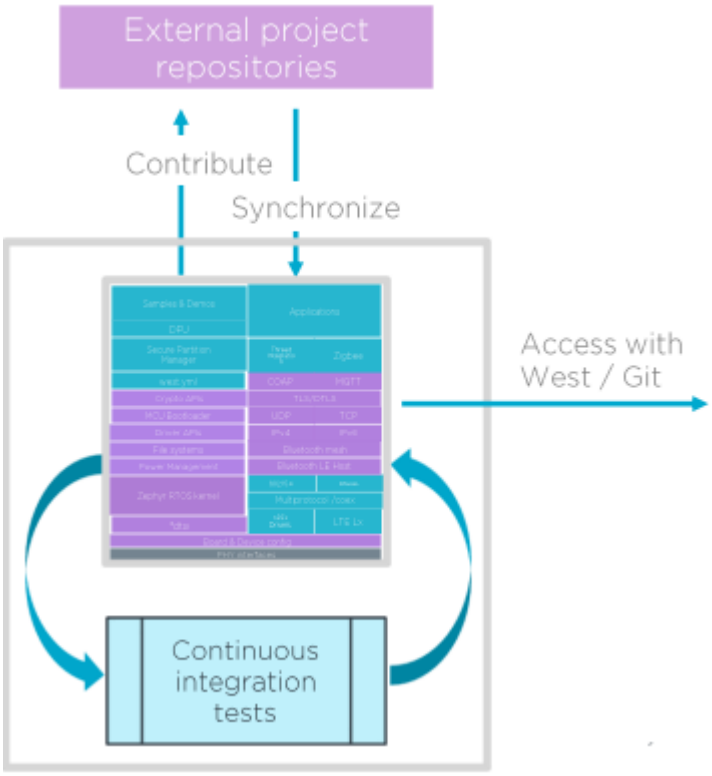


Figure 3: How code base is synchronizes

3.2 Tools and configuration

The figure illustrates the toolchain of nRF Connect SDK and which software are used to deliver a software product:

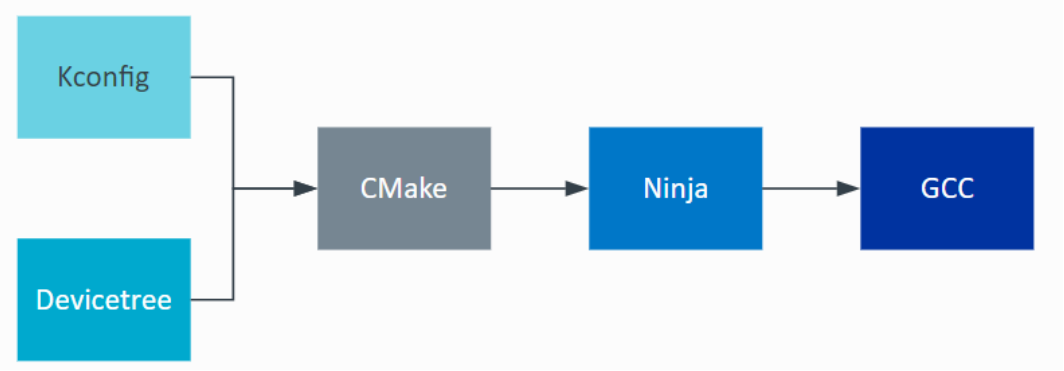


Figure 4: nRF Connect SDK tools and configuration methods

- ***Kconfig*** Generates definitions that configure libraries and subsystems.
- ***Devicetree*** Describe the hardware and pin layout. It is build for different PCB design without changing source code.
- ***CMake*** Use the information to build files.
- ***Ninja*** Use file from CMake to build the program.
- ***GCC*** Create the executable (hex file).

To manage source code and configurations, there are three steps:
First we have West which is a multi-repository management tool. It is used to manage workspace and make sure to have the good version of the 4 main repository.
The code base is generated from nRF Connect SDK and your code.
Then Kconfig sets up the feature configuration for compile.
Finally there is Device Tree which sets up the target board.

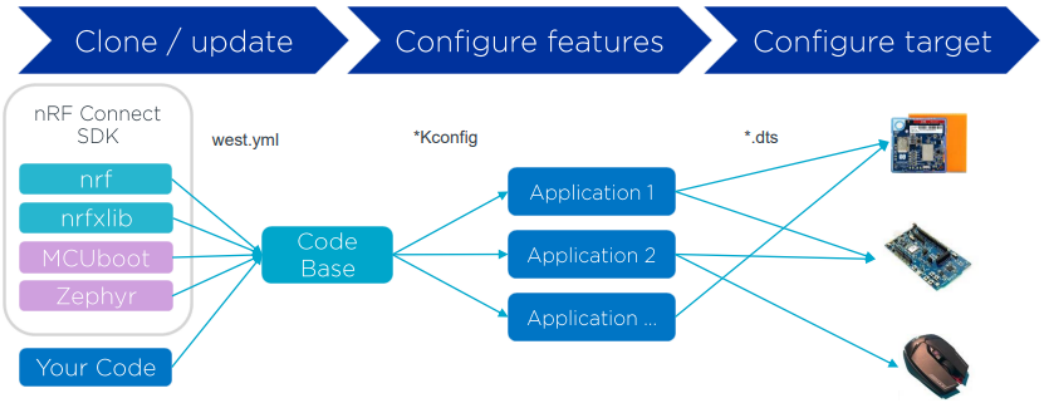


Figure 5: Manage source code and configuration

3.3 Installation of the environment

The set-up guide is available directly on the web site of NORDIC Semiconductor on the “Getting started” section.

The nRF Connect SDK installation procedure is:

1. Install nRF Command Line Tools:
   The nRF Command Line Tools is used for development, programming and debugging of Nordic Semiconductor's nRF51, nRF52, nRF53 and nRF91 Series devices.

https://www.nordicsemi.com/Products/Development-tools/nrf-command-line-tools/download

2. Install nRF connect for Desktop:

https://www.nordicsemi.com/Products/Development-tools/nrf-connect-for-desktop/download

3. Open nRF Connect for Desktop
   3.1. Install Toolchain Manager
   3.2. After the installation, click on the button “Open”

4. Install the latest nRF Connect SDK version

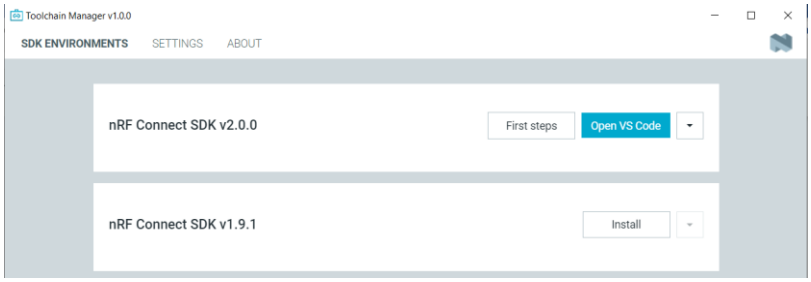


Figure 6: Toolchain Manager, select nRF Version

5. Install VS Code:

https://code.visualstudio.com/download

3.4 About Workspace

Developing applications is bound to make you work with workspace. A workspace is a collection of one or more files. Most of the time your application is contained in a single folder open as the workspace but you can include more folders with other applications and then have a multi-root workspace. Having a multi-root workspace allowed you to configure settings that apply to all folders. With VS code there is a JSON file <name>.code-workspace that regroups the folder of the workspace.

hello_world.code-workspace

```
{
  "folders": [
    {
      "name": "hello_world",
      "path": ".."
    },
    {
      "path": "..\\..\\..\\TB_Broadcaster_V1"
    },
    {
      "name": "pwm_sample",
      "path": "..\\..\\..\\pwm_sample"
    },
    {
      "name": "TB_Broadcaster",
      "path": "..\\..\\..\\TB_Broadcaster"
    },
    {
      "name": "TB_CentralV0",
      "path": "..\\..\\..\\TB_CentralV0"
    },
```

Figure 7: Contain of .code-worspace file

With nRF we are talking about west workspace, not to be confused with workspace from VS Code. West is a meta-tool used by Zephyr that provides a multiple repository management system. "West will clone a separate instance of nRF Connect SDK to its workspace and use it for the workspace application."[2] When creating a project with west workspace, you will find the four repositories presented with the structure chapter 12.

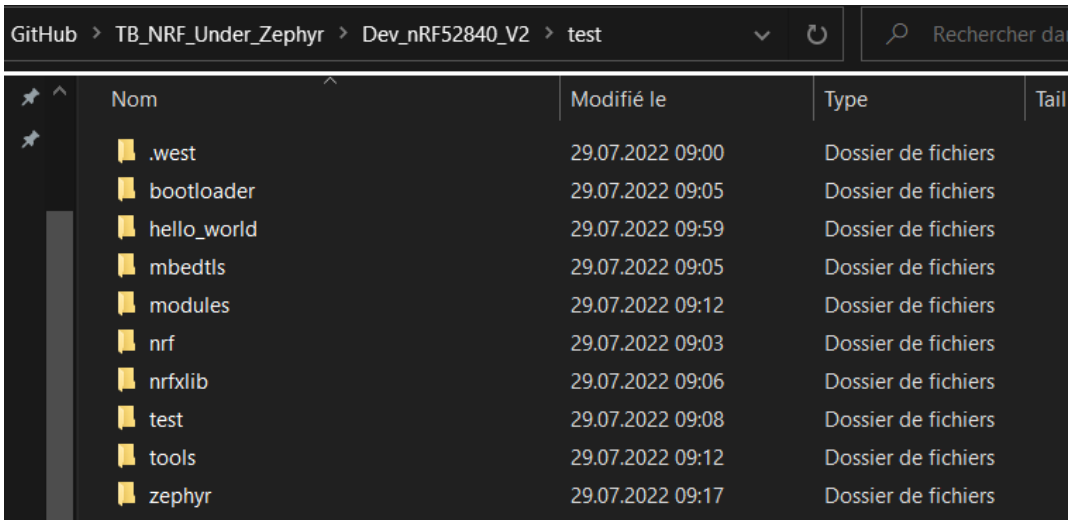


Figure 8: West workspace

[2] https://nrfconnect.github.io/vscode-nrf-connect/connect/create_app.html

When developing an application with nRF Connect SDK, you have the option of working with or without West workspace. When creating your first project, you have the choice between 2 types of application: freestanding or workspace.

It is recommended to choose freestanding if you are developing a single application. On the contrary, if you plan to work with several applications or if you wish to have a west workspace, we advise you to choose the 2nd solution.

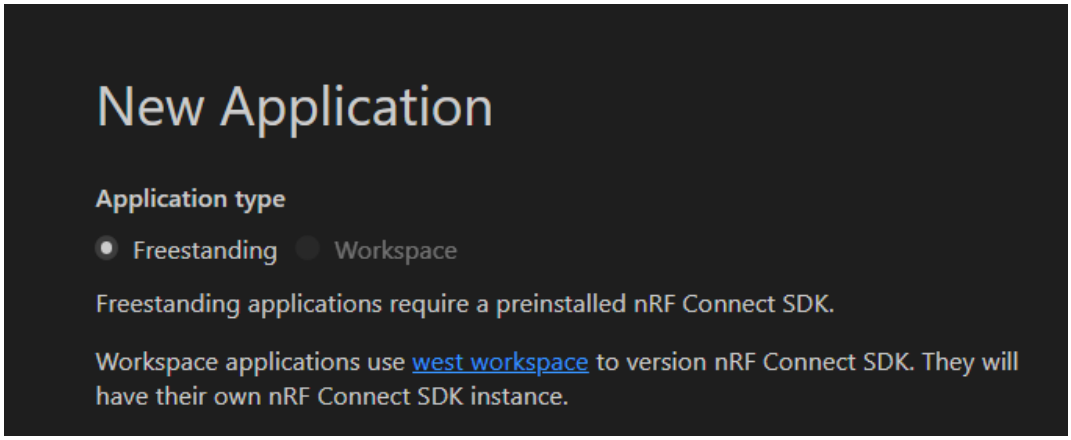


Figure 9: Create a new application

3.5 Creation of a project with Workspace

1. Open nRF Connect for Desktop
2. Open Toolchain Manager
3. Open VS Code with the version of nRF Connect SDK wanted
    a. The fact of going through the nRF Connect application and not directly through VS Code makes it possible to check if all the dependencies used are installed and updated.
4. If any folder or workspace ais open, close it. File → close Workspace
5. Go to the nRF Connect extension.
6. In the "Welcome" panel, click on Create a new application.
    a. Select Workspace for Application type.
    b. Choose the location of your project (nearest the root).
    c. Select sample to use.
    d. Name your project

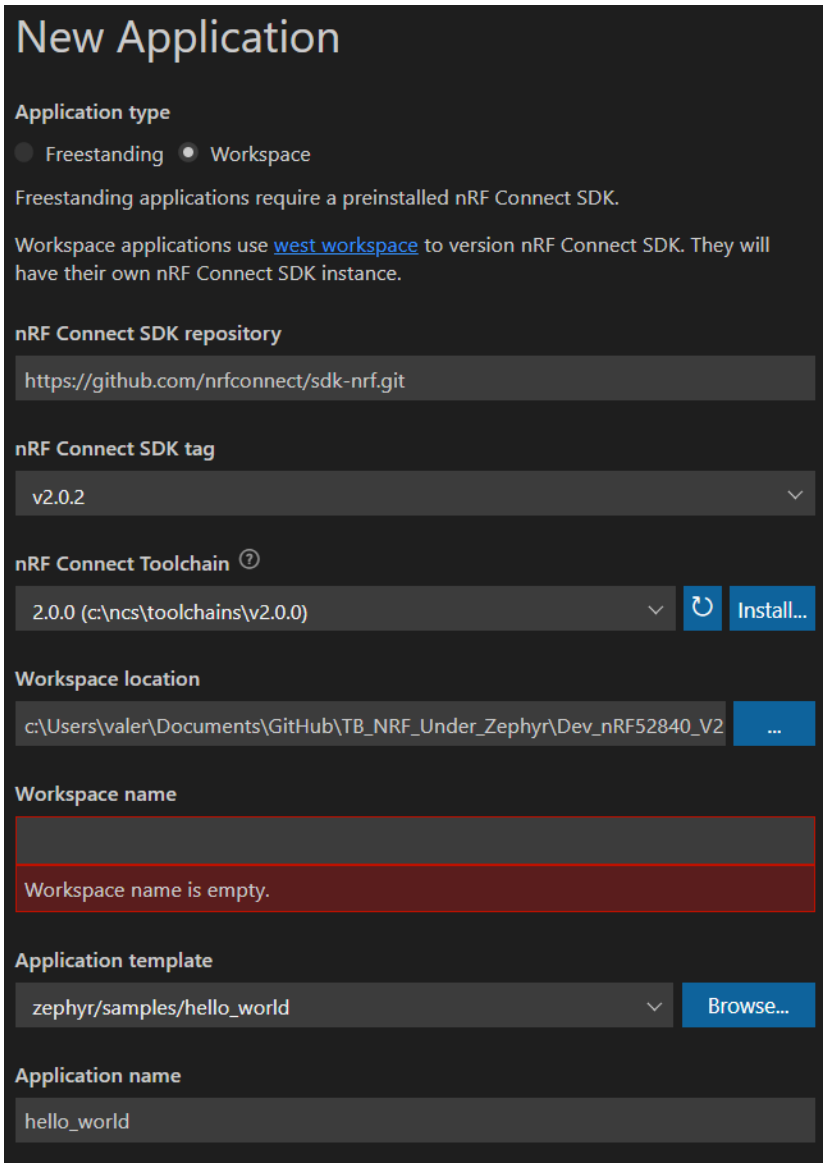


Figure 10: Create a new project with workspace

When creating the application, an error will occur and the project will not open. This is due to an indentation error in one of the files. You must go to the location of your project and find the `west.yml`. Make the changes as shown in the following figures:

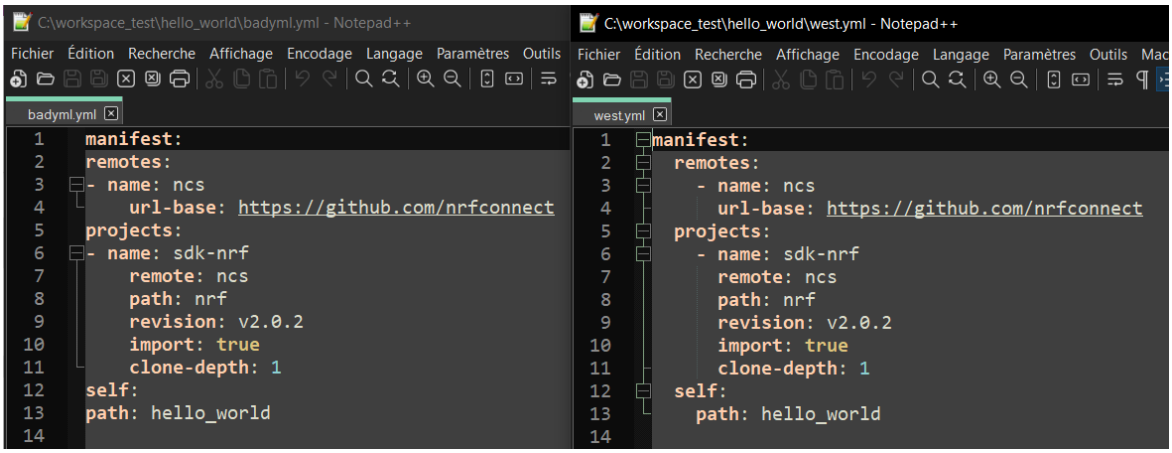


Figure 11: Correction to add for west.yml

7. Open Toolchain Manage
    a. Click on the arrow next to the Open VS Code button of the version of nRF Connect SDK that you are using. → Open command prompt...

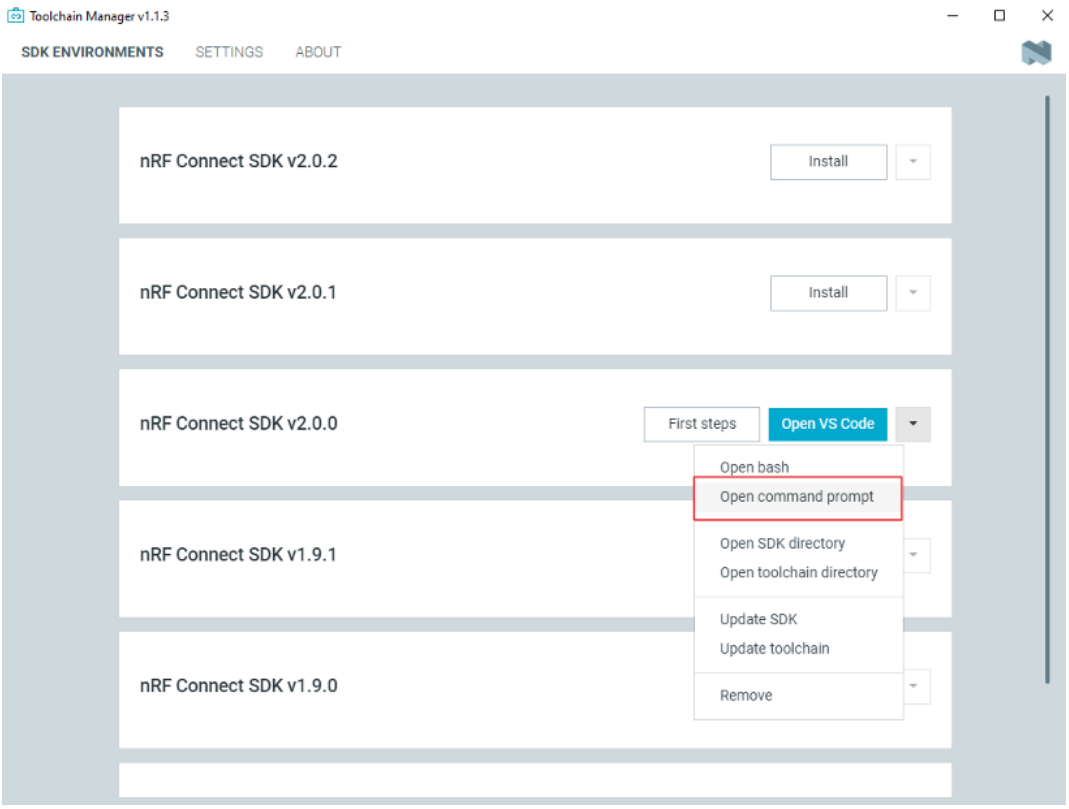


Figure 12: How to open command prompt

b. Go to the location of the project (same level as .west)
c. write the command: west update
d. close the window when finish

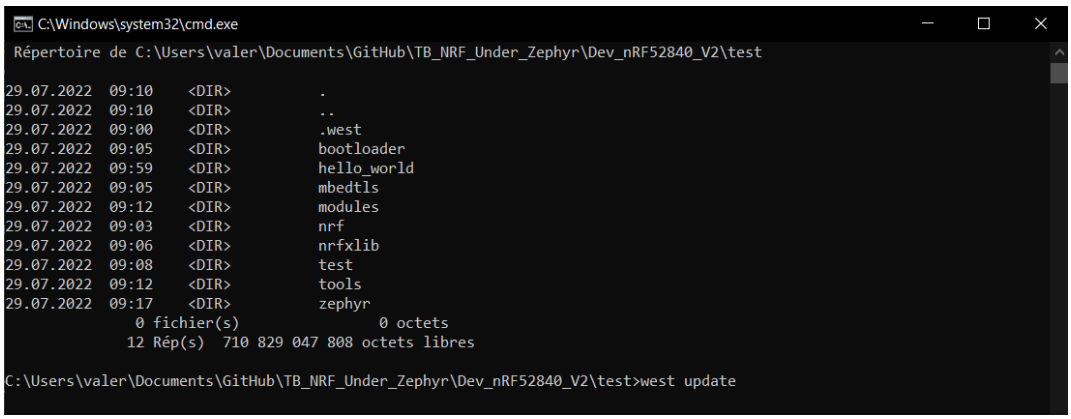


Figure 13: How to update West

8. Open VS Code with the version of nRF Connect SDK wanted
9. Go to the nRF Connect extension.
10. In the “Welcome” panel, click on add a new Application and search for your project.

General comment:

- The indentation error was reported to the Nordic team through several tickets on the Devzone. However, no solution has been provided yet.
- It is important to note that you should never open VS Code directly but always go through the nRF Connect for Desktop application. With nRF Connect SDK for Desktop and Toolchain Manager, you will have the west workspace with the chosen version of nRF Connect SDK.
- Choose Freestanding for new application type only if you had to add a project to an existing workspace.

3.6 Open an existing project using west

1. Open nRF Connect for Desktop
2. Open Toolchain Manager
    a. Click on the arrow next to the Open VS Code button of the version of nRF Connect SDK that you are using. → Open command prompt...
    b. Go to the location of the project (same level as .west)
    c. write the command: west update
    d. close the window when finish
3. Open VS Code with the version of nRF Connect SDK wanted
4. If any folder or workspace are open, close it.
5. Click on File → Open Workspace from File…

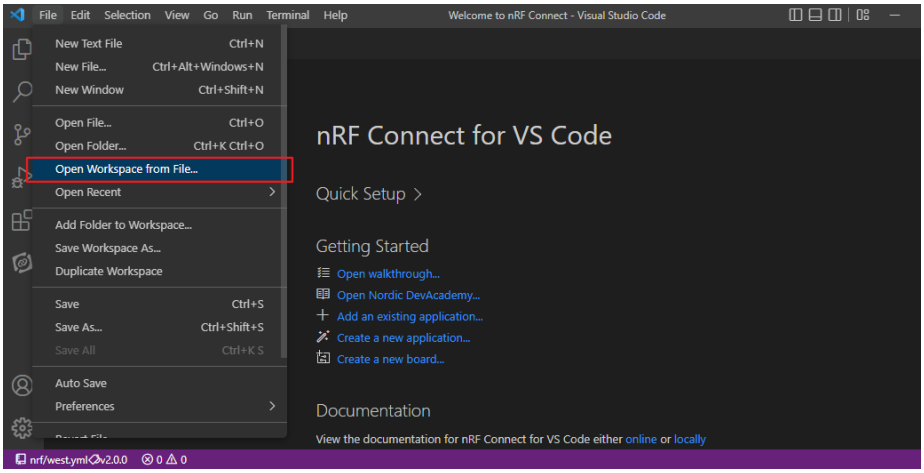


Figure 14: How to open workspace

6. Go to the project you want to open and select the JSON <name>.code-workspace
7. To add any project of the select worksapce,go to the nRF Connect extension, “Welcome” panel and click on + Add an existing application

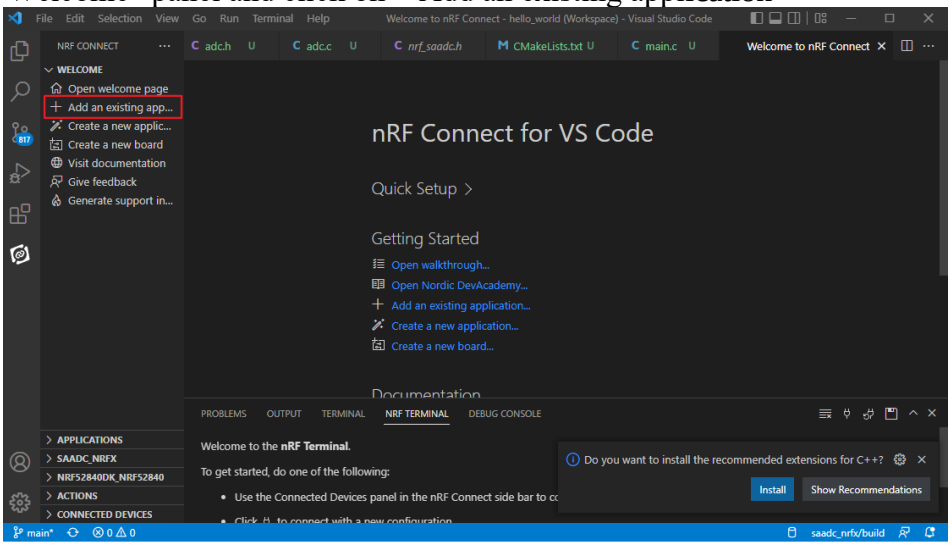


Figure 15: How to add an existing application

4 About Zephyr

4.1 What is an RTOS?

The goal of Real-Time Operating System (RTOS) is to ensure predictable and determinist execution pattern. Embedded system often have timing requirement and certain task must be done in a set time.
Each RTOS has a scheduler that decide which tasks to execute at which time based on setting priorities of different thread.

Since 2018, Nordic has include Zephyr RTOS in their new SDK. Zephyr is a recent and open source RTOS build for low power wireless. Zephyr was chosen because it is independent, scalable, stable and open source.

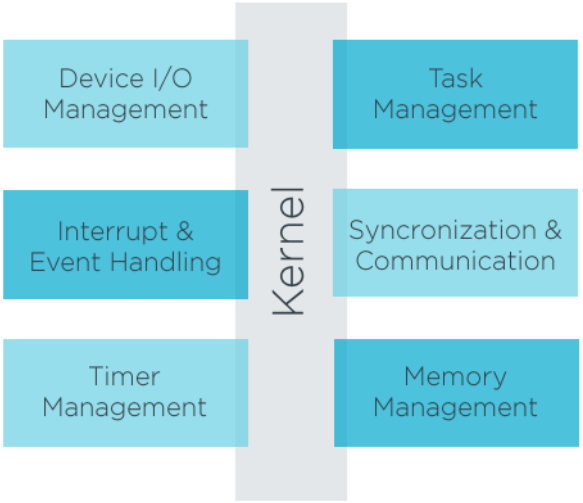


Figure 16: Zephyr Kernel[3]

[3] https://devzone.nordicsemi.com/cfs-file/__key/telligent-evolution-components-attachments/01-04-00-00-00-00-12-82/Introduction-to-nRF-Connect-SDK.pdf

4.2 Threads

A thread is a small set of instructions that can be managed independently by a scheduler. It allowed to split the program into two or more running tasks. With Zephyr, any number of threads can be created (limited only by available RAM). Only one thread can be executed at each time. Therefore there are several states in which a thread can be found:

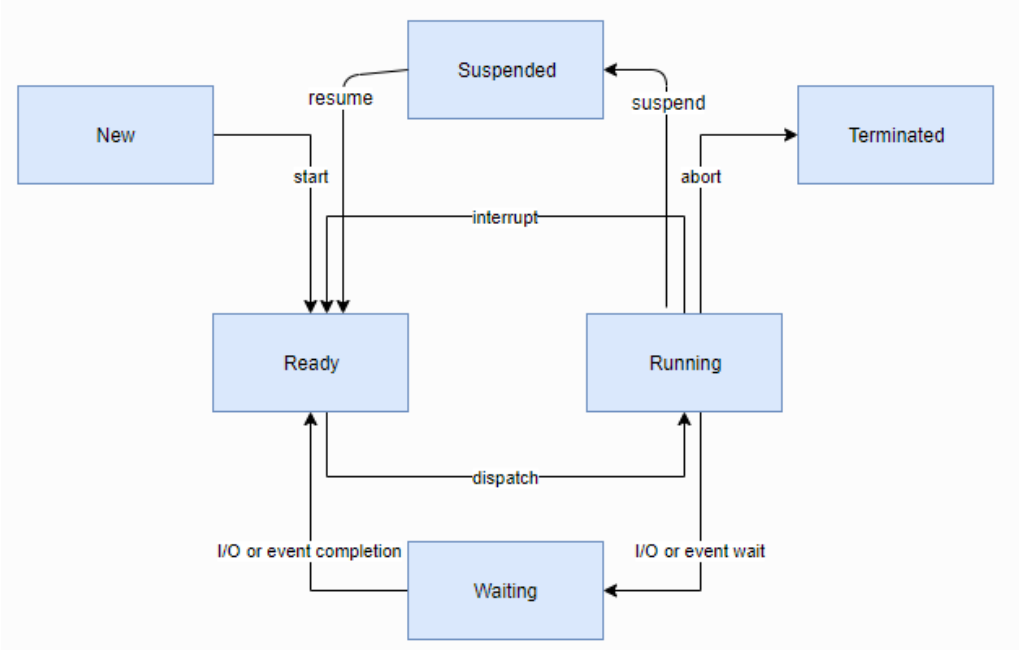


Figure 17: State diagram of thread

https://docs.zephyrproject.org/3.1.0/kernel/services/threads/index.html
There are several factors that make a thread unready:

- A delayed start.
- Waiting for a resource already used by another thread (semaphore).
- Waiting for a timeout.
- Thread suspended or terminated.

4.3 Create a Thread

A thread has the following properties:

- A **thread id** that references the thread.
- A **stack area** with a define **size** that represent the region of memory used for the thread.
- An **entry point function** which a maximum of 3 argument values.
- A scheduling **priority** to indicate if a thread is more prioritized than another thread. This is an integer value and can be either negative or non-negative. Lower priorities takes precedence over higher values.
- **Thread options**.
- A **start delay** that allowed to start thread later than is definition.
- An **execution mode**.

There is two possibility of create a thread: Either use the define `K_THREAD_DEFINE()` , or use the function `k_thread_create()`. Refer to documentation of Zephyr for more information.

5 Performance's analysis of the nRF52840 Nordic device under Zephyr

5.1 The interest of study latency and response time

It is relevant to notice the importance of timing when we are working with a real-time system. In a medical context it is more important to develop efficient and accurate applications. Indeed, an application with medical purposes must be accurate and transmit information concerning the patient's health as quickly as possible without affecting the rest of the application. Any disruption in the time scale can have serious consequences.

What is meant by **interrupt latency**? The amount of time that elapses between a device interrupt request and the first instruction of the corresponding ISR. It is important to determine the different possible cases that could lead to a modification of this latency and select which cases are the most interesting.

Another important time is the **response time**. This is the time that is the most interesting when we are developing application and usually the main factor that interest factory. Indeed the time consecrated to save/select/load context must be taken in count especially when we are working with a RTOS application with a scheduler working in the background.

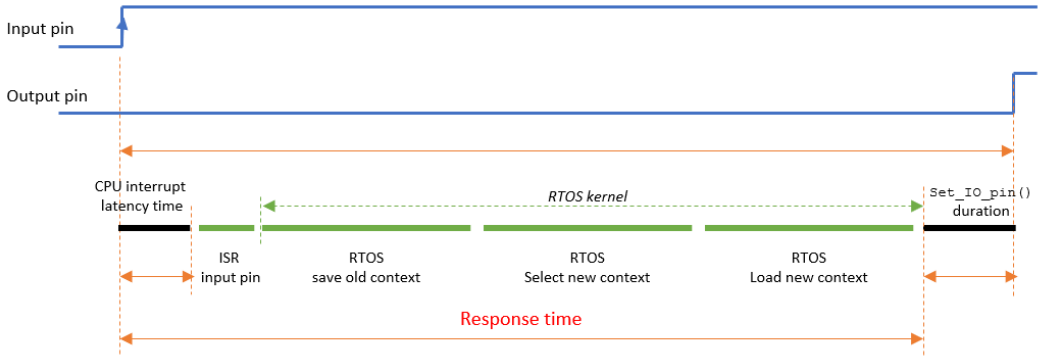


Figure 18: Study of time reaction

5.2 The Cortex-M4

The nRF52840 board has a Cortex-M4 high performance 32-bit processor running at 64MHz. His ultra-low power consumption and his optional deep sleep mode as well as his 3-stage pipeline Harvard architecture makes it ideal for embedded application.

"The Cortex-M4 processor closely integrates a configurable Nested Vectored Interrupt Controller (NVIC), to deliver industry-leading interrupt performance. The NVIC includes a Non-Maskable Interrupt (NMI) that can provide up to 256 interrupt priority levels. The tight integration of the processor core and NVIC provides fast execution of Interrupt Service Routines (ISRs), dramatically reducing the interrupt latency."[4]

When we talk about interrupt latency, we like to know how many cycles it takes to handle an interrupt. The ARM community provide the following table[5] :

Processors	Cycles with zero wait state memory
Cortex-M0	16
Cortex-M0+	15
Cortex-M3	12
Cortex-M4	12

Figure 19: Interrupt latency of Cortex-M processors

If we count an additional 17 cycles with FPU activated, we can calculate the approximate the interrupt latency :

$$Interrupt\ Latency\ = \frac{1}{64Mhz} \cdot (12 + 17) = 453ns$$

This value is ideal and does not represent the absolute reality. The interrupt latency depends on what the CPU is doing when an interrupt occurs.

[4] https://developer.arm.com/documentation/dui0553/b/?lang=en

[5] https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/beginner-guide-on-interrupt-latency-and-interrupt-latency-of-the-arm-cortex-m-processors

5.3 Set up and protocol

A simple way to measure interrupt latency is to use a button to create an interrupt and toggle a pin.
A list of the most interesting test has been establishing at the beginning of the project. Here's what it's made of:

- Measure output pin on/ output pin off latency:
    - Can be done by toggling an output pin
- Measure the latency from “signal on input pin” to “signal in ISR”
- Measure the latency from “signal on input pin” to “signal in output”
- Measure latency “signal on input pin” to “signal on output pin” with BT transfer in between in advertise mode.
- Measure latency “signal on input pin” to “signal on output pin” with BT transfer in between in connected mode.

In order to measure the different latencies, an oscilloscope was used to visualise the results. The oscilloscope used was an Analog discovery 2. In addition to measuring signals, it is possible to generate them. This is a particularly useful feature for testing the ADC (see chapter 7.2).

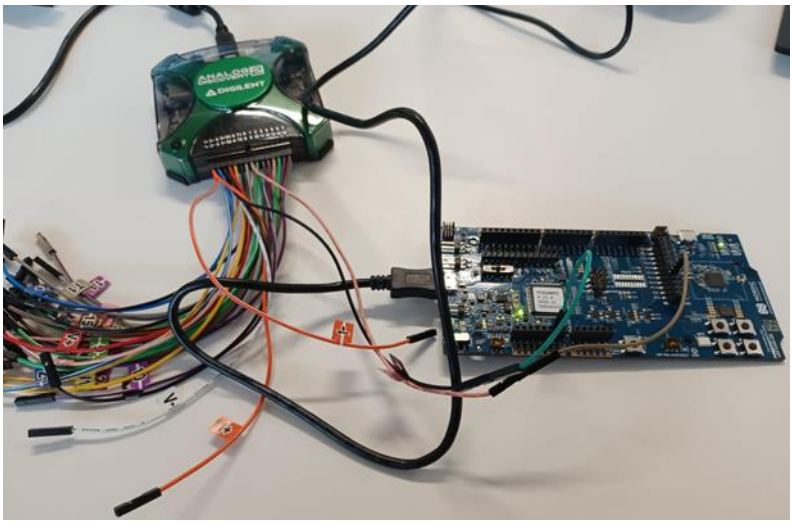


Figure 20: Test environment

5.4 Test and Results

5.4.2 Toggle an output pin

There are several methods for changing the value of a PIN. For this **first test**, the goal is to measure the time the processor takes to change the value of a pin. It's interesting to establish a list of all method allowing to modify output. We can set the value of a pin or directly change the value of port with mask.

```
Welcome to the test zone. There is quite multiple method to change the state of an output pin
Here the different method tested :

Test 1 :        gpio_pin_toggle(device,pin)
Test 2 :        gpio_pin_set(device,pin,value)
Test 3 :        gpio_pin_set_raw(device,pin,value)
Test 4 :        gpio_port_set_masked_raw(device,mask,value)
Test 5 :        gpio_port_set_bits_raw & gpio_port_clear_bits_raw  (device,port_pin)
Test 6 :        gpio_port_set_clr_bits_raw(device , gpio_port_pins_t set_pins, gpio_port_pins_t clear_pins)
Test 7 :        gpio_port_toggle_bits(device, port_pin)
```

Figure 21: List of all method test to toggle an output pin

I use the mode Logic with Waveform to visualize my pin. I connected D0 to my LED1:

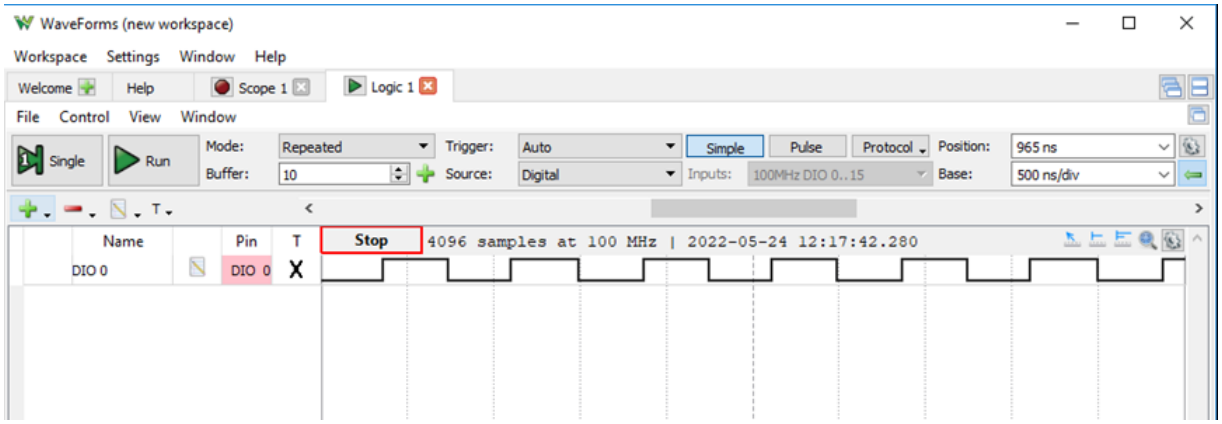


Figure 22: Measure process time

We can see in the below the different measure corresponding to each method. All measure were realised 3 times and the final value is the average of them:

			Meas 1	Meas 2	Meas 3	Mean
Pin	1. Set	Up	601 ns	592 ns	600 ns	590-600 n
		Down	650ns	640 ns	650 ns	650 ns
	2. Set_raw	Up	481 ns	477 ns	490 ns	480 ns
		Down	462 ns	460 ns	461 ns	460 ns
	3. Toggle	Up	520 ns	529 ns	538 ns	529 ns
		Down	540 ns	542 ns	538 ns	539 ns
Port	4. Set_mask_raw	Up				
		Down				
	5. Set OR clear	Up	490 ns	480 ns	479 ns	480 ns
		Down	460 ns	460 ns	461 ns	460 ns
	6. Set AND clear	Up	570 ns	571 ns	579 ns	580 ns
		Down	550 ns	551 ns	561 ns	550 ns
	7. Toggle	Up	579 ns	578 ns	578 ns	578 ns
		Down	593 ns	583 ns	581 ns	587 ns

Figure 23: Results of test 1 for interrupt latency with nRF52840

We can that there is not a notable difference between the various method to toggle a pin. If we go through the library GPIO given by Zephyr, we can notice that several method calling each other like `gpio_pin_set()` call `gpio_pin_set_raw()` which call `gpio_port_clear_bits/gpio_port_set_bits_raw()`. It can explain why some methods are a little bit faster than another.

Comment

At the beginning of the TB, the measurements were made on the nRF5340 before the company expressed its willingness to work with the nRF52840. I then measured values of 370ns by toggling a pin. Knowing that the nRF5340 has a clock 2 to 4 times faster than the nRF52840, the measurements are consistent. In the end we can see that the `set_raw()` function is the “fastest”.

5.4.3 Input to Output

Another interesting value to measure is the response time. It is an important factor when developing a new product. It is not possible to directly measure it with an oscilloscope. We must at first measure the total time between an interruption and a toggled LED.
For the **second test**, an interrupt is generated with a button and turn on a led. The. The code used for the measurement consists of the following steps:

1. Configure output and input pin
2. Add a callback depending on the button pressed
3. Write the interrupt routine. Here only the function `gpio_pin_toggle()` was called.

```
void button_pressed(const struct device *dev, struct gpio_callback *cb,
            uint32_t pins)
{
    gpio_pin_toggle(dev,PIN);
}
/* STEP 3 - Define a variable of type static struct gpio_callback */
static struct gpio_callback button_cb_data;

void main(void)
{
    const struct device *dev;
    int ret;

    dev = device_get_binding(LED0);
    if (dev == NULL) {
        return;
    }
    ret = gpio_pin_configure(dev, PIN, GPIO_OUTPUT_ACTIVE | FLAGS);
    if (ret < 0) {
        return;
    }
    ret = gpio_pin_configure(dev, SW0_GPIO_PIN, GPIO_INPUT | SW0_GPIO_FLAGS);
    if (ret < 0) {
        return;
    }
    /* STEP 4 - Configure the interrupt on the pin by calling the function gpio_pin_interrupt_configure()  */
    ret = gpio_pin_interrupt_configure(dev,
                       SW0_GPIO_PIN,
                       GPIO_INT_EDGE_TO_ACTIVE | GPIO_INT_DEBOUNCE);
    /* STEP 6 - Initialize the static struct gpio_callback variable   */
    gpio_init_callback(&button_cb_data, button_pressed, BIT(SW0_GPIO_PIN));
    /* STEP 7 - Add the callback function by calling gpio_add_callback()   */
    gpio_add_callback(dev, &button_cb_data);
    while (1) {
        /* STEP 8 - Remove the polling code and put the main thread to sleep */
        k_msleep(SLEEP_TIME_MS);
```

Figure 24: Code to measure response time

This time, I connected D0 to the led and D1 to the button on the Analog discovery. Here are the time I measured :

nRF52840 : ~21,5us

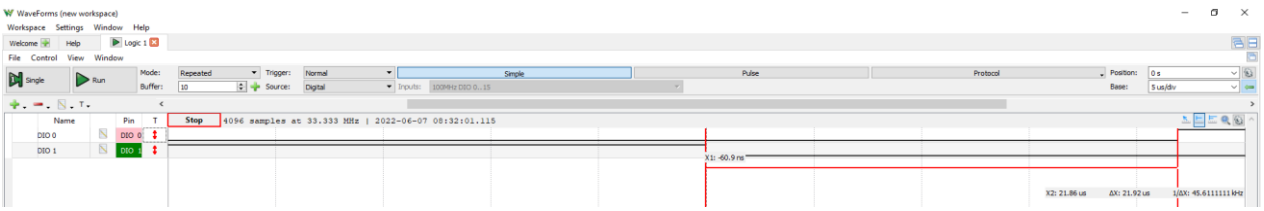


Figure 25: Toggle a pin after creating external interrupt

5.4.4 Response time

Now that we have realised some measure and get value, we can compete the Figure 18. The CPU interrupt latency time has been calculating with an equation the previous equation :

$$Interrupt\ Latency\ = \frac{1}{64Mhz} \cdot (12 + 17) = 453ns$$

The time to set an IO pin was measure with the first test:

$$\sim 500ns$$

Finally, the total time between an interrupt and a led set is:

$$\sim 21.5\mu s$$

We can then calculate the response time with the following equation:

$$Response\ time = Total\ time_{between\ Interrupt\ and\ set} - Time_{to\ set\ IO\ pin}$$
$$21\mu s = 21.5\mu s - 500ns$$

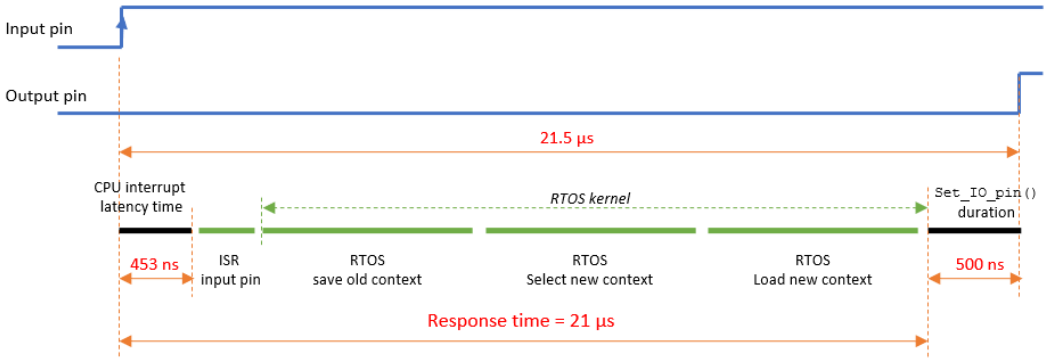


Figure 26: Measure value for response time and interrupt latency

6 Bluetooth communication on nRF52 under Zephyr

Nordic Semiconductor is a semiconductor company specialize in wireless communication technology that power IoT. Their Bluetooth Low Energy solution pioneered ultra-low power wireless and made them global market leader. Therefore, the Bluetooth Communication is a main part of this study. The goal is not to give a full description of how BLE function, but it is important to have a small introduction of Bluetooth.

6.1 Introduction of Bluetooth Low Energy

When developing Bluetooth application, this is important to define which type of communication we want, what information will be exchanged and if we must secure the communication.

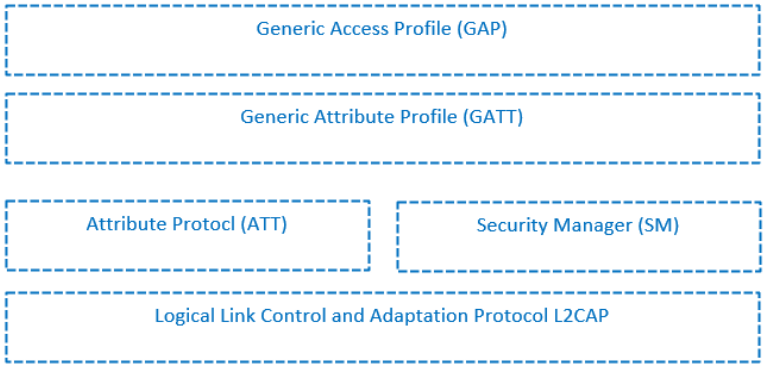


Figure 27: BLE Stack Protocol

The **GAP** (Generic Access Profile) define the discovery, connection and link. It makes device visible to the outside world.

After a connection, **GATT** (Generic Attribute Profile) service and characteristic can be used to exchange more data. GATT define the way that two BLE device transfer data back through service and characteristic.

ATT (attribute protocol) protocol describe service and characteristic.

SM (Security Manager) takes care of the security of data exchanges between two devices.

The L2CAP (Logical Link Control and Adaptation Layer Protocol) transfers data between the upper layers of the host and the lower layer protocol stack.

6.1.2 GAP Role

Bluetooth allows two devices to exchange information through wireless communication. Each device using Bluetooth Low Energy can be used in four different roles described in the following figure:

GAP role	Link Layer State
Broadcaster	Advertising
Observer	Scanning
Peripheral	Advertising + connection (slave)
Central	Scanning + initiating connection (master)

Tableau 3: Different role of BLE available

We can resume these roles in two specific case:

- One-way communication with a device that only advertises packets. This is called a **broadcaster**. A device in central or observer mode can then receive and read the information contained in the packet.
- A bi-directional communication. This is called connectivity between two devices. A device in **peripheral mode** will advertise to announce its presence. A second device in **central mode** will then request a connection.

Thanks to Bluetooth 5.3 it is possible that a device has the role of central and peripheral. For the purpose of this project, we will only be interested in the advertise mode without connection.

6.1.3 Bluetooth LE Advertising

When we talk about communication, we also talk about the frequency spectrum used. Here is the frequency spectrum used by Bluetooth Low Energy:

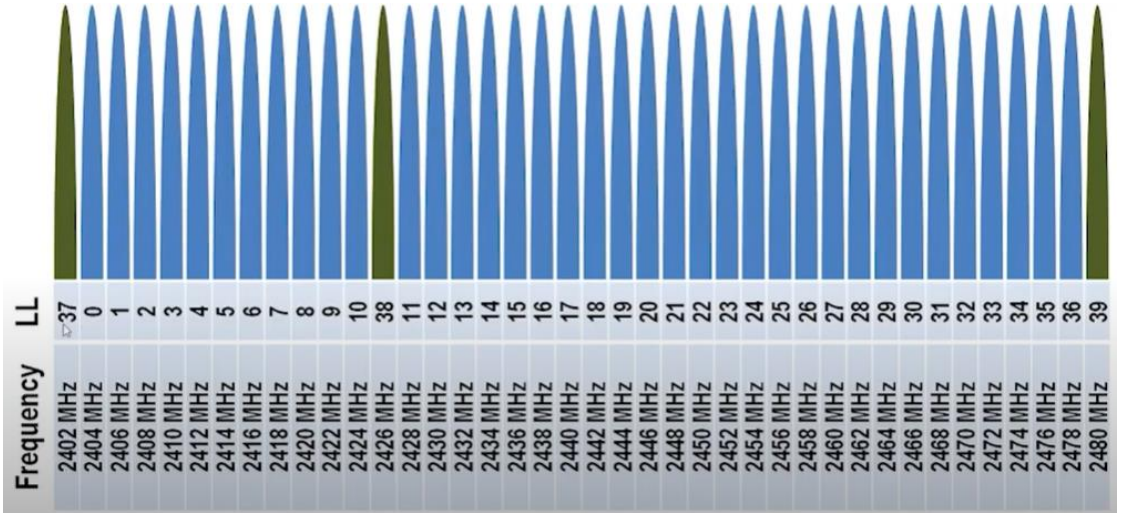


Figure 28: Frequency Spectrum of BLE (legacy mode)

The basic Bluetooth mode uses only three channels: 37-38-39. It is called the legacy mode. A new feature of Bluetooth 5.0 is the advertising extension. It allows more information to be transmitted in a packet and uses all channels in the spectrum. Each packet sent contains an indication of the next channel used.

There are several types of advertisement. An advertisement packet can be connectable or not connectable, scannable or not scannable. Scannable allows devices to advertise more data than can fit into one advertising packet.

Type	Scannable	Connectable
ADV_IND	Yes	Yes
ADV_SCAN_IND	Yes	
ADV_NONCONN_IND		
ADV_DIRECT_IND		Yes

Figure 29: Advertise type method

6.2 Bluetooth sample

6.2.2 Overview

The broadcaster and observer sample demonstrates how to use Bluetooth communication without connection. You can program the sample of broadcaster to a nrf52840 and the sample of observer to another nrf52840. If you don't have two board, you can easily use the application of nRF Connect on your smartphone to visualize advertise packet and create your own advertise packet. It is important to note that all the sample were written in C with C object-oriented pattern.

6.2.3 Implementation of Broadcaster

It is quite simple to realise a broadcaster. There are 3 elements that need to be defined to make a broadcaster:

1. Data to sent
2. Size of the data
3. Type of advertisement with its configuration

The first step is to **enable Bluetooth** with the function `bt_enable()` . This function must be called before any calls that require communication with the local Bluetooth Hardware. Then a broadcaster "object" is created. The users can even **pass the data and size** of data as argument or called the function `Broad_set_data()`. Finally, the **advertise is start** with the function `Board_start_adv()`.

The function `Board_start_adv()` contains the function `bt_le_adv_start()`. This function needs 5 parameters:

1. Advertising parameters
2. The data to send
3. Size of data
4. Data to use in scan response
5. Size of data

As we do not use scannable advertise packet, we don't need the two last parameters. First we have the advertising parameters:

```
/**
 * @brief Non-connectable advertising
 * Other param are set at 0 and id is set at bt_id_default
 *
 * @param _options   No advertiser option
 * @param _int_min   Minimum advertising interval (Range: 0x0020 to 0x4000)
 * @param _int_max   Maximum advertising interval (Range: 0x0020 to 0x4000)
 * @param _peer      Peer address, set to NULL for undirected advertising
 */
#define BT_LE_ADV_NCONN_CUSTOM BT_LE_ADV_PARAM(0 , \
                                               0x0020, \
                                               BT_GAP_ADV_FAST_INT_MAX_2, NULL)
```

Figure 30: advertising parameter

How data are organized in the advertise packet. The most interesting part of the advertise is the PDU (Protocol Data Unit) field. That is the field that includes "user-data" that your application is interested in sending on.[6]

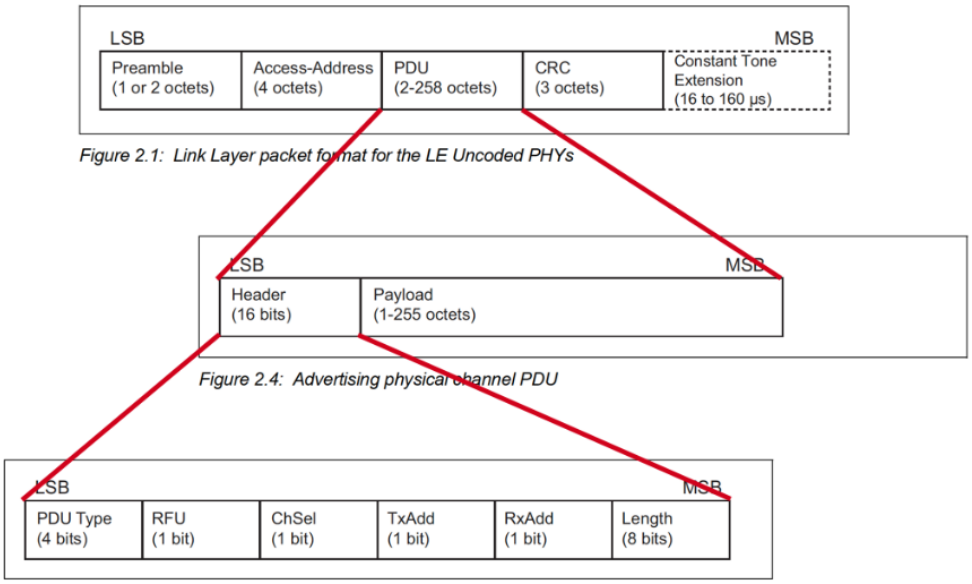


Figure 31: Architecture of advertise packet

- PDU Type Type of advertisement (here NON-CONNECTABLE)
- RFU Reserved for future use
- CHSel 1 if LE Channel selection Algorithm #2 is supported
- TxAdd 0 if address is public, 1 if random
- RxADD 0 if target's address is public, 1 if random
- Length Payload length

In the payload section, you will find the data you want to send. There is different type of data represented by data type. In this sample we used two types of data:

1. `BT_DATA_MANUFACTURER_DATA (0xFF)`
2. `BT_DATA_NAME_COMPLETE (0x09)`

In the payload field, you always have the type of data first, then the size of data and finally the data.

In this sample, you found your personal data under the manufacturer data. It is important to note that the first two bytes always represent the Bluetooth ID of your company given by the SIG. Here 0x025A is the id of the HES-SO Valais Wallis. We used "Ypsomed" as the name of the device.

```
static  struct bt_data msg_data;
msg_data.data_len = me->size;
msg_data.type = BT_DATA_MANUFACTURER_DATA;
msg_data.data = me->mfg_data;

struct bt_data adv[] = {msg_data,
BT_DATA(BT_DATA_NAME_COMPLETE, NON_CONNECTABLE_DEVICE_NAME, sizeof(NON_CONNECTABLE_DEVICE_NAME) - 1)};


err = bt_le_adv_start(BT_LE_ADV_NCONN_CUSTOM, adv, ARRAY_SIZE(adv), NULL, 0);
```

Figure 32: Construction of the advertise packet

[6] https://novelbits.io/bluetooth-low-energy-advertisements-part-1/

6.2.4 Implementation of Observer

The sample of the observer requires more work. If starting the scan is pretty simple then you have to sort out all the advertise packets to keep only the ones you are interested in.

As the broadcaster sample, the first thing to do is to **enable the Bluetooth**. The second step is to **initialize the scan**. You must call the function `bt_scan_init()` and give a `struct bt_scan_init_param`. This struct have 3 parameters:

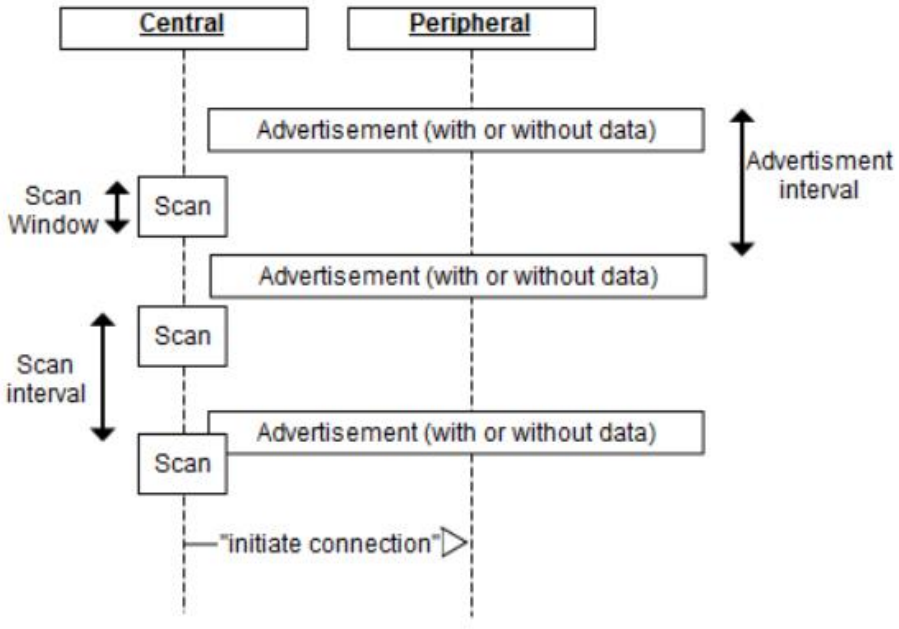
```
1. const struct bt_le_scan_param
2. bool connect_if_match
3. const struct bt_le_conn_param
```

The last two parameters are set to false and null because we are in non-connectable mode. The `bt_le_scan_param` is initialized in the following way:

```
const  struct bt_le_scan_param scan_parameter = {
        .type       = BT_LE_SCAN_TYPE_PASSIVE,
        .options    = BT_LE_SCAN_OPT_FILTER_DUPLICATE,
        .interval   = BT_GAP_SCAN_FAST_INTERVAL,
        .window     = BT_GAP_SCAN_FAST_WINDOW,
    };
```

Figure 33: Scan parameter

The scan type indicates if we want to send request to acquire more data from the broadcaster. As we are in non-scannable mode, we set this param at passive. You can also choose the interval and scan windows.



[7]

Figure 34: Scan window, scan interval, advertise interval

[7] https://devzone.nordicsemi.com/guides/short-range-guides/b/bluetooth-low-energy/posts/ble-central-tutorial

The third step, to realise observer, is to **add callback**. We need to define a `cb_data` structure to store callback functions pointers. There is 4 functions we can pass:

1. filter match
2. filter no match
3. connecting
4. connecting error

The last two functions are set NULL as we are in non-connectable mode. It is possible to set filters to filter the advertise packet. Unfortunately, this feature does not seem to work with nRF Connect SDK. A ticket has been opened in the devzone but no answer has been given.

The last step is to **start the scanning** with the `bt_scan_start function()`.

Even if the filters don't seem to work, it was necessary to find a solution to filter advertise packet and get data from it. A function to parse has been realise. The goal of this function is to go through payload, check if it contains the type of data wanted and give back data and the size of data.

```
uint32_t adv_report_parse(struct net_buf_simple* packet,struct data_p*  info)
{
    uint32_t  index = 0;
    uint8_t * p_data;

    p_data = packet->data;

    while (index < packet->len)
    {
        uint8_t field_length = p_data[index];
        uint8_t field_type   = p_data[index + 1];

        if (field_type == info->type)
        {

            memcpy(info->data,&p_data[index + 2],field_length-1);
            info->len = field_length-3;
            return 1;
        }
        index += field_length + 1;
    }
    return 0;
}
```

Figure 35: Method to parse data from advertise packet

The first argument is a pointer to the payload and the second one is a pointer on an homemade `struct data_p`, that contain the id type of data to search through the payload. If
A match is found, the data and the size of data is stock in the `data_p struct`.

For this sample, first we filter the advertise packet with the device name (0x025A). Then we get data by searching manufacturer data.

7 Samples

7.1 Use cases

7.2 SAADC

In electronics, an analog-to-digital converter (ADC) is a system that converts an analog signal, such as a sound picked up by a microphone or light entering a digital camera, into a digital signal[8]. An ADC converts a continuous-time and continuous-amplitude analog signal to a discrete-time and discrete-amplitude digital signal (sampling the input).

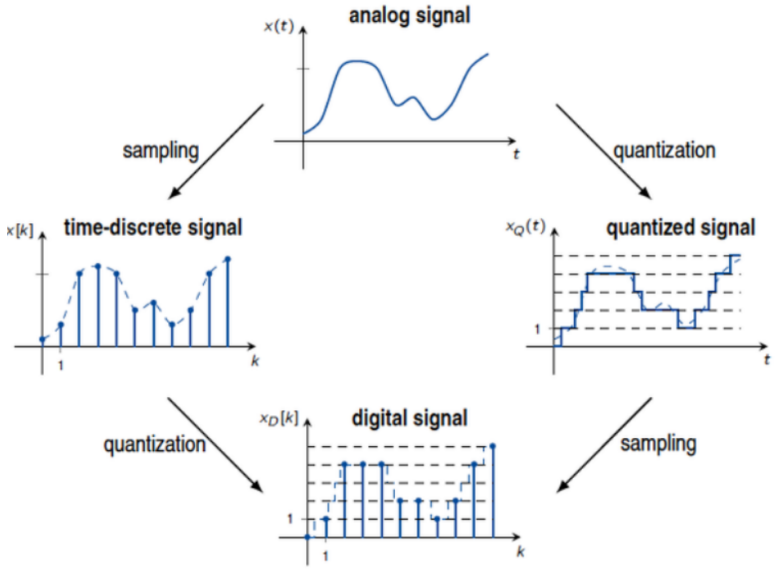


Figure 36:Scheme of ADC

With the nRF52840 we talk about SAAD (Successive approximation analog-to-digital converter).
There is up to 8 input channels that can be used for SAADC with the nRF52840:

Analog Input Channel	GPIO Pin Number
AIN0	P0.02
AIN1	P0.03
AIN2	P0.04
AIN3	P0.05
AIN4	P0.28
AIN5	P0.39
AIN6	P0.30
AIN7	P0.31

Tableau 4: Analog pin of the nRF52840 DK

In the description of the board, it is noted that we can have a 12-bit resolution but in reality, we are closer to a 10-bit resolution.

[8] Analog-to-Digital Converter - Embedded Artistry. https://embeddedartistry.com/fieldmanual-terms/analog-to-digital-converter/

7.3 ADC with nfrx drivers

7.3.2 Overview

The first sample of ADC has been realised with the nrfx drivers. Unlike Zephyr which has an extra layer of abstraction, we are closer to the hardware. This sample shows the use of callback after a sampling.

7.3.3 Implementation

To use the ADC with nrfx drivers, the first thing to do is adding the following configuration in the prj.conf :

$$CONFIG\_NRFX\_SAADC = y$$

You can then start to code.

The sample using callback, we must therefore **define interruption** for any ADC event. To do this, call the *IRQ_CONNECT* method with the address of the interrupt routine given as an argument: `nrfx_saadc_irq_handler`. Only after that, we can **initialize the saadc driver**.

We can then **create our ADC** "object". This struct is composed of three elements:

1. Array of struct `nrfx_saad_channel_t`. This contains configuration of each 8 channels planned for ADC.
2. Array of bool. Indicate which of the 8 channels are used for ADC.
3. Array of NRF enum name for the analog channel (NRF_SAADC_INPUT_AINx).

Then we call the ***Saadc_init*** method to set all activeChannel[i] values to false and fill the channelPin[i] array with the different enum of NRF.

For each channel use, you must **configure the respective channel** with the a*ctiveChannel* method.

```
if (channel >= 0 && channel < MAXCHANNEL)
{
    me->activeChannels[channel] = true;
    me->cfg_channels[channel].channel_config.gain = SAADC_GAIN;
    me->cfg_channels[channel].channel_config.reference = SAADC_REFERENCE;
    me->cfg_channels[channel].channel_config.acq_time = SAADC_ACQU_TIME;
    me->cfg_channels[channel].channel_config.mode = 0;
    me->cfg_channels[channel].channel_config.resistor_n = 0;
    me->cfg_channels[channel].channel_config.resistor_p = 0;
    me->cfg_channels[channel].pin_p = me->channelPin[channel];
```

Figure 37: Set configuration for saadc channel

In this sample, all the configuration are the same for all the saadc channel.

- `SAADC_GAIN` = Gain equal to 1
- `SAADC_REFERENCE` = internal reference
- `SAADC_ACQU_TIME` = 10us
- `Mode` = 0 (set to 1 for differential mode)
- `Resistor_n` = Internal resistor set at 0 by default
- `Resistor_p` = Internal resistor set at 0 by default
- `Pin_p` = NRF_SAAD_INPUT_AINx

After setting the configuration for all the channel used, we can call **Saadc_config_channel** that call the function `nrfx_saadc_channel_config`(). Finally we can call the `Saadc_start()` to **set** the last **advance configuration** and start to sample.

In `Saadc_start()`, we **set** the last configuration with **nrfx_saadc_adv_config_t**. The timer for sampling can be set. The `internal_timer_cc` parameter of this struct takes care of this. It is required to have a sampling period between 80 and 2047. To calculate this period, you just have to do the following operation:

$$\frac{16MHz(ADC_{clock})}{Freqency\ of\ sample}$$

After set `nrfx_saadc_config_t`, we can call the method `nrfx_saadc_advanced_mode_set()`. This method allow to **precise the resolution, give the advance setting and the handler event** for ADC. Before start measuring, we must configure the buffers that stock the sampling values. We **configure two buffers** to ensure double buffering of samples to avoid data loss when the sampling frequency is high. Finally, we **call function for triggering** the conversion in the configured mode.

Each time a saadc event occurred, the `event_handler()` is called. Each case is then processed to find out if the event was generated to warn of the end of sampling or if a buffer is full.

7.4 ADC with Zephyr

7.4.2 Overview

This second example of ADC has been realised with Zephyr library. In fact, Zephyr also uses the nrfx library but brings an additional layer of abstraction. Unlike the previous example, this one does not use a callback. But with this example, we use all the analog pins

7.4.3 Implementation

Because this example uses ADC with Zephyr library, the first thing to do is adding the following configuration in the prj.conf :

$$CONFIG\_ADC = y$$

As zephyr has a higher abstraction layer than the previous example, there is no need to call a method to initialize the ADC.

The first step is to **create an ADC** "object". This is a struct with the following parameters:

- `const struct device*` : A pointer to the device structure for the driver instance.
- `struct adc_channel_cfg`: A structure for specifying the configuration of an ADC Chanel. It contain 7 parameters:
  - Gain of ADC
  - Reference of ADC
  - Acquisition time
  - Id for the channel (you can set any ID)
  - mode of ADC (0 by default, 1 for differential)
  - input positive channel
  - input negative channel (only use for differential mode)
- `bool isInitialized[MAXCHANNEL]` : to indicate if a channel has been initialized.
- `uint8_t activeChannels[MAXCHANNEL]` : enum of ADC channel (NRF_SAADC_INPUT_AINx).
- `int16_t buffer[MAXCHANNEL]` : buffer to contain the sampling value.

When the constructor is called, all value are "reset" and the array of buffer is filled with 0 values.

The second step is to **initialize the hardware** with `Zadc_initHW()`. It will set the configuration and get the device. We kept the same configuration as the last sample:

```
me->cfg.gain = ADC_GAIN_1;
me->cfg.reference = ADC_REF_INTERNAL;
me->cfg.acquisition_time = ADC_ACQ_TIME_DEFAULT;
me->cfg.differential = 0;
me->cfg.input_positive = 0;
me->cfg.input_negative = 0;
```

Figure 38: ADC configuration (Zephyr version)

To use any channel for ADC, you must then call `Zadc_activateChannel()`. It will set the `activeChannels`[n°channel] = 1 to indicate this channel must be configured for ADC. The `adc_channel_setup()` method will then be called through `Zadc_initChannels()` and the **channel will finally be configured and ready for use**.

Unlike the previous example, we do not use a callback or timer to do periodic sampling. An infinite loop called each 500ms the `Zadc_readChannel()` to start a sampling. The `adc_read()` method is used to **read the value** from the channel and takes as a parameter an `adc_sequence struct` which contains information about the resolution of the ADC, the buffer where the value should be stored and its size.

7.5 PWM

A Pulse Width Modulation (PWM) is a modulation process for encoding the amplitude of a signal into a pulse width. PWM generate a pulse width modulated signals on GPIO and it allow to control the power supplied to various electrical device. It is particularly interesting to control AC/DC motors with it. A possible feature for the autoinjector of Ypsomed is to include a motor to control the injection of the syringe.

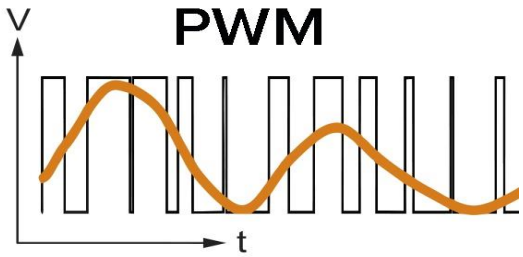


Figure 39: Representation of PWM[9]

The nRF52840 dk provide four PWM channels with individual polarity that can drive assigned GPIOs. The 4 channels used can be visualised on the following figure of the PWM module:

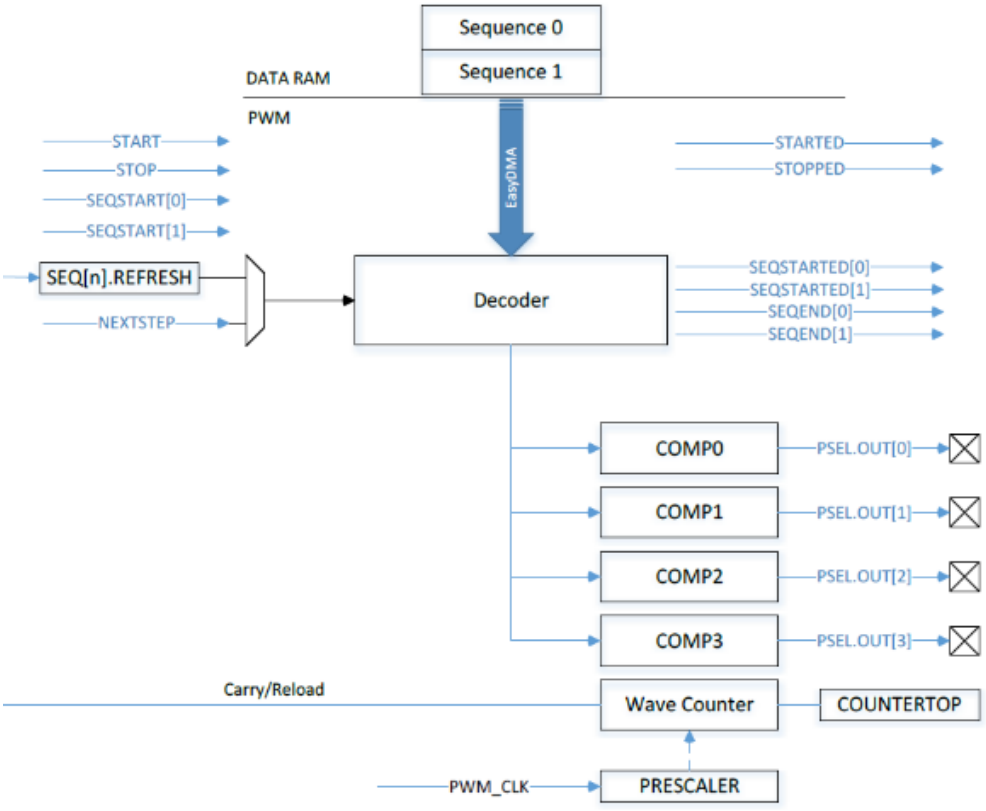


Figure 40: PWM Module[10]

[9] https://jimmywongiot.com/2021/06/01/advanced-pulse-width-modulation-pwm-on-nordic-nrf52-series/

[10] https://infocenter.nordicsemi.com/index.jsp?topic=%2Fps_nrf52840%2Fpwm.html

The maximum of PWM base clock frequency is 16MHz. This gives the following frequency range:

```
124  /** @brief PWM base clock frequencies. */
125  typedef enum
126  {
127      NRF_PWM_CLK_16MHz  = PWM_PRESCALER_PRESCALER_DIV_1,  ///< 16 MHz / 1 = 16 MHz.
128      NRF_PWM_CLK_8MHz   = PWM_PRESCALER_PRESCALER_DIV_2,  ///< 16 MHz / 2 = 8 MHz.
129      NRF_PWM_CLK_4MHz   = PWM_PRESCALER_PRESCALER_DIV_4,  ///< 16 MHz / 4 = 4 MHz.
130      NRF_PWM_CLK_2MHz   = PWM_PRESCALER_PRESCALER_DIV_8,  ///< 16 MHz / 8 = 2 MHz.
131      NRF_PWM_CLK_1MHz   = PWM_PRESCALER_PRESCALER_DIV_16, ///< 16 MHz / 16 = 1 MHz.
132      NRF_PWM_CLK_500kHz = PWM_PRESCALER_PRESCALER_DIV_32, ///< 16 MHz / 32 = 500 kHz.
133      NRF_PWM_CLK_250kHz = PWM_PRESCALER_PRESCALER_DIV_64, ///< 16 MHz / 64 = 250 kHz.
134      NRF_PWM_CLK_125kHz = PWM_PRESCALER_PRESCALER_DIV_128 ///< 16 MHz / 128 = 125 kHz.
135  } nrf_pwm_clk_t;
```

Figure 41: PWM frequency

A another interesting feature provide by the PWM module is the use of EasyDMA. A Direct memory access (DMA) allows certain hardware subsystems to access main system memory independently of the central processing unit (CPU). With the DMA, the nRF52840 can support a complex sequence waveform which plays from the Data RAM. We can have then autonomous and glitch-free update of duty cycle values directly from memory. For more details, see the documentation provided by Nordic Semiconductor.

7.6 PWM sample

7.6.2 Overview

As for the ADC use case, it is possible to use PWM with the nrfx drivers or with Zephyr library. Only the sample with Zephyr has been realise and this is the only example that does not apply the pattern C Oriented Object.

7.6.3 Implementation

To use Zephyr library, you must add the following configuration in the prj.conf :

$$CONFIG\_PWM = y$$

The implementation of PWM is done in 3 steps:

For the first step, we must **verify that the PWM device instance is ready** for use. The function `device_is_ready()` is called with a pointer to the device in question as parameter. The access to the pointer of the device wanted is made with the `struct pwm_dt_spec`.

The second step is reserved for the calibration. In this step, the calibration is used to find the maximum value for the PWM period. To do this, we start with a base period and divide this value by 2 as much as possible. When this is no longer possible, the last value tested will be used as the maximum value.

```
printk("Calibrating for channel %d...\n", pwm_led0.channel);
max_period = MAX_PERIOD;
while (pwm_set_dt(&pwm_led0, max_period, max_period / 2U)) {
    max_period /= 2U;
    if (max_period < (4U * MIN_PERIOD)) {
        printk("Error: PWM device "
               "does not support a period at least %lu\n",
               4U * MIN_PERIOD);
        return;
    }
}
```

Figure 42: Calibration of maximal value for PWM

The final step is dedicated to the initialisation of the PWM parameters. The function `pwm_set_dt()` is used with the following parameters:

- `Spec` PWM specification from device tree
- `period` Period (in nanoseconds)
- `pulse` Pulse width (in nanoseconds)

8 Demonstration

The different samples have been grouped together to provide a final example applying the features covered in this TB. As Ypsomed provided a medial context, this last example is inspired by their product, “Ypsomate”, and show a possible application in the future.

As mentioned in the Abstract chapter, Ypsomed develops solutions for people with diabetes. Let's take their Insulin pen and add our application.

When a patient has to inject himself with his insulin pen, the first step is to remove the cap of the pen. This is the trigger. It is therefore logical to represent this step by an interrupt that would be generated by a button. Our system, previously in sleep mode, would initiate the usage cycle.
After removing the plug, it is time for the injection. A PWM could control a motor that would push the right amount of Insulin.
After finishing the drug injection operation, we want to warn that the syringe has been used. This step is represented by Bluetooth communication. We use the ADC sample to send the values of an external signal via a notification packet.
Finally, a single button press puts the system back into sleep mode until a new cycle begins.

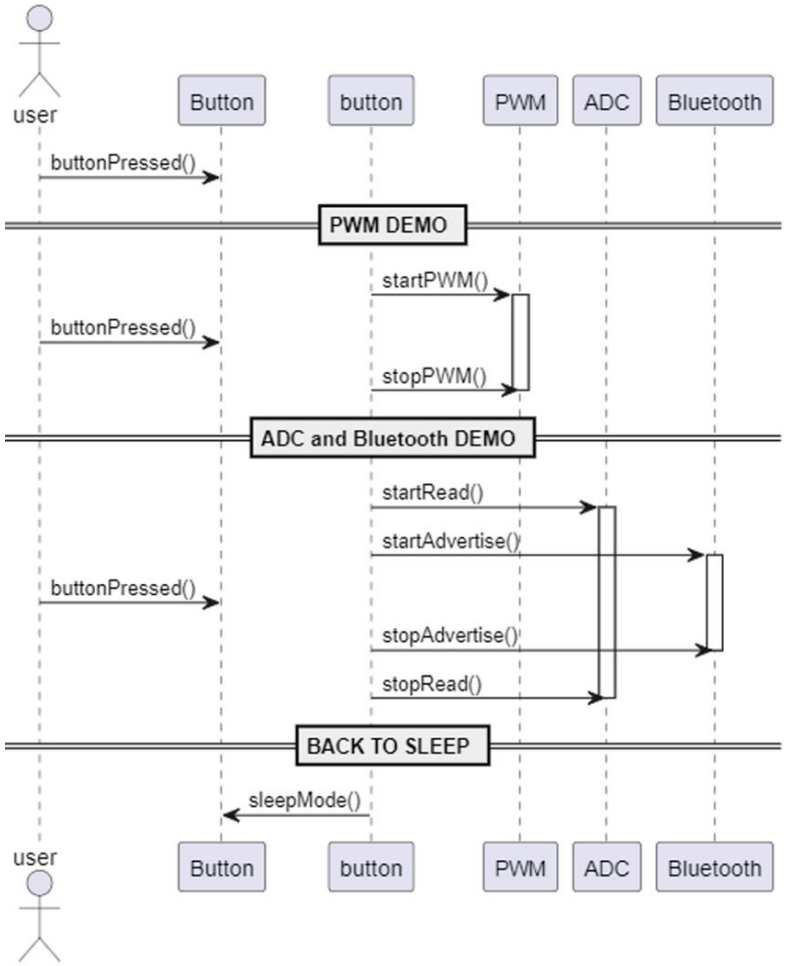


Figure 43: Sequence diagram of demonstration

9 Conclusion

Working with a recent Software Development Kit brings its own set of difficulties. Indeed, the transitions between new versions of the nRF Connect SDK on a Zephyr RTOS basis complicate the use of the programming environment. It is interesting to note the number of tickets still open with Nordic and Zephyr on the Devzone.

Thanks to a significant investment in time and perseverance, the different problematic meet while developing have been managed and the company Ypsomed has a stable basis and set of samples for further development.

The state of the art has allowed to understand the evolution of the devices offered by Nordic and the difference between the two SDK provided. My tutorial delivered allow any users to install the developing environment as well as understand the creation of project using west tool to manage repository and nRF Connect version.

The different measurements carried out with the analog discovery allowed to determine the response time of the nRF52840 under Zephyr RTOS, 21us. Due to lack of time, the sending advertise packet time with Bluetooth between two devices is missing.

Each example provided was built using an object-oriented C pattern. A total of 5 examples were developed so that the company Ypsomed could have a demonstration and implementation of several features that are useful in the development of future products or applications: Broadcaster, Observer, ADC with nrfx, ADC with Zephyr and PWM with Zephyr.

Developing with new technologies is often time consuming. Sometimes you encounter new problems that you have never seen before or you have less documentation than for other environments tools. One example encountered during the implementation of the observer sample concerns the method of filtering all devices that are advertising. I could provide my own solution to resolve this problem.

To conclude, working with a nRF device under Zephyr has real prospects for the future. In addition to working with low power devices using RTOS, the software will continue to be improved and more features to be added. A next step would be to test advertising extended mode to see any improvements in Bluetooth Low Energy communication.

Signature

Valérie Pompili

Savièse, 19.08.2022

Annexe
Annexe A: BT Specification V1

HES-SO Valais

SYND	ETE	TEVI
X	X	X

Données du travail de diplôme
Aufgabenstellung der Bachelorarbeit

FO 1.2.02.07.EB
che/31/05/2021

Filière / *Studiengang*	Année académique / *Studienjahr*	No TB / *Nr. BA*
SYND	**2021-22**	**IT/2022/79**
Mandant / *Auftraggeber* ☐ HES—SO Valais ☒ Industrie ☐ Etablissement partenaire *Partnerinstitution*	Etudiant / *Student* **Valérie Pompili** Professeur / *Dozent* **Medard Rieder**	Lieu d'exécution / *Ausführungsort* ☒ HES—SO Valais ☐ Industrie ☐ Etablissement partenaire *Partnerinstitution*
Travail confidentiel / *vertrauliche Arbeit* ☐ oui / ja ☒ non / nein	Expert / *Experte (données complètes)* **Mangold Stefan** - stefan.mangold@ypsomed.com Ypsomed AG, Brunnmattstrasse 6, 3401 Burgdorf	

Titre / *Titel* **Performance Analysis of Nordic NRF under Zephyr**

Description / Goal:

This Bachelor thesis intents to establish a collection of sample projects that are useful to establish a performance profile of the NRF52xxx (maybe the NRF53xxx) Bluetooth device family. The thesis is embedded into a medical context given by Ypsomed AG. Another constraint is given by the use of the Zephyr RTOS.

Tasks:

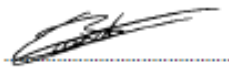
The following tasks must be solved for this thesis:

- Interrupt latency study: An IO pin is interrupted by a signal generator. The interruptions are transferred via Bluetooth to a host device, typically to a PC with a Nordic BTLE dongle.
- Interrupt priority study: Document and test the interrupt priority features under Zephyr / Nordic SDK.
- ADC performance study: An analog signal is sampled by the onboard ADC. The samples are transferred via Bluetooth to a host device.
- PWM Generation: Generation of a high frequency PWM signal to e.g. drive a motor driver.
- Bandwidth study: Measurement of the BTLE bandwidth under Zephyr.
- Extended Advertising study: Study and test of the "extended Advertising" feature of BT5 under Zephyr.
- FOTA / DFU study: Implement a sample FOTA / DFU scenario under Zephyr.

All these tasks are implemented under the NRF Connect environment using an NRF53xxx development kit. Each task must be entirely documented.

Deliverables:

- Any source code.
- Technical report
- Presentation slides

Signature ou visa / *Unterschrift oder Visum*	Délais / *Termine*
Responsable de l'orientation / *Leiter der Vertiefungsrichtung:*  ..... [1] Etudiant / Student : *Valéria Pompili*	Attribution du thème / *Ausgabe des Auftrags:* **16.05.2022** Présentation intermédiaire / *Zwischenpräsentation:* **20-21.06.2022** Remise du rapport final / *Abgabe des Schlussberichts:* **19.08.22, 12:00** Expositions / *Ausstellungen der Diplomarbeiten:* **24-26.08.2022** Défense orale / *Mündliche Verfechtung:* **Semaine/Woche 36 (5-9.09.2022)**

[1] *Par sa signature, l'étudiant-e s'engage à respecter strictement la directive DI.1.2.02.07 liée au travail de diplôme. Durch seine Unterschrift verpflichtet sich der/die Student/in, sich an die Richtlinie DI.1.2.02.07 der Diplomarbeit zu halten.*

Rapport reçu le / *Schlussbericht erhalten am* ..... Visa du secrétariat / *Visum des Sekretariats* .....

Annexe B: BT Specification V2

Priority I

- Interrupt latency study: An IO pin is interrupted by a signal generator.

  - Measure output pin on / output pin off latency
  - Measure the latency from "signal on input pin to signal in ISR
  - Measure the latency from signal on input pin to signal in output
  - Measure latency signal on input pin to signal on output pin with BT transfer in between in advertise mode.
  - Measure latency signal on input pin to signal on output pin with BT transfer in between in connected mode

  For the latter two measurements, the transfer is between two Nordic devices. Make them depend on BT parameters as advertise interval or connection interval. Also give some indications on energy consumption.

  Maybe, if the time allows, make this transfer arrive in a mobile phone with Android or IOS.

- Interrupt priority study
  - Answer this question: Can the advertise be started and stopped as needed in order to not be interrupted by the BT stack while waiting for other important not BT originated interrupts.
  - Answer also this question: Is it possible to not be interrupted inside a connection interval when there is not BT communication. This assumes a connection with another device and the possibility to know if the BT stack has sent the connection interval packet.

- ADC performance study: An analog signal is sampled by the onboard ADC. Characterize the ADC. Voltage range, precision, speed, number of channels...

- PWM Generation: Generation of a high frequency PWM signal to e.g. drive a motor driver.

Priority II

- Extended Advertising study: Study and test of the "extended Advertising" feature of BT5 under Zephyr. But only, if there is another device that can see the advanced advertising. Maybe include in this study the different normal advertise modes and their capabilities.

- FOTA / DFU study: Implement a sample FOTA / DFU scenario under Zephyr. Special feature: No fallback. This means there is no split partition. If the FOTA goes wrong, it must still be possible to connect via Bluetooth to the bootloader and to restart the FOTA.

List of Figures

Figure 1: nRF5280 DK board....10
Figure 2: Layers of nRF Connect SDK....12
Figure 3: How code base is synchronizes....13
Figure 4: nRF Connect SDK tools and configuration methods....14
Figure 5: Manage source code and configuration....14
Figure 6: Toolchain Manager, select nRF Version....15
Figure 7: Contain of .code-worspace file....16
Figure 8: West workspace....16
Figure 9: Create a new application....17
Figure 10: Create a new project with workspace....18
Figure 11: Correction to add for west.yml....19
Figure 12: How to open command prompt....19
Figure 13: How to update West....20
Figure 14: How to open workspace....21
Figure 15: How to add an existing application....21
Figure 16: Zephyr Kernel....22
Figure 17: State diagram of thread....23
Figure 18: Study of time reaction....24
Figure 19: Interrupt latency of Cortex-M processors....25
Figure 20: Test environment....26
Figure 21: List of all method test to toggle an output pin....27
Figure 22: Measure process time....27
Figure 23: Results of test 1 for interrupt latency with nRF52840....28
Figure 24: Code to measure response time....29
Figure 25: Toggle a pin after creating external interrupt....29
Figure 26: Measure value for response time and interrupt latency....30
Figure 27: BLE Stack Protocol....31
Figure 28: Frequency Spectrum of BLE (legacy mode)....33
Figure 29: Advertise type method....33
Figure 30: advertising parameter....34
Figure 31: Architecture of advertise packet....35
Figure 32: Construction of the advertise packet....35
Figure 33: Scan parameter....36
Figure 34: Scan window, scan interval, advertise interval....36
Figure 35: Method to parse data from advertise packet....37
Figure 36:Scheme of ADC....38
Figure 37: Set configuration for saadc channel....40
Figure 38: ADC configuration (Zephyr version)....41
Figure 39: Representation of PWM....43
Figure 40: PWM Module....43
Figure 41: PWM frequency....44
Figure 42: Calibration of maximal value for PWM....45
Figure 43: Sequence diagram of demonstration....46

List of Tables

Tableau 1: Pin configuration for GPIO 10
Tableau 2: nRF52840 DK specifications 11
Tableau 3: Different role of BLE available 32
Tableau 4: Analog pin of the nRF52840 DK 38