# Introduction

The bachelor’s Thesis is realised in collaboration with HES-SO Valais/Wallis, Nordic Semiconductor and NTNU. All the project is developed at Nordic Semiconductor in Trondheim, Norway.

Nordic Semiconductor is specialized in the development of SoC and provide a large range of tool to develop application using Bluetooth. Therefore, The Bluetooth Low Energy, BLE, is the main element of this project.

The main idea of this project is to measure the behaviour of Zephyr RTOS with nRF5x series SoC from Nordic Semiconductor. Zephyr is a recent RTOS developed for IoT application and provides a Bluetooth API.

Generally, the Nordic’s customers use the SoC with Bare Metal system and using libraries, SoftDevice and Software development kit, developed by Nordic Semiconductor. But, Nordic Semiconductor is interested by the possibility of using a RTOS with its products.

However, it is important for Nordic Semiconductor that the requirements to use Zephyr RTOS and the Bluetooth Low Energy with high traffic are respected and to compare the performances with a Bare Metal system using the SoftDevice.

It is not a question of which one is a best but to analyse if Zephyr RTOS is a good solution for tiny embedded system using Bluetooth Low Energy.

Before stating this project, I had no idea about the Bluetooth Low Energy and I never used the SoftDevice, the Software Development Kit and Zephyr RTOS. Hence, it was interesting to see which systems were the most easier to learn and to implement.

# Why Zephyr RTOS?

Here is the description from Zephyr Project GitHub Repository.

*“The Zephyr™* *Project is a scalable real-time operating system (RTOS) supporting multiple hardware architectures, optimized for resource constrained devices, and built with security in mind.*

*The Zephyr OS is based on a small-footprint kernel designed for use on resource-constrained systems: from simple embedded environmental sensors and LED wearables to sophisticated smart watches and IoT wireless gateways.*

*The Zephyr kernel supports multiple architectures, including ARM Cortex-M, Intel x86, ARC, NIOS II and RISC V, and a large number of* [*supported boards*](https://www.zephyrproject.org/doc/boards/boards.html)*.”*



Figure : Zephyr RTOS logo

## Philosophy

Zephyr is Apache 2.0 licensed and has a fully open development model. The terms and conditions of this license are mainly:

* all copies, modified or unmodified, are accompanied by a copy of the licence
* all modifications are clearly marked as being the work of the modifier
* all notices of copyright, trademark and patent rights are reproduced accurately in distributed copies
* the licensee does not use any trademarks that belong to the licensor

Hence, it is possible to anyone to access to the source code and to modify it as he wants regards to the license’s conditions.

In addition of the Apache license, Zephyr Project source is maintained on a public GitHub Repository. GitHub allow anyone to be a contributor of the project by notifying bugs or by proposing improvements.

To contribute to the project, a programmer pulls a request for a bug or improvement that will be analysed by one or several main developers of the project. If the request is approved, the code of the request is merge with the source code of the project.

## Benefits for Nordic Semiconductor

Nordic is currently member of the Zephyr RTOS. Therefore, it is already possible to use Zephyr RTOS with nRF5x series SoC. However, it is already possible to use different RTOS as FreeRTOS or Keil with Nordic’s produces. So, why is it interesting for Nordic Semiconductor?

At first, Zephyr RTOS provides all the tools to develop a IoT application with Bluetooth, Networking, I/O drivers API.

Secondly, Zephyr RTOS supports multiple hardware architectures and it allows to easily port an application on new nRF5x series release without changing the Nordic’s SoftDevices and SDK version.

Then, Zephyr RTOS is an Open Source Project. Hence, it is possible to adapt easily the RTOS and to add tools for a specific project and to contribute to the project.

The last reason concerns the memory. The nRF5x series allows to use complex RTOS that require more RAM and Flash memory as Zephyr. More, Zephyr is a fusion between the SDK and the SoftDevice and therefore saves more Flash and RAM memory.

|  |  |  |
| --- | --- | --- |
|  | SoftDevice | Zephyr RTOS |
| Flash | 136kBytes | 47kBytes |
| RAM | 8kBytes | 14kBytes |

Table : Memory requirement comparison

Note: The value used for the SoftDevice come from the examples of the SDK V13.0.0

## My Contribution

As Zephyr is a recent project, some tools, useful for my project, were not already provided by Zephyr. Therefore, I contributed to the Zephyr project during my Bachelor’s Thesis. My contribution is:

* Fix a bug on the nRF5x SPI driver configuration

Because of a lack of time, I had no time to pull request during the project. But it is planned to pull a request for the element I added for my project:

* nRF5840 GPIO Port 1 driver
* nRF5840 SPI2 driver
* Possibility to use the SPI and I2C on GPIO Port 1

## Sources

<https://www.zephyrproject.org/>

<https://github.com/zephyrproject-rtos/zephyr>

<https://nexus.zephyrproject.org/content/sites/site/org.zephyrproject.zephyr/dev/api/api.html>

<https://devzone.nordicsemi.com/blogs/1059/nrf5x-support-within-the-zephyr-project-rtos/>

<https://www.apache.org/licenses/LICENSE-2.0>

<http://oss-watch.ac.uk/resources/apache2>

# Specifications

To measure the behaviour of Zephyr RTOS and SoftDevice, it is important to define the use cases. As the Bluetooth Low Energy is at the centre of this project, the uses cases are defined by the BLE:

* **Peripheral**, one or several devices send data to a central
* **Central**, receive the data from peripheral and maintain the connection time

To realise the Bachelor’s Thesis, some materials are provided:

* **Nordic nRF52840 Development Kit** to implement the BLE peripheral and central
* **Nordic Power Profiler Kit** to measure the power consumption

The performances of the peripheral are tested with one central and one peripheral connected. As the BLE peripheral is the system that acquire data from sensor, an extension board is developed with different components:

* **A/D converter**
* **Accelerometer**
* **Interrupt generator**

Those devices allow to test the processor in real circumstances and to generate interruptions to stress the system. The frequency of the interruptions is incremented until the system crash. It is important that when system crash, it is still able to maintain the Bluetooth connection.

The peripheral must be able to maintain the connection with the central when the throughputs of the sensors and the BLE connection requirements are fast.

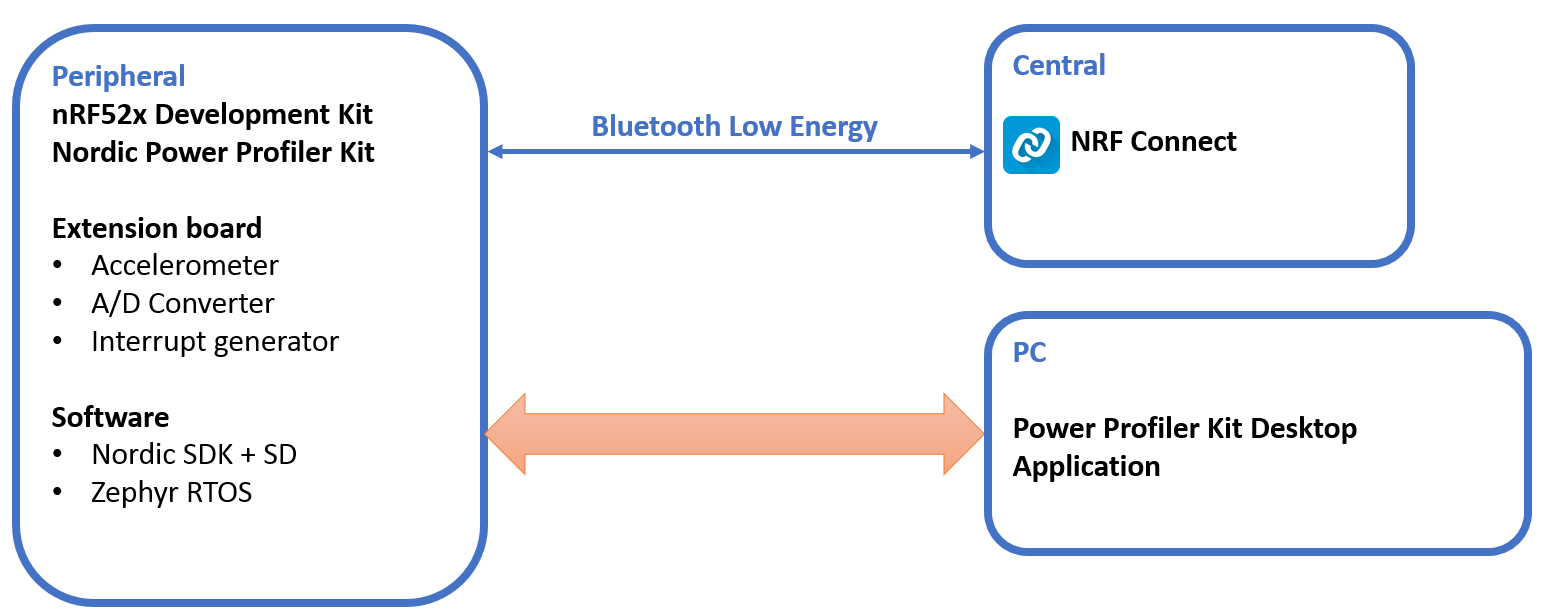


Figure : Behaviour measurement on BLE peripheral Schema Block

The performances of the peripheral are tested with one central and several peripherals connected. The central must be able to maintain the connection with all the peripherals when the BLE connection requirements are fast.

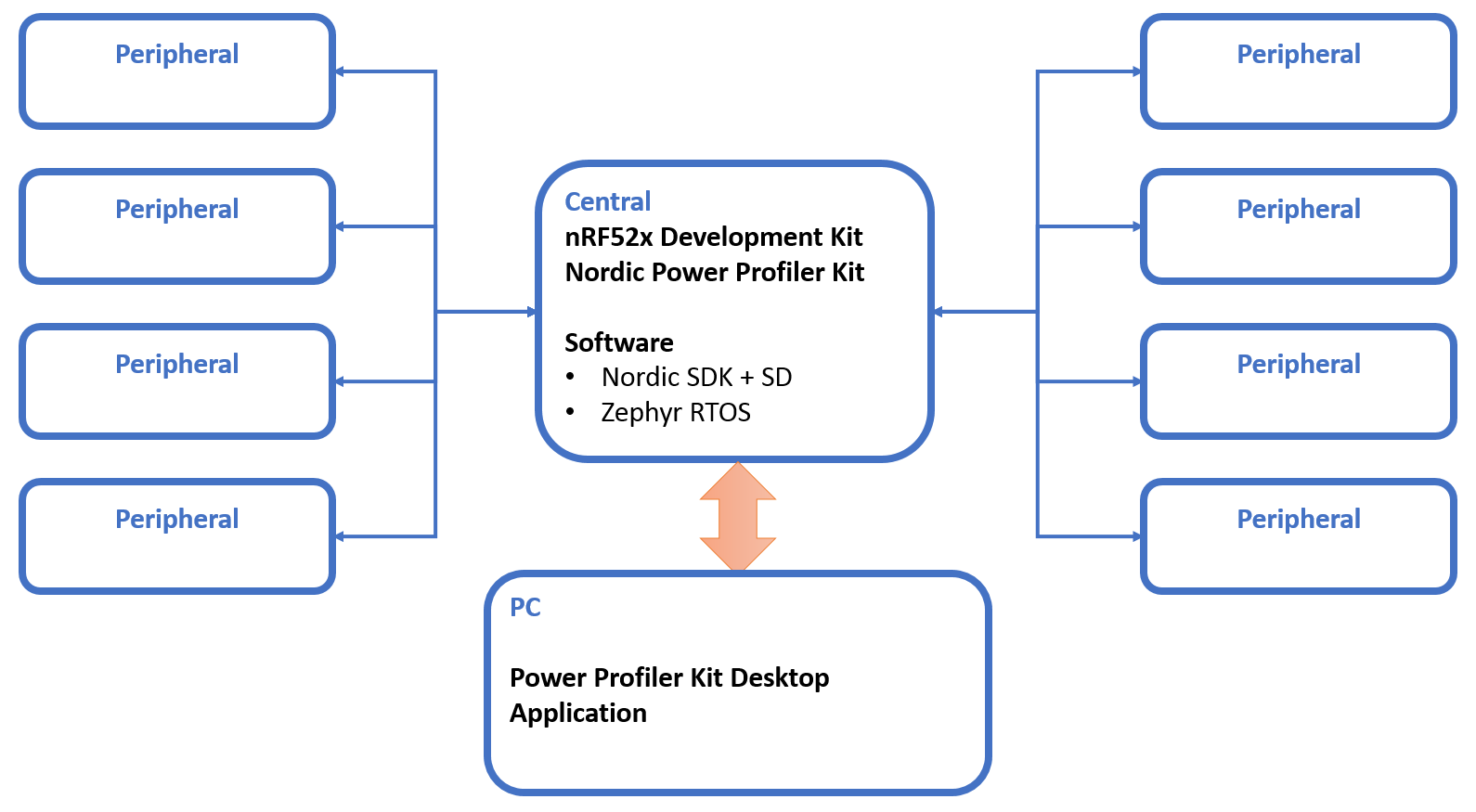


Figure :Behaviour measurement on BLE Central Schema Block

Then, the following performance are measured:

* **Power Consumption**
* **Interrupt latency**
* **Bluetooth Low Energy performance**

Finally, the measurements are compared. The results must not define which system is the best. This is only a representation of the performance of Zephyr RTOS and if this system is a good solution for tiny embedded system using Bluetooth Low Energy.

# Bluetooth Low Energy

## Introduction

## Requirement

## Test cases

## Sources

# Hardware

The Hardware is separate in six parts:

* **nRF52840 SoC**
* **Micro USB-B** to communicate with a PC and programme the chip
* **Power supply** provided by the Micro USB-B
* **Nordic Power Profiler** **Kit (PPK)** to measure the power consumption of the chip only
* **Connector interface** to connect the extension board, Power Profiler Kit and nRF52840 DK
* **Extension Board** to provide data to the chip

The extension board is the only part that is developed for this project. The Power supply 3V, the Micro USB-B and the nRF52840 SoC are on the nRF52840 Development Kit.

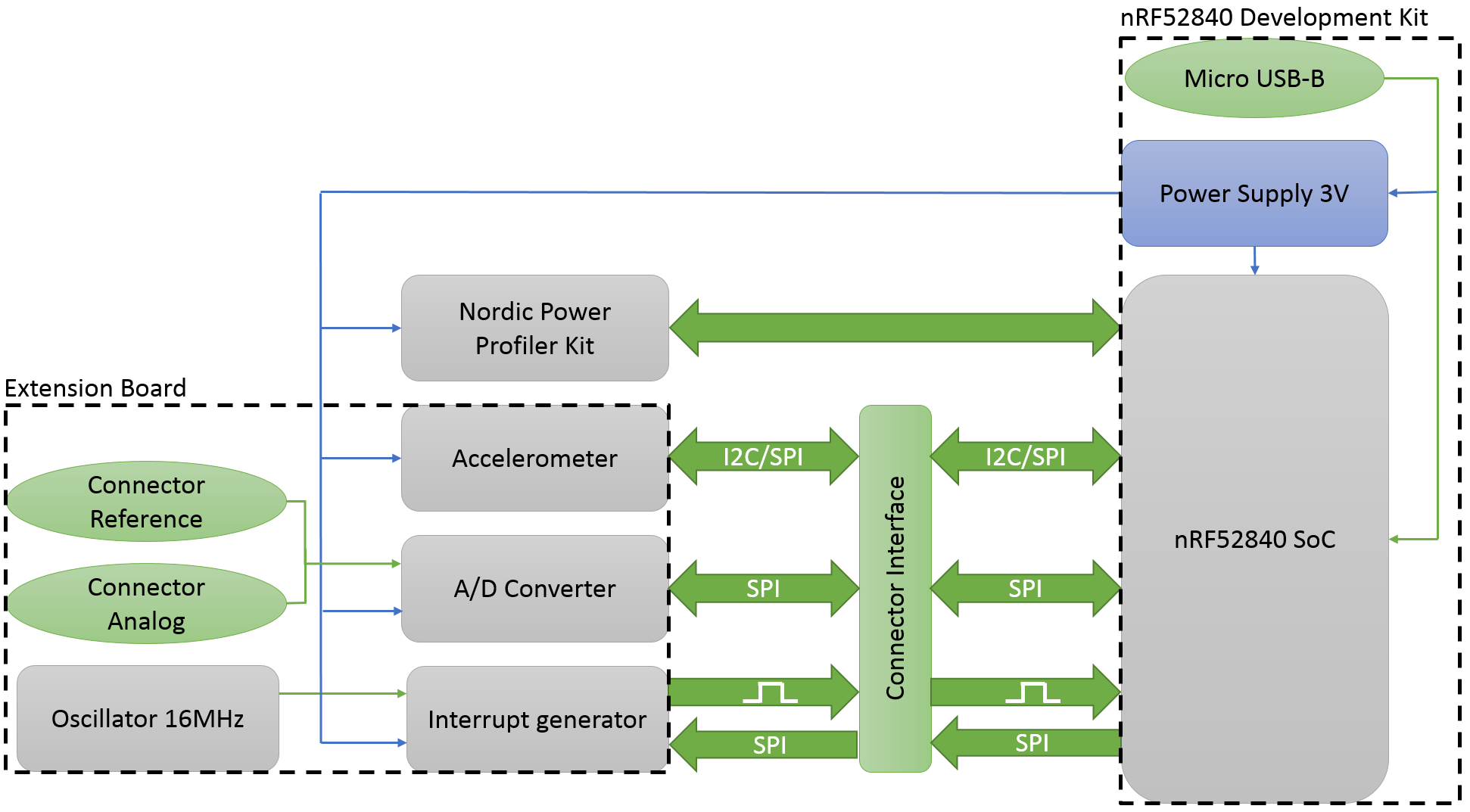


Figure : Hardware schema block

## nRF52840 SoC

The SoC used is the nRF52840. Here is the description from Nordic Semiconductor website.

*“The nRF52840 is an advanced multi-protocol SoC ideally suited for ultra-low power wireless applications. The nRF52840 SoC is built around a 32-bit ARM® Cortex™-M4F CPU with 1MB flash and 256kB RAM on chip. The embedded 2.4GHz transceiver supports Bluetooth® low energy (*[*Bluetooth 5*](https://www.nordicsemi.com/eng/Products/Bluetooth-5)*),* [*802.15.4*](https://www.nordicsemi.com/eng/Products/IEEE-802.15.4-Thread)*, ANT and proprietary 2.4GHz protocols. It is on-air compatible with existing nRF52 Series, nRF51 Series, and nRF24 Series products from Nordic Semiconductor.”*

In addition of the description above, those features are used for the project:

* **1.7 to 5.5V** power supply
* **80mA** current consumption max
* **PPI** – Programmable Peripheral Interconnect
* **48 x GPIOs** (32 x GPIOs PORT0, 16 x GPIOs PORT1)
* **2 x I2C** (100kHz, 250kHz, 400kHz)
* **4 x SPI** (125kHz, 250kHz, 500kHz, 1MHz, 2MHz, 4MHz, 8MHz, 16MHz, 32MHz)

There no important reason to use this processor. But the nRF52840 is the SoC that provides the largest number of features. Hence, it will avoid being limited with the measurements because of the capacity of the SoC.

## Micro USB-B

The connector Micro USB-B is used for different purposes:

* **To programme** the nRF52840 Chip
* **To provide 5V power supply** to the system
* **To Debug** the nRF52840 Chip
* **To transfer the data measured** with SEGGER Real Time Transfer

The connection SEGGER Real Time Transfer (RTT) is used by the Nordic Power Profiler Kit to transfer the data to a PC.

## Power Supply

The power supply transforms the power supply of 5V, provided by the Micro USB-B, to 3V to supply all the system. To transforms the power supply, there is a fixed 3V buck regulator and one voltage follower regulator on the nRF52840 DK.

Due to the low consumption of the system, the power supply of the nRF52840 DK is far enough to provide power to all the system.

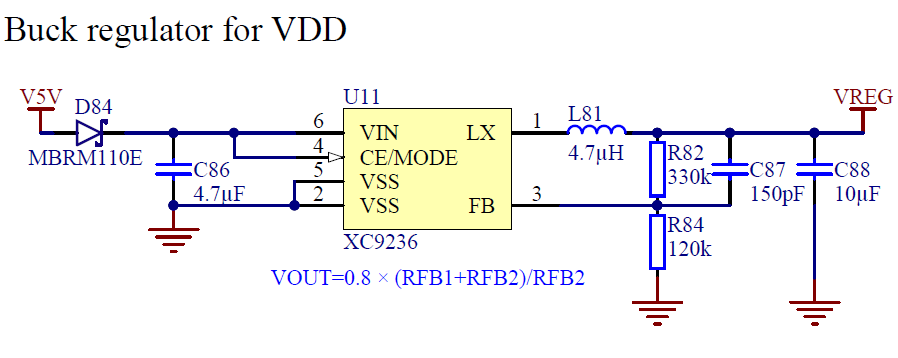


Figure : Regulator buck 3V schematic, from nRF52840 DK User Guide

## Connector Interface

The connector interface is defined by the nRF52840 development kit and is almost the same for all the Nordic’s DK.

The connector allows:

* **To supply the extension board and the Power Profiler Kit**
* **To measure the power consumption** of a nRF5x SoC
* **To access to the GPIOs** of a nRF5x SoC

The extension board and the Power Profiler Kit are plugged on the nRF52840 DK with the connector interface that allow an easy connection of the different part of the system.

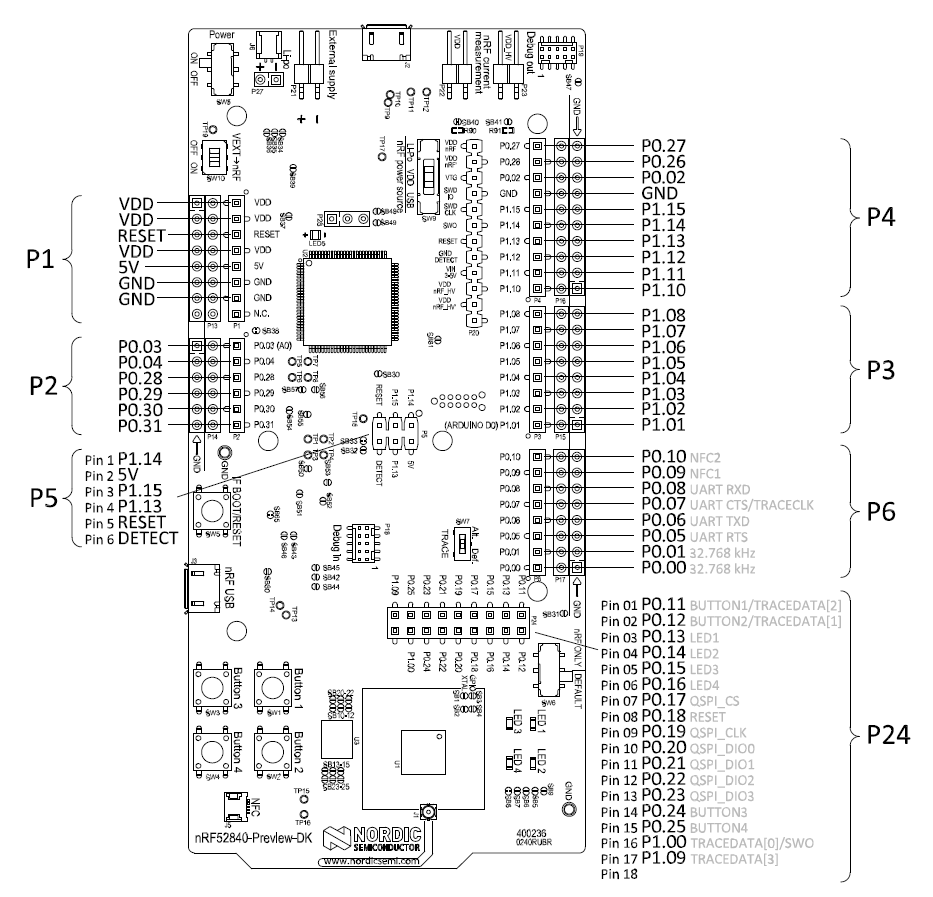


Figure : nRF52840 Preview DK board connectors, from nRF52840 DK User Guide

## Nordic Power Profiler Kit

The Nordic Power Profiler Kit is an easy use tool for the measurement and power consumption optimization of embedded solutions. It provides the following features:

* **1µA-70mA current measurement range**
* **0.2µA measurement resolution**
* **77kHz sampling rate**
* **Desktop application** in python allowing customization

The Nordic Power Profiler Kit is the best solution because it allows to measure only the power consumption of the nRF5x chip.

The Desktop application communicate with the PPK using the SEGGER Real Time Transfer of the nRF5x Chip. This connection can be used to transmit other measurements via RTT. The Desktop application can be modified to display those measurements.

## Extension Board

The extension board is the single part that the hardware is developed. It is connected to the nRF52840 DK using the interface connector.

The purpose of the extension board is to use the nRF52840 Chip under real conditions of use. To do that, the extension board provide the different elements.

* **A/D converter** that can be connected to a generator function or an external sensor.
* **Accelerometer**
* **Interrupt generator**

All components as some general criterion to ease the order:

* **Package easy to solder**
* **Same provider**

### A/D Converter

The A/D Converter provides a large quantity of data that the chip must be able to deal with no loss. As the large quantity of data to stress the chip, the way to get the data must stressful as well.

The component used is the Delta-Sigma ADC **MAX11200** that provides the following features:

* **24 bits Resolution and 8 bits register’s address**
* **2.7 to 3.6V** power supply
* **300µA** currentconsumption max
* **Reference Voltage**
* **SPI (SCL max 5MHz)** to calibrate and get the data
* **Ultra-low-power** with power-down mode

An ADC communicating with SPI is chosen due to the requirement to communicate fast.

The Analog input can be provided by a function generator (Connector BNC 50Ω) or by an external analogue sensor (Pin 2x1). Two resistances can be soldered to use the ADC in current loop system.

The Reference voltage can be the power supply voltage or an external reference if the external sensor has specific requirement.

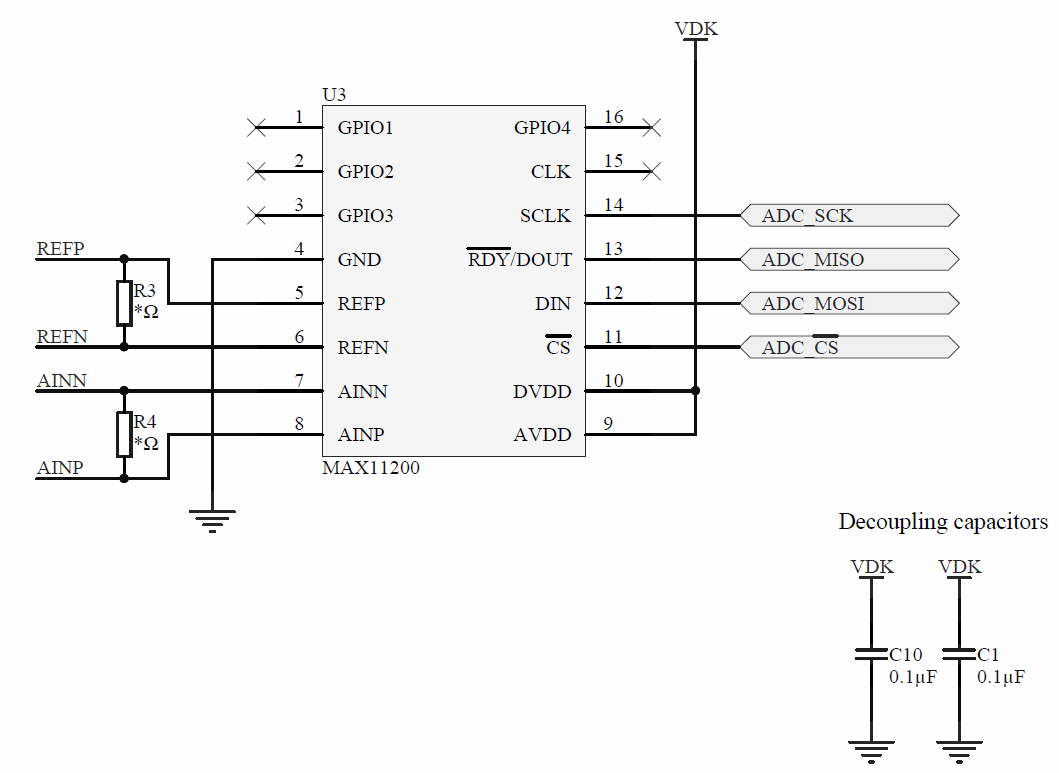


Figure :MAX11200 schematic

#### *Throughput*

The A/D Converter can be used in Continuous Conversion or Single -Cycle Conversion. For each mode, an oscillator intern can be selected, 2.4576MHz or 2.048MHz, to determine the data rate. Only the Continuous conversion is used.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Oscillator |  |  |  |  |  |
| 2.4576MHz | 60 | 120 | 240 | 480 | [sps] |
| 2.048MHz | 50 | 100 | 200 | 400 | [sps] |

Table : A/D Converter Data Rate in sample per second

The resolution of the A/D Converter is 24bits and the register address size is 8bits.

The frequency max of the SPI is 5MHz but due to the SPI frequency provided by the nRF52840, the frequency max of the SPI is 4MHz.

The throughput of the A/D Converter is interesting because it is near to the minimum connection interval of the BLE.

### Accelerometer

As the A/D Converter, the accelerometer provides a large quantity of data that the chip must be able to deal with no loss.

The component used is the accelerometer **LIS3DH** that provides the following features:

* **16 bits Resolution**
* **3-axis**
* **±2g/±4g/±8g/±16g**
* **1.7 to 3.6V** power supply
* **185µA** currentconsumption max
* **FIFO 32-level 6 bytes**
* **I2C (SCL max 400kHz)/SPI (SCL max 10MHz)** to get the data
* **2 Interrupt pins** to notify when new data are available
* **Ultra-low-power** with automatic power-down mode

An accelerometer communicating with I2C is chosen to use different features of the chip. However, the SPI can be used as well.

The LIS3DH provide a FIFO to store data. This FIFO can be read at one time with a frame of 192 bytes, 2 bytes per axis. This frame provides a good test to stress the chip.

Other features of the LIS3DH are the Click and Free-fall detection that generate interruption. Those features are interesting because they can simulate real interruption from a sensor.

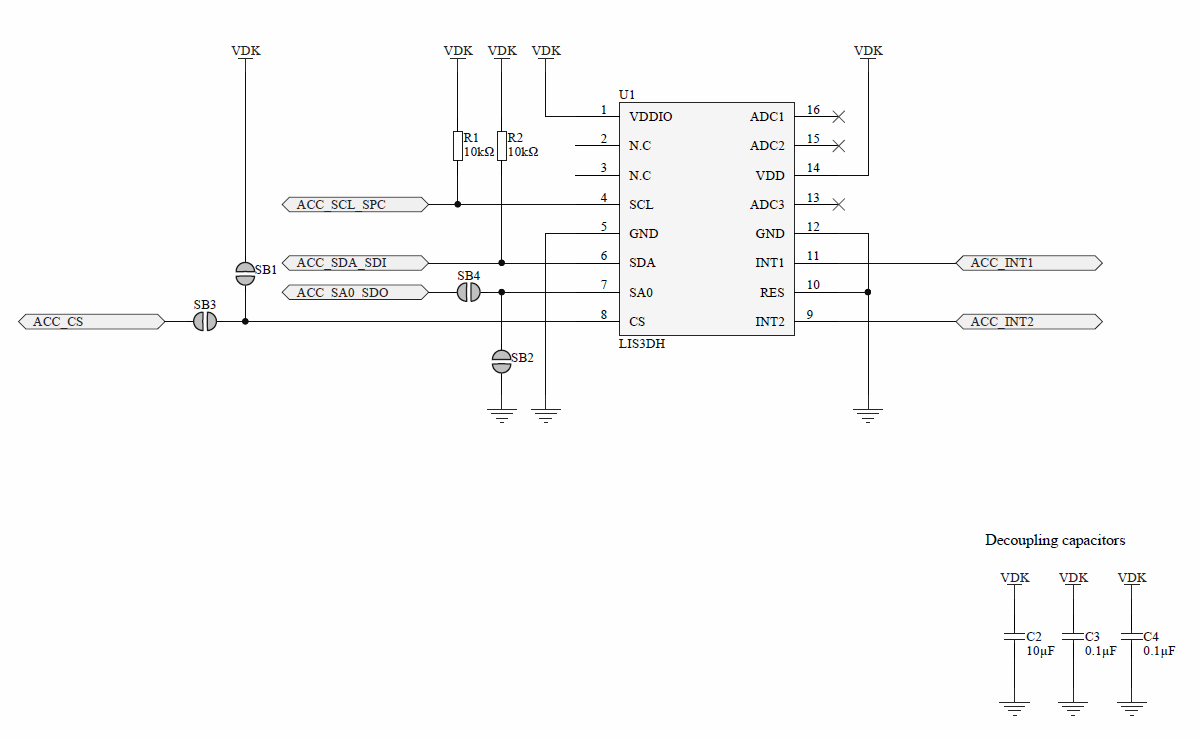


Figure :LIS3DH Schematic

#### *Throughput*

The accelerometer can perform measurements in different mode. For each mode, the data rate can slightly change. The LIS3DH provides a large range of data rate.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Mode |  |  |  |  |  |  |  |  |  |  |
| Low Power | 1 | 10 | 25 | 50 | 100 | 200 | 400 | 1.6k | 5.376k | [Hz] |
| Normal | 1 | 10 | 25 | 50 | 100 | 200 | 400 | 1.344k |  | [Hz] |
| High Resolution | 1 | 10 | 25 | 50 | 100 | 200 | 400 | 1.344k |  | [Hz] |

Table : Accelerometer Data Rate in Hertz

The resolution depends of the mode but it has no influence on the data frame.

* Low Power: 8bits
* Normal: 10bits
* High Power: 12bits

However, the accelerometer has a 32bytes FIFO for each byte of measurement, axis LSByte and axis MSByte. The register address size is 8 bits and the I2C requires 9 bits. More, the I2C requires a ACK bit between each byte transmitted.

The minimum data frame is calculated with the data of one axis. Then, the frequency max of the I2C is 400kHz.

The throughput of the Accelerometer is interesting due to the FIFO that create a long data frame and because it is near to the minimum connection interval of the BLE.

### Interrupt generator

The interrupt generator generates pulse that create interruptions in the programme. The period of interruptions can be easily changed to modify the test conditions.

The component used is the Programmable Waveform Generator **AD9837** that provides the following features:

* **16MHz Clock**
* **28 bits (0.06Hz)** **Resolutions**
* **2.3V to 5.5V power supply**
* **4.5mA** currentconsumption max
* **3 Wires SPI** to programme the waveform type and frequency
* **Low power** with power-down option

A 28Bits Register is used and programmable via SPI to calculate the frequency. The formula below defines the frequency:

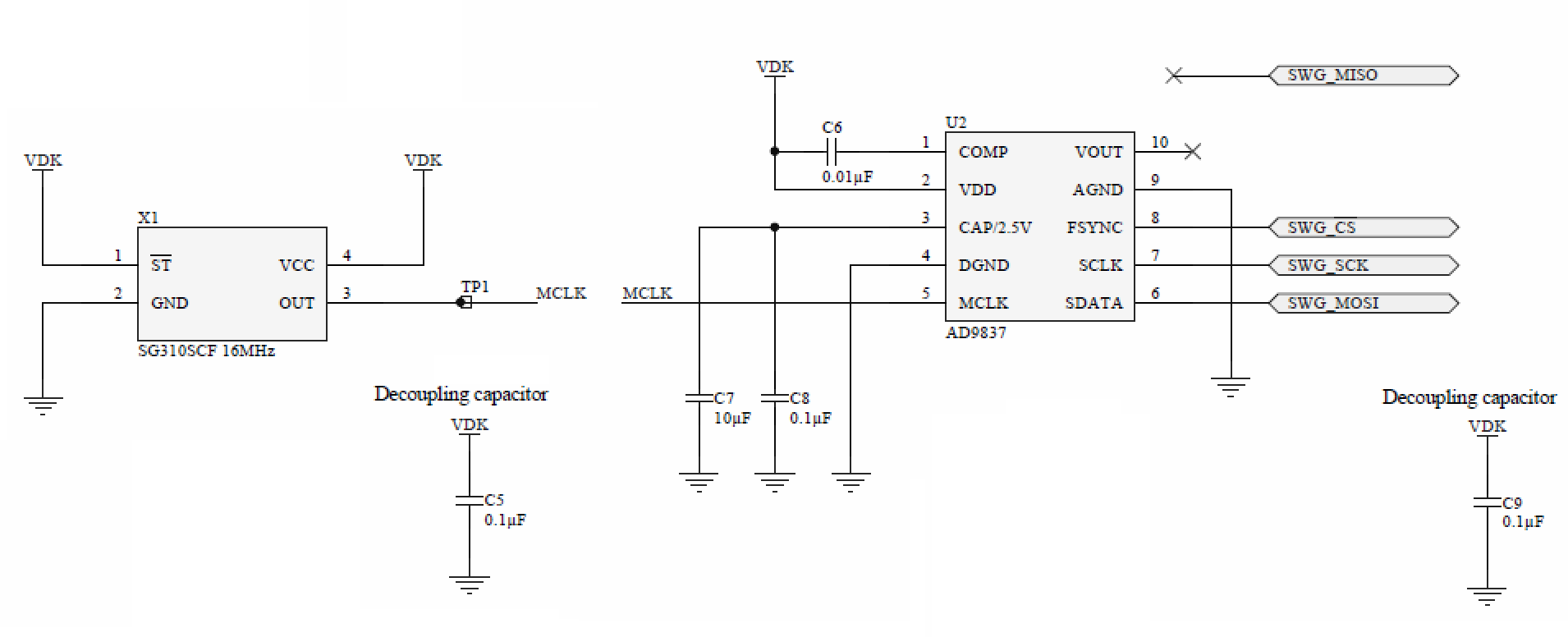


Figure : AD9837 schematics

## Annexes

* List of components
* Schematic
* PCB

## Sources

<https://www.nordicsemi.com/eng/Products/nRF52840>

<https://www.nordicsemi.com/eng/Products/nRF52840-Preview-DK>

<https://datasheets.maximintegrated.com/en/ds/MAX11200-MAX11210.pdf>

<http://www.st.com/content/ccc/resource/technical/document/datasheet/3c/ae/50/85/d6/b1/46/fe/CD00274221.pdf/files/CD00274221.pdf/jcr:content/translations/en.CD00274221.pdf>

<http://www.analog.com/media/en/technical-documentation/data-sheets/AD9837.PDF>

# Software

The Software is separated in four parts:

* **The Environment Layer**
* **The abstract Layer**
* **The Driver Layer**
* **The Application Layer**

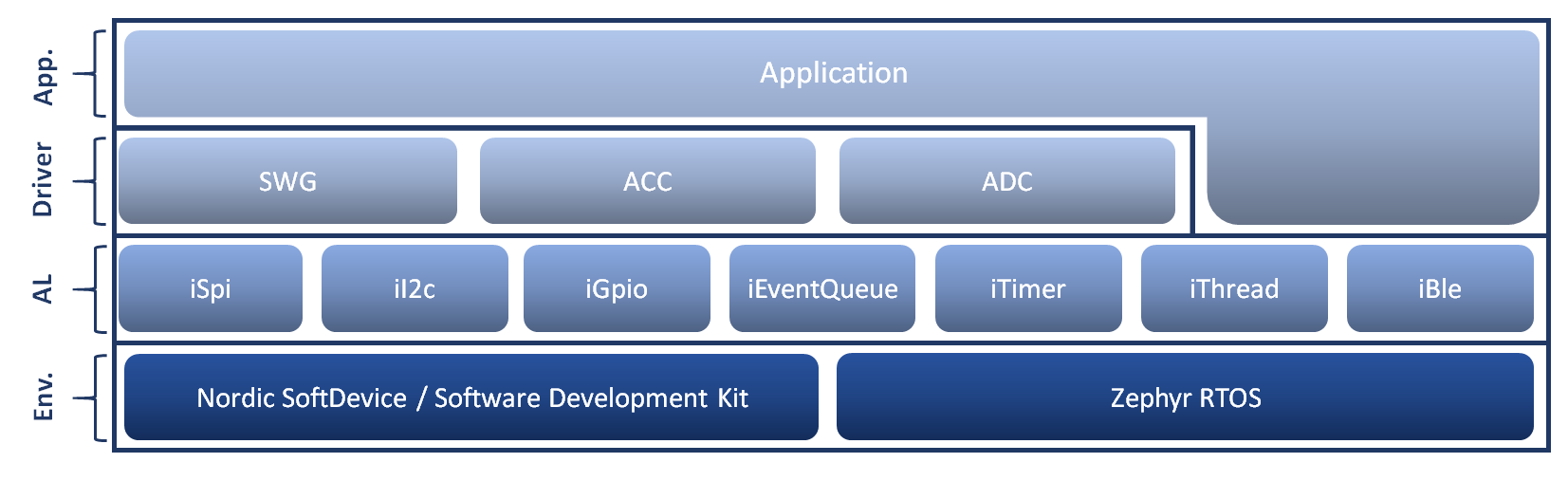


Figure : Architecture of the Software

## Environment Layer

The Environment Layer represents the different libraries and systems used. There are two elements, Nordic SD - SDK for the Bare Metal system and Zephyr RTOS.

#### Nordic

Nordic Semiconductor provides two libraries to help in the development of applications.

* **SoftDevice (SD)**, Bluetooth low energy (BLE) Central and Peripheral protocol stack
* **Software Development Kit (SDK)**, facilitate firmware development for different devices and applications

different version of the SD and SDK. As the nRF52840 is last SoC release, the version of the SD and SDK used for this project are the most recent.

The SoftDevice used is the S140 and the Software development kit used is nRF5 SDK v13.0.0. A new SDK version was release, v13.1.0, in the middle of the project but not used because the Abstract Layer was already developed.

#### Zephyr RTOS

Zephyr Project needs a Software Development Kit that contains all necessary tools and cross-compilers needed to build the kernel on all supported architectures. The version use for this project was SDK v0.9.1.

Zephyr Project is maintained on a public GitHub repository. Hence, the source code was frequently update to use the last version.

## Abstract Layer

The Abstract Layer interfaces the features of the systems used in the Environment Layer to use the same code for the Driver Layer and for the Application Layer.

iEventQueue and iThread developed by myself

## Driver Layer

The Driver Layer manages the communication with the device of the extension board. It allows to easily configure the device and acquire the data.

There is a driver for each components of the extension board:

* **SWG**, driver of the AD9837
* **ACC**, driver of the LIS3DH
* **ADC**, driver of the MAX11200

## Application Layer

The Application Layeruse all the below layer to create a peripheral and a central application.

#### Peripheral

UML to explain the sequence of the peripheral

Service BLE created

Thread created

#### Central

UML to explain the sequence of the central

Service BLE searched

Thread created

## Measurements Code

To perform the measurement, it is important that the code does not perturb the application in anyway. To limit the perturbation, the measurement code uses directly the register of the nRF52840 without any libraries.

The measurements performed are:

* Interrupt Latency
* Power Consumption
* BLE Radio state
* BLE Application → Controller stack propagation

### Interrupt Latency

UML to explain when the GPIO are enables

### Power Consumption

To measure correctly the power consumption, it is important to disable useless drivers in the configuration to not impact the results. It means to disable for:

**SD+SDK**

* SEGGER RTT

And for **Zephyr RTOS**

* UART driver
* Console driver

Unfortunately, no output on console is possible during the power consumption test to measure as much as possible the current consumption in real cases.

### Bluetooth Low energy

It is important to note a great difference between the SoftDevice and Zephyr RTOS.

* SoftDevice is confidential and no access is possible
* Zephyr RTOS is an open source project, hence all the source code is accessible.

This difference is important because, to compare the behaviour, the same tests must be performed and it is not possible to measure the BLE performance directly within the stack. It why the number of tests possible is limited. However, it is possible to use different peripherals of the nRF52840 to get some information.

#### BLE Radio State

The nRF52840’s Radio has a State Machine and be able to see the states of this radio is interesting because it allows to check if the device still sends or responses to the Bluetooth connection event.

BLE Always keep the connection

The operating state of the Radio is control via several events and task.

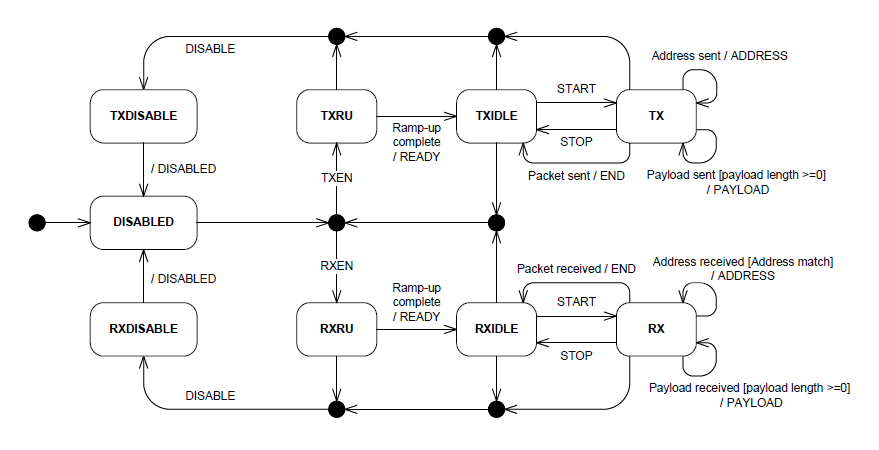


Figure : Bluetooth radio state of nRF52840, from nRF52840 Datasheet

The Radio state is measured using the following peripherals:

* Explain tasks and events of nRF5
* **PPI**, Programmable Peripheral Interconnect, enables peripherals to interact autonomously and it eliminates the need for CPU activity.
* **GPIOTE**, GPIO Tasks and Events, provides functionality for accessing GPIO pins using tasks and events
* **RADIO** transceiver used for the BLE transmission

The RADIO events are connected directly to the GPIO using PPI. The event used are:

* EVENTS\_TXREADY
* EVENTS\_RXREADY
* EVENTS\_END
* EVENTS\_ADDRESS
* EVENTS\_DISABLED
* EVENTS\_CRCOK

Events on which pin and an example of an output to explain how to analyse

It is then possible to see:

* When the device receives a packet
* When the device sends a packet
* When the CRC of the packet is correct

#### BLE Application → Controller Stack Propagation

Analysing the stack propagation is interesting because it allows to see how much time the application is locked to prepare BLE data.

The time to push notification and indication packets, for the peripheral, or read, write and CCD packets, for the central must be as short as possible to

Another interesting information is the time to push a. To put it another way, the time to push a packet within the BLE stack from the Application layer to the Controller Link Layer.

Figure of the stack with the layer involved

Events on which pin and an example of an output to explain how to analyse

UML to explain when the GPIO are enables

#### Other interesting BLE measurements

Because of a lake of time and the limitation regards the SoftDevice, some measurements:

Figure to summarize all the measurements with configuration and pins used

## Annexes

## Sources

<https://infocenter.nordicsemi.com/index.jsp>

<http://zephyr-docs.s3-website-us-east-1.amazonaws.com/online/dev/>

<https://github.com/zephyrproject-rtos/zephyr>

# Measurements

## Equipment

## Results

## Comparisons

# Conclusion