1 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 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 using libraries 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.

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 and which need a Real-Time Operating System.

Before stating this project, I had no idea about the Bluetooth Low Energy and I never used any SoC and libraries from Nordic Semiconductor or Zephyr RTOS. Hence, it was interesting to see which systems were the most easily to learn and to implement it.

Before talking about the specifications and the project in general, a small presentation of system used, Zephyr RTOS and SoftDevice/Software Development Kit from Nordic, is presented. Then, a theory on the base of the Bluetooth Low Energy is written to help for the rest of the reading.

2 SoftDevice and Software Development Kit

The SoftDevice, SD, and Software Development Kit, SDK, are libraries developed by Nordic Semiconductor for SoC of Nordic semiconductor.

Figure 1: Nordic Semiconductor Logo

2.1 SoftDevice

Here is the description form https://infocenter.nordicsemi.com/.

"A SoftDevice is a wireless protocol stack library for building System on Chip (SoC) solutions.

SoftDevices are precompiled into a binary image and functionally verified according to the wireless protocol specification, so that all you have to think about is creating the application. The unique hardware and software framework provide run-time memory protection, thread safety, and deterministic real-time behaviour… "

To resume, the SoftDevice is a library that provides an API to developed Bluetooth Low Energy Application. The library is already compiled and there is no way to access to the source code that is confidential.

The SoftDevice is an advantage because is easier the use of the peripherals Radio of a Nordic SoC and the way to create your wireless application.

However, there are several versions of the SoftDevice and each version is not necessarily compatible with all SoC from Nordic Semiconductor.

2.2 Nordic Software Development Kit

The Nordic Software Development is simply by Software Development Kit SDK in this report. Here is the description form https://www.nordicsemi.com/.

"The nRF5 SDK provides a rich and well tested software development environment for the nRF51 Series and nRF52 Series devices. […]

It includes a broad selection of drivers, libraries, examples for peripherals, SoftDevices, and proprietary radio protocols of the nRF51 Series and nRF52 Series."

Like the SoftDevice, the SDK make easier to create application with different peripherals of Nordic SoC. But, as well, there are several versions of the SDK and like the SoftDevice, each version is not necessarily compatible.

In an application, the SoftDevice is flashed in another part of the memory whereas the Software Development Kit is completely a part of the application.

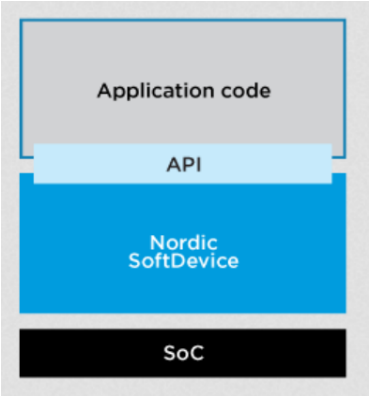


Figure 2: SoftDevice with an application in a Soc, from https://www.nordicsemi.com/

Nordic semiconductor already provides its SoftDevice and Software Development Kit to help to development application with Nordic SoC. Therefore, why using Zephyr RTOS?

2.3 Source

https://infocenter.nordicsemi.com/index.jsp
https://www.nordicsemi.com/eng/Products/Bluetooth-low-energy/nRF5-SDK
https://www.nordicsemi.com/Products/nRF51-Series-SoC

3 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."

Figure 3: Zephyr RTOS logo

3.1 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.

3.2 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 is interested because it provides all the tools to develop a IoT application with Bluetooth, Networking, I/O Drivers API with an Operating System. When a customer wants to use an RTOS, he need to use it in addition of the SoftDevice. However, with Zephyr RTOS, everything is in on system.

Thirdly, 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 ROM and RAM memory.

Put the memory requirement of my own application

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

Table 1: Comparison of memory requirement

Note: The value for the SoftDevice are the value indicated in the examples of the nRF5 SDK v13.0.0.

3.3 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
- Report a problem with the function to enter in IDLE mode
- Report a problem in the connection with nRF-Connect -> lose time

Because of a lack of time, I had no time to pull other requests during the project. But it is planned to pull requests 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

Despite I contributed a little bit to Zephyr RTOS, all those corrections and modifications make me waste time.

3.4 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

4 Bluetooth Low Energy

This chapter only gives a small introduction of the Bluetooth Low Energy. Therefore, the explanations are really simplified.

4.1 Introduction

The Bluetooth Low Energy (BLE) or Bluetooth smart was introduced as part of the Bluetooth 4.0 core specification in 2010. While there is some overlap with classic Bluetooth, there is a difference between Bluetooth and Bluetooth Low Energy.

Figure 4: Bluetooth Smart Logo

Like Bluetooth, BLE operates in the 2.4GHz ISM band. However, in contrast to classic Bluetooth, BLE is designed to provide significantly lower power consumption by remaining sleeping except for when a connection event is initiated. The connection event is explained at chapter **4.3.3.1 Connection Interval**.

On the other hand, Bluetooth can handle a lot of data, e.g. video or audio, but consumes battery life quickly and costs a lot more. BLE is used for applications that do not need to exchange large amounts of data, e.g. sensors, and can therefore run on battery power for years at a cheaper cost.

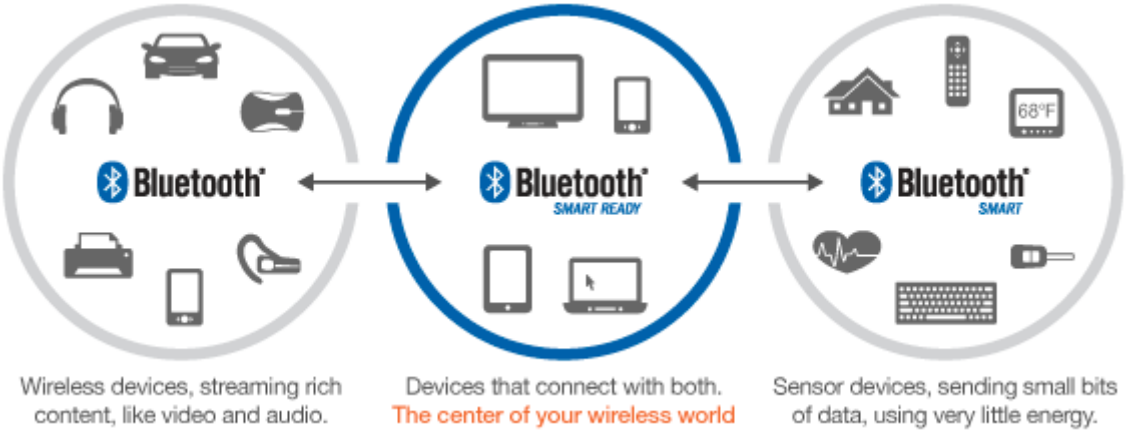


Figure 5: Uses of Classic Bluetooth and BLE, from https://www.bluetooth.org/

The BLE core 4.0 specification can be download at https://www.google.no/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0ahUKEwjY44ioy8XVAhXCb1AKHROJBYYQFggnMAA&url=https%3A%2F%2Fwww.bluetooth.org%2Fdocman%2Fhandlers%2Fdownloaddoc.ashx%3Fdoc_id%3D229737&usg=AFQjCNFY1IFeFAAWwimnoaWMsIRZQvPDSw

4.2 Roles

The Bluetooth Low Energy can be used in four different roles. One device may support multiple roles.

- **Broadcaster**, transmitter only
- **Observer**, receiver only
- **Peripheral,** usually slave / GATT Server
- **Central**, usually master / GATT Client

The Broadcaster and observer are independent. However, the peripheral and the central are bonded each other. It is always the central that initiates connection to peripherals and a central can support multiple connections. But the peripheral can only be connected to one central.

In this document, the peripheral is considered as a server and the central as a client which is the most current case.

4.3 Connection

The cycle of a BLE connection between a central and a peripheral is separated in three parts.

- **Advertising** (peripheral) and **scanning** (central)
- **Connection procedure** and **services discovery**
- **Connection events**

4.3.1 Advertising and Scanning

The advertising is performed by the peripheral. It consists to send information about the peripheral to any device that want to listen.

The scanning is performed by the central. It consists to listen the information sent by a peripheral. If the information satisfied the central, it runs a connection procedure with the peripheral.

Each advertise interval, the peripheral sends the same information about its. The central searches peripherals each scan interval during the scan windows.

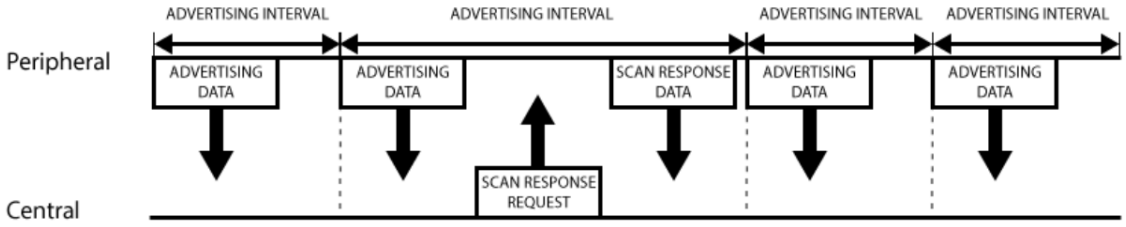


Figure 6: Scanning and advertising procedure, from learn.adafruit.com

4.3.2 Connection procedure and services discovery

When a central finds a peripheral with the information searched in the advertising, it sends a connection request and precise the parameters of the connection.

Then, the central performed a services discovery. A service is an application performed by the peripheral. The central search all the services whose is interested to get them data.

4.3.3 Connection Events

A Connection Event is sent by the central to say to a peripheral that it can transmit its data. In a BLE connection, four parameters handle the connection events:

- **Connection interval min**
- **Connection interval max**
- **Connection Timeout**
- **Slave Latency**

Those parameters are why the BLE is low power consumption because it allows to the peripheral to send the data only on a connection event and to sleep the rest of the time.

The connection parameters are always defined by the central. But the peripheral can suggest to the central parameters that are more suitable for him.

4.3.3.1 Connection Interval

Data are transmit only on connection events. Those connection events are sent by the central. The peripheral can only send data on a connection events.

4.3.3.2 Connection Timeout

However, the central must send a connection event and a peripheral must response otherwise a timeout is enable. When the timeout is passed and no other response from one of the devices, the link is considered lost and the devices close the link.

4.3.3.3 Slave Latency

On the other hand, the goal of the BLE is to be low power consumption. But it is a loss of power if the peripheral must response to the central and it has even no data. The Slave latency allows the peripheral to not response to several connection events to save batteries if it has no data.

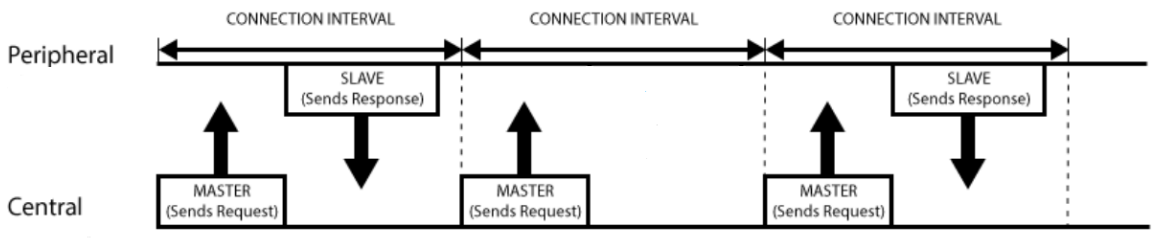


Figure 7: Connection events with slave latency of 1, from learn.adafruit.com

4.4 Profiles, Services and Characteristics

The profiles, services and characteristics defines the application and data contain by the devices. There are stored in the peripheral, called GATT server as well. Then, the central sends request to get information or value about them.

4.4.1 General Definitions

4.4.1.1 Bluetooth SIG

Bluetooth Special Interest Group (SIG) is the body that oversees the development of Bluetooth standards and the licensing of the Bluetooth technologies.

4.4.1.2 UUID

The Universally Unique Identifier (UUID) is a 128bits unique number attributed to:

- **Type of an attribute**
- **Services**, which is contained in the attribute value
- **Characteristics**, which is contained in the attribute value
- **Descriptors**, which is contained in the attribute value

Some UUID are already defined by the Bluetooth SIG. Those UUID are represented only with 16bits.

4.4.1.3 Handle

The attribute handle is a unique 16-bit identifier for each attribute on a peripheral server. The central use this value to access to an attribute and not the UUID.

4.4.1.4 Type

The attribute type is a UUID that define the kind of data present in the attribute. Typical types are:

- **Service declaration**
- **Characteristic declaration**
- **Characteristic value**
- **Characteristic descriptor**

4.4.1.5 Permission

The attribute permission specifies the operations allows on the attribute and the security requirements. Typical operations are:

- **Read**
- **Write**

Attention: If a characteristic allows the central to read it value and the attribute permission read is not enabled, the central is not able to read the value.

4.4.1.6 Attributes

An attribute is the smallest entity used to define services and characteristics by the GATT Server. An attribute has:

- **Handle**
- **Type**
- **Permission**
- **Value**, which can be a UUID, a sensor value, possible operations, …

The attributes are organized in an array and the handles are the number of the rows.

	Handle	Type	Permission	Value
Service Declaration	0x0000			
Characteristic Declaration	0x0001			
Characteristic Value	0x0002			
Characteristic Descriptor	0x0003			
Characteristic Declaration	0x0004			
Characteristic Value	0x0005			
Service Declaration	0x0006			
Characteristic Declaration	0x0007			
Characteristic Value	0x0008			
Characteristic Descriptor	0x0009			

Table 2: example of Attributes defined in a server

4.4.2 Profiles

Profiles are definitions of applications and specify general behaviours that *Bluetooth®* enabled devices use to communicate with other Bluetooth devices. With BLE, the profiles are defined in the peripheral server. A Profile is composed of several services.

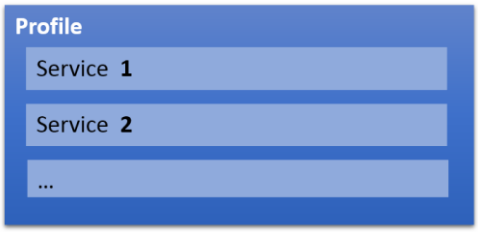


Figure 8: Profile

4.4.3 Services

Services are the part of the profile that define an application performed by the profile. Some services are already defined to easier developers to make applications and firmware compatible. Typical BLE services are:

- Current Time Service
- Battery Service
- Blood Pressure
- Continuous Glucose Monitoring
- Heart Rate
- …

A list of defined BLE profile can be found at https://www.bluetooth.com/specifications/gatt. Those services are defined by the Bluetooth (SIG). However, it is possible to create your own services.

The attributes contain within the services are:

- **Service declaration,** which contains the service UUID
- **The attributes of the characteristics**

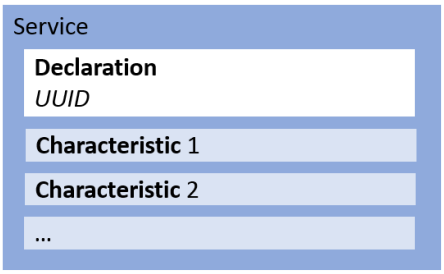


Figure 9: Service

4.4.4 Characteristics

The characteristic is a container for user data. As the services, some characteristics are already defined:

- Date Time
- Battery Level
- Blood Pressure Measurement
- Heart Rate Measurement
- …

The attributes contain within the characteristic are:

- **Characteristic declaration**, which provides metadata about the actual user data
- **characteristic value**, which is a full attribute that contains the user data in its value field
- **Descriptor** (optional), which further expand on the metadata contained in the characteristic declaration

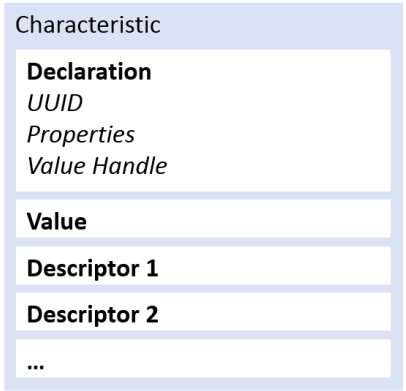


Figure 10: Characteristic

4.4.4.1 Characteristic declaration

The attributes characteristic declaration gives three information about the characteristic:

- **UUID of the characteristic**
- **Value handle**, handle of the attribute characteristic value

- **Properties**, operation permitted on the characteristic

Usually, the properties of the characteristic describe how the central accesses to the characteristic's data. The main properties are:

- **Read**, the central reads from the value of the characteristic
- **Write**, the central writes to the value of the characteristic
- **Notify**, the central receives automatically the new value of the characteristic
- **Indicate**, the central receives automatically the new value of the characteristic and sends a confirmation.

The operation Notify and Indicate are enabled by the central using the attribute descriptor. Then the peripheral collects the data, when the function Notify or Indicate are called, and sends the data each connection event.

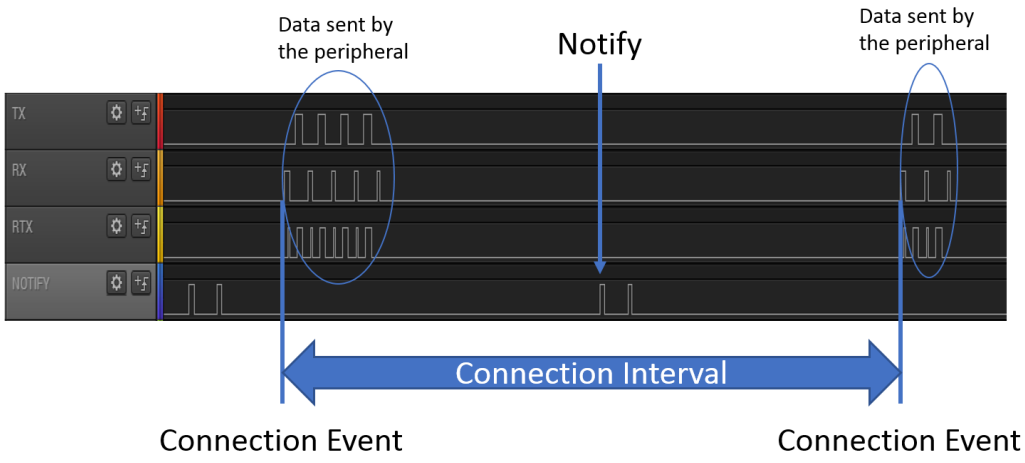


Figure 11: Measurement of Notification from Peripheral

4.4.4.2 Characteristic value

The characteristic value attribute contains the actual user data that the client can read from and write to for practical information exchanges. The value of a characteristic value attribute can contain any type of data imaginable.

4.4.4.3 Descriptor

The characteristic descriptor is used to provide the client additional information about the characteristic and its value.

The most important and commonly used is the Client Characteristic Configuration Descriptor (**CCCD**). The descriptor enabling or disabling the operation notify and indicate of the peripheral.

4.4.5 Example

Each white square is an attribute.

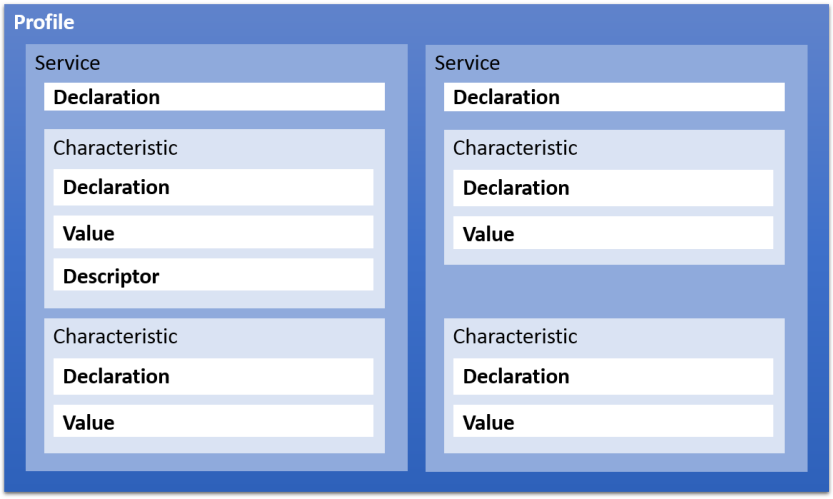


Figure 12: Profile template

It is possible to see the services and the characteristics in a GATT Server like a tree:

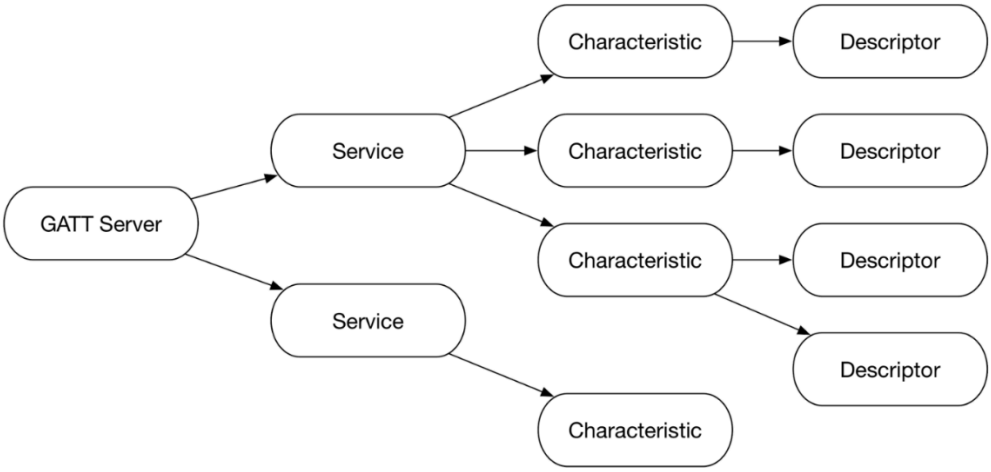


Figure 13: Peripheral services tree, from https://www.bignerdranch.com/

4.5 BLE Stack

The stack manages the communication Bluetooth, the services and the characteristics. Each system, Zephyr RTOS and SoftDevice, implement its own BLE Stack and it is why testing the behaviour of a Bluetooth stack with a high BLE traffic is important.

The Bluetooth Low Energy stack is separated in three parts:

- **Application**
- **Host,** which contains the configuration of device and the profiles and provides API for application
- **Controller,** which controls the connection requirements and the transceiver

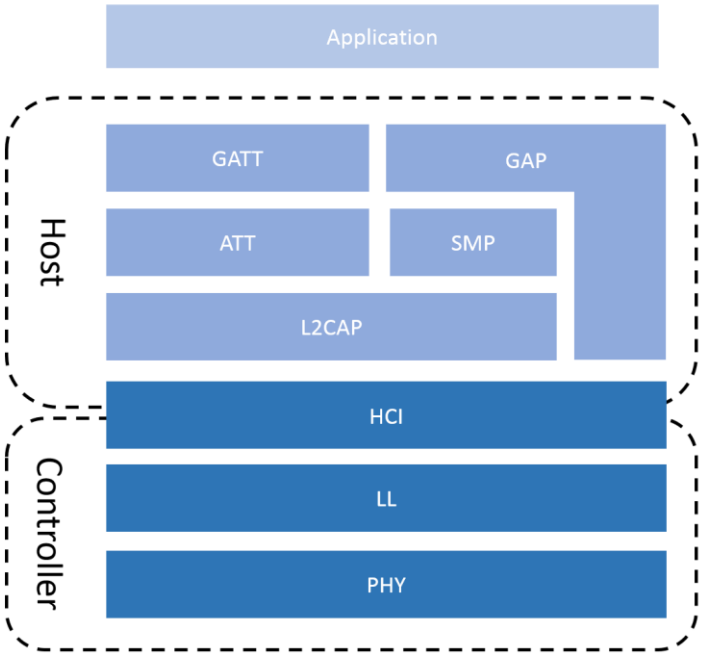


Figure 14: BLE Stack

4.5.1 Host

GAP, General Access Profile, is in control of scanning, advertising, security establishment and connections. It defines the roles of the devices and additional data, e.g. its name.

GATT, General Attribute Profile, defines how data is organized and exchanged in between different applications. The data in GATT is organized in services and characteristics.

ATT, Attribute Protocol, defines the data in the form of attributes. This form is defined at chapter **4.4.1.6 Attributes**

SMP, Security Manager Protocol, offers the security procedure to a BLE application. For example, the typical security procedures are device authentication, device authorization, data integrity, data confidentiality.

L2CAP, Logical link control and adaptation protocol, prepares packets. If a packet from the upper layers is too large, this layer fragmented and recombines it for the transfer.

4.5.2 Controller

HCI, Host Controller Interface, makes possible to interface a wide range of Hosts with the controller. Therefore, it is possible for a device to have several roles, Peripheral, Central, Broadcaster, Observer, at the same time.

LL, Link Layer, establishes and manages connections. It is in charge to send packets and to keep responding to or sending connection events at each connection interval.

PHY, Physical Layer, contains the analog communications circuitry used for modulating and demodulating analog signals and transforming them into digital symbols.

4.6 BLE Requirement

To ensure the correct communication between the devices and to keep the connection, BLE requires to take care of different element. It is important to set those requirements correctly to test the different systems.

4.6.1 Priority

It is important that no connection events are missed to avoid any unintentional disconnection. Hence, the BLE stack is the highest priority to no be suspended by any interrupt from other peripherals.

4.6.2 Number of connections

In the case of the role of the device is peripheral, the Peripheral is limited to be connected to only one central.

In the case of the role of the device is central, the central is not limited for number of peripheral connected. However, the number of connection is limited by the environment used.

	SoftDevice	Zephyr RTOS
Number Peripherals Connected	8	64

Table 3: Number max of connections for central

4.6.3 MTU

ATT Maximum Transmission Unit (MTU) is the maximum length of an ATT packet. An ATT packet contains the information of the Read, Write, Indicate, Notify request and response. As well, the MTU limits the size of the data contains in the attribute characteristic value.

The ATT MTU is defined by the L2CAP layer and can be theoretically anywhere between 23Bytes and infinity. However, this size is limited by the environment used.

	SoftDevice	Zephyr RTOS
RX MTU max [Bytes]	251	1300
TX MTU max [Bytes]	251	2000

Table 4: MTU max

The BLE packet structure has a payload of 33Bytes and each layer in the protocol stack takes cut. Hence the ATT protocol has 23Bytes left, the minimal MTU, and 20Bytes for data (MTU – opcode - handle).

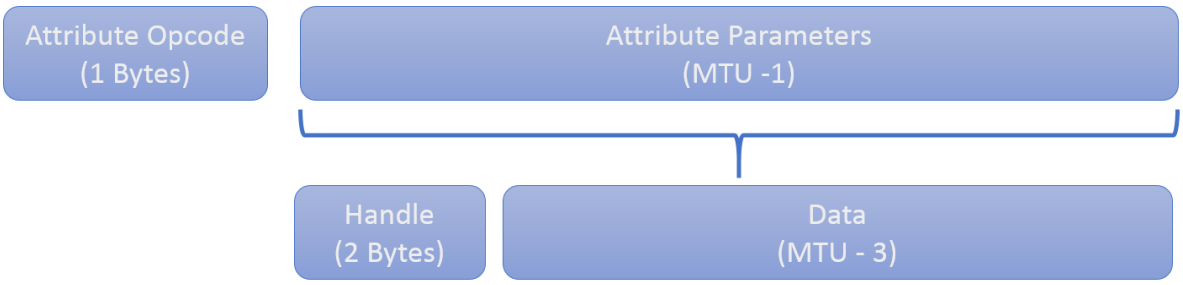


Figure 15: Attribute packet without authentication signature

When MTU is bigger than 23bytes, the L2CAP layer fragments and recombines ATT packets to send the ATT packet with several BLE packet at each connection interval.

4.6.4 Time Requirement

The connection, advertising and scanning parameters are defined in a specific range of value.

	Min	Max	Interval	Defined By
Connection Interval Min/Max	7.5 [ms]	4 [s]	1.25 [ms]	Central
Connection Timeout	100 [ms]	32 [s]	10 [ms]	Central
Slave Latency	0	499		Central
Scanning Interval	2.5 [ms]	10.24 [s]	0.625 [ms]	Central
Advertising Interval	20 [ms]	10.24 [s]	0.625 [ms]	Peripheral

Table 5: Parameters value

Each BLE application require different connection parameters. The table below show typical uses.

	Conn. Interval Min	Conn. Interval Max	Slave Latency	Conn. Timeout
Mouse	7.5 [ms]	15 [ms]	20	3 [s]
Keyboard	7.5 [ms]	30 [ms]	6	430 [ms]
Glucose Meter	10 [ms]	100 [ms]	0	4 [s]
Power Profiling	20 [ms]	100 [ms]	0	4 [s]
Heart Rate	400 [ms]	650 [ms]	0	4 [s]
Blood Pressure	500 [ms]	1 [s]	0	4 [s]
Current Time	500 [ms]	1 [s]	0	4 [s]
Health Thermometer	500 [ms]	1 [s]	0	4 [s]

Table 6: Time requirements for typical use cases

The values are the values indicated in the examples of the nRF5 SDK v13.0.0. Those values are used to define test cases for this project.

4.7 Sources

https://learn.adafruit.com/introduction-to-bluetooth-low-energy/introduction
https://www.safaribooksonline.com/library/view/getting-started-with/9781491900550/ch04.html
https://learn.mikroe.com/bluetooth-low-energy-part-1-introduction-ble/
https://www.link-labs.com/blog/bluetooth-vs-bluetooth-low-energy
https://www.bluetooth.com/specifications/profiles-overview
https://www.bluetooth.com/specifications/gatt
http://infocenter.nordicsemi.com/pdf/S132_SDS_v5.0.pdf
https://en.wikipedia.org/wiki/Bluetooth_Special_Interest_Group

5 Specifications

In this project, two systems are compared. The first one is a Bare Metal system, without operating system, using SD+SDK from Nordics. And the second is Zephyr RTOS. Both systems are implemented on a nRF52x SoC.

The elements compared are:

- **power consumption**
- **interrupt latency**
- **Bluetooth Low Energy behaviour**

The results must not define which system is the best. This is only a representation of the performance of Zephyr RTOS using the SD+SDK as reference.

The BLE roles interesting to analyse the behaviour are peripheral and central because of the connection requirements between the devices. Therefore, the two roles are tested.

- **Peripheral**, which sends data to a central
- **Central**, which receives data from peripherals and maintains the connection

To analyse the behaviour of the devices, uses cases must be defined to test the systems under different conditions. Those uses cases are inspired by application examples of the nRF5 SDK v13.0.0.

	Case 1 High	Case 2 Balanced	Case 3 Slow
Conn. Interval	7.5 [ms]	50 [ms]	400 [ms]
Slave Latency	0	0	0
Conn. Timeout	500 [ms]	500 [ms]	1 [s]
Inspired By	Mouse	Glucose Meter	Heart Rate

Table 7: Use cases to measure the behaviour

The use case 1 represents the worst case possible for a BLE device because of the lowest connection interval used. In this case, the central can handle the connection with only one peripheral because of the connection interval too short.

The test case 2 is for typical application that want to save as much power as possible but keep a high-throughput. It is interesting because it is possible to connect eight peripherals to a central, the maximum number the SoftDevice can handle.

The test case 3 is for typical application where the throughput is not important and the device want to save the battery. It is not the lowest connection interval possible but it is more common when a application need to keep sleeping a long time.

5.1 Peripherals

The BLE peripheral is the system that acquires data from sensors and sends them to the central. To test the behaviour of the peripheral, an extension board is developed with different sensors is developed. The components used are:

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

Those extension board allows to test the system in real circumstances and to generate interruptions to stress the system. The frequency of the interruptions is incremented to analyse how the system can handle them.

The A/D Converter and accelerometer are used as sensors because they are simple to implement and use the serial communication which allows to test other elements of the SDK and Zephyr RTOS.

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.

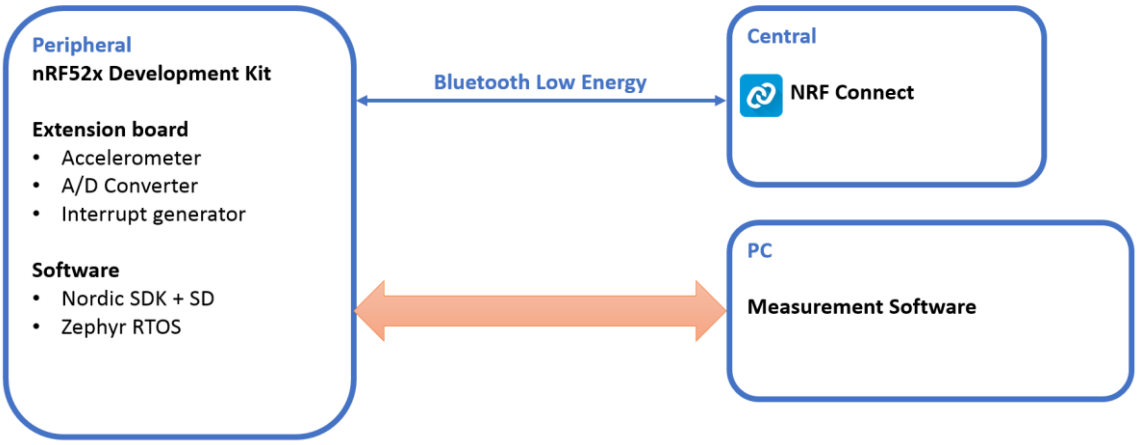


Figure 16: Behaviour measurement on BLE peripheral Schema Block

5.2 Central

The central is the system that receives data from peripherals and oversees the connection. To test the behaviour of the central, several peripheral send the same data than the peripheral developed. As the peripheral, the central is subject to a large amount of interrupt to be stressed.

The limitation of the SoftDevice is eight peripherals connected to the central which the maximum number of peripherals connected to our central.

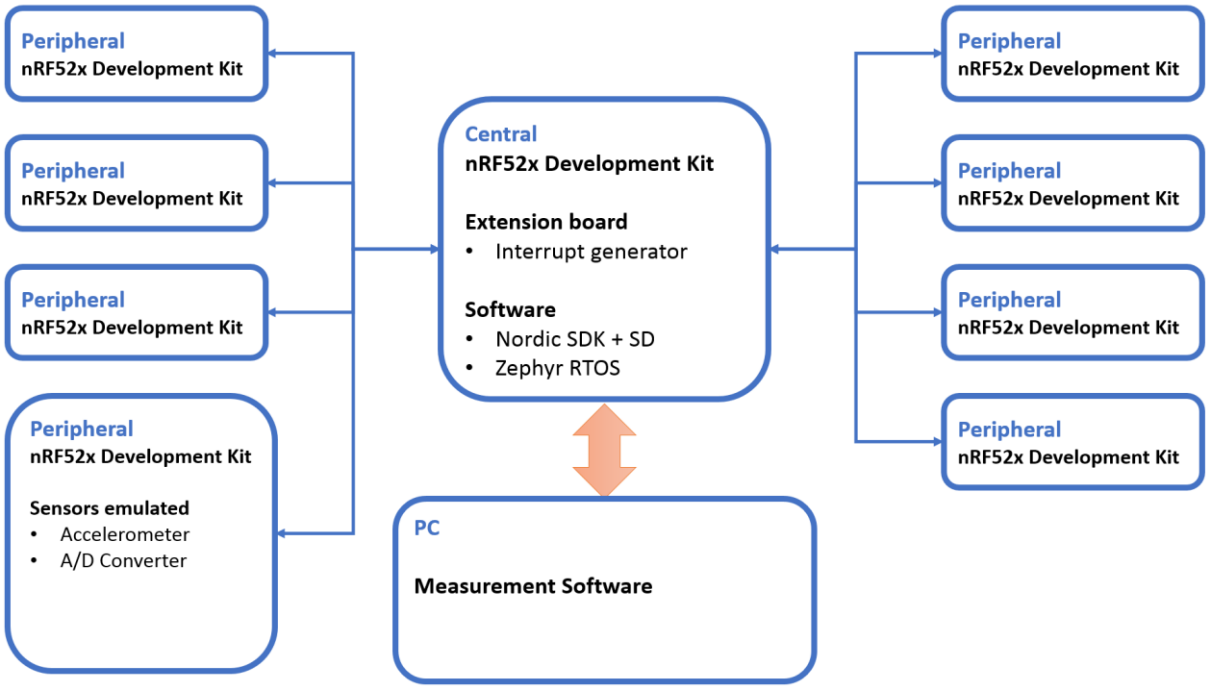


Figure 17: Behaviour measurement on BLE Central Schema Block

6 Hardware

UPDATE THE SCHEMATIC FIGURE

The hardware must provide a solution to provide data from an Accelerometer and A/D Converter, to generate interruption to stress the device and to measure the power consumption. It is separated 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.

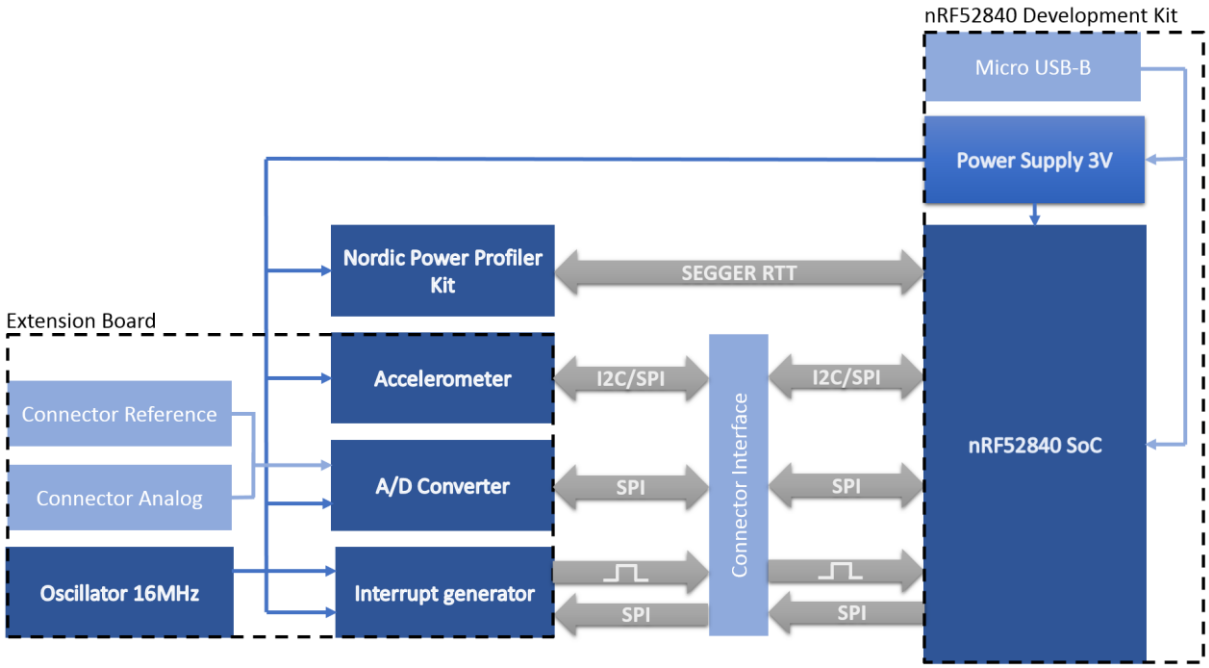


Figure 18: Hardware schema block

6.1 nRF52840 SoC

The SoC used is the nRF52840. Here is the description from https://www.nordicsemi.com/.

"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), 802.15.4, 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 and avoids being limited with the measurements because of the capacity of the SoC.

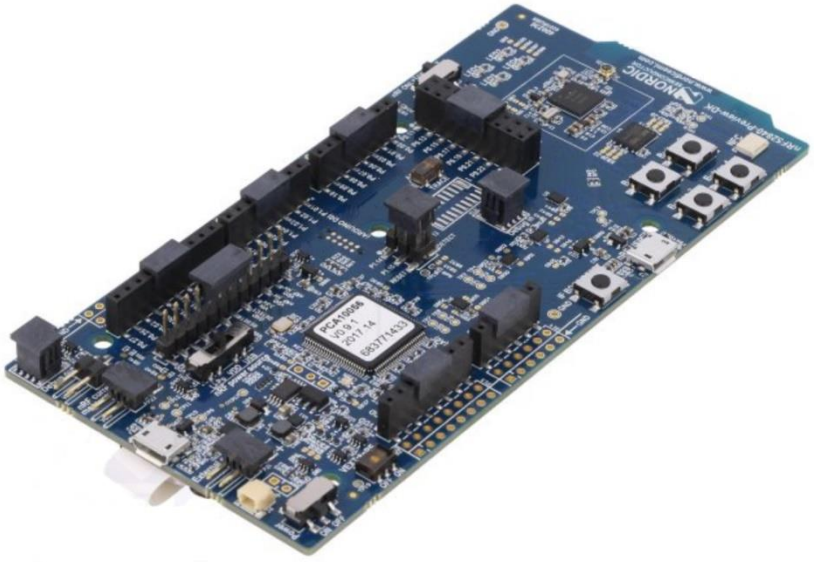


Figure 19: Nordic nRF52840 Preview Kit

6.2 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

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 on which the PPK is connected.

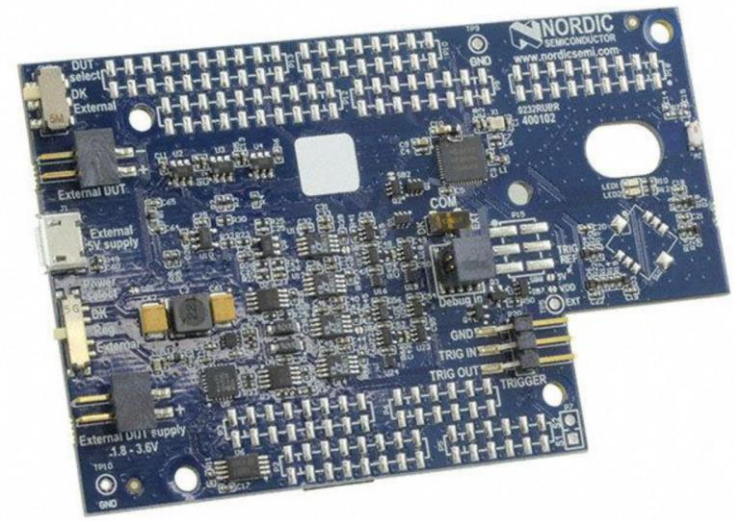


Figure 20: Nordic Power Profiler Kit view

6.3 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.

6.4 Power Supply

The power supply transforms the voltage 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.

$$Estimated\ current\ max = I_{nRF5\ DK} + I_{ADC} + I_{ACC} + I_{interrupt\ generator} + I_{ppk}$$
$$= 50mA + 300\mu A + 185\mu A + 4.5mA + 100mA = 134.985mA$$

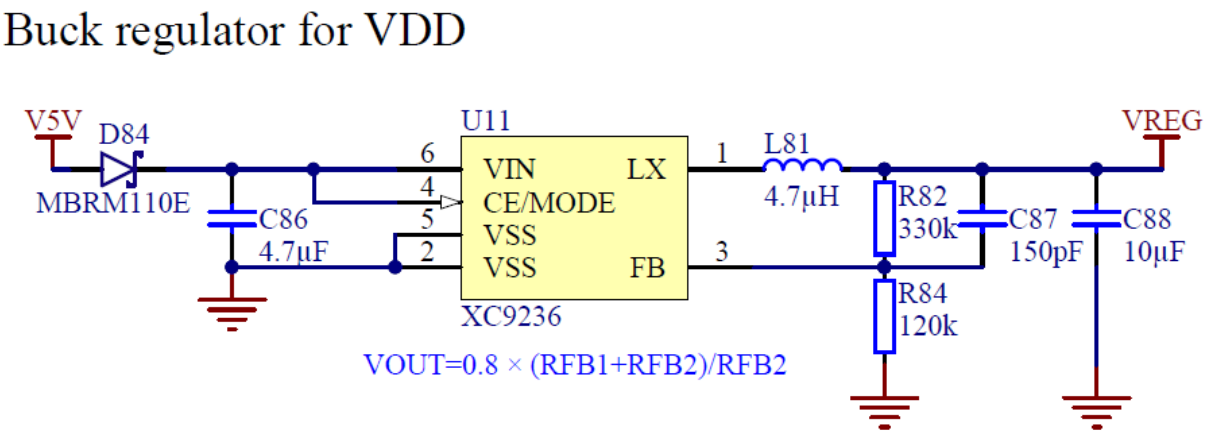


Figure 21: Regulator buck 3V schematic, from nRF52840 DK User Guide

6.5 Connector Interface

The connector interface is defined by the nRF52840 development kit and is almost the same for all the Nordic's DK, Arduino Uno shield compatible.

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 which allows an easy connection of the different part of the system.

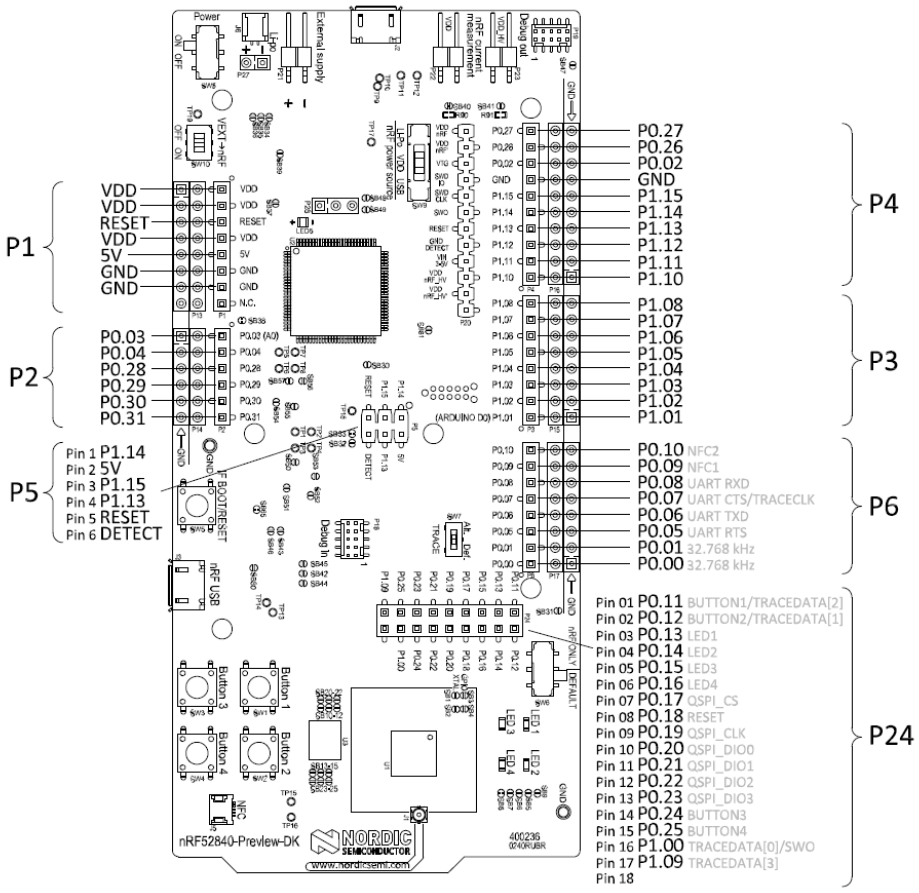


Figure 22: nRF52840 Preview DK board connectors, from nRF52840 DK User Guide

6.6 Extension Board

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

The purpose of the extension board is to use the nRF52840 Chip under real conditions of use. To do it, the extension board implements different components.

- **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**

All the hardware, Schematic and PCB, of the extension board is developed with the Software Altium Designer 17™.

6.6.1 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** current consumption 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).

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

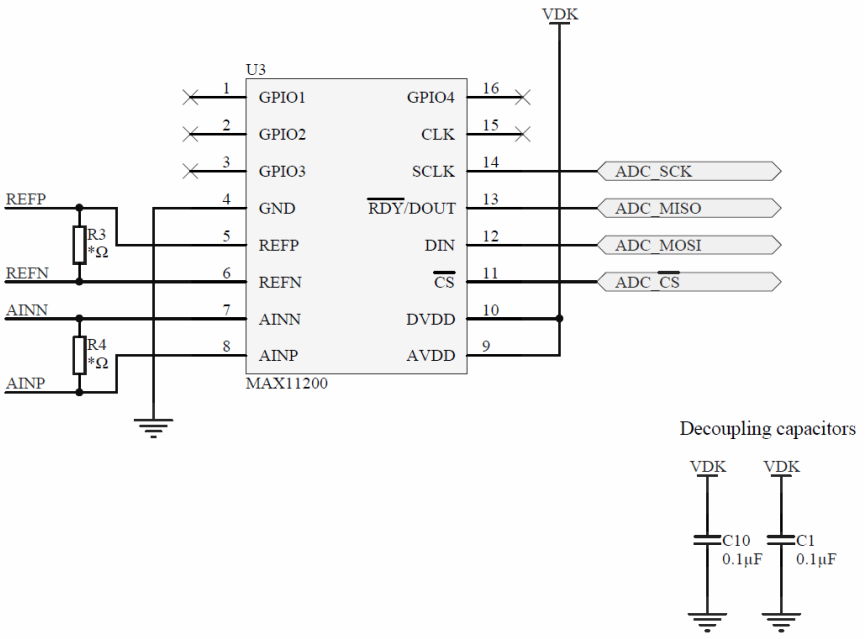


Figure 23:MAX11200 schematic

6.6.1.1 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.

Oscillator									
2.4576MHz	0.83	2.08	4.17	8.33	12.5	25	50	100	[sps]
2.048MHz	1	2.5	5	10	15	30	60	120	[sps]
Single-cycle Conversion Mode									

Oscillator					
2.4576MHz	60	120	240	480	[sps]
2.048MHz	50	100	200	400	[sps]
Continuous Conversion Mode					

Table 8: A/D Converter Data Rate in sample per second,

$$Throughput\ max = \frac{1}{Data\ Rate} = \frac{1}{480} = 2.08ms$$

$$Throughput \min = \frac{1}{Data\ Rate} = \frac{1}{50} = 1.2s$$

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

$$Data\ frame = 32bits = 8bits\ address + 24bits\ data$$

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.

$$T_{\max Data\ frame} = \frac{1}{f_{SPI}} \cdot 32bits = 8\mu s$$

The throughput of the A/D Converter is interesting because it can be easily adapted to the BLE connection interval use with the different cases.

6.6.2 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** current consumption 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. However, it is not used because the applications of Bluetooth Low Energy are not with large data packets.

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.

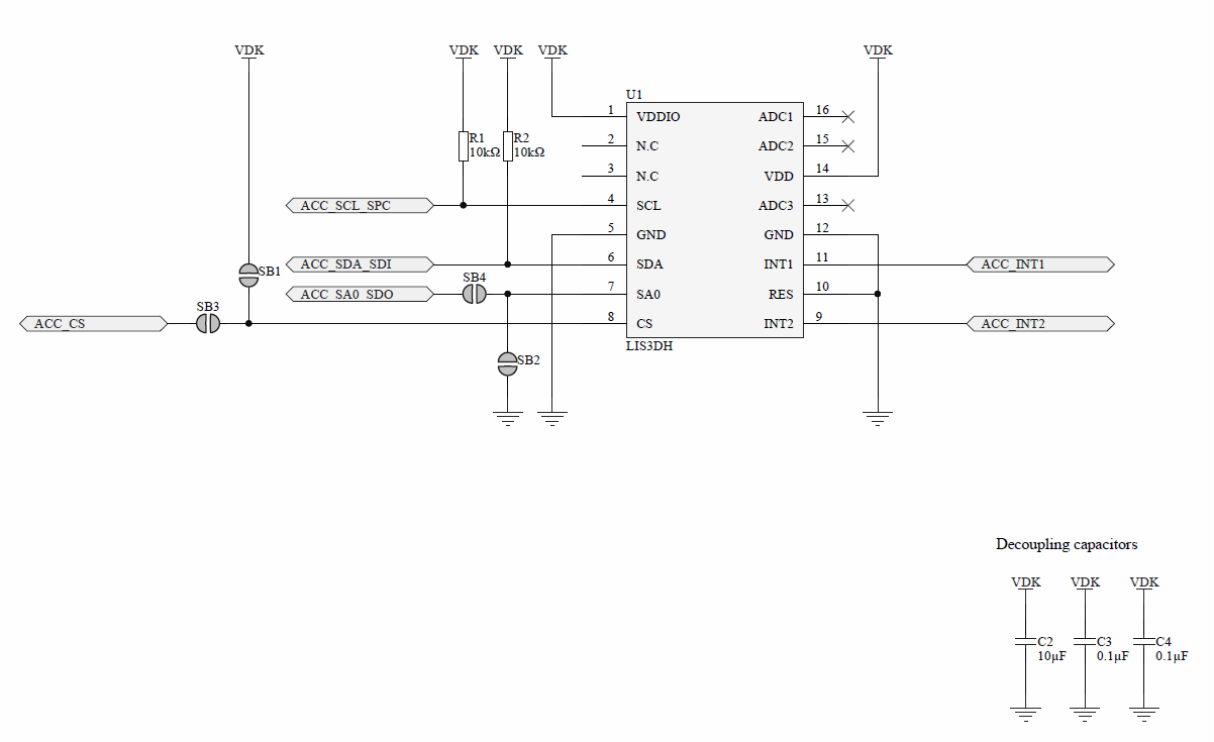


Figure 24:LIS3DH Schematic

6.6.2.1 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 which can be adapted with the BLE connection interval used.

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 9: Accelerometer Data Rate in Hertz

$$Throughput\ max = \frac{1}{Data\ Rate} = \frac{1}{5.376kHz} = 186\mu s$$

$$Throughput\ \min = \frac{1}{Data\ Rate} = \frac{1}{1Hz} = 1s$$

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 an ACK bit between each byte transmitted.

$$Data\ frame = 1bit\ start + 7bits\ I2C\ address + 1bit\ R/W + 8bits\ address + n_{fifo}(16bits\ X\ data + 16bits\ Y\ data + 16bits\ Z\ data) + n_{ack} + 1bits\ stop$$

$$Data\ frame\ max = 1 + 7 + 1 + \ 8 + 32 \cdot 6 \cdot 8 + \ n_{ack} + 1 = \ 1748bits$$

$$Data\ frame\ min = 1 + 7 + 1 + \ 8 + \ 16 \ + \ n_{ack} + 1 = \ 38bits$$

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

$$T_{\max Data\ frame} = \frac{1}{f_{I2C}} \cdot Data\ frame\ max = 4.37ms$$

$$T_{\min Data\ frame} = \frac{1}{f_{I2C}} \cdot Data\ frame\ min = 95\mu s$$

6.6.3 Interrupt generator

The interrupt generator generates pulse that create interruptions in the programme. The period of interruptions can be easily changed to increase the stress regards the system.

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** current consumption 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:

$$f = \frac{Clock}{Resolution} \cdot 28Bits\ Register = 0.06Hz \ \cdot 28Bits\ Register$$

$$fmax = 16MHz \rightarrow Tmin = \ 62.5ns \qquad fmin = 0.06Hz \rightarrow Tmax = 16.66s$$

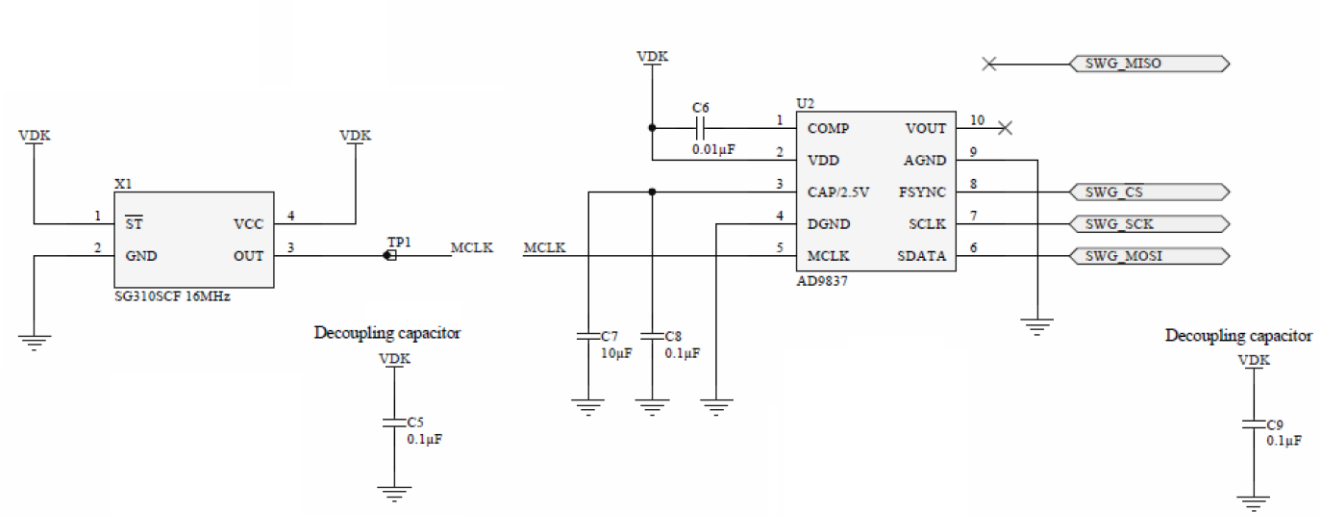


Figure 25: AD9837 schematics

6.7 Annexes

- List of components

- Schematic
- PCB
- Pin Map of components

6.8 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

7 Software

The Software implements SD/SDK and Zephyr RTOS and creates the BLE application used to analyse the behaviour of the different environments. It is separated in four parts:

- **Environment Layer**
- **Abstract Layer**
- **Driver Layer**
- **Application Layer,** which is use all the below layer to create a peripheral and a central application.

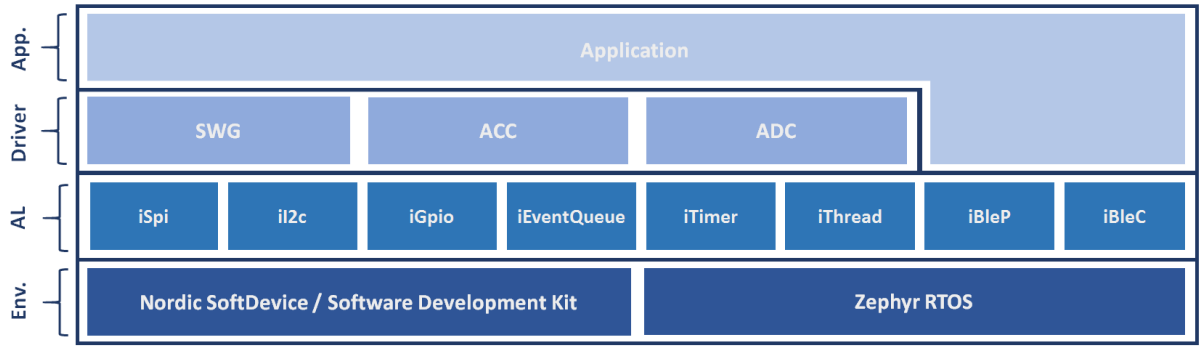


Figure 26: Architecture of the Software

7.1 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.

Those elements are explained at chapter **2. SoftDevice and Software Development Kit** and **3. Why Zephyr RTOS?**

7.1.1.1 Nordic SD - SDK

Different version of the SoftDevice and Software Development Kit existed. 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.

7.1.1.2 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.

7.2 Abstract Layer

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

The abstract layer is separated in seven parts:

- **iSpi**
- **iI2c**
- **iGpio**
- **iEventQueue**
- **iTimer**
- **iThread**
- **iBleP**, peripheral BLE
- **iBleC**, central BLE

The particularity of a Bare Metal is that it does not use an operating system. Therefore, it has no thread. However, it is possible to use an XF pattern to execute process when an event is push within a queue event. This pattern requires a scheduler to dispatch the event.

The SDK library provides a scheduler but decided to not use it and to create my own scheduler because the use of the SDK scheduler was not convenient to interface the SDK and Zephyr RTOS.

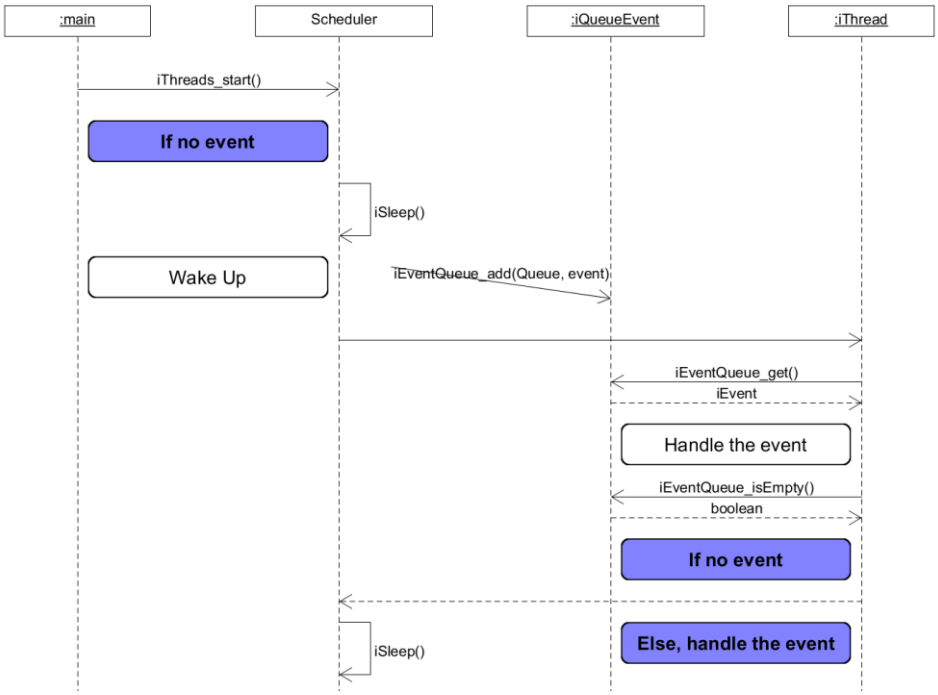


Figure 27: Scheduler for Bare Metal iThread

To fulfil the condition to use a XF pattern, each iThread must be built as the state machine below:

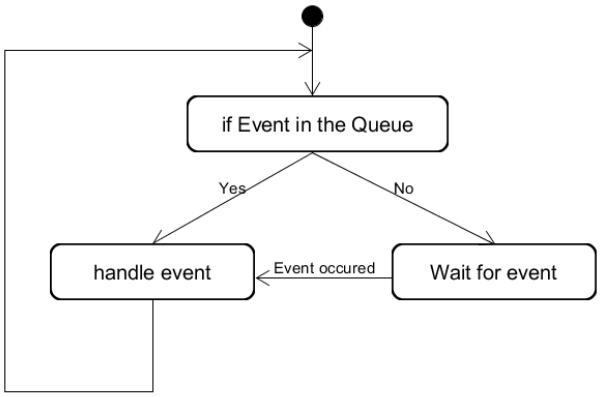


Figure 28: iThread architecture

7.3 Driver Layer

The Driver Layer manages the communication with the devices 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**, Square Wave Generator, driver of the AD9837
- **ACC**, Accelerometer, driver of the LIS3DH
- **ADC**, A/D Converter, driver of the MAX11200

7.4 Application Peripheral

The peripheral acquires the data from the A/D Converter and accelerometer and notify them to the central. In the same time, it serves to the GPIO interrupts.

The application is separated in four **threads**, each thread pops events from different event queues:

- **BLE thread**, which disables the drivers when the central is disconnected
- **ACC thread**, which notifies the central of the axis values
- **ADC thread**, which notifies the central of the A/D Converter values
- **SWG thread**, which increases the interrupt frequency

Then, various elements pushed events within the event queues:

- **BLE**, events when device is connected or disconnected
- **ACC GPIO interrupt 1**, events when new samples are ready
- **ACC GPIO interrupt 2**, events when clicks
- **ACC Button**, events to enable or disable the driver
- **ADC Timer**, events when new value ready, timer adapted to the sample rate of the converter
- **ADC Button**, events to enable or disable the driver
- **SWG Button**, events to increase the interrupt frequency
- **SWG Button**, events to enable or disable the driver

Another part of the application is used for the interrupt latency measurement:

- **GPIO Interrupt**, to measure the interrupt latency
- **Timer**, to perturb the GPIO interrupt.

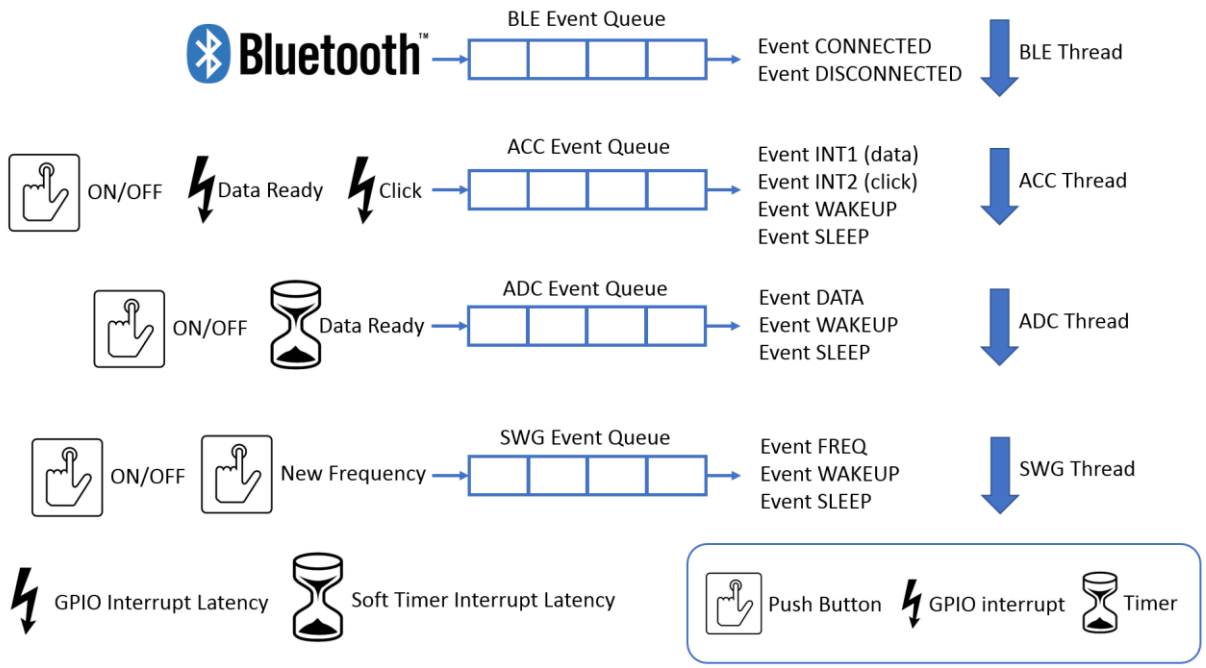


Figure 29: Peripheral application architecture

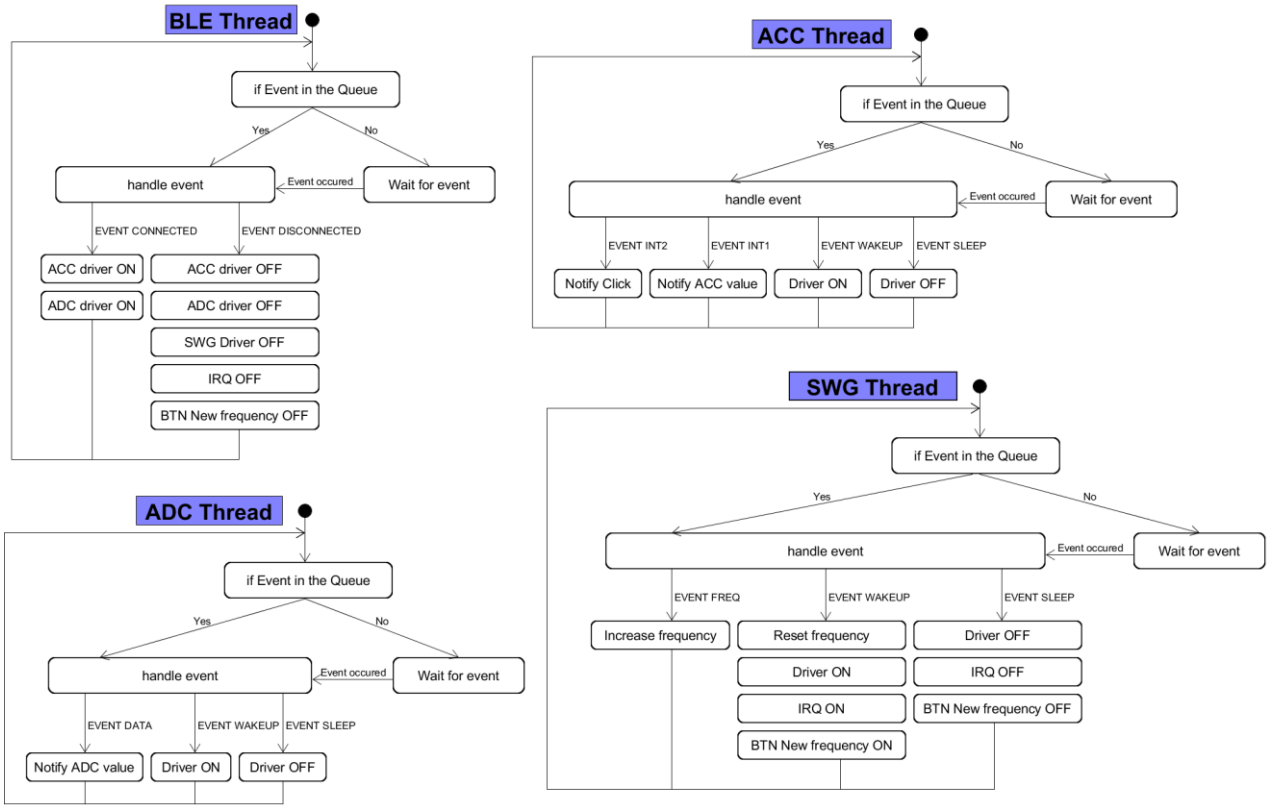


Figure 30: Threads process of the peripheral application

None BLE service already exists to send data from accelerometer or A/D converter. Therefore, simple custom services are made.

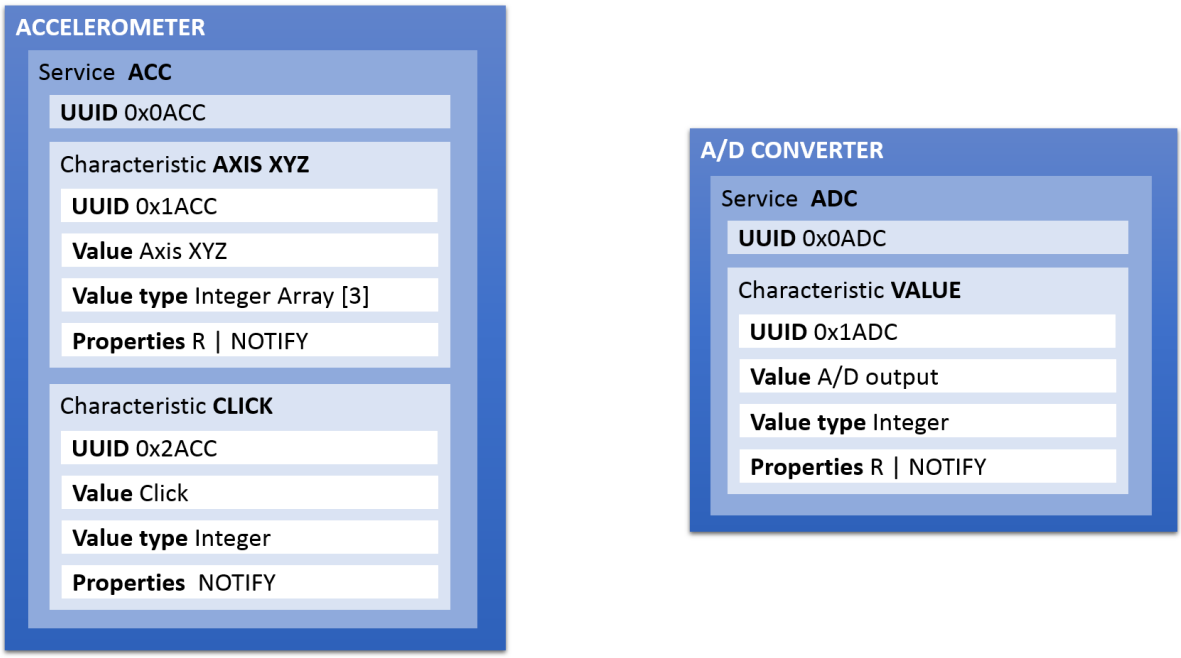


Figure 31: BLE services for peripheral

7.4.1 Configuration

It is possible to configure the peripheral to only use some drivers or disable the Bluetooth. The configuration possible are:

- **Enable/Disable** the Bluetooth
- **Set** the Advertisement Interval
- **Enable/Disable** the ACC
- **Set** the data rate of the ACC
- **Enable/Disable** the ADC
- **Set** the data rate of the ADC
- **Enable/Disable** the SWG
- **Set** the interval between each interrupt frequency
- **Enable/Disable** the software timer
- **Set** the interval of the software timer

7.5 Application Central

The central receives the data from peripherals. To create the peripherals communicating with the central, it is not relevant to use the extension boards created with several the peripherals. More, the nRF52840 Development Kit is in limited quantity. Hence, an application peripheral is developed on nRF52832 Development Kit using SoftDevice S132 to emulate values.

7.5.1 E-Peripheral

The application is called E-Peripheral for Emulated Peripheral. This application performs the same tasks than the peripheral developed above. The E-Peripheral is separated in three **threads**, each thread pops events from a different event queues:

- **BLE thread**, which disables the emulated drivers when the central is disconnected
- **ACC thread**, which notifies the central with emulated values
- **ADC thread**, which notifies the central with emulated values

Then, various elements pushed events within the event queue:

- **BLE**, events when device is connected or disconnected
- **ACC Timer Data**, events to emulate new samples ready
- **ACC Timer Click**, events to emulate clicks
- **ACC Data Button**, events to enable or disable the ACC Timer Data
- **ACC Click Button**, events to enable or disable the ACC Timer Click
- **ADC Timer**, events to emulate new values ready,
- **ADC Data Button**, events to enable or disable the ADC Timer

To indicate the state of the peripheral, two LEDs are used:

- **LED Started**, which indicates when the application is running
- **LED Connected**, which indicates when the device is connected to the central

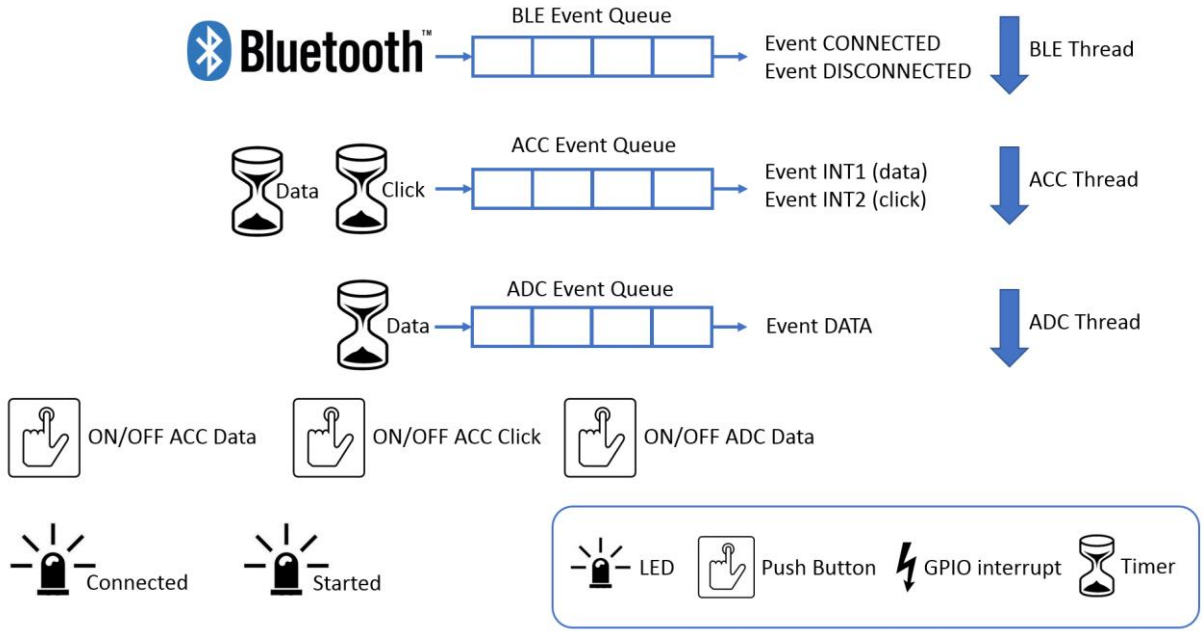


Figure 32: E-Peripheral application architecture

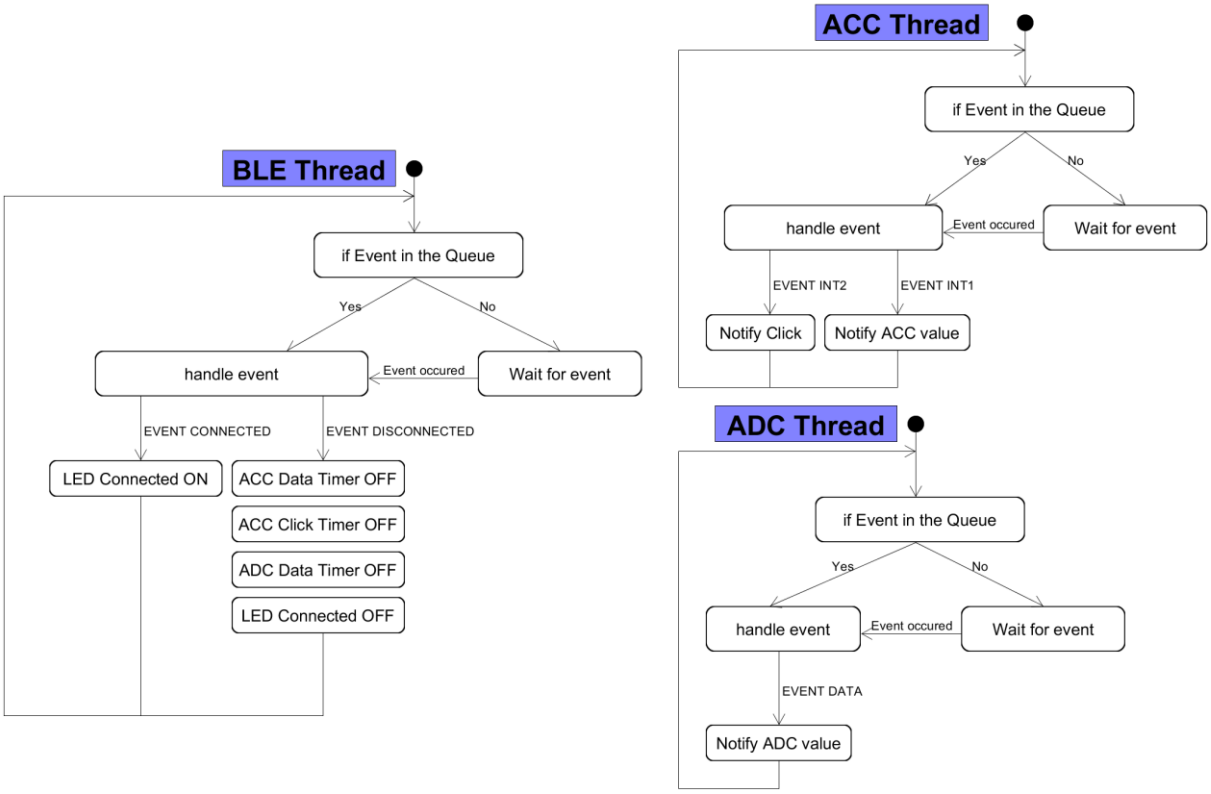


Figure 33: Threads process of the E-Peripheral application

The BLE services used are the same than the application peripheral developed for the measurements.

7.5.1.1 Configuration

The configuration of the E-Peripheral allows to:

- **Set** the Advertisement Interval
- **Set** the data rate of the emulated ACC
- **Set** the data rate of the emulated ADC

7.5.2 Central

The Central searches all the E-Peripherals by them name. Then, it performs the services discovery to find the services ACC and ADC. The Central is separated in two **threads**, each thread pop from a different event queue:

- **BLE thread**, which enables the notification of the connected peripheral
- **SWG thread**, which increases the interrupt frequency

Then, various elements pushed events within the event queue:

- **BLE**, events when device is connected, disconnected or the services discovery is finished
- **SWG Button**, event to increase the interrupt frequency
- **SWG Button**, to enable or disable the driver

Another part of the application is used for the measurement:

- **GPIO Interrupt**, to measure the interrupt latency
- **Timer**, to perturb the GPIO interrupt.

The Last part of the application takes care of calling the handler notifications from peripherals:

- **ACC Data**
- **ACC Click**
- **ADC Data**

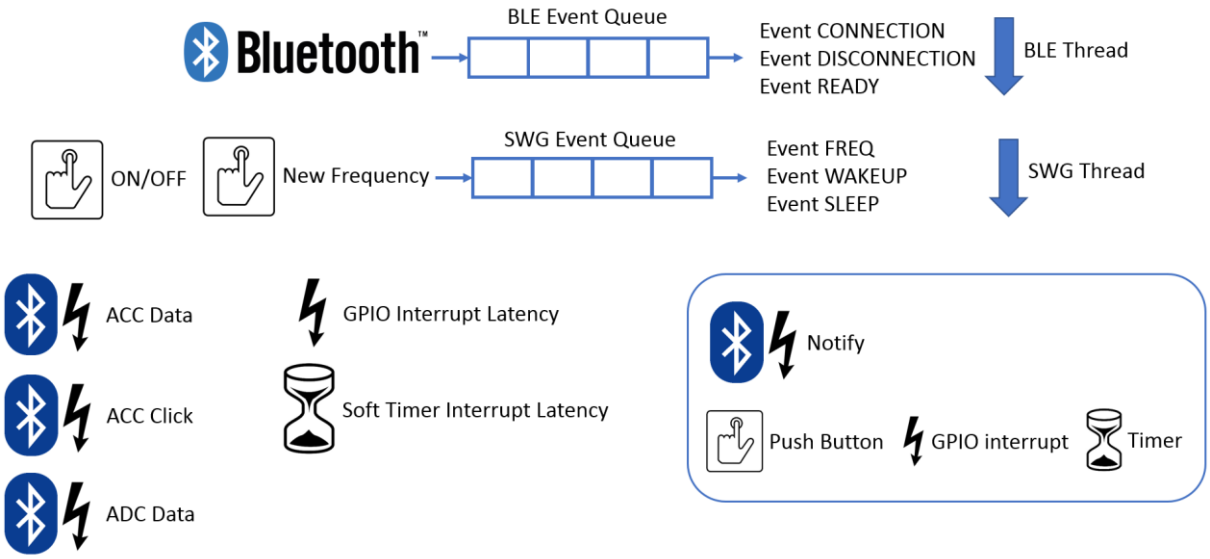


Figure 34: Central application architecture

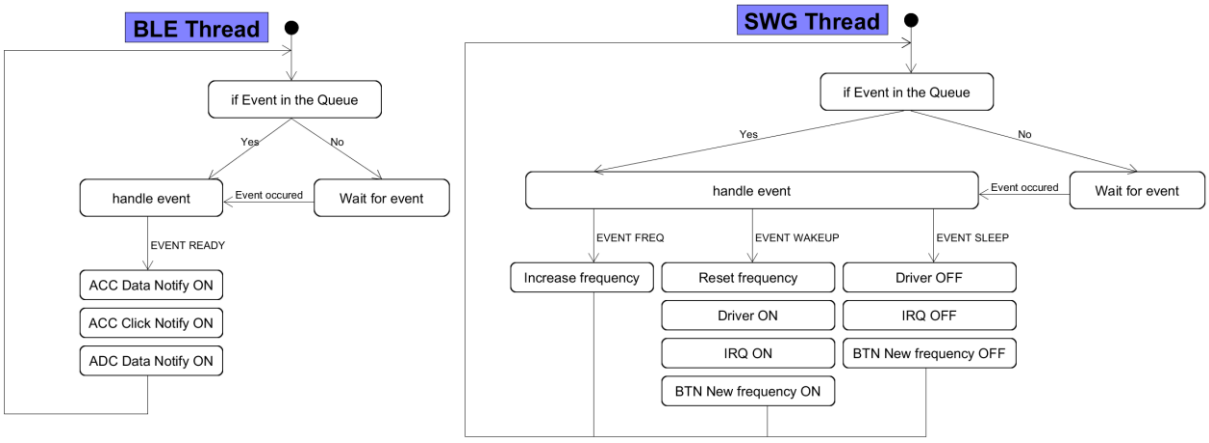


Figure 35: Threads process of the Central application

7.5.2.1 Configuration

It is possible to configure the peripheral to only use some drivers or disable the Bluetooth. The configuration allows to:

- **Enable/Disable** the Bluetooth
- **Set** the Scanning Interval
- **Set** the connection interval
- **Enable/Disable** the SWG
- **Set** the interval between each interrupt frequency
- **Enable/Disable** the software timer
- **Set** the interval of the software timer

7.6 Annexes

- Peripheral Application Map, buttons and leds
- E-Peripheral Application Map, buttons and leds
- Central Application Map, buttons and leds

7.7 Sources

https://infocenter.nordicsemi.com/index.jsp
https://nexus.zephyrproject.org/content/sites/site/org.zephyrproject.zephyr/dev/api/api.html
https://github.com/zephyrproject-rtos/zephyr

8 Measurements

The measurements determine the behaviour of the system in different condition. The different elements analysed are:

- Interrupt Latency
- Power Consumption
- Bluetooth Low Energy Behaviour

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 registers of the nRF52840 without any libraries.

8.1 Interrupt Latency

8.1.1 Why

The interrupt latency represents the time it takes to the interruption to be serviced. As the response time of real-time systems is important, the interrupt latency determines the minimum interval between interruptions to correctly handle them. Hence, the interrupt latency must be as low as possible.

The Interrupts latency have different sources that can contribute to the interrupt latency. They come principally from RTOS:

- When an RTOS disables interrupts while accessing critical OS data.
- When the context, e.g. status registers, is saved and resorted before and after processing the interrupt.
- When a context switch occurs to defer processing and to return to an RTOS task or threads.
- When an ISR interact with an RTOS by making system call such as semaphore.

However, the architecture of the CPU influences the interrupt latency as well. The interrupt latency of the CPU is constant and generally specified by the manufacturer. It is called the interrupt entry latency.

Here is the definition of interrupt entry latency from http://infocenter.arm.com/:

"The interrupt entry is the number of processor clock cycles between an interrupt signal arriving at the processor and the processor executing the first instruction of the interrupt handler."

The nRF52840 is built around a 32-bit ARM® Cortex™-M4F CPU. Its interrupt latency on entry is 12 cycles, plus a possible additional 17 cycles for Cortex-M4 with Floating Point Unit. The clock of the nRF52840 used is 64MHz, the interrupt entry converted in time is:

$$Interrupt\ Entry\ Latency = \frac{1}{64MHz} \cdot (12 + 17) = 453.125ns$$

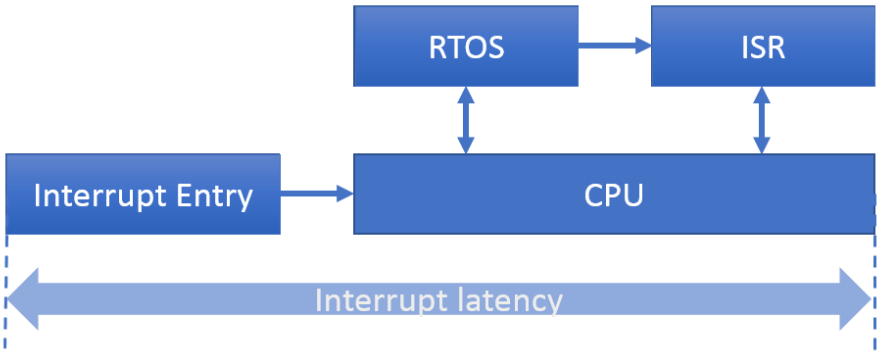


Figure 36: Interrupt latency schema block

8.1.2 Expected Results

First, the interrupt latency should not depend of the frequency of the interruptions. However, it could increase when several interrupts and timers occur in the same time.

Secondly, as Zephyr is an RTOS, the interrupt latency should be higher and less constant with Zephyr than a Bare Metal system.

8.1.3 How

The easier way to measure the interrupt latency is to toggle a pin on a GPIO interface. Using the GPIO allows to not disturb the system and to have a slight error. However, it is important to measure this error.

An input pin on a GPIO interface generates the interrupt and another output pin is toggled in the ISR. Then, the interval between the two signals is measured.

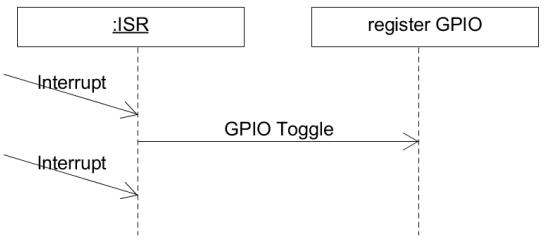


Figure 37: Sequence to measure the interrupt latency

The interrupt latency is usually given in processor clock or µs. As the measurement is a time interval, the result is given in µs.

8.2 Power Consumption

8.2.1 Why

The power consumption defines the autonomy of an embedded system. Longer the autonomy is, better is the embedded system. Hence, the power consumption must be as low as possible.

8.2.2 Expected Results

The power consumption should not be necessarily higher with Zephyr RTOS or a Bare Metal. However, it should increase with a higher frequency of interruption or when the BLE connection interval are faster.

8.2.3 How

The power consumption is measured with the Nordic Power Profiler Kit and no more code is implemented. However, 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.

The consumption results are given in I_{rms} because RMS represents the DC equivalent for current and voltage and is used to calculate the power.

8.3 Bluetooth Low Energy Behaviour

Measuring the behaviour of the Bluetooth Low Energy is difficult because:

- **SoftDevice is confidential** and the source code is not accessible
- **Zephyr RTOS is an open source** project, hence all the source code is accessible.

This difference is important because, to correctly compare a behaviour, the same tests must be performed and it is not possible to measure the BLE performance directly within the stack. It is why the number of tests possible is limited.

However, it is possible to use different peripherals of the nRF52840 to get some information. The different elements of the Bluetooth Low Energy measured are:

- **Advertising interval**, peripheral
- **Stack propagation delay**, peripheral, the time to push a data from APP to LL layer
- **Connection interval**, central
- **Connection event time**, central, time to communicate and receive data from peripherals BETTER NAME
- **Scan interval**, central
- **Scan window**, central

The advertising interval, connection interval, scan interval and scan window are measured only to see if the BLE stack is not perturb by the interrupt and other peripherals. The Bluetooth Core 4.0 strictly define those values.

The advertising interval, connection interval, scan interval, scan window and connection Event time are analysed using the radio's state of the nRF5x SoC.

8.3.1 BLE Radio State

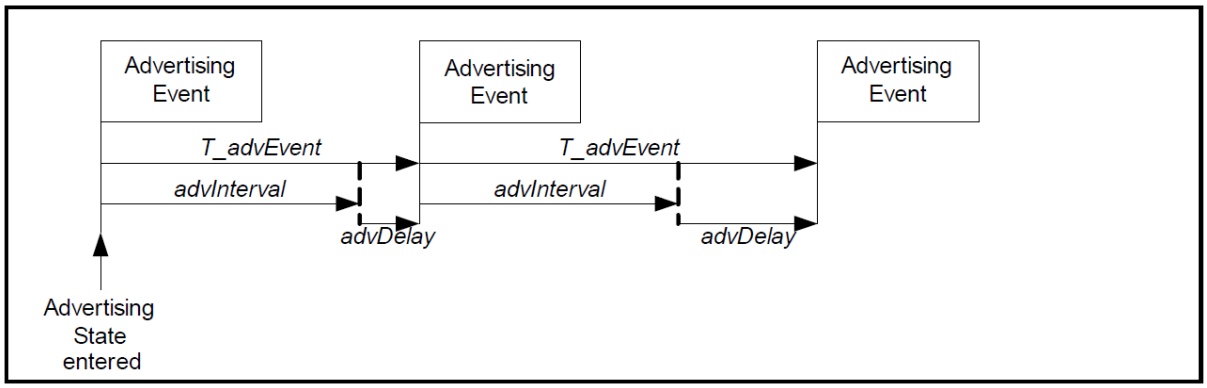
8.3.1.1 Why

It is important that BLE connection is not lost because one of the devices connected is not able to response correctly. Measuring the time of the different elements of a Bluetooth connection radio allows to see if the device still responses and analysed the time and the procedure to sends or receives a packet.

8.3.2 Expected Results

Generally, BLE stack is the highest priority of the system. Therefore, any external interrupt from other peripherals than the radio should not perturb the BLE communication

8.3.2.1 Advertising Interval



8.3.2.2 Connection Interval

8.3.2.3 Scan Interval

8.3.2.4 Scan Window

8.3.2.5 Connection Event time

The connection event time represents the time for the central to communicate with the peripherals connected and to receive them data. This time should not be higher than the connection interval.

It is not possible to give an expected result because it changes between implementations of BLE. However, it is interesting to compare the procedure between SoftDevice and Zephyr.

8.3.2.6 How

The nRF52840's Radio peripheral has a State Machine and the operating state of the Radio is control via several events and task.

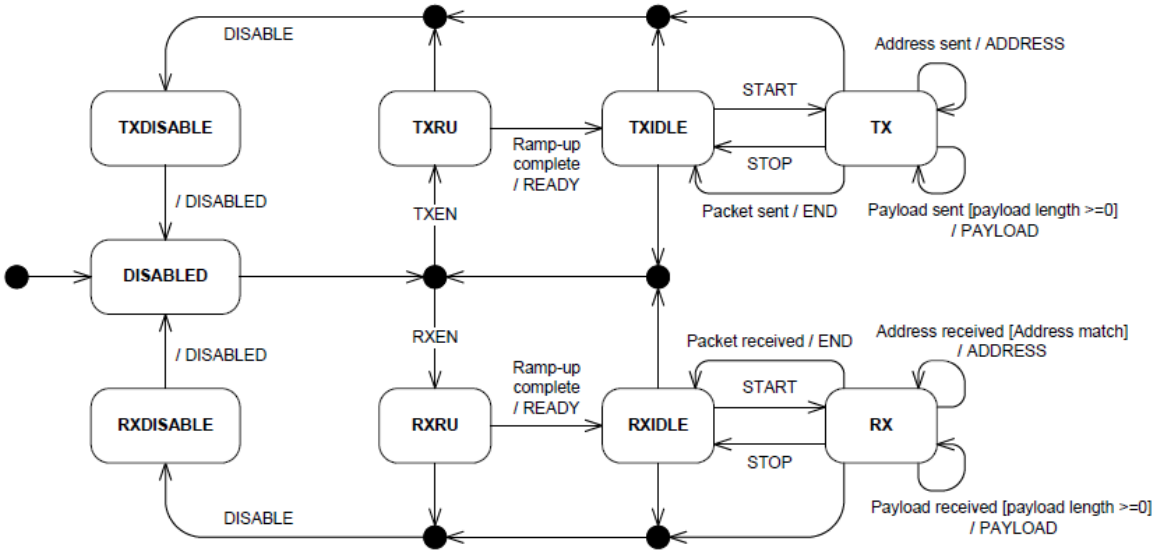


Figure 38: Bluetooth radio state of nRF52840, from nRF52840 Datasheet

State	Description
DISABLED	No operations are going on inside the radio and the power consumption is at a minimum
RXRU	The radio is ramping up and preparing for reception
RXIDLE	The radio is ready for reception to start
RX	Reception has been started and the addresses enabled in the RXADDRESSES register are being monitored
TXRU	The radio is ramping up and preparing for transmission
TXIDLE	The radio is ready for transmission to start
TX	The radio is transmitting a packet
RXDISABLE	The radio is disabling the receiver
TXDISABLE	The radio is disabling the transmitter

Table 10: Radio states description, from nRF52840 Datasheet

The nR52840 SoC provides registers named TASKS and EVENTS. A TASK register does some job like starting or stopping a module and an EVENT register is like a status register that indicates an event occurred. Then, the Radio state is analysed using the following peripherals:

- **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 registers.
- **RADIO** transceiver used for the BLE transmission.

The RADIO's EVENTS are connected directly to the GPIOS using PPI. Three GPIOS are used to represent The RADIO's states.
RENAME RTX TRANSCEIVER

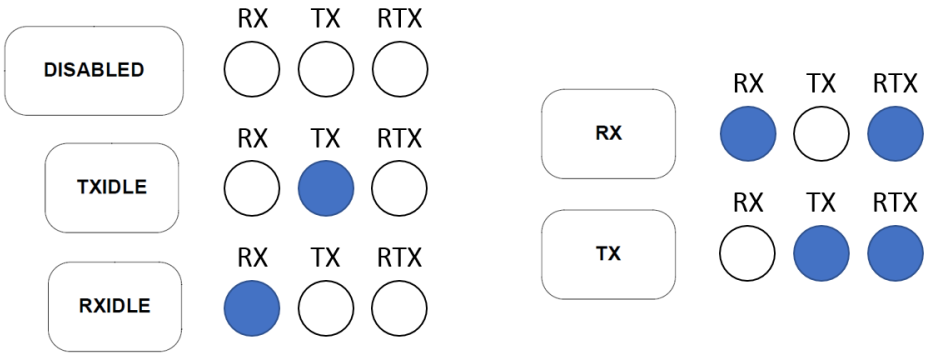


Figure 39: GPIOs enabled for each radio's states

Another EVENT, CRCOK, is used to see if the packet received was wrong. Finally, it is possible to see:

- When the device receives a packet
- When the device sends a packet

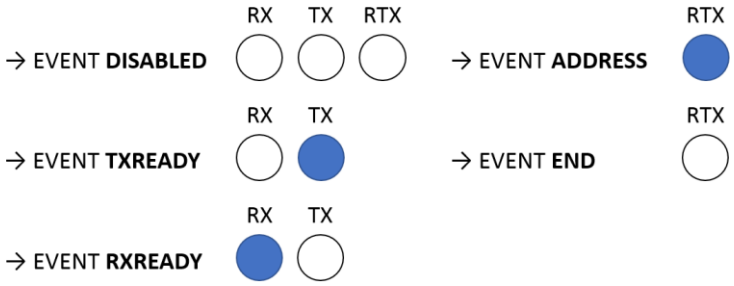


Figure 40: GPIOs enabled and disabled for each event

8.3.2.7 Stack Propagation Delay, APP to LL

8.3.2.8 Why

The stack propagation delay is the time for BLE driver to prepare data and push a packet within the BLE Link Layer TX Buffer. Then, this packet is sent when the peripheral receives a connection event.

Measuring the stack propagation delay allows to see how much time a process is locked to send data. On the other hand, it allows to see if BLE driver is still able to handle packets without error.

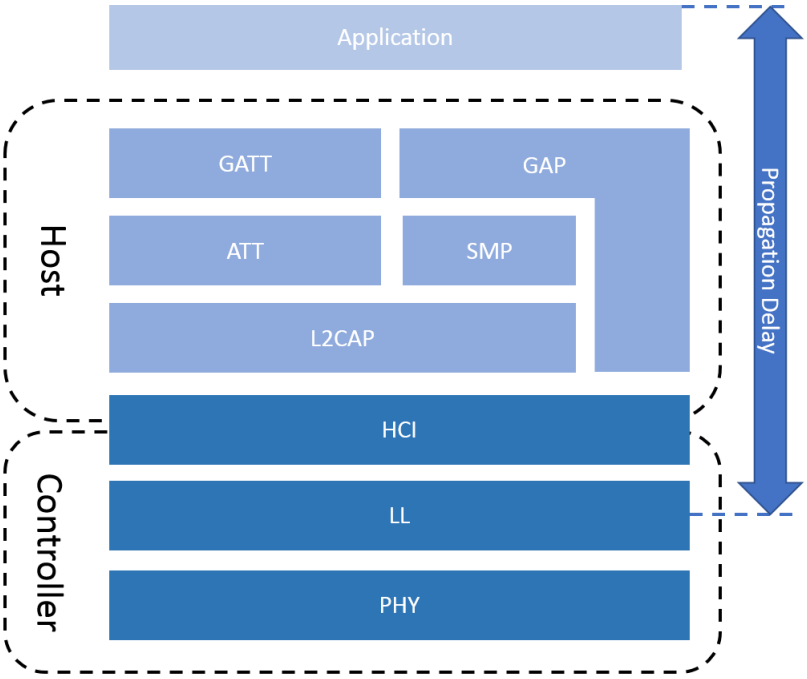


Figure 41: Layers include within the propagation delay

8.3.3 Expected Results

The stack propagation delay should not be influenced by the external interrupt. Secondly, the time should be almost the same between Zephyr RTOS and SoftDevice.

8.3.3.1 How

The stack propagation latency is measured using the GPIOs. Two GPIOs are used:

- **ERROR**, which indicates if an error occurred when a packet is pushed
- **REQUEST**, if notify or indicate with peripheral or write, read and enable CCCD with central

The GPIOs is turned on right before pushing a packet and it is turned off right after, the time when the signal is turn on represents the propagation delay.

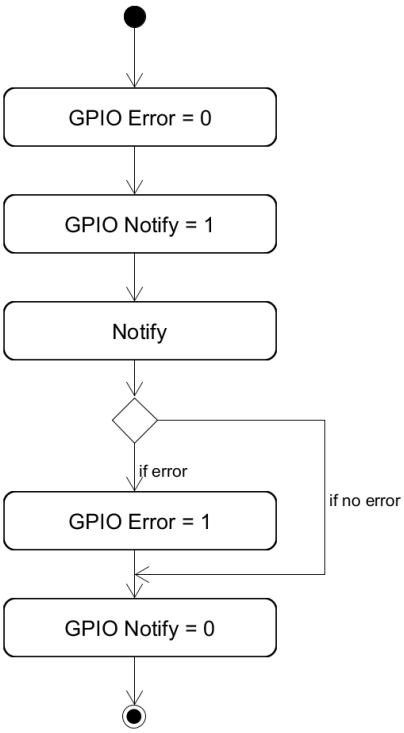


Figure 42: Measurement of stack propagation delay to notify

8.3.3.2 Other Interesting BLE Measurements

Because of a lake of time and the limitation regards the SoftDevice, some interesting measurements were not carried out:

- BLE stack propagation delay, LL to APP
- BLE radio interrupt latency
- BLE with encryption
- Integrity of the transmitted data

UPDATE THE FIGURE

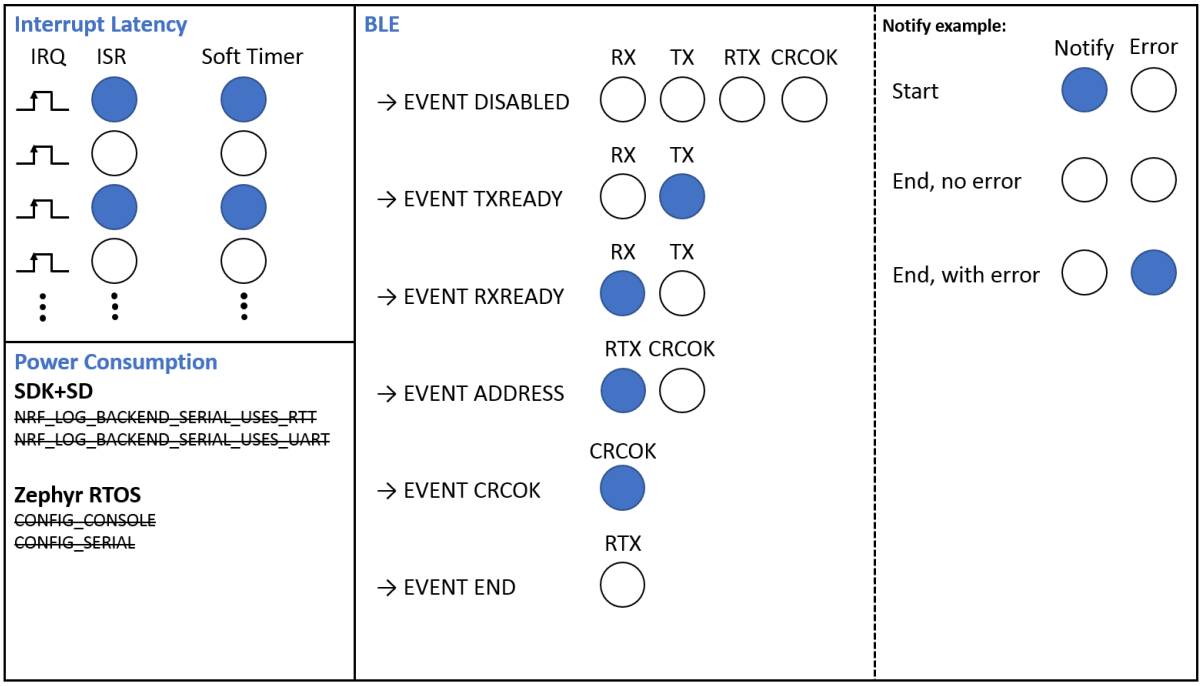


Figure 43: Measurements

8.4 Test Cases

The test cases define different condition under which one the interrupt latency, power consumption and BLE behaviour are measured. Testing under different cases allows to see how the systems behave when the conditions are changed.

The test cases are separated in four parts:

- **Basic measurements**, which measure the behaviour in sleep mode and the error to toggle GPIOs
- **Extension Board measurement**, which measure the power consumption and the interrupt latency when only the drivers are used
- **Peripheral measurements**, which measure power consumption and interrupt latency when advertising and connected.
- **Central measurements**, which measure the power consumption and interrupt latency when scanning and connected to several devices.

The data Rate and connection parameters are defined regards the use cases describe in the specifications, chapter **5. Specifications**.

	Case 1 High	Case 2 Balanced	Case 3 Slow
Conn. Interval	7.5 [ms]	50 [ms]	400 [ms]
Slave Latency	0	0	0
Conn. Timeout	500 [ms]	500 [ms]	1 [s]
Inspired By	Mouse	Glucose Meter	Heart Rate

Table 11: Use cases for measurements

8.4.1 Basic measurements

8.4.1.1 GPIO Error

The measurements performed determine the error to toggle, set and clear a GPIO. it is important to measure the error for the latency measurement.

To measure this error, two pins are toggled, set or clear and the error measured is the interval between the outputs of the pins.

8.4.1.2 Sleep mode Power Consumption

The measurements performed determine the power consumption in sleep mode when no drivers are enabled.

To measure the power consumption in sleep mode, the devices do nothing and is directly sleeping when the application run. All the drivers are disabled.

8.4.2 Extension Board measurement

The measurements performed determine the behaviour of the systems when only the serial communication, SPI and I2C, are used. The configuration of the extension board is the same than the use case 1.

ACC Data Rate	5 [ms]
ADC Data Rate	4.16 [ms]
Soft Timer	10 [ms]

Table 12: Extension Board measurements configuration

The table below describes the measurements performed:

	Int. Off, Drivers Off	Int. On, Drivers On	Int. H, Drivers Off	Int. S, Drivers Off	Int. H&S, Drivers Off	Int. H, Drivers On	Int. S, Drivers On	Int. H&S, Drivers On
SWG			I	I	I	I	I	I
ACC		I				I	I	I
ADC		I				I	I	I
Timer				I	I		I	I
Power Consumption	I	I	I	I	I	I	I	I
Hard. Interrupt Latency			I		I	I		I
Soft. Interrupt Latency				I	I		I	I

Table 13: List of the measurements performed for Extension Board

When the interrupt latency is measured, the interrupt frequency increases until the system is no more able to responses.

8.4.3 Peripheral measurements

The measurements performed determine the behaviour of the systems when the device is in a BLE peripheral role and connected to a central.

	Case 1 High	Case 2 Balanced	Case 3 Slow
Conn. Interval	7.5 [ms]	50 [ms]	400 [ms]
Slave Latency	0	0	0
Conn. Timeout	500 [ms]	500 [ms]	1 [s]
Advertising	50 [ms]	50 [ms]	50 [ms]
MTU	23 Bytes	23 Bytes	23 Bytes
BLE TX Buffer	15	15	15
ACC Data Rate	5 [ms]	20 [ms]	100 [ms]
ADC Data Rate	4.16 [ms]	16.66 [ms]	65 [ms]
Soft Timer	10 [ms]	10 [ms]	10 [ms]
Inspired By	Mouse	Glucose Meter	Heart Rate

Table 14: Peripheral measurements Configurations

It is not interesting to measure with different advertising interval because the device doesn't need to absolutely maintain a connection. However, it is interesting to have a quick view of its behaviour when advertising.

The table below describes the measurements performed:

	Adv., Int. Off, Drivers Off	Adv., Int. Off, Drivers On	Conn., Int. Off, Drivers Off	Conn., Int. Off, Drivers On	Adv., Int. On, Drivers Off	Conn., Int. On, Drivers Off	Conn., Int. On, Drivers On
BLE	I	I	I	I	I	I	I
SWG					I	I	I
ACC		I		I			I
ADC		I		I			I
Timer					I	I	I
BLE advertising	I	I					
BLE connected			I	I	I	I	I
Power Consumption	I	I	I	I	I	I	I
Advertising Interval					I		
Hard. Interrupt Latency					I	I	I
Soft. Interrupt Latency					I	I	I

Table 15: List of the measurements performed for Peripheral

The send the value of a sensor, the notification is usually used. It is why all the values are notified and not indicated. Secondly, it only influences the measure of the stack propagation delay with a slight different, event not notable.

When the interrupt latency is measured, the interrupt frequency increases until the system is no more able to responses.

8.4.4 Central measurements

The measurements performed determine the behaviour of the systems when the device is in a BLE central role and connected to one, four and eight peripherals.

	Case 1 High	Case 2 Balanced	Case 3 Slow
Conn. Interval	7.5 [ms]	50 [ms]	400 [ms]
Slave Latency	0	0	0
Conn. Timeout	500 [ms]	500 [ms]	1 [s]
Scan Window	50 [ms]	50 [ms]	50 [ms]
Scan Interval	200 [ms]	200 [ms]	200 [ms]
MTU	23 Bytes	23 Bytes	23 Bytes
BLE TX Buffer	15	15	15
ACC Data Rate	5 [ms]	20 [ms]	100 [ms]
ADC Data Rate	4 [ms]	16.5 [ms]	65 [ms]
Soft Timer	10 [ms]	10 [ms]	10 [ms]
Inspired By	Mouse	Glucose Meter	Heart Rate

Table 16: Central measurements Configurations

As for advertising in the peripheral measurement, it is not useful to measure with different scan interval.

The table below describes the measurements performed:

	Scan, Int. Off	Conn., Int. Off, Periph. 1	Conn., Int. Off, Periph. 4	Conn., Int. Off, Periph. 8	Scan, Int. On	Conn., Int. On, Periph. 1	Conn., Int. On, Periph. 4	Conn., Int. On, Periph. 8
BLE	I	I	I	I	I	I	I	I
SWG					I	I	I	I
Timer					I	I	I	I
BLE Scanning	I				I			
BLE conn. 1 Peripheral		I				I		
BLE conn. 4 Peripheral			I				I	
BLE conn. 8 Peripheral				I				I
Power Consumption	I	I	I	I	I	I	I	I
Scan Interval					I			
Conn. Interval						I	I	I
Hard. Interrupt Latency					I	I	I	I
Soft. Interrupt Latency					I	I	I	I

Table 17: List of the measurements performed for Central

When the interrupt latency is measured, the interrupt frequency increases until the system is no more able to responses.

The case 1 is not used when 4 and 8 peripherals are connected to the central because the time of the connections to all the peripheral is larger than the connection interval. Hence, it is useless to measure when more than one peripherals are connected.

8.5 Annexes

8.6 Sources

http://infocenter.nordicsemi.com/pdf/nRF52840_OPS_v0.5.pdf
https://community.arm.com/processors/b/blog/posts/beginner-guide-on-interrupt-latency-and-interrupt-latency-of-the-arm-cortex-m-processors
http://www.bogotobogo.com/Embedded/hardware_interrupt_software_interrupt_latency_irq_vs_fiq.php
https://blogs.mentor.com/colinwalls/blog/2012/06/05/measuring-interrupt-latency/
http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/ka16366.html

9 Results

LEGENDE UNDER MEASUREMENTS RESULT

All the results have been analysed with and approved by Vinayak Kariappa Chettimada, developer for Nordic Semiconductor and Zephyr Project.

9.1 Equipment

The equipment and software used for the measurements are:

- Saleae Logic Analyzer
- nRF-Connect Desktop
- Power Profiler Kit Desktop

The Logic Analyzer samples at 16MHz and its error is ±62.5ns.

9.2 Procedure

Apart for the power consumption, all the other measurements are taken with the Saleae Logic Analyzer.

9.2.1 Power Consumption

EXPLANATION

9.2.2 Interrupt latency

The value of the interrupt latency is an average of several measurements taken at three different captures at different points of time. Each measurement is taken where Soft-timer and BLE communication occurs.

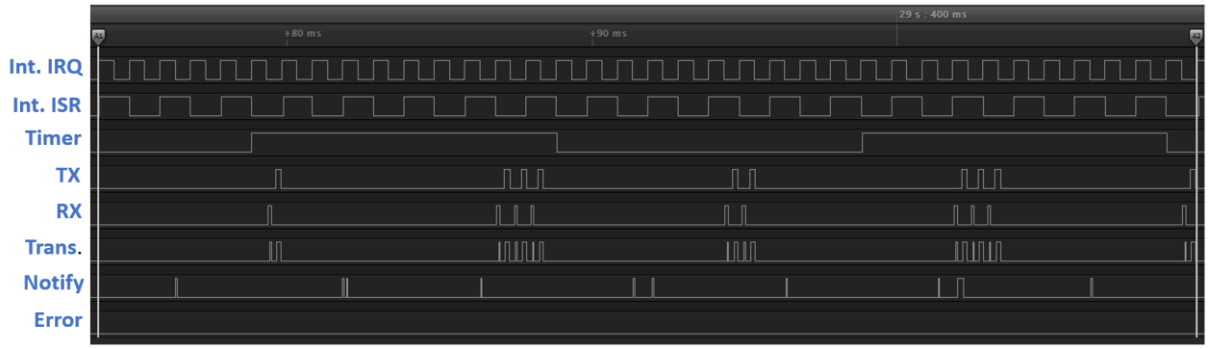


Figure 44: capture to measure the interrupt latency

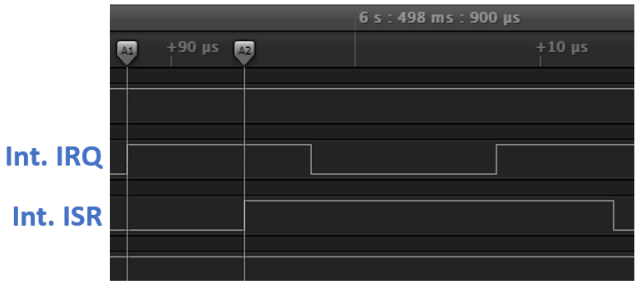


Figure 45: measurement of the interrupt latency

9.2.3 Bluetooth Low Energy Behaviour

The different values for BLE are an average of several measurements taken all the long the analyse. The different figures below show the measurement.

Figure 46: measurement of the stack propagation delay

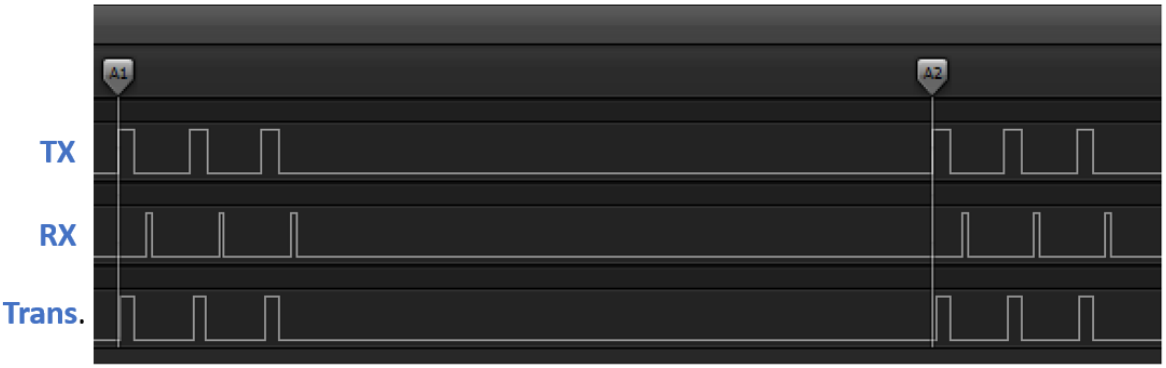


Figure 47: measurement of the advertisement interval

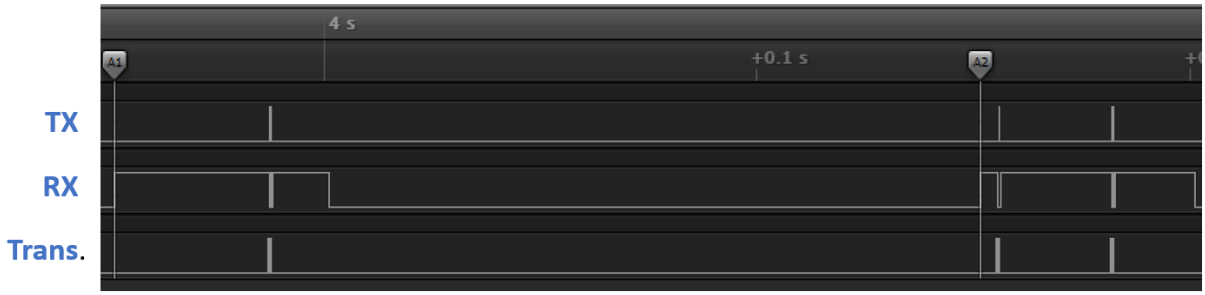


Figure 48: measurement of the scanning interval

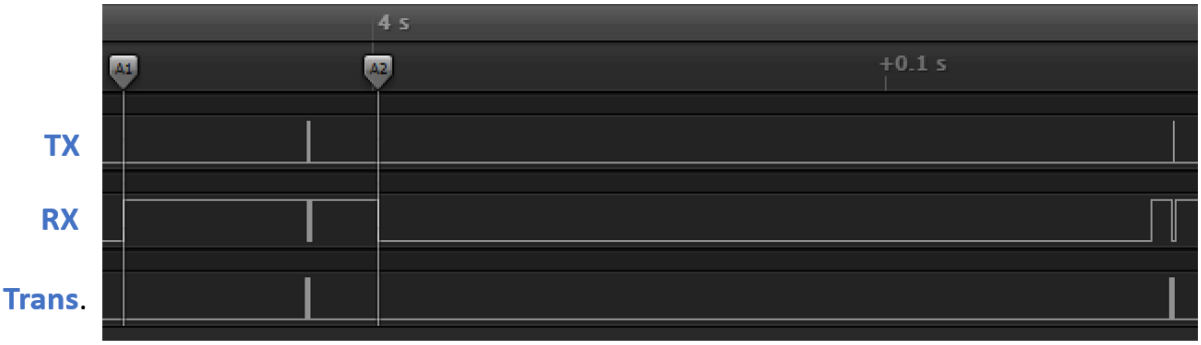


Figure 49: measurement of the scanning window

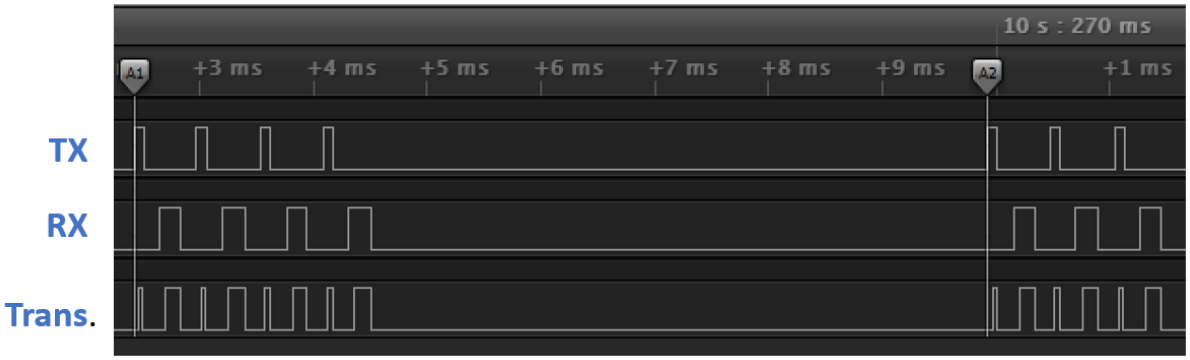


Figure 50: measurement of the connection interval

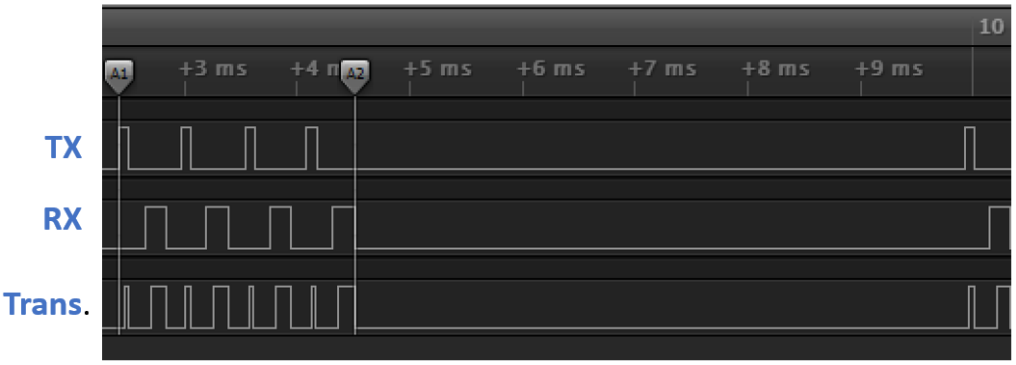


Figure 51: measurement of the connection event time

9.3 Results Basic measurements

9.3.1 GPIO Error

The time to set the GPIO compared are the time of the different way used in the measurement code. For the interrupt latency, the toggle is used. For the BLE propagation stack delay, the set/clear is used

GPIO Measurements Errors		
Registers	0	ns
Set / Clear	62.5	ns
Toggle	187.5	ns

Figure 52: Results of the GPIO errors

The error is the same for SD+SDK and Zephyr RTOS.

9.3.2 Sleep mode Power Consumption

The measurements are taken with different ticks of Zephyr RTOS to see if it has an influence on the power consumption.

Sleep Power Consumption			
SD + SDK	8.42		µA
Zephyr RTOS 1000 ticks/s	8.83	9.01	µA
Zephyr RTOS 500 ticks/s	349.32	9.09	µA
Zephyr RTOS 100 ticks/s	173.22	9.09	µA
		Sleep Fixed	

Figure 53: Power consumption in Sleep mode

The first series of power consumption measurements with Zephyr RTOS where wrong because the parameter of Zephyr sleep function was wrong. A second series of measurements were taken and it is possible to see that they are almost no difference between Zephyr and SD+SDK when in sleep mode.

Unfortunately, Zephyr API didn't explain very well how to use the different sleep function. It has been report and normally it should be corrected.

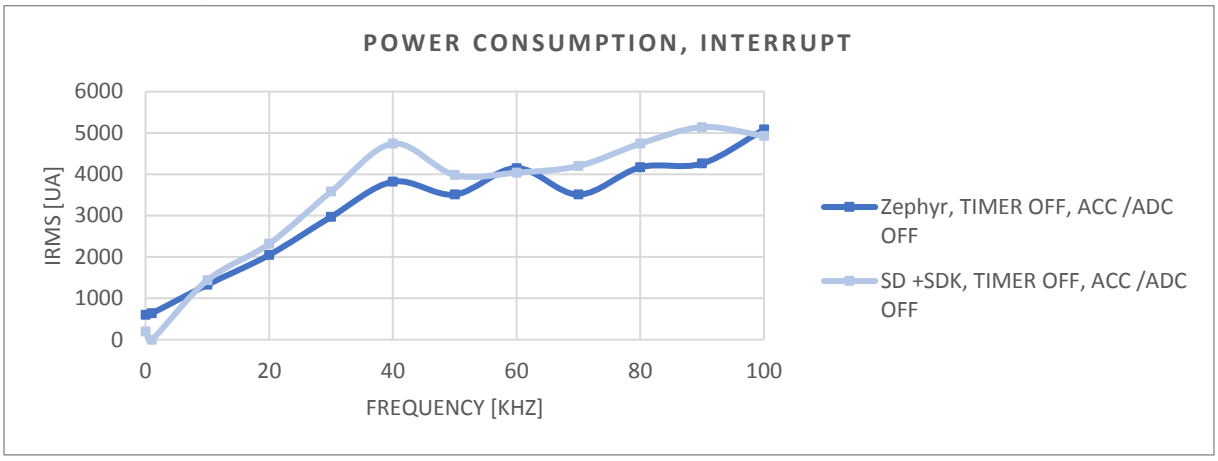
For the rest of the report, the values of SD+SDK are compared with the second series of Zephyr RTOS measurements.

9.4 Results Extension Board measurements

9.4.1 Power Consumption

For each case, the power consumption is measured with different interrupt frequencies.

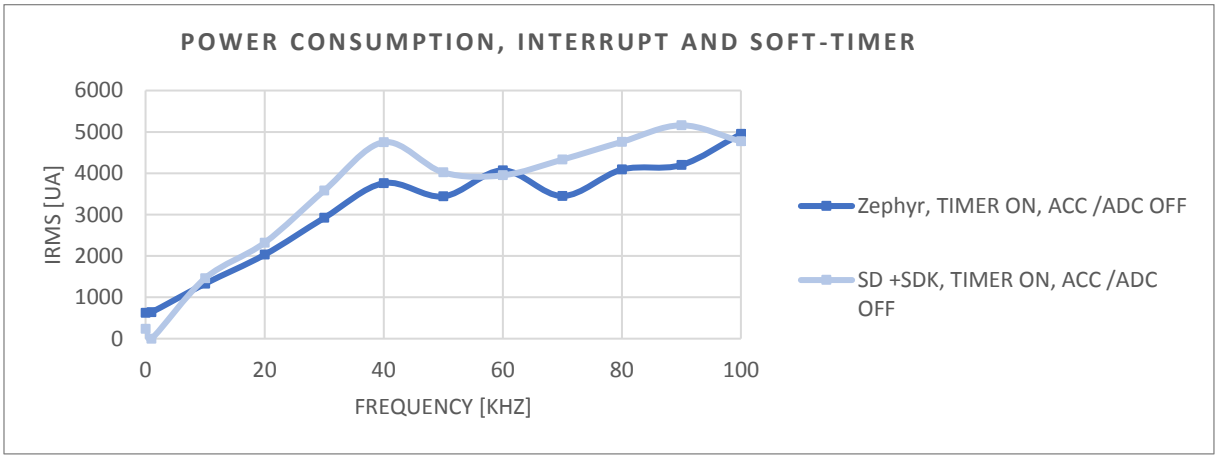
9.4.1.1 Interrupt enabled



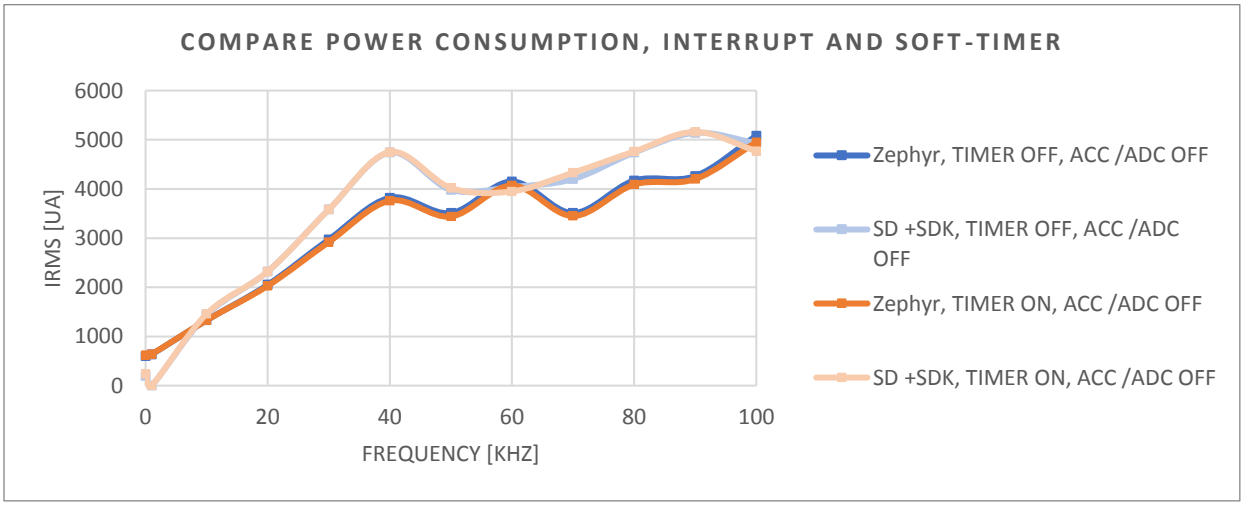
The power consumption of Zephyr is always lower than SD+SDK except at the beginning. This behaviour comes because when they are no interruption but the peripherals are enabled, there is always some noise with Zephyr RTOS and it why at this point its power consumption is higher.

This noise come certainly from the serial communication I2C, SPI and has been reported to be corrected.

9.4.1.2 Interrupt and soft-timer enabled

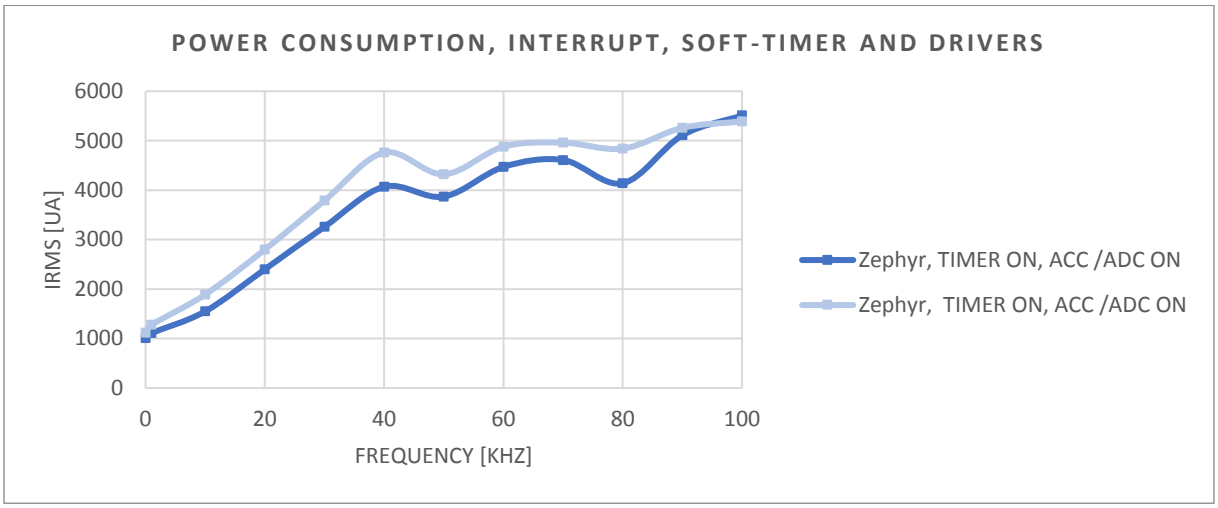


The Soft-timer does not really influence the power consumption as it is possible to see in the figure below.

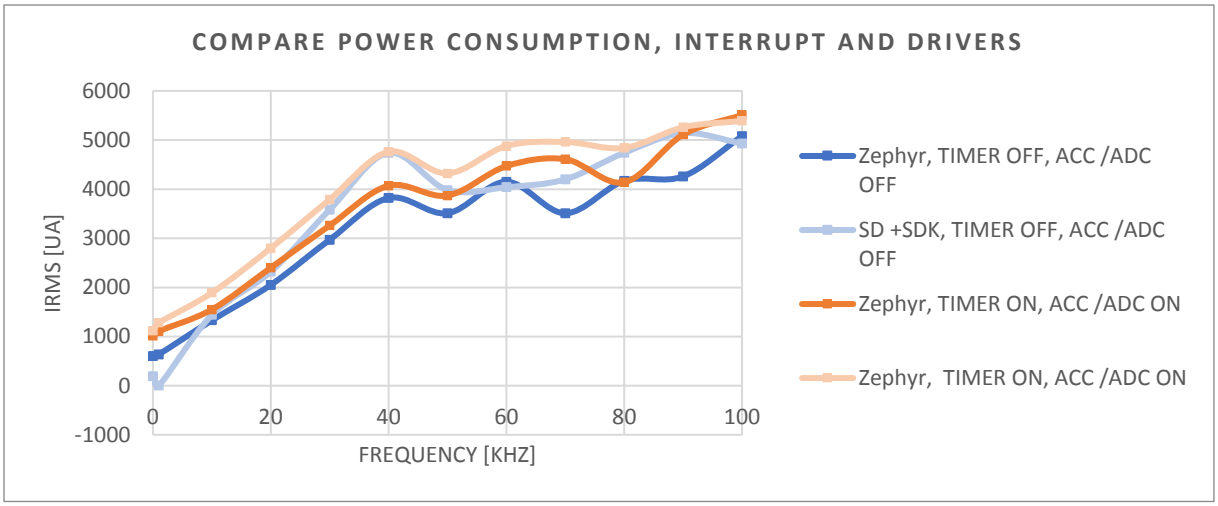


It is interesting to compare the power consumption but the use of a soft-timer is mainly to influence the interrupt latency and not the power consumption.

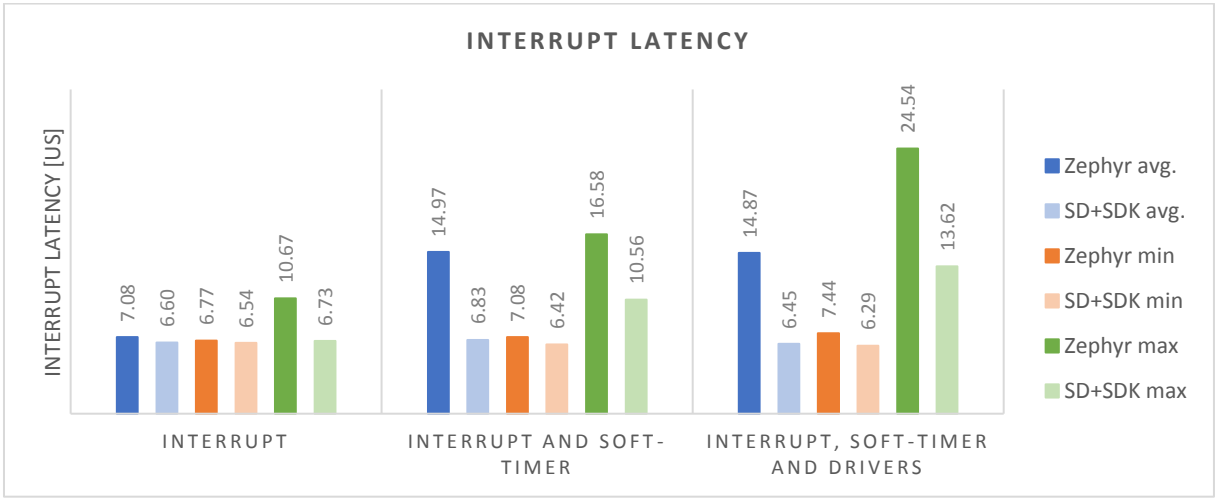
9.4.1.3 Interrupt, Soft-Timer and Drivers enabled



As when they are only the interrupts, the power consumption of Zephyr is slightly lower. When accelerometer and the A/D Converter, the power consumption increases but behaves in the same way.



9.4.2 Interrupt latency



The interrupt latency is always slower with SD+SDK because it is a Bare Metal system. It is possible to notice that with the SD+SDK, the interrupt latency is constant even when all the serial communication and soft-timer are enabled.

In contrast, the interrupt latency with Zephyr increases with the number of elements enabled. It starts at the first case at the same value than the SoftDevice and finished with higher value of 14.87µs.

The frequency of the interrupt does not influence the interrupt latency. Therefore, it is not useful to compare the values at different frequency.

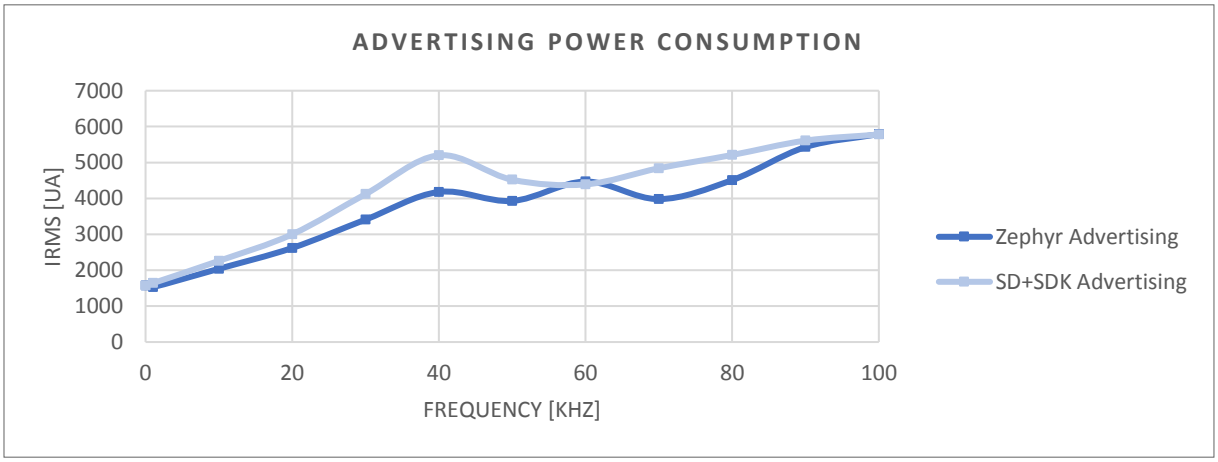
- Frequency where no more able to response

9.5 Results Peripheral measurements

9.5.1 Power consumption

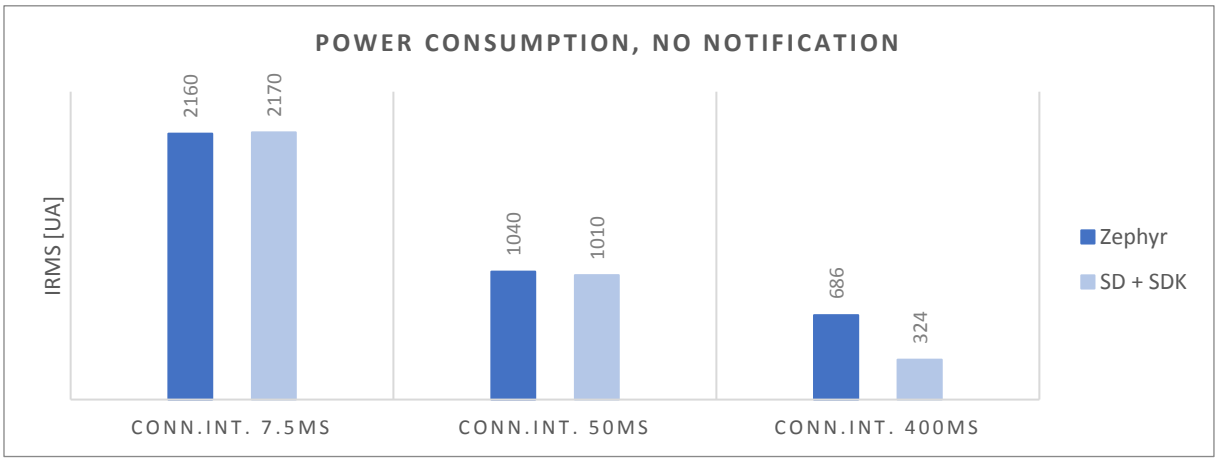
The power consumption when advertising and connected with notification are measured with different interrupt frequencies.

9.5.1.1 Advertising



As usual, the power consumption of Zephyr is slightly lower but the difference is not important.

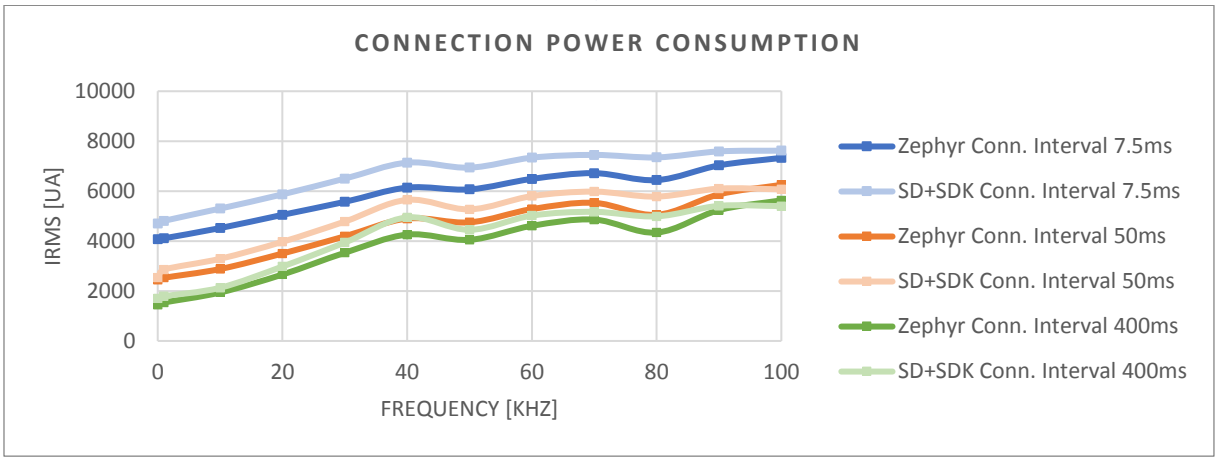
9.5.1.2 Connection without notification



We can see in this case that the power consumption is almost the same but when the connection interval is 400ms, the same problem occurs when measuring the power consumption of the Extension Board.

In this case, the power consumption with Zephyr is 686µA because of a residual noise. Usually, when the peripherals drivers are used, the power consumption can be lower than 600µA.

9.5.1.3 Connection with notification and interrupt



The power consumption decreases when the connection interval is slower. It is the typical behaviour of the BLE. The trend remains the same and the consumption of Zephyr is still slightly lower than the SoftDevice.

- Show Difference between transmission Zephyr and SoftDevice, no difference no influence

9.5.2 BLE Behaviour

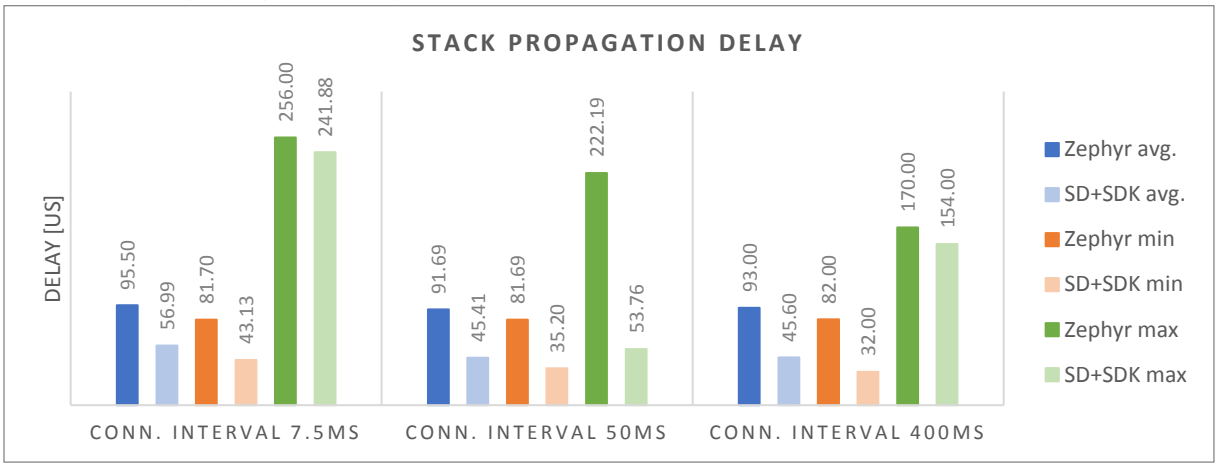
9.5.2.1 Advertising interval

Zephyr Advertising Interval	Meas.		Settings	
Avg.	55.70433	ms	50	ms
Min	50.96638	ms	50	ms
Max	59.97081	ms	50	ms

SD+SDK Advertising Interval	Meas.		Settings	
Avg.	54.6838	ms	50	ms
Min	50.16275	ms	50	ms
Max	59.98831	ms	50	ms

The advertising interval is correct and do not exceed 50ms + 10ms (advDelay). The variation of the time is due because the advDelay has to be generated randomly between 0 and 10ms.

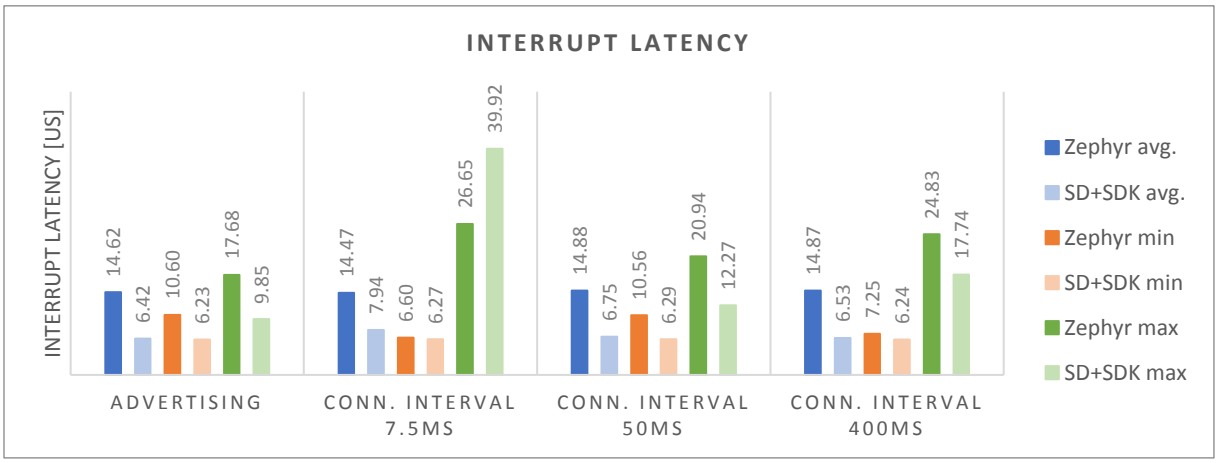
9.5.2.2 Stack propagation delay



The delay is constant with the connection interval. However, the time to push a packet with the SoftDevice is 80µs whereas it is 95µs with Zephyr.

- Maybe see the influence of difference frequency

9.5.3 Interrupt latency



The average interrupt latency with SD+SDK and Zephyr is constant between the different connection interval. The same result was observed without BLE.

However, the maximum interrupt latency is higher. It was about 15µs without the Bluetooth and 25µs in this case with the Bluetooth enabled.

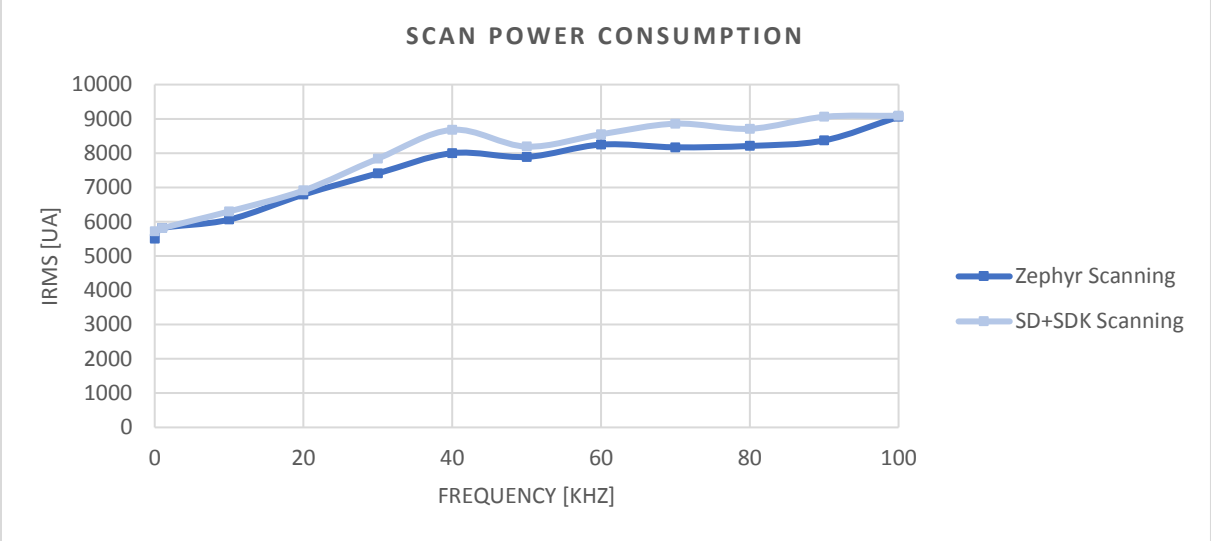
- Frequency where no more able to response

9.6 Results Central measurements

9.6.1 Power consumption

The power consumption when scanning and connected with several peripherals are measured with different interrupt frequencies.

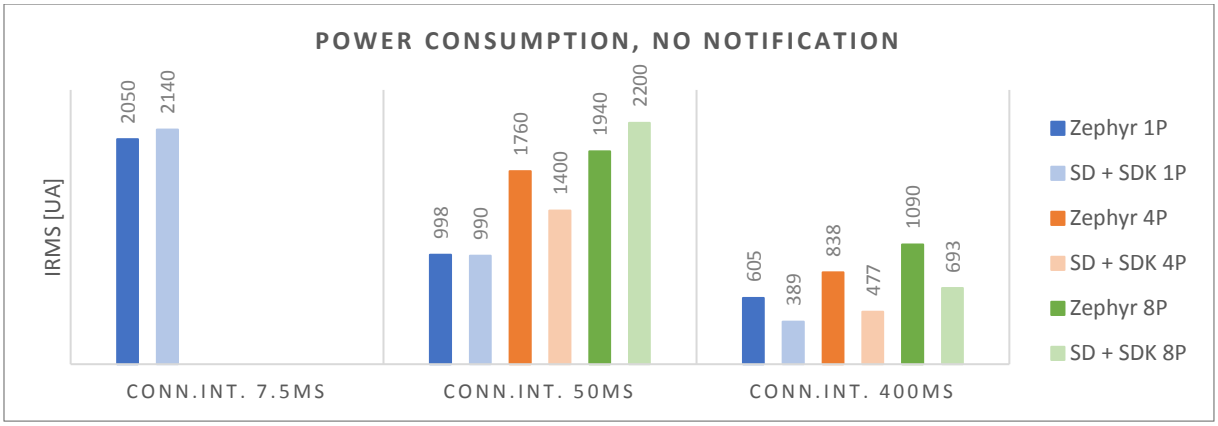
9.6.1.1 Scan



The behaviour remains the same than the previous measurements. The consumption of Zephyr is the lowest but the difference is thin.

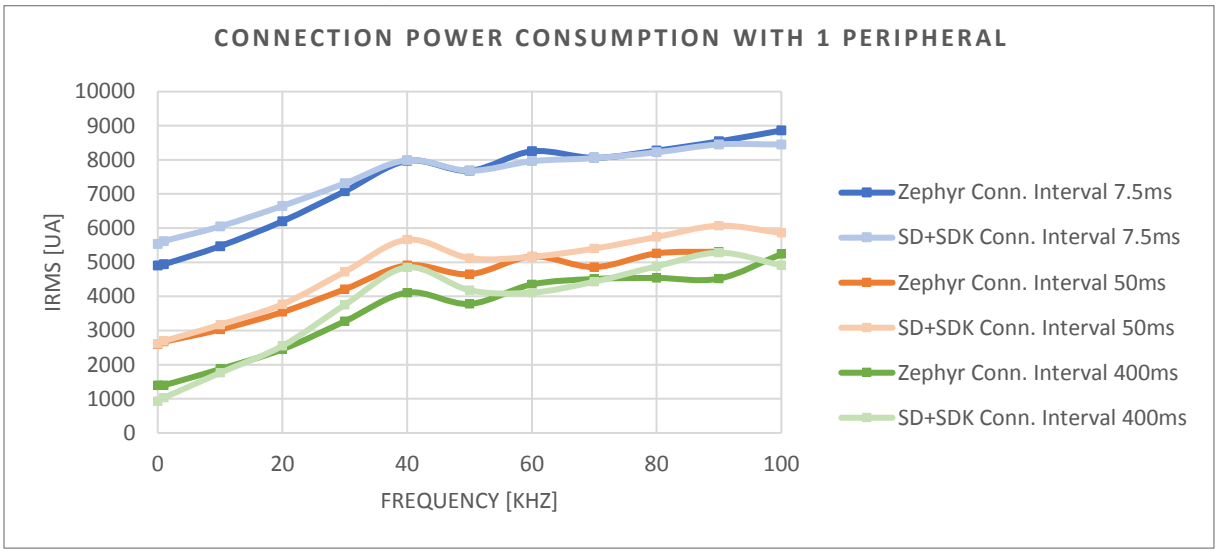
The power consumption is huge when scanning due to the scan window which enables the RADIO during a long period.

9.6.1.2 Connected without notification



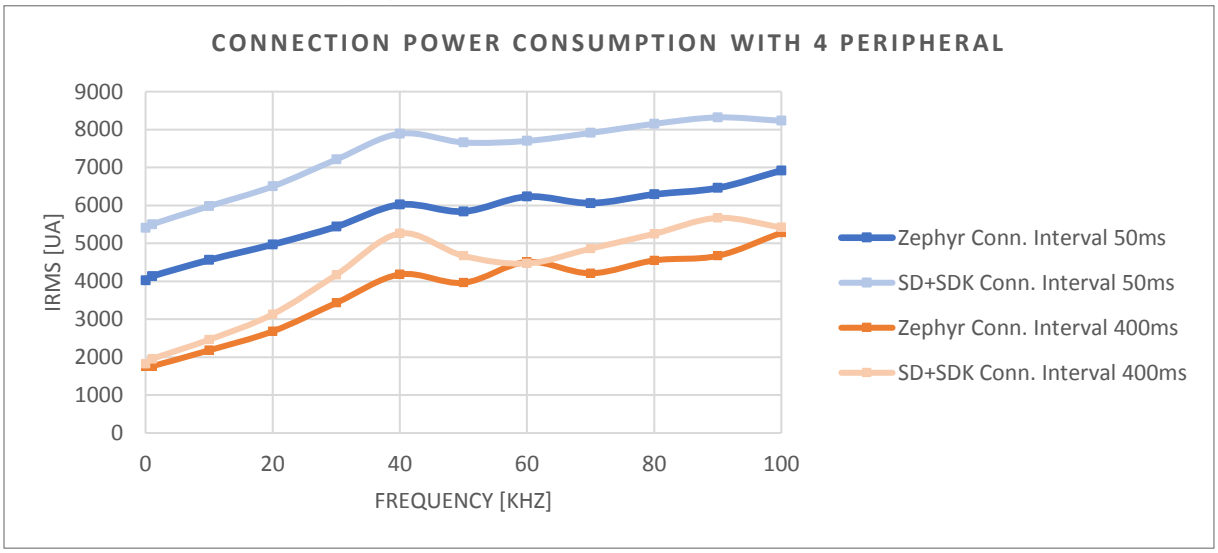
- No really difference, sometimes Zephyr higher and lower than SoftDevice.
- Same problem, zephyr never under 605uA.

9.6.1.3 Connection with 1 peripheral



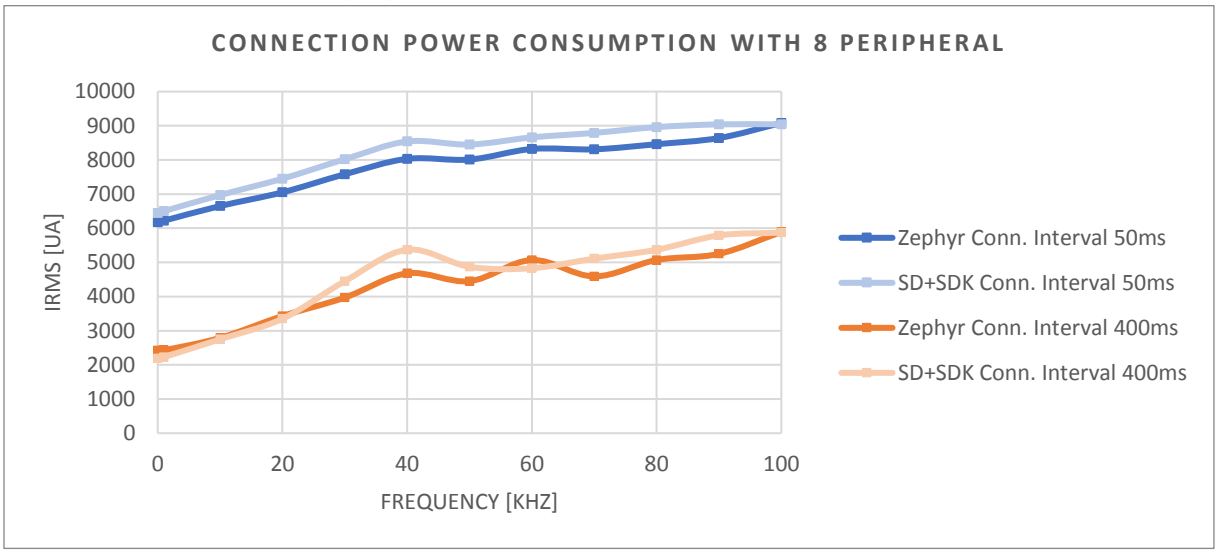
- Decrease with the connection event

9.6.1.4 Connection with 4 peripherals



- Why big difference when 50ms?
- No difference when 400ms

9.6.1.5 Connection with 8 peripherals



- Almost the same

9.6.2 BLE Behaviour

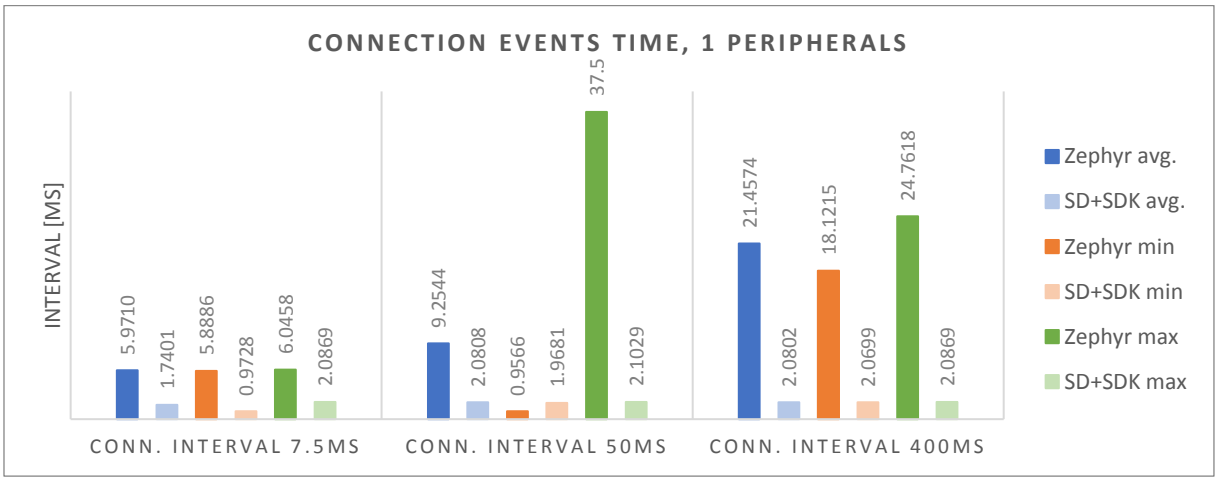
9.6.2.1 Scan interval

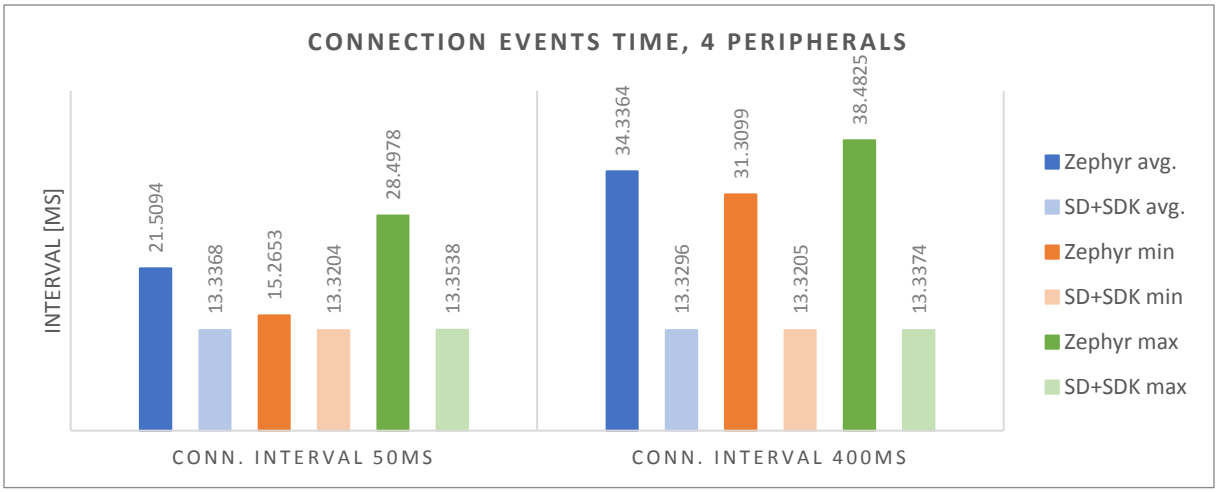
- Ok

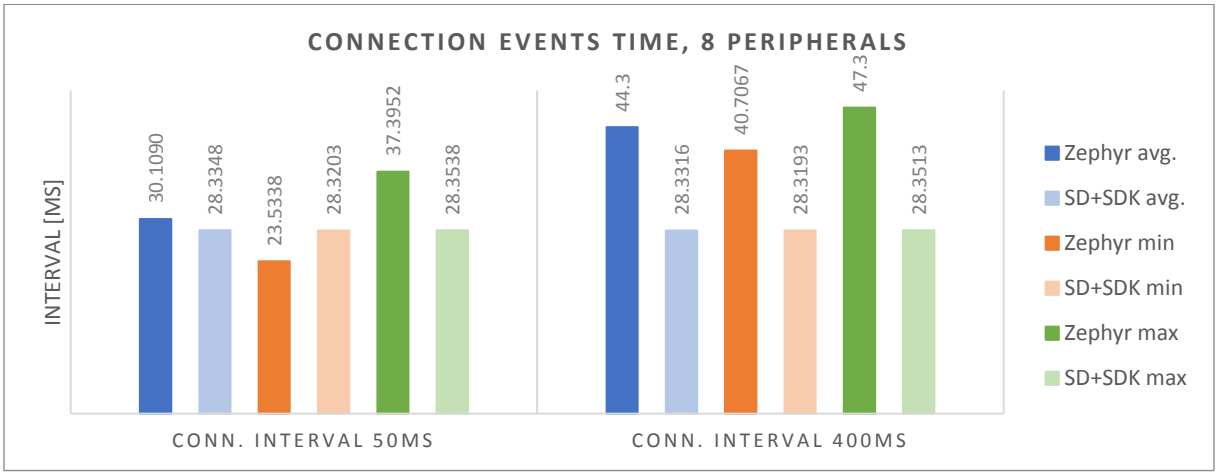
9.6.2.2 Connection interval

- Ok

9.6.2.3 Connection event time







It is possible to see a huge difference between the connection event time between Zephyr and SD. This difference is due to the way of each device handle the connection event. The way can be configurated and changed but the default process is used with the central.

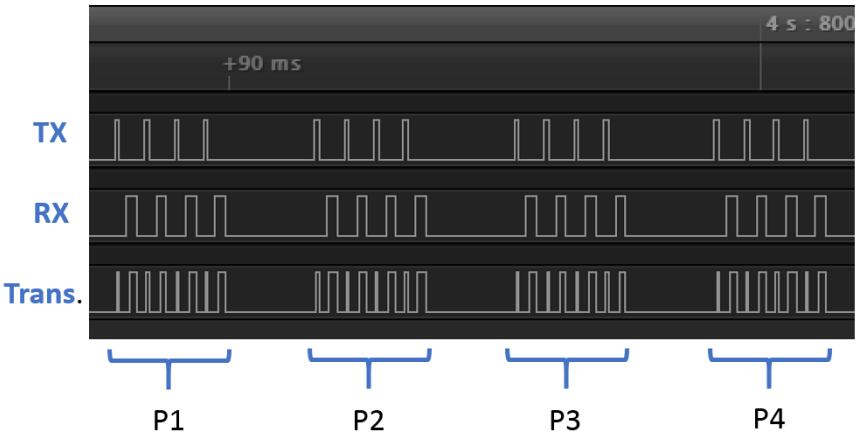


Figure 54: SoftDevice Connection event with 4 Peripherals

The SoftDevice send the connection event to each peripheral and allows them to send only four packets. Then the central close the connections

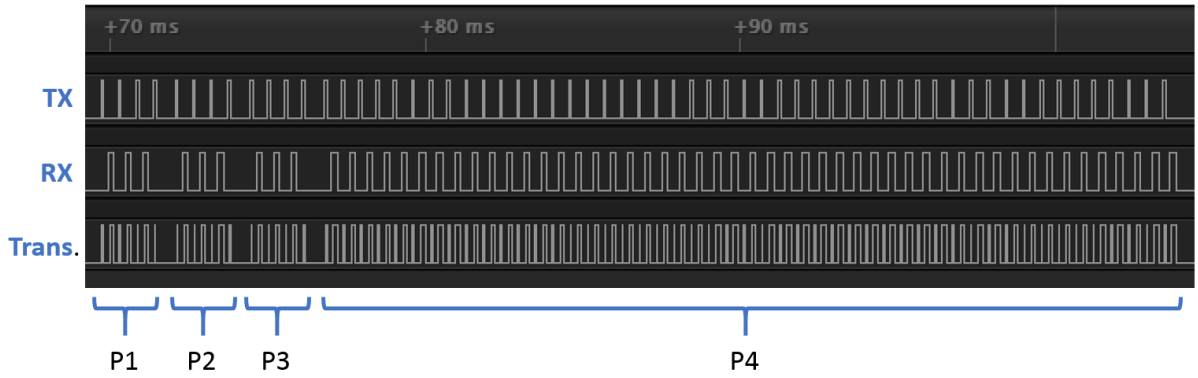


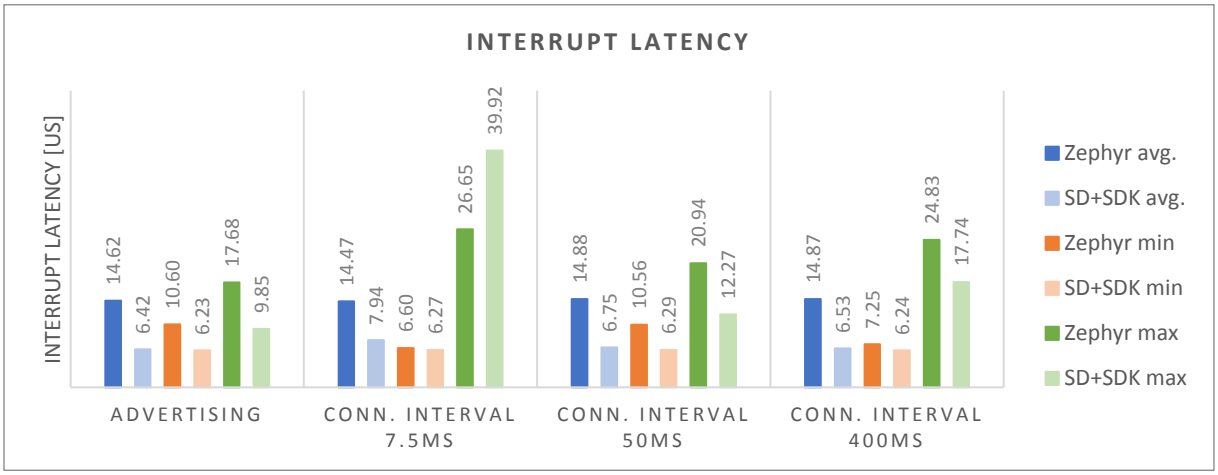
Figure 55: Zephyr Connection event with 4 Peripherals

Zephyr behaves in a different way. It allows the first peripherals to send only three packets and allows the last one, if time left before the next connection event, to send as many packets the peripherals has.

It is possible that the peripheral push other packets during the connection event. In this case, the packets are directly send to the central. Hence the connection between the peripheral and the central can be long.

The interrupt frequency has not influence on this measurement because the BLE controller and RADIO that performed the connection event have a higher priority than the GPIO.

9.6.3 Interrupt latency



The interrupt latency is the same than the interrupt latency measure with the peripheral.

9.7 Synthesis

It is possible to observe that the power consumption always decreases between an interrupt frequency of 40kHz and 50kHz. It is due to the interrupt latency.

As the interrupt latency maximum is about 20µs. The SoC will miss some interrupt from 50kHz (1/20 µs). Higher the interrupt latency is higher, more the power consumption decrease.

Regards the power consumption, there are no real difference between Zephyr RTOS and SD+SDK. However, Zephyr RTOS is generally lower than 500µA when all drivers are enables with a high interrupt frequency.

Nevertheless, Zephyr has a slight problem when all drivers are enabled without interrupt because it can be lower than 600µA. In this case, the power consumption of SD+SDK is better.

Regards the interrupt latency, SD+SDK is better, usually 6.5µs whereas Zephyr interrupt latency is 14µs. It is normal because Zephyr is an Operating System and handles thread and switching context. However, the interrupt latency is acceptable.

Regards the Bluetooth Low Energy, the advertising, scanning and connection intervals are correct and not perturb by the rest of the system. The links between the devices are correctly maintains.

The stack propagation delay with Zephyr, 95µs, is higher than 10µs of the delay with SD+SDK, 50µs. It is not the best results for Zephyr but it is not a large difference.

Regards the connection event time, it is difficult to compare because both systems behave in a different way.

TABLE WHICH RESUME EVERYTHINGS

9.8 Annexes

- table

10 Conclusion

11 Abbreviation

SoC: System on Chip
SD: SoftDevice
SDK: Software Development Kit
BLE: Bluetooth Low Energy
GAP General Access Protocol
GATT: General Attribute Protocol
ATT: Attribute Protocol
SMP: Security Manager Protocol
L2CAP: Logical link control and adaptation protocol
HCI: Host Controller Interface
LL: Link Layer
PHY: Physical Layer
MTU: Maximum Transmission Unit
ACC: Accelerometer
ADC: A/D Converter
SWG: Square Wave Generator
PPI: Programmable Peripheral Interconnect
GPIOTE: General Purpose I/O Tasks and Events