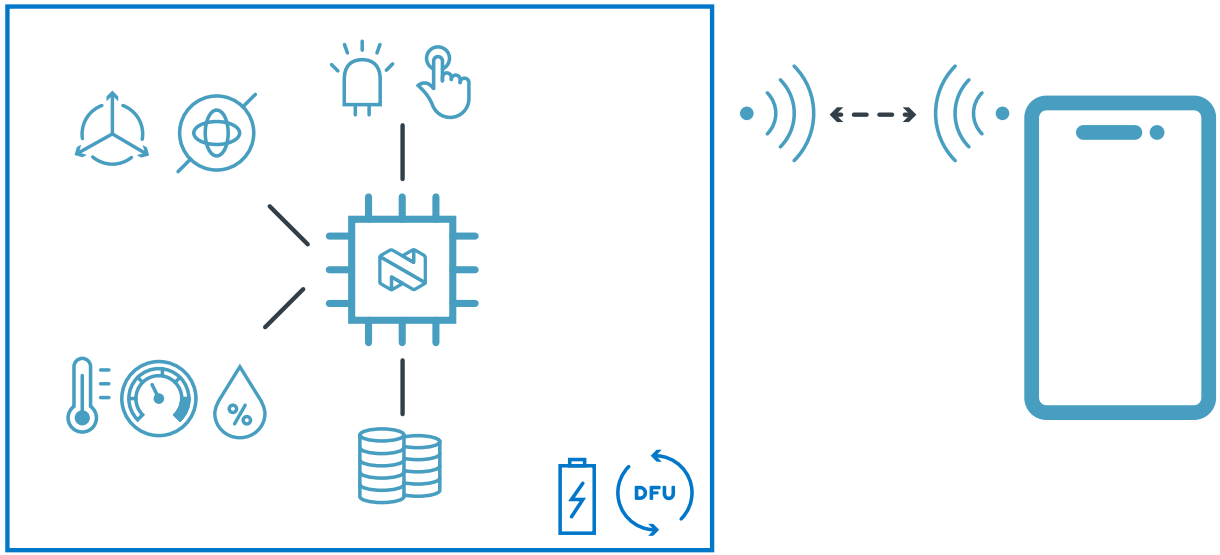
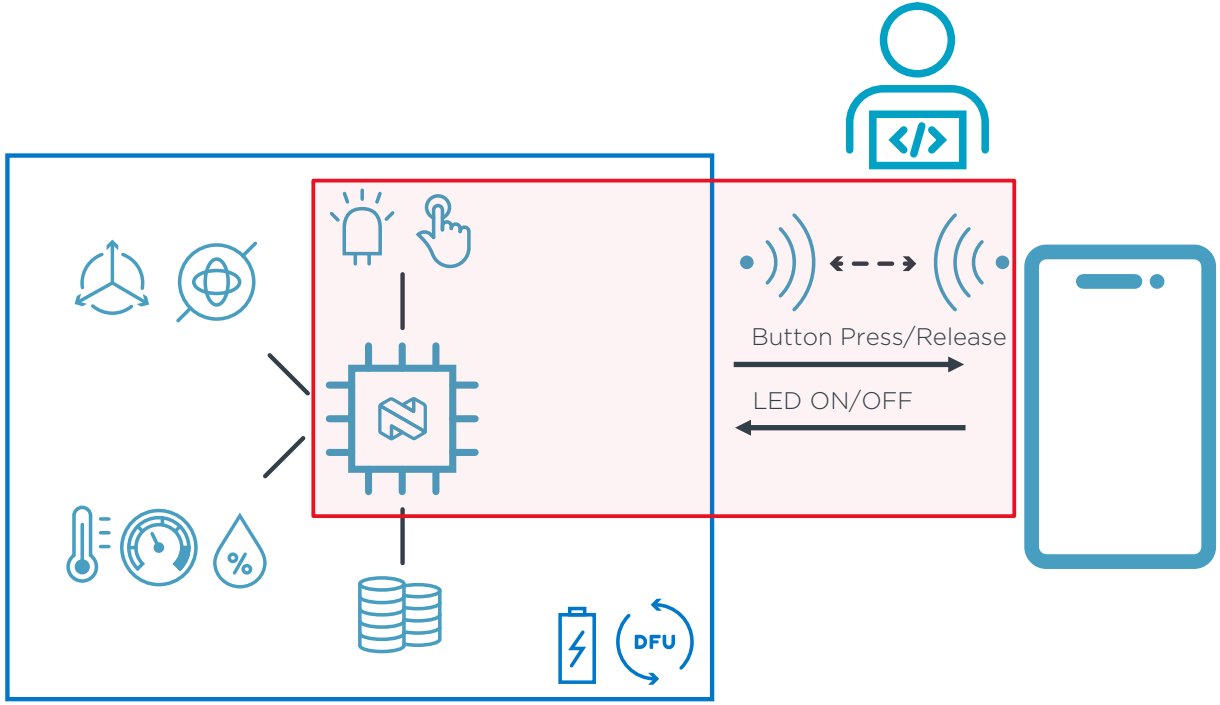
nRF Connect SDK Hands-on

Video 1 - Rapid prototyping with nRF Connect SDK

A typical Bluetooth LE application

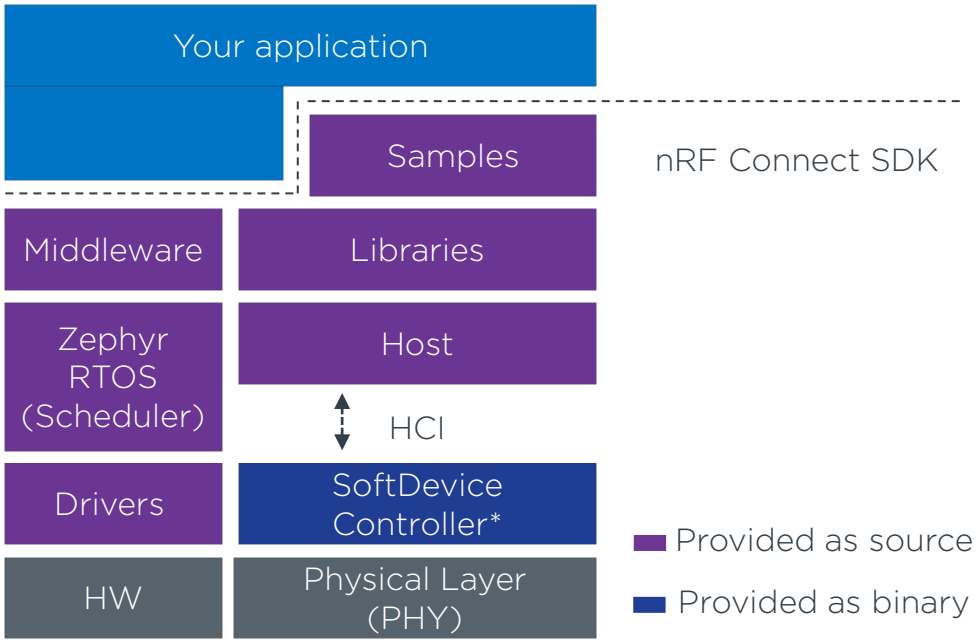


A typical Bluetooth LE application



Bluetooth LE support in nRF Connect SDK

- Qualified Bluetooth LE stack
  - Known for having the best mobile phone interoperability
- Rich set of Bluetooth LE libraries and services
- Large selection of samples
  - 42 (nrf\samples\bluetooth)
  - 76 (zephyr\samples\bluetooth)
- Applications
  - nRF Desktop , nRF5340 Audio
- Educational resources, rich set of tools, mobile apps, libraries



*) Open-source Zephyr Controller is also available, but the SoftDevice Controller is recommended

Quick Start

- Install nRF Command Line Tools
  - https://www.nordicsemi.com/Products/Development-tools/nRF-Command-Line-Tools
- Install nRF Connect for Desktop
  - https://www.nordicsemi.com/Products/Development-tools/nRF-Connect-for-Desktop
- Launch the Quick Start application in nRF Connect for Desktop
  - Follow the steps in Quick Start

Peripheral LBS sample

- LBS stands for LED Button Service
- The device is configured as a Bluetooth LE peripheral
- Learn more about Bluetooth LE connections
- Data exchange is done through two GATT characteristics
- Smart phone side
  - nRF Connect for Mobile (Android , iOS)
  - nRF Blinky (Android , iOS)
- Learn more about LBS

Service UUID

The 128-bit vendor-specific service UUID is `00001523-1212-EFDE-1523-785FEABCD123`.

Characteristics

This service has two characteristics.

Button Characteristic (`00001524-1212-EFDE-1523-785FEABCD123`)

Notify:

Enable notifications for the Button Characteristic to receive button data from the application.

Read:

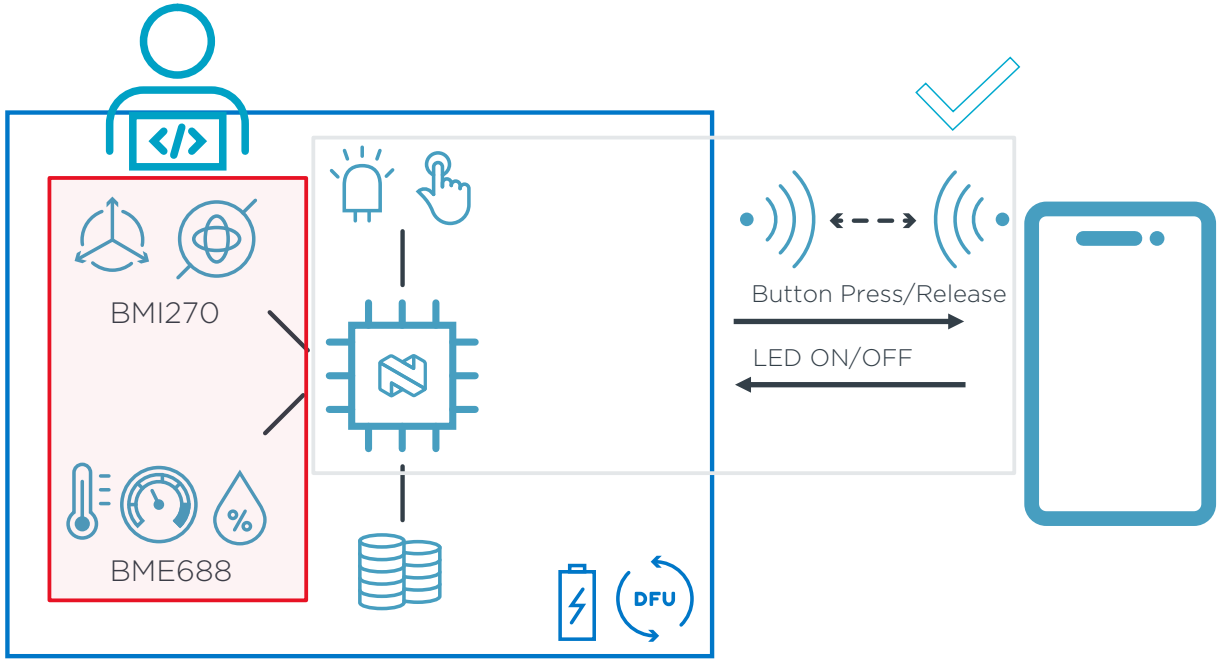
Read button data from the application.

LED Characteristic (`00001525-1212-EFDE-1523-785FEABCD123`)

Write:

Write data to the LED Characteristic to change the LED state.

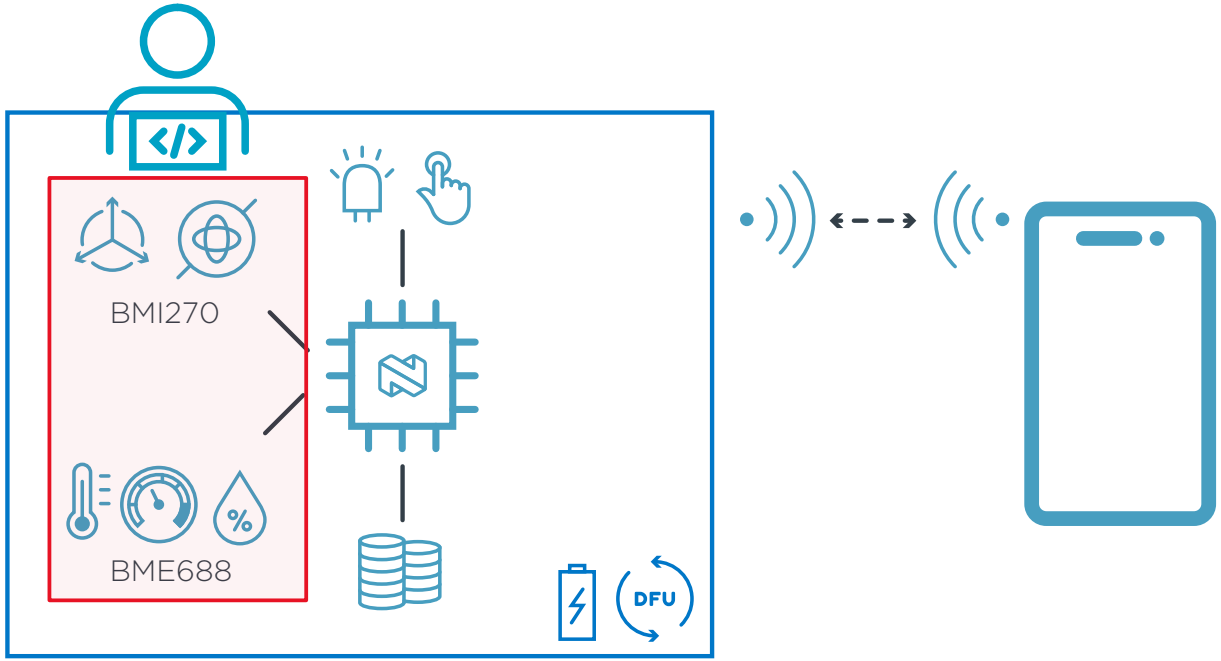
A typical Bluetooth LE application



nRF Connect SDK Hands-on

Video 2 - Sensors in nRF Connect SDK

A typical Bluetooth LE application



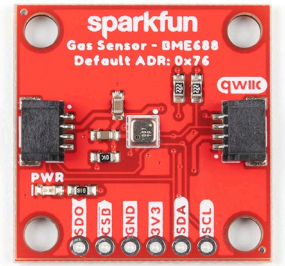
File structure

- Clear separation between source code and configuration files
- Software configurations are captured in Kconfig files
- Hardware configurations are captured in Devicetree files
- Source code: C/C++
- Further reading
  - Elements of an nRF Connect SDK application
  - Build and configuration system

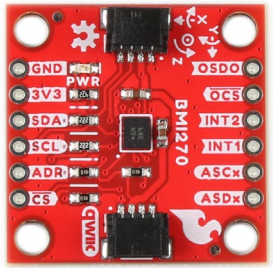
```
app/
├── boards              # Hardware specific software/hardware configuration
│   ├── <board_target>.conf
│   └── <board_target>.overlay
├── build
│   └── build_output
├── src                 # Source Code in C/C++
│   └── main.c
├── CMakeLists.txt
├── Kconfig
├── Kconfig.sysbuild
├── prj.conf             # Default application configuration file (Debugging enabled)
├── prj_minimal.conf
├── README.rst
├── sample.yaml
└── VERSION
```

Sensors in nRF Connect SDK / Zephyr

- Many sensor drivers are available in the SDK
- Unified API through the Sensor API
  - Ease of use & high level of portability
  - Data is exposed as sensor_value through channels
- We will first work on a simulated sensor
- Then, we will swiftly switch to physical sensors
  - Connected through an I2C bus
  - Environmental Sensor-BME688
  - 6DoF IMU - BMI270

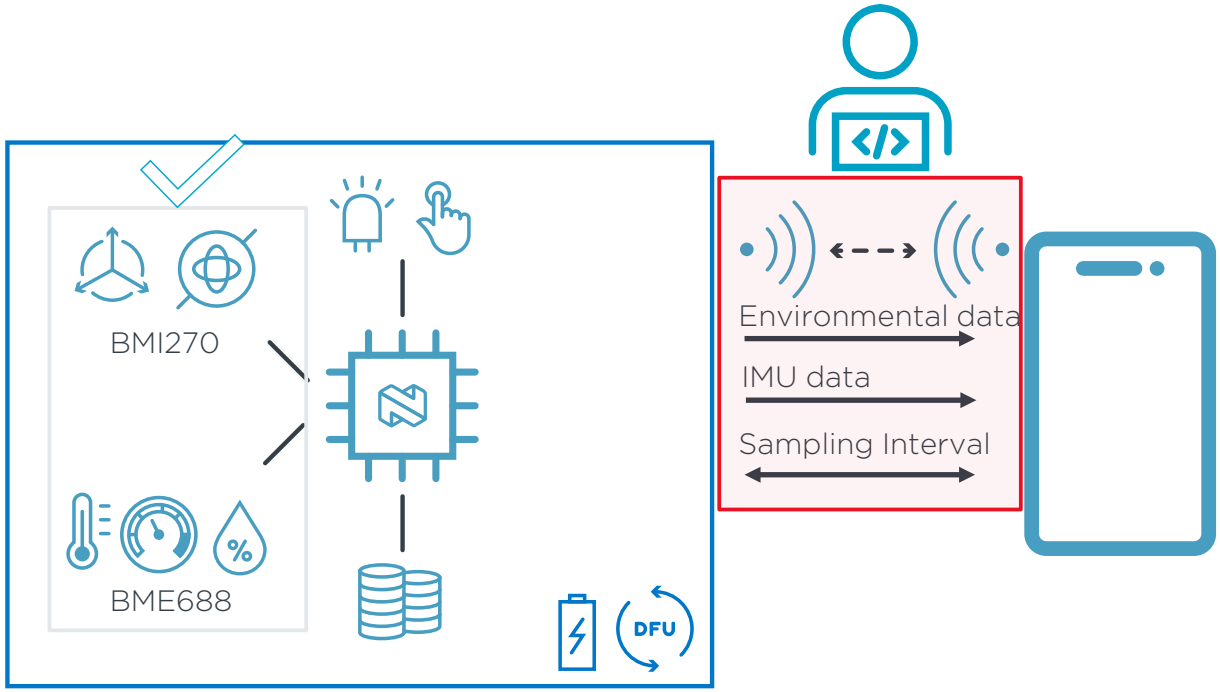


BME688



BMI270

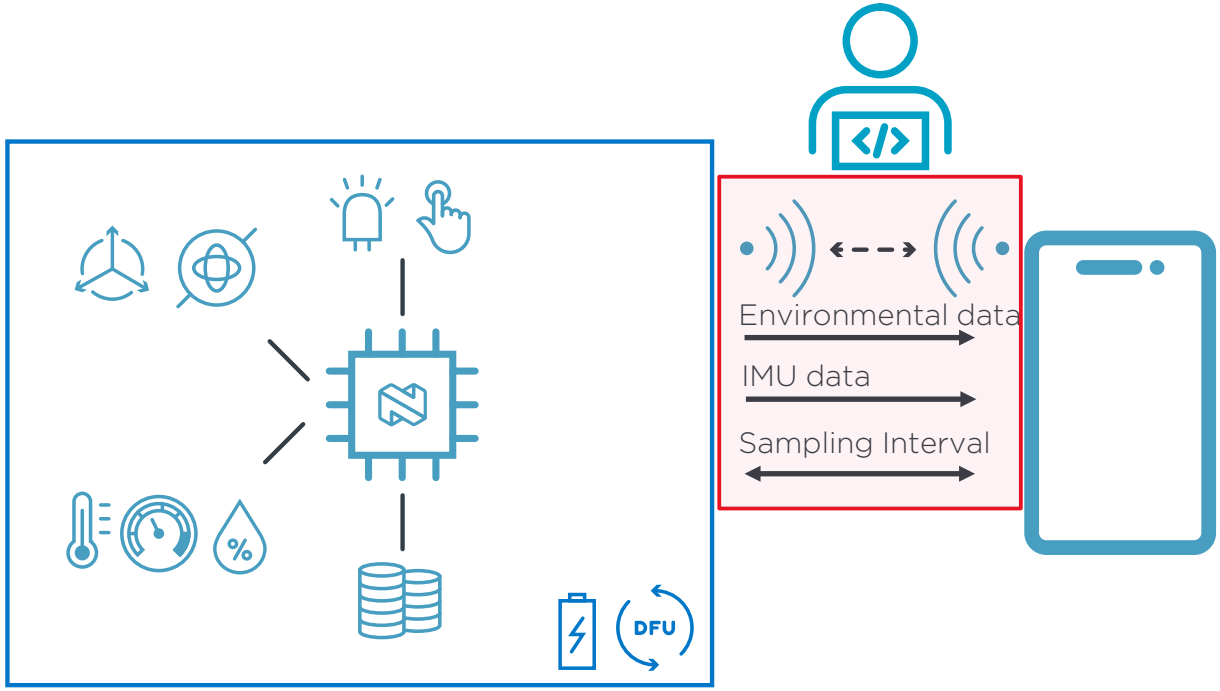
A typical Bluetooth LE application



nRF Connect SDK Hands-on

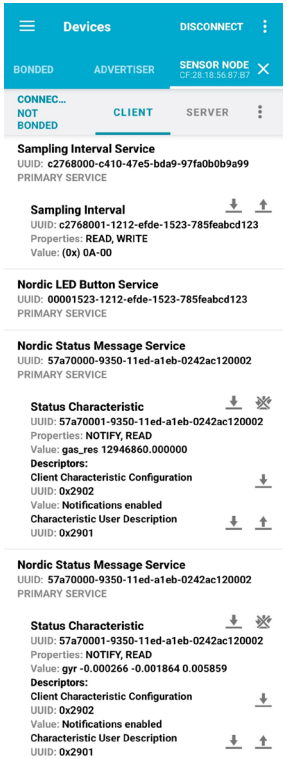
Video 3 - Send and receive data through Bluetooth LE

A typical Bluetooth LE application

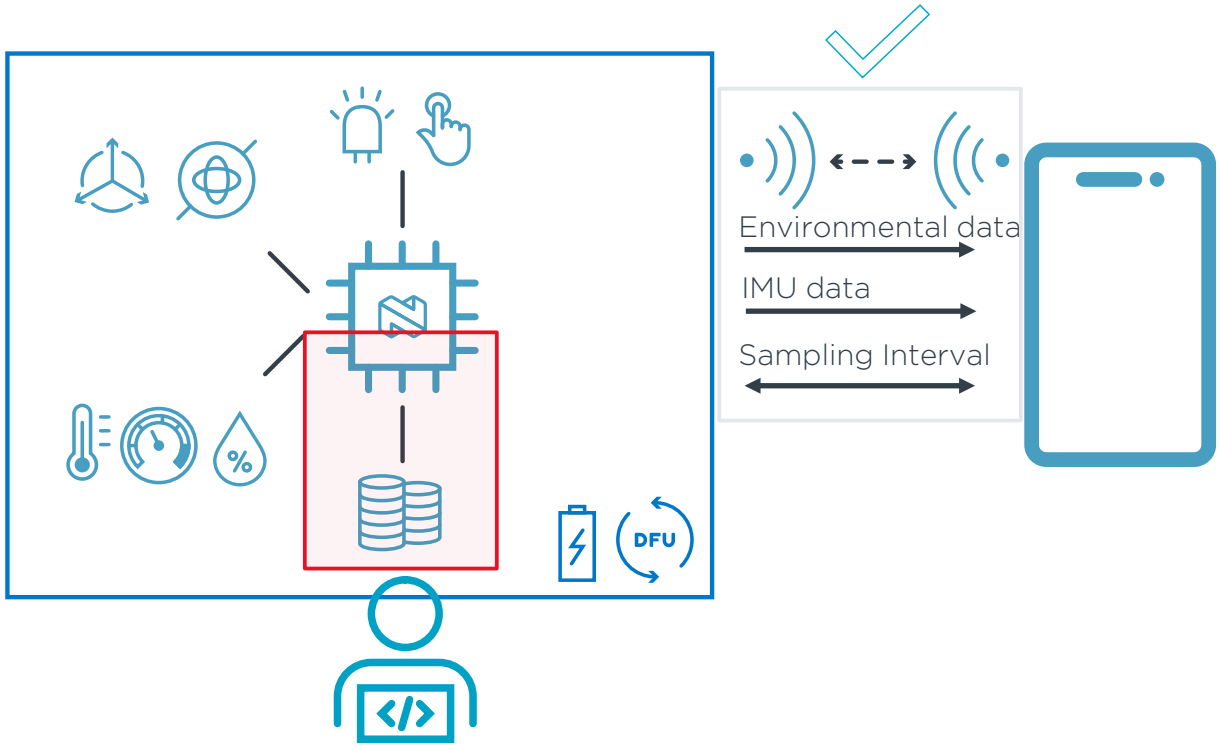


Data exchange through Bluetooth LE

- Sending sensors' data through a Bluetooth LE connection
  - Use Nordic Status Message Service (NSMS) to expose real-time sensor data over Bluetooth LE connection (read/notify operations)
    - One for environmental sensor or simulated sensor
    - One for IMU sensor or simulated sensor
- Controlling the sampling interval
  - Create a custom service for controlling the sampling interval (read/write operations)
- Further reading
  - Data exchange in Bluetooth LE



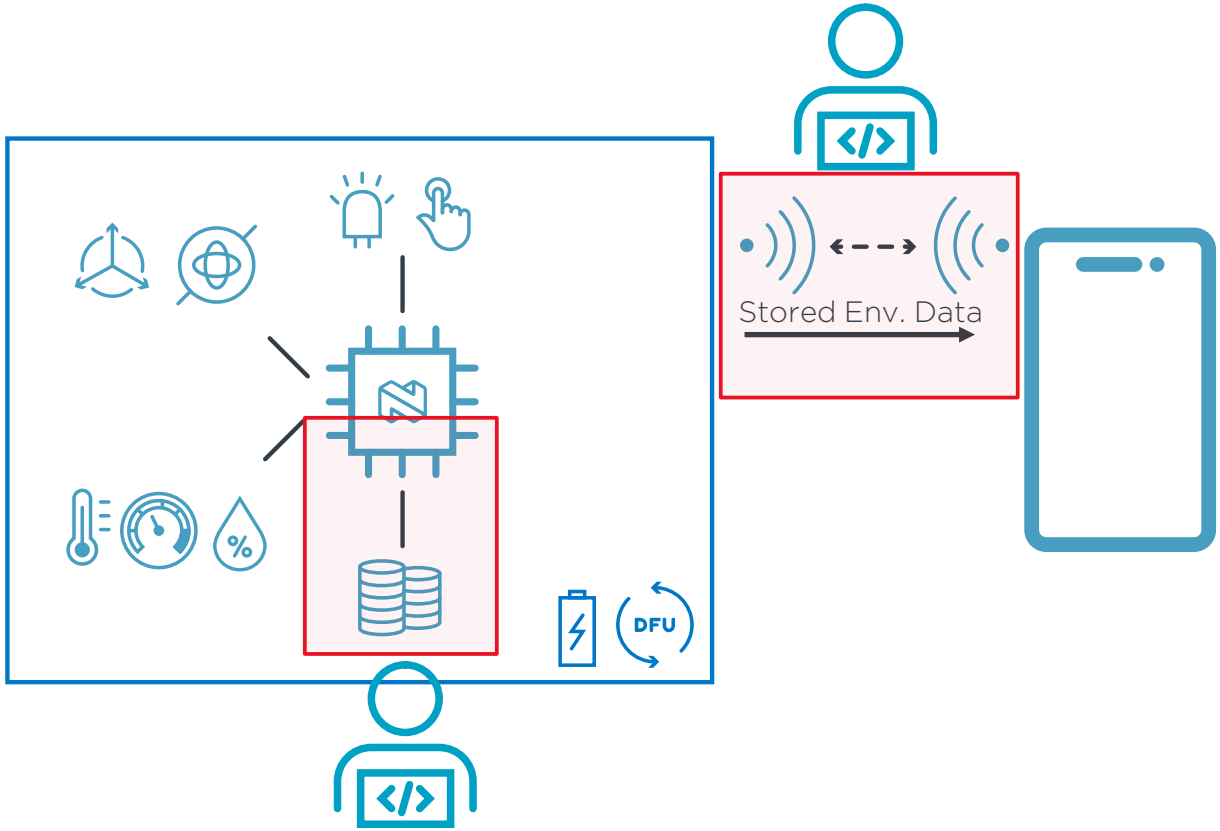
A typical Bluetooth LE application



nRF Connect SDK Hands-on

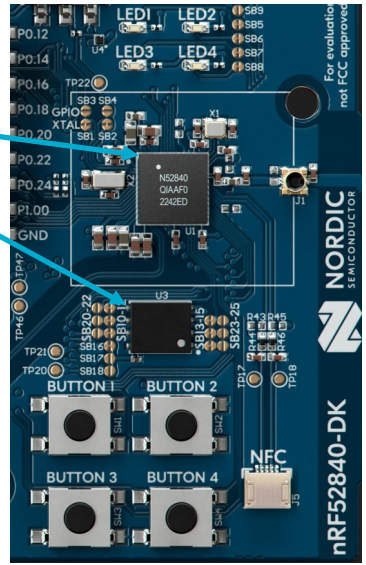
Video 4 - Storing data on non-volatile memory

A typical Bluetooth LE application



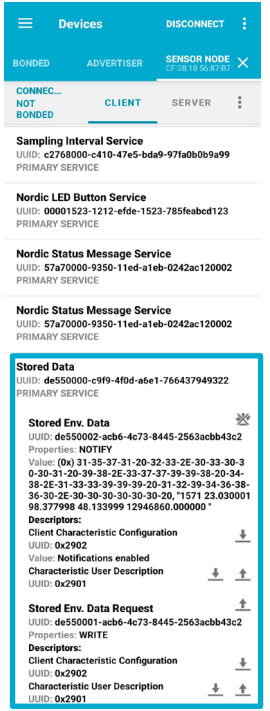
Storing sensor data on NVS

- Store the environmental sensor data on:
  - External non-volatile memory (on the DK)*
  - Internal non-volatile memory (inside the SoC)
- Create a thread dedicated to storing sensors' data `sensor_data_storage` to NVS
- Use the NVS subsystem in nRF Connect SDK to store the sensor data as id-data pair
- Pass the data from `sensor_data_collector` to `sensor_data_storage` through a message queue

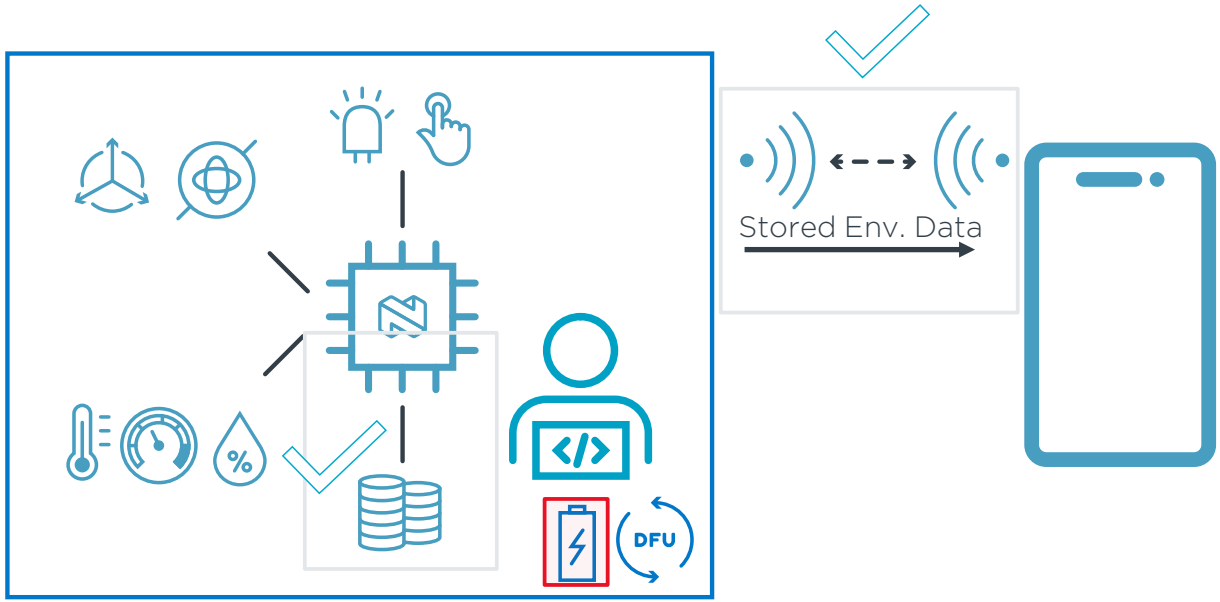


Getting the stored data via Bluetooth LE

- In Video 3, we got the current “real-time” data
- Now, we will get the stored data in NVS
- Create a custom service for “Stored Data”
  - Stored Env. Data Request (Write)
    - UUID: de550001-acb6-4c73-8445-2563acbb43c2
    - pass request as little-endian uint16_t
  - Stored Env. Data (Notify)
    - UUID: de550002-acb6-4c73-8445-2563acbb43c2



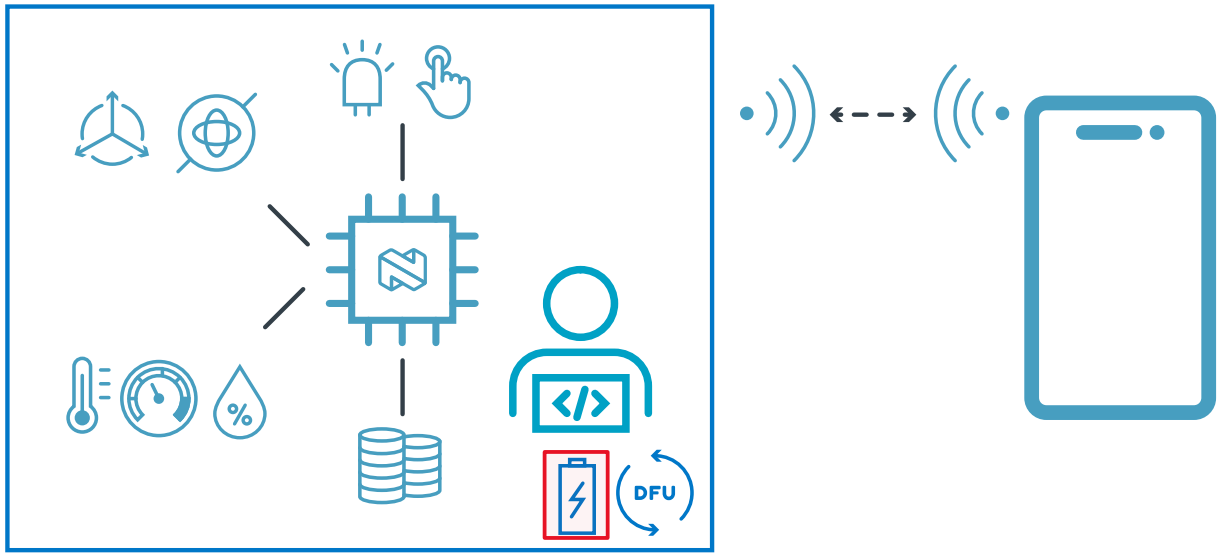
A typical Bluetooth LE application



nRF Connect SDK Hands-on

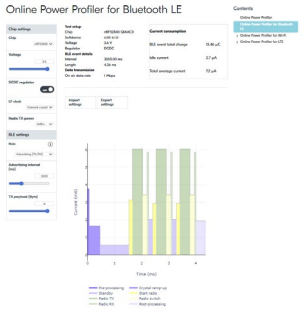
Video 5 - Power optimization and the PPK2

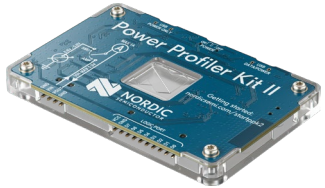
A typical Bluetooth LE application

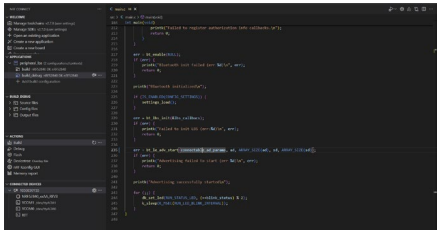


Power optimization and the PPK2

- Get an estimate
  - Online Power Profiler for Bluetooth LE
- Measure power consumption
  - Power Profiler Kit II (PPK2)
    - Preparing a DK for current measurement
    - Use the Power Profiler Desktop app.
- Optimize your application
  - Power optimization recommendations







Step 1 - Disable logging and unused peripherals

Logging enabled (**default**)

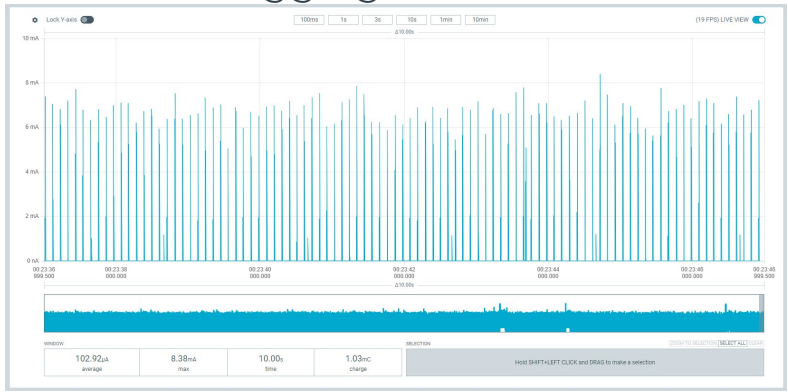
CONFIG_SERIAL=y

Average current = 776.59 µA

(Advertising TX/RX)

SoC: nRF52840

Logging disabled



CONFIG_SERIAL=n

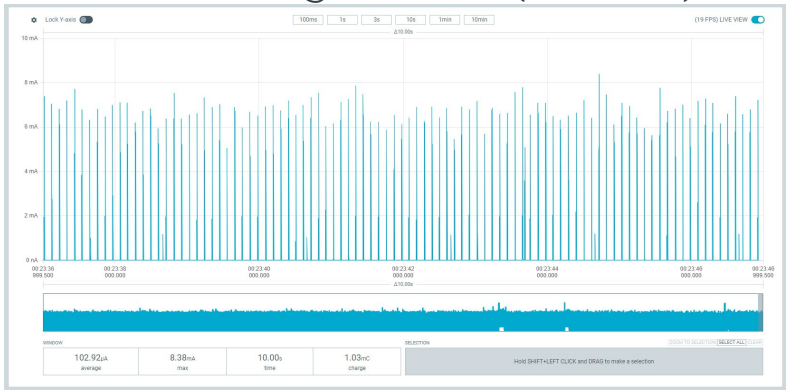
Average current = 102.92 µA

(Advertising TX/RX)

SoC: nRF52840

Step 2 - Change the advertising interval

Advertising interval (**default**)



~ 100 ms

Average current = 102.92 µA

(Advertising TX/RX)

SoC: nRF52840

Advertising interval

~ 3000 ms

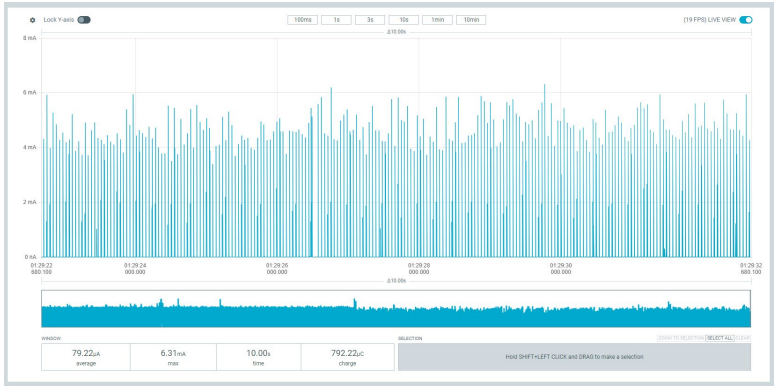
Average current = 8.37 µA

(Advertising TX/RX)

SoC: nRF52840

Step 3 – Request low power connection parameters

Connection param. (**default**)



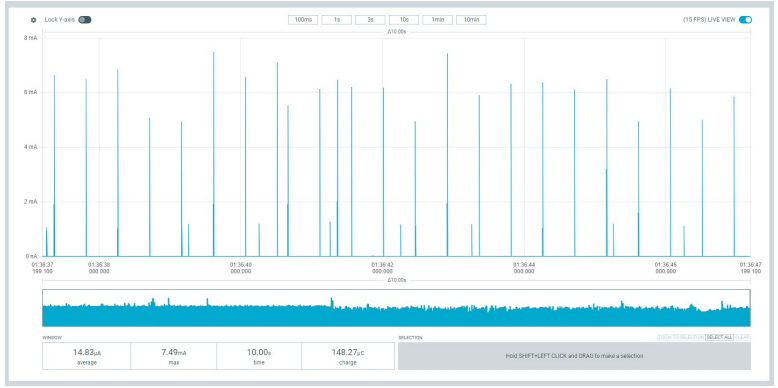
Interval: 45 ms, latency: 0, timeout: 420 ms

Average current = 79.22 µA

(Connection - Peripheral)

SoC: nRF52840

Connection param.



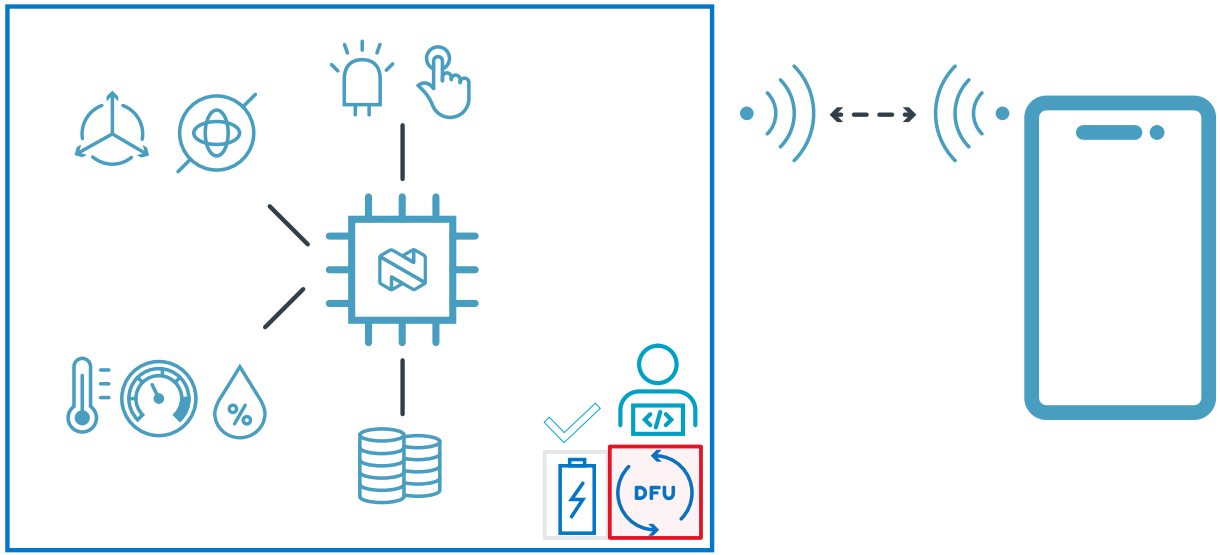
Interval: 150 ms, latency: 2, timeout: 2000 ms

Average current = 14.83 µA

(Connection - Peripheral)

SoC: nRF52840

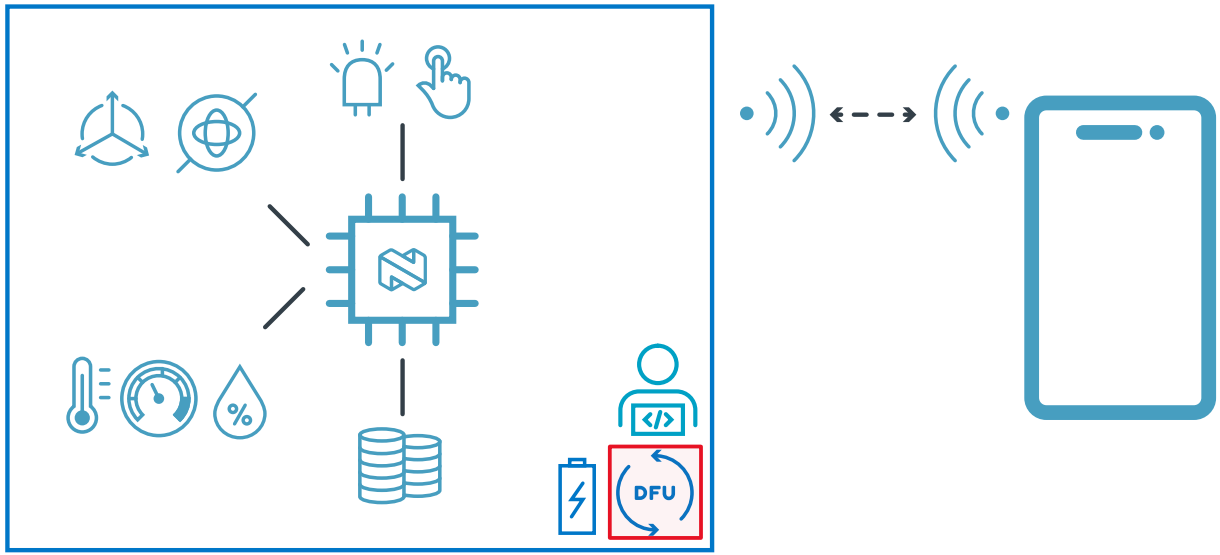
A typical Bluetooth LE application



nRF Connect SDK Hands-on

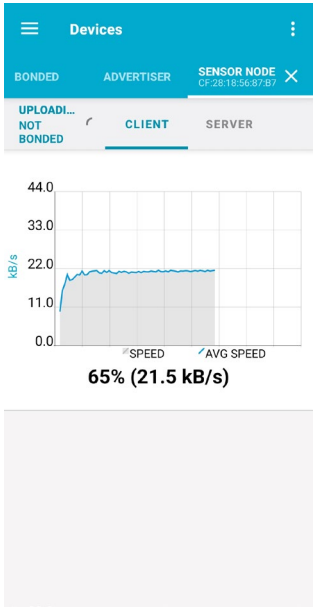
Video 6 - Adding FOTA over Bluetooth LE

A typical Bluetooth LE application

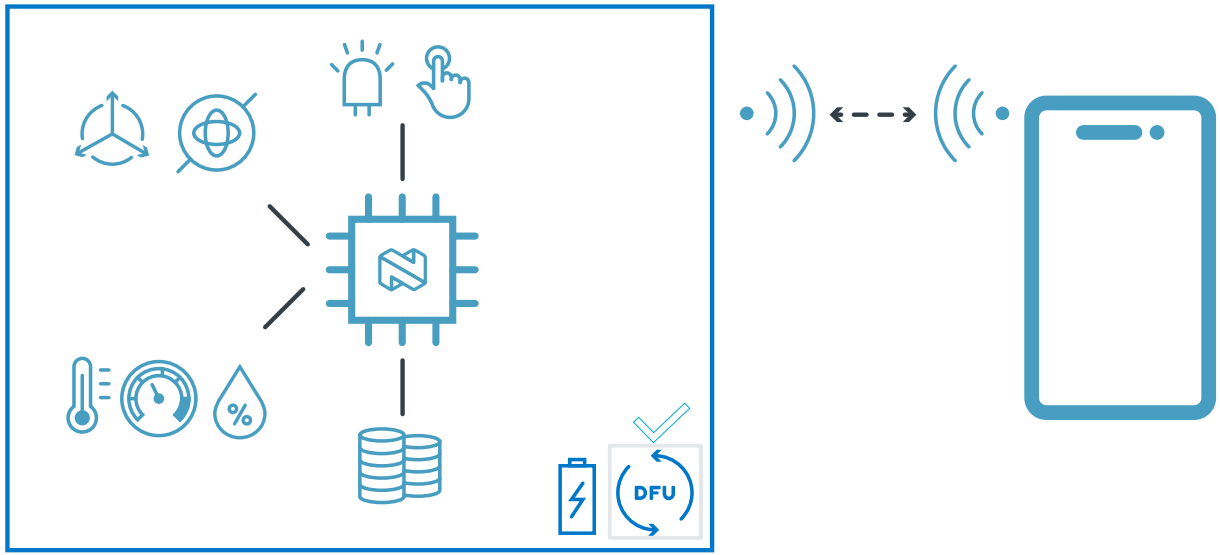


Adding FOTA over Bluetooth LE

- Get a bootloader (MCUboot)
  - `SB_CONFIG_BOOTLOADER_MCUBOOT=y`
- Enable FOTE over Bluetooth LE
  - `CONFIG_NCS_SAMPLE_MCUMGR_BT_OTA_DFU=y`
- Use Sysbuild and Partition Manager
- Learn more:
  - Adding Device Firmware Update (DFU/FOTA) Support in nRF Connect SDK
  - Lesson 8 – Bootloaders and DFU/FOTA



A typical Bluetooth LE application



The source code for the developed application is available on the Nordic DevAcademy GitHub.