# Rapport – Laboratoire 4

## Communication avec un périphérique BLE

**Cours** : Développement mobile avancé
**Date** : 10 mai 2024

**Enseignant** : Fabien Dutoit
**Assistant** : Elliot Ganty

**Étudiants :** Holzer Julien
Nascimento Santos Stéphane

# TABLE DES MATIÈRES :



## Introduction

Ce rapport répond aux questions théoriques du laboratoire consacré à la communication avec un périphérique Bluetooth Low Energy (BLE). Après avoir implémenté le scan, la connexion et les échanges de données avec un écran Espruino Pixl.js, nous analysons ici les choix techniques de représentation et de transmission de mesures (température, niveau de batterie) dans un contexte BLE.

## Question 5.1

*La caractéristique permettant de lire la température retourne la valeur en degrés Celsius, multipliée par 10, sous la forme d'un entier non-signé de 16 bits. Quel est l'intérêt de procéder de la sorte ? Pourquoi ne pas échanger un nombre à virgule flottante de type float par exemple ?*

Dans le contexte des communications Bluetooth Low Energy (BLE), les concepteurs de systèmes embarqués optent souvent pour une représentation de la température sous forme d'entier non-signé de 16 bits (uint16), multiplié par 10, plutôt que d'utiliser un nombre à virgule flottante (float). Cette décision repose sur plusieurs considérations techniques.

### Réduction de la taille des données transmises

Un entier de 16 bits occupe 2 octets en mémoire, tandis qu'un float standard en nécessite 4. Cette réduction de moitié de la taille des données est cruciale dans les communications BLE, où la capacité maximale d'un paquet est souvent limitée à 20 octets. En minimisant la taille des données, on améliore l'efficacité de la transmission et on réduit la consommation énergétique.

### Optimisation des performances sur microcontrôleur

Les microcontrôleurs utilisés dans les dispositifs BLE sont généralement conçus pour traiter efficacement les entiers. Beaucoup ne disposent pas d'une unité de calcul en virgule flottante, ce qui rend les opérations sur les flottants plus lentes et plus énergivores. En utilisant des entiers, on simplifie les calculs, on accélère le traitement des données et on prolonge la durée de vie de la batterie.

### Précision suffisante pour les applications courantes

Représenter la température en dixièmes de degré Celsius (par exemple, 23,5 °C codé comme 235) offre une précision adéquate pour la plupart des applications. Cette méthode permet de conserver une bonne précision tout en simplifiant le format des données.

L'utilisation d'un entier non-signé de 16 bits multiplié par 10 pour représenter la température permet de réduire la taille des données transmises, d'optimiser les performances des microcontrôleurs et de maintenir une précision suffisante pour les applications courantes. Ce qui s'avère donc judicieux dans le cadre des communications BLE.

## Question 5.2

*Le niveau de charge de la pile est à présent indiqué uniquement sur l'écran du périphérique, mais nous souhaiterions que celui-ci puisse informer le smartphone sur son niveau de charge restante. Veuillez spécifier la(les) caractéristique(s) qui composerai(en)t un tel service, mis à disposition par le périphérique et permettant de communiquer le niveau de batterie restant via Bluetooth Low Energy. Pour chaque caractéristique, vous indiquerez les opérations supportées (lecture, écriture, notification, indication, etc.) ainsi que les données échangées et leur format.*

Pour permettre à un périphérique Bluetooth Low Energy (BLE) de transmettre son niveau de charge au smartphone, il convient d'implémenter un service standardisé défini par le Bluetooth Special Interest Group (SIG). Le service le plus adapté à cette tâche est le Battery Service (UUID : 0x180F), qui expose une caractéristique centrale : la Battery Level Characteristic (UUID : 0x2A19).

### Battery Service

Ce service est conçu pour offrir une méthode standard de lecture de l'état de la batterie d'un appareil. Il est largement supporté par les plateformes mobiles et systèmes d'exploitation modernes, facilitant l'intégration côté client (smartphone ou tablette). Pour exposer le niveau de charge restante d'un périphérique BLE au smartphone, on implémente le **Battery Service**, défini par le Bluetooth SIG.

Ce service regroupe au minimum la caractéristique obligatoire suivante :

#### *Battery Level Characteristic*

- **Format des données** :
  - Type : entier non signé 8 bits (uint8)
  - Valeur : 0…100 → pourcentage de batterie restante
- **Opérations supportées** :
  - **Read** (lecture) : le client lit à la demande la valeur actuelle.
  - **Notify** (notification) :
    - le serveur envoie une notification dès qu'il détecte ≥ 1 % de variation depuis la dernière valeur transmise,
    - et/ou à la reconnexion si la valeur a changé pendant la déconnexion.

Il est important de noter que la spécification BAS 1.1 ne définit qu'une seule caractéristique obligatoire : la **Battery Level Characteristic**. Toutefois, des caractéristiques supplémentaires telles que **Battery Power State (UUID : 0x2A1A)** et **Battery Level State (UUID : 0x2A1B)** peuvent être implémentées pour fournir des informations plus détaillées sur l'état de la batterie, bien qu'elles ne soient pas obligatoires.

### *Battery Power State :*

- **Format des données** : struct >
  - Flags (uint8) → bits indiquant présence de champs supplémentaires
  - Power State (uint8) → présence batterie / en charge / décharge / source externe filaire / source externe sans fil
- **Opérations supportées** :
  - **Read**
  - **Notify** ou **Indicate** dès que l'un des champs change (ex. passage de « charging » à « discharging »).

### *Battery Level State :*

- **Format des données** : entier non signé 8 bits (uint8)
  - Valeurs possibles (exemple) :
    - 0 → Good
    - 1 → Low
    - 2 → Critical
- **Opérations supportées** :
  - **Read**
  - **Notify** dès que le niveau passe d'une zone à une autre (ex. Good → Low).

### *Descriptor (éventuellement)*

- **Champs principaux** :
  - **Name Space** : Bluetooth SIG
  - **Description** : « main » (pour la batterie principale)
- **Opérations supportées** : **Read** uniquement, pour expliquer l'unité et la présentation.

### Scénario multi-batteries

Si le périphérique embarquait plusieurs batteries (p. ex. principale et de secours), on pourrait instancier plusieurs fois le **Battery Service**, chaque instance exposant sa propre **Battery Level Characteristic** et son **Characteristic Presentation Format Descriptor**, afin de suivre indépendamment chaque source d'énergie.

## Conclusion

La représentation de la température par un entier 16 bits multiplié par 10 et l'utilisation du Battery Service standard BLE répondent aux contraintes de compacité, de performance et d'autonomie des systèmes embarqués. Ces formats assurent une transmission efficace et permettent une interopérabilité maximale avec les plateformes mobiles.

Référence :

Bluetooth SIG. (2012). *Battery Service Specification (BAS 1.1)*.
https://www.bluetooth.com/specifications/specs/bas-1-1/