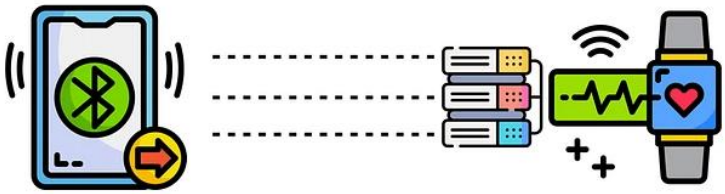
Bluetooth ATT & GATT Explained

Connection-Oriented Communication



VERSIONS BLUETOOTH LOW ENERGY (BLE)

4.0 Bluetooth	Premier protocole à mettre en œuvre la version Low Energy
4.1 Bluetooth	Vitesse supérieure à celle des versions précédentes Fixes the interference problem with 4G/LTE. Rétrocompatibilité avec les versions précédentes Un esclave peut être connecté à plusieurs maîtres simultanément.
4.2 Bluetooth	Possibilité pour les puces d'utiliser Bluetooth via Internet Protocol Version 6 (IPV6), pour un accès direct à Internet Par rapport à ses versions précédentes, Bluetooth 4.2 permet une transmission de données beaucoup plus rapide (250% plus rapide que les versions 4.0 et 4.1) La capacité des paquets envoyés a été multipliée par 10 par rapport aux versions précédentes. Rétrocompatible avec les versions précédentes
Bluetooth 5	Doublement de la vitesse de transmission (de 1Mbps à 2Mbps), et augmentation de la distance de lecture jusqu'à 4 fois avec la même puissance (on ne peut pas avoir les deux caractéristiques en même temps, juste une à la fois) Augmente la quantité de données pouvant être envoyées à un appareil BLE (jusqu'à 255 bytes) L'utilisation de la fonction qui permet une plus grande distance de lecture diminue considérablement la vitesse de transmission, car plus la distance entre les deux appareils est grande, plus la probabilité que certains bits soient perdus est grande, et faire les vérifications nécessaires pour éviter cela réduit les bits du message réel qui sont envoyés Rétrocompatible avec les versions précédentes

Year Introduced	Bluetooth Version	Feature
2004	2.0	Enhanced Data Rate
2007	2.1	Secure Simple Pairing
2009	3.0	High Speed with 802.11 Wi-Fi Radio
2010	4.0	Low-energy protocol
2013	4.1	Indirect IoT device connection
2014	4.2	IPv6 protocol for direct internet connection
2016	5.0	4x range, 2x speed, 8x message capacity + IoT

Atout majeur du Bluetooth Low Energy : **faible consommation d'énergie.**
De manière générale, le **BLE consomme moitié moins que le Bluetooth**.
Son coût reste relativement faible et la longévité de sa batterie non négligeable.
Utilisé pour le transfert périodique de petites quantités de données à courte portée.
La portée réelle dépend de la puissance du module radio. Mais comme une forte puissance nécessite davantage de batteries, le BLE a plutôt une portée réduite. Même s'il est parfaitement possible d'avoir une portée de 30 mètres, on va le plus souvent de 2 à 5 mètres.

Bluetooth Classic vs. BLE

Ces deux technologies sont utilisées à des fins très différentes. Le Bluetooth classique est employé pour traiter, transférer et échanger de nombreuses données sans interruption (par exemple en audio). Cependant, il consomme rapidement la vie de la batterie et coûte beaucoup plus cher.

Le BLE, quant à lui, est utilisé pour des applications ne nécessitant pas l'échange de grandes quantités de données et récupère par conséquent des informations relativement légères (comme l'heure ou la température par exemple). Proposant ainsi une connexion non continue, il peut fonctionner sur batterie pendant plusieurs années à un coût inférieur à celui du Bluetooth.

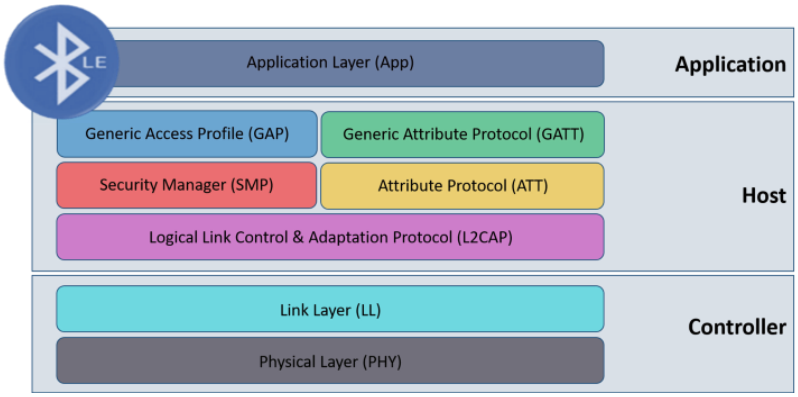
Le BLE fonctionne sur la bande Industrielle Scientifique et Médicale des 2,4 GHz.

Bluetooth Classic	BLE
Used for streaming applications such as audio streaming, file transfers, and headsets	Used for sensor data, control of devices, and low-bandwidth applications
Not optimized for low power, but has a higher data rate (3Mbps maximum compared to 2Mbps for BLE)	Meant for low power, low duty data cycles
Operates over 79 RF (radio frequency) channels	Operates over 40 RF channels.
Discovery occurs on 32 channels	Discovery occurs on 3 channels, leading to quicker discovery and connections than Bluetooth Classic

Table 1: Bluetooth Classic vs. BLE

Protocole

Le protocole BLE repose sur un certain nombre de couches :



Les couches GAP, GATT et ATT, sont essentielles dans l'échange de données.

- GAP (*Generic Access Profile*) : elle est responsable de l'établissement du lien et du contrôle de connexion entre deux appareils. Le GAP est ce qui rend un appareil visible au monde extérieur et détermine la manière dont deux appareils peuvent (ou ne peuvent pas) interagir.
- GATT (*Generic Attribute Profile*) : est une méthode de présentation de données utilisée par les périphériques BLE. Il s'agit d'un ensemble de règles décrivant comment structurer, présenter et transférer des données.
- ATT (*Attribute Protocol*) : repose sur une relation client/serveur. Le serveur dispose des informations (telles que les valeurs des capteurs) et envoie des **réponses**, des **notifications** et des **indications** (par exemple un capteur de température). Le client est celui qui veut accéder à ces informations (par exemple un ordinateur ou un smartphone). ATT permet à un serveur d'exposer à un client un ensemble de données.

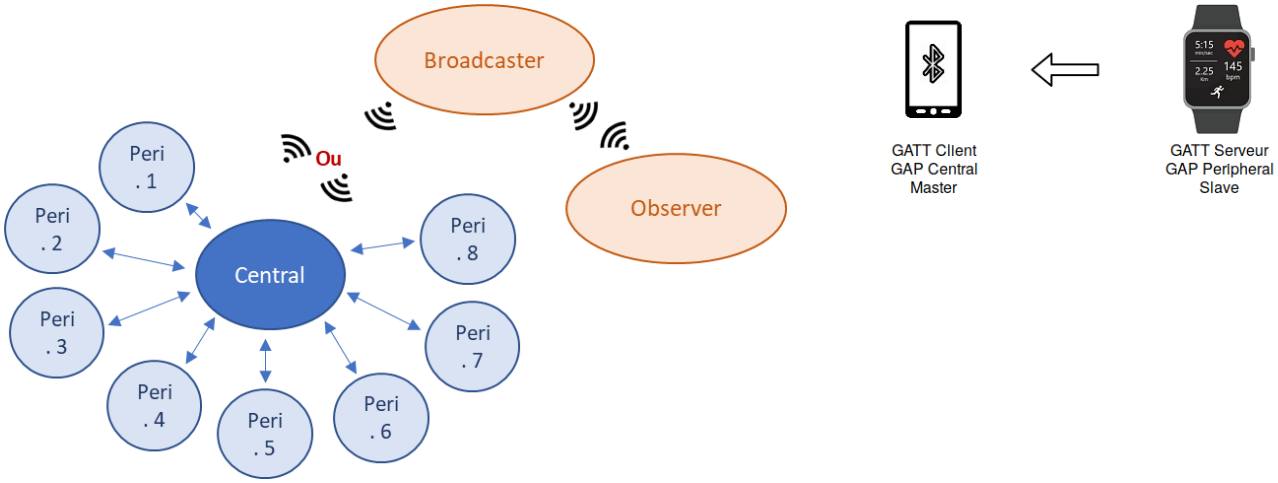
GAP

La couche GAP (Generic Access Profile) définit 4 rôles :

- **broadcaster**, **scanner** (ou observer), **peripheral** et **central**.

Il y a deux types de communication :

- le mode dit **connecté** : Peripheral <-> Central
- le mode **advertising** : Broadcaster -> Scanner (ou observer).



Le **Broadcaster** peut faire office de serveur. Ainsi, il a pour objectif de transmettre régulièrement des données à un appareil, mais il n'accepte aucune connexion entrante.

L'**Observer** peut seulement écouter et interpréter les données envoyées par un broadcaster. Dans cette situation-là, l'objet ne peut pas envoyer de connexions vers le serveur.

Le **Central** est souvent un smartphone ou une tablette. C'est un élément qui interagit de deux façons différentes : soit en mode advertising, soit en mode connecté. Il est alors le maître (*master*) et c'est de lui que part l'échange de données.

Le **Peripheral** est un esclave (*slave*). Il accepte les connexions du central et lui envoie des données de manière périodique. Ce système a pour objectif de packager les données de façon universelle via le protocole afin qu'elles soient comprises par les autres périphériques.

GATT et ATT

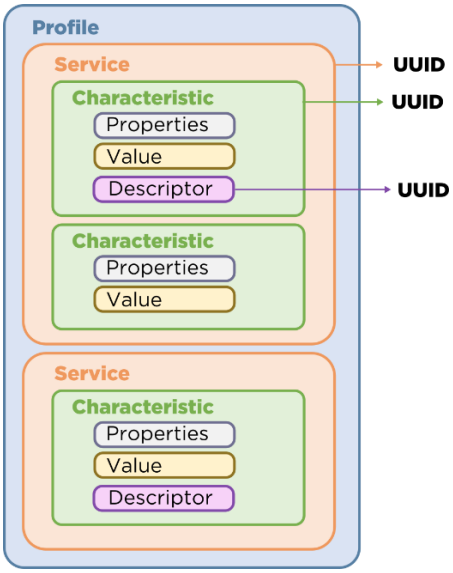
ATT est la structure qui définit les éléments de base appelés **attributs**, tels que les **services** et les **caractéristiques**, et les **descripteurs** utilisés dans un profil.

Un **attribut** est le terme générique pour tout type de données exposées par le serveur et définit la structure de ces données.

Le concept du GATT est de regrouper les attributs dans une **table d'attributs** dans un ordre spécifique et logique et en utilisant des identifiants numériques appelés **UUID** (Universally Unique IDdentifier). Il s'agit d'un ensemble de règles décrivant comment structurer, présenter et transférer des données en utilisant le protocole ATT.

Le groupement Bluetooth Special Interest Group (SIG)

Les transactions GATT dans BLE sont basées sur des objets imbriqués de haut niveau appelés profiles, services et caractéristiques, comme le montre l'illustration ci-dessous :



La philosophie du BLE repose beaucoup sur des configurations prédéfinies et des normes pour des applications spécifiques. Cela évite de redéfinir à chaque fois les choses. En tant que développeur, s'agit alors de regarder dans les tables du Bluetooth SIG si notre situation existe déjà ou non.

Attributs

Le protocole ATT définit une structure de données appelée **attribut** comme méthode standard pour organiser efficacement les données stockées, consultées et mises à jour sur le serveur.

Les attributs sont constitués des éléments suivants :

- un descripteur d'attribut (Attribute Handle)
- un type d'attribut défini par un UUID (Type of attribute (UUID))
- un ensemble d'autorisations (Attribute Permissions)
- une valeur (Attribute value)

Attribute Handle (Descripteur d'attribut)

Il s'agit d'une valeur de 16 bits que le serveur attribue à chacun de ses attributs – un peu comme une adresse. Cette valeur est utilisée par le client pour faire référence à un attribut spécifique. La plage des descripteurs est de 0x0001-0xFFFF.

UUID

Les attributs sont identifiés par des UUID (Universally Unique IDdentifier) est un nombre codé sur 128 bits.
On peut choisir un UUID aléatoire ou pseudo-aléatoire (ave l'outil *uuidgen*) pour des utilisations propriétaires, mais une plage d'UUID de la forme suivante est réservée pour les attributs standards :

`xxxxxxxx-0000-1000-8000-00805F9B34FB`

Le format l'usage et la génération sont spécifiés par la norme ISO/IEC 9834-8:2005.

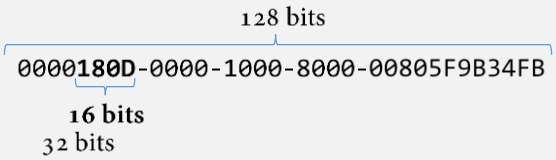
Pour plus de commodité, ces identificateurs sont représentés par des valeurs abrégé de 16 bits ou de 32 bits dans le protocole, plutôt que par les 128 bits requis pour un UUID complet.
Le Bluetooth SIG fournit une liste d'*UUID standards* avec des codes abrégé, donnés en hexadécimal.

UUID (16 bits)	Usage
0x00XX	Descripteurs de nom
0x18XX	Services
0x27XX	Unités
0x28XX	Déclarations
0x29XX	Descripteurs
0x2AXX	Caractéristiqurs

Liste officielle des familles d'UUID 16 bits reconnus

EXEMPLE :

La valeur de l'UUID pour une mesure de fréquence cardiaque a le code abrégé `0x180D` *en 16 bits.*



L'UUID est donc :

Attribute Permissions

Les permissions déterminent si un attribut peut être **lu** ou **écrit**, s'il peut être **notifié** ou **indiqué**, et quels niveaux de sécurité sont requis pour chacune de ces opérations. Ces permissions ne sont pas définies ou découvertes via le protocole d'attribut (ATT), mais plutôt définies à une couche supérieure (couche GATT ou couche Application).

Access Permissions

Similar to file permissions, access permissions determine whether the client can read or write (or both) an attribute value (introduced in "Value" on page 55). Each attribute can have one of the following access permissions:

None	The attribute can neither be read nor written by a client.
Readable	The attribute can be read by a client.
Writable	The attribute can be written by a client.
Readable and writable	The attribute can be both read and written by the client.

Value

La valeur de l'attribut contient le contenu réel des données de l'attribut. Il n'y a aucune restriction quant au type de données qu'elle peut contenir, bien que sa longueur maximale soit limitée à 512 octets par la spécification.

Profiles

Un profil est simplement une collection prédéfinie de services qui a été compilée soit par le Bluetooth SIG, soit par les concepteurs de périphériques. La liste complète des profils officiellement adoptés et basés sur le GATT est disponible sue portail des développeurs Bluetooth : *https://www.bluetooth.com/specifications/gatt*

EXEMPLES :

Le profil de fréquence cardiaque, combine le service de fréquence cardiaque et le service d'information sur les dispositifs (niveau de batterie).

Le profil de XXXXX, combine le service de fréquence cardiaque et le service d'information sur les dispositifs (niveau de batterie).

Services

Les services sont utilisés pour diviser les données en entités logiques et contiennent des blocs de données spécifiques appelés caractéristiques.

Un service peut avoir une ou plusieurs caractéristiques, et chaque service se distingue des autres services par un unique identifiant UUID.

Une liste complète des services BLE officiellement adoptés est disponible sur la page *Services* du portail des développeurs Bluetooth.

Un service doit toujours être déclaré par un attribut de déclaration de service :

Handle	Type	Permissions	Value	Value length
`0xXXXX`	$UUID_{primary\ service}$ = `0x2800` or $UUID_{secondary\ service}$ = `0x2801`	Read only	Service UUID	2,4, or 16 bytes

EXEMPLE :

Le groupement Bluetooth SIG a défini un service appelé service de fréquence cardiaque (Heart Rate Service). Les montres de ce type utilisent généralement au moins deux services :

- *Un service de fréquence cardiaque qui encapsule trois caractéristiques :*
  - *Caractéristique obligatoire de mesure de la fréquence cardiaque contenant la valeur de la fréquence cardiaque.*
  - *Une caractéristique facultative d'emplacement du capteur corporel.*
  - *Une caractéristique conditionnelle du point de contrôle de la fréquence cardiaque.*
- *Un service de batterie :*
  - *Caractéristique obligatoire du niveau de la batterie.*

Si l'on examine le service Heart Rate Service, par exemple, on constate que ce service possède officiellement un UUID de 16 bits, 0x180D, et contient jusqu'à 3 caractéristiques, bien que seule la première soit obligatoire : Heart Rate Measurement, Body Sensor Location and Heart Rate Control Point.

Allocation Type	Allocated UUID	Allocated for
GATT Service	0x1808	Glucose
GATT Service	0x1809	Health Thermometer
GATT Service	0x180A	Device Information
GATT Service	0x180D	Heart Rate
GATT Service	0x180E	Phone Alert Status
GATT Service	0x180F	Battery
GATT Service	0x1810	Blood Pressure

Le service devra donc être déclaré par l'attribut de déclaration de service de la manière suivante :

Handle	Type	Permissions	Value	Value length
`0x000E`	`0x2800`	Read only	`0x180D`	2 bytes

Caractéristiques

On peut considérer les caractéristiques comme des conteneurs pour les données utilisateur. Elles comprennent toujours au moins deux attributs :

- la **déclaration de la caractéristique** (qui fournit des métadonnées sur les données utilisateur réelles)
- la **valeur** de la caractéristique (qui est un attribut complet contenant les données utilisateur dans son champ de valeur)

Le tableau ci-dessous montre la déclaration d'une caractéristique par un attribut de déclaration de caractéristique (1ere ligne). La valeur de cet attribut contient 3 champs de données : Propriété, Value handle et Characteristic UUID. Suivit de la charcatéristique elle-même (2eme ligne) :

Handle	Type	Permissions	Value	Value length
`0xXXXX`	$\text{UUID}_{\text{characteristic declaration}}$ = `0x2803`	Read only	Properties, Value handle (`0xYYYY`), Characteristic UUID	2,4, or 16 bytes
`0xYYYY`	Characteristic UUID	Any	Actual value	variable

La propriété d'une caractéristique est représentée par un certain nombre de bits et qui définissent comment la valeur d'une caractéristique peut être utilisée. Quelques exemples : lire, écrire, écrire sans réponse, notifier, indiquer.

Properties	Value (Bit field)	Description (From BLE Core Specification v4.2)
Broadcast	0x01	If set, permits broadcasts of the Characteristic Value using Server Characteristic Configuration Descriptor. If set, the Server Characteristic Configuration Descriptor shall exist.
Read	0x02	If set, permits reads of the Characteristic Value
Write without response	0x04	If set, permit writes of the Characteristic Value without response
Write	0x08	If set, permits writes of the Characteristic Value with response
Notify	0x10	If set, permits notifications of a Characteristic Value without acknowledgement. If set, the Client Characteristic Configuration Descriptor shall exist.
Indicate	0x20	If set, permits indications of a Characteristic Value with acknowledgement. If set, the Client Characteristic Configuration Descriptor shall exist.
Authenticated Signed Writes	0x40	If set, permits signed writes to the Characteristic Value.
Extended Properties	0x80	If set, additional characteristic properties are defined in the Characteristic Extended Properties Descriptor. If set, the Characteristic Extended Properties Descriptor shall exist.

EXEMPLE :

Un thermomètre qui a le rôle de serveur peut laisser le client demander les données en continu avec la propriété READ. Ou bien il peut avertir le client uniquement lorsque la valeur de température a changé. Dans ce cas-là on utilise NOTIFY.

Comme pour les services, chaque caractéristique possède un UUID. On est libre d'utiliser les caractéristiques standard définies par le Bluetooth SIG (qui garantit l'interopérabilité entre les matériels et les logiciels compatibles BLE) ou de définir nos propres caractéristiques personnalisées que seuls nos périphériques et logiciels peuvent comprendre.

La caractéristique peut également contenir d'autres attributs qui contribuent à définir la valeur qu'elle détient :

- Les **descripteurs** : facultatifs, ils fournissent des informations supplémentaires sur une caractéristique. Par exemple : propriétés étendues, description de l'utilisateur, champs utilisés pour s'abonner aux notifications et aux indications, et un champ qui définit la présentation de la valeur, comme le format et l'unité de la valeur.

EXEMPLE :

Une caractéristique de valeur de température peut avoir une indication de ses unités (par exemple Celsius) et des valeurs maximales et minimales que le capteur peut mesurer.

Allocation Type	Allocated UUID	Allocated for
GATT Unit	0x272A	electric resistance (ohm)
GATT Unit	0x272B	electric conductance (siemens)
GATT Unit	0x272C	magnetic flux (weber)
GATT Unit	0x272D	magnetic flux density (tesla)
GATT Unit	0x272E	inductance (henry)
GATT Unit	0x272F	Celsius temperature (degree Celsius)
GATT Unit	0x2730	luminous flux (lumen)

Handle	Type	Permissions	Value	Value length
0x000F	$UUID_{descriptor}$=0x2902	Read and Write	0x272F	2 bytes

EXEMPLE :

La caractéristique Heart Rate Measurement est obligatoire pour le service Heart Rate Service, et utilise l'UUID : `0x2A37`*. Il commence par une valeur unique de 8 bits décrivant le format des données HRM (si les données sont UINT8 ou UINT16, etc.), puis inclut les données de mesure de la fréquence cardiaque qui correspondent à cet octet de configuration.*

Allocation Type	Allocated UUID	Allocated for
GATT Characteristic and Object Type	0x2A34	Glucose Measurement Context
GATT Characteristic and Object Type	0x2A35	Blood Pressure Measurement
GATT Characteristic and Object Type	0x2A36	Intermediate Cuff Pressure
GATT Characteristic and Object Type	0x2A37	Heart Rate Measurement
GATT Characteristic and Object Type	0x2A38	Body Sensor Location
GATT Characteristic and Object Type	0x2A39	Heart Rate Control Point
GATT Characteristic and Object Type	0x2A3F	Alert Status

La caractéristique devra donc être déclarée par l'attribut de déclaration de caractéristique de la manière suivante avec la première ligne. La propriété de la caractéristique est NOTIFY (0x10), le handle de la valeur (0x0010) et l'UUID de la caractéristique définit par Bluetooth SIG (0x2A37) La deuxième ligne est l'attribut qui contient la valeur de la mesure da la fréquence cardiaque (ici 167 battement par minutes) :

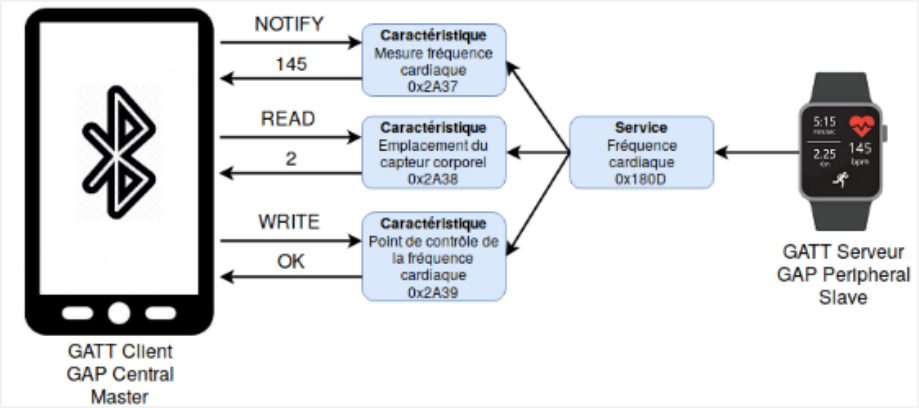
Handle	Type	Permissions	Value	Value length
`0x000F`	`0x2803`	Read only	`0x10`, `0x0010`, `0x2A37`	2 bytes
`0x0010`	`0x2A37`	Read only	167 (bmp)	1 byte

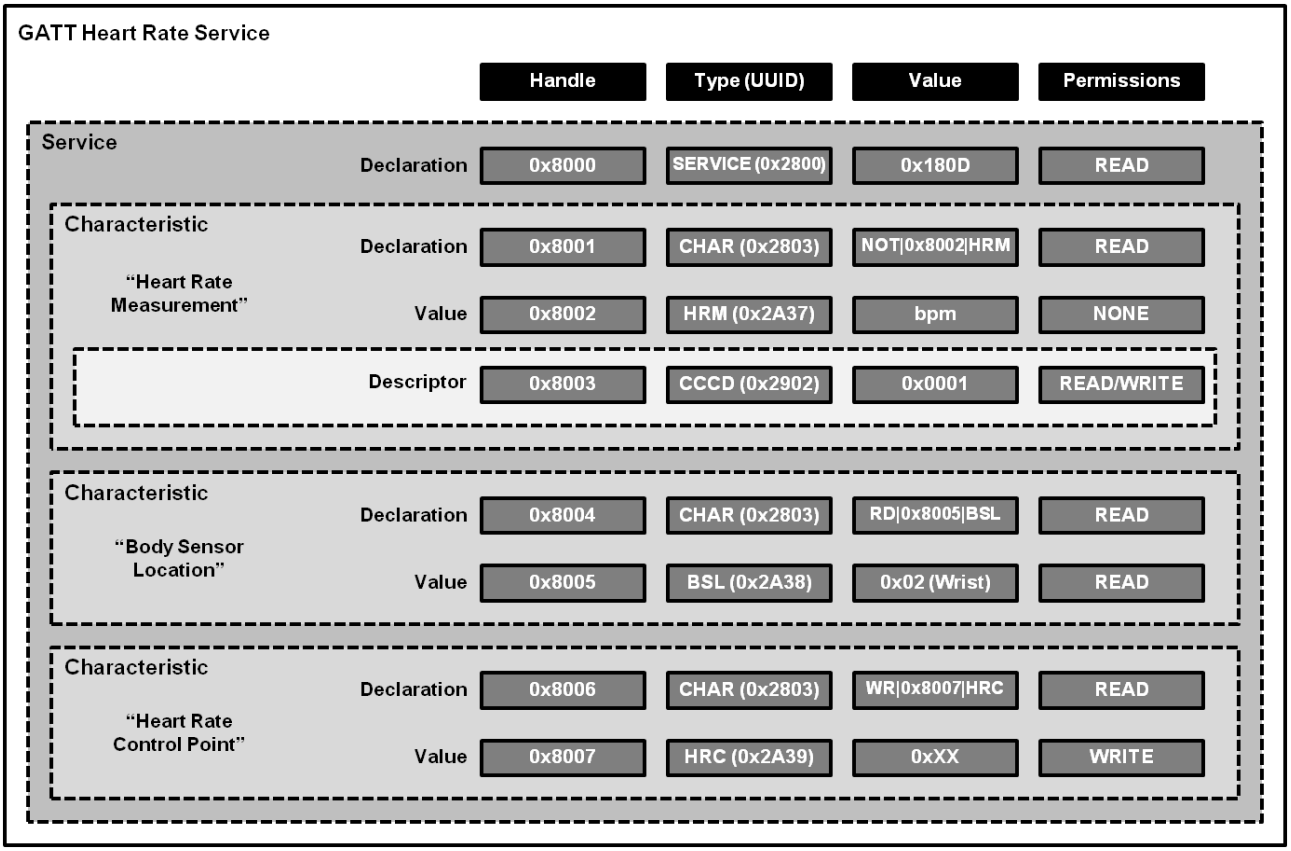
EXEMPLE :

Le profil de fréquence cardiaque présenté dans le tableau ci-dessous est un exemple : chaque ligne est un attribut et chaque attribut a un descripteur, un type, un ensemble d'autorisations et une valeur.

Au sommet de chaque groupe, il y a toujours un attribut de déclaration de service. L'UUID standard pour les déclarations de service est 0x2800. La valeur est un autre UUID définissant le type de service concerné : ici la valeur 0x180D qui est l'UUID défini par le groupement Bluetooth SIG pour le service de fréquence cardiaque

Heart Rate Profile	Handle	Type of attribute (UUID)	Attribute permission	Attribute value
Service Declaration	0x000E	**Service declaration** Standard UUIDservice 0x2800	Read Only, No Authentication, No Authorization	Heart Rate Service 0x180D
Characteristic Declaration	0x000F	**Characteristic declaration** Standard UUIDcharacteristic 0x2803	Read Only, No Authentication, No Authorization	Properties (Notify) Value Handle (0x0010) UUID for Heart Rate Measurement characteristic (0x2A37)
Characteristic Value Declaration	0x0010	**Heart Rate Measurement Characteristic** UUID found in the Characteristic declaration value 0x2A37	Higher layer profile or implementation specific.	Beats Per Minute E.g "167"
Descriptor Declaration	0x0011	**Client Characteristic Configuration Descriptor (CCCD)** Standard UUIDservice 0x2800	Readable with no authentication or authorization. Writable with authentication and authorization defined by a higher layer specification or is implementation specific.	Notification enabled 0x000X
Characteristic Declaration	0x0012	**Characteristic declaration** Standard UUIDcharacteristic 0x2803	Read Only, No Authentication, No Authorization.	Properties (READ), Value Handle (0x0011), UUID for Body Sensor Location (0x2A38)
Characteristic Value Declaration	0x0013	**Body Sensor Location** UUID found in the Characteristic declaration value 0x2A38	Higher layer profile or implementation specific	Sensor Location (8-bit integer) E.g. 3 equals "Finger"

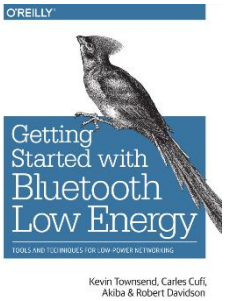




PROG ARDUINO

```
#include <ArduinoBLE.h>

BLEPeripheral blePeripheral;       // Bluetooth® Low Energy Peripheral Device
(the board you're programming)

BLEService heartRateService("180D"); // Bluetooth® Low Energy Heart Rate Service

// BLE Heart Rate Measurement Characteristic"

BLECharacteristic heartRateChar("2A37",  // standard 16-bit characteristic UUID

    BLERead | BLENotify, 2);  // remote clients will be able to get
notifications if this characteristic changes

                              // the characteristic is 2 bytes long as the first
field needs to be "Flags" as per Bluetooth® Low Energy specifications

                              //
https://developer.bluetooth.org/gatt/characteristics/Pages/CharacteristicViewer.
aspx?u=org.bluetooth.characteristic.heart_rate_measurement.xml
```

References



Kevin Townsend, Carles Cuf, Akiba, and Robert Davidson. 2014. Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking (1st. ed.). O'Reilly Media, Inc.

Page web de Thierry Vaira :
http://tvaira.free.fr/bts-sn/activites/activite-ble/bluetooth-ble.html