


BLE : Bluetooth Low Energy	
	
Sommaire :	
I - Présentation.....	1
II - Le GATT.....	2
III - BLE et Linux.....	2
IV - Theengs Gateway.....	5

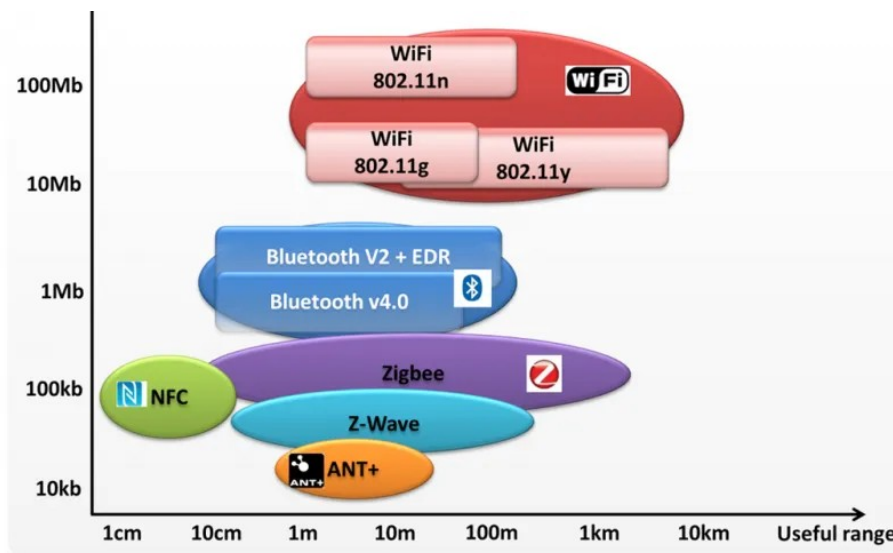
I - Présentation

Le **Bluetooth** est un protocole de communication sans fil, défini en **1999** (version **1.0**) par le consortium **Special Interest Group (SIG)**. En **2010**, la version **4.0** du standard dévoile la spécification **Bluetooth Low Energy (BLE)** qui évoluera jusqu'à la version **5.0** sortie en décembre **2016**.

Le **Bluetooth** « classique » est employé pour traiter, transférer et échanger de nombreuses données sans interruption (par exemple en audio), alors que le **Bluetooth LE**, est utilisé pour le transfert périodique de petites quantités de données (Température, heure, identifiant, etc.).

Le standard est basé sur la spécification **IEEE 802.15.1** qui définit les couches physiques et liaisons de modèle **OSI** pour des communications **WPAN (Wireless Personal Area Network) Bluetooth**.

Sur la figure suivante, on peut comparer les différents technologies **sans fil** en terme de **débit** et de **distance**, l'échelle des **débites** est homogène à celle de l'**énergie** consommée :



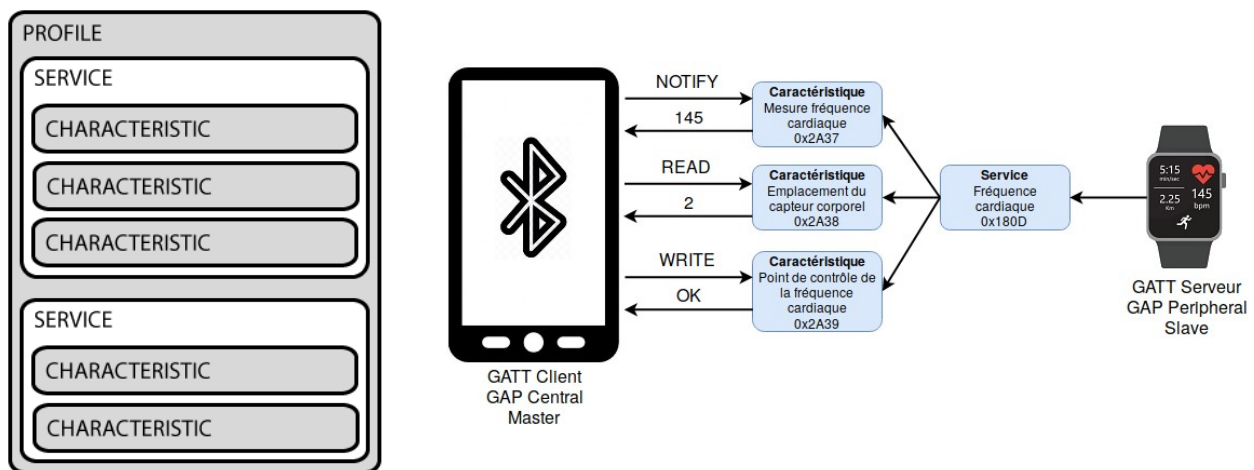
Généralement, avec le **Bluetooth Low Energy**, le maître est un ordinateur ou un smartphone nommé **central** et l'esclave un objet connecté nommé **périphérique**.

Le **Bluetooth** spécifie des **profils** qui effectuent des opérations haut niveau entre le **maître** et l'**esclave**. Le standard impose deux profils par défaut pour le **Bluetooth Low Energy (BLE)**, le **Generic Access Profile (GAP)** et le **Generic ATtribute Profile (GATT)**.

II - Le GATT

Le **GATT** (**G**eneric **A**TTtribute Profile) se base sur le format de données de l'Attribute Protocol (**ATT**) pour fournir une sorte de base de donnée. Le **GATT** utilise un système d'interaction serveur-client entre les deux appareils connectés. Le serveur est hébergé sur le périphérique qui met à disposition du central des informations (température, humidité, etc) rangées en **services** et **caractéristiques**.

Certains **services** et certaines **caractéristiques** sont prédéfinis par la spécification **Bluetooth** comme les services **Battery Service** ou **Heart Rate** (fréquence cardiaque).



Ainsi, un smartphone pourra se connecter automatiquement à un compteur de pulsation cardiaque juste en reconnaissant l'identifiant de ce service dans la trame d'**advertising**. Les caractéristiques peuvent, en fonction des permissions, être lues, écrites ou peuvent **notifier** ou indiquer au central un changement de valeur. La **notification** permet l'envoi direct de la nouvelle valeur au central alors que l'indication lui signale que celle-ci est prête à être lue. Ce système permet une économie d'énergie, car le central n'a plus besoin d'interroger régulièrement le serveur **GATT**.

III - BLE et Linux

BlueZ est la pile Bluetooth officielle de Linux sous licence GNU GPL. Elle s'appuie sur des composants logiciels en espace noyau et utilisateur. **BlueZ** est intégré au noyau Linux à partir de la version **2.4**.

Remarque : Le Raspberry Pi 3 dispose d'une puce Wifi et Bluetooth 4.x intégrée et prend donc en charge la technologie *Bluetooth Low Energy* (BLE). Voir https://elinux.org/Rpi_Bluetooth_LE.

Sur un système **Linux**, on peut lister le **périphérique local** (le **central**) à l'aide des commandes **hcitool dev** et **hciconfig -a** :

```
$ hcitool dev
```

```
Devices:
```

```
hci0 28:7F:CF:A7:57:FA
```

```
$ hciconfig -a
```

```
hci0: Type: Primary Bus: USB
```

```
BD Address: 28:7F:CF:A7:57:FA ACL MTU: 1021:4 SCO MTU: 96:6
```

```

UP RUNNING
RX bytes:10470 acl:2 sco:0 events:670 errors:0
TX bytes:53618 acl:2 sco:0 commands:483 errors:0
Features: 0xbf 0xfe 0x0f 0xfe 0xdb 0xff 0x7b 0x87
Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3
Link policy: RSWITCH SNIFF
Link mode: SLAVE ACCEPT
Name: 'ubuntu-NJ50-70CU'
Class: 0x1c010c
Service Classes: Rendering, Capturing, Object Transfer
Device Class: Computer, Laptop
HCI Version: 4.2 (0x8) Revision: 0x1100
LMP Version: 4.2 (0x8) Subversion: 0x1100
Manufacturer: Intel Corp. (2)
    
```

On peut lister les **périphériques distants** à l'aide de la commande **sudo hcitool lescan** :

```

$ sudo hcitool lescan
[sudo] Mot de passe de jcabianca :
LE Scan ...
65:A5:A5:72:A7:F4 (unknown)
65:A5:A5:72:A7:F4 (unknown)
04:EA:73:78:35:D3 (unknown)
A4:C1:38:8A:24:06 (unknown)
A4:C1:38:8A:24:06 ATC_8A2406
A4:C1:38:16:72:AC (unknown)
A4:C1:38:16:72:AC ATC_1672AC
    
```

Remarque : A l'aide de la page suivante https://github.com/atc1441/ATC_MiThermometer, le firmware du module **BLE XIAOMI Mijia LYWSD03MMC** a été mis à jour et maintenant il apparaît avec un nom du type **ATC_XXYYZZ** et non comme avant **LYWSD03MMC**.

On dispose alors des **adresses MAC** de nos **périphériques**. On peut alors se connecter en mode interactif (-I) à l'un des périphérique avec ces commandes :

```

$ sudo gatttool --device=A4:C1:38:16:72:AC -I
[A4:C1:38:16:72:AC][LE]>connect
    
```

Toutes les 10 secondes environ, un message de ce type est reçue par le central :

```

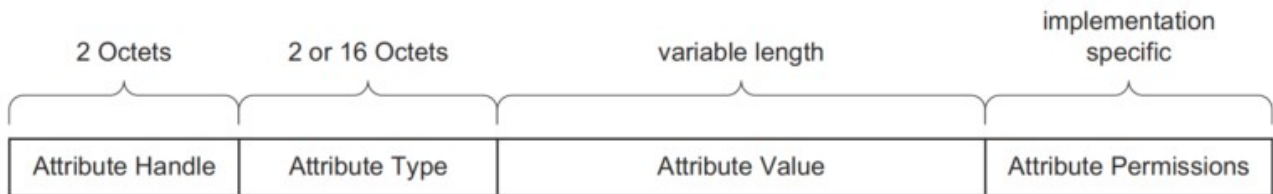
Notification handle = 0x0012 value: 0a 01
Notification handle = 0x0015 value: a8 16
Notification handle = 0x000e value: 50
    
```

Ce message peut être décodé de la façon suivante :

0a	01	a8	16	50
266		5800		80
Température * 10 (°C)		Hygrométrie * 100 (%)		Tension batterie (%)

- Handle 0x12 : Température : Hex 0A 01 => 0x010A = 266 = 26,6 °C.
- Si la température est négative, les 2 premiers octets affichent :
 - FF FF pour -0.1 °C / F6 FF pour -1.0 °C / 9C FF pour -10.0 °C
 - Soit la formule : Température = -65536 + Température
- Handle 0x15 : Hygrométrie : Hex A8 16 => 0x16A8 = 5800 = 58 %
- Handle 0x0e : Tension batterie en % : Hex 50 => 0x50 = 80 = 80 %

Le format de données de l'**Attribute Protocol (ATT)** est le suivant :



Attribute Handle : Identifiant de chaque attribut

Attribute Type : **UUID (Universally Unique Identifier)**

Attribute Value : Valeurs de chaque attribut

Attribute Permissions : Permissions de chaque attribut (read, write, notified ou indicated)

Les informations sont rangées en **services** et **caractéristiques**. La commande **primary** va explorer les **handles** des **services** primaires :

```
[A4:C1:38:16:72:AC][LE]> primary
attr handle: 0x0001, end grp handle: 0x0007 uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle: 0x0008, end grp handle: 0x000b uuid: 00001801-0000-1000-8000-00805f9b34fb
attr handle: 0x000c, end grp handle: 0x000f uuid: 0000180f-0000-1000-8000-00805f9b34fb
attr handle: 0x0010, end grp handle: 0x0016 uuid: 0000181a-0000-1000-8000-00805f9b34fb
attr handle: 0x0017, end grp handle: 0x001a uuid: 00010203-0405-0607-0809-0a0b0c0d1912
attr handle: 0x001b, end grp handle: 0x001e uuid: 00001f10-0000-1000-8000-00805f9b34fb
attr handle: 0x001f, end grp handle: 0x0020 uuid: 0000fe95-0000-1000-8000-00805f9b34fb
```

Pour ce périphérique, les **services primaires** sont au nombre de **7** et chaque service est identifié par un **handle**. Par exemple le **handle 0x0001** correspond à l'**uuid sur 16 bits** valant **0x1800**.

En comparant certains de ces **uuid** avec la nomenclature standardisée Bluetooth du fichier « **16-bit UUID Numbers Document.pdf** » disponible à l'adresse <https://btprodspecificationrefs.blob.core.windows.net/assigned-values/16-bit%20UUID%20Numbers%20Document.pdf> :

- handle 0x0001 : **1800** => Generic Access
- handle 0x0008 : **1801** => Alert Notification Service
- handle 0x000c : **180f** => Battery Service

La commande **char-desc** va explorer les **descripteurs de caractéristiques** et nous retourner tous les **handles** disponibles avec leurs numéros et leurs **uuid** :

```
[A4:C1:38:16:72:AC][LE]> char-desc
handle: 0x0001, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0002, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0003, uuid: 00002a00-0000-1000-8000-00805f9b34fb
handle: 0x0004, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0005, uuid: 00002a01-0000-1000-8000-00805f9b34fb
```

handle: 0x0006, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0007, uuid: 00002a04-0000-1000-8000-00805f9b34fb
handle: 0x0008, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0009, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000a, uuid: 00002a05-0000-1000-8000-00805f9b34fb
handle: 0x000b, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x000c, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x000d, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x000e, uuid: 00002a19-0000-1000-8000-00805f9b34fb
handle: 0x000f, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x0010, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0011, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0012, uuid: 00002a1f-0000-1000-8000-00805f9b34fb
handle: 0x0013, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x0014, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0015, uuid: 00002a6f-0000-1000-8000-00805f9b34fb
handle: 0x0016, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x0017, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0018, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x0019, uuid: 00010203-0405-0607-0809-0a0b0c0d2b12
handle: 0x001a, uuid: 00002901-0000-1000-8000-00805f9b34fb
handle: 0x001b, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x001c, uuid: 00002803-0000-1000-8000-00805f9b34fb
handle: 0x001d, uuid: 00001f1f-0000-1000-8000-00805f9b34fb
handle: 0x001e, uuid: 00002902-0000-1000-8000-00805f9b34fb
handle: 0x001f, uuid: 00002800-0000-1000-8000-00805f9b34fb
handle: 0x0020, uuid: 00002901-0000-1000-8000-00805f9b34fb

En comparant certains de ces **uuid** avec la nomenclature standardisée Bluetooth du fichier « **16-bit UUID Numbers Document.pdf** » disponible à l'adresse <https://btprodspecificationrefs.blob.core.windows.net/assigned-values/16-bit%20UUID%20Numbers%20Document.pdf>, nous pouvons en déduire ceci :

- handle 0x0003 : **2A00** => Device Name
- handle 0x000e : **2A19** => Battery Level
- handle 0x0012 : **2A1F** => Temperature Celsius
- handle 0x0015 : **2A6F** => Humidity

Pour lire le contenu d'un **handle**, on utilise la commande **char-read-hnd** suivie du n° du **handle** qui nous intéresse :

```
[A4:C1:38:9D:D1:8F][LE]> char-read-hnd 0x000e
Characteristic value/descriptor: 24
```

IV - Theengs Gateway

Theengs Gateway (<https://gateway.theengs.io/>) est une **passerelle BLE** vers **MQTT** qui peut interagir avec différentes solutions domotiques comme **NodeRED**, **AWS IOT**, **Home Assistant**, **OpenHAB**, **FHEM**, **IOBroker** ou **DomoticZ**.

Theengs Gateway utilise l'interface **Bluetooth** d'un **Raspberry Pi** ou d'un **PC** à l'aide d'une application écrite en **Python**.

Une solution équivalente nommée **OpenMQTTGateway** (<https://docs.openmqttgateway.com/>) permet de réaliser cette passerelle sur des équipements du type **ESP 82266** et **ESP 32**.

Pour installer **Theengs Gateway** sur un **Raspberry Pi**, il suffit :

- d'installer Python3 : **sudo apt install python3 ;**
- d'installer pip3 : **sudo apt install python3-pip ;**
- d'installer Theengs Gateway : **sudo pip3 install TheengsGateway ;**
- d'exécuter Theengs Gateway : **sudo python3 -m TheengsGateway -H "192.168.X.5" -u "login" -p "pwd" ;**
- Pour connaître les options d'exécution de Theengs Gateway : **sudo python3 -m TheengsGateway -h.**

Depuis un **client MQTT**, par exemple **MQTT Explorer**, on peut visualiser tous les périphériques de type **LYWSD03MMC ATC** via les **topics** **'home/TheengsGateway/BTtoMQTT/A4C138XXYYZZ'** :

The screenshot shows the MQTT Explorer interface. On the left, a tree view shows the topic hierarchy: 192.168.1.6 > esp32 (76 topics, 5174 messages) > homeassistant (24 topics, 48 messages) > SSYS (52 topics, 118219 messages) > home > TheengsGateway > BTtoMQTT. A list of device topics is shown, with 'A4C1381672AC' selected. On the right, the 'Value' field displays the JSON payload for this topic:

```
{
  "name": "ATC_1672AC",
  "id": "A4:C1:38:16:72:AC",
  "rssi": -56,
  "brand": "Xiaomi",
  "model": "LYWSD03MMC",
  "model_id": "LYWSD03MMC_ATC",
  "tempc": 23.9,
  "tempf": 75.02,
  "hum": 73,
  "batt": 82,
  "volt": 2.945
}
```

Remarques :

- Sous **Ubuntu**, on peut installer **Theengs Gateway** via **snap** à l'aide de la commande : **sudo snap install theengs-gateway**. Pour la documentation correspondante on peut aller sur la page <https://github.com/theengs/gateway-snap> ;
- **Theengs Gateway** peut être installé comme module (add-on) de **Home Assistant** ;
- Pour connaître les devices supportés par **Theengs Gateway** voir <https://docs.openmqttgateway.com/prerequisites/devices.html#for-ble-devices>.