Documentation sur l'avancement du TP : Sécurisation des objets

I – Scan des périphérique bluetooth :

On a utilisé la bibliothèque bleak.

Bleak est une bibliothèque Python pour communiquer avec des périphériques Bluetooth Low Energy (BLE) sur différentes plateformes. Elle offre une interface simple pour la recherche, la connexion et l'interaction avec ces périphériques, avec une documentation détaillée et une compatibilité multiplateforme.

```
async def device(mac):
    # scan for devices
    devices = await BleakScanner.discover()
    for d in devices:
        if d.address == mac:
            return d
    return None
```

Cette fonction, nous permet de faire la recherche des périphérique (discover()) et filtre la recherche en se basant sur l'adresse MAC passé en paramètre.

NB. Il faut avant tout importer les bibliothèques nécessaires :

```
import asyncio
from bleak import BleakClient, BleakScanner
```

asyncio : afin des faires des appels asynchrones.

II – Connection avec le péripherique :

```
async def main():
    d = await device('D1:6A:16:01:A2:EC')
    async with BleakClient(d.address) as client:
        if client.is_connected:
            ref = {}
            print('connected')
```

BleakClient, nous permet de se connecter à un périphérique bluetooth en se basant sur l'adresse MAC.

III –Affichage des services (API) sous forme d'un dictionnaire :

La forme souhaitée : UUID-nom du service: handle

```
if client.is_connected:
            ref = {}
            print('connected')
            services = await client.get_services()
            for service in services:
                for charac in service.characteristics:
                    temp = str(str(charac) + " UNKNOWN").split()
                    ref[temp[0] + '-' + temp[3]] = temp[1:][1][:-2]
```

J'ajoute « UNKNOWN » pour les services sans nom. Exemple : les services 2,4,8…

(cheap solution but it works😊).

Résulat:


```
{
'00002a00-0000-1000-8000-00805f9b34fb-UNKNOWN': '2', '00002a01-0000-1000-8000-
00805f9b34fb-UNKNOWN': '4', '00002a04-0000-1000-8000-00805f9b34fb-UNKNOWN':
'6', '00002aa6-0000-1000-8000-00805f9b34fb-UNKNOWN': '8', '00002a05-0000-1000-
8000-00805f9b34fb-UNKNOWN': '11', '8ec90003-f315-4f60-9fb8-838830daea50-
UNKNOWN': '15', '00002a29-0000-1000-8000-00805f9b34fb-UNKNOWN': '19',
'1b0d1201-a720-f7e9-46b6-31b601c4fca1-TRIP': '22', '1b0d1202-a720-f7e9-46b6-
31b601c4fca1-TEMP': '26', '1b0d1203-a720-f7e9-46b6-31b601c4fca1-BATT': '30',
'1b0d1204-a720-f7e9-46b6-31b601c4fca1-SOLAR': '34', '1b0d1205-a720-f7e9-46b6-
31b601c4fca1-NRJ': '38', '1b0d1206-a720-f7e9-46b6-31b601c4fca1-STATUS': '42',
'1b0d1207-a720-f7e9-46b6-31b601c4fca1-GPS': '46', '1b0d1208-a720-f7e9-46b6-
31b601c4fca1-RESET': '50', '1b0d1301-a720-f7e9-46b6-31b601c4fca1-MEM': '54',
'1b0d1302-a720-f7e9-46b6-31b601c4fca1-LIST': '58', '1b0d1303-a720-f7e9-46b6-
31b601c4fca1-NUM': '62', '1b0d1304-a720-f7e9-46b6-31b601c4fca1-READ': '66',
'1b0d1305-a720-f7e9-46b6-31b601c4fca1-SEND': '70', '1b0d1306-a720-f7e9-46b6-
31b601c4fca1-DEL': '74', '1b0d1401-a720-f7e9-46b6-31b601c4fca1-MODE': '79',
'1b0d1402-a720-f7e9-46b6-31b601c4fca1-FW': '83', '1b0d1403-a720-f7e9-46b6-
31b601c4fca1-BL': '86', '1b0d1404-a720-f7e9-46b6-31b601c4fca1-RESET': '89',
'1b0d1405-a720-f7e9-46b6-31b601c4fca1-STATE': '93', '1b0d1406-a720-f7e9-46b6-
31b601c4fca1-TIME': '97', '1b0d1407-a720-f7e9-46b6-31b601c4fca1-ACK': '100',
'1b0d1408-a720-f7e9-46b6-31b601c4fca1-PAUSE': '104', '1b0d1409-a720-f7e9-46b6-
31b601c4fca1-USR': '108', '1b0d140a-a720-f7e9-46b6-31b601c4fca1-LOG': '111',
'1b0d140b-a720-f7e9-46b6-31b601c4fca1-EVENT': '115', '00001503-0000-1000-8000-
00805f9b34fb-NAME': '120', '00001504-0000-1000-8000-00805f9b34fb-COML': '123',
```

```
'00001505-0000-1000-8000-00805f9b34fb-COMH': '126', '00001506-0000-1000-8000-
00805f9b34fb-HWVER': '129'
}
```

Maintenant ca sera plus facile d'appeler les api en se basant soit sur leur nom, UUID ou handle.

IV- Envoyer le payload du NUM vers LIST

- Lire la valeur de NUM :

```
for key in ref:
                if key.split('-')[-1] == 'NUM':
                    await client.read_gatt_char(int(ref[key]))
```

- Publier sur LIST

```
for key in ref:
      if key.split('-')[-1] == 'LIST':
           await client.write_gatt_char(int(ref[key]), bytearray(b'\x17\x00'))
```

pour afficher la résultat, il faut définir une fonction callback, qui va se lancer quand le LOG se met à jour :

```
def getNotified(sender,data):
    print('here is your data sir')
    print(sender, data)
```

ensuite il faut utiliser la fonction start_notify() en passant le handle et le nom de la fonction du callback:

```
for key in ref:
                if key.split('-')[-1] == 'ACK':
                    await client.start_notify(int(ref[key]), getNotified)
```

résultat :

```
here is your data sir
100
bytearray(b'\x02\x13\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00')
```

Étape suivante :

Je pense qu'il faut trouver un service avec la possibilité d'écrire le dessus, afin d'envoyer le bytearray quand on vient de récupérer.