Responsable du document : François DUCLOS

1. Présentation du BLE

Le Bluetooth Low Energy (BLE) et le Bluetooth sont tous deux des WPAN. Contrairement au Bluetooth, le BLE a pour but de transmettre de petites quantités de donnés. Il est donc parfait dans le développement d'appareils IoT.

Sa bande fréquentielle de fonctionnement est aux alentours des 2,4 GHz.

2. Activation du BLE sur la MP1

Cf : init.sh

Dans un premier temps, il faut reset les paramètres du module Bluetooth grâce à la librairie Bluez préinstallée sur le linux de la carte MP1.

```
root@stm32mp1:~# hciconfig reset
```

Réponse :

```
hci0:   Type: Primary  Bus: UART
        BD Address: 43:43:A1:12:1F:AC  ACL MTU: 1021:8  SCO MTU: 64:1
        UP RUNNING PSCAN
        RX bytes:3155 acl:0 sco:0 events:251 errors:0
        TX bytes:38277 acl:0 sco:0 commands:251 errors:0
```

Maintenant il faut réactiver le module Bluetooth à l'aide de son adresse ici hci0= 43:43:A1:12:1F:AC en utilisant la fonction "up".

```
root@stm32mp1:~# hciconfig hci0 up
```

Ici nous avons donc activé notre module Bluetooth de notre carte il ne reste plus qu'à rentrer les adresses MAC de nos Beacons dans la whitelist de notre module Bluetooth.

```
hcitool lewladd 02:04:73:9D:3B:4A #B1
hcitool lewladd 02:05:82:06:25:C2 #B2
hcitool lewladd 02:05:82:06:26:16 #B4
```

Aucune entrée de table des matières n'a été trouvée.

3. Détection et enregistrement des données de nos Beacons

Cf : btmon_search.sh

Avant de scanner nos Beacons pour obtenir les informations qui nous intéresses, il faut créer un fichier ou nous allons enregistrer toutes les informations dedans. Pour cela nous allons utiliser l'outil btmon inclue sur l'OS.

```
btmon -w btsnoop_hci > btsnoop_hci.txt &
```

Cette fonction va tourner en tache de fond, et enregistrer tout ce qu'il se passe dans le module Bluetooth de notre carte dans un fichier .txt

Maintenant nous pouvons scanner nos appareils aux alentours grâce à la librairie Bluez.

```
hcitool lescan --privacy --passive --whitelist &
```

Voilà à cette étape nous devrions avoir un fichier créé du nom de btsnoop_hci.txt contenant un grand nombre d'information sur nos trois beacons préalablement enregistrés dans la white list.

4. Traitement du fichier texte pour obtenir nos RSSI

Cf : RSSI.py

Pour obtenir les informations que nous désirons, il faut traiter le fichier .txt. Ici nous allons utiliser Python. Voici la fonction permettant de ressortir le RSSI correspondant à notre Beacon :

```python
def check_RSSI(addrMac):
    i=0
    j=0
    block_MACaddr = ""
    RSSI_number = 0
    with open('btsnoop_hci.txt') as temp_f:                          #Ouverture de notre fichier texte
        datafile = temp_f.read().split('HCI Event: LE Meta Event')   #On split dans une liste à chaque qu'il y a la chaine de
                                                                     #caractères suivante "HCI Event: LE Meta Event"
    for block in datafile:
        if addrMac in block:
            block_MACaddr = datafile[i]
            block_MACaddr_split = block_MACaddr.split(' ')
            for rssi in block_MACaddr_split :
                if 'RSSI' in rssi:                                   #Recherche du RSSI dans chaque block de la liste
                    RSSI_number=int(block_MACaddr_split[j+1])        #caster la chaine de caractères du RSSI en int pour la return
                j=j+1
            return RSSI_number
        i = i + 1
    return RSSI_number
```

5. Déduction de la distance grâce au RSSI

Cf : RSSI.py

Maintenant que nous avons notre RSSI, il faut traduire cette valeur en une distance à l'aide de la formule suivante :

Distance = 10^[(RSSI_1m – RSSI)/(10xN)]

- Distance en mètres
- RSSI_1m : est la valeur du RSSI à 1 mètre
- RSSI : est la valeur du RSSI trouvé
- N : est un coefficient selon la probabilité d'avoir un signal perturbé (2 : 6)

Nous pourrions utiliser des formules plus complexes mais pour la suite nous allons rester sur celle-ci.

```python
def check_distance(addrMac):
    distance = 0
    rssi_tab = [0 for i in range(3)]
    distance_tab = [0 for i in range(3)]
    for i in range(3):
        subprocess.run(["./btmon_search.sh"])
        distance_tab[i] = 10**((-53-check_RSSI(addrMac))/(10*4))
    for j in range(3):
        if(distance_tab[j]<=0.018):
            distance_tab[j] = 0
    for a in distance_tab:
        if(a==0):
            distance_tab.remove(0)
    if(len(distance_tab)==0):                                    #traitement de la valeur obtenue
        check_distance(addrMac)
    if(len(distance_tab)==2):
        distance = (distance_tab[0]+distance_tab[1])/2
    if(len(distance_tab)==3):
        distance = (distance_tab[0]+distance_tab[1]+distance_tab[2])/3
    if(len(distance_tab)==1):
        distance = (distance_tab[0])
    print(distance)
    return distance
```

```
LE Scan ...
02:05:82:06:25:C2 b2Dongle
LE Scan ...
02:05:82:06:25:C2 b2Dongle
LE Scan ...
1.9071597391913693
```

Nous sommes donc à 1,9m de notre beacon.

6. Automatisation de toutes ces fonctions

Cf : RSSI_scan.sh

Maintenant que nous avons chaque fonction, nous pouvons tout exécuter dans un fichier Bash qui va nous retourner la valeur de la distance entre notre MP1 et le beacon.

```
#!/bin/bash

./init.sh
python3 RSSI.py
```