THƯ VIỆN SỬ DỤNG BLE SIGMESH

Mục lục

I. TÍNH NĂNG

- Delete Network Tạo Network Mới
- Scan Các Thiết Bị Chưa Kết Nối
- Thêm Thiết Bị
- Điều Khiển On/Off Thiết Bị
- Điều Khiển Lightness Thiết Bị
- Xóa Thiết Bị Khỏi Mạng
- Tạo Group Mới
- Thêm Thiết Bị Vào Group
- Xóa Thiết Bị Khỏi Group
- Xóa Group
- Điều Khiển On/Off Group
- Điều Khiển Lightness Group

Tất cả dữ liệu được lưu vào file sigMesh_db.json (giống như mobile app có thể import file vào điện thoại để điều khiển)

Các hàm thư viện hỗ trợ:

- event() :lắng nghe sự kiện từ BLE trả về
- delete_network: xóa mạng tạo mạng mới
- scan_device: start/stop scan các thiết bị mới
- add_device: thêm thiết bị
- onoff_device: on/off thiết bị
- onoff_group: on/off group thiết bị
- lightness_device: điều khiển lightness cho đèn
- lightness_group: điều khiển lightness cho group đèn
- delete_node: xóa thiết bị khỏi mạng
- creat_group: tạo group mới
- add_group: thêm thiết bị vào group
- remove_group: xóa thiết bị ra khỏi group
- delete_group: xóa group

II. CÁCH SỬ DỤNG

Bước 1: import thư viện với tên uart

```
from uart import UART
```

Bước 2: init thư viện với port và baudrate cần sử dụng

Ví dụ: init thư viện với port COM6 và baud=115200

```
uart = UART('COM6', 115200)
```

Bước 3: tạo 1 function để lắng nghe sự kiện trả về từ BLE

function này cần lắng nghe liên tục

ví dụ: để lắng nghe sự kiện trả về từ BLE gọi hàm event() từ thư viện

```
data = uart.event()
```

Bước 4: gọi các hàm cần sử dụng

III. CÁCH SỬ DỤNG CÁC HÀM TRONG THƯ VIỆN

1. delete_network()

mục đích: xóa mạng đang chạy và tự tạo ra 1 mạng sigmesh mới với địa chỉ 199A

Send:

```
uart.delete_network()
```

Feedback:

```
{"method": "delete_network", "params": {"status": "00"}}
```

status=00 trạng thái thành công

2. scan_device(status)

mục đích: scan các thiết bị mới

hàm này có 1 input status(kiểu Bool)

+ True: start scan

+ False: stop scan

start scan:

Send:

```
uart.scan_device(True)
```

Feedback:

```
{"method": "scanStart", "params": {"status": "00"}}
```

data phản hồi thông tin của thiết bị scan được:

```
{"method": "unprovisioning", "params": {"addr type": "00", "gatt supported": "00", "rssi": "-218", "mac": "8C-AE-B1-51-47-19", "uuid": "018071902600008CAEB1514719020000"}}
```

- rssi: rssi của thiết bị

- mac: địa chỉ mac của thiết bị

- uuid: uuid của thiết bị, cần lưu uuid này lại để sử dụng cho bước thêm thiết bị

scan stop:

Send:

```
uart.scan_device(False)
```

Feedback:

```
{"method": "scanStop", "params": {"status": "00"}}
```

3. add_device(name, uuid)

lệnh này thời gian chờ nên để ít nhất 1 phút

mục đích: thêm thiết bị vào mạng hàm này có 2 input:

- name (kiểu String): đặt tên thiết bị
- uuid (kiểu String): uuid của thiết bị có được ở bước Scan

ví dụ: thêm thiết bị có uuid="018071902600008CAEB1514719030000" ; name="light 1"

Send:

```
uart.add_device("light 1","018071902600008CAEB1514719030000")
```

Feedback:

```
{"method": "add_device", "params": {"name": "light 1", "status": "00", "address": "0003"}}
```

- status:
  - + 00: thành công
  - + 01: thiết bị đã tồn tại, đặt tên khác
- name: tên của thiết bị đã set ở lệnh gửi đi
- address: địa chỉ của thiết bị, cần lưu giá trị này lại để sử dụng cho các lệnh điều khiển thiết bị sau

4. onoff_device(unicastAddress, status)

Mục đích: điều khiển on/off thiết bị hàm này có 2 input:

- unicastAddress (kiểu String): địa chỉ của thiết bị có được sau lệnh add_device
- status (kiểu Bool): trạng thái on/off cần điều khiển
  - + True: ON
  - + False: OFF

ví dụ: Điều khiển OFF đèn 0003

Send:

```
uart.onoff_device("0003", False)
```

Feedback:

```
{"method": "onOffStatus", "params": {"address": "0003", "status": "OFF"}}
```

5. lightness_device(unicastAddress, value)

Mục đích: điều khiển độ sáng đèn hàm này có 2 input:

- unicastAddress (kiểu String): địa chỉ của thiết bị có được sau lệnh add_device

- value (kiểu int): độ sáng của đèn 0->100%

ví dụ: Điều khiển đèn sáng 50%

Send:

```
uart.lightness_device("0003", 50)
```

Feedback:

```
{"method": "lightnessStatus", "params": {"address": "0003", "value": 50}}
```

6. delete_node(unicastAddress)

Mục đích: xóa thiết bị ra khỏi mạng hàm này có 1 input:

- unicastAddress (kiểu String): địa chỉ của thiết bị có được sau lệnh add_device

ví dụ: Xóa đèn 0003

Send:

```
uart.delete_node("0003")
```

Feedback:

```
{"method": "delete_node", "params": {"status": "00", "unicast": "0003"}}
```

7. creat_group(name)

Mục đích: Thêm group mới hàm này có 1 input:

- name (kiểu String): tên group cần thêm

ví dụ: Thêm group với tên group1

Send:

```
uart.creat_group("group1")
```

Feedback:

```
{"method": "creat_group", "params": {"status": "00", "name": "group1", "address": "C000"}}
```

- status: trạng thái, thành công = 00
    + 00: thành công
    + 01: group đã tồn tại, đặt tên khác
- name: tên group
- address: địa chỉ của group, cần lưu lại giá trị này để sử dụng cho các lệnh điều khiển group

8. add_group(addressDevice, addressGroup)

Mục đích: Thêm thiết bị vào group đã tạo hàm này có 2 input:

- addressDevice (kiểu String): địa chỉ của thiết bị cần thêm vào group, có được sau lệnh add_device

- addressGroup (kiểu String): địa chỉ của group cần thêm thiết bị vào, địa chỉ này có được ở lệnh tạo group

ví dụ: Thêm đèn 0003 vào group C000

Send:

```
uart.add_group("0003","C000")
```

Feedback:

```
{"method": "add_group", "params": {"status": "00"}}
```

9. onoff_group(unicastAddress, status)

Mục đích: điều khiển on/off group thiết bị hàm này có 2 input:

- unicastAddress (kiểu String): địa chỉ group có được sau lệnh tạo group

- status (kiểu Bool): trạng thái on/off cần điều khiển

    + True: ON

    + False: OFF

ví dụ: Điều khiển OFF group C000

Send:

```
uart.onoff_group("C000", False)
```

Feedback:

```
{"method": "onOffStatus", "params": {"address": "0003", "status": "OFF"}}
```

Trạng thái trả về là trạng thái của từng đèn trong group

10. lightness_group(unicastAddress, value)

Mục đích: điều khiển độ sáng group đèn hàm này có 2 input:

- unicastAddress (kiểu String): địa chỉ của group thiết bị có được sau lệnh tạo group

- value (kiểu int): độ sáng của đèn 0->100%

ví dụ: Điều khiển group đèn C000 sáng 50%

Send:

```
uart.lightness_group("C000", 50)
```

Feedback:

```
{"method": "lightnessStatus", "params": {"address": "0003", "value": 50}}
```

Trạng thái trả về là trạng thái của từng đèn trong group

11. remove_group(addressDevice, addressGroup)

Mục đích: xóa thiết bị ra khỏi group hàm này có 2 input:

- addressDevice (kiểu String): địa chỉ của thiết bị cần thêm vào group, có được sau lệnh add_device

- addressGroup (kiểu String): địa chỉ của group cần thêm thiết bị vào, địa chỉ này có được ở lệnh tạo group

ví dụ: xóa đèn 0003 ra khỏi group C000

Send:

```
uart.remove_group("0003","C000")
```

Feedback:

```
{"method": "remove_group", "params": {"status": "00"}}
```

12. delete_group(addressGroup)

Mục đích: xóa group hàm này có 1 input:

- addressGroup (kiểu String): địa chỉ của group, địa chỉ này có được ở lệnh tạo group

ví dụ: xóa group C000

Send:

```
uart.delete_group("C000")
```

Feedback:

```
{"method": "delete_group", "params": {"status": "00"}}
```

- status:

    + 00: thành công

    + 01: group chưa có