Xnode로 배우는

저전력 무선 네트워크 프로그래밍

9 Zigbee와 BLE

Zigbee와 BLE

- BLE(Bluetooth Low Energy)
  - 탁월한 전원 관리 기능으로 단거리 저 대역폭 연결 지원
  - 개인 영역 네트워크가 대용량 데이터 스트림을 처리할 필요가 없음
  - 배터리가 몇 달 또는 몇 년 동안 지속하여야 하는 상황에서 사용
  - 하드웨어가 필요한 통신 기능만 구현하도록 허용하는 클라이언트/서버 아키텍처 구현
    - 비용, 배터리 및 대역폭 절약
  - BLE 네트워크는 이론적으로 엄청난 수의 장치 포함
    - 대역폭, 물리적 공간 등의 범위는 단일 BLE 네트워크의 크기를 하위 수백 개의 노드로 제한

Zigbee와 BLE

- BLE 는 지점 간 프로토콜
  - 네트워크의 물리적 크기를 BLE의 일반적인 10미터 범위로 제한
  - 홈 오피스에는 적합하지만, 농업 모니터링 애플리케이션, 도시 거리 조명 제어 등에는 부적합
- 많은 BLE 애플리케이션은 스마트 폰을 게이트웨이로 사용하도록 설계
  - 스마트 폰이 있는 경우에만 작동
  - 스마트 워치 또는 피트니스 밴드와 같은 웨어러블의 경우 적합
  - 상업용 및 산업용 애플리케이션에 사용되는 센서 등 무인 상태인 경우
    - 스마트 폰 게이트웨이를 구현할 수 없거나 구현할 수 없으므로 부적합
- BLE 는 Bluetooth Classic보다 훨씬 낮은 대역폭
- 미디어 스트리밍에 효과적으로 사용 불가

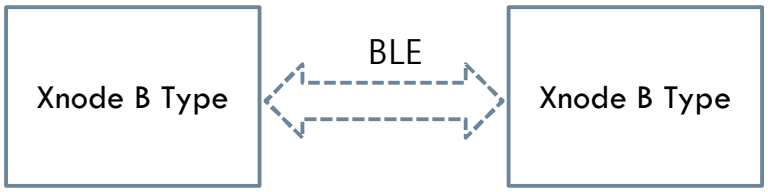
Zigbee와 BLE

- Xnode는 Zigbee통신과 BLE통신 모두 지원
- BLE와 Zigbee의 공통점
  - IEEE802.15 에 속함
  - 2.4GHz의 대역폭 사용
  - 저전력으로 구현 가능
- BLE와 Zigbee의 차이점
  - 통신 거리, 통신 속도, 네트워크 구성 등
  - 사용처에 따라 선택 사용 필요

Zigbee와 BLE

- 예제를 위한 준비물

준비물	
1	PC 1ea
2	XNode B Type 2ea
3	Micro type USB cable 2ea



- 예제 진행을 위해서 USB > Library > CORE > lib 폴더를 2대의 XNode B Type에 복사하여 사용하시기 바랍니다.

ble class

- pop 라이브러리의 ble class에서 제공하는 메소드
  - avtive(mode) : Xnode의 BLE 기능 활성 여부 결정
    - mode = 'True' : enable
    - mode = 'False' : disable
  - config('mac') : BLE MAC address 반환 (bytesarray)

ble class

- gap_scan(duration_ms=0): 주변의 BLE 장치의 advertisement 검색 및 반환
  - duration_ms : Millisecond 단위로 검색 시간 설정. 0일 경우 검색 결과 있을 때까지 검색.
  - advertisement format : 딕셔너리 타입
    - address: BLE MAC address (bytes object)
    - addr_type: address field에 포함된 address type
    - connectable: BLE central-mode devices의 연결 가능 여부.
    - rssi: received signal strength(dBm)
    - payload: raw advertisement payload (bytes object)
  - get() : 현재 queue에 있는 수신된 모든 advertisement 목록 반환
  - any() : 현재 queue에 수신된 advertisement 존재 여부 반환
  - stop() : 진행 중인 scan 작업 중지

ble class

- gap_advertise(interval_us, adv_data=None) : Start or stop GAP advertisements.
  - interval_us : micro sec 단위로 지정된 간격으로 advertisement 시작
    None으로 설정되면 advertisement 정지
    None이 아닌 경우 최소 20,000 micro sec ~ 40.96 sec 사이로 설정해야 함.
  - adv_data : advertisements에 포함될 payload. None이나 31byte 길이의 bytearray.
    다른 장치에서 유효한 BLE advertisement로 인식되려면 <adv_data>를 형식화 필요
    - "Hello" -> b"₩x06₩x08Hello"

ble class

- 현재 장치의 BLE MAC 주소를 출력하는 코드(모듈마다 MAC 주소 다름)

```
01:     from pop import ble
02:
03:     print("address : ", ble.config("mac"))
```

```
address :  b'\x08k\xd7*\xe0)'
```

advertisement

- advertisement data를 설정 및 실행하는 코드

```
01:     from pop import ble
02:
03:     ble.active(True)
04:
05:     payload = bytearray()
06:     payload.append(len("Hello") + 1)
07:     payload.append(0x08)
08:     payload.extend("Hello")
09:
10:     ble.gap_advertise(100000, payload)
```

- 코드 실행 후 advertisement scan 코드가 실행 중인 장치에서 검색된 결과

```
{'addr_type': 0, 'payload': b'\x06\x08Hello', 'connectable': True
, 'address': b'\x00\ro\xc7E\xdd', 'rssi': -32}
```

advertisement scan

- 주변의 BLE Advertisement를 5초간 검색 및 출력 코드

```
01:     from pop import ble
02:
03:     ble.active(True)
04:
05:     scan=ble.gap_scan(duration_ms=5000)
06:
07:     print("start scan")
08:     for i in scan:
09:         print(i)
10:     print("finish")
```

```
True
start scan
{'addr_type': 0, 'payload': b'\x02\x01\x06\x13\tXBee3 DigiMesh 2.4', 'connectable'
: True, 'address': b'\x08k\xd7*\xca\xac', 'rssi': -72}
finish
```

advertisement scan

- 원하는 advertisement를 찾을 때까지 ble advertisement를 검색하는 코드
  - ble.gap_scan()에 아무런 인자를 주지 않는 경우
    - 장치는 ble.gap_scan().stop()의 명령이 있을 때까지 계속해서 주변 ble advertisement 검색
  - payload에 Hello가 검색되면 프로그램 종료

```
01:     from pop import ble
02:
03:     ble.active(True)
04:
05:     scan=ble.gap_scan()
06:
07:     for i in scan:
08:         print(i)
09:         if str(i["payload"]).find('Hello') >= 0:
10:             scan.stop()
```

```
{'addr_type': 0, 'payload': b'\x06\x08Hello', 'connectable': True
, 'address': b'\x00\ro\xc7E\xdd', 'rssi': -37}
```

advertisement scan

- stop() 없이 with를 사용한 코드

```
01:         from pop import ble
02:
03:         ble.active(True)
04:
05:         with ble.gap_scan() as scan:
06:             for i in scan:
07:                 if 'Hello' in i["payload"]:
08:                     print(i)
09:                     break
```

```
{'addr_type': 0, 'payload': b'\x06\x08Hello', 'connectable': True
, 'address': b'\x00\ro\xc7E\xdd', 'rssi': -37}
```