BBTAN

Android 프로그램 가이드

2020년 10월 14일 수요일

Mobilian

개정이력

2019.02.15
- Android 프로그램 V1.0a 배포

2020.10.14
- BBTAN Library 안정성 업데이트
- Transport Layer 교체

1. 개요

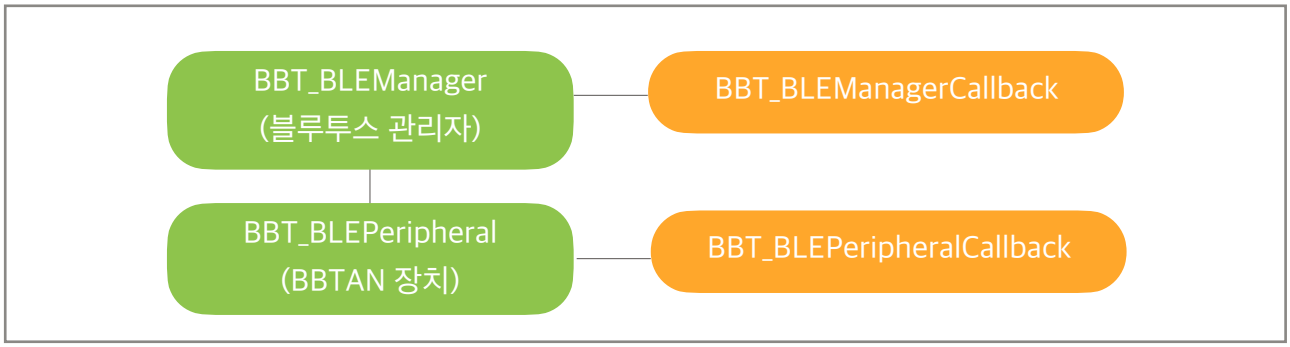
이 문서는 BBTAN을 Android에서 사용 할 수 있도록, 테스트 프로그램을 작성하는 시나리오를 설명한다.

- BBTANLibrary 프레임워크
- 프레임워크 사용 시나리오
  (1) BBTAN 장치 검색
  (2) 장치 연결 및 데이터 송.수신
- 테스트 앱 사용

2. BBTanLibrary 프레임워크

Android 개발자가 Bluetooth 장치와 연결 및 데이터 송수신에 도움을 주는 인터페이스를 포함하고 있어 블루투스에 관한 이해가 없이도 빠르게 개발 할 수 있도록 구성되어 있다.

프레임워크의 전체 구성은 <그림1>과 간략한 설명은 <표1>을 참고한다.



<그림1. 프레임워크 구성>

이름	설명
BBT_BLEManager **BBT_BLEManagerCallback**	- 블루투스 상태 알림 - 장치 Scan 상태 Notification callback
BBT_BLEPeripheral **BBT_BLEPeripheralCallback**	- Bluetooth 장치 - Bluetooth 장치의 상태 Notification callback

<표1. 클래스 설명>

2.1. BBT_BLEManager

BBT_BLEManager는 Bluetoth 장치의 검색 및 연결을 수행하는 관리자 역할을 수행한다. 이 클래스를 사용할 때는 클래스메소드 **static** BBT_BLEManager getInstance(Context context) 를 통해 접근하도록 한다.

블루투스 관리자 객체 사용

+ **static** BBT_BLEManager getInstance(Context context)

BBT_BLEManager의 단일 인스턴스이며, 접근시 자동으로 생성된다. 블루투스 관리자의 상태는 BBT_BLEManagerCallback interface를 구현하는 클래스는 주변의 장치가 scan되면 Notification을 받게된다.

Notification을 받기위해 **void** addListener(BBT_BLEManagerCallback callback)를 호출하고 더이상 Notification을 받지 않으려면 void removeListener(BBT_BLEManagerCallback callback)을 호출 해 준다.

** BBTAN_Test의 ScannerDialogFragment를 참조*

장치 스캔

- void startScan(List<ScanFilter> scanFilterList)

주변의 Bluetooth 장치의 스캔을 시작한다

** ScanFilter를 사용하면 주변의 장치중 Name / MAC / UUID / ServiceData / Manufacturer data 형식에 맞는 항목만을 검색 할 수 있다.*

- void stopScan()

스캔을 중지한다

2.2. BBT_BLEPeripheral

BBT_BLEPeripheral 클래스는 Core Bluetooth 클래스(BluetoothDevice)를 포함하고 있으며, BBTAN 장치의 정보, 연결상태 그리고 데이터를 송.수신하는 역할을 담당한다. 사용자는 BBT_BLEPeripheralCallback interface를 구현만 하면 1) 연결상태 Notification을받게되고 2)자동으로 Data를 수신받을 수 있다.

데이터 수신은 TCP / UDP 형태의 2가지 종류가 있으며 아래와 같다.

(TCP 형태 수신)

- void onBBT_PeripheralTCPDataReceived(BBT_BLEPeripheral device, **byte**[] buf);

(UDP 형태 수신)

- **void** onBBT_PeripheralUDPDataReceived(BBT_BLEPeripheral device, **byte**[] buf)

* BBT_BLEPeripheral 객체는 BBT_BLEManager를 통해 자동으로 생성되는 객체이므로 수동으로 인스턴스를 만들 필요는 없다.

Transport Layer 사용 / 사용해제

BBT Bluetooth 모듈은 Transport layer를 사용/비사용 모드로 설정 할 수 있는데, 부착된 MCU가 AT 명령어를 이용해 설정 할 수 있다(AT+GET_TRAN/AT+SET_TRAN). 해당 설정에 따라 앱 개발자는 BBTAN Library에서 Bluetooth Peripheral의 Transport class를 지정 해 주어야 한다. Android Library는 기본으로 Transport layer를 사용 하도록 되어 있으므로 Transport layer를 사용하지 않으려면 연결하기 전 BBT_BLEPeripheral::setTransportInstanceCallback 메소드를 이용해 재 설정 한다.

- Legacy transport (Transport layer 사용 안함)

```
mDevice.setTransportInstanceCallback(bbt_transportCallback ->    {
        return new BBT_TransportLegacy(bbt_transportCallback);
    });
```

- Transport layer 사용

```
mDevice.setTransportInstanceCallback(bbt_transportCallback ->    {
        return new BBT_TransportSlideWindow(bbt_transportCallback);
    });
```

* *BBT Bluetooth 장치는 기본으로 Transport Layer가 사용하지 않음으로 출고 됨*
* *ConnectDialogFragment Sample을 참조*

장치와 연결

- void connect()
  Bluetooth 장치와 연결을 시도한다.
- void disconnect()
  장치와 연결을 해제한다

상태정보

Bluetooth 장치의 상태는 BBT_BLEPeripheralCallback interface의 onBBT_PeripheralStateChanged 를 통해 전달된다.

```
    void onBBT_PeripheralStateChanged(BBT_BLEPeripheral device,
                                      BBT_BLEPeripheral.DeviceState state)
    public enum DeviceState
    {
        disconnected,               // 연결이 종료됨
        connecting,                 // 연결중
        connected,                  // 연결됨
        discover_service,           // 서비스를 탐색중
        discover_characteristics,   // 서비스 characteristics를 탐색중
        disconnecting,              // 연결 해제중
        ready,                      // * 송.수신가능
    }
```

* *DeviceState.**ready*** 상태일때 정상적으로 데이터를 송.수신 할 수 있다.

데이터 송신

데이터 송신은 writeTCP / writeUDP 메소드를 사용한다. 두 메소드의 차이점은 writeTCP는 내부의 Transport 계층에서 전송을 보장해주는 전송 방식이며, writeUDP는 일회성으로 write는 되지만 디바이스에서 수신 받지 못하더라도 재전송이 일어나지 않는 전송 방식이다.

- void writeTCP(byte[] buffer)
  * 전송이 보장 / 장치의 **Transport Layer** 사용으로 설정되어 있어야만 보장
- void writeUDP(byte[] buffer)
  * 전송이 보장되지 않는 일회성 전송방식.

데이터 수신

BBT_BLEPeripheralCallback을 구현하면 자동으로 데이터를 받게된다. 데이터는 송신과 마찬가지로 TCP/UDP 형태의 수신으로 나뉘어져 있다.

(TCP 형태 수신)

```
void onBBT_PeripheralTCPDataReceived(BBT_BLEPeripheral device, byte[] buf)
```

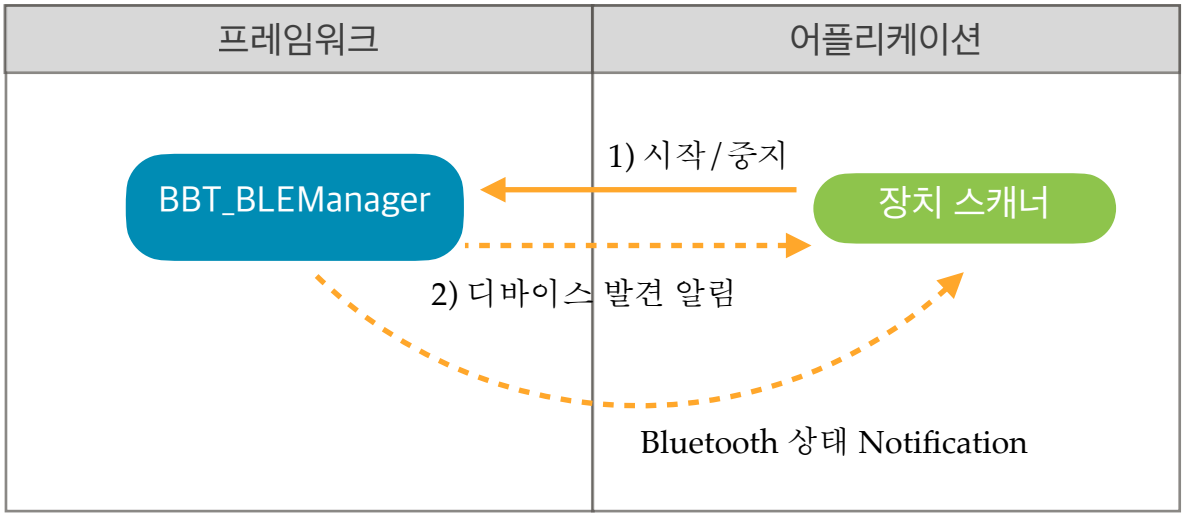
(UDP 형태 수신)

- **void** onBBT_PeripheralUDPDataReceived(BBT_BLEPeripheral device, **byte**[] buf)

3. 프레임워크 사용 시나리오

3.1. 장치 스캐너

주변에 있는 BBTAN 장치와 연결하기 위해서는 스캔작업을 해야하며, BBT_BLEManager 의 startScan/stopScan 메소드를 호출하면 각각 시작/중지 작업을 할 수 있다. BBT_BLEManager는 주변의 Bluetooth 장치가 발견되면 이를 BBT_BLEManagerCallback을 구현하고 addListener를 통해 등록된 인스턴스에 Notification을 보낸다.



<그림 1. 스캐너 구현>

스캐너 라이프 사이클

1. 스캐너 사용을 위해 *BBT_BLEManager* 인스턴스 생성

```
BBT_BLEManager manager = BBT_BLEManager.sharedInstance(this)
```

2. 스캐너 이벤트 수신 *Listener* 등록 / 시작

   스캐너 이벤트를 수신하기 위해 BBT_BLEManagerCallback 을 구현한다

   (ScannerDialogFragment 예제를 참조)

   manager.addListener(adapter);

   manager.startScan(null);

   * 더이상 이벤트를 수신 받지 않으려면 *removeListener*를 사용

3. 검색된 *Device* 수신

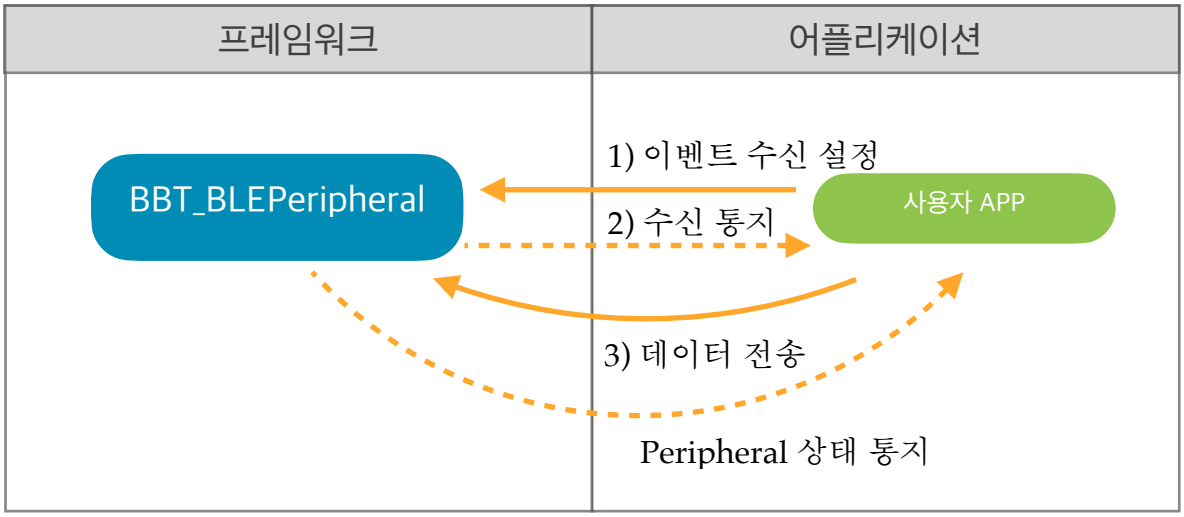
   ***void** onBLEManagerDiscovered(BBT_BLEPeripheral device)*

4. 스캐너 중지

   manager.stopScan();

3.2. 장치연결 및 송.수신

스캐너를 통해 발견된 블루투스 장치 (BBT_BLEPeripheral)와 연결을 하려면 BBT_BLEManager 의 connect / disconnect 메소드를 이용하며, 데이터 수신은 BBT_BLEPeripheralCallback를 구현하고 이벤트 수신을 설정하면 된다. 그리고 데이터 송신은 BBT_BLEPeripheral 객체의 writeXXX 메소드를 이용한다.



<그림3. BBTAN Peripheral 사용 흐름>

장치 사용 시나리오

```
BBT_BLEPeripheral peripheral       (스캐너를 통해 발견된 장치)

peripheral.addListener(callback); (장치 이벤트 수신 설정)

    …
    peripheral.writeTCP(data)      (데이터 송신)
    …

peripheral.removeListener(callback);  (장치 이벤트 수신 해제)
```

장치의 데이터 수신 및 상태 알림 *(BBT_BLEPeripheralCallback 구현)*

```
(장치와 연결되었을때)
void onBBT_PeripheralConnected(BBT_BLEPeripheral device)

(장치와 연결이 해제 되었을 때 호출)
void onBBT_PeripheralDisconnected(BBT_BLEPeripheral device)
```

(다비이스의 상태가 변경 되었을 때 호출)

```
void onBBT_PeripheralStateChanged(BBT_BLEPeripheral device,
                                  BBT_BLEPeripheral.DeviceState state)
```

(TCP 형태의 데이터를 수신 했을 때 호출 됨)

```
void onBBT_PeripheralTCPDataReceived(BBT_BLEPeripheral device, byte[] buf)
```

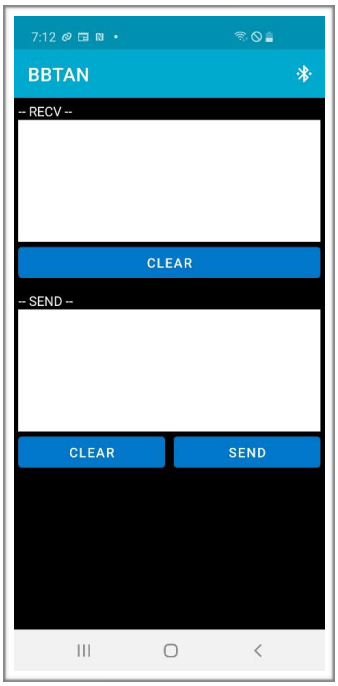
(UDP 형태의 데이터를 수신 했을 때 호출 됨)

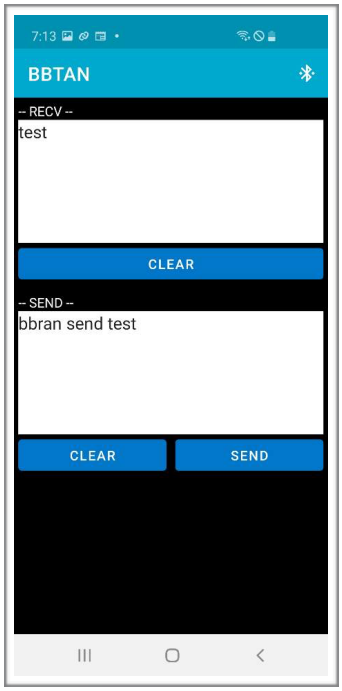
```
void onBBT_PeripheralUDPDataReceived(BBT_BLEPeripheral device, byte[] buf)
```

2. 테스트 앱 사용

2.1. 앱 실행 화면

1) 메인화면 2) 스캔화면 3) 테스트화면





- 메인화면 / 테스트화면

1) Bluetooth Button : 연결 할 장치를 스캔 / Disconnect 한다.
2) Clear : Send / Recv 텍스트를 삭제한다.
3) Send : 입력한 텍스트를 전송한다

- 스캔화면

주변에 있는 장치를 스캔하고 정보를 표시한다.

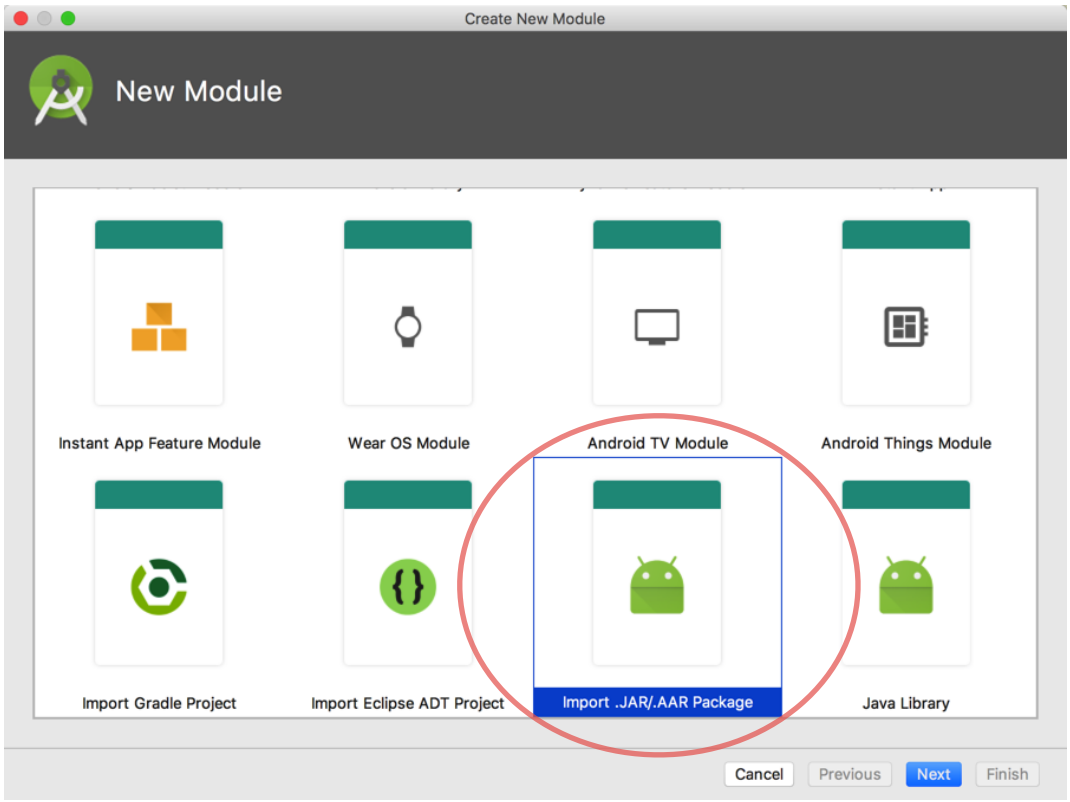
앱 개발 설정

(1) Android Studio를 이용 해 새로운 프로젝트를 생성한다.
(2) Gradle App 설정에서

- minSdkVersion을 18 이상으로 설정한다
- compileOptions 를 JavaVersion 1.8로 설정한다

```
android {
    compileSdkVersion 27
    defaultConfig {
        applicationId “com.mobilian.bbtantest"
        minSdkVersion 18
        targetSdkVersion 27
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner
"android.support.test.runner.AndroidJUnitRunner"
    }
…
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}
```

(3) File -> New -> New Module 을 선택하여 Library 폴더의 BBTanLibrary-release.aar을 추가한다.



(4) File -> Project Structure 에서 app 의 dependencies 항목에서 + 를 선택하여 Module Dependency 화면에서 BBTanLibrary-release 를 선택해준다