
BBTAN

iOS 프로그램 가이드 v1.0

2020년 10월 14일



개정이력

2019.02.15

- iOS 프로그램 V1.0a 배포

1. 개요

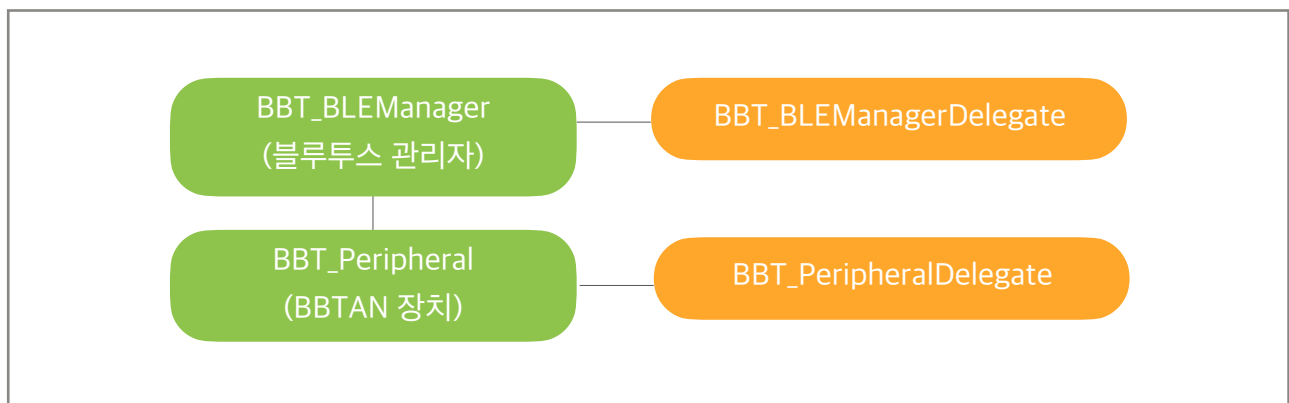
이 문서는 BBATAN을 iOS에서 사용 할 수 있도록, 테스트 프로그램을 작성하는 시나리오를 설명한다.

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

2. BBTanLibrary 프레임워크

iOS 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_BLEManagerDelegate	- 블루투스 상태 알림 - 장치 검색 및 연결
BBT_Peripheral BBT_PeripheralDelegate	- 장치와 연결상태 알림 - 데이터 송.수신

<표1. 클래스 설명>

2.1. BBT_BLEManager

BBT_BLEManager 는 Bluetooth ON/OFF 상태를 관리하고 Bluetooth 장치의 검색 및 연결을 수행하는 관리자 역할을 수행한다. 이 클래스를 사용할 때는 클래스메소드 + (BBT_BLEManager*) sharedInstance 를 통해 접근하도록 한다.

블루투스 관리자 객체 사용

+ (BBT_BLEManager*) sharedInstance

BBT_BLEManager의 단일 인스턴스이며, 접근시 자동으로 생성된다. 블루투스 관리자의 상태는 BBT_BLEManagerDelegate protocol을 구현하는 클래스는 통지를 받게 된다.

통지를 받는 방법은 addDelegate / removeDelegate를 호출하면된다. 단순 Delegate가 아닌 Multicast Delegate 이므로 addDelegate / removeDelegate를 호출 해 주어야 한다.

블루투스 상태

@property (nonatomic, readonly) BBT_BLEManagerState state

BBT_BLEManager의 현재 상태를 알 수 있는 Property

- BBT_BLEManagerState_ready, 블루투스가 켜진 상태 (즉시 idle상태로 이동)
- BBT_BLEManagerState_idle, idle 상태
- BBT_BLEManagerState_scanning, 장치를 스캔하는 상태
- BBT_BLEManagerState_unavailable, 블루투스가 꺼진 상태

@property (nonatomic, readonly) BOOL powerState

블루투스 전원이 ON/OFF인지 확인

장치 스캔

- (void) startDiscover
장치의 스캔을 시작한다
- (void) stopDiscover
장치의 스캔을 중지한다

장치와 연결

- (void) connect:(BBT_Peripheral*) device
장치와 연결을 시도한다.
* BBT_Peripheral 직접 생성하지 않고, 스캔시에 전달된 객체로 전달한다.
- (void) disconnect:(BBT_Peripheral*) device
장치와 연결을 해제한다
- + (BBT_Peripheral*) retrievePeripheralWithUUID:(NSString*) uuidString;
저장된 UUID를 이용해서 Card Peripheral을 생성한다.

2.2. BBT_Peripheral

BBT_Peripheral 클래스는 Core Bluetooth 클래스(CBPeripheral)를 포함하고 있으며, BBTAN 장치의 정보, 연결상태 그리고 데이터를 송.수신하는 역할을 담당한다. 사용자는 BBT_PeripheralDelegate 프로토콜을 구현만 하면 1) 연결상태를 통지받게되고 2)자동으로 Data를 수신받을 수 있다.

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

(TCP 형태 수신)

- (void)BBT_Peripheral:(id)peripheral didReceivedTCPData:(NSData*)data;

(UDP 형태 수신)

- (void)BBT_peripheral:(id)peripheral didReceivedUDPData:(NSData*)data;

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

장치 정보

@property (nonatomic, copy) NSString* name
장치의 이름

@property (nonatomic, readonly) BOOL ready
장치와 연결되어 있고 정상적으로 송.수신 가능한 상태를 표시

상태정보

@property (nonatomic, assign) BBT_PeripheralState state
현재 상태를 알 수 있으며 세부 상태는 다음과 같다

BBT_PeripheralState_disconnected	연결종료
BBT_PeripheralState_connecting	연결중
BBT_PeripheralState_connected	연결됨
BBT_PeripheralState_disconnecting	연결 해제중
BBT_PeripheralState_failedToConnect	연결실패
BBT_PeripheralState_discoveringServices	서비스탐색중
BBT_PeripheralState_discoveringCharacteristics	특성탐색중
BBT_PeripheralState_unavailable	사용불능
BBT_PeripheralState_ready	*송.수신가능
BBT_PeripheralState_notSupported	지원하지않음

* **BBT_PeripheralState_ready** 상태일때 정상적으로 데이터를 송.수신 할 수 있다.

데이터 송신

데이터 송신은 writeTCPData / writeUDPData 메소드를 사용한다. 두 메소드의 차이점은 writeTCPData는 내부의 Transport 계층에서 전송을 보장해주는 전송 방식이며, writeUDPData는 일회성으로 write는 되지만 디바이스에서 수신 받지 못하더라도 재전송이 일어나지 않는 전송방식이다.

- (void) writeTCPData:(NSData*)data;
 NSData를 장치로 전송한다.
 * 전송이 보장.
- (void) writeUDPData:(NSData*)data;
 * 전송이 보장되지 않는 일회성 전송방식.

데이터 수신

BBT_PeripheralDeleagate를 구현하고, AddDeleage 를 통해 이벤트를 수신하게 설정되어 있으면 자동으로 데이터를 받게된다. 데이터는 송신과 마찬가지로 TCP/UDP 형태의 수신으로 나뉘어져 있다.

(TCP 형태 수신)

- (void)BBT_Peripheral:(id)peripheral didReceivedTCPData:(NSData*)data;

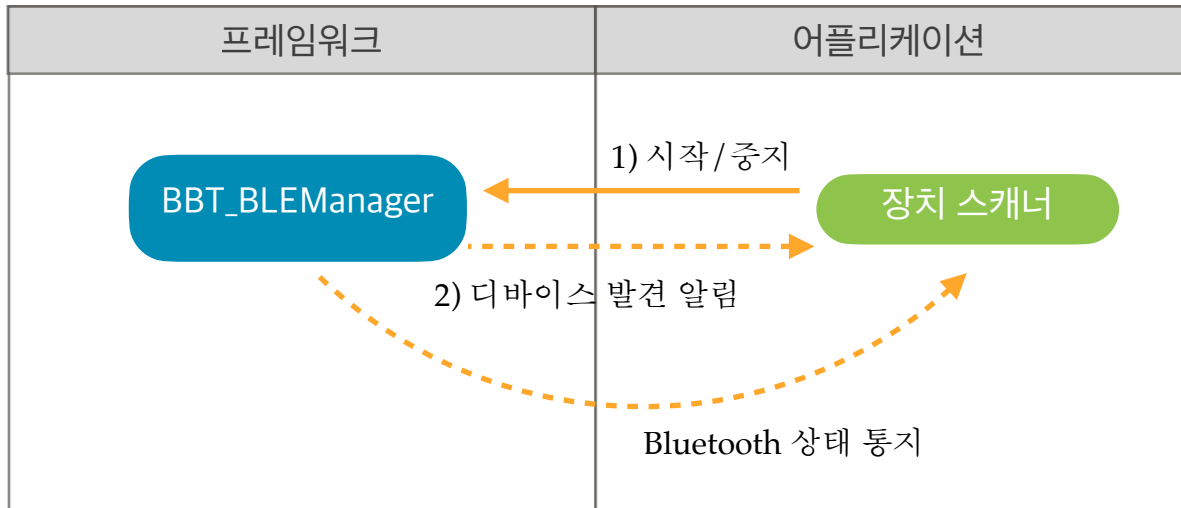
(UDP 형태 수신)

- (void)BBT_Peripheral:(id)peripheral didReceivedUDPData:(NSData*)data;

3. 프레임워크 사용 시나리오

3.1. 장치 스캐너

주변에 있는 BBTAN 장치와 연결하기 위해서는 스캔작업을 해야하며, BBT_BLEManager 의 startDiscover/stopDiscover 메소드를 호출하면 각각 시작/중지 작업을 할 수 있다. BBT_BLEManager는 카드리더 장치가 발견되면 이를 통지한다.



<그림 2. 스캐너 구현>

스캐너 라이프 사이클

```
[[BBT_BLEManager sharedInstance] addDelegate:self]; (이벤트 수신)

[[BBT_BLEManager sharedInstance] startDiscover]; (스캔시작)

...

[[BBT_BLEManager sharedInstance] stopDiscover]; (스캔중지)

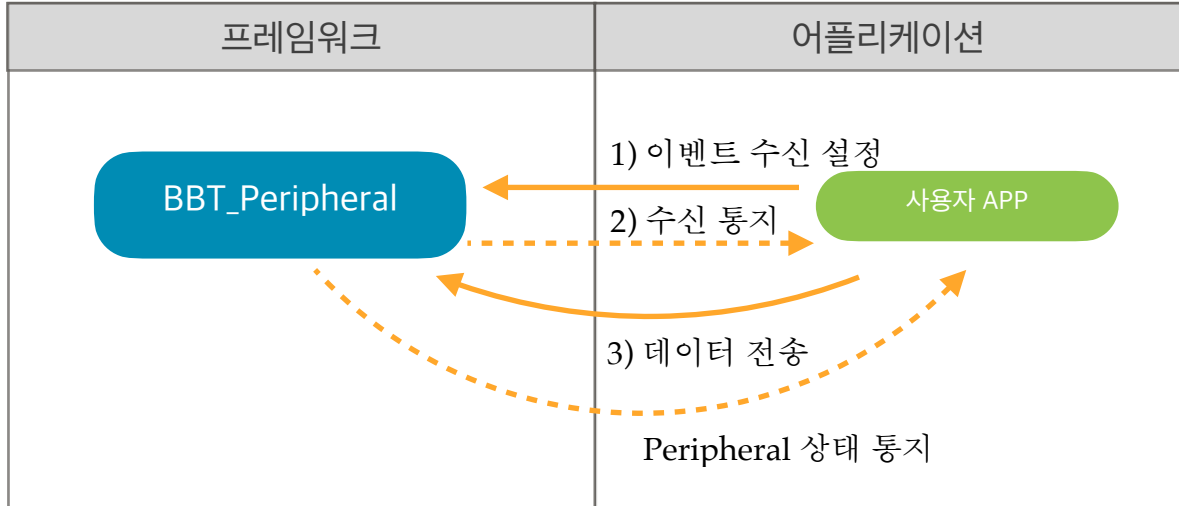
[[BBT_BLEManager sharedInstance] removeDelegate:self]; (이벤트 수신해지)
```

장치 발견 통지

```
- (void)manager:(id)manager didDiscoveredPeripheral:
(BBT_Peripheral *)peripheral
{
    ... 발견된 장치 표시
}
```

3.2. 장치연결 및 송.수신

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



<그림3. BBTAN Peripheral 사용 흐름>

장치 사용 시나리오

BBT_Peripheral* peripheral (스캐너를 통해 발견된 장치)

[self.peripheral addDelegate:self]; (장치 이벤트 수신 설정)

```

...
[self.peripheral writeTCPData:data] (데이터 송신)
...

```

[self.peripheral removeDelegate:self]; (장치 이벤트 수신 해제)

데이터 수신 (BBT_PeripheralDelegate 구현)

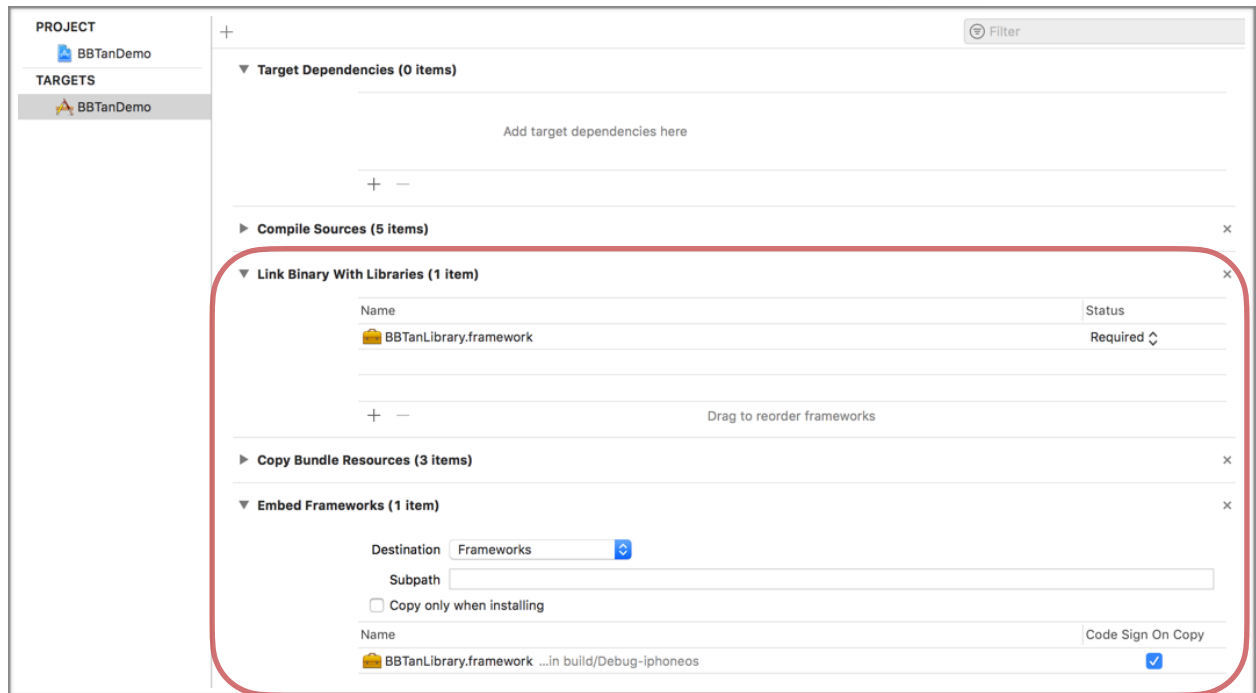
- (void)BBT_Peripheral:(id)peripheral didReceivedTCPData:(NSData *)data
TCP 데이터 수신
- (void)BBT_Peripheral:(id)peripheral didReceivedUDPData:(NSData *)data
UDP 형태의 데이터 수신

장치의 상태 수신

- (void)BBT_Peripheral:(id)peripheral didChangeState:
(BBT_PeripheralState)state

앱 개발 설정

- (1) Xcode 를 이용 해 새로운 프로젝트를 생성한다.
- (2) 배포된 BBTanLibrary.framework 를 Targets > Build Phases 에 추가 해 준다.



- (3) BBTLibrary 액세스를 위해 다음과 같이 헤더를 임포트 해 준다.

```
#import "BBTanLibaray/BLE_BLEManager.h"
```

- (4) 프레임워크 설명을 참고하여, BBT_BLEManager / BBT_Peripheral 클래스를 액세스한다.
 - BBT_BLEManager 는 Class member인 sharedInstance 를 통해 접근한다.
 - 위의 두 클래스는 단순 delegate가 아닌 Multicast delegate 이므로 addDelegate / removeDelegate를 통해 delegate 로 등록 해 준다.
 - BBT_BLEManager의 delegate 는 BBT_BLEManagerDelegate를 구현한다
 - BBT_Peripheral의 delegate는 BBT_PeripheralDelegate를 구현한다