Android FeasyBlue SDK API

Reference Manual

Version 1.0

Revision History

Version	Date	Notes	Author
1.0	2018/11/22	First Release	Younger

Table of Contents

1. Introduction

This reference manual presents design guidelines for software engineers that use Android FeasyBlue SDK to create Android App for Bluetooth connectivity requirements.

1.1 Android System Version Requirements

- Android 4.3 and above

1.2 Supported Android devices

- Phone or tablet with OS of Android 4.3 and above.

1.3 Supported Bluetooth Profile

- GATT (Generic Attribute Profile, relevant to BLE)
- SPP (Serial Port Profile)

2. Get started with FeasyBlue

2.1 General Tools

The development of FeasyBlue is base on Android Studio 3.0.1 and the version of Gradle is 4.1

2.2 FeasyBlue Demo App Project Setup

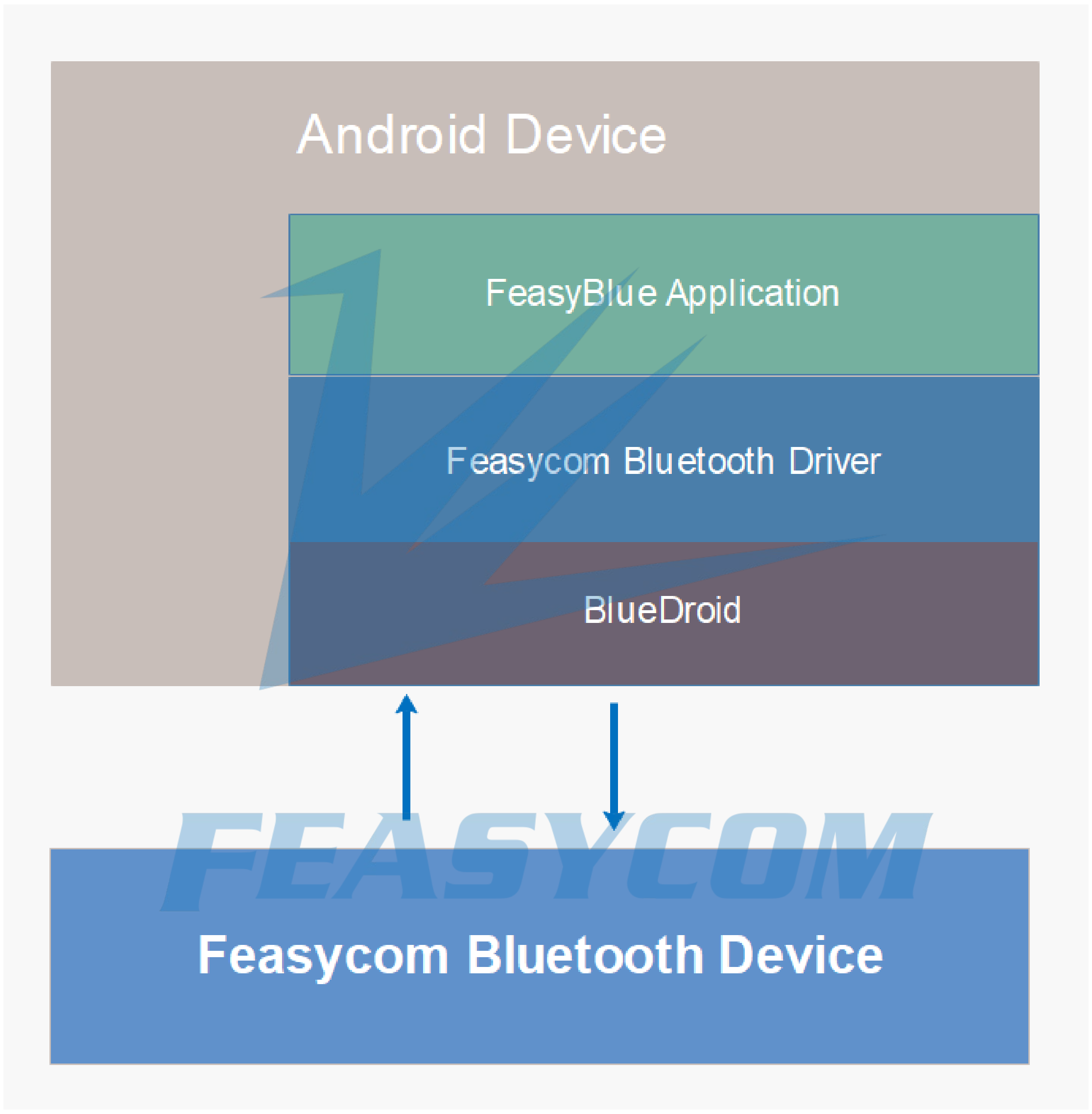
1. Start Android Studio 3.0.1
2. Choose "File->Open"
3. Browse the project folder
4. Open the project

2.3 Download and Run the FeasyBlue Demo App

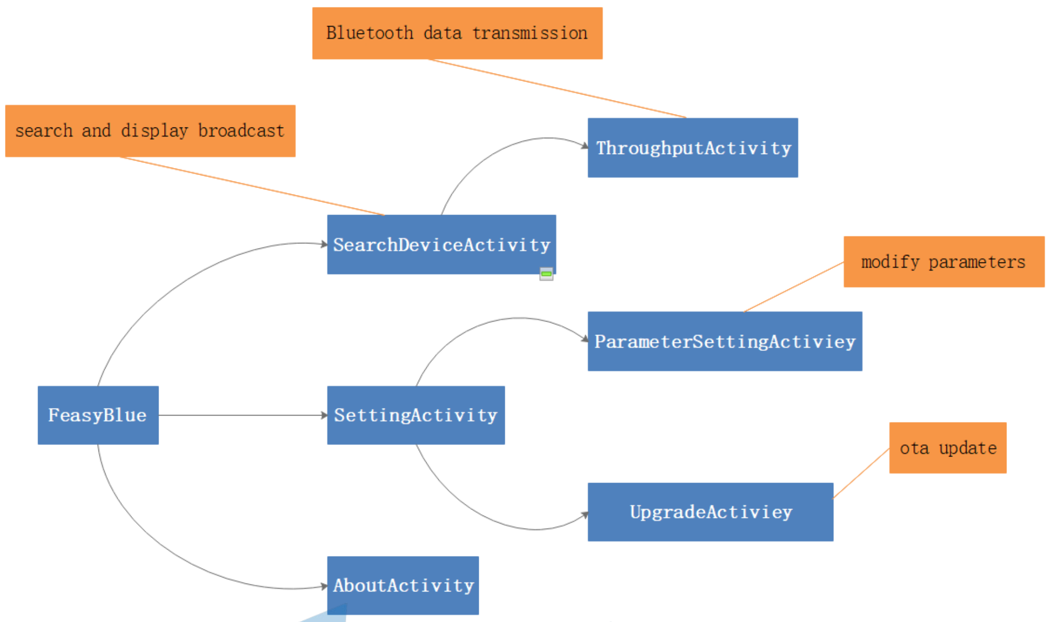
As a first test, we recommend you start with the Communication module. When the FeasyBlue APP started, it runs the Communication module by default, and it will scan the nearby Bluetooth devices automatically. Once there is a Feasycom Bluetooth module display on the device scanning list, you can try to connect it if it is connectable. After FeasyBlue connected to a Feasycom Bluetooth module, FeasyBlue will switch to a transmission page, then you can transferring data from or to Bluetooth module.

3. FeasyBlue Architecture

3.1 FeasyBlue System Architecture



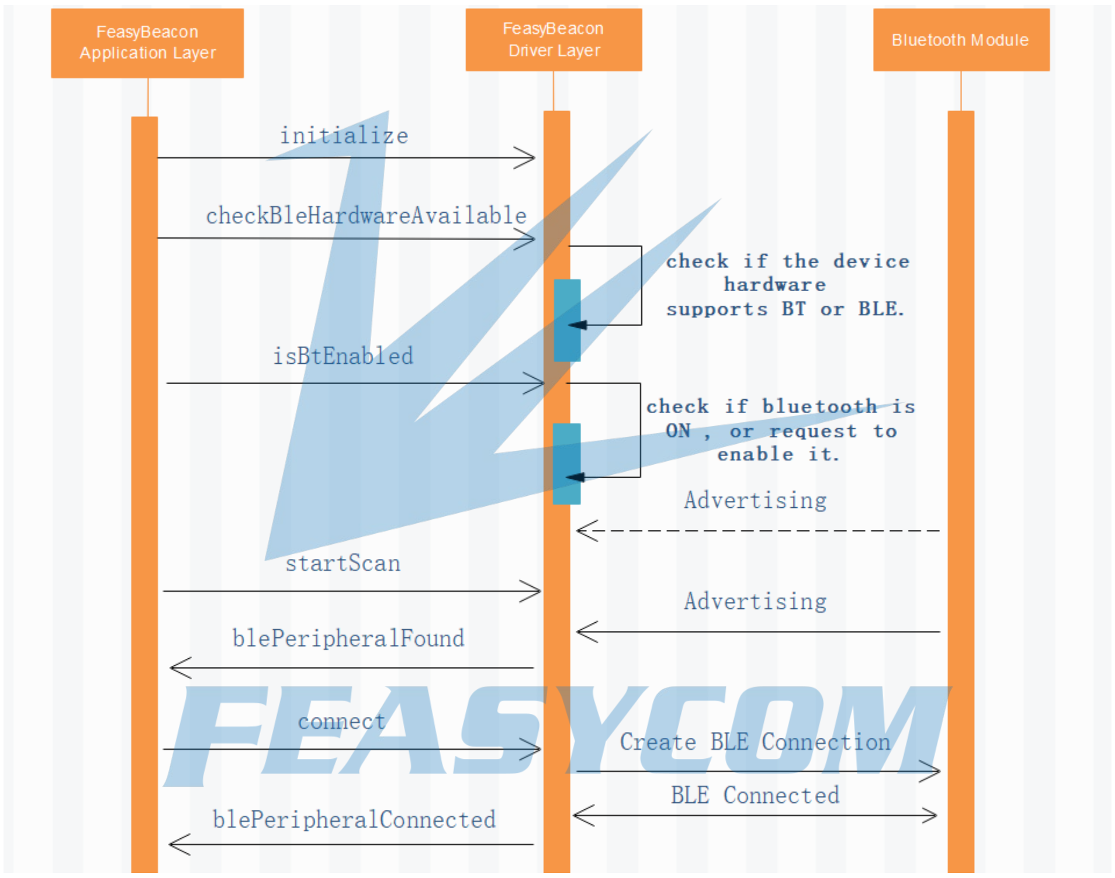
3.2 Activity


Bluetooth data transmission
search and display broadcast
ThroughputActivity
SearchDeviceActivity
modify parameters
ParameterSettingActiviey
FeasyBlue
SettingActivity
ota update
UpgradeActiviey
AboutActivity

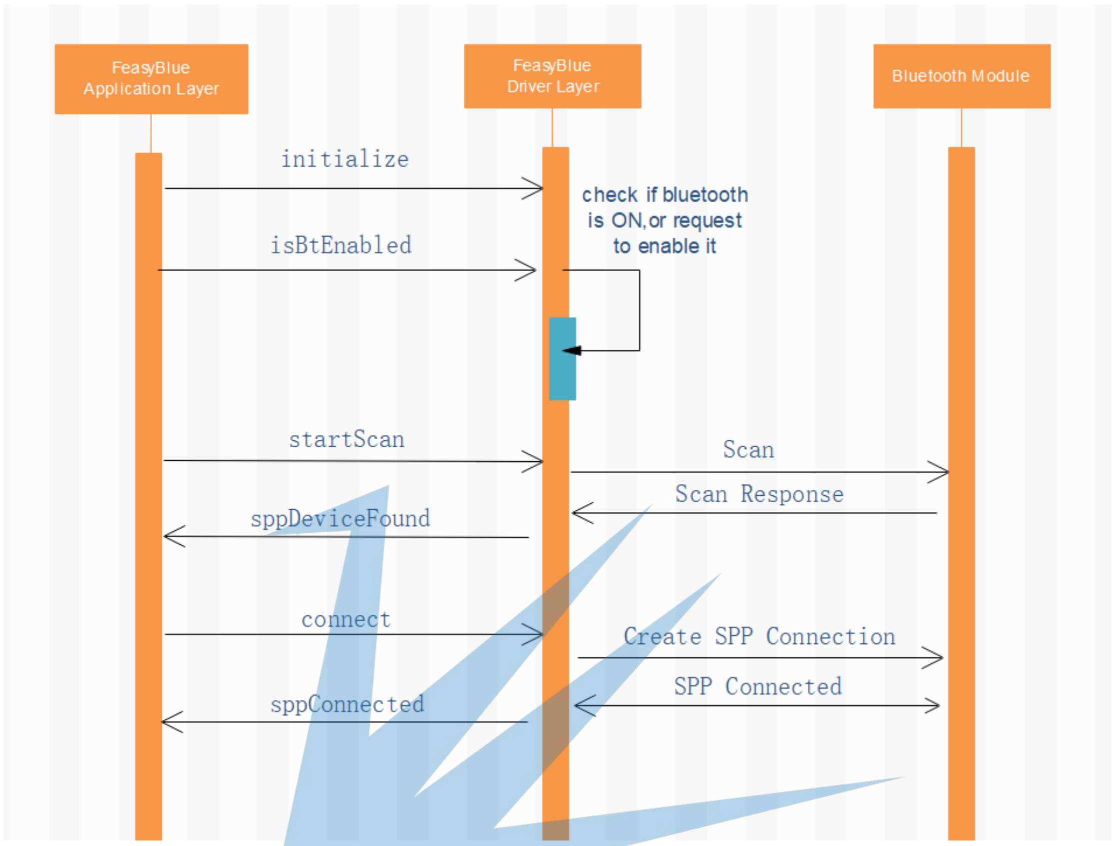
4.Operating Examples

4.1 Typical Initialization and Connection Setup

4.1.1 BLE



4.1.2 SPP


FeasyBlue Application Layer
FeasyBlue Driver Layer
Bluetooth Module
initialize
check if bluetooth is ON,or request to enable it
isBtEnabled
startScan
Scan
Scan Response
sppDeviceFound
connect
Create SPP Connection
SPP Connected
sppConnected

5. General APIs

5.1 CALLBACKS

5.1.1 BLE

```
/*
  * Peripheral found callback,
  * @param device                  The peripheral devce.
  * @param rssi                     The current RSSI of device, in dBm.
  * @param record                  The scan record.
  * @discussion                     Call startScan(),the discovered devices will be returned.
*/
-(void)blePeripheralFound(BluetoothDeviceWrapper device, int rssi, byte[] record)
```

```
/*
  * Peripheral connected callback,
  * @param gatt                     The gatt used by the connection process
  * @param device                  Current connected device.
  * @discussion                     This method is invoked when a connection is set up
  *                                  successfully
  */
-(void)blePeripheralConnected(BluetoothGatt gatt, BluetoothDevice device)
```

```
/*
  * Discover services callback,
  * @param gatt                     The gatt used by the connection process
    * @param device                Current connected device.
  * @param services               The array of services information.
*/
-(void)servicesFound(BluetoothGatt gatt,          BluetoothDevice device,
                                                    ArrayList<BluetoothGattService> service)
```

```
/*
  * Peripheral disconnected callback,
  * @param gatt                     The gatt used by the connection process
  * @param device                  Current connected device.
  * @discussion                     This method is invoked when a disconnect event occurs.
  */
-(void)blePeripheralDisonnected(BluetoothGatt gatt, BluetoothDevice device)
```

5.1.2 SPP

```
/*
  * Peripheral found callback,
  * @param device                    The peripheral devce.
  * @param rssi                       The current RSSI of device, in dBm.
  * @param record                    The scan record.
  * @discussion                      Call startScan(),the discovered devices will be returned.
*/
-(void)sppDeivceFound(BluetoothDeviceWrapper device, int rssi)
```

```
/*
  * Peripheral connected callback,
  * @param device                    Current connected device.
  * @discussion                      This method is invoked when a connection is set up
  *                                   successfully
  */
-(void)sppConnected(BluetoothDevice device)
```

```
/*
  * Peripheral disconnected callback,
  * @param device                    Current connected device.
  * @discussion                      This method is invoked when a disconnect event occurs.
  */
-(void)sppDisonnected(BluetoothDevice device)
```

5.2 METHODS

5.2.1 BLE

```
/*
*@discussion                         Initialization
*/
-(void)initialize()
```

```
/*
*@discussion                         Check if the device has available BT and BLE hardware
*/
boolean checkBleHardwareAvailable()
```

```
/*
  *@discussion                          check bluetooth is ON or not.
  */
```

```
boolean isBtEnabled()
```

```
/*
  *@param    time                    The scan time.
  * @discussion                       Start scan peripherals and stop after “time” ms.
*/
-(void)startScan(int time)
```

```
/*
  * @discussion                       Stop scan peripherals.
  */
-(void)stopScan()
```

```
/*
  * Connect peripheral,
*/
  boolean connect(BluetoothDeviceWrapper device, String pin2Connect);
```

```
/*
  * @discussion                       Disconnect peripheral.
  */
-(void)disconnect()
```

5.2.2 SPP

```
/*
*@discussion                          Initialization
*/
-(void)initialize()
```

```
/*
  *@discussion                          check bluetooth is ON or not.
  */
boolean isBtEnabled()
```

```
/*
  *@param    time                    The scan time.
  * @discussion                       Start scan peripherals and stop after “time” ms.
*/
-(void)startScan(int time)
```

```
/*
  * @discussion                       Stop scan peripherals.
  */
-(void)stopScan()
```

```
/*
  * Connect peripheral,
*/
  boolean connect(String mac);
```

```
/*
  * @discussion                          Disconnect peripheral.
  */
-(void)disconnect()
```

6. Communication APIs

6.1 CALLBACKS

6.1.1 BLE

```
/*
  * Peripheral send packet callback,
  * @param gatt                  The gatt used by the connection process.
  * @param device                The peripheral devce.
  * @param ch                    The gatt characteristic
  * @param percentage            The percentage of total data
  * @param sendByte              The sent data size.
  * @discussion                  one package to be sent the method will be invoked
*/
-(void)sendPacketProcess(BluetoothGatt gatt, BluetoothDevice device,
BluetoothGattCharacteristic ch, int percentage, byte[] sendByte);
```

```
/*
  * Response for characteristic value read,
  * @param gatt                The gatt used by the connection process
  * @param device              Current connected device.
  * @param service             The service of current characteristic
  * @param ch                  The current characteristic
  * @param strValue            received data in String form
  * @param hexString           received data in hex String form
  * @param rawValue            received data
  * @param timestamp           invalid value
  * @discussion                This method is called when data is returned from the
  *                                peripheral.
  */
-(void)readResponse (BluetoothGatt gatt, BluetoothDevice device,BluetoothGattService
service, BluetoothGattCharacteristic ch, String strValue,String hexString,byte[]
rawValue ,String timestamp)
```

6.1.2 SPP

```
/*
  * Peripheral send packet callback,
  * @param device                    The peripheral devce.
  * @param percentage                The percentage of total data
  * @param sendByte                  The sent data size.
  * @discussion                      one package to be sent the method will be invoked
*/
-(void)sendPacketProcess(BluetoothDevice device, int percentate, byte[] sendByte);
```

```
/*
  * Received packet callback,
  * @param dataByte               received data in byte form
  * @param dataString             received data in String form
  * @param hexString              received data in hex String form
  * @discussion                   This method is called when data is returned from the
  *                                    peripheral.
  */
-(void)packetReceived(byte[] dataByte, String dataString, String dataHexString)
```

6.2 METHODS

6.2.1 BLE

```
/*
  *@param    packet                   Data to send.
  * @discussion                      This method is suitable for sending large amounts of data.
*/
-(void)send(byte[] packet)
```

```
/*
  * @discussion                      Stop the current data transmission.
  */
-(void)stopSend()
```

```
/*
  *@param ch                         The gatt characterisc value.
*/
-(void)read(BluetoothGattCharacteristic ch);
```

6.2.2 SPP

/* *@param packet * @discussion */ -(void)send(byte[] packet)	 Data to send. This method is suitable for sending large amounts of data.

/*
  * @discussion
  */
-(void)stopSend()
	
	Stop the current data transmission.

7. Parameter Change APIs

(SPP and BLE shared the same set of Parameter Change APIs)

7.1 CALLBACKS

```
/*
 *@param    command             AT command.
 *@param    param               The parameter of AT command.
 *@param    status              The status of setting or querying a AT command.
*/
-(void)atCommandCallBack(String command , String param, String status)
```

7.2 METHODS

```
/*
 *@param    command             AT command.
 *@discussion                   Send AT command to module.
*/
-(void)sendATCommand(Set<String> command)
```

8. Device Firmware Upgrade APIs

(SPP and BLE shared the same set of Device Firmware Upgrade APIs)

8.1 CALLBACKS

```
/*
  *@param    command               Current OTA data transmission progress.
  *@param    status                The status of OTA.
  *@discussion                     Call startOTA(),the method will be callback.
*/
-(void)otaProgressUpdate(int percentage, int status)
```

8.2 METHODS

```
/*
  *@param    dfuFile                    Current OTA data transmission progress.
  *@param    restoreDefaultSettings     Whether to restore default settings.
  *@discussion                          Call startOTA(),the method will be callback.
*/
-(boolean)startOTA(byte[] dfuFile, boolean restoreDefaultSettings)
```