Secret Level: Public Information

Android BLE API

Instructions

V1.0

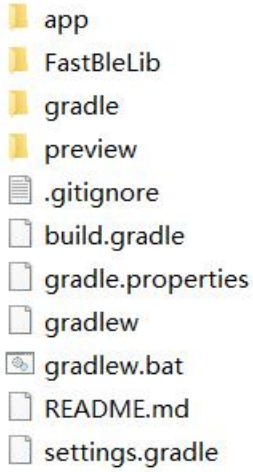
Version History

Version Information Management

Vision	Date	Update Record	Editor
V1.0	2019.01.19	First version	Eric

Content

1. Android BLE SDK introduction



1） ‘app’ : sample code
2） ‘FastBleLib’ : BLE Library file
3） ‘gradle’ :development environment configuration
4） ‘preview’: Sample screenshots

2. Code example

2.1 Summary

Android Bluetooth Low Energy :Bluetooth Rapid Development Framework. Use callback mode to process the Bluetooth operation including:scan, connect, notify, indicate, write, read etc.Each characteristic forms a corresponding notify relationship with a callback.

2.2 Code introduction

Initialization

```
bleManager = new BleManager(this);
```

Determine whether the device supports BLE

```
bleManager.isSupportBle();
```

Turn Bluetooth on or off

```
bleManager.enableBluetooth();
bleManager.disableBluetooth();
```

Scanning out all Bluetooth devices around

Get the surrounding Bluetooth Device object array

```
bleManager.scanDevice(new ListScanCallback(TIME_OUT) {
    @Override
    public void onLeScan(BluetoothDevice device, int rssi, byte[] scanRecord)
{
        super.onLeScan(device, rssi, scanRecord);
        Log.i(TAG, " Find device:  " + device.getName());
    }
    @Override
    public void onDeviceFound(BluetoothDevice[] devices) {
        Log.i(TAG, " Found " + devices.length + " devices ");
        for (int i = 0; i < devices.length; i++) {
            Log.i(TAG, "name:" + devices[i].getName() + "------mac:" + devices
[i].getAddress());
        }
    }
});
```

Connect directly to a device

After searching for the surrounding devices, you can select a device and connect it, and the parameters passed in are the Bluetooth Device object.

```
bleManager.connectDevice(sampleDevice, new BleGattCallback() {
    @Override
    public void onNotFoundDevice() {
        Log.i(TAG, " No device found! ");
    }
    @Override
    public void onFoundDevice(BluetoothDevice device) {
        Log.i(TAG, " Found device : " + device.getAddress());
    }
    @Override
    public void onConnectSuccess(BluetoothGatt gatt, int status) {
        Log.i(TAG, " connect Success!  ");
        gatt.discoverServices();
    }
    @Override
    public void onServicesDiscovered(BluetoothGatt gatt, int status) {
        Log.i(TAG, "Services discovered!  ");
        bleManager.getBluetoothState();
    }
    @Override
    public void onConnectFailure(BleException exception) {
        Log.i(TAG, " Connect failure or interruption:  " + exception.toString());
        bleManager.handleException(exception);
    }
});
```

Scan the device with the specified name and connect it

If you are sure that there is a bluetooth device with a known name around you, or you only need to connect the bluetooth device with the specified name and ignore the name of other devices.

```
bleManager.scanNameAndConnect(
        DEVICE_NAME,
        TIME_OUT,
        new BleGattCallback() {
            @Override
            public void onNotFoundDevice() {
```

```
            Log.i(TAG, " No device found! ");
        }
        @Override
        public void onFoundDevice(BluetoothDevice device) {
            Log.i(TAG, " Found device : " + device.getAddress());
        }
        @Override
        public void onConnectSuccess(BluetoothGatt gatt, int status) {
            Log.i(TAG, " connect Success!  ");
            gatt.discoverServices();
        }
        @Override
        public void onServicesDiscovered(BluetoothGatt gatt, int status) {
            Log.i(TAG, " Services discovered!  ");
            bleManager.getBluetoothState();
        }
        @Override
        public void onConnectFailure(BleException exception) {
            Log.i(TAG, " connect failure or interruption: " + exception.toString());
            bleManager.handleException(exception);
        }
    });
```

Scan the device with the specified MAC address and connect it

If you are sure that there is a bluetooth device with a known MAC around you, or you only need to connect the bluetooth device with the specified MAC and ignore the MAC of other devices.

```
bleManager.scanMacAndConnect(
        DEVICE_MAC,
        TIME_OUT,
        false,
        new BleGattCallback() {
            public void onNotFoundDevice() {
                Log.i(TAG, " No device found!  ");
            }
            @Override
            public void onFoundDevice(BluetoothDevice device) {
                Log.i(TAG, " Found device :" + device.getAddress());
            }
```

```
        @Override
        public void onConnectSuccess(BluetoothGatt gatt, int status) {
            Log.i(TAG, " connect Success!  ");
            gatt.discoverServices();
        }
        @Override
        public void onServicesDiscovered(BluetoothGatt gatt, int status) {
            Log.i(TAG, " Services discovered!  ");
            bleManager.getBluetoothState();
        }
        @Override
        public void onConnectFailure(BleException exception) {
            Log.i(TAG, " Connect failure or interruption: " + exception.toString());
            bleManager.handleException(exception);
        }
    });
```

notify ， listen data changes through callback

The callback and UUID in the parameter will form an association. Once the character of the UUID of the device change, the 'callback' will call back the result. This callbak will only exist, and is a one-to-one relationship with uuid.

```
bleManager.notify(
        UUID_SERVICE,
        UUID_NOTIFY,
        new BleCharacterCallback() {
            @Override
            public void onSuccess(BluetoothGattCharacteristic characteristic)
{
                Log.d(TAG, "notify result :  "
                        + String.valueOf(HexUtil.encodeHex(characteristic.getV
alue())));
            }
            @Override
            public void onFailure(BleException exception) {
                bleManager.handleException(exception);
            }
        });
```

stop notify ， remove callback

```
bleManager.stopNotify(UUID_SERVICE, UUID_NOTIFY);
```

indicate ， listen data changes through callback

```
bleManager.indicate(
        UUID_SERVICE,
        UUID_INDICATE,
        new BleCharacterCallback() {
            @Override
            public void onSuccess(BluetoothGattCharacteristic characteristic)
{
                Log.d(TAG, "indicate result :  "
                        + String.valueOf(HexUtil.encodeHex(characteristic.getV
alue())));
            }
            @Override
            public void onFailure(BleException exception) {
                Log.e(TAG, "indicate: " + exception.toString());
                bleManager.handleException(exception);
            }
        });
```

stop indicate ， remove callback

```
bleManager.stopIndicate(UUID_SERVICE, UUID_INDICATE);
```

write

```
bleManager.writeDevice(
        UUID_SERVICE,
        UUID_WRITE,
        HexUtil.hexStringToBytes(SAMPLE_WRITE_DATA),
        new BleCharacterCallback() {
            @Override
            public void onSuccess(BluetoothGattCharacteristic characteristic)
{
                Log.d(TAG, "write result: "
                        + String.valueOf(HexUtil.encodeHex(characteristic.getV
alue())));
```

```
            }
            @Override
            public void onFailure(BleException exception) {
                Log.e(TAG, "write: " + exception.toString());
                bleManager.handleException(exception);
            }
        });
```

read

```
bleManager.readDevice(
        UUID_SERVICE,
        UUID_WRITE,
        new BleCharacterCallback() {
            @Override
            public void onSuccess(BluetoothGattCharacteristic characteristic)
{
                Log.d(TAG, "read result: "
                        + String.valueOf(HexUtil.encodeHex(characteristic.getV
alue())));
            }
            @Override
            public void onFailure(BleException exception) {
                Log.e(TAG, "read: " + exception.toString());
                bleManager.handleException(exception);
            }
        });
```

manual remove callback

When the UUID as a parameter no longer notifies the data changes of the character corresponding to the uuid, it can be used to remove the callback corresponding to notify, indicate, write and read.

```
bleManager.stopListenCharacterCallback(UUID_NOTIFY);
```

Reset (disconnect this Bluetooth connection and remove all callbacks)

```
bleManager.closeBluetoothGatt();
```

Other

Other bluetooth operation can refer to the example code or find a method from the open class BleManager.