**1.** Add appropriate permissions in AndroidManifest.xml and Android SDK must be 18 above.

<uses-permission android:name=*"android.permission.BLUETOOTH"*/>

<uses-permission android:name=*"android.permission.BLUETOOTH\_ADMIN"*/>

<uses-feature android:name=*"android.hardware.bluetooth\_le"* android:required=*"true"*/>

**1.1.**Import MyTagSDK.jar into the project's LIBS, then right click the mouse, select the Build path-----add to Build path

**2.** When scanning device, please use BluetoothUtil class

**2.1.** To get BluetoothUtil object, please use mBluetoothUtil=BluetoothUtil.getInstance(Activity activity)

**2.2.** Please check if the Android device support Bluetooth 4.0

##### if (!mBluetoothUtil.isSupportBLE()) {

finish();

return;

}

**2.3.** Turn on Bluetooth on Android device, please use mBluetoothUtil.openBluetooth().

**2.4.** Please use scanLeDevices(final boolean enable, int scanTime,final BluetoothAdapter.LeScanCallback mLeScanCallback) of BluetoothUtil class to scan device, get or stop scanning device. When “enable” is “true”, it’s scanning mode; when “enable” is “false”, it’s stop scanning mode. ScanTime is scanning time; ScanTime can be null when enable is false; mLeScanCallback is interface callback. In this callback, we can scan and get the Ble device again.

BluetoothAdapter.LeScanCallback mLeScanCallback = new BluetoothAdapter.LeScanCallback() {

@Override

public void onLeScan(final BluetoothDevice device, int rssi,byte[] scanRecord) {

runOnUiThread(new Runnable() {

@Override

public void run() {

mLeDeviceListAdapter.addDevice(device);

mLeDeviceListAdapter.notifyDataSetChanged();

}

});

}

};

**3.** To pair bluetooth device, please use BluetoothService class

**3.1.** Example of getting BluetoothService, mBluetoothLeService=new BluetoothService(Activity activity);

**3.2.** Please use mBluetoothLeService.connect(address) to pair with bluetooth device. Please register BroadcastReceiver and receive information to examine if successful connect.

##### 3.3. Please register BroadcastReceiver, registerReceiver (mGattUpdateReceiver, BluetoothService.*makeGattUpdateIntentFilter*()); second parameter IntenFilter filter can be obtained through BluetoothService.makeGattUpdateIntentFilter()

**3.4.** OnReceive method in BroadcastReceiver, action may be get as following. No receive if action is useless:

**1.** BluetoothService.ACTION\_GATT\_CONNECTED (It will receive this broadcast after Ble device successful connect)

**2.** BluetoothService.ACTION\_GATT\_DISCONNECTED(It will receive this broadcast when Ble device failed connect or disconnect)

**3.** BluetoothService.ACTION\_GATT\_SERVICES\_DISCOVERED(It will receive this broadcast when bluetooth services is found. Rarely use)

**4.** BluetoothService.ACTION\_DATA\_ELECTRICITY(Receiving this broadcast, we can get electric quantity information of bluetooth device by intent.getIntExtra(BluetoothService.*EXTRA\_DATA*, 0) . Return value is int class. )

**5.** BluetoothService.ACTION\_DATA\_TEMPERATURE(Receiving this broadcast, we can get temperature information of bluetooth device by intent.getDoubleExtra(BluetoothService.*EXTRA\_DATA*, 0). Return value is double class. )

**6.** BluetoothService.ACTION\_DATA\_HUMIDITY(Receiving this broadcast, we can get humidity information of bluetooth device by intent.getDoubleExtra(

BluetoothService.*EXTRA\_DATA*, 0). Return value is double class. )

**7.** BluetoothService.ACTION\_DATA\_CLICK (Receiving this broadcast, indicates that bluetooth device perform single click button operation. Generally, this broadcast is to alert App for first level alarm or other operation. )

**8.** BluetoothService.ACTION\_DATA\_DOUBLECLICK(Receiving this broadcast, indicates that bluetooth device perform double click button operation)

**9.** BluetoothService.ACTION\_DATA\_LONG\_PRESS(Receiving this broadcast, indicates that bluetooth device perform long press button operation)

**10.** BluetoothService.ACTION\_DATA\_SECONDARY\_ALARM(Generally, this broadcast is to alert App for second level alarm, or other operation. Rarely use )

**3.5.** When receiving broadcast, we can get bluetooth device address by intent.getStringExtra(BluetoothService.DEVICEADDRESS) and know which bluetooth device is communicating.

**3.6.** After mBluetoothLeService.connect(address) programming, broadcast of BluetoothService.ACTION\_GATT\_CONNECTED will be received if successful connect. Then various kinds of operation can be performed.

**3.7.** After connection, Android App can write date to the bluetooth terminal. mBluetoothLeService.alarmBLE() is bluetooth terminal sound&light alarm, used for bluetooth terminal searching; mBluetoothLeService.stopAlarm() is bluetooth terminal mute setting, used for stop sound & light alarm of bluetooth terminal; mBluetoothLeService.close() is used for remote shutdown bluetooth terminal.

**3.8.** Attention: If keep fail connect to the bluetooth terminal during connection, please use mBluetoothLeService.disconnect() and mBluetoothLeService .close() to release resources when disconnect. Otherwise, it will block the connect of other devices.

**3.9.** If you do not understand, the specific details of the use of a reference to our code sample.