客户端蓝牙主控制服务：BluetoothClientService类：

public class BluetoothClientService extends Service {

//搜索到的远程设备集合

private List<BluetoothDevice> discoveredDevices = new ArrayList<BluetoothDevice>();

//蓝牙适配器

private final BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

//通讯线程

private BluetoothCommunThread communThread;

//控制信息广播的接收器

private BroadcastReceiver controlReceiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

String action = intent.getAction();

if (BluetoothTools.ACTION\_START\_DISCOVERY.equals(action)) {

//开始搜索

discoveredDevices.clear(); //清空存放设备的集合

bluetoothAdapter.enable(); //打开蓝牙

bluetoothAdapter.startDiscovery(); //开始搜索

} else if (BluetoothTools.ACTION\_SELECTED\_DEVICE.equals(action)) {

//选择了连接的服务器设备

BluetoothDevice device = (BluetoothDevice)intent.getExtras().get(BluetoothTools.DEVICE);

//开启设备连接线程

new BluetoothClientConnThread(handler, device).start();

} else if (BluetoothTools.ACTION\_STOP\_SERVICE.equals(action)) {

//停止后台服务

if (communThread != null) {

communThread.isRun = false;

}

stopSelf();

} else if (BluetoothTools.ACTION\_DATA\_TO\_SERVICE.equals(action)) {

//获取数据

Object data = intent.getSerializableExtra(BluetoothTools.DATA);

if (communThread != null) {

communThread.writeObject(data);

}

}

}

};

//蓝牙搜索广播的接收器

private BroadcastReceiver discoveryReceiver = new BroadcastReceiver() {

@Override

public void onReceive(Context context, Intent intent) {

//获取广播的Action

String action = intent.getAction();

if (BluetoothAdapter.ACTION\_DISCOVERY\_STARTED.equals(action)) {

//开始搜索

} else if (BluetoothDevice.ACTION\_FOUND.equals(action)) {

//发现远程蓝牙设备

//获取设备

BluetoothDevice bluetoothDevice = intent.getParcelableExtra(BluetoothDevice.EXTRA\_DEVICE);

discoveredDevices.add(bluetoothDevice);

//发送发现设备广播

Intent deviceListIntent = new Intent(BluetoothTools.ACTION\_FOUND\_DEVICE);

deviceListIntent.putExtra(BluetoothTools.DEVICE, bluetoothDevice);

sendBroadcast(deviceListIntent);

} else if (BluetoothAdapter.ACTION\_DISCOVERY\_FINISHED.equals(action)) {

//搜索结束

if (discoveredDevices.isEmpty()) {

//若未找到设备，则发动未发现设备广播

Intent foundIntent = new Intent(BluetoothTools.ACTION\_NOT\_FOUND\_SERVER);

sendBroadcast(foundIntent);

}

}

}

};

//接收其他线程消息的Handler

Handler handler = new Handler() {

@Override

public void handleMessage(Message msg) {

//处理消息

switch (msg.what) {

case BluetoothTools.MESSAGE\_CONNECT\_ERROR:

//连接错误

//发送连接错误广播

Intent errorIntent = new Intent(BluetoothTools.ACTION\_CONNECT\_ERROR);

sendBroadcast(errorIntent);

break;

case BluetoothTools.MESSAGE\_CONNECT\_SUCCESS:

//连接成功

//开启通讯线程

communThread = new BluetoothCommunThread(handler, (BluetoothSocket)msg.obj);

communThread.start();

//发送连接成功广播

Intent succIntent = new Intent(BluetoothTools.ACTION\_CONNECT\_SUCCESS);

sendBroadcast(succIntent);

break;

case BluetoothTools.MESSAGE\_READ\_OBJECT:

//读取到对象

//发送数据广播（包含数据对象）

Intent dataIntent = new Intent(BluetoothTools.ACTION\_DATA\_TO\_GAME);

dataIntent.putExtra(BluetoothTools.DATA, (Serializable)msg.obj);

sendBroadcast(dataIntent);

break;

}

super.handleMessage(msg);

}

};

客户端配对的链线程：BluetoothClientConnThead类：

public class BluetoothClientConnThread extends Thread{

private Handler serviceHandler; //用于向客户端Service回传消息的handler

private BluetoothDevice serverDevice; //服务器设备

private BluetoothSocket socket; //通信Socket

/\*\*

\* 构造函数

\* @param handler

\* @param serverDevice

\*/

public BluetoothClientConnThread(Handler handler, BluetoothDevice serverDevice) {

this.serviceHandler = handler;

this.serverDevice = serverDevice;

}

@Override

public void run() {

BluetoothAdapter.getDefaultAdapter().cancelDiscovery();

try {

socket = serverDevice.createRfcommSocketToServiceRecord(BluetoothTools.PRIVATE\_UUID);

BluetoothAdapter.getDefaultAdapter().cancelDiscovery();

socket.connect();

} catch (Exception ex) {

try {

socket.close();

} catch (IOException e) {

e.printStackTrace();

}

//发送连接失败消息

serviceHandler.obtainMessage(BluetoothTools.MESSAGE\_CONNECT\_ERROR).sendToTarget();

return;

}

//发送连接成功消息，消息的obj参数为连接的socket

Message msg = serviceHandler.obtainMessage();

msg.what = BluetoothTools.MESSAGE\_CONNECT\_SUCCESS;

msg.obj = socket;

msg.sendToTarget();

}

}

蓝牙数据传输：BluetoothCommunThread类：

**public** **class** BluetoothCommunThread **extends** Thread {

**private** Handler serviceHandler; //与Service通信的Handler

**private** BluetoothSocket socket;

**private** ObjectInputStream inStream; //对象输入流

**private** ObjectOutputStream outStream; //对象输出流

**public** **volatile** **boolean** isRun = **true**; //运行标志位

/\*\*

\* 构造函数

\* **@param** handler 接收消息

\*

\* **@param** socket

\*/

**public** BluetoothCommunThread(Handler handler, BluetoothSocket socket) {

**this**.serviceHandler = handler;

**this**.socket = socket;

**try** {

**this**.outStream = **new** ObjectOutputStream(socket.getOutputStream());

**this**.inStream = **new** ObjectInputStream(**new** BufferedInputStream(socket.getInputStream()));

} **catch** (Exception e) {

**try** {

socket.close();

} **catch** (IOException e1) {

e1.printStackTrace();

}

//发送连接失败消息

serviceHandler.obtainMessage(BluetoothTools.MESSAGE\_CONNECT\_ERROR).sendToTarget();

e.printStackTrace();

}

}

@Override

**public** **void** run() {

**while** (**true**) {

**if** (!isRun) {

**break**;

}

**try** {

Object obj = inStream.readObject();

//发送成功读取到对象的消息，消息的obj参数为读取到的对象

Message msg = serviceHandler.obtainMessage();

msg.what = BluetoothTools.MESSAGE\_READ\_OBJECT;

msg.obj = obj;

msg.sendToTarget();

} **catch** (Exception ex) {

//发送连接失败消息

serviceHandler.obtainMessage(BluetoothTools.MESSAGE\_CONNECT\_ERROR).sendToTarget();

ex.printStackTrace();

**return**;

}

}

//关闭流

**if** (inStream != **null**) {

**try** {

inStream.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**if** (outStream != **null**) {

**try** {

outStream.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**if** (socket != **null**) {

**try** {

socket.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}