第二十二章

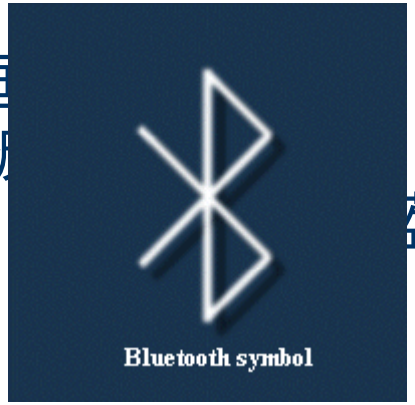
Android蓝牙

目标

- 什么是蓝牙
- Android对蓝牙的支持

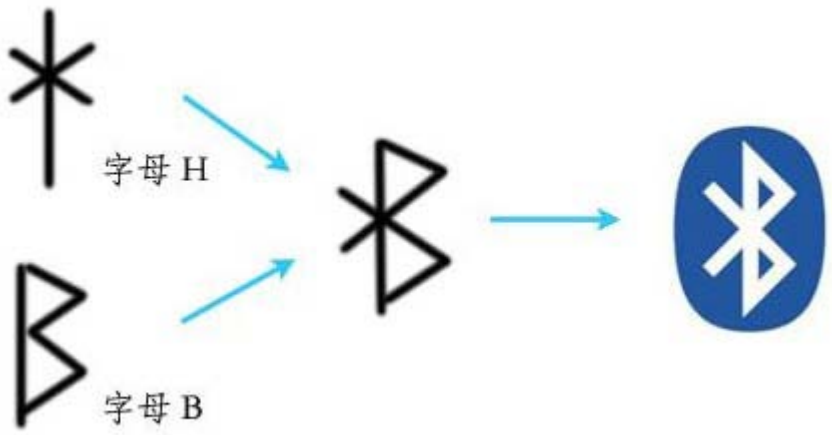
什么是蓝牙

- 蓝牙，是一种支持设备短距离通信（一般10m内）的无线电技术。能在包括移动电话、PDA、无线耳机、笔记本电脑、相关外设等众多设备之间进行无线信息交换。利用“蓝牙”技术，能够有效地简化移动通信终端设备之间的通信，也能够成功地简化设备与因特网Internet之间的通信，从而数据传输变得更加迅速高效，为无线通信拓宽道路。蓝牙采用分散式网络结构以及快跳频和短包技术，支持点对点及点对多点通信，工作在全球通用的2.4GHz ISM（即工业、科学、医学）频段。其数据速率为1Mbps。采用时分双工传输方案实现全双工传输。
- 蓝牙这个名称来自于第十世纪的一位丹麦国
Blatand , Blatand 在英文里的意思可以被
Bluetooth(蓝牙)因为国王喜欢吃蓝梅，
色的所以叫蓝牙



什么是蓝牙

- 蓝牙的创始人是瑞典爱立信公司，爱立信早在1994年就已进行研发。1997年，爱立信与其他设备生产商联系，并激发了他们对该项技术的浓厚兴趣。1998年2月，5个跨国大公司，包括爱立信、诺基亚、IBM、东芝及Intel组成了一个特殊兴趣小组（SIG），他们共同的目标是建立一个全球性的小范围无线通信技术，即现在的蓝牙。
- 蓝牙这个标志的设计：它取自 Harald Bluetooth 名字中的「H」和「B」两 __蓝牙标志的来历个字母，用古北欧字母来表示，将这两者结合起来，就成为了蓝牙的 logo



蓝牙技术优势

- 全球可用
  - Bluetooth 无线技术规格供我们全球的成员公司免费使用。许多行业的制造商都积极地在其产品中实施此技术，以减少使用零乱的电线，实现无缝连接、流传输立体声，传输数据或进行语音通信。Bluetooth 技术在 2.4 GHz 波段运行，该波段是一种无需申请许可证的工业、科技、医学 (ISM) 无线电波段。正因如此，使用 Bluetooth 技术不需要支付任何费用。但您必须向手机提供商注册使用 GSM 或 CDMA，除了设备费用外，您不需要为使用 Bluetooth 技术再支付任何费用。
- 设备范围
  - Bluetooth 技术得到了空前广泛的应用，集成该技术的产品从手机、汽车到医疗设备，使用该技术的用户从消费者、工业市场到企业等等，不一而足。低功耗，小体积以及低成本的芯片解决方案使得 Bluetooth 技术甚至可以应用于极微小的设备中。请在 Bluetooth 产品目录和组件产品列表中查看我们的成员提供的各类产品大全。

蓝牙技术优势

- 易于使用
  - Bluetooth 技术是一项即时技术，它不要求固定的基础设施，且易于安装和设置。您不需要电缆即可实现连接。新用户使用亦不费力 – 您只需拥有 Bluetooth 品牌产品，检查可用的配置文件，将其连接至使用同一配置文件的另一 Bluetooth 设备即可。后续的 PIN 码流程就如同您在 ATM 机器上操作一样简单。外出时，您可以随身带上您的个人局域网 (PAN)，甚至可以与其它网络连接。
- 全球通用的规格
  - Bluetooth 无线技术是当今市场上支持范围最广泛，功能最丰富且安全的无线标准。全球范围内的资格认证程序可以测试成员的产品是否符合标准。自 1999 年发布 Bluetooth 规格以来，总共有超过 4000 家公司成为 Bluetooth 特别兴趣小组 (SIG) 的成员。同时，市场上 Bluetooth 产品的数量也成倍的迅速增长。产品数量已连续四年成倍增长，安装的基站数量在 2005 年底也可能达到 5 亿个。

Android对蓝牙的支持

- Android包含了对蓝牙网络协议栈的支持，这使得蓝牙设备能够无线连接其他蓝牙设备交换数据。Android的应用程序框架提供了访问蓝牙功能的APIs。这些APIs让应用程序能够无线连接其他蓝牙设备，实现点对点，或点对多点的无线交互功能。
- 使用蓝牙APIs，一个Android应用程序能够实现下列功能：
  - 扫描其他蓝牙设备；
  - 查询本地蓝牙适配器(local Bluetooth adapter)用于配对蓝牙设备；
  - 建立RFCOMM信道(channels)；
  - 通过服务发现(service discovery)连接其他设备；
  - 数据通信；
  - 管理多个连接

蓝牙开发

- 要进行蓝牙通信需要完成下面四个步骤：
  - 设置蓝牙；
  - 发现已经配对或者可用的附近的蓝牙设备；
  - 连接设备；
  - 在不同设备之间传输数据；

相关类

- 所有可用的Bluetooth APIs都包含在android.bluetooth包里。下面是创建蓝牙连接的类的总览 ：

相关类

- BluetoothAdapter
  - 代表本地的蓝牙适配器(local Bluetooth adapter) (Bluetooth radio). BluetoothAdapter是所有蓝牙通信的入口点。使用BluetoothAdapter，你能够探测其他蓝牙设备，获得一个bonded(已配对)的设备列表，使用一个知名的(know)MAC地址实例化一个BluetoothDevice，并创建一个BluetoothServerSocket来监听其他设备的通信。
- BluetoothDevice
  - 代表一个远程蓝牙设备，使用BluetoothSocket对另一个远程设备发出连接请求，或者查询该远程设备的名字、地址、类和连接状态。
- BluetoothSocket
  - 代表一个蓝牙socket的接口(类似于TCP socket)。这是应用程序通过InputStream或者OutputStream与其他蓝牙设备交换数据的连接点。
- BluetoothServerSocket
  - 表示一个开放的服务器socket，监听进入的连接请求(类似于TCP的ServerSocket)。为了连接两个Android设备，其中一个必须打开一个server socket。当一个远程蓝牙设备发出一个连接请求并被接受时，BluetoothServerSocket将返回一个已连接的BluetoothSocket。
- BluetoothClass
  - 描述了一个蓝牙设备的普通特性和能力，它提供了一系列描述一个设备的主要和次要设备类别和服务的只读属性。尽管这个类并不总是可靠地描述一个设备所有的蓝牙profile(配置)和所支持的服务，但它作为设备类型的一个提示是不错的。

android蓝牙开发—权限

- 使用蓝牙功能，我们至少需要声明两方面的权限：BLUETOOTH和BLUETOOTH_ADMIN。
  - BLUETOOTH权限实现蓝牙通信，例如请求一个连接、接受一个连接和传输数据。
  - BLUETOOTH_ADMIN权限，初始化device discovery或者管理蓝牙设置(Bluetooth settings)。大多数应用程序必须具有这个权限才能够发现本地蓝牙设备，这个权限保护的其他能力(除了发现本地设备)不应该被使用，除非你的应用程序是在用户请求的时候能够修改蓝牙设置的管理者。
  - 注意：如果你想要使用BLUETOOTH_ADMIN权限，那么你首先必须有BLUETOOTH权限。
- 在manifest文件中声明程序的蓝牙权限。例如：

```
view plaincopy to clipboardprint?
<manifest ... >
  <uses-permission android:name="android.permission.BLUETOOTH" />
  ...
</manifest>
```

android蓝牙开发—设置蓝牙

- 在你的应用程序使用蓝牙进行通信之前，你需要确认你的设备支持蓝牙，如果支持，那么确认它已被启动。
- 如果你的设备不支持蓝牙，那么你应该关闭任何蓝牙特性。如果蓝牙被支持，那么你可以在你的程序中要求用户启动蓝牙。这需要两个步骤，并且要使用BluetoothAdapter这个类。

android蓝牙开发—设置蓝牙

- 如果你的设备不支持蓝牙，那么要关闭任何蓝牙功能。如果支持蓝牙但没有启动，则你可以在程序中要求用户启动蓝牙。启动蓝牙需要两个步骤，并且需要BluetoothAdapter类。
- 获得BluetoothAdapter类
- 任何蓝牙activity都需要BluetoothAdapter类。使用静态方法getDefaultAdapter()获得一个BluetoothAdapter的实例，这代表了设备本身的蓝牙适配器(the Bluetooth radio)。整个系统只有一个蓝牙适配器，你的程序可以通过获取到BluetoothAdapter实例与之交互。如果getDefaultAdapter()方法返回null则说明你的设备不支持蓝牙。
- 示例代码如下：
- ```
  BluetoothAdapter mBluetoothAdapter =
  BluetoothAdapter.getDefaultAdapter();
  if (mBluetoothAdapter == null) {
      // Device does not support Bluetooth
  }
  ```
  ```

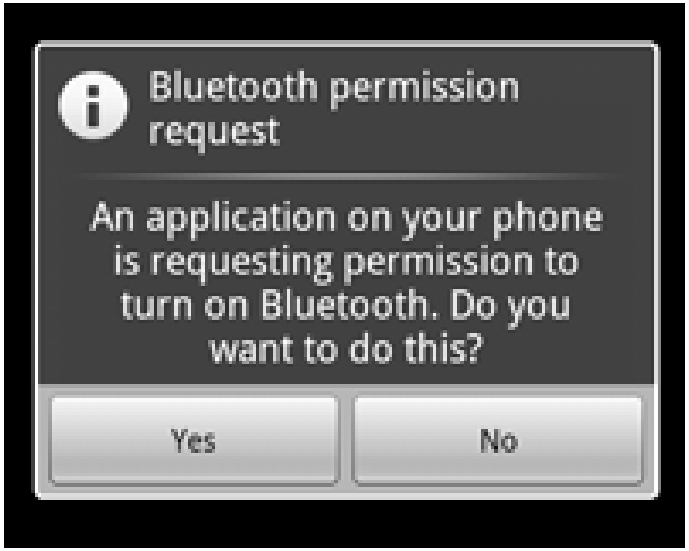
启动蓝牙

- 确保用户启动了蓝牙。调用isEnabled()方法来检查当前蓝牙是否启动。如果该方法返回false，那么说明蓝牙没有启动。这时需要使用“ACTION_REQUEST_ENABLE” action Intent作为参数，调用startActivityForResult()方法来请求启动蓝牙。这将通过系统设备来发出启动蓝牙的请求(不会停止你的程序)。例如：

```
if (!mBluetoothAdapter.isEnabled()) {
    Intent enableBtIntent = new
      Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
      startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
if (!mBluetoothAdapter.isEnabled()) {
     Intent enableBtIntent = new
      Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
      startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
}
```

启动蓝牙

- 执行如上的代码将会弹出一个对话框，请求启动蓝牙的用户权限，如Figure 1所示。如果用户点击“Yes”按钮，那么系统将开始启动蓝牙，启动蓝牙(有可能失败)之后你的程序将重新获得焦点。
- 如果启动蓝牙成功，你的Activity将在onActivityResult()回调函数中接收到一个RESULT_OK结果码。如果蓝牙启动失败或者用户不允许启动蓝牙，则会收到RESULT_CANCELED。



查询配对设备

- 在执行device discovery之前，最好在已配对的设备列表中查看所要发现的设备是否已经存在。通过调用getBondedDevices()函数可以获得代表已经配对的设备的BluetoothDevice集合。 例如，你可以查询所有已经配对的设备，然后通过一个ArrayAdapter添加和显示每个设备的名字给用户:

```
Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
// If there are paired devices
if (pairedDevices.size() > 0) {
 // Loop through paired devices
for (BluetoothDevice device : pairedDevices) {
// Add the name and address to an array adapter to show in a ListView
mArrayAdapter.add(device.getName() + "\n" + device.getAddress());    }}
Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();
// If there are paired devices
if (pairedDevices.size() > 0) {
 // Loop through paired devices
for (BluetoothDevice device : pairedDevices) {
// Add the name and address to an array adapter to show in a ListView
mArrayAdapter.add(device.getName() + "\n" + device.getAddress());
 }
}
```

- 为了建立一个连接，需要才能够BluetoothDevice对象中获取的是MAC地址。在这个例子中，MAC地址作为显示给用户的ArrayAdapter的一部分存储。只要有需要，可以把MAC地址提取出来。

发现设备

- 调用startDiscovery()开始设备发现的过程，这个过程是异步的，startDiscovery()方法会立即返回一个boolean的值表示启动是否成功。这个发现过程通常包括大约12秒的查询扫描，之后是在发现的设备中查询其蓝牙名称。
- 你的应用程序中必须注册一个ACTION_FOUND Intent的BroadcastReceiver，用于接收发现一个蓝牙设备时发出的信息。对于每一个设备，系统将广播ACTION_FOUND的Intent。这个Intent包含了一些附加数据域——EXTRA_DEVICE和EXTRA_CLASS，分别包含BluetoothDevice类和BluetoothClass类的实例。

- 下：

发现设备

- 下面代码展示了如何注册设备发现时的广播处理函数

```
// Create a BroadcastReceiver for ACTION_FOUND
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {    public
   void onReceive(Context context, Intent intent)
{
   String action = intent.getAction();
// When discovery finds a device
   if (BluetoothDevice.ACTION_FOUND.equals(action)) {
// Get the BluetoothDevice object from the Intent
   BluetoothDevice device =
   intent.getParcelableExtra(    BluetoothDevice.EXTRA_DEVICE);
// Add the name and address to an array adapter to show in a ListView
   mArrayAdapter.add(device.getName() + "\n" + device.getAddress());
        }
    }};
// Register the BroadcastReceiverIntentFilter
 filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter);
// Don't forget to unregister during onDestroy
```

- 为了初始化一个连接，我们需要从BluetoothDevice对象中获取MAC地址。
- 注意：执行设备发现这个过程，需要花费蓝牙适配器大量资源，是一个重量级过程。如果你发现一个设备并要连接它，最好先调用cancelDiscovery()方法来停止设备发现过程。如果你已经有一个连接，那么执行设备发现过程或导致连接的带宽大幅度减少，所以当你已经有连接的时候最好就不要执行设备发现过程了。

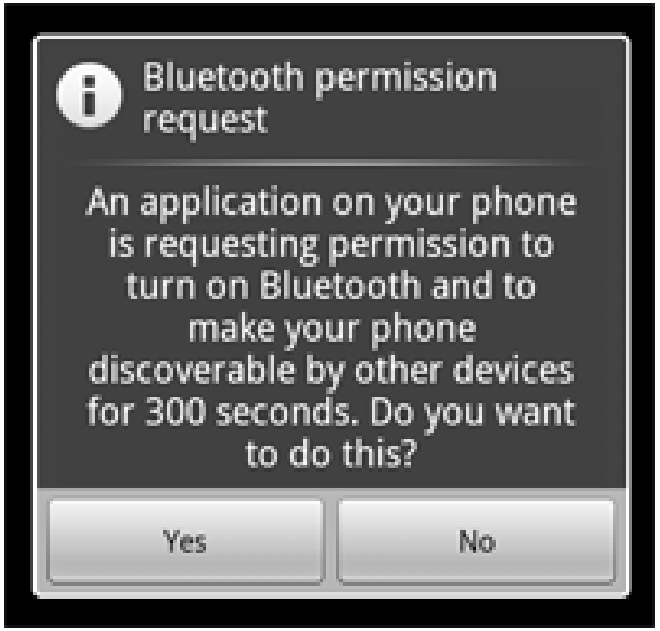
启动发现功能

- 想要你的设备能被其他设备发现，调用startActivityForResult(Intent,int),传递一个ACTION_REQUEST_DISCOVERABLE action Intent给它。这将发送一个请求给系统设置以启动可被发现模式。可被发现模式一般默认持续120秒，你可以通过给Intent添加一个EXTRA_DISCOVERABLE_DURATION Intent extra来更改可被发现模式的持续时间，这个时间最大是300秒。
- 示例：

```
view plaincopy to clipboardprint?
Intent discoverableIntent =
   newIntent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);
discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DU
   RATION, 300);
startActivity(discoverableIntent);
```

启动发现功能

- 一个对话框将会出现，请求用户权限来启动设备的可被发现模式，如Figure 2所示。如果用户点击“Yes”，那么设备在设定的时间内将是可被发现的。你的Activity将调用onActivityResult()回调函数。如果用户点击“No”，那么将产生一个错误，结果码将是Activity.RESULT_CANCELLED。



android蓝牙开发——连接设备

- 为了在两台设备上创建一个连接，必须在软件上实现服务器端和客户端的机制，因为一个设备必须必须打开一个server socket，而另一个必须初始化这个连接(使用服务器端设备的MAC地址进行初始化)。

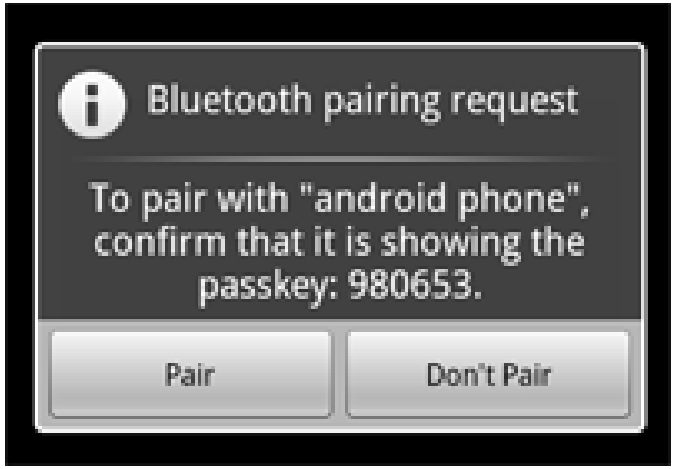
- 当服务器端和客户端在同一个RFCOMM信道上都有一个BluetoothSocket时，就可以认为它们之间建立了一个连接。在这个时刻，每个设备能获得一个输出流和一个输入流，也能够开始数据传输。本节介绍如何在两个设备之间初始化一个连接。

- 服务器端和客户端获得BluetoothSocket的方法是不同的，服务器端是当一个进入的连接被接受时才产生一个BluetoothSocket，客户端是在打开一个到服务器端的RFCOMM信道时获得BluetoothSocket的。

android蓝牙开发——连接设备

- 一种实现技术是，每一个设备都自动作为一个服务器，所以每个设备都有一个server socket并监听连接。然后每个设备都能作为客户端建立一个到另一台设备的连接。另外一种代替方法是，一个设备按需打开一个server socket，另外一个设备仅初始化一个到这个设备的连接。

- Note: 如果两个设备在建立连接之前并没有配对，那么在建立连接的过程中，Android框架将自动显示一个配对请求的notification或者一个对话框，如Figure 3所示。所以，在尝试连接设备时，你的应用程序无需确保设备之间已经进行了配对。你的RFCOMM连接将会在用户确认配对之后继续进行，或者用户拒绝或者超时之后失败



作为一个服务器进行连接

- 当你成功地连接了两台(或多台)设备时，每个设备都有一个已连接的BluetoothSocket。这时你可以在设备之间共享数据，乐趣才刚开始。使用BluetoothSocket，传输二进制数据的过程是简单的：
  - 分别通过getInputStream()和getOutputStream()获得管理数据传输的InputStream和OutputStream。
  - 通过read(byte[])和write(byte[])从流中读取或写入数据。
- 你必须使用一个线程专门用于数据的读或写。这是非常重要的，因为read(byte[])和write(byte[])方法都是阻塞调用。read(byte[])将会阻塞到流中有数据可读。write(byte[])一般不会阻塞，但当远程设备的中间缓冲区已满而对方没有及时地调用read(byte[])时将会一直阻塞。所以，你的线程中的主循环将一直用于从InputStream中读取数据

例子

```
private class ConnectedThread extends Thread {
private final BluetoothSocket mmSocket;
private final InputStream mmInStream;
private final OutputStream mmOutStream;
 public ConnectedThread(BluetoothSocket socket) {
mmSocket = socket;
InputStream tmpIn = null;
OutputStream tmpOut = null;
// Get the input and output streams, using temp objects because
// member streams are final
try {
tmpIn = socket.getInputStream();
tmpOut = socket.getOutputStream();
} catch (IOException e) { }
mmInStream = tmpIn;
mmOutStream = tmpOut;
}
```

例子

```
public void run() {
byte[] buffer = new byte[1024];  // buffer store for the stream
int bytes; // bytes returned from read()
// Keep listening to the InputStream until an exception occurs
while (true) {
try {
// Read from the InputStream
bytes = mmInStream.read(buffer);
// Send the obtained bytes to the UI Activity
mHandler.obtainMessage(MESSAGE_READ, bytes, -1, buffer).sendToTarget();
} catch (IOException e) {
break;
}
}
}
/* Call this from the main Activity to send data to the remote device */
public void write(byte[] bytes) {
try {
mmOutStream.write(bytes);
} catch (IOException e) { }
}
/* Call this from the main Activity to shutdown the connection */
public void cancel() {
try {
mmSocket.close();
} catch (IOException e) { }
}}
```

目标

- 什么是蓝牙
- Android对蓝牙的支持