Wireless Communication Using Bluetooth and Android

Michael Price
11/11/2013

Abstract

The purpose of this application note is to explain how to implement Bluetooth for wireless applications. The introduction covers the hardware and specifications behind the technology and the subsequent sections cover implementation for the Android platform. The tutorial outlines the steps to start a project, enable Bluetooth, and design the layout. This tutorial is aimed at people unfamiliar with java and the Android development environment.

Introduction

Bluetooth is a radio frequency standard used for short-range small-scale applications. Developed by the Bluetooth Special Interest Group, Bluetooth technology was designed to be highly compatible with various types of equipment and applications. It's low power consumption and ease of use makes it a very viable option for many short-range wireless applications.

Choosing a Bluetooth Module

Since Bluetooth is a standardized technology, there is less variation in operation between different modules. The main parameters to consider are signal transmission power, receiver sensitivity, power consumption, size, and packaging.

Transmission power and receiver sensitivity are directly related to the range of the module. For the lower end transmission power, the range can be as short as 10 meters. As the power increases, a Bluetooth connection can reach up to 150 meters under ideal conditions.

Power consumption for Bluetooth is comparatively low for a radio frequency technology. For mobile applications where power budget is limited, the power required for Bluetooth can become an issue. Some modules support a low energy version for short packet bursts. Streaming applications will require more throughput so higher power consumption is unavoidable, but this is not the case for some data applications. One manufacturer, BlueRadios, boasts that their BR-LE4.0-S2A module can send 20 bytes per second while drawing around 30 microAmps of current. This would correspond to 100 microWatts of power consumption which would correspond to 8,000 hours of operation using a coin-sized watch battery. This low-energy option would be ideal for text communication since 20 bytes corresponds to 10 ASCII characters. Even for two-way communication, five ASCII characters per second corresponds to approximately one word per second or 60 words per minute which far exceeds the average typist's speed.

In applications where data streaming is necessary, a dual mode enable low energy chip is the appropriate choice. While the current draw is still low at less than 30 milliAmps, battery life becomes an issue for mobile devices. For streaming applications, Bluetooth can support up to 3 Mbps data transfer. This rate is more than enough for two-way audio streaming that would be necessary to use a Bluetooth device as a mobile handset and there is still room to transmit more data if the application requires.

Pairing Bluetooth Devices

Pairing Bluetooth devices is generally a simple task. Typically devices have a discovery mode that allows them to function as both master and slave. During this mode, the device is able to see other Bluetooth devices in the area. Some devices require a PIN exchange to pair for security purposes, but oftentimes this PIN defaults to 0000. For Bluetooth modules, it isn't always clear how to force the device into discovery mode since there is no user interface. Typical chips automatically power up in discovery mode, but it is not clear if they will automatically pair with a previously used device. At this point, consulting the data sheet for the specific module is recommended. Once paired, the module will most likely switch to slave communication unless the module is connected to the master device and will be unavailable for pairing with other devices not currently in the network.

Deciding on Profile

The types of applications supported by a Bluetooth network are directly related to the available profiles. There are more than 30 standardized Bluetooth profiles available for use. Each profile is used for a different type of application, so an understanding of profile capabilities and applications is necessary. A few of the common profiles are listed as examples, but the full list should be consulted for proper implementation.

Generic Access Profile

Generic access profile manages the behavior of the Bluetooth links. It defines the actions of a device in standby and ensures that links can be established between devices.

Intercom Profile

The intercom profile is used for a type of voice communication. As expected from its name, IP is used to broadcast audio from one device at a time to all other devices on the network. In this profile, only one device can transmit audio at a time.

Serial Port Profile

Serial port profile is used to emulate the RS-232 serial port.

Headset

This profile is one of the most used profiles for mobile phones. It allows two-way audio streaming between the phone and the device and is used for voice calls.

Generic Object Exchange Profile

Generic Object Exchange Profile is used to exchange objects between devices. Objects are identified by their extensions before transfer and can include photos or videos. For this profile, the objects are pushed from the server to the client

File Transfer Profile

File Transfer Profile allows a client to browse and pull files from a server.

Synchronization Profile

This profile allows devices to synchronize information such as agendas.

Bluetooth for Android

Android is a popular platform that includes many easily-accessible tools. With the popularity of these mobile devices, it would be beneficial to understand how to setup and implement Bluetooth functionality for many applications. The methods in this section are specifically for java using Eclipse with the Android development kit and aimed at developers who are unfamiliar with the IDE. The only necessary preparation is to download and set up the IDE.

Setup Device

To start a new Project in Eclipse, go to File -> New -> Android Application Project. Name your application and leave the other fields set to the default. Eclipse will generate several files including the MainActivity.java class, AndroidManifest.xml, and activity_main.xml. The MainActivity.java class contains the code for the project and will require the most editing. The Android Manifest contains information about the application such as permissions and minimum device compatibility. The permissions will need to be updated to include access to Bluetooth. The activity_main.xml file contains the layout for the main activity class and serves as the user interface.

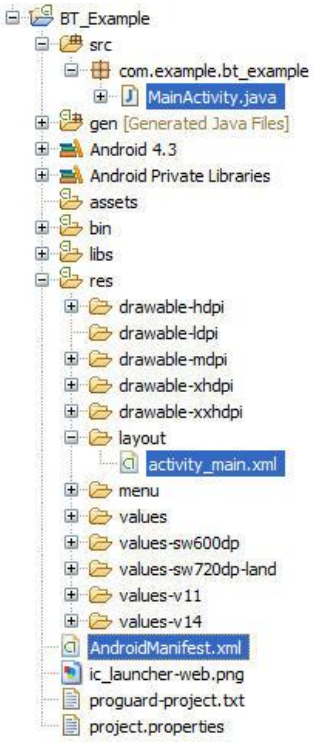

Figure 1 - Library

To allow access to the devices Bluetooth features, we need to first declare the permission in the manifest file. The manifest file already contains other information, so the only addition necessary is the following line:

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

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.bt_example"
    android:versionCode="1"
    android:versionName="1.0" >

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

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.bt_example.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Figure 2 - Permissions

After modifying the manifest file, open the MainActivity.java class for editing. We will want to set up a variable that corresponds to the phones Bluetooth adaptor. Add the following line after the line starting with public class MainActivity:

```
Private Bluetooth Adapter mBluetoothAdapter = null;
```

Now we can define the mBluetooth Adapter variable using the getDefaultAdapter method. In the onCreate method add the following line before the curly brace:

```
mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
```

If the device does not support Bluetooth, the variable will remain null and we can use an IF statement to quit the application. This code is not necessary for functionality but should be included in the final version of the class to improve stability.

The next step is checking that Bluetooth is enabled on the device and enabling it if necessary. We will use the isEnabled method to check that the adapter and enable it if necessary. The following block of code should be added after the onCreate method:

```
        public void onStart(){
                super.onStart();
                if (!mBluetoothAdapter.isEnabled()) {
                Intent enableBtIntent = new
    Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                startActivityForResult(enableBtIntent, 1);
         }
```

Now the device is setup for Bluetooth and we can begin to code the project for its specific function.

Sending a Message

In order to type a message to send over Bluetooth, we need to modify the layout to include a text box and a button. Open the activity_main.xml file and switch the tab at the bottom so the xml code is visible. Modify the code so that it matches the following block:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity" >

    <EditText android:id="@+id/edit_message"
        android:layout_weight = "1"
        android:layout_width="0dip"
        android:layout_height="wrap_content"
        android:hint="Enter a Message" />

     <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Send"
       android:onClick="sendMessage" />

</LinearLayout>
```

Make sure to change the RelativeLayout at the beginning and end to LinearLayout. Switch to the graphical layout tab to make sure it matches the layout shown in Figure 3. The button corresponds to a method called sendMessage that will run when clicked.

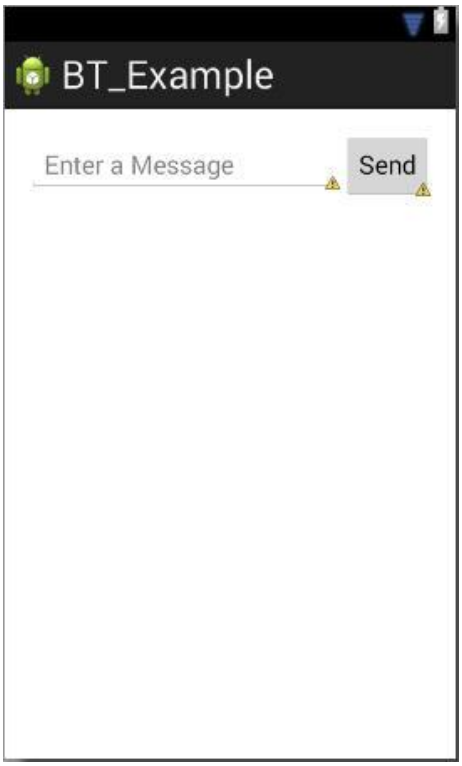

Figure 3 - Layout

Prior to sending a message, we'll need to write the method called by the button press. Before that, the Bluetooth connection needs to be initialized between the two devices. One device will be set up as the server and the other will be set up as the client. Examples of this can be found on the android developer page in the Bluetooth section. This setup is outside the scope of this application note and beyond the understanding of an engineer without an extensive programming background.

References

Labiod, Houda, Hossam Afifi, and Santis C. De. *Wi-fi, Bluetooth, Zigbee and Wimax*. Dordrecht: Springer, 2007. Internet resource.

http://developer.android.com/guide/topics/connectivity/bluetooth.html