Understanding Heart and Body Status from a Smartphone

– Wireless Programming on Android–

Guillaume Lopez
Living Environment Laboratory
The University of Tokyo, School of Engineering

Smartphone Sensor Web #4

「Table of Contents」

🗁. Class #4

★. Presentation of main points in Wireless Data Transmission Programming

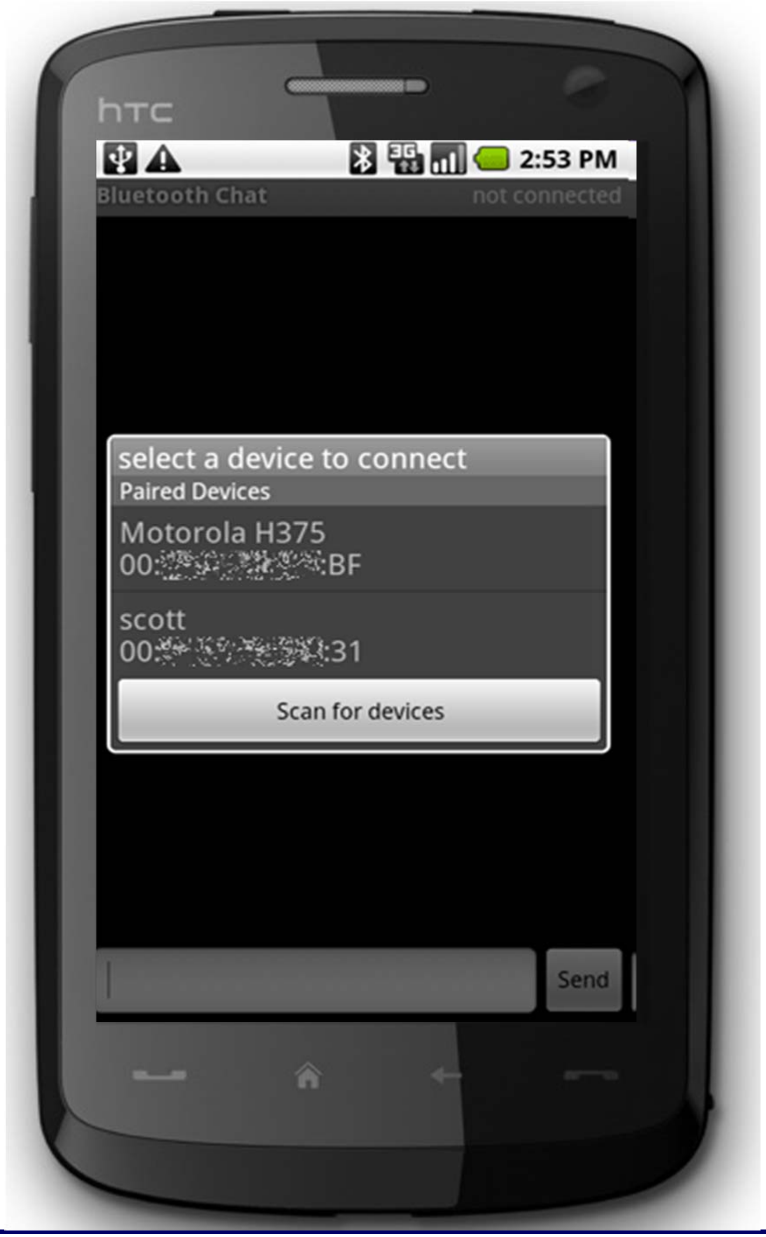
★. Example source code analysis ☺

Communication capabilities

- Wireless type
  - WIFI,
  - **Bluetooth,**
  - NFC

- Wired type
  - USB,

Wireless Programming on Android



Bluetooth Chat

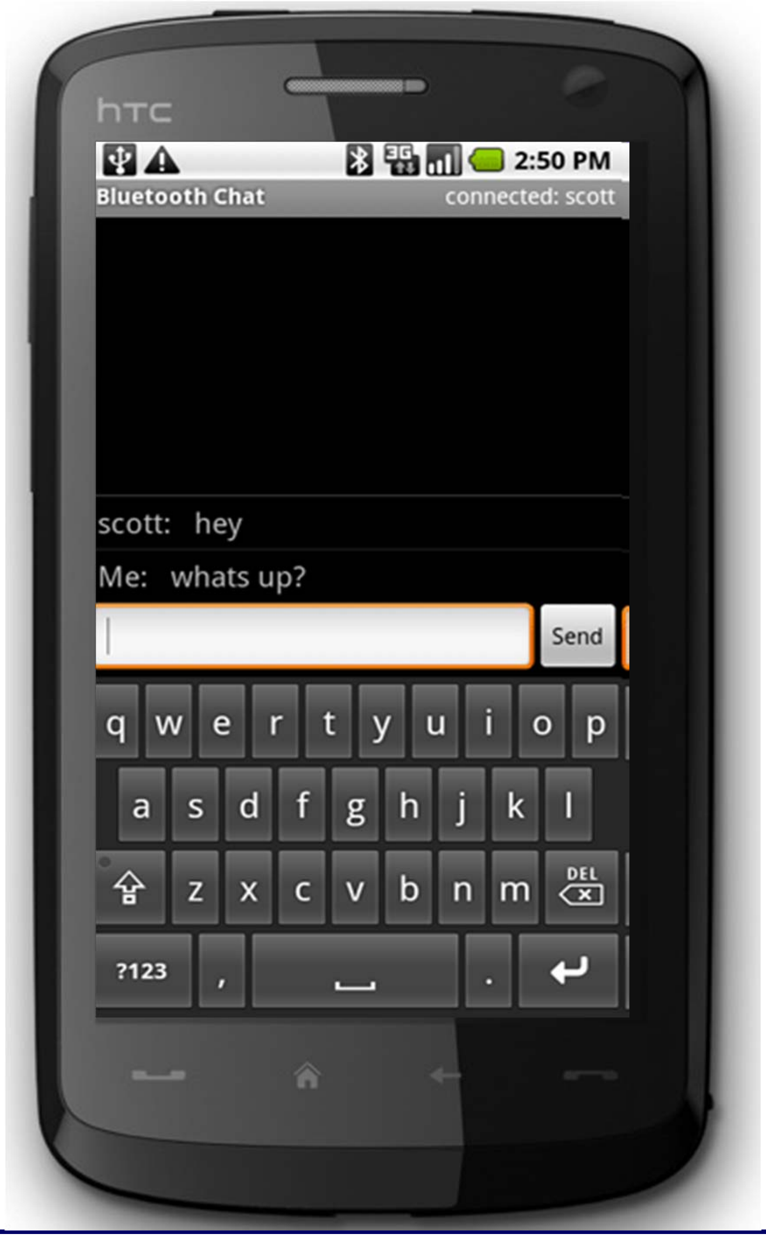
not connected

select a device to connect

Paired Devices

Motorola H375
00[redacted]:BF

scott
00[redacted]:31



Bluetooth Chat

connected: scott

scott: hey

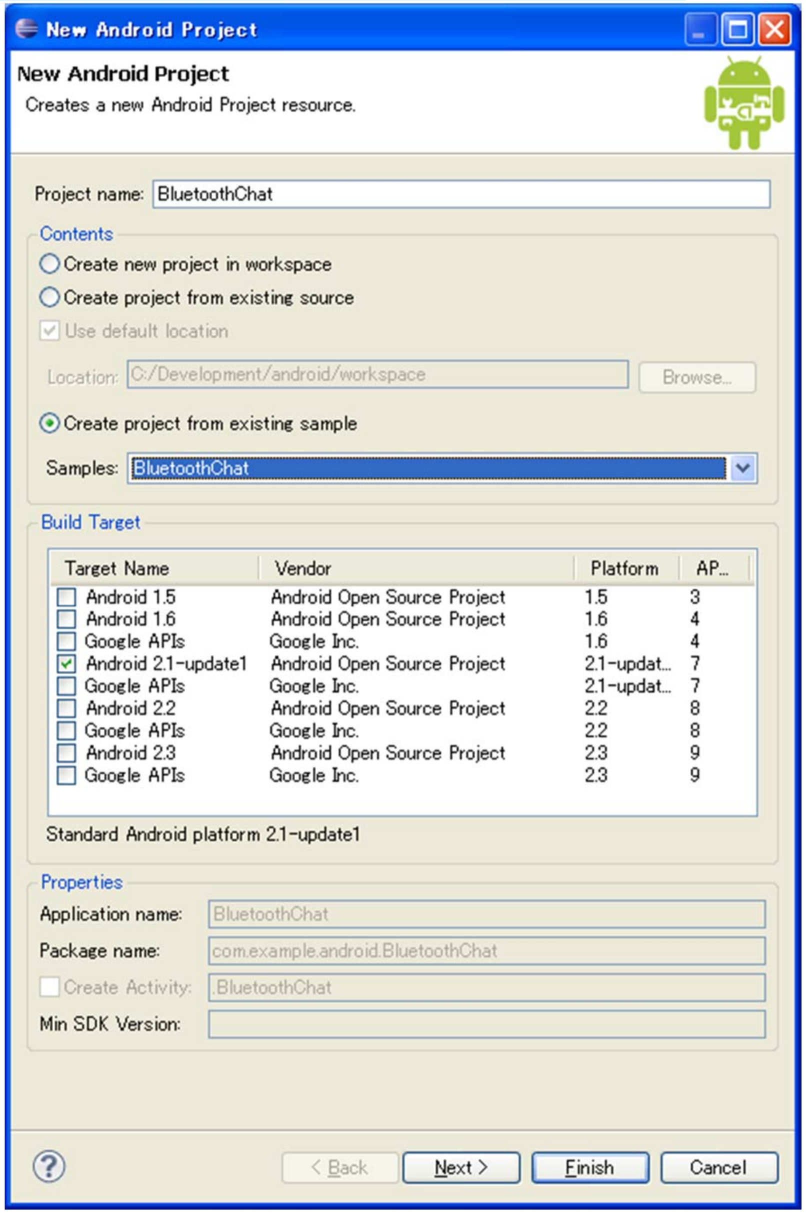
Me: whats up?

Fundamental Bluetooth Capabilities

- Point-to-point and Multipoint wireless features
  - Scanning for other Bluetooth devices
  - Querying the local Bluetooth adapter for paired Bluetooth devices
  - Manage multiple connections
  - Establishing RFCOMM channels/sockets
  - Connecting to a remote device
  - Transferring data over Bluetooth (bi-directional)

Sample Program Reading

- BluetoothChat
  - Create project from existing sample
  - Android 2.1-update1
- Content
  - BluetoothChat
  - BluetoothChatService
  - DeviceListActivity



New Android Project

New Android Project

Creates a new Android Project resource.

Project name: BluetoothChat

Contents

○ Create new project in workspace
○ Create project from existing source
☑ Use default location
Location: C:/Development/android/workspace
● Create project from existing sample
Samples: BluetoothChat

Build Target

	Target Name	Vendor	Platform	AP...
☐	Android 1.5	Android Open Source Project	1.5	3
☐	Android 1.6	Android Open Source Project	1.6	4
☐	Google APIs	Google Inc.	1.6	4
☑	Android 2.1-update1	Android Open Source Project	2.1-updat...	7
☐	Google APIs	Google Inc.	2.1-updat...	7
☐	Android 2.2	Android Open Source Project	2.2	8
☐	Google APIs	Google Inc.	2.2	8
☐	Android 2.3	Android Open Source Project	2.3	9
☐	Google APIs	Google Inc.	2.3	9

Standard Android platform 2.1-update1

Properties

Application name: BluetoothChat
Package name: com.example.android.BluetoothChat
☐ Create Activity: .BluetoothChat
Min SDK Version:

The 4 Basics: set-up, find, connect, transfer

- All of the Bluetooth APIs are available in the **android.bluetooth** package

- **BluetoothAdapter**: *the Bluetooth radio.* Set-Up

  It is the entry-point for all Bluetooth interaction.

  - Discover other Bluetooth devices,
  - Qquery a list of paired devices,
  - Instantiate a **BluetoothDevice** using a known MAC address,
  - Create a **BluetoothServerSocket** to listen for communications from other devices.

- **BluetoothDevice**: *remote Bluetooth device* Find

  Use this to

  - Request a connection with a remote device through a **BluetoothSocket**
  - Query information about the device (name, address, class, and pairing state).

The 4 Basics: set-up, find, connect, transfer

- **BluetoothSocket**: *interface for a Bluetooth socket*

  Connection point that allows an application to exchange data with another Bluetooth device via **InputStream** and **OutputStream**.

  Transfer

- **BluetoothServerSocket**: *socket that listens for incoming requests*

  In order to connect 2 Android devices, 1 device must open a server socket with this class.

  Return a connected **BluetoothSocket** when the connection is accepted.

  Connect

- **BluetoothClass**: *BT device general characteristics & capabilities*

  Read-only set of properties that define the device's major and minor device classes and its services.

  Does not reliably describe all Bluetooth profiles and services supported by the device, but is useful as a hint to the device type.

Permissions for Using Bluetooth

- Mandatory
  - BLUETOOTH_ADMIN

    Necessary to initiate *device discovery* or *manipulate settings*.
  - BLUETOOTH

    Necessary to perform any Bluetooth communication, such as *requesting a connection*, *accepting a connection*, and *transferring data*
  - Declare the Bluetooth permission(s) in your application manifest file.
- Declare in the application manifest file

```
<?xml version="1.0" encoding="utf-8"?>
<manifest … >

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

    <uses-sdk android:minSdkVersion="7" />

</manifest>
```

Setting Up Bluetooth

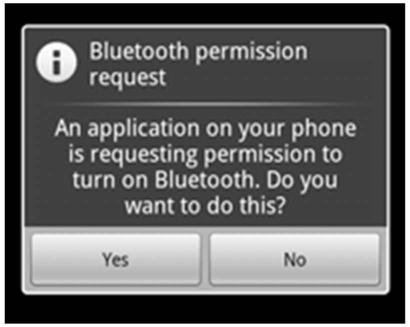
1. Get the BluetoothAdapter & Enable Bluetooth
   1. If Bluetooth is not supported, then disable any Bluetooth features.
   2. If Bluetooth is *supported, but disabled*, then you can request that the user enable Bluetooth without leaving your application.

```
BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
if (mBluetoothAdapter == null) {
    // Device does not support Bluetooth
}
Else {
   if (!mBluetoothAdapter.isEnabled()) {
       Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
       startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
   }
}
```

Check BT Support

Request to enable BT without stopping application

Enable BT



Bluetooth permission request

An application on your phone is requesting permission to turn on Bluetooth. Do you want to do this?

The enabling Bluetooth dialog

Finding Devices

- **Querying paired devices**

Check paired devices

Get MAC address from the BluetoothDevice object to initiate connection

```
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());
    }
}
```

- **Discovering devices**
  - Simply call `mBluetoothAdapter.startDiscovery();`
  - …

Finding Devices

- **Discovering devices**
  - Must register a BroadcastReceiver for the ACTION_FOUND Intent in order to receive information about each device discovered

```
// 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 BroadcastReceiver
IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
registerReceiver(mReceiver, filter); // Don't forget to unregister during
    onDestroy
```

Connecting Devices

You must implement both the server-side and client-side mechanisms

Procedure to set up a **server socket** and accept a connection:

1. Get a BluetoothServerSocket

   call listenUsingRfcommWithServiceRecord(String, UUID)

   The string is an identifiable name of your service (i.e. application name), which the system will write to a new Service Discovery Protocol (SDP) DB entry on the device.

   The UUID is also included in the SDP entry and will be the basis for the connection agreement with the client device. It uniquely identifies the service with which it wants to connect. It must match in order for the connection to be accepted.

2. Start listening for connection requests

   call accept()

   This is a blocking call. It will return when either a connection has been accepted or an exception has occurred. A connection is accepted only when a remote device has sent a connection request with a UUID matching the one registered with this listening server socket. When successful, accept() will return a connected BluetoothSocket.

3. Unless you want to accept additional connections, call close().

Connecting Devices

Procedure to set up a **client socket**:

1. Using the BluetoothDevice, get a BluetoothSocket

   call createRfcommSocketToServiceRecord(UUID).

   This initializes a BluetoothSocket that will connect to the BluetoothDevice. The UUID passed here must match the UUID used by the server device when it opened its BluetoothServerSocket (withlistenUsingRfcommWithServiceRecord(String, UUID)). Using the same UUID is simply a matter of hard-coding the UUID string into your application and then referencing it from both the server and client code.

2. Initiate the connection

   call connect().

   Upon this call, the system will perform an SDP lookup on the remote device in order to match the UUID. If the lookup is successful and the remote device accepts the connection, it will share the RFCOMM channel to use during the connection and connect() will return. This method is a blocking call. If, for any reason, the connection fails or the connect() method times out (after about 12 seconds), then it will throw an exception.

   Because connect() is a blocking call, this connection procedure should always be performed in a thread separate from the main Activity thread.

Managing a Connection

When you have successfully connected two (or more) devices, each one will have a connected BluetoothSocket.

You can share data between devices, using the BluetoothSocket:

1. Get the InputStream and OutputStream that handle transmissions through the socket, via getInputStream() and getOutputStream(), respectively.
2. Read and write data to the streams with read(byte[]) and write(byte[]).

Smartphone Sensor Web #4

「Location Sensitive Android Programming」

🗁. References

★. Android Dev Guide
http://developer.android.com/guide/topics/wireless/bluetooth.html