

# Bluetooth and Socket

# Sample Code

140.114.79.79/dropbox/BluetoothExample.zip

- ListView of paired devices
- Toast of discovered devices
- A button to open server connection listener
- Send “Hello World” when you click the paired device (you can see the message in Log.i)

# Setting up Bluetooth (1/2)

- Always check the permission first

```
<manifest ...>
```

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

```
  ...
```

```
</manifest>
```

# Make Sure your Device Supports BT

```
BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.  
getDefaultAdapter();  
  
if (mBluetoothAdapter == null) {  
    // Device does not support Bluetooth  
}
```

# Enable your Bluetooth

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

```
if (mBluetoothAdapter == null) {
```

```
    // Device does not support Bluetooth
```

```
}
```

```
else { // Device supports Bluetooth
```

```
    if (!mBluetoothAdapter.isEnabled()) {
```

```
        Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
```

```
        startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
```

```
    }
```

```
}
```

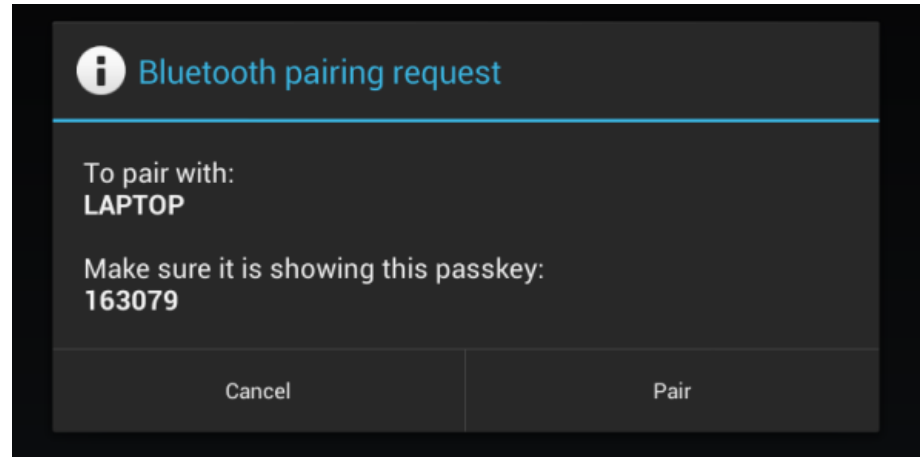
# Query Paired Devices

- Before device discovery, you may want to see if the desired device is already known
- To do so, call `getBondedDevices()`. This will return a set of `BluetoothDevices` representing paired devices

```
Set<BluetoothDevice> pairedDevices = mBluetoothAdapter.getBondedDevices();  
if (pairedDevices.size() > 0) { // If there are paired devices  
    for (BluetoothDevice device : pairedDevices) { // Loop through paired devices  
        Toast.makeText(getApplicationContext(), device.getName() + " Paired \n" + device.  
getAddress(), Toast.LENGTH_LONG).show();  
    }  
}
```

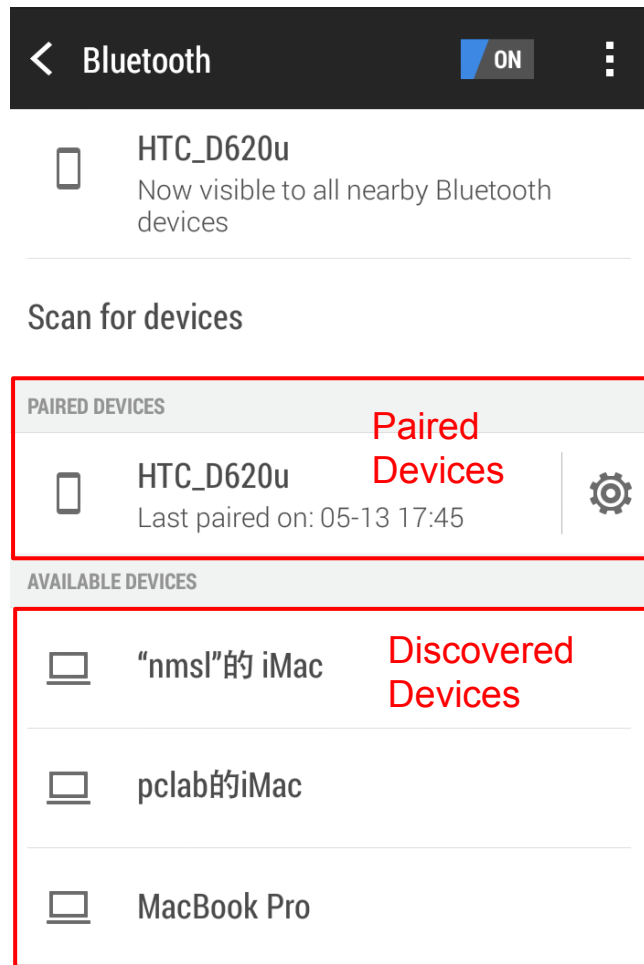
# What is pairing?

- To be paired means that two devices are aware of each other's existence, have a shared link-key that can be used for authentication



# Discovering (1/2)

- If the desired devices is not paired, we do discovering to find it.
- We need to use Intent filter with Action = “BluetoothDevice.ACTION\_FOUND”





# Discovering (2/2)

- After new an IntentFilter, we need to register a BroadcastReceiver to receive the information of other bluetooth devices

```
registerReceiver(mReceiver, filter);
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);
            Toast.makeText(getApplicationContext(),device.getAddress(), Toast.LENGTH_SHORT).
show();
        }
    }
};
```

# Pairing

- You can do pairing manually
- Or Just use UUID of your bluetooth to do socket connection. The Android system will automatically do pairing. (Next slide)
- You can use Intent to do pairing after discovering

```
Intent intent = new Intent();  
intent.putExtra(EXTRA_DEVICE_ADDRESS, DiscoveredMACaddress);
```

# Difference Between Connect and Pair

- Pair: two devices are known each other
- Connect: devices currently share an channel and are able to transmit data with each other
  - If the devices are paired, it will directly connect to the other device
  - if not, the android system will automatically do pairing and then connect to the other device

# Enable Discoverability

- To make other devices can see you
- Enable your discoverability using Intent

```
public void enableDiscoverability(){  
    Intent discoverableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_DISCOVERABLE);  
    discoverableIntent.putExtra(BluetoothAdapter.EXTRA_DISCOVERABLE_DURATION, 300);  
    startActivity(discoverableIntent);  
}
```

# Connection - Server

- You need to create a thread to wait for connection
- Please see the function “AcceptThread” in the sample code

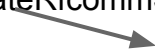
# Connection - Server (code)

```
while (true) {
    try {
        Log.i("Server:", "Waiting for connection");
        socket = mmServerSocket.accept();
    } catch (IOException e) {
        break;
    }
    // If a connection was accepted, new a thread to receive the message "Hello World" sent from client
    if (socket != null) {
        if (mConnectedThread != null) {mConnectedThread.cancel(); mConnectedThread = null;}
        mConnectedThread = new ConnectedThread(socket);
        mConnectedThread.start();
        Log.i("Server:", "one client connected");
    }
}
```

# Bluetooth Socket - client

- see the function “ConnectThread” in sample code

```
mmSocket=device.createRfcommSocketToServiceRecord(MY_UUID);  
mmSocket.connect();  
Log.i("Client:", "connect socket success");
```

 bluetooth device in server side

# Bluetooth Socket - read (server)

- read the function “Connected Thread” in your sample code

```
while (true) {  
    try {  
        // Read from the InputStream  
        bytes = mmlnStream.read(buffer);  
        String str = new String(buffer);  
        Log.i("Server get message:", str);  
    } catch (IOException e) {  
        break;  
    }  
}
```



# Bluetooth Socket - write (client)

- read the function “ConnectedThread” in the sample code

```
String message="Hello World!";
```

```
mConnectedThread.write(message.getBytes());
```

```
public void write(byte[] bytes) {
```

```
    try {
```

```
        mmOutputStream.write(bytes);
```

```
    } catch (IOException e) { }
```

```
}
```