藍芽連線

前置作業

範例與函式庫下載

- https://github.com/silencecork/AndroidBluetoothWorkshop2014

藍芽連線步驟

藍芽連線角色

- 至少兩個裝置，都有藍芽
- 裝置分為一個Server與多個Client
- Sever
  - 時常是負責提供資料的一端，e.g. Sensor的數值
  - 超過一個以上的連線由Server管理
- Client
  - 時常是負責接收資料的一端

藍芽配對

- Client和Server都須開啟藍芽
- 設定Server端為可搜尋狀態(Discoverable)

- Client進行裝置掃描

- Client針對找尋到要連線的裝置進行「配對」

- 裝置間的配對僅需進行一次，之後則可以直接連線

SERVER連線步驟

- 開啟藍芽
- 使用UUID建立Sever端 (BluetoothServerSocket)
- 等待連線

- 待Client連入後，建立InputStream, OutputStream 進行資料的接收與傳送

- 不使用時進行斷線

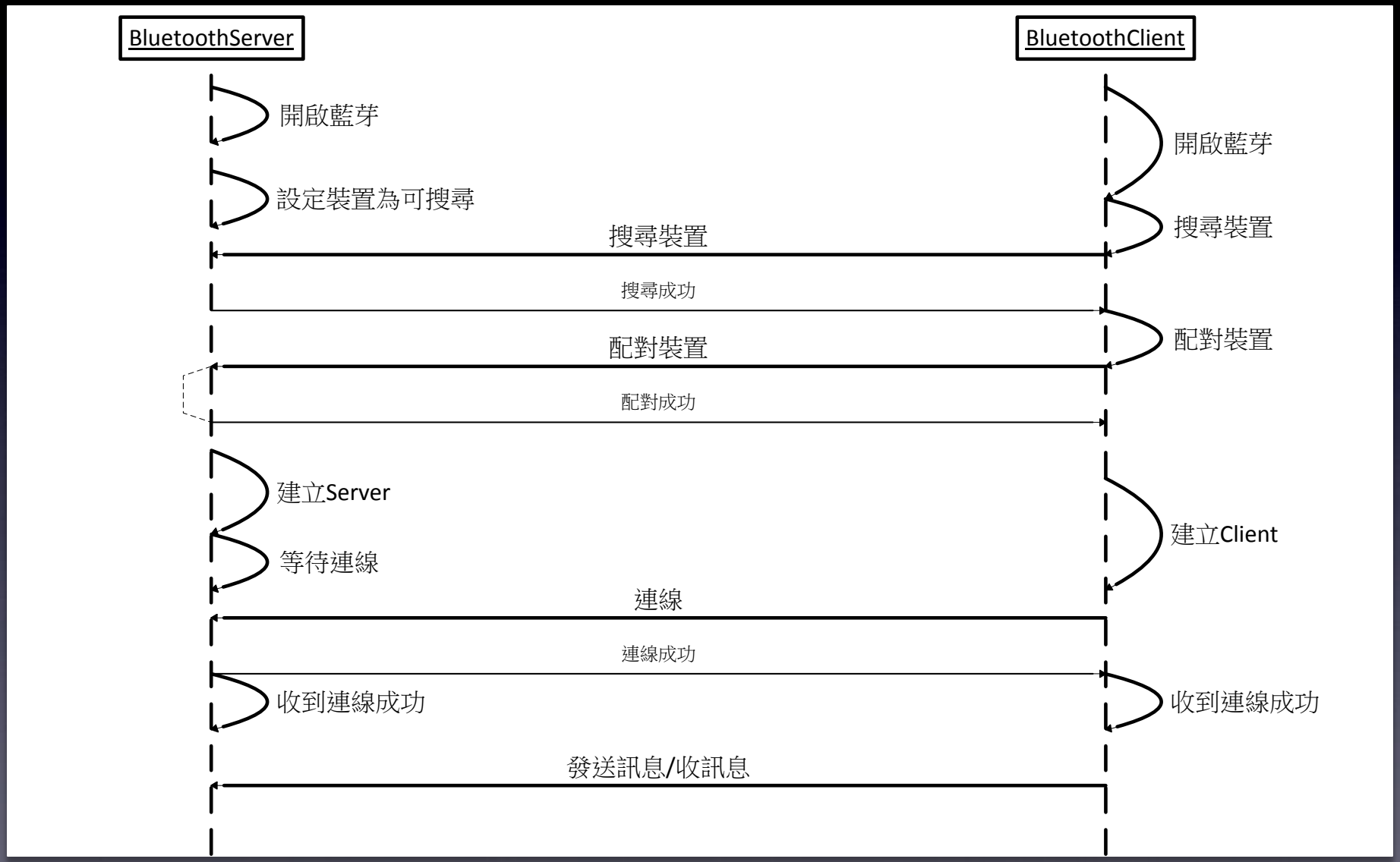
CLIENT連線步驟

- 開啟藍芽
- 建立藍芽Client端 (BluetoothSocket)
- 使用UUID對已配對且在等待連線的裝置進行連線

- 連入Server後，建立InputStream, OutputStream進行資料的接收與傳送

- 不使用時進行斷線

藍芽連線循序圖


BluetoothServer
BluetoothClient
開啟藍芽
開啟藍芽
設定裝置為可搜尋
搜尋裝置
搜尋裝置
搜尋成功
配對裝置
配對裝置
配對成功
建立Server
建立Client
等待連線
連線
連線成功
收到連線成功
收到連線成功
發送訊息/收訊息

使用函式庫

- 因為Android藍芽連線有許多細節需要處理
- 本範例將細節處理完畢，直接使用Library則不用處理連線問題、Stream問題
- 下頁為配對和連線的步驟對應的Library function

藍芽配對對應FUNCTION

- Client和Server都須開啟藍芽
  - LocalBluetoothManager.turnOnBluetooth()
- 設定Server端為可搜尋
  - LocalBluetoothManager.setDeviceDiscoverable()
- Client進行裝置掃描
  - LocalBluetoothManager.discoverDevice()
- Client針對找尋到要連線的裝置進行「配對」
  - LocalBluetoothManager.pairDevice()

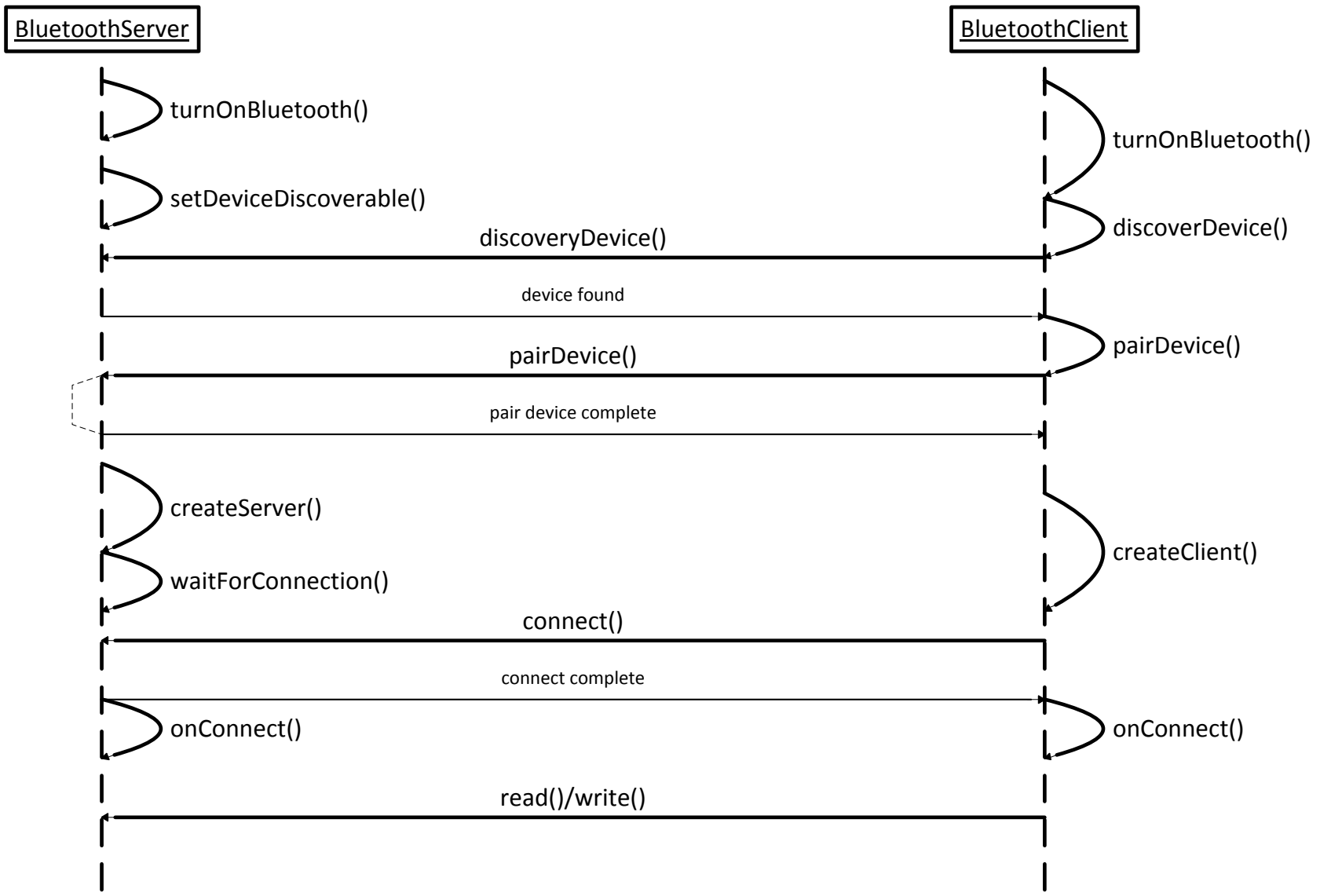
SERVER連線對應FUNCTION

- 使用UUID建立Sever端
  - BluetoothConnectionHelper.createServer()
- 等待連線
  - BluetoothConnectionHelper.waitForConnection()
- 待Client連入後，建立InputStream, OutputStream進行資料的接收與傳送
  - OnBluetoothMessageListener.onConnected()
  - OnBluetoothMessageListener.onMessageReceived()
  - OnBluetoothMessageListener.sendMessage()
- 不使用時進行斷線
  - BluetoothConnectionHelper.close()

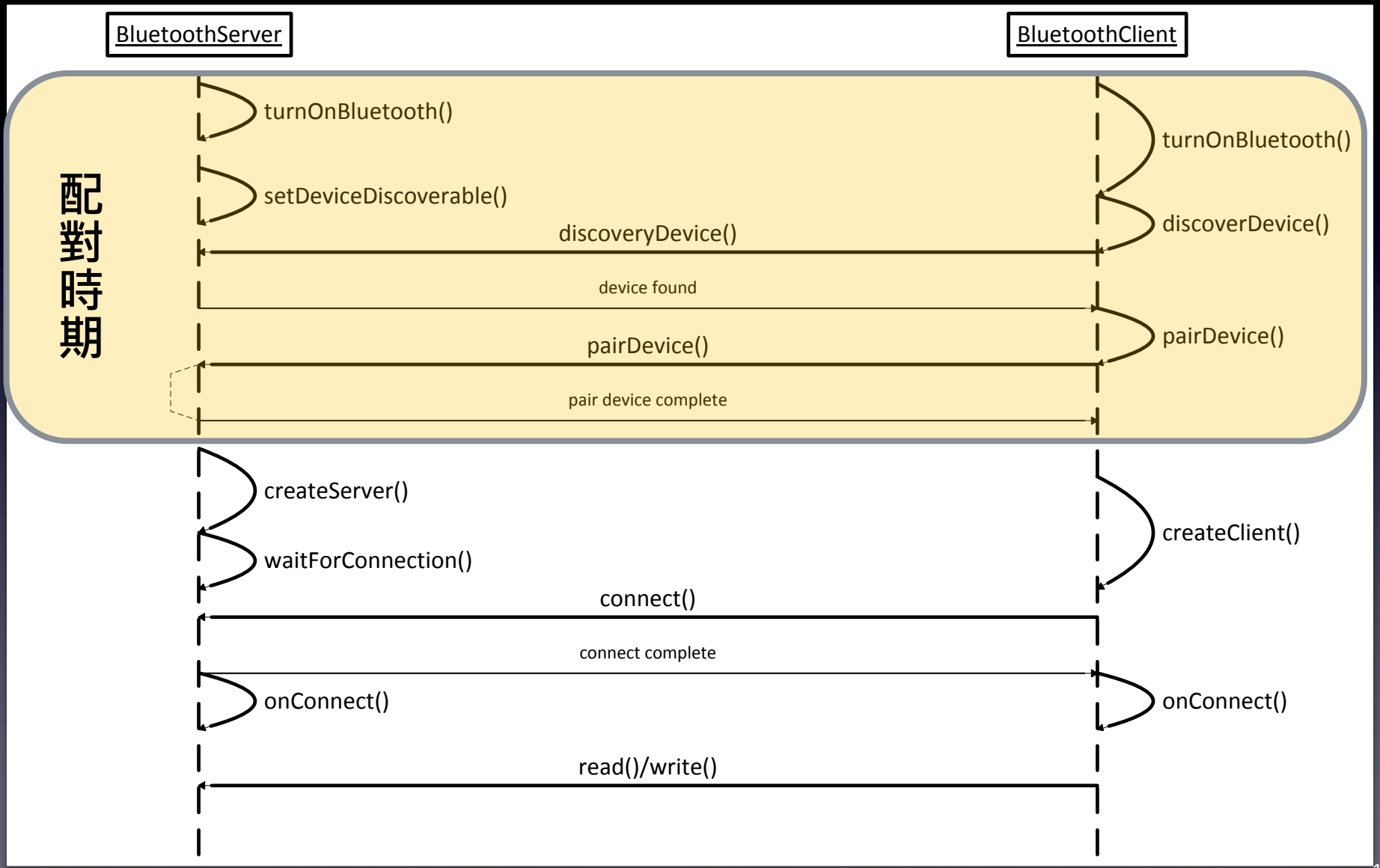
CLIENT連線步驟

- 建立藍芽Client端
  - BluetoothConnectionHelper.createClient()
- 使用UUID對已配對且在等待連線的裝置進行連線
  - BluetoothConnectionHelper.connect()
- 連入Server後，建立InputStream, OutputStream進行資料的接收與傳送
  - OnBluetoothMessageListener.onConnected()
  - OnBluetoothMessageListener.onMessageReceived()
  - OnBluetoothMessageListener.sendMessage()
- 不使用時進行斷線
  - BluetoothConnectionHelper.close()

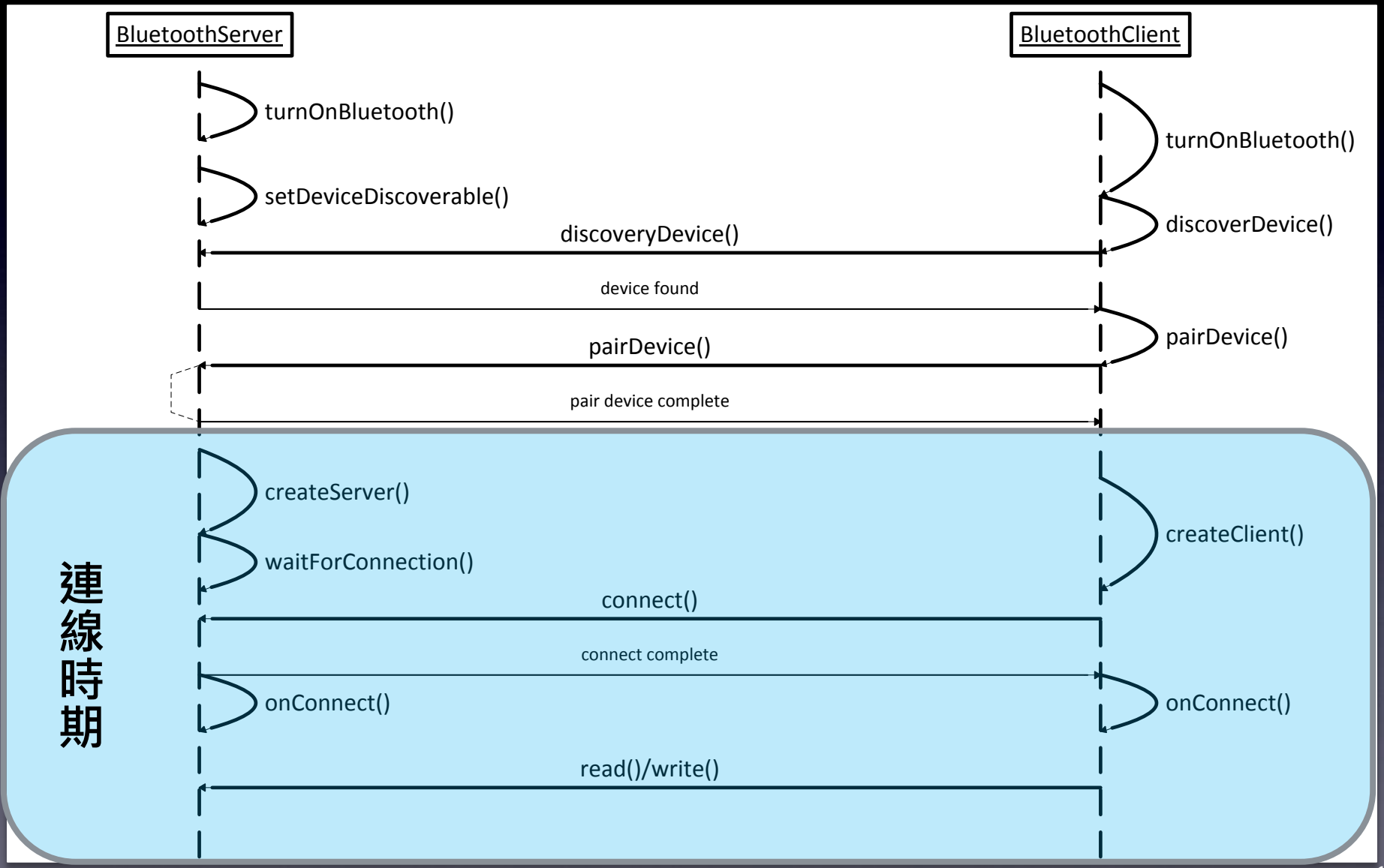
藍芽連線循序圖


BluetoothServer
BluetoothClient
turnOnBluetooth()
turnOnBluetooth()
setDeviceDiscoverable()
discoverDevice()
discoveryDevice()
device found
pairDevice()
pairDevice()
pair device complete
createServer()
createClient()
waitForConnection()
connect()
connect complete
onConnect()
onConnect()
read()/write()

藍芽連線循序圖


BluetoothServer
BluetoothClient
配對時期
turnOnBluetooth()
turnOnBluetooth()
setDeviceDiscoverable()
discoverDevice()
discoveryDevice()
device found
pairDevice()
pairDevice()
pair device complete
createServer()
createClient()
waitForConnection()
connect()
connect complete
onConnect()
onConnect()
read()/write()

藍芽連線循序圖


BluetoothServer
BluetoothClient
turnOnBluetooth()
turnOnBluetooth()
setDeviceDiscoverable()
discoverDevice()
discoveryDevice()
device found
pairDevice()
pairDevice()
pair device complete
連線時期
createServer()
createClient()
waitForConnection()
connect()
connect complete
onConnect()
onConnect()
read()/write()

BluetoothChat

ANDROID藍芽連線

權限設定

- 打開AndroidManifest.xml

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

- 加上述兩項權限，App才能夠使用藍芽連線

建立UUID

- Android之間的連線採用UUID為識別碼
  - UUID全名為通用唯一識別碼 (Universal Unique Identifier)

- 若要連線的Server裝置為Android 4.2之前的裝置或非Android的裝置，e.g. 如Adruino的藍芽晶片使用UUID
  00001101-0000-1000-8000-00805F9B34FB

- 若連線的Server裝置為Android 4.2含以上，則可到以下網址申請隨機的UUID
  - http://www.uuidgenerator.net/

準備使用LIBRARY

- 在使用藍芽連線的Library時，主要會應用到兩個class
- LocalBluetoothManager
  - 管理藍芽的開關、搜尋、配對
- BluetoothConnectionHelper
  - 建立Server、Client、收發訊息、斷線連線

準備使用LIBRARY

```
LocalBluetoothManager.getInstance().startSession(this)
```

- startSession()
  - 使用前一定都要呼叫這個方法，之後才能夠使用LocalBluetoothManager
  - 參數：目前所在的Activity

準備使用LIBRARY

```
LocalBluetoothManager.getInstance().endSession()
```

- endSession()
  - 不使用時一定要呼叫
  - 多半在Activity的onDestroy()呼叫

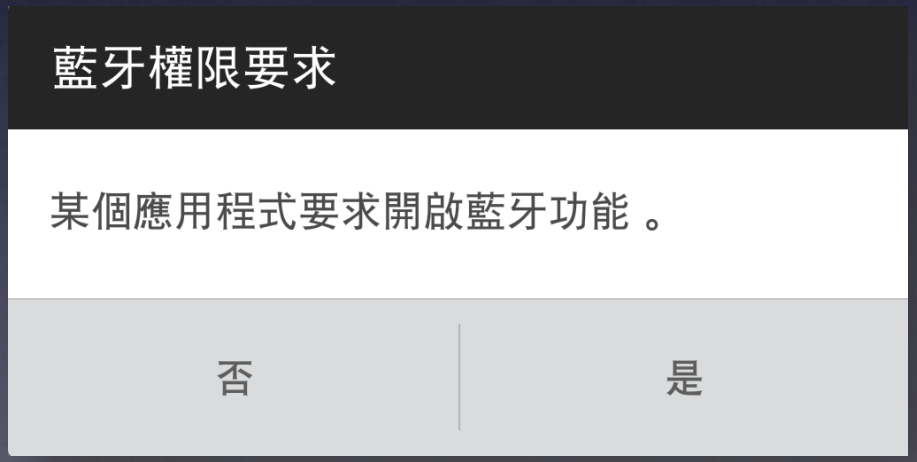
開啟藍芽

```
if (!LocalBluetoothManager.getInstance().isBluetoothTurnOn()) {
    LocalBluetoothManager.getInstance().turnOnBluetooth(this);
    return;
}
```

- isBluetoothTurnOn()
  - 檢查目前藍芽是否開啟
- turnOnBluetooth()
  - 開啟藍芽
  - 參數: 所在的Activity

開啟藍芽

- 在Android中開啟藍芽屬於系統保護的功能
- 要開啟藍芽必須得經由系統規定的流程
- 所以當呼叫turnOnBluetooth()時，會讓目前的Activity暫停，啟動系統外觀如Dialog的Activity



開啟藍芽

```
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (!LocalBluetoothManager.getInstance().isBluetoothTurnOn()) {
        finish();
    } else {
        waitConnection();
    }
}
```

- 當使用者選擇是否開啟，會回到原本的Activity，進入點是onActivityResult()

開啟藍芽

```
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (!LocalBluetoothManager.getInstance().isBluetoothTurnOn()) {
        finish();
    } else {
        waitConnection();
    }
}
```

- 在onActivityResult可以再次檢查藍芽是否開啟
- 沒有開啟就可以直接呼叫finish()結束目前的Activity

BluetoothChat

建立SERVER端

建立藍芽SERVER

```
private void waitConnection() {
    mHelper =
       BluetoothConnectionHelper.createServer(
              UUID.fromString(APP_UUID), 2);
    mHelper.setMessageReceiver(mListener);
    mHelper.waitForConnection();
}
```

- 呼叫BluetoothConnectionHelper.createServer()
  - 參數1: UUID，可見之前敘述
  - 參數2: Server可以接受的最大連線數，最大不得超過7

建立藍芽SERVER

```
private void waitConnection() {
    mHelper =
       BluetoothConnectionHelper.createServer(
              UUID.fromString(APP_UUID), 2);
    mHelper.setMessageReceiver(mListener);
    mHelper.waitForConnection();
}
```

- setMessageReceiver()
  - 因為藍芽傳接訊息是通過網路的多執行緒完成，使用Library只需要註冊Listener接收訊息即可
  - 參數Listener為OnBluetoothMessageListener

建立藍芽SERVER

```
private void waitConnection() {
    mHelper =
       BluetoothConnectionHelper.createServer(
              UUID.fromString(APP_UUID), 1);
    mHelper.setMessageReceiver(mListener);
    mHelper.waitForConnection();
}
```

- waitForConnection()
  - 呼叫了這個方法，Server才正式啟用，開始等待連線
- 所以使用上建議要有順序：**建立Server、註冊Listener、等待連線**

SERVER結束

```
BluetoothConnectionHelper.close();
```

- close()
  - 關閉Server連線，多半在Activity的onDestroy()呼叫
  - 呼叫後，其他連線的Client就會收到斷線的通知

訊息接收LISTENER

```
private OnBluetoothMessageListener mListener =
        new OnBluetoothMessageListener() {

        public void onMessageReceived(
                BluetoothDevice device, String message) { …… }

        public void onDisconnect(BluetoothDevice device) { …… }

        public void onConnected(BluetoothDevice device) { …… }

};
```

- onMessageReceived()
  - 當收到訊息時，會由這個method進入
  - 參數1：傳入訊息的Device
  - 參數2：傳入的訊息

訊息接收LISTENER

```
private OnBluetoothMessageListener mListener =
        new OnBluetoothMessageListener() {

        public void onMessageReceived(
                BluetoothDevice device, String message) { …… }

        public void onDisconnect(BluetoothDevice device) { …… }

        public void onConnected(BluetoothDevice device) { …… }

};
```

- onConnected(), onDisconnect()
  - 當連線和斷線時會出發的method
  - 參數：連線或斷線的裝置，不一定有值

SERVER傳送訊息

```
if (mHelper.isConnect()) {
      mHelper.sendMessage(content);
}
```

- isConnect()
  - 檢查是否已經連線
- sendMessage()
  - 傳送的內容
  - 參數：要傳送的字串，以英文數字為主，非智慧型手機可能無法解析中文字 (如Arduino)

設定為可搜尋

```
if (!LocalBluetoothManager.getInstance().isDeviceDiscoverable()) {
        LocalBluetoothManager.getInstance().setDeviceDiscoverable(this);
}
```

- 若Server要讓其他裝置看見，必須得設
- isDeviceDescoverable()
  - 檢查裝置目前是否為可以被搜尋到
- setDeviceDiscoverable()
  - 將裝置設定為可以被搜尋到
  - 參數：目前所在的Activity

設定為可搜尋

- 設定為可以搜尋到也是屬於系統保護的功能
- 只有Android作業系統能夠直接開啟這個功能
- 外部App都需要啟動預設Dialog形式的Activity

BluetoothChat

建立CLIENT端

顯示已配對裝置

```
List<BluetoothDevice> list =
LocalBluetoothManager.getInstance().getPaired
Devices();
```

- getPairedDevices()
  - 取得已經配對的裝置
  - 已經配對的裝置，且藍芽已經開啟的，可以直接進行連線

顯示已配對裝置

```
mAdapter = new BluetoothListAdapter(this);
mListView.setAdapter(mAdapter);
List<BluetoothDevice> list =
LocalBluetoothManager.getInstance().getPaired
Devices();
mAdapter.setDeviceList(list);
```

- BluetoothAdapter是輔助Library提供的簡易Adapter專門顯示BluetoothDevice的名稱

顯示已配對裝置

```
mAdapter = new BluetoothListAdapter(this);
mListView.setAdapter(mAdapter);
List<BluetoothDevice> list =
LocalBluetoothManager.getInstance().getPaired
Devices();
mAdapter.setDeviceList(list);
```

- 可以將找到的以配對裝置列表，呼叫setDeviceList()直接設定給BluetoothAdapter

搜尋裝置

```
LocalBluetoothManager.getInstance().discoverDevice(mDiscoverListener);
```

- discoveryDevice()
  - 掃描附近已經開啟藍芽可被掃瞄功能的裝置
  - 參數：OnBluetoothDiscoverEventListener
    負責接收掃描的結果
    - 因為掃描**是非同步**的，所以必須使用Listener得知搜尋結果

搜尋裝置

```
private OnBluetoothDiscoverEventListener mDiscoverListener =
new OnBluetoothDiscoverEventListener()
    public void discoveredDevice(BluetoothDevice device, int
rssi) { …… }
    public void discoverFinish() { …… }
};
```

- discoveredDevice()傳回搜尋到的裝置
  - 裝置的搜尋是搜尋到一個就馬上回傳一個
  - 參數1：搜尋到的裝置
  - 參數2：搜尋到的裝置訊號強弱值，可以藉此來判斷距離

搜尋裝置

```
private OnBluetoothDiscoverEventListener mDiscoverListener =
new OnBluetoothDiscoverEventListener()
    public void discoveredDevice(BluetoothDevice device, int
rssi) { …… }
    public void discoverFinish() { …… }
};
```

- discoverFinish()
  - 當搜尋結束時，這個方法就會被呼叫

停止搜尋裝置

```
LocalBluetoothManager.getInstance().stopDiscoverDevice()
```

- 停止搜尋裝置
- 目前使用discoverDevice(), getPairedDevices(), endSession(), pairDevice(), unpairDevice()時，Library都會停止搜尋裝置

配對裝置

```
if (!LocalBluetoothManager.getInstance().isPairedDevice(device)) {
        LocalBluetoothManager.getInstance().pairDevice(device);
}
```

- isPairedDevice()
  - 檢查參數帶入的BluetoothDevice是否為已經配對的裝置
  - 參數：要檢查的BluetoothDevice
- pairDevice()
  - 進行裝置配對
  - 參數：要配對的BluetoothDevice

配對裝置

- 配對裝置也屬於Android作業系統保護的功能
- 所以呼叫pairDevice()時，會出現如下的畫面

配對裝置

```
BluetoothDevice newDevice =
LocalBluetoothManager.getInstance().getDeviceWithLatestStatus(devic
e);
```

- getDeviceWithLatestStatus()
  - 取得特定裝置在OS中最新的狀態，因為APP可能更新的不夠快速
  - 參數：APP端儲存BluetoothDevice

建立CLIENT

```
mHelper = BluetoothConnectionHelper.createClient(UUID.fromString(APP_UUID), device);
mHelper.setMessageReceiver(mListener);
mHelper.connect();
```

- BluetoothConnectionHelper.createClinet()
  - 建立藍芽連線Client
  - 參數1：UUID，要與Server端相同
  - 參數2：Server的BluetoothDevice

建立CLIENT

```
mHelper = BluetoothConnectionHelper.createClient(UUID.fromString(APP_UUID), device);
mHelper.setMessageReceiver(mListener);
mHelper.connect();
```

- setMessageReceiver()
  - 參數：OnBluetoothMessageListener
- connect()
  - 開始與Server連線
  - 記得先設定OnBluetoothMessageListener再呼叫connect

CLIENT傳送訊息

```
if (mHelper.isConnect()) {
      mHelper.sendMessage(content);
}
```

- isConnect()
  - 檢查是否已經連線
- sendMessage()
  - 傳送的內容
  - 參數：要傳送的字串，以英文數字為主

CLIENT結束

```
BluetoothConnectionHelper.close();
```

- close()
  - 關閉client與Server的連線，多半在Activity的onDestroy()呼叫
  - 呼叫後，Server就會收到斷線的通知

總結

- 使用輔助Library，Server的程式碼大約100行、Client大約170行左右就可以完成(含空白行)

- 若要進階使用，依然得去研究Android的程式碼
- 也可以看輔助Library的程式碼，見BluetoothUtility

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

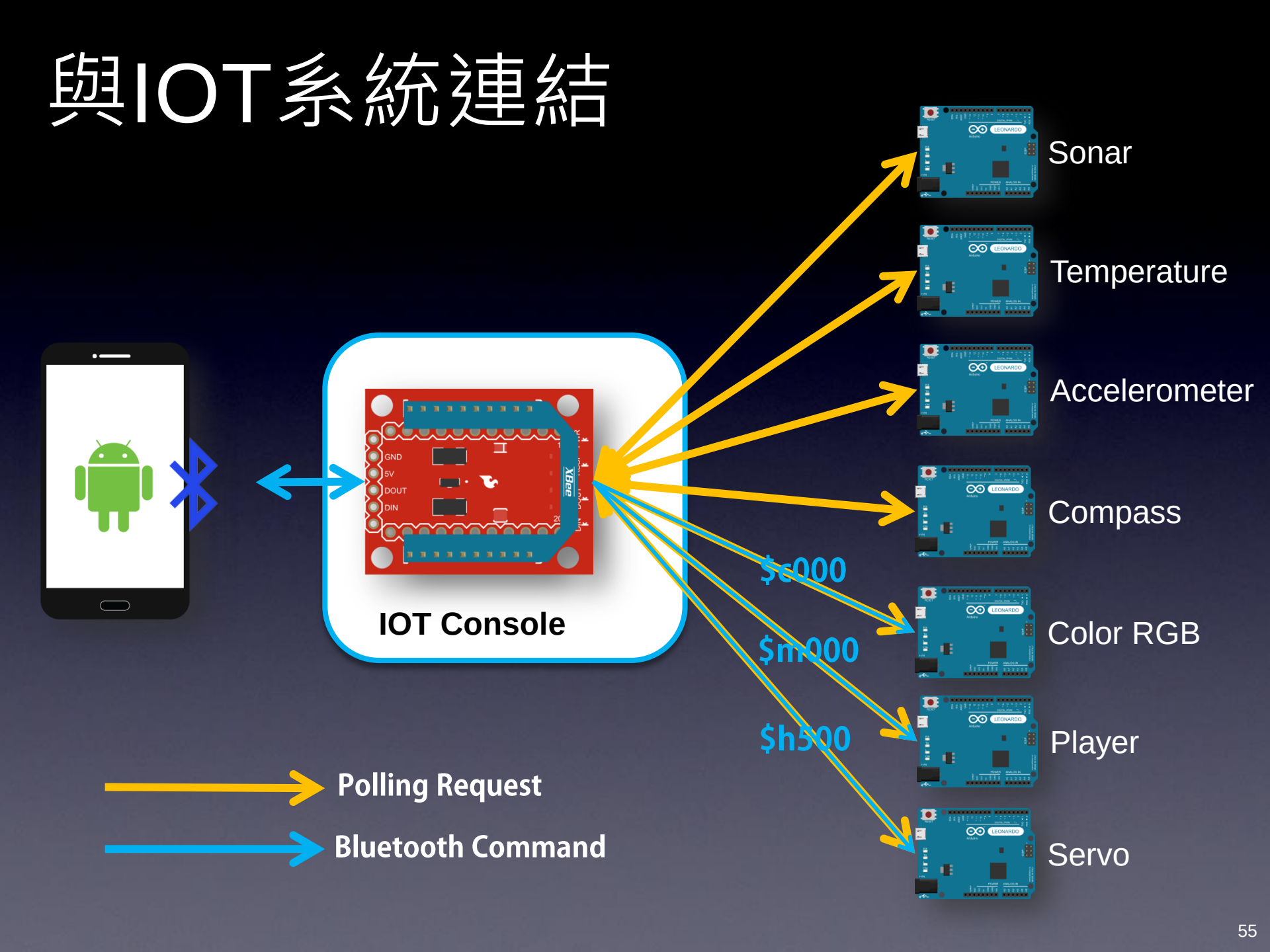
BluetoothArminno

與IOT系統連結

與IOT系統連結

- 藍芽因為有規格在，所以基本的搜尋、連線、資料傳送、接收，處理方式都相同
- 直接延伸使用範例BluetoothChat

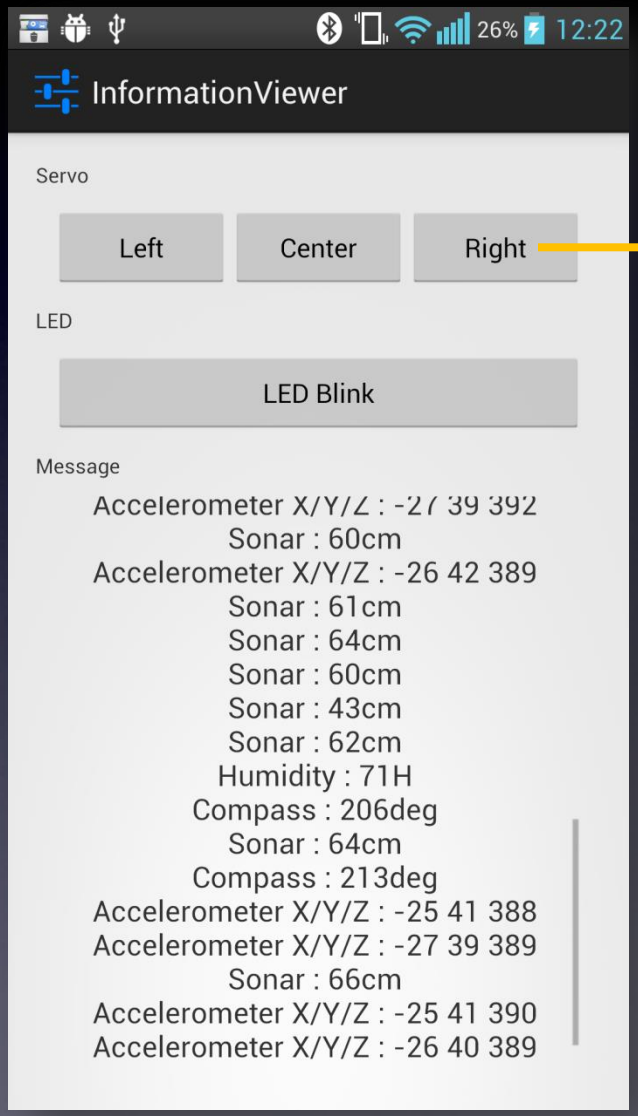
與IOT系統連結


Sonar
Temperature
Accelerometer
Compass
Color RGB
Player
Servo
$e000
$m000
$h500
GND
5V
DOUT
DIN
IOT Console
Polling Request
Bluetooth Command

與IOT系統連結

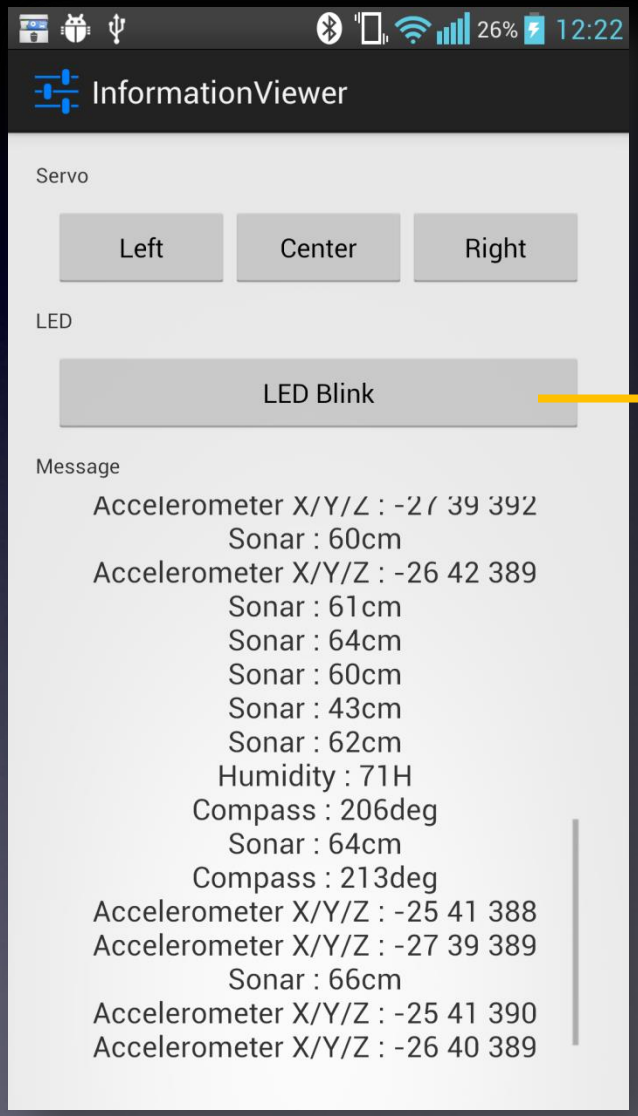
- Color RGB
  - $c000 (使RGB閃爍)
- Servo
  - $h200 (Servo轉左)
  - $h500 (Servo轉中)
  - $h800 (Servo轉右)
- Player
  - $m000 (播放第一首)
  - $m100 (播放第二首)
  - $m200 (播放第三首)

介面與程式碼



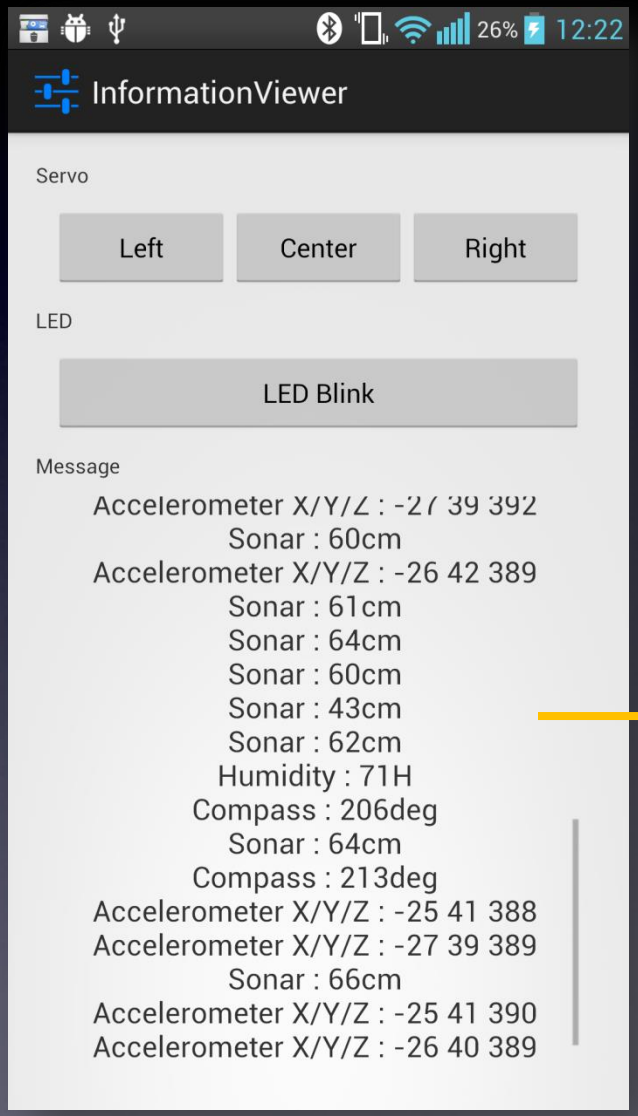

```
if (mHelper.isConnect()) {
    mHelper.sendMessage("$h800");
}
```

介面與程式碼




```
if (mHelper.isConnect()) {
    mHelper.sendMessage("$c000");
}
```

介面與程式碼




```
public void onMessageReceived(
      BluetoothDevice device,
      String message)
{
    if (message != null) {

        mCurrentText += message;
        mReceivedText.setText(
            mCurrentText);

    }
    mScrollView.fullScroll(
        ScrollView.FOCUS_DOWN);

}
```

Q & A