Android 專案說明文件

目前有三種版本的專案，分別為下列三者

1. MTU\_SIZE: 單導極架構，透過跟slave交換MTU\_size，就可以接收slave傳輸的大封包。
2. Connect: 三導極架構(master會依序連線在斷線)，並且有簡單同步功能(同步功能還要修改)。
3. Three\_connect: 三導極架構(依序連線在斷線)，可以接收大封包(不穩定，無法執行太久)。

補充:

下面連結為我們原始參考的sample code:

<https://android.googlesource.com/platform/development/+/cefd49aae65dc85161d08419494071d74ffb982f/samples/BluetoothLeGatt/>

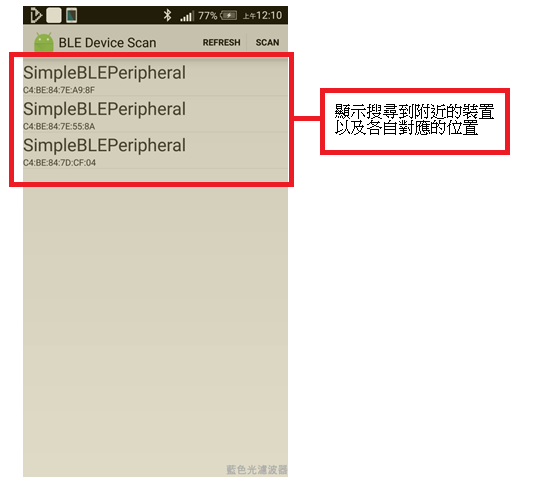
這三份專案程式都是依據這份code去修改的，此外在程式內看到ECG.java與Graph.java的這兩個內容可以不必去理會它，因為這是早期做專題時，撰寫的繪圖功能，如果要使用繪圖功能要再修改。

1. **共通程式碼說明**

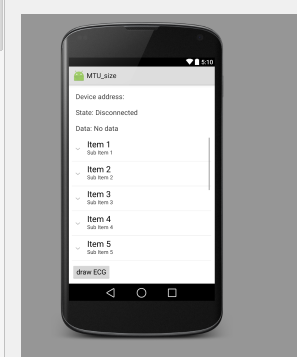
以下將介紹三個專案中程式碼差不多的部分。

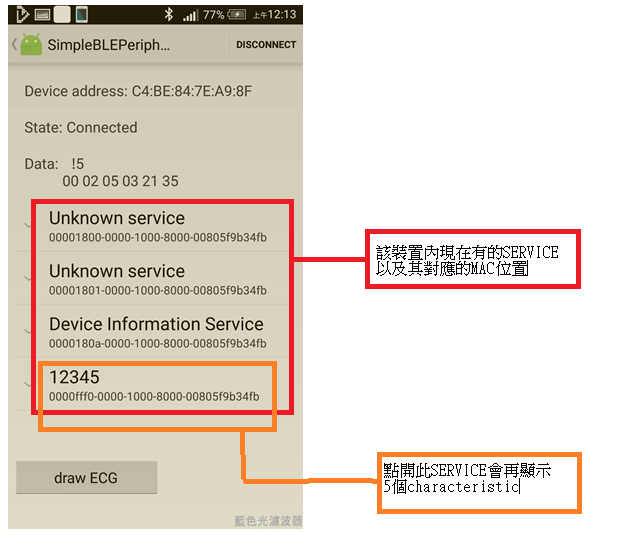
1. **介面程式說明**

1.一開始手機畫面為listitem\_device.xml，在這個介面會搜尋附近是否有可以連接的裝置，並將其顯示出來，由使用者選擇是否要連接。



2.確認自己要連接的裝置並點選後，介面變更為gatt\_services\_characteristic.xml，在這個畫面可以看到裝置是否有正確連線，如正確應會顯示該裝置的MAC位置，以及該裝置提供的SERVICE，目前我們使用的SERVICE名稱為”12345”，裡面包了5個GATT table的characteristic。



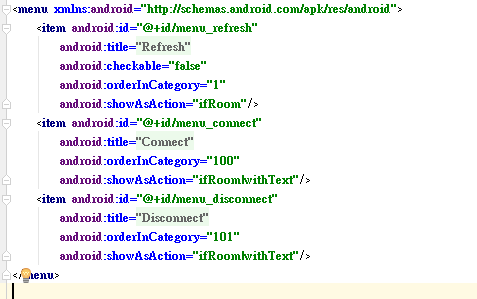


1. **手機選單說明:**

1.menu\_main.xml為一開始listitem\_device.xml介面右上角的按鈕，按鈕為scan時，按下去即會開始搜尋附近的裝置。按鈕為stop時，按下去即會停止搜尋附近的裝置。



2.gatt\_services.xml為gatt\_services\_characteristic.xml介面右上角的按鈕，按鈕為disconnect時，按下去會跟已連線裝置中斷連線。按鈕為connect時，按下去就會連接目前顯示位置的裝置。



1. **參數說明:**

1.string.xml為我們程式內所會用到的字串名稱，原始的名稱都放在這裡。



2.SampleGattAttributes.java為命名service與characteristic的UUID，我們可以將對應的UUID以字串的方式命名並存取它(簡化UUID的字串長度)，使其在手機端顯示的為其名稱。



1. **主程式說明:**

(注意: 有關繪圖程式碼的變數註解可不用理會，因為繪圖的function只針對單導極且傳輸20 bytes短封包的notification傳輸架構，無法用在三導極的傳輸架構)

1. DeviceScanActivity.java功能為掃描附近slave裝置，可以將其與介面listitem\_device.xml來一起看可以較好理解。

2. DeviceControlActivity.java主要功能可以寫在這支骨幹程式，跟slave裝置連線後開始執行如同步、資料傳輸等功能，可以將其與介面gatt\_services\_characteristic.xml一起看可以較好理解。

3. BluetoothLeService.java主要是處理跟點選介面無關的程式，可以視為背景程式的一種，主要負責處理DeviceScanActivity.java和DeviceControlActivity.java內某些與介面無關的內容。

JAVA程式碼說明

1. **DeviceScanActivity.java**

**private static final int *REQUEST\_ENABLE\_BT*** = 1

//確認藍芽系統可以開啟

**private static final long *SCAN\_PERIOD*** = 10000;

//掃描一次為10秒

**public void** onCreate(Bundle savedInstanceState) {……}

//確定手機是否可以支援藍芽系統

**public boolean** onCreateOptionsMenu(Menu menu) {……}

//在有無掃描時，介面鍵為Scan和Stop各自判斷的功能

**public boolean** onOptionsItemSelected(MenuItem item) {……}

//介面鍵為Scan和Stop各自運作的功能

**protected void** onResume(){……}

//確定手機藍芽系統是否有開啟

**protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {……}

//使用者不開啟藍芽系統的情況

**protected void** onListItemClick(ListView l, View v, **int** position, **long** id) {……}

//掃描後如果有找到裝置，將其列出的情況，並且判定當有觸控裝置選單後其將從listitem\_device.xml介面轉換到gatt\_services\_characteristic.xml介面，並且由主程式DeviceScanActivity.java跳入DeviceControlActivity.java

**private void** scanLeDevice(**final boolean** enable) {……}

//掃描情況的判斷

**private class** LeDeviceListAdapter **extends** BaseAdapter {……}

//掃描時找到裝置增加，裝置不見就消失的判定

**(2)DeviceControlActivity.java**

**private final** ServiceConnection **mServiceConnection** = **new** ServiceConnection(){……}

//確認手機與裝置的連接狀態

**private final** BroadcastReceiver **mGattUpdateReceiver** = **new** BroadcastReceiver(){……}

//決定手機與裝置是否要連線，確認底層SERVICE是否建立，判定是否有資料傳遞或接收

**public final** ExpandableListView.OnChildClickListener **servicesListClickListner** =  
**new** ExpandableListView.OnChildClickListener(){……}

//列出裝置的SERVICE，並且確認通道特性為何(READ、NOTIFY)

**public void** onCreate(Bundle savedInstanceState) {……}

//介面畫面的顯示

**public boolean** onCreateOptionsMenu(Menu menu) {……}

//在有無連線時，介面鍵為Connect和Disconnect各自判斷的功能

**public boolean** onOptionsItemSelected(MenuItem item) {……}

//介面鍵為Connect和Disconnect各自運作的功能

**private void** displayData(String data) {……}

//有資料傳遞接收時，在畫面顯示

**private void** displayGattServices(List<BluetoothGattService> gattServices) {……}

//顯示裝置的各個SERVICE與CHARCTERISTIC

**(3)BluetoothLeService.java**

**private final** BluetoothGattCallback **mGattCallback** = **new** BluetoothGattCallback(){……}

//用來判定SERVICE與CHARACTERISTIC的動作狀態

**private void** broadcastUpdate(**final** String action) {……}

//判斷是否又有新的資料傳進來

**private void** broadcastUpdate(**final** String action,**final** BluetoothGattCharacteristic characteristic) {……}

//判斷各個charcteristic要執行的情況

2.-------**else if** (!***UUID\_HEART\_RATE\_MEASUREMENT***.equals(characteristic.getUuid())){……}

//代表點下去是非characteristic4的資料顯示

3.-------**else if** (***UUID\_HEART\_RATE\_MEASUREMENT***.equals(characteristic.getUuid())){……}

//代表點下去是characteristic4的資料顯示

**public boolean** initialize(){……}

//低功號藍芽可以使用

**public boolean** connect(**final** String address) {……}

//根據給定的位置來連接情況

1. **專案功能說明**
2. MTU\_SIZE

此專案的master可以跟slave交換MTU大小。在DeviceControlActivity.java內有一個timer計時器，當它數到三秒時就會自動呼叫BluetoothLeService.java內交換MTU大小的相關function.

1. DeviceControlActivity.java



Timer要在onCreate內呼叫才可執行



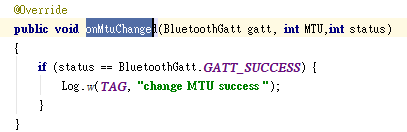
交換MTU大小的function為callMTU()，此function是寫在BluetoothLeService.java內。

1. BluetoothLeService.java

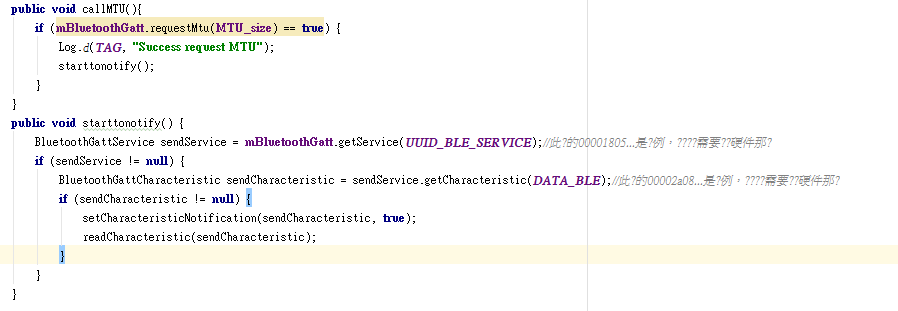
因為我們需要一次傳送248 bytes的資料，因此MTU\_size須設定為251 bytes。MTU\_size設定值公式為傳送的資料大小 + 3 bytes notification header。



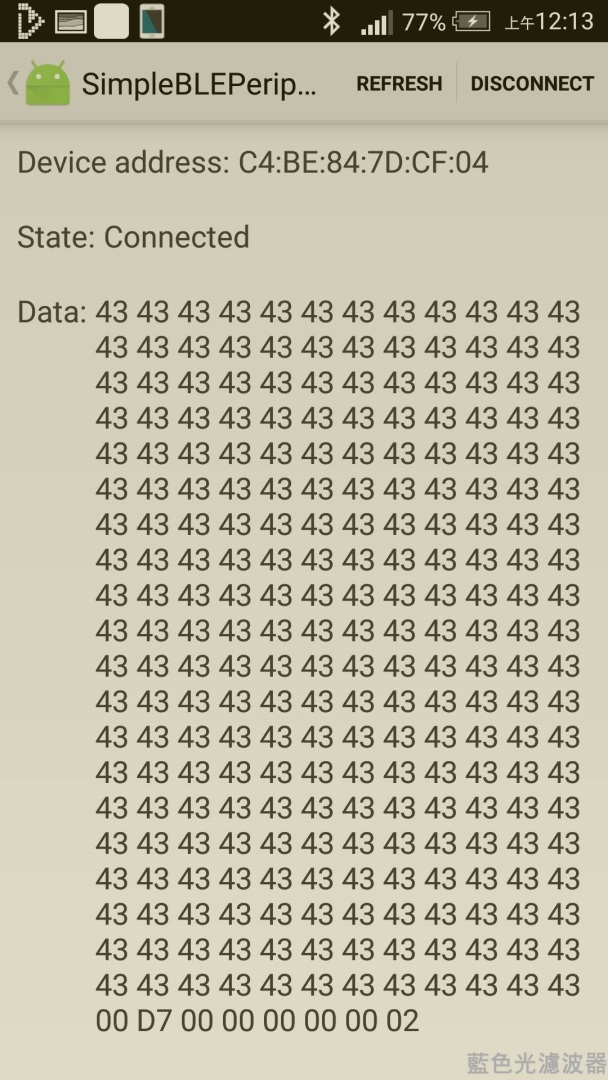
callMTU()的callback function為onMtuChange()，onMtuChange() 的input參數為固定的如下所示，function內可以讓使用者顯示一些log訊息。



下圖為callMTU()的程式碼位於BluetoothLeService.java。

starttonotify()的部分是為了將characteristic4的部分從read狀態修正為notify的狀態，因為原本只接收read response改為接收notification，如此一來其才可以一直持續的接收notification封包的資料。

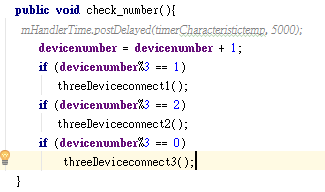
1. 手機顯示結果

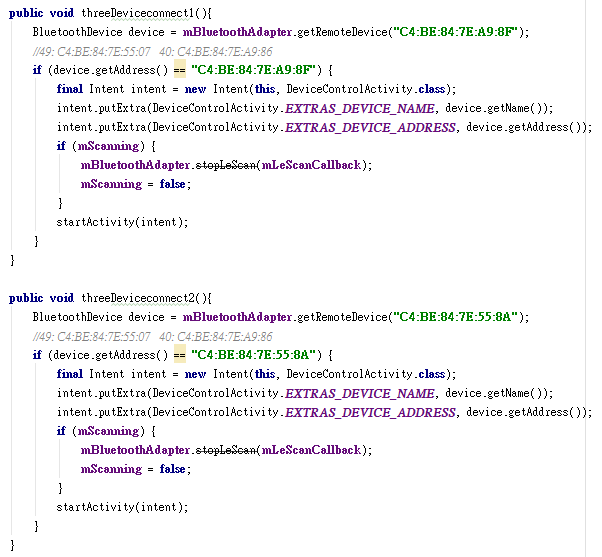


1. Connect

這個部份是我為了DEMO時可以展現三個導極依序連接與斷線，與三個導極的時間同步。我在DeviceScanActivity.java的部分用check\_number方式判斷目前要連接哪一個裝置。在DeviceControlActivity.java內加入兩個timer計時器，一個做每三秒的時間同步，一個做每十秒斷線換下一個裝置的時間計時。

(1)DeviceScanActivity.java



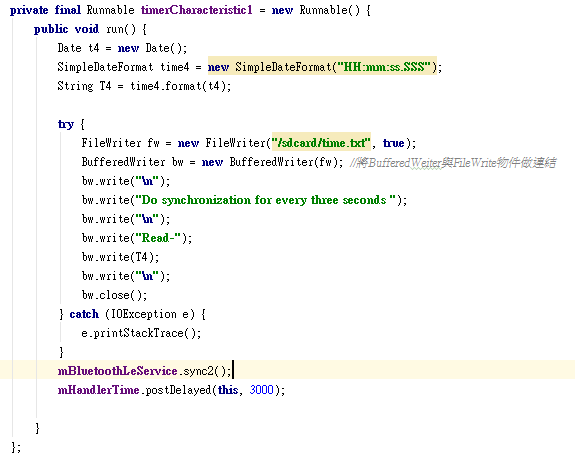


利用threeDeviceconnect1、2、3這三個function把slave三個裝置的MAC位置寫死，此function可判斷是不是我們欲連線的slave裝置。

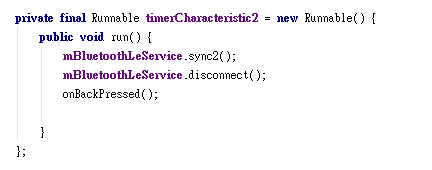
(2)DeviceControlActivity.java



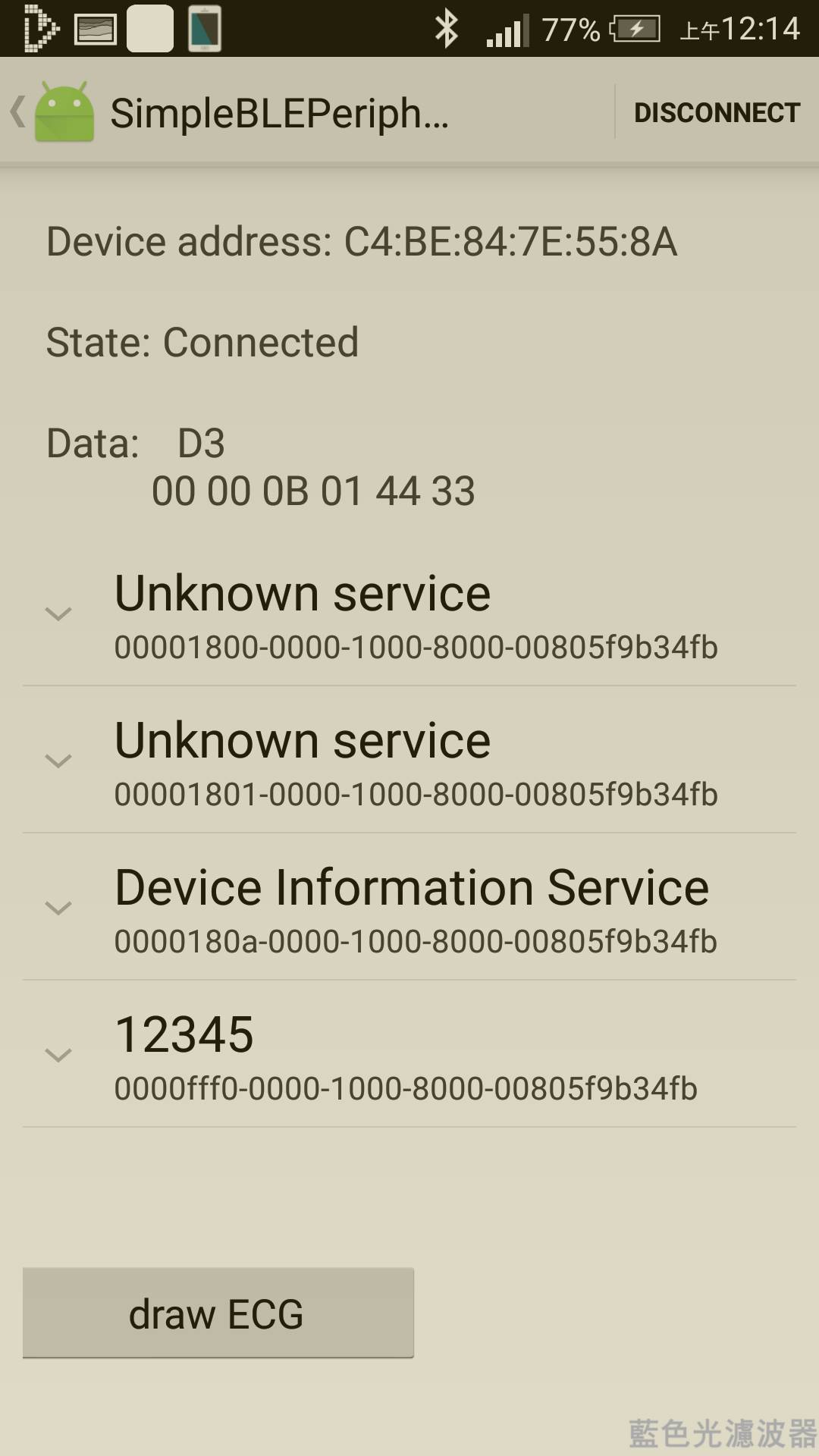
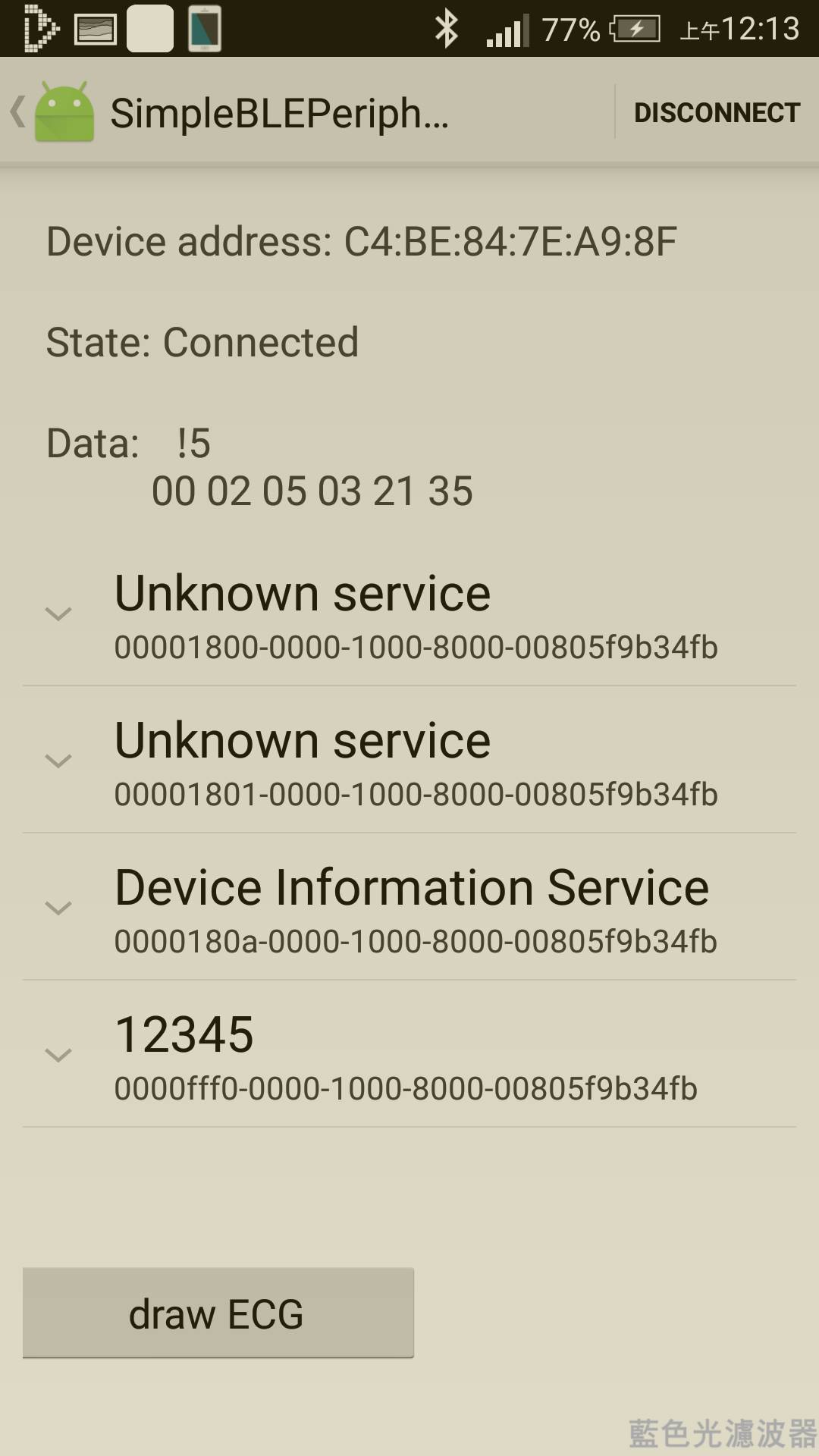
這兩個timer要呼叫在onCreate內才可執行



每三秒做一次時間同步，而時間資料會寫到slave的characteristic2。



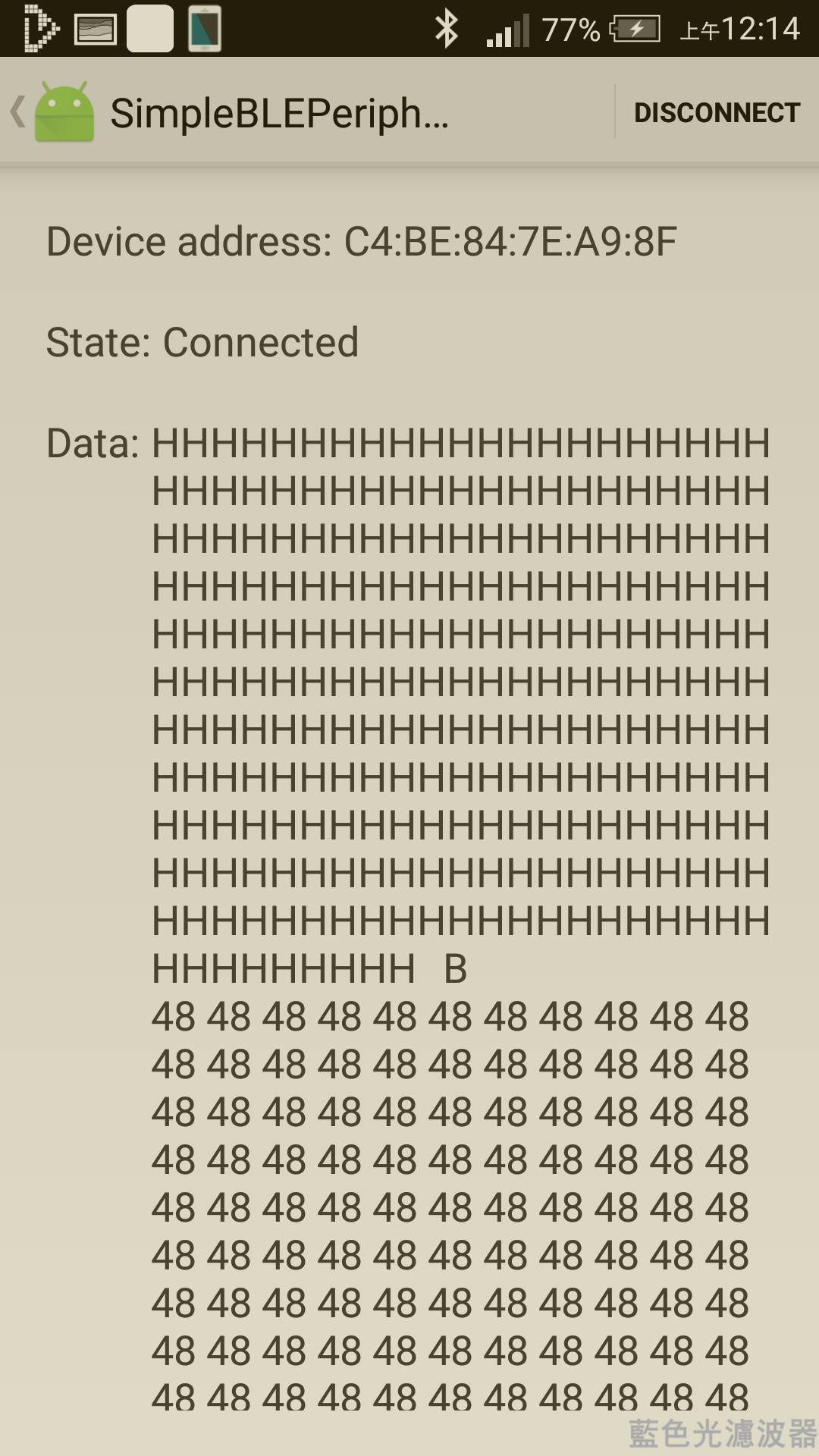
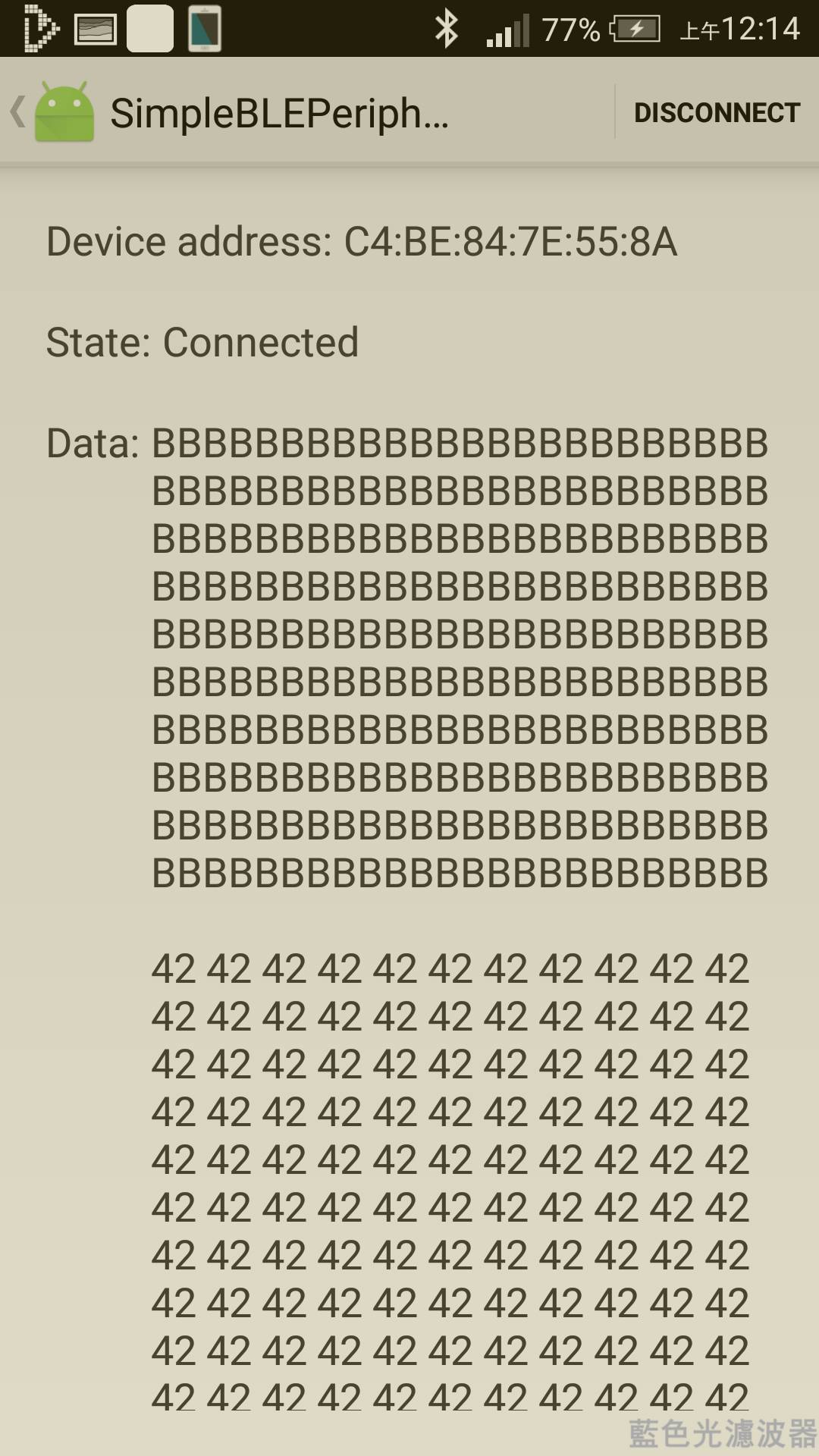
每十秒的計時結束後，就跟slave斷掉連線，並回到DeviceScanActivity.java內去連接下一個裝置。

(3)手機顯示結果

這兩張圖為側錄兩個不同裝置的時間傳送資料顯示結果(由於時間資料的大小跟以前不同，因此同步還需要配合slave程式修改)。

1. Three\_connect

其實此部分可以看成MTU\_SIZE與connect的兩個專案功能的整合，但是尚未完成，且沒有時間同步。在這份檔案內我實際做的情況為在DeviceScanActivity.java的部分用check\_number方式判斷該連接哪一個裝置。在DeviceControlActivity.java內加入兩個timer計時器，一個是連接後數三秒做callMTU()大小的變化，即可一直接收資料，另一個做每十秒斷線換下一個裝置的時間計時。所以內容大致上與前兩個一樣。

手機顯示結果

此為兩個不同裝置，斷線連線接收資料的結果，其問題會在再次連接回到第一個裝置時，就會斷線或無法接收新的資料。

**注意事項:**

此三份專案除了CONNECT專案是使用API18外，另外兩個專案都是使用API21的情況，這是因為有用requestMTU的內建函式，所以要到API21方可支援。不同的API差異只在下面檔案內的修改即可。

AndroidManifest.xml



如果有修正API大小只要修改



這兩個數字內的數值即可。