

第二章

藍芽連線

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
- Server
 - 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

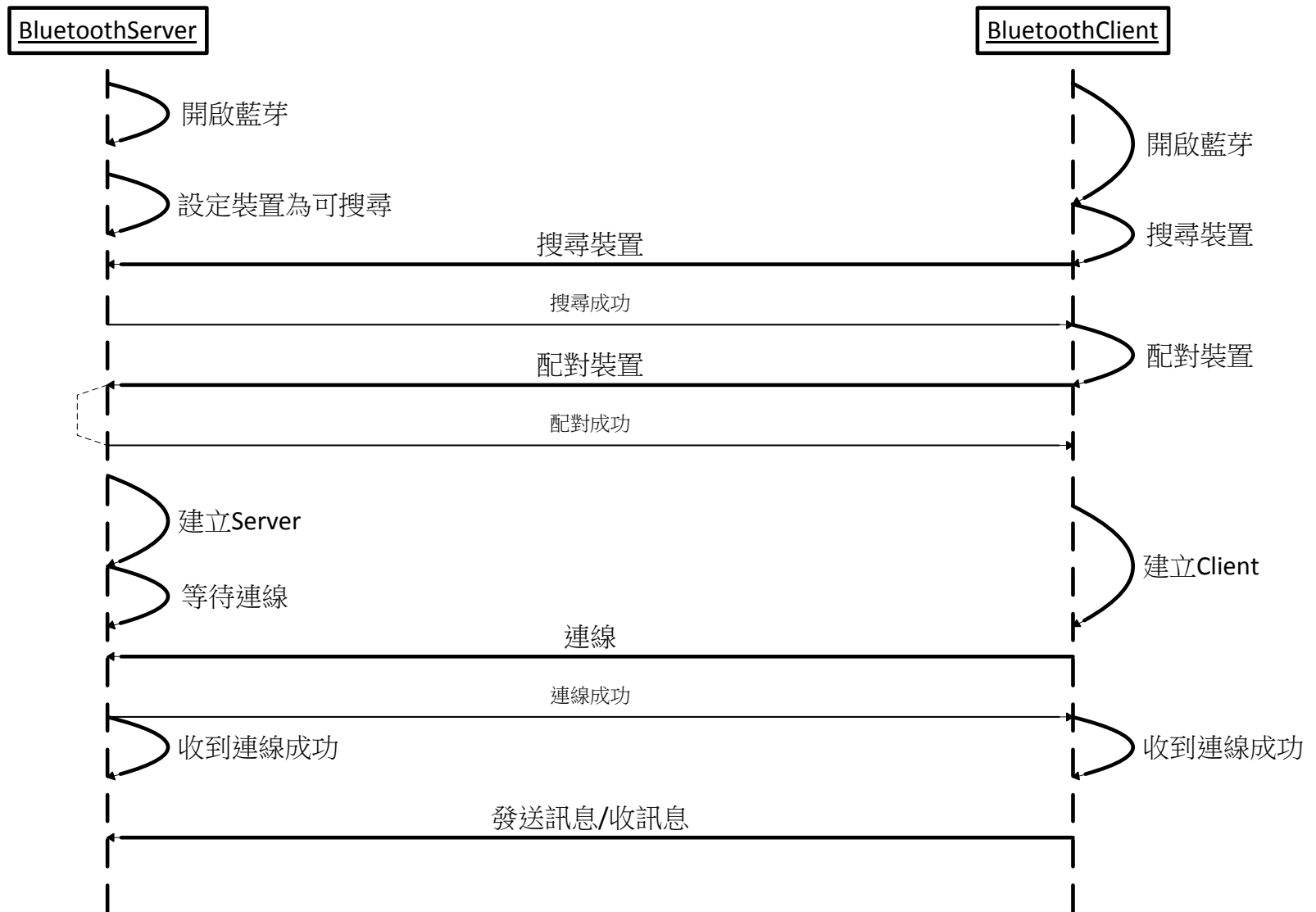
SERVER連線步驟

- 開啟藍芽
- 建立Sever端
- 等待連線
- 待Client連入後，在使用串流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



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開啟藍芽
 - `LocalBluetoothManager.turnOnBluetooth()`
- 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()`

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()`

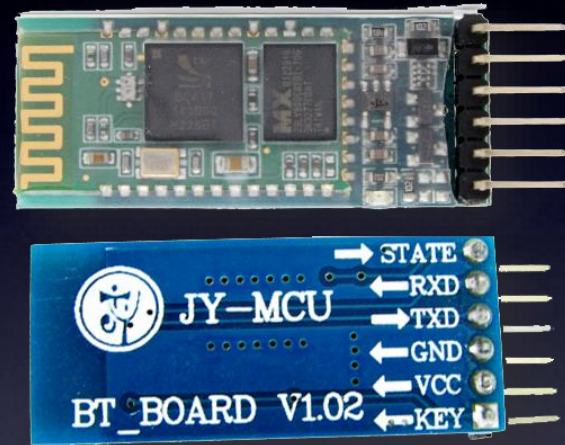
藍芽開關ARDUINO連接之LED

範例說明

- 以手機的藍芽程式控制Arduino Leonardo上的LED開關

藍芽HC-05

- 很普遍的藍芽晶片
- RXD為接收針腳
- TXD為傳送針腳
- GND接地線
- VCC電源
 - VCC33 – 3.3v
 - VCC50 – 5.0v
- KEY 寫入高電壓表示讓晶片進入AT Command模式

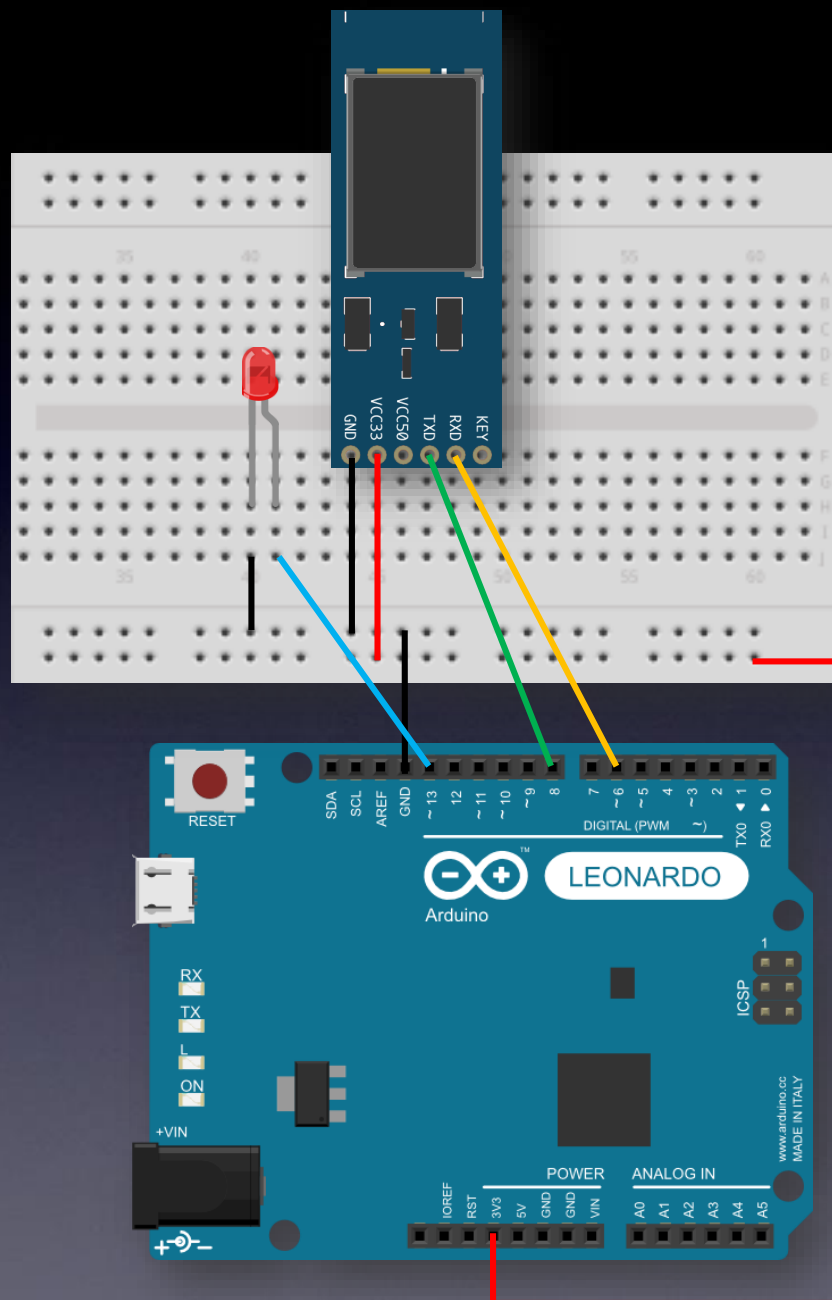


LED

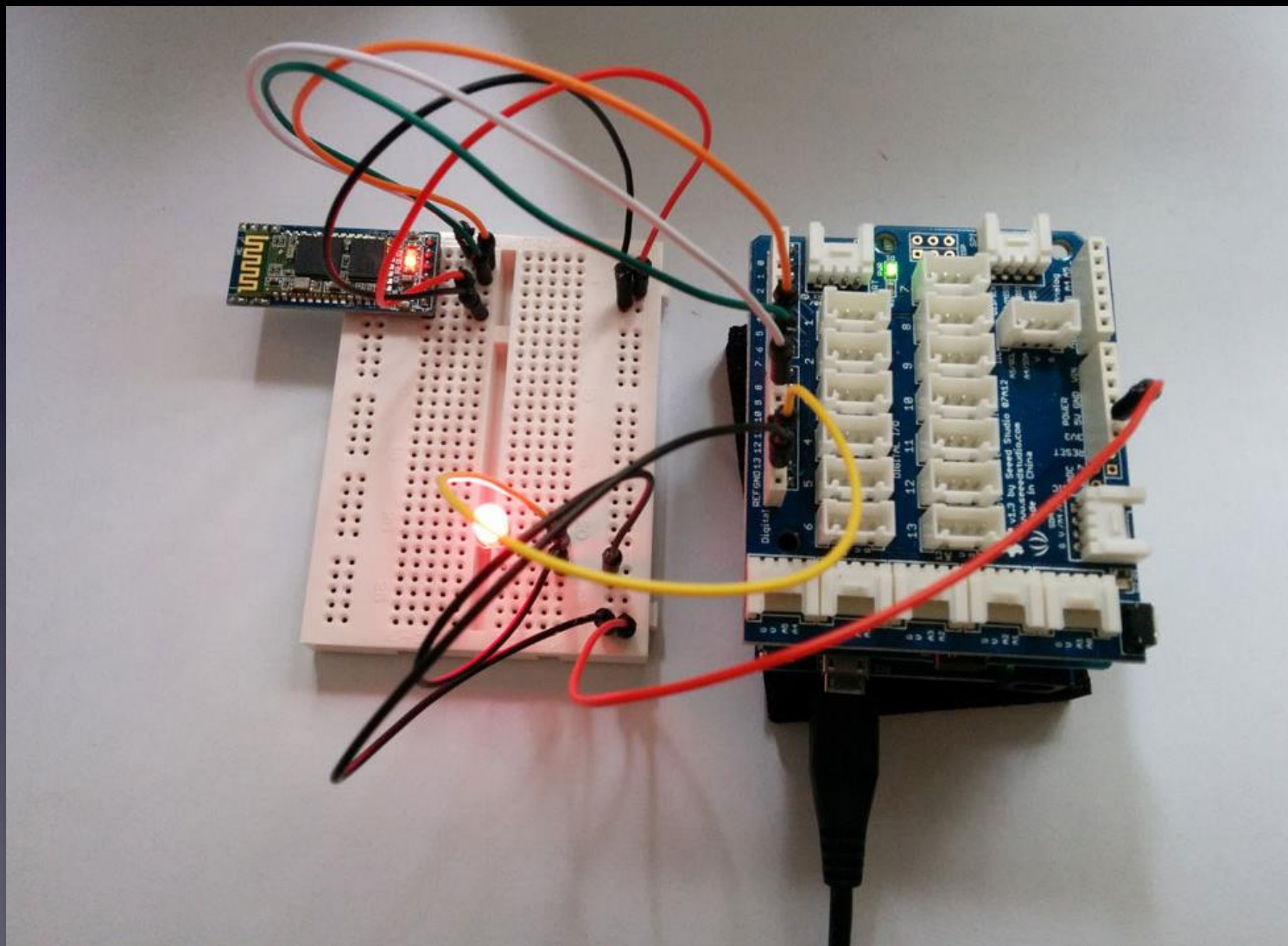
- 一般常見的3mm LED
- 長的針腳接Arduino任一數位輸出孔
- 短的針腳接接地線



電路圖



實機圖



ARDUINO程式碼

```
#include <SoftwareSerial.h>
const int RX = 8;
const int TX = 6;
const int LED = 13;
SoftwareSerial blueToothSerial(RX, TX);
void setup() {
    pinMode(LED, OUTPUT);
    blueToothSerial.begin(38400);
}
void loop() {
    if(blueToothSerial.available()){
        char c = blueToothSerial.read();
        if (c == '1') {
            digitalWrite(LED, HIGH);
        }
        else {
            digitalWrite(LED, LOW);
        }
    }
    delay(500);
}
```

ARDUINO程式碼

```
#include <SoftwareSerial.h>
const int RX = 8;
const int TX = 6;
const int LED = 13;
SoftwareSerial blueToothSerial(RX, TX);
void setup() {
    pinMode(LED, OUTPUT);
    blueToothSerial.begin(38400);
}
void loop() {
    if(blueToothSerial.available()){
        char c = blueToothSerial.read();
        if (c == '1') {
            digitalWrite(LED, HIGH);
        }
        else {
            digitalWrite(LED, LOW);
        }
    }
    delay(500);
}
```

使用SoftwareSerial作為藍芽傳遞資料的管道

ARDUINO程式碼

```
#include <SoftwareSerial.h>
const int RX = 8;
const int TX = 6;
const int LED = 13;
SoftwareSerial blueToothSerial(RX, TX);
void setup() {
    pinMode(LED, OUTPUT);
    blueToothSerial.begin(38400);
}
void loop() {
    if(blueToothSerial.available()){
        char c = blueToothSerial.read();
        if (c == '1') {
            digitalWrite(LED, HIGH);
        }
        else {
            digitalWrite(LED, LOW);
        }
    }
    delay(500);
}
```

Arduino板子的接收腳設定為8
傳送腳設定為6
LED則接在13

ARDUINO程式碼

```
#include <SoftwareSerial.h>
const int RX = 8;
const int TX = 6;
const int LED = 13;
SoftwareSerial blueToothSerial(RX, TX);
void setup() {
    pinMode(LED, OUTPUT);
    blueToothSerial.begin(38400);
}
void loop() {
    if(blueToothSerial.available()){
        char c = blueToothSerial.read();
        if (c == '1') {
            digitalWrite(LED, HIGH);
        }
        else {
            digitalWrite(LED, LOW);
        }
    }
    delay(500);
}
```

建立藍芽傳輸用的Serial

ARDUINO程式碼

```
#include <SoftwareSerial.h>
const int RX = 8;
const int TX = 9;
const int LED = 13;
SoftwareSerial bluetoothSerial(RX, TX);
void setup() {
    pinMode(LED, OUTPUT);
    bluetoothSerial.begin(38400);
}
void loop() {
    if(bluetoothSerial.available()){
        char c = bluetoothSerial.read();
        if (c == '1') {
            digitalWrite(LED, HIGH);
        }
        else {
            digitalWrite(LED, LOW);
        }
    }
    delay(500);
}
```

進入到Arduino初始化的function

ARDUINO程式碼

```
#include <SoftwareSerial.h>
const int RX = 8;
const int TX = 6;
const int LED = 13;
SoftwareSerial blueToothSerial(RX, TX);
void setup() {
    pinMode(LED, OUTPUT);
    blueToothSerial.begin(38400);
}
void loop() {
    if(blueToothSerial.available()){
        char c = blueToothSerial.read();
        if (c == '1') {
            digitalWrite(LED, HIGH);
        }
        else {
            digitalWrite(LED, LOW);
        }
    }
    delay(500);
}
```

設定LED的Pin腳是輸出用

ARDUINO程式碼

```
#include <SoftwareSerial.h>
const int RX = 8;
const int TX = 6;
const int LED = 13;
SoftwareSerial blueToothSerial(RX, TX);
void setup() {
    pinMode(LED, OUTPUT);
    blueToothSerial.begin(38400);
}
void loop() {
    if(blueToothSerial.available()){
        char c = blueToothSerial.read();
        if (c == '1') {
            digitalWrite(LED, HIGH);
        }
        else {
            digitalWrite(LED, LOW);
        }
    }
    delay(500);
}
```

設定SoftwareSerial的連線
速度為38400

ARDUINO程式碼

```
#include <SoftwareSerial.h>
const int RX = 8;
const int TX = 6;
const int LED = 13;
SoftwareSerial blueToothSerial(RX, TX);
void setup() {
    pinMode(LED, OUTPUT);
    blueToothSerial.begin(38400);
}
void loop() {
    if(blueToothSerial.available()){
        char c = blueToothSerial.read();
        if (c == '1') {
            digitalWrite(LED, HIGH);
        }
        else {
            digitalWrite(LED, LOW);
        }
    }
    delay(500);
}
```

HC-05晶片多半預設為38400
可以使用AT command改變

ARDUINO程式碼

```
#include <SoftwareSerial.h>
const int RX = 8;
const int TX = 6;
const int LED = 13;
SoftwareSerial blueToothSerial(RX, TX);
void setup() {
  pinMode(LED, OUTPUT);
  blueToothSerial.begin(38400);
}
void loop() {
  if(blueToothSerial.available()){
    char c = blueToothSerial.read();
    if (c == '1') {
      digitalWrite(LED, HIGH);
    }
    else {
      digitalWrite(LED, LOW);
    }
  }
  delay(500);
}
```

進入到Arduino迴圈的function

ARDUINO程式碼

```
#include <SoftwareSerial.h>
const int RX = 8;
const int TX = 6;
const int LED = 13;
SoftwareSerial blueToothSerial(RX, TX);
void setup() {
    pinMode(LED, OUTPUT);
    blueToothSerial.begin(9600);
}
void loop() {
    if(blueToothSerial.available()){
        char c = blueToothSerial.read();
        if (c == '1') {
            digitalWrite(LED, HIGH);
        }
        else {
            digitalWrite(LED, LOW);
        }
    }
    delay(500);
}
```

當藍芽連接後有接收到資料
available就會是true

ARDUINO程式碼

```
#include <SoftwareSerial.h>
const int RX = 8;
const int TX = 6;
const int LED = 13;
SoftwareSerial blueToothSerial(RX, TX);
void setup() {
    pinMode(LED, OUTPUT);
    blueToothSerial.begin(38400);
}
void loop() {
    if(blueToothSerial.available()) {
        char c = blueToothSerial.read();
        if (c == '1') {
            digitalWrite(LED, HIGH);
        }
        else {
            digitalWrite(LED, LOW);
        }
    }
    delay(500);
}
```

讀取出藍芽所接收到的內容
讀取出來的皆是char

ARDUINO程式碼

```
#include <SoftwareSerial.h>
const int RX = 8;
const int TX = 6;
const int LED = 13;
SoftwareSerial blueToothSerial(RX, TX);
void setup() {
    pinMode(LED, OUTPUT);
    blueToothSerial.begin(38400);
}
void loop() {
    if(blueToothSerial.available()){
        char c = blueToothSerial.read();
        if (c == '1') {
            digitalWrite(LED, HIGH);
        }
        else {
            digitalWrite(LED, LOW);
        }
    }
    delay(500);
}
```

若收到的是1，則讓LED亮起
反之，則讓LED暗下

ARDUINO程式碼

```
#include <SoftwareSerial.h>
const int RX = 8;
const int TX = 6;
const int LED = 13;
SoftwareSerial blueToothSerial(RX, TX);
void setup() {
    pinMode(LED, OUTPUT);
    blueToothSerial.begin(38400);
}
void loop() {
    if(blueToothSerial.available()){
        char c = blueToothSerial.read();
        if (c == '1') {
            digitalWrite(LED, HIGH);
        }
        else {
            digitalWrite(LED, LOW);
        }
    }
    delay(500);
}
```

讓程式暫停一下子
參數是暫停的微妙

ArduinoLEDController

ANDROID前置作業

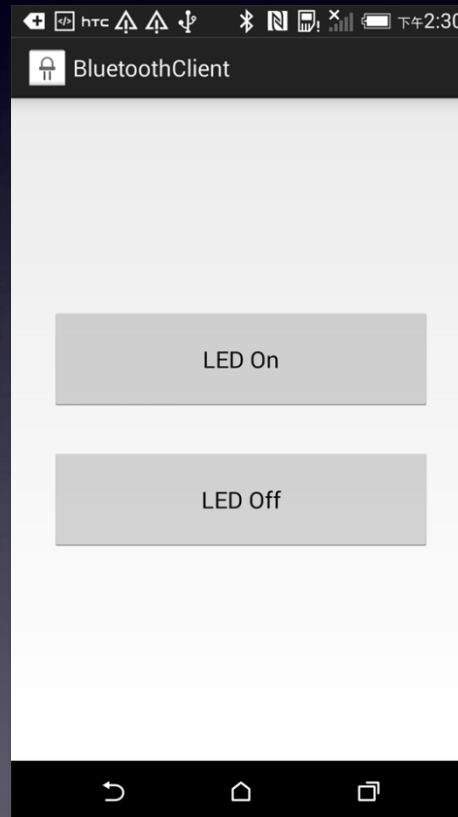
ANDROID送出控制

對應專案
bluetooth/android/
ArduinoLEDController

- 本專案有兩個Activity
- ClientConnectionActivity
 - 負責與Arduino裝置連線送出控制代碼0或1
- DiscoveryActivity
 - 負責找尋附近的裝置
 - HC-05藍芽晶片基本上搜尋都是開啟的
配對的預設密碼為1234

ANDROID送出控制

- 介面更改為兩個大型按鈕，分別是ON與OFF



權限設定

- 打開AndroidManifest.xml

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

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

- 加上上述兩項權限，App才能夠使用藍芽連線

權限設定

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.arduinoledcontroller"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="14" android:targetSdkVersion="21" />
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <application android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity android:name=".DiscoveryActivity"
            android:icon="@drawable/ic_launcher"
            android:screenOrientation="portrait" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".ClientConnectionActivity"
            android:icon="@drawable/ic_launcher"
            android:label="@string/bluetooth_client"
            android:screenOrientation="portrait" />
    </application>
</manifest>
```

準備使用LIBRARY

- 將bluetooth.jar擺進專案的lib資料夾中
- 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

ArduinoLEDController

ANDROID開啟藍芽、掃描及配對裝置

ANDROID 初始化


對應專案
bluetooth/android/
ArduinoLEDController

```
public class DiscoveryActivity extends Activity {  
    private ListView mListView;  
    private BluetoothListAdapter mAdapter;  
    private Button mScanButton;  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.discovery_activity);  
        mListView = (ListView) findViewById(R.id.device_list);  
        mAdapter = new BluetoothListAdapter(this);  
        mListView.setAdapter(mAdapter);  
        mListView.setOnItemClickListener(mItemClickListener);  
        mScanButton = (Button) findViewById(R.id.btn_scan);  
        mScanButton.setOnClickListener(mScanButtonOnClickListener );  
        LocalBluetoothManager.getInstance().startSession(this);  
        if (!LocalBluetoothManager.getInstance().isBluetoothTurnOn()) {  
            LocalBluetoothManager.getInstance().turnOnBluetooth(this);  
        } else {  
            showBindDevices();  
        }  
    }  
}
```

.....(之後還有)

ANDROID 初始化

```
public class DiscoveryActivity extends Activity {  
    private ListView mListView;  
    private BluetoothListAdapter mAdapter;  
    private Button mScanButton;  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.discovery_activity);  
        mListView = (ListView) findViewById(R.id.device_list);  
        mAdapter = new BluetoothListAdapter(this);  
        mListView.setAdapter(mAdapter);  
        mListView.setOnItemClickListener(mItemClickListener);  
        mScanButton = (Button) findViewById(R.id.btn_scan);  
        mScanButton.setOnClickListener(mScanButtonOnClickListener);  
        LocalBluetoothManager.getInstance().startSession(this);  
        if (!LocalBluetoothManager.getInstance().isBluetoothTurnOn()) {  
            LocalBluetoothManager.getInstance().turnOnBluetooth(this);  
        } else {  
            showBindDevices();  
        }  
    }  
}
```



.....(之後還有)

ANDROID 初始化

準備ListView和使用的Adapter

```
public class DiscoveryActivity extends AppCompatActivity {
    private ListView mListView;
    private BluetoothListAdapter mAdapter;
    private Button mScanButton;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.discovery_activity);
        mListView = (ListView) findViewById(R.id.device_list);
        mAdapter = new BluetoothListAdapter(this);
        mListView.setAdapter(mAdapter);
        mListView.setOnItemClickListener(mItemClickListener);
        mScanButton = (Button) findViewById(R.id.btn_scan);
        mScanButton.setOnClickListener(mScanButtonOnClickListener );
        LocalBluetoothManager.getInstance().startSession(this);
        if (!LocalBluetoothManager.getInstance().isBluetoothTurnOn()) {
            LocalBluetoothManager.getInstance().turnOnBluetooth(this);
        } else {
            showBindDevices();
        }
    }
}
```

.....(之後還有)

ANDROID 初始化

```
public class DiscoveryActivity extends Activity {
    private ListView mListView;
    private BluetoothListAdapter mAdapter;
    private Button mScanButton;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.discovery_activity);
        mListView = (ListView) findViewById(R.id.device_list);
        mAdapter = new BluetoothListAdapter(this);
        mListView.setAdapter(mAdapter);
        mListView.setOnItemClickListener(mItemClickListener);
        mScanButton = (Button) findViewById(R.id.btn_scan);
        mScanButton.setOnClickListener(mScanButtonOnClickListener );
        LocalBluetoothManager.getInstance().startSession(this);
        if (!LocalBluetoothManager.getInstance().isBluetoothTurnOn()) {
            LocalBluetoothManager.getInstance().turnOnBluetooth(this);
        } else {
            showBindDevices();
        }
    }
}
```

準備點下後會開始做藍芽掃描的
Button

.....(之後還有)

ANDROID 初始化

```
public class DiscoveryActivity extends Activity {  
    private ListView mListView;  
    private BluetoothListAdapter mAdapter;  
    private Button mScanButton;  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.discovery_activity);  
        mListView = (ListView) findViewById(R.id.device_list);  
        mAdapter = new BluetoothListAdapter(this);  
        mListView.setAdapter(mAdapter);  
        mListView.setOnItemClickListener(mItemClickListener);  
        mScanButton = (Button) findViewById(R.id.btn_scan);  
        mScanButton.setOnClickListener(mScanButtonOnClickListener);  
        LocalBluetoothManager.getInstance().startSession(this);  
        if (!LocalBluetoothManager.getInstance().isBluetoothTurnOn()) {  
            LocalBluetoothManager.getInstance().turnOnBluetooth(this);  
        } else {  
            showBindDevices();  
        }  
    }  
}
```

找出ListView並且建立Adapter
設定ListView需要的事件接收器

.....(之後還有)

ANDROID 初始化

```
public class DiscoveryActivity extends Activity {  
    private ListView mListView;  
    private BluetoothListAdapter mAdapter;  
    private Button mScanButton;  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.discovery_activity);  
        mListView = (ListView) findViewById(R.id.dev_list);  
        mAdapter = new BluetoothListAdapter();  
        mListView.setAdapter(mAdapter);  
        mListView.setOnItemClickListener(mItemClickListener);  
        mScanButton = (Button) findViewById(R.id.btn_scan);  
        mScanButton.setOnClickListener(mScanButtonOnClickListener);  
        LocalBluetoothManager.getInstance().startSession(this);  
        if (!LocalBluetoothManager.getInstance().isBluetoothTurnOn()) {  
            LocalBluetoothManager.getInstance().turnOnBluetooth(this);  
        } else {  
            showBindDevices();  
        }  
    }  
}
```

找出Button並且設定Button需要
的事件接收器

.....(之後還有)

ANDROID 初始化

```
public class DiscoveryActivity extends Activity {  
    private ListView mListView;  
    private BluetoothListAdapter mAdapter;  
    private Button mScanButton;  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.discovery_activity);  
        mListView = (ListView) findViewById(R.id.device_list);  
        mAdapter = new BluetoothListAdapter(this);  
        mListView.setAdapter(mAdapter);  
        mListView.setOnItemClickListener(  
            mScanButton = (Button) findViewById(R.id.button_scan);  
            mScanButton.setOnClickListener(  
                LocalBluetoothManager.getInstance().startSession(this);  
                if (!LocalBluetoothManager.getInstance().isBluetoothTurnOn()) {  
                    LocalBluetoothManager.getInstance().turnOnBluetooth(this);  
                } else {  
                    showBindDevices();  
                }  
            }  
        }  
    }  
}
```

呼叫startSession讓
LocalBluetoothManager初始化

.....(之後還有)

ANDROID 初始化

```
public class DiscoveryActivity extends Activity {  
    private ListView mListView;  
    private BluetoothListAdapter mAdapter;  
    private Button mScanButton;  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.discovery_activity);  
        mListView = (ListView) findViewById(R.id.device_list);  
        mAdapter = new BluetoothListAdapter(this);  
        mListView.setAdapter(mAdapter);  
        mListView.setOnItemClickListener(  
            mScanButton = (Button) findViewById(R.id.scan_button);  
            mScanButton.setOnClickListener(  
                LocalBluetoothManager.getInstance().startSession(this);  
                if (!LocalBluetoothManager.getInstance().isBluetoothTurnOn()) {  
                    LocalBluetoothManager.getInstance().turnOnBluetooth(this);  
                } else {  
                    showBindDevices();  
                }  
            }  
        );  
    }  
}
```

使用isBluetoothTurnOn檢查
目前藍芽是否開啟

.....(之後還有)

ANDROID 初始化

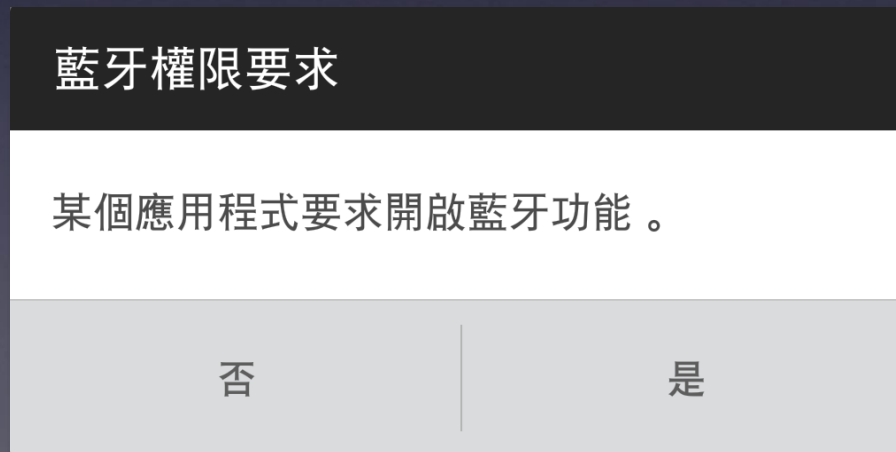
```
public class DiscoveryActivity extends Activity {  
    private ListView mListView;  
    private BluetoothListAdapter mAdapter;  
    private Button mScanButton;  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.discovery_activity);  
        mListView = (ListView) findViewById(R.id.device_list);  
        mAdapter = new BluetoothListAdapter(this);  
        mListView.setAdapter(mAdapter);  
        mListView.setOnItemClickListener(mItemClickListener);  
        mScanButton = (Button) findViewById(R.id.scan);  
        mScanButton.setOnClickListener(mScanButtonClickListener);  
        LocalBluetoothManager.getInstance().startSession(this);  
        if (!LocalBluetoothManager.getInstance().isBluetoothTurnOn()) {  
            LocalBluetoothManager.getInstance().turnOnBluetooth(this);  
        } else {  
            showBindDevices();  
        }  
    }  
}
```

若檢查沒開啟
呼叫turnOnBluetooth來開啟

.....(之後還有)

ANDROID 初始化

- `LocalBluetoothManager.getInstance().turnOnBluetooth(this)`
 - 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



ANDROID 初始化

```
public class DiscoveryActivity extends Activity {  
    private ListView mListView;  
    private BluetoothListAdapter mAdapter;  
    private Button mScanButton;  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.discovery_activity);  
        mListView = (ListView) findViewById(R.id.device_list);  
        mAdapter = new BluetoothListAdapter(this);  
        mListView.setAdapter(mAdapter);  
        mListView.setOnItemClickListener(mItemClickListener);  
        mScanButton = (Button) findViewById(R.id.btn_scan);  
        mScanButton.setOnClickListener(mScanButtonOnClickListener );  
        LocalBluetoothManager.getInstance().startSession(this);  
        if (!LocalBluetoothManager.getInstance().isBluetoothTurnOn()) {  
            LocalBluetoothManager.getInstance().turnOnBluetooth(this);  
        } else {  
            showBindDevices();  
        }  
    }  
}
```

已經開啟了
就顯示已經綁定過的藍芽裝置

.....(之後還有)

ANDROID 初始化

```
@Override  
public void onDestroy() {  
    super.onDestroy();  
    LocalBluetoothManager.getInstance().endSession();  
}
```

記得在Activity的onDestroy呼叫
endSession來關閉藍芽和釋放資源

ANDROID 開啟藍芽

- 前頁投影片提到藍芽開啟是受系統保護的功能
- 所以開啟藍芽後系統會透過特殊的方式通知我們就是在Activity的方法onActivityResult()

ANDROID 開啟藍芽

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

ANDROID 開啟藍芽

詢問使用者藍芽開啟畫面關閉後
系統會自動呼叫這個方法

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

ANDROID 開啟藍芽

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

在此處再次檢查藍芽是不是開啟的

ANDROID 開啟藍芽

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

若不是則直接關閉Activity

ANDROID 開啟藍芽

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

若開啟成功
顯示已經配對的藍芽裝置

ANDROID 顯示已配對裝置

```
private void showBindDevices() {  
    List<BluetoothDevice> list =  
LocalBluetoothManager.getInstance().getPairedDevices();  
    if (mAdapter != null) {  
        mAdapter.setDeviceList(list);  
    }  
}
```

ANDROID 顯示已配對裝置

```
private void showBindDevices() {  
    List<BluetoothDevice> list =  
    LocalBluetoothManager.getInstance().getPairedDevices();  
    mAdapter.setDeviceList(list);  
}
```

getPairedDevices可以得到
目前已經配對裝置的List

ANDROID 顯示已配對裝置

```
private void showBindDevices() {  
    List<BluetoothDevice> list =  
    LocalBluetoothManager.getInstance().getPairedDevices();  
    mAdapter.setDeviceList(list);  
}
```

直接使用BluetoothListAdapter
的setDeviceList方法讓畫面呈現

ANDROID 搜尋藍芽裝置

```
private OnClickListener mScanButtonOnClickListener =  
new OnClickListener()  
{  
    @Override  
    public void onClick(View v) {  
        mScanButton.setEnabled(false);  
        LocalBluetoothManager.getInstance()  
            .discoverDevice(mDiscoverListener);  
    }  
};
```

ANDROID 搜尋藍芽裝置

```
private OnClickListener mScanButtonOnClickListener =  
new OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        mScanButton.setEnabled(false);  
        LocalBluetoothManager.getInstance()  
            .discoverDevice(mDiscoverListener);  
    }  
};
```

使用View的setEnabled(false)停用按鈕
免得在掃描裝置同時又被按下第二次

ANDROID 搜尋藍芽裝置

```
private OnClickListener mScanButtonOnClickListener =  
new OnClickListener()  
{  
    @Override  
    public void onClick(View v) {  
        mScanButton.setEnabled(false);  
        LocalBluetoothManager.getInstance()  
            .discoverDevice(mDiscoverListener);  
    }  
};
```

呼叫discoverDevice
讓手機進行藍芽裝置掃描

ANDROID 搜尋藍芽裝置

```
private OnClickListener mScanButtonOnClickListener =  
new OnClickListener()  
{  
    @Override  
    public void onClick(View v) {  
        mScanButton.setEnabled(false);  
        LocalBluetoothManager.getInstance()  
            .discoverDevice(mDiscoverListener);  
    }  
};
```

藍芽裝置掃描是非同步的工作
所以設定事件接收器
負責接收掃描到的結果

ANDROID 搜尋藍芽裝置

```
private OnBluetoothDiscoverEventListener mDiscoverListener = new
OnBluetoothDiscoverEventListener() {
    @Override
    public void discoverFinish() {
        Toast.makeText(
            DiscoveryActivity.this,
            "discoverFinish",
            Toast.LENGTH_LONG).show();
        mScanButton.setEnabled(true);
    }
    @Override
    public void discoveredDevice(BluetoothDevice device, int rssi) {
        mAdapterter.addItem(device);
    }
};
```

ANDROID 搜尋藍芽裝置

```
private OnBluetoothDiscoverEventListener mDiscoverListener = new
OnBluetoothDiscoverEventListener() {
    @Override
    public void discoverFinish() {
        Toast.makeText(
            DiscoveryActivity.this,
            "discoverFinish",
            Toast.LENGTH_LONG).show();
        mScanButton.setEnabled(true);
    }
    @Override
    public void discoveredDevice(BluetoothDevice device, int rssi) {
        mAdapterter.addItem(device);
    }
};
```

OnBluetoothDiscoverEventListener
是掃描藍芽專用的事件接收器

ANDROID 搜尋藍芽裝置

discoverFinish是第一種事件
表示掃描藍芽裝置結束

```
private OnBluetoothDiscoverListener discoverListener = new
OnBluetoothDiscoverListener() {
    @Override
    public void discoverFinish() {
        Toast.makeText(
            DiscoveryActivity.this,
            "discoverFinish",
            Toast.LENGTH_LONG).show();
        mScanButton.setEnabled(true);
    }
    @Override
    public void discoveredDevice(BluetoothDevice device, int rssi) {
        mAdapterter.addItem(device);
    }
};
```


ANDROID 搜尋藍芽裝置

```
private OnBluetoothDiscoverEventListener mDiscoverListener = new
OnBluetoothDiscoverEventListener() {
    @Override
    public void discoverFinish() {
        Toast.makeText(
            DiscoveryActivity.this,
            "discoverFinish",
            Toast.LENGTH_LONG).show();
        mScanButton.setEnabled(true);
    }
    @Override
    public void discoveredDevice(BluetoothDevice device, int rssi) {
        mAdapterter.addItem(device);
    }
};
```

顯示Toast並且將
掃描裝置的按鈕重新啟用

ANDROID 搜尋藍芽裝置

```
private OnBluetoothDiscoverEventListener mDiscoverListener = new
OnBluetoothDiscoverEventListener() {
    @Override
    public void discoverFinish() {
        Toast.makeText(
            DiscoveryActivity.this,
            "discoverFinish",
            Toast.LENGTH_SHORT).show();
        mScanButton.setEnabled(true);
    }
    @Override
    public void discoveredDevice(BluetoothDevice device, int rssi) {
        mAdapterter.addItem(device);
    }
};
```



第二種事件，掃描到裝置
當藍芽掃描到一個裝置馬上就呼叫
這個方法一次

ANDROID 搜尋藍芽裝置

```
private OnBluetoothDiscoverEventListener mDiscoverListener = new
OnBluetoothDiscoverEventListener() {
    @Override
    public void discoverFinish() {
        Toast.makeText(
            DiscoveryActivity.this,
            "discoverFinish",
            Toast.LENGTH_LONG).show();
        mScanButton.setEnabled(true);
    }
    @Override
    public void discoveredDevice(BluetoothDevice device, int rssi) {
        mAdapterter.addItem(device);
    }
};
```

參數1是掃描到的裝置的資訊
參數2是該裝置的訊號強弱值

ANDROID 搜尋藍芽裝置

```
private OnBluetoothDiscoverEventListener mDiscoverListener = new
OnBluetoothDiscoverEventListener() {
    @Override
    public void discoverFinish() {
        Toast.makeText(
            DiscoveryActivity.this,
            "discoverFinish",
            Toast.LENGTH_LONG).show();
        mScanButton.setEnabled(true);
    }
    @Override
    public void discoveredDevice(BluetoothDevice device, int rssi) {
        mAdapter.addItem(device);
    }
};
```

將掃描到的裝置加入Adapter中
ListView就會呈現到畫面上

ANDROID 配對裝置與連線

- 當ListView呈現出掃描到的裝置，若裝置沒有配對過，是「不能進行連線」的
- 所以ListView的裝置被點到時，首先必須檢查該裝置有沒有配對過
- 若沒配對過，就要進行配對
- 已配對過，則進行連線

ANDROID 配對裝置與連線

```
private OnItemClickListener mItemClickListener = new OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id)  
    {  
        BluetoothDevice device = (BluetoothDevice) mAdapter.getItem(position);  
        BluetoothDevice newDevice =  
LocalBluetoothManager.getInstance().getDeviceWithLatestStatus(device);  
        if (!LocalBluetoothManager.getInstance().isPairedDevice(newDevice)) {  
            LocalBluetoothManager.getInstance().pairDevice(newDevice);  
        } else {  
            Intent intent = new Intent(DiscoveryActivity.this,  
ClientConnectionActivity.class);  
            intent.putExtra("device", newDevice);  
            startActivity(intent);  
        }  
    }  
};
```

ANDROID 配對裝置與連線

```
private OnItemClickListener mItemClickListener = new OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id)  
    {  
        BluetoothDevice device = (BluetoothDevice) mAdapter.getItem(position);  
        BluetoothDevice newDevice =  
LocalBluetoothManager.getInstance().getDeviceWithLatestStatus(device);  
        if (!LocalBluetoothManager.getInstance().isPairedDevice(newDevice)) {  
            LocalBluetoothManager.getInstance().pairDevice(newDevice);  
        } else {  
            Intent intent = new Intent(DiscoveryActivity.this,  
ClientConnectionActivity.class);  
            intent.putExtra("device", newDevice);  
            startActivity(intent);  
        }  
    }  
};
```

取得目前點選到的藍芽資訊

ANDROID 配對裝置與連線

```
private OnItemClickListener mItemClickListener = new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<BluetoothDevice> parent, View view, int position, long id)
    {
        BluetoothDevice device = (BluetoothDevice) mAdapter.getItem(position);
        BluetoothDevice newDevice =
        LocalBluetoothManager.getInstance().getDeviceWithLatestStatus(device);
        if (!LocalBluetoothManager.getInstance().isPairedDevice(newDevice)) {
            LocalBluetoothManager.getInstance().pairDevice(newDevice);
        } else {
            Intent intent = new Intent(DiscoveryActivity.this,
            ClientConnectionActivity.class);
            intent.putExtra("device", newDevice);
            startActivity(intent);
        }
    }
};
```

取得特定藍芽裝置在OS中最新的狀態
因為APP可能更新的不夠快速

ANDROID 配對裝置與連線

```
private OnItemClickListener mItemClickListener = new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id)
    {
        BluetoothDevice device = (BluetoothDevice) parent.getItem(position);
        BluetoothDevice newDevice = LocalBluetoothManager.getInstance().getDeviceWithLatestStatus(device);
        if (!LocalBluetoothManager.getInstance().isPairedDevice(newDevice)) {
            LocalBluetoothManager.getInstance().pairDevice(newDevice);
        } else {
            Intent intent = new Intent(DiscoveryActivity.this,
            ClientConnectionActivity.class);
            intent.putExtra("device", newDevice);
            startActivity(intent);
        }
    }
};
```

檢查該裝置是不是已經配對過

ANDROID 配對裝置與連線

```
private OnItemClickListener mItemClickListener = new OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id)
    {
        BluetoothDevice device = (BluetoothDevice) mAdapter.getItem(position);
        BluetoothDevice newDevice = 呼叫pairDevice進行配對
        LocalBluetoothManager.getInstance().getDeviceWithLatestStatus(device);
        if (!LocalBluetoothManager.getInstance().isPairedDevice(newDevice)) {
            LocalBluetoothManager.getInstance().pairDevice(newDevice);
        } else {
            Intent intent = new Intent(DiscoveryActivity.this,
            ClientConnectionActivity.class);
            intent.putExtra("device", newDevice);
            startActivity(intent);
        }
    }
};
```

ANDROID 配對裝置與連線

- 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



藍牙配對要求

裝置
H-C-2010-06-01

通常為 0000 或 1234

☐ PIN 碼含有字母或符號

您可能也必須在另一個裝置上輸入這個 PIN。

裝置配對完成後，連線時即可存取您的聯絡人和通話紀錄。

取消 確定

ANDROID 配對裝置與連線

```
private OnItemClickListener mItemClickListener = new OnItemClickListener() {  
    @Override  
    public void onItemClick(AdapterView<?> parent, View view, int position, long id)  
    {  
        BluetoothDevice device = (BluetoothDevice) mAdapter.getItem(position);  
        BluetoothDevice newDevice =  
LocalBluetoothManager.getInstance().getDeviceWithLatestStatus(device);  
        if (!LocalBluetoothManager.getInstance().isPairedDevice(newDevice)) {  
            LocalBluetoothManager.getInstance().pairDevice(newDevice);  
        } else {  
            Intent intent = new Intent(DiscoveryActivity.this,  
ClientConnectionActivity.class);  
            intent.putExtra("device", newDevice);  
            startActivity(intent);  
        }  
    }  
};
```

若是已經配對過的裝置
將裝置資訊放入Intent
切換至ClientConnectionActivity

ANDROID與ARDUINO藍芽連線

ANDROID 連線準備

```
public class ClientConnectionActivity extends Activity {
    private static final String APP_UUID = "00001101-0000-1000-8000-
00805F9B34FB";
    private BluetoothConnectionHelper mHelper;
    private Button mLedOnButton;
    private Button mLedOffButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.connection_activity);
        mLedOnButton = (Button) findViewById(R.id.btn_on);
        mLedOnButton.setOnClickListener(mLedOnClickListener);
        mLedOffButton = (Button) findViewById(R.id.btn_off);
        mLedOffButton.setOnClickListener(mLedOffClickListener);
        BluetoothDevice device = getIntent().getParcelableExtra("device");
        mHelper =
BluetoothConnectionHelper.createClient(UUID.fromString(APP_UUID), device);
        mHelper.setMessageReceiver(mListener);
        mHelper.connect();
    }
}
```

.....(還有更多)

ANDROID 連線準備

```
public class ClientConnectionActivity extends Activity {  
    private static final String APP_UUID = "00001101-0000-1000-8000-  
00805F9B34FB";  
    private BluetoothConnectionHelper mHelper;  
    private Button mLedOnButton;  
    private Button mLedOffButton;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.connection_activity);  
        mLedOnButton = (Button) findViewById(R.id.btn_on);  
        mLedOnButton.setOnClickListener(mLedOnClickListener);  
        mLedOffButton = (Button) findViewById(R.id.btn_off);  
        mLedOffButton.setOnClickListener(mLedOffClickListener);  
        BluetoothDevice device = getIntent().getParcelableExtra("device");  
        mHelper =  
BluetoothConnectionHelper.createClient(UUID.fromString(APP_UUID), device);  
        mHelper.setMessageReceiver(mListener);  
        mHelper.connect();  
    }  
}.....(還有更多)
```

目前在ClientConnectionActivity

ANDROID 連線準備

```
public class ClientConnectionActivity extends Activity {  
    private static final String APP_UUID = "00001101-0000-1000-8000-  
    00805F9B34FB";  
    private BluetoothConnectionHelper mHelper;  
    private Button mLedOnButton;  
    private Button mLedOffButton;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.connection_activity);  
        mLedOnButton = (Button) findViewById(R.id.btn_on);  
        mLedOnButton.setOnClickListener(mLedOnClickListener);  
        mLedOffButton = (Button) findViewById(R.id.btn_off);  
        mLedOffButton.setOnClickListener(mLedOffClickListener);  
        BluetoothDevice device = getIntent().getParcelableExtra("device");  
        mHelper =  
        BluetoothConnectionHelper.createClient(UUID.fromString(APP_UUID), device);  
        mHelper.setMessageReceiver(mListener);  
        mHelper.connect();  
    }  
}
```

先宣告連線要使用的UUID
這裡是使用序列Serial的服務

.....(還有更多)

ANDROID 連線準備

```
public class ClientConnectionActivity extends Activity {  
    private static final String APP_UUID = "00001101-0000-1000-8000-  
00805F9B34FB";  
    private BluetoothConnectionHelper mHelper;  
    private Button mLedOnButton;  
    private Button mLedOffButton;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.connection_activity);  
        mLedOnButton = (Button) findViewById(R.id.btn_on);  
        mLedOnButton.setOnClickListener(mLedOnClickListener);  
        mLedOffButton = (Button) findViewById(R.id.btn_off);  
        mLedOffButton.setOnClickListener(mLedOffClickListener);  
        BluetoothDevice device = getIntent().getParcelableExtra("device");  
        mHelper =  
        BluetoothConnectionHelper.createClient(UUID.fromString(APP_UUID), device);  
        mHelper.setMessageReceiver(mListener);  
        mHelper.connect();  
    }  
}
```

可參考網址，有常用藍芽UUID列表
<http://www.douban.com/group/topic/20009323/>

.....(還有更多)

ANDROID 連線準備

```
public class ClientConnectionActivity extends Activity {  
    private static final String APP_UUID = "00001101-0000-1000-8000-  
    00805F9B34FB";  
    private BluetoothConnectionHelper mHelper;  
    private Button mLedOnButton;  
    private Button mLedOffButton;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.connection_activity);  
        mLedOnButton = (Button) findViewById(R.id.btn_on);  
        mLedOnButton.setOnClickListener(mLedOnClickListener);  
        mLedOffButton = (Button) findViewById(R.id.btn_off);  
        mLedOffButton.setOnClickListener(mLedOffClickListener);  
        BluetoothDevice device = getIntent().getParcelableExtra("device");  
        mHelper =  
        BluetoothConnectionHelper.createClient(UUID.fromString(APP_UUID), device);  
        mHelper.setMessageReceiver(mListener);  
        mHelper.connect();  
    }  
    .....(還有更多)
```

宣告連線時要使用的
BluetoothConnectionHelper

ANDROID 連線準備

```
public class ClientConnectionActivity extends Activity {
    private static final String APP_UUID = "00001101-0000-1000-8000-
00805F9B34FB";
    private BluetoothConnectionHelper mHelper;
    private Button mLedOnButton;
    private Button mLedOffButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.connection_activity);
        mLedOnButton = (Button) findViewById(R.id.button_on);
        mLedOnButton.setOnClickListener(mLedOnClickListener);
        mLedOffButton = (Button) findViewById(R.id.button_off);
        mLedOffButton.setOnClickListener(mLedOffClickListener);
        BluetoothDevice device = getIntent().getParcelableExtra("device");
        mHelper =
BluetoothConnectionHelper.createClient(UUID.fromString(APP_UUID), device);
        mHelper.setMessageReceiver(mListener);
        mHelper.connect();
    }
}
.....(還有更多)
```

從Intent取出由DiscoveryActivity
傳來的藍芽裝置資訊

ANDROID 連線準備

```
public class ClientConnectionActivity extends Activity {  
    private static final String APP_UUID = "00001101-0000-1000-8000-  
00805F9B34FB";  
    private BluetoothConnectionHelper mHelper;  
    private Button mLedOnButton;  
    private Button mLedOffButton;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.connection_activity);  
        mLedOnButton = (Button) findViewById(R.id.btn_on);  
        mLedOnButton.setOnClickListener(mLedOnClickListener);  
        mLedOffButton = (Button) findViewById(R.id.btn_off);  
        mLedOffButton.setOnClickListener(mLedOffClickListener);  
        BluetoothDevice device = getIntent().getParcelableExtra("device");  
        mHelper =  
BluetoothConnectionHelper.createClient(UUID.fromString(APP_UUID), device);  
        mHelper.setMessageReceiver(mListener);  
        mHelper.connect();  
    }  
.....(還有更多)
```

createClient可以建立一個藍芽客戶端

ANDROID 連線準備

```
public class ClientConnectionActivity extends Activity {
    private static final String APP_UUID = "00001101-0000-1000-8000-
00805F9B34FB";
    private BluetoothConnectionHelper mHelper;
    private Button mLedOnButton;
    private Button mLedOffButton;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.connection_activity);
        mLedOnButton = (Button) findViewById(R.id.btn_on);
        mLedOnButton.setOnClickListener(mLedOnClickListener);
        mLedOffButton = (Button) findViewById(R.id.btn_off);
        mLedOffButton.setOnClickListener(mLedOffClickListener);
        BluetoothDevice device = getIntent().getParcelableExtra("device");
        mHelper =
BluetoothConnectionHelper.createClient(UUID.fromString(APP_UUID), device);
        mHelper.setMessageReceiver(mListener);
        mHelper.connect();
    }
}
.....(還有更多)
```

參數1傳入
UUID轉為UUID物件

ANDROID 連線準備

```
public class ClientConnectionActivity extends Activity {  
    private static final String APP_UUID = "00001101-0000-1000-8000-  
00805F9B34FB";  
    private BluetoothConnectionHelper mHelper;  
    private Button mLedOnButton;  
    private Button mLedOffButton;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.connection_activity);  
        mLedOnButton = (Button) findViewById(R.id.btn_on);  
        mLedOnButton.setOnClickListener(mLedOnClickListener);  
        mLedOffButton = (Button) findViewById(R.id.btn_off);  
        mLedOffButton.setOnClickListener(mLedOffClickListener);  
        BluetoothDevice device = getIntent().getStringExtra("device");  
        mHelper =  
BluetoothConnectionHelper.createClient(UUID.fromString(APP_UUID), device);  
        mHelper.setMessageReceiver(mListener);  
        mHelper.connect();  
    }  
}.....(還有更多)
```

參數2傳入要連線的
藍芽裝置資訊

ANDROID 連線準備

```
public class ClientConnectionActivity extends Activity {  
    private static final String APP_UUID = "00001101-0000-1000-8000-  
00805F9B34FB";  
    private BluetoothConnectionHelper mHelper;  
    private Button mLedOnButton;  
    private Button mLedOffButton;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.connection_activity);  
        mLedOnButton = (Button) findViewById(R.id.btn_on);  
        mLedOnButton.setOnClickListener(mLedOnClickListener);  
        mLedOffButton = (Button) findViewById(R.id.btn_off);  
        mLedOffButton.setOnClickListener(mLedOnClickListener);  
        BluetoothDevice device = BluetoothAdapter.getDefaultAdapter().getRemoteDevice(ParcelableExtra("device"));  
        mHelper =  
BluetoothConnectionHelper.getClient(UUID.fromString(APP_UUID), device);  
        mHelper.setMessageReceiver(mListener);  
        mHelper.connect();  
    }  
}
```

註冊事件接受器，可以接收到
藍芽伺服端回傳的訊息

.....(還有更多)

ANDROID 連線準備

```
public class ClientConnectionActivity extends Activity {  
    private static final String APP_UUID = "00001101-0000-1000-8000-  
00805F9B34FB";  
    private BluetoothConnectionHelper mHelper;  
    private Button mLedOnButton;  
    private Button mLedOffButton;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.connection_activity);  
        mLedOnButton = (Button) findViewById(R.id.btn_on);  
        mLedOnButton.setOnClickListener(mLedOnClickListener);  
        mLedOffButton = (Button) findViewById(R.id.btn_off);  
        mLedOffButton.setOnClickListener(mLedOffClickListener);  
        BluetoothDevice device = BluetoothAdapter.getDefaultAdapter().getParcelableExtra("device");  
        mHelper = new BluetoothConnectionHelper(this, ClientConnectionActivity.class, UUID.fromString(APP_UUID), device);  
        mHelper.setMessageReceiver(mListener);  
        mHelper.connect();  
    }  
}
```

呼叫connect與藍芽伺服端
進行連線

.....(還有更多)

ANDROID 連線準備

```
protected void onDestroy() {  
    super.onDestroy();  
    if (mHelper != null) {  
        mHelper.close();  
    }  
}
```

記得在onDestroy時
呼叫close與伺服器斷線

ANDROID 傳送訊息

```
private OnClickListener mLedOnClickListener = new OnClickListener(){
    @Override
    public void onClick(View v) {
        if (mHelper.isConnected()) {
            mHelper.sendMessage("1");
        }
    }
};

private OnClickListener mLedOffClickListener = new OnClickListener(){
    @Override
    public void onClick(View v) {
        if (mHelper.isConnected()) {
            mHelper.sendMessage("0");
        }
    }
};
```

ANDROID 傳送訊息

```
private OnClickListener mLedOnClickListener = new OnClickListener(){
    @Override
    public void onClick(View v) {
        if (mHelper.isConnected()) {
            mHelper.sendMessage("1");
        }
    }
};

private OnClickListener mLedOffClickListener = new OnClickListener(){
    @Override
    public void onClick(View v) {
        if (mHelper.isConnected()) {
            mHelper.sendMessage("0");
        }
    }
};
```



檢查目前是否與伺服器端
連線成功

ANDROID 傳送訊息

```
private OnClickListener mLedOnClickListener = new OnClickListener(){
    @Override
    public void onClick(View v) {
        if (mHelper.isConnected()) {
            mHelper.sendMessage("1");
        }
    }
};

private OnClickListener mLedOffClickListener = new OnClickListener(){
    @Override
    public void onClick(View v) {
        if (mHelper.isConnected()) {
            mHelper.sendMessage("0");
        }
    }
};
```

sendMessage()送出字串
此處送出1
Arduino收到後會開啟LED

ANDROID 傳送訊息

```
private OnClickListener mLedOnClickListener = new OnClickListener(){
    @Override
    public void onClick(View v) {
        if (mHelper.isConnected()) {
            mHelper.sendMessage("1");
        }
    }
};

private OnClickListener mLedOffClickListener = new OnClickListener(){
    @Override
    public void onClick(View v) {
        if (mHelper.isConnected()) {
            mHelper.sendMessage("0");
        }
    }
};
```



sendMessage()送出字串
此處送出0
Arduino收到後會關閉LED

ANDROID 客戶端事件接收

```
private OnBluetoothMessageListener mListener = new
OnBluetoothMessageListener() {
    @Override
    public void onMessageReceived(BluetoothDevice device, String message) {

    }
    @Override
    public void onDisconnect(BluetoothDevice device) {
        Toast.makeText(ClientConnectionActivity.this,
            "Disconnect", Toast.LENGTH_LONG).show();
        finish();
    }
    @Override
    public void onConnected(BluetoothDevice device) {
        Toast.makeText(ClientConnectionActivity.this,
            "Connect", Toast.LENGTH_LONG).show();
    }
};
```

ANDROID 客戶端事件接收

```
private OnBluetoothMessageListener mListener = new
OnBluetoothMessageListener() {
    @Override
    public void onMessageReceived(藍芽客戶端事件接收器, String message) {
    }
    @Override
    public void onDisconnect(BluetoothDevice device) {
        Toast.makeText(ClientConnectionActivity.this,
            "Disconnect", Toast.LENGTH_LONG).show();
        finish();
    }
    @Override
    public void onConnected(BluetoothDevice device) {
        Toast.makeText(ClientConnectionActivity.this,
            "Connect", Toast.LENGTH_LONG).show();
    }
};
```



ANDROID 客戶端事件接收

```
private OnBluetoothMessageListener mListener = new
OnBluetoothMessageListener() {
    @Override
    public void onMessageReceived(BluetoothDevice device, String message) {

    }
    @Override
    public void onDisconnect(BluetoothDevice device) {
        Toast.makeText(ClientConnectionActivity.this,
            "Disconnect", Toast.LENGTH_LONG).show();
        finish();
    }
    @Override
    public void onConnected(BluetoothDevice device) {
        Toast.makeText(ClientConnectionActivity.this,
            "Connect", Toast.LENGTH_LONG).show();
    }
};
```

第一種事件：
藍芽伺服器傳訊息過來

ANDROID 客戶端事件接收

```
private OnBluetoothMessageListener mListener = new
OnBluetoothMessageListener() {
    @Override
    public void onMessageReceived(BluetoothDevice device, String message) {

    }
    @Override
    public void onDisconnect(BluetoothDevice device) {
        Toast.makeText(ClientConnectionActivity.this,
            "Disconnect", Toast.LENGTH_LONG).show();
        finish();
    }
    @Override
    public void onConnected(BluetoothDevice device) {
        Toast.makeText(ClientConnectionActivity.this,
            "Connect", Toast.LENGTH_LONG).show();
    }
};
```




參數1 傳訊過來的藍芽裝置

ANDROID 客戶端事件接收

```
private OnBluetoothMessageListener mListener = new
OnBluetoothMessageListener() {
    @Override
    public void onMessageReceived(BluetoothDevice device, String message) {

    }
    @Override
    public void onDisconnect(BluetoothDevice device)
        Toast.makeText(ClientConnectionActivity.this,
            "Disconnect", Toast.LENGTH_LONG).show();
        finish();
    }
    @Override
    public void onConnected(BluetoothDevice device) {
        Toast.makeText(ClientConnectionActivity.this,
            "Connect", Toast.LENGTH_LONG).show();
    }
};
```



參數2 傳遞的訊息

ANDROID 客戶端事件接收

```
private OnBluetoothMessageListener mListener = new
OnBluetoothMessageListener() {
    @Override
    public void onMessageReceived(BluetoothDevice device, String message) {
        // 第二種事件：藍芽伺服器與你斷線
    }
    @Override
    public void onDisconnect(BluetoothDevice device) {
        Toast.makeText(ClientConnectionActivity.this,
            "Disconnect", Toast.LENGTH_LONG).show();
        finish();
    }
    @Override
    public void onConnected(BluetoothDevice device) {
        Toast.makeText(ClientConnectionActivity.this,
            "Connect", Toast.LENGTH_LONG).show();
    }
};
```

ANDROID 客戶端事件接收

```
private OnBluetoothMessageListener mListener = new
OnBluetoothMessageListener() {
    @Override
    public void onMessageReceived(BluetoothDevice device, String message) {

    }
    @Override
    public void onDisconnect(BluetoothDevice device) {
        Toast.makeText(ClientConnectionActivity.this,
            "Disconnect", Toast.LENGTH_LONG).show();
        finish();
    }
    @Override
    public void onConnected(BluetoothDevice device) {
        Toast.makeText(ClientConnectionActivity.this,
            "Connect", Toast.LENGTH_LONG).show();
    }
};
```

第三種事件：
藍芽伺服器與你連線成功
發生在呼叫connect後

總結

- 使用輔助Library，Client的程式碼大約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>

作業

- 使用Arduino接上溫度的Sensor與藍芽
- Android負責搜尋藍芽裝置並且與Arduino進行連線
- 連線成功後Android發出要求給Arduino回傳溫度
- Android接收溫度後要將溫度顯示在畫面上

Q & A