第 2 章　藍牙通訊淺談

內容

第 2 章　藍牙通訊淺談.....1
2-1 手機間的藍牙連線.....2
2-2 藍牙聊天室.....13
2-3　藍牙繪圖板.....42
2-4　總結.....42
2-5　實力評量.....43

本章重點	使用元件
藍牙通訊原理	Bluetooth Client Bluetooth Server
對話框設計	Notifier

我想大家對藍牙裝置並不陌生，現在大部份的科技產品都必備藍牙這個功能，但是你有試過用藍牙和朋友互傳即時短訊嗎，你有試過用藍牙同步讓你的朋友能隨時看見你正在畫的圖嗎，還記得 NINTENDO(任天堂)開發的 DS 系列主機剛出來的時候，每個人人手一台主機，透過 DS 上的 WIFI 裝置(好像不是耶)和朋友互傳訊息和圖畫，如今我們的 Android 手機也能透過藍牙裝置做到，本章我們將介紹 App Inventor 所提供的藍牙通訊元件，包含 Bluetooth Client 與 Bluetooth Server 等。

另外我們也會在程式中加入 Notifier 元件來設計各種不同的對話框，例如按下斷線鈕的時候，會先跳出一個確定視窗詢問使用者是否真的要中斷藍牙連線，確認之後才中斷。

本章節將依序以下列範例來介紹 Android 手機如何透過藍牙傳遞各種資訊，請參考表 2-1。

表 2-1　第 2 章範例列表

編號	名稱	說明
EX2-1	BTconnect	藍牙連線
EX2-2	BTchat	藍牙聊天室
EX2-3	BTdraw	藍牙繪圖板

2-1 手機間的藍牙連線

首先我們要先透過 BTconnect 這個範例教大家如何建立藍牙連線，並在程式中使用 Notifier 這個元件將可能遇到錯誤與藍牙連線不成功時的原因顯示在螢幕上，讓我們更容易除錯。在 BTchat 和 BTdraw 的範例中，也是建立基本的藍牙連線功能上來完成互傳訊息的功能，因此 BTconnect 所用的方法會在後面範例中重覆應用，可想而知，這個範例非常重要。

圖 2-1 藍牙連線執行畫面(阿吉補圖)

請依照下面指示完成此程式：

Designer

<STEP1>建立新專案

- 請先啟動 Designer，並建立一個新專案，將它命名為 BTconnect。

<STEP2>選擇程式元件

表 2-2 是 BTconnect 程式所需的程式元件。

元件類別	父類別	名稱	該元件用途
HorizontalArrangement	Screen Arrangement	HorizontalArrangement1	提供同一列可放兩個元件
Button	Basic	btnConnect	藍牙的連線與斷線
ListPicker	Basic	1stDevice	顯示藍牙連線的列表
BluetoothClient	Other stuff	BluetoothClient1	藍牙裝置
TinyDB	Other stuff	TinyDB1	存取與讀取資料的空間
Notifier	Other stuff	Notifier	顯示錯誤訊息

<STEP3>設定程式元件屬性

- 將 HorizontalArrangement 元件拖到 Screen1 中，並將 Width 設成 Fill parent。
- 將 Button 元件拖到 Screen1 的 HorizontalArrangement 元件中，將元件名稱改

為 btnConnect，並將 text 設成 Connect。

- 將 ListPicker 元件拖到 Screen1 的 HorizontalArrangement 元件中，將元件名稱改為 1stDevice(檢查名稱是 Ist or 1st)，並將 text 設成 Select Device，Width 設成 Fill parent。
- 將 BluetoothClient、TinyDB 以及 Notifier 等元件拖到 Screen1 中，完成後如圖 2-2。(類似的動作可合併)

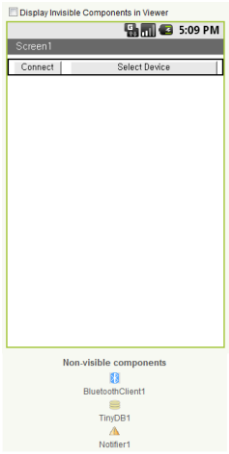

圖 2-2 Designer 完成畫面

Block Editor

<STEP4>建立名為 BTconnectDevice 的副程式

請新增以下指令：

- Built-In→Definition 的 procedure 指令，並將指令名稱改為 BTconnectDevice。
- Built-In→Definition 的 name 指令，並將指令名稱改為 MACaddrofBTDevice。
- Built-In→Control 的 if 指令。
- My Blocks→BluetoothClient1 的 BluetoothClient1.Connect 指令。
- My Blocks→My Definition 的 MACaddrofBTDevice 指令。
- My Blocks→btnConnect 的 set btnConnect.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Disconnect。

- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Device Connected。

請將各元件如圖 2-3 組合，在這裡我們定義了一個副程式 **BTconnectDevice**，它是用來建立藍牙連線，因為這段程式會常常用到，為了讓程式更簡潔，所以將它定義成副程式來使用。當 **BTconnectDevice** 被呼叫的時候，會將**欲連線裝置**的藍牙位址存在 MACaddrofBTDevice 這個變數中。接著藉由 if 判斷式來確認連線是否成功。BluetoothClinent1.Connect (MACaddrofBTDevice)代表要 BluetoothClinent1 元件對 MACaddrofBTDevice 這個藍牙位址的裝置進行連線。

當藍牙連線成功時，btnConnect 的按鍵名稱會從 Connect 變成 Disconnect，並在螢幕上會顯示 Device Connected 的提示字樣，告訴我們連線成功了。

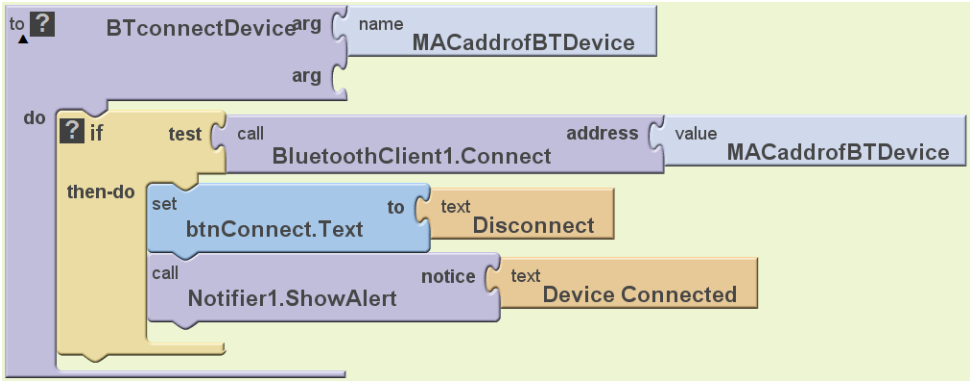

圖 2-3 建立名為 BTconnectDevice 的副程式

<STEP5>初始化設定

請新增以下指令：

- Built-In→Definition 的 variable 指令，並將指令名稱改為 DeviceMAC。
- Built-In→Text 的 text 指令，並將指令名稱刪去。
- My Blocks→Screen1 的 Screen1.Initialize 指令。
- My Blocks→btnConnect 的 set btnConnect.Enabled 指令。
- Built-In→Logic 的 false 指令。

- My Blocks→My Definitions 的 set global DeviceMAC 指令。
- My Blocks→TinyDB1 的 TinyDB1.GetValue 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 StoredDevice。
- Built-In→Control 的 if 指令。
- Built-In→Text 的 length 指令。
- My Blocks→My Definitions 的 DeviceMAC 指令。
- Built-In→Math 的 number 和大於框架指令。

將上面三個指令結合成一判斷式：**length(DeviceMAC)>0**

- Built-In→Control 的 ifelse 指令。
- My Blocks→BluetoothClient1 的 BluetoothClient1.IsDevicePaired 指令。
- My Blocks→My Definitions 的 DeviceMAC 指令。
- My Blocks→IsDevice 的 set 1stDevice.Text 指令。
- My Blocks→My Definitions 的 DeviceMAC 指令。
- My Blocks→btnConnect 的 set btnConnect.Enabled 指令。
- Built-In→Logic 的 true 指令。
- My blocks→My Definitions 的 set global DeviceMAC 指令。
- Built-In→Text 的 text 指令，並將指令名稱刪去。
- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Saved Device Is not Paired。
- My Blocks→TinyDB1 的 TinyDB1.StoreValue 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 StoredDevice。
- Built-In→Text 的 text 指令，並將指令名稱刪去。

請將各元件如圖 2-4 組合，Screen1.Initialize 這一段程式將會完成初始化的動作，也就是說程式開始執行後，btnConnect 按鍵是不能觸發的，同時我們宣告了一個 DeviceMAC 的變數，主要是用來儲存從 TinyDB1 得到的字串資料。從圖中可以看到在 TinyDB1 後面，我們建立了一個 StoredDevice 的標籤，這個標籤可

以視為一個資料夾，TinyDB1 得到的字串資料都會存在這個標籤中，如果沒有建立標籤的話就無法儲存 TinyDB1 的資料。

接著是兩個 if 判斷式，第一個 if 判斷式是判斷 DeviceMAC 是否真的得到一筆字串資料（如果資料存在，則字串長度必大於 0），而這筆資料就是欲連線裝置的藍牙位址。第二個 if…else 判斷式則是判斷是否與指定藍牙位址配對成功，如果配對成功，選單按鍵 1stDevice 的名稱會變成藍牙的位址，而 btnConnect 按鍵可以觸發；如果配對失敗，則將 DeviceMAC 儲存的字串(藍牙位址)消去，並在手機螢幕上顯示裝置「Saved Device is not Paired」的字樣，代表配對失敗。同時我們也要將 TinyDB1 標籤內儲存的字串消去（用一個空的字串蓋過去）。

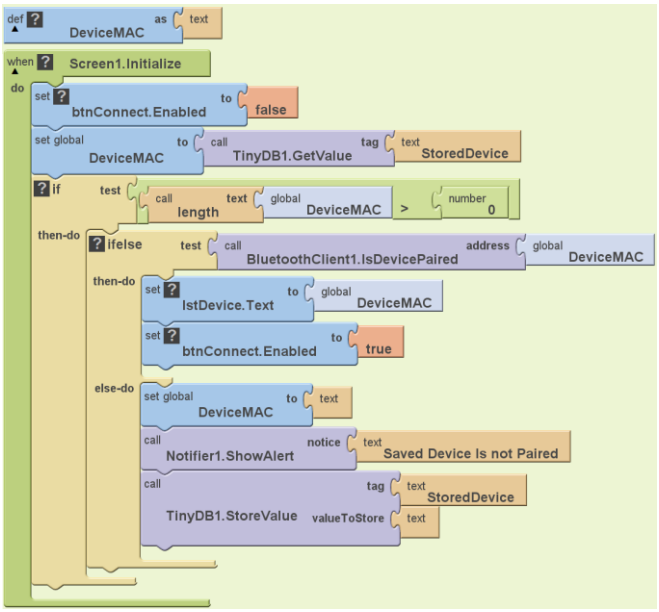

圖 2-4 初始化設定

<STEP6>錯誤處理

請新增以下指令：

- My Blocks→Screen1 的 Screen1.ErrorOccurred 指令。
- Built-In→Control 的 ifelse 指令。
- Built-In→Text 的 contains 指令。
- My Blocks→My Definitions 的 component 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 BluetoothClient。

- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- My Blocks→My Definitions 的 message 指令。
- My Blocks→BluetoothClient 的 BluetoothClient1.Disconnect 指令。
- My Blocks→btnConnect 的 set btnConnect.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Connect。
- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- Built-In→Text 的 make text 指令。
- Built-In→Text 的 text 指令，建立四個指令並分別打上下列的文字：

1	2	3	4
An Error Occurred in Component：	\nFunction：	\nError Number：	\nMessage：

- My Blocks→My Definitions 的 component、functionName、errorNumber 和 message 指令。

請將各元件如圖 2-5 組合，Screen1.ErrorOccurred 事件會在發生錯誤時自動被呼叫，並可用來抓出與錯誤的各種資訊。Screen1.ErrorOccurred 事件會自動宣告了四個變數（component、functionName、errorNumber 與 message），這四個變數分別代表發生錯誤的元件、指令、錯誤碼以及相關訊息。

接著我們做了一個判斷，判斷是否是 BluetoothClient 元件發生錯誤，如果是的話，就將藍牙發生錯誤的訊息顯示在螢幕上，並將藍牙斷線，最後將 btnConnect 按鍵的名稱改成 Connect；如果不是的話，則將該元件的名稱和各種錯誤訊息顯示在螢幕上。有關 Screen1.ErrorOccurred 事件請參閱本書附錄 B<App Inventor 指令集-MyBlocks>。

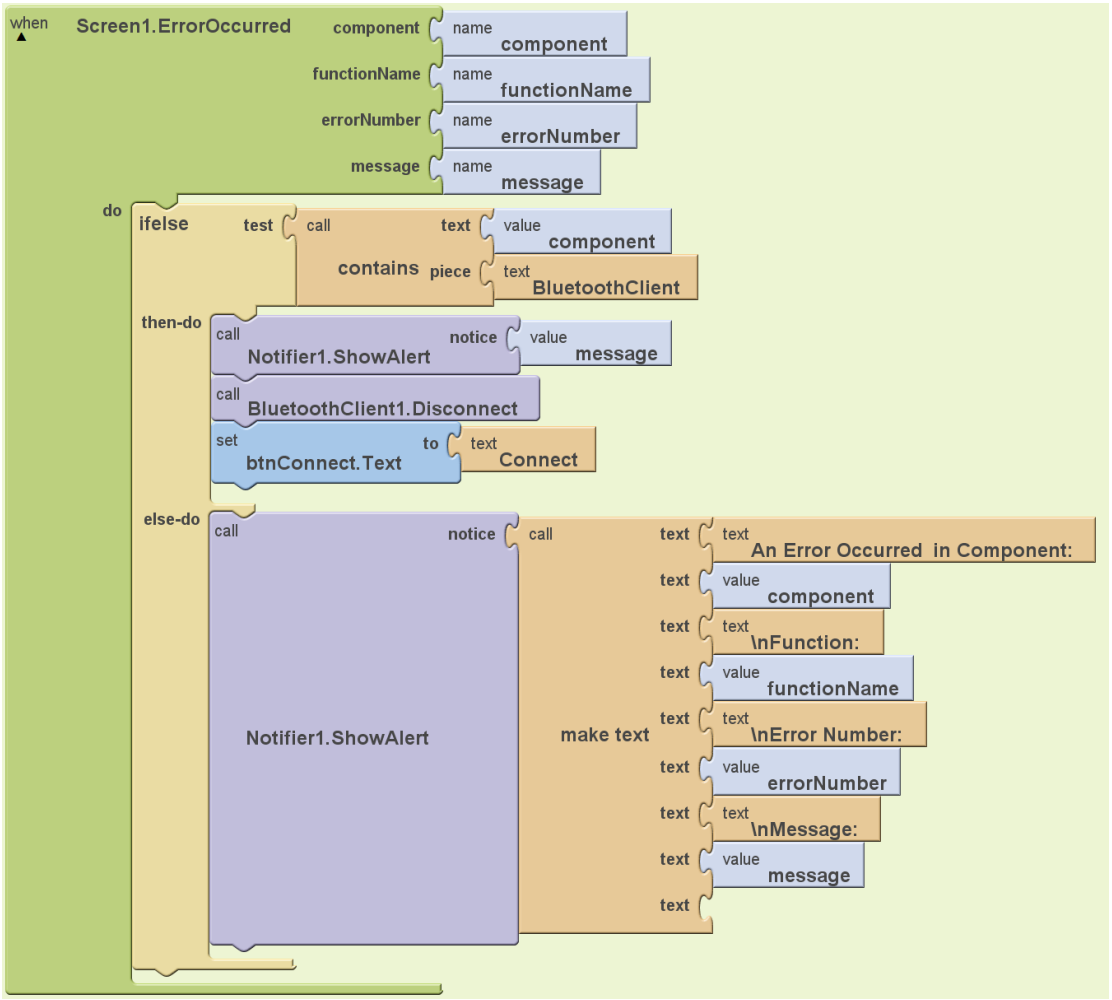

圖 2-5 錯誤處理

<STEP7>設定與選取藍牙裝置選單

請新增以下指令：

- My Blocks→1stDevice 的 1stDevice.BeforePicking 指令。
- My Blocks→BluetoothClient 的 BluetoothClient1.Disconnect 指令。
- My Blocks→btnConnect 的 set btnConnect.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Connect。
- My Blocks→1stDevice 的 set 1stDevice.Elements 指令。
- My Blocks→BluetoothClient 的 BluetoothClient1. AddressesAndNames 指令。
- My Blocks→1stDevice 的 1stDevice.AfterPicking 指令。
- My Blocks→My Definitions 的 set global DeviceMAC 指令。

- My Blocks→1stDevice 的 1stDevice.Selection 指令。
- My Blocks→TinyDB1 的 TinyDB1.StoreValue 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 StoredDevice。
- My Blocks→My Definitions 的 DeviceMAC 指令。
- My Blocks→1stDevice 的 set 1stDevice.Text 指令。
- My Blocks→My Definitions 的 DeviceMAC 指令。
- My Blocks→btnConnect 的 set btnConnect.Enabled 指令。
- Built-In→Logic 的 true 指令。

請將各元件如圖 2-6 組合，此段程式主要分為兩部份，1stDevice 清單選取前與選取後。在尚未選取藍牙位址時，藍牙應該是斷線狀態，且要將所有可進行連線的藍牙裝置名稱與位址都存入 1stDevice 清單中。接著在選取 1stDevice 清單其中某一藍牙裝置後，將此藍牙位址存入 DeviceMAC 變數與 TinyDB1 的 StoredDevice 標籤之中，並將 1stDevice 清單名稱變成所選取裝置的藍牙位址，最後讓 btnConnect 按鍵變成可觸發，隨時可準備連線。

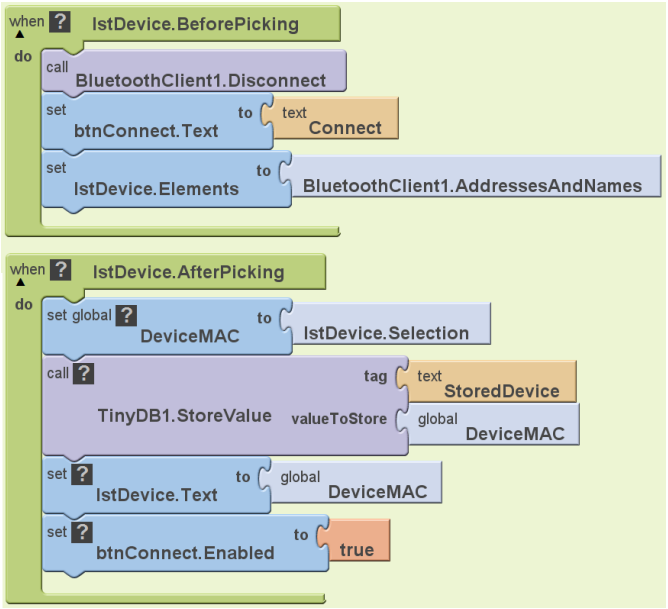

圖 2-6 藍牙位址選取

fafa(吉改到這)

<STEP8>藍牙連線與斷線

請新增以下指令：

- My Blocks→btnConnect 的 btnConnect.Click 指令。
- Built-In→Control 的 ifelse 指令。
- My Blocks→BluetoothClient 的 BluetoothClient1.IsConnected 指令。
- My Blocks→BluetoothClient 的 BluetoothClient1.Disconnect 指令。
- My Blocks→btnConnect 的 set btnConnect.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Connect。
- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Device Disconnected。
- Built-In→Control 的 ifelse 指令。
- My Blocks→btnConnect 的 btnConnect.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Connect。
- Built-In→Math 的等於框架指令。

請將上面三個指令結合成一判斷式： btnConnect.Text = Connect。

- Built-In→Control 的 ifelse 指令。
- Built-In→Text 的 length 指令。
- My Blocks→My Definitions 的 DeviceMAC 指令。
- Built-In→Math 的 number 和大於框架指令。

將上面三個指令結合成一判斷式： length(DeviceMAC)>0。

- My Blocks→My Definitions 的 BTconnectDevice 指令。
- My Blocks→My Definitions 的 DeviceMAC 指令。
- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 You have not selected a Device To Connect To。
- My Blocks→BluetoothClient 的 BluetoothClient1.Disconnect 指令。

- My Blocks→btnConnect 的 btnConnect.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Connect。
- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Device Disconnected。

請將各元件如圖 2-7 組合，此段程式我們將藍牙的連線與斷線寫在同一個按鍵裡，所以你會看到程式一開始就是一個判斷式，用來判斷按鍵按下時是連線狀態還是斷線狀態，如果是連線狀態的話，就將藍牙斷線，讓 btnConnect 按鍵名稱變回 Connect，且在螢幕上顯示裝置已經斷線的字樣；如果是斷線狀態的話，代表按鍵按下是要做連線的動作，且 btnConnect 按鍵在未連線時的名稱會是 Connect，因此在第二個判斷式中，我們要判斷 btnConnect 的名稱是否為 Connect，如果不是的話，代表現在仍有藍牙裝置在連線狀態，因此要和前面一樣做斷線的動作；如果 btnConnect 的名稱是 Connect 的話，代表是可以做連線的狀態，所以我們用第三個判斷式來判斷藍牙位址是否有儲存到 DeviceMAC 變數中，如果沒有儲存到此變數中，仍然無法做連線的動作，因此會在手機螢幕上顯示出你沒有選取要連結的裝置的字樣，而如果確定有存到此變數中的話，我們會將此變數放到一開始制作的副程式，做連線的動作。我們會用如此多的判斷式來確保藍牙是否能正確連線，是因為在這些地方都有可能出現連線錯誤的情況，如果有其中一個沒有做判斷的話，很有可能發生找不到錯誤的原因，而藍牙也沒能正確連線，這樣就很可惜我們花時間寫了一個這麼棒的程式!

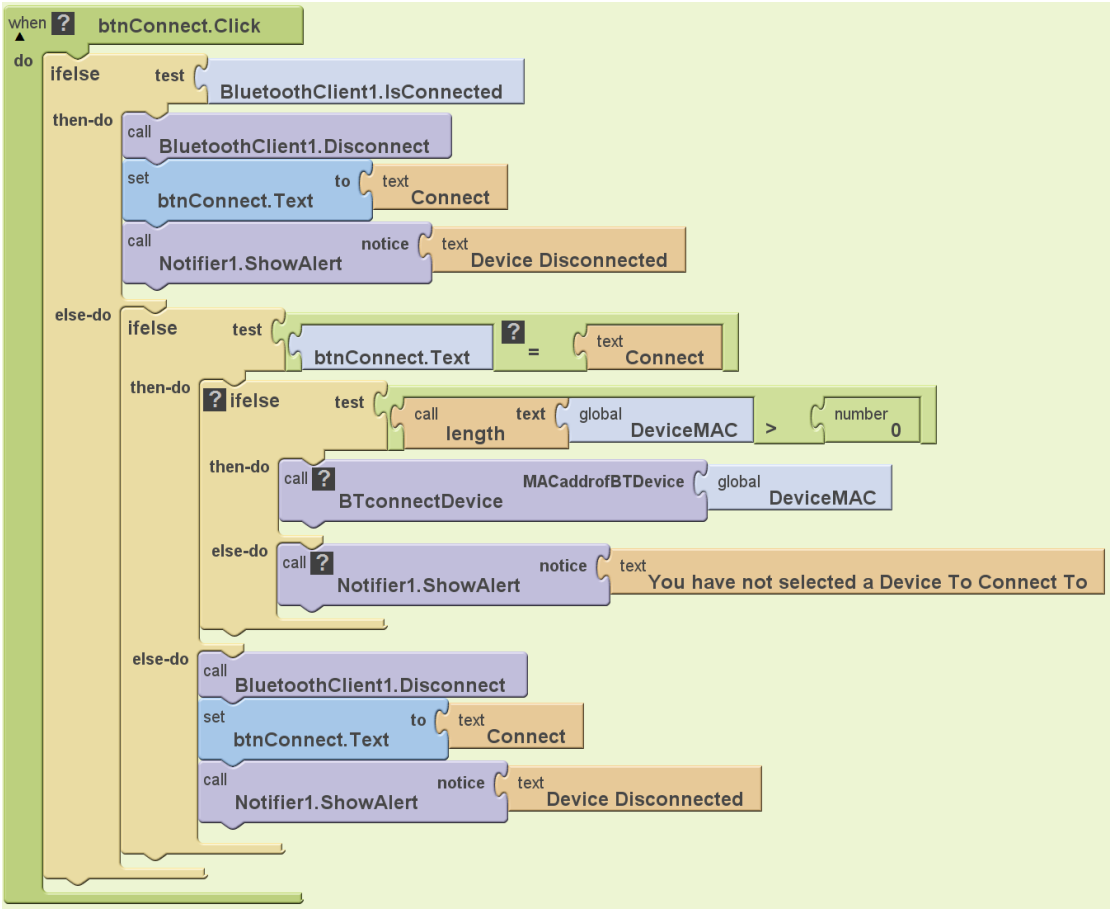

圖 2-7 藍牙的連線與斷線

2-2 藍牙聊天室

在了解藍牙連線的基本設定後，我們就可以將它應用在接下來的這個範例「藍牙聊天室」，從這個名字就可以知道我們要將手機間聊天的訊息互相傳輸，因此在這個範例中我們使用了一個新的元件 BluetoothServer，這是一個伺服器的元件，用來處理藍牙資料的傳輸與運算，透過它我們才能將兩台手機間做溝通，讓我們來把程式完成吧!

圖 2-8 藍牙聊天室執行畫面(補圖)

請依照下面指示完成此程式：

Designer

<STEP1>建立新專案

- 請先啟動 Designer，並建立一個新專案，將它命名為 BTchat。

<STEP2>選擇程式元件

表 2-3 是 BTchat 程式所需的程式元件。

元件類別	父類別	名稱	該元件用途
HorizontalArrangement	Screen Arrangement	HorizontalArrangement1	提供同一列可放兩個元件
Button	Basic	btnConnect	藍牙的連線與斷線
ListPicker	Basic	1stDevice	顯示藍牙連線的列表
TextBox	Basic	txtname.txtout	選取裝置與輸入訊息
Label	Basic	lblmode	顯示提示文字
CheckBox	Basic	chkready	裝置選取後的再確認
BluetoothClient	Other stuff	BluetoothClient1	藍牙裝置
BluetoothServer	Other stuff	BluetoothServer1	藍牙伺服器
TinyDB	Other stuff	TinyDB1	存取與讀取資料的空間
Notifier	Other stuff	Notifier1	顯示錯誤訊息
Clock	Basic	Clock1	狀態更新

<STEP3>設定程式元件屬性

- 將 HorizontalArrangement 元件拖到 Screen1 中，將元件名稱改為 Connect_window，並將 Visible 的打勾去掉和 Width 設成 Fill parent。(※Visible 打勾去掉的動作很重要且會影響到後面的程式，請務必確認有做此動作)
- 將 Button 元件拖到 Screen1 的 HorizontalArrangement 元件中，將元件名稱改為 btnConnect，並將 text 設成 Connect。
- 將 ListPicker 元件拖到 Screen1 的 HorizontalArrangement 元件中，將元件名稱改為 1stDevice，並將 text 設成 Select Device，Width 設成 Fill parent。

- 將 HorizontalArrangement 元件拖到 Screen1 中，將元件名稱改為 ClientMode_window，並將 Width 和 Height 都設成 Fill parent。
- 將 HorizontalArrangement 元件拖到 Screen1 的 ClientMode_window 中，並將 Width 和 Height 都設成 Fill parent。
- 將 TextBox 元件拖到 Screen1 的 ClientMode_window 中的 HorizontalArrangement1 裡面，將元件名稱改為 txtname，並完成下表的設定。

Hint	Width
Enter Your Name Here	Fill parent

- 將 Button 元件拖到 Screen1 的 ClientMode_window 中的 HorizontalArrangement1 裡面，將元件名稱改為 btnmode，並在 Text 的地方打上 Client Mode。
- 將 HorizontalArrangement 元件拖到 Screen1 的 ClientMode_window 中，並將 Width 設成 Fill parent。
- 將 Label 元件拖到 Screen1 的 ClientMode_window 中的 HorizontalArrangement2 裡面，將元件名稱改為 lblmode，並完成下表的設定。

Text	Width
Currently in Server mode	Fill parent

- 將 CheckBox 元件拖到 Screen1 的 ClientMode_window 中的 HorizontalArrangement2 裡面，將元件名稱改為 chkready，並完成下表的設定。

Text	Width
Check when Ready	Fill parent

- 將 HorizontalArrangement 元件拖到 Screen1 中，將元件名稱改為 chat_window，完成下表的設定。

Visible	Width	Height

不打勾	Fill parent	350pixels

(※Visible 打勾去掉的動作很重要且會影響到後面的程式，請務必確認有做此動作)

- 將 HorizontalArrangement 元件拖到 Screen1 的 chat_window 中，並將 Width 設成 Fill parent。
- 將 TextBox 元件拖到 Screen1 的 chat_window 中的 HorizontalArrangement3 裡面，將元件名稱改為 txtout，並完成下表的設定。

Hint	Width
(空白)	Fill parent

- 將 Button 元件拖到 Screen1 的 chat_window 中的 HorizontalArrangement3 裡面，將元件名稱改為 btnsend，並在 Text 的地方打上 Send。
- 將 Label 元件拖到 Screen1 的 chat_window 中，將元件名稱改為 lblout，並完成下表的設定。

text	Width	Height
(空白)	Fill parent	Fill parent

- 將 BluetoothClient 元件拖到 Screen1 中。
- 將 TinyDB 元件拖到 Screen1 中。
- 將 Notifier 元件拖到 Screen1 中。
- 將 BluetoothServer 元件拖到 Screen1 中。
- 將 Clock 元件拖到 Screen1 中，並將 TimerInterval 設成 200。

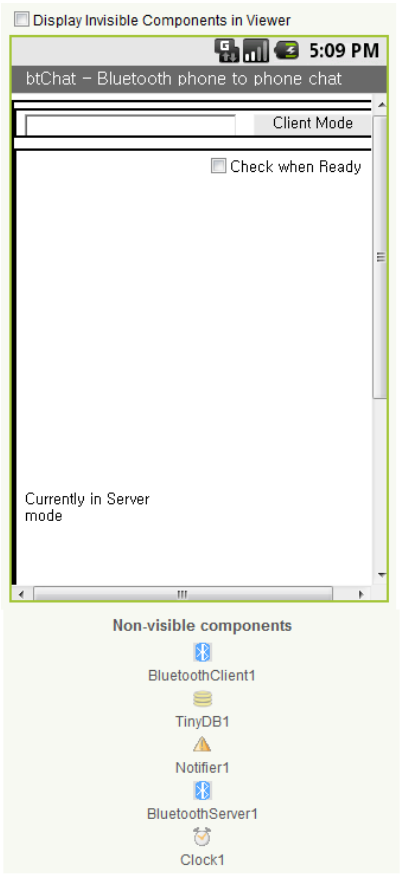

圖 2-9 Designer 完成畫面

Block Editor

<STEP4>建立副程式

請新增以下指令：

- Built-In→Definition 的 procedure 指令，並將指令名稱改為 BTconnectDevice。
- Built-In→Definition 的 name 指令，並將指令名稱改為 MACaddrofBTDevice。
- Built-In→Control 的 if 指令。
- My Blocks→BluetoothClient1 的 BluetoothClient1.Connect 指令。
- My Blocks→My Definition 的 MACaddrofBTDevice 指令。
- My Blocks→btnConnect 的 set btnConnect.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Disconnect。
- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Device Connected。

- Built-In→Definition 的 procedure 指令，並將指令名稱改為 timestamp。
- Built-In→Text 的 make text 指令。
- Built-In→Text 的 text 指令，新增兩個指令並將指令名稱改為 [和] 。
- My Blocks→Clock1 的 Clock1.FormatTime 指令。
- My Blocks→Clock1 的 Clock1.Now 指令。

請將各元件如圖 2-2 組合，在藍牙聊天室這個範例中，我們總共建立了兩個副程式，其中一個是建立藍牙連線用，在前一個藍牙連線的範例我們有做完整的說明，因此不再詳細介紹，另一個副程式是顯示目前時間用，在一般的即時聊天程式中，傳送的訊息主要有三種：時間、人名(裝置名稱)和發送的訊息，這個範例也同樣能做到，當呼叫這個副程式時，它會自動去抓取手機目前的時間，並將它回傳到程式中，在後面的程式我們便可將回傳的時間顯示在聊天室的頁面上。

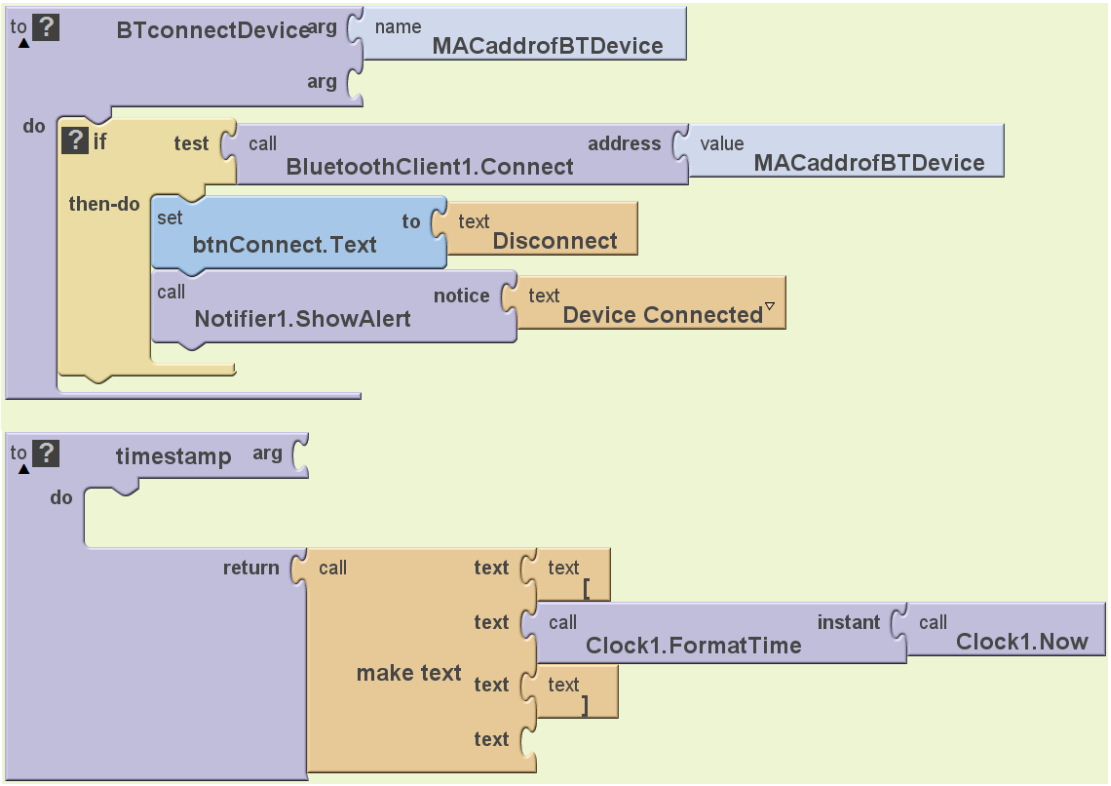

圖 2-10 建立連線與顯示時間的副程式

<STEP5>初始化設定

請新增以下指令：

- Built-In→Definition 的 variable 指令，並將指令名稱改為 mode。
- Built-In→Text 的 text 指令，並將指令名稱改為 server。
- Built-In→Definition 的 variable 指令，並將指令名稱改為 DeviceMAC。
- Built-In→Text 的 text 指令，並將指令名稱刪去。
- My Blocks→Screen1 的 Screen1.Initialize 指令。
- My Blocks→btnConnect 的 set btnConnect.Enabled 指令。
- Built-In→Logic 的 false 指令。
- My Blocks→My Definitions 的 set global DeviceMAC 指令。
- My Blocks→TinyDB1 的 TinyDB1.GetValue 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 StoredDevice。
- My Blocks→txtname 的 set txtname.Text 指令。
- My Blocks→TinyDB1 的 TinyDB1.GetValue 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 StoredName。
- Built-In→Control 的 if 指令。
- Built-In→Text 的 length 指令。
- My Blocks→txtname 的 txtname.Text 指令。
- Built-In→Math 的 number 和等於框架指令。

將上面三個指令結合成一判斷式： length(txtname.Text)=0。

- My Blocks→txtname 的 set txtname.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 unknown。
- Built-In→Control 的 if 指令。
- Built-In→Text 的 length 指令。
- My Blocks→My Definitions 的 DeviceMAC 指令。
- Built-In→Math 的 number 和大於框架指令。

將上面三個指令結合成一判斷式： length(DeviceMAC)>0。

- Built-In→Control 的 ifelse 指令。
- My Blocks→BluetoothClient1 的 BluetoothClient1.IsDevicePaired 指令。
- My Blocks→My Definitions 的 DeviceMAC 指令。
- My Blocks→IsDevice 的 set 1stDevice.Text 指令。
- My Blocks→My Definitions 的 DeviceMAC 指令。
- My Blocks→btnConnect 的 set btnConnect.Enabled 指令。
- Built-In→Logic 的 true 指令。
- My blocks→My Definitions 的 set global DeviceMAC 指令。
- Built-In→Text 的 text 指令，並將指令名稱刪去。
- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Saved Device Is not Paired。
- My Blocks→TinyDB1 的 TinyDB1.StoreValue 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 StoredDevice。
- Built-In→Text 的 text 指令，並將指令名稱刪去。

請將各元件如圖 2-11 組合，此段的初始化設定和前一個範例幾乎一樣，因此我們針對一些新功能和重點做說明，一開始我們宣告了兩個變數，mode 和 DeviceMAC，mode 變數是在後面的程式中用來判別誰是 server 端用的，因為在這個範例中的資料傳輸不再只是單向，而是雙向傳輸，可能是 server 對 user 或是 user 對 server，所以要搞清楚現在是那一端在送命令，如圖 2-12，DeviceMAC 變數在前面介紹過了，是用來儲存決定連線的藍牙位置。在初始化的程式中，我們在 TinyDB1 內建立了兩個標籤： StoredDevice 和 StoredName，並分別將藍牙位置(存在 DeviceMAC 變數中的)和裝置名稱(txtname.Text)存在標籤中，方便我們做傳輸；我想應該有人已經注意到了，在螢幕上的 txtname 這個指令，就是用來讓我們輸入自己想要的聊天室暱稱，且為了避免聊天室暱稱是空白時可能產生錯誤的情況，我們用第一個判斷式來判別裝置名稱是否為空白，如果是空白的，

會自動將聊天室暱稱設為 unknown，在後面的判斷式則分別用來判別藍牙位址是否成功儲存在 DeviceMAC 變數中和是否配對成功，畢竟這些都是有可能出錯的地方，做好萬全的檢查可以讓程式更強漸。

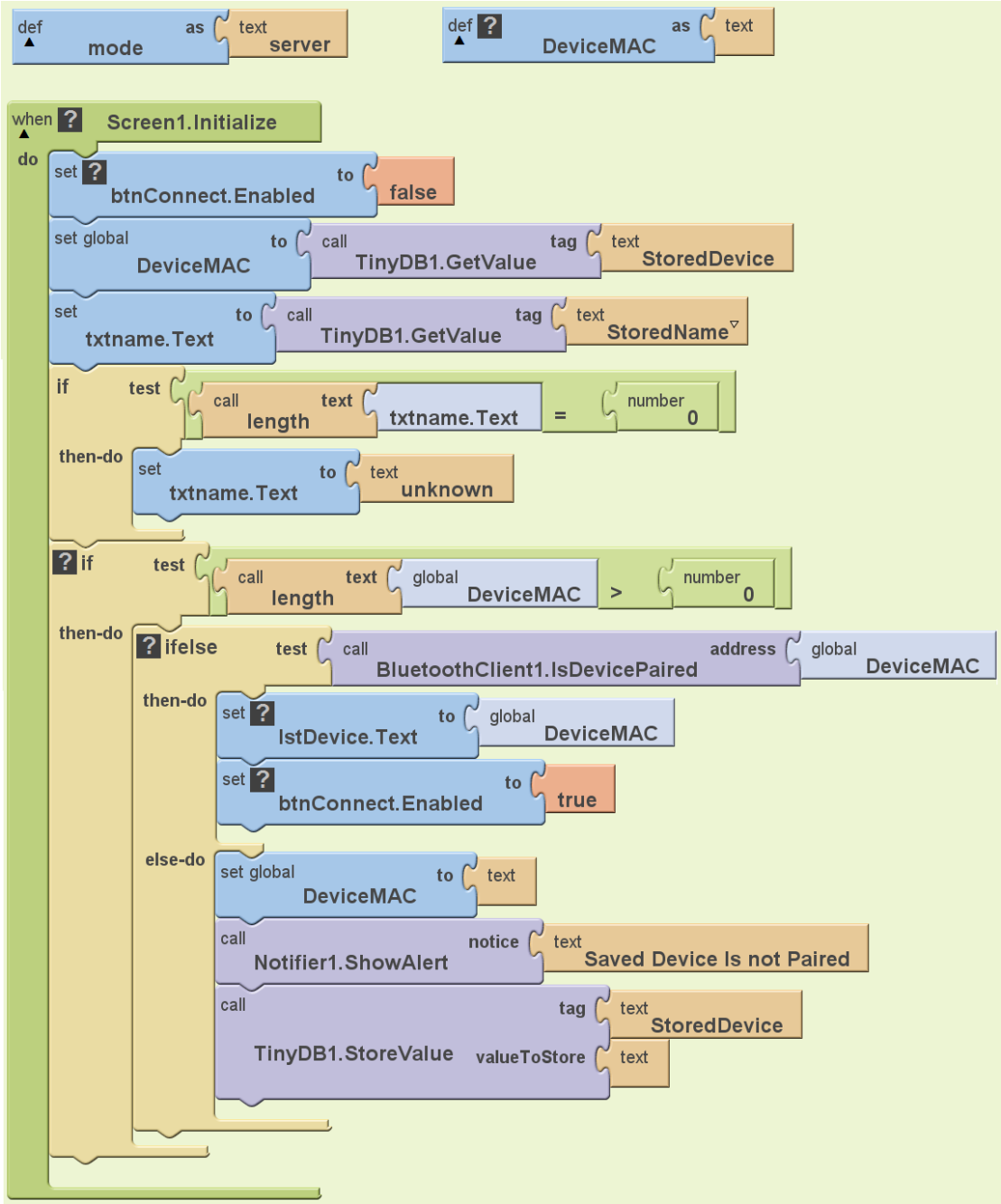

圖 2-11 初始化設定

圖 2-12 雙向傳輸

<STEP6>藍牙選單設定與選取

請新增以下指令：

- My Blocks→1stDevice 的 1stDevice.BeforePicking 指令。
- My Blocks→BluetoothClient 的 BluetoothClient1.Disconnect 指令。
- My Blocks→btnConnect 的 set btnConnect.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Connect。
- My Blocks→1stDevice 的 set 1stDevice.Elements 指令。
- My Blocks→BluetoothClient 的 BluetoothClient1. AddressesAndNames 指令。
- My Blocks→1stDevice 的 1stDevice.AfterPicking 指令。
- My Blocks→My Definitions 的 set global DeviceMAC 指令。
- My Blocks→1stDevice 的 1stDevice.Selection 指令。
- My Blocks→TinyDB1 的 TinyDB1.StoreValue 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 StoredDevice。
- My Blocks→My Definitions 的 DeviceMAC 指令。
- My Blocks→1stDevice 的 set 1stDevice.Text 指令。
- My Blocks→My Definitions 的 DeviceMAC 指令。
- My Blocks→btnConnect 的 set btnConnect.Enabled 指令。
- Built-In→Logic 的 true 指令。

請將各元件如圖 2-13 組合，此段程式是基本的藍牙連線設定，在前一範例中有詳細說明之。

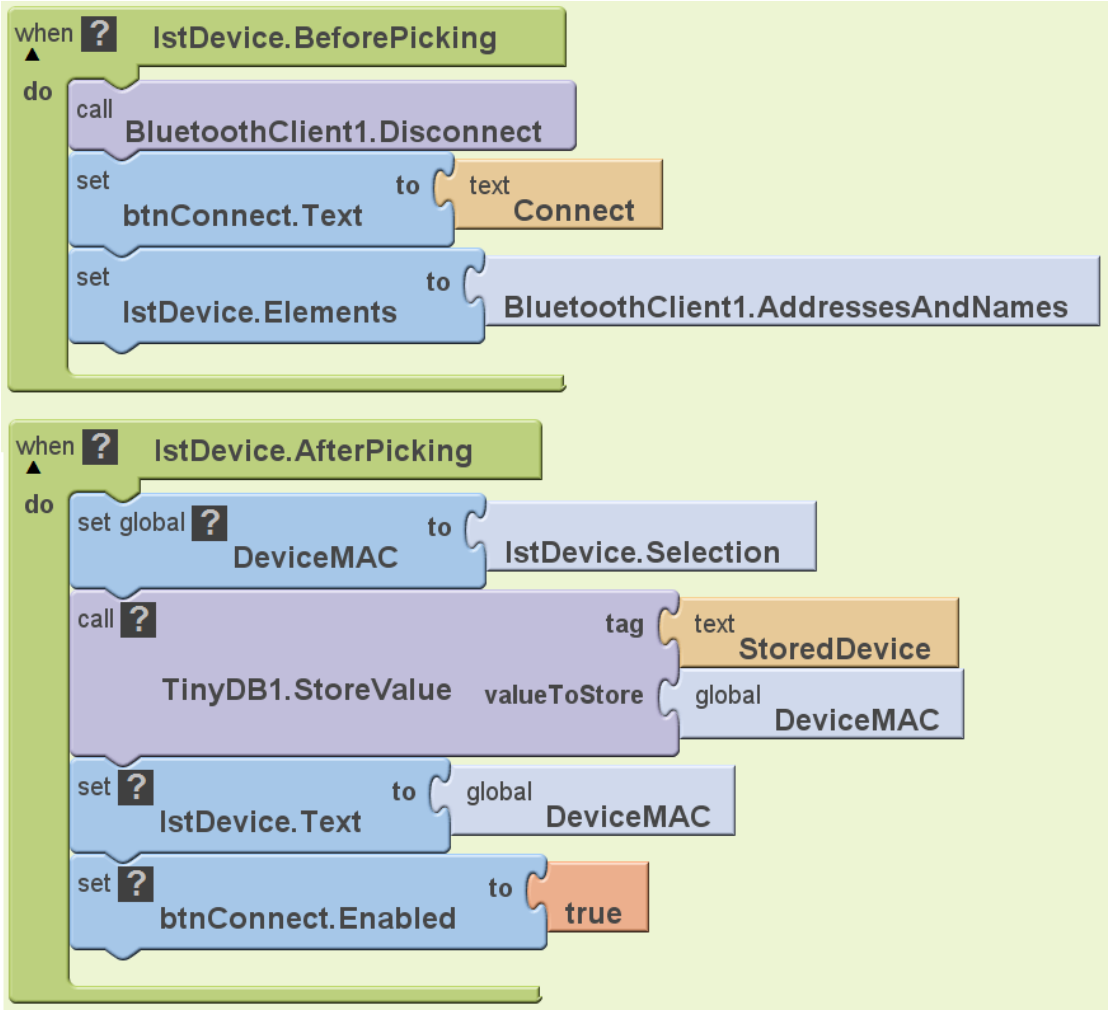

圖 2-13 藍牙位址選取

<STEP7>藍牙連線與斷線

請新增以下指令：

- My Blocks→btnConnect 的 btnConnect.Click 指令。
- Built-In→Control 的 ifelse 指令。
- My Blocks→btnConnect 的 btnConnect.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Connect。
- Built-In→Math 的等於框架指令。

將上面三個指令結合成一判斷式： btnConnect.Text = Connect

- Built-In→Control 的 ifelse 指令。

- Built-In→Text 的 length 指令。
- My Blocks→My Definitions 的 DeviceMAC 指令。
- Built-In→Math 的 number 和大於框架指令。

將上面三個指令結合成一判斷式： length(DeviceMAC)>0。

- My Blocks→My Definitions 的 BTconnectDevice 指令。
- My Blocks→My Definitions 的 DeviceMAC 指令。
- My Blocks→chat_window 的 set chat_window.Visible 指令。
- Built-In→Logic 的 true 指令。
- My Blocks→ClientMode_window 的 set ClientMode_window.Visible 指令。
- Built-In→Logic 的 false 指令。
- My Blocks→1stDevice 的 set 1stDevice.Visible 指令。
- Built-In→Logic 的 false 指令。
- My Blocks→My Definitions 的 set global mode 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 client。
- My Blocks→Connect_window 的 set Connect_window.Visible 指令。
- Built-In→Logic 的 true 指令。
- My Blocks→Clock1 的 set TimerEnabled 指令。
- Built-In→Logic 的 true 指令。
- My Blocks→btnConnect 的 set btnConnect.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Disconnect。
- My Blocks→TinyDB1 的 TinyDB1.StoreValue 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 StoredName。
- My Blocks→txtname 的 txtname.Text 指令。
- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 You have not selected a Device To Connect To。

- My Blocks→btnConnect 的 btnConnect.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Connect。
- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Device Disconnected。
- Built-In→Control 的 ifelse 指令。
- Built-In→Text 的 text=指令。
- My Blocks→My Definitions 的 mode 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 server。
- My Blocks→BluetoothServer1 的 BluetoothServer1.Disconnect 指令。
- My Blocks→BluetoothClient1 的 BluetoothClient1.Disconnect 指令。

請將各元件如圖 2-14 組合，這個連線與斷線的按鍵設定，在前一個範例中也有提到過，透過一個判斷式來判斷目前藍牙是否有連線，如果是連線狀態，按下此鍵就斷線，如果是斷線狀態的話，則做連線的動作；但是在這裡還有一個很重要的步驟，不要忘了我們使用了藍牙伺服器的元件，在連線狀態下，是做雙向的傳輸，因此在斷線時，需考慮到是那一方做斷線，而不是當 Server 端按下斷線，就將雙向的傳輸斷線，所以我們將 Server 端的斷線交給 Server 端的 Button，User 端的斷線交給 User 端的 Button。

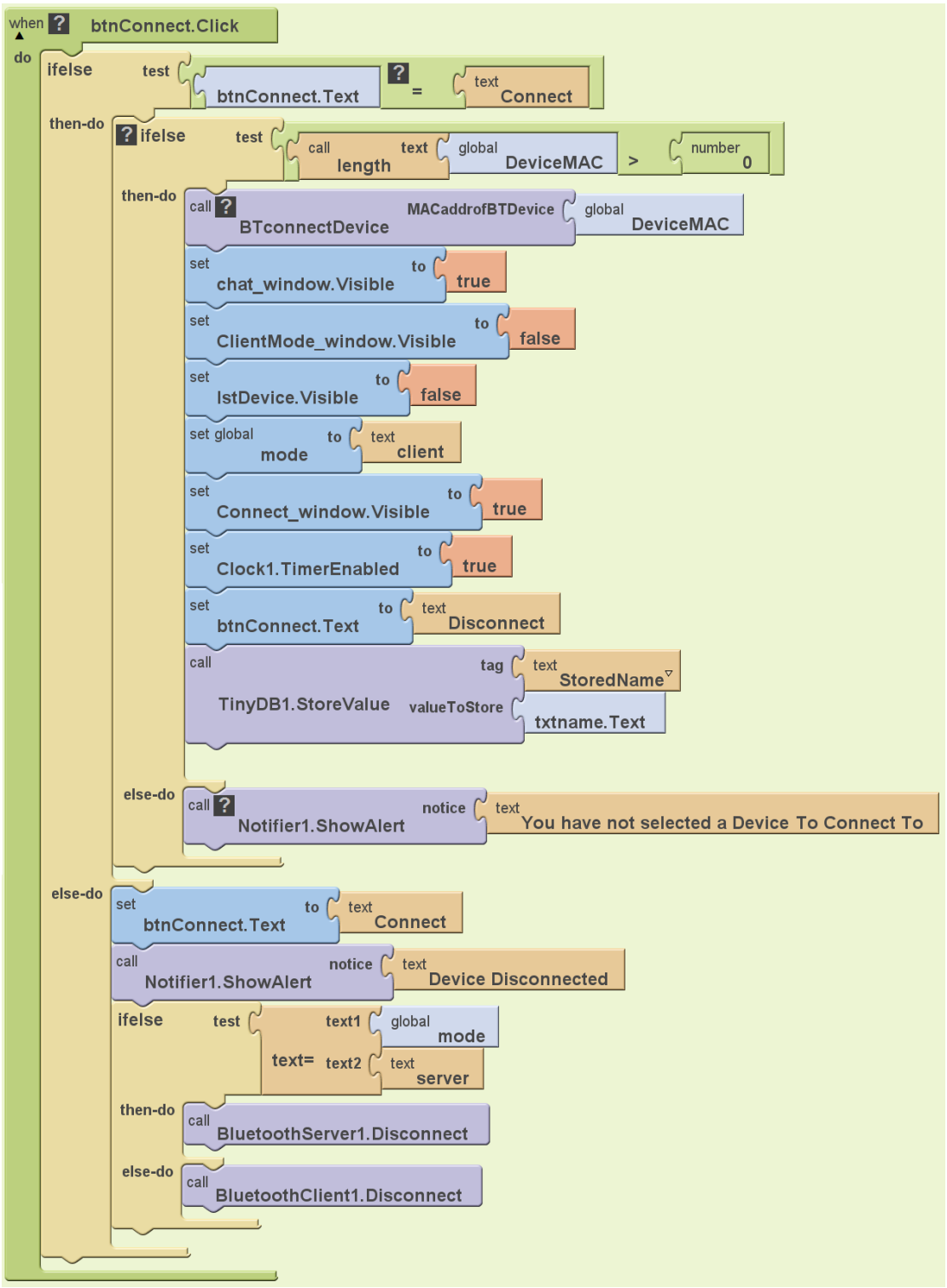

圖 2-14 連線與斷線

<STEP8>誰是 Server?誰是 User?

請新增以下指令：

- My Blocks→btnmode 的 btnmode.Click 指令。
- Built-In→Control 的 ifelse 指令。
- My Blocks→btnmode 的 btnmode.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Client Mode。
- Built-In→Math 的等於框架指令。

將上面三個指令結合成一判斷式： btnmode.Text = Client Mode

- My Blocks→Connect_window 的 set Connect_window.Visible 指令。
- Built-In→Logic 的 true 指令。
- My Blocks→BluetoothServer1 的 BluetoothServer1.StopAccepting 指令。
- My Blocks→btnmode 的 set btnmode.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Server Mode。
- My Blocks→My Definitions 的 set global mode 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 client。
- My Blocks→lblmode 的 set lblmode.text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Currently in Client mode。
- My Blocks→chkready 的 set chkready.Visible 指令。
- Built-In→Logic 的 false 指令。
- My Blocks→BluetoothServer1 的 BluetoothServer1.StopAccepting 指令。
- My Blocks→Connect_window 的 set Connect_window.Visible 指令。
- Built-In→Logic 的 false 指令。
- My Blocks→btnmode 的 set btnmode.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Client Mode。
- My Blocks→My Definitions 的 set global mode 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 server。
- My Blocks→lblmode 的 set lblmode.text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Currently in Server mode。

- My Blocks→Connect_window 的 set Connect_window.Visible 指令。
- Built-In→Logic 的 true 指令。

請將各元件如圖 2-15 組合，當藍牙連線成功後，我們的藍牙聊天室其實還不能啟動，因為我們還沒做伺服器的連線，但是連線前我們需要確定那一端是 Server，那一端是 User，因此我們用一個按鍵(btnMode)來決定，如果選擇的是 User 端，需要從藍牙列表中去找尋 Server 端的藍牙位址，這樣便完成單一方向的傳輸；如果選擇的是 Server 端，不用特意從列表去找尋 User 端的藍牙位址，只要等 User 端都連線完成後，透過一個確認鍵(checkready)便能完成 Server 端對 User 端的藍牙傳輸，確認鍵的設定與 Server 端的連線我們會在下一個步驟做詳細的說明。

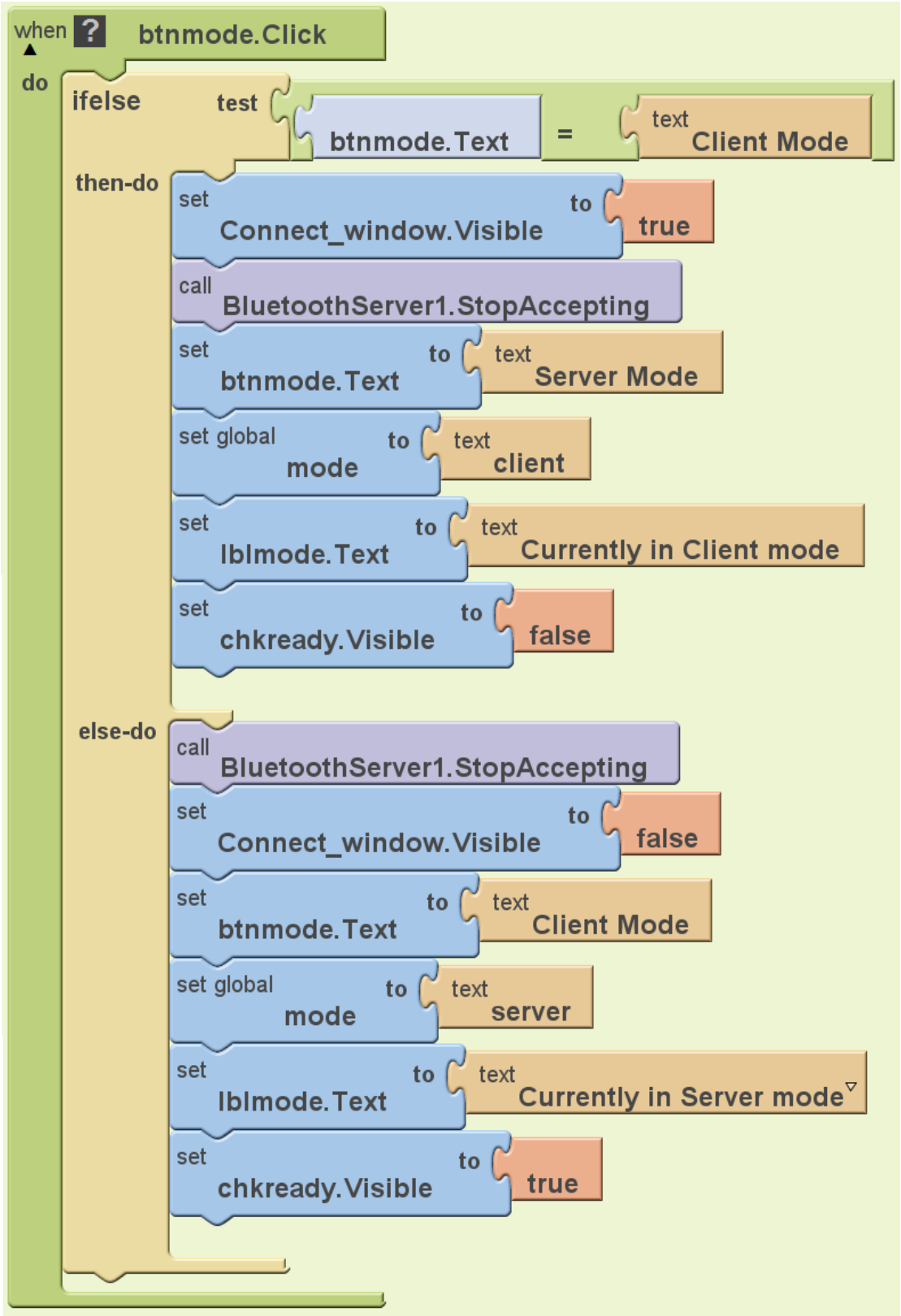

圖 2-15 決定 Server 與 User

<STEP9>藍牙伺服器開啟與連線

請新增以下指令：

- My Blocks→BluetoothServer1 的 BluetoothServer1.ConnectionAccepted 指令。
- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Device Connected。
- My Blocks→Clock1 的 set Clock1.TimerEnabled 指令。
- Built-In→Logic 的 true 指令。
- My Blocks→My Definitions 的 set global mode 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 server。
- My Blocks→ClientMode_window 的 set ClientMode_window.Visible 指令。
- Built-In→Logic 的 false 指令。
- My Blocks→btnConnect 的 set btnConnect.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Disconnect。
- My Blocks→btnConnect 的 set btnConnect.Enabled 指令。
- Built-In→Logic 的 true 指令。
- My Blocks→1stDevice 的 set 1stDevice.Visible 指令。
- Built-In→Logic 的 false 指令。
- My Blocks→chat_window 的 set chat_window.Visible 指令。
- Built-In→Logic 的 true 指令。
- My Blocks→Connect_window 的 set Connect_window.Visible 指令。
- Built-In→Logic 的 true 指令。
- My Blocks→TinyDB1 的 TinyDB1.StoreValue 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 StoredName。
- My Blocks→textname 的 textname.Text 指令。

- My Blocks→chkready 的 chkready.Changed 指令。
- Built-In→Control 的 ifelse 指令。
- My Blocks→chkready 的 chkready.Checked 指令。

- My Blocks→BluetoothServer1 的 BluetoothServer1.AcceptConnection 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 BlueChat。
- My Blocks→BluetoothServer1 的 BluetoothServer1.StopAccepting 指令。

請將各元件如圖 2-16 組合，從圖中可以看到程式分為兩個部份，伺服器的連線接受與確認鍵設定，也就是說當確認鍵打勾後，才會接受伺服器的連線，並且將裝置名稱命名為 BlueChat，伺服器成功連線後，會在螢幕上顯示”裝置已連線”的字樣，並將 Server 端的聊天室暱稱(textname)存入 TinyDB 中，方便後面程式設定。在這裡需要注意必須等所有的 User 端都連線後再做打勾的動作，否則會有 User 端沒接受到伺服器的連線。

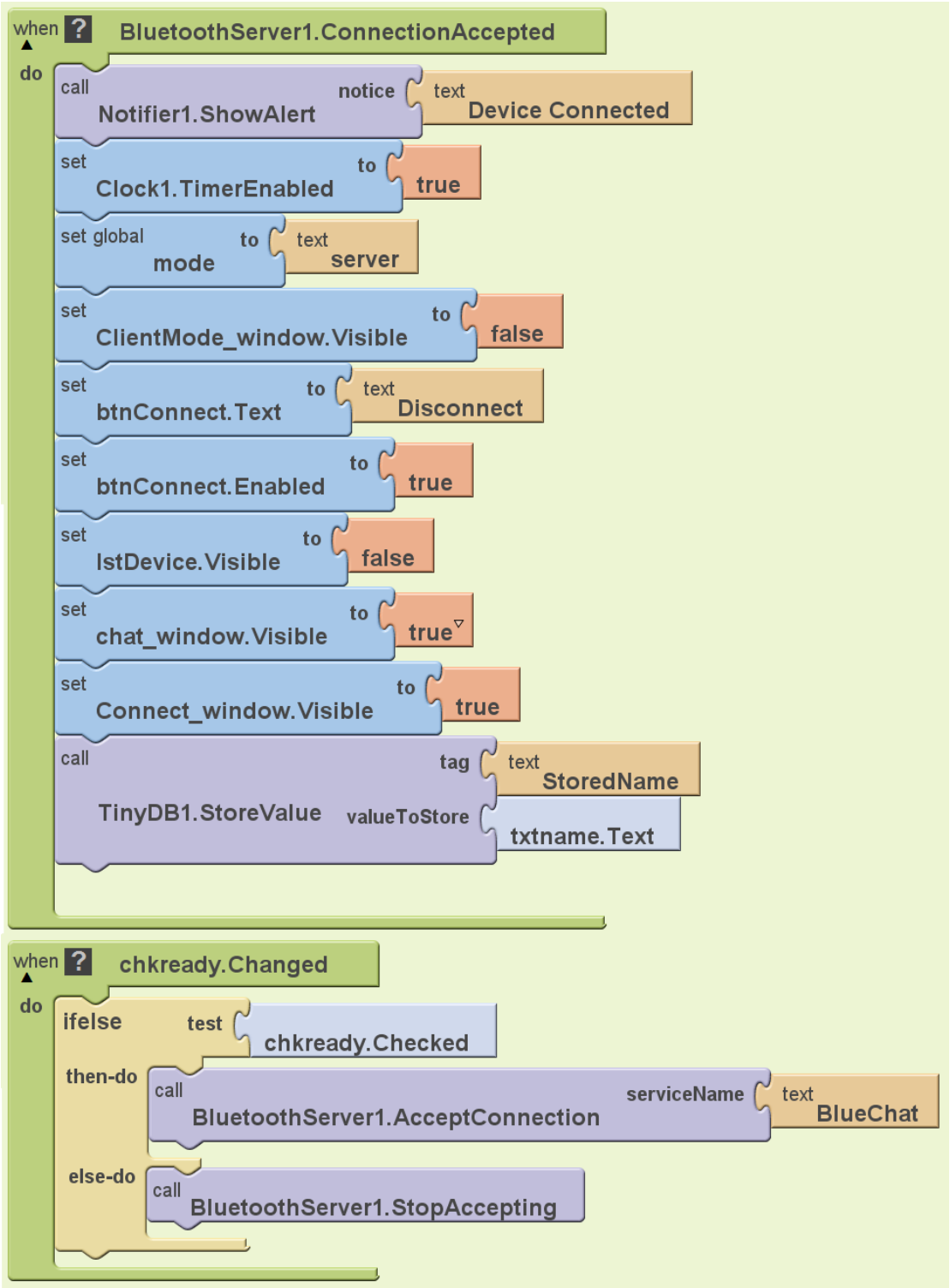

圖 2-16 伺服器正確連線

<STEP2>錯誤處理

請新增以下指令：

- My Blocks→Screen1 的 Screen1.ErrorOccurred 指令。
- Built-In→Control 的 ifelse 指令。
- Built-In→Logic 的 or 指令。
- Built-In→Text 的 contains 指令，新增兩個指令。
- My Blocks→My Definitions 的 component 指令，新增兩個指令。
- Built-In→Text 的 text 指令，新增兩個指令並將指令名稱改為 BluetoothClient 和 BluetoothServer。
- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- My Blocks→My Definitions 的 message 指令。
- Built-In→Control 的 ifelse 指令。
- Built-In→Text 的 text=指令。
- My Blocks→My Definitions 的 mode 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 server。
- My Blocks→BluetoothServer1 的 BluetoothServer1.Disconnect 指令。
- My Blocks→btnConnect 的 btnConnect.Text 指令，新增兩個指令。
- Built-In→Text 的 text 指令，新增兩個指令並將指令名稱都改為 Connect。
- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- Built-In→Text 的 make text 指令。
- Built-In→Text 的 text 指令，建立四個指令並分別打上下列的文字：

1	2	3	4
An Error Occurred in Component：	\nFunction：	\nError Number：	\nMessage：

- My Blocks→My Definitions 的 component、functionName、errorNumber 和 message 指令。

請將各元件如圖 2-17 組合，在前一個範例中，我們也做了錯誤處理，在這裡我們要將藍牙連線與伺服器連線可能發生錯誤的情況都考慮進來，當有任何一方發生連線錯誤，都會做斷線的命令，方便我們重新連線。

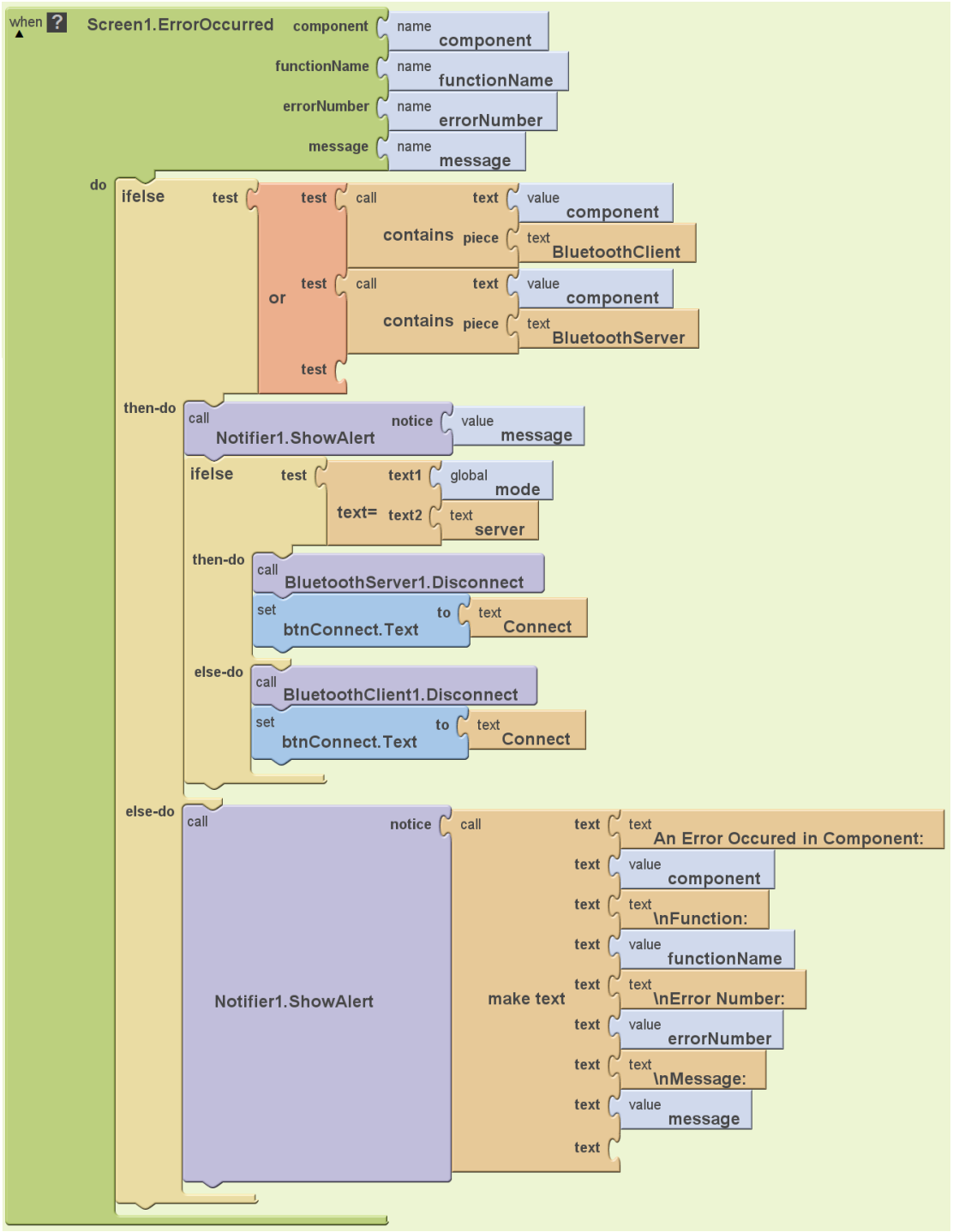

圖 2-17 錯誤處理

<STEP11>更新訊息

請新增以下指令：

- My Blocks→Clock1 的 Clock1.Timer 指令。
- Built-In→Control 的 ifelse 指令。
- Built-In→Text 的 text=指令。

- My Blocks→My Definitions 的 mode 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 server。
- Built-In→Control 的 ifelse 指令。
- My Blocks→BluetoothServer1 的 BluetoothServer1.IsConnected 指令。
- Built-In→Control 的 if 指令。
- My Blocks→BluetoothServer1 的 BluetoothServer1.BytesAvailableToReceive 指令。
- Built-In→Math 的 number 和大於框架指令。

將上面三個指令結合成一判斷式： BluetoothServer1.BytesAvailableToReceive > 0

- My Blocks→lblout 的 set lblout.Text 指令。
- Built-In→Text 的 make text 指令。
- My Blocks→My Definitions 的 timestamp 指令。
- My Blocks→BluetoothServer1 的 BluetoothServer1.ReceiveText 指令。
- My Blocks→BluetoothServer1 的 BluetoothServer1.BytesAvailableToReceive 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為\n。
- My Blocks→lblout 的 lblout.Text 指令。
- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Chat Partner Disconnected。
- My Blocks→Clock1 的 set Clock1.TimerEnabled 指令。
- Built-In→Logic 的 false 指令。
- My Blocks→btnConnect 的 set btnConnect.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Connect。
- My Blocks→chat_window 的 set chat_window.Visible 指令。
- Built-In→Logic 的 false 指令。
- Built-In→Control 的 ifelse 指令。

- Built-In→Text 的 text=指令。
- My Blocks→My Definitions 的 mode 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 server。
- My Blocks→Connect_window 的 set Connect_window.Visible 指令。
- Built-In→Logic 的 true 指令。
- My Blocks→ClientMode_window 的 set ClientMode_window.Visible。
- Built-In→Logic 的 true 指令。
- My Blocks→1stDevice 的 set 1stDevice.Visible 指令。
- Built-In→Logic 的 true 指令。
- Built-In→Control 的 ifelse 指令。
- My Blocks→BluetoothServer1 的 BluetoothServer1.IsConnected 指令。
- Built-In→Control 的 if 指令。
- My Blocks→BluetoothServer1 的 BluetoothServer1.BytesAvailableToReceive 指令。
- Built-In→Math 的 number 和大於框架指令。

將上面三個指令結合成一判斷式： BluetoothServer1.BytesAvailableToReceive > 0

- My Blocks→lblout 的 set lblout.Text 指令。
- Built-In→Text 的 make text 指令。
- My Blocks→My Definitions 的 timestamp 指令。
- My Blocks→BluetoothClient1 的 BluetoothClient1.ReceiveText 指令。
- My Blocks→BluetoothClient1 的 BluetoothClient1.BytesAvailableToReceive 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為\n。
- My Blocks→lblout 的 lblout.Text 指令。
- My Blocks→Notifier1 的 Notifier1.ShowAlert 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Chat Partner Disconnected。

- My Blocks→Clock1 的 set Clock1.TimerEnabled 指令。
- Built-In→Logic 的 false 指令。
- My Blocks→btnConnect 的 set btnConnect.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 Connect。
- My Blocks→chat_window 的 set chat_window.Visible 指令。
- Built-In→Logic 的 false 指令。
- Built-In→Control 的 ifelse 指令。
- Built-In→Text 的 text=指令。
- My Blocks→My Definitions 的 mode 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 server。
- My Blocks→Connect_window 的 set Connect_window.Visible 指令。
- Built-In→Logic 的 true 指令。
- My Blocks→ClientMode_window 的 set ClientMode_window.Visible。
- Built-In→Logic 的 true 指令。
- My Blocks→1stDevice 的 set 1stDevice.Visible 指令。
- Built-In→Logic 的 true 指令。

請將各元件如圖 2-18 組合，我們在這裡使用了一個時間指令，不斷的在確認 Server 端和 User 端現在是否仍在連線，或是有發生什麼情況導致斷線，當在連線狀態下，我們會持續更新目前的時間(timestamp 指令)，並確認目前 Server 端和 User 端接收到的字元數，將這些資訊顯示在螢幕上；如果發生斷線了，會在螢幕上顯示”與配對方已中斷連線”的字樣，並將 User 端的藍牙列表重新開啟，才能與 Server 端重新連線，切記不要將 Server 端的藍牙列表開啟，因為 Server 端不需刻意要去 User 端的藍牙位置。

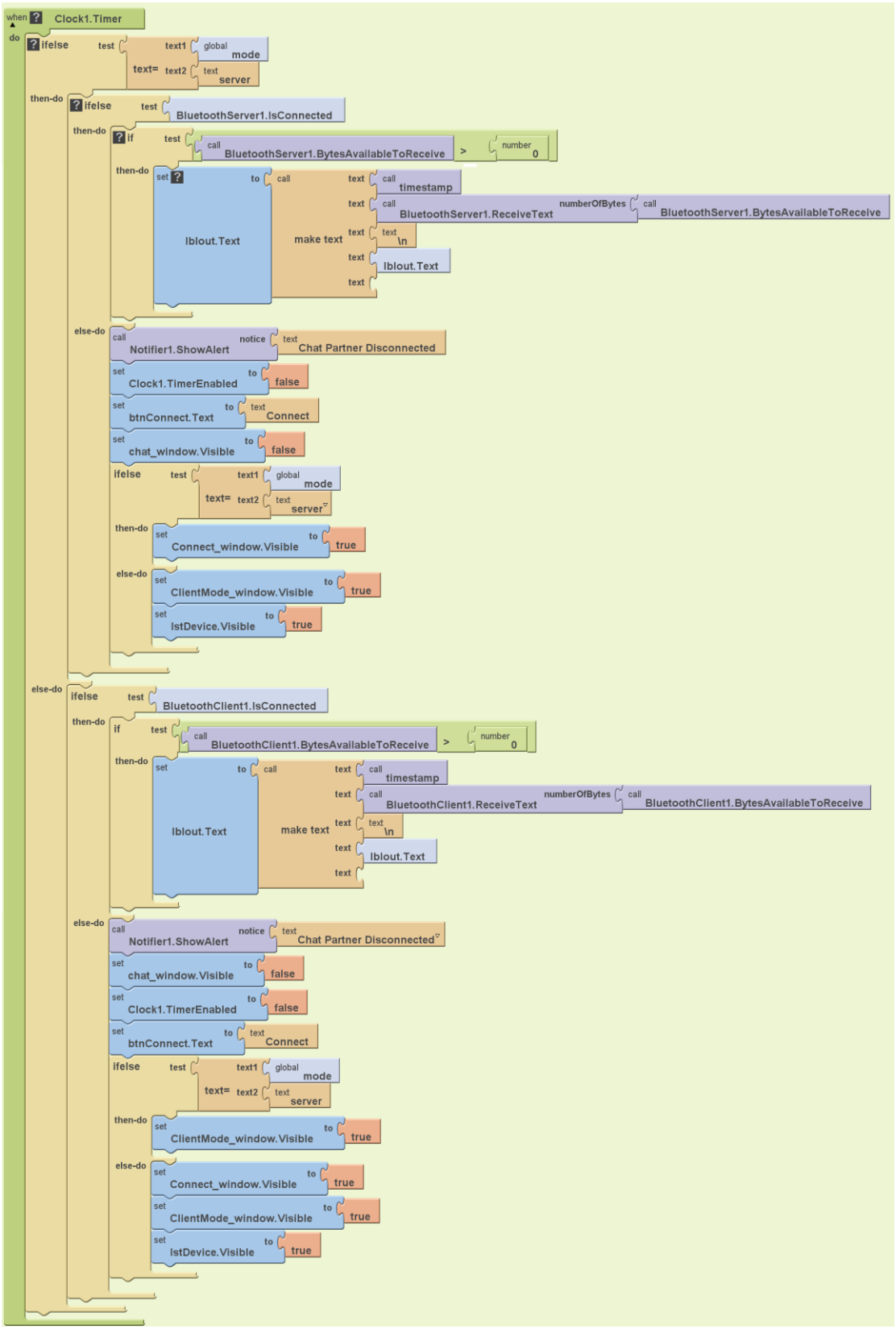
when Clock1.Timer
do
if else test text= text1 global mode text2 text server
then-do
if else test BluetoothServer1.IsConnected
then-do
if test call BluetoothServer1.BytesAvailableToReceive > number 0
then-do
set lblout.Text to call make text text timestamp text call BluetoothServer1.ReceiveText numberOfBytes call BluetoothServer1.BytesAvailableToReceive text text \n text lblout.Text text
else-do
call Notifier1.ShowAlert notice text Chat Partner Disconnected
set Clock1.TimerEnabled to false
set btnConnect.Text to text Connect
set chat_window.Visible to false
if else test text= text1 global mode text2 text server
then-do
set Connect_window.Visible to true
else-do
set ClientMode_window.Visible to true
set lstDevice.Visible to true
else-do
if else test BluetoothClient1.IsConnected
then-do
if test call BluetoothClient1.BytesAvailableToReceive > number 0
then-do
set lblout.Text to call make text text timestamp text call BluetoothClient1.ReceiveText numberOfBytes call BluetoothClient1.BytesAvailableToReceive text text \n text lblout.Text text
else-do
call Notifier1.ShowAlert notice text Chat Partner Disconnected
set chat_window.Visible to false
set Clock1.TimerEnabled to false
set btnConnect.Text to text Connect
if else test text= text1 global mode text2 text server
then-do
set ClientMode_window.Visible to true
else-do
set Connect_window.Visible to true
set ClientMode_window.Visible to true
set lstDevice.Visible to true

圖 2-18 訊息更新

<STEP12>傳送聊天訊息

- Built-In→Definition 的 variable 指令，並將指令名稱改為 outtext。
- Built-In→Text 的 text 指令。
- My Blocks→btnSend 的 btnSend.Click 指令。
- Built-In→Control 的 if 指令。
- Built-In→Text 的 length 指令。
- My Blocks→txtout 的 txtout.Text 指令。
- Built-In→Math 的 number 和大於框架指令。

將上面四個指令結合成一判斷式： length(txtout.Text)>0

- My Blocks→My Definitions 的 set global outtext 指令。
- Built-In→Text 的 make text 指令。
- My Blocks→txtname 的 txtname.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為>。
- My Blocks→txtout 的 txtout.Text 指令。
- My Blocks→txtout 的 set txtout.Text 指令。
- Built-In→Text 的 text 指令，並將指令名稱刪去。
- Built-In→Control 的 ifelse 指令。
- Built-In→Text 的 text=指令。
- My Blocks→My Definitions 的 mode 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為 server。
- My Blocks→BluetoothServer1 的 BluetoothServer1.SendText 指令。
- My Blocks→My Definitions 的 outtext 指令。
- My Blocks→BluetoothClient1 的 BluetoothClient1.SendText 指令。
- My Blocks→My Definitions 的 outtext 指令。
- My Blocks→lblout 的 set lblout.Text 指令。
- Built-In→Text 的 make text 指令。

- My Blocks→My Definitions 的 timestamp 指令。
- My Blocks→My Definitions 的 outtext 指令。
- Built-In→Text 的 text 指令，並將指令名稱改為\n。
- My Blocks→lblout 的 lblout.Text 指令。

請將各元件如圖 2-19 組合，在此段程式一開始，我們宣告了一個新的變數，用來放置我們的聊天室暱稱與要發送的訊息，透過第一個判斷式判斷是否有訊息要發出，也就是說我們沒辦法發出空白訊息，如果你想發出空白訊息的話，可以將此判斷式刪去或是改成大於等於零的時候發出訊息；利用變數 mode 來判別現在是那一端要發送訊息，最後我們將發送的時間、暱稱和發送的訊息顯示在螢幕上，成功的將藍牙聊天室完成了。

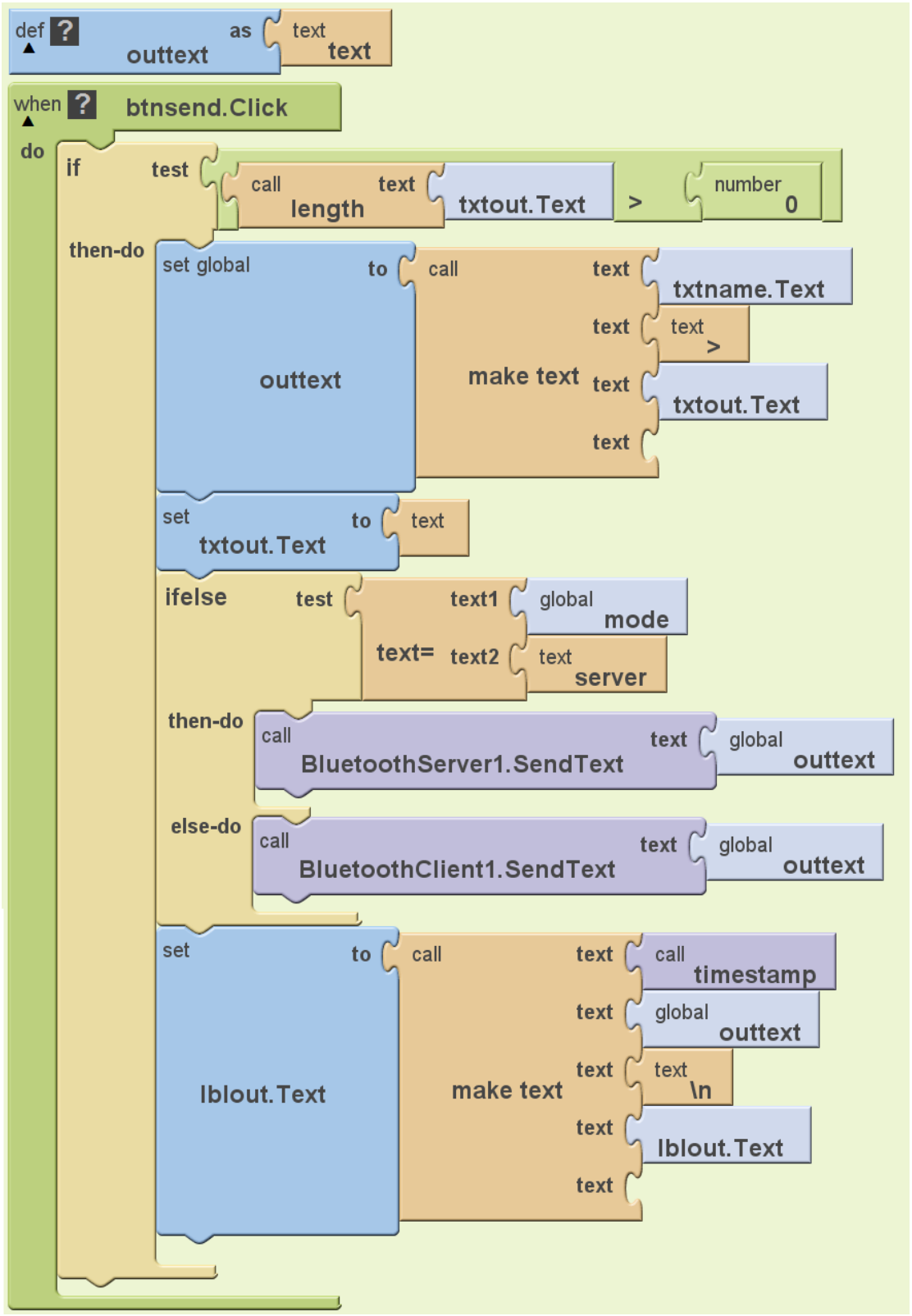
def ? outtext as text text
when ? btnsend.Click
do
if test call length text txtout.Text > number 0
then-do
set global outtext to call make text text txtname.Text text text > text txtout.Text text
set txtout.Text to text
ifelse test text= text1 global mode text2 text server
then-do call BluetoothServer1.SendText text global outtext
else-do call BluetoothClient1.SendText text global outtext
set lblout.Text to call make text text call timestamp text global outtext text text \n text lblout.Text text

圖 2-19 傳送聊天訊息

2-3 藍牙繪圖板

在前面的幾個範例中，我們已經對藍牙連線與藍牙伺服器的設定做了完整的介紹，在這個藍牙繪圖板的範例中，我們使用相同的設定方式，和第二個範例”藍牙聊天室”不一樣的地方是將傳送的資料從文字訊息變成圖片訊息，而在螢幕上繪畫出點、線、圓的方法，在前面章節的範例中都已經有詳細的介紹，因此在這裡我們不再對藍牙繪圖板做說明，如果你對這個範例仍有疑問，可以參考前面章節的繪圖範例，或是參考隨書附贈的光碟中，已經寫好的藍牙繪圖板程式。

2-4 總結

還記得你第一次將這個章節翻開來的時候，發現說這個章節怎麼比其它的章節更難、用了更多的判別式，程式也相對的更龐大了，但當你念完的時候，你會發現其實藍牙程式可以寫的很簡單，甚至你將一大堆判別式全都刪掉，程式仍然可以正常執行，那為什麼我們仍要執著於寫這麼多的判別式呢?這是為了使程式可以更 Robust(強健性)。何謂 Robust(強健性)?在工程上我們會解釋說:如果一個系統或是一個程式，因為一個很小的變因或變數，而使個系統容易變得不穩定，程式容易出現錯誤，那我們就會說這個系統或程式不夠 Robust；而要如何才能讓一個程式變得更 Robust?答案是要在程式容易發生錯誤的地方用更多的判斷式去避免錯誤發生。就拿藍牙聊天室這個範例來說明，有可能出現大問題的地方有:

1.選取的藍牙裝置是否有存放到暫存資料的空間中(TinyDB)

2.藍牙連線是否有配對成功

3.藍牙伺服器是否中途斷線

4.訊息是否有成功接收

將這些問題考慮進去，透過判別式來防止程式的錯誤，這樣才是一個成功的程式設計。

2-5 實力評量

1.(　)TinyDB 只能將資料存入，不能將資料讀出。

2.(　)藍牙伺服器可以用一對多的方式，與多位使用者同時連線。

3.(　)一個使用者可以與多個藍牙伺服器同時連線。

4.請在藍牙繪圖板的範例中，使用 Canvas.Touched 指令畫出一條線。

5.請在藍牙繪圖板的範例中，使用 Canvas.Touched 指令畫出長方形。

6.請在藍牙繪圖板的範例中，使用 Canvas.Touched 指令畫出圓。