其他元件 - Wi-Fi 通訊

作者:林煜衡

本章介紹

因為 App Inventor 2 中的 Wi-Fi 的模組是控制 Wi-Fi 開關或者連接,沒辦法傳送 或接收資料來控制 Arduino,所以是透過 Web 元件與 Button 開啟網頁,傳送資 料,並在 Arduino 製作一個簡易的 web server,使 App Inventor 2 的 Button 連接 網頁,或是在 URL 中直接輸入 IP 位置開啟網頁,以達成 Arduino 的 LED 開關 控制。

<u>Wi-Fi</u>

Wi-Fi 在中文裡又稱作「無線熱點」,是一個無線網路通信技術的品牌,由 Wi-Fi 聯盟(Wi-Fi Alliance)所持有,由無線乙太網相容聯盟所發佈的業界術語,指的 是無線資料傳輸技術與規格。Wi-Fi 採用 IEEE 802.11 系列標準之無線區域網路 (Wireless Local Area Network / Wireless LAN; WLAN)。使用 Wi-Fi 無線區域網 路可讓使用者透過數據機在 300 英尺範圍內(約 91 公尺)無線上網。Wi-Fi 網路傳 輸速度快,但距離短,多半設在咖啡店或機場等處,提供使用者穩定平順的無線 上網通訊。

屬性	功能				
	設定是否可儲存網頁回應的				
set Web1 • . AllowCookies • to •	cookies [,] 並用於後續的網路要求。只				
Web1 v . AllowCookies v	有 Android 2.3 版以後才支援				
	cookies °				
	Request header 是一個包含兩個子				
	清單的清單。				
set Web1 . RequestHeaders to					
Web1 • . RequestHeaders •	每個子清單的第一個元素是欄位名				
	稱;第二個元素則是欄位值 [,] 可能是				
	單一值或是一個包含多值的清單。				
set Web1 • . ResponseFileName • to	用來儲存網頁回應的檔案名稱。如果				
	SaveResponse 欄位設定為 true 但未指				
	定 ResponseFileName 的話,就會自動				

Web 元件常用介紹:

	產生新的檔名。
set Web1 • . SaveResponse • to •	設定是否要將網頁回應存在檔案中。
set Web1 v . Url v to K Web1 v . Url v	網路要求的 URL 路徑。
事件	功能
when Web1 GotFile url responseCode responseType fileName do	執行完一次網路取得檔案之後,呼叫 本事件。
when Web1GotText url responseCode responseType responseContent do	執行完一次網路取得文字檔案之後, 呼叫本事件。
方法	功能
call Web1 ▼ .BuildRequestData list ∎	將一個包含兩個子清單的清單轉換為 應用程式格式的字串,可用來傳給 Po stText。
call Web1 .ClearCookies	清除本 Web 元件的所有 cookies。

	執行一次 HTTP DELETE 要求,須設
	定 Url 屬性之後才能取得回應。如果
	SaveResponse 屬性為 true,回應會存
call Web1 .Delete	在某個檔案之中並呼叫 GotFile 事件。
	ResponseFileName 屬性可用來指定該
	檔案檔名。如果 SaveResponse 屬性為
	false,就會呼叫 GotText 事件。
	執行一次 HTTP GET 要求,須設定
	Url 屬性之後才能取得回應。如果
	SaveResponse 屬性為 true,回應會存
call Web1 .Get	在某個檔案之中並呼叫 GotFile 事件。
•	ResponseFileName 屬性可用來指定該
	│ 「▲」 │檔案檔名。如果 SaveResponse 屬性為
	false [,] 就會呼叫 GotText 事件。
	對指定 HTML 文字進行解碼。像
	& 、< 、> 、'以及
call Web1 ▼ .HtmlTextDecode htmlText ►	
	" 這惊的 HIML 子兀曾彼胜竹
ntmitext	" 這樣的 HTML 字元曾被解析 為 &, <, >, ', 與 ''。像 &#xhhhh 與
	" 這樣的 HTML 字元曾被解析 為 &, <, >, ', 與 ''。像 &#xhhhh 與 &#nnnn 會被解析為對應的字元。</th></tr><tr><th></th><th>& quot; 這樣的 HTML 字元曾被解析 為 &, <, >, ', 與 ''。像 & #xhhhh 與 & #nnnn 會被解析為對應的字元。 解碼指定的 JSON 物件以產生對應的</th></tr><tr><th></th><td>& quot; 這樣的 HTML 字元曾被解析 為 &, <, >, ', 與 ''。像 &#xhhhh 與 & #nnnn 會被解析為對應的字元。 解碼指定的 JSON 物件以產生對應的 AppInventor 值。一個 JSON 清單 [x,</td></tr><tr><th></th><th>& quot; 這樣的 HTML 字元曾被解析 為 &, <, >, ', 與 ''。像 &#xhhhh 與 & #nnnn 會被解析為對應的字元。 解碼指定的 JSON 物件以產生對應的 AppInventor 值。一個 JSON 清單 [x, y, z] 會被解碼為 (x y z) 這樣的清單。</th></tr><tr><th>call Web1 .JsonTextDecode</th><td>& quot; 這樣的 HTML 字元曾被解析 為 &, <, >, ', 與 ''。像 &#xhhhh 與 & & #nnnn 會被解析為對應的字元。 解碼指定的 JSON 物件以產生對應的 AppInventor 值。一個 JSON 清單 [x, y, z] 會被解碼為 (x y z) 這樣的清單。 具有 name A 與 value B 的 JSON 物</td></tr><tr><th>call Web1 .JsonTextDecode</th><th>& quot; 這樣的 HTML 字元曾被解析 為 &, <, >, ', 與 ''。像 &#xhhhh 與 & & #nnnn 會被解析為對應的字元。 解碼指定的 JSON 物件以產生對應的 AppInventor 值。一個 JSON 清單 [x, y, z] 會被解碼為 (x y z) 這樣的清單。 具有 name A 與 value B 的 JSON 物 件,例如 {name:123}, 會被解碼為會</th></tr><tr><th>call Web1 v .JsonTextDecode</th><th>& quot; 這樣的 HTML 字元曾被解析 為 &, <, >, ', 與 ''。像 &#xhhhh 與 & & #nnnn 會被解析為對應的字元。 解碼指定的 JSON 物件以產生對應的 AppInventor 值。一個 JSON 清單 [x, y, z] 會被解碼為 (x y z) 這樣的清單。 具有 name A 與 value B 的 JSON 物 件,例如 {name:123}, 會被解碼為會 被解碼為 ((name 123)) 這樣的清單,</th></tr><tr><th>call Web1 .JsonTextDecode</th><th>& quot; 這樣的 HTML 字元曾被解析 為 &, <, >, ', 與 ''。像 &#xhhhh 與 & & #nnnn 會被解析為對應的字元。 解碼指定的 JSON 物件以產生對應的 AppInventor 值。一個 JSON 清單 [x, y, z] 會被解碼為 (x y z) 這樣的清單。 具有 name A 與 value B 的 JSON 物 件,例如 {name:123}, 會被解碼為會 被解碼為 ((name 123)) 這樣的清單, 也就是一個包含一個二元素清單</th></tr><tr><th>call Web1 .JsonTextDecode</th><th>& quot; 這樣的 HTML 字元曾被解析 為 &, <, >, ', 與 ''。像 &#xhhhh 與 & #nnnn 會被解析為對應的字元。 解碼指定的 JSON 物件以產生對應的 AppInventor 值。一個 JSON 清單 [x, y, z] 會被解碼為 (x y z) 這樣的清單。 具有 name A 與 value B 的 JSON 物 件,例如 {name:123}, 會被解碼為會 被解碼為 ((name 123)) 這樣的清單, 也就是一個包含一個二元素清單 (name 123) 的清單。</th></tr></tbody></table>

	執行一次 HTTP POST 要求 · 須設定
	Url 屬性與指定檔案的路徑資料 · 如果
	SaveResponse 屬性為 true,回應會存
call Web1 .PostFile	在某個檔案之中並呼叫 GotFile 事件。
path 🕻	
	ResponseFileName 屬性可用來指定該
	檔案檔名。如果 SaveResponse 屬性為
	false,就會呼叫 GotText 事件。
	執行一次 HTTP POST 要求 · 須設定
	Url 屬性與指定文字。文字字元會以
	UTF-8 進行編碼。如果 SaveResponse
call Web1 .PostText	屬性為 true,回應會存在某個檔案之
text	中並呼叫 GotFile 事件。
	ResponseFileName 屬性可用來指定該
	ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為
	ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為 false,就會呼叫 GotText 事件。
	ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為 false,就會呼叫 GotText 事件。 執行一次 HTTP POST 要求,需設定
	ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為 false,就會呼叫 GotText 事件。 執行一次 HTTP POST 要求,需設定 Url 屬性與指定文字。文字字元會以指
	ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為 false,就會呼叫 GotText 事件。 執行一次 HTTP POST 要求,需設定 Url 屬性與指定文字。文字字元會以指 定格式來進行編碼。如果
call Web1 • .PostTextWithEncoding	ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為 false,就會呼叫 GotText 事件。 執行一次 HTTP POST 要求,需設定 Url 屬性與指定文字。文字字元會以指 定格式來進行編碼。如果 SaveResponse 屬性為 true,回應會存
call Web1 ▼ .PostTextWithEncoding text ■ encoding ■	ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為 false,就會呼叫 GotText 事件。 執行一次 HTTP POST 要求,需設定 Url 屬性與指定文字。文字字元會以指 定格式來進行編碼。如果 SaveResponse 屬性為 true,回應會存 在某個檔案之中並呼叫 GotFile 事件。
call Web1 ▼ .PostTextWithEncoding text ■ encoding ■	ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為 false,就會呼叫 GotText 事件。 執行一次 HTTP POST 要求,需設定 Url 屬性與指定文字。文字字元會以指 定格式來進行編碼。如果 SaveResponse 屬性為 true,回應會存 在某個檔案之中並呼叫 GotFile 事件。 ResponseFileName 屬性可用來指定該
call Web1 ▼ .PostTextWithEncoding text ■ encoding ■	ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為 false,就會呼叫 GotText 事件。 執行一次 HTTP POST 要求,需設定 Url 屬性與指定文字。文字字元會以指 定格式來進行編碼。如果 SaveResponse 屬性為 true,回應會存 在某個檔案之中並呼叫 GotFile 事件。 ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為

	執行一次 HTTP PUT 要求,須設定 Url
	屬性與指定檔案的路徑資料。如果
	SaveResponse 屬性為 true,回應會存
call Web1 ▼ .PutText text ►	在某個檔案之中並呼叫 GotFile 事件。
	ResponseFileName 屬性可用來指定該
	檔案檔名。如果 SaveResponse 屬性為
	false,就會呼叫 GotText 事件。
	執行一次 HTTP PUT 要求·須設定 Url
	屬性與指定文字。文字字元會以
	UTF-8 進行編碼。如果 SaveResponse
	屬性為 true,回應會存在某個檔案之
call Web1 .PutFile	中並呼叫 GotFile 事件。
path	
	ResponseFileName 屬性可用米指定該
	備 条 備 石 。 如 未 SaveResponse
	property 為 false,
	事件。
	事件。 執行一次 HTTP PUT 要求,須設定 Url
	事件。 執行一次 HTTP PUT 要求,須設定 Url 屬性與指定文字.文字中的字元會以指
	事件。 執行一次 HTTP PUT 要求,須設定 Url 屬性與指定文字.文字中的字元會以指 定 編 碼 格 式 來 進 行 編 碼 。 如 果
call Web1 • .PutTextWithEncoding	事件。 執行一次 HTTP PUT 要求,須設定 Url 屬性與指定文字.文字中的字元會以指 定編碼格式來進行編碼。如果 SaveResponse 屬性為 true,回應會存
call Web1 ▼ .PutTextWithEncoding text ► encoding ►	事件。 執行一次 HTTP PUT 要求·須設定 Url 屬性與指定文字.文字中的字元會以指 定編碼格式來進行編碼。如果 SaveResponse 屬性為 true,回應會存 在某個檔案之中並呼叫 GotFile 事件。
call Web1 ▼ .PutTextWithEncoding text encoding	 事件。 執行一次 HTTP PUT 要求·須設定 Url 屬性與指定文字.文字中的字元會以指 定編碼格式來進行編碼。如果 SaveResponse 屬性為 true,回應會存 在某個檔案之中並呼叫 GotFile 事件。 ResponseFileName 屬性可用來指定該
call Web1 ▼ .PutTextWithEncoding text ■ encoding ■	事件。 執行一次HTTPPUT要求,須設定Url 屬性與指定文字.文字中的字元會以指 定編碼格式來進行編碼。如果 SaveResponse屬性為true,回應會存 在某個檔案之中並呼叫GotFile事件。 ResponseFileName屬性可用來指定該 檔案檔名。如果SaveResponse屬性為
call Web1 ▼ .PutTextWithEncoding text ■ encoding ■	 事件。 執行一次HTTPPUT要求,須設定Url 屬性與指定文字.文字中的字元會以指 定編碼格式來進行編碼。如果 SaveResponse 屬性為 true,回應會存 在某個檔案之中並呼叫GotFile事件。 ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為 false,就會呼叫GotText事件。
call Web1 ▼ .PutTextWithEncoding text ■ encoding ■	事件。 執行一次HTTPPUT要求,須設定Url 屬性與指定文字.文字中的字元會以指 定編碼格式來進行編碼。如果 SaveResponse 屬性為 true,回應會存 在某個檔案之中並呼叫GotFile事件。 ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為 false,就會呼叫GotText事件。 將指定文字進行編碼,讓它可用在
call Web1 ▼ .PutTextWithEncoding text ■ encoding ■	事件。 執行一次HTTPPUT要求·須設定Url 屬性與指定文字.文字中的字元會以指 定編碼格式來進行編碼。如果 SaveResponse屬性為true,回應會存 在某個檔案之中並呼叫GotFile事件。 ResponseFileName屬性可用來指定該 檔案檔名。如果SaveResponse屬性為 false,就會呼叫GotText事件。 將指定文字進行編碼,讓它可用在 URL之中。
call Web1 • .PutTextWithEncoding text • encoding •	事件。 執行一次HTTPPUT要求,須設定Url 屬性與指定文字.文字中的字元會以指 定編碼格式來進行編碼。如果 SaveResponse 屬性為 true,回應會存 在某個檔案之中並呼叫GotFile事件。 ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為 false,就會呼叫GotText事件。 將指定文字進行編碼,讓它可用在 URL之中。 在某些情況下會需要在URL中輸入中
call Web1 • .PutTextWithEncoding text encoding	 事件。 執行一次HTTPPUT要求,須設定Url 屬性與指定文字.文字中的字元會以指 定編碼格式來進行編碼。如果 SaveResponse 屬性為 true,回應會存 在某個檔案之中並呼叫GotFile事件。 ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為 false,就會呼叫GotText事件。 將指定文字進行編碼,讓它可用在 URL之中。 在某些情況下會需要在URL中輸入中 文字,但是直接輸入中文字可能會因
call Web1 • .PutTextWithEncoding text encoding	事件。 執行一次HTTPPUT要求,須設定Url 屬性與指定文字.文字中的字元會以指 定編碼格式來進行編碼。如果 SaveResponse 屬性為 true,回應會存 在某個檔案之中並呼叫GotFile事件。 ResponseFileName 屬性可用來指定該 檔案檔名。如果 SaveResponse 屬性為 false,就會呼叫GotText事件。 將指定文字進行編碼,讓它可用在 URL之中。 在某些情況下會需要在URL中輸入中 文字,但是直接輸入中文字可能會因 為編碼問題而產生亂碼。因此需要此

範例-利用手機 Wi-Fi 控制 LED 開關

步驟一.

首先使用 Arduino 連接 Wi-Fi 模組(ESP-8266),目前在 Arduino 使用 Wi-Fi 模組的方法分為2種,如果開發板上沒有內建 Wi-Fi 晶片需要另外購買 Wi-Fi 模組連接 Arduino,連接時要請注意 Wi-Fi 模組的正極只能連接 3.3V,否則 會使晶片燒毀。基本上兩種方式的程式碼概念是一樣的,只有在 Wi-Fi 的設 定方式不同。

本範例使用內建有 Wi-Fi 模組的 NodeMcu 開發版,若使用無內建 Wi-Fi 模 組的開發版則必須將 ESP-8266 的 RX 接 Arduino 的 0; TX 接 1; VCC 和 CH_PD 接 3.3V; GND 接 GND。

LED 燈的正極則分別將接在開發版上的 PIN 接腳上、負極接 GND,一般開發版的接腳名稱為 1~13 的數字,部分開發版的接腳名稱會在數字前將上字母 D(例如:D1、D2、D3),通常 0、1 接腳是用來與其他開發版或通訊模組溝通在連接其他模組時盡量避免使用。



Wi-Fi 模組(ESP-8266、ESP-01) 內建 Wi-Fi 模組的開發板(Arduino UNO WIFI)



ESP-8266 與 Arduino 連接範例

步驟二.

● 連接成功後將 Arduino 程式碼上傳至開發板並找出 Arduino 的 IP · 上傳完成 後開啟 Serial Monitor 就能看到 Arduino 的 IP · 不同的設備 IP 可能會不同 · 請確認 IP 位置無誤。

1 2	<pre>#include <softwareserial.h> #include <esp8266wifi.h></esp8266wifi.h></softwareserial.h></pre>
3 4 5	ESP8266 WiFi(Serial1); 請輸入要連接的無線網路的名稱(SSID)及密碼(PASSWORD)
6 7 8	<pre>char ssid[] = "ssid"; char pass[] = "pass"; int keyIndex = 0;</pre> // // you 連接時請將 NodeMcu 與手機連接到同一無線網路設備,並確
9 10 11	int status = WL_IDLE_STATUS; 認裝置在同一區域網路內。 WiFiServer server(80);
12 13 • 14 15 16 17 18 19	<pre>void setup() { Serial.begin(9600); // initialize serial communication pinMode(LED_BUILTIN, OUTPUT);</pre>
20 21 • 22 23	<pre>// attempt to connect to Wifi network: while (status != WL_CONNECTED) { Serial.print("Attempting to connect to Network named: "); Serial.println(ssid); // print the network name (SSID);</pre>
24 25 26	<pre>// Connect to WPA/WPA2 network. Change this line if using open or WEP network: status = WiFi.begin(ssid, pass);</pre>
27 28 29 30 31	<pre>} server.begin();</pre>
32 33 • 34 35	<pre>void loop() { WiFiClient client = server.available(); // listen for incoming clients</pre>
36 * 37 38 39 * 40 * 41 42 43 *	<pre>if (client) { // if you get a client, Serial.println("new client"); // print a message out the serial port String currentLine = ""; // make a String to hold incoming data from the client while (client.connected()) { // loop while the client's connected if (client.available()) { // if there's bytes to read from the client, char c = client.read(); // read a byte, then Serial.write(c); // print it out the serial monitor if (c == '\n') { // if the byte is a newline character</pre>
44 45 46 47 • 48 49 50 51 52 52	<pre>// if the current line is blank, you got two newline characters in a row. // that's the end of the client HTTP request, so send a response: if (currentLine.length() == 0) { // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK) // and a content-type so the client knows what's coming, then a blank line: client.println("HTTP/1.1 200 OK"); client.println("Content-type:text/html"); client.println();</pre>
55 54 55 56 57 58 59 60	<pre>// the content of the HTTP response follows the header: client.print("Click here to light red "); client.print("Click here to light green "); client.print("Click here to light blue "); client.print("Click here to light blue "); client.print("Click here to light white "); client.print("Click here to light off br>"); client.print("Click here to light off br>");</pre>
61 62 63 64 65 • 66 67	<pre>// The HTTP response ends with another blank line: client.println(); // break out of the while loop: break; } else { // if you got a newline, then clear currentLine: currentLine = ""; }</pre>
68 * 69 70 71	<pre>} else if (c != '\r') { // if you got anything else but a carriage return character,</pre>

72 // Check client request if (currentLine.startsWith("GET /R")) { // GET /R to light red 73 🔻 74 led(1, 0, 0); 75 if (currentLine.startsWith("GET /G")) { // GET /G to light green 76 🔻 77 led(0, 1, 0); 78 if (currentLine.startsWith("GET /B")) { // GET /B to light blue 79 🕈 80 led(0, 0, 1); 81 if (currentLine.startsWith("GET /W")) { // GET /W to light white 82 • 83 led(1, 1, 1); 84 if (currentLine.startsWith("GET /o")) { // GET /o to light off 85 • 86 led(0, 0, 0); 87 3 } 88 89 } 90 // close the connection: client.stop(); 91 92 Serial.println("client disonnected"); 93 3 94 } 95 void printWifiStatus() { 96 🕈 // print the SSID of the network you're attached to: Serial.print("SSID: "); 97 98 99 Serial.println(WiFi.SSID()); 100 // print your WiFi shield's IP address: 101 102 IPAddress ip = WiFi.localIP(); 103 Serial.print("IP Address: "); 104 Serial.println(ip); 105 106 // print the received signal strength: 107 long rssi = WiFi.RSSI(); 108 Serial.print("signal strength (RSSI):"); 109 Serial.print(rssi); 110 Serial.println(" dBm"); 111 // print where to go in a browser: 112 Serial.print("To see this page in action, open a browser to http://"); 113 Serial.println(ip); 114 } 115 116 • void led(int R, int G, int B) { 117 digitalWrite(11, R); 118 digitalWrite(9, G); 119 digitalWrite(7, B); 120 121 Serial.print(G);
Serial.print(","); 122 123 Serial.print(", 124 Serial.print(B); 125 }

💿 COM4 (Linklt 7697)	_		\times
			Send
Host: 192.168.1.73			^
Connection: keep-alive			
Pragma: no-cache			
Cache-Control: no-cache			
User-Agent: Mozilla/5.0 (Windows NT 10.0; Windows NT 10.0)	54; x64) App	leWel
Save-Data: on			
Accept: image/webp, image/apng, image/*,*/*;q=0	. 8		
Referer: http://192.168.1.73/			
Accept-Encoding: gzip, deflate			
Accept-Language: zh-TW,zh;q=0.8,en-US;q=0.6,en	n;q=0.4	,zh-Cl	N;q=(
client disonnected			
new client			
client disonnected			
			~
<			>
Autoscroll No line ending V 9600 ba	uud ~	Clear	r output

請用電腦或手機連接 Arduino,並開啟瀏覽器輸入在步驟二得到的 IP 位置,測試 是否有成功連接,如果有可以試著點擊網頁的超連結,看看 Arduino 上的變化。

步驟三.

● 建立 App Inventor 2 專案 · 新增版面及程式碼。

Viewer				Components
	Display hidden con Check to see Previo WiFi LED Control -	nponents in Viewer ew on Tablet size. Linklt 7697	⊜"∥ 9:48	Screen1 HorizontalScrollArrangerr Button_setIP Al abal status
	Set IP	Wait Cor	nnection	TextBox1
				TableArrangement1 Button_G Rutton_W
	R	G	В	Button_off Button_R Button_B
	off	W		A Label1
				< Rename Delete
	Ĵ	\Box	Ū	Media
	No	n-visible componer O Web1	nts	Upload File

步驟四.

● 設定連接 Arduino 的 IP。

initia	lize global 📺	to (" htt	p://192.168.1.7	3″	請輸入	在步驟_	得到的	IP 位置
whe	n Button_se	tlP 🔹 .Click						
do	🖸 if 📢	C TextBox1 •). Text ▼ 🗲	•	" 🔲 "			
	then set	Web1 🔹 . Ur	🔽 to 🕅 Te	xtBox1 🔹	. Text 🔹			
	set	Label_status	. Text 🔹 t	o 🛛 호	join (¦ ″	Set IP to: "		
					(Tex	tBox1 🔻 . (Te	xt 🔹	
	else set	Label_status	🖌 . (Text 🔹) t	o († "	input your l	inklt IP		

步驟五.

● 自訂傳送函數。

	to sendData mess	age (command)	
do	set Web1 🔹 . Url	🔹 to 🜔 😒 join 🌔	get global ip 🔹
			get command 🔹
	set Label_status 🔻).Text 🔹 to 🚺 🙆	join (Web1 🔹 Url 💌
			(" 🔳 "
			🜔 get message 🔹
	call Web1 .Get		

步驟六.

● 設定 Button 的點擊觸發事件。



步驟七.

● 藉由傳送不同 message 與 command 在 Arduino 中進行 LED 的開關與顏色控

制。

<pre>// Check client request if (currentLine.startsWith("GET /R")) led(1,0,0);</pre>	11	GET	/R	to	light	red
<pre>} if (currentLine.startsWith("GET /G")) { led(0,1,0);</pre>	11	GET	/G	to	light	green
<pre>) if (currentLine.startsWith("GET /B"))) { led(0,0,1);</pre>	11	GET	/в	to	light	blue
<pre>} if (currentLine.startsWith("GET /W"))) { led(1,1,1);</pre>	11	GET	/₩	to	light	white
<pre>) if (ourrentLine.startsWith("GET /o"))) led(0,0,0);</pre>	11	GET	/0	to	light	off
)						

步驟八.

● 執行結果。



**	🗖 🕕 🛈 🕈 🧊 📶 70% 📄 15:20			
WiFi LED Control -	Linklt 7697			
Set IP	http://192.168.0.101/W: Forward			
check 7697 ip i	n serial monitor			
R	G	В		
off	W			

