實驗內容

1. Raspberry Pi 基本設定與功能測試
   - 作業系統

     由於我拿到的 Raspberry Pi 已經安裝了作業系統，我就沒有再從 flash OS image 開始做；我只有用 sudo apt update 跟 sudo apt -y dist-upgrade 更新系統。
   - USB-TTL

     我一開始拿到的 USB-TTL 傳輸線是有問題的，接到電腦上沒有辦法看到裝置。排除了線的問題以後，我有成功在 Linux 下透過 minicom 控制 raspberry pi。不過我家的電腦沒有 Linux，而我也沒有打算安裝，所以我原本想在 Windows 上透過 WSL（Windows Subsystem for Linux）和 minicom 控制 raspberry pi。但程式裝好以後我發現 WSL 不支援使用 USB 介面，所以也沒辦法用 USB-TTL。因為 WSL 的 USB 問題看起來不容易解決，我決定改用 mac 筆電接 raspberry pi。經過一番測試，我成功使用 minicom 和 macOS 內建的 screen 指令兩種方法控制 raspberry pi。但因為我是在 windows 上用 Mbed Studio，我覺得多開一台電腦有點麻煩，所以最後我還是採用 term emulator（MobaXterm）進行 serial communication。惟需注意使用 windows 的話需要下載舊版驅動程式（2009 年左右的版本），轉換晶片才能正常運作。[1]
   - SFTP server

     不需要特別設定，只要在 config 開啟 SSH 就能使用。
   - ble_scan_connect.py

     這個部分照著課程投影片的步驟做就好，在此不贅述。
2. Button 範例程式（mbed-os-example-ble-Button）

   我這次有更新 Mbed OS，出現的 problem 大部分照著課程投影片的指示處理就可以；課程投影片沒提到的基本上都是不難解決的問題。程式修改完成以後我用 BLE scanner 測試，沒有什麼問題。
3. LED 範例程式（mbed-os-example-ble-LED）

   同樣更新 Mbed OS；此範例程式需要處理的問題較少，不過會出現一個「'onDataWritten' is deprecated」的 warning。我花了一些時間研究之後還是無法解決，因為這不影響程式運作，所以我最後決定無視這個 warning。修改完成後同樣使用 BLE scanner 測試，確定可以從手機用 write 控制 STM32 上 LED 的開關。
4. 結合 Button 與 LED 範例程式

   把兩個 class（BatteryDemo 跟 LEDDemo）合在一起，並將 class 名稱改為

[1] 如果 USB-TTL 傳輸線用的是新版的晶片，用最新版的驅動程式應該就可以了。

BLE_HW。假如前兩個項目都沒有問題的話，合起來以後應該也不會遇到什麼問題，只要注意兩個 service（ButtonService 跟 LEDService）的 UUID 不能重複即可。

5. Notification 的接收

   Button service 其實就用了 notification，所以在此項就不另做一個有 notification 的 service，直接用 button service 來測試。

   - BLE Scanner

     這其實在第二項就操作過了，register notification 以後就可以看到 characteristic value 即時的變化。

   - Raspberry Pi

     原本的 ble_scan_connect.py 並沒有接收 notification 的功能，我參考[1]的內容，加入了接收 notification 的程式碼。值得注意的是在使用 bluepy 的 waitForNotifications()之前必須先開啟通知（見 enable_notify()）。我可以成功接收到通知，但處理通知內容的部分我還沒有成功。我目前的設定是等待五秒皆未收到通知時會顯示「Waiting…」，而收到通知時會顯示「Button pressed」；只要 print 的次數達到十次，程式就會結束。

6. 學號資料的提供與接收

   - 提供（STM32）

     寫一個新的 custom service「StuIDService」，此 service 有一個 private member 「 ReadOnlyArrayGattCharacteristic<char,9> stuID 」；在執行 main.cpp 時，此 characteristic 的值會被設定為學號。把這個 service 跟第四項得到的程式合在一起，完成以後會出現 warning「ISO C++11 does not allow conversion from string literal to 'char *'」；對這個警告我是直接無視，沒有多做研究。

   - 接收（BLE Scanner、Raspberry Pi）

     BLE scanner 的接收不需要特別設定，操作方法跟前面幾個項目完全相同。Raspberry Pi 的部分則是修改 read()之前 getCharacteristics()使用的 UUID 即可。

實驗結果

GitHub: https://github.com/chun9temp/2021ESLab/tree/main/HW4

執行的結果我用兩部演示影片呈現，一為「BLE Scanner.mp4」、一為「Raspberry Pi.mp4」，分別是用 BLE scanner 跟 Raspberry Pi 搭配執行相同程式的 STM32 得到的結果。STM32 使用的程式是最終版本的程式（即上傳到 GitHub 的版本），該程式包含了本次實驗的所有項目。

討論

1. minicom 的顯示效果其實沒有很好，顯示出來的畫面常常不太正確；不管是用 macOS 還是 Linux 都一樣。如果用 macOS 的 screen 指令或 windows 的 MobaXterm，設定上比較簡單，顯示效果也比較好。
2. 有兩個 warning 沒有消掉。我推測第一個（'onDataWritten' is deprecated）應該是新版的 Mbed OS 不希望處理單獨的 event handler，想要我們把所有會用到的 handler 合成一個包含個別 handler 的 handler 集合，再用 setEventHandler() 處理。但我還沒有研究出應該怎麼改。
3. 我現在還沒有辦法處理 notification 的 data；可能需要先搞清楚傳送時用的 datatype 和接收時預期收到的 datatype 分別是什麼。
4. 我沒有在 Raspberry Pi 上實作控制 LED 的功能（即 write value）。不過我認為這比接收通知簡單，參考一下 bluepy 的使用說明應該就能做出來。
5. EventQueue 會對使用 post 功能的 event 進行排程，並依序處理在 EventQueue 中的 event。
6. InterruptIn 會在接收到 interrupt（數位輸入發生改變）時立刻觸發 event。InterruptIn 不能使用等待、迴圈、print 之類可能造成系統停止的指令。

參考資料

[1] Code for notification? #124
https://github.com/IanHarvey/bluepy/issues/124