實驗內容

1. Wi-Fi 範例程式（mbed-os-example-wifi）
   首先 import 範例程式，我原本以為不更新 OS 的話直接跑應該不會出現問題，結果 compile 的時候出現了一堆錯誤，修正了一個又有另一個；而這些錯誤大部分都是 Mbed OS 裡面 python 程式的 print 指令沒有用 print()。這應該是因為以前的 Mbed OS 是用 python 2 寫的，而 Mbed Studio 是用 python 3 去跑。我原本想一個一個處理，但是發現問題根本無窮無盡，而且很多問題就算查了還是改不好，或者根本查不到解法，最後都只能註解掉。改了一段時間後，我發現自己好像是在改整個 OS，而這顯然不是我的工作。我也有思考過要不要改 Mbed Studio 的設定，讓它改用 python 2，但我查了一下發現似乎也不太容易。於是我決定還是更新 Mbed OS 跟 Wi-Fi 的 library，看哪種組合可以在 Mbed Studio 下成功 run；最後我使用的 library 是 mbed-os-5.15.7 以及 wifi-43362-mbed-os-5.15。另外，我在執行 http_demo 時發現所送出的 GET request 並沒有得到 OK 的 response；www.arm.com 是 301 Moved Permanently，而 ntu.edu.tw 則是直接出現 error。我懷疑是現在的網站幾乎都是用 https 的關係，於是我找了一個 http 的 host「info.cern.ch」測試，發現確實可以得到 200 OK 的 response。如果要連到 https 網站的話，需要處理 TLS 相關的內容，但那部分我就沒有嘗試了。
2. 感測器範例程式（DISCO_L475VG_IOT01-Sensors-BSP）
   一樣 import 範例程式，但考慮到前一項目最後使用的是 mbed-os-5.15.7，之後又需要結合 Wi-Fi 範例程式跟感測器範例程式，所以我決定先把 Mbed OS 升級到相同的版本，看是否能在該版本下運作。升級後並沒有遇到什麼特別的問題，只要把 ThisThread::sleep_for()改成 wait_us()就可以了。
3. 結合 Wi-Fi 與感測器範例程式
   在 Wi-Fi 範例程式加入 sensor 的 library（BSP_B-L475E-IOT01 tip），並將感測相關的程式碼放到 main.cpp 裡面。注意要包含開頭#include 以及 main 裡面 init 的內容，如果程式碼沒有遺漏的話，就可以經由此程式同時使用 Wi-Fi 與感測器的功能。
4. 將感測資料傳到 Host（socket server）
   參考上課投影片以及[1]，簡化與修改部分程式碼，實作 socket 的傳送與接收。STM32 的 main.cpp 加入 void data_sending(NetworkInterface *net)，把溫度、濕度、壓力以及 accelerometer、gyroscope、magnetometer 得到的資訊集合成字串，再用 socket.send()傳出去。Socket server 的部分，我是在 Windows 電腦上用 python 做，內容跟參考資料相去不遠。寫好以後在兩個裝置上執行程式；雖然 STM32 的 console 都沒有出現錯誤，但 host 端一直收不到 STM32 傳出來的資料。我以為是 server 程式或 STM32 上的程式的問題，試了很久才想到

可能是 STM32 連的是訪客網路的關係；我在訪客網路下沒辦法連到 router 的管理介面，或許在訪客網路上不能連到區域網路上的其他設備。結果改連非訪客網路以後，socket server 就成功收到資料了。

5. 視覺化感測資料（以網頁方式呈現）

   我沒有辦法從頭開始寫網頁，所以我在網路上找到 [2]、[3] 兩種範例程式，打算以其為基礎進行修改。

   - Flask + Socket.IO [2]

     這個範例程式的做法頗為單純；Server 會 host 一個以 main.html 為範本的網站，同時接收 client_test.py 傳來的 Socket.IO，當_mode 在 start 的時候就用 Socket.IO 裡的資訊進行繪圖。此外，這個範例的一大優點是繪圖的部分有做出 start、stop、reset 的功能，所以我一開始先嘗試的是這個。但我光是為了測試這個程式是否能正常運作就花了一些時間，關鍵在於把 server.py、client_test.py 跟 main.html[1]裡面的 localhost 都改成裝置的 IP。確定程式 server 跟 client 都能運作以後，我讓 server 接收 STM32 傳來的 socket，結果 server 顯示有收到東西（accept）卻讀不出資料。經過一番研究之後，我以為是我傳的 socket 沒有 eventName 的關係，於是我就想到兩種解法，一種是看 Socket.IO 是怎麼把 eventName 加在 socket 資訊裡面的，如果只是在開頭多加一些資訊的話，改一下 STM32 傳出來的 socket 內容就好了；另一種是看 server 有沒有辦法收沒有 eventName 的 socket。前者我查不太到資料；後者一部分是可行的，如果 client 用 send() 而非 emit()傳，確實是可以不用設定 eventName 的。我用 client_test.py 測試過，client 用 send()、server 用 on('message')的話確實是收得到資料的。但我把 server 中 on 的選項改成'message'或'json'都沒有用，還是讀不出 STM32 傳出來的資訊，只好先放棄這個方法。我後來認為這是 Socket.IO 跟 Mbed 的 TCPSocket 沒辦法直接相容的關係，請見討論的第 2 項。

   - Plotly + WebSocket [3]

     這個做法看起來是用 pywebsocket 的 standalone server 收 WebSocket，然後網頁（plot_graphs_from_websocket.html）會傳 request 給 server 並接收 server 傳回來的 WebSocket，並用 Plotly 畫出 WebSocket 中的資料；而 server 傳 WebSocket 給網頁的時候，又是用 send_graph_wsh.py 透過 WebSocket 傳來的資料。所以如果我要用[3]提供的 send_graph_wsh.py 改的話，變成要開一個 server 接收 STM32 傳來的 socket，把資料取出來後再用 send_graph_wsh.py 傳到 standalone server。但我不太能確定在 send_graph_wsh.py 裡面多做一個 socket server 是不是可行；如果純粹 import server.py 的資料的話，我也不確定怎麼即時更新變數的值。而且這個做法有點複雜，開兩個 server 好像有點多此一舉，所以我失敗了幾次以後就沒有再繼續嘗試。

[1] 第 46 行 io.connect 的 input 不用加 http://。

6. 視覺化感測資料（在 Host 端呈現）

用 json.loads 把 socket 傳來的資料轉成 python 的 dict，再把 12 種感測資料存成 list，用新的資料不斷 append list；但 list 的長度會維持在「buf」的值，超過的話會把第一個元素刪掉再 append 新資料。最後用 matplotlib.plot 進行繪圖；為了讓 matplotlib 連續繪圖，我參考[4]裡面提到的做法，最後用 clf()跟 pause(0.001)的組合達成目標。我在測試時使用的 buf 值是 30，而 STM32 傳送資料的間隔是 1 秒鐘，所以圖表會呈現 30 秒以前到現在的資料。另外，STM32 因為只需要 socket client 的功能，我簡化了 main.cpp 內容，把 scan_demo 跟 http_demo 的部分註解掉。

實驗結果

GitHub: https://github.com/chun9temp/2021ESLab/tree/main/HW3

一開始的兩個範例程式因為包含在第三項「結合 Wi-Fi 與感測器範例程式」裡，所以我在 GitHub 上只有放第三項的結果，請參考圖 1 以及 Results 資料夾內的文字檔「serial-output_3.txt」。第四項因為包含在第六項、第五項因為沒有成功，所以都沒有在此展示結果。第六項視覺化的部分，STM32 跟 socket server 在執行時的畫面可參考 Results 資料夾內的文字檔「serial-output_6.txt」和「Socket Server.png」（圖 2）；視覺化的結果請參考 Results 資料夾內的影片「Visualization.mp4」以及視窗截圖「Visualization.png」（圖 3）。從該影片可以看到感測資料的 buffer 是 30 筆；1:02 的時候我有拿起 STM32 並旋轉，可以看到 accelerometer、gyroscope、magnetometer 的數值都有明顯的改變。

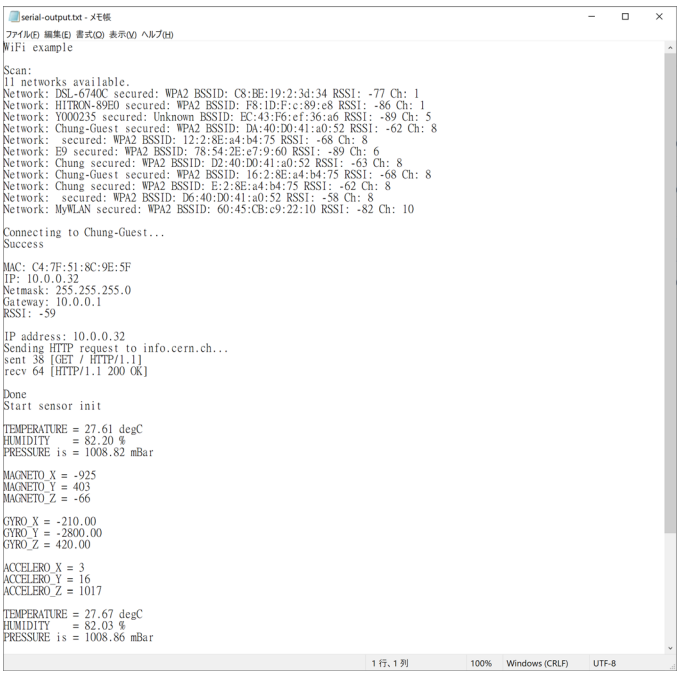

圖 1 結合 Wi-Fi 與感測器範例程式得到的結果

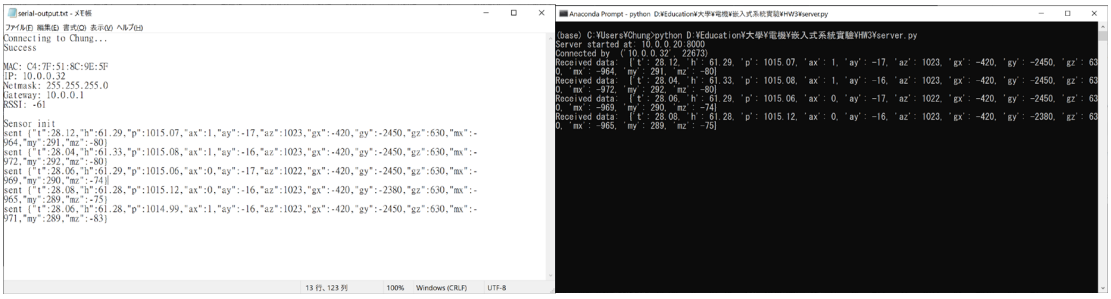

圖 2 Socket client（左）跟 server（右）執行時的畫面

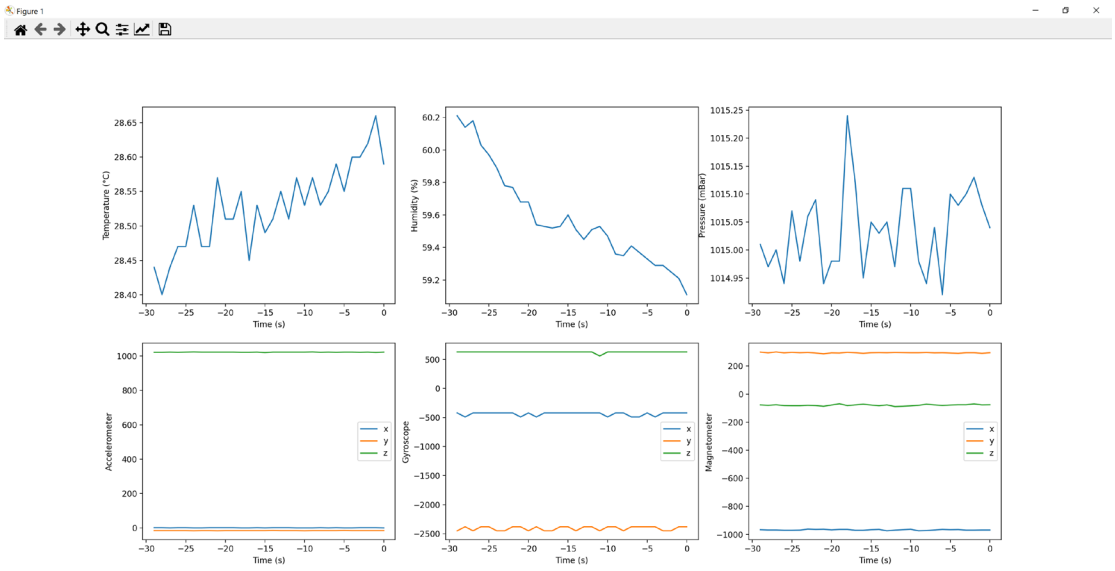

圖 3 感測資訊視覺化的結果

討論

1. 看起來讓訪客用訪客網路確實會比較安全。
2. [2]用 localhost 或 127.0.0.1 會出現問題，必須用實際的 IP 才能正常運作，不確定是不是 windows 的限制。
3. 關於用網頁進行視覺化並沒有成功的部分，我認為最大的問題是我一開始沒有意識到 Socket、WebSocket、Socket.IO 是三種不同的東西，而一直想要用 Mbed 的 TCPSocket 做。如果之後有需要做類似的東西的話，可能選 WebSocket 或 Socket.IO 做會簡單一點。
4. Matplotlib 即時繪圖的效果不是很好，介面非常卡，不確定是不是有更好的解法。可能找以即時繪圖為目標的視覺化程式來用會比較快。
5. 頻繁連線時常會連不上網路；不確定是無線路由器的問題還是 STM32 的問題。
6. 資料傳輸間隔時間較短（1 秒以下）時，傳輸結果並不是非常穩定，有時候會

出現 JSONDecodeError: Extra Data。

參考資料

[1] How to set up current work: Dev.27th
https://riino.site/2019/12/27/How-to-set-up-current-work-of-MotionDetect.html

[2] punkyoon / live-graph
https://github.com/punkyoon/live-graph

[3] Realtime graphs using Plotly and websockets
http://whatimade.today/realtime-graphs-using-ploty-and-websockets/

[4] Plotting in a non-blocking way with Matplotlib
https://stackoverflow.com/questions/28269157/plotting-in-a-non-blocking-way-with-matplotlib