

# 2017-2 Embedded System

Lab2: Tessel 2 Project

B02901137 吳元魁 / B02901011 趙祐毅

github: <https://github.com/ywk991112/ESLab-hw2>

Demo video: <https://www.youtube.com/watch?v=Bf1pBLWhoic>

Date: 2017/5/4

- 問題定義：我們都曾經面臨，電風扇不夠涼，或是太冷的情況，每次都要手動去調整旋鈕，我們希望透過環境溫溼度監測，自動調整電風扇的旋鈕，維持環境的舒適度。以下說明我們的 Project。

## 1. 分工

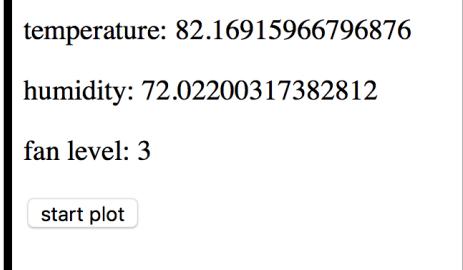
- Tessel 2 climate module, servo module, web server: 趙祐毅
- Tessel 2 testing, web client: 吳元魁

## 2. 程式 Platform

- Javascript on Tessel 2.
- Socket: Socket.io
- Module bundler: webpack

## 3. 系統具備哪些功能

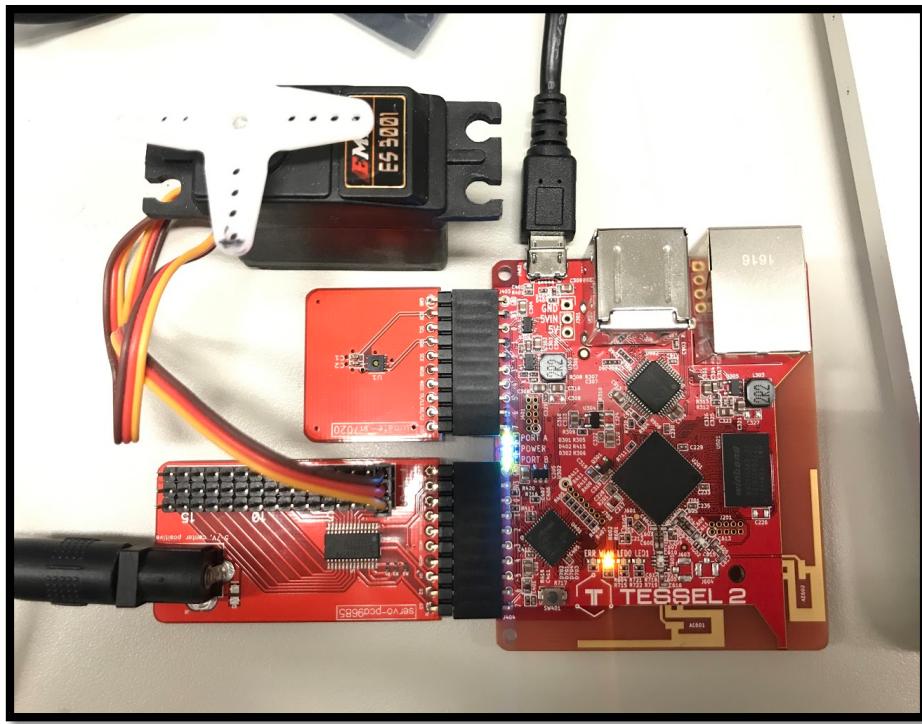
- Input: 透過 Climate module 量測環境溫度、濕度
- Data presentation: 顯示在 web 上 (localhost)，並且 Plot 出 Temperature 和 Humid。
- Output: 後端進行處理，並控制風扇強度 (0~5 級)，數字越大代表風力越強，用 Servo 馬達的角度代表不同級別。



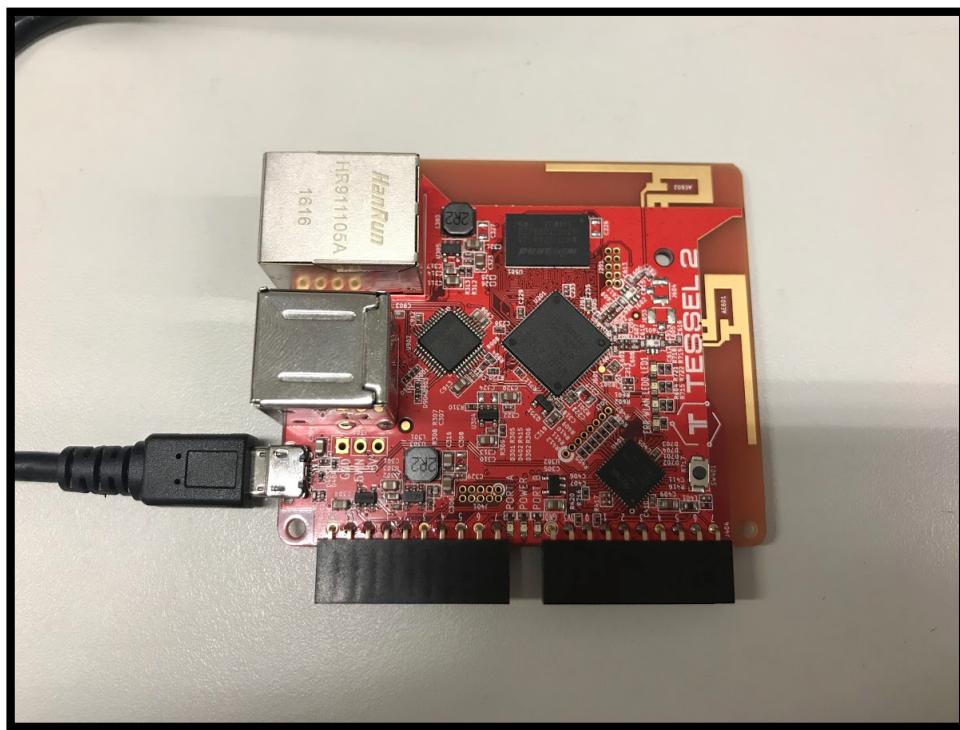
## 4. 系統架構

### a. 系統包含哪些子元件 (硬體 & 軟體) ?

- All (Connected)



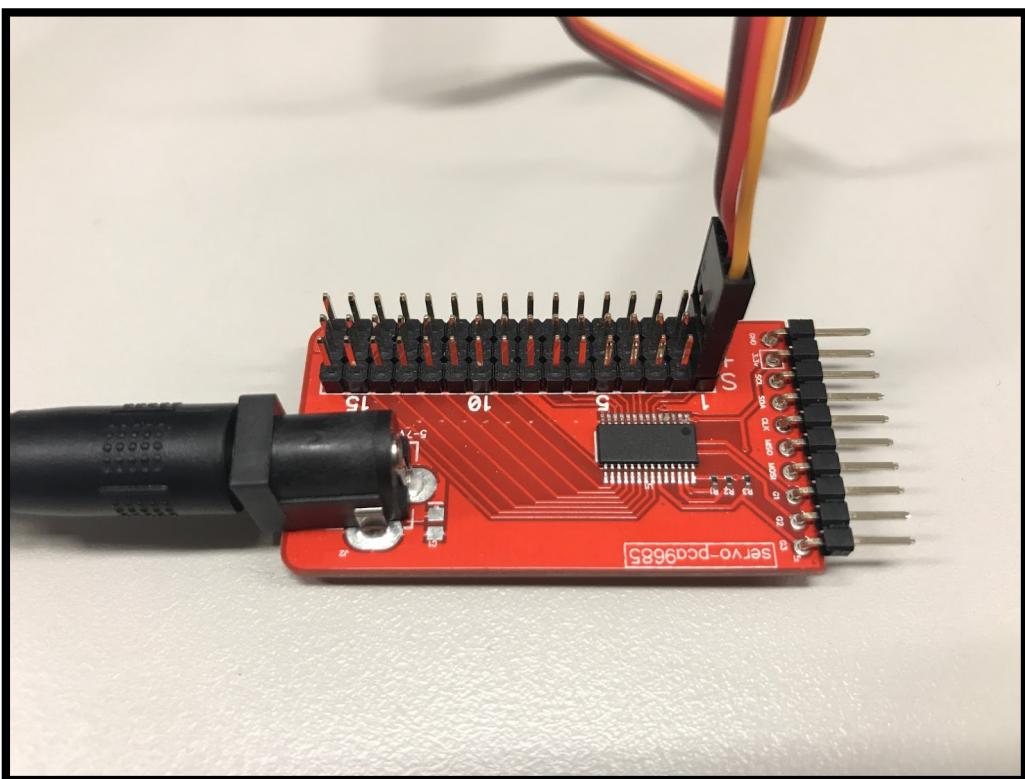
- Tessel 2



- Climate module



- Servo module



- Servo motor



## b. 每個元件如何運作？

- Module 使用

- Tessel 2

使用已寫好的 ‘tessel’ Module。

```
var tessel = require('tessel');
```

- Climate module

使用已寫好的 ‘climate’ module。並且指定位在 port ‘A’。

```
var climatelib = require('climate-si7020');
```

```
var climate = climatelib.use(tessel.port['A']);
```

// 讀取溫度、濕度

```
climate.readTemperature('f', function (err, temp) {
```

```
    climate.readHumidity(function (err, humid) {
```

```
});
```

```
});
```

- Servo module

使用已寫好的 ‘servo’ module。並且指定位在 port ‘B’。

```
var servolib = require('servo-pca9685');
```

```
var servo = servolib.use(tessel.port['B']);
```

```

// 控制的 motor 位在第一排
var servo1 = 1;
// 初始化 Servo 馬達，設定其位置，minPWM, maxPWM
servo.on('ready', function () {
    servo.configure(servo1, 0.05, 0.12);
});
// 控制 Servo 馬達的角度，position 必須在 0~1 之間
function update_servo(servo, servo1, position){
    servo.move(servo1, position);
};

```

- Servo motor

必須接上 5V 電源，並且接上 Servo module 的其中一排（目前設定為第一排）

- 後端 Server

以下對於後端 Server 的三個主要部分做詳細介紹。

- 架設 Server

運用 ‘http’，‘express’，‘socket.io’ 三個 Module。

```

var express      = require('express');
var app         = express();
var server      = require('http').Server(app);
var io = require('socket.io').listen(server);
var os = require('os');
var port = 8000;
server.listen(port, function () {
    console.log(`http://${os.hostname()}.local:${port}`);
});

```

以上實作，把 ‘express’，‘http’ 套件打開，並且設定出 Server，這個 Server 用 socket.io 來與 client 端做溝通，最後，設定聽的 Port 為 8000。console.log 把存許的網址寫出來，其實就是連到 tessel 端的 local: {port}.

這裡必須要讓 tessel 跟電腦連到同一個 wifi 底下才行。

- 設定 HTML 要求的物件

```
app.use(express.static(path.join(__dirname, '/public')));
```

前面已經設定好 app = express()，上面第一行設定 HTML 可以存取靜態資料夾，也就是把 ./public 這個資料夾設定為可以存取。

這裡必須注意要在 .tesselinclude 的地方加入 /public/index.html，在 run 的時候才會一並把這個檔案丟入 tessel。

在使用者連入時，會把 ./public 資料夾底下，index.html 這個 file 丟出去！

### - 處理資料並進行 Output

當每一次前端要求我們作 Update 或是後端主動做 Update 時，會拿到一筆溫度、濕度資料，這個時候就要進行計算，算出風扇的級等，分為 0~5。在本次實驗主要用濕度進行分級，因為溫度比較難以控制，不好實驗。分級方式如下：

Level	0	1	2	3	4	5
Humidity	<60%	60~65 %	65~70 %	70~75%	75~80%	>80%

### - 更新頻率

我們讓後端固定時間更新，並且用 Socket emit 紿給前端。

使用的是：

`setInterval(function () { ... }, 1000(update interval, ms));`

後面的數字代表跑這個 Loop 的時間間隔。

- Web 前端

前端的 framework 是用 react 寫的，和後端是用 socket.io 溝通，和後端建立好連線以後，變由 socket.io 接收由後端送來的即時資料，包括溫度、濕度。只要再接收到資料以後便立即更新 state。

圖表部份，有設置了一個 button，點下去以後，便會開始把資料的部份顯示在圖表上，每一秒為一個單位，即時更新當前和前四秒的濕度與溫度，動態的呈現在圖表中。

- 前端與後端溝通

我們運用 Socket.io 做溝通，主要有開一下幾個 Socket。

'update tem': 這個 Socket 在前端有接收口，後端處理完的資料會把讀到的數值送往前端。

'update hum': 這個 Socket 在前端也有接收口，後端處理完的資料會把讀到的數值送往前端。

'update fan': 在資料分析完之後，我們會知道目前 fan 的級別是多少，由這個 Socket 送往前端。

在自動更新的模式下，後端會每隔一段指定的時間，就 Update 溫度、濕度、風扇級別，並且用 Socket 傳到前端，當前端收到資料後，自然可以更新在頁面上，相

當簡單而方便。

---

### c. 前端顯示 Demo

我們有做兩個版本的前端，一個進行測試，一個是最後版本，先 Demo 測試版本 <Bebug version>，再 Demo <final version>，也就是最後在 Github 上的版本。

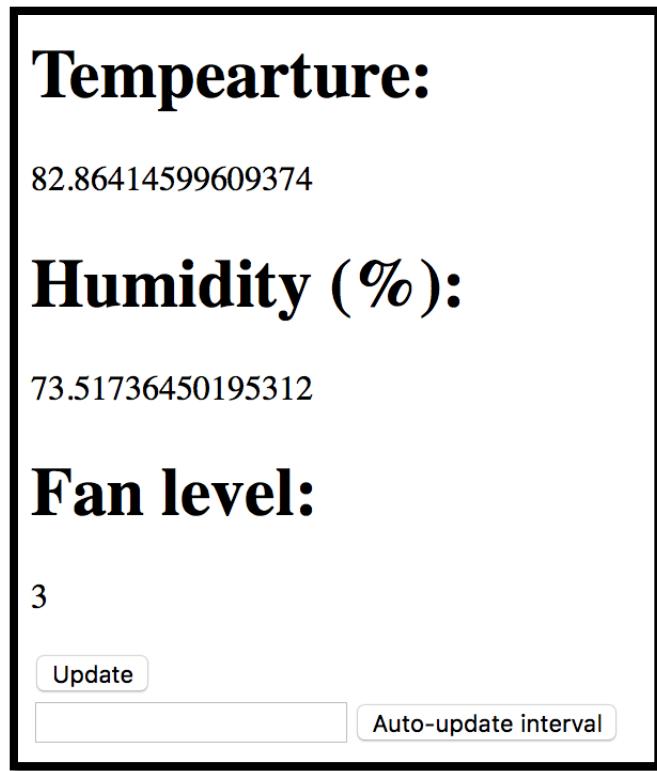
#### <Bebug version>

溫度的單位是華氏、濕度單位是相對濕度(%)，Fan level 顯示段數(0~5)。

在此版本，我們測試兩個按鈕：‘update’(手動更新)，‘auto-update interval’(設定更新頻率)

當前端按下 Update 時，會立刻進行一連串的讀資料、分析、送資料回前端。

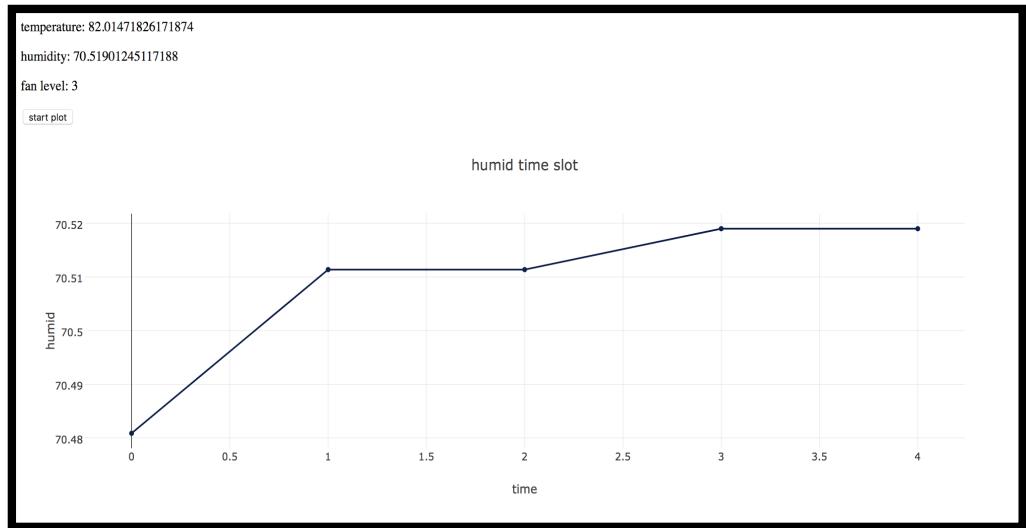
在最底下的欄位，填上數值(單位是 ms)。submit 之後就會透過 Socket 'update interval' 傳到 Server，更新數值。(但後來實作有些問題在)



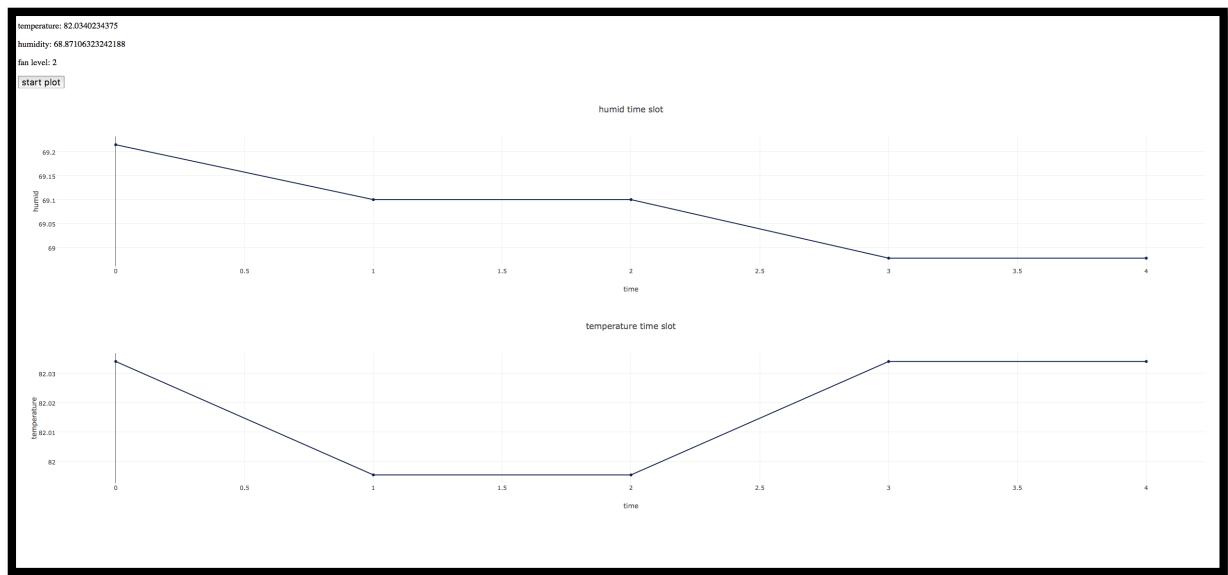
#### <Final version>

左上角顯示溫度、濕度、風扇強度值。

底下有兩個圖表，畫出折線變化圖。



( 整個網頁的圖 )



## 5. 遇到的問題

- 網路連線問題

可能是 Tessel2 本身網路連線不穩，在與手機熱點連線時，一直失敗，後來用實驗室的網路連線可以運作。但有時候也會突然沒有連到，來來回回試了幾次之後才有可以運作。

- 網頁存取問題

在我們進入網址存取時，會遲遲沒辦法 Load 完整個網頁，就當在那邊，可能是 html 裡面有些東西怪怪的，但只要按一下 Botton 就會可以 Work。

另外，我們發現 Chrome 瀏覽器完全開不了東西（在 Mac 上嘗試），也查不出原因為何，後來選用 Safari 使用就沒有什麼問題。

- 設定更新頻率

在設定更新頻率時，我們去改動 update\_interval 的值，但是卻發現沒辦法更動更新頻率，可能是因為在外面改動值，但在一開始讀入 update\_interval 時就已經固定住。

## 6. 結論與心得

- Tessel 是個相當 High level 的板子，用 JavaScript 這個語言，還有很多別人寫好的 Module，在試基礎的板子可以說相當輕鬆，簡單幾行 Code 就完成，讀 Documentation 也不會困難，相較於低階的 Arduino 等可以說方便許多。
- 在使用時，有遇到一些問題，可能因為比較不普及，網路上資源較少。
- 因為使用 JavaScript，若要把資料傳輸出來，可以用 web 方式，透過 wifi 傳出，相當方便。也試過用 usb cam 做影片串流，可以 Real time 顯示在網頁上。

## 7. 參考資料

- Tessel projects: <https://tessel.hackster.io/>
  - [tessel-av: camera](#)
  - [Fish feeder and watering system](#)
  - express + html (from camera video streaming)
  - [socket.io example](#)
  - [w3s css](#)
-