2021 年臺灣國際科學展覽會
優勝作品專輯

作品編號　200022

參展科別　環境工程

作品名稱　懸浮微粒三維偵測與預報系統

得獎獎項　大會獎 三等獎

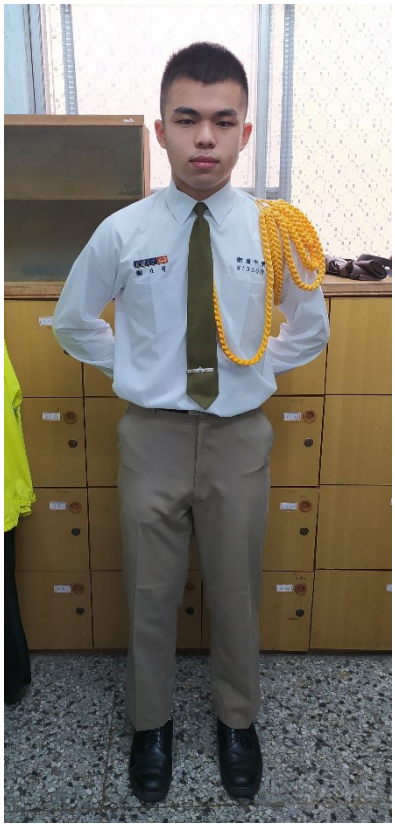
就讀學校　臺中市私立衛道高級中學

指導教師　莊秉潔、陳慶勳

作者姓名　謝在宥

關鍵詞　懸浮微粒、三維偵測、物聯網

作者簡介



我是謝在宥。就讀台中衛道中學高中部二年級。平時興趣: 動手實驗、騎自行車、參與學生組織、讀小說等。自國中起開始接觸電子電路便燃起了我極大的興趣，喜歡動手拆裝各式家電。很高興能將所學興趣發揮在科展上，將懸浮微粒偵測器結合三維立體測量。

立體測量的想法起源於國中，自第一代偵測器改良以來，使我在 Arduino 硬體上有更多想法。上高中接觸 VPython，解決了對於動畫模擬的遐想。將模擬及實際測量並行，能更驗證實驗之想法。

Abstract

In recent years, air quality has become an important issue for people’ s health and an indicator of living environment. In many areas, “fine particulate matter” fill the air, causing allergies and increasing the risk of lung cancer. This study explores the formation of stratification of suspended particles after being affected by gravity, air resistance and buoyancy. In order to understand the stratification phenomenon of suspended particles, the collision of suspended particles of different sizes (pm10, pm2.5 and pm1.0) and the energy attenuation of the particles caused by air resistance were simulated by the software VPython. In addition, we designed and assembled fine particulate matter detectors, and set up them on different vertical height floors of buildings and connected them to the internet. Air pollution in central Taiwan cannot be ignored. Taking the Taichung Coal-Fired Power Plant as an example, we explored its wind direction, geographical environment, and simulated it with a diffusion model. We selected several buildings and monitor the air quality values of different floors of each building, and the monitoring values are automatically uploaded to ThingSpeak, the cloud database of the Internet of Things, and the measurement values can also be monitored locally and downloaded. In the future, we will utilize the machine learning technology and the 3D air quality data we generated to optimize the air quality prediction.

中文作品摘要

近年空氣品質已是居住環境與健康的指標，「細懸浮微粒」充斥在空氣中，造成過敏，增加肺癌的危險。本研究探討懸浮微粒在受到重力、空氣阻力與空氣浮力影響後，形成分層。並利用 VPython 軟體模擬不同大小的懸浮微粒(pm10、pm2.5 與 pm1.0)於空間中碰撞及受到空氣阻力產生能量衰減，藉此了解不同微粒之分層現象。再實作以居住樓層不同的垂直高度，設計組裝架設「懸浮微粒三維偵測器」及物聯網。以台灣中部地區，日益增加的空汙狀況下，模擬以台中火力發電廠為例，探討其風向、地理環境、以擴散模式理論模擬後，選定數棟建築物，監測每棟建築物地面上不同高度的空氣品質數值。最後監測數值自動上傳至物聯網雲端資料庫 ThingSpeak，並可於使用者端監測及取得測量數值；期許再利用機器學習及歷史累積的三維空氣品質資料，將來更優化預測空氣品質數值之成效。

壹、 前言

一、研究動機:

我們每天呼吸的空氣中，「細懸浮微粒」(Fine Particulate Matter, PM2.5)是空氣中粒徑小於 2.5 μM 的極微小粒子，主要成分極可能有：礦物粉塵、硫鹽、硝酸鹽、氨、氯化鈉、黑碳和水，包含懸浮在空氣中之有機和無機物固體和液體的複雜混合物。因粒徑小，可透過人們的鼻腔、經由支氣管、深入肺泡、細支氣壁，干擾肺內的氣體交換等；更甚深入微血管，於人體內經由血液循環，影響人體各器官，造成過敏、氣喘、肺部疾病，長期暴露於「細懸浮微粒」，可引發心血管疾病、呼吸道疾病以及增加肺癌的危險 (Tseng *et al,* 2019)。

本研究設計實作組裝架設「懸浮微粒三維偵測器」，以 NABC 構想為動機：

- N(Need) – 市場上懸浮微粒測站大多採單點平面設立，較少有立體測站及其偵測數據。
- A(Approach) – 增加測站密集度或經過多次測量找到其關係並預估數值。
- B(Benefits) – 對於都市大樓多垂直興建，針對不同高度(樓層)屋外偵測，給予使用者數據，並能有預測數值之功能。
- C(Competition) - 現今多有簡易小型偵測器，使用者可自行架設，並非一定仰賴公家單位偵測系統所提供之資訊。

VPython 是由 Python 程式語言所擴展而來的一種三維空間動態模擬繪圖程式語言 (Scherer *et al*, 2000)。並以 VPython 模擬懸浮微粒(pm10、pm2.5、pm1.0)於空間中碰撞所產生的反彈作用力，最後因能量碰撞衰減而分層，藉此了解不同微粒之分層現象。以得出不同地面高度其空氣品質數值，利用機器學習及歷史資料的累積，期許未來能優化預測空氣品質數值之成效。

首先要突破目前市場上懸浮微粒測站大多採單點平面設立，少有立體垂直式測站及其偵測數據。現今有許多低成本的偵測器, 但精確度不錯 (Badura *et al*, 2018), 故被大量提供在區域性空氣品質的監測 (Feenstra *et al*, 2020)，於是自行設計組裝實作架設「懸浮微粒三維偵測器」監測空氣品質數值。再以 VPython 模擬懸浮微粒(pm10、pm2.5、pm1.0)於空間中碰撞所產生的反彈作用力，然後因能量碰撞衰減而分層，藉此了解不同微粒之分層現象。

最後整合針對台灣中部地區，受地理環境、季節風向、日益增加的空汙狀況下，對於空汙汙染源中火排放為例，以擴散模式理論，高斯分布模式模擬及搜尋出

測站間地理關係，並於地面上不同高度來實作架設「懸浮微粒三維偵測器」監測空氣品質數值。最後裝置及監測數值傳至 ThingSpeak – 是一個由 MathWorks 公司所支持的免費開源軟體物連網資料庫 (Anipindi, 2014)，並可於於後端監測測量數值，再利用機器學習及歷史資料的累積，期許未來能達成優化預測空氣品質數值之成效。

二、研究目的：

綜合上述實作與討論，本研究歸納如下:

(一). 利用 VPython 模擬懸浮微粒(PM10、PM2.5、PM1.0)於空間中碰撞所產生的反彈作用力。

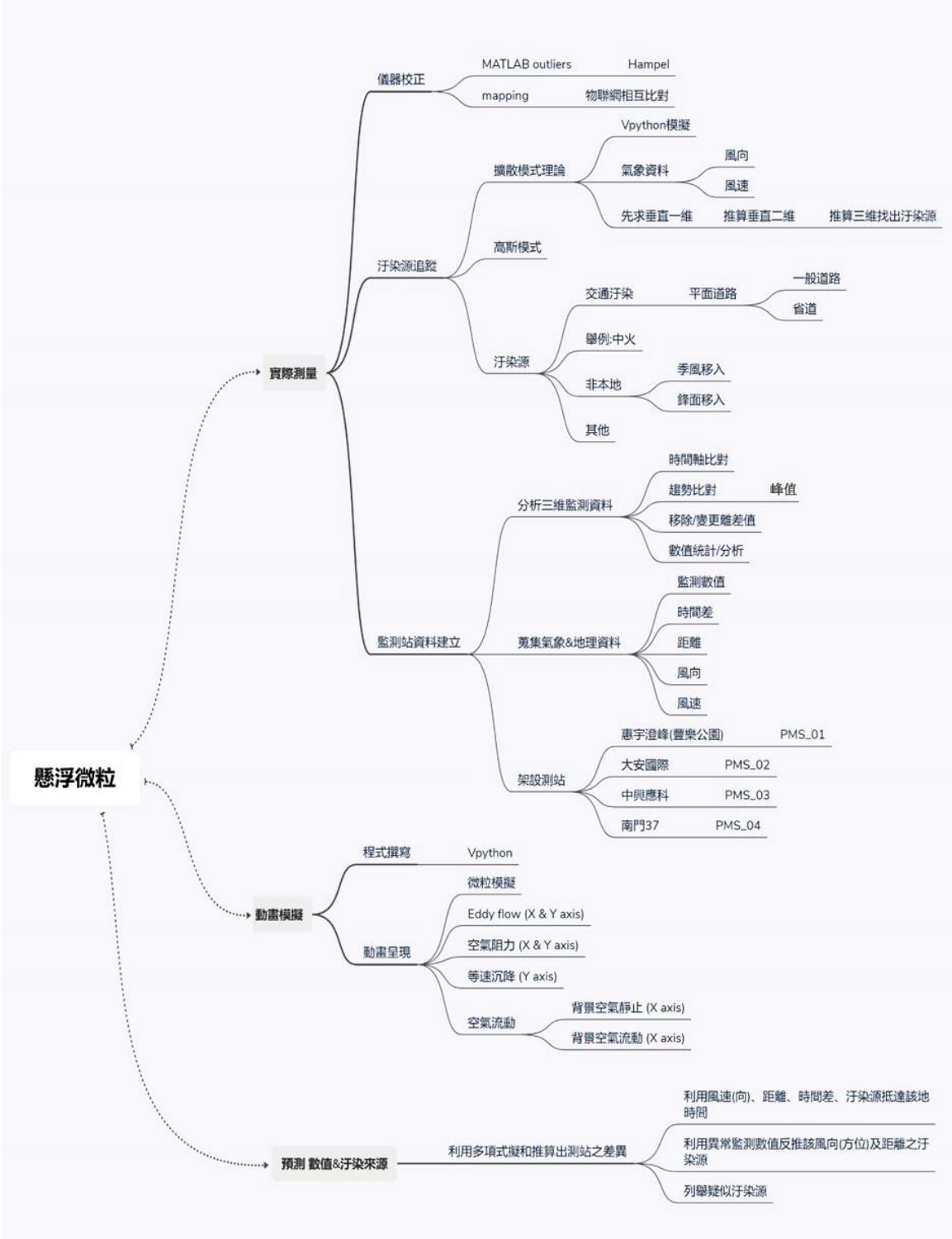
(二). 以擴散模式理論、高斯分布模式模擬及搜尋出測站間地理關係，並以地面上不同高度來實作架設「懸浮微粒三維偵測器」監測立體環境空氣品質數值。

(三). 利用Arduino與ThingSpeak構成遠端監控系統(物聯網)。

(四). 未來利用機器學習及歷史資料的累積，期許能達成預測及優化空氣品質數值之成效。

貳、 研究方法或過程

一、研究實驗架構圖



二、實驗規劃

(一)“懸浮微粒運動”模擬規劃

1.進入程式編寫與改寫

(二)實際測量規劃

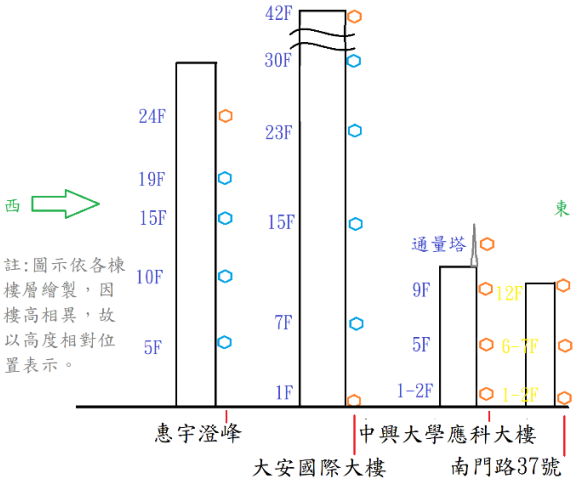
1.實際測量地點規劃

(1).地點一: 豐樂公園/惠宇澄峰大樓

(2).地點二: 大安國際大樓

(3).地點三: 中興大學應科大樓

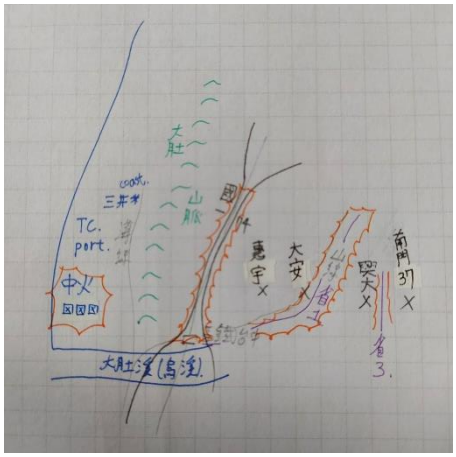
(4).地點四: 南門路 37 號大樓



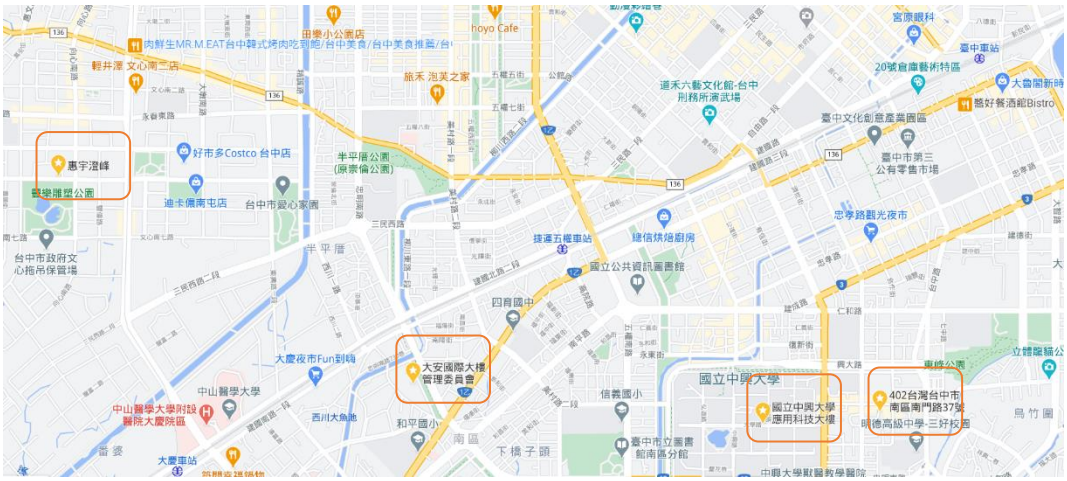
(三)測量目標汙染源

1.台中火力發電廠(燃媒排放)

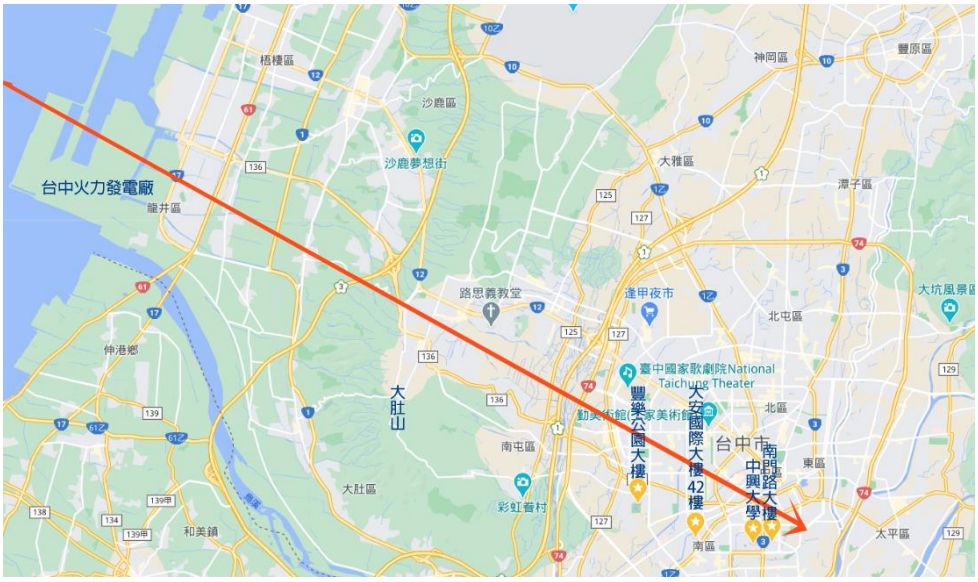
2.交通汙染



由最西邊的中火出發，因大肚山脈海拔約 100~200m，中火煙囪 250m，空汙可以輕易越過，加上大肚溪 河口及河床可能有海風沿河口向東移動，空汙進入南屯/南區後，經過第一監測截面(豐樂公園/惠宇澄峰大樓)，再往西經過鐵路及省一(復興路)，經 過第二監測截面(大安國際大樓/樓高 42 樓)，再往西經過健康公園與復興園道、較少汙染產生，相對有一定的沉降，再來來到第三監測截面(中興大學應科大樓 13 樓高樓)，最後再更往西下風處，經過省三/國光路，來到第四監測截面(南區 11 樓高大樓)。



風向：西風



三、“懸浮微粒運動”模擬

(一)、程式編寫

1. 程式碼編輯器
   - Visual Studio Code
2. 程式執行環境
   - Python 3.7.9 (64bit)
3. 程式碼(見附錄)

(二)、相關公式計算

1.球終端速度(無空氣浮力情況下)

(1). $$vt = \sqrt{\left(\frac{2mg}{\rho} \times AC_d\right)}$$

a.上列方程式中

- V_t 為 球體終端速度(y 軸向下)
- m 為 球體質量
- g 為 重力加速度
- ρ 為 落下環境流體密度(此處為空氣)
- A 為 球體投影面積($\pi d^2 / 4$，d 為球體直徑)
- C_d 為 阻力係數

b.此方程式適用於球體，符合程式模擬中假想。

2.球終端速度(受空氣浮力影響下)

```
#########################################################################################################
#球體終端速度
yAxisTerminalVelocityA=-(((2*massA*g)/(airDensity*shadowAreaA*dragCoefficient))**0.5)#A 球終端速度
yAxisTerminalVelocityB=-(((2*massB*g)/(airDensity*shadowAreaB*dragCoefficient))**0.5)#B 球終端速度
yAxisTerminalVelocityC=-(((2*massC*g)/(airDensity*shadowAreaC*dragCoefficient))**0.5)#C 球終端速度
```

(1). $$vt = \sqrt{\left(\frac{4gd}{3C_d}\right)\left(\rho s - \frac{\rho}{\rho}\right)}$$

a.上列方程式中

- V_t 為 球體終端速度(y 軸向下)
- g 為 重力加速度
- d 為球體直徑
- ρ 為 落下環境流體密度(此處為空氣)
- ρ_s 為 球體密度(此處值為 1.06，懸浮微粒密度)
- A 為 球體投影面積($\pi d^2 / 4$，d 為球體直徑)
- C_d 為 阻力係數

2.球體任意時間速度

```
ballA[i].v=ballA[i].v+ballA[i].a*dt # v=v0+a*t
```

3.球體認異時間軌跡

```
ballA[i].pos=ballA[i].pos+ballA[i].v*dt # S=s0+v*t
```

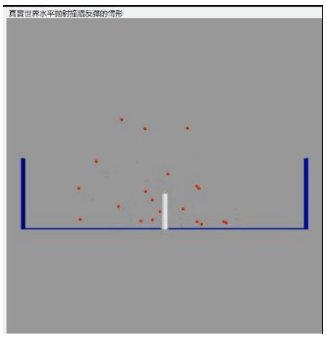
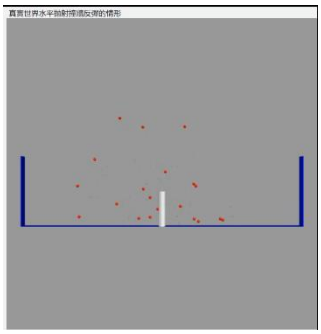
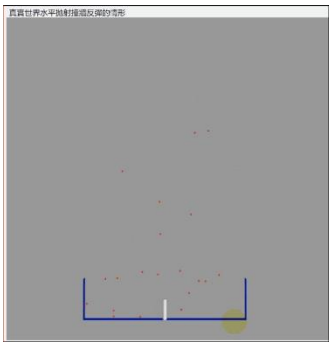
(三)、應變變因

1.VPython 模擬：

利用 VPython 模擬懸浮微粒(pm10、pm2.5、pm1.0)於空間中碰撞所產生的 反彈作用力，最後因能量碰撞衰減而分層，藉此了解不同微粒之分層現象。

2.程式執行動畫:

https://youtu.be/YuZ-IctOoDI

執行情況(圖一)	執行情況(圖二)	執行情況(圖三)
		

(四)、困境

1.執行程式後，受畫面影響，螢幕錄製效率低。

2.因球體位置比對(利用雙 for loop 及陣列比對碰撞)，程式運算量大播放慢。

3.未來加入 CUDA nvidia 顯卡整合技術，利用 GPU 加速程式運算(@jit, Just in Time)，並匯入 numba 模組，以確實增加程式運算效率。

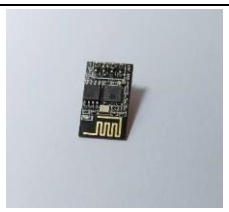
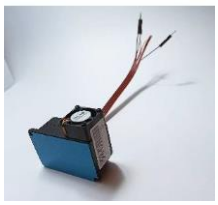
4.於增加運算效率後，可增加程式中的碰撞球體個數，增加碰撞機會，以更加符合實際環境下懸浮微粒碰撞模擬。

5.因 VPython 為 Python 之延伸開發資料庫，故無法直接運用 jit 運算於立體圖形計算，換而言之，目前尚無找尋到更佳的運算方式。

四、實際偵測

(一)、研究器材及硬體介紹

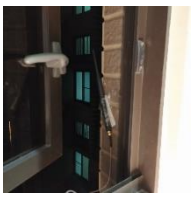
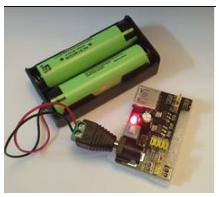
1.偵測器硬體介紹

			
麵包板電供(5v 3v3)	Arduino NANO	ESP8266-01S	麵包板
			
攀藤 PMS3003	GP2Y1014AU	DHT-11	電供(12v 1A/2A)

2.偵測器外殼介紹

外殼比較	舊式外殼(紙箱)	新式外殼(硬碟盒)
圖示		
缺點	不防水、防護性低、體積較大	散熱性低
優點	散熱性佳、可外接備用電源(電池)	耐摔、防水、固定性高、體積小巧

3.研究器材介紹

				
EX1200T	RT-N16	天線延長線	18650 電池*2/組	EX200
WIFI/AP 訊號延伸器	WIFI router	延伸梯間 收訊用	4.2V*2 備用電力	WIFI/(AP) 訊號延伸器

(二)、偵測器(測點)介紹

1.偵測器硬體介紹

(1).麵包板電供(5v 3v3): 偵測器(測點)主電源供應，輸出 5v 及 3v3。

(2).Arduino NANO: 偵測器(測點)主控制器，接受 PMS3003 及 DHT-11 回傳數值，並由 ESP8266-01s 發送出。

(3).ESP8266-01S: WIFI 傳輸模組。

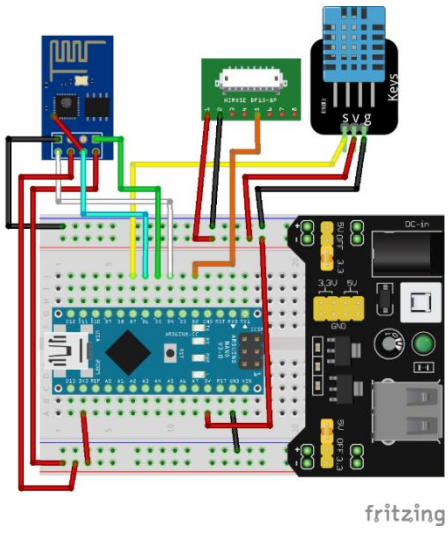
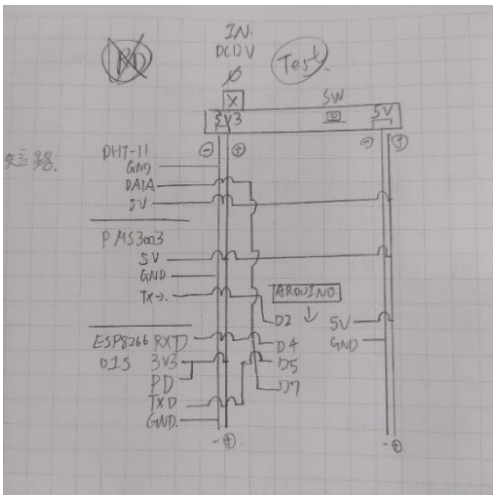
(4).麵包板: 配線用。

(5).PMS3003: 懸浮微粒偵測器，測量種類 PM10、PM2.5、PM1.0。類比訊號輸出。

(6).GP2Y1014AU: 懸浮微粒偵測器(用於第一代機型)，偵測懸浮微粒(總值)。數位訊號輸出(易受電阻值影響而不準確)。

(7).DHT-11: 溫溼度偵測器。類比訊號輸出。

(8).電供(12v 1A/2A): 市電 AC110VAC 轉 DC12V，1A 及 2A 供應電源。

	
自繪電路圖(ESP8266 in Sleep Mode connection)	手繪電路圖(第四代專用)

2.第一代偵測器

(1).採用 GP2Y1014AU 作為主偵測器。

(2).硬體設備:麵包板電供(5v 3v3)、Arduino NANO、ESP8266-01(New 1MB)、GP2Y1014AU(含 150Ω電阻、220uF 電容)、LED 指示燈(含 330Ω電阻)

(3).缺點(淘汰原因)

a.無法分別測量 PM10、PM2.5、PM1.0 數值。

b.數位訊號輸出，易受電阻值影響而數值不準確。

3.第二代偵測器改良

(1).採用 PMS3003 作為主偵測器

(2).硬體設備:麵包板電供(5v 3v3)、Arduino NANO、ESP8266-01S、PMS3003

(3).改良優點:表面防水、耐碰撞、空間大

(4).缺點

a.待偵測空氣樣本不流通

b.系統空氣不流通

c.體積大

d.配線困難(線路不易進入、WIFI 訊號屏蔽)

4.第三代偵測器改良

(1).採用 PMS3003 作為主偵測器

(2).硬體設備:麵包板電供(5v 3v3)、Arduino NANO、ESP8266-01S、PMS3003、18650 電池*2

(3).改良優點:防水、耐碰撞、體積小、電路集約、系統集約

(4).缺點

a.系統散熱不佳(過熱)

b.電池供電時間性短(耗電較快)

5.第四代偵測器改良

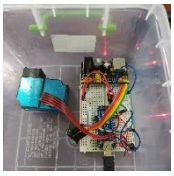
(1).採用 PMS3003 作為主偵測器

(2).硬體設備:麵包板電供(5v 3v3)、Arduino NANO、ESP8266-01S、PMS3003、電供(12v 1A/2A)

(3).改良優點:防水(防滲水、擋板)、散熱改良(對流)、耐碰撞(固定墊片)、體積小、市電接電供(12v 1A/2A)供電、WIFI 不受外殼屏蔽。

(4).缺點

a.偵測溫度仍與環境溫度有所差異(部分系統元件發熱)

第一代偵測器	第二代偵測器	第三代偵測器	第四代偵測器
			

6.第四代偵測器 機盒配置介紹	
(1).墊片*6 (2).出風散熱口 (3).固定夾*3 (4).電源線出口 (5).溫溼度空氣取樣口 (6).防雨擋板*3 (7).空氣樣本吸入口(懸浮微粒用) (8).被動抽風風扇(含於PMS3003)、兼散熱風扇	

(三)、偵測器(測點)**立體測站**架設位置

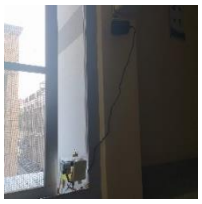
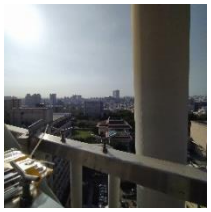
1. 豐樂公園/惠宇澄峰大樓 (24˚07'51.6"N 120˚38'38.4"E)

位置	開放陽台 1F (對面大樓)	開放陽台 8F (對面大樓)	陽台 15F	開放陽台 19F	開放陽台 24F
編號	pms01_D	pms01_C	Test_C_sensor	pms01_B	pms01_A
監測設備			與右圖 pms01_A 同		 1-24F-1
監測地點景象	民宅 (接洽中) 與右圖 1-24F-2 類似		民宅 (目前無法取得建置)	民宅 (目前無法取得建置) 與右圖 1-24F-2 類似	 1-24F-2

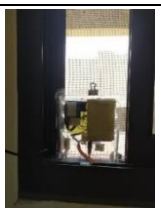
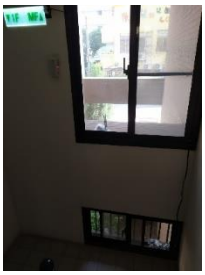
2. 大安國際大樓 (24˚07'23.5"N 120˚39'36.3"E)

位置	1F 後門	7 樓	15 樓	23 樓	30 樓	開放陽台 42F
編號	pms02_A	pms02_B	Test_B_sensor	Test_A_sensor	pms02_C	pms02_D
監測設備			與右圖 pms02_D 同	與右圖 pms02_D 同		
監測地點景象		生技公司 (接洽中/無適當對外陽台可裝機)	規劃中(目前無法取得建置)	會計師事務所 (接洽中)	外語公司 (目前無法取得建置)	

3. 中興大學應科大樓 (24˚07'17.5"N 120˚40'35.2"E)

位置	梯間 1-2F	梯間 5F	9F 資料接收室	約 12F 通量塔
編號	pms03_A	pms03_B	pms03_C	pms03_D
監測設備			 註:加固後照片	
監測地點景象			 註:西方可見大安國際大樓	

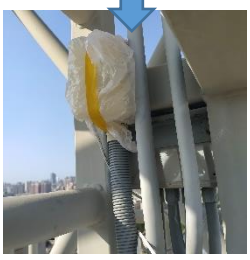
4. 南門路 37 號大樓 (24˚07'19.4"N 120˚40'55.1"E)

位置	梯間 1-2F	梯間 6-7F	梯間 11F
編號	pms04_B	pms04_C	pms04_D
監測設備			
監測地點景象		 註:樓層含挑高樓，增加一層樓高作為比較	

(四)、偵測器(測點)架設困境及問題

1.固定及防水處理

(1).應科大樓通量塔(編號:pms03_D)

	WIFI AP 防水盒 (含電源配線盒)	LAN (9F 資料接收室至第三塔面) 防水塑膠袋	pms03_D 強化固定及防水處理 (風大雨強，且地點不易維修)
防水作業	 	 	 電源延長線防水固定 

(2).通量塔架設成果

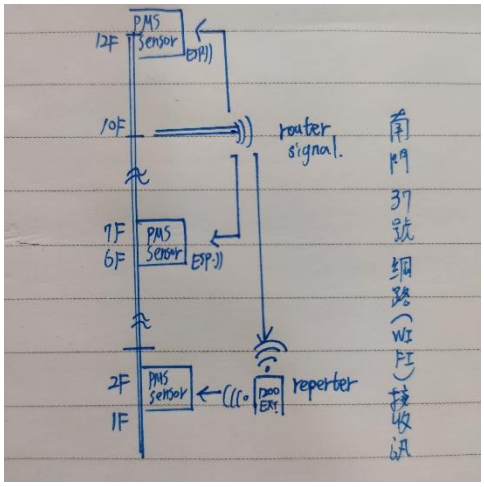
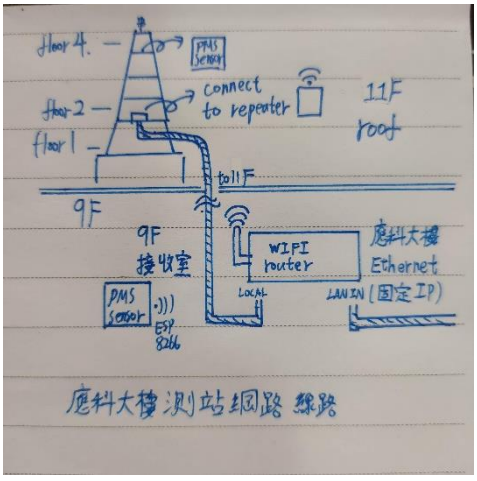
- 於 2020/11/7 降雨時確認數據監測無異常。
- 2021/01/21 登塔檢驗正常

1.架設地點詢問:需安裝偵測器的各大樓當中,洽談安裝的樓層尋得同意合作不易，多數因不認識而拒絕，是目前無法全數架設的主要困境。

2.pms_03C (9F 資料接收室)於 2021/01/05 下午斷訊，原因不明。後於 2021/01/21 檢修時了解因風大而掉落出窗外，麵包板電供脫落；經測試後設備正常，加固於窗框，持續監測。

3.pms_01A (開放陽台 24F)因冬季太陽直射(約 10:30-13:30)導致機組過熱(溫度顯示 60˚C，已超越測量範圍)，於 2021/01/05 斷訊。2021/01/12 檢修，更換 ESP8266-01S 模組，後黏貼鋁箔紙於盒外，監測溫度狀況明顯改善。

3.WIFI 訊號強度改善

(1).南門路 37 號-梯間 1-2F	(2).中興大學應科大樓-9F 資料接收室&12F 通量塔
	
梯間 1-2F 無法直接接收/傳輸 10F WIFI router 訊號(ESP8266 角度及大樓結構因素)，於 2F 窗台架設訊號延伸器。	於 9F 資料接收室裝設 wifi router，同時提供 9F 測站(pms03_C)WIFI 訊號，並同時連接內線網路至 11F 頂樓通量塔第三塔面，於第三塔面再裝設 WIFI AP(同訊號延伸器)。

4.電源供應

(1).電池供電

a.9V 充電電池供電

b.4.2V(18650 型)充電電池 2 顆串聯供電

(2).直流電源供電

a.USB 5V 連接 Arduino 供電

b.直流變壓器供電

(3).電池供電缺點

a.因 ESP8266 用電量與測試場地有異

- 實際場地 WIFI 傳輸距離較測試場地遠，ESP8266 所需傳輸功率(使用功率)亦提升。
- 裝設地點不易時常(每日)更換電池。

(五)、Arduino 程式介紹

1. 程式碼(見附錄)

2. PMS3003 DHT-11(ESP connected)

3.為維護實驗數據上傳正確，連結中程式碼已移除 WIFI 及 API Key 資料。

(六)、ThingSpeak 物聯網介紹

1.ThingSpeak Channel 後臺設定

2.Arduino 程式寫入 Write API key、ThingSpeak 伺服器 IP(184.106.153.149)

3.後臺觀看監測資料:

1A	1B	1C	1D	2A	2B	2C	2D
pms01_A	pms01_B	pms01_C	pms01_D	pms02_A	pms02_B	pms02_C	pms02_D
1163811	1163813	1163814	1163815	1163835	1163836	1163837	1163838
3A	3B	3C	3D	4A	4B	4C	4D
pms03_A	pms03_B	pms03_C	pms03_D	pms04_A	pms04_B	pms04_C	pms04_D
1164253	1164254	1164255	1168256	1168258	1164259	1164260	1164261

(七)、後端監測數據介紹

1.於各圖表(field)中，其 x 軸單位皆為時間(於一般情形下，每 300 秒偵測 1 筆資料)；y 軸為此監測設備各偵測項目，其單位如下。

2.各圖表(field)偵測名稱項目:

field	field1	field2	field3	field4	field5	field6	field7	field8
偵測項目	PM1.0 標準顆粒物	PM2.5 標準顆粒物	PM10 標準顆粒物	PM1.0 大氣環境下	PM2.5 大氣環境下	PM10 大氣環境下	濕度 range 20-90 %	溫度 range 0-50°c
項目名稱 (MATLAB 中分析用)	PM01.0 _SP	PM02.5 _SP	PM10.0 _SP	PM01.0 _AE	PM02.5 _AE	PM10.0 _AE	L_Humi	L_Temp
單位	$\mu g/m^3$	$\mu g/m^3$	$\mu g/m^3$	$\mu g/m^3$	$\mu g/m^3$	$\mu g/m^3$	%	°c

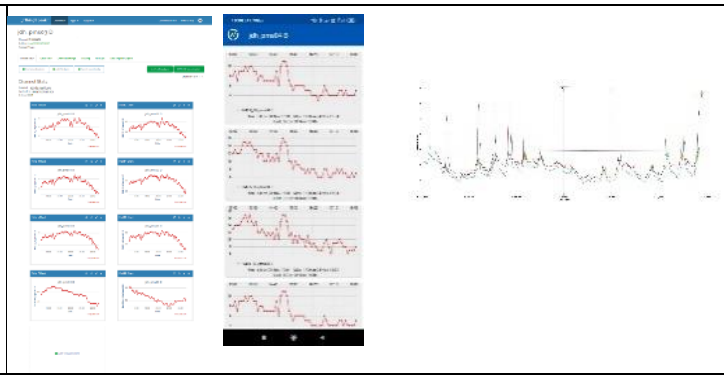
3. 大氣環境下/標準顆粒物選用

(1).監測數值中，pm1.0/pm2.5/pm10 於 PMS3003 中可偵測到兩種數值，標準顆粒物(CF=1)/ 大氣環境下(atmospher)

- CF=1 標準顆粒物-根據美國 TSI 公司的儀器校準(上方 SP)。
- 大氣環境下-根據中國氣象局的數據校準(上方 AE)。
- 標準顆粒物質量濃度值是指工業金屬顆粒物作為等效顆粒進行密度換算得到的質量濃度值，適用於工業生產環境。大氣環境顆粒物質量濃度值以空氣中主要污染物為等效顆粒進行密度換算，適用於普通室內外大氣環境。

(2).本實驗使用大氣環境下偵測為依據。

4.實際監看畫面

電腦網頁監看畫面 行動裝置端應用程式簡易監看畫面 MATLAB plot 繪圖	

(八)、監測資料傳輸路徑(上傳)

1.Arduino NANO 由 ESP8266 連接至指定 WIFI(2.4GHz) 。

2.PMS3003 及 DHT-11 每 300 秒偵測環境懸浮微粒及溫濕度。

3.Arduino NANO 接收 PMS3003 及 DHT-11 Tx 資料後，將資料量化後由 ESP8266 上傳至 ThingSpeak 指定 channel(透過 Arduino code 中 channel Write API key 指定)

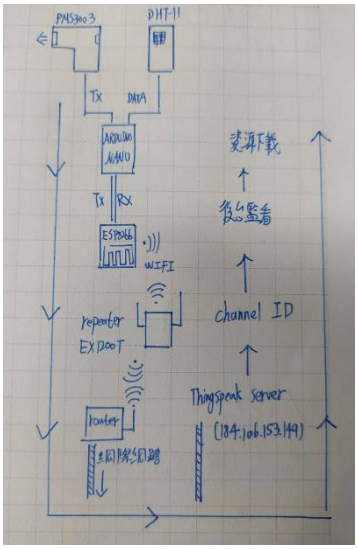
4.透過 LAN 傳輸資料至 WIFI router(部分機組需先經強波器)

5.透過 router 將資料傳輸至網際網路。

6.透過網際網路將資料傳輸至 ThingSpeak 伺服器 (184.106.153.149)

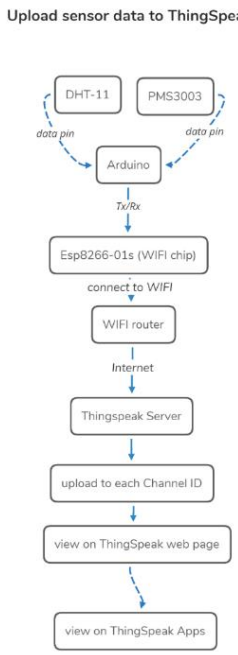
7.由 ThingSpeak 網頁(或行動裝置應用程式中)監看測站即時監測數值。

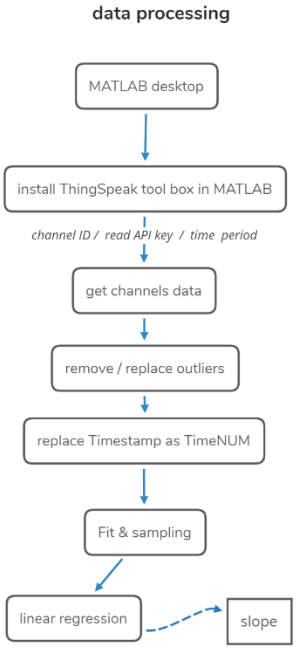
監測資料傳輸路徑示意圖



五、監測資料處理

1.資料處理流程圖





2.裝設前校正

(1).於偵測器裝設前，統一於室內同一地點進行同步偵測(共 16 臺+2 備用)，因出廠值有誤差, 需校正，取 2020/ 10/23~11/1 偵測數值。

2-(1)同一地點測量

3. 偵測資料正規化(normalization)

- 於 MATLAB 中進行數據校正。
- 取用數值為 2020/ 10/23~11/1 偵測數值之間同步校正之值。

(1).離群值(outliers)取代，將值與左右各 K(不含)筆資料標準差相減，若大於標準差 N 倍者將判定為離群值，並以此 2K 筆資料之中位數取代。(依各偵測器最有效離群值移除取左右 K 筆及標準差為 N 倍，以 pms03D 為例，K = 50，N =6)

(2).撰寫程式於各設備時，皆為每 300 秒偵測(讀取)一次數值，但受限各設備執行實際誤差及網速差異，故實際取樣間隔約為 315±15 秒 。

(3). 由於 MATLAB 中 X 軸為時間 Timestamps 時(非數字)，無法擬合曲線，故需要先利用 datenum function 將 Timestamps 變為數字，轉換後為 TimeNUM(數字)。

(4).fit function 的 “smoothin]=gspline”方法找出每個偵測器的擬合曲線，每個偵測器的擬合曲線同時間取樣(由 10/23~11/1, 每 300 秒取樣)。

(5). 將兩測站 (如 A 及 B 測站) 同時間的取值的數據點做圖，如果 A、B 測站偵測值無差異，理想測值應與 XY 座標上的 Y=X 這條直線吻合 (如 X 表 A 測站值，Y 表 B 測站值)，但如果有差異且為線性，可利用線性回歸法求取回歸方程截距及斜率，提供數據校正。利用 Matlab 的 fitlm functrion 以”RobustOpts”方法較不會受到偏差值影響，即可求取同一時間兩偵測器測值之比值, 提供數據校正。

參、 研究結果與討論

一、「懸浮微粒運動」模擬結果

(一).就目前執行結果分析，於三種球體(紅、綠、黑 分別對應 pm1.0、pm2.5、pm10 比例)間發生碰撞，因黑球質量較紅球小，在碰撞時受作用力及反作 用力影響下，黑球之加速度大於紅球；此結果俗稱彈的較遠，也是造成為何黑球於動畫中多移動速率較快且有較多分散於高空處。

二、「懸浮微粒運動」模擬結果討論

(一).「懸浮微粒運動」模擬中呈現較小顆粒子因反彈加速度較大而較易停留於高空中。

(二).受限於程式運算限制，懸浮微粒實際於單位空氣中所含實際個數(總重)與程式模擬中異，未來模擬情形可更貼近於實際環境情況。

三、實際偵測選定結果

(一).地點一: 豐樂公園/惠宇澄峰大樓

(二).地點二: 大安國際大樓

(三).地點三: 中興大學應科大樓

(四).地點四: 南門路 37 號大樓

四、實際偵測選定結果討論

(一).偵測目標物確認

(二).背景氣象影響因素--蒐集氣象資料(懸浮微粒、風向、風速)

(三).地理地形影響因素: 目前實驗已架設 4 棟大樓地點，於每個測點(大樓)中各選擇 4~5 層架設，形成一個立體的串聯網絡。由最西邊的中火出發，因大肚山脈海拔約 100~200m，中火煙囪 250m，空汙可以輕易越過，加上大肚溪　河口及河床可能有海風沿河口向東移動，空汙進入南屯/南區後，經過第一監測截面(豐樂公園/惠宇澄峰大樓)，再往西經過鐵路及省一(復興路)，經過第二監測截面(大安國際大樓/樓高 42 樓)，再往西經過健康公園與復興園道、較少汙染產生，相對有一定的沉降，再來來到第三監測截面(中興大學應科大樓 13 樓高大樓)，最後再更往西下風處，經過省三/國光路，來到第四監測站截面(南區 11 樓高大樓)。

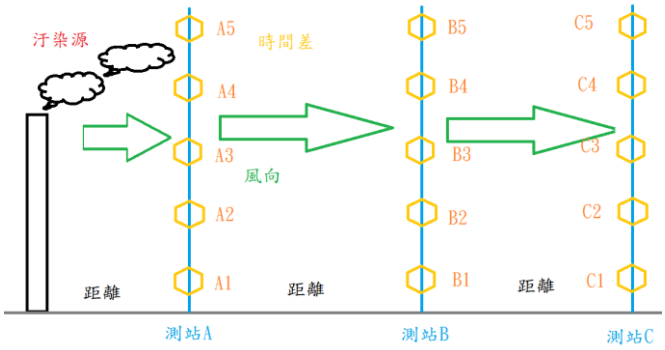
(四).測站地點選定---搜尋測站間地理資料(距離、方位)標定時間差

(五).討論:

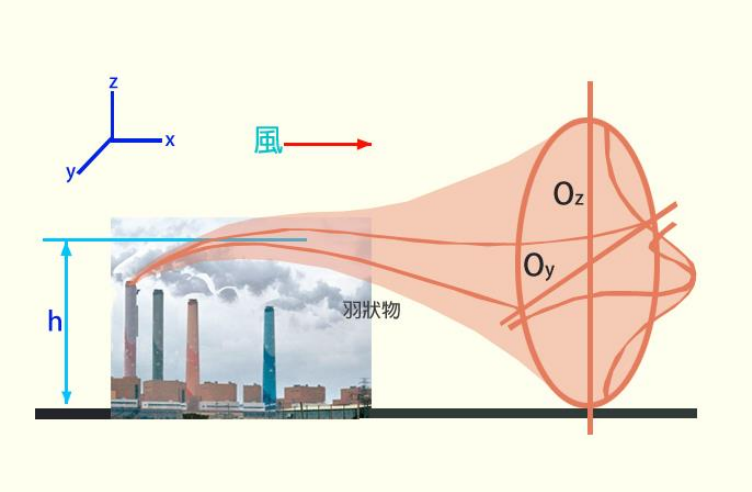
1.可利用測站間距離、風速，推算(預估)該批汙染物(峰值)抵達時間差。

2.三維測站，受風力強度不同，推算使汙染物停留時間長短有異而峰值有所明顯改變。

- 測站
- 距離
- 風向
- 風速
- 時間差
- 汙染源



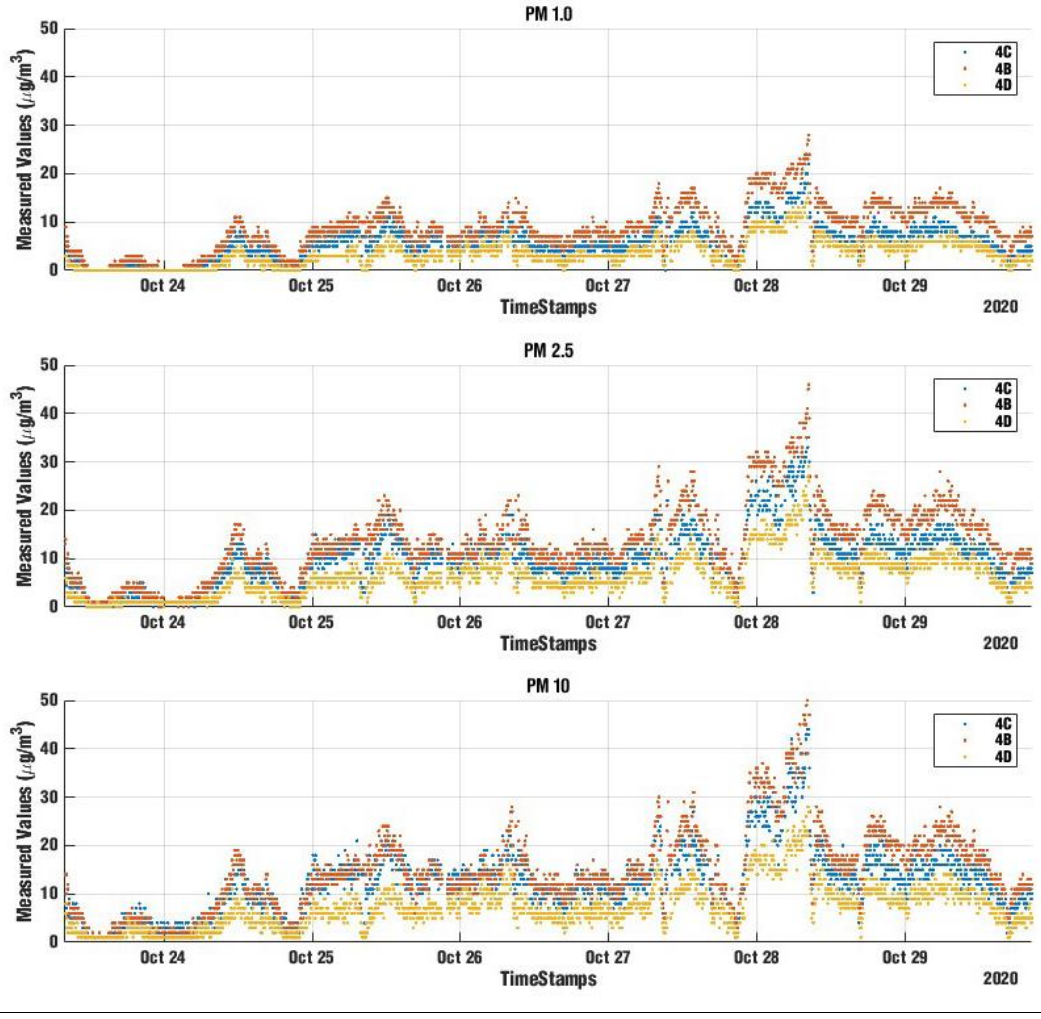
高斯分布



五、偵測值分析

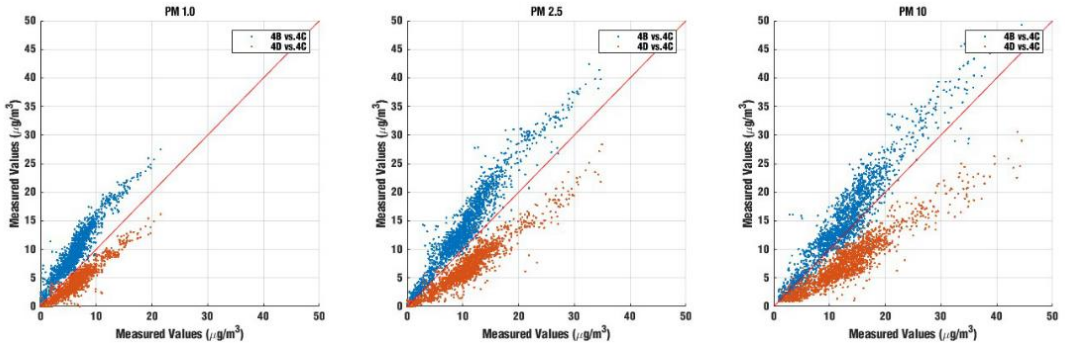
(一). 校正期間

1.在正式將測站分散測量前，先將所有機器在同一環境下運作。而後注意到機器所測得值有差異，以 4B、4C 及 4D 機器 (之後將放置於「南門路 37 號大樓」的不同樓層) 為例，4C 的數據有大於 4D 及小於 4B 的趨勢 。



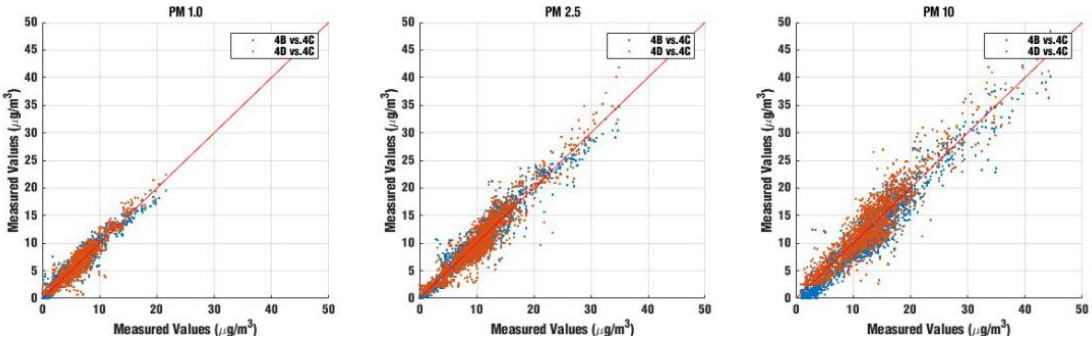
設備校正期間 4B、4C 及 4D 測站隨時間變化的懸浮微粒值比較。

2.將不同測站取同時間的偵測值作圖，可看到 4B vs 4C 與 4D vs 4C 的點皆不在斜率為 1 的直線上，亦顯示了不同測站間的測量值有誤差，且近似於線性。



不同測站取同時間的偵測值作圖，圖中紅線斜率為 1。

於是利用線性回歸方法求得截距與斜率，對於 4B 及 4D 的值進行校正，可見三測站校正後的值對於時間作圖已趨一致。



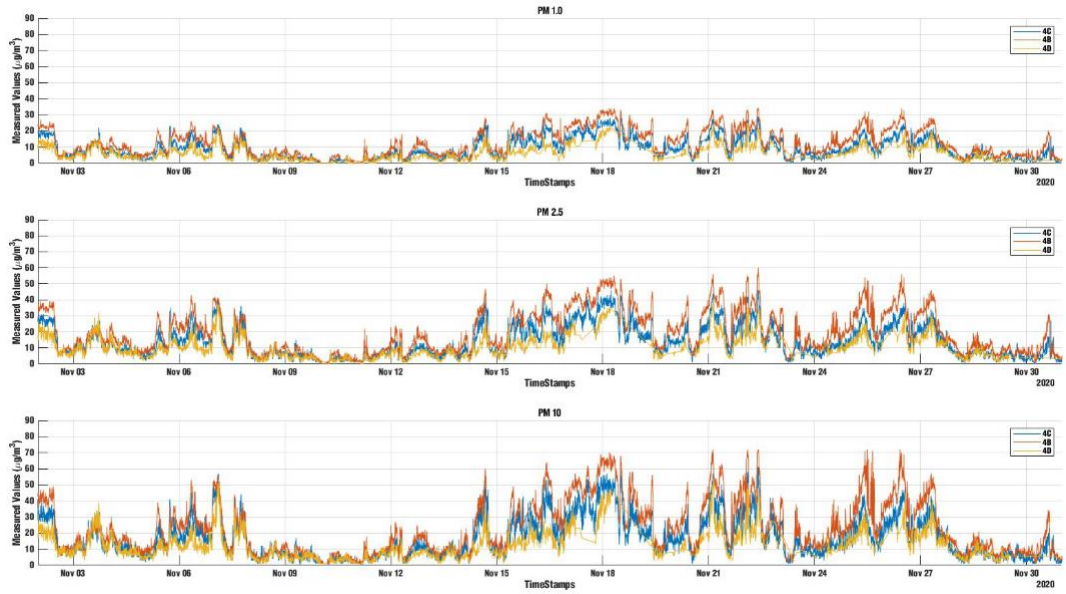
不同測站取同時間的校正偵測值作圖，圖中紅線斜率為 1。

且 4B vs 4C 與 4D vs 4C 的點皆與斜率 為 1 的直線吻合。

(二). 觀察期間

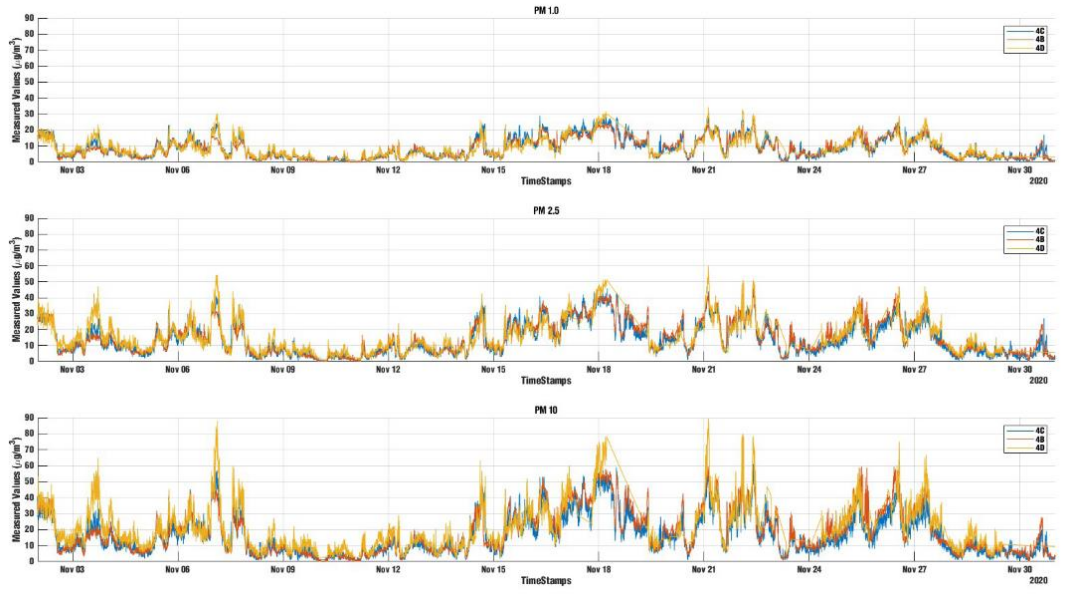
1.以 4B、4C 及 4D 的測站為例 (各放置在「南門路 37 號大樓」的 1、6 與 11 樓)，觀察數據時間取為 11 月 2 日至 12 月 1 日。

無校正數據圖為:



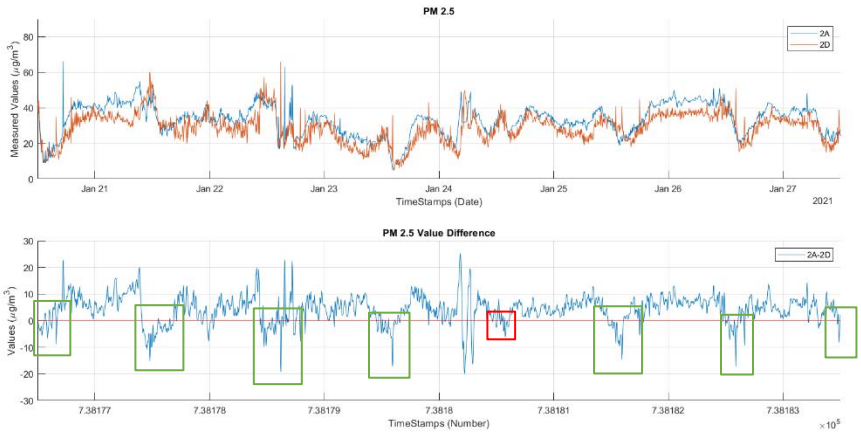
觀測期間 4B、4C 及 4D 測站隨時間變化的懸浮微粒值比較。

校正後數據為:



觀測期間 4B、4C 及 4D 測站隨時間變化的懸浮微粒校正值比較。

2.另外以樓距差距最大地點的測站為例（pms03 大安國際大樓 1 樓 (2A)與 42 樓 (2D)）。由於數據校正後有可能還是會失真，我們直接取兩測站未校正之前的值觀察其差值，期間為一周 2021/01/20 12:00 至 01/27 12:00，結果如下:



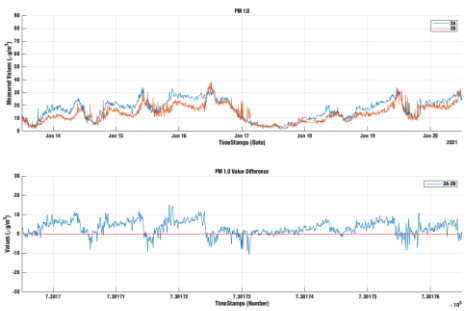
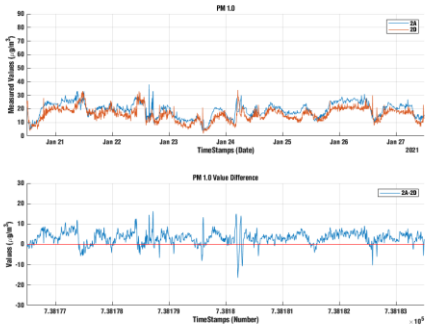
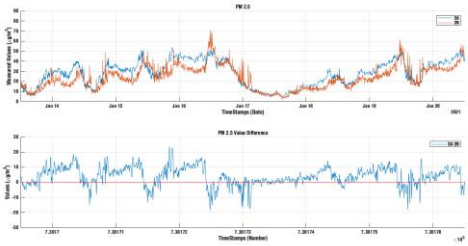
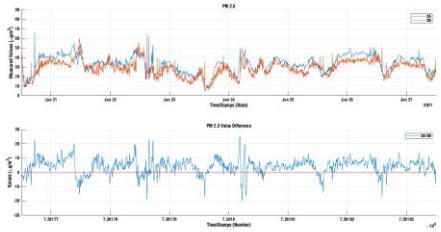
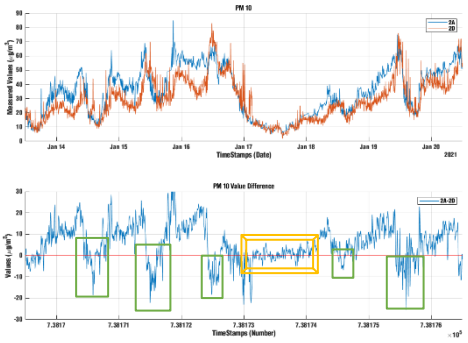
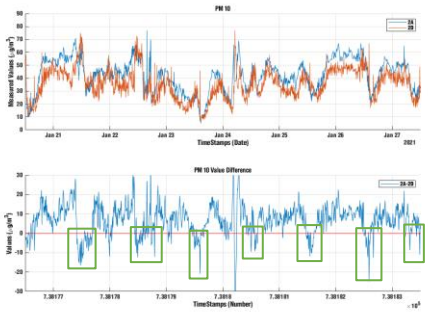
觀測期間 2A 及 2D 測站隨時間變化的 PM2.5 比較。
下方圖為兩測站測值之差，紅線為差值為 0 之線。

可看到兩測站的測值互有消長；我們注意到在這段期間一個規律性: 每天午間前後時分(上圖綠框部分) ， 2A (1 樓)與 2D (42 樓) 的差值會由正轉負，這有可能是因為這段期間高樓層空氣懸浮微粒變多，或是接近地面樓層因過了上下班時間而懸浮微粒變少，有趣的是 24 日(星期日)這差值減小 (上圖中紅框部分)，這有可能是當天較沒有上下班時的交通工具汙染，高低樓層差異變小。總體來說，低樓層似乎有較高的懸浮微粒濃度，也驗證了不同樓層的 PM2.5 是有差異的，這變因很多，不同時段區域的交通污染與風場方向皆有相關。

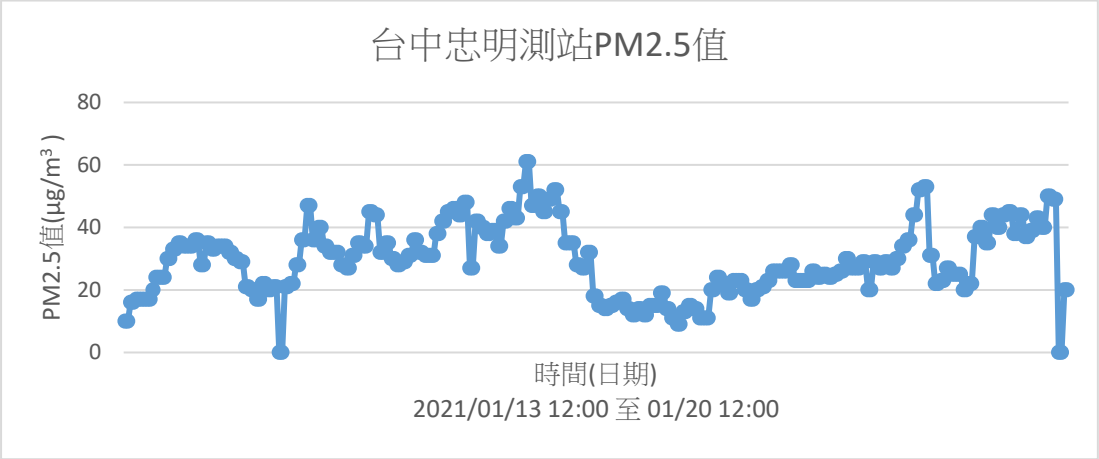
3.於樓距差距最大地點(pms03 大安國際大樓 1 樓 _2A 與 42 樓_2D)的測站一月份 PM1.0、PM2.5 和 PM10 資料與風場資料同步分析。

觀測期間 :

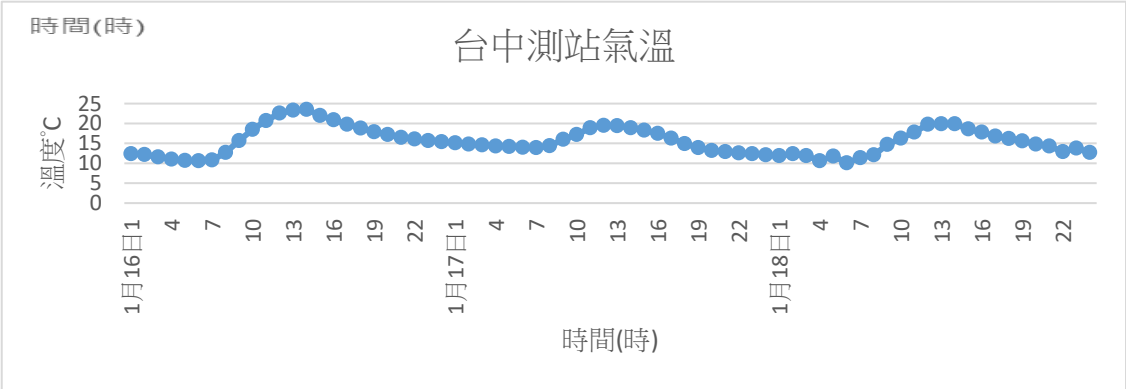
(左) 2021/01/13 12:00 至 01/20 12:00　　(右) 2021/01/20 12:00 至 2021/01/27 12:00

	
2021/01/13 12:00 至 01/20 12:00 **PM1.0** (2A & 2D 比較)	2021/01/20 12:00 至 2021/01/27 12:00 **PM1.0** (2A & 2D 比較)
	
2021/01/13 12:00 至 01/20 12:00 **PM2.5** (2A & 2D 比較)	2021/01/20 12:00 至 2021/01/27 12:00 **PM2.5** (2A & 2D 比較)
	
2021/01/13 12:00 至 01/20 12:00 **PM10** (2A & 2D 比較)	2021/01/20 12:00 至 2021/01/27 12:00 **PM10** (2A & 2D 比較)

可觀察到三種不同大小的懸浮微粒在高低樓層的差異有類似的趨勢。我們注意到 1 月 17 日(黃框)差異值與其他部分有顯著的不同，

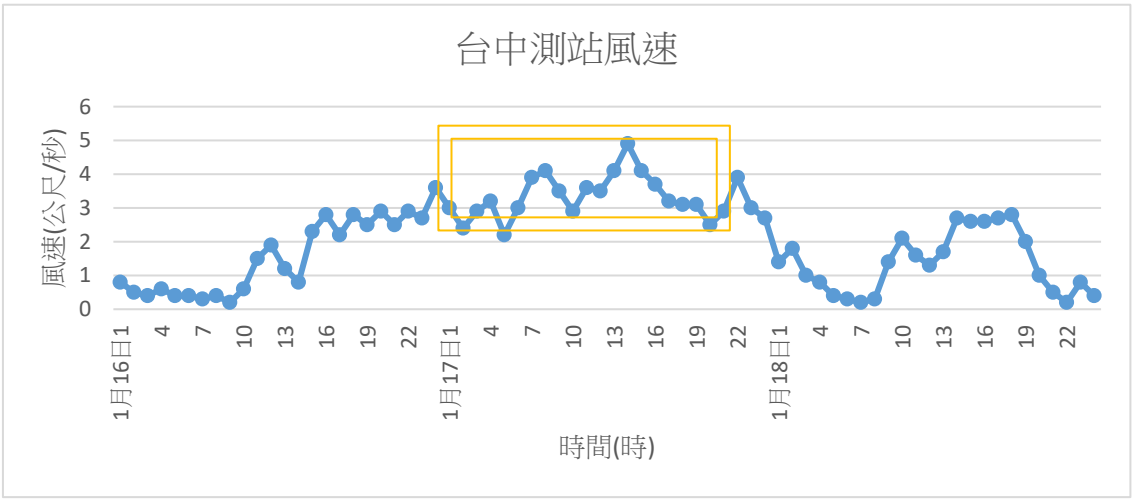


台中忠明測站 PM2.5 值(資料來源:環保署)



2021 年 1 月 16 日 1 時 至 1 月 18 日 24 時 中央氣象局 台中測站氣溫資料

(資料來源:中央氣象局)



2021 年 1 月 16 日 1 時 至 1 月 18 日 24 時 中央氣象局 台中測站風速資料

(資料來源:中央氣象局)

將台中忠明測站 PM2.5 值與大安國際大樓所測值比較，有相同趨勢。
由圖 2021/01/13 12:00 至 01/20 12:00 PM10 (2A & 2D 比較)及台中測站 1 月 17 日風速(兩黃框中)，推測風速愈大造成懸浮微粒測值較小，且可由自行測量資料發現上下樓層值差異較小，這也凸顯出三維偵測與一般平面偵測所提供數值的不同，對未來可有更多發展。

肆、 結論與應用

一、此偵測模型可於不同地區使用，探討汙染源(工廠煙囪、交通汙染)所影響該地區範圍進行環域分析及討論。

二、汙染物(懸浮微粒)擴散亦受到氣候影響，風向及風速為主要因素；而當地地理環境(地形、山脈、河谷)亦影響空氣流動方向。

三、將未來長期累積之三維監測數據加以分析，可了解該地區受氣候或地形之影響；若有足夠監測數據也可了解該地汙染源汙染因素及引響範圍，在未來汙染源改善空氣品質為一重要參考依據。

伍、 參考文獻

[1]林峻緯(2016)
2016 年細懸浮微粒成分討論-事件日與非事件日比較-以臺中市忠明測站為例
Discussion of Composition of PM2.5-Comparison of the Episodes and the Non-Episodes in 。中興大學 2016-Taichung Zhong-ming station
https://www.airitilibrary.com/Publication/alDetailedMesh?docid=U0005-2907201911330300
[2]陳思宇（2020）。能見度梯度與細及超細懸浮顆粒物分佈分佈關係（以台中地區為例）。國立中央大學
http://ir.lib.ncu.edu.tw:88/thesis/view_etd.asp?URN=106326001
[3]張基正(2018) 。中興大學
細懸浮微粒組成之季節性變化以及事件日成分組成探討-以 2018 台中市為例
Seasonal Variation of PM2.5 Compositions and Discussion of Episode – A case study in 2018 Taichung City
https://www.airitilibrary.com/Publication/alDetailedMesh?docid=U0005-3107201914551400
[4]謝怡茹(2020)。中興大學
台灣中南部細懸浮微粒分佈之時序特性
The Seasonal Trend Characteristics of PM2.5 in mid-south part of Taiwan
https://www.airitilibrary.com/Publication/alDetailedMesh?docid=U0005-2408202023424200

[5]Anipindi K (2014) An Introduction to ThingSpeak. *Internet of Things* https://www.codeproject.com/Articles/845538/An-Introduction-to-ThingSpeak

[6]Badura M, *et al*. (2018) Evaluation of Low-Cost Sensors for Ambient PM2.5 Monitoring. *Journal of Sensors* **2018.** DOI: 10.1155/2018/5096540.

[7]Feenstra B, *et al*. (2020) The AirSensor open-source R-package and DataViewer web application for interpreting community data collected by low-cost sensor networks. *Environmental Modelling & Software* **134**. DOI: 10.1016/j.envsoft.2020.104832.

[8]Scherer D, Dubois D and Sherwood B (2000) VPython: 3D interactive scientific graphics for students. *Computing in Science & Engineering* **2**: 56-62.

[9]Tseng CH, *et al*. (2019) The Relationship Between Air Pollution and Lung Cancer in Nonsmokers in Taiwan. *Journal of Thoracic Oncology. Journal of Thoracic Oncology* **14**: 784-792.

陸、 附錄

一、VPython 執行程式

```
import math

#各分量統一值設定
NumBallAll = 40
NumberOfBallA = NumBallAll
NumberOfBallB = NumBallAll
NumberOfBallC = NumBallAll
#註:NumberOfBall 數值不得為 0

# 定義值
e = 0.9          # 能量恢復係數
h = 50           # 小球初始離地高度
g = 9.8          # 地球重力加速度 9.8 m/s^2
t = 0            # 計算時間用參數
dt = 0.01        # 時間間隔

# 定義值
airDensity = 0.125           #空氣密度
particulateDensity = 1.06    #懸浮微粒密度
dragCoefficient = 0.47       #球體阻力係數
ballInitialVelocity = 50     #球體初始速度(倍)

# 球種設定
sizeRadiusA = 2      # A 小球半徑
sizeRadiusB = 0.5    # B 小球半徑
sizeRadiusC = 0.2    # C 小球半徑
# 註:A>B>C

#直徑設定
sizeDiameterA = sizeRadiusA*2 # A 球直徑
sizeDiameterB = sizeRadiusB*2 # B 球直徑
sizeDiameterC = sizeRadiusC*2 # C 球直徑

# 球體質量
m1 = particulateDensity*sizeRadiusA
m2 = particulateDensity*sizeRadiusB
m3 = particulateDensity*sizeRadiusC

#球體範圍放大倍率
```

```
ballPosPercent = 15

#球體投影面積
shadowAreaA = (sizeDiameterA**2)*pi/4
shadowAreaB = (sizeDiameterB**2)*pi/4
shadowAreaC = (sizeDiameterC**2)*pi/4


#設定畫布範圍
scene = canvas(title = " 真實世界水平拋射撞牆反彈的情形 ", width = 600, height
=600, x = 0, y = 0, center = vector(0, h/2, 0), background = vector(0.6,
0.6, 0.6))

#設定汙染源
chiemny1 = cylinder(pos = vector(0,-50,0), axis=vector(0,50,0), radius=5 )

#設定三個牆壁
floor1 = box(pos = vector(0, -50, 0), length = 400, height = 0.01, width =
10, color = color.blue)
floor2 = box(pos = vector(200, 0, 0), length = 1, height = 100, width = 10,
color = color.blue)
floor3 = box(pos = vector(-200, 0, 0), length = 1, height = 100, width =
10, color = color.blue)


#設定球 ABC，及球初速、加速、體積、質量
ballA=[]
NAc=NumberOfBallA #共有 N 個球
for  i in range(0,NAc):
    ball = sphere(pos = ballPosPercent*vector.random(), radius =
sizeRadiusA, color = color.red , make_trail=False)
    ballA.append(ball)
    ballA[i].v = ballInitialVelocity*vector.random()
    ballA[i].v.z = 0  #先讓所有球的 Z 方向速度為零
    ballA[i].pos.z = 0  #先讓所有球的 Z 方向速度為零
    ballA[i].a = vector(0, -g, 0) #y 方向加速度-g

    # print (i)
    print (ballA[i].v )

volumeA = (4/3)*pi*sizeRadiusA**3
massA = volumeA*particulateDensity
print (volumeA)
print (massA)
print (' ')


ballB=[]
NBc=NumberOfBallB #共有 N 個球
for  i in range(0,NBc):
```

```
    ball = sphere(pos = ballPosPercent*vector.random(), radius =
sizeRadiusB, color = color.green , make_trail=False)
    ballB.append(ball)
    ballB[i].v = ballInitialVelocity*vector.random()
    ballB[i].v.z = 0  #先讓所有球的 Z 方向速度為零
    ballB[i].pos.z  =0  #先讓所有球的 Z 方向速度為零
    ballB[i].a = vector(0, -g, 0) #y 方向加速度-g

    #print (i)
    print (ballB[i].v )

volumeB = (4/3)*pi*sizeRadiusB**3
massB = volumeB*particulateDensity
print (volumeB)
print (massB)
print (' ')

ballC=[]
NCc=NumberOfBallC #共有 N 個球
for  i in range(0,NCc):
    ball = sphere(pos = ballPosPercent*vector.random(), radius =
sizeRadiusC, color = color.black , make_trail=False)
    ballC.append(ball)
    ballC[i].v = ballInitialVelocity*vector.random()
    ballC[i].v.z = 0  #先讓所有球的 Z 方向速度為零
    ballC[i].pos.z = 0  #先讓所有球的 Z 方向速度為零
    ballC[i].a = vector(0, -g, 0) #y 方向加速度-g

    #print (i)


    print (ballC[i].v )

volumeC = (4/3)*pi*sizeRadiusC**3
massC = volumeC*particulateDensity
print (volumeC)
print (massC)
print (' ')

###############################################################################
#球體終端速度
yAxisTerminalVelocityA = -
(((2*massA*g)/(airDensity*shadowAreaA*dragCoefficient))**0.5) #A 球終端速度
yAxisTerminalVelocityB = -
(((2*massB*g)/(airDensity*shadowAreaB*dragCoefficient))**0.5) #B 球終端速度
yAxisTerminalVelocityC = -
(((2*massC*g)/(airDensity*shadowAreaC*dragCoefficient))**0.5) #C 球終端速度
print(yAxisTerminalVelocityA)
print(yAxisTerminalVelocityB)
print(yAxisTerminalVelocityC)
```

```
print (' ')
#############################################################################
#球體運動定義

#小球運動部分, 小球撞牆時反彈
while True :      #While true 裡的程式不斷執行
    rate(1)       #動畫更新時間間隔 1 秒


    #A 小球撞牆時反彈

    for i in range(0,NAc):
        ballA[i].v=ballA[i].v+ballA[i].a*dt # v=v0+a*t
        ballA[i].pos=ballA[i].pos+ballA[i].v*dt # S=s0+v*t
        #ballA[i].v.x=
        #ballA[i].v.y=
        if(abs(ballA[i].v.y) >= abs(yAxisTerminalVelocityA) ) : # 當 v.y >=
y 終端速度
            ballA[i].v.y = yAxisTerminalVelocityA    # 令 v.y = y 終端速度


    #撞地板的反彈 ball.y < floor.y

    for i in range(0,NAc):
        if(ballA[i].pos.y <= floor1.pos.y + ( sizeRadiusA +
floor1.height/2)  ):
            ballA[i].pos.y = floor1.pos.y+sizeRadiusA
            ballA[i].v.y = -ballA[i].v.y*e #反彈(v 反向)

    #撞右牆的反彈

        if( ballA[i].pos.x+sizeRadiusA >= floor2.pos.x  ) :
         ballA[i].pos.x = floor2.pos.x-sizeRadiusA
         ballA[i].v.x = -ballA[i].v.x*e #反彈(v 反向)

    #撞左牆的反彈

        if( ballA[i].pos.x-sizeRadiusA <= floor3.pos.x  ) :
         ballA[i].pos.x = floor3.pos.x+sizeRadiusA
         ballA[i].v.x = -ballA[i].v.x*e #反彈(v 反向)



    #\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\


    #B 小球撞牆時反彈
```

```
    for i in range(0,NBc):
        ballB[i].v=ballB[i].v+ballB[i].a*dt # v=v0+a*t
        ballB[i].pos=ballB[i].pos+ballB[i].v*dt # S=s0+v*t
        #ballB[i].v.x=
        #ballB[i].v.y=
        if(abs(ballB[i].v.y) >= abs(yAxisTerminalVelocityB) ) : # 當 v.y >=
y 終端速度
            ballB[i].v.y = yAxisTerminalVelocityB    # 令 v.y = y 終端速度

    #撞地板的反彈 ball.y < floor.y

    for i in range(0,NBc):
        if(ballB[i].pos.y <= floor1.pos.y + ( sizeRadiusB +
floor1.height/2)  ):
            ballB[i].pos.y = floor1.pos.y+sizeRadiusB
            ballB[i].v.y = -ballB[i].v.y*e #反彈(v 反向)

    #撞右牆的反彈

        if( ballB[i].pos.x+sizeRadiusB >= floor2.pos.x  ) :
         ballB[i].pos.x = floor2.pos.x-sizeRadiusB
         ballB[i].v.x = -ballB[i].v.x*e #反彈(v 反向)

    #撞左牆的反彈

        if( ballB[i].pos.x-sizeRadiusB <= floor3.pos.x  ) :
         ballB[i].pos.x = floor3.pos.x+sizeRadiusB
         ballB[i].v.x = -ballB[i].v.x*e #反彈(v 反向)


    #\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\


    #C 小球撞牆時反彈

    for i in range(0,NCc):
        ballC[i].v=ballC[i].v+ballC[i].a*dt # v=v0+a*t
        ballC[i].pos=ballC[i].pos+ballC[i].v*dt # S=s0+v*t
        #ballC[i].v.x=
        #ballC[i].v.y=
        if(abs(ballC[i].v.y) >= abs(yAxisTerminalVelocityC) ) : # 當 v.y >=
y 終端速度
            ballC[i].v.y = yAxisTerminalVelocityC    # 令 v.y = y 終端速度

    #撞地板的反彈 ball.y < floor.y

    for i in range(0,NCc):
```

```
        if(ballC[i].pos.y <= floor1.pos.y + ( sizeRadiusC +
floor1.height/2)  ):
            ballC[i].pos.y = floor1.pos.y+sizeRadiusC
            ballC[i].v.y = -ballC[i].v.y*e #反彈(v 反向)

    #撞右牆的反彈

        if( ballC[i].pos.x+sizeRadiusC >= floor2.pos.x  ) :
         ballC[i].pos.x = floor2.pos.x-sizeRadiusC
         ballC[i].v.x = -ballC[i].v.x*e #反彈(v 反向)

    #撞左牆的反彈

        if( ballC[i].pos.x-sizeRadiusC <= floor3.pos.x  ) :
         ballC[i].pos.x = floor3.pos.x+sizeRadiusC
         ballC[i].v.x = -ballC[i].v.x*e #反彈(v 反向)



    #-------------------------------------------------------------------------


    # 小球彼此間碰撞反彈 A to A

        n=0
        for i in range(n,NAc-1) :
            for j in range(n+1,NAc) :
                sd=ballA[i].pos- ballA[j].pos
                if (mag(sd) < 2*sizeRadiusA) :  #如果兩球心間距離小於球直徑，
下面要帶彈性碰撞公式

                    tempb=proj( ballA[i].v ,ballA[i].pos-ballA[j].pos ) # i
球在兩球心方向上的速度投影
                    tempc=proj( ballA[j].v ,ballA[i].pos-ballA[j].pos ) # j
球在兩球心方向上的速度投影
                  # print(tempb,tempc)
                    ballA[i].v=ballA[i].v -tempb+tempc
                    ballA[j].v=ballA[j].v -tempc+tempb

                #   ballA[i].pos=ballA[i].pos+ballA[i].v*dt # S=s0+v*t
                    ballA[j].pos=ballA[j].pos - sd*( 1-
mag(sd)/(2*sizeRadiusA) )


#////////////////////////////////////////////////////////////////////////


    # 小球彼此間碰撞反彈 B to B

        n=0
```

```
        for i in range(n,NBc-1) :
            for j in range(n+1,NBc) :
                sd=ballB[i].pos- ballB[j].pos
                if (mag(sd) < 2*sizeRadiusB) :  #如果兩球心間距離小於球直徑，
下面要帶彈性碰撞公式

                    tempb=proj( ballB[i].v ,ballB[i].pos-ballB[j].pos ) # i
球在兩球心方向上的速度投影
                    tempc=proj( ballB[j].v ,ballB[i].pos-ballB[j].pos ) # j
球在兩球心方向上的速度投影
                   # print(tempb,tempc)
                    ballB[i].v=ballB[i].v -tempb+tempc
                    ballB[j].v=ballB[j].v -tempc+tempb

                #   ballB[i].pos=ballB[i].pos+ballB[i].v*dt # S=s0+v*t
                    ballB[j].pos=ballB[j].pos - sd*( 1-
mag(sd)/(2*sizeRadiusB) )


#////////////////////////////////////////////////////////////////////////


    # 小球彼此間碰撞反彈 C to C

        n=0
        for i in range(n,NCc-1) :
            for j in range(n+1,NCc) :
                sd=ballC[i].pos- ballC[j].pos
                if (mag(sd) < 2*sizeRadiusC) :  #如果兩球心間距離小於球直徑，
下面要帶彈性碰撞公式

                    tempb=proj( ballC[i].v ,ballC[i].pos-ballC[j].pos ) # i
球在兩球心方向上的速度投影
                    tempc=proj( ballC[j].v ,ballC[i].pos-ballC[j].pos ) # j
球在兩球心方向上的速度投影
                   # print(tempb,tempc)
                    ballC[i].v=ballC[i].v -tempb+tempc
                    ballC[j].v=ballC[j].v -tempc+tempb

                #   ballC[i].pos=ballC[i].pos+ballC[i].v*dt # S=s0+v*t
                    ballC[j].pos=ballC[j].pos - sd*( 1-
mag(sd)/(2*sizeRadiusC) )


#////////////////////////////////////////////////////////////////////////

 # 小球彼此間碰撞反彈 A to B

        n=0
        for i in range(n,NAc-1) :
```

```
            for j in range(n+1,NBc) :
                sd=ballA[i].pos- ballB[j].pos
                if (mag(sd) < sizeRadiusA+sizeRadiusB) :  #如果兩球心間距離小
於球直徑，下面要帶彈性碰撞公式

                    tempb=proj( ballA[i].v ,ballA[i].pos-ballB[j].pos ) # i
球在兩球心方向上的速度投影
                    tempc=proj( ballB[j].v ,ballA[i].pos-ballB[j].pos ) # j
球在兩球心方向上的速度投影
                    print(tempb,tempc)
                    ballA[i].v=ballA[i].v -tempb+tempc
                    ballB[j].v=ballB[j].v -tempc+tempb

                 #   ballA[i].pos=ballA[i].pos+ballA[i].v*dt # S=s0+v*t
                    ballB[j].pos=ballB[j].pos - sd*( 1-
mag(sd)/(sizeRadiusA+sizeRadiusB) )


# 小球彼此間碰撞反彈 B to C

        n=0
        for i in range(n,NBc-1) :
            for j in range(n+1,NCc) :
                sd=ballB[i].pos- ballC[j].pos
                if (mag(sd) < sizeRadiusB+sizeRadiusC) :  #如果兩球心間距離小
於球直徑，下面要帶彈性碰撞公式

                    tempb=proj( ballB[i].v ,ballB[i].pos-ballC[j].pos ) # i
球在兩球心方向上的速度投影
                    tempc=proj( ballC[j].v ,ballB[i].pos-ballC[j].pos ) # j
球在兩球心方向上的速度投影
                    print(tempb,tempc)
                    ballB[i].v=ballB[i].v -tempb+tempc
                    ballC[j].v=ballC[j].v -tempc+tempb

                 #   ballA[i].pos=ballA[i].pos+ballA[i].v*dt # S=s0+v*t
                    ballC[j].pos=ballC[j].pos - sd*( 1-
mag(sd)/(sizeRadiusB+sizeRadiusC) )


# 小球彼此間碰撞反彈 A to C

        n=0
        for i in range(n,NAc-1) :
            for j in range(n+1,NCc) :
                sd=ballA[i].pos- ballC[j].pos
                if (mag(sd) < sizeRadiusA+sizeRadiusC) :  #如果兩球心間距離小
於球直徑，下面要帶彈性碰撞公式
```

```
                    tempb=proj( ballA[i].v ,ballA[i].pos-ballC[j].pos ) # i
球在兩球心方向上的速度投影
                    tempc=proj( ballC[j].v ,ballA[i].pos-ballC[j].pos ) # j
球在兩球心方向上的速度投影
                    print(tempb,tempc)
                    ballA[i].v=ballA[i].v -tempb+tempc
                    ballC[j].v=ballC[j].v -tempc+tempb

                #   ballA[i].pos=ballA[i].pos+ballA[i].v*dt # S=s0+v*t
                    ballC[j].pos=ballC[j].pos - sd*( 1-
mag(sd)/(sizeRadiusA+sizeRadiusC) )


#設定新時間
t=t+dt
```

二、Arduino (PMS3003 / DHT-11 / ESP8266_01s)程式

```
#include <SoftwareSerial.h>

// DHT
#include <DHT.h>

//for saving memory
//#define dhtPin 7      //讀取 DHT11 Data
//#define dhtType DHT11 //選用 DHT11
//DHT dht(dhtPin, dhtType); // Initialize DHT sensor

DHT dht(7, DHT11); // Initialize DHT sensor


SoftwareSerial PmsSerial(2, 3); // RX, TX
SoftwareSerial WifiSerial(4, 5); // RX, TX

static unsigned int pm_cf_10=0;            //定義全域變數
static unsigned int pm_cf_25=0;
static unsigned int pm_cf_100=0;
static unsigned int pm_at_10=0;
static unsigned int pm_at_25=0;
static unsigned int pm_at_100=0;



//constant Setup//
//for saving memory; setup in loop
//unsigned long nWaitingTime = 32767;  //每 300 秒測一次
//unsigned long nWaitingTime = 300000;  //每 300 秒測一次

//Other Setup//
//int nFirstTime=1 ;

//int i ;
//int nResult;


/*
 * choose the correct wifi SSID and PSW
 */

#define SSID  "指定 SSID"
#define PASS  "指定 PASS"


//ThinkSpeak Setup//
#define IP "184.106.153.149" // thingspeak.com

/*
 * choose the correct API key of each channel
```

```
 *
 * THIS IS *** SENSOR
 */


//指定 channel
String sApiKey1=" 指定 Write API KEY ";


void fnSendDebug(String lsCmd)
{

    Serial.print("SEND: ");
    Serial.println(lsCmd);

    WifiSerial.println(lsCmd);
}

void fnConnectingWifi(String lsState){
    int lnTimeOut;
    for (lnTimeOut=0 ; lnTimeOut<10 ; lnTimeOut++)
    {
      if(WifiSerial.find("OK"))
      {
          WifiSerial.println("RECEIVED: OK");
          break;
      }
      else if(lnTimeOut==9){
       Serial.print( lsState );
       Serial.println(" fail...\nExit2");
      }
      else
      {
        delay(500);
      }
    }
}

 void fnWifiConnection()
{
    String lsCmd="AT+CWJAP=\"";
    lsCmd+=SSID;
    lsCmd+="\",\"";
    lsCmd+=PASS;
    lsCmd+="\"";
    fnSendDebug(lsCmd);
    fnConnectingWifi("Wifi_connecting");
}

boolean fnSetWifi()
{
    WifiSerial.println("AT+CWMODE=1");
    fnWifiConnection();
}
```

```
void fnIniWifi()
{
   fnSendDebug("AT");
   fnConnectingWifi("sent AT");
   Serial.println("going to initialize wifi");
   fnSetWifi();
}

//int fnSentOnCloud(String lsApiKey, String lsF1, String lsF2, String lsF3,
String lsF4, String lsF5, String lsF6)
//int fnSentOnCloud(String lsApiKey, String lsF1, String lsF2, String lsF3,
String lsF4, String lsF5, String lsF6, String lsF7)
int fnSentOnCloud(String lsApiKey, String lsF1, String lsF2, String lsF3,
String lsF4, String lsF5, String lsF6, String lsF7, String lsF8)
{
  String lsGET = "GET /update?key=";

    // 設定 ESP8266 作為 Client 端
    String lsCmd = "AT+CIPSTART=\"TCP\",\"";
    lsCmd += IP;
    lsCmd += "\",80";
    fnSendDebug(lsCmd);
    if( WifiSerial.find( "Error" ) )
    {
        Serial.print( "RECEIVED: Error\nExit1" );
        return;
    }

    lsGET= lsGET + lsApiKey;

    //lsCmd = lsGET + "&field1=" + lsF1 + "&field2=" + lsF2 + "&field3=" +
lsF3 + "&field4=" + lsF4 + "&field5=" + lsF5 + "&field6=" + lsF6 +"\r\n";
    lsCmd = lsGET + "&field1=" + lsF1 + "&field2=" + lsF2 + "&field3=" +
lsF3 + "&field4=" + lsF4 + "&field5=" + lsF5 + "&field6=" + lsF6 +
"&field7=" + lsF7 + "&field8=" + lsF8 +"\r\n";
    //lsCmd = lsGET +
"&field1=100&field2=102&field3=120&field4=120&field5=180&field6=120&field7=
10&field8=99\r\n";

    WifiSerial.print( "AT+CIPSEND=" );

    //Serial.print("lsCmd: "+lsCmd);
    Serial.println(lsCmd.length());

    WifiSerial.println( lsCmd.length() );
    if(WifiSerial.find( ">" ) )
    {
        Serial.print(">");
        Serial.print(lsCmd);
        WifiSerial.print(lsCmd);
    }
    else
    {
```

```
        WifiSerial.print( "AT+CIPCLOSE" );
    }
    if( WifiSerial.find("OK") )
    {
        Serial.println( "RECEIVED: OK" );
        return 1;
    }
    else
    {
        Serial.println( "RECEIVED: Error\nExit2" );
        return 0;
    }
}



void getG5(unsigned char ucData)//獲取 G5 的值
{
  static unsigned int ucRxBuffer[250];
  static unsigned int ucRxCnt = 0;
  ucRxBuffer[ucRxCnt++] = ucData;
  if (ucRxBuffer[0] != 0x42 && ucRxBuffer[1] != 0x4D)//資料頭判斷
  {
    ucRxCnt = 0;
    return;
  }

  if (ucRxCnt > 16)//資料位元數判斷//G5T 為 16
  {
       pm_cf_10=(int)ucRxBuffer[4] * 256 + (int)ucRxBuffer[5];      //大氣
環境下 PM2.5 濃度計算
       pm_cf_25=(int)ucRxBuffer[6] * 256 + (int)ucRxBuffer[7];
       pm_cf_100=(int)ucRxBuffer[8] * 256 + (int)ucRxBuffer[9];
       pm_at_10=(int)ucRxBuffer[10] * 256 + (int)ucRxBuffer[11];
       pm_at_25=(int)ucRxBuffer[12] * 256 + (int)ucRxBuffer[13];
       pm_at_100=(int)ucRxBuffer[14] * 256 + (int)ucRxBuffer[15];


    if (pm_cf_25 >  999 || pm_cf_10 > 999 || pm_cf_100 >999)//如果 PM2.5 數值
>1000，返回重新計算
    {
      ucRxCnt = 0;
      return;
    }

    ucRxCnt = 0;
    return;
  }

}

void setup() {
```

```
   // DHT
   dht.begin();//啟動 DHT

   PmsSerial.begin(9600);
   PmsSerial.setTimeout(1500);
   WifiSerial.begin(115200);
   //WifiSerial.begin(9600);
   //WifiSerial.setTimeout(1500);
   Serial.begin(115200);

   //測試區
   //Serial.print("PM_CF1.0");Serial.print(",");
   //Serial.print("PM_CF2.5");Serial.print(",");
   //Serial.print("PM_CF10");Serial.print(",");
   //Serial.print("PM_AQI1.0");Serial.print(",");
   //Serial.print("PM_AQI2.5");Serial.print(",");
   //Serial.print("PM_AQI10");Serial.print(",");
   //Serial.print("(ug/m3)");Serial.print(",");
   //Serial.print("(ug/m3)");Serial.print(",");
   //Serial.print("(ug/m3)");Serial.print(",");
   //Serial.print("(ug/m3)");Serial.print(",");
   //Serial.print("(ug/m3)");Serial.print(",");
   //Serial.println("(ug/m3)");


   // test the connection to wifi AP
   WifiSerial.listen();
   /*
   if (WifiSerial.isListening())
   {
     Serial.println("WifiSerial.isListening");
   }else
   {
     Serial.println("WifiSerial.is not Listening");
   }
   */
    fnIniWifi();

}

int nFirstTime=1 ;

void loop()
{
  //Other Setup//

  int i ;
  int nResult;

  ///*
  if (nFirstTime==1)
  {
    delay(100);   // 0.1 second
    nFirstTime=0;
```

```
  }else
  {
    //delay(5000);
    //delay(nWaitingTime);
    //delay(32767);
    delay(300000);
  }

  ///*
  PmsSerial.listen();
  /*
  if (PmsSerial.isListening())
  {
    Serial.println("PmsSerial.isListening");
  }else
  {
    Serial.println("PmsSerial.is not Listening");
  }
  */

  while (PmsSerial.available()) {
    getG5(PmsSerial.read());
  }

  // this part is for retrieving useful values
  if (pm_cf_10 ==0 && pm_cf_25 ==0 && pm_cf_100 ==0)
  {
    Serial.println("All are 0s");
    // new added; restart
    nFirstTime=1;
    delay(1000);   // avoid to read too many times
    return;
  }

  //Serial.print(pm_cf_10);Serial.print(",");
  //Serial.print(pm_cf_25);Serial.print(",");
  Serial.println(pm_cf_10);

//  Serial.print(pm_at_10);Serial.print(",");
//  Serial.print(pm_at_25);Serial.print(",");
//  Serial.println(pm_at_100);

  //*/


  delay(3000);

 ///*

  //DHT part
//  float h = dht.readHumidity();//讀取濕度
//  float t = dht.readTemperature();//讀取攝氏溫度
//  if (isnan(h) || isnan(t)) {
```

```
//    Serial.println("無法從 DHT 傳感器讀取！");
//    return;
//  }




  WifiSerial.listen();
  if (WifiSerial.isListening())
  {
    Serial.println("WifiSerial.isListening");
  }else
  {
    Serial.println("WifiSerial.is not Listening");
  }
  delay(500);


  // try to connect wifi 10 times if errors happen
  for(i=0; i<5; i++)
  {
    //nResult=fnSentOnCloud(sApiKey1, String(pm_cf_10),
String(pm_cf_25),String(pm_cf_100),String(pm_at_10),String(pm_at_25),String
(pm_at_100));
    //nResult=fnSentOnCloud(sApiKey1, String(pm_cf_10),
String(pm_cf_25),String(pm_cf_100),String(pm_at_10),String(pm_at_25),String
(pm_at_100),String(dht.readHumidity()),String(dht.readTemperature()));
    nResult=fnSentOnCloud(sApiKey1, String(pm_cf_10),
String(pm_cf_25),String(pm_cf_100),String(pm_at_10),String(pm_at_25),String
(pm_at_100),String(int(dht.readHumidity())),String(dht.readTemperature()));
    //nResult=fnSentOnCloud(sApiKey1, String(1),
String(2),String(3),String(4),String(5),String(6),String(7),String(8));

    Serial.print("nResult: ");Serial.println(nResult);
    // check if sent
    if(nResult ==1)
    {
      // if sent successufully, then do new job
      break;
    }

    // if get errors too many times, redo initialize wifi
    if (i==2)
    {
      fnIniWifi();
      nFirstTime=1;
      break;
    }
    delay(3000);
  }

  delay(2000);
```

```
  pm_cf_10=0;
  pm_cf_25=0;
  pm_cf_100=0;
  pm_at_10=0;
  pm_at_25=0;
  pm_at_100=0;

}
```

【評語】200022

本研究以自製的偵測器(共四代)在不同高層處進行空氣品質監測，並利用 Arduino 以及 ThingSpeak 構成遠端監控系統，可提供即時的觀測資料，極富創意。建議可針對長期累積的空氣品質監測數據做進一步的判讀與分析。