

國立高雄科技大學
智慧商務系

資料結構期末專題報告

刪除兩側極端值再求平均值應用
-智慧植物照護系統

C109156226 黃柏凱

指導教授：謝文川

中華民國 112 年 1 月

目錄

第一章、緒論	1
第一節、研究動機及目的	1
第二節、問題討論與改善方法	1
第二章、實作與展示	2
第一節、系統架構	2
第二節、快速排序法 (QuickSort 函數)	3
第三節、剔除兩側極端值再取平均值(Trim_Mean 函數)	4
第四節、新舊程式比較與執行結果	6
第三章、結論	8
參考文獻	9

圖目錄

圖 1. 電路圖	2
圖 2. 快速排序法(Quick Sort) 函數	3
圖 3. 剔除兩側極端值再取平均值(Trim_Mean) 函數	5
圖 4. 新舊程式比較圖	6
圖 5. 執行結果	7
圖 6. 完成成品	7

第一章、緒論

第一節、研究動機及目的

「鋤禾日當午，汗滴禾下土」這句話所描述的就是農夫在農田認真工作的樣子。但是以位於赤道上的熱帶國家「台灣」，每至夏日時中午的氣溫攝氏溫度一下子就動輒 36、37 度左右，再加上工作時的疲勞，就很容易有中暑的情況發生，在新聞上偶爾也會看到有農夫因為中暑而倒在農田裡，無人發現的憾事，等到家屬發現時也來不及救急了。如果能夠避免在高溫的氣候下出門工作，就可以避免許多類似的事件發生。

我希望做出可以自動檢測農場環境數據，並依照節氣、氣候、時段自動判斷及執行澆水的控制系統，這樣就可以讓農夫們不用頂著炎炎烈日，艱苦的照顧農作物，在家就可以輕鬆的掌握農場狀況，並管理好農田。

第二節、問題討論與改善方法

1. 問題討論

在二年級的「感測器與物聯網實作」課程中，已經完成系統建置與程式開發。原本的作法是感測器在偵測數據之後，將每次收集到的數據上傳至伺服器保存。然而這個作法的缺點是，有時感測到的數據可能因為元件不穩定而偵測到差異比較大的數據，這些數據將造成判讀上的困擾。

2. 改善方法

為獲得更穩定且有效的數據，我將以這學期「資料結構」課程學習到的知識進行系統改善。我們將使用「陣列、快速排序法」來記錄及處理收集到的感測數據，並剔除陣列兩側的極端值後，再求取平均值作為最後要上傳至伺服器保存的數據。改善步驟如下：

- (1). 使用陣列(Array)儲存感測資料
- (2). 使用快速排序法(Quick Sort)將感測數據排序
- (3). 排除陣列最前面與最後面的極端值後，中間的數值再取平均值

第二章、實作與展示

第一節、系統架構

Wemos D1 mini 開發板直接透過無線網路基地台連線上網。此外，Wemos D1 mini 開發板定期檢測空氣溫溼度、土壤溫度、土壤溼度後，直接以 MQTT 方式傳送至 ThingSpeak 平台。在執行澆水動作方面，會參考氣象開放資料平臺的資料，如果是下雨天，則不澆水。另外，如果在澆水之後，土壤溼度沒有上升，則有可能發生缺水狀況，系統會自動透過 IFTTT 平台發送 Line 訊息給農夫，通知農夫即時作處置。本系統的電路圖如圖 1 所示。

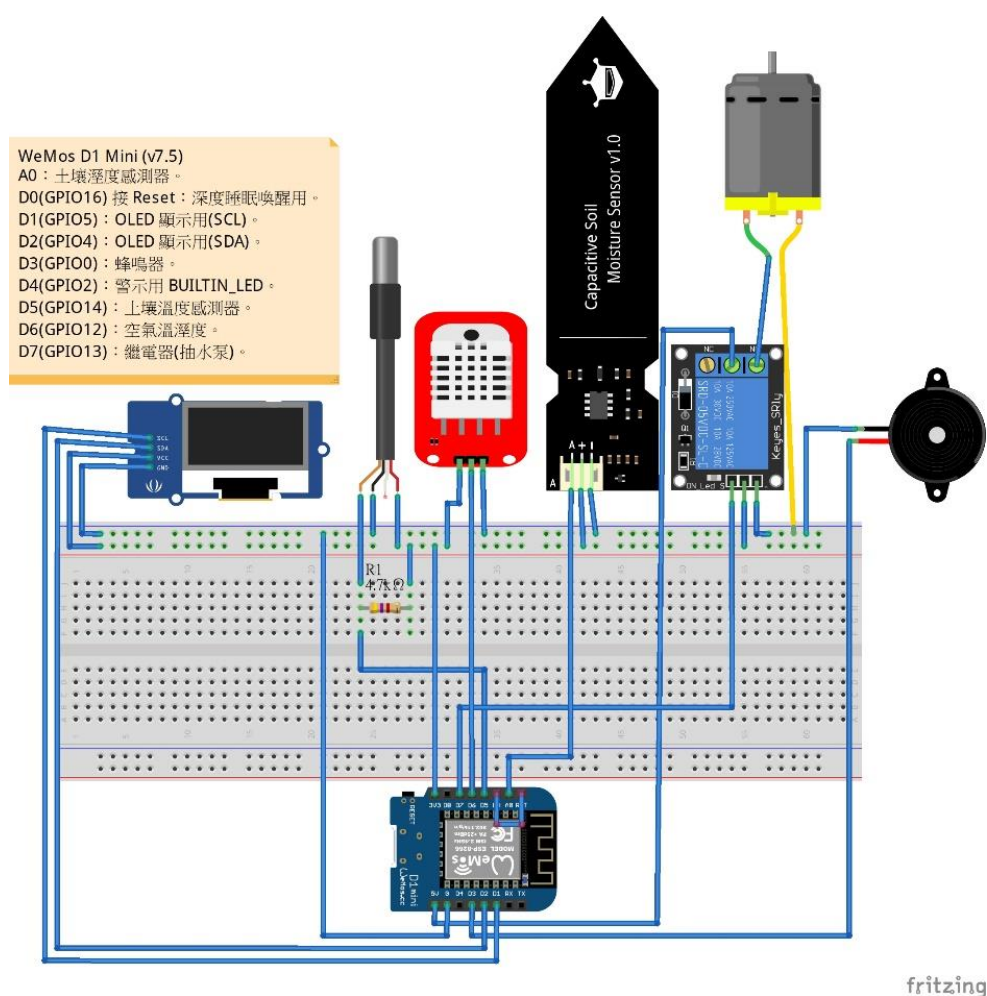


圖 1. 電路圖

第二節、快速排序法 (QuickSort 函數)

QuickSort 函數功能：

- 將傳入值 aSort 陣列使用快速排序法將元素由小到大排列。

QuickSort 函數具有下列引數(程式如圖 2 所示)：

- aSort：必要值，資料型態為“陣列(Array)”，這是要用來排序的陣列。

回傳值：

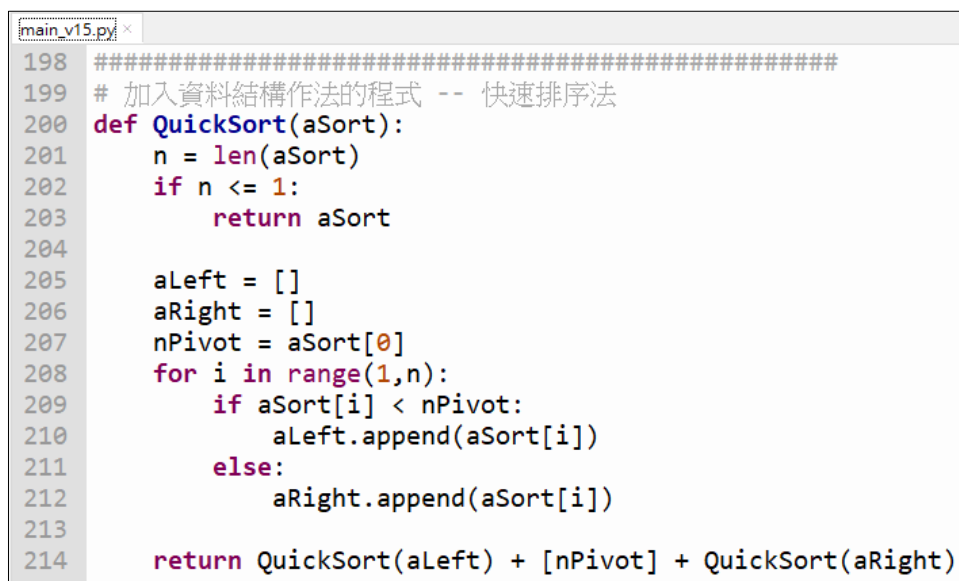
- 資料型態為“陣列(Array)”，這是已經由小到大排序後的陣列。

程式範例：

```
data = [85.5,80.5,84.4,87.05,78.25]
print("Data: {}".format(data))
data = QuickSort(data)
print("QuickSort: {}".format(data))
```

輸出：

```
Data: [85.5, 80.5, 84.4, 87.05, 78.25]
QuickSort: [78.25, 80.5, 84.4, 85.5, 87.05]
```

A screenshot of a code editor window titled 'main v15.py'. The code defines a QuickSort function. It starts with a comment in Chinese: '# 加入資料結構作法的程式 -- 快速排序法'. The function 'def QuickSort(aSort):' takes a list 'aSort' as input. It calculates the length 'n = len(aSort)'. If 'n' is less than or equal to 1, it returns 'aSort'. Otherwise, it creates two empty lists 'aLeft' and 'aRight', and selects the first element 'nPivot = aSort[0]'. It then iterates through 'aSort' from index 1 to 'n-1'. For each element, if it is less than 'nPivot', it is appended to 'aLeft'; otherwise, it is appended to 'aRight'. Finally, the function returns the concatenation of 'QuickSort(aLeft)', 'nPivot', and 'QuickSort(aRight)'.

```
198 #####
199 # 加入資料結構作法的程式 -- 快速排序法
200 def QuickSort(aSort):
201     n = len(aSort)
202     if n <= 1:
203         return aSort
204
205     aLeft = []
206     aRight = []
207     nPivot = aSort[0]
208     for i in range(1,n):
209         if aSort[i] < nPivot:
210             aLeft.append(aSort[i])
211         else:
212             aRight.append(aSort[i])
213
214     return QuickSort(aLeft) + [nPivot] + QuickSort(aRight)
```

圖 2. 快速排序法(Quick Sort) 函數

第三節、剔除兩側極端值再取平均值(Trim_Mean 函數)

Trim_Mean 函數功能：

- 將傳入值 aSort 陣列剔除陣列兩側的極端值後，再求取平均值。

Trim_Mean 函數具有下列引數(程式如圖 3 所示)：

- aSort：必要值，資料型態為“陣列(Array)”，這是要用來計算平均值的原始資料。
- fPercentage：必要值，料型態為“浮點數(Float)”，這是要用來剔除陣列兩側極端值的百分比。例如，如果 fPercentage = 0.4，則會從含有 5 個資料點 (5 x 0.4) 的資料集中消除 2 個資料點，亦即會消除資料集的最前面一個資料點和最後面一個資料點。

回傳值：

- 資料型態為“浮點數(Float)”，這是剔除陣列兩側的極端值後，再求取平均值(小數點取一位)。

程式範例：

```
data = [85.5,80.5,84.4,87.05,78.25]
print("Data: {}".format(data))
data = QuickSort(data)
print("QuickSort: {}".format(data))
print("Trim_Mean(Data, 0.4)= {}".format(Trim_Mean(data, 0.4)))
```

輸出：

```
Data: [85.5, 80.5, 84.4, 87.05, 78.25]
QuickSort: [78.25, 80.5, 84.4, 85.5, 87.05]
Sum = 250.4
Count = 3
Trim_Mean(Data, 0.4)= 83.5
```

```
main_v15.py ×
216 #####
217 # 加入資料結構作法的程式 -- 去除極端值後，再求平均數
218 def Trim_Mean(aSort, fPercentage):
219     n = round(len(aSort) * fPercentage / 2)
220     nSum = 0
221     nCount = 0
222
223     for i in range(n, len(aSort)-n):
224         nSum += aSort[i]
225         nCount += 1
226
227     print("Sum =", nSum)
228     print("Count =", nCount)
229     return round(nSum/nCount, 1)
```

圖 3. 剔除兩側極端值再取平均值(Trim_Mean) 函數

第四節、新舊程式比較與執行結果

原本的程式在收集完感測數據後，就以 MQTT 通訊上傳至 ThingSpeak 雲端平台，如圖 4 第 340~341 行程式所示。

第 359~374 行程式是收集完 5 次的數據後，將數據經由我們的改善方式，剔除兩側極端值再取平均值，再將數據上傳至 ThingSpeak 雲端平台。第 360~364 行程式為處理空氣溫度的程式，第 366~371 行程式為處理空氣溼度的程式，我們剔除極端值的百分比都是 0.4。

因為 ThingSpeak 免費版本，需每隔 15 秒才能傳送一次資料，所以原本的程式需考量這個間隔時間。但經過改善方法後，上傳數據的間隔時間一定會超過 15 秒，所以就註解掉第 343~351 行的程式。



```
main_v15.py x
340 # print("Send data to ThingSpeak") # 傳送資料到 ThingSpeak 平台
341 # sendMQTT()
342
343 # # 因為 ThingSpeak 免費版本，只能每隔 15 秒傳送一次資料，所以需休息幾秒
344 # t = 15
345 # if isWatering :
346 #     t = t - waterSeconds # 如果有澆水的話，就減掉澆水時間
347 # if isWaterScarcity :
348 #     t = t - 6 # 如果有缺水的話，就減掉傳送 LINE 警告訊息的時間
349 # if count > 0:
350 #     print("Rest {} secs for upload data to ThingSpeak".format(t))
351 #     time.sleep(t)
352
353 t = 5
354 print("Rest {} secs".format(t))
355 time.sleep(t)
356
357 print("")
358 # 抓 5 次資料後，就進入深度睡眠
359 if count >= 5:
360     # 加入資料結構作法的程式 -- 剔除兩側極端值再求平均值
361     print("airTemperature: {}".format(array_airTemperature))
362     array_airTemperature = QuickSort(array_airTemperature)
363     print("QuickSort: {}".format(array_airTemperature))
364     airTemp = Trim_Mean(array_airTemperature, 0.4)
365     print("airTemp: {}".format(airTemp))
366
367     print()
368     print("airHumidity: {}".format(array_airHumidity))
369     array_airHumidity = QuickSort(array_airHumidity)
370     print("QuickSort: {}".format(array_airHumidity))
371     airHum = Trim_Mean(array_airHumidity, 0.4)
372     print("airHum: {}".format(airHum))
373
374     print("Send data to ThingSpeak") # 傳送資料到 ThingSpeak 平台
375     sendMQTT()
```

圖 4. 新舊程式比較圖

圖 5 為經過改善後的執行結果，圖 6 為完成的作品。空氣溫度的 5 次數據分別為 [21.5, 21.5, 21.5, 21.5, 21.6]，經快速排序法排序後，再剔除前後各一筆資料，剩下的數據為 [21.5, 21.5, 21.5]，其平均值為 21.5。

空氣溼度的 5 次數據分別為 [50.1, 50.0, 50.0, 49.9, 49.9]，經快速排序法排序後，再剔除前後各一筆資料，剩下的數據為 [49.9, 50.0, 50.0]，其平均值為 50.0。

```
互動環境 (Shell) ×  
  
airTemperature: [21.5, 21.5, 21.5, 21.5, 21.6]  
QuickSort: [21.5, 21.5, 21.5, 21.5, 21.6]  
Sum = 64.5  
Count = 3  
airTemp: 21.5  
  
airHumidity: [50.1, 50.0, 50.0, 49.9, 49.9]  
QuickSort: [49.9, 49.9, 50.0, 50.0, 50.1]  
Sum = 149.9  
Count = 3  
airHum: 50.0  
Send data to ThingSpeak  
  
Deep Sleep  
Wake up after 600 Sec.
```

計算後的平均溫度

計算後的平均溼度

將數據上傳至 ThingSpeak

圖 5. 執行結果

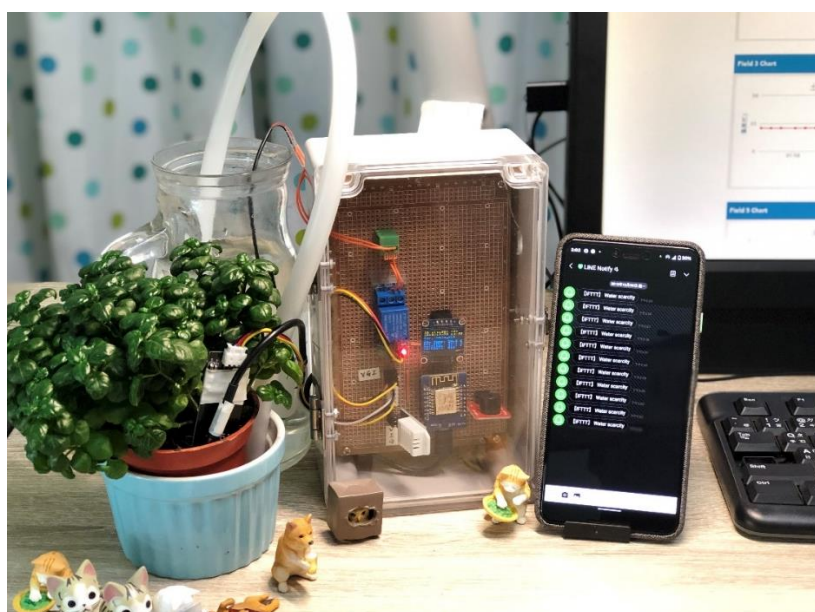


圖 6. 完成成品

第三章、結論

運用到上課所學到的知識，改善之前的智慧植物照護系統並且增加了下列優點：

1. 避免資料錯誤問題：

由於感測器偶爾可能出現不穩定造成資料錯誤。所以現在在藉由刪除極端值後，再求取平均值可以避開錯誤的感測資料，讓數據更有意義。

2. 減少不必要的上傳：

以前要上傳 5 次資料，現在只要上傳一次就好，節省網路傳輸資料量，並避免上傳相似資料過多造成資料爆炸。

3. 可以更省電：

Wifi 相比藍芽傳輸較耗電，在不影響數據有效性的前題下，減少上傳次數，說不定也可以讓整個系統更省電。

以上就是我的期末報告，在這學期的上課當中最印象深刻的是老師帶我們去高雄港旁的軟體園區，在參訪的過程當中也清楚的了解現在業界已經在實際運用上，是如何結合 5G 去做運用，並且也經過解說員的解說間接了解到現在業界缺少哪一方面的人才，只能說，在上課的過程當中不斷接收老師教導的知識，「讓我從原本的程式能跑就好的」觀念完全改觀，並且瞭解了到真正一個好的程式必須要有良好的資料結構撰寫習慣，在某些狀況下可以大幅改善程式運作的時間，並且使電腦在執行時可以使運作效率加倍，很慶幸我當初有選這堂課，吸收到跟以往其他課相比完全不同的知識。

參考文獻

- Damien P., George; Sokolovsky, Paul. (2023 年 1 月 24 日). Quick reference for the ESP32. 擷取自 MicroPython: <https://docs.micropython.org/en/latest/esp32/quickref.html>
- Microsoft. (2023 年 1 月 26 日). TRIMMEAN 函數. 擷取自 Microsoft 365 支援服務: <https://support.microsoft.com/zh-tw/office/trimmean-%E5%87%BD%E6%95%B8-d90c9878-a119-4746-88fa-63d988f511d3>
- Vincent. (2010 年 6 月 3 日). Excel-計算去極值的平均. 擷取自 學不完·教不停·用不盡: <https://isvincent.pixnet.net/blog/post/31094121-excel-%E8%A8%88%E7%AE%97%E5%8E%BB%E6%A5%B5%E5%80%BC%E7%9A%84%E5%B9%B3%E5%9D%87>
- 阿凱的 Excel. (2017 年 11 月 13 日). 剔除兩側極值求平均. 擷取自 每日頭條: <https://kknews.cc/zh-tw/code/eza5qgn.html>