OLED 三軸加速度計計步器實作

林士平
國立清華大學(新竹)
電機工程學系
新竹,中華民國 ROC
tommy787576@gmail.com

黃楠峻
國立清華大學(新竹)
電機工程學系
新竹,中華民國 ROC
h980703@kimo.com

摘要 — 隨著健康意識抬頭，計步器逐漸進入了我們的生活，步數更成為評判一個人每日運動量的指標，更有許多的健康相關指標奠基在步數之上，例如：消耗熱量計算、步行距離等等，由此可見準確計步的重要性。我們將實作一個手腕計步裝置，整個研究可以分為四個部分：首先是決定計步器所要使用的硬體設備與系統架構，接著要收取走路、跑步、上樓梯與下樓梯的訊號，再來利用 MATLAB 做訊號分析與演算法嘗試，成功後實現於計步器系統上，並且測試精準度。演算法嘗試花了我們最多時間，最後採用峰值檢測與設定閾值兩個條件來判定步數，這兩個方法解決了偽波峰和偽波谷的問題，並且讓計步裝置在跑步、上樓梯與下樓梯可以達到 95%以上的計步精準度，走路亦有接近 90%的水準。未來這個研究可以朝增進走路計步精準度、解決非走路劇烈晃動卻會判定為走路的問題，以及步態檢測的方向前進。

關鍵字 — 數位訊號處理、計步系統實作、三軸加速度計、峰值檢測、計步演算法測試與實作、偽波峰、偽波谷、閾值

I. 介紹

A. 研究背景

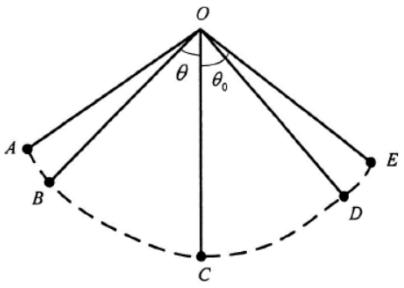
大約在 1960 年隨著健康意識抬頭，一天一萬步的說法被提出，計步器正式在日本上市。此時的計步器為機械式計步器，利用金屬球來回滑動感應手臂或腰部的抖動。當金屬球加速度大於運動門限（threshold）時則可以計步。由於運動門限不可調，機械式計步器很容易產生誤差，例如：走路稍微慢一點時，手臂甩動幅度較小則不會計步，因此電子計步器逐漸取代機械式計步器。

電子計步器通常內建三軸加速度感測器（Accelerometer）和微控制器（MCU, microcontroller），能感測使用者在 x, y, z 三軸方向上的加速度變化，並且內建複雜的演算法，因此計步可以較為準確。

計步器最基本的功能為統計步數，不管是走路、上下樓梯、跑步，不過隨著科技的發展，越來越多的功能集結在計步器上，形成各式各樣的穿戴式健康管理裝置，例如：運動手環、手機健康管理應用程式等。計步器已經逐漸成為我們生活的一部分，步數也成為每日運動量的參考，甚者許多運動健康指標例如：消耗熱量、運動速度、運動距離等，都是以計步為基礎推算而來，由此可見準確計步的重要性。

Arduino 是一個開放原始碼的開源硬體產品，基於 AVR 單晶片的微控制器，有豐富的介面、數字 I/O 口、模擬 I/O 口，同時支援 SPI, I^2C, UART 串列埠通訊，能用來接受從各類開關或感測器的輸入，並且能控制各種燈光、馬達和其他物理輸出裝置，藉此來開發各式各樣的專案。Arduino 的優勢是便捷靈活，自由度極大，可拓展效能非常高，因此受大廣大創客（Maker）喜愛。

本研究的目的是利用 Arduino 平台結合三軸加速度感測器和 OLED 顯示螢幕來實現手腕式電子計步器，並藉由 Matlab 訊號分析來輔助演算法的設計與測試。期望在嘗試各種不同演算法後可以達到精準計步的目標。



↑圖 1. 手臂擺動模型

B. 相關概念與研究

人體行走中，手臂的擺動可以看成鐘擺運動。每走一步手臂出現最高點、最低點、最高點的擺動過程(即圖 1.中的 A 到 E 或 E 到 A)，由物理推算可以得到下表(表 1.)：

表 1. 手臂擺動一個周期的加速度變化

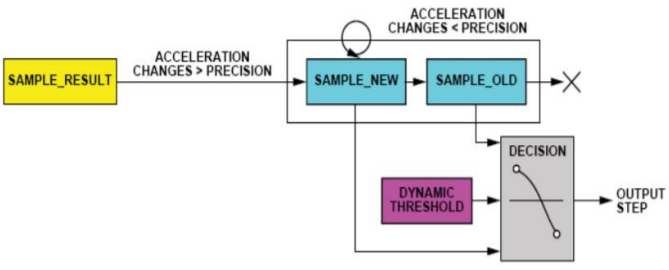
手臂擺動	A→B	B→C	C→D	D→E
切線加速度	減小	減小	增大	增大
法線加速度	增大	增大	減小	減小
合加速度	減小	增大	減小	增大

註：本表節錄自參考資料[1]。

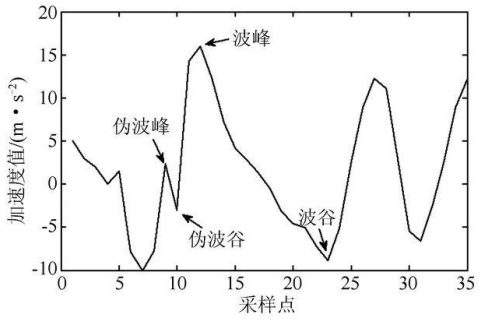
由此可見，人每走一步合加速度大致是一個正弦波型。後面我們實測出來的合加速度波型也符合此結果，這會是用來判斷一步的重要條件。

接著是相關計步演算法。參考資料[2]中的計步演算法大致上如圖 2.所示。此演算法採取動態閾值與動態精度，當加速度變化量大於精度時將加速度存入移位暫存器中，並且判斷原本在移位暫存器中的加速度(SAMPLE_OLD)和新存入的加速度(SAMPLE_NEW)是否有符合此關係式(式 1) ，如果符合便代表一步：

$$\text{SAMPLE\_OLD} > \text{DYNAMIC THRESHOLD} > \text{SAMPLE\_NEW} \quad (1)$$



↑圖 2. 動態閾值與動態精度



↑圖 3. 偽波峰和偽波谷

不過此種方法較不適合應用在手腕式計步器上，原因是手腕計步器合加速度的波型較為複雜，雜訊較多，此演算法會比較適合使用於放置在口袋或掛置於腰部的計步器上。

因此本次研究會採用參考資料[1]中的演算法來製作計步器，它專門為手腕計步器設計，因為走路合加速度波型會類似正弦波故以峰值檢測為基礎，並能解決偽波峰和偽波谷的問題(圖 3.)。論文中提到在跑步、走路、上下樓梯精確度可達 95%以上，我們在最後會測試是否真如他所說的，並且會和其他的計步裝置做精準度比較測試。

II. 研究方法與實作細節

研究方法分為四個部分：首先決定計步器與收資料的硬體，包括主控板、三軸加速度感測器、OLED 和藍芽模組；第二步是收集訊號，包括走路、跑步、上樓梯與下樓梯；第三步是利用 MATLAB 針對收到的訊號進行分析，並模擬演算法；最後一步將演算法在硬體上實現。以下為四個部分的細節：

A. 計步器硬體介紹與系統架構圖

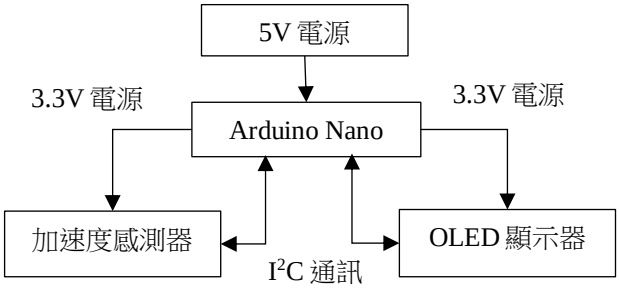
主控板採用 Arduino Nano，使用它的最大原因是體積小，長度*寬度為 4.5cm*1.8cm，適合用來開發手腕計步器，而且它有 14 個數位 I/O 引腳、支援 I²C 通信(A4:SDA 和 A5:SCL)與串口通訊(0:RX 和 1:TX)，並且可以輸出 3.3V 電壓供感測器、OLED 和藍芽模組使用。

感測器使用由 Adafruit 開發的 lsm9ds0 九軸感測器，裏頭包含加速度計、陀螺儀和磁力計，在本研究中我們只會使用到加速度計。此感測器加速度範圍選擇大，可選擇 2G、4G、6G、8G 和 16G，在此我們選擇 2G 因為對於正常的行走、跑步、上下樓梯 2G 即足夠使用，且靈敏度會比較高。人體運動時身體的振動頻率範圍在 0~20Hz，因此感測器的取樣頻率採用 50Hz，並且經過實測這個感測器取樣頻率絕對可以達到 50Hz。

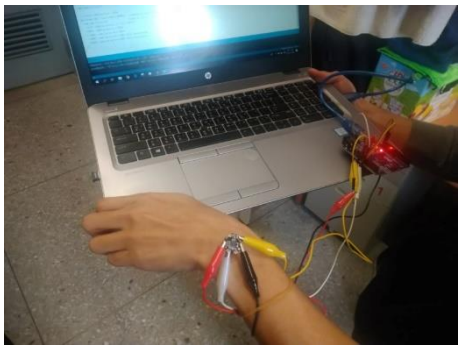
OLED 使用 Adafruit 開發的 Monochrome 128x32 I2C OLED graphic display，驅動晶片為 SSD1306，非常普及，因此網路資源多，且輸入電壓為 3.3V，低功耗適合穿戴式裝置開發。

最後是 HC05 藍芽模組，使用原因亦是非常普遍，且便於使用，其輸入電壓亦為 3.3V。

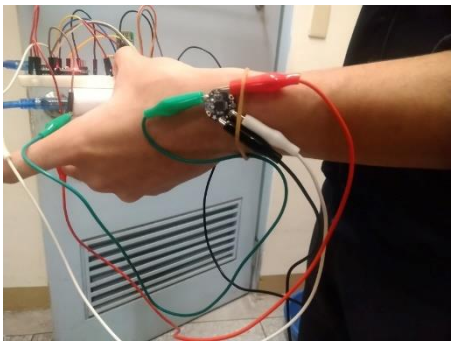
圖 4.為計步器的系統架構圖。



↑圖 4. 計步器系統架構圖



↑圖 5. 使用串口通訊(UART)採集數據

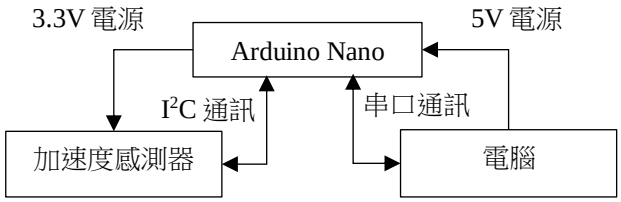


↑圖 6. 使用藍芽傳送來採集數據

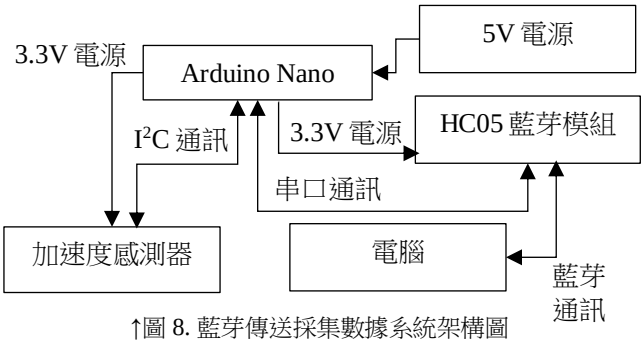
B. 數據收集場景與系統架構圖

一開始我們採用 UART 通訊來傳送感測器數據到電腦(圖 5.)，好處是連接方式簡單，且 Arduino IDE 附有序列埠監控視窗(Serial Monitor)，只要注意板子和序列埠監控視窗的鮑率一致便可以收到數據，不會有數據缺失的問題。然而缺點是收集訊號時必須兩個人走在一起：一個人拿著電腦，一個人戴著感測器走路，非常不方便且會影響到收訊品質，增加雜訊。

因此我們使用第二種方法：使用 HC05 模組將感測器資料利用藍芽傳輸至電腦(圖 6.)，好處是收到的訊號會較接近實際走路的狀況，且可以進一步收取跑步、上樓梯與下樓梯的訊號，不過缺點是藍芽有時會連線不穩造成數據缺失，且找尋適當的終端機(Terminal)花了我們不少時間，因為有些終端機收訊號的速度過慢，甚至會不停翻新訊號，最後使用 PuTTY 才解決這些問題。圖 7.和圖 8.為兩種方式的系統架構圖：



↑圖 7. 串口通訊採集數據系統架構圖



↑圖 8. 藍芽傳送採集數據系統架構圖

C. MATLAB 訊號分析

我們選擇將三軸加速度合一的原因是單看一軸的加速度訊號無法表示真正的運動特徵，而且由圖 9.和圖 10.與前面的手臂擺動模型探討可知合加速度會有類似正弦波的波型，後面的演算法會以此為基礎來做計步判定。

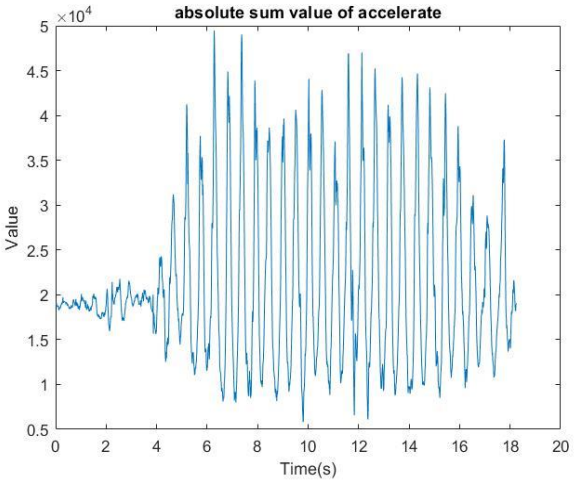
關於三軸加速度合一我們想到兩種計算方式，第一種是將三軸加速度取絕對值相加(式 2)：

$$abs\_a = abs(ax) + abs(ay) + abs(az) \quad (2)$$

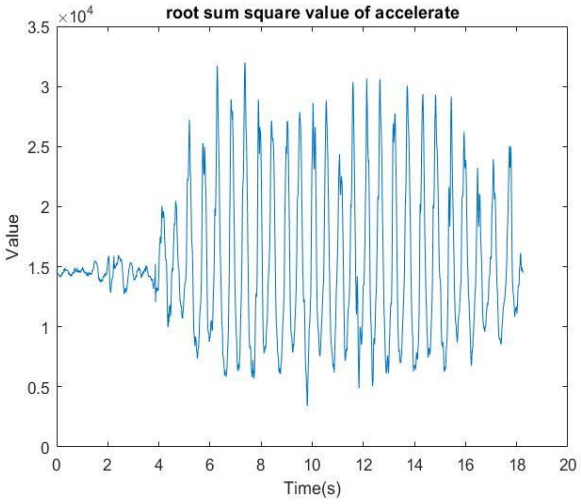
第二種是三軸加速度平方和開根號(式 3)：

$$a = \sqrt{ax^2 + ay^2 + az^2} \quad (3)$$

圖 9.和圖 10.是走路的三軸加速度訊號分別以絕對值相加和平方和開根號後於時域的波型圖。可以發現絕對值相加後的訊號變化範圍較平方和開根號大，但兩者變化趨勢大致相同。由於後面的計步演算法關心的是訊號波型與相對數值，且平方和開根號計算較絕對值相加複雜，硬體成本較高，所以我們選擇使用絕對值相加的方式來將三軸加速度合一。



↑圖 9. 使用絕對值相加三軸合一



↑圖 10. 使用平方和開根號

D. MATLAB 演算法模擬

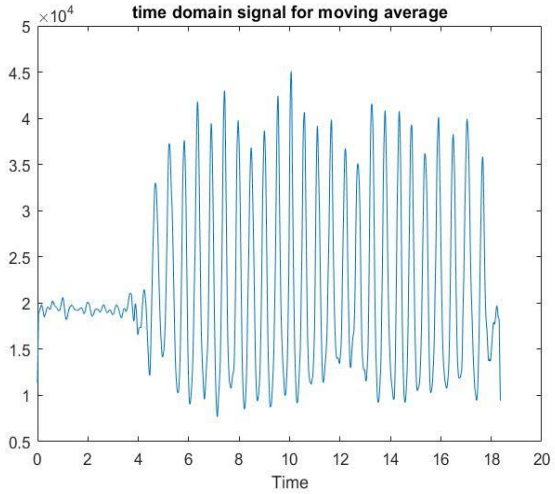
1) 決定濾波器：

為了使訊號中的尖刺消除使之平滑，需要將訊號進行濾波。我們考慮兩種濾波方法：移動平均濾波(Moving average)和三角窗濾波(Triangular window)。圖 11.和圖 12.是走路的三軸合一訊號以移動平均濾波和三角窗濾波後於時域的波型圖。

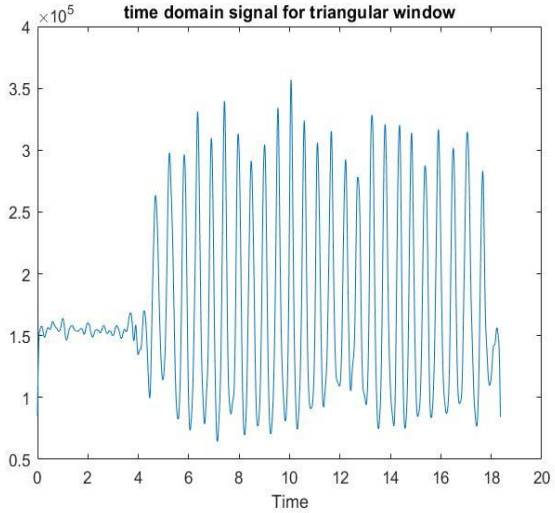
移動平均濾波時常用來做訊號平滑化，且可實時(Real time)實現，自然是我們第一個想到的方式，但是由於移動平均使用的矩形窗(Rectangular window)在時域擷取時的兩端突變較大，在邊緣的數值經濾波後可能會有突波訊號的產生，因此我們決定使用三角窗濾波。

三角窗濾波將窗口內的數值加權，離窗口中心越遠加權越小，由圖 11.和圖 12.的結果可觀察到雖然濾波後兩種方法相差不大且趨勢也大致相同，但此種濾波方式較可以避免因濾波而產生的突波，因此我們選擇使用三角窗濾波的方式。

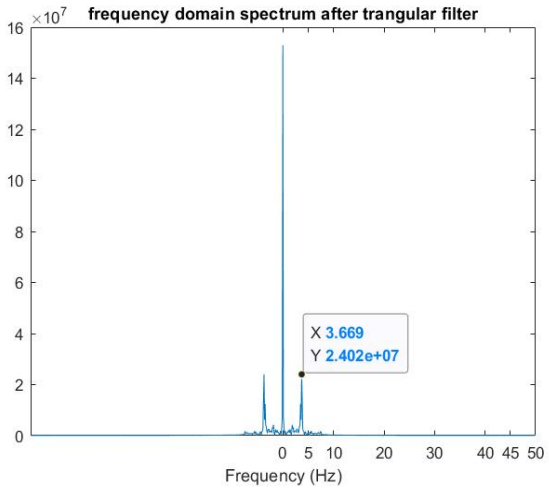
下頁圖 13.為走路的三軸加速度合一訊號經三角窗濾波後的頻譜，在 0Hz 的脈衝波(impulse)來自於三軸加速度合一訊號的 offset，將其平均值減掉便可去除，這個動作相當於將訊號沿著縱坐標軸做平移，但在此沒有必要因為後面的演算法關心的是訊號的波型與相對數值。



↑圖 11. 走路的三軸合一訊號以移動平均濾波



↑圖 12. 走路的三軸合一訊號以三角窗濾波



↑圖 13. 走路的三軸合一訊號以三角窗濾波

而在圖 13.中 3.67Hz 的脈衝波(impulse)則是我們想要得到的計步頻率，經由三角窗濾波後可以濾除高頻雜訊，凸顯我們需要的訊號。

2) 步伐判定的條件：

觀察上頁的圖 12.可以知道判斷是否走一步有兩個條件：第一個條件是要出現一對波峰和波谷，然而只有第一個條件還不夠，因為無法確認這對波峰和波谷會不會是偽波峰和偽波谷，所以還要有第二個條件。第二個條件是區域極大值減去區域極小值 > 閾值(threshold)，區域極大值和極小值是指從上次計步後開始不斷找尋極值，並且在下次計步後將極值回復原始值，再繼續尋找。必須符合以上兩個條件才能算一步。

3) 找尋峰谷值的方式：

我們接著探討如何找尋峰谷值，因為此計步演算法如同前面提到的，會以峰值檢測為基礎。找波峰和波谷的方法如下：

```
peak←0
valley←0
for all a[k] in the buffer do
    b[k]←a[k+1]-a[k]
end for
for all b[k] in the buffer do
    if b[0]>=0 AND b[1]>=0 AND b[2]>=0 AND b[3]<0 AND b[4]<0 AND b[5]<0
        peak←1
    else
    if b[0]<0 AND b[1]<0 AND b[2]<0 AND b[3]>=0 AND b[4]>=0 AND b[5]>=0
        valley←1
    end if
end for
```

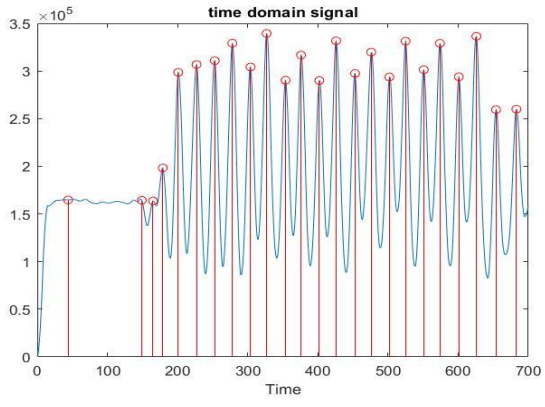
(以上虛擬碼(pseudo code)節錄自參考資料[1])

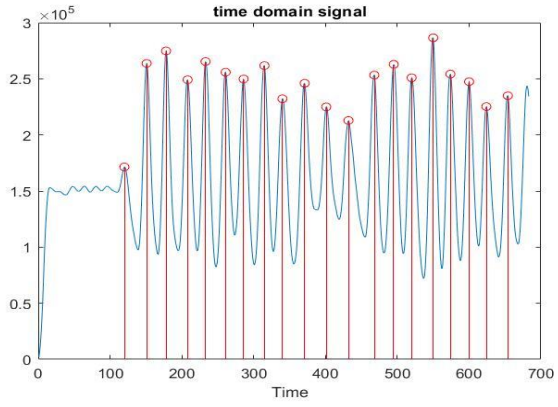
我們使用一個容量為 7 的緩衝器(buffer)儲存濾波後的訊號，並且通過差值濾波器(difference filter)得到 6 個差值，當前 3 個差值均 >= 0 且後 3 個差值 < 0 表示找到波峰，反之則為波谷。

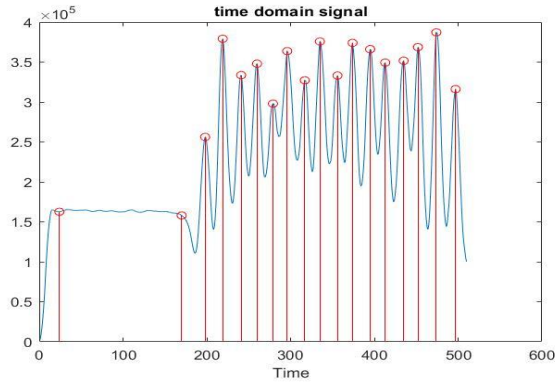
緩衝器的長度若設定太小可能會將雜訊突波誤判為波峰或波谷，設定過大則會太不靈敏導致一些波峰波谷無法被找到。經過模擬與之後的 Arduino 實測得到的最佳長度為 7。

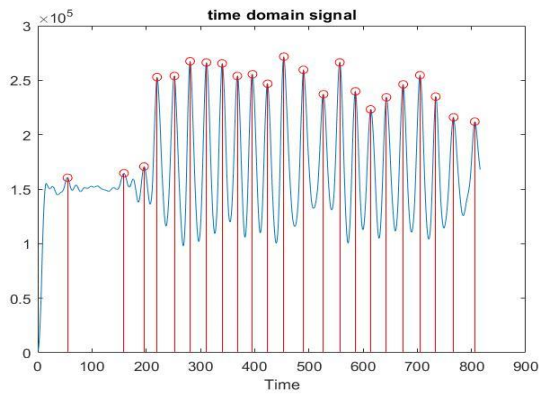
4) 演算法模擬結果：

圖 14.到圖 17.為演算法在 MATLAB 上模擬得到的結果，紅色的線代表判定為一步，從上一頁的模擬結果可發現在步行開始前會有被誤判的步數存在，但步行開始後計步都非常準確。

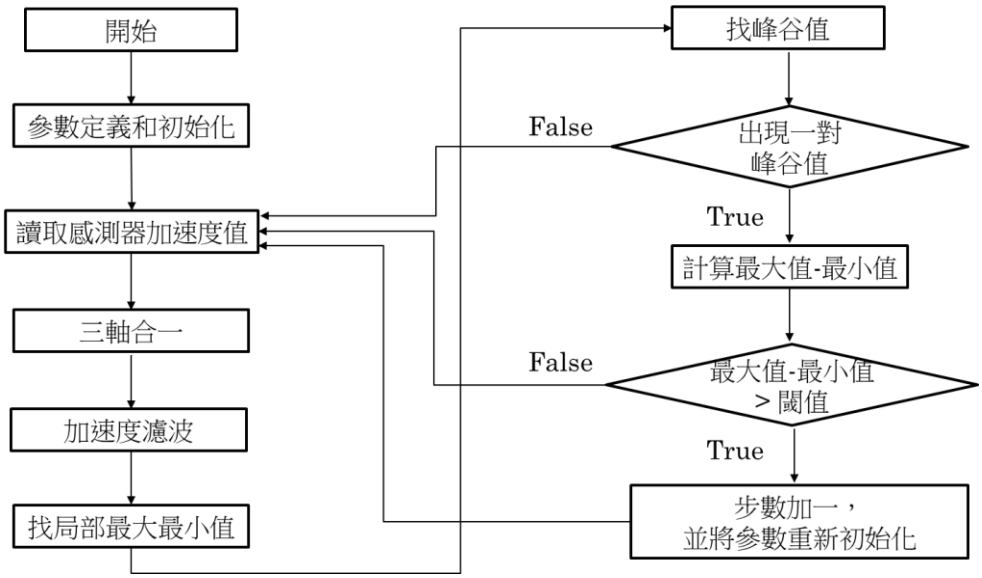








↑左上圖(圖 14.)為走路，實際 20 步，模擬得到 24 步；↑右上圖(圖 16.)為下樓梯，實際 20 步，模擬得到 20 步；
↑左下圖(圖 15.)為跑步，實際 15 步，模擬得到 18 步；↑右下圖(圖 17.)為上樓梯，實際 20 步，模擬得到 23 步；



↑圖 18.計步演算法流程圖

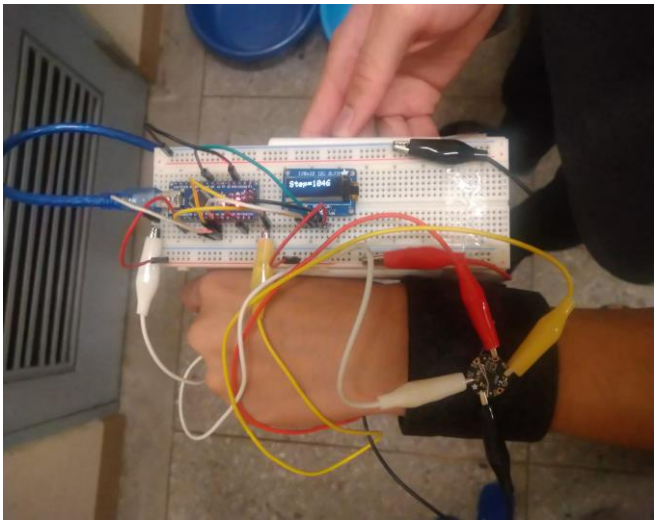
5) 演算法流程圖：

最終我們的計步演算法流程圖如圖 18.所示。首先初始各個參數值然後讀取三軸加速度並三軸合一，接著濾波完後找尋訊號的局部最大值與最小值，並且做峰值檢測找尋波峰和波谷，當出現一對波峰和波谷時，判斷局部最大值減局部最小值是否大於閾值，如果是則步數加一並將參數初始化，否則繼續讀取三軸加速度值，重新整個流程。整個演算法實時(Real time)處理，可以即時顯示步數更新的情況。

E. Arduino 硬體計步器實現

最後我們將該演算法打成 Arduino 的語言實現在如圖 4.架構的硬體上，整個裝置如下圖 19.所示。

實現在硬體上又和模擬不同，會有更多需要考慮的狀況，例如資料的傳輸速率、電源供應的穩定與否、硬體資源(如主控板的記憶體)是否足夠……等，並非模擬通過就百分之百可以實現在硬體上，前面我們有許多次的嘗試是在 MATLAB 模擬均沒有問題但實現在硬體上卻會無法正常計步。另外硬體實現還有另一個麻煩就是除錯(debug)較困難，Arduino 不像 MATLAB 或 PYTHON 有逐步偵錯的功能，因此除錯只能土法煉鋼，將資料用序列埠監控視窗印出來，如此還滿花時間的。



↑圖 19. 最終的計步裝置

III. 研究結果

表 2.到表 5.(本頁與下一頁)為計步精準度測試結果，表 6.為走路、跑步、上樓梯與下樓梯的總精準度的整理表。我們將計步器配戴於手腕，針對走路、跑步、上樓梯與下樓梯各測試了 9 次，其中有 3 次為 15 步、3 次為 30 步、3 次為 60 步，同時我們使用兩種手機應用程式做比較，表格中 A 裝置為我們的計步裝置、B 裝置為手機應用程式(計步器：免費計步器&卡路里追蹤工具)、C 裝置為手機應用程式(StepsApp 計步器)。另外計步精準度的定義如下(式 4) ：

$$\text{計步精準度} = \left(1 - \left|\frac{\text{實際步數} - \text{實測步數}}{\text{實際步數}}\right|\right) * 100\% \quad (4)$$

由表 6.可知我們的計步器(裝置 A)在跑步、上樓梯與下樓梯測試中精準度均可達 95%以上，走路亦有接近 90%的水準，由表 4.和表 5.可進一步得知在 15 步和 30 步上樓梯與下樓梯表現甚至比手機應用程式更好。另外我們的計步器在走路的情況下是最不準確的，這個問題也影響了上下樓梯的精準度，上下樓梯的精準度隨步數上升而下降，我們認為原因是樓梯和樓梯間會有平面需要走個 1,2 步，隨著步數增加，需要走的平面部分就會增加，推測因此影響了上下樓梯的精準度，不然爬樓梯的過程是非常準確的。最後是跑步，可以發現跑步的步數越少會越不準確，我們認為可能是因為在跑步初始與結束計步會有不準確的問題，如圖 15.所示。

表 2. 計步精準度測試(走路)

	裝置	15 步	30 步	60 步
走路	裝置 A 實測結果	15	30	57
		14	26	47
		12	28	49
	精準度	91.1%	93.3%	85.0%
	裝置 B 實測結果	14	30	59
		15	25	61
		14	31	59
	精準度	95.6%	95.6%	99.4%
	裝置 C 實測結果	15	30	60
		15	30	58
		15	29	58
	精準度	100%	98.9%	97.8%

表 3. 計步精準度測試(跑步)				
	裝置	15 步	30 步	60 步
跑步	裝置 A 實測結果	18	32	59
		16	32	58
		17	31	59
	精準度	86.7%	94.4%	97.8%
	裝置 B 實測結果	15	30	58
		15	32	60
		10	29	59
	精準度	88.9%	98.9%	98.3%
	裝置 C 實測結果	12	31	58
		15	33	55
		15	30	60
	精準度	93.3%	95.6%	96.1%

表 4. 計步精準度測試(上樓梯)				
	裝置	15 步	30 步	60 步
上樓梯	裝置 A 實測結果	16	30	57
		14	31	60
		14	30	59
	精準度	97.8%	98.9%	97.8%
	裝置 B 實測結果	14	29	61
		16	30	61
		13	33	58
	精準度	95.6%	97.8%	100%
	裝置 C 實測結果	13	31	59
		12	30	61
		14	31	60
	精準度	86.7%	97.8%	100%

表 5. 計步精準度測試(下樓梯)				
	裝置	15 步	30 步	60 步
下樓梯	裝置 A 實測結果	17	29	58
		14	30	56
		15	27	56
	精準度	97.8%	95.6%	94.4%
	裝置 B 實測結果	14	30	60
		14	30	58
		14	34	61
	精準度	93.3%	95.6%	99.4%
	裝置 C 實測結果	12	28	61
		14	29	57
		14	30	57
	精準度	88.9%	96.7%	97.2%

表 6. 總體計步精準度比較				
裝置	走路	跑步	上樓梯	下樓梯
A	88.2%	97.8%	98.7%	95.8%
B	97.8%	97.8%	100%	100%
C	98.4%	98.1%	98.7%	95.9%
註：本表將表 2.到表 5.走路、跑步、上樓梯與下樓梯不同步數的情況合在一起算計步精準度。				

IV. 結論

本次研究實際進行了一次計步器實作流程：從硬體的選擇與系統架構、收集訊號，到 MATLAB 訊號分析、MATLAB 演算法模擬，最後實現在計步器系統上。其中最重要的計步演算法採用峰值檢測搭配閾值設定，經過我們實測在跑步、上樓梯與下樓梯測試中精準度均可達 95%以上，走路亦有接近 90%的水準。

整個計步判定最大的兩個問題為怎麼樣算一步，與如何解決偽波峰和偽波谷的問題。因為經過物理理論與實際收訊號分析可以發現人走路的合加速度波型類似正弦波，所以當出現一對波峰波谷可以初步判定為一步，但波峰波谷亦有可能是偽波峰和偽波谷，所以研究中使用兩個方式來解決：首先是尋找波峰和波谷的緩衝器長度設置為 7，避免將雜訊突波誤判為波峰或波谷；第二是閾值設定，當找到波峰和波谷後必須確定區域極大值-區域極小值超過閾值，才能判定為一步。

接著是未來可以改善與增進的目標。在這邊要非常感謝四位評審老師給予我們的作品諸多重要意見：李祈均老師提出我們的計步器無法分別走路與非走路的強烈晃動，會是未來我們的努力目標，或許可以在演算法中加入時窗來放棄兩次過近的計步以除去非走路的計步。黃朝宗老師提出解決偽波峰和偽波谷可以使用 wavelet 濾波器，如果未來採用此濾波器或許可以解決走路計步較為不準的問題。劉奕汶老師提出未來此系統可以朝步態檢測的方向發展，我們覺得非常有趣，未來或許只要帶著此計步裝置就能判斷是誰在走路，或是正在做什麼運動。李夢麟老師提出三角濾波器其實就是做兩次平均濾波，聽到這個概念後我們豁然開朗，難怪三角濾波器可以讓波型變得更加平滑，其實它等效上就是用平均濾波濾兩次。

綜合上述整個研究未來可以朝幾個大方向前進：1.解決走路較為不精準的問題；2.解決非走路卻會計步的問題；3.朝步態檢測方向前進。非常感謝教授們的意見給予這個專題更多的可能。

V. 參考資料

[1] 步數檢測方法及在手腕式計步器中的應用研究。謝茹花。蘭州交通大學。

[2] 利用三軸數字加速度計 實現功能全面的計步器設計。Neil Zhao。