

摘要

作品名稱：無人機運送 AED

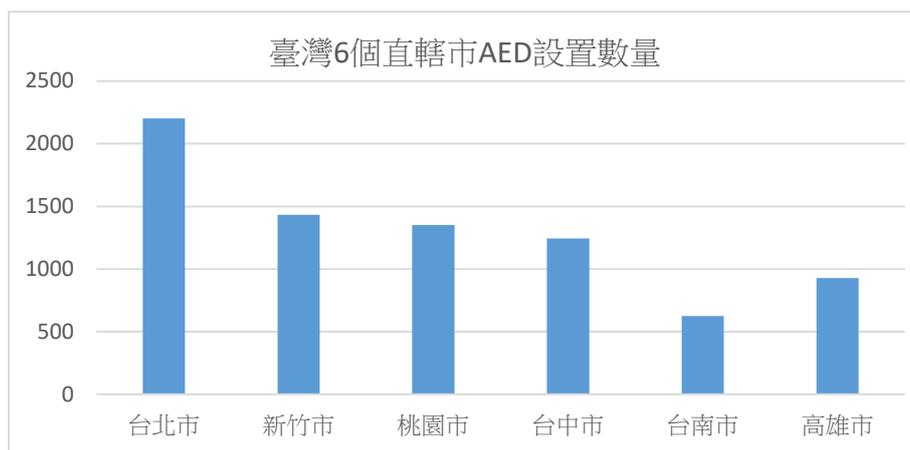
在台灣最近幾年來，因突發性的心肌梗塞而過世的人日漸增加。而且當疾病發生時，就算在一發生的時候叫救護車也不一定來得及，所以我們想出了一個辦法，在事件一發生時，旁邊路人拿起手機，使用我們製作的 APP，發送 GPS 信號到 ThingSpeak 資料庫進行存取，由無人機上的 Arduino WeMos D1 先連上網路，再去抓 ThingSpeak 資料庫裡的 GPS 資料，之後運用 Mavlink 協定將 GPS 資料寫成飛行計畫，傳到 Pixhawk，再讓無人機啟動並執行飛行計畫，無人機就會前往目的地救人，當 AED 使用完後放到無人機上並按下 APP 的按鈕，無人機就會執行返航的飛行計畫，飛回到無人機原本的所在地。

資電領域優等(1)

一、前言

某天健康教育老師在課堂上說，發生心肌梗塞的人在心臟停止 5 分鐘就會發生生命危險，接著播放了「史上最急迫！黃金 5 分鐘救命任務！」的影片，影片內容是有三個場景，每個場景都有一位在跑步方面很有成就的人與即將發生心肌梗塞的長輩，但是那些頂尖田徑國手都來不及在五分鐘內取得 AED 並使用成功，這代表 AED 架設的不普遍。

根據衛福部公共場所 AED 急救資訊網 2021 的 AED 地圖，台灣公共場所 AED 設置數量為 11071 個，由下圖表可知，6 大縣市設置 AED 的比例有點不太平均，最繁華的台北擁有最高密度的 AED，越往南的其他縣市比例卻下降許多，由此可知，還有許多地方 AED 的設置數量不足。



資料來源：衛福部公共場所 AED 急救資訊網 AED 登錄數

在台灣裡，每年因突然的心律不正常而死亡的人數有超過 6000 人，這數字是多麼驚人，叫了救護車一定會延誤救援時間，畢竟路上的狀況也是千變萬化的，我們轉而思考如果地面上會有交通問題的話，那就上天空吧，這樣在 AED 被需要的時候，救護車又無法及時到達，人也不用慌慌張張的到處尋找 AED，而是利用手機，讓 AED 直接送到使用者的附近。這不但減少了尋找的時間成本，也大大提升了患者的生存率。

二、研究方法(過程)

(一) 取得手機 GPS 定位的資料並存取至資料庫

APP Inventor2 的位置感測器變化就是隨著手機是否有移動而改變。

資電領域優等(1)

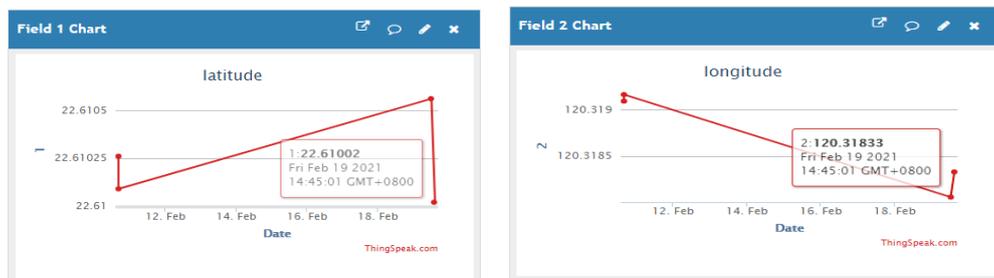
開啟程式時，會先偵測 GPS 供應商的名稱來判斷使用者是否有開啟”位置”，如果供應商名稱不為”GPS”，就會提醒使用者開啟，之後當位置感測器偵測到位置變化時，地圖顯示自身所在位置的經緯度並標記。

如果標記未出現時，卻按了傳送則會顯示”失敗!!”並且不會傳遞資料。

地圖上出現標記後，按下傳送，程式會將 ThingSpeak API Requests 裡的 Write a Channel Feed 網址跟 GPS 參數放入，資料就會傳到 ThingSpeak 資料庫。但這些定位系統不是那麼的精確，所以造成了目的地的座標偏離的狀況，所以必須在周遭持續走動，當地圖上的標記與實際相同時，再發送 GPS 信號啟動飛行，以免送錯地方。



(二) 用 Arduino WeMos D1 連上 Wi-Fi 並抓 ThingSpeak 資料庫資料



1. 程式說明

先安裝 Wi-Fi 開發版跟程式庫，使用 WeMos D1 (Retired) 開發版，將 Wi-Fi 的名稱跟密碼先放入程式，並將 ThingSpeak 資料庫的 ID 跟要抓取的欄位編號寫到程式裡，執行後，程式會先試著連上網路，當連上網路後，他會直接連到 ThingSpeak 資料庫並抓取欄位編號的資料。

```
WiFi.begin("OPPO A73","asdfghjkl");//WiFi.begin(wifi 名稱, wifi 密碼)
//抓取資料庫數據
Serial.println((ThingSpeak.readFloatField(1292840,1)),5);//資料一
Serial.println((ThingSpeak.readFloatField(1292840,2)),5);//資料二
```

資電領域優等(1)

(三) 用 Arduino 上傳飛行計畫至 Pixhawk 並執行任務

1. Arduino 用 Mavlink 協定的程式上傳飛行計畫

(1) 程式說明

安裝 Mavlink 程式庫，先設立一組軟串口，讓 Arduino 與 Pixhawk 連接，程式裡 Mission_count() 函數將我們指定的資料數打包，並依照打入的系統代號發送到 Pixhawk 上，之後由 MavLink_receive() 函數來將資料跟我們要寫入的飛行計畫處理並一一上傳，當上傳完指定的資料包後就會回傳值，並告訴我們上傳成功。

要創建飛行計畫內的指令，只要將 command 改成想要設定的指令名稱跟輸入那個指令所需的參數值即可。例：要設一個航點 (waypoint)，command 要設成 MAV_CMD_NAV_WAYPOINT，寫入航點的座標 (x, y, z)，最後將函數寫到 MavLink_receive() 函數裡。

```
uint16_t command = MAV_CMD_NAV_WAYPOINT; //要進行的動作
float param1 = 0;
float param2 = 0;
float param3 = 0;
float param4 = 0;
float x = 22.12345; //緯度
float y = 120.12345; //經度
float z = 5; //高度
```

2. Arduino 讓無人機執行任務

(1) Arduino 設置無人機的模式

用 Arduino 設定 Pixhawk 的飛行模式，只需將找 Mavlink 函式庫裡的 set_mode 裡所需的變數，再寫入程式並寫成封包，之後執行程式，Arduino 就會將其傳入飛控板裡，無人機就會依照參數改變成對應的模式。

(2) 執行飛行計畫

執行任務，Arduino 只要先解鎖無人機，之後將模式設為 AUTO 模式，再執行 MISSION_START，無人機就能依照飛行計畫去執行任務。要注意的是解鎖無人機只能在 Stabilize、Loiter 等特定模式下才能解鎖，所以在解鎖無人機前最好將模式先設模式為可以解鎖的模式。

(四) Arduino 偵測 AED 是否使用完並自動返航

1. Arduino 用壓重模組 HX711 偵測 AED 重量

(1) 壓重模組程式

將函式庫裡 HX711 程式庫安裝，使用前要先用校準重量精準的砝碼放

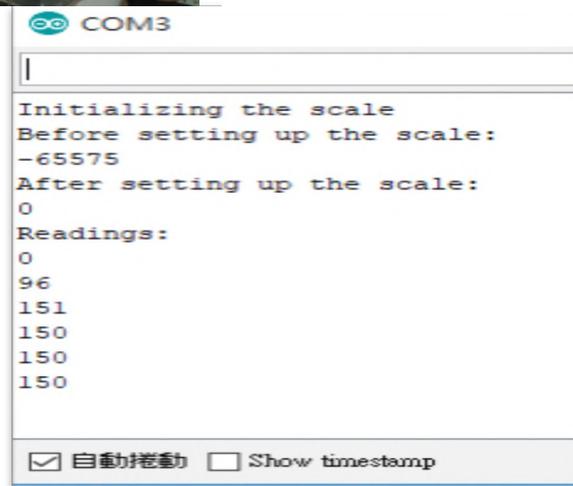
資電領域優等(1)

上壓重機上並在校準程式裡輸入砝碼的重量（公克），取得比例參數。

再將校準得到的比例參數放入壓重程式，就可以秤出物品的重量了，另外，我們裝上了LED燈，以辨識放上去的東西有達到一定重量。



```
const int sample_weight = 150;
if (scale.get_units(10) >= 100)
{
  digitalWrite(D7,HIGH);
  digitalWrite(D8,LOW);
}else{
  digitalWrite(D8,HIGH);
  digitalWrite(D7,LOW);
}
```



(2) APP Inventor2 歸還 AED 程式說明

當”傳送”按下後，將會切換到這，而資料也同時傳出到 ThingSpeak 資料庫，當 AED 被送到目的地之後，使用完後將 AED 放回原處等到壓重模組的綠燈亮起並按下”歸還 AED”後，會再把資料傳給 ThingSpeak，讓 Arduino 抓取，讓它知道無人機已經準備好可以返航。

雖然 APP Inventor 2 有內建的資料庫，但是無法顯示多個 GPS 信號，且會有多餘的資訊，怕 Arduino 抓錯資料。

3. Arduino 上傳返航的飛行計畫並執行

Arduino 先將預先寫好的返航飛行計畫上傳至 Pixhawk，之後將無人機解鎖並設無人機模式為 AUTO，再執行 MISSION_START，無人機就會執行返航的飛行計畫返回 Home 點。

(四) 軟體程式結合說明

1. 當有人用 APP 傳 GPS 資料時

Arduino 先確認壓重有到一定重量並從 Thingspeak 資料庫的第三個欄位確

資電領域優等(1)

認執行的數值，如果都滿足條件，就會開始抓取資料庫的第一個欄位跟第二個欄位的 GPS 資料，當確認都有抓到時，就會將 GPS 資料寫進 create_waypoint () 函數裡，並建立一個飛行計畫寫進 Pixhawk 裡，當飛行計畫上傳完了，Arduino 就會解鎖無人機並執行任務。

2. 當用 APP 按歸還 AED 時

Arduino 會先確認壓重有到一定重量並從 Thingspeak 資料庫的第三個欄位確認歸還的數值，如果條件都滿足，就會建立一個返航的飛行計畫並將其寫入 Pixhawk 裡，上傳完飛行計畫後，Arduino 就會解鎖無人機並執行返航。

三、研究結果

在本次研究中，我們設定數據傳輸器讓 Pixhawk 與 Mission Planner 互相連接，只要從 Mission Planner 規劃飛行計畫就可以飛行。利用 APP Inventor 編寫了一個簡易的程式，可以知道自身所處的 GPS 定位資料，並將資料傳送到 ThingSpeak 資料庫保存，透過 Arduino IDE 的編寫，讓 Arduino 板連接無線基地台網路，去抓取傳送到資料庫的 GPS 資料，進而透過 Mavlink 協定創建一個飛行計畫，讓無人機前往目標地，使用完 AED 後，透過壓重的判斷是否 AED 放上後，再傳送資料讓無人機自動返航回初始點。



四、結論

本作品中，以 Pixhawk、Mission Planner 和 WeMos D1 Wi-Fi 為控制無人機飛行的主軸。

(一) 以 Pixhawk 作為整體硬體核心的控制器，不但可以從 Mission Planner 接收到的指令轉成發送給馬達轉動的訊號，也可以將任何接收到的訊息，經由數

資電領域優等(1)

據傳輸器，以達到由 Pixhawk 控制無人機自動且穩定飛行的模式。

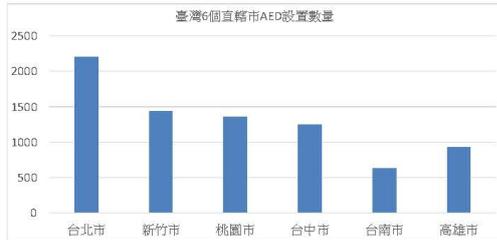
- (二) Mission Planner 負責觀看無人機的所在地(藉由 GPS 定位系統)。當無人機有警告發生時，Mission Planner 立即收到資訊，並迅速地解決當下所發生的問題，及時修改 Pixhawk 對無人機的飛行模式或飛行計劃(原地降落或立即返航)，以降低事後所造成的重大悲劇。
- (三) WeMos D1 Wi-Fi 是當環境處於收不到 GPS 訊號，但要展覽時所作出的對策，不然我們都是用 Mission Planner 來操作，畢竟我們要做的是讓無人機飛到不同目的地，而不是定點飛行。然而，用 WeMos D1 Wi-Fi 來控制有一個重大的缺點，就是無法自動穩定機身，對我們來說是一項需要花許多時間實驗，才有辦法解決的問題。
- (四)由於是我們第一次接觸無人機，所以在各個方面都需要查詢很多資料，包括組裝、設定等，另外我們所外加的 Arduino 板、RGB、無線充電及 HX711，這些都跟無人機沒有太大的關係，但我們慢慢地去組合，在所有零件不會衝突的條件下，成功讓原本互不相干的東西變成一個完整的主體。
- (五)我們製作的無人機，除了可以使用在 AED 不密集的地方，也可以在深山裡。比起等待救難隊員到達進行搶救，無人機可以更早到達且更方便，不太受地形限制。



研究動機

某天健康教育老師在課堂上說，發生心肌梗塞的人在心臟停止5分鐘就會發生生命危險，接著播放了「史上最急迫！黃金5分鐘救命任務！」的影片，影片內容是有三個場景，每個場景都有一位在跑步方面很有成就的人與即將發生心肌梗塞的長輩，雖然實際救援時間可能不到五分鐘，但是那些頂尖田徑選手都來不及在五分鐘內取得AED並使用成功，這代表AED架設的不普遍。

根據衛福部公共場所AED急救資訊網2021的AED地圖，台灣公共場所AED設置數量為11071個，由下圖表可知，6大縣市設置AED的比例有點不太平均，最繁華的台北擁有最高密度的AED，越往南的其他縣市比例卻下降許多，由此可知，還有許多地方AED的設置數量不足。



資料來源：衛福部公共場所AED急救資訊網 AED登錄數

在台灣裡，每年因突然的心律不正常而死亡的人數有超過6000人，這數字是多麼驚人，而且在這些數字裡，不只有年紀比較大的長輩，也有上班族，更何況是學生，在學校因為跑步時突然的心臟驟停而喪失性命的也不少。然而我們無法隨時注意身旁的人會發生什麼疾病，但我們也無法在金錢上有所提供。

叫了救護車一定會延誤救援時間，畢竟路上的狀況也是千變萬化的，我們轉而思考如果地面上會有交通問題的話，那就上天空吧，這樣在AED被需要時候，救護車又無法及時到達，就利用手機就可以等待AED自己送過來，人也不用慌慌張張的到處尋找AED，而是利用手機，讓AED直接送到使用者的附近。這不但減少了尋找的時間成本，也大大提升了患者的生存率。

研究目的

- 一、了解Pixhawk與Mission Planner的連接來控制無人機的飛行
- 二、研討怎麼取得手機GPS定位的資料，並傳送至資料庫
- 三、探討如何用Arduino連上行動無線基地台並讀取經緯度的資料
- 四、研究Arduino怎麼取代Mission Planner創建飛行計畫並執行
- 五、偵測AED使用完畢後，自動返回Home點



硬體組裝

- 1.組裝機臂與馬達的螺絲，四支機臂都裝上馬達之後，再將機臂與無人機的下方金屬底座用螺絲合在一起。
- 2.電變的線與馬達的線焊接，（馬達與電變的線不分正負），再把另一端電變的紅線和黑線焊接到分電盤上的正極（紅）、負極（黑）。
- 3.將與電池連接的母頭也焊上正極（紅）、負極（黑），母頭端接上電源模組，旁邊架設數據傳輸器，然後把固定電池的束帶綁在下方的金屬底座上。
- 4.接著把上方的金屬面板、減震板黏上，上面用泡棉膠把飛控板黏上，先確認GPS上的箭頭與飛控上的箭頭一致，再將GPS裝置與Arduino WeMos D1板固定在機臂上。
- 5.電子秤的部分放在飛控板的上方用螺絲固定住，然後把RGB模組用泡棉膠固定在無人機金屬底座的下方，再裝上高腳架。
- 6.無線充電接收器藉由轉接頭與鋰電池相接，放置高腳架的後端，進行無線充電。

設計架構



研究流程

