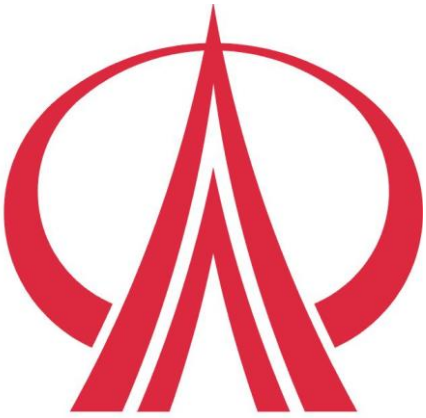
修 平 科 技 大 學
電 機 工 程 系

DEPARTMENT OF ELECTRICAL ENGINEERING

HSIUPING UNIVERSITY OF SCIENCE AND TECHNOLOGY

實務專題報告書

有 害 氣 體 偵 測 器



指 導 老 師： 程 兆 龍

專題製作學生：

四技電四乙 林哲遠 BD103009

四技電四乙 許鈞堯 BD103011

中 華 民 國 107 年 6 月 25 日

修 平 科 技 大 學

電機工程系

HSIU-PING UNIVERSITY OF SCIENCE

AND TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING

指導老師：程兆龍

專題製作學生：林哲遠、許鈞堯

製作日期：107 年 6 月 25 日

感謝

在專題製作的期間受老師及同學、學長的照顧指導，在專題製作上所給予我的寶貴意見及學業生活上的關懷，在無數熬夜過程中，幫忙解決所遇到的大大小小的困難，相互勉勵，讓此專題製作得以成功順利，特此感謝。相信這將是我一生中難忘的回憶之。最後，謹以此專題製作獻給我最愛的父母、家人以及所有關心我、幫助過我的師長與朋友們。

摘要

由於瓦斯爐擺設不良或者管線老舊或遭齧齒類動物破壞或通風不良等等因素，使我們在瓦斯外洩、還有一氧化碳因為無色無味不易察覺，所以我們設計有害氣體偵測器及警告器，當偵測到一些 氣體濃度過高時，偵測器可以即時偵測並發出警告聲，讓屋內的人得知有有害氣體過濃或外露現象，進行關閉瓦斯、離開屋內或者開啟窗戶的動作，以免發生悲劇。

我們使用的感測元件，配合市面上購得低成本支電子零件，來探討此專題。其設計可以氣體濃度過高的情況下，發出警報提醒使用者，並可以隨時掌握環境的改變以立即採取安全措施。

目錄

圖目錄

表目錄

第一章、 研究背景及目的

2014 年**高雄氣爆事**故是 2014 年 7 月 31 日 23 時 55 分以後至 8 月 1 日凌晨間，發生在臺灣高雄市前鎮區與苓雅區的多起石化氣爆炸事件。7 月 31 日約 21 時，民眾通報疑似有**瓦斯**洩漏。幾個小時後該區域發生連環爆炸，造成 32 人死亡、321 人受傷，並造成至少包括三多一、二路、凱旋三路、一心一路等多條重要道路嚴重損壞，周邊店家也因為爆炸破壞而造成重大經濟損失。

圖 1.1、高雄氣爆圖

此事件引起台灣各界的關注，這場災難算是人員的失疏，在尚未氣爆前，管線疑似有外洩的情況，從報案開始到氣爆發生，整整有三個多小時，第一時間災區檢測人員完全無法辨識，而誤估了嚴重性，被當成小規模的洩漏事件，因導致後來的氣爆事件。

有此知道提早預防與偵測的重要性，如果在氣爆前有提早偵測到有害氣體濃度過高，或者知道是何種有害氣體，就可以先防範以及做出應對的方案，把現場災害降到最低，最少也可以先把人員撤離災區，先達到零死傷的效果，後續災害再做處理。

因此，我們才會針對此項事件，來歸擬專題，對於有害氣體等，由網路收集相關資訊、電路等，來做更好的對策，加以偵測預防。

圖 1.2、高雄氣爆圖

第二章、　內容探討

一、文獻探討

我們有參考學長姐的資料，也上網收尋所需的資料，有找一些偵測器像 MQ-135 偵測範圍:氨(NH3):10ppm ~ 300ppm 、苯: 10ppm ~ 1000ppm、酒精 : 10ppm ~ 300ppm 、MQ-5 偵測範圍：200~10000ppm 天然氣感應器（液化石油氣、液化天然氣、異丁烷、丙烷、煤氣)、MQ-2 偵測範圍:300~10000ppm(液化氧、丁烷、丙烷、甲烷、酒精、氫氣）、 MQ-7 偵測範圍：10~1000ppm（一氧化碳）

二、專題內容

1、 氣體濃度感測器的應用

在日常生活中，我們常見的氣體感測器，幾乎就是氣體濃度感測器，如瓦斯濃度、酒精濃度、一氧化碳濃度、以及空氣中的濕度。最近工業上的安全日趨重視，使的各種氣體濃度感測器被廣泛的應用，特別是在化學工廠更加的注重。

另外的交工安全臨檢的酒精濃度偵測器、一般家庭的瓦斯洩氣警報器、地下室的一氧化碳或沼氣偵測器，都是氣體濃度感測器的應用實例。只要是瓦斯、一氧化碳、氫氣，有機溶劑等揮發性氣體及可燃性氣體，都能利用氣體感測器偵測。但是氣體感測器不同型號對於不同的氣體濃度也會有不同的反應，所以盡量不要用一種感測器來測量所有的氣體。

2、 氣體濃度感測器的偵測原理分類

表 1、偵測器種類

接觸燃燒式
構造：觸媒為白金線圈純度 999.999%，保持溫度 400° C 線圈先用鋁粉(單體)包起來，外面再加一層白金系觸媒。
檢知原理：氣體偵測素子一碰到可燃性氣體便會起觸媒作用。氣體濃度在爆炸下限(LEL)以下也會起氧化反應，造成發熱白金線圈溫度上升。氣體濃度與溫度、白金阻抗等成正比。利用電橋接回路可測定氣體濃度，再加上利用溫度補償阻抗，只要周遭有微小的溫度變化立即能有良好的溫度補償效應。
半導體式
構造:在一對白金屬線圈中夾著一體形成的金屬酸化物半導體 Sn02 的燒結體，線圈周圍由相同半導體材質所覆蓋。
檢知原理：經長期燒結工程處理，由金屬酸化物(Sn02)所造成 N 型半導體燒結體的內部內藏一對白金屬合金線圈(Pd-Ir) ，其中的一線圈加上電壓間加熱器用，約加熱至 350° C 來使用。半導體 SENSOR 在大氣中通電時 Sn02 粒子會吸著空氣中氧氣負離子，內部電子會把表面的負離子擠到中央，則通路就被阻礙。像 H2 之類的可燃氣體被吸附時會和表面的氧氣起反應，表面的氧氣負離子的濃度減

少，通路變寬，電流就容易流過。
熱半線型導體式
構造：經燒結金屬氧化物半導體(Sn02)包裹著白金線圈，它可保持溫度 300~450° C 的溫度，兼加熱器用。
檢知原理：熱傳導效果(固體熱傳導式)(PAT) ：當白金線圈通電時，金屬氧化物半導體(Sn02)的溫度 300~450° C，當可燃氣體流經 SENSOR 時造成電子濃度增加，因而半導體熱傳導度變佳，放熱性也變好，若半導體體溫度下降則白金線圈的阻抗也下降，此時白金與接觸燃燒式同樣具有溫度計功能。 電氣傳導效果(熱線型半導體式) (PAT.PEND.S.) ：偵測部分構造以電學來看白金線圈阻抗(Rh)半導體阻抗(Rs)兩者為並聯迴路。半導體吸附氣體時則 Rs 阻抗下降，合成阻抗 Rh 的阻抗相對也下降，此時熱傳導、電熱傳導變化可用電橋接迴路算出其電壓差。
氣體熱傳導式
構造：白金線圈外加上鋁，表面蒸著一層經惰性處理的玻璃物質。
檢知原理：利用標準氣體與測定氣體兩者熱傳導度知不同，當測定氣體進入時會將空氣的熱量奪走，白金線溫度因此下降。溫度下降與氣體濃度變化成直線正比例。白金線阻抗可由電橋迴路來表示。

薄膜式
構造：利用集積迴路技術在 1.5x3x06mm 的鋁基鈑知其中一面鋪有白金薄膜加熱器，另一面為白金薄膜電極，在電極面上形成半導體薄膜。
檢知原理：半導體表面如吸附氣體時，吸著分子和半導體會生成電子變化。
定電為電解式
構造：有 3 哥氣體電極(作用、比對、對極)及電解液構成，利用高耐藥性的塑膠容器包覆起來。
檢知原理：溶液和大氣的界面所造成的作用電及保持一定的電為且以直接電解方式。
隔膜流電電池式
構造：2 個電極(Pt-Pb)和隔膜及電解液所構成，為了使氣體在大氣中能在電極上快速反應，隔膜和電極必須非常密著。
檢知原理：貴金屬電極(Pt，Ag 等)和卑金屬電極(Pb)和電解液所構成，貴金屬電極透過鐵弗龍膜和空氣接觸，兩極間會產生電位差，接續負荷阻抗之後會有下列反應(貴金屬電極 02+2H2O+4e-40H-/卑金屬電極 2Pb2Pb2+2e-)其結果，空氣中氧氣濃度成正比例的電流從貴金屬電極到卑金屬電極流到外部迴路。起電力溫度依存，依據熱

感電組周圍溫度變化補償。
紅外線式
構造：以表面為加工技術製成的熱型紅外線感測器(microbolometer)會包凡兩個主要部分，一為熱輻射吸收層，另一為支撐此吸收層知浮板結構。
檢知原理：一般氣體分子的總能量是電子運動、振動、迴轉等能量的總合。紅外線式式利用紅外線領域(1~20μ)的振動能量。(氣體分子的種類不同，紅外線吸收的波長也不同。)紅外線領域的光照射到氣體分子時，因光的吸收會引起遷移作用吸收波長依分子的不同而異，即從波長可測知氣體的種類，吸收光可測知氣體的濃度。利用測定對象氣體的吸收特性作為波長的選擇，再將測定對象氣體封入偵測器，即可吸收地成份波長的紅外線

3、氣體偵測器的選擇

我們專題這次選擇 MQ-7、MQ-2 兩顆氣體感測器來做研究，這兩顆偵測器測的氣體，一氧化碳、瓦斯跟生活息息相關，像在居家瓦斯熱水器因為通風不量導致一氧化碳中毒、廚房管線老舊導致瓦斯外洩等。所以我們選擇跟生活最接近的氣體來做為偵測 。

4、MQ-7 介紹

圖 2.4.1、MQ-7 一氧化碳感測模組

表 2、MQ-7 標準工作環境

符號	參數名稱	技術條件	備註
Vc	回路電壓	≦10V	DC
VH	加熱電壓（高）	5.0V±0.2V	AC OR DC
VH	加熱電壓（低）	1.5V±0.1V	AC OR DC
RL	負載電阻	可調	
RH	加熱電阻	31Ω±3Ω	室溫
TH	加熱時間(高)	60±1 seconds	
TH	加熱時間(低)	90±1 seconds	
PH	加熱功耗	約 350mw	

表 3、MQ-7 環境條件

符號	參數名稱	技術參數	備註
Tao	使用溫度	-10℃~50℃	
Tas	存儲溫度	-20℃~70℃	建議使用範圍
RH	相對溫度	小於 95%RH	
O2	氧氣濃度	21%(標準規格) 氧氣濃度會影響 靈敏度特性	最小值度於 2%

表 4、MQ-7 靈敏度特性

符號	參數名稱	技術參數	備註
Rs	敏感體電阻	2KΩ-20KΩ	在 100ppmCO 中
α	濃度斜率	小於 0.6	Rs(300ppm)~(100ppm)
標準工作條件	溫度： 20℃±2℃ 相對溼度：65%±5%		
	Vc：5.0V±0.1V VH(高)：5.0V±0.1V VH(低)1.5±0.1V		
預熱時間	不短於 48 小時		探測範圍 10ppm ~ 1000ppm 一氧化碳

表 5、MQ-7 結構、外層

	部件	材料
1	氣體敏感層	二氧化錫
2	電極	金(Au)
3	測量電線引線	鉑(Pt) 4
4	加熱管	鎳鉻和金(Ni-Cr)
5	陶瓷管	三氧化二鋁
6	過濾層	活性碳顆粒
7	卡環	鍍鎳銅材(Ni-Cu)
8	基座	塑膠或尼龍
9	針狀管腳	鍍鎳銅材(Ni-C u)

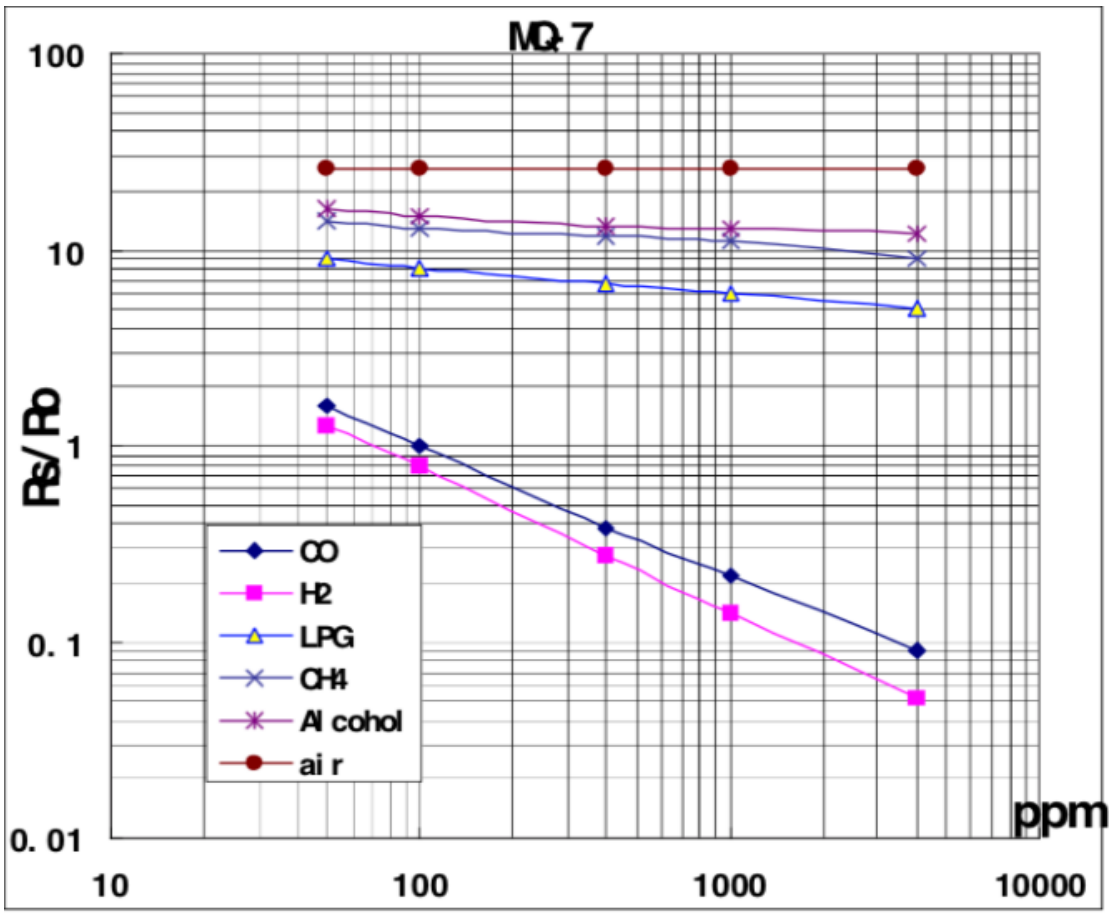


圖 2.4.2、 MQ-7 型氣敏元件的靈敏

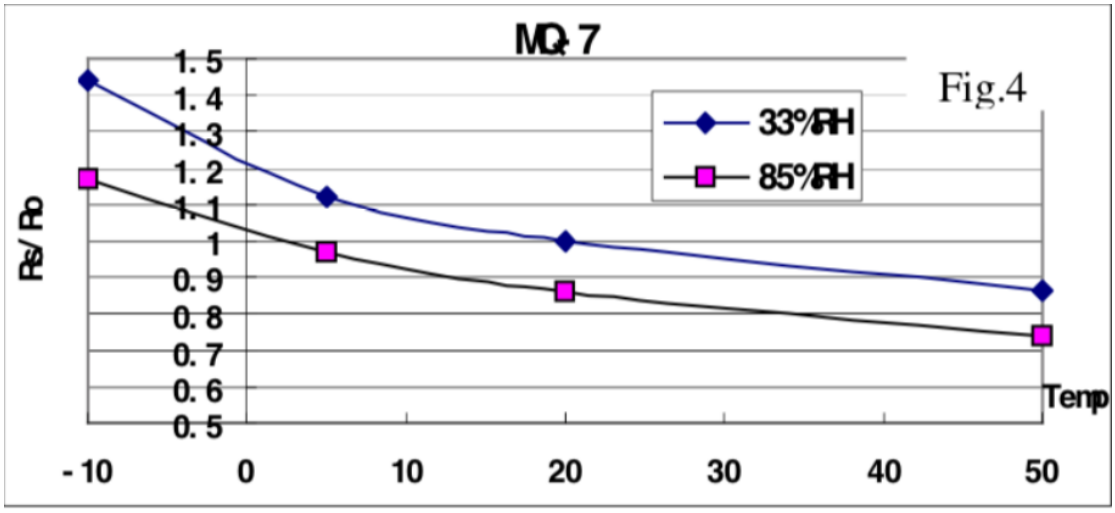


圖 2.4.3、 MQ-7 型氣敏元件的溫溼度

5、MQ-2 介紹

圖 2.4.4、MQ-2　煙霧感測器模組

表 6、MQ-2 標準工作環境

符號	參數名稱	技術條件	備註
Vc	回路電壓	≦15V	AC OR DC
VH	加熱電壓（高）	5.0V±0.2V	AC OR DC
RL	負載電阻	可調	
RH	加熱電阻	31Ω±3Ω	室溫
TH	加熱時間(高)	60±1 seconds	
TH	加熱時間(低)	90±1 seconds	
PH	加熱功耗	≦900mw	

表 7、MQ-2 環境條件

符號	參數名稱	技術參數	備註
Tao	使用溫度	-10℃~50℃	
Tas	存儲溫度	-20℃~70℃	建議使用範圍
RH	相對溫度	小於 95%RH	
O2	氧氣濃度	21%(標準規格) 氧氣濃度會影響 靈敏度特性	最小值度於 2%

表 8、MQ-2 靈敏度特性

符號	參數名稱	技術參數	備註
Rs	敏感體電阻	3KΩ-30KΩ	探測濃度範圍 100ppm-10000ppm 液化氣和丙烷 300ppm-5000ppm 丁烷 5000ppm-20000ppm 甲烷 300ppm-5000ppm 氫氣 100ppm-2000ppm 酒
α	濃度斜率	小於 0.6	
標準工作條件	溫度：20℃±2℃ Vc：5.0V±0.1V 相對溼度：65%±5% Vh：5.0V±0.1V		
預熱時間	不少於 24 小時		

表 9、MQ-2 結構、外層

	部件	材料
1	氣體敏感層	二氧化錫
2	電極	金(Au)
3	測量電線引線	鉑(Pt) 4
4	加熱管	鎳鉻和金(Ni-Cr)
5	陶瓷管	三氧
6	防爆網	100 目雙層不鏽鋼(SUB316)
7	卡環	鍍鎳銅材(Ni-Cu)
8	基座	電木
9	針狀管腳	鍍鎳銅材(Ni-C u)

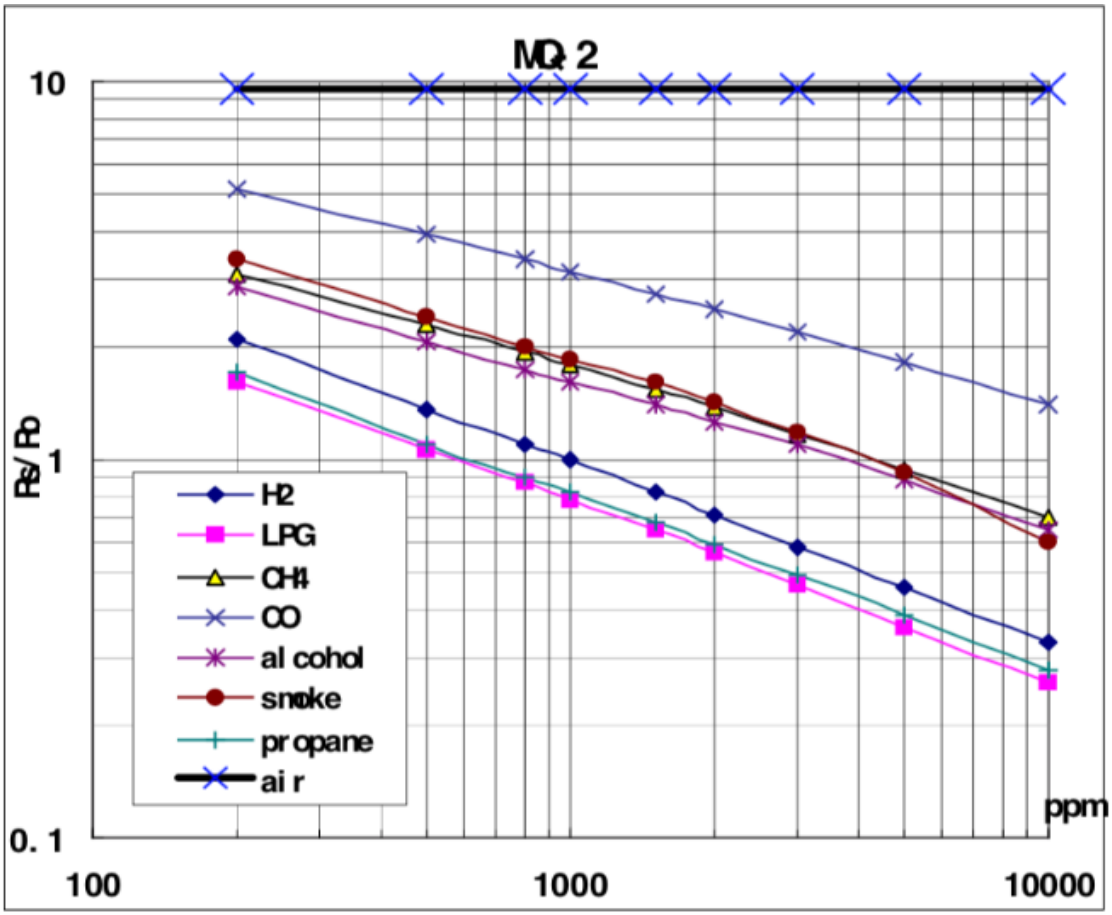


圖 2.4.5、 MQ-2 型氣敏元件的靈敏

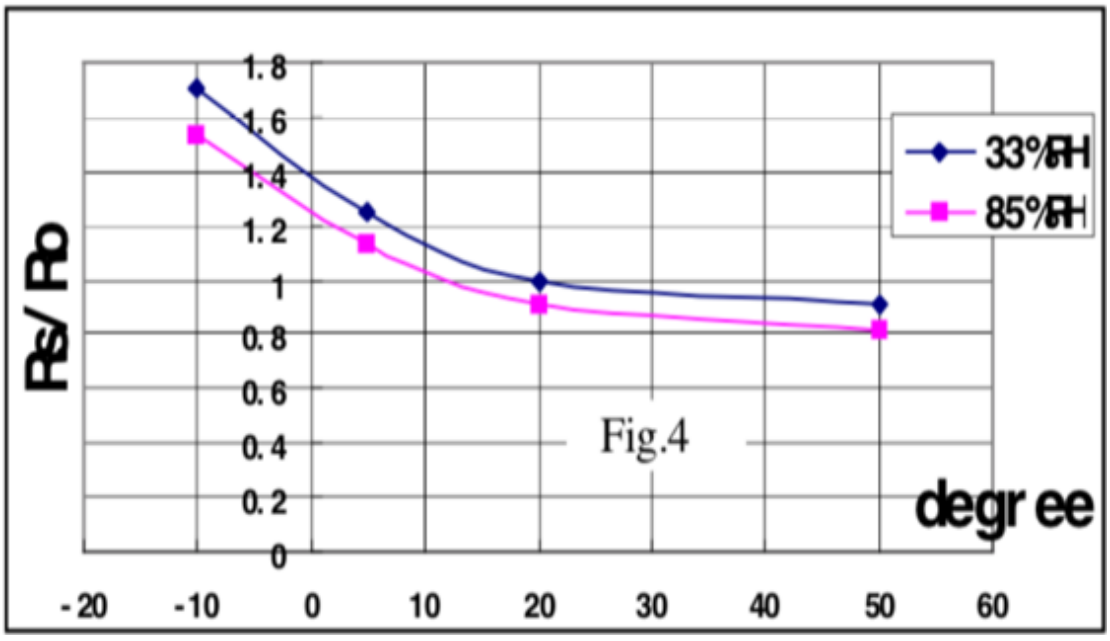


圖 2.4.6 、 MQ-2 型氣敏元件的溫溼度

三、軟體的介紹

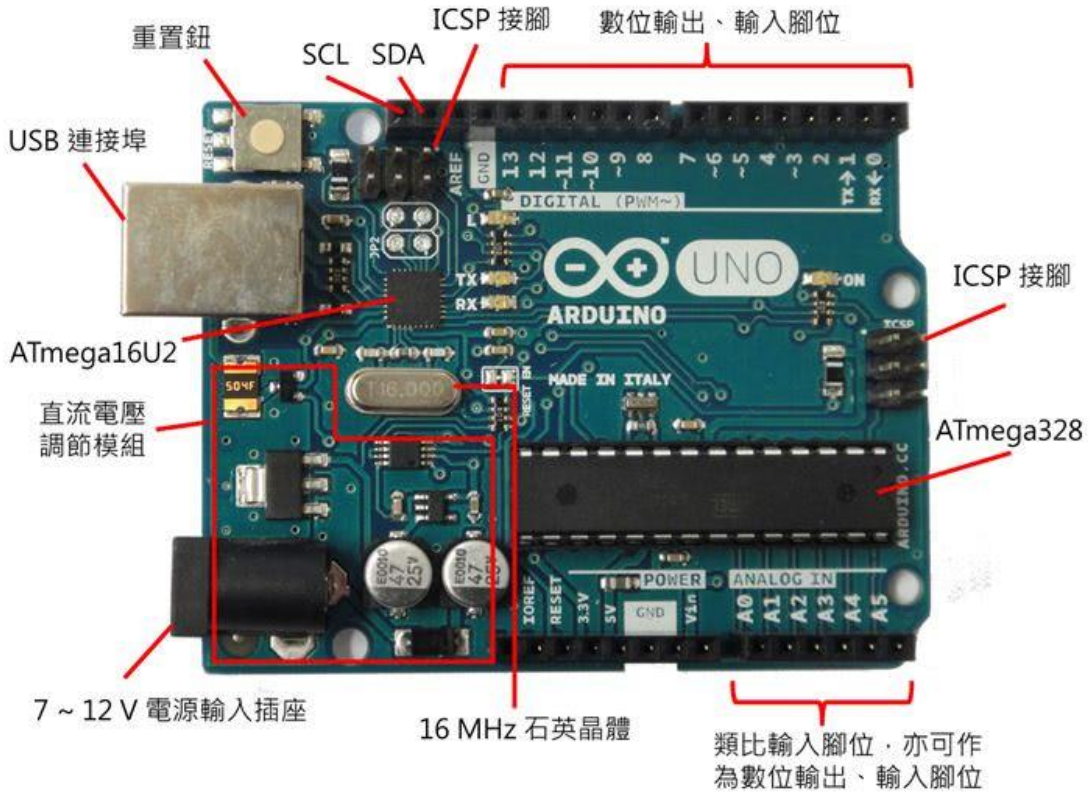


圖 2.5.1、Arduino Uno 整體圖

1、供電給 Arduino Uno

首先要認識，也是最重要的，就是如何提供電源給控制板，Arduino Uno 的運作直流電壓為 5V，以正確方式提供電源，不只是為了讓控制板正常運作，也為了避免損壞控制板。 特別注意，接下來的說明是針對 Arduino Uno，如果是其他控制板，請詳細閱讀 Arduino 官方網站上各控制板相關說明。Arduino Uno 可透過三個管道提供電源：USB 連接埠、電源輸入插座、Vin 腳位。透過 USB 連接埠提供給控制板的電源必須是 5V 的電壓，最基本的方式，就是透過 Type B USB

連接控制板，而另一頭是 Type A USB 連接個人電腦，這可以提供 5V、500mA 的電源給控制板，這也是本書一開始採取的方式，因為電腦要透過 USB 傳送程式給 Arduino。如果採藍牙連線或其他無線方式來傳送資料給控制板，不用連接個人電腦與控制板的話，那麼可以使用 USB 電源供應器，像是插頭或者行動電源等，請注意必須是 5V 的直流電壓，至少 2A 電流的輸出。

圖 2.5.2 使用 USB 供電給 Arduino Uno

如果要透過電源輸入插座供電，插頭直徑必須是 2.1 公釐，供應的電壓限制是 6V 至 20V，建議是直流 7V 至 12V，「低於 7V 的話，5V 輸出腳位可能電壓不足，控制板可能運作不正常，高於 12V 的話，電壓調節模組可能過熱而損壞控制板」。透過電源輸入插座供電時，可以使用電源供應器、9V 電池等來供應。

圖 2.5.3 使用電源輸入插座供電給 Arduino Uno

在圖 2.5.1 中，可看到電源輸入插座旁的直流電壓調節模組，這可以將輸入的直流電壓降為 5V 以及 3.3V，5V 除了可供控制板使用之外，也可以透過標示為 5V 的腳位輸出給其他電子電路元件使用，有些電子電路元件必須在 3.3V 才能正常運作，這可以透過標示為 3.3V 的腳位來提供。

透過 Vin 供電給控制板時，要求與透過電源輸入插座相同，Vin 提供的電壓，也是經由直流電壓調節模組調整為 5V 與 3.3V。由於直流電壓調節模組的輸入端連接了電源輸入插座供電與 Vin，因此，如果你透過電源輸入插座供電，Vin 也會量得與電源輸入插座相同的電

壓，因此，當有兩塊 Arduino 控制板需要電源時，可將接有過電源輸入插座的控制板上 Vin，連接至另一塊控制板的 Vin。

在〈Arduino Uno〉官方說明文件中提到，直接將 5V 電源接至控制板上 5V 腳位來提供電源，雖然是可行的，不過這種方式沒有經過電壓調節，可能會損壞控制板，因此並不建議！

控制板上有個標示為 ON 的 LED，用來指示電源供應狀況，如果在已提供電源的情況下，發現標示為 ON 的 LED 閃爍、黯淡或熄滅，應儘快拔除電源，檢查電源供應是否正常，以避免損壞控制板。Arduino Uno 控制板上有三個 GND 腳位，可作為電路上的電壓基準，應避免將 5V、3.3V、Vin 或其他輸出腳位直接連接 GND，以免造成短路，嚴重時會損壞控制板。

2、數位輸出、輸入腳位

顧名思義，這些腳位可輸出高電位 5V 與低電位 0V 的數位訊號，也可接受數位訊號，腳位編號為 0 到 13，不過通常會寫為 D0 到 D13 表示為數位腳位（D 代表 Digital）。圖 1.11 中可以看到，D3、D5、D6、D9、D10、D11 的數字編號旁，還有個波浪符號（~），這表示這些腳位可以用數位訊號來模擬出類比訊號，使用的方式是 PWM（Pulse Width Modulation），PWM 在之後的實作中會再加以說明。

要注意的是 D0 與 D1 這兩個數位腳位，分別被標示了 RX（Receiver）、TX（Transmitter），這兩個腳位用於序列埠傳送，且與 USB 序列埠連接，因此電腦若使用 USB 與控制板互傳資料時（可見到控制板上標示為 RX、TX 的 LED 閃爍），應避免使用 D0、D1 兩個腳位。

D13 腳位連接著控制板上標示為 L 的 LED，若是原廠控制板，預設會燒錄一個令 D13 定時切換高低電位的 Blink 程式，因此，首次接上電源時，會看到標示為 L 的 LED 不斷閃爍，這是初步檢視控制板是否功能正常的方式。

3、類比輸入腳位

圖 1.11 右下角有 A0 至 A5 六個腳位，可用來接受類比電壓輸入，但不能輸出類比電壓，如前所述，類比電壓必須透過數位腳位 D3、D5、D6、D9、D10、D11，以 PWM 模擬。

控制板上 ATmega328 內建類比數位轉換器（Analog-to-digital converter，簡稱 ADC），預設會將 0V 到 5V 轉換為 0 至 1023 的數值。對於輸出電壓為其他範圍的電路模組，可以透過 AREF 與 analogReference 函式，來提供參考電壓，例如對於輸出為 0V 到 3.3V

的電路模組，可以將 0V 至 3.3V 對應至 0 至 1023 的數值，獲得更高的解析度。

實際上，A0 至 A5 也可作為數位輸出、輸入腳位使用，此時 A0 至 A5 分別可視為 D14 至 D19。

4、重置鈕與 RESET

控制板左上方有個重置鈕，按下重置鈕會使得控制板重新執行使用者燒錄之程式，控制板下方有個 RESET 腳位，當它處於低電位時，也會有相同的重置效果。

5、其他腳位與元件

控制板右方有個顯而易見的晶片，那就是控制板的大腦 ATmega328，採雙列直插封裝（Dual in-line package），又稱 DIP 封裝，市面上有單獨銷售 ATmega328，如果控制板上的 ATmega328 損壞，可以使用小的一字起子或尖嘴鉗將之撬起，用新的 ATmega328 取代。

ATmega328 右上方有個 2x3 線上燒錄用的 ICSP（In-Circuit Serial Programming）腳位，可連 ISP 線上燒錄器

（In-systemprogrammer），對 ATmega328 進行程式燒錄，其中幾個腳位與 D11、D12、D13 連接在一起。

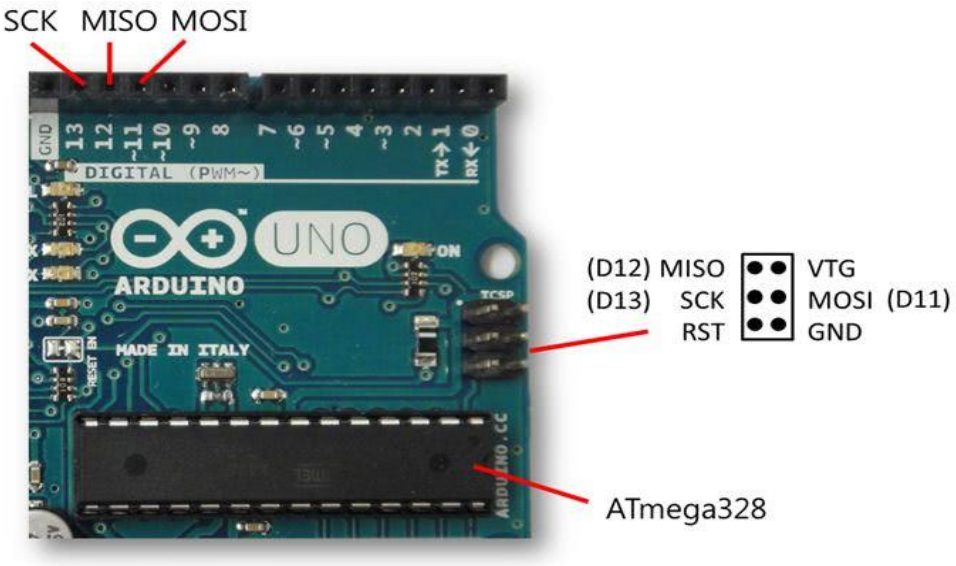


圖 2.5.4 ATmega328 與 ICSP

Arduino Uno 本身也可以當作線上燒錄器，用來燒錄另一台 Arduino 控制板上的晶片。

控制板左上也可看到控制 USB 的 ATmega16U2，預設燒錄了 USB 轉序列埠的韌體，你也可以透過鄰近的 ICSP 腳位來燒錄新的程式。

控制板左上方有 SCL（Serial Clock）、SDA（Serial Data）兩個腳位，這是用來連接支援 I2C（Inter IC）協定的設備，此協定只採用兩條接線，可在感測器與控制板如 Arduino 之間傳遞資訊，SCL 與 SDA 也分別與類比腳位 A5 與 A4 連接，方便接線之用。

D10、D11、D12、D13 是作為 SPI（Serial Peripheral Interface）協定的介面，SPI 使用四條線連接控制板與支援之設備。Arduino 同時支援 I2C 與 SPI 兩種協定，在一些介紹到使用 I2C 或 SPI 設備的 Arduino 書籍中，都會有相關的說明。

Arduino Uno 有兩個可設定中斷（Interrupt）處理的腳位，中斷 0 與中斷 1 分別為 D2 與 D3，可使用 attachInterrupt 函式來設定中斷服務常式（Interrupt Service Routine）。

控制板左下有個 IOREF 腳位，這可讓其他設備知道 Arduino 控制板的運作電壓，其左邊有個腳位目前（Arduino Uno R3）沒有任何作用。

表 10. Arduino Uno 規格介紹

名稱	Arduino Uno
尺寸	2.95 吋 x2.1 吋
處理器	ATMega328
每秒周轉速數	16MHz
存取記憶體	2kB
記憶體	32kb
EEPROM	1kb
輸入電壓	7-12 伏特
最低輸出功率	42 毫安培(.3 瓦)
Digital GPIO	14
類比輸入	6X10 位元
PWM	6
TWI/I2C	2
SPI	1
UART	1
Dev IDE	Arduino Tool

第三章、 專題成果與分析

一、 程式設計大綱

1. 一氧化碳偵測器 MQ-7 跟可燃氣體瓦斯煙霧偵測器 MQ-2 雙氣體偵測器裝備偵測。

2. MQ 系列的感測器是需要暖機的， 等待裡面的加熱器溫度穩定. 所以開機後設計 180 秒暖機時間倒數.倒數完畢後才會開始顯示讀值跟濃度警報。

3. 一個三段七段 LED 顯示器， 透過 Arduino Shield 上的按鈕可以隨意切換跟即時顯示 由 MQ-7 所傳回的一氧化碳濃度程度的讀值或是由 MQ-2 所傳回的可燃氣體瓦斯煙霧濃度程度的讀值。

4. 即時警報， 警報聲音的大小跟頻率會隨著不同濃度而改變. 洩漏氣體濃度越濃， 警報聲音越大聲跟越急促. 使用者不用隨時盯著三段七段 LED 顯示器， 可以透過耳朵所聽到的聲響， 立即感受或察覺目前所發生的瓦斯或一氧化碳洩漏的嚴重程度。

5. Smart I/O， 接上電腦 USB Port 後， 隨時透過 USB 以文字傳送更詳細的資訊供電腦端的程式使用。

二、部分程式解說

1、 3 位 7 段 LED 顯示器

3 位 7 段 LED 顯示器控制方面， 硬體接腳跟控制程式都跟這篇 BLOG 一模一樣. 字型表添加了這 6 個字 'G'， 'A'， 's'， 'C'， 'o'， ' '， 用在當使用者按了按鈕來切換兩個不同感測器的顯示時， 會先顯示一下提示字元 'GAs' 或是 'Co'， 來提醒使用者， 現在顯示的讀值是 Gas 瓦斯 (MQ-2) 呢? 還是 Co 一氧化碳 (MQ-7)。

```
const byte segs_data[17]={B01111110, B00110000,B01101101,
B01111001,B00110011,B01011011,B01011111,B01110000,
B01111111,B01110011,B00000001,B01011110,B01110111,
B01011011,B01001110,B00011101,B0} ;
```

然後 prep_Char(digit，index，dot)，用來準備所要顯示的字元，digit 是三位 7 段 LED 顯示器的位置，由左算起位置為 digit =1，digit=2，digit=3， 字形表的索引位置為 index， 要不要顯示小數點為 dot = trueor false.prep_LED(num)， 用來準備所要顯示的整數數字 numlightup_beep()，程式裡用來點亮 3 位 7 段 LED 顯示器並觸動壓電喇叭造出特定聲音的函式. 程式裡必須不斷的呼叫這個函式， 維持住 三個 LED 恆亮的錯覺跟產生特定頻率的聲響。

2、感測器 MQ-2、MQ-7 的處理

很直接， 直接用 analogRead 讀取 ADC 所傳回來 0 - 1024 間代表氣體濃度訊號的數值，

有稍微測試了一下， 設定了底下兩個警告或是警報的濃度訊號

CO_Val=compensate(analogRead(CO_In)，CO_scalar，CO_offset);

Cas_Val=compensate(analogRead(Gas_In)，Gas_scalar，Gas_offset);

```
const int CO_warning_lim = 80;
const int CO_alarm_lim = 600;
const int Gas_warning_lim = 200;
const int Gas_alarm_lim = 500;
```

設計當感測器讀值小於 warming_lim 時， 一概視為正常. 不會發出任何警報， 此時 濃度 level 視為 零，

level = 0

當感測器讀值介於 warming_lim 跟 alarm_lim 之間時， 此時 濃度 level 根據它們在這兩個之間的比

例， 切成 10 等分， level = 1 - 10 (利用 map 函式來做這樣的對應分配的動作)

當感測器讀值大於 alarm_lim 之間時， 視為濃度 level 最大， level = 11

int CO_level; CO 濃度 level

int Gas_level; Gas 濃度 level

if (CO_Val <= CO_warning_lim) CO_level = 0; 小於 warming_lim 時， 濃度 level=0

else if ((CO_Val > CO_warning_lim) && (CO_Val < CO_alarm_lim)) CO_level = map(CO_Val，CO_warning_lim，CO_alarm_lim，0，10); 利用 map() 切成 10 等分給濃度 level

else CO_level = 11; 濃度大於 alarm_lim， 濃度 level=11

if (Gas_Val <= Gas_warning_lim) Gas_level = 0; 小 於 warming_lim 時 ， 濃 度 level=0

else if ((Gas_Val > Gas_warning_lim) && (Gas_Val < Gas_alarm_lim)) Gas_level = map(Gas_Val，Gas_warning_lim，Gas_alarm_lim，0，10); 利用 map() 切成 10 等分給濃度 level

else Gas_level = 11; 濃度大於 alarm_lim， 濃度 level=11

3、 警報聲音的產生

透過 beep(freq， cycle) 函式來產生警報聲音， 每呼叫一次就會以頻率 freq 觸動喇叭 cycle 次數.

先用 ttt = 340000 / freq 算出觸動喇叭後需要等待的週期(方波)， 然後用個 do-loop 執行 cycle 次數. 這樣就可以產生一個特定頻率的聲音

```
void beep(unsigned long freq，unsigned int cycle)
{
 unsigned long ttt = 340000 / freq;
 for (unsigned int i=0;i<cycle;i++)
 {
 digitalWrite(speaker_out，HIGH);
 delayMicroseconds(ttt);
 digitalWrite(speaker_out，LOW);
 delayMicroseconds(ttt);
 }
}
```

聲音的頻率由兩個氣體的濃度的最高那個來決定， 濃度愈高， 聲音的頻率愈低， 人耳的關係對中頻的聲音較高頻感受為強! 所以當聲音頻率從高頻轉到中頻， 人耳感受會從弱慢慢轉強. 這樣就不用看 7 段 LED 顯示器就可以由頻率的變化知道氣體目前洩漏的嚴重性.

聲音頻率的高低由 beep_level 來決定， beep_level 等於兩個氣體濃

度 level， 濃度最濃的那一個.

```
 if (CO_level<Gas_level) beep_level = Gas_level;
 else beep_level = CO_level;
```

聲音頻率的高低由 beep_level 來決定，當 beep_level = 0 則不發出聲音. beep_level > 0，一樣利用 map()函式把 beep_level = 0 - 11 之間的數值，對應到頻率 beep_freq 從 50000-1000

```
void lightup_beep()
{
 unsigned long beep_freq = map(beep_level，0，11，50000，1000);

 if (beep_level>0) beep(beep_freq，5);

 }
}
```

4、主程式迴圈，中斷處理

兩個重要的狀態變數 prev_state 跟 main_state，當其為 0 表示七段 LED 顯示器顯示 Co 濃度，當其為 1 表示七段 LED 顯示器顯示 Gas 濃度. 當有人按了按鈕會觸發中斷，main_state 會被改變.檢查 prev_state 跟 main_state 的差異，就可以察覺有人按了按鈕，這時候要顯示'Co'或是'Gas'提醒字元於七段 LED 顯示器.

```
byte prev_state = 0;
volatile byte main_state = 0;

// 0 = Co， 1 = Gas

void sw_sense()
{
 if (millis() - time > 250)
```

```
 {
 time=millis();
 main_state ++ ;
 if (main_state==2) main_state=0;
 }
}
```

另外一個重要常數是 time_interval， 程式每等到 time_interval 所定義的 ms 秒數後才會讀取一次偵測器的濃度讀值.這裡是 1000 ms，所以每秒一次.

```
unsigned long time_interval =1000;
```

5、主程式迴圈介紹

```
void loop() {
...
 CO_Val = compensate(analogRead(CO_In)，CO_scalar，CO_offset);
```

讀 Co 感測器濃度訊號

```
Gas_Val = compensate(analogRead(Gas_In)，Gas_scalar，Gas_offset);
```

讀 Gas 感測器濃度訊號

... 透過 USB 序列埠傳給電腦， 假如有接電腦的話.

... 濃度跟 level 的處理

... 透過 USB 序列埠傳給電腦， 假如有接電腦的話.

```
unsigned long target_time = millis() + time_interval;
```

每 time_interval 後離開迴圈， 讀取感測器一次

```
while (millis() < target_time) 7 段 LED 顯示及喇叭發聲迴圈
{
if (prev_state!=main_state) 兩個不相等，代表有人剛按了按鈕產生了
中斷，main_state 被改
變了，所以要顯示一下'Gas'或'Co'
{
switch(main_state) 根據 main_state， 顯示 'Gas' 或是 'Co' 於七段
LED 顯示器
{
case 0 : prep_Char(0，14，false);
prep_Char(1，15，false);
prep_Char(2，16，false);
break;
case 1 : prep_Char(0，11，false);
prep_Char(1，12，false);
prep_Char(2，13，false);
}
prev_state = main_state; 更新 prev_state 跟 main_state 一樣.
for (int i=0;i<50;i++) lightup_beep();
}
else
{
switch(main_state) 根據 main_state， 決定是顯示 Co 讀值還是 Gas
讀值
```

```
  {
  case 0 : prep_LED(CO_Val);
  break;
  case 1 : prep_LED(Gas_Val);
  }
  }

  lightup_beep();
  }
}
```

三、完整程式列表

```
#define CO_In A0
#define Gas_In A1
#define speaker_out A2

// 7segments LED output pins defination
#define digit1 13
#define digit2 10
#define digit3 9
#define aa 12
#define bb 8
#define cc 6
#define dd 4
#define ee 3
#define ff 11
#define gg 7
#define dp 5

// time interval (mS) for measurement
unsigned long time_interval = 1000;
unsigned long warmup_time = 180000;

// beep control
int beep_level;

// data structure 8bits = dp aa bb cc dd ee ff gg
// 0 to 9, '-', 'G', 'A', 's', 'C', 'o', ''
const byte segs_data[17]={B01111110, B00110000, B01101101,
B01111001,B00110011, B01011011, B01011111, B01110000, B01111111,
B01110011, B00000001,B01011110, B01110111, B01011011,
B01001110, B00011101, B0} ;
const byte seg_pins[8]={gg,ff,ee,dd,cc,bb,aa,dp};
const byte digit_pins[3]={digit1,digit2,digit3};
byte seg7[3]={0,0,0};
```

```
// calibration constants
const float CO_scalar=1;
const float CO_offset=0;
const float Gas_scalar=1;
const float Gas_offset=0;

float compensate(float raw,loat scalar,float offset)
{
  return(scalar*raw + offset);
}
void prep_Char(byte digit,byte index,boolean dot)
{
  // digit = 0,1,2
  // index = 0..10

  seg7[digit] = segs_data[index];
  if (dot) seg7[digit] |= B10000000;
}
void prep_LED(int num)
{
  boolean neg = false;
  if (num<0) neg = true;
  num=abs(num);
  prep_Char(2,um % 10,true);
  num /=10;
  if (num==0) seg7[1] = 0;
  else prep_Char(1,um % 10,false);
  num /=10;
  if (neg) prep_Char(0,10,false);
  else
  {
    if (num==0) seg7[0] = 0;
    else prep_Char(0,um % 10,false);
  }
}
void lightup_beep()
{
  for (byte i=0;i<3;i++)
```

```
  {
   byte segments = seg7[i];
   if (segments==0) continue;
   // select digit
   for (byte j=0;j<3;j++)
   {
      // common anode,set output to HIGH to select digit what you
want,the other digits must be set to LOW
      // common cathode,set output to LOW to select digit what you
want,the other digits must be set to HIGH

      if (j==i) digitalWrite(digit_pins[j],HIGH);
      else digitalWrite(digit_pins[j],LOW);
   }

   // acording char data,light each segments on digit
   for ( byte j=0; j<8; j++)
  {
   // common anode,set output to LOW to light up LED,HIGH to off LED
   // common cathode,set output to HIGH to light up LED,LOW to off
LED

   if ((segments & B00000001) == 1) digitalWrite(seg_pins[j],LOW);
   else digitalWrite(seg_pins[j],HIGH);
   segments >>=1;
  }
  delay(5);

  unsigned long beep_freq = map(beep_level,,1,0000,000);
  if (beep_level>0) beep(beep_freq,);
  }
}
// beep
void beep(unsigned long freq,nsigned int cycle)
{
  unsigned long ttt = 340000 / freq;
  for (unsigned int i=0;i<cycle;i++)
  {
```

```
  digitalWrite(speaker_out,IGH);
  delayMicroseconds(ttt);
  digitalWrite(speaker_out,OW);
  delayMicroseconds(ttt);
  }
}

// interrupt service

byte prev_state = 0;
volatile byte main_state = 0;
// 0 = Co,1 = Gas

unsigned long time;

void sw_sense()
{
  if (millis() - time > 250)
  {
    time=millis();
    main_state ++ ;
    if (main_state==2) main_state=0;
  }
}


void setup() {
  // setup I/O
  pinMode(speaker_out,OUTPUT);

  // setup 7seg LED pins mode
  pinMode(digit1,OUTPUT);
  pinMode(digit2,OUTPUT);
  pinMode(digit3,OUTPUT);
  pinMode(aa,OUTPUT);
  pinMode(bb,OUTPUT);
  pinMode(cc,OUTPUT);
  pinMode(dd,OUTPUT);
```

```
  pinMode(ee,OUTPUT);
  pinMode(ff,OUTPUT);
  pinMode(gg,OUTPUT);
  pinMode(dp,OUTPUT);

  // attach ISR
  attachInterrupt(0,sw_sense,FALLING);

  // setup serial port
  Serial.begin(9600);

  // startup time
  unsigned long startup_time = millis();

  Serial.print(" Warming Up...");
  while (millis() - startup_time < warmup_time)
  {
    prep_LED((int)((warmup_time - millis() + startup_time)/1000));
    beep_level=0;
    lightup_beep();
  }
  prep_Char(0,14,false);
  prep_Char(1,15,false);
  prep_Char(2,16,false);
  for (int i=0;i<50;i++) lightup_beep();

}

int CO_Val;
int Gas_Val;
const int CO_warning_lim = 80;
const int CO_alarm_lim = 600;
const int Gas_warning_lim = 200;
const int Gas_alarm_lim = 500;

// beep control
```

```
void loop() {
// main for measurement test

  int CO_level;
  int Gas_level;


  CO_Val = compensate(analogRead(CO_In)，CO_scalar，CO_offset);

  Gas_Val = compensate(analogRead(Gas_In)，Gas_scalar，Gas_offset);


  Serial.print("CO Concentration: ");
  Serial.print(CO_Val);
  Serial.print(" Gas Concentration: ");
  Serial.print(Gas_Val);

  if (CO_Val <= CO_warning_lim) CO_level = 0;
else if ((CO_Val > CO_warning_lim) && (CO_Val < CO_alarm_lim))
CO_level = map(CO_Val，CO_warning_lim，CO_alarm_lim，0，10);
  else CO_level = 11;

  if (Gas_Val <= Gas_warning_lim) Gas_level = 0;
else if ((Gas_Val > Gas_warning_lim) && (Gas_Val < Gas_alarm_lim))
Gas_level= map(Gas_Val,Gas_warning_lim,Gas_alarm_lim,0,10);
  else Gas_level = 11;

  if (CO_level<Gas_level) beep_level = Gas_level;
  else beep_level = CO_level;

  Serial.print(" CO_Lvl:");
  Serial.print(CO_level);
  Serial.print(" Gas_Lvl:");
  Serial.println(Gas_level);

  unsigned long target_time = millis() + time_interval;

  while (millis() < target_time)
```

```
 {
 if (prev_state!=main_state)
 {
 switch(main_state)
 {
  case 0 : prep_Char(0,14,false);
                  prep_Char(1,15,false);
                  prep_Char(2,16,false);
                  break;
    case 1 : prep_Char(0,11,false);
                    prep_Char(1,12,false);
                    prep_Char(2,13,false);
 }
  prev_state = main_state;
  for (int i=0;i<50;i++) lightup_beep();
 }
 else
 {
 switch(main_state)
 {
 case 0 : prep_LED(CO_Val);
                break;
 case 1 : prep_LED(Gas_Val);
  }
 }
 lightup_beep();
  }
}
```

四、有害氣體偵測器之成品

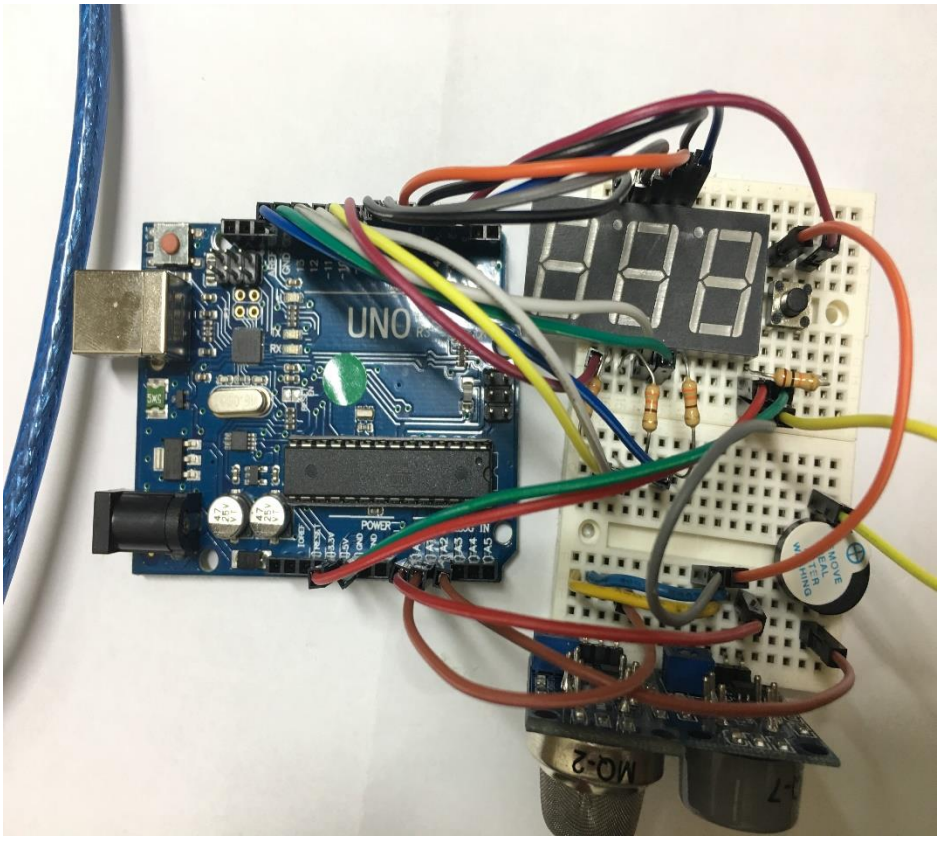


圖 4.1 有害氣體偵測器

說明:通電後，訊號輸入，Arduino Uno 訊號燈亮、MQ-2，MQ-7 通電指示燈亮以及三位元七段顯示器顯示 180，顯示意指暖機倒數 180 秒(感測元件須預熱）。

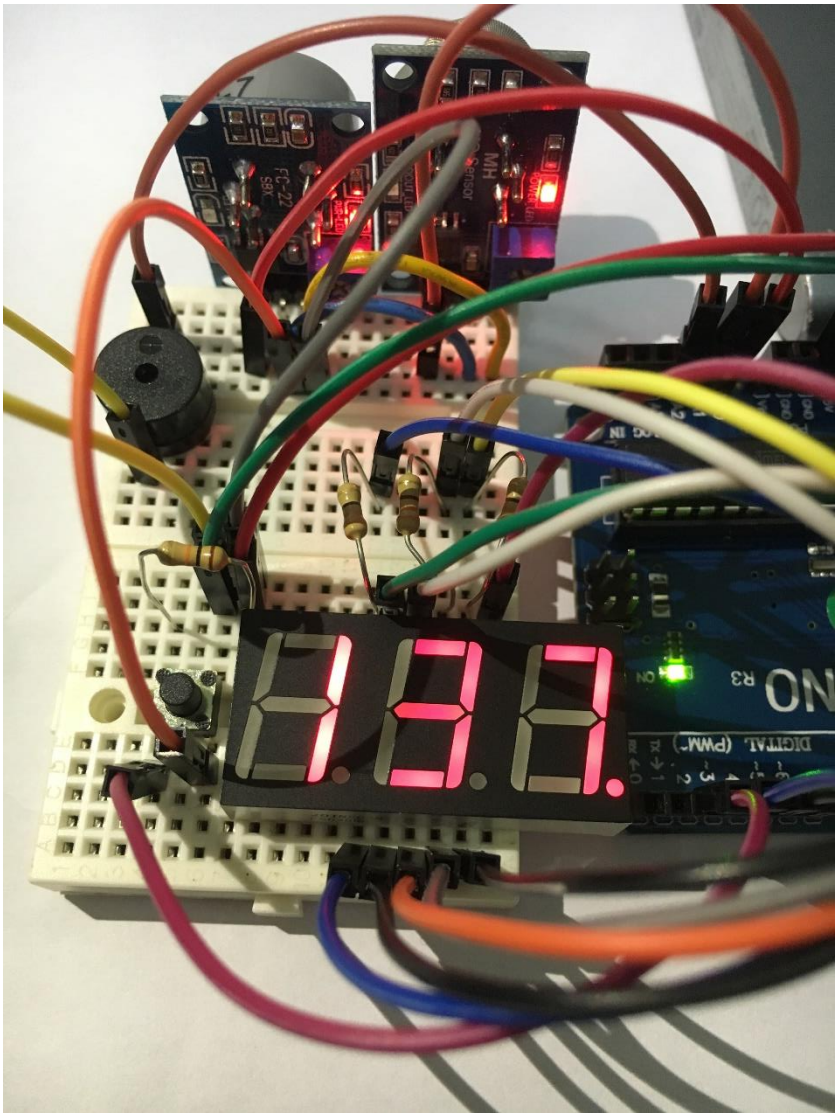


圖 4.2 運中之有害氣體偵測器

說明:180 秒後開始偵測，蜂鳴器開始低頻鳴響，顯示一氧化碳(Co)數值，按下按鈕開關可切換成瓦斯(Gas)，諾超過預設值，開始高頻鳴響。

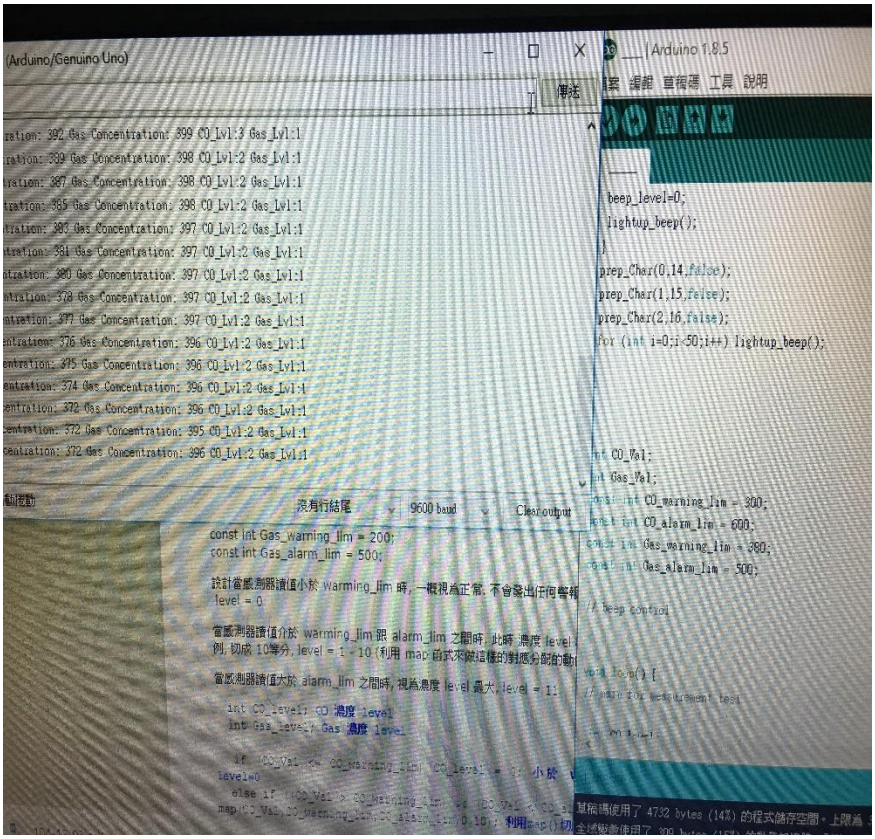


圖 4.3 監測偵測值

說明:偵測的數值可以透過電腦顯示，也可以透過電腦改動偵測數值的範圍，可把範圍值化成 10 等份，範圍值越高蜂鳴器蜂鳴聲越大，代表危險係數也越高。

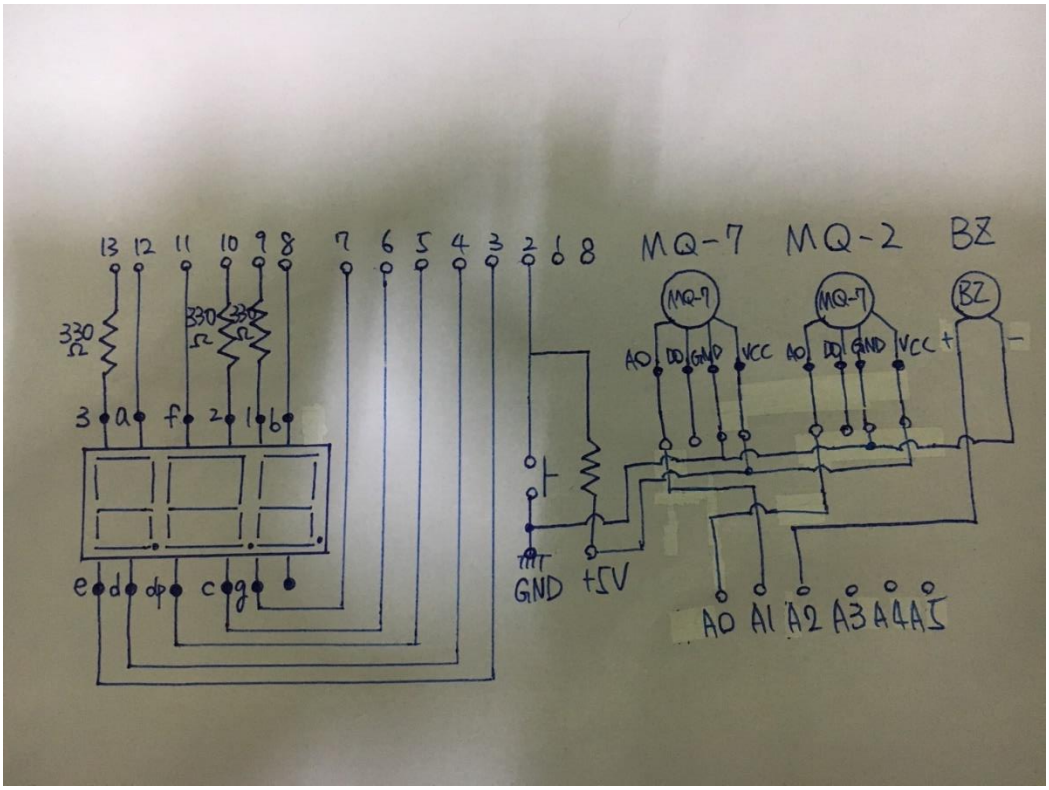


圖 4.4 有害氣體偵測器電路圖

五、結論

這次的專題讓我們受益良多,也清楚知道自己的不足之處，也讓我們學會了許多的新事物，包括這次最主要的 Arduino Uno 軟體、MQ 系列感測元件…等，此次專題的軟體其實沒有太多的涉略，也是我們花了許多的時間下去找尋與學習，雖然沒有非常融會貫通，但能做出此專題，也讓我們獲得滿大的成就感，但是畢竟沒有非常會運用，造成這次的專題有些許瑕疵,讓我們明白自己能力不足的地方,謝謝許多老師、學長的幫助與關懷，才能讓我們完成這次的專題。

第四章、 參考文獻

MQ-7 20~2000ppm 一氧化碳感測模組 | 廣華電子

http://shop.cpu.com.tw/product/45459/info/

MQ-2 300~10000ppm 煙幕感應器（液化氣、丁烷、丙烷、甲烷、酒精、氫氣）| 廣華電子

http://shop.cpu.com.tw/product/46518/info/

MQ-7 中文参数书 PDF_百度文库

https://wenku.baidu.com/view/1fec7fc16137ee06eff918ef.html
2110000001001_MQ-2.pdf
http://www.jin-hua.com.tw/upload/product/2110000001001_MQ-2.pdf
Arduino - Home https://www.arduino.cc/

一氧化碳/煙霧暨瓦斯氣體偵測警報器製作 [Arduino] @ 早安，苦命工程師的胡言亂語 :: 痞客邦 ::

http://ohiyooo2.pixnet.net/blog/post/402845699-%E4%B8%80%E6%B0%A7%E5%8C%96%E7%A2%B3-%E7%85%99%E9%9C%A7%E6%9A%A8%E7%93%A6%E6%96%AF%E6%B0%A3%E9%AB%94%E5%81%B5%E6%B8%AC%E8%AD%A6%E5%A0%B1%E5%99%A8%E8%A3%BD%E4%BD%9C-%5Bard

三位 LED 七段顯示器應用 [Arduino] @ 早安，苦命工程師的胡言亂語 :: 痞客邦 ::

http://ohiyooo2.pixnet.net/blog/post/398356945-%E4%B8%89%E4%BD%8D%E4%B8%83%E6%AE%B5led%E9%A1%AF%E7%A4%BA%E5%99%A8%E6%87%89%E7%94%A8-%5Barduino%5D

劉昭安 施宏偉(101)。居家瓦斯偵測器。修平科技大學電機工程系實務專題報告

附錄

項目	設備、材料名稱	規格	數量
1	筆電	WINDOWS	1 台
2	Arduino UNO 軟體	1.8.5	1 份
3	USB 傳輸線	A 公~B 公	1 條
4	Arduino UNO 硬體	一般規格	1 台
5	麵包版	一般規格	1 塊
6	杜邦線	一般規格	20 條
7	MQ-2	5V	2 個
8	MQ-7	5V	2 個
9	電阻器	330Ω	3 個
10	電阻器	10K	1 個
11	按鈕開關	一般規格	1 個
12	蜂鳴器	5V	1 個
13	3 位 7 段顯示器	(陽極)	1 個