中華民國第 59 屆中小學科學展覽會

作品說明書

高級中等學校組　農業與食品學科

佳作

052202

因「菜」施教－自製水耕蔬菜自動化栽培之研究

學校名稱：高雄市立高雄女子高級中學

作者：	指導老師：
高二 施惠馨 高二 張庭瑜 高二 黃彥蓉	呂雲瑞

關鍵詞：自動化、水耕蔬菜、Arduino

摘要

本研究自行開發的水耕蔬菜養殖機，**自動監控植物生長環境，透過自動加換水與營養液等方式做出相對回應，讓使用者只需負責採收與觀察。**為達成此目的，我們使用 Arduino UNO 板作為控制面板，**在室溫下**以影響水質主要的三個變因做為監測對象：**導電度計(EC)、酸鹼值計(pH)、溶氧量計(DO)，並自製水位計，自動控制植物最佳生長環境**。本研究水耕蔬菜較市售蔬菜小，但**硝酸鹽含量低**，較無病蟲災害，機器自動種植能使**蔬菜生長更為迅速且無須施用農藥**。此外，我們調整機台空間配置，並美化機器整體外觀，使**機器可架設在一般室內空間**。最後，我們估算自製水耕蔬菜機的成本，期望在**實際應用上能擴大規模，使單次種植棵數增加，降低成本、節省人力資源。**

壹、研究動機

在現在社會的都市叢林中，消費者可以很方便的取得他們想要的食物，但是也因為如此，很多人並不會特別留意自己買了什麼、吃了什麼，也不在意它們的生產過程，這樣的習慣可能會對健康造成危害，例如：人體一天攝取的硝酸鹽總量有 86%來自蔬菜，硝酸鹽是什麼？硝酸鹽原本就是大自然氮素循環的一部分，因為植物無法直接利用蛋白質，必須經由細菌將它分解成硝酸鹽，才能被植物吸收。而植物利用硝酸鹽來合成胺基酸及蛋白質，成為植物體內的營養成分。腸道及口腔中的細菌會將蔬菜中的硝酸鹽，分解產生亞硝酸鹽，這些亞硝酸鹽，最後在胃腸道中與次級胺，合成亞硝胺等致癌物質。成年人每公斤體重，能夠忍受 3.66 毫克的硝酸鹽，以 60 公斤成人而言，每日上限約為 220 毫克，若一日攝取單一菜種 300 克，其硝酸鹽含量不應超過 720ppm，但市售蔬菜硝酸鹽含量經常超出數倍，若進食高硝酸鹽含量葉菜，很容易超過一日限量。

如果能在家自己種植蔬菜，人們就可以自由選擇營養液，在蔬菜整個生長週期中，使用的水源、燈光、肥料都受直接的監控，便會降低吃到農藥或其他殘留化學藥物的風險。好處還不只如此，擁有自己的植物盆不僅美化了環境，還能節省出外買菜的時間與金錢。考慮到都市人生活的繁忙，或許沒有時間親自澆水、施肥，納入了現實因素後，我們希望設計出一臺自動蔬菜養殖機，透過電腦軟件與定時器、量測儀的結合，讓使用者不必全程參與植物照護，可以簡單方便的讀取液晶顯示器的數值，了解植物狀況。

貳、研究目的

一、利用 Arduino 設計出自動水耕蔬菜養殖機，並實際種植水耕蔬菜(小白菜)。

二、比較自種水耕蔬菜與市售有機蔬菜、一般蔬菜的差別。

三、比較自製養殖機與市售養殖機之優缺差異。

四、自製養殖機收成耗費成本之估算。

參、研究設備及器材

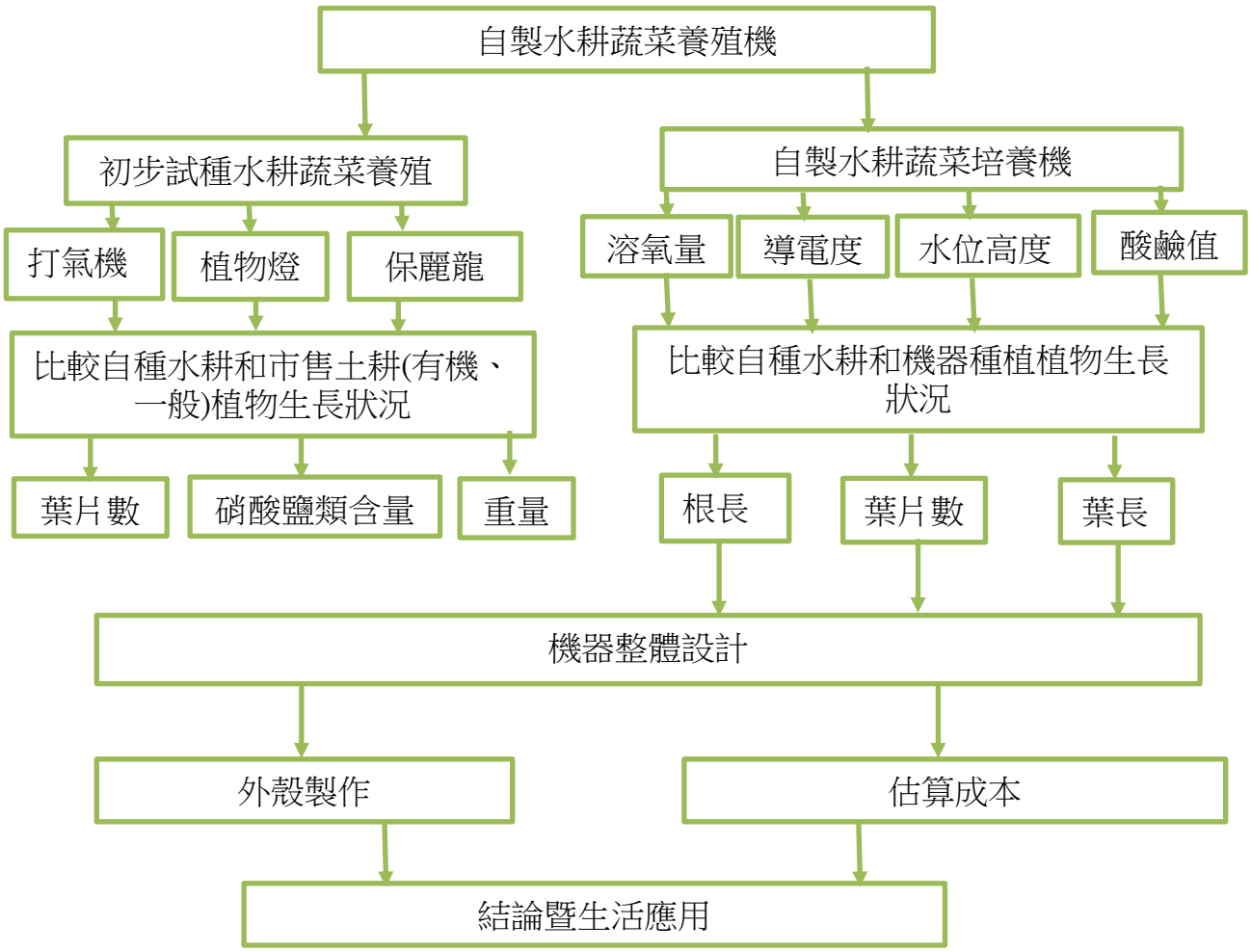
表(一) 本研究設備及器材

			
延長線	LED 多頭植物燈	保麗龍板	水管
			
營養液	打氣機	收納箱	沉水馬達
			
發泡棉	電池和電池盒	金鋼砂氣泡石	水質測試劑
			
電線	鋼釘	PH 感測器	水溫感測器

			
EC 計	Arduino	鐵架	溶氧量感測器

肆、研究過程或方法

表(二) 本研究流程



一、水耕蔬菜(小白菜)的種植

如上述所說，我們偏重收成效率及簡易性，並考慮到季節因素，我們選擇了水耕小白菜作為實驗主體。小白菜性喜涼爽，不耐熱，生長適溫為攝氏 15 度到 20 度，充足陽光利於生長，光線過熱將使植株虛弱，通常種植三至四周即可收成，也能適應水耕。比較水耕與土耕，水耕比較容易控制植物吸收的養分，需留意營養液成分標示即可，可減少蚊蟲侵害。至於時效部分，水耕植物根部吸收營養是直接而連續的，可使養殖時間縮短。

(一) 蔬菜種子育苗

圖(一) 小白菜育苗裝置照片

圖(二) 小白菜育苗裝置照片

本研究購買蔬菜種子後，直接於本裝置中育苗。我們使用海綿育苗，將種子至於海綿內，並浸泡在營養液中，前 24 小時用黑布遮擋，避免種子照射到光線，使其更容易發芽。經過約一天後，大約一半的種子成功發芽，這時，我們加入植物燈，適時補充光線，再兩天後，小白菜苗順利成長，便可以進行接下來的實驗。

(二) 第一代：純日照、發泡煉石、大陸妹和小白菜並用

圖(三) 第一代小白菜裝置

圖(四) 第一代大陸妹裝置

我們的實驗在冬天，一開始沒有使用植物燈，而是將裝置架設在容易曬到陽光的地方，但是發現小白菜長不大、植株軟爛，但我們以為是植物本身問題，所以另外嘗試種植大陸妹，希望能代替小白菜。

1.裝置架設步驟：

(1)準備 20 株小白菜(大陸妹)苗，泡水仔細洗去其根部土壤。

(2)用美工刀割出約 4×3×0.5 公分的發泡棉，中間割出一個小洞，預留可使植物穿過的空間。泡棉是為了使葉片可與水面隔離，補足發泡煉石的缺漏。

(3)準備兩個大收納箱，加水五公升，用奇異筆註記水面高度。加入葉肥 2 瓶蓋半，再加入發泡煉石，使其整體高度約至收納箱一半。放入植株，根部泡水，莖與葉均遠離水面。

(4)為了保持水中溶氧量足夠，我們加入 2 台打氣機，管子伸入水面下。

2.結果：

架設完成後，我們每周維持比例換水、加養液一次，並每天輪流量每棵植株的根長、葉長，並補足蒸發的水量。然而，不論是大陸妹或是小白菜，都一天比一天虛弱，最後全數死亡，檢討了第二代的問題後，我們試種了第三代。

(二)第二代成功經驗:打氣機、植物燈、保麗龍板、新養液

	
圖(五) 第二代小白菜與打氣機	圖(六) 第二代小白菜與植物燈

根據第一代失敗的經驗，我們發現即使有加入發泡棉與發泡煉石，植物的莖和葉仍會不小心泡到水，並因植株軟爛而死亡。為了解決泡水的問題，我們改以有切孔的保麗龍板，徹底隔絕植株除根部以外與水的接觸。另外，我們也察覺架設裝置的地點日照不足，所以加入多頭式植物燈，使植株得以挺立。最後，第一代添加的葉肥

無法提供植物所需的全部微量元素，因此我們改以新養液代替舊有的加在水中。

1.裝置架設步驟：

(1)如上述步驟(1)，準備 20 棵小白菜苗。

(2)用美工刀割出兩片 48×37.5 的保麗龍板，再在每片上割出 10 個 3×3 的小洞，並如上述步驟(2)，裁出 20 片發泡棉。

(3)準備兩個大收納箱，加水五公升，用奇異筆註記水面高度。加入葉肥 2 瓶蓋半，並加入 20 毫升新養液，將植株插入保麗龍板小洞中，再將保麗龍板蓋在收納箱上，並確保植株根部與水接觸。

(4)為了保持水中溶氧量足夠，我們加入各 4 台打氣機，管子伸入水面下。

(5)在收納箱附近裝設植物燈，並設定定時開關。

2.結果：

同樣維持一周加水、換養液一次，植物燈則是以開關時數比 1：1 滿足植物日照需求，並每天持續量測根長、葉長，也補充蒸發的水量。兩周後，平均葉長從原本的 4.29 公分成長至 8.34 公分，根長也從原來的 3.8 公分成長至 9.41 公分，最大的植株甚至根與葉都超過 15 公分，試種水耕蔬菜成功。

二、自製自動水耕蔬菜養殖機

(一)水耕蔬菜養殖機

我們選擇用 Arduino UNO 板，因為較容易上手，許多配件都與 Arduino UNO 板相容。我們設定 **pH 值、EC 值、溶氧量、水位偵測**植物生長環境，**當環境數值有所變動，則啟動馬達，做出相對回應**。同時，我們**固定植物燈照射時間**。

(二)儀器介紹

1.Arduino UNO 板：

我們選擇用 Arduino UNO 板作為控制面板。因為 Arduino UNO 彈性較大，入手門檻低、價格低，且有豐富的配件及程式碼可以參考。

2.PH 值感測器：

水耕營養液必須維持 pH 值穩定,以確保營養液元素維持在可被利用之離子形態。作物吸收養分時,陰、陽離子吸收不同,會造成 pH 值變化,就胡瓜與萵苣而言,為優先吸收銨態氮之作物,pH 就降低， 反之如白菜、菠菜等優先吸收硝酸態氮者,容易使 pH 值升高。一般 pH 值適合範圍為 5.5~6.5， pH 之高低對作物吸收營養有很大關係,pH 低於 5.5 時,鈣、鎂、鉬、硫、 磷等元素較難由根部吸收。若 pH 高於 6.5,鐵、錳、硼、銅與鋅較以難 吸收。pH 值為 7.0 以上,鐵元素在養液中容易形成氫氧化鐵($Fe(OH)_3$)沉澱而呈現缺乏狀態,甚至高於 8.0,則會出現缺錳及磷等生理障礙。因此,**若 pH 值不在合適範圍內,就代表此時養殖箱中的環境已不適合植物生長,這時,我們的裝置會啟動馬達換水,自動將原本養殖箱中的水抽出,並將新的調配好的營養液注入養殖箱中。**

3.EC 值感測器與溫度感測器:

電導度是指水中解離性無機鹽類之總和,而陽離子(鈣、鎂、鈉、鉀、 鐵、鋁等)及陰離子(碳酸鹽、硫酸鹽、氯、硝酸鹽)也是水中金屬鹽分濃度的另一項呈現指標。在作物生長過程中,電導度所產生之滲透壓會影響作物之水分吸收能力。水耕營養液之 EC 值會隨著水質變化而有差異,營養液之化學肥料一旦加入水中,均以離子形態存在,而營養液的離子濃度與 EC 值呈高度相關性,且隨著不同元素及不同量的組成而各有其最適合的值。作物所需之營養液配方一但決定後,可依配方中每一元素克當量之總和換算出 該配方之理論 EC 值,同時配合穩定的 pH 值,可使作物正常發育。因為溫度高低會影響 EC 值,所以 EC 值感測器會配合溫度感測器是來修正 EC 值。不同植物,需要不同的生長環境,**如萵苣適合的 EC 值約為 0.8,茼蒿適合的 EC 值約為 2.0,菠菜適合的 EC 值約為 1.1,小白菜適合的 EC 值為 1.3 多吸收氮量,以供生長所需。營養液經植物吸收後 EC 值會逐漸下降,當 EC 值降至 1.35 (姆歐/公分,mS/cm)以下時,裝置馬達會自動啟動,補充新的營養液使保持最適合的濃度。**

4.自製水位計:

水位如果太低,植物的根吸不到足夠的水分,會影響生長,導致植物倒塌,所以我們以水位計監控養殖箱的水位,當水位過低時,將啟動馬達補水。然而經過我

們的測試，**發現市售的水位計會被腐蝕，無法長時間放置在水中**。因此，我們自製水位計，用兩根鋼釘作為感測器，當水位過低時，鋼釘間形成斷路，Arduino 所讀取到的電壓即是外接的電壓，相對為高電壓，此時就啟動馬達加水 8 秒(100 毫升)。Arduino 持續讀取水位計，**直到水位回到正常值時，鋼釘尖端浸到水中，鋼釘間形成通路，所讀取到的電壓相對較低，就不需再啟動馬達。**

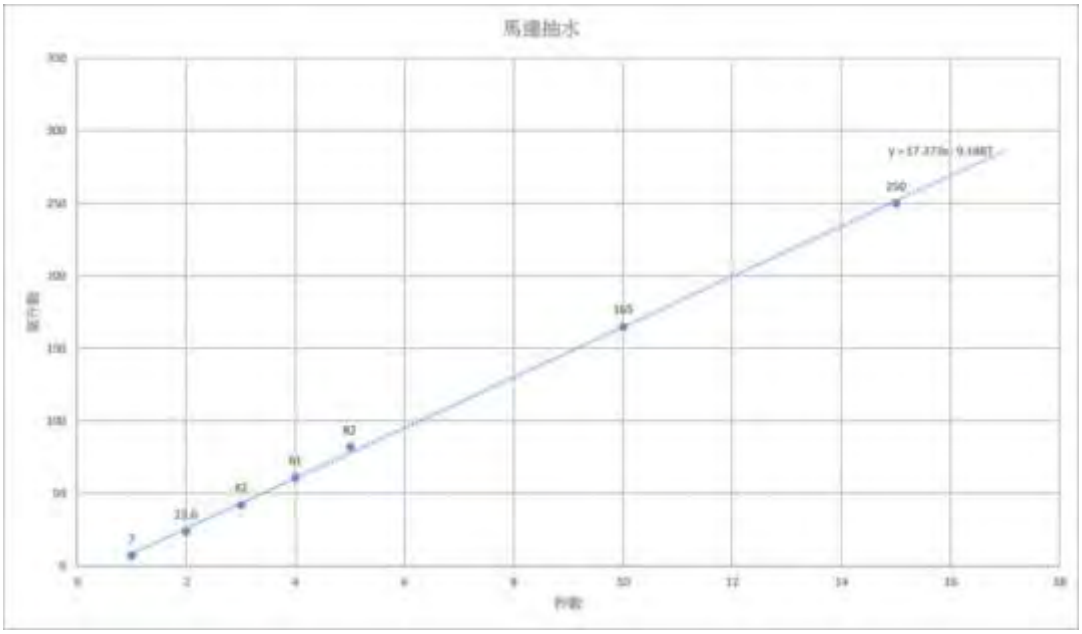
<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>圖(七) 被腐蝕的市售水位計</td><td>圖(八) 自製水位計</td></tr>
</table>

5.溶氧量感測器及打氣機：

因為所有植物的根，都必須有氧氣。在水質管理實務上，溶氧量被認知為是判斷水質好壞之主要指標。一般而言，濃度越高代表水質狀況越好。水中之飽和溶氧量受水溫以及水中含有雜質量之影響，水溫越高飽和溶氧量濃度就越低。當水中營養液之溶氧量為 5 ppm 以上，就不會有根腐病的問題，**若在於 2 ppm 以下，根系之生理營養不良，養分吸收少，尤其磷、鉀與錳元素明顯減少。若溶氧不足，LED 會亮起黃燈，提醒使用者開啟打氣機。**

6.打氣馬達：

當 pH 值、EC 值、水位不在正常範圍內時，啟動馬達直到數據回到正常值。因為水管的形狀，長度會影響馬達所抽的水量，所以我們先將水管固定，再測量秒數與抽水量的關係圖，進而訂定啟動馬達的秒數，根**據實測的結果，馬達每秒鐘的抽水量大約為 17(毫升，ml)。**



圖(九) 馬達抽水秒數與毫升數關係圖

7.LED(Light Emitting Diode，發光二極體)：

當感測器所測試到的數據有異常時，LED 燈會亮起紅燈，直到數據恢復至正常值時，LED 燈才會熄滅。我們同時設定 LED 燈與馬達配合，當馬達運轉時，即代表有數值異常，LED 燈就會亮起。

圖(十) 數據皆在正常值

圖(十一) pH 值異常

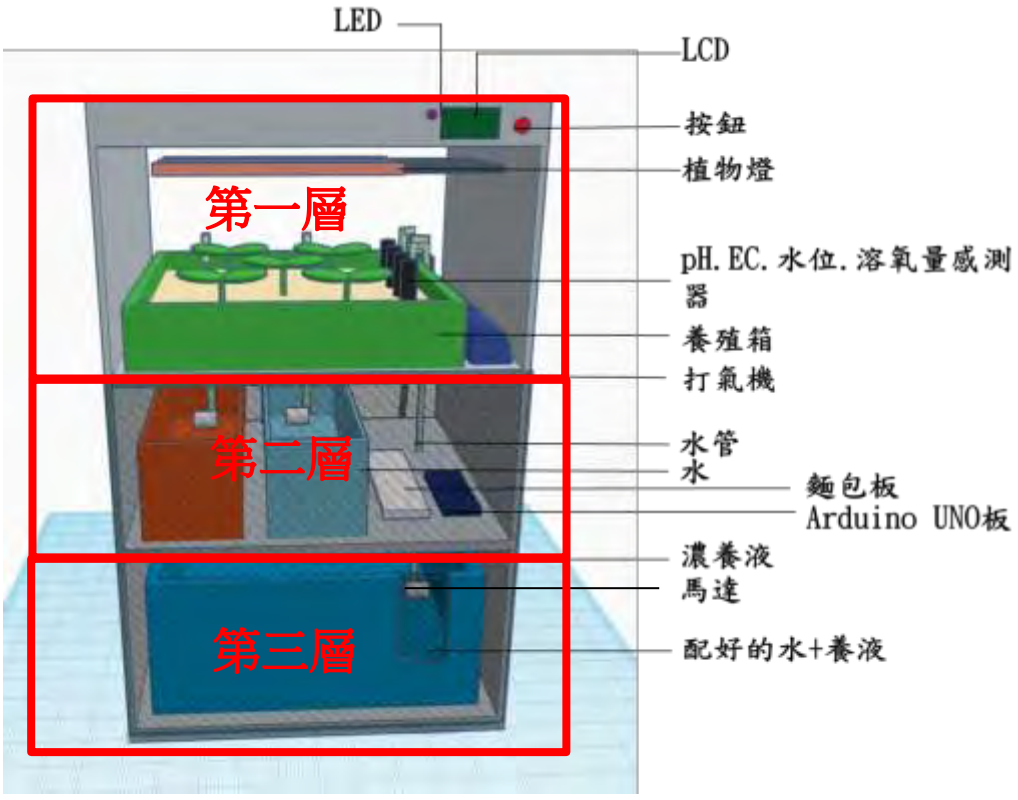
8.LCD(Liquid Crystal Display，液晶顯示器)：

為數據的顯示板，會**顯示**感測器所偵測到的 **pH、EC、水位計及溫度的數值**。

9.**裝置按鈕**：

因為不同種植物需要不同的生長環境，且同一種植物會因為季節的改變而有不同的需求，所以設定按鈕，**當按鈕被按下時，就能切換到其他模式，目前養殖機設計了 4 個模式，每個模式可根據需求預先輸入 pH 值範圍與 EC 值範圍。**

(三)3D 設計圖(本研究自行設計)



圖(十二) 水耕蔬菜養殖機設計圖(實際裝置如下圖九)

我們使用 Tinkercad 繪製了初步的設計圖，以有隔板的金屬外殼為支撐，共分為三層。每層之間以塑膠管子或電線聯繫，使其可以定時換水及補充養液，提供植物生長的穩定環境。

1.第一層：

這一層是植物**生長層**，綠色盆子內裝有 5 棵小白菜，如上述初代試種一樣，以保麗龍板固定。植物盆上方有植物燈，為了使每棵植物都能照到光，所以使用三頭植物燈。背**後連接一個打氣機，右邊放置 EC 計、pH 值計、自製水位計、溶氧計**，切開保麗龍板，使其能深入水中。金屬外殼上有 **LED、顯示器和按鈕，能提示使用者植物目前生長環境**。

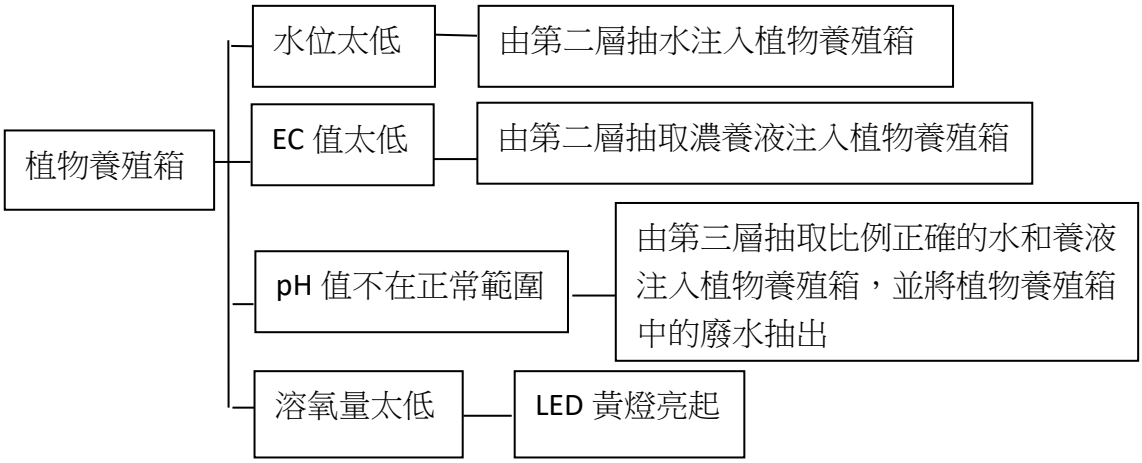
2.第二層：

這一層是主要**控制層，純水箱及營養液**皆放置在這一層，由 Arduino UNO 板控制，可以從最下層抽水供給上層植物、根據感測器數值的改變注入純水或養液，也可以回收植物的廢水流進廢水箱。

3.第三層：

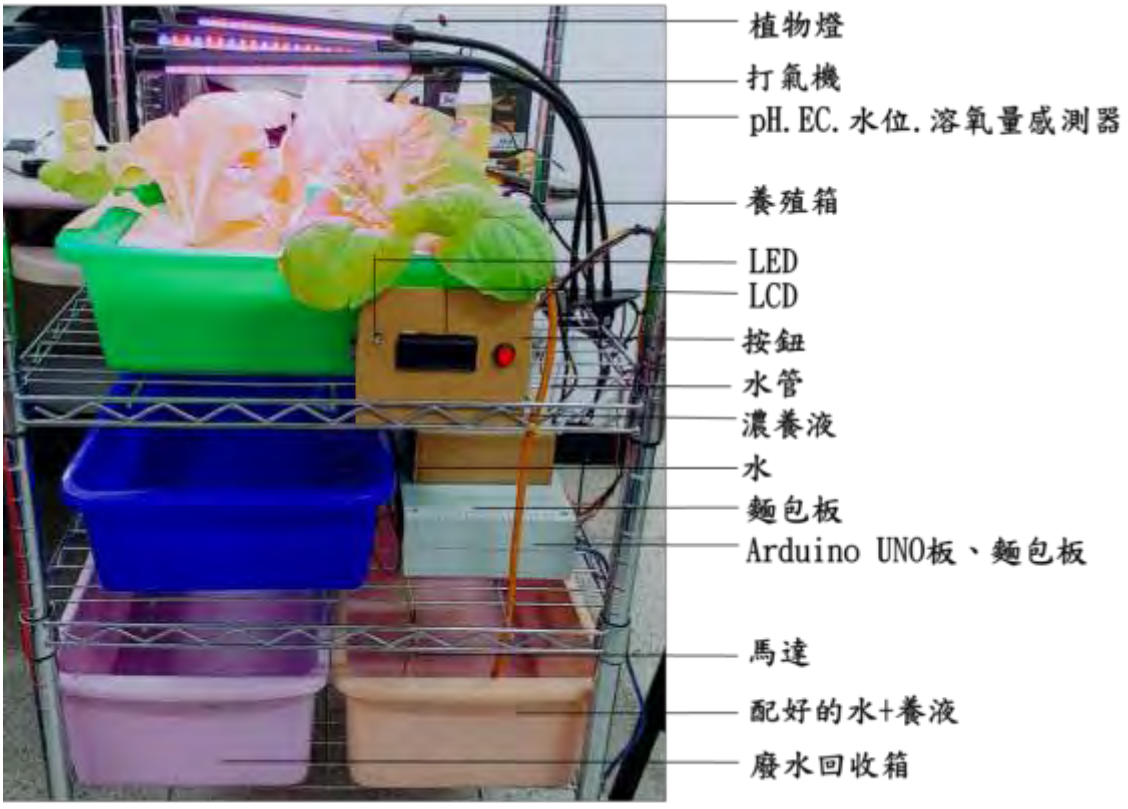
此層放置大水箱，水箱是調整好比例的水加養液，作為換水來源，其概念如下:

表(三) 植物養殖箱設計概念



三、實際機器製作

(一)第一代養殖機：



圖(十三) 第一代養殖機

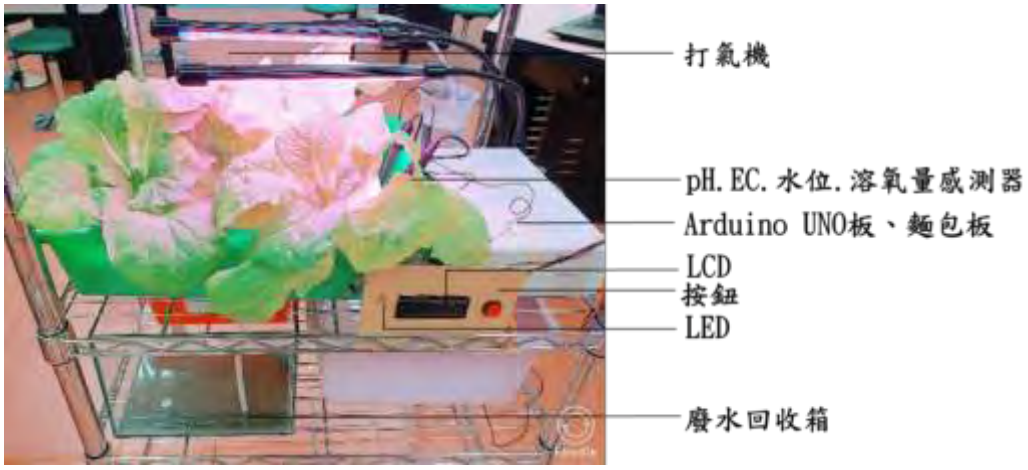
1.設計圖的修正：

原本我們預計將廢水直接排入排水孔，便沒有設計廢水箱擺放的空間，後來因為考慮到便利性的問題，所以我們將第三層隔開，分別放置廢水和配好的水+養液，這樣機器的位置就不受限制，可以依使用者的需要自行決定位置。

2.問題：

這個成品的體積仍然過於龐大，所以我們又再次改變了箱子的大小和擺放位置，**使養殖機的體積能夠再次縮小，做出第二代養殖機**。

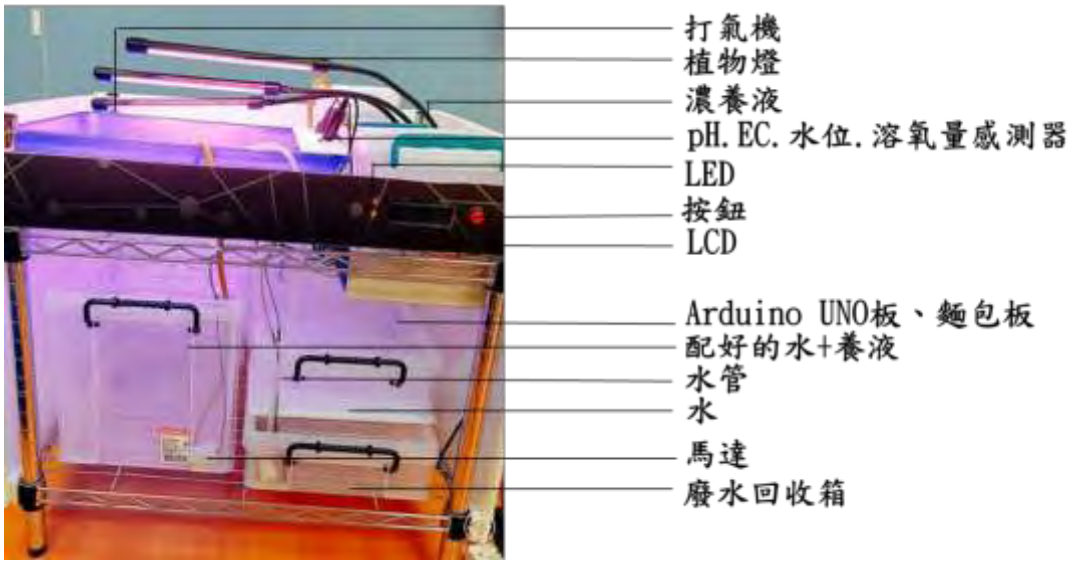
(二)第二代養殖機：



圖(十四) 第二代養殖機

基於體積過大的問題，我們改變了第一代養殖機的容器位置，製作出第二代養殖機。我們將原本的控制面板和濃養液瓶集中至第一層，廢水箱、換水箱及加水箱則集中至第二層。

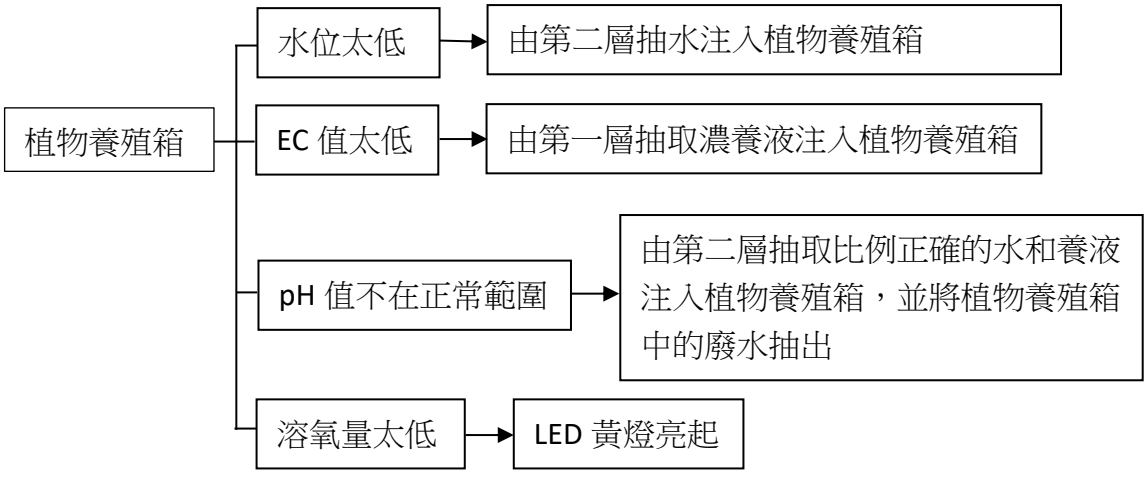
(三)第三代養殖機：



圖(十五) 第三代養殖機

我們在第二代的基礎上加裝門板、統一箱子格式，並將 Arduino 控制面板固定於第一層下方，有效節省空間，使機器整體更為小巧。同時，我們美化機器，使其外觀更為整齊。

表(四) 本研究整體運作模式(原理)



伍、研究結果

一、比較自種水耕蔬菜與市售有機蔬菜、一般蔬菜的差別。

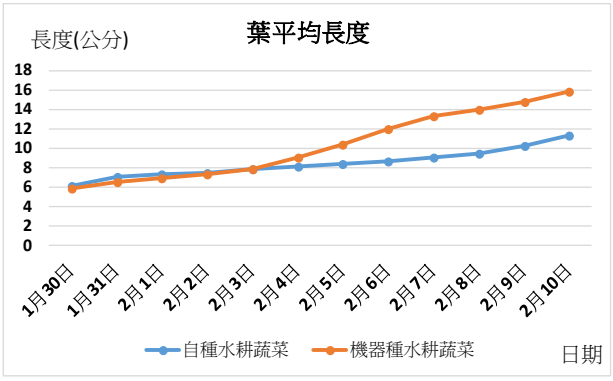
(一)自種水耕與市售蔬菜比較

表(五) 本研究水耕與市售蔬菜比較

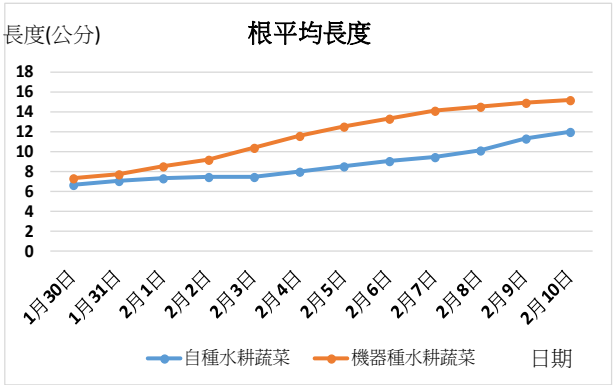
種類	自種水耕蔬菜	市售一般蔬菜	市售有機蔬菜
葉片數	16 片	9 片	10 片
重量(公克)	40.5	110.6	75.0
硝酸鹽測試			
程度	中等	最多	最少

由上表可知，**自種水耕蔬菜的葉片數最多**，其次是市售有機蔬菜，而市售一般蔬菜最少；重量市售一般蔬菜最重，其次是市售有機蔬菜，而自種蔬菜最輕；硝酸鹽含量市售一般蔬菜最多，**其次是自種水耕蔬菜**，市售有機蔬菜的最少。雖然自種水耕蔬菜葉片數較多，但在重量上仍無法媲美市售蔬菜；**硝酸鹽含量則較一般市售蔬菜低，**可降低食用風險，儘管其大於有機蔬菜，但仍不超過正常值每公升 2000 毫克。

(二)本研究機器成果與自種比較



圖(十二) 自種與機器葉的生長情形比較



圖(十三) 自種與機器根的生長情形比較

由上圖可知自種水耕蔬菜的葉在 12 天內，大約從 6 公分長到 11 公分，共長了 5 公分；機器種水耕蔬菜的葉在 12 天內，大約從 6 公分長到 16 公分，共長了 10 公分。所以自種水耕蔬菜葉的生長曲線較機器種水耕蔬菜的平緩。自種水耕蔬菜的根在 12 天內，大約從 7 公分長到 12 公分，共長了 5 公分；機器種水耕蔬菜的葉在 12 天內，大約從 7 公分長到 15 公分，共長了 8 公分。所以自種水耕蔬菜根的生長曲線較機器種水耕蔬菜的平緩。**可知用機器控制所種出的蔬菜，不論是葉還是根，生長幅度皆比自己手動控制的還要大**。

二、自製養殖機與市售養殖機之比較

(一) 自製養殖機機器成本

表(六) 本研究蔬菜養殖機成本

品項	價格(元)
溶氧計	6000
pH 計	1800
EC 計	2500
其他(註)	700
植物燈	600
架子	800
Arduino UNO 板	680
打氣機(台)	300
總價：13380 元	

(註):鋼釘、電阻、二極體、電池、LCD、LED、盒子、水管、保麗龍板

一般市售的水耕箱，手動水耕箱約 3000 至 4000 元、自動水耕種殖盆約 14000 元，看似價格低廉，然而即使是自動種植盆也只有光照及維持水循環的功能。若要使市售水耕箱能監測環境，必定會使用溶氧計、pH 計及 EC 計，其價格至少比原本多出 10000 元，再納入自動控制電路的成本，自動種殖盆售價將超過 24000 元，遠大於我們自

動水耕蔬菜養殖機的成本。**我們的儀器量測數據多元，不但可以監測環境，還能顯示數值跟自動加水及換水**。

三、自製水耕蔬菜養殖機收成耗費成本之估算

(一)收成一次耗費成本

表(七) 本研究蔬菜收成一次成本

品項	價格(元)
打氣機電費	5.3
植物燈電費	15.1
Arduino UNO 板電費	10
水費	0.13
養液費	35
總價：65.53 元	

1.電費：

(1)打氣機：

我們使用 3.5W 的打氣機，假設在一次生長周期三個禮拜內，都要開啟，則電費＝3.5×24(小時)×21(天)/1000＝1.764 度，又每度電費是 3 元，所以打氣機最高電費約 5.3 元。

(2)植物燈：

我們使用 20W 的植物燈，在一次生長周期三個禮拜內，每天開起 12 小時，則電費＝20×12(小時)×21(天)/1000＝5.04 度，所以植物燈電費約 15.1 元。

(3) Arduino UNO 板：

無法得知整體機器功率為何，所以我們大略估計為 10 元。

2.水費：

一開始加水 5 公升+換水 2 次 10 公升+水位計感應補水約 3 公升＝18 公升。一度水為 1000 公升，所以我們用了 0.018 度，一度水價 7.35 元，故收成一次水費成本 0.13 元。

3.養液費：

一罐 350 毫升、200 元，在一次生長周期三個禮拜內，約使用 60 毫升，也就是大約 35 元。

(二)平均成本

我們一次種 5 棵小白菜，三個禮拜即可收成，預估可以使用五年，約可以收成 87 次，也就是 435 棵小白菜。13380 / 435＝30.7 元，再加上收成一次單棵成本約 65 / 5＝13，所以單棵總成本為 43.7 元。由於整體成本過高，我們提出以下改善方式：若是能將此養殖機應用在大型養殖場，可大範圍監控植物生長環境、壓縮成本，假設一次種 50 棵小白菜，因機器本身成本不會改變，故單棵耗費機器成本降低為 3.07 元，電費與水費增加不多，主要成本在養液費，但如果使大量購買養液，成本不會超過 50 元，故水費加電費家養液費合計約 100 元，收成一次單棵成本約 100 元/ 50 ＝2 元，所以單棵總成本約 5 元。至於一般住戶，也可以增加單次種植棵數，如此能使此機器經濟價值提升。

陸、討論

一、自行設計水位計

測試水位計時，我們發現**市售的水位計**放在水裡一個小時以上時，**水位計會被嚴重腐蝕**，當水位計被腐蝕，便無法精準地顯示所偵測到的數值，經過多次的測試，發現無論哪一個廠牌的水位計都有同樣的狀況，無法長時間放置在水中，以達成我們長時間以及即時監控的目的。因此，我們**自製水位計，用兩根鋼釘作為感測器**，利用**導電的原理**，製作一個迴路，因為水中的離子可以導電，所以若是**鋼釘浸到液面，會因為水溶液電阻小而使讀取到的電壓相對較低，如果鋼釘未浸到液面，則形成斷路，所讀取到的電壓即外接的電壓，相對較高**，如此我們便能清楚的從**讀取到的電壓了解水位狀況**。利用鋼釘製作水位計，不僅不會被腐蝕，能長時間放置在潮濕的環境，更能滿足我們偵測水位的需求。

二、市售水耕蔬菜養殖機與自製水耕蔬菜養殖機功能上的差別

因為參考市售水耕蔬菜養殖機後，我們認為市售得水耕蔬菜養殖機的功能太少，不但需要自動換水以及加營養液，更無法即時監控養殖箱內的水質來提供植物最良好的生長環境。我們考量影響植物生長的因素，修正一般市售水耕蔬菜養殖機的缺點，再自製水耕蔬菜養殖機。我們**自製的機器能提供植物良好生長環境，亦能及時監控水質狀況，並透過 LED 燈以及顯示器的顯示，以最清楚及簡便的方式讓使用者了解水質的狀況。整體而言，自製的水耕蔬菜養殖機操作簡單，考量到的因素多，能提供植物最好的生長環境，改良並補足市售水耕蔬菜機缺漏**。

三、本研究程式碼的撰寫

我們利用程式碼使 Arduino UNO 板控制養殖機內的感測器。因為程式太過複雜，經由老師的協助，經過多次的修改找出程式碼邏輯和編譯的的錯誤，順利寫出基本的程式碼，但因感測器並非只需程式碼即能正常運作，又經過多次的校準，畫出感測器所讀取到的數據的函數圖形，進一步校準感測器的數值，使感測器能精準地讀取到正確的數據。由於 pH 感測器電壓不穩，所以我們控制 pH 感測器，使其每測量四十次後平均，並讀取數

值，每 0.8 秒印出數值，因為儀器需要通電一段時間後數值才會穩定，所以我們捨棄前十次的數據，經過十次的讀取後，數值才顯示在顯示器上，並決定是否啟動馬達，如此便解決了數據不準確的問題。為使所讀取到的數據能清楚的顯示讓使用者方便檢視，我們又進一步透過撰寫程式，所讀取到的數據顯示在顯示器上，使數據並非只能透過電腦才能檢視。我們期望使用者能以最簡單的方式了解水質狀況，所以以**程式控制 LED 燈，使其在感測數值異常時即馬達運轉時亮起**。而我們設定的按鈕是能改變數值的參數，使不同植物在不同條件下能使用不同的參數，我們利用**餘數的方式**，設定四種植物生長的參數，當按鈕被按下的次數除以四餘零時，為第一組參數，餘一時，為第二組參數，依此類推。經過多次的修正，程式碼已能精準的控制自製的水耕蔬菜養殖機，使其正常運作。**本系統還可擴充到 n 種植物生長的參數，只要把四改為 n 即可**。(**n=正整數**)

柒、結論

一、利用 Arduino 設計出自動水耕蔬菜養殖機，並實際種植水耕蔬菜

水耕蔬菜養殖機能有效節省人力成本，且機器自動種植的蔬菜生長較迅速。為達成此目的，我們使用 Arduino UNO 板作為控制面板，**在室溫下**以影響水質主要的三個變因做為監測對象：**導電度計(EC)、酸鹼值計(pH)、溶氧量計(DO)，並自製水位計，自動控制植物最佳生長環境**。此外，我們調整機台空間配置，並美化機器整體外觀，使**機器可架設在一般室內空間**。

二、自種水耕蔬菜與市售有機蔬菜、一般蔬菜的差別

水耕蔬菜品質穩定、易於種植，有大量發展空間。我們比較市售與水耕種植的差異，發現水耕種植蔬菜較一般市售蔬菜小，但是**硝酸鹽含量較低**，對人體帶來傷害較小，而有機蔬菜硝酸鹽含量最低。接著再跟機器自動種植的差異，**機器自動種植能使蔬菜長得更迅速**，較無病蟲災害，使得**蔬菜生長更為迅速且無須施用農藥**。

三、比較自製養殖機與市售養殖機之優缺差異

本研究自行開發設計的水耕蔬菜養殖機，與一般市售比較，多了能自動監控植物生長環境，並透過**自動換水、加水、加養液等方式做出相對回應，讓使用者只需負責採收的部分，節省金錢及人力。**因監測變因較多，功能性也較好。在考量相同功能的情況下，本研究整體成本可較一般市售價格節省快一半。可藉由擴大種植規模改善此問題；後我們調整機台空間配置，使機器更為**小巧美觀**，期望在實際應用上能擴大規模，使**單次種植棵數增加，降低成本、節省人力資源**。

四、自製養殖機收成耗費成本之估算

我們對養殖成本進行估算，在一次只種五顆小白菜的小規模種植情況下，單棵總成本偏高，但可藉由**擴大種植規模改善此問題**，當單次種植棵數到達 50 顆以上時，單顆成本將不到五元。

未來展望

一、自製溶氧量感測器：

由於溶氧量感測器是一個精密的儀器，所以價格較高且不易取得，事實上，溶氧量感測器占了本機器一半的成本。我們期望**自製溶氧量感測器**，製造出低價格的替代

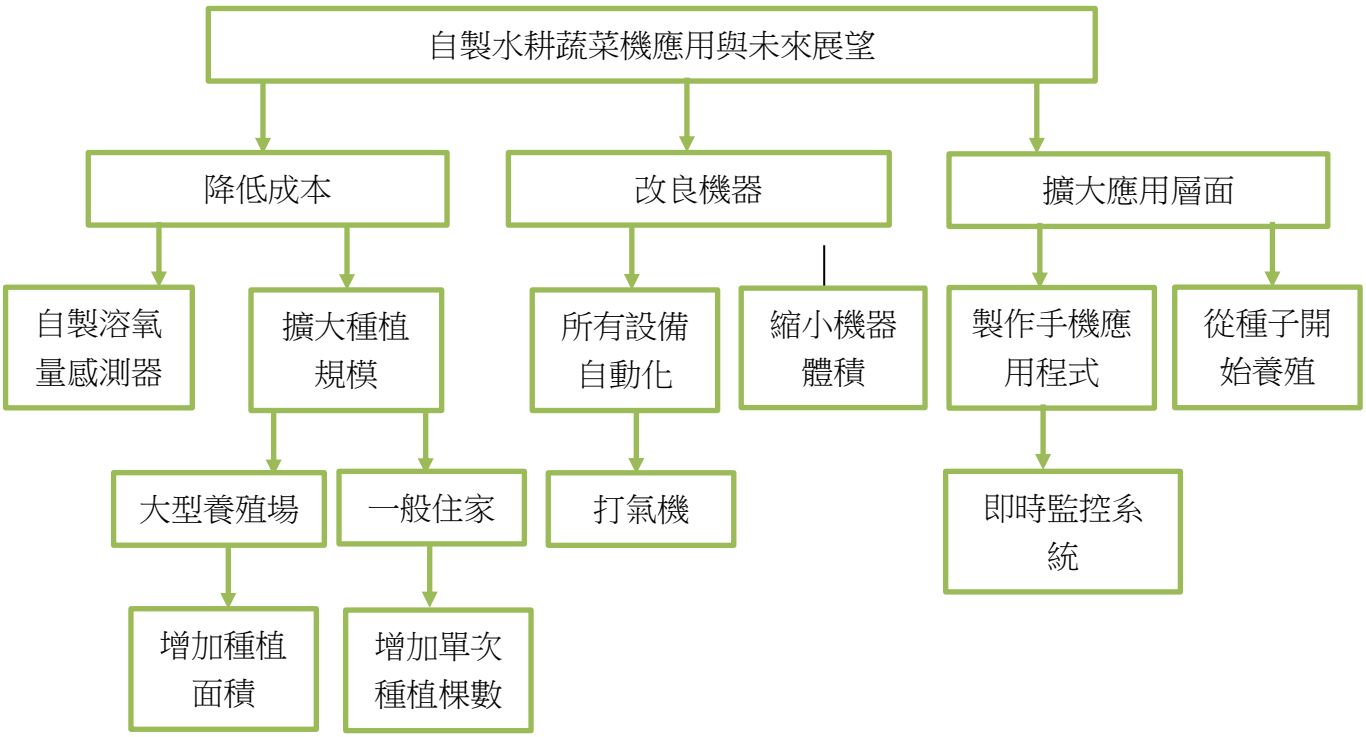
二、所有設備自動化：

目前我們尚未將打氣機自動化，我們希望使**打氣機與溶氧量感測器配合，當偵測到溶氧量不足時，再啟動打氣機，如此，機器能完全自動化**，讓使用者更方便操作，同時也能節省電力的消耗。

三、製作手機應用程式：

我們期望可以**做成手機應用程式(APP)，記錄植物生長趨勢，以達成即時監控植物生長狀況的目的**。

表(八) 本研究應用與未來展望



捌、參考資料及其他

一、林勛雄，萵苣栽培與管理

二、張珈錡、廖玉珠，植物對硝酸態氮和銨態氮之吸收與利用。

三、黃秉鈞，1845，生生不息的再生能源，台灣大學機械系，臺北。

四、田霄鴻、王朝輝、李生秀，1999，不同氮素型態及配比對蔬菜生長及品質的影響。

五、河村麴子(2013)。無農藥水耕栽培。新北市:教育之友。

六、蔡尚光(2013)。水耕栽培的魅力。新北市:淑馨。

七、尤崇魁(1900)。水耕栽培實務。新北市:淑馨(代理)。

八、高德錚，水耕栽培－精緻蔬菜生產技術之開發，台中區農推專訊 56 期，1986。

附錄

Arduino 水耕蔬菜養殖機程式碼

```
#include <OneWire.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <avr/pgmspace.h>
#include <EEPROM.h>

// 設定 LCD I2C 位址
// Set the pins on the I2C chip used for LCD connections:
// addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(0x3F, 2, 1, 0, 4, 5, 6, 7, 3,
POSITIVE);

#define StartConvert 0
#define ReadTemperature 1

int modeindex=0;
int loopcount=20;
float phmin, phmax, ecmin, ecmax;
const float ph1min=5.5;
const float ph1max=6.5;
const float ph2min=1;
const float ph2max=14;
const float ph3min=1;
const float ph3max=14;
const float ph4min=1;
const float ph4max=14;
const float ec1min=2;
const float ec1max=3.5;
const float ec2min=0;
const float ec2max=14;
const float ec3min=0;
const float ec3max=14;
const float ec4min=0;
const float ec4max=14;

const byte numReadings = 20;          //the number of
sample times
byte ECsensorPin = A1;    //EC Meter analog output,pin
on analog 1
byte DS18B20_Pin = 2; //DS18B20 signal, pin on digital 2
unsigned int
AnalogSampleInterval=25,printInterval=800,tempSampl
eInterval=850;    //analog sample interval;serial print
interval;temperature sample interval
unsigned int readings[numReadings];           // the
readings from the analog input
byte index = 0;                               // the index of the
current reading
unsigned long AnalogValueTotal = 0;
// the running total
unsigned int AnalogAverage = 0,averageVoltage=0;
// the average
unsigned long
AnalogSampleTime,printTime,tempSampleTime;
float temperature,ECcurrent;

#define SensorPin A0                  //pH meter Analog
output to Arduino Analog Input 0
#define Offset 14                     //deviation compensate
// #define LED 13
#define phsamplingInterval 20
#define phprintInterval 800
#define ArrayLenth    40        //times of collection
int pHArray[ArrayLenth];      //Store the average value of
the sensor feedback
int pHArrayIndex=0;
static unsigned long phsamplingTime = millis();
static unsigned long phprintTime = millis();
static float pHValue,phvoltage;
static float pHValue2=7;

const int watersensorPin= 2; //sensor pin connected to
analog pin A2
#define liprintInterval 800
#define lisamplingInterval 20
int liArray[ArrayLenth];      //Store the average value of
the sensor feedback
int liArrayIndex=0;
static unsigned long liprintTime = millis();
static unsigned long lisamplingTime = millis();
int liquid_level=500;
int liquidindex=0;

#define motorcheckInterval 800
static unsigned long motorcheckTime = millis();
const int motorPin3 = 3;
const int motorPin4 = 4;
const int motorPin5 = 5;
const int motorPin6 = 6;
int loop_c = 0;
int sw1value;
#define swpin 12
#define errorled 9

String slcd1,slcd2;

#define DoSensorPin    A3        //dissolved oxygen
sensor analog output pin to arduino mainboard
#define VREF 5000        //for arduino uno, the ADC
reference is the AVCC, that is 5000mV(TYP)
float doValue;            //current dissolved oxygen value,
unit; mg/L

#define EEPROM_write(address, p) {int i = 0; byte *pp =
(byte*)&(p);for(; i < sizeof(p); i++)
EEPROM.write(address+i, pp[i]);}
#define EEPROM_read(address, p)    {int i = 0; byte *pp =
(byte*)&(p);for(; i < sizeof(p); i++)
pp[i]=EEPROM.read(address+i);}

#define ReceivedBufferLength 20
char receivedBuffer[ReceivedBufferLength+1];       //
store the serial command
byte receivedBufferIndex = 0;

#define SCOUNT    30                  // sum of sample
point
int analogBuffer[SCOUNT];       //store the analog value
in the array, readed from ADC
int analogBufferTemp[SCOUNT];
int analogBufferIndex = 0,copyIndex = 0;

#define SaturationDoVoltageAddress 12
//the address of the Saturation Oxygen voltage stored in
the EEPROM
#define SaturationDoTemperatureAddress 16
//the address of the Saturation Oxygen temperature
stored in the EEPROM
float SaturationDoVoltage,SaturationDoTemperature;
float doaverageVoltage;

const float SaturationValueTab[41] PROGMEM =
```

```
{          //saturation dissolved oxygen concentrations at
various temperatures
14.46, 14.22, 13.82, 13.44, 13.09,
12.74, 12.42, 12.11, 11.81, 11.53,
11.26, 11.01, 10.77, 10.53, 10.30,
10.08, 9.86,   9.66,   9.46,   9.27,
9.08,   8.90,   8.73,   8.57,   8.41,
8.25,   8.11,   7.96,   7.82,   7.69,
7.56,   7.43,   7.30,   7.18,   7.07,
6.95,   6.84,   6.73,   6.63,   6.53,
6.41,
};

//Temperature chip i/o
OneWire ds(DS18B20_Pin);  // on digital pin 2

void setup() {
  // put your setup code here, to run once:
 // initialize serial communication with computer:
  Serial.begin(9600);
  lcd.begin(16,2);   // 初始化 LCD，一行 16 的字元，
共 2 行，預設開啟背光
  // initialize all the readings to 0:
  for (byte thisReading = 0; thisReading < numReadings;
thisReading++)
      readings[thisReading] = 0;
  AnalogSampleTime=millis();
  printTime=millis();
  tempSampleTime=millis();

  Serial.println("pH meter experiment!");       //Test the
serial monitor

  pinMode(watersensorPin, INPUT);//the liquid level
sensor will be an input to the arduino

  pinMode(motorPin3,OUTPUT);
  pinMode(motorPin4,OUTPUT);
  pinMode(motorPin5,OUTPUT);
  pinMode(motorPin6,OUTPUT);
  pinMode(errorled,OUTPUT);
  pinMode(swpin,INPUT_PULLUP);
  pinMode(DoSensorPin,INPUT);
  readDoCharacteristicValues();          //read
Characteristic Values calibrated from the EEPROM
}

void loop() {
sw1value=digitalRead(swpin);
  if (sw1value==LOW)
  {
     delay(50);
     while (digitalRead(swpin)==LOW)
     ;
     modeindex++;
     modeindex=modeindex % 4;
     Serial.println(modeindex);
     lcd.setCursor(0,0); // 設定游標位置在第一行行首
     lcd.print(modeindex);
  }

if (modeindex==0)
  {
    phmin=ph1min;
    phmax=ph1max;
    ecmin=ec1min;
    ecmax=ec1max;
  }
else if (modeindex==1)
  {
    phmin=ph2min;
    phmax=ph2max;
    ecmin=ec2min;
    ecmax=ec2max;
  }
else if (modeindex==2)
  {
    phmin=ph3min;
    phmax=ph3max;
    ecmin=ec3min;
    ecmax=ec3max;
  }
else   if (modeindex==3)
  {
    phmin=ph4min;
    phmax=ph4max;
    ecmin=ec4min;
    ecmax=ec4max;
  }

read_ph();
read_ec();
read_liquid();
read_do();

if(millis() - motorcheckTime > motorcheckInterval)
//Every 800 milliseconds, print a numerical
  {
    if (loop_c > loopcount)
    {

  lcd.setCursor(0,0); // 設定游標位置在第一行行首
  slcd1=String("")+modeindex+"
PH:"+twodigit(pHValue2) + " EC:" + twodigit(ECcurrent);
  lcd.print(slcd1);
  lcd.setCursor(0, 1); // 設定游標位置在第二行行首
  slcd2=String("")+"Lev:" + liquid_level+"
"+twodigit(temperature)+"C";
//twodigit(liquid_level);
  lcd.print(slcd2);

  if (pHValue2<phmin || pHValue2>phmax)
  {
     digitalWrite(errorled,HIGH);
     motor1start();
    delay(1000);
    motor1stop();

    motor4start();
    delay(1000);
    motor4stop();

     }

 else if (ECcurrent<ecmin)
  {
     digitalWrite(errorled,HIGH);
     motor2start();
    delay(1000);
    motor2stop();
     }

 else if (liquid_level>=800)
  {
     digitalWrite(errorled,HIGH);
     motor3start();
    delay(1000);
    motor3stop();
     }
     digitalWrite(errorled,LOW);
     loop_c = loopcount;
    }
    motorcheckTime=millis();
```

```
loop_c ++;
   }
}

void read_ec(void) {
   /*
     Every once in a while,sample the analog value and
calculate the average.
   */
   if(millis()-AnalogSampleTime>=AnalogSampleInterval)
   {
      AnalogSampleTime=millis();
       // subtract the last reading:
      AnalogValueTotal = AnalogValueTotal -
readings[index];
      // read from the sensor:
      readings[index] = analogRead(ECsensorPin);
      // add the reading to the total:
      AnalogValueTotal = AnalogValueTotal +
readings[index];
      // advance to the next position in the array:
      index = index + 1;
      // if we're at the end of the array...
      if (index >= numReadings)
      // ...wrap around to the beginning:
      index = 0;
      // calculate the average:
      AnalogAverage = AnalogValueTotal / numReadings;
   }
   /*
     Every once in a while,MCU read the temperature
from the DS18B20 and then let the DS18B20 start the
convert.
     Attention:The interval between start the convert and
read the temperature should be greater than 750
millisecond,or the temperature is not accurate!
   */
    if(millis()-tempSampleTime>=tempSampleInterval)
   {
      tempSampleTime=millis();
      temperature = TempProcess(ReadTemperature);
// read the current temperature from the    DS18B20
      TempProcess(StartConvert);
//after the reading,start the convert for next reading
   }
    /*
     Every once in a while,print the information on the
serial monitor.
   */
   if(millis()-printTime>=printInterval)
   {
      printTime=millis();
      averageVoltage=AnalogAverage*(float)5000/1024;
      Serial.print("Analog value:");
      Serial.print(AnalogAverage);      //analog
average,from 0 to 1023
      Serial.print("       Voltage:");
      Serial.print(averageVoltage);    //millivolt
average,from 0mv to 4995mV
      Serial.print("mV       ");
      Serial.print("temp:");
      Serial.print(temperature);       //current
temperature
      Serial.print("^C         EC:");

      float
TempCoefficient=1.0+0.0185*(temperature-25.0);
//temperature compensation formula: fFinalResult(25^C)
= fFinalResult(current)/(1.0+0.0185*(fTP-25.0));
      float
CoefficientVolatge=(float)averageVoltage/TempCoefficie
nt;
      if(CoefficientVolatge<150)Serial.println("No
solution!");      //25^C 1413us/cm<-->about 216mv    if
the voltage(compensate)<150,that is <1ms/cm,out of
the range
      else if(CoefficientVolatge>3300)Serial.println("Out
of the range!");    //>20ms/cm,out of the range
      else
      {

if(CoefficientVolatge<=448)ECcurrent=6.84*CoefficientV
olatge-64.32;      //1ms/cm<EC<=3ms/cm
         else
if(CoefficientVolatge<=1457)ECcurrent=6.98*Coefficient
Volatge-127;    //3ms/cm<EC<=10ms/cm
         else ECcurrent=5.3*CoefficientVolatge+2278;
//10ms/cm<EC<20ms/cm
         ECcurrent/=1000;        //convert us/cm to ms/cm
         Serial.print(ECcurrent,2);    //two decimal
         Serial.println("ms/cm");
      }
   }

}

/*
ch=0,let the DS18B20 start the convert;ch=1,MCU read
the current temperature from the DS18B20.
*/
float TempProcess(bool ch)
{
   //returns the temperature from one DS18B20 in DEG
Celsius
   static byte data[12];
   static byte addr[8];
   static float TemperatureSum;
   if(!ch){
            if ( !ds.search(addr)) {
                Serial.println("no more sensors on
chain, reset search!");
                ds.reset_search();
                return 0;
            }
            if ( OneWire::crc8( addr, 7) != addr[7]) {
                Serial.println("CRC is not valid!");
                return 0;
            }
            if ( addr[0] != 0x10 && addr[0] != 0x28) {
                Serial.print("Device is not
recognized!");
                return 0;
            }
            ds.reset();
            ds.select(addr);
            ds.write(0x44,1); // start conversion, with
parasite power on at the end
   }
   else{
            byte present = ds.reset();
            ds.select(addr);
            ds.write(0xBE); // Read Scratchpad
            for (int i = 0; i < 9; i++) { // we need 9 bytes
               data[i] = ds.read();
            }
            ds.reset_search();
            byte MSB = data[1];
            byte LSB = data[0];
            float tempRead = ((MSB << 8) | LSB); //using
two's compliment
            TemperatureSum = tempRead / 16;
      }
            return TemperatureSum;
}
```

```
void read_ph() {
   {
      if(millis()-phsamplingTime > phsamplingInterval)
   {

pHArray[pHArrayIndex++]=analogRead(SensorPin);
         if(pHArrayIndex==ArrayLenth)pHArrayIndex=0;
         phvoltage = avergearray(pHArray,
ArrayLenth)*5.0/1024;
         pHValue = -3.53*phvoltage+Offset;
         phsamplingTime=millis();
   }
   if(millis() - phprintTime > phprintInterval)      //Every
800 milliseconds, print a numerical
   {
   Serial.print("Voltage:");
            Serial.print(phvoltage,2);
            Serial.print("        pH value: ");

   Serial.println(pHValue,2);
            phprintTime=millis();
            pHValue2=pHValue;
      }

}

}
double avergearray(int* arr, int number){
   int i;
   int max,min;
   double avg;
   long amount=0;
   if(number<=0){
      Serial.println("Error number for the array to
avraging!/n");
      return 0;
   }
   if(number<5){     //less than 5, calculated directly
statistics
      for(i=0;i<number;i++){
         amount+=arr[i];
      }
      avg = amount/number;
      return avg;
   }else{
      if(arr[0]<arr[1]){
         min = arr[0];max=arr[1];
      }
      else{
         min=arr[1];max=arr[0];
      }
      for(i=2;i<number;i++){
         if(arr[i]<min){
            amount+=min;              //arr<min
            min=arr[i];
         }else {
            if(arr[i]>max){
               amount+=max;       //arr>max
               max=arr[i];
            }else{
               amount+=arr[i]; //min<=arr<=max
            }
         }//if
      }//for
      avg = (double)amount/(number-2);
   }//if
   return avg;
}

void read_liquid() {
  if(millis()-lisamplingTime > lisamplingInterval)
   {

liArray[liArrayIndex++]=analogRead(watersensorPin);
         if(liArrayIndex==ArrayLenth)liArrayIndex=0;
         liquid_level = avergearray(liArray, ArrayLenth);

         lisamplingTime=millis();
   }

   if(millis() - liprintTime > liprintInterval)      //Every 800
milliseconds, print a numerical
   {
   Serial.print("liquid level:");
   Serial.println(liquid_level);//prints out liquid level
sensor reading
   liprintTime=millis();

   }
}

void read_do(void)
{
static unsigned long analogSampleTimepoint = millis();
     if(millis()-analogSampleTimepoint > 30U)
//every 30 milliseconds,read the analog value from the
ADC
     {
        analogSampleTimepoint = millis();
        analogBuffer[analogBufferIndex] =
analogRead(DoSensorPin);        //read the analog value
and store into the buffer
        analogBufferIndex++;
        if(analogBufferIndex == SCOUNT)
              analogBufferIndex = 0;
     }

     static unsigned long tempSampleTimepoint = millis();
     if(millis()-tempSampleTimepoint > 500U)    // every
500 milliseconds, read the temperature
     {
        tempSampleTimepoint = millis();
      }

     static unsigned long printTimepoint = millis();
     if(millis()-printTimepoint > 1000U)
     {
        printTimepoint = millis();

for(copyIndex=0;copyIndex<SCOUNT;copyIndex++)
         {
            analogBufferTemp[copyIndex]=
analogBuffer[copyIndex];
         }
         doaverageVoltage =
getMedianNum(analogBufferTemp,SCOUNT) *
(float)VREF / 1024.0; // read the value more stable by
the median filtering algorithm
         Serial.print(F("Temperature:"));
         Serial.print(temperature,1);
         Serial.print(F("^C"));
         doValue =
pgm_read_float_near( &SaturationValueTab[0] +
(int)(SaturationDoTemperature+0.5) ) *
doaverageVoltage / SaturationDoVoltage;    //calculate
the do value, doValue = Voltage / SaturationDoVoltage *
SaturationDoValue(with temperature compensation)
         Serial.print(F(",    DO Value:"));
         Serial.print(doValue,2);
         Serial.println(F("mg/L"));
     }
```

```
    if(serialDataAvailable() > 0)
    {
        byte modeIndex = uartParse();    //parse the uart
command received
        doCalibration(modeIndex);        // If the correct
calibration command is received, the calibration
function should be called.
    }
}

boolean serialDataAvailable(void)
{
  char receivedChar;
  static unsigned long receivedTimeOut = millis();
  while ( Serial.available() > 0 )
  {
    if (millis() - receivedTimeOut > 500U)
    {
      receivedBufferIndex = 0;

memset(receivedBuffer,0,(ReceivedBufferLength+1));
    }
    receivedTimeOut = millis();
    receivedChar = Serial.read();
    if (receivedChar == '\n' || receivedBufferIndex ==
ReceivedBufferLength)
    {
  receivedBufferIndex = 0;
  strupr(receivedBuffer);
  return true;
    }else{
        receivedBuffer[receivedBufferIndex] =
receivedChar;
        receivedBufferIndex++;
    }
  }
  return false;
}

byte uartParse()
{
    byte modeIndex = 0;
    if(strstr(receivedBuffer, "CALIBRATION") != NULL)
        modeIndex = 1;
    else if(strstr(receivedBuffer, "EXIT") != NULL)
        modeIndex = 3;
    else if(strstr(receivedBuffer, "SATCAL") != NULL)
        modeIndex = 2;
    return modeIndex;
}

void doCalibration(byte mode)
{
    char *receivedBufferPtr;
    static boolean doCalibrationFinishFlag =
0,enterCalibrationFlag = 0;
    float voltageValueStore;
    switch(mode)
    {
      case 0:
      if(enterCalibrationFlag)
          Serial.println(F("Command Error"));
      break;

      case 1:
      enterCalibrationFlag = 1;
      doCalibrationFinishFlag = 0;
      Serial.println();
      Serial.println(F(">>>Enter Calibration
Mode<<<"));
      Serial.println(F(">>>Please put the probe into the
saturation oxygen water! <<<"));
      Serial.println();
      break;

     case 2:
      if(enterCalibrationFlag)
      {
          Serial.println();
          Serial.println(F(">>>Saturation Calibration
Finish!<<<"));
          Serial.println();
          EEPROM_write(SaturationDoVoltageAddress,
doaverageVoltage);

EEPROM_write(SaturationDoTemperatureAddress,
temperature);
          SaturationDoVoltage = doaverageVoltage;
          SaturationDoTemperature = temperature;
          doCalibrationFinishFlag = 1;
      }
      break;

        case 3:
        if(enterCalibrationFlag)
        {
            Serial.println();
            if(doCalibrationFinishFlag)
                Serial.print(F(">>>Calibration
Successful"));
            else
                Serial.print(F(">>>Calibration Failed"));
            Serial.println(F(",Exit Calibration
Mode<<<"));
            Serial.println();
            doCalibrationFinishFlag = 0;
            enterCalibrationFlag = 0;
        }
        break;
    }
}

int getMedianNum(int bArray[], int iFilterLen)
{
      int bTab[iFilterLen];
      for (byte i = 0; i<iFilterLen; i++)
      {
    bTab[i] = bArray[i];
      }
      int i, j, bTemp;
      for (j = 0; j < iFilterLen - 1; j++)
      {
    for (i = 0; i < iFilterLen - j - 1; i++)
          {
      if (bTab[i] > bTab[i + 1])
          {
    bTemp = bTab[i];
          bTab[i] = bTab[i + 1];
    bTab[i + 1] = bTemp;
       }
    }
      }
      if ((iFilterLen & 1) > 0)
  bTemp = bTab[(iFilterLen - 1) / 2];
      else
  bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1])
/ 2;
      return bTemp;
}

void readDoCharacteristicValues(void)
{
    EEPROM_read(SaturationDoVoltageAddress,
```

```
SaturationDoVoltage);
      EEPROM_read(SaturationDoTemperatureAddress,
SaturationDoTemperature);

if(EEPROM.read(SaturationDoVoltageAddress)==0xFF
&&
EEPROM.read(SaturationDoVoltageAddress+1)==0xFF
&&
EEPROM.read(SaturationDoVoltageAddress+2)==0xFF
&&
EEPROM.read(SaturationDoVoltageAddress+3)==0xFF)
      {
         SaturationDoVoltage = 1127.6;      //default
voltage:1127.6mv
         EEPROM_write(SaturationDoVoltageAddress,
SaturationDoVoltage);
      }

if(EEPROM.read(SaturationDoTemperatureAddress)==0x
FF &&
EEPROM.read(SaturationDoTemperatureAddress+1)==0x
FF &&
EEPROM.read(SaturationDoTemperatureAddress+2)==0x
FF &&
EEPROM.read(SaturationDoTemperatureAddress+3)==0x
FF)
      {
         SaturationDoTemperature = 25.0;      //default
temperature is 25^C

EEPROM_write(SaturationDoTemperatureAddress,
SaturationDoTemperature);
      }
}

String twodigit(float ss1)
{
      String gstr;

      // 開始處理  float ss1 的值

      long tmp = ss1;      // 整數部分

      long yytmp = (ss1 -tmp)*100+0.5;    //    小數部分;
從小數點後第三位做四捨五入(round)到第二位

      /// if(yytmp >= 100) yytmp=99;    // 改用以下方法
處理  ..

      if(yytmp >= 100) {      // 防錯

            yytmp=0;

            ++tmp;    // 0.99xyz.. +0.005 ===>    1.0pqr...

      } // it is 0.99xyz...

      gstr += tmp; // 整數部分
      gstr += ".";    // 小數點
      if(yytmp < 10) gstr += "0";

      gstr += yytmp;
        Serial.println(gstr);    // 印到串口
      return(gstr);
}

void motor1start() {
   digitalWrite(motorPin3,HIGH);
}
void motor2start() {
   digitalWrite(motorPin4,HIGH);
}
void motor3start() {
   digitalWrite(motorPin5,HIGH);
}

void motor4start() {
   digitalWrite(motorPin6,HIGH);
}

void motor1stop() {
   digitalWrite(motorPin3,LOW);
}
void motor2stop() {
   digitalWrite(motorPin4,LOW);
}
void motor3stop() {
   digitalWrite(motorPin5,LOW);
}
void motor4stop() {
   digitalWrite(motorPin6,LOW);
}
```

【評語】052202

1. 以自製水耕蔬菜對於小白菜生長效率之探討。

2. 成員分工明確且對整體實驗架構理解。

3. 宜使用適當之統計分析方法來處理數據並據以解釋。

作品海報

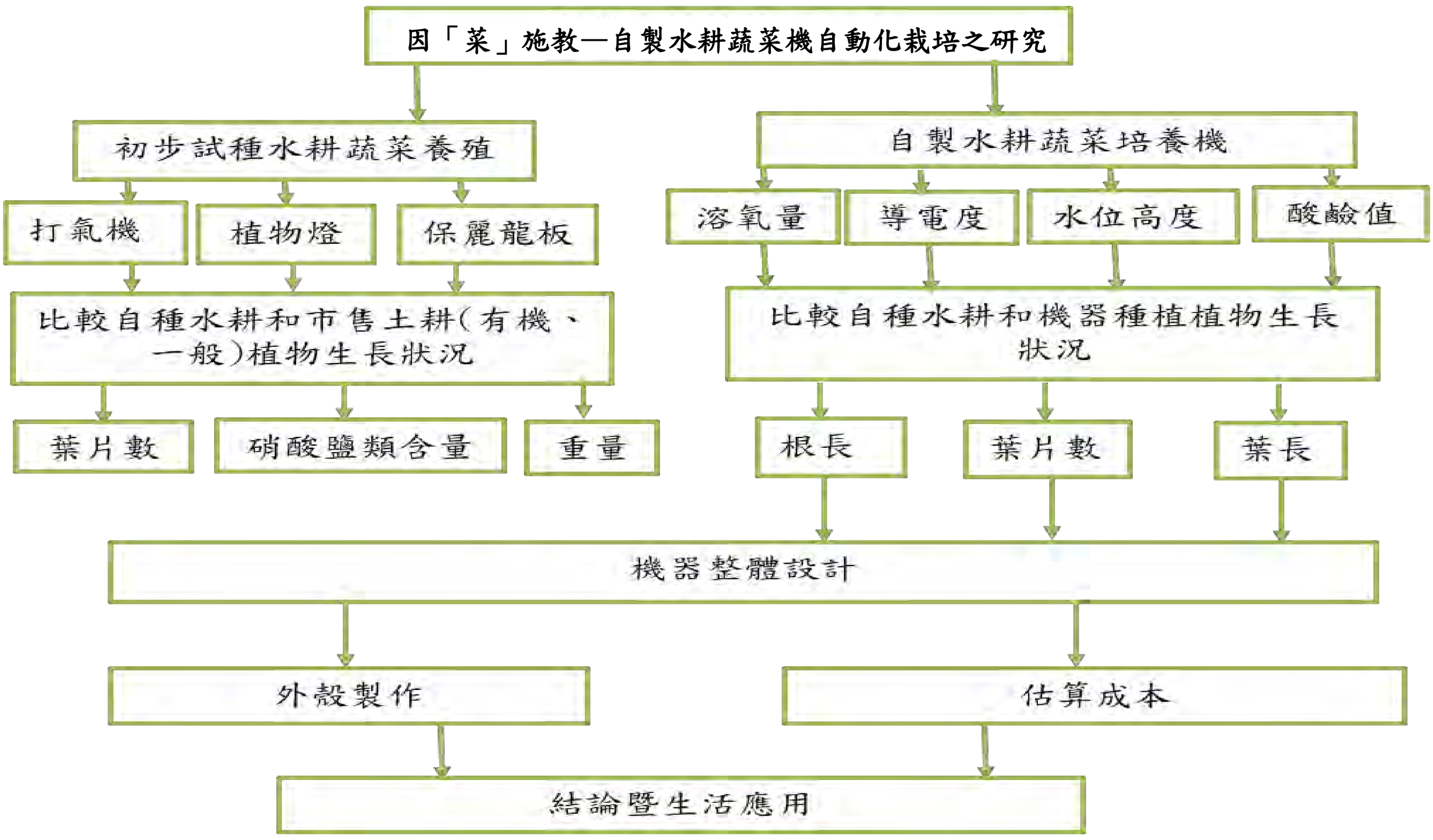
◎研究動機

在家自己種植蔬菜，自由選擇營養液，在整個生長週期中，使用的水源、燈光、肥料都受直接的監控，**降低吃到農藥或其他殘留化學藥物的風險**。不僅美化了環境，還能節省出外買菜的時間與金錢。我們希望設計出一臺**自動蔬菜養殖機**，透過電腦軟件與定時器、量測儀的結合，使用者**不需全程參與植物照護**，可以簡單方便**讀取液晶顯示器的數值，了解植物狀況**。

◎研究目的

一、**利用Arduino設計出自動水耕蔬菜養殖機**，並實際種植水耕蔬菜(以小白菜為例)。
二、比較**自種水耕蔬菜**與市售有機蔬菜、一般蔬菜的差別。
三、比較**自製養殖機**與市售養殖機之優缺差異。
四、**自製養殖機收成耗費成本之估算**。

◎研究流程



◎主要研究器材

溶氧量感測器	EC計	水溫感測器	pH值感測器	鋼釘(水位計)	電池和電池盒	Arduino模組
打氣機	沉水馬達	水質測試劑	多頭植物燈	保麗龍板	水管	發泡棉

◎研究步驟

一、種子育苗　　二、第一代水耕蔬菜　　三、第二代水耕蔬菜:打氣機、植物燈、保麗龍板

圖(一)小白菜育苗裝置

※結果：
我們嘗試從種子開始種植、使用海綿育苗，一天後，大約一半的種子成功發芽，這時我們適時補充光線，兩天後，小白菜苗順利成長。

圖(二)初代小白菜

※結果：
小白菜一天比一天虛弱，最後**全數死亡**，檢討了第一代的問題後，我們試種了第二代。

圖(三)二代小白菜

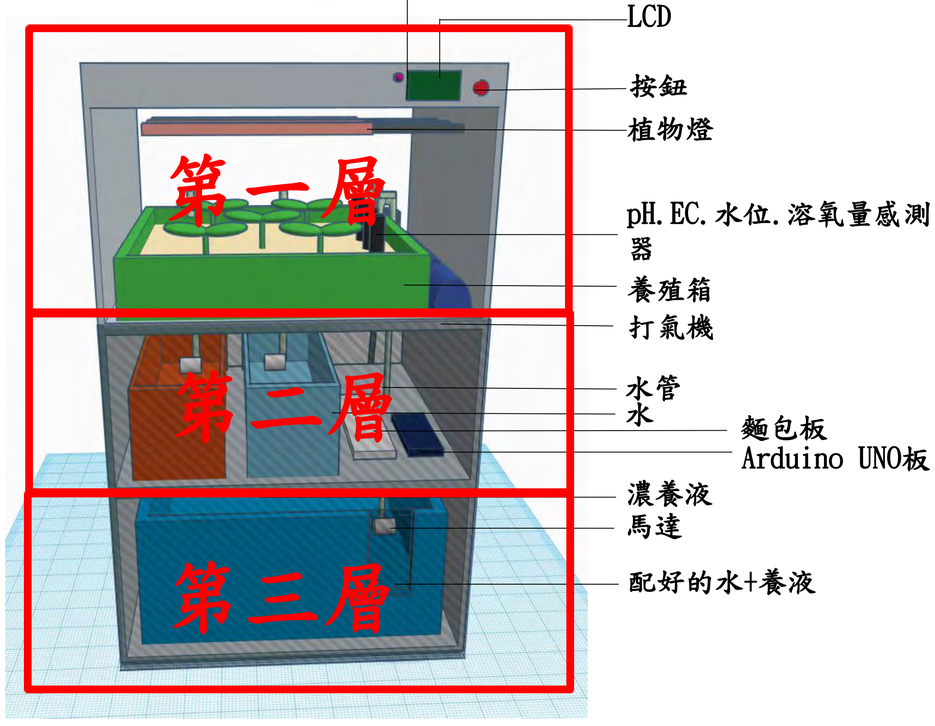
圖(四)二代小白菜

※結果：
同樣維持一周加水、換養液一次，**植物燈則是以開關時數比1：1滿足植物日照需求**，並每天持續量測根長、葉長，也補充蒸發的水量。兩周後，平均葉長從原本的6.1公分成長至11.03公分，根長也從原來的6.62公分成長至11.91公分，最大的植株甚至根與葉都超過15公分，初次試種水耕蔬菜**成功**。

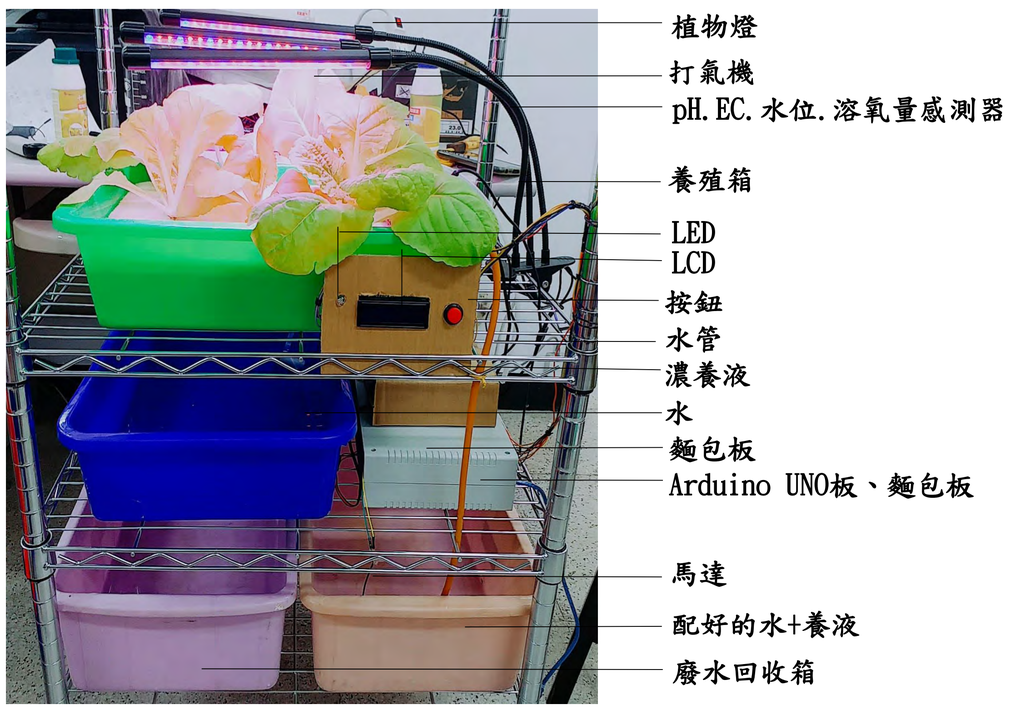
三、自製水耕蔬菜養殖機(本研究自行設計)

我們使用Tinkercad繪製出設計圖，根據此設計圖，製作出第一代養殖機，再經過改良製作出第二、第三代養殖機。

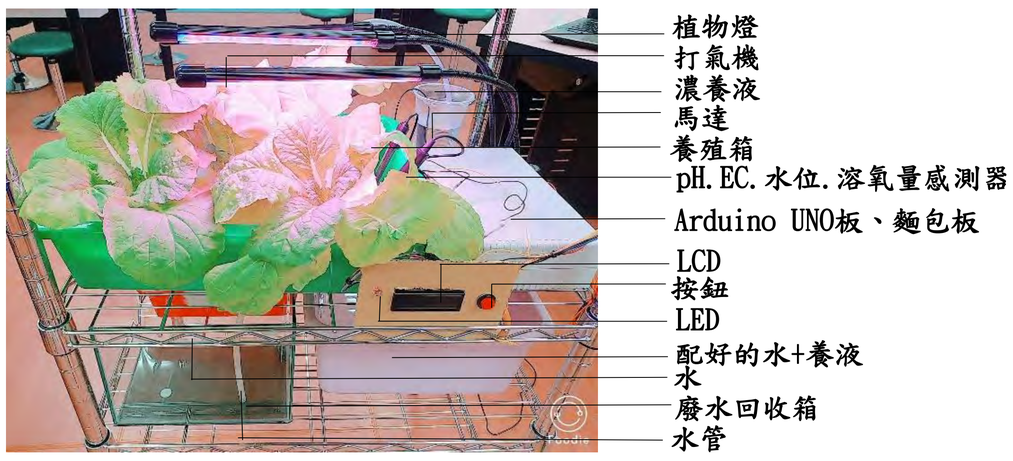
(一)設計圖



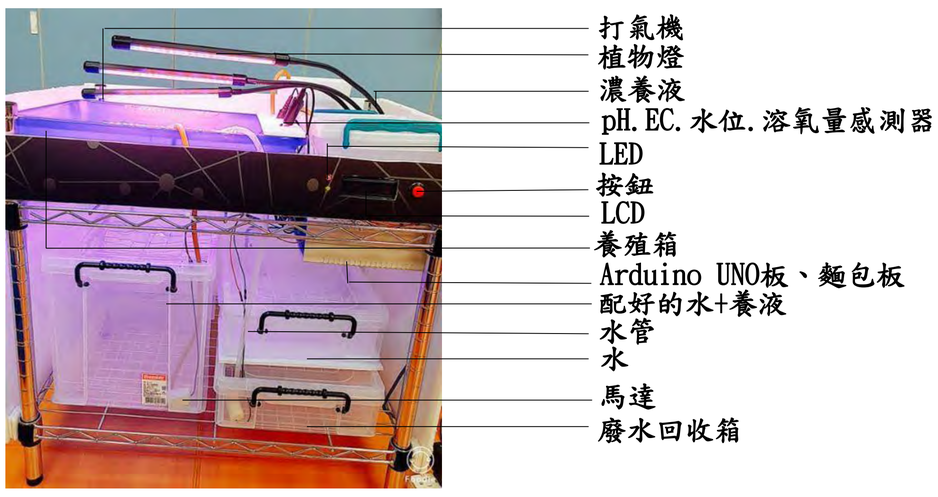
(二)第一代養殖機



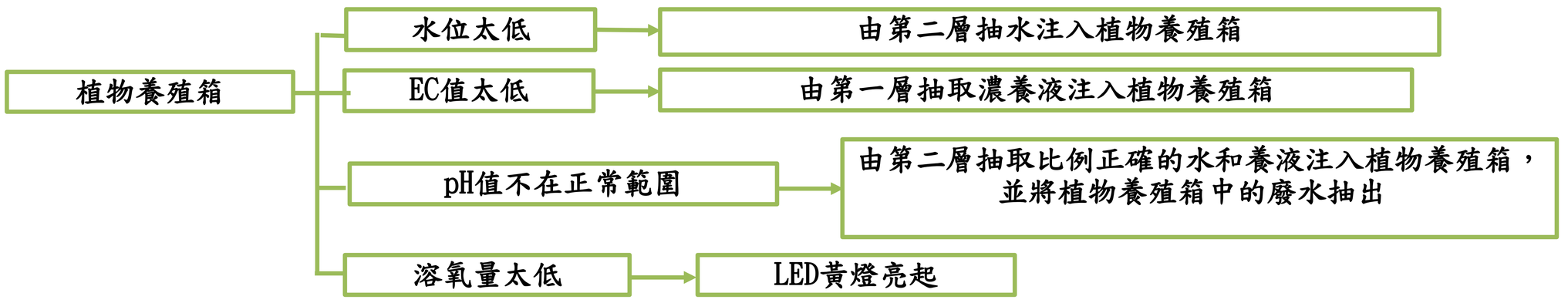
(三)第二代養殖機



(四)第三代養殖機



※運作模式(第三代)：



(三)儀器介紹

1.Arduino UNO板：

我們選擇用**Arduino UNO板**作為控制面板。因為Arduino UNO板**彈性較大，入手門檻低、價格低**，且有豐富的**配件及程式碼**可以參考。

2.pH值感測器：

水耕營養液**必須維持 pH 值穩定**，以確保**營養液元素維持在可被利用之離子形態**。若pH值不在合適範圍內，就代表此時養殖箱中的環境已不適合植物生長，這時，裝置會**啟動馬達換水**，**自動**將原本養殖箱中的水抽出，並將新的調配好的營養液注入養殖箱中。

3.EC值感測器與溫度感測器：

電導度(EC值)所產生之滲透壓會影響作物之水分吸收能力。電導度是指水中解離性無機鹽類之總和，而**營養液中的離子濃度與 EC 值呈高度相關性**。因**溫度高低會影響EC值**，所以EC值感測器會配合溫度感測器是來修正EC值。營養液經植物吸收後 EC 值會逐漸下降，當 EC 值不足時，裝置**馬達會自動啟動**，補充新的營養液使保持最適合的濃度。

4.水位計：

水位如果太低，植物的**根吸不到足夠的水分**，會導致植物**倒塌**。我們以水位計監控養殖箱的水位，當水位過低時，將啟動馬達**自動補水**，直到水位達到設定值。

5.溶氧量(DO)感測器及打氣機：

在水質管理實務上，溶氧量被認知為是**判斷水質好壞**之主要指標。一般而言，濃度越高代表水質狀況越好。水中之飽和溶氧量受水溫以及水中含有雜質量之影響，水溫越高飽和溶氧量濃度就越低。當水中營養液之溶氧量為 5 ppm 以上，就不會有根腐病的問題；若在 2 ppm 以下，根系之生理營養不良，養分吸收少。若溶氧不足，**LED會亮起黃燈，提醒使用者開啟打氣機**。

6.馬達：

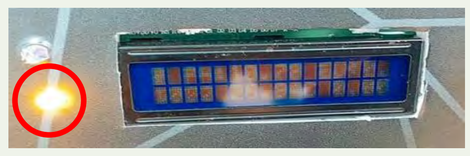
當**pH.EC.水位不在正常範圍內時，啟動馬達直到數據回到正常值**。經過測量，我們繪製出**秒數與抽水量**的關係圖，進而訂定啟動馬達的秒數，根據實測的結果，馬達每秒鐘的抽水量大約為17ml。

7.LED(Light Emitting Diode，發光二極體)：

當**感測器所測試到的數據有異常**時，LED燈會亮起**紅燈**，直到數據恢復正常，紅燈才會熄滅。我們使LED燈與馬達配合，當**馬達運轉**時，即代表有**數值異常**，LED燈就會**亮起**。 而溶氧量感測器則連接LED黃燈，當感測到溶氧量不足時，**LED即會亮起黃燈**，直到再次感測到正常值，黃燈才會熄滅。

圖(五)數據皆在正常值(無燈號)

圖(六)pH值異常(紅燈亮起)



圖(七)溶氧量過低(黃燈亮起)

8.LCD(Liquid Crystal Display，液晶顯示器)：

為數據的**顯示板**，會顯示感測器所偵測到的**pH、EC、水位計、溫度及DO的數值**。

9.按鈕：

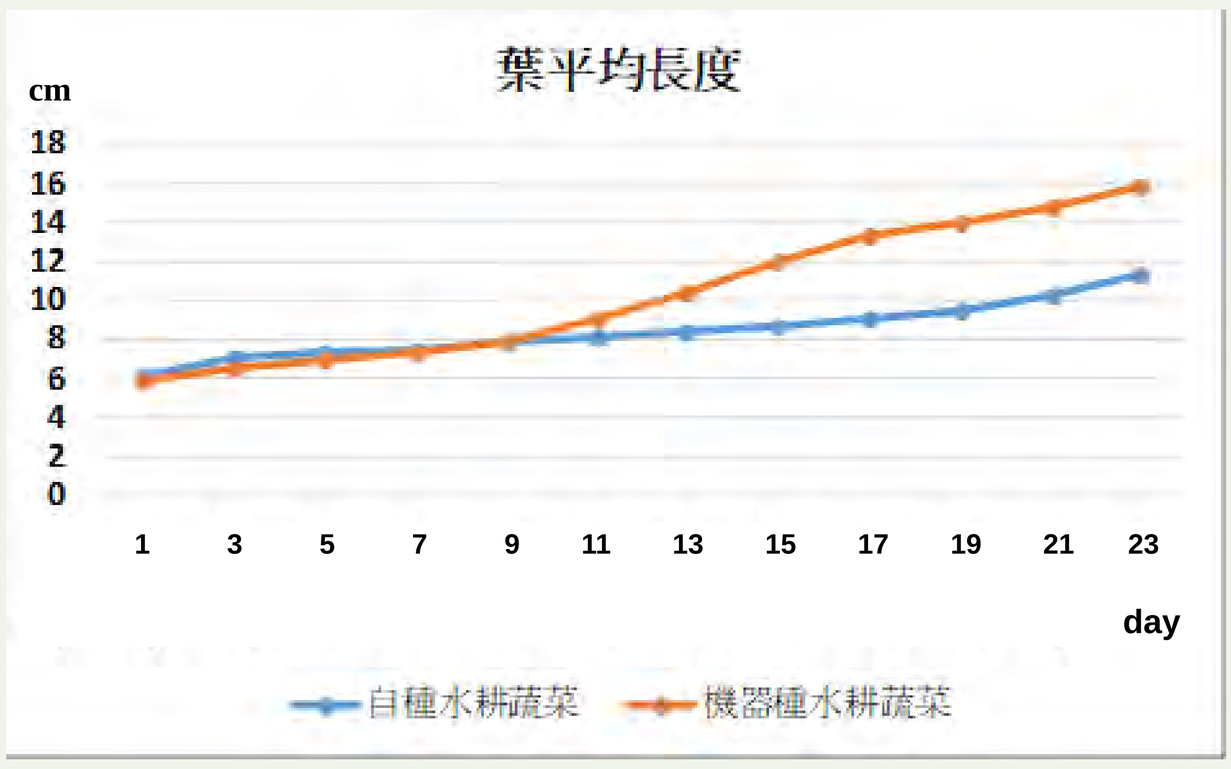
因為不同種植物需要不同的生長環境，同一種植物也會因為季節的改變而有不同的需求。所以我們設計按鈕，當按下按鈕時，就能**切換到其他模式**，目前養殖機設計了4個模式，每個模式可根據需求預先輸入pH值範圍與EC值範圍，本系統還**可擴充到n個模式**。

◎研究結果

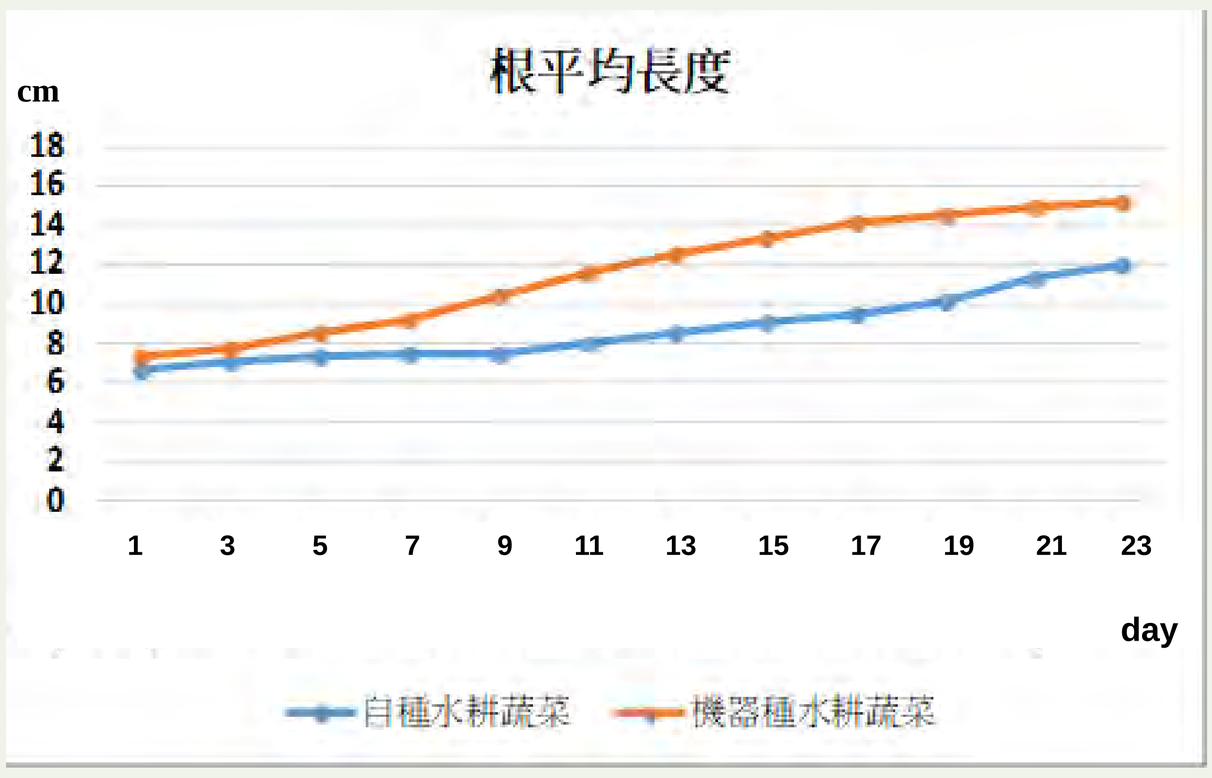
一、比較市售有機蔬菜、一般蔬菜的差別。

種類	自種水耕蔬菜	市售一般蔬菜	市售有機蔬菜
葉片數	16片	9片	10片
重量	40.5公克	110.6公克	75.0公克
硝酸鹽	中等	最多	最少

二、利用Arduino做出自動水耕蔬菜養殖機，並和自種水耕蔬菜比較。



葉的生長情形比較(三重覆平均值)



根的生長情形比較(三重覆平均值)

由圖可知用機器控制所種出的蔬菜，不論是葉還是根，生長幅度皆比自己手動控制的還要大。

三、比較自製養殖機與市售養殖機之差異，調整外觀，使其符合經濟效益。

(一)機器成本

品項	價格(元)
溶氧計	6000
pH計	1800
EC計	2500
其他	700
植物燈	600
架子	800
ArduinoUNO板	680
打氣機	300
總價：13380元	

(二)收成一次耗費成本

品項	價格(元)
打氣機電費	5.3
植物燈電費	15.1
Arduino電費	10
水費	0.13
養液費	35
總價：65.53元	

由於整體**成本過高**，我們提出以下改善方式：

1. **大型養殖場，大範圍監控植物生長環境**、降低成本。
2. **一般個人，增加單次種植棵數**，自栽自食，安全可靠。
3. **自製開發溶氧量計**，使得此自動化機器經濟價值提升。

◎討論

一、水位計：

我們自製水位計，用**兩根鋼釘**作為感測器，利用**導電**的原理，製作一個**迴路**，因為水中的離子可以導電，所以若是鋼釘浸到液面，會因為水溶液電阻小而使讀取到的電壓相對較低，如果鋼釘未浸到液面，則形成斷路，所讀取到的電壓即外接的電壓，相對較高，如此我們便能清楚的從**讀取到的電壓**了解水位狀況。

二、市售水耕蔬菜養殖機與自製水耕蔬菜養殖機功能上的差別：

我們找出了植物生長的所需考量到的因素，修正一般市售水耕蔬菜養殖機的缺點，再自製水耕蔬菜養殖機。整體而言，自製的水耕蔬菜養殖機**操作簡單**，**考量到的因素多**，能提供植物最好的生長環境，改良並補足市售水耕蔬菜機缺漏。

三、程式碼的撰寫所遇到的困難：

1. pH感測器：由於pH感測器**電壓不穩**，導致數值不準確，所以我們使其每測量四十次後平均後印出數值，我們捨棄前九次印出的數值，以第十次所印出的數值，顯示在顯示器上，並決定是否啟動馬達。
2. 按鈕：能**改變數值的參數**，使不同植物在不同條件下能使用不同的參數。利用**餘數**的方式，設定四種植物生長的參數，當按鈕被按下的次數除以四餘零時，為第一組參數，餘一時，為第二組參數，依此類推。本系統還**可擴充到n種植物生長的參數**，只要把四改為n即可。

◎結論

一、利用Arduino設計出自動水耕蔬菜養殖機，並實際種植水耕蔬菜：

水耕蔬菜養殖機能有效節省人力成本，且機器自動種植的蔬菜生長較迅速。為達成此目的，我們使用Arduino UNO板作為控制面板，**以對水質影響較大的四個變因：導電度(EC)、酸鹼值(pH)、溶氧量(DO)，自製水位計，監測控制植物最佳生長環境**。

二、自種水耕蔬菜與市售有機蔬菜、一般蔬菜的差別：

我們比較市售與水耕種植的差異，發現水耕種植蔬菜較一般市售蔬菜小，但是**硝酸鹽含量較低**，對人體帶來傷害較小，而有機蔬菜硝酸鹽含量最低。接著再跟機器自動種植的差異，**機器自動種植能使蔬菜長得更迅速**。

三、比較自製養殖機與市售養殖機之優缺差異：

本研究自行開發設計的水耕蔬菜養殖機，與一般市售比較，多了能自動監控植物生長環境，並透過**自動換水、加水、加養液等方式做出相對回應，讓使用者只需負責採收的部分，節省金錢及人力。**因監測變因較多，功能性也較好。在考量相同功能的情況下，本研究整體成本可較一般市售價格節省快一半。

四、**自製養殖機收成耗費成本之估算：**

我們對養殖成本進行估算，在一次只種五顆小白菜的小規模種植情況下，單棵總成本偏高，但可藉由**擴大種植規模改善此問題**，當單次種植棵數到達50顆以上時，單顆成本將不到五元。