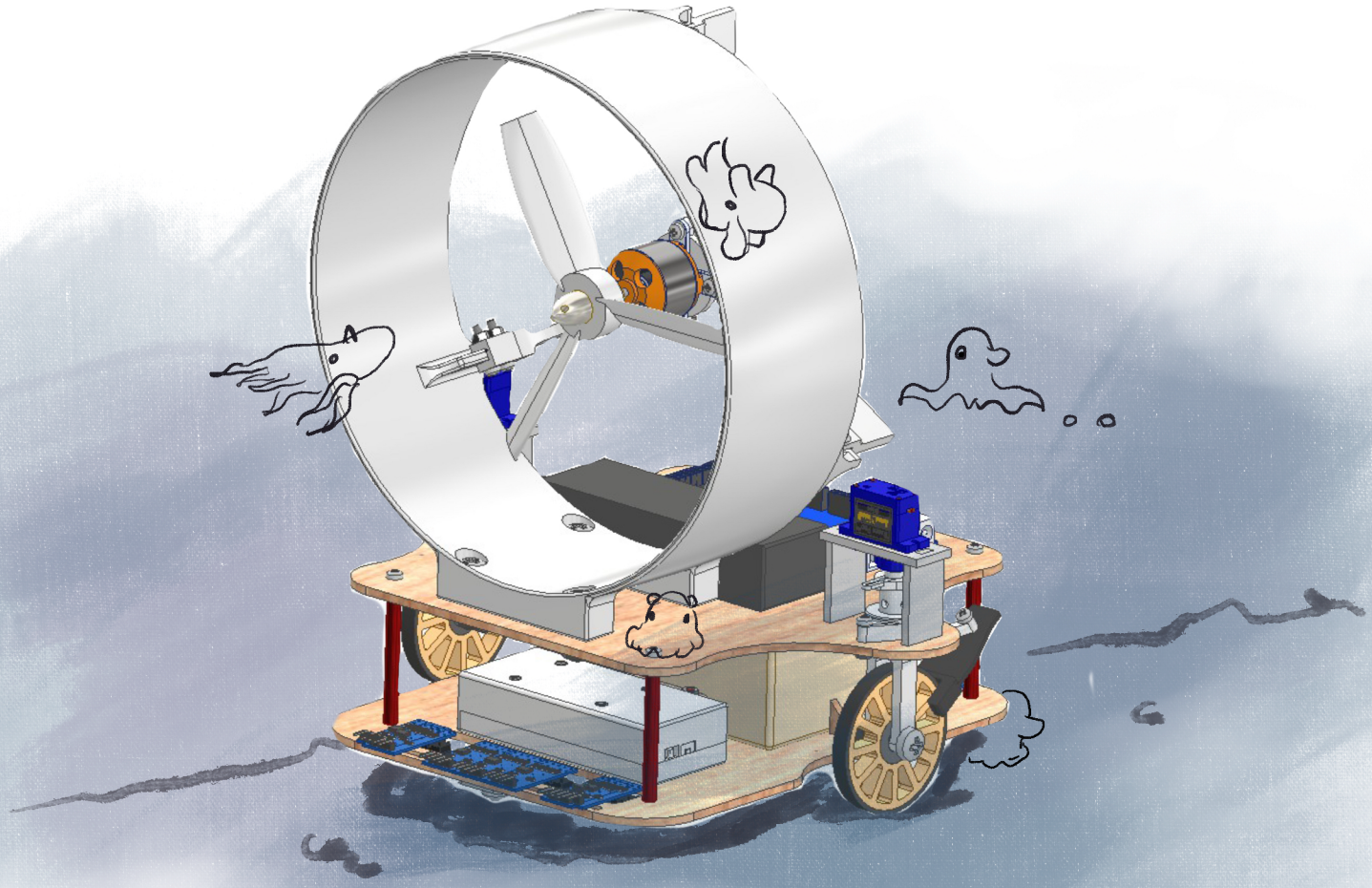
109-2 機械工程實務 成果報告書

FormulaAir: Propeller-Powered RacingVehicles

Team 12

小飛象章魚



教授	詹魁元	楊馥菱	劉建豪	林沛群	蔡耀陽	陳湘鳳
評分						

成員：

機械三 B07502011 楊宜瑄

機械三 B07502037 潘宣蓉

機械三 B07502039 吳宥廷

機械三 B07502069 陳旻彥

機械三 B07502165 賴昭蓉

目錄

1. 緒論

機械工程實務，為整合機械系各項科目的一門課，我們將從此門課中發揮三年所學，藉由團隊合作及不斷的討論，將激盪出的想法實踐於產品上。而今年的主題為「風動車」，在行總體規劃以前，本組組員先各自進行資料蒐集，藉歷屆學長姐的期末報告，從中找出氣動車各系統之可行的方案，此方法使我們能非常有效率的學習相關知識。

在大家都有氣動車的背景知識後，我們將分三階段進行氣動車規劃及製作：
(1) 構思及實作：提出各系統之相關問題後，大家進行腦力激盪，藉各自的創意及不斷的討論中構思出最好的模型，之後各部門再將想法實踐。
(2) 溝通及整合：此為各部門最需溝通的時刻，因各部並非完全獨立，而是環環相扣，故在做法上，每人都會參與兩個部門之討論，藉此增進共識，也能讓各系統整合更有效率。
(3) 收斂及穩定：在系統整合後，我們對此產品進行改良及優化。做法為在測試中提出各式問題，並藉大家不斷討論，取得共識，而後進行改良，才能使氣動車達到穩定狀態。

而在設計理念上，考慮預算及時間等因素，我們以「加工容易且迅速」、「拆裝維修方便」、「低成本」為主要依據，由此希望產品達到「以精簡的設計達到所需功能，並降低成本和製造與拆裝之失敗率」，並從此基礎下加入一些創意，使氣動車效能更佳。

因此，藉由上述概念，本組氣動車之各部門的設計構想及相關特色如下：

	設計構想	特色說明
風扇	設計：NACA 4-digit，配合 Javafoil (翼型設計)及 Javaprop(扇葉設計)	根據所需翼型輸入相關參數，可直接生成單片扇葉。
	製造：Inventor 及 3D 列印機	利用所得扇葉曲面組裝風扇，且可以快速產出實體，精度也在接收範圍。
	分析：ANSYS fluent 及實驗	可模擬出流場速度分布及扭矩等，並配合推力測試等實驗修正相關問題。
車體	轉向：伺服馬達垂直雙輪轉向	此概念為簡化機構，降低接合所造成之阻力。
	煞車：雙邊點煞系統	不易與其他系統干涉，雙邊則為保持車體平衡使之不傾斜。

	製造：Inventor 及雷射切割機	利用熟悉的繪圖軟體，並配合雷射切割機可以快速生產出所需機構。
控制	循跡：五路紅外線感測器	紅外線感測器可靈敏感測黑白變化，五路則為精準判斷車體偏移程度。
	車速：PID	搭配循跡判斷之結果調配車速，始控制更加穩定。

2. 功能需求

2.1. 驅動

在「機械工程實務期中測試規則」中，其規定車輛以螺旋槳葉片旋轉產生之風力為推進與煞車之動力來源，而在「機械工程實務期末驗收規則」中，以期中測試為基礎，增加爬坡任務，其驅動之推力需再額外克服車重之分量。故在測試進行中，如何利用風扇將車體穩定推進及爬升為很重要的一環。以下將所需元件進行功能及需求介紹。

(1) 風扇：最主要之動力來源，其扇葉設計及動力牽涉到流體相關理論，將於 3.1 節進行介紹。
(2) 馬達：提供馬達轉動之動力，其要求為高轉速以達到所需推力，使可以前進及爬坡。
(3) 風扇罩：此安裝於風扇旋轉速度之切線方向，以防高速旋轉之葉片因斷裂等因素而飛出，使人員受傷。

另外，驅動也與車體的相關阻力(如地面磨擦力、空氣阻力等)有關，也與輪軸摩擦力有關。故以驅動而言，其量化指標為：馬達提供之轉速、風扇在不同轉速下之推力、車體阻力常數、軸承之摩擦力。

2.2. 煞車

當車體通過第一循跡線到達第一停止區，以及通過第二循跡線到達第二停止區時，若只是單純停止風扇，車體可能會有慣性繼續前進而超過停止區；若是在期末的爬坡場地，則會受重力而向下滑動，無法成功停止在指定區域，所以需要煞車系統來達成停止的需求。煞車系統不只是停止的功能，在車體行進過程中若需要減速或暫停，皆能發揮作用，暫時停止車身動作以對後續做出反應。

2.3. 循跡

車體由起跑區出發後，要能找到循跡線並沿著循跡線穩定前進，且不能因為起始區和停止區的框線導致循跡誤判，因此，我們需要有一個能夠偵測尋跡線的穩定的感測器，並依據感測器回傳的值控制轉向機構，使車體自動校正前進的方向，盡可能穩定地在尋跡線上前進。

2.4. 轉向

轉向機構為控制車體方向的重要機構，尤其車體在切換循跡線以及循跡過程中。轉向需能及時且穩定，控制上提供一定的精確度、機構強度要能承受夠大的扭矩帶動車體轉彎。

2.5. 倒車

到第一停止區後，車體需倒退走回到第二停止區。期末測試和期中測試不同的是車體可以利用重力位能讓車子下滑，不需要風扇需往反方向吹才能使車體倒退。需要另外考慮的影響因素有以下三點：

(1) 下滑速度控制。需在第二停止區煞車停止。
(2) 方向控制。必須在過程中保持循跡。
(3) 若車子不慎在第二停止區前停止，需要使用風扇反向驅動已完成倒車。

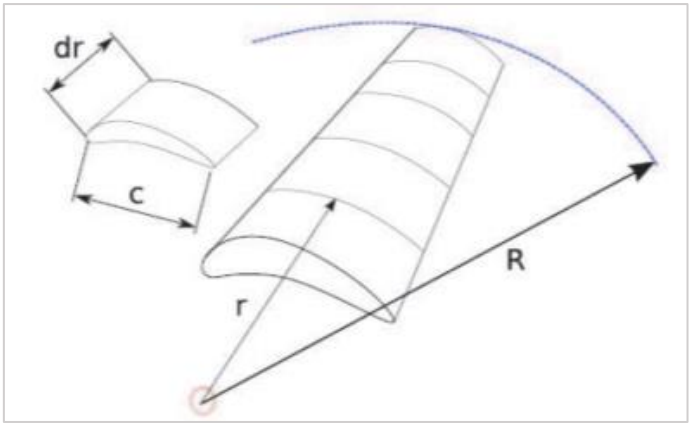
3. 設計概念與布置

3.1. 風扇

風扇動力系統為車體之主要驅動來源，其所提供動力影響車體之推力、移動速度及穩定度。以下將從扇葉設計之理論進行介紹，再配合軟體進行模擬分析，最後介紹本組所設計之各版本風扇。

3.1.1. Blade element momentum theory

Blade element momentum theory 為設計扇葉時，所使用之基礎理論。其概念在於將扇葉切分為 10 至 20 個小片段，並假設每一小片段之間無空氣動力學之交互作用，每一小片段所受之 rotational speed (= 角速度(Ω) × 半徑(r))、chord length (c)及 twist angle (γ)皆不相同，最後個別計算其升力和阻力，疊加計算後即為扇葉之總推力及扭矩，其概念如圖 3.1-1 所示。



(圖 3.1-1) The Blade Element Model [1]

而我們可以利用 Blade element theory，並引進 Power Coefficient (C_p)以代表葉片之推力。首先，為推導 Power Coefficient，先進行以下假設：如圖 3.1-2 所示，考慮一 Actuator Disk(在此處可想成裝有葉片之圓盤)，且假設流體為非黏性流體(inviscid fluid)，故根據 Bernoulli's equation，在區段$1-2$及區間$3-4$可有以下公式：

$$P_1 + \frac{\rho V_1^2}{2} = P_2 + \frac{\rho V_2^2}{2} \qquad (式\ 3.1.1)$$

$$P_3 + \frac{\rho V_3^2}{2} = P_4 + \frac{\rho V_4^2}{2} \qquad (式\ 3.1.2)$$

於此再進一步假設$P_1 = P_4$且$V_2 = V_3$，故得到：

$$P_2 - P_3 = \frac{\rho(V_1^2 - V_4^2)}{2} \qquad (式\ 3.1.3)$$

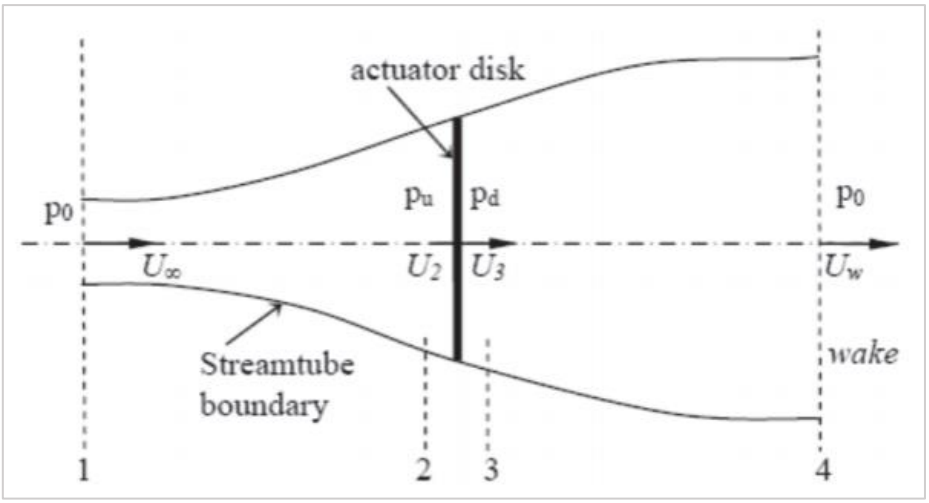
因此，根據式 3.1.3，我們定義

$$a = \frac{V_1 - V_2}{V_1} \qquad (式\ 3.1.4)[1]$$

並進一步假設

$$a' = \frac{1 - 3a}{4a - 1} \qquad (式\ 3.1.4)[1]$$

其中a'為 angular induction factor，其意義為每一環帶(annular ring)所輸出最大的功，在此忽略詳細推導。[1]



(圖 3.1-2) Actuator Disk Model [1]

藉上述假設，我們可以推導 Power Coefficient。首先，圖圖 3.1-3 為扇葉翼型之示意圖，並由此得到下列公式：

$$\begin{cases} V_{rel} = \dfrac{V_\infty(1-a)}{sin\phi} & (式\ 3.1.5) \\ \tan\phi = \dfrac{V_\infty(1-a)}{\Omega r(1+a')} = \dfrac{1-a}{(1+a')\lambda_r}\ (\lambda(r) = \dfrac{r\Omega}{V_\infty}) & (式\ 3.1.6) \\ dF_L = \dfrac{C_L}{2}\rho V_{rel}^2 cdr & (式\ 3.1.7) \\ dF_D = \dfrac{C_D}{2}\rho V_{rel}^2 cdr & (式\ 3.1.8) \\ dL = dF_L sin\phi - dF_D\cos\phi & (式\ 3.1.9) \\ dT = dF_L cos\phi - dF_D\sin\phi & (式\ 3.1.10) \end{cases}$$

令扇葉數為 B，並將每單位之扭矩之關係：dQ = rdL，得到

$$dQ = \frac{\mathrm{B}}{2}\rho V_{rel}^2(C_L sin\phi - C_D\cos\phi)crdr \qquad (式\ 3.1.11)$$

並定義 solidity ratio 為

$$\sigma = \frac{\mathrm{Bc}}{2\pi\mathrm{r}} \qquad (式\ 3.1.12)[1]$$

我們可以得到每一單元(element)之單位扭矩方程式：

$$dQ = \sigma\pi\rho\left(\frac{V_\infty(1-a)}{sin\phi}\right)^2(C_L sin\phi - C_D\cos\phi)\mathrm{r}^2 dr \qquad (式\ 3.1.13)$$

單位推力方程式則為：

$$dT = \sigma\pi\rho\left(\frac{V_\infty(1-a)}{sin\phi}\right)^2(C_L\cos\phi - C_D sin\phi)\mathrm{r}^2 dr \qquad (式\ 3.1.14)$$

因此，由式3.1.6、式3.1.13及式3.1.14，經整理後得到以下二方程式：

$$\frac{\mathrm{a}'}{1+\mathrm{a}'}=\frac{\sigma \mathrm{C_L}\cos\phi}{4\sin^2\phi} \qquad (式\ 3.1.15)$$

$$\frac{\mathrm{a}'}{1-\mathrm{a}'}=\frac{\sigma \mathrm{C_L}\cos\phi}{4\lambda_r\sin^2\phi}\left(1-\frac{\mathrm{C_D}}{\mathrm{C_L}}\cot\phi\right) \qquad (式\ 3.1.16)$$

故解出a 與 a′，得到：

$$\begin{cases} \mathrm{a}=\dfrac{1}{1+\left(\dfrac{4\sin^2\phi}{\sigma \mathrm{C_L}\cos\phi}\right)} & (式\ 3.1.17) \\ \mathrm{a}'=\dfrac{1}{\left(\dfrac{4\cos\phi}{\sigma \mathrm{C_L}}\right)-1} & (式\ 3.1.18) \end{cases}$$

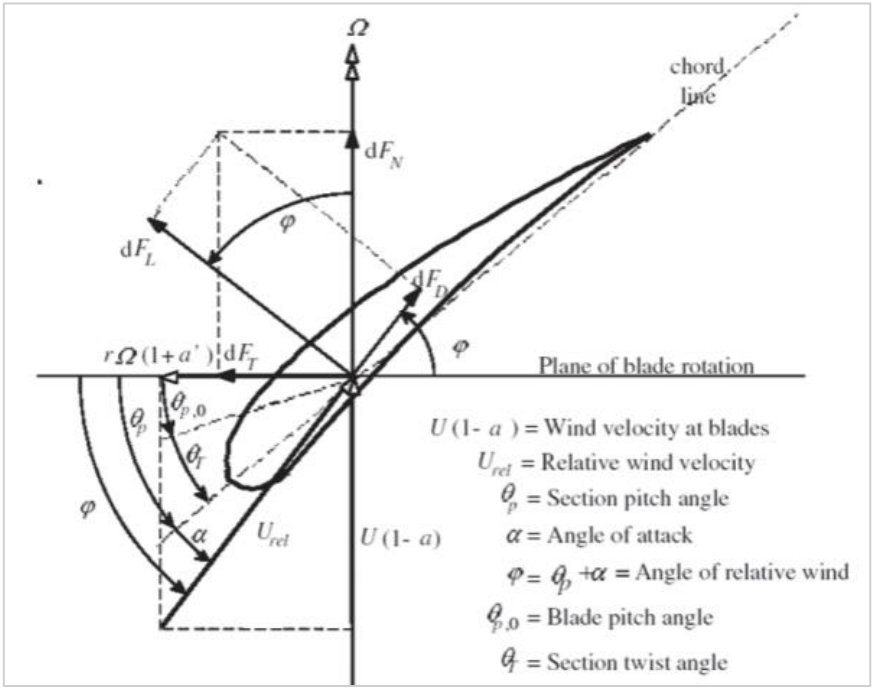
由式 3.317 及式 3.1.18 可得到：

$$\frac{\mathrm{a}}{\mathrm{a}'}=\frac{\lambda_r}{tan\phi} \qquad (式\ 3.1.19)$$

故由式 3.1.13、式 3.1.15及式 3.1.19，最後可以得到 Power Coefficient 之方程式：

$$\mathrm{C_p}=\frac{P}{\frac{1}{2}\rho V_\infty^3 A}=\frac{\int_{r_h}^{R}\Omega dQ}{\frac{1}{2}\rho V_\infty^3\pi R^2}=\frac{8}{\lambda^2}\int_{\lambda_h}^{\lambda}\lambda_r^3 a'(1-a)\left(1-\frac{C_D}{C_L}cot\phi\right)d\lambda_r \qquad (式\ 3.1.20)$$

因此，由式 3.1.20可知，在設計扇葉時，目標為$\mathrm{C_p}$最大化，故需適當配置$\mathrm{C_D}$及C_L之關係。



(圖 3.1-3) Blade Geometry for the analysis of a HAWT Rotor [1]

3.1.2. Actuator disk theory

以下為應用 Integral Momentum Equation 及 Actuator Disk Theory 推導風扇所受的推力(thrust)。下圖中 Actuator disk 為風扇。（圖 3.1-4）

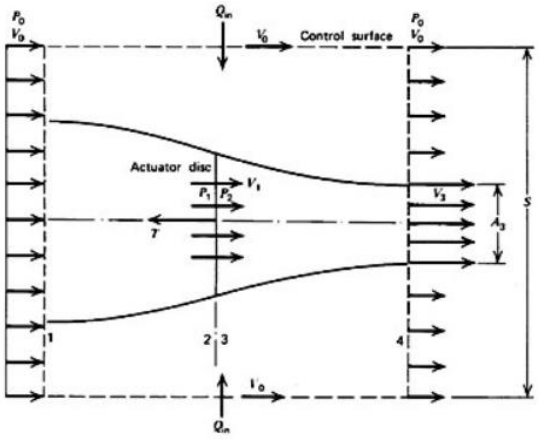


圖 3.1-4 流場示意圖[2]

由 Integral Momentum Equation 計算風扇所受的推力：

$$T = \sum \mathrm{F_x} = \int_s \mathrm{u_x}\rho(\vec{\mathrm{u}} \cdot \vec{\mathrm{n}})\mathrm{ds}$$
$$= \dot{\mathrm{m}}(\mathrm{u_3} - \mathrm{u_1}) \qquad (式\ 3.1.21)$$

又風扇對流體做功的功率可以由上下游動能變化得到：

$$power = \dot{\mathrm{m}}\left(\frac{\mathrm{u_3^2} - \mathrm{u_1^2}}{2}\right) \qquad (式\ 3.1.22)$$

由 Actuator Disk Theory，我們假設流場不可壓縮、非旋、無渦流、control volumn 以外環境不受到扇葉做功影響，在風扇(actuator disk)前後，壓力為不連續變化但流速為連續變化。(圖 3.1-5)

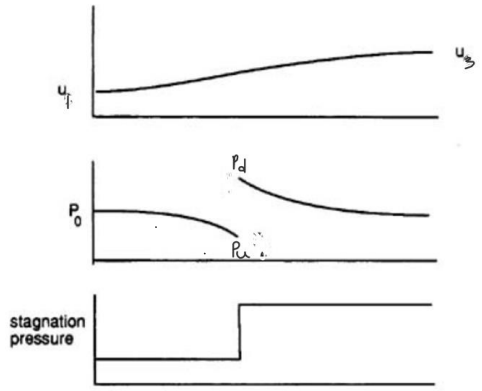


圖 3.1-5 扇葉周遭流速、壓力分布[2]

在這個假設下，風扇所受到的推力為：

$$T = \mathrm{A_{disk}}(\mathrm{p_d} - \mathrm{p_u}) \qquad (式\ 3.1.23)$$

風扇對流體做功的功率，可以以風扇處風速u_2表示：

$$\begin{aligned} power &= Tu_2 \\ &= A_{disk}(p_d - p_u)u_2 \qquad \ldots\ by(式\ 3.1.23) \\ &= \dot{m}(u_3 - u_1)u_2 \qquad \ldots by(式\ 3.1.21) \end{aligned} \qquad (式\ 3.1.24)$$

由(式 3.1.22)和(式 3.1.24)，得到

$$power = \dot{m}\left(\frac{u_3^2 - u_1^2}{2}\right) = \dot{m}(u_3 - u_1)u_2$$

$$\longrightarrow u_2 = \frac{u_1 + u_3}{2}$$

所以可以計算風扇處的流率

$$Q = A_{disk}u_2 = D_{disk}^2\pi/4 \times \frac{u_1 + u_3}{2}$$

為了計算扇葉所受到的推力(thrust)，我們繼續上面 Actuator Disk Theory 的推導。對風扇上游 x1 位子(壓力p_0平均流速u_1)和風扇右側(壓力p_u平均流速u_{disk})，我們得到 Bernoulli Equation：

$$p_u + \frac{1}{2}\rho u_{disk}^2 = p_0 + \frac{1}{2}\rho u_1^2$$

對風扇下游 x3 位子(壓力p_0平均流速u_3)和風扇左側(壓力p_d平均流速u_{disk})，我們得到 Bernoulli Equation：

$$p_d + \frac{1}{2}\rho u_{disk}^2 = p_0 + \frac{1}{2}\rho u_3^2$$

所以：

$$p_u - p_d = \frac{1}{2}\rho(u_3^2 - u_1^2)$$

代入(式 3.1.24)，得到推力公式：

$$T = A_{disk}(p_d - p_u) = \rho A_{disk}\frac{u_3^2 - u_1^2}{2} \qquad (式\ 3.1.25)$$

3.1.3. Javaprop & Javafoil 軟體介紹及使用

為適當調整扇葉攻角(α)及C_D及C_L之關係以得到理想上最大推力係數(C_p)，故在此使用 Javaprop & Javafoil，藉此得到最理想化之扇葉模型。此軟體之好處為其互動式介面在設計時可立即觀察到改變個參數後對流場之影響。

3.1.3.1. 翼型規格

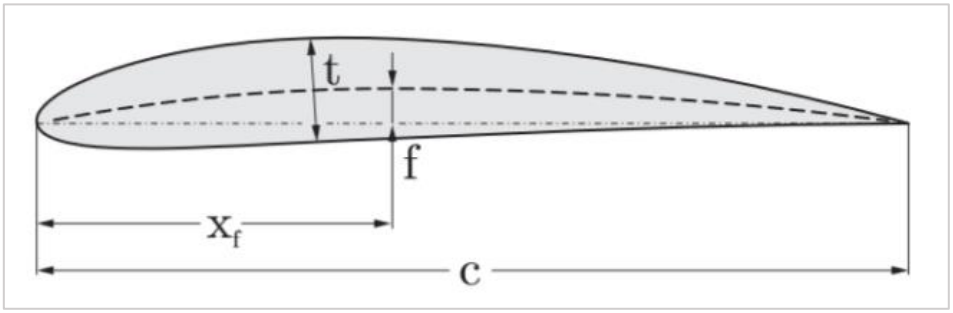
在翼型選擇上，我們採用現有資源以進行分析。NACA(美國國家航空諮詢委員會)將各式翼型以「NACA」四個字母與一串數字(4 碼或 5 碼)組成，此串數字所描述的幾何參數經特定方程轉換後即可得到翼型之精確形狀。而我們採用 NACA 4-digit，每一數字之幾何參數為：

1. 首位數字代表最大彎度占弦長之百分比 $(\frac{f}{c}\times 100\%)$。

2. 第二位數字代表最大彎度距機翼前緣之距離占弦長之十分比 $(\frac{xf}{c}\times 100\%)$。

3. 後兩位數字表機翼最大厚度占弦長之百分比 $(\frac{t}{c}\times 100\%)$。

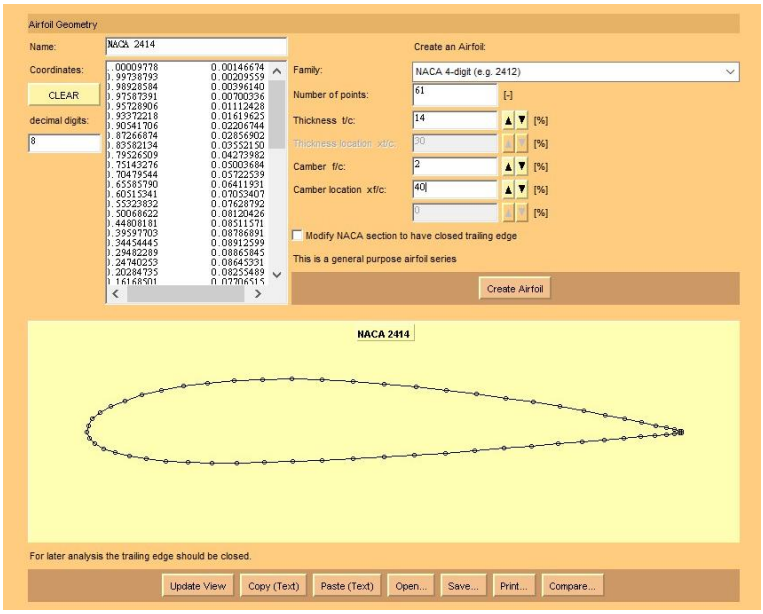
其示意圖如圖 3.1-6 所示。[4]



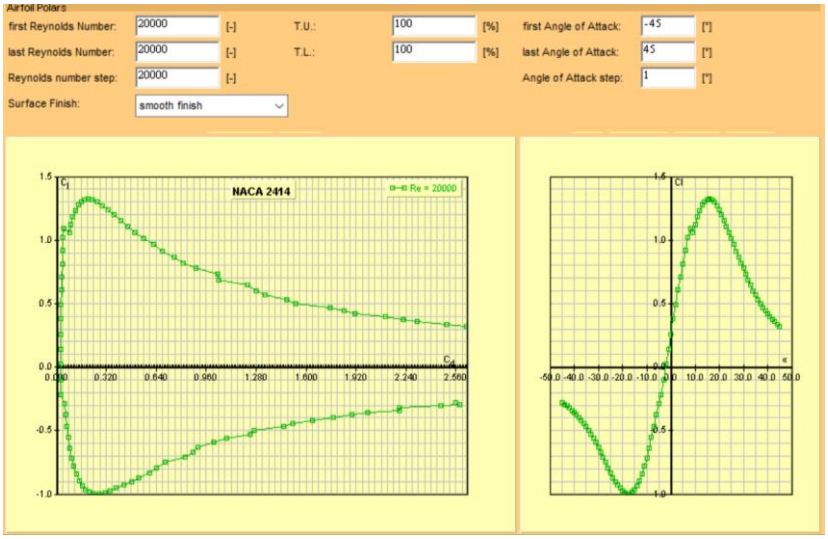
(圖 3.1-6) 翼型截面示意圖[4]

3.1.3.2. 風扇設計過程

首先使用 Javafoil，設定其參數後繪出翼型截面，如圖 3.1-7 所示，並點選 polar 頁籤，設定 Reynold number(Re) (此為假設值，其數值為 20000 以模擬紊流狀態)以及改變攻角，有此可看出其升力係數、阻力係數等參數隨攻角之變化，如結果圖 3.1-8 所示。

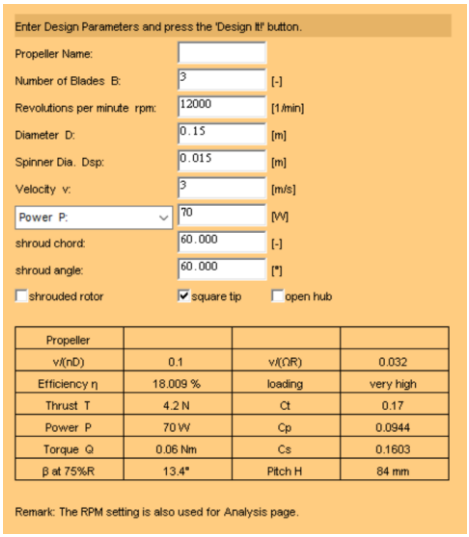


(圖 3.1-7) NACA2414 設定介面



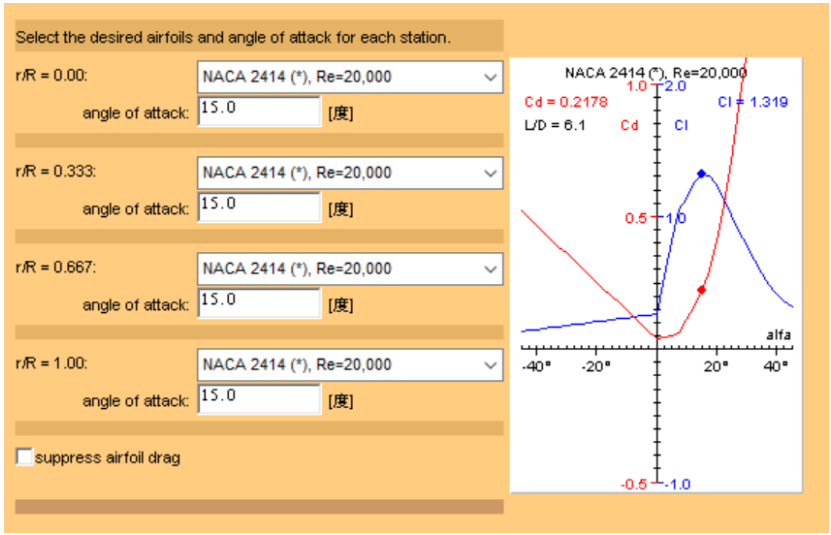
(圖 3.1-8) NACA2414 翼型攻角分析

於上述分析後選定翼型繪出，並輸出翼型之座標檔案，匯入 Javaprop，並設定以下參數：葉片直徑(D) = 0.15m、葉片數(B) = 3、工作速度(V) = 3 m/s (此處因讓車子處持續加速之狀態，故設定相對氣動車一半速度之較高數值)，如圖 3.1-9 所示。

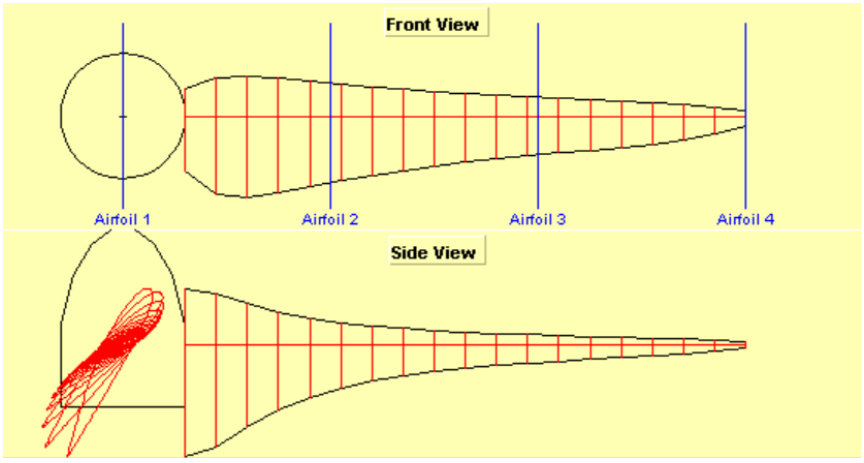


(圖 3.1-9) NACA2414 扇葉參數分析

而在攻角設定上，上節所述之理論而言，其推力隨攻角增大而增加，但同時阻力也會增加，故為增加推力，此處參考升阻比，而升阻比最大值所對應之攻角為我們所需之角度。另外，以整體扇葉而言，為維持各截面的升力平衡，對應之參數為各部的切線速率及其截面攻角，而半徑越大，切線速率越大，故截面越靠近根部(半徑小)，攻角越大，截面越靠近葉尖(半徑大)，攻角越小，以維持各處推力平均。故藉 Javaprop 找出最大升阻比(如圖 3.1-10)，並設定此升阻比所對應之攻角後，Javaprop 配合不同截面所對應之攻角，計算出最佳化葉片之幾何模型，如圖 3.1-11 所示。因此藉上述方法，我們可以由此設計出一理想化之扇葉，並得到其幾何形狀。



(圖 3.1-10) NACA2414 扇葉攻角設定



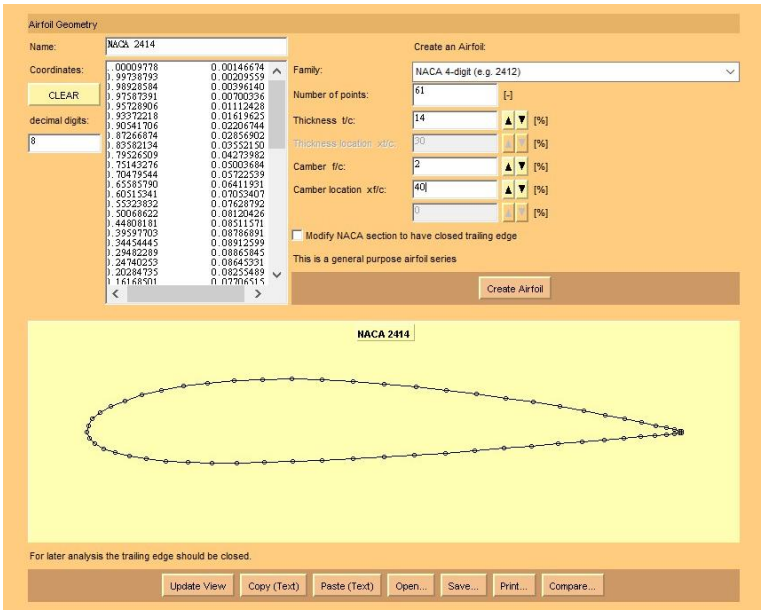
(圖 3.1-11) NACA2414 扇葉最佳化結果(單片)

3.1.4. 各版本翼型比較

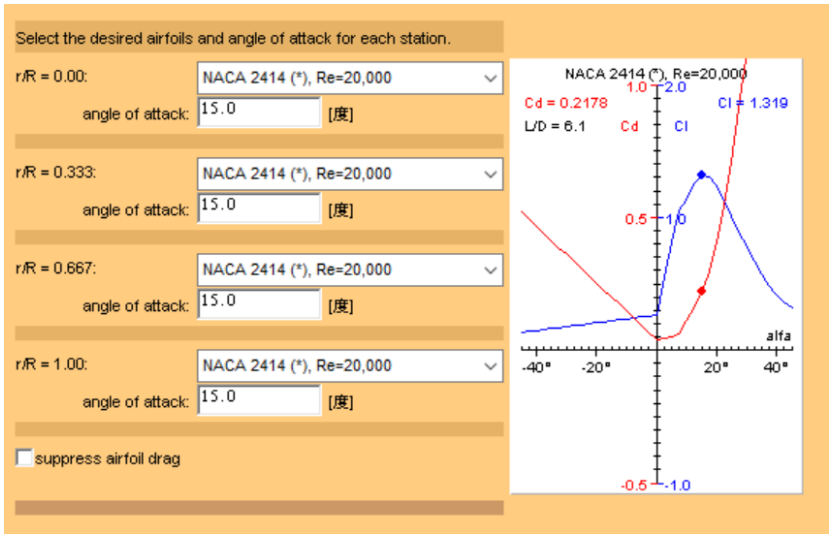
此節將介紹我們歷代所所用及設計之風扇，包括攻角選擇、翼型特型及其差異等。

3.1.4.1. NACA2414

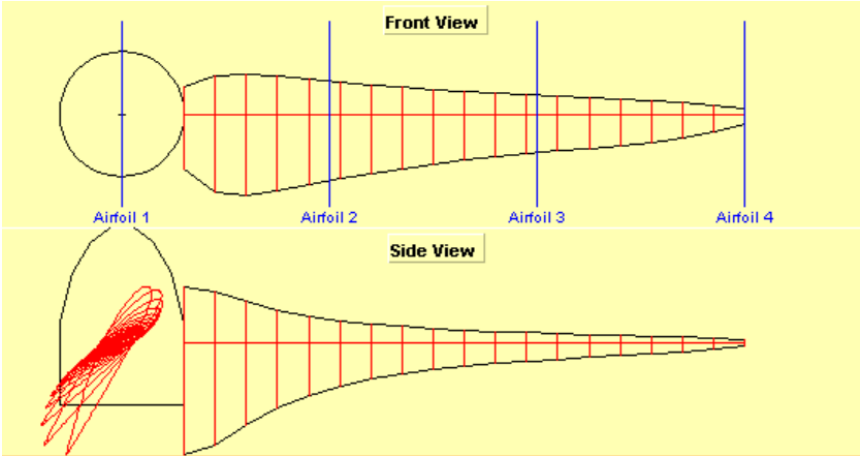
其設定之幾何形狀如圖 3.1-12 所示，其扇葉攻角設定如圖 3.1-13 所示。首先，由圖 3.1-12 可知，攻角在約 15 度時其升阻比最大(= 6.1)，故選擇 15 度做為攻角後所得到之扇葉最佳化結果如圖 3.1-14 所示。



(圖 3.1-12) NACA2414 設定介面



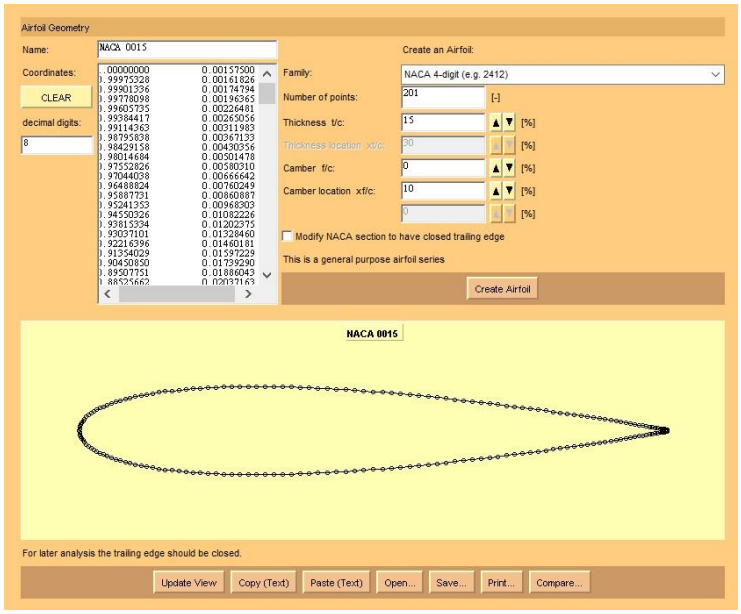
(圖 3.1-13) NACA2414 扇葉攻角設定



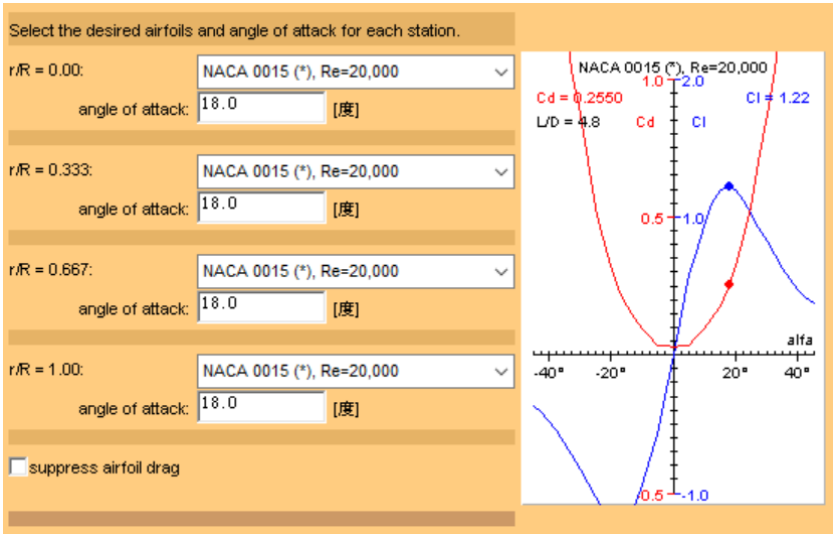
(圖 3.1-14) NACA2414 扇葉最佳化結果(單片)

3.1.4.2. NACA0015

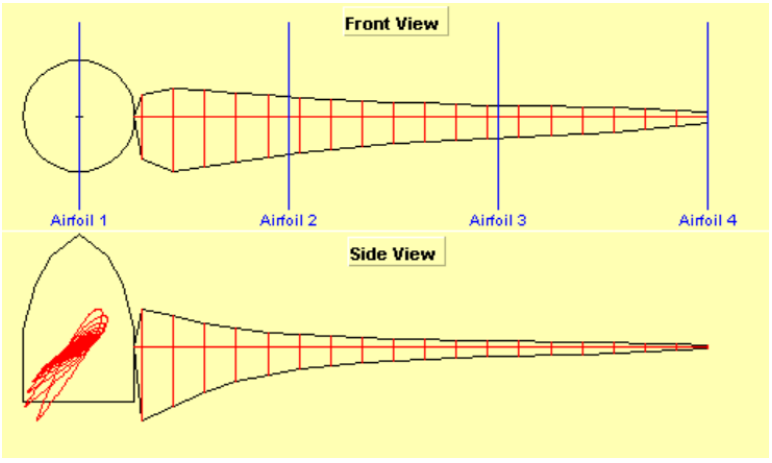
因想嘗試對稱截面之翼型，故將 NACA 之前兩碼設定為 0 使各處無彎度，其設定之幾何形狀如圖 3.1-15 所示，其扇葉攻角設定如圖 3.1-16 所示。首先，由圖 3.1-16 可知，攻角在約 18 度時其升阻比最大(= 4.8)，故選擇 18 度做為攻角後，得到之扇葉最佳化結果如圖 3.1-17 所示。



(圖 3.1-15) NACA0015 設定介面



(圖 3.1-16) NACA0015 扇葉攻角設定



(圖 3.1-17) NACA0015 扇葉最佳化結果(單片)

比較兩翼型(NACA2414 及 NACA0015)之結果。首先，NACA0015 之翼型較 NACA2414 之翼型對稱，且其尾端為尖端，此會於下節解釋。而比較兩者攻角，NACA0015 較 NACA2414 大，但其升阻比較小，推測原因為其截面為對稱型，使所產生升力較小。總之，利用軟體升成最佳化之風扇翼型，然而於比賽中，最後須讓風動車朝相反方向返回終點，因此，我們想設計一風扇，並讓風扇反向旋轉時產生方向相反之推力，將於下節進行介紹。

3.1.4.3. Oval-shaped airfoil

先前使用兩種 NACA airfoil 為原型，並透過軟體 Javafoil、Javaprop 生成風扇，扇葉受風面都受限於單一方向。但根據這次比賽需求，風動車要能「倒車」，因此需要風扇反轉時往車尾反吹的效果。所以我們決定自行設計風扇，使得兩個方向受風可以達到相同量值但方向相反的升力。

設計風扇要考慮的幾個項目是：(1) 尺寸 (2) 扇葉數量 (3) 扇葉截面 airfoil shape (4) 扇葉 twist 的角度。以下我會針對這四個細項說明其重要性，和設計的原由。

(1) 尺寸

表面積越大的 airfoil 能提供越大的升力，但風扇越大，代表受力面離車體底部越遠，會對車底輪子造成力矩而導致翻覆的可能，顧考量車子底盤大小後決定風扇半徑為 15 cm。

(2) 扇葉數量

對風扇來說，扇葉數量越少，效率越高，但也會導致震動和噪音。扇葉數量增加會則會使風扇更穩定，根據實驗，三葉的扇葉和四葉的扇葉有接近的效率[#1]，故我們選用三葉的扇葉來減輕設計負擔並達到良好效率。

(3) Airfoil shape

由於我們希望風扇以兩種不同方向旋轉，可以產生相同升力，所以翼型應該左右對稱，故選擇以橢圓形作為 airfoil。

回顧 NACA 0015 unchambered 的翼型(圖 3.1-18)，會發現其 leading edge 為圓弧狀、trailing edge 為尖端，事實上，將 trailing edge 作為尖端是為了要讓翼型接近真實流線的走向，這種輪廓設計稱做 streamlining，可以有效減少 airfoil 受到的風阻。(圖 3.1-19)

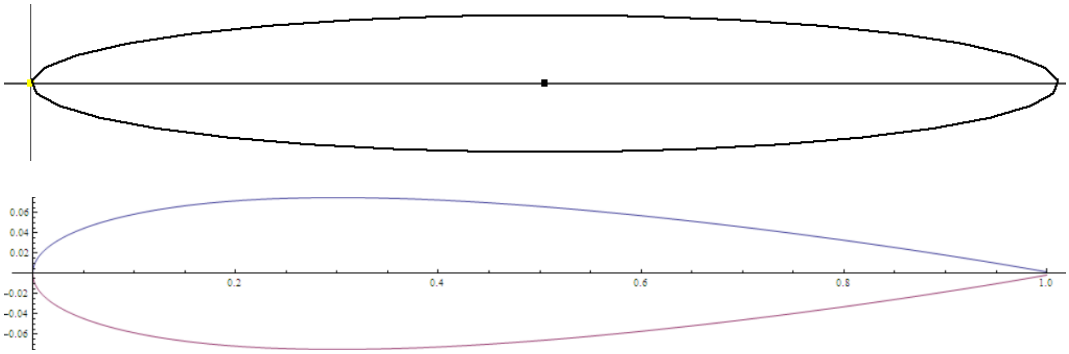


圖 3.1-18 (上)橢圓形 airfoil　(下)NACA 0015 具有尖端 trailing edge [6]

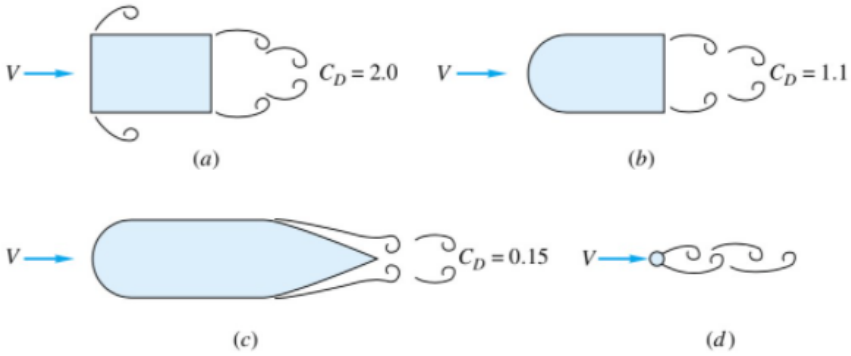


圖 3.1-19 streamlining 示意圖 [7]

然而我們為了對稱需求將 trailing edge 設計為圓弧，雖然會造成扇葉阻力增強，造成馬達扭力負擔，但對於扇葉推力主要貢獻者—升力，並不會有太大的影響，因為升力大小主要由攻角決定。下圖為 attack angle 和 lift 的關係。（圖 3.1-20）

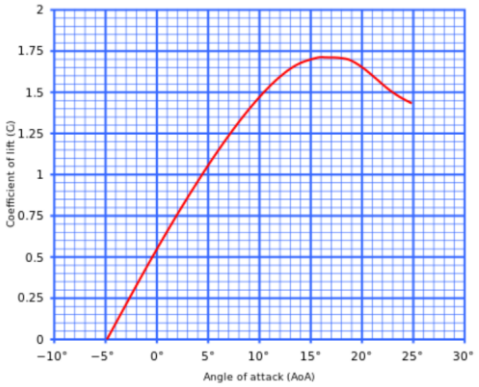


圖 3.1-20 lift coefficient –攻角 關係圖[8]

(4) twist

扇葉從中心到外緣通常會有扭轉，稱為 twist，這個設計是為了讓扇葉在每個位子所受的升力相當，才不會導致受力不均 bending 而斷裂。升力不均的原因是因為根據等式 $v = r\omega$，扇葉離中心越遠處，打向他的流體會有越快的流速，也就會產生越大的升力。

因此我們便思考，怎樣的 twist 才能減少扇葉受力不均的狀況？回顧扇葉上每個截面 airfoil 升力的公式：

$$lift = C_l \times \frac{\rho v^2 A}{2}$$

這裡 A ∝ chord 的長度 l，A ∝ r 該 airfoil 截面到中心的距離，lift coefficient C_l 和 airfoil 的幾何造型有關，雖然可以用複變解出在理想流的升力而算出 C_l，但在實際真實流場以及應用上， C_l 應由實驗結果得到。但一般來說，小角度攻角下 C_l 和攻角做圖接近線性，所以估計 $C_l \propto \alpha$ 。（圖 3.1-20）

根據以上探討，若要 lift 在各位子相同，twist 角度可以看似用$\propto \frac{1}{r^2 l}$的關係所得，然而事實上，當扇葉旋轉時風扇附近的風場十分複雜，加上車子本身在前進，打在扇葉 leading edge 時的入射角並不會是旋轉切線方向，所以攻角不會是 twist 的角度。觀察由先前實驗中 Javaprop 所生成高升阻比的的扇葉，可以發現 twist angle 遠比合理攻角角度大許多，約在 30 幾度，故參考該度數設計 oval-shaped 扇葉的 twist。

3.1.5. 風扇設計

我們根據上個小節所得到的一些原則，設計 airfoil 的參數，再利用 Inventor 將立體形狀斷面混成出來。由於流體力學計算需要許多實驗和模擬，無法以簡單的數學模型計算出最佳化的扇葉設計，所以我們先以直覺性的方式設計參數，事後再利用 Ansys 軟體計算該扇葉在直流馬達工作電壓可得的轉速輸出的推力大小，作為衡量該扇葉好壞的標準。

(1) 第一版風扇

在設計風扇參數的過程，我先以線性遞增的方式設計 chord 長度、twist 角度，但發現斷面混層出來的扇葉有些扭曲，因此再微調 chord 的長度及 twist 角度盡量使扇葉能合理的平滑伸展，達到 streamlining 的效果，風扇厚度取 7 mm 使其強度較強，避免從中折斷。下表為我們第一版風扇設計的參數。（圖 3.1-21）

和中心的距離r(mm)	15	23	31	39	47	55	70
chord中央寬度w(mm)	5	7					5
chord長度l(mm)	20	30	40	48	51.2	52	30
twist角度(deg)	35	30	25	20	15	10	6

考慮到一般 3D 列印一體成形風扇時，必須將風扇面向上列印，就會產生 3D 列印支撐件而造成扇葉表面粗糙使流體阻力增大，所以希望扇葉列印的方向為箭頭所指(圖 3.1-22 左)，故捨棄一體成形的列印方式，將三片分開來列印再進行組裝，如此一來扇葉表面就不會有支撐件，所以十分平滑。（圖 3.1-22 右）

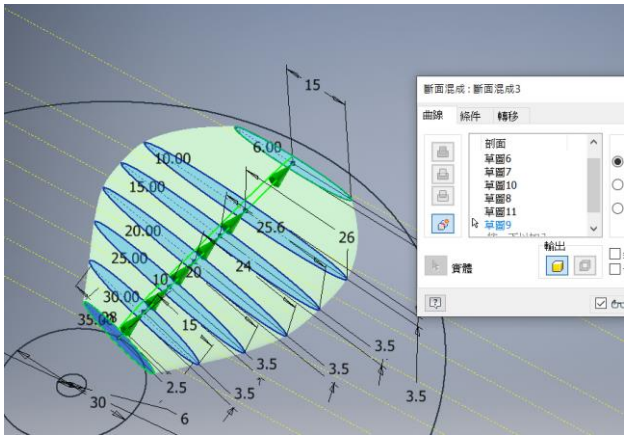
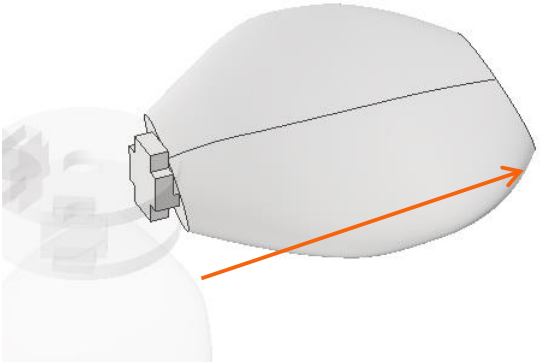


圖 3.1-21 inventor 斷面混成扇葉



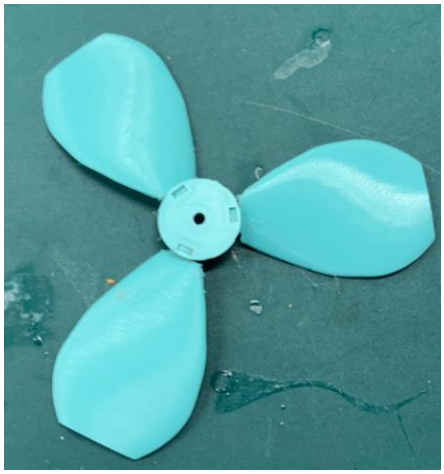


圖 3.1-22 (左) 3D 列印方向示意圖 (右)平滑成品

第一版扇葉最終設計組合檔：（圖 3.1-23）

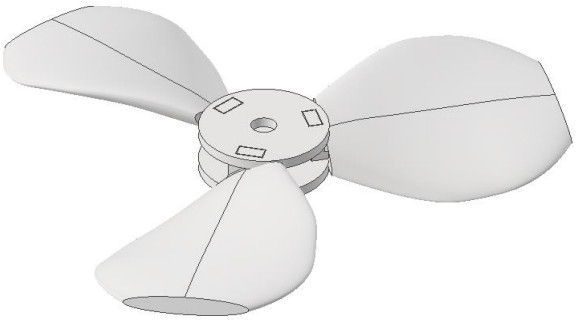


圖 3.1-23 第一版扇葉組合檔

(2) 第二版扇葉

經過實測，發現第一版扇葉有些問題。首先，風扇效能並沒有預期的好，即使轉速調很高，風洞車前進的速度還是太小。

我們在風扇前用手感受它吹出來的風場，發現和其它組比起來，我們的風場較寬廣而弱，連人站在旋轉切線方向都可以感受到些許風力，然而其它組的風扇吹出來的風卻是集中而強勁。這些能吹出強勁風力的風扇都有個特點：扇葉厚度薄、寬度小，因此我判斷問題為扇葉太厚導致 pressure drag force 太大。根據 Ansys 的模擬結果，亦發現第一版風扇旋轉時的流線確實較為平直寬廣、推力弱，而接下來介紹的第二版風扇能達到流線集中的效果。詳細內容於 4.1.1 探討。

另外，原先設計的「組合風扇」在某次實驗非預期的斷裂了。因為組合接頭處強度弱、截面小，成為應力集中點而容易 crack，但受限於尺寸限制，中軸處要塞三個扇葉的接頭就已經十分困難，若要再增大接頭面積不太可能。因此決定捨棄組合檔的構想，設計一體成形的風扇，再使用銼刀去除列印件上的不平滑面。

綜合上述兩個問題，決定以第一版風扇橢圓斷面混層的方式，重新設計參數，使風扇厚度、寬度減小以減少 drag force，twist 增大使風場集中，並將扇葉改為一體成形避免應力集中斷裂。

第二版設計參數：

和中心的距離r(mm)	6	14	30	46	69
chord中央寬度w(mm)	2.4				
chord長度l(mm)	16	20	30	34	10
twist角度(deg)	24.2	42	35	32	20

第二版設計參數組合檔：（圖 3.1-24）

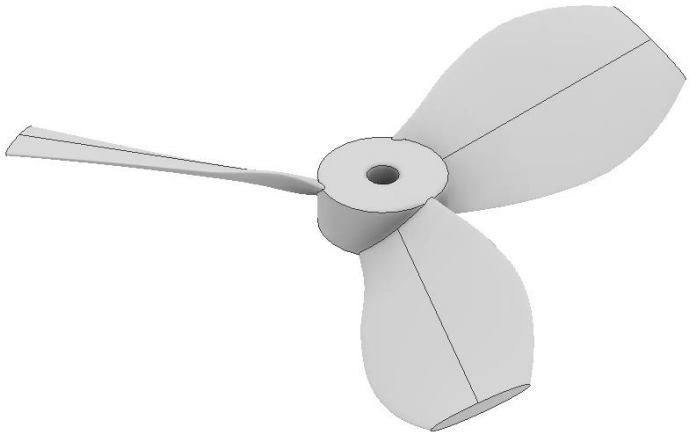


圖 3.1-24 第二版風扇

(3) 第三版扇葉

因為改良後的第二版風扇效果不錯，小轉速就可以產生不錯的推力，所以決定以第二版風扇的修改方向再改良風扇為厚度更小、寬度更小、twist 更大的扇葉。後來在 Ansys 上模擬也得到更好的效率，變使用它作為最終風扇版本。第三版參數：

和中心的距離r(mm)	6	14	30	46	69
chord中央寬度w(mm)	3	2.4			
chord長度l(mm)	13	18	20	20	14
twist角度(deg)	45	45	42	40	30

第三版設計圖：

圖 3.1-25 第三版風扇

3.2. 轉向機構

轉向機構為控制車體方向的重要機構，轉向若能及時且穩定將能降低控制的難度。以下將說明本組轉向機構的概念發想，以及後來做出的兩個版本。

3.2.1. 轉向機構概念

針對轉向，我們想到了三個方式，分別為:

(1)單輪轉向:

前輪使用單輪，可以直接用伺服馬達控制前輪的轉向，進而帶動車體轉向，這個方法的機構設計比較簡易，較容易組裝、維修，缺點是三輪車穩定度較低，可能會出現傾倒、翻覆的情形。

(2)阿克曼轉向機構:

阿克曼轉向機構的旋轉半徑較小且穩定度較高，缺點是連桿機構設計較為複雜，需考慮連桿能否承受伺服馬達施加的應力，且伺服馬達未配置於底盤正中間可能導致車體的重量分配偏向一邊。

(3)風扇轉向

用伺服馬達使風扇轉向，是比較直覺的想法，風扇繼續送風就能改變推力方向，讓車體轉向，且將風扇轉向也可以用於比賽中的倒車環節，將風扇旋轉180 度即可實現倒車目標。缺點是並非驅動輪子轉向，可能導致車體不穩定，出現翻覆的情形。

下表整理上述三種轉向方式之優缺點比較：

方式	優點	缺點
單輪轉向	1.機構設計簡易 2.零件不容易相互干涉 3.累積誤差小	1.穩定性較低 2.較容易傾倒、翻覆
Ackerman Steering	穩定度高	1.連桿受力大 2.轉向角度設計困難 3.組裝、維修不易 4.導致車體重量分配不平衡
風扇轉向	機構設計簡易	並非驅動輪子轉向可能導致車體不穩定、翻覆

3.2.2. 第一版設計

在衡量以上方案的優缺點後，我們第一版決定使用單輪轉向，用伺服馬達SG90 驅動前輪轉向，並在設計車體時注意配重問題，把車體重心降低以避免傾斜、翻覆的情形發生。

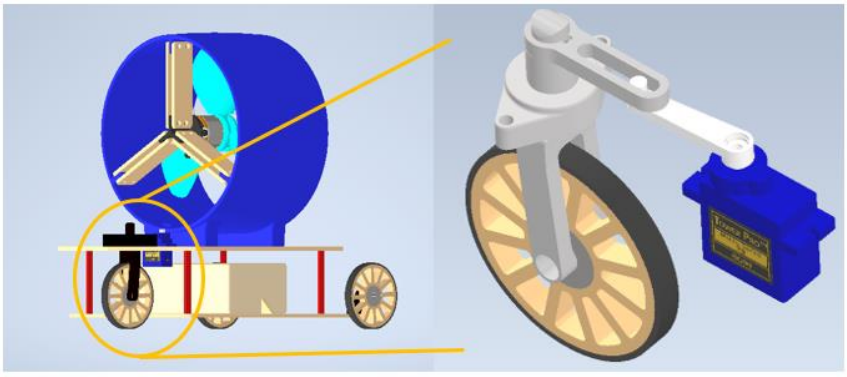


圖 3.2-1 第一版轉向機構

針對單輪轉向機構的設計，我們的採用單一前輪，且前輪及轉向軸皆有使用軸承加強固定，以加強穩定性、減少阻力和減少彎矩。而轉向機構的設計如上圖 3.2-1 所示，將伺服馬達延伸出一桿件，卡入轉向軸上的槽，當馬達轉動桿件時，槽亦會帶動轉向軸轉動，連帶地，前輪也會跟著轉動。而在得到馬達轉角和前輪轉角間的關係後，可以利用這個轉向機構做換線時的大幅度轉彎和循跡時的小幅度轉向角微調。

在第一版的設計中，我們發現了幾個問題。由於轉向角與馬達轉角並非線性關係，在控制時可能出現誤差。當馬達轉較小角度時，前輪轉向角度變化較

大；而當馬達轉至 30°以上時，前輪轉向角度的變化並不明顯，使得最大轉向角度被壓縮了。

3.2.3. 第二版設計

第一版設計經過期中測試後，為了修正問題並使控制能夠更加準確，而有了第二版設計，以下詳細說明。

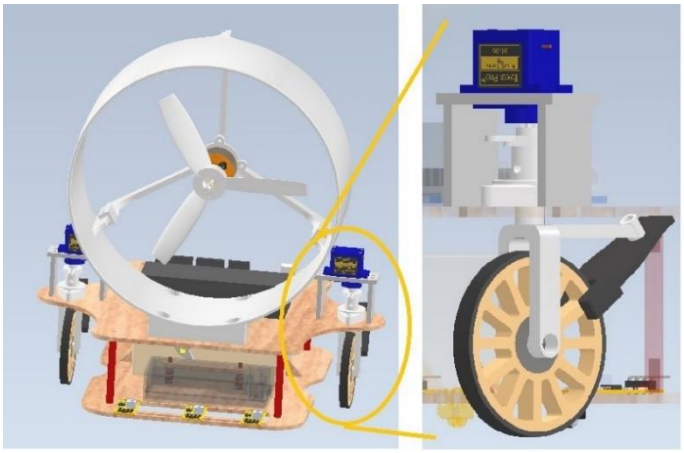


圖 3.2-2 第二版轉向機構

在第二版的轉向設計中，將伺服馬達與轉向軸設在同一直線上，將購買伺服馬達時所附的配件鎖固於轉向軸上，可使輪子的轉向角度與馬達輸出角度同步，且不必透過連桿機構傳動。

而考量到期末測試時場地需要爬坡，在坡面上需要更及時的轉向反應。原來第一版的前輪單輪轉向，前進時能快速修正方向，但在倒車時，轉向輪變到後側，轉向就需要多一些緩衝時間和距離，在控制上便要再準備一組和前進時不同的參數，如此會增加測試的不確定性。所以，第二版設計將轉向輪增為兩個，並設於兩側的車長一半處，前後各配置一個萬象輪以平衡車身。如此一來，轉向時車身便能夠直接平移，不需要迴轉半徑，角度也較精準。

3.3. 車輪與煞車.

3.3.1. 車輪設計

車輪是最直接接觸地面的部分，不只需要支撐起車體，也是能夠前進的關鍵，以下說明本組車輪的設計。

圖 3.3-1 本組製作之車輪

上圖為本組製作之車輪，外徑為 53 mm。以密集板為主要材料，為了增加強度使用兩片密集板疊合，並在最外側貼上橡皮墊以增加輪胎的摩擦力。中央則放置了軸承，降低車輪本身的摩擦力，使車身前進更滑順。

另外，車輪也被我們當作 encoder 使用。輪框中的輻條共有 12 條，可作為光柵，在車輪前裝配紅外線計數器即可計算輪子轉的圈數，進而推算車體前進的距離。也許 12 條這個數目算出的解析度不夠高，但足以作為輔助控制的一項數據。

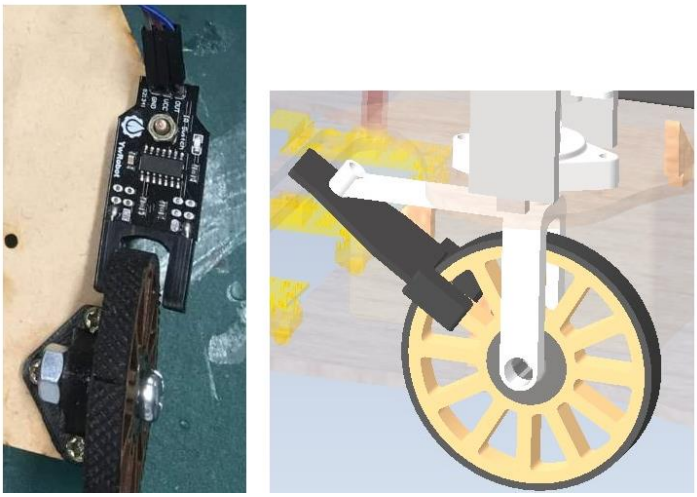


圖 3.3-2 Encoder 之裝配(左)為第一版(右)為第二版

在紅外線計數器的裝配上，如圖 3.3-2。第一個版本我們將它水平裝設於後輪的前方，且高度與車輪的圓心等高。由於第二版的車體並沒有像第一版那樣的固定輪，所以我們將第二版車輪的轉向軸延伸出一個桿件，用來固定紅外線計數器，且仍然與車輪的圓心在同一直線上。如此一來，紅外線計數器便能發揮作用並與車輪一同旋轉。

3.3.2. 煞車設計

期中賽道之煞車需要吸收車輛前進之動能，以達到在特定區域停下的功能。不同於期中賽道之測試要求，期末的煞車系統還需要使得車輛在斜玻上停滯，其中所增加的摩擦力需求會在設計分析部分進行分析。

初步構想有三形式:1.夾住車輪達到煞車的目的；2.設計包覆煞車皮脂桿件，透過桿件點地的方式消耗動能；3.輪胎連接一同軸之齒輪，藉由驅動一反向力止住輪胎旋轉。由於 1.和 3.空間配置的方式有限，最終我們選則讓一桿件觸地的方式做煞車。

圖 3.3-3、夾住車輪的設計靈感源自於腳踏車煞車[9]

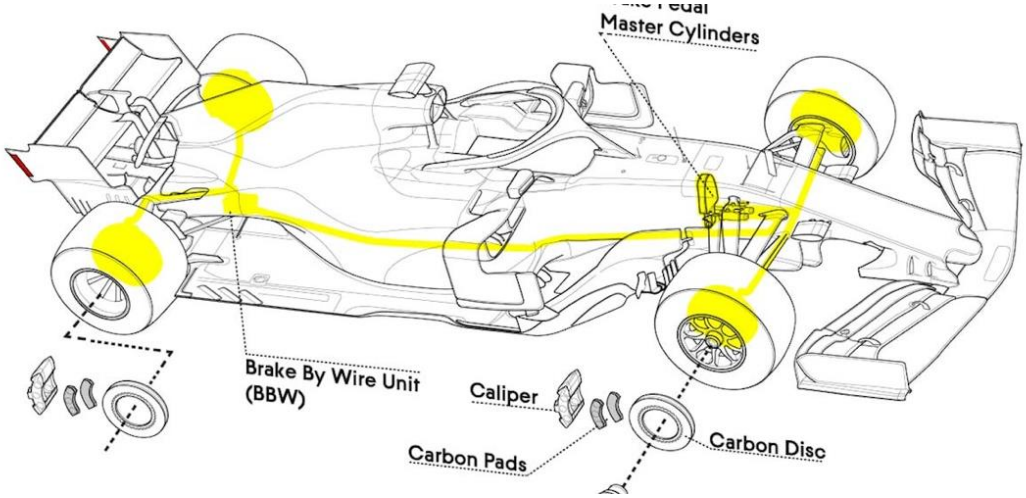


圖 3.3-4、靈感來源自方程式賽車 internal brake[10]

起初考慮到車子會需要在斜坡上轉彎，若只有單一點處地會造成車身側傾，而為避免車身向前點頭和車尾向後沉，我們希望煞車點接近全車質心位置。因此衍伸構想有二：齒輪組連桿機構用單一馬達驅動兩桿件同時點地(圖如下)和凸輪點地、彈簧復位機構。而最終由於我們希望簡化機構、減輕重量選擇彈凸輪、彈簧復位的方案。

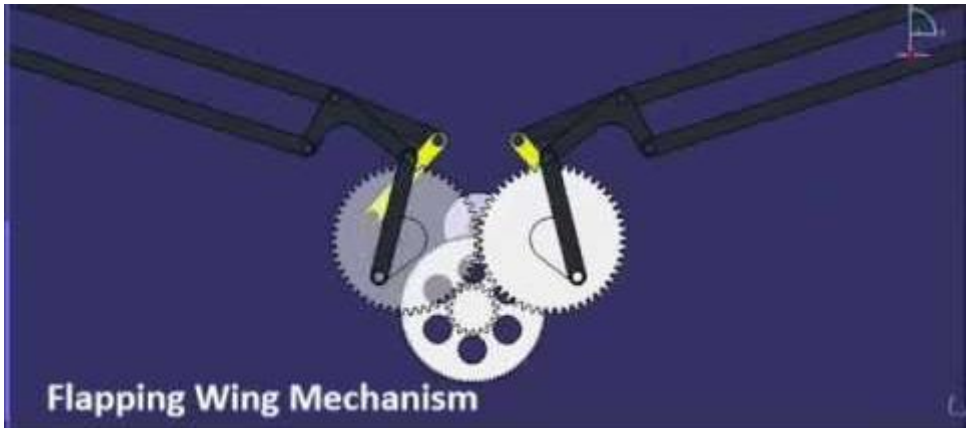


圖 3.3-5、齒輪組煞車概念來自鳥類玩具的振翅機構[11]

我們最終的煞車系統如下圖，煞車桿有二銅柱穿過底板，限制除垂直向平移之外的自由度；銅柱上套有兩個彈簧提供復位的力；頂版固定 SG-90 馬達接上一搖桿。

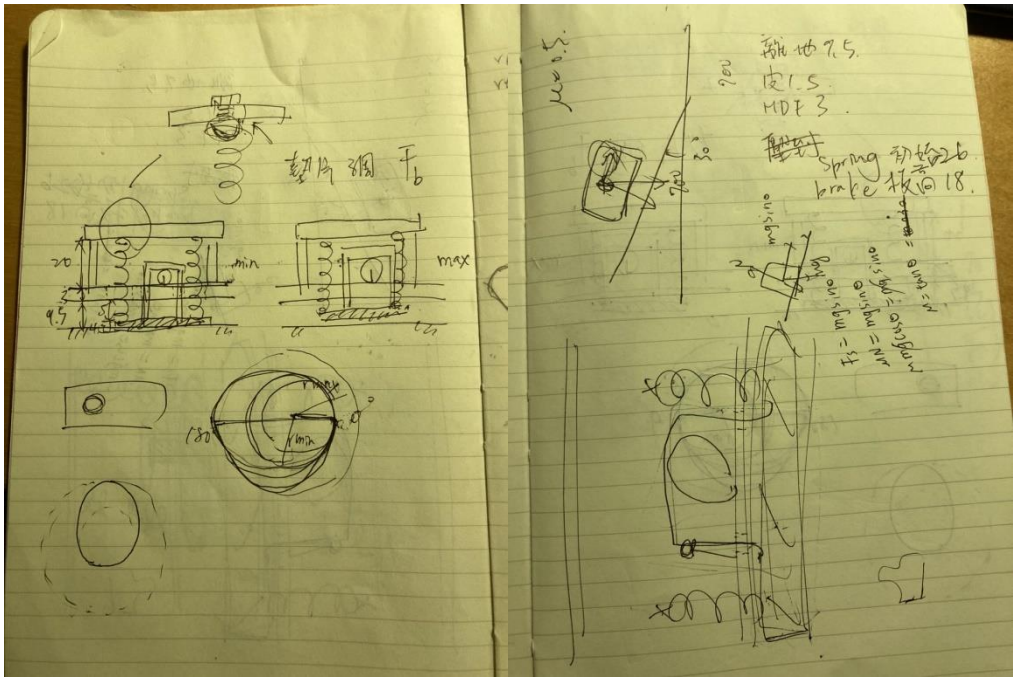
彈簧選用:

此系統彈簧規格要求為需承受彈簧桿本身的重量以達到復位的效果，彈簧桿本身質量估計僅 20g，我們使用上學期用在撞球車上的彈簧(k =50 gw/cm)，即可達到很好的復位效果。

凸輪形狀:

原先的設計中搖桿為一完整凸輪，不過礙於會與頂板干涉，凸輪桿件只有與煞車桿接觸的地方有凸輪輪廓。在應力分析後我們將其他部分挖除，達到結構穩定、不干社、減輕車重的效果。

圖 3.3-6、煞車系統配置



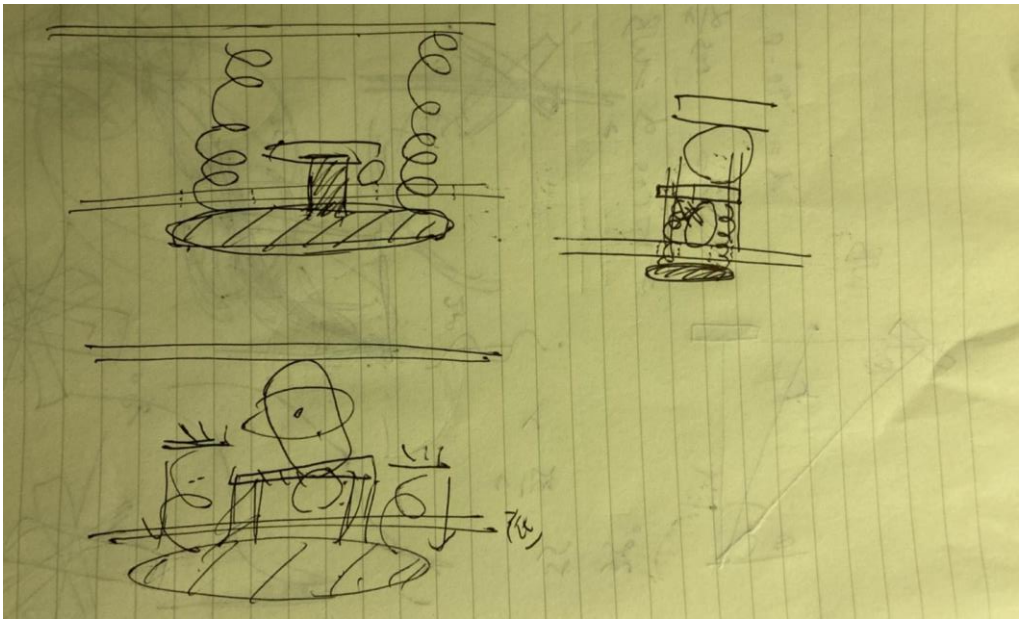


圖 3.3-7、凸輪搭配彈簧機構之發想手稿

3.4. 車體

3.4.1. 第一代車體

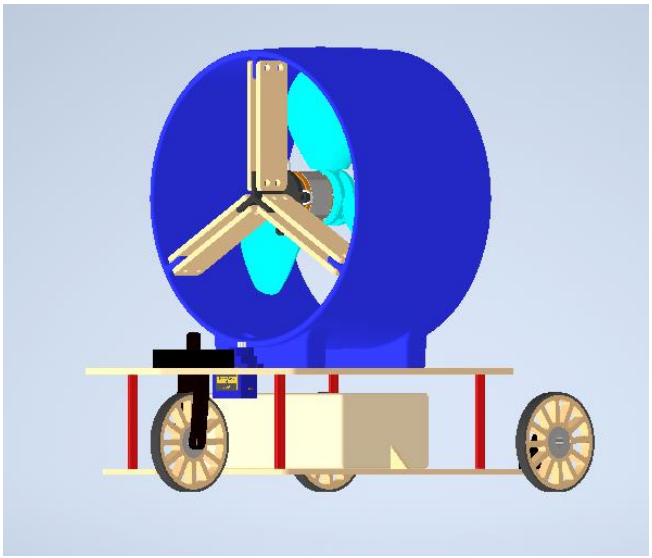


圖 3.4-1 第一代車體示意圖

車體部分大致分為上下兩層，以下分別說明兩層的配置：

(1)下層中央放置鋁箔包車手及控制板，於底板後方安裝兩個後輪，煞車則設置在車手前方。另外，由於此次測驗目標包含循跡功能，所以在底板的最前後兩端皆設置了紅外線感測器。

(2)上層後半放置風罩及風扇組，為本車的動力來源；前方則安裝有一個前輪及其轉向機構。

為有效利用空間及壓低車底重心，所以將大部分需固定的重物放置於下層，並將較需要活動空間的風罩風扇組還有轉向機構至於上層，如此上下層便不干涉，能各發揮作用。而在輪胎部分，為了減少阻力所以使用了軸承，使車體能完整利用風力推進。在前輪部分，希望能降低轉向軸與車體的摩擦，也裝設了軸承。

3.4.2. 第二代車體

由於期末設計有了爬坡的需求，在相同推力之下，車體設計需要盡可能輕量化。我們以這樣的設計理念進行車體整合，減少非必要材料使用，避免過度設計。

期末的車體設計承襲了期中的空間配置，一樣有上下夾層，兩層的所放置的大致元件也相同。以下分頂、底板分別說明圓件配置之緣由:

(1)頂板:

頂板元件有風扇組合，電池轉向馬達以及控制板。考量到要始重心集中，我們將質量較大的電池置中，並且使風扇組合貼齊電池。輪胎轉向由於改由馬達直接連結，因此必須放置在二輪正上方位於車體兩側。

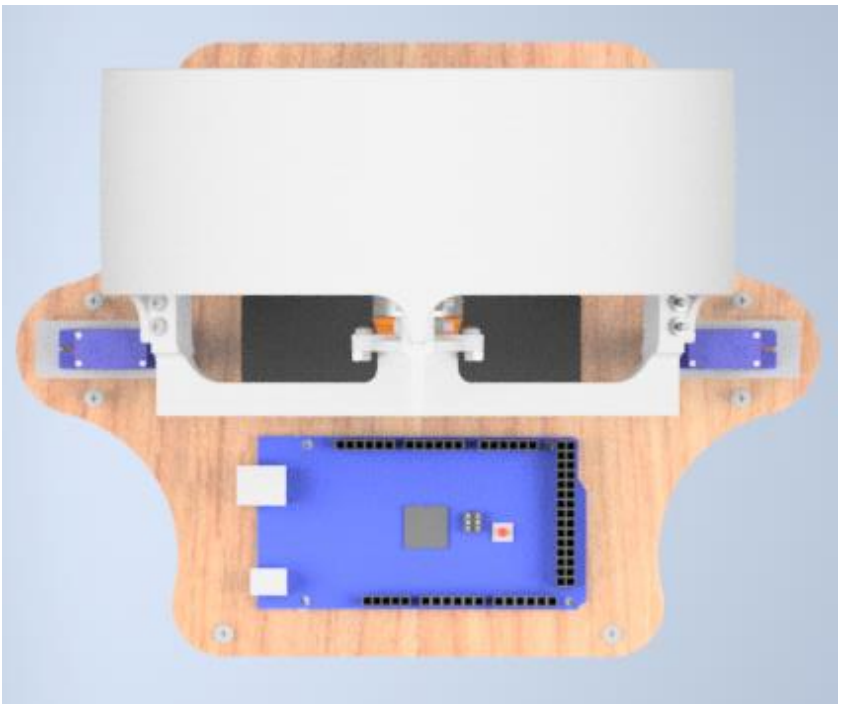


圖 3.4-2、頂板配置

(2)底板:

頂板設計中重量偏在前面的位置，底板設計為平衡車體重心位置，將鋁箔包位置移至偏後方，讓重心位置可以在輪軸上方。

煞車機構的位置選在重心位置附近，是為了避免車身在煞車狀態下前後傾倒。由於期中測試時發現可能需要依據實際動態調整紅外線感測器數量和位置，我們在底板切出長形可調鎖槽，方便隨時更改感測器的配置。

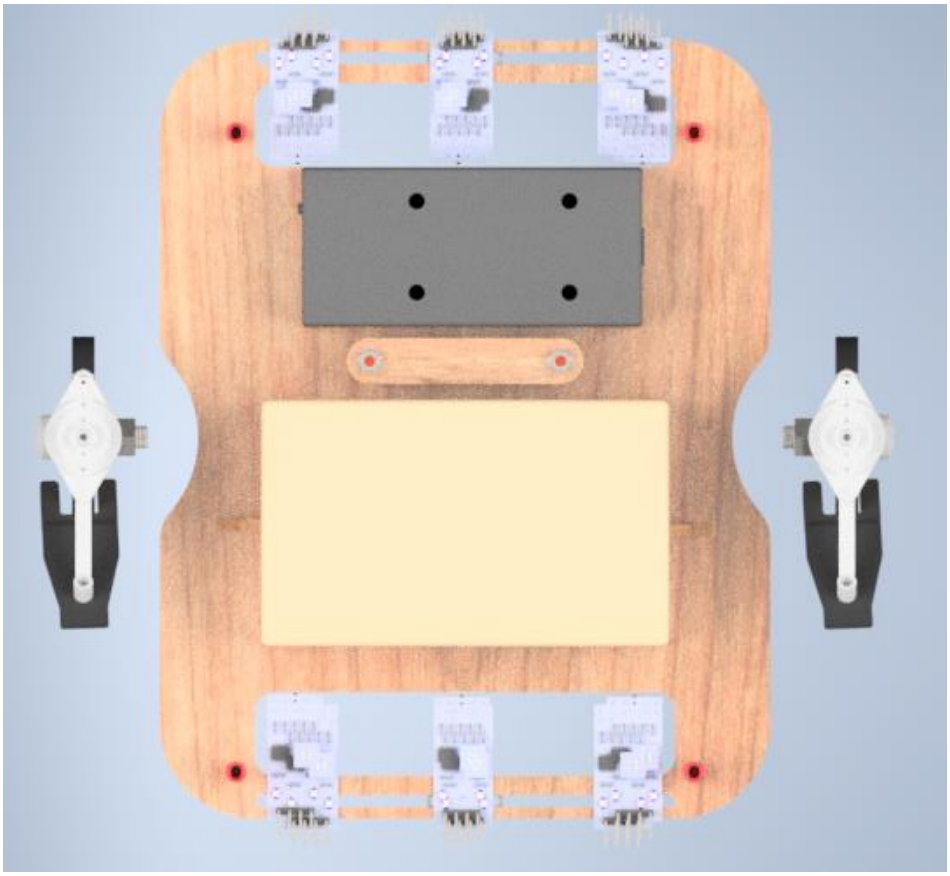


圖 3.4-3、底板配置

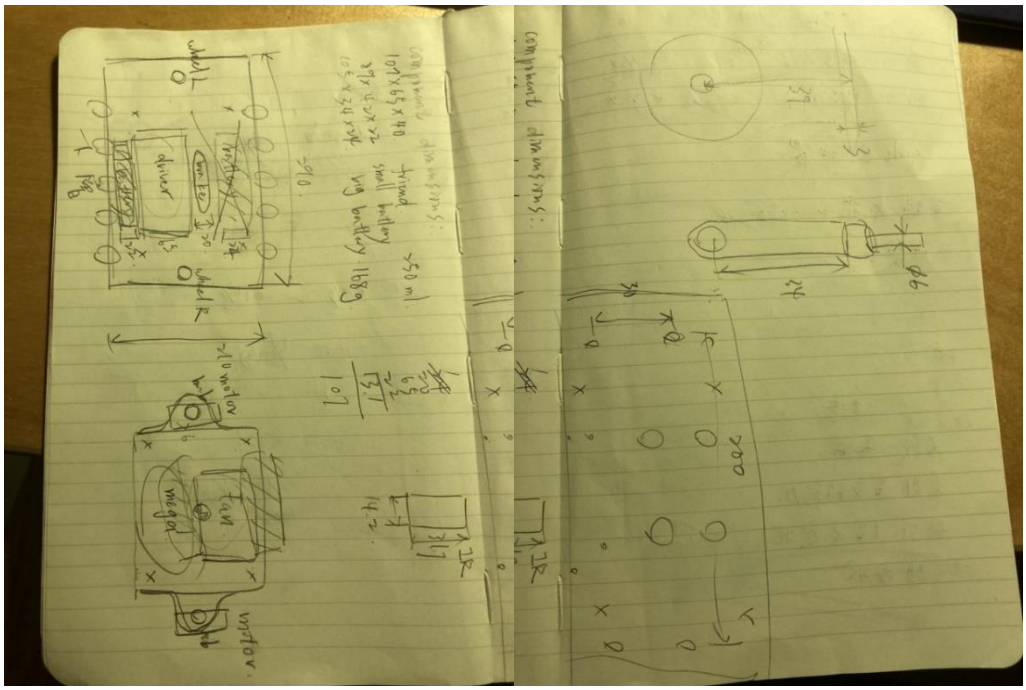


圖 3.4-4、設計手稿

3.5. 機電與控制系統

3.5.1. 控制系統配置

3.5.1.1 循跡模組

(1) 構想

循跡的控制流程可以用控制系統類比：Input 為轉彎角，Output 為車子所在的位子，Error 為偵測到車體和循跡線的差異。在循跡過程中，會根據差異量值進行轉向改動，若車體在循跡線左側，車體需向右調整，在循跡線右側，車體需向左調整，當差異大時，轉動幅度大，差異小時，轉動幅度小，為 P 控制。調整後，持續讀值觀察改動是否足夠，若偏離循跡線的狀況仍存在，持續調整轉向，直到車子行走至黑線中央，為 PI 控制的過程。

實作循跡有兩個方法，1. 紅外線感測器排成陣列感測黑線 2. 影像辨識找到黑線位子。前者的優點是搭建單純、程式較簡易，開發版 CPU 需求低，Arduino 即可，缺點是精度受限於紅外線陣列數量（車身寬度限制下最多只能放置 6 塊），且電子元件本身有誤差，所以要讓循跡模組正常運作需將每塊紅外線感測器校正，要做較多繁瑣的實驗。影像辨識的優點是精度高，但需使用 CPU 較強的 Raspberry pi 才能提供足夠運算資源，入門不容易。所以我們最後決定先以紅外線感測器作為循跡模組。

(2) 實作

紅外線感測器的運作原理是：每塊紅外線模組上都有一個發射器和一個感測器，發射器發射紅外線，打到物體時，物體吸收率影響反射回來的紅外線量值，此時感測器上的接收端接收反射訊號轉換為訊號輸出值，由於顏色越深的物體對電磁波吸收率越高，所以當訊號較低時(analog read 讀到較大的數字)，可以判斷地面為深色，即為黑線所在。以下介紹以紅外線模組陣列實作循跡的方法。（圖 3.5.1）

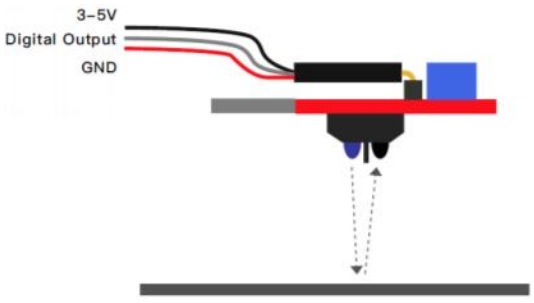


圖 3.5.1 紅外線接發器示意圖 [12]

使用左、中、右三個紅外線模組，可以用下圖 3.5.2 配置達到最基本的循跡功能，透過這些紅外線模組讀值的 3 個 bits，描述 23 種運動形式，如下表所示。（1: 感應到黑線高電位，0:感應到白色地板低電位）

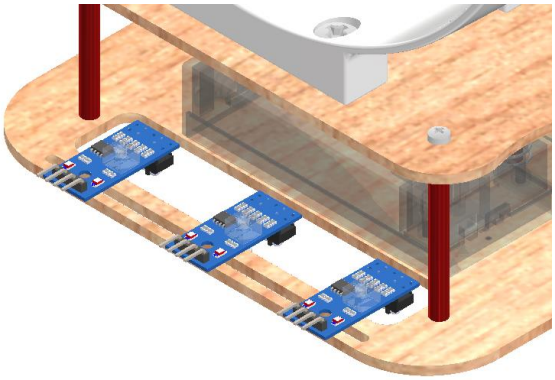


圖 3.5.2 三路紅外線循跡模組示意圖

IR 左	IR 中	IR 右	控制策略
0	0	0	前進
0	0	1	快速右轉
0	1	0	前進
0	1	1	慢速右轉
1	0	0	快速左轉
1	0	1	忽略
1	1	0	慢速左轉
1	1	1	停止

雖然三路循跡可以做到很基本的循跡，但實測上發現其精準度不夠。因為場地黑線線寬很窄，同一時間通常只有一個 Sensor 可以讀到黑線值。當車子判斷需左轉，但左轉幅度還不足以讓車體擺正時，車子可能就會開出線外，而我們理想上的控制最基本的要求是循跡模組有效感測範圍必須包含黑線，一旦離開感測範圍，所有循跡控制失效。雖然這個問題看似可以透過轉更大的角度來解決，但實際上，轉更大的角度會使車晃動過度劇烈，亦離開循跡線。

因此唯一的解決辦法是提高循跡陣列的精準度，故我們採用五路循跡。五路循跡配置如下圖 3.5.3，因為多了兩塊紅外線感測器，車子能知道黑線距離車體中央多遠，因此，當車體距離循跡線較遠時，可以以較大的轉角將車體拉回中間。概念上有如 PID 中 P 控制，而循跡持續監測黑線位子進行轉向調整有消除累積誤差的功能，為 PI 控制。車子倒轉時，亦使用五路循跡，裝在車體後側。

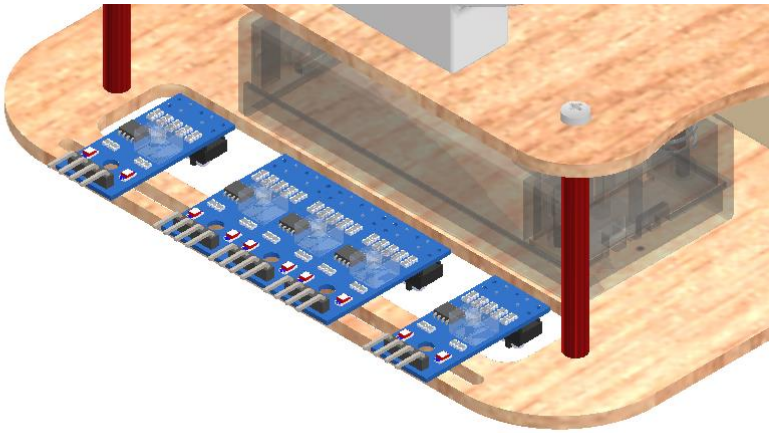


圖 3.5.3 五路紅外線循跡模組示意圖

(3) 控制

以下為我們針對五路循跡所做的控制介紹：

(i) 定義讀取黑線與否。

以前側循跡模組為例，定義 5 個 bits 分別代表最左邊(FLL)、左邊(FL)、中間(FM)、右邊(FR)、最右邊(FRR)是否讀到黑線（F 為 front 縮寫、L 為 left、M 為 middle、R 為 left）。這 5 個 bits 的 on/off 則由各自 analogRead()讀值和實驗 4.5.1 中各個紅外線對黑線白色背景讀值差異定義的閾值決定。

```
#define FLL 0
#define FL 1
#define FM 2
#define FR 3
#define FRR 4
const int IR[11] = {A14, A0, A8, A9 ,A15}
const int IRthres[11] = {150, 270, 720, 350, 300}

IRvalue[FLL] = (analogRead(IR[FLL]) > IRthres[FLL]);
IRvalue[FL] = (analogRead(IR[FL]) > IRthres[FL]);
IRvalue[FM] = (analogRead(IR[FM]) > IRthres[FM]);
IRvalue[FR] = (analogRead(IR[FR]) > IRthres[FR]);
IRvalue[FRR] = (analogRead(IR[FRR]) > IRthres[FRR]);
```

(ii) 定義轉彎角。

```
turn_angle = (-23)*IRvalue[FLL] + (-15)*IRvalue[FL] + (0)*IRvalue[FM] +
(15)*IRvalue[FR] + (23)*IRvalue[FRR];
```

轉彎角大小由車子偏移循跡線的距離決定，所以當最外側的紅外線偵測到黑線，表示偏離情形較嚴重，加權值為 23，內側表示偏移情形較不嚴重，加權值為 15。此外，因為有時車子會遇到橫向黑線，為了忽略該狀況，當紅外線感測讀值超過三顆，標示為忽略。

```
if((IRvalue[FLL] + IRvalue[FL] + IRvalue[FM] + IRvalue[FR] + IRvalue[FRR]) >= 3){
    turn_angle = 0;
    Serial.print(" bad_line");
}
```

3.5.1.2 編碼器及路徑控制

(1) 構想

由於這次比賽中，車子直走行徑的距離有限制，且在起始區和換軌道時，車子需在沒有循跡線的情況下，找到下一段循跡線，因此車子需要得知自己行經的路徑為何。

要讓車子知道自己的路徑有三種辦法：1.計時。根據需求，規劃行走或轉彎特定時間。車子僅能透過時間瞭解預先歸劃的路徑位子，無法應對現場突發狀況。比方說若轉彎時輪胎皮卡到，導致自旋，風洞車會繼續盲目的以預先規劃方式前進而完全脫離比賽場地。2.GPS 定位。精準定位系統，但 GPS 模版昂貴。3.編碼器。計算車輪兩輪旋轉圈數換算行徑距離，較不精準，且無法計算到車輪打滑的情況，與實際狀況有誤差。

為了降低成本及一定的精準度，我們選用編碼器作為路徑控制。由於編碼器本身受限於光柵數和取樣頻率，會有些許誤差，我們考量到整個行徑路徑中，主要兩大直線路徑有較可靠的循跡線可以作為依循，切換跑道時，亦可監測黑線判斷是否抵達目的，所以我們決定以循跡線作為主要位子提示、路徑控制為輔，以達到有效的路徑控制。

(2) 實作

一般市面上的編碼器通常需安裝在馬達上(構造如下圖 3.5.4)，然而我們的風動車的車輪並非馬達驅動，而是受到風扇推力推動，在這個狀況下，若將編碼器安裝於被動旋轉的輪軸上，可能導致輪軸扭轉阻力增加使得風洞車不易前進。

圖 3.5.4 編碼器 [13]

因此我們決定模仿光學編碼器的原理，自己製作編碼器。光學編碼器的原理是使用開槽圓盤作為光柵，其中一面有 LED，另一面則有光學電晶體，圓盤旋轉時會中斷光徑，因此產生的脈衝可指出軸心的方向和旋轉。如下示意圖。（圖 3.5.5）

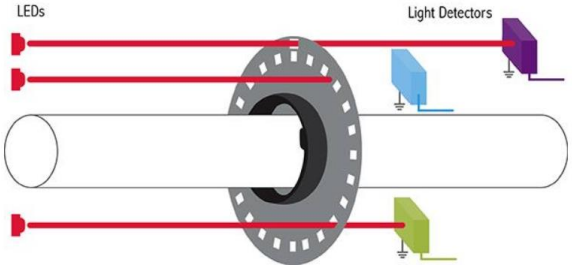


圖 3.5.5 光學編碼器示意圖［14］

我們將輪子挖洞作為光柵圓盤（不另外設計光柵裝設於輪軸，以減少組裝複雜性。）在輪子上設計各個相位角的細孔(圖 3.5.6 左)。LED 及電晶體則用紅外線收發器取代，由於其紅外線是打向自己裝置上的接收端，和上一小節所介紹的紅外線模組使用的 TCRT5000 不同，因此該紅外線收發器又被稱為「計數器」。安裝方式如下圖 3.5.6 右。

在程式控制上，由於光柵每轉動 30 度，紅外線就會經歷「接收->遮蔽->接收」的轉換過程，因此 digital read 會讀到兩次脈衝，此時即可換算得知輪子轉了 30 度，再由圓弧長關係式 $l = r\theta = 26.5 \times 30/180 \times \pi(mm)$ 得知車子向前走了 13.88 mm 的距離。

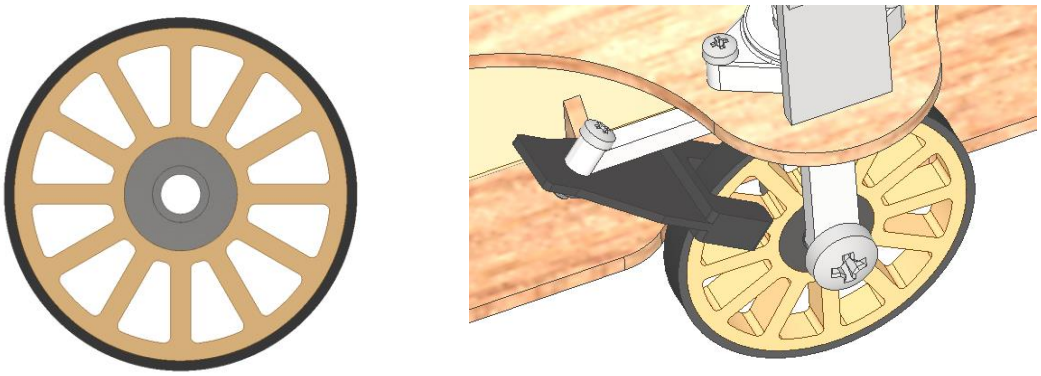


圖 3.5.6 （左)有光柵的輪子示意圖（右)計數器安裝方式

(3) 控制

在上述公式，我們知道行徑距離可由計數器換算，然而實際控制上，由於車子轉彎時左右輪行徑路徑不一致，若要計算左右輪各自圈數並換算為行走角度，又受限於光柵數量太少造成的誤差，因為一個光柵 13.88mm，若兩邊輪子皆誤差 13.88mm，車子輪距 200 mm，換算成角度 $= tan^{-1}(13.88 \times 2/200)$~8 度。當車子在循跡時，因為搖晃劇烈，該誤差累積會變得很可觀。所以我們實作上為：循跡時對搖晃情形加以校正，切換跑道要控制轉彎時，以煞車轉向後、放開煞車的方式讓車輛固定單邊輪子自旋，此時，只要紀錄有轉動那側的 encoder 值，就可以達到指定轉向角的功能了。

(a) 循跡：

只讀取一邊 encoder，當車子變換轉動角度的次數多，表示晃動較劇烈，但我們要測量的只有直線距離而不包含那些搖擺路徑，所以將 encoder 讀到的值扣除校正。

```
void right_encoder(){
    int bit = digitalRead(encoder_r);
    if(bit != right_state){
        right_state = bit;
        right_count += 1;
    }
}

...
while(1){
    right_encoder();
    if(right_count >= 6.8){
        break;
    }
    ...
    ...
    turn_angle = (-23)*IRvalue[FLL] + (-15)*IRvalue[FL] + (0)*IRvalue[FM]\
                + (15)*IRvalue[FR] + (23)*IRvalue[FRR];
    if(last_angle != turn_angle){
        right_count -= abs(turn_angle) * 0.07;
    }
    last_angle = turn_angle;
    ...
    TurnMotor.write(mid_angle + turn_angle);
}
```

(b) 控制轉向：

在切換循跡線時，控制轉彎皆為單邊輪子固定、單邊輪子轉動的自轉。如下圖 code，我們先將煞車放下，再轉動角度。因為我們不希望車體轉向輪擺到正確角度過程時，車子會一邊前進一邊大角度轉彎，而是等煞車放置好、角度轉好，將風扇馬力加大(由於轉彎是靠風力分力，風力輸入需大一點。)放開煞車，此時開始自轉，直到 Encoder 計數達到目標為止，再直走。

在我們控制中，所有除了循跡以外要求指定角度的的轉彎，皆透過這種方式。舉例來說，在剛進入第一軌道前會有轉角 90 度，切換跑道前亦會轉角約 75 度。但由於切換跑道前，車頭的方向受限於循跡過程的不確定性，所以進入第二跑道時無法確定切入跑道的角度，所以該區域採取自轉直到偵測到中央線(中央紅外線有讀值)，並未使用 Encoder。詳細於 3.5.3 介紹。

```
BrakeMotor.write(brake_on); // brake on
delay(500);
cur_angle = mid_angle - 40; // turn angle to 90 deg
TurnMotor.write(cur_angle);
fan_motor_write(78); // speed up the fan
delay(800); // wait for speed up
BrakeMotor.write(brake_off); // start turning
right_count = 0; // zero the count
while(1){ // turn until target angles meet
    right_encoder();
    if(right_count >= 28){
        break;
    }
}
TurnMotor.write(mid_angle);
```

3.5.1.3 風扇控制

以下將介紹如何利用<Servo.h>輸出訊號至雙向電變及馬達，使其達到前進、控制速度及爬坡等之目的。關於風扇所使用驅動硬體設備及其原理，於下一個章節 3.5.2 詳細介紹。

首先，Arduino 中的<Servo.h>套件，可精準控制馬達轉速，利用指令 Servowrite()，可將訊號輸出雙向電變，直至馬達，括號內則是輸入相對訊號大小，範圍從 0 至 180。而因本組使用雙向電變(bidirectional ESC)，故 Servoewrite()的數值控制上以 90 為中心，小於 90 為反轉，大於 90 為正轉，數值距 90 越遠，訊號強度越強，其訊號強度(duty)公式為：

$$\text{duty} = \frac{|\text{Servowrite()數值} - 90|}{90} \qquad (\text{式 } 3.5.2-1)$$

```
void setup(){
    Serial.begin(9600);
    FanMotor.attach(fan_pin);
    Seiral.print("Input value between 0~180 (90 central value)")
    fan_motor_write(90);
    delay(1000);
}
void loop(){
    fan_motor_write(value);
    if(Serial.available()){
        value = Serial.parseInt();
        Serial.println(value);
    }
}
```

下面程式碼為測試風動車風扇需調整至多少 duty，才可驅動車體前進，控制上可從 Arduino 中的 monitor 中輸入 Servowrite()數值，使風扇運轉。

另外，為預防風扇轉速過高而造成危險，此設計一函式如下，並在其中規定轉速的 threshold，即 Servowrite()之數值上下界，使 duty 處合理範圍內，也確保相關安全。

```
void fan_motor_write(int value){
    Serial.print("fan give a speed ");
    int upperlimit = 105;
    int lowerlimit = 75;
    if((value < lowerlimit) || (value > upperlimit)){
        FanMotor.write(90);
        Serial.println("\nfan error. exiting...");
        myexit(0);
    }
    else{
        Serial.println(value);
        FanMotor.write(value);
    }
}
```

因此，藉上述原理，我們可以配合風動車之不同目標，調整 Servowrite()數值，使之達成目的，如風動車循跡需放慢速度(duty 變小)、轉彎(同時煞車及轉向)時驅動力增加(duty 變大)、風動車倒車(Servowrite()從小於 90 變為大於 90)。

另外，duty 也可用於推算馬達提供之轉速。於 3.5.1 中有提到，無刷馬達中的 kV 值代表每 1 伏特下，馬達理論可提供的轉速，本組所選用之無刷馬達為 2300 kV，所使用電池伏特數為 11.1V。而 duty 為訊號強度，代表電壓輸出之比例，故在本組的電池中 50%為電池輸出全部之電壓(5.5V)，100 %為電池輸出全部之電壓(11.1V)。綜合上述，我們可以利用輸入 Servowrite()數值大小推得理論之馬達轉速，其公式為：

$$\omega\,[\text{rpm}] = 11.1[\text{V}] \times 2300[\text{kV}] \times \frac{|\text{Servoewrite()數值} - 90|}{90} \quad (\text{式 } 3.5.2-2)$$

而根據期中測試之經驗，本組 duty 大小約為 10~15%，故馬達理論轉速約為 2553~3829 rpm。

也因如此，電池電壓之穩定在風扇控制上非常重要，若多次使用造成電壓降低，其會直接影響轉速。故本組在電池充電上，會將電池電壓控制在 11.1V 至 11.7V，若電壓充電過高則會造成轉速太快，不易控制。

3.5.2. 機電系統

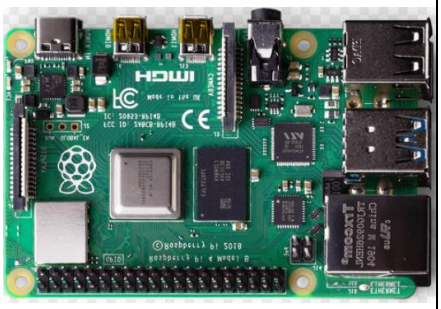
3.5.2.1. 元件選用

根據前面幾個章節描述的設計，系統電路設計中包含的元件為：

(1) 控制板：控制系統的計算核心。
(2) 伺服馬達：兩顆。一為控制煞車、一為控制轉向。
(3) 電變及無刷馬達：驅動並控制風扇高速轉動。
(4) 電池：兩枚。一為提供控制版電源、一為提供電變電源。
(5) 紅外線感測器：循跡模組。
(6) 編碼器：編碼模組，用於計算行徑軌跡。

以下將根據前面各章節描述的需求，介紹各個元件的規格比較及選用。

(1) 控制板選用：

	Arduino	Raspberry PI
	 圖 3.5.7 [15]	 圖 3.5.8 [16]
優點	(a) 網路資源多：Stackoverflow 或 Arduino 網站論壇平台皆有許多資源。 (b) 好上手：以 C 語言編寫，Arduino IDE 好操作，亦可使用 VScode 作為控制介面。 (c) 價格便宜 (d) IO 腳位多	(a) 搭載 Linux 核心作業系統，CPU 有排程能力，可平形化執行多個程序。 (b) CPU 較好，支援視覺辨識運算。
缺點	(a) 單一 CPU，不能控制程式不能平行化，所以 Sensor 讀值會有取樣頻率的問題要考量。	(a) 較難上手：需熟悉 Linux。 (b) 價格昂貴

綜合以上比較，我們最後選用 Arduino MEGA 板作為控制板，因其 Analog 腳位數量多，支援我們十一塊紅外線感測器的循跡模組，且 Arduino 使用 C 語言編寫，在控制及操作上較容易。

(2) 伺服馬達選用：

在進入第一條循跡線還有轉換跑道的階段，及循跡過程的轉彎控制，都需要有馬達驅動轉向機構，而伺服馬達有低轉速、扭力大、減速比大的特色，較符合我們的需求，因此我們整理出來了三種伺服馬達，如以下所示。

	SG90	MG90S	MG996R
	 圖 3.5.9 [17]	 圖 3.5.10 [18]	 圖 3.5.11 [19]
size(mm^3)	23*12.2*29	22.8*12.2*28.5	40.8*20*38
weight(g)	9	13.6	55
torque(kg/cm)	1.8	2	15
gear	塑膠齒輪	金屬銅齒	金屬齒輪

針對轉向的伺服馬達，我們其實不需要很大的扭矩，因此我們最後選擇尺寸最小、較為便宜的 SG90，較能實現我們車體輕量化的目標。至於煞車機構在期中實測上，因為車體輕、摩擦膠皮摩擦力夠，所以低扭力的 SG90 及可符合需求。故兩個功能都使用 SG90 作為驅動馬達。至於期末爬坡目標，原本預計進行爬坡實驗、確認煞車力道在滑坡面上是否足夠後，再決定維持期中所使用的 SG90 或者改為扭力較高的 MG996。

(3) 電變及無刷馬達：

風扇控制為調控風動車之整體前進速度以及其推力，其風扇轉速及產生之扭矩等皆會影響車體的前進速度。因此，在風扇的控制上，硬體設備之要求須以能產生足夠推力為首要目的，軟體編程的部分則是以 Arduino 為主，因其所開發之 library 已足夠良好控制相關指令如馬達轉速等。故以下討論以硬體零件為主。

在選擇馬達上，其種類有無刷馬達及有刷馬達二種，而在驅動風扇上選擇無刷馬達而非有刷馬達之原因如下：

(i) 低電壓、啟動快且易控制

(ii) 可靠度高、壽命長且高速化

(iii) 步產生電器之雜訊(意即步產生電波干擾)

(iv) 不產生火花，安全性高

(v) 換向時不易產生高溫之電弧及金屬屑

因此，在需要高速轉動下，選擇無刷馬達作為驅動風扇之來源較為適合。

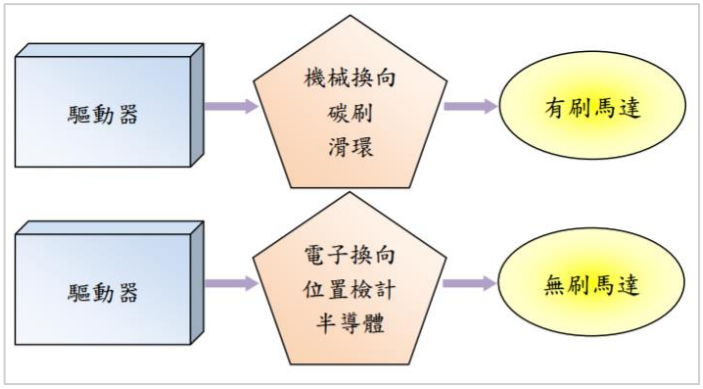


圖 3.5.12 有刷馬達及無刷馬達運作原理比較[20]

而在無刷馬達的選擇上，其最重要的參考指標為 kV 值。kV 值代表每 1 伏特下，馬達理論可提供的轉速(單位為 rpm)。我們此次選用 A2217 2300 kV 型號的無刷馬達，其中 2300 kV 代表當對馬達供給 1 伏特的電時，其轉速為 2300 rpm。

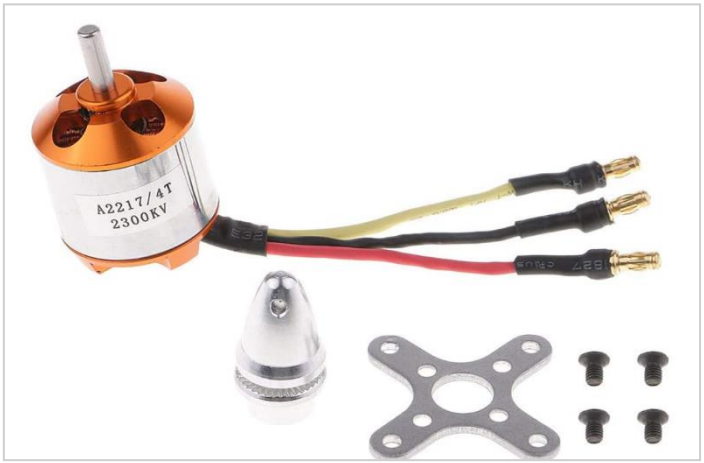


圖 3.5.13 A2217 2300kV 無刷馬達[21]

因無刷馬達需三相之交流電，此須以透過電子變速器產生。電變主要功能為將 Arduino 的訊號輸入於此後，並控制大電流輸出至無刷馬達。而我們預計使用之電變為 Skywalker 30A Bidirectional ESC，如圖 3.5.1-5 所示，除可以控電流外，還可將訊號反向輸出(bidirectional)，使無刷馬達達正反轉之效果，進而讓風動車可以達正向或反向前進。因此，藉電變之使用，在程式中只需透過 <Servo.h>函式庫即可直接控制無刷馬達的轉速，其程式原理會在 3.5.2 介紹。

圖 3.5.14 Skywalker 30A Bidirectional ESC[22]

(4) 電池：

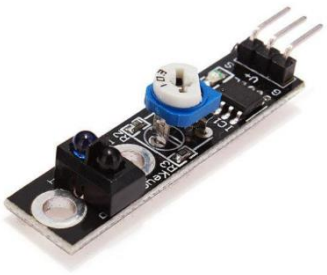
我們所使用的電池有兩種需求：一為提供控制板 3.7V 電壓，另一個為提供驅動高速旋轉的無刷馬達的 10V 左右的電壓，我們使用兩組電池來完成不同需求。

	18650 鋰電池	鋰聚電池
	 圖 3.5.15 [23]	 圖 3.5.16 [#24]
優點	可提供 3.7 伏特的電壓，是市面上相當常見的鋰電池，其優點為組合靈活，可以選用不同的電池盒來並聯、串聯，結合出符合需求的供電系統	可提供 11 伏特的電壓，其優點為蓄電能力強，也可經電便調整電壓後降指定電壓輸至無刷馬達中
缺點	缺點為降壓速度較快。	缺點為密度比一般的鋰電池大，且蓄電不足時會稍稍降低 1V 左右，影響風扇表現。

因此，考量無刷馬達需要大且穩定之電壓，我們選用 2250mAh 35C 11.1V 鋰聚電池作為驅動風扇無刷馬達之電力來源，原本計畫在期末測試前，再加上穩壓器使輸出電壓更穩。此外，由於車體主要耗電的風扇馬達為鋰聚電池供電，另一顆電池僅需提供 Arduino 基本 Sensor IO 和伺服馬達輸出電壓，所以 18650 續航力十分足夠。我們利用兩顆 18650 串連提供 7.4 伏特的電壓。

(5) 紅外線感測器：

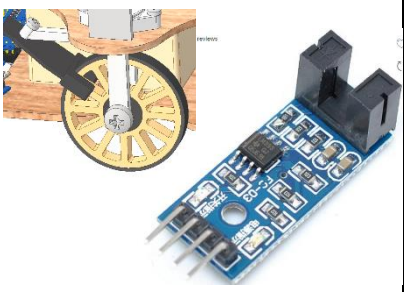
以下兩種為光華商場常見的兩種紅外線感測器型號。

	TCRT 5000	KY-033
	 圖 3.5.17 [25]	 圖 3.5.18 [26]
優點	可以讀取 Analog Input 及 Digital Input。Analog 讀值可以提供較多控制上的彈性。	價格較便宜。可以讀取 Digital Input。
缺點	元件之間閾值差異大，需校正。	只有 Digital Inpu。元件之間閾值差異大，需校正。

因為我們的紅外線模組一組就包含五顆紅外線感測器，所以紅外線感測器之間閾值若有太大差異就會嚴重影響循跡效果，所以選擇可以讀取 Analog Input 的 TCRT 5000，除了校正元件本身的電阻外，亦能從控制方面以 Analog 讀值觀察並利用程式控制不同紅外線感測器的讀值範圍，以精準校正不同紅外線感測器的差異。

(6) 編碼器：

編碼器可以自製，亦可使用電路元件，以下為兩種編碼器的比較。

	自製紅外線光學編碼器 （搭配 LM393 IRsensor）	AS5600 磁力絕對型編碼器
	 圖 3.5.19 [27]	 圖 3.5.20 [28]
簡介	以自製光柵搭配 LM393 紅外線收發器實作光學編碼器。（詳細參考 3.5.1.2 小節）	透過感測磁力變化來讀取旋轉位置的非接觸式系統， 利用霍爾效應可感測磁場改變方向，可量測徑相磁化軸上磁鐵的絕對角度，且沒有旋轉角度的限制，能消除任何同質外部雜散磁場的影響，並具有 12 位元的高解析度類比輸出，每圈有 4096 個位置表示。
優點	可客製化安裝。	精準度高。
缺點	精準度低。	安裝較不易。

由於風動車的車輪並非馬達驅動，而是受到風扇推力推動。若將編碼器安裝於被動旋轉的輪軸上，可能導致輪軸扭轉阻力增加使得風洞車不易前進，此外，為了客製化安裝，我們決定使用自製編碼器，詳細參考章節 3.5.1.2。

3.5.2.2. 電路圖配置

以下為機電系統的電路配置圖。

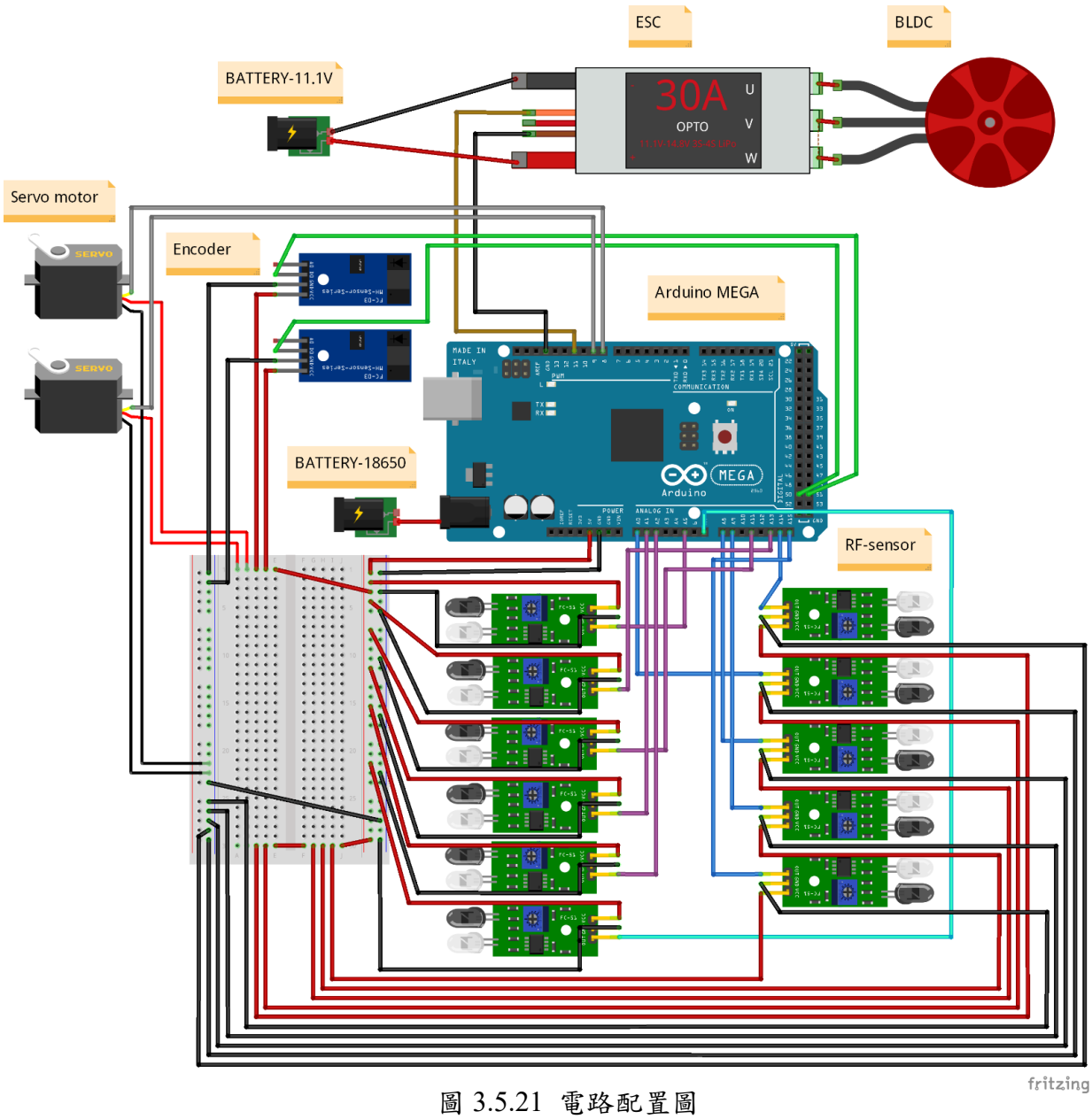


圖 3.5.21 電路配置圖

3.5.3. 整體控制規劃

根據本次比賽需求，我們將車子的行徑階段分為以下幾個 state

<state 1> 起始走到第一緩衝區的線上

<state 2> 第一條線循跡走到第二緩衝區

<state 3> 轉換跑到到第二緩衝區的線上

<state 4> 第二條線的循跡與倒車

<state 5> 循跡到第二停止區並停止

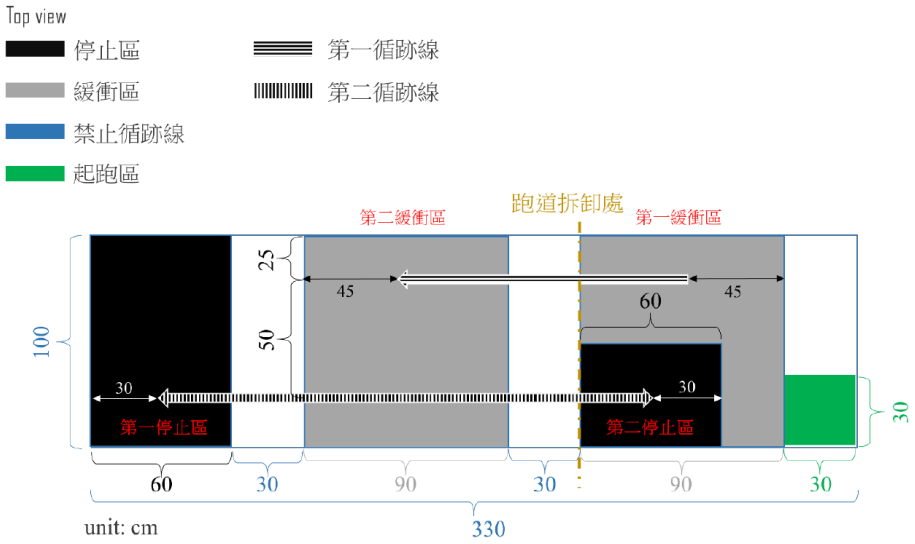


圖 3.5.22 場地示意圖

以下將細部介紹每一個 state 我們嘗試過的循跡、定位策略，並附上程式碼。

(1) <state 1> 起始走到第一緩衝區的線上

在這個階段，我們嘗試過兩種策略。第一種策略使用計秒並搭配紅外線感測器監測試否碰觸黑線，第二種方式使用 Encoder 回傳的距離資訊並搭配紅外線感測器監測是否碰觸黑線，以下將分別介紹。

(a) 策略一：

起始區時車頭朝前，開啟風扇後先朝前走一段後，將前輪向右轉，再慢慢回正，軌跡如圖所示。即

直走：先讓車子向前走 1 秒。

右轉：將前輪伺服轉向 22 度，每隔 4 秒減少 5 度，20 秒後車子完成回正。

找到循跡線：回正後 1 秒或紅外線感測器感測到第一條循跡線時煞車。

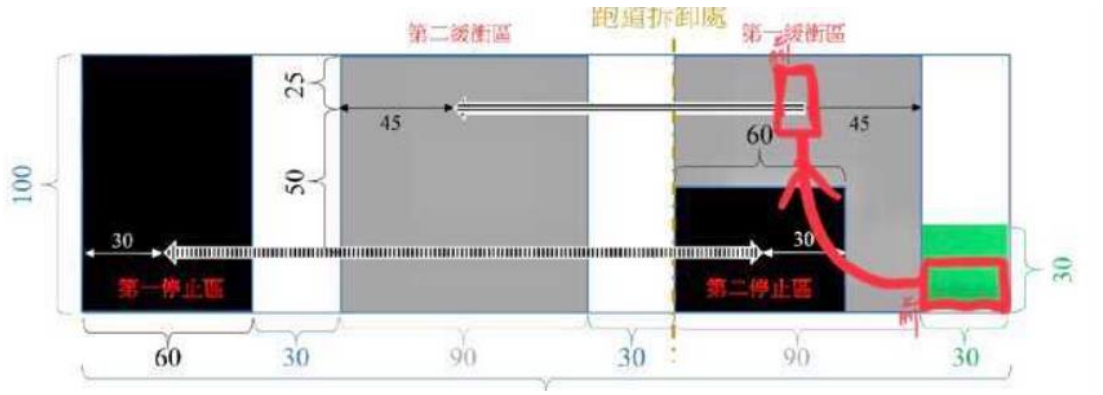


圖 3.5.23

程式碼：

```
for(cur_angle = mid_angle + 22; cur_angle > mid_angle; cur_angle += -5){
    TurnMotor.write(cur_angle);
    delay(400);
}
cur_angle = mid_angle;
TurnMotor.write(cur_angle);

// find the line and brake.
int b_init_time = millis();
while(1){
    int bit = (analogRead(IR[FM] > IRthres[FM]));
    if(bit || (millis() - b_init_time) > 1000)
        BrakeMotor.write(brake_on);
}
```

(b) 策略二：

起始區時車頭朝右，開啟風扇後先向前走一段後，左轉 90 度，再向前直走找到循跡線，軌跡如圖所示。即

直走：打開電源後車子向前直走，當左輪的 encoder 計數達到 40 時，車體煞車

左轉：前輪伺服轉 40 度，完成前輪伺服轉向後，放開煞車，讓車體開始轉彎，轉彎過程當右輪的 encoder 計數達到 42 時，完成轉彎

直走：前輪伺服轉正，車子繼續直走，直到右輪 encoder 計數達到 30。

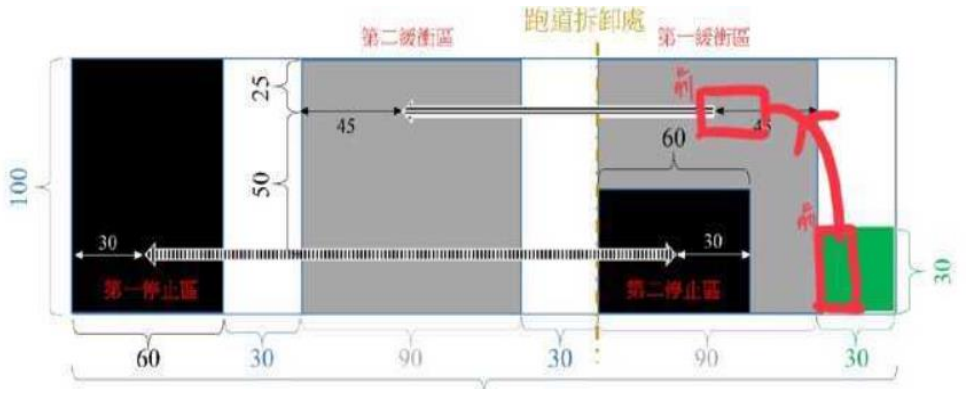


圖 3.5.24

程式碼：

```
fan_motor_write(80);
left_count = 0;

Serial.println("** Go straight");
while(1){
    left_encoder();
    if(left_count >= 48){
        break;
    }
}

BrakeMotor.write(brake_on); delay(500);
cur_angle = mid_angle - 40;
TurnMotor.write(cur_angle); delay(800);

Serial.println("** Start turning");
BrakeMotor.write(brake_off);

right_count = 0;
while(1){
    right_encoder();
    if(right_count >= 42){
        break;
    }
}

Serial.println("** Start straight");
TurnMotor.write(mid_angle);
right_count = 0;
while(1)
{
    right_encoder();
    if(right_count >= 30){
        break;
    }
}
```

經過實作後，我們發現用計時(策略一)的方式極容易受到電壓影響，當電壓不同時，同樣的秒數內的距離也會不同，因此很難找到一個適合的參數。因此我們最後決定使用策略二，用 Encoder 計數，儘管電壓不同、車速不同，我們依然能讓車子在固定的位置執行特定的動作。

(2) <state 2>第一條線循跡走到第二緩衝區

一開始我們循跡使用的策略是三路循跡，但發現 IR sensor 間的間距太大會導致循跡時車體 damping 太嚴重，因此我們將循跡改為五路循跡，減少 IR sensor 之間的距離。

(a) state 2 策略：

從車前左到右共五個 IR sensor，IR sensor 皆為 analog read，分別設定每個 IR sensor 的 threshold，當每個 IR sensor 超過他的 threshold 時，設值為 1；反之，設值為 0。循跡時前輪伺服轉角和 IR sensor 的讀值關係為：(詳細參見 3.5.1.2 章節)

```
turn_angle = (-23)*IRvalue[FLL] + (-15)*IRvalue[FL] + (0)*IRvalue[FM] +
(15)*IRvalue[FR] + (23)*IRvalue[FRR];
```

在這個階段我們比較常遇到兩個問題。第一個是循跡過程可能會不小心循到橫線，車子就會沿著橫線走掉，針對這個問題，我們的解決方法是當車前五個 IR sensor 有兩個以上讀到黑線，就判定該線為橫線，這個情況下不會調整前輪轉角。

第二是如果循跡時車子的 damping 比較大，會讓 encoder 的計數變多，導致之後判斷循跡結束的時機會失準。我們的解決方法是如果連續兩次判斷的轉角相同(比如上一次轉 23 度這次也轉 23 度，代表車子持續朝同一個方向偏移，即循跡的 damping 較大)，這個情況下我們就會減少目前記錄到的 encoder 的讀值，即 right_count -= abs(turn_angle) * 0.07

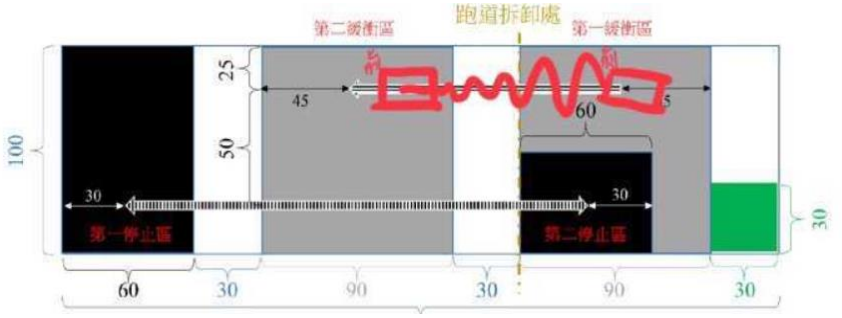


圖 3.5.25

程式碼：

```
fan_motor_write(86);
Serial.println("** Following the 1st line");
right_count = 0;
int turn_angle = 0;
int last_angle = 0;
int IRvalue[11] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

while(1){
    right_encoder();

    if(right_count >= 6.8){
        break;
    }
    IRvalue[FLL] = (analogRead(IR[FLL]) > IRthres[FLL]);
    IRvalue[FL] = (analogRead(IR[FL]) > IRthres[FL]);
    IRvalue[FM] = (analogRead(IR[FM]) > IRthres[FM]);
    IRvalue[FR] = (analogRead(IR[FR]) > IRthres[FR]);
    IRvalue[FRR] = (analogRead(IR[FRR]) > IRthres[FRR]);

    Serial.print("<IR_line> ");
    Serial.print(IRvalue[FLL]);
    Serial.print(" ");
    Serial.print(IRvalue[FL]);
    Serial.print(" ");
    Serial.print(IRvalue[FM]);
    Serial.print(" ");
    Serial.print(IRvalue[FR]);
    Serial.print(" ");
    Serial.print(IRvalue[FRR]);

    Serial.print(" <angle> ");

    if((IRvalue[FLL] + IRvalue[FL] + IRvalue[FM] + IRvalue[FR] + IRvalue[FRR]) > 1){
        turn_angle = 0;
        Serial.print(" bad_line");
    }
    else{
        turn_angle = (-23)*IRvalue[FLL] + (-15)*IRvalue[FL] + (0)*IRvalue[FM] +\
        (15)*IRvalue[FR] + (23)*IRvalue[FRR];
        if(last_angle != turn_angle){
            right_count -= abs(turn_angle) * 0.07;
        }
        last_angle = turn_angle;
    }
    TurnMotor.write(mid_angle + turn_angle);
    Serial.println(turn_angle);
}
TurnMotor.write(mid_angle);
```

(3) <state 3> 轉換跑到到第二緩衝區的線上

這個階段我們要從第一循跡線轉換跑道至第二循跡線，我們也嘗試了兩個策略，第一個策略是用 IR sensor 和計時，第二個策略是用 encoder 和 IR sensor，兩個策略的理想軌跡圖皆如圖所示。

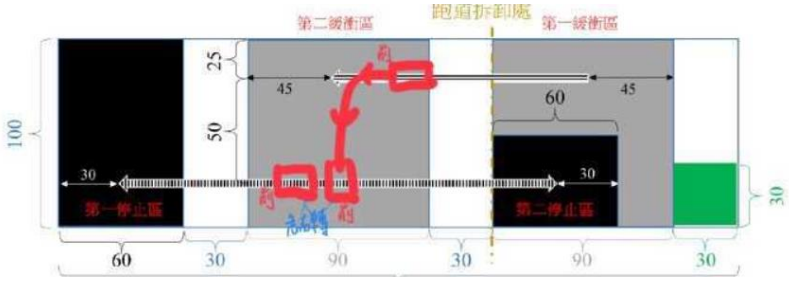


圖 3.5.26

(a) 策略一

- 轉向：將前輪伺服轉向 18 度，每隔 4 秒減少 6 度，12 秒後車子完成回正
- 找到循跡線 ： 回正後 1 秒或紅外線感測器感測到第二條循跡線時煞車
- 車身轉右 ： 煞車後將前輪伺服轉 40 度再放開煞車，並將前輪伺服轉回正中間

程式碼：

```
for(cur_angle = mid_angle + 18; cur_angle > mid_angle; cur_angle += -6){
    TurnMotor.write(cur_angle);
    delay(400);
}
cur_angle = mid_angle;
TurnMotor.write(cur_angle);

// find the 2nd-line and brake.
int b_init_time = millis();
while(1){
    int bit = (analogRead(IR[FM] > IRthres[FM]));
    if(bit || (millis() - b_init_time) > 1000)
        BrakeMotor.write(brake_on);
}
delay(2000);
// Turn until the car is straight to line
TurnMotor.write(mid_angle + 40);
delay(50);
BrakeMotor.write(brake_off);
delay(1800);
TurnMotor.write(mid_angle);
```

(b) 策略二：由於我們發現若等到車子中間的紅外線感測器偵測到黑線才煞車，煞車本身不是及時發生的，會有馬達控制煞車裝制的延遲，還有動摩擦轉為靜摩擦的延遲，但我們希望車子看到黑線就會即時煞車，為了解決該延遲，我們在車體最左側多安裝了一個紅外線感測器，目的是為了提前感測到黑線、提前開始煞車。如此一來變能恰巧在車頭壓在線上時就煞好車。

程式碼：

```
BrakeMotor.write(brake_on); delay(500);
cur_angle = mid_angle - 40;
TurnMotor.write(cur_angle);
fan_motor_write(78); delay(800);
Serial.println("** Start turning");
BrakeMotor.write(brake_off);
right_count = 0;
while(1){
    right_encoder();
    if(right_count >= 28)
        break;
}
fan_motor_write(81);
Serial.println("** Start straight to find 2nd line");
TurnMotor.write(mid_angle);
right_count = 0;
int flag = 0;
while(1){
    right_encoder();
    if( right_count >= 5 && flag == 0){
        fan_motor_write(86);
        flag = 1;
    }
    if(analogRead(IR[ADD]) > IRthres[ADD] && right_count > 3){
        Serial.println("stop");
        break;
    }
}
BrakeMotor.write(brake_on); delay(500);
cur_angle = mid_angle + 40;
TurnMotor.write(cur_angle);
fan_motor_write(81); delay(800);
Serial.println("** Start turning to be straight");
BrakeMotor.write(brake_off);
right_count = 0;
while(1){
    right_encoder();
    if(analogRead(IR[FRR]) > IRthres[FRR])
        break;
}
BrakeMotor.write(brake_on);
fan_motor_write(84); delay(500);
BrakeMotor.write(brake_off);
```

State3 最後決定使用策略二。原因和 state1 描述的狀況一樣，如果使用計時的方法太容易受到電壓變化的影響，測試中大約測試第三次就會因為電壓降下來而沒辦法用原本的參數達到目標，因此我們最後決定使用策略二。

(4) <state 4> 第二條線的循跡到第一停止區

這個階段的循跡方式和循跡第一條循跡線的策略相同，但因為循跡範圍較小，所以轉向角度可以設定較小。前輪伺服轉角和 IR sensor 讀值間的關係為：

```
1    turn_angle = (-18)*IRvalue[FLL] + (-12)*IRvalue[FL] + (0)*IRvalue[FM] +\
2                 (18)*IRvalue[FR] + (12)*IRvalue[FRR];
```

理想軌跡圖如圖所示。

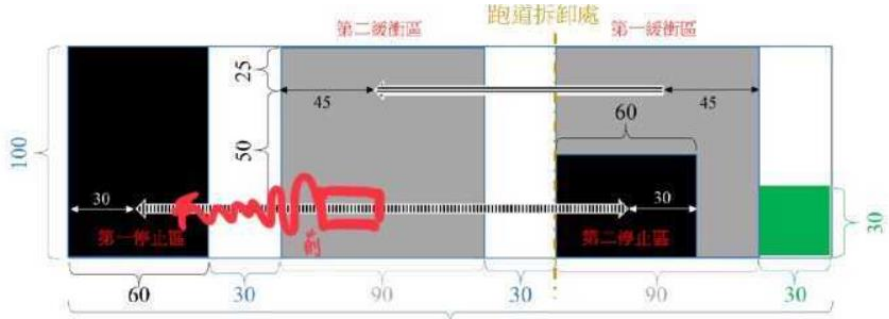


圖 3.5.27

(5) <state 5> 第一停止區倒車

在第一停止區停止 5 秒後，車子需要倒車到第一停止區，在這個階段我們嘗試了兩種策略。策略一是假設循跡到第一停止區時，車頭已穩定朝前，因此倒車時只要往回退即可直直地回到第二停止區(如圖所示)。策略二是比較保險的作法，也就是在倒車時循著第二循跡線回到停止區。

(a) 策略一：

將風扇反轉，讓車直接倒退，當右輪 encoder 計數達 200 時，車停止於第二停止區，理想軌跡圖如圖所示，但測試後我們發現，因為車在第二循跡線的循跡距離不夠長，車子到第二停止區時還未收斂(持續在 damping)，導致在第二停止區停止時車身不是直的，這時倒車將會偏離理想軌跡，如圖所示。

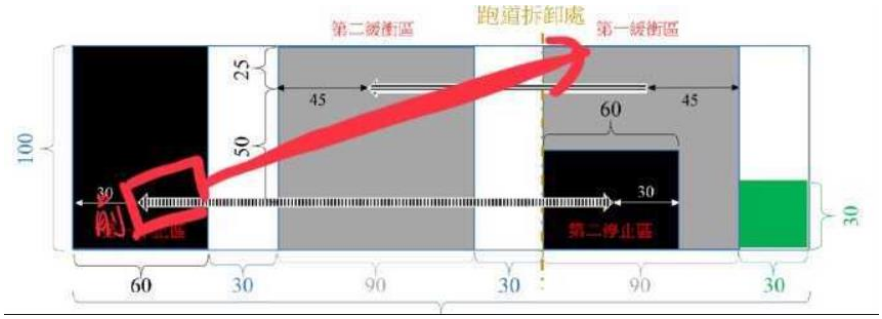


圖 3.5.28

(b) 策略二：車體後排有 5 個 IR sensor，使用後排的 IR sensor 進行循跡

前輪伺服和後排 IR sensor 讀數間的關係和先前相同

判斷車體在循跡倒車時經過三條橫線且右輪 encoder 計數達 200 後停止，理想

軌跡圖如圖所示。

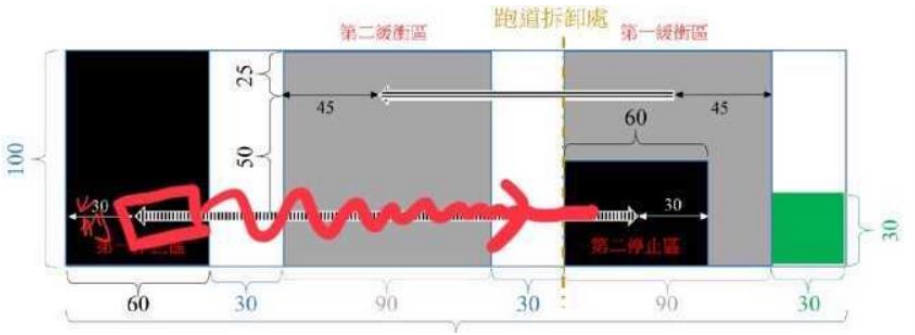


圖 3.5-29

State 5 決定使用策略二。經過測試，使用策略一時，沒辦法確保車體在進入第一停止區時方向是精準朝前的，因此直接倒車無法讓車回到第二停止區。所以我們最後選擇了策略二。這部分的程式碼和 state2 大致雷同，在此不另外附註。

完整控制程式碼，可參見第八章附件 8.2。

4. 設計分析與驗證

4.1. 風扇分析

於繪製好三代風扇之工程圖後，以下將先探討風扇個別於不同轉速模擬下呈現之流場分布、質量流量等物理參數表現，比較其差異，之後介紹風扇之相關實驗設計，以及其預期結果。

4.1.1. 流場模擬

討論風扇之模擬分析，我們使用 ANSYS 之 Fluid Flow(fluent)套件進行流場分析，將已繪製好之風扇檔案個別匯入，並輸入不同轉速，分析在不同轉速條件下其所造成之流場分布、理想最大速度及推力等參數，藉此可評估此何種風扇設計最合乎需求。ANSYS 之環境模擬假設如下：

(1) 假設流體在室溫空氣下進行模擬
(2) 忽略扇葉材質之相關性質

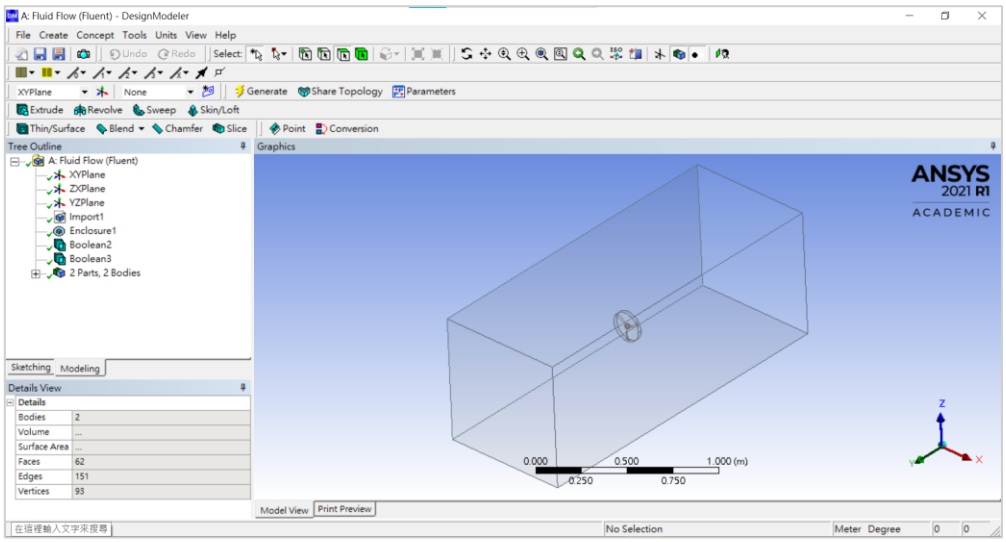
如此，以下流場分析主要分為四階段，將分步驟進行設定說明。

1. 幾何設定(Geometry) :

將風扇檔案轉為.step 檔後匯入軟體，開啟 Geometry 設定介面，先繪製一長方體 enclosure 作為流場邊界，截面設定為 0.6m，流場長度設定為 2m，上述為風扇所造成流場之影響範圍。之後將風扇置於前 1m 處，意即風扇進風口與出風口距離比為1:1。之後再以一圓盤將風扇包住，此目的為避免風扇旋轉時，其網格變形使數值難以收斂，故以一圓盤代替。其直徑只需比風扇大一些

即可，轉盤之大小越接近原始風扇大小模擬越為精準，故設定為風扇半徑加上0.5cm，使之能完整包覆風扇。

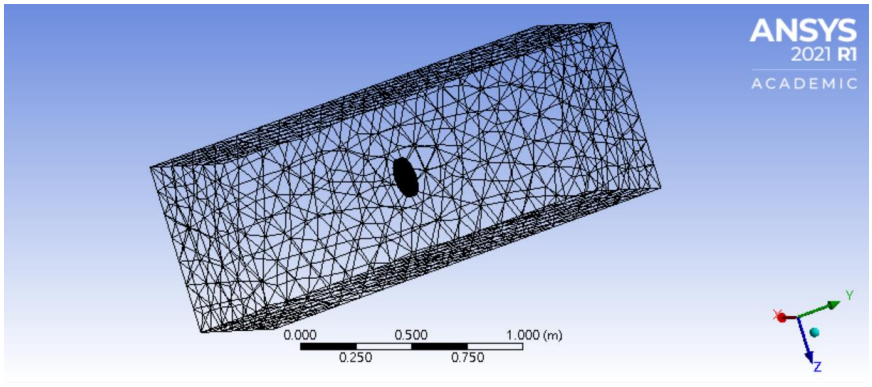
最後將此三樣物體行兩次 Boolean，第一次為外層為外層 enclosure 扣除圓盤，並在扣除時保留圓盤(勾選 preserve body)，第二個為旋轉圓盤扣除內部風扇，但此不保留風扇物件，因是以圓盤模擬風扇旋轉，分別為 enclosure 和 rotating body，以方便後續的設定。其介面如圖 4.1-1 所示。



(圖 4.1-1) Geometry 設定介面

2. 網格設定 (Mesh)

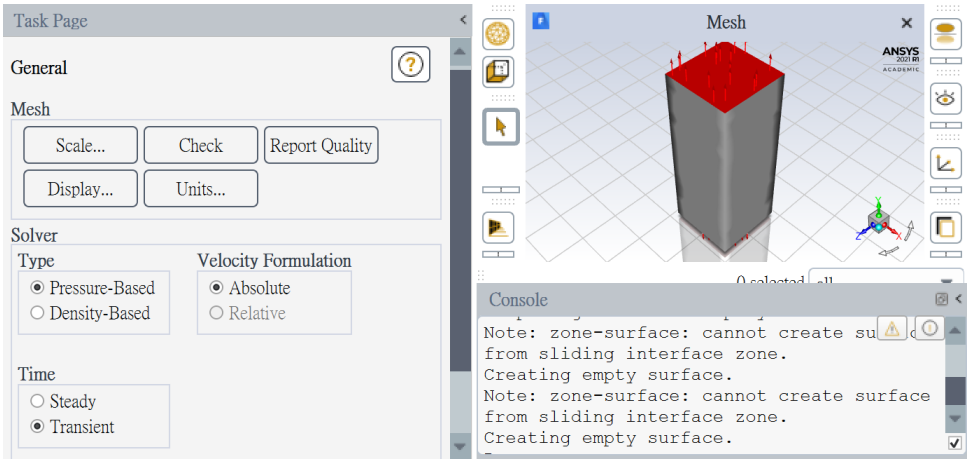
ANSYS 為行數值運算，須將目標物分成數塊小面積以行數值積分，稱為網格設定(Mesh)。先將 Physics Preference 設定為 CFD 後，再來設定網格大小。網格解析度會影響數值積分之精確度及時間複雜度，然而此風扇幾何形狀較為單純，故以預設值(最大網格約為 200mm，最小網格約為 1.0mm，網格數約為十萬)作為設定，如圖 4.1-2 所示。最後利用 name selection 將 enclosure 之入風口及出風口牆面設定名稱，分別為 outlet1 及 outlet2。



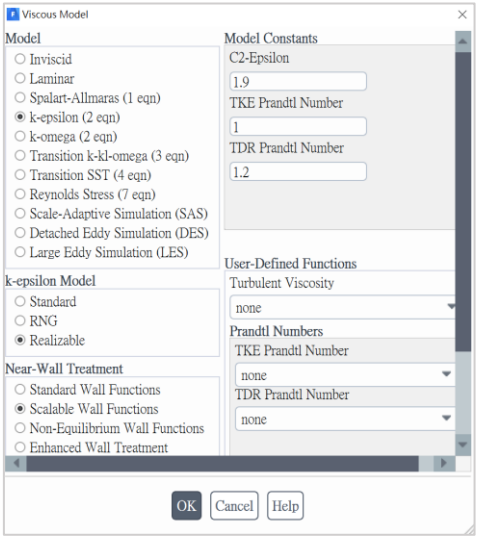
(圖 4.1-2) 網格設定

3. 初始設定 (set up)

此部分為設定計算之相關條件及環境參數等，故步驟較為繁瑣。先於 General 中調整為 Transient 開始(圖 4.1-3)。Models 選擇 Viscous Model 中的 k－ε 標準方程式、並選擇 realizable 及 scalable wall functions (圖 4.1-4)，其中 k 為紊流動能方程，ε為紊流擴散率方程，此數值方法用於模擬紊流流動之方程組，透過兩偏微分方程組以建構紊流之基本型態，並可以較精確的模擬扇葉的風場，其方程如圖 4.1-5 所示。之後在 cell zone conditions 中設定轉軸位置、轉軸方向 (轉向以右手法則判斷)與轉速(rpm) (圖 4.1-6)，須注意若風扇位置並非在絕對座標的正中央，則此處的旋轉位置座標就需要修正。另外，在 boundary condition 中，假設環境為無風狀態，故設定邊界條件之風速為 0。



(圖 4.1-3) Transient Setting



(圖 4.1-4) model 設定

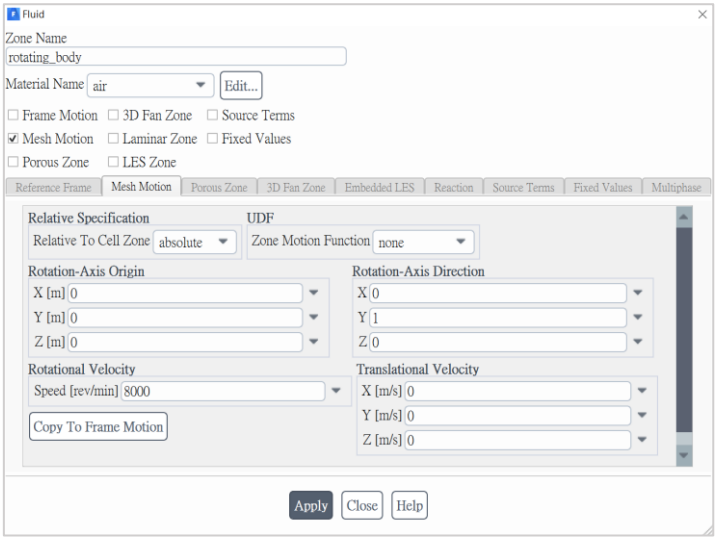
$$\frac{\partial}{\partial t}(\rho k)+\frac{\partial}{\partial x_j}(\rho k u_j)=\frac{\partial}{\partial x_j}\left[\left(\mu+\frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right]+P_k+P_b-\rho\epsilon-Y_M+S_k$$

$$\frac{\partial}{\partial t}(\rho\epsilon)+\frac{\partial}{\partial x_j}(\rho\epsilon u_j)=\frac{\partial}{\partial x_j}\left[\left(\mu+\frac{\mu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial x_j}\right]+\rho C_1 S\epsilon-\rho C_2\frac{\epsilon^2}{k+\sqrt{\nu\epsilon}}+C_{1\epsilon}\frac{\epsilon}{k}C_{3\epsilon}P_b+S_\epsilon$$

Where

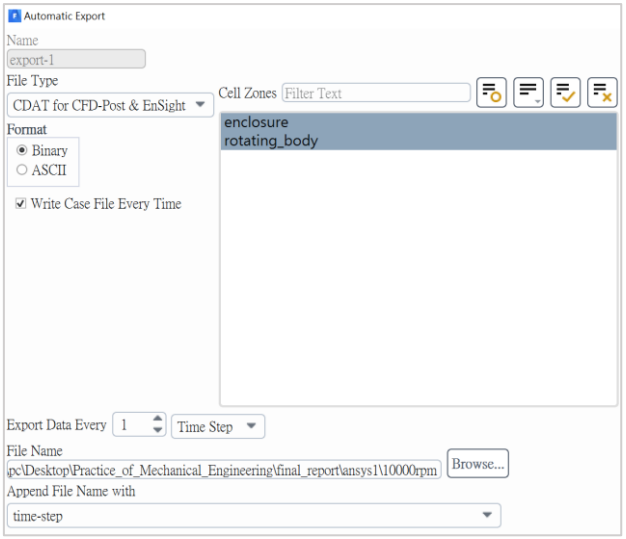
$$C_1=\max\left[0.43,\frac{\eta}{\eta+5}\right],\quad \eta=S\frac{k}{\epsilon},\quad S=\sqrt{2S_{ij}S_{ij}}$$

(圖 4.1-5) k－ε 標準方程[29]

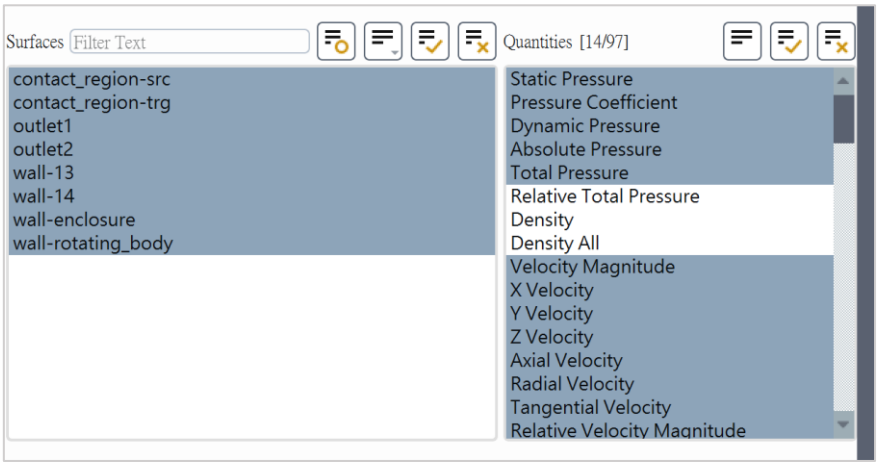


(圖 4.1-6) 轉速及轉軸設定

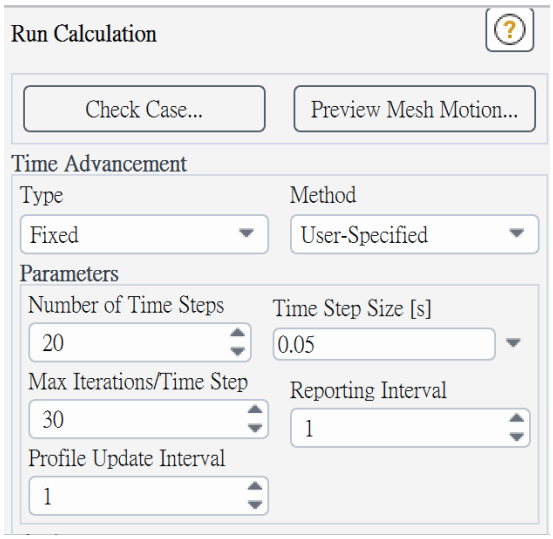
之後選取 Hybrid Initialize，設定分析結果之輸出位置，且將檔案類型更改為 CFD-Post Compatible (圖 4.1-7)，並選取欲得之物理量，如速度、壓力等(圖 4.1-8)。最後在 Run calculation 中，將模擬之時間單位設定為 0.05s，time step 共 30 次，Number of time steps 設為 20 次，其意義為數值計算的一步代表 0.05 秒，並在每一步中行 30 次迭代，最後總共走 20 步，數值結果為$0.05 \times 20 = 1$ 秒之結果，設定如圖 4.1-9 所示。其單位設定長短及迭代次數皆會影響最終數值之精確度及花費時間長短。[30][31]



(圖 4.1-7) 輸出檔案類型及路徑設定



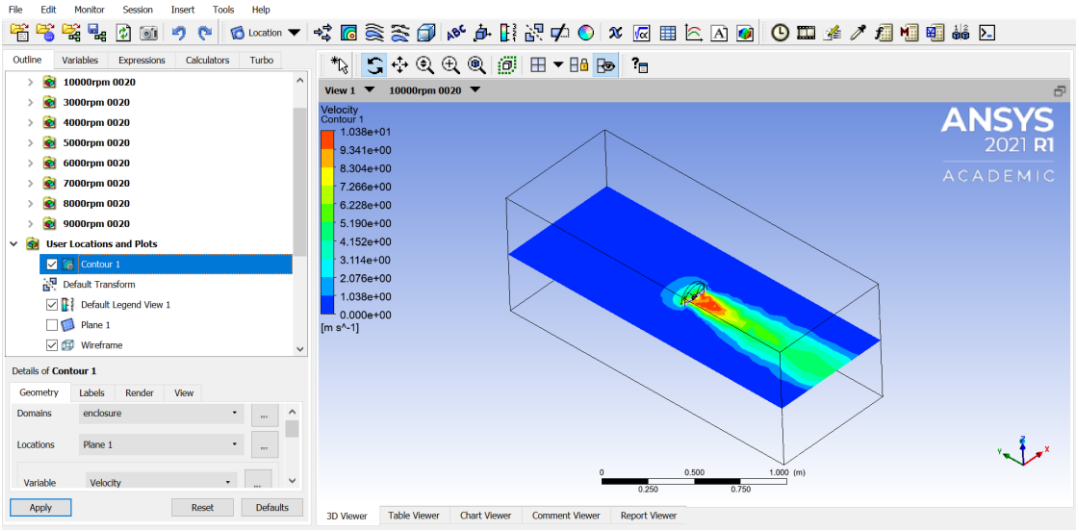
(圖 4.1-8) 參數輸出設定



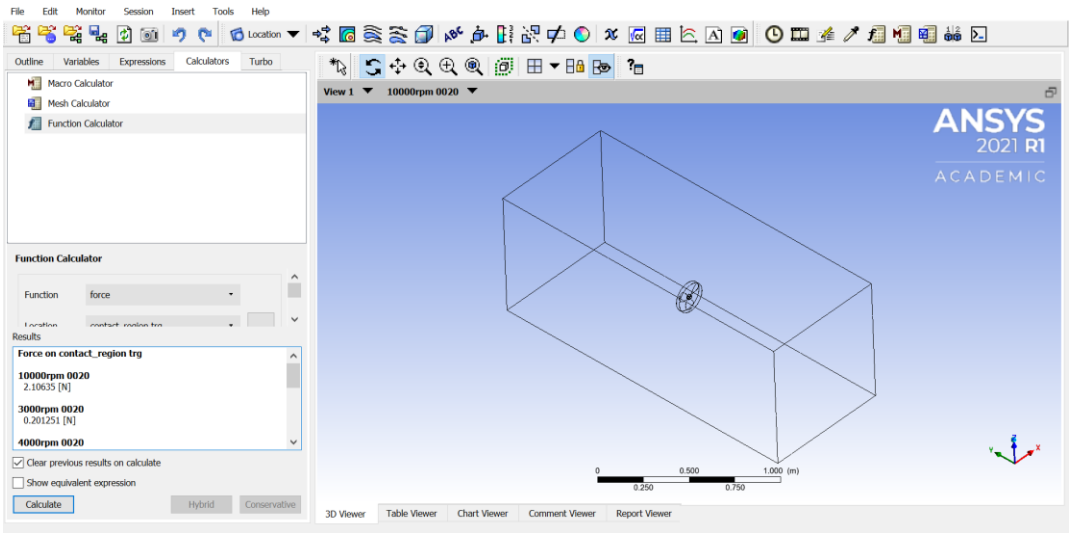
(圖 4.1-9) 時間迭代設定

4. Results

待數值運算後，即可利用所有資料檔(.cdat 檔)匯入後進行繪圖和相關參數計算。建立一平面(plane)後，選取工具列上 contour 功能，並在其設定中加入欲顯示之數值(如速度、壓力等)與所顯示之區域，如剛才所建立之平面等(圖 4.1.-10)，開啟可見性後即可顯示結果，也可以顯示 streamline、render 等不同流場表示法，可在設定中加入。另外，在 Calculator 頁籤中開啟 Function Calculator，設定欲計算之參數(如推力、mass flow 等)，即可得到相關結果 (圖 4.1-11)。



(圖 4.1-10) 繪圖設定



(圖 4.1-11) 參數計算設定

因此，藉上述步驟設定，我們可以在 ANSYS 上得到風扇模擬結果及相關參數之計算數值。以下將分析將利用 3.1.5 節(風扇設計)中提及的三種版本風

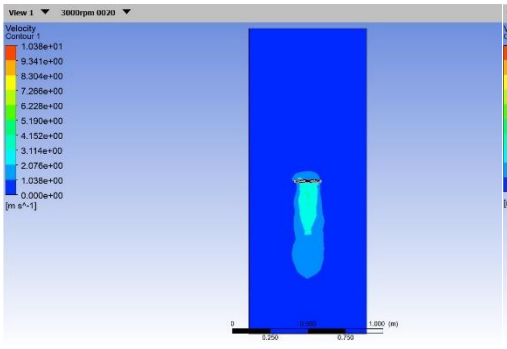
扇，於每次模擬時分別輸入不同轉速，由此比較不同轉速下的風扇推力、mass flow、最大速度及流場表現，並討論不同版本風扇之差異。

而在轉速設定上，因本組所使用之無刷馬達為 2250 kV，其中 kV 之參數意義為 1V 電壓下所能提供之轉速(rpm)，所選用電池電壓為 11.1 V。於期中測驗的經驗中，duty 設定約為 10 ~ 15%，其換算理論轉速約為 2498 rpm ~ 3746 rpm；於準備期末測驗的經驗中，duty 設定約為 30 ~ 40%，其換算理論轉速約為 7492 rpm ~ 9990 rpm。故在每次 ANSYS 模擬的轉速設定中，數值從 3000 rpm 開始設定，以 1000 rpm 為間隔直至 10000 rpm，共 8 種轉速，在三種版本風扇之情形下，共計模擬執行 24 次，並輸出 24 個資料檔 (.cdat 檔)。因此，藉計算及繪圖結果，以下將先討論各版本風扇在不同轉速下之速度分布及流場表現 (streamline)，而後比較各版本所產生的推力 (force)、質量流量 (mass flow) 及最大速度 (max. velocity)之差異，藉此選出最佳版本。

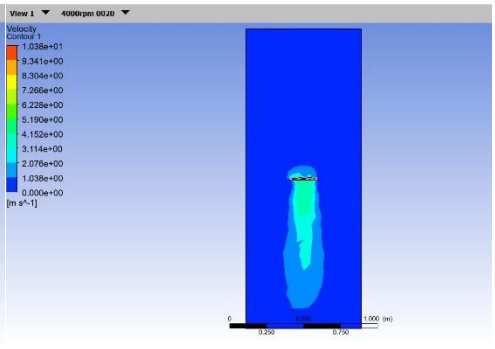
1. 第一版風扇

圖 4.1-12 至圖 4.1-19 為第一版風扇於模擬中在 3000 rpm 至 10000 rpm 之速度分布，其速度範圍從 0 至 10.38 m/s，可以發現當轉速越快時，風扇出風口速度增加(圖中顏色變深)，速度分布也隨轉速增加而逐漸散開。

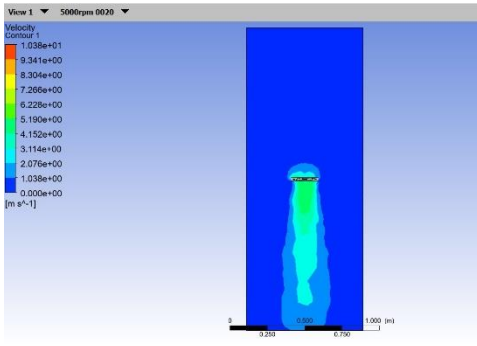
另外，圖 4.1-18 至圖 4.1-23 為 3000 rpm 至 10000 rpm 之流場表現 (streamline)，可以發現不同轉速分布差異不大，皆為從風扇入風口藉旋轉抽氣後，匯聚在一起後於出風口噴出(流線密度高)，但在出風口後端出有兩個類似 circulation 之現象產生(兩環形 streamline)，表此處流場與他處相比較為混亂。



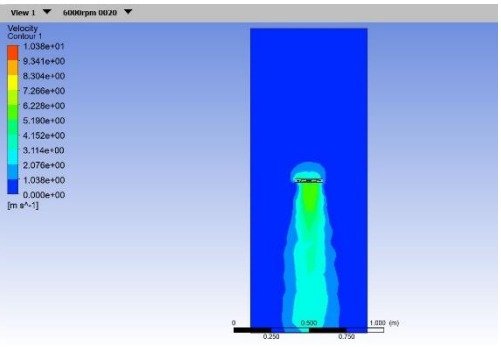
(圖 4.1-12) velocity @3000 rpm



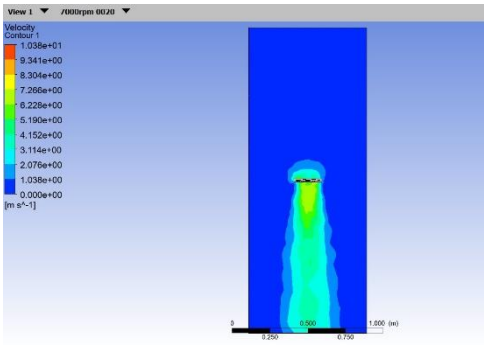
(圖 4.1-13) velocity @4000 rpm



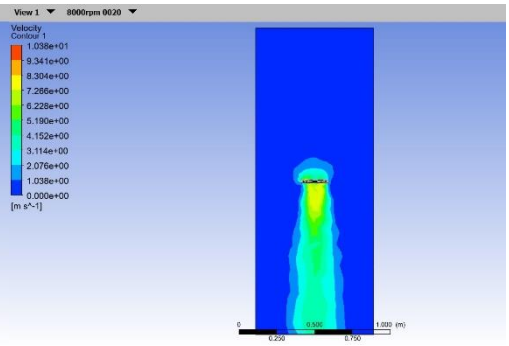
(圖 4.1-14) velocity @ 5000 rpm



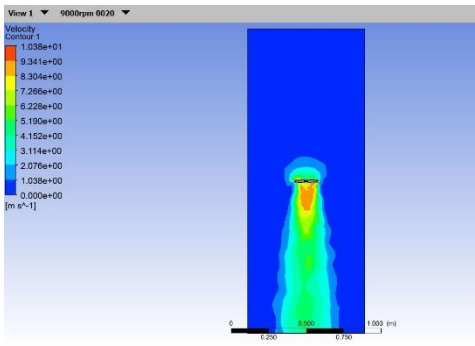
(圖 4.1-15) velocity @6000 rpm



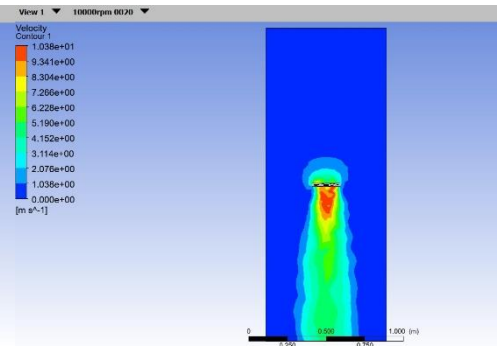
(圖 4.1-16) velocity @7000 rpm



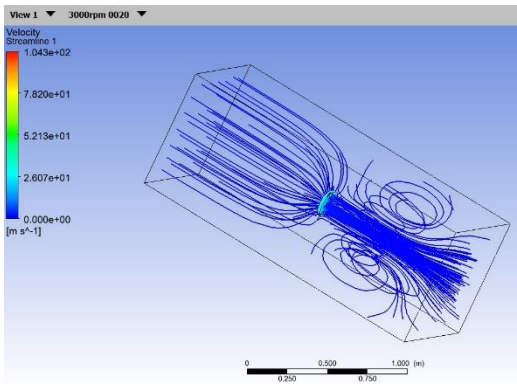
(圖 4.1-17) velocity @8000 rpm



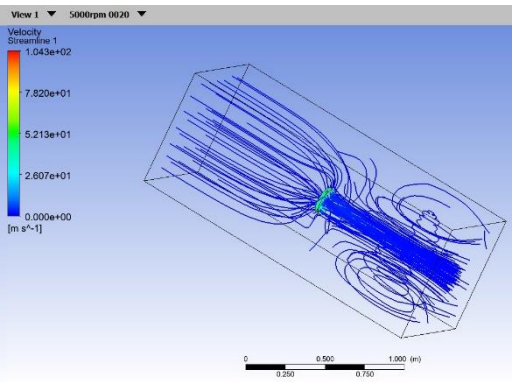
(圖 4.1-18) velocity @9000 rpm



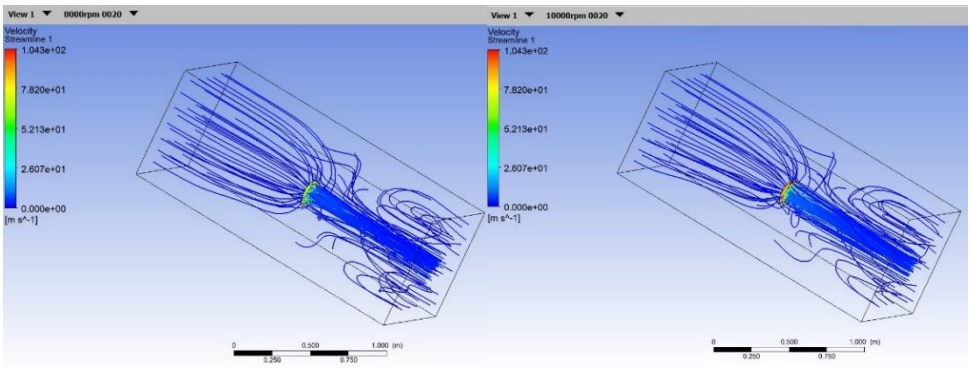
(圖 4.1-19) velocity @10000 rpm



(圖 4.1-20) streamline @3000 rpm



(圖 4.1-21) streamline @5000 rpm

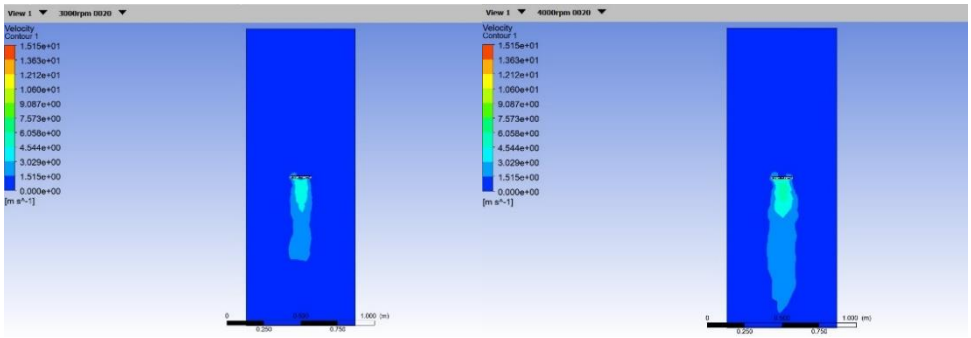


(圖 4.1-22) streamline @8000 rpm　　(圖 4.1-23) streamline @10000 rpm

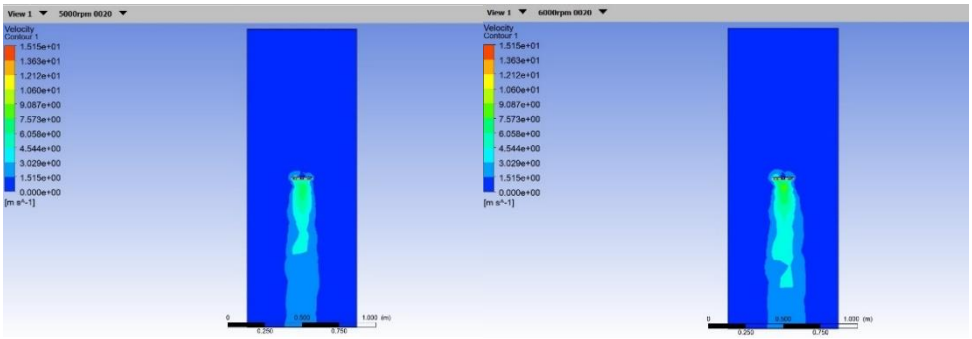
2. 第二版風扇

圖 4.1-24 至圖 4.1-31 為第二版風扇於模擬中在 3000 rpm 至 10000 rpm 之速度分布，其速度範圍從 0 至 15.15 m/s，可以發現當轉速越快時，風扇出風口速度增加(圖中顏色變深)，但速度分布較第一版風扇集中。

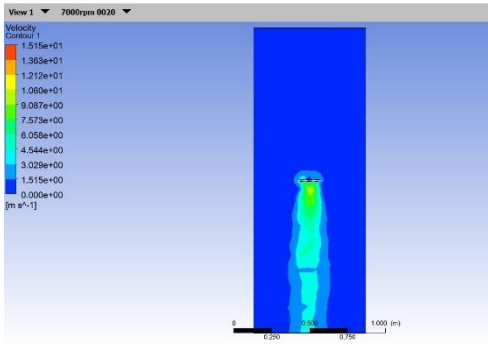
另外，圖 4.1-32 至圖 4.1-35 為 3000 rpm 至 10000 rpm 之流場表現(streamline)，可以發現不同轉速分布差異不大，且結果與第一版風扇類似，皆為從風扇旋轉抽氣後，匯聚後於出風口噴出，且在出風口後端出也有兩環形 streamline 之現象，但密度第一版風扇較低。最後，在高轉速時(如 10000rpm)，流線於出風口處有扭轉之現象，表風較為集中，不易發散，與第一版風扇的直流線相比，可想成扭轉現象使推力可有效的產生 torque，進而推動整個車體。



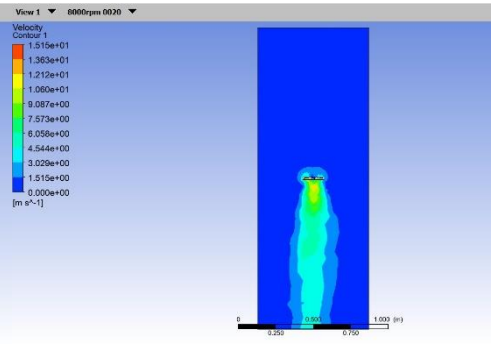
(圖 4.1-24) velocity @3000 rpm　　(圖 4.1-25) velocity @4000 rpm



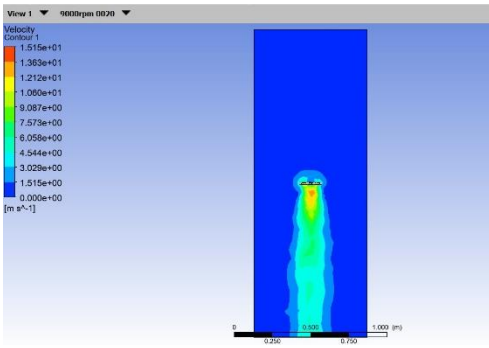
(圖 4.1-26) velocity @ 5000 rpm　　(圖 4.1-27) velocity @6000 rpm



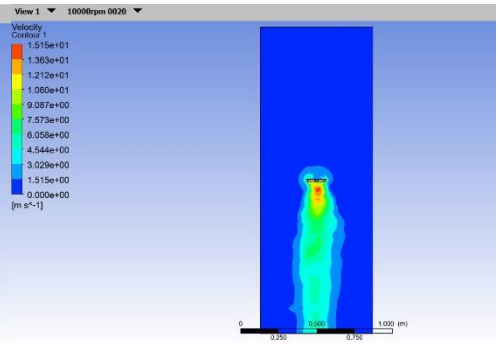
(圖 4.1-28) velocity @7000 rpm



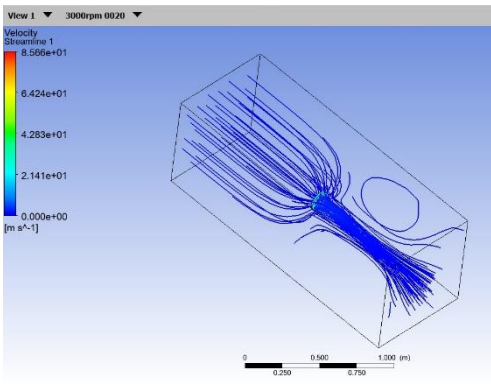
(圖 4.1-29) velocity @8000 rpm



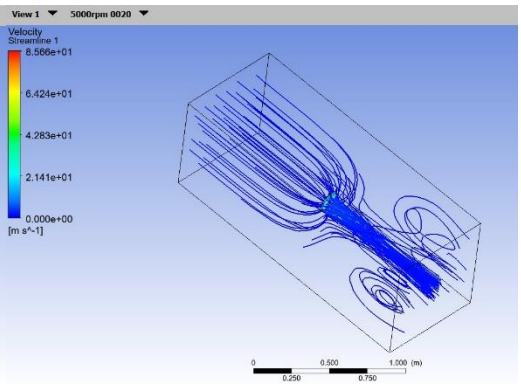
(圖 4.1-30) velocity @9000 rpm



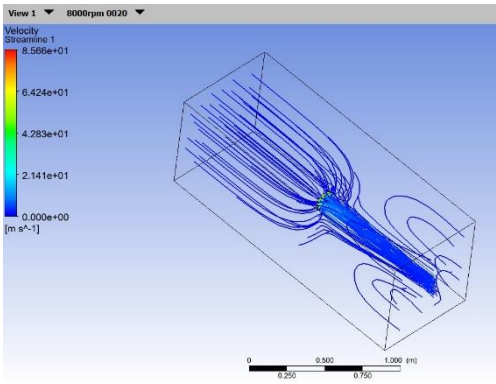
(圖 4.1-31) velocity @10000 rpm



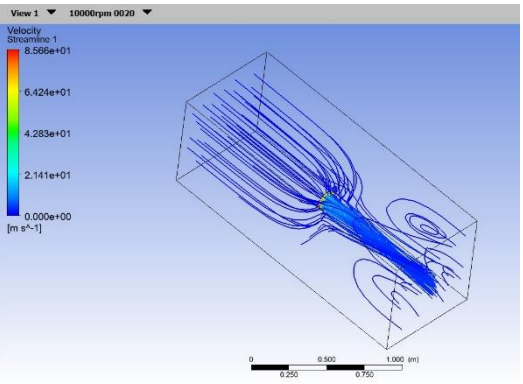
(圖 4.1-32) streamline @3000 rpm



(圖 4.1-33) streamline @5000 rpm



(圖 4.1-34) streamline @8000 rpm

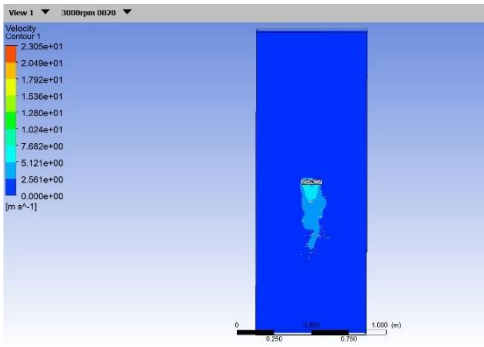


(圖 4.1-35) streamline @10000 rpm

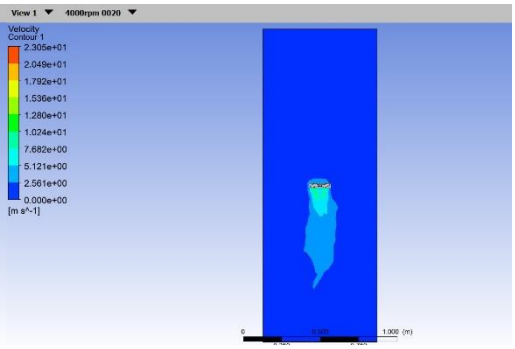
3. 第三版風扇

圖 4.1-36 至圖 4.1-43 為第三版風扇於模擬中在 3000 rpm 至 10000 rpm 之速度分布，其速度範圍從 0 至 23.05 m/s，可以發現當轉速越快時，風扇出風口速度增加(圖中顏色變深)，但速度分布較第二版風扇略為發散。

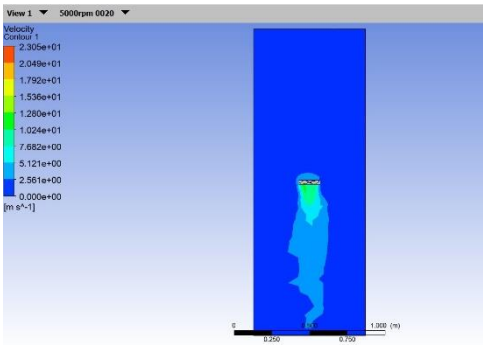
另外，圖 4.1-44 至圖 4.1-47 為 3000 rpm 至 10000 rpm 之流場表現(streamline)，可以發現不同轉速分布差異不大，且結果與前兩版風扇類似，皆為從風扇旋轉抽氣後，匯聚後於出風口噴出，但在出風口後端的環形 streamline 之現象較不明顯，氣流線最為整齊。最後，此版本風扇於低轉速時已有出風口處流線扭轉之現象，且轉速越高，扭轉現象及密度越為明顯，表風較為集中，不易發散，也可有效的產生 torque。



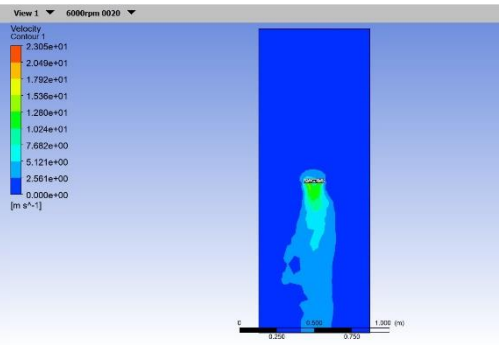
(圖 4.1-36) velocity @3000 rpm



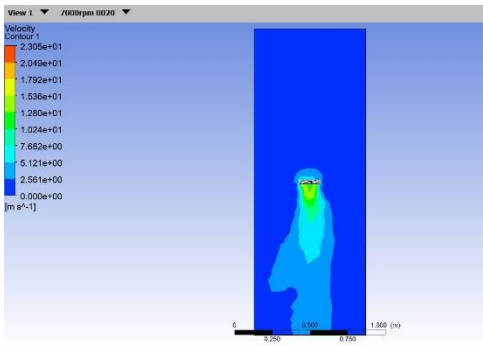
(圖 4.1-37) velocity @4000 rpm



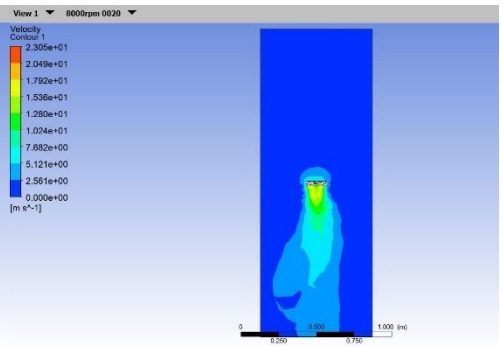
(圖 4.1-38) velocity @ 5000 rpm



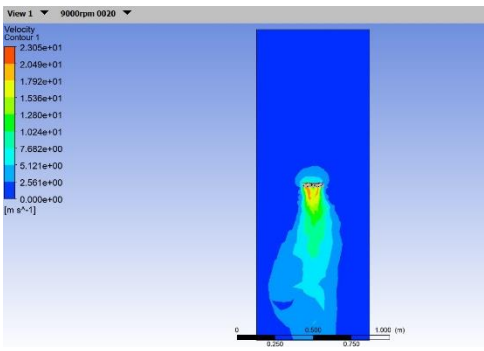
(圖 4.1-39) velocity @6000 rpm



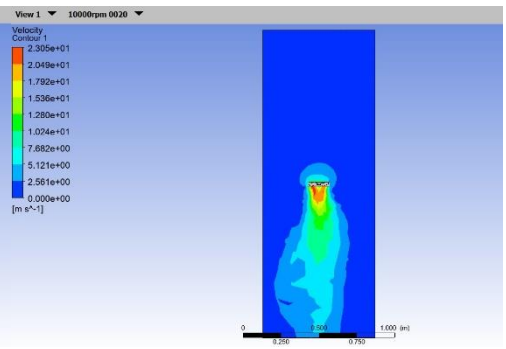
(圖 4.1-40) velocity @7000 rpm



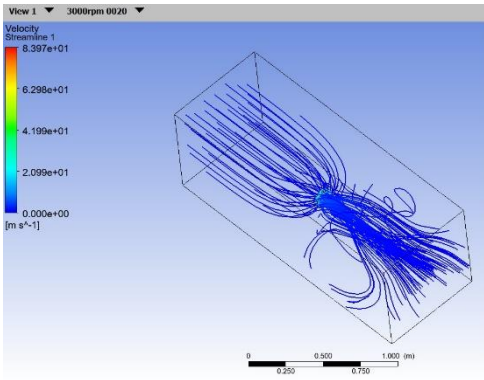
(圖 4.1-41) velocity @8000 rpm



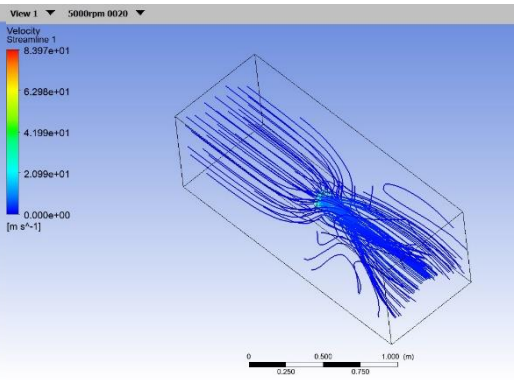
(圖 4.1-42) velocity @9000 rpm



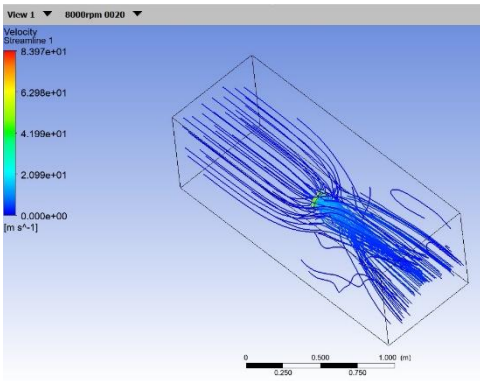
(圖 4.1-43) velocity @10000 rpm



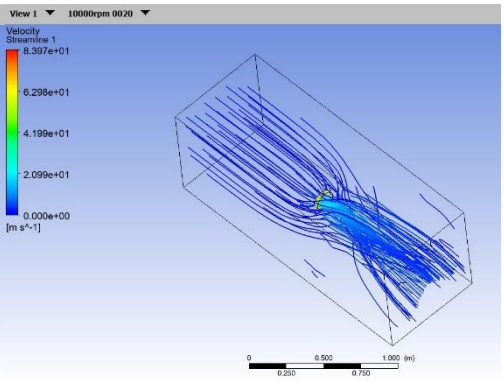
(圖 4.1-44) streamline @3000 rpm



(圖 4.1-45) streamline @5000 rpm



(圖 4.1-46) streamline @8000 rpm



(圖 4.1-47) streamline @10000 rpm

4. 三種版本風扇參數比較

以下將討論三種版本之風扇在不同轉速下的推力 (force)、質量流量 (mass flow)、最大速度 (max. velocity)表現，並相互比較與討論。

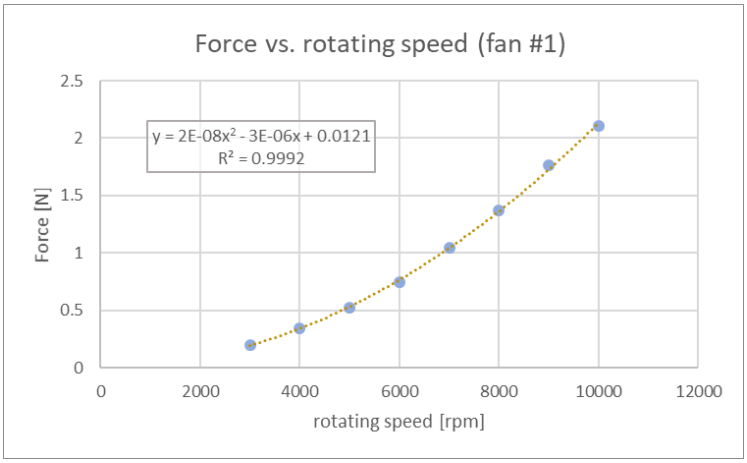
表 4.1-1 為三種版本風扇在不同轉速下各版本所對應之推力。首先，可以發現轉速越高，其推力也越高，其中以第三版風扇推力數值最高。而因模擬為數值解而非解析解，為確保結果之正確性，以下進行一簡單驗證：引用 3.1 節之式 3.1.25，描述推力及速度之關係：

$$T = A_{disk}(p_d - p_u) = \rho A_{disk} \frac{u_3^2 - u_1^2}{2} \qquad (式\ 3.1.25)$$

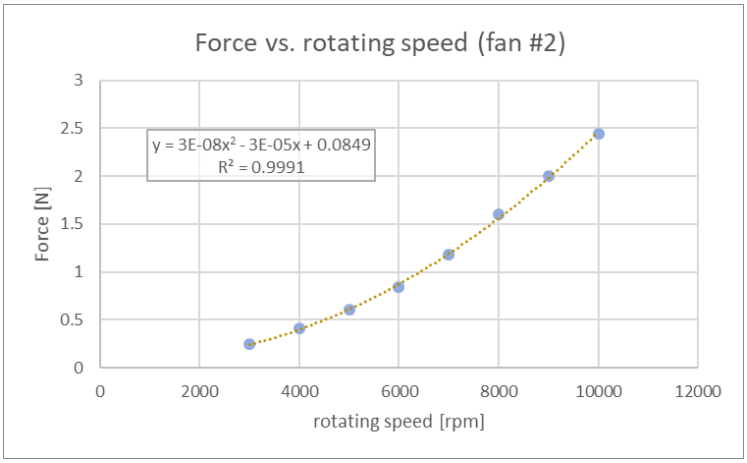
理想上，推力與其流速成二次正比，又$v = r\omega$，速度與轉速成正比，故推力理論上與轉速成二次正比(意即$T \propto \omega^2$)。因此，藉上述關係，將其推力及轉速作圖後經二階多項式回歸後，其結果如圖 4.1-48 至圖 4.1-50 所示。可以發現R^2範圍藉於 0.9992 至 1 間，表高度正相關，代表兩者數值符合理想關係。

表 4.1-1 不同轉速下各版本之 Force [N]

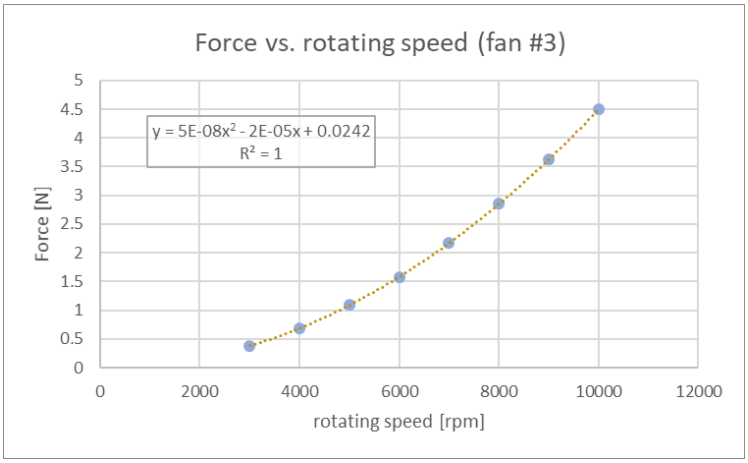
[rpm]	fan #1	fan #2	fan #3
3000	0.201251	0.242337	0.380379
4000	0.348694	0.40882	0.686472
5000	0.524147	0.608002	1.08774
6000	0.750985	0.83785	1.58123
7000	1.04212	1.17783	2.16703
8000	1.36963	1.60085	2.84955
9000	1.76391	1.99965	3.62624
10000	2.10635	2.43691	4.49363



(圖 4.1-48) Force vs. rotating speed (fan #1)



(圖 4.1-49) Force vs. rotating speed (fan #2)

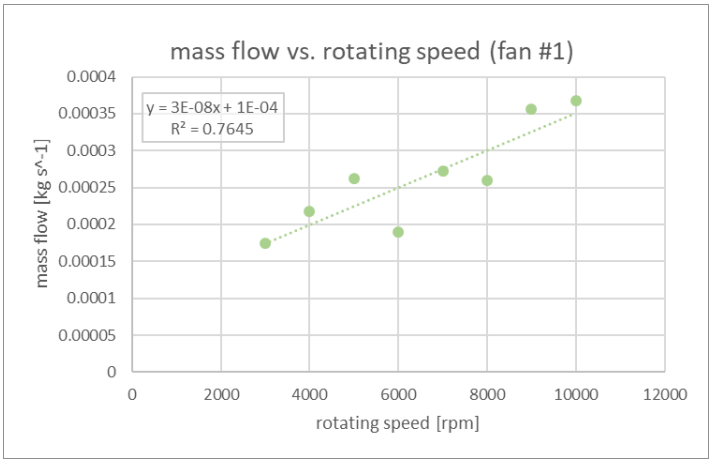


(圖 4.1-50) Force vs. rotating speed (fan #3)

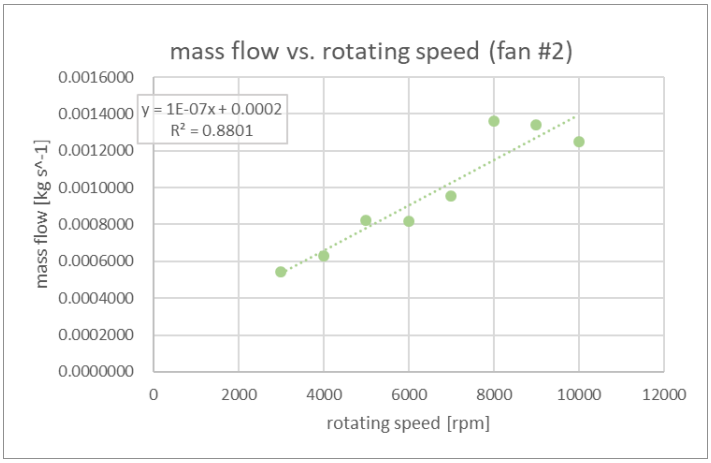
而討論 mass flow，表 4.1-2 為三種版本風扇在不同轉速下各版本所對應之 mass flow。可以發現轉速越高，其 mass flow 也越高，其中也以第三版風扇 mass flow 數值最高，與推力趨勢吻合(轉速越高 mass flow 也越高)。另外，將轉速與 mass flow 作圖，並行線性回歸，其結果如圖 4.1-51 至圖 4.1-53 所示。R^2 範圍藉於 0.7645 至 1 間，相關性無較推力與轉速之關係好，只有第三版風扇為高度正相關。推測此原因可能為考慮出風口之流線，由前面 streamline 圖可之第一版風扇及第二版風扇較第三版風扇混亂，在不同轉速下表現差異也較大，如此造成第一版風扇及第二版風扇在計算 mass flow(有效截面上單位時間通過之流體質量)時產生偏差，R^2較小。因此，此回歸結果與前面 streamline 圖之結果相呼應，第三版風扇之出風口的流線較為其他二者整齊，mass flow 最大，計算結果也與轉速成正比。

表 4.1-2 不同轉速下各版本之 mass flow [kg s^{-1}]

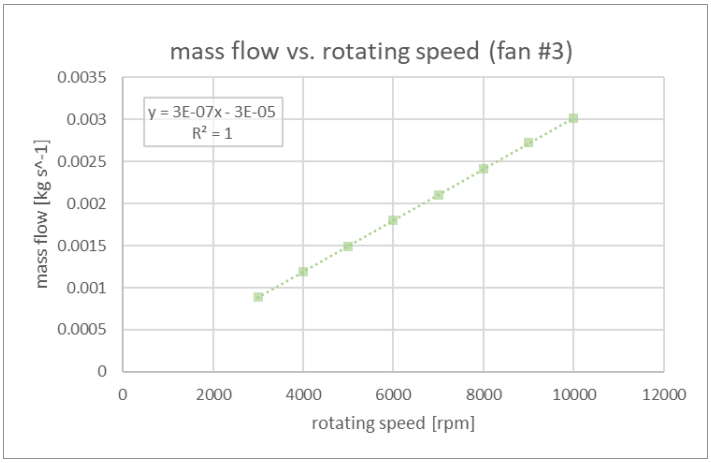
[rpm]	fan #1	fan #2	fan #3
3000	0.00017421	0.0005433	0.000891883
4000	0.00021771	0.0006306	0.0011877
5000	0.00026244	0.0008222	0.00148889
6000	0.00019004	0.0008167	0.00179658
7000	0.00027184	0.0009555	0.00210285
8000	0.00025961	0.0013617	0.00240912
9000	0.00035638	0.0013407	0.00272082
10000	0.000367602	0.0012515	0.00301155



(圖 4.1-51) mass flow vs. rotating speed (fan #1)



(圖 4.1-52) mass flow vs. rotating speed (fan #2)

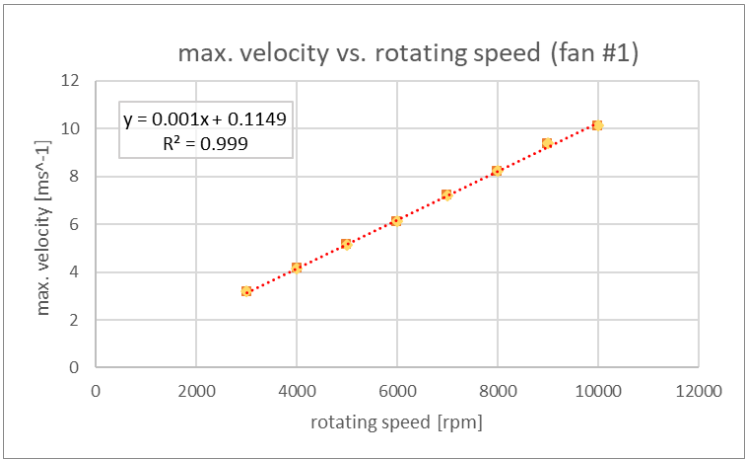


(圖 4.1-53) mass flow vs. rotating speed (fan #3)

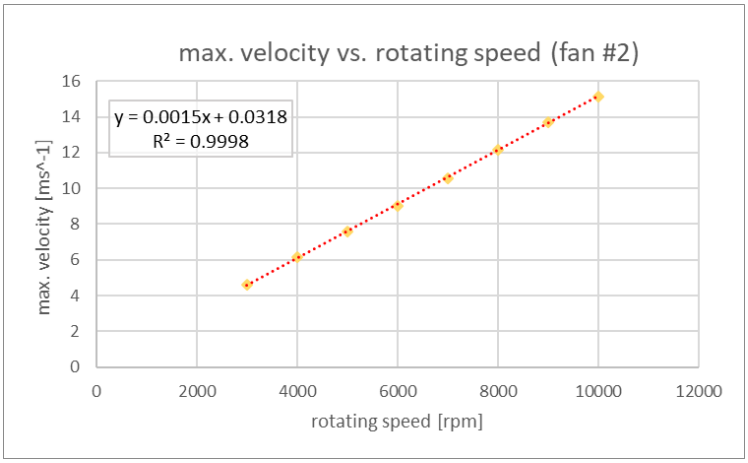
最後討論 max. velocity，表 4.1-3 為三種版本風扇在不同轉速下各版本所對應之 max. velocity。可以發現轉速越高，其 max. velocity 也越高，其中也以第三版風扇速度數值最高，與推力趨勢吻合(轉速越高，速度也越高)。另外，將轉速與 max. velocity 作圖，並行線性回歸，其結果如圖 4.1-54 至圖 4.1-56 所示。R^2範圍藉於 0.9998 至 1 間，皆為高度正相關，轉速與流場最大速度成正比，符合理想關係($v = r\omega$)。

表 4.1-3 不同轉速下各版本之 max. velocity [m s^{-1}]

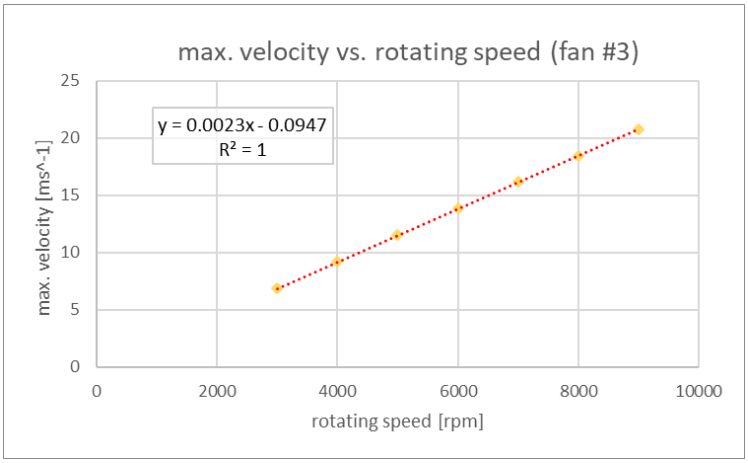
[rpm]	fan #1	fan #2	fan #3
3000	3.17807	4.59751	6.83878
4000	4.16537	6.13314	9.18088
5000	5.14402	7.56859	11.5107
6000	6.11849	9.0162	13.8343
7000	7.20793	10.5461	16.1508
8000	8.23429	12.1741	18.4542
9000	9.37416	13.699	20.7505
10000	10.1217	15.1454	23.0463



(圖 4.1-54) max. velocity vs. rotating speed (fan #1)



(圖 4.1-55) max. velocity vs. rotating speed (fan #2)



(圖 4.1-56) max. velocity vs. rotating speed (fan #3)

5. 結論

由上述分析及討埨，藉 ANSYS 將三版本風扇行流場分析及各參數計算後，可得以下結論：

(1) 速度分布以第二版風扇較為集中，但第三版風扇數值最高。
(2) streamline 在第一版風扇及第二版風扇較為混亂(似環形流線)，第三版風扇於出風口處有流線扭轉現象，表流線不易發散，可想成扭轉現象使推力可有效的產生 torque，進而推動整個車體。
(3) 推力以第三版風扇數值最大，且結果而言推力與轉速成二次正比。
(4) mass flow 以第三版風扇數值最大，且結果而言推力與 mass flow 成正比，另外，第一版風扇及第二版風扇相關性較小，推測原因為出風口流線較第三版風扇混亂，使不同轉速下 mass flow 計算產生偏差，變異性較大。
(5) max. velocity 以第三版風扇數值最大，且結果而言推力與 max. velocity 成正比。

故由上述結論，我們認為第三版風扇最符合我們的需求 (推力大且流場穩定)，其結果也與 3.1.5 節 (風扇設計)之結果 (經多次修改分析及實測後，第三版風扇在小轉速即可獲得所需推力)相呼應，此為本組最終版本之風扇。

4.1.2. 風扇推力實驗--風洞實驗

(1) 實驗儀器：轉動測速計、熱線風速儀、風洞裝置
(2) 實驗目的：瞭解風扇在不同 PWM duty 下產生之推力及其和轉速的關係。
(3) 實驗說明：

風洞車主要驅動動力源自於風扇，因此設計風扇需能產生足夠的推力。我們希望透過風洞實驗，分析該扇葉的產生推力效能為何，另外，亦紀錄輸入 PWM duty(Servo.write() input 值)和扇葉轉速、推力的關係。

風洞實驗主要是根據 Actuator Disk Theory 的推導，計算推力。其流程是先利用可視化煙線觀察經扇葉加速的流場，找到 streamtube 作為 control volume，量測各點流速及壓力分布後，根據 Actuator Disk Theory 分析，算出風洞車扇葉在不同風洞背景流速下產生的推力(thrust)大小。詳細公式推參考 3.1.2 章節。

(4) 實驗數據：(這邊使用量測實驗二的數據做為參考)

裝置示意圖如下。其中x_1, x_3為根據煙線所找到的 streamtube 要量測風速分佈的平面，我們在這個平面上取幾個不同高度位子h_1, h_3的點量測風速，p_u, p_d分別為上游、下游測量壓力的位子，另外在風扇轉動前會紀錄背景流速作為校正，數據如下表。（圖 4.1-57）

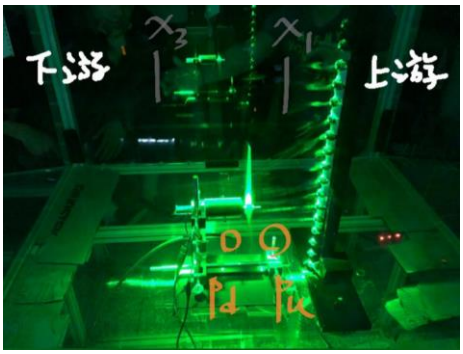


圖 4.1-57 風洞實驗裝置示意圖

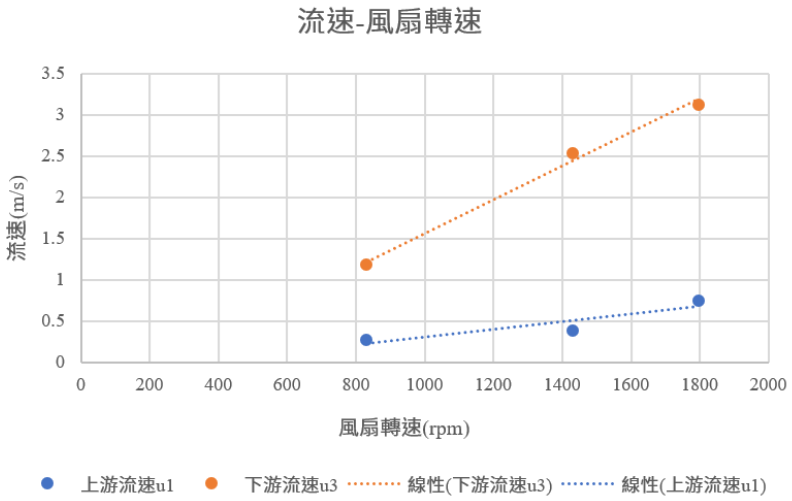
position	w1	w2	w3	average
背景流速U∞(m/s)	0.18	0.23	0.18	0.1967

Fan mode 1 Servo = 75, fan speed = 1796rpm				Fan mode 2 Servo = 80, fan speed = 1430rpm				Fan mode 3 Servo = 85, fan speed = 830.9rpm			
上游 Pu' = -1.87		下游 Pd' = 6.28		上游 Pu' = -1.38		下游 Pd' = 3.53		上游 Pu' = -0.86		下游 Pd' = 0.68	
h1(cm)	Uinlet(m/s)	h3 (cm)	Uthroat(m/s)	h1(cm)	Uinlet(m/s)	h3 (cm)	Uthroat(m/s)	h1(cm)	Uinlet(m/s)	h3 (cm)	Uthroat(m/s)
20	0.19	7	1.94	18	0.15	7	3.52	14	0.27	7	0.94
15	0.24			13.5	0.27			10.5	0.38		
10	1.06	3.5	4.1	9	0.79	3.5	3.65	7	0.45	3.5	2.35
5	1.55	0	3.94	4.5	0.83	0	1.03	3.5	0.53	0	0.85
0	1.71			0	0.87			0	0.72		
mean	0.95		3.32666667		0.582		2.73333333		0.47		1.38
var	0.50785		1.44853333		0.11792		2.18023333		0.02865		0.7077

將上表原始數據最後得到的位於 inlet 和 throat 的 mean velocity speed 減去背景風速U_∞，得到 U_1和U_3，整理數據如下表。

	電壓V	fan speed(rpm)	U1 (m/s)	U3 (m/s)
Fan mode 1	1.8	1796	0.7533	3.129967
Fan mode 2	1.2	1430	0.3853	2.536633
Fan mode 3	0.6	830.9	0.2733	1.1833

將上表所得風扇轉速和流速關係繪製如下圖，可以看出流速和風扇轉速有正比關係。



根據 3.1.2 推導的 Actuator Disk Theory 可以計算風扇處的流率如下式：

（A_{disk}和 D_{disk} 為分別為風扇投影面積和直徑）

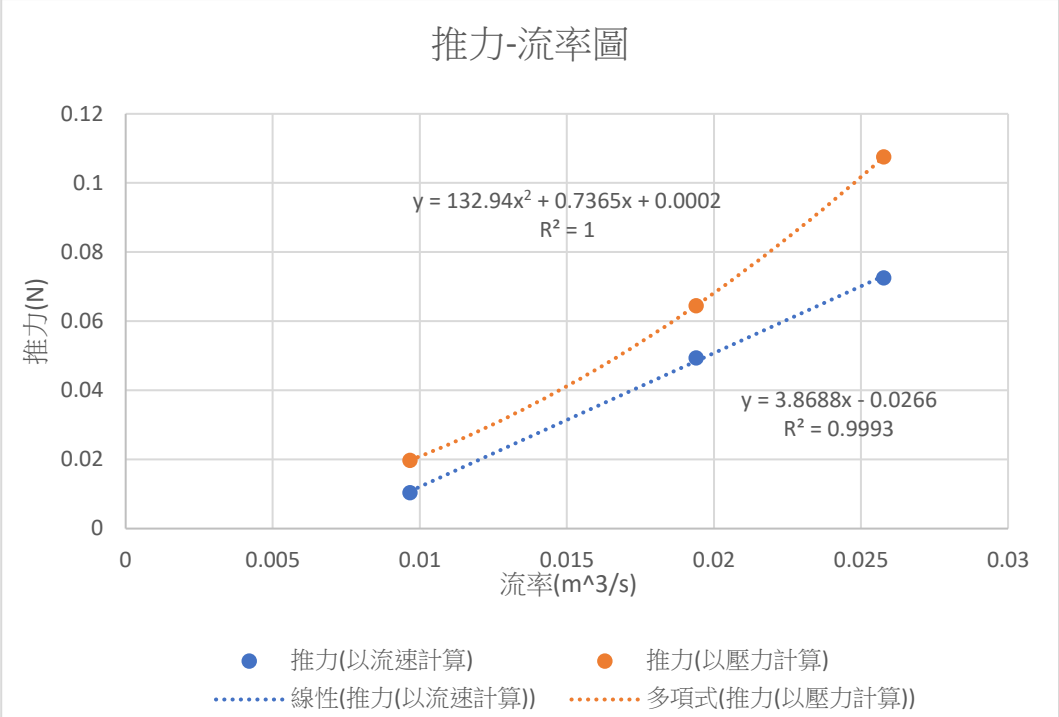
$$Q = A_{disk}u_2 = D_{disk}^2\pi/4 \times \frac{u_1 + u_3}{2}$$

推力如下式：

$$T = A_{disk}(p_d - p_u) = \rho A_{disk} \frac{u_3^2 - u_1^2}{2}$$

	電壓V	fan speed(rpm)	U2(m/s)	Q(m^3/s)
Fan mode 1	1.8	1796	1.941633333	0.025771722
Fan mode 2	1.2	1430	1.460966667	0.019391729
Fan mode 3	0.6	830.9	0.7283	0.009666884

	電壓V	fan speed(rpm)	U1 (m/s)	U3 (m/s)	U1/U3	推力(N)	Pd (pa)	Pu (pa)	推力(N)
Fan mode 1	1.8	1796	0.7533	3.129966667	0.240673	0.07252	6.84	-1.26	0.107513
Fan mode 2	1.2	1430	0.3853	2.536633333	0.151894	0.04939	4.09	-0.77	0.064508
Fan mode 3	0.6	830.9	0.2733	1.1833	0.230964	0.01042	1.24	-0.25	0.019777



(5) 實驗預期結果

根據風洞實驗，不僅能得到風扇轉速、流速關係，也能進一步換算出風扇轉速和推力的關係，這些實驗數據雖然對於控制沒有很直接的幫助，但有助於初步判斷扇葉性能、瞭解 duty 和轉速及推力的相關性，也能提供 4.6.1 Simulink 模擬驗證時重要參數。

在下個章節，我們設計另一個更有助於控制的加速度實驗，可以直接得到輸入 Servo.write()和風扇帶動車體實際產生的加速度關係。

4.1.3. 加速度實驗

(1) 實驗儀器：風動車、彈簧秤

(2) 實驗目的：得到車體加速度和風扇直流馬達指令的關係。

(3) 實驗說明：

在風洞實驗中，我們已求得風扇轉動特定轉速時會產生多少推力，然而實際車體所受的總推力和車輪摩擦力亦有關係。為了瞭解輸入指令後，風扇和地面摩擦對車體產生的總推力為何，以方便控制，我們設計加速度實驗如下：

在平坦地面上，將彈簧秤一端綁在固定的椅子上，另一端掛在車尾（圖 4.1-58），對風扇輸入 servo 指令使其轉動，此時彈簧秤讀值即為車體所受到前進的推力。這邊我們假設繩子剛性極好沒有任何拉伸。接著，我們改將彈簧稱掛在車頭，對風扇輸入 servo 反轉的指令，再紀錄一次上述實驗以得到車體倒車走時的加速度。

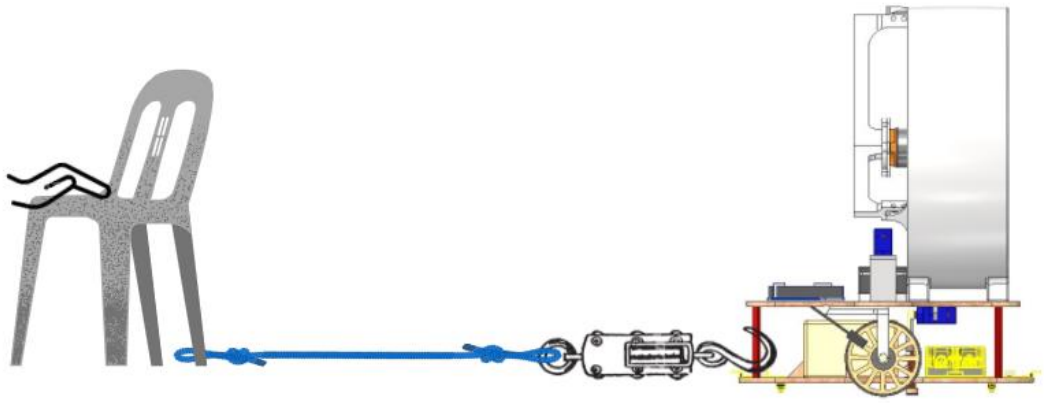


圖 4.1-58 加速度實驗裝置示意圖

(4) 實驗數據： (Servowrite = 90 代表停止、 > 90 逆時針轉、< 90 順時針轉)
註：因疫情因素無法實際量測數值，此以實驗設計為主，並以###代替數值

Car weight(kg)	###				
車子正走			車子倒走		
Servo write	推力(kgw)	加速度(m/s^2)	Servo write	推力(kgw)	加速度(m/s^2)
88	###	###	92	###	###
86	###	###	94	###	###
84	###	###	96	###	###
82	###	###	98	###	###
80	###	###	100	###	###
78	###	###	102	###	###
75	###	###	105	###	###

(5) 實驗預期結果：

在期中測試前，我們發現車子並非給輸入指令就會前進，因為輪子本身和地面的摩擦力需要克服，因此輸入數字太小時車體是不會前進的。在這個加速度實驗中，應該能找到驅動車子前進的最小輸入值。

此外，我們也發現車子倒走的推力明顯較前進還要弱，需要較大的輸入值才能推動車體。撇除雙向電變對於正反向等量值輸入可能導致輸出不一致的轉速外，車體幾何形狀也會影響。由於風罩在後側有一些固定直流馬達的支撐架，該支撐架會擋住風口、擾亂風場，使得出口處風力減弱無法有效推動風動車。上述這個現象應在風洞實驗也會發現，而本加速度實驗則能提供更好控制的參數。

4.1.4. 風扇模擬推力與爬坡加速度分析

(1) 實驗儀器：風動車、彈簧秤、斜坡

(2) 實驗目的：藉模擬之風扇推力，得到於斜坡上穩定上升所需轉速，並與實驗值比較。

(3) 實驗說明：

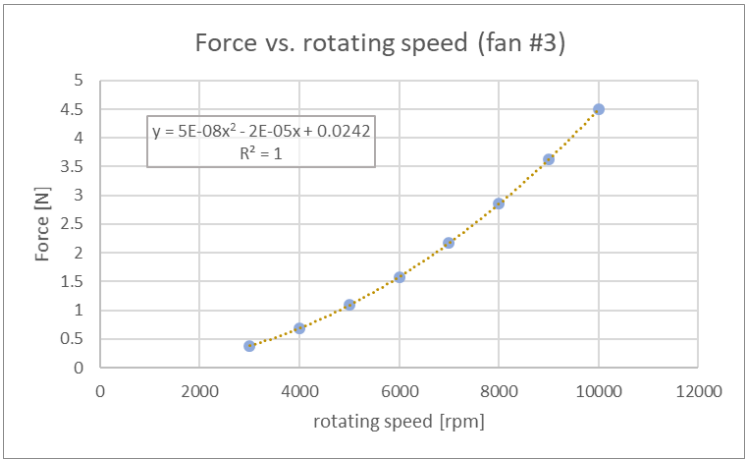
(a) 理論組

於 4.1.1 節中，藉 ANSYS 之流場分析，我們得知風扇在不同轉速下之推力。而期末測試中，爬坡為車體推力考量之一大因素，故考慮一坡道，理想上車體推力需克服地板摩擦力($f_s = \mu_s Mg\cos\theta$，考慮車輪純滾動)以及車重(Mg)於斜坡上之分量，意即

$$T \geq f_s + Mg \cdot \sin\theta = Mg(\mu_s \cos\theta + \sin\theta)$$

其中θ為斜坡傾斜角度。而由 4.1.1 節可知，推力與轉速成二次正比 ($R^2 = 1$)，利用模擬結果所作的二次回歸方程式及其作圖如下所示，故可得模擬數值下推力(T)及轉速(ω [rpm])之關係為：

$$T\ [N] = \omega^2 - 0.00002\omega + 0.0242$$



而為求得轉速與實際上訊號輸入(Servowrite)之關係，本組所使用無刷馬達為 2250 kV，其中 kV 之參數意義為 1V 電壓下所能提供之轉速(rpm)，所選用電池電壓為 11.1 V，故可得以下關係：

$$\omega\ [\text{rpm}] = 11.1 \times 2250 \times \text{duty}$$

其中 duty 為訊號輸出強度比。另外，程式上利用 Servoswite()控制，Servowrite = 90 代表停止、 > 90 逆時針轉、< 90 順時針轉，duty 與數值為線性關係，意即：

$$\text{duty} = \frac{90 - \text{servowrite}}{90}$$

因此，推力(T)與訊號輸入(Servowrite)之關係為：

$$\begin{cases} T\,[N] = \omega^2 - 0.00002\omega + 0.0242 \\ \omega\,[\text{rpm}] = 11.1 \times 2250 \times \dfrac{90 - \text{servowrite}}{90} \end{cases}$$

且計算推力需滿足以下不等式：$T \geq Mg(\mu_s cos\theta + sin\theta)$。如此，我們可求出於模擬推力結果為基礎下，在不同角度時，車體所需最小推力及 Servowrite 數值，此為理論組。

(b) 實驗組

於實驗組中，將風動車放於斜坡起時點上，此改良 4.3.1 之實驗，將彈簧秤一端用手拉住，另一端掛在車尾，並至於斜坡上，調整一固定角度，此時施予一力使風動車停至斜坡上，並讀取彈簧秤數值。之後改變斜坡角度，並重複上述步驟。此為實驗組

(4) 實驗數據：

註：因疫情因素無法實際量測數值，此以實驗設計為主，並以###代替數值

理論組				實驗組		
weight [kg]				weight [kg]		
μ_s				μ_s		
角度 [deg.]	推算最小推力 [N]	Servowrite()	加速度 [ms^-2]	角度 [deg.]	最小拉力 [N]	加速度 [ms^-2]
5	###	###	###	5	###	###
10	###	###	###	10	###	###
15	###	###	###	15	###	###
20	###	###	###	20	###	###
25	###	###	###	25	###	###
30	###	###	###	30	###	###

(5) 實驗預期結果：

在期中測試後，我們有先將測試風動車需施予給予多少 Servowrite()才可以順利穩定爬上坡，雖大約 20%~30% duty 即可達成，但發現轉速在一段時間後降低，使滑至地面，推測其原因可能與電源供應上還不穩定或 duty 量值太小，使推力不足，也因此需藉助模擬數值大約推得足夠的推力及 duty。藉由此實驗，我們可以將模擬情形與實際情形比較，由此推估出在不同角度下實際所需推力及轉速，使風動車在斜坡上進行其他行為(如尋機或轉彎)能更加穩定。

4.2. 轉向

4.2.1. 轉向角分析

以下將針對第一版轉向機構進行分析。

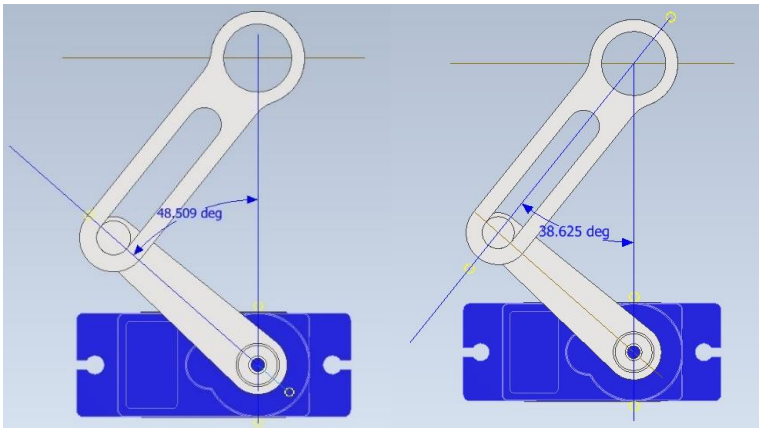


圖 4.2-1 第一代最大轉向角度

此連桿機構在伺服馬達來回轉動時能夠透過槽來帶動轉向軸。利用 Autodesk Inventor 建立馬達延伸桿件及轉向槽的模型，並約束馬達轉軸與車輪轉向軸位於同一直線、距離為 36 mm。測量出馬達最大可輸出角度為 48.509°，車輪對應之最大轉向角度為 38.625°，可見此機構所傳遞之旋轉角度並非呈線性關係，且真實轉向角度將小於伺服馬達輸出角度。

我們利用 Inventor 之動力學模擬，模擬伺服馬達的運動過程：先將桿件皆置中後，使連接伺服馬達的桿件轉至左方 45°角位置、再轉至右方 45°角位置，最後轉回中央位置。過程花費 4 秒，旋轉角速度為 ±45 deg/s。

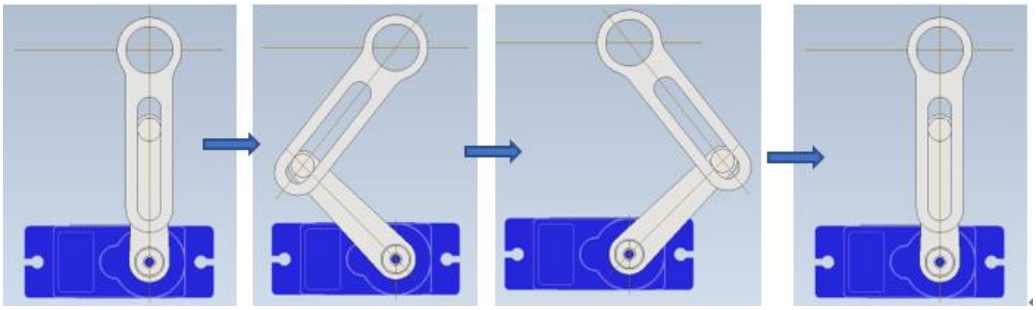


圖 4.2-2 轉向模擬示意圖

可以得到轉向角度關係圖如下：(以中央位置為 0°、逆時針方向為正)

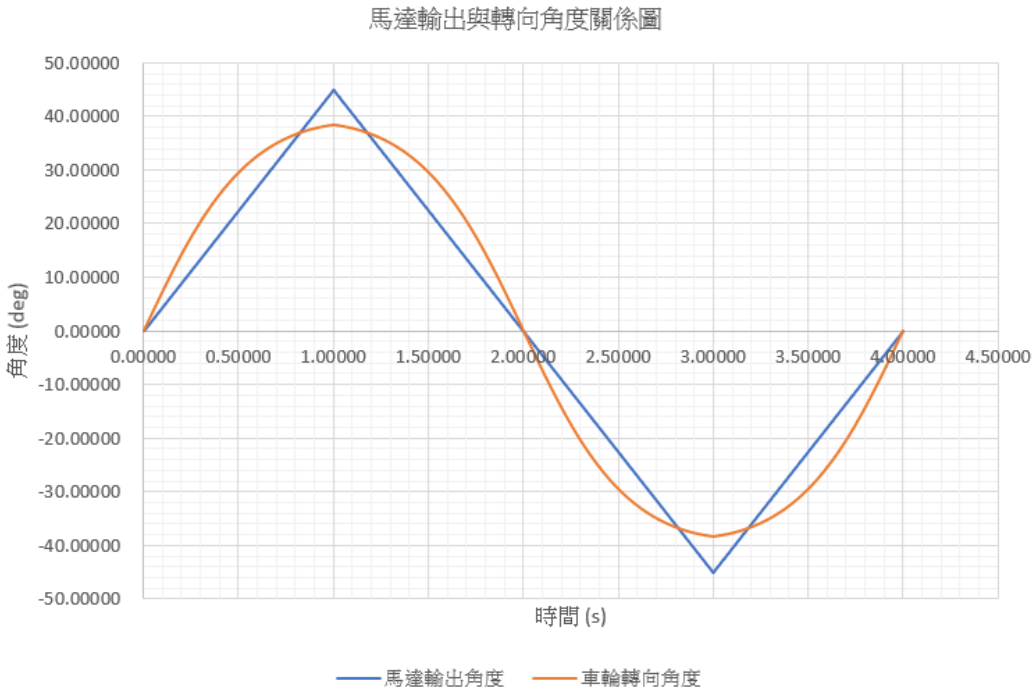


圖 4.2-3 馬達輸出與轉向角度關係圖

圖中藍線為馬達輸出角度、橘線為車輪轉向軸對應的旋轉角度。可以看出在伺服馬達等速旋轉的情況下，車輪轉向角度的變化更接近於正弦函數曲線。而我們大概能以 30°作為一分界，當馬達輸出的角度大於 30°時，轉向角度的變化會非常不明顯；而當馬達輸出小於 30°時，實際轉向角度皆會比馬達輸出角度還大。

這樣的現象會增加控制上的難度。由於控制端能夠改變的是馬達的旋轉角度，若與真正轉向角度的關係太過複雜，則只能透過一次次的測試來修正路徑，整體的效率不高。

因此在第二版轉向機構中，我們直接將伺服馬達置於轉向軸上，使馬達輸出角度與轉向角度一致，最大轉向角度也不會受限於機構。

4.2.2. 分力分析

第二版機構在轉向時，車體能夠以水平移動來轉換方向，也就是說在轉向過程中，車身並不會有任何轉動，而僅由車輪的旋轉來控制方向。所以，在車體行進過程中，風扇從頭到尾皆向後方吹，推力的方向並不會改變。當行進方向改變時，風扇推力便無法完全平行於前進方向。然而，只有平行前進方向的分力能繼續推動車體，以下將分析轉向角度與推力的關係。

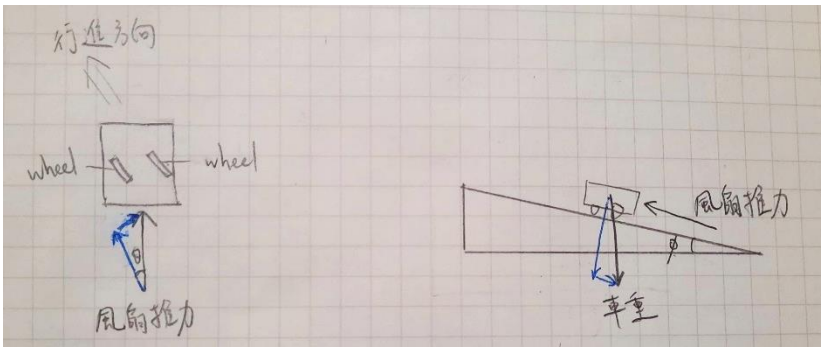


圖 4.2-4 推力分力示意圖

上圖中，假設最大推力為 F_{fan}，轉向角為 θ，坡面的仰角為 φ。
當車輪旋轉 θ 時的推力為$F_{fan} \times \cos\theta$ (N)。若要讓車體沿著斜坡向上，推力至少要大於$(\text{車重}) \times 9.8 \times \sin\phi$ (N)，預估車重為 1.202 kg (4.2.2 中說明)。

期末測試之指定仰角為 10°、15°、20°、25°。若令轉向角為 0°，則可計算出各仰角下所需風扇推力之最小值如下表：

表 4.2-1 各仰角下之推力最小值 F_{min}

仰角	10°	15°	20°	25°
最小推力 F_{min} (N)	2.0455	3.0488	4.0289	4.9782

表 4.2-2 風扇不同轉速下對應之推力

風扇轉速 (rpm)	風扇推力 F_{fan} (N)
3000	0.38038
4000	0.68647
5000	1.08774
6000	1.58123
7000	2.16703
8000	2.84955
9000	3.62624
10000	4.49363

上表利用 ANSYS 模擬風扇在不同轉速下所能產生的推力，比較兩表數據即可得知在不同仰角下適合使用的風扇轉速、以及當轉速達到 10000 rpm 時所能應付的最大轉向角度。以下表格整理不同仰角下的情況：

表 4.2-3 不同仰角下之要求

仰角	所需推力最小值 F_{min} (N)	適用最低轉速(rpm)	最大轉向角度(deg)
10°	2.0455	7000	62.9222
15°	3.0488	9000	47.2757
20°	4.0289	10000	26.2879
25°	4.9782	X	X

適用最低轉速參考表 4.2-2 中之整數值。由於$F_{\text{fan}} \times cos\theta > F_{min}$，取用 10000 rpm 的 F_{fan}，利用此關係式計算最大轉向角度$\theta = \cos^{-1}\left({F_{min}}/{F_{\text{fan}}}\right)$。但由於 ANSYS 模擬結果止於 10000 rpm，並不足以攀上 25°仰角的坡。實際上風扇轉速還可以提高，因此利用模擬結果之回歸線(見圖 4.1-50)計算 11000 及 12000 rpm 所能產生的推力，以及兩轉速在仰角 25°坡上分別能提供的最大轉向角度。

表 4.2-4 仰角 25°加大轉速後的最大轉向角度

風扇轉速 (rpm)	風扇推力 (N)	最大轉向角度 (deg)
11000	5.8542	31.7487
12000	6.9842	44.5384

4.3. 煞車

4.3.1. 摩擦力實驗

(1) 實驗儀器：重物、煞車皮、輪胎皮。

(2) 實驗目的：測試煞車皮和輪胎皮的淨摩擦係數，以利在期中測試和期末測試中計算車體能否在平面或斜坡上停住。

(3) 實驗說明：

(a) 實驗原理：

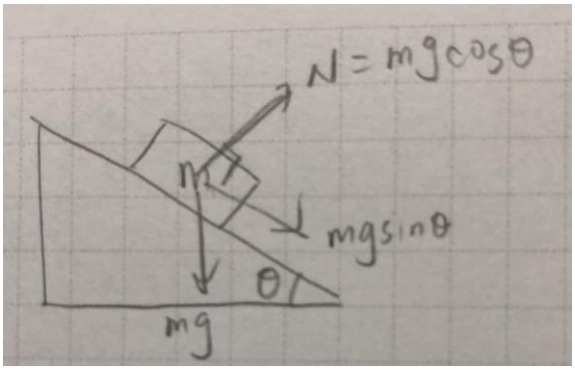
當重物在斜坡上恰要移動時，沿著斜面的重力分量會和最大靜摩擦力達靜力平衡，即:

$$mg\ sin\ sin\ \theta\ =\ Fs\ =\ \mu_S N\ =\ \mu_S mgcos\theta$$

由上式可知:

$$\mu_S\ =\ tan\theta$$

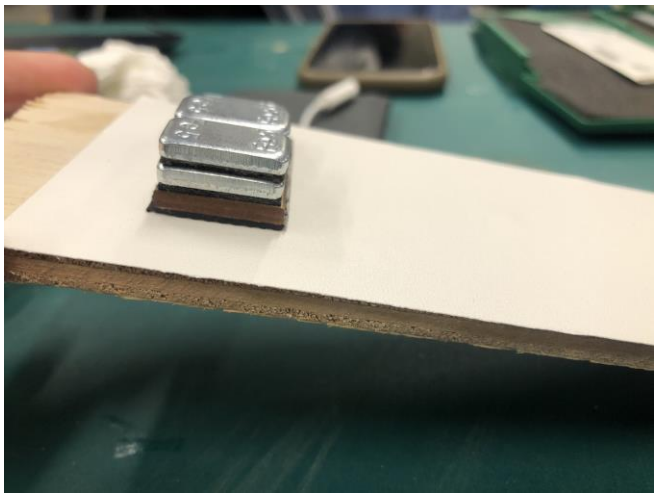
因此，藉由控制斜坡角度，在重物剛好滑下時，斜坡角度的tan值即為重物和斜坡接觸面的靜摩擦係數 (圖 4.3-1)。



(圖 4.3-1) 分力圖

(b) 實驗配置

將斜坡上貼上期中、期末測試時場地的表面材質，並將重物底部貼上我們的煞車皮，並準備手機內建的水平儀應用程式以測量坡度，如圖 4.3-2 所示。



(圖 4.3-2) 實驗配置圖

(c) 實驗步驟

將重物置於斜面上，緩慢增加斜坡角度，當重物恰滑動時，測量斜坡角度，重複五次上述步驟。

(4) 實驗數據：

(表 4.3-1) 實驗數據

#	角度	tanθ (靜摩擦係數)
1	30.7	0.59375655
2	31	0.600860619
3	30.6	0.591398351
4	31.5	0.612800788
5	28.7	0.547484008
average	30.5	0.589260063

(5) 實驗結果：

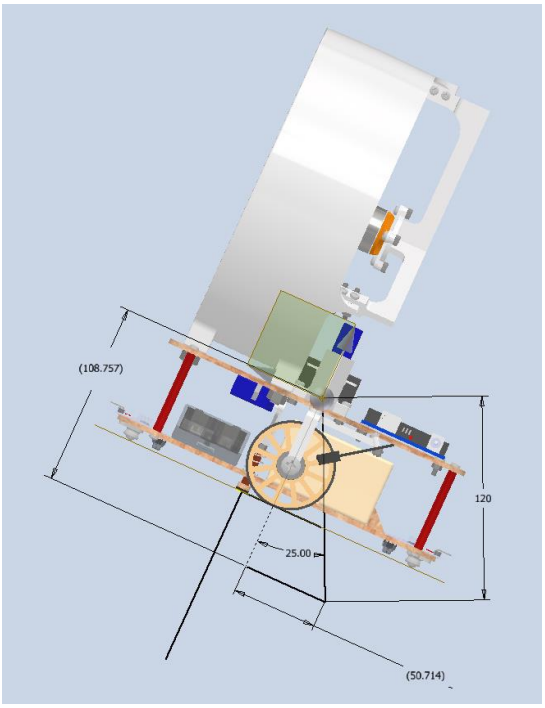
由數據我們可以發現，每次的量測結果差異不大，測得靜摩擦係數的平均值為 0.59。以此結果推論，當在期末測試的坡上時，尚未考慮伺服馬達施予煞車系統的力前，最大靜摩擦力就應足以使車體停在約 30 角度的斜坡上。

4.3.2. 煞車需求分析

如同設計概念中所提到的，期末的煞車系統需要使得車輛在斜坡上停滯，此部分將分析所需求的摩擦力。

已知條件:假設車重 1200g、質心使用軟體內件功能計算(方法在 4.4.2 質量與質心中說明)

我們以期末測試斜坡傾斜最大角度 25 度為設計需求，繪製風扇靜止情況下車身 free body diagram 如圖 4.3-3 所示：



(圖 4.3-3) 側視力平衡圖

透過力平衡圖可知若車重為 1200g，煞車所受正向力為 108.76gw，其所需摩擦力為 50.7gw。摩擦力實驗中所得到的靜摩擦係數為 0.59，108.76*0.59 = 64.1684，大於需求的 50.7，由此可知我們所使用的煞車皮是可以使用的。

4.4. 車體

4.4.1. 應力分析

應力分析包括期中第一版車子的應力分析以及期末第二版車子的應力分析。

1. 期中應力分析

為了避免車體零件因為應力過大而發生變形等情形，我們針對承受較大應力的幾個機構進行應力分析。

(1) 輪子

由於我們組是用三輪車，因此比起四輪車，每個輪子承受的應力較大，加上輪子的材料是密集板，我們擔心輪框或輪輻會發生變形，因此以下將針對前輪和後輪進行應力分析。

(a) 後輪

材料如圖 4.4-1 所示：

⊟ 材料

名稱	木材 (橡木)	
一般	質量密度	0.76 g/cm^3
	降伏強度	46.6 MPa
	極限抗拉強度	5.5 MPa
應力	楊氏模數	9.3 GPa
	蒲松氏比	0.0001 ul
	剪力模數	4.64954 GPa
零件名稱	wheel.ipt	

圖 4.4-1 後輪材料表

約束：固定約束輪子和培林的接觸面。

負載：車體約為 500gw，計算後得知質心距前輪 x 方向 96.31mm，距後輪 x 方向 91.19mm，由此得知每個後輪承受的徑向力約為 1.26N。

模擬結果：

⊟ **Von Mises 應力**

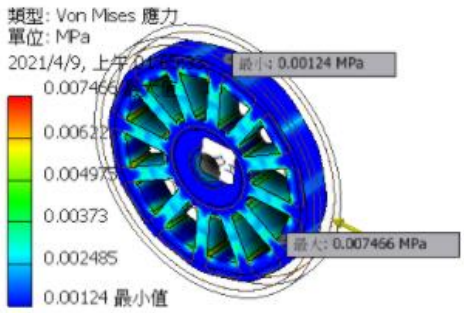


圖 4.4-2 後輪應力分析結果

從圖 4.4-2 的應力分析結果我們可以觀察到，受到最大 Von Mises 應力為 0.0075MPa，遠小於材料的降伏強度(木材降伏強度約為 46.6MPa)，承受最大應力的位置位於輪輻上。

(b) 前輪

材料如圖 4.4-3 所示。

⊟ **材料**

名稱	木材 (橡木)	
一般	質量密度	0.76 g/cm^3
	降伏強度	46.6 MPa
	極限抗拉強度	5.5 MPa
應力	楊氏模數	9.3 GPa
	蒲松氏比	0.0001 ul
	剪力模數	4.64954 GPa
零件名稱	wheel.ipt	

圖 4.4-3 前輪材料表

約束：固定約束輪子和培林的接觸面。

負載：車體約為 500gw，計算後得知質心距前輪 x 方向 96.31mm，距後輪 x 方向 91.19mm，由此得每個後輪承受的徑向力約為 2.04N。

模擬結果：

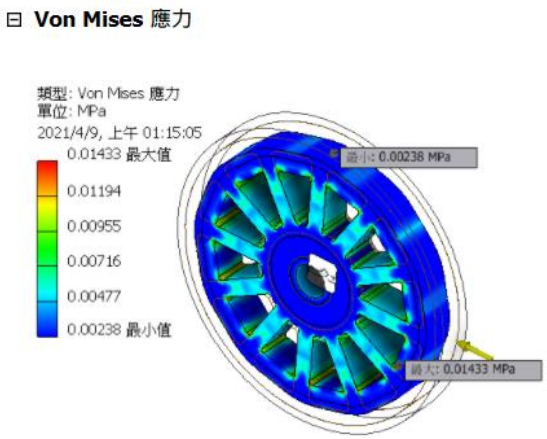


圖 4.4-4 前輪應力分析結果

從圖 4.4-4 的應力分析結果我們可以觀察到，受到最大 Von Mises 應力為 0.01433MPa，略大於後輪承受的最大 Von Mises 應力，但其仍遠小於材料的降伏強度(木材降伏強度約為 46.6MPa)，因此我們認為前輪亦不容易發生變形。

(2)底板材料

材料如圖 4.4-3 所示。

約束：固定約束鄰近前後輪的邊

負載：分別於五個銅柱處乘載 0.4N 的力，而車手(鋁箔包)置於底板上，重量約為 300gw，底面積為 107*65cm，因此將鋁箔包置放處設一個大小為 4.22666615 的壓力，如圖 4.4-5 所示。

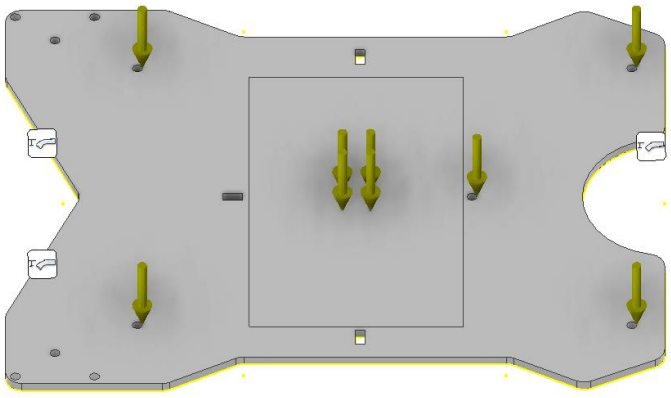


圖 4.4-5 底板模擬施力配置圖

模擬結果：

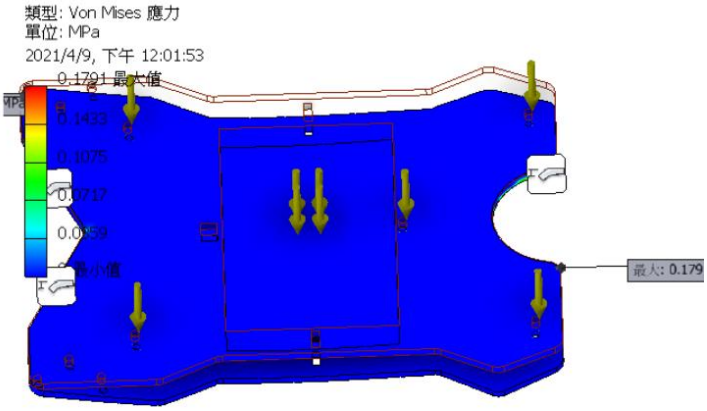


圖 4.4-6 底板應力分析結果

從圖 4.4-6 的應力分析結果我們可以觀察到，受到最大 Von Mises 應力為 0.179MPa，在底板圓弧邊緣發生應力集中，小於材料的降伏強度(木材降伏強度約為 46.6MPa)，因此我們認為前輪亦不容易發生 bending。

2. 期末應力分析

以下將針對第二版車體中，我們認為有可能出現形變的零件進行應力分析，欲分析的零件分別為車子頂板、車子底板以及馬達連接車輪的桿件。

(1) 頂板

分析原因：針對頂板作分析的原因為，頂板上放置著風罩、無刷馬達、鋰電池等重物，且輪子透過和頂板的接觸支撐車體，意即頂板受到了許多重物的重力以及輪子對於車體的向上支撐力，很有可能發生形變。

材料：材料使用密集板，因此使用 inventor 材料庫中最接近密集板性質的木材(樺木)，其性質如圖 4.4-7 所示。

名稱	木材 (樺木)	
一般	質量密度	0.55 g/cm^3
	降伏強度	56.3 MPa
	極限抗拉強度	6.3 MPa
應力	楊氏模數	10.3 GPa
	蒲松氏比	0.0001 ul
	剪力模數	5.14949 GPa
零件名稱	top_board.ipt	

圖 4.4-7 頂板材料表

負載：針對頂板，我們施加了三種向下的力以及一種向上的力。力的施加圖如圖 4.4-8 所示。

向下的力包括：

(a) 風罩和無刷馬達施予頂板的重力，力施加於風罩和頂板的接觸面積上

風罩、風扇的重力和約為 0.695kg，無刷馬達的重力約為 0.07kg，而風罩和頂板的接觸面積為$(6 \times 40 \times 2)mm^2$。

(b)鋰電池重，力施加於鋰電池和頂板的接觸面積

鋰電池重約為 200g，鋰電池和頂板的接觸面積約為$(24 \times 76)mm^2$。

(c)銅柱施予頂板的向下力

銅柱連接到底板，因此底板及底板上重物的重力會透過 4 根銅柱施予頂板向下的力。

底板上的重物有 18650 電池盒、兩個 18650 電池以及車手(鋁箔包飲料)，18650 電池盒和電池總重約 100g、車手重約 300g，這些重量透過 4 根銅柱施予頂板向下的力，換算後每個力為 0.98N。

向上的力包括：

(a) 輪子支撐車重，而輪子透過和馬達的連接桿件，施予頂板向上的支撐力，每個力的大小約為 3.7N，施加在頂板對輪子的 4 個鎖點上。

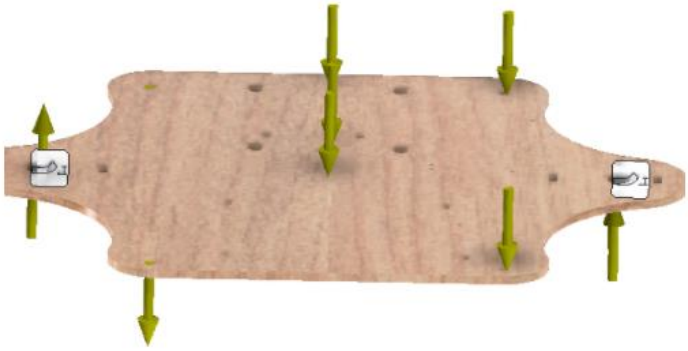


圖 4.4-8 頂板施力配置圖

固定約束：頂板的固定約束為連接輪子的 4 個鎖點，如圖 4.4-9 所示。

圖 4.4-9 頂板約束配置圖

模擬結果：

模擬結果如圖 4.4-10、圖 4.4-11 所示。由 Von Mises 應力分析結果，我們可以發現頂板所受應力皆在安全範圍內，最大值為 0.01MPa，遠小於密集板的降伏強度 56.3MPa。

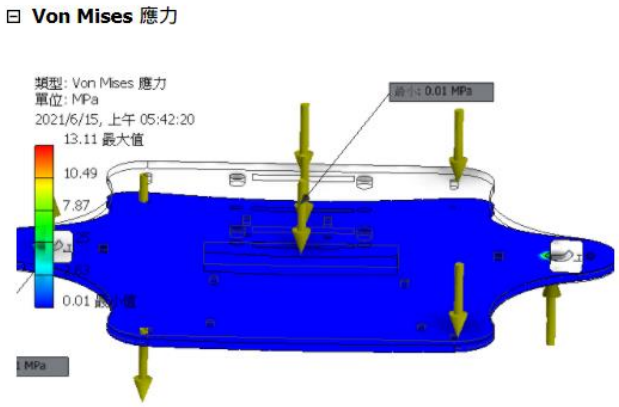


圖 4.4-10 頂板應力分析結果

由位移分析結果，我們可以觀察到位移最大的地方位於頂板的前端、中間，雖然可以觀察到集中的現象，但最大位移為 0.4908mm，在我們可以接受的範圍(5mm)之內。

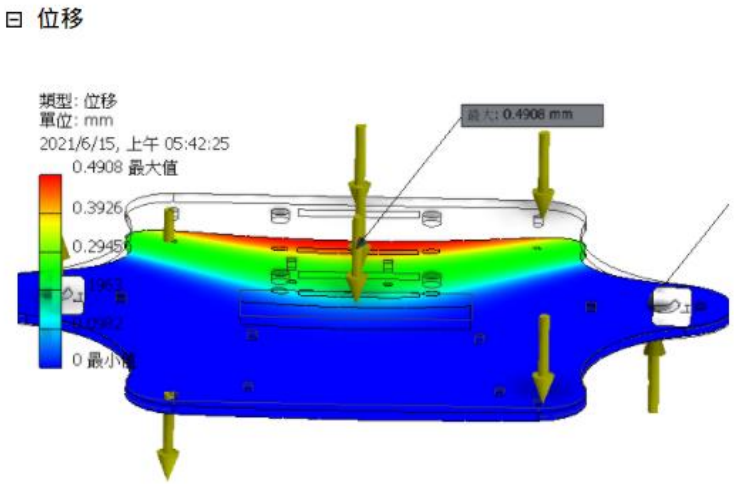


圖 4.4-11 頂板位移分析結果

(2) 底板

分析原因：針對底板作分析的原因為，底板承受底板上重物的重力和銅柱給予底板的向上支撐力，支撐力分布於底板外圍 4 個點，而重力集中於底板中間，這樣的力的分布可能會導致應力過大。

材料：材料使用密集板，因此使用 inventor 材料庫中最接近密集板性質的木材(樺木)，其性質如圖 4.4-7 所示。

負載：

針對底板，我們施加了一種向下的力以及一種向上的力。力的施加圖如圖 4.4-12 所示。

向下的力包括：

(a) 底板上重物的重力，底板上的重物有 18650 電池盒、兩個 18650 電池以及車手(鋁箔包飲料)，18650 電池盒和電池總重約 100g、車手重約 300g，總重約 400g。

向上的力包括：

(a) 銅柱施予底板的向上力:4 根銅柱施予底板向上的力以支撐重物重力，換算後每個力為 0.98N。

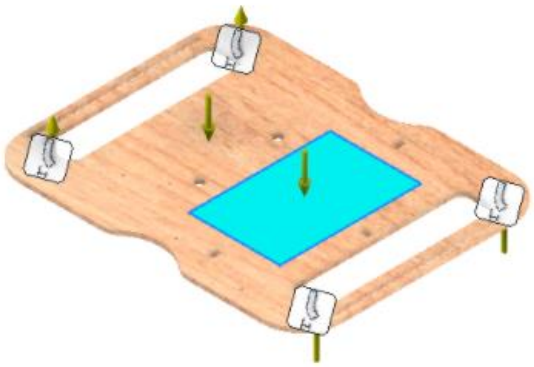


圖 4.4-12 底板施力配置圖

固定約束：底板的固定約束為銅柱所固定的 4 個點，如圖 4.4-13 所示。

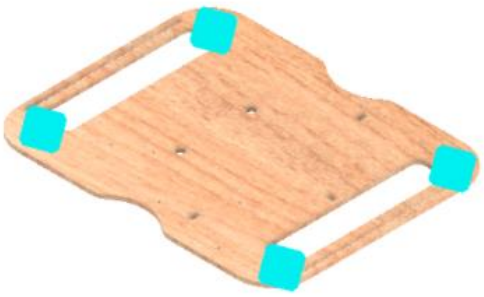


圖 4.4-13 底板約束配置圖

模擬結果：

模擬結果如圖 4.4-14、圖 4.4-15 所示。由 Von Mises 應力分析結果，我們可以發現底板所受應力皆在安全範圍內，最大值為 11.62MPa，較頂板 Von Mises 應力最大值 0.01MPa 高出許多，但仍小於密集板的降伏強度 56.3MPa，因此我們判斷底板發生應力過大情形的機率不高。

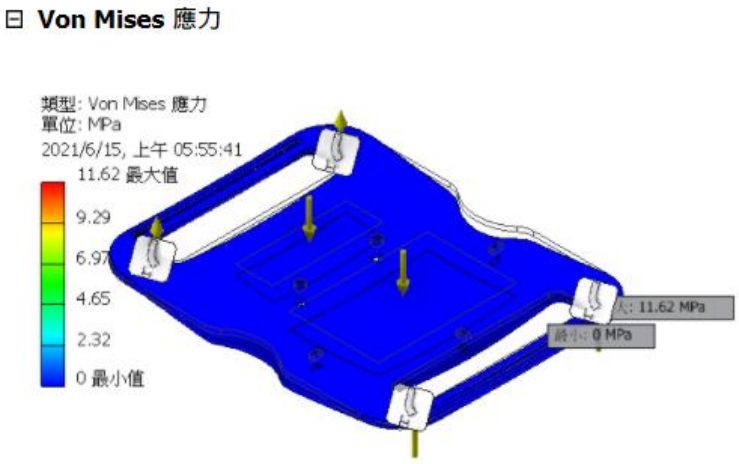


圖 4.4-14 底板應力分析結果

由位移分析結果，我們可以觀察到位移最大的地方位於底板中間，車手(鋁箔包飲料)放置處，雖然可以觀察到集中的現象，但最大位移為 0.07101mm 遠小於頂板形變量 0.4908mm，也在我們可以接受的範圍(5mm)之內。

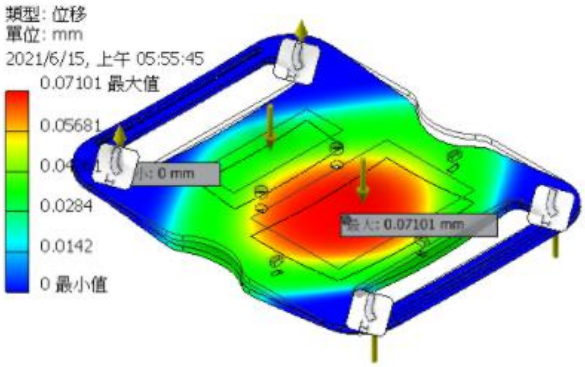


圖 4.4-15 底板位移分析結果

(3) 馬達和輪子間連接桿件

分析原因：輪子支撐車體的重量，而輪子和馬達的連接桿件及承受了該支撐力，又這個支撐件的形狀為細長型，若應力過大產生形變，很可能會因為影響到馬達和輪子間的關係，使得我們無法精準、穩定地操縱輪子。

材料：此桿件使用 3D 列印，因此分析時使用 inventor 材料庫中的 ABS 塑膠，其性質如圖 4.4-16 所示。

名稱	PC/ABS 塑膠	
一般	質量密度	0.357273 g/cm^3
	降伏強度	54.4 MPa
	極限抗拉強度	54.1 MPa
應力	楊氏模數	2.78 GPa
	蒲松氏比	0.4 ul
	剪力模數	0.992857 GPa
零件名稱	wheel-connect.ipt	

圖 4.4-16 桿件材料表

負載：

針對此桿件，我們設定車子在靜止的狀態下，所受到的力只有馬達施加的向下力以及輪子中心施加的向上力。力的施加圖如圖 4.4-17、圖 4.4-18 所示。

向上的力包括：

(a)輪子中心的向上力，每個輪子施予車體的向上支撐力為 7.4N，因此此支撐力分別施加於桿件和輪子間的兩個鎖點，每個點受到向上 3.7N 的力。

向下的力包括：

(a) 馬達施加的向下力，其力大小等同於輪子的支撐力 7.4N

圖 4.4-17 桿件施力圖 (1)

圖 4.4-18 桿件施力圖 (2)

固定約束：設定車子為靜止時，因此於桿件和頂板的連接處施加固定約束，如圖 4.4-19 所示。

圖 4.4-19 桿件約束圖

模擬結果：

模擬結果如圖 4.4-20、圖 4.4-21 所示。

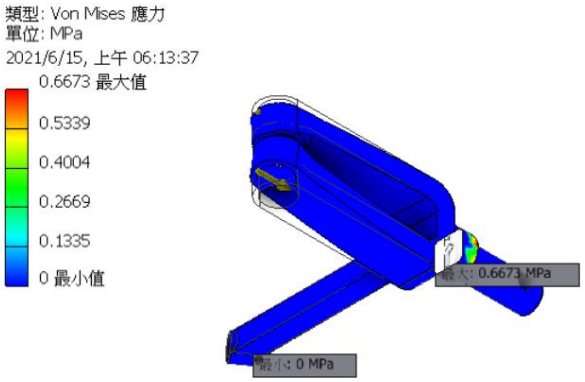


圖 4.4-20 桿件應力分析結果

由 Von Mises 應力分析結果，我們可以發現桿件所受應力皆在安全範圍內，最大值為 0.6673MPa，小於 ABS 塑膠的降伏強度 54.4MPa，因此我們判斷此桿件發生應力過大情形的機率不高。

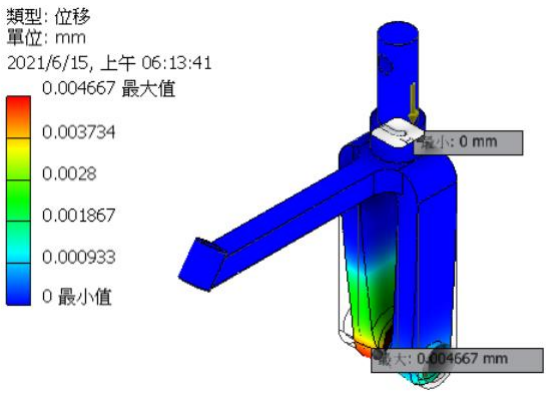


圖 4.4-21 桿件位移分析結果

由於此桿件會影響到車輪的控制，因此在這裡我們將位移的容許範圍訂為0.5mm。由位移分析結果，我們可以觀察到位移最大的地方位於桿件和輪子連接處，雖然可以觀察到集中的現象，但最大位移為 0.004667mm，在我們設定的可接受範圍(0.5mm)之內。

4.4.2. 質量與質心

實際在進行驗證時可以直接量測實體重量，而這次驗證中礙於無法實體測試，使用 CAD 估計重量與質心位置。

事先用軟體估測重量雖然較為不準確，但是在車輛研發上可以預先評估車輛性能並改變設計，不必待製造完成才做修正。方法將在 4.6 軟體建模提及。

1. 質量

在疫情前有測量各部件分開的質量，另外在 Inventor 中輸入材質後該零件質量也會列入計算。為了使整車重量估計更加準確，我們的總組合檔中放入的所有螺絲螺帽。取得質心方式為點選 iProperties 功能，iProperties 中選擇 Physical 可以看到預估的質量為 1.202kg = 1202g (圖 4.4-22、圖 4.4-23)。

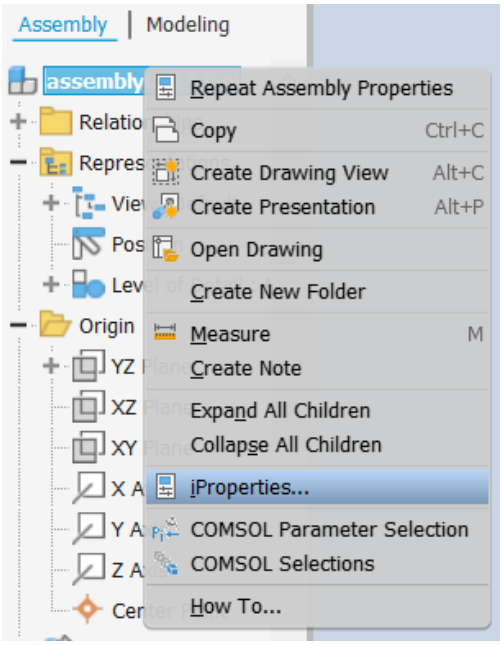


圖 4.4-22 點選 iProperties 功能

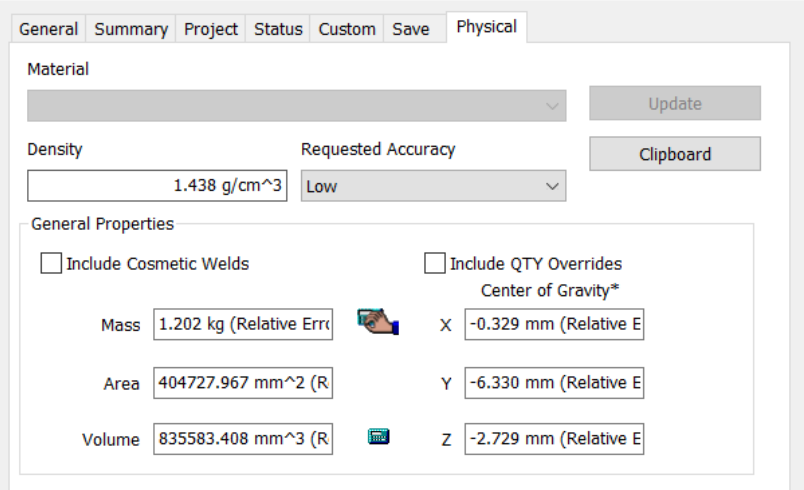


圖 4.4-23 iProperties 中選擇 Physical

2. 質心

使用 Inventor 建模，View 中點選 Center of Gravity 選項可以開啟質心位置。可以用於車輛性能的評估(圖 4.4-24)。

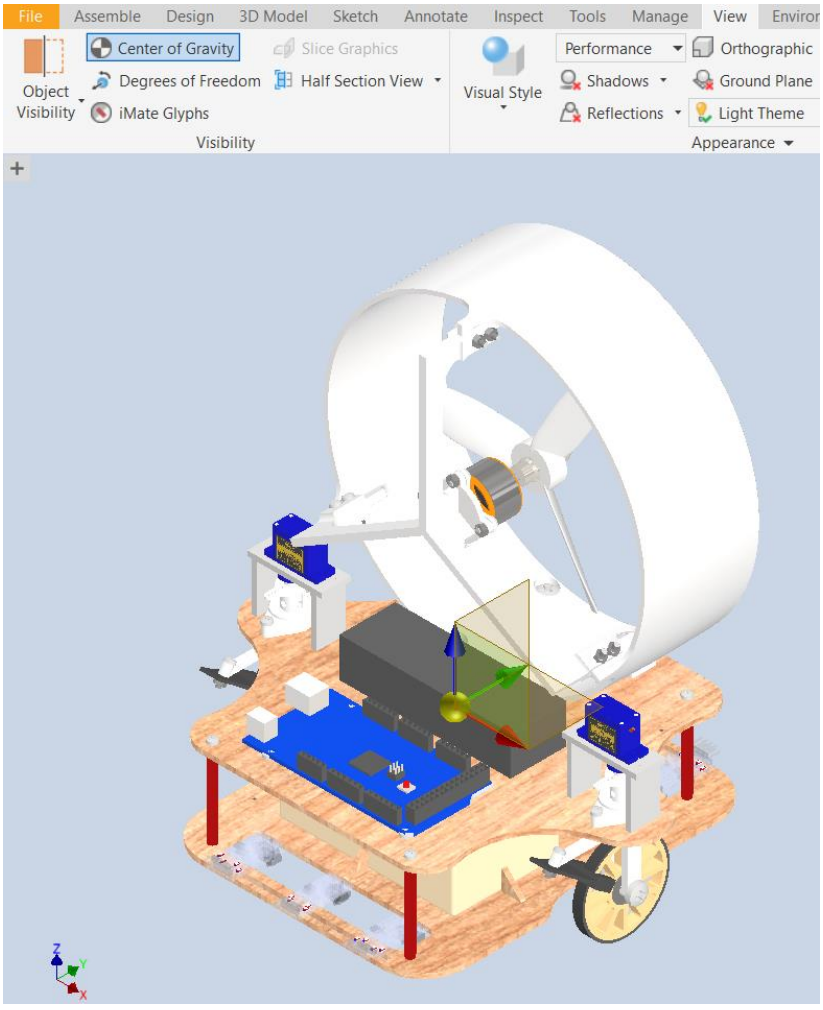


圖 4.4-24 View 中點選 Center of Gravity 選項

實體驗證時量測質量只需將車體放上磅秤即可，不過若需要取的質心位置的資料，仍須使用上述的方法。

4.4.3. 製造誤差、尺寸驗證

1. 密集板加工誤差

車體之頂板、底板、轉向馬達支架和煞車桿是由密集板製作。會造成誤差的因素有二：

(1) 雷射切割機造成之誤差：任何加工機具本身就帶有誤差，加上實作中心的機器長年大量切割，馬達、皮帶、雷射管必定會有正常損耗。

(2) 密集板造成之誤差：即便加工前有進行校正的動作，密集板本身的厚度不均會造成各處切割情形不一。如下圖(圖 4.4-25)，雷射射投有一凸透鏡，用以聚焦雷射光，打在切割材料上的雷射光為錐形。若密集板面高度有起伏，同一線段不同線寬也會不同，造成成品尺寸的誤差。

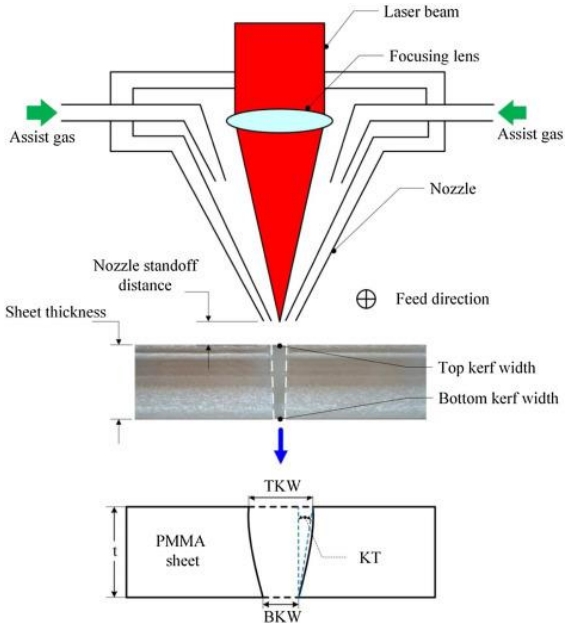


圖 4.4-25 雷射切割切口示意圖 [33]

2. 3D 列印件加工公差

風扇、風罩組合、煞車桿和轉向零件使用 PLA 材質、3D 列印方法加工。3D 列印的誤差來自機台的長損耗、未校準以及列印速度、溫度等設定。加上線材本身有寬度，不同的溫度濕度和線材品質皆會影響噴頭擠出的狀況，即便列印前機台都有調整設定也存在許多目前資源下無法控制的因素。

實體加工後，應測量成品之原尺寸和量測結果，並註明量測位置，可以加以分析不同設計之誤差值以及影響因素。由於期末時未成車，我們依據學長實際量測結果[32]估計雷射切割誤差值介於-0.3~0.95%之間，由於我們對於雷射切割加工件並沒有那麼高的精度要求，可以忽略加工誤差造成的影響。

4.5. 控制

4.5.1. 紅外線實驗

(1) 實驗儀器：TCRT5000 紅外線感測器、比賽場地。

(2) 實驗目的：得到每個 TCRT5000 在場地不同情境(黑線、凹洞、白底)下穩定的讀取數值範圍。

(3) 實驗說明：

試跑車子時，我們發現場地上有許多不平整的凹洞造成的陰影，如果只用黑線和白底設定 TCRT5000 讀取值的閾值來判斷黑線，在真實場地中會失準，因此，我們針對場地上的凹洞、黑線和白底場地皮，觀察每個 TCRT5000 在這三種場地上的讀值，以訂定出每個 TCRT5000 獨有的閾值。

(4) 實驗步驟：

(a) 選定一個 TCRT5000，將其對準場地黑線，紀錄其讀值，重複 5 次，如圖 4.5-1 所示。

(b) 對準場地凹洞，紀錄其讀值，重複 5 次

(c) 對準場地場地白皮，紀錄其讀值，重複 5 次

(d) 針對其他 TCRT5000，重複上述步驟

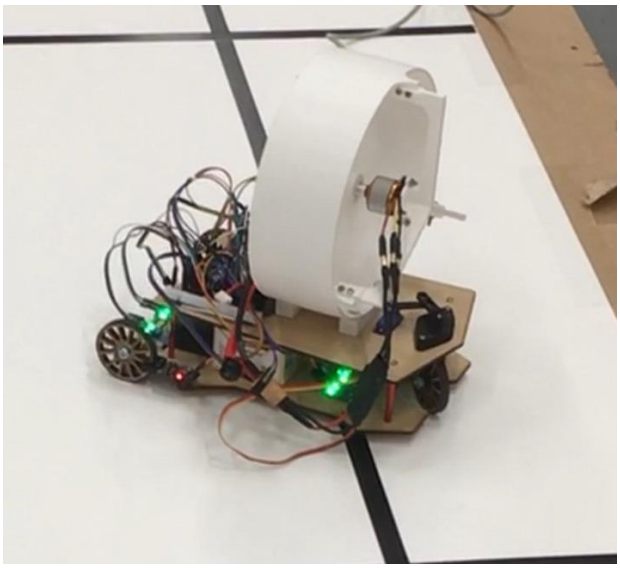


圖 4.5-1 實驗配置圖

(5) 實驗預期結果：

實驗預期結果如表 4.5-1 所示，我們希望能得到 TCRT5000 在黑線、凹洞和白底下穩定的讀值範圍，根據這些讀值範圍，我們就能訂定一個該 TCRT5000 的專屬閾值，讓之後在控制車子循跡或找線時，不會將凹洞誤判為黑線，導致車體沒有依照理想軌跡和策略移動。

表 4.5-1 實驗表格

TCRT5000編號	###		
次數	黑線讀值	凹洞讀值	白底讀值
1	###	###	###
2	###	###	###
3	###	###	###
4	###	###	###
5	###	###	###
平均			
標準差			

4.5.2. Encoder 精準度實驗

(1) 實驗儀器：風動車、encoder、比賽場地。

(2) 實驗目的：了解本組設計之編碼器(encoder)在直行及轉彎時之精準度。

(3) 實驗說明：

編碼器(encoder)是在輪子上設計各個相位角的細孔(本組以 30 度為單位)，再將計數器安裝於輪子上緣，計數器則採用紅外線收發器，使當接收器轉到細孔時讀值為 1，受遮蔽時(車框)讀值為 0，如此車輪旋轉下可計數風動車所走圈數，詳細原理詳見 3.5.2。

因此感測器為自行設計，故需了解其在風動車行走時其精準度之表現。而以下分為兩部分：

(a) 直行

將風動車置於一條直線上，其長度為 100 公分，執行程式使之直走，並同時記錄左右兩輪 encoder 之數值，計算 encoder 的距離。重複上述步驟 5 次，並計算相對誤差。此計算 encoder 的距離之公式為：

$$d_{encoder} = \frac{n \times 30}{360} \times (2\pi r)$$

其中 n 為 encoder 計數，r 為車輪半徑，而因本組 encoder 設計為 30 度為一單位，故計數一次車輪會行走$\frac{30}{360} = \frac{1}{12}$圈。

(b) 轉彎

因在實際測試上，本組之轉彎方式為於原地自旋一角度後直行，故以下按照此模式設計實驗。將風動車置於一條直線上，使車頭與車尾均在直線上(擺正)，在起始狀態時設定前輪轉角為 30 度，啟動煞車，並開啟風扇使之原地旋轉。旋轉一圈後停止，記錄左右兩輪 encoder 之數值，計算 encoder 的距離，並與實際所繞的圓周長進行比較。此計算 encoder 的距離之公式與(a)相同，實際行走距離則為以前後輪距離為半徑，繞行一圈所經過之圓周長。

(4) 實驗數據：

表 4.5-2 實驗表格

直行				轉彎			
車輪半徑 r [cm]				車輪半徑 r [cm]		前後輪距離 R [cm]	
實際距離 [cm]	100			實際距離 [cm]			
#	encoder 計數 n	計算距離 d [cm]	誤差	#	encoder 計數 n	計算距離 d [cm]	誤差
1	###	###	###	1	###	###	###
2	###	###	###	2	###	###	###
3	###	###	###	3	###	###	###
4	###	###	###	4	###	###	###
5	###	###	###	5	###	###	###
average d [cm]	###	std.	###	average d [cm]	###	std.	###

(5) 實驗預期結果

首先，在直線行走上，若風動車為等速，encoder 計數需與所經過距離成正比，繪製成圖表後應為一斜直線(計數與所經過距離)。因此，在我們的實驗中，所記錄之 5 個計數的誤差應不會相差甚遠。另外，於期中測試前有先行測試 encoder 之精準度，也發現所走圈數與實際距離大致符合比例關係。

4.6. 軟體建模

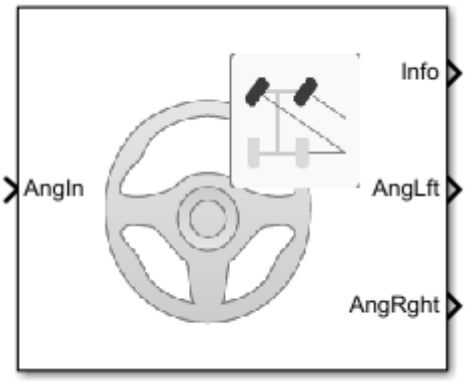
用軟體估測的結果雖然較為不準確，但是在研發上可以預先評估車輛性能並改變設計，不必等到製造都完才做修正。若有實際製作出車輛，可以將模擬結果比對實際動態，修正模型，在重新設計時可以預估趨勢，優化車體參數後再開始時實作。

值得一提的是這堂課的歷年來的題目類似，都可以將系統分為車體和控制兩大區塊。實作過程中經常發生的問題是負責控制的組員要等到車體成型後才開始調整程式。因此只要整個研發過程中稍有差錯，就會發生沒有足夠時間優改程式，接近測試日期才熬夜調整的情況。

我們進行軟體建模是希望能夠在確定大致參數時就可以開始撰寫控制程式，不僅能優化車體設計，也有充裕的時間能夠進行程式設計。

4.6.1. Simulink 資料研究

我們使用 Simulink 進行建模，並搭配 Vehicle Dynamics Blockset。[34] 以下為使用的車輛動態模組方塊：



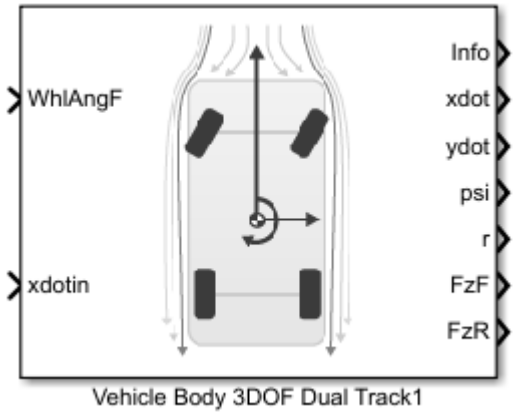
Kinematic Steering

Input:
AngIn - Steering angle, *δin*, in rad

Output:
AngLft - Left wheel angle, *δL*, in rad
AngRght - Right wheel angle, *δR*, in rad

預設車輛轉向為齒輪-齒條機構，可以透過輸入轉向幾何或著直接左輪和右輪關係之轉向曲線曲的左右輪分別輸出轉向角度。

期末車輛的轉向機構為兩輪平行轉向，由馬達直接驅動，因此我們選擇轉向模式為為平行轉向，設定 steering ratio(方向盤轉角/輪胎轉角)為 1。

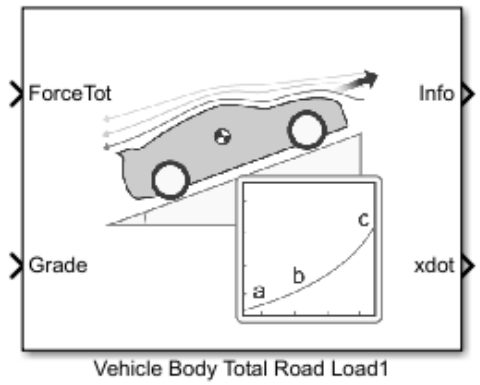


3DoF Vehicle Body

Input:
WhlAngF - Front wheel steering angles, δF, in rad
xdotin - Vehicle CG velocity along the vehicle-fixed x-axis, in m/s

Output:
xdot - Vehicle CG velocity along the vehicle-fixed *x*-axis, in m/s.
ydot - Vehicle CG velocity along the vehicle-fixed *y*-axis, in m/s.

這個三自由度的車體方塊可以計算車身縱向、橫向以及角度的速度，由於只能計算三個自由度，適用於沒有明顯垂直、翻滾或俯仰位移的車輛。

 Vehicle Body Road Load	Input: ForceTot - Tractive input force, Ftotal, in N Grade - Road grade angle, Θ, in deg Output: xdot - Vehicle total longitudinal velocity , in m/s
我們使用這個方塊模擬上坡速度。輸入推力及斜坡頃角並設定車重、磨擦係數等參數，會得到車子沿車頭方向上坡速度。	

 X-Y Graph	Input: Port1 - X-axis values Port2 - Y-axis values Output: Visual Plot of X-Y
此方塊可以讀取兩數值繪製一 X-Y 圖，在車輛模型中可用以繪製車體路徑。模型中，輸入為車子質心之 X、Y 方向速度積分即 X、Y 方向位移，我們藉由觀察輸出路徑驗證程式之輸入轉速、轉向方向是否能得到合理的結果。	

4.6.2. Hardware-in-the-loop 模型與驗證

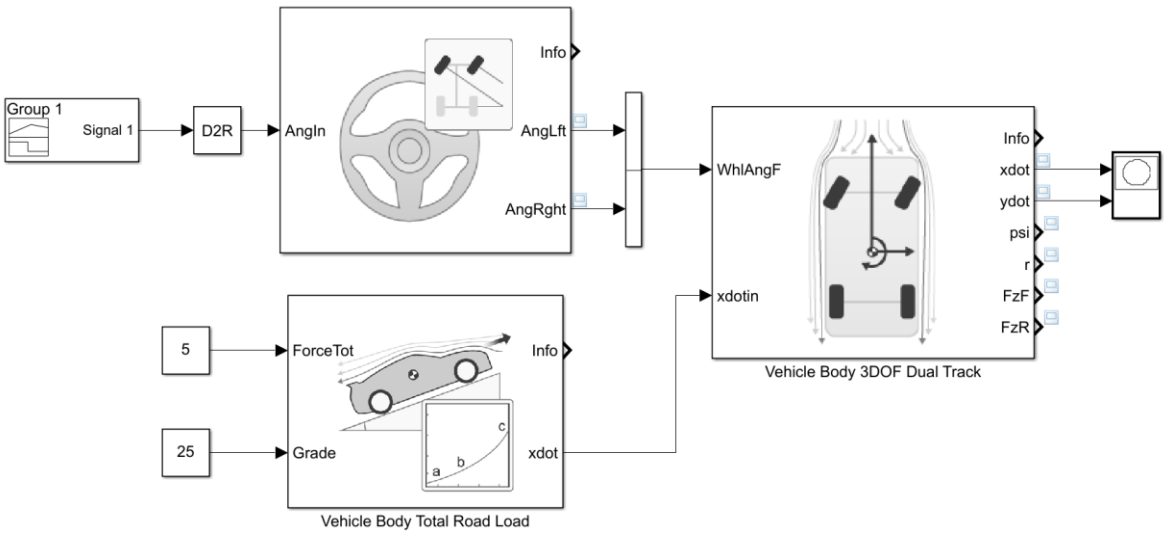


圖 4.6-1、依據期末爬坡測試所建立之 Simulink 模型

最終完成的 Simulink 模型如圖 4.6-1，我們使用最高傾角 25 度，另外輸入轉向訊號(圖 4.6-2)和風扇推力，可以得到車輛路徑如圖 4.6-3。

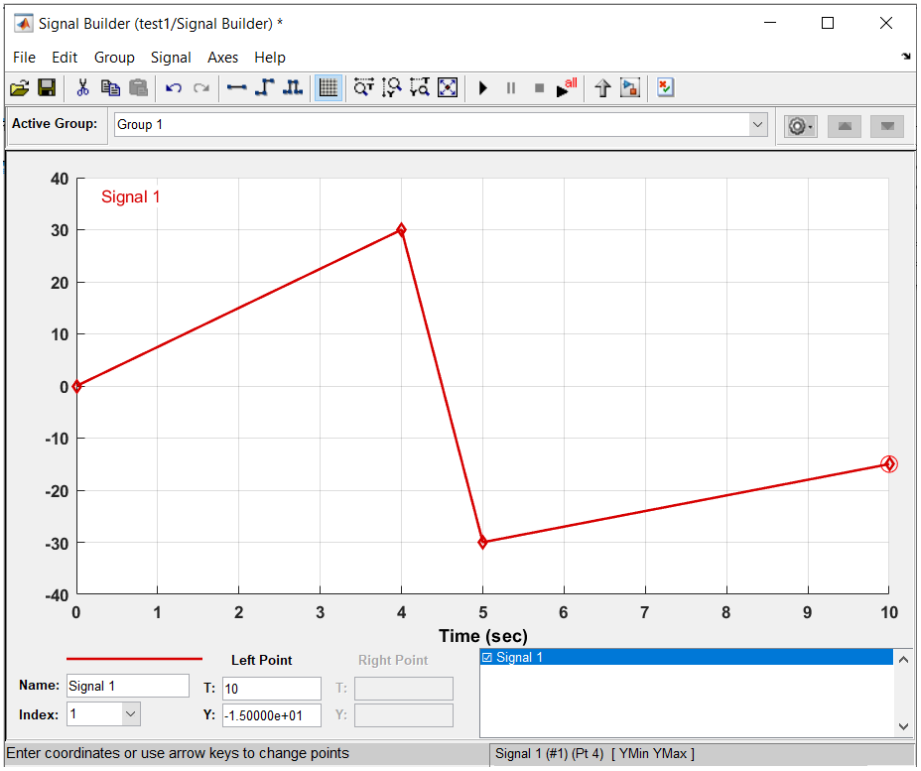


圖 4.6-2、輸入轉向角，數值為正、負為右、左轉角度

輸入的轉向角度 0~4 秒時為右轉零到 30 度，接著 5~10 秒左轉從 30 至 20 度。對應的路線下圖 4.6-3，一開始轉向角較小時，車子先直行一段距離，接者轉換了兩次方向，大致符合行徑的趨勢。

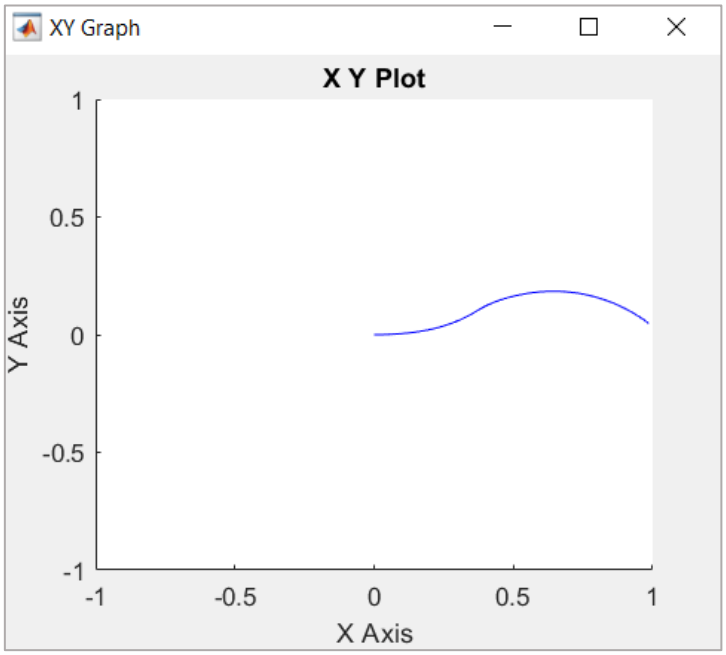


圖 4.6-3、輸出路徑圖

此模型中風扇推力是用預估的定值 5N，實際驗證時可依據表 4.1-1 找到程式控制的轉速對應的推力作為這部分的輸入，模擬會更加貼近實際的樣貌。礙於無法成車，我們無法驗證出路徑是否準確，因此模擬結果不足以作為我們修改。

5. 工作進度與分工

5.1. 團隊合作及分工

5.1.1. 團隊目標

(1) 做出車車：做出能完成比賽要求、車體組裝容易的車。
(2) 和樂融融：組員之間互動融洽，互相給予支持和陪伴，透過工作結束後的聚餐營造組內歸屬感。
(3) 學習新知：透過完成車子，複習這幾年在機械系學到的知識外，學習新的知識和技能。

5.1.2. 合作準則

本機械工程實務專題團隊之宗旨為「團隊氣氛與專題成果相等重要」，以下說明我們團隊運作規則及如何維持團隊氣氛。

(1) 本團隊鼓勵多元想法及意見。在每次的討論及專題製作時，大家應不吝嗇發言，說出個人的想法及建言，其他人也應重視他人意見並認真思考其可行性。藉不同的想法使此專題更加完善，也增加團隊合作之精神。

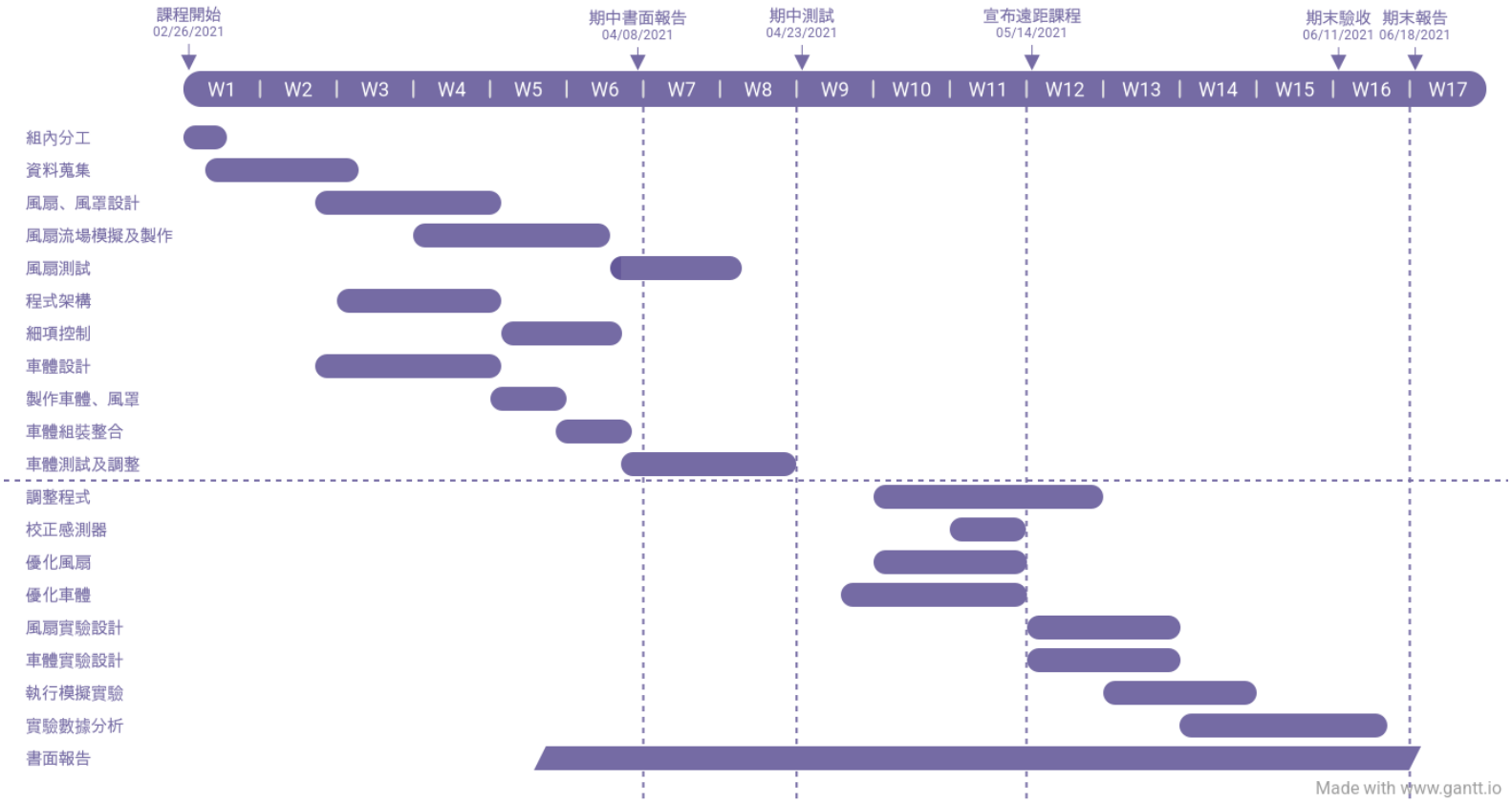
(2) 討論時以鼓勵取代批評，組員若提出的想法有改進的空間，其他組員應以稱讚之正向語氣回應並提出具體修改之建議，並藉此讓大家討論使此想法更加完善。禁止以責備或謾罵之語氣與他人應答。

(3) 本團隊有製作每人專屬的個人籤，若專題內容有工作分配及選擇問題，則以團隊以外的人抽籤決定。而本次未抽到的人負責下一次的進度，藉輪流負責不同的工作內容以分攤每人的工作量。

(4) 本團隊每周固定討論時間為週一 19：00 後，地點挑選不吵鬧且有工具及零件可即時測試的空間，討論內容以上週進度成果報告和問題討論為主。而為增加團隊向心力，在每周討論後皆會全組合照紀念，並一同參與會後聚餐(宵夜)以凝聚團隊感情。

(5) 專題成果固然重要，但若團隊氣氛不佳，則失去此專題合作之意義及快樂性。故若有人遭遇問題時必須適時提出讓大家了解，其他人也須幫助有困難的組員，藉相互幫助以發揮團隊合作之精神，也讓此團隊更加和諧。

5.2. 甘特圖



5.3. 成員貢獻

5.3.1. 分工方式

組員	工作分配
楊宜瑄 B07502011	車體、加工
潘宣蓉 B07502037	車體、控制
吳宥廷 B07502039	風扇(分析)、控制
陳旻彥 B07502069	車體、加工
賴昭蓉 B07502165	風扇(設計)、控制

5.3.2. 互評方式

在這次的團隊合作中，我們討論並列舉出了 4 項我們認為在團隊合作中極其重要的核心價值，根據每個項目，分別具體列出各個評分的標準，如下表所示：

指標+評分(%)	5 分代表	4 分代表	3 分代表	2 分代表	1 分代表
溝通能力(25%)	主動提出問題，並積極與成員討論	不會主動提出問題，但會認真給予他人回饋	會提出問題，但不會試圖幫別人解決問題	會提出問題，但沒有認真聽別人的回饋	對於遇到的問題不會提出也不會給予他人回饋
工作品質(25%)	提供高品質成果，重複檢查細節並修正	專注於任務上，大部分時間可以順利完成	大致理解自己分配到的任務，能於死線前完成	多數時間需要組員反覆提醒後才完成任務	提醒後仍完全無法完成任務，這個人不可靠，該拆組囉:)
意見發想(25%)	時常能主動提出多元且有效的解決方案	偶爾會主動提出有效的解決方案	較少會主動提出解決方案，但會和他人一起討論	不會主動想到解決方案，但會和他人一起討論	完全不思考，超級被動
參與度(25%)	能出席每次的會議，並參與每個討論	並非出席每次的會議，但出席時仍能參與每個討論	並非出席每次的會議，出席時仍參與大部分的討論	並非出席每次的會議，出席時也鮮少參與討論	鮮少出席會議，出席也不會參與討論

5.4. 會議記錄

會議記錄一

會議時間：2021.03.08

會議地點：永齡

出席成員：楊宜瑄、潘宣蓉、吳宥廷、陳旻彥、賴昭蓉

請假成員：無

一、上次會議追蹤事項：

1. 無

二、本次會議內容：

1. 分析比賽需求
   (1) 討論內容：根據比賽規則，粗略分析我們需要做到哪些事情？
   (2) 結論及預期目標：要做出一台可以被風扇驅動的車體、上面有很多感測器可以循跡，裡面包含完整的控制電路系統。
2. 怎麼分工
   (1) 討論內容：工作內容可以分為哪幾項？要怎麼分擔工作最妥當？
   (2) 結論及預期目標：根據機械設計原理的經驗，工作項目主要有四大項：

負責製作車體、負責寫程式和電路、負責加工，而這次的主題又另外包含風扇分析和製作。但分成四個工作又不好給五人團隊分工，所以決定改成混和式分工制度，每個工作項目都設計需求人數，然後大家排志願序分發！如果發生填同工作且同志願序的人數超過限制，就抽籤決定。

- 大家的志願序：

onion: 1. 風扇 2. 控制 3. 車體 4. 加工

yoyo: 1. 風扇 2. 控制 3. 車體 4. 加工

潘：1. 控制 2. 風扇 3. 加工 4. 車體

麵：1.車體 2.風扇 3.加工 4.控制

瑄：1.車體 2.加工 3.風扇 4.控制

- 最後的分組結果如下

	車體	風扇	控制&電路	加工組
人數及負責人	3: 麵、瑄、潘	2: 蔥、宥	3: 蔥、宥、潘	2: 麵、瑄
工作內容	轉向 (機構) 配置配重 風罩 輪胎 煞車	葉片	煞車控制 葉片控制 感測器	負責 3D 列印或雷切零件

3. 各組大概討論三週內進度

下週 車體：煞車 轉向形式 零件選購 材質選擇 風扇：資料蒐集 零件、 控制：暫無 加工：暫無
再下周: 車體：開畫 風扇：下週決定 控制：1.encoder 要選，配線，要實驗他的解析度，一些 test code 2.紅外線*3 前後，一些 test code 加工：臨時機動

三、臨時動議：

1. 無

四、下次會議時間：

2021.03.15 19:00~22:00

會議記錄二

會議時間：2021.03.15

會議地點：永齡

出席成員：楊宜瑄、潘宣蓉、吳宥廷、陳旻彥、賴昭蓉

請假成員：無

一、上次會議追蹤事項：

1. 這週要努力完成上週計畫要做的事情

二、本次會議內容：

1. 分析比賽需求

(1) 討論內容：本週各自進度

(2) 結論及預期目標：

控制：

1. IR Sensor (有挑幾個好的)
2. encoder 使用方式 (安裝方式、形狀(90 條)、計數器選擇)
3. 做出簡易測試平台(inc. IR sensor*5, Arduino Mega, 麵包坂)
4. 排出車車整體跑法

next: 確認 pseudo code (下次 meeting 前完成)、畫出 encoder

風扇：

next: 選用何種風扇、決定後續要用的分析及實驗

車體：

1. 輪子 v1
2. 車體初始架構 (現在只畫完底盤)

next: 把車架構畫完

三、臨時動議：

1. 無

四、下次會議時間：

2021.03.22 19:00~22:00

會議記錄三

會議時間：2021.03.24 10:00~12:00

會議地點：永齡實作中心

出席成員：楊宜瑄、潘宣蓉、陳旻彥(車體組開會)

請假成員：無

一、上次會議追蹤事項：

1. 無

二、本次會議內容：

1. 決定煞車及轉向使用馬達

(1) 討論內容：以輕量為原則，計算扭力是否足夠

(2) 結論及預期目標：皆使用 SG-90 馬達

2. 軸承使用

(1) 討論內容：是否都需要使用軸承?

(2) 結論及預期目標：輪鼓部分，轉向使用軸承

3. 煞車方案

(1) 討論內容：夾輪胎煞車/轉軸連接齒輪煞車/點地煞車

(2) 結論及預期目標：使用點地煞車

4. 輪胎皮選擇

(1) 討論內容：使用橡皮擦?橡膠皮? 需求是否為摩擦係數越大越好?

(2) 結論及預期目標：摩擦係數越大越好，使用 3M 煞車皮

三、臨時動議：

1. 午餐吃什麼

四、下次會議時間：

另行決定

會議記錄四

會議時間：2021.03.29 19:00~22:00

會議地點：永齡實作中心

出席成員：楊宜瑄、潘宣蓉、吳宥廷、陳旻彥、賴昭蓉

請假成員：無

一、上次會議追蹤事項：

無

二、本次會議內容：

1. 本週進度報告

(1) 討論內容：控制和風扇本週的設計進度

(2) 結論及預期目標：

控制：

1.各個 state 架構，研究 PID,millis(),encoder 寫法和用法

風扇：

1.研究 javaprop, javafoil 的流程，用 NACA 2414 為模型，產出 javaprop 最佳化後的三葉風扇。(他會最佳化所有 twist、攻角、節面大小)

2.因為考量到，翼型理論中上下翼型必須不對稱才能產生升力，但本次的 project 的風扇要能反轉，所以決定改用對稱之橢圓形斷面混成，自己繪製 blade，再使用 ansys 分析推力。但這種方法可能有風險，因為 try and error 的成分太高，下週再實驗看看！

車體：

1.SG90 CAD 圖

2.加工&摩擦力實驗

下週進度:組裝

2. 推力實驗

(1) 討論內容：下周風扇推力實驗準備

(2) 結論及預期目標：

1. yoyo: solving coding、電路設計、搞懂 ANSYS、讀好 element 理論

2. 蔥: 風扇 inventor、實驗底座 inventor、讀好 element 理論

需要的東西: 印好的風扇(蔥畫)、寫好的 code(yoyo 寫)、底座(蔥畫)、電子秤(永齡有)、電池(有)、電變(有)、開關(有)、arduino(有)

三、臨時動議：

1.車體優化方案

(1) 討論內容：修正轉向

(2) 結論及預期目標：

改善轉向機構，暫不考慮阿克曼

四、下次會議時間：

2021.3.29 19:00~22:00

會議記錄五

會議時間：2021.04.23 13:30~14:30

會議地點：工學院綜合大樓 429 室 (機械系學會)

出席成員：楊宜瑄、潘宣蓉、吳宥廷、陳旻彥、賴昭蓉

請假成員：無

一、上次會議追蹤事項：

1. 無

二、本次會議內容：

1. 鋰電池相關議題

(1) 討論內容：本組所使用的鋰電池為學長姊使用後所留下，故需考慮是否要買新電池及新的充電器

(2) 結論及預期目標：於周末時找尋新電池以及新的充電器。

2. 風洞車控制及程式

(1) 討論內容：於測試場地初步測試時發現起步時推力不足，且循跡不穩定，後面的步驟及程式碼仍要測試及修正

(2) 結論及預期目標：下次測試時就以下順序分別測試單一功能：轉彎、循跡、換線、停止後風扇反轉。另外，需選擇所用風扇。

3. 風洞車之風罩問題

(1) 討論內容：新製作的風罩之孔徑(6 mm)與車底板孔徑(3 mm)不符。

(2) 結論及預期目標：擇日再將車底板利用雷射切割機重新加工，使孔徑相符，另外需取得 M6 螺絲。

三、臨時動議：

1. 無

四、下次會議時間：

2021.04.16 19:00~22:00

會議記錄六

會議時間：2021.05.24 19:00~21:00

會議地點：Google Meet

出席成員：楊宜瑄、潘宣蓉、吳宥廷、陳旻彥、賴昭蓉

請假成員：無

一、上次會議追蹤事項：

1. 已修正車體頂板之孔徑，可成功與風罩組合
2. 車體之控制及程式礙於實作中心關閉，可能無法再測試修正

二、本次會議內容：

1. 期末報告之大綱

(1) 討論內容：由於本學期之課程改為遠距，車體製作以及實驗都受到影響，於會議上共同討論各項實驗之設計以及期末報告整體內容。

(2) 結論及預期目標：訂出報告之大綱及子標題。

2. 期末報告之分工

(1) 討論內容：訂定大綱及子標題後，考量各組員在風動車期中作品中負責的部分，將相關項目分配給組員撰寫內容，並在共編文件中標上專屬顏色。

(2) 結論及預期目標：將報告分配完畢，並約定下周討論時間以及完成報告第三章內容。

三、臨時動議：

1. 本學期風動車之命名：小飛象章魚。

靈感取自 Google 共編文件上的匿名動物。

四、下次會議時間：

2021.05.31 19:00~22:00

會議記錄七

會議時間：2021.05.31 19:00~22:00

會議地點：Google Meet

出席成員：楊宜瑄、潘宣蓉、吳宥廷、陳旻彥、賴昭蓉

請假成員：無

一、上次會議追蹤事項：

無

二、本次會議內容：

1. 期末報告進度
   (1) 討論內容：報告本週各組員完成之期末報告進度
   (2) 結論及預期目標：已完成第三章報告進度，主要剩下第四章的分析和實驗規劃
2. 實驗規劃討論
   (1) 討論內容：實驗規劃要求和形式
   (2) 結論及預期目標：訂定出實驗規劃報告模板，包括目的、說明、實驗內容、結果數據分析和結論，請各實驗規劃負責人依照模板撰寫

三、臨時動議：

1. 期末報告封面繪製
   (1) 討論內容：是否要自己繪製期末報告封面
   (2) 結論及預期目標：由賴昭蓉負責繪製期末報告封面，主題是我們的車子--小飛象章魚

四、下次會議時間：

2021.06.09 19:00~22:00

6. 課程回饋

(1) 吳宥廷

修習完這門課，首先非常謝謝組員的付出，大家都很辛苦！而對於這門課的建議，雖然老師編排課程很辛苦(謝謝教授及助教的付出)，每一年也會有所調整，但我認為這門課依然存在許多問題，如經費及資源、相關安全、所付出時間與獲得不成比例、場地公平性等。或許有些人認為可以藉此訓練實作及獲得合作溝通的能力，但我個人認為開設一門課以獲取相關能力，此門課卻設定讓一半機械系的學生於測試前一天在永齡實作中心通宵，做法似乎不太恰當。因此我認為可以降低整體課程負擔，規則的公平性需要修正，安全性也需注意。如此，或許藉由此方法，除了可以在適度壓力及所花費時間成本下獲得同等的相關技巧，對老師及學生的負擔也不會這麼重。再次謝謝老師、助教及組員的付出。

(2) 賴昭蓉

終於完成機工實務這門課，果然如學長姐所說是門十學分的重課！期中測試以前就深刻感受到爆肝程度，過程中各種心酸血淚史。回想起來，大學這三年學到的材力、流力、自控知識都不知不覺變成自己的工具，幫助我完成這門課程，當然更重要的還是那些一起做事、一起抱怨、一起在永齡通宵吃宵夜、一起遠距開會兼聊天的組員，還有出借各種工具充電器、在我們燒壞東西跑來安慰我們、贈送乖乖吊飾的朋友們。

在這門實作課裡，我負責的工作是設計扇葉還有程式設計。有些東西我覺得挺有意義的，像是在設計風扇還有做流體力學相關的量測實驗的過程，重新複習了很多邊界層、翼型理論的知識，對流力有更深的體悟，很喜歡這部分的課程安排。但有些東西我覺得不太合理，像題目控制設計還有場地布置。首先，將「循跡」和「計畫性行走」的區域混和穿插，我覺得挺矛盾的，概念上似乎是要同學做出一台部分自動化、部分預先歸劃的車，那何不全程寫成預先規劃？導致很多組別計秒寫死拿到滿分，失去這門課學習控制的內涵。更不好的是場地在劃分計分區域時，居然使用和循跡線相同的黑線，導致循跡頻頻失敗，每走二十公分就要被場地線干涉一次，簡直讓人崩潰，且我們在寫控制系統時，因為不希望寫死，設計了很多以黑線作為判斷依據的程式碼，像是轉彎擺正時、搭配 encoder 及是否感測到黑線，結果規劃得看似周延，放上充滿橫向黑線的場地變成一片混亂。此外，還有場地壓克力板突起導致地勢起伏的問題、場地上的凹洞貼了白色貼紙隱藏的迷你凹透鏡效應導致紅外線失真的問題，雖然這些問題都可以被視作需要突破難關的考核，我們確實也一直努力在突破，不眠不休的花了兩個晚上，但回想起來有點不甘心，努力的調整各種參數只是為了配合不太合理的場地設計，結果寫死的組別反而拿到最高分，想努力寫好控制系統的卻遇到重重阻礙。

當然，場地瑕疵導致控制難度太高是一回事，我們也記取教訓打算在期末測試前，事先瞭解整個場地實際狀況，針對場地會有的問題提前設計硬體機構後再設計程式，例如說發現場地會不明突起，就把車體做重讓他更穩，發現轉彎會被場地線搞，就把轉向做成平移轉彎不改變車頭方向，但萬萬沒想到疫情直接讓我們失去期末測試的機會～

(3) 潘宣蓉

這門課的 loading 實在是不可否認的重，在這學期當中也發生了很多變數，雖然最後取消了期末測試有點可惜，但在這大半學期裡能夠每個禮拜好幾天跟組員熬到很晚，有時候甚至一起吃完晚餐、宵夜、早餐，其實也是很特別的體驗!且大家在遇到困難後，一起嘗試得到最佳的解決方案，很喜歡這種明確分工、信任組員可以好好完成，又一起合作解決問題的信任和歸屬感，很慶幸有這組的大家一起面對很多困難、一起爆肝通宵!

至於在這門課程中遇到的困難，可以理解許多困難都是在實作中必定會遇到的，像是風扇推力和分析時有誤差或是車體零件干涉等等，學習解決這些問題是我們在這門實作課程中應該要學會的事；但是有一些困難，像是場地當中有一些除了黑線以外的小凹洞，會影響到紅外線感測器判讀，或是場地的厚紙板連接出會翹起來，形成一個小階梯，且場地會隨著時間的推移產生各式各樣的變化，要解決這些困難需要花很多時間，且場地隨時間的變動性會讓我們不確定自己的車在未來能否順利完成任務，只能不斷測試車子到比賽前的最後一刻，當然這其中也得到了許多應對困難的經驗，但用鐵架場地的組別就可以大大避免這個問題了，因此希望這門課程之後能夠考慮比賽場地的公平性和穩定性。

最後還是要感謝這門課，畢竟這樣的實作經驗依然是難能可貴的，也能感受到助教盡力協助同學，教授很仔細地看我們的報告並給予回饋，機械系同學互相幫助、互相鼓勵，還有組員一起崩潰、一起努力，這些都是在這門課當中讓我們願意繼續努力的動力，很感謝大家，也很謝謝這門課帶給我的知識和經驗。

(4) 楊宜瑄

我認為機工實務是很難得的學習機會，和其他機械系課程不一樣的是有空間、資源和目標相同的團隊一起合作，也有學長姊們豐富的經驗可以參考。我也感受到這門課在訓練我們考量完整系統配置的能力，除了要針對各次系統的需求進行設計，也要考慮系統之間的配合與取捨。不過這門課有些部分我認為可以改進：

首先是加工方式不夠成熟。大部分組別使用的是雷切密集板和 3D 列印塑膠的方式加工，這些加工方式很難運用機械設計的零件選用、材料力學的應力分析、公差配合等基礎能力。

第二點是專題題目的設計使得硬體本身不會有失效的情形。同學們要達成測試要求最好的方式是快速的製造出硬體，保留大量時間調整控制程式，而需要增減感測元件時才會修改車體。有時候為了趕上進度就隨便的安排配置，根本無法好好優化外觀和機構，也忽略應有的研發流程，形成"調參大會"的亂象。

我相信這門課本意是讓學生從設計發想開始，製造過程中同步進行驗證，經過修改、微調後完成成品。我想有限的資源下可能很難設計更全面、讓大家都滿意的的課程，但希望可以針對這些部分做修改，讓學弟妹有更好的學習環境。

(5) 陳旻彥

終於到了這門課的尾聲，學期前雖然有所聽聞、也經歷了上學期的機設，但到了學期中才真正體會到了這門課的負擔。感謝所有組員這學期的努力，不管是一開始的設計發想，還是到動手開始做車，小組的氛圍一直都很不錯，大家也都能想辦法達成任務。很高興有這些厲害的組員，從一起吃宵夜到一起看日出，可惜無法一起挑戰期末測試。

再來也很感謝老師們跟助教們對於這堂課付出的心力，相信在學期開始前老師們已進行過數次討論，學期間也可以看見助教在布置場地，畢竟要整合機械系五大組所學也真的不容易。但期中測試前一周的晚上在永齡看到同學們有的測試成功後歡呼，有人因為車子遲遲走不好而愁眉苦臉，這本來就是很正常的，直到我看了一眼時鐘。

無論如何，這門課都到了尾聲，希望大家之後都能保持健康、好好睡覺！

7. 參考文獻

[1] Aerodynamics of Wind Turbines, Emrah Kulunk, New Mexico Institute of Mining and Technology, USA, https://cdn.intechopen.com/pdfs/16241/InTech-Aerodynamics_of_wind_turbines.pdf

[2] https://web.mit.edu/16.unified/www/FALL/thermodynamics/notes/node86.html

[3] https://web.mit.edu/16.unified/www/FALL/thermodynamics/notes/node86.html

[4]「NACA 翼型」，維基百科，自由的百科全書，https://zh.wikipedia.org/wiki/NACA%E7%BF%BC%E5%9E%8B

[5] https://www.propellerpages.com/?c=articles&f=2006-02-17_Number_of_blades

[6] https://nasa.fandom.com/wiki/NACA_airfoil?file=NACA0015_a.png

[7] Frank M.White, Fluid mechanics 8th

[8] https://en.wikipedia.org/wiki/Lift_coefficient

[9]https://cycling.biji.co/index.php?q=news&act=info&id=94354

[10] https://motorsport.tech/formula-1/formula-one-brakes-explained

[11] https://www.youtube.com/watch?v=pqcWR9qfkk4&ab_channel=MoharemWK

[12]https://www.fablab.nknu.edu.tw/UploadFile/PBCStoreApplyReportFile/489_Report1_%E5%B8%82%E7%AB%8B%E6%96%87%E8%8F%AF%E5%9C%8B%E5%B0%8F_2019122011550.pdf

[13] https://www.rhydolabz.com/arduino-compatible-boards-accesories-for-arduino-c-152_169/rotary-encoder-module-for-arduino-p-2193.html

[14]https://www.digikey.tw/zh/articles/weighing-the-advantages-and-tradeoffs-of-encoder-technologies

[15] https://store.arduino.cc/usa/arduino-uno-rev3

[16]https://tw.element14.com/buy-raspberry-pi

[17] https://www.towerpro.com.tw/product/sg90-360-degree-continuous-rotation-servo/

[18]https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.rencheng.com.tw%2Fproducts_detail%2FMG90S&psig=AOvVaw1OHzr6UkWcIEr7e_jmQIW9&ust=1624092205785000&source=images&cd=vfe&ved=0CAsQjhxqFwoTCICAtIzloPECFQAAAAAdAAAAABAD

[19]https://shopee.tw/%E3%80%90RK%E9%9B%BB%E5%AD%90%E3%80%91MG996R-MG996%E8%88%B5%E6%A9%9F-%E8%BB%8A%E8%88%B9%E6%A9%9F%E6%A2%B0%E6%89%8B-13kg%E5%A4%A7%E6%89%AD%E5%8A%9B%E9%87%91%E5%B1%AC%E9%BD%92%E8%BC%AA%E8%88%B5%E6%A9%9F%E4%BC%BA%E6%9C%8D%E5%99%A8-i.132812476.2324358967

[20]http://www.imtf4.tw/topicdetail.php?t=776
[21]https://www.amazon.com/Flameer-A2217-2300KV-Brushless-Motor-Propellers/dp/B07KX974DH
[22]https://www.walmart.ca/en/ip/Bidirectional-20a-30a-40a-50a-60a-Brushless-Esc-for-Rc-Car-boat-Remote-Control-Ship-Pneumatic-Underwater-Propelle-Color-40A/5L5IAAR3CQLC
[23]https://battery9999.com/18650%E9%8B%B0%E9%9B%BB%E6%B1%A0%E4%BB%8B%E7%B4%B9/
[24]https://www.gensace.de/gens-ace-2250mah-6-6v-2s1p-transmitter-pack.html
[25]https://www.hotmcu.com/ir-reflective-sensor-tcrt5000-p-184.html
[26]https://www.ruten.com.tw/item/show?21610035304331
[27]https://www.wareorigin.com/products/lm393-chip-motor-measuring-comparator-speed-sensor-module-slot-type-ir-optocoupler-for-mcu-arduino
[28]https://www.mouser.tw/new/ams/ams-as5600-so-ek-ab/
[29] “K-epsilon”, Simscale, https://www.simscale.com/docs/simulation-setup/global-settings/k-epsilon
[30] JavaProp User’s Guide, https://www.mh-aerotools.de/airfoils/java/JavaProp%20Users%20Guide.pdf
[31] “CFDTutorial- AxialFansimulation|ANSYSFluent”, Youtube
[32] Final Report of Group #10, Practice of Mechanical Engineering (2019), Department of Mechanical Engineering, National Taiwan University.
[33] https://www.sciencedirect.com/science/article/pii/S2238785419317089
[34] https://www.mathworks.com/help/vdynblks/index.html

8. 附件

8.1. BOM 表

編號	項目名稱	數量	單價 (元)	總價 (元)	來源	材質/規格
F.風扇系統				0		
F01	風罩	1		0	3Dprint	PLA
F02	風扇	1		0	3Dprint	PLA
F03	馬達支架	1		0	3Dprint	PLA
F04	馬達	1	290	290	光華商場	A2212 2300KV
C.車體						
C01	頂板	1	10	10	雷射切割	密集板
C02	底板	1	10	10	雷射切割	密集板
C03	輪子	2	0	0	雷射切割	密集板
C04	轉向馬達支架1	2	0	0	雷射切割	密集板
C05	轉向馬達支架2	4	0	0	雷射切割	密集板
C06	轉向-馬達端	2	0	0	3Dprint	PLA
C07	轉向桿輪端	2	0	0	3Dprint	PLA
C08	轉向軸承固定件	2	0	0	3Dprint	PLA
C09	煞車桿	1	0	0	3Dprint	PLA
C10	煞車機構(上)	1	0	0	雷射切割	密集板
C11	煞車機構(下)	1	0	0	雷射切割	密集板
C12	牛眼輪	2	0	0	3Dprint	PLA
C13	煞車馬達支架	2	0	0	雷射切割	密集板
C14	蜜豆奶擋板	3	0	0	雷射切割	密集板
C15	鋁柱	4	10	40	蝦皮	
C16	輪胎皮	1	10	10	五金行	3M
C17	輪胎軸承	2	55	110	蝦皮	
C18	轉向軸承	2	55	110	蝦皮	
C19	銅柱	2	5	10	蝦皮	銅合金
C20	彈簧	2	20	40	五金行	不鏽鋼
C21	M6螺絲	6	2	12		
C22	M3螺絲	N	50	50		
C23	M2螺絲	4	1	4		
C24	蜜豆奶	1	12	12	便利商店	300 ml
E.動力和控制系統						
E01	Arduino Mega	1	400	400	光華商場	
E02	18650	2	110	220	光華商場	3350mAh
E03	18650電池盒	1	43	43	祥昌電子商場	
E04	鋰電池	1	450	450	蝦皮	2250mA 3S(11.1V)
E05	LM393	2	80	160	祥昌電子商場	
E06	TCRT5000	11	47.5	522.5	祥昌電子商場	
E07	開關	1	30	30	祥昌電子商場	
E08	電變	1	450	450	蝦皮商場	bidirectional 30A
E09	SG-90	2	70	140	今華電子	
total				2983.5		

8.2. 程式碼

```
#include <Servo.h>
#include <SoftwareSerial.h>

////////////////////////////
/////// Motor object ///////
////////////////////////////
Servo FanMotor;
Servo TurnMotor;
Servo BrakeMotor;

////////////////////////////
//////// PIN ///////////
////////////////////////////

// MOTOR
const int fan_pin = 11;
const int turnmotor_pin = 8;
const int brakemotor_pin = 9;

// ENCODER
const int encoder_l = 50;
const int encoder_r = 51;

// IR id -- index for IR[6] and IRthres[6]
#define FLL 0
#define FL 1
#define FM 2
#define FR 3
#define FRR 4
#define BL 5
#define BM 6
#define BR 7
#define ADD 8
#define BRR 9
#define BLL 10

// IR
const int IR[11] = {A14, A0, A8, A9 ,A15, A5, A13, A11, A7, A1, A2};

// IR threshold
const int IRthres[11] = {150, 270, 720, 350, 300, 490, 600, 920, 100, 450, 600};

////////////////////////////////////
//////// Global Variables /////////
////////////////////////////////////
int mid_angle = 86;
int cur_angle = mid_angle;

double init_time = 0;
double time = 0;

int brake_off = 125;
int brake_on = 80;

```

```
float left_count = 0;
int left_state = 0;
float right_count = 0;
int right_state = 0;

////////////////////////////////////
///////// Global Functions /////////
////////////////////////////////////
void fan_motor_write(int value){
    Serial.print("fan give a speed ");
    int upperlimit = 105;
    int lowerlimit = 75;
    if((value < lowerlimit) || (value > upperlimit)){
        FanMotor.write(90);
        Serial.println("\nfan error. exiting...");
        myexit(0);
    }
    else{
        Serial.println(value);
        FanMotor.write(value);
    }
}

void myexit(int n){
    if(n == 0){
        fan_motor_write(90);
        BrakeMotor.write(brake_on);
        TurnMotor.write(mid_angle);
        Serial.println("exit");
        delay(2000);
        BrakeMotor.write(brake_off);
        delay(200);
        exit(0);
    }else{
        exit(-1);
    }
}

void left_encoder(){
    int bit = digitalRead(encoder_l);
    if(bit != left_state){
        left_state = bit;
        left_count += 1;
        Serial.print("<L_encoder> ");
        Serial.print(left_count);
        Serial.println("  ");
    }
}

void right_encoder(){
    int bit = digitalRead(encoder_r);
    if(bit != right_state){
        right_state = bit;
        right_count += 1;
```

```
109             Serial.print("<R_encoder> ");
110             Serial.print(right_count);
111             Serial.println("  ");
112         }
113     }
114
115     //////////////////////////
116     ///////// Setup //////////
117     //////////////////////////
118     void setup(){
119         Serial.begin(9600);
120         FanMotor.attach(fan_pin);
121         fan_motor_write(90);
122         delay(1000);
123         TurnMotor.attach(turnmotor_pin);
124
125         TurnMotor.write(mid_angle+10);
126         delay(200);
127         TurnMotor.write(mid_angle-10);
128         delay(200);
129         TurnMotor.write(mid_angle-1);
130
131         BrakeMotor.attach(brakemotor_pin);
132         BrakeMotor.write(brake_off);
133
134         pinMode(encoder_l, INPUT_PULLUP);
135         pinMode(encoder_r, INPUT_PULLUP);
136
137         digitalRead(encoder_l);
138         digitalRead(encoder_r);
139
140         pinMode(IR[FLL], INPUT);
141         pinMode(IR[FL], INPUT);
142         pinMode(IR[FM], INPUT);
143         pinMode(IR[FR], INPUT);
144         pinMode(IR[FRR], INPUT);
145         pinMode(IR[BL], INPUT);
146         pinMode(IR[BM], INPUT);
147         pinMode(IR[BR], INPUT);
148         pinMode(IR[ADD], INPUT);
149         pinMode(IR[BLL], INPUT);
150         pinMode(IR[BRR], INPUT);
151
152         delay(2000);
153         Serial.println("Initialized");
154     }
155
156
157
158     //////////////////////////
159     ///////// Loop ///////////
160     //////////////////////////
161     // loop() only call 1 time
162     void loop(){
```

```
void loop(){
    fan_motor_write(80);
    left_count = 0;

    Serial.println("** Go straight");
    while(1){
        left_encoder();
        if(left_count >= 48){
            break;
        }
    }

    BrakeMotor.write(brake_on); delay(500);
    cur_angle = mid_angle - 40;
    TurnMotor.write(cur_angle); delay(800);

    Serial.println("** Start turning");
    BrakeMotor.write(brake_off);

    right_count = 0;
    while(1){
        right_encoder();
        if(right_count >= 42){
            break;
        }
    }

    Serial.println("** Start straight");
    TurnMotor.write(mid_angle);
    right_count = 0;
    while(1)
    {
        right_encoder();
        if(right_count >= 30){
            break;
        }
    }

    fan_motor_write(86);
    Serial.println("** Following the 1st line");
    right_count = 0;
    int turn_angle = 0;
    int last_angle = 0;
    int IRvalue[11] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

    while(1){
        right_encoder();

        if(right_count >= 6.8)
            break;

        IRvalue[FLL] = (analogRead(IR[FLL]) > IRthres[FLL]);
        IRvalue[FL] = (analogRead(IR[FL]) > IRthres[FL]);
        IRvalue[FM] = (analogRead(IR[FM]) > IRthres[FM]);
```

```
216          IRvalue[FR] = (analogRead(IR[FR]) > IRthres[FR]);
217          IRvalue[FRR] = (analogRead(IR[FRR]) > IRthres[FRR]);
218
219          Serial.print("<IR_line> ");
220          Serial.print(IRvalue[FLL]);
221          Serial.print(" ");
222          Serial.print(IRvalue[FL]);
223          Serial.print(" ");
224          Serial.print(IRvalue[FM]);
225          Serial.print(" ");
226          Serial.print(IRvalue[FR]);
227          Serial.print(" ");
228          Serial.print(IRvalue[FRR]);
229
230          Serial.print(" <angle> ");
231
232          if((IRvalue[FLL] + IRvalue[FL] + IRvalue[FM] + IRvalue[FR] + IRvalue[FRR]) > 2){
233              turn_angle = 0;
234              Serial.print(" bad_line");
235          }
236          else{
237              turn_angle = (-23)*IRvalue[FLL] + (-15)*IRvalue[FL] + (0)*IRvalue[FM] +\
238                           (15)*IRvalue[FR] + (23)*IRvalue[FRR];
239              if(last_angle != turn_angle){
240                  right_count -= abs(turn_angle) * 0.07;
241              }
242              last_angle = turn_angle;
243          }
244          TurnMotor.write(mid_angle + turn_angle);
245          Serial.println(turn_angle);
246      }
247      TurnMotor.write(mid_angle);
248
249      //// 1st 循跡結束 ////
250      BrakeMotor.write(brake_on); delay(500);
251      cur_angle = mid_angle - 40;
252      TurnMotor.write(cur_angle);
253      fan_motor_write(78); delay(800);
254      Serial.println("** Start turning");
255      BrakeMotor.write(brake_off);
256      right_count = 0;
257      while(1){
258          right_encoder();
259          if(right_count >= 28)
260              break;
261      }
262
263      fan_motor_write(81);
264      Serial.println("** Start straight to find 2nd line");
265      TurnMotor.write(mid_angle);
266      right_count = 0;
267      int flag = 0;
268      while(1){
269          right_encoder();
```

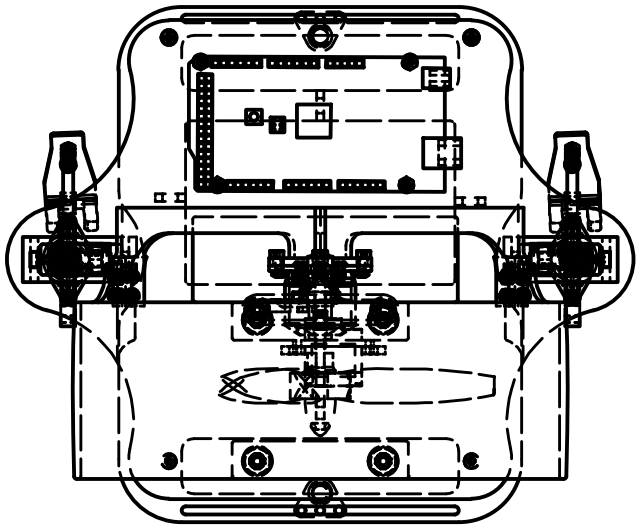
```
270         if( right_count >= 5 && flag == 0){
271             fan_motor_write(86);
272             flag = 1;
273         }
274         if(analogRead(IR[ADD]) > IRthres[ADD] && right_count > 3){
275             Serial.println("stop");
276             break;
277         }
278     }
279 
280     BrakeMotor.write(brake_on); delay(500);
281     cur_angle = mid_angle + 40;
282     TurnMotor.write(cur_angle);
283     fan_motor_write(81); delay(800);
284     Serial.println("** Start turning to be straight");
285     BrakeMotor.write(brake_off);
286 
287     right_count = 0;
288     while(1){
289         right_encoder();
290         if(analogRead(IR[FRR]) > IRthres[FRR])
291             break;
292     }
293     BrakeMotor.write(brake_on);
294     fan_motor_write(84); delay(500);
295     BrakeMotor.write(brake_off);
296 
297 
298     //// 2nd 循跡開始 ////
299 
300     Serial.println("** Following the 2nd line");
301     left_count = 0;
302     turn_angle = 0;
303     last_angle = 0;
304 
305     flag = 0;
306 
307     while(1){
308         left_encoder();
309 
310         if(flag == 1 && left_count >= 13){
311             break;
312         }
313         IRvalue[FLL] = (analogRead(IR[FLL]) > IRthres[FLL]);
314         IRvalue[FL] = (analogRead(IR[FL]) > IRthres[FL]);
315         IRvalue[FM] = (analogRead(IR[FM]) > IRthres[FM]);
316         IRvalue[FR] = (analogRead(IR[FR]) > IRthres[FR]);
317         IRvalue[FRR] = (analogRead(IR[FRR]) > IRthres[FRR]);
318 
319         Serial.print("<IR_line> ");
320         Serial.print(IRvalue[FLL]);
321         Serial.print(" ");
322         Serial.print(IRvalue[FL]);
323         Serial.print(" ");
```

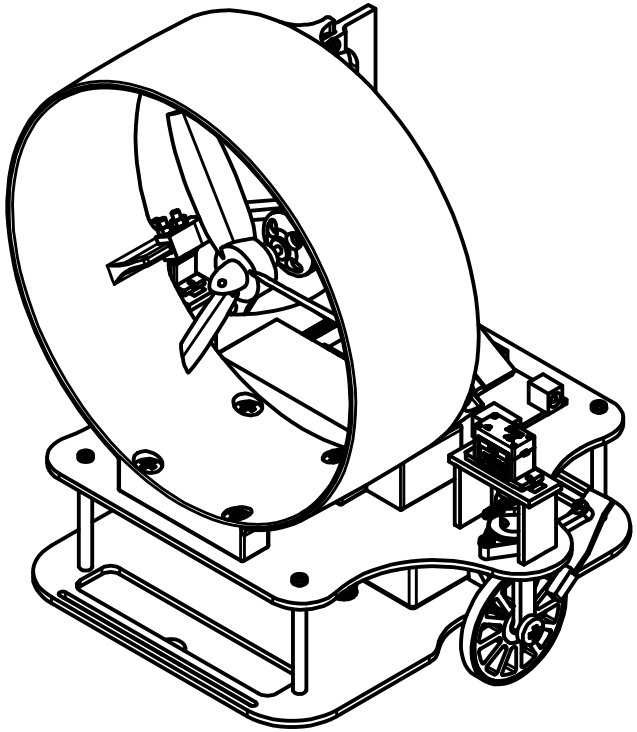
```
324             Serial.print(IRvalue[FM]);
325             Serial.print(" ");
326             Serial.print(IRvalue[FR]);
327             Serial.print(" ");
328             Serial.print(IRvalue[FRR]);
329 
330             Serial.print(" <angle> ");
331             if((IRvalue[FLL] + IRvalue[FL] + IRvalue[FM] + IRvalue[FR] + IRvalue[FRR]) > 2){
332                 turn_angle = 0;
333                 Serial.print("bad_line");
334                 flag = 1;
335                 left_count = 0;
336             }
337             else{
338                 turn_angle = (-18)*IRvalue[FLL] + (-12)*IRvalue[FL] + (0)*IRvalue[FM] +\
339                              (12)*IRvalue[FR] + (18)*IRvalue[FRR];
340                 if(last_angle != turn_angle){
341                     right_count -= abs(turn_angle) * 0.07;
342                 }
343                 last_angle = turn_angle;
344             }
345             TurnMotor.write(mid_angle + turn_angle);
346             Serial.println(turn_angle);
347         }
348         TurnMotor.write(mid_angle);
349         BrakeMotor.write(brake_on);
350 
351         delay(6000);
352         fan_motor_write(100);
353         TurnMotor.write(mid_angle+10);
354         delay(200);
355         TurnMotor.write(mid_angle-10);
356         delay(200);
357         TurnMotor.write(mid_angle-1);
358         delay(500);
359         BrakeMotor.write(brake_off);
360 
361         Serial.println("** Start straight");
362         right_count = 0;
363         int badline_count = 0;
364         while(1){
365             right_encoder();
366 
367             if(right_count >= 200 && badline_count >= 3)
368                 break;
369             IRvalue[BLL] = (analogRead(IR[BLL]) > IRthres[BLL]);
370             IRvalue[BL] = (analogRead(IR[BL]) > IRthres[BL]);
371             IRvalue[BM] = (analogRead(IR[BM]) > IRthres[BM]);
372             IRvalue[BR] = (analogRead(IR[BR]) > IRthres[BR]);
373             IRvalue[BRR] = (analogRead(IR[BRR]) > IRthres[BRR]);
374 
375             Serial.print("<IR_line> ");
376             Serial.print(IRvalue[BLL]);
377             Serial.print(" ");
```

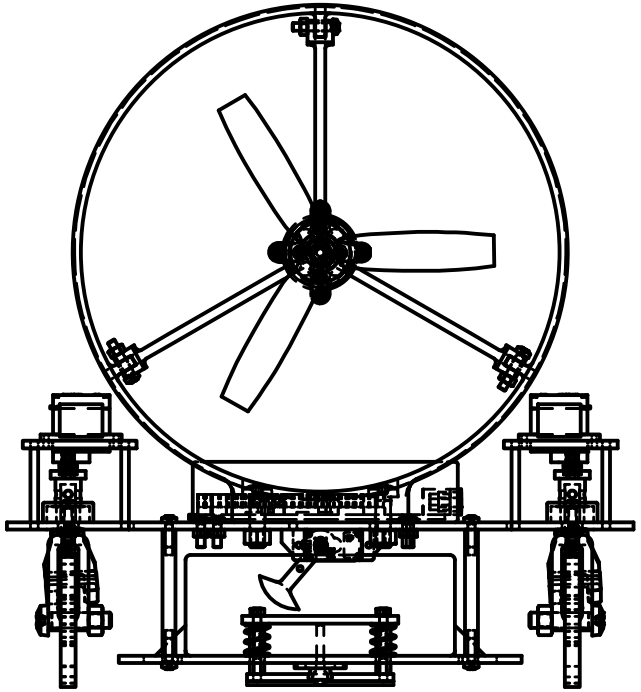
```
            Serial.print(IRvalue[BL]);
            Serial.print(" ");
            Serial.print(IRvalue[BM]);
            Serial.print(" ");
            Serial.print(IRvalue[BR]);
            Serial.print(" ");
            Serial.print(IRvalue[BRR]);

            Serial.print(" <angle> ");

            if((IRvalue[BLL] + IRvalue[BL] + IRvalue[BM] + IRvalue[BR] + IRvalue[BRR]) > 2){
                turn_angle = 0;
                Serial.print(" bad_line");
                badline_count += 1;
            }
            else{
                turn_angle = (-25)*IRvalue[BLL] + (-15)*IRvalue[BL] + (0)*IRvalue[BM] +\
                             (15)*IRvalue[BR] + (25)*IRvalue[BRR];
            }
            TurnMotor.write(mid_angle + turn_angle);
            Serial.println(turn_angle);

        }
        //// 2nd 循跡結束 ////
        myexit(0);
    }
```

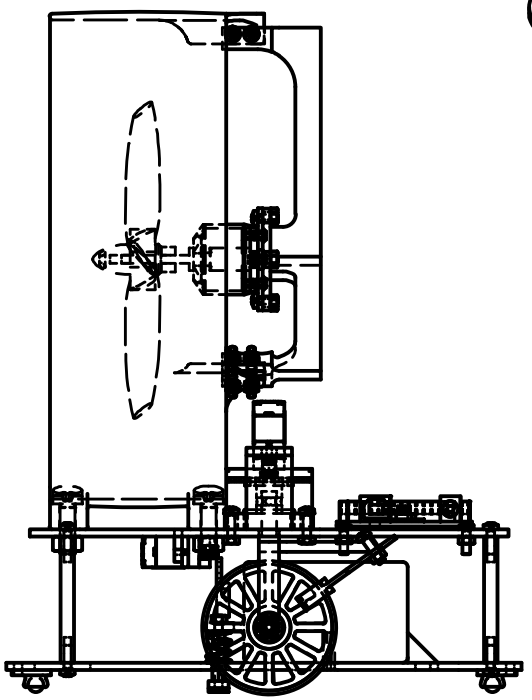
8.3. 工程圖

見下頁



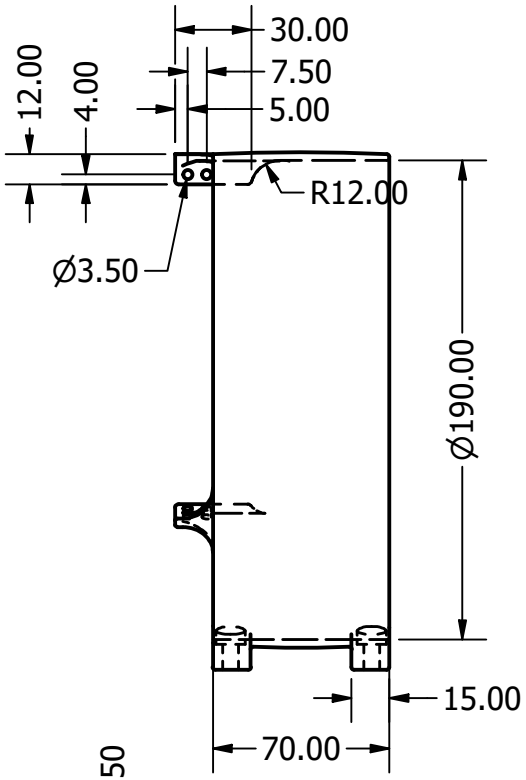


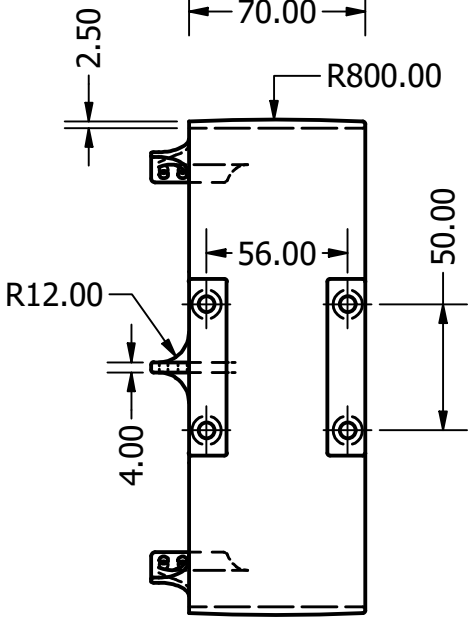




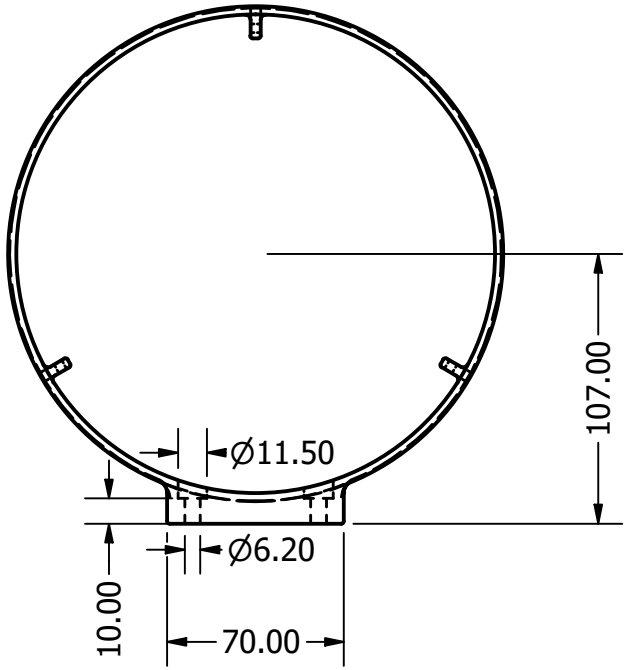
TITLE
小飛象章魚

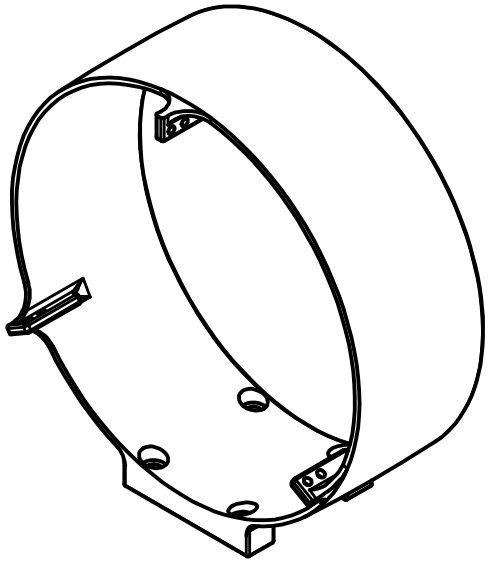
SCALE	GROUP
1 / 3	機械工程實務 第12組



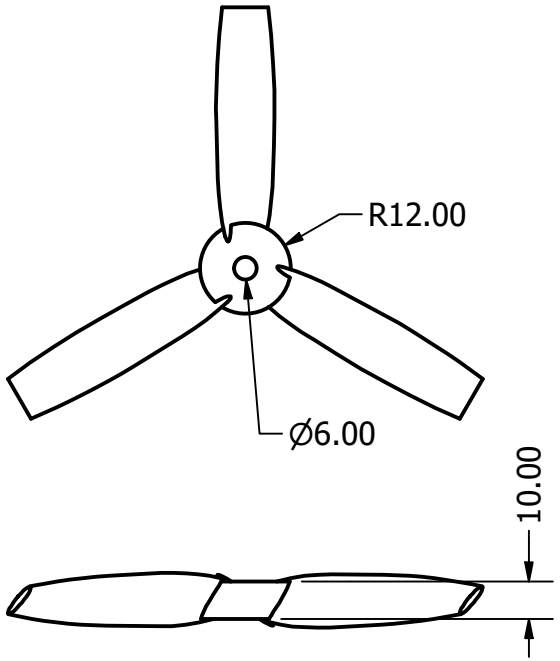


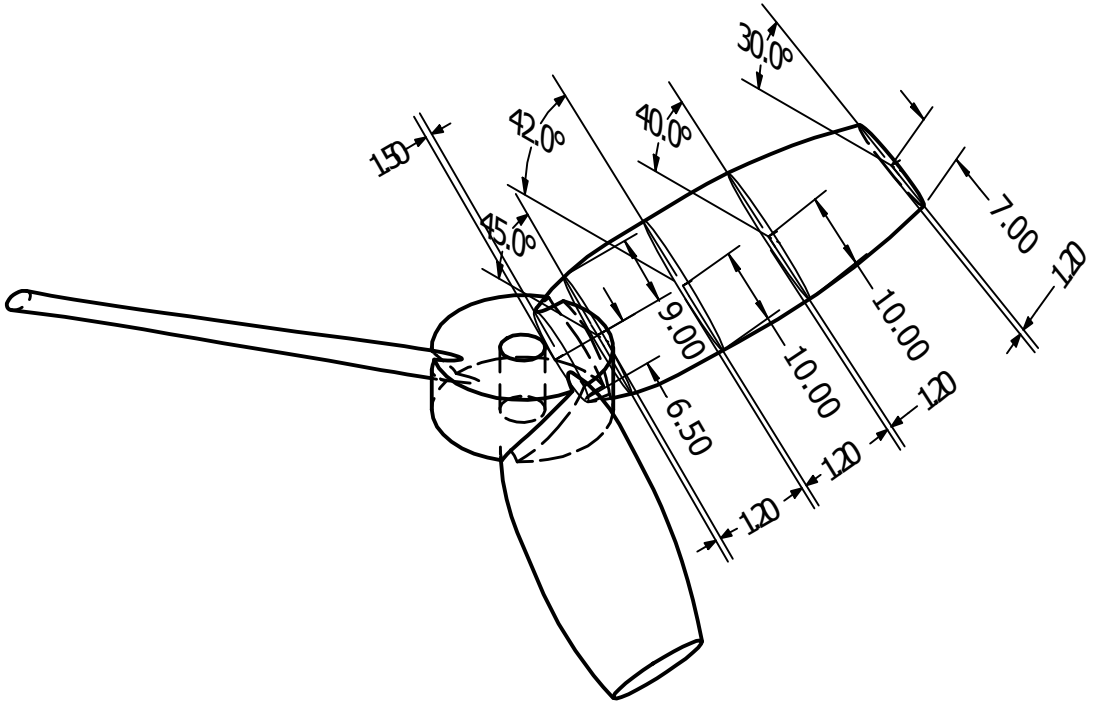
註:其餘圓角皆為R1.00



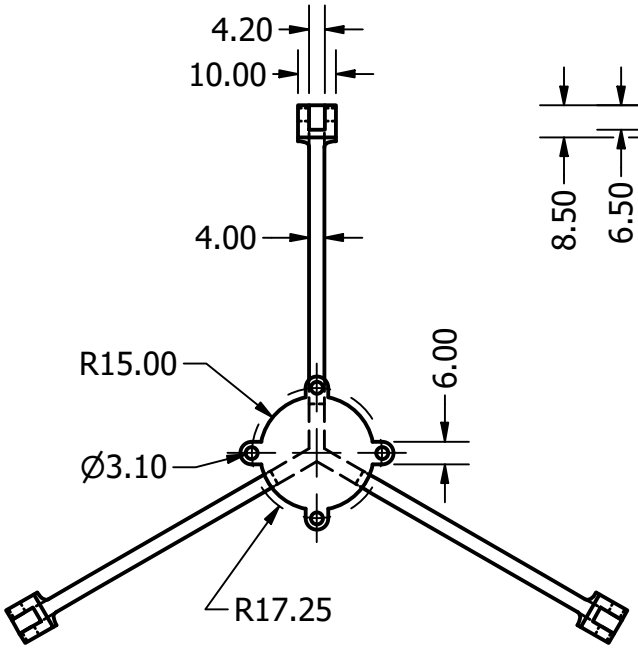


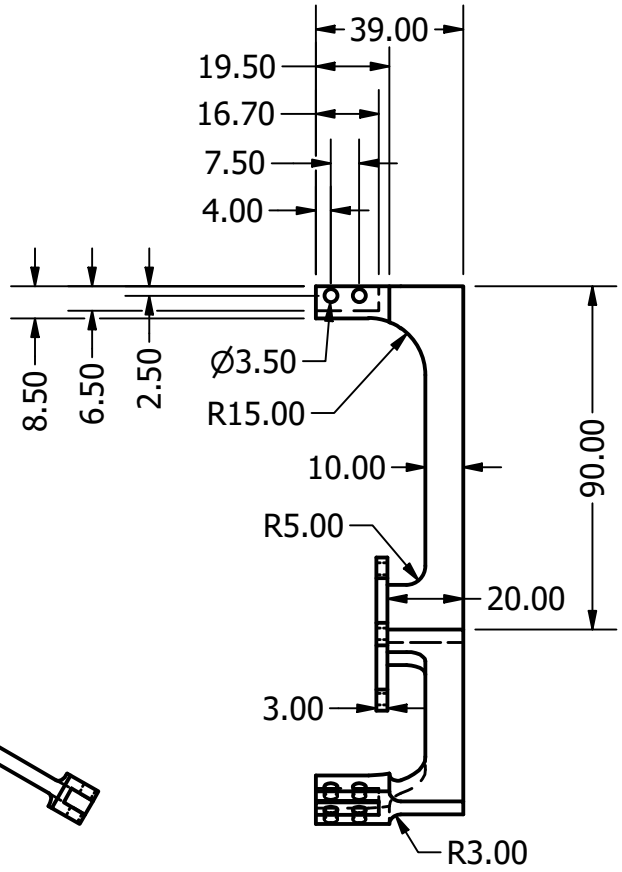
DESIGNER 陳旻彥	DATE 2021/6/17	TITLE 風罩	
MATERIAL PLA	DWG NO F01		
SCALE 1 / 3	QUANTITY 1	SYSTEM NAME Fan	GROUP 機械工程實務 第12組

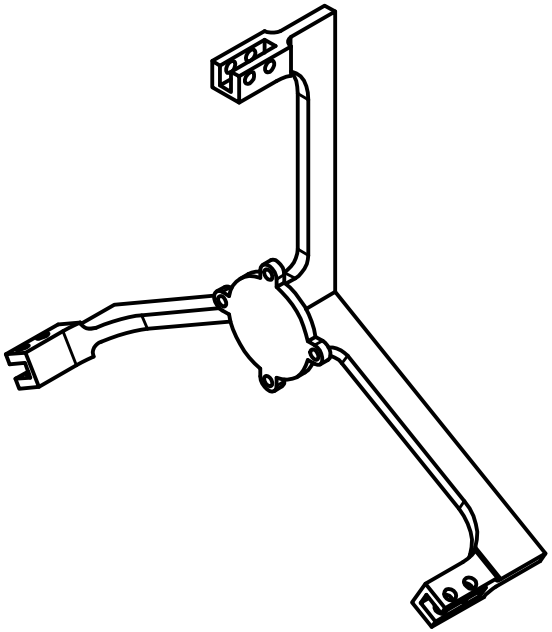




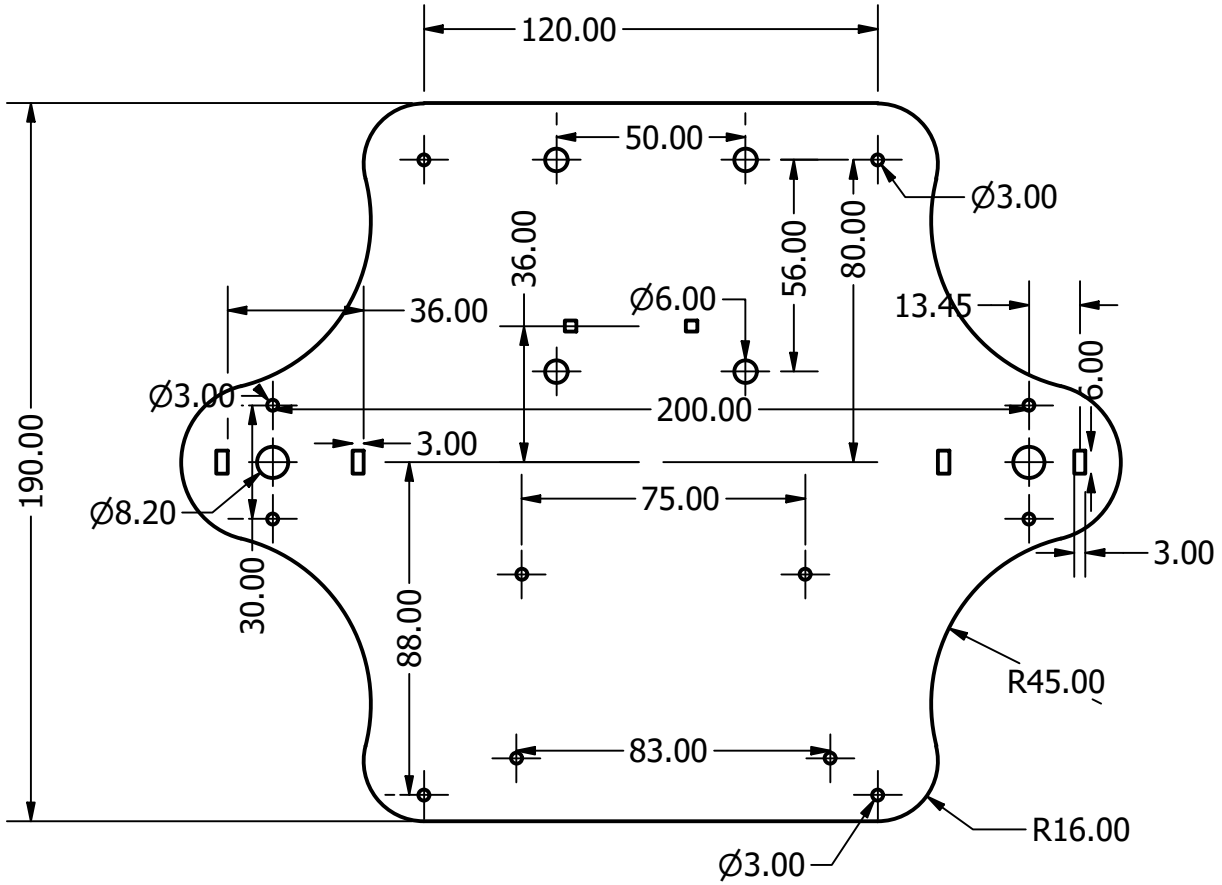
DESIGNER 賴昭蓉	DATE 2021/6/14	TITLE 風扇	
MATERIAL PLA	DWG NO F02		
SCALE 1 : 1	QUANTITY 1	SYSTEM NAME Fan	GROUP 機械工程實務 第12組



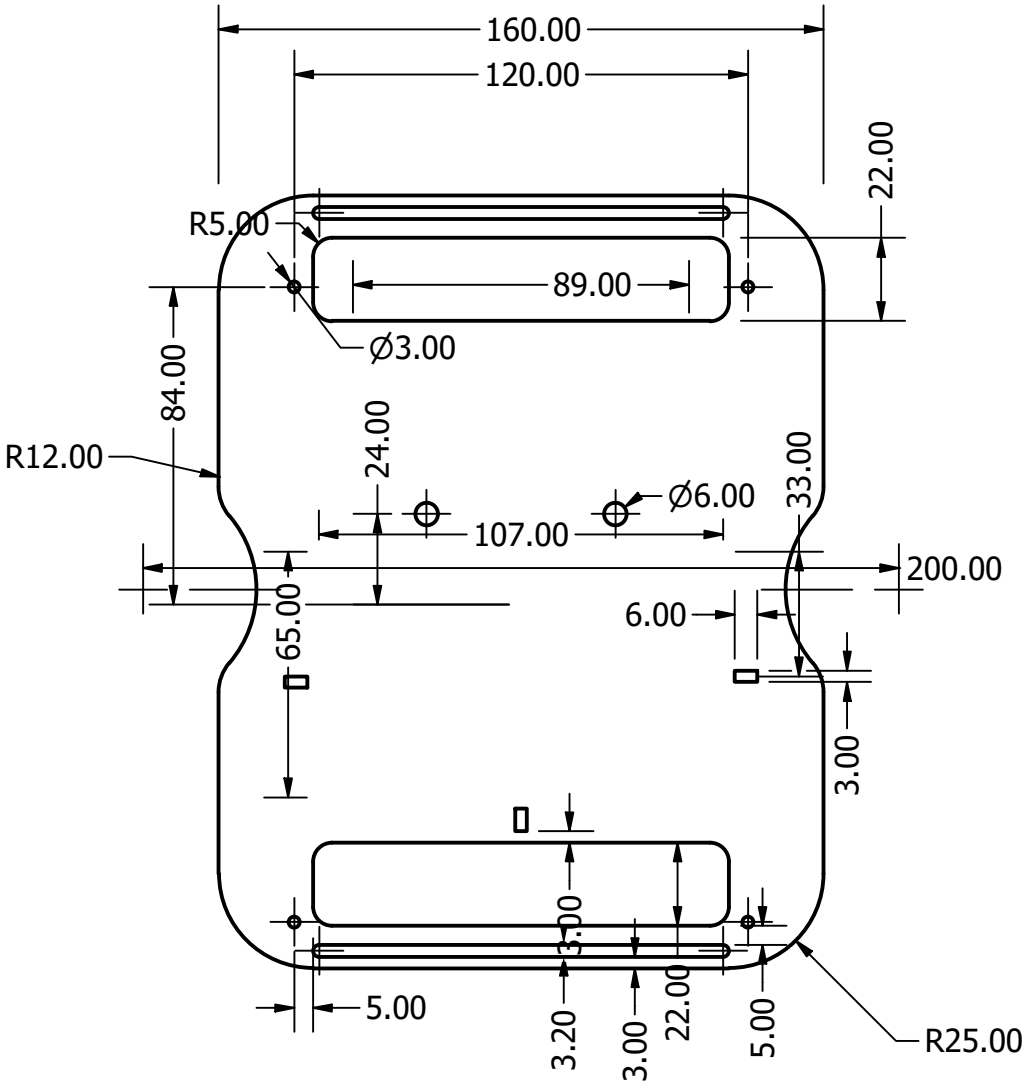




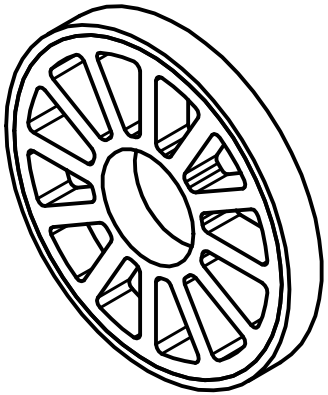
DESIGNER 陳旻彥	DATE 2021/6/17	TITLE 馬達支架	
MATERIAL PLA	DWG NO F03		
SCALE 1 / 2	QUANTITY 1	SYSTEM NAME Fan	GROUP 機械工程實務 第12組

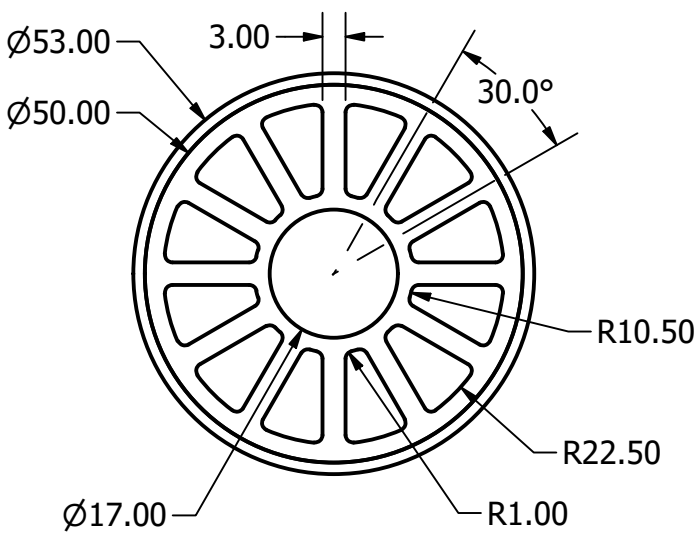


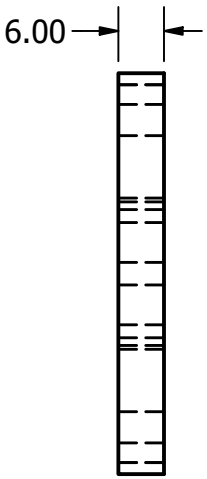
DESIGNER 楊宜瑄	DATE 5/12/2021	TITLE 頂板	
MATERIAL 3mm MDF	DWG NO C01		
SCALE 1 / 2	QUANTITY 1	SYSTEM NAME Chassis	GROUP 機械工程實務 第12組



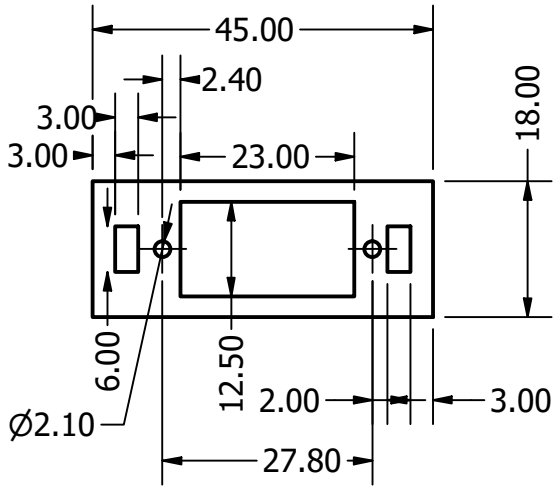
DESIGNER 楊宜瑄	DATE 5/14/2021	TITLE 底板	
MATERIAL 3mm MDF	DWG NO C02		
SCALE 1 / 2	QUANTITY 1	SYSTEM NAME Chassis	GROUP 機械工程實務 第12組



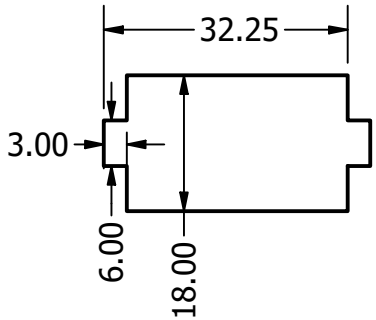




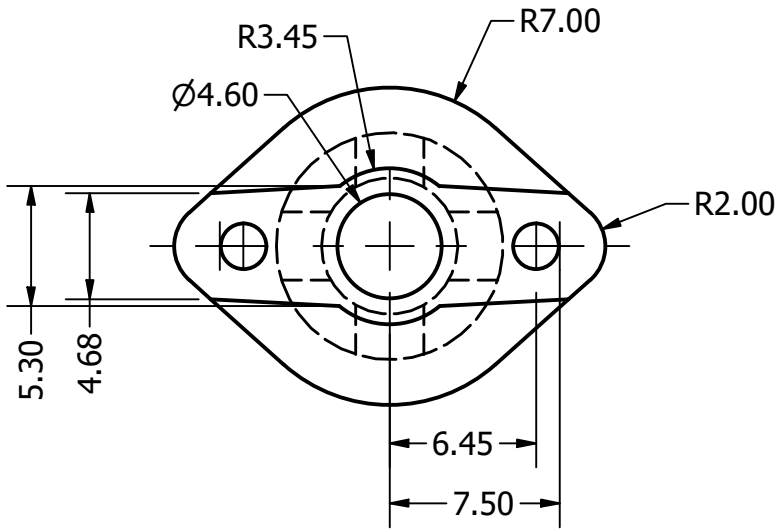
DESIGNER 陳旻彥	DATE 2021/6/17	TITLE 車輪	
MATERIAL 3mm MDF	DWG NO C03		
SCALE 1 : 1	QUANTITY 2	SYSTEM NAME Chassis	GROUP 機械工程實務 第12組

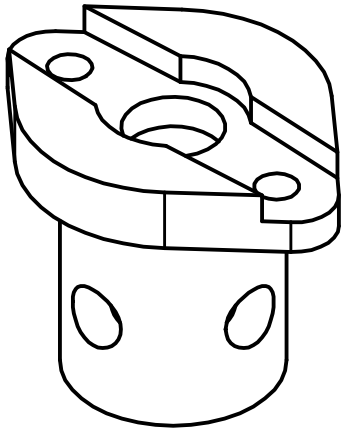


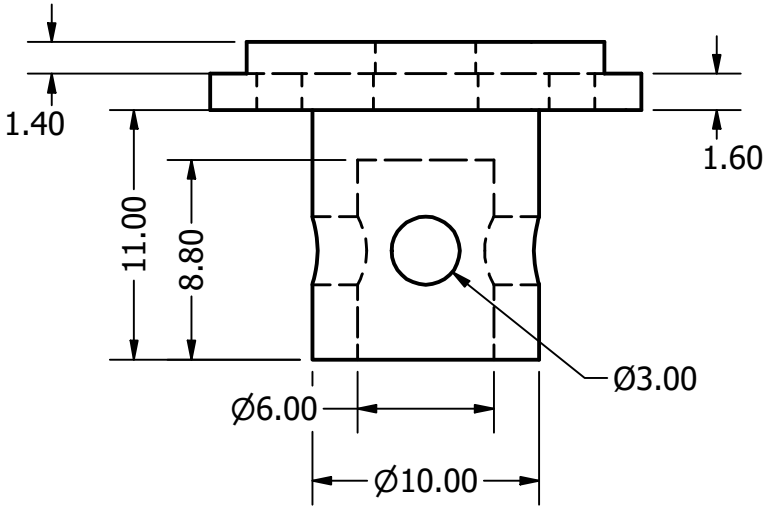
DESIGNER 楊宜瑄	DATE 12/28/2020	TITLE 轉向馬達支架1	
MATERIAL 3mm MDF	DWG NO C04		
SCALE 1 : 1	QUANTITY 2	SYSTEM NAME Chassis	GROUP 機械工程實務 第12組

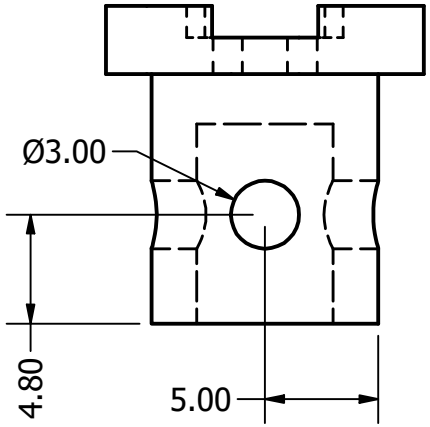


DESIGNER 楊宜瑄	DATE 5/21/2021	TITLE 轉向馬達支架2	
MATERIAL 3mm MDF	DWG NO C05		
SCALE 1 : 1	QUANTITY 4	SYSTEM NAME Chassis	GROUP 機械工程實務 第12組

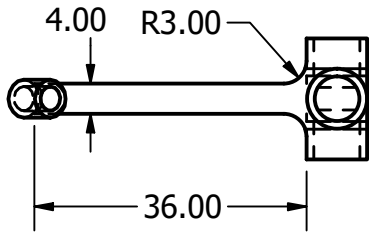


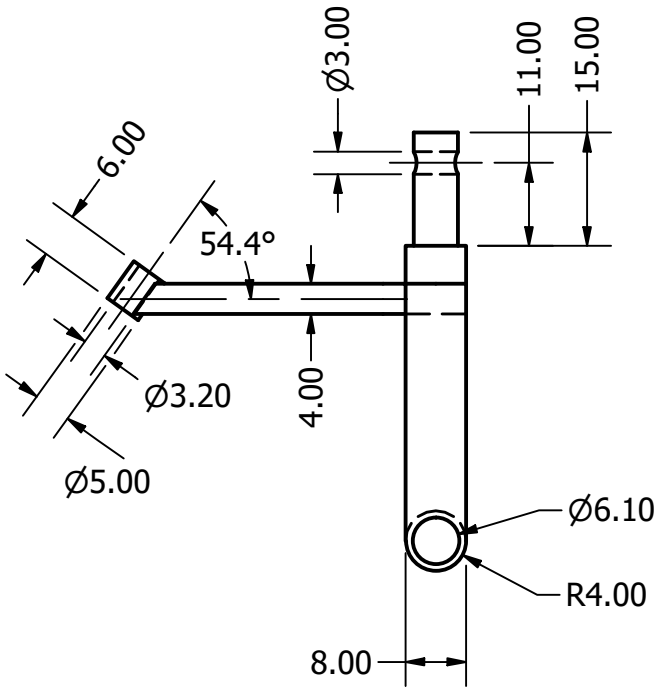


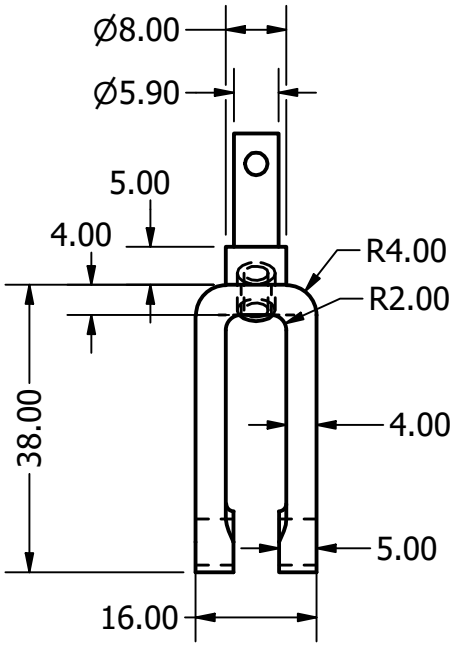


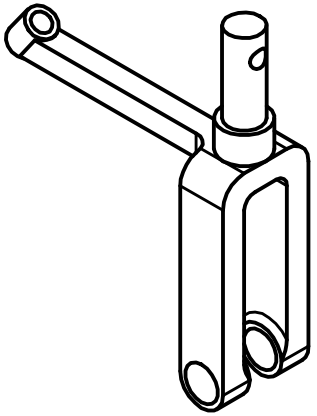


DESIGNER 陳旻彥	DATE 2021/6/17	TITLE 轉向馬達端	
MATERIAL PLA	DWG NO C06		
SCALE 3 : 1	QUANTITY 2	SYSTEM NAME Chassis	GROUP 機械工程實務 第12組

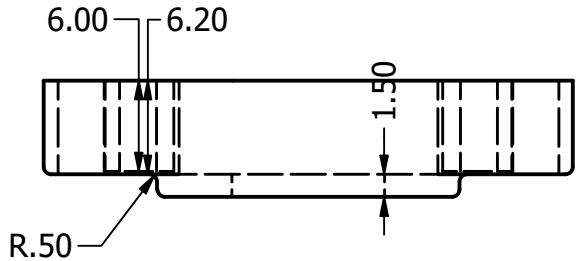


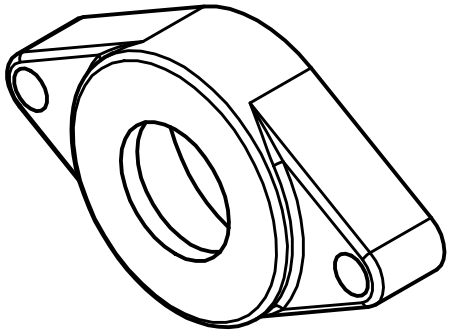


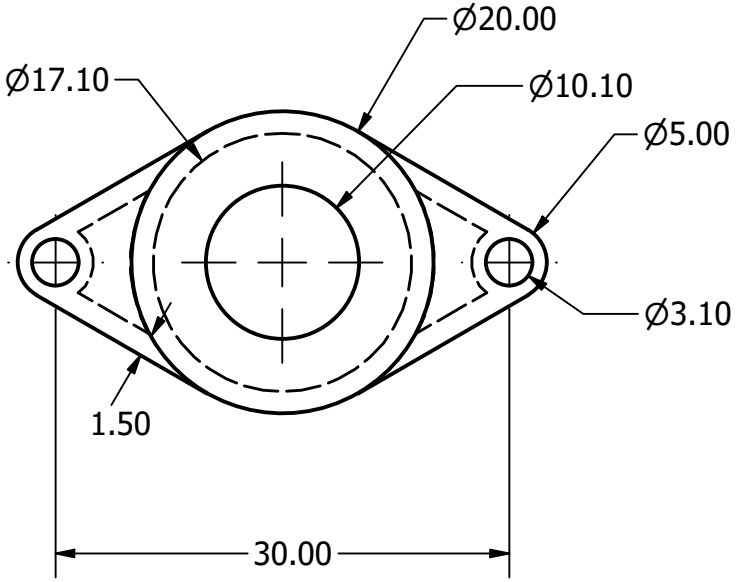


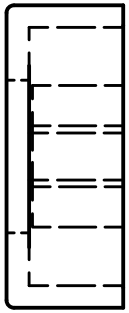


DESIGNER 陳旻彥	DATE 2021/6/17	TITLE 轉向桿輪端	
MATERIAL PLA	DWG NO C07		
SCALE 1 : 1	QUANTITY 2	SYSTEM NAME Chassis	GROUP 機械工程實務 第12組

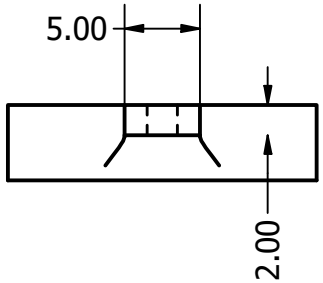


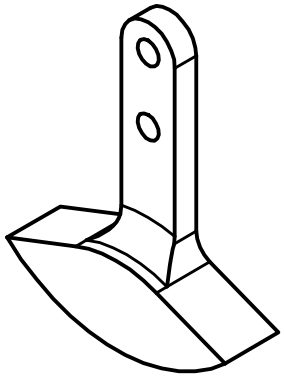


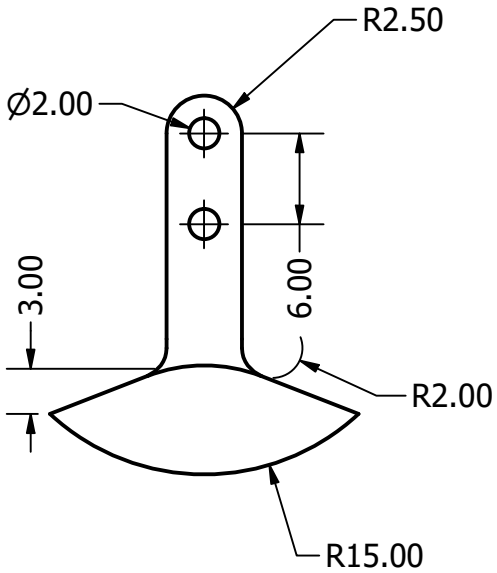


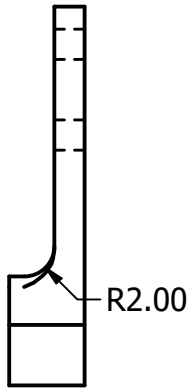


DESIGNER 楊宜瑄	DATE 4/7/2021	TITLE 轉向軸承固定件	
MATERIAL PLA	DWG NO C08		
SCALE 2 : 1	QUANTITY 2	SYSTEM NAME Chassis	GROUP 機械工程實務 第12組

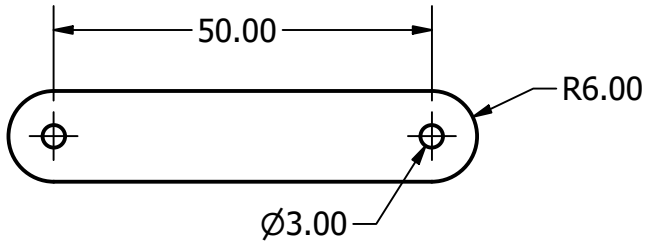




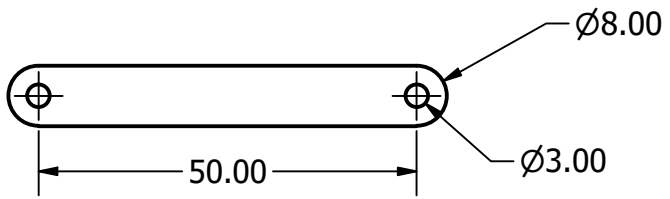




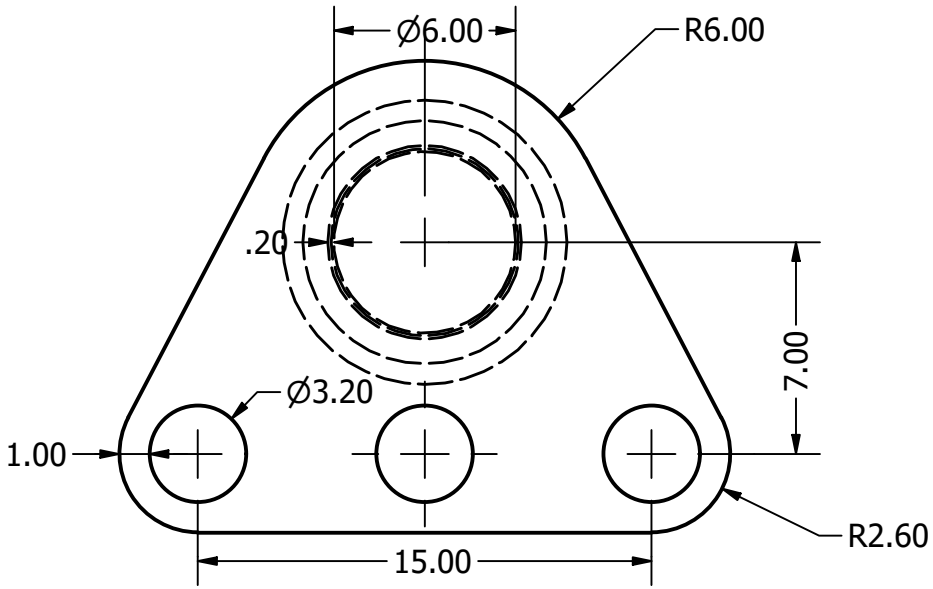
DESIGNER 楊宜瑄	DATE 4/17/2021	TITLE 煞車桿	
MATERIAL PLA	DWG NO C09		
SCALE 2 : 1	QUANTITY 1	SYSTEM NAME Chassis	GROUP 機械工程實務 第12組

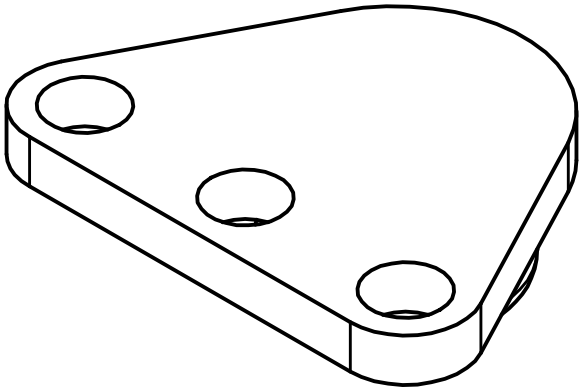


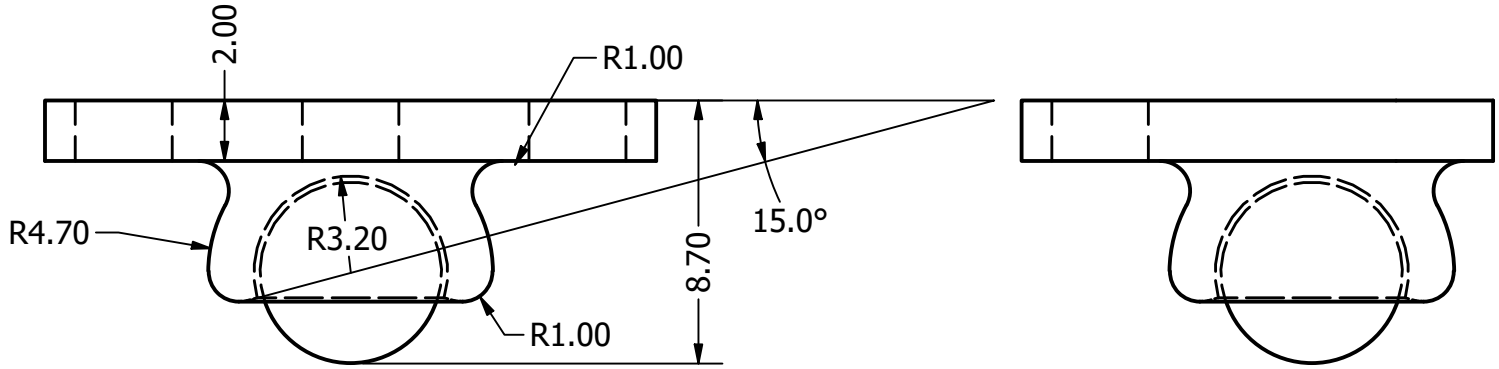
DESIGNER 楊宜瑄	DATE 4/14/2021	TITLE 煞車機構(上)	
MATERIAL 3mm MDF	DWG NO C10		
SCALE 1 : 1	QUANTITY 1	SYSTEM NAME Chassis	GROUP 機械工程實務 第12組



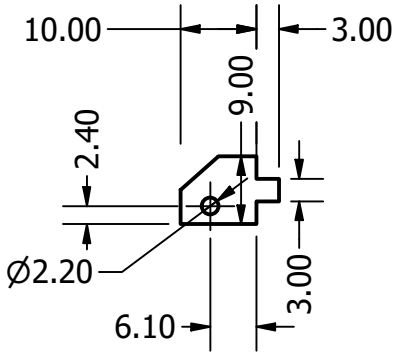
DESIGNER 楊宜瑄	DATE 4/14/2021	TITLE 煞車機構(下)	
MATERIAL 3mm MDF	DWG NO C11		
SCALE 1 : 1	QUANTITY 1	SYSTEM NAME Chassis	GROUP 機械工程實務 第12組



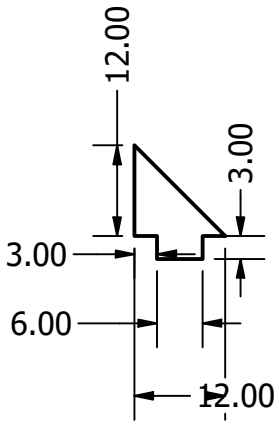




DESIGNER 楊宜瑄	DATE 5/31/2021	TITLE 牛眼輪	
MATERIAL PLA	DWG NO C12		
SCALE 4 : 1	QUANTITY 2	SYSTEM NAME Chassis	GROUP 機械工程實務 第12組



DESIGNER 楊宜瑄	DATE 4/17/2021	TITLE 煞車馬達支架	
MATERIAL 3mm MDF	DWG NO C13		
SCALE 1:1	QUANTITY 2	SYSTEM NAME Chassis	GROUP 機械工程實務 第12組



DESIGNER 楊宜瑄	DATE 4/7/2021	TITLE 蜜豆奶擋板	
MATERIAL 3mm MDF	DWG NO C14		
SCALE 1:1	QUANTITY 3	SYSTEM NAME Chassis	GROUP 機械工程實務 第12組