國 立 虎 尾 科 技 大 學
機 械 設 計 工 程 系 暨 精 密 機 械 工 程 科

專 題 製 作 報 告

ODOO PLM 在協同設計上的應用 - 以鋼球平衡台設計為例

Application of ODOO PLM in collaborative design - taking the Design of Steel Ball Balancing Platform as an example

指導教授： 嚴 家 銘 老 師
班 級： 四 設 三 乙
學 生： 陳 岳 樫 (41023218)
蔡 弦 霖 (41023248)
鄭 立 揚 (41023251)
謝 鴻 元 (41023254)

中華民國 一一三 年 六 月

國立虎尾科技大學
機械設計工程系暨精密機械工程科
學生專題製作合格認可證明

專題製作修習學生： 四設三乙 41023218 陳岳樫
四設三乙 41023248 蔡弦霖
四設三乙 41023251 鄭立揚
四設三乙 41023254 謝鴻元

專題製作題目 ：ODOO PLM 在協同設計上的應用
-以鋼球平衡台設計為例

經評量合格，特此證明

評 審 委 員：_____

_____

_____

指 導 老 師：_____

系 主 任：_____

中 華 民 國 一一三 年 1 月 1 日

摘 要

本研究旨在探討如何利用 ODOO PLM 進行協同設計，以提高團隊合作效率和品質。通過分析 ODOO PLM 在協同設計過程中的應用效果，並提出相關的優化建議，以改善設計流程並推動協同設計的應用。

以鋼球平衡台設計為例，我們將透過 ODOO PLM 和 GitHub 進行協同設計、管理、製造執行及整合功能。設計過程中，我們將使用 Geogebra、Onshape 和 Solidworks 等工具設計機構，並透過 CoppeliaSim 和 Python 進行 PID 控制模擬。同時，使用自行維護的 3D 列印機製作所需零件，以實現虛實整合之目標。最後根據 ODOO PLM 和 GitHub 的記錄歷程，評估協同作業的工作模式。

關鍵字: 比例-積分-微分控制器 (PID)、產品生命周期管理 (PLM)、協同 (CD)、CoppeliaSim、Github

Abstract

This study aims to explore the utilization of ODOO PLM for collaborative design to enhance team cooperation efficiency and quality. By analyzing the application effectiveness of ODOO PLM in collaborative design processes and proposing relevant optimization suggestions, the research seeks to improve design workflows and promote the application of collaborative design.

Using the design of a steel ball balancing platform as an example, collaborative design, management, manufacturing execution, and integration functionalities will be conducted through ODOO PLM and GitHub. Throughout the design process, tools such as Geogebra, Onshape, and Solidworks will be employed to design mechanisms, with CoppeliaSim and Python utilized for PID control simulation. Additionally, required components will be fabricated using a self-maintained 3D printer to achieve the goal of virtual and real integration. Finally, based on the record history of ODOO PLM and GitHub, the collaborative operation mode will be evaluated.

Keywords: proportional–integral–derivative controller (PID), Product Lifecycle Management (PLM),collaborative(CD), CoppeliaSim ,Github

誌 謝

本專題能完成有著許多人員的幫忙，大四學長他們不吝嗇地將往年的製作經驗傳授給我們，讓我們在製作的時候少走了許多錯路，還總是貼心找出重點提醒我們可以加以描述。再來是我們的指導教授嚴家銘教授，他提供了多方面的資訊，拋出問題並給予建議，擬定了我們小組研究和學習的方向，討論也時常提出建議以及未來發展，得以順利解決遇到的技術問題，同時也給了相當程度的自由，讓小組得以有彈性去尋探索及摸索，而本專題組員也充分地付出了許多，讓專題研究能順利完成，從中獲益良多，特此感謝。

目　錄

圖 目 錄

表　目　錄

第一章　簡介

本專題係藉由鋼求平衡台設計，探討協同設計作業之工作模式。

1.1 研究流程

我們將藉由以下流程探討、研究、並分析協同工具在協同設計上的應用。

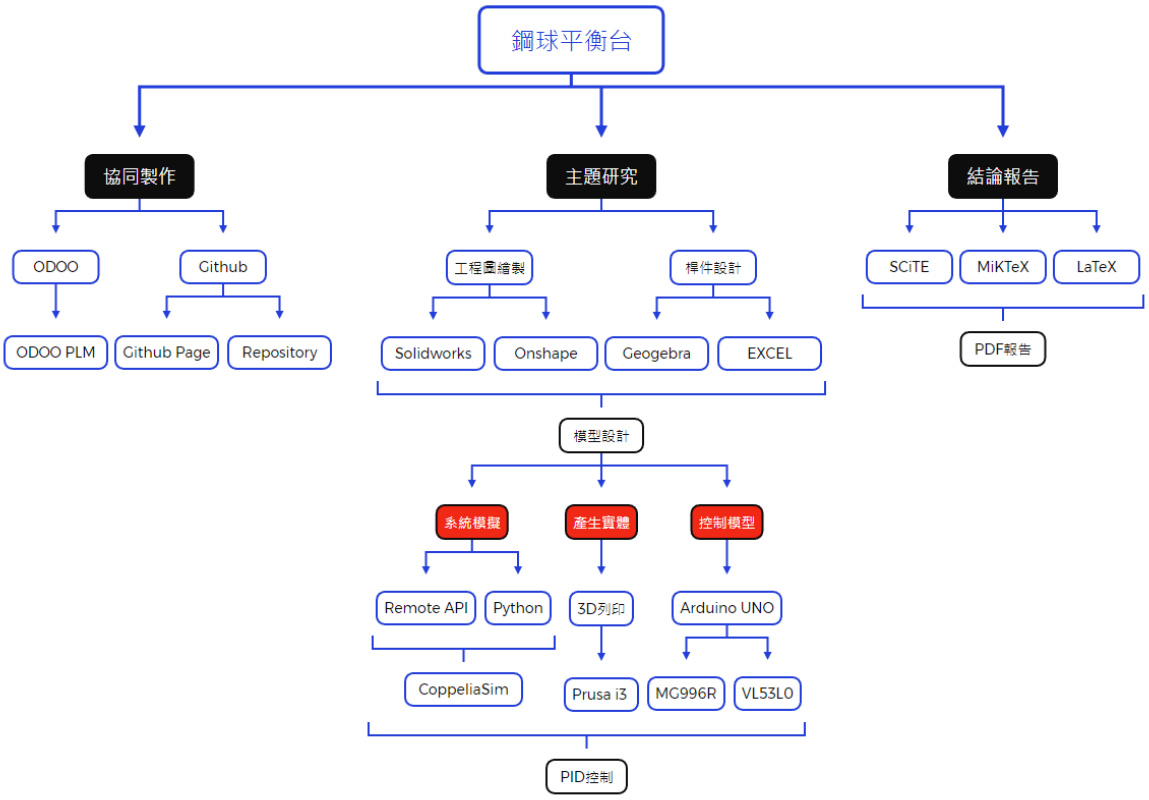


圖　1.1: 流程圖

1.2 專題環境介紹

1.2.1 協同環境

(一)ODOO PLM

ODOO 是一個開源的企業管理軟體，旨在幫助公司管理各種業務流程，包括銷售、庫存、會計、製造、資源規劃和人力資源等。ODOO 提供了一個集成的平台，讓用戶可以輕鬆地管理他們的業務活動。我們將在後面的章節詳細介紹其中的產品生命週期管理 (PLM) 功能。

圖 1.2: odoo 標誌

(二)GitHub

GitHub 是一個基於網絡的程式碼管理和協作平台，為開發者提供了一個集中式的位置來存儲、管理和共享他們的程式碼項目。它使用 Git 版本控制系統，允許用戶追蹤文件的變更、對其進行版本控制，並輕鬆地進行協作和交流。

圖 1.3: github 標誌

1.2.2 研究環境

(一)Solvespace

Solvespace 是一個開源的參數化 3D CAD（計算機輔助設計）軟體，旨在幫助用戶創建和編輯 3D 模型。它具有簡單直觀的用戶界面和豐富的功能，同時還支持多平台運行，包括 Windows、Linux 和 macOS。

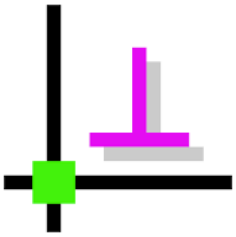


圖 1.4: Solvespace 標誌

(二)Onshape

Onshape 是一個基於雲端的三維計算機輔助設計（CAD）平台，提供了全功能的 CAD 工具，讓用戶可以通過網絡瀏覽器在任何設備上進行建模和設計。它的強大功能和靈活性使得用戶能夠輕鬆地創建複雜的三維模型，進行裝配設計、模擬分析以及技術文件繪製等工作。同時，Onshape 的即時協作功能還允許多個用戶同時訪問和編輯同一個設計文件，從而實現更加高效的團隊合作和設計工作流程。

圖 1.5: Onshape 標誌

(三)SolidWorks

SolidWorks 是一款由 Dassault Systèmes 開發的三維計算機輔助設計（CAD）軟體，廣泛應用於機械設計、工程設計和製造等領域，並且具有強大的建模功能，使得用戶可以快速、精確地創建複雜的三維模型，從而進行裝配設計、渲染、模擬分析等工作。

圖 1.6: SolidWorks 標誌

(四)Geogebra

Geogebra 是一款免費的動態數學（幾何）軟體，他的名稱是由 **Geo**metry（幾何）和 Al**gebra**（代數）所組成，主要功能包含 CAS 計算機、科學計算機、3D 計算機、計算與繪圖。其特點為能建立幾何對象，並保持它們之間的關係，可以用來觀察圖形變化或者製作簡單的動畫、快速的實驗數學上的想法，以製作教學演示材料。

圖 1.7: Geogebra 標誌

(五)CoppeliaSim

CoppeliaSim（舊稱 V-REP）是由瑞士 Coppelia Robotics AG 開發的先進機器人模擬器，主要應用於機器人研究和教育。其支援分佈式控制架構，允許使用 Python、Lua 和 C/C++ 等語言進行同步和異步控制。內建運動學引擎和多種物理模擬庫，提供精確的運動和剛體模擬。CoppeliaSim 能構建複雜的模型和場景，也能使用插件擴展功能，進行運動規劃及影像處理。

圖 1.8: CoppeliaSim 標誌

1.2.3 報告環境

LaTeX

LaTeX 是一種專業的排版系統，通常用於製作科學、技術和學術文檔，如論文、報告、書籍等。與常見的文字處理軟體，和 Microsoft Word 相比，LaTeX 以其強大的排版能力和對數學公式的支援而聞名。

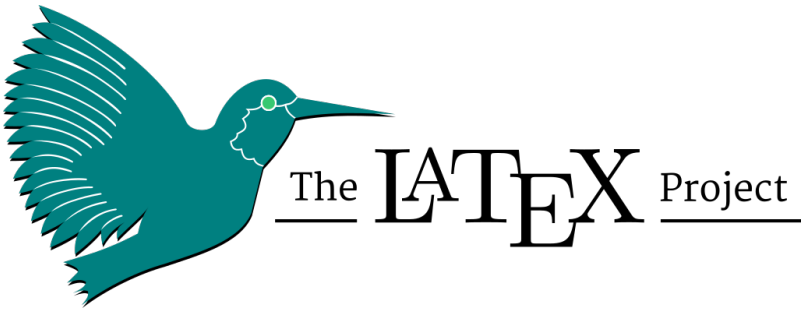


圖 1.9: LaTeX 標誌

第二章　背景與動機

專案製作過程中團隊的資訊共享至關重要，而版本控制則是協同設計的核心。我們借助 ODDO 中的產品生命週期管理功能，記錄產品製造過程中的每個步驟。如果在專案製作中出現錯誤，我們能夠快速定位問題並解決。我們希望透過實際製作一項產品，展現 ODDO PLM 在團隊協同上的應用。

製造業企業在當今競爭激烈的市場環境中面臨著各種挑戰，包括產品生命週期管理（PLM）、協同設計和生產流程優化等。隨著全球化和數位化的發展，企業需要更有效的方法來管理產品開發過程，以滿足客戶需求並保持競爭力。

之所以選擇鋼球平衡台，是因為其結合自動控制、機構學、計算機概論、電腦輔助設計 (CAD)、電腦輔助製造 (CAM) 等課程所學知識，並且具有相當的複雜度，能夠展現團隊的專業能力和協同合作的能力。

第三章　協同設計概述

所謂協同設計是指為了完成某一個目標，由數多名成員戮力同心以自身的專業知識和技能，共同努力創建、開發解決複雜的問題達到協同設計的理念。協同設計強調組員集體決策、創建創新且有效的解決方案，協同產品開發能讓製造商透過網路工具即時連線，為整個企業提供存取貢獻、查看和保護產品資料的能力，讓協同合作更加安全，為企業帶來更多利益。

在這網路發達的世代，一個企業要在這充滿競爭的環境下生存，協同設計在產品開發中被已廣泛使用，協同設計可以確保最終產品滿足使用者需求且與保持一致，最重要的是能夠客製化產品以滿足客戶的需求，產品開發中，透過協同產品的幫忙有助於有效解決問題、快速原型設計。它使設計師、工程師、行銷人員和其他相關利益相關者能夠共同工作，分享他們的專業知識，並共同為創新和成功產品的開發做出貢獻。

第四章　ODOO PLM 簡介

Odoo 產品生命週期管理 (PLM) 模組可以幫助企業對於產品的生命週期管理：從客戶需求 → 設計開發 → 產品測試 → 大量生產 → 產品維護 → 產品停產下架，PLM 模組中可以為這個工作項目指派負責人員以及完成所需時間，負責這個項目的人員就可以對進度進行提交、修改，或是提出時間延長讓項目有更充裕的時間完成，每次項目提交都會以歷史紀錄保存，方便團隊追溯產品的修改紀錄。

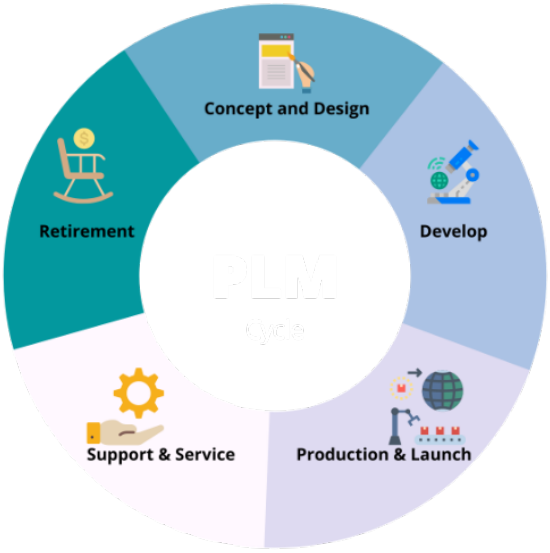


圖　4.1: PLM structure

第五章　ODOO PLM 在協同設計中的應用

在這章節中我們將使用 1D 系統的鋼球平衡台來展示 ODOO 中產品生命週期 (PLM) 的功能，首先我們來到 ODOO 主畫面並選取產品生命週期 (PLM)。

圖　5.1: ODOO 主畫面

進到 PLM 概覽之後選擇主資料選單中的產品。

圖　5.2: PLM 概覽

接下來按下新增產品進入到新增畫面

圖　5.3: 新增產品

我們以鋼球平衡台作為範例。

選擇物料清單來新增鋼球平衡台所需的零件。

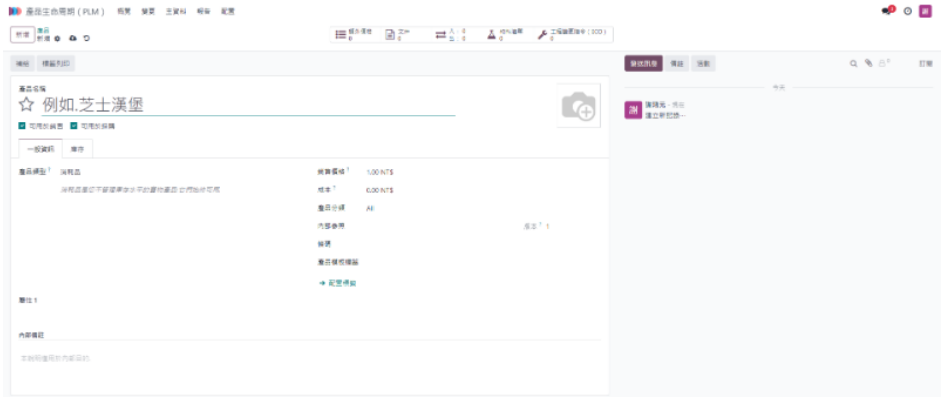


圖 5.4: 新增畫面

圖 5.5: 鋼球平衡台

圖 5.6: 物料清單

將所需零件加入後，這些零組件會自動出現在剛才提到的產品中。

圖 5.7: 產品列表

將產品設定完後回到 PLM 概覽並選取新產品介紹下方的工程變更。

圖 5.8: 工程變更選項

在這個工程變更指令頁面中我們選擇要製作的產品和物料清單，並且指派工作給各單位組員也可以設定完成期限或留下備註。

圖 5.9: 工程變更指令頁面

當我們設定完成後製作鋼球平衡台這項任務就會出現在頁面上。

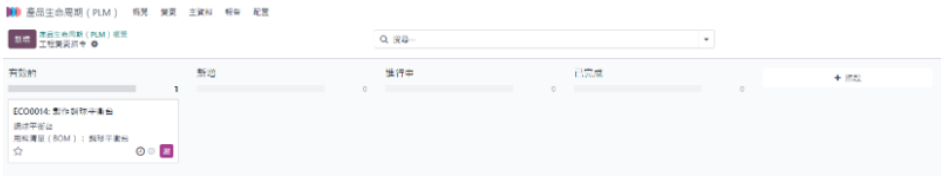


圖 5.10: 工程變更指令總覽

團隊中的主管可以藉由拖曳將圖塊任務移到相對應的狀態底下，假如專案已完成，主管可將圖塊移到已完成區域，這些狀態可依情形不同做修改或增加。

圖 5.11: 已完成之項目

若想更改用料清單可以使用主頁面中用料清單 (BOM) 更新的功能，使用方法與建立新產品雷同此處就不多贅述。

第六章　案例研究：鋼球平衡台的設計

在鋼球平衡台中我們會用到兩種不同領域的理論，數學系統模型以牛頓力學推導運動方程式後使用拉氏轉換將時域轉變成頻域，而另外一項就是自動控制中常見的 PID 控制器。

6.1 數學系統模型

球體的動態是由物理定律推導出，以微分方程式來表達，我們將使用牛頓力學來得到球的運動方程式，並使用拉氏轉換解之。

6.1.1 簡化與假設

為了得到球在平板上的運動方程式我們需假設球的幾何型態是完全球形且均質、球在平台上只在 X 方向移動、球在平台上只做滾動無滑動並且不考慮摩擦力。

6.1.2 運動方程式

球的絕對加速度方程式由參考書籍 [後期填入] 得到。

$$\mathbf{a}_a = \dot{\omega} \times \mathbf{r} + \omega \times (\omega \times \mathbf{r}) + 2\omega \times \mathbf{v}_{\text{rel}} + \mathbf{a}_{\text{rel}} \tag{6.1}$$

接下來我們將 6.1 式改寫為6.2，式中 $\mathbf{e}_{k1}$ 和 $\mathbf{e}_{i1}$ 代表單位向量，x_p 代表球相對於座標系的位置，α_1 代表平台的傾角。

$$\mathbf{a}_1 = \ddot{\alpha_1}\mathbf{e}_{k1} \times x_p\mathbf{e}_{i1} + \dot{\alpha_1}\mathbf{e}_{k1} \times (\dot{\alpha_1}\mathbf{e}_{k1} \times x_p\mathbf{e}_{i1}) + 2\dot{\alpha_1}\mathbf{e}_{k1} \times \dot{x_p}\mathbf{e}_{i1} + \ddot{x_p}\mathbf{e}_{i1} \tag{6.2}$$

將6.2經過簡化整理後得到

$$\mathbf{a}_1 = \left(\ddot{x}_p - x_p\dot{\alpha_1}^2\right)\mathbf{e}_{i1} + \left(x_p\ddot{\alpha}_1 + 2\dot{\alpha}_1\dot{x}_p\right)\mathbf{e}_{j1} \tag{6.3}$$

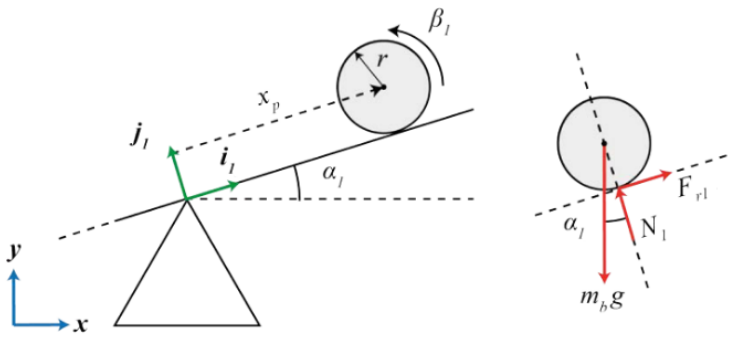


圖 6.1: 圖 6-1

在圖6.1的自由體圖中，從力矩的平衡可以看出球的剩餘力。

$$I_b\ddot{\beta}_1 = F_{r1}r \tag{6.4}$$

I_b 是球的質量慣性矩，β_1 是球相對於其初始位置在平台中心的角度，r 是球的半徑，F_r 是來自平台對球的作用力，我們假設求在平台上並無滑動所以我們可以根據位置定義相對角度 β。

$$\beta_1 = -\frac{x_p}{r} \tag{6.5}$$

為了求解6.4中的 F_r，我們將6.5式的二次時間導數代入6.4式中得到6.6式。

$$F_r = -\frac{I_b \ddot{x_p}}{r^2} \tag{6.6}$$

球在平台上受到的力和平台對球施加的力之間的平衡，由6.3式中的加速度和6.6中的力導致，由此得到動態系統的運動方程式。

$$\left(\frac{I_b}{r^2} + m_b\right)\ddot{x_p} + m_b g \sin\alpha_1 - m_b x_p \dot{\alpha_1}^2 = 0 \tag{6.7}$$

為了做拉式轉換我們稍微整理方程式。

$$\ddot{x} = \frac{m_b r_b^2 \left(x_p \dot{\alpha}_1^2 - g\sin\alpha_1\right)}{m_b r_b^2 + I_b} \tag{6.8}$$

接下來我們在 $X_p = 0$,$\alpha_1 = 0$ 對6.8式作線性化。

$$\ddot{x} = \frac{m_b g \alpha_1 r^2}{m_b {r_b}^2 + I_b} \tag{6.9}$$

當 α_1 出現小變動時線性化可得6.9式。接下來當我們將 I_b 也就是球體的質量慣性矩代入我們可以得到6.10，我們可以觀察到該系統的運動方程式和該球體的半徑和質量無關。

$$\ddot{x} = \frac{5}{7} g \alpha_1 \tag{6.10}$$

最後我們對6.10作拉式轉換得到6.11

$$s^2X = \frac{5}{7}gA_1 \tag{6.11}$$

在推導運動運動方程式後我們得到了平台角度和球的關係式，接下來我們利用 geogebra 進行模擬得到馬達轉角和平台角度的關係圖(圖6.2)。

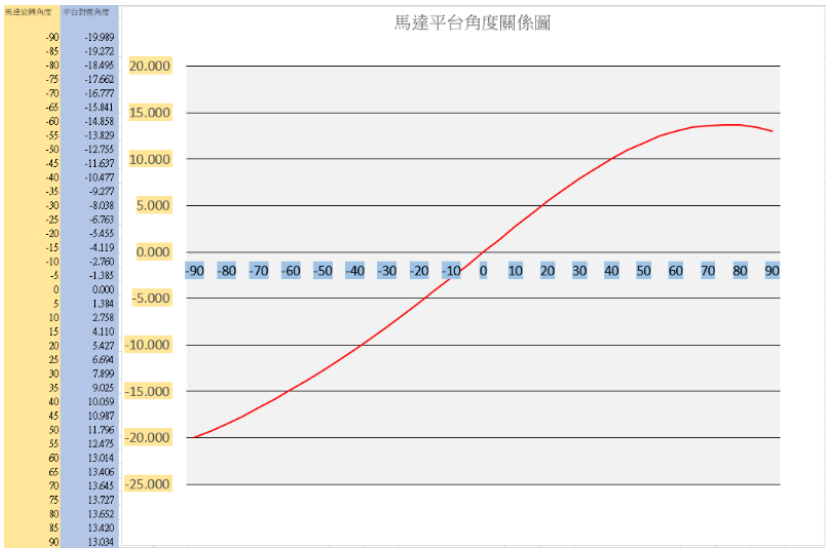


圖 6.2: 馬達轉角和平台角度關係圖

由 (圖6.2) 可觀察到，超過正 70 度的部分由於桿件設計反而造成平台角度下降，故將該段行程捨棄，利用 python 生成擬合曲線 (圖6.3)，得到馬達轉角和平台的關係式6.12。

$$\alpha = 0.233371\theta - 0.293753 \tag{6.12}$$

得到方程式後做線性化並拉式轉換就能透過6.11和6.12獲得整體系統轉移函數。

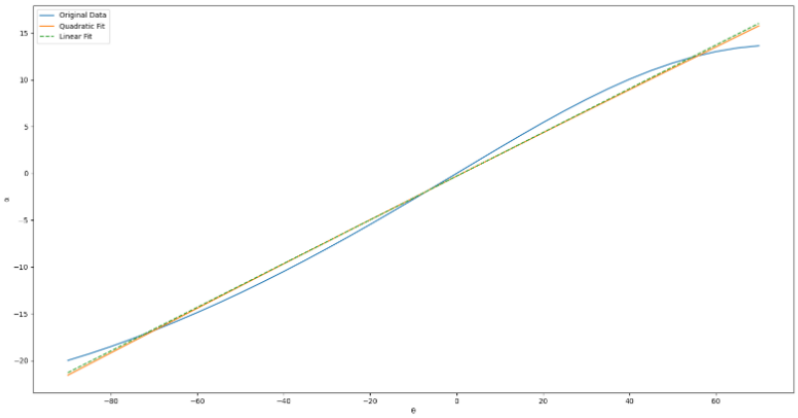

Original Data
Quadratic Fit
Linear Fit
15
10
5
0
−5
−10
−15
−20
−80
−60
−40
−20
0
20
40
60

圖 6.3: 擬合曲線

6.2 一維系統架構

6.2.1 設計理念

我們以鋼球平衡台作為專題的主體，然後寫程式驅動雷射測距感測器當鋼球遠離時 platform，當鋼球靠近時 platform 放下，重複此動作直至鋼球平衡台平衡。

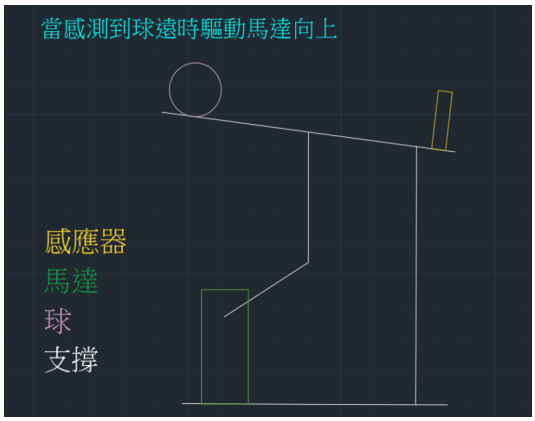


圖 6.4: 2D 草圖 (1)

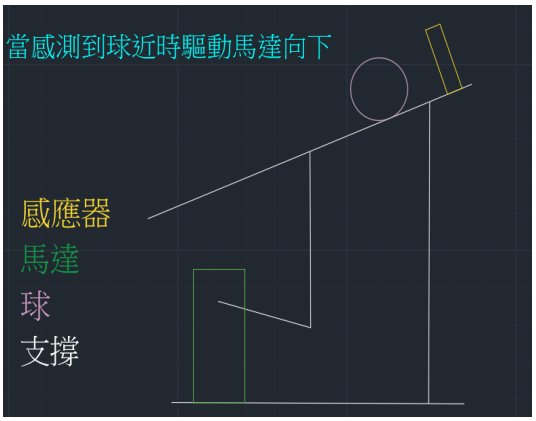


圖 6.5: 2D 草圖 (2)

6.2.2 馬達角度所對應之平台角度關係

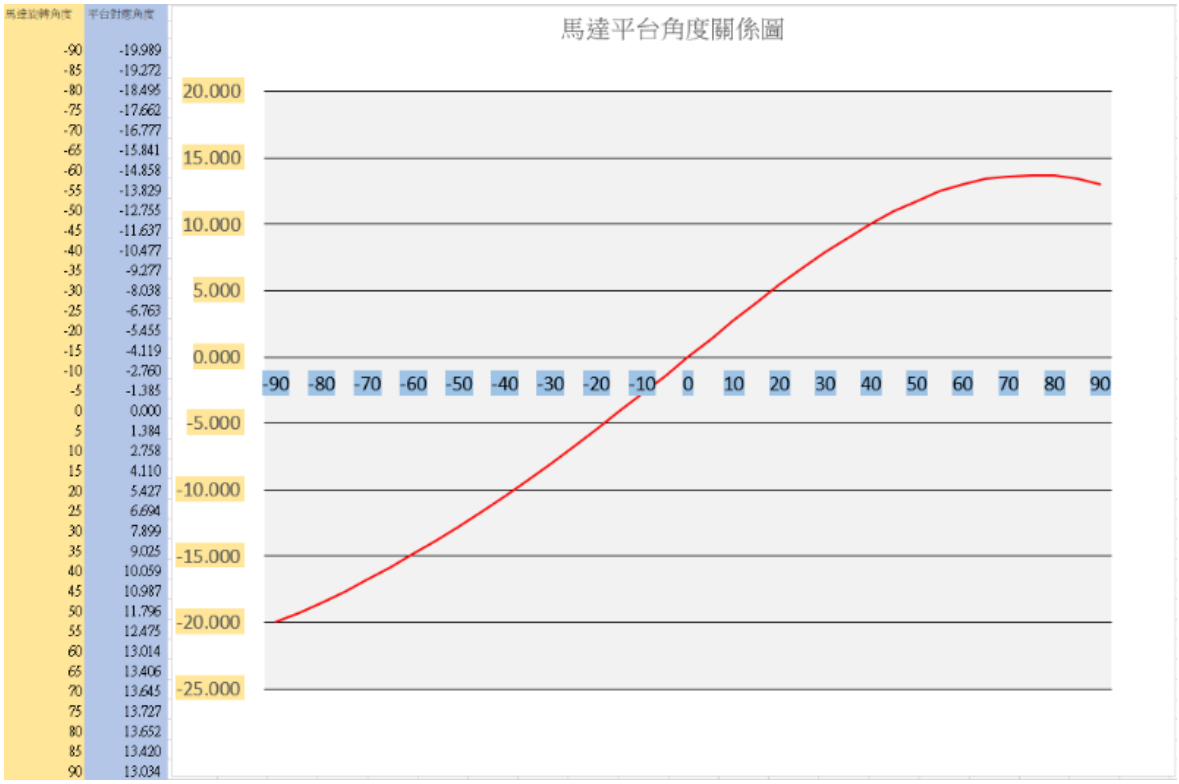


圖 6.6: 馬達角度關係圖

使用 SolidWorks 2023 進行繪圖。

圖 6.7: SOLIDWORKS

6.2.3 platform

第一版本鋼球平衡台的 platform 軌道長度為 200mm 整體的寬度為 30mm 並給定深度填料 11mm。

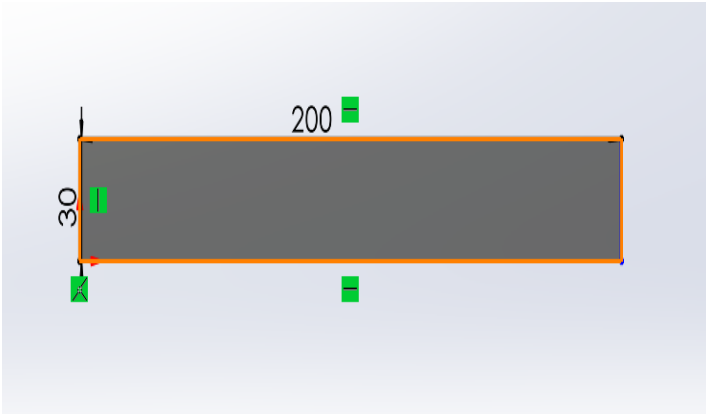


圖 6.8: PLATFORM 草圖 (1)

圖 6.9: 編輯特徵 (1)

軌道上方寬度為 8.5mm，下方為 7.2mm，深 4mm 並將紅圈處深伸長除料選擇完全貫穿。

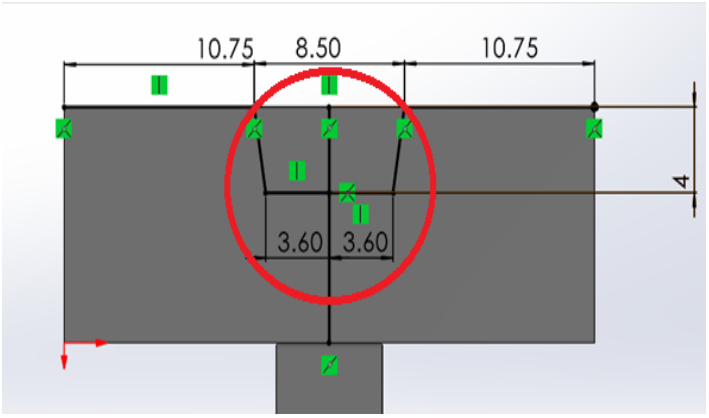


圖 6.10: PLATFORM 草圖 (2)

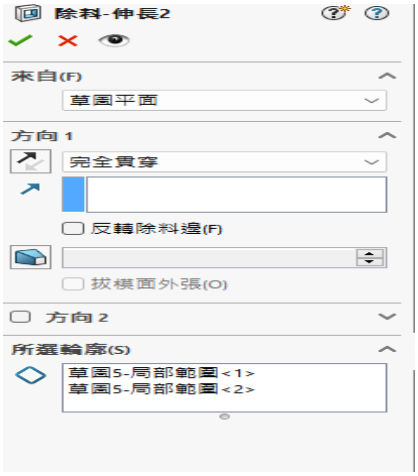


圖 6.11: 編輯特徵 (2)

下方配合處長 30mm 寬 6mm，繪製好圖形草圖。

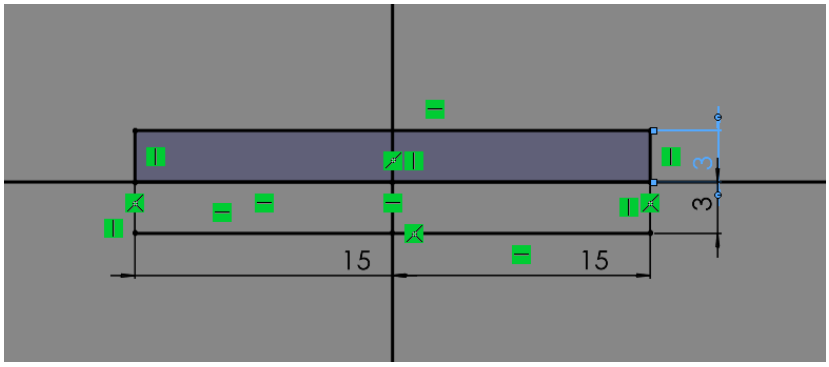


圖 6.12: PLATFORM 草圖 (3)

給予尺寸後伸長填料 25mm。

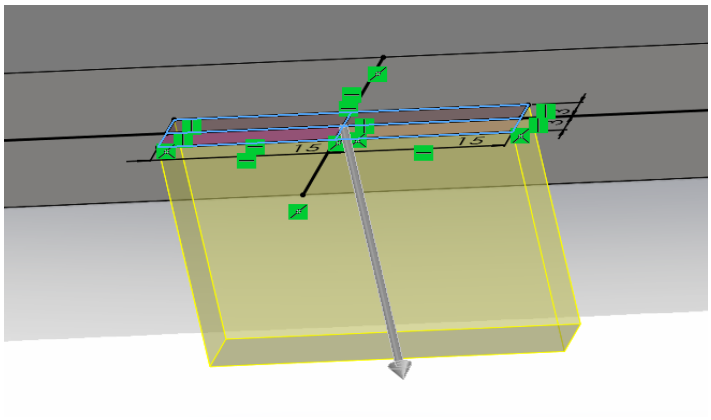


圖 6.13: PLATFORM 草圖 (4)

圖 6.14: 編輯特徵 (3)

填料完成後再圖形上畫一個 2.9mm 的小孔並進行伸長除料以便與其他零件配合。

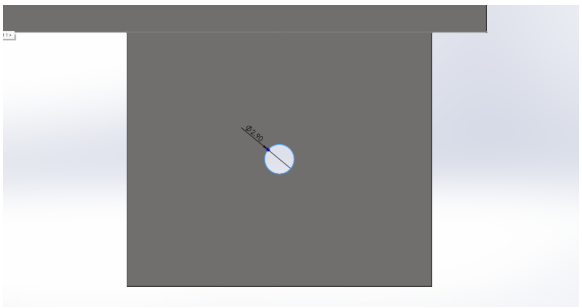


圖 6.15: PLATFORM 草圖 (5)

第一版 platform 完成圖。

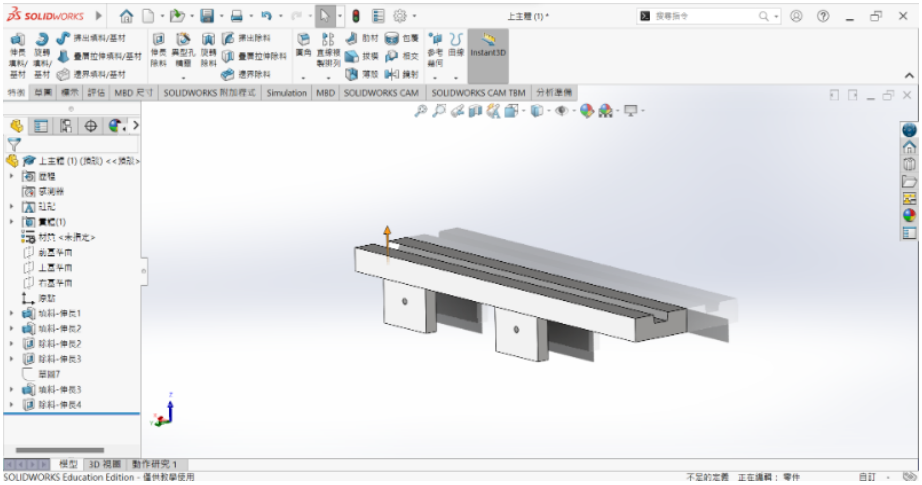


圖 6.16: PLATFORM 零件圖

修改部分

軌道上方增加長 26.2mm 寬 2mm 的貫穿凹槽，用於放置感應器。

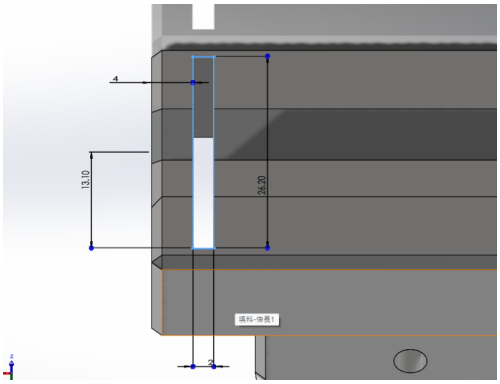


圖 6.17: 修改 PLATFORM 草圖 (1)

下方接合處新增 R10 圓角。

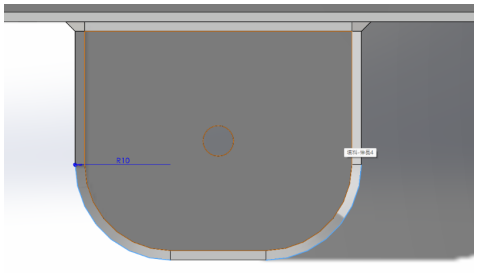


圖 6.18: 修改 PLATFORM 草圖 (2)

最終 Platform 零件圖。

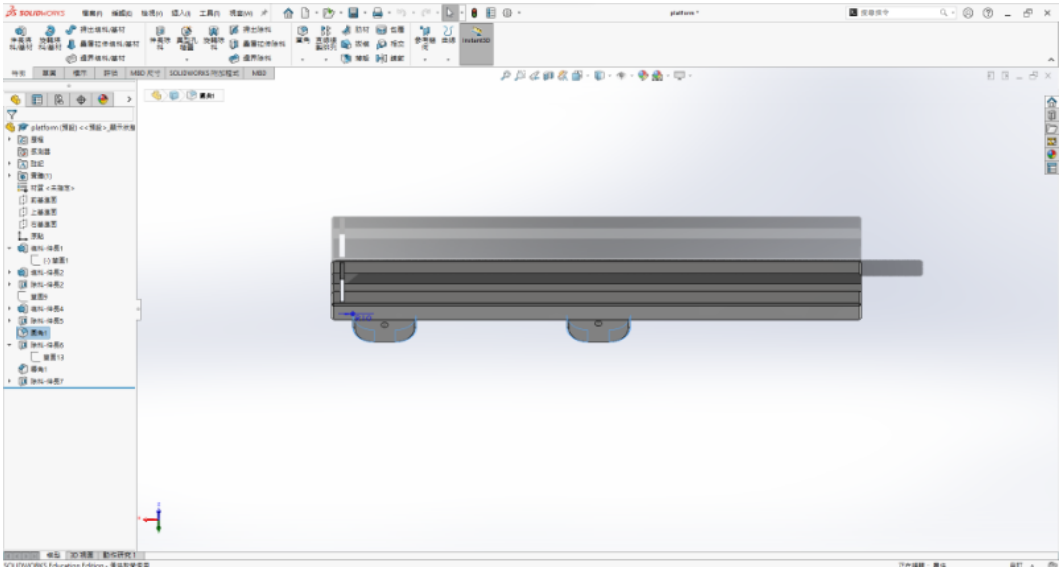


圖 6.19: 最終 PLATFORM 零件圖

3D 列印成果

圖 6.20: PLATFORM 3D 列印完成圖

6.2.4 base

第一版 Base 底的長為 237mm 寬為 150mm。

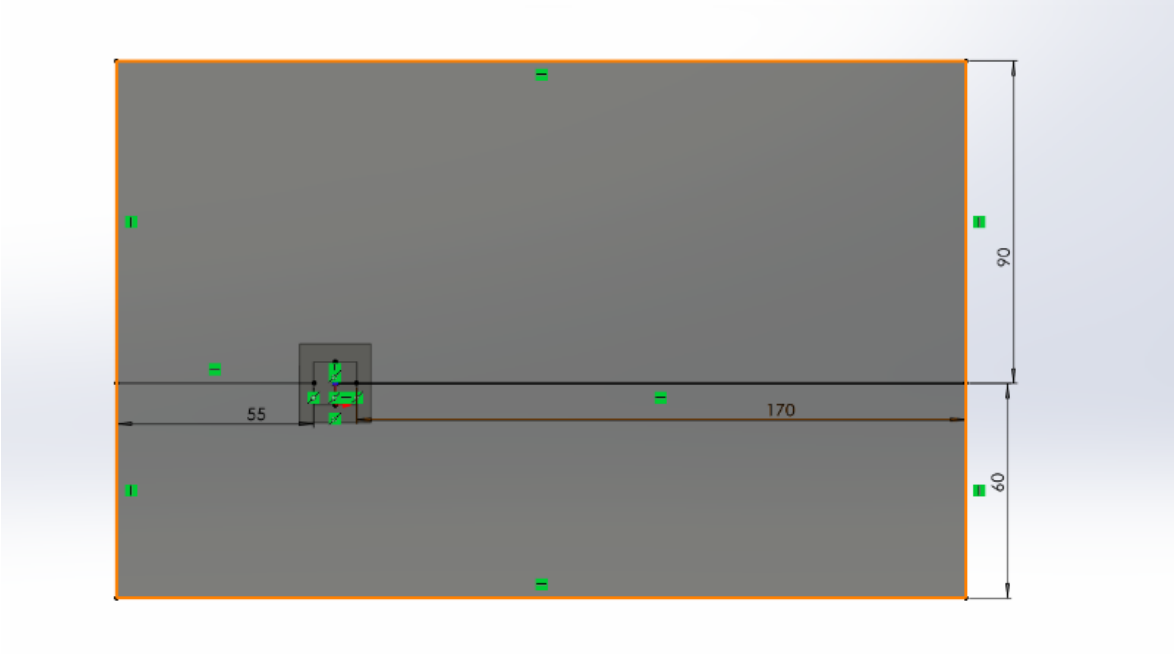


圖 6.21: BASE 草圖 (1)

在底板長 55mm 寬 54mm 處繪製一個 12mm × 12mm 的方形柱並向上填料 100mm。

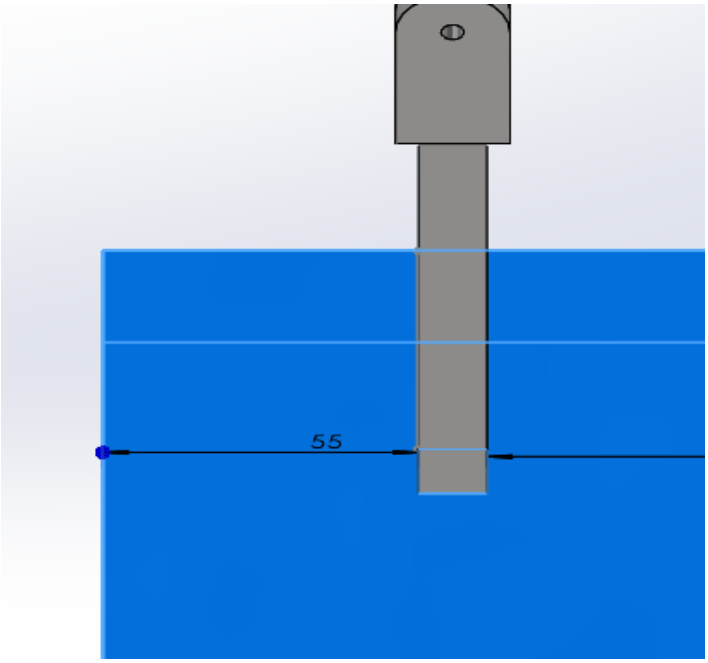


圖 6.22: BASE 草圖 (2)

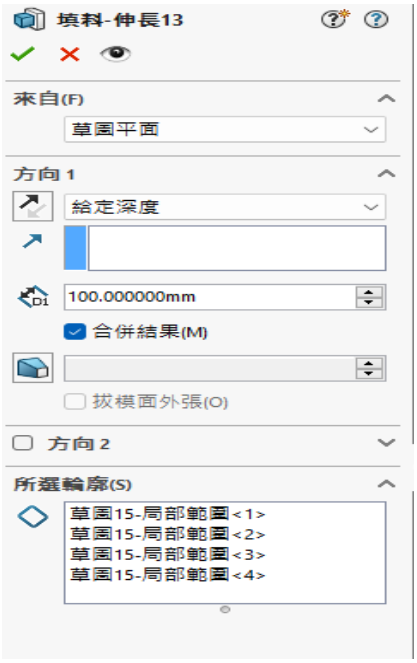


圖 6.23: 編輯特徵 (4)

在方柱上方繪製一個長 30.28mm 寬 20 的長方體然後在長方體上畫直徑 20mm 的半圓最後在圓的中心繪製一個 3.98mm 的小孔最後在長方體上畫一個長 30mm 寬 10.8mm 的小長方體伸長除料選擇完全貫穿方便與上方 platform 配合。

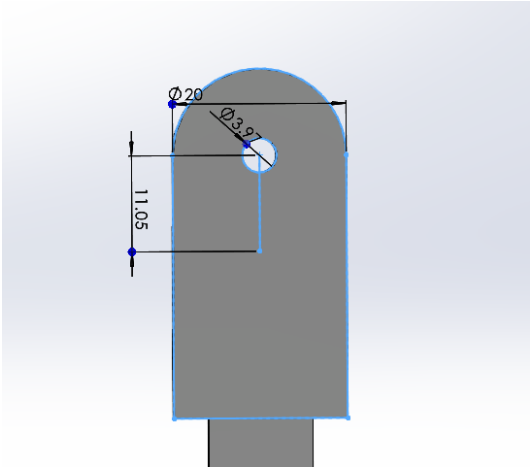


圖 6.24: BASE 草圖 (3)

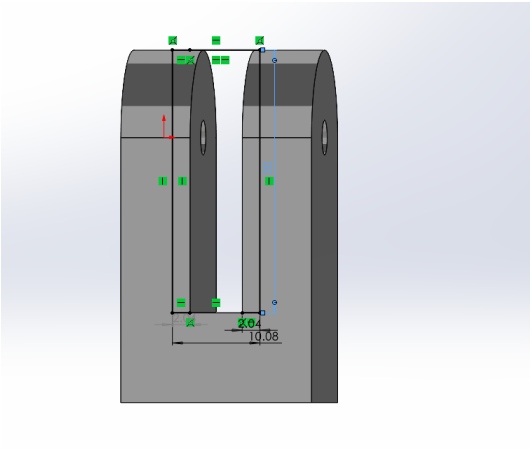


圖 6.25: BASE 草圖 (4)

在距離方柱中心長 129mm 寬 25mm 處繪製一個長 31mm 寬 20mm 向上填料 7mm 的小平台用來定位馬達。

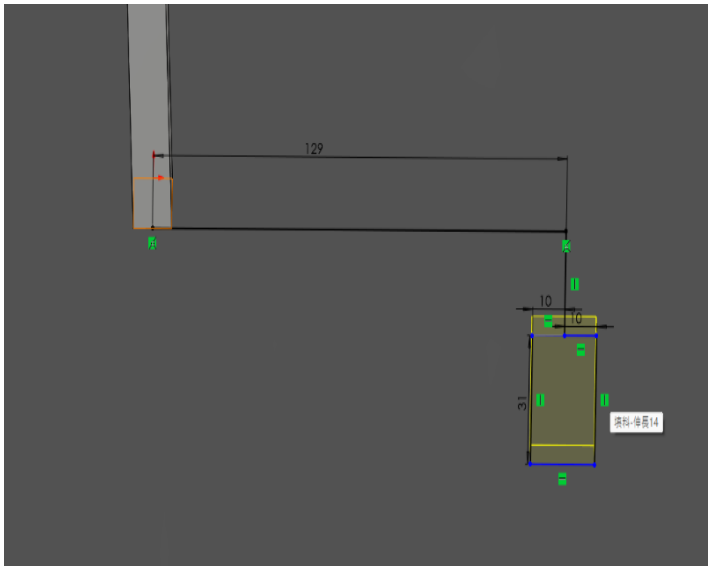


圖 6.26: BASE 草圖 (5)

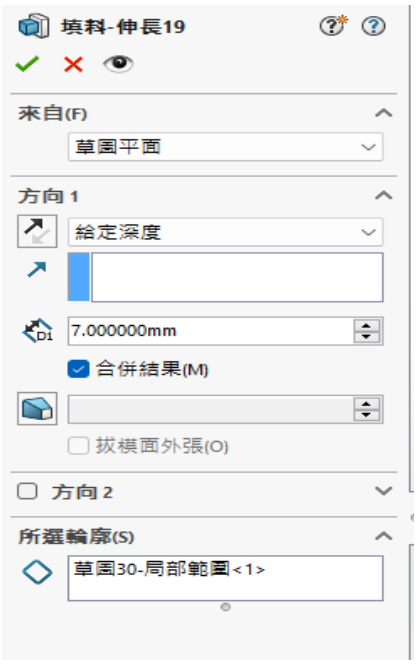


圖 6.27: 編輯特徵 (5)

並且在兩邊加畫底 15mm 高 45mm 的三角形支撐架防止馬達晃動。

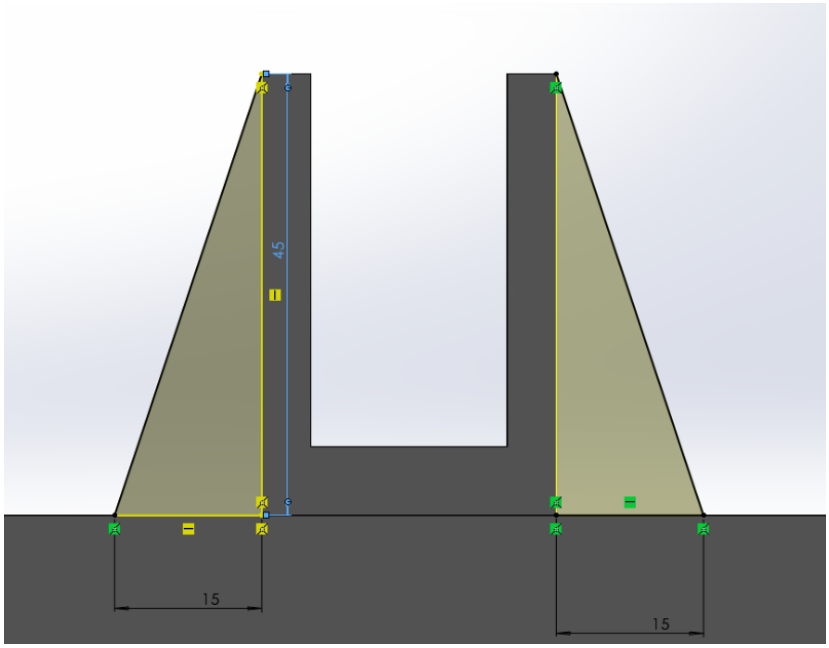


圖 6.28: BASE 草圖 (6)

第一版 base 完成圖。

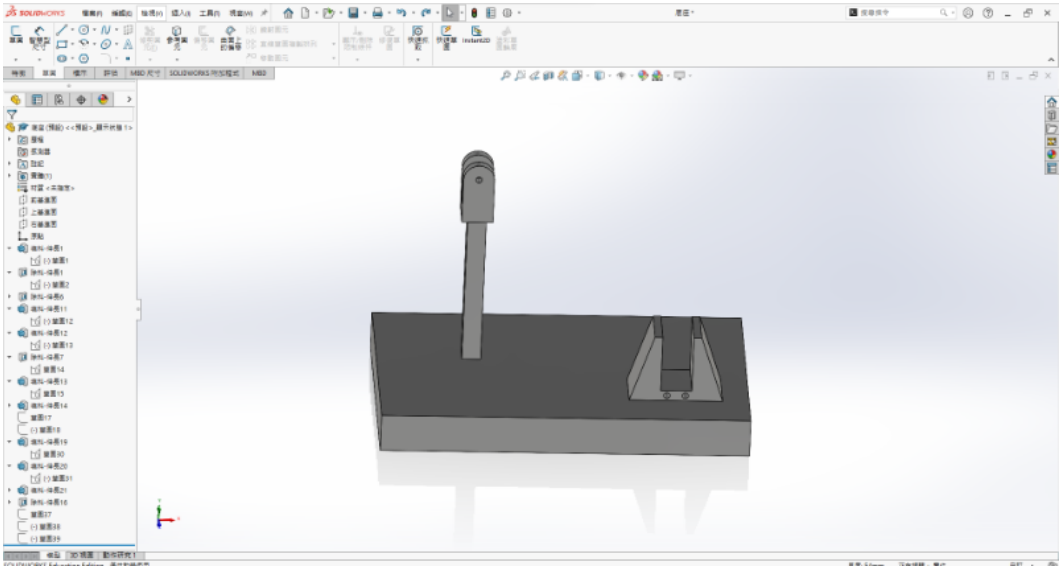


圖 6.29: BASE 零件圖

修改部分

底部去除浪費的部分改以長 165.6mm 圓半徑 22.35mm 的直狹槽代替。

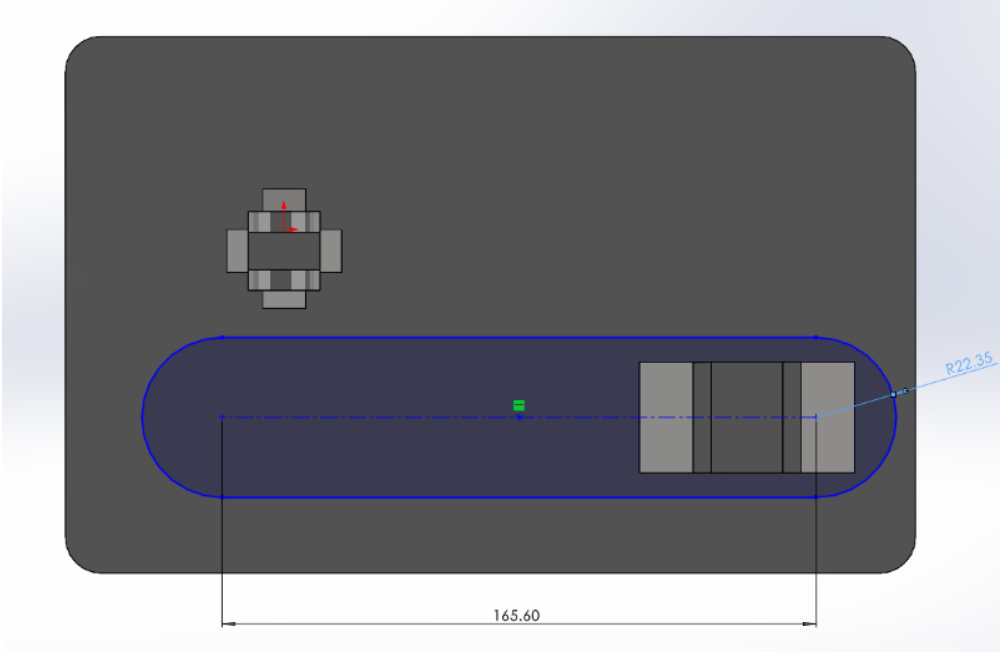


圖 6.30: 修改 BASE 草圖 (1)

為了好收納將左方柱子拔除留下一凹槽方便後續配合及螺絲孔。

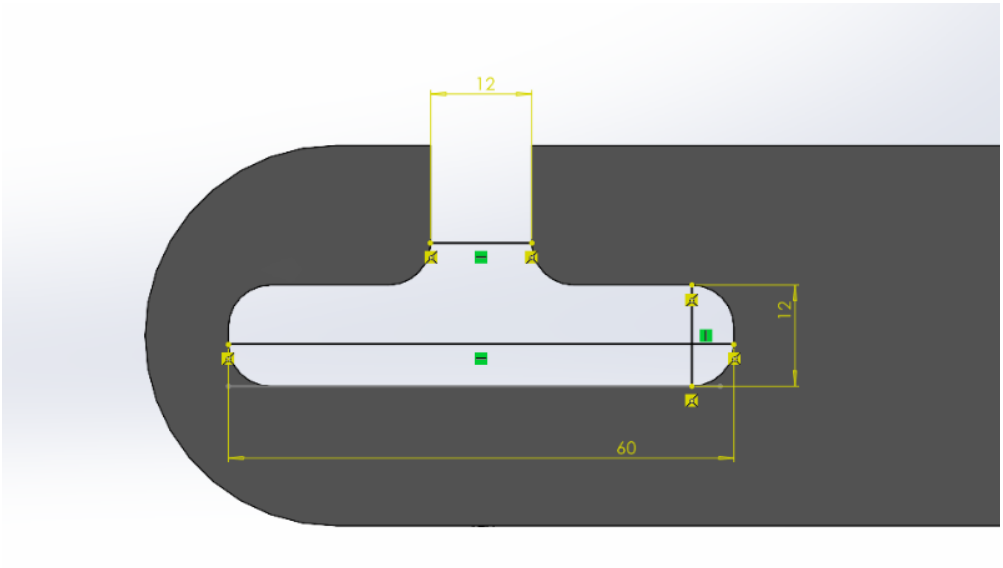


圖 6.31: 修改 BASE 草圖 (2)

最終 base 完成圖。

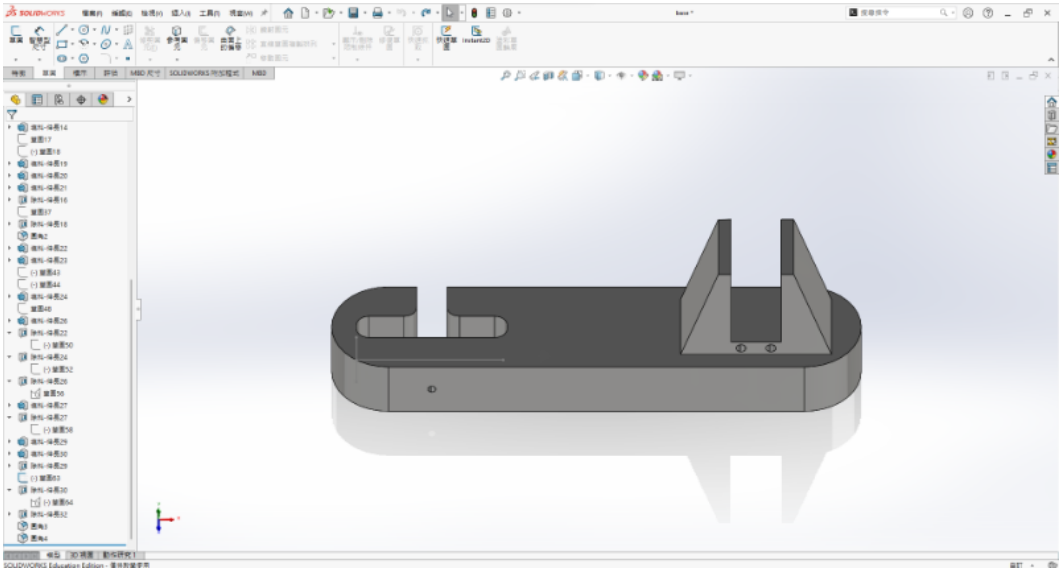


圖 6.32: 最終 BASE 零件圖

3D 列印成果

圖 6.33: BASE 3D 列印完成圖

6.2.5 support

從底座拔除的部分目的好收納尺寸都沒有改變。最終 support 完成圖。

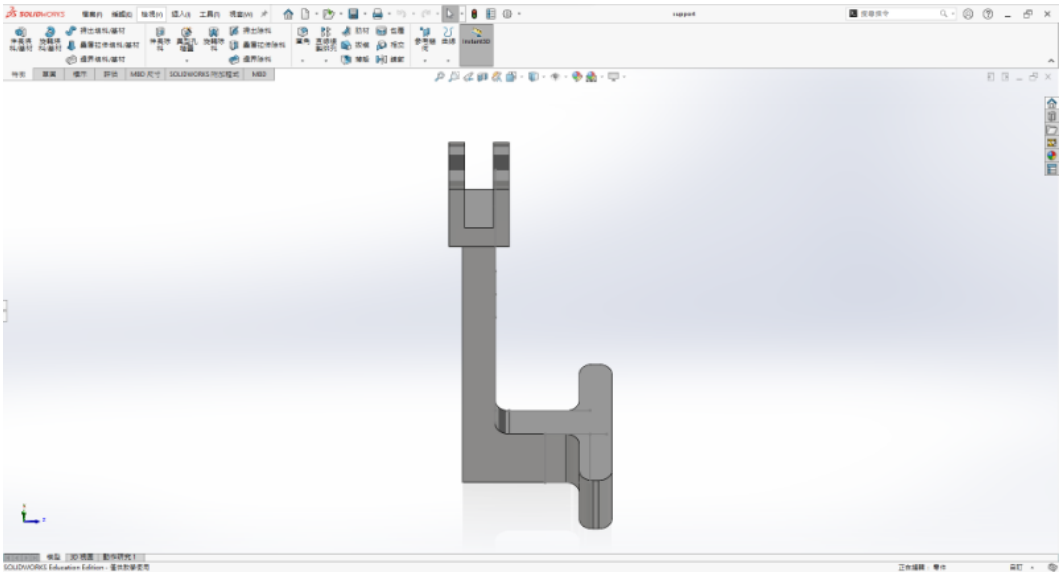


圖 6.34: 最終 SUPPORT 零件圖

3D 列印成果

圖 6.35: SUPPORT 3D 列印完成圖

6.2.6 link

上方連結處與 support 尺寸一致皆為長 30.28mm 寬 20 的長方體然後在長方體上畫直徑 20mm 的半圓最後在圓的中心繪製一個 3.98mm 的小孔最後在長方體上畫一個長 30mm 寬 10.8mm 的小長方體伸長除料選擇完全貫穿方便與上方 platform 配合。

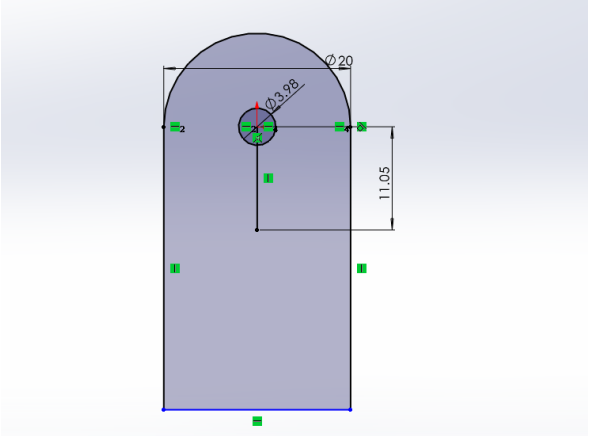


圖 6.36: LINK 草圖 (1)

下方為 12mm × 12mm 的方柱並填料 68mm 方便與馬達進行配合。

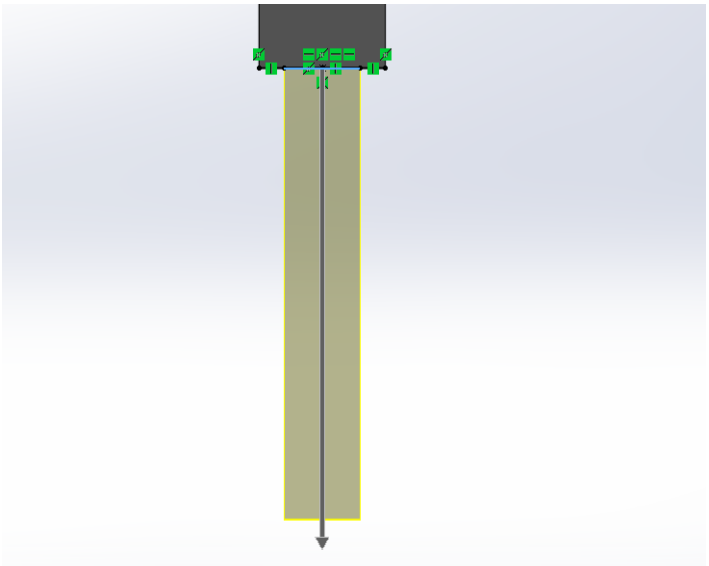


圖 6.37: LINK 草圖 (2)

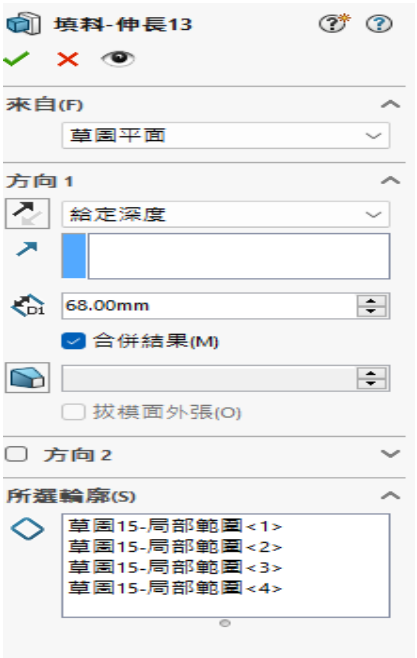


圖 6.38: 編輯特徵 (6)

最終 link 完成圖。

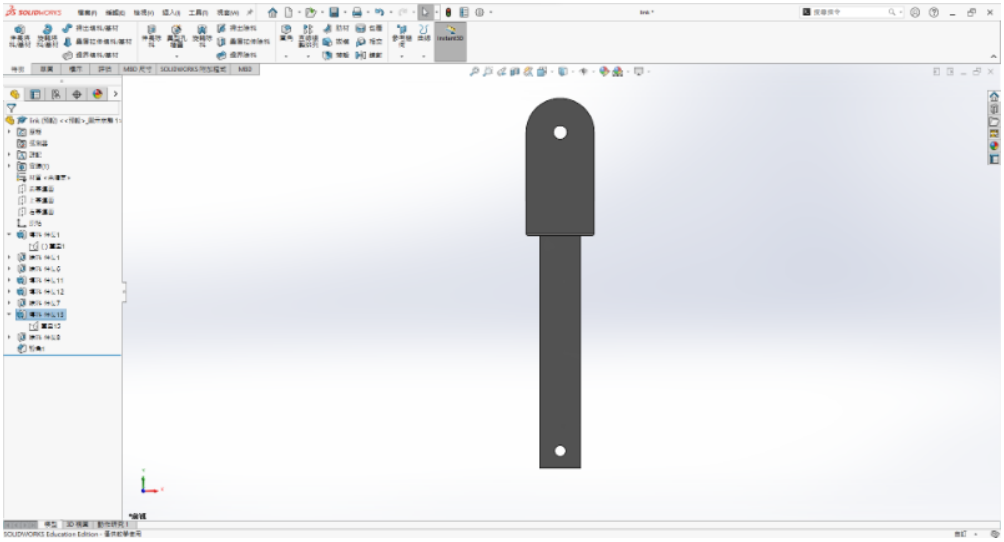


圖 6.39: 最終 LINK 零件圖

3D 列印成果

圖 6.40: LINK 3D 列印完成圖

6.2.7 crank

畫一個長 36mm 寬 5mm 的 crank 填料 3mm 並留兩個 3.2mm 的圓孔方便鎖上螺栓配合 base 與 link

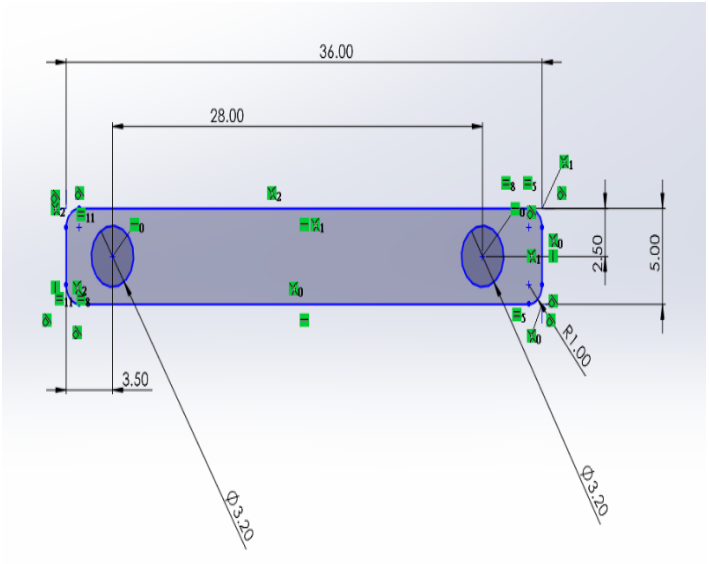


圖 6.41: crank 草圖 (1)

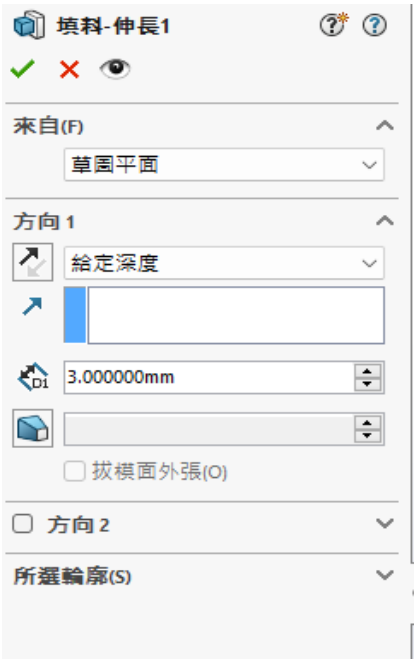


圖 6.42: 編輯特徵 (7)

最終 crank 完成圖

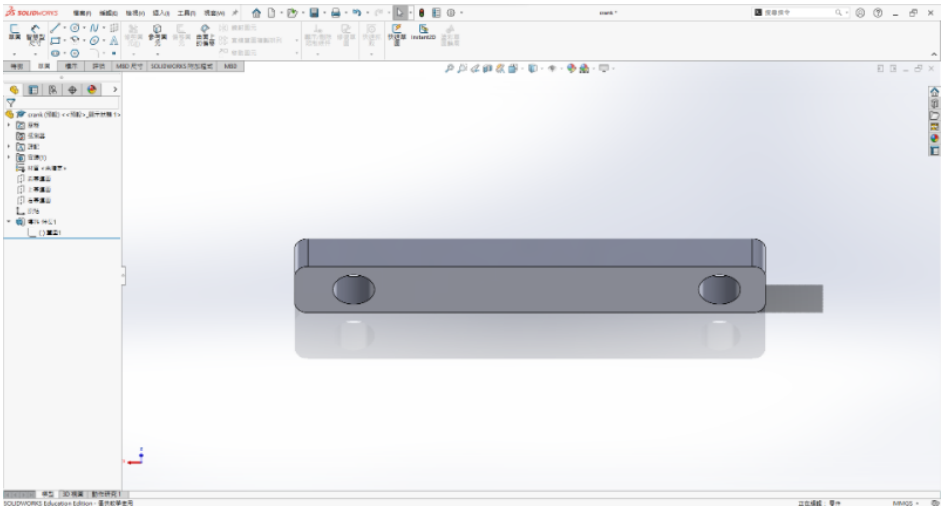


圖 6.43: 最終 crank 零件圖

6.2.8 baffle

繪製一個配合 platform，兩兩相組合的檔塊，透過螺絲逼緊的方式夾緊於 platform，保留可調整位置、可拆除的特性，以輔助調整控制參數，如此一來當參數錯誤時，球不會直接飛離平台。而上方的拱型則是避免干擾到雷射感測器的訊號。

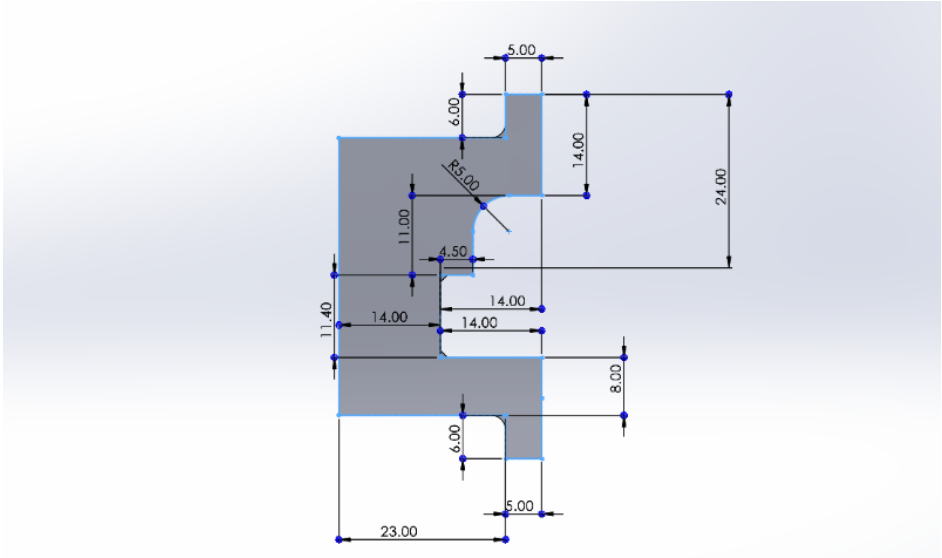


圖 6.44: baffle 草圖 (1)

最終 baffle 完成圖

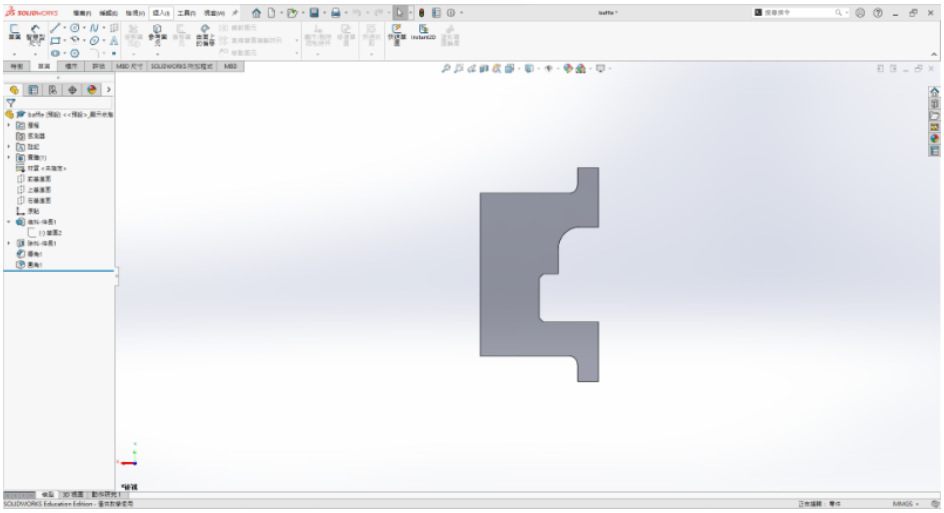


圖 6.45: 最終 baffle 零件圖

6.2.9 assemble

組合完成圖。

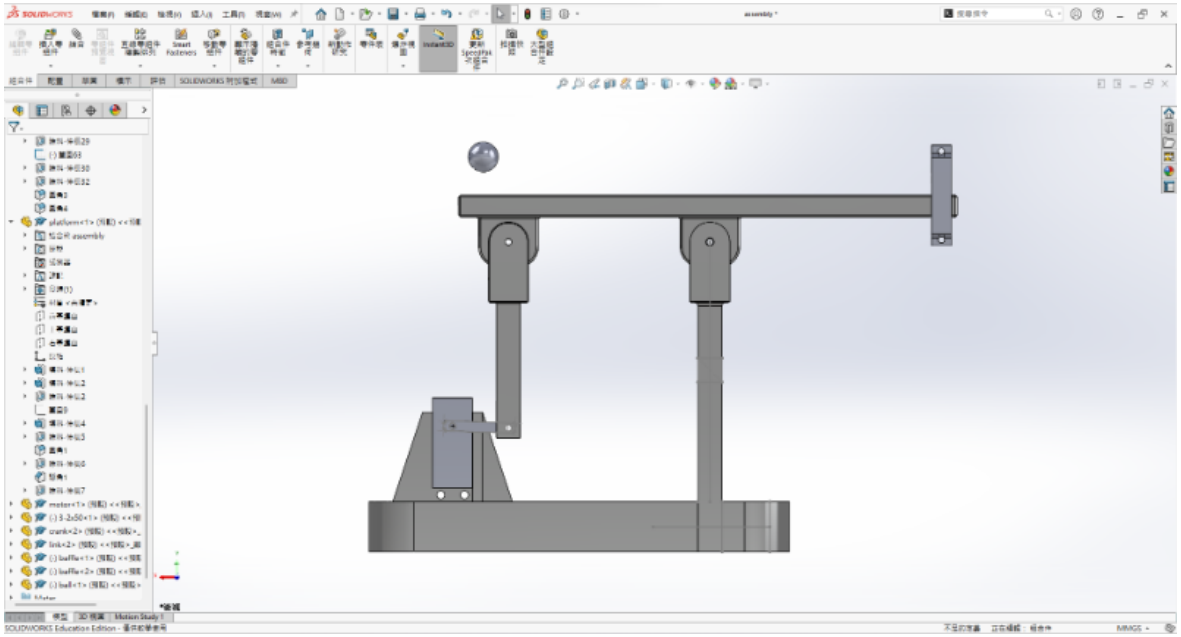


圖 6.46: ASSEMBLE 組合圖 (1)

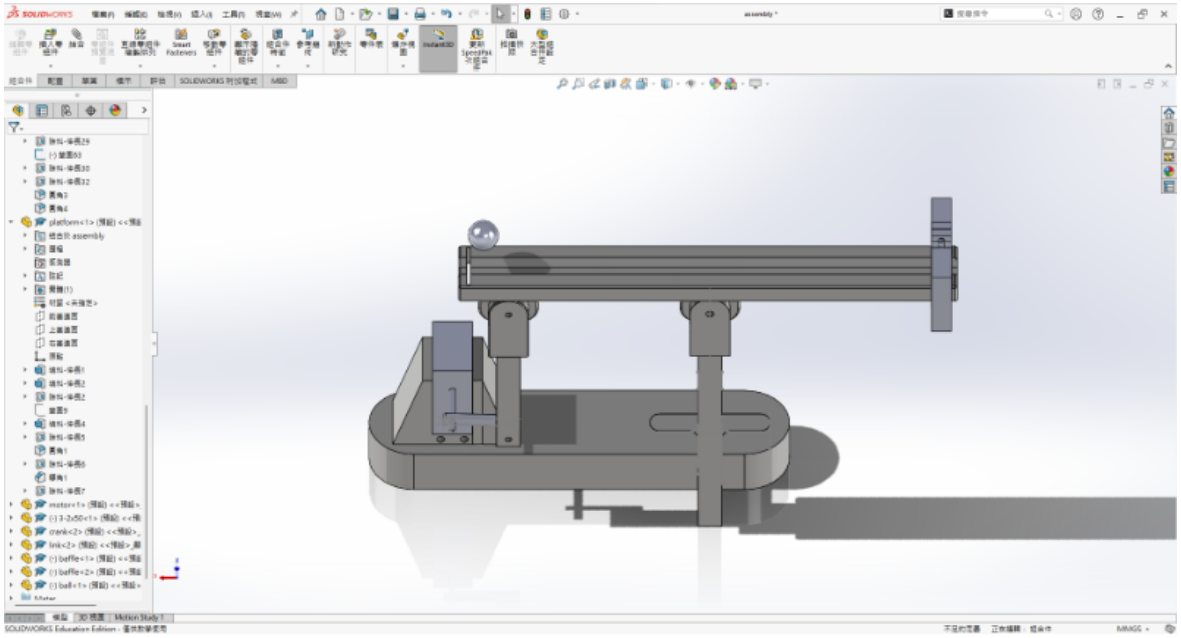


圖 6.47: ASSEMBLE 組合圖 (2)

6.2.10 驅動方式

使用金屬齒輪伺服馬達配合程式控制平台，程式放置於 6-3。

6.3 控制系統設計與結果

最終的實體作品，我們選用 Arduino UNO 作為開發板，VL53L0 作為傳感器輸入，MG 996R 伺服馬達進行輸出。

6.3.1 控制系統

Arduino 板的控制程式如下:

導入函式庫

```
#include <Wire.h>        // 包含 I2C 通信的函式庫
#include <VL53L0X.h>     // 包含 VL53L0X 距離感測器的函式庫
#include <Servo.h>       // 包含控制伺服馬達的函式庫
```

宣告物件和常數

```
// 宣告物件
VL53L0X sensor; // 宣告一個 VL53L0X 類別的物件，稱為 sensor
Servo motor;    // 宣告一個 Servo 類別的物件，稱為 motor

// PID 控制常數
const float kp = 4.0;  // 比例增益
const float ki = 2;    // 積分增益
const float kd = 3.0;  // 微分增益
const float tt = 1000; // 時間延遲常數，單位為毫秒

// 初始化誤差和積分項
float error_sum = 0.0;          // 誤差積分項的初始值
float last_error = 0.0;         // 上一次的誤差
unsigned long last_time = 0;    // 上一次的時間
```

初始化設定

```
void setup() {
  // 設置串口通信速率
  Serial.begin(115200);  // 設置串口通信速率為115200
  Wire.begin();          // 初始化 I2C 通信

  // 初始化感測器
  sensor.setTimeout(500);  // 設置感測器超時時間為500毫秒
  if (!sensor.init()) {    // 初始化感測器，如果失敗則進入循環
    Serial.println("Failed to detect and initialize sensor!");
    while (1);  // 如果感測器初始化失敗，停止程式
  }
  sensor.startContinuous();  // 開始連續測量

  // 初始化伺服馬達
  motor.attach(9);  // 連接馬達到9號引腳
  motor.write(90);  // 將馬達設置到90度，即中間位置
  delay(tt);        // 延遲一段時間以確保馬達回到中間位置

  last_time = micros();  // 初始化時間
}
```

獲取距離值

```
float getDistance() {
  float distance = sensor.readRangeContinuousMillimeters(); // 讀取距離，單位毫米
  if (sensor.timeoutOccurred()) {
    Serial.print("Timeout");  // 如果超時，輸出提示
    return -1;  // 返回 -1 表示發生超時
  } else {
    // 限制距離範圍在35到300毫米之間
    if (distance < 35) {
      distance = 35;
    } else if (distance > 300) {
      distance = 300;
    }
    return distance;  // 返回有效的距離值
  }
}
```

控制馬達

```
void controlMotor(float target_distance, float dt) {
  float distance = getDistance();  // 獲取距離
  if (distance > 0) {  // 如果距離值有效
    // 輸出目標距離和實際距離
    Serial.print("Target: ");
    Serial.print(target_distance);
    Serial.print(" mm Distance: ");
    Serial.print(distance);

    // 計算誤差
    float error = target_distance - distance;
    Serial.print(" Error: ");  // 輸出誤差
    Serial.print(error);

    // 更新誤差總和
    error_sum += error * dt;

    // 防止積分項積累過多
    error_sum = constrain(error_sum, -100, 100);

    // 計算微分項
    float derivative = (error - last_error) / dt;

    // 計算控制信號
    float control_signal = kp * error + ki * error_sum + kd * derivative;
    if (control_signal < 30) {  // 如果控制信號小於30，不使用積分項
      control_signal = kp * error + kd * derivative;
      Serial.print(" NO KI");
    }
    // 將控制信號映射到有效的角度範圍內
    int angle = constrain(map(control_signal, -1000, 1000, 180, 0), 0 , 180);
    Serial.print(" Control_signal: ");  // 輸出控制信號
    Serial.print(control_signal);
    Serial.print(" angle: ");  // 輸出馬達角度
    Serial.println(angle);
    motor.write(angle);  // 設置馬達角度

    // 更新上一次的誤差
    last_error = error;
  }
}
```

主迴圈

```
void loop() {
  // 檢查是否有串口輸入
  if (Serial.available() > 0) {
    char key = Serial.read();  // 讀取串口輸入
    if (key == 'q') {  // 如果收到 'q' 指令
      motor.detach();  // 停止馬達
      while (1);  // 停止程式
    }
  }

  // 計算時間差
  unsigned long current_time = micros();  // 獲取當前時間（微秒）
  float dt = (current_time - last_time) / 1000000.0; // 計算時間差，並將微秒轉換
      為秒
  last_time = current_time;  // 更新上一次的時間

  controlMotor(90, dt);  // 設定目標距離為 90 mm

  delay(100);  // 延遲以匹配時間步長 (dt)
}
```

第七章　評估與結果

7.1 結果呈現

7.1.1 Github Page

本組專題成果網頁係藉由藉由本組指導教授嚴家銘教授所開發的 cmsimde 子模組進行維護。

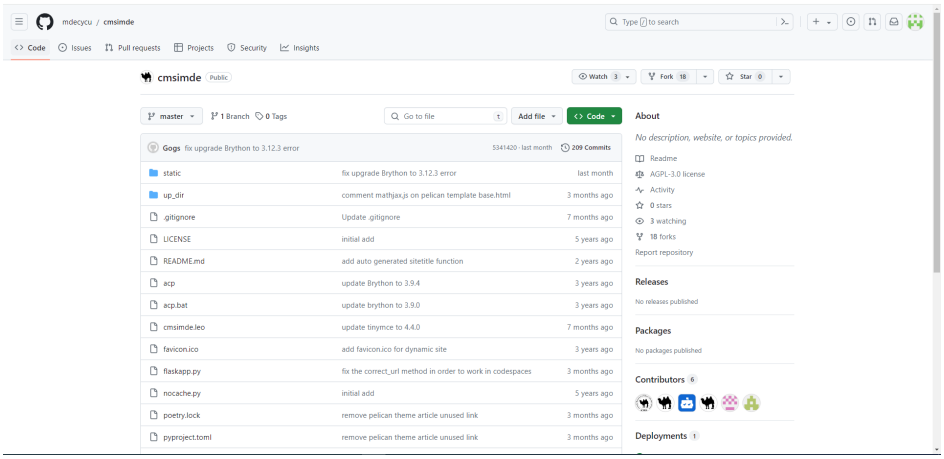


圖　7.1: cmsimde

組員只要在可攜環境的近端倉內使用終端機執行 cms.bat，就會以 python 運行 cmsimde 資料夾中的 wsgi.py，接著就會透過 flaskapp.py 等檔案啟動近端的動態網頁。組員即可透過瀏覽器編輯動態網頁，編輯完成後亦可透過編輯器內的選項將動態網頁轉換為靜態，並推送至 Github 倉儲中，由遠端倉儲生成網頁，網址為https://mde.tw/pj4101。

在此環境之下，本組組員能非常輕鬆的協同對網頁進行維護。

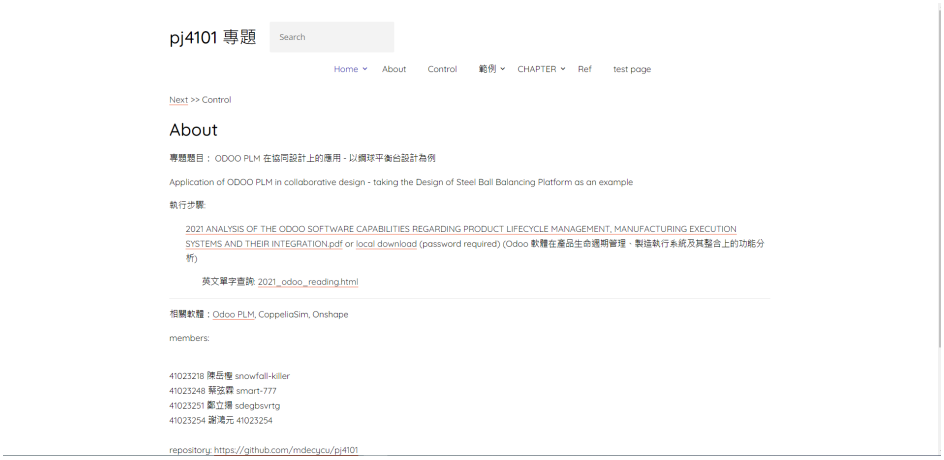


圖 7.2: 本組專題網頁

7.1.2 Repository

7.1.3

7.1.4

7.2 結果分析

7.2.1

7.2.2

7.2.3

7.3 討論

7.3.1

7.3.2

7.3.3

第八章　結論

在本次的專題中，我們使用 Odoo 作為產品設計的基底，並且結合 SolidWorks 和 Coppeliasim 設計零件及模擬，再加入 github 進行協同，最後使用 3D 列印列印零件加以組裝然後配合機電控制實現從零開始生產並且製造出一個產品。在這個專裡我們學到了每種程式不同功用，並使用各個程式的優缺點加以揉合後產生一加一大於二的功效。

然後我們在設計、仿真、協同、管理方面充分利用不同軟體平台的功能。首先我們使用 SolidWorks 來設計產品的尺寸及 3D 模型，導入 Coppeliasim 中進行組合後的仿真和其運動的虛擬測試，以確保產品能正常運動和使用。同時在 Odoo 中的產品生命週期管理 (PLM) 模組創建相對應的管理系統，以管理產品的生產、庫存和銷售，然後將過程在 github 裡面建立倉儲與其分支進行協同，最後 3D 列印出整個零件加入機電控制。這要求我們要具備跨越不同軟體平台的技術能力和整合能力，以確保各個方面的協調和一致性。

有了這些便捷的軟體後我們可以先在 SolidWorks 進行零件的初次設計，然後到 Coppeliasim 中進行確實的仿真模擬。在 Coppeliasim 我們可以模擬產品在不同條件下的行為和性能，並進行虛擬測試及後續的改進及優化。這使我們能夠更好地了解產品的現實狀況及日後所需改進的問題，並更有效地設計和製造產品。

我們認為 ODOO 不僅能定義產品的屬性還能根據產品的類型選定不同的模組進行使用，甚至還可以設置生產的流程，物料清單及生產的計畫等等，其中最厲害的莫過於產品生命週期管理 (PLM) 模組，能從需求到設計開發到產品測試到大量生產到產品維護再到產品停產下架完成一

整套的產品週期流程，可謂是從零到有甚至於再到產品的終結。

整個專題我們學到了如何獨當一面的設計和如何與其他成員協同分工，在這個過程中不僅學到了許多新的技能和知識，更體驗到了團隊合作和創新思維的重要性，這些在未來職場上班時都將會寶貴的經驗。

參 考 文 獻

[1] https://cn.comsol.com/multiphysics/fea-software?parent=finite-element-method-042-62-22

[2] https://www.guyuehome.com/37628

[3] https://hdl.handle.net/11296/b8emug

[4] https://forums.autodesk.com/autodesk/attachments/autodesk/915/18197/2/一种四足步行机器人结构设计与分析.pdf

[5] https://www.mdpi.com/2218-6581/12/1/28

[6] https://zh.wikipedia.org/zh-tw/有限元素法

[7] https://wiki.mbalib.com/zh-tw/有限单元法

[8] https://github.com/chaitravi-ce/Eklavya-QuadrupedMotionSimulation

[9] https://www.mdpi.com/2076-3417/11/9/3762

附錄

LaTeX

LaTex 為一種程式語言，支援標準庫 (Standard Libraries) 和外部程式庫 (External Libraries)，不過與一般程式語言不同的是，它可以直接表述 Tex 排版結構，類似於 PHP 之於 HTML 的概念。但是直接撰寫 LaTex 仍較複雜，因此可以藉由 Markdown 這種輕量的標註式語言先行完成文章，再交由 LaTex 排版。此專題報告採用編輯軟體為 LaTeX，綜合對比 Word 編輯方法，LaTeX 較為精準正確、更改、製作公式等，以便符合規範、製作。

表 1: 文字編輯軟體比較表

	相容性	直觀性	文件排版	數學公式	微調細部
LaTeX	√		√	√	√
Word		√			√

- 特點:

1. 相容性：以 Word 為例會有版本差異，使用較高版本編輯的文件可能無法以較低的版本開啟，且不同作業系統也有些許差異；相比 LaTeX 可以利用不同編譯器進行編譯，且為免費軟體也可移植至可攜系統內，可以搭配 Github 協同編譯。

2. 文件排版：許多規範都會要求使用特定版型，使用文字編譯環境較能準確符合規定之版型，且能夠大範圍的自定義排定所需格式，並能不受之後更改而整體格式變形。

3. 數學公式呈現：LaTex 可以直接利用本身多元的模組套件加入、編輯數學公式，在數學推導過程能夠快速的輸入自己需要的內容即可。

4. 細部調整：在大型論文、報告中有多項文字、圖片、表格，需要調整細部時，要在好幾頁中找尋，而 LaTeX 可以分段章節進行編譯，再進行合併處理大章節。

足端軌跡

利用 GeoGebra 軟體求得各種足端軌跡所需的轉軸角度。

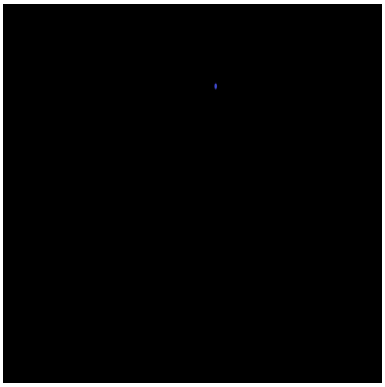


圖 1: 足端軌跡 (直線)

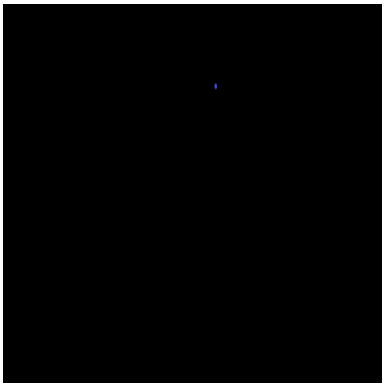


圖 2: 足端軌跡 (橢圓)

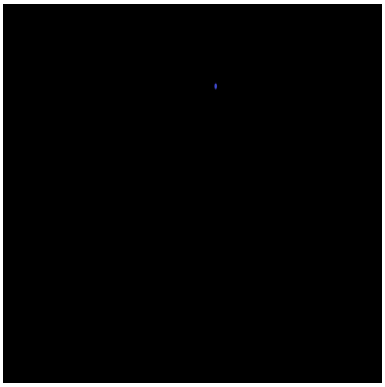


圖 3: 足端軌跡 (圓形)

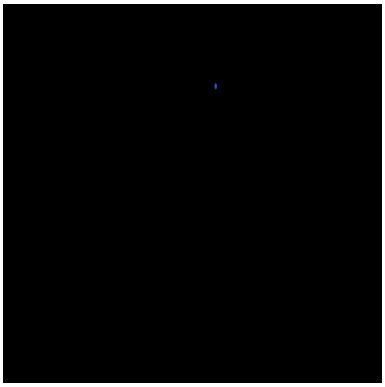


圖 4: 足端軌跡 (弧線)

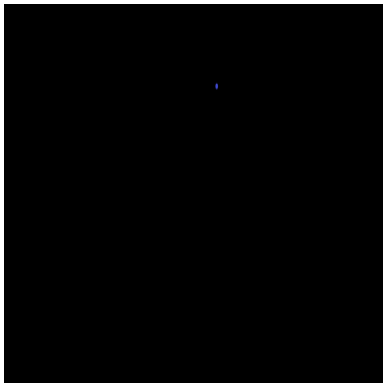
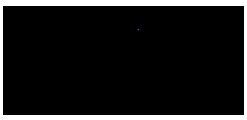
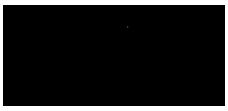


圖 5: 足端軌跡 (不規則)

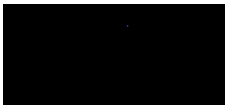
作者簡介



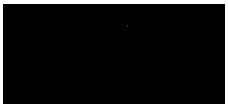
姓名　：陳岳樫
學號　：41023218
就讀學校：國立虎尾科技大學 機械設計工程系
經歷　：台中市致用高級中學
機械科



姓名　：蔡弦霖
學號　：41023248
就讀學校：國立虎尾科技大學 機械設計工程系
經歷　：國立台南高級工業職業學校
機械科



姓名　：鄭立揚
學號　：41023251
就讀學校：國立虎尾科技大學 機械設計工程系
經歷　：台南市黎明高級中學



姓名　：謝鴻元
學號　：41023254
就讀學校：國立虎尾科技大學 機械設計工程系
經歷　：國立台北科技大學附屬桃園農工高級中等學校
動力科

【05】

分類編號：112-4-CAE-3006、3004-1 有限元素法在四足機器人上的應用　一百一十三級