

Fifty Shades of Hues
Fully Automatic Color Mixer

作品說明書

何文璣 吳亞倫 鄭勝宇

指導教授： 陳政維 王新博

July 2, 2025

Abstract

本研究提出一套全自動調色系統，採用 Janua 廣告顏料為原料，旨在協助使用者在缺乏專業色彩基礎的情況下，一鍵獲得目標色彩。系統以 Raspberry Pi 3B 為控制核心，結合蠕動泵、步進馬達與 TCS34725 色彩感測器，並透過直觀的 Web 介面提供「數位選色」與「實體掃描」兩種模式：使用者只需輸入目標 RGB 比例或掃描實體目標顏色，系統即時計算並執行調配。軟體端運用 Mixbox 模型與 Kubelka–Munk 理論，透過非負最小二乘法求解顏料配比，並結合閉迴路回饋機制對混色結果進行動態校正。實驗結果顯示，本系統能在短時間內自動完成調色，最終顏色與目標色之可見色差位於可接受範圍，有效降低材料浪費並提升使用者體驗；且其模組化架構具備優良擴充性，可延伸應用於其他繪畫媒材及工業色彩配色場域。

Contents

1	Introduction	3
1.1	研究動機	3
1.2	功能概述：雙模式色彩擷取	3
2	Literature Review	3
3	Methodology	5
3.1	System Overview	5
3.2	Hardware Design	6
3.3	Color Calibration	7
3.4	Software Design	8
3.4.1	使用者介面層（Web UI）	8
3.4.2	核心邏輯層（Core API）	9
3.4.3	硬體控制代理層（Hardware Agent）	9
3.5	Mixing Algorothm	10
4	Results	12
4.1	機械結構	12
4.2	調色結果	12
5	Conclusion	13
	Acknowledgements	14

1 Introduction

1.1 研究動機

在目前國高中美術教學現場中，水彩顏料的調配往往仰賴學生的經驗與直覺。初學者因缺乏充分的色彩理論與實務技巧，在調色過程中容易出現顏料用量掌握不佳的情況，不僅導致材料的浪費與成本增加，更可能降低學生的學習動機及作品品質。儘管市場上已有部分數位輔助工具，但多數方案仍侷限於理論層面的色彩配比建議，缺乏能即時回饋並精準落實於實際操作的系統。本研究旨在透過精準、直觀的自動調色機制，協助學生無需扎實的色彩學基礎，也能迅速達成目標色彩的精準調配，以有效降低材料耗損並提升學習成就感與創作動機。

1.2 功能概述：雙模式色彩擷取

本系統提供兩種互補且高效的色彩擷取模式：

1. **數位介面選色**：使用者可透過網頁介面選取所需之任意 RGB 數值，系統即時推算並提供對應之顏料配比。
2. **實體色彩掃描**：系統整合色彩感測器以擷取實際物體之顏色，透過感測器量測所得的數位 RGB 數值，計算出相應的顏料混合比例。

此外，系統於顏料調配過程中將持續透過色彩感測器進行即時監測，動態更新混色進度並自動執行色彩校正，以確保最終色彩精確符合目標設定。

2 Literature Review

Sochorová and Jamriška (2021) 提出的 Mixbox 模型提供了一種透過電腦精確預測顏色混合結果的方法。該研究指出，傳統直接線性混合 RGB 色彩的方法無法符合現實色彩混合的直覺經驗（例如黃色與藍色混合應為綠色而非紫色），故提出了一個更符合實務的色彩混合方法，並利用數學模型與最佳化技巧解決此問題。

Mixbox 模型的數學架構概述如下：

1. **給定基色比例求混和出的 RGB 值 $RGB = \text{mix}(P, c)$ ：**

根據 Kubelka–Munk 模型（以下簡稱 KM），每種顏料皆有其吸收光譜 $K(\lambda)$ 與散射光譜 $S(\lambda)$ 。給定基色顏料集合

$$P = \{(K_i(\lambda), S_i(\lambda))\}_{i=1}^N \quad (\text{在論文中為 RGBW}), \quad c = [c_1, c_2, \dots, c_N],$$

則混合後之吸收與散射光譜分別為：

$$K_{\text{mix}}(c, \lambda) = \sum_{i=1}^N c_i K_i(\lambda), \quad (1)$$

$$S_{\text{mix}}(c, \lambda) = \sum_{i=1}^N c_i S_i(\lambda). \quad (2)$$

KM 模型可推得混合後反射光譜：

$$R_{\text{mix}}(c, \lambda) = 1 + \frac{K_{\text{mix}}(c, \lambda)}{S_{\text{mix}}(c, \lambda)} + \sqrt{\left(\frac{K_{\text{mix}}(c, \lambda)}{S_{\text{mix}}(c, \lambda)}\right)^2 + 2\frac{K_{\text{mix}}(c, \lambda)}{S_{\text{mix}}(c, \lambda)}}.$$

加入反射修正項 (k_1 與 k_2 為量測之反射常數) 後：

$$R'_{\text{mix}}(c, \lambda) = \frac{(1 - k_1)(1 - k_2) R_{\text{mix}}(c, \lambda)}{1 - k_2 R_{\text{mix}}(c, \lambda)}.$$

在一般日光 D_{65} 下，令 $\lambda = [380, 750]$ ，則可求得 XYZ 空間座標：

$$X(c) = \int_{\lambda} \bar{x}(\lambda) D_{65}(\lambda) R'_{\text{mix}}(c, \lambda) d\lambda, \quad (3)$$

$$Y(c) = \int_{\lambda} \bar{y}(\lambda) D_{65}(\lambda) R'_{\text{mix}}(c, \lambda) d\lambda, \quad (4)$$

$$Z(c) = \int_{\lambda} \bar{z}(\lambda) D_{65}(\lambda) R'_{\text{mix}}(c, \lambda) d\lambda. \quad (5)$$

最後再映射回 RGB 空間：

$$\text{mix}(P, c) = \begin{bmatrix} R(c) \\ G(c) \\ B(c) \end{bmatrix} = \frac{1}{Y_{D_{65}}} \begin{bmatrix} 3.2406 & -1.5372 & -0.4986 \\ -0.9689 & 1.8758 & 0.0415 \\ 0.0557 & -0.2040 & 1.0570 \end{bmatrix} \begin{bmatrix} X(c) \\ Y(c) \\ Z(c) \end{bmatrix}, \quad (6)$$

$$Y_{D_{65}} = \int_{\lambda} \bar{y}(\lambda) D_{65}(\lambda) d\lambda. \quad (7)$$

2. 由目標 RGB 色求基色比例 $c = \text{unmix}(P, RGB)$ ：

可表示為一最佳化問題：

$$\text{unmix}(P, RGB) = \arg \min_c \| \text{mix}(P, c) - RGB \|^2.$$

3. 留數修正：

某些 RGB 值無法純粹透過顏料混合獲得（僅可透過光混色），因此定義留數 r ：

$$r = RGB - \text{mix}(P, c) = \text{mix}(P, \text{unmix}(P, RGB)).$$

由於光量可線性疊加，因此可將基色比例與留數合併為向量 z (latent space)：

$$z = \text{enc}(RGB) = \begin{bmatrix} c \\ r \end{bmatrix} = \begin{bmatrix} \text{unmix}(P, RGB) \\ RGB - \text{mix}(P, \text{unmix}(P, RGB)) \end{bmatrix}, \quad (8)$$

$$RGB = \text{dec}(z) = \text{mix}(c) + r. \quad (9)$$

4. 基色替代：

若混色結果超出 RGB 空間範圍，會被投影至最近的 RGB 值，導致不可逆。為避免此現象，文中提出尋找替代基色集合 Q ：

$$Q = \arg \min_Q [E_{\text{push}}(Q) + \alpha E_{\text{pull}}(P, Q)], \quad (10)$$

其中 $E_{\text{push}}(Q)$ 限制混色結果落於 RGB 空間內， $E_{\text{pull}}(P, Q)$ 控制與原有基色差異， α 為權重參數（論文取 10^5 ）。

由以上運算與想法，可對整個模型給出簡短數學形式定義：

1. 一個顏料 $p = (K_i(\lambda), S_i(\lambda))$ 可由其吸收光譜與散射光譜表示，蒐集所有顏料 p 構成顏料空間 \mathbb{P} 。
2. RGB 值是 \mathbb{R}^3 中的一個元素。
3. 假設調色所使用的基色顏料有 N 個，基色顏料集合為 $P = \{p_i \mid p_i \in \mathbb{P}, 1 \leq i \leq N\}$ ，所有此類 P 構成集合 \mathbb{P}_N 。
4. 向量空間 $V_N(\mathbb{R}^N)$ 中的 c 表示基色顏料混和比例。
5. $\text{mix}_V(P, c): \mathbb{P}_N \times V_N \rightarrow \mathbb{R}^3$ ：給定基色顏料與比例，得到混合後 RGB。
6. $\text{unmix}_V(P, RGB): \mathbb{P}_N \times \mathbb{R}^3 \rightarrow V_N$ ：給定 RGB 與基色顏料，求混和比例。
7. 潛在空間 $L_N(\mathbb{R}^{N+3})$ 中的 z 同時包含基色比例與紅黃藍光修正量。
8. $\text{mix}_L(P, z): \mathbb{P}_N \times L_N \rightarrow \mathbb{R}^3$ ：透過 $\text{mix}_V(P, c)$ 與光修正量給出 RGB。
9. $\text{unmix}_L(P, RGB): \mathbb{P}_N \times \mathbb{R}^3 \rightarrow L_N$ ：結合 mix_V 與 unmix_V ，求得基色比例與光修正量。

3 Methodology

3.1 System Overview

為實現本研究所提出之自動調色系統，我們設計了一套完整且整合度高的軟硬體架構。系統核心為單板電腦 (Raspberry Pi 3B)，搭配色彩感測器 (TCS34725)、蠕動

泵、步進馬達與 LED 照明系統等元件，並透過直觀的 Web 介面提供使用者操作。

系統運作流程為：使用者透過 Web 介面選擇目標顏色或掃描實體物體取得顏色資料，系統隨即自動計算所需顏料比例，透過蠕動泵與馬達進行精準混色。在混色過程中，系統將持續透過感測器即時回饋當前顏色資訊，並動態調整以確保最終調色結果與目標顏色一致。

實務上，系統分為三個主要部分：硬體設計負責設備組裝與功能實現；色彩校正處理感測器數值精確性問題；軟體設計則負責整合前端介面、核心運算與硬體控制。本章將逐步說明上述三個部分的設計理念與實作細節。

3.2 Hardware Design

本研究所設計之調色機使用的主要硬體設備包括：

1. **蠕動泵**：提供高精度、定量的液體傳輸，透過管道擠壓推動液體，確保液體與機械完全隔離（如圖 1 所示）。

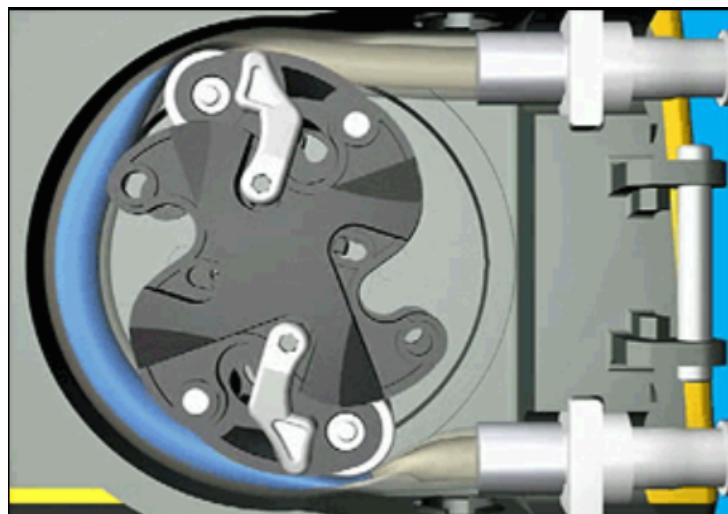


Figure 1: 蠕動泵運作示意¹

2. **TCS34725 RGB 色彩感測器**：精確偵測 RGB 光譜數值與總光通量（Clear 通道）。
3. **12V 電源供應器**：提供系統各硬體元件穩定的電力輸入。
4. **Raspberry Pi 3B**：作為系統核心控制器，負責軟硬體整合與運算控制。
5. **光耦合繼電器**：防止蠕動泵的啟動電流損害 Raspberry Pi。

¹來源：Njmcca (2013)，*Peristaltic pump in motion*，取自 https://commons.wikimedia.org/wiki/File:Peristaltic_pump.gif。授權：CC BY-SA 3.0。

6. 降壓模組：提供穩定的電壓輸入至各元件。
7. 步進馬達：用於混合顏料之機械動作。
8. LED 燈：位於容器頂部，提供穩定且均勻的照明環境以提高感測準確度。

以上元件按照圖 2 電路圖所示進行系統整合。

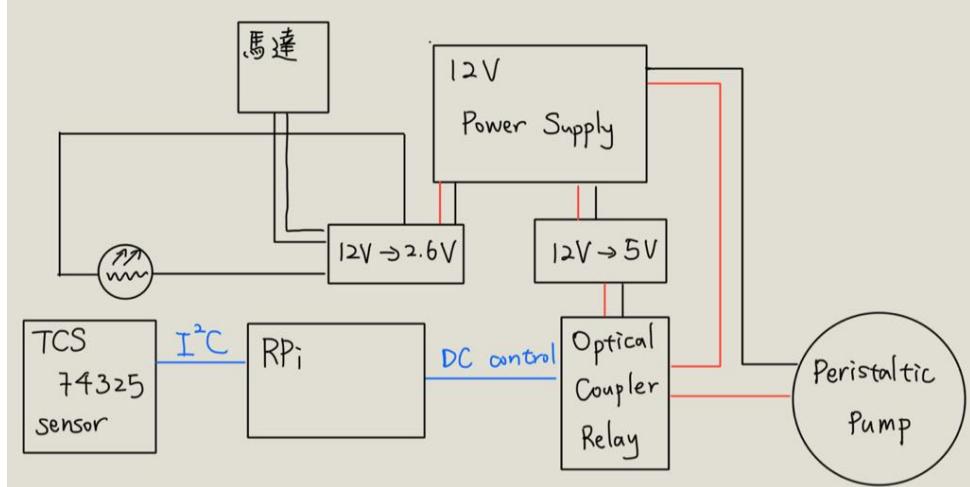


Figure 2: 電路圖

3.3 Color Calibration

TCS34725 顏色感測器會輸出四個通道：R、G、B 與 C，每個通道的數值範圍是 0 到 65535。其中 C (Clear) 通道代表整體光強度。要把資料輸入演算法之前，我們必須先把這些 RGBC 讀值轉換成標準 RGB 值，並且讓 RGB 值可以被 mixbox 讀進去。為了確保後續的演算法能以正確的色彩空間重建顏料之顏色，本專題對 TCS34725 四通道到 RGBC 感測器進行校正流程。整體流程如下：

1. **前置設定**：固定色彩感測器的整合時間 (integration time) 與光源
2. **白紙校正**：以高反射率的印刷白紙為基準，當成各個顏色通道最大進光亮的數值
在此步驟中，會得到色彩感測器中四個通道 RGBC 的光通量，並且會得到：

$$R_{wc} = \frac{R_w}{C_w}, \quad G_{wc} = \frac{G_w}{C_w}, \quad B_{wc} = \frac{B_w}{C_w}$$

並且我們將上述的 R_{wc}, G_{wc}, B_{wc} 當成是標準的 255 之值。

3. **將 TCS34725 所測到的顏料 RGBC 轉為 RGB**：

$$R_c = \frac{R}{C}, \quad G_c = \frac{G}{C}, \quad B_c = \frac{B}{C}$$

綜合白紙校正，我們可以得到色彩感測器所感知到的 RGB 值：

$$R_{norm} = \frac{R_c}{R_{wc}}, \quad G_{norm} = \frac{G_c}{G_{wc}}, \quad B_{norm} = \frac{B_c}{B_{wc}}$$

4. 色彩校正矩陣 (Color Correction Matrix, CCM) :

$$VM = R \Rightarrow M = V^+R$$

其中 V 為我們以上述方式測量色彩感測器所感知到三個標準顏料的 RGB 值， R 上網尋找這三種顏料標準的 RGB 值， M 即為 CCM。以建立出 CCM 的方式，我們可以將感測器感知到的色彩空間轉為實際上的標準色彩空間。

5. 在 UI 介面上的 Gamma correction：依照 IEC 61966-2-1 規範，假設 c 螢幕真正要發出的亮度比例， c' 是壓縮成符合人眼的非線性數字：

$$c' = \begin{cases} 12.92c, & c \leq 0.0031308, \\ (1 + 0.055)c^{\frac{1}{2.4}} - 0.055, & c > 0.0031308 \end{cases}$$

3.4 Software Design

本研究的軟體系統採用三層式微服務架構（圖 3），各層之間透過明確的 API 介面進行通訊，以確保模組化的高擴展性、易維護性與獨立性。

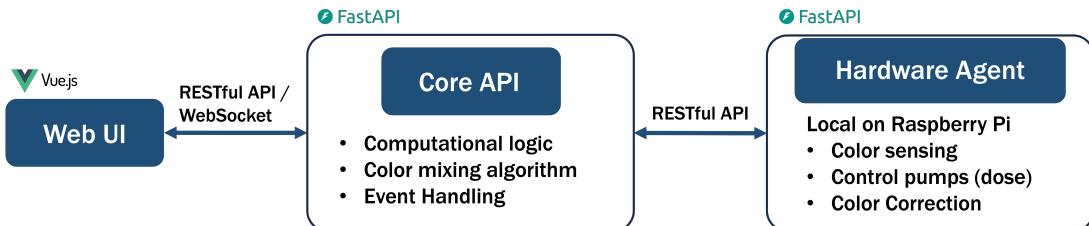


Figure 3: 軟體架構圖

3.4.1 使用者介面層 (Web UI)

使用者介面採用 Vue.js 前端框架設計，考量其對即時更新與高互動性的強大支援。介面包含顏色選擇器與即時混色狀態顯示，用戶可直覺選擇目標顏色或查看當前顏色感測器的即時回饋，並透過 RESTful API 將混色請求即時傳送至後端 Core API。此外，系統亦透過 WebSocket 實現資料即時推送，確保介面能即時反映後端的最新狀態，提高使用者的操作體驗。

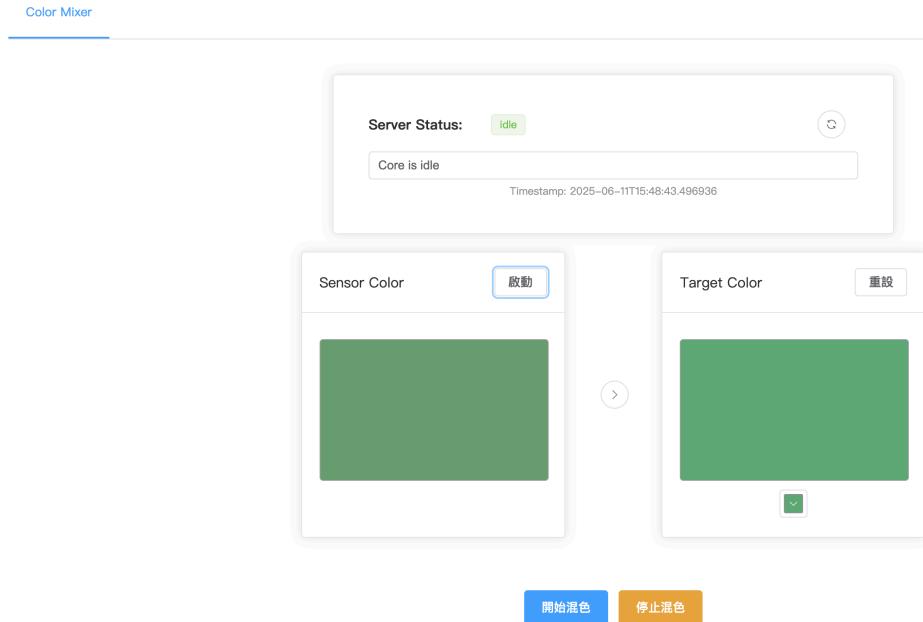


Figure 4: 前端 Web UI 畫面截圖

3.4.2 核心邏輯層 (Core API)

Core API 使用 Python 語言及 FastAPI 框架建構，專責核心的混色演算法計算與流程控制。當接收到前端請求後，系統透過 Mixbox 模型實作 Kubelka–Munk 色彩理論，並運用 numpy 進行高效的矩陣運算，計算精確的顏料配比。此外，Core API 在系統內部使用 WebSocket 以非同步方式，將顏色數據與狀態更新及時回傳給前端，讓使用者得以即時掌握混色進度。

為提高系統穩定性及容錯能力，本系統以 Python 內建的 `asyncio` 進行非同步任務管理，透過全域狀態（Global state）搭配 `asyncio.Lock()` 確保併發處理時不會產生資源競爭問題。每個混色任務以 `asyncio.create_task()` 獨立執行，允許有效地處理任務取消、異常狀態與錯誤回報，進而達成高穩定性及高效能的系統運作。

3.4.3 硬體控制代理層 (Hardware Agent)

Hardware Agent 為部署於 Raspberry Pi 上的硬體控制後端，同樣採用 Python 搭配 FastAPI 框架設計，負責與色彩感測器、蠕動泵及步進馬達等硬體設備進行直接溝通。所有硬體控制指令均封裝成 RESTful API，以清楚定義的端點 (endpoints) 供 Core API 呼叫，確保軟硬體之間通訊的標準化。

此外，系統採用 driver 層進行硬體抽象化，讓不同硬體設備的驅動程式能夠在不影響核心邏輯下輕鬆地被替換或升級。為應對硬體層面常見的異步需求（如啟停控制），系統使用非同步函式搭配 `asyncio` 提高硬體指令的即時反應能力，避免異常狀況下的設備誤動作與材料浪費。

透過以上的分層架構與非同步設計，本研究所建立的軟體系統具備良好的可擴展性、可維護性及使用者體驗，充分展現了微服務架構在自動調色系統中的優勢與實踐價值。

3.5 Mixing Algorithm

我們所設計的混色演算法負責將使用者指定的目標顏色自動調配出實體顏料混合。整體流程結合了顏色空間轉換、數值比例計算與控制迴路，並透過 API 串聯軟硬體，以完成從輸入到出料的所有步驟。以下說明混色演算法的核心運作流程：

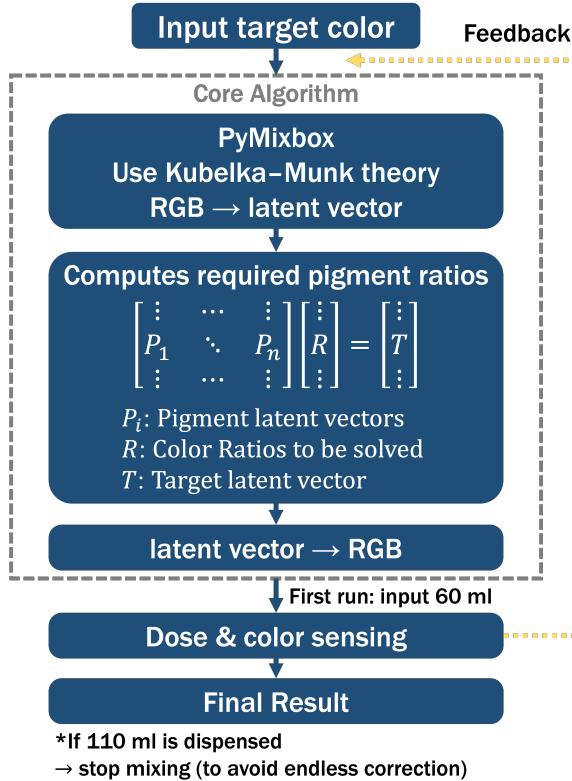


Figure 5: 演算法流程圖

1. **預處理顏料數值 (palette)**：在系統啟動時，會從 `palette.json` 檔案中讀取所有基底顏料集 P 中每一個基底顏料 p_i 的潛在空間向量 z_i ，並構造出基底顏料矩陣 $M = [z_1, z_2, \dots, z_N]$ 。
2. **目標色輸入與轉換**：系統接收使用者輸入的目標顏色的線性 RGB 值 $(R, G, B) \in \mathbb{R}^3$ ，並經由 PyMixbox 實作的 $unmix(P, RGB)$ 將目標 RGB 值打到對應的潛在空間目標向量 T 。
3. **計算顏料比例**：考慮比例方程式 $MR = T$ ， R 是個基底顏料的添加比例，透過非負最小二乘法 (Non-negative Least Squares, NNLS) 求解此方程組，可以得到各

基礎顏料所需的添加比例 R (所有比例皆為非負值)。這些比例代表了各色顏料在混合中所需的相對用量。

4. **加料與初次混合**：根據上述計算出的比例，系統透過 FastAPI 控制下的硬體裝置將各顏料定量加入混合容器中，進行第一輪混合。初次混合時預計投放總量約 60 單位的顏料 (各色依計算出的 R 的比例折算)，此舉可讓系統保留後續校正的空間。
5. **色彩感測與回饋**：第一輪混合完成後，系統使用色彩感測器讀取目前混合液的顏色 T' ，並將其與目標色比較，令 $\Delta E = \|T - T'\|$ 。如果感測結果已在容許的色差範圍內，即 $\Delta E < 0.02$ ，則表示混色成功；若存在明顯偏差，演算法會根據色差進行回饋調整。系統將當前混合所得的顏色視為新的基準色，計算其與目標色之間的差異 $T - T'$ ，再放回比例方程式使用 NNLS 解算出需要額外添加的各色顏料比例，並按此比例加入相應顏料以進行顏色校正。
6. **迴圈控制與終止條件**：上述加料、感測與回饋的過程在 FastAPI 控制流程下循環執行，不斷調整顏料配比並逐步逼近目標色。當混合總添加量達到上限 (例如 110 單位) 或感測顏色已達到目標要求 (色差在容許範圍內) 時，迴圈終止。若已投入上限容量仍未匹配目標，則停止混色以避免無休止的修正，並以目前混色結果作為最終產物。透過上述自動循環機制，系統最終能夠獲得與目標色相符 (或在容許誤差內) 的混合顏色。

4 Results

4.1 機械結構

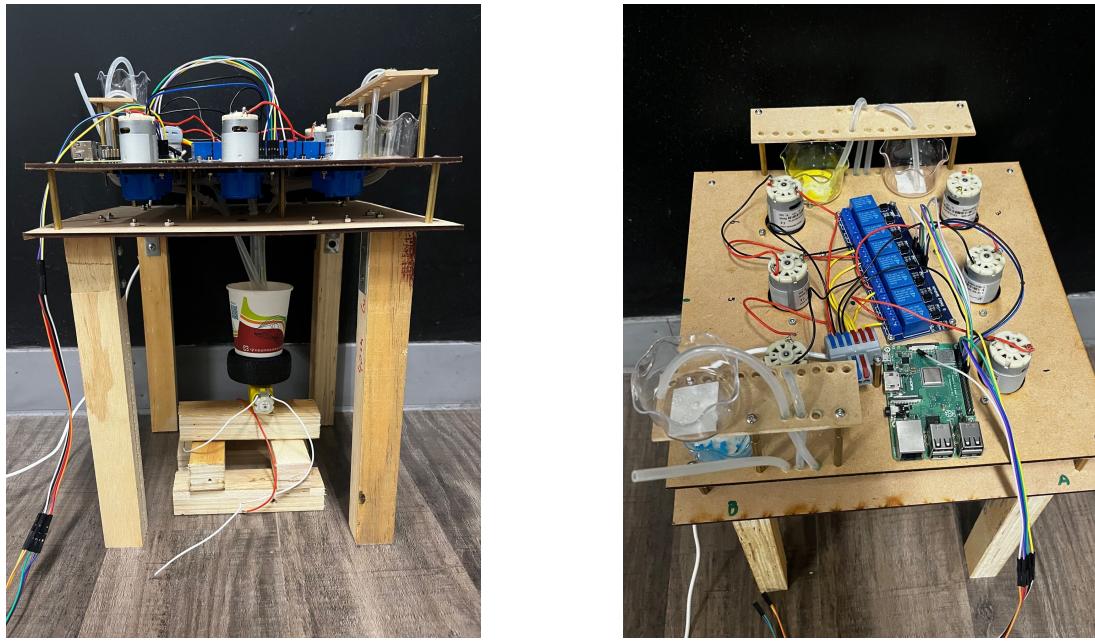


Figure 6: 最終機構之側視與上視外觀

4.2 調色結果

1. Web UI 點選顏色並自動調配

使用者於 Web 介面選取目標色後，系統立即計算顏料比例並完成調色，結果如圖 7 所示。

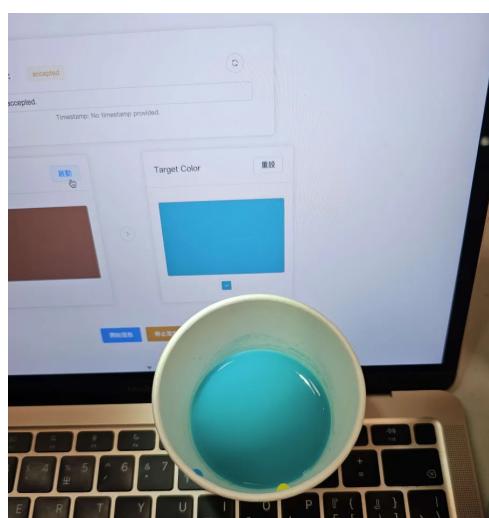


Figure 7: Web UI 選色與調配結果

2. 掃描既有顏色並重現

以感測器掃描一份已調配好的顏料作為目標色，系統據此計算配方並重新調色。圖下為目標顏料，圖上為本機器調配之顏料（圖 8）。混色過程影片請見：
<https://youtu.be/kJjAyAqb1aQ>



Figure 8: 感測掃描後的調色重現結果

5 Conclusion

本專題成功做出一套自動調色的系統，從理論建構、硬體整合到軟體介面皆已驗證可行。透過 Kubelka-Munk 混合顏料的光譜模型、mixbox 開源演算法，以及以矩陣求出所需的顏料比例，實測誤差能夠讓視覺認為兩者相似。搭配 Web UI 與色彩感測器的雙模式取色，使用者可以在短時間內就獲得高精度的配方。

本系統架構可模組化並且成本可控，可擴充至壓克力顏料、數位印刷等不同媒材，具備商品化與跨領域應用潛力。此系統也可以用來調配天然染料或食用色素等等，只要顏色感測器能達到偵測透明顏料的效果。

Acknowledgements

在本專題之中，承蒙許多師長、同學與單位的協助，方能順利完成。謹向以下致上誠摯的感謝：

首先感謝指導教授陳政維教授與王新博教授，以及電資工程入門設計與實作的助教們，不吝分享他們對於色彩學的知識，並適時的給予提醒、建議與鼓勵；也謝謝學校的經費單位，提供我們購買所需材料的所有資金。同時感謝 PyMixbox、FastAPI、NumPy 與 Vue.js 等開源社群的貢獻，無償提供程式庫，使我們能快速搭建演算法與前端介面。最後，感謝我們這組三位認真的組員，在硬體與軟體實作上都實作的很完善，尤其在最後一起做出成品時的相互鼓勵與陪伴，讓系統運作更為流暢，也讓這次的專題更有意義！

References

- scrtwpns (2021). *mixbox: Pigment-Based Color Mixing*. <https://github.com/scrtwpns/mixbox>. Initial commit: 03 Dec 2021; Accessed: 13 Jun 2025.
- Sochorová, Šárka and Ondřej Jamriška (Dec. 2021). “Practical pigment mixing for digital painting”. In: *ACM Trans. Graph.* 40.6. ISSN: 0730-0301. DOI: [10.1145/3478513.3480549](https://doi.org/10.1145/3478513.3480549). URL: <https://doi.org/10.1145/3478513.3480549>.