

Color Deviation Terminator

組員：黃書霈、陳昱亨、古佳儂、鄭博升

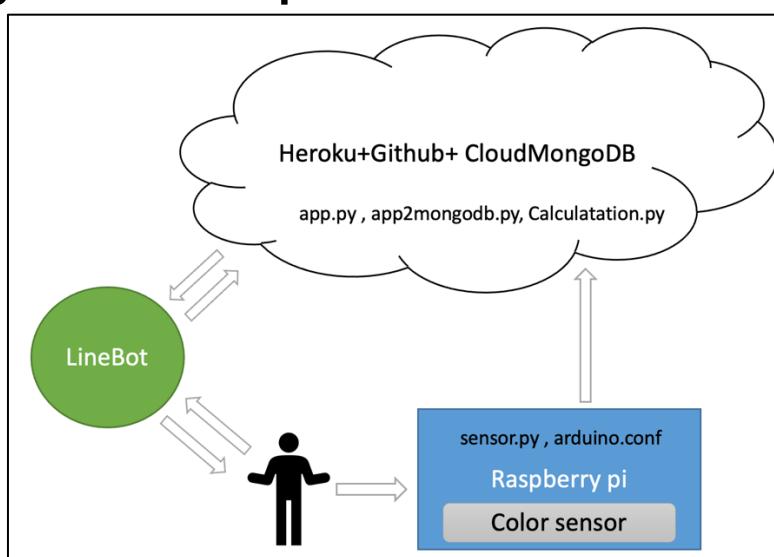
Abstract

大部分商標在設計後往往都有記錄下其特定的色碼來顯示圖案的顏色，但經過不同的媒介輸出產品後，可能會得到不同的色碼值且圖案顏色有明顯的肉眼差異。我們透過 Linebot 來接收以及回傳，透過將 TCS3200 色彩傳感器把測得的實際產品顏色色碼值，傳到 Sever 端所連接的 Cloud MongoDB 資料庫和透過計算程式來和理想色碼值進行色差運算後，再回傳到 Linebot，告知使用者實際產品顏色和理想顏色之間的色差值情況以及建議採用的色碼值。

I. Introduction

在現今廣告宣傳頻繁的時代，每個商標的圖案都有特定的色碼，這些顏色所代表的意義都不同，如果在製作商標時，因為色差導致顏色與原本想表達的色彩不同，對商家來說反而會造成非常大的損失。藉由 Color Deviation Terminator，在印製前，進行色差的校正，將色差縮小，並且結合 linebot，讓我們能輕鬆便利的接收測量的結果，即時做出修正，印出接近我們理想的色碼圖。

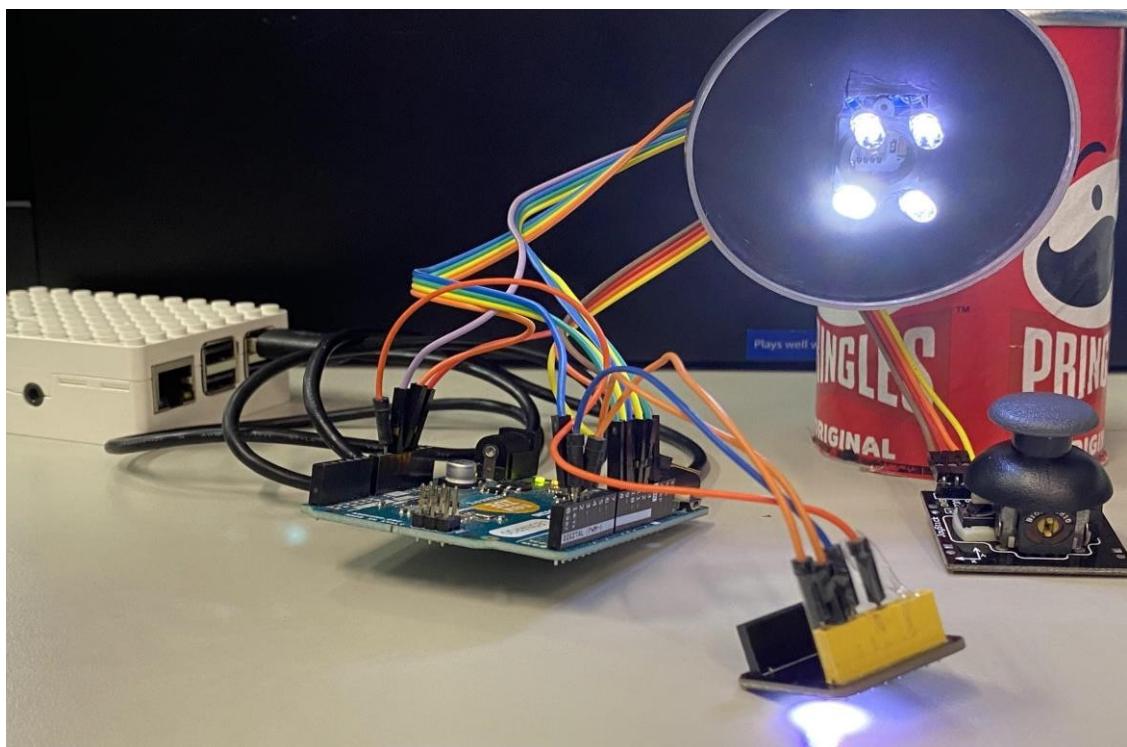
II. Project Concept



主要分成 user 端 與 server 端 兩部分，server 端負責儲存感測器測得到的色碼值以及使用者預期的色碼值，user 端則透過 Linebot 與 server 端溝通。使用者輸入理想色碼值(R,G,B)傳給 Linebot 後，操作感測器來取得實際色碼值(R,G,B)，傳到 Cloud Sever 後進行色差運算，再將誤差結果以及調色建議藉由 Linebot 告知使用者。

III. Hardware design

完整架構圖：



A. Raspberry Pi 3 Model B+

使用作為感測器傳送數據的平台，利用 mqtt supervisor 負責將測得的數據傳回 Cloud mongoDB。



B. Color Sensor

使用 TCS3200 色彩傳感器，包括一個 TAOS、TCS3200、RGB 傳感器芯片和 4 個白色 LED，在感測一個物體的顏色時，需輪流讓三個顏色通過濾波器檢測，並分析感測器的回傳的脈衝，經過換算後得到顏色的 RGB 值，以此方式感測顏色。



C. PS3 Joystick as Bottom

利用 PS3 搖桿按下的功能代替按鈕零件，主要功能是判斷目前工作目的是校準還是發送，按下後必須先使用白色和黑色對我們的 color sensor 進行初始化校準，並在五秒後傳送當下測量的物體顏色的(RGB)數值傳送到 Cloud mongoDB 資料庫。



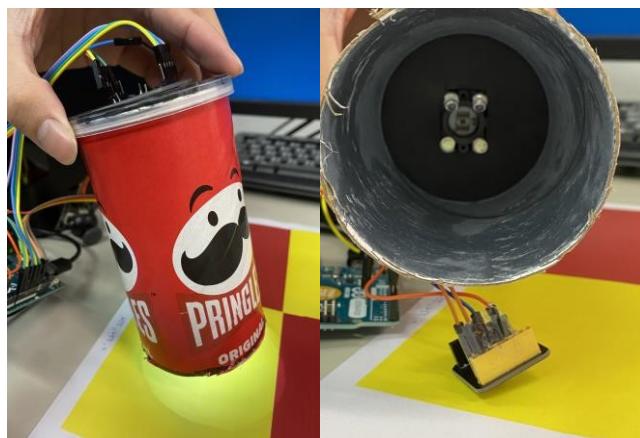
D. RGB LED as Guide Light

按下按鈕後，以指示燈閃爍藍燈的方式來提示進行校準動作，但在未按下按鈕時，則能顯示當下測得的大概顏色。



E. fixed focus

利用品客的盒子來做固定焦距，焦距的長度經過實驗後採 11cm。



IV. Software design

Sensor 端與 User 端之間的訊息傳輸主要透過 Linebot 和 Heroku 來達成。

Sever 端和 User 端之間的訊息傳輸主要透過 git 來達成

A. Linebot

使用者和 Heroku app 的互動介面，使用者輸入理想色碼透過 Heroku app 來取得色差數據以及建議調整數值比例。

Linebot 操作介紹

Step1 : 輸入 hi 紹予操作提示。

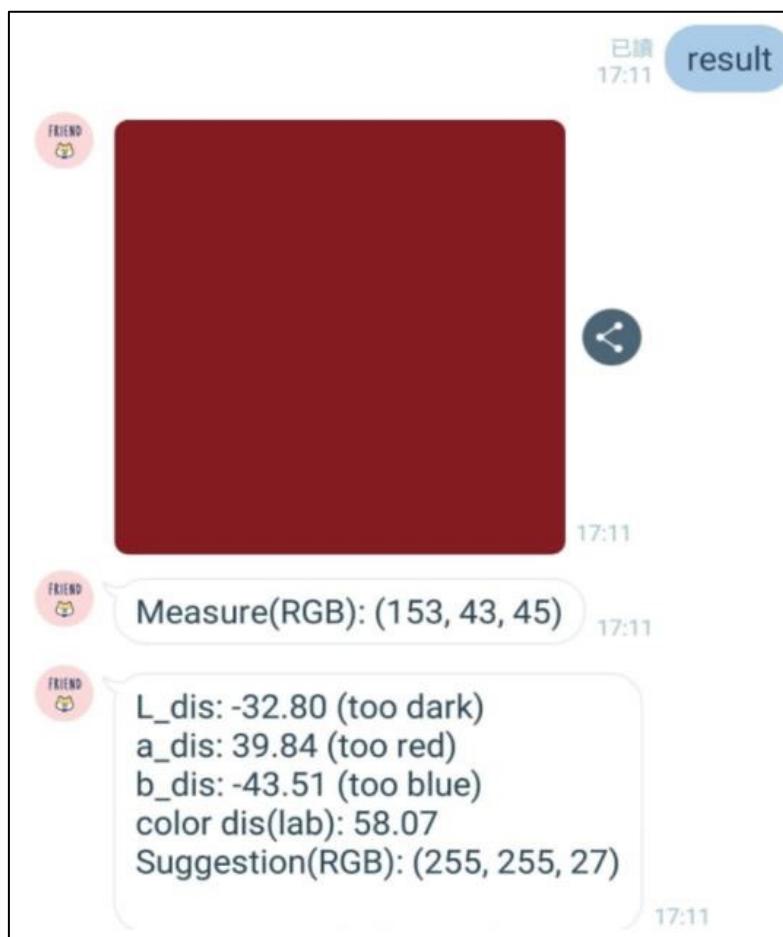


Step2 : 輸入色碼(RGB)，獲得該色碼值的顏色圖，並將理想色碼傳入 Heroku app。



Step3 : 使用樹莓派連接感測器按下按鈕進行校準後，實際測試色卡的(RGB)數值，並傳回 Cloud mongoDB。

Step4 : 輸入 result 後分別取得測量色彩、色差結果、建議調整顏色(RGB)。



B. Server

架設的伺服器：

1. MongoDB(架設於 Heroku 連接 Cloud MongoDB) :

使用 Heroku app 上串接 Cloud MongoDB，用於儲存量測所獲得的色彩值和時間戳記。DB 裡面共有兩張資料表(line, sensor)，分別存傳過來的色彩值，其中 line 表上則有多儲存了 UserId 用來指定用戶。

I. LineBot Input:

在 Heroku app 分別會執行 app.py 和 app2mongodb.py，當 LineBot Server 接收到要存入資料庫的資料時會直接透過 Heroku 傳送到 Cloud MongoDB，並且採用使用 Subprocess 的方式輸入給 MongoDB。

II. Sensor Input:

利用 supervisor 設計執行腳本 arduino.conf 將所有伺服器架設在 raspberry pi 背景不斷監控並自動執行 command , 透過 sensor.py 使用 Subprocess 的方式來獲得 color sensor 所量測的數值並直接傳到 Cloud mongodb.

2. LineBot Server(架設於 Heroku) :

使用 python 套件 Flask , 將 python 作為一個網站 server , 可以接受 HTTP Request , 讓 LineBot 可以從網站知道如何回覆 Line 訊息。

3. LineBot 圖庫 FTP Server

Line 為了傳送顏色照片 , 需要以 https based 的檔案超連結利用 Heroku 加設後的 https 連結來取得

C. Sensor

用 Arduino 控制 TCS3200 色彩傳感器、PS3 Joystick、RGB LED , 接上電源後 , 感測器就會開始不斷測量色碼 , 按下 PS3 Joystick 按鈕後 , 即開始初始化校正 , 同時 RGB LED 燈會閃爍 , 並在校正完後 3 秒 , 取得當下色碼傳至 server 端 , RGB LED 燈也會同時顯示測得的顏色。 在色彩傳感器校正方面 , 採用黑色與白色進行校正 , 此方法可避免需要分別找到純粹的紅、綠和藍。

D. Color Difference Calculation:

將理想色碼與真實色碼分別轉成 Lab , 並將真實色碼的 Lab 值減去理想色碼的 Lab 值 , 取得 L、a、b 的差值 , 運用 L、a、b 的差值計算色差 , 並將理想色碼的 Lab 分別減去計算所獲得的 L、a、b 的差值 , 取得建議調整結果。

V. Experiment

硬體：印表機（透過圖書館 1 樓彩色印表機）

軟體：小畫家（編輯處理圖片）

測試顏色：

1. 麥當勞紅(R=218, G=41 , B=28)
2. 麥當勞黃(R=255, G=199, B=44)

濾鏡：使用彩色 L 夾作為濾鏡(藉此增加實際色差)



理想麥當勞圖案



使用理想色碼列印出麥當勞黃(左)與麥當勞紅(右)



使用計算後的建議色碼列印後的麥當勞黃(左)與麥當勞紅(右)

實驗結果:

測試物	測得色碼(RGB)	建議調整色碼	測得色碼 (使用建議色碼)
麥黃(麥當勞黃)	(208, 148, 99)	(255, 255, 2)	(247, 231, 139)
麥紅(麥當勞紅)	(198, 66, 75)	(224, 20, 4)	(197, 89, 90)
麥紅+綠 L 夾	(179, 132, 122)	(207, 0, 0)	(191, 54, 48)
麥黃+綠 L 夾	(200, 176, 138)	(255, 255, 0)	(89, 151, 58)
麥紅+藍 L 夾+紅 L 夾	(192, 155, 156)	(180, 0, 0)	(186, 138, 138)
麥黃+藍 L 夾+紅 L 夾	(210, 174, 160)	(255, 228, 0)	(212, 171, 128)
麥黃+綠 L 夾+紅 L 夾	(211, 184, 151)	(255, 216, 0)	(203, 188, 160)
麥紅+綠 L 夾+紅 L 夾	(189, 146, 162)	(183, 0, 0)	(185, 155, 155)

VI. CONCLUSION

Color Deviation Terminator 使用顏色感測器，搭配 linebot 機器人，與使用者互動，提取實際物體的色碼，並與理想色碼進行校正。在單純針對麥當勞色碼的校正上，建議色碼所印出的結果確實較原始色碼好，但在加入濾鏡改變測試顏色後，建議色碼效果較無明顯變好，推測是在 LAB 計算的過程中，產生溢位情況，目前尚未找出一個合適的解決方式，來針對溢位數值進行調整，只能單純讓溢位的值停在邊界，也就是 L 介於 0~100 之間，A 和 B 介於 -128~127 之間，另外在顏色測量方面，也盡量遮蔽其他光源並在多次嘗試後，將距離固定為 11 公分，但每次測得的數值還是會有些許的不同，在濾鏡添加部分，我們採用的是有色的 L 夾，來對印出的紙張增加更多的色差情況，不確定 L 夾是否對顏色感測器的顏色識別造成其他未知影響。

目前的 Color Deviation Terminator 可取得概略的物體色碼，並給予精確的 LAB 差值計算，但在色碼建議部分，還有很多改進空間，使用者可參考色差數值與建議，再根據個人經驗作更細微的調整。