本書並不是學習 Python 的第一本書。我不會教你基礎知識。然而，我會透過一系列不同一般的專案，向你展示如何使用 Python 解決現實世界中的各種問題。在完成這些專案的過程中，你將探索 Python 程式語言的細微之處，並學習如何使用一些流行的 Python 程式庫。但也許更重要的是，你會學到如何將問題分解成各個部分、發展出解決問題的演算法，然後使用 Python 從頭開始實作出解決方案。

解決現實世界中的問題可能很困難，因為這些問題往往是開放式的，需要不同領域的專業知識。但 Python 提供了有助於解決問題的工具。克服困難並找到解決實際問題的方法，是你成為專業程式設計師道路上最重要的部分。

本書涵蓋些什麼？

讓我們快速瀏覽一下本書的各個章節。第一部介紹幾個專案，幫助你輕鬆入門。

第 1 章：Koch 雪花（Snowflake） 使用遞迴函式和 `turtle` 圖形繪製有趣的碎形

第 2 章 萬花尺（Spirographs） 利用參數方程式和 `turtle` 圖形繪製曲線，就像用萬花尺玩具所畫出的曲線一樣

第二部介紹利用數學模型對現實世界的現象進行電腦模擬的專案。

第 3 章：Conway 的生命遊戲 使用 `numpy` 和 `matploblib` 實作著名的細胞自動機，根據幾條簡單的規則生成不斷演化、栩栩如生的系統

第 4 章：使用 Karplus-Strong 演算法的泛音 逼真地模擬彈撥樂器的聲音，並用 `pyaudio` 播放音效

第 5 章：成群的仿鳥（Boids） 使用 `numpy` 和 `matplotlib` 實作 Boids 演算法，並模擬鳥類成群飛舞的行為

第三部中的專案將向你介紹如何使用 Python 讀取和處理 2D 影像。

第 6 章：ASCII 藝術 透過將影像變換為基於文字的 ASCII 藝術，介紹 Python Imaging Library（PIL）的分支 `Pillow`

第 7 章：相片拼貼（Photomosaics） 將網格狀的小影像拼接成可識別的大影像

第 8 章：立體圖（Autostereograms） 利用深度圖和像素操作，在 2D 影像中嵌入 3D 影像的錯覺

在第四部中，你將學習使用著色器（shaders）和 OpenGL 程式庫，透過直接使用圖形處理器（GPU）快速且有效率地描繪 3D 圖形。

第 9 章：了解 OpenGL　介紹使用 OpenGL 建立簡單 3D 圖形的基礎知識

第 10 章：環面（Torus）上的 Conway 生命遊戲　重溫第 3 章中的模擬，在 3D 環面上將其變為現實

第 11 章：容積描繪（Volume Rendering）　實作容積光線投射（volume ray casting）演算法來描繪容積資料，這是一種常用於 MRI 和 CT 掃描等醫學成像的技術

最後，在第五部中，你將使用 Raspberry Pi 和其他電子元件在嵌入式系統上部署 Python。

第 12 章：Raspberry Pi Pico 上的 Karplus-Strong 演算法　使用 MicroPython 在小型的 Raspberry Pi Pico 微控制器上實作第 4 章中的 Karplus-Strong 演算法，製作可演奏的電子樂器

第 13 章：使用 Raspberry Pi 實現雷射音樂視覺效果　利用 Raspberry Pi 上的 Python 來控制兩面旋轉鏡和一部雷射，製作出能對聲音做出反應的雷射光影秀

第 14 章：IoT 花園　使用 Bluetooth Low Energy（藍牙低功耗）技術將 Raspberry Pi 與執行 CircuitPython 的 Adafruit 硬體連接起來，建立監控花園溫度和溼度的 IoT（物聯網）系統

第 15 章：Raspberry Pi 上的音訊機器學習　透過在 Raspberry Pi 上實作語音辨識系統，用 TensorFlow 介紹令人興奮的機器學習世界

每章結尾都有「實驗！」部分，就如何擴充專案或進一步探索當前主題提出建議。

第二版有哪些新內容？

第二版新增了五個專案，包括第 1 章中的 Koch 雪花專案和第 10 章中的 3D 版 Conway 生命遊戲。最重要的更新在硬體部分，該部分經過精簡，只關注基於 Raspberry Pi 的系統，而不是 Raspberry Pi 和 Arduino 的結合體。因此，第五部的每個專案要麼是新專案（第 12 章、第 14 章和第 15 章），要麼就是經過徹底重新設計的舊專案（第 13 章）。

僅仰賴 Raspberry Pi 可以簡化硬體專案的設定過程，並使本書的焦點保持在 Python 程式設計上。不再需要在 Python 和用 Arduino 程式語言（一種版本

的 C++）編寫的程式碼之間切換。透過更新後的第五部，你還將體驗到使用 MicroPython 和 CircuitPython 編寫程式的樂趣，這兩種 Python 經過最佳化，可在資源受限的嵌入式系統上執行。

第二版的其他重要更新包括：

- 在第 4 章中使用 `pyaudio` 而非 `pygame` 播放 WAV 檔案
- 比較線性搜尋演算法和 k-d 樹狀資料結構在為第 7 章中的照片拼貼尋找最佳匹配影像時的效能
- 在第 8 章中指引如何建立自己的深度圖（depth maps）以生成立體圖
- 附錄 A 中使用標準的 Python Anaconda 發行版簡化安裝說明

除了這些具體的更新外，整本書都經過審查、更正和澄清，並根據需要更新了程式碼，以反映自第一版釋出以來 Python 的變化。

為何選擇 Python ？

Python 是探索程式設計的理想語言。作為一種多典範語言，它為你提供很大的彈性，讓你可以靈活地架構自己的程式。你可以把 Python 當作指令稿語言（*scripting* language）單純執行程式碼，也可以將其作為程序式語言（*procedural* language）來將程式組織成相互呼叫的函式集合，還可以將其當成物件導向語言（*object- oriented* language）來使用類別、繼承和模組以建立階層架構。這種變通性讓你可以挑選最適合特定專案的程式設計風格。

使用 C 或 C++ 等傳統語言進行開發時，必須先編譯和連結程式碼才能執行。使用 Python，你可以在編輯後直接執行它（在幕後，Python 會將你的程式碼編譯成中介的 bytecode，然後由 Python 直譯器執行，但這些過程對使用者來說是透明不可見的）。實務上，反覆修改和執行程式碼的過程在使用 Python 時會簡單得多。

Python 有非常小型的一組簡單但強大的資料結構。如果你已經了解字串、串列、元組、字典、串列概括式（list comprehensions）和基本控制結構（如 `for` 和 `while` 迴圈），那麼你就有了很好的起點。Python 的語法簡潔而富有表現力，只需幾行程式碼就能輕鬆完成複雜的運算。一旦你熟悉了 Python 的內建模組和第三方模組，你就會擁有一套多樣化的工具以解決實際問題，比如本書所涉及的那些。從 Python 呼叫 C/C++ 程式碼或者反向進行呼叫都有標準的方法可循，而且因為在 Python 中可以找到幾乎任何事情都能做的程式庫，所以在大型專案中很容易將 Python 與其他語言模組結合起來。這就是為什麼 Python 被認為是一種很好的黏著劑語言（*glue* language），因為它可以

輕鬆地整合各種不同的軟體元件。第四部中的 3D 圖形專案展示了 Python 如何與類似 C 語言的 OpenGL Shading Language 並肩工作，而在第 14 章中，你甚至可以混合使用 HTML、CSS 和 JavaScript，為你的 IoT 花園監視器建立 Web 介面。真實的軟體專案通常會混合使用多種軟體技術，而 Python 非常適合這種分層架構。

Python 還以 Python 直譯器（interpreter）的形式提供便利的工具。透過它，你可以輕鬆檢查程式碼語法，進行快速計算，甚至測試發展中的程式碼。我編寫 Python 程式碼時，會同時開啟三個視窗：文字編輯器、shell 和 Python 直譯器。在編輯器中開發程式碼時，我會把函式或類別匯入直譯器，並在過程中對它們進行測試。

將新模組整合到程式碼之前，我也會使用直譯器來初步了解它們。舉例來說，在為第 14 章中的 IoT 花園專案開發程式碼時，我希望測試 `sqlite3` 資料庫模組。於是我調出 Python 直譯器並嘗試了以下運算，以確保我理解如何建立資料庫並新增條目：

```
>>> import sqlite3
>>> con = sqlite3.connect('test.db')
>>> cur = con.cursor()
>>> cur.execute("CREATE TABLE sensor_data (TS datetime, ID text, VAL numeric)")
>>> for i in range(10):
...   cur.execute("INSERT into sensor_data VALUES (datetime('now'),'ABC', ?)", (i, ))
>>> con.commit()
>>> con.close()
>>> exit()
```

然後，為了確保它能正常運作，我做了下列操作來檢索新增的一些資料：

```
>>> con = sqlite3.connect('test.db')
>>> cur = con.cursor()
>>> cur.execute("SELECT * FROM sensor_data WHERE VAL > 5")
>>> print(cur.fetchall())
[('2021-10-16 13:01:22', 'ABC', 6), ('2021-10-16 13:01:22', 'ABC', 7),
 ('2021-10-16 13:01:22', 'ABC', 8), ('2021-10-16 13:01:22', 'ABC', 9)]
```

本範例演示了 Python 直譯器作為強大開發工具的實際用途。你不需要編寫完整的程式就能進行快速實驗；只需開啟直譯器就可以開始了。這僅是我喜歡 Python 的眾多原因之一，也是我認為你同樣會喜歡上它的原因。

2

萬花尺

你可以使用萬花尺（Spirograph）玩具（如圖 2-1 所示）來繪製數學曲線（mathematical curves）。這種玩具由一大一小兩個大小不同、帶有塑膠齒輪的圓環組成。小環上有幾個小孔。將原子筆或鉛筆穿過其中小孔，然後在大齒輪（內側有齒輪齒）內轉動小齒輪，持續讓齒輪相互接觸，就可以畫出無窮無盡複雜而奇妙的對稱圖案。

在本專案中，你將使用 Python 製作類似萬花尺曲線的動畫。程式將使用參數方程式（parametric equations）來描述萬花尺圓環的動作，並繪製曲線（我稱之為 *spiros*，即「繁花曲線」）。你會把繪製完成的圖形儲存為 PNG 影像檔案。程式會隨機繪製 spiros，你也能使用命令列選項以特定參數繪製繁花曲線。

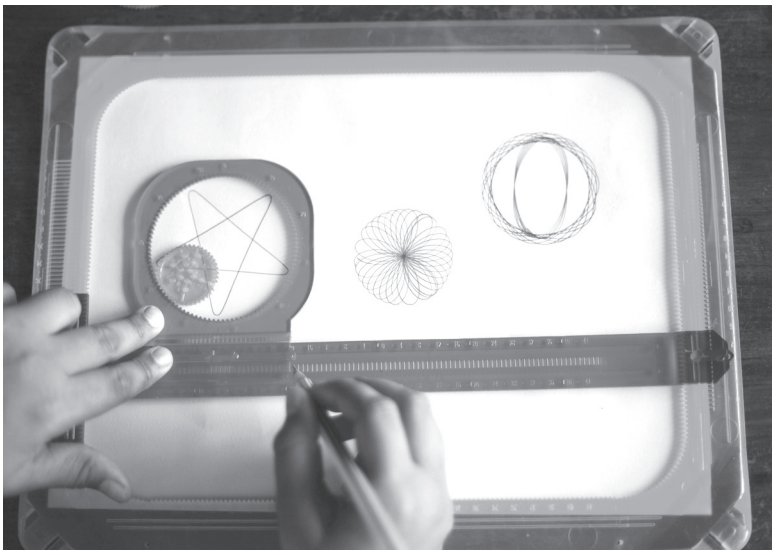


圖 2-1：萬花尺玩具

在本專案中，你將學習如何在電腦上繪製 spiros，還會學到如何進行以下操作：

- 使用參數方程式生成曲線。
- 使用 `turtle` 模組將曲線當作一系列直線繪製。
- 使用計時器（timer）製作圖形動畫。
- 將圖形儲存為影像檔案。

需要提醒的是：我選擇使用 `turtle` 模組來繪製 spiros 主要是出於教育目的，也因為它很有趣，但 `turtle` 的速度很慢，對於建立效能要求很高的圖形而言並不理想（畢竟，你能要求烏龜跑多快呢？）。如果你想快速繪製某些東西，還有更好的方法，你將在接下來的專案裡探索其中的一些選項。

運作方式

這個專案的關鍵是使用*參數方程式*（*parametric equations*），也就是將曲線上各點的座標以具有一或多個變數（variables）的函數（functions）表示，這些變數被稱為*參數*（*parameters*）。你會把那些參數的值代入方程式，計算出構成 spiro 圖案的點。然後將那些點輸入給 `turtle` 模組以繪製曲線。

了解參數方程式

要了解參數方程式的工作原理，我們先來看一個簡單的例子：圓（circle）。考慮以 2D 平面的原點為圓心、半徑（radius）為 r 的一個圓。這個圓由 x 座標和 y 座標滿足方程式 $x^2 + y^2 = r^2$ 的所有點組成。然而，這並不是參數方程式。參數方程式會根據其他變數（參數）的變化，給出 x 和 y 所有可能的值。

現在，考慮下列方程式：

$$x = r\cos(\theta)$$
$$y = r\sin(\theta)$$

這些方程式是圓的參數表示（*parametric* representation），其中的參數是 θ，即點 (x, y) 相對於正 x 軸的角度（angle）。在這些方程式中，(x, y) 的任何值都將滿足原始的 $x^2 + y^2 = r^2$ 方程式。隨著 θ 從 0 變為 2π，這些方程式產生的 x 座標和 y 座標將構成一個圓。圖 2-2 展示這種模式。

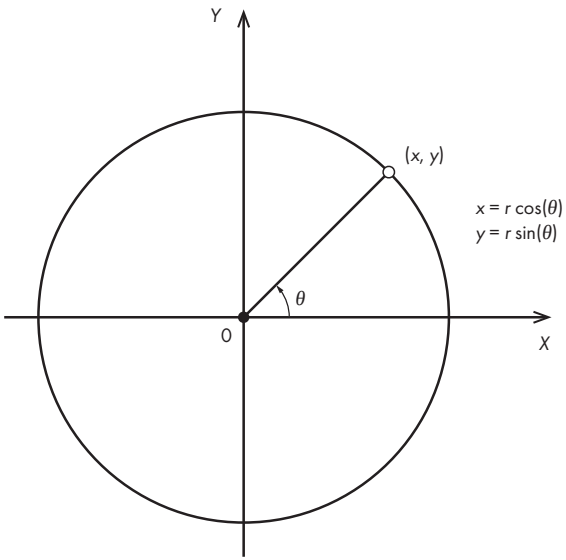


圖 2-2：用參數方程式描述一個圓

請記住，這兩個方程式適用於以座標系原點（origin）為圓心的圓。你可以將圓心從點 $(0, 0)$ 平移到點 (a, b)，從而將圓置於 XY 平面上的任意一點。如此，更一般化的參數方程式就變成了 $x = a + r\cos(\theta)$ 和 $y = b + r\sin(\theta)$。

建立萬花尺玩具的參數方程式模型與發展出圓的參數方程式並無太大區別，因為本質上，萬花尺只不過是在繪製兩個相互交錯的圓。圖 2-3 顯示了類似萬花尺運動方式的數學模型。這個模型沒有齒輪；萬花尺玩具中使用齒輪，

只是為了防止打滑，而在數學建模（mathematical modeling）的理想世界中，你不必擔心任何東西會打滑。

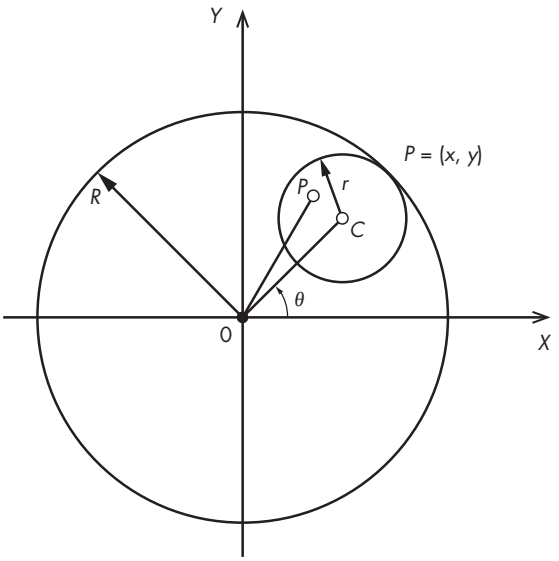


圖 2-3：萬花尺玩具的數學模型

在圖 2-3 中，C 是小圓的圓心，P 是筆尖，而 θ 是 C 相對於正 x 軸的角度。大圓的半徑是 R，小圓的半徑是 r。你可以用變數 k 來表示兩個半徑之比，如下：

$$k = \frac{r}{R}$$

線段 $\overline{PC}$ 告訴你筆尖距離小圓的圓心有多遠。你可以用變數 l 來表示 $\overline{PC}$ 與小圓半徑 r 的比值，就像這樣：

$$l = \frac{\overline{PC}}{r}$$

現在，你可以將這些變數合併為以下的參數方程式，表示小圓在大圓內旋轉時，P 點（筆尖）的 x 座標和 y 座標：

$$x = R\left((1 - \mathrm{k})\cos(\theta) + lk\cos\left(\frac{1-\mathrm{k}}{k}\theta\right)\right)$$

$$y = R\left((1 - \mathrm{k})\sin(\theta) - lk\sin\left(\frac{1-\mathrm{k}}{k}\theta\right)\right)$$

NOTE 這些曲線被稱為內旋輪線（*hypotrochoids*）。雖然方程式看起來有點嚇人，但推導起來卻非常簡單。如果你想了解更多數學知識，請參閱 Wikipedia 上的 Spirographs 頁面：*http://en.wikipedia.org/wiki/Spirograph*。

圖 2-4 是利用這些方程式所繪製的範例曲線。在繪製這個曲線時，我將 R 設為 220，r 設為 65，l 設為 0.8。為這三個參數選擇不同的值，然後遞增角度 θ，就可以繪製出無窮無盡的迷人曲線。

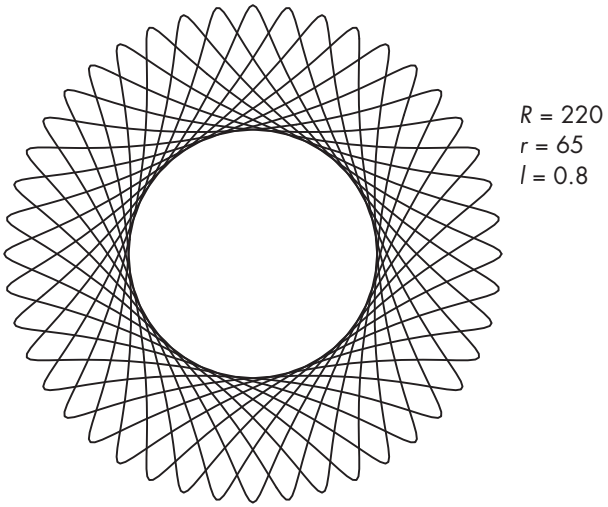


圖 2-4：範例曲線

剩下的唯一任務就是判斷何時停止繪製，因為萬花尺需要小圓繞大圓旋轉很多圈才能形成完整的圖案。你可以透過觀察內圓和外圓的半徑之比，計算出萬花尺的週期（*periodicity*，萬花尺要多久才會開始重複）：

$$\frac{r}{R}$$

將分子（numerator）和分母（denominator）除以最大公因數（*greatest common divisor*，*GCD*），從而化簡這個分數（fraction）。然後，分子就會告訴你曲線需要多少週期才能完成。舉例來說，在圖 2-4 中，(r, R) 的 GCD 是 5：

$$\frac{r}{R} = \frac{65}{220} = \frac{(65/5)}{(220/5)} = \frac{13}{44}$$

這告訴我們，小圓繞大圓轉 13 圈後，曲線將開始重複。分母中的 44 表示小圓繞自己圓心旋轉的次數，暗示了這個曲線的形狀。若你數一下圖 2-4 中的花瓣（或波瓣），就會發現正好有 44 個！

只要用簡化形式 r/R 表示半徑比，就會發現繪製 spiro 的參數 θ 之範圍是 [0, $2\pi r$]。這告訴你何時停止繪製 spiro。以圖 2-4 為例，當 θ 達到 26π（即 $2\pi \times 13$）時就會停止。若不知道角度的終止範圍，最後就會繞回來，沒必要地重複曲線。

執行萬花尺動畫

現在是時候執行你的程式了：

```
$ python spiro.py
```

預設情況下，*spiro.py* 程式會同時隨機繪製四個 spiros，如圖 2-5 所示。按 S 儲存影像，按 T 開關游標，按空白鍵重啟動畫。

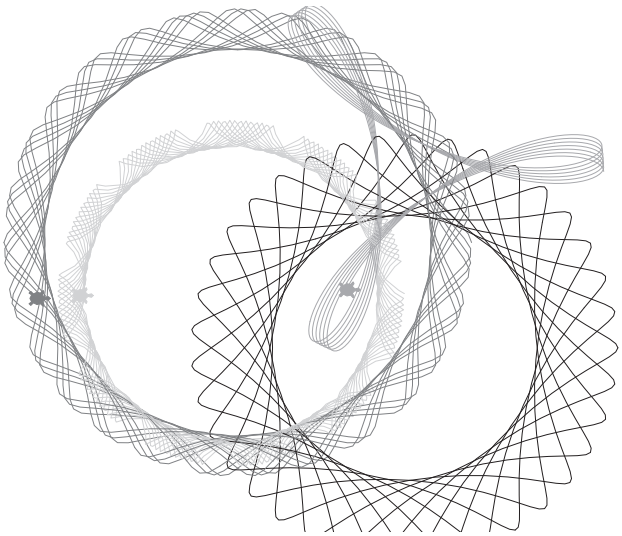


圖 2-5：*spiro.py* 的執行範例

現在再次執行程式，這次在命令列傳入參數，以繪製一個特定的 spiro：

```
$ python spiro.py --sparams 300 100 0.9
```

圖 2-6 顯示了輸出結果。如圖 2-6 所示，程式碼按照使用者指定的參數繪製了單個 spiro，而圖 2-5 則顯示繪製多個隨機 spiros 的動畫。

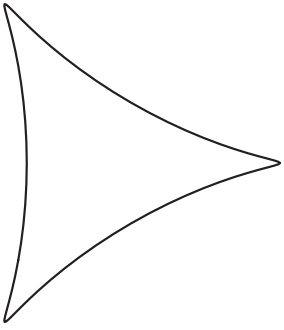


圖 2-6：帶特定參數的 *spiro.py* 執行範例

13

使用 Raspberry Pi 實現雷射音樂視覺效果

在第 12 章中，你使用微小的微控制器 Pico 來產生音調。在本章中，你將使用功能更強大的嵌入式系統 Raspberry Pi，根據聲音訊號產生有趣的雷射圖案。

上一章的 Pico 配備了帶有 ARM Cortex-M0 雙處理器的 RP2040 微控制器，執行速度可達 133 MHz，並具備 264 KB 的 RAM（random access memory）、和外部快閃記憶體晶片（flash chip）上 2MB 的非揮發性儲存空間（nonvolatile storage）。相較之下，Raspberry Pi 3B+ 的 ARM Cortex-A53 處理器效能要強得多，工作時脈為 1.4 GHz，擁有 1 GB 的 RAM 和數 GB 的儲存空間，具體取決於你所用的 SD 卡。雖然與標準的桌上型電腦或筆記型電腦相比，Raspberry Pi 仍顯遜色，但不同於 Pico，它能夠執行基於 Linux 的作業系統和完整的 Python。

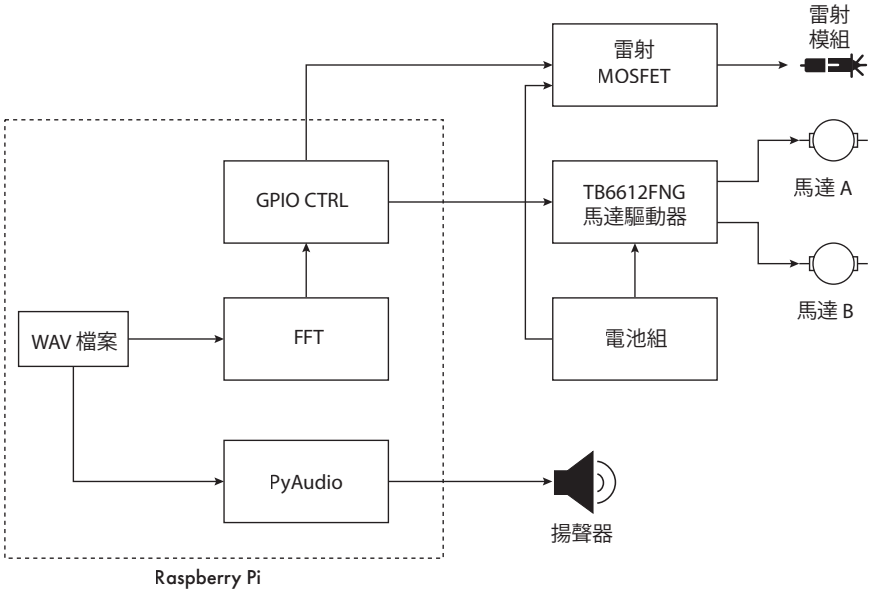


圖 13-1：雷射音訊專案的方塊圖

Raspberry Pi 將以兩種方式使用 WAV 檔案。它會透過 `pyaudio` 使用所連接的揚聲器播放檔案，同時使用一種稱為 *FFT*（*fast Fourier transform*，快速傅立葉變換）的數學技術即時分析音訊資料。Pi 會使用來自 FFT 的資料，藉由其 GPIO（general-purpose input/output）腳位驅動馬達和雷射，但為了保護 Pi 免受損壞，你不會將其直接連線到這些外部元件，而是透過馬達驅動器電路板（motor driver board）和 MOSFET 間接連線。開始之前，我們先來詳細了解一下專案的各個面向是如何運作的。

使用雷射產生圖案

在本專案中要產生雷射圖案，你會使用雷射模組（laser module）和接附到兩個小型 DC 馬達軸上的兩面反射鏡，如圖 13-2 所示。把雷射想像成一束強烈的光線，即使投射到很遠的地方，仍然能聚焦在很小的點上。之所以能夠聚焦，是因為光束的組織方式使其波形只朝一個方向傳播，並且彼此相位（phase）一致。如果將雷射投照到平面鏡（圖 13-2 中的鏡面 A）的表面，即使馬達在旋轉，反射投影也會保持在固定點上。因為雷射的反射面垂直於馬達的旋轉軸，所以鏡子就好像根本沒在旋轉一樣。

現在，假設鏡面與馬達軸（motor shaft）成一定角度，如圖 13-2 右側所示（鏡面 B）。隨著軸的旋轉，投影點的軌跡會畫出橢圓形，如果馬達旋轉得夠快，觀眾就會把移動的點看成連續的形狀。

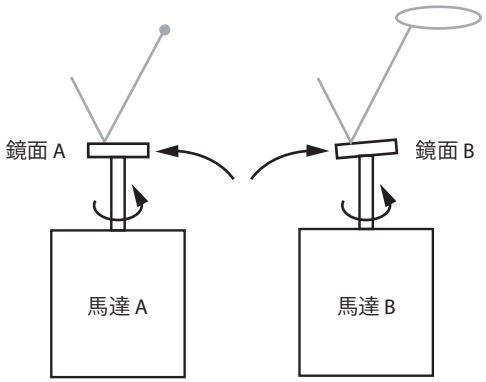


圖 13-2：平面鏡（鏡面 A）反射出一個點。傾斜鏡（鏡面 B）的反射點在馬達旋轉時形成一個圓。

如果兩面鏡子都是傾斜的，並且透過安排，使得從鏡面 A 反射出的點投射到鏡面 B 上，會怎樣呢？現在，當馬達 A 和 B 旋轉時，反射點產生的圖案將是馬達 A 和 B 兩個旋轉運動的組合，從而產生有趣的圖案，如圖 13-3 所示。

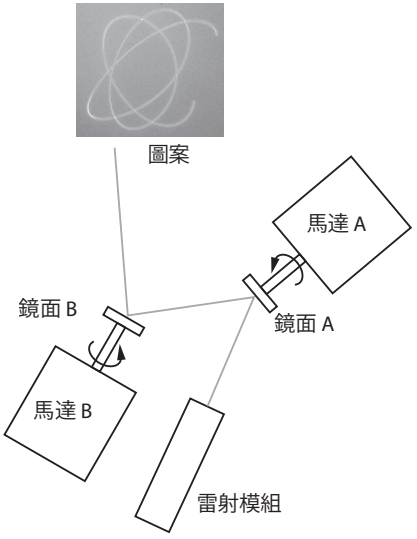


圖 13-3：將雷射從兩面旋轉的傾斜鏡面上反射出來，會產生有趣的複雜圖案。

產生的確切圖案取決於兩個馬達的轉速和旋轉方向，但它們將與第 2 章中探索的萬花尺所產生的內旋輪線（hypotrochoids）相似。

馬達控制

你將使用 Raspberry Pi 透過一種稱為 *PWM*（*pulse width modulation*，脈衝寬度調變）的技術來控制馬達的速度和方向。這是一種透過傳送快速接通和

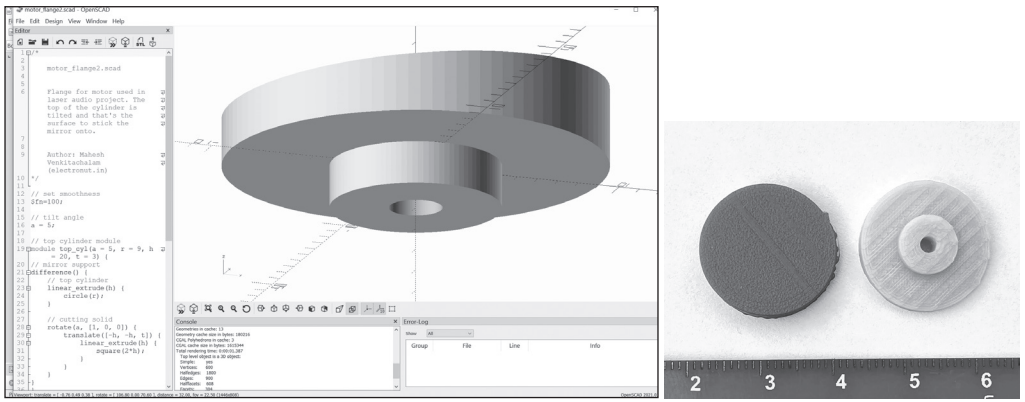
設定 Raspberry Pi

要設定 Raspberry Pi，請參閱附錄 B。按照附錄中的說明操作，並確保你已經安裝了本專案所需的 `numpy` 和 `pyaudio` Python 套件。你將透過 SSH（Secure Shell）在 Raspberry Pi 上編寫程式碼。你可以設定好 Microsoft Visual Studio Code，以便使用連自筆記型電腦或桌上型電腦的 SSH 在 Pi 上遠端作業。附錄 B 中對此也有說明。

建構雷射顯示器

在連接所有硬體之前，你應該為雷射顯示器準備好馬達和雷射模組。首要任務是將反射鏡安裝到馬達上。每個反射鏡都必須與馬達軸成略為傾斜的角度。一種方法是使用熱熔膠。接附鏡子時，將鏡子面朝下放在平坦表面上，然後在中心點滴一滴熱熔膠。小心地將馬達軸浸入膠中，使其與鏡面保持接近垂直但稍微傾斜的角度，直到熱熔膠變硬。

更好的辦法是使用帶有斜面的馬達盤（motor flange），這樣就可以輕易將鏡子黏在上面。但是要到哪裡去找這樣的元件呢？你可以使用 3D 列印技術自己製作！圖 13-10(a) 是我使用免費開源程式 OpenSCAD 製作的 3D 設計圖。你可以從本書的 GitHub 儲存庫下載該設計。圖 13-10(b) 顯示的是 3D 列印出來的元件。雷射首先擊中的鏡子將使用傾斜度較小（5 度）的馬達盤，第二個鏡子則使用傾斜度較大（10 度）的馬達盤。



(a) (b)

圖 13-10：OpenSCAD 模型 (a) 和 3D 列印出來的馬達盤 (b)

如果你有 3D 列印機，可以自行列印馬達盤，或者透過 3D 列印服務列印（無論哪種方法，成本都不高）。拿到元件後，用強力瞬間膠將馬達盤固定到馬達軸上，並將鏡子黏到馬達盤上。圖 13-11 顯示了完全組裝好的元件。

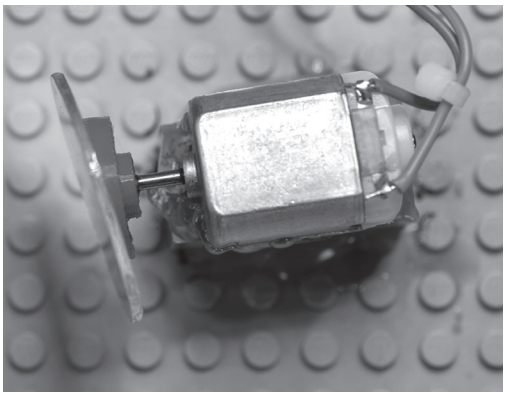


圖 13-11：將反射鏡以略為傾斜的角度安裝到每個馬達軸上

要測試組合好的元件，請用手旋轉鏡子，同時用雷射模組照射它。投影到平面上時，你應該會發現雷射點的反射會呈橢圓形移動。也對第二面鏡子做同樣的事。由於相對於馬達軸的角度較大，它應該會形成更寬的橢圓。

排列鏡面

接下來，排列好雷射模組與反射鏡，使雷射從鏡面 A 反射到鏡面 B，如圖 13-12 所示。確保在鏡面 A 的整個旋轉範圍內，從鏡面 A 反射的雷射光都維持在鏡面 B 的圓周內（這需要一些試驗）。要測試這種佈置，可手動旋轉鏡面 A。此外，還要確保鏡面 B 的位置正確，使得從其表面反射的光線在兩個鏡面的完整旋轉範圍內都落在某個平坦的表面（如牆壁）上。

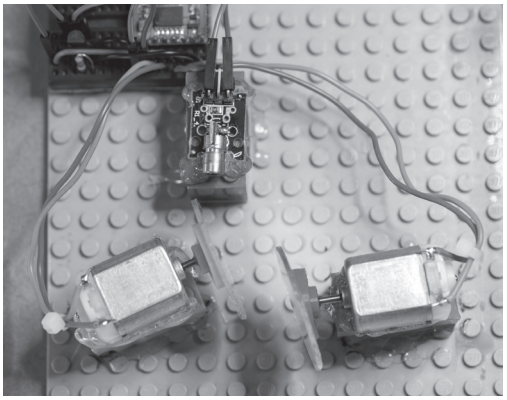


圖 13-12：雷射器和反射鏡的排列

NOTE 在調整排列方式時，需要讓雷射指示器保持開啟狀態。要那麼做，請像這樣執行專案程式碼：`python laser_audio.py --test_laser`。此命令單純只是開啟控制雷射模組的 MOSFET，我們將在本章後面討論。