中華民國第 65 屆中小學科學展覽會

作品說明書

國中組　生活與應用科學科(一)

032806

眾樂樂合成樂團

學校名稱： 臺中市立爽文國民中學

作者：	指導老師：
國一 施天海	李雅純
國一 吳振銘	洪碧霞

關鍵詞： Web Audio API、聲音合成器

摘要

本研究透過運用 AI 技術輔助程式設計聲音合成器程式，以探索聲音波形生成與觀察分析聲音頻，實驗透過不同波形振盪器，並調控 ADSR 參數，與濾波器組合，我們用不同的合成方法，獲得各類樂器聲音的最佳合成參數，再以先前完成的自製樂器加以改進，透過 Arduino 偵測串聯電阻分壓電路傳遞琴鍵按壓的演奏訊號至電腦，再經由合成器運算合成輸出對應的樂音，藉由選定不同樂器模組實現樂團合奏。在研究過程中，為解決電腦端即時運算合成音色的耗時問題，我們預先合成並暫存不同樂器音色，提升演奏時的反應速度。最終，本研究在開源程式與 AI 的輔助下開發出一套支援多人即興演奏的音樂合成器系統，提供創新的音樂創作與演出方式，降低傳統樂器練習的成本與噪音問題。

壹、前言

一、研究動機

在學習各種樂器的過程中，許多練習用的樂器價格不菲，特別是像馬林巴琴這樣昂貴的樂器。此外，在家中練習時，樂器的音量可能會打擾鄰居，限制了隨時隨地盡情演奏的可能性。

傳統的音樂創作主要依賴現有樂器或數位音樂軟體，但這些方式受限於樂器種類與預設音色，限制了創作的多樣性。

受到網路上許多樂團利用日常物品創作音樂的啟發，我們希望延續之前的「STOMP-自製電音 BAND」專案[1]，進一步探討聲音合成理論，開發基於 Arduino 的自製琴鍵與網頁聲音合成器，實現多種樂器音色的即時合成與多人合奏功能。

在研究初期，我們運用人工智慧檢索相關研究，發現過去的研究中已有類似的作品。例如，民國 72 年的「新式電腦音樂合成器」[2]與民國 106 年的「MIDI 樂器（電子琴）」[3]等。

本研究期待結合上述科展研究的優點，進一步延伸。計畫使用 Arduino 作為接收控制、輸入音樂訊號，並由電腦控制聲音合成程式並發送聲音。

此外鍵盤結構設計，改用自製鍵盤的方式，將以模組化可抽換方式，以符合我們可以演奏更多不同樂器的目標。

通過這些改進，我們希望開發出一種可自由調整音色的電子樂器，讓使用者能夠隨時創作並演奏自己喜愛的音樂，並解決傳統樂器價格高昂和練習時噪音干擾的問題。

二、 研究目的及問題

(一)聲音合成技術相關：

如何透過不同的波形振盪器（例如正弦波、方波、鋸齒波）與濾波器組合，模擬馬林巴琴、鋼琴與鼓等樂器的音色？

如何調整 ADSR（Attack, Decay, Sustain, Release）參數，使合成的樂器音色更接近真實樂器？

(二)硬體設計相關：

如何設計一套模組化的 Arduino 琴鍵電路裝置，使其能夠方便更換並對應不同的樂器音色？

(三)應用功能相關：

如何實現一套基於 Web Audio API 的聲音合成器系統，讓多人同時進行即興演奏並輸出合奏效果？

如何解決電腦端即時運算合成音色的耗時問題，提升多人演奏時的即時反應速度？

三、 文獻探討：

(一)聲音合成技術相關：

1. 加法合成器（Additive Synthesizer）是一種聲音合成技術，廣泛應用於音樂與聲音設計領域。其基本原理是透過將多個簡單的波形（通常是正弦波）疊加，創造出更複雜的音色。這種方法源自傅立葉分析理論，該理論指出任何複雜聲音都可以分解為一系列不同頻率與振幅的正弦波分量。加法合成器利用這一概念，允許使用者控制各個正弦波的頻率、振幅，甚至包絡（例如開始-衰減-保持-釋放，ADSR），以模擬真實樂器音色或創造全新聲音。

加法合成的歷史可追溯至早期電子樂器，例如 1930 年代問世的 Hammond 風琴。這款樂器使用「音輪」（tonewheel）技術，結合多個正弦波產生豐富的音色，並透過拉桿讓使用者調整不同頻率的強度，形成千變萬化的聲音效果。相較於其他合成方法，如減法合成（從複雜波形中移除部分頻率）或頻率調變合成（FM），加法合成的特點在於精細控制每一個聲音分量，提供高度的靈活性。然而，這也帶來挑戰：需要同時操控大量參數，早期硬體受限於技術，實現起來較為複雜。

在現代，加法合成已廣泛應用於軟體合成器中，例如 Ableton Live 的 Operator 或 Logic Pro 的 Sculpture 插件。這些工具讓使用者能輕鬆調整多個振盪器（oscillator）的參數，並即時聽到結果。與傳統硬體相比，數位技術還允許使用波表或逆傅立葉變換等方法，提升

效率與聲音品質。值得一提的是，加法合成的應用不僅限於音樂創作，還出現在電影音效設計與遊戲聲音開發中，用於模擬自然聲或製造科幻效果。

2. 減法合成器（Subtractive Synthesizer）是一種常見的聲音合成技術，廣泛應用於電子音樂創作與聲音設計中。其**基本原理是從一個初始的複雜波形（如鋸齒波或方波）開始，透過濾波器（filter）移除特定的頻率成分，進而塑造所需的音色。**與加法合成器將多個簡單波形疊加不同，減法合成的核心在於「減少」與「雕琢」，從豐富的聲音素材中提取出目標音調。這種方法操作相對直觀，且能產生溫暖、動態的音色，因此成為合成器設計的主流技術之一。

減法合成的歷史與電子音樂的發展密切相關。最早的商業化減法合成器可追溯至 1960 年代，如羅伯特·穆格（Robert Moog）發明的 Moog 合成器。這款設備使用振盪器（oscillator）產生初始波形，並配備低通濾波器（low-pass filter）移除高頻部分，讓聲音從尖銳變得柔和。使用者還可透過包絡生成器（envelope generator）調整音量或濾波器的變化，形成類似真實樂器的起伏感。這些特性使減法合成器在流行音樂、搖滾與電子舞曲中廣受歡迎，例如 1970 年代的迪斯可音樂與現代的浩室（House）曲風。

減法合成器的基本結構包括振盪器、濾波器與放大器三大模組。振盪器負責產生初始波形（如鋸齒波、方波或三角波），這些波形通常富含諧波（harmonics）。濾波器則決定哪些頻率被保留或移除，例如低通濾波器保留低頻、高通濾波器保留高頻。最後，放大器控制聲音的整體音量，並與包絡配合調整聲音的動態。現代軟體合成器，如 Serum 或 Massive，將這些功能數位化，讓使用者能更精確地調整參數，甚至即時預覽效果。

3. 頻率調製合成 (Frequency Modulation Synthesis, FM Synthesis) 頻率調製合成 (FM Synthesis) 是一種音頻合成技術，通過使用一個波形（調製器/Modulator）來調製另一個波形（載波/Carrier）的頻率 (Frequency)，以產生複雜且動態的音色 (Timbre)。這種技術因其能夠生成豐富的諧波 (Harmonics) 和獨特的聲音紋理可以生成遠超簡單振盪器的豐富諧波結構。FM 合成的優勢在於其能夠以少量計算資源生成高度動態的音色，和生成複雜音色的能力而聞名。

(1)金屬音色：高調製指數和非整數頻率比可產生鈴聲或金屬敲擊聲。

(2)打擊樂：快速衰減的包絡適用於鼓聲或敲擊聲。

(3)鍵盤音色：例如電鋼琴或 DX7 風格的音色，常用於流行音樂。

(4)環境音效：緩慢變化的調製指數可用於創建氛圍音景。

但其挑戰在於參數設置的複雜性，需要經驗來精確控制音色。

FM 合成通過一個振盪器 (Oscillator) 的輸出改變另一個振盪器的頻率，從而生成新的諧波結構。其核心思想是讓調製器波形快速改變載波波形的頻率，產生複雜的音色變化。

(1)載波 (Carrier)：最終輸出的聲音信號，其頻率是可聽的主要音調。

(2)調製器 (Modulator)：用於改變載波頻率的振盪器，其輸出影響載波的頻率變化。

(3)調製指數 (Modulation Index)：控制調製器對載波頻率影響的程度，通常決定生成的諧波數量和音色複雜度。

FM 合成的音色由以下幾個參數決定，這些參數允許音樂家精確控制聲音的特性，載波與調製器頻率比 (C:M Ratio)載波頻率 (fc) 與調製器頻率 (fm) 的比例對音色有顯著影響。

常見的頻率比	FM 合成的輸出
整數比（例如 1:1、2:1）：產生和諧的音色，類似管風琴或弦樂	$y(t) = Ac * \sin(2\pi * fct + I * \sin(2\pi * fmt))$ Ac：載波振幅 (Carrier Amplitude) fc：載波頻率 (Carrier Frequency) fm：調製器頻率 (Modulator Frequency) I：調製指數 (Modulation Index)
非整數比（例如 1:1.414）：生成非諧波音色，類似金屬或打擊樂聲音	

表 1-1 FM 合成頻率比

調製指數 I 越大，生成的諧波越多，音色越豐富；反之，I 較小時，音色更接近簡單的正弦波。調製指數 (I) 決定調製器改變載波頻率的幅度。較高的調製指數會增加側頻帶 (Sidebands)，使音色更複雜。

(二)Arduino 微電腦琴鍵電路裝置

Arduino 是一款開源的微電腦控制器板，搭載微控制器（通常為 Atmel AVR 系列），並提供整合開發環境（IDE）供使用者編寫程式。其主要特色包括：

1、 開發簡單：支援 C/C++ 語言，內建豐富函數庫，適合初學者。

2、 參考資料豐富：擁有龐大的社群支持與線上資源。

3、模組化設計：可連接多種電子元件，如 LED、感測器、馬達等。

常見型號包括 Arduino UNO 和 Arduino NANO：

1、 Arduino UNO：尺寸約為 68.6mm x 53.4mm，適合通用應用。

2、Arduino NANO：更小型化（43mm x 18mm），適合嵌入式項目。 兩者均透過 USB 線與電腦連接，供電並傳輸程式。

Arduino 可搭配多種電子裝置實現自動控制應用，例如：

1、感測器控制：如利用溫濕度感測器控制風扇。

2、輸出控制：如用可變電阻調節 LED 亮度或馬達轉速。

3、通訊應用：如紅外線遙控家電或藍牙模組數據傳輸。

4、機械應用：如伺服機控制機械手臂或製作自走車。

圖 1-1 Arduino UNO

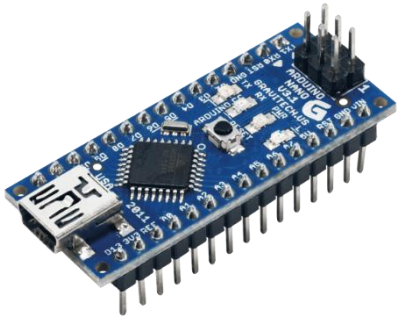


圖 1-2 Arduino NANO

在本研究中， 如果要使用 Arduino 測量外部電壓，則必須使用 Arduino 開發板的模擬輸入引腳(A0 至 A5 腳位)，因 Arduino 的輸入電壓限制為 5V，這時 Arduino 的模擬輸入引腳僅能測試最高 5V 的電壓。因為 Arduino 所用 AVR 晶片為 10 位元 AD，所以此模組的類比解析度為 0.00489V（5V/1023）。

預計使用 Arduino UNO 作為實驗用微電腦，負責讀取演奏感測器訊號並透過序列通訊傳送到網頁，實現音樂演奏的硬體基礎。其具備較多的開源腳位，並支援較多的類比電壓輸入腳位，我們要接受不同類比訊號時，有較多支援，所以預計以 Arduino UNO(圖 2-1)作為我們研究操作的電子材料囉！

設計琴鍵鍵盤時為求能達到至少 8 度音域 60 個琴鍵， 僅用 Arduino 數位輸入的方式明顯不足，由參考文獻[1]得知，可使用電阻分壓的原理，設計變電壓輸出的電路 ，引導 Arduino 感測按壓琴鍵鍵盤依按壓位置不同，經由按鍵開關導通串接不同數量的電阻值的電路迴路，偵測到不同的電壓值即擴展更多音階被偵測到。

電阻分壓公式是一個基本的電路定律，用於計算兩個串聯電阻如何分壓。當兩個電阻 R1 和 R2 串聯連接到電源電壓 Vin 時，輸出電壓 Vout （在 R2 兩端的電壓）可由以下公式計算：

$$Vout=Vin \times \frac{R2}{R1+R2}$$

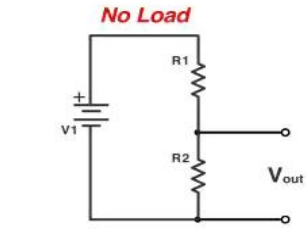
圖 1-3 電阻分壓原理	分壓公式推導
 $V_{out} = V_1 \frac{IR_2}{I(R_1 + R_2)} = \frac{V_1 R_2}{(R_1 + R_2)}$	串聯電路的特性： (1)串聯電阻的總電阻 Rt = R1+R2 。 (2)電流 I 由歐姆定律 Vout=I*Rt 推算 ：I = Vi/ (R1+R2) (3)根據歐姆定律，電阻 R2 兩端的電壓為： Vout = I * R2 代入 I = Vi / (R1+R2) Vout = (Vi / (R1+R2)) * R2 整理後得到分壓公式 **輸出電壓 = 輸入電壓 * 電阻 2 / (電阻 1 + 電阻 2)** **Vout = Vi * R2 / (R1+R2)**

表 1-2 電阻分壓公式推導

上拉電阻 設電阻 r1=0

則 輸出電壓 5V = 輸入電壓 5V * 電阻 r2/(電阻 r2+ 0)

下拉電阻 設電阻 r2=0 則 輸出電壓 0v = 輸入電壓 5V * 0 /(0+電阻 r1)

應用分壓公式推算鍵盤電路迴路所需的電阻值

漸變遞增 Vout 腳位偵測電壓 推導公式 求 電阻 r2 最佳解

Vout = vcc * r2 / (r2 + r1) 分壓公式

(r2+r1) * Vout = vcc*r2

(r2* Vout +r1 * Vout) = vcc*r2

r1 * Vout = vcc*r2 - r2* Vout

r1 * Vout = r2 * (vcc - Vout)

(r1 * Vout)/(vcc - Vout) = r2 * (vcc - Vout) /(vcc - Vout)

(r1 * Vout)/(vcc - Vout) = r2 * 1/1

(r1 * Vout)/(vcc - Vout) = r2

類比輸入 A0 到 A5，每個腳位負責偵測，以電阻分壓公式 5V 電壓分配 12 個琴鍵推算出每一 組琴鍵電壓值差距為 0.42v 計算鍵盤電路迴路所需的串接的電阻值如下

琴鍵 1 [r2 電阻 0.00Ω = 電壓 0.00v * 1097Ω) / (5v -0.00v)]

琴鍵 2 [r2 電阻 99.73Ω = 電壓 0.42v * 1097Ω) / (5v -0.42v)]

琴鍵 3 [r2 電阻 219.40Ω = 電壓 0.83v * 1097Ω) / (5v -0.83v)]

琴鍵 4 [r2 電阻 365.67Ω = 電壓 1.25v * 1097Ω) / (5v -1.25v)]

琴鍵 5 [r2 電阻 548.50Ω = 電壓 1.67v * 1097Ω) / (5v -1.67v)]

琴鍵 6 [r2 電阻 783.57Ω = 電壓 2.08v * 1097Ω) / (5v -2.08v)]

琴鍵 7 [r2 電阻 1097.00Ω = 電壓 2.50v * 1097Ω) / (5v -2.50v)]

琴鍵 8 [r2 電阻 1535.80Ω = 電壓 2.92v * 1097Ω) / (5v -2.92v)]

琴鍵 9 [r2 電阻 2194.00Ω = 電壓 3.33v * 1097Ω) / (5v -3.33v)]

琴鍵 10 [r2 電阻 3291.00Ω = 電壓 3.75v * 1097Ω) / (5v -3.75v)]

琴鍵 11 [r2 電阻 5485.00Ω = 電壓 4.17v * 1097Ω) / (5v -4.17v)] -

琴鍵 12 [r2 電阻 12067.00Ω = 電壓 4.58v * 1097Ω) / (5v -4.58v)]

(三) 應用 Web Serial API 連接 Arduino 微控制器

Web Serial API 是一個新興的網頁介面，允許瀏覽器與序列埠設備（如 Arduino、微控制器）進行通訊。將其應用於聲音合成器設計，可實現硬體與網頁的互動，提升合成器的控制體驗。

1、**功能概述**：Web Serial API 提供讀取和寫入序列埠數據的能力，支援與外部硬體設備的雙向通訊。使用者可通過瀏覽器直接控制硬體，例如琴鍵按鈕或各種感測器，來調節聲音參數。

2、**硬體控制**：透過序列埠接收外部設備的輸入（如琴鍵按壓的電壓變化數值）將電壓變化以 JSON 字串編碼後傳輸到電腦設備，由電腦網頁應用程式接收到序列資料後解碼處理。

Web Serial API 限制僅在安全上下文（HTTPS）下運行，且需要使用者明確授權。需注意仍有瀏覽器相容性問題：目前僅 Chrome、Edge 等瀏覽器支援。網頁應用程式需處理序列埠通訊的延遲與錯誤管理。

範例程式碼展示如何使用 Web Serial API 讀取外部設備數據並調整音頻頻率

```
async function connectSerial() {
  const port = await navigator.serial.requestPort();
  await port.open({ baudRate: 115200 }); //連線速率
  const reader = port.readable.getReader();
  const ctx = new AudioContext();
  const oscillator = ctx.createOscillator(); //
  oscillator.type = 'sine'; //波形
  oscillator.connect(ctx.destination);
  oscillator.start();

  while (true) {
    const { value, done } = await reader.read();
    if (done) break;
    const frequency = new TextDecoder().decode(value).trim() ;
     // 假設輸入範圍映射到頻率
    oscillator.frequency.setValueAtTime(frequency, ctx.currentTime);
  }
}
// 需要使用者明確授權 主動按下連線按鈕 才能開始進行通訊連結
document.getElementById('connect').addEventListener('click',
connectSerial);
```

(四)應用 Web Audio API 設計聲音合成器

Web Audio API 是一種用於網頁應用程式中控制和合成聲音的 JavaScript API。參考文獻[6] 它於 2011 年開始發展，現已被現代瀏覽器（如 Chrome、Firefox、Safari）廣泛支持。透過 Web Audio API，我們可以在網頁上創建音樂、音效，甚至是聲音合成器，特別適合用來學習程式設計和音樂的結合。

Web Audio API 的核心是音頻圖（audio graph），這是一個由多個音頻節點（audio nodes）組成的系統。這些節點就像聲音工廠中的機器，負責產生、處理和輸出聲音。音頻圖的組成包括：

音源節點 （Source Nodes）	產生聲音	例如振盪器（OscillatorNode）可以生成不同波形的聲音
處理節點 （Effect Nodes）	處理聲音	例如增益節點（GainNode）控制音量，濾波器節點（BiquadFilterNode）改變聲音頻譜。
目標節點 （Destination Nodes）	輸出聲音	通常是揚聲器（context.destination）

表 1-3 音頻圖的組成

這些節點通過 connect() 方法連接，形成一個音頻處理鏈。例如，振盪器連接到增益節點，增益節點再連接到揚聲器，最終輸出聲音。核心元件：AudioContext 是音頻操作的中心樞紐，所有的音頻處理都在這個上下文中進行。創建方式如下：

```
const context = new AudioContext();
```

振盪器（OscillatorNode）是生成聲音的基礎節點。它可以產生不同類型的波形，包括：

波形	正弦波 （sine）	方波 （square）	鋸齒波 （sawtooth）	三角波 （triangle）
音色	聲音柔和，類似笛聲	聲音尖銳，類似早期電子遊戲音效	聲音粗糙，類似弦樂	介於正弦波和方波之間

表 1-4 振盪器產生的波型和音色

波形決定了聲音的音色，而頻率（以赫茲 Hz 為單位）決定了音調。聲音的音調由頻率決定，頻率越高，音調越高。以下是常用音符的頻率表

音符	C4	D4	E4	F4	G4	A4	B4
頻率(Hz)	261.63	293.66	329.63	349.23	392.00	440.00	493.88

表 1-5 常用音符的頻率表

在聲音合成器設計中，ADSR 是一個非常重要的概念，它是用來描述聲音包絡（envelope）的標準模型。ADSR 有四個階段組成：

聲音包絡	A(Attack)	D(Decay）	S（Sustain）	R（Releas）
描述聲音隨時間變化的音量模式	聲音從無到有的階段，即音量從 0 增加到最大值的時間。例如，按下鋼琴鍵時，聲音會迅速變大。	聲音從最大值下降到一個穩定值的階段。例如，鋼琴聲音在達到最大音量後會稍微減弱，但仍持續發聲。	聲音在穩定狀態下的音量，這個階段會持續到你鬆開按鍵。例如，風琴聲音在按住鍵盤時會保持穩定音量。	聲音從穩定值衰減到 0 的階段，即鬆開按鍵後聲音逐漸消失的時間。例如，鋼琴聲音在鬆開鍵後會慢慢減弱。
GainNode 設定音量隨時間的變化	音量從 0 上升到 1（最大值）	音量從 1 下降到某個穩定值（Sustain 水平）	音量保持在穩定值，直到鬆開按鍵	音量從穩定值下降到 0

表 1-6 ADSR 四個階段

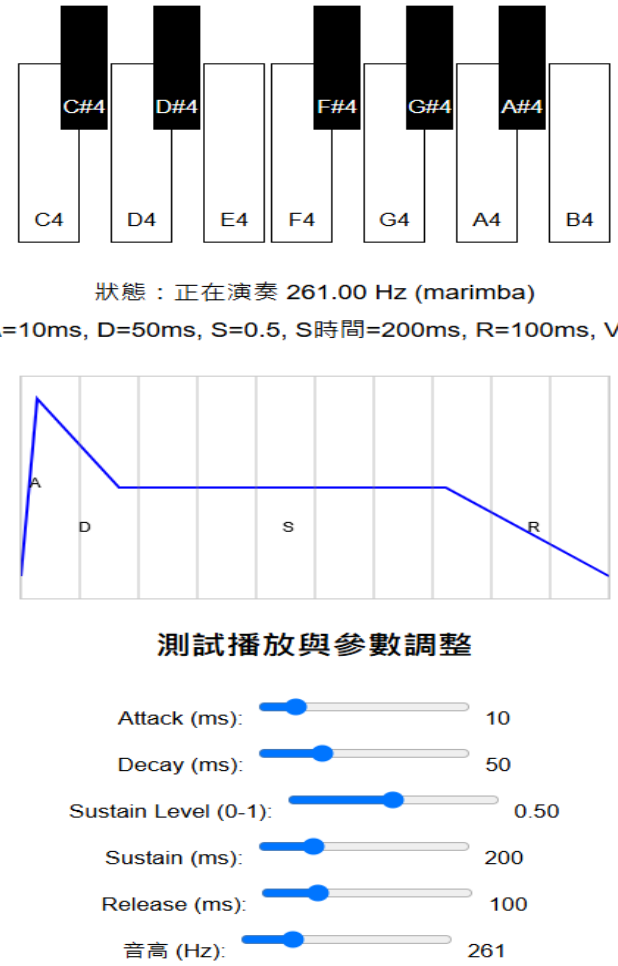


圖 1-4 web Audio api ADSR 範例程式

透過調整 ADSR 的參數，我們可以讓聲音合成器模擬各種樂器的聲音，甚至創造全新的音色。

聲音特性 / 樂器種類	A(Attack) 攻擊	D(Decay） 衰減	S（Sustain） 保持	R（Releas） 釋放
鋼琴	快速且短	短暫的衰減	無保持，為 0	中等長度的釋放
風琴	快速且短	無衰減，為 0	長時間的保持且高	快速的釋放且短
小提琴	緩慢且長	短暫的衰減	長時間的保持	緩慢的釋放且長

表 1-7 樂器種類和聲音特性表

(五)應用 HTML5 Canvas 2D 繪製圖表與互動動畫

HTML5 Canvas 2D 提供了一個強大的繪圖介面，可用於聲音合成器的視覺化呈現，增強互動性和視覺吸引力。

結合 Web Audio API，Canvas 2D 可以即時繪製音頻波形、頻譜或其他視覺效果。	1、**動態繪圖功能**：使用 CanvasRenderingContext2D 繪製 2D 圖形，如線條、曲線或圖形。 2、**音頻視覺化**：結合 AnalyserNode 獲取音頻的時域或頻域數據，繪製波形圖或頻譜圖。 3、**互動性**：透過滑鼠或觸控事件，允許使用者調整聲音參數並即時更新視覺效果。
應用於聲音合成器	1、**波形視覺化**：顯示當前聲音的波形，幫助使用者理解聲音的動態變化。 2、**頻譜分析**：繪製實時頻譜圖，展示不同頻率的能量分佈。 3、**控制介面**：設計互動式旋鈕或滑桿，結合 Web Audio API 動態調整音頻參數。
挑戰	1、**性能問題**：高頻率繪圖可能導致性能瓶頸，需優化渲染邏輯。 2、不同螢幕解析度的適配問題，需使用相對單位或動態縮放。

表 1-8 結合 Web Audio API，Canvas 2D 應用於聲音合成器分析表

範例程式碼展示如何使用 Canvas 2D 繪製實時音頻波形：

```
// 創建一個 AudioContext 物件，用於管理音頻處理的上下文
const ctx = new AudioContext();
// 創建一個 AnalyserNode，用於分析音頻數據（如時域或頻域數據）
const analyser = ctx.createAnalyser();
// 設定分析器的 FFT（快速傅里葉變換）大小，決定頻譜分析的解析度，
2048 表示高解析度
analyser.fftSize = 2048;
// 創建一個 OscillatorNode，用於生成基本波形的聲音
const oscillator = ctx.createOscillator();
// 設定振盪器的波形類型為正弦波（'sine'），可選值包括 'square'、
'sawtooth'、'triangle'
oscillator.type = 'sine';
// 將振盪器連接到分析器節點，以便分析生成的音頻數據
oscillator.connect(analyser);
// 將分析器連接到 AudioContext 的輸出（通常是揚聲器），以播放聲音
analyser.connect(ctx.destination);
// 啟動振盪器，開始生成聲音
oscillator.start();
// 獲取 HTML 中的 <canvas> 元素，用於繪製音頻波形
```

```
const canvas = document.getElementById('waveCanvas');
// 獲取 Canvas 的 2D 繪圖上下文，用於執行繪圖操作
const canvasCtx = canvas.getContext('2d');
// 獲取分析器的頻域數據長度，與 fftSize 相關（通常為 fftSize / 2）
const bufferLength = analyser.frequencyBinCount;
// 創建一個 Uint8Array 陣列，用於儲存時域數據（音頻波形的振幅數據）
const dataArray = new Uint8Array(bufferLength);
// 定義 draw 函數，用於動態繪製音頻波形
function draw() {
// 使用 requestAnimationFrame 實現動畫迴圈，確保平滑更新畫面
requestAnimationFrame(draw);
// 從分析器獲取當前的時域數據（波形數據），儲存到 dataArray
analyser.getByteTimeDomainData(dataArray);
// 設定畫布的填充顏色為白色（RGB: 255, 255, 255）
canvasCtx.fillStyle = 'rgb(255, 255, 255)';
// 清除畫布並填充白色背景，範圍為整個畫布大小
canvasCtx.fillRect(0, 0, canvas.width, canvas.height);
// 設定繪製線條的寬度為 2 像素
canvasCtx.lineWidth = 2;
// 設定線條顏色為黑色（RGB: 0, 0, 0）
canvasCtx.strokeStyle = 'rgb(0, 0, 0)';
// 開始一個新的繪圖路徑
canvasCtx.beginPath();
// 計算每個數據點在畫布上的水平間距（畫布寬度除以數據長度）
const sliceWidth = canvas.width / bufferLength;
// 初始化水平座標 x 為 0
let x = 0;
// 遍歷時域數據，將每個數據點轉換為畫布上的座標並繪製
for (let i = 0; i < bufferLength; i++) {
// 將數據值（0-255）標準化為 -1 到 1 的範圍（128 是中間值）
const v = dataArray[i] / 128.0;
```

```
// 將標準化值映射到畫布的垂直座標（高度的一半）
const y = (v * canvas.height) / 2;
// 如果是第一個數據點，移動畫筆到起點
if (i === 0) {
canvasCtx.moveTo(x, y);} else {
// 否則，繪製一條線到當前點
canvasCtx.lineTo(x, y);}
// 更新水平座標，移動到下一個數據點的位置
x += sliceWidth; }
// 繪製一條線到畫布右邊界的中間高度，確保波形閉合
canvasCtx.lineTo(canvas.width, canvas.height / 2);
// 執行繪圖操作，渲染波形線條
canvasCtx.stroke();
// 啟動動畫，開始繪製波形
draw();
```

貳、研究設備及器材

一、研究設備：

windows 系統筆電一台、Arduino UNO 一片 、三用電表一台

二、研究器材：

cat6 網路雙絞線、電焊工具組、 PCB 電路板、可變電阻、精密電阻、瓦楞紙版、塑膠瓦楞版、鋁箔紙、銅箔貼紙、絕緣膠帶、透明膠帶、雙面膠帶、熱熔膠/熱熔膠槍

參、研究過程或方法

(一)研究架構圖：

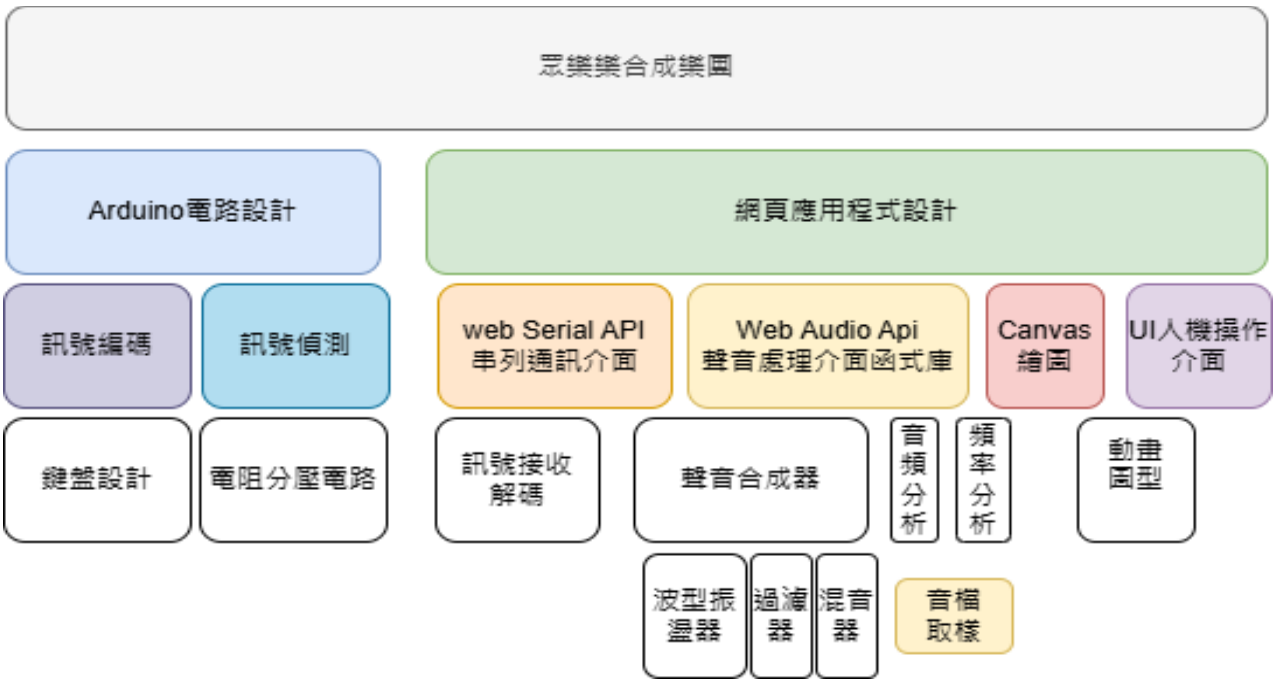


圖 3-1 研究架構

(二) 研究一：聲音分析與比對

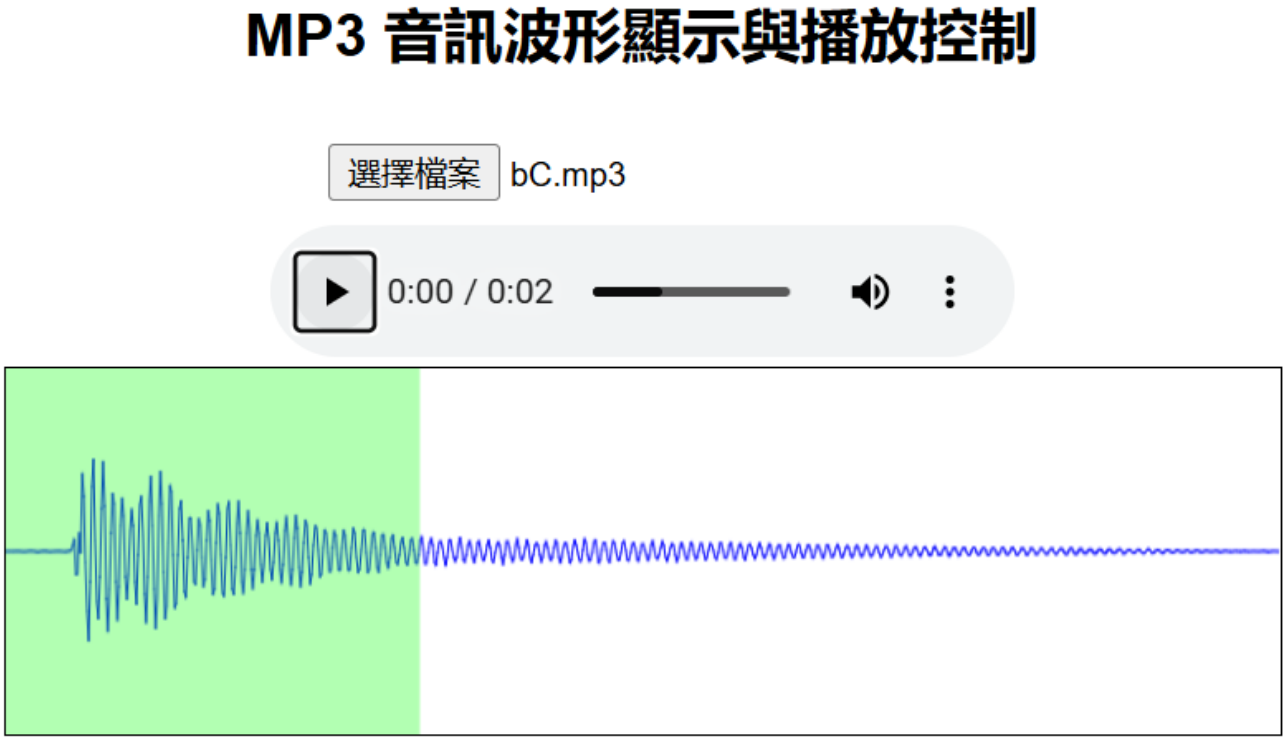


圖 3-2 讀取聲音 顯示聲波形圖

	馬林巴琴	鋼琴	大鼓
以 Audacity 軟體進行初步聲音處理，生成波形圖與頻譜圖、成為合成音的參考基準	基頻與少量泛音為主，衰減較快	豐富的諧波，具較長的釋放階段	呈現低頻主導的特徵，伴隨短暫的噪聲成分
利用 Web Audio API 開發的聲音合成器生成對應音色	正弦波（sine wave）作為主振盪器	結合正弦波與三角波	混合低頻正弦波（60 Hz）與白噪聲
	低通濾波器（截止頻率 1500 Hz）過濾高頻	低通濾波器（2500 Hz）	低通濾波器（150 Hz）
	ADSR 設為 A: 0.005s, D: 0.5s, S: 0.05, R: 0.4s；	ADSR 為 A: 0.01s, D: 0.6s, S: 0.5, R: 0.8s	ADSR 為 A: 0.01s, D: 0.25s, S: 0, R: 0.2s
合成音生成後，同樣以 Audacity 分析其波形與頻譜，並與錄音比對。	合成音的衰減曲線與真實錄音高度相似，但泛音略顯單薄，建議加入微量鋸齒波提升豐富度	合成音的初始攻擊與釋放階段接近真實，但諧波分佈不足，需增加多個振盪器模擬複雜泛音	合成音的低頻特性還原良好，但噪聲成分偏強，需調整噪聲比例並細化濾波器設定

表 2-1 不同樂器合成頻譜表

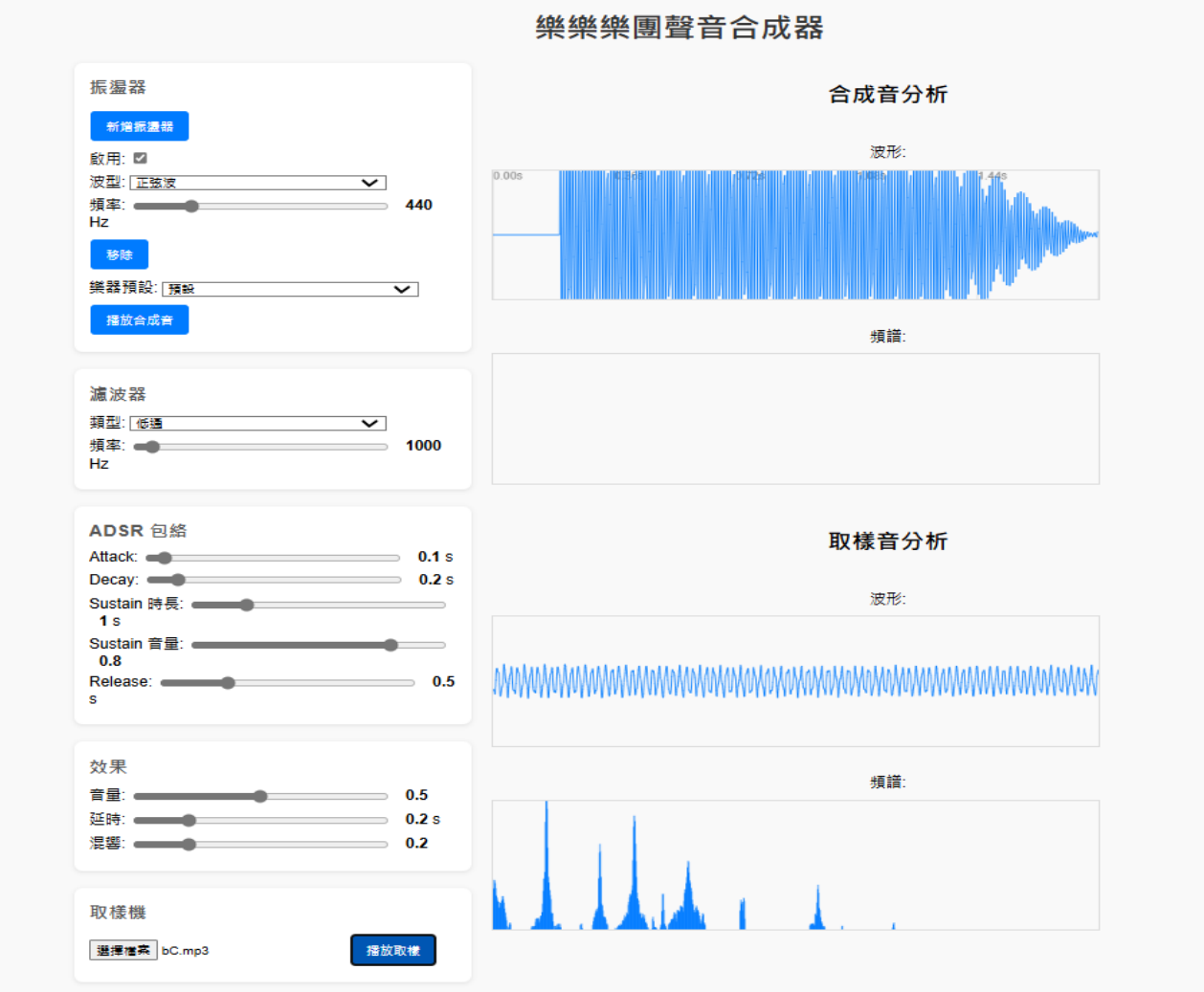


圖 3-3 觀察比較合成聲音與取樣音檔聲音波形

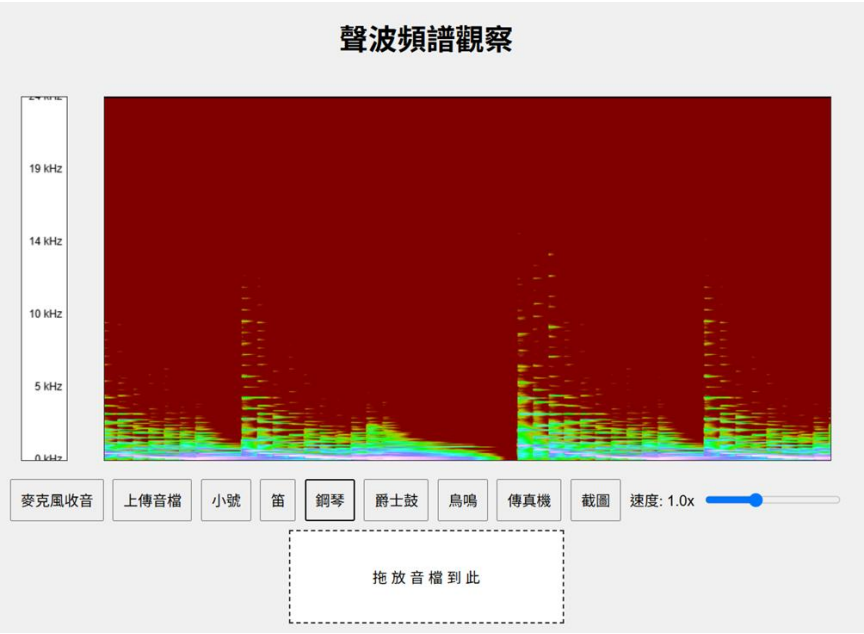


圖 3-4 觀察各種聲音頻譜分佈圖 (/0428/x006.html)

本研究透過錄製真實樂器聲音並與合成音進行波形與頻譜比對，分析波形振盪器、濾波器及 ADSR（Attack, Decay, Sustain, Release）包絡參數對音色的影響，進而找出模擬馬林巴琴、鋼琴與鼓等樂器的最佳參數。

本研究確立了各樂器的初步合成模型，並記錄其波形特性與控制參數，為後續硬體與軟體整合提供了數據基礎。

(三) 研究二：合成器的探討

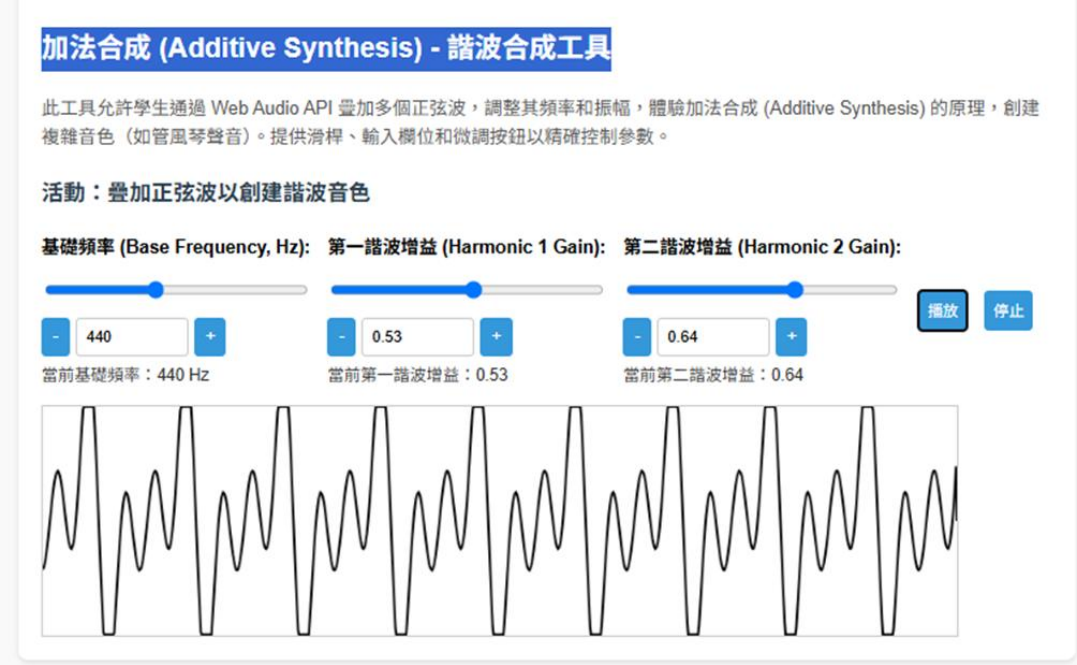


圖 3-5 諧波合成圖工具圖

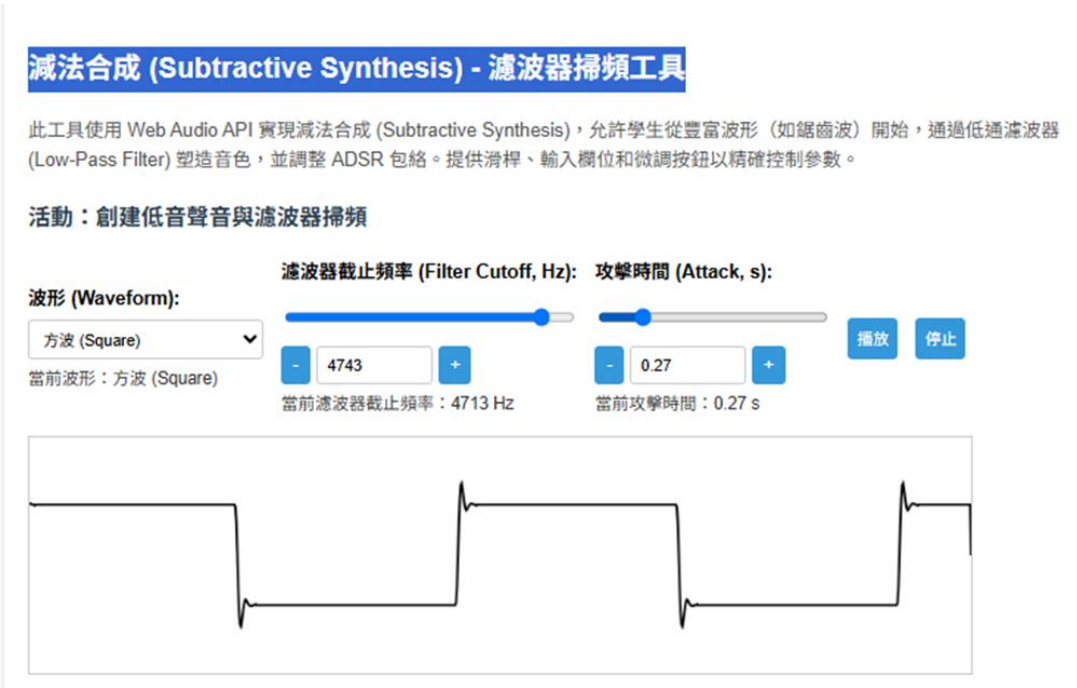


圖 3-6 濾波器掃頻工具圖

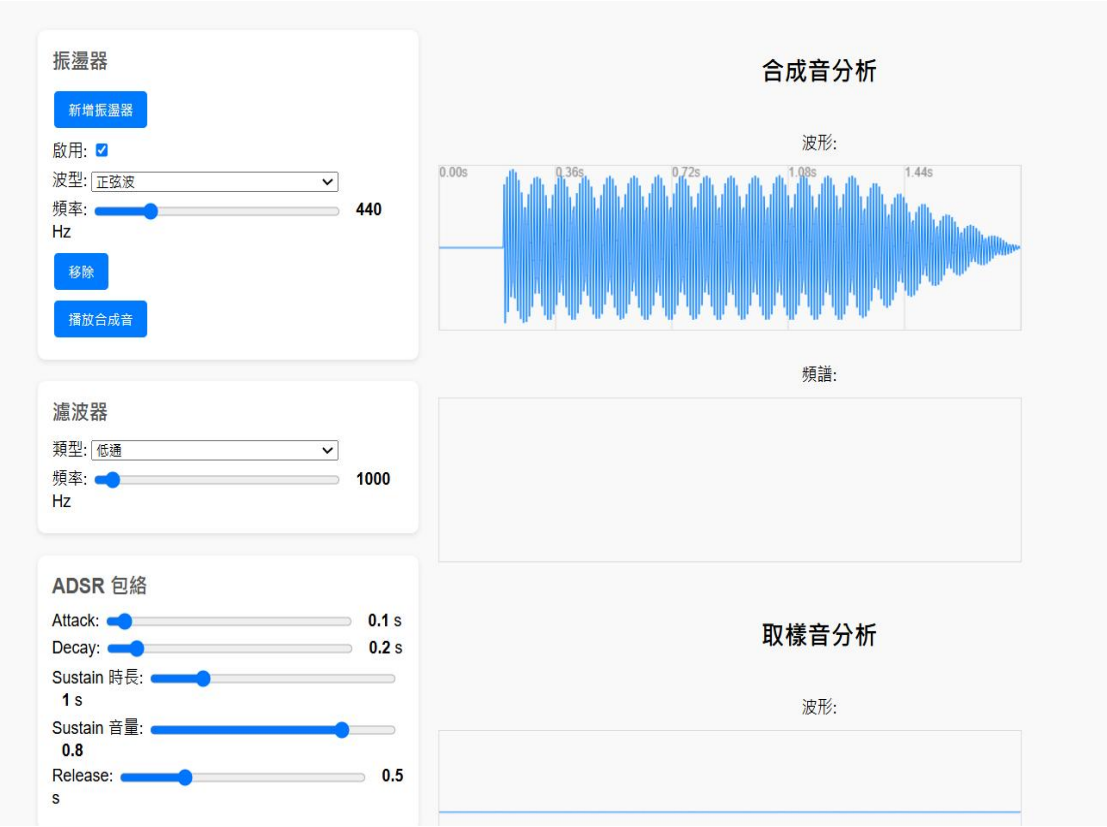


圖 3-7 Web Audio API 設計聲音合成器

聲音合成技術是電子音樂與互動媒體的重要組成部分。傳統的硬體合成器（如 Moog 或 Yamaha DX7）雖然功能強大，但成本高且不易整合到現代應用中。隨著 Web Audio API 的發展，網頁端的聲音合成成為可能，且具有跨平台、低成本的優勢。本研究利用 HTML、JavaScript 和 Web Audio API，設計了一款網頁合成器，並針對不同樂器聲音的特徵，優化其合成參數，以接近真實樂器的聽覺效果。合成器由以下模組組成：

- **振盪器（Oscillator）**：生成基礎波形（正弦波、方波、鋸齒波、三角波、白噪聲、粉紅噪聲）。
- **濾波器（Filter）**：調整頻率成分（低通、高通、帶通）。
- **ADSR 包絡（Envelope）**：控制音量隨時間變化的特性（Attack、Decay、Sustain、Release）。
- **效果處理**：包括音量控制、延時（Delay）和混響（Reverb）。
- **取樣機**：支援上傳外部音檔並播放。
- **分析**： AnalyserNode 獲取音頻的時域數據 取出聲音波形數值轉換為頻譜數值。

這些模組透過 Web Audio API 的音頻節點（AudioNode）串聯，形成完整的聲音處理鏈。

1、程式核心功能

	功能	實現
振盪器模組	允許用戶新增多個振盪器，每個振盪器可獨立設定波形類型與頻率	用 audioContext.createOscillator() 生成標準波形。透過 createNoise() 函數生成白噪聲與粉噪聲。每個振盪器可啟用/停用，並動態添加或移除。
濾波器	透過雙二階濾波器（Biquad Filter）過濾頻率成分	audioContext.createBiquadFilter() 支援低通、高通、帶通濾波。頻率範圍為 20-20000 Hz，用戶可透過滑桿調整。
ADSR 包絡	模擬樂器聲音的起音、衰減、持續與釋放階段	使用 gainNode.gain 的 linearRampToValueAtTime 方法，按時間線性變化音量。參數包括： Attack：起音時間（0-2 秒）。 Decay：衰減時間（0-2 秒）。 Sustain Time：持續時間（0-5 秒）。 Sustain Level：持續音量（0-1）。 Release：釋放時間（0-2 秒）。
效果處理	音量：透過 gainNode.gain 控制整體音量。 延時：使用 audioContext.createDelay()，延時時間範圍為 0-1 秒。 混響：透過 audioContext.createConvolver() 生成簡單的脈衝響應，模擬空間感。	
視覺化	波形：使用 OfflineAudioContext 渲染合成音波形，並以 Canvas 繪製。 頻譜：透過 AnalyserNode 實時分析頻率分佈，顯示頻譜圖。	

表 3-1 程式核心功能分析表

2、聲音合成流程

(1)用戶設定振盪器、濾波器、ADSR 和效果參數。

(2)點擊「播放合成音」按鈕，觸發 playSynth() 函數。

(3)合成器生成聲音，並同時渲染波形與頻譜。

(4)聲音透過濾波器、延時與混響處理後輸出至揚聲器。

3、如何合成真實樂器聲音

要合成接近真實樂器的聲音，需根據樂器的物理特性與聲學特徵調整參數：

(1)波形選擇：

a、弦樂（如小提琴、吉他）常用鋸齒波，模擬弓弦或撥弦的豐富諧波。

b、打擊樂（如馬林巴琴、鋼琴）偏向正弦波，強調基頻與簡單泛音。

c、鼓聲（如大鼓、小鼓）結合低頻正弦波與噪聲，模擬鼓皮振動與噪聲成分。

d、管樂（如長笛）使用正弦波，突出純淨的音色。

(2)ADSR 包絡：

a、短促打擊樂（如馬林巴琴、大鼓）具有快速 Attack 和短 Sustain。

b、持續性樂器（如小提琴、長笛）需要較長的 Sustain 與平滑的 Release。

(3)濾波器：

a、低通濾波器用於柔化尖銳的高頻（如鋼琴、吉他）。

b、高通濾波器用於去除低頻噪聲（如長笛）。

c、帶通濾波器適用於強調特定頻段（如小鼓）。

(4)效果：

a、響模擬演奏環境的空間感（如鋼琴、小提琴）。

b、延時增加聲音的層次感（如吉他）。

4、樂器參數設定表格

以下為程式中預設的樂器參數，根據聲學特徵優化：

表 3-1 不同樂器程式中預設的樂器參數

樂器	振盪器設置	濾波器	ADSR 參數	音量	延時 (s)	混響
馬林巴琴	1. 正弦波 (261.63 Hz) 2. 正弦波 (523.25 Hz) 3. 正弦波 (784.00 Hz)	低通 (1500 Hz)	A: 0.005, D: 0.5, ST: 0.1, SL: 0.05, R: 0.4	0.7	0.05	0.3
吉他	1. 鋸齒波 (110 Hz) 2. 鋸齒波 (220 Hz) 3. 方波 (330 Hz)	低通 (1000 Hz)	A: 0.02, D: 0.4, ST: 1.5, SL: 0.3, R: 0.6	0.5	0.15	0.2
大鼓	1. 正弦波 (60 Hz) 2. 白噪聲 (100 Hz)	低通 (150 Hz)	A: 0.01, D: 0.25, ST: 0, SL: 0, R: 0.2	0.8	0	0.1
小鼓	1. 三角波 (180 Hz) 2. 白噪聲 (500 Hz) 3. 粉噪聲 (1000 Hz)	帶通 (2000 Hz)	A: 0.01, D: 0.2, ST: 0, SL: 0, R: 0.15	0.7	0	0.15
鋼琴	1. 正弦波 (261.63 Hz) 2. 正弦波 (523.25 Hz) 3. 三角波 (784.00 Hz)	低通 (2500 Hz)	A: 0.01, D: 0.6, ST: 1.5, SL: 0.5, R: 0.8	0.6	0.1	0.25
長笛	1. 正弦波 (523.25 Hz) 2. 正弦波 (1046.50 Hz) 3. 正弦波 (1568.00 Hz)	高通 (700 Hz)	A: 0.15, D: 0.1, ST: 2, SL: 0.7, R: 0.3	0.5	0.05	0.2
小提琴	1. 鋸齒波 (261.63 Hz 2. 鋸齒波 (523.25 Hz) 3. 鋸齒波 (784.00 Hz)	低通 (3500 Hz)	A: 0.25, D: 0.15, ST: 2, SL: 0.6, R: 0.5	0.5	0.2	0.3

註釋：

- A: Attack (起音), D: Decay (衰減), ST: Sustain Time (持續時間), SL: Sustain Level (持續音量), R: Release (釋放)。
- 頻率值（如 261.63 Hz）為 C4 音符及其倍頻泛音，模擬和聲結構。

(四)程式設計

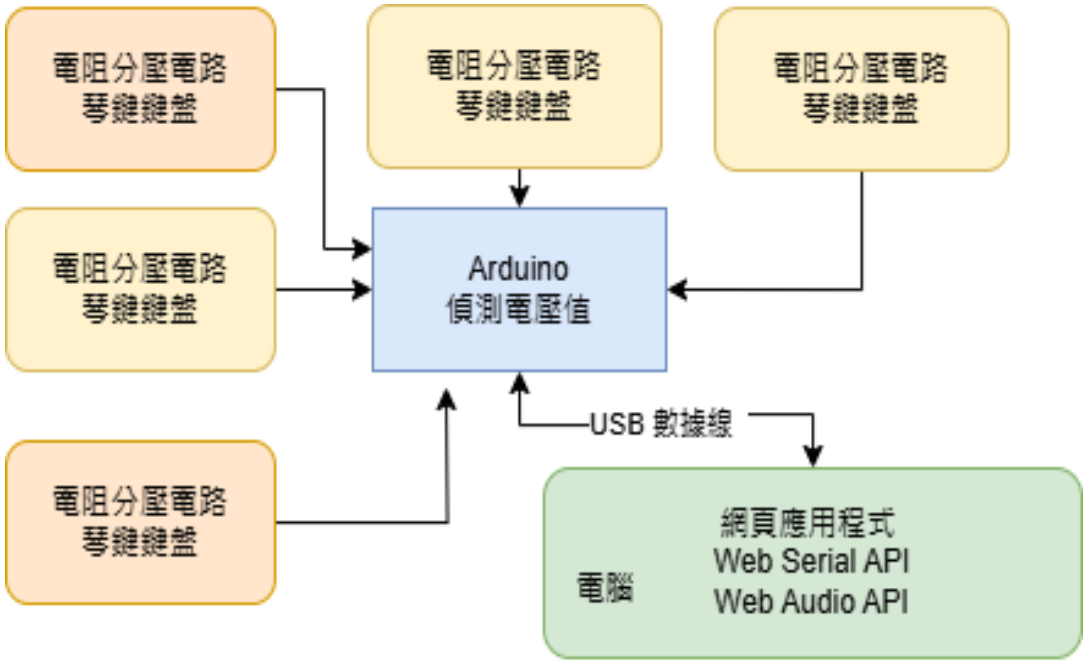


圖 3-8 程式設計流程圖

本研究在程式設計的部分，主要分為 Arduino 的電壓偵測將偵測到的類比電壓演奏訊號, 轉換成 json 字串格式 用串列通訊 傳送到筆電,網頁應用程式以 Web Serial API 接收與再由 web Audio API 合成播放聲音 的自製網頁聲音合成器，其分述如下：

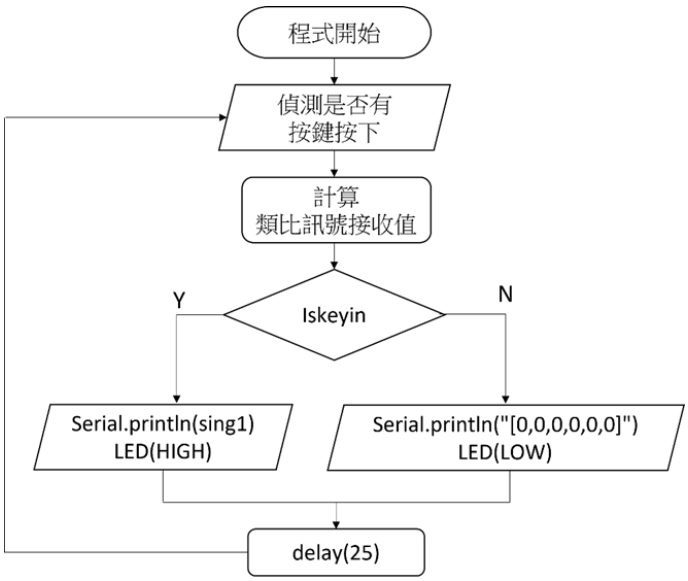


圖 3-9 訊號傳送流程圖

a、Arduino 的程式設計

b、偵測類比電壓訊號（如來自感測器的演奏輸入）。

c、將電壓值轉換為結構化的 JSON 字串。

透過串列通訊（Serial Communication）傳送到筆電。

(五)網頁應用程式端的程式設計

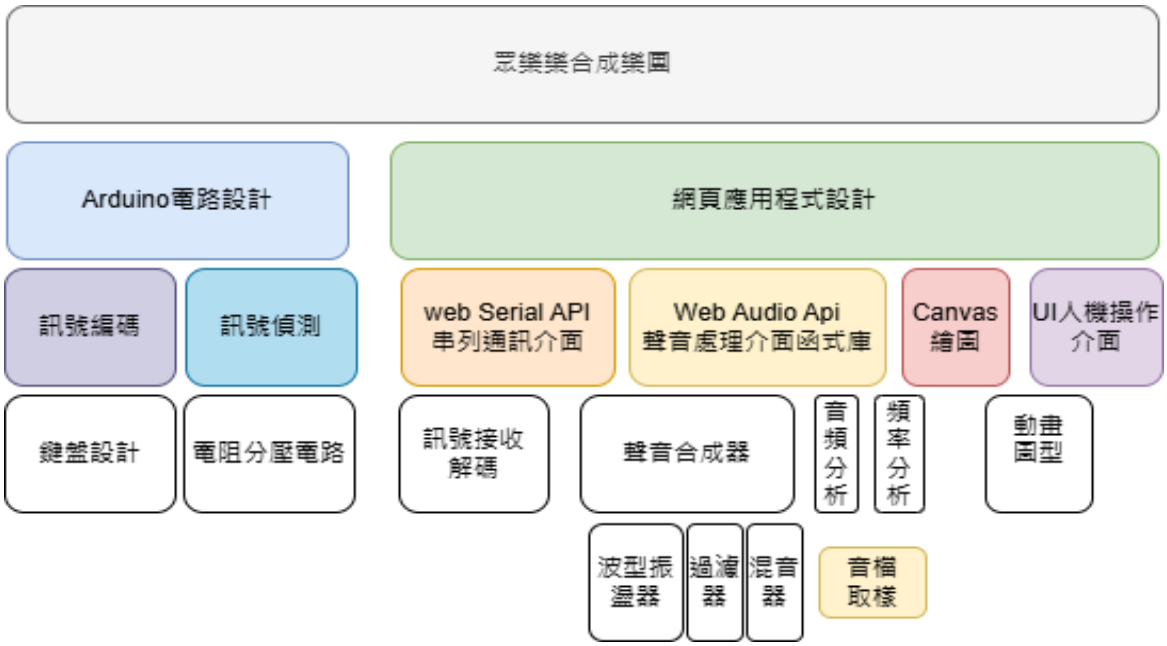


圖 3-10 網頁應用程式端的程式設計圖

1、 使用 Web Serial API 接收 Arduino 傳來的 JSON 數據

(1)anavigator.serial.requestPort() 讓用戶選擇 Arduino 的串口。

(2)port.readable.getReader() 持續讀取串口數據。

(3)接收到的 JSON 字串被解析為物件，提取頻率值。

2、 使用 Web Audio API 根據接收到的頻率數據合成並播放聲音

(1)AudioContext 是聲音合成的核心物件。

(2)OscillatorNode 生成指定頻率的聲波（這裡使用正弦波）。

(3)GainNode 控制音量，防止聲音過大。

(4) 每次接收到新頻率時，播放短暫的聲音（0.1 秒）。

(六)整體運作流程

1、琴鍵的按壓 透過串聯電阻分壓電路,由 Arduino 偵測感測器電壓 A0 ~ A5 ，將其轉換打包成 JSON 字串（例如 [1023,1023,1023,1023,1023,1023]）。

2、JSON 字串透過 USB 串口傳送到筆電。

3、網頁應用程式使用 Web Serial API 接收並解析 JSON 數據。程式比對類比電壓值換算成對應樂器演奏的音符代號。

4、根據解析出的代號查表得到，合成音色的各項參數(音色,音符.頻率)，交由 Web Audio API 合成對應 的聲音並播放。

(七)程式實現細節

1、 振盪器生成

(1)程式支援動態添加振盪器，每個振盪器透過 addOscillator() 函數創建，並可選擇波形與頻率。

(2)噪聲生成使用自定義算法（白噪聲為隨機值，粉噪聲為濾波後的噪聲）。

2、 ADSR 實現

(1) applyADSR() 函數根據用戶設定的參數，動態調整 gainNode 的音量曲線。

(2)使用線性斜坡（linearRampToValueAtTime）實現平滑過渡。

3、 效果處理

(1)混響透過 createReverb() 函數生成隨機衰減的脈衝響應，模擬自然殘響。

(2)延時透過 DelayNode 實現，參數可調。

4、 視覺化

(1)波形使用 OfflineAudioContext 預渲染，確保精確性。

(2)頻譜透過 AnalyserNode 實時計算並繪製。

5、 參數儲存與匯出

(1) saveParamsButton 將當前設定儲存為 JSON 檔案。

(2)exportAudioButton 目前僅播放聲音，需引入第三方庫（如 lamejs）實現 MP3 匯出。

肆、研究結果及討論

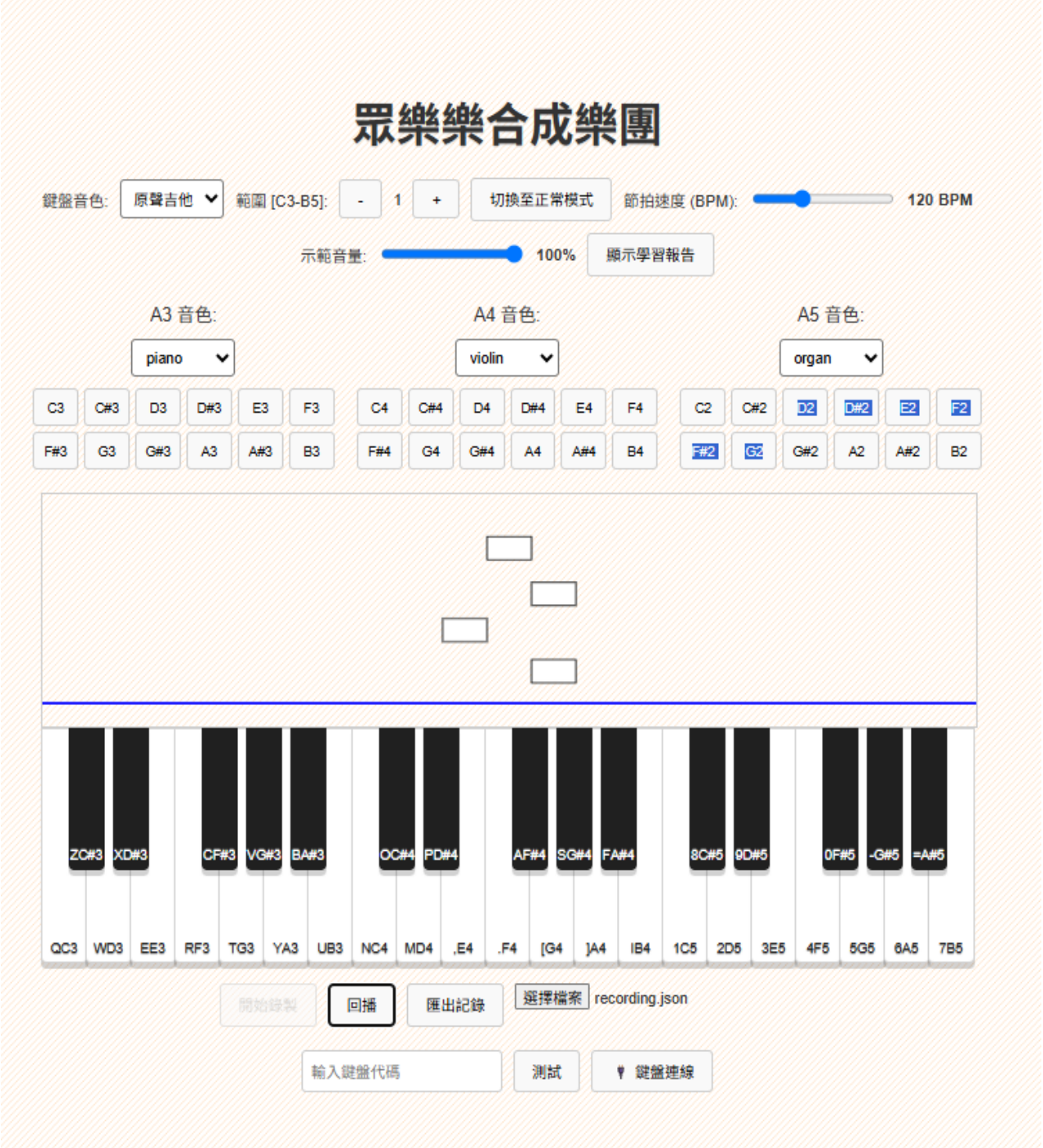


圖 4-1. 演奏播放 使用者操作介面外觀

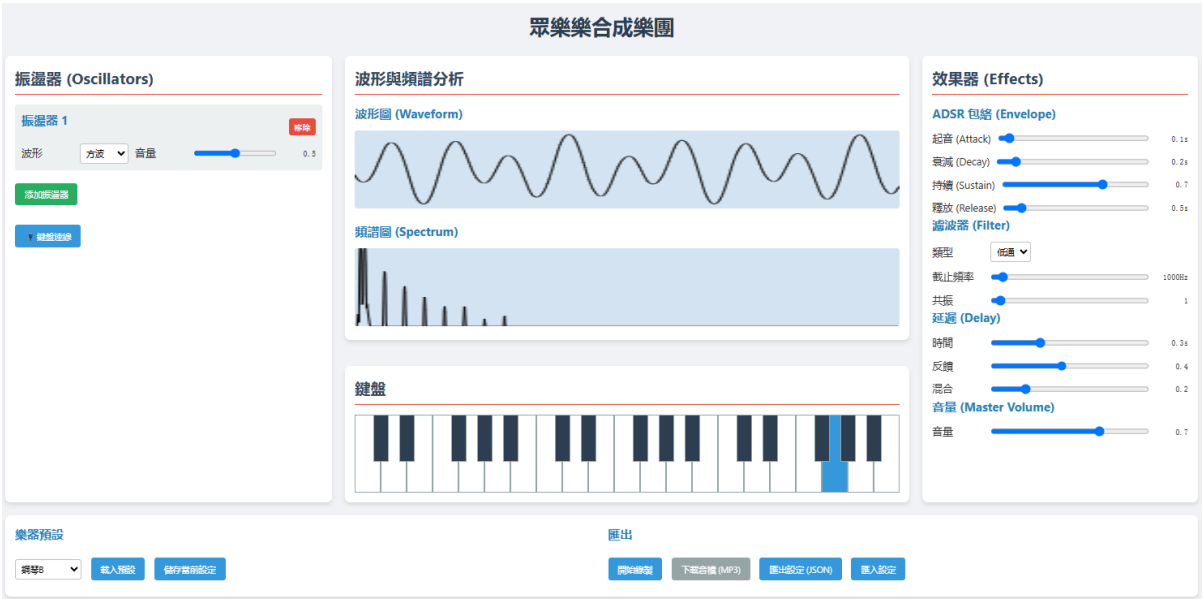


圖 4-2 聲音合成器使用者介面

本研究參考引用開源程式[ ],開發了網頁應用程式，該系統透過 Arduino 的序列介接（Serial Interface）讀取演奏訊號，並根據不同樂器設定播放事先合成好的音檔。研究重點在於實現 Arduino 與網頁之間的即時通訊，利用 Web Serial API 接收演奏訊號，並透過 Web Audio API 播放預載音檔。本報告詳細說明程式設計原理，特別針對序列通訊模組（application.js）進行分析，並闡述如何根據演奏訊號觸發不同樂器的音檔播放。

一、系統設計與功能說明

(一)整體架構

系統由以下主要部分組成：

1、**琴鍵樂器硬體設計：**利用串聯電阻分壓電路實現琴鍵輸入，每個琴鍵開關對應一個設定的電阻值，連接到 Arduino 的類比輸入腳位。依按下琴鍵所改變電阻值，產生不同電壓，經 Arduino 的 10 位元 ADC（0-1023）轉換為數位值，解析度約 4.9mV（5V/1024）。

2、**資料處理：**Arduino 讀取類比輸入腳位的電壓類比值，根據電壓範圍映射到特定琴鍵（音符），透過 analogRead() 獲取數值，程式內判斷電壓範圍對應的琴鍵。

3、**JSON 生成與傳輸：**Arduino 將琴鍵資料（如音符名稱、頻率）封裝成 JSON 格式，透過序列埠（Serial）以 115200 波特率傳送到電腦，網頁應用程式使用 Web Serial API 接收 JSON 訊號並解碼。

4、**網頁應用程式：**使用 JavaScript 解析 JSON，根據音符資料驅動 Web Audio API 合成音檔,並存入音頻緩衝區 播放對應音頻，並以 Canvas 2D API 顯示互動動畫。

5、網頁應用程式：

(1)**虛擬鍵盤介面**：提供手動觸發音檔的鍵盤與按鈕。

(2)**序列通訊模組**：使用 Web Serial API 接收 Arduino 數據（application.js）。

(3)**音檔播放模組**：使用 Web Audio API 播放預載音檔（假設在 audioPreload.js 中）。

(4)**視覺化模組**：以 Canvas 繪製類比訊號長條圖（drawbar.js）。

(二)程式核心功能

1、 **虛擬鍵盤與音色選擇**：

實現了一個涵蓋 C3 到 B5 的 36 鍵虛擬鍵盤，支援鋼琴、管風琴、原聲吉他、電音、木琴、小提琴、長笛、銅管和低音等九種音色。使用者可通過自製 Arduino 硬體連線，接收（A0-A5）輸入並映射到鍵盤按鍵或按鈕，實現外部硬體控制。（鍵盤輸入（Q、Z、W 等按鍵）或滑鼠點擊虛擬鍵盤觸發音符，並可選擇不同音色來改變音效。

(1)**鍵盤映射**：keyboard 物件定義了鍵盤按鍵（keyCode）與音符（例如 'C,3'）的對應關係，支援 QWERTY 鍵盤輸入。

(2)**動態生成鍵盤**：fnCreateKeyboard 函數根據音符範圍（C3-B5）動態創建黑鍵和白鍵，設置 CSS 樣式以模擬真實鋼琴鍵盤。

(3)**音色選擇**：通過 <select> 元素動態填充可用音色，支援多個音色選擇器（主鍵盤及 A3-A5 按鈕組）。

(4)**事件處理**：使用 evtListener 根據設備類型（行動裝置或桌面）綁定觸控或滑鼠事件，確保跨平台相容性。

2、 **Arduino 硬體整合**：

支援通過 Web Serial API 與 Arduino 硬體連線整合，接收模擬輸入（A0-A5）並映射到鍵盤按鍵或按鈕，實現外部硬體控制。即時顯示電壓數據的波形圖，提供直觀的硬體反饋，幫助除錯。

(1)**串口通信**：connectArduino 使用 Web Serial API 建立與 Arduino 的連線，解析串口數據流並觸發 detectKeyCode。

(2)**電壓映射**：getKeyCodeA0 等函數根據電壓閾值將模擬輸入轉換為鍵盤按鍵或按鈕索引，支援 A0-A2（鍵盤）和 A3-A5（按鈕組）。

(3)**波形視覺化**：reDraw 函數使用 Canvas 繪製電壓波形圖，顯示 A0-A5 的即時數據，輔助硬體除錯。

3、 **錄製與回播功能：**

使用者可以錄製演奏的音符序列，包括音高、八度、音色、開始時間和持續時間，並支援回播功能。playback 根據錄製的音符序列和節拍速度（tempo）定時播放音符，確保與原演奏同步。**錄製邏輯：startRecording 清空錄製陣列並啟動錄製，fnPlayKeyboard 在錄製模式下將音符資訊（包括時間戳）存入 recording 陣列。**錄製的數據可匯出為 JSON 格式檔案，方便儲存和分享，且可匯入先前錄製的檔案以進行回播或練習。exportRecording 將錄製數據轉為 JSON 並生成下載連結；importRecording 解析上傳的 JSON 檔案，恢復錄製數據。

4、 **練習模式與學習報告：**

提供練習模式，顯示即將播放的音符並以視覺化方式（Canvas 繪圖）呈現音符下落，幫助使用者練習演奏。學習報告功能統計正確音符數量、總音符數量及正確率，協助使用者評估學習進度。

5、 **音頻處理與合成：**

利用 Web Audio API 和自定義的 Synth 類別實現音頻合成，定義了音符頻率（_notes）和波形生成函數（_mod），通過 generate 方法合成指定音色、音符和八度的音頻緩衝區。支援多種波形（正弦波、方波、鋸齒波、三角波）及 ADSR 包絡（Attack、Decay、Sustain、Release）調整音量，模擬樂器的音色特性。音頻信號經過卷積混響（Convolver）和濾波器（Biquad Filter）處理，增強音效真實感。通過預快取 preCacheAllSounds 方法預生成所有音色和音符的音頻緩衝區，減少即時生成音頻的延遲，提升播放流暢度。AudioContext 創建音頻緩衝區，支援立體聲（雙聲道）輸出

6、 **視覺化與響應式設計：**

鍵盤和按鈕的視覺效果（如按下時高亮）提升了互動體驗。使用 HTML5 Canvas 實現音符下落動畫和電壓波形圖，增強學習與除錯的直觀性。使用 CSS 實現響應式設計，適應行動裝置（縮放鍵盤和畫布）與桌面環境。

7、 **視覺反饋：**

.key.active 和 .ensemble-button.active 類別定義了按下時的高亮效果（橙色背景、移除陰影），提升使用者互動感。

伍、結論

「眾樂樂團聲音合成器」成功實現了多種樂器聲音的合成，並透過參數調整接近真實效果。本研究展示了 Web Audio API 在聲音合成中的潛力，並整理了各樂器的參數設定，為後續研究提供了參考基礎。

一、根據研究一：

透過 Web Audio API 的 AnalyserNode，成功實現聲音的時域與頻域分析，生成實時波形與頻譜圖，驗證聲音特徵的提取與比對功能。研究顯示，結合 FFT（**快速傅立葉變換**）可精準識別音高與音色，適用於音樂教育與聲音辨識應用。然而，高解析度 FFT（如 fftSize=2048）增加計算負擔，需優化演算法以提升低階設備的效能。未來可整合機器學習進行自動音符辨識，提升比對準確性。

二、根據研究二：

聲音合成器的探討利用 Web Audio API 的 OscillatorNode 和 GainNode，設計模組化聲音合成器，疊加多個 OscillatorNode 生成的不同頻率正弦波，可模擬樂器的泛音結構。使用多個振盪器組合不同頻率與幅度，逼近真實樂器的音色，但複雜度高還需要精確的頻譜分析數據來設定泛音比例，及需大量振盪器來模擬複雜音色（如鋼琴）。透過 BiquadFilterNode 濾波（如低通濾波）模擬樂器的音色衰減。**研究發現，適當調整濾波器的截止頻率與共振參數可模擬管樂（如薩克斯風）的溫暖音色。**減法合成簡單高效，適合模擬單音色樂器，但對複雜音色（如弦樂合奏）的動態變化模擬不足。未來加入動態濾波器包絡（如 ADSR）以提升音色逼真度。使用 FM 合成模擬電鋼琴與鈴鐺聲，**發現調變指數的精細調整對音色真實性至關重要**。FM 合成能以較少計算資源生成豐富音色，特別適合金屬質感樂器（如銅管）。然而，參數調校複雜，需參考真實樂器頻譜數據。未來可整合音色資料庫，簡化參數設定流程。

真實樂器的音色包含複雜的泛音、包絡（ADSR）與動態變化，單一合成技術難以全面還原，需結合多種方法。合成高品質樂器真實音色需要疊加多種複雜合成方法增加瀏覽器負擔，導致延遲或效能瓶頸，通過預快取方法預生成所有音色和音符的音頻緩衝區，減少即時生成音頻的延遲，提升播放流暢度。

三、根據研究三：

程式設計成功整合 Web Audio API、Web Serial API 與 Canvas 2D API，實現互動式虛擬音樂合成器，結合 Arduino 自製樂器，提供音樂創作與學習平台。系統整合硬體透過 Arduino 的串聯電阻分壓電路，精確讀取琴鍵輸入，並利用 Web Serial API 實現即時通訊，成功映射

至虛擬鍵盤 。

1、**音頻合成：**Web Audio API 的 Synth 類別支援多種波形（正弦波、方波等）與 ADSR 包絡，結合卷積混響與濾波器，模擬真實樂器音色。預快取音頻緩衝區有效降低播放延遲，提升流暢度。

2、**視覺化與互動：**Canvas 2D API 實現音符下落動畫與電壓波形圖，增強學習與除錯直觀性。

3、**錄製與學習：**錄製功能支援音符序列（含音高、音色、時序）儲存與回播，JSON 匯出/匯入便於分享。練習模式與學習報告（正確率統計）有效輔助音樂學習。

四、技術挑戰與未來改進解決方案：

訊號穩定性：電阻值誤差與雜訊影響琴鍵辨識，透過設定明確電壓閾值與濾波演算法改善準確性。感測器電壓範圍可能因環境變化漂移，需校正或使用更高精度感測器。

1、**硬體升級：**採用 16 位元外部 ADC（如 ADS1115）提升琴鍵數量與精確度。

2、**多鍵支持：**目前逐一判斷鍵值，無法處理同時敲擊多鍵，未來研習測試多點觸控支援以實現和弦演奏。

3、**效能瓶頸：**即時音頻合成需多振盪器同時運行 與 Canvas 渲染對低階設備造成負擔，透過預快取音頻與動態調整畫布尺寸緩解。

4、**相容性：**Web Serial API 僅限 Chrome/Edge 支援，需使用者授權，限制部分應用場景。

5、**聲音真實性：**目前的音色選擇雖然包含多種樂器，但音色的真實度和多樣性仍有限，當前合成器使用簡單波形與濾波器，無法完全模擬複雜樂器的泛音結構。**未來可引入更多的音色參數調整功能（如諧波比例、濾波器類型）加入 FM 合成或物理建模技術。**

6、**通訊優化：**改用 WebSocket 或無線模組（如 ESP32 Wi-Fi）取代序列埠，提升跨平台相容性與穩定性。

7、**音色逼真度：**整合 AI 分析真實樂器頻譜，自動優化合成參數，進一步模擬複雜音色。

8、**參數精細化：**可增加更多控制參數（如泛音強度、調製深度）以提升聲音豐富度。

9、**功能擴展：**新增 MIDI 支援與雲端音色庫，實現更豐富的音樂創作與多人協作。

10、**改進方向：**目前錄製檔案僅儲存於本地，無法實現雲端同步或多人協作。**整合後端 API（如 Firebase）以支援錄製檔案的雲端儲存和分享，實現多人合奏或線上學習功能。**

本程式設計展示了硬體與網頁技術的整合潛力，提供直觀的音樂創作與學習體驗，未來可透過技術優化與功能擴展，應用於音樂教育、虛擬樂團與互動藝術領域。

陸、參考資料及其他

[1]民國 112 年國小科展-STOMP-自製電音 BAND https://twsf.ntsec.gov.tw/activity/race-1/63/pdf/NPHSF2023-082821.pdf

[2]民國 72 年高中科展-新式電腦音樂合成器 https://twsf.ntsec.gov.tw/activity/race-1/23/pdf/23h/196.pdf

[3]民國 106 年高中科展-「MIDI 樂器（電子琴）」https://www.ntsec.edu.tw/Science-Content.aspx?a=6821&fld=&key=&isd=1&icop=10&p=1&sid=13767

[4]天花板隨記- Arduino 筆記(85)：電壓感測器(Voltage Sensor)https://atceiling.blogspot.com/2020/10/arduino85voltage-sensor.html

[5]groc 3 x.ai 人工智慧 https://grok.com/chat/

[6] Web Audio API - Web APIs | MDN

[7] Simple synth keyboard - Web APIs | MDN

[8] Note Frequency Chart | muted.io

[9] WebSynths : Browser-based musical instruments

[10]「聲音 audio 」 https://web.ntnu.edu.tw/~algo/Audio.html

[11] 傅立葉變換 what is the Fourier Transform https://youtu.be/spUNpyF58BY?si=2EVEyDhuxZuU2Swo

[12]音樂鍵盤 - JS 動態音訊合成器 由 Keith William Horwood 創作 2013 https://keithwhor.com/music/

圖 1-3 取自 Digidey 公司網頁資源 https://www.digikey.tw/zh/resources/conversion-calculators/conversion-calculator-voltage-divider)

圖 4-1 圖為 AI 輔助程式設計的合成器程式 /a2025/2025/0603x1/的擷取畫面 參考開源程式 https://keithwhor.com/music/

圖 4-2 為 AI 輔助程式設計的合成器程式 /a2025/2025/0501/0308a/a1.html 的擷取畫面 參考開源程式 https://keithwhor.com/music/

【評語】032806

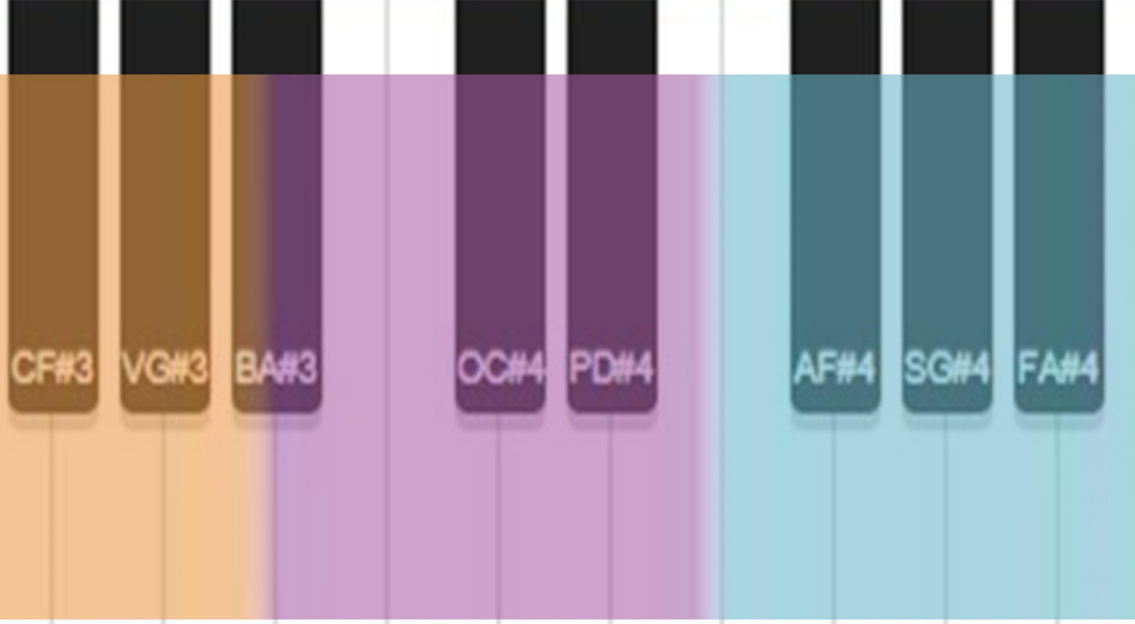
1. 跨領域整合能力強，結合電子電路、聲音科學、網頁程式與藝術創作，整體系統由硬體、軟體、通訊協定到人機互動，具實用性與可擴充性。可作為音樂教學工具或樂器開發參考。

2. 此作品的系統實作和討論完整。但主要貢獻在於做出電子鍵盤的彈奏介面。目前同一時間只能允許單音彈奏和發聲，造成此作品的功能和實用性受到限制，建議未來能改善這個缺點。

3. 目前僅比對波形與頻譜圖，建議補充頻譜相似度分析或使用者問卷評估，佐證音色模擬與使用體驗。

作品海報

眾樂樂 合成 樂團


CF#3
VG#3
BA#3
OC#4
PD#4
AF#4
SG#4
FA#4
RF3
TG3
YA3
UB3
NC4
MD4
,E4
.F4
[G4
]A4
IB4

壹、研究動機

在學習各種樂器的過程中，許多練習用的樂器價格不菲，特別是像馬林巴琴這樣昂貴的樂器。此外，在家中練習時，樂器的音量可能會打擾鄰居，限制了隨時隨地盡情演奏的可能性。傳統的音樂創作主要依賴現有樂器或數位音樂軟體，但這些方式受限於樂器種類與預設音色，限制了創作的多樣性。受到網路上許多樂團利用日常物品創作音樂的啟發，我們希望延續之前的「STOMP-自製電音BAND」專案[1]，進一步探討聲音合成理論，開發基於Arduino的自製琴鍵與網頁聲音合成器，實現多種樂器音色的即時合成與多人合奏功能。

在研究初期，我們運用人工智慧檢索相關研究，發現過去的研究中已有類似的作品。例如，民國72年的「新式電腦音樂合成器」[2]與民國106年的「MIDI樂器（電子琴）」[3]等。本研究期待結合上述科展研究的優點，進一步延伸。計畫使用Arduino作為接收控制、輸入音樂訊號，並由電腦控制聲音合成程式並發送聲音。此外鍵盤結構設計，改用自製鍵盤的方式，將以模組化可抽換方式，以符合我們可以演奏更多不同樂器的目標。通過這些改進，我們希望開發出一種可自由調整音色的電子樂器，讓使用者能夠隨時創作並演奏自己喜愛的音樂，並解決傳統樂器價格高昂和練習時噪音干擾的問題。

貳、研究目的及問題

一、聲音合成技術相關

如何透過不同的波形振盪器（例如正弦波、方波、鋸齒波）與濾波器組合，模擬馬林巴琴、鋼琴與鼓等樂器的音色？如何調整ADSR（Attack, Decay, Sustain, Release）參數，使合成的樂器音色更接近真實樂器？

二、硬體設計相關

如何設計一套模組化的Arduino琴鍵電路裝置，使其能夠方便更換並對應不同的樂器音色？

三、應用功能相關

如何實現一套基於Web Audio API的聲音合成器系統，讓多人同時進行即興演奏並輸出合奏效果？

如何解決電腦端即時運算合成音色的耗時問題，提升多人演奏時的即時反應速度？

參、研究設備及器材

一、研究設備：

windows 系統筆電一台、Arduino UNO一片、三用電表一台

二、研究器材：

cat6網路雙絞線、電焊工具組、 PCB電路板、可變電阻、精密電阻、瓦楞紙版、塑膠瓦楞版、鋁箔紙、銅箔貼紙、絕緣膠帶、透明膠帶、雙面膠帶、熱熔膠/熱熔膠槍

肆、研究過程或方法

圖3-1 研究架構圖

(二)研究一：聲音分析與比對

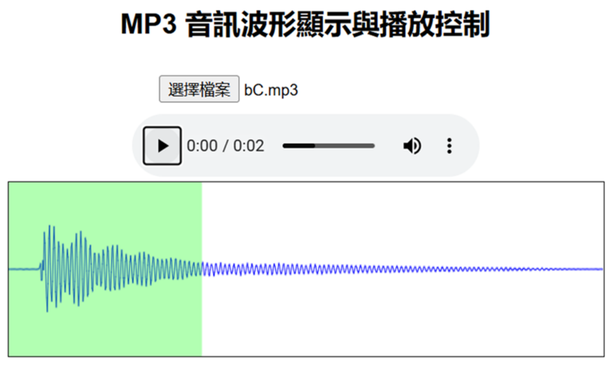


圖3-2 讀取聲音 顯示聲波形圖

	馬林巴琴	*鋼琴*	*大鼓*
Audacity 軟體進行初步聲音處理	以基頻與少量泛音為主，衰減較快	豐富的諧波，具較長的釋放階段	呈現低頻主導的特徵，伴隨短暫的噪聲成分
Web Audio API 開發的聲音合成器生成對應音色	正弦波（sine wave）作為主振盪器，低通濾波器（截止頻率 1500 Hz）過濾高頻	正弦波與三角波，低通濾波器（2500 Hz）	混合低頻正弦波（60 Hz）與白噪聲，低通濾波器（150 Hz）
ADSR	A: 0.005s, D: 0.5s, S: 0.05, R: 0.4s	A: 0.01s, D: 0.6s, S: 0.5, R: 0.8s	A: 0.01s, D: 0.25s, S: 0, R: 0.2s

表 2-1　不同樂器合成頻譜表

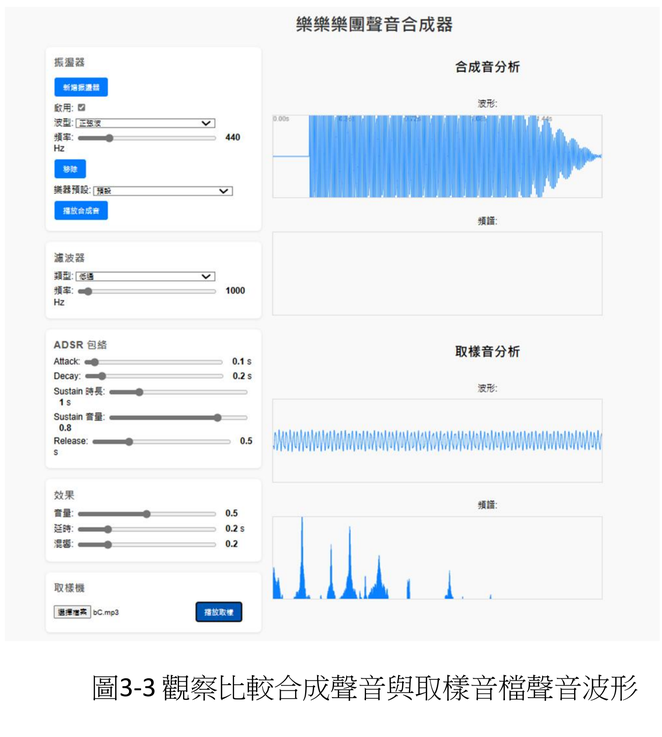


圖3-3 觀察比較合成聲音與取樣音檔聲音波形

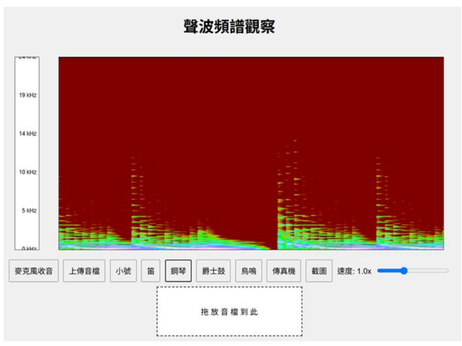


圖3-4 觀察各種聲音頻譜分佈圖 (/0428/x006.html)

本研究透過錄製真實樂器聲音並與合成音進行波形與頻譜比對，分析波形振盪器、濾波器及ADSR（Attack, Decay, Sustain, Release）包絡參數對音色的影響，進而找出模擬馬林巴琴、鋼琴與鼓等樂器的最佳參數。本研究確立了各樂器的初步合成模型，並記錄其波形特性與控制參數，為後續硬體與軟體整合提供了數據基礎。

(三)研究二：合成器的探討

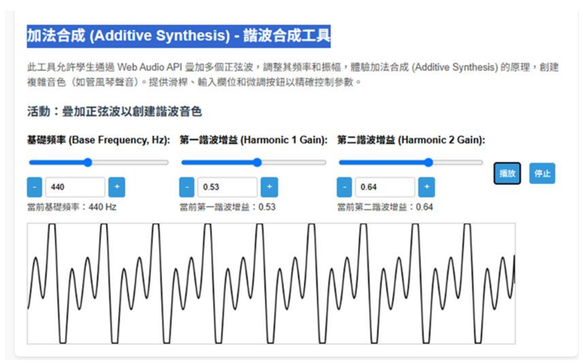


圖3-5 諧波合成圖工具圖

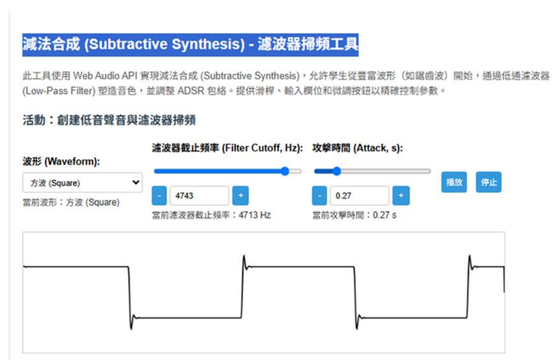


圖3-6 濾波器掃頻工具圖

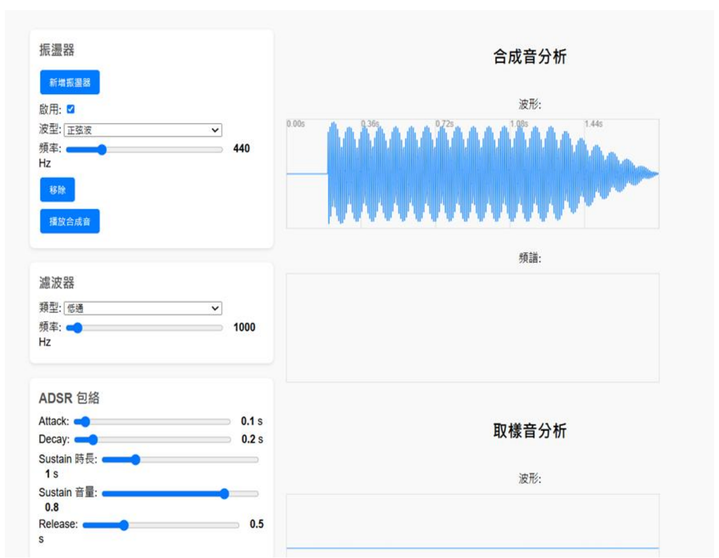


圖3-7 Web Audio API 設計聲音合成器

合成器由以下模組組成：

●**振盪器（Oscillator）**：生成基礎波形（正弦波、方波、鋸齒波、三角波、白噪聲、粉紅噪聲）。

●**濾波器（Filter）**：調整頻率成分（低通、高通、帶通）。

●**ADSR 包絡（Envelope）**：控制音量隨時間變化的特性（Attack、Decay、Sustain、Release）。

●**效果處理**：包括音量控制、延時（Delay）和混響（Reverb）。

●**取樣機**：支援上傳外部音檔並播放。

●**分析**： AnalyserNode 獲取音頻的時域數據 取出聲音波形數值轉換為頻譜數值。

聲音合成技術是電子音樂與互動媒體的重要組成部分。傳統的硬體合成器（如 Moog 或 Yamaha DX7）雖然功能強大，但成本高且不易整合到現代應用中。隨著 Web Audio API 的發展，網頁端的聲音合成成為可能，且具有跨平台、低成本的優勢。本研究利用 HTML、JavaScript 和 Web Audio API，設計了一款網頁合成器，並針對不同樂器聲音的特徵，優化其合成參數，以接近真實樂器的聽覺效果。這些模組透過 Web Audio API 的音頻節點（AudioNode）串聯，形成完整的聲音處理鏈。

1、程式核心功能

程式核心功能	功能	實現
振盪器模組	允許用戶新增多個振盪器，每個振盪器可獨立設定波形類型與頻率	使audioContext.createOscillator() 生成標準波形。透過 createNoise() 函數生成白噪聲與粉噪聲。每個振盪器可啟用/停用，並動態添加或移除。
濾波器	透過雙二階濾波器（Biquad Filter）過濾頻率成分	audioContext.createBiquadFilter() 支援低通、高通、帶通濾波。頻率範圍為 20-20000 Hz，用戶可透過滑桿調整。
ADSR 包絡	模擬樂器聲音的起音、衰減、持續與釋放階段	使用gainNode.gain的linearRampToValueAtTime方法，按時間線性變化音量。參數包括： ■Attack：起音時間（0-2秒）。 ■Decay：衰減時間（0-2秒）。 ■Sustain Time：持續時（0-5秒）。 ■Sustain Level：持續音量（0-1秒）。 ■Release：釋放時間（0-2秒）。
效果處理	**音量**：透過 gainNode.gain 控制整體音量。 **延時**：使用 audioContext.createDelay()，延時時間範圍為 0-1 秒。 **混響**：透過 audioContext.createConvolver() 生成簡單的脈衝響應，模擬空間感。	
視覺化	**波形**：使用OfflineAudioContext渲染合成音波形，並以Canvas繪製。 **頻譜**：透過 AnalyserNode 實時分析頻率分佈，顯示頻譜圖。	

表3-1 程式核心功能分析表

2、聲音合成流程

(1)用戶設定振盪器、濾波器、ADSR 和效果參數。
(2)點擊「播放合成音」按鈕，觸發 playSynth() 函數。
(3)合成器生成聲音，並同時渲染波形與頻譜。
(4)聲音透過濾波器、延時與混響處理後輸出至揚聲器。

3、如何合成真實樂器聲音

要合成接近真實樂器的聲音，需根據樂器的物理特性與聲學特徵調整參數：

波形選擇	**弦樂**（*如小提琴、吉他*）常用鋸齒波，模擬弓弦或撥弦的豐富諧波。 **打擊樂**（*如馬林巴琴、鋼琴*）偏向正弦波，強調基頻與簡單泛音。 **鼓聲**（*如大鼓、小鼓*）結合低頻正弦波與噪聲，模擬鼓皮振動與噪聲成分。 **管樂**（*如長笛*）使用正弦波，突出純淨的音色。
ADSR 包絡	**短促打擊樂**（*如馬林巴琴、大鼓*）具有快速 Attack 和短 Sustain。 **持續性樂器**（*如小提琴、長笛*）需要較長的 Sustain 與平滑的 Release。
濾波器	**低通濾波器**用於柔化尖銳的高頻（*如鋼琴、吉他*）。 **高通濾波器**用於去除低頻噪聲（*如長笛*）。 **帶通濾波器**適用於強調特定頻段（*如小鼓*）。
效果	**響模擬演奏環境的空間感**（*如鋼琴、小提琴*）。 **延時增加聲音的層次感**（*如吉他*）。

4、樂器參數設定表格

以下為程式中預設的樂器參數，根據聲學特徵優化：

	振盪器設置	濾波器	ADSR 參數	音量	延時 (s)	混響
馬林巴琴	1. 正弦波 (261.63 Hz) 2. 正弦波 (523.25 Hz) 3. 正弦波 (784.00 Hz)	低通 (1500 Hz)	A: 0.005, D: 0.5, ST: 0.1, SL: 0.05, R: 0.4	0.7	0.05	0.3
吉他	1. 鋸齒波 (110 Hz) 2. 鋸齒波 (220 Hz) 3. 方波 (330 Hz)	低通 (1000 Hz)	A: 0.02, D: 0.4, ST: 1.5, SL: 0.3, R: 0.6	0.5	0.15	0.2
大鼓	1. 正弦波 (60 Hz) 2. 白噪聲 (100 Hz)	低通 (150 Hz)	A: 0.01, D: 0.25, ST: 0, SL: 0, R: 0.2	0.8	0	0.1
小鼓	1. 三角波 (180 Hz) 2. 白噪聲 (500 Hz) 3. 粉噪聲 (1000 Hz)	帶通 (2000 Hz)	A: 0.01, D: 0.2, ST: 0, SL: 0, R: 0.15	0.7	0	0.15
鋼琴	1. 正弦波 (261.63 Hz) 2. 正弦波 (523.25 Hz) 3. 三角波 (784.00 Hz)	低通 (2500 Hz)	A: 0.01, D: 0.6, ST: 1.5, SL: 0.5, R: 0.8	0.6	0.1	0.25
長笛	1. 正弦波 (523.25 Hz) 2. 正弦波 (1046.50 Hz) 3. 正弦波 (1568.00 Hz)	高通 (700 Hz)	A: 0.15, D: 0.1, ST: 2, SL: 0.7, R: 0.3	0.5	0.05	0.2
小提琴	1. 鋸齒波 (261.63 Hz 2. 鋸齒波 (523.25 Hz) 3. 鋸齒波 (784.00 Hz)	低通 (3500 Hz)	A: 0.25, D: 0.15, ST: 2, SL: 0.6, R: 0.5	0.5	0.2	0.3

表 3-1 不同樂器程式中預設的樂器參數

(四)Arduino程式設計

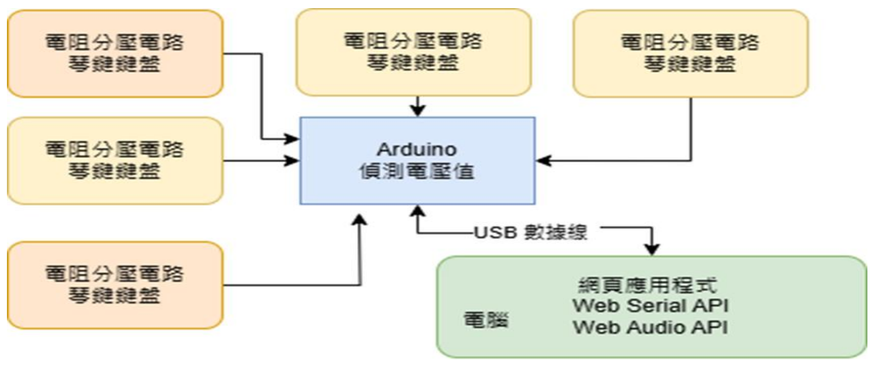
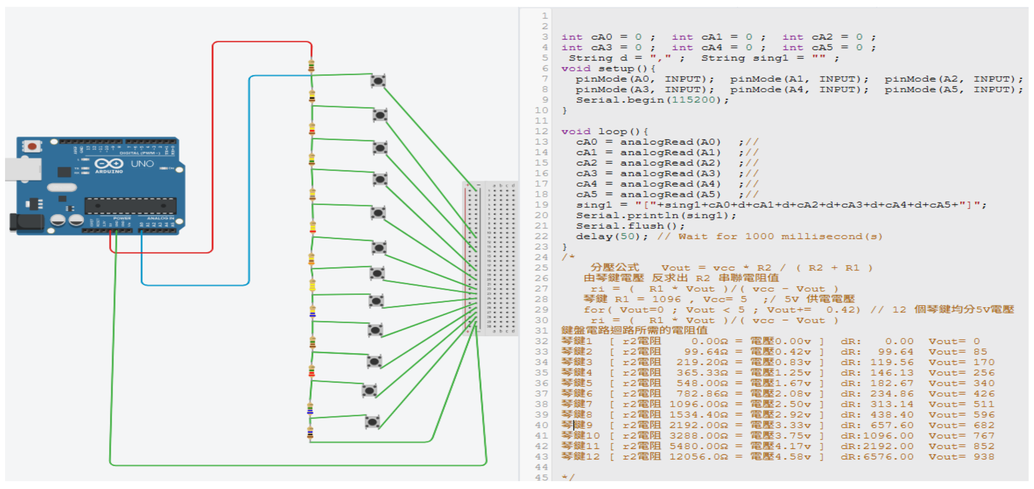


圖3-8 程式設計流程圖



```
int cA0 = 0 ;  int cA1 = 0 ;  int cA2 = 0 ;
int cA3 = 0 ;  int cA4 = 0 ;  int cA5 = 0 ;
 String d = "," ;  String sing1 = "" ;
void setup(){
  pinMode(A0, INPUT);  pinMode(A1, INPUT);  pinMode(A2, INPUT);
  pinMode(A3, INPUT);  pinMode(A4, INPUT);  pinMode(A5, INPUT);
  Serial.begin(115200);
}

void loop(){
  cA0 = analogRead(A0)  ;//
  cA1 = analogRead(A1)  ;//
  cA2 = analogRead(A2)  ;//
  cA3 = analogRead(A3)  ;//
  cA4 = analogRead(A4)  ;//
  cA5 = analogRead(A5)  ;//
  sing1 = "["+sing1+cA0+d+cA1+d+cA2+d+cA3+d+cA4+d+cA5+"]";
  Serial.println(sing1);
  Serial.flush();
  delay(50); // Wait for 1000 millisecond(s)
}
/*
    分壓公式   Vout = vcc * R2 / ( R2 + R1 )
   由琴鍵電壓 反求出 R2 串聯電阻值
    ri = ( R1 * Vout )/( vcc - Vout )
   琴鍵 R1 = 1096 , Vcc= 5 ;/ 5V 供電電壓
   for( Vout=0 ; Vout < 5 ; Vout+=  0.42) // 12 個琴鍵均分5v電壓
    ri = ( R1 * Vout )/( vcc - Vout )
鍵盤電路迴路所需的電阻值
琴鍵1  [ r2電阻    0.00Ω = 電壓0.00v ]  dR:   0.00  Vout= 0
琴鍵2  [ r2電阻   99.64Ω = 電壓0.42v ]  dR:  99.64  Vout= 85
琴鍵3  [ r2電阻  219.20Ω = 電壓0.83v ]  dR: 119.56  Vout= 170
琴鍵4  [ r2電阻  365.33Ω = 電壓1.25v ]  dR: 146.13  Vout= 256
琴鍵5  [ r2電阻  548.00Ω = 電壓1.67v ]  dR: 182.67  Vout= 340
琴鍵6  [ r2電阻  782.86Ω = 電壓2.08v ]  dR: 234.86  Vout= 426
琴鍵7  [ r2電阻 1096.00Ω = 電壓2.50v ]  dR: 313.14  Vout= 511
琴鍵8  [ r2電阻 1534.40Ω = 電壓2.92v ]  dR: 438.40  Vout= 596
琴鍵9  [ r2電阻 2192.00Ω = 電壓3.33v ]  dR: 657.60  Vout= 682
琴鍵10 [ r2電阻 3288.00Ω = 電壓3.75v ]  dR:1096.00  Vout= 767
琴鍵11 [ r2電阻 5480.00Ω = 電壓4.17v ]  dR:2192.00  Vout= 852
琴鍵12 [ r2電阻 12056.0Ω = 電壓4.58v ]  dR:6576.00  Vout= 938
*/
```

圖 3-9 偵測琴鍵訊號電路圖

透過Arduino的A0至A5類比輸入腳位，偵測琴鍵電路的電壓訊號，作為演奏輸入。將偵測到的電壓值轉換為JSON字串格式，透過串列通訊（Serial Communication）傳送至筆電。

琴鍵電路的電壓分壓原理基於以下公式： **Vout = vcc * R2 / (R2 + R1)**
均分12個琴鍵電路的電壓值，反推出 R2 串聯電阻值 ri = (R1 * Vout)/(vcc - Vout)
範例計算 已知條件：**(R1=1096Ω，Vcc=5V供電電壓）則12個琴鍵電路所對應的(R2)電阻值**

琴鍵	R2電阻 (Ω)	電壓 (V)	類比值	電阻差 (Ω)
1	0	0	0	0
2	99.64	0.42	85	99.64
3	219.2	0.83	170	119.56
4	365.33	1.25	256	146.13
5	548	1.67	340	182.67
6	782.86	2.08	426	234.86
7	1096	2.5	511	313.14
8	1534.4	2.92	596	438.4
9	2192	3.33	682	657.6
10	3288	3.75	767	1096
11	5480	4.17	852	2192
12	12056	4.58	938	6576

(五)網頁應用程式端的程式設計

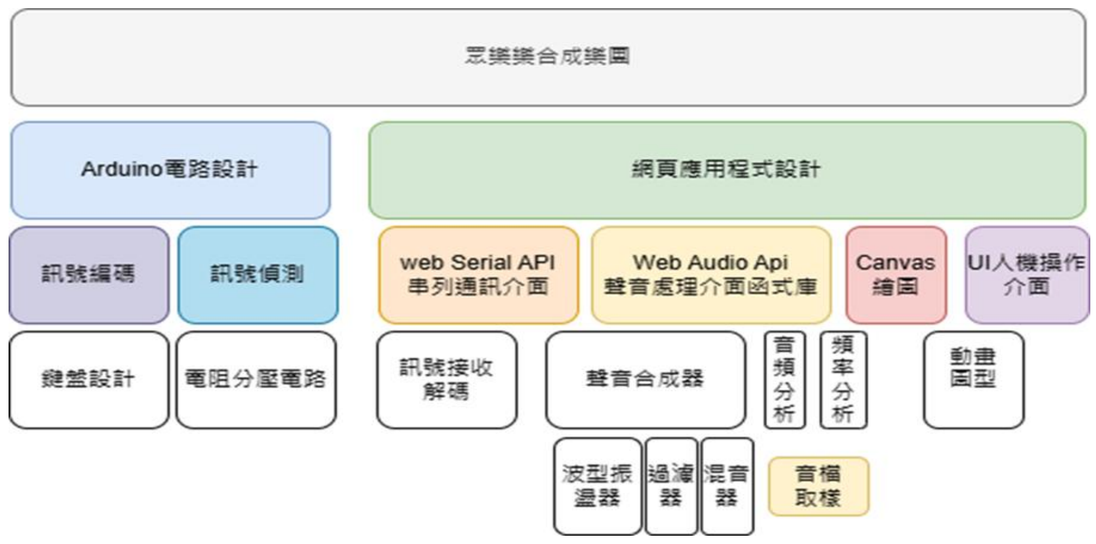
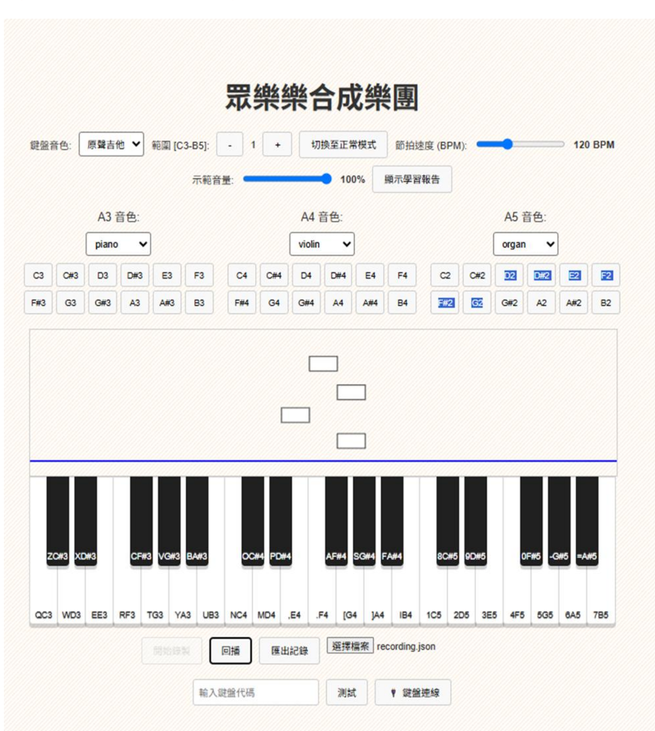


圖3-10 網頁應用程式端的程式設計圖

程式實現細節	
振盪器生成	(1)程式支援動態添加振盪器，每個振盪器透過addOscillator()函數創建，並可選擇波形與頻率。 (2)噪聲生成使用自定義算法（白噪聲為隨機值，粉噪聲為濾波後的噪聲）。
ADSR 實現	(1) applyADSR() 函數根據用戶設定的參數，動態調整 gainNode 的音量曲線。 (2)使用線性斜坡（linearRampToValueAtTime）實現平滑過渡。
效果處理	(1)混響透過 createReverb() 函數生成隨機衰減的脈衝響應，模擬自然殘響。 (2)延時透過 DelayNode 實現，參數可調。
視覺化	(1)波形使用 OfflineAudioContext 預渲染，確保精確性。 (2)頻譜透過 AnalyserNode 實時計算並繪製。
參數儲存與匯出	(1) saveParamsButton 將當前設定儲存為 JSON 檔案。 (2)exportAudioButton 目前僅播放聲音，需引入第三方庫（如 lamejs）實現 MP3 匯出。

肆、研究結果及討論



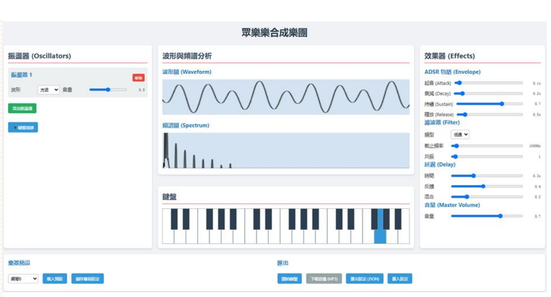


圖4-2 聲音合成器使用者介面

圖4-1. 演奏播放 使用者操作介面外觀

本研究參考引用開源程式,開發了網頁應用程式，該系統透過 Arduino 的序列介接（Serial Interface）讀取演奏訊號，並根據不同樂器設定播放事先合成好的音檔。研究重點在於實現 Arduino 與網頁之間的即時通訊，利用 Web Serial API 接收演奏訊號，並透過 Web Audio API 播放預載音檔。本報告詳細說明程式設計原理，特別針對序列通訊模組（application.js）進行分析，並闡述如何根據演奏訊號觸發不同樂器的音檔播放。

一、系統設計與功能說明

(一)整體架構

系統由以下主要部分組成		
琴鍵樂器硬體設計	利用串聯電阻分壓電路實現琴鍵輸入，每個琴鍵開關對應一個設定的電阻值，連接到 Arduino 的類比輸入腳位。依按下琴鍵所改變電阻值，產生不同電壓，經 Arduino 的 10 位元 ADC（0-1023）轉換為數位值，解析度約 4.9mV（5V/1024）。	
資料處理	Arduino 讀取類比輸入腳位的電壓類比值，根據電壓範圍映射到特定琴鍵（音符），透過analogRead()獲取數值，程式內判斷電壓範圍對應的琴鍵。	
JSON 生成與傳輸	Arduino 將琴鍵資料封裝成 JSON 格式，透過序列埠（Serial）以 115200波特率傳送到電腦，網頁應用程式使用 Web Serial API 接收 JSON 訊號並解碼。	
網頁應用程式	使用 JavaScript 解析 JSON，根據音符資料驅動 Web Audio API 合成音檔,並存入音頻緩衝區 播放對應音頻，並以Canvas 2D API顯示互動動畫。	
網頁應用程式	**虛擬鍵盤介面**	提供手動觸發音檔的鍵盤與按鈕
	序列通訊模組	使用 Web Serial API 接收 Arduino 數據
	音檔播放模組	使用 Web Audio API 播放預載音檔（假設在 audioPreload.js 中）
	視覺化模組	以 Canvas 繪製類比訊號長條圖

伍、結論

「眾樂樂團聲音合成器」成功實現了多種樂器聲音的合成，並透過參數調整接近真實效果。本研究展示了 Web Audio API 在聲音合成中的潛力，並整理了各樂器的參數設定，為後續研究提供了參考基礎。

根據研究一	透過 Web Audio API 的 AnalyserNode，成功實現聲音的時域與頻域分析，生成實時波形與頻譜圖，驗證聲音特徵的提取與比對功能。研究顯示，結合 FFT（快速傅立葉變換）可精準識別音高與音色，適用於音樂教育與聲音辨識應用。然而，高解析度 FFT（如 fftSize=2048）增加計算負擔，需優化演算法以提升低階設備的效能。未來可整合機器學習進行自動音符辨識，提升比對準確性。
根據研究二	聲音合成器的探討利用 Web Audio API 的 OscillatorNode 和 GainNode，設計模組化聲音合成器，疊加多個 OscillatorNode 生成的不同頻率正弦波可模擬樂器的泛音結構。使用多個振盪器組合不同頻率與幅度，逼近真實樂器的音色，但複雜度高還需要精確的頻譜分析數據來設定泛音比例，及需大量振盪器來模擬複雜音色（如鋼琴）。透過 BiquadFilterNode 濾波（如低通濾波）模擬樂器的音色衰減。研究發現，適當調整濾波器的截止頻率與共振參數可模擬管樂（如薩克斯風）的溫暖音色。減法合成簡單高效，適合模擬單音色樂器，但對複雜音色（如弦樂合奏）的動態變化模擬不足。未來加入動態濾波器包絡（如 ADSR）以提升音色逼真度。使用 FM 合成模擬電鋼琴與鈴鐺聲，**發現調變指數的精細調整對音色真實性至關重要**。FM 合成能以較少計算資源生成豐富音色，特別適合金屬質感樂器（如銅管）。然而，參數調校複雜，需參考真實樂器頻譜數據。未來可整合音色資料庫，簡化參數設定流程。 **真實樂器的音色包含複雜的泛音、包絡（ADSR）與動態變化，單一合成技術難以全面還原，需結合多種方法。**合成高品質樂器真實音色需要疊加多種複雜合成方法增加瀏覽器負擔，導致延遲或效能瓶頸，通過預快取方法預生成所有音色和音符的音頻緩衝區，減少即時生成音頻的延遲，提升播放流暢度。
根據研究三	**音頻合成**：Web Audio API 的 Synth 類別支援多種波形（正弦波、方波等）與 ADSR 包絡，結合卷積混響與濾波器，模擬真實樂器音色。預快取音頻緩衝區有效降低播放延遲，提升流暢度。 **視覺化與互動**：Canvas 2D API 實現音符下落動畫與電壓波形圖，增強學習與除錯直觀性。 **錄製與學習**：錄製功能支援音符序列（含音高、音色、時序）儲存與回播，JSON 匯出/匯入便於分享。練習模式與學習報告（正確率統計）有效輔助音樂學習。

技術挑戰與未來改進解決方案

硬體升級	採用 16 位元外部 ADC（如 ADS1115）提升琴鍵數量與精確度
多鍵支持	目前逐一判斷鍵值，無法處理同時敲擊多鍵，未來研習測試多點觸控支援以實現和弦演奏
效能瓶頸	即時音頻合成需多振盪器同時運行 與 Canvas 渲染對低階設備造成負擔，透過預快取音頻與動態調整畫布尺寸緩解
相容性	Web Serial API 僅限 Chrome/Edge 支援，需使用者授權，限制部分應用場景
聲音真實性	目前的音色選擇雖然包含多種樂器，但音色的真實度和多樣性仍有限，當前合成器使用簡單波形與濾波器，無法完全模擬複雜樂器的泛音結構。未來可引入更多的音色參數調整功能（如諧波比例、濾波器類型）加入 FM 合成或物理建模技術
通訊優化	改用 WebSocket 或無線模組（如 ESP32 Wi-Fi）取代序列埠，提升跨平台相容性與穩定性
音色逼真度	整合AI分析真實樂器頻譜，自動優化合成參數，進一步模擬複雜音色
參數精細化	可增加更多控制參數（如泛音強度、調製深度）以提升聲音豐富度
功能擴展	新增 MIDI 支援與雲端音色庫，實現更豐富的音樂創作與多人協作
改進方向	目前錄製檔案僅儲存於本地，無法實現雲端同步或多人協作。整合後端 API（如 Firebase）以支援錄製檔案的雲端儲存和分享，實現多人合奏或線上學習功能

本程式設計展示了硬體與網頁技術的整合潛力，提供直觀的音樂創作與學習體驗，未來可透過技術優化與功能擴展，應用於音樂教育、虛擬樂團與互動藝術領域。

陸、參考資料及其他

[1]民國112年國小科展-STOMP-自製電音BAND https://twsf.ntsec.gov.tw/activity/race-1/63/pdf/NPHSF2023-082821.pdf
[2]民國72年高中科展-新式電腦音樂合成器 https://twsf.ntsec.gov.tw/activity/race-1/23/pdf/23h/196.pdf
[3]民國106年高中科展-「MIDI樂器（電子琴）」 https://www.ntsec.edu.tw/Science-Content.aspx?a=6821&fld=&key=&isd=1&icop=10&p=1&sid=13767
[4]天花板隨記- Arduino筆記(85)：電壓感測器(Voltage Sensor)https://atceiling.blogspot.com/2020/10/arduino85voltage-sensor.html
[5]groc 3 x.ai 人工智慧 https://grok.com/chat/
[6] Web Audio API - Web APIs | MDN
[7] Simple synth keyboard - Web APIs | MDN
[8] Note Frequency Chart | muted.io
[9] WebSynths : Browser-based musical instruments
[10]「聲音 audio 」 https://web.ntnu.edu.tw/~algo/Audio.html
[11] 傅立葉變換 what is the Fourier Transform https://youtu.be/spUNpyF58BY?si=2EVEyDhuxZuU2Swo
[12]音樂鍵盤 - JS 動態音訊合成器 由Keith William Horwood創作 2013 https://keithwhor.com/music/
圖1-3 取自Digidey 公司網頁資源 https://www.digikey.tw/zh/resources/conversion-calculators/conversion-calculator-voltage-divider)
圖4-1 圖為 AI輔助程式設計的合成器程式 /a2025/2025/0603x1/的擷取畫面 參考開源程式https://keithwhor.com/music/
圖4-2 為 AI輔助程式設計的合成器程式 /a2025/2025/0501/0308a/a1.html 的擷取畫面 參考開源程式https://keithwhor.com/music/