雙聲部音樂撥放器

居家智慧監控實習

控制二乙 12號
林澄鈺

指導老師:
張漢民

目錄

摘要

此實作是開發出一款具有「按鈕控制」與「多聲部同步」功能的音樂播放器。

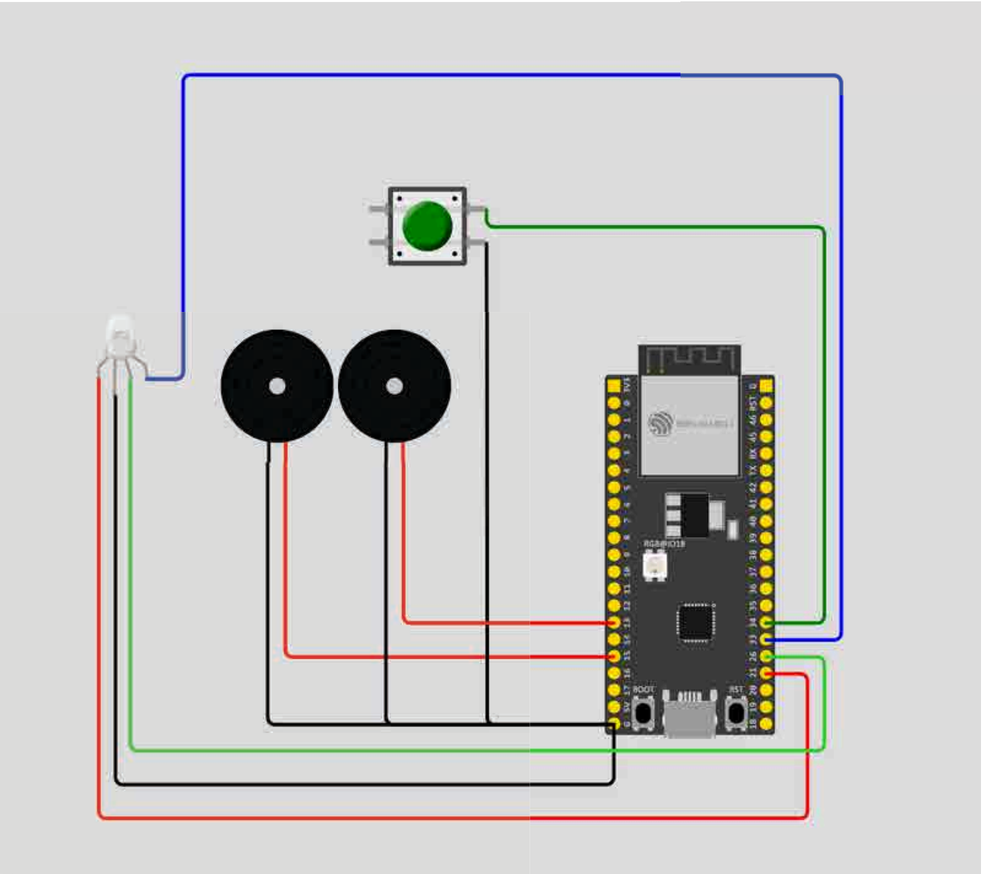
利用 Arduino 控制板結合 RGB LED 與雙蜂鳴器，透過按鈕長短按的狀態，執行播放、暫停、重置等多種狀態。運用 millis函數建立狀態機，解決傳統 delay指令造成的單一與不便，成功使主旋律與伴奏在單一開發板上的節拍同步並輸出。

動機

自高二上學期的程式實習課以來，我掌握到了一些程式、函數的用法，並接上硬體元件做實際應用。由於我喜歡音樂，在控制蜂鳴器的單元時，我不滿足只播放單音，要再下一個音、一段旋律，直到一首完整的歌。

本次的實作是整合過去所學的RGB LED顏色控制、millis狀態機結構、按鈕開關模式及控制蜂鳴器聲音，做出一個音樂撥放器，並播放出自己喜歡的音樂

主題與功能



主題:

　藉由按鈕狀態控制音樂播放器，實現音樂的開始、暫停與重播，並由RGB LED的顏色來判斷目前的工作狀態。

主題與功能

詳細功能:

1.待機: 開機為待機狀態，RGB LED亮綠燈

2.播放: 按下按鈕，音樂播放，同時RGB LED為紅燈閃爍

3.自動循環: 若音樂結束，會等待一定時間，並重新播放

4.暫停/繼續: 音樂播放中若單按按鈕，音樂停止，RGB LED亮藍燈，再次按下則繼續播放

5.重置: 音樂播放中若長按按鈕，音樂停止，RGB LED亮白燈。
當長按達預設時間並鬆手，返回待機狀態，RGB LED為綠光，下次按下按鈕將從頭播放

實作步驟

1.定義音符頻率

```
#define C2  65
#define Cs2 69   // C#2 / Db2
#define D2  73
#define Ds2 78   // D#2 / Eb2
#define E2  82
#define F2  87
#define Fs2 92   // F#2 / Gb2
#define G2  98
#define Gs2 104  // G#2 / Ab2
#define A2  110
#define As2 117  // A#2 / Bb2
#define B2  123

#define C3  131
#define Cs3 139  // C#3 / Db3
#define D3  147
#define Ds3 156  // D#3 / Eb3
#define E3  165
#define F3  175
#define Fs3 185  // F#3 / Gb3
#define G3  196
#define Gs3 208  // G#3 / Ab3
#define A3  220
#define As3 233  // A#3 / Bb3
#define B3  247

// 中音區（C4 ~ B4，包含標準 C4）
#define C4  262
```

2.建立陣列，把音符、節拍等透過陣列編碼，形成一首歌

```
{C4, 1}, {E4, 1}, {C4, 1}, {E4, 1}, {D4, 1}, {Fs4, 1}, {D4, 1}, {Fs4, 1}, {E4, 1},{G4, 1}, {E4, 1}, {G4, 1}, {B3, 1}, {D4, 1}, {B3, 1}, {D4, 1},
{C4, 1}, {E4, 1}, {C4, 1}, {E4, 1}, {D4, 1}, {Fs4, 1}, {D4, 1}, {Fs4, 1}, {E4, 1},{G4, 1}, {E4, 1}, {G4, 1}, {B3, 1}, {D4, 1}, {B3, 1}, {D4, 1},
{G5, 1}, {Fs5, 1}, {E5, 1},   {D5, 2.38},{B4, 0.31}, {C5, 0.31},{D5, 0.31}, {C5, 0.31},{B4, 0.31}, {A4, 0.31}, {G4, 1}, {Fs4, 1}, {G4, 1},  {B4, 2.38},{D4, 2},{REST, 0.01}, {D4, 1},
{E4, 1}, {Fs4, 1}, {G4, 1},{D5, 2.38},{B4,2.38},{A4, 2.38},{G4, 1}, {A4, 1},{B4, 3.76},{D4, 2.38},{REST, 0.4},
{B4, 1}, {B4, 1},{B4, 1},{A4, 2.38},{D5, 2.38}, {B4, 2.38},{G4, 1}, {REST, 1}, {G4, 1}, {A4, 1}, {G4, 1},{Fs4, 1}, {G4, 1},
{E4, 2.38},{B4, 1},{A4, 3.76},{A4, 2.38},{D5, 1}, {B4, 1}, {REST, 1}, {B4, 1}, {C5, 1},{B4, 1},  {A4, 2.38},
{B4, 1}, {B4, 1},{B4, 1},{A4, 2.38}, {D5, 2.38},{E5, 2.38},{B4, 1}, {REST, 1}, {G4, 1}, {A4, 1},{G4, 1},   {Fs4, 1}, {G4, 1},
{E4, 2.38},{B4, 1},{A4, 3.76},{A4, 2.38},{E5, 1},{D5, 1}, {REST, 1},{B4, 1}, {C5, 1},{B4, 1},{A4, 1},{G4, 1},
{B4, 3},{A4, 3},{B4, 2},{Fs5, 3},{D5, 3},{B4, 3}, {E5, 2},{REST, 1},{B4, 1},{B4, 2},{REST, 1},{G4, 1},{B4, 1},{B4, 1},{B4, 1},{A4, 2},{B4, 1},{G4, 2},
{B4, 3}, {A4, 3},{B4, 2},{Fs5, 3},{D5, 3},{B4, 2},{Ds5, 1.63},{D5, 1.63},{C5, 1.12},{C5, 1.63},{As4, 1.63},{C5, 1.12},{D5, 2},{REST, 1},
{D5, 1},{D5, 1},{E5, 1},{Fs5, 2},
{G5, 3.76},{Fs5, 1},  {Fs5, 3.76}, {E5, 1},{E5, 2.38},{REST, 1},{D5, 1}, {D5, 1},{D5, 1},{D5, 1},{G5, 2.38}, {Fs5, 2.38},{D5, 2.38},{B4, 1},{C5, 1}, {D5, 3.76}, {REST, 1},{D5, 1},{D5, 1},{E5, 1},{Fs5, 2.38},
{G5, 3.76},{Fs5, 1},  {Fs5, 3.76},{G5, 1}, {G5, 2.38},    {REST, 1}, {D5, 1},
{D5, 1},{D5, 1},{E5, 1},   {D5, 2.38}, {B5, 1},{C6, 1},{B5, 2.38},{G5, 1},{A5, 1},{G5, 3.76},{REST, 1},{A4, 1},{B4, 1},{D5, 1},{E5, 1},{D5, 1},
{C4, 1}, {E4, 1}, {C4, 1}, {E4, 1}, {D4, 1}, {Fs4, 1}, {D4, 1}, {Fs4, 1}, {E4, 1},{G4, 1}, {E4, 1}, {G4, 1}, {B3, 1}, {D4, 1}, {B3, 1}, {D4, 1},
{C4, 1}, {E4, 1}, {C4, 1}, {E4, 1}, {D4, 1}, {Fs4, 1}, {D4, 1}, {Fs4, 1}, {E4, 1},{G4, 1}, {E4, 1}, {G4, 1}, {B3, 1}, {D4, 1}, {B3, 1}, {D4, 1},
{B4, 1}, {B4, 1},{B4, 1}, {A4, 2.38}, {D5, 2.38}, {B4, 2.38}, {G4, 1}, {REST, 1}, {G4, 1}, {A4, 1}, {G4, 1},{Fs4, 1}, {G4, 1},
{E4, 2.38},{B4, 1}, {A4, 3.76},{A4, 2.38},{D5, 1}, {B4, 1}, {REST, 1}, {B4, 1}, {C5, 1},{B4, 1},{A4, 2.38},
{B5, 1}, {B5, 1},   {B5, 1}, {A5, 2.38},{D6, 2.38}, {E6, 2.38},  {B5, 1}, {REST, 1},{G5, 1},{A5, 1}, {G5, 1}, {Fs5, 1}, {G5, 1}, {E5, 2.38},  {B5, 1},{A5, 3.76},  {A5, 2.38}, {E6, 1},{D6, 1},{REST, 1},
{G5, 1}, {A5, 1}, {G5, 1},{Fs5, 1}, {G5, 1}
};
Note song2[] = {
  {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1},{REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1},
  {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1},{REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1},
  {C3, 1}, {REST, 1}, {C3, 1},  {REST, 1},{D5, 1},    {REST, 1},{D3, 1},{REST, 1},  {B3, 1},{REST, 1}, {E3, 1}, {REST, 1}, {B2, 1},  {REST, 1}, {B2, 1},{REST, 1.01},
  {C3, 1}, {REST, 1}, {C3, 1}, {REST, 1}, {D3, 1},    {REST, 1}, {D3, 1},   {REST, 1}, {E3, 1},    {REST, 1}, {E3, 1}, {REST, 1}, {B2, 1}, {D3, 1},   {D3, 1}, {+3, 1},{REST, 0.4},
  {E2, 1}, {REST, 1}, {C3, 1}, {Fs2, 1}, {REST, 1},  {D3, 1}, {D2, 1}, {REST, 1},{E3, 1},    {D2, 1}, {G2, 1}, {D2, 1}, {B1, 1}, {G2, 1}, {G2, 2.38},
  {E2, 1}, {REST, 1}, {C3, 1}, {Fs2, 1}, {REST, 1}, {D3, 1}, {G2, 1}, {REST, 1}, {E3, 1}, {G2, 1}, {E3, 1}, {B2, 1}, {G2, 1}, {B2, 1}, {Fs2, 2.38},
  {C3, 1}, {REST, 1}, {C2, 1}, {D3, 1}, {REST, 1}, {D2, 1}, {E3, 1},{REST, 1}, {E2, 1},{B2, 1},{B1, 1},  {B1, 1}, {G2, 1}, {D2, 1},   {B2, 2.38},
  {C3, 1}, {REST, 1}, {C2, 1}, {D3, 1}, {REST, 1},{D2, 1},  {E3, 1}, {REST, 1}, {E2, 1},  {A3, 1},{REST, 1}, {E2, 1}, {B3, 2.38},  {E2, 2.38},
  {D3, 6.38},{D3, 2}, {E3, 6.38},  {E3, 2}, {C3, 4.76},  {C3, 4.76},  {C3, 3.76}, {C3, 2},   {C3, 3.38},
  {D3, 6.38}, {D3, 2}, {E3, 6.38},  {E3, 2}, {C3, 1},{D3, 1},{D#3, 1},{C4, 1},  {G5, 1},{D#4, 1},{C4, 1},{G4, 1},   {D4, 1},{REST, 1},
  {REST, 1},{REST, 1},{REST, 1},{REST, 2},
  {C4, 1}, {E4, 1}, {C4, 1}, {E4, 1}, {D4, 1}, {Fs4, 1}, {D4, 1}, {Fs4, 1},  {E4, 1},{G4, 1},   {E4, 1}, {G4, 1},{B3, 1}, {D4, 1}, {B3, 1},  {D4, 1},{C4, 1}, {E4, 1}, {C4, 1}, {E4, 1}, {D4, 1}, {Fs4, 1}, {D4,
  {C4, 1}, {E4, 1}, {C4, 1}, {E4, 1}, {B3, 1}, {Fs4, 1}, {D#4, 1}, {Fs4, 1},  {E4, 1}, {G4, 1}, {Fs4, 1}, {A4, 1},
  {G4, 1}, {B4, 1}, {D3, 1}, {D4, 1},{C4, 1}, {E4, 1}, {C4, 1}, {E4, 1},{D4, 1}, {Fs4, 1}, {D4, 1}, {Fs4, 1},{E4, 1}, {Fs4, 1}, {G4, 1}, {REST, 1},{REST, 1}, {G4, 1},{G4, 2.38},
  {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1},{REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1},
  {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1},{REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1}, {REST, 1},
  {C4, 1}, {E4, 1}, {C4, 1}, {E4, 1}, {D4, 1},   {Fs4, 1}, {D4, 1}, {Fs4, 1}, {E4, 1},  {G4, 1}, {E4, 1}, {G4, 1}, {B3, 3}, {D4, 1}, {B3, 1}, {D4, 1},
  {C4, 1}, {E4, 1}, {C4, 1}, {E4, 1}, {D4, 1},{Fs4, 1}, {D4, 1}, {Fs4, 1}, {E4, 1},{G4, 1}, {E4, 1}, {G4, 1}, {B3, 1}, {D4, 1}, {B3, 1}, {D4, 1},
  {C5, 1}, {REST, 1}, {C5, 1},  {REST, 1},{D5, 1},{REST, 1},{D5, 1},{REST, 1},{E5, 1},{REST, 1},{E5, 1},{REST, 1},{B4, 1},{REST, 1},{B4, 1},{REST, 1}, {C5, 1},{REST, 1},{C5, 1},{REST, 1},{D5, 1},{REST, 1},{D5,
  {REST, 1},{REST, 1},{REST, 1},{REST, 1}
};
```

3.設定參數、副程式、接腳

```
int total = sizeof(song) / sizeof(Note);
int total2 = sizeof(song2) / sizeof(Note);
// ------------------------------------------------------------------------
int pin = 15;              // 蜂鳴器連接腳位 (GPIO 15)
int pin2 = 13;
float base_duration = 130;  // 音符基準時間 (單位: ms，決定節奏快慢)
float note_interval = 50;  // 音符間的短暫間隔 (單位: ms，避免聲音黏在一起)
const int channel1 = 0;
const int channel2 = 2;
// ------------------------------------------------------------------------
int i = 0;       // 當前播放的音符索引
int j = 0;       // song2 的索引
int state = 1;   // 狀態機控制旗標：
                 // 1：播放音符
                 // 2：音符間隔
                 // 3：切換至下個音符
                 // 4：播放結束
int state2 = 1;
// ------------------------------------------------------------------------
unsigned long previousMillis = 0;   // 上一次狀態開始的時間
unsigned long previousMillis2 = 0;
unsigned long previousMillisSong = 0;
int note_duration = 0;
int note_duration2 = 0;             // 當前音符應該持續的時間
// ------------------------------------------------------------------------
#include <OneButton.h>          // 引用OneButton函式庫
#define pb 34                   // 定義按鈕引腳
OneButton button(pb, true);     // 建立OneButton物件，名稱為button，按鈕為低態動作
boolean startPlay = 0;          // 0：停止播放，1：開始播放
//-------------------------------------------------------------------------
```

```
// 單擊處理
void singleClick() {
  Serial.println("Button Single Click.");
  startPlay = !startPlay;     // 開始播放或停止播放
    if(startPlay == 0)          // 停止播放時
  {
    ledcWriteTone(pin, 0);
    ledcWriteTone(pin2, 0);   // 靜音
    show_color(4);
  }
}

// 雙擊處理
void doubleClick() {
  Serial.println("Button Double Click.");
}

// 長按開始
void longPressStart() {
  Serial.println("Button Long Press Start.");
  ledcWriteTone(pin, 0);   // 靜音
  ledcWriteTone(pin2, 0);
  show_color(1);
}

// 長按過程中
void longPress() {
  Serial.println("Button Long Press ......");
  startPlay = 0;  // 停止播放
  show_color(1);
}

// 長按結束
void longPressStop() {
  Serial.println("Button Long Press Stop.");
  i = 0;
  j = 0;      // 索引值歸零(重新開始)
  state = 1;  // 進入狀態1(重新開始)
  state2 = 1;
  if(startPlay == 0)   // 永遠執行{.....}內的程式
  {
      show_color(3);
      delay(500);
  }
```

```
214  int rgbPin[3] = {21,26,33}; // R,G,B接腳(記得到時候是21.22.23)
215  int color[5][3] = {{0,0,0},{255,255,255},{255,0,0},{0,255,0},{0,0,255}};  // {Black, White, Red, Green, Blue}
216  void show_color(int x)
217  {
218    for(int i=0; i<=2; i++)
219    {
220      analogWrite(rgbPin[i],color[x][i]);
221    }
222  }
223  int ledtime=0;
224  //--------------------------------------------------------------------
225  void setup() {
226    //--------------------------------------------------------------------
227    // ESP32 LEDC 設定
228    ledcAttachChannel(pin, 2000, 10, channel1);  // 指定通道 0
229    ledcAttachChannel(pin2, 2001, 10, channel2); // 指定通道 1   // 設定腳位, 初始頻率 2000Hz, 解析度 10 bits
230    ledcWriteTone(pin, 0);        // 初始狀態為靜音     // 設定腳位, 初始頻率 2000Hz, 解析度 10 bits
231    ledcWriteTone(pin2, 0);        // 初始狀態為靜音
232    //--------------------------------------------------------------------
233    Serial.begin(9600);  // 啟用串列埠監看視窗
234    //--------------------------------------------------------------------
235    button.attachClick(singleClick);              // 單擊
236    button.attachDoubleClick(doubleClick);        // 雙擊
237    button.attachLongPressStart(longPressStart);  // 長按開始
238    button.attachDuringLongPress(longPress);      // 長按過程中
239    button.attachLongPressStop(longPressStop);    // 長按結束
240    //--------------------------------------------------------------------
241
242  }
243
```

4.各狀態的功能

RGB LED

```
244  void loop() {
245    //-------------------------------------------------------------
246    button.tick(); // 定期偵測按鈕狀態
247    //-------------------------------------------------------------
248
249    if (startPlay == 1) // 當startPlay為1時，狀態機才運作
250    {
251      unsigned long currentMillis = millis();  // 當前的millis()時間
252      unsigned long currentMillis2 = millis();
253     if (currentMillis2 - previousMillis2 >= 500)
254    {
255      previousMillis2 = currentMillis2;
256      if(ledtime>2)
257     {
258     ledtime=0;
259     }
260         show_color(ledtime);
261       ledtime=ledtime+2;
262    }
```

蜂鳴器1

```
265    switch(state)
266    {
267      case 1:   // 播放音符
268        if (i < total)  // 還有音符沒播完
269        {
270          note_duration = base_duration * song[i].length; // 計算當前音符持續時間
271          ledcWriteTone(pin, song[i].frequency);          // 發出聲音
272          previousMillis = currentMillis;                 // 更新前一次的millis()時間
273          state = 2;                                      // 進入狀態2
274        }
275        else  // 所有音符已經播完
276        {
277          previousMillis = currentMillis; // 更新前一次的millis()時間
278          state = 4;  // 進入狀態4
279        }
280        break;
281
282      case 2:   // 音符間隔
283        if (currentMillis - previousMillis >= note_duration)  // 音符播放時間到
284        {
285          ledcWriteTone(pin, 0);          // 靜音
286          previousMillis = currentMillis; // 更新前一次的millis()時間
287          state = 3;                      // 進入狀態3
288        }
289        break;
290
291      case 3:   // 切換至下一個音符
292        if (currentMillis - previousMillis >= note_interval)  // 間隔時間到
293        {
294          i++;        // 索引值+1
295          state = 1;  // 進入狀態1 (State 1 會自己重置計時器，所以這裡不用重置)
296        }
297        break;
298
299      case 4:   // 播放結束
```

蜂鳴器2

```
switch(state2)
{
  case 1:   // 播放音符
    if (j < total2)  // 還有音符沒播完
    {
      note_duration2 = base_duration * song2[j].length; // 計算當前音符持續時間
      ledcWriteTone(pin2, song2[j].frequency);          // 發出聲音
      previousMillisSong = currentMillis;                 // 更新前一次的millis()時間
      state2 = 2;                                       // 進入狀態2
    }
    else  // 所有音符已經播完
    {
      previousMillisSong = currentMillis; // 更新前一次的millis()時間
      state2 = 4;  // 進入狀態4
    }
    break;

  case 2:   // 音符間隔
    if (currentMillis - previousMillisSong >= note_duration2)  // 音符播放時間到
    {
      ledcWriteTone(pin2, 0);          // 靜音
      previousMillisSong = currentMillis; // 更新前一次的millis()時間
      state2 = 3;                      // 進入狀態3
    }
    break;

  case 3:   // 切換至下一個音符
    if (currentMillis - previousMillisSong >= note_interval)  // 間隔時間到
    {
      j++;        // 索引值+1
      state2 = 1;  // 進入狀態1 (State 1 會自己重置計時器，所以這裡不用重置)
    }
    break;

  case 4:   // 播放結束
    if (currentMillis - previousMillis >= 2000)  // 暫停2秒時間到
    {
      j = 0;      // 索引值歸零(重新開始)
      state2 = 1;  // 進入狀態1(重新開始)
    }
    break;
}
```

問題與解決

問題:
一首歌通常會有主旋律與伴奏，過去常見的做法就是同時執行兩個一樣的程式，主旋律跟伴奏，擁兩個開發板去手動控制旋律與伴奏。然而，這很難使兩者節拍同步，所以我這次打算用millis狀態機在同一快板子上做兩個聲部，這樣能更精準的節奏配合。

不過，在實作過程中，其中有兩個是我一直遇到的問題

1.兩聲部會互相干擾

問題

剛開始，我檢查了很多遍是否遺漏了哪些參數，結果都沒有明顯的改善。
由於我才剛學程式沒多久，學到的很有限，與其這樣一直檢查，不如我就把問題透過Ai (Gemini)協助分析。

Ai 提出
針對「通道（Channel）」、「Timer」、「RGB LED 衝突」等逐一排查。

解決

我按照它的提示一一逐步的做修正，直到我把兩個蜂鳴器的頻率調不一樣(一個2000Hz，另一個2001Hz)時，就成功了

而Ai給我的解釋是:
「這是因為硬體共享同一個 Timer 或引腳頻率設定機制，當兩者頻率完全相同時，容易造成硬體計時器的資源爭奪與訊號疊加衝突。」

2.節拍隨時間偏移

問題

我是按照樂譜的節拍與比例一個個設定，但兩聲部合起來卻會越來越亂

這是因為我忽略了一個重要的事-->「間隔時間」。
間隔時間是為了避免聲音黏在一起，單聲部時可以不去考慮它，但到了兩個聲部，為了要讓節拍一致，必須要考慮進去。

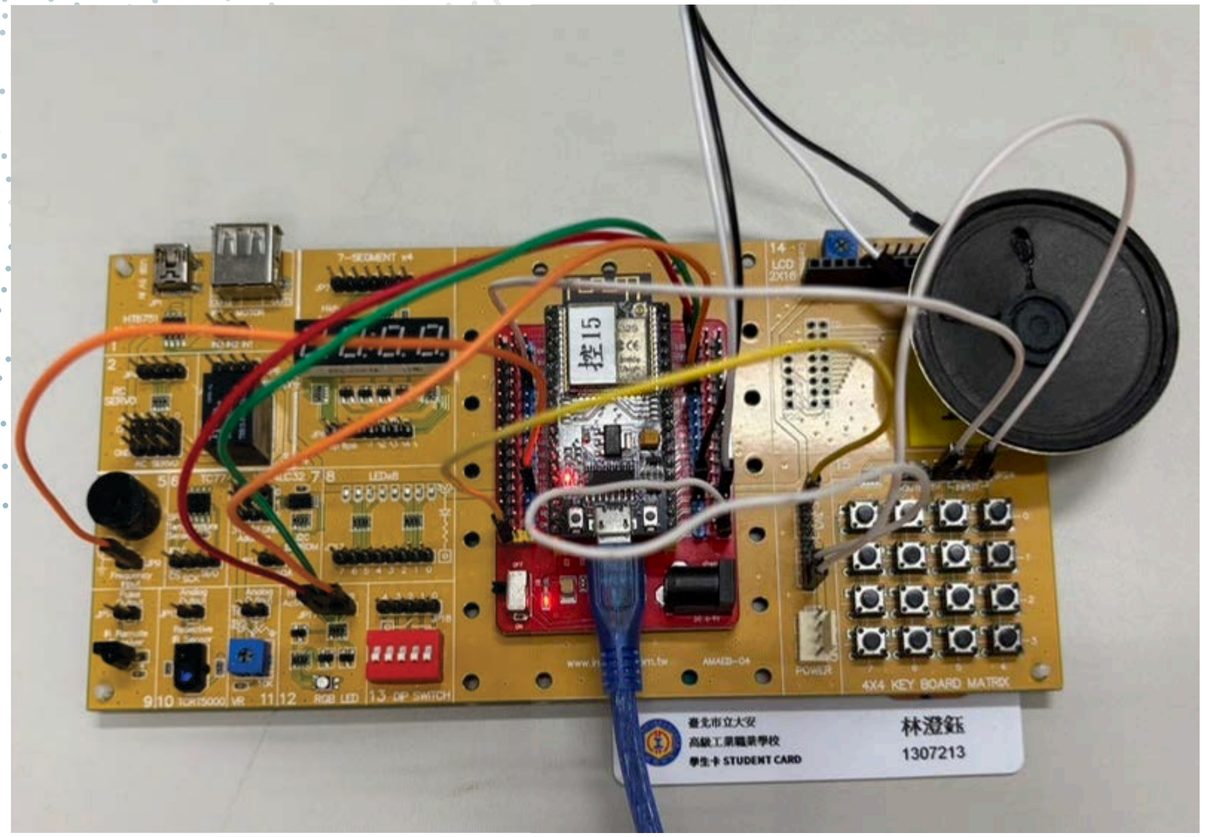
解決

舉例來說，以4拍來說，當主旋律播1個4拍長音，而伴奏為4個1拍短音，由於每一個音有間隔，這樣伴奏比主旋律多了3個間隔時間，因此我們必須把這多出的時間算入主旋律音符的持續時間

50/130=0.38... <----------秒數換算要加的值

50ms {C4, 4+0.38*3}　<----------補上多出的秒數
50ms {E4,1}, 50ms {E4,1}, 50ms {E4,1}, 50ms {E4,1} <----------多了 3 個 50ms

實作成果



連結

https://www.youtube.com/watch?v=hUKjgdTyq1k

https://youtu.be/ZNOQA4zXGaI

心得討論

由於這是對過去所學的內容進行一次整合，把過往積累的程式拆解重組，而形成這次的程式，省了一些不必要的時間，不過為了做出雙聲部在一個開發版，花了很多時間找問題、除錯。

我遇到的問題有些超出我目前所學的範圍，我只好透過Ai引導我找出問題，Ai真的很方便，能快速地分析其中可能的問題，省了我很多時間與力氣。
在伴奏節拍的部分，我是自己校正每個音符的拍子，這是花最多時間的，但當成功聽到兩聲部節拍對上並達到預期的目標，我很高興，感受到程式實作中的樂趣。

隨著Ai的進步，它為我們提供不少便利與節省時間，但若只是一味的用Ai給的程式解決問題，而放棄用大腦思考，這樣是學不到東西的。因此我把它當作一個輔助我學習的工具，縮短開發時的瓶頸。由於日常有很多事要做，很多想加的功能還未實現，像是透過七段顯示器顯示撥放時長，雙擊按鈕來決定只播一個聲部等等，如果哪天有空，我會想嘗試挑戰一下!

資料來源

https://wp.640629.xyz/

https://gemini.google.com/u/2/app?pageId=none

https://wokwi.com/