智慧居家監控實習

作品：音樂播放器

控制二乙 06 陳詡涵

指導老師：張漢民

目錄

壹、摘要

本次課程利用Onebutton函式庫操作按鈕控制蜂鳴器播放音樂，並讓LED顯示播放狀態，以實現按鈕單擊播放/暫停，雙擊切換歌曲，長按回到待機狀態，而LED在播放時跟著旋律閃爍，暫停時保持恆亮。
同時用millis()，取代傳統的delay()，以確保音樂在播仿的同時操作按鈕仍可更改狀態。並使用Ardiuno UNO板完成目標。

貳、研究動機

1. 結合外部元件與函式庫完成綜合應用：除了基礎的蜂鳴器音樂撥放，本次實作同時整合其他元件。透過導入Onebutton函式庫來處理按鈕開關狀態，同時讓LED指示燈能精準地與音樂播放狀態同步連動。
2. 使用millis()解決傳統delay()函式造成的程式阻塞問題：過去在撰寫音樂撥放程式時，使用delay()的話系統必須等待時間過去後，才繼續執行下個指令，導致撥放音樂同時，無法執行其他任務。透過millis()以順利達成多工處理。

參、主題

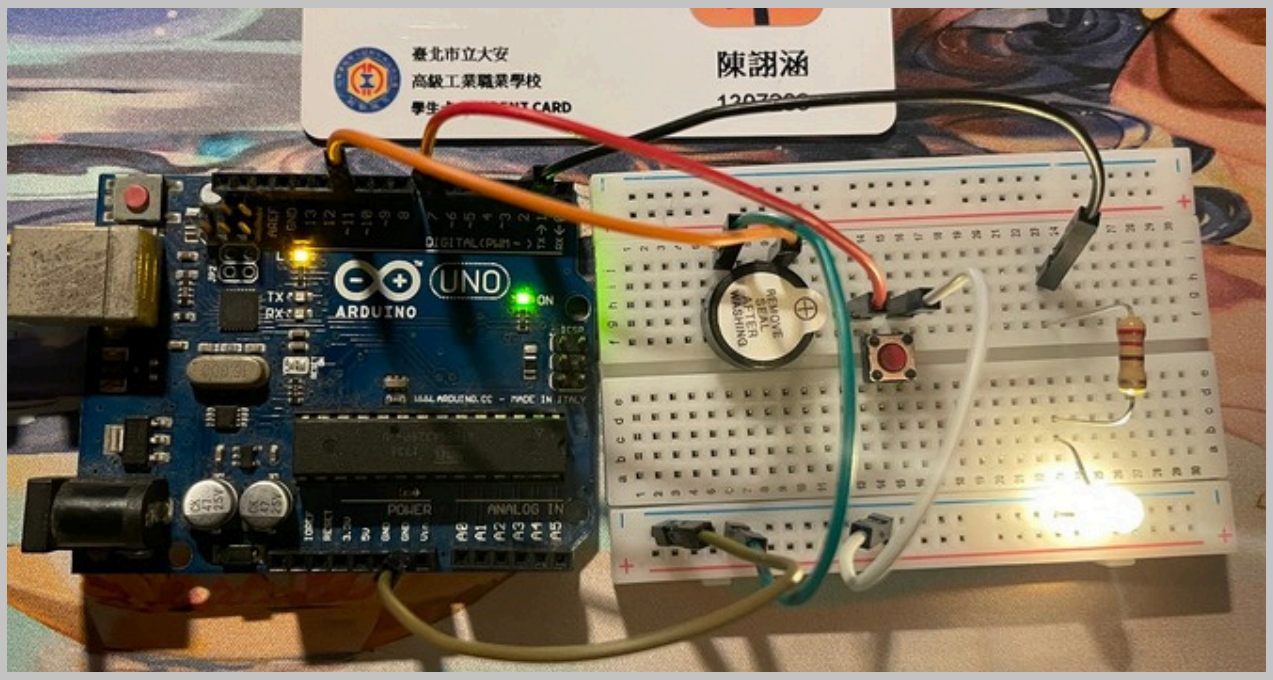
(一)主要硬體裝置：

1. Arduino UNO控制器
2. 蜂鳴器
3. 發光二極體
4. 按鈕開關
5. LED限流電阻
6. 麵包板

(二)功能說明

- 開機後 音樂暫停 LED燈亮
- 按一下按鈕-播放/暫停
- 按兩下按鈕-切換歌曲
- 長按按鈕-reset
- 當音樂播放時 LED跟著閃爍
- 當音樂暫停時 LED恆亮

肆、實作步驟(硬體)



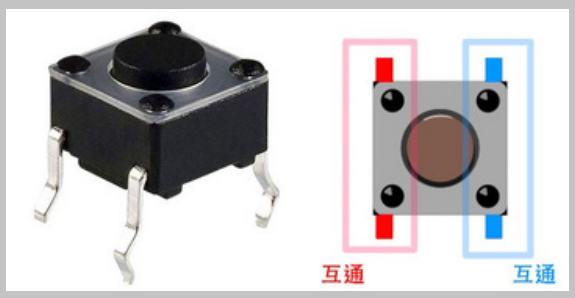
接線

- Pin11接至蜂鳴器
- Pin 7 接至按鈕開關
- Pin 2 接至LED

元件

蜂鳴器

發光二極體



按鈕開關

肆、實作步驟(軟體)

(一)定義音符頻率

```
// 音符範圍：C3~C7（涵蓋低音到高音），包含所有升降音（♯、♭）
// 單位：赫茲（Hz）

// 低音區（C3 ~ B3）
#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
#define Cs4 277  // C#4 / Db4
#define D4  294
#define Ds4 311  // D#4 / Eb4
#define E4  330
#define F4  349
#define Fs4 370  // F#4 / Gb4
#define G4  392
#define Gs4 415  // G#4 / Ab4
#define A4  440
#define As4 466  // A#4 / Bb4
#define B4  494
```

```

// 高音區（C5 ~ B5）
#define C5  523
#define Cs5 554  // C#5 / Db5
#define D5  587
#define Ds5 622  // D#5 / Eb5
#define E5  659
#define F5  698
#define Fs5 740  // F#5 / Gb5
#define G5  784
#define Gs5 831  // G#5 / Ab5
#define A5  880
#define As5 932  // A#5 / Bb5
#define B5  988

// 超高音區（C6 ~ B6）
#define C6  1047
#define Cs6 1109  // C#6 / Db6
#define D6  1175
#define Ds6 1245  // D#6 / Eb6
#define E6  1319
#define F6  1397
#define Fs6 1480  // F#6 / Gb6
#define G6  1568
#define Gs6 1661  // G#6 / Ab6
#define A6  1760
#define As6 1865  // A#6 / Bb6
#define B6  1976

```

```
// 超高音區（C7）
#define C7  2093
#define Cs7 2217  // C#7 / Db7
#define D7  2349
#define Ds7 2489  // D#7 / Eb7
#define E7  2637
#define F7  2794
#define Fs7 2960  // F#7 / Gb7
#define G7  3136
#define Gs7 3322  // G#7 / Ab7
#define A7  3520
#define As7 3729  // A#7 / Bb7
#define B7  3951
#define REST 0            // 休止符（頻率 0 即無聲）
struct Note {
  int frequency;          // 音頻（Hz）
  float length;             // 音長（為 base_duration 的倍數）
};

```

(二)音樂資料：音符序列與音符播放時間

```
// 建立一個 Note 物件陣列，名稱為 song
Note song[] = {
  {E6, 2}, {D6, 1}, {C6, 2}, {D6, 1}, {E6, 1.5}, {F6, 0.5}, {E6, 1}, {D6, 3},
  {E6, 2}, {D6, 1}, {C6, 2}, {D6, 1}, {E6, 1.5}, {F6, 0.5}, {E6, 1}, {D6, 3},
  {E6, 2}, {D6, 1}, {C6, 2}, {D6, 1}, {E6, 1.5}, {F6, 0.5}, {E6, 1}, {D6, 3},
  {E6, 2}, {D6, 1}, {C6, 2}, {D6, 1}, {E6, 1.5}, {F6, 0.5}, {E6, 1}, {D6, 1},
  {G4, 1}, {G4, 1}, {F5, 1}, {F5, 1}, {E5, 1}, {F5, 1}, {E5, 1}, {D5, 1},
  {D5, 1}, {D5, 1}, {C5, 0.5}, {C5, 0.5},{F5, 1}, {E5, 1}, {D5, 1},
  {D5, 1}, {C5, 1}, {C5, 0.5}, {D5, 0.5}, {E5, 3},
  {E5, 3}, {E5, 1}, {G5, 1}, {C6, 1},
  {B5, 2}, {C6, 1}, {B5, 2}, {C6, 1},
  {B5, 0.5}, {A5, 0.5}, {G5, 1}, {G5, 1}, {D5, 1}, {F5, 1},
  {F5, 2}, {E5, 1}, {E5, 2}, {G4, 1},
  {F5, 1}, {E5, 1},{D5, 1}, {E5, 2}, {G5, 1},
  {C5, 2},{C5, 3},{C5 ,1},
  {D5, 1},{C5,1.5},{C5, 0.5},{C5,1},{G5,1},{C5,1},
  {F5,2},{E5,1},{D5,1},{C5,1},{C5,1},
  {C5,5},{C5,0.5},{D5,0.5},
  {E5,1},{E5,1},{D5,1},{F5,1},{E5,1},{D5,1},
  {D5,2},{C5,1},{F5,1},{E5,1},{D5,1},
  {D5,1},{C5,1},{C5,0.5},{D5,0.5},{E5,3},
  {E5,1},{G5,1},{C6,1},
  {B5,2},{C5,1},{B5,2},{C5,1},
  {B5,0.5},{A5,0.5},{G5,2},{G5,1},{D5,1},{F5,1},
  {F5,1},{E5,1},{E5,1},{E5,2},{G4,1},
  {F5,1},{E5,1},{D5,1},{E5,2},{G5,1},
  {C5,5},{C5,1}
};
//第二首歌 千與千尋 那個夏天 前奏
Note song1[] = {
{REST, 1}, {E5 ,1}, {E5 ,1}, {E5 ,1}, {D5 ,1}, {E5 ,1}, {A5, 1},
{E5 ,1}, {D5, 1}, {D5, 1}, {D5, 5},
{REST, 1}, {D5 ,1}, {D5 ,1}, {D5 ,1}, {C5, 1}, {D5 ,1}, {G5, 1},
{D5 ,1}, {C5, 1}, {B4, 1}, {C5,1}, {C5, 3}, {A4,0.5}, {B4, 0.5},
{C5, 1}, {C5, 1}, {C5, 1}, {C5, 1}, {C5, 3}, {A4, 0.5}, {B4, 0.5},
{C5, 1}, {C5, 1}, {C5, 1}, {C5,1}, {B4, 1}, {C5, 2}, {A4, 0.5}, {B4, 0.5},
{C5,1}, {C5, 1}, {C5, 1}, {C5, 1}, {C5, 1}, {G5, 2}, {C5, 1},
{D5, 8},
};
```

(三)定義接角、變數、設定函式庫

```
// 音符總數 = melody 陣列總位元組數 / 單一整數位元組數
int total = sizeof(song) / sizeof(Note);
int total1 = sizeof(song1) / sizeof(Note);
// ------------------------------------------------------------------------
int pin = 11;              // 蜂鳴器連接腳位 (GPIO 15)
int base_duration = 200;  // 音符基準時間 (單位: ms，決定節奏快慢)
int note_interval = 100;  // 音符間的短暫間隔 (單位: ms，避免聲音黏在一起)
// ------------------------------------------------------------------------
int i = 0;       // 當前播放的音符索引
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 song_change =0; //切換歌曲
// ------------------------------------------------------------------------
unsigned long previousMillis = 0;   // 上一次狀態開始的時間
int note_duration = 0;              // 當前音符應該持續的時間
// ------------------------------------------------------------------------
#include <OneButton.h>          // 引用OneButton函式庫
#define pb 7                   // 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

//------------------------------------------------------------------------
```

(四)Onebutton函式庫副程式

```
// 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
  {
    noTone(pin);   // 靜音
  }
}

// 雙擊處理
void doubleClick() {
  Serial.println("Button Double Click. Change song");
  song_change++;
  if (song_change >1) song_change=0;
  i = 0;       // 當前播放的音符索引
  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
  noTone(pin);   // 靜音
}

// 長按開始
void longPressStart() {
  Serial.println("Button Long Press Start.");
  noTone(pin);   // 靜音
}

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

// 長按結束
void longPressStop() {
  Serial.println("Button Long Press Stop.");
  i = 0;       // 索引值歸零(重新開始)
  state = 1;  // 進入狀態1(重新開始)
}
```

(五)主要啟動程式(開機狀態)

```
void setup() {
  //-------------------------------------------------------------------
  Serial.begin(9600);  // 啟用串列埠監看視窗
  //-------------------------------------------------------------------
  button.attachClick(singleClick);              // 單擊
  button.attachDoubleClick(doubleClick);        // 雙擊
  button.attachLongPressStart(longPressStart);  // 長按開始
  button.attachDuringLongPress(longPress);      // 長按過程中
  button.attachLongPressStop(longPressStop);    // 長按結束
  //-------------------------------------------------------------------
  pinMode(2,OUTPUT);
}
```

(六)音樂播放程式(包括LED控制)

```
201  void loop() {
202    //-------------------------------------------------------
203    button.tick(); // 定期偵測按鈕狀態
204    //-------------------------------------------------------
205    if (startPlay == 0){
206    digitalWrite(2,HIGH);
207    }
208    else if (startPlay == 1) // 當startPlay為1時，狀態機才運作
209    {
210      unsigned long currentMillis = millis();  // 當前的millis()時間
211
212      switch(state)
213      {
214        case 1:   // 播放音符
215        if (song_change == 0){
216          if (i < total)  // 還有音符沒播完
217          {
218            note_duration = base_duration * song[i].length; // 計算當前音符持續時間
219            tone(pin, song[i].frequency);          // 發出聲音
220            previousMillis = currentMillis;                // 更新前一次的millis()時間
221            digitalWrite(2,HIGH);   //第23腳輸出HIGH --> LED0 ON
222            state = 2;                                     // 進入狀態2
223          }
224          else  // 所有音符已經播完
225          {
226            previousMillis = currentMillis; // 更新前一次的millis()時間
227            state = 4;  // 進入狀態4
228          }
229        }
230        else if(song_change == 1){
231          if (i < total1)  // 還有音符沒播完
232          {
233            note_duration = base_duration * song1[i].length; // 計算當前音符持續時間
234            tone(pin, song1[i].frequency);          // 發出聲音
235            previousMillis = currentMillis;                // 更新前一次的millis()時間
236            digitalWrite(2,HIGH);  //第23腳輸出HIGH --> LED0 ON
237            state = 2;                                     // 進入狀態2
238          }
239          else  // 所有音符已經播完
240          {
241            previousMillis = currentMillis; // 更新前一次的millis()時間
242            state = 4;  // 進入狀態4
243          }
244        }
245        break;
247        case 2:   // 音符間隔
248          if (currentMillis - previousMillis >= note_duration)  // 音符播放時間到
249          {
250            noTone(pin);          // 靜音
251            previousMillis = currentMillis; // 更新前一次的millis()時間
252            digitalWrite(2,LOW);
253            state = 3;                      // 進入狀態3
254          }
255          break;
256
257        case 3:   // 切換至下一個音符
258          if (currentMillis - previousMillis >= note_interval)  // 間隔時間到
259          {
260            digitalWrite(2,HIGH);  //第23腳輸出HIGH --> LED0 ON
261            i++;        // 索引值+1
262            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)
263          }
264          break;
```

(七)音樂結束程式(包括LED控制)

```
266         case 4:   // 播放結束
267           if (currentMillis - previousMillis >= 2000)  // 暫停2秒時間到
268           {
269             digitalWrite(2,HIGH);  //第23腳輸出HIGH --> LED0 ON
270             i = 0;       // 索引值歸零(重新開始)
271             state = 1;  // 進入狀態1(重新開始)
272           }
273           break;
274       }
275     }
276
277   }
```

伍、實作結果(影片連結)

https://youtu.be/P7qTltdvsxw

陸、心得與討論

本次的課程，我學到了輸入不同頻率會讓蜂鳴器播出不同音階。
結合之前所學過的按鈕開關、LED控制、millis()函式以及Onebutton函式庫，完成音樂播放器。
同時我還加入了點兩下按鈕切換歌曲的功能，過程中，我以為我能以原本的陣列直接換成新陣列，但後來詢問老師之後，老師跟我說要交換陣列的話應該要用提取他的值，而不是直接b[ ]=a[ ]，所以我就重複寫了兩次音樂播放的部分，分別給不同的陣列去運作。
在修改程式的途中，我以為燈要閃可以在case1裡面放燈跟delay()就好，但後來發現這樣音樂結束得時候等待下一次循環播放時，燈不會恆亮，所以只好跟著case一起設狀態了。
此次課程我學到了很多，例如學會millis()的用法，對我以後專題應用蠻有幫助的。

柒、參考資料

蜂鳴器、ledcWriteTone()進階練習：結合按鈕開關、蜂鳴器、LED指示燈的音樂播放器 – 漢之民也

https://notebooklm.google.com/notebook/f3dcd7f8-1d00-4245-a634-6620b541f831

https://gemini.google.com/share/987cbd2402a3