以產生 100Hz 的 Sawtooth 為例：

每個週期可以使用的取樣點為

$$\frac{25600}{100}=256 \quad \text{（個取樣點）}$$

每個週期的取樣點上升一隔 Amplitude 的數為

$$\frac{255}{256}\fallingdotseq 1$$

所需的時間為

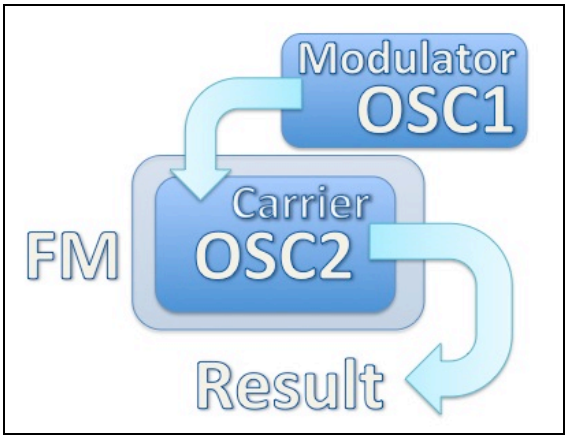
$$\frac{625}{16000000}\fallingdotseq 0.000039$$

$0.000039\times 25600=0.9984\fallingdotseq 1$（second）

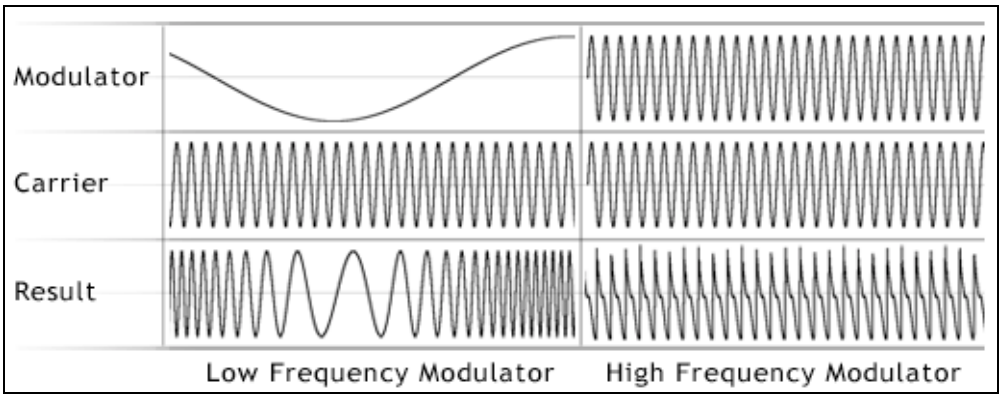
```
ISR(TIMER1_COMPA_vect){
  //Timer1 Interrupt
  //time變數上升，直到達到週期數後RESET time變數
  time += 1;
  if (time >= perSample)
    time = 0;
  /*
    波形產生
  */
  //Waveform直接輸出份配給8個pin，為二進位 8 bit
  digitalWrite(6,(Waveform&64)>>6);
  digitalWrite(7,(Waveform&128)>>7);
  PORTB = Waveform>>2;

}
```

圖表 26：Time Interrupt ISR 函數

4.1.4 頻率調變合成法（Frequency Modulation Synthesis）



圖表 27：FM 合成法流程圖



圖表 28：FM 波形示意圖

資料來源：

http://www.image-line.com/support/FLHelp/html/img_plug/plugin_gen_sytrus_fmdemo.gif

頻率調變合成法（Frequency Modulation Synthesis）[14]，為 John Chowning 所發明。作法是以一個外部訊號（調變波）來帶動另一個頻率訊號（載波），也就是載波會隨著調變波的改變而改變其頻率，此方法不影響振幅。

```
Modulationfrequency = analogRead(A4);  //調變頻率的輸入，來調變載波
```

圖表 29：FM 合成法 code

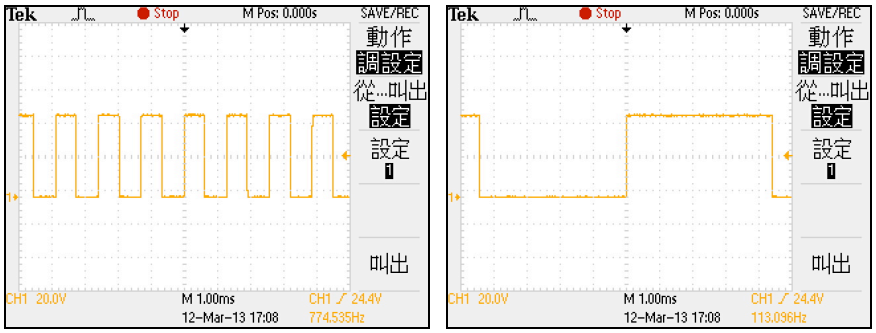
在程式碼中只需要使用 analogRead()這函數來接收外部訊號（意即調變波），來帶動原本震盪器所產生的訊號（意即載波）。包括 LFO 或是任何頻率訊號皆可輸入，便可產生 FM 的效果。

4.2 波型控制（Waveform Shaping）

本研究的 Voltage Controlled Oscillator 擁有 Waveform shaping [15]的功能，使用上述 DDS（Direct Digital Synthesis）的方法。本研究共可產生六種波型，方波、鋸齒波、反鋸齒波、三角波、正弦波、亂數波型（噪音）。這些波型皆以 Time domain 的狀態下 Amplitude 的不同，模擬類比訊號所產生不同的電壓波型。

4.2.1 方波（Square wave）

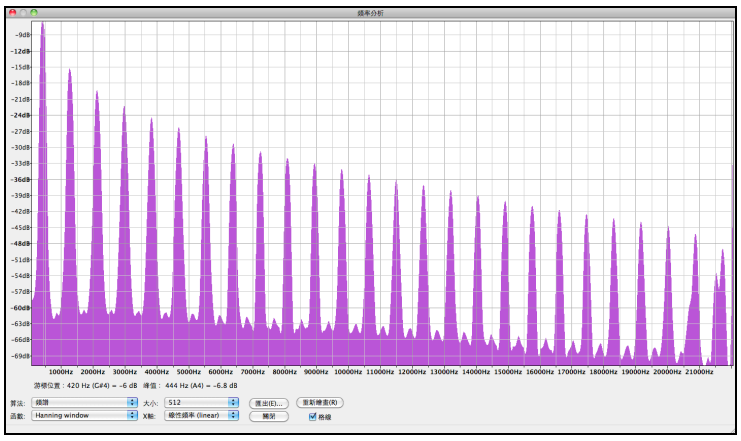
本研究的方波以 Duty Cycle 為 50%的週期產生。



圖表 30：示波器測量：方波

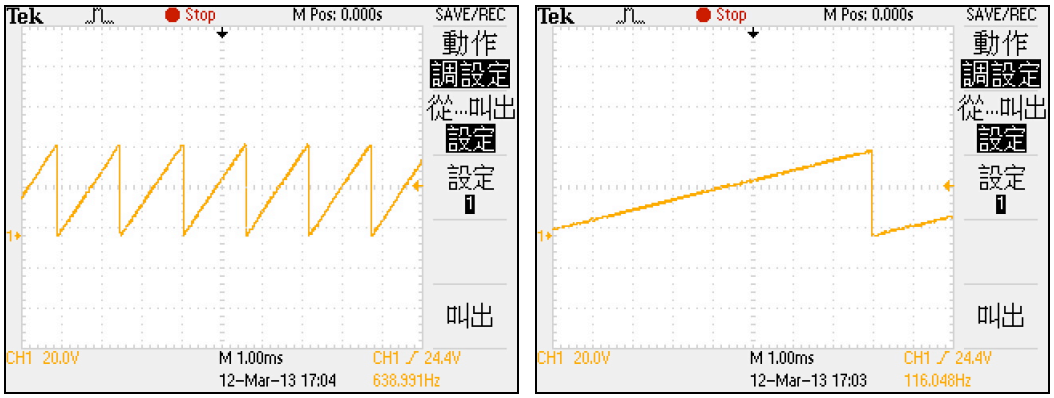
```
ISR(TIMER1_COMPA_vect){
  //Timer1 Interrupt
  //time變數上升，直到達到週期數後RESET time變數
  time += 1;
  if (time >= perSample)
    time = 0;

  //方波產生
  pulseWidth = perSample / 2; //設定Duty Cycle為50%
  if (pulseWidth <= time){  //週期的一半
    Waveform = 255;  //最大值255，HIGH
  }
  else{
    Waveform = 0;  //最小值為0，LOW
  }

  //Waveform直接輸出份配給8個pin，為二進位 8 bit
  digitalWrite(6,(Waveform&64)>>6);
  digitalWrite(7,(Waveform&128)>>7);
  PORTB = Waveform>>2;
}
```

圖表 31：方波 code



圖表 32：頻譜分析：方波

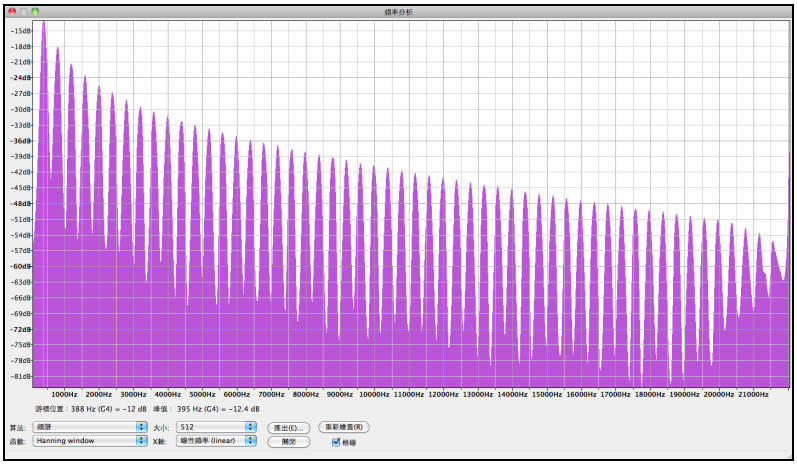
4.2.2 鋸齒波（Sawtooth wave）



圖表 33：示波器測量：鋸齒波

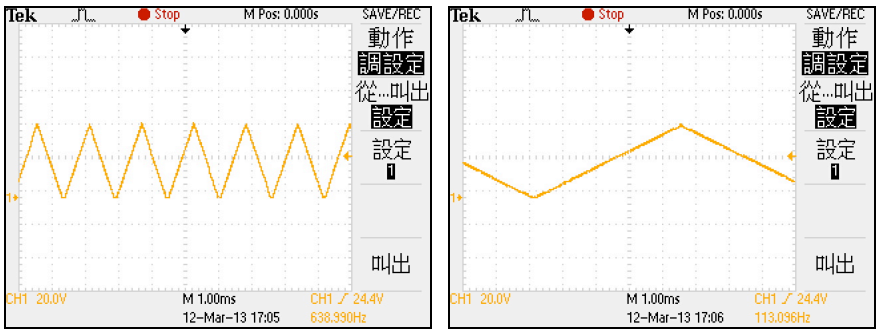
```
ISR(TIMER1_COMPA_vect){
  //Timer1 Interrupt
  //time變數上升，直到達到週期數後RESET time變數
  time += 1;
  if (time >= perSample)
    time = 0;
  //鋸齒波產生
  if (time==0){
    //當time為0，則sawtoothWave也歸0
    sawtoothWave=0;
  }
  else
 //每次sawtoothWave加上sawtoothAmp為
 //當下SAMPLE點的值 (Amplitude)
    sawtoothWave+=sawtoothAmp;

  Waveform = sawtoothWave;

  //Waveform直接輸出份配給8個pin，為二進位 8 bit
  digitalWrite(6,(Waveform&64)>>6);
  digitalWrite(7,(Waveform&128)>>7);
  PORTB = Waveform>>2;
}
```

圖表 34：鋸齒波 code



圖表 35：頻譜分析：鋸齒波

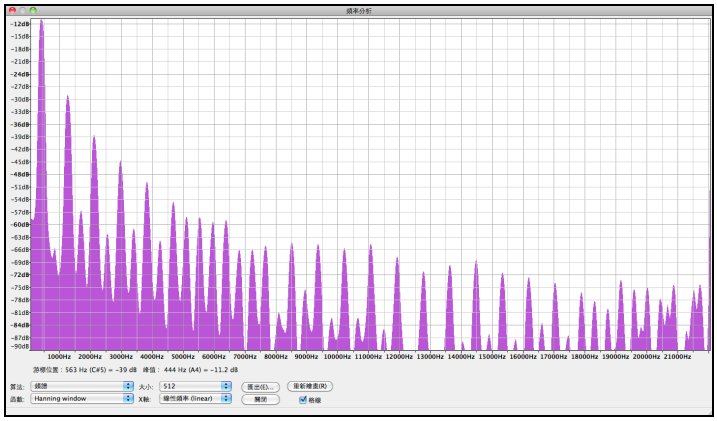
4.2.3 三角波（Triangle wave）



圖表 36：示波器測量：三角波

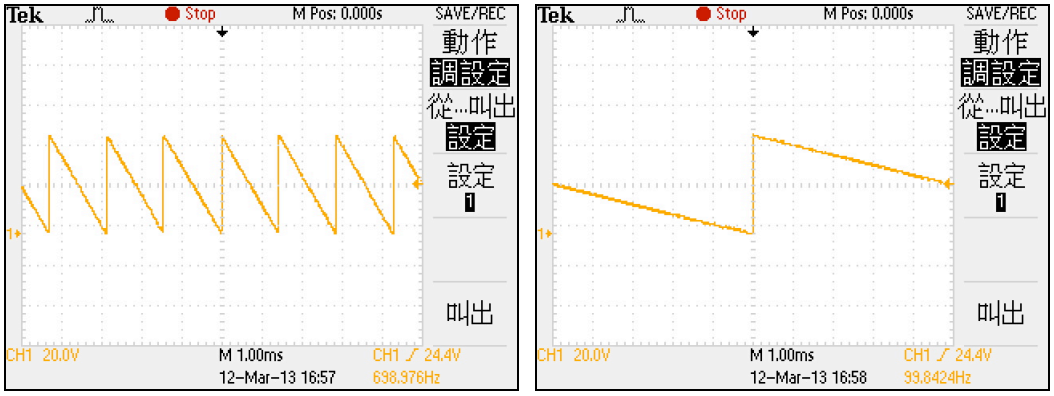
```
ISR(TIMER1_COMPA_vect){
  //Timer1 Interrupt
  //time變數上升，直到達到週期數後RESET time變數
  time += 1;
  if (time >= perSample)
    time = 0;
  //三角波產生
  if((perSample-time) > time) {  //週期前半
    if (time == 0)
      triangleWave = 0;
    else
      //每次triangleWave加上triangleAmp為
      //當下SAMPLE點的值 (Amplitude)
      triangleWave += triangleAmp;
  }
  else{
    //每次triangleWave減去triangleAmp為
    //當下SAMPLE點的值 (Amplitude)
    triangleWave -= triangleAmp;
  }
  if (triangleWave>255) //受限最大值
    triangleWave = 255;
  else if (triangleWave<0) //受限最小值
    triangleWave = 0;

  Waveform = triangleWave;

  //Waveform直接輸出份配給8個pin，為二進位 8 bit
  digitalWrite(6,(Waveform&64)>>6);
  digitalWrite(7,(Waveform&128)>>7);
  PORTB = Waveform>>2;
}
```

圖表 37：三角波 code



圖表 38 頻譜分析：三角波

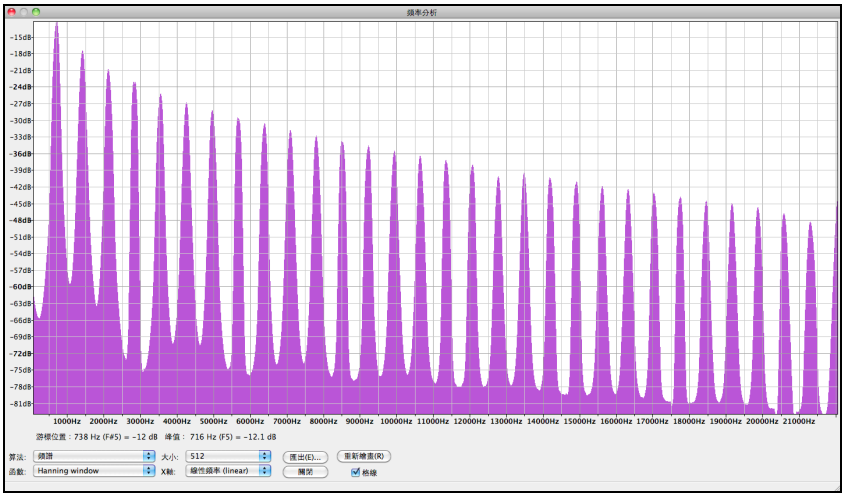
4.2.4 反鋸齒波（Reversed Sawtooth wave）



圖表 39：示波器測量：反鋸齒波

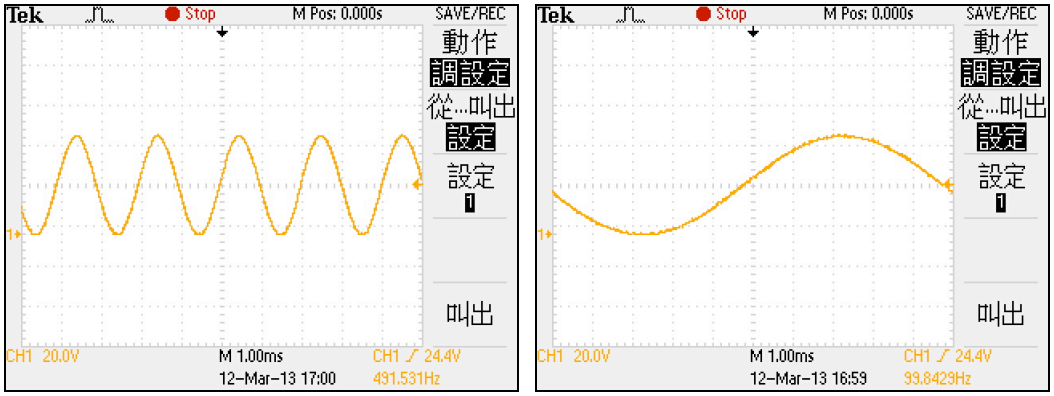
```
ISR(TIMER1_COMPA_vect){
  //Timer1 Interrupt
  //time變數上升，直到達到週期數後RESET time變數
  time += 1;
  if (time >= perSample)
    time = 0;
  //反鋸齒波產生
  if (time<255){
     //每次rampWave減去rampAmp為
     //當下SAMPLE點的值 (Amplitude)
    rampWave-=rampAmp;
  }
  else
    rampWave=255;  //ramp復原為255

  Waveform = rampWave;

  //Waveform直接輸出份配給8個pin，為二進位 8 bit
  digitalWrite(6,(Waveform&64)>>6);
  digitalWrite(7,(Waveform&128)>>7);
  PORTB = Waveform>>2;
}
```

圖表 40：反鋸齒波 code



圖表 41：頻譜分析：反鋸齒波

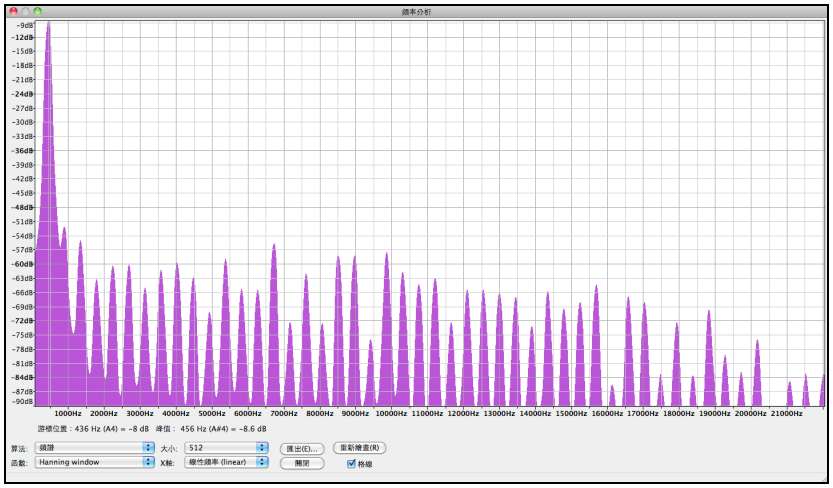
4.2.5 正弦波（Reversed Sawtooth wave）



圖表 42：示波器測量：正弦波

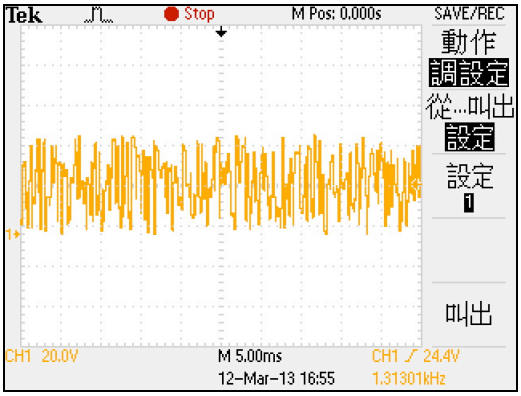
```
ISR(TIMER1_COMPA_vect){
  //Timer1 Interrupt
  //time變數上升，直到達到週期數後RESET time變數
  time += 1;
  if (time >= perSample)
    time = 0;

  //正弦波產生
  //每次sinWave為sinWave陣列當下SAMPLE點的值 (Amplitude)
  sinWave = time*sinAmp;
  //讀取sin20000陣列裡的數值
  Waveform = pgm_read_byte_near(sin20000 + sinWave);

  //Waveform直接輸出份配給8個pin，為二進位 8 bit
  digitalWrite(6,(Waveform&64)>>6);
  digitalWrite(7,(Waveform&128)>>7);
  PORTB = Waveform>>2;
}
```

圖表 43：正弦波 code



圖表 44：頻譜分析：正弦波

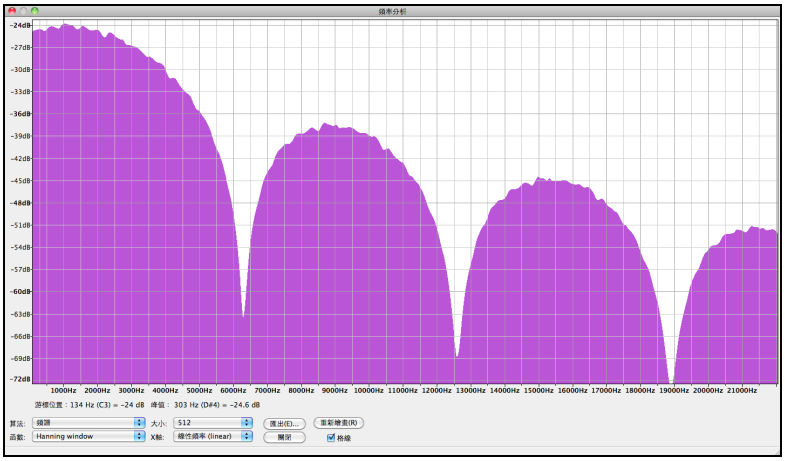
4.2.6 亂數波形（Noise wave）



圖表 45 示波器測量：亂數波形

```
ISR(TIMER1_COMPA_vect){
  //Timer1 Interrupt
  //time變數上升，直到達到週期數後RESET time變數
  time += 1;
  if (time >= perSample)
    time = 0;

  //亂數波型產生（噪音）
  //random函數給予亂數0至255
  noiseWave=random(0,255);
  Waveform = noiseWave;

  //Waveform直接輸出份配給8個pin，為二進位 8 bit
  digitalWrite(6,(Waveform&64)>>6);
  digitalWrite(7,(Waveform&128)>>7);
  PORTB = Waveform>>2;
}
```

圖表 46：亂數波形 code



圖表 47：頻譜分析：亂數波形

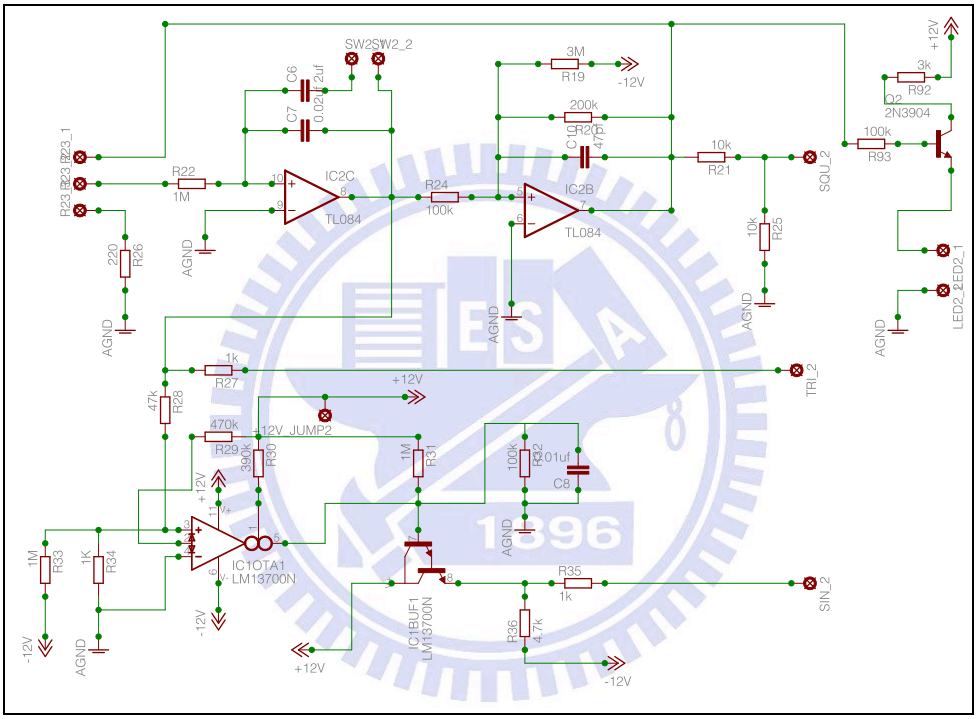
4.3 低頻震盪器（Low Frequency Oscillator）

低頻震盪器，與震盪器幾乎相同。低頻震盪器通常是指低於 20Hz 以下的頻率，這是人耳幾乎無法聽見的頻率，而人耳所能聽見的範圍大約是 20Hz~20000Hz，因此被廣泛使用於電子合成樂器的控制項目。

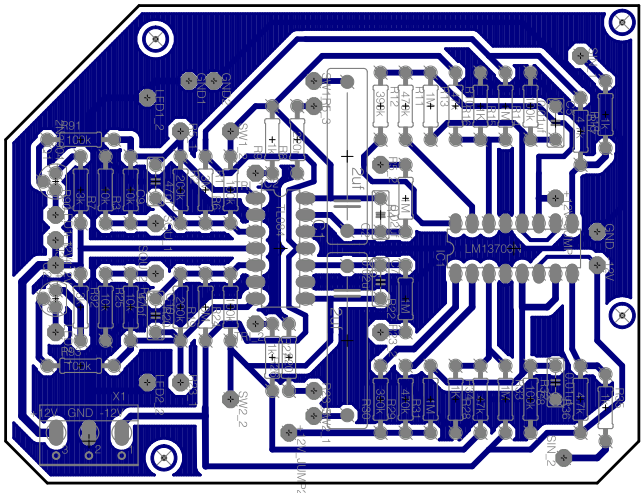
資料參考：

網站：Music From Outer Space

http://www.musicfromouterspace.com/



圖表 48：低頻震盪器原理圖



圖表 49：低頻震盪器 Layout 圖

4.4 邏輯閘功能（Logic Function）

邏輯閘（Logic gate）[16]是最為基礎數字運算的系統，透過組合能達到不同的運算效果，而運算結果為真和假也就是二進位的 1 或 0。有許多積體電路和電晶體即可做到這樣的運算，但由於複雜的運算需要大量的晶體或電路，導致材料的需求及實驗的難易度提昇，進而有廠商推出可以撰寫程式碼的邏輯運算單晶片，大大提昇數學運算系統的可行性。邏輯運算分為 AND（且）、OR（或）、XOR（互斥或）、NOT（反）；也可以將 NOT 加在 AND、OR、XOR 後面，則為 NAND（反且）、NOR（反或）、XNOR（反互斥或）。

本研究以數位方式產生週期頻率訊號，二進位做運算。加入邏輯運算及 mod 運算至系統中，可當做是一種效果器或是篩選器，讓接收進來的訊號透過邏輯運算，重新產生新的特殊波形。

```
void loop(){
  anaState0 = analogRead(A0);  //INPUT訊號1
  anaState1 = analogRead(A1);  //INPUT訊號2
}
ISR(TIMER1_COMPA_vect){

  // 訊號1  AND  訊號2
  anaStateCOUNT=(anaState1 & anaState0);

  wave=anaStateCOUNT;

  digitalWrite(6,(wave&64)>>6);
  digitalWrite(7,(wave&128)>>7);
  PORTB = wave>>2;
}
```

圖表 50：邏輯閘 code，以 AND mode 為範例

4.4.1 AND mode

AND gate 稱且閘、與閘，當兩個訊號輸入為高（1）時，輸出才會為高（1）。

表格 1：AND mode 範例真值表

Example X AND Y					
Input（Decimal）		Input（Binary）		Output（Decimal）	Output（Binary）
X	Y	X	Y	X AND Y	X AND Y
100	50	01100100	00110010	32	00100000

4.4.2 OR mode

OR gate 稱或閘，當兩個訊號輸入為低（0）時，輸出才會為低（0）。

表格 2：OR mode 範例真值表

Example X OR Y					
Input（Decimal）		Input（Binary）		Output（Decimal）	Output（Binary）
X	Y	X	Y	X AND Y	X AND Y
100	50	01100100	00110010	118	01110110

4.4.3 XOR mode

XOR gate 稱異或閘、互斥或閘，當兩個訊號有其中一個輸入為高（1）時，輸出會為高（1），其他則為低（0）。

表格 3：XOR mode 範例真值表

Example X XOR Y					
Input（Decimal）		Input（Binary）		Output（Decimal）	Output（Binary）
X	Y	X	Y	X AND Y	X AND Y
100	50	01100100	00110010	86	01010110

4.4.4 NAND mode

NAND gate 稱反且閘、非與閘、反及閘，當兩個訊號輸入為高（1）時，輸出才會為低（0）。

表格 4：NAND mode 範例真值表

Example X NAND Y					
Input（Decimal）		Input（Binary）		Output（Decimal）	Output（Binary）
X	Y	X	Y	X AND Y	X AND Y
100	50	01100100	00110010	223	11011111

4.4.5 NOR mode

NOR gate 稱反或閘、非或閘、或非閘，當兩個訊號輸入為低（0）時，輸出才會為高（1）。

表格 5：NOR mode 範例真值表

Example X NOR Y					
Input（Decimal）		Input（Binary）		Output（Decimal）	Output（Binary）
X	Y	X	Y	X AND Y	X AND Y
100	50	01100100	00110010	137	10001001

4.4.6 XNOR mode

XNOR gate 稱反互斥或閘、互斥或非閘，當兩個訊號有其中一個輸入為高（1）時，輸出會為低（0），其他則為高（1）。

表格 6：NOR mode 範例真值表

Example X XNOR Y					
Input（Decimal）		Input（Binary）		Output（Decimal）	Output（Binary）
X	Y	X	Y	X AND Y	X AND Y
100	50	01100100	00110010	169	10101001

4.4.7 MOD mode

modulo 簡稱 mod，為二元運算中的一種。是將某整數除以另個整數取得其餘數，可用作分類排序及加密解密…等應用。

表格 7：MOD mode 範例真值表

Example X MOD Y					
Input（Decimal）		Input（Binary）		Output（Decimal）	Output（Binary）
X	Y	X	Y	X AND Y	X AND Y
10	1	00001010	00000001	0	00000000
22	3	00010110	00000011	1	00000001
30	4	00011110	00000100	2	00000010
45	6	00101101	00000110	3	00000011

4.4.8 NMOD mode

反 MOD，意即將某整數除以另個整數後取餘數，再執行 NOT。

表格 8：NMOD mode 範例真值表

Example X MOD Y					
Input（Decimal）		Input（Binary）		Output（Decimal）	Output（Binary）
X	Y	X	Y	X AND Y	X AND Y
10	1	00001010	00000001	0	00000000
22	3	00010110	00000011	254	11111110
30	4	00011110	00000100	253	11111101
45	6	00101101	00000110	252	11111100

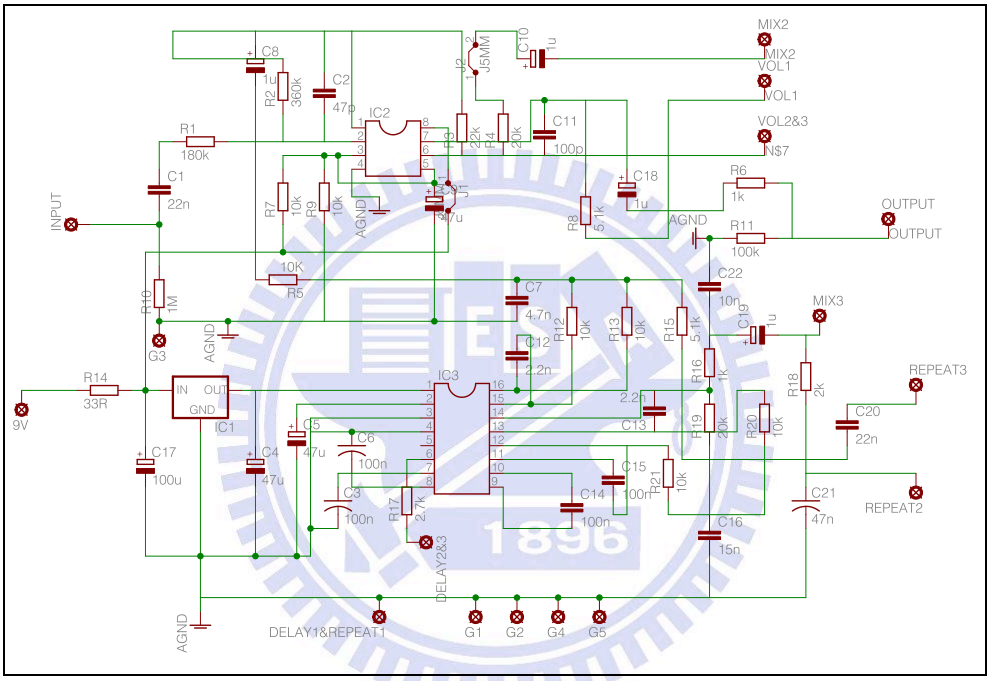
4.5 延遲功能（Delay Function）

Delay function 是使用 PT2399 Echo Processor IC，參考 IC 的 datasheet 來修改。這項目包括 Volume、Mix、Repeat、Delay 控制項目。PT2399 為低價的 Echo Processor IC，常使用於 Echo 或是 Delay 的效果器中。

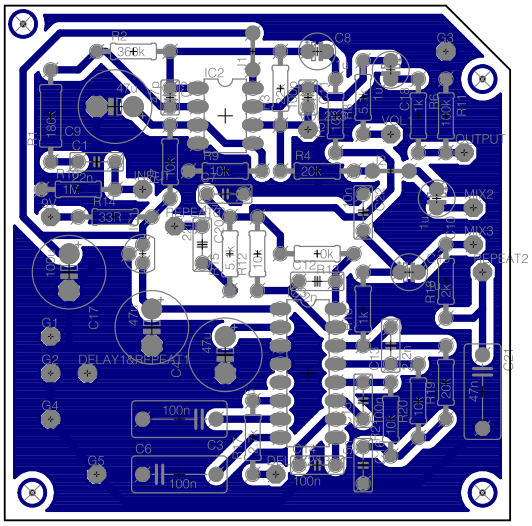
資料參考：

網站：PT2399 datasheet

http://sound.westhost.com/pt2399.pdf



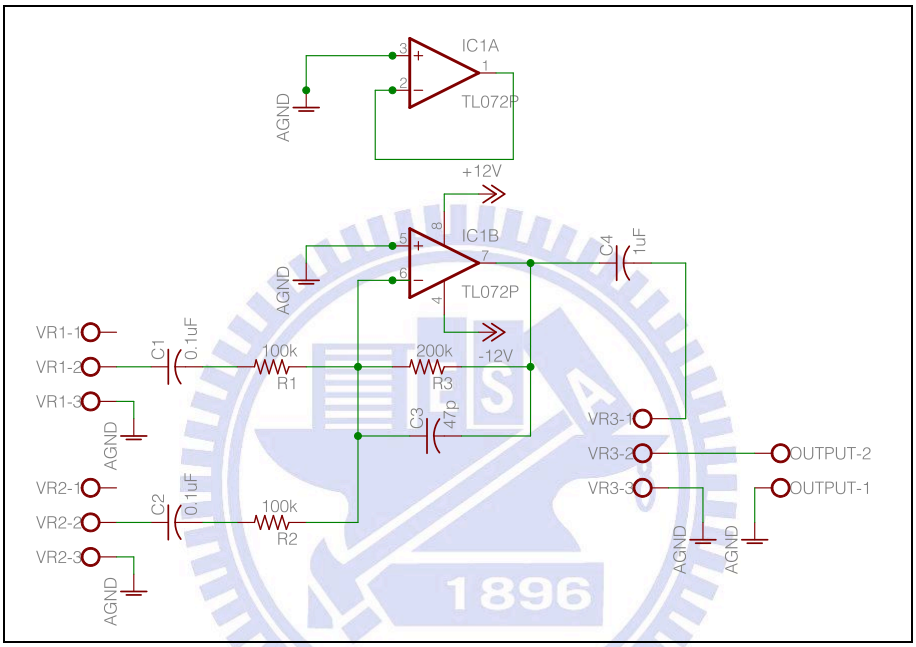
圖表 51：Delay Function 原理圖



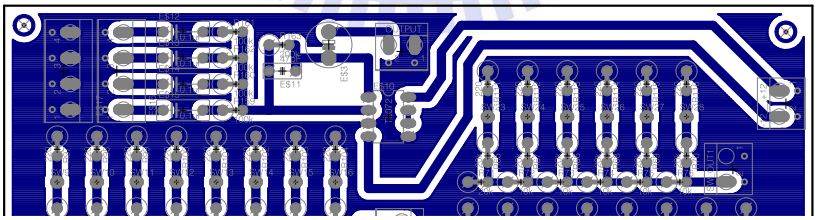
圖表 52：Delay Function Layout 圖

4.6 混音器功能（Audio Mixer）

本研究內建 Audio Mixer，共有四個訊號輸入供使用者選用。Audio Mixer 可以將原本未調變的聲音與效果調變後的聲音加總在一起；也可以透過每個聲音的大小比例，透過 Audio Mixer 上的 Gain 來調整，自由的混合出使用者所喜愛的聲音。本研究以簡易的訊號加總電路將所有訊號加總，在透過 OP 訊號放大 IC（TL074）將訊號輸出。



圖表 53：Audio Mixer 原理圖

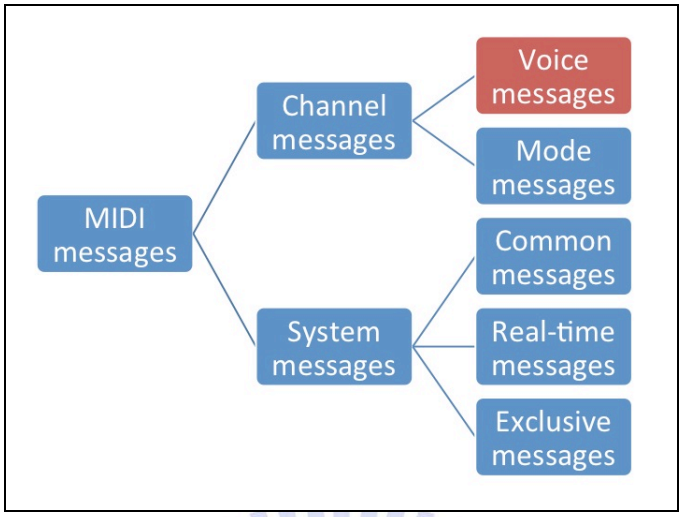


圖表 54：Audio Mixer Layout 圖

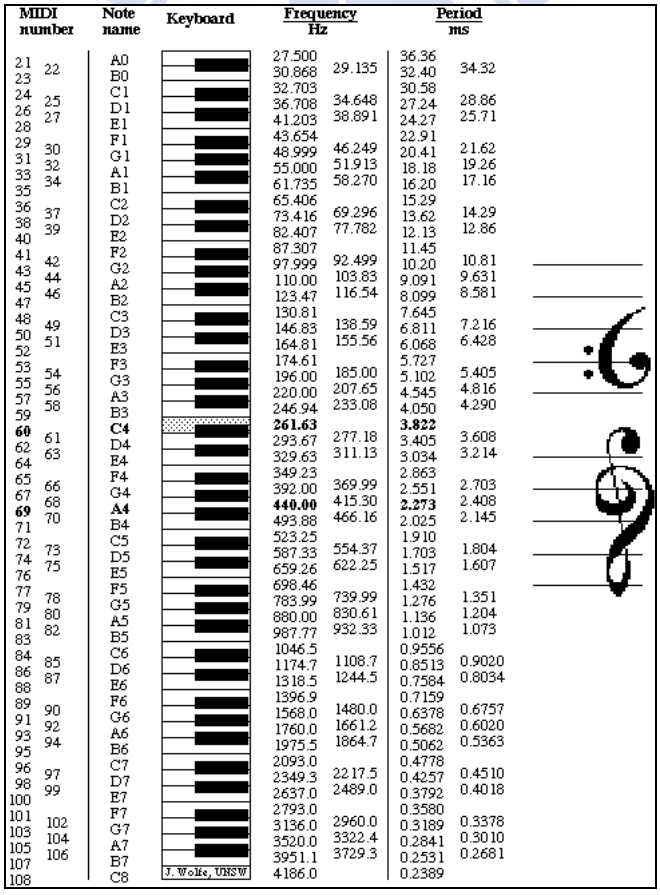
4.7 MIDI 介面（MIDI Interface）

MIDI（Musical Instrument Digital Interface）[17]，包含 IN 及 OUT，皆使用 5 pin 的連接座，但只使用其中 3 個 pin。MIDI 訊息的傳輸速率高達 312500 bps（bit per second），而 MIDI 訊息的架構分成兩種，Channel messages 和 System messages，這兩種架構裡又存在著幾種不同格式的訊息傳送方法。本研究的 MIDI 訊息的輸入與輸出，用以送出及接收 note

訊息；因此本研究控制的部份為 Channel messages 裡的 Voice message，其餘型態的訊息本研究並無使用。



圖表 55：MIDI 訊號結構圖



圖表 56：頻率對應 MIDI note

資料來源：

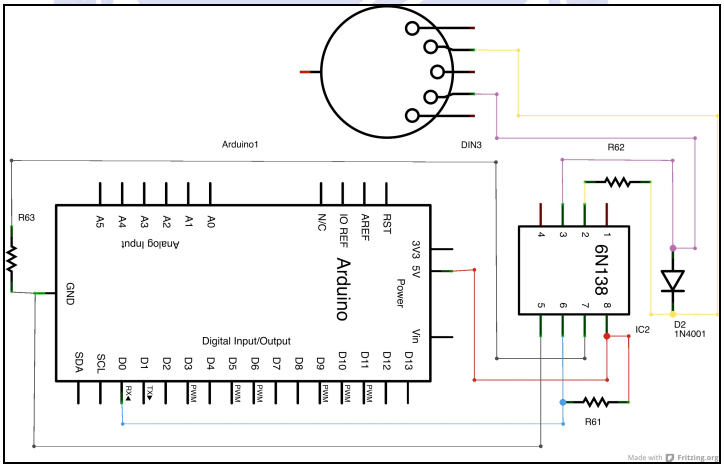
http://www.phys.unsw.edu.au/jw/graphics/notes.GIF

4.7.1 MIDI IN

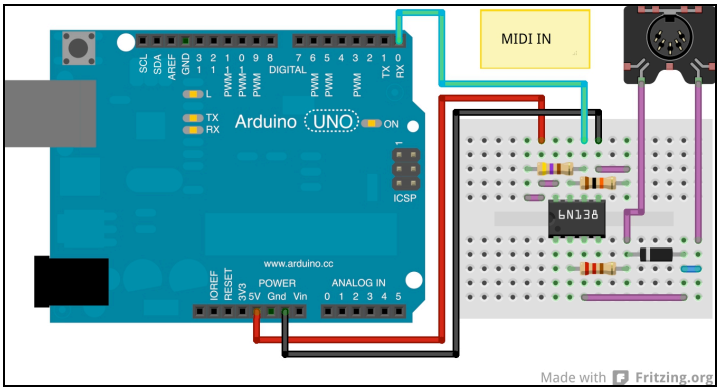
本研究透過 MIDI IN 所接收到 Channel messages 裡的 Voice messages，將訊息裡的 note number 對應於電子樂器所發出的頻率音高。

```
void setup() {
  Serial.begin(31250);
}
byte note2frequency[] PROGMEM={261};  //假設陣列裡的第60個為261
void MIDIreceive() {
  if(Serial.available()){
    cmd=Serial.read();
    note=Serial.read();
    velocity=Serial.read();
    if(cmd==144){
      //假設note=60
      frequency = pgm_read_byte_near(note2frequency + note);
    }
  }
}
void loop() {
  MIDIreceive()
}
```

圖表 57：MIDI IN code



圖表 58：MIDI IN 原理圖



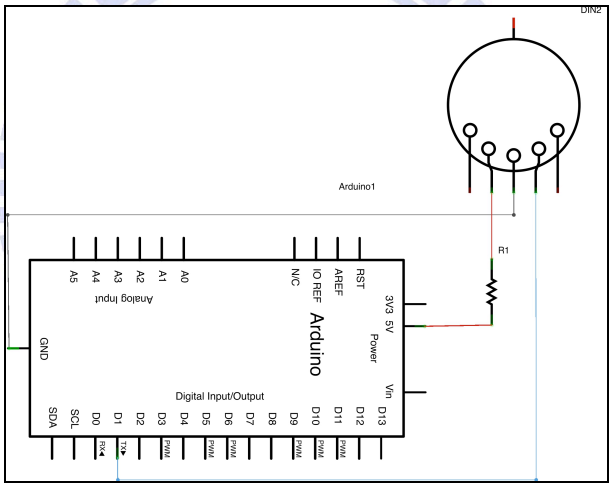
圖表 59：MIDI IN 線路圖

4.7.2 MIDI OUT

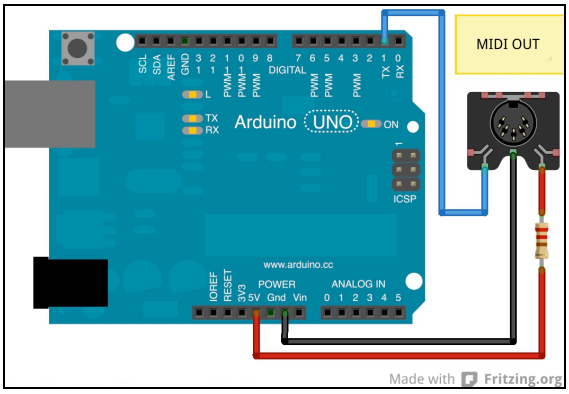
本研究按照 MIDI 訊號中 Channel messages 裡 Voice messages 的格式，將 channel、pitch 及 velocity 一同送出。

```
void setup() {
  Serial.begin(31250);
}
void MIDIsend(int cmd, int pitch, int velocity) {
  Serial.write(cmd);
  Serial.write(pitch);
  Serial.write(velocity);
}
void loop() {
   MIDIsend(144, 60, 127);  //送出 channel 1，note 60，velocity 127
   delay(100);
   MIDIsend(144, 60, 0);  //channel 1，note 60，velocity 0 將音量歸0
   delay(100);
}
```

圖表 60：MIDI OUT code



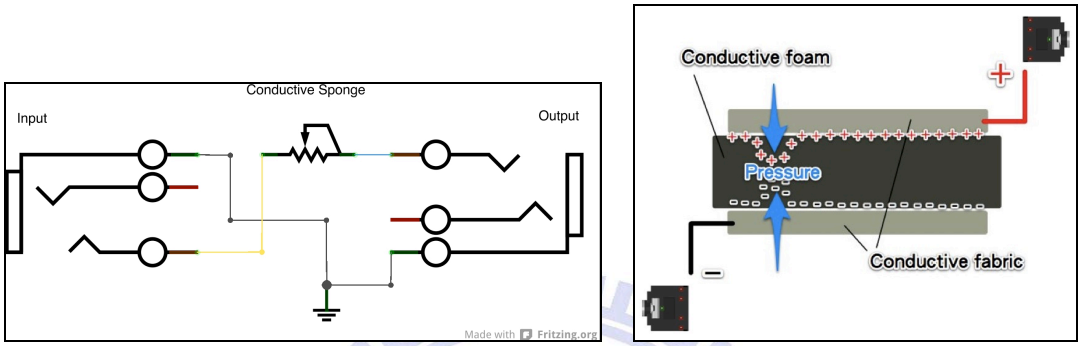
圖表 61：MIDI OUT 原理圖



圖表 62：MIDI OUT 線路圖

4.8 音量踏板（Volume Pedal）

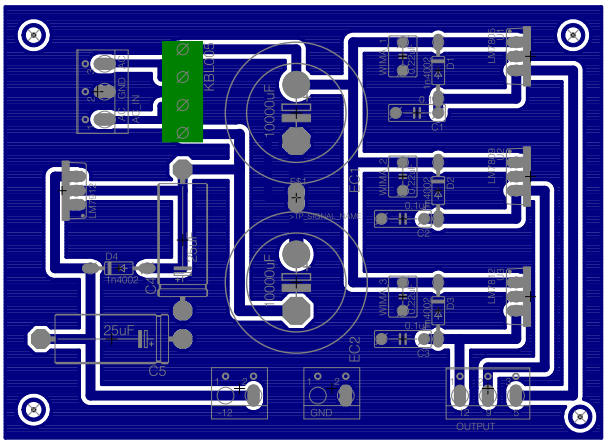
本研究加入可以使用外接音量踏板的選項，以導電海綿及導電布自製力度感應器，將此力度感應器當做音量踏板使用。大面積的力度感應，可以用腳來做踩踏的動作，透過導電海綿擠壓所造成的電阻變化，當做是音量控制的元件，達到樂器演奏時力度的變化。不論是使用本研究自製的音量踏板或是任何市售的音量踏板皆可以使用。



圖表 63：自製力度感應音量踏板示意圖

4.9 電源系統（Power System）

本研究為了讓電源獨立供應，設計了電源系統，採用 110 伏特的交流電（AC）透過變壓轉換成直流電（DC）。系統以 24V C.T.（含中心抽頭）的變壓器將 110VAC 轉為 24VAC，再透過橋式整流器將 AC 半波整流成正負 DC；正負電源會先以大容量電容做濾波，使電壓保持穩定；此時電壓約為 12×1.414＝16.9 約 17 伏特（負電源約為-17 伏特），接著以穩壓晶片 LM7812 與 LM7809 作為穩壓於 12V 和 9V 的系統，另外以 LM7912 穩壓於－12V。透過電源系統的搭配，只需插上一般市電插座便可直接使用。

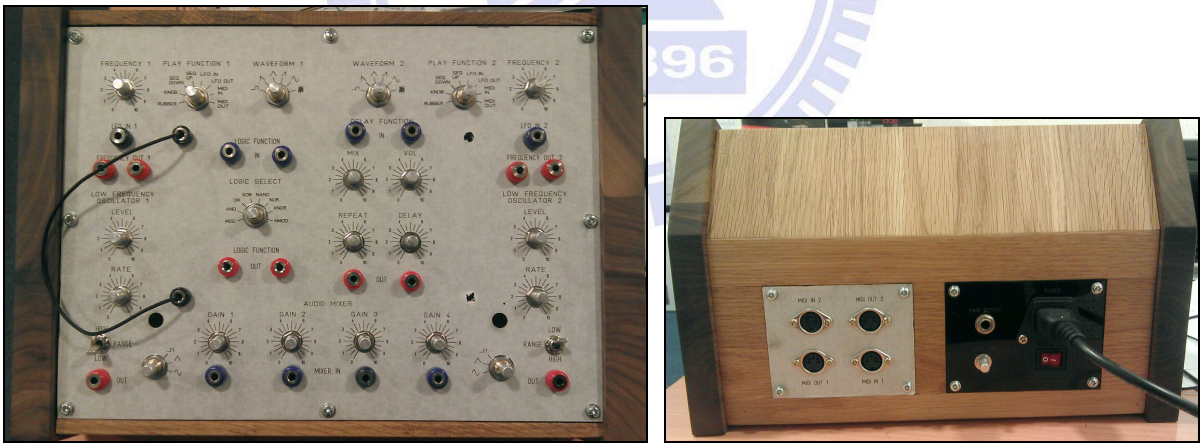


圖表 64：電源系統 Layout 圖

五、結論與未來展望

5.1 結論

本研究拉伸介面合成樂器設計與實作（Design and Implement of Elastic Interface Synthesizer）的成果為一台新介面電子合成樂器。此合成樂器以兩塊 Arduino 建立成兩個電壓控制震盪器的 Polyphonic 合成樂器。演奏頻率最高為 1046Hz（約為 C6），透過橡膠彈力介面，使用者可以很直覺的控制演奏時需要的音高；由於無刻度及指法的介面設計，使用者透過直覺的方式，來控制樂器的頻率音高。使用者可以自由的調整，包括方波、鋸齒波、三角波、反鋸齒波、正弦波及噪音；也可以自由的調整演奏模式，這些模式包含用來控制頻率的旋鈕或者是橡膠彈力拉伸介面、Sequencer、Low Frequency Oscillator、MIDI IN 或 OUT 的功能；而音色合成效果部分，目前有 Logic Function 和 Delay Function 兩種，透過這兩個 Function 讓音色合成上添加了許多變化。本研究完整實作出樂器的功能，並且能與各種外部樂器連結，讓喜愛聲音合成或是合成樂器的愛好者，皆能體驗到與以往不同介面合成樂器的魅力。



圖表 65：拉伸介面合成樂器完成品，正面（左）背面（右）

5.2 未來展望

根據本研究的製作過程與實作結果，整理出以下七點未來發展的方向或改進的項目：

5.2.1 增加聲音的品質

本研究使用 Arduino UNO 為主要核心，因 digital pin 的使用限制，只能使用 8 個 pin 來做 8 bit 的波形產生。為了增加聲音的品質，未來可以使用較為高階的 Arduino Mega 或是透過其他硬體或 IC 的輔助，使其波形的解析度能高達 16 或 32 bit，聲音的品質必然更加良好。

5.2.2 降低成品與資源使用

本研究使用三塊 Arduino UNO 來作為整體核心，且波形解析度為 8 bit，欲增加聲音的品質及降低製作成本，減少 Arduino 的使用與 Arduino 腳位的消耗是必要的。由於波形的解析度提昇至 16 或 32 bit bit 數上升，直接對應的是硬體的上升，相對的也是製作成本的上升。為了解決成本與降低資源使用的問題，又必須顧全聲音的品質，未來在設計上可以加入其他的硬體或是 IC 的輔助，如：Multiplexer、Shift register 或是直接以 Microcontroller ATmega2560 來設計整體電路…等。

5.2.3 加入其他控制功能

本研究目前以完善的 VCO 為來設計，若能增加更多的控制功能，將會為樂器帶來更多的聲音變化與可能性。如：更完整的 Sequencer，能調整節奏變化的選項；各種 Filter，LPF（Low Pass Filter）、BPF（Band Pass Filter）、HPF（High Pass Filter）…等；或是可以調整 ADSR 的 Envelope…等，皆能讓合成的音色豐富度大大提昇。

5.2.4 增加震盪器的數量

本研究目前以兩個發聲數的 Polyphonic 樂器設計，未來可以考慮增加更多的發聲數以提昇合成音色的複雜度與豐富性。如一個 Oscillator 為 16 bit 所設計，則四個發聲源就必須使用到 64 bit 的硬體。

5.2.5 橡膠彈力操控介面的伸長量

本研究目前所使用的橡膠彈力拉伸介面，其材質所展現的伸長量約是未伸長前的兩倍，如有更好的材質可以取代，則操控的效果一定可以更好。

5.2.6 增加 1V/OCT 功能

1V/OCT 這功能，是將電壓規格化的分配，當上升 1 伏特時，則上升一個八度，也就是說一個八度被平均的分配於 1 伏特當中。未來透過這樣的功能，任何有 1V/OCT 的合成樂器皆可以做串連，能做到穩定的音階變化，勢必能讓本研究有更多的操控選擇。

參考文獻

[1]Robert A. Moog, Voltage-Controlled Electronic Music Modules, Journal of the Audio Engineering Society, July 1965.
[2] Dr. Garth Paine; Dr. Jon Drummond, Developing an Ontology of New Interfaces for Realtime Electronic Music Performance, Electroacoustic Music Studies (EMS) 2009.
[3]Steve Mann; Ahmedullah Sharifi; Mike Hung; Russell Verbeeten, THE HYDRAULOPHONE: INSTRUMENTATION FOR TACTILE FEEDBACK FROM WATER FOUNTAIN FLUID STREAMS AS A NEW MULTIMEDIA INTERFACE, IEEE2006
[4] Roland Lamb; Andrew N. Robertson, Seaboard: a new piano keyboard-related interface combining discrete and continuous control, NIME2011
[5] Yu-Chung Tseng; Che-Wei Liu; Tzu-Heng Chi; Hui-Yu Wang, Sound Low Fun, NIME2011
[6] Larisa Katherine Montanaro, A Singer's Guide to Performing Works for Voice and Electronics. May, 2004, P.46
[7] MARCELO M. WANDERLEY; PHILIPPE DEPALLE, Gestural Control of Sound Synthesis, IEEE2004.
[8] Paradiso, J.A. Electronic music: new ways to play. IEEE1997, DECEMBER 1997.
[9] Bert Bongers, Physical Interfaces in the Electronic Arts
[10] HAROLD G.ALLES, Music Synthesis Using Real Time Digital Techniques, PROCEEDINGS OF THE IEEE,VOL. 68, NO. 4, APRIL 1980.
[11] Grzegorz Popek, Marian Kampik, Low-Spur Numerically Controlled Oscillator Using Taylor Series Approximation, OWD 2009, 17–20 October 2009.
[12] Nasser Sadati, Behnam Sedighi, A Non-Linear Neural D/A Converter for Direct Digital Frequency Synthesizers, 2006 International Joint Conference on Neural Networks, July 2006.
[13]Thomas L. Floyd, Digital Fundamentals With VHDL, 2003 by Pearson Education, Inc.P.806-808
[14] John M. Chowning, The Synthesis of Complex Audio Spectra by Means of Frequency Modulation, Journal of the audio Engineering Society, 1973.
[15] Mark Jenkins , Analog Synthesizers, 2007. P.4-5
[16]Thomas L. Floyd, Digital Fundamentals With VHDL, 2003 by Pearson Education, Inc.P.110-137.
[17]Ze-Nian Li; Mark S. Drew, Fundamentals of Multimedia, International Edition, 2004 by Pearson Education, Inc. P.139-147.