多功電子琴

https://youtu.be/rseT9PQvgxQ

陳品鈞(411086013)

國立臺北大學 通訊工程系

內容摘要

這次實驗專題主要應用 DTFM 信號技術來製作模擬電子琴。在這過程中，我們採用了多種功能以豐富其聲音效果。這包括了利用中斷服務（ISR）和 LCD 顯示屏來展示當前設置，以及使用 keypad 和 DIP 進行控制。此外，我們還運用了 buffer 來產生回音效果等多種聲音變化，從而增強了電子琴的聲音輸出多樣性。

1. 背景介紹

本次實驗專題的程式是一個進階版電子琴音頻處理系統，專門設計以配合 easyDSP-Expansion Board 使用。該系統運用 DTMF(雙音多頻）技術來產生多種不同的音調，並支持多樣的音頻輸出模式。此外，系統設計了四個按鈕（PB1、PB2、PB3、PB4），每個按鈕可切換至一種特定的音頻輸出模式，從而提供使用者更多樣化的音效選擇。

2. 方法與理論說明

首先我們會在 main 中的 while(1)新增 switch 功能，一個 switch 是用來使 keypad 運作的，而另一個 switch 則是用來控制 LCD 的切換以及要顯示甚麼內容。

以下程式碼是我們利用 switch1 控制 LCD 是否要顯示目前的模式:

(1)每個分別顯示當前的模式

```
}
if(Read_SW(1)==1){
switch(a)
      {
      case 1:
          LCD_PUT_CMD(LCD_FIRST_LINE);
          LCD_PUT_CHAR('N');
          LCD_PUT_CHAR('o');
          LCD_PUT_CHAR('r');
          LCD_PUT_CHAR('m');
          LCD_PUT_CHAR('a');
          LCD_PUT_CHAR('l');
          LCD_PUT_CHAR(' ');
          LCD_PUT_CMD(LCD_SECOND_LINE);
                          LCD_PUT_CHAR('O');
                          LCD_PUT_CHAR('u');
                          LCD_PUT_CHAR('t');
                          LCD_PUT_CHAR('p');
                          LCD_PUT_CHAR('u');
                          LCD_PUT_CHAR('t');
                  break;
      case 2:
          LCD_PUT_CMD(LCD_FIRST_LINE);
          LCD_PUT_CHAR('E');
          LCD_PUT_CHAR('c');
          LCD_PUT_CHAR('h');
          LCD_PUT_CHAR('o');
          LCD_PUT_CHAR(' ');
          LCD_PUT_CHAR(' ');
          LCD_PUT_CHAR(' ');
                                  break;
      case 3:
          LCD_PUT_CMD(LCD_FIRST_LINE);
          LCD_PUT_CHAR('f');
          LCD_PUT_CHAR(' ');
          LCD_PUT_CHAR('L');
           LCD_PUT_CHAR('o');
          LCD_PUT_CHAR('w');
          LCD_PUT_CHAR(' ');
                                  break;
      case 4:
          LCD_PUT_CMD(LCD_FIRST_LINE);
          LCD_PUT_CHAR('R');
          LCD_PUT_CHAR('i');
          LCD_PUT_CHAR('g');
          LCD_PUT_CHAR('h');
          LCD_PUT_CHAR('t');
          LCD_PUT_CHAR(' ');
          LCD_PUT_CMD(LCD_SECOND_LINE);
          LCD_PUT_CHAR('O');
          LCD_PUT_CHAR('u');
          LCD_PUT_CHAR('t');
          LCD_PUT_CHAR('p');
          LCD_PUT_CHAR('u');
          LCD_PUT_CHAR('t');
                                  break;
      }
}
```

(2)SWITCH 切斷則甚麼都不顯示

```
    }
    else{
        switch(a)
                {
                case 1:
                    LCD_PUT_CMD(LCD_FIRST_LINE);
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CMD(LCD_SECOND_LINE);
                                LCD_PUT_CHAR(' ');
                                LCD_PUT_CHAR(' ');
                                LCD_PUT_CHAR(' ');
                                LCD_PUT_CHAR(' ');
                                LCD_PUT_CHAR(' ');
                                LCD_PUT_CHAR(' ');
                            break;
                case 2:
                    LCD_PUT_CMD(LCD_FIRST_LINE);
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CMD(LCD_SECOND_LINE);
                                        LCD_PUT_CHAR(' ');
                                        LCD_PUT_CHAR(' ');
                                        LCD_PUT_CHAR(' ');
                                        LCD_PUT_CHAR(' ');
                                        LCD_PUT_CHAR(' ');
                                        LCD_PUT_CHAR(' ');
                                        break;
                case 3:
                    LCD_PUT_CMD(LCD_FIRST_LINE);
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CMD(LCD_SECOND_LINE);
                                        LCD_PUT_CHAR(' ');
                                        LCD_PUT_CHAR(' ');
                                        LCD_PUT_CHAR(' ');
                                        LCD_PUT_CHAR(' ');
                                        LCD_PUT_CHAR(' ');
                                        LCD_PUT_CHAR(' ');
                                        break;
                case 4:
                    LCD_PUT_CMD(LCD_FIRST_LINE);
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CMD(LCD_SECOND_LINE);
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                                        break;
                }
    }
}
```

接著我們會在 INT4 中對聲音的變化進行處理，有回音，降低聲音頻率以及切換為右聲道 3 種模式。

回音(利用一個緩衝區來達成):

```
delayed = buffer[i];
left_output = sample + delayed;
output_left_sample(left_output);
buffer[i] = sample + delayed * GAIN;

if(++i >= BUF_SIZE) i=0;
output_left_sample(left_output);
```

降低頻:

```
sample=10000*(sin(2.0*PI*DTMF_count*row*b/SAMPLING_FREQ));
DTMF_count++;
```

在這裡我們透過修改 b(程式中設置 0.7)的數值來達成降低頻率的效果。

右聲道:

```
output_right_sample(sample);
```

最後，由於我們有使用中斷，會在程式中加入中斷的程式碼，PB1,2,3,4 分別對應到的中斷為 INT5,6,7,8，以下是我們的程式碼

```
interrupt void INT5_ISR(void){
    a=1;
    b=1;
    return;
}
interrupt void INT6_ISR(void){
    a=2;
    b=1;
    return;
}
interrupt void INT7_ISR(void){
    a=3;
    b=0.7;
    return;
}
interrupt void INT8_ISR(void){
    a=4;
    b=1;
    return;
}
```

3. 結果與討論

在這個實驗的程式設計中，當使用者按下 keypad 上預設的九個按鍵之一時，將產生不同頻率的聲音，功能類似於電子琴。此外，程式還設定了四種不同的輸出模式，這些模式可以通過按下不同的按鍵（PB1、PB2、PB3、PB4）來切換：

1. PB1 - Normal Output: 按下 PB1 時，LCD 顯示“Normal Output”，並且電子琴將正常播放聲音，不會對音訊有任何調整。
2. PB2 - Echo Output: 切換至“Echo Output”模式後，音訊會有回音效果。這是通過使用程式中的一個環形緩衝區來實現的，而該緩衝區儲存並混合先前的聲音樣本。
3. PB3 - f Low Output: 啟動降低頻率模式後，LCD 顯示“f Low Output”。在此模式下，輸出的聲音頻率會降低(0.7)，從而產生不同的音色。

4. PB4 - Right Output: 按下 PB4 後，進入“Right Output”模式。在這個模式下，聲音僅從右聲道輸出，而左聲道則改為靜音。

我們還設計了 SWITCH1 用於控制 LCD 的顯示內容。根據目前被按下的按鍵，LCD 會顯示當前的輸出模式，切斷時則關閉，使得使用者可以容易地識別目前所處的音效模式。

4. 結論、心得與未來展望

這次實驗專題，讓我深刻體會到軟硬體結合的強大潛力和創意實現的無限可能。在此實驗中，我們利用 DTMF 信號技術和 easyDSP-Expansion Board，成功地將按鍵操作轉化為多樣化的音頻輸出，這不僅加深了我對音頻處理技術的理解，也使我更加了解之前所學可以如何實際應用。

特別有趣的是，通過簡單的按鍵（PB1、PB2、PB3、PB4）操作，我們能夠實現多種不同的聲音輸出模式。從標準的「Normal Output」到帶有回音的「Echo Output」，再到特殊效果的「f Low Output」和「Right Output」，每一種模式都鮮明地展示了音頻處理技術的魅力。這不僅增進了我對數位信號處理的實際應用知識，也讓對於用軟體控制硬體輸出感到十分感興趣。

此外，這次實驗還強調了使用者介面的重要性。透過 LCD 的即時反饋，使用者可以輕鬆識別當前所選擇的音效模式，這種直觀的設計對於操作的便利性至關重要。

這次實驗不僅讓我學到了音頻處理的技術原理，還提供了實際操作和應用的機會。這是一次寶貴的學習經驗，對我的學術和職業發展都有著重要的意義。

5. 參考資料

https://lms3.ntpu.edu.tw/pluginfile.php/168361/mod_resource/content/1/11.%E6%95%B8%E4%BD%8DIO%E6%8E%A7%E5%88%B6%E8%88%87%E5%A4%96%E9%83%A8%E4%B8%AD%E6%96%B7%E6%93%8D%E4%BD%9C%E7%B7%B4%E7%BF%92.pdf

https://lms3.ntpu.edu.tw/pluginfile.php/168331/mod_resource/content/1/10.%E6%95%B8%E4%BD%8DIO%E6%8E%A7%E5%88%B6%E8%88%87LCD%E9%A1%AF%E7%A4%BA%E5%99%A8%E6%93%8D%E4%BD%9C%E7%B7%B4%E7%BF%92.pdf

https://lms3.ntpu.edu.tw/pluginfile.php/166266/mod_resource/content/1/8.%E6%95%B8%E4%BD%8DIO%E6%8E%A7%E5%88%B6%E8%88%87SW%E6%93%8D%E4%BD%9C%E7%B7%B4%E7%BF%92.pdf

https://lms3.ntpu.edu.tw/pluginfile.php/172213/mod_resource/content/1/18.%E9%9F%B3%E6%BA%90%E5%9B%9E%E9%9F%B3%28echo%29%E7%9A%84%E8%A3%BD%E4%BD%9C.pdf

附錄

1. 作品影片連結: https://youtu.be/rseT9PQvgxQ
2. 程式碼:

```
#include "easyDSP-Expansion_Board.h"
#include "math.h"

#define SAMPLING_FREQ 12000
#define PI 3.14159265358979
#define loop_time 8000/2
#define GAIN 0.6
#define BUF_SIZE 3000
unsigned short row=0,col=0,sample=0;
unsigned short digital=15,old_digital=15;
short DTMF_count=0,DTMF_flag=0;
short buffer[BUF_SIZE];
short left_output,delayed;
int i= 0;

int a=1;
float b=1;
```

```
long count=0;

int main(void)
    {
   Board_Init(); //Initial easyDSP-
Expansion_Board

Setup_Audio_Init(FS_12000_HZ,ADC_GAIN
_0DB,DAC_ATTEN_0DB,LINE_INPUT);

   while(1)
   {
        digital = Read_keypad();

        switch(digital)
                {
                case 1:
                  row=0;
                  break;
                case 2:
                  row=524.0;

                  break;
                case 3:
                  row=392.0;

                  break;
                case 4:
                  row=0;

                  break;
                case 5:
                  row=588.0;

                  break;
                case 6:
                  row=440.0;

                  break;
                case 7:
                  row=0;

                  break;
                case 8:
                  row=0;

                  break;
                case 9:
                  row=0;
                  break;
                case 0:
                  row=0;

                  break;
                case 10: //-->A
                  row=494.0;

                  break;
                case 11:
                  row=349.0;

                  break;
                case 12:
                  row=0;

                  break;
                case 13:

                  row=330.0;


                  break;
                case 14:
                  row=294.0;
                  break;
                case 15:

                  row=262.0;

                  break;
                }
            if(DTMF_flag==1)
            {
                if(old_digital != digital)
                {
                DTMF_count = 0;
                DTMF_flag = 0;
                }
            }
            if(Read_SW(1)==1){
            switch(a)
                    {
                    case 1:

      LCD_PUT_CMD(LCD_FIRST_LINE);
                            LCD_PUT_CHAR('N');
                            LCD_PUT_CHAR('o');
```

```
                    LCD_PUT_CHAR('r');
                    LCD_PUT_CHAR('m');
                    LCD_PUT_CHAR('a');
                    LCD_PUT_CHAR('l');
                    LCD_PUT_CHAR(' ');

LCD_PUT_CMD(LCD_SECOND_LINE);

LCD_PUT_CHAR('O');

LCD_PUT_CHAR('u');

LCD_PUT_CHAR('t');

LCD_PUT_CHAR('p');

LCD_PUT_CHAR('u');

LCD_PUT_CHAR('t');
                                break;
            case 2:

LCD_PUT_CMD(LCD_FIRST_LINE);
                    LCD_PUT_CHAR('E');
                    LCD_PUT_CHAR('c');
                    LCD_PUT_CHAR('h');
                    LCD_PUT_CHAR('o');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR(' ');

break;
            case 3:

LCD_PUT_CMD(LCD_FIRST_LINE);
                    LCD_PUT_CHAR('f');
                    LCD_PUT_CHAR(' ');
                    LCD_PUT_CHAR('L');
                     LCD_PUT_CHAR('o');
                    LCD_PUT_CHAR('w');
                    LCD_PUT_CHAR(' ');

break;
            case 4:

LCD_PUT_CMD(LCD_FIRST_LINE);
                    LCD_PUT_CHAR('R');
                    LCD_PUT_CHAR('i');
                    LCD_PUT_CHAR('g');
                    LCD_PUT_CHAR('h');
                    LCD_PUT_CHAR('t');
                    LCD_PUT_CHAR(' ');

LCD_PUT_CMD(LCD_SECOND_LINE);
                    LCD_PUT_CHAR('O');
                    LCD_PUT_CHAR('u');
                    LCD_PUT_CHAR('t');
                    LCD_PUT_CHAR('p');
                    LCD_PUT_CHAR('u');
                    LCD_PUT_CHAR('t');

break;
                }
        }
        else{
            switch(a)
                                {
                                case 1:

LCD_PUT_CMD(LCD_FIRST_LINE);

LCD_PUT_CHAR(' ');

LCD_PUT_CHAR(' ');

LCD_PUT_CHAR(' ');

LCD_PUT_CHAR(' ');

LCD_PUT_CHAR(' ');

LCD_PUT_CHAR(' ');

LCD_PUT_CHAR(' ');

LCD_PUT_CMD(LCD_SECOND_LINE);

LCD_PUT_CHAR(' ');

LCD_PUT_CHAR(' ');

LCD_PUT_CHAR(' ');

LCD_PUT_CHAR(' ');

LCD_PUT_CHAR(' ');

LCD_PUT_CHAR(' ');
```

```
break;
                    case 2:
LCD_PUT_CMD(LCD_FIRST_LINE);
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CMD(LCD_SECOND_LINE);
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
break;
                    case 3:
LCD_PUT_CMD(LCD_FIRST_LINE);
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CMD(LCD_SECOND_LINE);
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
break;
                    case 4:
LCD_PUT_CMD(LCD_FIRST_LINE);
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CMD(LCD_SECOND_LINE);
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
LCD_PUT_CHAR(' ');
break;
```

```
                                }
        }

   }
}
//------------------------------
// interrupt service routine #4
//------------------------------
interrupt void INT4_ISR(void)
{
      if(DTMF_count < loop_time)
      {

sample=10000*(sin(2.0*PI*DTMF_count*ro
w*b/SAMPLING_FREQ));
          DTMF_count++;
      }
      else
      {
            sample = 0;
            DTMF_flag=1;
            old_digital = digital;
            DTMF_count = loop_time;
      }

      output_left_sample(sample);
      output_right_sample(sample);
      if(a==1){

             output_left_sample(sample);
                }
      else if(a==2){
               delayed = buffer[i];
               left_output = sample + delayed;
               output_left_sample(left_output);
               buffer[i] = sample + delayed *
GAIN;

               if(++i >= BUF_SIZE) i=0;
               output_left_sample(left_output);

                }
      else if(a==3){
            output_left_sample(sample);
      }
      else if(a==4){
            output_right_sample(sample);
      }
```

```
      return;
}
interrupt void INT5_ISR(void){
      a=1;
      b=1;
      return;
}
interrupt void INT6_ISR(void){
      a=2;
      b=1;
      return;
}
interrupt void INT7_ISR(void){
      a=3;
      b=0.7;
      return;
}
interrupt void INT8_ISR(void){
      a=4;
      b=1;
      return;
}
```