VE373 Project Report:

Electronic Air Band

Prepared by

Tao Zui 5123709084

Zhang Heng 5123709228

Li Bo 5123709038

Prepared for

Dr. Gang Zheng

VE373 Lecturer, SJTU-UM Joint Institute

Table of Contents

1 Introduction 1
2 Design Overview 1
3 Air drum block 2
3.1 Tom Drums and cymbal 2
3.1.1 Acceleration sensors 2
3.1.2 Control button 3
3.1.3 LED indicator 4
3.2 Hi-Hat and Bass Drum 4
3.2.1 Light sensor 4
3.2.2 LED indicator 5
3.3 UART communication 5
3.3.1 TTL/USB converter module 5
3.3.2 UART communication 5
4 Air piano block 6
4.1 Ultrasonic block 6
4.2 Big structure 8
4.3 Error discussion 9
4.4 Possible improvement 9
5 Matlab 9
6 Test Plan and Results 10
7 Weakness 10
7.1 The unstable light sensor input 10
7.2 The input capture interrupts failure 11
7.3 The range of the ultrasonic sensor 11
7.4 The sample time of the ADC 11
7.5 The delay of the sound generation 11
7.6 The project need two PIC32 boards 11

8 Appendix 12
8.1 Matlab 12
8.2 Air Drum 13
8.3 Air piano 19

1 Introduction

Playing instruments is a very good way to relax ourselves and nowadays more and more people start to learn the musical instruments and play during their free time. However, some instruments, including piano and drum, are very inconvenient to carry and also very expensive. Therefore, air instrument is becoming more and more popular these days. It is a kind of electronic devices that people just need to mimic the playing gestures in the air and the device will play the sound just like the real instrument. To meet the needs that music lovers can play musical instruments everywhere they like, our team designed the electronic air band.

The entire system is based on two PIC32 microcontrollers, one is for the air piano block and the other is for the air drum block. When the users weave the jazz drum sticks, the sensors on the stick will detect the motion and send the data to the PIC32 board. Similarly, in the air piano block the sensor will detect the position your hand so that it will know which key you are playing. After processing the data, the PIC32 board will send the data to the computer through the serial communication. Then the matlab will play the desired sound that is prepared in advance. The air band can provide mostly six kinds of drum sound, including one cymbal, one tom, two snare drums and two bass drums. Also, it can provide up to 16 kinds of piano notes.

The structure of the report is as follows. Section 2 will give a high level description of the project. Section 3,4 and 5 will give a detailed description of each functional component of our design. Section 6 introduces the tests and the evaluation result. Finally we will discuss the pitfalls of our project and draw the conclusion.

2 Design Overview

The figure below shows the functional block diagram of our design. (Figure 2.1)

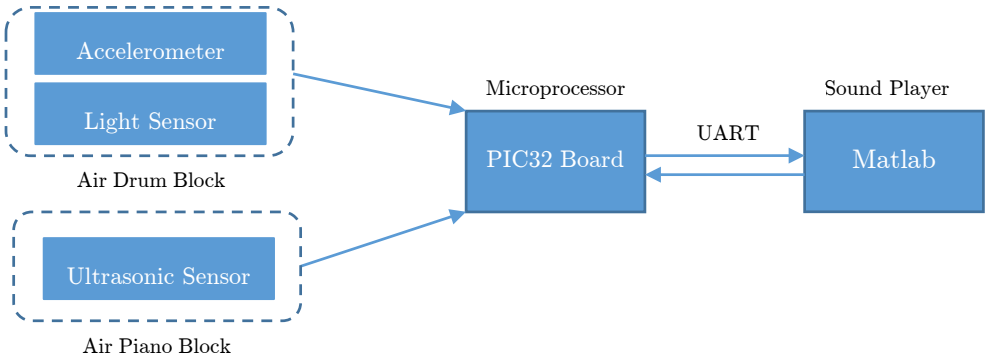

Figure 2.1: Component Level Block Diagram

As shown in the diagram, we will use the accelerometer to measure the movement of the hands. In addition, two buttons are attached on the stick so that when you press the button, the computer will give different sound. Then we will use the light sensor to detect the feet movement. When you tap your feet, the computer will play the sound of the bass drum. As for the air piano block, the ultrasonic sensor will measure the distance between the finger and the sensor itself. All the values will be transferred to the PIC32 Board. Then it will communicate with the computer through UART and DMA. Using matlab, the computer can recognize the signal and play the right sound. The detailed designs and functionalities of each component block are shown in the next sections.

3 Air drum block

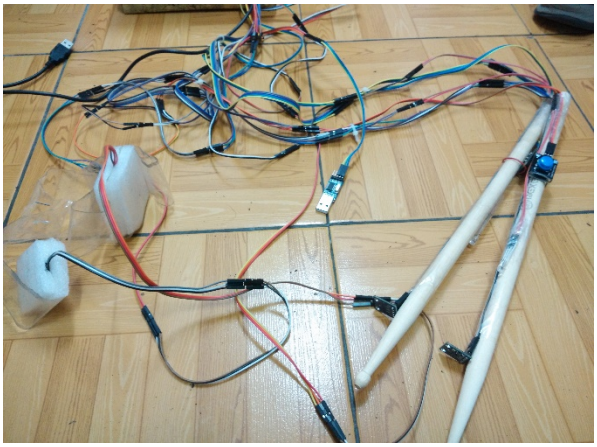

Figure 3.1: Air drum Overview

The air drum block includes three major parts, the first part is three Tom-Tom Drums(hand drums) and one cymbal, the second part is Hi-Hat and Bass Drum (foot drums), the last part is the UART communication. There are two acceleration sensors working as three Tom-Tom Drums(hand drums) and one cymbal. Two light sensors are working as Hi-Hat and Bass Drum (foot drums). And one USB to TTL module working for UART communication

3.1 Tom Drums and cymbal

3.1.1 Acceleration sensors

The acceleration sensor we used is the GY-61 ADXL335 module. They are placed on the drumsticks.

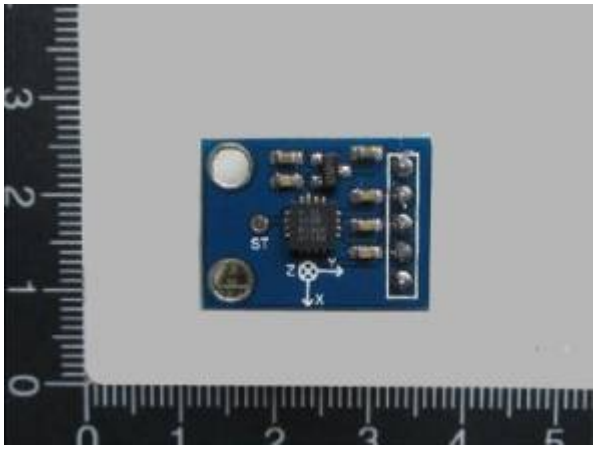

Figure 3.2: GY-61 ADXL335 module

It can detect the angle position in X,Y,Z three directions. We setup a certain sample time an use ADC scan mode to detect the difference between the angles in X,Y,Z. When the difference reaches the critical value, the program will generate a signal to indicate that the drum is hit. Also the magnitude of the difference will be recorded and send through UART to allow matlab to generate sound with different volume. Since we have six analog inputs, so we use the ADC scan mode two make sure that every hit on the drum will be detected. The ADC setup is as followed.

```
AD1PCFG = 0xFFC0;
AD1CON1 = 0x00E4;
AD1CON2 = 0x0414;
AD1CON3 = 0x0000;
AD1CSSL = 0x003F;
```

Figure 3.3: ADC scan mode setup

3.1.2 Control button

The control buttons are also placed on the drumsticks.

Figure 3.4: Control button

It is used to control which drum the drum stick is hitting. For the left hand, the control button determines if you are hitting a higher pitch drum or a lower pitch drum. For the right hand, the control button determines if you

are hitting a high pitch drum or a cymbal.

3.1.3 LED indicator

There are two led indicators on each drumstick to indicate that if the drum is hit or not.

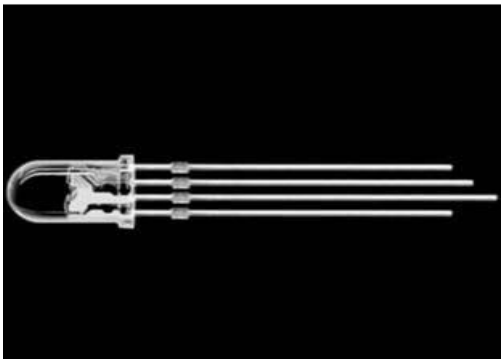

Figure 3.5: The RGB LED

For the left hand it is basically a digital output controlled LED. But for the right hand, the LED is controlled by a PWM output. So the light intensity will be strong when the drum is hit hardly. And the light intensity will be weak when the drum is hit weakly. The PWM setup is as followed.

```
if(d2>0x10000){d2=0x10000;}
PR3=100;
OC2CON=0x800E;
OC2RS = 100*(0x10000-d2)/(0x10000);
OC2R = 100*(0x10000-d2)/(0x10000);
TMR3=0;
//PORTDbits.RD1=1;
T3CON=0x8010;
```

Figure 3.6: The PWM setup

3.2 Hi-Hat and Bass Drum

3.2.1 Light sensor

There are two light sensors on two feet to control the Hi-Hat and Bass Drum.

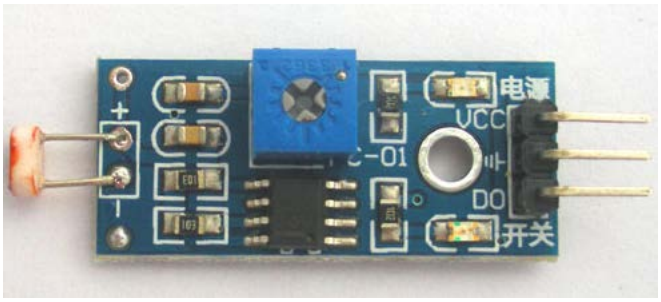

Figure 3.7: Light sensor

The left foot is Hi-Hat the right feet is Bass Drum. It is placed under the feet. I can detect the change of the light intensity so that when the drum is hit, the change on the light intensity will be detected be the light sensor. We use the Change Notice interrupt to capture the change on the sensors' output pins.

```
CNCON = 0x8000;//Enable Change Notice module
CNEN = 0x00000003;//Enable individual CN pins
CNPUE = 0x00000003;//
```

Figure 3.8: Change Notice setup

3.2.2 LED indicator

There are also two led indicator on the breadboard to indicate if on drum is hit or not.

3.3 UART communication

3.3.1 TTL/USB converter module

We use the TTL/USB converter module to facilitate the communication between computer and the PIC32 board.

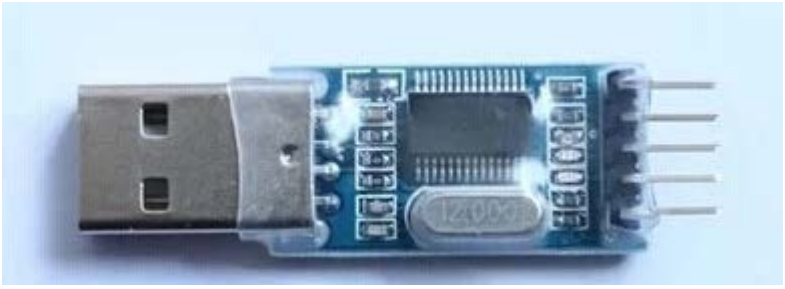

Figure 3.9 TTL/USB converter module

3.3.2 UART communication

We use the UART to communicate between the computer and PIC32. We used UART1A, with 8bits data,

1 even parity bits and 2 stop bits mode to ensure the communication is stable and error free.

```
void UARTconfig(){
    U1ABRG=65;
    U1AMODEbits.STSEL=1;
    U1AMODEbits.PDSEL=1;
    U1AMODEbits.BRGH=0;
    U1AMODEbits.UEN=0;
    U1AMODEbits.ON=1;
    U1ASTAbits.UTXEN=1;
    TRISF=0x0000;
}
```

Figure 3.10: UART setup

Also to ensure the speed and stability, we used DMA to help UART communication, So that the UART communication will not take kernel's resources and time.

```
IEC1CLR=0x00010000;
IFS1CLR=0x00010000;
DMACONSET=0x00008000;
DCH0CON=0x3;
DCH0ECON=0;
DCH0SSA=KVA_TO_PA(&data);;
DCH0DSA=KVA_TO_PA(&U1ATXREG);
DCH0SSIZ=1;
DCH0DSIZ=1;
DCH0CSIZ=1;
DCH0INTCLR=0x00ff00ff;
DCH0CONSET=0x80;
```

```
if(PORTEbits.RE2==0)
{data=0xA0+0x1F*d2/(0x10000);
DCH0CONSET=0x80;
DCH0ECONSET=0x00000080;}
else{data=0xC0+0x1F*d2/(0x10000);
DCH0CONSET=0x80;
DCH0ECONSET=0x00000080;}
```

Figure 3.11: DMA setup and DMA usage

4 Air piano block

The air piano block includes two parts, one part is an ultrasonic distance detector and another one is an 8-key LED keypad.

4.1 Ultrasonic block

The ultrasonic sensor is HC-SR04, which is shown in the plot.

Figure 4.1: HC-SR04 ultrasonic sensor

Its functionality is illustrated with the following plot.

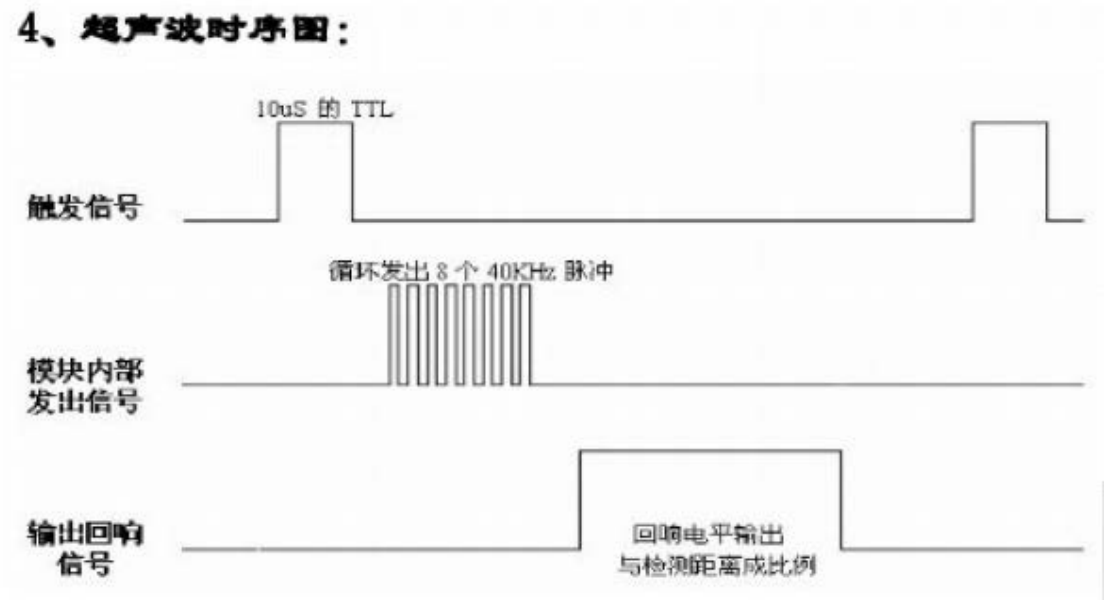

Figure 4.2: HC-SR04 function principle

When we need a distance, MCU will generate a pulse with minimum width 10us and output it to the trigger port of HC-SR04. Then ultrasonic sensor, when an echo response comes back, will transmit the received square wave back to MCU through echo port.

At the MCU part, we use input capture to capture the rising edge and falling edge.

```
void ICconfig(){
  INTCONbits.MVEC = 1;
  asm("di");

  IC3CONbits.ICTMR = 1; //
  IC3CONbits.ICI = 1; // interrupt one every 2nd capture
  IC3CONbits.ICM = 6; // capture every edge
  IC3CONbits.FEDGE = 1; // capture rising edge first

  IC2CONbits.ICTMR = 1; //
  IC2CONbits.ICI = 1; // interrupt one every 2nd capture
  IC2CONbits.ICM = 6; // capture every edge
  IC2CONbits.FEDGE = 1; // capture rising edge first

  IC3CONSET = 0x8000;
  IC2CONSET = 0x8000;


  IPC2SET = 0x1F00; // interrupt of IC2 and IC3
  IPC3SET = 0x1F00;
  IFS0CLR = 0x2200;
  IEC0SET = 0x2200;

  asm("ei");
}
```

Figure 4.3: Setup of IC

```
  temp1 = IC2BUF;
  temp2 = IC2BUF;
  record = distance2;
distance2 = ((temp2 - temp1)*V_SOUND * scale )/(PBCLK);
```

Figure 4.4: interrupt of IC

Two temporary variables, temp1 and temp2, are used to read out the buffer. Then the distance value is calculated using the equation of distance2. We use timer2 to be capture source.

4.2 Big structure

The full air piano prototype is shown here:

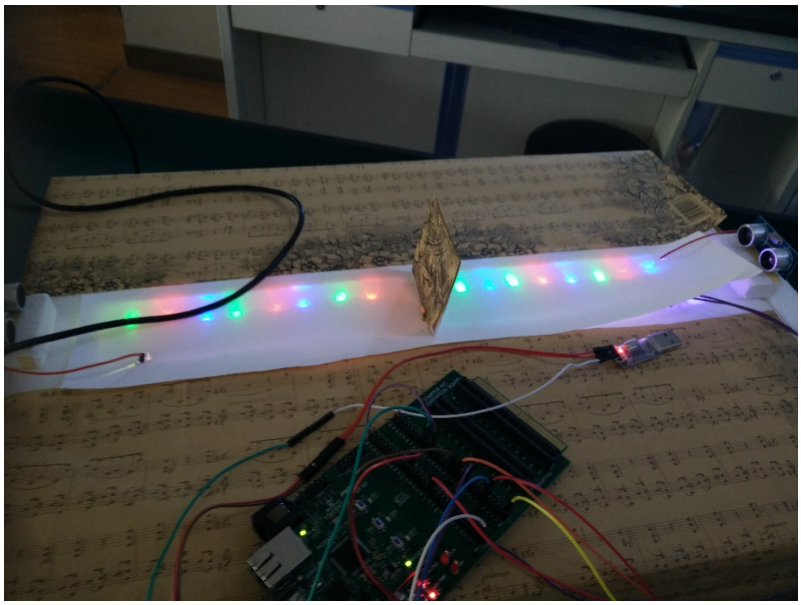

Figure 4.5: Air piano

LEDs are used to denote the location of different keys like a real piano. When some objects stands in front of one ultrasonic sensor, this sensor will detect it and respond with a distance calculated by the equation in Figure 4.4. In order to transfer the distance to corresponding sound, we use UART module of PIC32 to transmit data to computer and then compute will decode the data and play correct sound file.

```
if (predistance==24)
{
 if(5 <= distance3 && distance3 <= 6)
U1ATXREG =0b00100000; // do
 else if (distance3==8 )
U1ATXREG = 0b00100001;
 else if (distance3==10)
U1ATXREG = 0b00100010;
 else if (distance3==12)
U1ATXREG = 0b00100011;
 else if (distance3==15)
U1ATXREG = 0b00100100;
 else if (distance3==17)
U1ATXREG = 0b00100101;
 else if (distance3==19)
U1ATXREG = 0b00100110;
 else if (distance3==20)
U1ATXREG = 0b00100111;
}
```

Figure 4.6: Data transmission

Predistance is used as a reference because we want to avoid the scenario that when one keeps putting his/her finger in one position, there will be just one sound output which is how a real piano does. When one moves out finger, predistance will be the fixed value from ultrasonic sensor to that vertical board in the middle of the keypad as shown in Figure 4.2.1.

4.3 Error discussion

When we finish this air piano, the performance is limited. The major reason is that the accuracy of ultrasonic sensor is just 1cm which is far from enough for a piano. What happened is that when one put a finger in adjacent two keys, it possible that two different would be played.

4.4 Possible improvement

After a few discussion with TAs and other students, we guess that infrared light sensor might be a better choice because of better accuracy and less inferences.

5 Matlab

After the data processing in the PIC32 board, it will send the signal which is defined by the protocol to the computer. Here, matlab will be responsible for communication and playing the sound.

The serial function in the matlab is used for UART communication. The configuration in the matlab will be identical to the setup on the board otherwise it will only receive 0. Each time we will use the fread function to read one byte data from the input FIFO. Then depending on the value, we use sound function to play the proper sound. Since the board may send duplicate values due to the sensor, we use pause to generate the delay between each reading so that we can filter duplicate values. As for the drum sound since the value stands for the acceleration of your hand, so the code will modify the peak value of the sound frequency so that different volumes can be achieved.

6 Test Plan and Results

To test the functionality of our design, we first need to test separate parts. First of all is the model level testing including UART, Chang Notice, ADC and so on. Then we will test each sensors if their functionality is complete. Once all the sensors work properly, we assemble them to the air drum block as well as the air piano block. For the air drum block, we first make sure that the sensor will give the data to the PIC32 board at the right time, then we test for different movements to make sure that if we shake the sticks at different speed, the sensor will give the proper measured value. As for the air piano block, the most important test procedure is to adjust the positions of the ultrasonic sensor and the LED light so that the position of the finger can be accurately measured. After these, we test the functionality of the matlab code. Since these two are relatively separate blocks, after the test of the individual block is completed, the whole test procedure is done.

The test shows that matlab code works well. As for the embedded system part, the accelerometer can stably measure the data but the acceleration is not so accurate. The light sensor is very sensitive to the environment light so we have to adjust the light around it. As for the air piano block, it is less stable than the air drum. For a static object, the measurement is very accurate for the ultrasonic sensor, but since the finger is moving all the time when we play the piano, it can not always give the right data to the PIC32 board.

7 Weakness

7.1 The unstable light sensor input

The foot drum uses the light sensor as the input. The problem is that the light sensor is too sensitive to the changing of the light intensity in the environment.

The first problem is that the light sensor needed to be configured every time the device setup. We need to change sensitivity every time to make sure that the light sensor can detect the difference of the light intensity between the place that under our feet and environment.

The second problem is that the light sensor will be unstable if we use our foot drum in a very low speed. Since the input of the CN pin can generate several change notice and get into the interrupt several times. We write

a program in matlab to prevent generating duplicated sounds in a short period, but the problem in the hardware part still exist.

7.2 The input capture interrupts failure

The air piano is based on the ultrasonic sensor. The problem is that when we use the input capture interrupts to catch the feedback form the ultrasonic sensor. It always can work well for several times. But then the program counter jump to a unknown memory location which leads to suspend of the program. After trying several different IC pins and external interrupt pins, we have no choice but to poll the flag of the input capture interrupt. The problem is that the timing of polling flag is not accurate, which lead to inaccurate in distance measurement and leads to wrong sound generation.

7.3 The range of the ultrasonic sensor

The ultrasonic sensor has a using range, inside the range, the output will be accurate, but outside the range the error becomes very large, then we can't get the right distance. We use two ultrasonic sensors to solve the problem, but it is still not accurate when we reach the places which are correspond to the high pitch sound.

7.4 The sample time of the ADC

We are using ADC to convert the data we get from the acceleration sensor. But the problem is that if the ADC sample rate is too fast, the acceleration will be too small and then we can't distinguish if the drum is hit or not. Also during one hit the acceleration sensor may detect several times. But if we use a longer sample rate, we may not be able to detect several hit of the drum in a short period. Also, the ADC converting time will affect the performance of the drum.

7.5 The delay of the sound generation

The first problem is that since we use matlab to do the sound generation the delay time will be longer than using the pic32 to direct generate sound. Since the matlab needs to put all the sound data into the audio card and generate sound, it will take longer time. We try to use a sound files with smaller size to solve the problem. But the problem still exists.

The second problem is that the UART communication only allows 1byte at a time, So there sure will be an order of the UART communication. Then the problem is that when two or more drums are hit at the same time, the sound can't be generated simultaneously. There will always be a little bit difference between different sounds.

7.6 The project need two PIC32 boards

Since we need to determine a time period that we don't send a new signal through UART, we need timer for each single sensor. So we need 6 timers for 6 sensors. But the PIC32 only have 5 timers, it forced us to use two PIC32 board to make the project work.

8 Appendix

8.1 Matlab

```
% open the port, change COM4 to match your port namclce
mySerial = serial('COM4');
mySerial.baudrate=2400;
mySerial.timeout=1000;
mySerial.Databits=8;
record=zeros(1000,2);
i=1;
fopen(mySerial);
disp('Reading...');
t2=0;
t3=0;
while (1)
%pause(0.02);
tmp=fread(mySerial,1);
t1=cputime;
if (t1-t3>0.05)
    display(tmp);
switch tmp(1,1)
    case 32             %Piano Sound
    sound(A1,Fs1);
    case 33
    sound(B1,Fs1);
    case 34
    sound(C1,Fs1);
    case 35
    sound(C1,Fs1);
    case 36
    sound(D1,Fs1);
    case 37
    sound(E1,Fs1);
    case 38
    sound(F1,Fs1);
    case 39
    sound(G1,Fs1);
    case 40
    sound(A2,Fs1);
    case 41
    sound(B2,Fs1);
    case 42
    sound(C2,Fs1);
    case 43
    sound(D2,Fs1);
    case 44
    sound(E2,Fs1);
    case 45
    sound(F2,Fs1);
    case 46
    sound(G2,Fs1);
    case 47
    sound(A3,Fs1);
```

```
        case 48
        sound(B3,Fs1);
end
t3=cputime;
end
if (t1-t2>0.3)
if ((tmp>=64) && (tmp<96)) %Left Feet Drum
        sound(Bassdrum1/2,Fs2);t2=cputime;record(i,1)=tmp;record(i,2)=cputime;i=i+1;
end
if ((tmp>=96) && (tmp<128)) %Right Feet Drum
        sound(Bassdrum2*2,Fs2);t2=cputime;record(i,1)=tmp;record(i,2)=cputime;i=i+1;
end
end
if ((tmp>=128) && (tmp<150)) %Left Hand Drum
        sound(snaredrum1/2,Fs2);record(i,1)=tmp;record(i,2)=cputime;i=i+1;
else if ((tmp>=150) && (tmp<160))
        sound(snaredrum1*2,Fs2);record(i,1)=tmp;record(i,2)=cputime;i=i+1;
        end
end
if ((tmp>=160) && (tmp<180)) % Right Hand Drum
        sound(snaredrum2/2,Fs2);record(i,1)=tmp;record(i,2)=cputime;i=i+1;
else if ((tmp>=180) && (tmp<192))
        sound(snaredrum2*2,Fs2);record(i,1)=tmp;record(i,2)=cputime;i=i+1;
        end
end
if ((tmp>=192) && (tmp<212)) %Cymbal
        sound(cymbal1/2,Fs2);record(i,1)=tmp;record(i,2)=cputime;i=i+1;
else if ((tmp>=212) && (tmp <224))
        sound(cymbal1*2,Fs2);record(i,1)=tmp;record(i,2)=cputime;i=i+1;
        end
end
if ((tmp>=224) && (tmp<241)) %Tom
        sound(tom1/2,Fs2);record(i,1)=tmp;record(i,2)=cputime;i=i+1;
else if ((tmp>=241) && (tmp <256))
        sound(tom1*2,Fs2);record(i,1)=tmp;record(i,2)=cputime;i=i+1;
        end
end
tmp=0
end
% close the port
fclose(mySerial);
disp('Complete');
close all;
```

8.2 Air Drum

```
 #include <plib.h>
#include <p32xxxx.h>

#pragma interrupt T3_ISR ipl7 vector 12
#pragma interrupt T4_ISR ipl7 vector 16
#pragma interrupt T5_ISR ipl7 vector 20
#pragma interrupt CN_ISR ipl7 vector 26
#pragma interrupt ADC_ISR ipl7 vector 27
#pragma interrupt ISR_INT1 ipl7 vector 7
#pragma interrupt IC4_ISR ipl7 vector 17
```

```
int x11=300;
int x12=300;
int x21=300;
int x22=300;
int y11=300;
int y12=300;
int y21=300;
int y22=300;
int z11=300;
int z12=300;
int z21=300;
int z22=300;
int d11=0;
int d12=0;
int d13=0;
int d21=0;
int d22=0;
int d23=0;
int count=0;
int last1=0;
int last2=0;
int dang=0;
int i=0;
int re4=1;
char data=0;
char trans=0;

void IC4_ISR()
{
        int ggg=IC4BUF;
        while(U1ASTAbits.UTXBF);
        U1ATXREG=0x60;
        PORTDbits.RD4=0;
        PR5=40000;
        T5CON=0x8030;
        TMR5=0;
        IFS0CLR=0x20000;
}



void ISR_INT1()
{       dang++;
        while(U1ASTAbits.UTXBF);
        U1ATXREG=0x40;
        PORTDbits.RD3=0;

        PR5=40000;
        T5CON=0x8000;
        TMR5=0;
    IFS0CLR=0X80;
}
void CN_ISR (void)
{
    IEC1CLR = 0x0001;
```

```
if(PR5==50000)
{

	if(PORTCbits.RC14==1)
	{

		dang++;
		while(U1ASTAbits.UTXBF);
		data=0x40;
		DCH0CONSET=0x80;
		DCH0ECONSET=0x00000080;
		PORTDbits.RD3=0;

		PR5=40000;
		T5CON=0x8030;
		TMR5=0;
		CNCON = 0x0000;
	}
	else if(PORTCbits.RC13==1)
	{

		while(U1ASTAbits.UTXBF);
		if (PORTCbits.RC13==1)
		data=0x60;
		DCH0CONSET=0x80;
		DCH0ECONSET=0x00000080;
		PORTDbits.RD4=0;
		PR5=40000;
		T5CON=0x8030;
		TMR5=0;
		CNCON = 0x0000;
	}
}
	IFS1CLR = 0x0001;//clear interrupt flag
	IEC1SET = 0x0001;//

}



void T3_ISR()
{
	IFS0CLR = 0x00001000;
	if(PR3==50000)
	{
	PORTDbits.RD1=1;
	T3CON=0;
	TMR3=0;}
	else if(PR3==100)
	{
		count++;
		if(count==5000)
		{
			count=0;
			OC2CON=0;
			T3CON=0;
```

```
                TMR3=0;
            }
        }
}


void T4_ISR()
{
        IFS0CLR = 0x00010000;
        PORTDbits.RD2=1;
        T4CON=0;
        TMR4=0;
}


void T5_ISR()
{
        IFS0CLR = 0x00100000;
        if(PR5==40000)
        {
            CNCON = 0x8000;
            TMR5=0;
            T5CON=0x0;
            PR5=50000;
            T5CON=0x8050;
        }
        else if(PR5==50000)
        {PORTDbits.RD3=1;
        PORTDbits.RD4=1;
        T5CON=0;
        TMR5=0;}
}

void ADC_ISR()
{
         //int d11=d12=d13=d21=d22=d23=0;
        int d1=0;
        int d2=0;

         if(T4CONbits.ON==0)
         {
            x11=ADC1BUF0;
            y11=ADC1BUF1;
            z11=ADC1BUF2;
            d11=x11-x12;
            d12=y11-y12;
            d13=z11-z12;
            d1=d11*d11+d12*d12+d13*d13;
         if ((d12>0)&&(d1>10000))
        {       TMR4=0;
            //dang++;
            PORTDbits.RD2=0;
            T4CON=0x8050;
            while(U1ASTAbits.UTXBF);
            if(d1>0x10000){d1=0x10000;}
            if(PORTEbits.RE1==0)
```

```
        {//U1ATXREG=0x80+0x1F*d1/(0x10000);
        data=0x80+0x1F*d1/(0x10000);
        DCH0CONSET=0x80;
        DCH0ECONSET=0x00000080;
        }
        else {//U1ATXREG=0xE0+0x1F*d1/(0x10000);
        data=0xE0+0x1F*d1/(0x10000);
        DCH0CONSET=0x80;
        DCH0ECONSET=0x00000080;
        }
       //i==0x80+(0x1F*d1/0x10000);

}
else{
TMR4=0;
PR4=50000;
T4CON=0x8010;
      }
  }
if(T3CONbits.ON==0)
  {
        x21=ADC1BUF4;
        y21=ADC1BUF5;
        z21=ADC1BUF6;
        d21=x21-x22;
        d22=y21-y22;
        d23=z21-z22;
       d2=d21*d21+d22*d22+d23*d23;
      if ((d22>0)&&(d2>7000))
      {
            if(d2>0x10000){d2=0x10000;}
            PR3=100;
            OC2CON=0x800E;
            OC2RS = 100*(0x10000-d2)/(0x10000);
            OC2R = 100*(0x10000-d2)/(0x10000);
            TMR3=0;
            //PORTDbits.RD1=1;
            T3CON=0x8010;
            while(U1ASTAbits.UTXBF);

            if(PORTEbits.RE2==0)
            {data=0xA0+0x1F*d2/(0x10000);
            DCH0CONSET=0x80;
            DCH0ECONSET=0x00000080;}
            else{data=0xC0+0x1F*d2/(0x10000);
            DCH0CONSET=0x80;
            DCH0ECONSET=0x00000080;}
      }
      else{
      TMR3=0;
      PR3=50000;
      T3CON=0x8000;
      }
  }
  x12=x11;
  y12=y11;
```

```
        z12=z11;
        x22=x21;
        y22=y21;
        z22=z21;


}

void UARTconfig(){
      U1ABRG=65;
      U1AMODEbits.STSEL=1;
      U1AMODEbits.PDSEL=1;
      U1AMODEbits.BRGH=0;
      U1AMODEbits.UEN=0;
      U1AMODEbits.ON=1;
      U1ASTAbits.UTXEN=1;
      TRISF=0x0000;
      //TRISD=0x0000;
      //TRISB=0x0000;
}
int main()
{
            INTCONbits.MVEC = 1; // Enable multiple vector interrupt
            // Enable all interrupts
            asm ("di");

            UARTconfig();

            PR3=50000;
            PR4=50000;
            PR5=50000;
            TMR4=0;
            TMR5=0;
            TMR3=0;

            IPC3SET = 0x0000001C; //IPC4SEet priority level = 7
            IFS0CLR = 0x00001000;
            IEC0SET = 0x00001000;

            IPC4SET = 0x0000001C; //Set priority level = 7
            IFS0CLR = 0x00010000;
            IEC0SET = 0x00010000;

            IPC5SET = 0x0000001C; //Set priority level = 7
            IFS0CLR = 0x00100000;
            IEC0SET = 0x00100000;

            IPC6SET = 0x1C000000;
            IFS1CLR = 0x00000002;
            IEC1SET = 0x00000002;

            IEC1CLR=0x00010000; // disable DMA channel 0 interrupts
            IFS1CLR=0x00010000; // clear existing DMA channel 0 interrupt flag
            DMACONSET=0x00008000; // enable the DMA controller
            DCH0CON=0x3; // channel 0 disabled, priority 3, no chaining
            DCH0ECON=0; // no start or stop IRQ, no pattern match
```

```
        DCH0SSA=KVA_TO_PA(&data);; // transfer source physical address
        DCH0DSA=KVA_TO_PA(&U1ATXREG); // transfer destination physical address
        DCH0SSIZ=1; // source size 101 bytes
        DCH0DSIZ=1; // destination size 1 bytes
        DCH0CSIZ=1; // 1 bytes transferred per event
        DCH0INTCLR=0x00ff00ff; // clear existing events, disable all interrupts
        DCH0CONSET=0x80; // turn channel on

        TRISC = 0xFFFF;
        TRISD = 0xFF00;
        TRISE = 0xFFFF;
        PORTDbits.RD4=1;
        PORTDbits.RD3=1;
        PORTDbits.RD2=1;
        PORTDbits.RD1=1;

        AD1PCFG = 0xFFC0;
        AD1CON1 = 0x00E4;
        AD1CON2 = 0x0414;
        AD1CON3 = 0x0000;
        AD1CSSL = 0x003F;


        CNCON = 0x8000;//Enable Change Notice module
        CNEN = 0x00000003;//Enable individual CN pins
        CNPUE = 0x00000003;//

        IPC6SET = 0x001C0000;//Set priority level = 7
        IFS1CLR = 0x0001;//Clear interrupt flag
        IEC1SET = 0x0001;//


        asm ("ei");

        AD1CON1SET = 0x8000;

    while(1)
{
if(PORTEbits.RE4==1){re4=1;}

int k=0;

if((PORTEbits.RE4==0)&&(re4==1))
{
while(U1ASTAbits.UTXBF);
        U1ATXREG=0x60;
        PORTDbits.RD4=0;
        PR5=40000;
        T5CON=0x8030;
        TMR5=0;
        re4=0;
}
}
}
```

8.3 Air piano

```
#include <p32xxxx.h>
```

```
#include <plib.h>
#define V_SOUND 17000 // 17000 cm/s
#define PBCLK      2500000 //2.5MHz
#define scale      256
//#pragma interrupt IC5_ISR ipl7 vector 21
//#pragma interrupt IC2_ISR ipl6 vector 9
#pragma interrupt TMR2_ISR ipl7 vector 8
int distance3 = 0;
int temp1 = 0;
int temp2 = 0;
int reduancy = 0;
int status3=0;
int predistance = 0;

void TMRconfig()
{
   T2CON = 0x70;// 1:256
   PR2 = 0xffff;
   T2CONSET = 0x8000;
   TMR2 = 0;
   IEC0SET = 0x100;
   IFS0CLR = 0x100;
   IPC2SET = 0x1F;
}

void DelayUsec(int num) {
   IFS0CLR = 0x10;
   TMR1=0;
   PR1=num; // base time is 3.2 us
   T1CON = 0x8010; // TMR2 on, prescale 1:8 = 0.3125 MHz
   while(IFS0bits.T1IF==0){
   }
   IFS0CLR = 0x10;
   T1CONCLR = 0x8000; // TMR2 off
}

void get_distance()
{
      PORTD = 0xc0;                    //TRIGGER HIGH RD6, RD 7 high
      DelayUsec(10);                   //10uS Delay
      PORTDCLR = 0xc0;                       //TRIGGER LOW RD0, RD 1
}
void TMR2_ISR()
{
       IFS0CLR=0x100;
       get_distance();
}


void ICconfig(){
   INTCONbits.MVEC = 1;
   IC5CONbits.ICTMR = 1; //
   IC5CONbits.ICM = 1; // capture every edge
   IC5CONbits.SIDL=1;
   IC5CONbits.FEDGE = 1; // capture rising edge first
   IC5CONSET = 0x8000;
```

```
    IFS0CLR = 0x200000;
}
void UARTconfig(){
    U1ABRG=65;
    U1AMODEbits.STSEL=1;
    U1AMODEbits.PDSEL=1;
    U1AMODEbits.BRGH=0;
    U1AMODEbits.UEN=0;
    U1AMODEbits.ON=1;
    U1ASTAbits.UTXEN=1;
    TRISF=0x0000;
}

main(){
    TRISD = 0x1c00; //RD9 as Input PIN (ECHO) 2 RD10 as Input PIN (ECHO) 1
    ICconfig();
    PORTD = 0 ;
    UARTconfig();
    TMRconfig();
    get_distance();
while(1)
{
        while(IFS0bits.IC5IF == 0){}
        if(status3 == 1)
        {
                temp2 = IC5BUF;
                distance3 = ((temp2 - temp1)* V_SOUND * scale )/(PBCLK);
                status3 = 0;
                if(predistance == 22){
                        if(5 <= distance3 && distance3 <= 6)
                                U1ATXREG =0b00100000; // do
                          else if (distance3==8 )
                                U1ATXREG = 0b00100001;
                          else if (distance3==10)
                                U1ATXREG = 0b00100010;
                          else if (distance3 == 12)
                                U1ATXREG = 0b00100011;
                          else if (distance3==15)
                                U1ATXREG = 0b00100100;
                          else if (distance3==17)
                                U1ATXREG = 0b00100101;
                          else if (distance3==19)
                                U1ATXREG = 0b00100110;
                          else if (distance3==20)
                                U1ATXREG = 0b00100111;
                }
                predistance = distance3;
        //      DelayUsec(65535);
                get_distance();
                TMR2 = 0;
        }
        else if(status3 == 0)
        {
             temp1 = IC5BUF;
              status3 = 1;
        }
```

```
while(IC5CONbits.ICBNE){
      reduancy = IC5BUF;
}
IFS0bits.IC5IF= 0;}}
```