简易电子琴的设计和实现

一、功能设计

电子琴主要功能：

1、弹奏功能——能够发出从降1到升7共21个音符的音。

2、播放功能——能够通过控制外接无源蜂鸣器来演奏歌曲，目前已录入2首歌曲。（清华校歌和偏爱）

功能操作手册：

主循环：指示灯L3、L4、L5、L6全亮，表示开始选择功能——此时按下K1表示选择弹奏功能；按下K2表示选择播放功能。

弹奏功能：1、进入弹奏功能时L5、L6灭；

2、按下K1—K7可以发出音符1-7的音调；

3、按下K8键可以切换音域；

4、L3和L4用以表示音域：L3亮、L4灭——低音

L3灭、L4亮——中音

L3亮、L4亮——高音

5、弹奏过程中同时按下K2和K8可以退出弹奏功能，返回最初的功能选择界面，4灯全亮。

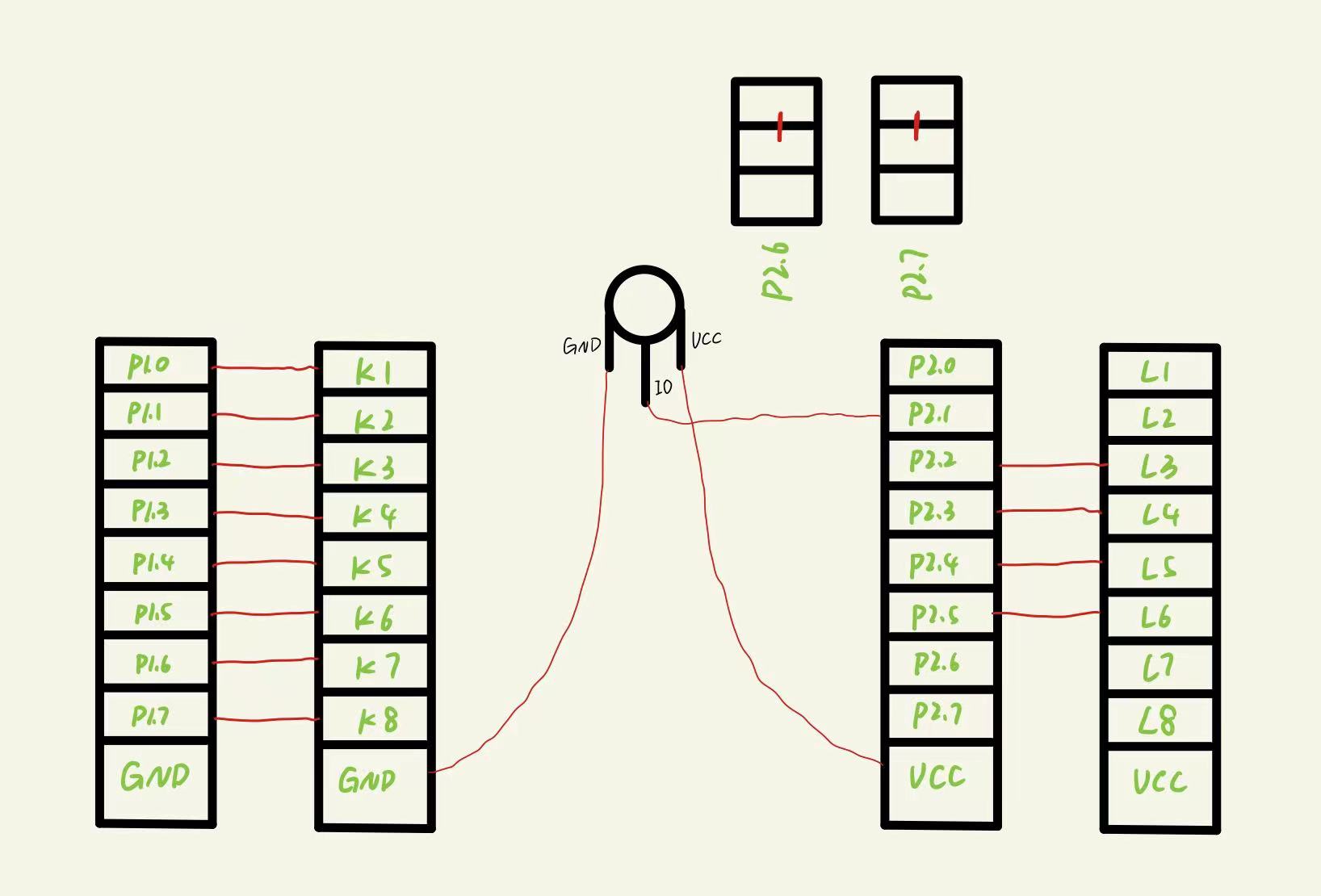
播放功能：1、进入播放功能以后L3、L4灭，L5、L6亮，表示可以选择曲目；

2、按下K1表示选择曲目1，此时仅L5亮，开始播放曲目1；按下K2表示选择曲目2，此时仅L6亮，开始播放曲目2；

3、曲目播放完毕自动回到最初的功能选择界面，4个灯全亮。

4、曲目播放的过程中可以通过按K8终止播放，返回最初的功能选择界面，4灯全亮。

二、电子琴硬件构成示意图



三、已实现功能

目前两个功能均已实现。

四、困难与解决方法

1、STM430板的RAM过小，如何将一整首曲子储存下来是一个问题；

最后利用关键词const，将曲子转化为数组存储在ROM里，解决了存谱子的问题。

2、STM430引脚太少，将P1.0到P1.6全拿来做琴键，加上P2.1用以PWM输出，P2.6和P2.77接外部晶振。剩余的引脚用来达到控制和显示功能还是有点欠缺。

最后利用一个管脚多个功能的做法，将有限的管脚进行复用，解决了管脚少的问题。

3、调试过程中出现了谱子转化正常，但是一些发音不正常的现象，由于程序大体已完成，检查错误时比较困难。

最后通过逐个功能检测的方法，锁定了问题在于多个else if语句的处理顺序上，问题最终得到解决。

五、收获

1、课程开始前，在搭建平台时感觉CCS是个十分复杂的软件，没想到在短短一周之内能够掌握其基本的功能，自己的改变是巨大的。

2、明白了底层硬件处理信息的一些方式，对寄存器、时钟、定时器等东西也有了更多的了解。

3、自己掌握了一些利用CCS编程的能力，认识到能够利用单片机来做到一些自己想要的功能。

课程小建议：实验课动手的时间过于紧张，基本每天都要晚上回去再调试一个多小时才能完成。（也有可能只是我自己比较慢）建议增加课上自己动手的时间，稍微减轻一点课外的压力，课外可以有更多的时间对理论课的知识进行系统的整理。

六、程序源代码

**#include** "msp430.h"

**#define** pace 1.9 //pace用来调整整首歌的节奏

**const** **char** sccr0[21]={250,222,198,187,166,148,132,124,111,98,93,83,73,65,62,55,49,46,41,36,32};

**const** **char** sccr1[21]={25,22,19,18,16,14,13,12,11,9,9,8,7,6,6,5,4,4,4,3,3};

**const** **char** pumian1[207]={22,65,67,37,7,64,34,22,35,35,39,38,68,37,67,7,82,67,68,67,7,52,63,63,35,64,

4,22,22,65,67,37,37,68,69,39,67,69,9,52,69,69,68,11,41,67,37,68,67,7,22,69,

70,69,70,39,67,67,67,65,68,8,22,22,40,40,40,69,69,38,68,69,9,52,69,69,69,

12,38,69,69,9,52,65,65,35,36,67,39,68,8,8,9,9,44,44,14,67,74,44,74,22,67,

44,73,72,72,71,39,9,69,68,67,65,67,38,69,9,64,67,38,69,22,65,67,38,69,9,9,

22,44,44,44,73,72,72,43,74,52,67,67,44,74,73,72,41,69,9,69,68,67,65,9,69,

68,67,65,9,69,68,67,65,65,38,38,8,52,44,74,73,72,74,74,71,12,42,22,71,70,

69,70,70,68,69,68,8,8,73,73,72,73,44,44,79,9,42,42,71,71,39,68,8,8,22,22,

22,22};

**const** **char** pumian2[115]={7,37,39,11,11,12,44,42,11,11,9,9,41,39,7,5,37,38,11,11,

12,12,42,44,11,9,8,39,38,7,8,11,41,40,11,12,12,43,42,11,

14,44,44,12,14,11,41,42,11,11,12,42,42,11,9,8,38,39,11,11,

7,37,37,7,9,8,39,38,7,7,12,42,42,11,9,8,39,38,7,7,

14,14,14,22,12,12,12,22,11,11,12,11,8,9,11,11,

14,14,14,22,12,12,12,22,11,11,12,11,8,39,38,7,7,

22,22};

//pumian1和pumian2存储了两首歌的乐谱

**void** **init0**(); //初始化管脚

**void** **pauseinit**(); //初始化中断有关管脚

**void** **light0**(); //lightx函数用以控制四个信号灯的亮灭状态

**void** **light1\_1**();

**void** **light1\_2**();

**void** **light1\_3**();

**void** **light2**();

**void** **light2\_1**();

**void** **light2\_2**();

**unsigned** pauseif=0; //用来中断音乐播放功能

**int** **main** ( **void** )

{ WDTCTL = WDTPW + WDTHOLD;

**unsigned** **int** state1=0; //state1用来存放音域，0为低音，1为中音，2为高音

**unsigned** **int** key=22; //key用来记录按下的音调()

**unsigned** **int** i;

**unsigned** **int** yd;

**unsigned** **int** sel;

init0(); //初始化引脚

pauseinit();

**while**(1){

light0();

pauseif=0; //pauseif为1时中止当前功能，回到功能选择界面

**while**((P1IN&BIT0)!=0 & (P1IN&BIT1)!=0); //功能选择界面

**if**((P1IN&BIT0)==0){ //表示K1被按下，进入了弹奏功能

**\_\_delay\_cycles**(500000); //稍作延时，以免一次按键被多次响应

**while**(pauseif==0){

key=22;

**if**((P1IN & BIT7)==0) {

state1++;

**\_\_delay\_cycles**(500000);

**if**(state1==3) state1=0;

}

**if**(state1==0) light1\_1();

**else** **if**(state1==1) light1\_2();

**else** **if**(state1==2) light1\_3();

**if**((P1IN & BIT0)==0) key=7\*state1;

**else** **if**((P1IN & BIT1)==0) key=7\*state1 + 1;

**else** **if**((P1IN & BIT2)==0) key=7\*state1 + 2;

**else** **if**((P1IN & BIT3)==0) key=7\*state1 + 3;

**else** **if**((P1IN & BIT4)==0) key=7\*state1 + 4;

**else** **if**((P1IN & BIT5)==0) key=7\*state1 + 5;

**else** **if**((P1IN & BIT6)==0) key=7\*state1 + 6;

**if**(key != 22){

TA1CCR0=sccr0[key];

TA1CCR1=sccr1[key];

**\_\_delay\_cycles**(400000\*pace);

TA1CCR0=32767;

TA1CCR1=32767;

}

**else** **if**(key == 22){

TA1CCR0=32767;

TA1CCR1=32767;

}

}

} //功能1结束

**else** **if**((P1IN&BIT1)==0){ //表示K2被按下，进入了播放功能

light2();

**\_\_delay\_cycles**(500000);

**while**((P1IN&BIT0)!=0 & (P1IN&BIT1)!=0); //再次等待按键来选择曲子

**if**((P1IN&BIT0)==0){ //表示K1被按下，选择了第一首曲子

sel=115; //sel存储了这首曲子的音符总数

light2\_1();

}

**else** **if**((P1IN&BIT1)==0) { //表示K2被按下，选择了第二首曲子

sel=207;

light2\_2();

}

i=0;

**while**((i<sel) & (pauseif==0)){

**if**(sel==207) yd=pumian1[i];

**else** **if**(sel==115) yd=pumian2[i];

**if**(yd==22){

TA1CCR0=32767;

TA1CCR1=32767;

**\_\_delay\_cycles**(400000\*pace);

}

**else** **if**(yd==52){

TA1CCR0=32767;

TA1CCR1=32767;

**\_\_delay\_cycles**(200000\*pace);

}

**else** **if**(yd==82){

TA1CCR0=32767;

TA1CCR1=32767;

**\_\_delay\_cycles**(100000\*pace);

}

**else** **if**(yd>=30 & yd<60){

TA1CCR0=sccr0[yd-30];

TA1CCR1=sccr1[yd-30];

**\_\_delay\_cycles**(180000\*pace);

TA1CCR0=32767;

TA1CCR1=32767;

**\_\_delay\_cycles**(20000\*pace);

}

**else** **if**(yd>=60){

TA1CCR0=sccr0[yd-60];

TA1CCR1=sccr1[yd-60];

**\_\_delay\_cycles**(80000\*pace);

TA1CCR0=32767;

TA1CCR1=32767;

**\_\_delay\_cycles**(20000\*pace);

}

**else** {

TA1CCR0=sccr0[yd];

TA1CCR1=sccr1[yd];

**\_\_delay\_cycles**(370000\*pace);

TA1CCR0=32767;

TA1CCR1=32767;

**\_\_delay\_cycles**(30000\*pace);

}

i++;

}

} //功能2结束

}

}

**void** **init0**(**void**){

P2SEL |=BIT1; //P2.1为PWM输出

P2SEL2 &=~BIT1;

P2DIR |=BIT1;

P2DIR|=BIT2+BIT3+BIT4+BIT5; //P2的2、3、4、5口输出，分别控制L3、L4、L5、L6

P2OUT|=BIT2+BIT3+BIT4+BIT5; //4个信号指示灯初始设为灭

P1OUT = 0xff; //P1的8个引脚分别接收8个按键的信号

P1REN = 0xff;

TA1CTL |=TASSEL0; //打开TA1，用CCR1输出，并设置初始值

TA1CCR0=250;

TA1CCTL1=OUTMOD\_2;

TA1CCR1=25;

TA1CTL |=TACLR+MC0;

TA1CCR0=32767;

TA1CCR1=32767;

}

**void** **pauseinit**(){

\_DINT();

P1IES |=BIT7;

P1IFG &=~BIT7;

P1IE |=BIT7;

\_EINT();

}

**void** **light0**(){

P2OUT &=~(BIT2+BIT3+BIT4+BIT5);

}

**void** **light1\_1**(){

P2OUT &=~BIT2;

P2OUT |=BIT3+BIT4+BIT5;

}

**void** **light1\_2**(){

P2OUT &=~BIT3;

P2OUT |=BIT2+BIT4+BIT5;

}

**void** **light1\_3**(){

P2OUT &=~(BIT2+BIT3);

P2OUT |=BIT4+BIT5;

}

**void** **light2**(){

P2OUT &=~(BIT4+BIT5);

P2OUT |=BIT2+BIT3;

}

**void** **light2\_1**(){

P2OUT &=~BIT4;

P2OUT |=BIT2+BIT3+BIT5;

}

**void** **light2\_2**(){

P2OUT &=~BIT5;

P2OUT |=BIT2+BIT3+BIT4;

}

**#pragma** vector=PORT1\_VECTOR

**\_\_interrupt** **void** **pause1**(){

**if**((P1IFG&BIT7)!=0){

P1IFG&=~BIT7;

**if**((P2OUT&BIT2)!=0 & (P2OUT&BIT3)!=0) pauseif=1;

**else** **if**((P1IN & BIT1)==0) pauseif=1;

}

}