单片机实验课程实验
简易电子琴的设计与实现

祝尔乐

2995441811@qq.com

2021 年 9 月 20 日

1 电子琴的构成及功能描述

1.1 组成部件

msp430G2553 单片机及拓展板，按键，有源蜂鸣器，LED

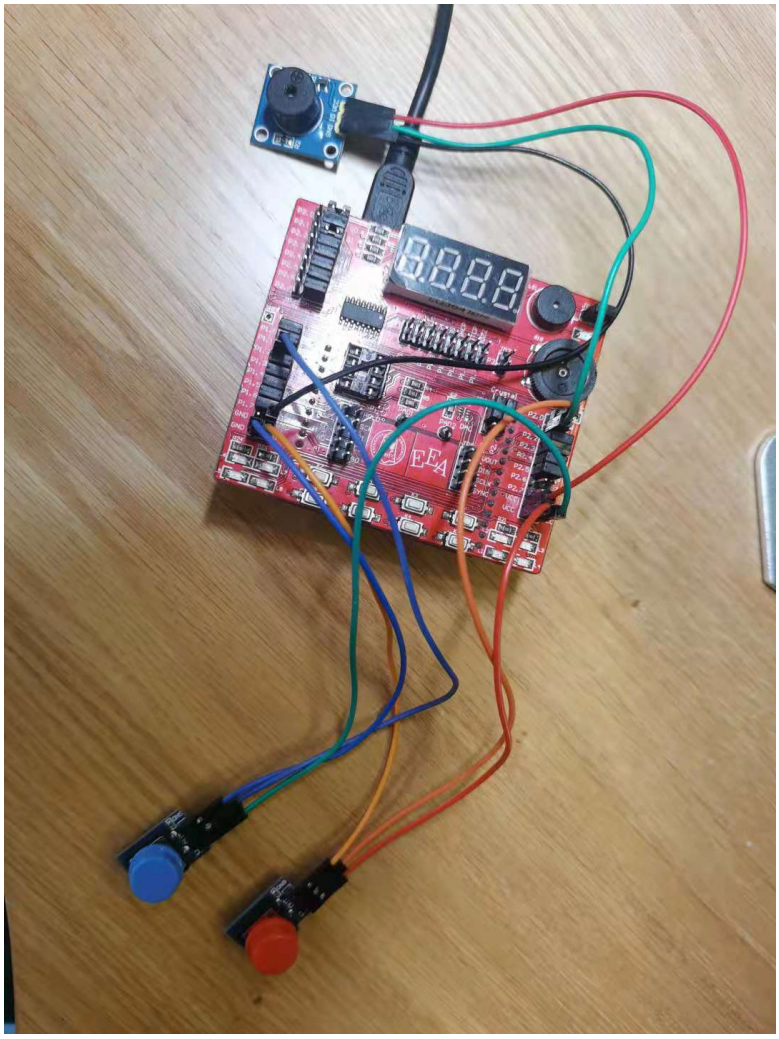

图 1: 电子琴外观

1.2 功能

1) 弹奏功能，可以实现 21 个音符的弹奏，并且弹奏七个音阶会产生不同的灯光效果; 弹奏时按下琴键后发出相应的音符，松开后停止，每次只能发出一种声音；2) 播放功能，可以实现三首曲目的播放，播放过程中随着音乐节奏灯光产生不同的变化，并且播放过程中可以选择暂停/继续播放，停止播放，加速播放，慢速播放；3) 切换功能，在弹奏功能或者播放功能进行的时候，同时按下红蓝二键，可以切换到另一个功能；

1.3 接线图

硬件基本连接示意图如下：

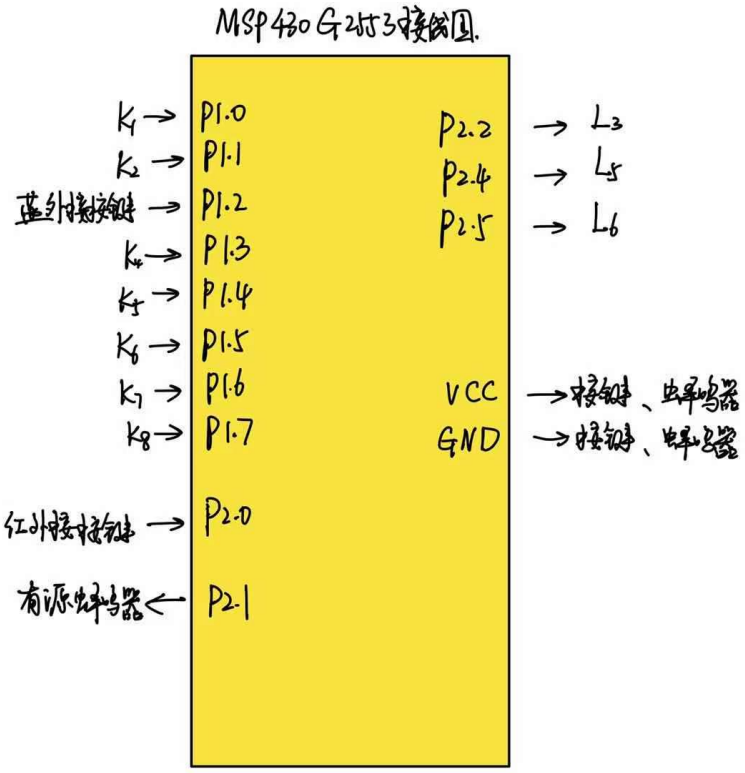

图 2: 接线图

2 实验代码

本实验开发采用的是 TI 公司平台开发的 CCS9.30，代码语言为 C 语言。

2.1 Tone.h

本头文件定义了对应音调频率所需要的 TA1CCR0 值和节拍时长

```
1  /*
2  * Tone.h
3  *
4  *  Created on: 2021年8月27日
5  *      Author: ZEL
6  */
7
8  #ifndef TONE_H_
9  #define TONE_H_
10
```

```
11 //音调
12 #define do_l 249
13 #define re_l 222
14 #define mi_l 198
15 #define fa_l 186
16 #define sol_l 166
17 #define la_l 148
18 #define si_l 132
19 #define do_m 124
20 #define re_m 110
21 #define mi_m 98
22 #define fa_m 93
23 #define sol_m 82
24 #define la_m 73
25 #define si_m 65
26 #define do_h 62
27 #define re_h 55
28 #define mi_h 49
29 #define fa_h 46
30 #define sol_h 41
31 #define la_h 36
32 #define si_h 32
33 #define non 32768 //无音
34 #define jmp 0 //跳转
35
36 //节拍
37 //3/4
38 #define b1_3 2
39 #define b2_3 4
40 #define b4_3 8
41 #define b8_3 16
42
43 #define b1_2 3
44 #define b1 6
45 #define b2 12
46 #define b3 18
47 #define b4 24
48
49 #endif /* TONE_H_ */
```

2.2 Music.h

本头文件在 Flash ROM 存放了三首播放曲目的乐谱

```
1 /*
```

```
/*
 * Music.h
 *
 *  Created on: 2021年8月27日
 *      Author: ZEL
 */

#ifndef MUSIC_H_
#define MUSIC_H_

//设置播放音乐时的灯光引脚
#define L3_pin BIT2
#define L5_pin BIT4
#define L6_pin BIT5


//青花瓷
const unsigned int QHC[92] = {
    non, sol_m, sol_m, mi_m,
    re_m, mi_m, la_l,
    re_m, mi_m, sol_m, mi_m,
    re_m,
    non, sol_m, sol_m, mi_m,
    re_m, mi_m, sol_l,
    re_m, mi_m, sol_m, re_m,
    do_m,
    non, do_m, re_m, mi_m,
    sol_m, la_m, sol_m, fa_m,
    sol_m, mi_m, mi_m, re_m,
    re_m,
    non, do_m, re_m, do_m,
    do_m, re_m, do_m, re_m,
    re_m, mi_m, sol_m,
    mi_m, mi_m,
    //
    non, sol_m, sol_m, mi_m,
    re_m, mi_m, la_l,
    re_m, mi_m, sol_m, mi_m,
    re_m,
    non, sol_m, sol_m, mi_m,
    re_m, mi_m, sol_l,
    re_m, mi_m, sol_m, re_m,
    do_m,
    non, do_m, re_m, mi_m,
    sol_m, la_m, sol_m, fa_m,
```

```
46      sol_m, mi_m, mi_m, re_m,
47      re_m, sol_l,
48      mi_m, re_m, re_m,
49      do_m
50  };
51
52  const unsigned int Beats_QHC[92] = {
53      b1, b1, b1, b1,
54      b1, b1, b2,
55      b1, b1, b1, b1,
56      b4,
57      b1, b1, b1, b1,
58      b1, b1, b2,
59      b1, b1, b1, b1,
60      b4,
61      b1, b1, b1, b1,
62      b1, b1, b1, b1,
63      b1, b1, b1, b1,
64      b4,
65      b1, b1, b1, b1,
66      b1, b1, b1, b1,
67      b1, b2, b1,
68      b4,
69      //
70      b1, b1, b1, b1,
71      b1, b1, b2,
72      b1, b1, b1, b1,
73      b4,
74      b1, b1, b1, b1,
75      b1, b1, b2,
76      b1, b1, b1, b1,
77      b4,
78      b1, b1, b1, b1,
79      b1, b1, b1, b1,
80      b1, b1, b1, b1,
81      b3, b1,
82      b2, b1, b1,
83      b4
84  };
85
86  const int note_num_QHC = 92;
87
88  //菊花台
89  const unsigned int JHT[24] = {
```

```
90      do_m, re_m,
91      mi_m,
92      mi_m, sol_m,
93      la_m,
94      la_m, mi_h,
95      re_h, do_h,
96      do_h, la_m,
97      sol_m,
98      la_m, sol_m,
99      mi_m, re_m,
100     do_m,
101     la_l, do_m,
102     re_m,
103     re_m, do_m,
104     re_m
105 };
106 const unsigned int Beats_JHT[24] = {
107     b2, b2,
108     b4,
109     b2, b2,
110     b4,
111     b2, b2,
112     b2, b2,
113     b2, b2,
114     b4,
115     b2, b2,
116     b2, b2,
117     b4,
118     b2, b2,
119     b4,
120     b2, b2,
121     b4,
122 };
123 const unsigned int note_num_JHT = 24;
124
125 //闪亮的日子
126 const unsigned int SLDRZ[32] = {
127     mi_m, mi_l, mi_l,
128     mi_m, re_m, do_m, re_m, jmp,
129     mi_m, la_m,
130     mi_m, re_m, do_m, re_m, jmp,
131     mi_m, non, do_m, do_m,
132     re_m, re_m, mi_m,
133     la_l, non, la_l,
```

```
    la_l, do_m, si_l, la_l, jmp,
    la_l, non
};
const unsigned int Beats_SLDRZ[32] = {
    b8_3, b2_3, b2_3,
    b2, b2_3, b1_3, b1_3, b2_3,
    b8_3, b4_3,
    b2, b2_3, b1_3, b1_3, b2_3,
    b8_3, b2_3, b1_3, b1_3,
    b4_3, b2, b2_3,
    b8_3, b2_3, b2_3,
    b2, b2_3, b1_3, b1_3, b2_3,
    b8_3, b4_3
};
const unsigned int note_num_SLDRZ = 31;

#endif /* MUSIC_H_ */
```

2.3 E.c

主文件

```
/*
* Electronic_Organ.c
*
*  Created on: 2021年8月27日
*      Author: ZEL
*/
#include "msp430.h"
#include "Tone.h"
#include "Music.h"
#define MOD_1 1 //放歌模式
#define MOD_2 2 //弹奏模式
unsigned int mode;

//音速
#define LOW_SPEED 0x2000
#define SPEED 0x1000
#define HIGH_SPEED 0x800
unsigned int play_speed = SPEED;

//音量
#define VOLUME 10
#define HIGH_VOLUME 50
unsigned int volume = VOLUME;
```

```
24
25 /*
26 * Key_Law
27 * 1.0 k1 mi
28 * 1.1 k2 si
29 * 1.2--key_blue --up
30 * 1.3 k4 la
31 * 1.4 k5 re
32 * 1.5 k6 sol
33 * 1.6 k7 do
34 * 1.7 k8 fa
35 * 2.0 key_red --low
36 * 2.1 buzz
37 */
38 void delay();
39 //蜂鸣器控制函数，传入指定的CCR0，产生特定的音调
40
41
42 void buzz(unsigned int TA1CCR0_set, unsigned int beat)
43 {
44     if(TA1CCR0_set != jmp)
45     {
46         TA1CCR0=TA1CCR0_set;
47         TA1CCR1=TA1CCR0_set / volume;
48         delay(beat);
49         TA1CCR0 = 0;
50     }
51 }
52
53 //延时函数
54 void delay(unsigned int beat)
55 {
56     unsigned int i;
57     for(i = 0;i < beat; i++)
58     {
59         unsigned int j;
60         for(j = 0; j < play_speed; j++);
61     }
62 }
63
64 //歌曲四拍间的暂停
65 void pause()
66 {
67     unsigned int j;
```

```
68      for(j = 0; j < play_speed;j++);
69  }
70
71  //歌曲生成函数
72  void Song_generate(const unsigned int *Song,
73                     const unsigned int *Beats,
74                     const unsigned int note_num,
75                     unsigned int Led_pin)
76  {
77      unsigned int i;
78      unsigned int beats_played = 0;
79      //原速播放
80      play_speed = SPEED;
81      volume = VOLUME;
82      for(i = 0;i < note_num; i++)
83      {
84          P2OUT ^= Led_pin; //灯光取反
85          buzz(Song[i], Beats[i]);
86          beats_played += Beats[i];
87          if(beats_played == b4)
88          {
89              beats_played = 0;
90              pause();
91          }
92          if((P1IN & BIT7) == 0) //按下K8, 停止播放
93          {
94              P2OUT |= Led_pin;
95              return;
96          }
97      }
98      P2OUT |= Led_pin;
99  }
100
101 //歌曲播放模式
102 void Songs_singing_MODE()
103 {
104     P2OUT &= ~BIT3;
105     //按下蓝键, 切换至弹奏模式
106     if((P1IN & BIT2) != 0 && (P2IN & BIT0) != 0)
107     {
108         mode = MOD_2; //模式切换
109     }
110     else if((P1IN & BIT4) == 0) //K5按下, 播放青花瓷, L3亮
111     {
```

```
112             Song_generate(QHC, Beats_QHC, note_num_QHC, L3_pin);
113         }
114         else if((P1IN & BIT5) == 0) //K6按下，播放菊花台，  L5亮
115         {
116             Song_generate(JHT, Beats_JHT, note_num_JHT, L5_pin);
117         }
118         else if((P1IN & BIT6) == 0) //K7按下，  播放闪亮的日子，  L6亮
119         {
120             Song_generate(SLDRZ, Beats_SLDRZ, note_num_SLDRZ, L6_pin);
121         }
122     }
123
124     //弹奏模式
125     void Play_song_MODE( )
126     {
127         //按下红键，切换至播放模式
128         if((P1IN & BIT2) != 0 && (P2IN & BIT0) != 0)
129         {
130             mode = MOD_1; //模式切换
131         }
132         if((P1IN & BIT0) == 0) //1.0 --mi
133         {
134             while((P1IN & BIT0) == 0)
135             {
136                 P2OUT ^= BIT5;
137                 if((P1IN & BIT2) != 0) buzz(mi_h, b1_2);
138                 else if((P2IN & BIT0) != 0) buzz(mi_l, b1_2);
139                 else buzz(mi_m, b1_2);
140                 P2OUT ^= BIT5;
141             }
142         }
143         else if((P1IN & BIT1) == 0)
144         {
145             while((P1IN & BIT1) == 0)
146             {
147                 P2OUT ^= BIT2;
148                 P2OUT ^= BIT4;
149                 P2OUT ^= BIT5;
150                 if((P1IN & BIT2) != 0) buzz(si_h, b1_2);
151                 else if((P2IN & BIT0) != 0) buzz(si_l, b1_2);
152                 else buzz(si_m, b1_2);
153                 P2OUT ^= BIT2;
154                 P2OUT ^= BIT4;
155                 P2OUT ^= BIT5;
```

```
156             }
157         }
158         else if((P1IN & BIT3) == 0)
159         {
160             while((P1IN & BIT3) == 0)
161             {
162                 P2OUT ^= BIT4;
163                 P2OUT ^= BIT5;
164                 if((P1IN & BIT2) != 0) buzz(la_h, b1_2);
165                 else if((P2IN & BIT0) != 0) buzz(la_l, b1_2);
166                 else buzz(la_m, b1_2);
167                 P2OUT ^= BIT4;
168                 P2OUT ^= BIT5;
169             }
170         }
171         else if((P1IN & BIT4) == 0)
172         {
173             while((P1IN & BIT4) == 0)
174             {
175                 P2OUT ^= BIT4;
176                 if((P1IN & BIT2) != 0) buzz(re_h, b1_2);
177                 else if((P2IN & BIT0) != 0) buzz(re_l, b1_2);
178                 else buzz(re_m, b1_2);
179                 P2OUT ^= BIT4;
180             }
181         }
182         else if((P1IN & BIT5) == 0)
183         {
184             while((P1IN & BIT5) == 0)
185             {
186                 P2OUT ^= BIT2;
187                 P2OUT ^= BIT5;
188                 if((P1IN & BIT2) != 0) buzz(sol_h, b1_2);
189                 else if((P2IN & BIT0) != 0) buzz(sol_l, b1_2);
190                 else buzz(sol_m, b1_2);
191                 P2OUT ^= BIT2;
192                 P2OUT ^= BIT5;
193             }
194         }
195         else if((P1IN & BIT6) == 0)
196         {
197             while((P1IN & BIT6) == 0)
198             {
199                 P2OUT ^= BIT2;
```

```
200                 if((P1IN & BIT2) != 0) buzz(do_h, b1_2);
201                 else if((P2IN & BIT0) != 0) buzz(do_l, b1_2);
202                 else buzz(do_m, b1_2);
203                 P2OUT ^= BIT2;
204             }
205         }
206         else if((P1IN & BIT7) == 0)
207         {
208             while((P1IN & BIT7) == 0)
209             {
210                 P2OUT ^= BIT2;
211                 P2OUT ^= BIT4;
212                 if((P1IN & BIT2) != 0) buzz(fa_h, b1_2);
213                 else if((P2IN & BIT0) != 0) buzz(fa_l, b1_2);
214                 else buzz(fa_m, b1_2);
215                 P2OUT ^= BIT2;
216                 P2OUT ^= BIT4;
217             }
218         }
219     }
220
221     int main()
222     {
223         WDTCTL = WDTPW + WDTHOLD; //关闭看门狗
224         P2SEL |=BIT1; //置 P2.1为定时器 TA1 的 PWM 输出引脚
225         P2SEL2 &=~(BIT1); //P2.1 为比较器 1 的 PWM 输出引脚
226         P2DIR |= BIT1;
227         //选择 TA1 计数时钟为 ACLK, 使用上电复位设置, 即 32768Hz
228         TA1CTL |=TASSEL0;
229
230     /*  设置增计数方式, 使计数器从 0 开始计数,
231         计数到 TA1CCR0 后又从 0 计数。*/
232         TA1CCTL1|=OUTMOD1;
233
234         TA1CTL |=TACLR+MC0;
235
236         //LED引脚设置, p2.2, 2.4, 2.5输出 //
237         P2SEL &= ~(BIT2 + BIT4 + BIT5);
238         P2SEL2 &= ~(BIT2 +BIT4 + BIT5);
239         P2DIR |= (BIT2 + BIT4 + BIT5);
240         P2OUT |= (BIT2 + BIT4 + BIT5);
241
242         //开关引脚设置, p1全为输入 p2.0输入
243         P1SEL = 0x00;
```

```
244     P1SEL2 = 0x00;
245     P1DIR = 0x00;
246     P1REN = 0xff;
247     P1OUT = 0xfb;
248
249     P2SEL &= ~BIT0;
250     P2SEL2 &= ~BIT0;
251     P2DIR &= ~BIT0;
252     P2REN |= BIT0;
253     P2OUT &= ~BIT0;
254
255     //中断引脚设置1.0, 1.1, 1.3,
256     P1IES |= (BIT0 + BIT1 + BIT3);
257     P1IFG &=~(BIT0 + BIT1 + BIT3);
258     P1IE |= (BIT0 + BIT1 + BIT3);
259
260     TA1CCR0 = 0;
261
262     mode = MOD_1;
263     _EINT();
264     while(1){
265         if(mode == MOD_1)
266         {
267             Songs_singing_MODE();
268         }
269         else if(mode == MOD_2)
270         {
271             Play_song_MODE();
272         }
273     };
274 }
275
276
277 //中断
278 #pragma vector=PORT1_VECTOR
279 __interrupt void port_ISR( )
280 {
281     if(mode == MOD_1)//模式1
282     {
283         if((P1IN & BIT0) == 0) //k1 ——暂停
284         {
285             unsigned int temp = TA1CCR0;
286             TA1CCR0 = 0;
287             while((P1IN & BIT0) != 0);
```

```
288                 TA1CCR0 = temp;
289             }
290             else if((P1IN & BIT1) == 0) //k2 快速模式
291             {
292                 play_speed = HIGH_SPEED;
293             }
294             else if((P1IN & BIT3) == 0) //k4 慢速模式
295             {
296                 play_speed = LOW_SPEED;
297
298             }
299         }
300         P1IFG &=~(BIT0 + BIT1 + BIT3);
301     }
```

3 功能实现

3.1 弹奏功能

1. 通过单片机拓展板上的七个按键配合上外接的红蓝二键可以实现 21 个音符的弹奏;

表 1: 琴键与音阶对应关系

	$K7$	$K5$	$K1$	$K8$	$K6$	$K4$	$K2$
低音：+ 红键	do_l	re_l	mi_l	fa_l	sol_l	la_l	si_l
中音：不加	do_m	re_m	mi_m	fa_m	sol_m	la_m	si_m
高音：+ 蓝键	do_h	re_h	mi_h	fa_h	sol_h	la_h	si_h

2. 在弹奏七个音阶会产生不同的灯光效果; 对应关系如下：

表 2: 灯光与音阶对应关系

灯光	$L3$	$L5$	$L6$
do	1	0	0
re	0	1	0
mi	0	0	1
fa	1	1	0
sol	1	0	1
la	0	1	1
si	1	1	1

3. 弹奏时按下琴键后发出相应的音符，松开后停止，每次只能发出一种声音；

3.2 播放功能

1. 可以实现三首曲目的播放，每首歌的播放对应一种灯光：K7——闪亮的日子，K5——青花瓷，K6——菊花台；

2. 播放过程中随着音乐节拍灯光进行闪动，歌曲结束后灯光置灭；

3. 播放过程中可以选择暂停/继续播放（K1)，停止播放 (K8)，加速播放 (K2)，慢速播放 (K4)；

3.3 切换功能

在弹奏功能或者播放功能进行的时候，同时按下红蓝二键，可以切换到另一个功能

4 项目演示视频

演示视频在以下链接：https://cloud.tsinghua.edu.cn/d/94c0ab05f8b34ed3b001/

5 课程收获

1. 学习了单片机的基本构造和 CPU 的部分原理，加深了对计算机结构的理解；

2. 通过各项实验锻炼了分析、调试程序的能力，学习了有关接线、硬件接口的知识；

3. 通过电子琴的设计巩固了单片机的相关知识，提高了“学以致用”的能力，为今后的科研和科创活动打下了基础；