# 第2章 八阶光立方实现蓝牙控制项目设计

### 2.2.1 主程序模块

#### 2.相关代码

//蓝牙功能代码如下：

#ifndef LightCube\_h

#define LightCube\_h

#include "Arduino.h"

#define u8 unsigned char

#define s8 signed char

#define u16 unsigned short

class LightCube

{

public:

LightCube(int SER0 ,int SCK0 ,int RCK0 );

void storey(u8 \*a);

void frame(u8 \*a,u8 v); //表示一帧,a是一帧编码起始地址

// v表示一帧画面扫描的次数

// 可以看作这帧显示的时间

void AllOff(void); //光立方512个灯全关—清屏

void AllOn(u8 v); //光立方512个灯全开

void LtoRScan(u8 v); //侧面从左向右依次点亮

void RtoLScan(u8 v); //侧面从右向左依次点亮

void FtoBScan(u8 v); //从前排向后排依次点亮

void BtoFScan(u8 v); //从后排向前排依次点亮

void UtoDScan(u8 v); //从上向下依次点亮

void DtoUScan(u8 v); //从下向上依次点亮

void Anticlockwise(s8 num,s8 v); //Anticlockwise表示逆时针旋转 num:表示旋转周数, v表示速度

void Clockwise(s8 num,s8 v); //Clockwise表示顺时针旋转，num表示旋转周数, v表示速度

void RightClockwise(s8 num,s8 v); //从右侧看顺时针旋转，num表示旋转周数, v表示速度

void Cube(u8 empty,u8 kind,u8 v); //立方体动画；empty=0表示空, empty=1表示实

//kind=0表示左上角, /kind=1表示右上角, /kind=2表示左下角, /kind=3表示右下脚；v表示速度

void Rain(s8 down, s8 cycle \_index, u8 speed); //下雨效果

//cycle\_index为旋转周数；speed为速度

void Up(s8 num,s8 v); //上移；num为旋转周数；v为速度

void RotateFace(s8 empty,s8 Clockwise,s8 turns,u8 speed);//面中心旋转

//empty=1表示空心旋转,否则表示实心旋转

//Clockwise=1表示顺时针，否则表示逆时针

//turns为执行次数；speed为速度

void RotateFaceC(s8 num,s8 v); //面侧边旋转,num执行次数；v速度

void RotateHookFace(s8 Clockwise,s8 turns,u8 speed); //曲面旋转;Clockwise=1表示逆时针旋转,否则表示顺时针旋转，turns为执行次数；speed为速度

void Sandglass(s8 v); //沙漏；v表示速度

void WaterOne(s8 x,s8 y,u8 speed); //一个水滴

void WaterTwo(s8 x1,s8 y1,s8 x2,s8 y2,u8 speed); //二个水滴

void WaterThr(s8 x1,s8 y1,s8 x2,s8 y2,s8 x3,s8 y3,u8 speed); //三个水滴

void Animation(const u8 \*ARRAY\_tab,u8 frame\_num,u8 cycle\_index,u8 speed);

//\*动画，ARRAY\_tab,array.h中数组；

//frame\_num表示帧数;cycle\_index表示循环次数,speed表示滚动速度

void PrintOpen(u8 speed);

void Print(const u8 \*ARRAY\_tab);

void FanOne(s8 turns, u8 speed); //fan表示扇形

void FanTwo(s8 turns, u8 speed);

void ConnectThree(u8 speed);

void ConnectFour(u8 speed);

void ConnectFive(u8 speed);

void ConnectSix(const u8 \*ARRAY\_tab,u8 frame\_num,u8 cycle\_index,u8 speed);

//cycle\_index表示循环次数,speed表示滚动速度

void ConnectSeven(u8 speed);

private:

int \_SER0;

int \_SCK0;

int \_RCK0;

};

#endif

#include "Arduino.h"

#include "LightCube.h"

#define u8 unsigned char

#define s8 signed char

#define u16 unsigned short

//初始化函数

LightCube::LightCube(int SER0 ,int SCK0 ,int RCK0 )

{

pinMode(SER0,OUTPUT);

\_SER0 = SER0;

pinMode(SCK0,OUTPUT);

\_SCK0 = SCK0;

pinMode(RCK0,OUTPUT);

\_RCK0 = RCK0;

}

const u8 flash\_tab[]=

{

0X03,0X07,0X06,0X08,0X10,0X20,0X40,0X80,//0

0X07,0X07,0X0F,0X1C,0X18,0X20,0X40,0X80,

0X0F,0X0F,0X1F,0X3F,0X3C,0X78,0X60,0X80,

0X1F,0X1F,0X3F,0X3F,0X7F,0X7C,0X70,0X80,

0X3F,0X3F,0X3F,0X7F,0X7F,0X7F,0XF8,0XC0,

0X7F,0X7F,0X7F,0X7F,0XFF,0XFF,0XFF,0XF0,

0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF//6

};

void LightCube::storey(u8 \*a)//层填充函数，控制某层灯点亮方式

{

u8 i,j,num;

for(i=0;i<8;i++)

{

num=a[i]; //将数组中数输入寄存器

for(j=0;j<8;j++) //串行数据输入

{

if(num&0x80)

digitalWrite(\_SER0,HIGH); // SER串行输入端口

else

digitalWrite(\_SER0,LOW);

digitalWrite(\_SCK0,LOW); //上升沿，输入到移位寄存器

delayMicroseconds(1);

digitalWrite(\_SCK0,HIGH);

num<<=1;

}

}

}

void LightCube::frame(u8 \*a,u8 v) //表示一帧,a是一帧编码起始地址

// v表示一帧画面扫描的次数

// 可以看作这帧显示的时间

{

s8 i,j,num; //s8 有符号定义

while(v--)

{

num=0x01;

for(i=0;i<8;i++) //层数控制

{

num<<=i;

digitalWrite(\_RCK0,LOW);

for(j=0;j<8;j++) //串行数据输入

{

digitalWrite(\_SER0,LOW);

delayMicroseconds(1);

digitalWrite(\_SCK0,LOW); //上升沿，输入到移位寄存器

delayMicroseconds(1);

digitalWrite(\_SCK0,HIGH);

}

for(j=0;j<8;j++) //串行数据输入

{

if(num&0x80)

digitalWrite(\_SER0,HIGH); // SER串行输入端口

else

digitalWrite(\_SER0,LOW);

digitalWrite(\_SCK0,LOW); //上升沿，输入到移位寄存器

delayMicroseconds(1);

digitalWrite(\_SCK0,HIGH);

num<<=1;

}

storey(a+i\*8); //层填充函数，控制某层灯点亮方式

digitalWrite(\_RCK0,HIGH);

num=0x01;

delayMicroseconds(2); // 层显示时间

}

}

}

//全局清屏

void LightCube::AllOff(void)

{

u8 i,j;

for(i=0;i<9;i++) //层数控制

{

digitalWrite(\_RCK0,LOW);

for(j=0;j<8;j++) //串行数据输入

{

digitalWrite(\_SER0,LOW);

delayMicroseconds(1);

digitalWrite(\_SCK0,LOW); //上升沿，输入到移位寄存器

delayMicroseconds(1);

digitalWrite(\_SCK0,HIGH);

delayMicroseconds(1);

}

}

digitalWrite(\_RCK0,HIGH);

delay(2);

}

// 光立方全点亮

void LightCube::AllOn(u8 v)

{

u8 k[65]={

0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,

0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,

0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,

0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,

0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,

0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,

0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,

0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF

};

frame(k,v);

}

//侧面从左向右依次点亮

void LightCube::LtoRScan(u8 v)

{

u8 b[64]={0 };

s8 x,z;

for(z=0;z<8;z++)

{

b[z\*8]=0xff;

}

frame(b,v);

for(x=1;x<8;x++)

{

for(z=0;z<8;z++)

{

b[z\*8+x]=0xff;

b[z\*8+(x-1)]=0;

}

frame(b,v);

}

}

//侧面从右向左依次点亮

void LightCube::RtoLScan(u8 v)

{

u8 b[64]={0};

s8 x,z;

for(z=0;z<8;z++)

{

b[z\*8+7]=0xff;

}

frame(b,v);

for(x=6;x>=0;x--)

{

for(z=0;z<8;z++)

{

b[z\*8+x]=0xff;

b[z\*8+(x+1)]=0;

}

frame(b,v);

}

}

//从前排向后排依次点亮

void LightCube::FtoBScan(u8 v)

{

u8 b[64]={0};

s8 i,y;

for(i=0;i<64;i++)

b[i]=0x01;

frame(b,v);

for(y=0;y<7;y++)

{

for(i=0;i<64;i++)

b[i]<<=1;

frame(b,v);

}

}

//从后排向前排依次点亮

void LightCube::BtoFScan(u8 v)

{

u8 b[64]={0};

s8 i,y;

for(i=0;i<64;i++)

b[i]=0x80;

frame(b,v);

for(y=0;y<7;y++)

{

for(i=0;i<64;i++)

b[i]>>=1;

frame(b,v);

}

}

//从上向下依次点亮

void LightCube::UtoDScan(u8 v)

{

u8 b[64]={0};

s8 x,z;

for(x=0;x<8;x++)

{

b[x]=0xff;

}

frame(b,v);

for(z=1;z<8;z++)

{

for(x=0;x<8;x++)

{

b[z\*8+x]=0xff;

b[(z-1)\*8+x]=0;

}

frame(b,v);

}

}

//从下向上依次点亮

void LightCube::DtoUScan(u8 v)

{

u8 b[64]={0};

s8 x,z;

for(x=0;x<8;x++)

{

b[7\*8+x]=0xff;

}

frame(b,v);

for(z=6;z>=0;z--)

{

for(x=0;x<8;x++)

{

b[z\*8+x]=0xff;

b[(z+1)\*8+x]=0;

}

frame(b,v);

}

}

//Anticlockwise表示逆时针选好转num表示旋转周数，v表示速度

void LightCube::Anticlockwise(s8 num,s8 v)

{

s8 i,k;

u8 b[64]={0};

for(i=0;i<64;i++) b[i]=0x80;

frame(b,v);

while(num--)

for(i=0;i<28;i++)

{

if(i<7)

{

for(k=0;k<8;k++)

{

b[k\*8]|=(0x80>>(i+1));

b[k\*8+7-i]=0;

}

}

else if(i<14)

{

for(k=0;k<8;k++)

{

b[k\*8+i-6]=0x01;

b[k\*8]>>=0x01;

}

}

else if(i<21)

{

for(k=0;k<8;k++)

{

b[k\*8+i-14]=0;

b[k\*8+7]|=(0x01<<(i-13));

}

}

else if(i<28)

{

for(k=0;k<8;k++)

{

b[k\*8+27-i]=0x80;

b[k\*8+7]<<=1;

}

}

frame(b,v);

}

AllOff(); //清屏

}

//Clockwise表示顺时针旋转，num表示旋转周数，v表示速度

void LightCube::Clockwise(s8 num,s8 v)

{

s8 i,k;

u8 b[64]={0};

for(i=0;i<64;i++) b[i]=0x80;

frame(b,v);

while(num--)

for(i=0;i<28;i++)

{

if(i<7)

for(k=0;k<8;k++)

{

b[k\*8+7]|=(0x80>>(i+1));

b[k\*8+i]=0;

}

else if(i<14)

for(k=0;k<8;k++)

{

b[k\*8+13-i]=0x01;

b[k\*8+7]>>=0x01;

}

else if(i<21)

for(k=0;k<8;k++)

{

b[k\*8+21-i]=0;

b[k\*8]|=(0x01<<(i-13));

}

else if(i<28)

for(k=0;k<8;k++)

{

b[k\*8+i-20]=0x80;

b[k\*8]<<=1;

}

frame(b,v);

}

AllOff(); //清屏

}

//从右侧看顺时针旋转

void LightCube::RightClockwise(s8 num,s8 v)

{

s8 i,k;

u8 a[64]={0};

for(i=0;i<8;i++)

a[7\*8+i]=0xff;

frame(a,v);

while(num--)

for(i=0;i<28;i++)

{

if(i<7)

for(k=0;k<8;k++)

{

a[(6-i)\*8+k]=0x01;

a[7\*8+k]>>=1;

}

else if(i<14)

for(k=0;k<8;k++)

{

a[k]|=0x01<<(i-6);

a[(14-i)\*8+k]=0;

}

else if(i<21)

for(k=0;k<8;k++)

{

a[(i-13)\*8+k]=0x80;

a[k]<<=1;

}

else

for(k=0;k<8;k++)

{

a[7\*8+k]|=0x80>>(i-20);

a[(i-21)\*8+k]=0;

}

frame(a,v);

}

AllOff();

}

//在X、Y、Z轴上任意方向和位数移动

s8 X\_AXIS=0;

s8 Y\_AXIS=1;

s8 Z\_AXIS=2;

s8 MINUS=0; //负向

s8 PULS=1; //正向

// kind=0表示 x轴, kind=1表示y轴, kind=2表示z轴；direction=0表示负向, direction=1表示正向;length表示移动位数，不能为8

void moveXYZ(u8 \*a,s8 kind,s8 direction,s8 length){

s8 i,x,z;

if(kind==0) //x轴

{

if(direction==1) //正向

for(z=0;z<8;z++)

{

for(x=7;x>=length;x--)

a[z\*8+x]=a[z\*8+(x-length)];

for(x=0;x<length;x++)

a[z\*8+x]=0;

}

else // 负向

for(z=0;z<8;z++)

{

for(x=length;x<8;x++)

a[z\*8+(x-length)]=a[z\*8+x];

for(x=(8-length);x<8;x++)

a[z\*8+x]=0;

}

}

else if(kind==1) //y轴

{

if(direction==1) //正向

for(i=0;i<64;i++)

a[i]<<=length;

else //负向

for(i=0;i<64;i++)

a[i]>>=length;

}

else //z轴

{

if(direction==1) //正向

for(x=0;x<8;x++)

{

for(z=7;z>=length;z--)

a[z\*8+x]=a[(z-length)\*8+x];

for(z=0;z<length;z++)

a[z\*8+x]=0;

}

else //负向

for(x=0;x<8;x++)

{

for(z=length;z<8;z++)

a[(z-length)\*8+x]=a[z\*8+x];

for(z=(8-length);z<8;z++)

a[z\*8+x]=0;

}

}

}

//正方体轮廓,outline表示轮廓,length表示边长，范围为0<leng<=8,注意n不能为0

void cubeLine(u8 \*a,s8 n)

{

s8 i,j;

for(i=0;i<64;i++)

a[i]=0;

j=0xff>>(8-n);

a[0]=j;

a[n-1]=j;

a[(n-1)\*8]=j;

a[(n-1)\*8+n-1]=j;

for(i=0;i<n;i++)

{

j=(0x01|(0x01<<(n-1)));

a[i\*8]|=j;

a[i\*8+n-1]|=j;

a[i]|=j;

a[(n-1)\*8+i]|=j;

}

}

//实心正方体,fill表示充满,length表示边长，范围为0<leng<=8,注意n不能为0

void CubeFill(u8 \*a, s8 length)

{

s8 x, z;

for(z = 0;z < 8; z++)

for(x = 0; x < 8; x++)

{

if(z < length && x < length)

a[ z \* 8 + x] = 0xff >> (8 - length);

else

a[z \* 8 + x] = 0;

}

}

//立方体动画，empty=0表示空心, empty=1表示实心; kind=0表示左上角, kind=1表示右上角, kind=2表示左下角, kind=3表示右下脚；num表示旋转周数；v表示速度

void LightCube::Cube(u8 empty,u8 kind,u8 v)

{

u8 a[64]={0};

s8 i;

for(i=1;i<=8;i++)//生成

{

if(empty==0)

cubeLine(a,i);

else

CubeFill(a,i);

if(kind==0)

{;}

else if(kind==1)

moveXYZ(a,0,1,8-i);

else if(kind==2)

moveXYZ(a,2,1,8-i);

else

{

moveXYZ(a,0,1,8-i);

moveXYZ(a,2,1,8-i);

}

frame(a,v);

}

for(i=7;i>0;i--) //退出

{

if(empty==0)

cubeLine(a,i);

else

CubeFill(a,i);

if(kind==0)

moveXYZ(a,0,1,8-i);

else if(kind==1)

{

moveXYZ(a,0,1,8-i);

moveXYZ(a,2,1,8-i);

}

else if(kind==2)

{;}

else

moveXYZ(a,2,1,8-i);

frame(a,v);

}

}

const u8 rain\_tab[]=

{

0X00,0X84,0X01,0X00,0X00,0X01,0X02,0X44,

0X84,0X00,0X24,0X08,0X04,0X20,0X80,0X00,

0X00,0X20,0X00,0X00,0X20,0X00,0X40,0X00,

0X00,0X04,0X00,0X40,0X00,0X00,0X08,0X00,

0X40,0X00,0X10,0X00,0X00,0X04,0X04,0X11,

0X00,0X41,0X00,0X00,0X80,0X00,0X40,0X00,

0X00,0X00,0X00,0X24,0X01,0X18,0X00,0X00,

0X22,0X10,0X02,0X00,0X40,0X00,0X02,0X00

};

//下雨效果

void LightCube::Rain(s8 down, s8 cycle\_index, u8 speed)

{

u8 b[64] = {0};

s8 x, z;

if(down == 1) //落下

{

for(x = 0; x < 8; x++)

b[x] = rain\_tab[x];

frame(b, speed);

for(z = 1; z < 8; z++)

{

moveXYZ(b, Z\_AXIS, PULS, 1);

for(x = 0; x < 8; x++)

b[x] = rain\_tab[z \* 8 + x];

frame(b, speed);

}

while(cycle\_index--)

{

for(z = 0; z < 8; z++)

{

moveXYZ(b, Z\_AXIS, PULS, 1);

for(x = 0; x < 8; x++)

b[x] = rain\_tab[z \* 8 + x];

frame(b, speed);

}

}

}

else //上升

{

for(x = 0; x < 8; x++)

b[56 + x] = rain\_tab[x];

frame(b, speed);

for(z = 1; z < 8; z++)

{

moveXYZ(b, Z\_AXIS, MINUS, 1);

for(x = 0; x < 8; x++)

b[56 + x] = rain\_tab[z \* 8 + x];

frame(b, speed);

}

while(cycle\_index--)

{

for(z = 0; z < 8; z++)

{

moveXYZ(b, Z\_AXIS, MINUS, 1);

for(x = 0; x < 8; x++)

b[56 + x] = rain\_tab[z \* 8 + x];

frame(b, speed);

}

}

}

}

//光立方上移

void LightCube::Up(s8 num,s8 v)

{

u8 a[64]={0};

s8 X,Z;

while(num--)

{

for(X=0;X<8;X++)

a[56+X]=0xff;

frame(a,v);

for(Z=1;Z<8;Z++)

{

moveXYZ(a,2,0,1);

for(X=0;X<8;X++)

a[56+X]=0xff;

frame(a,v);

}

for(Z=0;Z<8;Z++)

{

if(num==0&&Z==7)

continue; //结束本次循环，而不是终止整个语句的循环

moveXYZ(a,2,0,1);

frame(a,v);

}

}

for(Z=0;Z<7;Z++)

{

moveXYZ(a,2,1,1);

frame(a,v+0);

}

}

//旋转条

const u8 rotate\_tab[]=

{

0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,//0

0x00,0x01,0x06,0x08,0x10,0x60,0x80,0x00,

0x00,0x00,0x01,0x0e,0x70,0x80,0x00,0x00,

0x00,0x00,0x00,0x0f,0xf0,0x00,0x00,0x00,

0x00,0x00,0x00,0xf0,0x0f,0x00,0x00,0x00,

0x00,0x00,0x80,0x70,0x0e,0x01,0x00,0x00,

0x00,0x80,0x60,0x10,0x08,0x06,0x01,0x00,

0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01,

0x40,0x20,0x20,0x10,0x08,0x04,0x04,0x02,

0x20,0x10,0x10,0x10,0x08,0x08,0x08,0x04,

0x10,0x10,0x10,0x10,0x08,0x08,0x08,0x08,

0x08,0x08,0x08,0x08,0x10,0x10,0x10,0x10,

0x04,0x08,0x08,0x08,0x10,0x10,0x10,0x20,

0x02,0x04,0x04,0x08,0x10,0x20,0x20,0x40//13

};

//面旋转，empty=1表示空心旋转,否则表示实心旋转 ,Clockwise=1表示顺时针旋转，否则表示逆时针旋转

void LightCube::RotateFace(s8 empty, s8 Clockwise, s8 turns, u8 speed)

{

u8 b[64] = {0};

s8 i, x, z;

while(turns--)

{

if(Clockwise == 1)

for(i = 13; i >= 0; i--)

{

for(z = 0; z < 8; z++)

{

for(x = 0; x < 8; x++)

{

if(x>1 && x<6 && z>1 && z<6 && empty==1)

b[z \* 8 + x] = rotate\_tab[i \* 8 + x] & 0xc3;

else

b[z \* 8 + x] = rotate\_tab[i \* 8 + x];

}

}

frame(b, speed);

}

else

for(i = 0; i < 14; i++)

{

for(z = 0; z < 8; z++)

{

for(x = 0; x < 8; x++)

{

if(x>1 && x<6 && z>1 && z<6 && empty==1)

b[z \* 8 + x] = rotate\_tab[i \* 8 + x] & 0xc3;

else

b[z \* 8 + x]=rotate\_tab[ i \* 8 + x];

}

}

frame(b, speed);

}

}

}

const u8 tab\_xuanzhuantiao[]=

{ //旋转条

0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80,//0

0x00,0x01,0x06,0x08,0x10,0x60,0x80,0x00,

0x00,0x00,0x01,0x0e,0x70,0x80,0x00,0x00,

0x00,0x00,0x00,0x0f,0xf0,0x00,0x00,0x00,

0x00,0x00,0x00,0xf0,0x0f,0x00,0x00,0x00,

0x00,0x00,0x80,0x70,0x0e,0x01,0x00,0x00,

0x00,0x80,0x60,0x10,0x08,0x06,0x01,0x00,

0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01,

0x40,0x20,0x20,0x10,0x08,0x04,0x04,0x02,

0x20,0x10,0x10,0x10,0x08,0x08,0x08,0x04,

0x10,0x10,0x10,0x10,0x08,0x08,0x08,0x08,

0x08,0x08,0x08,0x08,0x10,0x10,0x10,0x10,

0x04,0x08,0x08,0x08,0x10,0x10,0x10,0x20,

0x02,0x04,0x04,0x08,0x10,0x20,0x20,0x40//13

};

//旋转条

const u8 tab\_xuanzhuantiao2[]=

{

0X01,0X02,0X04,0X08,0X10,0X20,0X40,0X80,//0

0X02,0X04,0X08,0X00,0X10,0X20,0X40,0X80,//

0X04,0X08,0X00,0X10,0X20,0X20,0X40,0X80,//

0X08,0X10,0X10,0X20,0X20,0X40,0X40,0X80,//

0X10,0X10,0X20,0X20,0X20,0X40,0X40,0X80,//

0X20,0X20,0X40,0X40,0X40,0X80,0X80,0X80,//

0X40,0X40,0X40,0X40,0X80,0X80,0X80,0X80,//

0X80,0X80,0X80,0X80,0X80,0X80,0X80,0X80,//7

0X80,0X80,0X80,0X80,0X40,0X40,0X40,0X40,//

0X80,0X80,0X00,0X40,0X40,0X40,0X20,0X20,//9

0X80,0X80,0X40,0X40,0X20,0X20,0X10,0X10,//

0X80,0X40,0X40,0X20,0X20,0X10,0X10,0X08,//

0X80,0X40,0X20,0X20,0X10,0X10,0X08,0X04,//

0X80,0X40,0X20,0X10,0X08,0X08,0X04,0X02,//

0X80,0X40,0X20,0X10,0X08,0X04,0X02,0X01,//14

0X80,0X40,0X20,0X10,0X0C,0X02,0X01,0X00,//15

0X80,0X40,0X30,0X0C,0X02,0X01,0X00,0X00,//

0X80,0X60,0X18,0X06,0X01,0X00,0X00,0X00,//17

0XC0,0X30,0X0E,0X01,0X00,0X00,0X00,0X00,//

0XE0,0X1C,0X03,0X00,0X00,0X00,0X00,0X00,//19

0XF0,0X0F,0X00,0X00,0X00,0X00,0X00,0X00,//

0XFF,0X00,0X00,0X00,0X00,0X00,0X00,0X00,//21

0X0F,0XF0,0X00,0X00,0X00,0X00,0X00,0X00,//22

0X07,0X38,0XC0,0X00,0X00,0X00,0X00,0X00,

0X03,0X0C,0X20,0XC0,0X00,0X00,0X00,0X00,

0X01,0X02,0X0C,0X30,0XC0,0X00,0X00,0X00,

0X01,0X02,0X04,0X08,0X30,0XC0,0X00,0X00,

0X01,0X02,0X04,0X08,0X10,0X60,0X80,0X00,

0X01,0X02,0X04,0X08,0X10,0X20,0X40,0X80,

0X01,0X02,0X04,0X08,0X10,0X20,0X20,0X40,

0X01,0X02,0X04,0X08,0X08,0X10,0X20,0X20,

0X01,0X02,0X04,0X04,0X08,0X08,0X10,0X10,

0X01,0X01,0X02,0X02,0X04,0X04,0X08,0X08,

0X01,0X01,0X01,0X02,0X02,0X02,0X04,0X04,

0X01,0X01,0X01,0X01,0X02,0X02,0X02,0X02,

0X01,0X01,0X01,0X01,0X01,0X01,0X01,0X01,//35

0X02,0X02,0X02,0X02,0X01,0X01,0X01,0X01,//36

0X04,0X04,0X02,0X02,0X02,0X01,0X01,0X01,

0X08,0X08,0X04,0X04,0X02,0X02,0X01,0X01,

0X10,0X10,0X08,0X08,0X04,0X04,0X02,0X01,

0X20,0X10,0X00,0X08,0X08,0X04,0X02,0X01,

0X40,0X20,0X10,0X08,0X08,0X04,0X02,0X01,

0X80,0X40,0X20,0X10,0X08,0X04,0X02,0X01,

0X00,0X80,0X40,0X20,0X18,0X04,0X02,0X01,

0X00,0X00,0X80,0X40,0X38,0X04,0X02,0X01,

0X00,0X00,0X00,0X80,0X60,0X1C,0X02,0X01,

0X00,0X00,0X00,0X00,0X80,0X70,0X0E,0X01,

0X00,0X00,0X00,0X00,0X00,0XC0,0X3C,0X03,

0X00,0X00,0X00,0X00,0X00,0X00,0XF0,0X0F,

0X00,0X00,0X00,0X00,0X00,0X00,0X00,0XFF,//49

0X00,0X00,0X00,0X00,0X00,0X00,0X0F,0XF0,//50

0X00,0X00,0X00,0X00,0X00,0X03,0X1C,0XE0,

0X00,0X00,0X00,0X00,0X03,0X0C,0X30,0XC0,

0X00,0X00,0X00,0X01,0X06,0X18,0X20,0XC0,

0X00,0X00,0X01,0X02,0X0C,0X30,0X40,0X80,

0X00,0X01,0X02,0X04,0X18,0X20,0X40,0X80,//56

};

//面中心旋转,empty=1表示空心旋转,否则表示实心旋转,Clockwise=1表示顺时针旋转，否则表示逆时针旋转

//turns执行次数； speed速度

void LightCube::RotateFaceC(s8 num,s8 v)

{

u8 a[64]={0};

s8 i,j,k;

while(num--)

for(i=0;i<56;i++)

{

for(j=0;j<8;j++)

for(k=0;k<8;k++)

a[j\*8+k]=tab\_xuanzhuantiao2[i\*8+k];

frame(a,v);

}

}

//曲面旋转;Clockwise=1表示逆时针旋转,否则表示顺时针旋转

void LightCube::RotateHookFace(s8 Clockwise, s8 turns, u8 speed)

{

u8 b[64] = {0};

s8 i, x, z;

for(z = 0;z < 8; z++)

for(x = 0;x < 8; x++)

b[z \* 8 + x] = rotate\_tab[x];

frame(b, 1);

while(turns--) //主循环

{

if(Clockwise == 1)

for(i = 13; i >= 0; i--)

{

moveXYZ(b, Z\_AXIS, PULS, 1);

for(x = 0; x < 8; x++)

b[x] = rotate\_tab[i \* 8 + x];

frame(b, speed);

}

else

for(i = 0; i < 14; i++)

{

moveXYZ(b, Z\_AXIS, PULS, 1);

for(x = 0; x < 8; x++)

b[x] = rotate\_tab[i \* 8 + x];

frame(b, speed);

}

}

for(i = 0;i < 7;i++)

{

moveXYZ(b, Z\_AXIS, PULS, 1);

for(x = 0; x < 8; x++)

b[x] = rotate\_tab[x];

frame(b, speed);

}

}

//沙漏

const s8 sandglass\_tab00[]={ //从下往上

0,0,0,0,0,0,0,

0,1,2,3,4,5,6,

7,7,7,7,7,7,7,

7,6,5,4,3,2,1,//28

1\*8+1,1\*8+1,1\*8+1,1\*8+1,1\*8+1,

1\*8+1,1\*8+2,1\*8+3,1\*8+4,1\*8+5,

1\*8+6,1\*8+6,1\*8+6,1\*8+6,1\*8+6,

1\*8+6,1\*8+5,1\*8+4,1\*8+3,1\*8+2, //20

2\*8+2,2\*8+2,2\*8+2,

2\*8+2,2\*8+3,2\*8+4,

2\*8+5,2\*8+5,2\*8+5,

2\*8+5,2\*8+4,2\*8+3, //12

3\*8+3,3\*8+3,3\*8+4,3\*8+4, //4+4

4\*8+3,4\*8+3,4\*8+4,4\*8+4,

5\*8+2,5\*8+2,5\*8+2,

5\*8+2,5\*8+3,5\*8+4,

5\*8+5,5\*8+5,5\*8+5,

5\*8+5,5\*8+4,5\*8+3, //12

6\*8+1,6\*8+1,6\*8+1,6\*8+1,6\*8+1,

6\*8+1,6\*8+2,6\*8+3,6\*8+4,6\*8+5,

6\*8+6,6\*8+6,6\*8+6,6\*8+6,6\*8+6,

6\*8+6,6\*8+5,6\*8+4,6\*8+3,6\*8+2, //20

56+0,56+0,56+0,56+0,56+0,56+0,56+0,

56+0,56+1,56+2,56+3,56+4,56+5,56+6,

56+7,56+7,56+7,56+7,56+7,56+7,56+7,

56+7,56+6,56+5,56+4,56+3,56+2,56+1, //28

};

const s8 sandglass\_tab10[]={ //从上往下

56+0,56+0,56+0,56+0,56+0,56+0,56+0,

56+0,56+1,56+2,56+3,56+4,56+5,56+6,

56+7,56+7,56+7,56+7,56+7,56+7,56+7,

56+7,56+6,56+5,56+4,56+3,56+2,56+1, //28

6\*8+1,6\*8+1,6\*8+1,6\*8+1,6\*8+1,

6\*8+1,6\*8+2,6\*8+3,6\*8+4,6\*8+5,

6\*8+6,6\*8+6,6\*8+6,6\*8+6,6\*8+6,

6\*8+6,6\*8+5,6\*8+4,6\*8+3,6\*8+2, //20

5\*8+2,5\*8+2,5\*8+2,

5\*8+2,5\*8+3,5\*8+4,

5\*8+5,5\*8+5,5\*8+5,

5\*8+5,5\*8+4,5\*8+3, //12

4\*8+3,4\*8+3,4\*8+4,4\*8+4,

3\*8+3,3\*8+3,3\*8+4,3\*8+4, //4+4

2\*8+2,2\*8+2,2\*8+2,

2\*8+2,2\*8+3,2\*8+4,

2\*8+5,2\*8+5,2\*8+5,

2\*8+5,2\*8+4,2\*8+3, //12

1\*8+1,1\*8+1,1\*8+1,1\*8+1,1\*8+1,

1\*8+1,1\*8+2,1\*8+3,1\*8+4,1\*8+5,

1\*8+6,1\*8+6,1\*8+6,1\*8+6,1\*8+6,

1\*8+6,1\*8+5,1\*8+4,1\*8+3,1\*8+2, //20

0,0,0,0,0,0,0,

0,1,2,3,4,5,6,

7,7,7,7,7,7,7,

7,6,5,4,3,2,1, //28

};

//移动距离

const s8 sandglass\_tab01[]={//移动距离

0,1,2,3,4,5,6,

7,7,7,7,7,7,7,

7,6,5,4,3,2,1,

0,0,0,0,0,0,0, //28

1,2,3,4,5,

6,6,6,6,6,

6,5,4,3,2,

1,1,1,1,1, //20

2,3,4,

5,5,5,

5,4,3,

2,2,2, //12

3,4,4,3,

3,4,4,3, //4+4

2,3,4,

5,5,5,

5,4,3,

2,2,2, //12

1,2,3,4,5,

6,6,6,6,6,

6,5,4,3,2,

1,1,1,1,1, //20

0,1,2,3,4,5,6,

7,7,7,7,7,7,7,

7,6,5,4,3,2,1,

0,0,0,0,0,0,0, //28

};

//沙漏

void LightCube::Sandglass(s8 v)

{

u8 b[64]={0};

u16 i;

for(i=0;i<128;i++) //点

{

b[sandglass\_tab00[i]]=(0x01<<sandglass\_tab01[i]);

frame(b,v);

b[sandglass\_tab00[i]]=0;

}

for(i=0;i<128;i++)//8点

{

b[sandglass\_tab10[i]]|=(0x01<<sandglass\_tab01[i]);

if(i>=8)

b[sandglass\_tab10[i-8]]^=(0x01<<sandglass\_tab01[i-8]);//^表示异或

frame(b,v);

}

b[7]|=0x01;

b[0]=0x01;

frame(b,v);

for(i=1;i<128;i++) //线

{

if(i<8)

b[8-i]=0;

b[sandglass\_tab00[i]]|=(0x01<<sandglass\_tab01[i]);

frame(b,v);

}

frame(b,100);

for(i=0;i<128;i++) //线

{

b[sandglass\_tab10[i]]^=(0x01<<sandglass\_tab01[i]);//^表示异或

frame(b,v);

}

}

const u16 water\_tab[]=

{

0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0080,0x0140,0x0080,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,

0x0000,0x0000,0x0000,0x0000,0x0000,0x0080,0x0140,0x0220,0x0140,0x0080,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,

0x0000,0x0000,0x0000,0x0000,0x01c0,0x0220,0x0410,0x0410,0x0410,0x0220,0x01c0,0x0000,0x0000,0x0000,0x0000,0x0000,

0x0000,0x0000,0x0000,0x03e0,0x0410,0x0808,0x0808,0x0808,0x0808,0x0808,0x0410,0x03e0,0x0000,0x0000,0x0000,0x0000,

0x0000,0x0000,0x07f0,0x0808,0x1004,0x1004,0x1004,0x1004,0x1004,0x1004,0x1004,0x0808,0x07f0,0x0000,0x0000,0x0000,

0x0000,0x0ff8,0x1004,0x2002,0x2002,0x2002,0x2002,0x2002,0x2002,0x2002,0x2002,0x2002,0x1004,0x0ff8,0x0000,0x0000,

0x1ffc,0x2002,0x4001,0x4001,0x4001,0x4001,0x4001,0x4001,0x4001,0x4001,0x4001,0x4001,0x4001,0x2002,0x1ffc,0x0000

};

//一个水滴

void LightCube::WaterOne(s8 x, s8 y, u8 speed)

{

u8 b[64]={0};

s8 i, r;

b[0 \* 8 + x] = 0x01<<y;

frame(b, speed);

for(i = 0; i < 7; i++) //下落

{

b[(i + 1) \* 8 + x] = b[i \* 8 + x];

b[i \* 8 + x] = 0;

frame(b, speed);

}

for(r= 0; r < 7; r++)

{

for(i = 0; i < 8; i++)

b[ 7 \* 8 + i] = water\_tab[r \* 16 + 7 - x + i] >> (7 - y);

frame(b, speed + 5);

}//扩散

}

//两个水滴

void LightCube::WaterTwo(s8 x1, s8 y1, s8 x2, s8 y2, u8 speed)

{

u8 b[64] = {0};

s8 i, r;

b[0 \* 8 + x1] = 0x01 << y1;

frame(b, speed);

for(i = 0; i < 7; i++)

{

moveXYZ(b, Z\_AXIS, PULS, 1);

if(i == 1)

{

b[0 \* 8 + x2] = 0x01<<y2;

}

frame(b, speed);

}

for(r = 0; r < 9; r++)

{

if(r < 2)

moveXYZ(b, Z\_AXIS, PULS,1);

else

for(i = 0; i < 8; i++)

b[7 \* 8 + i]=water\_tab[(r - 2) \* 16 + 7 - x2 + i] >> (7 - y2);

if(r < 7)

for(i = 0; i < 8; i++)

b[7 \* 8 + i]|=(water\_tab[r \* 16 + 7 - x1 + i] >> (7 - y1));

frame(b, speed + 5);

}

}

//三个水滴

void LightCube::WaterThr(s8 x1, s8 y1, s8 x2, s8 y2, s8 x3, s8 y3, u8 speed)

{

u8 b[64] = {0};

s8 i, r;

b[0 \* 8 + x1] = 0x01 << y1;

frame(b, speed);

for(i = 0;i < 7; i++)

{

moveXYZ(b, Z\_AXIS, PULS, 1);

if(i == 1)//第二点

b[0 \* 8 + x2] = 0x01 << y2;

if(i == 3)

b[0 \* 8 + x3] = 0x01 << y3; //第三点

frame(b, speed);

}

for(r = 0; r < 11; r++)//扩散

{

if(r < 4)

moveXYZ(b, Z\_AXIS, PULS, 1);

if(r < 11 && r >= 4)

for(i = 0; i < 8; i++)

b[7 \* 8 + i] = water\_tab[(r - 4) \* 16 + 7 - x3 + i] >> (7 - y3);

if(r<9 && r>=2)

for(i = 0; i<8; i++)

b[7 \* 8 + i] |= water\_tab[(r - 2) \* 16 + 7 - x2 + i] >> (7 - y2);

if(r < 7)

for(i = 0; i<8; i++)

b[7 \* 8 + i] |= (water\_tab[r \* 16 + 7 - x1 + i] >> (7 - y1));

frame(b, speed + 5);

}

}

//动画驱动函数Animation，)//num表示帧数，n表示循环次数

void LightCube::Animation(const u8 \*ARRAY\_tab,u8 frame\_num,u8 cycle\_index,u8 speed)

{

s8 i, j;

u8 b[64]={0};

for(j = 0; j<64; j++)

b[j] = ARRAY\_tab[j] ;

frame(b,speed);

while(cycle\_index--)

{

for(i=1;i<frame\_num;i++)

{

for(j = 0; j<64; j++)

b[j] = ARRAY\_tab[i\*64 + j] ;

frame(b,speed);

}

}

}

void LightCube::PrintOpen(u8 speed)

{

u8 x, y;

u8 b[64] = {0};

for(y = 0; y<8; y++)

{

b[0] |= 0x01<<y;

frame(b,speed-3);

}

for(x = 1; x<8;x++)

{

b[x]=0xff;

frame(b,speed);

}

}

//从上向下打印

void LightCube::Print(const u8 \*ARRAY\_tab)

{

u8 p;

s8 x, y, z;

u8 b[64] = {0}, c[8] = {0};

for(x = 0; x<8; x++)

b[x] = 0xff;

for(x = 0; x<8; x++) //行切换

{

c[x] = ARRAY\_tab[x];

for(y = 0; y<8; y++) //点的确定

{

p = c[x] & (0x01<<y);

if(p)

{

for(z = 1; z<7; z++)

{

if(z==1)

{

b[8 \* z + x] = p;

frame(b, 1);

}

else

{

b[8 \* (z - 1) + x] = 0;

b[8 \* z + x] = p;

frame(b, 1);

}

}

b[48 + x] = 0;

b[56 + x] |= p;

frame(b, 4);

}

}

}

frame(b, 40);

for(y = 0; y<8; y++)

{

for(x = 0; x<8; x++)

b[56 + x] = b[56 + x]>>1;

frame(b, 10);

}

}

const u8 fan\_tab[]=

{

0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80, //0

0x40,0x40,0x40,0x40,0x80,0x80,0x80,0x80,

0x20,0x20,0x40,0x40,0x40,0x40,0x80,0x80,

0x10,0x10,0x20,0x20,0x40,0x40,0x80,0x80,

0x08,0x08,0x10,0x20,0x20,0x40,0x80,0x80,

0x04,0x08,0x08,0x10,0x20,0x40,0x40,0x80,

0x02,0x04,0x08,0x10,0x10,0x20,0x40,0x80,

0x01,0x02,0x04,0x08,0x10,0x20,0x40,0x80, //7

0x00,0x01,0x02,0x04,0x18,0x20,0x40,0x80,

0x00,0x00,0x01,0x06,0x08,0x10,0x60,0x80,

0x00,0x00,0x00,0x01,0x06,0x18,0x60,0x80,

0x00,0x00,0x00,0x00,0x03,0x0c,0x30,0xc0,

0x00,0x00,0x00,0x00,0x00,0x03,0x3c,0xc0,

0x00,0x00,0x00,0x00,0x00,0x00,0x0f,0xf0,

0x00,0x00,0x00,0x00,0x00,0x00,0x00,0xff, //14

};

const u8 connect\_one\_tab[]=

{

0XFF,0X81,0X81,0X81,0X81,0X81,0X81,0XFF,

0XFF,0XFF,0Xc3,0Xc3,0Xc3,0Xc3,0XFF,0XFF,

0XFF,0XFF,0XFF,0XE7,0XE7,0XFF,0XFF,0XFF,

0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,

0X7F,0XFF,0XFF,0XFF,0XFF,0XFF,0XFF,0XFE,

0X3F,0X7F,0XFF,0XFF,0XFF,0XFF,0XFE,0XFC,

0X1F,0X3F,0X7F,0XFF,0XFF,0XFE,0XFC,0XF8,

0X0F,0X1F,0X3F,0X7E,0XFE,0XFC,0XF8,0XF0,

0X07,0X0F,0X1F,0X3E,0X7C,0XF8,0XF0,0XE0,

0X03,0X07,0X0E,0X1C,0X38,0X70,0XE0,0XC0,

0X01,0X02,0X04,0X08,0X10,0X20,0X40,0X80,//10

};

void LightCube::ConnectThree(u8 speed)

{

u8 b[64] = {0};

u8 i, x, z;

for(z = 0; z < 8; z++)

for(x = 0; x < 8; x++)

b[z\*8 + x] = 0X80;

frame(b, 2);

for(i = 0; i<3; i++)

{

for(x = 0; x < 8; x++)

{

b[i\*8 + x] = 0X00;

b[(7-i)\*8 + x] = 0X00;

}

frame(b, speed);

}

for(x = 0; x < 8; x++)

b[4 \* 8 + x] = 0X00;

frame(b, speed);

for(x = 0; x < 3; x++)

{

b[3 \* 8 + x] = 0X00;

b[3 \* 8 + (7 - x)] = 0X00;

frame(b, speed);

}

b[3 \* 8 + 3] = 0X00;

frame(b, speed);

b[3 \* 8 + 4] = 0X00;

frame(b, speed + 10);

}

void LightCube::ConnectFour(u8 speed)

{

u8 b[64] = {0};

u8 x, z;

s8 i = 7, j;

for(z = 0; z < 8; z++)

for(x = 0; x < 8; x++)

b[z\*8 + x] = rain\_tab[z\*8+x];

frame(b, 1);

while(i--)

{

for(x = 0; x < 8; x++)

b[x] = (b[x] | b[8 + x]);

for(z = 1; z < 8; z++)

for(x = 0; x < 8; x++)

b[z\*8 + x] = b[(z + 1) \* 8 + x];

for(x = 0; x < 8; x++)

b[56 + x] = 0x00;

frame(b, speed);

}

for(j = 0; j < 8; j++)

{

for(x = 0; x < 8; x++)

b[x] |= 0x01<<j;

frame(b, speed);

}

for(j = 0 ;j < 13; j++)

{

if(j < 7)

for(x = 0;x < 8; x++)

{

b[(j + 1) \* 8 + x] = 0x80;

b[x] <<= 1;

}

else

for(x = 0 ; x < 8 ; x++)

{

b[ 7 \* 8 + x] |= 0x80>>(j-6);

b[( j - 7 ) \* 8 + x] = 0;

}

frame(b, speed);

}

}

void LightCube::ConnectFive(u8 speed)

{

u8 b[64] = {0};

u8 i;

s8 x;

for(x = 0;x < 8; x++)

b[7 \* 8 + x] = 0xFF;

for(i = 0 ;i < 7; i++)

{

for(x = 0;x < 8; x++)

{

b[(6-i) \* 8 + x] = 0x80;

b[56 + x] <<= 1;

}

frame(b, speed);

}

}

//frame\_num表示帧数，cycle\_index表示循环次数，speed表示速度

void LightCube::ConnectSix(const u8 \*ARRAY\_tab,u8 frame\_num,u8 cycle\_index,u8 speed)

{

s8 i, j;

u8 b[64]={0};

for(j = 0; j<64; j++)

b[j] = 0x00 ;

while(cycle\_index--)

{

for(i=(frame\_num - 1);i >= 0;i--)

{

for(j = 0; j<64; j++)

b[j] = ARRAY\_tab[i\*64 + j] ;

frame(b,speed);

}

}

for(i = 0; i < 7;i++)

{

moveXYZ(b, X\_AXIS, MINUS, 1);

moveXYZ(b, Y\_AXIS, MINUS, 1);

moveXYZ(b, Z\_AXIS, MINUS, 1);

frame(b,5);

}

}

void LightCube::ConnectSeven(u8 speed)

{

u8 b[64] = {0};

s8 i, x;

for(x = 0; x < 8; x++)

b[x] = fan\_tab[56 + x];

for(i = 6; i >= 0; i--)

{

moveXYZ(b, Z\_AXIS, PULS, 1);

for(x = 0; x < 8; x++)

b[x] = fan\_tab[i \* 8 + x];

frame(b, speed);

}

for(i = 0; i < 7; i++)

{

moveXYZ(b, Z\_AXIS, PULS, 1);

for(x = 0; x < 8; x++)

b[x] = fan\_tab[x];

frame(b, speed);

}

}

#include "Array.h" // 动画数组

#include "LightCube.h"

LightCube lightCube(5, 6, 7); // SER0 , SCK0 , RCK0

void setup() {

Serial.begin(9600);

lightCube.AllOff(); // 全部关闭

}

void loop() {

while(Serial.available())

{

char c=Serial.read();

if(c=='c')//面旋转

{

lightCube.ConnectSeven(6); // 连接动画7 参数：速度

lightCube.FanOne(2, 5); // 参数：速度

lightCube.FanTwo(2, 5); // 参数：速度

lightCube.RotateFace(1, 1, 4, 8); //面中心旋转，参数：第一个参数可为1/0，表示空心/实心、第二个参数可为1/0表示正/反转、第三个参数表示旋转次数、第四个参数表示速度

lightCube.RotateHookFace(0, 4, 5); //曲面中心旋转，参数：第一个参数可为1/0，表示顺时/逆时针、第三个参数表示旋转次数、第四个参数表示速度

lightCube.RotateHookFace(1, 4, 5); //曲面旋转

lightCube.RotateFaceC(1, 10);//面侧边旋转，参数：第一个是旋转次数、第二个是速度

lightCube.AllOff(); // 全部关闭

Serial.print(" Over");

}

else if(c=='b')//沙漏

{

lightCube.Sandglass(2); //沙漏 参数：速度

lightCube.PrintOpen(10);

Serial.print(" Over");

}

else if(c=='g')//I Love You

{

for (int num = 0; num < 4; num++)

{

lightCube.Print(print\_tab + 8 \* num);

}

Serial.print(" Over");

}

else if(c=='e')//面扫描

{

lightCube.UtoDScan(10); //由上至下依次点亮，参数：速度

lightCube.Up(1, 10); //由上而下扫描，由下而上依次点亮 参数：第一个参数表示循环次数、第二个参数表示速度

lightCube.DtoUScan(10); //从下向上依次点亮，参数：速度

lightCube.LtoRScan(10); //侧面从左向右依次点亮，参数：速度

lightCube.RtoLScan(10); //侧面从右向左依次点亮，参数：速度

lightCube.FtoBScan(10); //从前排向后排依次点亮，参数：速度

lightCube.BtoFScan(10); //从后排向前排依次点亮，参数：速度

lightCube.AllOff(); // 全部关闭

Serial.print(" Over");

}

else if(c=='a')//烟花

{

lightCube.WaterOne(4, 5, 10); //第一、第二个参数表示烟花1的x轴、y轴坐标值，第三个参数表示速度

lightCube.WaterTwo(2, 3, 6, 4, 10); //第一、第二个参数表示烟花1的x轴、y轴坐标值，第三、第四个参数表示烟花2的x轴、y轴坐标值，第五个参数表示速度

lightCube.WaterThr(1, 4, 2, 6, 5, 7, 10); //第一、第二个参数表示烟花1的x轴、y轴坐标值，第三、第四个参数表示烟花2的x轴、y轴坐标值，第五、第六个参数表示烟花3的x轴、y轴坐标值，第七个参数表示速度

lightCube.AllOff(); // 全部关闭

Serial.print(" Over");

}

else if(c=='d')//雨水

{

lightCube.Rain(0, 5, 6); //雨水动画 参数：第一个参数可为0/1表示上/下、第二个参数表示循环次数、第三个参数表示速度

lightCube.Rain(1, 5, 6); //雨水动画

lightCube.AllOff(); // 全部关闭

Serial.print(" Over");

}

else if(c=='h')//周旋

{

lightCube.ConnectFour(7); // 连接动画4，参数：速度

lightCube.RightClockwise(1, 5); //从右侧看顺时针旋转，参数：第一个参数表示旋转周数、第二个参数表示速度

lightCube.ConnectFive(10); // 连接动画5，参数：速度

lightCube.Anticlockwise(1, 5); //anticlockwise逆时针旋转参数：第一个参数表示旋转周数、第二个参数表示速度

lightCube.Clockwise(2, 5); //clockwise表示顺时针旋转参数：第一个参数表示旋转周数、第二个参数表示速度

Serial.print(" Over");

}

else if(c=='f')//立方收缩

{

lightCube.ConnectThree(5);// 连接动画3 ，参数：速度

lightCube.Cube(0, 0, 6);//立方体动画，参数：第一个参数可为0/1 实/空，第二个参数可为0 /1、 /2/3，分别表示左上角/右上角/左下角/右下角，第三个参数表示速度

lightCube.Cube(1, 3, 10); //立方体动画

lightCube.Cube(0, 2, 10); //立方体动画

Serial.print(" Over");

}

else

{

lightCube.AllOn(50); // 全部打开，参数：速度

lightCube.AllOff(); // 全部关闭

Serial.print(" Over");

}

}

}

### 2.2.2 HC-05蓝牙模块

#### 2.相关代码

//HC-05蓝牙模块进入AT模式，代码如下：

#include <SoftwareSerial.h>

// 引脚10为RX，接HC-05模块的的TXD

// 引脚11为TX，接HC-05模块的的RXD

SoftwareSerial BT(10, 11);

char val;

void setup() {

Serial.begin(38400);

Serial.println("BT is ready!");

// HC-05默认，38400

BT.begin(38400);

}

void loop() {

if (Serial.available()) {

val = Serial.read();

BT.print(val);

}

if (BT.available()) {

val = BT.read();

Serial.print(val);

}

}

/

### 2.2.3 音乐频谱模块

#### 2.相关代码

#define u8 unsigned char

#define s8 signed char

#define u16 unsigned short

#define DEBUG 0 // 调试参数

#define DEBUG\_OUTFFT 1 // 调试参数

#define LOG\_OUT 1 // 使用日志输出函数

#define FFT\_N 256 //设置FFT点数为256

#define COU4 4 // FFT取值

#define SHOW\_MODE 0 // 显示模式

#include<FFT.h>

const int latchPin = 7;//RCK

const int clockPin = 6;//SCK

const int dataPin = 5;//SER

const int LightCube\_col = 8; // 显示列数量

const int LightCube\_row = 8; // 每列显示分辨率

const int samples = 1; // 采样次

int allOn[64] = {}; //每帧画面数据

u8 al[64] = {0};

int fft\_data[8] = {};

const int fftData\_noise[8] = {24, 10, 9, 7, 9, 8, 8, 5};

// 根据DEBUG\_OUTFFT 参数调试输出的噪声值（无音频输入时的值）

void setup()

{

pinMode(latchPin, OUTPUT);

pinMode(clockPin, OUTPUT);

pinMode(dataPin, OUTPUT);

TIMSK0 = 0;

ADCSRA = 0xe5; // 设置模数转换为自由模式

ADMUX = 0x40; // 使用模数转换

DIDR0 = 0x01; // 关掉模数转换的数字输入

#if FASTADC

sbi(ADCSRA, ADPS2)

cbi(ADCSRA, ADPS1)

cbi(ADCSRA, ADPS0)

#endif

Serial.begin(115200);

if (DEBUG)

Serial.println("Begin...");

delay(100);

}

void loop()

{

switch (SHOW\_MODE) {

case 0:

cube0(al, getFFT() , COU4);

break;

case 1:

cube1(al, getFFT() , COU4);

break;

case 2:

cub1(al, getFFT() , COU4);

break;

case 3:

all(al, getFFT() , COU4);

break;

}

}

void storey(u8 \*a) //层填充函数，控制某层灯点亮方式

{

u8 i, j, num;

for (i = 0; i < 8; i++)

{

num = a[i]; //将数组中数输入寄存器

for (j = 0; j < 8; j++) //串行数据输入

{

if (num & 0x80)

digitalWrite(dataPin, HIGH); // SER串行输入端口

else

digitalWrite(dataPin, LOW);

digitalWrite(clockPin, LOW); //上升沿，输入到移位寄存器

delayMicroseconds(1);

digitalWrite(clockPin, HIGH);

num <<= 1;

}

}

}

void frame(u8 \*a) //表示一帧,a是一帧编码起始地址

// v表示一帧画面扫描的次数

// 可以看作这帧显示的时间

{

s8 i, j, num; //s8 有符号定义

num = 0x01;

for (i = 0; i < 8; i++)//层数控制

{

num <<= i;

digitalWrite(latchPin, LOW);

for (j = 0; j < 8; j++) {//串行数据输入

digitalWrite(dataPin, LOW);

delayMicroseconds(1);

digitalWrite(clockPin, LOW); //上升沿，输入到移位寄存器

delayMicroseconds(1);

digitalWrite(clockPin, HIGH);

}

for (j = 0; j < 8; j++) //串行数据输入

{

if (num & 0x80)

digitalWrite(dataPin, HIGH); // SER串行输入端口

else

digitalWrite(dataPin, LOW);

digitalWrite(clockPin, LOW); //上升沿，输入到移位寄存器

delayMicroseconds(1);

digitalWrite(clockPin, HIGH);

num <<= 1;

}

storey(a + i \* 8); //层填充函数，控制某层灯点亮方式

digitalWrite(latchPin, HIGH);

num = 0x01;

delayMicroseconds(5); // 层显示时间

}

}

void frame(u8 \*a, s8 v) //表示一帧,a是一帧编码起始地址

// v表示一帧画面扫描的次数

// 可以看作这帧显示的时间

{

s8 i, j, num; //s8 有符号定义

while (v--)

{

num = 0x01;

for (i = 0; i < 8; i++) //层数控制

{

num <<= i;

digitalWrite(latchPin, LOW);

for (j = 0; j < 8; j++) //串行数据输入

{

digitalWrite(dataPin, LOW);

delayMicroseconds(1);

digitalWrite(clockPin, LOW); //上升沿，输入到移位寄存器

delayMicroseconds(1);

digitalWrite(clockPin, HIGH);

}

for (j = 0; j < 8; j++) //串行数据输入

{

if (num & 0x80)

digitalWrite(dataPin, HIGH); // SER串行输入端口

else

digitalWrite(dataPin, LOW);

digitalWrite(clockPin, LOW); //上升沿，输入到移位寄存器

delayMicroseconds(1);

digitalWrite(clockPin, HIGH);

num <<= 1;

}

storey(a + i \* 8); //层填充函数，控制某层灯点亮方式

digitalWrite(latchPin, HIGH);

num = 0x01;

delayMicroseconds(5); // 层显示时间

}

}

}

void cube0(u8 \*a, s8 n[8] , int c) //正方体，外轮廓0<n<=8,注意n不能为0

{

s8 i, j , k;

n[c] /= samples; // 采样取平均

n[c] -= fftData\_noise[c]; // 去噪声

k = n[c];

Serial.print(k);

Serial.print("/");

if (k > LightCube\_row) // 大于最大分辨率

{

k = LightCube\_row;

}

if (!(k > 0)) // 值小

{

k = 0;

}

Serial.print(k);

Serial.println(" ");

for (i = 0; i < 64; i++)

a[i] = 0;

j = 0xff >> (8 - k);

a[0] = j;

a[k - 1] = j;

a[(k - 1) \* 8] = j;

a[(k - 1) \* 8 + k - 1] = j;

for (i = 0; i < k; i++)

{

j = (0x01 | (0x01 << (k - 1)));

a[i \* 8] |= j;

a[i \* 8 + k - 1] |= j;

a[i] |= j;

a[(k - 1) \* 8 + i] |= j;

}

frame(a);

}

void cube1(u8 \*a , s8 n[8] , int c) //实心正方体0<=n<=8

{

s8 x, z, k;

n[c] /= samples; //采样取平均

n[c] -= fftData\_noise[c]; //去噪声

k = n[c];

Serial.print(k);

Serial.print("/");

if (k > LightCube\_row) //大于最大分辨率

{

k = LightCube\_row;

}

if (!(k > 0)) //值小

{

k = 0;

}

Serial.print(k);

Serial.println(" ");

for (z = 0; z < 8; z++)

for (x = 0; x < 8; x++)

{

if (z < k && x < k)

a[z \* 8 + x ] = 0xff >> (8 - k);

else

a[z \* 8 + x ] = 0;

}

frame(a);

}

void cub1(u8 \*a , s8 n[8] , int c) //实心正方体由下向上0<=n<=8

{

s8 x, z, k;

n[c] /= samples; //采样取平均

n[c] -= fftData\_noise[c]; //去噪声

k = n[c];

Serial.print(k);

Serial.print("/");

if (k > LightCube\_row) //大于最大分辨率

{

k = LightCube\_row;

}

if (!(k > 0)) // 值小

{

k = 0;

}

Serial.print(k);

Serial.println(" ");

for (z = 0; z < 8; z++)

{

for (x = 8 - k; x < 8; x++) {

a[x \* 8 + z] = 0xff;

}

for (x = 0; x < 8 - k; x++)

{

a[x \* 8 + z] = 0x0;

}

}

frame(a, 3);

}

void all(u8 \*a , s8 n[8] , int c) // 0<=n<=8 、

{

s8 x, z, k;

n[c] /= samples; //采样取平均

n[c] -= fftData\_noise[c]; //去噪声

k = n[c];

Serial.print(k);

Serial.print("/");

if (k > LightCube\_row) //大于最大分辨率

{

k = LightCube\_row;

}

if (!(k > 0)) //值小

{

k = 0;

}

Serial.print(k);

Serial.println(" ");

for (z = 0; z < 8; z++)

{

for (x = 8 - k; x < 8; x++)

{

a[x \* 8 + z] = 0xff;

}

for (x = 0; x < 8 - k; x++)

{

a[x \* 8 + z] = 0x0;

}

}

frame(a);

}

int getFFT()

{

memset(fft\_data, 0, sizeof(fft\_data) / sizeof(int));

for (int c = 0; c < samples; c++) //频率采集

{

cli();

for (int i = 0 ; i < FFT\_N \* 2 ; i += 2)

{

while (!(ADCSRA & 0x10)); //等待ADC做好准备

ADCSRA = 0xf5; //重启ADC

byte m = ADCL; //获取ADC数据

byte j = ADCH;

int k = (j << 8) | m;

k -= 0x0200;

k <<= 6;

fft\_input[i] = k; //将真实数据放入偶数箱

fft\_input[i + 1] = 0; //将奇数箱设置为0

}

fft\_window(); //窗口数据更好的频率响应

fft\_reorder(); //在执行FFT之前重新排序数据

fft\_run(); //处理FFT中的数据

fft\_mag\_log(); //取FFT的输出

sei();

if (DEBUG)

{

Serial.println("start");

for (byte f = 0 ; f < FFT\_N / 2 ; f++)

{

Serial.print(fft\_log\_out[f]); //输出数据

Serial.print(" ");

}

Serial.println();

}

for (int n = 0; n < LightCube\_col; n++)

{

int a\_val = 0;

for (byte i = n \* 16 ; i < (n + 1) \* 16 ; i++)//128个数据，8个输出灯柱

//故（128/8）16个数据取平均值

{

a\_val += fft\_log\_out[i];

}

a\_val /= 16; // 取平均值 ave/16

a\_val /= 2.5; //根据得到的值或显示效果，适当更改该值为3

fft\_data[n] += a\_val; //为消除噪声取平均值，将本次的数据放入平均值数组中对应的位置

}

}

return fft\_data;

}

### 2.2.4 输出模块

#### 2.相关代码

//蓝牙功能输出相关代码如下：

#include "Array.h" // 动画数组

#include "LightCube.h"

LightCube lightCube(5, 6, 7); // SER0 , SCK0 , RCK0

void setup() {

Serial.begin(9600);

lightCube.AllOff(); // 全部关闭

}

void loop() {

while(Serial.available())

{

char c=Serial.read();

if(c=='c')//面旋转

{

lightCube.ConnectSeven(6); // 连接动画7，参数：速度

lightCube.FanOne(2, 5); // 参数：速度

lightCube.FanTwo(2, 5); // 参数：速度

lightCube.RotateFace(1, 1, 4, 8); //面中心旋转 参数：第一个参数可为1/0表示空心/实心，第二个参数可为1/0表示正/反转，第三个参数表示旋转次数，第四个参数表示速度

lightCube.RotateHookFace(0, 4, 5); //曲面中心旋转 参数：第一个参数可为1/0 表示顺时/逆时针旋转、第二个参数表示旋转次数，第三个参数表示速度

lightCube.RotateHookFace(1, 4, 5); //曲面旋转

lightCube.RotateFaceC(1, 10); //面侧边旋转 参数：第一个参数表示旋转次数，第二个参数表示速度

lightCube.AllOff(); // 全部关闭

Serial.print(" Over");

}

else if(c=='b')//沙漏

{

lightCube.Sandglass(2); //沙漏，参数：速度

lightCube.PrintOpen(10);

Serial.print(" Over");

}

else if(c=='g')//I Love You

{

for (int num = 0; num < 4; num++)

{

lightCube.Print(print\_tab + 8 \* num);

}

Serial.print(" Over");

}

else if(c=='e')//面扫描

{

lightCube.UtoDScan(10); //由上至下依次点亮，参数：速度

lightCube.Up(1, 10); //由上而下扫描，由下而上一次点亮 参数：第一个参数表示旋转次数，第二个参数表示速度

lightCube.DtoUScan(10); //从下向上依次点亮，参数：速度

lightCube.LtoRScan(10); //侧面从左向右依次点亮，参数：速度

lightCube.RtoLScan(10); //侧面从右向左依次点亮，参数：速度

lightCube.FtoBScan(10); //从前排向后排依次点亮，参数：速度

lightCube.BtoFScan(10); //从后排向前排依次点亮，参数：速度

lightCube.AllOff(); // 全部关闭

Serial.print(" Over");

}

else if(c=='a')//烟花

{

lightCube.WaterOne(4, 5, 10); //烟花1，参数：第一、第二个参数表示烟花1的x轴、y轴坐标值，第三个参数表示速度

lightCube.WaterTwo(2, 3, 6, 4, 10); //烟花2，参数：第一、第二个参数表示烟花1的x轴、y轴坐标值，第三、第四个参数表示烟花2的x轴、y轴坐标值，第五个参数表示速度

lightCube.WaterThr(1, 4, 2, 6, 5, 7, 10); //烟花3，参数：第一、第二个参数表示烟花1的x轴、y轴坐标值，第三、第四个参数表示烟花2的x轴、y轴坐标值，第五、第六个参数表示烟花3的x轴、y轴坐标值，第七个参数表示速度

lightCube.AllOff(); // 全部关闭

Serial.print(" Over");

}

else if(c=='d')//雨水

{

lightCube.Rain(0, 5, 6); //雨水动画，参数：第一个参数可为0/1表示上/下，第二个参数可为循环次数，第三个参数表示速度

lightCube.Rain(1, 5, 6); //雨水动画

lightCube.AllOff(); // 全部关闭

Serial.print(" Over");

}

else if(c=='h')//周旋

{

lightCube.ConnectFour(7); // 连接动画4，参数：速度

lightCube.RightClockwise(1, 5); //右侧边顺时针旋转 参数：第二个参数表示旋转周数、第三个参数表示速度

lightCube.ConnectFive(10); // 连接动画5，参数：速度

lightCube.Anticlockwise(1, 5); //anticlockwise表示逆时针旋转 参数：第二个参数表示旋转周数、第三个参数表示速度

lightCube.Clockwise(2, 5); //clockwise表示顺时针旋转 参数：第二个参数表示旋转周数、第三个参数表示速度

Serial.print(" Over");

}

else if(c=='f')//立方收缩

{

lightCube.ConnectThree(5);// 连接动画3，参数：速度

lightCube.Cube(0, 0, 6);//立方体动画，参数：第一个参数可为0/1表示实/空、第二个参数可为0 /1 /2 /3，表示左上角/右上角、左下角、右下角，第三个参数表示速度

lightCube.Cube(1, 3, 10); //立方体动画

lightCube.Cube(0, 2, 10); //立方体动画

Serial.print(" Over");

}

else

{

lightCube.AllOn(50); // 全部打开，参数：速度

lightCube.AllOff(); // 全部关闭

Serial.print(" Over");

}

}

}

# 第3章 乐光宝盒项目设计

### 3.2.1 主程序模块

#### 2.相关代码

#include <SoftwareSerial.h>

#include <Adafruit\_NeoPixel.h> //彩色LED库<https://download.csdn.net/download/u014313945/9653501>

#define PIN 6 //设置LED引脚

#define MAX\_LED 60 //特定的程序，最后一个数字是LED光带中数量

//Adafruit\_NeoPixel strip = Adafruit\_NeoPixel( MAX\_LED, PIN, NEO\_RGB + NEO\_KHZ800 ); //彩色LED库的内部设置

int a = 2;// 将Arduino开发板的引脚2连接至US-100 的Echo/RX

int b = 3;// 将Arduino开发板的引脚3连接至US-100 的Trig/TX

#include"shining.h"

#include"tone.h"

#include"sky.h"

#include"superwave.h"

// 引脚10为RX，接HC-05的TXD

// 引脚11为TX，接HC-05的RXD

SoftwareSerial BT(10, 11);//R接蓝牙T,T接蓝牙R

char val;

void setup()

{

pinMode(a,INPUT);// 设置a为输入模式

pinMode(b,OUTPUT);// 设置b为输出模式

strip.begin(); //启动LED函数库

strip.show(); //使所有LED灭

void ledz();//使所有LED灭

void leda(); //使所有LED亮，并发出大红和红色的光

void ledb();//使全部LED发出淡蓝色和蓝绿色的光

void ledc();//使全部LED发出金黄色的光

void ledd();//使全部LED发出绿色的光

void lede();//使全部LED发出大红色的光

void ledf();//使全部LED发出湖蓝和淡蓝的光

void ledg();//使全部LED发出绿色和橘黄色的光

void ledh();//使全部LED发出淡蓝色的光

void ledi();//使全部LED发出湖蓝和浅蓝的光

void ledj();//使全部LED发出红色和大红色的光

ledy();//使全部LED发出深蓝、淡紫、红、橘黄、金黄、绿、蓝绿、天蓝、紫、大红、橘红、湖蓝、淡蓝、蓝、浅蓝色的光

Serial.begin(38400);// 设置波特率为38400

pinMode(9, OUTPUT);

BT.begin(38400);

}

void loop()

{

while (Serial.available()) {

val = Serial.read();

BT.write(val);

Serial.print(val);

}

while (BT.available()) {

val = BT.read();

Serial.print(val);

}

if(val=='b')

{

for (int thisNote = 0; thisNote < 80; thisNote++) {

int noteDuration = 1000 / noteDurations[thisNote];

tone(9, melody[thisNote], noteDuration);

int pauseBetweenNotes = noteDuration \* 1.30;

delay(pauseBetweenNotes);

noTone(9);

{

if(melody[thisNote]==NOTE\_A3) //当扬声器发出A3声调时，执行以下程序

{

ledz(); //使所有LED灭

leda(); //使所有LED亮，并发出大红和红色的光

}

if(melody[thisNote]==NOTE\_B3)

{

ledz();//使所有LED灭

ledb();//使全部LED发出淡蓝色和蓝绿色的光

}

if(melody[thisNote]==NOTE\_C3)

{

ledz();//使所有LED灭

ledc();//使全部LED发出金黄色的光

}

if(melody[thisNote]==NOTE\_D4)

{

ledz();//使所有LED灭

ledd();//使全部LED发出绿色的光

}

if(melody[thisNote]==NOTE\_E3)

{

ledz();//使所有LED灭

lede();//使全部LED发出大红色的光

}

if(melody[thisNote]==NOTE\_E4)

{

ledz();//使所有LED灭

ledf();//使全部LED发出湖蓝和淡蓝的光

}

if(melody[thisNote]==NOTE\_F3)

{

ledz();//使所有LED灭

ledg();//使全部LED发出绿色和橘黄色的光

}

if(melody[thisNote]==NOTE\_G3)

{

ledz();//使所有LED灭

ledh();//使全部LED发出淡蓝色的光

}

if(melody[thisNote]==NOTE\_FS3)

{

ledz();//使所有LED灭

ledi();//使全部LED发出湖蓝和浅蓝的光

}

if(melody[thisNote]==NOTE\_GS3)

{

ledz(); //使所有LED灭

ledj();//使全部LED发出红色和大红色的光

}

}

}

}

if(val=='a')

{

long c = 0;//脉冲时间

long d = 0;//脉冲距离

// 通过Trig/Pin 发送脉冲，触发US-100 测距

digitalWrite(b,LOW); // 先拉低，以确保脉冲识别正确

delayMicroseconds(2);// 等待2us

digitalWrite(b,HIGH);// 开始通过Trig/Pin 发送脉冲

delayMicroseconds(12);// 设置脉冲宽度为12us (>10us)

digitalWrite(b,LOW);// 结束脉冲

c = pulseIn(a,HIGH);// 计算US-100 返回的脉冲时间

d = c\*0.34/2; //距离=脉冲时间\*声波的速度（340/s）/2

if(d>50&&d<=150) //当脉冲距离>50mm并且<=150的时候，执行以下程序

{

ala(); //扬声器发出A4的音调

ledz(); //使所有LED灭

leda(); //使所有LED亮，并发出大红和红色的光

}

if(d>150&&d<=200)

{

asi(); //扬声器发出B4的音调

ledz();//使所有LED灭

ledb();//使全部LED发出淡蓝色和蓝绿色的光

}

if(d>200&&d<=250)

{

bdo(); //扬声器发出C5的音调

ledz();//使所有LED灭

ledc();//使全部LED发出金黄色的光

}

if(d>250&&d<=300)

{

bre(); //央视网哪个区发出D5的音调

ledz();//使所有LED灭

ledd();//使全部LED发出绿色的光

}

if(d>300&&d<=350)

{

bmi(); //扬声器发出E6的音调

ledz();//使所有LED灭

lede();//使全部LED发出大红色的光

}

if(d>350&&d<=400)

{

bfa(); //扬声器发出F5的音调

ledz();//使所有LED灯灭

ledf();//使全部LED发出湖蓝和淡蓝的光

}

if(d>400&&d<=450)

{

bso(); //扬声器发出G5的音调

ledz();//使所有LED灭

ledg();//使全部LED发出绿色和橘黄色的光

}

if(d>450&&d<=500)

{

bla(); //扬声器发出A5的音调

ledz();//使所有LED灭

ledh();//使全部LED发出淡蓝色的光

}

if(d>500&&d<=550)

{

bsi(); //扬声器发出B5的音调

ledz();//使所有LED灭

ledi();//使全部LED发出湖蓝和浅蓝的光

}

if(d>550&&d<=600)

{

cdo(); //扬声器发出C6的音调

ledz();//使所有LED灭

ledj();//使全部LED发出红色和大红色的光

}

if(d>600&&d<=800)

{

cre(); //扬声器发出D6的音调

ledz();//使所有LED灭

ledk();//使全部LED发出大红色和金黄色的光

}

if(d<=50)

{

noTone(9);

ledz();//使所有LED灭

ledy();//使全部LED发出深蓝、淡紫、红、橘黄、金黄、绿、蓝绿、天蓝、紫、大红、橘红、湖蓝、淡蓝、蓝、浅蓝色的光

}

if(d>800)

{

noTone(9);

ledz();//使所有LED灭

ledy();//使全部LED发出深蓝、淡紫、红、橘黄、金黄、绿、蓝绿、天蓝、紫、大红、橘红、湖蓝、淡蓝、蓝、浅蓝色的光

}

delay(400);// 每S测量2.5次

}

}

### 3.2.2 US-100模块

#### 2.相关代码

int a = 2;// 将Arduino开发板引脚2连接至US-100的Echo/RX

int b = 3;// 将Arduino开发板引脚3连接至US-100的Trig/TX

void setup() {

pinMode(a,INPUT);// 设置a为输入模式

pinMode(b,OUTPUT);// 设置b为输出模式

Serial.begin(9600);

}

void loop() {

long c = 0;//脉冲时间

long d = 0;//脉冲距离

// 通过Trig/Pin 发送脉冲，触发US-100 测距

digitalWrite(b,LOW); // 先拉低，以确保脉冲识别正确

delayMicroseconds(2);// 等待2us

digitalWrite(b,HIGH);// 开始通过Trig/Pin发送脉冲

delayMicroseconds(12);// 设置脉冲宽度为12us (>10us)

digitalWrite(b,LOW);// 结束脉冲

c = pulseIn(a,HIGH);// 计算US-100 返回的脉冲时间

d = c\*0.34/2; //距离=脉冲时间\*声波的速度（340/s）/2

delay(300);

Serial.print(d);

Serial.println("mm");

}

### 3.2.3 HC-05模块

#### 2.相关代码

#include <SoftwareSerial.h>

// 开发板引脚10为RX，接HC-05的TXD

// 开发板引脚11为TX，接HC-05的RXD

SoftwareSerial BT(8, 9);//R接蓝牙T,T接蓝牙R

char val;

void setup() {

Serial.begin(38400);

Serial.println("BT is ready!");

// HC-05默认，38400

BT.begin(38400);

}

void loop() {

while (Serial.available()) {

val = Serial.read();

BT.write(val);

Serial.print(val);

}

while (BT.available()) {

val = BT.read();

Serial.print(val);

}

}

### 3.2.4 输出模块

#### 2.相关代码

##### 1）手动演奏模式

#include <Adafruit\_NeoPixel.h> //彩色LED库

#include"shining.h"

# define PIN 6 //设置LED引脚

# define MAX\_LED 60 //特定的程序，最后一个数字是LED光带中LED的数量

Adafruit\_NeoPixel strip = Adafruit\_NeoPixel( MAX\_LED, PIN, NEO\_RGB + NEO\_KHZ800 ); //彩色LED库的内部设置

int a = 2;// 将Arduino开发板引脚2连接至US-100的Echo/RX

int b = 3;// 将Arduino开发板引脚3连接至US-100的Trig/TX

void setup() {

pinMode(a,INPUT);// 设置a为输入模式

pinMode(b,OUTPUT);// 设置b为输出模式

strip.begin(); //启动LED函数库

strip.show(); //将所有的LED灭掉

}

void loop() {

long c = 0;//脉冲时间

long d = 0;//脉冲距离

// 通过Trig/Pin 发送脉冲，触发US-100 测距

digitalWrite(b,LOW); // 先拉低，以确保脉冲识别正确

delayMicroseconds(2);// 等待2us

digitalWrite(b,HIGH);// 通过Trig/Pin 发送脉冲

delayMicroseconds(12);// 设置脉冲宽度为12us (>10us)

digitalWrite(b,LOW);// 结束脉冲

c = pulseIn(a,HIGH);// 计算US-100 返回的脉冲时间

d = c\*0.34/2; //距离=脉冲时间\*声波的速度（340/s）/2

if(d>50&&d<=150) //当脉冲距离>50mm并且<=150的时候，执行以下程序

{

ala(); //扬声器发出A4的音调

ledz(); //使所有LED灭

leda(); //使全部LED发出红色和大红色的光

}

if(d>150&&d<=200)

{

asi(); //扬声器发出B4的音调

ledz();//使所有LED灭

ledb();//使全部LED发出淡蓝色和蓝绿色的光

}

if(d>200&&d<=250)

{

bdo(); //扬声器发出C5的音调

ledz();//使所有LED灭

ledc();//使全部LED发出金黄色的光

}

if(d>250&&d<=300)

{

bre(); //扬声器发出D5的音调

ledz();//使所有LED灭

ledd();//使全部LED发出绿色的光

}

if(d>300&&d<=350)

{

bmi(); //扬声器发出E5的音调

ledz();//使所有LED灭

lede();//使全部LED发出大红色的光

}

if(d>350&&d<=400)

{

bfa(); //扬声器发出F5的音调

ledz();//使所有LED灭

ledf();//使全部LED发出湖蓝和淡蓝的光

}

if(d>400&&d<=450)

{

bso(); //扬声器发出G5的音调

ledz();//使所有LED灭

ledg();//使全部LED发出绿色和橘黄色的光

}

if(d>450&&d<=500)

{

bla(); //扬声器发出A5的音调

ledz();//使所有LED灭

ledh();//使全部LED发出淡蓝色的光

}

if(d>500&&d<=550)

{

bsi(); //扬声器发出B5的音调

ledz();//使所有LED灭

ledi();//使全部LED发出湖蓝和浅蓝的光

}

if(d>550&&d<=600)

{

cdo(); //扬声器发出C6的音调

ledz();//使所有LED灭

ledj();//使全部LED发出红色和大红色的光

}

if(d>600&&d<=800)

{

cre(); //扬声器发出D6的音调

ledz();//使所有LED灭

ledk();//使全部LED发出大红色和金黄色的光

}

if(d<=50)

{

noTone(9);

ledz();//使所有LED灭

ledy();//使全部LED发出深蓝、淡紫、红、橘黄、金黄、绿、蓝绿、天蓝、紫、大红、橘红、湖蓝、淡蓝、蓝、浅蓝色的光

}

if(d>800)

{

noTone(9);

ledz();//使所有LED灭

ledy();//使全部LED发出深蓝、淡紫、红、橘黄、金黄、绿、蓝绿、天蓝、紫、大红、橘红、湖蓝、淡蓝、蓝、浅蓝色的光

}

delay(400);// 每S测量2.5次

}

void ala() //设置一个函数:以后想要使用{ }里程序的话，可以直接打函数名，如：ala

{

tone(9,440); //声音函数（LED引脚频率）发出A4的音调

}

void asi()

{

tone(9,493); //发出B4的音调

}

void bdo()

{

tone(9,523); //发出C5的音调

}

void bre()

{

tone(9,587); //发出D5的音调

}

void bmi()

{

tone(9,659); //发出E5的音调

}

void bfa()

{

tone(9,698); //发出F5的音调

}

void bso()

{

tone(9,784); //发出G5的音调

}

void bla()

{

tone(9,880); //发出A5的音调

}

void bsi()

{

tone(9,988); //发出B5的音调

}

void cdo()

{

tone(9,1046); //发出C6的音调

}

void cre()

{

tone(9,1175); //发出D6的音调

}

##### 2）自动播放模式

#include"shining.h"

#define NOTE\_B0 31 //....7

#define NOTE\_C1 33 //...1

#define NOTE\_CS1 35

#define NOTE\_D1 37 //...2

#define NOTE\_DS1 39

#define NOTE\_E1 41 //...3

#define NOTE\_F1 44 //...4

#define NOTE\_FS1 46

#define NOTE\_G1 49 //...5

#define NOTE\_GS1 52

#define NOTE\_A1 55 //...6

#define NOTE\_AS1 58

#define NOTE\_B1 62 //...7

#define NOTE\_C2 65 //..1

#define NOTE\_CS2 69

#define NOTE\_D2 73 //..2

#define NOTE\_DS2 78

#define NOTE\_E2 82 //..3

#define NOTE\_F2 87 //..4

#define NOTE\_FS2 93

#define NOTE\_G2 98 //..5

#define NOTE\_GS2 104

#define NOTE\_A2 110 //..6

#define NOTE\_AS2 117

#define NOTE\_B2 123 //..7

#define NOTE\_C3 131 //.1

#define NOTE\_CS3 139

#define NOTE\_D3 147 //.2

#define NOTE\_DS3 156

#define NOTE\_E3 165 //.3

#define NOTE\_F3 175 //.4

#define NOTE\_FS3 185

#define NOTE\_G3 196 //.5

#define NOTE\_GS3 208

#define NOTE\_A3 220 //.6

#define NOTE\_AS3 233

#define NOTE\_B3 247 //.7

#define NOTE\_C4 262 //1

#define NOTE\_CS4 277

#define NOTE\_D4 294 //2

#define NOTE\_DS4 311

#define NOTE\_E4 330 //3

#define NOTE\_F4 349 //4

#define NOTE\_FS4 370

#define NOTE\_G4 392 //5

#define NOTE\_GS4 415

#define NOTE\_A4 440 //6

#define NOTE\_AS4 466

#define NOTE\_B4 494 //7

#define NOTE\_C5 523 //1.

#define NOTE\_CS5 554

#define NOTE\_D5 587 //2.

#define NOTE\_DS5 622

#define NOTE\_E5 659 //3.

#define NOTE\_F5 698 //4.

#define NOTE\_FS5 740

#define NOTE\_G5 784 //5.

#define NOTE\_GS5 831

#define NOTE\_A5 880 //6.

#define NOTE\_AS5 932

#define NOTE\_B5 988 //7.

#define NOTE\_C6 1047 //1..

#define NOTE\_CS6 1109

#define NOTE\_D6 1175 //2..

#define NOTE\_DS6 1245

#define NOTE\_E6 1319 //3..

#define NOTE\_F6 1397 //4..

#define NOTE\_FS6 1480

#define NOTE\_G6 1568 //5..

#define NOTE\_GS6 1661

#define NOTE\_A6 1760 //6..

#define NOTE\_AS6 1865

#define NOTE\_B6 1976 //7..

#define NOTE\_C7 2093 //1...

#define NOTE\_CS7 2217

#define NOTE\_D7 2349 //2...

#define NOTE\_DS7 2489

#define NOTE\_E7 2637 //3...

#define NOTE\_F7 2794 //4...

#define NOTE\_FS7 2960

#define NOTE\_G7 3136 //5...

#define NOTE\_GS7 3322

#define NOTE\_A7 3520 //6...

#define NOTE\_AS7 3729

#define NOTE\_B7 3951 //7...

#define NOTE\_C8 4186 //1....

#define NOTE\_CS8 4435

#define NOTE\_D8 4699 //2....

#define NOTE\_DS8 4978

int melody[] = {

NOTE\_A3,//.6

NOTE\_B3,//.7

NOTE\_C4,//1

NOTE\_C4,//1

NOTE\_B3,//.7

NOTE\_C4,//1

NOTE\_E4,//3

NOTE\_B3,//.7

NOTE\_B3,//.7

NOTE\_B3,//.7

NOTE\_E3,//.3

NOTE\_A3,//.6

NOTE\_A3,//.6

NOTE\_G3,//.5

NOTE\_A3,//.6

NOTE\_C4,//1

NOTE\_G3,//.5

NOTE\_G3,//.5

NOTE\_G3,//.5

NOTE\_E3,//.3

NOTE\_F3,//.4

NOTE\_F3,//.4

NOTE\_E3,//.3

NOTE\_F3,//.4

NOTE\_C4,//1

NOTE\_C4,//1

NOTE\_E3,//.3

NOTE\_E3,//.3

NOTE\_E3,//.3

NOTE\_C4,//1

NOTE\_B3,//.7

NOTE\_B3,//.7

NOTE\_FS3,//.4#

NOTE\_FS3,//.4#

NOTE\_B3,//.7

NOTE\_B3,//.7

NOTE\_B3,//.7

0,

NOTE\_A3,//.6

NOTE\_B3,//.7

NOTE\_C4,//1

NOTE\_C4,//1

NOTE\_B3,//.7

NOTE\_C4,//1

NOTE\_E4,//3

NOTE\_B3,//.7

NOTE\_B3,//.7

NOTE\_B3,//.7

NOTE\_E3,//.3

NOTE\_E3,//.3

NOTE\_A3,//.6

NOTE\_C4,//1

NOTE\_G3,//.5

NOTE\_A3,//.6

NOTE\_C4,//1

NOTE\_G3,//.5

NOTE\_G3,//.5

NOTE\_G3,//.5

NOTE\_E3,//.3

NOTE\_F3,//.4

NOTE\_C4,//1

NOTE\_B3,//.7

NOTE\_B3,//.7

NOTE\_C4,//1

NOTE\_D4,//2

NOTE\_E4,//3

NOTE\_C4,//1

NOTE\_C4,//1

0,

NOTE\_C4,//1

NOTE\_B3,//.7

NOTE\_A3,//.6

NOTE\_B3,//.7

NOTE\_GS3,//.5#

NOTE\_A3,//.6

NOTE\_A3,//.6

0,

NOTE\_C4,//1

NOTE\_D4,//2;

int noteDurations[79] = {

8,8,

4,8,8,4,4,

4,4,4,4,

4,8,8,4,4,

4,4,4,4,

4,8,8,8,8,4,

4,4,4,4,

4,8,8,4,4,

4,4,4,8,8,

4,8,8,4,4,

4,4,4,8,8,

4,8,8,4,4,

4,4,4,4,

4,8,8,4,4,

4,8,4,4,8,

8,8,4,4,4,

4,4,4,8,8, };

void setup() {

   for (int thisNote = 0; thisNote < 79; thisNote++)

{

    int noteDuration = 1000/noteDurations[thisNote];

    tone(9, melody[thisNote],noteDuration);

    int pauseBetweenNotes = noteDuration \* 1.30;

    delay(pauseBetweenNotes);

noTone(9);

{

if(melody[thisNote]==NOTE\_A3) //当扬声器发出A3声调时，执行以下程序

{

ledz(); //使所有LED灭

leda(); //使所有LED亮，并发出大红和红色的光

}

if(melody[thisNote]==NOTE\_B3)

{

ledz();//使所有LED灭

ledb();//使全部LED发出淡蓝色和蓝绿色的光

}

if(melody[thisNote]==NOTE\_C3)

{

ledz();//使所有LED灭

ledc();//使全部LED发出金黄色的光

}

if(melody[thisNote]==NOTE\_D4)

{

ledz();//使所有LED灭

ledd();//使全部LED发出绿色的光

}

if(melody[thisNote]==NOTE\_E3)

{

ledz();//使所有LED灭

lede();//使全部LED发出大红色的光

}

if(melody[thisNote]==NOTE\_E4)

{

ledz();//使所有LED灭

ledf();//使全部LED发出湖蓝和淡蓝的光

}

if(melody[thisNote]==NOTE\_F3)

{

ledz();//使所有LED灭

ledg();//使全部LED发出绿色和橘黄色的光

}

if(melody[thisNote]==NOTE\_G3)

{

ledz();//使所有LED灭

ledh();//使全部LED发出淡蓝色的光

}

if(melody[thisNote]==NOTE\_FS3)

{

ledz();//使所有LED灭

ledi();//使全部LED发出湖蓝和浅蓝的光

}

if(melody[thisNote]==NOTE\_GS3)

{

ledz();//使所有LED灭

ledj();//使全部LED发出红色和大红色的光

}

}

  }

}

##### 3）引用的shining.h文件

#include <Adafruit\_NeoPixel.h> //彩色LED库

#include"tone.h"

# define PIN 6 //设置LED引脚

# define MAX\_LED 60 //特定的程序，最后一个数字是LED光带中LED的数量

Adafruit\_NeoPixel strip = Adafruit\_NeoPixel( MAX\_LED, PIN, NEO\_RGB + NEO\_KHZ800 ); //彩色LED库的内部设置

void ledj()//使全部LED发出红色和大红色的光

{

//颜色变量设置

uint32\_t a = strip.Color(191,0,255); //深蓝

uint32\_t b = strip.Color(112,218,214); //淡紫

uint32\_t m = strip.Color(0,120,0); //红色

uint32\_t n = strip.Color(69,255,0); //橘黄

uint32\_t e = strip.Color(255,255,0); //金黄

uint32\_t f = strip.Color(139,34,34); //绿色

uint32\_t g = strip.Color(255,0,255); //蓝绿色

uint32\_t h = strip.Color(0,0,255); //天蓝

uint32\_t i = strip.Color(0,128,128); //紫色

uint32\_t j = strip.Color(0,255,0); //大红

uint32\_t k = strip.Color(165,255,0); //橘红

uint32\_t o = strip.Color(111,0,175); //湖蓝

uint32\_t p = strip.Color(31,0,95); //淡蓝

uint32\_t q = strip.Color(85,85,0); //蓝

uint32\_t r = strip.Color(25,25,0); //浅蓝

//设置LED发出的颜色

strip.setPixelColor(0, j);

strip.setPixelColor(1, m);

strip.setPixelColor(2, j);

strip.setPixelColor(3, m);

strip.setPixelColor(4, j);

strip.setPixelColor(5, m);

strip.setPixelColor(6, j);

strip.setPixelColor(7, m);

strip.setPixelColor(8, j);

strip.setPixelColor(9, m);

strip.setPixelColor(10, j);

strip.setPixelColor(11, m);

strip.setPixelColor(12, j);

strip.setPixelColor(13, m);

strip.setPixelColor(14, j);

strip.setPixelColor(15, m);

strip.setPixelColor(16, j);

strip.setPixelColor(17, m);

strip.setPixelColor(18, j);

strip.setPixelColor(19, m);

strip.setPixelColor(20, j);

strip.setPixelColor(21, m);

strip.setPixelColor(22, j);

strip.setPixelColor(23, m);

strip.setPixelColor(24, j);

strip.setPixelColor(25, m);

strip.setPixelColor(26, j);

strip.setPixelColor(27, m);

strip.setPixelColor(28, j);

strip.setPixelColor(29, m);

strip.setPixelColor(30, j);

strip.setPixelColor(31, m);

strip.setPixelColor(32, j);

strip.setPixelColor(33, m);

strip.setPixelColor(34, j);

strip.setPixelColor(35, m);

strip.setPixelColor(36, j);

strip.setPixelColor(37, m);

strip.setPixelColor(38, j);

strip.setPixelColor(39, m);

strip.setPixelColor(40, j);

strip.setPixelColor(41, m);

strip.setPixelColor(42, j);

strip.setPixelColor(43, m);

strip.setPixelColor(44, j);

strip.setPixelColor(45, m);

strip.setPixelColor(46, j);

strip.setPixelColor(47, m);

strip.setPixelColor(48, j);

strip.setPixelColor(49, m);

strip.setPixelColor(50, j);

strip.setPixelColor(51, m);

strip.setPixelColor(52, j);

strip.setPixelColor(53, m);

strip.setPixelColor(54, j);

strip.setPixelColor(55, m);

strip.setPixelColor(56, j);

strip.setPixelColor(57, m);

strip.setPixelColor(58, j);

strip.setPixelColor(59, m);

strip.show();

}

void ledi()//使全部LED发出湖蓝和浅蓝的光

{

//颜色变量设置:

uint32\_t a = strip.Color(191,0,255); //深蓝

uint32\_t b = strip.Color(112,218,214); //淡紫

uint32\_t m = strip.Color(0,120,0); //红色

uint32\_t n = strip.Color(69,255,0); //橘黄

uint32\_t e = strip.Color(255,255,0); //金黄

uint32\_t f = strip.Color(139,34,34); //绿色

uint32\_t g = strip.Color(255,0,255); //蓝绿色

uint32\_t h = strip.Color(0,0,255); //天蓝

uint32\_t i = strip.Color(0,128,128); //紫色

uint32\_t j = strip.Color(0,255,0); //大红

uint32\_t k = strip.Color(165,255,0); //橘红

uint32\_t o = strip.Color(111,0,175); //湖蓝

uint32\_t p = strip.Color(31,0,95); //淡蓝

uint32\_t q = strip.Color(85,85,0); //蓝

uint32\_t r = strip.Color(25,25,0); //浅蓝

strip.setPixelColor(0, o);

strip.setPixelColor(1, r);

strip.setPixelColor(2, o);

strip.setPixelColor(3, r);

strip.setPixelColor(4, o);

strip.setPixelColor(5, r);

strip.setPixelColor(6, o);

strip.setPixelColor(7, r);

strip.setPixelColor(8, o);

strip.setPixelColor(9, r);

strip.setPixelColor(10, o);

strip.setPixelColor(11, r);

strip.setPixelColor(12, o);

strip.setPixelColor(13, r);

strip.setPixelColor(14, o);

strip.setPixelColor(15, r);

strip.setPixelColor(16, o);

strip.setPixelColor(17, r);

strip.setPixelColor(18, o);

strip.setPixelColor(19, r);

strip.setPixelColor(20, o);

strip.setPixelColor(21, r);

strip.setPixelColor(22, o);

strip.setPixelColor(23, r);

strip.setPixelColor(24, o);

strip.setPixelColor(25, r);

strip.setPixelColor(26, o);

strip.setPixelColor(27, r);

strip.setPixelColor(28, o);

strip.setPixelColor(29, r);

strip.setPixelColor(30, o);

strip.setPixelColor(31, r);

strip.setPixelColor(32, o);

strip.setPixelColor(33, r);

strip.setPixelColor(34, o);

strip.setPixelColor(35, r);

strip.setPixelColor(36, o);

strip.setPixelColor(37, r);

strip.setPixelColor(38, o);

strip.setPixelColor(39, r);

strip.setPixelColor(40, o);

strip.setPixelColor(41, r);

strip.setPixelColor(42, o);

strip.setPixelColor(43, r);

strip.setPixelColor(44, o);

strip.setPixelColor(45, r);

strip.setPixelColor(46, o);

strip.setPixelColor(47, r);

strip.setPixelColor(48, o);

strip.setPixelColor(49, r);

strip.setPixelColor(50, o);

strip.setPixelColor(51, r);

strip.setPixelColor(52, o);

strip.setPixelColor(53, r);

strip.setPixelColor(54, o);

strip.setPixelColor(55, r);

strip.setPixelColor(56, o);

strip.setPixelColor(57, r);

strip.setPixelColor(58, o);

strip.setPixelColor(59, r);

strip.show();

}

void ledh()//使全部LED发出淡蓝色的光

{

//颜色变量设置:

uint32\_t a = strip.Color(191,0,255); //深蓝

uint32\_t b = strip.Color(112,218,214); //淡紫

uint32\_t m = strip.Color(0,120,0); //红色

uint32\_t n = strip.Color(69,255,0); //橘黄

uint32\_t e = strip.Color(255,255,0); //金黄

uint32\_t f = strip.Color(139,34,34); //绿色

uint32\_t g = strip.Color(255,0,255); //蓝绿色

uint32\_t h = strip.Color(0,0,255); //天蓝

uint32\_t i = strip.Color(0,128,128); //紫色

uint32\_t j = strip.Color(0,255,0); //大红

uint32\_t k = strip.Color(165,255,0); //橘红

uint32\_t o = strip.Color(111,0,175); //湖蓝

uint32\_t p = strip.Color(31,0,95); //淡蓝

uint32\_t q = strip.Color(85,85,0); //蓝

uint32\_t r = strip.Color(25,25,0); //浅蓝

strip.setPixelColor(0, p);

strip.setPixelColor(1, p);

strip.setPixelColor(2, p);

strip.setPixelColor(3, p);

strip.setPixelColor(4, p);

strip.setPixelColor(5, p);

strip.setPixelColor(6, p);

strip.setPixelColor(7, p);

strip.setPixelColor(8, p);

strip.setPixelColor(9, p);

strip.setPixelColor(10, p);

strip.setPixelColor(11, p);

strip.setPixelColor(12, p);

strip.setPixelColor(13, p);

strip.setPixelColor(14, p);

strip.setPixelColor(15, p);

strip.setPixelColor(16, p);

strip.setPixelColor(17, p);

strip.setPixelColor(18, p);

strip.setPixelColor(19, p);

strip.setPixelColor(20, p);

strip.setPixelColor(21, p);

strip.setPixelColor(22, p);

strip.setPixelColor(23, p);

strip.setPixelColor(24, p);

strip.setPixelColor(25, p);

strip.setPixelColor(26, p);

strip.setPixelColor(27, p);

strip.setPixelColor(28, p);

strip.setPixelColor(29, p);

strip.setPixelColor(30, p);

strip.setPixelColor(31, p);

strip.setPixelColor(32, p);

strip.setPixelColor(33, p);

strip.setPixelColor(34, p);

strip.setPixelColor(35, p);

strip.setPixelColor(36, p);

strip.setPixelColor(37, p);

strip.setPixelColor(38, p);

strip.setPixelColor(39, p);

strip.setPixelColor(40, p);

strip.setPixelColor(41, p);

strip.setPixelColor(42, p);

strip.setPixelColor(43, p);

strip.setPixelColor(44, p);

strip.setPixelColor(45, p);

strip.setPixelColor(46, p);

strip.setPixelColor(47, p);

strip.setPixelColor(48, p);

strip.setPixelColor(49, p);

strip.setPixelColor(50, p);

strip.setPixelColor(51, p);

strip.setPixelColor(52, p);

strip.setPixelColor(53, p);

strip.setPixelColor(54, p);

strip.setPixelColor(55, p);

strip.setPixelColor(56, p);

strip.setPixelColor(57, p);

strip.setPixelColor(58, p);

strip.setPixelColor(59, p);

strip.show();

}

void ledg()//使全部LED发出绿色和橘黄色的光

{

//颜色变量设置:

uint32\_t a = strip.Color(191,0,255); //深蓝

uint32\_t b = strip.Color(112,218,214); //淡紫

uint32\_t m = strip.Color(0,120,0); //红色

uint32\_t n = strip.Color(69,255,0); //橘黄

uint32\_t e = strip.Color(255,255,0); //金黄

uint32\_t f = strip.Color(139,34,34); //绿色

uint32\_t g = strip.Color(255,0,255); //蓝绿色

uint32\_t h = strip.Color(0,0,255); //天蓝

uint32\_t i = strip.Color(0,128,128); //紫色

uint32\_t j = strip.Color(0,255,0); //大红

uint32\_t k = strip.Color(165,255,0); //橘红

uint32\_t o = strip.Color(111,0,175); //湖蓝

uint32\_t p = strip.Color(31,0,95); //淡蓝

uint32\_t q = strip.Color(85,85,0); //蓝

uint32\_t r = strip.Color(25,25,0); //浅蓝

strip.setPixelColor(0, n);

strip.setPixelColor(1, f);

strip.setPixelColor(2, n);

strip.setPixelColor(3, f);

strip.setPixelColor(4, n);

strip.setPixelColor(5, f);

strip.setPixelColor(6, n);

strip.setPixelColor(7, f);

strip.setPixelColor(8, n);

strip.setPixelColor(9, f);

strip.setPixelColor(10, n);

strip.setPixelColor(11, f);

strip.setPixelColor(12, n);

strip.setPixelColor(13, f);

strip.setPixelColor(14, n);

strip.setPixelColor(15, f);

strip.setPixelColor(16, n);

strip.setPixelColor(17, f);

strip.setPixelColor(18, n);

strip.setPixelColor(19, f);

strip.setPixelColor(20, n);

strip.setPixelColor(21, f);

strip.setPixelColor(22, n);

strip.setPixelColor(23, f);

strip.setPixelColor(24, n);

strip.setPixelColor(25, f);

strip.setPixelColor(26, n);

strip.setPixelColor(27, f);

strip.setPixelColor(28, n);

strip.setPixelColor(29, f);

strip.setPixelColor(30, n);

strip.setPixelColor(31, f);

strip.setPixelColor(32, n);

strip.setPixelColor(33, f);

strip.setPixelColor(34, n);

strip.setPixelColor(35, f);

strip.setPixelColor(36, n);

strip.setPixelColor(37, f);

strip.setPixelColor(38, n);

strip.setPixelColor(39, f);

strip.setPixelColor(40, n);

strip.setPixelColor(41, f);

strip.setPixelColor(42, n);

strip.setPixelColor(43, f);

strip.setPixelColor(44, n);

strip.setPixelColor(45, f);

strip.setPixelColor(46, n);

strip.setPixelColor(47, f);

strip.setPixelColor(48, n);

strip.setPixelColor(49, f);

strip.setPixelColor(50, n);

strip.setPixelColor(51, f);

strip.setPixelColor(52, n);

strip.setPixelColor(53, f);

strip.setPixelColor(54, n);

strip.setPixelColor(55, f);

strip.setPixelColor(56, n);

strip.setPixelColor(57, f);

strip.setPixelColor(58, n);

strip.setPixelColor(59, f);

strip.show();

}

void ledf()//使全部LED发出湖蓝和淡蓝的光

{

//颜色变量设置

uint32\_t a = strip.Color(191,0,255); //深蓝

uint32\_t b = strip.Color(112,218,214); //淡紫

uint32\_t m = strip.Color(0,120,0); //红色

uint32\_t n = strip.Color(69,255,0); //橘黄

uint32\_t e = strip.Color(255,255,0); //金黄

uint32\_t f = strip.Color(139,34,34); //绿色

uint32\_t g = strip.Color(255,0,255); //蓝绿色

uint32\_t h = strip.Color(0,0,255); //天蓝

uint32\_t i = strip.Color(0,128,128); //紫色

uint32\_t j = strip.Color(0,255,0); //大红

uint32\_t k = strip.Color(165,255,0); //橘红

uint32\_t o = strip.Color(111,0,175); //湖蓝

uint32\_t p = strip.Color(31,0,95); //淡蓝

uint32\_t q = strip.Color(85,85,0); //蓝

uint32\_t r = strip.Color(25,25,0); //浅蓝

strip.setPixelColor(0, o);

strip.setPixelColor(1, p);

strip.setPixelColor(2, o);

strip.setPixelColor(3, p);

strip.setPixelColor(4, o);

strip.setPixelColor(5, p);

strip.setPixelColor(6, o);

strip.setPixelColor(7, p);

strip.setPixelColor(8, o);

strip.setPixelColor(9, p);

strip.setPixelColor(10, o);

strip.setPixelColor(11, p);

strip.setPixelColor(12, o);

strip.setPixelColor(13, p);

strip.setPixelColor(14, o);

strip.setPixelColor(15, p);

strip.setPixelColor(16, o);

strip.setPixelColor(17, p);

strip.setPixelColor(18, o);

strip.setPixelColor(19, p);

strip.setPixelColor(20, o);

strip.setPixelColor(21, p);

strip.setPixelColor(22, o);

strip.setPixelColor(23, p);

strip.setPixelColor(24, o);

strip.setPixelColor(25, p);

strip.setPixelColor(26, o);

strip.setPixelColor(27, p);

strip.setPixelColor(28, o);

strip.setPixelColor(29, p);

strip.setPixelColor(30, o);

strip.setPixelColor(31, p);

strip.setPixelColor(32, o);

strip.setPixelColor(33, p);

strip.setPixelColor(34, o);

strip.setPixelColor(35, p);

strip.setPixelColor(36, o);

strip.setPixelColor(37, p);

strip.setPixelColor(38, o);

strip.setPixelColor(39, p);

strip.setPixelColor(40, o);

strip.setPixelColor(41, p);

strip.setPixelColor(42, o);

strip.setPixelColor(43, p);

strip.setPixelColor(44, o);

strip.setPixelColor(45, p);

strip.setPixelColor(46, o);

strip.setPixelColor(47, p);

strip.setPixelColor(48, o);

strip.setPixelColor(49, p);

strip.setPixelColor(50, o);

strip.setPixelColor(51, p);

strip.setPixelColor(52, o);

strip.setPixelColor(53, p);

strip.setPixelColor(54, o);

strip.setPixelColor(55, p);

strip.setPixelColor(56, o);

strip.setPixelColor(57, p);

strip.setPixelColor(58, o);

strip.setPixelColor(59, p);

strip.show();

}

void lede()//使全部LED发出大红色的光

{

//颜色变量设置:

uint32\_t a = strip.Color(191,0,255); //深蓝

uint32\_t b = strip.Color(112,218,214); //淡紫

uint32\_t m = strip.Color(0,120,0); //红色

uint32\_t n = strip.Color(69,255,0); //橘黄

uint32\_t e = strip.Color(255,255,0); //金黄

uint32\_t f = strip.Color(139,34,34); //绿色

uint32\_t g = strip.Color(255,0,255); //蓝绿色

uint32\_t h = strip.Color(0,0,255); //天蓝

uint32\_t i = strip.Color(0,128,128); //紫色

uint32\_t j = strip.Color(0,255,0); //大红

uint32\_t k = strip.Color(165,255,0); //橘红

uint32\_t o = strip.Color(111,0,175); //湖蓝

uint32\_t p = strip.Color(31,0,95); //淡蓝

uint32\_t q = strip.Color(85,85,0); //蓝

uint32\_t r = strip.Color(25,25,0); //浅蓝

strip.setPixelColor(0, j);

strip.setPixelColor(1, j);

strip.setPixelColor(2, j);

strip.setPixelColor(3, j);

strip.setPixelColor(4, j);

strip.setPixelColor(5, j);

strip.setPixelColor(6, j);

strip.setPixelColor(7, j);

strip.setPixelColor(8, j);

strip.setPixelColor(9, j);

strip.setPixelColor(10, j);

strip.setPixelColor(11, j);

strip.setPixelColor(12, j);

strip.setPixelColor(13, j);

strip.setPixelColor(14, j);

strip.setPixelColor(15, j);

strip.setPixelColor(16, j);

strip.setPixelColor(17, j);

strip.setPixelColor(18, j);

strip.setPixelColor(19, j);

strip.setPixelColor(20, j);

strip.setPixelColor(21, j);

strip.setPixelColor(22, j);

strip.setPixelColor(23, j);

strip.setPixelColor(24, j);

strip.setPixelColor(25, j);

strip.setPixelColor(26, j);

strip.setPixelColor(27, j);

strip.setPixelColor(28, j);

strip.setPixelColor(29, j);

strip.setPixelColor(30, j);

strip.setPixelColor(31, j);

strip.setPixelColor(32, j);

strip.setPixelColor(33, j);

strip.setPixelColor(34, j);

strip.setPixelColor(35, j);

strip.setPixelColor(36, j);

strip.setPixelColor(37, j);

strip.setPixelColor(38, j);

strip.setPixelColor(39, j);

strip.setPixelColor(40, j);

strip.setPixelColor(41, j);

strip.setPixelColor(42, j);

strip.setPixelColor(43, j);

strip.setPixelColor(44, j);

strip.setPixelColor(45, j);

strip.setPixelColor(46, j);

strip.setPixelColor(47, j);

strip.setPixelColor(48, j);

strip.setPixelColor(49, j);

strip.setPixelColor(50, j);

strip.setPixelColor(51, j);

strip.setPixelColor(52, j);

strip.setPixelColor(53, j);

strip.setPixelColor(54, j);

strip.setPixelColor(55, j);

strip.setPixelColor(56, j);

strip.setPixelColor(57, j);

strip.setPixelColor(58, j);

strip.setPixelColor(59, j);

strip.show();

}

void ledd()//使全部LED发出绿色的光

{

//颜色变量设置:

uint32\_t a = strip.Color(191,0,255); //深蓝

uint32\_t b = strip.Color(112,218,214); //淡紫

uint32\_t m = strip.Color(0,120,0); //红色

uint32\_t n = strip.Color(69,255,0); //橘黄

uint32\_t e = strip.Color(255,255,0); //金黄

uint32\_t f = strip.Color(139,34,34); //绿色

uint32\_t g = strip.Color(255,0,255); //蓝绿色

uint32\_t h = strip.Color(0,0,255); //天蓝

uint32\_t i = strip.Color(0,128,128); //紫色

uint32\_t j = strip.Color(0,255,0); //大红

uint32\_t k = strip.Color(165,255,0); //橘红

uint32\_t o = strip.Color(111,0,175); //湖蓝

uint32\_t p = strip.Color(31,0,95); //淡蓝

uint32\_t q = strip.Color(85,85,0); //蓝

uint32\_t r = strip.Color(25,25,0); //浅蓝

strip.setPixelColor(0, f);

strip.setPixelColor(1, f);

strip.setPixelColor(2, f);

strip.setPixelColor(3, f);

strip.setPixelColor(4, f);

strip.setPixelColor(5, f);

strip.setPixelColor(6, f);

strip.setPixelColor(7, f);

strip.setPixelColor(8, f);

strip.setPixelColor(9, f);

strip.setPixelColor(10, f);

strip.setPixelColor(11, f);

strip.setPixelColor(12, f);

strip.setPixelColor(13, f);

strip.setPixelColor(14, f);

strip.setPixelColor(15, f);

strip.setPixelColor(16, f);

strip.setPixelColor(17, f);

strip.setPixelColor(18, f);

strip.setPixelColor(19, f);

strip.setPixelColor(20, f);

strip.setPixelColor(21, f);

strip.setPixelColor(22, f);

strip.setPixelColor(23, f);

strip.setPixelColor(24, f);

strip.setPixelColor(25, f);

strip.setPixelColor(26, f);

strip.setPixelColor(27, f);

strip.setPixelColor(28, f);

strip.setPixelColor(29, f);

strip.setPixelColor(30, f);

strip.setPixelColor(31, f);

strip.setPixelColor(32, f);

strip.setPixelColor(33, f);

strip.setPixelColor(34, f);

strip.setPixelColor(35, f);

strip.setPixelColor(36, f);

strip.setPixelColor(37, f);

strip.setPixelColor(38, f);

strip.setPixelColor(39, f);

strip.setPixelColor(40, f);

strip.setPixelColor(41, f);

strip.setPixelColor(42, f);

strip.setPixelColor(43, f);

strip.setPixelColor(44, f);

strip.setPixelColor(45, f);

strip.setPixelColor(46, f);

strip.setPixelColor(47, f);

strip.setPixelColor(48, f);

strip.setPixelColor(49, f);

strip.setPixelColor(50, f);

strip.setPixelColor(51, f);

strip.setPixelColor(52, f);

strip.setPixelColor(53, f);

strip.setPixelColor(54, f);

strip.setPixelColor(55, f);

strip.setPixelColor(56, f);

strip.setPixelColor(57, f);

strip.setPixelColor(58, f);

strip.setPixelColor(59, f);

strip.show();

}

void ledc()//使全部LED发出金黄色的光

{

//颜色变量设置:

uint32\_t a = strip.Color(191,0,255); //深蓝

uint32\_t b = strip.Color(112,218,214); //淡紫

uint32\_t m = strip.Color(0,120,0); //红色

uint32\_t n = strip.Color(69,255,0); //橘黄

uint32\_t e = strip.Color(255,255,0); //金黄

uint32\_t f = strip.Color(139,34,34); //绿色

uint32\_t g = strip.Color(255,0,255); //蓝绿色

uint32\_t h = strip.Color(0,0,255); //天蓝

uint32\_t i = strip.Color(0,128,128); //紫色

uint32\_t j = strip.Color(0,255,0); //大红

uint32\_t k = strip.Color(165,255,0); //橘红

uint32\_t o = strip.Color(111,0,175); //湖蓝

uint32\_t p = strip.Color(31,0,95); //淡蓝

uint32\_t q = strip.Color(85,85,0); //蓝

uint32\_t r = strip.Color(25,25,0); //浅蓝

strip.setPixelColor(0, e);

strip.setPixelColor(1, e);

strip.setPixelColor(2, e);

strip.setPixelColor(3, e);

strip.setPixelColor(4, e);

strip.setPixelColor(5, e);

strip.setPixelColor(6, e);

strip.setPixelColor(7, e);

strip.setPixelColor(8, e);

strip.setPixelColor(9, e);

strip.setPixelColor(10,e);

strip.setPixelColor(11, e);

strip.setPixelColor(12, e);

strip.setPixelColor(13, e);

strip.setPixelColor(14, e);

strip.setPixelColor(15, e);

strip.setPixelColor(16, e);

strip.setPixelColor(17, e);

strip.setPixelColor(18, e);

strip.setPixelColor(19, e);

strip.setPixelColor(20, e);

strip.setPixelColor(21, e);

strip.setPixelColor(22, e);

strip.setPixelColor(23, e);

strip.setPixelColor(24, e);

strip.setPixelColor(25, e);

strip.setPixelColor(26, e);

strip.setPixelColor(27, e);

strip.setPixelColor(28, e);

strip.setPixelColor(29, e);

strip.setPixelColor(30, e);

strip.setPixelColor(31, e);

strip.setPixelColor(32, e);

strip.setPixelColor(33, e);

strip.setPixelColor(34, e);

strip.setPixelColor(35, e);

strip.setPixelColor(36, e);

strip.setPixelColor(37, e);

strip.setPixelColor(38, e);

strip.setPixelColor(39, e);

strip.setPixelColor(40, e);

strip.setPixelColor(41, e);

strip.setPixelColor(42, e);

strip.setPixelColor(43, e);

strip.setPixelColor(44, e);

strip.setPixelColor(45, e);

strip.setPixelColor(46, e);

strip.setPixelColor(47, e);

strip.setPixelColor(48, e);

strip.setPixelColor(49, e);

strip.setPixelColor(50, e);

strip.setPixelColor(51, e);

strip.setPixelColor(52, e);

strip.setPixelColor(53, e);

strip.setPixelColor(54, e);

strip.setPixelColor(55, e);

strip.setPixelColor(56, e);

strip.setPixelColor(57, e);

strip.setPixelColor(58, e);

strip.setPixelColor(59, e);

strip.show();

}

void ledb()//使全部LED发出淡蓝色和蓝绿色的光

{

//颜色变量设置

uint32\_t a = strip.Color(191,0,255); //深蓝

uint32\_t b = strip.Color(112,218,214); //淡紫

uint32\_t m = strip.Color(0,120,0); //红色

uint32\_t n = strip.Color(69,255,0); //橘黄

uint32\_t e = strip.Color(255,255,0); //金黄

uint32\_t f = strip.Color(139,34,34); //绿色

uint32\_t g = strip.Color(255,0,255); //蓝绿色

uint32\_t h = strip.Color(0,0,255); //天蓝

uint32\_t i = strip.Color(0,128,128); //紫色

uint32\_t j = strip.Color(0,255,0); //大红

uint32\_t k = strip.Color(165,255,0); //橘红

uint32\_t o = strip.Color(111,0,175); //湖蓝

uint32\_t p = strip.Color(31,0,95); //淡蓝

uint32\_t q = strip.Color(85,85,0); //蓝

uint32\_t r = strip.Color(25,25,0); //浅蓝

strip.setPixelColor(0, p);

strip.setPixelColor(1, g);

strip.setPixelColor(2, p);

strip.setPixelColor(3, g);

strip.setPixelColor(4, p);

strip.setPixelColor(5, g);

strip.setPixelColor(6, p);

strip.setPixelColor(7, g);

strip.setPixelColor(8, p);

strip.setPixelColor(9, g);

strip.setPixelColor(10,p);

strip.setPixelColor(11, g);

strip.setPixelColor(12, p);

strip.setPixelColor(13, g);

strip.setPixelColor(14, p);

strip.setPixelColor(15, g);

strip.setPixelColor(16, p);

strip.setPixelColor(17, g);

strip.setPixelColor(18, p);

strip.setPixelColor(19, g);

strip.setPixelColor(20, p);

strip.setPixelColor(21, g);

strip.setPixelColor(22, p);

strip.setPixelColor(23, g);

strip.setPixelColor(24, p);

strip.setPixelColor(25, g);

strip.setPixelColor(26, p);

strip.setPixelColor(27, g);

strip.setPixelColor(28, p);

strip.setPixelColor(29, g);

strip.setPixelColor(30, p);

strip.setPixelColor(31, g);

strip.setPixelColor(32, p);

strip.setPixelColor(33, g);

strip.setPixelColor(34, p);

strip.setPixelColor(35, g);

strip.setPixelColor(36, p);

strip.setPixelColor(37, g);

strip.setPixelColor(38, p);

strip.setPixelColor(39, g);

strip.setPixelColor(40, p);

strip.setPixelColor(41, g);

strip.setPixelColor(42, p);

strip.setPixelColor(43, g);

strip.setPixelColor(44, p);

strip.setPixelColor(45, g);

strip.setPixelColor(46, p);

strip.setPixelColor(47, g);

strip.setPixelColor(48, p);

strip.setPixelColor(49, g);

strip.setPixelColor(50, p);

strip.setPixelColor(51, g);

strip.setPixelColor(52, p);

strip.setPixelColor(53, g);

strip.setPixelColor(54, p);

strip.setPixelColor(55, g);

strip.setPixelColor(56, p);

strip.setPixelColor(57, g);

strip.setPixelColor(58, p);

strip.setPixelColor(59, g);

strip.show();

}

void leda()//使所有LED亮，并发出大红和红色的光

{

//颜色变量设置

uint32\_t a = strip.Color(191,0,255); //深蓝

uint32\_t b = strip.Color(112,218,214); //淡紫

uint32\_t m = strip.Color(0,120,0); //红色

uint32\_t n = strip.Color(69,255,0); //橘黄

uint32\_t e = strip.Color(255,255,0); //金黄

uint32\_t f = strip.Color(139,34,34); //绿色

uint32\_t g = strip.Color(255,0,255); //蓝绿色

uint32\_t h = strip.Color(0,0,255); //天蓝

uint32\_t i = strip.Color(0,128,128); //紫色

uint32\_t j = strip.Color(0,255,0); //大红

uint32\_t k = strip.Color(165,255,0); //橘红

uint32\_t o = strip.Color(111,0,175); //湖蓝

uint32\_t p = strip.Color(31,0,95); //淡蓝

uint32\_t q = strip.Color(85,85,0); //蓝

uint32\_t r = strip.Color(25,25,0); //浅蓝

strip.setPixelColor(0, j);

strip.setPixelColor(1, m);

strip.setPixelColor(2, j);

strip.setPixelColor(3, m);

strip.setPixelColor(4, j);

strip.setPixelColor(5, m);

strip.setPixelColor(6, j);

strip.setPixelColor(7, m);

strip.setPixelColor(8, j);

strip.setPixelColor(9, m);

strip.setPixelColor(10, j);

strip.setPixelColor(11, m);

strip.setPixelColor(12, j);

strip.setPixelColor(13, m);

strip.setPixelColor(14, j);

strip.setPixelColor(15, m);

strip.setPixelColor(16, j);

strip.setPixelColor(17, m);

strip.setPixelColor(18, j);

strip.setPixelColor(19, m);

strip.setPixelColor(20, j);

strip.setPixelColor(21, m);

strip.setPixelColor(22, j);

strip.setPixelColor(23, m);

strip.setPixelColor(24, j);

strip.setPixelColor(25, m);

strip.setPixelColor(26, j);

strip.setPixelColor(27, m);

strip.setPixelColor(28, j);

strip.setPixelColor(29, m);

strip.setPixelColor(30, j);

strip.setPixelColor(31, m);

strip.setPixelColor(32, j);

strip.setPixelColor(33, m);

strip.setPixelColor(34, j);

strip.setPixelColor(35, m);

strip.setPixelColor(36, j);

strip.setPixelColor(37, m);

strip.setPixelColor(38, j);

strip.setPixelColor(39, m);

strip.setPixelColor(40, j);

strip.setPixelColor(41, m);

strip.setPixelColor(42, j);

strip.setPixelColor(43, m);

strip.setPixelColor(44, j);

strip.setPixelColor(45, m);

strip.setPixelColor(46, j);

strip.setPixelColor(47, m);

strip.setPixelColor(48, j);

strip.setPixelColor(49, m);

strip.setPixelColor(50, j);

strip.setPixelColor(51, m);

strip.setPixelColor(52, j);

strip.setPixelColor(53, m);

strip.setPixelColor(54, j);

strip.setPixelColor(55, m);

strip.setPixelColor(56, j);

strip.setPixelColor(57, m);

strip.setPixelColor(58, j);

strip.setPixelColor(59, m);

strip.show();

}

void ledz()//使所有LED灭

{

//颜色变量设置

uint32\_t u = strip.Color(0,0,0); //灭

strip.setPixelColor(0, u);

strip.setPixelColor(1, u);

strip.setPixelColor(2, u);

strip.setPixelColor(3, u);

strip.setPixelColor(4, u);

strip.setPixelColor(5, u);

strip.setPixelColor(6, u);

strip.setPixelColor(7, u);

strip.setPixelColor(8, u);

strip.setPixelColor(9, u);

strip.setPixelColor(10, u);

strip.setPixelColor(11, u);

strip.setPixelColor(12, u);

strip.setPixelColor(13, u);

strip.setPixelColor(14, u);

strip.setPixelColor(15, u);

strip.setPixelColor(16, u);

strip.setPixelColor(17, u);

strip.setPixelColor(18, u);

strip.setPixelColor(19, u);

strip.setPixelColor(20, u);

strip.setPixelColor(21, u);

strip.setPixelColor(22, u);

strip.setPixelColor(23, u);

strip.setPixelColor(24, u);

strip.setPixelColor(25, u);

strip.setPixelColor(26, u);

strip.setPixelColor(27, u);

strip.setPixelColor(28, u);

strip.setPixelColor(29, u);

strip.setPixelColor(30, u);

strip.setPixelColor(31, u);

strip.setPixelColor(32, u);

strip.setPixelColor(33, u);

strip.setPixelColor(34, u);

strip.setPixelColor(35, u);

strip.setPixelColor(36, u);

strip.setPixelColor(37, u);

strip.setPixelColor(38, u);

strip.setPixelColor(39, u);

strip.setPixelColor(40, u);

strip.setPixelColor(41, u);

strip.setPixelColor(42, u);

strip.setPixelColor(43, u);

strip.setPixelColor(44, u);

strip.setPixelColor(45, u);

strip.setPixelColor(46, u);

strip.setPixelColor(47, u);

strip.setPixelColor(48, u);

strip.setPixelColor(49, u);

strip.setPixelColor(50, u);

strip.setPixelColor(51, u);

strip.setPixelColor(52, u);

strip.setPixelColor(53, u);

strip.setPixelColor(54, u);

strip.setPixelColor(55, u);

strip.setPixelColor(56, u);

strip.setPixelColor(57, u);

strip.setPixelColor(58, u);

strip.setPixelColor(59, u);

}

void ledk()//使全部LED发出大红色和金黄色的光

{

//颜色变量设置:

uint32\_t a = strip.Color(191,0,255); //深蓝

uint32\_t b = strip.Color(112,218,214); //淡紫

uint32\_t m = strip.Color(0,120,0); //红色

uint32\_t n = strip.Color(69,255,0); //橘黄

uint32\_t e = strip.Color(255,255,0); //金黄

uint32\_t f = strip.Color(139,34,34); //绿色

uint32\_t g = strip.Color(255,0,255); //蓝绿色

uint32\_t h = strip.Color(0,0,255); //天蓝

uint32\_t i = strip.Color(0,128,128); //紫色

uint32\_t j = strip.Color(0,255,0); //大红

uint32\_t k = strip.Color(165,255,0); //橘红

uint32\_t o = strip.Color(111,0,175); //湖蓝

uint32\_t p = strip.Color(31,0,95); //淡蓝

uint32\_t q = strip.Color(85,85,0); //蓝

uint32\_t r = strip.Color(25,25,0); //浅蓝

strip.setPixelColor(0, j);

strip.setPixelColor(1, e);

strip.setPixelColor(2, j);

strip.setPixelColor(3, e);

strip.setPixelColor(4, j);

strip.setPixelColor(5, e);

strip.setPixelColor(6, j);

strip.setPixelColor(7, e);

strip.setPixelColor(8, j);

strip.setPixelColor(9, e);

strip.setPixelColor(10, j);

strip.setPixelColor(11, e);

strip.setPixelColor(12, j);

strip.setPixelColor(13, e);

strip.setPixelColor(14, j);

strip.setPixelColor(15, e);

strip.setPixelColor(16, j);

strip.setPixelColor(17, e);

strip.setPixelColor(18, j);

strip.setPixelColor(19, e);

strip.setPixelColor(20, j);

strip.setPixelColor(21, e);

strip.setPixelColor(22, j);

strip.setPixelColor(23, e);

strip.setPixelColor(24, j);

strip.setPixelColor(25, e);

strip.setPixelColor(26, j);

strip.setPixelColor(27, e);

strip.setPixelColor(28, j);

strip.setPixelColor(29, e);

strip.setPixelColor(30, j);

strip.setPixelColor(31, e);

strip.setPixelColor(32, j);

strip.setPixelColor(33, e);

strip.setPixelColor(34, j);

strip.setPixelColor(35, e);

strip.setPixelColor(36, j);

strip.setPixelColor(37, e);

strip.setPixelColor(38, j);

strip.setPixelColor(39, e);

strip.setPixelColor(40, j);

strip.setPixelColor(41, e);

strip.setPixelColor(42, j);

strip.setPixelColor(43, e);

strip.setPixelColor(44, j);

strip.setPixelColor(45, e);

strip.setPixelColor(46, j);

strip.setPixelColor(47, e);

strip.setPixelColor(48, j);

strip.setPixelColor(49, e);

strip.setPixelColor(50, j);

strip.setPixelColor(51, e);

strip.setPixelColor(52, j);

strip.setPixelColor(53, e);

strip.setPixelColor(54, j);

strip.setPixelColor(55, e);

strip.setPixelColor(56, j);

strip.setPixelColor(57, e);

strip.setPixelColor(58, j);

strip.setPixelColor(59, e);

strip.show();

}

void ledy()//使全部LED发出深蓝、淡紫、红、橘黄、金黄、绿、蓝绿、天蓝、紫、大红、橘红、湖蓝、淡蓝、蓝、浅蓝色的光

{

//颜色变量设置:

uint32\_t a = strip.Color(191,0,255); //深蓝

uint32\_t b = strip.Color(112,218,214); //淡紫

uint32\_t m = strip.Color(0,120,0); //红色

uint32\_t n = strip.Color(69,255,0); //橘黄

uint32\_t e = strip.Color(255,255,0); //金黄

uint32\_t f = strip.Color(139,34,34); //绿色

uint32\_t g = strip.Color(255,0,255); //蓝绿色

uint32\_t h = strip.Color(0,0,255); //天蓝

uint32\_t i = strip.Color(0,128,128); //紫色

uint32\_t j = strip.Color(0,255,0); //大红

uint32\_t k = strip.Color(165,255,0); //橘红

uint32\_t o = strip.Color(111,0,175); //湖蓝

uint32\_t p = strip.Color(31,0,95); //淡蓝

uint32\_t q = strip.Color(85,85,0); //蓝

uint32\_t r = strip.Color(25,25,0); //浅蓝

strip.setPixelColor(0, h);

strip.setPixelColor(1, b);

strip.setPixelColor(2, m);

strip.setPixelColor(3, n);

strip.setPixelColor(4, f);

strip.setPixelColor(5, q);

strip.setPixelColor(6, a);

strip.setPixelColor(7, o);

strip.setPixelColor(8, i);

strip.setPixelColor(9, j);

strip.setPixelColor(10, k);

strip.setPixelColor(11, g);

strip.setPixelColor(12, r);

strip.setPixelColor(13, e);

strip.setPixelColor(14, h);

strip.setPixelColor(15, b);

strip.setPixelColor(16, m);

strip.setPixelColor(17, n);

strip.setPixelColor(18, f);

strip.setPixelColor(19, q);

strip.setPixelColor(20, o);

strip.setPixelColor(21, i);

strip.setPixelColor(22, j);

strip.setPixelColor(23, k);

strip.setPixelColor(24, g);

strip.setPixelColor(25, r);

strip.setPixelColor(26, p);

strip.setPixelColor(27, e);

strip.setPixelColor(28, h);

strip.setPixelColor(29, b);

strip.setPixelColor(30, m);

strip.setPixelColor(31, n);

strip.setPixelColor(32, f);

strip.setPixelColor(33, q);

strip.setPixelColor(34, a);

strip.setPixelColor(35, o);

strip.setPixelColor(36, i);

strip.setPixelColor(37, j);

strip.setPixelColor(38, k);

strip.setPixelColor(39, g);

strip.setPixelColor(40, r);

strip.setPixelColor(41, p);

strip.setPixelColor(42, e);

strip.setPixelColor(43, h);

strip.setPixelColor(44, b);

strip.setPixelColor(45, m);

strip.setPixelColor(46, f);

strip.setPixelColor(47, q);

strip.setPixelColor(48, a);

strip.setPixelColor(49, o);

strip.setPixelColor(50, i);

strip.setPixelColor(51, j);

strip.setPixelColor(52, k);

strip.setPixelColor(53, g);

strip.setPixelColor(54, r);

strip.setPixelColor(55, p);

strip.setPixelColor(56, e);

strip.setPixelColor(57, h);

strip.setPixelColor(58, b);

strip.setPixelColor(59, m);

strip.show();

}

# 第4章 音乐游戏项目设计

## 4.2 模块介绍

### 4.2.1 输入模块

#### 2.相关代码

int fsrPin0 = A0;// A0引脚

int fsrPin1=A1;

int fsrPin2=A2;

int fsrPin3=A3;

int fsrReading0;

int fsrReading1;

int fsrReading2;

int fsrReading3;

void setup(void) {

pinMode(fsrPin0,INPUT);

pinMode(fsrPin1,INPUT);

pinMode(fsrPin2,INPUT);

pinMode(fsrPin3,INPUT);

Serial.begin(9600);

}

void loop(void) {

fsrReading0 = analogRead(fsrPin0);

fsrReading1 = analogRead(fsrPin1);

fsrReading2 = analogRead(fsrPin2);

fsrReading3 = analogRead(fsrPin3);

if (fsrReading0 > 5) {

Serial.print(1);

delay(500);

}

if (fsrReading1 > 5) {

Serial.print(2);

delay(500);

}

if (fsrReading2 > 5) {

Serial.print(3);

delay(500);

}

if (fsrReading3 > 5) {

Serial.print(4);

delay(500);

}

}

### 4.2.2 Processing界面显示模块

#### 2.相关代码

/\*本程序需要在Processing内添加库文件Minim，选择速写本→引用库文件→添加库文件，在Libraries中搜索Minim并下载。\*/

import ddf.minim.\*; //minim库文件

import Processing.serial.\*; //串口库文件

PImage backimg; //背景图

int keystate1=0; //4个按键状态

int keystate2=0;

int keystate3=0;

int keystate4=0;

int imageframe=8; //击中图片显示的帧数

int imagestate=0; //击中图片状态

int bcount=0; //鼓点计数器

int gamechoose=0;

int bgmstate=0; //音乐状态

int score=0; //总分统计

int combo=0; //连击统计

int df=1; //难度

int songnum; //歌曲编号

int sa; //串口读数

int menu=1;

int menu1=0; //菜单状态

int menu2=0;

int menu3=0;

int menu4=0;

int menu5=0;

int playstate=0; //游戏状态

int serialstate=1; //串口状态

int frame1=0; //串口敲击时鼓面变化的帧数

int frame2=0;

int frame3=0;

int frame4=0;

int bcolor; //连击统计颜色判断

AudioPlayer bgm; //音乐播放器

Minim mbgm; //文件接口

BufferedReader reader; //鼓点时间文件读取

Minim minim1; //鼓点音效文件接口

Minim minim2;

AudioSample inside; //鼓点音效播放接口

AudioSample outside;

String line; //鼓点时间文件读取行数

Timer timer; //计时器

Drumbeat[] db=new Drumbeat[40]; //实例化多个鼓点类

Mybutlisten mybut; //按键监听与鼓面变化类

Drum mydrum; //大鼓类

PImage img; //击中时的图片接口

int mykey; //储存当前按键

Serial serial; //串口初始化

int stime, bstyle, nstime, nbstyle; //储存map中某行的数据

int[] st; //储存对应编号鼓点类的出现时间

String song; //歌曲文件名

String songmap; //鼓点时间文件名

int beginX=746; //小球起始位置

int beginY=50;

color borderColor=color (50, 50, 50); //迷宫边界颜色

float x, y; //运动中小球坐标

int posx\_timer,posy\_timer; //读取小球坐标、半径

float radius\_timer;

int startTime; //计时开始时间

float endTime; //计时结束时间

Mission1 mission1; //第一关地图

Mission2 mission2; //第二关地图

CURRENTRecord currentRecord1; //第一关记录

CURRENTRecord currentRecord2; //第二关记录

int missionNo=0; //选择的关卡号

Ball ball; //小球相关参数

Timer1 timer\_record; //记录存档

int n=0; //判断选择记录显示内容

int m1=0,m2=0;

AudioPlayer bgm2; //音乐播放器

String song2;

int bgmstate2=0;

Minim mbgm2;

void setup()

{

println(Serial.list());

try {

serial=new Serial(this, "COM5", 9600);

//设置串口, 在Arduino开发板对应的引脚

}

catch (Exception e) {

println("没有连接到Arduino开发板设备");

serialstate=0; //不启用串口判断

}

PFont font = createFont("黑体", 25); //字体设置

textFont(font);

backimg=loadImage("background.png"); //背景图片

backimg.resize(800,800);

mbgm=new Minim(this); //实例化背景音乐minim类

imageMode(CENTER); //图片绘制起点设置为中心

img = loadImage("taiko-hit300.png"); //读取击中图片

minim1=new Minim(this); //实例化鼓点音效minim类

minim2=new Minim(this);

mybut=new Mybutlisten('c', 'b', 'd', 'h');

//定义键盘按键, c、b为鼓面(红), d、h为鼓边(蓝)

mydrum=new Drum(100, 225, 100, 100); //鼓面

st=new int[50]; //数组实例化

for (int k=0; k<40; k++)

db[k]=new Drumbeat(); //实例化40个鼓点

frameRate(120);

size(800, 800);

timer=new Timer();

inside=minim1.loadSample("in.wav", 512); //读取鼓点音效

outside=minim2.loadSample("out.wav", 512);

ball=new Ball(beginX, beginY, 20, borderColor);

currentRecord1=new CURRENTRecord();

currentRecord2=new CURRENTRecord();

timer\_record=new Timer1();

mbgm2=new Minim(this);

bgm2=mbgm2.loadFile("5.mp3");

}

void draw()

{

background(backimg); //背景

if(menu==1)

{ fill(0, 102, 153);

textSize(25);

text("游戏选择", 620, 120);

text("A.音乐打击", 600,170);

text("B.音乐小球", 600,220);

if (keyPressed)

if (key=='a'||key=='A')

{

gamechoose=1;

menu=0;

menu1=1;

}

if (key=='b'||key=='B')

{

gamechoose=2;

menu=0;

menu3=1;

}

}

if (playstate==1)

{ playsetup();

}

fill(#000000);

if (menu1==1&&gamechoose==1) //一级菜单

{

fill(0, 102, 153);

text("选择难度", 0, 20);

text("1.普通", 100, 20);

text("2.困难", 200, 20);

text("q.返回选择界面",200,50);

if (keyPressed)

if (key=='1')

{

df=1; // 1为普通，2为困难

menu1=0; //关闭一级菜单

menu2=1; //开启二级菜单

}

if (key=='2')

{

df=2;

menu1=0;

menu2=1;

}

if(key=='q'||key=='Q')

{df=1;

menu=1;

menu1=0;

menu2=0;

gamechoose=0;

menu3=0;

menu4=0;

menu5=0;

}

}

if (menu2==1&&gamechoose==1) //二级菜单

{

fill(0, 102, 153);

text("选择歌曲", 0, 20);

text("3.千本樱", 100, 20);

text("4.妖精的尾巴", 350, 20);

text("5.红莲の弓矢", 100, 50);

text("6.FLYING FAFNIR ", 350, 50);

if (keyPressed)

if (key=='3')

{

songnum=1;

menu2=0;

}

if (key=='4')

{

songnum=2;

menu2=0;

}

if (key=='5')

{

songnum=3;

menu2=0;

}

if (key=='6')

{

songnum=4;

menu2=0;

}

}

if (menu2==0&&menu1==0&&gamechoose==1) //三级菜单

{

fill(0, 102, 153);

text("7.开始游戏", 0, 20);

text("8.重新开始", 150, 20);

text("9.返回菜单", 300, 20);

text("连击统计：", 100, 50);

text("得分：",100, 100);

if (keyPressed)

{

if (key=='9') //返回一级菜单

{

stime=0;

nstime=0;

bcount=0;

for (int i=0; i<39; i++)

{

db[i].resetbeat(i);

}

timer.timereset();

menu1=1;

bgmstate=0;

playstate=0;

bgm.pause();

/\*for (int i=0; i<39; i++)

{

db[i].resetbeat(i);

}\*/

}

if (key=='7'&&playstate!=1&&menu1==0) //开始游戏

{

songselect(songnum);

bgm=mbgm.loadFile(song, 1024); //读取音频文件

reader=createReader(songmap); //读取map

firstread();

playgame();

} else if (key=='8') //重新开始

{

stime=0;

nstime=0;

try {

reader.close(); //释放文件缓存

}

catch(Exception e)

{

println("close file error");

}

if (bgmstate==0)

{

bgm=mbgm.loadFile(song, 1024); //读取音频文件

bgm.play();

}

reader=createReader(songmap);

bgm.rewind();

timer.timereset();

for (int i=0; i<39; i++) //重置所有鼓点

{

db[i].resetbeat(i);

}

bcount=0;

score=0;

combo=0;

firstread();

/\*timer.timereset();\*/

timer.start();

bgmstate=1;

playstate=1;

/\*for (int i=0; i<39; i++) //重置所有鼓点

{

db[i].resetbeat(i);

}\*/

}

}

}

if (timer.showtime(stime, timer.now())&&bgmstate==1)

//使用自定义计时器,并判断鼓点是否应该出现

{

if (bgmstate==1)

try {

line = reader.readLine(); //读取map文件下一行

}

catch (IOException e) {

e.printStackTrace();

line = null;

}

if (line!=null) //读取成功后储存鼓点出现的时间与类型

{

String data[]=split(line, ',');

println(data[0]);

nstime=int(data[0]); //下一个鼓点出现时间

nbstyle=int(data[1]); //下一个鼓点类型

} else {

timer.timereset(); //map读取后重置timer，游戏停止

bgmstate=0;

bgm.pause();

try {

reader.close();

}

catch(Exception e)

{

println("close erro");

}

}

db[bcount].resetbeat(bcount); //初始化对应鼓点

st[bcount]=bstyle; //储存对应的鼓点类型

bcount=(bcount+1)%40;

println(bcount);

}

if (playstate==1) //游戏中

for (int i=0; i<39; i++) //每个鼓点移动

{

act(i);

}

if (stime!=nstime) //下一行数据赋值

{

stime=nstime;

bstyle=nbstyle;

}

if (imagestate==1&&imageframe>0) //击中鼓点持续显示图片

{

image(img, 250, 220, 100, 100);

imageframe--;

} else {

imagestate=0;

}

if(menu==0&&menu3==1&&menu4==0&&menu5==0&&gamechoose==2)

//选择游戏二

{

fill(0,102,153);

textSize(25);

text("W.开始",20,20);

text("S.记录",120,20);

text("q.返回主页面",20,50);

if(keyPressed)

{

if(key=='w'||key=='W')

{

menu3=0;

menu4=1;

menu5=0;

}

if(key=='s'||key=='S')

{

menu3=0;

menu4=0;

menu5=1;

}

if(key=='q')

{

df=1;

menu=1;

menu1=0;

menu2=0;

gamechoose=0;

menu3=0;

menu4=0;

menu5=0;

}

}

}

if(menu4==1&&gamechoose==2) //选择关卡

{

background(backimg);

fill(153,102,0);

textSize(30);

text("A.Mission 1",500,170);

text("D.Mission 2",500,230);

if(keyPressed)

{

if(key=='a'||key=='A')

{

menu3=0;

menu4=0;

menu5=0;

missionNo=1;

}

if(key=='d'||key=='D')

{

menu3=0;

menu4=0;

menu5=0;

missionNo=2;

}

}

fill(0,210,200);

textSize(25);

text("Press 'R' to return",20,50);

if(key=='r'||key=='R')

{

key='a';

menu3=1;

menu4=0;

menu5=0;

}

}

if(menu5==1&&gamechoose==2) //查看记录

{

background(backimg);

fill(0,102,153);

textSize(35);

text("Record",500,230);

textSize(25);

if(m1==0&&m2==0)

{

text("Mission 1 "+"no record",270,520);

text("Mission 2 "+"no record",270,570);

}

else if(m1==1&&m2==0)

{

text("Mission 1 "+currentRecord1.currentRecord+" s",300,520);

text("Mission 2 "+"no record",270,570);

}

else if(m1==0&&m2==1)

{

text("Mission 1 "+"no record",270,520);

text("Mission 2 "+currentRecord2.currentRecord+" s",300,570);

}

else if(m1==1&&m2==1)

{

text("Mission 1 "+currentRecord1.currentRecord+" s",300,520);

text("Mission 2 "+currentRecord2.currentRecord+" s",300,570);

}

else

{

textSize(30);

text("Error",360,400);

}

textSize(25);

text("Press 'R' to return",20,50);

if(key=='r'||key=='R')

{

key='a';

menu3=1;

menu4=0;

menu5=0;

}

}

if(missionNo!=0) //关卡一

{

switch(missionNo)

{

case 1:

bgm2.play();

m1=1;

mission1=new Mission1();

//timer\_record=new Timer();

if(n==0)

{

timer\_record.start1(); //计时开始

startTime=timer\_record.start1();

n++;

//println("n1="+n);

}

background(204);

mission1.display();

ball.update(x, y);

if(posy\_timer==774&&(posx\_timer>=5)&&(posx\_timer<=100))

{

if(n==1)

{

currentRecord1.lastRecord=timer\_record.stop();

//println("entered"+currentRecord1.lastRecord);

n++;

}

win(); //通关，显示成绩

textSize(30);

text("Time: "+currentRecord1.lastRecord+"s",460,450);

currentRecord1.judge(); //更新游戏记录

//println("currentrecord1="+currentRecord1.currentRecord);

//println("lastrecord1="+currentRecord1.lastRecord);

if(n==2) //恢复小球参数至初始状态

{

if(key=='r'||key=='R')

{

key='a';

bgm2.pause();

ball.reset();

}

}

}

break;

case 2: //关卡二

bgm2.play();

m2=1;

mission2=new Mission2();

if(n==0)

{

timer\_record.start1();

startTime=timer\_record.start1();

n++; //n=1

//println("n1="+n);

}

background(204);

mission2.display();

ball.update(x, y);

if(posy\_timer==774&&(posx\_timer>=5)&&(posx\_timer<=100))

{

if(n==1)

{

currentRecord2.lastRecord=timer\_record.stop();

//println("entered"+currentRecord2.lastRecord);

n++; //n=2

}

win();

textSize(30);

text("Time: "+currentRecord2.lastRecord+"s",460,450);

//println("n2="+n);

currentRecord2.judge();

//println("currentrecord2="+currentRecord2.currentRecord);

//println("lastrecord2="+currentRecord2.lastRecord);

if(n==2)

{

if(key=='r'||key=='R')

{

key='a';

bgm2.pause();

ball.reset();

}

}

}

break;

default:break;

}

}

}

void keyReleased()

{

keystate1=0; //重置按键状态

keystate2=0;

keystate3=0;

keystate4=0;

}

void keyPressed()

{

mykey=mybut.listenbut(); //获取当前按键, 1、2为鼓面, 3、4为鼓边

}

void playsetup()

{

stroke(0); //边缘

strokeWeight(1); //边缘粗细

fill(100, 160);

rect(0, 150, 800, 150);

line(200, 150, 200, 300);

stroke(255);

strokeWeight(2);

fill(100);

ellipse(250, 220, 50, 50); //灰色敲击点

fill(0, 102, 153);

text(combo, 300, 50); //初始连击计数0

text(score, 300, 100); //初始得分0

mydrum.displaydrum(); //左鼓界面绘制

mydrum.hitten(mykey); //鼓面变化

serialsetup(); //串口控制

}

void act(int n) //每一帧的动作

{

db[n].showbeat(st[n]); //显示对应类型的鼓点

db[n].move(); //鼓点移动

}

void serialsetup() //串口敲击时鼓面变化处理

{

if (serialstate==1) //读取串口成功

{

sa=serial.read(); //读取串口的值

if (sa==49) // 1的ASCII码

{

bcolor=0; //颜色

mydrum.serialhitten(1); //左鼓面

frame1=8; //显示帧数

}

if (frame1>0) //鼓面变化持续一定帧数

{

fill(0);

arc(100, 225, 100, 100, HALF\_PI, HALF\_PI+PI);

frame1--;

}

if (sa==50)

{

bcolor=0;

mydrum.serialhitten(2); //右鼓面

frame2=8;

}

if (frame2>0)

{

fill(0);

arc(100, 225, 100, 100, 3\*HALF\_PI, 3\*HALF\_PI+PI);

frame2--;

}

if (sa==51)

{

bcolor=8;

mydrum.serialhitten(3); //左鼓边

frame3=8;

}

if (frame3>0)

{

stroke(0);

fill(#F01111);

arc(100, 225, 100, 100, HALF\_PI, HALF\_PI+PI);

frame3--;

}

if (sa==52)

{

bcolor=8;

mydrum.serialhitten(4); //右鼓边

frame4=8;

}

if (frame4>0)

{

stroke(0);

fill(#F01111);

arc(100, 225, 100, 100, 3\*HALF\_PI, 3\*HALF\_PI+PI);

frame4--;

}

}

}

void playgame()

{

combo=0;

score=0;

playstate=1; //游戏状态

bgm.setGain(-10); //背景音量

bgm.play(); //播放音乐

timer.start(); //计时器开始工作

bgmstate=1; //音乐状态为1(播放)

}

void firstread() //读取第一行数据

{

try {

line = reader.readLine(); //读取第一行

}

catch (IOException e) {

e.printStackTrace();

line = null;

}

String data[]=split(line, ','); //数据以逗号为分隔放入data数组

stime=int(data[0]); //出现时间

println(stime);

bstyle=int(data[1]); //鼓点类型

}

void songselect(int songn) //歌曲选择

{

if (songn==1)

{

song="1.mp3"; //读取音频文件

if (df==1)

songmap="1.1.txt"; //读取map

else

songmap="1.2.txt";

}

if (songn==2)

{

song="2.mp3"; //读取音频文件

if (df==1)

songmap="2.1.txt"; //读取map

else

songmap="2.2.txt";

}

if (songn==3)

{

song="3.mp3"; //读取音频文件

if (df==1)

songmap="3.1.txt"; //读取map

else

songmap="3.2.txt";

}

if (songn==4)

{

song="4.mp3"; //读取音频文件

if (df==1)

songmap="4.1.txt"; //读取map

else

songmap="4.2.txt";

}

}

class Drumbeat

{

float x=-50; //初始坐标

float y=225;

float speed=5; //每帧移动距离

int fx; //击中时坐标

int fy;

int sx; //miss时的坐标

int sy;

int tx; //终点坐标

int ty;

int dcolor; //敲击颜色

int combostate; //连击状态

int num; //鼓点编号

int missstate; //错误鼓点

void resetbeat(int nums)

{

x=825; //运动前初始坐标

y=225;

speed=5; //每帧移动5像素

combostate=0; //连击状态

num=nums;

missstate=0; //错误鼓点

}

float drumx() //返回当前鼓点x坐标

{

return x;

}

void showbeat(int s) //显示对应颜色鼓点

{

if (s==0||s==4)

{

dcolor=0;

stroke(255);

strokeWeight(5);

fill(#F01111); //红色

ellipse(x, y, 50, 50);

} else if (s==8||s==12)

{

dcolor=8;

stroke(255);

strokeWeight(5);

fill(#0A9CFA); //蓝色

ellipse(x, y, 50, 50);

}

}

void move() //鼓点移动

{

if (x>-50)

x=x-speed;

if (x-250<50&&x-250>-50)

//连击统计

if (dcolor==bcolor)

if (keyPressed&&combostate==0||serialstate==1&&sa!=-1)

{

imagestate=1; //激活图片

imageframe=8; //图片显示帧数

combo++; //连击统计数加一

combostate=1; //激活连击状态

fill(0, 102, 153);

text(combo, 100, 100); //显示连击统计数

score++;

}

if (x-250<-60&&combostate==0&&missstate==0) //连击中断

{

combo=0;

missstate=1;

}

}

}

class Drum //鼓面类

{

int x; //鼓面坐标

int y;

int rl; //左半径

int rr; //右半径

Drum(int a, int b, int c, int d) //构造函数

{

x=a;

y=b;

rl=c;

rr=d;

}

void displaydrum() //显示鼓面

{

fill(#F01111);

strokeWeight(8);

stroke(#0A9CFA);

arc(x, y, rl, rr, HALF\_PI, HALF\_PI+PI);

arc(x, y, rl, rr, 3\*HALF\_PI, 3\*HALF\_PI+PI);

}

void serialhitten(int u) //串口敲击判断

{

if (u==1)

{

fill(0);

arc(x, y, rl, rr, HALF\_PI, HALF\_PI+PI); //左鼓面

inside.trigger(); //播放鼓声

keystate1=1;

} else if (u==2)

{

fill(0);

arc(x, y, rl, rr, 3\*HALF\_PI, 3\*HALF\_PI+PI); //右鼓面

inside.trigger();

keystate2=1;

} else if (u==3)

{

stroke(0);

fill(#F01111);

arc(x, y, rl, rr, HALF\_PI, HALF\_PI+PI); //左鼓边

outside.trigger();

keystate3=1;

} else if (u==4)

{

stroke(0);

fill(#F01111);

arc(x, y, rl, rr, 3\*HALF\_PI, 3\*HALF\_PI+PI); //右鼓边

outside.trigger();

keystate4=1;

}

}

void hitten(int u) //按键敲击判断

{

if (keyPressed)

{

if (u==1)

{

fill(0);

arc(x, y, rl, rr, HALF\_PI, HALF\_PI+PI);

if (keystate1==0)

inside.trigger();

keystate1=1;

} else if (u==2)

{

fill(0);

arc(x, y, rl, rr, 3\*HALF\_PI, 3\*HALF\_PI+PI);

if (keystate2==0) inside.trigger();

keystate2=1;

} else if (u==3)

{

stroke(0);

fill(#F01111);

arc(x, y, rl, rr, HALF\_PI, HALF\_PI+PI);

if (keystate3==0)

outside.trigger();

keystate3=1;

} else if (u==4)

{

stroke(0);

fill(#F01111);

arc(x, y, rl, rr, 3\*HALF\_PI, 3\*HALF\_PI+PI);

if (keystate4==0)

outside.trigger();

keystate4=1;

}

}

}

}

class Mybutlisten //按键监听类

{

char inl; //左内鼓面

char inr; //右内鼓面

char outl; //左鼓边

char outr; //右鼓边

Mybutlisten(char a, char b, char c, char d) //设置按键

{

inl=a;

inr=b;

outl=c;

outr=d;

}

int listenbut() //监听按键

{

if (key==inl)

{

bcolor=0; //0代表红色, 8代表蓝色

return 1;

} else if (key==inr)

{

bcolor=0;

return 2;

} else if (key==outl)

{

bcolor=8;

return 3;

} else if (key==outr)

{

bcolor=8;

return 4;

} else

return 0;

}

}

class Timer

{

float savedtime; //计时器何时开始

int timestate=0;

void start() {

//当计时器开启, 它将当前时间以ms为单位存储下来

savedtime = millis();

timestate=1;

}

boolean showtime(float showtimes, float nt) //判断鼓点是否应该出现

{

float passedtime = nt;

if ((showtimes-940)<=passedtime&&timestate==1)

{

return true;

} else {

return false;

}

}

void timereset() //重置计时器

{

timestate=0;

savedtime=0;

}

float now() //返回计时器运行的时间

{

return millis()-savedtime;

}

}

class Mission1 //小球第一关地图

{

void display()

{

background(backimg); //背景

rectMode(CORNER); //障碍物形状位置

fill(borderColor);

noStroke();

rect(102, 279, 73, 700, 7);

rect(276, 0, 73, 650,7);

rect(451, 100, 73, 700,7);

rect(625, 0, 73, 700,7);

rect(0, 0, 800, 5);

rect(0, 795, 800, 5);

rect(0, 0, 5, 800);

rect(795, 0, 5, 880);

fill(255);

stroke(0);

}

}

class Mission2 //小球第二关地图

{

void display()

{

background(backimg);

fill(borderColor);

noStroke();

beginShape();

vertex(200,100);

vertex(795,100);

vertex(795,795);

vertex(100,795);

vertex(100,500);

vertex(200,500);

vertex(200,700);

vertex(700,700);

vertex(700,200);

vertex(200,200);

endShape(CLOSE);

beginShape();

vertex(5,300);

vertex(600,300);

vertex(600,600);

vertex(300,600);

vertex(300,500);

vertex(500,500);

vertex(500,400);

vertex(500,400);

vertex(5,400);

endShape(CLOSE);

rectMode(CORNER);

rect(0, 0, 800, 5);

rect(0, 795, 800, 5);

rect(0, 0, 5, 800);

rect(795, 0, 5, 880);

fill(255);

stroke(0);

}

}

class Ball { //小球参数

private int posx, posy;

private float radius;

private float dx;

private float dy;

private boolean border;

private color d;

Ball(int posx, int posy, int radius, color e ){ //小球坐标

this.posx=posx;

this.posy=posy;

this.radius=radius;

this.d=e;

}

void display() { //小球显示

fill(0, 100, 0);

ellipseMode(RADIUS);

ellipse(posx, posy, radius, radius);

fill(255);

posx\_timer=posx;

posy\_timer=posy;

radius\_timer=radius;

}

void move() { //遇障碍物速度减为0

if (isXLeftBorderDetective())

if (dx>0)

dx=0;

if (isXRightBorderDetective())

if (dx<0)

dx=0;

if (isYUpBorderDetective())

if (dy<0)

dy=0;

if (isYDownBorderDetective())

if (dy>0)

dy=0;

posx-=dx;

posy+=dy;

}

//通过键盘或压力传感器输入控制小球速度及运动方向

void update(float ax, float ay) {

if(keyPressed && (key == CODED)||serialstate==1)

{

if(serialstate==1)

{

sa=serial.read();

}

if(keyCode == LEFT||sa==49)

{

dx = 2;

}

if(keyCode == RIGHT||sa==50)

{

dx = -2;

}

if(keyCode == UP||sa==51)

{

dy = -2;

}

if(keyCode == DOWN||sa==52)

{

dy = 2;

}

}

move();

display();

}

void reset() //通关后小球恢复初始参数

{

posx=beginX;

posy=beginY;

n=0;

missionNo=0;

posx\_timer=0;

posy\_timer=0;

radius\_timer=0;

menu3=1;

menu4=0;

menu5=0;

}

//判断小球是否触界

boolean isXLeftBorderDetective() { //判断小球左侧是否触界

border=false;

color c;

c=get((int)(posx-radius-1), (int)posy);

if (d==c)

border=true;

return border;

}

boolean isXRightBorderDetective() { //判断小球右侧是否触界

border=false;

color c;

c=get((int)(posx+radius+1), (int)posy);

if (d==c)

border=true;

return border;

}

boolean isYUpBorderDetective() { //判断小球上侧是否触界

border=false;

color c;

c=get((int)(posx), (int)(posy-radius-1));

if (d==c)

border=true;

return border;

}

boolean isYDownBorderDetective() { //判断小球下侧是否触界

border=false;

color c;

c=get((int)(posx), (int)(posy+radius+1));

if (d==c)

border=true;

return border;

}

}

class Timer1 //计时器

{

int savedTime; //储存开始时间

int passedTime; //储存经过时长

Timer1() {

}

int start1() //开始计时

{

savedTime = millis();

return millis();

}

int stop() //结束计时

{

passedTime = millis()-startTime;

return passedTime/1000; //返回通关时间

}

}

void win() //显示通关画面

{

background(backimg);

fill(0,102,153);

textSize(25);

text("Press 'R' to return",20,50);

textSize(50);

text("FINISH",520,280);

}

class CURRENTRecord //保存记录

{

int lastRecord=10000;

int currentRecord=10000;

CURRENTRecord(){

}

void judge() //判断此次通关时间是否大于以前保存的记录

{

if(lastRecord<currentRecord)

currentRecord=lastRecord; //更新记录

}

}

# 第5章 节奏大师之疯狂打地鼠项目设计

### 5.2.1 主程序模块

#### 2.相关代码

#define LED1 13

#define LED2 12

#define LED3 11

#define LED4 10

#define LED5 9

#define LED6 A1

#define KEY1 2

#define KEY2 3

#define KEY3 4

#define KEY4 5

#define KEY5 6

#define KEY6 A2

int ledPin=A3;

int capval1,capval2,capval3,capval4,capval5,capval6,capval7,capval8;

int KEY1\_NUM = 0; //存放变量

int KEY2\_NUM = 0;

int KEY3\_NUM = 0;

int KEY4\_NUM = 0;

int KEY5\_NUM = 0;

int KEY6\_NUM = 0;

int a=1;

int b=0;

int c=1;

int flag=0;

int flagg=0;

void setup()

{

pinMode(LED1,OUTPUT); //定义LED为输出引脚

pinMode(LED2,OUTPUT);

pinMode(LED3,OUTPUT);

pinMode(LED4,OUTPUT);

pinMode(LED5,OUTPUT);

pinMode(LED6,OUTPUT);

pinMode(KEY1,INPUT\_PULLUP); //定义KEY为带上拉输入引脚

pinMode(KEY2,INPUT\_PULLUP);

pinMode(KEY3,INPUT\_PULLUP);

pinMode(KEY4,INPUT\_PULLUP);

pinMode(KEY5,INPUT\_PULLUP);

pinMode(KEY6,INPUT\_PULLUP);

pinMode(ledPin, OUTPUT);

}

void loop()

{

if(flag!=1){

switch(a) {

case 1: led='1'; a=0;c++;digitalWrite(LED1,1);b=1;

break;

case 2:led='2'; a=0;c++;digitalWrite(LED2,1);b=2;

break;

case 3:led='3'; a=0;c++;digitalWrite(LED3,1);b=3;

break;

case 4:;led='1'; a=0;c++;digitalWrite(LED1,1);b=1;

break;

case 5:led='1'; a=0;c++; digitalWrite(LED1,1);b=1;

break;

case 6:led='2'; a=0;c++;digitalWrite(LED2,1);b=2;

break;

case 7:led='3'; a=0;c++;digitalWrite(LED3,1);b=3;

break;

case 8:led='1'; a=0;c++;digitalWrite(LED1,1);b=1;

break;

case 9:led='3'; a=0;c++;digitalWrite(LED3,1);b=3;

break;

case 10:led='4';updateSensorData(); a=0;c++; digitalWrite(LED4,1);b=4;

break;

case 11:led='5'; a=0;c++;digitalWrite(LED5,1);b=5;

break;

case 12:led='3'; a=0;c++; digitalWrite(LED3,1);b=3;

break;

case 13:led='4'; a=0;c++;digitalWrite(LED4,1);b=4;

break;

case 14:led='5'; a=0;c++;digitalWrite(LED5,1);b=5;

break;

case 15:led='5'; a=0;c++;digitalWrite(LED5,1);b=5;

break;

case 16:led='6'; a=0;c++;digitalWrite(LED6,1);b=6;

break;

case 17:led='5'; a=0;c++;digitalWrite(LED5,1);b=5;

break;

case 18:led='4'; a=0;c++;digitalWrite(LED4,1);b=4;

break;

case 19:led='3'; a=0;c++;digitalWrite(LED3,1);b=3;

break;

case 20:led='1'; a=0;c++;digitalWrite(LED1,1);b=1;

break;

case 21:led='5'; a=0;c++;digitalWrite(LED5,1);b=5;

break;

case 22:led='6'; a=0;c++;digitalWrite(LED6,1);b=6;

break;

case 23:led='5'; a=0;c++;digitalWrite(LED5,1);b=5;

break;

case 24:led='4'; a=0;c++;digitalWrite(LED4,1);b=4;

break;

case 25:led='3'; a=0;c++;digitalWrite(LED3,1);b=3;

break;

case 26:led='1'; a=0;c++;digitalWrite(LED1,1);b=1;

break;

case 27:led='3'; a=0;c++;digitalWrite(LED3,1);b=3;

break;

case 28:led='5'; a=0;c++;digitalWrite(LED5,1);b=5;

break;

case 29:led='1'; a=0;c++;digitalWrite(LED1,1);b=1;

break;

case 30:led='3'; a=0;c++;digitalWrite(LED3,1);b=3;

break;

case 31:led='5'; a=0;c++;digitalWrite(LED5,1);b=5;

break;

case 32:led='1';a=0;c++;digitalWrite(LED1,1);b=1;

break;

default:

break;

}

}

if(a==33)

{

flag=1;

led='8';

a++;

}

switch(b){

case 1: ScanKey1(); break; //按键扫描程序，当按下时，子程序会修改KEY\_NUM的值

case 2: ScanKey2(); break;

case 3: ScanKey3(); break;

case 4: ScanKey4(); break;

case 5: ScanKey5(); break;

case 6: ScanKey6(); break;

}

if(KEY1\_NUM == 1) //是否按下，如果<span style="font-family:Arial, Helvetica, sans-serif;">ScanKey函数扫描到按键就会设置KEY\_NUM值为1</span>

{

digitalWrite(LED1,0); //LED1灭

}

if(KEY2\_NUM == 1)

{

digitalWrite(LED2,0); //LED2灭

}

if(KEY3\_NUM == 1)

{

digitalWrite(LED3,0); //LED4灭

}

if(KEY4\_NUM == 1)

{

digitalWrite(LED4,0); //LED4灭

}

if(KEY5\_NUM == 1)

{

digitalWrite(LED5,0); //LED5灭

}

if(KEY6\_NUM == 1)

{

digitalWrite(LED6,0); //LED6灭

}

}

void ScanKey1() //按键扫描程序

{

KEY1\_NUM = 0; //清空变量

if(digitalRead(KEY1) == LOW) //有按键按下

{

delay(20); //延时去抖动

if(digitalRead(KEY1) == LOW) //有按键按下

{

KEY1\_NUM = 1; //变量设置为1

a=c;

while(digitalRead(KEY1) == LOW){

tone(ledPin, 262, 10);

}; //等待按键松手

}

}

}

void ScanKey2() //按键扫描程序

{

KEY2\_NUM = 0; //清空变量

if(digitalRead(KEY2) == LOW) //有按键按下

{

delay(20); //延时去抖动

if(digitalRead(KEY2) == LOW) //有按键按下

{

KEY2\_NUM = 1; //变量设置为1

a=c;

while(digitalRead(KEY2) == LOW){

tone(ledPin, 294, 10);

}; //等待按键松手

}

}

}

void ScanKey3() //按键扫描程序

{

KEY3\_NUM = 0; //清空变量

if(digitalRead(KEY3) == LOW) //有按键按下

{

delay(20); //延时去抖动

if(digitalRead(KEY3) == LOW) //有按键按下

{

KEY3\_NUM = 1; //变量设置为1

a=c;

while(digitalRead(KEY3) == LOW){

tone(ledPin, 330, 10);

}; //等待按键松手

}

}

}

void ScanKey4() //按键扫描程序

{

KEY4\_NUM = 0; //清空变量

if(digitalRead(KEY4) == LOW) //有按键按下

{

delay(20); //延时去抖动

if(digitalRead(KEY4) == LOW) //有按键按下

{

KEY4\_NUM = 1; //变量设置为1

a=c;

while(digitalRead(KEY4) == LOW){

tone(ledPin, 350, 10);

}; //等待按键松手

}

}

}

void ScanKey5() //按键扫描程序

{

KEY5\_NUM = 0; //清空变量

if(digitalRead(KEY5) == LOW) //有按键按下

{

delay(20); //延时去抖动

if(digitalRead(KEY5) == LOW) //有按键按下

{

KEY5\_NUM = 1; //变量设置为1

a=c;

while(digitalRead(KEY5) == LOW){

tone(ledPin, 393, 10);

}; //等待按键松手

}

}

}

void ScanKey6() //按键扫描程序

{

KEY6\_NUM = 0; //清空变量

if(digitalRead(KEY6) == LOW) //有按键按下

{

delay(20); //延时去抖动

if(digitalRead(KEY6) == LOW) //有按键按下

{

KEY6\_NUM = 1; //变量设置为1

a=c;

while(digitalRead(KEY6) == LOW){

tone(ledPin, 441, 10);

}; //等待按键松手

}

}

}

### 5.2.2 ESP8266模块

#### 2.相关代码

#include <Wire.h> //调用库文件

#include "./ESP8266.h"

#define SSID "iPhone"

#define PASSWORD "12345678"

#define WLAN\_SECURITY WLAN\_SEC\_WPA2

#define IDLE\_TIMEOUT\_MS 3000 // 无数据等待时间

#define HOST\_NAME "172.20.10.4" //可改成自己的服务器地址和端口

#define HOST\_PORT (8081)

#include <SoftwareSerial.h>

SoftwareSerial mySerial(8, 7); //定义软串口，引脚7为RX，引脚8为TX

ESP8266 wifi(mySerial);

//ESP8266 wifi(Serial1); //定义一个ESP8266的对象

unsigned long net\_time1 = millis(); //数据上传服务器时间

char led='7';

String postString; //用于存储发送数据的字符串

//String jsonToSend; //用于存储发送的JSON格式参数

void setup()

{

//初始化串口波特率

Wire.begin();

Serial.begin(9600);

while(!Serial);

//ESP8266初始化

Serial.print("setup begin\r\n");

Serial.print("FW Version:");

Serial.println(wifi.getVersion().c\_str());

if (wifi.setOprToStationSoftAP()) {

Serial.print("to station + softap ok\r\n");

} else {

Serial.print("to station + softap err\r\n");

}

if (wifi.joinAP(SSID, PASSWORD)) { //加入无线网

Serial.print("Join AP success\r\n");

Serial.print("IP: ");

Serial.println(wifi.getLocalIP().c\_str());

} else {

Serial.print("Join AP failure\r\n");

}

if (wifi.disableMUX()) {

Serial.print("single ok\r\n");

} else {

Serial.print("single err\r\n");

}

Serial.print("setup end\r\n");

if (wifi.createTCP(HOST\_NAME, HOST\_PORT)) {

//建立TCP连接，如果失败，不能发送该数据

Serial.print("create tcp ok\r\n");}

}

void loop()

{

updateSensorData();

}

void updateSensorData() {

//postString将存储传输请求，格式很重要

postString = "POST "; //POST发送方式，后要有空格

postString += "/process\_post?led="; //接口process

postString += led; //要发送的数据

postString += " HTTP/1.1"; //空格+传输协议

postString += "\r\n";

postString += "Host: "; //Host：+空格

postString += HOST\_NAME;

postString += "\r\n";

postString += "Content-Type: application/x-www-form-urlencoded\r\n";//编码类型

postString += "\r\n"; //不可删除

const char \*postArray = postString.c\_str(); //将str转化为char数组

Serial.println(postArray);

wifi.send((const uint8\_t\*)postArray, strlen(postArray));

//send发送命令，参数必须是这两种格式，尤其是(const uint8\_t\*)

Serial.println("send success");

}

### 5.2.3 服务器模块

#### 相关代码

var express = require('express');

var app = express();

var bodyParser = require('body-parser');

var fs = require('fs');

var util = require('util');

// 创建 application/x-www-form-urlencoded 编码解析

var urlencodedParser = bodyParser.urlencoded({ extended: false })

app.use(express.static('public'));

var aj;

var server = app.listen(8081, function() {

var host = server.address().address

var port = server.address().port

console.log("应用实例，访问地址为 http://%s:%s", host, port)

})

var io = require('socket.io').listen(server);

io.sockets.on('connection', function(socket) {

console.log('User connected');

socket.on('disconnect', function() {

console.log('User disconnected');

});

});

app.post('/process\_post', urlencodedParser, function(req, res) {

// 输出 JSON 格式

var response = {

"led": req.query.led

};

console.log(response);

console.log("get post/r/n");

io.sockets.emit('message', { text: response.led });

res.end(JSON.stringify(response));

})

### 5.2.3 前端模块

#### 2.相关代码

##### 1）CSS部分

\* {

margin: 0;

padding: 0;

}

.first {

margin-left: auto;

margin-right: auto;

}

.first img {

position: absolute;

width: 100%;

text-align: center;

z-index: -1;

}

li {

list-style: none;

}

.first ul {

position: fixed;

left: 43%;

top: 37%;

}

button {

border: 0px;

background-color: transparent;

background-image: none;

font-family: 华文彩云;

font-size: 2em;

line-height: 2em;

color: rgb(226, 246, 10);

}

button:focus {

outline: none;

}

button:hover {

font-size: 2.2em;

}

.options {

display: none;

}

#btnback {

display: none;

position: absolute;

left: 18%;

top: 7%;

}

.dishu\_bd li,

.dishu li,

.hammer li {

position: absolute;

display: none;

z-index: -1;

}

.dishu li img {

width: 135%;

}

#one\_1 {

left: 20%;

top: 26%;

}

#two\_1 {

left: 43%;

top: 26%;

}

#three\_1 {

left: 68%;

top: 26%;

}

#four\_1 {

left: 15%;

top: 73%;

}

#five\_1 {

left: 43%;

top: 73%;

}

#six\_1 {

left: 71%;

top: 73%;

}

#one\_2 {

left: -8%;

top: -18%;

}

#two\_2 {

left: 15.5%;

top: -18%;

}

#three\_2 {

left: 40%;

top: -17.5%;

}

#four\_2 {

left: -12%;

top: 30%;

}

#five\_2 {

left: 16%;

top: 31%;

}

#six\_2 {

left: 43%;

top: 31%;

}

#one\_3 {

left: -8%;

top: -12%;

}

#two\_3 {

left: 15.5%;

top: -12%;

}

#three\_3 {

left: 39.5%;

top: -12%;

}

#four\_3 {

left: -12%;

top: 37%;

}

#five\_3 {

left: 16%;

top: 37%;

}

#six\_3 {

left: 43%;

top: 37%;

}

.time {

position: absolute;

right: 10%;

top: 8%;

font-family: 华文彩云;

font-size: 5em;

line-height: 2em;

color: white;

}

#recordd {

position: absolute;

right: 48%;

top: 40%;

font-family: 华文彩云;

font-size: 5em;

line-height: 2em;

color: rgb(226, 246, 10);

display: none;

}

##### 2）HTML部分

<!DOCTYPE html>

<html lang="en">

<head>

<title>节奏大师之疯狂打地鼠</title>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link href="css/dishu.css" rel="stylesheet">

</head>

<script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.0.3/socket.io.js"></script>

<script src="dishu.js"></script>

<body>

<main class="first">

<img src="photo/bgd.jpg" alt="">

<ul class="main\_menu" id="ulmenu">

<li><button type="button" onclick="Begin();">游戏开始</button></li>

<li><button type="button" onclick="Options();">查看记录</button></li>

</ul>

<button type="button" id="btnback" onclick="Back();">返回</button>

</main>

<main class="second">

<ul class="dishu">

<li id="one\_1"><img src="photo/dishu.png" alt=""></li>

<li id="two\_1"><img src="photo/dishu.png" alt=""></li>

<li id="three\_1"><img src="photo/dishu.png" alt=""></li>

<li id="four\_1"><img src="photo/dishu.png" alt=""></li>

<li id="five\_1"><img src="photo/dishu.png" alt=""></li>

<li id="six\_1"><img src="photo/dishu.png" alt=""></li>

</ul>

<ul class="dishu\_bd">

<li id="one\_2"><img src="photo/dishu\_bd.png" alt=""></li>

<li id="two\_2"><img src="photo/dishu\_bd.png" alt=""></li>

<li id="three\_2"><img src="photo/dishu\_bd.png" alt=""></li>

<li id="four\_2"><img src="photo/dishu\_bd.png" alt=""></li>

<li id="five\_2"><img src="photo/dishu\_bd.png" alt=""></li>

<li id="six\_2"><img src="photo/dishu\_bd.png" alt=""></li>

</ul>

<ul class="hammer">

<li id="one\_3"><img src="photo/hammer.png" alt=""></li>

<li id="two\_3"><img src="photo/hammer.png" alt=""></li>

<li id="three\_3"><img src="photo/hammer.png" alt=""></li>

<li id="four\_3"><img src="photo/hammer.png" alt=""></li>

<li id="five\_3"><img src="photo/hammer.png" alt=""></li>

<li id="six\_3"><img src="photo/hammer.png" alt=""></li>

</ul>

<p class="time" id="timer">0</p>

<p id="recordd">暂无</p>

</main>

</body>

</html>

##### 3）JavaScript部分

var aj = -1;

var led;

var jishi = 0;

var record = 500;

var socket = io.connect('http://127.0.0.1:8081');

var x;

socket.on('message', function(data) {

led = data.text;

console.log(led);

var one\_1 = document.getElementById('one\_1');

var two\_1 = document.getElementById('two\_1');

var three\_1 = document.getElementById('three\_1');

var four\_1 = document.getElementById('four\_1');

var five\_1 = document.getElementById('five\_1');

var six\_1 = document.getElementById('six\_1');

var one\_2 = document.getElementById('one\_2');

var two\_2 = document.getElementById('two\_2');

var three\_2 = document.getElementById('three\_2');

var four\_2 = document.getElementById('four\_2');

var five\_2 = document.getElementById('five\_2');

var six\_2 = document.getElementById('six\_2');

var one\_3 = document.getElementById('one\_3');

var two\_3 = document.getElementById('two\_3');

var three\_3 = document.getElementById('three\_3');

var four\_3 = document.getElementById('four\_3');

var five\_3 = document.getElementById('five\_3');

var six\_3 = document.getElementById('six\_3');

switch (Number(led)) {

case 1:

switch (aj) {

case 1:

one\_1.style.display = 'none';

one\_2.style.display = 'block';

one\_3.style.display = 'block';

var t = setTimeout("one\_2.style.display = 'none';one\_3.style.display = 'none';", 300)

break;

case 2:

two\_1.style.display = 'none';

two\_2.style.display = 'block';

two\_3.style.display = 'block';

var t = setTimeout("two.style.display = 'none';two\_3.style.display = 'none';", 300)

break;

case 3:

three\_1.style.display = 'none';

three\_2.style.display = 'block';

three\_3.style.display = 'block';

var t = setTimeout("three\_2.style.display = 'none';three\_3.style.display = 'none';", 300)

break;

case 4:

four\_1.style.display = 'none';

four\_2.style.display = 'block';

four\_3.style.display = 'block';

var t = setTimeout("four\_2.style.display = 'none';four\_3.style.display = 'none';", 300)

break;

case 5:

five\_1.style.display = 'none';

five\_2.style.display = 'block';

five\_3.style.display = 'block';

var t = setTimeout("five\_2.style.display = 'none';five\_3.style.display = 'none';", 300)

break;

case 6:

six\_1.style.display = 'none';

six\_2.style.display = 'block';

six\_3.style.display = 'block';

var t = setTimeout("six\_2.style.display = 'none';six\_3.style.display = 'none';", 300)

break;

default:

break;

}

one\_1.style.display = 'block';

aj = 1;

break;

case 2:

switch (aj) {

case 1:

one\_1.style.display = 'none';

one\_2.style.display = 'block';

one\_3.style.display = 'block';

var t = setTimeout("one\_2.style.display = 'none';one\_3.style.display = 'none';", 300)

break;

case 2:

two\_1.style.display = 'none';

two\_2.style.display = 'block';

two\_3.style.display = 'block';

var t = setTimeout("two\_2.style.display = 'none';two\_3.style.display = 'none';", 300)

break;

case 3:

three\_1.style.display = 'none';

three\_2.style.display = 'block';

three\_3.style.display = 'block';

var t = setTimeout("three\_2.style.display = 'none';three\_3.style.display = 'none';", 300)

break;

case 4:

four\_1.style.display = 'none';

four\_2.style.display = 'block';

four\_3.style.display = 'block';

var t = setTimeout("four\_2.style.display = 'none';four\_3.style.display = 'none';", 300)

break;

case 5:

five\_1.style.display = 'none';

five\_2.style.display = 'block';

five\_3.style.display = 'block';

var t = setTimeout("five\_2.style.display = 'none';five\_3.style.display = 'none';", 300)

break;

case 6:

six\_1.style.display = 'none';

six\_2.style.display = 'block';

six\_3.style.display = 'block';

var t = setTimeout("six\_2.style.display = 'none';six\_3.style.display = 'none';", 300)

break;

default:

break;

}

two\_1.style.display = 'block';

aj = 2;

break;

case 3:

switch (aj) {

case 1:

one\_1.style.display = 'none';

one\_2.style.display = 'block';

one\_3.style.display = 'block';

var t = setTimeout("one\_2.style.display = 'none';one\_3.style.display = 'none';", 300)

break;

case 2:

two\_1.style.display = 'none';

two\_2.style.display = 'block';

two\_3.style.display = 'block';

var t = setTimeout("two\_2.style.display = 'none';two\_3.style.display = 'none';", 300)

break;

case 3:

three\_1.style.display = 'none';

three\_2.style.display = 'block';

three\_3.style.display = 'block';

var t = setTimeout("three\_2.style.display = 'none';three\_3.style.display = 'none';", 300)

break;

case 4:

four\_1.style.display = 'none';

four\_2.style.display = 'block';

four\_3.style.display = 'block';

var t = setTimeout("four\_2.style.display = 'none';four\_3.style.display = 'none';", 300)

break;

case 5:

five\_1.style.display = 'none';

five\_2.style.display = 'block';

five\_3.style.display = 'block';

var t = setTimeout("five\_2.style.display = 'none';five\_3.style.display = 'none';", 300)

break;

case 6:

six\_1.style.display = 'none';

six\_2.style.display = 'block';

six\_3.style.display = 'block';

var t = setTimeout("six\_2.style.display = 'none';six\_3.style.display = 'none';", 300)

break;

default:

break;

}

three\_1.style.display = 'block';

aj = 3;

break;

case 4:

switch (aj) {

case 1:

one\_1.style.display = 'none';

one\_2.style.display = 'block';

one\_3.style.display = 'block';

var t = setTimeout("one\_2.style.display = 'none';one\_3.style.display = 'none';", 300)

break;

case 2:

two\_1.style.display = 'none';

two\_2.style.display = 'block';

two\_3.style.display = 'block';

var t = setTimeout("two\_2.style.display = 'none';two\_3.style.display = 'none';", 300)

break;

case 3:

three\_1.style.display = 'none';

three\_2.style.display = 'block';

three\_3.style.display = 'block';

var t = setTimeout("three\_2.style.display = 'none';three\_3.style.display = 'none';", 300)

break;

case 4:

four\_1.style.display = 'none';

four\_2.style.display = 'block';

four\_3.style.display = 'block';

var t = setTimeout("four\_2.style.display = 'none';four\_3.style.display = 'none';", 300)

break;

case 5:

five\_1.style.display = 'none';

five\_2.style.display = 'block';

five\_3.style.display = 'block';

var t = setTimeout("five\_2.style.display = 'none';five\_3.style.display = 'none';", 300)

break;

case 6:

six\_1.style.display = 'none';

six\_2.style.display = 'block';

six\_3.style.display = 'block';

var t = setTimeout("six\_2.style.display = 'none';six\_3.style.display = 'none';", 300)

break;

default:

break;

}

four\_1.style.display = 'block';

aj = 4;

break;

case 5:

switch (aj) {

case 1:

one\_1.style.display = 'none';

one\_2.style.display = 'block';

one\_3.style.display = 'block';

var t = setTimeout("one\_2.style.display = 'none';one\_3.style.display = 'none';", 300)

break;

case 2:

two\_1.style.display = 'none';

two\_2.style.display = 'block';

two\_3.style.display = 'block';

var t = setTimeout("two\_2.style.display = 'none';two\_3.style.display = 'none';", 300)

break;

case 3:

three\_1.style.display = 'none';

three\_2.style.display = 'block';

three\_3.style.display = 'block';

var t = setTimeout("three\_2.style.display = 'none';three\_3.style.display = 'none';", 300)

break;

case 4:

four\_1.style.display = 'none';

four\_2.style.display = 'block';

four\_3.style.display = 'block';

var t = setTimeout("four\_2.style.display = 'none';four\_3.style.display = 'none';", 300)

break;

case 5:

five\_1.style.display = 'none';

five\_2.style.display = 'block';

five\_3.style.display = 'block';

var t = setTimeout("five\_2.style.display = 'none';five\_3.style.display = 'none';", 300)

break;

case 6:

six\_1.style.display = 'none';

six\_2.style.display = 'block';

six\_3.style.display = 'block';

var t = setTimeout("six\_2.style.display = 'none';six\_3.style.display = 'none';", 300)

break;

default:

break;

}

five\_1.style.display = 'block';

aj = 5;

break;

case 6:

switch (aj) {

case 1:

one\_1.style.display = 'none';

one\_2.style.display = 'block';

one\_3.style.display = 'block';

var t = setTimeout("one\_2.style.display = 'none';one\_3.style.display = 'none';", 300)

break;

case 2:

two\_1.style.display = 'none';

two\_2.style.display = 'block';

two\_3.style.display = 'block';

var t = setTimeout("two\_2.style.display = 'none';two\_3.style.display = 'none';", 300)

break;

case 3:

three\_1.style.display = 'none';

three\_2.style.display = 'block';

three\_3.style.display = 'block';

var t = setTimeout("three\_2.style.display = 'none';three\_3.style.display = 'none';", 300)

break;

case 4:

four\_1.style.display = 'none';

four\_2.style.display = 'block';

four\_3.style.display = 'block';

var t = setTimeout("four\_2.style.display = 'none';four\_3.style.display = 'none';", 300)

break;

case 5:

five\_1.style.display = 'none';

five\_2.style.display = 'block';

five\_3.style.display = 'block';

var t = setTimeout("five\_2.style.display = 'none';five\_3.style.display = 'none';", 300)

break;

case 6:

six\_1.style.display = 'none';

six\_2.style.display = 'block';

six\_3.style.display = 'block';

var t = setTimeout("six\_2.style.display = 'none';six\_3.style.display = 'none';", 300)

break;

default:

break;

}

six\_1.style.display = 'block';

aj = 6;

break;

case 7:

x = setInterval("clk()", 1000);

break;

case 8:

window.clearInterval(x);

if (record >= jishi) {

record = jishi;

}

jishi = 0;

switch (aj) {

case 1:

one\_1.style.display = 'none';

one\_2.style.display = 'block';

one\_3.style.display = 'block';

var t = setTimeout("one\_2.style.display = 'none';one\_3.style.display = 'none';", 300)

break;

case 2:

two\_1.style.display = 'none';

two\_2.style.display = 'block';

two\_3.style.display = 'block';

var t = setTimeout("two\_2.style.display = 'none';two\_3.style.display = 'none';", 300)

break;

case 3:

three\_1.style.display = 'none';

three\_2.style.display = 'block';

three\_3.style.display = 'block';

var t = setTimeout("three\_2.style.display = 'none';three\_3.style.display = 'none';", 300)

break;

case 4:

four\_1.style.display = 'none';

four\_2.style.display = 'block';

four\_3.style.display = 'block';

var t = setTimeout("four\_2.style.display = 'none';four\_3.style.display = 'none';", 300)

break;

case 5:

five\_1.style.display = 'none';

five\_2.style.display = 'block';

five\_3.style.display = 'block';

var t = setTimeout("five\_2.style.display = 'none';five\_3.style.display = 'none';", 300)

break;

case 6:

six\_1.style.display = 'none';

six\_2.style.display = 'block';

six\_3.style.display = 'block';

var t = setTimeout("six\_2.style.display = 'none';six\_3.style.display = 'none';", 300)

break;

default:

break;

}

aj = 8;

break;

}

})

function Begin() {

var ullist = document.getElementById('ulmenu');

var btnback = document.getElementById('btnback');

var btnback = document.getElementById('btnback');

ullist.style.display = 'none';

btnback.style.display = 'block';

console.log('ok');

}

function Options() {

var btnback = document.getElementById('btnback');

var ullist = document.getElementById('ulmenu');

var recordd = document.getElementById('recordd');

ullist.style.display = 'none';

btnback.style.display = 'block';

recordd.innerHTML = record;

recordd.style.display = 'block';

console.log('ok');

}

function Back() {

var btnback = document.getElementById('btnback');

var ullist = document.getElementById('ulmenu');

var recordd = document.getElementById('recordd');

ullist.style.display = 'block';

btnback.style.display = 'none';

recordd.style.display = 'none';

}

function clk() {

var timer = document.getElementById('timer');

jishi++;

timer.innerHTML = jishi;

}

# 第6章 基于红外测距的虚拟电子琴项目设计

### 6.2.1 主程序模块

#### 2.相关代码

char GP2D12[4];

char a,b;

int m;

int val;

int FlagFree,FlagHalffree,FlagPlayer,FlagCreative,FlagSupervision; //五个模式的执行标志flag

int flag\_halffree\_finish;

int distance\_key; //测距回传对应的琴键音

int distance\_key1,distance\_key2;

int key\_save[4];

unsigned long time1;

unsigned long time2;

unsigned long playtime;

void loop(void) {

delay(5000);

m=Serial.parseInt();//读取

if(m==1)//选择模式1，对应flag置1，其余置0

{

FlagFree=1;

FlagHalffree=0;

FlagPlayer=0;

Serial.println("Free Playing mode...");

Serial.println("Do you want to record it?"); //选择创作模式

Serial.println("1 Yes.");

Serial.println("2 No.");

delay(5000);

m=Serial.parseInt();

if(m==1) //选中创作模式

{

FlagCreative=1;

FlagFree=1;

Serial.println("Free playing mode with recording...");

Serial.println("Input 9 for re-select.");//复位

}

else if(m==2) //不选创作模式

{

FlagCreative=0;

FlagFree=1;

Serial.println("Free playing mode without recording...");

Serial.println("Input 9 for re-select.");

}

else

{

FlagFree=0;//报错，重选

m=1;

Serial.println("Input error!");

}

}

else if(m==2)//选中半自由模式，对应flag置1，其余置0

{

FlagFree=0;

FlagHalffree=1;

FlagPlayer=0;

Serial.println("Do you want to be supervised?");//选择监督模式

Serial.println("1 Yes.");

Serial.println("2 No.");

delay(5000);

m=Serial.parseInt();

if(m==1) //选中监督模式

{

FlagSupervision=1;

FlagHalffree=1;

Serial.println("Free playing mode with being supervised...");

Serial.println("Input 9 for re-select.");

}

else if(m==2) //不选监督模式

{

FlagSupervision=0;

FlagHalffree=1;

Serial.println("Free playing mode without being supervised...");

Serial.println("Input 9 for re-select.");

}

else //报错，重选

{

FlagHalffree=0;

m=2;

Serial.println("Input error!");

Serial.println("Do you want to be supervised?");

Serial.println("1 Yes.");

Serial.println("2 No.");

}

}

else if(m==3)//选中播放模式，对应flag置1，其余置0

{

FlagFree=0;

FlagHalffree=0;

FlagPlayer=1;

Serial.println("Player mode...");

Serial.println("Input 9 for re-select.");

}

else//报错，重选

{

FlagFree=0;

FlagHalffree=0;

FlagPlayer=0;

Serial.println("Input error! Please re-select.");

Serial.println("1.Free playing mode");

Serial.println("2.Semi-free playing mode");

Serial.println("3.player");

}

### 6.2.2 SD卡读写模块

#### 2.相关代码

SD卡初始化：

#include <SimpleSDAudio.h>

void DirCallback(char \*buf) {

Serial.println(buf);

}

#define BIGBUFSIZE (2\*512)

uint8\_t bigbuf[BIGBUFSIZE];

int freeRam () {

extern int \_\_heap\_start, \*\_\_brkval;

int v;

return (int) &v - (\_\_brkval == 0 ? (int) &\_\_heap\_start : (int) \_\_brkval);

}

void setup()

{

Serial.begin(9600);

while (!Serial) {

;

}

Serial.print(F("Free Ram: "));

Serial.println(freeRam());

SdPlay.setWorkBuffer(bigbuf, BIGBUFSIZE);

Serial.print(F("\nInitializing SimpleSDAudio V" SSDA\_VERSIONSTRING " ..."));//初始化SD卡

if (!SdPlay.init(SSDA\_MODE\_FULLRATE | SSDA\_MODE\_MONO | SSDA\_MODE\_AUTOWORKER)) { //初始化失败

Serial.println(F("initialization failed. Things to check:"));

Serial.println(F("\* is a card is inserted?"));

Serial.println(F("\* Is your wiring correct?"));

Serial.println(F("\* maybe you need to change the chipSelect pin to match your shield or module?"));

Serial.print(F("Error code: "));

Serial.println(SdPlay.getLastError());

while(1);

}

else {

Serial.println(F("Welcome to the Visual Keyboard.")); //初始化成功，进入模式选择

Serial.println("\*Please select a function\*"); // 选择自由/半自由/播放

Serial.println("1.Free playing mode");

Serial.println("2.Semi-free playing mode");

Serial.println("3.player");

}

}

//播放琴键音函数：

void define\_key(int key)//琴键音定义函数，对应C大调3个八度

{

if(key==21)//低音1

{

Serial.print("C21 ");

SdPlay.setFile("C21.AFM");

SdPlay.play();

}

else if(key==22)//低音2

{

Serial.print("C22 ");

SdPlay.setFile("C22.AFM");

SdPlay.play();

}

else if(key==23)//低音3

{

Serial.print("C23 ");

SdPlay.setFile("C23.AFM");

SdPlay.play();

}

else if(key==24)//低音4

{

Serial.print("C24 ");

SdPlay.setFile("C24.AFM");

SdPlay.play();

}

else if(key==25)//低音5

{

Serial.print("C25 ");

SdPlay.setFile("C25.AFM");

SdPlay.play();

}

else if(key==26)//低音6

{

Serial.print("C26 ");

SdPlay.setFile("C26.AFM");

SdPlay.play();

}

else if(key==27)//低音7

{

Serial.print("C27 ");

SdPlay.setFile("C27.AFM");

SdPlay.play();

}

else if(key==31)//中音1

{

Serial.print("C31 ");

SdPlay.setFile("C31.AFM");

SdPlay.play();

}

else if(key==32)//中音2

{

Serial.print("C32 ");

SdPlay.setFile("C32.AFM");

SdPlay.play();

}

else if(key==33)//中音3

{

Serial.print("C33 ");

SdPlay.setFile("C33.AFM");

SdPlay.play();

}

else if(key==34)//中音4

{

Serial.print("C34 ");

SdPlay.setFile("C34.AFM");

SdPlay.play();

}

else if(key==35)//中音5

{

Serial.print("C35 ");

SdPlay.setFile("C35.AFM");

SdPlay.play();

}

else if(key==36)//中音6

{

Serial.print("C36 ");

SdPlay.setFile("C36.AFM");

SdPlay.play();

}

else if(key==37)//中音7

{

Serial.print("C37 ");

SdPlay.setFile("C37.AFM");

SdPlay.play();

}

else if(key==41)//高音1

{

Serial.print("C41 ");

SdPlay.setFile("C41.AFM");

SdPlay.play();

}

else if(key==42)//高音2

{

Serial.print("C42 ");

SdPlay.setFile("C42.AFM");

SdPlay.play();

}

else if(key==43)//高音3

{

Serial.print("C43 ");

SdPlay.setFile("C43.AFM");

SdPlay.play();

}

else if(key==44)//高音4

{

Serial.print("C44 ");

SdPlay.setFile("C44.AFM");

SdPlay.play();

}

else if(key==45)//高音5

{

Serial.print("C45 ");

SdPlay.setFile("C45.AFM");

SdPlay.play();

}

else if(key==46)//高音6

{

Serial.print("C46 ");

SdPlay.setFile("C46.AFM");

SdPlay.play();

}

else if(key==47)//高音7

{

Serial.print("C47 ");

SdPlay.setFile("C47.AFM");

SdPlay.play();

}

else if(key==100)//结束/报错音

{

if(FlagSupervision)

Serial.print("wrong");

else

Serial.print("finished");

SdPlay.setFile("C100.AFM");

SdPlay.play();

}

else Serial.println("not found");

//else Serial.println(SdPlay.getLastError());

}

### 6.2.3 红外测距模块

#### 2.相关代码

float read\_gp2d12\_range(byte pin) //一次测距函数

{

int tmp;

tmp = analogRead(pin);//读取模拟引脚电压值

if (tmp < 3)return -1;

return (6787.0 /((float)tmp - 3.0)) - 4.0;//使用测距函数转化为距离

}

int ave\_gp2d12\_range(byte pin)//平均测距函数

{

int a,b;

int val[5];

int ave\_dis,all\_dis=0;

int GP2D12;

int Max,Min;

for(int i=0;i<5;i++)//5次测距

{

GP2D12=read\_gp2d12\_range(pin);//调用一次测距函数

a=GP2D12/10;//转化为整形

b=GP2D12%10;

val[i]=a\*10+b;

all\_dis=all\_dis+val[i];//求和

if(i==0)

{

Max=val[0];

Min=val[0];

}

else

{

if(Max<val[i]) Max=val[i];//比较出最大值、最小值

if(Min>val[i]) Min=val[i];

}

delay(10);

}

ave\_dis=(all\_dis-Max-Min)/3;//余下3个值取平均

return ave\_dis;

}

int define\_distance(int i,int val)//琴键音、距离转换函数

{

if(i==0)//第一个传感器

{

if(val>=4.5&&val<=40.5)//有效距离

{

if(val>4.5&&val<9.5) distance\_key=100;//每个有效区域对应的琴键音

else if(val>9.5&&val<14.5) distance\_key=21;

else if(val>14.5&&val<19.5) distance\_key=22;

else if(val>19.5&&val<24.5) distance\_key=23;

else if(val>24.5&&val<29.5) distance\_key=24;

else if(val>29.5&&val<32.5) distance\_key=25;

else if(val>32.5&&val<40.5) distance\_key=26;

}

else distance\_key=-1;//回传无效琴键音

if(distance\_key==key\_save[0])//判断状态是否变化

distance\_key=-1;//不变化回传无效琴键音

else key\_save[0]=distance\_key;//若变化回传当前琴键音，并更新状态

}

else if(i==1)//第二个传感器

{

if(val>=4.5&&val<=44.5)

{

if(val>4.5&&val<9.5) distance\_key=27;

else if(val>9.5&&val<14.5) distance\_key=31;

else if(val>14.5&&val<24.5) distance\_key=32;

else if(val>24.5&&val<30.5) distance\_key=33;

else if(val>30.5&&val<44.5) distance\_key=34;

}

else distance\_key=-1;

if(distance\_key==key\_save[1])

distance\_key=-1;

else key\_save[1]=distance\_key;

}

else if(i==2)//第三个传感器

{

if(val>=4.5&&val<=35.5)

{

if(val>4.5&&val<11.5) distance\_key=35;

else if(val>11.5&&val<16.5) distance\_key=36;

else if(val>16.5&&val<20.5) distance\_key=37;

else if(val>20.5&&val<27.5) distance\_key=41;

else if(val>27.5&&val<35.5) distance\_key=42;

}

else distance\_key=-1;

if(distance\_key==key\_save[2])

distance\_key=-1;

else key\_save[2]=distance\_key;

}

else if(i==3)//第四个传感器

{

if(val>=4.5&&val<=33.5)

{

if(val>4.5&&val<11.5) distance\_key=43;

else if(val>11.5&&val<18.5) distance\_key=44;

else if(val>18.5&&val<22.5) distance\_key=45;

else if(val>22.5&&val<27.5) distance\_key=46;

else if(val>27.5&&val<33.5) distance\_key=47;

}

else distance\_key=-1;

if(distance\_key==key\_save[3])

distance\_key=-1;

else key\_save[3]=distance\_key;

}

else distance\_key=-1;

return distance\_key;//回传琴键音

}

### 6.2.4 数据处理模块

#### 1.自由弹奏模式

##### 2）相关代码

while(FlagFree){

// if(Serial.available())

m=Serial.parseInt();//读取是否重选模式

if(m==9)

{

FlagFree=0;

FlagHalffree=0;

FlagPlayer=0;

Serial.println("1.Free playing mode");

Serial.println("2.Semi-free playing mode");

Serial.println("3.player");

}

else

{

if(FlagCreative==0)

{

for(int i=0;i<4;i++)

{

val=ave\_gp2d12\_range(i+1);//调用测距函数

define\_key(define\_distance(i,val));//调用播放函数

}

}

#### 2.创作模式

##### 2）相关代码

else {

int N;

int create\_opern[999];

int create\_length[999];

int c\_finish\_flag=1;

for(int i=0;i<999;i++)

{

gettime();//调用获取间隔函数

if(distance\_key1!=-1) //如果琴键音有效

{

// N=(playtime+31)/62;//将测得时间归一化，并四舍五入

create\_opern[i]=distance\_key1;//储存琴键音

//create\_length[i]=N;//储存时长

Serial.print(N);

if(distance\_key1==100)//如果检测到结束音

{

//create\_length[i]=0;

c\_finish\_flag=1;//回放flag置1

i=999;//跳出循环

}

else c\_finish\_flag=0;

// distance\_key1=distance\_key2;

}

else

{

i=i-1;//若琴键音无效则重新储存

}

}

if(c\_finish\_flag==1)

{

Serial.println("Record finished! Do you want to play it?" );//选择是否回放

Serial.println("1.Yes.");

Serial.println("2.No.");

delay(5000);

m=Serial.parseInt();//读取

if(m==9)//检测是否有复位信号

{

FlagFree=0;

FlagHalffree=0;

FlagPlayer=0;

Serial.println("1.Free playing mode");

Serial.println("2.Semi-free playing mode");

Serial.println("3.player");

}

else

{

if(m==1)//回放

{

c\_finish\_flag=0;

for(int i=0;i<999;i++)

{

if(create\_opern[i]!=100)

{

define\_key(create\_opern[i]);//依次读取播放

// for(int j=0;j<create\_length[i];j++)//延时所记录的时间

// delay(62);

}

else

{

define\_key(100);//若检测到结束信号，回放结束

i=999;

}

delay(1000);

}

}

else if(m==2)//不回放

{

c\_finish\_flag=0;

for(int i=0;i<999;i++)

{

if(create\_opern[i]!=0)

{

create\_opern[i]=0;//清空储存

//create\_length[i]=0;

}

else i=999;

}

Serial.println("Record has been emptied!");

}

else

{

Serial.println("Input error!Please re-select.");

}

}

}

}

}

}

//int cnt=1;//状态记录变量

/\*void gettime()//获取时间函数

{

for(int i=0;i<4;i++)

{

val=ave\_gp2d12\_range(i+1);//读取距离

if(cnt=1)//状态1

{

distance\_key1=define\_distance(i,val);//读取琴键音1

if(distance\_key1!=-1)//若有效

{

time1=millis();//获取时间1

define\_key(distance\_key1);//播放

cnt=2;

}

else//无效

{

if(i==3)i=-1;//持续循环直至测到有效输入

else i=i;

define\_key(-1);//无效音

}

}

else if(cnt=2)状态2

{

distance\_key2=define\_distance(i,val);//读取琴键音2

if(distance\_key2!=-1)//若有效

{

time2=millis();//获取时间2

define\_key(distance\_key2);//播放

i=4;//跳出循环

}

else

{

if(i==3)i=-1;//若无效，则持续循环直至输入有效

else i=i;

define\_key(-1);//无效音

}

}

}

playtime=time2-time1;//计算时间差

time1=time2;//将时间2作为下一次调用函数的时间1

delay(250);

}

\*/

void gettime()//获取时间函数

{

for(int i=0;i<4;i++)

{

val=ave\_gp2d12\_range(i+1);//测距

distance\_key1=define\_distance(i,val);//匹配琴键音

if(distance\_key!=-1)//若有效

{

time1=millis();//获取时间1

define\_key(distance\_key);//播放

time2=millis();//获取时间2

i=4;

}

else//若无效，持续循环直至输入有效

{

if(i==3)i=0;

else i=i;

define\_key(-1);//无效音

}

}

}

#### 3.半自由弹奏模式

本部分主要包括半自由弹奏模式的功能介绍及相关代码。

##### 1）功能介绍

该模式提前输入了两支曲谱。选择其中一个，将活动挡板遮挡在22个有效区域内，即可自动播放选中曲子的第一个音。当用户认为该琴键音持续时长已足够，即可进行下一次遮挡，自动播放选中曲子的下一个音……如此循环，直到结束音响起，弹奏结束。用户只需控制每个音的持续时长，而不必在意音准，即可“弹奏”完成整支曲子如图6-11所示。

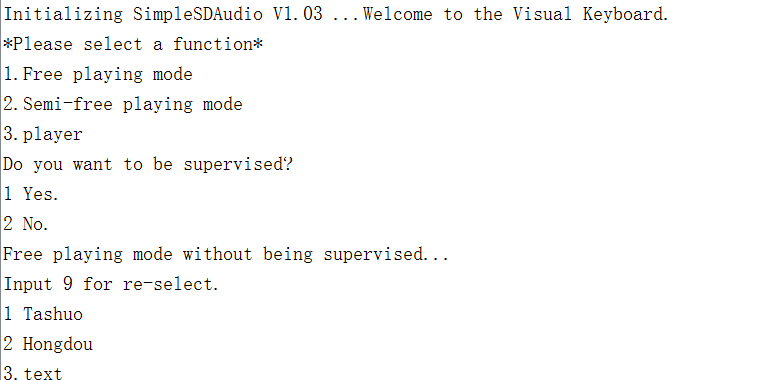


图6-10 半自由弹奏模式串口监视器图

##### 2）相关代码

while(FlagHalffree)//半自由弹奏模式

{

int flag\_halffree=1;

Serial.println("1 Tashuo");//选择曲目

Serial.println("2 Hongdou");

Serial.println("3.text");

delay(5000);

m=Serial.parseInt();

if(m==9)//检测复位信号

{

FlagFree=0;

FlagHalffree=0;

FlagPlayer=0;

Serial.println("1.Free playing mode");

Serial.println("2.Semi-free playing mode");

Serial.println("3.player");

}

else

{

while(flag\_halffree)

{

if(m==1)

{

int Opern[999]={ //《她说》曲谱

41,41,37,37,35,31,36,

41,41,37,37,35,31,35,

41,41,37,37,41,42,42,

41,41,37,37,41,42,42,41,43,

41,41,42,41,42,41,46,

41,41,42,41,42,41,45,

43,42,41,41,41,41,41,42,42,

41,41,37,41,42,41,44,43,44,43,44,43,44,43,44,45,

43,44,45,41,46,41,47,46,47,41,46,45,

42,43,44,43,45,37,36,35,36,37,45,43,

42,43,44,43,44,43,44,42,43,44,43,44,43,44,45,43,

43,44,45,41,46,41,37,36,37,41,47,45,

42,43,44,43,45,45,43,45,43,45,46,43,43,43,42,42,43,44,44,

41,41,37,37,41,42,42,41,41,

100

};

if(FlagSupervision)//监督模式

Supervision(Opern);

else Halffree\_Playing(Opern);//非监督模式

}

else if(m==2)

{

flag\_halffree=0;

int Opern[999]=//《红豆》曲谱

{

25,26,32,31,32,31,32,

25,26,32,31,32,33,32,

25,26,32,31,31,26,32,33,32,31,32,31,31,26,25,

25,26,32,31,32,31,32,

25,26,32,31,32,33,32,

25,26,32,31,31,26,32,33,32,31,36,35,35,33,32,

31,32,35,31,32,33,33,32,32,31,33,41,37,36,33,

37,36,35,36,36,35,34,35,

33,32,31,32,31,26,33,32,32,

31,32,35,31,32,33,

33,32,32,31,33,41,37,41,36,

36,43,42,37,35,33,36,

33,32,31,32,36,36,35,32,33,32,31,

100

};

if(FlagSupervision)//监督模式

Supervision(Opern);

else Halffree\_Playing(Opern);//非监督模式

}

else if(m==3)

{

int Opern[999]={//test模式

21,22,23,24,25,26 };

if(FlagSupervision)//监督模式

Supervision(Opern);

else Halffree\_Playing(Opern);//非监督模式

}

else

{

Serial.println("Error! Please re-select.");//报错

}

}

}

}

void define\_halffree\_playing(int i,int key)//半自由弹奏匹配琴键音，功能基本同上

{

val=ave\_gp2d12\_range(i+1);

distance\_key=define\_distance(i,val);

if(distance\_key!=-1)

{

define\_key(key);

flag\_halffree\_finish=1;

}

else

{

define\_key(-1);

flag\_halffree\_finish=0;

}

}

void Halffree\_Playing(int Opern[])//半自由模式（不监督）

{

for (int j=0;j<999;j++)//循环读取曲谱

{

if(Opern[j]==0)//若读取到空值

j=999;//跳出循环

while(Opern[j]<21||Opern[j]>47)//读取到错误值

j++;//继续读取下一个

for(int i=0;i<4;i++)//每个传感器依次读取

{

define\_halffree\_playing(i,Opern[j]);//调用半自由模式下琴键匹配函数

if(flag\_halffree\_finish)//若琴键音有效，跳出循环

i=4;

else //否则依次读取直至有效

{

if(i==3) i=-1;

else i=i;

}

}

if(Opern[j]==100)//读取到结束音，跳出循环

j=999;

}

}

#### 4.监督模式

本部分主要包括功能介绍及相关代码。

##### 1）功能介绍

此模式为半自由弹奏模式的一个分支，在此模式下，选中一支指定曲目。读取第一个储存音符的数组，当用户遮挡不在该琴键音对应的正确范围时，会播放“报错音”（即结束音），只有当遮挡在正确范围内，才会播放对应的琴键音，并读取数组中下一个数据进行监督……如此循环，当播放结束音的时候，监督结束，如图6-11所示。

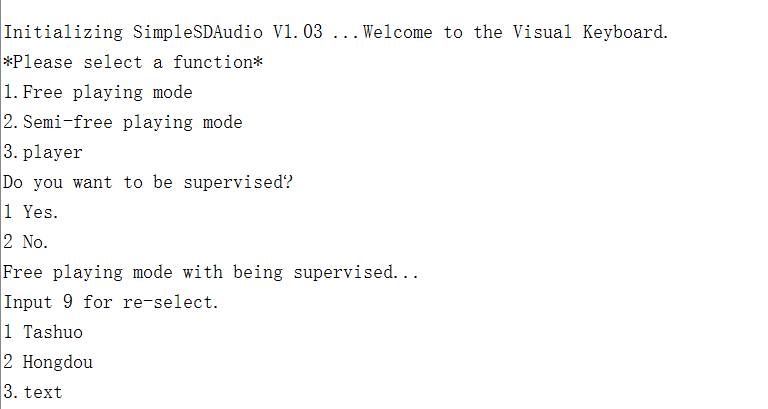


图6-11监督模式串口监视器截图

##### 2）相关代码

void Supervision(int Opern[])//监督模式函数

{

int supervision\_key;

for(int j=0;j<999;j++)

{

if(Opern[j]==0)//若读取结束，播放两次结束音提示

{

j=999;

FlagSupervision=0;

define\_key(100);

define\_key(100);

}

while(Opern[j]<21||Opern[j]>47)

j++;

for(int i=0;i<4;i++)

{

val=ave\_gp2d12\_range(i+1);

supervision\_key=define\_distance(i,val);

if(supervision\_key>20&&supervision\_key<48) //琴键音有效

{

if(supervision\_key==Opern[j])//若与曲谱内琴键音相符合，则播放对应琴键音

{

define\_key(Opern[j]);

i=4;//跳出循环

}

else

{

define\_key(100);//否则播放报错音

if(i==3)i=-1;//接着循环

else i=i;

}

}

else//琴键音无效则持续循环

{

i=i;

if(i==3)

i=-1;

define\_key(-1);

}

delay(250);

}

}

}

#### 5.音乐播放模式

本部分主要包括功能介绍和相关代码。

##### 1）功能介绍

通过读取SD卡中已录入的音乐文件，实现音乐播放器的功能（开始、暂停、重选等），如图6-12所示。

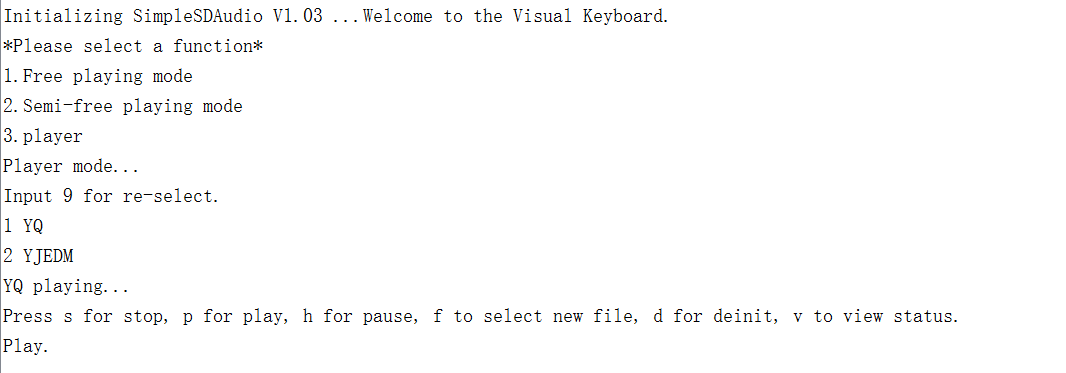


图6-12 播放模式串口监视器图

##### 2）相关代码

while(FlagPlayer)//播放器模式

{

char c\_player;

int flag\_player=1;

int c\_flag=1;

Serial.println("1 YQ");

Serial.println("2 YJEDM");

delay(5000);

m=Serial.parseInt();

if(m==9)//检测复位信号

{

FlagFree=0;

FlagHalffree=0;

FlagPlayer=0;

Serial.println("1.Free playing mode");

Serial.println("2.Semi-free playing mode");

Serial.println("3.player");

}

else

{

while(flag\_player)

{

if(m==1)//播放《氧气》

{

flag\_player=0;

SdPlay.setFile("YQ.AFS");

Serial.println("YQ playing...");

Serial.println(F("Press s for stop, p for play, h for pause, f to select new file, d for deinit, v to view status."));

}

else if(m==2)//播放《遇见二丁目》

{

flag\_player=0;

SdPlay.setFile("YJEDM.AFS");

Serial.println("YJEDM playing...");

Serial.println(F("Press s for stop, p for play, h for pause, f to select new file, d for deinit, v to view status."));

}

else

{

c\_flag=0;

Serial.println("Error! please re-select");//报错

Serial.println("1.YQ");

Serial.println("2.YJEDM");

}

}

while(c\_flag) {

SdPlay.worker(); // You can remove this line if you like - worker is not necessary

if(Serial.available()) {

c\_player = Serial.read();

switch(c\_player) {

case 's'://停止

SdPlay.stop();

Serial.println(F("Stopped."));

break;

case 'p'://播放

SdPlay.play();

Serial.println(F("Play."));

break;

case 'h'://暂停

SdPlay.pause();

Serial.println(F("Pause."));

break;

case 'd'://移除

SdPlay.deInit();

Serial.println(F("SdPlay deinitialized. You can now safely remove card. System halted."));

while(1) ;

break;

case 'f'://重选

c\_flag = 0;

break;

case 'v'://显示状态

Serial.print(F("Status: isStopped="));

Serial.print(SdPlay.isStopped());

Serial.print(F(", isPlaying="));

Serial.print(SdPlay.isPlaying());

Serial.print(F(", isPaused="));

Serial.print(SdPlay.isPaused());

Serial.print(F(", isUnderrunOccured="));

Serial.print(SdPlay.isUnderrunOccured());

Serial.print(F(", getLastError="));

Serial.println(SdPlay.getLastError());

Serial.print(F("Free RAM: "));

Serial.println(freeRam());

break;

}

}

}

}

}

# 第7章 智能弹奏尤克里里项目设计

### 7.2.1 主程序模块

#### 2.相关代码

#include <Servo.h>

Servo b1; //定义拨弦4个舵机对象

Servo b2;

Servo b3;

Servo b4;

Servo a2; //定义按弦3个舵机对象

Servo a4;

Servo a7;

int apos[4][2] ={

{0,0},{25,0},{50,23},{25,5}

}; //按弦舵机的角度位置

int bpos[5][2] = {

{0,0},{85,60},{58,33},{90 ,62},{83,52}

}; //拨弦舵机的角度位置

String s=" "; //初始化字符串常量

void setup()

{

Serial.begin(9600); //波特率为9600与串口传输数据

a2.attach(8); //3个按弦舵机引脚

a4.attach(9);

a7.attach(10);

b1.attach(2); //4个拨弦舵机引脚

b2.attach(3);

b3.attach(4);

b4.attach(5);

a2.write(apos[1][0]); //3个按弦舵机的初始位置

a4.write(apos[2][0]);

a7.write(apos[3][0]);

b1.write(bpos[1][0]); //4个拨弦舵机的初始位置

b2.write(bpos[2][0]);

b3.write(bpos[3][0]);

b4.write(bpos[4][0]);

}

int song1[] = //音符数组对应的歌曲1

{

1234,0,0,1234,0,0,

1234,0,0,1234,0,0,

1,2,3,1,0,

1,2,3,1,0,

3,4,5,0,

3,4,5,0,

5,6,5,4,3,1,0,

5,6,5,4,3,1,0,

3,5,1,0,

3,5,1,0

};

int song2[] = //音符数组对应的歌曲2

{

1234,0,0,1234,0,0,

1234,0,0,1234,0,0,

3,3,4,5,0,

5,4,3,2,0,

1,1,2,3,3,2,2,0,

3,3,4,5,0,

5,4,3,2,0,

1,1,2,3,2,1,1,0,

2,2,3,1,0,

2,3,4,3,1,0,

2,3,4,3,2,0,

1,2,5,0,

3,3,4,5,0,

5,4,3,4,2,0,

1,1,2,3,2,1,1,0

};

int song3[] = //音符数组对应的歌曲3

{

1234,0,0,1234,0,0,

1234,0,0,1234,0,0,

1,1,5,5,6,6,5,0,

4,4,3,3,2,2,1,0,

5,5,4,4,3,3,2,0,

5,5,4,4,3,3,2,0,

1,1,5,5,6,6,5,0,

4,4,3,3,2,2,1,0

};

int song4[] = //音符数组对应的歌曲4

{

1234,0,0,1234,0,0,

1234,0,0,1234,0,0,

3,3,3,4,5,3,2,

1,1,1,2,3,3,7,7,

6,3,2,

6,3,2,

6,3,2,1,1,

3,2,5,4,3,2,

5,4,3,2,5,3,2,

6,3,2,6,3,2,

4,3,4,3,1,

4,3,4,3,1,2,1,1

};

int song5[] = //音符数组对应的歌曲5

{

1234,0,0,1234,0,0,

1234,0,0,1234,0,0,

3,0,5,0,1,0,0,

2,0,3,0,5,0,0,

1,2,3,5,2,0,0,

3,5,1,0,0,

2,3,6,0,0,

2,5,2,3,1,0,

6,0,5,6,0,1,0,

3,6,3,5,2,0,

3,5,1,0,

2,3,6,0,

2,5,2,3,1,0

};

int songSize1=sizeof(song1)/sizeof(song1[0]); //计算歌曲1的音符个数

int songSize2=sizeof(song2)/sizeof(song2[0]); //计算歌曲2的音符个数

int songSize3=sizeof(song3)/sizeof(song3[0]); //计算歌曲3的音符个数

int songSize4=sizeof(song4)/sizeof(song4[0]); //计算歌曲4的音符个数

int songSize5=sizeof(song5)/sizeof(song5[0]); //计算歌曲5的音符个数

int b1State = 0, b2State = 0, //初始化舵机位置

b3State = 0, b4State = 0,

a2State = 0, a4State = 0,

a7State = 0;

int servoPlay1(int beat = 250) //歌曲1的弹奏

{

for(int i = 0;i<songSize1;i++) //循环弹奏直到歌曲音符弹奏完

{

Serial.println(song1[i]); //在串口监视器中打印音符

switch (song1[i]) //选择音符所需弦

{

case 1: //“1”音所需的弦

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

a2.write(apos[1][0]);

break;

case 2: //“2”音所需的弦

a2.write(apos[1][1]);

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

break;

case 3: //“3”音所需的弦

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

a4.write(apos[2][1]);

break;

case 4: //“4”音所需的弦

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

a4.write(apos[2][0]);

break;

case 5: //“5”音所需的弦

b1State = 1- b1State;

b1.write(bpos[1][b1State]);

break;

case 6: //“6”音所需的弦

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

a7.write(apos[3][0]);

break;

case 7: //“7”音所需的弦

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

a7.write(apos[3][1]);

case 0: //无音符节拍既不拨弦也不按弦

break;

case 1234:

a2.write(apos[1][0]);

a4.write(apos[2][1]);

a7.write(apos[3][0]);

b1State = 1- b1State;

b1.write(bpos[1][b1State]);

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

default:

break;

}

delay(beat); //弹奏完延迟250ms

}

}

int servoPlay2(int beat = 250) //歌曲2的弹奏

{

for(int i = 0;i<songSize2;i++) //循环弹奏直到歌曲音符弹奏结束

{

Serial.println(song2[i]); //在串口监视器中打印音符

switch (song2[i]) //选择音符所需的弦

{

case 1: //“1”音所需的弦

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

a2.write(apos[1][0]);

break;

case 2: //“2”音所需的弦

a2.write(apos[1][1]);

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

break;

case 3: //“3”音所需的弦

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

a4.write(apos[2][1]);

break;

case 4: //“4”音所需的弦

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

a4.write(apos[2][0]);

break;

case 5: //“5”音所需的弦

b1State = 1- b1State;

b1.write(bpos[1][b1State]);

break;

case 6: //“6”音所需的弦

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

a7.write(apos[3][0]);

break;

case 7: //“7”音所需的弦

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

a7.write(apos[3][1]);

case 0: //无音符节拍既不拨弦也不按弦

break;

case 1234:

a2.write(apos[1][0]);

a4.write(apos[2][0]);

a7.write(apos[3][0]);

b1State = 1- b1State;

b1.write(bpos[1][b1State]);

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

default:

break;

}

delay(beat); //弹奏完延迟250ms

}

}

int servoPlay3(int beat = 250) //歌曲3的弹奏

{

for(int i = 0;i<songSize3;i++) //循环弹奏直到歌曲音符弹奏结束

{

Serial.println(song3[i]); //在串口监视器中打印音符

switch (song3[i]) //选择音符所需的弦

{

case 1: //“1”音所需的弦

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

a2.write(apos[1][0]);

break;

case 2: //“2”音所需的弦

a2.write(apos[1][1]);

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

break;

case 3: //“3”音所需的弦

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

a4.write(apos[2][1]);

break;

case 4: //“4”音所需的弦

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

a4.write(apos[2][0]);

break;

case 5: //“5”音所需的弦

b1State = 1- b1State;

b1.write(bpos[1][b1State]);

break;

case 6: //“6”音所需的弦

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

a7.write(apos[3][0]);

break;

case 7: //“7”音所需的弦

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

a7.write(apos[3][1]);

case 0: //无音符节拍既不拨弦也不按弦

break;

case 1234:

a2.write(apos[1][0]);

a4.write(apos[2][1]);

a7.write(apos[3][0]);

b1State = 1- b1State;

b1.write(bpos[1][b1State]);

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

default:

break;

}

delay(beat); //弹奏完延迟250ms

}

}

int servoPlay4(int beat = 250) //歌曲4的弹奏

{

for(int i = 0;i<songSize4;i++) //循环弹奏直到歌曲音符弹奏完

{

Serial.println(song4[i]); //在串口监视器中打印音符

switch (song4[i]) //选择音符所需的弦

{

case 1: //“1”音所需的弦

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

a2.write(apos[1][0]);

break;

case 2: //“2”音所需的弦

a2.write(apos[1][1]);

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

break;

case 3: //“3”音所需的弦

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

a4.write(apos[2][1]);

break;

case 4: //“4”音所需的弦

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

a4.write(apos[2][0]);

break;

case 5: //“5”音所需的弦

b1State = 1- b1State;

b1.write(bpos[1][b1State]);

break;

case 6: //“6”音所需的弦

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

a7.write(apos[3][0]);

break;

case 7: //“7”音所需拨的弦

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

a7.write(apos[3][1]);

case 0: //无音符节拍既不拨弦也不按弦

break;

case 1234:

a2.write(apos[1][0]);

a4.write(apos[2][0]);

a7.write(apos[3][0]);

b1State = 1- b1State;

b1.write(bpos[1][b1State]);

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

default:

break;

}

delay(beat); //弹奏完延迟250ms

}

}

int servoPlay5(int beat = 250) //歌曲5的弹奏

{

for(int i = 0;i<songSize5;i++) //循环弹奏直到歌曲音符弹奏结束

{

Serial.println(song5[i]); //在串口监视器中打印音符

switch (song5[i]) //选择音符所需的弦

{

case 1: //“1”音所需的弦

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

a2.write(apos[1][0]);

break;

case 2: //“2”音所需的弦

a2.write(apos[1][1]);

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

break;

case 3: //“3”音所需的弦

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

a4.write(apos[2][1]);

break;

case 4: //“4”音所需的弦

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

a4.write(apos[2][0]);

break;

case 5: //“5”音所需的弦

b1State = 1- b1State;

b1.write(bpos[1][b1State]);

break;

case 6: //“6”音所需的弦

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

a7.write(apos[3][0]);

break;

case 7: //“7”音所需的弦

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

a7.write(apos[3][1]);

case 0: //无音符节拍既不拨弦也不按弦

break;

case 1234:

a2.write(apos[1][0]);

a4.write(apos[2][0]);

a7.write(apos[3][0]);

b1State = 1- b1State;

b1.write(bpos[1][b1State]);

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

default:

break;

}

delay(beat); //弹奏完延迟250ms

}

}

void loop()

{

if(Serial.available())

{

s=Serial.readString();

Serial.print(s);

if(s=="a.")

{

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

a2.write(apos[1][0]);

}

if(s=="b.")

{

b2State = 1- b2State;

b2.write(bpos[2][b2State]);

a2.write(apos[1][1]);

}

if(s=="c.")

{

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

a4.write(apos[2][1]);

}

if(s=="d.")

{

b3State = 1- b3State;

b3.write(bpos[3][b3State]);

a4.write(apos[2][0]);

}

if(s=="e.")

{

b1State = 1- b1State;

b1.write(bpos[1][b1State]);

}

if(s=="f.")

{

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

a7.write(apos[3][0]);

}

if(s=="g.")

{

b4State = 1- b4State;

b4.write(bpos[4][b4State]);

a7.write(apos[3][1]);

}

if(s=="A.")

{

servoPlay1(350);

delay(300);

}

if(s=="B.")

{

servoPlay2(350);

delay(300);

}

if(s=="C.")

{

servoPlay3(350);

delay(300);

}

if(s=="D.")

{

servoPlay4(350);

delay(300);

}

if(s=="E.")

{

servoPlay5(350);

delay(300);

}

}

}

### 7.2.2 HC-05蓝牙模块

#### 2.相关代码

void setup() {

Serial.begin(38400);

}

void sendcmd()

{

Serial.println("AT"); //表示开始进入AT模式

while(Serial.available())

{

char ch;

ch = Serial.read();

Serial.print(ch);

}

delay(1000);

//不需要改动的参数可在后期注释掉

Serial.println("AT+NAME?");//设置名称，把name改成名字即可

while(Serial.available())

{

char ch;

ch = Serial.read(); //从串口读取名字

Serial.print(ch); //打印名字

}

delay(1000);

Serial.println("AT+CMODE=1");//蓝牙连接模式为任意地址连接模式，也就是蓝牙可以被任意设备连接

while(Serial.available())

{

char ch;

ch = Serial.read();  //从串口读取蓝牙连接模式

Serial.print(ch); //打印蓝牙连接模式

}

delay(1000);

Serial.println("AT+ADDR==??");//修改蓝牙地址，把？？改正地址即可

while(Serial.available())

{

char ch;

ch = Serial.read(); //从串口读取蓝牙地址

Serial.print(ch); //打印蓝牙地址

}

delay(1000);

Serial.println("AT+PSWD=1234");//设置密码

while(Serial.available())

{

char ch;

ch = Serial.read(); //从串口读取密码

Serial.print(ch); //打印密码

}

delay(1000);

Serial.println("AT+ROLE=0");//蓝牙模式为从模式

while(Serial.available())

{

char ch;

ch = Serial.read(); //从串口读取蓝牙模式

Serial.print(ch); //打印蓝牙模式

}

delay(1000);

Serial.println("AT+UART=9600,0,0");//蓝牙通信串口波特率为9600，停止位为1，无校验位

while(Serial.available())

{

char ch;

ch = Serial.read(); //从串口读取蓝牙通信串口波特率

Serial.print(ch); //打印蓝牙通信串口波特率

}

delay(1000);

}

void loop() {

sendcmd();

}

### 7.2.3 手机端APP制作

### 7.2.4 舵机的调试

#### 2.相关代码

#include <Servo.h>

Servo s1; //定义2个舵机对象

Servo s2;

int pos[3][2] = {{0,0},{90,70},{80,60}};//舵机的旋转角度

void setup() {

Serial.begin(9600);

s1.attach(8); //舵机引脚

s2.attach(9);

s1.write(pos[1][0]); //舵机的位置

s2.write(pos[2][0]);

}

int song[] = //音符数组

{

2,2,2,2

};

int songSize=sizeof(song)/sizeof(song[0]); //计算音符个数

int s1State = 0, s2State = 0; //初始化舵机位置

int servoPlay(int beat = 250) //对应音符舵机的位置控制

{

for(int i = 0;i<songSize;i++)

{

Serial.println(song[i]); //在串口监视器中打印音符

switch (song[i]) //各音符所需拨的弦

{

case 1:

s2State = 1- s2State;

s2.write(pos[2][s2State]);

s1.write(pos[1][0]);

break;

case 2:

s2State = 1- s2State;

s2.write(pos[2][s2State]);

s1.write(pos[1][1]);

break;

default:

break;

}

delay(beat);

}

}

void loop()

{

servoPlay(350);

delay(300);

}

# 第8章 身临其境项目设计

### 8.2.1 主程序模块

#### 2.相关代码

//运行代码

#include <SoftwareSerial.h>

#include <DFPlayer\_Mini\_Mp3.h>

//实例化软串口

SoftwareSerial mySoftwareSerial(2, 3); // RX, TX

uint16\_t volume=3; //初始音量

void setup()

{

Serial.begin(9600);

mySoftwareSerial.begin(9600);

mp3\_set\_serial (mySoftwareSerial);

mp3\_set\_volume (volume);

pinMode(13, OUTPUT);

pinMode(12,OUTPUT);

pinMode(11,OUTPUT);

Serial.println("start read loop");

while (Serial.available())

Serial.read();

while (mySoftwareSerial.available())

mySoftwareSerial.read();

}

char cmd;

void loop()

{

if (Serial.available())

{

cmd=Serial.read();

if (cmd=='1')

mp3\_play (); //播放

if (cmd=='2')

mp3\_pause (); //暂停

if (cmd=='3')

{

volume=volume+3;

mp3\_set\_volume(volume);

}//音量调大

if (cmd=='4')

{

volume=volume-3;

mp3\_set\_volume(volume);

} //音量调小

if(cmd=='8')

mp3\_prev (); //上一曲

if(cmd=='9')

mp3\_next (); //下一曲

if (cmd=='5')

void mp3\_random\_play (); // 随机播放

if (cmd=='6'){

mp3\_stop (); //停止播放

digitalWrite(13,LOW);

digitalWrite(12,LOW);

digitalWrite(11,LOW);

}

if (cmd=='a'){

mp3\_play (1); //循环播放指定曲目

mp3\_single\_loop(1);

Serial.println("(1,2)");

digitalWrite(13,HIGH); //黄灯亮

}

if (cmd=='b'){

mp3\_play (2); //循环播放指定曲目

mp3\_single\_loop(1);

Serial.println("(1,3)");

digitalWrite(12,HIGH); //红灯亮

}

if (cmd=='c'){

mp3\_play (3); //循环播放指定曲目

mp3\_single\_loop(1); //循环播放指定曲目

Serial.println("(1,1)");

digitalWrite(11,HIGH); //绿灯亮

}

if(cmd=='d'){

mp3\_play (4); //循环播放指定曲目

mp3\_single\_loop(1); //循环播放指定曲目

Serial.println("(2,1)");

for (int value = 0 ; value < 255; value=value+1){

analogWrite(13, value);

delay(5);

}

for (int value = 255; value >0; value=value-1){

analogWrite(13, value);

delay(5);

}

} //呼吸灯

if(cmd=='e'){

mp3\_play(5);

mp3\_single\_loop (1); //循环播放指定曲目

digitalWrite(11,HIGH);

delay(100);

digitalWrite(12,HIGH);

delay(100);

digitalWrite(13,HIGH);

delay(100);

digitalWrite(13, LOW);

delay(100);

digitalWrite(12, LOW);

delay(100);

digitalWrite(11, LOW);

delay(100);

} //流水灯

delay(500);

if(cmd!='e'&cmd!='d')

cmd='\0';

//}

}

### 8.2.2 HC-05模块

#### 2.相关代码

#include <SoftwareSerial.h>

SoftwareSerial mySerial(2, 3); // 定义HC-05的引脚RX，TX

void setup()

Serial.begin(38400); //初始化串口通信，将波特率设置为38400

while (!Serial); //等待串口通信

Serial.println("O");

mySerial.begin(4800); //初始化蓝牙通信，将波特率设置为9600

mySerial.println("K");

}

void loop(){

if (mySerial.available()){ //将HC-05收到的信号发送到Arduino开发板

Serial.write(mySerial.read());

}

if (Serial.available()) { //将Arduino开发板的信号发送到HC-05

mySerial.write(Serial.read());

}

}

/\*1.设置模块的工作模式为任意设备连接模式AT+COMDE=1

2.蓝牙名称AT+NAME=’hc05’

3.配对密码1234 \*/

### 8.2.3 输出模块

#### 2.相关代码

//DFPlayer\_Mini\_Mp3.h

#include "Arduino.h"

#include "SoftwareSerial.h"

uint8\_t send\_buf[10] = {

0x7E, 0xFF, 06, 00, 00, 00, 00, 00, 00, 0xEF};

uint8\_t recv\_buf[10];

//\* void(\*send\_func)() = NULL;

//\* HardwareSerial \*hserial = NULL;

//\* SoftwareSerial \*sserial = NULL;

//\* boolean is\_reply = false;

// 7E FF 06 0F 00 01 01 xx xx EF

// 0 -> 7E is start code

// 1 -> FF is version

// 2 -> 06 is length

// 3 -> 0F is command

// 4 -> 00 is no receive

// 5~6 -> 01 01 is argument

// 7~8 -> checksum = 0 - ( FF+06+0F+00+01+01 )

// 9 -> EF is end code

void mp3\_set\_reply (boolean state);

void mp3\_fill\_cmd (uint8\_t cmd, uint16\_t arg);

void mp3\_fill\_cmd (uint8\_t cmd);

//void fill\_uint16\_bigend (uint8\_t \*thebuf, uint16\_t data);

//error because it is biggend mode in mp3 module

//void fill\_uint16 (uint8\_t \*thebuf, uint16\_t data) {

// \*(uint16\_t\*)(thebuf) = data;

//}

void mp3\_set\_serial (HardwareSerial &theSerial);

void mp3\_set\_serial (SoftwareSerial &theSerial);

//void h\_send\_func ();

//void s\_send\_func ();

//void mp3\_send\_cmd ();

uint16\_t mp3\_get\_checksum (uint8\_t \*thebuf);

void mp3\_fill\_checksum ();

void mp3\_play\_physical (uint16\_t num);

void mp3\_play\_physical ();

void mp3\_next (); //上一曲

void mp3\_prev (); //下一曲

//0x06 set volume 0-30

void mp3\_set\_volume (uint16\_t volume);

//0x07 set EQ0/1/2/3/4/5 Normal/Pop/Rock/Jazz/Classic/Bass

void mp3\_set\_EQ (uint16\_t eq);

//0x09 set device 1/2/3/4/5 U/SD/AUX/SLEEP/FLASH

void mp3\_set\_device (uint16\_t device);

void mp3\_sleep ();

void mp3\_reset ();

void mp3\_pause (); //暂停

void mp3\_stop (); //停止播放

void mp3\_play (); //播放

//specify a mp3 file in mp3 folder in your tf card, "mp3\_play (1);" mean play "mp3/0001.mp3"

void mp3\_play (uint16\_t num);

void mp3\_get\_state ();

void mp3\_get\_volume (); //音量

void mp3\_get\_u\_sum ();

void mp3\_get\_tf\_sum ();

void mp3\_get\_flash\_sum ();

void mp3\_get\_tf\_current ();

void mp3\_get\_u\_current ();

void mp3\_get\_flash\_current ();

//set single loop

void mp3\_single\_loop (boolean state); //循环播放

void mp3\_single\_play (uint16\_t num);

void mp3\_DAC (boolean state);

void mp3\_random\_play (); //随机播放

//DFPlayer\_Mini\_Mp3.cpp

#include <Arduino.h>

#include <SoftwareSerial.h>

//#include "DFPlayer\_Mini\_Mp3.h"

extern uint8\_t send\_buf[10];

extern uint8\_t recv\_buf[10];

static void(\*send\_func)() = NULL;

static HardwareSerial \* \_hardware\_serial = NULL;

static SoftwareSerial \* \_software\_serial = NULL;

static boolean is\_reply = false;

void mp3\_set\_reply (boolean state) {

is\_reply = state;

send\_buf[4] = is\_reply;

}

static void fill\_uint16\_bigend (uint8\_t \*thebuf, uint16\_t data) {

\*thebuf = (uint8\_t)(data>>8);

\*(thebuf+1) = (uint8\_t)data;

}

//calc checksum (1~6 byte)

uint16\_t mp3\_get\_checksum (uint8\_t \*thebuf) {

uint16\_t sum = 0;

for (int i=1; i<7; i++) {

sum += thebuf[i];

}

return -sum;

}

//fill checksum to send\_buf (7~8 byte)

void mp3\_fill\_checksum () {

uint16\_t checksum = mp3\_get\_checksum (send\_buf);

fill\_uint16\_bigend (send\_buf+7, checksum);

}

void h\_send\_func () {

for (int i=0; i<10; i++) {

\_hardware\_serial->write (send\_buf[i]);

}

}

void s\_send\_func () {

for (int i=0; i<10; i++) {

\_software\_serial->write (send\_buf[i]);

}

}

//void mp3\_set\_serial (HardwareSerial \*theSerial) {

void mp3\_set\_serial (HardwareSerial &theSerial) {

\_hardware\_serial = &theSerial;

send\_func = h\_send\_func;

}

void mp3\_set\_serial (SoftwareSerial &theSerial) {

\_software\_serial = &theSerial;

send\_func = s\_send\_func;

}

void mp3\_send\_cmd (uint8\_t cmd, uint16\_t arg) {

send\_buf[3] = cmd;

fill\_uint16\_bigend ((send\_buf+5), arg);

mp3\_fill\_checksum ();

send\_func ();

}

void mp3\_send\_cmd (uint8\_t cmd) {

send\_buf[3] = cmd;

fill\_uint16\_bigend ((send\_buf+5), 0);

mp3\_fill\_checksum ();

send\_func ();

}

void mp3\_play\_physical (uint16\_t num) {

mp3\_send\_cmd (0x03, num);

}

void mp3\_play\_physical () {

mp3\_send\_cmd (0x03);

}

void mp3\_next () {

mp3\_send\_cmd (0x01);

}

void mp3\_prev () {

mp3\_send\_cmd (0x02);

}

//0x06 set volume 0-30

void mp3\_set\_volume (uint16\_t volume) {

mp3\_send\_cmd (0x06, volume);

}

//0x07 set EQ0/1/2/3/4/5 Normal/Pop/Rock/Jazz/Classic/Bass

void mp3\_set\_EQ (uint16\_t eq) {

mp3\_send\_cmd (0x07, eq);

}

//0x09 set device 1/2/3/4/5 U/SD/AUX/SLEEP/FLASH

void mp3\_set\_device (uint16\_t device) {

mp3\_send\_cmd (0x09, device);

}

void mp3\_sleep () {

mp3\_send\_cmd (0x0a);

}

void mp3\_reset () {

mp3\_send\_cmd (0x0c);

}

void mp3\_play () {

mp3\_send\_cmd (0x0d);

}

void mp3\_pause () {

mp3\_send\_cmd (0x0e);

}

void mp3\_stop () {

mp3\_send\_cmd (0x16);

}

//play mp3 file in mp3 folder in your tf card

void mp3\_play (uint16\_t num) {

mp3\_send\_cmd (0x12, num);

}

void mp3\_get\_state () {

mp3\_send\_cmd (0x42);

}

void mp3\_get\_volume () {

mp3\_send\_cmd (0x43);

}

void mp3\_get\_u\_sum () {

mp3\_send\_cmd (0x47);

}

void mp3\_get\_tf\_sum () {

mp3\_send\_cmd (0x48);

}

void mp3\_get\_flash\_sum () {

mp3\_send\_cmd (0x49);

}

void mp3\_get\_tf\_current () {

mp3\_send\_cmd (0x4c);

}

void mp3\_get\_u\_current () {

mp3\_send\_cmd (0x4b);

}

void mp3\_get\_flash\_current () {

mp3\_send\_cmd (0x4d);

}

void mp3\_single\_loop (boolean state) {

mp3\_send\_cmd (0x19, !state);

}

//add

void mp3\_single\_play (uint16\_t num) {

mp3\_play (num);

delay (10);

mp3\_single\_loop (true);

//mp3\_send\_cmd (0x19, !state);

}

void mp3\_DAC (boolean state) {

mp3\_send\_cmd (0x1a, !state);

}

void mp3\_random\_play () {

mp3\_send\_cmd (0x18);

}

//运行代码

#include <DFPlayer\_Mini\_Mp3.h>

#include <SoftwareSerial.h>

//实例化软串口

SoftwareSerial mySerial(10, 11); // RX, TX

void setup () {

Serial.begin (9600);

mySerial.begin (9600);

mp3\_set\_serial (mySerial); //set softwareSerial for DFPlayer-mini mp3 module

delay(1); //wait 1ms for mp3 module to set volume

mp3\_set\_volume (15);

}

void loop () {

mp3\_play (1); //播放第一首曲目

delay (6000);

mp3\_next (); //下一曲

delay (6000);

mp3\_prev (); //上一曲

delay (6000);

mp3\_play (4); //播放第四首歌曲

delay (6000);

}

//呼吸灯

int ledPin = 10;

void setup() {

pinMode(ledPin,OUTPUT);

}

void loop(){

for (int value = 0 ; value < 255; value=value+1){

analogWrite(ledPin, value);

delay(5);

}

for (int value = 255; value >0; value=value-1){

analogWrite(ledPin, value);

delay(5);

}

}

//流水灯

void setup() {

pinMode(8, OUTPUT);

pinMode(9, OUTPUT);

pinMode(10, OUTPUT);

pinMode(11, OUTPUT);

pinMode(12, OUTPUT);

pinMode(13, OUTPUT);

}

void loop() {

digitalWrite(8, HIGH);

delay(100);

digitalWrite(9, HIGH);

delay(100);

digitalWrite(10,HIGH);

delay(100);

digitalWrite(11,HIGH);

delay(100);

digitalWrite(12,HIGH);

delay(100);

digitalWrite(13,HIGH);

delay(100);

digitalWrite(13, LOW);

delay(100);

digitalWrite(12, LOW);

delay(100);

digitalWrite(11, LOW);

delay(100);

digitalWrite(10, LOW);

delay(100);

digitalWrite(9, LOW);

delay(100);

digitalWrite(8, LOW);

delay(100);

}

# 第9章 基于温度感应的LED表情控制音乐水杯项目设计

### 9.2.1 DS18B20模块

#### 2.相关代码

#include <OneWire.h>

OneWire ds(10); // 连接Arduino开发板引脚10

void setup(void) {

Serial.begin(9600);

}

void loop(void) {

byte i;

byte present = 0;

byte type\_s;

byte data[12];

byte addr[8];

float celsius, fahrenheit;

if ( !ds.search(addr)) {

Serial.println("No more addresses.");

Serial.println();

ds.reset\_search();

delay(250);

return;

}

Serial.print("ROM =");

for( i = 0; i < 8; i++) {

Serial.write(' ');

Serial.print(addr[i], HEX);

}

if (OneWire::crc8(addr, 7) != addr[7]) {

Serial.println("CRC is not valid!");

return;

}

Serial.println();

//显示芯片类型

switch (addr[0]) {

case 0x10:

Serial.println(" Chip = DS18S20");

type\_s = 1;

break;

case 0x28:

Serial.println(" Chip = DS18B20");

type\_s = 0;

break;

case 0x22:

Serial.println(" Chip = DS1822");

type\_s = 0;

break;

default:

Serial.println("Device is not a DS18x20 family device.");

return;

}

ds.reset();

ds.select(addr);

ds.write(0x44,1); //开始转换

delay(1000);

present = ds.reset();

ds.select(addr);

ds.write(0xBE);

Serial.print(" Data = ");

Serial.print(present,HEX);

Serial.print(" ");

for ( i = 0; i < 9; i++) {

data[i] = ds.read();

Serial.print(data[i], HEX);

Serial.print(" ");

}

Serial.print(" CRC=");

Serial.print(OneWire::crc8(data, 8), HEX);

Serial.println();

// 把数据转换为实际温度

unsigned int raw = (data[1] << 8) | data[0];

if (type\_s) {

raw = raw << 3;

if (data[7] == 0x10) {

raw = (raw & 0xFFF0) + 12 - data[6];

}

} else {

byte cfg = (data[4] & 0x60);

if (cfg == 0x00) raw = raw << 3; // 9 bit的解析需93.75 ms

else if (cfg == 0x20) raw = raw << 2; // 10 bit 的解析需187.5 ms

else if (cfg == 0x40) raw = raw << 1; // 11 bit的解析需 375 ms

// 默认是12 bit的解析, 750 ms的转换时间

}

celsius = (float)raw / 16.0;

fahrenheit = celsius \* 1.8 + 32.0;

Serial.print(" Temperature = ");

Serial.print(celsius);

Serial.print(" Celsius, ");

Serial.print(fahrenheit);

Serial.println(" Fahrenheit");

}

### 9.2.2 MAX7219ENG模块

#### 2.相关代码

下面示例是用DS18B20控制温度，从而控制点阵显示不同图案的代码

#include <DS18B20.h>

#include <LedControl.h>

#include <OneWire.h>

#include <LedControl.h>

int DIN = 12;

int CS = 11;

int CLK = 10;

#include <OneWire.h>

#include <DS18B20.h>

DS18B20 ds(2); //DS18B20传感器信号引脚接Arduino开发板的引脚2

LedControl lc=LedControl(DIN,CLK,CS,0);

void setup(){

Serial.begin(9600);

lc.shutdown(0,false); //MAX72XX开始处于节能模式

lc.setIntensity(0,15); // 设置亮度到最大

lc.clearDisplay(0); // 清除显示

}

void loop(){

byte smile[8]= {0xFF,0x99,0xA5,0xC3,0xBD,0xA5,0xA5,0xFF};//快乐

byte neutral[8]= {0x00,0xEE,0x22,0x00,0x00,0x3E,0x02,0x00};//悲伤

byte frown[8]= {0x00,0x24,0x5A,0x00,0x00,0x3C,0x00,0x00};//平静

Serial.print("Current Temperature is ");//串口输出温度数值

Serial.print(ds.getTempC());

Serial.println(" C");

float temp\_val=ds.getTempC();

if (temp\_val>30){

printByte(neutral);

delay(1000);

lc.clearDisplay(0);

delay(1000);

}else if (temp\_val>20){

printByte(smile);

delay(1000);

lc.clearDisplay(0);

delay(1000);

}else if (temp\_val>10){

printByte(frown);

delay(1000);

lc.clearDisplay(0);

delay(1000);

}

}

void printByte(byte character [])

{

int i = 0;

for(i=0;i<8;i++)

{

lc.setRow(0,i,character[i]);

}

}

### 9.2.3 音乐输出模块

#### 2.相关代码

通过传感器来采集温度，以及对扬声器设置不同的频率，利用定时器中断来完成对音乐节拍长度的控制，实现扬声器播放乐曲。

//1．当温度超过20℃小于30℃时播放《葫芦娃》

#define NOTE\_D0 -1

#define NOTE\_D1 294

#define NOTE\_D2 330

#define NOTE\_D3 350

#define NOTE\_D4 393

#define NOTE\_D5 441

#define NOTE\_D6 495

#define NOTE\_D7 556

#define NOTE\_DL1 147

#define NOTE\_DL2 165

#define NOTE\_DL3 175

#define NOTE\_DL4 196

#define NOTE\_DL5 221

#define NOTE\_DL6 248

#define NOTE\_DL7 278

#define NOTE\_DH1 589

#define NOTE\_DH2 661

#define NOTE\_DH3 700

#define NOTE\_DH4 786

#define NOTE\_DH5 882

#define NOTE\_DH6 990

#define NOTE\_DH7 112

/\*以上定义是把每个音符和频率值对应起来，后面可以随意编写D调的各种歌曲。这里用NOTE\_D+数字表示音符，NOTE\_DH+数字表示上面高音音符，NOTE\_DL+数字表示低音音符\*/

#define WHOLE 1

#define HALF 0.5

#define QUARTER 0.25

#define EIGHTH 0.25

#define SIXTEENTH 0.625

//用英文对应节拍

int tune[] =

{

NOTE\_DH1,NOTE\_D6,NOTE\_D5,NOTE\_D6,NOTE\_D0,

NOTE\_DH1,NOTE\_D6,NOTE\_D5,NOTE\_DH1,NOTE\_D6,NOTE\_D0,NOTE\_D6,

NOTE\_D6,NOTE\_D6,NOTE\_D5,NOTE\_D6,NOTE\_D0,NOTE\_D6,

NOTE\_DH1,NOTE\_D6,NOTE\_D5,NOTE\_DH1,NOTE\_D6,NOTE\_D0,

NOTE\_D1,NOTE\_D1,NOTE\_D3,

NOTE\_D1,NOTE\_D1,NOTE\_D3,NOTE\_D0,

NOTE\_D6,NOTE\_D6,NOTE\_D6,NOTE\_D5,NOTE\_D6,

NOTE\_D5,NOTE\_D1,NOTE\_D3,NOTE\_D0,

NOTE\_DH1,NOTE\_D6,NOTE\_D6,NOTE\_D5,NOTE\_D6,

NOTE\_D5,NOTE\_D1,NOTE\_D2,NOTE\_D0,

NOTE\_D7,NOTE\_D7,NOTE\_D5,NOTE\_D3,

NOTE\_D5,

NOTE\_DH1,NOTE\_D0,NOTE\_D6,NOTE\_D6,NOTE\_D5,NOTE\_D5,NOTE\_D6,NOTE\_D6,

NOTE\_D0,NOTE\_D5,NOTE\_D1,NOTE\_D3,NOTE\_D0,

NOTE\_DH1,NOTE\_D0,NOTE\_D6,NOTE\_D6,NOTE\_D5,NOTE\_D5,NOTE\_D6,NOTE\_D6,

NOTE\_D0,NOTE\_D5,NOTE\_D1,NOTE\_D2,NOTE\_D0,

NOTE\_D3,NOTE\_D3,NOTE\_D1,NOTE\_DL6,

NOTE\_D1,

NOTE\_D3,NOTE\_D5,NOTE\_D6,NOTE\_D6,

NOTE\_D3,NOTE\_D5,NOTE\_D6,NOTE\_D6,

NOTE\_DH1,NOTE\_D0,NOTE\_D7,NOTE\_D5,

NOTE\_D6,

};//整首曲子的音符部分，用一个序列整数定义

float duration[]=

{

1,1,0.5,0.5,1,

0.5,0.5,0.5,0.5,1,0.5,0.5,

0.5,1,0.5,1,0.5,0.5,

0.5,0.5,0.5,0.5,1,1,

1,1,1+1,

0.5,1,1+0.5,1,

1,1,0.5,0.5,1,

0.5,1,1+0.5,1,

0.5,0.5,0.5,0.5,1+1,

0.5,1,1+0.5,1,

1+1,0.5,0.5,1,

1+1+1+1,

0.5,0.5,0.5+0.25,0.25,0.5+0.25,0.25,0.5+0.25,0.25,

0.5,1,0.5,1,1,

0.5,0.5,0.5+0.25,0.25,0.5+0.25,0.25,0.5+0.25,0.25,

0.5,1,0.5,1,1,

1+1,0.5,0.5,1,

1+1+1+1,

0.5,1,0.5,1+1,

0.5,1,0.5,1+1,

1+1,0.5,0.5,1,

1+1+1+1

};//整首曲子的接拍部分，定义序列duration，浮点（数组的个数和前面音符的个数是一一对应的）

int length;//定义一个变量，后面用来表示共有多少个音符

int tonePin=5; // 扬声器的引脚

void setup()

{

pinMode(tonePin,OUTPUT); //设置扬声器的引脚为输出模式

length = sizeof(tune)/sizeof(tune[0]); //用sizeof函数，可以查出tone序列里有多少个音符

}

void loop()

{

for(int x=0;x<length;x++) //循环音符的次数

{

tone(tonePin,tune[x]); //此函数依次播放tune序列里的数组，即每个音符

delay(400\*duration[x]); //每个音符持续的时间，即节拍duration，400是调整时间，值越大曲子速度越慢，反之，越快

noTone(tonePin); //停止当前音符，进入下一音符

}

delay(5000); //等待5s后，循环重新开始

}

//2．当温度超过20℃小于30℃时播放《蓝精灵》

#define C3 165

#define C4 175

#define C5 196

#define C6 220

#define C7 247

#define D0 -1

#define D1 262

#define D2 294

#define D3 330

#define D4 349

#define D4s 370

#define D5 392

#define D6 440

#define D7 494

#define E1 523

#define E1s 554

#define E2 587

#define E3 659

#define E4 698

#define E4s 740

#define E5 784

#define E6 880

//音阶对应的频率

#define WHOLE 4

#define HALF 2

#define QTR 1

//定义全拍和半拍

int tune[] =

{

D3,D4, D5,E3,E1,D5,D3,D2,D3,D4,E2,D7,D4,D2,D1,D2,D3,D2,D3,D4,D3,D4,

D5,D3,D4,D6,D5,D6, D7,E1,E1s,E2,E2,E3,E4,E4s,E5,D0,C5, D1,D3,C5,D3,D1,D3,C5,

D3,D4,D5,D4,D3,D4,D5,D4,D3,D4,D5,D5,D4s,D5,E3,E1,

//在山的那边海的那边有一群蓝精灵

D2,D3,D4,D3,D4,E2,D7,D2,D3,D4,D3,D4,D6,D5,

//他们活泼又聪明，他们调皮又灵敏

D3,D4,D5,D4,D3,D4,D5,D4,D3,D4,D5,D5,D4s,D5,E3,E2,

//他们自由自在生活在那绿色的大森林

D2,D3,D4,D3,D4,E2,E1,D7,D6,D7,E1,

//他们善良勇敢相互都关心

E1,D5,D5,E3,E3,E1,D5,D3, D3,D5,D4,E2,E2,D7,D4,D2,

//哦，可爱的蓝精灵，哦，可爱的蓝精灵

D2,D3,D4,D5,D4,D3,D4,D5,D4,D3,D4,D5,D5,D4s,D5,E1,E3,

//他们齐心合力开动脑筋斗败了格格巫

D2,D3,D4,D3,D4,E2,E1,D7,D6,D7,E1,

//他们唱歌跳舞快乐多欢欣

}; //曲子的音符部分

int duration[]=

{

1,1,2,2,2,2,4+2,1,1,2,2,2,2,4+2,1,1,2,1,1,2,1,1,

2,1,1,2,1,1,1,1,1,1,1,1,1,1,2,2,4,2,2,2,2,2,2,2,

1,1,2,2,2,2,2,2,2,2,2,1,1,2,2,4+2,

1,1,2,2,2,2,4+2,1,1,2,2,2,2,4+2,

1,1,2,2,2,2,2,2,2,2,2,1,1,2,2,4+2,

1,1,2,2,2,2,2,2,2,2,4,

4,4,4,2,2,4,4,4+4,4,4,4,2,2,4,4,4+4,

4+2,1,1,2,2,2,2,2,2,2,2,2,1,1,2,2,4+2,

1,1,2,2,2,2,2,2,2,2,4+4

};

//曲子的节拍部分，用一个序列，定义为duration

int length; //定义一个变量，后面用来表示共有多少个音符

int tonePin=5; //扬声器用的引脚

void setup()

{

pinMode(tonePin,OUTPUT); //设置扬声器的引脚为输出模式

length = sizeof(tune)/sizeof(tune[0]);

}

void loop()

{

for(int x=0;x<length;x++) //循环音符的次数

{

tone(tonePin,tune[x]); //此函数依次播放tune序列里的数组，即每个音符

delay(100\*duration[x]); //每个音符持续的时间，即节拍duration，100是调整时间，值越大曲子速度越慢，反之，越快

noTone(tonePin); //停止当前音符，进入下一音符

}

delay(2000);//等待2s后，循环重新开始

}

# 第10章 旋转音乐盒项目设计

### 10.2.1 主程序模块

#### 2.相关代码

##### 1）音调部分头文件代码

//D调

#define NOTE\_D0 -1

#define NOTE\_D1 294

#define NOTE\_D2 330

#define NOTE\_D3 350

#define NOTE\_D4 393

#define NOTE\_D5 441

#define NOTE\_D6 495

#define NOTE\_D7 556

#define NOTE\_DL1 147

#define NOTE\_DL2 165

#define NOTE\_DL3 175

#define NOTE\_DL4 196

#define NOTE\_DL5 221

#define NOTE\_DL6 248

#define NOTE\_DL7 278

#define NOTE\_DH1 589

#define NOTE\_DH2 661

#define NOTE\_DH3 700

#define NOTE\_DH4 786

#define NOTE\_DH5 882

#define NOTE\_DH6 990

#define NOTE\_DH7 1112

//G调

#define NOTE\_G0 -1

#define NOTE\_G1 393

#define NOTE\_G2 441

#define NOTE\_G3 495

#define NOTE\_G4 556

#define NOTE\_G5 624

#define NOTE\_G6 661

#define NOTE\_G7 742

#define NOTE\_GL1 196

#define NOTE\_GL2 221

#define NOTE\_GL3 234

#define NOTE\_GL4 262

#define NOTE\_GL5 294

#define NOTE\_GL6 330

#define NOTE\_GL7 371

#define NOTE\_GH1 786

#define NOTE\_GH2 882

#define NOTE\_GH3 990

#define NOTE\_GH4 1049

#define NOTE\_GH5 1178

#define NOTE\_GH6 1322

#define NOTE\_GH7 1484

//以上是把每个音符和频率值进行对应，在这里只放了歌曲中用到的D调和G调部分，项目文件中存放了完整的各种音调所对应的频率，方便后面的音乐编写

#define WHOLE 1

#define HALF 0.5

#define QUARTER 0.25

#define EIGHTH 0.25

#define SIXTEENTH 0.625

//以上是用英文对应了拍子，便于后面音乐节奏的编写

##### 2）歌曲部分头文件代码

//定义乐曲1

int tune1[] =

{

NOTE\_G6,NOTE\_GH1,NOTE\_G7,NOTE\_G6,NOTE\_G5,NOTE\_G6,

NOTE\_G3,NOTE\_G6,NOTE\_G5,

NOTE\_G6,NOTE\_GH1,NOTE\_G7,NOTE\_GH1,NOTE\_G7,NOTE\_G6,NOTE\_G7,

NOTE\_GH1,NOTE\_GH1,NOTE\_GH2,

NOTE\_GH3,NOTE\_GH3,NOTE\_GH3,NOTE\_GH3,NOTE\_GH2,

NOTE\_GH1,NOTE\_G7,NOTE\_G6,NOTE\_G7,NOTE\_G5,

NOTE\_G6,NOTE\_GH1,NOTE\_G7,NOTE\_G6,NOTE\_G5,

NOTE\_G6,NOTE\_G0,

NOTE\_G0,NOTE\_G0,NOTE\_G0,NOTE\_G0,

};//星语心愿

float duration1[]=

{

1+0.5,0.5+0.5,0.5,0.5,0.5,0.5,

1+1+1,0.5,0.5,

1+0.5,0.5,0.5,0.25,0.25,0.5,0.5,

1+1+1,0.5,0.5,

0.5,0.5,0.5,0.5,1+1,

0.5,0.5,0.5,0.5,1+1,

1,1,1,0.5,0.5,

1+1+1,1,

1,1,1,1,

};

//定义乐曲2

int tune2[] =

{

NOTE\_D3,NOTE\_D3,NOTE\_D4,NOTE\_D5,NOTE\_D5,NOTE\_D4,NOTE\_D3,NOTE\_D2,

NOTE\_D1,NOTE\_D1,NOTE\_D2,NOTE\_D3,NOTE\_D3,NOTE\_D2,NOTE\_D2,

NOTE\_D0,

NOTE\_D3,NOTE\_D3,NOTE\_D4,NOTE\_D5,NOTE\_D5,NOTE\_D4,NOTE\_D3,NOTE\_D2,

NOTE\_D1,NOTE\_D1,NOTE\_D2,NOTE\_D3,NOTE\_D2,NOTE\_D1,NOTE\_D1,

NOTE\_D0,

NOTE\_D2,NOTE\_D2,NOTE\_D3,NOTE\_D1,NOTE\_D2,NOTE\_D3,NOTE\_D4,NOTE\_D3,NOTE\_D1,

NOTE\_D2,NOTE\_D3,NOTE\_D4,NOTE\_D3,NOTE\_D2,NOTE\_D1,NOTE\_D2,NOTE\_DL5,

NOTE\_D0,

NOTE\_D3,NOTE\_D3,NOTE\_D4,NOTE\_D5,NOTE\_D5,NOTE\_D4,NOTE\_D3,NOTE\_D2,

NOTE\_D1,NOTE\_D1,NOTE\_D2,NOTE\_D3,NOTE\_D2,NOTE\_D1,NOTE\_D1,

};//欢乐颂

float duration2[]=

{

1,1,1,1,1,1,1,1,1,1,1,1,2,0.5,1,

1,

1,1,1,1,1,1,1,1,1,1,1,1,2,0.5,1.5,

1,

1,1,1,1,1,0.5,0.5,1,1,1,0.5,0.5,1,1,1,1,1.5,

1,

1,1,1,1,1,1,1,1,1,1,1,1,2,0.5,1.5,

};

##### 3）主函数部分

#include "yindiao.h"

#include "song.h"

#include <Servo.h> // 舵机

Servo myservo;

int pos = 0;

int length;//定义音符的个数

int tonePin=8;//蜂鸣器的引脚

int Sensor\_pin = 10;//人体红外引脚

void setup()

{

myservo.attach(9);//舵机

pinMode(Sensor\_pin,INPUT); //红外

pinMode(tonePin,OUTPUT);//蜂鸣器

pinMode(1,OUTPUT);//LED1

pinMode(2,OUTPUT);// LED 2

pinMode(3,OUTPUT);// LED 3

pinMode(4,OUTPUT);// LED 4

digitalWrite(1,LOW);

digitalWrite(2,LOW);//设置LED1、LED2为低电平

length = sizeof(tune1)/sizeof(tune1[0]);

length = sizeof(tune2)/sizeof(tune2[0]);//用sizeof函数查看tone序列里共有多少个音符

}

void loop()

{

int val=digitalRead(Sensor\_pin); //定义参数存储人体红外传感器读到的状态

if(val == 1) //如果检测到有动物运动（在检测范围内）

{

servo();

}

else

{

return;

}

delay(5000);

}

void servo()

{

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);//设置LED3和4为高电平点亮

int a;

for(a=0;a<1;a++)

{

for(pos = 0; pos < 180; pos += 1) // 从0~180°

{

myservo.write(pos);

delay(10);

}

singsong1();

for(pos = 180; pos>=1; pos-=1) // 从180~0°

{

myservo.write(pos);

delay(20);

}

singsong2();

digitalWrite(3,LOW);

digitalWrite(4,LOW);

}

}

void singsong1()

{

for(int x=0;x<length;x++)//循环音符的次数

{

tone(tonePin,tune1[x]);//此函数依次播放tune序列里的数组，即每个音符

digitalWrite(1,HIGH);

delay(400\*duration1[x]);//每个音符持续的时间，即节拍duration，调整时间越大曲子速度越慢，反之，曲子速度越快

digitalWrite(1,LOW);

delay(100\*duration1[x]);//使LED1跟随歌曲的节奏闪烁

noTone(tonePin);//停止当前音符，进入下一音符

}

digitalWrite(1,LOW);

delay(5000);//可以等待5s后，循环重新开始

}

void singsong2()

{

for(int x=0;x<length;x++)//循环音符的次数

{

tone(tonePin,tune1[x]);//此函数依次播放tune序列里的数组，即每个音符

digitalWrite(2,HIGH);

delay(300\*duration2[x]);//每个音符持续的时间，即节拍duration，调整时间越大曲子速度越慢，反之，曲子速度越快

digitalWrite(2,LOW);

delay(100\*duration2[x]);//使LED2跟随歌曲的节奏闪烁

noTone(tonePin);//停止当前音符，进入下一音符

}

delay(5000);//等待5s后，循环重新开始

digitalWrite(2,LOW);

}

### 10.2.2 人体红外感应模块

#### 2.相关代码

void loop()

{

int val=digitalRead(Sensor\_pin); //定义参数存储人体红外传感器读到的状态

if(val == 1) //如果检测到有动物运动（在检测范围内）

{

servo();

}

else

{

return;

}

delay(5000);

}

### 10.2.3 输出模块

#### 2.相关代码

void servo()

{

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);//设置LED3和4为高电平点亮

int a;

for(a=0;a<1;a++)

{

for(pos = 0; pos < 180; pos += 1) // 从0~180°

{

myservo.write(pos);

delay(10);

}

singsong1();

for(pos = 180; pos>=1; pos-=1) // 从180~0°

{

myservo.write(pos);

delay(20);

}

singsong2();

digitalWrite(3,LOW);

digitalWrite(4,LOW);

}

}

# 第11章 蓝牙音乐播放器项目设计

### 11.2.1 SD卡模块

#### 2.相关代码

#include <SimpleSDAudio.h>

void DirCallback(char \*buf) {

Serial.println(buf);

}

char AudioFileName[16];

#define BIGBUFSIZE (2\*512)

uint8\_t bigbuf[BIGBUFSIZE];

int freeRam () {

extern int \_\_heap\_start, \*\_\_brkval;

int v;

return (int) &v - (\_\_brkval == 0 ? (int) &\_\_heap\_start : (int) \_\_brkval);

}

void setup()

{

Serial.begin(9600);

while (!Serial) {

;

}

Serial.print(F("Free Ram: "));

Serial.println(freeRam());

SdPlay.setWorkBuffer(bigbuf, BIGBUFSIZE);

Serial.print(F("\nInitializing SimpleSDAudio V" SSDA\_VERSIONSTRING " ..."));

//SdPlay.setSDCSPin(10);

if (!SdPlay.init(SSDA\_MODE\_FULLRATE | SSDA\_MODE\_MONO)) {

Serial.println(F("initialization failed. Things to check:"));

Serial.println(F("\* is a card is inserted?"));

Serial.println(F("\* Is your wiring correct?"));

Serial.println(F("\* maybe you need to change the chipSelect pin to match your shield or module?"));

Serial.print(F("Error code: "));

Serial.println(SdPlay.getLastError());

while(1);

} else {

Serial.println(F("Wiring is correct and a card is present."));

}

}

void loop(void) {

uint8\_t count=0, c, flag;

Serial.println(F("Files on card:"));

SdPlay.dir(&DirCallback);

ReEnter:

count = 0;

Serial.println(F("\r\nEnter filename (send newline after input):"));

do {

while(!Serial.available()) ;

c = Serial.read();

if(c > ' ') AudioFileName[count++] = c;

} while((c != 0x0d) && (c != 0x0a) && (count < 14));

AudioFileName[count++] = 0;

Serial.print(F("Looking for file... "));

if(!SdPlay.setFile(AudioFileName)) {

Serial.println(F(" not found on card! Error code: "));

Serial.println(SdPlay.getLastError());

goto ReEnter;

} else {

Serial.println(F("found."));

}

Serial.println(F("Press s for stop, p for play, h for pause, f to select new file, d for deinit, v to view status."));

flag = 1;

while(flag) {

SdPlay.worker();

if(Serial.available()) {

c = Serial.read();

switch(c) {

case 's':

SdPlay.stop();

Serial.println(F("Stopped."));

break;

case 'p':

SdPlay.play();

Serial.println(F("Play."));

break;

case 'h':

SdPlay.pause();

Serial.println(F("Pause."));

break;

case 'd':

SdPlay.deInit();

Serial.println(F("SdPlay deinitialized. You can now safely remove card. System halted."));

while(1) ;

break;

case 'f':

flag = 0;

break;

case 'v':

Serial.print(F("Status: isStopped="));

Serial.print(SdPlay.isStopped());

Serial.print(F(", isPlaying="));

Serial.print(SdPlay.isPlaying());

Serial.print(F(", isPaused="));

Serial.print(SdPlay.isPaused());

Serial.print(F(", isUnderrunOccured="));

Serial.print(SdPlay.isUnderrunOccured());

Serial.print(F(", getLastError="));

Serial.println(SdPlay.getLastError());

Serial.print(F("Free RAM: "));

Serial.println(freeRam());

break;

}

}

}

}

### 11.2.2 HC-06模块

#### 2.相关代码

#include <SoftwareSerial.h>

void setup(){

Serial.begin(9600); //初始化串口通信，并将波特率设置为9600

}

void loop(){

if (Serial.available()) {

val = Serial.read();

BT.print(val);

}

//如果接收到蓝牙模块的数据，输出到屏幕

if (BT.available()) {

val = BT.read();

Serial.print(val);

}

ReEnter:

count = 0;

Serial.println(F("\r\nEnter filename (send newline after input):"));

do {

while(!Serial.available()) ;

c = Serial.read();

if(c > ' ') AudioFileName[count++] = c;

}

while((c != 0x0d) && (c != 0x0a) && (count < 14));

AudioFileName[count++] = 0;

Serial.print(F("Looking for file... "));

if(!SdPlay.setFile(AudioFileName)) {

Serial.println(F(" not found on card! Error code: "));

Serial.println(SdPlay.getLastError());

goto ReEnter;

}

}

### 11.2.3 LCD1602模块

#### 2.相关代码

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

LiquidCrystal\_I2C lcd(0x27,16,2);

void setup(){

lcd.init();

lcd.backlight();

}

void loop(void) {

if(!SdPlay.setFile(AudioFileName)) {

Serial.println(F(" not found on card! Error code: "));

Serial.println(SdPlay.getLastError());

lcd.setCursor(0,1);

lcd.print("Input again ");

goto ReEnter;

} else {

Serial.println(F("found."));

lcd.print("Current state:");

lcd.setCursor(0,1);

lcd.print("Song is found ");

}

Serial.println(F("Press s for stop, p for play, h for pause, f to select new file, d for deinit, v to view status."));

flag = 1;

while(flag) {

SdPlay.worker();

if(Serial.available()) {

c = Serial.read();

switch(c) {

case 's':

SdPlay.stop();

Serial.println(F("Stopped."));

lcd.setCursor(0,1);

lcd.print("Song is stopped ");

break;

case 'p':

SdPlay.play();

Serial.println(F("Play."));

lcd.setCursor(0,1);

lcd.print("Song is playing ");

break;

case 'h':

SdPlay.pause();

Serial.println(F("Pause."));

lcd.setCursor(0,1);

lcd.print("Song is paused ");

break;

case 'd':

SdPlay.deInit();

Serial.println(F("SdPlay deinitialized. You can now safely remove card. System halted."));

while(1) ;

break;

case 'f':

lcd.setCursor(0,1);

lcd.print("Ready to select ");

flag = 0;

break;

}

### 11.2.4 音频放大电路模块

# 第12章 吉他手套项目设计

### 12.2.1 弯曲传感器模块

#### 2.相关代码

#define AD1 A1 //定义引脚A1

#define AD2 A2 //定义引脚A2

#define AD3 A3 //定义引脚A3

#define AD4 A4 //定义引脚A4

#define AD5 A5 //定义引脚A5

int Intensity1 = 0;

int Intensity2 = 0;

int Intensity3 = 0;

int Intensity4 = 0;

int Intensity5 = 0;//引脚数值

void setup() //程序初始化

{

Serial.begin(9600);//设置波特率9600

}

void loop()//程序主体循环

{

Intensity5 = analogRead(AD5); //读取引脚AD5的值，存入Intensity变量

Serial.print("Intensity5 = "); //串口输出"Intensity = "

Serial.println(Intensity5); //串口输出Intensity的变量值，并换行

Intensity4 = analogRead(AD4); //读取引脚AD5的值，存入Intensity变量

Serial.print("Intensity4 = "); //串口输出"Intensity = "

Serial.println(Intensity4); //串口输出Intensity的变量值，并换行

Intensity3 = analogRead(AD3); //读取引脚AD5的值，存入Intensity变量

Serial.print("Intensity3 = "); //串口输出"Intensity = "

Serial.println(Intensity3); //串口输出Intensity的变量值，并换行

Intensity2 = analogRead(AD2); //读取引脚AD5的值，存入Intensity变量

Serial.print("Intensity2 = "); //串口输出"Intensity = "

Serial.println(Intensity2); //串口输出Intensity的变量值，并换行

Intensity1 = analogRead(AD1); //读取引脚AD1的值，存入Intensity变量

Serial.print("Intensity1 = "); //串口输出"Intensity = "

Serial.println(Intensity1); //串口输出Intensity的变量值，并换行

delay(500); //延时500ms

}

### 12.2.2输出模块

#include <SimpleSDAudio.h>

void DirCallback(char \*buf) {

Serial.println(buf);

}

char AudioFileName[16];

#define BIGBUFSIZE (2\*512)

uint8\_t bigbuf[BIGBUFSIZE];

int freeRam () {

extern int \_\_heap\_start, \*\_\_brkval;

int v;

return (int) &v - (\_\_brkval == 0 ? (int) &\_\_heap\_start : (int) \_\_brkval);

}

void setup()

{

Serial.begin(9600);

while (!Serial) {

;

}

Serial.print(F("Free Ram: "));

Serial.println(freeRam());

SdPlay.setWorkBuffer(bigbuf, BIGBUFSIZE);

Serial.print(F("\nInitializing SimpleSDAudio V" SSDA\_VERSIONSTRING " ..."));

if (!SdPlay.init(SSDA\_MODE\_FULLRATE | SSDA\_MODE\_MONO | SSDA\_MODE\_AUTOWORKER)) {

Serial.println(F("initialization failed. Things to check:"));

Serial.println(F("\* is a card is inserted?"));

Serial.println(F("\* Is your wiring correct?"));

Serial.println(F("\* maybe you need to change the chipSelect pin to match your shield or module?"));

Serial.print(F("Error code: "));

Serial.println(SdPlay.getLastError());

while(1);

} else {

Serial.println(F("Wiring is correct and a card is present."));

}

}

void loop(void) {

uint8\_t count=0, c, flag;

Serial.println(F("Files on card:"));

SdPlay.dir(&DirCallback);

ReEnter:

count = 0;

Serial.println(F("\r\nEnter filename (send newline after input):"));

do {

while(!Serial.available()) ;

c = Serial.read();

if(c > ' ') AudioFileName[count++] = c;

} while((c != 0x0d) && (c != 0x0a) && (count < 14));

AudioFileName[count++] = 0;

Serial.print(F("Looking for file... "));

if(!SdPlay.setFile(AudioFileName)) {

Serial.println(F(" not found on card! Error code: "));

Serial.println(SdPlay.getLastError());

goto ReEnter;

} else {

Serial.println(F("found."));

}

Serial.println(F("Press s for stop, p for play, h for pause, f to select new file, d for deinit, v to view status."));

flag = 1;

while(flag) {

SdPlay.worker();

if(Serial.available()) {

c = Serial.read();

switch(c) {

case 's':

SdPlay.stop();

Serial.println(F("Stopped."));

break;

case 'p':

SdPlay.play();

Serial.println(F("Play."));

break;

case 'h':

SdPlay.pause();

Serial.println(F("Pause."));

break;

case 'd':

SdPlay.deInit();

Serial.println(F("SdPlay deinitialized. You can now safely remove card. System halted."));

while(1) ;

break;

case 'f':

flag = 0;

break;

case 'v':

Serial.print(F("Status: isStopped="));

Serial.print(SdPlay.isStopped());

Serial.print(F(", isPlaying="));

Serial.print(SdPlay.isPlaying());

Serial.print(F(", isPaused="));

Serial.print(SdPlay.isPaused());

Serial.print(F(", isUnderrunOccured="));

Serial.print(SdPlay.isUnderrunOccured());

Serial.print(F(", getLastError="));

Serial.println(SdPlay.getLastError());

Serial.print(F("Free RAM: "));

Serial.println(freeRam());

break;

}

}

}

}

上述代码需要使用另外的Arduino库文件，同时以此方式做出的音频播放器仅支持.afm格式，下载SimpleSDAudio库文件，安装后打开tools选择需要转换的格式，推荐使用全速单通道模式，把后缀为WAV的音乐文件拖进批处理中，转换结束后按任意键退出。转换出的afm格式音乐文件会出现在converted里，这时候音乐文件准备就绪。将上步转换出来的.afm文件复制到SD卡中即可。(附件下载地址http://hackerspace-ffm.de/wiki/index.php?title=Datei:SimpleSDAudio\_V1.03.zip )，音频格式转换操作如图12-6所示。

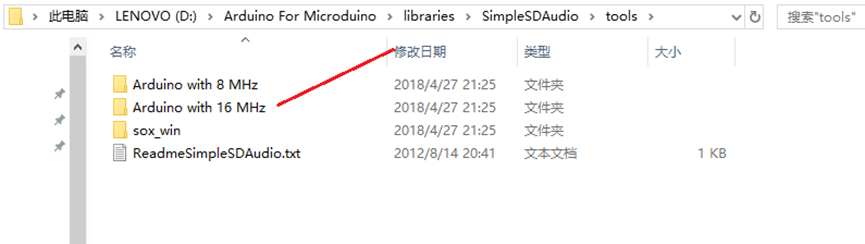


图12-6 音频格式转换操作

# 第13章 音频文件净化器

### 13.2.1 SD卡读取模块

#### 2.相关代码

#include "SD.h"  //读取SD卡的库   
#include "SimpleSDAudio"  //播放音频文件的库  
#include "SPI.h"  //SD卡的SPI库  
void setup()

{

Serial.begin(9600);

while (!Serial) {

}

Serial.print(F("\nInitializing SD card...")); //串口提示信息

if (!SdPlay.init(SSDA\_MODE\_FULLRATE | SSDA\_MODE\_MONO)) {

Serial.println(F("initialization failed. ")); //读取失败

Serial.println(SdPlay.getLastError());

while(1);

} else {

Serial.println(F("Wiring is correct and a card is present."));//成功读取

}

pinMode(2, INPUT\_PULLUP); //设置数字读取引脚，以便后续设置按钮

}

int song\_number=0; //歌曲代号，改变播放更多曲目

boolean debounce=true;

void loop(void) {

if (digitalRead(2)==LOW && debounce == true){ //按键被按下

song\_number++;

if (song\_number==3) //歌曲最大循环数目

{song\_number=1;}

if (song\_number ==1){ //播放歌曲一

delay(3000);

Serial.print(F("Looking for 1.AFM... "));

if(!SdPlay.setFile("2.AFM")) {

Serial.println(F(" not found on card! Error code: "));

Serial.println(SdPlay.getLastError());

while(1);

} else {

Serial.println(F("found."));

}

Serial.print(F("Playing... "));

SdPlay.play();

while(!SdPlay.isStopped()) {

SdPlay.worker();

}

Serial.println(F("done."));}

if (song\_number ==2){ //播放歌曲二

delay(3000);

Serial.print(F("Looking for 2.AFM... "));

if(!SdPlay.setFile("1.AFM")) {

Serial.println(F(" not found on card! Error code: "));

continue; //如不存在则继续循环

Serial.println(SdPlay.getLastError());

while(1);

} else {

Serial.println(F("found."));

}

Serial.print(F("Playing... "));

SdPlay.play();

while(!SdPlay.isStopped()) {

SdPlay.worker();

}

Serial.println(F("done."));}

}

if (digitalRead(2)==HIGH) //Avoid debounce

{debounce = true; }

}

### 13.2.2 数字信号处理模块

#### 2.相关代码

class Complex    
{    
public:    
    Complex(){real=0;imag=0;}    
    Complex(double r,double i){real=r; imag=i;}    
    Complex operator+(const Complex &c2);    
    Complex operator-(const Complex &c2);    
    Complex operator\*(const Complex &c2);    
    Complex operator/(const Complex &c2);    
    void display();    
private:    
    double real;    
    double imag;    
};    
//下面定义成员函数    
//复数相加： (a+bi)+(c+di)=(a+c)+(b+d)i.    
Complex Complex::operator+(const Complex &c2)    
{    
    Complex c;    
    c.real=real+c2.real;    
    c.imag=imag+c2.imag;    
    return c;    
}    
//复数相减：(a+bi)-(c+di)=(a-c)+(b-d)i.    
Complex Complex::operator-(const Complex &c2)    
{    
    Complex c;    
    c.real=real-c2.real;    
    c.imag=imag-c2.imag;    
    return c;    
}    
//复数相乘：(a+bi)(c+di)=(ac－bd)+(bc+ad)i.    
Complex Complex::operator\*(const Complex &c2)    
{    
    Complex c;    
    c.real=real\*c2.real-imag\*c2.imag;    
    c.imag=imag\*c2.real+real\*c2.imag;    
    return c;    
}    
//复数相除：(a+bi)/(c+di)=(ac+bd)/(c^2+d^2) +(bc-ad)/(c^2+d^2)i    
Complex Complex::operator/(const Complex &c2)    
{    
    Complex c;    
    c.real=(real\*c2.real+imag\*c2.imag)/(c2.real\*c2.real+c2.imag\*c2.imag);    
    c.imag=(imag\*c2.real-real\*c2.imag)/(c2.real\*c2.real+c2.imag\*c2.imag);    
    return c;    
}    
void Complex::display()    
{    
    cout<<"("<<real<<","<<imag<<"i)"<<endl;    
}

void bitrp (float xreal [], float ximag [], int n)

{

// 该程序完成了位反转置换功能

int i, j, a, b, p;

for (i = 1, p = 0; i < n; i \*= 2)

{

p ++;

}

for (i = 0; i < n; i ++)

{

a = i;

b = 0;

for (j = 0; j < p; j ++)

{

b = (b << 1) + (a & 1); // b = b \* 2 + a % 2;

a >>= 1; // a = a / 2;

}

if ( b > i)

{

swap (xreal [i], xreal [b]);

swap (ximag [i], ximag [b]);

}

}

}

void FFT(float xreal [], float ximag [], int n)

{

// 快速傅立叶变换，将复数x变换后仍保存在x中，xreal, ximag 分别是x的实部和虚部

float wreal [N / 2], wimag [N / 2], treal, timag, ureal, uimag, arg;

int m, k, j, t, index1, index2;

bitrp (xreal, ximag, n);

// 计算1的前n/ 2个n次方根的共轭复数 W'j = wreal [j] + i \* wimag [j] , j = 0, 1, ... , n / 2 - 1

arg = - 2 \* PI / n;

treal = cos (arg);

timag = sin (arg);

wreal [0] = 1.0;

wimag [0] = 0.0;

for (j = 1; j < n / 2; j ++)

{

wreal [j] = wreal [j - 1] \* treal - wimag [j - 1] \* timag;

wimag [j] = wreal [j - 1] \* timag + wimag [j - 1] \* treal;

}

for (m = 2; m <= n; m \*= 2)

{

for (k = 0; k < n; k += m)

{

for (j = 0; j < m / 2; j ++)

{

index1 = k + j;

index2 = index1 + m / 2;

t = n \* j / m; // 旋转因子w的实部在wreal []中的下标为t

treal = wreal [t] \* xreal [index2] - wimag [t] \* ximag [index2];

timag = wreal [t] \* ximag [index2] + wimag [t] \* xreal [index2];

ureal = xreal [index1];

uimag = ximag [index1];

xreal [index1] = ureal + treal;

ximag [index1] = uimag + timag;

xreal [index2] = ureal - treal;

ximag [index2] = uimag - timag;

}

}

}

}

void IFFT (float xreal [], float ximag [], int n)

{

// 快速傅立叶逆变换

float wreal [N / 2], wimag [N / 2], treal, timag, ureal, uimag, arg;

int m, k, j, t, index1, index2;

bitrp (xreal, ximag, n);

// 计算1的前n / 2 个 n 次方根 Wj = wreal [j] + i \* wimag [j] , j = 0, 1, ... , n / 2 - 1

arg = 2 \* PI / n;

treal = cos (arg);

timag = sin (arg);

wreal [0] = 1.0;

wimag [0] = 0.0;

for (j = 1; j < n / 2; j ++)

{

wreal [j] = wreal [j - 1] \* treal - wimag [j - 1] \* timag;

wimag [j] = wreal [j - 1] \* timag + wimag [j - 1] \* treal;

}

for (m = 2; m <= n; m \*= 2)

{

for (k = 0; k < n; k += m)

{

for (j = 0; j < m / 2; j ++)

{

index1 = k + j;

index2 = index1 + m / 2;

t = n \* j / m; // 旋转因子w的实部在wreal []中的下标为t

treal = wreal [t] \* xreal [index2] - wimag [t] \* ximag [index2];

timag = wreal [t] \* ximag [index2] + wimag [t] \* xreal [index2];

ureal = xreal [index1];

uimag = ximag [index1];

xreal [index1] = ureal + treal;

ximag [index1] = uimag + timag;

xreal [index2] = ureal - treal;

ximag [index2] = uimag - timag;

}

}

}

for (j=0; j < n; j ++)

{

xreal [j] /= n;

ximag [j] /= n;

}

}

void FFT\_test ()

{

char inputfile [] = "input.txt"; // 从文件 input.txt 中读入原始数据

char outputfile [] = "output.txt"; // 将结果输出到文件output.txt 中

float xreal [N] = {}, ximag [N] = {};

int n, i;

FILE \*input, \*output;

if (!(input = fopen (inputfile, "r")))

{

printf ("Cannot open file. ");

exit (1);

}

if (!(output = fopen (outputfile, "w")))

{

printf ("Cannot open file. ");

exit (1);

}

i = 0;

while ((fscanf (input, "%f%f", xreal + i, ximag + i)) != EOF)

{

i ++;

}

n = i; //要求n为2的整数幂

while (i > 1)

{

if (i % 2)

{

fprintf (output, "%d is not a power of 2! ", n);

exit (1);

}

i /= 2;

}

FFT (xreal, ximag, n);

fprintf (output, "FFT: i real imag ");

for (i = 0; i < n; i ++)

{

fprintf (output, "%4d %8.4f %8.4f ", i, xreal [i], ximag [i]);

}

fprintf (output, "================================= ");

IFFT (xreal, ximag, n);

fprintf (output, "IFFT: i real imag ");

for (i = 0; i < n; i ++)

{

fprintf (output, "%4d %8.4f %8.4f ", i, xreal [i], ximag [i]);

}

if ( fclose (input))

{

printf ("File close error. ");

exit (1);

}

if ( fclose (output))

{

printf ("File close error. ");

exit (1);

}

}

int main ()

{

FFT\_test ();

return 0;

}

double gaussrand()                   //生成均值为0，方差为1的高斯白噪声  
{  
static double V1, V2, S;  
static int phase = 0;  
double X;  
if (phase == 0) {  
do {  
double U1 = (double)rand() / RAND\_MAX;  
double U2 = (double)rand() / RAND\_MAX;  
V1 = 2 \* U1 - 1;  
V2 = 2 \* U2 - 1;  
S = V1 \* V1 + V2 \* V2;  
} while (S >= 1 || S == 0);  
X = V1 \* sqrt(-2 \* log(S) / S);  
}  
else  
X = V2 \* sqrt(-2 \* log(S) / S);  
phase = 1 - phase;  
return X;  
}  
const int N = 12365;     //测试数据时选取的点数  
//粘贴读取代码读到x[];  
double Xk[N] = fft(x, N);//原数据  
double sigma = 0.1;  
double v[N];//零均值平稳高斯白噪声  
for (int i = 0; i < N; i++)  
v[i] = gaussrand();  
double y[];//污染信号  
for (int i = 0; i < N; i++)  
y[i] = x[i] + sigma\*v[i];  
double Yk = fft(y, N);  
double Ry;  
xcorr(y,Rz);  
double Gy = fft(Ry, N);  
double Rsz = xcorr(z, y);  
double Gsz = fft(Rsz, N);  
double H[N];  
for (int i = 0; i < N; i++)  
H[i] = Gsz[i] / Gz[i];              //维纳滤波器的传递函数  
double S[N];  
for (int i = 0; i < N; i++)  
S[i]=H[i]\*Py[i];  
double ss = real(ifft(S));

struct {  
 char RIFF[4];    //头部分RIFF  
 long int size0;//存的是后面所有文件大小  
 char WAVE[4];  
 char FMT[4];  
 long int size1;//存的是保存的大小，包含这之后，data前面几个，共16个  
 short int ;  
 short int channel;  
 long int samplespersec;//每s采样数  
 long int bytepersec;  
 short int blockalign;  
 short int bitpersamples;  
 char DATA[4];  
 long int size2;//声音采样的大小  
};

WavHead head = { { 'R','I','F','F' },0,{ 'W','A','V','E' },{ 'f','m','t',' ' },16,  
1,1,N,N,1,8,{ 'd','a','t','a' },  
0 };  
if (SD.exists("2.wav")) {  //检查example.txt文件是否存在  
Serial.println("2.wav exists.");  //如果存在输出信息example.txt exists.至串口  
}  
else {  
Serial.println("2.wav doesn't exist.");  //不存在输出信息example.txt doesn't exist.至串口  
}  
myFile = SD.open("2.wav", FILE\_WRITE);

myFile.println(head);  
myFile.println(ss);  
myFile.close();  //关闭文件

### 13.2.3 输出模块

#### 2.相关代码

#include "SD.h"  //读取SD卡的库   
#include <SimpleSDAudio.h> //播放音频文件的库  
#include "SPI.h"  //SD卡的SPI库  
#define SD\_ChipSelectPin 4  //在4号引脚选择SD卡

#include “FFT.h” //FFT库

#include “Radom.h” //互相关函数计算，自相关函数计算相关库

void setup()

{

Serial.begin(9600);

while (!Serial) {

}

Serial.print(F("\nInitializing SD card...")); //串口提示信息

if (!SdPlay.init(SSDA\_MODE\_FULLRATE | SSDA\_MODE\_MONO)) {

Serial.println(F("initialization failed. ")); //读取失败

Serial.println(SdPlay.getLastError());

while(1);

} else {

Serial.println(F("Wiring is correct and a card is present."));//成功读取

}

pinMode(2, INPUT\_PULLUP); //设置数字读取引脚，以便后续设置按钮

}

int song\_number=0; //歌曲代号，改变播放更多曲目

boolean debounce=true;

void loop(void) {

if (digitalRead(2)==LOW && debounce == true){ //按键被按下

song\_number++;

if (song\_number==3) //歌曲最大循环数目

{song\_number=1;}

if (song\_number ==1){ //播放歌曲一

delay(3000);

Serial.print(F("Looking for 1.AFM... "));

if(!SdPlay.setFile("2.AFM")) {

Serial.println(F(" not found on card! Error code: "));

Serial.println(SdPlay.getLastError());

while(1);

} else {

Serial.println(F("found."));

}

Serial.print(F("Playing... "));

SdPlay.play();

while(!SdPlay.isStopped()) {

SdPlay.worker();

}

Serial.println(F("done."));}

if (song\_number ==2){ //播放歌曲二

delay(3000);

Serial.print(F("Looking for 2.AFM... "));

if(!SdPlay.setFile("1.AFM")) {

Serial.println(F(" not found on card! Error code: "));

continue; //如不存在则继续循环

Serial.println(SdPlay.getLastError());

while(1);

} else {

Serial.println(F("found."));

}

Serial.print(F("Playing... "));

SdPlay.play();

while(!SdPlay.isStopped()) {

SdPlay.worker();

}

Serial.println(F("done."));}

}

if (digitalRead(2)==HIGH) //避免抖动

{debounce = true; }

If(SD.exists(“2.wav”))

{ double \*x[N]=getwavf(“1.wav”);

complex Xk[N] = fft(x, N); //原数据  
double sigma = 0.1;  
double v[N];   
for (int i = 0; i < N; i++)  
v[i] = gaussrand();  
double y[N]; //污染信号  
for (int i = 0; i < N; i++)  
y[i] = x[i] + sigma\*v[i];  
complex Yk = fft(y, N);  
double Ry;  
xcorr(y,Rz);  
complex Gy = fft(Ry, N);  
complex Rsz = xcorr(z, y);  
complex Gsz = fft(Rsz, N);  
complex H[N];  
for (int i = 0; i < N; i++)  
H[i] = Gsz[i] / Gz[i];              //维纳滤波器的传递函数  
complex S[N];  
for (int i = 0; i < N; i++)  
S[i]=H[i]\*Py[i];  
double ss = real(ifft(S));

givewavf(“2.wav”,ss);

}

}